Программные системы: теория и приложения

Двуязычный электронный научный журнал

Bilingual Online Scientific Journal

Program Systems: Theory and Applications

Том **15** Выпуск 2(61) 2024 г.

Содержание

Click the flag at a top corner of any page to switch the language, please!

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ПРОГРАММНОЕ И АППАРАТНОЕ ОБЕСПЕЧЕНИЕ РАСПРЕДЕЛЕННЫХ И СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМ

УДК 004.724.2 + 004.272.43
10.25209/2079-3316-2024-15-2-3-19

Распределенная арифметика в оптическом канале на основе фотонных коммутаторов

Виктор Сергеевич **Подлазов**✉
Институт проблем управления имени В. А. Трапезникова РАН, Москва, Россия

Аннотация. В статье рассматривается фотонная сеть с распределенным управлением, состоящая из нескольких узлов, связанных общим каналом, в котором за время передачи одного числа выполняется единая операция над числами, которые параллельно передаются всеми узлами. Рассматриваются такие операции как суммирование или нахождение максимума (минимума) чисел, передаваемых последовательно по двоичным разрядам. Предполагается, что разряды чисел передаются парафазными оптическими сигналами, а общий канал строится из фотонных коммутаторов этих сигналов.

Ключевые слова и фразы: общий канал, распределенное управление, фотонный коммутатор, фотонные мультиплексор и демультиплексор, вычисление в общем канале, последовательные числа

Для цитирования: Подлазов В. С. *Распределенная арифметика в оптическом канале на основе фотонных коммутаторов* // Программные системы: теория и приложения. 2024. **Т. 15**. № 2(61). С. 3–19. `https://psta.psiras.ru/read/psta2024_2_3-19.pdf`

Введение

Развитие компьютерной элементной базы привело к появлению проектов фотонных компьютеров и фотонных элементов для них [1, 2]. Продолжением этой тенденции стало развитие проектов систем на основе локальных сетей, в которых данные передаются вместе с командами, образую spatial data flow структуры [3]. Их частным случаем являются структуры с распределенной параллельной обработкой данных во время и в процессе их передачи по сети – *вычисления в общем канале* (ВОК) [4].

При ВОК выполняется групповая операция над множеством чисел, передаваемых узлами параллельно, за время передачи по общему каналу сети одного числа (результата операции). При этом эта операция выполняется без промежуточных буферизаций промежуточных значений результата. Последнее свойство обеспечивает превосходство ВОК по быстродействию над традиционным сетевыми реализациям распределенных вычислений, которые требуют последовательных передач нескольких чисел и последовательного выполнения нескольких операций над ними.

Операция ВОК выполняется на аппаратном уровне за время пересылки по каналу сети только одного числа – формирующегося результата операции. Эта операция выполняется без тактовых задержек в функциональных блоках при узлах сети, состоящих из нескольких фотонных коммутаторов выбранного вида.

Традиционное выполнение распределенных операций складывается из нескольких последовательных передач чисел между узлами сети и арифметических операций над ними. Конкретно: распределенная операция над N числами может быть выполнена одним из двух крайних способов. Во-первых, это последовательная пересылка $(N-1)$ числа с промежуточными $(N-1)$ операцией над ними. Во-вторых, это парные передачи чисел, выполняемые параллельно за $\log_2 N$ последовательных их передач и за такое же количество операций в узлах. При этом указанные действия выполняются программным образом, что многократно увеличивает время их исполнения.

Ниже рассматривается реализация операций ВОК в оптическом канале с фотонными коммутаторами для использования таких широко известных его преимуществ как малое энергопотребление, высокая помехозащищенность и более высокие частоты передачи, связанные с независимостью параметров оптических сигналов от свойств окружающей среды.

Далее рассматриваются реализации операций ВОК на вынесенных из узлов функциональных блоках, составленных из выбранных фотонных

коммутаторов, и способ выполнения операций в них, который не использует обратной связи с узлами в процессе выполнения операций.

Для выполнения операций ВОК значение каждой логической переменной передается по каналу в парафазном виде, т.е. по двум линиям 0 и 1. Значение 0 передается сигналом в линии 0 при отсутствии сигнала в линии 1, а значение 1 – сигналом в линии 1 при отсутствии сигнала в линии 0. Линии 0 и 1 проходят через конвертер C_i каждого i-го узла сети (рисунок 1). Каждый конвертер выполняет унарную операцию O_i преобразования входной переменной x_i в выходную логическую переменную $y_i = O_i x_i$.

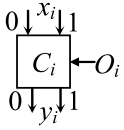
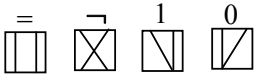


Рисунок 1. Конвертор фотонных переменных

Операция O_i выполняет следующие преобразования: повторения $y_i = x_i$ («=»), отрицания $y_i = \neg x_i$ («¬»), перевода в значение $y_i = 0$ («0») и перевода в значение $y_i = 1$ («1»). Состояния конвертера C_i при выполнении операции O_i представлены на рисунке 2.



O_i: {«=», «¬», «1», «0»}

Рисунок 2. Состояния операции O_i конвертера Ci

Операция O_i в свою очередь задается по значению управляющей логической переменной u_i, выдаваемой i-м узлом сети. В результате конвертер C_i выполняет бинарную операцию $y_i = x_i \aleph u_i$, которая может быть любой из 16 бинарных логических операций. В таблицах 1a– 2b приводятся таблицы истинности ряда используемых логических операций.

Таблица 1. Таблицы истинности операций $\vee$ и $\wedge$

(a) функция «И» ($\aleph = \wedge$)

x	u	O	y
0	0	«0»	0
0	1	«=»	0
1	0	«0»	0
1	1	«=»	1

(b) функция «ИЛИ» ($\aleph = \vee$)

x	u	O	y
0	0	«=»	0
0	1	«1»	1
1	0	«=»	1
1	1	«1»	1

Таблица 2. Таблицы истинности операций $\oplus$ и ∇

(a) функция «Сложение по mod2» ($\aleph = \oplus$)

x	u	O	y
0	0	«=»	0
0	1	«¬»	1
1	0	«=»	1
1	1	«¬»	0

(b) функция «Начальный запуск» ($\aleph = \nabla$)

x	u	O	y
0	0	«=»	0
0	1	«0»	0
1	0	«=»	1
1	1	«0»	0

При линейном размещение N узлов сети и последовательном размещении их конвертеров можно за время передачи по общему каналу одного числа образовать в нем сумму размещенных по узлам чисел или выделить из них максимальное число [4]. Для этого числа передаются как двоичные числа последовательно по разрядам. Для суммирования они передаются младшими разрядами вперед и используются операции $\oplus$ и $\wedge$ для образования двоичной суммы и значения переноса в следующий разряд соответственно. Для нахождения максимального числа все числа передаются старшими разрядами вперед, и в ней используется операция $\vee$.

Представленная схема позволяет выполнять арифметические и логические операции над значениями, представленными фотонными сигналами, на базе коммутаторов без использования специальных фотонных арифметико-логических устройств типа [1]. Саму сеть можно образовать из двух ветвей (восходящей и нисходящей) с подключением каждого узла к обеим ветвям. Восходящая ветвь используется для образования результата, а нисходящая ветвь – для оповещения всех о нем. Можно также использовать кольцевую сегментированную сеть с двукратной передачей чисел – для образования результата и оповещения о нем.

Сеть для ВОК можно построить из оптоэлектронных переключателей на базе интерферометров Маха-Зандера [5]. Их более практичная форма представлена в [6]. Из таких переключателей можно собрать оптоэлектронный коммутатор 4×4, в котором передача данных через коммутатор ведется оптическими сигналами, а управление коммутатором осуществляется узлом сети электронным образом [7]. В настоящее время быстродействие таких переключателей доходит до 100 Гбит/сек [8] и они имеют достаточно высокую плотность упаковки [9].

Однако построение ВОК на оптоэлектронных переключателях имеет своим недостатком необходимость преобразования значений разрядов между оптической и электронной формами в процессе выработки управляющих воздействий в каждом разряде на основе результата из предыдущего разряда. Этот недостаток, во-первых: снижает быстродействие фотонных коммутаторов до быстродействия электронных коммутаторов и, во-вторых, увеличивает их энергопотребление.

В статье рассматривается возможность выполнения операций ВОК на базе разных фотонных элементов без использования внешних оптоэлектронных преобразований. В качестве базовых фотонных элементов можно использовать фотонные демультиплексор $D\,1\times 4$ и мультиплексор $M\,4\times 1$, собираемые из нескольких слоев тонких пленок с электрооптическими или магнитооптическими свойствами (рисунок 3) [**10**, **11**].

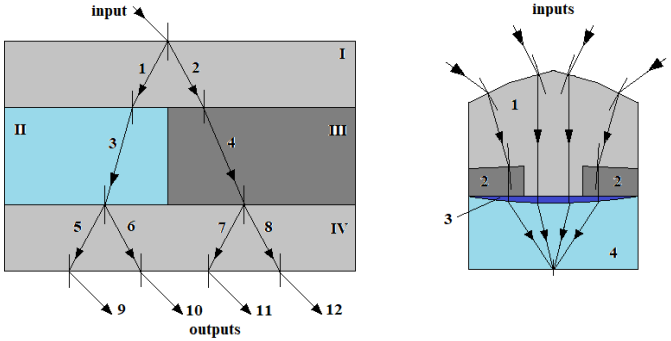


Рисунок 3. Структура коммутационных ячеек (демультиплексор и мультиплексор), основанных на оптически прозрачных метаматериальных и ферритовых пленках

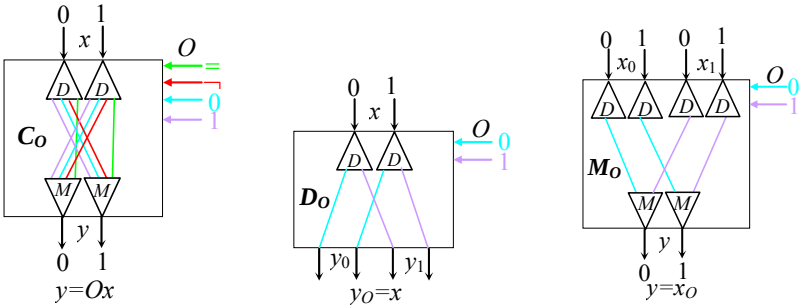
Особенность их применения состоит в том, что для управления коммутацией используется два оптических сигнала на разных частотах λ_1 и λ_2. В демультиплексоре они преобразуются в 4 управляющих сигнала электронной или магнитной природы и подаются на два направляющих слоя, каждый из которых задает два направления распространения информационного сигнала, поступившего на его оптический вход. В результате входной оптический сигнал направляется на один из 4-х оптических выходов.

Заметим, что эти элементы в совокупности с одноразрядными фотонными линиями задержки были использованы для построения фотонных неблокируемых системных сетей [**12**, **13**].

1. Набор необходимых фотонных элементов

Управляемый фотонный конвертер C_0 собирается из двух демультиплексоров D и двух мультиплексоров M (рисунок 4*а*). В каждом демультиплексоре используется 4 управляющих сигнала, определяющих для конвертера значение управляющей операции O и задающих выходы демультиплексора. Например, значения «0» и «1» подаются на первый направляющий слой, а значения «$=$» и «$\neg$» – на второй слой.

В дальнейшем наряду с конвертером C_0 используются управляемые демультиплексор D_0 (рисунок 4*б*) и мультиплексор M_0 (рисунок 4*в*), построенные аналогичным образом.



(*а*) C_0 (*б*) D_0 (*в*) M_0

Рисунок 4. Схемы управляемых фотонного конвертера C_0, демультиплексора D_0 и мультиплексора M_0

Образование значений операции O из значений управляющей фотонной переменной u задается схемами на рисунке 5. Эти схемы можно заменить

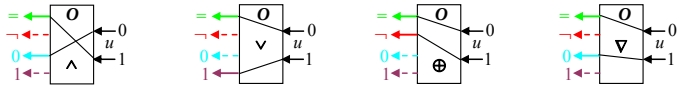


Рисунок 5. Схема образования значений операции O

на одну универсальную с использованием коммутатора с электронным управлением [7] от узла сети. Однако для простоты в данной работе используются разные неуправляемые схемы реализации операции O.

Как результат соединения схем рисунка 4*а* и схемы рисунка 5 управляемый конвертер будет обозначаться как 2-входовая схема с передаваемой переменной x, управляющей переменной u и реализуемой логической функцией $C(u, \aleph)$ (рисунок 6). При этом переменная x обозначается стрел-

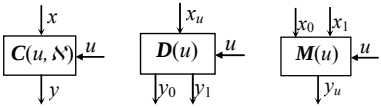


Рисунок 6. Окончательное обозначение управляемых конвертера $C(u, \aleph)$, демультиплексора $D(u)$ и мультиплексора $M(u)$

кой «ласточкин хвост», а переменная u – прямой стрелкой. Аналогично, управляемые демультиплексор и мультиплексор, полученные объединением схем рисунка 4, будут обозначаться как $D(u)$ и $M(u)$ (рисунок 6).

2. Распределенный фотонный сумматор

Для выполнения фотонной операции ВОК «сумма» при каждом узле сети формируется свой исполнительный блок U, состоящий из ранее определенных конвертеров и мультиплексоров, который осуществляет преобразования значений разрядов передаваемых чисел.

Первый узел сети выдает по парафазной линии значений в свой блок U_2 слот X_1, содержащий двоичное многоразрядное число, начиная с младших разрядов, а по парафазной синхролинии – слот S той же разрядности. Передача слота S начинается с передачи его метки s^0, которая задается одновременной передачей значений 0 и 1, т.е. передачей сигналов по обеим парафазным линиям. Прохождение метки s^0 запускает передачу слотов X_1 и S, которые передаются синхронно по разрядам. В слоте X_1 разряды передается переменными x_1^i, а в слоте S –переменными s^i, где i – это номер разряда. Первый разряд слота S имеет значение 1 ($s^1 = 1$), а остальные его разряды – значения 0 ($s^i = 0$ при $i > 1$).

Любой другой узел сети ($1 < k \leqslant N$) передает по линии значений в свой блок U_k слот X_k, содержащий значения переменных x_k^i для каждого разряда. Кроме того, в этот блок поступает слот S, прошедший через исполнительный блок предыдущего узла. Прохождение метки s^0 запускает в узле k передачу слота X_k, что поразрядно синхронизирует его в блоке U_k со слотом S.

На рисунке 7 представлена схема блока U_2, подсоединяемого к первому и второму сетевым узлам распределенного сумматора. Точки на линиях

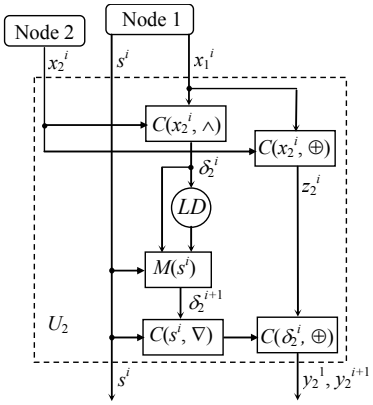


Рисунок 7. 2-узловой распределенный сумматор

обозначают узлы разделения световых сигналов посредством ветвления оптических линий. Проблема ослабления оптических сигналов может быть решена посредством использования фотонных усилителей на выходе блоков U_k [**14**].

На его входы U_2 подаются слот S и слоты X_1 и X_2, а на выходе образуется слот Y_2, содержащий сумму чисел из слотов X_1 и X_2. В этом блоке на конвертере $C(x_2^i, \oplus)$ складываются по mod2 значения переменных x_1^i и x_2^i, образуя промежуточный результат в виде переменной z_2^i. На конвертере $C(x_2^i, \nabla)$ из переменных x_1^i и x_2^i образуется переменная δ_2^i, которая после прохождения по линии задержки LD длительностью

в один разряд, образует переменную δ_2^{i+1}, задающую значение переноса в следующий разряд. Через мультиплексор $M(s^i)$ пропускается значение δ_2^1 в первом разряде и δ_2^{i+1} в любом другом разряде. Через конвертер $C(s^i, \nabla)$ проходят переменные δ_2^1 и δ_2^{i+1}. Конвертор обнуляет первую из них и пропускает вторую без изменения (таблица 2b) для остальных разрядов. Конвертор $C(\delta_2^i, \oplus)$ образует окончательный результат поразрядных вычислений в виде переменных y_2^i, составляющих слот Y_2.

На рисунке 8 (верхняя половина) представлена временная диаграмма

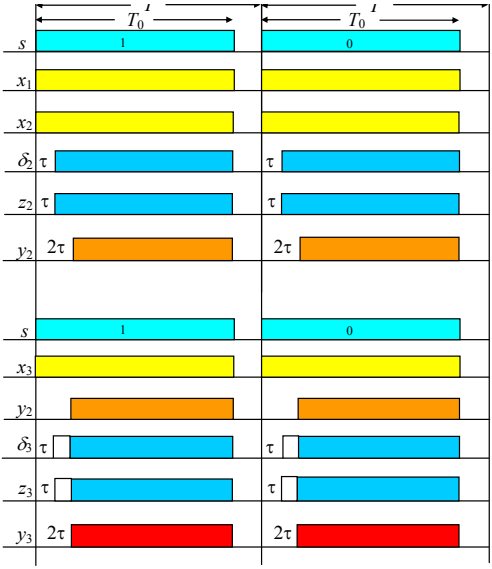


Рисунок 8. Временная диаграмма распределенного сумматора

работы блока U_2 в первых двух разрядах с использованием следующих обозначений.

Период разряда задается как T, длительность активных сигналов в нем – как T_0, а время настройки конвертера или мультиплексора по управляющей переменной u – как τ. Необходимо отметить, что переменная δ_2^i проходит устройства $M(s^i)$ и $C(s^i, \nabla)$ без задержки, так как они уже настроились к моменту ее появления. Как результат выходная переменная y_2^i во всех разрядах появляется с задержкой на 2τ и задается сигналами длительности $T_0 - 2\tau$.

На рисунке 9 представлена схема N-узлового распределенного фотонного сумматора. На входы блока U_k $(k > 2)$ подаются слот S и слоты Y_{k-1} и X_k, а на выходе образуется слот Y_k. Работ всех блоков синхронизируется слотом S, и все операции в блоках выполняются параллельно относительно 1-го разряда слота S.

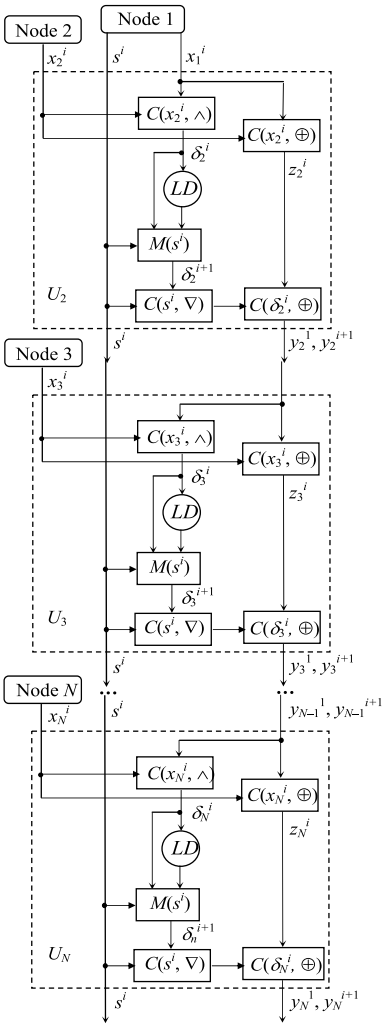


Рисунок 9. N-узловой распределенный сумматор

На рисунке 8 представлена также (нижняя половина) временная диаграмма работы блоков U_2 и U_3 сумматора в первых двух разрядах и в тех же обозначениях. Следует отметить, что нижняя половина диаграммы справедлива для любого блока U_k при $k > 2$. Это означает, что выходная переменная y_k^i появляется с задержкой на 2τ относительно начала синхросигналов s_i при любых k и i, т.е. при любом числе узлов и во всех разрядах чисел и задается сигналами длительности $T_0 - 2\tau$.

3. Распределенный фотонный «максимизатор»

Для выполнения фотонной операции ВОК «максимум» при каждом узле сети формируется свой исполнительный блок U, который осуществляет преобразования значений разрядов передаваемых чисел.

Первый узел сети по парафазной линии значений выдает в свой блок U слот X_1, содержащий двоичное многоразрядное число, начиная со старших разрядов, а по парафазной синхролинии – слот S той же разрядности. Любой другой узел k сети ($1 < k \leqslant N$) передает по линии значений в свой блок U_k слот X_k, содержащий переменные x_k^i, начиная со старшего разряда. Кроме того, в этот блок поступает слот S, прошедший через исполнительный блок предыдущего узла.

На рисунке 10 представлена схема блока U_2, подсоединяемого к первому и второму сетевого узлам распределенного «максимизатора». На его входы подаются слот S и слоты X_1 и X_2, а на выходе образуется слот Y_2, содержащий максимальное число из слотов X_1 и X_2.

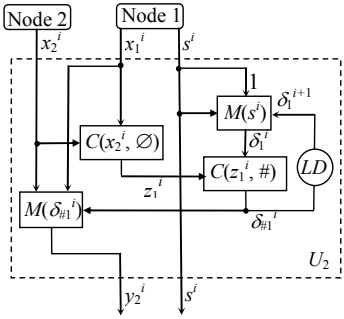


Рисунок 10. 2-узловой распределенный «максимизатор»

По значениям синхросигнала s^i посредством мультиплексора $M(s^i)$ формируется начальное значение переменной $\delta_1^1 = 1$, которая играет роль переменной состояния «максимизатора» во всех разрядах $i \geqslant 1$. Переменная δ_1^i должна находиться в значении $\delta_1^i = 1$ для всех разрядов $i < j$, для которых $x_1^i \geqslant x_2^i$, и находиться в значении $\delta_1^i = 0$ для всех разрядов $i > j$ после разряда j, для которого $x_1^j < x_2^j$. Такую последовательность значений δ_1 формируют конвертеры $C(x_2^i, \varnothing)$ и $C(z_1^i, \#)$ совместно с мультиплексором $M(s^i)$ и линией задержки LD на один разряд. Первый конвертер по значениям переменных x_1^i и x_2^i формирует значение промежуточной переменной z_1^i, реализуя логическую функцию «Переключение» по таблице 3a при $k \geqslant 2$. Второй конвертер формирует значение переменной $\delta_{\#1}^i$, реализуя логическую функцию «Повторение» по таблице 3b при $k \geqslant 2$.

Мультиплексор $M(s^i)$ и линия задержки LD образуют значения переменной $\delta_1^{i+1} = \delta_{\#1}^i$ для следующего разряда. При этом схема об-

Таблица 3. Таблицы истинности для блока U_k $(k \geqslant 2)$

(a) функция «Переключение» $(\aleph = \varnothing)$

x_{k-1}	x_k	O	z_{k-1}
0	0	«0»	1
0	1	«=»	0
1	0	«0»	1
1	1	«=»	1

(b) функция «Повторение» $(\aleph = \#)$

z_{k-1}	δ_{k-1}	O	$\delta_{\#(k-1)}$
0	0	«=»	0
0	1	«1»	0
1	0	«=»	0
1	1	«1»	1

разования значений операции O (см. Введение) задается рисунком 11 (аналогом рисунка 5). Для тех разрядов $i < j$, для которых сохраняется

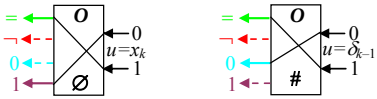


Рисунок 11. Схема образования значений операции O

значение $\delta 1 = 1$, через мультиплексор $M(\delta^i_{\#1})$ формируется значение выходной переменной $y^i_2 = x^i_1$. Однако, для тех разрядов $i > j$, для которых сохраняется значение $\delta 1 = 0$, через мультиплексор $M(\delta_{\#1})$ формируется значение выходной переменной $y^i_2 = x^i_2$. При этом для разряда j формируется значение $y^j_2 = x^j_2 = 1$.

На рисунке 12 приводится временная диаграмма формирования разных переменных для разрядов j и $j+1$, где τ обозначает время настройки конвертеров и мультиплексора, а T и T_0 – длительности разряда и активного сигнала в нем. Видно, что активный сигнал укорачивается на время не большее 3τ, которое не зависит от значения j.

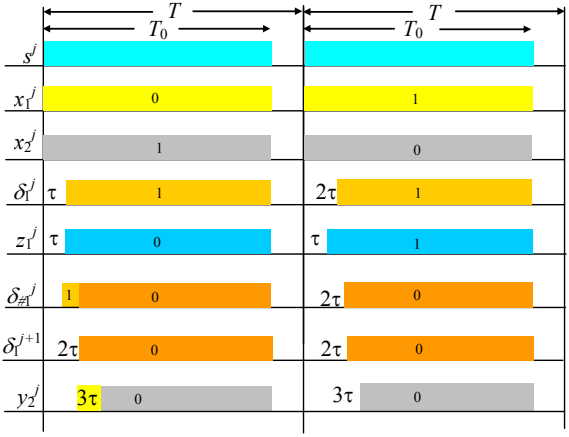


Рисунок 12. Временная диаграмма 2-узлового «максимизатора» для j-го разряда

На рисунке 13 приводится схема N-узлового распределенного «максимизатора». Каждый блок U_k $(k > 2)$ повторяет блок U_2 с заменой

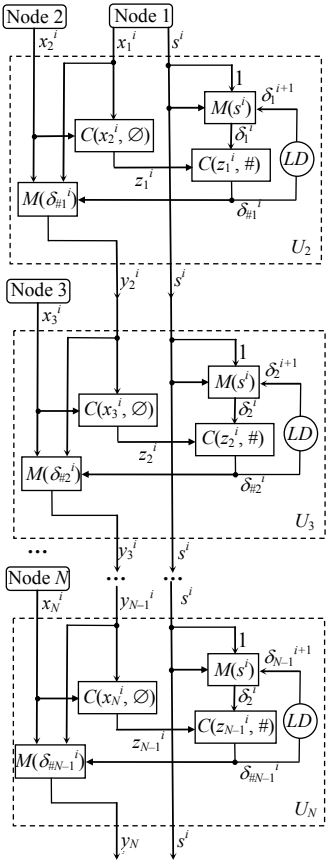


Рисунок 13. N-узловой распределенный «максимизатор»

x_1 на y_{k-1} и заменой индекса 2 на индекс k. При этом длительность активного сигнала $T_0 - 3\tau$ не зависит от значения k.

Заключение

В статье показана принципиальная реализуемость распределенных операций ВОК «максимум» и «сумма» на основе фотонных коммутаторов с парафазным представлением значений логических переменных. Также на основе этих операций можно осуществлять в сети [4] сортировку массива чисел за время передачи N чисел; умножение чисел за время передачи одного числа; скалярное произведение N чисел за время передачи двух чисел и ряд других операций. Все они являются операциями

магистрального (трубопроводного) типа.

В неблокируемых сетях с прямыми каналами [15, 16] можно организовывать множество таких операций, дополнительно размещая в слотах данных адреса узлов и коды операций, реализуемые в них. Способы построения таких сетей и способы реализации в них множества магистральных операций это тема отдельной работы.

Список использованных источников

[1] Stepanenko S. *Structure and Implementation Principles of a Photonic Computer* // EPJ Web of Conferences.– 2019.– Vol. **224**.– id. 04002.– 7 pp. ↑4, 6

[2] Степаненко С. А. *Оптический логический элемент* (*варианты*), Патент № 2 723 906 С1.– 18.06.2020.– ид. 06. ↑4

[3] Абрамов С. М., Степаненко С. А. *О подходах к разработке программного обеспечения для фотонной вычислительной машины*, Национальный Суперкомпьютерный Форум (НСКФ-2023) (Россия, Переславль-Залесский, ИПС имени А.К. Айламазяна РАН, 28 ноября–01 декабря 2023 года). ↑4

[4] Прангишвили И. В., Подлазов В. С., Стецюра Г. Г. *Локальные микропроцессорные вычислительные сети*, Глава 6.– М.: Наука.– 1984.– 175 с. ↑4, 6, 14

[5] Stanley A. I., Singh G., Eke J., Tsuda H. *Mach–Zehnder interferometer: A review of a perfect all optical switching structure* // *Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing*, eds. Afzalpulkar N., Srivastava V., Singh G., Bhatnagar D., New Delhi: Springer.– 2016.– ISBN 978-81-322-2638-3.– Pp. 415–425. ↑6

[6] Sacher W. D., Green W. M. J., Gill D. M., Assefa S., Barwicz T., Khater M., Kiewra Ed., Reinholm C., Shank S. M., Vlasov Yu. A., Poon J. K. S. *Binary phase-shift keying by coupling modulation of microrings* // Optics Express.– 2014.– Vol. **22**.– No. 17.– Pp. 20252–20259. ↑6

[7] Green W. M. J., Yang M., Assefa S., Van Campenhout J., Lee B. G., Jahnes C. V., Doany F. E., Schow C. L., Kash J. A., Vlasov Y. A. *Silicon electro-optic* 4×4 *non-blocking switch array for on-chip photonic networks* // *Los Angeles, California United States, 6–10 March 2011*, OSA/OFC/NFOEC 2011.– ISBN 978-1-55752-906-0.– id. OThM1. ↑6, 8

[8] Yen T.-H., Hung Y.-J. *Fabrication-insensitive CWDM (de)multiplexer based on cascaded Mach-Zehnder interferometers on silicon-on-insulator* // Journal of Lightwave Technology.– 2020.– Vol. **39**.– No. 1.– Pp. 146–153. ↑6

[9] Gui Y., Nouri B. M., Miscuglio M., Amin R., Wang H., Khurgin J. B., Dalir H., Sorger V. J. *100 GHz micrometer-compact broadband monolithic ITO Mach–Zehnder interferometer modulator enabling 3500 times higher packing density* // Nanophotonics.– 2022.– Vol. **11**.– No. 17.– Pp. 4001–4009. ↑6

[10] Vytovtov K., Barabanova E., Zouhdi S. *Optical switching cell based on metamaterials and ferrite films*, 12th International Congress on Artificial Materials for Novel Wave Phenomena (Metamaterials) (Espoo, Finland, August 2018–01 September 2018).– 2018.– Pp. 424–426. ↑7

[11] Barabanova E. A., Vytovtov K. A. *The control system elements of the new generation optical switching cell* // Journal of Physics: Conference Series.– 2019.– Vol. **1368**.– No. 2.– id. 022002.– 9 pp. ↑7

[12] Vytovtov K. A., Barabanova E. A., Podlazov V. S. *Model of next-generation optical switching system* // *Distributed Computer and Communication Networks*, DCCN 2018, Communications in Computer and Information Science.– vol. **919**, Cham: Springer.– ISBN 978-3-319-99447-5.– Pp. 377–386. ↑7

[13] Barabanova E. A., Vytovtov K. A., Vishnevskiy V. M., Podlazov V. S. *High-capacity strictly non-blocking optical switches based on new dual principle*, 5th International Scientific Conference on Information, Control, and Communication Technologies (ICCT-2021) (Astrakhan, Russian Federation, 4–7 October 2021) // Journal of Physics: Conference Series.– 2021.– Vol. **2091**.– id. 012040.– 17 pp. ↑7

[14] Roelkens G., Raz O., Green W. M. J., Assefa S., Tassaert M., Keyvaninia S., Vandoorne K., Van Thourhout D., Baets R., Vlasov Y. *Towards a low-power nanophotonic semiconductor amplifier heterogeneously integrated with SOI waveguides*, 7th IEEE International Conference on Group IV Photonics (Beijing, China, 01–03 September 2010).– Pp. 16–18. ↑9

[15] Подлазов В. С. *Самомаршрутизируемая неблокируемая системная сеть с прямыми каналами: сложность и быстродействие* // Программные системы: теория и приложения.– 2022.– Т. **13**.– № 4(55).– С. 47–76. URL ↑15

[16] Подлазов В. С. *Разные неблокируемые самомаршрутизируемые системные сети с прямыми каналами* // Программные системы: теория и приложения.– 2023.– Т. **14**.– № 3.– С. 115–138. URL ↑15

Поступила в редакцию 30.10.2023;
одобрена после рецензирования 15.01.2024;
принята к публикации 16.01.2024;
опубликована онлайн 05.04.2024.

Рекомендовал к публикации *к.ф.-м.н. С. А. Романенко*

Информация об авторе:

Виктор Сергеевич Подлазов

Д. т. н., гл.н.с. Института проблем управления им. В.А. Трапезникова РАН, научные интересы: архитектуры интерконнекта и маршрутизация в суперкомпьютерных системах

0000-0002-9175-1138

e-mail: *podlazov@ipu.ru*
podlazov@gmail.com

Декларация об отсутствии личной заинтересованности: *благополучие автора не зависит от результатов исследования.*

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article HARDWARE AND SOFTWARE FOR DISTRIBUTED AND SUPERCOMPUTER SYSTEMS

UDC 004.724.2 + 004.272.43
10.25209/2079-3316-2024-15-2-3-19

Distributed ALUs based on photonic switches

Viktor Sergeevich **Podlazov**
V. A. Trapeznikov Institute of Control Sciences of RAS, Moscow, Russia

Abstract. The article examines a photonic network with distributed control, consisting of several nodes connected by a common channel, in which, during the transmission of one number, a single operation is performed on numbers that are transmitted in parallel by all nodes. Such operations as summing or finding the maximum (minimum) of numbers transmitted sequentially across binary bits are considered. It is assumed that the bits of numbers are transmitted by paraphase optical signals, and the common channel is built from photonic switches of these signals. (*In Russian*).

Key words and phrases: common channel, distributed control, photonic switch, photonic multiplexer and demultiplexer, calculation in a common channel, sequential numbers

2020 *Mathematics Subject Classification*: 65Y05; 68Q10

For citation: Viktor S. Podlazov. *Distributed ALUs based on photonic switches*. Program Systems: Theory and Applications, 2024, **15**:2(61), pp. 3–19. (*In Russ.*). https://psta.psiras.ru/read/psta2024_2_3-19.pdf

References

[1] S. Stepanenko. "Structure and Implementation Principles of a Photonic Computer", *EPJ Web of Conferences*, **224** (2019), id. 04002, 7 pp.

[2] S. A. Stepanenko. *Optical logic element (options)*, Patent No 2 723 906 C1, 18.06.2020, id. 06 (In Russian).

[3] S. M. Abramov, S. A. Stepanenko. "On approaches to developing software for a photonic computer", Nacional'nyj Superkomp'yuternyj Forum (NSKF-2023) (Rossiya, Pereslavl'-Zalesskij, IPS imeni A.K. Ajlamazyana RAN, 28 noyabrya–01 dekabrya 2023 goda) (In Russian).

[4] I. V. Prangishvili, V. S. Podlazov, G. G. Stecyura. *Local microprocessor computer networks*, Glava 6, Nauka, M., 1984, 175 pp. (In Russian).

[5] A. I. Stanley, G. Singh, J. Eke, H. Tsuda. "Mach–Zehnder interferometer: A review of a perfect all optical switching structure", *Proceedings of the International Conference on Recent Cognizance in Wireless Communication & Image Processing*, eds. Afzalpulkar N., Srivastava V., Singh G., Bhatnagar D., Springer, New Delhi, 2016, ISBN 978-81-322-2638-3, pp. 415–425.

[6] W. D. Sacher, W. M. J. Green, D. M. Gill, S. Assefa, T. Barwicz, M. Khater, Ed. Kiewra, C. Reinholm, S. M. Shank, Yu. A. Vlasov, J. K. S. Poon. "Binary phase-shift keying by coupling modulation of microrings", *Optics Express*, **22**:17 (2014), pp. 20252–20259.

[7] W. M. J. Green, M. Yang, S. Assefa, Campenhout J. Van, B. G. Lee, C. V. Jahnes, F. E. Doany, C. L. Schow, J. A. Kash, Y. A. Vlasov. "Silicon electro-optic 4×4 non-blocking switch array for on-chip photonic networks", *Los Angeles, California United States, 6–10 March 2011*, OSA/OFC/NFOEC 2011, ISBN 978-1-55752-906-0, id. OThM1.

[8] T.-H. Yen, Y.-J. Hung. "Fabrication-insensitive CWDM (de)multiplexer based on cascaded Mach-Zehnder interferometers on silicon-on-insulator", *Journal of Lightwave Technology*, **39**:1 (2020), pp. 146–153.

[9] Y. Gui, B. M. Nouri, M. Miscuglio, R. Amin, H. Wang, J. B. Khurgin, H. Dalir, Sorger V. J. . "100 GHz micrometer-compact broadband monolithic ITO Mach–Zehnder interferometer modulator enabling 3500 times higher packing density", *Nanophotonics*, **11**:17 (2022), pp. 4001–4009.

[10] K. Vytovtov, E. Barabanova, S. Zouhdi. "Optical switching cell based on metamaterials and ferrite films", 12th International Congress on Artificial Materials for Novel Wave Phenomena (Metamaterials) (Espoo, Finland, August 2018–01 September 2018), 2018, pp. 424–426.

[11] E. A. Barabanova, K. A. Vytovtov. "The control system elements of the new generation optical switching cell", *Journal of Physics: Conference Series*, **1368**:2 (2019), id. 022002, 9 pp.

[12] K. A. Vytovtov, E. A. Barabanova, V. S. Podlazov. "Model of next-generation optical switching system", *Distributed Computer and Communication Networks*, DCCN 2018, Communications in Computer and Information Science, vol. **919**, Springer, Cham, ISBN 978-3-319-99447-5, pp. 377–386.

[13] E. A. Barabanova, K. A. Vytovtov, V. M. Vishnevskiy, V. S. Podlazov. "High-capacity strictly non-blocking optical switches based on new dual principle", 5th International Scientific Conference on Information, Control, and Communication Technologies (ICCT-2021) (Astrakhan, Russian Federation, 4–7 October 2021), *Journal of Physics: Conference Series*, **2091** (2021), id. 012040, 17 pp.

[14] G. Roelkens, O. Raz, W. M. J. Green, S. Assefa, M. Tassaert, S. Keyvaninia, K. Vandoorne, Thourhout D. Van, R. Baets, Y. Vlasov. "Towards a low-power nanophotonic semiconductor amplifier heterogeneously integrated with SOI waveguides", 7th IEEE International Conference on Group IV Photonics (Beijing, China, 01–03 September 2010), pp. 16–18.

[15] V. S. Podlazov. "Multichannel non-blocking system area network with direct channels", *Program Systems: Theory and Applications*, **13**:4(55) (2022), pp. 47–76 (In Russian).

[16] V. S. Podlazov. "Multichannel non-blocking system area network with direct channels", *Program Systems: Theory and Applications*, **14**:3 (2023), pp. 115–138 (In Russian).

RU ENG ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ, НЕЙРОННЫЕ СЕТИ

УДК 004.93:004.35
10.25209/2079-3316-2024-15-2-21-35

Жестовое управление полетом малого беспилотного летательного аппарата

Николай Сергеевич **Абрамов**[1], Вита Викторовна **Саттарова**[2], Виталий Петрович **Фраленко**[3]✉, Михаил Вячеславович **Хачумов**[4]

[1, 3, 4] Институт программных систем им. А. К. Айламазяна РАН, Веськово, Россия
[2] Российский университет дружбы народов, Москва, Россия
[4] Российский государственный гуманитарный университет, Москва, Россия
[4] Федеральный исследовательский центр «Информатика и управление» РАН, Москва, Россия
[4] МИРЭА - Российский технологический университет, Москва, Россия

Аннотация. Рассмотрена задача построения жестовых команд для управления малым беспилотным летательным аппаратом, таким как квадрокоптер. Получаемые видеокамерой команды идентифицируются классификатором на основе сверточной нейронной сети, а мультимодальный интерфейс управления, оснащенный интеллектуальным решателем, преобразует их в команды управления квадрокоптером. Нейронные сети из библиотеки моделей нейронных сетей Ultralytics позволяют выделять целевые объекты в кадре в режиме реального времени. Команды управления квадрокоптером поступают в специализированную программу на смартфоне, разработанную на базе симулятора полетов DJI SDK, которая посылает команды по радиоканалу дистанционного управления.

Исследовано качество распознавания разработанных жестовых команд для квадрокоптеров DJI Phantom 3 standard edition. Представлено краткое руководство в виде сценариев работы оператора с беспилотными транспортными средствами. Раскрыты перспективы жестового управления несколькими транспортными средствами в экстремальных условиях с учётом сложностей безопасности совместного полета и взаимодействия летательных аппаратов в ограниченном пространстве.

Ключевые слова и фразы: беспилотный летательный аппарат, управление, жесты, сверточная нейронная сеть, Ultralytics, интеллектуальный интерфейс, распознавание

Благодарности: Исследование выполнено за счет гранта Российского научного фонда № 21–71–10056, https://rscf.ru/project/21-71-10056/

Для цитирования: Абрамов Н. С., Саттарова В. В., Фраленко В. П., Хачумов М. В. *Жестовое управление полетом малого беспилотного летательного аппарата* // Программные системы: теория и приложения. 2024. **Т. 15**. № 2(61). С. 21–35. https://psta.psiras.ru/read/psta2024_2_21-35.pdf

Введение

В настоящее время большое количество исследований и разработок посвящено развитию беспилотных летательных аппаратов (БПЛА). Для повышения надежности и удобства взаимодействия с БПЛА в дополнение к типовым пультовым операциям управления применяют жестовые и речевые команды [1]. Для реализации такого подхода необходимы многомодальные человеко-машинные интерфейсы, адаптированные к различным типам БПЛА. Жесты подразделяются на статические и динамические и используются для передачи информации в компьютер с целью управления компьютером, летательным аппаратом, роботом и т.д.

В диссертационной работе [2] представлена комплексная методология захвата, отслеживания и распознавания динамических жестов в видеопотоке. В рамках этой методологии разработаны алгоритм захвата и отслеживания кисти человека в видеопотоке на сложном фоне; алгоритм и вычислительно-эффективная модель для распознавания жестов, основанная на нечетких конечных автоматах. Разработана методология мультимодального распознавания сцен, определяемых жестами, с использованием нечетких операторов агрегирования. Экспериментально показано, что предложенная архитектура системы распознавания динамических жестов позволяет с высокой степенью надежности распознавать в реальном времени жесты независимо от индивидуума.

В диссертационной работе [3] предложены метод извлечения изображения руки из дальностного изображения человека на основе анализа связанности точек изображения в трехмерном пространстве; метод распознавания позиции кончиков пальцев и точек соединения пальцев с ладонью руки на основе анализа контура изображения руки; метод скелетизации дальностного изображения, основанный на непрерывном скелетном представлении бинарного изображения, поиске граничных точек фигуры и создании диаграммы Вороного для этих точек; метод распознавания статических и динамических жестов рук и пальцев руки для жестовой азбуки глухонемых.

В исследовании [4] представлено решение проблемы восстановления и отслеживания трехмерного положения, ориентации и полной артикуляции человеческой руки по данным от Kinect-сенсора. Решается задача оптимизации, в которой осуществляется поиск параметров модели руки, которые минимизируют расхождение между внешним видом и трехмерной структурой данных от сенсора. 3D-трекинг движения рук выполняется в реальном времени.

В работе [5] представлена техника взаимодействия, позволяющая с помощью жестов рук управлять функциями камеры, такими как панорамирование, наклон и затвор. В основе предложенной техники — алгоритм Лукаса-Канаде.

В исследовании [6] выполнена оценка возможности нейронных сетей помогать в распознавании жестов рук в арабском жестовом языке, проведены эксперименты с нейронными сетями прямого распространения и рекуррентными нейронными сетями. Предложенная система с полностью рекуррентной архитектурой демонстрирует точность 95% при распознавании статических жестов.

В работе [7] представлен интерактивный интерфейс пользователя для распознавания жестов рук американского жестового языка с использованием буквосочетаний на пальцах. Жесты классифицируются с помощью метода случайных лесов (от англ. «random forest»). Классификатор жестов интегрирован с английским словарем для ускорения написания текстов.

В исследовании [8] представлен прототип мультимодальной системы, который интегрирует методы распознавания лица, жестов и речи для поддержки мультимодальной возможности взаимодействия человека с компьютером. Разработана многоуровневая система с несколькими камерами для наблюдения за лицом пользователя, жестами тела и пространственным расположением в комнате. Используя речевой ввод, система более точно воспринимает намерения пользователя.

В работе [9] описан механизм управления квадрокоптером с помощью жестов и поз. Этот механизм делает человеко-машинное взаимодействие более интуитивным, удобным и отзывчивым на потребности пользователя.

Управление жестовыми командами может осуществляться с использованием различных сенсорных устройств. Например, для квадрокоптера *Parrot AR.Drone 2.0*[URL] от компании Parrot в работе [10] была предложена бесконтактная система управления с использованием трехмерного сенсора Kinect. Однако, универсальным и распространенным видом современного устройства ввода информации является, несомненно, видеокамера. Перспективным шагом можно считать построение комбинированных интерфейсов, отличающихся большей надежностью. Например, такой интерфейс представлен в работе [11], где для распознавания жестов оператора БПЛА используется нейросетевая архитектура Yolo 5, а для обработки и распознавания речи — различные многослойные искусственные нейронные сети (ИНС).

Выделим несколько инструментальных средств распознавания образов.

Метод инвариантных моментов [12]. Здесь распознавание позиции и ориентации руки в бинарном изображении осуществляется посредством вычисления моментов изображения, при условии, что фон изображения однороден и рука является доминирующим объектом в изображении. Метод реализуется для 2D-и 3D-режимов и позволяет упростить процесс сравнения жеста с эталонами. Методы позволяют корректно, в пределах точности представления изображений, сравнивать два изображения. Инвариантный подход к распознаванию допускает точную математическую постановку задачи, позволяющую задавать классы объектов. Метод был практически применен в диссертационной работе [3].

Метод *DTW* (англ. Dynamic Time Warping, алгоритм динамической трансформации временной шкалы). Идентификация формы осуществляется путем сравнения «скелета» руки с эталонами. Под скелетом здесь понимается набор опорных точек ладони, идентифицирующих положение ладони и пальцев. Для оценки степени схожести предварительно производится развертка скелетов, где по оси абсцисс откладываются номера точек в строгом соответствии с номерами их обхода в скелете, а по оси ординат координаты точек. После развертки расстояние между объектами оценивается за полиномиальное время с помощью алгоритма динамической трансформации шкалы времени.

Известно, что рекуррентная ИНС с архитектурой LSTM (Long Short-Term Memory) [13] показывает хорошие результаты при распознавании жестов, пример ее использования приведен в исследовании [14], где полученное значение показателя «точность» («accuracy») не менее 0.90, а в среднем 0.93. В работе [15] было проведено обучение нейронной сети Mask-RCNN для распознавания жестов рук, проанализированы существующие способы распознавания жестов, исследованы преимущества и недостатки рассмотренных методов. Предложена собственная архитектура сверточной нейронной сети для решения задачи классификации жестов. Проведена оценка точности работы сети в зависимости от расстояния между камерой и рукой, а также в зависимости от сложности жеста.

Предлагаемый в настоящей работе интерфейс управления БПЛА обеспечивает повышенную надежность и гибкость за счет использования стандартного кнопочного управления в комбинации с голосовыми и жестовыми командами. При управлении со стационарного рабочего места управление оказывается максимально эффективным. Однако, беспилотники могут работать и в режиме передачи управления людям, находящимся в местах, где имеются пожары, задымления, высокий уровень шума. Именно в этих условиях важно получать надежно команды управления, по этой причине и вводится мультимодальность управления,

позволяющая продублировать команды, при этом с помощью ИНС определяется степень уверенности и выбирается наиболее вероятная команда.

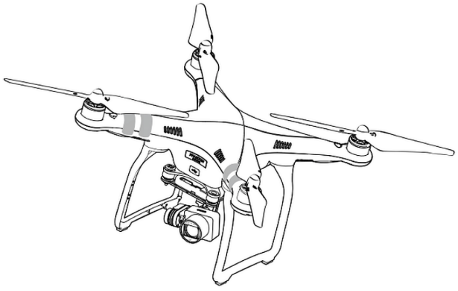
Управление БПЛА с помощью жестов включает несколько ключевых элементов:

(*1*) видеокамера, способная регистрировать жесты пользователя;
(*2*) алгоритмы распознавания, способные классифицировать и преобразовать жест в команду;
(*3*) команды управления, осуществляющие преобразование набора жестов в команды БПЛА;
(*4*) интеллектуальный интерфейс – программно-техническое средство, через которое пользователь взаимодействует с аппаратом.

Оценка эффективности управления квадрокоптером на основе системы жестовых команд

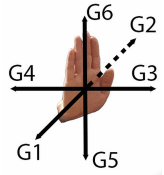
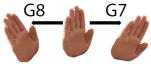
В настоящем исследовании исходным материалом служили базовые команды управления. Рассматривались движения по всем направлениям пространства, включая перемещение, поворот вокруг вертикальной оси, взлет и посадку. Для экспериментов с управлением жестами использовался квадрокоптер модели *DJI Phantom 3 Standard Edition*. Данный аппарат оснащен многофункциональной двухповоротной видеокамерой. Общий вид квадрокоптера и его основные характеристики представлены в таблице 1.

Таблица 1. Общий вид и основные характеристики задействованного БПЛА

Квадрокоптер DJI Phantom 3 Standard Edition	**Основные характеристики**
	(*а*) стабилизированный 3-осевой подвес с камерой; (*б*) встроенная камера: 2.7K видео и 12MP фото; (*в*) радиоканал WiFi 2.4G; (*г*) дальность связи 1 км; (*д*) интеллектуальный аккумулятор 15.2 V, 480 mAh.

Предлагаемая система команд управления БПЛА приведена в таблице 2.

Таблица 2. Система команд для управления БПЛА

Команда	**Назначение**	**Описание способа управления**	**Вид жеста**
Взлет	Взлет на высоту 2.3 метра.	Жестовая команда *«палец вверх»*.	
Приземление	Снижение и посадка в текущих координатах.	Жестовая команда *«палец вниз»*.	
Перемещение	Перемещение по трем осям пространства.	Перемещение ладони по трем осям в шести направлениях: G1 – вперед, G2 – назад, G3 – влево, G4 – вправо, G5 – вниз, G6 – вверх.	
Домой	Взлет на высоту 30 м. и возврат по прямой в координаты начальной точки, посадка.	Жестовая команда *«руки, сложенные в форме крыши»*, команда может выполняться автоматически в случае потери связи с БПЛА.	
Поворот	Поворот вокруг оси рыскания.	Наклон ладони по двум направлениям: G7 – поворот вокруг оси против часовой стрелки, G8 – поворот вокруг оси по часовой стрелке.	
Остановка	Прекращение движения (зависание в воздухе).	Жестовая команда *«поднятый кулак»*.	

Инструкции для БПЛА, сформированные наземной станцией после обработки жестовых команд оператора, поступают в специализированную программу на смартфоне, разработанную на базе DJI SDK, которая посылает команды через пульт и далее радиоканал.

Для оценки возможностей актуальной версии 8.1.47 программного пакета *Ultralytics*[URL] по локализации целевых объектов (жестов для управления БПЛА) проведен ряд экспериментов с поддерживаемыми этим пакетом конфигурациями ИНС: Yolo 5, Yolo 8, Yolo 8 Ghost, Yolo 8 Rtdetr, Yolo 9 и оригинальной архитектурой Rtdetr от Baidu [**16**]. Во всех случаях, кроме последнего и предпоследнего, использовались m-модификации ИНС, для Yolo 9 – c-модификация, а в последнем случае – l-модификация. Работа ИНС проверялась на собственном датасете с командами для управления БПЛА, содержащем жестовые команды «взлет», «приземление», «остановка» и «домой», применяемые при мультимодальном управлении БПЛА [**11**]; использовалась разметка с помощью *ориентированных*[URL] и *неориентированных*[URL] прямоугольных областей; всего 247 обучающих изображений и 122 тестовых. Тестирование выполнялось в режиме с половинной точностью (параметр *half*=True).

Повышение параметра *imgsz* (влияет на масштабирование поступающих изображений) с исходных 640 пикселей, заложенных по умолчанию в программный код библиотеки, до 1120 позволяет уверенно распознавать жесты, показываемые оператором на удалении. В процессе валидации результатов с пороговым значением $conf$ (уровень уверенности ИНС) отсеиваются те зоны, потенциально содержащие жест, для которых $conf \geqslant 0.01$. Среди оставшихся выбираются те, у которых значение $iou \geqslant 0.7$ (уровень пересечения с оригинальным жестом в зоне). Значения параметров подобраны экспериментально.

Для исследования качества распознавания на вход ИНС подавались изображения, измененные до размера 1120x640 пикселей. При тестировании использовалось следующее аппаратное обеспечение: процессор общего назначения Intel Core i7-6850k (6 ядер / 12 потоков, 4 ГГц), оперативная память 32 ГБ, видеокарта Nvidia GeForce RTX 3060.

Результаты решения задачи выделения управляющих жестов (для неориентированных прямоугольников) представлены в таблице 3. Основные показатели качества – mAP50-95 и значение F1-меры, достигаемой при указанном в таблице пороге уверенности.

Таблица 3. Результаты работы ИНС для локализации управляющих жестов (для неориентированных прямоугольников)

Вид ИНС	Команда	Точность	Полнота	mAP50-95	F1	Порог уверенности
Yolo 5	взлет	**1.000**	0.965	0.871		
	приземление	0.999	**1.000**	0.869		
	остановка	0.997	**1.000**	0.841		
	домой	0.997	**1.000**	0.802		
	все	0.998	0.991	0.846	0.99	0.723
Yolo 8	взлет	**1.000**	0.924	0.749		
	приземление	**1.000**	0.850	0.659		
	остановка	**1.000**	0.997	0.694		
	домой	0.928	**1.000**	0.591		
	все	0.982	0.943	0.673	0.96	0.583
Yolo 9	взлет	**1.000**	0.999	0.868		
	приземление	0.971	**1.000**	0.853		
	остановка	**1.000**	**1.000**	0.851		
	домой	**1.000**	**1.000**	0.854		
	все	0.993	**1.000**	0.856	**1.00**	0.242
Yolo 8 Ghost	взлет	**1.000**	**1.000**	0.812		
	приземление	0.996	**1.000**	0.861		
	остановка	**1.000**	**1.000**	0.829		
	домой	0.997	**1.000**	0.825		
	все	0.998	**1.000**	0.831	**1.00**	0.795
Yolo 8 Rtdetr	взлет	0.998	**1.000**	0.857		
	приземление	0.986	**1.000**	0.794		
	остановка	0.998	**1.000**	0.758		
	домой	0.997	**1.000**	0.864		
	все	0.995	**1.000**	0.818	**1.00**	0.861
Rtdetr (Baidu)	взлет	0.971	**1.000**	0.805		
	приземление	0.882	**1.000**	0.736		
	остановка	0.995	**1.000**	0.634		
	домой	0.954	**1.000**	0.756		
	все	0.950	**1.000**	0.733	0.97	0.616

В случае использования неориентированных прямоугольников архитектуры Yolo 8 Ghost, Yolo 8 Rtdetr и Yolo 9 обеспечили безошибочное выделение всех жестовых команд без ложных срабатываний, однако, максимальную точность локализации обеспечивает новая Yolo 9, она же

имеет самое высокое значение mAP50-95 = 0.856. Детальное исследование данных, получаемых от ИНС, показало, что иногда в кадре находится несколько объектов (не более двух, как показали эксперименты), однако за счет порога уверенности можно добиться полного исключения ложного обнаружения. В данном случае Yolo 9 обрабатывает кадр видеопотока за 19.5 мс.

Результаты решения задачи выделения управляющих жестов (для ориентированных прямоугольников) представлены в таблице 4. Вариант Yolo 8 Rtdetr исключен из рассмотрения ввиду того, что слой RTDETR-Decoder в этой архитектуре нельзя заменить на OBB-слой.

Таблица 4. Результаты работы ИНС для локализации управляющих жестов (для ориентированных прямоугольников)

Вид ИНС	**Команда**	**Точность**	**Полнота**	**mAP50-95**	**F1**	**Порог уверенности**
Yolo 5	взлет	**1.000**	**1.000**	0.914		
	приземление	**1.000**	**1.000**	0.978		
	остановка	0.996	**1.000**	0.965		
	домой	0.996	**1.000**	0.935		
	все	0.988	**1.000**	0.948	**1.00**	0.785
Yolo 8	взлет	0.994	**1.000**	0.937		
	приземление	**1.000**	**1.000**	0.977		
	остановка	0.998	**1.000**	0.951		
	домой	**1.000**	**1.000**	0.961		
	все	0.998	**1.000**	0.956	**1.00**	0.793
Yolo 9	взлет	**1.000**	**1.000**	0.918		
	приземление	**1.000**	**1.000**	0.977		
	остановка	**1.000**	**1.000**	0.960		
	домой	0.993	**1.000**	0.913		
	все	0.998	**1.000**	0.942	**1.00**	0.747
Yolo 8 Ghost	взлет	0.988	**1.000**	0.924		
	приземление	**1.000**	**1.000**	0.961		
	остановка	0.998	**1.000**	0.936		
	домой	0.996	**1.000**	0.926		
	все	0.996	**1.000**	0.937	**1.00**	0.796
Rtdetr (Baidu)	взлет	0.991	**1.000**	0.928		
	приземление	0.999	**1.000**	0.971		
	остановка	0.998	**1.000**	0.964		
	домой	**1.000**	**1.000**	0.949		
	все	0.997	**1.000**	0.953	**1.00**	0.745

Все варианты архитектур обеспечили 100%-ые полноту и F1-меру.

Однако, Yolo 8 обеспечила самое высокое значение показателя mAP50-95 = 0.956 (среднее для всех жестов) в сочетании с точностью 0.998. Если сравнивать эту ИНС с лучшей для неориентированных прямоугольников, переход к ориентированным обеспечил рост mAP50-95 на величину 0.1, что весьма существенно. Yolo 8 обрабатывает кадр видеопотока за 12.8 мс. Yolo 8 Ghost решает ту же задачу за 9.6 мс, то есть позволяет снизить временные затраты на 25%. Учитывая, что в данной задаче обе сети обеспечивают F1-меру 100%, приоритет следует отдать Yolo 8 Ghost.

Далее приведем краткий сценарий работы оператора с БПЛА:

(*1*) подготовить полетный план: задать координаты доступных для полета точек (широта, долгота, высота);

(*2*) активировать составляющие систему: БПЛА, пульт (для связи БПЛА с наземной станцией), мобильное приложение (клиент) и главную программу (сервер); для управления жестами к серверу должна быть подключена видеокамера;

(*3*) в мобильном приложении выбрать настройки: установить порт для связи с сервером и координаты «домашней точки», флаги передачи телеметрической информации, видеопотока с бортовой камеры БПЛА и использования режима симулятора компании-разработчика квадрокоптера; после этого установится связь между сервером и БПЛА, и главная программа будет готова принимать команды оператора;

(*4*) зафиксировать положение оператора в кадре видеокамеры сервера таким образом, чтобы в кадр попадали руки оператора;

(*5*) для управления жестами оператор показывает в видеокамеру один из них: «взлет», «приземление», «домой», «остановка»; при этом следует показывать жест несколько секунд: это сделано для того, чтобы исключить ложные срабатывания системы распознавания;

(*6*) посредством нажатия левой кнопки мыши по кнопкам интерфейса главной программы оператор может отдать команды БПЛА в соответствии с таблицей 2;

(*7*) посредством зажатия левой кнопки мыши на трехмерной карте местности полета в центральной части интерфейса, в которой отображено текущее положение БПЛА, осуществляется смещение «камеры-наблюдателя» 3D-сцены; зажатие колеса мыши позволяет поворачивать сцену, а зажатие правой кнопки мыши – позволяет управлять «зумом»;

(*8*) после выполнения полетного задания оператору следует дать команду «приземление» или «домой» любым описанным выше способом;

(*9*) после завершения полета выключить БПЛА, пульт, мобильное приложение и остановить работу сервера.

Заключение

Рассмотрен и прошел первичную апробацию подход к управлению малым БПЛА типа квадрокоптер на основе жестовых команд. Базовые команды управления движением летательного аппарата включают перемещения в пространстве, повороты вокруг вертикальной оси, взлет и посадку. Эксперименты показали достаточную для практического использования точность распознавания ряда жестовых команд с применением сверточных искусственных нейронных сетей, соответствующую зарубежным аналогам [4–9, 15], задача полностью решена с помощью архитектур Yolo 8 Ghost и Yolo 9 для режимов с ориентированными и неориентированными прямоугольниками для выделения зон с жестами.

Дальнейшим расширением подхода может служить управление жестами несколькими БПЛА. Выполнение сложных групповых заданий вызывает необходимость решения задач, связанных с безопасным совместным полетом и взаимодействием автономных летающих роботов в процессе функционирования на одних и тех же участках. Например, когда в условиях пожара перед несколькими БПЛА в процессе выполнения спасательных работ возникает необходимость в совместном поднятии в воздух тяжелого объекта для смены его местоположения. В этом случае каждый БПЛА должен быть способным к коллективному взаимодействию элементов группы как интеллектуальных агентов. Подобные сложные задачи интеллектуального взаимодействия автономных БПЛА требуют расширения системы команд для придания специфики каждому летательному аппарату. Кроме того, требуется наделение интерфейса и БПЛА некоторыми интеллектуальными функциями.

Список использованных источников

[1] Абрамов Н. С., Талалаев А. А., Фраленко В. П., Хачумов М. В. *Система мультимодального управления и визуализации полета беспилотного летательного аппарата* // Авиакосмическое приборостроение.– 2023.– № 9.– С. 3–11. doi ↑22

[2] Алфимцев А. Н. *Разработка и исследование методов захвата, отслеживания и распознавания динамических жестов*, Диссертация на соискание ученой степени кандидата технических наук.– М..– 2008.– 167 с. ↑22

[3] Нагапетян В. Э. *Методы распознавания жестов руки на основе анализа дальностных изображений*, Диссертация на соискание ученой степени кандидата физико-математических наук.– М..– 2008.– 117 с. ↑22, 24

[4] Oikonomidis I., Kyriazis N., Argyros A. A. *Efficient model-based 3D tracking of hand articulations using Kinect* // *Proceedings of the 22nd British Machine Vision Conference 2011*, BMVC'11 (Dundee, UK, August 29–September 2, 2011).– 2011.– ISBN 1-901725-43-X.– 11 pp. URL doi ↑22, 31

[5] Shaowei C., Tanaka J. *Interacting with a self-portrait camera using motion-based hand gestures // Proceedings of the 11th Asia-Pacific Conference on Computer-Human Interaction 2013*, APCHI'13 (Bangalore, India, September 24–27, 2013), New York: ACM.– 2013.– ISBN 978-1-4503-2253-9.– Pp. 93–101. ↑23, 31

[6] Maraqa M. R., Al-Zboun F., Dhyabat M., Zitar R. A. *Recognition of Arabic Sign Language (ArSL) using recurrent neural networks* // Journal of Intelligent Learning Systems and Applications.– 2012.– Vol. **4**.– No. 1.– Pp. 41–52. ↑23, 31

[7] Pugeault N., Bowden R. *Spelling it out: real-time ASL fingerspelling recognition // Proceedings of the IEEE International Conference on Computer Vision Workshops 2011*, ICCV'11 (Barcelona, Spain, 06–13 November 2011).– 2011.– Pp. 1114–1119. ↑23, 31

[8] Zhao R., Wang K., Divekar R., Rouhani R., Su H., Ji Q. *An immersive system with multi-modal human-computer interaction // Proceedings of the 13th IEEE International Conference on Automatic Face & Gesture Recognition 2018* (Xi'an, China, 15–19 May 2018).– 2018.– Pp. 517–524. ↑23, 31

[9] Sanna A., Lamberti F., Paravati G., Manuri F. *A kinect-based natural interface for quadrotor control* // Entertainment Computing.– 2013.– Vol. **4**.– No. 3.– Pp. 179–186. ↑23, 31

[10] Нагапетян В. Э., Хачумов В. М. *Распознавание жестов руки в задаче бесконтактного управления беспилотным летательным аппаратом* // Автометрия.– 2015.– Т. **51**.– № 2.– С. 103–109. ↑23

[11] Абрамов Н. С., Емельянова Ю. Г., Талалаев А. А., Фраленко В. П., Хачумов М. В. *Архитектура мультимодального интерфейса для управления беспилотным летательным аппаратом* // Известия высших учебных заведений. Авиационная техника.– 2022.– № 3.– С. 55–63. ↑23, 27

[12] Hu M. K. *Visual pattern recognition by moment invariants* // IRE Transactions on Information Theory.– 1962.– Vol. **8**.– No. 2.– Pp. 179–187. ↑24

[13] Hochreiter S., Schmidhuber J. *Long short-term memory* // Neural Computation.– 1997.– Vol. **9**.– No. 8.– Pp. 1735–1780. ↑24

[14] Жуковская В. А., Пятаева А. В. *Рекуррентная нейронная сеть для распознавания жестов русского языка с учетом языкового диалекта Сибирского региона // ГрафиКон 2022: материалы 32-й Международной конференции по компьютерной графике и машинному зрению* (Рязань, 19–22 сентября 2022 г.).– 2022.– С. 538–547. ↑24

[15] Булыгин Д. А., Мамонова Т. Е. *Распознавание жестов рук в режиме реального времени* // Научный вестник НГТУ.– 2020.– № 1(78).– С. 25–40. ↑24, 31

[16] Zhao Y., Lv W., Xu S., Wei J., Wang G., Dang Q., Liu Y., Chen J., *DETRs beat YOLOs on real-time object detection.*– 2023.– 14 pp. arXiv:2304.08069 [cs.CV] ↑27

Поступила в редакцию 09.02.2024;
одобрена после рецензирования 13.03.2024;
принята к публикации 17.03.2024;
опубликована онлайн 22.04.2024.

Рекомендовал к публикации *д.ф.-м.н. А. М. Елизаров*

Информация об авторах:

Николай Сергеевич Абрамов

к.т.н., ведущий научный сотрудник ИЦМС ИПС им. А.К. Айламазяна РАН. Область научных интересов: математические методы синтеза, обработки и анализа изображений и сигналов, искусственный интеллект и принятие решений, интеллектуальный анализ данных и распознавание образов, геометрия.

0000-0002-1612-3879
e-mail: *n-say@nsa.pereslavl.ru*

Вита Викторовна Саттарова

Студент Российского университета дружбы народов им. Патриса Лумумбы. Область научных интересов: интеллектуальный анализ данных и распознавание образов, искусственный интеллект и принятие решений, управление робототехническими системами.

0009-0008-0425-6958
e-mail: *1032201655@pfur.ru*

Виталий Петрович Фраленко

Кандидат технических наук, ведущий научный сотрудник ИЦМС ИПС им. А.К. Айламазяна РАН. Область научных интересов: интеллектуальный анализ данных и распознавание образов, искусственный интеллект и принятие решений, параллельно-конвейерные вычисления, сетевая безопасность, диагностика сложных технических систем, графические интерфейсы.

0000-0003-0123-3773
e-mail: *alarmod@pereslavl.ru*

Михаил Вячеславович Хачумов

Кандидат физико-математических наук, старший научный сотрудник ИЦМС ИПС им. А.К. Айламазяна РАН, старший научный сотрудник ФИЦ «Информатика и управление» РАН, доцент РУДН им. Патриса Лумумбы, доцент МИРЭА. Область научных интересов: интеллектуальный анализ данных и распознавание образов, искусственный интеллект и принятие решений, управление робототехническими системами.

0000-0001-5117-384X
e-mail: *khmike@inbox.ru*

Авторы внесли равный вклад в подготовку публикации.

Декларация об отсутствии личной заинтересованности: *благополучие авторов не зависит от результатов исследования.*

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article ARTIFICIAL INTELLIGENCE, INTELLIGENCE SYSTEMS, NEURAL NETWORKS

UDC 004.93:004.35
10.25209/2079-3316-2024-15-2-21-35

Gesture control of small unmanned aerial vehicle flight

Nikolai Sergeevich **Abramov**[1], Vita Viktorovna **Sattarova**[2], Vitaly Petrovich **Fralenko**[3], Mikhail Vyacheslavovich **Khachumov**[4]

[1, 3, 4] Ailamazyan Program Systems Institute of RAS, Ves'kovo, Russia
[2] RUDN University, Moscow, Russia
[4] Russian state university for the humanities, Moscow, Russia
[4] Federal Research Center "Computer Science and Control" of RAS, Moscow, Russia
[4] MIREA - Russian Technological University, Moscow, Russia

Abstract. The problem of constructing gesture commands for controlling a small unmanned aerial vehicle, such as a quadcopter, is considered. Commands coming from a video camera are identified by a classifier based on a convolutional neural network, and the multimodal control interface equipped with an intelligent solver converts them into control commands for the quadcopter. Neural networks from the Ultralytics neural network library allow selecting targets in a frame in real-time. The commands are sent to a specialized program on a smartphone, developed on the basis of DJI SDK flight simulators, which then sends commands via the remote control channel.

The quality of recognition of developed gesture commands for DJI Phantom 3 standard edition quadcopters is investigated, and a brief guide in the form of operator work scenarios with unmanned vehicles is provided. The prospects of gesture control of several vehicles in extreme conditions have been revealed, considering the complex safety challenges of joint flight and interaction of aircraft in confined space. (*In Russian*).

Key words and phrases: unmanned aerial vehicle, control, gestures, convolutional neural network, Ultralytics, intelligent interface, recognition

2020 *Mathematics Subject Classification*: 68T45; 68T07, 68T40

Acknowledgments: This work was financially supported by the Russian Science Foundation, project № 21–71–10056, https://rscf.ru/project/21-71-10056/

For citation: Nikolai S. Abramov, Vita V. Sattarova, Vitaly P. Fralenko, Mikhail V. Khachumov. *Gesture control of small unmanned aerial vehicle flight*. Program Systems: Theory and Applications, 2024, **15**:2(61), pp. 21–35. (*In Russ.*). https://psta.psiras.ru/read/psta2024_2_21-35.pdf

References

[1] N. S. Abramov, A. A. Talalaev, V. P. Fralenko, M. V. Xachumov. "Multimodal control and visualization system for unmanned aerial flight", *Aviakosmicheskoe priborostroenie*, 2023, no. 9, pp. 3–11 (in Russian).

[2] A. N. Alfimcev. *Development and research of methods for capturing, tracking and recognizing dynamic gestures*, Dissertaciya na soiskanie uchenoj stepeni kandidata texnicheskix nauk, M., 2008, 167 pp. (in Russian).

[3] V. E. Nagapetyan. *Hand gesture recognition methods based on range image analysis*, Dissertaciya na soiskanie uchenoj stepeni kandidata fiziko-matematicheskix nauk, M., 2008, 117 pp. (in Russian).

[4] I. Oikonomidis, N. Kyriazis, A. A. Argyros. "Efficient model-based 3D tracking of hand articulations using Kinect", *Proceedings of the 22nd British Machine Vision Conference 2011*, BMVC'11 (Dundee, UK, August 29–September 2, 2011), 2011, ISBN 1-901725-43-X, 11 pp.

[5] C. Shaowei, J. Tanaka. "Interacting with a self-portrait camera using motion-based hand gestures", *Proceedings of the 11th Asia-Pacific Conference on Computer-Human Interaction 2013*, APCHI'13 (Bangalore, India, September 24–27, 2013), ACM, New York, 2013, ISBN 978-1-4503-2253-9, pp. 93–101.

[6] M. R. Maraqa, F. Al-Zboun, M. Dhyabat, R. A. Zitar. "Recognition of Arabic Sign Language (ArSL) using recurrent neural networks", *Journal of Intelligent Learning Systems and Applications*, **4**:1 (2012), pp. 41–52.

[7] N. Pugeault, R. Bowden. "Spelling it out: real-time ASL fingerspelling recognition", *Proceedings of the IEEE International Conference on Computer Vision Workshops 2011*, ICCV'11 (Barcelona, Spain, 06–13 November 2011), 2011, pp. 1114–1119.

[8] R. Zhao, K. Wang, R. Divekar, R. Rouhani, H. Su, Q. Ji. "An immersive system with multi-modal human-computer interaction", *Proceedings of the 13th IEEE International Conference on Automatic Face & Gesture Recognition 2018* (Xi'an, China, 15–19 May 2018), 2018, pp. 517–524.

[9] A. Sanna, F. Lamberti, G. Paravati, F. Manuri. "A kinect-based natural interface for quadrotor control", *Entertainment Computing*, **4**:3 (2013), pp. 179–186.

[10] V. E. Nagapetyan, V. M. Khachumov. "Gesture recognition in the problem of contactless control of an unmanned aerial vehicle", *Optoelectronics, Instrumentation and Data Processing*, **51**:2 (2015), pp. 192–197.

[11] N. S. Abramov, Yu. G. Emel'yanova, A. A. Talalaev, V. P. Fralenko, M. V. Khachumov. "Multimodal interface architecture for unmanned aerial vehicle control", *Russian Aeronautics*, **65**:3 (2022), pp. 498–506.

[12] M. K. Hu. "Visual pattern recognition by moment invariants", *IRE Transactions on Information Theory*, **8**:2 (1962), pp. 179–187.

[13] S. Hochreiter, J. Schmidhuber. "Long short-term memory", *Neural Computation*, **9**:8 (1997), pp. 1735–1780.

[14] V. A. Zhukovskaya, A. V. Pyataeva. "Recurrent neural network for recognition of gestures of the Russian language, taking into account the language dialect of the Siberian region", *GrafiKon 2022: materialy 32-j Mezhdunarodnoj*

konferencii po komp'yuternoj grafike i mashinnomu zreniyu (Ryazan', 19–22 sentyabrya 2022 g.), 2022, pp. 538–547 (in Russia).

[15] D. A. Bulygin, T. E. Mamonova. "Recognition of hand gestures in real time", *Nauchnyj vestnik NGTU*, 2020, no. 1(78), pp. 25–40 (in Russian).

[16] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, J. Chen,. *DETRs beat YOLOs on real-time object detection*, 2023, 14 pp. arXiv 2304.08069 [cs.CV]

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
Обзорная статья МЕТОДЫ ОПТИМИЗАЦИИ И ТЕОРИЯ УПРАВЛЕНИЯ

УДК 004.415.53
10.25209/2079-3316-2024-15-2-37-86

Систематический обзор методов составления тестовых инвариантов

Софья Федоровна **Якушева**[1]✉, Антон Сергеевич **Хританков**[2]
[1, 2] Московский физико-технический институт, Москва, Россия
[2] Высшая школа экономики, Москва, Россия

Аннотация. Тестирование инвариантами (metamorphic testing) – один из наиболее эффективных методов тестирования программ, для которых сложно подбирать тестовые примеры и формулировать тестовые оракулы. При тестировании инвариантами вместо проверки правильности вывода программы на отдельных наборах входных данных проверяется выполнение тестового инварианта (metamorphic relation) – функции от нескольких наборов исходных данных и соответствующих им ответов программы. Составление тестовых инвариантов требует понимания решаемой программой задачи и творческого подхода.

Предлагаемый систематический обзор посвящён выявлению широкоприменимых методик получения инвариантов и повторяющихся приёмов составления инвариантов в разных научных областях. На основе проведенного анализа предложена классификация инвариантов на шесть основных типов, выявлены типовые преобразования исходных данных, используемые при составлении инвариантов в нескольких областях знаний. Результаты обзора будут полезны исследователям в примененении тестирования инвариантами на практике к верификации наукоемких программ и алгоритмов машинного обучения. (*Связанные тексты статьи на русском и на английском языках*)

Ключевые слова и фразы: тестирование инвариантами, тестовый инвариант, тестирование программного обеспечения, проблема формулирования тестового оракула

Для цитирования: Якушева С. Ф., Хританков А. С. *Систематический обзор методов составления тестовых инвариантов* // Программные системы: теория и приложения. 2024. **Т. 15**. № 2(61). С. 37–86. (*Русс.+англ.*).
https://psta.psiras.ru/read/psta2024_2_37-86.pdf

Введение

Контроль качества программного обеспечения (ПО) является важной частью процесса разработки. На этапе контроля проверяется, что ПО соответствует всем предъявленным к ней требованиям, а значит, будет вести себя ожидаемым и заранее известным образом. Отклонения поведения программного обеспечения от заявленных требований регулярно становятся причинами авиационных происшествий, аварий с участием самоуправляемых автомобилей, неудач космических миссий, нарушений безопасности данных [1]. Несоответствие требованиям является веским основанием для доработки разрабатываемой системы или выведения из эксплуатации уже используемой.

Тестирование – один из методов контроля качества программ в процессе разработки. Тестирование позволяет выявлять некорректное или ошибочное поведение программы, нарушения функциональных и нефункциональных требований, не предусмотренные документацией сценарии использования. Тестовым оракулом (test oracle) [2] называется функция, которая определяет правильность ответа программы. Простейший тестовый оракул сравнивает ответ программы с заранее известным эталоном (см. рисунок 1). Более сложным примером является проверка правильности найденного гамильтонова пути в графе: достаточно проверить наличие в пути всех вершин графа и наличие рёбер между соседними парами вершин пути, эталон для такой проверки не требуется. В плохо формализуемых случаях, если требования сформулированы неточно или требуется высокий уровень экспертизы, то в качестве тестового оракула могут выступать пользователь, эксперт, другая программа.

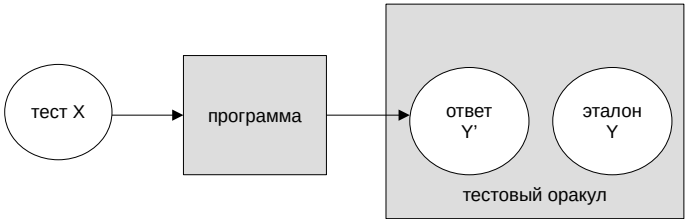


Рисунок 1. Пример проверки правильности прохождения теста. В роли оракула выступает сравнение ответа с эталоном.

При тестировании наукоемких, недетерминированных, распределенных программных комплексов существенным затруднением является так называемая проблема формулирования тестового оракула. Проблема

формулирования тестового оракула (test oracle problem) [3] состоит в сложности составления тестовых оракулов – в частности, автоматических. Она актуальна для задач, требующих полного перебора для нахождения точного решения, например, в биоинформатике и комбинаторике; задач машинного обучения из-за затратности процесса сбора и разметки тестовых выборок; задач создания художественных объектов из-за необходимости привлечения человека для оценки выполнения плохо сформулированных требований. Тем не менее, были предложены различные методы так или иначе решить эту проблему, например, тестирование свойств (property-based testing), в котором проверяется выполнение заданного соотношения между входами и выходами программы и, как его разновидность, тестирование инвариантами (metamorphic testing).

1. Тестирование инвариантами

1.1. Определение и примеры

Метод тестирования инвариантами (metamorphic testing) [4] применяется во многих предметных областях и развивается в нескольких направлениях [5, 6]. Идея метода заключается в том, что вместо проверки правильности каждого конкретного ответа программы проверяется выполнение тестового инварианта (metamorphic relation), вычисляемого для нескольких наборов исходных данных и соответствующих им ответов программы. Тестовый инвариант для программы – это функция вида

$$R(x_1, x_2, \ldots, x_n, f(x_1), f(x_2), \ldots, f(x_n)) \longrightarrow \{0, 1\}, \tag{1}$$

где $n \geq 2$ – общее количество запусков в тестовом случае, x_i – исходные данные для i-го запуска в тестовом случае, а $f(x_i)$ – i-й ответ программы. Методика тестирования инвариантами состоит из следующих шагов. Сначала задаем способ получения входных данных, в том числе с использованием генератора данных. Далее запускаем программу на каждом наборе исходных данных для нескольких запусков в тестовом случае и проверяем выполнение инварианта. Нарушение выполнения инварианта свидетельствует о наличии ошибок или несоответствии требованиям. Таким образом, вместо прямой проверки ответов, в методе используется так называемый производный тестовый оракул (derived oracle) [6]. Построение такого инварианта для ряда задач может естественным образом следовать из свойств решаемой программой задачи и быть проще для разработчика программы или исследователя.

Один из видов тестовых инвариантов можно описать следующим образом. Если два элемента последовательности исходных данных удовлетворяют некоторому соотношению I, то два соответствующих ответа программы должны удовлетворять соотношению O (см. рисунок 2):

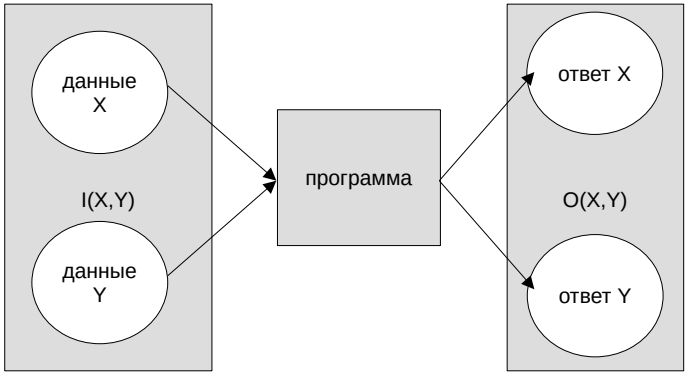


Рисунок 2. Простой пример тестирования инвариантами. Тестовый инвариант – функция $R(x, y, X, Y) = I(x, y) \cap O(X, Y)$. Наличие зависимости $I(x, y)$ между исходными данными должно повлечь наличие зависимости $O(X, Y)$ между ответами.

$$(2)\quad R(x_1, x_2, \ldots, x_n, f(x_1), f(x_2), \ldots, f(x_n)) = \\ = I(x_1, x_2, \ldots, x_n) \cap O(f(x_1), f(x_2), \ldots, f(x_n)),$$

где I – преобразование исходных данных, O – ожидаемое отношение между ответами, $n \geq 2$ – общее количество запусков в тестовом случае, x_i – исходные данные для i-го запуска в тестовом случае, а $f(x_i)$ – i-й ответ программы (см. рисунок 2). Будем рассматривать инварианты при условиях, что выполнение тестовых запусков происходит независимо, а начальное состояние программы в каждом запуске известно.

Для примера приведём несколько инвариантов для проверки правильности выполнения запросов к реляционной базе данных, содержащей таблицу $T = a|b|c|...$, где a, b, c – столбцы таблицы.

- Пусть A и B – два условия поиска. Тогда ответ на запрос с условием $A \cap B$ – подмножество ответа на запрос с условием A (аналогично B).
- Пусть A – условие поиска. Тогда ответы на запрос с условием A и запрос с условием $\neg A$ не пересекаются.

- Пусть A – условие поиска. Тогда количество ответов на запрос с условием A при сортировке по возрастанию столбца a и при сортировке по убыванию столбца a одинаково.

1.2. Терминология

Стоит отметить, что рассматриваемая тема пока мало освящена в русскоязычной литературе, а потому не имеет устоявшейся общепринятой терминологии. В работах встречается термин «метаморфное тестирование», являющийся транскрипцией английского термина «metamorphic testing».

В русскоязычном тексте использованы термины «тестирование инвариантами» и «тестовый инвариант» применимо к рассматриваемой методике. Такие термины гораздо легче понять и запомнить из-за простой тесной связи с определением. В математической литературе инвариантом называют величину (или свойство), остающееся постоянным в пределах рассматриваемой ситуации. Под тестовым инвариантом обычно подразумевается функция, которая должна сохранять свое значение при предполагаемых изменениях параметров. Сходное по смыслу значение термин «инвариант» имеет и в физике, например в сочетании «инвариант движения». Формула 1.1 задаёт такую функцию, то есть по сути «metamorphic relation» – это тестовый инвариант (хотя буквально оно переводится как метаморфное отношение). Кроме того, тестовые инварианты потенциально могут быть получены как следствия из математической модели, реализуемой программой. Поэтому термины «тестовый инвариант» и «тестирование инвариантами» точно отражают суть метода.

2. Цели и методы исследования

На текущем этапе развития тестирования инвариантами одной из нерешенных задач является получение методики составления тестовых инвариантов. На данный момент не разработано широко применимого метода их составления для программ, решающих разные прикладные задачи в разных областях. Это приводит к необходимости трудоемкого составления инвариантов практически с нуля в каждом конкретном случае.

Для преодоления затруднений в составлении тестовых инвариантов в данном исследованиии поставлены следующие задачи.

(*1*) Выявить общеприменимые и повторяющиеся в разных задачах методики построения тестовых инвариантов, которые могут быть полезны в новых задачах. Результаты приведены в разделе 4.

(2) Определить, наблюдаются ли особенности в применении метода тестирования инвариантами в области машинного обучения и смежных областях, позволяющие получать более эффективные инварианты. Подробнее в разделе 5.

(3) Выделить непрямые методы получения инвариантов, а также способы комбинирования инвариантов с другими методами, чтобы упростить применение метода тестирования инвариантами в нестандартных случаях. Результаты собраны в разделе 7.

(4) Выявить упущения и возможности для дальнейшего развития методик построения тестовых инвариантов. Результаты приведены в разделах 3 и 6.

Для достижения целей исследования применен метод систематических обзоров [7], предложенный Б. Китченхам для проведения мета-исследований (secondary study) в области программной инженерии. Применение метода систематического обзора позволяет улучшить воспроизводимость результатов обзора и обоснованность выводов.

Для отбора публикаций использованы следующие критерии: новизна, публикация в период с 2018 по 2023 годы, доступность читателю, релевантность теме исследования, детальное и ясное описание используемых тестовых инвариантов, наличие в работе применимых на практике результатов, возможность обобщения применяемых идей и методик. Качество предлагаемых инвариантов не рассматривалось как критерий поиска, поскольку общего количества исследований по теме недостаточно для применения статистических методов определения средней эффективности того или иного инварианта.

Для того, чтобы изложить базовые результаты тестирования инвариантами, в обзор также включены основополагающие работы в каждой области, некоторые из которых изданы до 2018 года. Для их отбора использовались критерии: релевантность теме, цитируемость, глубина и универсальность предлагаемых идей и методов, практическая ценность. Поиск производился по тем же правилам, но ограничение на дату публикации не применялось. При поиске дополнительно использовались списки источников из наиболее известных англоязычных обзорных статей [4, 6] по данной теме. Также основополагающими работами заменены источники, цитирующие и переиспользующие тестовые инварианты из этих более ранних работ без существенных доработок.

Поиск публикаций производился в открытых источниках (Google Scholar) на английском языке по ключевым словам «metamorphic testing»,

«metamorphic relation», «neural network», «classification», «maps», «graphs», «natural language processing» и комбинациям этих слов. Некоторые дополнительные публикации были найдены при анализе списков источников из этих публикаций. Наличие в тексте источника слова «metamorphic» является необходимым, поскольку в англоязычной литературе не встречается альтернативного названия данного метода. Из поиска исключались патенты и цитирования, обзорные статьи по смежным областям и различным методам тестирования, а также статьи, только упоминающие применение тестирования инвариантами. Поиск проводился с 14 июня по 13 сентября 2023 года.

По ключевым словам «metamorphic testing» за период с 2019 года по 2023 найдено более двух тысяч публикаций, что говорит об актуальности и интересе исследователей к этой теме. Из них по критериям отобрано и изучено около 120 работ, 66 из которых включены в данное исследование.

Также был проведен поиск русскоязычных источников в библиотеке eLibrary. По ключевым словам «тестирование инвариантами», «метаморфическое тестирование» и «метаморфное тестирование» были найдены только 1 работа по теме (краткие тезисы [**8**]) и 1 работа из смежной области (верификация блок-схем [**9**]).

Из каждого источника отбиралось описание предметной области, решаемая задача и предлагаемые инварианты. Далее эта информация группировалась по предметным областям, в каждой области выявлялись наиболее популярные идеи и методики. Из источников, посвящённым описанию методов получения инвариантов, отбиралось также краткое описание предлагаемого метода.

Окончательно обобщаемость методик определялась на заключительном этапе обработки данных по совокупности отобранных источников. На этом этапе формировалась общая классификация методик, а также формулировались базовые и общеприменимые методики составления инвариантов. Корме того, производилось объединение в группы других методов получения инвариантов (композиции, статистические подходы, поиск, комбинации с другими методами).

3. Предлагаемая классификация инвариантов

Анализ приводимых в изученных работах тестовых инвариантов показал, что в большинстве случаев предлагаемые инварианты могут быть представлены в виде (2). Составлена классификация тестовых инвариантов

(см. таблицу 1) по основным преобразованиям I, использованным в этих работах для получения исходных данных $I(x_1, x_2, \ldots, x_n)$ при составлении тестовых инвариантов.

Таблица 1. Классификация по преобразованиям исходных данных

Преобразование	Смысл
LT linear transform	линейное преобразование (в том числе изменение в большую или меньшую сторону, умножение на константу)
SM symmetry	симметрия относительно прямой, плоскости, отношения между объектами
PR permutative	перестановка элементов множества исходных данных
IE inclusive / exclusive	включение / исключение отдельных элементов из множества
BC blur / change	замена или изменение, добавление случайного шума
UD unite / divide	объединение / разбиение на части

В классификацию не включены немногочисленные редкие инварианты, обладающие настолько сильной привязкой к конкретной задаче, что их обобщение на данный момент не представляет интереса. Несколько таких примеров приведено в разделе 4.2.

4. Базовые методики составления инвариантов для проверки моделей и алгоритмов

При изложении использованных в литературе методик составления инвариантов будем сразу обозначать в скобках, к какому из приведенных ранее классов её можно отнести. Например, умножение числовых параметров на отличную от нуля константу можно отнести к линейным преобразованиям (LT).

4.1. Общеприменимые методики

Общеприменимые методики для проверки моделей и алгоритмов — это изменение параметров, перестановка равнозначных параметров и добавление шума.

Часто используется изменение параметров (LT), по которому можно однозначно предсказать изменение результата (улучшится, ухудшится, изменится на известную величину или не изменится вовсе). В качестве примеров можно привести изменение количества вычислительных

устройств [10], изменение параметров эпидемиологической модели [11, 12], параметров и гиперпараметров нейросетевых моделей [13].

Встречается также применение методики перестановок равнозначных параметров (PR) – чаще всего в таком случае можно ожидать неизменности результата исполнения тестируемой программы. Например, не должны влиять на результат вычислений изменение порядка пользователей облачных сервисов [10], исполняемых операций в условиях многопоточности [14], входных данных в стохастической оптимизации [15]. А порядок добавления пожеланий при бронировании гостиницы с помощью автоматического помощника [16] не должен влиять на результат поиска.

Наложение шума (BC) – широко применяемая методика (ошибки [17, 18], шум [19–21], искажения [22], использование особенностей языка [23]). Чаще всего ожидается, что зашумление не изменит результат, либо же его не улучшит. При замене исходных данных (BC) может ожидаться изменение результата. В качестве примера можно привести уже упомянутый инвариант из раздела 1 с условиями A и $\neg A$ – при таком изменении запроса ожидается, что ответ полностью изменится и не будет пересекаться с предыдущим.

Реже встречается методика добавления или удаления (UD) частей исходных данных. Например, добавление новых пользователей облачных сервисов [10], добавление и удаление единиц языка из текста [24], частей входных данных [18, 25].

Методика проверки с помощью симметрий более активно используется в области изображений и геоинформатике. Например, отражение шахматной доски [26] при тестировании алгоритмов игры в шахматы, отражение изображений относительно осей [27], обращение видеоряда по времени [27], повороты и отражения изображений карт [28], отражения выявляемых алгоритмами кластеров относительно прямой [29]. Как правило, применение симметрии к исходным данным влечёт симметричное изменение ответов.

4.2. Использование особенностей математической модели и предметной области

Эффективные в выявлении ошибок инварианты можно получить, используя особенности математической модели решаемой программой задачи. Например, в работе [30] для проверки правильности получаемых выборок из статистического распределения используется специфическое свойство этого распределения. Для проверки компиляторов глубоких нейронных сетей [31] в оптимизированный код добавляется код, изменяющий получаемый вычислительный граф, но не меняющий сути программы (IE). Для тестирования шахматных алгоритмов [26] используются и свойства

доски, и иерархия фигур: производится отражение доски (SM), фигуры заменяются на фигуры другого цвета (PR) или на более сильные или слабые (BC). Для проверки алгоритмов хеширования, использующих матричные преобразования, и алгоритма RSA [32], применяются матричные преобразования (LT) и добавление модуля (LT), по которому вычисляется остаток. Для проверки другого алгоритма хеширования [33] изменяется исходное сообщение – добавляются или удаляются биты (IE) [33], а сообщения разбивается на части (UD), при этом ожидается изменение значения хеш-функции.

Иногда для составления инвариантов бывают полезны особенности предметной области. В работе [34] по тестированию системы сравнительного генетического анализа используется возможность разделения мутаций на непересекающиеся классы. В работе [35] по тестированию размеченных изображений географических карт применяются соображения, что шоссе и здания не имеют общих площадей, а закрытые области одних путей не пересекаются другими путями. В работе по тестированию приложения с микросервисной архитектурой [36] используются особенности платежной системы. Например, сумма балансов кредиторов и должников после переноса данных с одного микросервиса на другой должна остаться прежней.

5. Методики построения инвариантов в разных областях

5.1. Поисковые алгоритмы

При тестировании поисковых алгоритмов проверяется, изменяется ли ответ ожидаемым образом при добавлении или удалении условий поиска (IE): при добавлении нового условия в нём не должны появляться новые элементы [36, 37], а при удалении условий, сооответственно, исчезать старые. В работе по тестированию помощника для бронирования гостиниц используется изменение порядка добавления условий поиска (PR) [16]. В работе [27] используется симметрия: ответ, отсортированный по убыванию, можно получить из ответа, отсортированного по возрастанию, если записать его в обратном порядке (SM). Также проверяются неизменность цены товара в зависимости от поискового запроса (BC) [27], неизменность ответа при запросе на другом языке (BC) [27]. Ответы на поисковый запрос при отправке из разных частей земного шара (BC) должны совпадать [38], но из-за распределенности систем и различия между копиями базы данных они могут различаться, что может свидетельствовать о недостаточной слаженности работы.

5.2. Машинное обучение и анализ данных

В связи со сложностью задачи тестирования систем машинного обучения и алгоритмов анализа данных, в данной области используется большое

количество интересных методик и преобразований исходных данных. Часто встречается переразбиение обучающей выборки (UD): добавление или исключение элемента из выборки [29, 39, 40], добавление или исключение целых классов [39, 40], дублирование элементов обучающей выборки [39]. Используются изменение метки элемента данных (BC)[40], добавление и исключение признаков (IE) [29, 39, 40]. Исключение незначащих признаков не должно значительно менять качество классификации, а исключение значащих – не улучшать. Перестановки (PR) признаков [39, 40], меток классов [39, 40], элементов обучающей выборки [29], как правило, не должны значительно менять итоговое качество классификатора. В работе о предсказаниях свойств белка используется преобразования белков, гарантированно изменяющие их функции [41] (BC), и от модели ожидается изменение предсказания.

Реже встречаются линейные преобразования (LT): применение аффинного преобразования к признакам выборки [40], и повороты и преобразования системы координат в пространстве [29] (на итоговое качество такие преобразования должны влиять лишь в незначительной степени). Работа [29] по тестированию алгоритмов кластеризации предлагает ещё несколько методик: сжатие выделенных кластеров к их центрам (LT), добавление элементов внутрь выпуклой оболочки кластеров (UD), добавление выбросов (UD). Идея симметрии (SM) нашла применение в виде методики отражения кластеров относительно прямых [29].

5.3. Глубокое обучение и нейронные сети

Методики создания инвариантов для тестирования глубоких нейронных сетей частично повторяют методики для машинного обучения в целом. Например, линейное преобразование признаков (LT) [13, 19, 42]), перестановка меток классов (PR) [13], варьирование разбиения данных на обучающую и тестовую выборки (UD) [13], дублирование данных (UD) [13], увеличение или уменьшение значений параметров алгоритма обучения (LT), таких как скорость обучения (learning rate), параметр α в нелинейной функции активации ReLU, число эпох обучения, число нейронов в слоях [13].

В задаче обработки изображений применяют и другие преобразования. Для проверки качества распознавания лиц применяют изменение цвета волос, бровей, пола, добавление очков, усов и бороды (BC) [43], для самоуправляемых автомобилей – изменение погоды (BC) [44, 45] или замену фона (BC) [46]. Полезной оказывается идея применения неравенств для степеней уверенности модели при детекции на изображении и на его частях (UD) [47]. Применяются симметрии (SM): отражение [27], обращение видеоряда [27], а также преобразования перспективы (LT) [48] и изменение яркости (LT) [48]. Новой идеей является добавление водяных знаков (BC)

и масок (UD) [48]. Для тестирования модели на изображения добавлялись объекты, которых раньше не было в выборке (UD) [49].

Для проверки устойчивости (robustness) моделей применяется зашумление (BC) [19–21], искажение данных (BC) [22]. Используются перестановки элементов обучающей выборки (PR) [42], обучающих и целевых признаков (PR) [42], каналов изображения [48].

5.4. Обработка естественного языка

Для составления инвариантов в области обработки естественного языка чаще других используются изменения исходных данных, не меняющие ответ системы. По-видимому, это связано со сложностями в соотнесении изменений в тексте с изменениями его смысла или других рассматриваемых параметров. Также используются особенности написания и грамматики языков, например, английского и китайского.

Часто применяются замены, которые не должны изменять ответ в рамках решаемой задачи (BC). Для распознавания дискриминации используется замена морфологического или смыслового свойства сущности, например, пола [50]. Для проверки устойчивости используются замены слова на его синоним, перевод на другой язык или не имеющее к нему отношения слово [16, 23, 51], символов на похожие (визуально или фонетически) или на '*' [23]. Используются изменение залога (BC), замена определённых слов, например, «до» на «после» (BC), изменение порядка членов предложения (PR) [51], зашумление опечатками (BC) [23], пропуск слов (IE) [16]. Исследователи, работавшие с китайской письменностью, предложили еще несколько интересных идей [23]: разбиение слов и иероглифов на части (UD), слияние буквосочетаний (UD), перестановку букв в слове (PR), сокращение до акронима (UD), вставку специальных предложений-индикаторов (IE) для распознавания. Подобные преобразования могут быть полезны для оценки устойчивости моделей к шуму.

В области переводов с одного языка на другой используется замена существительного на другое для определения согласованности переводов (BC) [52]. Также используются свойства симметричности (SM): сравниваются исходный текст и его обратный перевод с другого языка [27, 53]; прямой перевод с языка A на язык B и перевод с языка A на язык B через третий язык C [54].

Тестирование инвариантами успешно применяется к системам, работающим с именованными сущностями. В работе [24] по распознаванию сущностей в тексте применяются преобразования, сохраняющие существующие сущности: перестановки абзацев (PR), перестановки списка добавленных случайных слов (PR), склеивание предложений. Добавление

и удаление случайных слов, предложений и абзацев (UD) не должно приводить к потере ранее распознанных сущностей. В работе по нахождению отношений между сущностями [55] используются обмен сущностей местами (SM), замены сущностей (BC). Данные преобразования не влияют на наличие отношения и используют его симметричность.

5.5. Графовые модели. Геоинформатика

Графовые модели применяются в разных задачах, например, при поиске маршрутов [28], моделировании взаимодействия генов [17]. Графовую модель можно структурно изменять, добавляя или удаляя вершины и рёбра. Например, для составления инвариантов для инструмента моделирования генных регуляторных сетей [17] применяются преобразования, заведомо увеличивающие или уменьшающие показатели: изменения параметра ребра (LT) или вершины, добавление или удаление вершин и рёбер разных свойств (IE). В работе [56] по тестированию модели, имитирующей распространение новостей в социальной сети, также используются изменения параметров с предсказуемым результатом: изменение весов, типов и параметров вершин-соседей и т.д.

Важными математическими свойствами графовой модели для геоданных являются симметрия отношения достижимости и наличие неравенства треугольника. Исходя из них, предлагается перемена местами начала и конца маршрута (SM) [27, 28] и применение неравенства треугольника к длинам маршрутов A-C-B и A-B (UD) [28] и к их стоимостям (UD) [38]. Также неравенство треугольника применяется при добавлении и удалении препятствий (IE) [28], добавлении условий (IE) [38]. Проверяется устойчивость (BC): малое изменение начальных точек не сильно меняет ответ [38], ответ на повторный запрос через малый промежуток времени повторяет первоначальный [27].

В области размеченных географических карт и поиска маршрутов по картам, помимо вышеупомянутых свойств, часто применяются преобразования, изменяющие лишь представление изображения карты. Например, изменение системы координат [28] (LT), повороты относительно точки и отражения изображения карты относительно прямой [28] (SM). В итоге такие преобразования не изменяют результат, а в случае изменения расстояний – изменяют длину пути пропорционально.

В работе по проверке размеченных карт [35] используются особенности предметной области. Проверяется, что у каждого пути есть хотя бы одно начало и один конец, у каждой кольцевой развязки есть вход и выход, закрытые области одних путей не пересекаются другими путями.

5.6. Биоинформатика (тексты)

Тестирование инвариантами применяется для проверки выравнивателей – программ, которые соотносят считанные секвенатором небольшие участки нуклеиновых кислот (т.н. прочтения, reads) с референсным геномом и определяют их координаты. Если для прочтения успешно вычислены его координаты на геноме, оно называется картируемым, иначе – некартируемым.

Примером применения симметрии (SM) является использование перевернутых прочтений [17, 18]. Также предлагаются следующие преобразования: изменение порядка прочтений во входе (PR) [18], перестановки алфавита (PR) [17], добавление [18, 25] и удаление [18] прочтений (UD), использование только картируемых или некартируемых прочтений (UD) [18, 25], расширение прочтений (IE) [18], добавление и удаление множеств прочтений (UD) [17], изменение максимального допустимого количества несовпадений прочтения с участком генома [17]. Также, если сравнивать прочтения с соответствующими участками генома, можно реализовать добавление [17] и удаление [17, 18] ошибок в прочтениях (BC), замену ошибок на ошибки другого типа (BC) [17].

5.7. Компиляторы

Компилятор – это программа, преобразующая код, написанный на языке программирования, в набор машинных кодов. Основной идеей, применяемой для тестирования компиляторов, является создание программы, эквивалентной исходной. Для этого используется добавление мертвого кода и тождественных функций (IE) [57]. Применяется сокращение программы через сложный анализ покрытия кода (BC) [58].

На данный момент созданы компиляторы для преобразования высокоуровневой модели глубокой нейронной сети в оптимизированный исполняемый код [31]. Для тестирования такого компилятора применяется добавление сложных конструкций, не меняющих суть модели, но изменяющих получаемый граф вычислений (IE).

6. Анализ применения методик построения инвариантов по областям

На основании найденных часто применяемых идей преобразований входных данных для тестирования инвариантами была составлена сводная таблица 2. В ячейках таблицы приведены объекты, параметры или свойства, изменяемые согласно указанной в колонке идее при составлении тестовых инвариантов в области применения, указанной слева. Можно заметить, что часто изменяются минимальные неделимые единицы из рассматриваемой области, такие как вершина или ребро графа, элемент выборки, языковая единица, либо их представление в памяти компьютера.

Таблица 2. Изменения единиц входных данных в инвариантах

Область	LT	SM	PR	IE	BC	UD
Поисковые алгоритмы		порядок выдачи	порядок условий	условие	язык, точка отправки	
Задачи машинного обучения	шкала, параметры	симметричный объект	равнозначные признаки	признаки	метка	выборка
Глубокие нейронные сети	признаки, гиперпараметры	симметричный объект	признаки, классы	новые объекты	объект, фон	выборка, картинка
Геоинформатика	система координат	система координат, отношения	начало и конец пути	препятствия, условия	точка	пути
Графовые модели	ребро	начало и конец пути		вершины		пути
Обработка естественного языка		языки, сущности	единицы языка	слова, индикаторы	единицы языка, язык	единицы языка
Биоинформатика		прочтения, алфавит	прочтения, алфавит	прочтения, мутации, гены	прочтения, ошибки	прочтения
Компиляторы				синтаксические конструкции	текст программы	

Не для всех рассмотренных областей были предложены инварианты всех шести выделенных типов, что может послужить направлением дальнейших исследований. Кроме того, для построения инвариантов использованы не все возможные единицы рассматриваемых областей. Например, в графе можно зашумлять пропускные способности и другие характеристики вершин и рёбер (BC), а сами рёбра переставлять (PR) при хранении графа в виде списков смежности. В области естественного языка можно попытаться использовать векторное представление слов для применения линейных преобразований (LT). В области распознавания объектов на изображениях можно использовать изображения-коллажи для сравнения результатов (UD). Для поисковых запросов можно изменять параметр фильтра в большую или меньшую сторону (LT), сравнивать количество ответов для разных отрезков значений рассматриваемого параметра (UD) – например, что общее количество ответов равно суммам количеств ответов для двух половин этого отрезка.

Таким образом, полученная таблица отображает следующий алгоритм составления инвариантов: исследователи рассматривают минимальные единицы в предметной области и применяют к ним преобразования из шести указанных групп. Если при этом удаётся однозначно сказать, к какому изменению ответа должно приводить преобразование, получается тестовый инвариант.

7. Другие методы получения и применения инвариантов

Помимо получения инвариантов напрямую из постановки задачи или свойств модели, возможно составление композиций уже существующих инвариантов, использование методов математической статистики для применения к стохастическим системам, комбинирование с другими методами. Эти приёмы также способствуют получению новых, иногда более эффективных инвариантов, поэтому рассмотрим их подробнее.

Применение методов статистики в тестировании инвариантами. При применении тестирования инвариантами к стохастическим системам могут возникнуть случайные ошибки, поэтому для проверки правильности инвариантов могут применяться статистические подходы. Одна из первых работ по применению статистики в тестировании инвариантами [30] в качестве примера рассматривает тестирование свойств реализации конкретного статистического распределения. В более поздних работах встречается применение методов статистики для определения выполнения инвариантов: ANOVA [54], ранговая корреляция Спирмена [59], критерий для проверки наличия статистически значимых различий

[60, 61]. Обобщение этого подхода предлагается в работе [74]: тестовый инвариант представляется в виде композиции процедуры получения выборки и статистического критерия.

Поиск инвариантов. Не всегда инварианты легко выводятся из постановки задачи или модели алгоритма решения. Кроме того, этот процесс на текущий момент плохо автоматизирован. Поэтому отдельное направление в тестировании инвариантами посвящено поиску инвариантов [62, 63]. В работе [64] инварианты ищутся в виде полиномиальных функций, а в [65] поиск производится динамически по категориям исходных данных и ответов. Работа [66] предлагает использование алгоритма SVM на графах программ для предсказания инвариантов. Также предложена идея поиска описаний на естественном языке, которые можно преобразовать в тестовые инварианты [67].

Специальные виды и композиции инвариантов. В одной из ранних работ [68] предложена идея итеративно подбирать исходные данные для тестовых случаев. Эта идея развита в работе [69]: для тестирования линейной модели напрямую используется ответ, полученный в результате первого тестового запуска. В работе [39] предложена идея многомерных инвариантов. В работе по обнаружении ошибок на основе спектров [70] вводится понятие инвариантного среза (metamorphic slice).

Ранней работой по композициям инвариантов является работа [71]. В ней рассматриваются сложные функции-композиции, а также показывается, что эффективность композиции обычно превосходит эффективность отдельных инвариантов. В статье [72] проведено теоретическое исследование возможностей композиции и рассмотрены некоторые примеры. В работе по проверке географических систем [73] используются композиции инвариантов вида z(y(x())) с 2-4 уровнями вложенности, причём используемые простые инварианты предназначены для проверки разных свойств: тестовых требований, свойств программы, свойств алгоритма и т.д. В работе [34] предложен метод составления композиций инвариантов для систем с помощью логических функций.

Комбинации с другими методами. В работе [21] тестирование инвариантами комбинируется с фаззингом: данные лидара немного искажаются произвольным образом, что позволяет оценить устойчивость системы. Интересным приемом является использование тестирования инвариантами и адаптивного случайного тестирования [75]: при генерации последующего теста измерялось расстояние до трёх предшествующих. Кроме этого, тестирование инвариантами применяется в обнаружении ошибок на основе спектров [70] и в сочетании с генетическими алгоритмами [76].

Заключение

Систематически рассмотрены методики построения тестовых инвариантов в разных областях знаний. Проведен анализ доступной литературы, выявлены наиболее часто встречающиеся приемы и подробно рассмотрены популярные области применения тестирования инвариантами.

В результате исследования

- выделено шесть часто встречающихся типов изменений исходных данных, использующихся при составлении тестовых инвариантов: линейные преобразования, симметрии, перестановки, добавление или удаление элементов, замена или зашумление, а также объединение или разбиение на части;
- проанализированы инварианты для задач из разных областей знаний и проведено их сравнение;
- рассмотрены непрямые методы получения инвариантов, такие как создание композиций имеющихся инвариантов, применение статистических подходов, комбинирование с другими методами.
- получена сводная таблица основных изменяемых единиц в каждой области и выявлены возможности для составления новых инвариантов.

Результаты проведённого анализа могут способствовать построению общей универсальной теории поиска тестовых инвариантов, популяризации тестирования инвариантами, возникновению новых приложений и технологий автоматизации тестирования. Мы надеемся, что наша работа поможет исследователям проще и быстрее применять этот метод.

Список использованных источников

[1] Wong W. E., Debroy V., Surampudi A., Kim H., Siok M. F. *Recent catastrophic accidents: Investigating how software was responsible*, 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement (09-11 June 2010, Singapore).– 2010.– Pp. 14–22. doi ↑38

[2] Howden W. *Theoretical and empirical studies of program testing* // IEEE Transactions on Software Engineering.– 1978.– Vol. **SE-4**.– No. 4.– Pp. 293–298. doi ↑38

[3] Barr E. T., Harman M., McMinn P., Shahbaz M., Yoo S. *The oracle problem in software testing: A survey* // IEEE Transactions on Software Engineering.– 2015.– Vol. **41**.– No. 5.– Pp. 507–525. doi ↑39

[4] Chen T. Y., Cheung S. C., Yiu S. M. *Metamorphic testing: a new approach for generating next test cases.*– 2020.– 11 pp. arXiv 2002.12543 ↑39, 42

[5] Chen T. Y., Tse T. *New visions on metamorphic testing after a quarter of a century of inception* // *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.– 2021.– Pp. 1487–1490. ↑39

[6] Chen T. Y., Kuo F.-C., Liu H., Poon P.-L., Towey D., Tse T., Zhou Z. Q. *Metamorphic testing: A review of challenges and opportunities* // ACM Computing Surveys.– 2018.– Vol. **51**.– No. 1.– Pp. 1–27. ↑39, 42

[7] Kitchenham B. *Procedures for performing systematic reviews*, Keele University Technical Report TR/SE-0401.– Keele, UK: Keele University.– 2004.– 33 pp. ↑42

[8] Фальковский Р. Р. *Метаморфное тестирование программ улучшения изображений* // *XIX Международная телекоммуникационная конференция молодых ученых и студентов «МОЛОДЕЖЬ И НАУКА»*, Тезисы докладов.– Т. 3, М.: НИЯУ МИФИ.– 2015.– ISBN 978-5-7262-2223-3.– С. 176–177. ↑43

[9] Миронов А. М. *Верификация программ методом инвариантов* // Интеллектуальные системы. Теория и приложения.– 2017.– Т. **21**.– № 4.– С. 31–49. ↑43

[10] Núñez A., Cañizares P. C., Núñez M., Hierons R. M. *TEA-Cloud: A formal framework for testing cloud computing systems* // IEEE Transactions on Reliability.– 2020.– Vol. **70**.– No. 1.– Pp. 261–284. ↑45

[11] Pullum L. L., Ozmen O. *Early results from metamorphic testing of epidemiological models* // *2012 ASE/IEEE International Conference on BioMedical Computing (BioMedCom)* (14-16 December 2012, Washington, DC, USA).– IEEE.– 2012.– Pp. 62–67. ↑45

[12] Ramanathan A., Steed C. A., Pullum L. L. *Verification of compartmental epidemiological models using metamorphic testing, model checking and visual analytics* // *2012 ASE/IEEE International Conference on BioMedical Computing (BioMedCom)* (14-16 December 2012, Washington, DC, USA).– IEEE.– 2012.– Pp. 68–73. ↑45

[13] Ellis J. D., Iqbal R., Yoshimatsu K. *Verification of the neural network training process for spectrum-based chemical substructure prediction using metamorphic testing* // Journal of Computational Science.– 2021.– Vol. **55**.– id. 101456. ↑45, 47

[14] Sun C.-A., Dai H., Geng N., Liu H., Chen T. Y., Wu P., Cai Y., Wang J. *An interleaving guided metamorphic testing approach for concurrent programs* // ACM Transactions on Software Engineering and Methodology.– 2023.– Vol. **33**.– No. 1.– id. 8.– 21 pp. ↑45

[15] Yoo S. *Metamorphic testing of stochastic optimisation* // *2010 Third International Conference on Software Testing, Verification, and Validation Workshops* (06-10 April 2010, Paris, France).– IEEE.– 2010.– Pp. 192–201. ↑45

[16] Bozic J., Wotawa F. *Testing chatbots using metamorphic relations* // *Testing Software and Systems*, 31st IFIP WG 6.1 International Conference, ICTSS 2019 (15-17 October 2019, Paris, France), Lecture Notes in Computer Science.– vol. **11812**, eds. Gaston C., Kosmatov N., Le Gall P., Cham: Springer.– 2019.– ISBN 978-3-030-31279-4.– Pp. 41–55. ↑45, 46, 48

[17] Chen T. Y., Ho J. W., Liu H., Xie X. *An innovative approach for testing bioinformatics programs using metamorphic testing* // BMC bioinformatics.– 2009.– Vol. **10**.– id. 24.– 12 pp. doi ↑45, 49, 50

[18] Giannoulatou E., Park S.-H., Humphreys D. T., Ho J. W. *Verification and validation of bioinformatics software without a gold standard: a case study of BWA and Bowtie* // BMC bioinformatics.– 2014.– Vol. **15**.– id. S15.– 8 pp. doi ↑45, 50

[19] Tian Y., Pei K., Jana S., Ray B. *DeepTest: Automated testing of deep-neural-network-driven autonomous cars* // *Proceedings of the 40th International Conference on Software Engineering* (27 May 2018-3 June 2018, Gothenburg, Sweden), New York: ACM.– 2018.– ISBN 978-1-4503-5638-1.– Pp. 303–314. doi ↑45, 47, 48

[20] Wu C., Sun L., Zhou Z. Q. *The impact of a dot: Case studies of a noise metamorphic relation pattern* // *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)* (26 May 2019, Montreal, QC, Canada).– IEEE.– 2019.– Pp. 17–23. doi ↑45, 48

[21] Zhou Z. Q., Sun L. *Metamorphic testing of driverless cars* // Communications of the ACM.– 2019.– Vol. **62**.– No. 3.– Pp. 61–67. doi ↑45, 48, 53

[22] Nakajima S., Chen T. Y. *Generating biased dataset for metamorphic testing of machine learning programs* // *Testing Software and Systems*, 31st IFIP WG 6.1 International Conference, ICTSS 2019 (15-17 October 2019, Paris, France).– Springer.– 2019.– Pp. 56–64. doi ↑45, 48

[23] Wang W., Huang J.-t., Wu W., Zhang J., Huang Y., Li S., He P., Lyu M. R. *Mttm: Metamorphic testing for textual content moderation software* // *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*.– IEEE.– 2023.– Pp. 2387–2399. doi ↑45, 48

[24] Srinivasan M., Shahri M. P., Kahanda I., Kanewala U. *Quality assurance of bioinformatics software: a case study of testing a biomedical text processing tool using metamorphic testing* // *Proceedings of the 3rd International Workshop on Metamorphic Testing* (27 May 2018-03 June 2018, Gothenburg, Sweden), New York: ACM.– ISBN 978-1-4503-5729-6.– Pp. 26–33. doi ↑45, 48

[25] Troup M., Yang A., Kamali A. H., Giannoulatou E., Chen T. Y., Ho J. W. *A cloud-based framework for applying metamorphic testing to a bioinformatics pipeline* // *Proceedings of the 1st International Workshop on Metamorphic Testing* (14-22 May 2016, Austin, Texas), New York: ACM.– 2016.– ISBN 978-1-4503-4163-9.– Pp. 33–36. doi ↑45, 50

[26] Méndez M., Benito-Parejo M., Ibias A., Núñez M. *Metamorphic testing of chess engines* // Information and Software Technology.– 2023.– Vol. **162**.– id. 107263. doi ↑45

[27] Zhou Z. Q., Sun L., Chen T. Y., Towey D. *Metamorphic relations for enhancing system understanding and use* // IEEE Transactions on Software Engineering.– 2018.– Vol. **46**.– No. 10.– Pp. 1120–1154. doi ↑45, 46, 47, 48, 49

[28] Zhang J., Zheng Z., Yin B., Qiu K., Liu Y. *Testing graph searching based path planning algorithms by metamorphic testing // 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)* (01-03 December 2019, Kyoto, Japan).– IEEE.– 2019.– id. 158.– 9 pp. doi ↑45, 49

[29] Xie X., Zhang Z., Chen T. Y., Liu Y., Poon P.-L., Xu B. *METTLE: A METamorphic testing approach to assessing and validating unsupervised machine learning systems* // IEEE Transactions on Reliability.– 2020.– Vol. **69**.– No. 4.– Pp. 1293–1322. doi ↑45, 47

[30] Guderlei R., Mayer J. *Statistical metamorphic testing testing programs with random output by means of statistical hypothesis tests and metamorphic testing // Seventh International Conference on Quality Software (QSIC 2007)* (11-12 October 2007, Portland, OR, USA).– IEEE.– 2007.– Pp. 404–409. doi ↑45, 52

[31] Xiao D., Liu Z., Yuan Y., Pang Q., Wang S. *Metamorphic testing of deep learning compilers* // Proceedings of the ACM on Measurement and Analysis of Computing Systems.– 2022.– Vol. **6**.– No. 1.– id. 15.– 28 pp. doi ↑45, 50

[32] Sun C.-a., Wang Z., Wang G. *A property-based testing framework for encryption programs* // Frontiers of Computer Science.– 2014.– Vol. **8**.– Pp. 478–489. doi ↑46

[33] Mouha N., Raunak M. S., Kuhn D. R., Kacker R. *Finding bugs in cryptographic hash function implementations* // IEEE Transactions on Reliability.– 2018.– Vol. **67**.– No. 3.– Pp. 870–884. doi ↑46

[34] Iakusheva S., Khritankov A. *Composite metamorphic relations for integration testing // Proceedings of the 2022 8th International Conference on Computer Technology Applications* (12-14 May 2022, Vienna, Austria), New York: ACM.– 2022.– ISBN 978-1-4503-9622-6.– Pp. 98–105. doi ↑46, 53

[35] Iqbal M. *Metamorphic testing of advanced driver-assistance systems: Implementing Euro NCAP standards on OpenStreetMap // 2023 IEEE/ACM 8th International Workshop on Metamorphic Testing (MET)* (14 May 2023, Melbourne, Australia).– IEEE.– 2023.– Pp. 1–8. doi ↑46, 49

[36] Luo G., Zheng X., Liu H., Xu R., Nagumothu D., Janapareddi R., Zhuang E., Liu X. *Verification of microservices using metamorphic testing // Algorithms and Architectures for Parallel Processing*.– V. I, 19th International Conference, ICA3PP 2019 (9-11 December 2019, Melbourne, VIC, Australia).– Springer.– 2020.– Pp. 138–152. doi ↑46

[37] Segura S., Parejo J. A., Troya J., Ruiz-Cortés A. *Metamorphic testing of RESTful web APIs // Proceedings of the 40th International Conference on Software Engineering* (27 May 2018–3 June 2018, Gothenburg, Sweden), New York: ACM.– 2018.– ISBN 978-1-4503-5638-1.– Pp. 882. doi ↑46

[38] Brown J., Zhou Z. Q., Chow Y.-W. *Metamorphic testing of navigation software: A pilot study with Google Maps*, University of Wollongong Research Online.– 2018.– 12 pp. URL ↑46, 49

[39] Jia M., Wang X., Xu Y., Cui Z., Xie R. *Testing machine learning classifiers based on compositional metamorphic relations* // International Journal of Performability Engineering.– 2020.– Vol. **16**.– No. 1.– Pp. 67–77. doi ↑47, 53

[40] Saha P., Kanewala U. *Fault detection effectiveness of metamorphic relations developed for testing supervised classifiers // 2019 IEEE International Conference On Artificial Intelligence Testing (AITest)* (04-09 April 2019, Newark, CA, USA).– IEEE.– 2019.– Pp. 157–164. ↑47

[41] Shahri M. P., Srinivasan M., Reynolds G., Bimczok D., Kahanda I., Kanewala U. *Metamorphic testing for quality assurance of protein function prediction tools // 2019 IEEE International Conference On Artificial Intelligence Testing (AITest)* (04-09 April 2019, Newark, CA, USA).– IEEE.– 2019.– Pp. 140–148. ↑47

[42] Dwarakanath A., Ahuja M., Sikand S., Rao R. M., Bose R. J. C., Dubash N., Podder S. *Identifying implementation bugs in machine learning based image classifiers using metamorphic testing // Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis* (16-21 July 2018, Amsterdam, Netherlands), New York: ACM.– 2018.– ISBN 978-1-4503-5699-2.– Pp. 118–128. ↑47, 48

[43] Zhu H., Liu D., Bayley I., Harrison R., Cuzzolin F. *Datamorphic testing: A methodology for testing ai applications*.– 2019.– 39 pp. arXiv:1912.04900 ↑47

[44] Zhang M., Zhang Y., Zhang L., Liu C., Khurshid S. *DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems // Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering* (3-7 September 2018, Montpellier, France), New York: ACM.– 2018.– ISBN 978-1-4503-5937-5.– Pp. 132–142. ↑47

[45] Raif M., Ouafiq E.-M., El Rharras A., Chehri A., Saadane R. *Metamorphic testing for edge real-time face recognition and intrusion detection solution // 2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)* (26-29 September 2022, London, United Kingdom).– IEEE.– 2022.– Pp. 1–5. ↑47

[46] Zhang Z., Wang P., Guo H., Wang Z., Zhou Y., Huang Z. *DeepBackground: Metamorphic testing for Deep-Learning-driven image recognition systems accompanied by Background-Relevance* // Information and Software Technology.– 2021.– Vol. **140**.– id. 106701. ↑47

[47] Xu L., Towey D., French A. P., Benford S., Zhou Z. Q., Chen T. Y. *Enhancing supervised classifications with metamorphic relations // Proceedings of the 3rd International Workshop on Metamorphic Testing* (27 May 2018-03 June 2018, Gothenburg, Sweden).– 2018.– Pp. 46–53. ↑47

[48] Yan R., Wang S., Yan Y., Gao H., Yan J. *Stability evaluation for text localization systems via metamorphic testing* // Journal of Systems and Software.– 2021.– Vol. **181**.– id. 111040. ↑47, 48

[49] Park H., Waseem T., Teo W. Q., Low Y. H., Lim M. K., Chong C. Y. *Robustness evaluation of stacked generative adversarial networks using metamorphic testing // 2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)* (02 June 2021, Madrid, Spain).– IEEE.– 2021.– Pp. 1–8. ↑48

[50] Ma P., Wang S., Liu J. *Metamorphic testing and certified mitigation of fairness violations in NLP models*, IJCAI'20: Twenty-Ninth International Joint Conference on Artificial Intelligence (7-15 January 2021, Yokohama, Japan).– 2021.– Pp. 458–465.– id. 64. ↑48

[51] Chen S., Jin S., Xie X. *Validation on machine reading comprehension software without annotated labels: a property-based method // Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (23-28 August 2021, Athens, Greece), New York: ACM.– 2021.– ISBN 978-1-4503-8562-6.– Pp. 590–602. ↑48

[52] Sun L., Zhou Z. Q. *Metamorphic testing for machine translations: MT4MT // 2018 25th Australasian Software Engineering Conference (ASWEC)* (26-30 November 2018, Adelaide, SA, Australia).– IEEE.– 2018.– Pp. 96–100. ↑48

[53] Gao W., He J., Pham V.-T. *Metamorphic testing of machine translation models using back translation // 2023 IEEE/ACM International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest)* (15 May 2023, Melbourne, Australia).– IEEE.– 2023.– Pp. 1–8. ↑48

[54] Pesu D., Zhou Z. Q., Zhen J., Towey D. *A Monte Carlo method for metamorphic testing of machine translation services // Proceedings of the 3rd International Workshop on Metamorphic Testing* (27 May 2018, Gothenburg, Sweden), New York: ACM.– 2018.– ISBN 978-1-4503-5729-6.– Pp. 38–45. ↑48, 52

[55] Sun Y., Ding Z., Huang H., Zou S., Jiang M. *Metamorphic testing of relation extraction models* // Algorithms.– 2023.– Vol. **16**.– No. 2.– Pp. 102. ↑49

[56] Raunak M. S., Olsen M. M. *Metamorphic testing on the continuum of verification and validation of simulation models // 2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)* (02 June 2021, Madrid, Spain).– IEEE.– 2021.– Pp. 47–52. ↑49

[57] Donaldson A. F., Lascu A. *Metamorphic testing for (graphics) compilers // Proceedings of the 1st International Workshop on Metamorphic Testing* (14-22 May 2016, Austin, Texas), New York: ACM.– 2016.– ISBN 978-1-4503-4163-9.– Pp. 44–47. ↑50

[58] Le V., Afshari M., Su Z. *Compiler validation via equivalence modulo inputs* // ACM Sigplan Notices.– 2014.– Vol. **49**.– No. 6.– Pp. 216–226. ↑50

[59] Zhou Z. Q., Tse T., Witheridge M. *Metamorphic robustness testing: Exposing hidden defects in citation statistics and journal impact factors* // IEEE Transactions on Software Engineering.– 2019.– Vol. **47**.– No. 6.– Pp. 1164–1183. ↑52

[60] Ahlgren J., Berezin M., Bojarczuk K., Dulskyte E., Dvortsova I., George J., Gucevska N., Harman M., Lomeli M., Meijer E., Sapora S. *Testing web enabled simulation at scale using metamorphic testing // 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (25-28 May 2021, Madrid, Spain).– IEEE.– 2021.– ISBN 978-1-6654-3869-8.– Pp. 140–149. ↑53

[61] Rehman F. u., Izurieta C. *Statistical metamorphic testing of neural network based intrusion detection systems // 2021 IEEE International Conference on Cyber Security and Resilience (CSR)* (26-28 July 2021, Rhodes, Greece).– IEEE.– 2021.– Pp. 20–26. ↑53

[62] Tambon F., Antoniol G., Khomh F. *HOMRS: High order metamorphic relations selector for deep neural networks.*– 2021.– 33 pp. arXiv:2107.04863 ↑53

[63] Zhang P., Zhou X., Pelliccione P., Leung H. *RBF-MLMR: A multi-label metamorphic relation prediction approach using RBF neural network* // IEEE Access.– 2017.– Vol. **5**.– Pp. 21791–21805. ↑53

[64] Zhang J., Chen J., Hao D., Xiong Y., Xie B., Zhang L., Mei H. *Search-based inference of polynomial metamorphic relations* // *ASE '14: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering* (15-19 September 2014, Vasteras, Sweden), New York: ACM.– 2014.– ISBN 978-1-4503-3013-8.– Pp. 701–712. ↑53

[65] Sun C.-A., Fu A., Poon P.-L., Xie X., Liu H., Chen T. Y. *Metric $^{+}$+: A metamorphic relation identification technique based on input plus output domains* // IEEE Transactions on Software Engineering.– 2019.– Vol. **47**.– No. 9.– Pp. 1764–1785. ↑53

[66] Kanewala U., Bieman J. M., Ben-Hur A. *Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels* // Software: Testing, Verification and Reliability.– 2016.– Vol. **26**.– No. 3.– Pp. 245–269. ↑53

[67] Blasi A., Gorla A., Ernst M. D., M. Pezzè, Carzaniga A. *MeMo: Automatically identifying metamorphic relations in Javadoc comments for test automation* // Journal of Systems and Software.– 2021.– Vol. **181**.– id. 111041. ↑53

[68] Wu P. *Iterative metamorphic testing* // *29th Annual International Computer Software and Applications Conference (COMPSAC'05)*.– V. 2 (26-28 July 2005, Edinburgh, UK).– IEEE.– 2005.– ISBN 0-7695-2413-3.– Pp. 19–24. ↑53

[69] Yang Y., Li Z., Wang H., Xu C., Ma X. *Towards effective metamorphic testing by algorithm stability for linear classification programs* // Journal of Systems and Software.– 2021.– Vol. **180**.– id. 111012. ↑53

[70] Xie X., Wong W. E., Chen T. Y., Xu B. *Metamorphic slice: An application in spectrum-based fault localization* // Information and Software Technology.– 2013.– Vol. **55**.– No. 5.– Pp. 866–879. ↑53

[71] Liu H., Liu X., Chen T. Y. *A new method for constructing metamorphic relations* // *2012 12th International Conference on Quality Software* (27-29 August 2012, Xi'an, China).– IEEE.– 2012.– Pp. 59–68. ↑53

[72] Qiu K., Zheng Z., Chen T. Y., Poon P.-L. *Theoretical and empirical analyses of the effectiveness of metamorphic relation composition* // IEEE Transactions on software engineering.– 2020.– Vol. **48**.– No. 3.– Pp. 1001–1017. ↑53

[73] Hui Z.-W., Huang S., Chua C., Chen T. Y. *Semiautomated metamorphic testing approach for geographic information systems: An empirical study* // IEEE Transactions on Reliability.– 2019.– Vol. **69**.– No. 2.– Pp. 657–673. ↑53

[74] Iakusheva S., Khritankov A. *Metamorphic testing for recommender systems* // *Analysis of Images, Social Networks and Texts*, AIST 2023, Lecture Notes in Computer Science.– vol. **14486**, eds. Ignatov D.I. et al., Cham: Springer.– 2024.– ISBN 978-3-031-54533-7. ↑53

[75] Hui Z.-w., Wang X., Huang S., Yang S. *MT-ART: A test case generation method based on adaptive random testing and metamorphic relation* // IEEE Transactions on Reliability.– 2021.– Vol. **70**.– No. 4.– Pp. 1397–1421. ↑53

[76] Sobania D., Briesch M., Röchner P., Rothlauf F. *MTGP: Combining metamorphic testing and genetic programming*, Genetic Programming. EuroGP 2023, Lecture Notes in Computer Science.– vol. **13986**, eds. Pappa G., Giacobini M., Vasicek Z., Cham: Springer.– 2023.– ISBN 978-3-031-29572-0.– Pp. 324–338. ↑53

Поступила в редакцию 22.11.2023;
одобрена после рецензирования 30.03.2024;
принята к публикации 31.03.2024;
опубликована онлайн 14.05.2024.

Рекомендовал к публикации *Анд. В. Климов*

Информация об авторах:

Софья Федоровна Якушева

аспирант, ассистент кафедры алгоритмов и технологий программирования Московского физико-технического института. Научные интересы: разработка и верификация программного обеспечения, тестирование инвариантами, машинное обучение.

0009-0000-1423-1123
e-mail: yakusheva.sf@phystech.edu

Антон Сергеевич Хританков

к.ф-м.н., доцент (Московский физико-технический институт), доцент (Высшая школа экономики). Научные интересы: методология программной инженерии, машинное обучение, доверенный искусственный интеллект.

0000-0003-2889-9436
e-mail: akhritankov@hse.ru

Вклад авторов: *С. Ф. Якушева* – 70% (методология, формальный анализ, расследование, сбор материала, написание черновой версии, доработка и редактирование, визуализация); *А. С. Хританков* – 30% (идея, методология, валидация, доработка и редактирование, наставничество, администрирование).

Декларация об отсутствии личной заинтересованности: *благополучие авторов не зависит от результатов исследования.*

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Review Article OPTIMIZATION METHODS AND CONTROL THEORY

UDC 004.415.53
10.25209/2079-3316-2024-15-2-37-86

A systematic review of methods for deriving metamorphic relations

Sofia Fedorovna **Iakusheva**[1], Anton Sergeevich **Khritankov**[2]

[1, 2] Moscow Institute of Physics and Technology, Moscow, Russia
[2] Higher School of Economics, Moscow, Russia

Abstract. Metamorphic testing is one of the most effective methods of testing programs with the test oracle problem. This problem declares that it is impossible to know whether the test answer is correct for one reason or another. Metamorphic testing uses metamorphic relations to check the program correctness. Metamorphic relation is a function of several test inputs and corresponding outputs of the program. Developing metamorphic relations can be a non-trivial task.

This systematic review is dedicated to identifying general derivation techniques for metamorphic relation as well as techniques pertinent to particular domains. As a result, we propose a classification of techniques into six main types and compile a comparative table of input data transformations for testing tasks in different domains. Findings of this review will help researchers to apply metamorphic testing in practice. (*Linked article texts in Russian and in English*).

Key words and phrases: metamorphic testing, metamorphic relation, software testing, test oracle problem

2020 *Mathematics Subject Classification*: 97P99; 97U99

For citation: Sofia F. Iakusheva, Anton S. Khritankov. *A systematic review of methods for deriving metamorphic relations*. Program Systems: Theory and Applications, 2024, **15**:2(61), pp. 37–86. (*In Russian, in English*).
https://psta.psiras.ru/read/psta2024_2_37-86.pdf

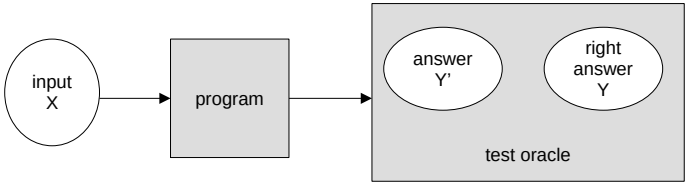


FIGURE 1. An example of checking the test. The role of the oracle is to compare the answer with the right one.

Introduction

Software quality control is a crucial part of the development process. At this stage, one verifies that the software meets all the set requirements, which means it will behave in an expected and previously known manner. Software deviations from the stated requirements regularly become the causes of aircraft crashes, self-driving car accidents, failures of space missions, and data security breaches [1]. Non-compliance with the requirements is a good reason for modifying the developed system or decommissioning the one already in use.

Software testing is one of the methods for quality monitoring during the software development process. Software testing identifies incorrect program behavior, violations of functional and non-functional requirements, and usage scenarios not provided in the documentation. A test oracle [2] is a function that determines the correctness of a program's response. The simplest test oracle compares the program response with a previously known answer (see Figure 1).

A more complex example could be an oracle that checks the correctness of the Hamiltonian path in a graph: it is enough to check the inclusion of all vertices of the graph in the path with the connecting edges; this test does not require an answer known a priori. In poorly formalized cases, if the requirements are inaccurate or a high level of expertise is necessary, then a user, an expert, or another program can be taken as a test oracle.

When testing knowledge-intensive, non-deterministic, distributed software systems, the so-called test oracle problem became a significant

obstacle. The test oracle problem [3] is the difficulty in devising test oracles, especially automatic ones. The problem is relevant in situations that require exhaustive search to find an exact solution. Such situations appear in bioinformatics and combinatorics, in machine learning tasks due to the costly process of test data collection and labeling or tasks of creating artistic objects due to the need for human evaluation of conformance to the poorly stated requirements. However, there are various methods to approach this problem. Property-based testing, for instance, checks the fulfillment of a given relationship between the inputs and outputs of a program. Metamorphic testing is a variation of this method.

1. Metamorphic testing

1.1. Definitions and examples

Metamorphic testing [4] is used in many areas and has several development directions [5, 6]. The idea of the method is to check a metamorphic relation between stimuli and responses instead of the correctness of each specific program response. A metamorphic relation is a function calculated on several program inputs and outputs:

$$R(x_1, x_2, \ldots, x_n, f(x_1), f(x_2), \ldots, f(x_n)) \longrightarrow \{0, 1\}, \tag{1}$$

where $n \geqslant 2$ denotes the number of runs in the test case, x_i — input data for the i-th run in the test case, $f(x_i)$ — the i-th output. The metamorphic testing technique consists of the following steps. First, we set the procedure for obtaining input data which can involve using a data generator. Next, we run the program on generated inputs in the test case and check the if the relation holds. If there are input data for which R is zero, then there are errors or non-compliances with the requirements. Thus, instead of directly checking the answers, the method uses the so-called derived test oracle [6]. The construction of such a relation for a number of problems can naturally follow from the properties of the problem solved by the program and can be simpler for a program developer or researcher.

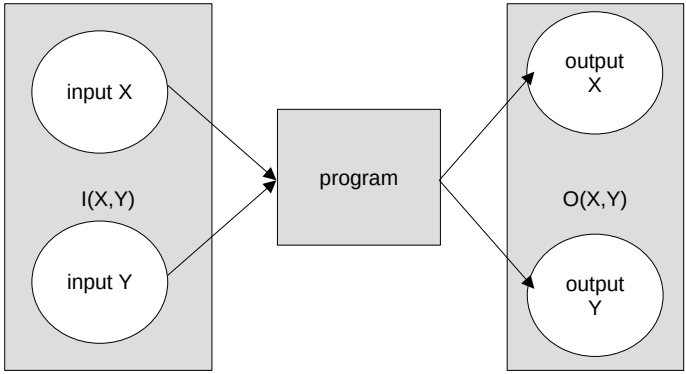


FIGURE 2. A simple example of metamorphic testing. The metamorphic relation is a function $R(x, y, X, Y) = I(x, y) \cap O(X, Y)$. Presence of dependence $I(x, y)$ between test inputs should cause the presence of the dependence $O(X, Y)$ between outputs.

A common metamorphic relation has the following statement. If two elements of the input data sequence satisfy some relation I, then the two corresponding program responses must satisfy the relation O (see Figure 2):

$$R(x_1, x_2, \ldots, x_n, f(x_1), f(x_2), \ldots, f(x_n)) = \\ = I(x_1, x_2, \ldots, x_n) \cap O(f(x_1), f(x_2), \ldots, f(x_n)), \tag{2}$$

where I is the transformation of the input data, O is the expected relationship between the outputs, $n \geqslant 2$ is the total number of test runs in the test case, x_i is the input data for the i-th run in the test case, and $f(x_i)$ is the i-th program output (see Figure 2). We will consider relations under the conditions that test runs are independent and that the initial state of the program in each run is known.

As an example, let us describe several metamorphic relations for checking the correct execution of queries to a relational database containing a table $T = a|b|c|...$, where a, b, c are the table columns.

- Let A and B be two search conditions. Then the answer of a query with condition $A \cap B$ is a subset of the answer of a query with condition A (similar to B).
- Let A be a search condition. The answers to the query with condition A and the query with condition $\neg A$ do not intersect.

- Let A be a search condition. The number of answers to a query with condition A when sorting in ascending order of column a and when sorting in descending order of column a is the same.

1.2. Terminology

Metamorphic testing is rarely discussed in Russian scientific literature, so it does not have a common terminology in Russian. Some researchers use the term *метаморфное тестирование* which is the literal translation of the English term *metamorphic testing*.

Authors use terms *тестирование инвариантами* (*invariant testing*) for metamorphic testing and *тестовый инвариант* (*test invariant*) for metamorphic relation in Russian text. Such terms are much easier to understand and to remember due to the simple close relation with the definition. In mathematical literature, invariant means a value or a property that is constant in the situation under observation. Test invariant usually means a function that preserves its value if the parameters change as expected. In physics, an invariant has a similar meaning, for example, in the phrase *time translation invariance*. The formula 1.1 defines such a function, so by meaning *metamorphic relation* we assume a *test invariant*. Moreover, a *test invariant* can be derived as a corollary from a mathematical model of the problem solved by the program. So the used terms *тестовый инвариант* and *тестирование инвариантами* are suitable translations from English.

2. Research goals and methods

At the current stage of the development of metamorphic testing, one of the unsolved problems is developing a methodology for devising metamorphic relations. A widely applicable method is still missing for programs that solve different applied problems in many areas. This leads to a effort-intensive compilation of metamorphic relations practically from scratch in each specific case.

In this study, we set the following tasks to overcome difficulties in devising metamorphic relations.

(*1*) Identify common and reusable techniques (patterns) for devising metamorphic relations in different problem areas. The results are given in Section 4.

(2) Determine specific traits of metamorphic testing for machine learning and data analysis programs that may allow for constructing more effective relations. More details are in Section 5.

(3) Identify indirect methods for obtaining the relations, as well as ways to combine relations with other methods in order to simplify the use of metamorphic testing in non-standard cases. The results are collected in Section 7.

(4) Identify omissions and opportunities for further development of methods for devising metamorphic relations. The results are given in Section 3 and Section 6.

We apply a systematic review method [7] to achieve the goals of the study. B. Kitchenham proposed this method for conducting meta-research (secondary study) in software engineering. This method can improve the reproducibility of the results and the validity of the conclusions.

To select publications, the following criteria were used: novelty, publication in the period from 2018 to 2023, accessibility to the reader, relevance to the research topic, detailed and clear description of the test invariants used, the presence of practical results in the work, and the possibility of generalizing the ideas used and techniques. The quality of the proposed invariants was not considered a search criterion, since the total number of studies on the topic is not enough to apply statistical methods for determining the average effectiveness of a particular invariant.

In order to outline the basic results of invariant testing, the review also includes seminal works in each area, some of which were published before 2018. To select them, the following criteria were used: relevance to the topic, citation, depth and versatility of the proposed ideas and methods, and practical value. The search followed the same rules, but no publication date restriction was applied. During the search, lists of sources from the most famous English-language review articles [4, 6] on this topic were additionally used. Also, seminal works have been replaced by sources that cite and reuse test invariants from these earlier works without significant modifications.

We searched for publications in Google Scholar in English using the keywords “metamorphic testing”, “metamorphic relation”, “neural network”, “classification”, “maps”, “graphs”, “natural language processing” and combinations of these words. We identify some additional publications by reviewing the reference lists from these publications. The presence of the word “metamorphic” in the source text is necessary since there are no alternative names for this method in the English literature. The search excluded patents and citations, review articles on related fields and various testing methods, and articles that only mentioned usage of metamorphic testing. The search was conducted from June 14 to September 13, 2023.

We found more than two thousand publications using the keywords “metamorphic testing” for the period from 2019 to 2023. This fact indicates the relevance and interest of researchers in this topic. We selected and studied about 120 papers and included 66 of them in this study.

We also searched for Russian-language publications in eLibrary. We found only one work on the topic, a short paper [8], and one article from a related field of verification of flowcharts [9].

We reviewed a description of the subject area, the solved problem, and the relations proposed in each publication. Then, we grouped this information by subject areas and identified the most popular ideas and techniques in each area. We also selected a brief description of the method from the reviewed articles that describe techniques for devising the relations.

Finally, we generalize the selected approaches. At this stage, we form a classification of methods for developing metamorphic methods based on input data transformations. Using this classification, we describe patterns for developing relations that can be applied in many areas.

3. Classification of metamorphic relations

An analysis of the selected metamorphic relations showed that they could be rewritten in the same form (2). We formulated a classification of metamorphic relations based on the types of input data transformations

Table 1. Classification by transformation of input data

Transformation	Meaning
LT linear transform	linear transformation (including increasing or decreasing, multiplication by a constant)
SM symmetry	reflectional and rotational symmetries, symmetrical relations and others
PR permutative	permutation of elements of initial data
IE inclusive / exclusive	inclusion/exclusion of elements from a set
BC blur / change	replacement or modification, adding random noise
UD unite / divide	merging or splitting input data

I (see table 1). These transformations allow us to obtain the input data $I(x_1, x_2, \ldots, x_n)$ for metamorphic relations.

The classification we describe does not include a few rare metamorphic relations tied to a specific problem so that their generalization is currently of no interest. A few examples of such relations are described in section 4.2.

4. Patterns for devising metamorphic relations

In this section, when we describe the methods used in the literature for compiling metamorphic relations, we will immediately indicate in parentheses, which of the previously identified classes it belongs to. For example, multiplying numerical parameters by a non-zero constant can be classified as linear transformations (LT).

4.1. Common techniques

Popular techniques for testing models and algorithms are changing parameters, rearranging equivalent parameters, and adding noise.

Often, researchers use a parameter change (LT) to clearly predict the change in the result (whether it will improve, worsen, change by a known amount, or not change at all). Examples include changing the number

of computing devices [10], changing the parameters of the epidemiological model [11, 12], parameters and hyperparameters of neural network models [13].

Permutations of equivalent parameters or inputs (PR) are a popular method. Frequently, one can expect the same output of the program execution. For example, permutation of users of cloud services [10], operations performed under multi-threading conditions [14], and input data in stochastic optimization [15] should not affect the program output. The order in which one adds requirements when booking a hotel using a bot [16] should not affect the search results.

Adding noise (BC) is a widely used technique: errors [17, 18], noise [19–21], distortions [22], use of language features [23]. Usually, researchers expect that the added noise will not change the output, or at least will not improve it. If one replaces the original data (BC), the result usually be expected to change. For example, we can cite the already mentioned relation from section 1 with the conditions A and $\neg A$. With such a change in the input, the output should completely change and not intersect with the previous one.

Adding or removing parts of the source data (UD) is less common. For example, adding new users of cloud services [10], adding and removing language units from the text [24], parts of the input data [18, 25].

The technique of using symmetrical inputs is fruitful in the areas of imagery and geoinformatics. For example, reflection of a chessboard [26] when testing chess playing algorithms, reflection of images relative to the axes [27], time reversal of video sequences [27], rotations and reflections of card images [28], reflections of detected data clusters over a line [29].

4.2. Using the properties of the mathematical model and the subject area

One can use the properties of a mathematical model of the problem or a program to obtain metamorphic relations. For example, Guderlei et al. [30] compare the sampling results to the given statistical distribution. Researchers test deep neural network compilers [31] by adding source code that leads an equivalent program but changes the resulting computational graph (IE). Both the properties of the chessboard and the ranking of the pieces are used to test chess algorithms [26], for example, reflections of the board (SM), replacement of the pieces with ones of a different color (PR),

or stronger or weaker ones (BC). In article [32], the authors propose to apply matrix transformations (LT) and modulo operations (LT) to construct metamorphic relations for testing hashing algorithms that use matrix transformations and the RSA algorithm. Removing particular bits in the message (IE) and splitting the message into parts (UD) are applied to test another hashing algorithm [33]. In these cases, the hash value should change after transformations.

Sometimes, domain features are useful for constructing metamorphic relations. Testing a comparative genetic analysis system [34] uses the ability to separate mutations into non-overlapping classes. The work on testing driver assistant systems [35] uses some considerations about labeled maps; for example, highways and buildings do not have common areas, and forbidden parts of some paths do not intersect other paths. The work on testing an application with a microservice architecture [36] uses the features of the payment system. For example, the sum of creditor and debtor balances should remain the same after moving data from one microservice to another.

5. Methods for devising metamorphic relations in different areas

5.1. Search algorithms

When testing search algorithms, one checks if the program output changes in the expected way after adding or removing search conditions (IE). New elements should not appear in the output after adding a new condition [36, 37], and old ones should not disappear after removing a condition. A paper on testing the hotel booking bot uses changing the order of adding search terms (PR) [16]. Another paper [27] uses symmetry: search results sorted in descending order can be obtained from the results sorted in ascending order after reversing (SM). The product price should not depend on the search query (BC) [27], and the search results should be the same for different languages (BC) [27] and for users around the world (BC). Such a relation would test if outputs differ due to the inconsistencies among different database replicas in a distributed system.

5.2. Machine learning and data analysis

Due to the complexity of testing machine learning systems and data analysis algorithms, researchers propose many interesting techniques

and transformations of input data. One of the common techniques is a repartition of the training dataset (UD). For example, adding or excluding an element from the data sample [29, 39, 40], adding or excluding entire classes [39, 40], duplicating elements of the training set [39], changing labels of data elements (BC)[40], addition and exclusion of features (IE) [29, 39, 40]. The exclusion of non-important features should not significantly change the classification quality, and the exclusion of significant ones should not improve. Permutations (PR) of features [39, 40], class labels [39, 40], elements of the training set [29], as a rule, should not significantly change the final quality of the classifier. A paper on testing prediction tools for protein properties [41] uses transformations of proteins that are guaranteed to change their function (BC), and the model should also change the prediction.

Linear transformations (LT) are less popular in this area. One can use the application of an affine transformation to sample features [40], rotations, and transformations of the coordinate system in space [29] (such transformations should not significantly affect the program output). Xie et al. [29] propose several more techniques for testing clusterization algorithms: compressing selected clusters to their centers (LT), adding elements inside the convex hull of clusters (UD), and adding outliers (UD). The idea of symmetry (SM) results as a technique of reflecting clusters over a line [29].

5.3. Deep learning and neural networks

One can use the methods proposed for testing machine learning models to test deep neural networks. For example, linear feature transformation (LT) [13, 19, 42]), permutation of class labels (PR) [13], repartition of data into train and test samples (UD) [13], data duplication (UD) [13], increasing or decreasing the learning algorithm parameters (LT) (e.g. learning rate, the α parameter in the nonlinear activation function ReLU, the number of training epochs, the number of neurons in layers [13]).

Another set of techniques is applied for testing image processing algorithms. Checking for quality of facial recognition uses changing the hair color, eyebrows, and sex, and adding glasses, mustache, and beard (BC) [43]. Testing of self-driving cars involves changing the weather (BC) [44, 45] or replacing the background (BC) [46] on the pictures. Xu et al. [47] propose the idea of using inequalities for the detection logits on the

image and on its parts (UD). Other techniques use symmetries (SM): reflection [27], video reversal [27], as well as perspective transformations (LT) [48] and brightness change (LT) [48]. A new idea is to add watermarks (BC) and masks (UD) [48]. Park et al. [49] add new objects to the images (UD) to test the model.

One can check robustness of the model with noise reduction (BC) [19–21] and data distortion (BC) [22]. Another technique is to use the permutations of elements of the training sample (PR) [42], training and target features (PR) [42], image channels [48].

5.4. Natural language processing

A connection between changes in the text and changes in its meaning or other parameters under consideration is unclear. Therefore, input data transformations that do not change the system output are the most popular for testing natural language processing systems. Thus, techniques for input data transformation use the spelling and grammar features of languages, for example, English and Chinese.

A common technique is to add a substitution that should not change the program output (BC). Replacement of a morphological or semantic entity property, for example, gender [50], is used to recognize discrimination. Robustness check uses the replacement of a word with its synonym, translation into another language or an unrelated word [16, 23, 51], replacement of symbols with similar ones (visually or phonetically) or with '*' [23]. Other transformations are voice change (BC), replacement of certain words, for example, «before» with «after» (BC), change in the order of sentence members (PR) [51], noise with typos (BC) [23], word skipping (IE) [16]. Wang et al. [23] propose several more ideas for systems to analyze the Chinese language: splitting words and characters into parts (UD), merging letter combinations (UD), rearranging letters in a word (PR), abbreviating to an acronym (UD), inserting special offers-indicators (IE) for recognition. Such transformations can be fruitful for assessing the robustness of the model.

In automatic translation, the replacement of a noun for another helps to determine translation consistency (BC) [52]. One can also use symmetries (SM): the source text and its back translation from another language [27, 53]; direct translation from language A to language B and translation from language A to language B via a third language C [54].

Researchers successfully apply metamorphic testing to named entity recognition systems. An article [24] on recognizing entities in the text uses

transformations that preserve existing entities: paragraph permutations (PR), permutations of a list of added random words (PR), and sentence merging. Adding and deleting random words, sentences, and paragraphs (UD) should not result in the disappearance of previously recognized entities. Sun et al. [55] use swapping entities (SM) and replacing entities (BC) to test a system that searches for relationships between entities. These transformations do not affect the existence of the relationship and use its symmetry.

5.5. Graph models. Geoinformatics

Graph models are used for various tasks like searching for routes [28] and modeling gene interactions [17]. One can structurally modify the graph model by adding or removing vertices and edges. For example, Chen et al. [17] consider a tool for modeling gene regulatory networks. They use transformations that obviously increase or decrease indicators, for example, changes in the edge or vertex parameter (LT), adding or removing vertices and edges with different properties (IE). The work [56] on testing a model that simulates the spread of news in a social network also uses changes in parameters with predictable results: changes in weights, types, and parameters of neighboring vertices, and others.

Important mathematical properties of the graph model for geographical data are the symmetry of the reachability and the triangle inequality. Based on them, several authros propose to swap the beginning and the end of the route (SM) [27, 28] and apply the triangle inequality to the lengths of routes A-C-B and A-B (UD) [28] and to their costs (UD) [38]. The triangle inequality also applies when adding and removing obstacles (IE) [28], adding conditions (IE) [38]. Some relations use robustness assumptions (BC): a small change in the starting points does not dramatically change the program output [38], and the same request after a short period of time results in the same output [27].

In cartography and route planning software, researchers additionally use transformations that change only the presentation of the map image. For example, changing the coordinate system [28] (LT), rotational symmetries, and reflecting the map over a line [28] (SM). As a result, such transformations do not change the output. When the scale of the image is changed, they change the path length proportionally.

Iqbal et al. [35] use domain-specific features of labeled maps. They check that each path has at least one start and one finish, every roundabout has an entrance and an exit, and the forbidden parts of some paths do not intersect other paths.

5.6. Bioinformatics (texts)

Metamorphic testing is applied to testing aligners. An aligner is a program that determines the coordinates of small sections of nucleic acids read by a sequencer (so-called reads) on the reference genome. If an aligner calculates the read coordinates successfully, this read is called mapped; otherwise, it is called unmapped.

The use of inverted reads [17, 18] is an example of reflections. Researchers also propose changing the order of reads in the input (PR) [18], permutation of letters in the alphabet (PR) [17], adding [18, 25] and deleting [18] reads (UD), using only mapped or unmapped reads (UD) [18, 25], reads expansion (IE) [18], adding and removing sets of reads (UD) [17], changing the maximum allowed number of mismatches [17]. Also, one can add [17] and remove [17, 18] errors in reads (BC), replace errors with errors of a different type (BC) [17]. These operations may require a comparison of reads with the corresponding regions of the reference genome.

5.7. Compilers

A compiler is a program that converts code written in a programming language into a set of machine codes. The main idea used to test compilers is to create a program equivalent to the original one. Donaldson et al. [57] add dead code and identity functions (IE). Le et al. [58] use program reduction (BC) through sophisticated code coverage analysis.

Nowadays, some specialized compilers transform a high-level deep neural network model into optimized executable code. Xiao et al. [31] add complex constructions to test such a compiler. These constructions create a duplicate of the initial program but change the resulting computation graph (IE).

6. Analysis of the application of methods for relation construction by areas

We compile a summary table 2 based on the input data transformations patterns. The table cells contain objects, parameters, and properties. The column indicates the method of input transformation, and the row indicates the area. One can note that these transformations often change the minimal indivisible units from the area under consideration, such as a vertex or an edge of a graph, a sample element, a language unit, or their representation in computer memory.

Table 2. Transformations of input data for metamorphic relations

Area	LT	SM	PR	IE	BC	UD
Search algorithms		output order	order of conditions	a condition	language, send position	
Machine learning	a scale, parameters	symmet-rical object	equivalent properties	properties	a label	a sample
Deep neural networks	properties, hyperparameters	symmet-rical object	properties, classes	new objects	an object, a background	a sample, an image
Geoinformatics	coordinate system	coordinate system, relations	a start and a finish of the path	obstacles, conditions	a point	paths
Graph models	an edge	a start and a finish of the path		vertices		paths
Natural language processing		languages, entities	language items	words, indicators	language and items	language items
Bioinformatics		reads, an alphabet	reads, an alphabet	reads, mutations, genes	reads, errors	reads
Compilers				syntax constructions	text of the program	

We were not able to find relations of all six identified types for all the considered areas, which may serve as a direction for further research. In addition, discovered relations have not yet used transformations of all possible units of the areas under consideration. For example, one can add noise to the capacities and other characteristics of vertices and edges in a graph (BC) and rearrange the edges themselves (PR) when storing the graph in the form of adjacency lists. In natural language processing, one can try to use the vector representation of words to apply linear transformations (LT). In object detection, collage images can help to compare results (UD). For search queries, one can change the filter parameter (LT) and compare the number of responses for different parameter ranges (UD). For example, the total number of responses equals the sum of the number of responses for the two halves of this range.

Thus, the resulting table shows the following algorithm for compiling relations. Researchers consider the minimal units in the subject area and apply transformations from the six specified groups. One can obtain a metamorphic relation if it is possible to say unambiguously what change in the output the transformation should lead to.

7. Other methods for metamorphic relations

In addition to obtaining relations directly from the problem statement or model properties, it is possible to make compositions of already existing metamorphic relations, combine them with the methods of mathematical statistics for application to stochastic systems, and combine them with other methods. These techniques also help to obtain new, sometimes more effective, relations, so let us consider them in more detail.

Application of statistical methods in metamorphic testing. Random errors may occur during the testing of stochastic systems with metamorphic testing, so one can use statistical approaches to verify the correctness of relations. One of the first works on statistics in metamorphic testing [**30**] considers testing the properties of a specific statistical distribution implementation as an example. More recent works use statistical methods to determine the fulfillment of relations (ANOVA [**54**], Spearman's rank correlation [**59**], a criterion for testing the presence of statistically significant differences [**60**, **61**]). The work on testing multi-armed bandits [**74**] proposes another generalization: the relation is defined as a composition of the procedure for obtaining a sample and a statistical criterion.

Search for relations. Relations are not always easily derived from the problem statement or model of the solution algorithm. In addition, this process is currently poorly automated. Therefore, a separate direction in metamorphic testing considers searching for approaches for metamorphic relations [62, 63]. Zhang et al. [64] propose an approach for searching metamorphic relations as polynomial functions, and Sun et al. [65] search the relations dynamically by categories of inputs and outputs. The work [66] proposes the SVM algorithm application on program graphs to predict relations. Blasi et al. [67] propose to search for descriptions in natural language that can be converted into metamorphic relations.

Special types and compositions of relations. One of the early works in this field [68] proposes iteratively selecting input data for test cases. The paper [69] develops this idea and applies it to testing the linear model. The output of the first test run is used to get the next one. The work [39] proposes the idea of multidimensional metamorphic relations. The work on spectrum-based fault localization [70] introduces the concept of a metamorphic slice.

An early work on relation composition [71] examines non-trivial composition functions and shows that the defect discovery effectiveness of the composition usually exceeds the total effectiveness of individual relations. Authors of [72] conduct a theoretical study of the possibilities of composition and discuss some examples. The article on checking geographic systems [73] uses relations compositions of the form z(y(x())) with 2-4 functions, and uses the simple relations intended to check various properties: test requirements, program properties, algorithm properties, etc. The work on testing a bioinformatic pipeline [34] proposes a method for compiling compositions of metamorphic relations for systems using logical functions.

Combinations with other methods. Metamorphic testing can be combined with fuzzing. hlZhou et al. [21] slightly distort the LIDAR data, which makes it possible to evaluate the system's robustness. An interesting technique is the use of metamorphic testing and adaptive random testing [75]: when generating a subsequent test, the distance to the three previous ones is measured. In addition, metamorphic testing is used in spectrum-based fault localization [70] and in combination with genetic algorithms [76].

Conclusion

This systematic review examines methods and patterns for constructing metamorphic relations in various fields of knowledge. We analyze the available literature, identify the most frequently applied techniques, and examine popular application areas of metamorphic testing in detail.

As a result of the study,

- We have identified six common types of input data transformations that are used in the formulation of metamorphic relations: linear transformations, symmetries, permutations, adding or removing elements, replacing or adding noise, and merging or splitting into parts.
- We have analyzed and compared relations for problems from different fields of knowledge;
- We have considered indirect methods for obtaining relations, such as creating compositions of existing ones, using statistical approaches, and combining them with other methods;
- We have obtained a summary table of the variable units in each area and identify opportunities for compiling new relations.

The results of this analysis can contribute to the development of a general theory of building metamorphic relations, the popularization of metamorphic testing, and the creation of new test automation technologies and applications. We hope our work will help researchers apply metamorphic testing more easily and quickly.

References

[1] W. E. Wong, V. Debroy, A. Surampudi, H. Kim, M. F. Siok. "Recent catastrophic accidents: Investigating how software was responsible", 2010 Fourth International Conference on Secure Software Integration and Reliability Improvement (09-11 June 2010, Singapore), 2010, pp. 14–22. ↑63

[2] W. Howden. "Theoretical and empirical studies of program testing", *IEEE Transactions on Software Engineering*, **SE-4**:4 (1978), pp. 293–298. ↑63

[3] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, S. Yoo. "The oracle problem in software testing: A survey", *IEEE Transactions on Software Engineering*, **41**:5 (2015), pp. 507–525. ↑64

[4] T. Y. Chen, S. C. Cheung, S. M. Yiu. *Metamorphic testing: a new approach for generating next test cases*, 2020, 11 pp. arXiv 2002.12543 ↑64, 67

[5] T. Y. Chen, T. Tse. "New visions on metamorphic testing after a quarter of a century of inception", *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 1487–1490. ↑64

[6] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. Tse, Z. Q. Zhou. "Metamorphic testing: A review of challenges and opportunities", *ACM Computing Surveys*, **51**:1 (2018), pp. 1–27. ↑64, 67

[7] B. Kitchenham. *Procedures for performing systematic reviews*, Keele University Technical Report TR/SE-0401, Keele University, Keele, UK, 2004, 33 pp. ↑67

[8] R. R. Fal'kovskij. "Metamorphic testing of image enhancement programs", *XIX Mezhdunarodnaya telekommunikacionnaya konferenciya molodyx uchenyx i studentov "MOLODEZh' I NAUKA"*, Tezisy dokladov. V. 3, NIYaU MIFI, M., 2015, ISBN 978-5-7262-2223-3, pp. 176–177 (in Russian). ↑68

[9] A. M. Mironov. "Verification of programs by the method of invariants", *Intellektual'nye sistemy. Teoriya i prilozheniya*, **21**:4 (2017), pp. 31–49 (in Russian). ↑68

[10] A. Núñez, P. C. Cañizares, M. Núñez, R. M Hierons. "TEA-Cloud: A formal framework for testing cloud computing systems", *IEEE Transactions on Reliability*, **70**:1 (2020), pp. 261–284. ↑70

[11] L. L. Pullum, O. Ozmen. "Early results from metamorphic testing of epidemiological models", *2012 ASE/IEEE International Conference on BioMedical Computing (BioMedCom)* (14-16 December 2012, Washington, DC, USA), IEEE, 2012, pp. 62–67. ↑70

[12] A. Ramanathan, C. A. Steed, L. L. Pullum. "Verification of compartmental epidemiological models using metamorphic testing, model checking and visual analytics", *2012 ASE/IEEE International Conference on BioMedical Computing (BioMedCom)* (14-16 December 2012, Washington, DC, USA), IEEE, 2012, pp. 68–73. ↑70

[13] J. D. Ellis, R. Iqbal, K. Yoshimatsu. "Verification of the neural network training process for spectrum-based chemical substructure prediction using metamorphic testing", *Journal of Computational Science*, **55** (2021), id. 101456. ↑70, 72

[14] C.-A. Sun, H. Dai, N. Geng, H. Liu, T. Y. Chen, P. Wu, Y. Cai, J. Wang. "An interleaving guided metamorphic testing approach for concurrent programs", *ACM Transactions on Software Engineering and Methodology*, **33**:1 (2023), id. 8, 21 pp. ↑70

[15] S. Yoo. "Metamorphic testing of stochastic optimisation", *2010 Third International Conference on Software Testing, Verification, and Validation Workshops* (06-10 April 2010, Paris, France), IEEE, 2010, pp. 192–201. ↑70

[16] J. Bozic, F. Wotawa. "Testing chatbots using metamorphic relations", *Testing Software and Systems*, 31st IFIP WG 6.1 International Conference, ICTSS 2019 (15-17 October 2019, Paris, France), Lecture Notes in Computer Science, vol. **11812**, eds. Gaston C., Kosmatov N., Le Gall P., Springer, Cham, 2019, ISBN 978-3-030-31279-4, pp. 41–55. ↑70, 71, 73

[17] T. Y. Chen, J. W. Ho, H. Liu, X. Xie. "An innovative approach for testing bioinformatics programs using metamorphic testing", *BMC bioinformatics*, **10** (2009), id. 24, 12 pp. doi ↑70, 74, 75

[18] E. Giannoulatou, S.-H. Park, D. T. Humphreys, J. W. Ho. "Verification and validation of bioinformatics software without a gold standard: a case study of BWA and Bowtie", *BMC bioinformatics*, **15** (2014), id. S15, 8 pp. doi ↑70, 75

[19] Y. Tian, K. Pei, S. Jana, B. Ray. "DeepTest: Automated testing of deep-neural-network-driven autonomous cars", *Proceedings of the 40th International Conference on Software Engineering* (27 May 2018-3 June 2018, Gothenburg, Sweden), ACM, New York, 2018, ISBN 978-1-4503-5638-1, pp. 303–314. doi ↑70, 72, 73

[20] C. Wu, L. Sun, Z. Q. Zhou. "The impact of a dot: Case studies of a noise metamorphic relation pattern", *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)* (26 May 2019, Montreal, QC, Canada), IEEE, 2019, pp. 17–23. doi ↑70, 73

[21] Z. Q. Zhou, L. Sun. "Metamorphic testing of driverless cars", *Communications of the ACM*, **62**:3 (2019), pp. 61–67. doi ↑70, 73, 78

[22] S. Nakajima, T. Y. Chen. "Generating biased dataset for metamorphic testing of machine learning programs", *Testing Software and Systems*, 31st IFIP WG 6.1 International Conference, ICTSS 2019 (15-17 October 2019, Paris, France), Springer, 2019, pp. 56–64. doi ↑70, 73

[23] W. Wang, J.-t. Huang, W. Wu, J. Zhang, Y. Huang, S. Li, P. He, M. R. Lyu. "Mttm: Metamorphic testing for textual content moderation software", *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, IEEE, 2023, pp. 2387–2399. doi ↑70, 73

[24] M. Srinivasan, M. P. Shahri, I. Kahanda, U. Kanewala. "Quality assurance of bioinformatics software: a case study of testing a biomedical text processing tool using metamorphic testing", *Proceedings of the 3rd International Workshop on Metamorphic Testing* (27 May 2018-03 June 2018, Gothenburg, Sweden), ACM, New York, ISBN 978-1-4503-5729-6, pp. 26–33. doi ↑70, 73

[25] M. Troup, A. Yang, A. H. Kamali, E. Giannoulatou, T. Y. Chen, J. W. Ho. "A cloud-based framework for applying metamorphic testing to a bioinformatics pipeline", *Proceedings of the 1st International Workshop on Metamorphic Testing* (14-22 May 2016, Austin, Texas), ACM, New York, 2016, ISBN 978-1-4503-4163-9, pp. 33–36. doi ↑70, 75

[26] M. Méndez, M. Benito-Parejo, A. Ibias, M. Núñez. "Metamorphic testing of chess engines", *Information and Software Technology*, **162** (2023), id. 107263. doi ↑70

[27] Z. Q. Zhou, L. Sun, T. Y. Chen, D. Towey. "Metamorphic relations for enhancing system understanding and use", *IEEE Transactions on Software Engineering*, **46**:10 (2018), pp. 1120–1154. doi ↑70, 71, 73, 74

[28] J. Zhang, Z. Zheng, B. Yin, K. Qiu, Y. Liu. "Testing graph searching based path planning algorithms by metamorphic testing", *2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC)* (01-03 December 2019, Kyoto, Japan), IEEE, 2019, id. 158, 9 pp. doi ↑70, 74

[29] X. Xie, Z. Zhang, T. Y. Chen, Y. Liu, P.-L. Poon, B. Xu. "METTLE: A METamorphic testing approach to assessing and validating unsupervised machine learning systems", *IEEE Transactions on Reliability*, **69**:4 (2020), pp. 1293–1322. doi ↑70, 72

[30] R. Guderlei, J. Mayer. "Statistical metamorphic testing testing programs with random output by means of statistical hypothesis tests and metamorphic testing", *Seventh International Conference on Quality Software (QSIC 2007)* (11-12 October 2007, Portland, OR, USA), IEEE, 2007, pp. 404–409. doi ↑70, 77

[31] D. Xiao, Z. Liu, Y. Yuan, Q. Pang, S. Wang. "Metamorphic testing of deep learning compilers", *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, **6**:1 (2022), id. 15, 28 pp. doi ↑70, 75

[32] C.-A. Sun, Z. Wang, G. Wang. "A property-based testing framework for encryption programs", *Frontiers of Computer Science*, **8** (2014), pp. 478–489. doi ↑71

[33] N. Mouha, M. S. Raunak, D. R. Kuhn, R. Kacker. "Finding bugs in cryptographic hash function implementations", *IEEE Transactions on Reliability*, **67**:3 (2018), pp. 870–884. doi ↑71

[34] S. Iakusheva, A. Khritankov. "Composite metamorphic relations for integration testing", *Proceedings of the 2022 8th International Conference on Computer Technology Applications* (12-14 May 2022, Vienna, Austria), ACM, New York, 2022, ISBN 978-1-4503-9622-6, pp. 98–105. doi ↑71, 78

[35] M. Iqbal. "Metamorphic testing of advanced driver-assistance systems: Implementing Euro NCAP standards on OpenStreetMap", *2023 IEEE/ACM 8th International Workshop on Metamorphic Testing (MET)* (14 May 2023, Melbourne, Australia), IEEE, 2023, pp. 1–8. doi ↑71, 74

[36] G. Luo, X. Zheng, H. Liu, R. Xu, D. Nagumothu, R. Janapareddi, E. Zhuang, X. Liu. "Verification of microservices using metamorphic testing", *Algorithms and Architectures for Parallel Processing*. V. I, 19th International Conference, ICA3PP 2019 (9-11 December 2019, Melbourne, VIC, Australia), Springer, 2020, pp. 138–152. doi ↑71

[37] S. Segura, J. A. Parejo, J. Troya, A. Ruiz-Cortés. "Metamorphic testing of RESTful web APIs", *Proceedings of the 40th International Conference on Software Engineering* (27 May 2018–3 June 2018, Gothenburg, Sweden), ACM, New York, 2018, ISBN 978-1-4503-5638-1, pp. 882. doi ↑71

[38] J. Brown, Z. Q. Zhou, Y.-W. Chow. *Metamorphic testing of navigation software: A pilot study with Google Maps*, University of Wollongong Research Online, 2018, 12 pp. URL ↑74

[39] M. Jia, X. Wang, Y. Xu, Z. Cui, R. Xie. "Testing machine learning classifiers based on compositional metamorphic relations", *International Journal of Performability Engineering*, **16**:1 (2020), pp. 67–77. doi ↑72, 78

[40] P. Saha, U. Kanewala. "Fault detection effectiveness of metamorphic relations developed for testing supervised classifiers", *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)* (04-09 April 2019, Newark, CA, USA), IEEE, 2019, pp. 157–164. doi ↑72

[41] M. P. Shahri, M. Srinivasan, G. Reynolds, D. Bimczok, I. Kahanda, U. Kanewala. "Metamorphic testing for quality assurance of protein function prediction tools", *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)* (04-09 April 2019, Newark, CA, USA), IEEE, 2019, pp. 140–148. ↑72

[42] A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. J. C. Bose, N. Dubash, S. Podder. "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing", *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis* (16-21 July 2018, Amsterdam, Netherlands), ACM, New York, 2018, ISBN 978-1-4503-5699-2, pp. 118–128. ↑72, 73

[43] H. Zhu, D. Liu, I. Bayley, R. Harrison, F. Cuzzolin. *Datamorphic testing: A methodology for testing ai applications*, 2019, 39 pp. arXiv 1912.04900 ↑72

[44] M. Zhang, Y. Zhang, L. Zhang, C. Liu, S. Khurshid. "DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems", *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering* (3-7 September 2018, Montpellier, France), ACM, New York, 2018, ISBN 978-1-4503-5937-5, pp. 132–142. ↑72

[45] M. Raif, E.-M. Ouafiq, Rharras A. El, A. Chehri, R. Saadane. "Metamorphic testing for edge real-time face recognition and intrusion detection solution", *2022 IEEE 96th Vehicular Technology Conference (VTC2022-Fall)* (26-29 September 2022, London, United Kingdom), IEEE, 2022, pp. 1–5. ↑72

[46] Z. Zhang, P. Wang, H. Guo, Z. Wang, Y. Zhou, Z. Huang. "DeepBackground: Metamorphic testing for Deep-Learning-driven image recognition systems accompanied by Background-Relevance", *Information and Software Technology*, **140** (2021), id. 106701. ↑72

[47] L. Xu, D. Towey, A. P. French, S. Benford, Z. Q. Zhou, T. Y. Chen. "Enhancing supervised classifications with metamorphic relations", *Proceedings of the 3rd International Workshop on Metamorphic Testing* (27 May 2018-03 June 2018, Gothenburg, Sweden), 2018, pp. 46–53. ↑72

[48] R. Yan, S. Wang, Y. Yan, H. Gao, J. Yan. "Stability evaluation for text localization systems via metamorphic testing", *Journal of Systems and Software*, **181** (2021), id. 111040. ↑73

[49] H. Park, T. Waseem, W. Q. Teo, Y. H. Low, M. K. Lim, C. Y. Chong. "Robustness evaluation of stacked generative adversarial networks using metamorphic testing", *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)* (02 June 2021, Madrid, Spain), IEEE, 2021, pp. 1–8. ↑73

[50] P. Ma, S. Wang, J. Liu. "Metamorphic testing and certified mitigation of fairness violations in NLP models", IJCAI'20: Twenty-Ninth International Joint Conference on Artificial Intelligence (7-15 January 2021, Yokohama, Japan), 2021, pp. 458–465, id. 64. ↑73

[51] S. Chen, S. Jin, X. Xie. “Validation on machine reading comprehension software without annotated labels: a property-based method”, *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (23-28 August 2021, Athens, Greece), ACM, New York, 2021, ISBN 978-1-4503-8562-6, pp. 590–602. ↑73

[52] L. Sun, Z. Q. Zhou. “Metamorphic testing for machine translations: MT4MT”, *2018 25th Australasian Software Engineering Conference (ASWEC)* (26-30 November 2018, Adelaide, SA, Australia), IEEE, 2018, pp. 96–100. ↑73

[53] W. Gao, J. He, V.-T. Pham. “Metamorphic testing of machine translation models using back translation”, *2023 IEEE/ACM International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest)* (15 May 2023, Melbourne, Australia), IEEE, 2023, pp. 1–8. ↑73

[54] D. Pesu, Z. Q. Zhou, J. Zhen, D. Towey. “A Monte Carlo method for metamorphic testing of machine translation services”, *Proceedings of the 3rd International Workshop on Metamorphic Testing* (27 May 2018, Gothenburg, Sweden), ACM, New York, 2018, ISBN 978-1-4503-5729-6, pp. 38–45. ↑73, 77

[55] Y. Sun, Z. Ding, H. Huang, S. Zou, M. Jiang. “Metamorphic testing of relation extraction models”, *Algorithms*, **16**:2 (2023), pp. 102. ↑74

[56] M. S. Raunak, M. M. Olsen. “Metamorphic testing on the continuum of verification and validation of simulation models”, *2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)* (02 June 2021, Madrid, Spain), IEEE, 2021, pp. 47–52. ↑74

[57] A. F. Donaldson, A. Lascu. “Metamorphic testing for (graphics) compilers”, *Proceedings of the 1st International Workshop on Metamorphic Testing* (14-22 May 2016, Austin, Texas), ACM, New York, 2016, ISBN 978-1-4503-4163-9, pp. 44–47. ↑75

[58] V. Le, M. Afshari, Z. Su. “Compiler validation via equivalence modulo inputs”, *ACM Sigplan Notices*, **49**:6 (2014), pp. 216–226. ↑75

[59] Z. Q. Zhou, T. Tse, M. Witheridge. “Metamorphic robustness testing: Exposing hidden defects in citation statistics and journal impact factors”, *IEEE Transactions on Software Engineering*, **47**:6 (2019), pp. 1164–1183. ↑77

[60] J. Ahlgren, M. Berezin, K. Bojarczuk, E. Dulskyte, I. Dvortsova, J. George, N. Gucevska, M. Harman, M. Lomeli, E. Meijer, S. Sapora. “Testing web enabled simulation at scale using metamorphic testing”, *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (25-28 May 2021, Madrid, Spain), IEEE, 2021, ISBN 978-1-6654-3869-8, pp. 140–149. ↑77

[61] F. u. Rehman, C. Izurieta. “Statistical metamorphic testing of neural network based intrusion detection systems”, *2021 IEEE International Conference on Cyber Security and Resilience (CSR)* (26-28 July 2021, Rhodes, Greece), IEEE, 2021, pp. 20–26. ↑77

[62] F. Tambon, G. Antoniol, F. Khomh. *HOMRS: High order metamorphic relations selector for deep neural networks*, 2021, 33 pp. arXiv 2107.04863 ↑78

[63] P. Zhang, X. Zhou, P. Pelliccione, H. Leung. "RBF-MLMR: A multi-label metamorphic relation prediction approach using RBF neural network", *IEEE Access*, **5** (2017), pp. 21791–21805. ↑78

[64] J. Zhang, J. Chen, D. Hao, Y. Xiong, B. Xie, L. Zhang, H. Mei. "Search-based inference of polynomial metamorphic relations", *ASE '14: Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering* (15-19 September 2014, Vasteras, Sweden), ACM, New York, 2014, ISBN 978-1-4503-3013-8, pp. 701–712. ↑78

[65] C.-A. Sun, A. Fu, P.-L. Poon, X. Xie, H. Liu, T. Y. Chen. "Metric $^{+}+$: A metamorphic relation identification technique based on input plus output domains", *IEEE Transactions on Software Engineering*, **47**:9 (2019), pp. 1764–1785. ↑78

[66] U. Kanewala, J. M. Bieman, A. Ben-Hur. "Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels", *Software: Testing, Verification and Reliability*, **26**:3 (2016), pp. 245–269. ↑78

[67] A. Blasi, A. Gorla, M. D. Ernst, Pezzè M., A. Carzaniga. "MeMo: Automatically identifying metamorphic relations in Javadoc comments for test automation", *Journal of Systems and Software*, **181** (2021), id. 111041. ↑78

[68] P. Wu. "Iterative metamorphic testing", *29th Annual International Computer Software and Applications Conference (COMPSAC'05)*. V. 2 (26-28 July 2005, Edinburgh, UK), IEEE, 2005, ISBN 0-7695-2413-3, pp. 19–24. ↑78

[69] Y. Yang, Z. Li, H. Wang, C. Xu, X. Ma. "Towards effective metamorphic testing by algorithm stability for linear classification programs", *Journal of Systems and Software*, **180** (2021), id. 111012. ↑78

[70] X. Xie, W. E. Wong, T. Y. Chen, B. Xu. "Metamorphic slice: An application in spectrum-based fault localization", *Information and Software Technology*, **55**:5 (2013), pp. 866–879. ↑78

[71] H. Liu, X. Liu, T. Y. Chen. "A new method for constructing metamorphic relations", *2012 12th International Conference on Quality Software* (27-29 August 2012, Xi'an, China), IEEE, 2012, pp. 59–68. ↑78

[72] K. Qiu, Z. Zheng, T. Y. Chen, P.-L. Poon. "Theoretical and empirical analyses of the effectiveness of metamorphic relation composition", *IEEE Transactions on software engineering*, **48**:3 (2020), pp. 1001–1017. ↑78

[73] Z.-W. Hui, S. Huang, C. Chua, T. Y. Chen. "Semiautomated metamorphic testing approach for geographic information systems: An empirical study", *IEEE Transactions on Reliability*, **69**:2 (2019), pp. 657–673. ↑78

[74] S. Iakusheva, A. Khritankov. "Metamorphic testing for recommender systems", *Analysis of Images, Social Networks and Texts*, AIST 2023, Lecture Notes in Computer Science, vol. **14486**, eds. Ignatov D.I. et al., Springer, Cham, 2024, ISBN 978-3-031-54533-7. ↑77

[75] Z.-w. Hui, X. Wang, S. Huang, S. Yang. "MT-ART: A test case generation method based on adaptive random testing and metamorphic relation", *IEEE Transactions on Reliability*, **70**:4 (2021), pp. 1397–1421. ↑78

[76] D. Sobania, M. Briesch, P. Röchner, F. Rothlauf. "MTGP: Combining metamorphic testing and genetic programming", Genetic Programming. EuroGP 2023, Lecture Notes in Computer Science, vol. **13986**, eds. Pappa G., Giacobini M., Vasicek Z., Springer, Cham, 2023, ISBN 978-3-031-29572-0, pp. 324–338. ↑78

Received	22.11.2023;
approved after reviewing	30.03.2024;
accepted for publication	31.03.2024;
published online	14.05.2024.

Recommended by *Andrey V. Klimov*

Information about the authors:

Sofia Fedorovna Iakusheva

Research interests: development and verification of programs, metamorphic testing, machine learning

0009-0000-1423-1123

e-mail: yakusheva.sf@phystech.edu

Anton Sergeevich Khritankov

Research interests: software engineering, machine learning, trustworthy artificial intelligence

0000-0003-2889-9436

e-mail: akhritankov@hse.ru

Authors contribution: *Sofia F. Iakusheva* — 70% (software, investigation, resources, data curation, writing (review & editing), visualization, supervision); *Anton S. Khritankov* — 30% (methodology, software, formal analysis, visualization, project administration, funding acquisition).

The authors declare no conflicts of interests.

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья искусственный интеллект, интеллектуальные системы, нейронные сети

УДК 004.421+004.891.2+510.649
10.25209/2079-3316-2024-15-2-87-112

Использование распределенных вычислений при моделировании предметной области в универсальной силлогистике

Юрий Михайлович **Сметанин**[1]✉
[1] Удмуртский государственный университет, Ижевск, Россия

Аннотация. Неклассическая пропозициональная логика L_{S_2} построена на базе алгебраической системы, содержащей булеву алгебру множеств и два отношения между множествами: $\subset$ и $=$. Ближайшим аналогом ее является силлогистика Аристотеля, математической моделью которой является алгебраическая система с Булевой алгеброй множеств и одним отношением $\subset$. Недостатком силлогистик, в основе которых лежит алгебраическая система с одним отношением $\subset$, является многосмысловость интерпретации их формул и атомарных суждений.

Под логико-семантической моделью предметной области в данной работе мы понимаем совокупность формулы универсальной силлогистики L_{S_2} и ее семантического значения, в качестве которого выступает конечное множество неотрицательных целых чисел. Предлагается алгоритм вычисления семантического значения конъюнктивной правильно построенной формулы L_{S_2}, обладающий высоким уровнем параллелизма на уровне задач, на уровне данных и на уровне алгоритмов, реализующих операции над составляющими множествами. В силу особенностей операций объединения, пересечения и дополнения универсума над конечными множествами все процессы их вычисления и решения подзадач происходят на битовом уровне и, как правило, эффективно реализуются на алгоритмических языках. В предлагаемом алгоритме переход на битовый уровень и обратно реализуется набором программных средств.

Ключевые слова и фразы: Силлогистика, дискретные диаграммы Венна, логико-семантическая моделъ, фронтальный алгоритм, распределенные вычисления

Для цитирования: Сметанин Ю. М. *Использование распределенных вычислений при моделировании предметной области в универсальной силлогистике* // Программные системы: теория и приложения. 2024. **Т. 15**. № 2(61). С. 87–112. https://psta.psiras.ru/read/psta2024_2_87-112.pdf

Введение

Рассмотрены алгоритмические вопросы построения логико-семантических моделей в универсальной силлогистике L_{S_2}, которая, по существу, является обобщением и детализацией математической модели (алгебраической системы), включающей одно отношение $\subseteq$ и вырожденную алгебру Буля. Обобщенная математическая модель лежащая в основе силлогистики L_{S_2} есть алгебраическая система с невырожденной булевой алгеброй конечных множеств и двумя отношениями строгого включения и равенства $\subset$, $=$.

Силлогистика Аристотеля базируется на алгебраической системе с невырожденной булевой алгеброй множеств и одним отношением $\subseteq$.

Построенная логико - семантическая модель определяется совокупностью правильно построенной формулой (ППФ) L_{S_2} и ее семантического значения, представленного дискретной диаграммой Венна (A-онтологией). ППФ выражает логическое содержание модели, а A-онтология представляет объем n-арного логического отношения определяемого ППФ.

В силлогистических теориях для представления семантики используют обычные диаграммы Венна и называют их модельными схемами, а множества, из которых они составлены, модельными множествами.

Прикладные аспекты использования построенных логико-семантических моделей заключаются в решении следующих задач.

Задача 1. *Даны посылки $P_1, P_2, \ldots, P_k$ в виде ППФ, доказать, что из них логически следует ППФ заключения Z. При этом задача 1 решается не путем построения логического вывода, а путем проверки логического следования $P_1 \& P_2 \& \ldots \& P_k \models Z$ в семантическом смысле. То есть в проверке того, что выполнение логического отношения, выражаемого посылками, влечет выполнение логического отношения, задаваемого заключением. Верификация логического следования осуществляется алгоритмически с использованием специально разработанного исчисления конституентных множеств* [5].

Задача 2. *Дан список посылок $P_1, P_2, \ldots, P_k$. Вывести интересные с прикладной точки зрения следствия из них.*

Задача 3. *Как нужно изменить посылки, чтобы получить из них требуемое следствие.*

Эта задача в нашей формализации рассматривалась в докладе на НСКФ-23 и будет являться темой нашей следующей публикации.

Использование обычных диаграмм Венна для решения задач 1 и 2 систематизировано в монографии [2]. Трудности применения этих диаграм связаны с собственно их построением и визуальным анализом диаграммы большой размерности. Установлено, что для большого числа модельных множеств построение наглядной диаграммы является нетривиальной задачей, имеющей важное прикладное значение. [3, 4].

Многоместные логические отношения между модельными множествами выражаются в силлогистике как соотношения между их объемами, выявить которые в обычной диаграмме можно лишь при ее надлежащей визуализации. Предлагаемый нами подход с использованием дискретных диаграмм Венна позволяет выражать модельные множества конечными множествами из неотрицательных целых чисел. Это позволяет от исчисления отношений объемов на картинках перейти к исчислению соотношений объемов модельных множеств, естественным способом представленных в компьютере с помощью битовых наборов.

Таким образом, построение логико-семантической модели предметной области для большого числа модельных множеств, является значимым результатом, так как позволяет решать задачу 1, не прибегая к построению логического вывода, который имеет экспоненциальную сложность и его практическое применение связано с трудоемкими преобразованиями исходных посылок например, для метода резолюций это процедура унификации.

В работе [7] указывается, что «*... во многих случаях классический вывод не годится для решения прикладных задач, называемых конструктивными. Их особенность заключается в том, что по выводу требуется произвести некоторое построение, заданное целью задачи. Так, если цель есть дизъюнция* $A(\bar{x}) \vee B(\bar{x})$*, то вывод должен давать способ определения по параметрам* $\bar{x}$*, входящим в* A *и* B*, какой из случаев имеет место для каждого конкретного набора параметров, и, если цель есть* $\exists \bar{x} F(\bar{x})$*, то из вывода должен извлекаться способ построения такого* $\bar{x}$.»

Предлагаемый нами алгебраический подход к решению задач 1–3 является конструктивным, об эффективности реализованного последовательного алгоритма построения логико-семантической модели можно судить по тому, что модель задачи о паровом катке смотри ([7] стр. 144) он строит в течение нескольких секунд. Дискретная диаграмма Венна, выражающая смысл посылок задачи, изображена в работе [5] на рисунке 7,

из которого легко сделать вывод, что доказываемое следствие имеет место. В работе [5] также рассматриваются различные примеры решения задач 1–3.

Таким образом, задача вычисления дискретной диаграммы Венна и преобразования ее для решения задач 1–3, размерность которых превышает 22[1], весьма актуальна для разработки алгоритмов искусственного интеллекта. Описываемый в данной работе параллельный алгоритм является развитием фронтального алгоритма решения задачи SAT [6].

Разработанная силлогистическая система L_{s_2} имеет областью интерпретации множество $U^0 = \{0, 1, \ldots, 2^n - 1\}$, где n – число модельных множеств, используемых в ее суждениях. Для интерпретации суждений Ф в этой силлогистике (вычисления их логического объема) конъюнктивные суждения приводятся к формуле $F_s(X_1, X_2, \ldots, X_n) = U$, ее значение $U = F(X_1, X_2, \ldots, X_n)$ принимается за смысловое содержание Ф. Для верификации логического следования нужно использовать исчисление конституентных множеств и логику L_{S_2}. Задача верификации логического следования в семантическом смысле для классической логики высказываний в работе [5] сведена к необходимости вычисления

$$F_p(x_1, x_2, \ldots, x_n) \models F_s(x_1, x_2, \ldots, x_n) \equiv F_p(\widetilde{x}_n) \models F_s(\widetilde{x}_n)$$

множеств U_p и U_s, входящих в суждения неклассической многозначной логики L_{S_2} вида

$$U_p = F_p(X_1^0, X_2^0, \ldots, X_n^0), \qquad U_s = F_s(X_1^0, X_2^0, \ldots, X_n^0),$$

получаемых из формул $F_p(\widetilde{x}_n), F_s(\widetilde{x}_n)$ заменой булевых переменных x_i на фиксированные для данного n множества целых неотрицательных чисел (конституентые множества) X_i^0. При этом доказано, что логическое следование

$$F_p(\widetilde{x}_n) \models F_s(\widetilde{x}_n)$$

имеет место при наличии отношений $U_p \subset U_s$ либо $U_p = U_s$.

Между булевыми переменными x_i и модельными множествами X_i установлено однознозначное соответствие – булева переменная является характеристической функцией соответствующего модельного множества (смотри формулу (11)). Тип данных «множество» эффективно реализуется в языках программирования в универсуме $\{0, 1, \ldots, 2^n - 1\}$ для небольших значений $n \leq 8$. В общем случае сложность вычисления операций булевой алгебры множеств имеет экспоненциальную оценку.

[1]Алгоритм построения дискретной диаграммы реализован для числа переменных не более 22.

1. Используемые теоретические результаты

Модель предметной области деятельности (ПОД) представляет собой правильно построенную формулу (ППФ) в L_{S_2}. Атомарные суждения логики (1) выражают объемные отношения множеств в универсуме U. Семантика дана равносильностями (2)–(4).

$$NOB_S = \langle \underbrace{A(X,Y)}_{S_1}, \underbrace{Eq(X,Y)}_{S_2}, \underbrace{IO(X,Y)}_{S_3}, \underbrace{X=U}_{S_4}, \underbrace{X \subset U}_{S_5} \rangle, \tag{1}$$

$$A(X,Y) \equiv (X \subset Y) \cdot (X \subset U) \cdot (X' \subset U) \cdot (Y \subset U) \cdot (Y' \subset U) \tag{2}$$

$$Eq(X,Y) \equiv (X = Y) \cdot (X \subset U) \cdot (X' \subset U) \cdot (Y \subset U) \cdot (Y' \subset U) \tag{3}$$

$$IO(X,Y) \equiv (X \cdot Y \neq \emptyset) \cdot (X \cdot Y' \neq \emptyset) \cdot (X' \cdot Y \neq \emptyset) \cdot (X' \cdot Y' \neq \emptyset) \tag{4}$$

Смысловое содержание атомарных суждений выражается утверждениями S_1, S_2, S_3, S_4, S_5 естественного языка относительно множеств X и Y, являющихся подмножествами универсума U:

S_1: $A(X,Y)$ – «Все элементы множества X являются элементами множества Y, но не наоборот», то есть $X \subset Y$ и $X \not\subset Y$.

S_2: $Eq(X,Y)$ – «Множества X, Y совпадают», то есть $X = Y$.

S_3: «Множества X, Y логически независимы». Это означает, что между элементами множеств $X, Y, X' = U \setminus X$ и $Y' = U \setminus Y$ нет отношений включения и равенства.

Утверждения S_1–S_3 подразумевают непустоту и неуниверсальность множеств X и Y, то есть $(X \subset U) \cdot (X' \subset U) \cdot (Y \subset U) \cdot (Y' \subset U)$.

S_4: «Множество совпадает с универсумом», то есть $X = U$.

Утверждения S_1–S_4 имеют односмысловое содержание в отличие от утверждения 5.

S_5: «Множество строго включено в универсум», то есть $X \subset U$.

Это утверждение имеет два смысла: либо $(X = \emptyset)$, либо $(X \subset U) \cdot (X' \subset U)$. Его можно записать в виде исключающей дизъюнкции
$(X = \emptyset) \oplus (X \subset U) \cdot (X' \subset U)$.

Все ППФ делятся на два класса: конъюнктивные (КППФ) и неконъюнктивные (НКППФ). Различие между ними заключается в том, что первые являются односмысловыми (однозначными), а вторые многосмысловыми (многозначными). Расчеты и принятие решений (выводы) в рамках модели сводятся к вычислению и анализу семантического значения ППФ.

Семантическим значением КППФ является множество неотрицательных целых чисел (конституентное множество), которое определяет дискретную диаграмму Венна [1][2]. Семантическим значением НКППФ является семейство конституентных множеств.

Предложен алгоритм вычисления семантического значения, который обладает высоким уровнем параллелизма на уровне задач, уровне данных, уровне алгоритмов, реализующих операции над конституентными множествами. Ввиду особенностей операций объединения, пересечения и дополнения до универсума над конечными множествами все процессы их вычислений и решение подзадач происходят на битовом уровне и, как правило, эффективно реализованы в алгоритмических языках. В предлагаемом алгоритме переход к битовому уровню и обратно осуществляется совокупностью программных средств.

Сложность вычисления по этому алгоритму зависит от числа модельных множеств, из которых конструируются понятия в ПОД, формы их представления и, в общем случае, совпадает со сложностью задачи поиска и перечисления всех выполняющих подстановок логического уравнения $F(\tilde{x}_n) = 1$, где $F(\tilde{x}_n)$ – конъюнктивная нормальная форма алгебры логики Буля.

Диаграммы Венна [2], используются в силлогистиках Аристотеля, Васильева, Керрола, Порецкого. Семантическое значение формул этих силлогистик выражается диаграммами Венна, которые можно построить по модельными схемами (5). Эти модельные схемы представляют алгебраическую систему в которой фиксируется нумерация и объем ее модельных множеств (7).

$$M_S = \left\langle \Omega(\widetilde{\aleph}_n), \aleph_1, \aleph_2, \ldots, \aleph_n \right\rangle, \qquad \widetilde{\aleph}_i = \left\langle \aleph_1, \aleph_2, \ldots, \aleph_n \right\rangle, \tag{5}$$

где Ω–универсум, $\aleph_i \subseteq \Omega$ – модельные множества. Число таких схем не более $2^{(2^n)}$. Каждая из этих схем определяется объемами модельных множеств которые ее образуют. Модельные множества $\widetilde{\aleph}_i = \langle \aleph_1, \aleph_2, \ldots, \aleph_n \rangle$, с фиксированной нумерацией и объемами, определяют семейство непустых конституент (6).

$$\aleph_1{}^{\sigma_1} \cdot \aleph_2{}^{\sigma_2} \cdot \ldots \cdot \aleph_n{}^{\sigma_n}, \aleph_i^{\sigma_i} = \begin{cases} \aleph_i, & \sigma_i = 1 \\ \aleph_i{}', & \sigma_i = 0. \end{cases} \tag{6}$$

Пусть множества $\widetilde{\aleph}_i$ схемы (5) являются подмножествами носителя алгебраической системы (АС) (7).

[2]Смотри рисунок 2 на стр. 167 в публикации [1]

$$\Lambda = \langle \Omega(\widetilde{\aleph}_n), \{ \quad \cdot, \quad + , \quad \prime \quad \}, \{\subseteq\}\rangle. \tag{7}$$

Носитель $\Omega(\widetilde{\aleph}_n)$ АС может быть представлен как объединение непустых конституент (6).

Удобство использования этой АС, в частности, в том, что по теореме Стоуна имеет место изоморфизм с АС (8), являющейся математической моделью Булевой логики высказываний с вырожденной булевой алгеброй

$$\lambda = \langle \omega(\tilde{x}_n), \{ \cdot, + , \prime \}, \{\leq\}\rangle, \qquad \omega = \{1\}, \qquad x_i \in \omega, \qquad i = \overline{1,n}. \tag{8}$$

Недостатком является многосмысловость интерпретации атомарных суждений вышеперечисленных силлогистик, например, в традиционной силлогистики Аристотеля смотри (13).

В наших исследованиях этот недостаток устранен за счет использования модели (9) с носителем из конституентных множеств и двумя отношениями $\subset$ и $=$.

Универсум Ω и модельные множества можно рассматривать как множества из номеров конституент, составляющих M_S (5). Нумерацию естественно производить, зафиксировав порядок индексов модельных множеств. При этом каждой конституенте (6) сопоставляется набор из нулей и единиц $\langle \sigma_1, \sigma_2, \ldots, \sigma_n \rangle$ и соответствующее ему десятичное число, которое будем называть номером конституенты. Множества, состоящие из номеров конституент далее называются *конституентными множествами.*

Рассмотрим алгебраическую систему (АС) для представления n-арной дискретной модельной схемы. Обозначим через $B(\widetilde{X}_n)$ универсум состоящий из номеров всех непустых конституент, которые могут быть построены из конечной системы $\widetilde{X}_n = \langle X_1, X_2, \ldots, X_n \rangle$ модельных конституентных множеств посредством операций объединения, пересечения и дополнения до универсума. $B(\widetilde{X}_n)$ включает также пустое множество. Рассмотрим АС (9), где W_F=$\{+, \cdot, ' \}$, W_R=$\{=, \subset\}$ [3],

$$\langle B(\widetilde{X}_n), \mathrm{W}_F, \mathrm{W}_R \rangle. \tag{9}$$

Выражающую эту алгебраическую систему модельную схему (10) с данной нумерацией X_i будем называть дискретной диаграммой Венна или

[3] W_F=$\{+, \cdot, ' \}$-операции объединение, пересечение, дополнение до универсума Булевой алгебры множеств

A-онтологией.

$$A = \langle U, X_1, X_2, \ldots, X_n \rangle, \qquad \widetilde{X}_i = \langle X_1, X_2, \ldots, X_n \rangle \tag{10}$$

Морфизм в форме гомоморфизма с АС (8) устанавливается также с помощью характеристической функции (11) множеств $X_i, X_i \subseteq U$

$$\forall e \in U \left(x_i(e) = \begin{cases} 1, & e \in X_i \\ 0, & e \notin X_i \end{cases}, \quad i = \overline{1,n} \right) \tag{11}$$

Существует однозначное соответствие между равенством ППФ булевой алгебры логики вида

$$F(\tilde{x}_n) = 1 \tag{12}$$

и равенством

$$F(\widetilde{X}_n) = U,$$

являющимся атомарным суждением (1). При этом булевы переменные являются характеристическими функциями соответствующих модельных множеств.

В публикациях [1, 5] категорическим атрибутивным суждениям NOB_S и ППФ логики L_{S_2} алгоритмически ставится в соответствие формула (формулы) булевой алгебры конституентных множеств. Вычисленное по этой формуле (формулам) конституентное множество (множества) есть ее семантическое значение.

Семантическим значением КППФ является контитуентное множество, семантическим значением НКППФ – семейство конституентных множеств.

Вместо модельных множеств X_i можно подставить любые (ППФ) Булевой алгебры множеств $F(\widetilde{X}_n)$.

Атомы NOB_S (1) являются альтернативой базису силлогистики Аристотеля $A_S = \langle AXY, EXY, IXY, OXY \rangle$, категорические суждения которого неоднозначно интерпретируются в 15 бинарных модельных схемах (смотри рисунок 1). Например, в NOB_S общеутвердительное Аристотелево суждение $AXY \equiv$«все X есть Y», интерпретируемое в рамках традиционной силлогистики в семи невырожденных Жергонновых отношениях $G_6, G_7, G_9, G_{11}, G_{13}, G_{14}, G_{15}$, имеет 2 смысла.

$$AXY \equiv \underbrace{Eq(X,Y)}_{G_9} + \underbrace{A(X,Y)}_{G_{13}}. \tag{13}$$

Если интепретировать это же суждение в пятнадцати модельных схемах, то оно имеет семь смыслов задаваемых дизъюнкцией попарно несовместных конъюнкций атомов из (1):

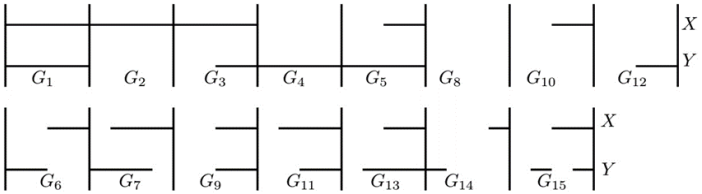


Рисунок 1. 15 бинарных Жергонновых логических отношений

$$AXY \equiv \underbrace{(X=Y)\cdot(X=U)}_{G_1} + \underbrace{(X'=U)\cdot(Y=U)}_{G_4} + \underbrace{(X=Y)\cdot(X'=U)}_{G_8} +$$
$$\underbrace{Eq(X,Y)}_{G_9} + \underbrace{A(X,Y)}_{G_{13}} + \underbrace{(X\subset U)\cdot(X'\subset U)\cdot(Y=U)}_{G_5} +$$
$$\underbrace{(X'=U)\cdot(Y\subset U)\cdot(Y'\subset U)}_{G_{12}}.$$[4] Из квадрата Пселла следует, что отрицание общеутвердительного суждения есть суждение частноотрицательное OXY оно в общем случае имеет восемь смыслов:
$OXY = AXY' \equiv OXY \equiv G_2(X,Y) + G_3(X,Y) + G_6(X,Y) + G_7(X,Y) + G_{10}(X,Y) + G_{11}(X,Y) + G_{14}(X,Y) + G_{15}(X,Y)$

Представленные выше НКППФ обладают важным свойством ортогональности их дизъюнктивных членов, выражающейся в том, что любая их конъюнкция несовместима в том смысле, что не существует A-онтологии, в которой бы выполнялись все отношения этой конъюнкции. Такие НКППФ называются нормализованными.

В [5] показано, что любую НКППФ можно за счет эквивалентных преобразований представить в виде конъюнкции ортогональных дизъюнктов, что позволяет вычислять ее семантическое значение как совокупность семантических значений отдельных дизъюнктов.

Набор M непустых конституент дискретной диаграммы Венна (A-онтологии) (10) называется ее характеристическим множеством. Следуя Порецкому П. С. будем также называть это множество *единицей*, множество номеров пустых конституент – *нулем* и обозначать как N. Любое модельное множество схемы (10) равно объединению некоторых конституент из M.

[4] Отметим, что за пределами традиционной силлогистики Аристотеля, находясь в рамках здравого смысла, невозможно использовать вырожденные Жергоновы отношения включения и равенства $G_1, G_2, G_3, G4, G_5, G_8, G_{10}, G_{12}$ для интерпретации причинно-следственных связей в ПОД, если хотя бы одно из множеств в них пусто либо является универсумом.

Модельные множества X_i из (10), как и *единицу*, будем задавать конституентными множествами.

Определено понятие закона, выполнимой формулы и противоречия.

Закон – это КППФ, семантической значение которой есть отношение независимости в совокупности модельных множеств, которые использованы при ее построении, смотри (21).

Например, для семейства из n модельных множеств $\widetilde{X}_n$ законом будет любая КППФ вида $IO(X_i, X_j) : i \neq j, \quad i, j \in \{1, 2, \ldots, n\}$.

Модельную схему, составленную из конституентных множеств, будем называть A-онтологией или дискретной диаграммой Венна.

НКППФ не может быть законом.Это выполнимая формула либо противоречие.

Выполнимой называется КППФ формула, в семантическом значении которой выполняются все логические отношения, задаваемые ее конъюнктами. В противном случае КППФ называется противоречием.

НКППФ выполнима, если выполним хотя бы один из ее ортогональных дизъюнктов, в противном случае НКППФ называется противоречием.

Разработано исчисление конституентных множеств [**1**, **5**, **6**], положения которого позволяют вводить в отношение независимости в совокупности (21) логические отношения из NOB_S, используемые в КППФ. Это делается за счет объявления в нем пустыми некоторого множества конституент.

A-онтология с универсумом $U = \{0, 1, 2, \ldots, 2^n - 1\}$ и с модельными множествами определенными в (21) задает логическое отношение независимости в совокупности для этих модельных множеств. Построение A-онтологии в которой выражается семантика КППФ осуществляется последовательным введением в атомарных конъюнктов КППФ вида $A(X, Y)$ и $Eq(X, Y)$ объявляя пустыми конституенты $X \cdot Y'$ и $X' \cdot Y$ на основании тождеств

$$X \cdot Y' = \emptyset \equiv X' + Y = U \tag{14}$$

$$(X \cdot Y' + X' \cdot Y = \emptyset) \equiv (X' + Y) \cdot (X + Y') = U \tag{15}$$

Показано, что атомарные отношения $X = U$, $A(X, Y)$ и $Eq(X, Y)$ вводятся в отношение (21) единственным способом за счет отнесения к *нулю* N коституент, номера которых выражаются конституентными множествами X', $X \cdot Y'$ и $X \cdot Y' + X' \cdot Y$ соответственно. При этом отношение независимости $IO(X, Y)$ может быть введено неоднозначно, также как и отношение $X \subset U$.

Поэтому, если в КППФ имеются конъюнкты, выражающие отношения $IO(X,Y)$ и $X \subset U$, то их наличие в семантическом значении КППФ просто проверяется после введения отношений $X = U$, $A(X,Y)$ и $Eq(X,Y)$.

Если все отношения, задаваемые конъюнктами КППФ, выполняются, то такая формула считается выполнимой, в противном случае противоречием.

Замечание 1. *Доказано, что семантическое значение КППФ получаемое таким способом:*

(1) *получается одинаковым вне зависимости от порядка введения в исходное отношение* (21) *конъюнктов КППФ;*

(2) *объем получаемого семантического значения получается максимальным среди всех, в которых выполняются отношения из КППФ;*

(3) *добавление в полученное семантическое значение любой конституенты из* нуля *приводит к невыполнению хотя бы одного из отношений задаваемых конъюктами КППФ;*

(4) *для вычисления семантического значения КППФ ее атомарным конъюнктам, отличным от $IO(X,Y)$ и $X \subset U$, ставятся в соответствие множества $U = X$, $U = X' + Y$, и $U = (X' + Y) \cdot (X + Y')$ для конъюнктов $U = X$, $A(X,Y)$ и $Eq(X,Y)$ соответственно;*

(5) *семантическое значение КППФ получается как пересечение множеств U, полученных в пункте (4);*

(6) *при вычислении множеств U пункта (4) в качестве модельных множеств можно для вычисления любого атомарного конъюнкта использовать модельные множества X_i^0 из отношения* (21).

Интенсиональной моделью для категорических суждений NOB_S и конъюнктивных ППФ L_{S_2} в нашем подходе является A-онтология (10) – дискретный аналог модельной схемы, преимуществом которой является ее представление конечным конституентным множеством, что позволяет просто программно реализовать исчисление конституентных множеств.

Несомненным достоинством подхода на основе невырожденной булевой алгебры является то, что построенная силлогистическая теория является универсальной, имеет синтаксическую экстенсиональную интерпретацию в виде формулы алгебры множеств, интенсиональную интерпретацию в виде конституентного множества, вычисляемого по формуле. Таким образом, схема интерпретации алгоритмически разрешима.

Семантическое значение КППФ выражается универсумом A-онтологии.

Сама A-онтология $I_n = \langle U, X_1, X_2, \ldots, X_n \rangle$ позволяет наглядно изобразить семантику КППФ[1].

Универсуму U любой A-онтологии, в которой $X_i = U \cdot {X_i}^0$ можно сопоставить множество равносильных формул из модельных множеств, выражающих U. Поэтому A-онтологии I_n с единицей $M(I_n)$ можно сопоставить атомарное суждение $F(\widetilde{X}_n) = U$, здесь $F(\widetilde{X}_n)$ – ППФ алгебры множеств равносильная совершенной нормальной форме Кантора $SNFK$ у которой $M(SNFK) = U$.

Например, пусть A-онтология I_4 определяется универсумом $U = \{5, 7, 8, 9, 12, 13, 15\}$. Тогда ей можно сопоставить КППФ (16).

$$X_1 \cdot {X_3}' + X_2 \cdot X_4 = U. \tag{16}$$

Множество (16) представлено как объединение неортогональных множеств так как $(X_1 \cdot {X_3}') \cdot (X_2 \cdot X_4) = \{13\} \neq \emptyset$.

Равносильное равенство (17) с левой частью в форме $SNFK$

$$\begin{array}{l} {X_1}' \cdot X_2 \cdot {X_3}' \cdot X_4 + {X_1}' \cdot X_2 \cdot X_3 \cdot X_4 + X_1 \cdot {X_2}' \cdot {X_3}' \cdot {X_4}' + \\ X_1 \cdot X_2' \cdot {X_3}' \cdot X_4 + X_1 \cdot X_2 \cdot X_3' \cdot {X_4}' + X_1 \cdot X_2 \cdot {X_3}' \cdot X_4 + \\ X_1 \cdot X_2 \cdot X_3 \cdot X_4 = U \end{array} \quad . \tag{17}$$

Еще одно равносильное равенство (18) предствляет универсум для I_4. Его левая часть есть объединение ортогональных множеств.

$$X_1' \cdot X_2 \cdot X_4 + X_1 \cdot X_3' + X_1 \cdot X_2 \cdot X_3 \cdot X_4 = U \tag{18}$$

Все три КППФ (16), (17), (18) имеют одинаковое семантическое значение.

Имеет место функциональная полнота атомарных суждений (1), то есть любая A-онтология (n-арное логическое отношение) может быть выражена КППФ в L_{S_2}

Первая версия M-алгоритма вычисления единицы КППФ в L_{S_2} реализована программно на языке $FreePascal$. Сложность вычислений экспоненциально зависит от числа $n-8$, где n - число модельных множеств.

Объем используемой памяти также экспоненциально зависит от $n-8$. Например, для 22 переменных канонический универсум занимает $2^{22-8} \cdot 2^5 = 0,5$ мегабайт дисковой памяти и представлен массивом из базовых множеств [0..255] содержащим 16384 элемента. Если число переменных будет 40, то объем будет $2^{40-8} \cdot 2^5 = 2^37 = 2^{10} \cdot 2^{10} \cdot 2^{10} \cdot 128 = 128$ гигабайт. Время вычисления единицы линейно зависит от количества операций K_{op} и равно

$$T = K_{op} \cdot t_0 \cdot 2^{22-8}.$$

Здесь $t^0 = \max\{t(+), t(\cdot), t^0$ это максимальное время выполнения операций объединения, пересечения и дополнения до универсума для базового множества. К этому числу нужно добавить время для потоковой записи и считывания множеств из массивов.

Недостатком этого алгоритма является большие потребности в памяти оперативной и дисковой для размещения модельных множеств, а достоинством – скорость вычислений, так как один элементы массивов содержащих множества обрабатываются как стандартные множества в языка Free Pascal. Способ эффективного распараллеливания алгоритма вычислений и структура данных описана в [6].

Во фронтальном алгоритме решения задачи SAT, посредством которого можно также решать задачу поиска всех выполняющих подстановок, реализован другой способ представления множеств, изложенный в работе [8], дающий более компактное представление конституентных множеств.

В данной статье представлен алгоритм распределенных вычислений семантического значения КППФ L_{S_2} построенный на основе фронтального алгоритма [6].

2. Представление исходных данных алгоритма вычисления семантического значения ППФ L_{S_2}

В этом разделе описаны структура данных для вычисления и представления конституентных множеств.

2.1. n-арное отношение логической независимости модельных множеств

Рассмотрим алгебраическую систему и A-онтологию (19) и (20)

$$A_s{}^0 = \langle U^0(\tilde{X_n}{}^0) = \{0, 1 \ldots, 2^n - 1\}, \{+, \cdot, '\}, \{\subset, =\}\rangle, \tag{19}$$

в которых опорное множество $U^0(\widetilde{X}_n^0)$ представляет собой объединение множеств номеров непустых конституент (20), представленное как (21).

$$\widetilde{X}_n^0 = \langle X_1{}^0, X_2{}^0, \ldots, X_n{}^0\rangle, \tag{20}$$

где

$$U^0(\widetilde{X}_n^0) = \begin{bmatrix} 1 & * & \ldots & * & * \\ * & 1 & \ldots & * & * \\ & & \ldots & & \\ * & * & \ldots & 1 & * \\ * & * & \ldots & * & 1 \end{bmatrix} \begin{matrix} = X_1^0 \\ = X_2^0 \\ \ldots \\ = X_{n-1}^0 \\ = X_n^0 \end{matrix}, \tag{21}$$

где

$$1 = \{1\}, * = \{0,1\}, X_i^0 = \langle \underbrace{\{0,1\}}_{1} \times \underbrace{\{0,1\}}_{2} \times \cdots \times \underbrace{\{1\}}_{i} \times \cdots \times \underbrace{\{0,1\}}_{n} \rangle$$

Алгебраическая система (19) и A-онтология (20) называются каноническими и определяют логическое отношение независимости в совокупности модельных множеств. Объявляя некоторые конституенты пустыми, мы получаем зависимую в совокупности дискретную диаграмму Венна.

Итак, логическое содержание получаемого при этом n-арного отношения будем выражать КППФ в L_{S_2}. Алгоритм вычисления семантического значения (объема) понятия, выражаемого логическим содержанием КППФ с использованием распределенных вычислений, рассматривается ниже.

Доказано, что логическое содержание в виде КППФ может быть выражено множеством способов по известному *нулю* и *единице*.

3. Структура данных, используемых для построения логико-семантической модели предметной области

Алгоритм распределенных вычислений модели предметной области в L_{S_2} далее сокращенно Md-алгоритм описан в следующем разделе. Этот алгоритм по заданной КППФ, состоящей из конъюнкций атомов L_{S_2} вычисляет ее единицу, которой соответствует дискретная диаграмма Венна – A-онтология.

КППФ есть конъюнкция атомарных суждений NOB_S вида

$$A(X,Y), Eq(Z,V), W = U,$$

где множества X, Y, Z, V, W представлены ППФ Булевой алгебры составленных из модельных конституентных множеств.

На первом этапе каждый из этих атомов заменяется на равенства

$$U_A = X' + Y; U_{Eq} = (Z' + V) \cdot (Z + V'); U_{W=U} = W. \tag{22}$$

Затем из правых частей этих равенств формируется равенство для вычисления единицы КППФ, которая и есть логико-семантическая модель.

Левая часть итогового равенства (24) – это универсум мира, в котором должны выполняться логические условия, задаваемые атомами КППФ, правая часть есть пересечение всех частных универсумов *единиц*, определяемых равенствами (22). Например, для формулы (23)

$$A(X,Y) \cdot Eq(Z,V) \cdot (W = U) \tag{23}$$

получится следующая задача найти U :

$$U = (X' + Y) \cdot (Z' + V) \cdot (Z + V') \cdot W \tag{24}$$

Каждая частная *единица* подвергается ортогонализации по Порецкому C-система

$$A_1 + A_2 + \ldots A_k$$

заменяется на ортогональную

$$A_1 + A'_1 \cdot A_2 + \ldots + A'_1 \cdot A'_2 \cdot \ldots \cdot A_k \tag{25}$$

Этим представление исходных данных для вычисления $U-$ *единицы* в Md-алгоритме отличается от представления исходных данных во фронтальном алгоритме. В нем логическое уравнение $F(\tilde{x}_n) = 1$ представлено как конъюнктивная нормальная форма, из которой строится C-система [8] конституентных множеств, выражающая нуль атомарного равенства

$$N\big(F(\widetilde{X}_n) = U\big).$$

Для того, чтобы найти единицу $M\big(F(\widetilde{X}_n) = U\big)$, мы находим дополнение построенной C-системы как пересечение дополнений каждого ее элемента. Организация вычислений этих пересечений описана в в виде фронтального алгоритма [1]. Поскольку структура исходных данных фронтального алгоритма включена в структуру данных M_D-алгоритма, то форма задачи вычисления левой части равенства (24) остается такой же, как во фронтальном алгоритме. Это задача нахождения пересечений объединения множеств, составляющих *единицы* каждого конъюнкта КППФ, либо *единиц* клозов во фронтальном алгоритме.

Замечание 2. *В результате решения задачи* (24) *получается множество U как объединение ортогональных конституентных множеств.*

В [1] показано, что количество операций пересечения множеств, составляющих промежуточные *единицы*, зависит от порядка вычисления операций пересечения *единиц* конъюнктов КППФ.

Минимизация числа операций умножения достигается за счет приоритетного выполнения умножений *единиц*, которые представлены меньшим числом ортогональных множеств.

При решении задачи нахождения результата пересечения единиц всех конъюнктов возникает проблема быстрого возрастания объема промежуточных результатов вычислений для числа модельных множеств свыше 30.

Данная проблема может быть решена при последовательных вычислениях выгрузкой этих промежуточных результатов из оперативной памяти в долговременную.

Ограничив объем используемых в оперативной памяти промежуточных результатов допустимой порцией, мы приходим к классическому алгоритму с возвратами, дерево вычислений которого развивается в глубину. При этом на каждом шаге на выполнимость для очередного клоза тестируется сразу множество подстановок.

Фронтальный алгоритм реализован. Логическое уравнение в форме КНФ=1 генерируется случайным образом. Отмечена зависимость выражающаяся в том, что при уменьшении ширины фронта поиска время решения задачи увеличивается. Чрезмерное увеличение ширины текущего фронта вычислений заводит вычислительный процесс в *тупик* (deadlock=нехватка памяти).

Этот алгоритм можно подвергнуть распараллеливанию. Это позволяет получать решение SAT с помощью распределенных вычислений. Предлагаемый ниже M_D-алгоритм является обобщением фронтального алгоритма, поэтому также может быть использован для решения SAT.

4. M_D-алгоритм распределенных вычислений для построения логико-семантической модели предметной области

Отличие предлагаемого алгоритма о фронтального алгоритма из работы [1] заключается в том, что фронтальный алгоритм осуществляет поиск хотя бы одной выполняющей подстановки логического уравнения $F_p(x_1, x_2, \ldots, x_n) = 1$, левая часть которого есть КНФ. Изоморфнное преобразование этого уравнения в атомарное равенство NOB_S вида $F_s(X_1, X_2, \ldots, X_n) = U$, составленное из модельных множеств, позволяет находить выполняющие подстановки для каждой входящей в логическое уравнение КНФ более простым способом, нежели вычисления по соответствующей ей формуле Булевой алгебры множеств. Этот способ описан в вышеупомянутой работе на страницах 173 – 176.

Отличие иллюстрируется в примере 1. В M_D алгоритме для вычисления единиц посылок примера необходим интерпретатор для вычисления значений формул Булевой алгебры множеств.

Пример 1. *Рассмотрим первую задачу Порецкого. Между птицами зоосада существует 5 отношений:*

(1) *Птицы певчие - крупные или обладающие качеством Y.*

(2) *Птицы, не имеющие качества Y - или не крупные, или не имеют качества X.*

(3) *Птицы певчие в соединении с крупными объединяют всех птиц с качеством X.*

(4) *Каждая некрупная птица есть и певчая, или обладающая качеством X.*

(5) *Между птиц с качеством X совсем нет таких птиц с качеством Y, которые не будучи певчими, были бы крупные.*

Требуется

- *Определить, были ли птицы качества X певчие или нет.*
- *Узнать, то же в отношении птиц качества Y.*
- *Найти, были ли среди качества X птицы качества Y и наоборот.*

Решение. В логике классов Порецкого в качестве общеутвердительного суждения принимаетя равенство $X = X \cdot Y$ равносильное по смыслу Аристотелеву AXY в традиционной силлогистике и такое же двусмысленное. Пусть

X – множество птиц качества X,
Y – множество птиц качества Y,
S – множество певчих птиц,
G – множество крупных птиц.

Логические условия задачи, записанные в форме утверждений о равенстве либо строгом включении множеств, имеют вид (26). Конъюнкция данных посылок преобразуется в НКППФ, составленную как дизъюнкция 32 КППФ, попарная конъюнкция которых является противоречием. Таким образом, логико-семантическая модель задачи может иметь 32 смысловых содержания. Проиллюстрируем построение единиц для случая, когда каждое общеутвердительное суждение имеет смысл строгого включения (27).

$$
\begin{aligned}
&P1. \quad (S \subset Y + G) + (S = Y + G);\\
&P2. \quad (Y\,' \subset X' + G\,') + (Y\,' = X\,' + G\,');\\
&P3. \quad (X \subset S + G) + (X = S + G);\\
&P4. \quad (G\,' \subset S + X) + (G\,' = S + X);\\
&P5. \quad (X \subset (Y \cdot S\,' \cdot G)') + (X = (Y \cdot S' \cdot G)').
\end{aligned}
\tag{26}
$$

$$
\begin{aligned}
&P1. \quad (S \subset Y + G);\\
&P2. \quad (Y\,' \subset X' + G\,');\\
&P3. \quad (X \subset S + G);\\
&P4. \quad (G\,' \subset S + X);\\
&P5. \quad (X \subset (Y \cdot S\,' \cdot G)')).
\end{aligned}
\tag{27}
$$

Вычислим единицы посылок.

$M(A(S, Y + G)) = S' + Y + G = \{0, 1, 2, 3, \underline{5}, \underline{7, 8, 9}, 10, 11, \underline{12, 13}, 14, \underline{15}\}$
$M(A(Y', X' + G')) = Y + X' + G' = \{0, 1, 2, 3, 4, \underline{5}, 6, \underline{7, 8, 9}, 11, \underline{12, 13, 15}\}$
$M(A(X, S + G)) = X' + S + G = \{0, 1, 4, \underline{5}, 6, \underline{7, 8, 9}, \underline{10, 11, 12, 13}, 14, \underline{15}\}$
$M(A(G', S + X)) = G + S + X = \{2, 3, 4, \underline{5}, 6, \underline{7, 8, 9}, 10, 11, \underline{12, 13}, 14, \underline{15}\}$
$M(A(X, (Y \cdot S' \cdot G)') = X' + (Y \cdot S' \cdot G)' =$
$\{0, 1, 2, 3, 4, \underline{5}, 6, \underline{7, 8, 9}, 10, \underline{12, 13}, 14, \underline{15}\}$

Модель задачи представляется конституентным множеством, U_r являющимся пересечением единиц посылок. Дискретная диаграмма Венна для него показана на рисунке 2.

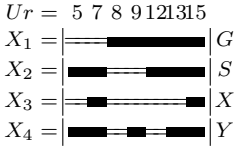


Рисунок 2. Модель задачи о птицах в ней находятся ответы на поставленные вопросы

Для вычисления единиц посылок на первом этапе M_D-алгоритма необходимо применять интерпретатор ППФ Булевой алгебры множеств. Вычисление пересечения единиц посылок составляет второй этап M_D-алгоритма, в котором реализуются распределенные вычисления. Последняя единица в результате вычисления множества $X' + (Y \cdot S' \cdot G)'$ в алгебре кортежей имеет вид C-системы составленной из ортогональных множеств

$$
\begin{bmatrix}
* & * & 0 & *\\
0 & * & 1 & *\\
1 & 1 & 1 & *\\
1 & 0 & 1 & 0
\end{bmatrix}
= \{1..10, 12..15\}.
$$

Согласно пункту (*б*) замечания 1 в качестве значений модельных множеств

взяты их значения из канонической A-онтологии для $n = 4$

$$\begin{array}{ll} G = & \left[\begin{array}{cccc} 1 & * & * & * \end{array}\right. \\ S = & \left.\begin{array}{cccc} * & 1 & * & * \end{array}\right. \\ X = & \left.\begin{array}{cccc} * & * & 1 & * \end{array}\right. \\ Y = & \left.\begin{array}{cccc} * & * & * & 1 \end{array}\right] . \end{array}$$

Пусть дана КППФ $\Phi(\widetilde{X}_n)$ силлогистики L_{S_2}, представленная конъюнкцией атомарных утверждений NOB_S вида

$$A(F_1(\widetilde{X}_n^0), F_2(\tilde{X}_n^0)), Eq(F_3(\widetilde{X}_n^0), F_4(\tilde{X}_n^0)), F_0(\widetilde{X}_n^0) = U, \tag{28}$$

где $F_i(\widetilde{X}_n^0), i = \overline{0,4}$ – ППФ алгебры множеств, а X_i^0 – конституентные множества канонической A-онтологии (21). Тогда, как показано выше, задача вычисления единицы этой КППФ сводится к вычислению пересечений конституентных множеств $M(i)$ для каждого конъюнкта (28)

$$M(\Phi(\widetilde{X}_n^0)) = M(1) \cdot M(2) \cdot \ldots \cdot M(k). \tag{29}$$

Первым промежуточным результатом вычислений будем считать $M(1)$, то есть $R(1) = M(1)$. В однопроцессорном варианте вычислительного процесса при неограниченном ресурсе оперативной памяти алгоритм вычислений сводится к последовательному вычислению пересечения всех единиц, то есть ставится задача уровня 1 вычисления множества $Z(1)$

$$Z(1) = R[1] \cdot M[2] \cdot \ldots \cdot M[k] = R[1] \cdot \prod_{i=2}^{k} M(i) \tag{30}$$

Если принять, что задача уровня i определяется как

$$Z(i) = R(i) \cdot M(i+1) \cdot \ldots \cdot M(k),$$

то список оставшихся множеств $M(i)$, для нахождения их пересечения, определяется i-м промежуточным результатом $R(i)$.

Таким образом, промежуточный результат определяет дальнейший порядок действий. Например, для $i = 3$ необходимо найти пересечение $R(3) \cdot M(4) \cdot \ldots \cdot M(k)$.

Состояния вычислительного процесса задаются последовательностью

$$R(1), R(2), R(k-1), R(k);$$

$$R(1) = M(1); R(i) = R(i-1) \cdot M(i), R(k) = R(\text{к} - 1) \cdot M(k). \tag{31}$$

$R(k)$ это решение задачи (30).

Объемы памяти, занимаемые множествами $R(i), M(j)$, пропорциональны количеству ортогональных множеств (25), из объединения которых они состоят. Обозначим эти объемы как $|R(i)|$, $|M(j)|$.

Информация о текущем объеме памяти, занимаемом промежуточным результатом $R(i) - |R(i)|$, и объеме памяти $|\mathrm{M}(i+1)|$ дает возможность прогноза объема следующего промежуточного результата, который определяется как

$$|R(i)| \cdot |M(j)|.$$

Это позволяет избежать *тупика* и планировать распараллеливание вычислительного процесса для построения логико-семантической модели.

Пусть для распределенных вычислений используются персональные компьютеры. На них устанавливается программа для решения задач типа (30). На одном из компьютеров реализован почтовый сервер, на котором установлено программное обеспечение для решения задачи (30) по рекуррентной схеме (31) и программа Супервизор для формирования, рассылки подзадач и получения полных либо частичных результатов их решения.

С учетом выбранной кодировки множеств и их представления в оперативной памяти в виде динамических массивов или списков ортогональных компонент, выберем предельный размер списка либо массива, например, в количестве $L_0 = 2^{20}$ элементов. По текущему состоянию $R(i)$, зная объем множества промежуточных результатов $|R(i-1)|$ и $|M(i)|$, можно оценить возможность краха вычислительного процесса из - за нехватки памяти. Обозначим это гипотетическое состояние как $Deadlock(i)_M$, что означает реально наступившую нехватку памяти на i-м шаге, либо прогноз ее наступления. Признаком $Deadlock(i)_M$ является выполнение неравенства (32)

$$|R(i-1)| \cdot |M(i)| \geq L_0 \tag{32}$$

Выполнение этого условия означает, что при вычисление пересечения

$$R(i-1) \cdot M(i)$$

может быть не завершено из-за нехватки памяти. Состояние $Deadlock(i)_M$ может не наступить, если при нахождении пересечения компонент множеств $R(i-1)$ и $M(i)$ получится достаточное количество результатов в виде пустых множеств. Тогда возможно будет вычислить $R(i)$. Тем не менее, будем считать,что следующий шаг вычислений запрещен и выполнение неравенства (32) требует перестройки вычислительного процесса, а именно его распараллеливания.

К этому моменту задача (30) решена частично с результатом $R(i-1)$, который сохраняется в файле с именем содержащим информацию о номере $i-1$, например, для $i=5$ имя будет $R\_4.set$.

Далее Супервизор разбивает множество $R(i-1)$ на $r=2^{10}$, либо $r=2^{10}+1$ подмножеств $R(i-1,j), \overline{j=1,r}$, сохраняя каждое в файл с именем содержащий информацию о значениях $i-1$ и j. Для $i=5, j=6$ файл будет иметь имя $R\_4\_6.set$. Все имена файлов сводятся в список уровня $i-LF\_R\_i$, содержащем информацию о подзадачах i-го уровня. В результате мы получим r подзадач i-го уровня, которые можно распределять по компьютерам, принимающим участие в распределенных вычислениях.

Эти компьютеры в начале своей работы регистрируются на сервере и получают уникальный номер. Также им пересылаются программы для вычисления пересечений множеств и формирования подзадач, а также файлы, содержащие единицы $M(j), j \in \{1, 2 \ldots, k\}$.

При подключении компьютера к распределенным вычислениям ему пересылается часть промежуточного результата, полученного перед предполагаемым $Deadlock(i)_M$. Это ножество $R(i-1)_{part}$, которое хранится в соответствующем файле, например, для в файле $R\_4\_6.set$.

Таким образом, подключаемый к процессу вычислений компьютер получает задание решить подзадачу

$$Z(i-1) = R(i-1)_{part} \cdot \prod_{j=i}^{k} M(j)$$

Элементы списка LF_R_i состоят из кортежей $\langle Nf, State, Result \rangle$. Первый элемент кортежа содержит имя файла подзадачи, например, $R\_4\_6.set$, второй элемент кортежа указывает на состояние процесса решения подзадачи. $State$ содержит информацию о том, распределена ли исполнителю задача с указанием его номера, либо нет. $Result$ указывает на результат завершения исполнения задачи. Таким образом, задача может быть не распределена либо направлена на исполнение указанному исполнителю. Результат завершения исполнения подзадачи может быть следующий:

1) подзадача завершилась аварийно;

2) в результате завершения процесса, сопоставленного подзадаче получено ее решение, которое является частью решения задачи $Z(1)$. Результат в виде файла отправляется на головной компьютер;

3) процесс решения подзадачи завершился формированием новых подзадач и списка LF_R_j.[5] аналогичного списку LF_R_i головного компьютера. Этот список отправляется на сервер и на головном компьютере формируется список списков подзадач различного уровня;

4) в результате выполнения подзадачи получено пустое множество.

Аварийное завершение означает, что подзадачу нужно передать новому исполнителю. Если результат завершения соответствует пункту 3), то сформированный список подзадач дополняет ранее сформированные списки и пополняет фронт работ (подзадач), которые супервизору нужно распределять между исполнителями.

Подзадачи, процесс выполнения которых завершен не аварийно, удаляются из списков подзадач. И результат их выполнения добавляется в файл ортогональных множеств входящих в решение задачи первого уровня.

Процесс выполнения распределенных вычислений заканчивается, когда выполнение всех сформированных в вычислительном процессе задач заканчивается.

В результате мы получим логико-семантическую модель в форме единицы задачи $Z(1)$, которая определяет состав модельных множеств $\widetilde{X}_i$ и дискретную диаграмму Венна. Это дает возможность решать задачи выявления неочевидных причинно-следственных отношений в предметной области.

Фронтом выполнения подзадач управляет Супервизор, комбинируя стратегии «поиск в глубину» и «поиск в ширину».

Заключение

Предлагаемый способ распределенных вычислений имеет простое описание по сравнению со всеми алгоритмами, основанными на идеях алгоритмов проверки выполнимости отдельно взятых подстановок, за исключением поисковых алгоритмов, корые не гарантируют нахождение всех выполняющих подстановок уравнения (12). Также этот способ решения не предъявляет высокие требования к вычислительным ресурсам. Теоретической основой его является силлогистическая система L_{S_2}

Предлагаемый алгоритм отличается от семейств алгоритмов основанных на DPLL и CDCL. Существенное сходство данных семейств

[5] j указывает на уровень сформированных подзадач

алгоритмов в том, что логика их работы основана на анализе значений булевых переменных в Булевой логике высказываний. Ее математической моделью служит алгебраическая система с одноэлементным носителем и одним отношением $\subseteq$, [6], и вырожденной (degenerate) булевой алгеброй с одноэлементым носителем равным носителю и тремя известными операциями, которые в булевой логике трактуются ка конъюнкция, дизъюнкция и отрицание.

Выполняющая подстановка ищется путем последовательного присваивания значений булевых переменных. Сложность описания фронтального алгоритма и его модификации M_D-алгоритма значительно проще, чем у алгоритмов из этих семейств. В этом можно убедиться перейдя по ссылкам в авторские публикации [7].

При анализе литературных источников не обнаружено аналогов для M_D-алгоритма. Требуются исследования для сравнения его эффективности с другими. Однако, уже сейчас можно сказать, что он обладает высоким уровнем параллелизма на уровне задач, на уровне данных, на уровне алгоритмов, реализующих операции над составляющими множествами. В силу особенностей операций объединения, пересечения и дополнения универсума над конечными множествами.

Автор не ставил целью своей работы построение самого эффективного алгоритма вычисления всех выполняющих подстановок. Алгоритмы, решающие эту задачу в литературе отсутствуют, за исключением алгоритмов вычисления таблиц истинности. Целью работы было построение алгоритма вычисления семантичесокого значения конъюнкции посылок в виде КППФ, который бы заканчивал вычисления за приемлемое время, для достаточного для решения практических задач числа модельных множеств[8] при решении задач верификации логического следования, сложность которых не позволяет их решать современными системами логического вывода.

Для нахождения приемлемой верхней границы числа модельных множеств необходимо провести эксперименты, а для этого нужно реализовать алгоритм программно.

---

[6] их обычно трактуют как $\{1\}$ или просто 1 и $\leqslant$, знаком 0 заменяют пустое множество

[7] https://ru.wikipedia.org/wiki/Алгоритм CDCL, https://ru.wikipedia.org/wiki/DPLL.

[8] автор считает, что таким числом является число ≈ 256

Список использованных источников

[1] Сметанин Ю. М. *Фронтальный алгоритм решения SAT задачи* // Программные системы: теория и приложения.– 2022.– Т. **13**.– № 4(55).– С. 163–179. ↑92, 94, 96, 98, 101, 102

[2] Кузичев А. С. *Диаграммы Венна.*– М.: Наука.– 1968.– 253 с. ↑89, 92

[3] Rodgers P., Stapleton G., Chapman P. *Visualizing sets with linear diagrams* // ACM Trans. Comput.-Hum. Interact..– 2015.– Vol. **22**.– No. 6.– id. 27.– 39 pp. ↑89

[4] Lamy J., Tsopra R. *RainBio: Proportional visualization of large sets in biology* // IEEE Transactions on Visualization and Computer Graphics.– 2020.– Vol. **26**.– No. 11.– Pp. 3285–3298. ↑89

[5] Сметанин Ю. М. *Верификация логического следования в неклассической многозначной логике* // Известия института математики и информатики УдГУ.– 2017.– Т. **50**.– С. 62–82. ↑88, 89, 90, 94, 95, 96

[6] Сметанин Ю. М. *Верификация логического следования с использованием исчисления конституентных множеств и соответствий Галуа* // Программные системы: Теория и приложения.– 2017.– Т. **8**.– № 2(33).– С. 69–93. ↑90, 96, 99

[7] Васильев С. Н., Жерлов А. К., Федосов Е. А., Федунов Б. Е. *Интеллектное управление динамическими системами.*– М.: Физматлит.– 2000.– ISBN 5-9221-0050-5.– 352 с. ↑89

[8] Кулик Б. А., Зуенко А. А., Фридман А. Я. *Алгебраический подход к интеллектуальной обработке данных и знаний.*– СПб.: Изд-во Политехн. ун-та.– 2010.– ISBN 978-5-7422-2836-3.– 235 с. ↑99, 101

Поступила в редакцию	29.02.2024;
одобрена после рецензирования	09.04.2024;
принята к публикации	17.04.2024;
опубликована онлайн	17.05.2024.

Рекомендовал к публикации *д.ф.-м.н. Н. Н. Непейвода*

Информация об авторе:

Юрий Михайлович Сметанин

к.ф.- м.н., зав. кафедой математического анализа ИМИТиФ УдГУ, Ижевск. Научные интересы прикладная логика, алгоритмы логико - семантического моделирования

0000-0002-2508-4939

e-mail: *gms1234gms@rambler.ru*

Декларация об отсутствии личной заинтересованности: *благополучие автора не зависит от результатов исследования.*

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article ARTIFICIAL INTELLIGENCE, INTELLIGENCE SYSTEMS, NEURAL NETWORKS

UDC 004.421+004.891.2+510.649
10.25209/2079-3316-2024-15-2-87-112

The use of distributed computing in domain modeling in universal syllogistics

Iurii Mikhailovich **Smetanin**[1]
[1] Udmurt State University, Izhevsk, Russia

Abstract. We consider the algorithmic aspects of calculations in the logical-semantic models of universal syllogistics L_{S_2}. This non-classical propositional logic is based on an algebraic system containing a Boolean algebra of sets and two relations between sets: $\subset$ and $=$. Its closest analogue is Aristotle's syllogistics, the model of which is an algebraic system with a Boolean algebra of sets and one relation $\subset$. The disadvantage of syllogistics based on an algebraic system with a single relation $\subset$ is the ambiguity of the interpretation of their formulas and atomic judgments.

In this work, by a logical-semantic model of a subject area we understand the totality of the universal syllogistic formula L_{S_2} and its semantic meaning, which is a finite set of non-negative integers. We propose an algorithm for computing semantic value of a conjunctive well-constructed formula L_{S_2}, which has a high level of parallelism at the task level, at the data level, and at the level of algorithms realizing operations on constituent sets. Due to the peculiarities of union, intersection and complement operations over finite sets, all the processes of their computation and solution of subtasks occur at the bit level and, as a rule, are efficiently implemented in algorithmic languages. In the proposed algorithm, the transition to the bit level and back is realized by a set of software tools. (*In Russian*).

Key words and phrases: Syllogistics, discrete Venn diagrams, logical-semantic models, frontal algorithm, distributed computing

2020 *Mathematics Subject Classification*: 68T30; 03B45, 68W10

For citation: Iurii M. Smetanin. *The use of distributed computing in domain modeling in universal syllogistics*. Program Systems: Theory and Applications, 2024, **15**:2(61), pp. 87–112. (*In Russ.*). https://psta.psiras.ru/read/psta2024_2_87-112.pdf

References

[1] Yu. M. Smetanin. "Front-end algorithm for solving the SAT problem", *Program Systems: Theory and Applications*, **13**:4(55) (2022), pp. 163–179 (in Russian).

[2] A. S. Kuzichev. *Venn Diagrams*, Nauka, M., 1968, 253 pp. (in Russian).

[3] Rodgers P. , G. Stapleton, P. Chapman. "Visualizing sets with linear diagrams", *ACM Trans. Comput.-Hum. Interact.*, **22**:6 (2015), id. 27, 39 pp.

[4] Lamy J. , Tsopra R. . "RainBio: Proportional visualization of large sets in biology", *IEEE Transactions on Visualization and Computer Graphics*, **26**:11 (2020), pp. 3285–3298.

[5] Yu. M. Smetanin. "Verification of the logical sequence in nonclassical multivalued logic", *Izvestiya instituta matematiki i informatiki UdGU*, **50** (2017), pp. 62–82 (in Russian).

[6] Yu. M. Smetanin. "Verification of logical consequence, using the calculus of constituent sets and correspondences of Galois", *Program Systems: Theory and Applications*, **8**:2(33) (2017), pp. 69–93 (in Russian).

[7] S. N. Vasil'ev, A. K. Zherlov, E. A. Fedosov, B. E. Fedunov. *Intelligent Control of Dynamic Systems*, Fizmatlit, M., 2000, ISBN 5-9221-0050-5, 352 pp. (in Russian).

[8] B. A. Kulik, A. A. Zuenko, A. Ya. Fridman. *Algebraic Approach to Intellectual Processing of Data and Knowledge*, Izd-vo Politexn. un-ta, SPb., 2010, ISBN 978-5-7422-2836-3, 235 pp. (in Russian).

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья искусственный интеллект, интеллектуальные системы, нейронные сети

УДК 004.932.72
10.25209/2079-3316-2024-15-2-113-137

Применение нейронных сетей сиамской архитектуры в задачах классификации продуктов различных категорий на прилавках универсама

Александр Владимирович **Смирнов**[1]✉, Игорь Петрович **Тищенко**[2]

Аннотация. В настоящей работе представлено исследование на тему применения нейронных сетей сиамской архитектуры в задачах классификации различных продуктов питания на прилавках универсальных магазинов. Сиамские сети – это особый класс нейросетевых архитектур, объединяющий в себе две свёрточные подсети. Его часто используют в задачах сопоставления объектов, поскольку по сравнению с традиционными свёрточными нейронными сетями он не требует большого количества обучающих данных. В ходе работ сгенерирован собственный набор данных, включающий пять различных категорий продуктов. В результате удалось достичь точности в 97.5% при обучении.

Ключевые слова и фразы: сиамские нейронные сети, набор данных, продукты питания

Для цитирования: Смирнов А. В., Тищенко И. П. *Применение нейронных сетей сиамской архитектуры в задачах классификации продуктов различных категорий на прилавках универсама* // Программные системы: теория и приложения. 2024. **Т. 15**. № 2(61). С. 113–137. https://psta.psiras.ru/read/psta2024_2_113-137.pdf

Введение

Среди подходов к распознаванию объектов на изображении можно выделить два основных направления: непосредственное распознавание объекта по его характерным чертам и особенностям, и сравнение объекта интереса с шаблоном заранее известного класса объектов. С непосредственным распознаванием объектов хорошо справляются классические свёрточные нейронные сети. Тем не менее, для достижения высоких показателей точности распознавания необходимо создать обширный набор данных для каждого из исследуемых классов, а также в зависимости от от формы и внешнего вида объектов интереса требуется использовать сеть более сложной архитектуры.

С другой стороны, есть ряд задач, где в непосредственном определении класса/типа объекта нет необходимости. Например, если необходимо определить автомобиль на изображении или мотоцикл, то совершенно не важна марка конкретного автомобиля или мотоцикла. Для решения подобных задач часто используется нейронная сеть сиамской архитектуры.

Нейронные сети сиамской архитектуры были впервые представлены в начале 1990-х годов в работе [1] для решения проблемы проверки подлинности подписей как задачи сопоставления изображений. Авторы использовали набор данных, состоящий из образцов подписей 219 человек, часть из которых была специально подделана. В ходе работ авторы также провели экспериментальное исследование нескольких архитектур используемых подсетей и получили результат в 95.5% успешного определения подлинности подписей.

Сейчас нейронные сети сиамской архитектуры используются для анализа графических данных различных источников. Так в работе [2] авторы поставили цель изучить и объяснить концепцию сходства изображений с использованием сиамских сетей и технологии Grad-CAM[1]. В результате им удалось получить область активации нейронной сети на изображении, а также составить ряд рекомендаций по созданию собственного набора данных для более точного обучения сети.

Работа [3] содержит экспериментальное исследование обучения сиамских нейронных сетей для классификации объектов с минимальным набором обучающих данных. Здесь авторы используют различные метода улучшения точности обучения среди которых можно выделить трансферное обучение и имитацию процесса ансамблевого обучения. В итоге им удалось

[1] *What is the Grad CAM method?*

добиться точности классификации в 69%–78% при использовании от 5 до 20 обучающих изображений на класс.

Статья [4] также посвящена классификации объектов с применением сиамских нейронных сетей. Однако, в данной работе авторы рассматривают в первую очередь иерархию классов и используют некоторые модификаторы, среди которых можно отметить матрицы затрат и убывающий иерархический множитель. В экспериментах участвовали следующие наборы данных: CIFAR-100[2], Amazon Product Reviews[3] и Fashion MNIST[4]. В результате была получена точность классификации от 64% до 90% в зависимости от набора данных, что сопоставимо с точностью (65%–84%) полученной при стандартной классификации.

В статье [5] сиамские сети используются для распознавания патернов человеческой деятельности. Авторы применяют такие технологии как тройная функция потерь и Bi-LSTM[5] для распространения данных внутри сети в обоих направлениях. Таким образом, на наборе данных UCI HAR[6] авторам удалось получить точность равную 93.9%.

В работе [6] рассматривается применение сиамских нейронных сетей для классификации изображений горных пород. Отличительной особенностью данной работы является разработанная авторами архитектурная надстройка пространственного преобразования (STN), которая используется для извлечения критически важных областей изображений. Благодаря этому, авторы достигли точности классификации в 87.14%–97.75%, что в среднем на 2% больше, чем при использовании классических глубинных нейронных сетей.

Иное применение нейронных сетей сиамской архитектуры представлено в статье [7]. Здесь целью авторов было определение принадлежности сгенерированного изображения какому-либо нейросетевому генератору. В частности, авторы использовали сгенерированные изображения лиц людей, для чего им понадобился набор данных, созданный при помощи 10 различных генеративных нейросетевых архитектур. Полученная в итоге точность составила более 90%.

Также сиамские сети используются для анализа сенсорных данных. Например, в работе [8] сеть используется для идентификации подключённой

[2] *CIFAR-100*
[3] *Amazon Product Reviews*
[4] *Fashion MNIST*
[5] *Bidirectional LSTM*
[6] *UCI-HAR*

нагрузки. В данном подходе Сеть обучается на изображениях графиков зависимости потребляемого тока от времени различных бытовых приоров (холодильник, настольный ПК, дрель и т.д.). В качестве подсетей авторы использует Lenet-5[7]. Тестирование предложенного метода проводилось на наборах данных WHITED[8] и PLAID[9]. В результате была получена точность идентификации более 98%.

Нейронные сети сиамской архитектуры уже достаточно широкого применяются для обработки медицинских данных. Любые данные, имеющие эквивалент в виде изображений (данные спектроскопии, снимки МРТ, эндоскопические изображения и т.д.) могут быть проанализированы искусственными нейронными сетями. К примеру, статья [9] посвящена классификации клинически значимых бактерий на основе Рамановская спектроскопии с применением сиамских нейронных сетей. Авторы данной стать использовали набор спектральных данных Рамана, который содержит 5420 (6 видов бактерий, около 900 спектров на класс). Точность классификации оставила около 83.61%, что сопоставимо с точностью при полученной при использовании глубинных нейронных сетей – 84.13%.

В другой работе [10] анализируются небольшие образцы эндоскопических изображений для классификации лейкоплакии голосовых связок[10] с использованием сиамской нейронной сети. Предложенный авторами метод показал лучшую точность классификации – 97.56%, в сравнении с нейронными сетями архитектур AlexNet[11], VGG Net[12], Google Inception[13], ResNet[14] и др.

Статья [11] затрагивает проблему оценки вариабельности контраста различных МРТ-сканеров. Разные МРТ-сканеры имеют различные параметры конфигурации из-за чего полученные снимки одной и той же части тела могут иметь расхождения, что усложняет диагностику и создаёт проблемы с точки зрения воспроизводимости снимков. Целью авторов данной работы является определить принадлежность снимка конкретному МРТ-сканеру. Для этого они используют сиамскую сеть с тройной

[7] *Архитектура LeNet-5*
[8] *WHITED*
[9] *PLAID*
[10] *Лейкоплакия гортани*
[11] *AlexNet — свёрточная нейронная сеть для классификации изображений*
[12] *VGG Very Deep Convolutional Networks*
[13] *Advanced Guide to Inception v3*
[14] *ResNet (34, 50, 101): «остаточные» CNN для классификации изображений*

функцией потерь[15] в купе с классическим методами классификации. В результате, авторам удалось добиться точности в 93.15%.

Некоторое распространение получило применение сиамских нейронных сетей для анализа снимков Дистанционного Зондирования Земли (ДЗЗ)[16]. Так в работе [12] применяется глубинная сиамская сеть с ручным извлечением признаков для классификации гиперспектральных изображений. Основной проблемой авторы называют ограниченную доступность меченых данных ДЗЗ. Архитектура сиамских нейронных сетей позволяет решить проблему недостаточного количества обучающих данных. Таким образом, экспериментальные результаты показали точностью 95.17% и 93.25% для наборов данных Pavia U[17] и Indian Pines[18].

Анализируя работы, связанные с применением сиамских нейронных сетей в задачах классификации и распознавания объектов, можно сделать вывод о том, что при использовании данного подхода можно добиться точности сопоставимой с той, что достигается при использовании классических свёрточных нейронных сетей, при этом используя сравнительно небольшие наборы данных.

В настоящей статье будет рассмотрен метод классификации продуктов различных категорий на прилавках универсамов с использованием сиамских нейронных сетей, при достигнутой точности обучения в 97.5%.

1. Постановка задачи. Актуальность

Настоящая статья посвящена исследованию и разработке метода применения нейронных сетей сиамской архитектуры в задачах распознавания различных продуктов питания на прилавках универсальных магазинов. Основная задача данного исследования заключается в разработке работоспособной концепции гибкой подсистемы распознавания различных продуктов/товаров, которая в последствии может быть применена в так называемых автоматизированных роботах-мерчандайзерах.

Исходя из определения профессии, мерчандайзер[19] это – товаровед или помощник товароведа, человек, представляющий производственную или торговую компанию в торговых сетях. Отвечает за выкладку товара, установку сопутствующего необходимого оборудования, размещает

[15] *Реализация функции потерь Triplet Loss в Python*
[16] *Дистанционное зондирование Земли*
[17] *Pavia University Hyperspectral Dataset*
[18] *Indian Pines Hyperspectral Dataset*
[19] *Мерчандайзер*

POS-материалы. Основная задача мерчандайзера – контроль наличия всего ассортимента компании на полках магазина и расположение его в наиболее благоприятных для покупки местах. Очевидно, что мерчандайзер выполняет важную роль в функционировании магазина и формировании прибыли, так как отсутствие на прилавках в нужный момент определённых товаров стоит продавцам миллиардов долларов в год.

Благодаря развитию робототехники и алгоритмов компьютерного зрения часть операций человека-мерчандайзера можно переложить на автоматизированного специального робота. Робот-мерчандайзер представляет собой платформу с вертикальной камерой, передвигающуюся между стеллажами и холодильниками, и выполняющую фото-видео съёмку. Камера робота обычно обладает гораздо более высоким разрешением, чем стационарная камера. С точки зрения экономики, это решение показало себя наиболее эффективным, потому что на магазин достаточно одного робота.

В данный момент применение роботов-мерчандайзеров не носит массовый характер, однако, их уже применяют в крупнейшей мире торговой сети Walmart[20] в США (рисунок 1), а также в сети Auchan[21]

Рисунок 1. Робот-мерчандайзер в сети гипермаркетов Walmart

в Португалии (рисунок 2).

[20] *Новый сотрудник Walmart – робот-сканер*

[21] *Четыре новых способа контролировать товар на полке*

Рисунок 2. Робот-мерчандайзер в сети гипермаркетов Auchan

Предпосылки к применению подобных роботов также наблюдаются и в России в сети магазинов «Пятёрочка»[22].

В связи с ростом популярности роботизированных систем в повседневной жизни, в частности, роботов-мерчандайзеров для крупных торговых сетей, появляется спрос на технологии распознавания и классификации товаров, что подтверждает актуальность исследования, представленного в настоящей работе.

2. Сиамские нейронные сети. Архитектура, обучение и тест

2.1. Архитектура используемой сети

Сиамские нейронные сети (Siamese Neural Network, SNN) – это класс архитектур нейронных сетей, предназначенных для сравнения и измерения сходства между парами входных выборок. Термин «сиамский» происходит от идеи, что архитектура сети (рисунок 3) состоит из парных нейронных сетей (часто свёрточных), которые идентичны по структуре и имеют одинаковый набор весовых коэффициентов. Каждая сеть обрабатывает одну входную выборку из пары, а их выходные данные сравниваются для определения сходства или различия между двумя экземплярами входных данных.

Сиамские сети предназначены для решения задач, где прямое обучение с помеченными выборками ограничено или затруднено, поскольку сеть после обучения способна различать похожие и непохожие экземпляры, не требуя явных меток классов.

[22] *НПО «Андроидная техника» создает роботов для розничных магазинов*

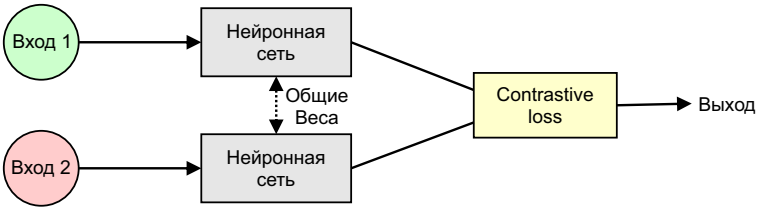


Рисунок 3. Пример архитектуры сиамской нейронной сети

Архитектура сиамской сети обычно состоит из трех основных компонентов: используемая подсеть, метрика сходства и функция контрастных потерь.

Подсеть является основным компонентом архитектуры сиамской сети. Она отвечает за извлечение ключевых признаков и особенностей из входных выборок. Обычно подсети представляют собой свёрточные нейронные сети, состоящие из слоёв свёртки и/или полносвязных слоёв, которые обрабатывают входные данные и создают некоторый дескриптор исследуемого объекта. Распределяя одинаковые веса между идентичными подсетями, модель учится извлекать схожие признаки и особенности для аналогичных входных данных, что позволяет эффективно их сравнивать.

Метрика сходства используется для сравнения сгенерированных дескрипторов и измерения сходства или различия между двумя входными данными. Выбор метрики сходства зависит от конкретной задачи и характера входных данных. Обычно в качестве метрики сходства используется евклидово расстояние, косинусное сходство или коэффициент корреляции.

Функция контрастных потерь – это функция, основанная на подсчёте расстояния, в отличие от более традиционных функций прогнозирования ошибки. Данная функция используется для анализа входных данных, при котором две схожие точки имеют малое евклидово расстояние, а две различные точки имеют большое евклидово расстояние.

Функция контрастных потерь высчитывается по формуле

$$(1-Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{\max(0, m - D_W)\}^2,$$

где D_W (евклидово расстояние) определяется по формуле:

$$\sqrt{\{G_W(X_1) - G_W(X_2)\}^2}.$$

Среди преимуществ сиамских нейронных сетей можно выделить следующее:

✓ Нет необходимости в большом наборе данных для обучения. Сеть способна обучиться на небольшом наборе данных, что также позволяет компенсировать дисбаланс классов.

✓ Устойчивость к аффинным преобразованиям изображений, таким как поворот и масштабирование.

✓ Семантическое сходство. Нейронная сеть сиамской архитектуры анализирует пространство признаков чтобы сформировать представление о схожести/различии изображений вместо того, чтобы просто извлекать статические признаки с помощью операции свёртки.

К недостаткам сиамских нейронных сетей можно отнести:

– Требуется больше времени на обучение по сравнению с традиционными свёрточными нейронными сетями.

– Не предоставляет данные о вероятности определения объекта к какому-либо классу.

Для выполнения поставленных задач по распознаванию и классификации продуктов различных категорий использовалась нейронная сеть сиамской архитектуры с двумя идентичными свёрточными подсетями, которые содержат три слоями свёртки, три полносвязных слоя и имеют следующую конфигурацию:

```
(cnn1): Sequential
Conv2d(3, 96, kernel_size=(11, 11), stride=(4, 4))
ReLU(inplace=True)
MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
Conv2d(96, 256, kernel_size=(5, 5), stride=(1, 1))
ReLU(inplace=True)
MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil\_mode=False)
Conv2d(256, 384, kernel_size=(3, 3), stride=(1, 1))
ReLU(inplace=True)
    (fc1): Sequential
Linear(in_features=384, out_features=1024, bias=True)
ReLU(inplace=True)
Linear(in_features=1024, out_features=256, bias=True)
ReLU(inplace=True)
Linear(in_features=256, out_features=2, bias=True)
```

2.2. Создание набора данных для обучения и тестирования

Для создания обучающего и тестового набора данных использовались фотографии прилавков и стеллажей (рисунок 4), находящихся

Рисунок 4. Пример фотографий, используемых для создания набора данных

в популярных сетевых универсамах/супермаркетах.

Поиск фотографий осуществлялся в интернете, что является относительно простым способом поиска различных изображений, однако, ракурс съёмки, разрешение и общее качество изображений может сильно варьироваться.

Исходя из характерных особенностей внешнего вида и относительной популярности некоторых продуктов, набор обучающих и тестовых данных содержал следующие классы объектов:

- Напитки типа «Кола» (cola) – напитки популярного бренда и всевозможные аналоги. Имеет вид пластиковой бутылки с тёмным, почти чёрным содержимым и этикеткой красного или оттенков красного цвета.
- Молочные и кисломолочные продукты (milk) – молочные и кисломолочные продукты различных брендов. Имеет вид пластиковой бутылки с белым или кремово-белым содержимым и светлой этикеткой с примесью различных цветов.

- Растительные масла (oil) – растительные масла различных брендов. Имеет вид пластиковой бутылки с золотисто-жёлтым содержимым и этикеткой, также содержащей оттенки жёлтого цвета.
- Колбасные изделия (sausage) – колбасные изделия различных брендов. Имеет продолговатую цилиндрическую форму с оболочкой розового, светло-розового или бежевого цвета.
- Минеральные воды (water) – питьевая и минеральная вода различных брендов. Имеет вид пластиковой бутылки с бесцветным содержимым. Цвет бутылок и этикеток может варьироваться от светло голубого до зелёного.

Выбор данных видов продуктов обусловлен тем, что они встречаются практически в любом современном универсаме или супермаркете и имеют достаточно много схожих особенностей внешнего вида, чтобы их можно было сгруппировать в отдельные классы, которые в свою очередь будут отличаться друг от друга. Группировка была применена в следствии невозможности адекватного распределения отдельных продуктов по классам из-за отсутствия оных в нужном количестве на найденных в интернете фотографиях, а также в следствии их визуального сходства.

Всего было использовано 370 фотографий прилавков/стеллажей с различными продуктами, из которых 102 фотографии содержали объекты представленных ранее классов. Образцы объектов для классов извлекались из найденных изображений методом вырезания соответствующей области с последующем сохранением в виде отдельных изображений. Таким образом количество образцов для каждого из классов составило:

- класс «cola»: 224
- класс «milk»: 240
- класс «oil»: 126
- класс «sausage»: 225
- класс «water»: 227

Несмотря на то, что сиамские сети способны адекватно обучаться при дисбалансе классов, было принято решение о балансировании классов, что потенциально может увеличить точность обучения сети. Процедура балансирования заключалась в добавлении недостающих экземпляров класса, которыми являлись зеркальные копии случайных уже существующих экземпляров. Итого в каждом классе количество экземпляров было выровнено до 252, из которых 220 на обучение и 32

на тест. На рисунке 5 показаны примеры выборок по классам полученного набора данных.

(*а*) класс «cola»

(*б*) класс «milk»

(*в*) класс «oil»

(*г*) класс «sausage»

(*д*) класс «water»

Рисунок 5. Пример экземпляров объектов созданного набора данных

Размер экземпляров варьировался в зависимости от габаритов искомого объекта на исходном изображении. Однако, этот факт не повлиял на процесс обучения сиамской нейронной сети, так как перед подачей экземпляров на вход, они приводились к единому размеру – 100x100 пикселей. Тем не менее, габариты объектов интереса были учтены непосредственно при их распознавании.

2.3. Обучение нейронной сети сиамской архитектуры

Программирование архитектуры используемой сиамской нейронной сети, её обучение и тестирование происходило с использованием язы-

ка программирования Python и фреймворка PyTorch[23]. PyTorch – это фреймворк предназначенный для машинного обучения. Он включает в себя набор инструментов для работы с моделями, используется в обработке естественного языка, компьютерном зрении и других похожих направлениях.

Обучение используемой сиамской нейронной сети происходило со следующими параметрами:

- batch_size: 16 – размер (количество) данных, посылаемых на вход сети каждую эпоху;
- epochs: 80 – количество эпох обучения нейронной сети;
- функция потерь: ContrastiveLoss – функция контрастных потерь;
- оптимизатор: Adam – один из методов оптимизации обучения включённых в фреймворк.

Для обучения был использован 8-ми ядерный CPU AMD Ryzen 7 3700X с частотой 4 Ггц на ядро. На рисунке 6 приведён график изменения

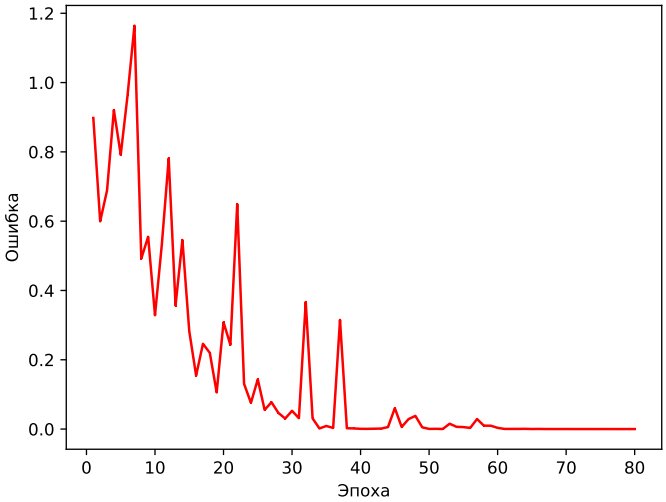


Рисунок 6. График изменения ошибки обучения

ошибки обучения в зависимости от эпох.

[23] *PyTorch*

В конце обучения ошибка была близка к 0 и составила 0.0001069. На рисунке 7 показан результат запуска обученной сиамской нейронной сети на тестовой выборке.

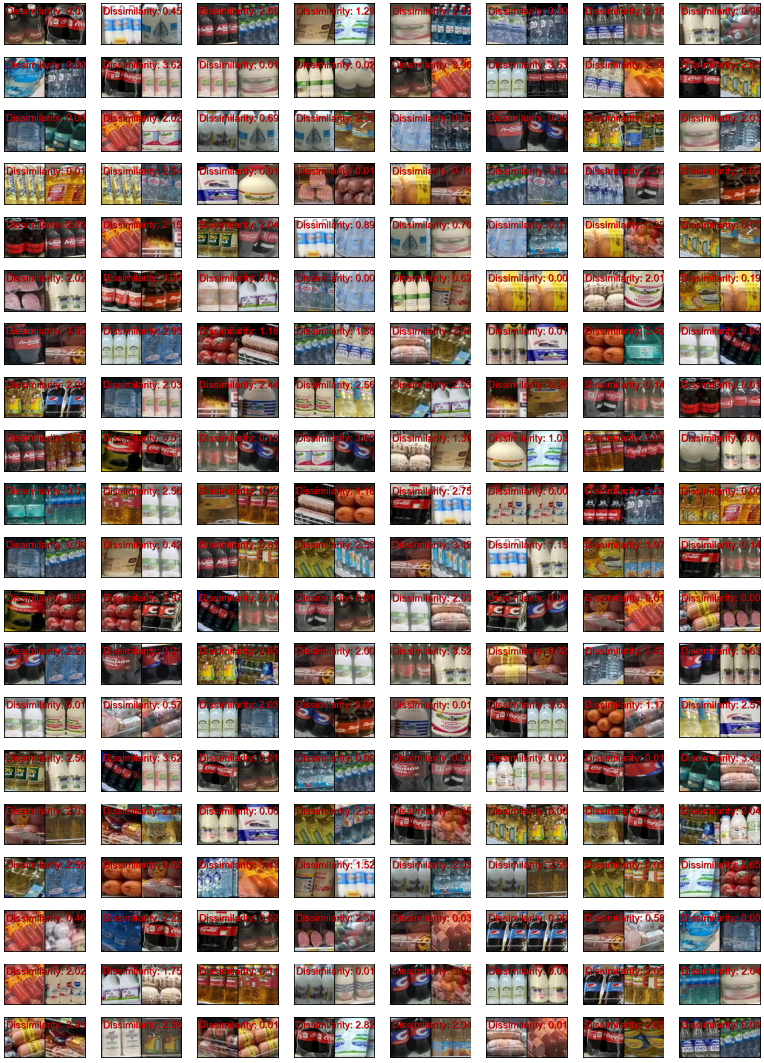


Рисунок 7. Пары входных изображений с рассчитанным показателем «непохожести»

В сеть подавались пары изображений из обучающей выборки. На выходе рассчитывался показатель «непохожести» (dissimilarity), отражающий евклидово расстояние между дескрипторами входных изображений. Таким образом, чем меньше значение данного показателя, тем более схожи объекты на входных изображениях.

2.4. Подсчёт точности обучения

В качестве метрики подсчёта точности обучения сиамской нейронной сети была использован показатель F-score[24], который высчитывается по формуле:

$$\text{Fscore} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \tag{1}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{2}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{3}$$

(4)

Здесь следует ввести следующие обозначения:

True Positive(TP) – истинно-положительное решение. Искомый объект обнаружен.

True Negative(TN) – истинно-отрицательное решение. Объект, который не является искомым не был обнаружен.

False Positive(FP) – ложно-положительное решение. Объект, который не является искомым был детектирован как искомый.

False Negative(FN) – ложно-отрицательное решение. Объект, который является искомым не был обнаружен.

Precision (точность) – отношение TP к TP + FP. Это доля объектов, названными классификатором положительными и при этом действительно являющимися положительными.

Recall (полнота) – отношение TP к TP + FN. Это то, какую долю объектов положительного класса из всех объектов положительного класса нашёл алгоритм.

Поскольку сиамская нейронная сеть на выходе даёт информацию об относительном расстоянии объектов друг от друга, что является в некотором понимании мерой схожести (меньше расстояние – более

[24] *Что такое F-score и для чего он используется?*

похожие объекты), то для определения принадлежности объектов к одному классу был введено пороговое значение расстояния. Если расстояние между объектами ниже данного порога, то они считаются принадлежащими одному классу, в противном случае разным классам. В таблице 1 перечислены пороговые значения и полученная точность обучения, рассчитанная на тестовой выборке.

Таблица 1. Точность обучения, полученная на тестовой выборке с различными пороговыми значениями расстояния

Пороговое значение	Точность по F-score
0.5	0.961
0.8	0.975
1.0	0.975
1.5	0.969

3. Экспериментальное тестирование обученной сиамской нейронной сети

3.1. Подготовка эталонных данных

Тестирование обученной сиамской нейронной сети проводилось на фотографиях прилавков и стендов популярных сетей универсамов и супермаркетов, которые содержали продукты исследуемых классов, то есть тех, что были представлены в созданном ранее наборе данных.

По мимо тестового набора изображений, также были подготовлены так называемые Ground Truth[25] данные, то есть эталонные данные необходимые для подсчёта результирующей точности классификации. Ground Truth данные представляют собой список с координатами габаритных прямоугольников искомых объектов с указанием принадлежности к определённому классу и названием тестового изображения, на котором находится искомый объект. Каждая строка этого списка содержит:

label_name – название класса;

bbox_x – координата X левого верхнего угла габаритного прямоугольника объекта;

bbox_y – координата Y левого верхнего угла габаритного прямоугольника объекта;

[25] *Ground Truth*

bbox_width – ширина габаритного прямоугольника объекта;
bbox_height – высота габаритного прямоугольника объекта;
image_name – название содержащего объект изображения;
image_width – ширина содержащего объект изображения;
image_height – высота содержащего объект изображения.

Разметка изображений для получения Ground Truth данных происходила с использованием онлайн сервиса Make Sense[26]. На рисунке 8 показан

Рисунок 8. Пример размеченных изображений. Зелёные рамки – габаритные прямоугольники Ground Truth для искомых объектов

пример размеченных изображений.

При тестировании использовался подход на основе сканирующего окна[27]. Размер сканирующего окна определялся для каждого класса отдельно. Высотой и шириной окна являлись значения среднего геометрического по высоте и ширине экземпляров класса. Шаг окна по горизонтали

[26] *Make Sence*
[27] *Принцип сканирующего окна*

и вертикали составлял 1/3 от его ширины и высоты соответственно.

В процессе тестирования, область исходного изображения под сканирующим окном подавалась на вход обученной сиамской нейронной сети в паре с эталонным экземпляром класса, выбранным случайно из тестовой выборки. Затем сеть рассчитывала показатель «непохожести» (dissimilarity) для пары входных данных, который показывал на сколько образец схож с эталоном. Если показатель dissimilarity был ниже неготорого порогового значения (подбиралось экспериментально), то образец считался экземпляром того же класса, что и эталон, и его координаты на исходном изображении, а также его ширина и высота сохранялись в отдельный список для последующего визуального отображения и подсчёта точности распознавания/классификации. На рисунке 9 показан результат работы сиамской нейронной сети.

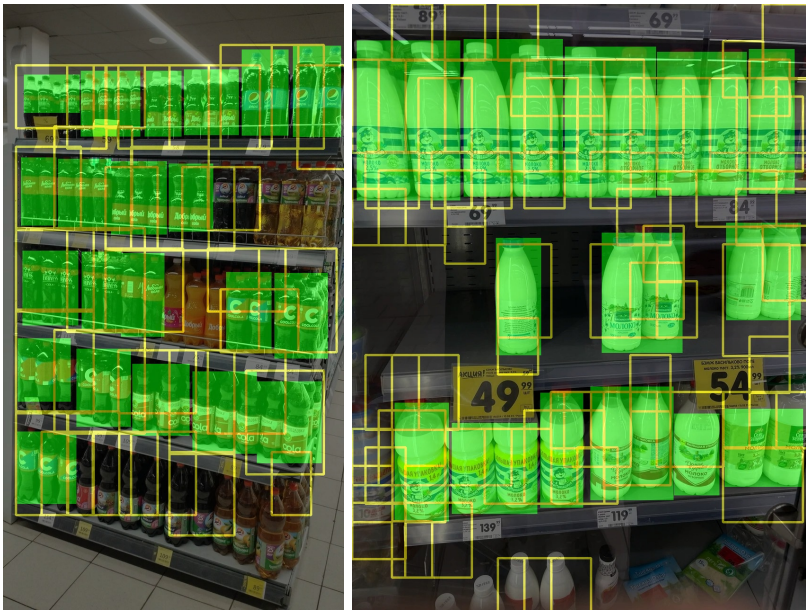


Рисунок 9. Исходные изображения после классификации продуктов: зелёные прямоугольники – объекты Ground Truth; жёлтые рамки – объекты, определённые сиамской нейронной сетью

3.2. Подсчёт точности классификации продуктов сиамской нейронной сетью

Точность классификации продуктов рассчитывалась с использованием метрики F-score, что была описана в п. 2.4 настоящей статьи. В данном случае для определения принадлежности полученных габаритных прямоугольников к эталонным данным (Ground Truth) было использовано значение IoU. Intersection over Union (IoU, пересечение по объединению), также известное как коэффициент Жаккара[28] – это число от 0 до 1, показывающее, насколько у двух объектов (эталонного и текущего) совпадает внутренний «объём» (рисунок 10).

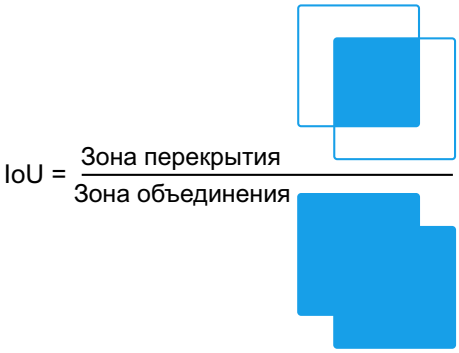


Рисунок 10. Демонстрация расчёта значения IoU

Таким образом, объект считался обнаруженным если габаритный прямоугольник его эталонных данных и габаритный прямоугольник, полученный от нейронной сети, имели значение пересечения по объединению (IoU) не менее 0.5. В виду особенностей метода сканирующего окна, одному эталонному объекту (габаритному прямоугольнику) могло соответствовать несколько габаритных прямоугольников, сгенерированных сиамской нейронной сетью. Все габаритные прямоугольники сгенерированные сетью и имеющие значение IoU >= 0.5 с эталонным, считались за успешное детектирование искомого объекта.

В таблице 2 представлены данные о точности классификации продуктов сиамской нейронной сетью, обученной на наборе данных из п. 2.2 настоящей статьи.

[28] *Jaccard index*

Таблица 2. Средняя точность обученной сиамской сети по классам, рассчитанная по метрике F-score в результате тестирования, а также показатели Precision (точность) и Recall (полнота)

Класс	F-score	Precision	Recall	Кол-во эталонных объектов	Кол-во найденных объектов
cola	0.401	0.298	0.890	265	230
milk	0.425	0.306	0.836	255	220
oil	0.577	0.477	0.787	138	120
sausage	0.277	0.177	0.786	244	200
water	0.413	0.304	0.842	298	241

3.3. Анализ результатов экспериментального тестирования

Средняя точность классификации различных категорий продуктов с применением сиамской нейронной сети по метрике F-score составила 0.419 (41.9%). Минимальная точность была зафиксирована на классе «sausage» и составила 0.277 (27.7%), максимальная на классе «oil» – 0.577 (57.7%). Следует отметить показатель Precision, среднее значение которого – 0.312, что свидетельствует о большом количестве ложных срабатываний, причиной которых могло стать присутствие объектов, которые визуально были схожи с искомыми. Например, на исходных изображения для класса «cola» были зафиксированы объекты (рисунок 11) визуально похожие на объекты класса, но по факту являющиеся другими напитками.

Однако показатель Recall находиться на достаточно высоком уровне и в среднем составляет 0.828, что в свою очередь говорит о высоком проценте детектирования искомых объектов на исходном изображении.

Также стоит перечислить ряд факторов, которые повлияли на точность классификации. К ним можно отнести следующее:

– Низкое качество используемых изображений.
– Различное разрешение используемых изображений.
– Ракурс съёмки и дистанция до искомых объектов на используемых изображениях.

Точно определить влияние каждого из перечисленных факторов достаточно затруднительно. Тем не менее, стоит учесть тот факт, что

Рисунок 11. Ложные срабатывания при выполнении классификации: 1) напитки типа «Байкал»; 2) напитки типа «Квас»

из-за специфики использования роботов-мерчандайзеров, изображения, полученные с их камер, будут лишены практически всех перечисленных недостатков. Это объясняется тем, что робот передвигается по определённой траектории и выполняет сьемку всех продуктов с одного ракурса и дистанции, используя одну или несколько камер со схожим качеством и разрешением съёмки.

Вывод

Исходя из всего вышеперечисленного, можно сделать вывод о том, что подход к классификации продуктов основанный на использовании сиамских нейронных сетей имеет по крайней мере теоретическую работоспособность, о чём свидетельствует точность по F-score полученная при обучении и равная 97.5%. Основными факторами низкой точности, полученной при экспериментальном тестировании, является разнородность и низкое качество используемых тестовых данных. При исключении данных факторов, подход, описанный в настоящей статье, может быть

использован в качестве составной части системы компьютерного зрения робота-мерчандайзера.

Список использованных источников

[1] Bromley J., Guyon I., LeCun Y., Säckinger E., Shah R. *Signature verification using a "Siamese" time delay neural network* // *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..– 1993.– Pp. 737–744. URL ↑114

[2] Livieris I. E., Pintelas E., Kiriakidou N., Pintelas P. *Explainable image similarity: integrating siamese networks and grad-CAM* // Journal of Imaging.– 2023.– Vol. **9**.– No. 10.– Pp. 224. doi ↑114

[3] Valero-Mas J. J., Gallego A. J., Rico-Juan J. R. *An overview of ensemble and feature learning in few-shot image classification using siamese networks* // Multimedia Tools and Applications.– 2024.– Vol. **83**.– No. 7.– Pp. 19929–19952. doi ↑114

[4] Понамарёв В. В., Китов В. В., Китов В. А. *Учёт иерархии классов при классификации объектов с помощью сиамских нейронных сетей* // Прикладная математика и информатика.– 2023.– № 73.– С. 38–57. URL ↑115

[5] Byung-Rae C., Binod V. *Enhancing human activity recognition with siamese networks: a comparative study of contrastive and triplet learning approaches* // Electronics.– 2024.– Vol. **13**.– No. 9.– Pp. 1739. doi ↑115

[6] Qiqi Z., Sai W., Shun T., Liangbin Y., Kunlun Q., Qingfeng G. *RockS^{2}Net: Rock image classification via a spatial localization siamese network* // Computers & Geosciences.– March 2024.– Vol. **185**.– id. 105560. doi ↑115

[7] Abady L., Wang J., Tondi B., Barni M. *A siamese-based verification system for open-set architecture attribution of synthetic images* // Pattern Recognition Letters.– April 2024.– Vol. **180**.– Pp. 75–81. doi ↑115

[8] Lingxia L., Ju-Song K., Fanju M., Miao Y. *Non-intrusive load identification based on retrainable Siamese network* // Sensors.– 2024.– Vol. **24**.– No. 8.– Pp. 2562. doi ↑115

[9] Contreras J., Mostafapour S., Popp J., Bocklitz T. *Siamese networks for clinically relevant bacteria classification based on Raman spectroscopy* // Molecules.– 2024.– Vol. **29**.– No. 5.– Pp. 1061. doi ↑116

[10] Zhenzhen Y., Botao H., Zhenghao S., Minghua Z., Shuangli D., Haiqin L., Xinhong H., Xiaoyong R., Yan Y. *Vocal cord leukoplakia classification using Siamese network under small samples of white light endoscopy images* // Otolaryngology–Head and Neck Surgery.– 2024.– Vol. **170**.– No. 4.– Pp. 1099–1108. doi ↑116

[11] Polsinelli M., Li H. B., Mignosi F., Zhang L., Placidi G. *Siamese network to assess scanner-related contrast variability in MRI* // Image and Vision Computing.– May 2024.– Vol. **145**.– id. 104997. doi ↑116

[12] Ranjan P., Girdhar A. *Deep Siamese network with handcrafted feature extraction for hyperspectral image classification* // Multimedia Tools and Applications.– 2024.– Vol. **83**.– No. 1.– Pp. 2501–2526. doi ↑117

Поступила в редакцию 22.03.2024;
одобрена после рецензирования 10.06.2024;
принята к публикации 10.06.2024;
опубликована онлайн 22.06.2024.

Рекомендовал к публикации *д.ф.-м.н. А. М. Елизаров*

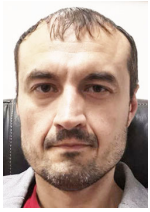
Информация об авторах:

Александр Владимирович Смирнов

Младший научный сотрудник Лаборатории методов обработки и анализа изображений, Институт Программных Систем имени А. К. Айламазяна РАН. Научные интересы: компьютерное зрение; нейронные сети; робототехника; автоматизация и управление

0000-0002-7104-1462

e-mail: asmirnov_1991@mail.ru



Игорь Петрович Тищенко

Кандидат технических наук, зав. Лабораторией методов обработки и анализа изображений, Институт Программных Систем имени А. К. Айламазяна РАН. Научные интересы: компьютерное зрение; нейронные сети; робототехника; автоматизация и управление

0000-0002-0369-0524

e-mail: igor.p.tishchenko@gmail.com

Вклад авторов: *А. В. Смирнов* – 95% (идея, методология, программное обеспечение, валидация, формальный анализ, расследование, сбор материала, курирование данных, написание черновой версии, доработка и редактирование); *И. П. Тищенко* – 5% (наставничество, администрирование).

Декларация об отсутствии личной заинтересованности: *благополучие авторов не зависит от результатов исследования.*

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article ARTIFICIAL INTELLIGENCE, INTELLIGENCE SYSTEMS, NEURAL NETWORKS

UDC 004.932.72
10.25209/2079-3316-2024-15-2-113-137

Application of neural networks of Siamese architecture in problems of classifying products of various categories on supermarket shelves

Alexander Vladimirovich **Smirnov**[1], Igor Petrovich **Tishchenko**[2]

Abstract. This paper presents a study on the application of Siamese architecture neural networks in problems of classifying various food products on the shelves of department stores. Siamese networks are a special class of neural network architectures that combine two convolutional subnets. This type of neural networks is often used in object matching problems and has an important advantage over traditional convolutional neural networks, namely the absence of the need for a large amount of training data. During the work, we generated our own data set, including five different product categories. As a result, it was possible to achieve a tonality of 97.5% during training. (*In Russian*).

Key words and phrases: Siamese neural networks, dataset, foodstuffs

2020 *Mathematics Subject Classification*: 68T10; 68T45

For citation: Alexander V. Smirnov, Igor P. Tishchenko. *Application of neural networks of Siamese architecture in problems of classifying products of various categories on supermarket shelves*. Program Systems: Theory and Applications, 2024, **15**:2(61), pp. 113–137. (*In Russ.*).
https://psta.psiras.ru/read/psta2024_2_113-137.pdf

References

[1] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah. "Signature verification using a "Siamese" time delay neural network", *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993, pp. 737–744. URL

[2] I. E. Livieris, E. Pintelas, N. Kiriakidou, P. Pintelas. "Explainable image similarity: integrating siamese networks and grad-CAM", *Journal of Imaging*, **9**:10 (2023), pp. 224. doi

[3] J. J. Valero-Mas, A. J. Gallego, J. R. Rico-Juan. "An overview of ensemble and feature learning in few-shot image classification using siamese networks", *Multimedia Tools and Applications*, **83**:7 (2024), pp. 19929–19952. doi

[4] V. V. Ponamaryov, V. V. Kitov, V. A. Kitov. "Accounting for class hierarchy in object classification using Siamese neural networks", *Computational Mathematics and Modeling*, 2024. doi

[5] C. Byung-Rae, V. Binod. "Enhancing human activity recognition with siamese networks: a comparative study of contrastive and triplet learning approaches", *Electronics*, **13**:9 (2024), pp. 1739. doi

[6] Z. Qiqi, W. Sai, T. Shun, Y. Liangbin, Q. Kunlun, G. Qingfeng. "RockS^{2}Net: Rock image classification via a spatial localization siamese network", *Computers & Geosciences*, **185** (March 2024), id. 105560. doi

[7] L. Abady, J. Wang, B. Tondi, M. Barni. "A siamese-based verification system for open-set architecture attribution of synthetic images", *Pattern Recognition Letters*, **180** (April 2024), pp. 75–81. doi

[8] L. Lingxia, K. Ju-Song, M. Fanju, Y. Miao. "Non-intrusive load identification based on retrainable Siamese network", *Sensors*, **24**:8 (2024), pp. 2562. doi

[9] J. Contreras, S. Mostafapour, J. Popp, T. Bocklitz. "Siamese networks for clinically relevant bacteria classification based on Raman spectroscopy", *Molecules*, **29**:5 (2024), pp. 1061. doi

[10] Y. Zhenzhen, H. Botao, S. Zhenghao, Z. Minghua, D. Shuangli, L. Haiqin, H. Xinhong, R. Xiaoyong, Y. Yan. "Vocal cord leukoplakia classification using Siamese network under small samples of white light endoscopy images", *Otolaryngology–Head and Neck Surgery*, **170**:4 (2024), pp. 1099–1108. doi

[11] M. Polsinelli, H. B. Li, F. Mignosi, L. Zhang, G. Placidi. "Siamese network to assess scanner-related contrast variability in MRI", *Image and Vision Computing*, **145** (May 2024), id. 104997. doi

[12] P. Ranjan, A. Girdhar. "Deep Siamese network with handcrafted feature extraction for hyperspectral image classification", *Multimedia Tools and Applications*, **83**:1 (2024), pp. 2501–2526. doi

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
Обзорная статья ПРОГРАММНОЕ И АППАРАТНОЕ ОБЕСПЕЧЕНИЕ РАСПРЕДЕЛЕННЫХ И СУПЕРКОМПЬЮТЕРНЫХ СИСТЕМ

УДК 004.272+004.382.2+004.8+004.43
10.25209/2079-3316-2024-15-2-139-473

Новое поколение GPGPU и сопутствующего оборудования: микроархитектура и производительность вычислительных систем от серверов до суперкомпьютеров

Михаил Борисович **Кузьминский**[1]✉
[1] Институт органической химии им. Н. Д. Зелинского РАН, Москва, Россия

Аннотация. Дан обзор современного состояния GPGPU с ориентацией их применения на традиционные задачи HPC (и в меньшей степени ИИ). К базовым GPGPU в обзоре отнесены Nvidia V100 и A100. В качестве GPGPU нового поколения рассмотрены Nvidia H100, AMD MI100 и MI200, Intel Ponte Vecchio (Data Center GPU Max), а также BR100 от Biren Technology. Проанализированы и сопоставлены микроархитектура и аппаратные показатели этих GPGPU, важные для задач HPC и ИИ, а также важнейших дополнительных аппаратных средств для построения вычислительных систем с применением GPGPU – центральных процессоров, специализированных для работы с GPGPU нового поколения, и межсоединений. Дается краткая информация об использующих их серверах, в том числе multi-GPU, и новых применяющих эти GPGPU суперкомпьютерах, где были получены данные о достигаемой производительности при работе с GPGPU.

Кратко рассмотрены SDK фирм-производителей GPGPU и программные средства других фирм, включая математические библиотеки. Приводятся примеры, демонстрирующие важные для достижения максимальной производительности средства широко используемых моделей программирования, способствующие при этом непереносимости программных кодов на другие модели GPGPU.

Особое внимание обращено на возможности применения тензорных ядер и их аналогов в современных GPGPU разных фирм. Это относится и к расчетам с пониженной (относительно стандартного для HPC формата FP64) и смешанной точностью, актуальным вследствие резкого роста достигаемой производительности при их использовании в тензорных ядрах GPGPU. Анализируются данные о достигаемой ими реальной производительности в тестах и приложениях для HPC и ИИ. Вкратце рассматривается и применение в GPGPU современных библиотек пакетной линейной алгебры, в том числе для HPC-приложений. (*Связанные тексты статьи на русском и на английском языках*)

Ключевые слова и фразы: GPGPU, V100, A100, H100, Grace, GH200 Grace Hopper, MI100, MI200, Ponte Vecchio, Data Center GPU Max, BR100, CUDA, HIP, DPC++, Fortran, производительность, HPC, ИИ, глубокое обучение

Благодарности:

Для цитирования: Кузьминский М. Б. *Новое поколение GPGPU и сопутствующего оборудования: микроархитектура и производительность вычислительных систем от серверов до суперкомпьютеров* // Программные системы: теория и приложения. 2024. **Т. 15**. № 2(61). С. 139–473. (*Русс.+англ.*). https://psta.psiras.ru/read/psta2024_2_139-473.pdf

Введение

Широко распространенной особенностью современных вычислительных систем от отдельных серверов до суперкомпьютеров является применение гетерогенного построения серверов и узлов кластеров, очень часто содержащих не только процессоры (ЦП), но и акселераторы, в первую очередь GPU, области применения которых стремительно расширяются. Здесь имеются в виду GPGPU, но далее будет применяться сокращение GPU.

Актуальность и области применения GPU. В настоящее время GPU активно применяются для широкого ряда очень важных задач, включая высокопроизводительные вычисления (HPC) и ИИ (к задачам ИИ в данном обзоре отнесены также все задачи любых типов машинного обучения, но особенно применение GPU актуально для глубокого обучения). Кроме того, расширяется применение нескольких GPU в одном сервере (multi-GPU). Все это связано с основным направлением роста производительности вычислительных систем главным образом за счет сильного роста числа ядер и соответственно распараллеливания на них. Все возрастающая со временем важность энергоэффективности также способствует ориентации на применение GPU, содержащих гораздо больше функционально более простых ядер, чем в ЦП, но с пониженной частотой. Так, из 50 суперкомпьютеров-лидеров списка Green500 за июнь 2023 года только четыре не использовали GPU. При этом два из них – японские суперкомпьютеры NA-J2 и MN-3 применяли специализированные редко используемые многоядерные процессоры (сопроцессоры), а два других – базировались на многоядерных ARM-процессорах Fujitsu A64FX [1]. Другим важным плюсом GPU является достижение высокой плотности упаковки больших вычислительных ресурсов, что особенно ярко проявляется в серверах, содержащих по несколько GPU (multi-GPU).

Что касается часто указываемого применения GPU в ЦОД, то термин ЦОД теперь практически заменил собой ранее используемый термин вычислительный центр, который может рассматриваться и как одна из частей ЦОД [2]. Реально в ЦОД часто предполагается применение GPU для указанных выше задач HPC и ИИ, а сам термин ЦОД ориентируется прежде всего на применение облачной технологии. Далее в тексте упоминания про ЦОД относятся именно к облачной технологии, задачи которой в обзоре не обсуждаются. Здесь рассматриваются конкретные тесты производительности и приложения для HPC и ИИ, хотя GPU стали отмечаться как основной ускоритель и в других областях, например, сортировок при работе с базами данных [3].

В качестве иллюстрации распространенности применения современных GPU можно воспользоваться суперкомпьютерным списком Top500 [4], поскольку он дает интересные статистические данные. В Top500 мировые лидеры производительности применяют GPU уже давно. Ярким исключением из этого правила в последние годы был японский суперкомпьютер Fugaku, узлы которого были гомогенные и содержали только ЦП – A64FX. Он возглавлял этот список больше двух лет. Вероятно, такому успеху Fugaku способствовало достаточно большое число ядер (48 вычислительных) в A64FX [5]. Однако в 2022 году появился новый китайский суперкомпьютер Sunway (преемник занимающего 7 место в Top500 Sunway TaihuLight, который в [6] отнесен к «пред-экзамасштабным»), содержащий в узлах гетерогенные 260-ядерные процессоры SW26010 – без GPU [6]. В качестве другого также не очень широко применяемого альтернативного GPU варианта можно упомянуть и огромные специализированные для задач ИИ процессоры Cerebras WSE-2, содержащие 850 тысяч ядер [7]. Однако WSE-2 не поддерживают точности больше, чем FP32, а суперкомпьютер Andromeda на их базе [8] в списке Top500 соответственно отсутствует.

Статистические данные списка Top500 июня 2020 года [9] указали на применение акселераторов в 26,6% суперкомпьютеров из Top500 (в 20,2% суперкомпьютеров применялись Nvidia V100). В июньском списке 2023 года акселераторы применялись в 32,4% всех суперкомпьютеров (в 14,2% применялись Nvidia V100, в 14,2% применялись Nvidia A100, в 2% – AMD MI250X, в 1% – Nvidia H100) [4]. Однозначное современное лидерство в Top500 GPU от Nvidia и сегодня очевидно и предсказуемо на ближайшее будущее. Все приводимые далее в обзоре данные относятся к июньскому списку Top500 2023 года, и по умолчанию под списком Top500 далее имеется в виду этот июньский список.

Особенности ближайших ожидаемых GPU. В настоящее время становится возможной интеграция в одном корпусе ЦП и GPU. Для персональных компьютеров с обычными графическими процессорами аналогичная интеграция с ЦП давно известна, например, в виде AMD APU (Accelerated Processing Unit), а здесь имеется в виду интеграция серверного ЦП с GPU. У AMD такая интеграция в APU предполагается в MI300 [10]. Intel говорила про свой план объединения чиплетов x86 совместно с чиплетами GPU под названием Falcon Shores еще в 2022 году [11, 12], но его реализация потребует не один год. В определенном смысле аналогичная разработка от Nvidia, Grace Hopper [13–15], появится на рынке раньше. Все это направление – возможный путь активизации применения собственно GPU.

Ограничения и трудности применения GPU. Надо иметь в виду, что GPU установлены только в 32,4% всех суперкомпьютеров из Top500 [4] (в июньском списке 2022 года было 30,2%). Выполнение расчета исключительно на GPU связано с применением в них высокоскоростной памяти, но имеющей не очень большую (и фиксированную) емкость по сравнению с возможной емкостью оперативной памяти серверов. Для определенных исходных данных приложений (объектов исследования) это может вообще вызвать неэффективность работы на GPU. Поэтому, например, в руководстве по средствам специализированного распараллеливания на GPU Nvidia, CUDA [16], есть раздел, посвященный превышению нужной емкости памяти доступному на GPU объему памяти. В современных версиях GPU CUDA обеспечивает и возможность работы с виртуальной памятью [16]. Однако ясно, что реальная работа с виртуальной памятью может привести к сильным потерям производительности.

Расчет на GPU предполагает использование приложений, которые прекрасно распараллеливаются на большом числе ядер. Классическим примером этого являются задачи молекулярной динамики и, особенно — задачи ИИ. К сожалению, это может не выполняться для определенных приложений или даже областей HPC. Поэтому в руководстве по настройке применяющих CUDA приложений (для используемой в A100 архитектуры Nvidia Ampere) [17] первым пунктом рекомендаций указано нахождение способа распараллелить последовательный код, что может означать необходимость создания новых усовершенствованных алгоритмов, допускающих распараллеливание там, где в «естественном» алгоритме оно могло бы отсутствовать. Это может не очень отвечать и ожиданиям специалистов в соответствующих областях HPC, которые часто могут самостоятельно программировать приложения для этой области.

Поскольку эффективное применение GPU предполагает очень высокий уровень масштабируемости распараллеливания, это требует и применения SDK, специализированных для GPU, а может и увеличения объема исходного кода. Многие приложения HPC такому уровню распараллеливания исходно не отвечали. Кроме того, оптимизация до высокого ожидаемого уровня производительности GPU часто требует большой ручной работы, что оказывается особенно сложно при переносе кода с одного типа GPU на другой. Все это усложняет работу программистов и может вызывать у них определенное отторжение.

Ситуация в определенной степени упрощается в случаях наличия небольших частей программы, лимитирующих производительность (в мире GPU они становятся программными ядрами), которые часто являются

типовыми математическими задачами. И с течением времени все больше HPC-приложений формируют возможность их работы на GPU. В этом направлении двигаются, например, и квантовохимические программные комплексы, давно работающие и на суперкомпьютерах. Тем не менее, для наиболее массовых из применяемых там современных методов без явного неэмпирического учета электронной корреляции слишком большие времена выполнения могут быть связаны с несколькими в математическом плане разными типами вычислений (и не всегда относящихся к широко распространенным в математическом плане; особенно это так для применения гауссовских базисных функций), что требует соответственно гораздо большего объема программирования. Для расчета распространенным квантовохимическим методом DFT в базисе плоских волн демонстрация возможных времен выполнения различными частями программы приведена, например, в [18].

Другим важнейшим вопросом для использования GPU являются стоимостные показатели. Если не стоит задача любой ценой добиться приемлемого времени выполнения (что, возможно, достижимо только с применением GPU), то актуальным становится вопрос, насколько увеличивается стоимость компьютера при добавлении в его состав GPU, и насколько при этом возрастает производительность приложения.

В качестве иллюстрации укажем данные об ускорении при работе известного программного комплекса Quantum Espresso, ориентированного на расчеты в базисе плоских волн квантовохимическим методом DFT. Иллюстрация приложением квантовой химии, а не популярной на GPU молекулярной динамики, выбрана здесь специально – задачи квантовой химии давно решаются и на суперкомпьютерах, но возможности квантовохимических расчетов на GPU стали появляться позднее, чем в классической молекулярной динамике – это сложнее в реализации, и достигаемые ускорения часто более маленькие. Расчеты по Quantum Espresso 6.5 были проведены на сервере с 18-ядерным Intel Xeon E5-2697 v4 (2,3 ГГц), и добавление V100 давало ускорение в диапазоне 1,4–3,7 раза [19]. Достигаемое ускорение, естественно, зависит от рассчитываемого объекта. К тому же эта модель Xeon начала выпускаться Intel еще в начале 2016 года, и время расчета сопоставлялось с использованием только одного процессора.

Что касается цен на GPU нового поколения – это относится, естественно, к платам с GPU (например, OAM-модуля), то они в обзоре не обсуждаются, поскольку соответствующие «официальные» (например, рекомендуемые производителем) цены на такую новую аппаратуру обычно не доступны.

При этом надо иметь в виду, что применение GPU часто дает более высокую энергоэффективность, и при этом требует относительно небольшой площади, так что в оценках GPU лучше использовать не просто цену, а совокупную стоимость владения (TCO).

Современные реалии роста применения GPU. Все указанные выше возможные затруднения постепенно решаются путем создания новых методов расчета и алгоритмов, новых программных моделей для SDK, а также за счет усовершенствования аппаратуры GPU. Естественно, это отражается и в нарастании числа работающих на GPU приложений. Ареал применения GPU постоянно растет, что, естественно, наиболее ярко проявляется при работе с GPU от Nvidia и было убедительно продемонстрировано на последних конференциях GTC 34 (2022 год) и 35 (2023 год).

С течением времени и число попадающих в Top500 суперкомпьютеров с GPU возрастает — с GPU проще получить высокую производительность в тесте HPL. В наиболее мощных (по оценкам теста HPL) суперкомпьютерах мира GPU обычно используются. В первых двадцати лидерах Top500 GPU отсутствуют только в трех суперкомпьютерах (хотя в китайском Tianhe-2A, замыкающем первую десятку, также используется акселератор Matrix-2000, но это не GPU); процент использования GPU уменьшается далее — при рассмотрении большего числа суперкомпьютеров.

Все это становится слабо значимым по сравнению с ростом использования ИИ, охватывающего все новые области применения — это в первую очередь и определяет требования к GPU (рынок HPC по сравнению с ИИ пренебрежимо мал). Современные суперкомпьютеры, входящие в Top500, также становятся ориентированными на задачи ИИ.

В обзоре современного мирового рынка графических процессоров, проведенном известным китайским аналитиком электронной промышленности Хэ Личжуном, также предполагается продолжение роста этой отрасли [**20**].

Актуальность обзора, выбор рассматриваемых GPU и областей их анализа. Общим современным обзором разных типов акселераторов, включая и GPU, можно считать [**21**] из известной европейской серии BPG (Best Practice Guide). Сегодня GPU характеризуются сверхбыстрым развитием, и можно говорить уже о появлении GPU нового поколения. В обзоре анализ производительности GPU ориентируется в первую очередь на задачи HPC. Имеются публикации, в которых реализуется слияние традиционных для HPC областей с ИИ, например, квантовой молекулярной динамики (QMD) и ИИ [**22**], или вычислительной гидродинамики и

ИИ [23]. Однако объединение в настоящее время традиционных задач HPC с методами ИИ может оказаться и не нужным (например, для QMD – см. [24]).

Однако в настоящее время наблюдается также интеграция HPC, задач ИИ и обработки больших объемов данных (в качестве примера работ за последние 2 года можно указать [25–30]), и разговор идет уже о будущих тестах производительности для этой области [31]. В качестве иллюстрации активного продвижения работ в этом направлении можно отметить и объединение в новые команды знаменитых для HPC разработчиков средств распараллеливания mvapich2 в университете Огайо (США), где теперь создаются работающие поверх mvapich2 программные средства – High-Performance Deep Learning (HiDL) [32] и High-Performance Big Data (HiBD) [33].

Учитывая потенциально широкое применение ИИ в коммерческой сфере, и соответствующую усиленную ориентацию на ИИ современных GPU [34], в данном обзоре задачи ИИ учитываются для оценок производительности, хотя статья нацелена в первую очередь на традиционные HPC.

Актуальность анализа современных GPU еще возрастает в связи с появлением новейших GPU с более высокой производительностью (с их применением создаются и предполагается создавать суперкомпьютеры EFLOPS-уровня) и с необходимостью оптимального их выбора для приобретения и использования соответствующих аппаратно-программных средств. К настоящему времени GPU проделали очень большой путь развития своих архитектур и показателей производительности. Условно к современному «базовому» их поколению в данном обзоре отнесены Nvidia V100 и A100. Так выбрано и потому, что в V100 впервые стали применяться тензорные ядра, и из-за широты использования этих GPU на современных суперкомпьютерах и в серверах. Именно на V100 и A100 базируются данные про GPU обзора [21].

С уходящими V100 и A100 в данном обзоре будет проводиться сопоставление GPU нового поколения – AMD Instinct (Radeon Instinct) MI100 и семейства MI200, Nvidia Hopper (H100), Intel Ponte Vecchio (теперь Intel выпускает целую серию GPU, Data Center GPU Max, для которых указано это кодовое слово), и отчасти новейших китайских BR100 от Biren Technology. Эти GPU условно отнесены к новому поколению, в том числе потому, что именно с их применением впервые преодолен экзафлопсный барьер или планируется его дальнейшее преодоление (это относится к GPU AMD, Nvidia и Intel). BR100 включены сюда в том числе из-за существенно более высоких указанных показателей производительности

по сравнению с A100 [35]: на уровне процессоров китайские разработчики ранее не превосходили процессоры из США и Японии (например, ARM Kunpeng 920 [5] или x86-процессор Zhaoxin [36]), но появление в 2022 году процессоров SW26010pro, содержащих 390 ядер с общей пиковой производительностью больше 14 TFLOPS (по умолчанию в тексте данного обзора предполагается двойная точность, FP64) [37] и GPU BR100 дали такое высокое достижение производительности, которое, можно сказать, впервые позволило китайской индустрии превзойти показатели некоторых аналогичных продуктов США.

Актуальность сравнительного анализа AMD MI100 и MI200 с вышеуказанными GPU от Nvidia, Intel и Biren Technology представляется очевидной. AMD MI250X начали производиться в первую очередь для ставшего первым в мире экзафлопсным суперкомпьютером Frontier [38–40], но уже используются в десятке разных суперкомпьютеров из Top500, в том числе и в занимающем третье место суперкомпьютере LUMI [41, 42], и GPU реально определяют достигнутую там максимальную производительность в Top500. Известные суперкомпьютеры Summit и Sierra с V100 в узлах целый ряд лет расположены в первой десятке Top500. GPU Intel X^e-HPC Ponte Vecchio будут использоваться в суперкомпьютере Aurora Аргоннской национальной лаборатории США, где предполагаемая пиковая производительность с двойной точностью будет превышать 2 EFLOPS [43], и в модернизации суперкомпьютера SuperMUC-NG в суперкомпьютерном центре Лейбница в Германии [44].

Кроме того, на GPU нового поколения строятся и наиболее энергоэффективные суперкомпьютеры – так, Henri с H100 возглавляет Green500, а суперкомпьютеры с MI250X занимают там все места со 2 по 7 позиции.

Актуальность сравнения MI100 с GPU Nvidia была отмечена недавно в [45], а теперь сопоставление GPU стало еще важнее в связи с появлением новых более высокопроизводительных и более энергоэффективных GPU. MI100 и MI200 уже активно применяются в HPC и ИИ. Большое внимание в обзоре обращено на данные о производительности GPU MI250X и A100, полученные с применением новейших содержащих их суперкомпьютерных систем HPE/Cray EX [46].

Новое поколение GPU отличается не только построением на них экзамасштабных суперкомпьютеров (Frontier – на MI250X [39], Aurora – на Ponte Vecchio [47], а предполагавшийся для замены занимающего 9-е место в Top500 суперкомпьютера Selene – на H100 [48]), но и использованием с ними других специализированных аппаратных средств. Прежде всего, это межсоединения (например, Nvidia NVLinik [13, 49, 50], AMD Infinity Fabric,

также отличающееся регулярным усовершенствованием версий [10], или стандарт CXL в BR100 [51]). Эти тесно связанные с GPU аппаратные средства рассматриваются в данном обзоре. Некоторые серверные процессоры – например, ARM – процессоры Grace от Nvidia для работы с GPU Hopper [13–15, 52] или AMD EPYC Zen 3 с (предположительной) поддержкой Infinity Fabric 3.0 в кристалле ввода-вывода [53] исходно предполагались для работы совместно с новыми GPU, и также рассматриваются в обзоре.

Упомянутые ЦП могут ориентироваться и на применение в HPC или ИИ без GPU. Intel Xeon серии Max [54] (имеющие кодовое слово Sapphire Rapids), которые с самого начала предполагались для применения в содержащих GPU узлах суперкомпьютера Aurora, могут использоваться независимо от GPU.

Для будущих суперкомпьютеров EFLOPS-уровня могут представлять интерес разрабатываемые в рамках европейской процессорной инициативы (EPI) акселераторы EPAC, которые базируются на RISC-V с возможностью работы с векторами длиной в 256 чисел формата FP64 [55]– но это акселераторы, не относящиеся к GPU, а EPAC пока на стадии разработки (в наличии имеется только его тестовая версия 1.0 [56]), и конструируется чиплет из ряда плиток различного типа, где EPAC – лишь одна из них [57]. Соответственно EPAC не относится к тематике данного обзора.

Обзор состоит из разделов с подразделами. В разделе 1 рассмотрены общие аппаратные и программные особенности GPU разных производителей. В разделе 2 анализируются новые китайские GPU, Birentech BR100. В разделе 3 анализируются Intel Data Center GPU Max (Ponte Vecchio). В разделе 4 анализируются GPU Nvidia: в подразделе 4.1 – A100, а в подразделе 4.2 – H100. В разделе 5 анализируются GPU AMD MI200. В заключении сделаны выводы общего характера.

Во всех разделах рассмотрены аппаратные и программные средства (SDK) для соответствующих GPU, и дается обзор имеющихся данных об их производительности. В разделах 3 и 5, и в подразделах 4.1 и 4.2 это реализовано в виде отдельных более низкоуровневых подразделов. При сопоставлении данных, в первую очередь о производительности, использовалось также сравнение с данными о производительности Nvidia V100, а в разделе 5 имеется отдельный подраздел 5.3.1 с данными о производительности AMD MI100.

В обзоре вынужденно используется весьма большое число сокращений. Автор часто дает и расшифровки общеизвестных сокращений, имея в виду возможное прочтение текста специалистами из разных областей. Список используемых в нескольких разных разделах обзора (в разделах про разные GPU) сокращений приведен в приложении.

1. Общее для GPU разных производителей

Прежде чем рассматривать конкретные GPU разных производителей, нужно хотя бы перечислить основные используемые на современных графических процессорах средства разработки программ (программные модели) с ориентацией на задачи HPC и ИИ. Максимальную производительность обычно достигают с применением, естественно, четко ориентированных на аппаратные средства производителей средств SDK (в первую очередь моделей программирования): для Nvidia – CUDA (Compute Unified Device Architecture) [16], для AMD – HIP (Heterogeneous Computing Interface for Portability) [58], часть общего программного стека ROCm (более низкоуровневые программные средства в этом разделе обзора не рассматриваются). Заметное появление на рынке альтернативных Nvidia производителей GPU повысило интерес к компонентам SDK, работающим с ускорителями разного типа. HIP уже имеет возможность работы с Nvidia GPU [58].

Среди не ориентированных на работу с GPU определенного производителя средств программирования прежде всего отметим OpenACC и современные версии OpenMP (поддержка работы с ускорителями появилась еще в OpenMP версии 4.0, а с 2021 года имеется уже спецификация 5.2 [59]). Позднее в качестве средств разработки программ для GPU и ПЛИС-акселераторов стал шире применяться OpenCL (Open Computing Language) [60], а затем и SYCL [61] – открытый стандарт для гетерогенного программирования. SYCL является разработкой Khronos Group, которая с версии SYCL 2020 базируется на C++17.

Широкое распространение может получить DPC++ (Data Parallel C++, разработка Intel) [62] – также открытый межархитектурный язык, построенный на C++ и SYCL. DPC++ использует SYCL с расширениями, которые предполагается включить в будущие версии стандарта SYCL. OpenCL, SYCL и DPC++ можно применять и для ЦП. Из них DPC++ представляется сейчас наиболее продвинутым; на его базе уже появился тест производительности [63].

Наконец, к числу упоминаемых здесь программных средств для GPU можно отнести и *Kokkos* [64, 65]. Kokkos (поддержанный в проекте Министерства энергетики США) ориентирован на экзамасштабные суперкомпьютеры, использует C++ и нацелен на отсутствие привязки к оборудованию. В нем в качестве бэк-энд можно применять, в частности, CUDA, HIP, SYCL и OpenMP. В качестве знаменитого примера приложения, применяющего Kokkos, можно указать комплекс программ LAMMPS для задач молекулярной динамики [66].

Перечисленные программные средства отражают рост применения именно C/C++ в областях HPC и ИИ. Вопрос достигаемой производительности по сравнению, например, с CUDA-программами на GPU Nvidia требует дальнейшего изучения.

Функциональная простота ядер GPU дает возможность быстрого переключения контекста нити с активной на пассивную и обратно, что не характерно для ЦП. Для высокомасштабируемого распараллеливания, применяемого при работе на большом числе функционально относительно простых ядер в обзоре на русском языке используется термин нити (для threads), что дает возможность использовать для применяемых в GPU Nvidia stream термин поток.

Многолетнее господство Nvidia на рынке GPU привело к широкому использованию в этой области терминов, введенных Nvidia. Появление нового поколения GPU, в том числе от других фирм, характеризовалось использованием ими и других терминов для тех же вещей (большинство из них SIMT-термины для API – CUDA, HIP, OpenCL и других). Соответственно имеется много публикаций и докладов на конференциях, где приводятся соответствия между терминами разных производителей, в том числе в табличной форме (см., например, [**67**, **68**]). В таблице 1 такое сопоставление сделано для целей данного обзора.

Все строки таблицы, кроме двух последних, являются API-терминами. В двух последних строках приведены термины имеющих аналогии важных аппаратных компонент GPU разных производителей. Эта таблица не включает систему терминов, используемых для BR100.

В таблице в правом столбце приведены жирным шрифтом термины, которые будут использоваться далее в настоящем обзоре в качестве общих для GPU разных производителей (хотя в разделах про конкретного производителя используется и его терминология).

Используемые производителями GPU термины могут различаться в зависимости от применения для аппаратуры или для программного обеспечения, и модифицироваться с появлением новых моделей. Так, AMD применяет термин wavefront в руководствах по архитектуре и ISA рассматриваемых в обзоре GPU этой фирмы – но в современном руководстве HIP используется только warp [**58**]. При этом появление нового поколения GPU Nvidia вызвало появление для них нового термина – кластера блоков нитей для GPU H100 [**16**] в иерархии различных уровней групп нитей.

Таблица 1. Сопоставление терминов, используемых различными производителями GPU, и модели программирования

Nvidia (CUDA)	AMD (HIP)	Intel (oneAPI/ SYCL)	Расшифровка; **используемый в обзоре общий термин** (если используется общий термин)
Thread	Work item; Thread	Work-item	Индивидуальная нить (они работают совместно в группе нитей – warp или в sub-group); **нить**.
Warp	Wavefront; (иногда Warp)	Sub-group	Набор операций (нитей), которые выполняются синхронно, выполняют одни и те же инструкции и следуют по одному и тому же пути потока управления: группа параллельных нитей, выполняемых аппаратным блоком (в Nvidia SM их 32) – это наименьшая вычислительная единица, на них разбивается блок нитей; **варп**.[3]
Thread block	Workgroup	Work-group	Группа варпов/подгрупп, одновременно выполняющихся на GPU. Могут синхронизироваться вместе и общаться через разделяемую память, в GPU Nvidia выполняется на одном SM; **блок нитей**.
Grid	Grid	ND-range	Сетка из блоков нитей, верхний уровень иерархии системы нитей в целом GPU; **сетка нитей**
Streaming Multi-processor (SM)	Compute Unit (CU)	X^e core	Аналог функционально упрощенного процессорного ядра ЦП. В Intel Data Center GPU Max X^e-ядро содержит несколько АЛУ SIMD-типа.
Tensor core	Matrixcore unit	Matrix-Engine (XMX)	Вычислительный блок для обработки умножения маленьких матриц (GEMM-операций, со смешанной точностью); **тензорные ядра**.
Shared memory	Shared memory	Local memory[1]	Высокоскоростная (подобная кэшу) память малой емкости, обеспечивающая связь между варпами в блоке нитей/рабочей группе; **разделяемая память**.
Global Memory (*общий термин Nvidia, AMD и Intel*)			Память DRAM, доступная в GPU; ее данные проходят через несколько уровней кэш-памяти
Device[2]			GPU (с памятью); **устройство**
Host[2]			Процессор и память (более обще – вся часть компьютера без GPU); **хост**
Kernel[2]			Часть программы, выполняемая на GPU (функция в C, подпрограмма в Fortran). Ядра устройств могут работать параллельно с ЦП; **программное ядро**

Термины Nvidia взяты из [16]; AMD – из [58]; Intel – из [69].

[1] неполное соответствие

[2] общий термин разработчиков GPU

[3] от *warp* (англ.) – нити основы на ткацком станке (*примечание редактора*)

Разные уровни групп нитей позволяют эффективно организовать высокомасштабируемое SIMT-распараллеливание. Поскольку может возникать ситуация, когда варп ожидает данных из памяти, активный расчет переключается тогда на другой варп. В классических GPU от Nvidia для этого SM содержит планировщик варпов (он формирует варп-группу нитей) и диспетчер, занимающийся активизацией выполнения варп. Аналогичные аппаратные блоки существуют и в GPU других производителей.

Еще одной очень важной общей особенностью современных GPU является работа с данными различной точности, включая операции смешанной точности. Это при использовании уменьшенной точности и соответственно числа бит для представления числа дает возможность достигать в разы более высокую пиковую производительность, уменьшать требования к емкости памяти GPU и к ее пропускной способности, очень часто лимитирующей производительность; при уменьшении требуемой емкости памяти уменьшенный объем обмена данными GPU с ЦП также может увеличить производительность. При работе с традиционными ЦП это не имеет смысла, и все расчеты с плавающей запятой в HPC проводятся традиционно с FP64. Однако для очень активно развивающихся задач ИИ работы с нейронными сетями используют умножение матриц, и найдена возможной работа с меньшей точностью и со смешанной точностью.

Типичным для использования в моделях глубокого обучения форматом данных является одинарная точность, FP32, но во многих работах была показана достаточность более низкой точности, например, FP16 [70]. Время расчета для задач глубокого обучения лимитируется умножениями матриц, на что и ориентированы тензорные ядра в GPU Nvidia или их аналоги в других GPU нового поколения. Впервые тензорное ядро в GPU появились в V100, и с самого начала оно рассматривалось как интегрированная в GPU специализированная для приложений микросхема (ASIC, application specific integrated circuits) – см., например, [71]).

Формула (1) отражает BLAS-функцию **GEMM** (здесь $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ – двумерные матрицы, размерность $\mathbf{A}$ равна $M \times K$, размерность $\mathbf{B}$ – $K \times N$, у матрицы $\mathbf{C}$ размерность $M \times N$)

$$C = \alpha\mathbf{A} \times \mathbf{B} + \beta\mathbf{C} \tag{1}$$

Уже в первых тензорных ядрах (в V100) формат FP32 применялся для $\mathbf{C}$, а FP16 – для $\mathbf{A}$ и $\mathbf{B}$ [72]. В A100 можно использовать BF16 для $\mathbf{A}$ и $\mathbf{B}$, и TF32 для $\mathbf{C}$ (хотя в A100 в тензорных ядрах стало можно работать и с форматом FP64) [73].

Подобные матричные операции со смешанной точностью выполняются в современных GPU на специальных матричных блоках (см. терминологии разных производителей в таблице 1) и проводятся для матриц очень маленьких размеров из фиксированного набора. Например, в тензорных ядрах A100 для всех матриц из формулы (1) с форматом FP64 $M \times N \times K = 8 \times 4 \times 8$ [17].

Многие форматы чисел с плавающей запятой уменьшенной точности начали использоваться именно на GPU (в первую очередь – для задач ИИ); базовые параметры форматов с пониженной (относительно FP64) точностью приведены в таблице 2 (в этой таблице приведены только форматы для чисел с плавающей запятой – но в ИИ бывает и работа с целыми числами уменьшенной до 8 бит длины, INT8).

Таблица 2. Форматы чисел с плавающей запятой уменьшенной точности в GPU

Формат чисел	Число бит			
	Знак числа	Экспонента	Мантисса	В регистре[1]
FP32	1	8	23	32
TF32	1	8	10	32
TF32+	1	8	15	32
FP16	1	5	10	16
BF16	1	8	7	16
FP8-E4M3	1	4	3	8
FP8-E5M2	1	5	2	8

[1] формат TF32+ поддерживается только в BR100 [35], а форматы FP8 – в H100 [78]

В этой таблице использованы данные таблицы 11 в [79] с добавлением строки формата TF32+ для BR100 [35].

Некоторые из этих форматов уменьшенной точности не поддерживаются стандартом IEEE-754 [74], но поддерживаются конкретными моделями производителя GPU (TF32, TF32+, BF16, форматы FP8). Здесь следует отметить, что для глубокого обучения считаются эффективными форматы TF32 и BF16 (см., например, [17, 75]). В TF32 для мантиссы используются те же 10 бит, как для FP16, но из-за более длинной экспоненты диапазон представляемых чисел больше, что важно для задач ИИ [76]. В [77] рассмотрена возможность применения для глубокого обучения форматов FP8.

Поскольку использование форматов с уменьшенной точностью на GPU может приводить к очень важному росту производительности, понемногу возможность работы с уменьшенной (относительно FP64) точностью стала не только применяться в ИИ, но изучаться и в других известных областях

HPC, в том числе: FP32 в CFD (были и попытки работы с FP16) [80], FP32 в классической молекулярной динамике (в [81] рост производительности на FP32 измерялся не на GPU), FP32 в квантовой молекулярной динамике (также были и попытки работы с FP16) [82,83], в квантовой химии [84,85]. Соответствующий рост производительности может быть обусловлен не только прямым увеличением собственно производительности ядер GPU за счет уменьшения точности, но и возможным кардинальным уменьшением требований к емкости памяти GPU.

Естественно, стали появляться и исследования о достижении приемлемой точности результата при работе с уменьшенной точностью в математических методах, например, при решении уравнения Пуассона [86]. В [87] предложены методы коррекции возможных ошибок относительно FP32 при вычислениях с FP16 и TF32 (при работе на тензорных ядрах A100). Понятно, что при работе с пониженной точностью требуются достаточно подробные систематические исследования, которые могут не успевать проводиться из-за сверхбыстрого развития современных GPU и появления новых форматов данных в них.

Даже в ИИ применение, например, TF32 с уменьшенной относительно FP32 мантиссой делает актуальными подробные исследования из-за возможных проблем со сходимостью глубокого обучения. Поэтому интересным может быть доступный для BR100 формат TF32+, имеющий большее число бит для мантиссы, чем в TF32. А, например, в молекулярной динамике современные комплексы программ, работающие на GPU, часто имеют опции, разрешающие расчеты с пониженной точностью (например, для двухточечных взаимодействий атом-атом) и в большом числе случаев дают при этом приемлемые результаты – однако иногда это приводит к ошибке. Публикации, формулирующие, в каких случаях возникают такие ошибки, автору неизвестны. Для квантовой химии ситуация может быть сложнее, например, в итерациях с самосогласованием полной энергии. Автор в настоящее время вообще с настороженностью относится к HPC-расчетам с пониженной точностью.

До сих пор тензорные ядра рассматриваются как ASIC, ориентированные на задачи машинного обучения (см., например, [88]), хотя в последнее время появляются работы, ориентированные на распространение области применения умножения матриц тензорными ядрами на HPC (см., например, [89]). В целом для HPC необходимо базироваться на производительности, достигаемой конкретными приложениями. В [90] было обнаружено, что из 77 отобранных там известных тестов производительности для HPC GEMM использовалась только в 10. И с точки

зрения энергоэффективности применение эмуляции форматов FP64 и FP32 пониженной точностью на тензорных ядрах V100 существенно уступало использованию там векторных ядер. Поэтому для более широкого использования тензорных ядер на HPC с точки зрения автора необходима аппаратная поддержка в них FP64, что у Nvidia началось на A100.

Вопросы работы с числами различной точности имеют более общее значение не только для GPU. Например, в [**91**] предложен новый блок совмещенных операций «умножить-и-сложить», ориентированный на задачи HPC и ИИ, для работы с форматами различной точности. Некоторая информация о достигаемой точности при работе на тензорных ядрах при умножении матриц с использованием операций смешанной точности очень кратко приведена далее в разделе 4.1.4, где рассматривается достигаемая производительность на A100.

2. Новые китайские GPU BR100

В обзоре рассмотрение начинается с Biron Technology BR100 по ряду причин. Этот ускоритель достаточно существенно отличается по построению от более традиционных графических процессоров Nvidia и AMD, даже в форме используемой терминологии. Публикации с оценками производительности BR100 в тестах и приложениях практически отсутствуют, а перспективы дальнейшего их производства стали сомнительными из-за введенных США санкций. Поэтому используемая для BR100 терминология в таблице 1 не приводилась. Тем не менее анализ BR100 представляется интересным в том числе как возможной эффективной альтернативы современным GPU Nvidia.

Появление этих GPU ярко проявилось по двум причинам – высокой скорости разработки (сделаны были «практически с нуля» всего за 3 года) и декларированным превосходством в производительности данного китайского GPU над Nvidia A100 (H100 тогда просто еще не существовало).

Следует сразу отметить очень четкую ориентацию BR100 на работу в области ИИ, что позволило разработчикам провести ясную градацию важности при конструировании микроархитектуры. Основным доступным источником информации о BR100 (используемым и в данном обзоре) является доклад на конференции Hot Chips 34 2022 года [**35**], а уточняющая информация доступна на сайте разработчика [**51**]. Еще небольшие уточнения были в интервью руководителя Biren Technology Чжан Вэня [**92**]. Определенные сопоставления характеристик BR100 с другими GPU далее проводятся только относительно A100, так как эта модель в обзоре отнесена к числу базовых.

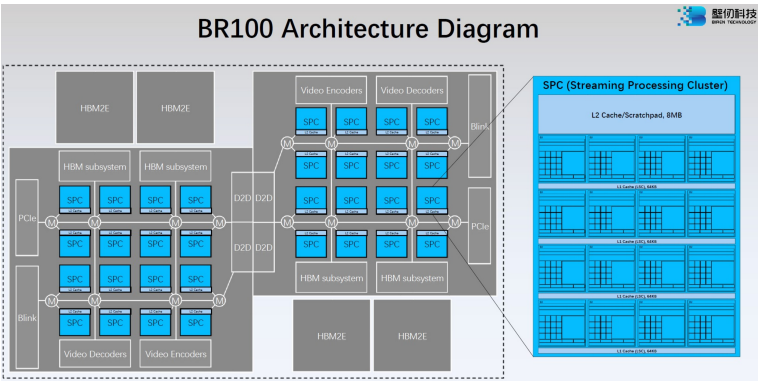


Рисунок 1. Микроархитектура BR100 (рисунок из [35])

Общая микроархитектура BR100 представлена на рисунке 1 [35].

Под семейством BR100 имеются в виду две разные модели – BR100 и более простая, более дешевая BR104, так что теперь BirenTech использует и наименование BR10X [51]. Общие характеристики, демонстрирующие успехи семейства BR100, приводятся обычно для модели BR100 (см. также таблицу 3). Здесь следует указать, что BR100 использует чиплеты и базируется на применении двух плиток (см. рисунок 1), а изготавливается по TSMC-технологии 7 нм (CoWoS 2.5D [35]) и содержит 77 миллиардов транзисторов с общей площадью 1074 мм^2. В BR104 плитка одна, и многие показатели также в два раза меньше, чем у BR100 (см. таблицу 3).

Основной представленной на рисунке 1 компонентой, определяющей достигаемую производительность BR100, является SPC (Streaming Processing Cluster), который можно отчасти считать неким аналогом Nvidia GPC (Graphics Processing Cluster) в A100. Каждая из двух плиток BR100 имеет по 16 SPC.

Каждый SPC содержит 16 исполнительных блоков EU (Execution Unit), которые и содержат собственно вычислительные компоненты GPU – 16 векторных ядер (V-ядра), и одно тензорное ядро TDA (Tensor Data Accelerator) [35]. При целевой тактовой частоте 1 ГГц (она заметно ниже ускоренной частоты ядер в A100, см. ниже Таблицу 13) и знании количества достигаемых в V-ядре FP32-результатов за такт (FP64 в BR100 не поддерживается, что связано с ориентацией на ИИ) это дает возможность рассчитать пиковую производительность с FP32. В таблице 3 приведены данные о пиковой производительности, достигаемой при работе с TDA (поскольку они особенно актуальны для задач ИИ, на который в первую очередь ориентируется BR100) [51]. Достигнутые (для актуальных для ИИ форматов данных) величины пиковой производительности лишь немного ниже исходных ожиданий [35] и в 1,5–2 раза выше, чем у A100.

На самом деле в иерархии от уровня SPC к EU в BR100 есть промежуточный уровень – блоки CU (Compute Unit), каждый из которых может содержать 4, 8 или 16 блоков EU (см. правую часть рисунка 1). CU можно считать аналогом SM (Streaming Multiprocessor) в A100.

CU из четырех EU имеет кэш L1 (LSC) емкостью 64 КБ. Следующим в иерархии памяти является кэш L2 емкостью 8 МБ на один SPC, что дает 256 МВ на весь BR100. Память HBM2E емкостью 64 ГБ (в современных моделях A100 емкость увеличена до 80 ГБ – см., например, [93]) имеет ширину интерфейса 4096 бит с пропускной способностью, которая также ниже, чем у A100 с 80 ГБ [93] (см. таблицу 3).

Таблица 3. Сопоставление показателей BR100 и A100

Показатели	BR100[1] (Walli 100P)	BR104[2] (Walli 104P)	A100-PCIe[5]	A100-SXM4[5]
Технология, нм	7 (TSMC)			
Форм-фактор	OAM	Полноразмерная двухслотовая плата PCIe	PCIe	SXM
Пиковая производительность:[4]				
FP32 (TFLOPS)	240	112		
TF32+ (TFLOPS)	480	224	156/312[3,6]	156/312[3,6]
BF16 (TFLOPS)	960	448	312/624[3]	312/624[3]
INT8 (TOPS)	1920	896	624/1248[3]	624/1248[3]
Тип и объем памяти	HBM2E 64 ГБ	HBM2E 32 ГБ	HBM2E 40 ГБ	HBM2E 80 ГБ
Ширина шины памяти (бит)	4096	2048	5120	5120
Пиковая пропускная способность (Тбайт/с)	1,64	0,819	1,9	2,0
Межсоединение с GPU, его пиковая пропускная способность (Гбайт/с)	BLink (8 портов ×8), 448	BLink (3 порта ×8), 192	NVLink3, 600	NVLink3, 600
Межсоединение с ЦП	PCIe-5.0, ×16 с поддержкой CXL	PCIe-5.0, ×16 с поддержкой CXL	PCIe-v4 ×16	NVLink3
TDP, Ватт	550	300	250	400

[1] см. [99];
[2] см. [100];
[3] через слэш приведены данные при использовании разреженности;
[4] приведены данные с использованием тензорных ядер ;
[5] данные из [93, 94];
[6] для A100 приведены данные для TF32.

Для достижения высокой производительности GPU важна пропускная способность памяти, и некоторое отставание здесь BR100 от A100 компенсируется огромной емкостью кэша L2 (в A100 емкость кэша L2 гораздо меньше – 40 МВ [94, 95]). Для поддержания более эффективной работы

кэша L2 в BR100 может использоваться Near Memory Computing [92]. Очевидно, это определенный аналог парадигмы Near Memory Processing, близкой к парадигме PIM, processing-in-memory, цель которых – пространственное объединение вычислительных блоков с памятью и сильное сокращение передач данных между ними [96], применный для задач ИИ [97, 98].

Что касается пропускной способности, она не менее важна для связи между двумя плитками BR100, и составляет 896 ГБ/с [35], что позволяет воспринимать BR100 как один общий GPU.

Для связи с ЦП применяется PCIe-5.0 (×16), с поддержкой CXL (Compute Express Link) – межсоединения с когерентным кэшем [101, 102], которое имеет явную тенденцию к стандартизации.

В одном сервере может быть установлено до 8 GPU BR100, и для коммуникаций между такими GPU используются каналы связи «точка-точка» (т. е. между каждой парой GPU) BLink, где используется SerDes (сериализатор/десериализатор) [92]. Один BLink имеет двунаправленную пропускную способность 64 ГБ/с [35], соответственно у 7 каналов BLink, связывающих один GPU со всеми другими, суммарная пропускная способность составляет 448 ГБ/с. Выбор полной поддержки всех соединений точка-точка дает BR100 преимущество из-за отсутствия возможной конкуренции при совместном использовании несколькими GPU пропускной способности межсоединения, в то время как обмен данными между GPU через ЦП дает свои минусы в пропускной способности и задержке [3]. Поэтому, как отмечено в [3], топология такого межсоединения весьма важна.

Для связи между GPU A100 с форм-фактором SXM4 [103] используется межсоединение Nvidia NVLink3 [94], где для соединения GPU-GPU используется 12 каналов с пропускной способностью по 50 ГБ/с – соответственно получается двунаправленная полоса пропускания 600 ГБ/с [94], что гораздо больше, чем у BR100. У связи BR100 с ЦП пропускная способность (896 ГБ/с) существенно выше, чем 600 ГБ/с у NVLink3. В варианте A100 с PCIe-4.0 пропускная способность связи между GPU такая же, как у BR100, 64 ГБ/с [93].

Понятно, что оценкой эффективности межсоединения GPU может быть только измеренная производительность теста или приложения, которая для BR100 в данный момент практически отсутствует. Наблюдающаяся ориентации приложений на минимизацию всех коммуникаций между ЦП и GPU (это – одно из основных правил оптимизации в CUDA на A100 [104]) позволяет предположить, что масштабирование производительности с ростом числа GPU в сервере с A100/SXM4 (в случае немаленьких потребностей в таких коммуникациях) будет выше, чем в сервере с BR100.

Плюсом BR100 является применение OAM Spec v1.1 [92] – быстро распространяющегося и фактически претендующего на стандартизацию форм-фактора OAM (OCP Accelerator Module) [105] (соответствующий модуль с BR100 именуется Walli100 [92]); при этом BR100 располагается на плате UBB [92], которая тоже может стать стандартом будущего [105]. Для сравнения, в A100 с высокоскоростным межсоединением GPU NVLink3 используется собственный форм-фактор Nvidia SXM, при этом максимальное число GPU в сервере (например, в знаменитом для ИИ Nvidia DGX) также равно 8 [106].

Конечно, BR100 обеспечивает целый ряд других возможностей, здесь не обсуждаемых – в том числе 4 блока специальных функций (Special Function Units – SFU, аналоги ранее известных в GPU Nvidia, используемые в том числе для расчета элементарных функций); возможности работы с NUMA и UMA; поддержку до 8 виртуальных GPU (Secure Virtual Instance, SVI – аналог Nvidia MIG, Multi-Instance GPU), что позволяет эффективно работать с GPU одновременно нескольким приложениям, не поддерживающим хорошее масштабирование до полного GPU [35, 99]. Классические шейдерные блоки для обработки изображений в BR100 вообще отсутствуют, что связано с однозначной ориентацией этого GPU на ИИ (видеокодек поддерживается [99], но это в обзоре не обсуждается).

Еще одним важнейшим современным параметром GPU является TDP – по данным таблицы 3, BR100 нужно больше энергопотребления, чем A100. Если посчитать энергоэффективность (производительность на Ватт), то, например для актуального для ИИ формата BF16 в BR100 она выше, чем у A100 с SXM4 или PCIe. Созданный сервер Haixuan OAM [92, 99] содержит 8 BR100 и имеет пиковую производительность (для BF16) около 8 PFLOPS при максимальной TDP 7 КВт [92] (см. рисунок 2) [35]; совместно с Inspur, известным производителем multi-GPU серверов для ИИ, планируются и кластерные решения [92].

В BR104 в два раза меньше не только производительность и емкость памяти (см. таблицу 3), но и число портов Blink равно 3 – соответственно в одном сервере можно установить меньше GPU. Про анонс сервера Wallen Technology Wallace 104 с BR104 было сообщено в ряде СМИ. Здесь важен еще форм-фактор – для BR104 используется полноразмерная двухслотовая плата [100]. Это можно сравнить с A100-PCIe – на их основе Nvidia делает для серверов HGX-модули с PCIe- форм-фактором, содержащие в том числе и по два GPU, связанные через мост NVLink Bridge [93].

Что касается архитектуры BR100 (Bi Liren) в общем смысле, а не микроархитектуры, то здесь следует отметить большой набор поддерживаемых форматов данных (в таблице 3 производительность приведена только для некоторых из них): INT8, INT16, INT32, FP16, BF16, FP32, TF32+. Про них, в первую очередь про оригинальную разработку Biren Technology

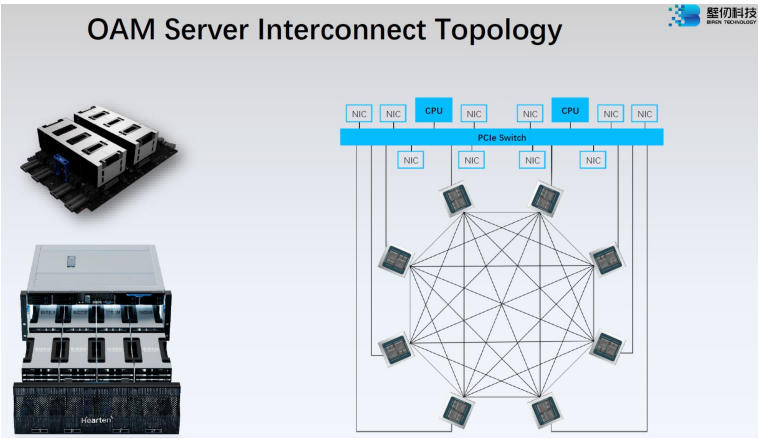


Рисунок 2. Топология межсоединения в сервере с BR100 (рисунок из [35])

TF32+, информация доступна на ряде сайтов (см., например, [107]). TF32+ представляется как попытка усовершенствования известного формата TF32 для тензорных ядер от Nvidia [94] (см. таблицу 3), с помощью реализации которого разработчики BR100 хотели увеличить точность и производительность [51]).

Для работы с BR100 был разработан набор средств программного обеспечения BIRENSUPA (BIREN Scalable Unified Parallel Architecture) [92] – драйверы, компилятор BRCC с поддержкой расширенного C++, библиотеки программ и другие средства, ориентированные в первую очередь на глубокое обучение [35, 108]. BIRENSUPA имеет похожие на NVidia CUDA парадигму программирования и стиль языка, и также использует уникальные для BR100 аппаратные возможности [92]. Но публикаций, демонстрирующих использование этих программных средств и реально достигаемую производительность, на момент написания обзора не было.

Что касается производительности для ИИ, то для BR104 имеются данные для двух тестов из известного набора тестов производительности стадии выводов машинного обучения *MLPerf inference datacenter* версии 2.1 [109, 110] для ЦОД. Результаты тестов MLPerf inference datacenter показывают, как быстро обученная нейронная сеть может выполнять задачи вывода на новых исходных данных.

Первый тест для классификации изображений из состава MLPerf Inference datacenter базируется на ResNet (residual neural network) с использованием знаменитой технологии искусственной нейронной сети; ResNet многократно расширялась и модернизировалась, и использовалась, например, и для обработки изображений для диагностики COVID-19 (см.,

например, [111]). В другом тесте для работы с натуральным языком, BERT (Bidirectional Encoder Representations from Transformers) используется модель машинного обучения на основе преобразователя для предварительного обучения при обработке естественного языка с применением двунаправленного кодировщика [112]. Этот метод крайне широко используем и наиболее знаменит, вероятно, благодаря его применению Google.

Результаты этих знаменитых тестов, представленные в [109] для серверов, имеющих 4 или 8 GPU BR104, а также для серверов с четырьмя или 8 GPU A100, приведены в таблице 4, в которой отобраны наиболее высокие из достигнутых показатели производительности.

Таблица 4. Данные тестов производительности MLperf 2.1 inference datacenter (декабрь 2022) на серверах с GPU

Тип GPU	Число GPU в сервере	Производитель и модель сервера	Классификация изображений (A=99%)		Обработка естественного языка (A=99,9%)	
			server (запросов/с)	offline (образцов/с)	server (запросов/с)	offline (образцов/с)
BR104, PCIe	4	Inspur NF5468M66[1]	150027	212391	8993	11106
	8	Inspur NF5468M66-P[2]	200052	424660	13952	22134
A100, SXM, 80GB	4	Lenovo SR670v2[3]	150027	174180	Нет данных	
	4	Dell PowerEdge XE8545[4]	128029	131364	5297	5476
A100, PCIe, 80GB	8	ASUS ESC8000A-E11[5]	270066	283838	11496	13129
A100, SXM, 80GB	8	Inspur N5688M6[6]	313069	347202	Нет данных	
	8	Inspur N5488A5[7]	290066	346954	13594	14977
H100, SXM, 80GB	1	Nvidia Preview[8]	58995	81292	6195	7921

[1] с Intel Xeon Gold 6354 и suInfer;
[2] с Intel Ice Lake-SP 8368 и suInfer;
[3] с Xeon Platinum 8360Y при 2,40 ГГц и с CUDA 11.6;
[4] с EPYC 7763 и с MaxQ, TensorRT 8.4.2 и CUDA 11.6;
[5] с 64-ядерным EPYC 7763, TensorRT 8.4.0 и CUDA 11.6;
[6] с Xeon Platinum 8358, TensorRT 8.4.2 и CUDA 11.7;
[7] с EPYC 7713, TensorRT 8.4.2 и CUDA 11.7;
[8] с 8-ядерным EPYC 7252, с TensorRT 8.5.0 и CUDA 11.8.

Указанные данные для GPU BR104 и A100 относятся к классу доступных (т. е. соответствующие серверы можно приобрести). В этих тестах требуемая точность вывода (A) составляет 99% или 99.9% относительно FP32. Но надо иметь в виду, что достигаемая производительность может существенно зависеть от используемых систем разработки программного

обеспечения. Для A100, кроме базовых средств CUDA, для достижения высокой производительности применялись средства специального программного обеспечения Nvidia TensorRT [113]; для BR104 использовались средства suInfer.

Данные о производительности MLperf inference datacenter 2.1 с использованием 4 и 8 GPU BR104 в серверах Inspur показали преимущество BR104 по производительности в тесте обработки естественного языка (NLP) с моделью BERT в полтора-два раза при использовании в сервере одинакового количества BR104 или A100. В «автономном» сценарии теста классификации изображений с моделью ResNet на серверах с четырьмя и 8 GPU серверы с BR104 также быстрее серверов с A100 (см. таблицу 4), а в «серверном» сценарии этого теста в серверах с четырьмя GPU A100 не опережают серверы с BR104, но существенно быстрее их при использовании 8 GPU. Это может быть связано и с преимуществом NVLink3 относительно BLink при таком количестве GPU.

В этих тестах может требоваться меньше вычислительных ресурсов, чем в MLPerf Training, часто выполняемом на GPU A100: MLPerf Training измеряет время, необходимое для обучения моделей машинного обучения до целевого уровня точности, там основные задачи – собственно обучение.

Приведенные выше данные про BR100 явно свидетельствуют о создании его для конкуренции с GPU Nvidia, чему должны способствовать не только более высокие показатели пиковой производительности BR100 и высокая достигаемая производительность на тестах для ИИ. Четкая ориентация на сверхбыстро развиваемые и коммерчески актуальные задачи ИИ (что дало возможность более узкой и экономичной ориентации аппаратуры BR100), и поддержка претендующих на стандартизацию аппаратных средств должны способствовать понижению стоимости BR100, что сочетается с исходно скорее типичной для китайских производителей более низкой стоимостью относительно продукции западных стран.

Однако ситуация с BR100 резко поменялась из-за введенных в 2022 году США санкций, вследствие чего TSMC прекратила изготовление и поставки микросхем BR100. Это широко обсуждалось в разных СМИ, здесь следует только отметить, что этот запрет направлен против возможной конкуренции с Nvidia и не способствует убыстрению развития мирового рынка GPU.

3. Intel Data Center GPU Max (Ponte Vecchio)

Выбор в качестве следующего объекта анализа Intel Ponte Vecchio (далее используется сокращение PVC) как нового поколения GPU связан с тем, что они в виде нескольких разных моделей Data Center GPU Max на момент написания обзора только появились на рынке, и научные публикации об их производительности почти отсутствуют. Информация

Intel о доступных моделях PVC менялась во время написания обзора, а архитектура PVC (и используемые программные средства SDK) достаточно сильно отличаются от более «традиционных» архитектур GPU от Nvidia и AMD.

Появление PVC ожидалось не один год, в сентябре 2022 года Intel объявила о начале поставок PVC на суперкомпьютер Aurora в Аргоннской национальной лаборатории США. Но в июньском списке Top500 2023 года этот суперкомпьютер отсутствовал.

3.1. Аппаратные средства PVC

Intel разработала архитектуру X^e («eXascale for everyone») для ее применения в самых разных классах графических процессоров (не только GPGPU), для работы и с персональными компьютерами, и с серверами. В X^e общая архитектура команд (ISA), а микроархитектуры Intel использует 4 разных – для каждого класса графических процессоров микроархитектура своя [**69**]; в PVC используется микроархитектура X^e HPC-GPU [**114**, **115**]. Для ЦОД Intel предлагает продукты Data Center GPU, вкючающие две серии – Data Center GPU Max [**116**], – это официальное название, заменившее использование кодового слова Ponte Vecchio (в обзоре используется сокращение PVC), и серию Data Center GPU Flex [**117**] с микроархитектурой X^e-HPG.

В обзоре рассматривается только семейство ориентированных на HPC GPU с микроархитектурой X^e HPC-GPU – Data Center GPU Max (PVC ориентирован на работу и в экзамасштабных суперкомпьютерах; под PVC далее подразумевается старшая модель этой серии, Max 1550 – для формирования суперкомпьютера Aurora Intel фактически поставлялся такой GPU, и данные практически о нем были представлены в ряде публикаций, цитируемых здесь в обзоре). Intel указывает на 3 различных модели в этом семействе, данные о рекомендуемой стоимости которых на сайте ark.intel.com на момент написания обзора ожидаемо не были представлены – в соответствии с соответствующими отсутствиями аналогичных рекомендаций цен у других производителей GPU нового поколения. В таблице 5 приведены базовые параметры различных моделей PVC. В этой таблице приведены только показатели GPU, актуальные в первую очередь для классических задач HPC. А, например, число блоков в PVC для обработки трассировки лучей, которые могут использоваться и для задач ИИ (см., например, [**118**]), здесь не приведены (в PVC их столько же, сколько X^e-ядер – 128) – аппаратные возможности PVC для трассировки лучей в обзоре по сути не рассматриваются.

Для изготовления PVC с самого начала планировалась применять трехмерную укладку с базированием на плитках (tiles) [**114**, **120**]. К этому Intel могло подтолкнуть и определенное отставание в собственной полупроводниковой технологии (иногда формулирующееся как слухи [**121**]). По данным доклада, проведенного в рамках знаменитого экзамасштабного вычислительного проекта ECP [**67**], PVC планировался к поставкам в 2021 году.

Таблица 5. Базовые показатели моделей Data center GPU Max серии (PVC)

Модель GPU	Число ядер X^e	Число XMX	Число XVE	Базовая частота, ГГц	Максимальная частота, ГГц	Емкость памяти, ГБ	Пропускная способность памяти, ГБ/с	TDP, Ватт
1100	56	448	448	1,0	1,55	48	1228,8	300
1350	112	896	896	0,75	1,55	96	2457,6	450
1450	128	1024	1024	нет данных		128	нет данных	600
1550	128	1024	1024	0,9	1,6	128	3276,8	600

XMX – матричные устройства, XVE – векторные устройства. Данные таблицы взяты из разных по времени версий [116] и [119]. Модели 1350 и 1450 помечены красным цветом, так как они не представлены на сайте ark.intel.com на момент написания обзора.

В PVC используется трехмерная технология Co-EMIB (Co-Embedded Multi-Die Interconnect Bridge) с применением чиплетов, способствующая высокой производительности, а с PVC можно говорить о работе с архитектурой PIM (processing-in-memory), направленной на решение проблемы обменов большими объемами данных с памятью [122]. Построение из нескольких кристаллов (die) трехмерных процессоров в [123, 124] указывается как прогрессивный способ для обеспечения продолжения действия закона Мура, а трехмерную память стали делать уже достаточно давно, что имело место не только для HBM (см., например, [125]).

Здесь нельзя не указать на альтернативный вариант интеграции в целое разных кристаллов (плиток у Intel в PVC) с применением чиплетов, используемый AMD, в том числе при построении GPU MI200. Современный обзор чиплетов см. в [126], а современные GPU AMD рассматриваются ниже. Необходимо отметить также разрабатываемый консорциумом ряда фирм, в том числе Intel, AMD, ARM, Google и TSMC, стандарт для межсоединения между кристаллами, включающий не только физический уровень – сейчас имеются спецификации UCIe [127].

Модель PVC содержит 100 миллиардов транзисторов, расположенных в 47 функциональных плитках (термоплитки здесь не учитываются), объединенных в 5 узлов [116, 128–130]. Из этого большого числа плиток две плитки являются базовыми, 16 – вычислительными плитками (compute tile – см. рисунок 3 [131]), 8 – плитками памяти HBM2E. Внутри PVC устроен как 2 аппаратных стека [129] – на каждой из двух базовых плиток укладывается по 8 вычислительных плиток и по 4 плитки памяти (логические взаимосвязи этих компонентов PVC представлены на рисунке 3). Базовые плитки содержат в том числе интерфейсы PCIe и каналы к HBM2E.

Эти плитки, как и плитки межсоединения между отдельными GPU PVC, X^e Link [129, 130] будут вкратце рассмотрены ниже. В PVC планировались еще плитки кэша Rambo (Random Access Memory, Bandwidth Optimized) [129], но для X^e-архитектуры в [114] они указаны как необязательные, и в техническом описании микроархитектуры [130] они отсутствуют.

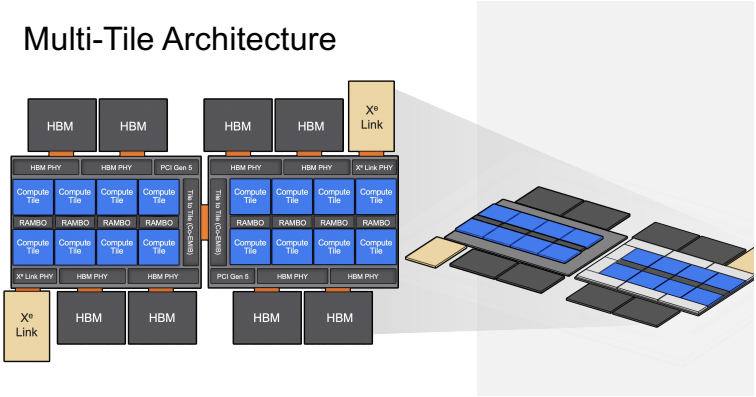


Рисунок 3. Архитектура PVC на базе плиток (рисунок из [129])

Реализация продвинутой многокристальной 3D-технологии подробнее рассмотрена в [128]. Рассмотрение технологий не относится к тематике данного обзора, следует отметить только использование в PVC 24-слойной подложки, обеспечивающей работу с технологией трехмерного стекирования Foveros [132] и с 2.5D-технологией EMIB (Embedded Multi-die Interconnect Bridge) [133], которые при совместном использовании именуются Co-EMIB [122]. EMIB используется для обеспечения работы локальных внутренних межсоединений; Foveros с технологией Intel 7 (бывшая Enhanced 10nm SuperFin) используется в базовых плитках. Плитки памяти сделаны по технологии Intel 7, вычислительные плитки – по TSMC N5, а плитки памяти HBM2E – по TSMC N7. Изготавливается PVC для форм-фактора OAM 1.1 [134]), дающего высокую плотность упаковки оборудования и, как было отмечено выше в разделе 2, претендующего на применение в качестве стандарта. Плитки далее рассматриваются не в технологическом плане, а просто как некие блоки микроархитектуры – соответственно некоторые типы плиток здесь вообще не упоминаются.

Здесь необходимо указать, что Intel использует и другие термины (не только плитки) в отношении иерархий микроархитектур класса X^e [69, 130]. Подломтики (subslice) используются не для PVC, это аналог X^e-ядер в PVC. В PVC (Data Center GPU Max) ниже уровня всего GPU имеется 2 уровня иерархии – ломтик (X^e HPC Slice) и стек (X^e HPC stack). Ломтик имеет 16 X^e-ядер и 16 устройств трассировки лучей. Стек содержит 4 ломтика (соответственно с 64 X^e-ядрами и 64 устройствами для ускорения трассировки лучей) [69, 115]. Кроме того, стек имеет кэш L2, контроллер памяти, межсоединение PCIe-v5 и 8 каналов X^e Link (см. далее). PVC масштабируется до двух стеков – до 128 X^e-ядер [116, 130]. X^e – ядра являются аналогами SM в GPU Nvidia, и сопоставление их количества часто используется при сравнении различных GPU (а также и с количеством ядер в ЦП).

Общее описание микроархитектуры PVC имеется в [130]. Каждая вычислительная плитка содержит 8 X^e-ядер, которых на весь PVC соответственно 128 штук. Каждое X^e-ядро имеет по 8 векторных устройств XVE (eXtended Vector Engine) с длиной векторов 512 бит и по 8 матричных устройств XMX (X^e Matrix eXtension), работающих с операндами длиной 4096 бит [69, 129, 130], и имеет интерфейс X^e Link, базирующийся на когерентном межсоединении CXL [135] (также разработанном ранее Intel).

Соответственно в PVC всего 1024 векторных устройства XVE и 1024 матричных устройства XMX, которые являются аналогом тензорных ядер Nvidia (это было указано в таблице 1 во введении), которые имеются не только в GPU V100 и A100, но и в других графических процессорах Nvidia [136].

Устройства XVE работают с 512-битными операндами и используют операции умножение-и-сложение. Это дает на X^e-ядро 256 FLOPS за такт для FP64 (и FP32 тоже) [129].

В таблице 6 приведено количество выполняемых операций за такт

Таблица 6. Количество операций за такт, выполняемых в одном X^e-ядре и в иерархии выполняющих расчеты блоков для разных форматов данных [69]

Исполняющие блоки	Формат данных	Число операций за такт
8 × XVE	FP64 FP32 FP16	256 256 512
8 × XMX	TF32 FP16 BF16 INT8	2048 4096 4096 8192
Ломтик (slice) – исполняющих блоков и операций за такт в 16 раз больше		
Стек – исполняющих блоков и операций за такт еще в 4 раза больше		
Двухстековый PVC (модель 1550) – исполняющих блоков и операций за такт еще в 2 раза больше		

XMX не поддерживает использование формата FP64.

с разными форматами данных, доступными для XVE и XMX [129, 130]. Это позволяет рассчитывать пиковые производительности P с использованием тактовой частоты ν. Так, для форматов FP64 или FP32 при работе с XVE $P = 128 \times 256 \times \nu$, что дает 52,4 TFLOPS (в таблице 7 приведены округленные до целого числа из обзора Intel [130]). Учитывая приведенные в таблице 5 возможные величины ν, становится ясно, что это число рассчитано в предположении работы на максимальной, а не на базовой частоте.

Из данных таблицы 7 можно прийти к выводу, что при использовании векторных (без применения матричных блоков) операций пиковая производительность для FP64, традиционная для HPC, монотонно растет по мере даты начала выпуска новых моделей GPU.

Таблица 7. Сопоставление векторных и матричных (XMX) пиковых производительностей Intel PVC и GPU других фирм для разных форматов данных

Производительность для разных форматов	PVC	A100	H100-SXM	H100-PCIe	MI250X
FP64 (TFLOPS)[1]	52	9,7	33,5	25,6	47,9
FP32 (TFLOPS)[1]	52	19,5	66,9	51,2	47,9
XMX TF32 (TFLOPS)	419	156	494,7	378	нет
XMX BF16 (TFLOPS)	832	312	989,4	756	383
XMX FP16 (TFLOPS)	832	312	989,4	756	383
XMX INT8 (TOPS)	1664	624	1978,9	1513	383

[1] Данные без использования матричных операций (XMX в PVC не поддерживает FP64). С применением тензорных ядер пиковая производительность H100, A100 и MI250X в два раза выше. Данные Nvidia H100 взяты из [78], для A100 – из [73]; данные по PVC – из [130, 138].

В задачах ИИ нормален формат FP32, и можно использовать FP16/BF16 в том числе на XMX (см. текст выше в обсуждении формулы (1)). Операция извлечения квадратного корня в формате FP32, согласно [69], дает в XVE четыре результата за такт.

Для графических процессоров Intel наблюдаются не очень низкие частоты, например у Arc Alchemist бывают и больше 2 ГГц [137]. Ранее в [129] указывалось на ожидание пиковой производительности FP32 не менее 45 TFLOPS – это, скорее скорее всего, означает, что по сравнению с прошлогодним прототипом в PVC тактовую частоту Intel реально удалось поднять.

Но надо иметь в виду, что потребляемая при этом мощность растет пропорционально частоте, а увеличение частоты для поддержания работоспособности микросхемы часто требует еще и увеличения напряжения, квадрату которого пропорциональна эта мощность. TDP у рассматриваемой модели PVC составляет 600 Вт, и предполагает жидкостное охлаждение [129] (см. также сравнения моделей Data Center GPU Max в таблице 5). В [129] указывается и на другую модель PVC с возможным воздушным охлаждением и TDP 450 Вт – это соответствует модели Data Center GPU Max 1350.

Однако в середине 2023 года Intel объявила о прекращении производства модели Max 1350 в связи с созданием модифицированного варианта Max 1550 с воздушным охлаждением, и о планах производства новой модели Max 1450 [139].

Таблица 8 дает сопоставление PVC (по умолчанию имеется в виду единственная поставляемая в момент написания обзора модель 1550) с другими рассматриваемыми в обзоре GPU по другим важнейшим, в том числе для производительности, показателям.

Таблица 8. Сопоставление аппаратных показателей PVC с другими современными GPU

Показатели	PVC	H100	A100-SXM4-40GB	MI250X
Технология	Intel 7 (7 нм) + TSMC N5 + TSMC N7	TSMC 4N (5 нм)	TSMC N7 (7 нм)	TSMC (6 нм)
Емкость памяти, Гбайт	128	80	40	128
Тип памяти	HBM2E	HBM3	HBM2	HBM2E
Ширина шины памяти, бит	8192	5120	5120	8192
Пропускная способность, ТБ/с	3,3	3,0	1,6	3,2
Форм-фактор	OAM	SXM	SXM	OAM
TDP, Вт	600	700	400	560

Вследствие высокой вычислительной производительности современных GPU реально достигаемая производительность при работе с ними часто лимитируется пропускной способностью памяти, что бывало и раньше (см., например, [140]), и теперь часто бывает (см., например, [141, 142]).

Обсуждение иерархии памяти PVC следует начать с файла регистров, который имеет емкость 64 МБ (по 512 КБ на каждое X^e-ядро) и обеспечивает пропускную способность 419 ТБ/с [120]. Кэш L1 традиционно разделяется на кэш команд и кэш данных, который имеет емкость 512 КБ на ядро [129, 143]. В PVC есть еще SLM (Shared Local Memory), с суммарной емкостью 8 МБ на весь PVC [69] – см. рисунок 4.

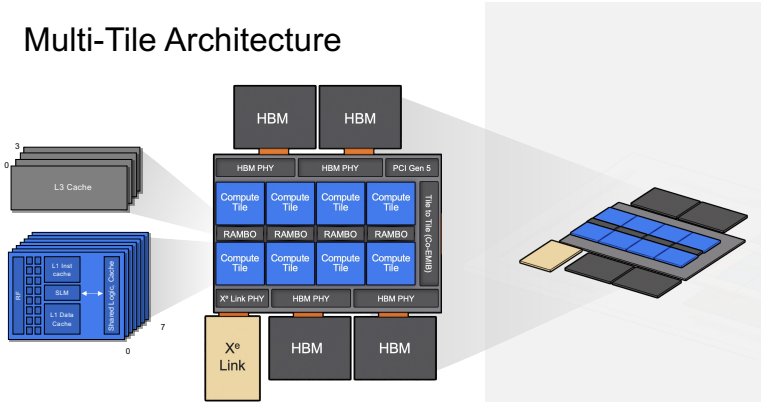


Рисунок 4. Иерархия памяти в PVC (рисунок из [129])

Каждая вычислительная плитка имеет 4 Мбайт L1 [129], всего 64 МБ на PVC, с пропускной способностью 105 ТБ/с [120]. Кэш L2 емкостью 408 МБ (по 204 МВ на каждый стек) [143] на базовой плитке [129] имеет пропускную способность 13 ТБ/с [120]. Данные для других моделей Data Center GPU Max приведены в таблице 9.

Таблица 9. Иерархия памяти и межсоединения моделей Data Center GPU Max серии

Модель GPU	Кэш L1, МБ на GPU/ядро X^e	Кэш L2, МБ на GPU/стек X^e HPC	Емкость HBM2E на стек X^e HPC, ГБ	Пропускная способность ТБ/с	Число портов X^e Link
1110	28/0,5	108	48[1]	1,2	6
1350	48/0,5	216/108	48	2,5	16
1450	64/0,5	408/204	64	Нет данных	
1550	64/0,5	408/204	64	3,3	16

[1] в модели 1110 всего 1 стек. Данные таблицы взяты из [69, 116] и [119].Модели 1350 и 1450 помечены красным цветом, так как они не представлены на сайте ark.intel.com на момент написания обзора.

Кроме того, в [129] указано на наличие в базовой плитке традиционного для обычных ЦП буфера быстрой переадресации TLB (который тоже можно считать некоторым видом кэша), и еще SRAM-кэша RAMBO – см. рисунок 4. Этот кэш состоит из 4 банков емкостью по 3,75 МБ на базовой плитке, содержащей также коммутатор для кэша [129]. Но в техническом обзоре Intel Data Center GPU Max[130] про наличие кэша RAMBO не упоминается.

Собственно память HBM2E в PVC объединяет до 128 ГБ (8 плиток) и работает по каналу шириной 8192 бит [120]. Как указано в [69], пропускная способность GTI (Graphics Technology Interface, соединяющего GPU с остальной частью вычислительной системы) у одного стека PVC составляет 1024 байт/такт для чтения или записи, что для двухстекового PVC с максимальной частотой 1,6 ГГц (для модели 1550) дает пропускную способность около 3,3 ТБ/с (см. таблицу 5). Показатели этой памяти для других моделей Data Center GPU Max также приведены в этой таблице.

Хотя каждый из двух связанных через интерфейс EMIB (со скоростью до 230 ГБ/с в обоих направлениях) стеков имеет «свой» кэш L2 и «свою» память HBM2E емкостью 64 ГБ [130], 128 ГБ HBM2E являются общей памятью всего PVC (который может быть виден программистам как один GPU), и каждый стек имеет прямой доступ к памяти HBM2E другого стека.

Прежде чем рассмотреть масштабирование вычислительных ресурсов с помощью имеющихся в Data Center GPU Max аппаратных средств X^e Link, обеспечивающих когерентное межсоединение между GPU в сервере [130], необходимо указать на отличия разных моделей GPU этой серии (см. таблицы 5, 9). Кроме старшей модели Max 1550, имеются еще данные о моделях Max 1450, 1350 и 1100 (как отмечено выше, модель 1350 Intel производить более не планирует).

Особо интересна будущая модель Max 1450, имеющая то же, что и Max 1550 число X^e-ядер. Max 1110 состоит не из двух, а из одного стека. Числу X^e – ядер в разных моделях пропорциональна общая емкость

HBM2E и кэшей L1 и L2: в модели Max 1100 соответствующие емкости равны 28 и 108 МБ. Естественно, с уменьшением числа X^e-ядер в моделях уменьшается и TDP (см. данные в таблице 5).

Здесь можно провести аналогию с тремя разными моделями GPU MI200 от AMD: старшая модель MI250X имеет два кристалла GCD (Graphics Compute Die), как и MI250 (у которой просто меньшее число ядер), а младшая модель MI210 имеет только один GCD (см. ниже в разделе про GPU от AMD).

Младшая модель Max 1100 отличается еще уменьшенным числом физических портов X^e Link (см. таблицу 9), но в ней используется и отличный от OAM формфактор – PCIe AIC [116] (add-in card). В PVC, который можно отнести к разряду систем на чипе (SoC) [114] имеется интерфейс PCIe 5.0 × 16 [130] с пропускной способностью около 63 ГБ/с [69]; во всех моделях Data Center GPU Max PCIe 5.0 используется для связи GPU с хостом [116].

Плитка X^e-Link поддерживает в PVC 8 каналов X^e Link и коммутатор, с обеспечением прямой связи каждого канала с каждым (см. рисунок 5) [143]. В двух стеках имеется соответственно 16 каналов X^e Link [69].

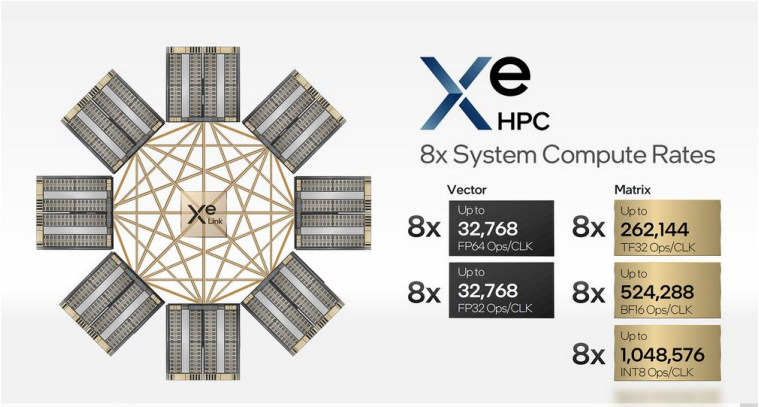


Рисунок 5. Межсоединение между PVC (рисунок из [146])

В [130] указано на пропускную способность каждого канала X^e Link 26,5 ГБ/с в каждом направлении. X^e Link предназначен для обеспечения когерентной связи между стеками X^e HPC. Это соответственно включает как внутреннюю связь внутри PVC (OAM), так и между несколькими PVC (OAM) [130]. Это используемое для связи между несколькими GPU в сервере межсоединение позволяет эффективно работать в режиме единой общей памяти SYCL (Unified Shared Memory, USM) [69].

Важным представляется то, как логически выглядят для пользователя модели Max 1550 и Max 1450. Если учесть указанную выше пропускную

способность X^e Link, то пропускная способность связи между двумя стеками гораздо ниже пропускной способности памяти на одном стеке. В похожих в этом плане AMD MI250X/MI250 поэтому каждый GCD там логически выглядит как один GPU. В [144] указано, что пользователи рассматривают две плитки PVC как два процессора. Однако в руководстве Intel по оптимизации oneAPI на GPU [145] указано, что переход от применения одного стека к двум требует просто изменения переменной окружения, т. е. Max 1550 дает возможность выбора между работой с ним в форме двух разных или одного общего GPU.

PVC официально стал поставляться Intel уже во время написания обзора, и многие детали и на аппаратном уровне были пока недоступны, например, величины задержек, что важно в том числе для связей через X^e Link, в том числе между стеками PVC.

В [130] указывается на 2 вида поддерживаемых Intel решений для масштабирования вычислительных систем за счет роста числа стеков X^e HPC: масштабирование внутри (scale-up) – внутри multi-GPU серверов, и масштабирование вовне (scale-out), которое относится к кластеру, содержащему узлы с GPU. Выше говорилось фактически о масштабировании scale-up.

В scale-out решении можно масштабироваться до кластера, содержащего максимум до 64 OAM, взаимосвязанных через X^e Link Glueless. Дальнейшее масштабирование с применением еще и Infiniband позволяет построить систему, имеющую до 512 OAM, связанных межсоединением [130]. «Бесклеевые» межсоединения (там нет промежуточных микросхем) известны весьма давно, они дают очень низкие задержки [147].

Сначала предполагается использовать конфигурацию Tuscany, в которой общая плата с четырьмя OAM может подсоединяться к существующему серверу через PCIe 5.0. Возможны варианты с Data Center GPU Max с TDP 450 или 600 Вт, с воздушным или жидкостным охлаждением [130]. С учетом данных таблицы 5 очевидно, ранее имелись в виду модели GPU Max 1350 и 1550 соответственно.

Использовать PVC исходно предполагалось в двухсокетных серверах с новыми ЦП Xeon Sapphire Rapids (теперь именуемыми Xeon Max: именно они применялись в данных Intel о производительности серверов [116]). Предлагаемые Intel в настоящее время Xeon Max обеспечивают работу с DDR5 и HBM2E (см., например, [148]). Использование HBM в этих ЦП может сильно увеличить производительность, что было показано в [149].

В суперкомпьютере Aurora вычислительные узлы являются двухпроцессорными серверами (с ЦП Xeon Max 9480), содержащими еще по 6 PVC [47]. Кроме создаваемого суперкомпьютера Aurora, PVC планируется применять и на европейском экзамасштабном суперкомпьютере с процессорами SiPearl Rhea [150]. Безусловно, это усиливает потенциальные

позиции PVC на рынке, но важнее будет учитывать реальности достигаемой на приложениях производительности, переноса на PVC программного обеспечения и стоимостные показатели. Data Center GPU Max, естественно, предполагается применять не только на суперкомпьютерах, но и на уровне маленьких кластеров и отдельных серверов с GPU – о таких серверах уже объявили знаменитые производители, в том числе Dell, Lenovo и Supermicro.

3.2. Программные средства и первые стартовые данные о производительности PVC

Что касается программного обеспечения, то имеющаяся публикация [120] с учетом и работ для создаваемого суперкомпьютера Aurora Аргоннской национальной лаборатории США [67,151,152] свидетельствует об ускоренном движении Intel в сторону линии стандарта SYCL от Khronos Group [153], расширенным Intel в виде DPC++ (Data Parallel C++) [154]. Актуальность этого направления очевидна, поскольку DPC++ ориентирован на работу с разными акселераторами, что дает переносимость в том числе и между разными GPU. Компилятор DPC++ от Intel (он является коммерческим [154]) входит в общие программные средства Intel oneAPI [155], которые могут работать и на процессорах, и на ПЛИС-ускорителях Intel, что дает разработчикам возможность создавать в определенной степени единый код программ. Intel активно разрабатывает новые версии oneAPI – так, сейчас уже доступна oneAPI 2023 [155].

Пакет oneAPI включает широкий набор средств – компиляторы, отладчики, профилировщики, библиотеки и специализированные средства для работы с ИИ. Компилятор oneAPI DPC++/ C++ дает возможность работы и с GPU от Nvidia и AMD [156]. Для задач оптимизации при работе с oneAPI Intel подготовила специальное руководство [69]. Большим преимуществом самого DPC++ следует считать наличие *компилятора с открытым исходным кодом* [157].

В [158] для V100 на тесте STREAM (triad) достигнутая пропускная способность в варианте с DPC++ была близка к полученной с использованием OpenCL и CUDA, а применение SGEMM из библиотеки oneMKL дало 66% от пиковой производительности. Очевидно, что эти результаты могут быть улучшены при работе с новыми версиями oneAPI.

Поскольку oneAPI является открытой основанной на стандартах унифицированной моделью программирования [159], ориентированной на работу и с процессорами, и с акселераторами разного типа и разных фирм [156], с учетом прекращения монополии GPU от Nvidia и перехода на применение GPU разных фирм это однозначно перспективно. Но ясно, что в направлении унификации могут развиваться предложения разных фирм (например, программные средства HIP от AMD уже ориентированы не только на GPU AMD), и в этой активно развиваемой области сложно предсказать, что станет наиболее востребованным.

Intel, естественно, позаботилась и об актуальной задаче возможного переноса кода с GPU Nvidia на PVC, и в oneAPI предоставляет и средства DPC++ Compatibility Tool (часто используемое сокращение – DPCT), которые автоматически переводят код CUDA на DPC++. По данным [160], такой автоматический перенос, как правило, делает это для 90–95% кода CUDA, а полное завершение переноса всего кода требует ручной работы. Насколько эффективно все это будет работать на GPU с точки зрения производительности, данных пока, естественно, практически нет. В [154] проведено предварительное исследование, но оно не относится к рассматриваемым в обзоре GPU, и неясно, насколько это актуально для задач HPC. В [161] из опыта переноса SGEMM из библиотеки MAGMA сделан вывод, что DPCT можно успешно использовать для начального переноса кода CUDA на DPC++, однако здесь использованы результаты для графических процессоров, не относящихся к числу рассматриваемых в обзоре.

В качестве примеров работ по переносу использующих GPU приложений на DPC++ следует указать перенос на DPC++ нескольких известных приложений молекулярной динамики – NAMD [162], GROMACS (перенесенный на SYCL, с использованием компилятора DPC++ [163,164]); LAMMPS тоже может работать на PVC – но благодаря применению средств Kokkos. Данные об аналогичных переносах приложений из других областей имеются в [154]. В [165] рассмотрены вопросы использования SYCL на аппаратных средствах Nvidia и AMD благодаря применению hipSYCL с разными бэк-эндами (теперь hipSYCL и openSYCL именуются *AdaptiveCpp*).

Задачи переноса программных кодов с CUDA на oneAPI рассматривались уже в ряде статей, поскольку средства oneAPI применимы к разным графическим процессорам, появившимся и до PVC. Так, в [166] для задач численного интегрирования перенос с CUDA (с V100) на oneAPI потребовал специальной ручной оптимизации, прежде чем производительность с oneAPI стала несильно ниже, чем с CUDA. В [167] при переносе кода для CFD с неструктурированной сеткой на SYCL для A100 и V100 был использован hipSYCL, но сделан вывод, что для достижения максимальной оптимизации по-прежнему CUDA лучше. Для этой же области CFD в [168] был получен аналогичный результат. В этих работах отмечено, что более хорошие результаты для DPC++/SYCL могут получаться в дальнейшем по мере роста качества оптимизации компиляторами. В докладе на суперкомпьютерной конференции в NASA по переносу с CUDA на oneAPI программного комплекса FUN3D (для CFD, с решением уравнений Навье-Стокса) [169] по сути также указано на целесообразность ручной оптимизации кода на конкретную архитектуру.

Следует указать и на определенный недостаток oneAPI, связанный исключительно с базированием на C++, поскольку это не помогает кодам, исходно использующим современный Fortran. Для CUDA имеется аналог – CUDA Fortran [170]. При этом к современному языку Fortran

и при работе на GPU стали относить версии стандарта с поддержкой объектно-ориентированных средств [171]. В современных версиях OpenMP, реализованных и в компиляторах Fortran [172], имеются подходящие средства для работы с GPU. Но указанный выше недостаток связан с задержками разработок возможных новых расширений современных стандартов Fortran со средствами для эффективной работы с GPU.

Более подробный анализ oneAPI от Intel здесь нецелесообразен ввиду крайне малого количества научных публикаций, демонстрирующих производительность PVC по сравнению с другими современными GPU, включая сравнения с работой с другими программными средствами – например, CUDA. Intel добилась многого в первую очередь с oneAPI, поскольку эти средства, включая DPC++, стали применяться и на других графических процессорах Intel – например, в работе с известной библиотекой Ginkgo [173]. Кроме того, уже реализуется и обучение DPC++ студентов, изучающих программирование гетерогенных вычислительных систем [174].

Среди единичных публикаций с данными о производительности Data Center GPU Max укажем на две работы, в которых производительность модели Max 1100 была сопоставлена с GPU Nvidia и AMD. Статья [175] ориентирована на поддержку современных версий OpenMP, позволяющих работать и на GPU. Здесь на Max 1100 (с применением Intel oneAPI 2023) и на A100-40GB (с применением Nvidia NVHPC 23.3) проведены расчеты по относящемуся к области CFD мини-приложению LULESH с использованием OpenMP-распараллеливания, и утверждается, что A100 был на 34% лучше, чем Max 1100 (хотя выигрыш в производительности у A100 зависел от размера используемой в расчете задачи и менялся от 15 до 42%). Однако большая емкость памяти Max 1100 позволила провести расчет по LULESH для большей размерности, где памяти на A100 не хватало.

В [176] исследована переносимость SYCL на разные аппаратные платформы и получены данные о производительности на ряде различных использующих распараллеливание с применением SYCL приложений (в основном в области CFD), выполнявшихся на Max 1100 (с oneAPI 2023.1), A100-40GB (с CUDA 11.6) и MI250X (с ROCm 5.4.2; здесь использовалась только одна из двух GCD-частей полного GPU, см. об этом ниже в разделе 5.1.1).

Все отобранные приложения являются связанными памятью (она лимитирует их производительность) и используют методы структурированных и неструктурированных сеток. На тесте BabelStream triad полученная в [176] пропускная способность памяти составила 1310 ГБ/с для A100, 1290 ГБ/с для MI250X и 803 ГБ/с для Max 1100.

В этих приложениях применялись высокоуровневые специализированные для таких методов средства распараллеливания, которые генерируют

разные более низкие коды распараллеливания, в том числе SYCL. В [176] на каждом GPU для каждого приложения были использованы разные варианты распараллеливания. В оптимальных для производительности A100 вариантах этот GPU во всех приложениях был заметно быстрее, чем Max 1100 во всех вариантах. При аналогичном сопоставлении производительности Max 1100 с MI250X в одних случаях быстрее был Max 1100, в других – MI250X.

Рассмотренная далее информация о производительности PVC относится уже к модели Max 1550. В докладе [120] приведены данные об ускорении в PVC (с применением средств SYCL в DPC++) относительно A100 с использованием CUDA (хотя были приведены и данные для A100 с SYCL) – по 8 тестам производительности, из которых к традиционной области HPC можно отнести SYCL HP Linpack (очевидно, реализация HPL с применением SYCL). Здесь было достигнуто ускорение в 1,5 раза. Среди других тестов самым известным представляется Google-тест для хэширования, hashtable, в котором было достигнуто максимальное ускорение в 2,5 раза. В других тестах достигнутое ускорение было в диапазоне от 1,4 до 1,8. Понятно, что все эти числа требуют более детального уточнения. Интересно, что в некоторых этих тестах применение SYCL на A100 дало процентов на 10 больше производительность, чем при работе с CUDA.

Другие данные о производительности PVC приведены в [116] – об ускорении в 12,8 раз расчета по знаменитому программному комплексу молекулярной динамики LAMMPS на двухпроцессорном сервере с Intel Xeon Max 9480 и с шестью Data Center GPU Max- относительно двухпроцессорного сервера с Xeon 8380. Учитывая, что здесь использовано 6 GPU, а сопоставление проведено с Xeon 8380, которые уступают по максимально достигнутой производительности с плавающей запятой в широко распространенных тестах SPEC CPU2017 процессорам AMD EPYC 7763 в вариантах speed и rates [177], интереснее было бы сравнение производительности с другими современными GPU.

Актуальной публикацией, дающей реальные данные о производительности Data Center GPU Max 1550 с сопоставлением относительно A100-80GB является [178], где приводятся не только данные о достигаемой производительности, но и о переносе кода с CUDA на SYCL. Здесь с использованием формата FP32 проведены вычисления молекулярным докингом (предельно упрощенная специализированная методика для расчета ориентации близко расположенных молекул, например, лигандов относительно протеина), которые часто используются для конструирования лекарств.

Для этого был осуществлен перенос CUDA-варианта известной программы AutoDock-GPU на SYCL с применением средств DPCT с последующей ручной доработкой. При переходе от вычислений на одном стеке Max 1550 к расчетам на двух стеках (просто изменяя переменную окружения) производительность разных тестовых расчетов возрастала от 6% до 58%.

Стоит отметить, кстати, и найденное соотношение достигаемой на A100 производительности расчетов по CUDA и SYCL-версиям AutoDock. Из 10 проведенных расчетов CUDA-вариант оказался быстрее (от 1,24 до 3,64 раза) в девяти, а SYCL-вариант был быстрее в одном.

Расчеты по SYCL-версии AutoDock использованием обоих стеков Max 1550 были быстрее, чем на A100 с CUDA-версией в 7 из 10 расчетов (максимально – в 1,88 раза быстрее); A100 был успешнее в более крупных расчетах (с большим числом атомов или числа вращающихся связей). Но все достигнутые ускорения Max 1550 относительно A100 были меньше, чем отношения пиковых производительностей (для FP32) этих GPU. Как отмечено в [**178**], расчеты здесь носили пока предварительный характер, и в коде AutoDock предполагаются усовершенствования.

В [**179**] получены данные о достигаемой производительности PVC при использовании средств пакетной линейной алгебры из библиотеки ginkgo для итеративного решателя уравнений Навье-Стокса (в приложении решеточной гидродинамики PeleLM). При переключении с режима использования одного стека Max 1550 на два стека производительность возрастает в 1,5–2 раза, и больший рост производительности относится к задачам большей размерности.

В [**179**] получены также аналогичные данные о производительности A100-80GB и H100-PCIe. Сопоставление этих результатов показывает, что Max 1550 с использованием одного стека быстрее H100 в среднем в 1,3 раза, а при использовании двух стеков – в 2,4 раза.

Надо отметить, что эта работа также носит во многом предварительный характер. Так, в ней указана пиковая производительность Max 1550, равная 45,8 TFLOPS – меньше, чем современное «официальное» значение 52 TFLOPS (вероятно, была доступна более низкая тактовая частота). В расчетах на A100 и H100 использованы средства CUDA 11.8.0, в то время как на H100 обычно применяется версия 12 (например, во всех данных Nvidia о производительности H100 в тестах MLPerf используется версия 12.2, см. раздел 4.2.5).

Даже эти немногие публикации показывают высокую достигаемую производительность PVC. Данных о пиковой производительности недостаточно для качественных оценок реальной производительности. Большая пиковая производительность Max 1550 по сравнению с аналогичным показателем H100 не обязательно дает преимущества в реально достигаемой производительности приложений. Нужно также отметить, что приведенные выше в таблице 7 показатели пиковой производительности не включают данные о пиковой производительности GPU Nvidia и AMD для FP64 с использованием тензорных ядер (с их применением показатели старших моделей этих GPU выше, см. таблицу 20), поскольку XMX в PVC этот формат не поддерживают. Ясно, что для практических успехов PVC и средств oneAPI/DPC++ предстоит проделать еще большой объем работы, требующий определенного времени.

3.3. Резюме по Intel PVC

Сложное в технологическом плане производство PVC может давать не очень высокий процент выхода годных микросхем PVC (что могло дать и замедление поставок PVC для Aurora, который в июньском списке Top500 2023 года отсутствует) и способствовать повышению стоимости. Фактически об этом же высказано ранее в [**180**]. Как Intel Max 1550, так и AMD MI250X можно считать состоящими из двух логических GPU. Возникает вопрос, какие сопоставления проводить – на один логический GPU или на полный (физический). Вопрос связан и со стоимостными показателями, которые могут сильно различаться.

Если сравнить с целым AMD MI250X, PVC по пиковой производительности немного опережает MI250X на обычных векторных операциях (см. таблицу 7), но имеет и немного более высокий TDP. При этом опережение по пиковой производительности новых GPU от Intel и AMD относительно Nvidia H100 не означает опережение в реальной производительности на приложениях.

Недостатком PVC для определенных приложений в области HPC можно считать отсутствие поддержки формата FP64 в XMX. При использовании тензорных ядер в H100-SXM4 (или аналогов в MI250X) эти GPU существенно опережают PVC по пиковой производительности с двойной точностью (см. также таблицу 20 ниже).

С учетом того, что Nvidia уже предлагает суперчип GH200 Grace Hopper с интеграцией H100 с ARM-процессорами (см. раздел 4.2.3), а на выставке электроники CES (Consumer Electronics Show) в начале 2023 года AMD анонсировала GPU MI300 с интегрированным ЦП (в виде 3D-чиплета, предположение о готовности – на 2023 год), с точки зрения автора, существенно более широкое применение GPU Intel в HPC может реализоваться несколько позже, после реализации ожидаемых успешных технологических процессов Intel (особенно 18A), которые не ожидаются в 2023 году. От планируемого ранее на 2023 год «второго поколения» PVC с кодовым именем Rialto Bridge [**180**], содержащего 160 ядер X^e, Intel недавно отказалась, так что следует ожидать скорее выпуск GPU Falcon Shores или его последующий вариант XPU с интеграцией GPU и ЦП x86 [**181**] – вероятно, все определяющим является успех в технологии.

Чуть ли не более актуальным (по сравнению с конкуренцией Intel с Nvidia за часть рынка GPU – где участвует также AMD) автору сейчас представляется внедрение при поддержке Intel программного обеспечения для GPU в направлении SYCL—> DPC++ (и oneAPI), поскольку это не только позволяет дальше отойти от узкой ориентации на конкретные GPU (Nvidia CUDA и отчасти AMD HIP), но и может стать когда-то

эффективным для новых многоядерных серверных процессоров, число ядер в которых все продолжает расти. Возможно, применение SYCL может сразу стать наиболее привлекательным для модернизации использующих C++ приложений, еще не портированных на GPU.

Однако не следует и преувеличивать возможности нового стандарта SYCL. Так, появление в средствах CUDA для H100 нового уровня в иерархии системы нитей – кластера блоков нитей (см. раздел 4.2.5) – не имеет аналогов в SYCL. Следует также иметь в виду высказывания, что SYCL не решает всех проблем переносимости производительности, и может возникать необходимость использования разных алгоритмов для разных аппаратных средств [**176**].

4. GPU от Nvidia: от A100 до H100

Как отмечено выше, к базовым GPU здесь отнесены Nvidia V100 и A100, которые наиболее широко используются в настоящее время для областей HPC и ИИ, в том числе в суперкомпьютерах. Поскольку V100 стал доступен еще в 2017 году, в обзоре на уровне микроархитектур речь идет только об A100; показатели V100 приводятся только в сопоставляющих таблицах. Кроме того, упоминаются наиболее важные вычислительные блоки и типы данных, впервые появившиеся в A100 (которых не было в V100). Подробная информация о всех улучшениях различных компонент микроархитектуры A100 относительно V100 имеется в [**73**].

Самый современный из «базовых» GPU Nvidia, A100, выпускается уже с 2020 года, и подробного анализа его микроархитектуры здесь не проводится, считая соответствующую информацию уже достаточно хорошо известной. Но в микроархитектуре H100, естественно, очень многое совпадает с A100, и соответственно после анализа A100 будет удобно говорить в основном об усовершенствованиях в H100. В соответствии с этим здесь приводится только рисунок SM от H100 (A100 здесь лишь немного отличается – и ниже будет просто указано, в чем отличия). Аналогичное относится и к программному обеспечению – в обзоре больше внимания уделяется тому, что целесообразно использовать на GPU нового поколения.

Данные о достигаемой производительности A100 в приложениях и тестах производительности также рассматриваются отчасти ограничено, так как основной целью обзора были GPU нового поколения, производительность которых и рассматривается в том числе со сравнением ее относительно V100 и A100.

4.1. GPU A100

4.1.1. *Микроархитектура A100*

В современных поколениях ЦП рост производительности для HPC осуществляется в первую очередь за счет прогресса микроархитектуры, а не за счет улучшения ISA. В современных GPU Nvidia данные о микроархитектуре также важнейшие, но для максимизации производительности, особенно для задач ИИ, особенно важны и новые улучшения/расширения всех уровней API CUDA и низкоуровневого варп. Средства CUDA можно отнести к низкоуровневым потому, что можно работать на более высоком уровне – например, использовать только готовые функции библиотек, или работать с директивами. Есть более низкий по отношению к CUDA уровень, где используется виртуальная система команд (ISA) для GPU Nvidia – PTX (Parallel Thread Execution) [**182**], и соответственно ассемблер [**183**], но он, естественно, применяется при программировании очень редко. Строго говоря, и PTX является промежуточным уровнем – он затем преобразовывается в двоичный код для конкретной модели GPU.

Под улучшением архитектуры в первую очередь в обзоре имеется в виду микроархитектура. Но есть более общая архитектура – для разных моделей графических процессоров в микроархитектурах сохраняются ее некие базовые общие особенности.

В A100 используется архитектура Ampere GA100. A100 был первым GPU, выпущенным с такой архитектурой – в 2020 году. Затем появились в том числе менее вычислительно мощные модели GA100. В них использовано меньшее число SM, и они содержат меньшее число тензорных ядер, но более высокопроизводительных. Но в отличие от A100 в их тензорных ядрах нет поддержки FP64 [**184**]. Кроме того, после введенных США санкций в отношении Китая Nvidia стала производить не подпадающие под эти ограничения GPU A800 с уменьшенными по сравнению с A100 вычислительными возможностями. Информация об этих GPU имеется, например, в известной базе данных Techpowerup о производимых в мире GPU [**185**] (пользоваться ею потребителям предлагали в Nvidia). В обзоре далее рассматривается только A100.

Наиболее полным изложением архитектуры A100 является [**73**], на котором и основано последующее рассмотрение его микроархитектуры в данном обзоре. Изложение в [**73**] отчасти интегрировано с распараллеливанием на базе SIMT (с использованием CUDA) из-за глубокой связи CUDA с аппаратными средствами GPU Nvidia. Это относится в том числе к иерархии различных видов памяти в GPU и CUDA (важнейших компонент для реализации высокой производительности), и к иерархии различных уровней образования больших групп параллельно выполняемых нитей в CUDA. В таблице 1 все это перечислено вкратце в терминологическом

плане. Однако не все используемые в CUDA типы памяти имеют однозначный аппаратный эквивалент, бывает и интеграция их разных типов в общих аппаратных средствах A100 (см. ниже в рассмотрении иерархии памяти A100). Кроме того, [**73**] отсылает иногда к рассмотренному в описании Nvidia архитектуры V100.

Базовые показатели A100 в сопоставлении с аналогичными показателями V100 и H100 приведены в таблице 10. В технологическом плане за счет более современной 7нм технологии TSMC в A100 при практически той же площади кристалла число транзисторов возросло более чем вдвое по сравнению с V100, а TDP увеличился не так сильно.

Таблица 10. Сопоставление важных для традиционных HPC показателей GPU Nvidia V100, A100 SXM и H100 (данные из [**73**, **78**, **185**])

Показатели GPU	V100	A100 SXM	H100 PCIe	H100 SXM
Архитектура	Volta	Ampere	Hopper	Hopper
Технология от TSMC	12 нм FFN	7 нм N7	4N[1]	4N[1]
Площадь кристалла, мм^2	815	826	814	814
Число транзисторов (млрд)	21,1	54,2	80	80
TDP (Вт)	300	400	350	700
Форм-фактор	SXM2	SXM4	PCIe-v5	SXM5
Число SM	80	108	114	132
Векторных ядер FP64 в SM	32	32	64	64
Векторных ядер FP32 в SM	64	64	128	128
Ядер INT32 в SM	64	64	64	64
Тензорных ядер в SM	8	4[2]	4	4
Повышенная частота, ГГц	1,53	1,41	1,76[4]	1,98[4]
Емкость файла регистров в SM	256 КБ	256 КБ	256КБ	256КБ
Емкость разделяемой памяти в SM	До 96 КБ[3]	До 164 КБ[3]	До 228 КБ[3]	До 228 КБ[3]
Емкость кэша L2	6144 КБ	40960 КБ	50 МБ	50 МБ
Тип/емкость памяти, Гбайт	HBM2/16 или 32	HBM2E/40 или 80	HBM2E/80[5]	HBM3/80[5]
Ширина шины памяти, бит	4096	5120	5120	5120
Частота памяти, МГц	876	1215 или 1593	1593	1313
Пропускная способность памяти, ГБ/с	897	1555 или 2039	2039	1681

[1] настроенная для Nvidia;
[2] Число матричных операций умножить-и-сложить за такт в тензорных ядрах A100 в 4 раза больше, чем у V100;
[3] Емкость разделяемой памяти в V100 и A100 конфигурируема;
[4] для FP32 и FP64, для меньшей точности частота немного меньше;
[5] есть модель с HBM3/96 ГБ и другой частотой GPU.

В таблице 10 приведены данные для двух разных моделей A100 с форм-фактором SXM, отличающихся емкостью памяти (40 или 80 ГБ). Кроме того, имеются две другие модели A100 с межсоединением PCIe,

которые также могут иметь емкость 40 или 80 GB. Такое несколько ограниченное представление моделей A100 в данной таблице выбрано по двум причинам. Во-первых, для удобства чтения и сопоставления с другими моделями GPU от Nvidia в формате данной публикации. Во-вторых, обсуждаемые в этом месте обзора пиковые производительности (см. таблицу 12 ниже) рассчитывались для повышенных тактовых частот, которые у всех четырех разных моделей A100 совпадают. Кроме того, позднее в разделе 5, описывающем GPU нового поколения от AMD, с которыми сопоставление всех параметров, включая производительность, представляется одним из наиболее актуальных в данном обзоре, соответствующие показатели разных моделей A100 приведены в удобном более развернутом виде (см. таблицы 20 и 21).

Для стартового понимания производительности A100 (для начала пиковой) естественно базироваться на исполнительных блоках SM: пиковая производительность всего A100 просто пропорциональна числу SM, которое указано в таблице 10. В иерархии от одного SM до полного A100 имеется 2 вида кластеров: кластеры обработки текстур (TPC, Texture Processing Clusters), содержащие по 2 SM, и кластеры GPC, содержащие по 8 TPC. В архитектуре Ampere (в GA100) допустимо иметь 8 GPC и соответственно 128 SM [**73**]. Смысл GPC с точки зрения GPGPU (если отвлечься от текстур) – это группа SM, физически расположенных близко друг к другу, что дает локальность параллельно обрабатываемых данных с доступом к ним с более высокой пропускной способностью и более низкими задержками (так сказать, некий локальный аналог PIM).

Общее построение SM, из множества которых складывается вычислительная мощность A100 (и V100, и H100) видно на рисунке 6. Хотя это рисунок SM из H100 [**78**]), на макроуровне рисунка отличия от H100 у A100 просматриваются легко: в A100 нет Tensor Memory Accelerator; емкость D-кэша L1 192 КБ против 256 КБ на H100; и в H100 соответственно новая версия тензорного ядра. Но главное – это то, что в H100 в каждом из 4 разделов SM (что здесь имеется в виду под разделом SM, понятно из рисунка 6) есть в 2 раза больше векторных устройств FP64, FP32 и INT32. В A100 в каждом из разделов имеется по 8 блоков FP64, и по 16 блоков FP32 и INT32.

Поскольку в SM выполняются варпы, состоящие из 32 нитей, в каждом разделе имеется варп-планировщик, дающий 32 нити за такт, и диспетчер с той же «производительностью». Здесь проявляется тесное переплетение аппаратных средств GPU Nvidia и средств CUDA. Блок нитей CUDA (включающий варпы) приписан к одному SM. Уточнение действий варп-планировщика рассматривается не в [**73**], а в руководстве по CUDA [**16**], где указано, что SM статически распределяет свои варпы между своими планировщиками. Затем каждый планировщик выдает одну инструкцию

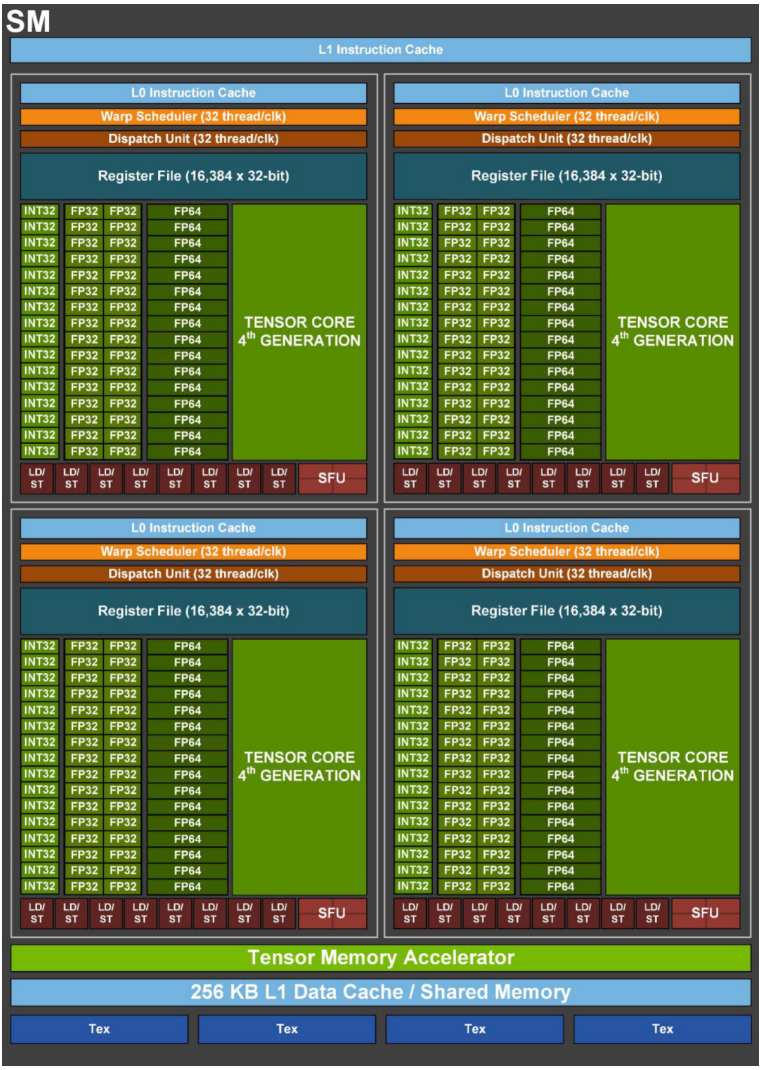


Рисунок 6. Общее построение SM в H100 (рисунок из [78])

для одного из назначенных ему варпов, который готов к выполнению, если таковой имеется. Варп-планировщик служит для обеспечения загрузки вычислительных устройств раздела SM: если варп ожидает, например, завершения операций с памятью, то выполняться может другой варп, а диспетчер перенаправляет отобранные команды в вычислительные устройства (данные они берут из регистров).

Действия варп-планировщика могут зависеть от версии вычислительных возможностей (Compute Capability), описанных в [16] для разных моделей GPU; сказанное выше справедливо для V100, A100 и H100. Эти возможности для данных моделей приведены в таблице 11 по данным [78].

Таблица 11. Технические характеристики вычислительных возможностей (CC) разных версий в зависимости от GPU Nvidia

Параметры	V100	A100	H100
Версия Compute Capability	7.0	8.0	9.0
Нитей на 1 варп	32		
Максимум варпов на 1 SM	64		
Максимум блоков нитей (CTA) на 1 SM	32		
Максимум блоков нитей / кластеров блоков нитей	Нет		16
Максимальное число 32-разрядных регистров на нить	255		
Максимальное число регистров на 1 SM	65536		
Максимальное число регистров на блок	65536		
Максимальное число нитей в блоке	1024		
Разделяемая память на 1 SM, КБ	До 96	До 164	До 228
Число ядер FP32 на 1 SM	64		128
Число ядер FP64 на 1 SM	32		64

Каждый из 4 разделов SM имеет файл регистров, и 8 устройств загрузки/сохранения для доступа к памяти. У каждого раздела SM есть свой I-кэш L0 (I-кэш L1 является общим для всего SM), но обычно производительность в HPC и ИИ не требует специальной ориентации кода на их применение. В анализе производительности GPU Nvidia вообще часто стартуют с уровня SM, поскольку они содержат и общие на все 4 раздела аппаратные компоненты, в том числе текстурную память и общий D-кэш L1 (SM можно считать некоторым аналогом процессорного ядра в ЦП).

Каждый SM в A100 и в V100 имеет по 32 векторных CUDA-ядра для FP64, что позволяет SM выполнять по 32 операции «умножить-и-сложить» за такт, что дает 64 FLOPS за такт (для FP32 – все в два раза больше). Соответственно пиковая производительность GPU при работе с векторными ядрами получается умножением 64 FLOPS на число SM (их в A100 больше, чем в V100) и на тактовую частоту, что и дает более высокую производительность у A100 (см. таблицу 10). Но основной прогресс в производительности теперь ориентирован в первую очередь на ИИ и соответственно на применение тензорных ядер, которые могут давать больше результатов за такт, чем обычные векторные блоки – и соответственно более высокие пиковые производительности для различных форматов данных.

Тензорные ядра в A100 работают с матрицами разных возможных размеров в зависимости от используемых форматов данных (список всех возможностей имеется в [16]). Для традиционных HPC нужна работа с FP64, что впервые стало поддерживаться в тензорных ядрах на аппаратном уровне в A100 – об FP64 здесь и пойдет речь. Данные о пиковой производительности для других форматов данных, в первую очередь актуальных для ИИ, приведены в таблице 12.

Таблица 12. Пиковые производительности GPU V100, A100, H100 (TFLOPS, для целочисленных операций – TOPS)

Формат данных	V100	A100	H100 PCIe	H100 SXM4
FP16	31,4	78	204.9	267.6
FP16[1]	125	312/624	756/1513	989,4/1978,9
BF16[1]	—	312/624	756/1513	989,4/1978,9
FP32	15,7	19,5	51,2	66,9
TF32[1]	—	156/312	378/756	494,7/989,4
FP64	7,8	9,7	25,6	33,5
FP64[1]	—	19,5	51,2	66,9
INT8[1]	—	624/1248	1513/3026	1978,9/3957,8

[1] значения при использовании тензорных ядер;

Данные из [73, 78, 185]. Через слэш приведены производительности с применением разреженности, когда она доступна.

В A100 для двойной точности появилась аппаратная реализация вычислений для формулы (1), новая матричная команда «умножить-и-сложить» (Double Precision MMA), которая заменяет 8 аналогичных команд DFMA в V100 и дает 128 результатов FP64 за такт – в 2 раза больше, чем V100 [73]. Для FP64 в A100 на самом низком варп-уровне применима операция WMMA, с которой можно работать с матрицами размерностей 8×4 или 4×8 и аккамуляторными матрицами 8×8 (т. е. $M = N = 8$ и $K = 4$ для формулы (1)) [16]. Как указано в [89], операции WMMA могут осуществляться над чуть более крупными матрицами, чем поддерживается аппаратурой тензорного ядра, и при выполнении операции WMMA одновременно используется несколько тензорных ядер.

В поставляемом Nvidia A100 имеется 108 SM [73], так что достигаемая пиковая производительность равна $108 \times 128 \times \nu$, где ν – тактовая частота. Отсюда следует, что информация о пиковой производительности, приводимая Nvidia [73] и использованная в таблице 12, относится к ускоренной (boost) частоте. Пользователи A100 имеют возможность управлять тактовой частотой, что позволяет регулировать TDP и стабилизировать измеряемую производительность – так, в [186]) для исследований достигаемой GEMM производительности использовалась частота 1005 МГц.

Современный профилировщик Nsight Compute [**187**] для A100 фиксирует по умолчанию базовую тактовую частоту, что дает повторяемость результатов, но соответственно более низкие пиковые производительности, чем указанные в таблице 12. Базовые частоты зависят от конкретной используемой модели A100 (их 4 самых важных для данного обзора – на них обычно производились расчеты и выполнялись тесты производительности для HPC и ИИ), они отличаются емкостью HBM2E и форм-фактором. В таблице 13 приведены величины их базовых частот из базы данных [**185**].

Кроме того, в разных моделях A100 отличаются и используемые частоты памяти и соответственно ее пропускная способность. Только две модели A100 с емкостью памяти 40 ГБ имеют одинаковые частоты, а поскольку ширина шины памяти (5120 бит) одинакова у всех моделей A100, то и теоретическая пропускная способность у моделей A100 с 40 ГБ памяти совпадает (ее можно посчитать, умножив ширину шины на частоту памяти). В таблице 13 также приведены частоты памяти разных моделей A100 (данные из [**185**]).

Таблица 13. Базовые тактовые частоты различных моделей A100 и их памяти

Модель GPU A100	Базовая частота	Частота памяти
A100-PCIe-40GB	765 МГц	1215 МГц
A100-SXM4-40GB	1095 МГц	1215 МГц
A100-PCIe-80GB	1065 МГц	1512 МГц
A100-SXM4-80GB	1275 МГц	1593 МГц

Но вернемся к определяющим максимальную пиковую производительность A100 тензорным ядрам. Они даже при таких маленьких аппаратно поддерживаемых матрицах имеют еще возможность учета в них мелкозернистой разреженности, что относится к форматам пониженной относительно FP64 точности и дает там повышение пиковой производительности в 2 раза [**94**] (см. таблицу 10). Эта разреженность в V100 не поддерживалась и ориентирована в основном на задачи ИИ (см. подробнее в [**73**]).

Поскольку многим приложениям HPC умножения матриц не требуются (целесообразность поддержки его аппаратными средствами вообще дискутируется [**90**]), не менее важной для производительности A100 с двойной точностью является пиковая производительность векторных CUDA-ядер FP64, не имеющих отношения к тензорным ядрам (см. рисунок 6).

В одном SM имеется 32 векторных блока FP64 (соответствующие векторные блоки FP32, которых в каждом SM в два раза больше, традиционно именуются CUDA-ядрами) [**73**]. Для всех 108 SM это дает соответственно 6912 результатов за такт (благодаря применению команды «умножить-и-сложить»), что после умножения на ускоренную тактовую частоту и дает пиковую производительность A100 в 9,7 TFLOPS для двойной точности – без применения тензорных ядер (см. таблицу 10).

Кроме того, в каждом SM имеется 4 блока SU, ускоряющих расчет трансцендентных функций. Это может быть практически важным для самых разных приложений, однако SFU были доступны в GPU Nvidia давно, еще до V100, но только для формата FP32 (подробнее см. [188]).

Поскольку достигаемая производительности GPU так часто зависит от пропускной способности памяти, теперь следует обратиться к иерархии памяти в A100. Поскольку традиционно большое количество параллельно выполняемых на GPU нитей требует много регистров, в A100, как и в более старых GPU, в каждом из 4 разделов SM имеется файл 32-разрядных регистров емкостью 64 КБ (см. рисунок 6). Каждая выполняемая нить использует собственные локальные регистры из соответствующего файла.

Следующим за регистрами в иерархии памяти уровнем в A100 является D-кэш L1 и разделяемая память с общей емкостью 192 КБ. Он обеспечивает высокую пропускную способность и низкую задержку; реальные данные имеются для V100 [189]. Эта память является общей для всего SM (и соответственно для блока нитей в CUDA). В CUDA возможно динамическое изменение объема разделяемой памяти.

Предпоследний уровень иерархии памяти на пути к глобальной памяти HBM2E – кэш L2 емкостью 40 МБ – рассматривается в [73] в общем с HBM2E подразделе. Пространства этих уровней памяти являются общими для всех SM и всех работающих на GPU приложений. Емкость кэша L2 по сравнению с V100 выросла более чем в 6 раз. Этот кэш считывает и записывает в память HBM2E устройства. Для более высокого распараллеливания в A100 для каждого SM была нужна более высокая пропускная способность. Для этого кэш L2 был образован в виде иерархии разделов, и каждый раздел кэширует данные ближе к доступу к SM, что снижает задержку [94].

Возросшие по сравнению с V100 ширина интерфейса HBM2 и частота памяти дали рост пропускной способности примерно в 1,7 раза (см. таблицу 10). Чтобы увеличить эффективные величины пропускной способности DRAM и емкости кэша L2, в A100 для задач ИИ имеется аппаратура сжатия разреженных данных (достигается сжатие до 4 раз в DRAM, до 2 раз в L2) [190].

В CUDA используются свои типы памяти, некоторые из которых реально располагаются вместе на одном типе аппаратной памяти. В отличие от разделяемой памяти, локальная память является личной памятью каждой нити. В терминологии CUDA области глобальной и локальной памяти (в отличие от глобальной она кэшируется [16]) именуются памятью устройства и располагаются в HBM2E. Постоянная память (живущая только во время работы приложения, в этой памяти разрешено только чтение) располагается в памяти устройства (постоянная память кэшируется). Что касается памяти текстур, она располагается в каждом SM (см. рисунок 6) и кэшируется в специальном кэше [73].

В A100 появились средства MIG, ориентированные на решение проблемы возможной для ЦОД проблемы недостаточности использования ресурсов A100 определенными приложениями, где применение A100 стало бы соответственно неэффективным. A100 может функционировать как до 7 изолированных экземпляров графических процессоров [73], перенастраиваемых на лету для удовлетворения динамически изменяющихся потребностей [94], т.е. получаются виртуализированные GPU, имеющие более маленькие вычислительные ресурсы.

Новая функция MIG обеспечивает улучшенную изоляцию клиентов (в том числе виртуальных машин и процессов) и QoS для многопользовательских и виртуализированных GPU, что особенно полезно для поставщиков облачных услуг [73]. Для MIG очень важны новые технологии обработки сбоев в архитектуре Nvidia Ampere – для обеспечения надлежащей изоляции и безопасности между клиентами, использующими один GPU.

Как указано в [94], поддерживаются два типа экземпляров MIG. Один тип изолирует вычислительные ресурсы, но не систему памяти, что позволяет операционной системе планировать процессы с упрощенным администрированием. Другой тип дополнительно обеспечивает функциональную изоляцию и изоляцию производительности в системе памяти. В этом случае A100 назначает каждому MIG свои собственные физические пути (в том числе к кэшу L2 и интерфейсу памяти), обеспечивая дополнительную безопасность для облачных вычислений [94].

Использование MIG позволяет расширить целесообразность применения A100 на более широкую область работ. Подробную информацию о настройке и использовании MIG см. в [191].

В руководстве Nvidia по A100-PCIe-80GB [192] описаны возможности средств программного управления электропитанием, позволяющие настроить предельную потребляемую мощность графической платы или производительность на Ватт. В A100 имеется, естественно, еще много важных для достижения высокой производительности возможностей, среди которых следует в первую очередь указать на поддержку асинхронного копирования (на уровне ISA) – оно загружает данные непосредственно из глобальной памяти в разделяемую память в SM, минуя файл регистров, и может выполняться в фоновом режиме, пока SM выполняет другие вычисления (это заодно снижает энергопотребление). Подробнее об этом и других возможностях A100 см. [73]; краткое рассмотрение асинхронного выполнения передач данных и вычислений проводится далее в разделе 4.2 про H100, в котором соответствующие возможности были существенно расширены.

Описание межсоединений A100 с использованием каналов NVLink проведено ниже в разделе 4.1.2, поскольку оно отчасти относится уже к дополнительным к GPU аппаратным средствам – так, это взаимосвязано уже и с соединениями с ЦП через PCIe.

4.1.2. *Межсоединения для A100 и вычислительные системы с A100: от серверов до суперкомпьютеров*

Межсоединения NVLink для связи между несколькими GPU Nvidia в сервере возникли еще до появления V100. Исходно возникала проблема недостаточно высокой пропускной способности PCIe для связи GPU с ЦП. Общий обзор используемых для GPU межсоединений различных фирм имеется в [**193**]. NVLink – это канал, а устройства могут иметь по несколько каналов, и связываются через ячеистую сеть.

Естественно, очень много общего с NVLink3 в A100 имеет вторая версия (NVLink2), используемая в V100. В [**194**] подробно рассмотрены различные показатели производительности работы NVLink2 с V100. NVLink2 (вместо более медленного PCIe 3.0) используется и для связи с IBM Power9, и обеспечивает когерентность кэша.

Как и PCIe, NVLink2 является пакетной шиной, но NVLink2 эффективнее при передаче небольших пакетов – с 16 байт заголовка можно передать 256 полезных байт. В этом межсоединении используется топология ячеистой структуры для соединений точка-точка, что дает более высокую общую пропускную способность по сравнению с топологией дерева в PCIe. Соединения состоят из нескольких полнодуплексных каналов, которые обмениваются данными со скоростью 25 ГБ/с в каждом направлении. Устройство имеет до шести каналов, и их можно объединить в 3 канала с общей скоростью по 75 ГБ/с. И PCIe, и NVLink обеспечивают двунаправленные передачи, но NVLink имеет еще прямой доступ к страничной памяти ЦП [**194**]. В [**194**] приводятся данные о достигаемой пропускной способности и задержках NVLink2.

Применение коммутатора для связи между GPU Nvidia не только способствует росту масштабируемости производительности – но дает крайне важное (особенно для современных задач ИИ) увеличение объема доступной памяти GPU. NVSwitch является неблокирующимся коммутатором, который имеет 18 портов NVLink, и дает возможность подсоединения через NVLink до 16 GPU, обеспечивая каждому каналу двунаправленную пропускную способность 50 ГБ/с – и соответственно суммарную пропускную способность коммутатора 900 ГБ/с [**195**]. Например, в реализации на материнских платах, содержащих по 8 GPU, каждый из них подключен к 6 коммутаторам платы, имеющим связи и с коммутаторами другой платы. Коммуникации GPU внутри платы требуют один проход через NVSwitch, с GPU второй платы – 2 прохода через NWSwitch. Для обеспечения надежности коммуникаций в каналах при передаче используется циклическое избыточное кодирование (CRC), а внутри коммутаторов (например, для маршрутизации) применяется код ECC (подробнее см. в [**195**]). Ограничения по пропускной способности на серверах с V100 могут скорее возникать при передаче данных по PCIe между GPU и CPU.

В современных multi-GPU серверах с A100 используются уже NVLink3. Низкоуровневые детали, объясняющие, за счет чего удалось достигнуть повышения пропускной способности и снижение задержек относительно V100 с NVLink2, рассмотрены в [94]. С точки зрения пользователя главным является то, что при практически не изменившейся пропускной способности каналов их число возросло с 6 на V100 до 12 на A100.

В multi-GPU серверах, использующих NVLink, соответственно возможны различные конкретные топологии [190]. Например, в ориентированном на задачи ИИ сервере DGX A100 имеется 8 GPU и 6 микросхем NVSwitch, и каждый GPU подключается к каждому NVSwitch с помощью двух каналов NVLink3 [196]. Это иллюстрируется рисунком 7 (для связи с EPYC Rome используется PCIe-коммутатор PEX) [73].

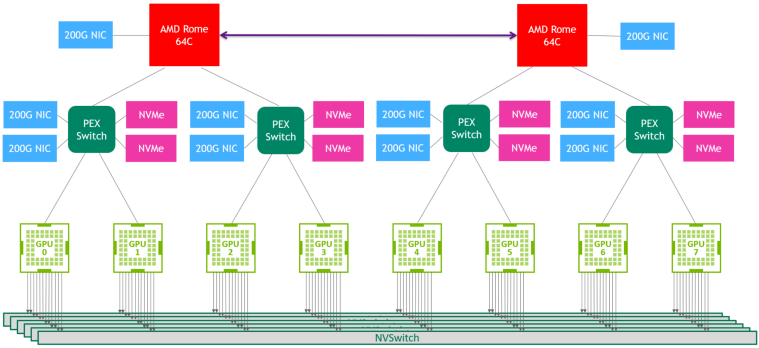


Рисунок 7. Межсоединение GPU в сервере DGX A100 (рисунок из [73])

NVLink3 обеспечивает также улучшенные функции обнаружения ошибок и восстановления [190]. Хотя, как указано выше, один канал NVLink в A100 имеет пропускную способность, как и в V100, – 25 ГБ/с в каждом направлении, но он использует вдвое меньше пар сигналов на канал по сравнению с V100. Удвоение числа каналов в A100 по сравнению с V100 дало общую пропускную способность 600 ГБ/с для всего A100 по сравнению с 300 ГБ/с для V100.

В [3] на разных серверах с четырьмя GPU V100 и на сервере DGX A100 с 8 A100 и двумя 64-ядерными процессорами AMD EPYC 7742 проведены измерения пропускной способности передач данных от хоста к устройству или наоборот. Для A100 с PCIe-v4 она составила около 25 ГБ/с (где-то три четверти от теоретической величины 32 ГБ/с для PCIe×16), что соответствует ожиданиям авторов [3]. В этой работе было обнаружено также некоторое уменьшение пропускной способности двунаправленных обменов данных ЦП с «удаленными» A100 по сравнению с «локальными» (имеющими прямую связь по PCIe с соответствующим ЦП), что было связано с конкуренцией за работу с PCIe.

Понятно, что расчеты с multi-GPU серверами для достижения ожидаемо высокой производительности вероятно могут требовать существенной модернизации программ или приложений, работающих с одним GPU (и даже возможно более тонкой оптимизации с учетом некоторой «асимметрии» разных GPU в сервере). Примерами могут быть работы [197, 198]. Причем такая модернизация включает также алгоритмы, и может охватывать и самые базовые вещи – например, GEMM (см. [199] и [200]).

Ясно, что серверы, содержащие один или несколько GPU A100, поставляются всеми ведущими производителями серверов. Серверы multi-GPU ориентированы в основном на задачи ИИ, хотя появляются программные средства для HPC в областях, где GPU вообще не очень часто применялись. Например, в приложении QUICK реализовано распараллеливание на уровне multi-GPU в том числе и для наиболее массово распространенного квантовохимического метода DFT в гауссовском базисе [198].

DGX A100 является ориентированным на ИИ знаменитым «классическим» сервером Nvidia, содержащим 8 GPU A100 и 6 микросхем NVSwitch [201]. Он использует для этого модуль HGX A100, и дает конкретную фиксированную топологию межсоединения с применением SXM4. A100 имеет два варианта с емкостью памяти 40 или 80 ГБ, соответственно имеется два варианта сервера – DGX A100 320GB System и DGX A100 640GB System, с памятью на систему емкостью 1 и 2 ТБ соответственно. В DGX A100 имеется два 64-ядерных процессора EPYC 7742 Roma, которые связаны с A100 через коммутаторы PCI (с шинами PCIe-4.0×16). Для связи между серверами используются адаптеры Nvidia ConnectX-6 или ConnectX-7 (Infiniband HDR /200 Гб/с Ethernet) [202].

DGX A100 является готовой к работе системой, поставляемой вместе с операционной системой (на базе Ubuntu Linux), содержащей специальные средства для управления и мониторинга. Подробности (включая и реализованную топологию используемых межсоединений) см. в [202].

Кроме того, Nvidia возрождает давно не использовавшийся класс рабочих станций – предлагая DGX Station A100 (про разные типы систем DGX см. [203]).

Nvidia пошла еще дальше на пути поставок быстро разворачиваемых вычислительных систем, и создала платформу DGX SuperPOD, базирующуюся на строительных блоках со стойками, содержащими по 5 DGX A100, с поставкой кластерных систем, имеющих от 4 до 28 стоек (см., например, [196]). Такое построение строительных блоков затем было модернизировано [204]. Такие ориентированные на ИИ системы могут быть развернуты всего за несколько недель [196, 204]. Например, в Top500 такой суперкомпьютер Nvidia Selene на базе DGX A100, инсталлированный в 2020 году, занимает 9 место. Очевидно, DGX A100 и более крупные системы ориентированы на упрощение работы в соответствующих ЦОД.

Укажем на еще более мощные из использующих A100 суперкомпьютеры, входящие в первую десятку Top500. Вычислительные узлы итальянского суперкомпьютера Leonardo (инсталлирован в 2022 году), занимающего четвертое место в Top500 [**205**] содержат один 32-ядерный процессор Intel Xeon Platinum 8358/2.6 ГГц (Ice Lake) и 4 GPU A100-SXM4-40GB, а для связи между узлами используется Infiniband HDR200 (продукция Nvidia с топологией Dragonfly+).

Восьмую строчку в Top500 занимает инсталлированный в 2021 году суперкомпьютер Perlmutter Национального центра энергетических исследований США (NERSC, а оператор – знаменитая в суперкомпьютерном мире американская Лоуренсовская лаборатория, LBNL) [**206**]. В основных его вычислительных узлах используется 64-ядерный процессор EPYC 7763/2,45 ГГц и 4 GPU A100 (в большинстве узлов – с 40 GB HBM, и еще в 256 – с 80 GB HBM). В Perlmutter используются известные аппаратные средства построения суперкомпьютеров HPE Cray EX235N и соответствующее межсоединение HPE Slingshot 11. Интересно, что в этих двух последних из вышеупомянутых суперкомпьютерах, ориентированных не только на решение задач ИИ, в узлах применяется по 4 A100.

4.1.3. *Средства SDK от Nvidia для A100*

Общий состав средств SDK. Как и рассмотренные выше аппаратные средства A100, так и средства SDK являются предшественниками соответствующих средств H100. Но SDK являются непрерывно развивающимися, и их различные версии обычно могут работать и для базовых GPU Nvidia, и для GPU нового поколения. После анализа SDK для A100 в отношении SDK для H100 достаточно указать на усовершенствования.

SDK от Nvidia из-за многолетнего однозначного господства Nvidia на рынке GPU стали хорошо известны. Основу для HPC составляют, естественно, средства NVHPC – см. на сайте производителя, [**207**]. Там приведен и полный список входящих в эти программные средства компонент, большинство названий которых (в первую очередь математических библиотек) уже объясняют, зачем они нужны. Возможность свободного доступа к NVHPC является плюсом для Nvidia. На момент написания обзора была доступна HPC SDK Version 23 [**208**].

В NVHPC входят средства с поддержкой платформ x86-64 (AMD и Intel), OpenPower и ARM. Они включают компиляторы соответствующего языка (с поддержкой средств OpenMP и OpenACC для ЦП), ассемблер и редактор связей для целевых процессоров с параметрами из командной строки. Эти компиляторы – nvc (ISO C11), nvc++ (ISO C++17) и nvfortran. Последний (поддерживает Fortran 2003 и ряд функций Fortran 2008) для работы с GPU может использовать функции распараллеливания Fortran, OpenMP и OpenACC.

Наконец, nvcc – драйвер компилятора CUDA C/C++ для GPU Nvidia с применением специализированных для GPU опций создает оптимизированный код для GPU Nvidia и управляет хост-компилятором. Последняя на момент написания обзора версия CUDA была 12.2. nvcc скрывает от разработчика сложные детали компиляции CUDA, и все не относящиеся к CUDA шаги компиляции перенаправляет к хост-компилятору со специальными опциями командной строки [208]. Эффективному развитию nvcc способствует его базирование на средствах знаменитого компилятора LLVM [209]. Далее в разделе 4.2.5 про средства CUDA для H100 описана общая схема построения разных компиляторов Nvidia вплоть до уровня выполнения, включающая и новые ранее не использовавшиеся языки программирования.

Математические библиотеки HPC SDK включают, в частности, cuBLAS, cuSPARSE, cuRAND и cuSOLVER (для решения задач линейной алгебры, в том числе задачи на собственные значения). cuSOLVERmp обеспечивает эти функции при работе с распределенной памятью в многоузловых и multi-GPU системах. Библиотека cuFFT дополняется библиотекой cuFFTmp для выполнения аналогичных cuSOLVERmp расширенных функций.

Низкоуровневая библиотека cuTENSOR предназначена для задач линейной алгебры на тензорных ядрах, и может использоваться не только для задач ИИ, но и в областях HPC. Две очень важные библиотеки обеспечивают коммуникации. Это – NCCL (Nvidia Collective Communications Library), дающая примитивы коммуникаций для многоузловых и multi-GPU систем с GPU Nvidia, и NVSHMEM для PGAS-распараллеливания по стандарту OpenSHMEM. Здесь память может в определенном смысле интегрироваться в кластере с GPU Nvidia в узлах, используя коммуникации с NVLink, PCIe и Infiniband.

NVHPC включает еще ряд средств, в том числе отладчик CUDA-GDB и профилировщик Nsight Compute (для работы с ним можно использовать библиотеку профилирования NVTX) [208].

Следует также отметить, что традиционные средства распараллеливания MPI в кластерах из нескольких узлов, содержащих GPU, также могут, естественно, применяться. Но здесь важным может быть использование такими MPI, которые могут использовать давние аппаратно-программные возможности GPU Nvidia и средств CUDA – GPUDirect, обеспечивающие передачу данных (RDMA) через PCIe минуя обращение в ЦП – напрямую через сетевую плату. Такие возможности давно имеются, например, в MVAPICH2 [210].

Программные средства CUDA. CUDA является самым знаменитым и распространенным API для работы с GPU, обсуждавшимся во многих публикациях. AMD для своих GPU разработала аналогичные средства

HIP [58]. Intel в OneAPI использует DPC++, но общее построение распараллеливания там также использует модель SIMT, первоначально предложенную Nvidia, и в oneAPI применяются термины, аналогичные используемым в CUDA (см. таблицу 1 выше). Аппаратные средства современных GPU тесно взаимосвязаны с SIMT, поэтому здесь также необходимо ультракраткое пояснение основы CUDA (полное описание см. [16]).

Грубо говоря, в модели программирования CUDA пишется последовательный код, который параллельно исполняется многими нитями на GPU, а каждая нить работает со своей частью общих данных, для чего получает свой идентификатор. Хотя работающая на хосте CUDA-программа может использовать не только C, но и C++, выполняемое на устройстве программное ядро использует расширения относительно C.

В CUDA исходная задача разбивается на набор независимых подзадач, которые считаются на собственных блоках нитей. У каждой подзадачи свой блок нитей, и она решается всеми нитями этого блока. Нити могут взаимодействовать между собой только в пределах одного блока. Блок нитей – это массив нитей, который может быть одно-, двух- или трехмерным.

Нити внутри блока нитей используют разделяемую память (сверхбыструю – грубо говоря, на уровне кэша) и взаимодействуют через барьерную синхронизацию (при обращении к этой функции дальнейшая работа невозможна, пока все нити не войдут в нее). Для обмена данными между разными блоками нитей применяется глобальная память.

Блок нитей является центральным объектом для программирования распараллеливания в CUDA (хоть с появлением H100 выше блока нитей в CUDA появился новый уровень, кластер блока нитей, это является уникальным для H100). Верхний уровень иерархии нитей – над блоками нитей – это сетка нитей, – одномерный, двухмерный или трехмерный массив блоков нитей.

У всех блоков нитей совпадают размерности и размер (число нитей в блоке). Размер сетки нитей (число блоков нитей) и размер блока – это встроенные переменные. Любой блок нитей в сетке имеет индекс блока в сетке. Каждая нить имеет индекс, задаваемый одним, двумя или тремя неотрицательными целыми числами.

Для индексаций нитей и блоков программное ядро использует для «наиболее масштабируемого» случая трехмерные целочисленные вектора – встроенные переменные threadIdx и blockIdx

Каждый блок нитей разбивается на варпы, и все нити варпа принадлежат одному блоку. Физически одновременно выполняются только нити в пределах одного варпа. Нити на разных варпах могут находиться на разных стадиях выполнения программы. В CUDA-программе нет

необходимости явно работать на уровне варпов. Во всех рассматриваемых в обзоре GPU Nvidia варпы имеют 32 нити. Работа на уровне варпов может понадобиться только для максимизации достигаемой производительности в CUDA.

У Nvidia CUDA нет ограниченного базирования исключительно на C/C++. Мы проиллюстрируем работу с CUDA на примере CUDA Fortran, привлекательного для остающегося популярным в HPC Fortran (возможность применения Fortran для работы с графическими процессорами Nvidia является существенным плюсом фирмы). Новейшая (к моменту появления июньского списка Top500 2023 года) версия nvfortran была 23.3.

Расширения CUDA Fortran позволяют выполнять в Fortran-программе написание подпрограмм и функций для выполнения на графическом процессоре, включая объявление переменных, выделенных в памяти устройства GPU и выделение динамической памяти в памяти устройства GPU. Естественно, CUDA Fortran имеет поддержку операции копирования данных из памяти хоста в память GPU и обратно, и вызов подпрограмм GPU с хоста. CUDA Fortran обеспечивает программный доступ к тензорным ядрам, взаимодействие с CUDA C, использование асинхронных передач между хостом и графическим процессором (асинхронная передача позволяет проводить вычисления на устройстве одновременно с передачей данных) и много других нужных для современных GPU возможностей [211].

Простейший пример того, как устроена программа на CUDA Fortran из руководства Nvidia, [211] цитируется ниже.

Код на хосте

```
1  program t1
2  use cudafor
3  use mytests
4  integer, parameter:: n = 100
5  integer, allocatable, device:: iarr(:)
6  integer h(n)
7  istat = cudaSetDevice(0)
8  allocate(iarr(n))
9  h = 0; iarr = h
10 call test1<<<1,n>>> (iarr)
11 h = iarr
12 print *,&
13 "Errors: ", count(h.ne.(/ (i,i=1,n) /))
14 deallocate(iarr)
15 end program t1
```

Код на устройстве

```
1  module mytests
2  contains
3  attributes(global)  &
4  subroutine test1( a )
5  integer, device:: a(*)
6  i = threadIdx%x
7  a(i) = i
8  return
9  end subroutine test1
10 end module mytests
```

В коде хоста использование модуля `cudafor` (строка 2) обеспечивает интерфейсы к библиотеке времени выполнения хоста CUDA (в данной программе – к `cudaSetDevice()`, где выбирается номер устройства `0` – это вызов API в строке 7). В строке 3 указывается на использование модуля на устройстве (этот модуль содержит вызываемую подпрограмму

test1). В 8-й строке программы распределяется массив iarr на устройстве, а в следующей строке инициализируются данные и на хосте, и на устройстве [211].

Выполняемое на устройстве программное ядро вызывается в строке 10; <<<1,n>>> здесь означает, что программное ядро выполняется n нитями GPU (в более общем случае n в таком синтаксисе указывает число нитей в блоке, а перед запятой указывается число блоков нитей, которое здесь равно 1). В строке 11 результаты выполнения программного ядра передаются в массив хоста, а в 14-й строке массив iarr на ЦП освобождается.

Что касается правой части фортранного текста, выполняемого на устройстве, там используется префикс attributes(global), что является расширением в языке CUDA Fortran. Атрибут global означает, что соответствующий код выполняется на устройстве, но вызывается из хоста [211].

Шестая и седьмая строки кода для устройства представляют собой замену цикла do в обыкновенном Fortran:

```
1  do i=1,n
2  a(i)=i
3  enddo
```

Поскольку подпрограмма test1 выполняется на GPU, то она параллельно выполняется несколькими нитями на GPU, каждая из которых идентифицируется встроенной переменной ThreadIdx(она используется как индекс элемента массива).

Хотя внешне приведенный выше примитивный текст на CUDA Fortran создает ощущение простоты, реально не просто имеется множество других языковых конструкций, а происходит смена парадигмы – в модулях устройств исчезают обычные циклы, вместо них все базируется на вышеописанных SIMT-основах (применение директив генерации программного CUF-ядра из цикла является способом упрощения программирования, когда компилятор сам генерирует программное ядро для части кода хоста со вложенными циклами).

CUDA Fortran позволяет работать совместно с другими программными средствами. Например, возможно совместное использование OpenACC и CUDA Fortran в одной программе; в программе с OpenMP можно использовать память, управляемую CUDA [211].

По URL вида https://docs.nvidia.com/hpc-sdk/pdf/hpc*Vv*.pdf, где *Vv* – все цифры версии SDK (например, *Vv*=233 для версии 23.3 на момент написания обзора), доступны описания интерфейсов библиотек (это API) к CUDA Fortran.

В завершение краткой информации о CUDA Fortran следует отметить точку зрения сотрудников Nvidia, указанную в их известной книге [212],

что применение CUDA Fortran будет нацеленно в первую очередь на тех программистов, которым важно обеспечение переносимости программных средств, ясность и удобство сопровождения кода при достижении приемлемой производительности. Но легкость написания кода дает и другой важный эффект, увеличение производительности труда (уменьшение сроков разработки программ).

Естественно, программные средства класса SDK, которые могут применяться при работе с GPU Nvidia, имеются и у других разработчиков. Например, HPE Cray MPI [213] с поддержкой CUDA (этот MPI корректно копирует данные из памяти устройства в память сетевой платы (и обратно) путем неявного копирования данных сначала в память хоста, а оттуда в сетевую плату, или напрямую (минуя память хоста) при поддержке GPUDirect RDMA [214], которая имеется в A100. MVAPICH2-GDR [215] также может взаимодействовать с CUDA. Про средства AMD HIP, которые могут работать и на GPU Nvidia, упоминалось выше. Такие не создававшиеся Nvidia средства SDK здесь обсуждаться не будут – но возможно их рассмотрение в разделах обзора про GPU нового поколения.

4.1.4. *Данные о производительности A100*

Хотя обзор нацелен на GPU нового поколения, представляется нужным привести данные о достигаемой производительности A100 и его ускорения относительно V100. Информация о производительности A100, дающая еще и сопоставление с GPU нового поколения, рассматривается далее в соответствующих разделах про эти GPU, и здесь может не приводиться.

Про пиковые производительности говорилось ранее, тут рассматриваются данные тестов производительности и приложений. Надо иметь в виду, что поставки GPU A100 от Nvidia, начавшиеся в 2020 году, сопровождались таким активным появлением новых моделей A100, в том числе в связи с одновременно пробуждавшейся конкуренцией с появляющимися в это время GPU нового поколения от AMD, что авторы некоторых публикаций, где использовались новейшие A100, не всегда однозначно точно указывали параметры использовавшейся модели.

Здесь полезно сделать небольшое введение об эффективном использовании тензорных ядер для умножения плотных матриц и о применении смешанной точности. Для работы с ИИ это достаточно естественно, для обычных работающих с FP64 HPC-приложений, эффективно работающих на GPU, также часто предполагается применение DGEMM. Обсуждение вопросов, насколько актуальна аппаратная поддержка DGEMM для массовых HPC, и можно ли обходиться более низкой смешанной точностью см. в [90].

Для HPC чаще требуется работа DGEMM с большими квадратными матрицами, а тензорное ядро A100 выполняет аппаратные операции над матрицами фиксированных маленьких размеров (см. выше в разделе 4.1.1). Но тензорных ядер в A100 много, а умножение больших матриц сводится к умножению маленьких подматриц. Программисты могут просто использовать *cublasDgemm*, и для большого по сравнению с аппаратно поддерживаемыми тензорными ядрами размера матриц (16384×16384) в [**216**] указывается на достижение близкой к пиковой для тензорных ядер A100 производительности 19,4 TFLOPS. В [**216**] исследовано и энергопотребление, и найдено, что не всегда повышение мощности коррелирует с повышением производительности.

Надо отметить, что в [**216**] использовался размер матрицы, кратный 2^k. Это соответствует указанию в руководстве Nvidia по работе с матрицами [**217**] на тензорных ядрах (ориентированном на задачи ИИ) – там отмечено, что производительность выше при размерностях матриц, кратных 128 байт (для FP64). В [**217**] описано, как GEMM в cuBLAS реализуется путем разбиения выходной матрицы на фрагменты, которые приписываются блокам нитей, и обсуждается выбор оптимального компромисса между размером умножаемых подматриц и требованием задействования всех аппаратных ресурсов GPU путем максимального распараллеливания. Но эти разные предлагаемые варианты GEMM важнее для не очень больших исходных матриц, и результаты в [**217**] демонстрируются для характерной для ИИ низкой точности.

Для графического процессора Nvidia Titan RTX, где DGEMM эмулировался с помощью тензорных ядер из-за отсутствия в них аппаратной поддержки формата FP64 [**218**], график зависимости производительности DGEMM из cuBLAS от размерностей квадратных матриц показывает определенные скачки достигаемой производительности. Для A100 такие скачки производительности GEMM при изменении размерностей матриц при работе с FP16 продемонстрированы в [**217**]. Очевидно, аналогичная зависимость производительности A100 в DGEMM для наиболее актуальных для HPC достаточно больших квадратных матриц также имеет это свойство, но автору демонстрирующие это статьи не известны.

Так что для эффективной работы с HPC на GPU необходимо не только усложненное программирование возможно исходно естественно последовательных кодов, но полезно и знание вероятного поведения производительности на уровне BLAS. В целом формируется ощущение, что вопросы применения тензорных ядер этих GPU Nvidia для традиционных задач HPC в формате FP64 с обычными плотными матрицами предстоит еще изучать – аппаратные средства GPU развивались слишком быстро, с исходной ориентацией вовсе не на HPC.

Ряд исследований посвящен использованию тензорных ядер и просто с точки зрения применимости смешанной точности. Так, в [**219**] на V100 и A100 исследованы различные вопросы такой арифметики, в том числе с точки зрения режимов округления. Работы по программному обеспечению умножения матриц с использованием тензорных ядер и их эффективному применению продолжаются. В [**220**] исследована возможность применения многословной арифметики – когда матрицы представляются в виде невычисленной суммы двух или более матриц более низкой точности, а произведение матриц вычисляется через умножение их составляющих пониженной точности. Это было проделано, например, на тензорных ядрах A100 и V100 для исходных матриц **A** и **B** в формуле (1) с FP16-числами, а накопительных матриц – с числами FP32. Были изучены ошибки округления и предложено применение уменьшающих ошибки округления алгоритмов. Цель этого – улучшение соотношения производительности и точности [**220**].

Для задач ИИ актуальна работа с матрицами, имеющими специфическую разреженность. Данные о такой работе с тензорными ядрами для V100, A100 и H100 имеются в [**221**].

Далее приводится часть информации о производительности A100, которую автор считает наиболее актуальной, особенно для задач HPC. Более высокая производительность A100 относительно V100 очевидна и подтверждена во многих статьях. В первую очередь это связано с поддержкой FP64 в тензорных ядрах, более высокими пиковыми производительностями, большей емкостью кэша L2 и большей пропускной способностью памяти в A100. Но это не означает, что выигрыш будет получен в любом коде (особенно не оптимизированном), при использовании любой версии CUDA, при любых размерностях задач и так далее. Так, в [**222**] на некоторых кодах, приводимых Nvidia в качестве примеров использования CUDA [**223**], V100 опередил A100.

Ну, а в качестве некой стартовой интегральной оценки ускорения A100 относительно V100 воспользуемся данными теста SPEC ACCEL v1.2 (использующего 19 разных приложений)[226], где полученные Lenovo максимальные результаты для A100-PCIe-40GB относительно V100S-PCIe-32GB лучше примерно в 1,7 раза. Другую интегральную оценку производительности A100 дают данные тестов SPEChpc 2021, для получения которых применяются и современные суперкомпьютеры с GPU [**225**]. Однако в официальных результатах этих тестов [**226**] (на 14.06.2023) нет данных для вычислительных систем с одинаковым количеством GPU A100 и V100. Сопоставительные данные о производительности multi-GPU серверов с несколькими A100 представлены ниже в разделе про производительность H100.

Микротесты и низкоуровневые тесты, максимально приближенные к аппаратуре A100. К этой группе тестов условно можно отнести несколько публикаций. В [227] исследовано влияние на производительность применения асинхронного копирования на уровне микротестов и в модернизированном интегральном наборе тестов Rodinia более высокого уровня (для отличных от задач ИИ приложений университета Вирджинии).

В [79] использовались микротесты для измерения задержек и пропускных способностей (производительности) выполнения команд на тензорных ядрах (PTX-уровень). В [228] производительность программных ядер для связанных памятью (т.е. где работа с памятью ограничивает производительность) итерационных методов решения систем линейных уравнений исследована на V100 и A100.

В [229] был создан низкоуровневый синтетический тест, примененный для определения производительности и энергопотребления сервера DGX A100 с восемью GPU A100 и сервера DGX-2 с 16 GPU V100 с использованием динамического масштабирования частоты и напряжения. Для измерения производительности применялся тест производительности с плавающей запятой Mandelbrot [230], реализованный на CUDA PTX, дающий максимальную приближенность к пиковым показателям из-за высокого распараллеливания по данным, практически отсутствия задержек выполнения из-за доступа в память и ветвлений. В этих серверах применяются 64-ядерные EPYC 7742 из-за поддержки в них PCIe-4.0, недоступной у Intel Xeon. При оптимальной по энергоэффективности конфигурации A100 достиг для формата FP64 энергоэффективность на уровне 51 GFLOPS/Вт и 91 GFLOPS/Вт при работе без и с применением тензорных ядер соответственно [229].

Для обеспечения высокой масштабируемости приложений важны коммуникации между GPU в multi-GPU сервере и между узлами кластера из таких серверов. В исследовании [231], ориентированном на рекомендательные модели глубокого обучения (Deep Learning Recommendation Models, DLRM – широко применяемыми интернет-компаниями для предсказаний того, что может понравиться потребителям, когда доступно множество вариантов), получены данные о пропускной способности таких коммуникаций с применением NCCL и MPI для систем, содержащих A100 и V100.

Тесты пропускной способности памяти. Данные о традиционных тестах производительности можно начать с тестов пропускной способности – она очень часто лимитирует производительность и важна для других рассматриваемых далее тестов. Здесь надо иметь в виду, что из-за различных используемых программных моделей для одного и того же теста могут получаться разные численные результаты. Ярким примером этого является *BabelStream* [232], который является «переносом» широко используемого для ЦП теста STREAM (см., например, данные для ARM-процессоров [5]).

В BabelStream не учитывается время передачи данных между хостом и устройством, и он имеет реализации практически на всех используемых для GPU моделях программирования. По сравнению с классическим стандартным тестом STREAM [233] в BabelStream добавлен еще тест на скалярное произведение векторов, и сделаны некоторые другие модификации [232].

В [234, 235] полученные данные о пропускной способности для всех тестов BabelStream для разных размеров массивов показывают обычно существенное ускорение A100 относительно V100. Для массива размером 8,2 ГБ V100 достигает в этих тестах от 800 до 840 ГБ/с, а A100 – от 1,33 до 1,4 ТБ/с. Хотя для массивов небольшого размера A100 имеет меньшую пропускную способность, чем V100, для массивов больших размеров A100 для большинства тестов BabelStream быстрее V100 примерно в 1,7 раза [234]. Надо иметь в виду, что эти данные были получены вскоре после появления A100 и относятся, очевидно, к первой модели A100-40GB, а использовалась тогда версия CUDA 11. Аналогичные данные для пропускной способности во всех тестах BabelStream с использованием разных размеров массивов для A100 и V100 представлены в [173].

В [42] получены данные для пропускной способности (с обычным форматом FP64) всех 5 программных ядер BabelStream: copy (a[i] = b[i]), mul (a[i] = b * c[i]), add (a[i] = b[i] + c[i]), triad (a[i] = b[i] + d * c[i]), dot sum (= sum + a[i] * b[i]) – и для каждого из них на 5 разных моделях программирования, для A100 (на сервере с двумя 64-ядерными EPYC 7H12 + 4 A100-40GB) и для V100 (на сервере с двумя 20-ядерными Xeon Gold 6230 + 4 V100-32GB). В тестах применялось по одному GPU. В качестве единицы отсчета, дающей максимальную пропускную способность A100 и V100, была взята полученная при CUDA-варианте BabelStream.

Наш анализ данных [42] показывает, что во всех тестах из BabelStream с использованием OpenMP и Kokkos на A100 приближение к CUDA-значению в процентах ближе, чем на V100 (за исключением add и triad в Kokkos), а достигаемое приближение к CUDA-показателям выглядит приемлемым.

Во многом тесты BabelStream дают удобную возможность сопоставления эффективности реализации разных программных моделей разных GPU. Например, в [236] сделана Fortran-реализация BabelStream с использованием применимых для GPU средств DO CONCURRENT из Fortran-2008, что является дополнением к другим возможностям работы с GPU на Fortran – с применением средств OpenMP, OpenACC или CUDA Fortran. Сопоставительные данные производительности разных реализаций BabelStream были получены для A100-40GB и A100-80GB. С использованием NVHPC 22.7 для A100-80GB на C++ и Fortran во всех 5 тестах из BabelStream была получена пропускная способность на уровне 1,7–1,8 ТБ/с.

При этом достигнутые величины пропускной способности на OpenMP и CUDA-вариантах C++ и Fortran практически совпадали (максимальная разница около одного процента), за исключением программного ядра dot, где CUDA C++ был на 13% медленнее CUDA Fortran (после изменения параметра настройки BabelStream отставание C++ уменьшилось до одного процента).

В данном обзоре систематическое рассмотрение эффективности различных моделей программирования на GPU не проводится, и эти данные были приведены для иллюстрации поддерживаемого автором данного обзора мнения авторов [**236**] о недостаточной активности развития моделей программирования с применением средств Fortran для гетерогенных архитектур. Существенно более высокие полученные в [**236**] показатели для A100 в BabelStream по сравнению с [**234**, **235**] в первую очередь связаны с применением в [**236**] более современной модели A100 (в A100-80GB пропускная способность памяти выше); в [**237**] тест BabelStream triad дал 1732 ГБ/с для A100-80GB против 1399 ГБ/с для A100-40GB.

Данные о достигаемой пропускной способности памяти для задач ИИ не менее важны, чем для HPC. В [**231**] предложен набор тестов для задач DLRM, который включает специализированный на DLRM тест пропускной способности памяти, где она оценивается при работе с форматами FP32 и FP16. Для A100-40GB была достигнута пропускная способность около 1,4 ТБ/с для FP32, для FP16 – немного ниже. Для V100 с FP32 достигается более 800 МБ/с, а с FP16 – менее 800 МБ/с [**231**].

Учитывая работы со смешанной точностью в тензорных ядрах, соответствующие показатели пропускной способности памяти используются для оптимизации работы активно развиваемой переносимой на GPU разных производителей библиотеки разреженной линейной алгебры Ginkgo [**173**]. В [**238**] имеются данные о производительности A100 на тесте mixbench, в котором применяются программные ядра со смешанной вычислительной интенсивностью (отношением числа выполненных операций с плавающей запятой к числу переданных байтов) для разных форматов данных и с разными программными моделями [**239**], что дает и оценки пропускной способности. Это актуально в основном для ИИ и здесь не рассматривается.

Производительность в линейной алгебре (BLAS). Как и в других тестах производительности для GPU, для задач линейной алгебры появился ряд используемых в GPU особенностей, отличных от традиционного использования на ЦП. Это связано в основном с частым использованием более низких форматов точности и поддержки операций с маленькими матрицами в тензорных ядрах, что в первую очередь актуально для задач ИИ. Это привело к BLAS пониженной точности, а также к созданию пакетной линейной алгебры (решение многих задач линейной алгебры с маленькими независимыми матрицами; размер пакета здесь – число матриц в нем) [**240**].

Соответственно появились специализированные для GPU библиотеки линейной алгебры (здесь в первую очередь имеются ввиду не библиотеки фирм-разработчиков GPU, а библиотеки, ориентированные на работу с GPU разных производителей) – и появляются статьи, где исследуется производительность работы со средствами этих библиотек.

Здесь следует отметить упомянутую выше библиотеку Ginkgo для разреженной линейной алгебры; данные о достигаемой производительности на A100 с ее применением имеются в [**173**, **234**, **235**, **237**, **238**]. Для пакетной линейной алгебры применима известная библиотека MAGMA, модули которой показывали производительность выше соответствующих модулей библиотек производителей GPU; данные о ее производительности при работе на A100 представлены, например, в [**234**] и [**241**]. Для экзамасштабных вычислений появилась библиотека libCEED центра CEED (Center for Efficient Exascale Discretizations) министерства энергетики США, где на тензорных ядрах GPU Nvidia достигается производительность, близкая к пиковым значениям [**242**], но соответствующие количественные показатели в этой работе относятся не к A100, а к V100.

Анализ производительности для задач плотной линейной алгебры естественно начать с GEMM, поскольку только это дает возможность достижения максимальной производительности при работе с тензорными ядрами A100. Но вследствие принципиальной важности GEMM и работы с тензорными ядрами общие особенности этого для A100 были уже проанализированы выше в начале раздела 4.1.4. Здесь рассматриваются публикации, дающие численные оценки производительности GEMM с разными форматами данных.

В [**222**] приведены данные о полученной производительности для умножения матриц разных форматов данных на A100 и V100 для примеров программных ядер CUDA от Nvidia. Здесь не стояла задача оптимизации, использовались нетипичные для традиционных задач HPC небольшие размерности матриц, и это только иллюстрирует тот факт, что можно использовать код умножения матриц, который будет на V100 выполняться быстрее, чем на A100.

В руководстве Nvidia по умножению матриц для задач глубокого обучения [**217**] показано, как сильно возрасла производительность GEMM с FP16 на V100 при переходе от cuBLAS v10 к cuBLAS v11, что иллюстрирует быстрый прогресс даже в стабильных программных средствах Nvidia. В этом руководстве приведены и типичные зависимости производительности A100-SXM4-80GB для таких GEMM от размерностей матриц.

В [**231**] в набор тестов для DLRM включен тест для GEMM, и используются вызовы GemmEx из cuBLAS. Для V100 с FP32 этот тест относится к работе с векторными ядрами FP32, а для смешанной точности

с FP16 – с тензорными ядрами. Для A100-40GB тест с тензорными ядрами дополнительно включает работы с TF32 и BF16. Но приводимые в [231] достигнутые величины производительности относятся к пакетному варианту GEMM (см. рисунок 8; BS на данном рисунке – размер пакета),

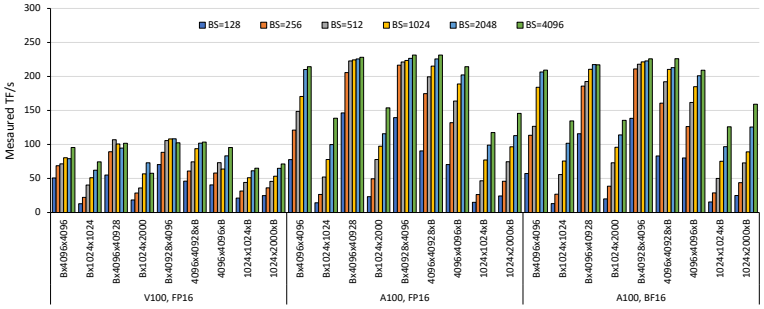


Рисунок 8. Производительность пакетной реализации GEMM – с вызовами GemmEx из cuBLAS (рисунок из [231])

что связано с небольшими (указанными на рисунке) размерами отдельных матриц. На этом рисунке видно, что производительность A100 с FP16/BF16 в разы больше, чем у V100 с FP16, что соответствует основной ориентации современных GPU в первую очередь на задачи ИИ.

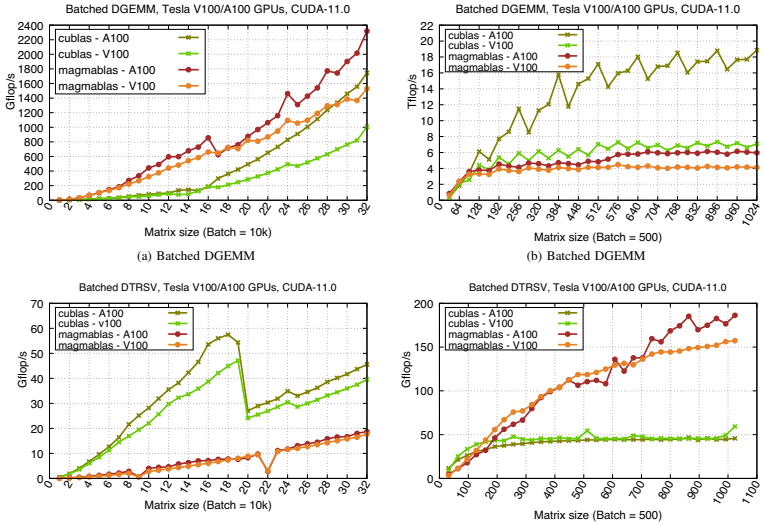
Но производительность пакетных вариантов GEMM исследовалась и для формата FP64. Данные, показывающие существенный рост производительности при выполнении DGEMM в пакетном режиме на GPU, имеются в [242]. Пакетные варианты DGEMM для типовых задач HPC, можно сказать, ранее не использовались – в том смысле, что соответствующие исследования только начали появляться в последние лет десять. Здесь следует отметить решение уравнения гидродинамики сжимаемых жидкостей с конечными элементами высокого порядка [243] (см. также [244, 245]).

В задачах вычислительной химии пакетное DGEMM использовано в известном комплексе программ CP2K для расчетов квантовохимическим методом DFT с периодическими граничными условиями в базисе плоских волн [246]. В [247] отмечена эффективность пакетного умножения матриц для расчета матрицы попарных расстояний, используемой в задачах молекулярной динамики.

Появление в A100 поддержки FP64 тензорными ядрами может существенно усилить актуальность пакетного DGEMM. Но применение этих средств сейчас в большей степени относится к стадии разработки, выяснениям, где это можно использовать, что отображается и в современных работах. Так, в [248] сделан вывод о целесообразности применения пакетного DGEMM в CFD-приложении Nektar++.

В [249] в узлах суперкомпьютера Vega (в Институте информатики, Словения), содержащих по два ЦП (64-ядерных EPYC 7H12) и по четыре A100, проведены расчеты с применением квантового метода Монте-Карло. Хотя время операции в пакетном GEMM было ниже, в предложенном в [249] подходе к расчету методом Монте-Карло с применением MPI-распараллеливания задач GEMM общая производительность всего моделирования оказалось намного выше, чем с применением пакетного DGEMM. Но в этой работе тензорные ядра не использовались.

Поддержка пакетного варианта DGEMM имеется в разных библиотеках для A100. В [234] достигаемая на A100 производительность пакетного DGEMM исследована с использованием средств MAGMA и cuBLAS (CUDA 11.0). Для задач небольшого размера пакетный DGEMM MAGMA достигает 2,4/1,6 TFLOPS для A100/V100. Это на 33/60% быстрее, чем в cuBLAS, что связано со специальной оптимизацией для маленьких матриц в MAGMA. При этом пакетная реализация DGEMM в MAGMA тензорные ядра не использовала (в отличие от cuBLAS). Но на более крупных задачах cuBLAS достигала 18/6 TFLOPS для A100/V100 (подробнее см. рисунок 9, где представлена также производительность другой подпрограммы из BLAS, DTRSV); в целом на пакетном DGEMM A100 быстрее V100 до 1,5 раз в MAGMA и до 3 раз в cuBLAS [234].



(a) Batched DGEMM (b) Batched DGEMM

(c) Batched DTRSV (d) Batched DTRSV

Рисунок 9. Производительность MAGMA и cuBLAS на DGEMM и DTRSV (рисунок из [234])

Видно, что в cuBLAS-варианте при этом достигается хорошее приближение к пиковой тензорной производительности A100 для FP64.

В [234] представлены также данные о производительности A100 и V100 в реализациях MAGMA и cuBLAS для пакетных BLAS DTRSV и факторизации LU и QR (в LU и QR на A100 MAGMA опережала cuBLAS при всех размерностях, а в DTRSV – только на средних).

Для завершения обсуждения здесь производительности в задачах пакетной линейной алгебры на A100 следует указать данные о производительности применяемой QR-факторизации плотных матриц с форматами FP64 и FP32 [241]. Наивысшие данные по производительности с сильным опережением всех альтернатив здесь достигнуты с применением библиотеки MAGMA. Достигнутые асимптотические производительности (больше 2,6 TFLOPS для квадратных матриц FP64) далеки от теоретических пиковых значений потому, что размеры матриц для пакетной GEMM недостаточно велики.

Из других публикаций относительно BLAS на A100 необходимо отметить данные [238] для может даже более распространенного (чем GEMM) в HPC умножения матрицы на вектор (программное ядро GEMV) и для скалярного произведения векторов (программное ядро DOT) – см. рисунки 10,11. Здесь достигаемая производительность для FP64 и

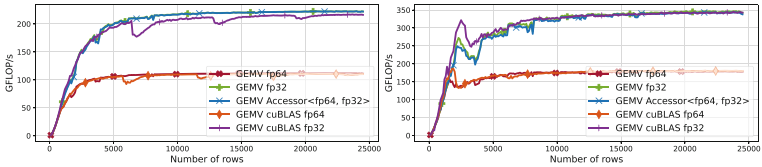


Рисунок 10. Производительность A100 при умножении матрицы на вектор (рисунок – из [238])

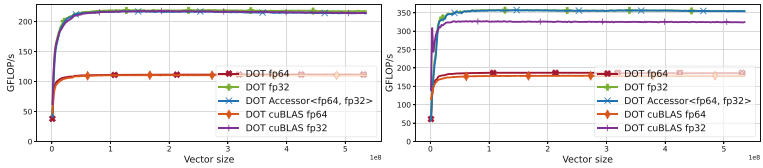


Рисунок 11. Производительность A100 при скалярном произведении векторов (рисунок из [238])

FP32 с применением специальных средств доступа к памяти для разных форматов данных библиотеки Ginkgo (на рисунках помечено как accessor) для A100 и V100 сопоставлена с данными cuBLAS. Здесь видно, как существенно A100 опережает V100 по производительности в разных реализациях BLAS. Аналогичные результаты получены в [238] для другого модуля умножения матрицы на вектор из BLAS, TRSV.

Что касается работы с разреженными матрицами, то для A100 есть ряд публикаций с данными о производительности умножения разреженной матрицы на вектор (SpMV в BLAS), что актуально для разных приложений. В [173, 234, 235] и [238] представлены данные о производительности A100 и V100 для SpMV в формате FP64. Но результат здесь зависит и от выбранного формата представления разреженности, и от конкретных отобранных в тесте матриц, и, естественно, от используемой библиотеки (здесь применялись cuSPARSE и Ginkgo), и на «макроуровне» не так информативен. Отметим лишь максимально достигнутые производительности 220 GFLOPS и 135 GFLOPS на A100 и V100 соответственно [173], что близко к ожидаемым (вероятно, в roofline-моделировании) авторами этой статьи верхним границам 230 GFLOPS и 140 GFLOPS соответственно.

Поскольку выбор оптимального формата хранения разреженных матриц также является нетривиальной задачей, в [250] для ее решения на A100 и V100 было предложено применять машинное обучение.

Быстрое преобразование Фурье (БПФ). Важные данные о производительности A100 дает достигаемая производительность БПФ, актуального для разных задач и HPC, и ИИ. В [251] демонстрируются данные о производительности A100 с применением созданной библиотеки SYCL FFT по сравнению с cuFFT. Хотя SYCL FFT однозначно уступает cuFFT по производительности, цель разработчиков SYCL FFT – переносимость на GPU разных производителей. При этом отставание SYCL FFT во многом связано с большими задержками на запуск (диспетчеризацию) их программных ядер, что для задач большой размерности менее важно.

Другой ориентированной на работу с GPU разных разработчиков библиотекой БПФ является библиотека с открытым исходным кодом VkFFT, данные о производительности которой на A100-40GB имеются в [252]. В VkFFT поддерживаются данные двойной, одиночной и половинной точности, а достигаемая производительность на A100 обычно больше, чем с cuFFT [252].

В [253] разработана библиотека tcFFT для одно- и двухмерного БПФ с использованием тензорных ядер. При работе с FP16 tcFFT превосходит по производительности cuFFT (с CUDA 11.0) в одномерном и двумерном БПФ как на A100, так и на V100 (исключение – одномерный БПФ с небольшими размерностями). В [253] показано и существенное превосходство по производительности в A100 относительно V100 для одно- и двухмерного БПФ; эти данные иллюстрируются на рисунке 12 для двухмерного БПФ [253].

Вычислительная химия:

(A) *Молекулярная динамика.* Задачи молекулярной динамики стали выступать в качестве классических, одними из первых демонстрируемых примеров роста производительности GPU их разработчиками.

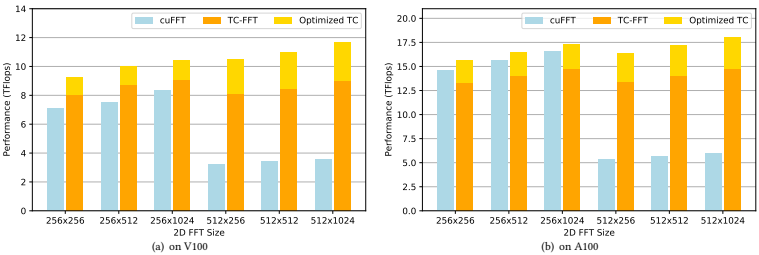


Рисунок 12. Производительность cuFFT и tcFFT на V100 и A100 (рисунок из [253])

Так, в [190] указывается на рост производительности в известных приложениях молекулярной динамики AMBER, GROMACS, NAMD и LAMMPS в 1,5–1,9 раза на A100 по сравнению с V100 (1,9 раза относится к LAMMPS).

Подробным и глубоким исследованием, посвященным производительности LAMMPS в расчетах различных молекулярных систем с применением разных потенциалов и разных GPU, в том числе A100, является [254]. Здесь использовался кластер из multi-GPU серверов (HPE/Cray EX в вычислительной системе Airforce Weather, AFW HPC11). В базировавшихся на A100 узлах использовано по одному 64-ядерному EPYC 7713 и по четыре A100-40GB. LAMMPS для поддержки переносимости кода использует программные средства Kokkos, а статья направлена на достижение оптимального выбора их параметров, и приводятся только данные о соответствующей относительной производительности.

(б) *Молекулярный докинг.* В [42] получены данные о производительности мини-приложения *miniBUDE* [255] на базе ориентированного на молекулярный докинг для задач обнаружения потенциальных лекарств (расчеты комплексов лигандов с протеином) приложения BUDE (Bristol University Docking Engine), в котором изменение свободной энергии Гиббса для связи лиганд-протеин считается предельно просто, эмпирически (то есть метод проще, чем молекулярная механика). BUDE исходно ориентировалось на GPU, а miniBUDE эффективно переносится на самые разные варианты SDK для GPU. В расчетах здесь используется FP32, и A100 примерно на 30% быстрее, чем V100.

Другая программа для молекулярного докинга, AutoDock, в [256] была модернизирована до AutoDock-GPU с обеспечением возможности работы на GPU Nvidia. Там для A100 было найдено достижение более высокой производительности при использовании CUDA, чем при работе с SYCL.

(в) *Квантовая химия.* В [**198**] описана кардинальная модернизация авторами пакета программ с открытым исходным кодом для вычислительной химии *QUICK*, дающая возможность проведения расчетов квантовохимическими методами HF и DFT (наиболее распространенным на сегодня в мире) в гауссовском базисе на multi-GPU серверах, в том числе с A100. Особенностью расчетов всех современных квантовохимических методов в таком базисе является необходимость вычисления $O(N^4)$ интегралов (N-размерность базиса), что не сводится к широко распространенным математическим методам.

Эти вычисления (а также соответствующих градиентов) требуют основного времени расчета методом HF или DFT. Для распараллеливания здесь были использованы средства CUDA C и OpenMPI. Каждый из вышеуказанных интегралов от сгруппированных (contracted) базисных функций рассчитывается одной нитью на PTX-уровне, что позволило достигнуть высокой производительности для такой сложной задачи. Распараллеливание в QUICK возможно от уровня одного GPU до кластера из multi-GPU серверов. Все расчеты выполняются с необходимым для квантовой химии форматом FP64; тензорные ядра не используются из-за отсутствия необходимости умножения матриц в этих методах.

Приведенные результаты времен выполнения расчетов для различных молекул, содержащих от нескольких десятков до нескольких сотен атомов в Infiniband HDR-кластере с узлами DGX (по 4 V100 SXM2 с двумя Xeon Gold 6248 на узел) показали высокую параллельную эффективность и практически линейную масштабируемость производительности от 1 до 16 GPU. На сервере DGX A100 с восемью A100 параллельная эффективность немного ниже, чем с V100 (у V100 меньше SM), но зато время расчета намного ниже [**198**]. QUICK быстро развивается [**257**–**259**], а достигаемое очень высокое ускорение в расчетах по DT относительно обычных двухпроцессорных серверов с x86-64 делает его привлекательным и для достаточно массовых расчетов.

Альтернативный подход к обеспечению высокоэффективных расчетов в гауссовском базисе методом DFT с применением GPU предложен в [**260**] и был реализован на A100. В [**260**] разработаны новые алгоритмы для расчетов кулоновских и обменных вкладов в матрицу одноэлектронного гамильтониана DFT, и создана программа, генерирующая CUDA-коды (программные ядра разные для разных угловых моментов). На молекулярных системах размером от нескольких сотен до более тысячи атомов там было получено достаточно хорошее масштабирование производительности с использованием до 128 GPU A100 на суперкомпьютере Perlmutter.

Другой демонстрацией достигаемости высокого уровня масштабирования распараллеливания в кластере из multi-GPU серверов с A100

на задачах квантовой химии является работа [261], где расчеты были проведены в суперкомпьютерном центре Samsung Electronics (на суперкомпьютере SSC-21, занимающем 20-е место в Top500), в узлах которого применялись серверы HPE с двумя 32-ядерными EPYC 7543/2,8 ГГц, памятью 1 Тбайт и 8 A100-80GB (узлы связаны по Infiniband HDR200).

Здесь по расширенному по сравнению с DFT методу TDDFT (с зависимостью от времени, для расчета возбужденных состояний) с использованием гауссовских базисных функций проведены расчеты молекулы протеина, содержащей свыше 40 тысяч атомов с использованием до 256 A100. В программе для распараллеливания использовались OpenMPI, OpenMP и средства CUDA Fortran (из NVHPC SDK 22.3). Использовался один ранг MPI на каждый узел, там процессоры (они связаны с A100 через PCIe) порождают число нитей OpenMP, равное числу GPU в узле, а для общения между A100 используется NVLink. Полученное ускорение несильно отклоняется от линейного масштабирования вплоть до 256 A100; даже при 256 GPU эффективность распараллеливания превышает 80% [261].

В еще одной работе, проведенной в рамках ECP-проекта [262], представлены времена расчета обменно-корреляционной составляющей одноэлектронного гамильтониана квантовохимического метода DFT с использованием гауссовских базисных функций по новому программному комплексу NWChemEx (разрабатываемому на основе известного программного комплекса NWChem). Соответствующая часть кода была сделана на CUDA. В расчете протеина (убиквитин, более тысячи атомов) исследована зависимость времени вычисления от числа использовавшихся при распараллеливании GPU на суперкомпьютерах Perlmutter (с 4 A100 в каждом узле) и Summit (с 6 V100 в каждом узле). Для оптимизации расчета разработан код на уровне PTX, а для оптимального использования распределенной памяти применялись средства NCCL/NVSHMEM. Время расчета всех компонент программы уменьшается с ростом числа GPU до 32 почти линейно, причем ухудшающие отклонения от линейности при использовании V100 немного больше. Сами эти времена на одном A100 меньше раза в полтора-два, чем на V100.

Вычислительная аэро- и гидродинамика (CFD). Данные о производительности CFD-приложений являются такими же «типичными» в докладах производителей GPU, как и данные по молекулярной динамике, и соответственно публикаций с такими данными для A100 появилось уже много.

Так, в [263] вычисление по программе OpenCFD-SCU на A100 потребовало 3,036 секунды на один шаг расчета по сравнению с 4,702 секундами на V100.

В [264] приводятся данные о производительности в известном коде Nek5000/RS, применяющем специализированную библиотеку OCCA (Open Library Concurrent Compute Abstraction) для разных типов GPU. Программные ядра на A100 там поддерживают 2,1–2,2 TFLOPS (FP64) для оператора Пуассона и 3,1–3,8 TFLOPS (FP64) для оператора адвекции (большие числа в диапазонах относятся к большим размерностям). В предобуславливателе давления прямой оператор Пуассона на более грубых многосеточных уровнях реализует 2,5–3,9 TFLOPS (FP32), а сглаживатель Шварца поддерживает 2,5–5,1 TFLOPS (FP32). Соответствующие показатели на V100 при этом раза в полтора ниже.

Аналогичные данные о более высокой производительности A100 относительно V100 в 1,55 раза при работе NekRS имеются в [265].

В [190] указано на увеличение производительности в 1,7 раза на A100 относительно V100 знаменитого программного комплекса NASA, FUN3D [266]. Понятно, что это требует дополнительной информации с конкретизацией расчета.

FUN3D – это комплекс программ для решения задач CFD с нестуктурированной сеткой, исходно написанный на Fortran, часть текста которого была перенесена на C++ CUDA в виде программных ядер библиотеки FLUDA, причем было создано даже мини-приложение. Расчеты здесь проводятся в формате FP64 для большинства переменных, и со смешанной точностью FP32/FP64 для задач линейной алгебры [267].

На конференции по аэрокосмическим исследованиям 2023 года был целый ряд докладов, где приведены данные по производительности A100 в разных задачах CFD, в том числе по сравнению с V100. В [268] найдено, что три четверти общего времени выполнения FUN3D с применением A100-SXM-40GB на сетке с 3,7 миллионами точек занимали программные ядра, связанные памятью. Достигаемая производительность коррелирует с пропускной способностью, и расчет на A100 в рамках общей исследовательской модели (common research model, CRM) NASA в околозвуковых условиях с применением усредненных по Рейнольдсу уравнений Навье – Стокса был в 1,67 раза быстрее, чем расчет на V100-SXM-16GB.

В [269] в аэродинамическом анализе для электрических самолетов с вертикальным взлетом и посадкой с помощью программы моделирования больших вихрей на системе с восемью A100 время расчета с использованием FP32 было 1,4 часа против 2,9 часа на системе с восемью V100.

В [270] выполнялось моделирование больших вихрей неустойчивости потока на multi-GPU системах с четырьмя V100 и с восемью A100. Данные зависимости производительности от числа GPU для систем на A100 и V100 показывают, что при одинаковом числе GPU (до 4) производительность A100 в разы выше при работе как с FP64, так и с FP32, и достигается хорошее масштабирование производительности до 8 A100.

Еще данные о CFD-производительности серверов с A100 и V100 на этой конференции представлены в [271], а в [272] данные о производительности получены для систем с четырьмя GPU A100 или V100.

В [273] представлены данные о производительности решения CFD-задач решеточным методом Больцмана на A100 и V100. Работа [274] посвящена собственным усовершенствованным реализациям решеточных методов Больцмана с сопоставлением производительности на A100-40GB и V100 с использованием FP32 и FP64 и, естественно, показывает существенный рост производительности A100 по сравнению с V100.

В [275] приведены данные о достигаемой в рамках программы URANOS производительности при использовании A100 или V100. URANOS реализована на Fortran 90 и распараллелена с применением OpenACC и MPI, предназначена для точного моделирования сжимаемых пристенных течений (решения системы уравнений Навье-Стокса в трехмерной декартовой системе от низких до высоких чисел Маха и Рейнольдса). Расчеты проводились на двух разных итальянских суперкомпьютерах с V100 в узлах, и на DGX-A100, где и была получена максимальная производительность. Во всех вычислительных системах максимальная производительность получена при использовании не стандартной реализации MPI, а поддерживающей GPU от Nvidia, в которой используется прямой обмен данными между устройствами, не обращаясь к хосту.

В [276] для задач точного моделирования высокоскоростных потоков в различных схемах решения проведено сопоставление эффективности разных GPU Nvidia, в том числе A100 и V100, с применением не только формата FP64 как основного, но и FP32. Моделирование шума сверхзвуковой струи (13 миллионов ячеек, 400000 итераций) на одном A100 с использованием CUDA дало расчетное время 34,5 часа. Посчитав отношения различных времен расчетов на A100 и V100, приведенных в [276], мы получили диапазон ускорений A100 относительно V100 от 1,4 до 1,9 раз.

Задачи искусственного интеллекта. Что касается производительности A100 для задач ИИ, здесь рассмотрены только два с точки зрения автора более важных источника данных. Во-первых это, конечно, данные о производительности машинного обучения – тестов MLPerf training [277]; в момент написания обзора актуальными были данные для версии 2.1 [278]. Ниже рассматривается более сопоставимый «закрытый» раздел этих тестов. В рамках него возможные данные делятся на доступные в облаке, доступные только локально и предварительные. В таблице 14 приведены локально доступные данные о производительности серверов с A100. По правилам результаты могут предоставляться для отдельных тестов из общего списка. Но здесь целью было максимальное сопоставление, и отдельные

достигнутые максимальные величины производительности по каждому из тестов, полученные разными фирмами, не сопровождавшиеся многими соответствующими данными для других тестов, в таблице не приведены.

Таблица 14. Производительность машинного обучения в тестах MLPerf Training v2.1 [278]. Данные доступны локально (Available on-premise)

Задача	Тип GPU, их число	Время расчета, минут	Отправитель
Классификация изображений	A100-sxm-80gb, 4	54,956	Asustek[1]
	A100-pcie-80gb, 8	30,756	Asustek[2]
	A100-sxm-80gb, 4	54,231	Dell[3]
Сегментация медицинских изображений	A100-sxm-80gb, 4	49,446	Asustek[1]
	A100-pcie-80gb, 8	25,855	Asustek[2]
	A100-sxm-80gb, 4	47,253	Dell[3]
Обнаружение объектов, легковесное	A100-sxm-80gb, 4	161,699	ASUSTek[1]
	A100-PCIe-80GB, 8	89,137	ASUSTek[2]
	A100-SXM-80GB, 4	222,199	Dell[3]
Обнаружение объектов, тяжеловесное	A100-SXM-80GB, 4	78,535	ASUSTek[1]
	A100-PCIe-80GB, 8	43,081	ASUSTek[2]
	A100-SXM-80GB, 4	83,712	Dell[3]
Распознавание речи	A100-SXM-80GB, 4	59,989	ASUSTek[1]
	A100-PCIe-80GB, 8	32,534	ASUSTek[2]
	A100-SXM-80GB, 4	55,086	Dell[3]
NLP	A100-SXM-80GB, 4	33,431	ASUSTek[1]
	A100-PCIe-80GB, 8	24,186	ASUSTek[2]
	A100-SXM-80GB, 4	32,792	Dell[3]
Рекомендация (DLRM)	A100-SXM-80GB, 4	3,147	ASUSTek[1]
	A100-SXM-80GB, 4	4,309	Dell[3]
Обучение с подкреплением	A100-PCIe-80GB, 8	161,647	ASUSTek[2]

[1] с EPYC 7773X, GPU TDP 400 Вт;
[2] с 2×EPYC 7763, GPU TDP 300 Вт;
[3] с 2×EPYC 7763, GPU TDP 500 Вт;

В таблице отобраны только данные, отправитель которых предоставил результаты по 7 тестам из 8 имеющихся в MLPerf training 2.1. Для версии 2.1 данные о производительности V100 отсутствуют, но кардинальное ускорение A100 относительно V100 можно косвенно оценить из данных MLPerf training HPC 2.0 [279]. Полученная производительность тестов может зависеть не только от собственно характеристик GPU, но и от конкретной используемой вычислительной системы и программного обеспечения.

Здесь это больше не обсуждается, но ниже, в разделе 4.2.6, представлены данные о производительности A100 в новых версиях тестов MLPerf – training 3.0 и другого набора тестов для вывода машинного обучения, inference datacenter 3.1. Там производительность A100 сопоставляется с H100 (эти данные стали доступны уже после практического завершения работы над обзором).

Другая имеющая важную информацию о показателях производительности A100 для различных областей ИИ публикация [**231**] в узком смысле ориентирована на DLRM. Было отмечено, что эти модели имеют нетипичные требования по сравнению с другими типами моделей глубокого обучения. В [**231**] с применением A100 и V100 получены данные для разных тестов производительности, в том числе в кластере из серверов GPU, что актуально для больших высокомасштабируемых ЦОД, и сделаны рекомендации по возможным усовершенствованиям таких моделей.

Все приведенные данные производительности естественно и однозначно свидетельствуют о преимуществе A100 относительно V100, в том числе в производительности, что наиболее сильно должно проявляться в HPC-приложениях, требующих умножения матриц FP64 и/или большой емкости памяти. Отставание V100 меньше в HPC-приложениях с FP64, не использующих умножения матриц в тензорных ядрах: векторных ядер FP64 в одном SM у обоих GPU одинаковое число.

4.2. Новые GPU Nvidia H100

4.2.1. *H100 — микроархитектурные реализации Hopper*

Основные показатели, характеризующие H100 в сопоставлении с A100, приведены в таблице 10, а рисунок 6 иллюстрирует общее строение SM в H100, дающее основы для понимания архитектуры H100 и масштабирования производительности с числом SM. Кроме рассматриваемых в данном обзоре моделей H100, Nvidia стала производить GPU H800, также имеющие архитектуру Hopper, и не подпадающие под санкции США в отношении Китая в связи с пониженным до приемлемого уровнем производительности в H800. Эти GPU в обзоре не рассматриваются; информация о них доступна, например, в [**185**].

Все приводимые далее в настоящем разделе данные о микроархитектуре Hopper и ее реализации в H100 основаны на общем описании Nvidia [**78**], а при необходимости более подробной информации приводятся ссылки на соответствующие источники.

Благодаря проведенному выше рассмотрению A100 и архитектуры Ampere, при анализе H100 можно сосредоточиться только на усовершенствованиях в H100 относительно A100, так как общее построение H100 и A100, как и их SM, примерно одинаковое. Общая иерархия построения GPU от уровня SM до уровня полного GPU у H100 и A100 не отличается, и используемая терминология не меняется: из двух SM образуется кластер TPC, а из набора TPC образуются кластеры GPC, которых на GPU всего 8. Но у разных GPU в этой иерархии отличаются количественные показатели.

Как и для A100, у H100 имеются максимально доступные количественные показатели, поддерживаемые архитектурой Hopper (они указываются для GH100). Реально Nvidia предлагает две модели с разными

Таблица 15. Интегральные показатели GH100 и моделей H100

Показатели	GH100	H100-SXM5	H100-PCIe
Число SM	144	132	114
Число TPC	72	66	57
Число GPC	8	8	8
Кэш L2	60 МБ	50 МБ	50 МБ
Число стеков HBM	6 (HBM2E или HBM3)	5 HBM3	5 HBM2E

форм-факторами, реализующие архитектуру Hopper, и их количественные показатели в этой иерархии приведены в таблице 15.

В GH100 в каждом GPC имеется по 9 TPC, но в реальных моделях H100 это не обязано выполняться. В GH100, H100-SXM5 и H100-PCIe используется по 10 контроллеров памяти (по 512 бит каждый), что и дает общую ширину памяти 5120 бит (см. таблицу 10). Самое важное – что модели H100 с разным форм-фактором существенно отличаются по производительности просто из-за разного числа SM, а также из-за разной памяти. Более детально это видно не в таблице 15, а в таблицах 10 и соответственно 12. Пиковую производительность в H100 можно посчитать так же, как это делалось выше для A100 – просто у H100 увеличилось число SM, и в H100 в каждом из 4 разделов SM есть в два раза больше векторных устройств FP64, FP32 и INT32.

По сравнению с A100, кроме количественного роста в H100 числа доступных SM и числа содержащихся в каждом SM векторных ядер, а на общем уровне H100 – емкости кэша L2 и других показателей, целый ряд принципиально важных усовершенствований в аппаратуре H100 кардинально тесно связано с CUDA и будет рассмотрен далее в анализе CUDA-расширений для H100. Это относится и к асинхронному выполнению, и к новому модулю – тензорному ускорителю памяти (Tensor Memory Accelerator, TMA) для эффективной передачи больших блоков данных между глобальной и разделяемой памятью (см. рисунок 6).

Наконец, в H100 было создано или усовершенствовано по сравнению с A100 очень большое количество новых аппаратных средств, которые не относятся к тематике данного обзора в узком смысле его ориентации на HPC и ограниченности рассмотрения задач ИИ. Даже перечисление новых таких средств может занять не один абзац – это многочисленные усовершенствования MIG-технологии (особенно актуально для работы с облачной технологией), средств безопасности и так далее. В ISA были добавлены новые команды, например, для актуальных задач геномики – это имеет весьма важное, но узкое значение. Куча добавленного относится к обработке видео и изображений, задачам ИИ. Из последних отметим только появление аппаратных средств для моделей-трансформеров, применяемых для NLP. Как и при рассмотрении A100 выше, аппаратные средства межсоединений рассмотрены в отдельном разделе, посвященном серверам и вычислительным системам с H100.

4.2.2. *Вычислительные системы с H100*

В данном разделе рассматриваются использующие серверные процессоры x86-64 вычислительные системы Nvidia (от серверов до суперкомпьютеров) на базе H100, а также межсоединения H100 и ARM-процессоры Grace от Nvidia, которые будут использоваться в серверах с H100.

Сеть NVLink для H100. Для масштабирования производительности с ростом числа используемых GPU в серверах (который стал современной характерной тенденцией) и быстрого обмена данными GPU с ЦП (и с другими GPU) межсоединения кардинально важны. С точки зрения автора, здесь Nvidia удалось добиться очень яркого прорыва вперед (см. таблицу 16).

Таблица 16. Технические характеристики межсоединений GPU Nvidia согласно [78, 280]

Технические характеристики	V100	A100	H100
Число линий (дифференциальных пар) на порт NVLink	8	4	2
Пропускная способность канала NVLink одно /двунаправленная, ГБ/с	25/50	25/50	25/50
Число каналов NVLink в GPU	6	12	18
Общая пропускная способность каналов NVLink, ГБ/с	300	600	900
Полная пропускная способность коммутатора, ТБ/с	2,4	4,8	7,2

NVLink4 в H100 как высокоскоростное межсоединение с низкой задержкой и поддержкой функций отказоустойчивости обеспечивает пропускную способность 900 ГБ/с – в 1,5 раза больше, чем NVLink3 [78]. Важно, что такая пропускная способность идеально сочетается с другими пропускными способностями новых суперчипов Nvidia, использующих ARM-архитектуру, которые будут рассмотрены далее.

NVLink4 использует две дифференциальные пары в каждом направлении (их в 2 раза меньше, чем было в канале NVLink3 с той же пропускной способностью) для формирования единого канала с эффективной пропускной способностью 25 ГБ/с в каждом направлении. Поэтому H100 теперь имеет 18 каналов NVLink4 против 12 каналов в A100 [78].

Но еще важнее то, что NVLink4 теперь поддерживает сеть NVLink, обеспечивающую возможность безопасной связи между собой до 256 H100 в разных узлах (серверах) с использованием топологии толстого дерева. Соответствующий кластер имеет 32 узла по 8 H100 в каждом. В сети NVLink имеется сетевое адресное пространство и аппаратные средства преобразования адресов в H100 с обеспечением изоляции сетевого адресного пространства и отдельных адресных пространств разных H100. Ранее в NVLink все GPU использовали общее адресное пространство [78].

Коммутатор NVSwitch исходно давал возможность объединения памяти нескольких GPU. Используемый для работы с H100 коммутатор NVSwitch3 обеспечивает аппаратное ускорение коллективных операций с многоадресной передачей, что для коллективов с небольшим размером блоков до двух раз увеличивает пропускную способность и уменьшает задержку по сравнению с NCCL на A100. Это существенно снижает нагрузку на SM при коллективных коммуникациях [78].

Использование сети NVLink и NVSwitch3 позволяет создавать крупномасштабные сети NVLink Switch System с очень высоким уровнем пропускной способности. Узлы в такой сети соединяются через второй («внешний» по отношению к узлам) уровень коммутаторов NVSwitch, которые находятся в модулях коммутатора за пределами узлов и соединяют несколько узлов вместе. Подключенные узлы способны предоставить 57,6 ТБ/с пропускной способности всех-со-всеми (all-to-all) [78].

Результатом этого становится построение содержащих H100 NUMA-систем с очень высоким достигаемым уровнем масштабирования объема памяти, что отвечает тенденциям современных больших обучающих моделей ИИ.

Серверы и кластеры с H100. GPU H100 могут размещаться в сервере с применением соответствующих модулей Nvidia. Nvidia поставляет модули HGX (грубо говоря – аналог платы графического процессора), которые содержат в себе H100 и могут использоваться другими фирмами для создания сервера, содержащего H100. HGX H100 – это блок, содержащий 4 или 8 H100. Конфигурация HGX с четырьмя H100 имеет полностью взаимосвязанные двухточечные соединения, а в конфигурации с восемью H100 полная пропускная способность между H100 обеспечивается через коммутатор NVSwitch.

Еще один тип модуля с H100 от Nvidia – это H100 CNX, где кроме H100 имеются возможности сетевой платы Nvidia ConnectX-7 SmartNIC, которая обеспечивает пропускную способность 400 Гб/с в варианте Infiniband NDR400 или Ethernet 400 [78]. Понятно, что модули HGX в первую очередь ориентированы на задачи ИИ, а CNX – на HPC.

Конфигурация сервера в части, относящейся к H100, определяется этими модулями. Поэтому здесь мы отметим только уже поставлявшиеся для применения в составе суперкомпьютеров серверы DGX H100, ориентированные в первую очередь на задачи ИИ (хотя о своих серверах с H100-PCIe и HGX уже объявили разные фирмы, например, Supermicro и Gigabyte).

Система DGX H100 с 8 H100 содержит два независимых блока обработки данных (DPU, data processing unit) Nvidia Bluefild-3 и 8 адаптеров ConnectX-7 [78]. DGX H100, кроме практически полной готовности к решению задач ИИ, идеально подходит для работы с ИИ в облачной технологии, поддержка которой начинается с уровня одного H100.

В кластере DGX H100 SuperPOD (минимальная конфигурация – 32 узла DGX H100) по сравнению с SuperPOD A100 вместо двух уровней коммутаторов Infiniband используется один уровень коммутаторов NVSwitch3, а максимальная длина кабеля от коммутатора к коммутатору увеличена с 5 метров для DGX A100 до 8 метров для DGX H100.

В июньской версии Top500 2023 года представлено 5 суперкомпьютеров, использующих H100-PCIe в серверах на базе x86-64. Наивысшая производительность среди этих суперкомпьютеров (14-е место в Top500) достигнута в предшественнике анонсированного Nvidia суперкомпьютера EOS [**281**]. В этом кластере используется 128 узлов DGX SuperPOD. Серверы DGX H100 работают с Ubuntu 22.04 и содержат 56-ядерный Xeon Platinum 8480/2 ГГц (Intel Sapphire Rapids, четвертое поколение масштабируемых ЦП Xeon) и NVidia ConnectX-7 для Infiniband NDR.

Другие попавшие в Top500 суперкомпьютеры с H100 находятся во второй сотне списка Top500 и далее. Там используются и серверы других фирм, и другие серверные ЦП, в том числе AMD EPYC. Из этих суперкомпьютеров следует отметить занимающий 255-е место Henri, который одновременно возглавляет аналогичный июньский список Green500. В Henri используется Infiniband HDR, а в узлах – 32-ядерные процессоры Xeon Platinum 8362/2,8 ГГц (Intel Ice Lake). Необходимо отметить также ожидаемый суперкомпьютер Kestrel [**282**].

Уже сам факт применения DGX H100 в суперкомпьютере позволяет предположить вероятную его ориентацию на задачи ИИ и работу с облачными технологиями.

4.2.3. *Аппаратные средства Nvidia ARM для работы с H100*

Применение ARM-процессоров в высокопроизводительных серверах с GPU H100 должно стать яркой реальной иллюстрацией успехов ARM в конкуренции с традиционными серверными процессорами x86-64, которые использовались в поставляемых во время написания обзора серверах Nvidia с H100. Но применение ARM в серверах с GPU Nvidia началось ранее: многоядерные ARM-процессоры уже используются в серверах с A100 (например, от Gigabyte [**283**]; теперь эти серверы поставляются в Россию и Уругвай, а Gigabyte уже предлагает серверы с H100).

В 2023 году начались пробные поставки Nvidia модулей, содержащих ARM-процессоры для работы с H100. Информация по архитектуре и производительности ЦП ARM от Nvidia (Grace) для работы с H100 пока весьма ограничена; имевшиеся оценки производительности Grace от Nvidia на уровне тестов и приложений относились скорее к ожиданиям.

Поэтому рассмотрение аппаратных средств ARM от Nvidia (модуля суперчипа Grace, пробные поставки которого начались, и суперчипа Grace Hopper, где суперчип Grace интегрирован с H100) здесь очень краткое.

Суперчипы ARM Grace. Суперчип Grace являтся по сути реализацией SoC, содержащей два ARM-процессора и память, с поддержкой PCIe. Как отмечено в [13], он создан специально для суперкомпьютеров и HPC.

Анализируемые далее данные о Grace базируются на [284], а дополнительные ссылки приводятся только для информации из других источников. Основные технические характеристики Grace собраны в таблице 17.

Таблица 17. Основные технические характеристики суперчипа Grace

Архитектура ядер	Neoverse V2 (ARM 9.0-A) с 4×SVE2(128 бит)
Число ARM-ядер	144 (2 процессора Grace)
Пиковая производительность	7.1 TFLOPS (FP64)
Кэш L1	64 КБ I-кэш + 64 КБ D-кэш (на ядро)
Кэш L2	1 МБ (на ядро)
Кэш L3	2×117 МБ
Технология памяти	LPDDR5X (с ECC)
Пропускная способность памяти	До 1 ТБ/с
Емкость памяти	240/480/960 ГБ
Пропускная способность NVLink-C2C (двунаправленная)	900 ГБ/с
Каналы PCIe	8×PCIe-v5 ×16 с возможностью разделения
Их суммарная пропускная способность	1 ТБ/с
TDP	500 Вт (с памятью)

Источники: [284, 285]

Первое, что надо отметить – базирование Grace на новейших ARM-ядрах для HPC, Neoverse V2 [286], которые были анонсированы осенью 2022 года. Это 64-разрядные ARM-ядра с OoO-выполнением команд, поддерживающие SVE2-команды с векторами длиной 128 бит. Суперчип Grace состоит из двух 72-ядерных процессоров Grace, связанных поддерживающим когерентность памяти каналом NVLink-C2C с пропускной способностью 900 ГБ/с, что устраняет взаимодействие между сокетами обычных серверов как узкое место [13]. При этом пиковая производительность (FP64) суперчипа, 7,1 TFLOPS, не сильно ниже, чем у A100 без применения тензорных ядер.

Надо отметить, что в современных ядрах Neoverse V2, ориентированных на ЦП с большим числом ядер, I-кэш L1 контролируется четностью, а D-кэш L1 защищен кодами ECC. D-кэш L2 ядер ARM Neoverse V2 тоже защищен кодами ECC и может иметь емкость 1 или 2 МБ [286]; в Grace Nvidia выбрала вариант с 1 МБ. Кэш L3, естественно, также защищен кодами ECC; другие подробности микроархитектуры Neoverse V2 см. в [286].

Интересной и очень важной разработанной Nvidia особенностью суперчипа Grace является масштабируемая когерентная фабрика SCF (Scalable Coherency Fabric), имеющая сеточную структуру, по которой

распределены ядра и разделы кэша L3 (SCC на рисунке 13), и обеспечивающая поток данных между ядрами, NVLink-C2C, памятью и системным вводом-выводом с суммарной половинной пропускной способностью более 3,2 ТБ/с. SCF создан для масштабирования за пределы одного ЦП

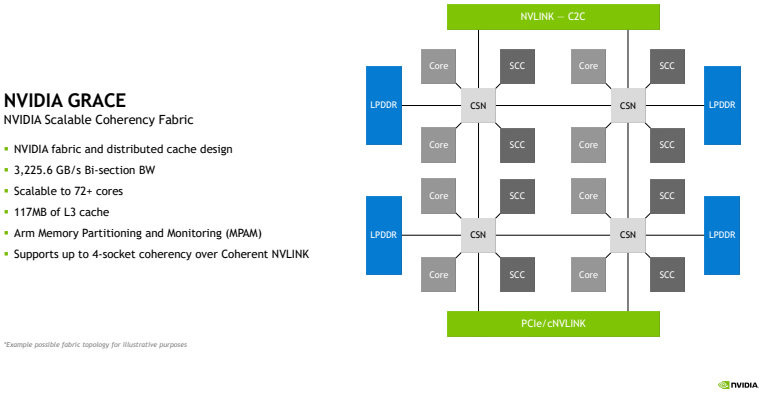


Рисунок 13. Общая микроархитектура Grace (рисунок из [13])

Grace, образуя суперчип со 144 ядрами. Ядра и разделы кэша L3 (SCC на рисунке 13) распределены по сетке, а узлы коммутации кэша CSN (Cache Switch Nodes) выполняют функции интерфейса между ядрами ЦП, кэшем и остальной частью системы [13].

Суперчип Grace имеет также специальные средства, поддерживающие парционирование аппаратных ресурсов, что может быть эффективно использовано при работе в облачной среде [284].

Разработчики Nvidia объясняют свой выбор 32-канальной технологии памяти LPDDR5X как «золотой середины» между HBM2E и DDR5 по параметрам емкости, пропускной способности, стоимости и энергопотреблению [13]. Максимальная теоретическая пропускная способность LPDDR5X в суперчипе Grace (1 ТБ/с) немного выше пропускной способности NVLink-C2C (см. таблицу 17).

Нужно отметить также поддержку в Grace четырех каналов PCIe-v5 ×16, (кроме того, для задач управления имеется еще 2 низкоскоростных канала PCIe-v5×2) [13]. Это сразу выдвигает этот суперчип в лидеры среди ЦП по производительности ввода-вывода.

Но TDP у суперчипа Grace (это двухпроцессорная SoC) сопоставима с TDP современных GPU; это больше, чем у A100 и H100 PCIe (но ниже, чем у H100 SXM). Для работы с суперчипом Grace может применяться воздушное или жидкостное охлаждение [284].

В [284] приводятся также оценки некоторых тестов производительности Grace, ожидаемые Nvidia. Они относятся к данным теста STREAM

(до 400 ГБ/с для одного ЦП Grace и до 800 ГБ/с для всего суперчипа), SPECrate2017_int_base (370 и 740 для одного ЦП Grace и суперчипа соответственно) и существенным величинам ускорений в некоторых приложениях (включая области CFD и предсказания погоды) относительно двухпроцессорного сервера с AMD EPYC 7763 (Milan). Появились и другие первоначальные данные, но их пока совершенно недостаточно.

Nvidia говорит при этом о двукратном повышении энергоэффективности относительно традиционных используемых в ЦОД ЦП [284] – это может стать главным плюсом Grace. О планах создания суперкомпьютеров на базе Grace объявили Лос-Аламосская национальная лаборатория США и Швейцарский национальный вычислительный центр [14].

4.2.4. *Суперчипы Grace Hopper.*

Кроме Grace, Nvidia анонсировала и Grace Hopper, в которых Grace (два 72-ядерных процессора) интегрирован вместе с H100. Основные базовые показатели этого суперчипа примерно являются суммой показателей рассмотренных выше H100 и Grace, и зависят от того, какой из вариантов каждой этих двух интегрированных компонент применяется в Grace Hopper.

В версии 1.01 документа [287], на которой базируется далее анализ суперчипов Grace Hopper, указано на емкость LPDDR5X для Grace в суперчипе Grace Hopper до 512 ГБ, на применение в H100-компоненте суперчипа памяти HBM3 емкостью до 96 ГБ, и на наличие двух вариантов суперчипа Grace Hopper – с поддержкой работы с NVLink4 (см. рисунок 14) и без нее (тогда системы с такими суперчипами соединяются через Infiniband NDR). Понятно, что интеграция двух ЦП, H100 и памяти дает высокий TDP, в [287] указано 1000 Вт.

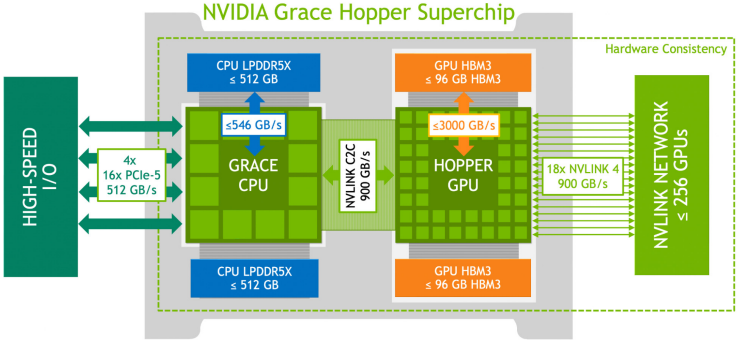


Рисунок 14. Grace Hopper с поддержкой работы с NVLink4 (рисунок из [287])

Уже после завершения работы над данным разделом обзора на этом же URL [287] Nvidia разместила несколько новых версий данного документа (последняя известная автору версия – 1.11), где рассматриваемые суперчипы называются теперь GH200 Grace Hopper. Самые главные изменения там связаны с появлением новых моделей GH200, в которых вместо HBM3 будет использоваться HBM3E емкостью 141 ГБ с пропускной способностью 4,8 ТБ/с (против 4 ТБ/с при работе с HBM3). Основные характеристики архитектуры GH200 Grace Hopper, естественно, не поменялись. Но поскольку из-за быстрого развития GH200 и соответствующих вычислительных систем в 2023 году в более поздних версиях этого документа появлялись новые модификации, и модернизировались иногда даже используемые названия, анализ GH200 в данном обзоре проведен в ограниченном объеме и далее основан на версии 1.01 документа.

В качестве межсоединения процессора Grace и H100 используется NVLink-C2C, обеспечивающий высокую пропускную способность и низкую задержку (см. Рисунок 15) [287]. NVLink-C2C обеспечивает

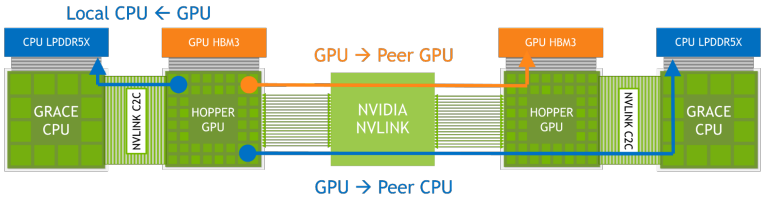


Рисунок 15. Общая микроархитектура Grace Hopper (рисунок из [287])

когерентность памяти и аппаратную поддержку атомарных операций на общесистемном уровне, что позволяет улучшить производительность примитивов синхронизации. Когерентность памяти позволяет разработчикам переносить из Grace в H100 и обратно только нужные данные (а не целые страницы памяти) и, естественно, упрощает программирование гетерогенных приложений. Повышается также производительность доступа к нелокальной памяти (например, когда нити ЦП и H100 обращаются к памяти, расположенной на другом устройстве) [287].

Ключевой особенностью суперчипа GH200 является применение расширенной памяти графического процессора (Extended GPU Memory, EGM). Каждый GPU Hopper из вычислительной системы с сетью NVLink на базе GH200 имеет доступ и к LPDDR5X-памяти всех процессоров Grace, и к HBM-памяти всех GPU, т.е. общий объем памяти для GPU кардинально расширяется. Образуется унифицированная память с общими таблицами

страниц, общее виртуальное адресное пространство. Скорость доступа H100 к локальной памяти LPDDR5X задается NVLink-C2C, а к любой удаленной памяти – NVlink4 [287] (см. рисунки 14 и 15).

Серверы с GH200 уже предлагаются, например, Gigabyte [288]. Nvidia предложила вычислительные системы с GH200 – платформы MGX GH200 и DGX GH200. Модульная платформа MGX GH200 (есть варианты с HBM3 или с HBM3e) максимально похожа на сервер с одним суперчипом GH200, без поддержки системы коммутации NVLink. Из узлов с MGX GH200 можно образовывать традиционные кластеры, где для связи можно использовать Infiniband или Ethernet, работая через DPU.

DGX GH200 является суперкомпьютером для ИИ, поскольку там объединяются 256 суперчипов с применением межсоединения NVLink, давая при этом адресное пространство памяти до 144 ТБ. Краткая информация об этих предназначенных в первую очередь для задач ИИ платформах Nvidia представлена в версии 1.11 документа [287].

С точки зрения возможности построения самых мощных суперкомпьютеров основу для сверхвысокого уровня масштабирования производительности дает применение GH200 с каналами NVLink4 совместно с NVSwitch3; это обеспечивает оригинальное двухуровневое масштабирование Nvidia. Его возможности иллюстрирует рисунок 16, который относится к работе с вариантом GH200 с поддержкой NVLink [287].

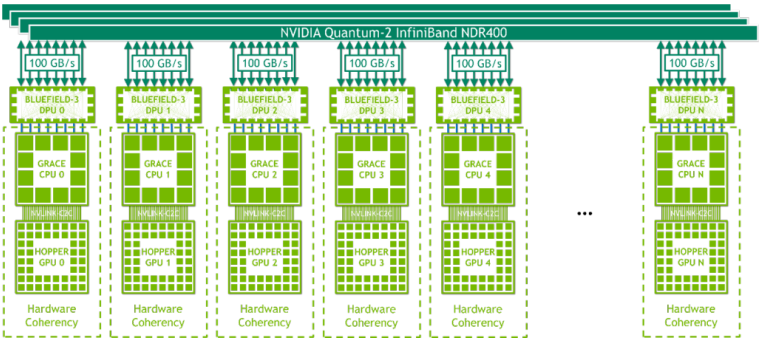


Рисунок 16. Общее построение межсоединений GH200 с поддержкой NVLink (рисунок из [287])

На первом уровне можно объединить набор суперчипов GH200 в единый домен, связанный через коммутаторную систему NVSwitch3 (с общей пропускной способностью всех нитей GPU в домене до 900 ГБ/с на суперчип). Применение этих межсоединений позволяет образовать общую память всего домена, включая как LPDDR5X-память процессоров

Grace, так и HBM3-память H100 [287]. В конце 2023 года Nvidia объявила о другой вычислительной системе (GH200 NLV32), размещающейся в одной стойке – с гораздо меньшим доменом, чем в максимальной конфигурации DGX GH200. Уже на этом уровне обеспечивается очень высокий уровень производительности; Nvidia также называет эту систему суперкомпьютером (см., например, [289]).

На втором уровне возможно объединение в кластер из доменов с применением самых современных широко распространенных межсоединений. Для этого используются поддерживаемые в Grace каналы PCIe-v5. На рисунке 16 представлен возможный вариант такого образования кластера из доменов с применением DPU Bluefild-3. Для связи с Grace в этом DPU имеются 32 линии PCIe-v5, что имеет суммарную двунаправленную пропускную способность 256 ГБ/с (этот DPU имеет еще 32 ГБ собственной памяти). Для образования кластера из доменов DPU имеет еще 1 или 2 порта с пропускной способностью до 400 Гбит/с (они могут работать с Infiniband NDR400 или с Ethernet 400) [290], что и показано на рисунке 16. Применение DPU снимает с Grace необходимость управлять передачей данных по межсоединению кластера.

Понятно, что масштабирование на такой системе будет сложной задачей даже для ИИ. И автору неизвестно, даже посредством анонса, о возможности такого «варианта» и уровня масштабирования систем с GPU от других фирм. В целом ясно, что Nvidia с GH200 ориентируется на поставки готовых вычислительных систем.

4.2.5. *Расширения CUDA для H100*

Все изложенное ниже в данном разделе базируется на описании аппаратуры H100 [78]. Но сначала здесь целесообразно указать на общее построение платформы CUDA в ее части, относящейся к компиляторам, поскольку с течением времени появляются и новые языки программирования, которые также могут использоваться для получения программных ядер, выполняемых на GPU Nvidia. При этом платформа CUDA предоставляет разным компиляторам унифицированный программный стек, и создание компилятора для нового языка программирования в основном сводится к написанию только фронт-энд части компилятора.

Эти компиляторы базируются на LLVM (в [78] указано на использование его 7-й версии). Кроме используемых для работы на GPU возможностей ISO-стандартов C++ и Fortran, а также их CUDA-расширений, еще в 2021 году в CUDA 11.4 у Nvidia появился CUDA-Python [291], а имеются также разработки для Julia и других языков программирования.

Соответствующие фронт-энд компоненты этих компиляторов генерируют промежуточное представление (IR), именуемое NVVM IR [292], которое базируется на знаменитом LLVM IR. Далее с использованием средств библиотеки libNVVM генерируется PTX-код, из виртуальной ISA которого и получается программное ядро, выполняемое на GPU [78].

Важнейшим для достижения высокой производительности работы GPGPU является обеспечение локальности данных, дающей высокоскоростной доступ к близлежащим уровням памяти, и асинхронность выполнения, когда собственно вычисления производятся независимо (одновременно) с передачей данных. Ранее (и для A100) модель программирования CUDA включала блоки нитей, из которых и образовывалась сетка нитей. При этом локальность обеспечивалась тем, что блок нитей выполнялся на одном SM. В H100 число SM существенно возросло, и в общую иерархию нитей добавлен еще один уровень, кластер блоков нитей, который включает набор блоков нитей, охватывающих несколько SM. Все это отображается в H100 аппаратной реализацией с новым уровнем локализации памяти более крупного типа, поскольку кластеры блоков нитей в H100 работают одновременно на SM внутри GPC с обеспечением быстрого обмена данными между нитями в кластере.

Нити кластера могут напрямую обращаться к разделяемой памяти других SM с помощью операций загрузки, сохранения и атомарных операций (которые не могут быть частично выполнены). Для этого образуется распределенная разделяемая память (DSMEM, Distributed Shared Memory). На рисунке 17 показано, как происходит обмен данными между блоками нитей в A100 и в H100 с DSMEM. Кластер блоков нитей имеет доступ к большей, чем у одного блока нитей, емкости разделяемой памяти — к виртуальному алресному пространству разделяемой памяти. По сравнению с применением глобальной памяти DSMEM ускоряет обмен данными между блоками нитей примерно в 7 раз [78].

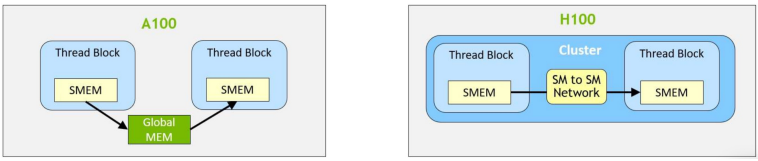


Рисунок 17. Работа с DSMEM в A100 и H100 (рисунок из [78])

С другой стороны, с точки зрения автора появление кластера блоков нитей на новой модели GPU является яркой демонстрацией того, что широко используемые модели программирования для GPU, требующие максимально возможного достижения производительности, автоматически становятся плохо переносимыми на другие модели GPU.

Другие яркие расширения в CUDA для H100 относятся к асинхронному выполнению, обеспечивающему перекрытие (одновременное выполнение) перемещения данных, вычисления и синхронизации [78]. Для этого в H100 появился новый модуль асинхронного копирования памяти TMA и новый барьер асинхронных транзакций. TMA может передавать большие (вплоть до емкости разделяемой памяти) блоки данных и многомерные тензоры из глобальной памяти в разделяемую память и обратно.

Операция TMA является асинхронной и использует асинхронные барьеры на основе разделяемой памяти, как в A100. Кроме того, одна нить в варпе может выбираться для выполнения асинхронной операции, а потом несколько нитей с помощью барьера могут ожидать завершения передачи данных. TMA освобождает нити для выполнения другой независимой работы, так как после создания нитью так называемого дескриптора копии перед запуском TMA все остальные функции TMA сам выполняет аппаратно.

Асинхронные барьеры впервые появились в A100, а в H100 появился новый примитив для асинхронных копий памяти, барьер асинхронных транзакций. Команда записи в разделяемую память передает не только данные, но и число транзакций. Барьер асинхронных транзакций блокирует нити по команде ожидания до тех пор, пока все нити-производители не выполнят операцию «Прибытие», и сумма всех счетчиков транзакций не достигнет ожидаемого значения. Подробнее см. в [78]. Эти возможности используются, естественно, для работы с кластерами блоков нитей.

4.2.6. *Первоначальные данные о производительности H100*

Данные по производительности H100 в тестах и на приложениях HPC на момент возникновения июньского списка Top500 2023 года практически отсутствовали, если не считать стартовые данные в докладе на Hot Chips 34 [293]. Следует также указать на доступные (на 14.06.2023) данные крохотного варианта (tiny suite) теста SPEChpc 2021 [226], где для сервера Lenovo ThinkSystem SR655 V3 представлены результаты с одним и двумя H100-PCIe-80GB (с EPYC 9654P). Мы сравнили эти данные с результатами SPEChpc 2021 от Lenovo для одного и двух A100-PCIe-80GB на другом сервере ThinkSystem SR670 V2 (с Xeon Platinum 8380). Для базового варианта теста (пиковый вариант для A100 не проводился) полученная производительность на одном H100 больше, чем у A100 в 1,33 раза, а на двух H100 аналогичное ускорение было в 1,51 раза. Но надо иметь в виду, что для распараллеливания здесь применялись средства OpenACC (плюс MPI).

Другие интересные данные о производительности H100 относятся к известному приложению молекулярной динамики Amber версии 22 [294]. Там для типовых для этого приложения тестовых молекулярных систем проведены расчеты, дающие оценки производительности (число промоделированных наносекунд за день расчета) на разных графических процессорах. Рассчитанные нами из этих величин относительные производительности показывают ускорение H100 относительно A100 в 1,11–1,28 раза для дигидрофолатредуктазы (23558 атомов), в 1,27 раз – для нуклеосомы (25095 атомов), в 1,42–1,43 раза для протеина из 90906 атомов, в 1,40 раза для целлюлозы с большим числом макромолекул (408609 атомов), в 1,35 раза для спутника вируса табачной мозаики (1067095 атомов).

В расчетах более маленьких молекулярных систем ускорения, полученные на H100, бывали и больше, чем указанные выше, но мы эти данные здесь не приводим. Но полезно отметить, что для всех рассчитывавшихся в [294] молекулярных систем близкую производительность к A100 имеет Nvidia RTX 3090, а к H100 – Nvidia RTX 4090.

Есть еще другие данные о производительности Amber22 на H100 в сравнении с производительностью на A100, но они рассмотрены ниже в разделе 5.3.2 в сопоставлении с производительностью на MI250.

Но в первую очередь GPU нового поколения обычно обращены на решения задач ИИ, поэтому первоначальные данные о производительности появляются именно здесь. Для H100 сначала стали доступны предварительные результаты Nvidia по одному из ставших классическим для ИИ (для машинного обучения) набору тестов MLCommon – MLPerf Training в версии 2.1 [278].

Представленные в [278] времена расчета для H100 (в «закрытом»/предварительном разделе MLPerf Training) были получены на системе DGX H100, содержащей кроме 8 H100 два процессора Xeon (56 ядер) и работающей с дистрибутивом Ubuntu 20. Анализ данных для системы DGX A100 с $8 \times A100$-SXM-80GB на дающих сопоставимые с H100 результаты тестах показывает, что время расчета на H100 раза в два меньше, чем на A100, за исключением теста обработки естественного языка, где H100 быстрее в 3,8 раза.

В конце июня 2023 года появились данные о производительности H100 и A100 на новой версии MLPerf Training v3.0 [295], а в сентябре – на новой версии MLPerf inference datacenter v.3.1 [298], которые иллюстрируются в таблицах 18 и 19.

Данные для этих таблиц были отобраны в целях максимального возможного сопоставления производительности, но, поскольку были

Таблица 18. Данные по производительности H100 и GH200 в тестах MLPerf inference datacenter v. 3.1 (все данные категории available); данные отобраны (и округлены до двух значащих цифр) для максимальной сопоставимости для точности в 99%

Количество GPU	1	2	4	8
Классификация изображений/ResNet				
H100-PCIe-80GB	47(55)[1]	106(115)[2]	206(200)[2]	368(443)[1]
H100-SXM-80GB	73(89)[3]		312(354)[3]	584(704)[4]
GH200-96GB	77(93)			
A100-PCIe-80GB			147(158)[5]	
A100-SXM-80GB				305(340)[6]; 290(326)[7]
NLP/BERT				
H100-PCIe-80GB	4,6(5,7)[1]	9,1(12)[2]	18(23)[2]	35(46)[1]
H100-SXM-80GB	7.3(8.8)[3]		29(36)[3]	56(70)[4]
GH200-96GB	7,7(10)			
A100-PCIe-80GB			12(13)[5]	
A100-SXM-80GB				25(28)[6]; 25(28)[7]

Данные о производительности (число запросов в сценарии server и число проб, в скобках - в сценарии offline) **приведены не за секунду, а за миллисекунду**.

Все расчеты проведены с использованием TensorRT 9.0.0 и CUDA 12.2.

[1] Данные Nvidia на Gigabyte G482-Z54, с 2×EPYC 7742;
[2] Данные Dell на Dell PowerEdge R760x с 2×Xeon Platinum 8480+
[3] Данные Dell на Dell PowerEdge XE9640 с 2×Xeon Platinum 8468
[4] Данные Nvidia на DGX H100 с 2×Xeon Platinum 8480C
[5] Данные Dell на Dell PowerEdge R750x с 2×Xeon Gold 6338
[6] Данные HPE на HPE ProLiant XL675d Gen10 Plus с 2×EPYC 7763
[7] Данные Oracle на Oracle BM.GPU.A100-v2.8 с 2×EPYC 7J13

фактически получены уже после завершения отбора данных для обзора, подробно здесь не анализируются.

В таблице 18 приведены отобранные данные из набора тестов MLPerf inference datacenter версии 3.1. Данные этой таблицы показывают очевидный рост производительности H100 по сравнению с A100, более тонкий рост производительности H100-SXM относительно H100-PCIe и увеличенную производительность GH200 емкостью 96 ГБ по сравнению с H100-SXM (по сравнению с H100-PCIe – тем более). Это демонстрирует увеличение производительности для задач ИИ при работе с одним GPU H100 в интегрированном с Grace варианте GH200 с HBM-памятью емкостью 96 GB. Кроме того, эти данные показывают хорошую масштабируемость с ростом числа применяемых GPU в сервере от 1 до 8.

Таблица 19. Время вычисления (в минутах) в тестах машинного обучения MLPerf Training v3.0 [295]

Количество GPU	2	4	8
Классификация изображений			
A100-PCIe		58;[1] 61[2]	32[3]
A100-SXM		54[4]	27[5]
H100-PCIe	82[6]	45;[2] 39[7]	20;[8] 21[9]
H100-SXM		27;[10] 26[11]	13;[12] 13[13]
Сегментация медицинских изображений			
A100-PCIe		47;[1] 48[2]	
A100-SXM		47[4]	23[5]
H100-PCIe	67[6]	32;[2] 31[7]	19;[8] 18[9]
H100-SXM		22;[10] 22[11]	12;[12] 12[13]
Обнаружение объектов, легковесное			
A100-PCIe		171;[1] 176[2]	
A100-SXM		222[4]	79[5]
H100-PCIe		114;[2] 107[7]	54;[8] 56[9]
H100-SXM		72;[10] 72[11]	37;[12] 37[13]
Обнаружение объектов, тяжеловесное			
A100-PCIe		86;[1] 81[2]	47[3]
A100-SXM		83[4]	38[5]
H100-PCIe		62;[2] 55[7]	28;[8] 28[9]
H100-SXM		40[10]	19;[12] 20[13]
Распознавание речи			
A100-PCIe		63;[1] 64[2]	
A100-SXM		55[4]	29[5]
H100-PCIe	77[6]	51;[2] 45[7]	23;[8] 28[9]
H100-SXM		27;[10] 27[11]	19;[12] 19[13]
Обработка естественного языка			
A100-PCIe		45;[1] 51[2]	
A100-SXM		32[4]	15
H100-PCIe	43[6]		10;[8] 10[9]
H100-SXM		11;[10] 11[11]	5,4;[12] 5,4[13]
Рекомендация (DLRM)			
A100-SXM			8,4[5]
H100-SXM		8,8[10]	4,3;[14] 4,3[12]

[1] Данные от ASUSTeK на ESC4000-E11 с 2×Xeon Platinum 8462Y+
[2] Данные от Dell на R750x с 2×Xeon Gold 6338
[3] Данные от Lenovo на ThinkSystem SR670 V2 Server с 2×Xeon Platinum 8360Y
[4] Данные от Dell на XE8545 с 2×EPYC 7763 отправитель
[5] Данные от Dell на XE9680 с 2×Xeon Platinum 8480+
[6] Данные от Quanta Cloud Technology на D54Q-2U с 2×Xeon Gold 6430
[7] Данные от Dell на R760x с 2×Xeon Platinum 8480+
[8] Данные от ASUSTeK на ESC8000A-E12 с 2×EPYC 9654
[9] Данные от Supermicro на AS-4125GS-TNRT с 2×EPYC 9554
[10] Данные от Supermicro на SYS-421GU-TNX с 2×Xeon Platinum 8460H
[11] Данные от Dell на XE8640 с 2×Xeon Platinum 8468
[12] Данные от Dell на XE9680 с 2×Xeon Platinum 8470
[13] Данные от Supermicro на AS-8125GS-TNHR с 2×EPYC 9634
[14] Данные от GIGABYTE на G593-SD0 с 2×Xeon Platinum 8480+

Все данные в таблице относятся к доступным локально (available on-premise), получены на GPU с емкостью памяти 80 GB и **округлены до двух значащих цифр**. Информацию об используемых в тестах моделях ИИ и применяющихся наборах исходных данных см. в [295].

В таблице 19 приведена выборка из данных примерно одной шестой части всех представленных в [**295**] наборов данных о тестах производительности MLPerf Training v3.0. Там имеются, например, результаты и для большего числа использовавшихся GPU. Такой большой объем представленных в этой таблице данных позволяет сопоставить производительность H100-SXM, H100-PCIe и A100-SXM/PCIe при разном числе используемых GPU в сервере, с применением разного программного обеспечения и разных хостов.

Данные этой таблицы показывают, что возможные разумные вариации аппаратных средств хостов (понятно, что здесь использовались ЦП, содержащие десятки ядер) и программных средств, выбранные производителями серверов для этих тестов, не оказывают сильного влияния на достигаемую производительность, и поэтому в данном обзоре не анализируются.

Данные этой таблицы практически во всех тестах показывают хорошее масштабирование с числом GPU в сервере до 8. В качестве нулевого приближения для сопоставления производительности можно считать линейной зависимость производительности от числа GPU, хотя так бывает не всегда – например, при переходе от 4 к 8 H100-PCIe в тесте сегментации медицинских изображений ускорение было только процентов на 70.

Из данных таблицы видно, какая заметно более высокая производительность достигается при работе с H100-SXM относительно работы с H100-PCIe (например, раза в полтора больше на 4 GPU в тестах классификации изображений и сегментации изображений); или что производительность раза в два больше у H100-SXM, чем у A100 на этих же тестах с тем же числом GPU; или что H100-PCIe раза в полтора быстрее A100 при таких же условиях, и так далее. Бывает и скорее недостаточно высокое для ИИ масштабирование с числом GPU (например, при переходе от 4 к 8 H100-PCIe в тесте сегментации медицинских изображений). Но подобный анализ, по понятным причинам, здесь не проводится.

4.3. Резюме по GPU Nvidia

Несмотря на появление альтернативных Nvidia GPU от AMD и Intel, нет никаких признаков ослабления позиций GPU от Nvidia в аппаратном и программном плане – их применение в количественном плане будет продолжать быстро развиваться. Средства программного обеспечения для GPU Nvidia представлены наиболее широко, и новые программные средства обычно появляются для них раньше, чем для GPU конкурентов. Архитектуры новых поколений GPU от Nvidia дают высокий уровень совместимости с более ранними моделями GPU. Соответственно переносимость программного обеспечения на более новые модели существенно выше, чем достигаемая при переходе на работу с GPU конкурентов.

Хотя теоретически доля применяемых GPU от Nvidia на мировом рынке может упасть из-за появления там GPU от AMD и Intel, общемировое усиление работ в области ИИ будет способствовать активизации применения GPU от Nvidia. Возможные преимущества GPU Intel и AMD по производительности нужно доказывать на реальных приложениях. С точки зрения области HPC стоит скорее обратить внимание на все большую ориентацию Nvidia на задачи ИИ.

5. GPU нового поколения от AMD

К GPU нового поколения в обзоре относятся GPU, появившиеся после базовых GPU Nvidia V100 и A100. AMD за последние годы прошла большой путь не только в области успешной конкуренции с x86-процессорами Intel (AMD EPYC оттесняют Intel Xeon не только на рынке традиционных серверов, но и на рынке серверов с GPU). Появление в конце 2020 года MI100 [**296**], а в следующем году и семейства MI200 [**297**, **299**] показало, что с GPU Nvidia начинается конкуренция. Именно GPU MI100 и MI200 стали первыми выпускавшимися представителями GPU, отнесенных в данном обзоре к новому поколению.

MI100 иногда рассматривался как предшественник GPU AMD в ожидаемых суперкомпьютерах (кластер Spock с MI100 в узлах – как предшественник Frontier [**254**]; MI100 начал применяться в узлах LUMI, который занял третье место в июньском списке Top500, когда в его узлах уже стали использоваться MI250X [**42**]). Архитектура MI100 уже обсуждалась в публикациях – см., например, обзор [**21**]. По аналогии с V100 здесь мы ограничимся только сводными техническими характеристиками MI100 (см. таблицы 20, 21), но проведем анализ данных о достигаемой производительности этого GPU, в том числе в сопоставлении с другими рассматриваемыми в обзоре GPU.

Актуальность анализа GPU из семейства MI200 повышается тем, что с их применением сейчас построен не только суперкомпьютер №1 в Top500, впервые в мире перешагнувший барьер 1 EFLOPS (Frontier), и третий в списке Top500 (LUMI), относящийся к Евросоюзу (поскольку LUMI установлен в Финляндии – это демонстрирует и существенный европейский успех в суперкомпьютерной области): GPU MI250X применяются в 2% суперкомпьютеров из Top500, что отражает и очередной успех производителя, HPE/Cray, в аппаратных средствах которого MI250X размещались.

Таблица 20. Спецификации современных GPU AMD и Nvidia

Спецификации GPU	MI100	MI210	MI250	MI250X	A100 с PCIe	A100 с SXM4	H100 с PCIe	H100 с SXM5
Технология TSMC, нм	7	6			7		4	
Число активных ядер (потоковых процессоров)[1]	7680	6656	13312	14080	6912	6912	14592	16896
Число активных CU[2]	120	104	2×104	2×110	108	108	114	132
Частота GPU базовая/ускоренная (МГц)	1000/1502	1000/1700			765/1410 или 1065/1410	1095/1410 или 1275/1410	1095/17554	1590/19804
Пиковая производительность: FP64 (TFLOPS)	11,5	22,6	45,3	47,9	9,7	9,7	25,6	33,5
FP64 с тензорными ядрами (TFLOPS)	нет	45,3	90,5	95,7	19,5	19,5	51,2	66,9
FP32 (TFLOPS)	23,1	22,6	45,3	47,9	19,5	19,5	51,2	66,9
FP32 с тензорными ядрами (TFLOPS)	46,1	45,3	90,5	95,7	Нет поддержки: входные данные меньшей точности			
FP16/BF16 (TFLOPS)	184,6/92,3	181	362,1	383	78/312[3]		205[3]	268[3]
INT8 (TOPS)	184,6	181	362,1	383	624		1513	1978,9

[1] Низкоуровневые SIMD-ядра с поддержкой команд «умножить-и-сложить», у Nvidia именуются еще как CUDA-процессоры (матричные/тензорные ядра здесь не учитываются);

[2] для Nvidia – потоковые мультипроцессоры (SM);

[3] при использовании разреженности – в два раза выше; Производительность с FP16, BF16 и INT8 относится к тензорным ядрам.

Таблица 21. Спецификации современных GPU AMD и Nvidia (*продолжение*)

Спецификации GPU	MI100	MI210	MI250	MI250X	A100 с PCIe	A100 с SXM4	H100 с PCIe	H100 с SXM5
Тип памяти в GPU	HBM2	HBM2E						HBM3
Шина памяти, бит	4096		8192		5120			
Емкость памяти, ГБ	32	64	2×64		40 или 80		80[5]	
Частота памяти, МГц	1200	1600[6]			1215 или 1512	1215 или 1593	1593	1313
Пропускная способность, Гбайт/с	1229	1638	3277 ($2 \times 1638,4$)		1555 или 1935	1555 или 2039	2039	1681
Интерфейс GPU с GPU или с процессором	Infinity Fabric 2.0	Infinity Fabric 3.0			с ЦП: PCIe v4	NVLink-3.0	с ЦП: PCIe v5	NVLink-4.0
Его пропускная способность, Гбайт/с	3×92	3×100	6×100		с ЦП: 64	600	с ЦП: 128	900
Кэш L1, КБ	16 на CU				192 на SM		256 на SM	
Кэш L2, МБ	8	16			40 или 80	40	50	
Энергопотребление, Вт[4]	; 300 PSU: 700	; 300 PSU: 700	500; 560 PSU: 900	500; 560 PSU: 900	250 или 300 PSU: 700	400 PSU:800	350 PSU: 750	700 PSU: 1100

[4] есть модель с HBM3/96 ГБ и другой частотой GPU.

[5] для тензорных операций с пониженной (меньше FP32) точностью ускоренная частота меньше;

[6] TDP: число после точки с запятой у AMD – пиковое, PSU: предлагаемая мощность для Power Supply Unit;

Данные таблицы взяты из базы данных [185] и с сайтов производителей.

5.1. GPU AMD MI200

5.1.1. *Микроархитектура и технические характеристики разных моделей MI200*

Семейство MI200 включает в себя три разные модели – MI210 [**300**], MI250 [**301**] и MI250X [**302**], – из которых старшая модель MI250X появилась и стала поставляться первой в суперкомпьютер Frontier [**303**], лидер списка Top500.

При переходе AMD от производства своих традиционных графических процессоров Radeon Instinct на GPU нового поколения сменилась и архитектура. Если ранее в графических процессорах AMD применялась архитектура GCN (Graphics Core Next), то для MI100 была разработана CDNA (Compute DNA) [**304**], а в MI200 она модернизировалось уже во вторую версию CDNA 2 [**305**]. При изменениях этих архитектур, естественно, сохраняется определенная преемственность, в первую очередь у вычислительных блоков, усовершенствование которых во многом происходит по технологическим и количественным показателям.

Для аналога базового вычислительного блока SM в GPU Nvidia для GPU AMD используется термин CU (см. таблицу 1), который применяется также и в BR100. Достаточно подробная информация о построении CU и его основных блоках для CDNA имеется в [**304**]. CU в CDNA 2 содержит, в частности, диспетчер для работы с нитями, файлы обычных и векторных регистров, ядра (в том числе матричные, см. ниже), кэш L1 и память LDS (Local Data Share) [**305**]. Разделяемая память LDS (часто именуемая также LSM, Local Share Memory) используется нитями внутри варпа (wavefront в терминологии AMD, см. таблицу 1) [**305**]. Емкости кэша L1 для MI100 и различных моделей MI200, как и другие важнейшие показатели этих GPU AMD, в сравнении с A100 и H100 приведены в таблицах 20 и 21. С учетом задержек в поставках PVC и возможной заморозки производства BR100 наибольший интерес в данном обзоре и в целях сопоставления GPU вообще в настоящее время представляют именно GPU из этой таблицы.

Но прежде всего следует указать на технологические показатели. MI100 стали выпускаться по технологии TSMC 7 нм (как и у A100), а MI200 – по технологии 6 нм (только в последних только что появившихся H100 у Nvidia стало 4 нм). В MI100 содержится 25,6 млрд транзисторов при площади 750 мм^2 – а в MI250X их уже 58,2 млрд (при меньшей площади 724 мм^2) [**185**]. Эти показатели важны в том числе потому, что связаны с возможными стоимостными показателями и величинами TDP.

Эти данные показывают на близость таких показателей у MI100 и V100 (см. еще таблицу 10). При этом в MI250X число транзисторов несколько больше, чем у A100 (там их 54,2 млрд), а площадь кристалла MI250X при этом заметно ниже (в A100 – 826 мм^2). Соответственно возникают идеи о возможной сопоставимости MI100 и V100, а также MI250X и A100 – что было опробовано в ряде публикаций с данными о производительности. Что касается сопоставления технологических показателей MI250X и A100, оно требует принципиального уточнения, о чем речь пойдет чуть ниже.

К сравнению MI250X с A100 следует добавить сопоставление с появившимися в 2023 году H100. Здесь новый технологический уровень 4 нм позволил Nvidia разместить 80 млрд транзисторов с уменьшенной (по сравнению с A100) площадью 814 мм^2.

Основные показатели GPU MI100, MI200, A100 и H100 приведены для сопоставления в таблицах 20 и 21, а на рисунке 18 представлен более высокий (чем CU) уровень масштабирования вычислительных мощностей в CDNA 2 – GCD, который имеет в том числе 4 вычислительных блока CE (Compute Engine), содержащих по 28 CU (два столбца по 14 CU на рисунке) [**305**]. Из 112 физических CU два CU отключены [**41**]— соответственно в таблице 20 указано на 110 CU.

Кристалл GCD имеет принципиально важное значение для технологии AMD: в старших моделях MI250 и MI250X содержатся по 2 GCD [**305**], соединенных как чиплет [**299**].

В качестве межсоединения GCD применяется Infinity Fabric 3.0 (о нем речь пойдет в следующем разделе обзора), но его пропускная способность сильно ниже, чем пропускная способность памяти GPU (см. таблицу 21). При программировании MI250 и MI250X будут восприниматься как два различных GPU [**305**]. Соответственно, возвращаясь к технологическому сопоставлению с GPU A100 и H100, MI250/MI250X можно воспринимать и как пары GPU. И в расчете на один GCD MI250X число транзисторов в нем, грубо говоря, в 2 раза меньше, чем в A100.

Если перейти от архитектуры CDNA 2 к поставляемым моделям MI200, то они отличаются числом активных CU (см. таблицу 20). По данным [**305**], каждый CU содержит по 64 ядра (64 шейдерных ядра или потоковых процессоров в терминологии AMD, эквивалентные четырем SIMD-блокам с векторами по 16 чисел) – это некие аналоги CUDA-ядер в GPU Nvidia. Каждое такое ядро может выполнять команды «умножить-и-сложить» с FP64-числами. Тогда пиковая производительность GPU (FLOPS) в два раза больше произведения числа таких ядер GPU на тактовую частоту. По сложившейся традиции производители используют в расчетах ускоренную частоту – такие числа и приведены в данной таблице.

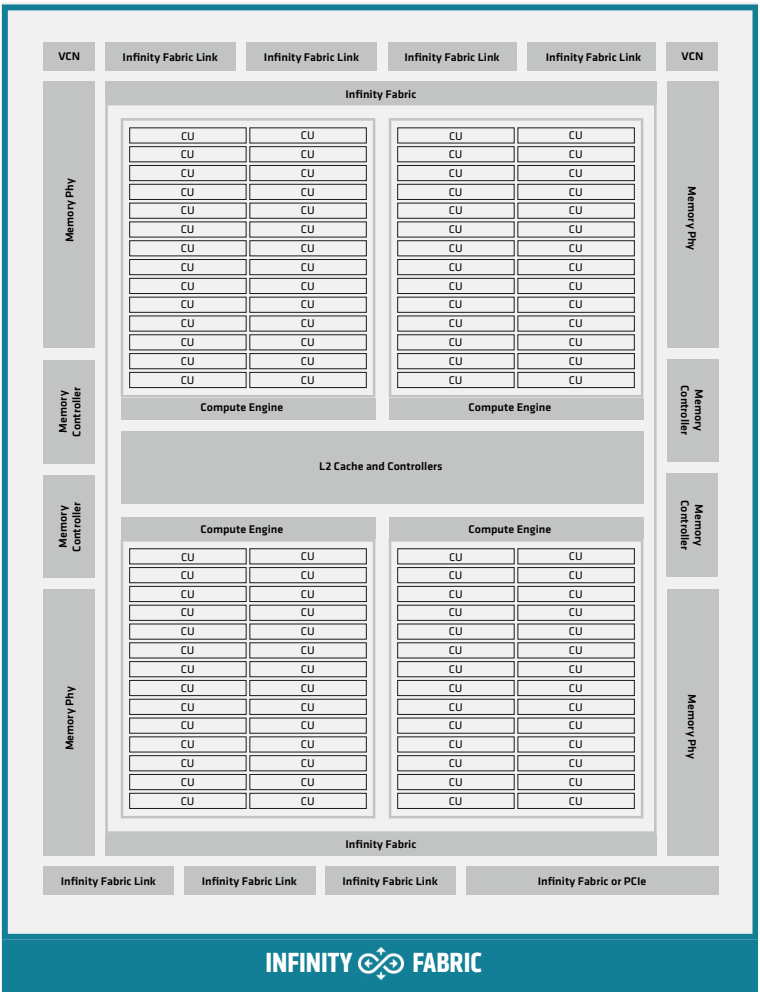


Рисунок 18. Общая построение кристалла GCD (рисунок из [305])

В CDNA2 располагающиеся в CU АЛУ стали 64-разрядными, и появилась возможность упаковать два числа FP32 на место одного FP64, и работать с ними, например, в командах «умножить-и-сложить» [305]. Пиковая производительность MI250X при работе с упакованными FP32 удваивается до 95,7 TFLOPS без использования матричных ядер.

В CDNA появились матричные ядра (аналоги тензорных ядер в GPU Nvidia), которые могут выполнять команды MFMA (Matrix-Fused-

Multiply-Add) – «умножить-и-сложить» над матрицами маленького размера с использованием смешанной точности, но там максимальная точность – это FP32 [304], как и в V100. Только в CDNA 2 – и соответственно в MI200 (аналогично A100) MFMA может работать и с форматом FP64. Допустимые размеры матриц с FP64 ($M \times N \times K$ в формуле (1)): $16 \times 16 \times 4$ или $4 \times 4 \times 4$ [299].

Здесь полезно отметить особенность набора команд MFMA (это набор, так как для каждого сочетания форматов чисел, используемых в разных матрицах, есть собственный вариант команды MFMA) и аналогичных команд WMMA в GPU Nvidia. Они выполняют операцию, выражающуюся формулой (1)) при $\alpha = \beta = 1$, но результат помещается в другую матрицу **D** вместо **C** в левой части этой формулы. Поддержка работы со специфическими разреженными матрицами для задач ИИ (как в A100) в CDNA 2 отсутствует. Документация по ISA для CDNA 2 имеется в [306].

Для расчета пиковых производительностей MI100 и MI200 нужно знать число получаемых результатов за такт в одном CU; эти данные представлены в таблице 22. Например, в каждом CU в CDNA 2 имеется по 4 матричных ядра, которые дают по 64 результата FP64 за такт [305], что после умножения общего числа матричных ядер на 64 и на тактовую частоту дает приведенную в таблице 20 пиковую производительность при работе с матричными ядрами.

Таблица 22. Число получаемых результатов за такт на одном CU в CDNA (MI100) и CDNA 2 (MI200)

Формат	Векторными устройствами		Матричными устройствами	
	CDNA/MI100	CDNA 2/MI200	CDNA/MI100	CDNA 2/MI200
FP64	64 FLOPS	128 FLOPS	—	256 FLOPS
FP32	128 FLOPS	128 FLOPS	256 FLOPS	256 FLOPS
FP16	—	—	1024 FLOPS	1024 FLOPS
BF16	—	—	512 FLOPS	1024 FLOPS
INT8	—	—	1024 числа	1024 числа

Данные из [307]. Там приводится также число тактов, требуемых для выполнения каждого типа операции MFMA. При выполнении MFMA смешанного формата с результатами FP32 за такт получается 1024 FLOPS.

Вообще, если сопоставить представленные в таблице 20 данные о пиковой производительности, то видно, что при работе с векторными ядрами MI100 опережает по этому показателю A100 (и естественно V100, см. таблицу 12) для FP64 и FP32, давая этим потенциальное преимущество для всех работающих с векторами приложений HPC (если там производительность не лимитируется умножением матриц). Поддержка FP64 в матричных ядрах MI100 отсутствует, а при точности ниже FP32 по этому показателю производительности MI100 конкурентоспособен с V100, сильно уступая A100.

На векторных ядрах содержащие по два GCD модели MI250 и MI250X превосходят по пиковой производительности с плавающей запятой модели A100. Это же имеет место и при работе с матричными/тензорными ядрами (за исключением FP32). Однако A100 для задач ИИ при использовании специальной разреженности матриц тензорными ядрами при работе с пониженной относительно FP64 точностью по пиковой производительности впереди.

Реально достигаемая производительность принципиально зависит и от других факторов, в первую очередь иерархии памяти – но сначала надо определиться, как производится сопоставление: логически модели MI250 и MI250X представлены как два GPU. Если поделить представленные в таблице 20 пиковые производительности этих двух моделей на два, то для FP64 все модели MI200 по-прежнему сильно опережают A100, а на FP32 их преимущество небольшое (для тензорных ядер у A100 приводятся и более высокие показатели, но при этом исходные данные не поддерживаются в стандартном формате FP32).

При сопоставлении с одним GCD только что появившиеся H100 по пиковой производительности с FP64 уже превосходят MI200, а на более низких точностях это превосходство большое. При этом нужно отметить, что пиковая производительность одного GCD MI250X для формата FP64 очень близка к H100-PCIe (см. таблицу 20). Систематическое сопоставление этих GPU на реально достигаемых производительностях во время подготовки обзора было невозможно из-за отсутствия соответствующих публикаций, а в практическом плане актуальность в ближайшее время может уже получить и сопоставление H100/GH200 с ожидаемыми к появлению MI300.

В CDNA 2 есть и другие исполнительные блоки, которые имеют отношение к работе с рисунками и видео, но они здесь не рассматриваются – например, в GCD имеется два блока VCN для задач машинного обучения при работе с изображениями и видео (см. рисунок 18).

Теперь следует обратиться к иерархии памяти – второй основной составляющей, определяющей производительность работы GPU. Наиболее детальные данные по всей иерархии памяти MI200 представлены в [**299**]. Кэш L1 используется для работы с файлом векторных регистров. Но ограничиться прямым сопоставлением емкости D-кэша L1 в CDNA 2 с емкостью такого кэша в A100 нельзя из-за того, что в CDNA 2 имеется еще отдельная разделяемая память LDS емкостью 64 КБ на один CU [**41**]. В V100 и A100 соответствующий кэш (размер которого приводится выше в таблице 21) унифицирован, и включает в себя L1 и разделяемую память.

Расположенный в GCD кэш L2 имеет емкость 8 МБ и является 16-канальным наборно-ассоциативным. Кэш L2 в GCD имеет общую скорость передачи данных с кэшами нижнего уровня 4096 КБ за такт [299]. Точнее говоря, кэш L2 в GCD состоит из 32 частей (slices), каждая из которых передает по 128 байт/такт, или суммарно 6,96 ТБ/с [41]. Через 3-е поколение межсоединения Infinity Fabric данные кэша L2 могут обмениваться с памятью HBM2E данными на скорости 2 КБ за такт. Подсоединенная к одному GCD память HBM2E имеет емкость 64 ГБ с пропускной способностью 1,6 ТБ/с [299].

Модели MI250 и MI250X имеют по два GCD. Соответственно на уровне всей модели перечисленные в предыдущем абзаце показатели удваиваются, как и приведено в таблице 21. Infinity Fabric в CDNA 2 стало обеспечивать когерентность кэша между GCD, и память HBM2E, как и кэш L2 обоих GCD являются общими [299]. Если сопоставление с A100 проводится относительно одного GCD, то емкость и пропускная способность HBM2E в CDNA 2 выше, чем для A100 – 40GB. Но Nvidia стала затем производить модели A100 с удвоенной емкостью HBM2E (80 ГБ), в которых емкость и пропускная способность стали больше, чем у одного GCD в MI250/MI250X.

Сравнивая перечислявшиеся в таблице 21 показатели памяти, можно сказать об их сопоставимости в разных моделях GPU AMD и Nvidia (MI100 и V100, MI200 и A100; H100 здесь не учитывается). Однако яркое отличие имеет место в емкости кэшей L1 и L2: в A100 они сильно больше, чем в GCD, что может иметь важное значение для достигаемой производительности. В MI250/MI250X емкость кэша L2 в расчете на один GCD осталась такой же, как в MI100 (8 МБ), чуть больше, чем в V100 (6 МБ) [185]. Но в A100 эта емкость была резко увеличена и стала в разы больше, чем у MI200. Что касается кэша L1, то в MI100 его емкость была всего 16 КБ (плюс LDS емкостью 64 КБ) на CU [262] против 128 КБ на SM в V100 (хоть эта память частично используется в качестве разделяемой памяти) [185]. В CU MI200 емкость кэша L1 также осталась как у MI100 (и также имеется отдельная память LDS).

При этом емкость кэша L1 в 8 раз меньше, чем у унифицированного кэша V100, и дополнительная LDS это может не компенсировать – даже после выделения части емкости унифицированного кэша V100 под разделяемую память, остающаяся под кэш L1 емкость там может оказаться гораздо больше, чем в MI100 и MI200. Это уже вызывало проблемы с производительностью (см. об этом далее в разделе 5.3.2.2 про производительность MI250/MI250X), так что по сравнению с A100 кэш-память MI200 является потенциальным узким местом.

Во всяком случае, в рассмотренных далее публикациях о производительности уже отмечались недостатки кэш-памяти в MI100 и MI200. Возможно, что для оптимизации приложений для MI200 там целесообразно применение не простой модели линии крыши (roofline), а эмпирической с учетом кэш-памяти.

Если сравнивать TDP (см. таблицу 21), то у сопоставляемых (с A100) GPU MI200 эти показатели также достаточно близки, хотя TDP моделей A100-SXM повыше. Также надо отметить, что указанные в этой таблице в строке TDP числа для GPU AMD относятся, как указано в документах [300–302], к TBP (Total Board Power).

В целом все уже рассмотренное выше говорит о том, что на аппаратном уровне важные для производительности показатели MI100 сопоставимы с V100, а у MI200 (в расчете на один GCD) в основном сопоставимы с A100, что делает интересным и актуальным сопоставление соответствующих реально достигаемых показателей их производительности.

Последнее, что следует обсудить в данном разделе – о поддержке PCIe в GPU MI100 и MI200. Ранее эта шина использовалась для связи между устройством и хостом, и ее ограниченная пропускная способность могла становиться узким местом для производительности GPU. Когда стали предлагаться серверы multi-GPU, эта шина могла бы применяться и для связи между GPU, что также стало бы узким местом в сервере. В рассматриваемых в данном разделе обзора GPU они все имеют специальное высокоскоростное межсоединение для связи между GPU (Infinity Fabric у AMD, NVLink у NVidia). Но для связи хоста с устройством применение такого специализированного межсоединения предполагает его аппаратную поддержку процессором хоста.

Это всегда выполнялось в GPU AMD, поскольку Infinity Fabric первоначально было ориентировано на связь между ЦП AMD. Для NVLink это не так – и требовало применения PCIe для связи с обычными серверными процессорами x86-64. Поэтому преимуществом IBM Power9 стала поддержка в нем интерфейса с NVLink для связи с V100, благодаря чему Power9 применяется и в таких знаменитых суперкомпьютерах, как Summit и Sierra. Во всех сравниваемых в таблице 21 GPU (кроме H100) имеется поддержка PCIe-v4 ×16 с пропускной способностью 64 ГБ/с.

Но здесь есть принципиальное различие этих GPU от AMD и от Nvidia: PCIe в GPU AMD для связи с ЦП EPYC не применяется, а является обычно используемым PCIe-интерфейсом, к которому подсоединяются сетевые платы, например, для связи между узлами кластера. Применение Infinity Fabric в GPU AMD дает им преимущество по пропускной способности соединения с ЦП, в том числе и по сравнению с A100-PCIe. Имеющийся

в MI100/MI200 интерфейс PCIe ориентирован на применение его совместно с поддерживающими эти GPU версиями MPI для обмена сообщениями между серверами, не вовлекая в это ЦП – это используется, например, в библиотеке плотной линейного алгебры SLATE, ориентированной на подобные системы с распределенной памятью [**309**].

Рассмотрение межсоединений GPU в данном обзоре проводится в разделах, посвященным вычислительным системам на базе GPU и считается как дополнительная специальная аппаратура для GPU, как и возможно специальные поддерживающие соответствующий интерфейс процессоры. Infinity Fabric поэтому рассматривается в следующем разделе.

5.1.2. *Вычислительные системы на базе MI200*

Прежде всего здесь необходим анализ межсоединения Infinity Fabric. Оно исходно было близко к PCIe и в настоящее время используется для связи между процессорами EPYC в серверах. Как и NVLink, у Infinity Fabric по мере развития появлялись новые версии, в CDNA 2 это уже третье поколение [**299**].

В MI200 имеются разные варианты соединений Infinity Fabric, подобранные для оптимальной работы в разных физических условиях (например, для коротких связей между GCD внутри одного GPU межсоединение специальное). Один канал Infinity Fabric имеет ширину 16 бит в одном направлении (так было и в MI100). Пропускная способность одного такого канала в CDNA 2 равна 50 или 100 ГБ/с в одном или двух направлениях, а между двумя GCD в одном GPU используются 4 канала с общей пропускной способностью 400 ГБ/с [**299**] (см. рисунок 19 с топологией связи в узлах суперкомпьютера LUMI [**41**]).

Для связи между GCD из разных GPU в multi-GPU сервере используется один или два канала Infinity Fabric с двунаправленной пропускной способностью соответственно 100 или 200 ГБ/с. Наконец, для связи MI250X с ЦП EPYC использует другой вариант Infinity Fabric с двунаправленной пропускную способность 72 ГБ/с для связи с каждым GCD; кроме того, через PCIe-канал подсоединяется межсоединение узлов, например, Cray Slingshot-11 [**41**, **305**] (см. рисунок 19).

Сами серверы с MI200 могут использовать вариант с одним или двумя ЦП EPYC, и там возможны разные топологии межсоединения [**305**]. На рисунке 19 [**41**] представлена топология для сервера с четырьмя MI250X и одним EPYC – это соответствует используемым узлам суперкомпьютеров LUMI [**41**] и Frontier [**303**].

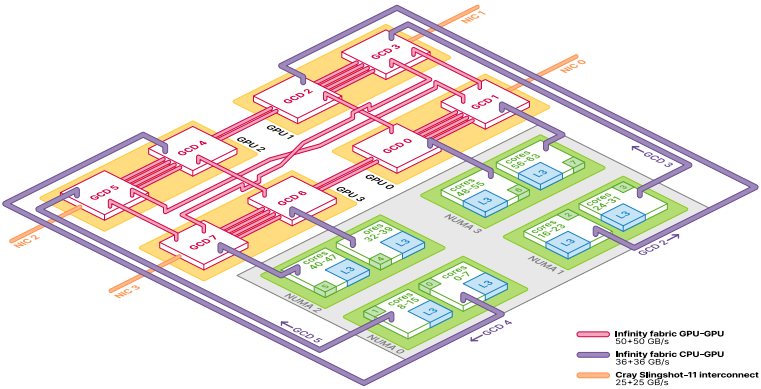


Рисунок 19. Использование Infinity Fabric в узле суперкомпьютера LUMI с четырьмя MI250X (рисунок из [41])

Рисунок 19 соответствует флагманской топологии узла для HPC; для него в [299] указано на общую пропускную способность Infinity Fabric 1,54 ТБ/с. В [305] в качестве «основного направления» для HPC и ИИ представлена другая топология сервера с четырьмя MI250, связанных с двумя ЦП EPYC через коммутатор PCIe, а также иллюстрируется еще одна топология для флагманского сервера для задач ИИ уже с восемью MI250 и двумя EPYC.

Свои модификации работы с Infinity Fabric имеет MI210, имеющий один GCD. MI210 обеспечивает пропускную способность 64 ГБ/с для связи с ЦП без необходимости применения коммутаторов PCIe, а для межсоединения MI210-MI210 можно использовать 3 канала Infinity Fabric с общей пропускной способностью 300 ГБ/с. Наконец, для multi-GPU сервера с MI210 тоже предлагаются варианты флагманский и «основного направления» [305].

Разные топологии могут иметь место и в multi-GPU серверах Nvidia. Однако наличие разного типа связи Infinity Fabric может усложнять оптимизацию высокомасштабируемого программирования в multi-GPU серверах с MI200. Однако в [308], где рассмотрены различные варианты межсоединений Infinity Fabric на сервере с четырьмя GPU MI250X, для пропускной способности связи ЦП-GPU не было выявлено значимых NUMA-эффектов на уровне API HIP.

В узлах суперкомпьютера Frontier используется по одному 64-ядерному процессору EPYC 7A53 (работающему на частоте 2 ГГц) с 4 MI250X. Этот ЦП с кодовым названием Trento был создан специально для работы с MI250X в узлах Frontier. Емкость кэша L3 в этом ЦП равна 32 МБ у каждой

из восьми 8-ядерных групп (всего 256 МБ на ЦП). ЦП сконфигурирован как 4 NUMA-узла, к каждому из которых подсоединяется по 128 GB памяти, так что ее емкость в сервере составляет 512 ГБ (8 каналов DDR4-3200) [41, 299, 303]. Выше была рассмотрена возможная топология применения Infinity Fabric в multi-GPU серверах с MI250X (см. рисунок 19); в [41] указано на важность правильной привязки узла NUMA к GPU, которая может иметь решающее значение для оптимизации производительности приложения. Frontier создавался Cray/HPE, и для связи серверных узлов использовано межсоединение Slingshot-11 [299].

В занимающем третье место в Top500 суперкомпьютере LUMI используются такие же аппаратные средства, как во Frontier [41, 42]. Здесь надо еще указать на активно использующуюся в исследованиях систему Crusher, содержащую 192 таких же узла, что и Frontier (9408 узлов) и LUMI (2560 узлов), и применяющуюся для тестирования и разработки. Еще раньше предшественником Frontier считался кластер Spock, имеющий 36 узлов, содержащих по 4 MI100 [310] (см. таблицу 23 ниже).

Естественно, практически все ведущие производители серверов мира поставляют модели, содержащие GPU MI200. Применение их в модулях стандартного форм-фактора OAM (OCP Accelerator Module) [305] способствует быстрому и широкому распространению таких серверов. Доступны самые разные серверы – от 1U до 4U, а также 6U и 10U, содержащие от 1 до 8 GPU (а также 10 и 20) MI100, MI210 и MI250. Эти серверы могут иметь 1–2 ЦП AMD EPYC архитектур Zen 2 и Zen 3, а также масштабируемые процессоры Intel Xeon 2-го и 3-го поколений. Список таких серверов доступен в [311].

Для достижения максимальной производительности при работе с GPU MI250/MI250X требуется жидкостное охлаждение (без него иметь больше 500 Вт обычно невозможно). Классическим примером этого является блейд-сервер HPE Cray EX235a [46], используемый в соответствующих суперкомпьютерах.

5.2. SDK для MI100 и MI200

Рассмотрение SDK в этом разделе обзора сильно ограничено, поскольку большинство соответствующих компонент от AMD является прямым аналогом описанных выше SDK от Nvidia, названия которых тоже часто очень похожи. SDK от AMD для рассматриваемых GPU именуется ROCm (Radeon Open Compute, а m – от "multi-GPU computing"в основе этих средств [312]). В качестве определенного преимущества SDK от AMD можно указать на его доступность в виде исходного текста [312], в то время как SDK от Nvidia просто свободно доступен.

Если в 2020 году, когда появился MI100, была доступна версия ROCm 4.0 [**312**], то к середине 2023 года перед появлением июньского списка Top500 была доступна уже версия ROCm 5.5.1 [**313**]. Если основой SDK от NVidia следует считать API CUDA, то основой от AMD является API HIP [**58**]. HIP как модель программирования крайне близка к CUDA, и там используются аналогичные термины (см. таблицу 1). В HIP при работе с GPU AMD используется варп (wavefront в терминологии AMD) размером 64 нити [**313**], а не 32, как для SM в Nvidia, что связано с архитектурой CU.

Но программы с GPU Nvidia стали переноситься на графические процессоры AMD безотносительно к появлению MI100. Так, информация о достигаемой производительности при переносе различных программных комплексов из популярной области применения современных GPU, классической молекулярной динамики, представлена в [**314**].

В качестве преимущества HIP над CUDA можно указать возможность применения HIP при работе на GPU от Nvidia, дающую возможность переносимости программного обеспечения для GPU AMD на аппаратуру Nvidia. Но здесь возникают два важных уточняющих момента. Во-первых, нужны данные о сопоставлении достигаемой производительности при работе HIP относительно достигаемой с применением CUDA (об этом речь пойдет чуть ниже). Поскольку HIP использует при работе с GPU Nvidia определенные бэк-энды от CUDA, это снимает возможные проблемы.

Обычно считается, что применение HIP почти не вызывает ухудшения производительности по сравнению с прямым кодированием на CUDA [**42**]. Реально демонстрирующие это данные для одного из алгоритмов для решения задачи CFD с неструктурированной сеткой имеются в [**168**]. Но в новой версии CUDA для H100 имеются расширения, из которых наиболее ярким можно считать изменение иерархии применяемых наборов нитей (появился кластер блоков нитей). Соответственно потенциальные проблемы эффективности HIP для GPU Nvidia могут сохраняться.

Стек программного обеспечения ROCm на нижнем уровне включает драйверы. В настоящее время поддержка ROCm обеспечивается в Linux (драйверы входят в состав модуля ядра, который можно инсталлировать в дистрибутивах Ubuntu, RHEL и SLES). Понятно, что ROCm включает все типичные компоненты для SDK. Имеется компилятор, именуемый ныне ROCmCC (с поддержкой HIP, OpenMP и OpenCL), который основан на Clang/LLVM [**313**]. Естественно, имеется профилировщик, отладчик и библиотека для трассировки производительности.

Математические библиотеки бывают с префиксами roc или hip. Префикс roc используется в названиях библиотек, оптимизированных для GPU AMD, а префикс hip – для библиотек, где в качестве бэк-энд применимы средства для GPU Nvidia [313]. С учетом префикса полные названия этих библиотек делают очевидными их предназначения и здесь не приводятся. Имеются и библиотеки коммуникаций, аналогичные соответствующим средствам от Nvidia. Например, аналогом NCCL от Nvidia в ROCm является RCCL.

Благодаря даже синтаксической похожести HIP и CUDA представляется естественным наличие в ROCm средства преобразования из CUDA в HIP, HIPIFY [313]. Но в общем случае это следует рассматривать как первый полуавтоматический шаг на пути такого преобразования, который может потребовать и дальнейшей ручной доработки хотя бы потому, что не все CUDA API поддерживаются в HIP (см., например, [42]). Здесь следует указать и на некую аналогию HIPIFY со средствами Intel DPCT.

Важным представляется указать здесь на проводящийся в Гейдельбергском университете (Германия) проект hipSYCL с открытым исходным кодом (теперь именуемый *AdaptiveCpp*[URL]) – реализация SYCL, предназначенная для ЦП и графических процессоров разных производителей. В [165] уже была предпринята попытка некой реализации oneAPI без компилятора DPC++, с помощью hipSYCL. Сравнение производительности MI250X при использовании разных вариантов реализации SYCL, DPC++, а также HIP и OpenMP на разных приложениях проведено в [176].

Еще одним важным для работы с GPU AMD является проект GPUFORT [315]. Эти средства осуществляют преобразование одного исходного кода в другой исходный код. Возможно два варианта такого преобразования исходного текста. Во-первых, из CUDA Fortran (или возможно OpenACC) в вариант Fortran с директивами OpenMP версии 4.5. Далее полученный текст может компилироваться с помощью AOMP (базирующегося на Clang/LLVM компилятора, имеющего фронт-энд для Fortran). Во-вторых, возможно задействование средств HIPFORT, которые обеспечивают интерфейс Fortran и HIP времени выполнения и доступ к математическим библиотекам. Если компиляция идет для выполнения программы на GPU Nvidia, HIPFORT дает интерфейс со средствами CUDA времени выполнения и соответствующими математическими библиотеками [316].

Что касается Fortran, следует отметить наличие поддержки рассматриваемых GPU AMD вне ROCm в знаменитом HPE Cray Fortran, где имеется и отсутствующая в ROCm поддержка OpenACC.

Ясно, что обеспечивается и работа средств более традиционного распараллеливания MPI и SHMEM. В [317] исследована производительность ряда поддерживающих работу с CDNA и CDNA 2 вариантов MPI – OpenMPI, Cray MPICH, MVAPICH2-GDR (а также средств RCCL), в том числе при использовании межсоединения Cray Slingshot-10 в знаменитом кластере Spock (он имеет в узлах 64-ядерный EPYC 7662 и четыре MI100).

Мы не говорим здесь о программных средствах (фреймворках) для задач ИИ, имеющихся в составе ROCm, поскольку они специализированы только на ИИ, а обзор в первую очередь обращен на HPC – но ROCm, естественно, интегрирован с основными фреймворками [313].

В заключение данного раздела можно отметить, что и в аппаратуре, и в программных средствах GPU MI100 и MI200 не проявляется такой яркой ориентации в первую очередь на задачи ИИ (а на HPC – скорее вторично), как в GPU H100 (и отчасти в A100). У Nvidia это проявляется не только в документации, но и в появляющихся новых суперкомпьютерах: в отличие от более «традиционно ориентированного» Frontier с AMD MI250X, H100 используется в суперкомпьютерах из Top500, ориентированных на задачи ИИ. Nvidia выпускает уже подготовленные и ориентированные на ИИ высокопроизводительные вычислительные системы вплоть до суперкомпьютерного уровня.

5.3. Данные о производительности MI100 и MI200 в тестах и на приложениях

К моменту появления июньского списка Top500 стало доступно немало статей, в которых рассматривалась производительность этих GPU в разных тестах и на разных приложениях и, в том числе, со сравнениями производительности относительно V100 и A100. В большинстве случаев, когда имелось достаточно публикаций для конкретных рассматриваемых моделей GPU AMD, в данном обзоре для анализа приоритетными были работы, относящиеся к HPC (но многие данные про ИИ также приводятся). Среди этих публикаций также были отобраны статьи, относящиеся к тестам производительности широко распространенных и актуальных математических задач. Из публикаций по приложениям в данном обзоре также отбирались традиционные известные в HPC области, например, задачи вычислительной химии или CFD, с приоритетом на известные в мире приложения.

Но здесь необходимо особо отметить работы по проекту ECP и сводные данные обзорного характера по более чем 20 приложениям (точнее говоря,

проектам), специально ориентированным на экзамасштаб – например, [262]. Там приведено большое число дающих сопоставление производительности данных для MI100, MI250X, V100 и A100. Поскольку оценки производительности в предельном случае интересны для конкретного приложения или хотя бы приложения из соответствующей области, во многих случаях полезно смотреть на данные в [262], где производительность представлена и на уровне одного GPU.

Однако надо иметь в виду, что в ECP применяются в основном не всемирно используемые приложения, а их специальные варианты (возможно, части), сделанные для ориентации на экзамасштаб, или новые разработки. И это может быть вовсе неэффективным и не дающим точные оценки для производительности в более типичных приложениях, для маленьких вычислительных систем или при расчетах не столь сложных объектов. Кроме того, это только что полученные первоначальные данные продолжающихся работ по разным проектам, с использованием в GPU AMD ранних версий ROCm, и результаты явно будут усовершенствованы (некоторые уточняющие статьи, относящиеся к конкретным проектам уже появились). Соответственно здесь появлялось весьма большое число критических замечаний относительно многих компонентов ROCm разных версий от 4.2.0 до 4.5.0, которые AMD пыталась быстро исправить.

Поскольку очень много из приводимых далее данных о производительности GPU MI100 и MI200 было получено с использованием известных суперкомпьютеров (в том числе из числа лидеров в Top500), краткая информация о таких вычислительных системах (в том числе применявшихся и для сопоставления с GPU от Nvidia) сведена в таблице 23.

5.3.1. *Производительность MI100*

Здесь будут рассмотрены сопоставительные данные о производительности MI100 по сравнению с V100 и A100 (все относительно MI200 рассматривается далее).

Что касается сопоставления производительности GPU MI100 и A100, то, хотя данные таблицы 20 показывают небольшие преимущества MI100 по пиковой производительности (например, с FP64), на эти показатели для GPU надо обращать ограниченное внимание – не реже производительность коррелируют здесь с пропускной способностью памяти, а точнее надо смотреть на данные модели линии крыш. Имеющиеся данные статей в целом однозначно указывают на большие (часто в разы) преимущества A100 относительно MI100 по производительности, хотя бывают и исключения.

Таблица 23. Вычислительные системы, узлы которых активно применялись в публикациях о производительности GPU, цитируемых в данном обзоре; для суперкомпьютеров из Top500 в скобках приведен их номер в списке

Вычислительная система	Число и тип GPU в узле	Число и тип ЦП в узле	Межсоединение (между узлами)[1]
Frontier (1) [303]	4×MI250X	1×EPYC 7A53(Trento)	4×HPE Slingshot-11
LUMI (3) [318]	4×MI250X	1×EPYC 7A53(Trento)	4×HPE Slingshot-11
Crusher [319]	4×MI250X	1×EPYC 7A53(Trento)	4×HPE Slingshot-11
Spock [320]	4×MI100	1×EPYC 7662(Rome)	1×HPE Slingshot-10
Leonardo (4) [205]	4×A100-40GB	1×Xeon Platinum 8358	4×Nvidia Infiniband HDR
Perlmutter (8) [206]	4×A100-40GB	1×EPYC 7763	4×HPE Slingshot-11
ThetaGPU [321]	8×A100-40GB	2×EPYC (Rome)	8×Infiniband HDR
Polaris (19) [322]	4×A100-40GB	1×EPYC 7543P (Milan)	Slingshot-10
Summit (5) [323]	6×V100-16GB	2×Power9	2×Mellanox Infiniband EDR

[1] Указаны межсоединения, использованные в цитируемых в обзоре публикациях

Поэтому приведем здесь только несколько подтверждений этому, в основном в интегральном плане (с широким охватом приложений) – а не базируясь на отдельных конкретных примерах. Например, в [45] приведены соответствующие сказанному выше данные на 6 разных приложениях и мини-приложениях (из разных областей науки), входящих в проект ECP или связанных с ними мини-приложениях. По данным [45] нами было рассчитано, во сколько раз достигаемая производительность A100 была выше MI100: для AMR-Wind (CFD-часть в проекте прогнозного моделирования потока ветряных электростанций ExaWind) – на трех разных программных ядрах в 1,7–5,3 раза; в квантовохимическом мини-приложении GAMESS RI-MP2 – в 5,8 раза; в мини-приложении TestSNAP для квантового потенциала SNAP, используемого затем в приложении молекулярной динамики LAMMPS (из проекта EXAALT для задач термоядерного синтеза) на трех разных программных ядрах – в 2,2–7,9 раза и так далее.

В качестве другого примера данных о производительности интегрального характера по еще большему числу проектов и приложений из ECP укажем чуть более позднюю работу [262] (см. также [324]); комментарии о [262] приведены выше. Наши вычисления относительной

производительности по данным [262] дают, что A100 был быстрее где-то в два раза на задачах CFD (NekRS), и в три раза – на задачах квантовой химии (Gamess, сборка фокиана) и молекулярной динамики (LAMMPS). Для TestSNAP A100 был быстрее MI100 в 2,8 раза.

Более новая версия ROCm 4.5.2 на другом приложении CFD, STREAmS-2 [325], также дала похожие величины отставания MI100 (см. рисунок 20) [325]. Подробнее эти данные рассмотрены ниже в разделе 5.3.2.2. Мало вероятно, что прогресс в новых версиях ROCm может полностью нивелировать такое отставание.

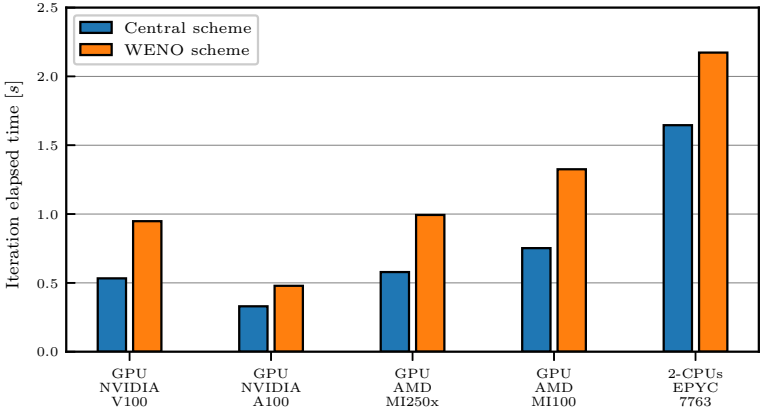


Рисунок 20. Время на итерацию (с) для STREAmS-2 с сеткой $420 \times 250 \times 320$ для двух расчетных схем: центральной схемы оценки потока и схемы WENO (рисунок из [325])

В [225] производительность с применением MI100, A100 и V100 исследовалась на SPEChpc 2021, содержащем компоненты из приложений разных областей HPC, но данные для MI100 были получены в отличном от использовавшегося в других GPU маленьком варианте SPEChpc.

Производительность приложений является наиболее практически актуальной, но соответствующие данные о производительности MI100 относительно A100, показывающие аналогичное приведенным выше данным, имеются и для конкретных математических методов – например, для БПФ [251], для QR-разложения квадратных матриц по библиотеке MAGMA для одинарной и двойной точности [241], для тестов пропускной способности BabelStream [42]. О тестах BabelStream есть и другие данные для их реализации с применением Fortran-средств DO CONCURRENT [236], где пропускная способность памяти MI100 слабо отличалась от C++ -варианта BabelStream (иногда Fortran давал более высокие результаты). Пропускная способность как для A100-80GB, так и для A100-40GB в [236], естественно, найдена гораздо больше, чем у MI100.

Современные GPU, их SDK и использующие их приложения являются сложными; приложения требуют для разных GPU разные параметры для оптимизации, а из всех правил бывают исключения. Так, в [241] данные на том же QR-разложении с rocBLAS показывают преимущество MI100 относительно cuBLAS с A100. В [326] в рамках программного кода AnySeq/GPU для задач биоинформатики с использованием ядер FP32 в MI100, V100 и A100 производительность (в триллионах обновлений ячеек в секунду, TCUPS) составила соответственно 3,8, 1,7 и 3,3 TCUPS. Но в основном в публикациях сильное отставание MI100 от A100 по производительности проявляется достаточно четко.

Сопоставление производительности MI100 с V100 представляется более актуальным, в том числе как возможное дополнение или база для сопоставления A100 и MI200. Здесь достигаемые производительности действительно бывают достаточно сопоставимы, и пиковая производительность FP64 у MI100 все-таки заметно больше, чем у V100. Это дает шансы MI100 получить более высокую производительность для приложений с высокой вычислительной интенсивностью (в терминологии модели линии крыши). Сразу приведем яркий пример этого в [262], где на квантовохимическом приложении NWChemEX в расчете с вычислительно интенсивным методом связанных кластеров CCSD сопоставление производительности узлов Spock (по 4 MI100) и Summit (по 6 V100) показало преимущество Spock. Там NWChemEX использовала средства CUDA на Summit и HIP (ROCm 4.3.0) на Spock.

Рассчитанные нами величины ускорения Spock относительно Summit (по данным Table 11 в [262]) дают, что на одном узле Spock полное время CCSD-расчета в 1,5 раза меньше, чем на узле Summit, а время дополнительного более сложного расчета поправки T для тройных возбуждений на узле Spock меньше в 1,9 раза.

С ростом числа узлов до 4 преимущество производительности Spock при равном числе узлов быстро уменьшается, и на 4 узлах Spock быстрее только по времени расчета T [262]. Нужно сказать, что, наверное, большинство современных приложений на GPU являются чаще связанными памятью, а V100 по ее пропускной способности менее отстает от MI100.

В [262] приведены сопоставительные данные о производительности глубокого обучения для MI100 и V100 в рамках двух проектов ECP – CANDLE (прецизионная медицина для онкологии) и ExaLearn (цель проекта понятна из его названия). В CANDLE производительность BERT умеренно больше на MI100, чем на V100, но не в той степени, на которую указывают относительные характеристики этих GPU (очевидно, в [262] имеются в виду пиковые производительности аппаратных средств). Для ExaLearn в [262] указано на сильно более низкую производительность MI100.

Важно, что кроме более высокой пиковой производительности, MI100 показала более высокую, чем у V100, пропускную способность памяти на всех компонентах теста BabelStream с применением широкого набора разных программных моделей [42]. В этой же работе производительность приложения miniBUDE (молекулярный докинг) на MI100 оказалось выше, чем на V100-32GB и почти совпала с производительностью A100-40GB.

Очень актуальным является сопоставление производительности MI100 и V100 на DGEMM, которое было проведено в [242] для пакетной и обычной реализации умножения матриц. Соответствующие данные представлены на рисунке 21 с применением ROCm 4.2 (hipBLAS) для MI100 и CUDA 11.2 (cuBLAS) для V100. Параметры m, n, k на оси абсцисс здесь

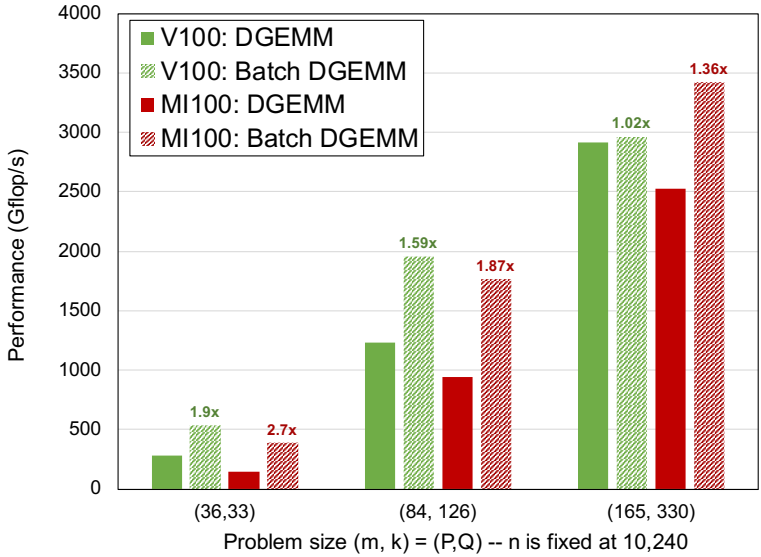


Рисунок 21. Производительность MI100 и V100 на DGEMM (рисунок из [242])

соответствуют размерностям матриц M, N, K в формуле (1). Видно, что MI100 может и опережать V100. В [242] приводятся также данные для тестов производительности CEED из ECP, показывающие сопоставимую производительность MI100 и V100, и о производительности на NekRS, где MI100 отставал на 15% от V100. На те же 15% отставания указывается и в [265].

В [327] проводились расчеты по квантовохимическому приложению QUICK в сочетании с приложением молекулярной динамики AMBER (по схеме QM/MM). В стартовых тестах MI100 давал производительность раза

в два меньше, чем V100, из-за слишком активного использования регистров в программных ядрах AMD. После доработок в расчетах систем больших размеров производительности MI100 и V100 оказались сопоставимыми (на маленьких системах V100 был гораздо быстрее). Программное ядро, считающее градиент обменно-корреляционного потенциала, на MI100 считало гораздо быстрее, чем на V100 [**327**].

В относящейся к ECP работе [**328**] сопоставлялась производительность MI100 (в Spock) и V100 (в Summit) в рамках известного в области CFD теста производительности HipBone (он ориентирован на работу с GPU, базируется на известном использующем C++ тесте NekBone, но не покрывает полностью его функциональность [**142**]), и сравнена производительность при разных степенях полинома у программного ядра оператора Пуассона, фактически определяющего производительность HipBone. В HipBone используется FP64, но в некоторых подзадачах предварительной обработки применяется FP32.

На одном GPU полученная в [**328**] производительность MI100 (с ROCm v4.5.0) примерно одинакова с производительностью V100 (с CUDA v11.0.3) на всех степенях полинома – и иногда быстрее MI100, а иногда – V100. Наивысшая производительность достигается при наивысшей использованной степени полинома (N=15): для V100 – 2101,4 GFLOPS, для MI100 – 2135,2 GFLOPS; была показана также хорошая масштабируемость на узлах Spock и Summit при числе рангов MPI до 32 и 48 соответственно [**328**].

В [**262**] на основании данных для некоторых проектов ECP отмечается сопоставимость производительности MI100 и V100 – это говорится, например, для данных AMR-Wind (наша оценка ускорения V100 относительно MI100 по рисунку 20 в [**262**] – 1,4 раза).

В проекте ExaSMR (моделирование активной зоны малого модульного реактора путем сочетания двух частей, нейтронной физики и CFD – для последней применяется NekRS) код Shift с HIP (ROCm 4.2) для первой части сначала уступал V100 с CUDA раза в два, но после оптимизации отставание составило всего 20%. В NekRS-части ExaSMR в некоторых программных ядрах отставание MI100 от V100 на Summit составляло всего несколько процентов [**262**]; наш расчет по данным таблиц 42 и 43 из [**262**] дает ускорение V100 относительно MI100 на 10–20%.

В других программных компонентах других проектов ECP в [**262**] указывалось на более серьезное отставание производительности MI100. На приложении QMCPACK (для квантовых расчетов методом Монте-Карло) в расчетах суперячейки оксида никеля в разных случаях достигались сильно разные отставания, но в целом MI100 давал значительно более низкую производительность, и критиковались компиляторы AMD.

В TestSNAP MI100 в Spock отстает от V100 в Summit примерно в 2 раза (наш расчет по таблице 19 в [**262**]). В проекте ECP с квантовохимическим приложением Gamess сборка фокиана в разных типах базиса с длиной от 4 до 12 тысяч орбиталей MI100 медленнее в 1,4–2,6 раза (наш расчет по таблице 15 в [**262**]). Естественно, в [**262**] имеется много других данных с сопоставлением производительности V100 и MI100. / В [**329**] производительность MI100 и V100 сопоставлена с применением мини-приложения MiniMDock для молекулярного докинга. В оптимальных для производительности на GPU вариантах (с HIP и CUDA) MiniMDock при использовании лигандов средних и больших размеров MI100 (Spock, ROCm 4.5) уступал по производительности V100 (Summit, NVHPC 21.11) на 32 и 39% соответственно, но на маленьких лигандах был немного быстрее, чем V100.

В [**330**] проводилась оптимизация некоторых критически важных программных ядер для CFD-приложения FUN3D, производительность которого на A100 обсуждалась выше. Для всех из них, кроме программного ядра вязкого потока, MI100 (с ROCm 4.2.0) был на 18–53% медленнее, чем V100 (с CUDA 11.2).

Естественно, MI100 применяется в самых разных областях, не только в HPC, но и для задач ИИ, например, для глубокого обучения в модели DLRM [**331**].

Можно говорить о сопоставимости MI100 по производительности с V100, хотя MI100 чаще отставал. Многие приводимые выше данные о производительности MI100 относятся к предварительным результатам, полученным сразу после появления MI100 в доступности, и эти результаты могут быть еще достаточно существенно улучшены на новых версиях ROCm (версии CUDA выглядят гораздо более стабильными). Но в связи с однозначными преимуществами всех показателей MI210 относительно MI100, не только указанных в таблицах 20 и 21 но и всех других известных автору на момент написания обзора, интерес к MI100, по мнению автора, относится к применению MI100 как уже приобретенного, а приобретение нового MI100 нецелесообразно по сравнению с MI210. Поэтому более полное рассмотрение имеющихся данных о производительности MI100 здесь не проводится.

5.3.2. *Производительность MI200*

Если посмотреть на данные таблиц 20 и 21, то ясно, что в расчете на один GCD модели MI210 и MI250 почти не отличаются, но GCD в MI250X имеет небольшое увеличение числа CU, которое должно давать небольшое увеличение производительности. Поэтому, хотя мы начнем анализ достигавшейся производительности с ряда известных автору (на момент подготовки обзора) актуальных статей с данными про MI210, это может представлять интерес и для оценок двух других моделей семейства MI200.

5.3.2.1. Производительность MI210. В качестве базовых показателей достигаемой производительности MI210 укажем сначала на данные Dell, представленные этой фирмой для своих серверов PowerEdge R750x – для умножения матриц с двойной и одинарной точностью (DGEMM и SGEMM) [**332**], в том числе в сопоставлении с данными для MI100. В расчетах Dell на MI210 использовались тензорные (матричные) ядра и ROCm 5.1.3. Для DGEMM была достигнута производительность 28,3 TFLOPS, на SGEMM – 32,2 TFLOPS. Эти данные [**332**] свидетельствуют о большом ускорении в MI210 относительно MI100 для DGEMM – в 3,6 раза (но в MI100 в тензорных ядрах нет поддержки FP64), и невысоком достигнутом проценте от пиковой производительности в MI210 (62,5%).

Для SGEMM на MI210 в была получена производительность на 9% больше, чем в MI100 [**332**]. Но FP32 поддерживается в тензорных ядрах MI100, что дает MI100 даже чуть более высокую пиковую производительность одинарной точности, чем у MI210 (см. таблицы 20 и 21).

В тесте HPL производительность MI210 достигает 18,2 TFLOPS на 1 GPU и почти линейно растет до 4 GPU (до 72,6 TFLOPS), а производительность MI100 на 1 GPU в разы меньше, и медленнее растет с числом GPU в сервере [**332**].

Для HPL есть и данные самой AMD, позволяющие продемонстрировать и возможное влияние версии ROCm [**333**]. Здесь с использованием более новой ROCm 5.2.0 и rocHPL 6.0.0 на одном GPU тест HPL дал 21,07 TFLOPS (в 1,16 раз быстрее, чем у Dell), на 4 GPU – 81,097 TFLOPS (в 1,12 раз быстрее), на 8 GPU – 159,73 TFLOPS. Как отмечается в [**333**], производительность может меняться и в зависимости от версии драйвера. Мы предполагаем, что влияние отличия других компонент использованных в тестах серверов в [**333**] и [**332**] здесь можно считать незначительным.

На тесте HPL-AI со смешанной точностью FP16/FP32/FP64 на 4 GPU в [**333**] получена производительность 444,77 TFLOPS.

В [**332**] приведены также данные о производительности MI210 относительно MI100 на приложении молекулярной динамики LAMMPS для нескольких разных ее типов (в том числе реактивной молекулярной динамики) и потенциалов – при работе с FP64 MI210 быстрее на 18–30%, а с FP32 – на 12%. Данные о производительности LAMMPS для реактивной молекулярной динамики приводит также и AMD с применением более новой версии ROCm [**333**].

Первые статьи с данными о производительности MI210 стали появляться для приложений в разных областях. Например, в [**334**] приводятся данные о производительности на разработанном в этой статье приложении для молекулярного докинга Uni-Dock, превосходящем по производительности известное приложение AutoDock-GPU (хотя подробные данные о производительности Uni-Dock в [**334**] приводятся для V100 и MI100).

Среди работ, дающих сопоставление производительности MI210 и GPU Nvidia, укажем на данные в области CFD [268] для упоминавшегося выше в разделе про производительность A100 приложения FUN3D с его библиотекой FLUDA для GPU. На основе данных таблицы 4 в [268] мы получаем, что A100-40GB быстрее MI210 в 1,44 раза (а MI210 быстрее V100-16GB в 1,29 раза).

В [335] было проведено сопоставление производительности MI210 и A100-40GB при работе фреймворка глубокого обучения PyTorch (использовался PyTorch 2.0-20230102) с применением стеков ROCm 5.4.2 и CUDA 11.8. В этой статье был предложен TorchBench – новый набор тестов производительности программного стека PyTorch. Для сравнения, классический набор тестов MLPerf для глубокого обучения, результаты которого с применением GPU обсуждались выше в данном обзоре, включает в себя 8 моделей глубокого обучения, а в TorchBench их 48. TorchBench имеет открытый исходный код и благодаря широкому множеству репрезентативных моделей позволяет, например, выявлять и ситуации с понижением производительности при работе PyTorch на GPU.

Сравнение произведено в 32-битной конфигурации TorchBench (используемой по умолчанию). Имеющаяся в A100 поддержка TF32 дает повышенную производительность за счет уменьшения точности, поэтому не все модели могут его использовать. Для глубокого обучения и вывода обученной нейронной сети с FP32 в одних моделях быстрее был MI210, в других – A100, что не дает четкого преимущества по производительности одному из этих GPU [335].

И вообще MI210 сразу стал применяться для задач, связанных с ИИ. Например, в [336] разработана методика для перекрытия коммуникаций и вычислений, ориентированная на DLRM, и создано программное ядро для MI210. В [337] предложен и реализован на MI210 усовершенствованный алгоритм для повышения производительности сверточной нейронной сети (CNN), который с использованием библиотеки глубокого обучения MIOpen и исходных данных ResNet50 позволил получить 74% от пиковой производительности MI210 с одинарной точностью.

5.3.2.2. Производительность MI250/MI250X. Что касается GPU MI250/MI250X, содержащих по 2 GCD, то потенциальная их конкуренция с A100 по производительности была продемонстрирована самими фирмами-разработчиками. В июне 2022 года Nvidia привела данные, демонстрирующие более высокую производительность (и энергоэффективность) A100-SXM-80G относительно MI250 в серверах с одним и четырьмя GPU (считая MI250 как один GPU) на классических приложениях молекулярной динамики AMBER, GROMACS, LAMMPS, NAMD и OpenMM [338], а в августе 2022 г. AMD привела противоположные по производительности

данные для всех этих приложений кроме GROMACS – но показав вместо данных для этой программы преимущество в производительности на тесте HPCG [**339**]. Понятно, что такое сопоставление требует дополнительных данных об использованных версиях приложений и SDK фирм, применявшихся в расчетах молекулярных систем и видов потенциалов, и так далее. Ниже в анализе производительности MI250/MI250X приводятся и данные из публикаций о производительности в молекулярной динамике.

То, что производительность MI250X высока, очевидно следует из успехов GPU AMD в Top500, где суперкомпьютеры Frontier и LUMI занимают первое и третье места с производительностью по HPL 1194,0 и 309,1 PFLOPS соответственно, а по HPCG – 14,1 и 3,4 PFLOPS соответственно [**4**]. Достаточно широкое сопоставление производительности 8 разных приложений из разных областей HPC (в основном не из числа массово используемых в мире HPC) на Frontier и Summit проведено в [**340**], где показывая преимущества Frontier дается и общая стартовая сравнительная оценка производительности MI250X и V100.

Но поскольку такие успехи на суперкомпьютерах во многом связаны с высоким уровнем масштабирования с числом узлов, приведем здесь еще только одну иллюстрацию, основанную на сравнении производительности узлов с A100 и с MI250X – суперкомпьютера Perlmutter и кластера Crusher (краткие данные об этих вычислительных системах имеются в таблице 23). Их узлы содержат по 4 GPU A100 и MI250X соответственно.

В [**341**] исследована производительность кода рентгеновской трассировки с использованием средств Kokkos (а также варианта с CUDA) на разном числе узлов Perlmutter и Crusher (там в качестве бэк-эндов применялись соответственно CUDA и HIP). Большинство вычислительного времени на GPU там занимает работа программного ядра nanoBraggSpots; использование Kokkos позволило добиться повышения производительности nanoBraggSpots более чем на 60% на узел Crusher по сравнению с Perlmutter [**341**]. Данные рисунка 2 этой статьи показывают, что производительность nanoBraggSpots при равном числе узлов от 32 до 128 в разы больше на Crusher, а применение Kokkos дало заметно более высокую производительность, чем обычное применение CUDA.

Но далее надо иметь в виду, какое сопоставление GPU Nvidia и AMD интересно в первую очередь. AMD MI250 и MI250X имеют по два логических GPU (по 2 GCD), а A100 – это один GPU. С точки зрения программирования MI250 или MI250X – это два GPU, и интересно сопоставление A100 с одним GCD, что часто (но не всегда) и делалось в публикациях. И тут же встает вопрос о стоимостях – если они у A100 и MI250X сопоставимы, интересно сравнение с целым MI250X. Но здесь сравнение стоимостных показателей или TCO не проводится (оно не только зависит от места приобретения и применения GPU, но и меняется со временем).

Базовые тесты производительности и производительность математических библиотек. Данные наиболее важных базовых тестов производительности для узлов Frontier и Crusher приведены на сайте Окриджской национальной лаборатории США, которой принадлежат и эти суперкомпьютеры, и Summit [342]. Некоторые из этих данных относятся и к ЦП узлов (EPYC 7A53). Для работы с MI250X (с одним GCD) там использованы ROCm 5.3.0.

В [342] имеется информация о достигнутых пропускных способностях памяти MI250X, обменов данными между ЦП и GPU, и между GPU. Там представлены данные о пропускной способности во всех пяти компонентах теста BabelStream (классического варианта с FP64) с применением HIP и OpenCL в зависимости от размерности одномерных массивов теста.

По данным [342], достигнутая в BabelStream пропускная способность HBM составляет 77–86% от пиковой величины (для ЦП с его DRAM достигается 82–90%). Наивысшие величины достигались здесь при работе copy (чуть меньше – в mul) -компоненты BabelStream с применением hipcc, максимум около 1,38 ТБ/с. Но в некоторых компонентах BabelStream применение OpenCL давало более высокую пропускную способность, чем hipcc, а с тестом скалярного произведения векторов (dot) hipcc (в отличие от OpenCL) дал аномально низкую пропускную способность.

Пропускная способность памяти MI250X (одного GCD) в [324] изучалась также при выполнении SpMV из библиотеки Ginkgo с применением разных форматов хранения разреженных матриц.

Пропускная способность ЦП-GPU (измерялась пропускная способность hipMemcpy) в [342] составила 25,6 ГБ/с (71% от пиковой), а между GPU – в MPI-тесте osu_bw из микротестов OSU [343] (с использованием Cray MPICH 8.1.23) – от 37.6 ГБ/с (75.2% от пиковой) на одном канале Infinity Fabric до 145.3 ГБ/с (72.7% от пиковой) на четырех каналах Infinity Fabric [342]. В [342] последние измерения относятся к односторонней передаче – а в таблице 19 этого обзора приводится пиковая величина 100 ГБ/с на один канал для двусторонних передач.

В [342] имеются также данные о производительности MI250X на GEMM (с использованием hipBLAS), включая и DGEMM. Для GEMM с FP16 там использован конкретный тест CORALGEMM [344], производительность которого росла вплоть до наивысшей размерности матриц свыше 8 тысяч. Для FP32 и FP64, кроме этого теста, в [342] применялся еще другой конкретный тест gpu_xgemm [345] (см. рисунок 22).

Выбор того или иного конкретного теста не дает здесь принципиальных изменений производительности – она в основном определяется, естественно, выбором FP32 или FP64. Никаких выдающихся результатов (типа достижения 90% и более от пикового значения) применение hipBLAS здесь не дает. Но интересно, связаны ли здесь скачки на кривых производительности от размерности матриц с «оптимальной» ее четностью.

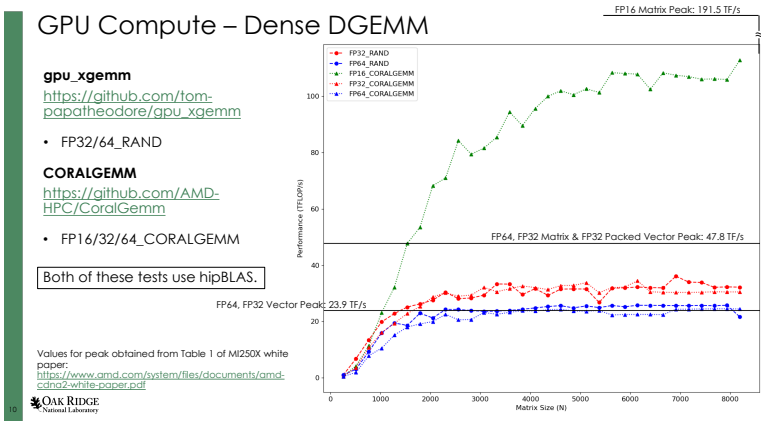


Рисунок 22. Производительность GEMM на одном GCD MI250X (рисунок из [342])

В [342] представлены еще данные теста mixbench, но эти результаты, как и выше в разделе для A100, здесь не рассматриваются, поскольку они более интересны скорее для задач ИИ.

В [333] представлены данные о производительности MI250 в тестах HPL (с rocHPL 6.0.0), HPL-AI (с HPL-AI-1.0.0) и HPCG 3.0 с базированием на ROCm 5.2.0. На одном GPU достигнутая производительность составила 40,45 TFLOPS (около 90% от пиковой векторной величины), а на 4 GPU — 161,97 TFLOPS. Отметим, что в [333] результаты относятся к полным GPU с двумя GCD. На HPL-AI (модуль с FP16/32/64) на 4 GPU производительность достигла 930,44 TFLOPS. На HPCG было получено 488,8 GFLOPS для одного GPU и 1927,7 TFLOPS на 4 GPU.

Вычислительная химия. Учитывая, что активное сопоставление AMD и Nvidia производительности MI250X и A100 происходило во многом на задачах молекулярной динамики, изложение здесь начнем именно с этой области вычислительной химии. Здесь сразу следует отметить, что достижение высокой производительности при переносе приложений молекулярной динамики с GPU Nvidia на GPU AMD является непростой задачей. Перенос таких программных комплексов с CUDA на HIP рассмотрен в [314] (хотя там не рассматривались анализируемые в данном обзоре модели GPU AMD).

AMD приводит свои данные о достигнутой производительности приложения молекулярной динамики LAMMPS 2022 в нескольких известных тестах этого приложения, полученные на GPU MI250 и MI210, и сопоставляет эти данные с результатами на A100 [346]. Один MI250 обгоняет A100 по производительности на всех тестах (в то время как

MI210, содержащий один GCD, чаще отстает от A100-PCIe-80GB). AMD продемонстрировала также хорошее масштабирование производительности LAMMPS вплоть до 4 GPU на сервере [346].

В [254] проведено подробное и глубокое исследование, направленное на повышение производительности LAMMPS после портирования его с аппаратных средств GPU Nvidia (в статье использованы Summit с V100 и вычислительная система AFW HPC11 с A100-40GB) на GPU MI100 (в кластере Spock) и MI250X (в кластере Crusher). В LAMMPS для поддержания работы с разными аппаратными средствами используется библиотека Kokkos. Для оптимизации производительности (она изучалась для нормального формата FP64) LAMMPS на GPU Nvidia и AMD применялись модели линии крыш для каждого программного ядра, а в оценках достигаемой производительности было задействовано 6 разных потенциалов и расчеты жидких, металлических, гранулированных, биологических и полимерных систем размером до 55 миллионов атомов.

Поддержка работы LAMMPS на GPU с ориентированным на дальнодействующее взаимодействие потенциалом PPPM была реализована в [254] с использованием БПФ, для чего на GPU AMD использованы средства rocFFT. При работе с потенциалом ReaxFF на молекулярной системе с 256 тысячами атомов было достигнуто около 20% (5,6 TFLOPS) от пиковой производительности FP64 одного GCD в Crusher.

В [254], несмотря на большой проделанный объем тщательного и широкого исследования, указано на полученные оценки производительности как на предварительные, что связано, вероятно, с отмеченными в этой статье активными разработками ROCm (в работе использовались ROCm v4.5.0).

В [254] указано на различное поведение достигаемой производительности при изменении параметров Kokkos для GPU AMD и Nvidia. Это можно считать иллюстрацией сложности переноса программного обеспечения с одного типа GPU на другой с сохранением эффективной оптимизации кода.

В [262] анализируется производительность для квантового потенциала SNAP (в части EXAALT-проекта ECP), используемого затем в рамках работы с LAMMPS. Хотя больше информации о производительности здесь было получено для MI100, которого MI250X, кончено, сильно опережал, один GCD MI250X в самом начале 2021 года в этом расчете существенно отставал даже от V100, а от A100 – гораздо больше. Это было связано с нехваткой для SNAP емкости кэша L1 в CU MI250X – в V100 и в A100 для SNAP использовалась емкость в 96 КБ на 1 SM, в 6 раз больше, чем в MI100 и MI200 на 1 CU (см. таблицы 20 и 21).

Работа над оптимизацией программных ядер SNAP продолжаются [262], но хотя это были скорее предварительные результаты, дальнейший прогресс предполагается в первую очередь в части использования здесь средств Kokkos. В [262] указано, что частота попадания в кэш L1 у SNAP составляет (для самого большого программного ядра SNAP) 66% для MI250X против 90% для V100.

В [347] AMD продемонстрировала хорошую масштабируемость NAMD – приложения молекулярной динамики на известных тестах производительности APOA1 и STMV NVE при использовании до 8 MI250.

Выше в самом начале раздела 5.3.2.2 было указано об отсутствии предоставленных в 2022 году данных AMD о производительности GROMACS на MI250/MI250X, что было связано с временными проблемами портирования кода GROMACS, и было быстро исправлено.

В [333] имеются данные о производительности MI250 на наиболее известных приложениях молекулярной динамики AMBER, GROMACS, LAMMPS и NAMD. Приведем ряд примеров. Для GROMACS в известном тесте STMV (Satellite Tobacco Mosaic Virus) на одном MI250X получена производительность 34,2 нс/день, а на 4 MI250X – 89,26 нс/день. Для NAMD 3.0 в известном тесте STMV NVE на одном MI250X получена производительность 19,87 нс/день, а на 4 MI250X – 77,132 нс/день.

В версии GROMACS 2023.1 в качестве бэк-энда для MI200 применяется SYCL, что показало хорошее масштабирование достигаемой производительности в тесте STMV при использовании до 8 MI250X. Обнаружено, что пока не все возможности CDNA 2, которые могут давать существенный рост производительности GROMACS, используются компилятором AMD [348].

Для AMBER22 здесь представлены данные о производительности на 8 известных тестах AMBER. Так, на тесте Cellulose Production NPT 4fs была получена производительность 227,2 нс/день. Мы сопоставили данные [333] с результатами, доступными в [349], и объединили эти данные в таблице 24.

Таблица 24. Производительность (нс/день) AMBER22 на разных GPU

Тест	Производительность A100-PCIe[1]	Производительность MI250[2]	Производительность H100-PCIe[1]
JAC Production NVE 4fs	1199,22	1871	1479,32
JAC Production NPT 4fs	1194,5	1794	1424,90
STMV Production NPT 4fs	52,02	80,65	70,15

[1] данные [349] (данные на 21.03.2023). Использованы Amber 22 Update 1; AmberTools 22 Update 1; Nvidia CUDA 11.4

[2] данные [333] с ROCm 5.2.0, Amber container 22.amd_100

Но здесь необходимо привести важные уточнения о сопоставлении данных о производительности различных приложений молекулярной динамики на различных GPU. Приводимые данные о расчете касаются в первую очередь рассчитываемой молекулярной системы (где важно общее число атомов), что определяется самим названием теста. Используемый потенциал в тесте также фиксирован. Но время расчета зависит еще и от шага по времени (часто измеряется в фемтосекундах, fs в таблице 24), и радиуса отсечки взаимодействий. Поэтому иногда приводимые данные о достигнутой производительности могли оказаться не сопоставимыми. По крайней мере времена шага в источниках данных таблицы 24 совпадают.

Но надо также иметь в виду, что в этой таблице данные из [**349**]) получены с далеко не новой версией CUDA; сама AMD сопоставляла в [**333**] с CUDA 11.6, старая версия CUDA возможно недостаточно эффективна для H100 – в цитируемых в этом обзоре публикациях применялась и CUDA 12.2. И все-таки эти данные, как и приведенные выше в данном обзоре, говорят о том, что MI250 имеет реальную конкуренцию по производительности в молекулярной динамике с A100 (и может быть с H100), возможно, и опережая по производительности, а зависимость от версий используемых SDK очевидна.

Еще одна статья, [**350**], в которой исследована производительность MI250, находится «на стыке» молекулярной динамики и ИИ. В [**350**] с применением GPU V100, A100 и MI250 проведено довольно большое количество вычислений с использованием пакета программ DeePMD-kit для моделирования молекулярной динамикой с применением потенциалов машинного обучения (вместо применения для этого метода QM/MM). DeepMD-kit позволяет интегрировать эти потенциалы с приложениями LAMMPS, Amber, OpenMM и GROMACS. Нами на основании таблицы IV из [**350**] проведены вычисления ускорений расчетов на A100-80GB по сравнению с одним GCD MI250. Для расчетов молекулярной динамикой по LAMMPS для FP64 они составили от 6% до 2,5 раз. Отсутствие в [**350**] деталей об использованных версиях программного обеспечения (например, о перенесенной на GPU AMD версии LAMMPS, использованной для расчетов молекулярной динамикой) позволяет отнести эти величины скорее к предварительным оценкам, но все-таки дающим определенную информацию о производительности MI250.

Работу [**351**] также можно отчасти (в определенном математическом смысле) считать имеющей интерес и для задач молекулярной динамики – здесь для вычисления евклидова минимального остовного дерева по усовершенствованному авторами алгоритму использованы A100 (с NVCC 11.5) и MI250X (1 GCD, с ROCm 4.5) и проведены расчеты по 12 разным наборам данных. По полученным данным о производительности на рисунке 6 из [**351**] мы рассчитали относительное ускорение A100 по сравнению с 1 GCD, оно лежит в диапазоне 1,5–1,7.

Вообще складывается впечатление, что стабильные надежные данные о сопоставлении производительности MI200 с GPU Nvidia в задачах молекулярной динамики пока скорее отсутствуют (в смысле того, что появляются все новые улучшенные показатели).

В [352] исследованы данные по достигаемой производительности в рамках проекта NWChemEX в ECP с использованием MI250X и A100 в узлах суперкомпьютеров Crusher и Polaris соответственно (см. таблицу 23). NWChemEX ориентирован в том числе на расчеты высокоточным квантовохимическим методом CCSD(T), в котором основное время расчета уходит на являющуюся возмущением к CCSD поправку за счет тройных возбуждений (T), требующую $O(n^7)$ вычислений, где n-размерность базиса. Программное ядро для расчета этой поправки было реализовано в разных вариантах с применением HIP, CUDA и DPC++; для этого использовались компиляторы clang-16, gcc-10.3.0 и CUDA-11.4.4/ROCm-5.1.0 (для Polaris/Crusher соответственно), а для распараллеливания между узлами использовали Cray-mpich 8.1.16.

В таблице 25 приведены полученные времена расчета поправки T для молекулярного фрагмента убиквитина (это данные таблицы III в [352]) на одном GPU. Из этих данных следует, что один GCD в MI250X считал чуть быстрее, чем A100-40GB (как на HIP против CUDA, так и с применением SYCL), а на двух GCD производительность MI250X возрастала в 1,9 раза. При этом достигаемая с применением DPC++ производительность оказалась довольно близка к полученной при использовании CUDA или HIP.

Однако в [352] на A100 было также найдено, что в SYCL-реализации программное ядро для расчета T предъявляет очень высокие требования к кэшам L1 и L2, и сгенерированный PTX показал большое количество загрузок и сохранений в локальную (частную в терминах SYCL) память. Добавление одного ключа clang дало рост производительности в 2,5 раза. Применение другой #pragma директивы clang увеличило производительность еще на 20%. Но в этом PTX-коде не использовались FMA операции «умножить-и-сложить», а после добавления другого ключа clang дополнительное улучшение производительности было еще примерно на 20% [352]. Результаты в таблице 25 оптимизацию включают.

Таблица 25. Время расчета поправки T (в секундах) на GPU [352]

GPU	Модель программирования		
	SYCL	CUDA	HIP
MI250X (1 GCD)	17,41	—	15,56
MI250X (2 GCD)	8,97	—	8,12
A100	18,23	16,14	—

Кроме того, в [352] продемонстрирована масштабируемость вплоть до большого числа узлов в Crusher и Polaris, которая сильно зависит от использованных типов базиса.

В [353] с применением еще «пилотного» состояния суперкомпьютера LUMI-G (подкластер LUMI с GPU MI250X в узлах) проведены расчеты с использованием известного квантовохимического приложения CP2K с усовершенствованными авторами методиками для обменной части метода Хартри-Фока и корреляционных методов с периодическими граничными условиями, но с применением гауссовского базиса, дающего возможность пользоваться поблочной разреженностью. Была получена хорошая масштабируемость производительности и эффективность распараллеливания вплоть до 32 узлов LUMI-G.

Решеточная квантовая хромодинамика (LQCD). В [354] исследована производительность доступного на GitHub приложения SIMULATeQCD для квантовой хромодинамики при работе на одном или нескольких узлах (multi-GPU серверах) кластерных систем, в том числе Perlmutter и Frontier (см. таблицу 25). Хотя данные о производительности при работе с MI250X здесь, как и во многих других публикациях, отнесены к разряду предварительных, представляет интерес проведенное здесь сопоставление с производительностью A100-40GB (хотя понятно, что при использовании более современных версий ROCm и дополнительной оптимизации программного кода достигаемая производительность может существенно повыситься в ближайшие времена).

В качестве бэк-энда в SIMULATeQCD применяются программные коды с CUDA или HIP. В [354] приведены данные о производительности при использовании до 256 A100 и GCD; мы ограничимся здесь данными тестов производительности оператора Дирака HISQ с масштабированием в пределах одного узла (см. таблицу 26).

Таблица 26. Сильная и слабая масштабируемость производительности (в TFLOPS) оператора Дирака HISQ в узлах Perlmutter и Frontier

GPU в узле	Число GPU (число GCD для MI250X)	Производительность (сильная масштабируемость)	Производительность (слабая масштабируемость)
A100-40GB	1	—	1,35746
	4	5,06892	4,53759
MI250X	1	—	0,92974
	2	1,63049	1,51439
	4	3,00675	2,89454
	8	4,69513	5,57654

Для сильного масштабирования использовалась глобальная решетка 96^4, для слабого масштабирования – локальная решетка 32^4. Данные этой таблицы взяты из [354].

Хотя результаты для MI250X и были указаны как предварительные, достигнутая производительность оказалась ниже ожидаемой авторами на основании спецификаций MI250X. Вообще здесь в качестве первого приближения можно сказать, что один GCD по производительности несколько отстает от A100, а полный MI250X (2 GCD) его несколько опережает.

В [**355**] сравнена производительность на узлах суперкомпьютеров Big Red 200 (по 4 A100-40GB на узел) и Crusher (по 4 MI250X на узел) с применением известной программы QUDA для решеточной квантовой хромодинамики. Из-за плохого масштабирования с числом узлов в обоих системах укажем только на данные о производительности для одного узла – узел с MI250X (с восемью GCD) был на 16% быстрее узла с A100. Преимущество MI250X в [**355**] было объяснено связанной с памятью производительностью QUDA при двукратном опережении MI250X по пропускной способности памяти по сравнению с этой моделью A100 (см. таблицу 21).

Вычислительная аэро- и гидродинамика (CFD). Анализ данных о производительности MI250X и MI250 в CFD-приложениях начнем с данных AMD [**333**] о производительности MI250 в стандартном тесте HPC Motorbike для CFD-приложения OpenFOAM (662,3 с на одном MI250 и 209,84 с – на четырех).

В [**356**] для численного моделирования реальных устройств сгорания использован специальный решатель PeleLMeX, производительность которого исследована на Crusher с MI250X и Summit с V100. При небольшом числе используемых GPU MI250X найден в полтора раза более высокопроизводительным, чем V100.

В [**357**] были проведены расчеты в области динамики многочастичных столкновений для описания гидродинамических взаимодействий на основе частиц, с использованием библиотеки Cabana 1.0-dev на основе Kokkos 3.5.00, с применением A100 и MI250. Сопоставление multi-GPU серверов, содержащих 4 A100 или 4 MI250 показало, что при меньших размерах рассчитываемой системы MI250 быстрее на 7%, а на большем размере A100 быстрее на 19% (по данным из рисунка 3 в [**357**]). Это позволяет говорить о сопоставимости производительности полного (с двумя GCD) MI250 и одного A100.

Очень интересная и актуальная статья с сопоставительными данными по производительности MI250X, MI100, A100 и V100 [**325**] относится решению уравнений Навье-Стокса для сжимаемого потока с использованием конечно-разностной дискретизации – для турбулентных течений, ограниченных стенками. Там была рассмотрена реализация переносимого на ЦП x86-64, GPU AMD и Nvidia приложения STREAmS-2, разработанного на основе базировавшегося на объектно-ориентированном (с применением Fortran 2008) приложении STREAmS-1.

Для работы с GPU от Nvidia в [**325**] используются средства CUDA Fortran (HPC SDK 22.11), а с GPU от AMD – HIPFORT (ROCm 4.5.2 на MI100 и ROCm 5.0.2 на MI250X). Для портирования кода STREAmS из CUDA Fortran в HIPFORT в [**325**] были созданы собственные средства PyconvertSTREAmS, поскольку GPUFORT, по оценкам авторов [**325**], находился скорее на еще исследовательской стадии разработки. Но для оптимизации полученный в PyconvertSTREAmS код также может требовать последующей ручной доработки.

Расчеты с анализом масштабирования производительности на нескольких узлах были проведены в [**325**] в кластерах *EuroHPC JU*: для MI250X – на LUMI, для A100 – на Leonardo (см. таблицу 23).

При сравнении расчетов на одном GPU в MI250X использовался один GCD; данные о соответствующих временах вычисления по двум различным расчетным схемам в STREAmS-2 приведены выше на рисунке 20. Размеры использованной в расчетах сетки были выбранными максимальными для возможного размещения в емкости памяти V100 [**325**].

Данные рисунка 20 указывают на близость достигнутой производительности у одного GCD MI250X и V100; MI100 был существенно медленнее V100, а A100 – существенно быстрее одного GCD. Как отмечено в [**325**], это не соответствует соотношениям пиковой производительности (с FP64) этих GPU, и там был проведен более тщательный анализ времен выполнения отдельных программных ядер, который показал аналогичные результаты по программным ядрам с однозначным лидерством A100 по производительности. В [**325**] приведены и примеры неожиданного увеличения достигавшейся производительности GCD при небольших изменениях в программных ядрах.

Проведенный в [**325**] анализ с применением модели линии крыши (с учетом только HBM) обнаружил, что расчеты на GCD в соответствии с ожиданиями для CFD были всегда связаны памятью, а на A100 часто оказывались в вычислительно интенсивной области. Было найдено, что перемещение данных для GCD значительно выше, чем для A100, значит A100 извлекает из регистров и кэширует чаще, чем GCD. Кроме того, в [**325**] указано на большое использование регистров в применявшейся взвешенной существенно неосциллирующей расчетной схемы (WENO).

К вышеописанным данным [**325**] целесообразно добавить дополнительные комментарии. Что касается данных о более высокой производительности V100 относительно MI100, это выглядит скорее естественным (об этом указывалось выше в разделе о производительности MI100). Тем не менее, V100 бывает в двух моделях [**185**] – с памятью емкостью 16 ГБ и (появившаяся позднее) с 32 ГБ, а используемая в [**325**] модель V100 с 16 ГБ при работе с большим размером сетки (но помещающимся в 32 ГБ MI100) будет просто неприемлемой.

Что касается MI250X, то, во-первых, предположение о возможно более слабом кэшировании соответствует отмеченным выше недостаткам кэш-памяти MI250X по сравнению с A100 (см. также таблицы 20 и 21). Во-вторых, в [**142**] указывается на недостаток компилятора AMD (в ROCm v4.5.2) по сравнению с компилятором Nvidia (в CUDA v11.0.3): CUDA использует значительно меньше регистров на варп, чем компилятор AMD, и обеспечивает гораздо более высокую загрузку варпов на SM, что способствует более высокой производительности GPU Nvidia.

В-третьих, актуально и сопоставление производительности полного MI250X (у наверняка более дешевого MI250 производительность должна быть близка) с A100, которое в [**325**] не проводилось. Если сделать естественное предположение о хорошей масштабируемости STREAmS-2 в рамках целого MI250X, то поделив время расчета с MI250X на рисунке 20 на два в качестве начального приближения, будут получаться уже сопоставимые с A100 времена расчета. Кроме того, надо иметь в виду и отсутствие достаточной информации об уровне достигаемой оптимизации при работе с HIPORT. Все это никак не противоречит факту достижения в STREAmS-2 более высокого процента от пиковой производительности A100 по сравнению с GCD.

В [**325**] получены также данные, показывающие высокую масштабируемость производительности STREAmS-2 – например, эффективность более 80% при использовании до 16 узлов Leonardo и до 32 узлов LUMI, и хорошие результаты и на большем числе узлов, дающие STREAmS-2 высокий потенциал для приложений динамики сжимаемых жидкостей.

В лаборатории Argonne National Lab [**144**] проведены исследования производительности приложения Nek5000/RS на ряде суперкомпьютеров, использующих GPU MI250X, A100 и V100 в узлах с масштабированием от одного GPU до многих узлов. Расчеты проводились по модернизированному варианту Nek5000 для работы с GPU (NekRS версии 22.0) в рамках входящего в ECP проекта ExaSMR для генерации наборов данных виртуального моделирования ядерного реактора с высокоточными соединенными физическими моделями явлений в реакторе. В таблице 27 представлены

Таблица 27. Данные о производительности NekRS на одном GPU с использованием CUDA и HIP для GPU Nvidia и AMD соответственно

Система	Устройство (device)	Относительная производительность
Summit	V100-16GB	1,00
ThetaGPU	A100-40GB	1,57
Perlmutter	A100-40GB	1,62
Polaris	A100-40GB	1,62
Spock	MI100-32GB	0,84
Crusher	MI250X-64GB (1 GCD)	1,32

данные о производительности NekRS для одностержневого моделирования на одном GPU.

В соответствии с этой таблицей, один GCD на Crusher обеспечивает 1,32-кратный прирост производительности решения уравнения Навье-Стокса по сравнению с одним V100 на Summit. A100 по производительности опережает один GCD, и в целом по данным [**144**] производительность одного GCD составляет около 85% от одного A100.

В [144] было проведено, в частности, и сопоставление масштабирования производительности на Frontier (с ROCm 5.1.0 и Cray-mpich 8.1.17) и Crusher (с разными версиями SDK – ROCm 5.1.0, ROCm 5.2.0, Cray-mpich 8.1.16 и Cray-mpich 8.1.19). В Crusher ROCm 5.1.0 дал производительность на 2–5% быстрее, чем ROCm 5.2.0, а производительность на Frontier была лучше, чем на Crusher.

В [358] для задач моделирования крупных вихрей изучена производительность кодов NekRS и AMR-Wind для моделирования течений в пограничном слое атмосферы с использованием суперкомпьютеров Summit и Crusher (с узлами, содержащими V100 и MI250X соответственно) в вариантах сильного и слабого масштабирования. Для NekRS было показано, что один GCD MI250X на Crusher обеспечивает производительность, сравнимую с одним V100 на Summit.

В [142] представлены данные о производительности MI250X (в сопоставлении с MI100 и V100) с применением базирующегося на NekBone теста HipBone (прокси-приложение HipBone рассматривалось выше в разделах про производительность A100 и MI100), использующего библиотеку конечных элементов libParanumal (разрабатываемую в рамках ECP) со средствами OCCA для абстракции между различными моделями параллельного программирования – например, OpenMP, OpenCL, CUDA, HIP и SYCL. В HipBone, как и в Nekbone, в качестве сетки используется структурированный куб из однородных шестигранников.

Расчеты в [142] проводились на вычислительных системах Summit, Spock и Crusher (см. таблицу 23) с использованием в их узлах CUDA 11.0.3, ROCm 4.5.0 и ROCm 4.5.2 соответственно. Тестирование проводилось на одном GCD, что позволяет потенциально использовать более половины общей вычислительной мощности MI250X и более высокие тактовые частоты, чем при одновременном выполнении одной и той же рабочей нагрузки на обоих GCD. Однако существенной разницы в производительности между программными ядрами, выполняемыми на одном GCD или на обоих GCD в MI250X одновременно не наблюдалось. Было также найдено, что компилятор Nvidia использует значительно меньше регистров на варп по сравнению с компилятором AMD, что обеспечивает гораздо более высокую загрузку варпов на SM.

В [142] проведено также подробное исследование сильного и слабого масштабирования в использовавшихся вычислительных системах, которое здесь не обсуждается.

В [359] приведены данные о производительности на MI250X, MI100, V100 и A100 приложения NekRS и прокси-приложения HipBone. Для расчетов применялись вычислительные системы Summit, Spock, Crusher и ThetaGPU (см. таблицу 23).

Производительность оператора Пуассона (как и у других операторов в сеточных методах – расчета результата воздействия оператора на функцию в точках сетки), определяющая время расчета HipBone, для разных степеней использующегося полинома у одного GCD на 20–30% больше, чем у V100 (см. рисунок 43 в [**359**]).

Все устройства достигали наивысшей производительности HipBone при наивысшей степени использовавшегося полинома (15) – у V100 2101,4 GFLOPS, у MI100 – 2135,2 GFLOPS, у одного GCD 2774,9 GFLOPS [**142**,**359**]. Для NekRS один GCD на Crusher на разных программных ядрах давал от 57% до 88% от производительности A100, по полному времени расчета – 71% [**359**].

Выше был рассмотрен ряд публикаций, касающихся производительности ориентированных на работу с GPU программных средств NekRS и последующих NekBone и HipBone. Они все ориентированны на обеспечение их переносимости между разными типами GPU, используют средства C++ (хотя у NekBone имеется и версия с CUDA Fortran [**358**]), но базируются на использовавших Fortran Nek5000, и активно вовлечены в работы в рамках ECP.

Но у этого направления с применением C++ есть альтернатива, которая также базируется на Nek5000 и ориентирована на переносимость на разные типы ускорителей, и она также использовалась в исследованиях достигаемой производительности с применением рассматриваемых в обзоре GPU. В [**171**] для области прямого численного моделирования турбулентности в приложениях устойчивого судоходства проведено моделирование обтекания ротора Флеттнера (при Re=30000) и его взаимодействия с турбулентным пограничным слоем на кластерах с multi-GPU серверами в узлах, использующими A100 и MI250X.

Для этого в [**171**] были использованы и модернизированы также базирующиеся на Nek5000 программные средства Neko для работы с неструктурированными сетками. В Neko используются объектно-ориентированные возможности современных стандартов Fortran для управления распределением памяти и обеспечения многоуровневых абстракций стека решателя, что дает возможность вычислений на различных архитектурах от обычных ЦП до разных типов ускорителей.

В [**171**] уровень абстракции устройства применяется для управления памятью устройства, передачей данных и запуском ядра из Fortran, и разработаны бэк-энды на CUDA и HIP. Расчеты здесь проводились на кластере Alvis, имеющем в узлах по 4 A100-SXM4 (использовалась версия CUDA 11.1.1) с межсоединением Mellanox ConnectX-6 (2×400 Гб/с) – с применением OpenMPI 4.0.5, и на кластере HPE Cray EX, имеющем в узлах по 4 MI250X (использовалась версия ROCm 4.5.2) с межсоединением HPE Slingshot 10 – с применением Cray MPICH 8.1.14. На хосте в Alvis применялся gcc 10.2, в кластере HPE – Cray cce 13.0.

В хостах обоих кластеров имелось по 512 ГБ памяти DDR4 (с двумя Intel Xeon Gold 6338 на узел Alvis и с AMD EPYC 7A53 на узел HPE).

Расчеты по Neko были связаны памятью, что, по мнению авторов, дает возможность вычислительно-связанным приложениям получить преимущества благодаря более высокой пиковой производительности FP64 в MI250X [171]. В [171] сделан вывод, что два GCD в MI250X соответствуют двум A100 с точки зрения производительности. Среднее время на временной шаг отличается менее чем на 5%, если сравнивать два A100 с одним MI250X (см. рисунок 23). Но это контрастирует

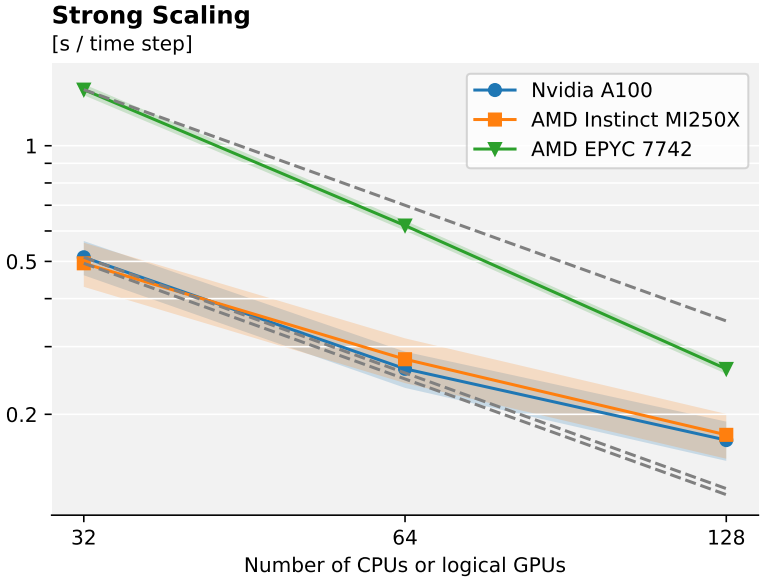


Рисунок 23. Сильное масштабирование. Производительность по времени за временной шаг: заштрихованные области относятся к стандартному отклонению; оранжевая и синяя линии представляют системы с GPU; логических GPU в каждом MI250X по два (рисунок из [171])

с обсуждавшимися выше данными, например [359], что один GCD MI250X имеет производительность ближе к 71% от производительности одного A100.

Ряд публикаций, где получены данные о производительности MI250 или MI250X, относятся к магнитогидродинамике. В [360] разработан новый код Idefix для нерелятивистской гидродинамики и магнитогидродинамики, основанный на Kokkos, который в тестах для магнитогидродинамики на сервере с четырьмя MI250X показал производительность в 2,57 раза

быстрее, чем на сервере с четырьмя V100. В [360] показана также гораздо более высокая энергоэффективность при работе с MI250, чем при использовании ЦП.

В [361] проведены расчеты по программе Athena++ (точнее, использовался код AthenaK для работы с GPU) для общерелятивистской магнитогидродинамики с применением узлов Crusher (с четырьмя MI250X на узел) и Polaris (с четырьмя A100 на узел). В сопоставлении с одинаковым числом логических GPU (т. е. сравнивая число GCD для MI250X с числом A100) Polaris, грубо говоря, в пару раз быстрее при любом числе логических GPU (см. рисунок 32 в [361]), и при этом масштабирование в обоих вычислительных системах хорошее. Это позволяет предположить, что при сопоставлении A100 с целыми MI250X они по производительности конкурентоспособны.

В [362] приводятся данные о производительности MI250X, MI100, A100-40GB и V100-16GB в кластерных системах с использованием PARTHENON-HYDRO – мини-приложения (на основе кода астрофизической магнитогидродинамики Athena++), состоящего из около 1,4 тысячи строк на C++ для 1D, 2D и 3D сжимаемой гидродинамики в многоуровневых сетках.

Полученная производительность (здесь рассматриваемая просто как некоторая относительная величина) составляет 5,7 – для MI250X (два GCD, с ROCm 5.1.0); 4,2 – для A100 (с CUDA 11.5); 2,7 и 2,15 – для V100 и MI100 соответственно. Это означает, что полный MI250X сильно опережает по производительности V100 и MI100, и быстрее A100 (хотя в расчете на один GCD MI250X отстает от A100). В [362] рассмотрены также данные о достигаемых сильной и слабой масштабируемости в использованных для расчетов системах, но это здесь не анализируется.

Физика плазмы. В принципе эта область также относится к CFD, но здесь выделена в отдельную часть, так как по физике плазмы появился целый ряд публикаций, в которых применялись MI250/MI250X и были получены интересные данные о производительности.

Так, в [363] была показана более высокая производительность MI250 по сравнению с V100 при решении кинетического уравнения Власова для динамики плазмы заряженных частиц.

В [364] исследована производительность кода CGYRO, который решает пятимерные гирокинетические уравнения Максвелла, описывающие эволюцию микротурбулентности плазмы в устройствах магнитного термоядерного синтеза. Для работы с GPU (MI250X во Frontier и A100 в Perlmutter) там были использованы средства OpenACC (из HPE Cray Fortran для MI250X и из Nvidia Fortran для A100), а также библиотеки cuFFT и hipFFT для A100 и MI250X соответственно.

В начальном тесте CGYRO на базе узла с MI250X был значительно медленнее, чем на базе узла с A100, но после оптимизации стал быстрее с MI250X. Но в [**364**] сопоставлялись расчеты при одинаковом числе узлов и GPU в этих суперкомпьютерах, соответственно быстрее A100 оказался полный MI250X из двух GCD.

В [**365**] время расчета по приложению HiPACE++ для моделирования плазменных ускорителей квазистатическим алгоритмом частиц в ячейках было сопоставлено для A100-80GB и MI250X (один GCD). В этом случае все основные данные для расчета во время вычисления целиком располагаются в памяти GPU. На A100 применялись средства CUDA 11.0, а для расчета на GCD (вероятно, с HIP) использованы возможности раннего тестового доступа к Crusher. В вычислениях использовался FFT, а rocFFT, по данным [**365**], тогда отличался плохой производительностью при размере сетки, не являющимся степенью двух. Время вычисления на GCD на двух разного типа стадиях расчета составляло от 0,7 до 1,6 и от 0,9 до 3,7 раз по сравнению с временем на A100.

AMD приводит приложение PIConGPU, также использующее алгоритм частиц в ячейках, в качестве примера приложения в области физики плазмы, адаптированного для работы на GPU MI200 путем применения бэк-энда ALPAKA (Abstraction Library for Parallel Kernel Acceleration), работающего поверх HIP/ROCm [**366**].

Метод частиц в ячейках вообще широко используется в приложениях физики плазмы, и в расчетах с применением MI250X он также применялся. В [**367**] исследовано 3D-моделирование взаимодействия лазера с веществом на массивно-параллельном коде PIC WarpX с использованием GPU на Frontier, Summit и Perlmutter (близкое к этому исследование проведено также в [**368**]).

В [**369**] проанализирована производительность использующего средства Kokkos линейного итерационного решателя TFQMR (в том числе более быстрого пакетного варианта), доступного в библиотеке PETSc (Portable Exensible Toolkit for Scientific Computing), что интересно для физики плазмы при работе с оператором столкновений Ландау. Производительность MI250X и A100 здесь была оценена при использовании узлов кластеров Perlmutter и Crusher.

Вышеупомянутые работы [**365**, **367**, **368**] имеют прямое отношение к проектам из ECP. Как и многие другие из используемых в разделе публикаций, здесь использованы полученные в только что появившихся вычислительных системах с GPU MI250/MI250X новейшие данные, которые часто относили к предварительным в связи с возможным быстрым прогрессом новых версий SDK от AMD.

Задачи искусственного интеллекта. Хотя AMD не ставила задачи ИИ на самое первое место для потенциального применения MI200, эти GPU сразу стали применяться и для ИИ. Так, в [**28**] использована интеграция HPC и глубокого обучения: для приложения Монте-Карло (для термодинамических расчетов в материаловедении) был разработан набор суррогатных моделей глубокого обучения и проведены расчеты на многих узлах суперкомпьютеров Summit с V100 (с применением стека CUDA) и Crusher с MI250X (с применением стека ROCm). На разных размерах моделей один GCD был быстрее V100 до 15% (а целый MI250X соответственно быстрее до 2,3 раз). Crusher относительно Summit показал также лучшее масштабирование производительности.

Эти GPU стали применяться для задач ИИ в самых разных областях, в которых используется ИИ. Так, в [**370**] MI250X использовался для задач сегментации митохондрий.

После развертывания европейского суперкомпьютера LUMI с MI250X, вероятно в связи с применением разных языков в Европе, там стали активно появляться публикации по обработке естественного языка на MI250X, в том числе с использованием собственных модифицированных моделей BERT (см., например, [**371**–**374**]).

5.4. Резюме по GPU AMD

Приведенные выше данные о производительности GPU MI200 – по большей части MI250 и MI250X – безусловно говорят об определенной конкурентоспособности этих GPU по отношению к A100. И AMD с ROCm, и разработчики приложений активно продвигаются вперед для достижения более высокой производительности.

Ясно, что уже представленные данные о производительности часто не соответствуют ожиданиям, базирующимся на более высоких пиковых производительностях этих GPU от AMD по сравнению с A100 (имеется в виду уже для одного GCD). Хотя во многих случаях тесты производительности были связаны памятью, а не вычислительно интенсивными в модели линии крыши, это все равно не коррелирует со многими отставаниями по производительности относительно A100, а также идентифицируемыми иногда «провалами» производительности MI250X (см., например, данные [**328**]).

В ряде публикаций это связывается с недостаточно большой емкостью кэш-памяти (в основном L1) относительно A100, и использованием в генерируемых компиляторами AMD кодах слишком большого числа регистров по сравнению со средствами Nvidia CUDA, хотя в ряде случаев причины недостаточной по сравнению с ожиданием производительности остаются невыясненными. Имеющиеся замечания относительно программных средств SDK от AMD коррелируют с четким указанием во многих из приводившихся выше публикаций на полученные данные по производительности как на предварительные, соответственно предполагающими улучшения в связи с быстрым развитием SDK.

Поскольку данные о более высокой производительности MI250/MI250X также имеют место быть, важным на настоящий момент времени следует считать факт недостаточно высокого уровня предсказуемости сравнительной производительности этих GPU относительно A100.

Здесь нельзя пытаться базироваться на «статистическом анализе» сравнительных данных о производительности, а нужно основываться на данных для конкретных приложений и объектах исследования (которые в первом приближении имеются и в данном обзоре), а также на близких (в чисто математическом плане) используемых методах, и учитывать уже обнаруженные недостатки MI250/MI250X и их SDK (последние, возможно, могут быть исправлены в новых версиях).

Но надо иметь в виду и возможную противоположность выводов об эффективности MI200 при сравнительном учете цен и TCO, и соответственно выбрать сопоставление с A100 полных MI250/MI250X или отдельных GCD.

В момент написания обзора ожидается уже появление MI300 и суперкомпьютера El Capitans в 2023 году, в Ливерморской национальной лаборатории Лоуренса (США), с интеграцией ЦП архитектуры Zen 4 в MI300. Это подтверждает вероятный прогресс AMD в GPU и в суперкомпьютерах ближайшего будущего. В [**375**] указано на сохраняющуюся маленькую величину кэша L1, что могло бы стать потенциально слабым местом MI300.

По отношению к HPC следует указать на большую ориентацию AMD на эту область по сравнению с большим направлением на ИИ у Nvidia, что проявляется и в создаваемых суперкомпьютерах из Top500.

Заключение

В данном обзоре выше были обсуждены преимущества и возможные недостатки GPU. Что касается конкретных данных по производительности, о них много сказано выше, и их целесообразно сочетать и со стоимостными показателями (включая и энергетические показатели), которые здесь не обсуждаются. Ниже приводится в основном общее резюме чуть другого, более общего и/или более краткого характера.

1. Появляется конкуренция современных разработчиков GPU по производительности и другим показателям, в том числе не только из США, но возможно и из Китая; ожидается и европейский акселератор, который также может применяться на экзамасштабных суперкомпьютерах. Соответственно современная почти монополия GPU от Nvidia будет, вероятно, понемногу уменьшаться.

2. Общим технологическим направлением построения GPU становится применение мультикристалльных технологий (чиплетов), впервые ставшими активно применяться скорее AMD [**376**]. Это обеспечивает масштабирование производительности при возможности ее реализации имеющим финансовую эффективность способом, при незначительном в настоящее время увеличении задержки, в том числе межсоединений между вычислительными ядрами.
3. Общей тенденцией может стать интеграция ЦП и GPU (AMD MI300, Intel Falcon Shore, Nvidia GH200), а также нескольких GPU внутри одной модели (как в AMD MI200 и в Intel PVC).
4. Сверхбыстрое развитие GPU приводит к тому, что попытки стандартизации каких-либо аппаратных компонент пока реализуются ограничено. Так, применение стандарта форм-фактора OAM в MI200, PVC и BR100 сочетается с применением Nvidia собственных форм-факторов SXM, что выглядит достаточно понятно в свете поставок Nvidia собственных готовых к работе вычислительных систем от multi-GPU серверов до кластеров.

   Стандартизация еще сложнее для межсоединений GPU — все рассмотренные GPU использовали собственный, отличный от конкурентов выбор. Здесь причиной ориентации не на стандарты является скорее конкурентная борьба за лидерство по производительности.
5. Общим направлением является расширение применения multi-GPU систем для задач ИИ и HPC. В соответствующих серверах необходимо использовать по два содержащих десятки ядер процессора, в качестве которых применимы не только Intel Xeon и AMD EPYC, но и ARM-процессоры (классическим может стать вариант с Nvidia Grace или GH200). Альтернативой может быть и применение одного, содержащего свыше полусотни ядер серверного процессора (такие ЦП предлагают сегодня и AMD, и Intel).

   Традиционные для HPC задачи квантовой химии ранее использовали GPU в первую очередь для расчетов в базисах плоских волн или вычислительно более сложными методами с учетом электронной корреляции, где основное время вычисления приходилось на общематематические методы. Для широко распространенных задач квантовой химии, где производительность лимитируется временем расчета двухэлектронных интегралов в гауссовском базисе (в методах HF и DFT), достижение высокой эффективности расчета на GPU, дающее и хорошую масштабируемость производительности при работе с несколькими GPU, было достигнуто только в самое последнее время (соответствующие данные представлены в обзоре). Это дает возможность более активного применения multi-GPU систем и для задач квантовой химии.

6. Что касается BR100, они могут быть актуальны в первую очередь разработчикам программного обеспечения, в том числе в научной области (поскольку приложений, работающих на BR100, практически нет) – и возможно, особенно в странах, где разработчики программ и потребители GPU имеют существенные финансовые ограничения. Однако отсутствие в BR100 формата FP64, традиционного в классической HPC-области, приведет, вероятно, к основной ориентации BR100 на задачи ИИ. Но перспективы применения BR100 непонятны из-за санкций США.
7. Учитывая определенные задержки развития в используемой Intel полупроводниковой технологии и задержки начала поставок Ponte Vecchio по сравнению с исходными планами Intel, и возникающую необходимость перехода на работу с программной моделью DPC++ /oneAPI, можно предположить, что более резкое продвижение Intel вперед с GPU можно ожидать после появления GPU Falcon Shores или последующего интегрированного с x86 устройства (скорее после реализации новой перспективной технологии 18A).

   Наиболее ярким вкладом Intel в развитие направления применения GPU сегодня представляется разработка и поддержка модели программирования DPC++.
8. Не вызывает сомнений активное расширение применения GPU H100 от Nvidia, с более быстрым ростом в области ИИ. Как уже выпускаемые, так и ближайшие ожидаемые GPU Nvidia H100 (GH200 также фактически содержат H100) будут очень актуальны для задач ИИ и HPC. Но наиболее сильным рывком вперед по масштабируемости производительности выглядят возможности построения кластеров с H100 с применением сетей NVLink.

   Преимуществом аппаратных средств Nvidia в первую очередь для задач ИИ являются поставки уже готовых к работе с ИИ серверных систем DGX и кластеров DGX SuperPOD, которые относятся к суперкомпьютерному уровню. Хотя такие системы могут применяться и для задач HPC, нужны готовые к ним приложения. Другим современным преимуществом H100 является огромный набор эффективно взаимодействующих и дополняющих друг друга средств SDK для HPC и ИИ.
9. Альтернативные Nvidia GPU от AMD представляются в настоящее время наиболее актуальными в первую очередь для расчетов по уже показавшим высокую производительность на AMD MI200 приложениям, или ожидаемым к такому в самое ближайшее время, а также как поле для разработки нового программного обеспечения,

включая портирование уже существующих приложений. С точки зрения потенциального использования приложений в недалеком будущем это может быть целесообразно для любых областей HPC и ИИ.

Применение этих GPU, очевидно, будет расширяться в первую очередь в науке и в деятельности разработчиков программных средств. Имеющаяся определенная сопоставимость в производительности с рассматривавшимися GPU от Nvidia означает повышенную важность ценовых показателей.

10. С учетом тенденций настоящего времени можно предположить, что будет расширяться применение на GPU универсальных средств разработки программного обеспечения, а не ориентированных на определенную архитектуру производителя.

Список использованных источников

[1] *Top500 the list*, The Green500 ranking, 61st edition.– 2023. URL ↑140

[2] Tschudi W., Xu T., Sartor D., Stein J. *High-performance data centers. A research roadmap*, LBNL-53483.– 2004.– 53 pp. URL ↑140

[3] Maltenberger T., Ilic I., Tolovski I, Rab T. *Evaluating multi-GPU sorting with modern interconnects // SIGMOD'22: Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA, June 12–17, 2022), New York: ACM.– 2022.– ISBN 978-1-4503-9249-5.– Pp. 1795–1809. doi ↑140, 157, 188

[4] *Top500 the list*, The Top500 ranking, 61st edition.– 2023. URL ↑141, 142, 254

[5] Кузьминский М. Б. *Современные серверные ARM-процессоры для суперЭВМ: A64FX и другие. Начальные данные тестов производительности //* Программные системы: теория и приложения.– 2022.– Т. **13**.– № 1(52).– С. 63–129. doi URL ↑141, 146, 198

[6] Gao J., Zheng F., Qi F, Ding Y, Li H., Lu H., He W., Wei H., Jin L., Liu X., Gong D., Wang F., Zheng Y., Sun H., Zhou Z., Liu Y., You H. *Sunway supercomputer architecture towards exscale computing: analysis and practice //* Science China Information Sciences.– 2021.– Vol. **64**.– No. 4.– id. 141101.– 21 pp. doi URL ↑141

[7] Selig J. *The cerebras software development kit: A technical overview.*– Cerebras systems Inc..– 2022.– 8 pp. URL ↑141

[8] *Andromeda, a 13.5 Million Core AI Supercomputer*, a section on the Cerebras company site.– 2024. URL ↑141

[9] *Top500 the list*, List Statistics of TOP500, 61st edition.– 2023. URL URL ↑141

[10] Morgan T. P. *Chip roadmaps unfold, crisscrossing and interconnecting, at AMD*, The Next Platform.– Stackhouse Publishing.– 2022. URL ↑141, 147

[11] Shah A. *Intel Reiterates Plans to Merge CPU, GPU High-performance Chip Roadmaps*, Tabor network.– HPCwire.– 2022. URL ↑141

[12] Morgan T. P. *The Increasingly Graphic Nature Of Intel Datacenter Compute*, The Next Platform.– Stackhouse Publishing.– 2022. URL ↑141

[13] Evans J. *Nvidia Grace // 2022 IEEE Hot Chips 34 Symposium (HCS)* (Cupertino, CA, USA, 21–23 August 2022).– IEEE.– 2022.– Pp. 1–20. doi ↑141, 146, 147, 217, 218

[14] Elster A. C., Haugdahl T. A. *Nvidia Hopper GPU and Grace CPU Highlights* // Computing in Science and Engineering.– 2022.– Vol. **24**.– No. 2.– Pp. 95–100. doi ↑141, 147, 219

[15] Evans J. *Inside Grace*, Featured Playlists, GPU Technology Conference (GTC), Nvidia On-Demand.– Nvidia.– 2022. URL ↑141, 147

[16] *CUDA C++ Programming Guide*, Release 12.4.– Nvidia.– 2024.– 544 pp. URL ↑142, 148, 149, 150, 180, 182, 183, 185, 192

[17] *Ampere Tuning Guide*, Release 12.4.– Nvidia.– 2024.– 22 pp. URL ↑142, 152

[18] Zhang Z., Jiao S., Li J., Wu W., Wan L., Qin X., Hu W., Yang J. *KSSOLV-GPU: An efficient GPU-enabled MATLAB toolbox for solving the Kohn-Sham equations within density functional theory in plane-wave basis set* // Chinese Journal of Chemical Physics.– 2021.– Vol. **34**.– No. 5.– Pp. 552–564. doi ↑143

[19] Giannozzi P., Baseggio O., Bonfà P., Brunato D., Car R., Carnimeo I., Cavazzoni C., de Gironcoli S., Delugas P., Ruffino . F., Ferretti A., Marzari N., Timrov I., Urru A., Baroni S. *Quantum ESPRESSO toward the exascale* // The Journal of chemical physics.– 2020.– Vol. **152**.– No. 15.– id. 154105. doi ↑143

[20] Хэ Личжун *Темпы локализации графических процессоров ускоряются, и новые команды продолжают появляться*, Краткий отчет об отрасли.– Пекин: Capital Securities.– 2022.– 15 с. (Китайский) URL ↑144

[21] Bispo J., Barbosa J., Silva P., Morales C., Myllykoski M., Ojeda-May P., Bialczak M., Uchroński M., Włodarczyk A., Wauligmann P., Krishnasamy E., Varrette S., Lührs S. *Best Practice Guide: Modern Accelerators*/ ed. Shoukourian H..– PRACE.– 2021.– 111 pp. URL ↑144, 145, 229

[22] Finkelstein J., Smith J. S., Mniszewski S. M., Barros K., Negre C. F. A., Rubensson E. H., Niklasson A. M. N. *Quantum-based molecular dynamics simulations using tensor cores* // Journal of Chemical Theory and Computation.– 2021.– Vol. **17**.– No. 10.– Pp. 6180–6192. doi ↑144

[23] Posey S., Luitjens J., Hennigh O., Oberlin S. *GPU-based HPC and AI developments for CFD* (Maui, Hawaii, USA, July 11-15, 2022).– 2022.– id. ICCFD11-3803.– 5 pp. URL ↑145

[24] Schade R., Kenter T., Elgabarty H., Lass M., Schütt O., Lazzaro A., Pabst H., Mohr S., Hutter J., Kühne T. D., Plessl C. *Towards electronic structure-based* ab-initio *molecular dynamics simulations with hundreds of millions of atoms* // Parallel Computing.– July 2022.– Vol. **111**.– id. 102920.– 11 pp. doi ↑145

[25] Terzo O., Martinovič J (eds.) *HPC, Big Data, and AI Convergence Towards Exascale: Challenge and Vision*, 1st ed..– CRC Press.– 2022.– ISBN 9781003176664.– 322 pp. doi ↑145

[26] Nowicki M., Górski Ł., Bała P. *PCJ Java library as a solution to integrate HPC, Big Data and Artificial Intelligence workloads* // Journal of Big Data.– 2021.– Vol. **8**.– No. 1.– Pp. 1–21.– id. 62. doi ↑145

[27] Yin F., Shi F. *A comparative survey of Big Data computing and HPC: from a parallel programming model to a cluster architecture* // International Journal of Parallel Programming.– 2022.– Vol. **50**.– No. 1.– Pp. 27–64. doi ↑145

[28] Yin J., Wang F., Shankar M. *Strategies for integrating deep learning surrogate models with HPC simulation applications // 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (Lyon, France, 2022).– IEEE.– 2022.– ISBN 978-1-6654-9747-3.– Pp. 01–10. doi ↑145, 270

[29] Sukumar S. R., Balma J. A., Rickett C. D., Maschhoff K. J., Landman J., Yates C. R., Chittiboyina A. G., Peterson Y. K., Vose A., Byler K., Baudry J., Khan I. A. *The convergence of HPC, AI and Big Data in rapid-response to the COVID-19 pandemic // Driving Scientific and Engineering Discoveries Through the Integration of Exeriment, Big Data, and Modeling and Simulation: 21st Smoky Mountains Computational Sciences and Engineering, SMC 2021, Virtual Event, October 18-20, 2021, Revised Selected Papers*, Communications in Computer and Information Science.– vol. **1512**.– 2022.– ISBN 978-3-030-96497-9.– Pp. 157-172. doi ↑145

[30] Ejarque J., Badia R. M., Albertin L., f, Aloisio G., Baglione E., Becerra Y., Boschert S., Berlin J. R., D'Anca A., Elia D., Exrtier F., Fiore S., Flich J., Folch A., Gibbons S. J., Koldunov N., Lordan F., Lorito S., Løvholt F., Macías J., Volpe M. *Enabling dynamic and intelligent workflows for HPC, data analytics, and AI convergence* // Future Generation Computer Systems.– September 2022.– Vol. **134**.– Pp. 414–429. doi ↑145

[31] Ihde N., Marten P., Eleliemy A., Poerwawinata G., Silva P., Tolovski I., Ciorba F. M., Rabl T. *A survey of Big Data, High Performance Computing, and Machine Learning benchmarks*, Technology Conference on Performance Evaluation and Benchmarking, Lecture Notes in Computer Science.– vol. **13169**, Cham: Springer.– 2021.– ISBN 978-3-030-94436-0.– Pp. 98–118. doi ↑145

[32] *High-Performance Deep Learning Project (HiDL)*, NOWLAB: Network Based Computing Lab.– Ohio state university. URL ↑145

[33] *High-Performance Big Data Project (HiBD)*, NOWLAB: Network Based Computing Lab.– Ohio state university. URL ↑145

[34] Jeon W., Ko G., Lee J., Lee H., Ha D., Ro W. W. *Deep learning with GPUs* // Advances in Computers.– 2021.– Vol. **122**.– Pp. 167–215. doi ↑145

[35] Hong M., Xu L. *Biren BR100 GPGPU: Accelerating Datacenter Scale AI Computing*, 2022 IEEE Hot Chips 34 Symposium (HCS) (Cupertino, CA, USA).– 2022.– Pp. 1–22. doi ↑146, 152, 154, 155, 157, 158, 159

[36] Shilov A. *Russian Company Taps China's Zhaoxin x86 CPU to Replace AMD, Intel CPUs*, Tom's Hardware.– New York: Future US.– 2022. URL ↑146

[37] Shang H., Li F., Zhang Y., Zhang L., Fu Y., Gao Y., Wu Y., Duan X., Lin R., Liu X., Liu Y., Chen D. *Exreme-scale* ab initio *quantum Raman spectra simulations on the leadership HPC system in China* // *SC'21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, New York: ACM.– November 2021.– ISBN 978-1-4503-8442-1.– id. 6.– 13 pp. doi ↑146

[38] Schneider D. *The Exascale Era is Upon Us: The Frontier supercomputer may be the first to reach 1,000,000,000,000,000,000 operations per second* // IEEE Spectrum.– January 2022.– Vol. **59**.– No. 1.– Pp. 34–35. doi ↑146

[39] Dongarra J., Geist A. *Report on the Oak Ridge National Laboratory's Frontier System*, Technical Report ICL-UT-22-05.– Oak Ridge National Laboratory.– 2022.– Accessed 15.10.2023. URL ↑146

[40] *Frontier Spec Sheet*, Oak Ridge National Laboratory.– UT-Battelle.– 2019.– 4 pp. URL ↑146

[41] *GPU nodes — LUMI-G*, Hardware documentation.– LUMI (Large Unified Modern Infrastructure) consortium. URL ↑146, 233, 236, 237, 239, 240, 241

[42] Markomanolis G. S., Alpay A., Young J., Klemm M., Malaya N., Esposito A., Heikonen J., Bastrakov S., Debus A., Kluge T., Steiniger K., Stephan J., Widera R., Bussmann M. *Evaluating GPU programming models for the LUMI supercomputer* // *Supercomputing Frontiers*, Lecture Notes in Computer Science (Asian Conference on Supercomputing Frontiers).– vol. **13214**, Cham: Springer.– 2022.– ISBN 978-3-031-10419-0.– Pp. 79–101. doi ↑146, 199, 206, 229, 241, 242, 243, 247, 249

[43] *Aurora*, Argonne Leadership Computing Facility.– Argonne National Laboratory. URL ↑146

[44] Peckham O. *LRZ announces new phase of SuperMUC-NG Supercomputer with Intels Ponte Vecchio GPU*, Tabor network.– HPCwire.– 2021. URL ↑146

[45] Kwack J. H., Tramm J., Bertoni C., Ghadar Y., Homerding B., Rangel E., Knight C., Parker S. *Evaluation of performance portability of applications and mini-apps across AMD, Intel and Nvidia GPUs* // *2021 International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (14 November 2021, St. Louis, MO, USA).– IEEE.– 2021.– ISBN 978-1-6654-2439-4.– Pp. 45–56. doi ↑146, 246

[46] *HPE Cray Supercomputing EX*.– Hewlett Packard Enterprise Development LP.– 2024. URL ↑146, 241

[47] Bertoni C., Parker S. *Aurora overvew*, ALCF SDL Workshop (October 6, 2022).– 2022.– 20 pp. URL ↑146, 170

[48] Morgan T. P. *The NVSwitch Fabric That Is The Hub Of The DGX H100 SuperPOD*, The Next Platform.– Stackhouse Publishing.– 2022. URL ↑146

[49] Ishii A., Wells R. *The Nvlink-Network switch: Nvidia's switch chip for high communication-bandwidth superpods // 2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 Aug., 2022, Cupertino, CA, USA).– IEEE.– 2022.– ISBN 978-1-6654-6028-6.– Pp. 1–23. ↑146

[50] Eassa A., Ishii A., Wells R. *Upgrading Multi-GPU Interconnectivity with the Third-Generation Nvidia NVSwitch*, Technical blog, Nvidia developer.– 2022. ↑146

[51] *BR100 series general purpose GPU chip*.– Shanghai: Biren Technology.– 2023. ↑147, 154, 155, 159

[52] Andersch M., Palmer G., Krashinsky R., Stam N., Mehta V., Brito G., Ramaswamy S. *Nvidia Hopper Architecture In-Depth*, Technical blog, Nvidia developer.– 2022. ↑147

[53] Alcorn P. *From Opteron to Milan: Crusher Supercomputer Comes Online With New AMD CPUs and MI250X GPUs*, Tom's Hardware.– New York: Future US.– 2022. ↑147

[54] *Intel Xeon CPU Max series product overview*.– Intel.– 2023.– Accessed 15.10.2023. ↑147

[55] *Accelerator Processor Stream*.– European Processor Initiative.– 2022. ↑147

[56] *EPI EPAC1.0 RISC-V test chip samples delivered*, News.– European Processor Initiative.– 2021. ↑147

[57] Kovač M., Notton P., Hofman D., Knezović J. *How Europe is preparing its core solution for exascale machines and a global, sovereign, advanced computing platform* // Mathematical and Computational Applications.– 2020.– Vol. **25**.– No. 3.– Pp. 46. ↑147

[58] *HIP Programming Guide*, Version 5.0.– 2023.– Accessed 15.10.2023. ↑148, 149, 150, 192, 242

[59] *OpenMP Application Programming Interface*, Version 5.2.– OpenMP Architecture Review Board.– 2021.– 669 pp. ↑148

[60] *Khronos OpenCL Registry*, Formatted specifications and other related documentation.– Khronos Group. ↑148

[61] *SYCL 2020 Specification*, rev. 6.– Khronos Group.– 2022.– 585 pp. ↑148

[62] *DPC++ Part 1: An Introduction to the New Programming Model*.– Intel (Accessed 15.10.2023). ↑148

[63] Bavarsad N. N., Makrani H. M., Sayadi H., Landis L., Rafatirad S., Homayoun H. *HosNa: A DPC++ benchmark suite for heterogeneous architectures // 2021 IEEE 39th International Conference on Computer Design (ICCD)* (24–27 October 2021, Storrs, CT, USA).– IEEE.– 2021.– ISBN 978-1-6654-3219-1.– Pp. 509–516. ↑148

[64] Trott C., Berger-Vergiat L., Poliakoff D., Rajamanickam S., Lebrun-Grandie D., Madsen J., Al Awar N., Gligoric M., Shipman G., Womeldorff G. *The Kokkos EcoSystem: comprehensive performance portability for high performance computing* // Computing in Science & Engineering.– 2021.– Vol. **23**.– No. 5.– Pp. 10–18. ↑148

[65] Trott C. R., Lebrun-Grandié D., Arndt D., Ciesko J., Dang V., Ellingwood N., Gayatri R., Harvey E., Hollman D. S., Ibanez D., Liber N., Madsen J., Miles J., Poliakoff D., Powell A., Rajamanickam S., Simberg M., Sunderland D., Turcksin B., Wilke J. *Kokkos 3: Programming model extensions for the exascale era* // IEEE Transactions on Parallel and Distributed Systems.– 2021.– Vol. **33**.– No. 4.– Pp. 805–817. ↑148

[66] Moore S. *The state of the LAMMPS KOKKOS package*, SAND2021-9785C.– Albuquerque, NM: Sandia National Lab.– 2021.– Accessed 15.10.2023. ↑148

[67] Ghadar Y., Applencourt T., Homerding B., Harms K., Hammond J. *SYCL Programming Model for Aurora*, 2020 ECP Annual Meeting.– 2020. ↑149, 162, 171

[68] Van Oostrum R., Chalmers N., Mc Dougall D., Bauman P., Curtis N., Malaya N., Wolfe N. *AMD GPU Hardware Basics*, Frontier Application Readiness Kick-Off Workshop.– 2019.– 55 pp. ↑149

[69] *Intel oneAPI GPU Optimization Guide Release 2022.3*.– Intel (Accessed 15.10.2023). ↑150, 162, 164, 165, 166, 167, 168, 169, 171

[70] Khudia D., Huang J., Basu P., Deng S., Liu H., Park J., Smelyanskiy M. *Fbgemm: Enabling high-performance low-precision deep learning inference*.– 2021.– 5 pp. arXiv 2101.05615 ↑151

[71] Carrasco R., Vega R., Navarro C. A. *Analyzing GPU tensor core potential for fast reductions // 2018 37th International Conference of the Chilean Computer Science Society (SCCC)* (05–09 November 2018, Santiago, Chile).– IEEE.– 2018.– ISBN 9781538692349.– Pp. 1–6. ↑151

[72] Gupta G. *Using Tensor Cores for Mixed-Precision Scientific Computing*, Technical blog, Nvidia developer.– 2019. ↑151

[73] *Nvidia A100 Tensor Core GPU Architecture*, V1.0.– Nvidia.– 2020.– 82 pp. ↑151, 166, 177, 178, 179, 180, 183, 184, 185, 186, 188

[74] *754-2019 — IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2019, Revision of IEEE 754-2008.– 2019.– 84 pp. ↑152

[75] Kalamkar D., Mudigere D., Mellempudi N., Das D., Banerjee K., Avancha S., Vooturi D. T., Jammalamadaka N., Huang J., Yuen H., Yang J., Park J., Heinecke A., Georganas E., Srinivasan S., Kundu A., Smelyanskiy M., Kaul B., Dubey P. *A study of BFLOAT16 for deep learning training*.– 2019.– 10 pp. arXiv 1905.12322 ↑152

[76] Stosic D., Micikevicius P. *Accelerating AI Training with Nvidia TF32 Tensor Cores*, Technical blog, Nvidia developer.– 2021. ↑152

[77] Micikevicius P., Stosic D., Burgess N., Cornea M., Dubey P., Grisenthwaite R., Ha S., Heinecke A., Judd P., Kamalu J., Mellempudi N., Oberman S., Shoeybi M., Siu M., Wu H. *Fp8 formats for deep learning*.– 2022.– 9 pp. arXiv 2209.05433 ↑152

[78] *Nvidia H100 Tensor Core GPU Architecture*, Includes final GPU / memory clocks and final TFLOPS performance specs, V1.04.– Nvidia.– 2023.– 71 pp. ↑152, 166, 179, 180, 181, 182, 183, 212, 214, 215, 222, 223, 224

[79] Sun W., Li A., Geng T., Stuijk S., Corporaal H. *Dissecting tensor cores via microbenchmarks: latency, throughput and numerical behaviors* // IEEE Transactions on Parallel and Distributed Systems.– 2022.– Vol. **34**.– No. 1.– Pp. 246–261. ↑152, 198

[80] Lehmann M., Krause M. J., Amati G., Sega M., Harting J., Gekle S. *Accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit, and customized 16-bit number formats* // Physical Review E.– 2022.– Vol. **106**.– No. 1.– id. 015308. ↑153

[81] Domke J., Matsumura K., Wahib M., Zhang H., Yashima K., Tsuchikawa T., Tsuji Y., Podobas A., Matsuoka S. *Double-precision FPUs in high-performance computing: an embarrassment of riches? 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (20–24 May 2019, Rio de Janeiro, Brazil).– IEEE.– 2019.– ISBN 978-1-7281-1246-6.– Pp. 78–88. ↑153

[82] Schade R., Kenter T., Elgabarty H., Lass M., Schütt O., Lazzaro A., Pabst H., Mohr S., Hutter J., Kühne T. D., Plessl C. *Towards electronic structure-based* ab-initio *molecular dynamics simulations with hundreds of millions of atoms* // Parallel Computing.– 2022.– Vol. **111**.– id. 102920.– 11 pp. ↑153

[83] Schade R., Kenter T., Elgabarty H., Lass M., Kühne T. D., Plessl C. *Breaking the exascale barrier for the electronic structure problem in* ab-initio *molecular dynamics*.– 2022.– 6 pp. arXiv 2205.12182 ↑153

[84] Yu V. W., Govoni M. *GPU acceleration of large-scale full-frequency GW calculations*.– 2022.– 54 pp. arXiv 2203.05623 ↑153

[85] Eriksen J. J. *Efficient and portable acceleration of quantum chemical many-body methods in mixed floating point precision using OpenACC compiler directives* // Molecular Physics.– 2017.– Vol. **115**.– No. 17–18.– Pp. 2086–2101. ↑153

[86] Ruda D., Turek S., Ribbrock D., Zajac P. *Very fast FEM Poisson solvers on lower precision accelerator hardware*, ECCOMAS Congress 2022 (5–9 June 2022, Oslo, Norway).– 2022.– 24 pp. ↑153

[87] Ootomo H., Yokota R. *Recovering single precision accuracy from Tensor Cores while surpassing the FP32 theoretical peak performance* // The International Journal of High Performance Computing Applications.– 2022.– Vol. **36**.– No. 4.– Pp. 475–491. ↑153

[88] Jain A., Sharma N. *Accelerated AI inference at CNN-based machine vision in ASICs: A design approach* // ECS Transactions.– 2022.– Vol. **107**.– No. 1.– Pp. 5165. ↑153

[89] Gallet B., Gowanlock M. *Computing double precision Euclidean distances using GPU tensor cores*.– 2022.– 10 pp. arXiv 2209.11287 ↑153, 183

[90] Domke J., Vatai E., Drozd A., Chen P. T, Oyama Y., Zhang L., Salaria S., Mukunoki D., Podobas A., Wahib M. T, Matsuoka S. *Matrix engines for high performance computing: A paragon of performance or grasping at straws?2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (17–21 May 2021, Portland, OR, USA).– IEEE.– 2021.– ISBN 978-1-6654-4066-0.– Pp. 1056–1065. ↑153, 184, 195

[91] Tan H., Yan R., Yang L., Huang L., Xiao L., Yang Q. *Efficient multiple-precision and mixed-precision floating-point fused multiply-accumulate unit for HPC and AI applications* // *Algorithms and Architectures for Parallel Processing*, 22nd International Conference ICA3PP 2022 (Copenhagen, Denmark, October 10–12, 2022), Lecture Notes in Computer Science.– vol. **13777**, Cham: Springer Nature Switzerland.– 2023.– ISBN 978-3-031-22676-2.– Pp. 642–659. doi ↑154

[92] *Эксклюзивное интервью с руководителями Biren Technology: деконструкция первого 7-нм графического процессора компании*, Обзор от компании MooreElite.com (Hefei).– 2022 (Китайский). URL ↑154, 157, 158, 159

[93] *Nvidia A100 Tensor Core GPU Datasheet*, V1.0.– Nvidia.– 2020.– 3 pp. URL ↑156, 157, 158

[94] Choquette J., Lee E., Krashinsky R., Balan V., Khailany B. *3.2 The A100 Datacenter GPU and Ampere Architecture* // *2021 IEEE International Solid-State Circuits Conference (ISSCC)* (13–22 February 2021, San Francisco, CA, USA).– IEEE.– 2021.– ISBN 9781728195506.– Pp. 48–50. doi ↑156, 157, 159, 184, 185, 186, 188

[95] *Nvidia A100 tensor core GPU architecture*, V1.0.– Nvidia.– 2020.– 82 pp. URL ↑156

[96] Hassanpour M., Riera M., González A. *A survey of near-data processing architectures for neural networks* // Machine Learning and Knowledge Extraction.– 2022.– Vol. **4**.– No. 1.– Pp. 66–102. doi ↑157

[97] Gómez-Luna J., Guo Y., Brocard S., Legriel J., Cimadomo R., Oliveira G. F., Singh G., Mutlu O. *An experimental evaluation of machine learning training on a real processing-in-memory system*.– 2022.– 21 pp. arXiv 2207.07886 ↑157

[98] Niu D., Li S., Wang Y., Han W., Zhang Z., Guan Y., Guan T., Sun F., Xue F., Duan L., Fang Y., Zheng H., Jiang X., Wang S., Zuo F., Wang Y., Yu B., Ren Q., Xie Y. *184QPS/W 64Mb/mm^2 3D logic-to-DRAM hybrid bonding with process-near-memory engine for recommendation system* // *IEEE International Solid-State Circuits Conference (ISSCC)* (20–26 February 2022, San Francisco, CA, USA).– IEEE.– 2022.– Pp. 1–3. doi ↑157

[99] *BiLi 106M*, Product details.– Shanghai: Biren Technology.– 2020–2023. URL ↑156, 158

[100] *BiLi 106B, 106C*.– Shanghai: Biren Technology.– 2020–2023. URL ↑156, 158

[101] Blankenship R., Wagh M. *Introducing the CXL 3.1 Specification*.– Compute express link consortium.– 2022.– 27 pp. URL ↑157

[102] Coughlin T. *Digital storage and memory* // Computer.– 2022.– Vol. **55**.– No. 1.– Pp. 20–29. doi ↑157

[103] *Nvidia A100 Tensor Core GPU Datasheet*.– Nvidia.– 2021.– 3 pp. URL ↑157

[104] *Ampere Tuning Guide*, Release 12.4.– Nvidia.– 2024.– 22 pp. URL ↑157

[105] *Server/OAI*, Wiki page.– Open computers project. URL ↑158

[106] *Nvidia DGX A100*, Datasheet.– Nvidia.– 2023.– 2 pp. URL ↑158

[107] Morgan T. P. *China launches the inevitable indigenous GPU*, The Next Platform.– Stackhouse Publishing.– 2022. ↑159

[108] *BIRENSUPA software development platform*, Product details.– Shanghai: Biren Technology.– 2023. ↑159

[109] *MLPerf inference: datacenter benchmark suite results.*– MLCommons. ↑159, 160

[110] Reddi V. J., Cheng C., Kanter D., Mattson P., Schmuelling G., Carole-Wu J., Anderson B., Breughe M., Charlebois M., Chou W., Chukka R., Coleman C., Davis S., Deng P., Diamos G., Duke J., Fick D., Gardner J. S., Hubara I., Idgunji S., Jablin T. B., Jiao J., John T. S., Kanwar P., Lee D., Liao J., Lokhmotov A., Massa F., Meng P., Micikevicius P., Osborne C., Pekhimenko G., Rajan A. T. R., Sequeira D., Sirasao A., Sun F., Tang H., Thomson M., Wei F., Wu E., Xu L., Yamada K., Yu B., Yuan G., Zhong A., Zhang P., Zhou Y. *Mlperf inference benchmark // 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)* (30 May 2020–03 June 2020, Valencia, Spain).– IEEE.– 2020.– ISBN 978-1-7281-4661-4.– Pp. 446–459. ↑159

[111] Saad M. H., Hashima S., Sayed W., El-Shazly E. H., Madian A. H., Fouda M. M. *Early diagnosis of COVID-19 images using optimal CNN hyperparameters //* Diagnostics.– 2023.– Vol. **13**.– No. 1.– id. 76. ↑160

[112] Devlin J., Ming-Chang W., Lee K., Toutanova K. *BERT: Pre-training of deep bidirectional transformers for language understanding // Human Language Technology: Conference of the North American Chapter of the Association of Computational Linguistics.*– V. 1, NAACL-HLT 2019 (June 2–June 7, 2019, Minneapolis, Minnesota, USA).– ACL.– 2019.– ISBN 978-1-950737-13-0.– Pp. 4171–4186. ↑160

[113] *Nvidia TensorRT, an SDK for high-performance deep learning inference*, Web site, Nvidia developer.– Nvidia. ↑161

[114] Blythe D. *The X^e GPU architecture // 2020 IEEE Hot Chips 32 Symposium (HCS)* (16–18 August 2020, Palo Alto, CA, USA).– IEEE.– 2020.– ISBN 978-1-7281-7129-6.– Pp. 1–27. ↑162, 163, 169

[115] Blythe D. *X^e HPC Ponte Vecchio // 2021 IEEE Hot Chips 33 Symposium (HCS)* (22–24 August 2021, Palo Alto, CA, USA).– IEEE.– 2021.– ISBN 978-1-6654-1397-8.– Pp. 1–34. ↑162, 164

[116] *Intel data center GPU Max series product brief.*– Intel (Accessed 15.10.2023). ↑162, 163, 164, 168, 169, 170, 174

[117] *Intel data center GPU flex series product brief.*– Intel (Accessed 15.10.2023). ↑162

[118] Dhote D., Virmani C., Krishna K. G., Raghav S. *The science of ray tracing //* International Journal of Computer Applications.– 2020.– Vol. **176**.– No. 42.– Pp. 15–20. ↑162

[119] *Intel data center GPU Max series.*– Intel (Accessed 15.10.2023). ↑163, 168

[120] Jiang H. *Intel's Ponte Vecchio GPU: architecture, systems and software // 2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 August 2022, Cupertino, CA, USA).– IEEE.– 2022.– ISBN 978-1-6654-6028-6.– Pp. 1–29. doi ↑162, 167, 168, 171, 174

[121] Sidorova M., Gorbushin L., Koneva N. *Analytical review of electronic devices of modern supercomputing systems*, Proceedings of the International Russian Automation Conference, RusAutoCon2021 (September 5-11, 2021, Sochi, Russia), Lecture Notes in Electrical Engineering.– vol. **857**, Cham: Springer.– 2022.– ISBN 978-3-030-94201-4.– Pp. 25–33. doi ↑162

[122] Tian W., Li B., Li Z., Cui H., Shi J., Wang Y., Zhao J. *Using chiplet encapsulation technology to achieve processing-in-memory functions* // Micromachines.– 2022.– Vol. **13**.– No. 10.– Pp. 1790. doi ↑163, 164

[123] Moore S. K. *3 paths to 3D processors* // IEEE Spectrum.– 2022.– Vol. **59**.– No. 6.– Pp. 24–29. doi ↑163

[124] Zhang S., Li Z., Zhou H., Li R., Wang S., Kyung-Paik W., He P., *Recent prospectives and challenges of 3D heterogeneous integration* // e-Prime-Advances in Electrical Engineering, Electronics and Energy.– 2022.– id. 100052. doi ↑163

[125] Hadidi R., Asgari B., Mudassar B. A., Mukhopadhyay S., Yalamanchili S., Kim H. *Demystifying the characteristics of 3D-stacked memories: A case study for Hybrid Memory Cube // 2017 IEEE international symposium on Workload characterization (IISWC)* (01–03 October 2017, Seattle, WA, USA).– IEEE.– 2017.– Pp. 66–75. doi ↑163

[126] Ma X., Wang Y., Wang Y., Cai X., Han Y. *Survey on chiplets: interface, interconnect and integration methodology* // CCF Transactions on High Performance Computing.– 2022.– No. 4.– Pp. 43–52. doi ↑163

[127] *Universal chiplet interconnect express specifications*.– Universal Chiplet Interconnect Express.– 2023. URL ↑163

[128] Gomes W., Koker A., Stover P., Ingerly D., Siers S., Venkataraman S., Pelto C., Shah T., Rao A., O' .,Mahony, Karl E., Cheney L., Rajwani I., Jain H., Cortez R., Chandrasekhar A., Kanthi B., Koduri R. *Ponte Vecchio: A multi-tile 3D stacked processor for exascale computing // 2022 IEEE International Solid-State Circuits Conference (ISSCC)* (20–26 February 2022, San Francisco, CA, USA).– IEEE.– 2022.– ISBN 978-1-6654-2800-2.– Pp. 42–44. doi ↑163, 164

[129] Gomes W., Koker A., Stover P., Ingerly D., Siers S., Venkataraman S., Pelto C., Shah T., Rao A., O'Mahony F., Karl E., Cheney L., Rajwani I., Jain H., Cortez R., Chandrasekhar A., Kanthi B., Koduri R. *Ponte Vecchio: A multi-tile 3D stacked processor for exascale computing*, HPC user forum, Accelerated Computing Systems and Graphics Group.– 2021. URL ↑163, 164, 165, 166, 167, 168

[130] *Intel data center GPU Max series technical overview*.– Intel.– 2023 (Accessed 15.10.2023). URL ↑163, 164, 165, 166, 168, 169, 170

[131] Moore S. K. *Behind Intel's HPC chip that will pierce the exascale barrier*, Blog, IEEE Spectrum.– IEEE.– 2022. URL ↑163

[132] Ingerly D. B., Amin S., Aryasomayajula L., Balankutty A., Borst D., Chandra A., Cheemalapati K., Cook C. S., Criss R., Enamul K., Gomes W., Jones D., Kolluru K. C., Kandas A., G.-Kim S., Ma H., Pantuso D., Petersburg C. F., Phen-givoni M., Pillai A. M., Sairam A., Shekhar P., Sinha P., Stover P., Telang A., Zell Z. *Foveros: 3D integration and the use of face-to-face chip stacking for logic devices* // *2019 IEEE International Electron Devices Meeting (IEDM)* (07–11 December 2019, San Francisco, CA, USA).– IEEE.– 2019.– ISBN 978-1-7281-4033-9.– Pp. 19.6.1-19.6.4. doi ↑164

[133] Mahajan R., Sankman R., Patel N., Dae-Kim W., Aygun K., Qian Z., Mekonnen Y., Salama I., Sharan S., Iyengar D., Mallik D. *Embedded multi-die interconnect bridge (EMIB)–a high density, high bandwidth packaging interconnect* // *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)* (31 May 2016–03 June 2016, Las Vegas, NV, USA).– IEEE.– 2016.– Pp. 557–565. doi ↑164

[134] Irani S. *Hang SK Intel Ponte Vecchio compute accelerator OAM product and system*, 2021 OCP Global Summit.– 2021. URL ↑164

[135] Tekin A., A.Durak T., Piechurski C., Kaliszan D.,Sungur F. A., Robertsén F., Gschwandtn P. *State-of-the-art and trends for computing and interconnect network solutions for HPC and AI*, Partnership for Advanced Computing in Europe.– PRACE.– 2021.– 38 pp. URL ↑165

[136] Sun W., Li A., Geng T., Stuijk S., Corporaal H. *Dissecting tensor cores via microbenchmarks: latency, throughput and numerical behaviors* // IEEE Transactions on Parallel and Distributed Systems.– 2023.– Vol. **34**.– No. 1.– Pp. 246–261. doi ↑165

[137] *Intel Products formerly Alchemist.*– Intel (Accessed 15.10.2023). URL ↑166

[138] Watts D. *Lenovo ThinkSystem and ThinkAgile GPU Summary*, Product Guide.– Lenovo press.– 2024.– 71 pp. URL ↑166

[139] Liu Zh. *Intel Axes Data Center GPU Max 1350, Preps New Max 1450 for 'Different Markets'*, Tom's Hardware.– New York: Future US.– 2023. URL ↑166

[140] Vuduc R., Chandramowlishwaran A., Choi J., Guney M.(E.), Shringarpure A. *On the limits of GPU acceleration* // *Proceedings of the 2nd USENIX conference on Hot topics in parallelism*, HotPar'10 (June 14–15, 2010, Berkeley, CA, USA), Berkeley: USENIX Association.– 2010.– id. 13.– 6 pp. URL ↑167

[141] Hanindhito B., Gourounas D., Fathi A.,Trenev D., Gerstlauer A., John L. K. *GAPS: GPU-acceleration of PDE solvers for wave simulation* // *ICS '22: Proceedings of the 36th ACM International Conference on Supercomputing* (June 28–30, 2022, Virtual Event), NeW York: ACM.– 2022.– ISBN 978-1-4503-9281-5.– id. 30.– 13 pp. doi ↑167

[142] Chalmers N., Mishra A., Mc Dougall D., Warburton T. *HipBone: A performance-portable GPU-accelerated C++ version of the NekBone benchmark* // The International Journal of High Performance Computing Applications.– 2023.– Vol. **37**.– No. 5.– Pp. 560-577. doi ↑167, 250, 263, 265, 266

[143] Philippe J.-L. *Intel HW roadmap and architecture specifics*, OneAPI workshop with FocusCoE.– 2022.– 48 pp. URL ↑167, 169

[144] Min M., Yu-Lan H., Fischer P., Rathnayake T., Holmen J. *Nek5000/RS Performance on Advanced GPU Architectures*, ANL-22/81.– Argonne, IL: Argonne National Lab.(ANL).– 2022..– 30 pp. ↑170, 264, 265

[145] *oneAPI GPU Optimization Guide*, edition 2023.1.– Intel.– 2023.– 411 pp. URL ↑170

[146] Blythe D. *XeHPC ponte vecchio*, 2021 IEEE hot chips 33 symposium (HCS).– 2021.– Pp. 1–34. doi ↑169

[147] van der Steen A. J. *Overview of recent supercomputers*: Dongarra J. J. Van der Steen A. J. // *High-performance computing systems: Status and outlook*, Acta Numerica.– vol. **21**.– 2012.– Pp. 379–474 . doi URL ↑170

[148] *Intel Xeon CPU Max Series*, Product Brief.– Intel..– 2023.– 3 pp. URL ↑170

[149] Shipman G. M., Swaminarayan S., Grider G., Lujan J., Zerr R. J. *Early performance results on 4th Gen Intel Xeon scalable processors with DDR and Intel Xeon processors, codenamed sapphire rapids with HBM*.– 2022.– 5 pp. arXiv:2211.05712 ↑170

[150] *SiPearl: collaboration with Intel to accelerate exascale supercomputing deployment in Europe*, Press release.– The Silicon Pearl.– 2 pp. URL ↑170

[151] Parker S. *Future ALCF Systems*, 2021 ALCF Computational Performance Workshop.– Argonne National Laboratory.– 2021.– 25 pp. URL ↑171

[152] Ghadar Y., Williams T. *An Overview of Aurora, Argonnes Upcoming Exascale System*, ALCF Developer Session (December 11 2019).– 2020.– 45 pp. URL ↑171

[153] *SYCL*, The SYCL main page.– Khronos Group. URL ↑171

[154] Castaño G., aqir-Rhazoui Y., García C., Prieto-Matías M. *Evaluation of Intel's DPC++ Compatibility Tool in heterogeneous computing* // Journal of Parallel and Distributed Computing.– 2022.– Vol. **165**.– Pp. 120–129. doi ↑171, 172

[155] *Intel oneAPI 2023 Release: Preview the Tools*.– Intel (Accessed 15.10.2023). URL ↑171

[156] *Intel oneAPI Plug-Ins from Codeplay for Nvidia and AMD GPUs*.– Intel (Accessed 15.10.2023). URL ↑171

[157] *oneAPI DPC++ Compiler documentation*, LLVM documentation.– Intel.– 2024. URL ↑171

[158] *Benchmarking the performance of oneAPI on heterogeneous computing Platforms*, webinar slides.– Moasys, Intel Software.– 2021.– 30 pp. URL ↑171

[159] *OneAPI Specifications*.– Unified Acceleration Foundation.– 2024. URL ↑171

[160] Wang Z., Plyakhin Y., Sun C., Zhang Z., Jiang Z., Huang A., Wang H. *A source-to-source CUDA to SYCL code migration tool: Intel DPC++ Compatibility Tool // IWOCL '22: Proceedings of the 10th International Workshop on OpenCL* (May 10–12, 2022, Bristol, United Kingdom), New York: ACM.– 2022.– ISBN 978-1-4503-9658-5.– id. 17.– 2 pp. doi ↑172

[161] Fortenberry A., Tomov S. *Extending MAGMA portability with OneAPI // 2022 Workshop on Accelerator Programming Using Directives (WACCPD)* (13–18 November, 2022, Dallas, TX, USA).– IEEE.– 2022.– ISBN 978-1-6654-9019-1.– Pp. 22–31. doi ↑172

[162] Hardy D. J., Choi J., Jiang W., Tajkhorshid E. *Experiences porting NAMD to the Data Parallel C++ programming model // IWOCL '22: Proceedings of the 10th International Workshop on OpenCL* (May 10–12, 2022, Bristol, United Kingdom), New York: ACM.– 2022.– ISBN 978-1-4503-9658-5.– id. 15.– 5 pp. doi ↑172

[163] Alekseenko A., Páll S., Lindahl E. *Experiences with adding SYCL support to GROMACS // IWOCL '21: Proceedings of the 9th International Workshop on OpenCL* (April 27–29, 2021, Munich, Germany), New York: ACM.– ISBN 978-1-4503-9033-0.– Pp. 1.– id. 17. doi ↑172

[164] *GROMACS Highlights*.– GROMACS development team.– 2023. URL ↑172

[165] Alpay A., Soproni B., Wünsche H., Heuveline V. *Exploring the possibility of a hipSYCL-based implementation of oneAPI // IWOCL '22: Proceedings of the 10th International Workshop on OpenCL* (May 10–12, 2022, Bristol, United Kingdom), New York: ACM.– 2022.– ISBN 978-1-4503-9658-5.– id. 10.– 12 pp. doi ↑172, 243

[166] Sakiotis I., Arumugam K., Paterno M., Ranjan D., Terzić B., Zubair M. // *High Performance Computing*, 38th International Conference ISC High Performance 2023 (May 21–25, 2023, Hamburg, Germany), Lecture Notes in Computer Science.– vol. **13948**, Cham: Springer.– 2023.– ISBN 978-3-031-32040-8.– Pp. 339–358. doi ↑172

[167] Reguly I. Z., Owenson A. M. B., Powell A., Jarvis S. A., Mudalige G. R. *Under the hood of SYCL — an initial performance analysis with an unstructured-mesh CFD application // High Performance Computing*, 36th International Conference ISC High Performance 2021 (June 24–July 2, 2021, Virtual Event), Lecture Notes in Computer Science.– vol. **12728**, Cham: Springer.– 2021.– ISBN 978-3-030-78712-7.– Pp. 391–410. doi ↑172

[168] Walden A. C., Zubair M., Nielsen E. J. *Performance and portability of a linear solver across emerging architectures // Accelerator Programming Using Directives*, 7th International Workshop WACCPD 2020 (November 20, 2020, Virtual Event), Lecture Notes in Computer Science.– vol. **12655**, Cham: Springer.– 2021.– ISBN 978-3-030-74223-2.– Pp. 61–79. doi ↑172, 242

[169] Zubair M., Stone C., Walden A., Nielsen E. *Experiences in Moving CUDA-Optimized Kernels to Intel GPUs using oneAPI*, SC21.– 2021.– 22 pp. URL ↑172

[170] *Nvidia HPC Compilers User's Guide*, DU-09862-001-V2023, version 2023.– 173 pp. URL ↑172

[171] Karp M., Massaro D., Jansson N., Hart A., Wahlgren J., Schlatter P., Markidis S. *Large-scale direct numerical simulations of turbulence using GPUs and modern Fortran* // The International Journal of High Performance Computing Applications.– 2023.– Vol. **37**.– No. 5.– Pp. 487–502. ↑173, 266, 267

[172] *OpenMP Compilers & Tools*.– OpenMP.– 2023. ↑173

[173] Cojean T., Tsai Y. H. M., Anzt H. *Ginkgo—A math library designed for platform portability* // Parallel Computing.– 2022.– Vol. **111**.– id. 102902. ↑173, 199, 200, 201, 205

[174] Fuentes J., López D., González S. *Teaching heterogeneous computing using DPC++* // *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (30 May 2022–03 June 2022, Lyon, France).– IEEE.– 2022.– ISBN 978-1-6654-9747-3.– Pp. 354–360.– id. 102902. ↑173

[175] Fridman Y., Tamir G., Oren G. *Portability and scalability of OpenMP offloading on state-of-the-art accelerators* // *High Performance Computing*, ISC High Performance 2023 International Workshops (May 21–25, 2023, Hamburg, Germany), Lecture Notes in Computer Science.– vol. **13999**, Cham: Springer.– 2023.– ISBN 978-3-031-40842-7.– Pp. 378–390. ↑173

[176] Reguly I. Z. *Evaluating the performance portability of SYCL across CPUs and GPUs on bandwidth-bound applications* // *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W'23 (November 12–17, 2023, Denver, CO, USA), New York: ACM.– 2023.– ISBN 979-8-4007-0785-8.– Pp. 1038–1047. ↑173, 174, 177, 243

[177] *SPEC CPU2017 Results*.– Standard Performance Evaluation Corporation. ↑174

[178] Solis-Vasquez L., Mascarenhas E., Koch A. *Experiences migrating CUDA to SYCL: A molecular docking case study* // *Proceedings of the 2023 International Workshop on OpenCL*, IWOCL'23 (April 18–20, 2023, Cambridge, United Kingdom), New York: ACM.– 2023.– ISBN 979-8-4007-0745-2.– Pp. 1–11.– id. 15. ↑174, 175

[179] Nguyen P., Nayak P., Anzt H. // *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '23 (November 12–17, 2023, Denver, CO, USA), New York: ACM.– 2023.– ISBN 979-8-4007-0785-8.– Pp. 1048–1058. arXiv 2308.08417 ↑175

[180] Morgan T. P. *One New Feature For Intel's HPC Compute Engines: Contrition*, The Next Platform.– Stackhouse Publishing.– 2022. ↑176

[181] Morgan T. P. *Aurora In A Socket: What Intel's Falcon Shores XPU Might Do*, The Next Platform.– Stackhouse Publishing.– 2022. ↑176

[182] *Nvidia Parallel Thread Execution ISA*, Release 8.4.– Nvidia.– 2024.– 598 pp. ↑178

[183] *Inline PTX Assembly in CUDA*.– Vol. **1**, Release 12.4.– Nvidia.– 2024.– 16 pp. URL ↑178

[184] *Nvidia Ampere GA102 GPU Architecture*, Updated with Nvidia RTX A6000 and Nvidia A40 Information, V2.0.– Nvidia.– 2021.– 53 pp. URL ↑178

[185] *GPU Specs Database*, A reference list of most graphics cards released in recent years.– TechPowerUp. URL ↑178, 179, 183, 184, 212, 231, 232, 237, 263

[186] Osama M., Merrill D., Cecka C., Garland M., Owens J. D. // *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPoPP'23 (25 February 2023–1 March 2023, Montreal, QC, Canada), New York: ACM.– 2023.– ISBN 979-8-4007-0015-6.– Pp. 429–431. doi arXiv 2301.03598 ↑183

[187] *Nsight Compute*, The User Guide for Nsight Compute, v2024.1.1.– Nvidia. URL ↑184

[188] Li A., Song S. L., Wijtvliet M., Kumar A., Corporaal H. *SFU-driven transparent approximation acceleration on GPUs* // *Proceedings of the 2016 International Conference on Supercomputing*, ICS'16 (June 1–3, 2016, Istanbul, Turkey), New York: ACM.– 2016.– ISBN 978-1-4503-4361-9.– Pp. 1–14.– id. 15. doi ↑185

[189] Jia Z., Maggioni M., Staiger B., Scarpazza D. P. *Dissecting the Nvidia volta GPU architecture via microbenchmarking*.– 2018.– 66 pp. arXiv 1804.06826 ↑185

[190] Choquette J., Gandhi W., Giroux O., Stam N., Krashinsky R. *Nvidia A100 tensor core GPU: performance and innovation* // IEEE Micro.– 2021.– Vol. **41**.– No. 2.– Pp. 29–35. doi ↑185, 188, 206, 209

[191] *Multi-Instance GPU User Guide*, RN-08625-v2.0.– Nvidia.– 2024.– iv+53 pp. URL ↑186

[192] *Nvidia A100 80GB PCIe GPU*, Product Brief, PB-10577-001_v03.– Nvidia.– 2022. URL ↑186

[193] Li A., Song S. L., Chen J., Li J., Liu X., Tallent N. R., Barker K. J. *Evaluating modern GPU interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect* // IEEE Transactions on Parallel and Distributed Systems.– 2019.– Vol. **31**.– No. 1.– Pp. 94–110. doi ↑187

[194] Lutz C., Breß S., Zeuch S., Rabl T., Markl V. *Pump up the volume: Processing large data on GPUs with fast interconnects* // *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD'20 (June 14–19, 2020, Portland, OR, USA), New York: ACM.– 2020.– ISBN 978-1-4503-6735-6.– Pp. 1633–1649. doi ↑187

[195] *Nvidia NVSwitch*, Technical Overview.– Nvidia.– 2018.– 8 pp. URL ↑187

[196] Špeťko M., Vysocký O.,Jansík B.,Říha L. *DGX-A100 Face to Face DGX-2—performance, power and thermal behavior evaluation* // Energies.– 2021.– Vol. **14**.– No. 2.– id. 376.– 18 pp. doi ↑188, 189

[197] Choi Y. R., Nikolskiy V., Stegailov V. *Matrix-matrix multiplication using multiple GPUS connected by Nvlink* // *2020 Global Smart Industry Conference (GloSIC)* (17–19 November 2020, Chelyabinsk, Russia).– IEEE.– 2020.– ISBN 9781728180755.– Pp. 354–361. doi ↑189

[198] Manathunga M., Jin C., Cruzeiro V. W. D., Miao Y., Mu D., Arumugam K., Keipert K., Aktulga H. M., Merz jr K. M., Götz A. W. *Harnessing the power of multi-GPU acceleration into the quantum interaction computational kernel program* // Journal of Chemical Theory and Computation.– 2021.– Vol. **17**.– No. 7.– Pp. 3955–3966. ↑189, 207

[199] Choi Y. R., Nikolskiy V., Stegailov V. *Matrix-matrix multiplication using multiple GPUS connected by Nvlink* // *2020 Global Smart Industry Conference (GloSIC)* (17–19 November 2020, Chelyabinsk, Russia).– IEEE.– 2020.– ISBN 9781728180755.– Pp. 354–361. ↑189

[200] Choi Y. R., Stegailov V. *Multi-GPU GEMM algorithm performance analysis for Nvidia and AMD GPUs connected by NVLink and PCIe* // *Mathematical Modeling and Supercomputer Technologies: 22nd International Conference*, Revised Selected Papers, 22nd International Conference MMST 2022 (November 14–17, 2022, Nizhny Novgorod, Russia), Cham: Springer.– 2022.– ISBN 978-3-031-24144-4.– Pp. 281–292. ↑189

[201] *Nvidia DGX A100*, Datasheet.– Nvidia.– 2023.– 2 pp. ↑189

[202] *Nvidia DGX A100*, User Guide, DU-09821-001_v01.– Nvidia.– 2023.– 126 pp. ↑189

[203] *Nvidia DGX Platform*, Cloud & Data Center.– Nvidia.– 2024. ↑189

[204] *Nvidia DGX SuperPOD: Scalable Infrastructure for AI Leadership*, Reference Architecture, RA-09950-001.– Nvidia.– 2021.– 30 pp. ↑189

[205] *Leonardo HPC System*, Technical info.– Leonardo Pre-exascale Supercomputer.– 2024. ↑190, 246

[206] *Perlmutter Architecture*, NERSC documentation.– NERSC. ↑190, 246

[207] *Nvidia HPC SDK*, Overview, Containers.– Nvidia. ↑190

[208] *Nvidia HPC SDK Version 24.3 Documentation*.– Nvidia.– 2024. ↑190, 191

[209] *CUDA LLVM Compiler*, Nvidia Developer.– Nvidia. ↑191

[210] Potluri S., Hamidouche K., Venkatesh A., Bureddy D., Panda K. *Efficient inter-node MPI communication using GPUDirect RDMA for InfiniBand clusters with Nvidia GPUs* // *2013 42nd International Conference on Parallel Processing* (01–04 October 2013, Lyon, France).– IEEE.– 2013.– ISBN 978-0-7695-5117-3.– Pp. 80–89. ↑191

[211] *Nvidia CUDA Fortran programming guide*, HPC SDK documentation, version 24.3.– Nvidia.– 2024. ↑193, 194

[212] Ruetsch G., Fatica M. *CUDA Fortran for scientists and engineers. Best practices for efficient CUDA Fortran programming*, 1st ed..– Morgan Kaufmann.– 2013.– ISBN 978-0-12-416970-8.– 338 pp. ↑194

[213] Oyanagi S. *HPE Cray MPI update*, Slides, SC'21 ANL MPICH BOF (November 17, 2021).– 6 pp. ↑195

[214] *Developing a Linux kernel module using GPUDirect RDMA*, v12.4.– Nvidia.– 2024.– 48 pp. ↑195

[215] *MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, RoCE, and Slingshot*, User guide, Version 2.3.7/ ed. Dhabaleswar K..– NBCL.– 2024. URL ↑195

[216] Bhalachandra S., Austin B., Williams S., Wright N. J. *Understanding the impact of input entropy on FPU, CPU, and GPU power.*– 2022.– 6 pp. arXiv 2212.08805 ↑196

[217] *Nvidia Docs. Matrix multiplication background*, User's Guide, Deep learning, DU-09799-001_v001.– Nvidia.– 2023.– 17 pp. URL ↑196, 201

[218] Mukunoki D., Ozaki K., Ogita T., Imamura T. *DGEMM using tensor cores, and its accurate and reproducible versions* // *High Performance Computing*, 35th International Conference, ISC High Performance 2020 (June 22–25, 2020, Frankfurt/Main, Germany), Lecture Notes in Computer Science.– vol. **12151**, Cham: Springer.– 2020.– ISBN 978-3-030-50742-8.– Pp. 230–248. doi ↑196

[219] Fasi M., Higham N. J., Mikaitis M., Pranesh S. *Numerical behavior of Nvidia tensor cores* // PeerJ Computer Science.– 2021.– id. 7e330. doi ↑197

[220] Fasi M., Higham N. J., Lopez F., Mary T., Mikaitis M. *Matrix multiplication in multiword arithmetic: error analysis and application to GPU tensor cores* // SIAM Journal on Scientific Computing.– 2023.– Vol. **45**.– No. 1.– Pp. C1–C19. doi ↑197

[221] Li S., Osawa K., Hoefler T. *Efficient quantized sparse matrix operations on tensor cores.*– 2022.– 13 pp. arXiv 2209.06979 ↑197

[222] Tao D., Tiuan J. *Performance of Sample CUDA Benchmarks on Nvidia Ampere A100 vs Tesla V100.*– GitHub Inc..– 2021. URL ↑197, 201

[223] *CUDA Samples*, Reference Manual, TRM-06704-001_v11.2.– 142 pp. URL ↑197

[224] *All ACCEL Results Published by SPEC.*– Standard Performance Evaluation Corporation.– 2024. URL ↑

[225] Brunst H., Chandrasekaran S., Ciorba F., Hagerty N., Henschel R., Juckeland G., Li J., Vergara V. G. M., Wienke S., Zavala M. // *2022 22nd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)* (16-19 May 2022, Taormina, Italy).– 2022.– ISBN 978-1-6654-9956-9.– Pp. 675–684. doi arXiv 2203.06751 ↑197, 247

[226] *All HPC2021 Results Published by SPEC.*– SPEC.– 2024. URL ↑197, 224

[227] Svedin M., Chien S. W. D., Chikafa G., Jansson N., Podobas A. *Benchmarking the Nvidia GPU lineage: From early K80 to modern A100 with asynchronous memory transfers* // *Proceedings of the 11th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies* (June 21–23, 2021, Online Germany), New York: ACM.– 2021.– ISBN 978-1-4503-8549-7.– id. 9.– 6 pp. doi arXiv 2106.04979 ↑198

[228] Zhang L., Wahib M., Chen P., Meng J., Wang X., Endo T., Matsuoka S. *Persistent kernels for iterative memory-bound GPU applications.*– 2022. arXiv 2204.02064 ↑198

[229] Špeťko M., Vysocký O., Jansík B., Říha L. *DGX-A100 face to face DGX-2—performance, power and thermal behavior evaluation* // Energies.– 2021.– Vol. **14**.– No. 2.– Pp. 376. doi ↑198

[230] Jansik B. *Mandelbrot benchmark*.– Accessed 15.10.2023. URL ↑198

[231] Mudigere D., Hao Y., Huang J., Jia Z., Tulloch A., Sridharan S., Liu X., Ozdal M., Nie J., Park J., Luo L., Yang J. A., Gao L., Ivchenko D., Basant A., Hu Y., Yang J., Ardestani E. K., Wang X., Komuravelli R., Ching-Chu H., Yilmaz S., Li H., Qian J., Feng Z., Ma Y., Yang J., Wen E., Li H., Yang L., Sun C., Zhao W., Melts D., Dhulipala K., KKishore R., Graf T., Eisenman A., Matam K. K., Gangidi A., Chen G. J., Krishnan M., Nayak A., Nair K., Muthiah B., Khorashadi M., Bhattacharya P., Lapukhov P., Naumov M., Mathews A., Qiao L., Smelyanskiy M., Jia B., Rao V. *Software-hardware co-design for fast and scalable training of deep learning recommendation models* // *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22 (June 18–22, 2022, New York, USA), New York: ACM.– 2022.– ISBN 978-1-4503-8610-4.– Pp. 993–1011. doi arXiv 2104.05158 ↑198, 200, 201, 202, 212

[232] Deakin T., Price J., Martineau M., McIntosh-Smith S. *Evaluating attainable memory bandwidth of parallel programming models via BabelStream* // International Journal of Computational Science and Engineering.– 2018.– Vol. **17**.– No. 3.– Pp. 247–262. doi URL ↑198, 199

[233] McCalpin J. D. *Memory bandwidth and machine balance in current high performance computers*, IEEE computer society technical committee on computer architecture (TCCA) newsletter.– 1995.– 7 pp. URL ↑199

[234] Tsai Y. M., Cojean T., Anzt H. *Evaluating the performance of Nvidia's A100 Ampere GPU for sparse and batched computations* // *2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)* (12 November 2020, GA, USA).– IEEE.– 2020.– ISBN 978-0-7381-1048-6.– Pp. 26–38. doi arXiv 2008.08478 ↑199, 200, 201, 203, 204, 205

[235] Tsai Y. M., Cojean T., Anzt H. *Evaluating the performance of Nvidia's A100 Ampere GPU for sparse linear algebra computations*.– 2020.– 9 pp. arXiv 2008.08478 ↑199, 200, 201, 205

[236] Hammond J. R., Deakin T., Cownie J., McIntosh-Smith S. *Benchmarking Fortran DO CONCURRENT on CPUs and GPUs using BabelStream* // *2022 IEEE/ACM International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)* (13-18 November 2022, Dallas, TX, USA).– IEEE.– 2022.– ISBN 978-1-6654-5185-7.– Pp. 82–99. doi ↑199, 200, 247

[237] Balos C. J. *Reproduced computational results report for "Ginkgo": a modern linear operator algebra framework for high performance computing"* // ACM Transactions on Mathematical Software (TOMS).– 2022.– Vol. **48**.– No. 1.– id. 3.– 7 pp. doi ↑200, 201

[238] Grützmacher T., Anzt H., Quintana-Ortí E. S. *Using Ginkgo's memory accessor for improving the accuracy of memorybound low precision BLAS* // Software: Practice and Experience.– 2021.– Vol. **53**.– No. 1, Special Issue: New Trends in High-Performance Computing: Software Systems and Applications.– Pp. 81–98. ↑200, 201, 204, 205

[239] *Mixbench*, A benchmark suite for GPUs on mixed operational intensity kernels, Openbenchmarking.org.– Phoronix Media.– 2024. ↑200

[240] Dong T., Haidar A., Luszczek P., Tomov S., Abdelfattah A., Dongarra J. *Magma batched: A batched blas approach for small matrix factorizations and applications on gpus* // Journal of Latex class files.– 2015.– Vol. **14**.– No. 8.– Accessed 15.10.2023. ↑200

[241] Abdelfattah A., Tomov S., Dongarra J. *Batch QR factorization on GPUs: design, optimization, and tuning*, Computational Science – ICCS 2022 (June 21–23, 2022, London, UK), Lecture Notes in Computer Science.– vol. **13350**, Cham: Springer.– 2022.– ISBN 978-3-031-08750-9.– Pp. 60–74. ↑201, 204, 247, 248

[242] Abdelfattah A., Barra V., Beams N., Bleile R., Brown J., Jean-Camier S., Carson R., Chalmers N., Dobrev V., Dudouit Y., Fischer P., Karakus A., Kerkemeier S., Kolev T., Yu-Lan H., Merzari E., Min M., Phillips M., Rathnayake T., Rieben R., Stitt T., Tomboulides A., Tomov S., Tomov V., Vargas A., Warburton T., Weiss K. *GPU algorithms for efficient exascale discretizations* // Parallel Computing.– 2021.– Vol. **108**.– id. 102841.– 10 pp. ↑201, 202, 249

[243] Dong T., Dobrev V., Kolev T., Rieben R., Tomov S., Dongarra J. *A step towards energy efficient computing: Redesigning a hydrodynamic application on CPU-GPU* // *2014 IEEE 28th International Parallel and Distributed Processing Symposium* (19–23 May 2014, Phoenix, AZ, USA).– IEEE.– 2014.– ISBN 978-1-4799-3800-1.– Pp. 972–981. ↑202

[244] Abdelfattah A., Baboulin M., Dobrev V., Dongarra J., Earl C., Falcou J., Haidar A., Karlin I., Kolev T., Masliah I., Tomov S. *High-performance tensor contractions for GPUs* // Procedia Computer Science.– 2016.– Vol. **80**.– Pp. 108–118. ↑202

[245] Heinecke A., Henry G., Hutchinson M., Pabst H. *LIBXSMM: accelerating small matrix multiplications by runtime code generation* // *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC'16 (13–18 November 2016, Salt Lake City, UT).– IEEE.– 2016.– ISBN 978-1-4673-8815-3.– Pp. 981–991. ↑202

[246] Bethune I., Reid F., Lazzaro A. *CP2K Performance from Cray XT3 to XC30.*– Cray User Group (CUG).– 2014.– 11 pp. ↑202

[247] Sedova A., Tharrington A., Messer B. *Portability in scientific computing: The molecular dynamics non-bonded forces calculation as a case study.*– 2018.– 25 pp. ↑202

[248] Waugh H., McIntosh-Smith S. *On the use of BLAS libraries in modern scientific codes at scale* // *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, 17th Smoky Mountains Computational Sciences and Engineering Conference SMC 2020 (August 26–28, 2020, Oak Ridge, TN, USA), Communications in Computer and Information Science.– vol. **1315**, Cham: Springer.– 2020.– ISBN 978-3-030-63392-9.– Pp. 67–79. doi ↑202

[249] Mijić N., Davidović D. *Batched matrix operations on distributed GPU's with application in theoretical physics* // *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)* (23–27 May 2022, Opatija, Croatia).– IEEE.– 2022.– ISBN 978-953-233-103-5.– Pp. 293–299. doi arXiv 2203.09353 ↑203

[250] Stylianou C., Weiland M. *Optimizing sparse linear algebra through automatic format selection and machine learning.*– 2023.– 10 pp. arXiv 2303.05098 ↑205

[251] Pascuzzi V. R., Goli M. *Benchmarking a proof-of-concept performance portable SYCL-based fast Fourier transformation Library* // *Proceedings of the 10th International Workshop on OpenCL*, International Workshop on OpenCL IWOCL'22 (May 10–12, 2022, Bristol, United Kingdom), New York: ACM.– 2022.– ISBN 978-1-4503-9658-5.– id. 20.– 9 pp. doi arXiv 2203.09384 ↑205, 247

[252] Tolmachev D. *VkFFT-a performant, cross-platform and open-source GPU FFT library* // IEEE Access.– 2023.– Vol. **11**.– Pp. 12039–12058. doi ↑205

[253] Li B., Cheng S., Lin J. *tcFFT: Accelerating half-precision FFT through tensor cores.*– 2021.– 10 pp. arXiv 2104.11471 ↑205, 206

[254] Hagerty N., Melesse Vergara V., Tharrington A. *Studying performance portability of LAMMPS across diverse GPU-based platforms*, S2 World 2020. CUG 2021 & 2022. PN_HCP. HeteroPar 2022 // Concurency and computation. Practice and Experience.– 2023.– Vol. **35**.– No. 28.– id. e7895. doi ↑206, 229, 257

[255] Poenaru A., Lin W.-C., McIntosh-Smith S. *A performance analysis of modern parallel programming models using a compute-bound application* // *ISC High Performance 2021: High Performance Computing*, Lecture Notes in Computer Science.– vol. **12728**, Cham: Springer.– 2021.– ISBN 978-3-030-78712-7.– Pp. 332–350. doi ↑206

[256] Solis-Vasquez L., Tillack A. F., Santos D.,-Martins, Koch A., Le S.,Grand, Forli S. *Benchmarking the performance of irregular computations in AutoDock-GPU molecular docking* // Parallel Computing.– 2022.– Vol. **109**.– id. 102861.– 12 pp. doi ↑206

[257] Manathunga M., Aktulga H. M., Götz A. W., Merz K. M. *Quantum mechanics/molecular mechanics simulations on Nvidia and AMD graphics processing units* // J. Chem. Inf. Model.– 2023.– Vol. **63**.– No. 3.– Pp. 711-717. doi ↑207

[258] Cruzeiro V. W. D., Manathunga M., Merz K. M., Götz A. W. *Open-source multi-GPU-accelerated QM/MM simulations with AMBER and QUICK* // J. Chem. Inf. Model.– 2021.– Vol. **61**.– No. 5.– Pp. 2109–2115. doi ↑207

[259] Shajan A., Manathunga M., Götz A. W., Merz K. M. Jr. *Geometry optimization: A comparison of different open-source geometry optimizers* // J. Chem. Theory Comput.– 2023.– Vol. **19**.– No. 21.– Pp. 7533–7541. ↑207

[260] Williams-Young D. B., Asadchev A., Popovici D. T., Clark D., Waldrop J., Windus T., Valeev E. F., de Jong W. A. *Distributed memory, GPU accelerated Fock construction for hybrid, Gaussian basis density functional theory* // The Journal of Chemical Physics.– 2023.– Vol. **158**.– No. 23.– id. 234104. ↑207

[261] Kim I., Jeong D., Won-Son J., Hyung-Kim J., Rhee Y. M., Jung Y., Choi H., Yim J., Jang I., Kim D. S. *Kohn-Sham time-dependent density functional theory on the massively parallel GPUs* // npj Computational Materials.– 2023.– Vol. **9**.– id. 81.– 12 pp. ↑208

[262] Siegel A., Draeger E., Deslippe J., Evans T. M., Francois M., Germann T., Martin D., Hart W. *Application Results on Early Exascale Hardware*, No. ORNL/TM-2022/2437.– Oak Ridge, TN (US): Oak Ridge National Lab.(ORNL).– 2022. ↑208, 237, 245, 246, 247, 248, 250, 251, 257, 258

[263] Dang G., Liu S., Guo T., Dang J., Li X. *Direct numerical simulation of compressible turbulence accelerated by graphics processing unit: An open-source high accuracy accelerated computational fluid dynamic software* // Physics of Fluids.– 2022.– Vol. **34**.– No. 12.– id. 126106. ↑208

[264] Min M., Tomboulides A. *Simulating Atmospheric Boundary Layer Turbulence with Nek5000/RS*, No. ANL-22/79.– Argonne, IL: Argonne National Lab.(ANL).– 2022.– 34 pp. ↑209

[265] Fischer P., Kerkemeier S., Min M., Yu-Lan H., Phillips M., Rathnayake T., Merzari E., Tomboulides A., Karakus A., Chalmers N., Warburton T. *NekRS, a GPU-accelerated spectral element Navier–Stokes solver* // Parallel Computing.– 2022.– Vol. **114**.– id. 102982.– 13 pp. ↑209, 249

[266] *FUN3D is a Computational Fluid Dynamics (CFD) suite of tools actively developed at NASA that benefits Aeronautics, Space Technology, and Exloration by modeling fluid flow*, 14.0.2-16d1333.– NASA Official: David P. Lockard. ↑209

[267] Nastac G., Walden A., Nielsen E. J., Frendi K. *Implicit thermochemical nonequilibrium flow simulations on unstructured grids using GPUs*, AIAA Scitech 2021 Forum (11–15, 19-21 January 2021, virtual event).– 2021.– id. AIAA 2021-0159. ↑209

[268] Nastac G., Walden A., Wang L., Nielsen E. J., Liu Y., Opgenorth M., Orender J., Zubair M. *A multi-architecture approach for implicit computational fluid dynamics on unstructured grids*, AIAA Scitech 2023 Forum (23–27 January 2023, National Harbor, MD & Online).– 2023.– id. AIAA 2023-1226. ↑209, 253

[269] Pasquariello V., Bunk Y., Eberhardt S., Pei-Huang H., Matheis J., Ugolotti M., Hickel S. *GPU-accelerated simulations for eVTOL aerodynamic analysis*, AIAA Scitech 2023 Forum (23–27 January 2023, National Harbor, MD & Online).– 2023.– id. AIAA 2023-2107. ↑209

[270] Regev T., Nestmann J., Garzuzi A., Greenblatt D., Frankel S. *GPU-accelerated high-fidelity implicit large eddy simulations of coanda cylinder flow instabilities*, AIAA Scitech 2023 Forum (23–27 January 2023, National Harbor, MD & Online).– 2023.– id. AIAA 2023-0272. ↑209

[271] Kakumani H. C. V., Chamarthi A. S., Hoffmann N., Frankel S. H. *GPU-accelerated numerical study of temperature effects in choked under-expanded supersonic jets*, AIAA SCITECH 2023 Forum (23–27 January 2023, National Harbor, MD & Online).– 2023.– id. AIAA 2023-0976. ↑210

[272] Sitaraman J., Jude D. *Development of GPGPU capable multi-solver overset methods*, AIAA SCITECH 2023 Forum (23–27 January 2023, National Harbor, MD & Online).– 2023.– id. AIAA 2023-0042. ↑210

[273] Mortazawy M., Rao M., Jilesen J., Work D., Shock R. *Early Stage Vehicle Aerodynamics Development using a GPU Based LBM CFD Solver*, SAE Technical Paper No2023-01-0560.– 2003.– 7 pp. ↑210

[274] Kummerländer A., Dorn M., Frank M., Krause M. J. *Implicit propagation of directly addressed grids in lattice Boltzmann methods* // Concurrency and Computation: Practice and Experience.– 2023.– Vol. **35**.– No. 8.– id. e7509. ↑210

[275] De Vanna F., Avanzi F., Cogo M., Sandrin S., Bettencourt M., Picano F., Benini E. *URANOS: A GPU accelerated Navier-Stokes solver for compressible wall-bounded flows* // Computer Physics Communications.– 2023.– id. 108717.– 18 pp. ↑210

[276] Chandravamsi H., Chamarthi A. S., Hoffmann N., Frankel S. H. *On the application of gradient based reconstruction for flow simulations on generalized curvilinear and dynamic mesh domains* // Computers & Fluids.– 2023.– id. 105859.– 28 pp. ↑210

[277] Mattson P., Cheng C., Coleman C., Diamos G., Micikevicius P., Patterson D., Tang H., Gu-Wei Y., Bailis P., Bittorf V., Brooks D., Chen D., Dutta D., Gupta U., Hazelwood K., Hock A., Huang X., Ike A., Jia B., Kang D., Kanter D., Kumar N., Liao J., Ma G., Narayanan D., Oguntebi T., Pekhimenko G., Pentecost L., Reddi V. J., Robie T., John T. S., Tabaru T., Carole-Wu J., Xu L., Yamazaki M., Young C., Zaharia M. *MLPerf training benchmark* // *Proceedings of Machine Learning and Systems*.– V. 2, MLSys 2020, eds. I. Dhillon, D. Papailiopoulos, V. Sze.– 2020.– Pp. 336–349. arXiv:1910.01500 ↑210

[278] *MLPerf Training v.2.1 Results*.– ML Commons. ↑210, 211, 225

[279] *MLPerf Training HPC v2.0 results*.– ML Commons. ↑211

[280] *Nvidia NVLink and NVLink Switch*, Cloud & Data Center.– Nvidia. ↑214

[281] *PRE-EOS 128 NODE DGX SuperPOD — Nvidia DGX H100, Xeon Platinum 8480C 56C 2GHZ, Nvidia H100 Tensor core GPUs, Nvidia ConnectX-7 NDR 400G Infiniband*.– Top500.org.– 2023. ↑216

[282] *Kestrel System Configuration*, National Renewable Energy Laboratory Computing Systems.– Alliance for Sustainable Energy.– 2023. ↑216

[283] *G242-P36 (rev. 100)*, Продукция.– GIGA-BYTE Technology.– 2024. URL ↑216
[284] *Nvidia Grace CPU Superchip Whitepaper*, V1.1.– Nvidia.– 2024.– 20 pp. URL ↑217, 218, 219
[285] *Nvidia Grace CPU Superchip*, Datasheet.– Nvidia.– 2024.– 3 pp. URL URL ↑217
[286] *Arm Neoverse V2 Core Technical Reference Manual*.– Accessed 15.10.2023. URL ↑217
[287] *Nvidia GH200 Grace Hopper Siperchip Architecture*, V1.01.– Nvidia.– 2024.– 39 pp. URL ↑219, 220, 221, 222
[288] *H263-V11*, Products, rev. LAW1.– Giga-Byte Technology. URL ↑221
[289] Petty H., Goldwasser I., Desale P. *One Giant Superchip for LLMs, Recommenders, and GNNs: Introducing Nvidia GH200 NVL32*, Technical blog, Nvidia developer.– 2023. URL ↑222
[290] *Nvidia BlueField-3 networking platform*, Datasheet.– Nvidia.– 2023.– 2 pp. URL ↑222
[291] *CUDA Python Manual*, v. 12.4.0.– Nvidia.– 2024. URL ↑222
[292] *Nvidia NVVM IR Specification*, V. 12.4.– Nvidia.– 2024.– 80 pp. URL ↑223
[293] Choquette J. *Nvidia Hopper GPU: scaling performance // 2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 August 2022, Cupertino, CA, USA).– IEEE.– 2022.– ISBN 978-1-6654-6028-6.– Pp. 1–46. doi ↑224
[294] *Amber22: pmemd.cuda performance information*, The Amber project/ ed. Kollman P..– 2023. URL ↑225
[295] *MLPerf Training v.3.0 Results*.– MLCommons.– 2024. URL ↑225, 227, 228
[296] *AMD Instinct™ MI100 Accelerators*, Overview.– AMD.– 2020. URL ↑229
[297] *AMD Instinct™ Accelerators*, Products.– AMD. URL ↑229
[298] *MLPerf Inference Datacenter v.3.1 Results*.– MLCommons. URL ↑225
[299] Smith A., James N. *AMD Instinct™ MI200 series accelerator and node architectures // 2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 August 2022, Cupertino, CA, USA).– IEEE.– 2022.– ISBN 978-1-6654-6028-6.– 23 pp. doi ↑229, 233, 235, 236, 237, 239, 240, 241
[300] *AMD Instinct™ MI210 Accelerators*.– AMD.– 2022. URL ↑232, 238
[301] *AMD Instinct™ MI250 drivers & support*.– Accessed 15.10.2023. URL ↑232, 238
[302] *AMD Instinct™ MI250X drivers & support*.– Accessed 15.10.2023. URL ↑232, 238
[303] *Frontier User Guide*.– Oak Ridge National Laboratory.– 2024. URL ↑232, 239, 241, 246
[304] *Introducing AMD CDNA architecture*.– AMD.– 2020.– 11 pp. URL ↑232, 235
[305] *Introducing AMD CDNA 2 Architecture*, White paper, AMD Instinct MI200.– Advanced Micro Device.– 2021.– 17 pp. URL ↑232, 233, 234, 235, 239, 240, 241
[306] *"AMD Instinct MI200" instruction set architecture*, Reference Guide, AMD Instinct MI200.– Advanced Micro Devices.– 2021.– 275 pp. URL ↑235

[307] Sitaraman C., Chalmers N., Malaya N., McDougal D., O'Reilly O, van Oostrum R. *AMD matrix cores*, GPU open, AMD Labs notes/ ed. Greathouse R..– Advanced Micro Devices.– 2023. ↑235

[308] Pearson C. *Interconnect bandwidth heterogeneity on AMD MI250x and Infinity fabric.*– 2023. arXiv 2302.14827 ↑240

[309] Gates M., YarKhan A., Sukkari D., Akbudak K., Cayrols S., Bielich D., Abdelfattah A., Farhan M. A., Dongarra J. *Portable and efficient dense linear algebra in the beginning of the exascale era // 2022 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (13–18 November 2022, Dallas, TX, USA).– IEEE.– 2022.– ISBN 978-1-6654-6021-7.– Pp. 36–46. ↑239

[310] Melesse Vergara V. G., Budiardja R. D., Davis M. J., Ezell M. A., Hanley J. A., Zimmer C. J., Brim M. J., Elwasif W. R., Dietz D. T. *Approaching the final Frontier: lessons learned from the deployment of HPE/Cray EX Spock and Crusher supercomputers // Cray User Group 2022 Proceedings*, CUG (May 2, 2022 – May 5, 2022).– Oak Ridge National Lab. (ORNL).– 2022. ↑241

[311] *AMD Instinct Accelerator Qualified Servers Q4 2022*, Reference Guide, AMD Instinct MI200.– Advanced Micro Devices.– 2022.– 3 pp. ↑241

[312] *Welcome to AMD ROCm Platform*, Revision e2b73a17.– Advanced Micro Devices.– 2021. ↑241, 242

[313] *AMD ROCm documentation*, V. 6.1.1.– 2024. ↑242, 243, 244

[314] Kondratyuk N., Nikolskiy V., Pavlov D., Stegailov V. *GPU-accelerated molecular dynamics: State-of-art software performance and porting from Nvidia CUDA to AMD HIP* // The International Journal of High Performance Computing Applications.– 2021.– Vol. **35**.– No. 4.– Pp. 312–324. ↑242, 256

[315] Charrier D. et al. *GPUFORT: S2S translation tool for CUDA Fortran and Fortran+X in the spirit of hipify*, AMD ROCm Software.– GitHub Inc.. ↑243

[316] *AMD ROCm documentation.*– AMD.– 2024. ↑243

[317] Khorassani K. S., Chen-Chen C., Ramesh B., Shafi A., Subramoni H., Panda D. K. *High Performance MPI over the Slingshot Interconnect* // Journal of Computer Science and Technology.– 2023.– Vol. **38**.– No. 1.– Pp. 128–145. ↑244

[318] *Welcome to the LUMI supercomputer user guide.*– LUMI (Large Unified Modern Infrastructure) consortium. ↑246

[319] *Crusher Quick-Start Guide.*– Oak Ridge National Laboratory.– 2024. ↑246

[320] *Spock Quick-Start Guide.*– Oak Ridge National Laboratory.– 2023. ↑246

[321] *ThetaGPU Machine Overview.*– Argonne National Laboratory.– Accessed 15.10.2023. ↑246

[322] *Polaris Machine Overview.*– Argonne National Laboratory.– 2023. ↑246

[323] *Summit User Guide*, Alpine.– Oak Ridge National Laboratory.– 2023. ↑246

[324] Heroux M. A. et al. *ECP software technology capability assessment report*, No. ORNL/TM-2022/2651, V3.0.– Oak Ridge National Lab.– 2022.– 237 pp. URL ↑246, 255

[325] Sathyanarayana S., Bernardini M., Modesti D., Pirozzoli S., Salvadore F. *High-speed turbulent flows towards the exascale: STREAmS-2 porting and performance.*– 2023.– 32 pp. arXiv 2304.05494 ↑247, 262, 263, 264

[326] Müller A., Schmidt B., Membarth R., Leißa R., Hack S. *AnySeq/GPU: A novel approach for faster sequence alignment on GPUs.*– 2022.– 11 pp. arXiv 2205.07610 ↑248

[327] Manathunga M., Aktulga H. M., Götz A. W., Merz K. M. jr. *Quantum mechanics/molecular mechanics simulations on Nvidia and AMD Graphics Processing Units* // Journal of Chemical Information and Modeling.– 2023.– Vol. **63**.– No. 3.– Pp. 711–717. doi ↑249, 250

[328] Kolev T., Fischer P., Austin A. P., Barker A. T., Beams N., Brown J., Jean-Camier S., Chalmers N., Dobrev V., Dudouit Y., Ghaffary L., Kerkemeier S., Yu-Lan H., Merzari E., Min M., Pazner W., Rathnayake T., Shephard M. S., Siboni M. H., Smith C. W., Thompson J. L., Tomov S., Warburton T. *High-order algorithmic developments and optimizations for large-scale GPU-accelerated simulations*, ECP Milestone Report: WBS 2.2.6.06, Milestone CEED-MS36.– 2021.– 51 pp. URL ↑250, 270

[329] Thavappiragasam M., Elwasif W., Sedova A. *Portability for GPU-accelerated molecular docking applications for cloud and HPC: can portable compiler directives provide performance across all platforms?.*– 2022.– 10 pp. arXiv 2203.02096 ↑251

[330] Stone C. P., Walden A., Zubair M., Nielsen J. *Accelerating unstructured-grid CFD algorithms on Nvidia and AMD GPUs* // *2021 IEEE/ACM 11th Workshop on Irregular Applications: Architectures and Algorithms (IA3)* (15 November 2021, St. Louis, MO, USA).– 2021.– ISBN 978-1-6654-1126-4.– Pp. 19–26. doi ↑251

[331] Pumma S., Vishnu A. *Semantic-aware lossless data compression for Deep Learning Recommendation Model (DLRM)* // *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)* (15 November 2021, St. Louis, MO, USA).– IEEE.– 2021.– ISBN 978-1-6654-1124-0.– Pp. 1–8. doi ↑251

[332] Han F., Kumar N. *HPC Application Performance on Dell PowerEdge R750xa Servers with the AMD Instinct MI210 Accelerator.*– Dell.– 2022. URL ↑252

[333] *AMD Instinct MI200 Series Accelerators Benchmarks.*– AMD.– 2024. URL ↑252, 256, 258, 259, 262

[334] Yu Y., Cai C., Wang J., Bo Z., Zhu Z., Zheng H. *Uni-dock: Gpu-accelerated docking enables ultralarge virtual screening* // Journal of Chemical Theory and Computation.– 2023.– Vol. **19**.– No. 11.– Pp. 3336–3345. doi ↑252

[335] Hao Y., Zhao X., Bao B., Berard D., Constable W., Aziz A., Liu X. *TorchBench: benchmarking PyTorch with High API surface coverage.*– 2023.– 13 pp. arXiv 2304.14226 ↑253

[336] Punniyamurthy K., Beckmann B. M., Hamidouche K. *Optimizing distributed ML communication with fused computation-collective operations.*– 2023.– 12 pp. arXiv 2305.06942 ↑253

[337] Guo Y., Lu L., Zhu S. *Novel accelerated methods for convolution neural network with matrix core* // The Journal of Supercomputing.– 2023.– Vol. **79**.– No. 17.– Pp. 19547–19573. doi ↑253

[338] Eassa A., Porter C. *Fueling high-performance computing with full-stack innovation*, Technical blog, Nvidia developer.– 2022. URL ↑253

[339] *Driving the Industry into the Exascale Era with AMD Instinct Accelerators*, AMD Community.– AMD.– 2022. URL ↑254

[340] Budiardja R. D., Berrill M., Eisenbach M., Jansen G. R., Joubert W., Nichols S., Rogers D. M., Tharrington A., Messer O. E. B., *Ready for the Frontier: preparing applications for the world's first exascale system* // *High Performance Computing*, ISC High Performance 2023, LNCS.– No. 13948, Cham: Springer.– 2023.– ISBN 978-3-031-32040-8.– Pp. 182–201. doi ↑254

[341] Wittwer F., Sauter N. K., Mendez D., Poon B. K., Brewster A. S., Holton J. M., Wall M. E., Hart W. E., Bard D. J., Blaschke J. P. *Accelerating X-ray tracing for exascale systems using Kokkos.*– 2022.– 6 pp. arXiv 2205.07976 ↑254

[342] Papatheodore T. *Frontier/Crusher node performance*, Frontier Training Workshop (February 16, 2023).– Oak Ridge National Laboratory.– 13 pp. URL ↑255, 256

[343] *MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, RoCE, and Slingshot*, NOWLAB: Network Based Computing Lab.– NBCL. URL ↑255

[344] Kurzak J., Malaya N., Klemm M., Hiew E. *Matrix Multiply Stress Test.*– GitHub inc..– 2023. URL ↑255

[345] Papatheodore T. *GPU XGEMM*, Benchmark.– 2023. URL ↑255

[346] *Enhancing LAMMPS simulations with AMD instinct accelerators: unleashing performance and scalability*, Solution Brief, High Performance Computing.– AMD.– 2023.– 4 c. URL ↑256, 257

[347] *HPC Comes to Life with AMD Instinct GPUs and NAMD*, Solution Brief, High Performance Computing.– AMD.– 2022.– 4 pp. URL ↑258

[348] Pall S., Alekseenko A. *GROMACS 2023: Readiness on the AMD GPU Heterogeneous Platform* // PDC Newsletters.– Jun 13 2023.– No. 1. URL ↑258

[349] *Molecular Dynamics. Nvidia GPU Benchmarks AMBER 22*, Blog.– Exxact.– 2023. URL ↑258, 259

[350] Zeng J., Zhang D., Lu D., Mo P., Li Z., Chen Y., Rynik M., Huang L., Li Z., Shi S., Wang Y.,; Ye H., Tuo P., Yang J., Ding Y., Li Y., Tisi D., Zeng Q., Bao H., Xia Y., Huang J., Muraoka K., Wang Y., Chang J., Yuan F., Bore S. L., Cai C., Lin Y., Wang B., Xu J., Zhu J.-X., Luo C., Zhang Y., Goodall R. E. A., Liang W., Singh A. K., Yao S., Zhang J., Wentzcovitch R., Han J., Liu J., Jia W., York D. M., E W., Car R., Zhang L., Wang H. *DeePMD-kit v2: A software package for deep potential models* // The Journal of chemical physics.– 2023.– Vol. **159**.– No. 5.– id. 054801. doi ↑259

[351] Prokopenko A., Sao P., Lebrun-Grandie D. *A single-tree algorithm to compute the Euclidean minimum spanning tree on GPUs* // *ICPP '22: Proceedings of the 51st International Conference on Parallel Processing* (29 August 2022–1 September 2022, Bordeaux, France), New York: ACM.– 2022.– ISBN 978-1-4503-9733-9.– id. 14.– 10 pp. doi ↑259

[352] Bagusetty A., Panyala A., Brown G., Kirk J. *Towards cross-platform portability of coupled-cluster methods with perturbative triples using SYCL* // *2022 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (13–18 November 2022, Dallas, TX, USA).– IEEE.– 2022.– ISBN 978-1-6654-6021-7.– Pp. 81–88. doi ↑260

[353] Bussy A., Schütt O., Hutter J. *Sparse tensor based nuclear gradients for periodic Hartree–Fock and low-scaling correlated wave function methods in the CP2K software package: A massively parallel and GPU accelerated implementation* // The Journal of Chemical Physics.– Apr 28 2023.– Vol. **158**.– No. 16.– id. 164109. doi ↑261

[354] Mazur L., Bollweg D., Clarke D. A., Altenkort L., Kaczmarek O., Larsen R., Hai-Shu T., Goswami J., Scior P., Sandmeyer H., Neumann M., Dick H., Ali S., Kim J., Schmidt C., Petreczky P., Mukherjee S. *SIMULATeQCD: A simple multi-GPU lattice code for QCD calculations* // Computer Physics Communications.– July 2024.– Vol. **300**.– id. 109164. doi ↑261

[355] Gottlieb S., Jeong H., Strelchenko A. *Two-link staggered quark smearing in QUDA*.– 2023.– 10 pp. arXiv 2301.05518 ↑262

[356] Mullowney P., Thomas S., Carr A. K., Swirydowicz K., Day M., Esclapez L. *Novel solver algorithms for nearly singular linear systems arising in combustion modelling*, NREL/PR-2C00-81907, 2022 SIAM Conference on Parallel Processing for Scientific Computing (February 23, 2022).– National Renewable Energy Lab. (NREL).– 2022. URL ↑262

[357] Halver R., Junghans C., Sutmann G. *Using heterogeneous GPU nodes with a Cabana-based implementation of MPCD* // Parallel Computing.– September 2023.– Vol. **117**.– id. 103033. doi ↑262

[358] Min M., Brazell M., Tomboulides A., Churchfield M., Fischer P., Sprague M. *Towards exascale for wind energy simulations*.– 2022.– 16 pp. arXiv 2210.00904 ↑265, 266

[359] Kolev T., Fischer P., Abdelfattah A., Beams N., Brown J., Jean-Camier S., Carson R., Chalmers N., Dobrev V., Dudouit Y., Ghaffari L., Joshi A. Y., Kerkemeier S., Lan Y.-H., Mc Dougall D., Medina D., Min M., Mishra A., Pazner W., Phillips M., Ratnayaka T., Shephard M. S., Siboni M. H., Smith C. W., Thompson J. L., Tomboulides A., Tomov S., Tomov V., Warburton T. *High-order algorithmic developments and optimizations for more robust exascale applications*, ECP Milestone Report. WBS 2.2.6.06, Milestone CEED-MS38.– 2022.– 76 pp. URL ↑265, 266, 267

[360] Lesur G. R. J., Baghdadi S., Wafflard-Ernandez G., Mauxion J., Robert C. M. T., Van den Bossche M. *IDEFIX: a versatile performance-portable Godunov code for astrophysical flows* // Astronomy and Asrtrophysics.– September 2023.– Vol. **677**.– id. A9.– 17 pp. doi ↑267, 268

[361] White C. J., Mullen P. D., Yan-Jiang F., Davis S. W., Stone J. M., Morozova V., Zhang L. // The Astrophysical Journal.– 2023.– Vol. **949**.– No. 2.– id. 103.– 29 pp. doi ↑268

[362] Grete P., Dolence J. C., Miller J. M., Brown J., Ryan B., Gaspar A., Glines F., Swaminarayan S., Lippuner J., Solomon C. J., Shipman G., Junghans C., Holladay D., Stone J. M., Roberts L. F. *Parthenon—a performance portable block-structured adaptive mesh refinement framework* // The International Journal of High Performance Computing Applications.– 2023.– Vol. **37**.– No. 5.– Pp. 465–486. doi ↑268

[363] Schild N., Räth M., Eibl S., Hallatschek K., Kormann K. *A performance portable implementation of the semi-Lagrangian algorithm in six dimensions* // Computer Physics Communications.– February 2024.– Vol. **295**.– id. 108973. doi ↑268

[364] Sfiligoi I., Belli E. A., Candy J., Budiardja R. D. *Optimization and Portability of a Fusion OpenACC-based Fortran HPC code from Nvidia to AMD GPUs* // *PEARC '23: Practice and Experience in Advanced Research Computing* (Portland, OR, USA, July 23–27, 2023), New York: ACM.– July 2023.– ISBN 978-1-4503-9985-2.– Pp. 246–250. doi ↑268, 269

[365] Diederichs S., Benedetti C., Huebl A., Lehe R., Myers A., Sinn A., J.-Vay L., Zhang W., Thévenet M. *HiPACE++: a portable, 3D quasi-static particle-in-cell code* // Computer Physics Communications.– September 2022.– Vol. **278**.– id. 108421. doi ↑269

[366] *Breaking Barriers in Plasma Physics with PIConGPU and AMD Instinc MI250 GPU*, Solution Brief, High Performance Computing.– AMD.– 2023.– 4 pp. URL ↑269

[367] Fedeli L., Huebl A., Boillod-Cerneux F., Clark T., Gott K., Hillairet C., Jaure S., Leblanc A., Lehe R., Myers A., Piechurski C., Sato M., Zaim N., Zhang W., Jean-Vay L., Vincenti H. *Pushing the Frontier in the design of laser-based electron accelerators with groundbreaking mesh-refined particle-in-cell simulations on exascale-class supercomputers // 2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)* (Dallas, Texas, USA, November 13–18, 2022).– IEEE.– 2022.– id. 3.– 12 pp. doi ↑269

[368] Huebl A., Lehe R., Zoni E., Shapoval O., Sandberg R. T., Garten M., Formenti A., Jambunathan R., Kumar P., Gott K., Myers A., Zhang W., Almgren A., Mitchell C. E., Qiang J., Grote D., Sinn A., Diederichs S., Thevenet M., Fedeli L., Clark T., Zaim N., Vincenti H., Jean-Vay L., *From compact plasma particle sources to advanced accelerators with modeling at exascale.*– 2023.– 4 pp. arXiv 2303.12873 ↑269

[369] Adams M. F., Wang P., Merson J., Huck K., Knepley M. G. *A performance portable, fully implicit Landau collision operator with batched linear solvers.*– 2024.– 20 pp. arXiv 2209.03228 ↑269

[370] Thawakar O., Anwer R. M., Laaksonen J., Reiner O., Shah M., Khan F. S. *3D mitochondria instance segmentation with spatio-temporal transformers.*– 2023.– 10 pp. arXiv 2303.12073 ↑270

[371] Samuel D., Kutuzov A., Touileb S., Velldal E., Øvrelid L., Rønningstad E., Sigdel E., Palatkina A. *NorBench–A benchmark for Norwegian language models.*– 2023.– 16 pp. arXiv 2305.03880 ↑270

[372] Yankovskaya L., Tars M., Tättar A., Fishel M. *Machine translation for low-resource Finno-Ugric languages // Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)* (Tórshavn, Faroe Islands, May 22–24, 2023).– University of Tartu Library.– 2023.– ISBN 978-99-1621-999-7.– Pp. 762–771. URL ↑270

[373] Charpentier L., Wold S., Samuel D., Rønningstad E. *BRENT: Bidirectional retrieval enhanced Norwegian transformer // Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)* (Tórshavn, Faroe Islands, May 22–24, 2023).– University of Tartu Library.– 2023.– ISBN 978-99-1621-999-7.– Pp. 202–214. URL ↑270

[374] Samuel D., Øvrelid L. *Tokenization with factorized subword encoding // Findings of the Association for Computational Linguistics: ACL 2023* (Toronto, Canada, July 9–14, 2023).– ACL.– 2023.– ISBN 9781959429623.– Pp. 14143–14161. doi ↑270

[375] *AMD Radeon Instinct MI300*, GPU Specs Database.– TechPowerUp. URL ↑271

[376] Naffziger S., Beck N., Burd T., Lepak K., Loh G. H., Subramony M., White S. *Pioneering chiplet technology and design for the AMD EPYC™ and Ryzen™ processor families: Industrial product // 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)* (Valencia, Spain, 14–18 June 2021).– IEEE.– 2021.– ISBN 978-1-4503-9086-6.– Pp. 57–70. doi ↑272

Приложение 1. Список сокращений, используемых в нескольких разделах обзора (для GPU разных производителей)

Сокращения для аппаратных средств

АЛУ Арифметико-Логическое Устройство *1, 5.1*

ЦОД Центр Обработки Данных . . . . . *Введение, 2, 3.1, 4.1, 4.2*

ЦП Центральный Процессор . . . . . *Введение, 1, 2, 3.1, 3.3, 4.1, 4.2, 5, 5.1, 5.2, 5.3, 5.4*

ASIC Application Specific Integrated Circuit *1*

CXL Compute Exress Link . . . . . *Введение, 2, 3.1*

ECC Error-Correcting Code *4.1, 4.2*

ECP Exascale Compute Project *3.1, 4.1, 5.3*

EPI European Processor Initiative . . . . . *Введение*

GPU Graphics Processing Unit . . . . . ***Введение**, 1, 2, 3, 3.1, 3.2, 3.3, 4, 4.1, 4.2, 4.3, 5, 5.1, 5.2, 5.3, 5.4*

GPGPU General-purpose GPU . . . . . ***Введение**, 3.1, 4.1, 4.2*

HBM High Bandwidth Memory *3.1, 4.1, 4.2, 5, 5.3*

ISA Instruction Set Architecture *1, 3.1, 4.1, 4.2, 5.1*

NUMA Non-Uniform Memory Access *2, 4.2, 5.1*

OoO Out-of-Order *4.2*

PIM Processing-In-Memory *2, 3.1, 4.1*

SIMT Single Instruction, Multiple Threads *1, 4.1*

SoC System-on-Chip *3.1, 4.2*

SVE2 Scalable Vector Extension 2 (ARM) *4.2*

TCO Total Cost of Ownership . . . . . *Введение, 5.3, 5.4*

TDP Thermal Design Power *2, 3.1, 3.3, 4.1, 4.2, 5, 5.1*

Сокращения для математической и программистской области

БПФ Быстрое Преобразование фурье *4.1, 5.3*

API Application Programming Interface *1, 4.1, 5.1, 5.2*

BLAS Basic Linear Algebra Subprograms *1, 4.1, 5.3*

HPC High Performance Calculations . . . . . *Введение, 1, 3.1, 3.2, 3.3, 4, 4.1, 4.2, 4.3, 5.1, 5.2, 5.3, 5.4*

HPCG High Performance Conjugate Gradients 5.3

HPL High Performance Linpack . . . . . Введение, 3.2, 5.3

LLVM Low Level Virtual Machine (раннее сокращение; теперь – знаменитый набор компиляторов и инструментариев) 4.1, 4.2, 5.2

PGAS Partitioned Global Address Space 4.1

SDK Software Development Kit . . . Введение, 1, 3, 4.1, 5.2, 5.3, 5.4

Сокращения для областей применения GPU и приложений

ИИ Искуственный Интеллект . . . . . Введение, 1, 2, 3.1, 3.2, 4, 4.1, 4.2, 4.3, 5.1, 5.2, 5.3, 5.4

CFD Computational fluid dynamics 3.2, 4.1, 4.2, 5.2, 5.3

DFT Density Functional Theory . . . . . Введение, 4.1, 5.4

DLRM Deep Learning Recommendation Models 4.1, 4.2, 5.3

BERT Bidirectional Encoder Representations from Transformers 2, 4.2, 5.3

NLP Natural Language Processing 2, 4.1, 4.2

Сокращения, общие для GPU разных фирм

SU Special Function Units 4.1

OAM OpenCompute Accelerator Module Введение, 2, 3.1, 5.1, 5.3, 5.4

Сокращения для GPU Nvidia

CUDA Compute Unified Device Architecture . . . . . Введение, 1, 2, 3.2, 3.3, 4.1, 4.2, 5, 5.1, 5.2, 5.3, 5.4

GPC GPU (или Graphics) Processing Clusters 2, 4.1, 4.2

GTC GPU Technology Conference . . . . . Введение

MIG Multi-Instance GPU 2, 4.1, 4.2

NCCL Nvidia Collective Communications Library . . 4.1, 4.2, 5.2

PTX Parallel Threads eXecution 4.1, 4.2, 5.3

SM Streaming Multiprocessor . 1, 2, 3.1, 4, 4.1, 4.2, 5, 5.1, 5.2, 5.3

SXM Server PCI Exress Module 2, 3.1, 3.3, 4.1, 4.2, 5, 5.1, 5.3, 5.4

TPC Texture Processing Clusters 4.1, 4.2

Сокращения для GPU *Intel*

AIC Add-in Card *3.1*

DPCT DPC++ Compatibility Tool *3.2, 5.2*

DPC++ Data Parallel C++ . . *Введение, 1, 3.2, 3.3, 4.1, 5.2, 5.3, 5.4*

PVC Ponte Vecchio *3, 3.1, 3.2, 3.3, 5.1, 5.4*

Сокращения для GPU *AMD*

CU Compute Unit *1, 2, 5, 5.1, 5.2, 5.3*

GCD Graphics Compute Die *3.1, 3.2, 5.1, 5.3, 5.4*

HIP Heterogeneous Computing Interface for Portability . . . . . *Введение, 1, 3.2, 3.3, 4.1, 5.1, 5.2, 5.3*

Сокращения для GPU *BR100*

CU *1, 2, 5, 5.1, 5.2, 5.3*

SVI Secure Virtual Instance *2*

Поступила в редакцию 16.10.2023;
одобрена после рецензирования 24.01.2024;
принята к публикации 01.03.2024;
опубликована онлайн 28.06.2024.

Рекомендовал к публикации *д.ф.-м.н. С. М. Абрамов*

Информация об авторе:

Михаил Борисович Кузьминский

старший научный сотрудник лаборатории компьютерного обеспечения химических исследований, кандидат химических наук ИОХ РАН. Научные интересы – высокопроизводительные вычисления, аппаратура ЭВМ, вычислительная химия.

0000-0002-3944-8203

e-mail: *kus@free.net*

Декларация об отсутствии личной заинтересованности: *благополучие автора не зависит от результатов исследования.*

EN RU
Review Article

PROGRAM SYSTEMS: THEORY AND APPLICATIONS
HARDWARE AND SOFTWARE FOR DISTRIBUTED AND SUPERCOMPUTER SYSTEMS

ISSN 2079-3316

UDC 004.272+004.382.2+004.8+004.43
10.25209/2079-3316-2024-15-2-139-473

New generation of GPGPU and related hardware: computing systems microarchitecture and performance from servers to supercomputers

Mikhail Borisovich **Kuzminsky**[1]✉

[1] Zelinsky Institute of Organic Chemistry of RAS, Moscow, Russia

Abstract. An overview of the current state of GPGPUs is given, with orientation towards their using to traditional HPC tasks (and less to AI). The basic GPGPUs in the review include Nvidia V100 and A100. Nvidia H100, AMD MI100 and MI200, Intel Ponte Vecchio (Data Center GPU Max), as well as BR100 from Biren Technology are considered as new generation GPGPUs. The important for HPC and AI tasks microarchitecture and hardware features of these GPGPUs, as well as the most important additional hardware for building computer systems with GPGPUs, that are CPUs specialized (albeit only possible for the initial period of their use) for working with the new generation of GPGPUs and interconnects — are analyzed and compared. Brief information is given about the servers (including multi-GPUs) using them, and new supercomputers (using these GPGPUs), where data on the achieved performance when working with GPGPUs was obtained.

The SDK of GPGPU manufacturers and software (including mathematical libraries) from other firms are briefly reviewed. Examples are given that demonstrate the tools of widely used programming models that are important for achieving maximum performance, while contributing to the non-portability of program codes to other GPGPU models.

Particular attention is paid to the possibilities of using tensor cores and their analogues in modern GPGPUs from other companies, including the possibility of using calculations with reduced (relative to the standard for HPC FP64 format) and mixed precision, which are relevant due to the sharp increase of the achieved performance when using them in GPGPU tensor cores. Data is analyzed on their "real-world" performance in benchmarks and applications for HPC and AI. The use of modern batch linear algebra libraries in GPGPU, including for HPC applications, is also briefly discussed.

(*Linked article texts in Russian and in English*).

Key words and phrases: GPGPU, V100, A100, H100, Grace, GH200 Grace Hopper, MI100, MI200, Ponte Vecchio, Data Center GPU Max, BR100, CUDA, HIP, DPC++, Fortran, performance, HPC, AI, deep learning

2020 *Mathematics Subject Classification*: 65Y05; 68M20

Acknowledgments:

For citation: Mikhail B. Kuzminsky. *New generation of GPGPU and related hardware: computing systems microarchitecture and performance from servers to supercomputers*. Program Systems: Theory and Applications, 2024, **15**:2(61), pp. 139–473. (*In Russian, in English*). https://psta.psiras.ru/read/psta2024_2_139-473.pdf

Introduction

A widespread feature of modern computing systems, from servers to supercomputers, is the use of a heterogeneous construction of servers and cluster nodes, very often containing not only processors (CPUs), but also accelerators, primarily GPUs, areas of use of which are rapidly expanding. Here we mean GPGPU, but in what follows the abbreviation GPU will be used.

Relevance of GPUs and areas of their using. Currently GPUs are actively used for a wide range of very important tasks, including high-performance computing (HPC) and AI (AI tasks in this review also include all tasks of any type of machine learning, but the use of GPUs is especially relevant for deep learning). In addition, the use of multiple GPUs in one server (multi-GPU) is expanding. All this is connected with the main direction of growth in the performance of computing systems, mainly due to the strong increase in the number of cores and, accordingly, parallelization on them. The increasing importance of power efficiency over time also contributes to the focus on the use of GPUs, which contain many more functionally simpler cores than in the CPU, but with a reduced frequency. Thus, of the 50 leading supercomputers on the Green500 list for June 2023, only four did not use a GPU. Moreover, two of them — the Japanese supercomputers NA-J2 and MN-3 — used specialized rarely used multi-core processors (coprocessors), and the other two were based on multi-core ARM processors Fujitsu A64FX [1]. Another important advantage of GPUs is the achievement of high-density packaging of large computing resources, which is especially evident in servers containing several GPUs (multi-GPU).

As for the often cited use of GPUs in data centers, the term data center has now practically replaced the previously used term computer center, which can also be considered [2] as one of the parts of the data center. In reality, data centers often assume the use of GPUs for the above HPC and AI tasks, and the term data center itself is focused primarily on the use of cloud technology. Further in the text, references to data centers refer specifically to cloud technology, the tasks of which are not discussed in the review — specific benchmarks and applications for HPC and AI are considered here. Although GPUs began to be noted as the main accelerator in other areas, for example, sorting when working with databases [3].

To illustrate the widespread use of modern GPUs, we can use the Top500 supercomputer list, as it provides interesting statistics (see, for example, [4]). In the Top500, the world's performance leaders have been using GPUs for a long time. A notable exception to this rule in recent years was the Japanese supercomputer Fugaku, whose nodes were homogeneous and contained only the CPUs — the A64FX. He topped this list for more than two years. Probably, such a success of Fugaku was facilitated by the rather large number of cores (48 computing) in the A64FX [5]. However, in 2022, a new Chinese supercomputer Sunway appeared (the successor to the Sunway TaihuLight, which ranks 7th in the Top500, and in [6] classified as "pre-exascale"), containing heterogeneous 260-core SW26010 processors in its nodes — without a GPU. As another not very widely used alternative GPU option, we can mention the huge, specialized for AI tasks, Cerebras WSE-2 processors, containing 850 thousand cores [7]. But WSE-2 does not support greater precision than FP32, and the Andromeda supercomputer based on them [8] is accordingly absent from the Top500 list.

Statistics from the June 2020 Top500 list [9] indicated the use of accelerators in 26.6% of Top500 supercomputers (20.2% of supercomputers used Nvidia V100). In the June 2023 list, accelerators were used in 32.4% of all supercomputers (13% used Nvidia V100, 15.6% used Nvidia A100, 2.2% used AMD MI250X and MI210, 2% used Nvidia H100) [4]. The unambiguous modern leadership in the Top500 GPUs from Nvidia is obvious today and predictable for the near future. All data presented later in this review refers to the June 2023 Top500 list, and by default the Top500 list below refers to this June list.

Features of the next expected GPUs. Integration of CPU and GPU in one die is now becoming possible. For personal computers with conventional graphics processors, similar integration with the CPU has long been known, for example, in the form of the AMD APU (Accelerated Processing Unit), but here we mean the integration of the server CPU with the GPU. AMD expects such integration into the APU in the MI300 [10]. Intel talked about its plan to combine x86 chiplets together with GPU chiplets called Falcon Shores back in 2022 [11, 12], but its implementation will take more than one year. In a certain sense, a similar development from Nvidia, Grace Hopper [13–15], appears on the market earlier. But this whole direction is a possible way to intensify the use of the GPU itself.

Limitations and difficulties of using GPU. It must be kept in mind that GPUs are installed in only 32.4% of all supercomputers from the Top500 [4] (in the June 2022 list it was 30.2%). Performing calculations exclusively on GPUs is associated with the use of high-speed memory, but having a fixed and not very large capacity compared to the possible memory size of the servers or certain input data of applications (research objects), this may cause inefficiency on the GPU altogether. Therefore, for example, in the manual for specialized parallelization tools on Nvidia GPUs, CUDA [16], there is a section dedicated to exceeding the required memory capacity of the memory available on the GPU. In modern versions of GPU CUDA also provides the ability to work with virtual memory [16]. However, it is clear that actually working with virtual memory can lead to severe performance losses.

GPU computing involves the use of applications that are highly parallelized across a large number of cores. A classic example of this is molecular dynamics tasks and, especially, AI area. But this may not hold true for certain applications or even HPC areas. Therefore, in the tuning guide for applications using CUDA (for the Nvidia Ampere architecture used in the A100) [17], the first point of recommendations is to find a way to parallelize sequential code, which may mean the need to create new, improved algorithms that allow parallelization where in the "natural" algorithm it might be missing. It may also not meet the expectations of specialists in the relevant HPC fields, who may often be programming applications for this field themselves.

Since effective use of GPUs requires a very high level of parallelization scalability, this also requires the use of SDKs specialized for GPUs, and possibly an increase in the size of source code. Many HPC applications did not natively respond to this level of parallelism. Additionally, optimizing to the high expected level of GPU performance often requires many manual work, which is especially difficult when porting code from one type of GPU to another. All this complicates the work of programmers and can cause them some rejection.

The situation is simplified to a certain extent in cases where there are small parts of the program that limit performance (in the GPU world they become program kernels), which are often typical mathematical problems. And over time, more and more HPC applications are becoming capable

of running on GPUs. For example, quantum chemical software systems that have been running on supercomputers for a long time are also moving in this direction. But for the most widespreads of the modern methods used there (without explicit non-empirical computations of electronic correlation), too long execution times can be associated with several mathematically different types of calculations (and not always related to those that are widespread in mathematical area; This is especially true for the use of gaussian basis functions), which requires a correspondingly much larger programming. For calculation by the widely used quantum chemical DFT method in a plane wave basis, a demonstration of the possible execution times of various parts of the program is given, for example, in [18].

Other important characteristics of using GPUs are cost indicators. If the goal is not to achieve an acceptable execution time at any cost (which is perhaps achievable only with the use of a GPU), then the relevant question becomes how much the cost of a computer increases when adding a GPU to it, and how much the application performance increases.

As an illustration, we indicate the acceleration data when calculating with the well-known Quantum Espresso software package, which is focused on calculations in the basis of plane waves using the quantum chemical DFT method. An illustration using the application of quantum chemistry, rather than the popular molecular dynamics on GPUs, was chosen here specifically — problems of quantum chemistry have long been performed on supercomputers, but the possibilities of quantum chemical calculations on GPUs began to appear later than in classical molecular dynamics — it is more difficult to implement, and the speedups achieved often smaller. Calculations by Quantum Espresso 6.5 were performed on a server with an 18-core Intel Xeon E5-2697 v4 (2.3 GHz), and adding V100 gave speedup in the range of 1.4-3.7 times [19]. The achieved acceleration naturally depends on the object being calculated. But we must keep in mind that this Xeon model began to be produced by Intel back in early 2016, and the calculation time was compared using only one processor.

As for the prices for new generation GPUs — this naturally applies to boards with a GPU (for example, an OAM module), they are not discussed in the review, since the corresponding "official" (for example, recommended by the manufacturer) prices for such new equipment are usually not available. But keep in mind that GPUs often provide greater power efficiency while requiring a relatively small square, so it's best to use total cost of ownership (TCO) rather than just price when evaluating GPUs.

Modern realities of growing GPU use. All of the above possible difficulties are gradually being resolved by creating new calculation methods and algorithms, new programming models for the SDK, as well as by improving GPU hardware. Naturally, this is reflected in the growing number of applications running on GPUs. The GPU application area is constantly growing, which, naturally, is most clearly demonstrate with Nvidia GPUs and was convincingly demonstrated at the latest GTC 34 (2022) and 35 (2023) conferences.

As time goes on, the number of supercomputers with GPUs included in the Top500 increases — with a GPU it is easier to obtain high performance in the HPL benchmark. The most powerful (as measured with the HPL) supercomputers in the world typically use GPUs. In the first twenty leaders of the Top500, only three supercomputers do not use GPUs (although the Chinese Tianhe-2A, which closes the top ten, also uses the Matrix-2000 accelerator, but this is not a GPU); The percentage of GPU utilization decreases further when considering a larger number of supercomputers.

But all this becomes weakly significant compared to the growing use of AI, which is covering more and more new areas of using — this primarily determines the requirements for GPUs (the HPC market is negligibly small compared to AI). Modern supercomputers included in the Top500 are also becoming AI-focused.

A review of the current global GPU market by renowned Chinese electronics industry analyst Chen Lizhong also suggests continued growth in the industry [**20**].

Relevance of the review, selection of GPUs under consideration and areas of their analysis. As general modern overview of different types of accelerators, including GPUs, can be considered [**21**] from the famous European BPG (Best Practice Guide) series. But today GPUs are characterized by ultra-fast development, and we can already talk about the emergence of a new generation of GPUs. In this review, GPU performance analysis focuses primarily on HPC tasks. There are publications that implement the fusion of traditional HPC fields with AI, for example, quantum molecular dynamics (QMD) and AI [**22**], or computational fluid dynamics and AI [**23**]. But currently combining traditional HPC tasks with AI methods may not be necessary (for example, for QMD — see [**24**]).

However, currently there is also an integration of HPC, AI tasks and processing of large volumes of data (as an example of work in the last 2 years, we can cite [25–30]), and benchmarks for this area have already appeared [31]. As an illustration of the active progress of work in this direction, we can note the consolidation of the well-known HPC developers of mvapich2 parallelization tools into new teams at Ohio University (US), where software tools running on top of mvapich2 are now being created — High-Performance Deep Learning (HiDL) [32] and High-Performance Big Data (HiBD) [33].

Given the potentially widespread use of AI in the commercial area, and the corresponding increased AI focus of modern GPUs [34], this review considers AI tasks for performance evaluations, although the paper is aimed primarily at traditional HPCs. The relevance of the analysis of modern GPUs is even increasing due to the emergence of the latest GPUs with higher performance (with their use, EFLOPS-level supercomputers are being created and are expected to be created) and the need for their optimal selection for the acquisition and use of appropriate hardware and software. To date, GPUs have come a very long way in the development of their architectures and performance indicators. Conventionally, the Nvidia V100 and A100 are classified as the modern "basic" generation in this review. This was chosen both because the V100 was the first to use tensor cores, and because of the breadth of use of these GPUs on modern supercomputers and servers. This GPUs review data are based on the V100 and A100 [21].

With the V100 and A100, this review will compare new generation GPUs — AMD Instinct (Radeon Instinct) MI100 and the MI200 family, Nvidia Hopper (H100), Intel Ponte Vecchio (Intel now produces a whole series of GPUs, Data Center GPU Max for which this is a codename), and partly the latest Chinese BR100 from Biren Technology. These GPUs are conventionally classified as a new generation, including because it was with their use that the exascale barrier was first overcome or it is planned to be further overcome (this applies to GPUs from AMD, Nvidia and Intel). The BR100 is included here due to its significantly higher specified performance indicators compared to the A100 [35]: at the processor level, Chinese developers have not previously outperformed processors from the US and Japan (for example, the ARM Kunpeng 920 [5] or the Zhaoxin x86 processor [36]), but the appearance in 2022 of the SW26010pro processors containing 390

cores with a total peak performance of more than 14 TFLOPS (by default in the text of this review, double precision, FP64 is assumed) [37] and the BR100 GPU gave such a high performance achievement that, one might say, for the first time allowed the Chinese industry to surpass the performance of some similar US products.

The relevance of a comparative analysis of AMD MI100 and MI200 with the above GPUs from Nvidia, Intel and Biren Technology seems obvious. The AMD MI250X began to be produced primarily for Frontier, which became the world's first exascale supercomputer [38–40], but is already used in about ten different supercomputers from the Top500, including the third-ranking supercomputer LUMI [41, 42], and GPUs actually determine the maximum performance, achieved there in the Top500. The well-known supercomputers Summit and Sierra with V100 nodes have been in the top ten Top500 for a number of years. Intel X^e-HPC Ponte Vecchio GPUs will be used in the US Argonne National Laboratory's Aurora supercomputer, where peak double precision performance is expected to exceed 2 EFLOPS [43], and in the SuperMUC-NG supercomputer upgrade at the Leibniz Supercomputing Center in Germany [44].

In addition, the most energy-efficient supercomputers are built on new-generation GPUs — for example, Henri with H100 heads the Green500, and supercomputers with MI250X occupy all the places there from 2 to 7 positions.

The relevance of comparing MI100 with Nvidia GPUs was recently noted in [45],and now GPU comparison has become even more important due to the emergence of new higher performance and more energy efficient GPUs. MI100 and MI200 are already actively used in HPC and AI. Much attention is paid to the performance data of the MI250X and A100 GPUs, obtained using the latest HPE/Cray EX supercomputer systems containing them [46].

The new generation of GPUs is distinguished not only by the construction of exascale supercomputers on them (Frontier — on MI250X [39], Aurora — on Ponte Vecchio [47], and the Selene supercomputer, which ranks 9th in the Top500, was supposed to be replaced with H100 [48]), but also by using other specialized hardware with them. First of all, these are interconnects (for example, Nvidia NVLinik [13, 49, 50], AMD Infinity Fabric, also characterized by regular improvements of versions [10], or the

CXL standard in BR100 [51]). These hardware closely related to GPU are discussed in this review. Some server processors — for example, ARM — Grace processors from Nvidia for work with GPU Hopper [13–15, 52] or AMD EPYC Zen 3 with (alleged) support of Infinity Fabric 3.0 in the I/O die [53]) were originally intended to work in conjunction with new GPUs, and are also discussed in the review.

But these CPUs could also be targeted at HPC or AI applications without using of GPUs. The Intel Xeon Max series [54] (codenamed Sapphire Rapids), which was originally intended to be used in the GPU-containing nodes of the Aurora supercomputer, can be used independently of the GPU.

For future EFLOPS-level supercomputers, the EPAC accelerators being developed within the European Processor Initiative (EPI) may be of interest, which are based on RISC-V with the ability to work with vectors of length 256 numbers in the FP64 format [55]– but these are accelerators that are not related to the GPUs, and EPAC is still at the development stage (only its test version 1.0 is available [56]), and a chiplet is being constructed from a number of tiles of various types, where EPAC is only one of them [57]. Accordingly, EPAC is not within the scope of this review.

The review consists of sections with subsections. Section 1 discusses the general hardware and software features of GPUs from different manufacturers. Section 2 analyzes the new Chinese GPU, Birentech BR100. Section 3 analyzes Intel Data Center GPU Max (Ponte Vecchio). Section 4 analyzes Nvidia GPUs: in Section 4.1 — A100, and in Section 4.2 — H100. Section 5 analyzes the AMD MI200 GPUs. In conclusion, general conclusions are drawn.

All sections review the hardware and software (SDK) for the respective GPUs and provide an overview of available performance data. In Section 3 and Section 5, and in Section 4.1 and Section 4.2, this is implemented as separate lower-level subsections. When comparing data, primarily on performance, a comparison was also used with Nvidia V100 performance data, and in Section 5 there is a separate Section 5.3.1 with AMD MI100 performance data.

The review necessarily uses a very large number of abbreviations. The author often provides explanations of well-known abbreviations, keeping in mind the possible reading of the text by specialists from different fields. A list of abbreviations used in several different sections of the review (in sections about different GPUs) is given in the appendix.

1. Common features for GPUs from different manufacturers

Before considering specific GPUs from different manufacturers, it is necessary to at least list the main software development tools (programming models) used on modern GPUs with a focus on HPC and AI tasks. Maximum performance is usually achieved using, of course, SDKs that are clearly focused on hardware manufacturers: for Nvidia — CUDA (Compute Unified Device Architecture) [16], for AMD — HIP (Heterogeneous Computing Interface for Portability) [58], part of the overall ROCm software stack (lower-level software tools are not discussed in this section of the review). The noticeable appearance of alternative GPU manufacturers to Nvidia on the market has increased interest in SDK components that work with various types of accelerators. HIP already has the ability to work with Nvidia GPU [58].

Among the programming tools that are not oriented towards working with the GPU of a certain manufacturer, we first note OpenACC and modern versions of OpenMP (support for working with accelerators appeared in OpenMP version 4.0, and since 2021 there is already a 5.2 specification [59]). Later, OpenCL (Open Computing Language) [60], and then SYCL [61] — an open standard for heterogeneous programming, became more widely used as tools for developing programs for GPUs and PGA accelerators. SYCL is developed by the Khronos Group, and (beginning from SYCL 2020) is based on C++17.

Data Parallel C++, developed by Intel (DPC++) [62] is also an open cross-architecture language built on C++ and SYCL — may become widespread. DPC++ uses SYCL with extensions that are expected to be included in future versions of the SYCL standard. OpenCL, SYCL and DPC++ can also be used for CPUs. Of these, DPC++ now appears to be the most advanced; A benchmarks already appeared on its basis [63].

Finally, GPU software mentioned here also includes Kokkos [64, 65]. Kokkos (supported in a US Department of Energy project) targets exascale supercomputers, uses C++, and aims to be «hardware-neutral». It can use, in particular, CUDA, HIP, SYCL and OpenMP as a back-end. A famous example of an application using Kokkos is the LAMMPS package of programs for molecular dynamics [66].

The listed software tools reflect the growing use of C/C++ in the areas of HPC and AI. But the question of the achieved performance compared, for example, with CUDA programs on Nvidia GPUs requires further study.

The functional simplicity of GPU cores makes it possible to quickly switch thread context from active to passive and back, which is not true for the CPU.

Nvidia's long-term dominance of the GPU market has led to the widespread use of terms for GPUs proposed by Nvidia. But the emergence of a new generation of GPUs, including from other companies, was characterized by their use of other terms for the same things (most of them are SIMT terms for APIs — CUDA, HIP, OpenCL and others). Accordingly, there are many publications and conference reports that provide correspondences between terms from different manufacturers, including in tabular form (see, for example, [**67**, **68**]). Below in Table 1 such a comparison is made for the purposes of this review.

All rows of the table, except the last two, are API terms. The last two lines contain terms for similar important hardware components of GPUs from different manufacturers. This table does not include the terminology used for the BR100.

The table in the right column shows in bold the terms that will be used later in this review as common for GPUs from different manufacturers (although in sections about a specific manufacturer its terminology is also used).

The used by GPU manufacturers terms may vary depending on their using for hardware or software, and may change as new models become available. Thus, AMD uses the term wavefront in the architecture and ISA manuals discussed in the review of this company's GPUs — but in the modern HIP manual only warp is used [**58**]. And the emergence of a new generation of Nvidia GPUs caused the emergence of a new term for them — a cluster of thread blocks for the H100 GPU [**16**] in the hierarchy of various levels of thread groups.

Table 1. A comparison of terms used by various GPU manufacturers, including their programming models

Nvidia (CUDA)	AMD (HIP)	Intel (oneAPI/ SYCL)	Desrription; **the general term used in the review** (if used as a general term)
Thread	Work item; Thread	Work-item	Individual thread (they work together in a group of threads — in a warp or in a sub-group); **thread**
Warp	Wavefront; (sometimes Warp)	Sub-group	A set of operations (threads) that execute synchronously, execute the same instructions, and follow the same control flow path: a group of parallel threads executed by a hardware unit (Nvidia SM has 32 of them) is the smallest computing unit, a block of threads per they are divided; **warp**
Thread block	Workgroup	Work-group	A group of warps/sub-groups running concurrently on a GPU (running on a single SM on an Nvidia GPU). Can synchronize together and communicate using shared memory; **thread block**
Grid	Grid	ND-range	A grid of blocks of threads, the top level of the hierarchy of the thread system of the entire GPU; **thread grid**
Streaming Multiprocessor (SM)	Compute Unit (CU)	X^e core	An analogue of a functionally simplified CPU core (contain parallel ALUs). For example, in Intel Data Center GPU Max X^e core contains several SIMD-type ALUs.
Tensor core	Matrix core unit	Matrix Engine (XMX)	Computational unit for multiplying small matrices (similar to GEMM, with mixed precision); **tensor core**
Shared memory	Shared memory	Local memory[1]	High-speed (cache-like) low-capacity memory shared by all threads in a thread block/work-group; **shared memory**
Global Memory[2]			DRAM memory available in the GPU; its data passes through several levels of cache memory
Device[2]			GPU (with the memory); **device**
Host[2]			Processors and memory (more generally — the entire part of the computer without the GPU); **host**
Kernel[2]			Part of a program executed on the GPU (function in C, subroutine in Fortran). Kernel can run in parallel with the CPU; **kernel**

[1] Incomplete compliance;
[2] a common term for all GPU developers.

Nvidia terms are taken from [16]; AMD — from [58]; Intel — from [69].

Different levels of thread groups allow you to effectively organize highly scalable SIMT parallelization. Since a situation may arise where a warp is waiting for data from memory, the active calculation then switches to another warp. In classic Nvidia GPUs, for this purpose, SM contains a warp scheduler (it forms a warp group of threads) and a dispatch unit for activating warp execution. Similar hardware units exist in GPUs from other manufacturers.

Another very important common feature of modern GPUs is the ability to work with data of varying precision, including mixed precision operations. This, when using reduced precision and, accordingly, the number of bits to represent a number, makes it possible to achieve several times higher peak performance, reduce the requirements for GPU memory size and its bandwidth, which very often limits performance. When reducing the required memory capacity, the reduced amount of GPU communication with the CPU can also improve performance. This doesn't make sense when working with traditional CPUs, and all floating point calculations in HPC are done traditionally with FP64. However, for very actively developing AI areas, working with neural networks uses matrix multiplication, and it has been found possible to work with lower precision and with mixed precision.

The typical data format for use in deep learning is single precision, FP32, but many works have shown that lower precision, such as FP16, is sufficient [**70**]. The calculation time for deep learning is limited usually by matrix multiplications, which is what tensor cores in Nvidia GPUs or their analogues in other new generation GPUs are focused on. The tensor core in a GPU first appeared in the V100, and from the very beginning it was considered as an application specific integrated circuit (ASIC) integrated into the GPU — see, for example, [**71**]).

Formula (1) reflects the BLAS function **GEMM** (here **A**, **B**, **C** are two-dimensional matrices, the dimension of **A** is $M \times K$, the dimension of **B** is $K \times N$, the dimension of matrix **C** is $M \times N$).

$$C = \alpha A \times B + \beta C \tag{1}$$

Already in the first tensor cores (in V100), the FP32 format was used for **C**, and FP16 — for **A** and **B** [**72**]. In the A100 you can use BF16 for **A** and **B**, and TF32 for **C** (although in the A100 it is now possible to work with the FP64 format in tensor cores) [**73**].

Table 2. Reduced precision floating point formats on GPUs

Number format	Number of bits			
	Number's sign	Exponent	Mantissa	In register[1]
FP32	1	8	23	32
TF32	1	8	10	32
TF32+	1	8	15	32
FP16	1	5	10	16
BF16	1	8	7	16
FP8-E4M3	1	4	3	8
FP8-E5M2	1	5	2	8

[1] TF32+ format is only supported by BR100 [35], and FP8 formats are supported by H100 [78].

This table uses data from Table 11 in [79] with the addition of a TF32+ format line for BR100 [35].

Similar mixed-precision matrix operations are performed in modern GPUs on special matrix blocks (see terminologies from different manufacturers in Table 1) and are carried out for matrices of very small sizes from a fixed set. For example, in tensor cores A100 for all matrices from formula (1) with FP64 format $M \times N \times K = 8 \times 4 \times 8$ [17]. Many reduced-precision floating-point number formats have begun to be used on GPUs (primarily for AI tasks); The basic parameters of formats with reduced (relative to FP64) precision are shown in Table 2 (this table shows only formats for floating point numbers — but in AI it is also possible to work with integers reduced to 8 bits in length, INT8).

Some of these reduced precision formats are not supported by the IEEE-754 standard [74], but are supported by specific GPU manufacturer models (TF32, TF32+, BF16, and FP8 formats). It should be noted here that the TF32 and BF16 formats are considered effective for deep learning (see, for example, [17, 75]). TF32 uses the same 10 bits for the mantissa as FP16, but due to the longer exponent, the range of numbers represented is larger, which is important for AI tasks [76]. And in [77] the possibility of using FP8 formats for deep learning is considered.

Since using reduced-precision formats on the GPU can lead to very important performance increase, little by little the ability to work with reduced (relative to FP64) precision has begun not only to be used in AI, but to be studied in other well-known HPC areas, including: FP32 in CFD

(there were also attempts to work with FP16) [80], FP32 in classical molecular dynamics (in [81] the performance increase on FP32 was measured not on the GPU), FP32 in quantum molecular dynamics (there were also attempts to work with FP16) [82, 83], in quantum chemistry [84, 85]. The corresponding increase in performance may be due not only to a direct increase in the actual performance of the GPU cores due to a decrease in precision, but also to a possible dramatic reduction in the requirements for GPU memory capacity.

Naturally, studies began to appear on achieving acceptable precision of results when working with reduced precision in mathematical methods, for example, when solving the Poisson equation [86]. Methods for correcting possible errors relative to FP32 during calculations with FP16 and TF32 (when working on A100 tensor cores) are proposed in in [87]. It is clear that when working with reduced precision, fairly detailed systematic studies are required, which may not have time to be carried out due to the ultra-fast development of modern GPUs and the creation of new data formats in them.

Even in AI, the use of, for example, TF32 with a mantissa reduced relative to FP32 makes detailed studies relevant due to possible problems with the convergence of deep learning. Therefore, the TF32+ format available for BR100, which has a larger number of bits for the mantissa than in TF32, may be interesting. And, for example, in molecular dynamics, modern software packages running on the GPU often have options that allow calculations with reduced precision (for example, for two-point atom-atom interactions) and in a large number of cases give acceptable results — however, sometimes this leads to error. The author is not aware of publications that formulate in which cases such errors occur. For quantum chemistry the situation may be more complicated, for example, in iterations with self-consistent total energy. The author is currently generally wary of HPC calculations with reduced precision.

But until now, tensor cores are considered as ASICs focused on machine learning tasks (see, for example, [88]), although recently there have been works aimed at expanding the application of matrix multiplication with tensor cores to HPC (see, for example, [89]). But in general, for HPC it's necessary to be based on the performance achieved by specific applications. And in [90] it was found that of the 77 known HPC benchmarks selected

there, only 10 used GEMM. And from the point of view of power efficiency, the use of emulation of FP64 and FP32 formats with reduced precision on the V100 tensor cores is significantly worse than the use of vector cores there. Therefore, for wider use of tensor cores on HPC, from the author's point of view, they need hardware support for FP64, which Nvidia began with the A100.

In general, issues of working with numbers of different precision are of more general importance, not only for GPUs. For example, in [**91**] proposed a new combined multiply-and-add unit targeting HPC and AI tasks to handle formats having different precision. Some information about the achievable precision when running on tensor cores for matrix multiplication using mixed-precision operations is given very briefly later in Section 4.1.4, which discusses the achievable performance on the A100.

2. New Chinese GPU BR100

This review starts with the Biron Technology BR100 for a number of reasons. This accelerator differs quite significantly in design from more traditional Nvidia and AMD GPUs, even in the form of terminology used. There are almost no publications assessing the performance of the BR100 in benchmarks and applications, and the prospects for their further production have become doubtful due to US sanctions. Therefore, the terminology used for BR100 was not shown in Table 1. Nevertheless, the analysis of BR100 seems interesting, including as a possible effective alternative to modern Nvidia GPUs.

The appearance of these GPUs was clearly evident for two reasons — the high speed of development (they were made "almost from scratch" in just 3 years) and the declared superiority in performance of this Chinese GPU over the Nvidia A100 (the H100 simply did not exist then).

It should immediately be noted that the BR100 is very clearly focused on working in the field of AI, which allowed the developers to make a clear gradation of importance when designing the microarchitecture. The main available source of information on the BR100 (also used in this review) is a report at the 2022 Hot Chips 34 conference [**35**], and additional information is available on the developer's website [**51**]. There were also minor clarifications in the interview with Biren Technology director Zhang Wen [**92**]. Certain comparisons of the characteristics of the BR100 with other GPUs are then carried out only in relation to the A100, since this model is classified as a base model in the review.

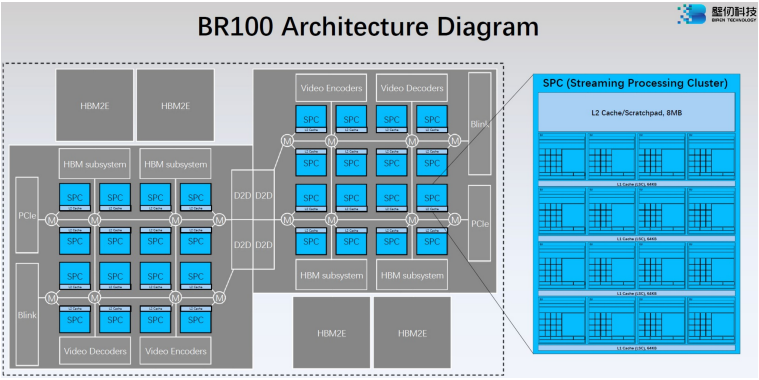


FIGURE 1. BR100 microarchitecture (figure from [35])

The overall microarchitecture of the BR100 is shown in Figure 1 [35].

The BR100 family contains two different models — the BR100 and the simpler, cheaper BR104, so BirenTech now also uses the BR10X name [51]. General characteristics demonstrating the success of the BR100 family are usually given for the BR100 model (see also Table 3). It should be noted here that the BR100 uses chiplets, is based on the use of two tiles (see Figure 1), and is manufactured using TSMC 7 nm technology (CoWoS 2.5D [35]). BR100 contains 77 billion transistors with a total area of 1074 mm^2. The BR104 has one tile, and many indicators are also half that of the BR100 (see Table 3).

The main component shown in Figure 1 that determines the achieved performance of the BR100 is the SPC (Streaming Processing Cluster), which can be partly considered a kind of analogue of the Nvidia GPC (Graphics Processing Cluster) in the A100. Each of the two BR100 tiles has 16 SPCs.

Each SPC contains 16 EU (Execution Unit) blocks, which contain the actual computing components of the GPU — 16 vector cores (V-cores), and one tensor core TDA (Tensor Data Accelerator) [35]. With a target clock frequency of 1 GHz (it is noticeably lower than the accelerated core frequency in the A100, see Table 13 below) and knowing the number of FP32 results achieved per clock cycle in the V-core (FP64 is not supported in the BR100, which is due to the focus on AI), this makes it possible to calculate peak performance with FP32. Table 3 provides data on the peak performance achieved when working with TDA (since they are especially relevant for AI tasks, which the BR100 is primarily focused on) [51]. The peak performance values achieved (for AI-relevant data

Table 3. Comparison of BR100 and A100 specifications

GPUs	BR100[1] (Walli 100P)	BR104[2] (Walli 104P)	A100-PCIe[5]	A100-SXM4[5]
Technology, nm	7 (TSMC)			
Form factor	OAM	Full-length two slot PCIe board	PCIe	SXM
Performance (peak):[4]				
FP32 (TFLOPS)	240	112		
TF32+ (TFLOPS)	480	224	156/312[3,6]	156/312[3,6]
BF16 (TFLOPS)	960	448	312/624[3]	312/624[3]
INT8 (TOPS)	1920	896	624/1248[3]	624/1248[3]
Memory type and capacity	HBM2E 64 GB	HBM2E 32 GB	HBM2E 40 GB	HBM2E 80 GB
Memory bus width (bits)	4096	2048	5120	5120
Peak Bandwidth (TB/s)	1.64	0.819	1.9	2.0
Interconnect to GPU, Peak Bandwidth (GB/s)	BLink (8 ports ×8), 448	BLink (3 ports ×8), 192	NVLink3, 600	NVLink3, 600
Interconnect to CPU	PCIe-5.0, ×16 with CXL support	PCIe-5.0, ×16 with CXL support	PCIe-v4 ×16	NVLink3
TDP, W	550	300	250	400

[1] see [99];
[2] see [100];
[3] after the slash data is given when using sparsity;
[4] data using tensor cores are presented;
[5] data from [93, 94];
[6] for A100 data is given for TF32.

formats) are only slightly below initial expectations [35] and 1.5–2 times higher than in A100.

In fact, in the hierarchy from the SPC to EU level in BR100 there is an intermediate level — CU (Compute Unit) blocks, each of which can contain 4, 8 or 16 EU blocks (see the right side of Figure 1) [35]. CU can be considered an analogue of SM (Streaming Multiprocessor) in A100.

A CU with four EUs has a 64 KB L1 cache (LSC). Next in the memory hierarchy is the L2 cache with a capacity of 8 MB per SPC, which gives 256 MB for the entire BR100. HBM2E memory with a capacity of 64 GB (in modern A100 models the capacity is increased to 80 GB — see, for example, [93]) has an interface width of 4096 bits with a bandwidth that is also lower than that of the A100 with 80 GB [93] (see Table 3).

To achieve high GPU performance, memory bandwidth is important, and some lag here between the BR100 and the A100 is compensated by the huge capacity of the L2 cache (the A100 has a much smaller L2 cache capacity — 40 MB [94, 95]). To maintain more efficient operation of the L2 cache in the BR100, use Near Memory Computing [92]. Obviously, this is a certain analogue of the Near Memory Processing paradigm, close to the PIM, processing-in-memory paradigm, the goal of which is to spatially combine computing units with memory and greatly reduce data transfers between them [96], applicable for AI tasks [97, 98].

Turning to the bandwidth, it is equally important for communication between two BR100 tiles, and is 896 GB/s [35], which allows the BR100 to be treated as one common GPU.

For communication with the CPU, PCIe-5.0 (×16) is used, with support for CXL (Compute Express Link) — an interconnect with support for cache coherence [101, 102], which has a clear tendency towards standardization.

Up to 8 BR100 GPUs can be installed in one server, and for communication between such GPUs, point-to-point communication channels (i.e., between each pair of GPUs) BLink are used, which uses SerDes (serializer/deserializer) [92]. One BLink has a bidirectional bandwidth of 64 GB/s [35], respectively, for 7 BLink channels connecting one GPU to all others, the total bandwidth is 448 GB/s. Choosing to fully support all point-to-point connections gives the BR100 an advantage due to the absence of potential contention when multiple GPUs share interconnect bandwidth, while GPU-to-GPU communication via the CPU has its downsides in bandwdith and latency [3]. Therefore, as noted in [3], the topology of such interconnect is very important.

For communication between the A100 GPU with the SXM4 form factor [103] the Nvidia NVLink3 interconnect [94] is used, where 12 channels with a bandwidth of 50 GB/s are used to connect the GPU-GPU — accordingly, a bidirectional bandwidth of 600 GB/s is obtained [94], which is much more than for BR100. The BR100's communication with the CPU has significantly higher bandwidth (896 GB/s) than the NVLink3's 600 GB/s. And in the A100 with PCIe-4.0 model, the communication bandwidth between GPUs is the same as that of the BR100, 64 GB/s [93].

It is clear that the effectiveness of a GPU interconnect can only be assessed by the measured performance of benchmarks or applications, which is practically non-existent for the BR100 at the moment. The observed orientation of applications to minimize all communications between the CPU and GPU (this is one of the basic rules of optimization in CUDA on the A100 [104]) suggests that scaling performance with increasing number of GPUs in a server with A100/SXM4 (in the case of rather large requirements for such communications) will be higher than in a server with BR100.

The advantage of BR100 is the use of OAM Spec v1.1 [92] — a rapidly spreading and actually claiming to standardize the OAM (OCP Accelerator Module) form factor [105] (the corresponding module with BR100 is called Walli100 [92]); And the BR100 is located on the UBB board [92], which can also become the standard of the future [105]. For comparison, the A100 with a high-speed NVLink3 GPU interconnect uses Nvidia's own SXM form factor, while the maximum number of GPUs in a server (for example, in the famous Nvidia DGX for AI) is also 8 [106].

Of course, BR100 provides a number of other features not discussed here — including 4 blocks of special functions (Special Function Units — SFU, analogues previously known in Nvidia GPUs, used, among other things, to calculate elementary functions); ability to work with NUMA and UMA; support for up to 8 virtual GPUs (Secure Virtual Instance, SVI — analogous to Nvidia MIG, Multi-Instance GPU), which allows several applications (that do not support good scaling to a full GPU) to effectively work with the one GPU simultaneously [35, 99]. Classic shader blocks for image processing are completely absent in the BR100, which is due to the unique orientation of this GPU towards AI (the video codec is supported [99], but this is not discussed in the review).

Another important modern GPU parameter is TDP — according to Table 3, BR100 needs more power consumption than A100. If we calculate power efficiency (performance per watt), then, for example, for the BF16 format relevant for AI in the BR100 it is higher than that of the A100 with SXM4 or PCIe (without using sparsity). The created Haixuan OAM server [92, 99] contains 8 BR100s and has a peak performance (for BF16) of about 8 PFLOPS with a maximum TDP of 7 kW [92] (see Figure 2) [35]. Together with Inspur, a well-known manufacturer of multi-GPU servers for AI, cluster solutions are also planned [92].

The BR104 not only has half the performance and memory capacity (see Table 3), but also the number of Blink ports is 3 — accordingly, fewer GPUs can be installed in one server. The announcement of the Wallen Technology Wallace 104 server with BR104 was reported in a number of media. The form factor is also important here — the BR104 uses a full-size two-slot board [100]. This can be compared with the A100-PCIe — based on them, Nvidia produces, for example, HGX modules (with a PCIe form factor) for servers, including two connected via the NVLink Bridge GPUs [93].

Regarding the BR100 (Bi Liren) architecture in a general sense, and not about the microarchitecture, it should be noted that there is a large set of supported data formats (Table 3 shows the performance for only some

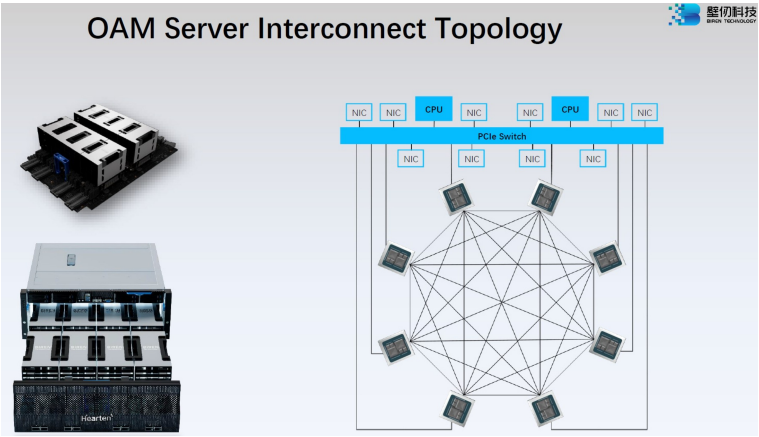


FIGURE 2. Interconnect topology in a server with BR100 (Figure from [35])

of them): INT8, INT16, INT32, FP16, BF16, FP32, TF32+. Information about them, primarily about the original development of Biren Technology TF32+, is available on a number of sites (see, for example, [107]). TF32+ looks like an attempt to improve the well-known TF32 format for tensor cores from Nvidia [94] (see Table 3), with the help of which the BR100 developers wanted to increase precision and performance [51]).

To work with BR100, a set of software tools BIRENSUPA (BIREN Scalable Unified Parallel Architecture) [92] — was developed — drivers, the BRCC compiler with support for extended C++, program libraries and other tools focused primarily on deep learning [35, 108]. BIRENSUPA has a programming paradigm and language style similar to NVidia `CUDA`, and also uses hardware capabilities unique to the BR100 [92]. But there were no publications demonstrating the use of these software tools and the actual performance achieved at the time of writing the review.

As for performance for AI, for BR104 there is data for two tests from the well-known set of benchmarks for the machine learning inference stage, *MLPerf inference datacenter*[URL] version 2.1 [109, 110] for data centers. The MLPerf inference datacenter benchmarks results show how quickly a trained neural network can perform inference tasks on new input data.

The first benchmark for image classification from the MLPerf Inference datacenter is based on ResNet (residual neural network) using the famous artificial neural network technology; ResNet has been expanded and modernized many times, and has been used, for example, for image processing for the diagnosis of COVID-19 (see, for example, [111]). Another

Table 4. MLperf 2.1 inference datacenter benchmarks data (December 2022) on servers with GPUs

GPU models	Number of GPUs in the server	Server model and manufacturer	Image classification (A=99%)		Natural Language Processing (A=99.9%)	
			server (queries/s)	offline (samples/s)	server (queries/s)	offline (samples/s)
BR104, PCIe	4	Inspur NF5468M66[1]	150027	212391	8993	11106
	8	Inspur NF5468M66-P[2]	200052	424660	13952	22134
A100, SXM, 80GB	4	Lenovo SR670v2[3]	150027	174180	No data	
	4	Dell PowerEdge XE8545[4]	128029	131364	5297	5476
A100, PCIe, 80GB	8	ASUS ESC8000A-E11[5]	270066	283838	11496	13129
A100, SXM, 80GB	8	Inspur N5688M6[6]	313069	347202	No data	
	8	Inspur N5488A5[7]	290066	346954	13594	14977
H100, SXM, 80GB	1	Nvidia Preview[8]	58995	81292	6195	7921

[1] with Intel Xeon Gold 6354 and suInfer;
[2] with Intel Ice Lake-SP 8368 and suInfer;
[3] with Xeon Platinum 8360Y 2.40 GHz and with CUDA 11.6;
[4] with EPYC 7763 and with MaxQ, TensorRT 8.4.2 and CUDA 11.6;
[5] with 64-kernel EPYC 7763, TensorRT 8.4.0 and CUDA 11.6;
[6] with Xeon Platinum 8358, TensorRT 8.4.2 and CUDA 11.7;
[7] with EPYC 7713, TensorRT 8.4.2 and CUDA 11.7;
[8] with 8-kernel EPYC 7252, with TensorRT 8.5.0 and CUDA 11.8.

benchmark for work with natural language, BERT (Bidirectional Encoder Representations from Transformers), uses a transformer-based machine learning model to pre-train natural language processing using a bidirectional encoder [**112**]. This method is extremely widely used and is probably most famous for its use by Google LLC.

The results of these famous tests, presented in [**109**] for servers with 4 or 8 BR104 GPUs, as well as for servers with 4 or 8 A100 GPUs, are shown in Table 4, which selects the highest performance results achieved. The specified data for GPU BR104 and A100 refers to the class of available (that is, the corresponding servers can be purchased).

In these tests, the required accuracy of inference (A) is 99% or 99.9% relative to FP32. But we must keep in mind that the achieved performance may significantly depend on the software development systems used. For

the A100, in addition to basic CUDA tools, to achieve high performance, To achieve high performance, special Nvidia TensorRT software [**113**] was used; For BR104, suInfer tools were used.

Performance data from MLperf inference datacenter 2.1 using 4 and 8 BR104 GPUs in Inspur servers showed the performance advantage of BR104 in the natural language processing (NLP) test with the BERT model by one and a half to two times when using the same number of BR104 or A100 in the server. In the "offline" scenario of the image classification test with the ResNet model on servers with 4 and 8 GPUs, servers with BR104 are also faster than servers with A100 (see Table 4), and in the "server" scenario of this test on servers with 4 GPUs, A100 is not ahead of servers with BR104, but significantly faster than them when using 8 GPUs. This may also be due to the advantage of NVLink3 over BLink with such a number of GPUs.

These tests may require less computational resources than MLPerf Training, often run on the A100 GPU: MLPerf Training measures the time required to train machine learning models to a target level of accuracy, where the main tasks are the actual training.

The above data about the BR100 clearly indicates that it is designed to compete with Nvidia GPUs, which should be supported not only by the higher peak performance of the BR100 and the high performance achieved in AI tests. A clear focus on extremely rapidly developing and commercially relevant AI tasks (which made it possible to target the BR100 hardware more narrowly and economically), and the use of hardware that claims to be standardized should help reduce the cost of the BR100, which is combined with the initially rather typical for Chinese manufacturers lower cost relative to products Western countries.

However, the situation with the BR100 changed dramatically due to US sanctions imposed in 2022, as a result of which TSMC stopped manufacturing and supplying BR100 chips. This has been widely discussed in various media, but here it should only be noted that this ban is aimed against possible competition with Nvidia and does not contribute to the acceleration of the development of the global GPU market.

3. Intel Data Center GPU Max (Ponte Vecchio)

The choice of Intel Ponte Vecchio (hereinafter abbreviated PVC) as a new generation of GPUs as the next object of analysis is due to the fact that at the time of writing the review they had just appeared on the market in the form of several different Data Center GPU Max models, and scientific publications about their performance are almost are missing.

Intel's information about available PVC models was changing at the time of writing this review. And the PVC architecture (and the software SDKs used) are quite different from the more "traditional" GPU architectures from Nvidia and AMD.

The appearance of PVC has been expected for several years; in September 2022, Intel announced the start of supply of PVC to the Aurora supercomputer at the Argonne National Laboratory in the US. But this supercomputer is missing from the June 2023 Top500 list.

3.1. PVC hardware

Intel developed the X^e architecture ("eXascale for everyone") for use in a wide variety of classes of GPUs (not just GPGPU), to work with both PCs and servers. X^e has a common instruction set architecture (ISA), and Intel uses 4 different microarchitectures — each class of GPU has its own microarchitecture [**69**]; PVC uses the X^e HPC GPUs [**114**, **115**]. For data centers, Intel offers Data Center GPU products, including two series — Data Center GPU Max [**116**], — this is the official name that replaces the use of the Ponte Vechio code word (abbreviation PVC in this review), and the Data Center GPU Flex series [**117**] with X^e-HPG microarchitecture.

The review considers only the family of HPC-oriented GPUs with the X^e HPC-GPU microarchitecture — Data Center GPU Max (PVC is designed to work in exascale supercomputers; PVC further means the senior model of this series, Max 1550 — for the formation of the Aurora supercomputer, Intel supplied this GPUs, and data about it have been presented in a number of publications cited here in the review). Intel points to 3 different models in this family, the recommended price data for which was expectedly not provided on the website ark.intel.com at the time of writing the review — in accordance with the corresponding lack of similar price recommendations from other manufacturers of new generation GPUs. Table 5 shows the basic specifications of various PVC models. This table shows only GPU specifications that are primarily relevant for classic HPC tasks. And, for example, the number of blocks in PVC for processing ray tracing, which can also be used for AI tasks (see, for example, [**118**]), is not given here (there are as many of them in PVC as X^e cores — 128) — because hardware capabilities of PVC for ray tracing are also not discussed in the review.

For the production of PVC, it was planned from the very beginning to use three-dimensional laying based on tiles [**114**, **120**]. Intel could have been pushed to this point by a certain lag in its own semiconductor technology (sometimes formulated as rumors [**121**]). According to a report from the famous exascale computing project ECP [**67**], PVC was planned for delivery in 2021.

TABLE 5. Basic characteristics of Data center GPU Max series models (PVC)

GPU model	Number of			Frequency (GHz)		Memory		TDP, Watt
	X^e cores	XMX	XVE	Base	Max.	capacity, GB	bandwidth, GB/s	
1100	56	448	448	1.0	1.55	48	1228.8	300
1350	112	896	896	0.75	1.55	96	2457.6	450
1450	128	1024	1024	no data		128	no data	600
1550	128	1024	1024	0.9	1.6	128	3276.8	600

XMX(X^e Matrix eXtensions) — matrix units, XVE (X^e Vector Engines)— vector units. The data in the table is taken from different (in time) versions [116] and [119]. Models 1350 and 1450 are marked in red because they are not listed on ark.intel.com at the time of review writing.

PVC uses three-dimensional Co-EMIB (Co-Embedded Multi-Die Interconnect Bridge) technology using chiplets, which promotes high performance, and with PVC we can talk about working with PIM architecture, aimed at solving the problem of exchanging large amounts of data with memory [122]. The construction of three-dimensional processors from several crystals (dies) in [123, 124] is indicated as a progressive way to ensure the continuation of Moore's law, and three-dimensional memory began to be made quite a long time ago, which was the case not only for HBM (see, for example, [125]).

Here it is necessary to point out an alternative option for integrating different dies into a whole (Intel's tiles in PVC) using chiplets, used by AMD, including when building the MI200 GPU. For a modern overview of chiplets, see [126], and modern AMD GPUs are discussed below. It should also be noted that the standard chiplets interconnect, which is being developed by a consortium of a number of companies, including Intel, AMD, ARM, Google and TSMC, includes not only the physical layer — there is now a UCIe 1.0 specification [127].

Model PVC contains 100 billion transistors arranged in 47 functional tiles (thermal tiles are not counted here), combined into 5 nodes [116, 128–130]. Of this large number of tiles, two tiles are basic, 16 are compute tiles (see Figure 3 [131]), 8 are HBM2E memory tiles. The PVC is structured as 2 hardware stacks [129] — each of the two base tiles contains 8 compute tiles and 4 memory tiles (the logical relationships of these PVC components are presented in Figure 3). Basic tiles also contain PCIe interfaces and channels to HBM2E.

These tiles, as well as the interconnect tiles between GPU PVCs, X^e Link [129, 130] will be briefly discussed below. Rambo (Random Access Memory, Bandwidth Optimized) cache tiles were also planned for PVC [129], but for the X^e architecture in [114] they are listed as optional, and they are not included in the microarchitecture datasheet [130].

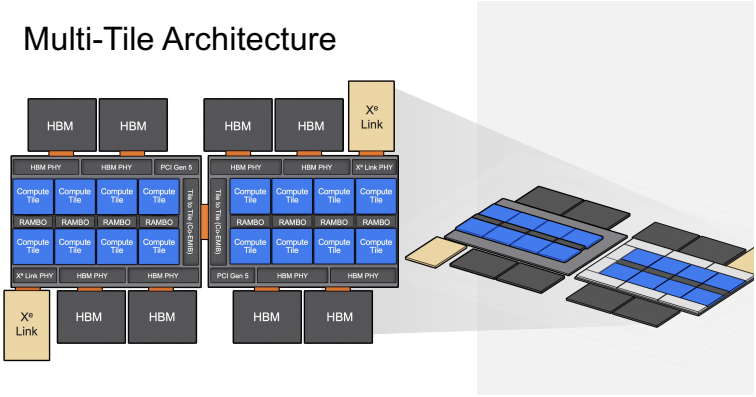


FIGURE 3. Tile-based PVC architecture (drawing from [129])

The implementation of advanced multi-chip 3D technology is discussed in more detail in [128]. Consideration of technologies is not within the scope of this review; It should be noted only the use in PVC of a 24-layer substrate, which ensures operation with overos three-dimensional stacking technology [132] and with 2.5D EMIB (Embedded Multi-die Interconnect Bridge) technology [133], which when used together they are called Co-EMIB [122]. EMIB is used to support local internal interconnects; Foveros with Intel 7 technology (formerly Enhanced 10nm SuperFin) is used in base tiles. Memory tiles are made using Intel 7 technology, Compute tiles are made using TSMC N5 technology, and HBM2E memory tiles are made using TSMC N7 technology. PVC is manufactured for the OAM 1.1 form factor [134]), which provides high equipment packaging density and, as noted above in Section 2, is intended to be used as a standard. Tiles are further considered not in technological terms, but simply as certain blocks of microarchitecture — accordingly, some types of tiles are not mentioned here at all.

It should be noted here that Intel also uses other terms (not just tiles) to refer to X^e class of microarchitectures hierarchies [69, 130]. Subslices are not used for PVC, they are an analogue of X^e cores in PVC. In PVC (Data Center GPU Max, below the level of the entire GPU, there are 2 levels of hierarchy — a slice (X^e HPC Slice) and a stack (X^e HPC stack). Slice has 16 X^e cores and 16 ray tracers. The stack contains 4 slices (respectively with 64 X^e cores and 64 ray tracing acceleration devices) [69, 115]. In addition, the stack has an L2 cache, a memory controller, a PCIe-v5 interconnect and 8 X^e Link channels (see below). PVC scales up to two stacks — up to 128 X^e cores [116, 130]. X^e cores are analogous to SM in Nvidia GPUs, and comparing their number is often used when comparing different GPUs (as well as the number of cores in the CPU).

TABLE 6. The number of operations per clock cycle performed in one X^e core and in the hierarchy of units performing calculations for different data formats [69]

Execution units	Data format	Number of operations per cycle
8 × XVE	FP64 FP32 FP16	256 256 512
8 × XMX	TF32 FP16 BF16 INT8	2048 4096 4096 8192
Slice — 16 times more execution units and operations per clock cycle		
Stack — executing blocks and operations per clock cycle are 4 times more		
Double-stack PVC (model 1550) — still 2 times more execution units and operations per clock cycle		

XMX does not support the use of the FP64 format

A general description of the PVC microarchitecture is available in [130]. Each computing tile contains 8 X^e cores, of which there are 128 pieces for the entire PVC. Each X^e core has 8 XVE vector engines with 512-bit vector lengths and 8 XMX matrix engines working with 4096-bit operands [69, 129, 130], and has a X^e Link, interface based on a coherent interconnect CXL [135] (also previously developed by Intel).

Accordingly, the PVC has a total of 1024 XVE vector engines and 1024 XMX matrix engines. XMX are an analogue of Nvidia tensor cores (this was indicated in Table 1 in the introduction), which are found not only in the V100 and A100 GPUs, but also in other Nvidia graphical processors [136].

XVE units operate on 512-bit operands and use multiply-and-add operations. This gives the X^e core 256 FLOPS per clock for FP64 (and for FP32 too) [129].

Table 6 shows the number of operations performed per clock cycle with the different data formats available for XVE and XMX [129, 130]. This allows peak performance P to be calculated using clock frequency ν. So, for FP64 or FP32 formats when working with XVE $P = 128 \times 256 \times \nu$, that provides 52.4 TFLOPS (Table 7 shows numbers rounded to the nearest integer, taken from the Intel overview [130]). Considering the possible values of ν, given in Table 5, it becomes clear that this numbers are calculated assuming operation at the maximum, and not at the base frequency.

From the data in Table 7, we can conclude that when using vector (without using matrix blocks) operations, peak performance for FP64, traditional for HPC, grows monotonically with the start date of new GPU models.

Table 7. Comparison of vector and matrix (XMX) peak performance of Intel PVC and GPUs of other companies for different data formats

Performance for different formats	PVC	A100	H100-SXM	H100-PCIe	MI250X
FP64 (TFLOPS)[1]	52	9.7	33.5	25.6	47.9
FP32 (TFLOPS)[1]	52	19.5	66.9	51.2	47.9
XMX TF32 (TFLOPS)	419	156	494.7	378	No
XMX BF16 (TFLOPS)	832	312	989.4	756	383
XMX FP16 (TFLOPS)	832	312	989.4	756	383
XMX INT8 (TOPS)	1664	624	1978.9	1513	383

[1] Data without matrix operations (XMX in PVC does not support FP64). With the use of tensor cores, the peak performance of the H100, A100 and MI250X is twice as high. Data for Nvidia H100 are taken from [78], for A100 — from [73]; data om PVC — from [130, 138].

In the field of AI the FP32 format is normal, and it's possible use FP16/BF16, including on XMX (see the text above in the discussion of formula (1)). The square root operation in FP32 format, according to [69], gives four results per clock cycle in XVE.

For Intel graphical processors, not very low frequencies are observed; For example, Arc Alchemist can reach more than 2 GHz [137]. Previously, in [129] it was indicated that the FP32 peak performance was expected to be at least 45 TFLOPS — this most likely means that Intel managed to increase the clock frequency compared to last year's prototype in PVC.

Nevertheless we must keep in mind that the power consumed in this case increases in proportion to the frequency, and increasing the frequency to maintain reliable operation of the chip often also requires an increase in voltage, the square of which is proportional to this power. The TDP of the PVC model under consideration is 600 W, and assumes liquid cooling [129] (see also comparisons of Data Center GPU Max models in Table 5). In [129] another PVC model with possible air cooling and a TDP of 450 W is indicated — this corresponds to the Data Center GPU Max 1350 model.

However, in mid-2023, Intel announced the discontinuation of the Max 1350 model due to the creation of a modified version of the Max 1550 with air cooling, and plans to produce a new Max 1450 model [139].

Table 8 provides a comparison of PVC (by default, this refers to the only model 1550 available at the time of writing the review) with other GPUs considered in the review in terms of other very important for performance indicators.

TABLE 8. Comparison of PVC hardware specifications with other modern GPUs

GPUs	PVC	H100	A100-SXM4-40GB	MI250X
Technology	Intel 7 (7 nm) & TSMC N5 & TSMC N7	TSMC 4N (5 nm)	TSMC N7 (7 nm)	TSMC (6 nm)
Memory capacity, GB	128	80	40	128
Memory type	HBM2E	HBM3	HBM2	HBM2E
Memory bus width, bits	8192	5120	5120	8192
Bandwidth, TB/s	3.3	3.0	1.6	3.2
Form factor	OAM	SXM	SXM	OAM
TDP, W	600	700	400	560

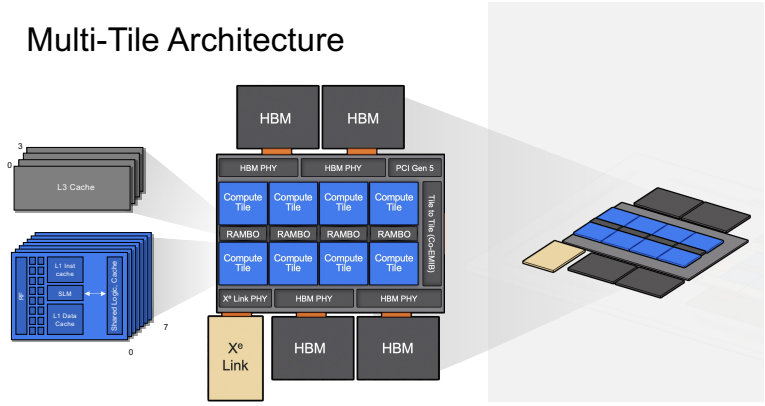


FIGURE 4. Memory hierarchy in PVC (Figure from [129])

Due to the high computing performance of modern GPUs, actual performance achieved when working with them is often limited by memory bandwidth, which happened before (see, for example, [140]), and now often happens (see, for example, [141, 142]).

A discussion of the PVC memory hierarchy should begin with the register file, which has a capacity of 64 MB (512 KB per X^e core) and provides a bandwidth of 419 TB/s [120]. The L1 cache is traditionally divided into an instruction cache and a data cache, which has a capacity of 512 KB per core [129, 143]. The PVC also has SLM (Shared Local Memory) with a total capacity of 8 MB for the entire PVC [69] — see Figure 4.

Each compute tile has 4 MB L1 [129], a total of 64 MB per PVC, with a bandwidth of 105 TB/ [120]. The 408 MB L2 cache (204 MB per stack)[144] on the base tile [129] has a bandwidth of 13 TB/s [120]. Data for other Data Center GPU Max models is shown in Table 9.

Table 9. Memory hierarchy and interconnects of data center GPU Max series models

GPU model	L1 cache, MB — per GPU/per one X^e core	L2 cache, MB –per GPU/per one X^e HPC stack	HBM2E capacity per one X^e HPC stack, GB	HBM2E bandwidth, TB/s	Number of X^e Link ports
1110	28/0.5	108	48[1]	1.2	6
1350	48/0.5	216/108	48	2.5	16
1450	64/0.5	408/204	64	no data	
1550	64/0.5	408/204	64	3.3	16

[1] Model 1110 has only 1 stack. The table data is taken from [69, 116] and [119]. Models 1350 and 1450 are marked in red because they are not listed on ark.intel.com at the time of writing.

In addition, in [129] it is indicated that the base tile contains a TLB buffer, traditional for conventional CPUs (which can also be considered some kind of cache), and also a RAMBO SRAM cache — see Figure 4. This cache consists of 4 banks with a capacity of 3.75 MB on a base tile that also contains a cache switch [129]. But the technical review of Intel Data Center GPU Max[130] does not mention the presence of a RAMBO cache.

The HBM2E memory itself in PVC combines up to 128 GB (8 tiles) and operates over a channel with a width of 8192 bits [120]. As stated in [69], the GTI (Graphics Technology Interface that connects the GPU to the rest of the computing system) bandwidth of a single PVC stack is 1024 bytes/cycle for reading or writing, which for a dual-stack PVC with a maximum frequency of 1.6 GHz (for 1550 model) provides bandwidth of approximately 3.3 TB/s (see Table 5). The bandwidth of this memory for other Data Center GPU Max models is also shown in this table.

Although each of the two EMIB-linked (up to 230 GB/s in both directions) stacks has its "own" L2 cache and 64 GB HBM2E memory [130],

Before considering scaling computing resources using the X^e Link hardware available in Data Center GPU Max which provides coherent interconnection between GPUs in the server [130], it is necessary to point out the differences between different GPU models of this series (see Tables 5, 9). In addition to the older Max 1550 model, there is also data on the Max 1450, 1350 and 1100 models (as noted above, Intel no longer plans to produce the 1350 model).

Of particular interest is the future Max 1450 model, which has the same number of X^e cores as the Max 1550. And Max 1110 consists not of two, but of one stack. The number of X^e cores in different models is proportional to the total capacity of HBM2E and L1 and L2 caches:

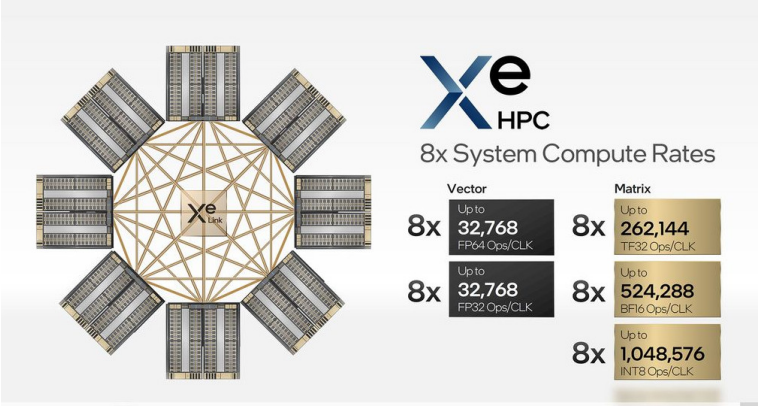


FIGURE 5. Interconnect between PVCs (Figure from [146])

in the Max 1100 model the corresponding capacities are 28 and 108 MB. Naturally, as the number of X^e cores in the models decreases, the TDP also decreases (see data in Table 5).

Here we can draw an analogy with three different models of GPU MI200 from AMD: the top model MI250X has two GCD (Graphics Compute Die), like the MI250 (which simply has fewer cores), and the bottom model MI210 has only one GCD (see below in the section about GPUs from AMD).

The bottom Max 1100 model is also distinguished by a reduced number of physical X^e Link ports (see Table 9), but it also uses a form factor different from OAM — PCIe AIC (add-in card) [116] (add-in card). PVC, which can be classified as a system on a chip (SoC) [114] has a PCIe 5.0 $\times$16 interface [130] with a bandwidth of about 63 GB/s [69]; In all Data Center models, GPU Max PCIe 5.0 is used to communicate between the GPU and the host [116].

The X^e Link tile supports embedded switch and 8 X^e Link channels, with each channel communicating directly to each (see Figure 5) [143]. And in two stacks there are respectively 16 X^e Link channels [69]. In [130] reported a per-X^e Link bandwidth of 26.5 GB/s in each direction. X^e Link is designed to provide coherent communication between X^e HPC stacks. This accordingly includes both internal communication in PVC (OAM) and between multiple PVCs (OAM) [130]. This interconnect, used for communication between multiple GPUs in a server, allows for efficient operation in SYCL with USM (Unified Shared Memory) mode [69].

What's important is how the Max 1550 and Max 1450 logically look to the user. Given the X^e Link bandwidth mentioned above, the

communication bandwidth between the two stacks is much lower than the memory bandwidth on a single stack. In the AMD MI250X/MI250, which is similar in this regard, therefore each GCD there logically looks like one GPU. In [144] states that users treat two PVC tiles as two processors. However, in Intel's oneAPI GPU Optimization Guide [145] states that moving from one stack to two requires simply changing an environment variable, meaning the Max 1550 gives you the choice of running it as two different GPUs or as one big GPU.

PVC officially began to be supplied by Intel already at the time of writing the review, and many details at the hardware level were not yet available, for example, latencies values, which is also important for communications via X^e Link, including between PVC stacks.

In [130], two types of solutions supported by Intel for scaling computing systems due to an increase in the number of X^e HPC stacks are indicated: scaling inside (scale-up) — inside multi-GPU servers, and scaling outside (scale-out), which refers to the cluster, containing GPU nodes. Above we actually talked about scale-up scaling.

The scale-out solution can scale to a cluster containing a maximum of 64 OAMs interconnected via X^e Link Glueless. Further scaling along with Infiniband allows to build a system with up to 512 interconnected OAMs [130]. "Glueless" interconnects ((where are no intermediate chips) have been known for a very long time; They provide very low latencies [147].

And first it is planned to use the Tuscany configuration, in which a common board with four OAMs can be connected to the server via PCIe 5.0. Options are available with Data Center GPU Max with TDP 450 or 600 W, with air or liquid cooling [130]. Taking into account the data in Table 5, they probably meant the GPU Max 1350 and 1550 models, respectively.

PVC was originally intended to be used in two-socket servers with the new Xeon Sapphire Rapids CPUs (now called Xeon Max they were used in Intel's server performance data [116]). Intel's currently offered Xeon Max supports DDR5 and HBM2E (see, for example, [148]). Using HBM in these CPUs can greatly improve performance, as shown in [149].

In the Aurora supercomputer, the computing nodes are dual-processor servers (with a Xeon Max 9480 CPUs) containing another 6 PVCs [47]. In addition to the Aurora supercomputer being created, PVC is planned to be used on the European exascale supercomputer with SiPearl Rhea processors [150]. Of course, this strengthens PVC's potential position in the market, but it will be more important to consider the realities of performance achieved in applications, transfer of software to PVC and

cost indicators. Data Center GPU Max of course, is intended to be used not only on supercomputers, but also at the level of small clusters and separate servers with GPUs — such servers have already been announced by famous manufacturers, including Dell, Lenovo and Supermicro.

3.2. Software tools and first initial data on PVC performance

As for software, the existing publication [120], taking into account the work for the Aurora supercomputer being created at the Argonne National Laboratory in the US [67, 151, 152] indicates the accelerated movement of Intel towards the SYCL standard line from the Khronos Group [153], extended by Intel in the form of DPC++ (Data Parallel C++) [154]. The relevance of this direction is clearly, since DPC++ is focused on working with different accelerators, which provides portability, including between different GPUs. Intel's DPC++ compiler is part of the overall Intel oneAPI software [155], which can also run on Intel processors and PGA accelerators, giving developers the ability to create a degree of common program code. Intel is actively developing new versions of oneAPI — for example, oneAPI 2023 is now available [155].

The oneAPI software includes a wide range of tools — compilers, debuggers, profilers, libraries and specialized tools for working with AI. The oneAPI DPC++/C++ compiler makes it possible to work with GPUs from Nvidia and AMD [156]. For optimization tasks when working with oneAPI, Intel has prepared a special guide [69]. A big advantage of DPC++ itself is the presence of an open source compiler [157].

In [158], for V100 on the STREAM (triad) benchmark, the bandwidth achieved with DPC++ was close to that obtained using OpenCL and CUDA, and the use of SGEMM from the oneMKL library gave 66% of the peak performance. Obviously, these results can be improved when working with new versions of oneAPI.

Since oneAPI is an open, standards-based unified programming model [159], aimed at working with both processors and accelerators of different types and from different companies [156], taking into account the end of the GPU monopoly from Nvidia and the transition to the use of GPUs from different companies, this is definitely promising. But it is clear that proposals from different companies may evolve in the direction of unification (for example, AMD's HIP software is already focused not only on AMD GPUs), and in this actively developing area it is now difficult to predict what will become the most used.

Intel, of course, has also taken care of the pressing issue of possibly porting code from Nvidia GPUs to PVC, and in oneAPI it also provides the DPC++ Compatibility Tool (often used abbreviation — DPCT), which automatically translates CUDA code to DPC++. According to [160], such automatic porting typically does this for 90–95% of the CUDA code, and full completing the porting of all code requires manual work. And how effectively all this will work on the GPU in terms of performance, of course, there is practically no data yet. In [154] does some preliminary research, but it does not apply to the GPUs covered in this review, and it is unclear how relevant it is to HPC workloads. In [161] based on the experience of porting SGEMM from the MAGMA library, it was concluded that DPCT can be successfully used for the initial port of CUDA code to DPC++, but the results used here are for GPUs that are not included in the review.

In general, as examples of work on porting GPU-using applications to DPC++ should indicate the porting of several well-known molecular dynamics applications to DPC++ — NAMD [162], GROMACS (ported to SYCL, using the DPC++ compiler [163, 164]); LAMMPS can also work on PVC — but through the use of Kokkos. Data about similar porting of other areas applications are available in [154]. In [165] discusses the use of SYCL on Nvidia and AMD hardware by using hipSYCL with different backends (now hipSYCL is renamed to *AdaptiveCpp*).

The tasks of porting program codes from CUDA to oneAPI have already been considered in a number of articles, since oneAPI tools are applicable to various graphical processors that appeared before PVC. Thus, in [166], for numerical integration problems, porting from CUDA (with V100) to oneAPI required special manual optimization before performance with oneAPI became slightly lower than with CUDA. In [167], when porting unstructured grid CFD code to SYCL for the A100 and V100, hipSYCL was used, but concluded that CUDA was still better for achieving maximum optimization. For the same CFD area, a similar result was obtained in [168]. These works noted that better results for DPC++/SYCL can be obtained in the future as the quality of optimization by compilers increases. The report at the supercomputer conference at NASA on the transfer of the FUN3D software package from CUDA to oneAPI (for CFD, with the solution of the Navier-Stokes equations) [169] essentially also indicated the advisability of manually optimizing the code for a specific architecture.

It is also worth pointing out that oneAPI has a certain disadvantage associated with being based solely on C++, since this does not help codes that natively use modern Fortran. For CUDA there is CUDA Fortran [170]. At the same time, the modern Fortran language when working on the

GPU began to be considered versions of the standard with support for object-oriented tools [171]. And modern versions of OpenMP, also implemented in Fortran compilers [172], have suitable tools for working with GPUs. But the above disadvantage is associated with delays in the development of possible new extensions of modern Fortran standards with tools for efficient work with GPUs.

A more detailed analysis of Intel's oneAPI is not practical here due to the very small number of scientific publications demonstrating the performance of PVC compared to other modern GPUs, including comparisons with work with other software such as CUDA. Intel has achieved a lot primarily with oneAPI, as these tools, including DPC++, have begun to be used on other Intel GPUs — for example, working with the well known Ginkgo library [173]. In addition, DPC++ learning for students studying programming of heterogeneous computing systems is already being implemented [174].

Among the isolated publications with data about the performance of Data Center GPU Max we point out two works in which the performance of the Max 1100 model was compared with Nvidia and AMD GPUs. Article [175] is focused on supporting modern versions of OpenMP, which allow to work on GPUs. Here, using Max 1100 (with Intel oneAPI 2023) and A100-40GB (with Nvidia NVHPC 23.3), calculations were performed on the CFD-related mini-application LULESH using OpenMP parallelization, and it was claimed that the A100 was 34% better than Max 1100 (although the performance gain for the A100 depended on the size of the problem used in the calculation and varied from 15 to 42%). However, the large memory size of the Max 1100 made it possible to perform LULESH calculations for higher problem size, where the memory on the A100 was not enough.

In [176], the portability of SYCL to different hardware platforms was studied and performance data was obtained on a number of different applications using parallelization with SYCL (mainly in the area of CFD), executed on Max 1100 (with oneAPI 2023.1), A100-40GB (with CUDA 11.6) and MI250X (with ROCm 5.4.2; Only one of the two GCDs of the full GPU was used here, see Section 5.1.1 below for more information).

All selected applications are memory-bound (which limits their performance) and use structured and unstructured grid methods. On the BabelStream triad benchmark, the memory bandwidth obtained in [176] was 1310 GB/s for the A100, 1290 GB/s for the MI250X, and 803 GB/s for the Max 1100.

These applications used high-level, method-specific parallelization tools that generated various lower-level parallelization codes, including SYCL. In [176], different parallelization variants were used on each GPU for each application. In the optimal (for A100 performance) variants, this GPU was noticeably faster in all applications than the Max 1100 in any variants. And with a similar comparison of the performance of the Max 1100 with the MI250X, in some cases the Max 1100 was faster, in others the MI250X.

The information on PVC performance discussed below applies to the Max 1550 model. The report [120] provides data on acceleration in PVC (using SYCL tools in DPC++) relative to the A100 using CUDA (although data was also provided for the A100 with SYCL) — according to 8 benchmarks, of which the traditional HPC area includes SYCL HP Linpack (obviously, an HPL implementation using SYCL). Here an acceleration of 1.5 times was achieved. Among other benchmarks, the most famous is Google's hashing benchmark (hashtable), in which a maximum speedup of 2.5 times was achieved. In other benchmarks, the acceleration achieved was in the range of 1.4 to 1.8. It is clear that all these numbers require more detailed clarification. Interestingly, in some of these tests, using SYCL on the A100 gave about 10 percent better performance than using CUDA.

Other data on the performance of PVC are given in [116] — about a 12.8 times acceleration of calculations using the famous LAMMPS molecular dynamics software package on a dual-processor server with an Intel Xeon Max 9480 and six Data Center GPU Max — relative to a dual-processor server with a Xeon 8380. Considering, that 6 GPUs are used here, and the comparison is made with the Xeon 8380, which are inferior by the maximum achieved floating-point performance in the widespread SPEC CPU2017 benchmarks to the AMD EPYC 7763 processors in the speed and rates variants [177], it would be more interesting to compare performance with other modern GPUs.

An important publication that provides real data on the performance of the Data Center GPU Max 1550 compared to the A100-80GB is [178], which provides not only data on the achieved performance, but also on porting code from CUDA to SYCL. Here, using the FP32 format, molecular docking calculations (an extremely simplified specialized technique for calculating the orientation of closely located molecules, for example, ligands relative to a protein), which are often used for drug design, were performed.

For this purpose, the CUDA version of the well-known AutoDock-GPU program was transferred to SYCL using DPCT tools, followed by manual modification. When moving from calculations on a single Max 1550 stack to calculations on two stacks (simply changing the environment variable), the performance of various test calculations increased from 6% to 58%.

It is worth noting, by the way, the found ratio between the performance of calculations achieved on the A100 using CUDA and SYCL versions of AutoDock. Of the 10 calculations performed, the CUDA version was faster (from 1.24 to 3.64 times) in nine, and the SYCL version was faster in one.

Calculations using the SYCL version of AutoDock using both Max 1550 stacks were faster than on the A100 with the CUDA version in 7 out of 10 calculations (maximum 1.88 times faster); A100 was more successful in larger calculations (with more atoms or numbers of rotatable bonds). But all the achieved accelerations of the Max 1550 relative to the A100 were less than the ratios of the peak performance (for FP32) of these GPUs. As noted in [**178**], the calculations here are still preliminary, and improvements are expected in the AutoDock code.

In [**179**], data were obtained on the achieved performance of PVC using package linear algebra tools from the ginkgo library for an iterative solver of the Navier-Stokes equations (in the PeleLM lattice hydrodynamics application). When switching from a single Max 1550 stack to dual stacks, performance increases by 1.5 to 2 times, and the larger performance gains apply to larger task sizes.

In [**179**] also obtained similar performance data for A100-80GB and H100-PCIe. A comparison of these results shows that the Max 1550 using a single stack is on average 1.3 times faster than the H100, and 2.4 times faster when using two stacks.

It should be noted that this work is also largely preliminary. Thus, it lists the peak performance of the Max 1550 as 45.8 TFLOPS — less than the current "official" value of 52 TFLOPS (a lower clock speed was probably available). The calculations on the A100 and H100 used CUDA 11.8.0, while with the H100 typically used version 12 (for example, all Nvidia H100 MLPerf benchmarks used version 12.2, see Section 4.2.6).

Even these few publications show the high achievable performance of PVC. Peak performance data is insufficient to make good estimates of actual performance. The Max 1550's greater peak performance compared to that of the H100 does not necessarily translate into an advantage in actual application performance achieved. It should also be noted that the peak performance data given above in Table 7 do not include data on the peak performance of Nvidia and AMD GPUs for FP64 using tensor cores (with their use, the performance of top models of these GPUs is higher, see Table 20), since XMX in PVC this format is not support. It is clear that for PVC and oneAPI/DPC++ tools to be practically successful, there is still a lot of work to be done, which will require some time.

3.3. Summary for Intel PVC

Technologically complex PVC production may not produce a very high percentage of usable PVC chips (which could also result from a slowdown in the supply of PVC for Aurora, which is not on the June 2023 Top500 list) and contribute to an increase in cost. In fact, the same thing was stated earlier in [**180**].

Both the Intel Max 1550 and the AMD MI250X can be considered to consist of two logical GPUs, which raises the question of which comparisons should be made — for one logical GPU or for a full (physical) one, which is also associated with cost indicators that can vary greatly.

When compared with the whole AMD MI250X, the PVC is slightly ahead of the MI250X in terms of peak performance in conventional vector operations (see Table 7), but also has a slightly higher TDP. But the peak performance advantage of the new GPUs from Intel and AMD relative to the Nvidia H100 does not mean an advantage in real application performance.

A disadvantage of PVC for certain HPC applications can be considered the lack of support for the FP64 format in XMX. When using the Tensor Cores in the H100-SXM4 (or the equivalent in the MI250X), these GPUs significantly outperform PVC in terms of peak double precision performance (see also Table 20 below).

Taking into account the fact that Nvidia already offers the GH200 Grace Hopper superchip with H100 integration with ARM processors (see Section 4.2.3), and at the CES (Consumer Electronics Show) in early 2023, AMD announced the MI300 GPU with an integrated CPU (in the form 3D chiplet, assumption of readiness — for 2023), from the author's point of view, a significantly wider use of Intel GPUs in HPC may be realized somewhat later, after the implementation of the expected successful Intel technological processes (especially 18A), which are not expected in 2023. Intel recently abandoned the "second generation" PVC previously planned for 2023, codenamed Rialto Bridge [**180**], containing 160 X^e cores, so we should rather expect the release of the Falcon Shores GPU or its subsequent XPU variant with integration of the GPU and x86 CPU [**181**] — success in technology probably determines everything.

Almost more important (compared to the competition between Intel and Nvidia for part of the GPU market — where AMD also participates) the author now seems to be the introduction of Intel-supported software for GPUs in the direction of SYCL — > DPC++ (and oneAPI), since this is not only allows us to move further away from a narrow focus on specific GPUs (Nvidia CUDA and partly AMD HIP), but it may also one day become

effective for new multi-core server processors, the number of cores in which continues to grow. Perhaps the use of SYCL may immediately become the most attractive for modernizing C++ applications that have not yet been ported to the GPU.

However, one should not exaggerate the capabilities of the new SYCL standard. Thus, the appearance in CUDA tools for H100 of a new level in the hierarchy of the thread system — a cluster of thread blocks (see Section 4.2.5) — has no analogues in SYCL. It should also be kept in mind that SYCL does not solve all performance portability issues, and it may be necessary to use different algorithms for different hardware [**176**].

4. GPU from Nvidia: from A100 to H100

As noted above, the base GPUs here include the Nvidia V100 and A100, which are currently most widely used in the HPC and AI fields, including in supercomputers. Since the V100 became available back in 2017, this microarchitecture review only covers the A100; V100 specifications are shown in comparison tables only. In addition, the most important computational units and data types that first appeared in the A100 (which were not present in the V100) are mentioned. Detailed information about all the improvements to the various components of the A100 microarchitecture relative to the V100 is available in [**73**].

The most modern of the "basic" Nvidia GPUs, the A100, has been produced since 2020, and a detailed analysis of its microarchitecture is not carried out here, considering the relevant information is already quite well known. But in the microarchitecture of the H100, naturally, a lot coincides with the A100, and accordingly, after analyzing the A100, it will be convenient to talk mainly about improvements in the H100. Accordingly, only the figure of SM from the H100 is shown here (the A100 is only slightly different here — and it will simply be stated below what the differences are). The same applies to software — the review focuses in more detail on what is appropriate to use on a new generation GPUs.

Data on the achieved performance of the A100 in applications and benchmarks are also considered somewhat limited, since the main purpose of the review was the new generation GPUs, the performance of which is also considered with a comparison with respect to the V100 and A100.

4.1. GPU A100

4.1.1. *A100 microarchitecture*

In current CPUs generations, HPC performance gains come primarily from microarchitecture advancements rather than from ISA improvements. n current CPU generations, HPC performance gains come primarily from microarchitecture advancements rather than from ISA improvements. In modern Nvidia GPUs, microarchitecture data is also the most important, but to maximize performance, especially for AI tasks, new improvements/extensions to all levels of the CUDA API and low-level warp are especially important. CUDA tools can be classified as low-level because it is possible to work at a higher level — for example, use only already ready library functions, or work with directives. There is a lower level in relation to CUDA, where a virtual instruction system (ISA) for Nvidia GPUs is used — PTX (Parallel Thread eXecution) [**182**], and accordingly an assembler [**183**], but it is, naturally, used very rarely in programming. Strictly speaking, PTX is an intermediate level — it is then converted into binary code for a specific GPU model.

By improving the architecture, the review primarily refers to the microarchitecture. But there is a more general architecture — for different models of graphics processors, its microarchitectures retain some of its basic common features.

The A100 uses Ampere GA100 architecture. The A100 was the first GPU released with this architecture in 2020. Then the other, including less computationally powerful, GA100 models appeared. They use a smaller number of SMs and contain a smaller number of tensor cores, but more high-performance ones. But unlike the A100, their tensor cores do not support FP64 [**184**]. In addition, after the US imposed sanctions against China, Nvidia began to produce the A800 GPUs, which is not subject to these restrictions, with reduced computing capabilities compared to the A100. Information about these GPUs is available, for example, in the well-known Techpowerup database on GPUs produced in the world [**185**] (on the Nvidia website consumers were offered to use this database). The review below only considers the A100.

The most complete consideration of the A100 architecture is in [**73**], on which the subsequent consideration of its microarchitecture in this review is based. The consideration in [**73**] is partly integrated with SIMT-based parallelization (using CUDA) due to CUDA's deep coupling with Nvidia GPU hardware. This also applies to the hierarchy of different types of memory in GPU and CUDA (extremely important components for realizing high performance), and to the hierarchy of different levels of formation of large groups of parallel threads in CUDA. Table 1 lists all this briefly in terminological terms. However, not all memory types used in CUDA have

TABLE 10. Comparison of Nvidia V100, A100 SXM and H100 GPU specifications important for traditional HPC (data from [73, 78, 185])

GPUs	V100	A100 SXM	H100 PCIe	H100 SXM
Architecture	Volta	Ampere	Hopper	Hopper
TSMC Technology	12 nm FFN	7 nm N7	4N[1]	4N[1]
Chip area, mm^2	815	826	814	814
Number of transistors ($\times 10^9$)	21.1	54.2	80	80
TDP (W)	300	400	350	700
Form factor	SXM2	SXM4	PCIe-v5	SXM5
Number of SMs	80	108	114	132
Number of FP64 vector cores in SM	32	32	64	64
Number of FP32 vector cores in SM	64	64	128	128
Number of INT32 cores in SM	64	64	64	64
Number of tensor cores in SM	8	4[2]	4	4
Boost Clock, GHz	1.53	1.41	1.76[4]	1.98[4]
Register file size in SM	256 KB	256 KB	256 KB	256 KB
Shared memory capacity in SM	Up to 96 KB[3]	Up to 164 KB[3]	Up to 228 KB[3]	Up to 228 KB[3]
L2 cache capacity	6144 KB	40960 KB	50 MB	50 MB
Memory type/capacity, GB	HBM2/16 or 32	HBM2E/40 or 80	HBM2E /80[5]	HBM3/80[5]
Memory bus width, bits	4096	5120	5120	5120
Memory Clock, MHz	876	1215 or 1593	1593	1313
Memory bandwidth, GB/s	897	1555 or 2039	2039	1681

[1] modified for Nvidia;
[2] The number of matrix multiply-and-add operations per clock cycle in the A100 tensor cores is 4 times greater than that of the V100;
[3] Shared memory capacity in V100, A100 and H100 is configurable;
[4] for FP32 and FP64, for less precision the frequency is slightly lower;
[5] there is a model with HBM3/96 GB and a different GPU frequency.

a unique hardware equivalent. There is also the integration of various types of CUDA memory into one hardware type of memory A100. In addition, [73] sometimes refers to the V100 architecture discussed in the other Nvidia description.

The basic specifications of the A100 in comparison with the similar specifications of the V100 and H100 are shown in Table 10. In technological terms, due to the more modern 7nm TSMC technology in the A100, with almost the same chip area, the number of transistors has more than doubled compared to the V100, but the TDP has not increased as much strongly.

Table 10 shows data for two different A100 models with the SXM form factor, differing in memory capacity (40 or 80 GB). In addition, there

are two other A100 models with PCIe interconnect, which can also have capacities of 40 or 80 GB. This somewhat limited representation of the A100 models in this table was chosen for two reasons. Firstly, for ease of reading and comparison with other GPU models from Nvidia in the format of this publication. And secondly, the peak performance discussed at this point in the review (see Table 12 below) was calculated for increased clock speeds, which are the same for all four different A100 models. In addition, further in Section 5, which describes the new generation GPUs from AMD (the comparison with which seems to be one of the most actual in this review), a more detailed comparison of all characteristics, including performance, with different A100 models is carried out (see Table 20 and Table 21).

For a starting understanding of the performance of the A100 (to begin with the peak) it is natural to base it on the SM execution units: the peak performance of the entire A100 is simply proportional to the number of SMs, which is indicated in Table 10. In the hierarchy from one SM to the full A100 there are 2 types of clusters: texture processing clusters (TPC, Texture Processing Clusters), containing by 2 SMs, and GPC clusters, containing by 8 TPCs. In the Ampere architecture (in GA100), it is permissible to have 8 GPCs and, accordingly, 128 SMs [**73**]. And the sense of GPC from the point of view of GPGPU (if we ignore textures): it is a group of SMs physically located close to each other, which gives locality of parallel processed data with access to them with higher bandwidth and lower latencies (so to speak, a kind of localised analogue of PIM).

The general construction of SMs, the many of which make up the computing power of the A100 (also both V100 and H100) can be seen in Figure 6. Although this is a figure of an SM from the H100 [**78**]), at the macro level of the figure, the differences in the A100 from the H100 are easily visible: the A100 does not have Tensor Memory Accelerator; L1 D-cache capacity is 192 KB versus 256 KB on H100; And in H100, accordingly, there is a new version of the tensor core. But the main thing is that in the H100 in each of the 4 SM sections (what is meant here by the SM section is clear from Figure 6) there are 2 times more vector FP64, FP32 and INT32 engines. In the A100, each section has 8 FP64 enginess, and 16 FP32 and INT32 engines.

Since SM runs warps consisting of 32 threads, each partition has a warp scheduler that produces 32 threads per clock and a dispatcher with the same "performance". This reveals the close intertwining of Nvidia GPU hardware and CUDA tools. A block of CUDA threads (containing warps) is assigned to one SM. And clarification of the operations of the warp scheduler is considered not in [**73**], but in the CUDA manual [**16**], where it is stated that SM statically distributes its warps between its schedulers. And then each scheduler issues one instruction for one of its assigned warps,

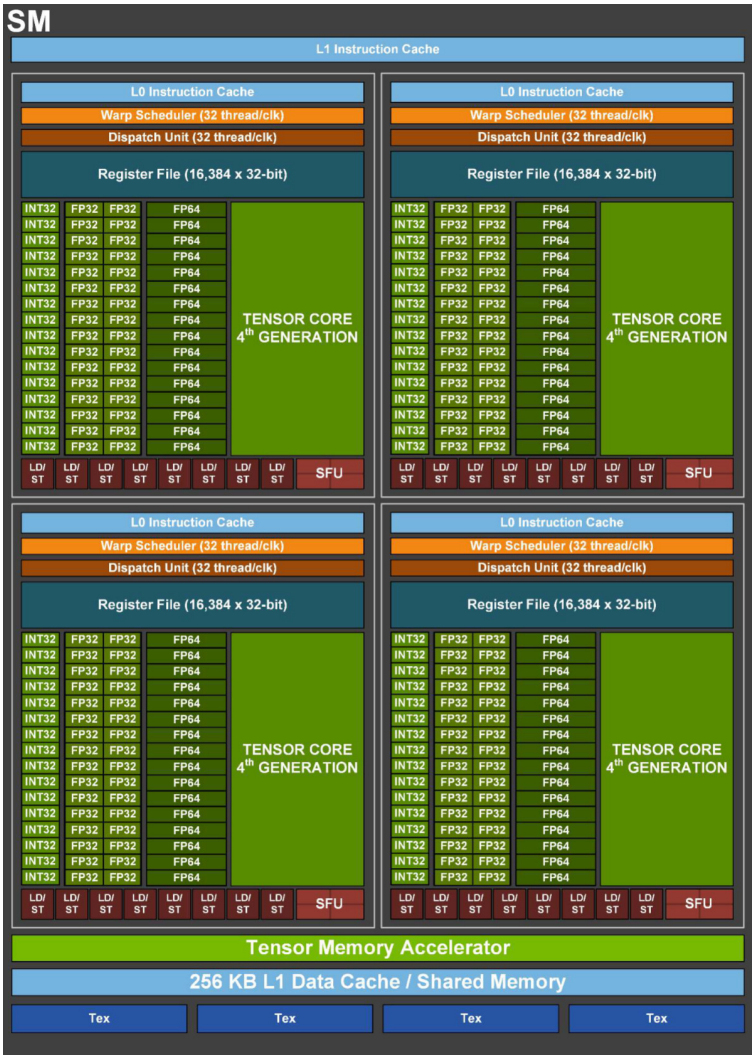


FIGURE 6. General construction of SM in GH100 (Figure from [78])

which is ready to be executed, if there is one. The warp scheduler is used to ensure that the computing engines of the SM section are loaded: if a warp is waiting, for example, for memory operations to complete, then another warp can be executed. And the dispatcher redirects the selected commands to computing engines (they take data from registers).

Table 11. Technical characteristics of different versions of Compute Capability (CC) depending on Nvidia GPUs

GPUs	V100	A100	H100
Compute capability version	7.0	8.0	9.0
Threads per one warp	32		
Maximum warps per one SM	64		
Maximum thread blocks (CTA) per one SM	32		
Maximum thread blocks/thread block clusters	No		16
Maximum number of 32-bit registers per thread	255		
Maximum number of registers per one SM	65536		
Maximum number of registers per block	65536		
Maximum number of threads per block	1024		
Shared memory per one SM, KB, Up to	96	164	228
Number of FP32 cores per one SM	64		128
Number of FP64 cores per one SM	32		64

The data in the table is taken from [78]

The actions of the warp scheduler may depend on the version of Compute Capability described in [16] for different GPU models; The above is true for V100, A100 and H100. These capabilities for these models are shown in Table 11.

Each of the 4 SM sections has a register file, and 8 load/store devices for memory access. Each SM section has its own L0 I-cache (the L1 I-cache is shared across the entire SM), but typically performance in HPC and AI does not require code to be specifically targeted to their use. When analyzing the performance of Nvidia GPUs, they often start from the SM level, since they also contain hardware components common to all 4 sections, including texture memory and a common L1 D-cache (SM can be considered some kind of analogue of the processor core in the CPU).

Each SM in the A100 and V100 has 32 vector CUDA cores for FP64, which allows the SM to perform 32 multiply-and-add operations per clock, resulting in 64 FLOPS per clock (for FP32 — everything is twice as much). Accordingly, the peak GPU performance when working with these vector cores is obtained by multiplying 64 FLOPS by the number of SMs (there are more of them in the A100 than in the V100) and the clock frequency, which gives higher performance for the A100 (see Table 10). But the main advances in performance are now focused primarily on AI and, accordingly, on the use of tensor cores, which can produce more results per clock cycle than conventional vector units — and correspondingly higher peak performance for various data formats.

TABLE 12. Peak performance of GPUs V100, A100, H100 (TFLOPS, for integer operations — TOPS)

Data format	V100	A100	H100 PCIe	H100 SXM4
FP16	31.4	78	204.9	267.6
FP16[1]	125	312/624	756/1513	989.4/1978.9
BF16[1]	—	312/624	756/1513	989.4/1978.9
FP32	15.7	19.5	51.2	66.9
TF32[1]	—	156/312	378/756	494.7/989.4
FP64	7.8	9.7	25.6	33.5
FP64[1]	—	19.5	51.2	66.9
INT8[1]	—	624/1248	1513/3026	1978.9/3957.8

[1] values when using tensor cores.
Following the slash numbers are the performances using sparsity when available.
Data from [73, 78, 185].

Tensor cores in A100 work with matrices of different possible sizes depending on the data formats used (a list of all possibilities is available in [16]). Traditional HPC requires working with FP64, which was first supported in tensor cores at the hardware level in the A100 — FP64 is what we will be talking about here. And data on peak performance for other data formats, primarily actual for AI, are shown in Table 12.

The A100 introduces a hardware implementation of the calculations for formula (1) for double precision, a new matrix instruction "multiply-and-add" (Double Precision MMA), which replaces the 8 similar DFMA instructions in the V100 and produces 128 FP64 results per clock — 2 times more than V100 [73]. For FP64 in A100, at the lowest warp level, the WMMA operation is applicable, with which you can work with matrices of dimensions 8×4 or 4×8 and accumulator matrices 8×8 (i.e. $M=N=8$ and $K=4$ for formula (1)) [16]. As noted in [89], WMMA operations can be performed on slightly larger matrices than supported by the tensor core hardware, and multiple tensor cores are used simultaneously when performing a WMMA operation.

The Nvidia shipped A100 has 108 SMs [73], so the peak performance achieved is $108\times128\times\nu$, where ν is the clock speed. It follows that the peak performance information provided by Nvidia in [73] and used in Table 12 refers to the boost frequency. A100 users have the ability to control the clock frequency, which allows them to adjust the TDP and stabilize the measured performance — for example, in [186]) a frequency of 1005 MHz was used to study the performance achieved by GEMM.

Table 13. Base clock speeds of various A100 models and their memory

Model GPU A100	Base frequency	Memory frequency
A100-PCIe-40GB	765 MHz	1215 MHz
A100-SXM4-40GB	1095 MHz	1215 MHz
A100-PCIe-80GB	1065 MHz	1512 MHz
A100-SXM4-80GB	1275 MHz	1593 MHz

The current Nsight Compute profiler [**187**] for the A100 defaults fix to a base clock speed, which gives repeatable results, but correspondingly lower peak performance than those listed in Table 12. Base frequencies depend on the specific A100 model used (these are the 4 most important for this review — they were usually used for calculations and benchmarks for HPC and AI), they differ in HBM2E size and form factor. Table 13 shows the values of their base frequencies from the database [**185**].

But in addition, the memory frequencies used and, accordingly, its bandwidth differ in different A100 models. Only two A100 models with 40 GB memory capacity have the same frequencies, and since the memory bus width (5120 bits) is the same for all A100 models, the theoretical bandwidth of the A100 models with 40 GB memory is the same (it can be calculated by multiplying the bus width by memory frequency). Table 13 also shows the memory frequencies of different A100 models (data from [**185**]).

But let's return to the tensor cores that determine the maximum peak performance of the A100. Even with such small hardware-supported matrices, they still have the ability to take into account fine-grained sparsity, which applies to formats with reduced precision relative to FP64 and gives there a two-fold increase in peak performance [**94**] (see Table 10). This sparsity was not supported in V100 and is mainly targeted at AI tasks (see [**73**] for details).

Since many HPC applications do not require matrix multiplication (the feasibility of supporting it in hardware is generally debated [**90**]), equally important to the double-precision performance of the A100 is the peak performance of the FP64 vector CUDA cores, which are not related to the tensor cores (see Figure 6).

One SM has 32 FP64 vector units (the corresponding FP32 vector units, of which there are twice as many in each SM, are traditionally called CUDA cores) [**73**]. For all 108 SMs, this gives a corresponding 6912 results per clock (thanks to the use of the multiply-and-add instruction), which when multiplied by the boosted frequency gives the A100 a peak performance of 9.7 TFLOPS for double precision — without the use of tensor cores (see Table 10).

In addition, each SM has 4 SUs, that speed up the calculations of transcendental functions. This can be practically important for a wide variety of applications, but SFUs have been available on Nvidia GPUs for a long time, even before the V100, but only for the FP32 format (see [188] for more details).

Since achievable GPU performance is so often dependent on memory bandwidth, we now need to look at the memory hierarchy in the A100. Because traditionally large numbers of threads running in parallel on a GPU require many registers, the A100, like older GPUs, has a 64 KB 32-bit register file in each of the 4 SM sections (see Figure 6). Each executing thread uses its own local registers from the corresponding file.

The next level in the memory hierarchy in the A100 is the L1 D-cache and shared memory with a total capacity of 192 KB. It provides high bandwidth and low latency; Real data are available for [189]. This memory is common to the entire SM (and accordingly to the block of threads in CUDA). In CUDA, it is possible to dynamically change the amount of shared memory.

The penultim level of the memory hierarchy on the path to the HBM2E global memory, the 40 MB L2 cache, is discussed in [73] in a common subsection with HBM2E. The spaces of these memory levels are common to all SMs and all applications running on the GPU. The L2 cache capacity has increased more than 6 times compared to the V100. This cache reads and writes to the HBM2E device's memory. Higher parallelization in the A100 required higher bandwidth per SM. To achieve this, the L2 cache split into partitions using a hierarchical crossbar structure, and each partition caches data nearer to the accessing SMs, reducing latency [94].

And the increased width of the HBM2 interface and memory frequency compared to the V100 gave an increase in bandwidth by approximately 1.7 times (see Table 10). To increase the effective DRAM bandwidth and L2 cache capacity, the A100 hardware features of sparse data compression provides up to 4× compression in DRAM, up to 2× compression in L2 cache for AI workloads [190].

CUDA uses its own types of memory, some of which actually reside together on the same type of hardware memory. Unlike shared memory, local memory is the private memory of each thread. In CUDA terminology, global and local memory areas (as opposed to global, the latter is cached [16]) are called device memory and are located in HBM2E. Constant memory (lives only while the application is running, this memory is read-only) is located in the device memory (constant memory is cached). As for the texture memory, it is located in each SM (see Figure 6) and cached in a special cache [73].

The A100 introduces MIG tools to address a possible data center problem of underutilization of A100 resources by certain applications where the use of A100 would become correspondingly ineffective. The A100 can function as up to 7 isolated GPU instances [73], reconfigurable on the fly to meet dynamically changing needs [94], i.e. the result is virtualized GPUs with smaller computing resources.

The new MIG feature provides improved client (including virtual machines and processes) isolation and QoS for multi-tenant and virtualized GPUs, which is especially useful for cloud service providers [73]. New fault handling technologies in the Nvidia Ampere architecture are very important to MIG to ensure proper isolation and security between clients using the same GPU.

As stated in [94], two types of MIG instances are supported. One type isolates computing resources but not the memory system, allowing the operating system to schedule processes with simplified administration. The other type further provides functional and performance isolation in the memory system. In this case, A100 assigns each MIG its own physical paths (including to the L2 cache and memory interface), providing additional security for cloud computing [94].

Using MIG allows you to expand the exeediency of using of the A100 to a wider area of work. For more information on setting up and using MIG, see [191].

Nvidia's A100-PCIe-80GB manual [192] describes software power management capabilities that allow to customize graphics card's power consumption or performance per watt. The A100 naturally has many more features that are important for achieving high performance, among which we should first of all mention support for asynchronous copying (at the ISA level) — it loads data directly from global memory into shared memory in SM, bypassing the register file, and can run in the background while the SM performs other calculations (this also reduces power consumption). For more information on this and other A100 features, see [73]; A brief discussion of asynchronous data transfers and computations is provided later in Section 4.2 about the H100, in which the corresponding capabilities have been significantly expanded.

A description of A100 interconnects using NVLink channels is provided below in Section 4.1.2, since it partly applies to hardware additional to the GPU — this also includes connections to the CPU via PCIe.

4.1.2. *Interconnections for A100 and computing systems with A100: from servers to supercomputers*

NVLink interconnects for communication between multiple Nvidia GPUs in a server predate the V100. Initially, there was a problem that PCIe bandwidth was not high enough to connect the GPU to the CPU. A general overview of the interconnects used by various companies for GPUs is available in [**193**]. NVLink is a channel, and devices can have multiple channels and communicate through a mesh network.

Naturally, the second version (NVLink2) used in the V100 has a lot in common with NVLink3 in the A100. Various performance metrics for NVLink2 running with V100 are discussed in detail in [**194**]. NVLink2 (instead of the slower PCIe 3.0) is used to both communicate with IBM Power9 and provide cache coherence.

Like PCIe, NVLink2 is a packet bus, but NVLink2 is more efficient when transferring small packets — with 16 bytes of header, 256 payload bytes can be transferred. This interconnect uses a mesh topology for point-to-point connections, resulting in higher overall bandwidth compared to the tree topology in PCIe. Connections consist of multiple full-duplex links that exchange data at 25 GB/s in each direction. The device has up to six channels, and they can be combined into 3 channels with a total speed of 75 GB/s. Both PCIe and NVLink provide bidirectional transfers, but NVLink also has direct access to CPU page memory [**194**]. In [**194**] are given data on the achievable bandwidth and latency of NVLink2.

Using a switch to communicate between Nvidia GPUs not only improves performance scalability, but also provides an extremely important (especially for modern AI tasks) increase in the amount of available GPU memory. NVSwitch is a non-blocking switch (inernally — fully connected crossbar), has 18 NVLink ports and makes it possible to connect up to 16 V100 GPUs via NVLink2, providing each channel with a bidirectional bandwidth of 50 GB/s — and, accordingly, a total switch bandwidth of 900 GB/s [**195**]. For example, in implementations on motherboards containing 8 GPUs, each of them is connected to 6 switches on the board, which also have connections with switches on another board. GPU communications inside the board require one NVSwitch traversal, and with the GPU of the second board — two NWSwitch traversals. Cyclic Redundancy Coding (CRC) is used to ensure communication reliability on links, and ECC is used within switches (for example, for routing) (see [**195**]). for details). Bandwidth limitations on servers with V100 may occur more likely when transferring data over PCIe between the GPU and CPU.

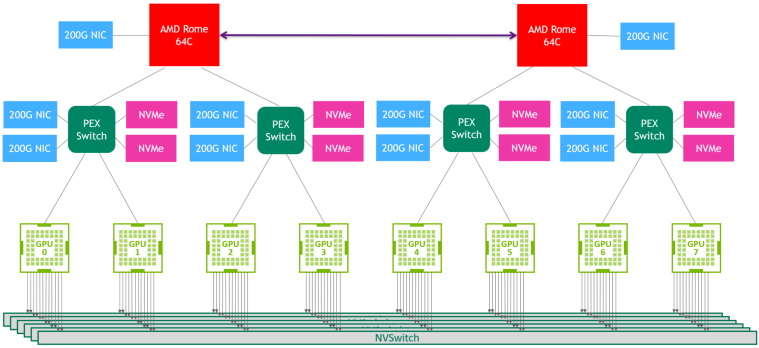


FIGURE 7. GPUs interconnect in the DGX A100 server (Figure from [73])

Multi-GPU servers with A100 already use NVLink3. Low-level details explaining why increased bandwidth and reduced latency relative to V100 with NVLink2 were obtained, are described in [94]. From the user's point of view, the main thing is that with no change in channel bandwidth, the number of channels has increased from 6 on the V100 to 12 on the A100.

In multi-GPU servers using NVLink, various specific topologies are accordingly possible [190]. For example, the AI-oriented DGX A100 server has 8 GPUs and 6 NVSwitch chips, and each GPU is connected to each NVSwitch using two NVLink3 links [196]. This is illustrated in Figure 7 (PCIe PEX switch is used to communicate with EPYC Rome) [73].

NVLink3 also provides improved error detection and recovery features [190]. Although, as stated above, the A100's single NVLink channel has the same throughput as the V100 — 25 GB/s in each direction — but it uses half as many signal pairs per channel compared to the V100. Doubling the number of channels in the A100 compared to the V100 gave a total throughput of 600 GB/s for the entire A100, compared to 300 GB/s for the V100.

In [3], on different servers with 4 V100 GPUs and on a DGX A100 server with 8 A100 and two 64-core AMD EPYC 7742 processors, the throughput of data transfers from the host to the device or vice versa was measured. For the A100 with PCIe-v4, it was about 25 GB/s (about three-quarters of the theoretical value of 32 GB/s for PCIe×16), which corresponds to the authors' expectations [3]. This work also revealed a slight decrease in the bandwith of bidirectional data exchanges between CPUs with "remote" A100 compared to "local" ones (having a direct PCIe connection with the corresponding CPU), which was due to competition for work with PCIe.

It is clear that calculations with multi-GPU servers to achieve the expected high performance may probably require significant modernization of programs or applications that work with one GPU (and even perhaps more subtle optimization taking into account some "asymmetry" of different GPUs in the server). Examples include works [197, 198]. Such modernization also includes algorithms, and can cover the most basic things — for example, GEMM (see [199] and [200]).

It is clear that servers containing one or more A100 GPUs are being supplied by all leading server manufacturers. Multi-GPU servers are focused primarily on AI workloads, although HPC software is emerging in areas where GPUs have not been used much at all. For example, the QUICK application implements parallelization at the multi-GPU level, including for the most widely used quantum chemical method DFT in the Gaussian basis [198].

The DGX A100 is an AI-focused Nvidia famous "classic" server containing 8 A100 GPUs and 6 NVSwitches [201]. It uses the HGX A100 module for this, and gives a specific fixed SXM4 interconnect topology. The A100 has two variants with a memory size of 40 or 80 GB, respectively, there are two server options — DGX A100 320GB System and DGX A100 640GB System, with memory size of 1 and 2 TB per system, respectively. The DGX A100 has two 64-core EPYC 7742 Roma processors, which are connected to the A100 via PCI switches (with PCIe-4.0 $\times$16 buses). For communication between servers, Nvidia ConnectX-6 or ConnectX-7 adapters (Infiniband HDR /200 Gb/s Ethernet) are used [202].

The DGX A100 is a ready-to-use system supplied with an operating system (based on Ubuntu Linux) containing special management and monitoring tools. For details (including the implemented topology of the interconnects used), see [202].

In addition, Nvidia is reviving a class of workstations that has not been used for a long time — with the DGX Station A100 offering (for different types of DGX systems, see [203]).

Nvidia has gone even further in providing fast deployment computing systems and created the DGX SuperPOD platform, based on building blocks with racks containing of five DGX A100, with the delivery of cluster systems having from 4 to 28 racks (see, for example, [196]). This arrangement of building blocks was then modernized [204]. These AI-centric systems can be deployed in just a few weeks [196, 204]. For example, in the Top500, such a supercomputer Nvidia Selene based on DGX A100, installed in 2020, takes 9th place. Obviously, the DGX A100 and larger systems are aimed at simplifying work in their respective data centers.

Let's point out even more powerful supercomputers using the A100, which are included in the top ten Top500. The computing nodes of the Italian supercomputer Leonardo (installed in 2022), which ranks fourth in the Top500 [205], contain one 32-core Intel Xeon Platinum 8358/2.6 GHz processor (Ice Lake) and 4 GPU A100-SXM4-40GB, and for communication between nodes is used Infiniband HDR200 (Nvidia products with Dragonfly+ topology).

And the eighth place in the Top500 is occupied by the Perlmutter supercomputer installed in 2021 of the US National Energy Research Center (NERSC, and the operator is the American Lawrence Laboratory, LBNL, famous in the supercomputer world) [206]. Its main computing nodes use a 64-core EPYC 7763/2.45 GHz processor and 4 A100 GPUs (most nodes use A100-40GB, and another 256 have A100-80GB). Perlmutter uses the well-known HPE Cray EX235N supercomputer hardware and the corresponding HPE Slingshot 11 interconnect. Interestingly, these last two of the above-mentioned supercomputers, focused not only on solving AI problems, use 4 A100 per node.

4.1.3. *Nvidia SDK Tools for A100*

General composition of SDK tools. Like the A100 hardware discussed above, the SDK tools are also predecessors to the corresponding H100 tools. But SDKs are constantly evolving, and different versions of them can usually work for both base Nvidia GPUs and next-generation GPUs. After analyzing the A100 SDK here, in relation to the H100 SDK it is enough to point out the improvements.

Nvidia's SDKs have become well known due to Nvidia's clear dominance of the GPU market for many years. The basis for HPC is, of course, NVHPC tools — see the developer's website, [207]. There is also a complete list of components included in these software tools, most of the names of which (primarily mathematical libraries) already explain why they are needed. The ability to freely access NVHPC is a plus for Nvidia. At the time of writing this review, HPC SDK Version 23 was available [208].

NVHPC includes compilers supporting x86-64 (AMD and Intel), OpenPower and ARM platforms. They call the corresponding language compilers (with support for OpenMP and OpenACC tools for the CPU), assembler and link editor for target processors with command line parameters. These compilers are nvc (ISO C11), nvc++ (ISO C++17) and nvfortran. The latter (supports Fortran 2003 and a number of Fortran 2008 features) can use the parallelization features of Fortran, OpenMP and OpenACC to work with the GPU.

Finally, nvcc, a CUDA C/C++ compiler driver for Nvidia GPUs, uses GPU-specific options to generate optimized code for Nvidia GPUs and manage the host compiler. The latest version of CUDA at the time of writing was 12.2. nvcc hides the complex details of CUDA compilation from the developer, and redirects all non-CUDA compilation steps to the host compiler with special command line options [**208**]. The effective development of nvcc is facilitated by its basing on the famous LLVM compiler [**209**]. Further Section 4.2.5 about CUDA tools for the H100 describes the general design of different Nvidia compilers down to the runtime level, including new previously unused programming languages.

HPC SDK math libraries include, in particular, cuBLAS, cuSPARSE, cuRAND, and cuSOLVER (for solving linear algebra problems, including eigenvalue problems). cuSOLVERmp provides these functions when working with distributed memory in multi-node and multi-GPU systems. The cuFFTmp library complement the cuFFT library to perform similar advanced functions as in cuSOLVERmp.

The low-level cuTENSOR library is designed for linear algebra problems on tensor cores, and can be used not only for AI problems, but also in HPC areas. Two very important libraries provide communications. These are NCCL (Nvidia Collective Communications Library), which provides communication primitives for multi-node and multi-GPU systems with Nvidia GPUs, and NVSHMEM for PGAS parallelization using the OpenSHMEM standard. Here the memory can, in a sense, be integrated in a cluster with Nvidia GPUs in the nodes, using communications with NVLink, PCIe and Infiniband.

NVHPC includes a number of other tools, including the CUDA-GDB debugger and the Nsight Compute profiler (it is possible to use the NVTX profiling library to work with it) [**208**].

It should also be noted that traditional MPI parallelization tools on multi-node clusters containing GPUs can naturally also be used. But what may be important here is the use of such MPIs, which can use the long-standing hardware and software capabilities of Nvidia GPUs and CUDA tools — GPUDirect, which provide data transfer (RDMA) via PCIe bypassing the CPU — directly through the network card. Such capabilities have long been available, for example, in MVAPICH2 [**210**].

CUDA Toolkit. CUDA is the most famous and widespread API for working with GPUs, discussed in many publications. AMD has developed similar HIP tools for its GPUs [**58**]. Intel's OneAPI uses DPC++, but the

overall parallelization design there also uses the SIMT model originally proposed by Nvidia, and oneAPI uses terms similar to those used in CUDA (see Table 1 above). The hardware of modern GPUs is closely connected with SIMT, so an ultra-brief explanation of the basics of CUDA is also necessary here (for a full description, see [**16**]).

Roughly speaking, in the CUDA programming model, sequential code is written, which is executed in parallel by many threads on the GPU, and each thread works with its own part of the common data, for which it receives its own identifier. Although a CUDA program running on the host can use not only C but also C++, the core software running on the device uses extensions relative to C.

In CUDA, the original task is divided into a set of independent subtasks, which are calculated on their own thread blocks. Each subtask has its own block of threads, and it is solved by all threads of this block. Threads can interact with each other only within the same block. A thread block is an array of threads that can be one-, two-, or three-dimensional.

Threads inside a block of threads use shared memory (ultra-fast — roughly speaking, at the cache level) and interact through barrier synchronization (when accessing this function, further work is impossible until all threads enter it). And global memory is used to exchange data between different blocks of threads.

The thread block is the central object for parallelization programming in CUDA (although with the advent of H100, a new level has appeared above the thread block in CUDA, the thread block cluster, this is unique to H100). The top level of the thread hierarchy — above the thread blocks — is the thread grid, a one-dimensional, two-dimensional or three-dimensional array of thread blocks.

All blocks of threads have the same dimensions and size (number of threads in a block). Thread grid size (number of thread blocks) and block size are built-in variables. Any block of threads in a grid has a block index in the grid. Each thread has an index specified by one, two or three non-negative integers.

For indexing threads and blocks, the software core uses, for the "most scalable" case, three-dimensional integer vectors — the built-in variables threadIdx and blockIdx.

Each block of threads is divided into warps, and all threads of a warp belong to one block. Only threads within the same warp are physically executing simultaneously. And threads at different warps can be at different

stages of program execution. There is no need to exlicitly work at the warp level in a CUDA program. All Nvidia GPUs discussed in this review have warps of 32 threads. Working at the warp level may only be necessary to maximize the achievable performance in CUDA.

Nvidia CUDA is not limited to being based solely on C/C++. We will illustrate working with CUDA using the example of CUDA Fortran, which is attractive because Fortran remains popular in HPC (the ability to use Fortran to work with Nvidia GPUs is a significant advantage of the company). The newest (as of the June 2023 Top500 list) version of nvfortran was 23.3.

CUDA Fortran extensions allow you to write subroutines and functions in a Fortran program for execution on the graphical processor, including declaring variables allocated in GPU device memory and allocating dynamic memory in GPU device memory. Naturally, CUDA Fortran has support for copying data from host memory to GPU memory and vice versa, and calling GPU routines from the host. CUDA Fortran provides software access to tensor cores, cooperation with CUDA C, the use of asynchronous transfers between the host and the GPU (asynchronous transfer allows calculations to be performed on the device simultaneously with data transfer) and many other capabilities needed for modern GPUs [**211**].

Below is a simple example of how a program is built in CUDA Fortran (quoted from the Nvidia manual, [**211**]).

On the host

```
program t1
use cudafor
use mytests
integer, parameter:: n = 100
integer, allocatable, device::
    iarr(:)
integer h(n)
istat = cudaSetDevice(0)
allocate(iarr(n))
h = 0; iarr = h
call test1<<<1,n>>> (iarr)
h = iarr
print *,&
"Errors: ", count(h.ne.(/ (i,i=1,n
    ) /))
deallocate(iarr)
end program t1
```

On the device

```
module mytests
contains
attributes(global)
    &
subroutine test1( a
    )
integer, device:: a
    (*)
i = threadIdx%x
a(i) = i
return
end subroutine test1
end module mytests
```

On the left is the code that runs on the host, and on the right is the code that runs on the device.

In the host code, the use of the `cudafor` module (line 2) provides interfaces to the CUDA host runtime library (in this program, to `cudaSetDevice()`, where the device number 0 is selected — this is the API call on line 7). Line 3 indicates the use of a module on the device (this module contains the test1 subroutine that is called). The 8th line of the program allocates the iarr array on the device, and the next line initializes data on both the host and the device [**211**].

The kernel running on the device is called on line 10; `<<<1,n>>>` here means that the kernel is executed by n GPU threads (more generally, n in this syntax indicates the number of threads in the block, and before the comma the number of thread blocks is indicated, which here is 1). On line 11, the results of the kernel execution are transferred to the host array, and on line 14, the CPU array is deallocated.

On the right side of the Fortran text running on the device, the prefix attributes(global) is used, which is an extension in the CUDA Fortran language. The global attribute means that the corresponding code runs on the device but is called from the host [**211**].

The 6th and 7th lines of code for the device are a replacement for the do loop in usual Fortran:

```
1  do i=1,n
2  a(i)=i
3  enddo
```

Since the test1 subroutine is executed on the GPU, it is executed in parallel by several threads on the GPU, each of which is identified by the built-in variable ThreadIdx (it is used as the index-of the array element).

Since the test1 subroutine is executed on the GPU, it is executed in parallel by several threads on the GPU, each of which is identified by the built-in variable ThreadIdx (it is used as the index-of the array element).

CUDA Fortran allows you to work in conjunction with other software. For example, it is possible to use OpenACC and CUDA Fortran together in one program; A program with OpenMP can use CUDA-managed memory [**211**].

Descriptions for CUDA Fortran are permanently available at the URL https://docs.nvidia.com/hpc-sdk/pdf/hpc*Vv*.pdf, where *Vv* are the all version digits of the SDK (e. g. *Vv*=233 for version 23.3 at the time of writing the review).

To conclude the brief information about CUDA Fortran, it should be noted the point of view of Nvidia employees, indicated in their famous book [212], that the use of CUDA Fortran will be aimed primarily at those programmers for whom it is important to ensure software portability, clarity and ease of maintainability of code when achieving acceptable performance. But the ease of writing code also has another important effect, increasing labor productivity (reducing program development time).

Naturally, other developers also have SDK-class software that can be used when working with Nvidia GPUs. For example, HPE Cray MPI [213] with CUDA support (this MPI correctly copies data from the device memory to the network card memory (and vice versa) by implicitly copying the data first to the host memory, and from there to the network card, or directly (bypassing the host memory) with support for GPUDirect RDMA [214], which is available in A100. MVAPICH2-GDR [215] can also interface with CUDA. AMD HIP tools, which can also work on Nvidia GPUs, were mentioned above. Such SDK tools that were not developed by Nvidia will not be discussed here, but they may be discussed in the sections of the review about new generation GPUs.

4.1.4. *A100 Performance Data*

Although the review is aimed at new generation GPUs, it seems necessary to provide data on the achieved performance of the A100 and its acceleration relative to the V100. Information on the performance of the A100, which also provides a comparison with the new generation GPUs, is discussed further in the relevant sections about these GPUs, and may not be presented here.

Peak performance was discussed earlier, data from benchmarks and applications are discussed here. It must be kept in mind that deliveries of the A100 GPU from Nvidia, which began in 2020, were accompanied by such an active emergence of new A100 models, including due to the simultaneously awakening competition with the new generation GPUs from AMD emerging at that time, that the authors of some publications, where the latest A100 were used, the parameters of the model used were not always clearly indicated.

It is useful here to give a short introduction about the efficient use of tensor cores for dense matrix multiplication and the use of mixed precision. For working with AI, this is enough natural; For ordinary HPC applications FP64, that run efficiently on the GPU, the use of DGEMM is also often assumed. For a discussion of how actual is hardware support for DGEMM for mainstream HPCs, and whether lower mixed precision can be used, see [90].

HPC often requires DGEMM to work with large square matrices, and the A100 tensor core performs hardware operations on matrices of fixed small sizes (see above in Section 4.1.1). But there are many tensor cores in A100, and multiplication of large matrices boil down to multiplication of small submatrices. And programmers can simply use cublasDgemm, and for a large matrix size (compared to hardware-supported in tensor cores) 16384×16384, [**216**] indicates achieving 19.4 TFLOPS close to the peak performance for the A100 tensor cores. In [**216**], power consumption was also studied, and it was found that an increase in power does not always correlate with an increase in performance.

It should be noted that in [**216**] a matrix size multiple of 2^k was used. This is consistent with Nvidia's guide to working with matrices [**217**] on tensor cores (focused on AI areas) — it notes that performance is higher with matrix sizes that are multiples of 128 bytes (for FP64). In [**217**] describes how GEMM in cuBLAS is implemented by partitioning the output matrix into fragments that are assigned to blocks of threads, and discusses the optimal trade-off between the size of the submatrices to be multiplied and the requirement to utilize all GPU hardware resources through maximum parallelization. But these different proposed variants of GEMM are more important for not very large input matrices, and the results in [**217**] are demonstrated for the low precision typical for AI.

For the graphical processor Nvidia Titan RTX, where DGEMM was emulated using tensor cores due to their lack of hardware support for the FP64 format [**218**], a plot of the performance of DGEMM from cuBLAS versus the dimensions of square matrices shows certain jumps in the achieved performance. And for the A100, such jumps in GEMM performance when changing matrix dimensions when working with FP16 are demonstrated in [**217**]. Obviously, a similar dependence of the performance of A100 in DGEMM for the most relevant large square matrices for HPC also has this property, but the author is not aware of any articles demonstrating this.

So to work effectively with HPC on a GPU, not only sophisticated programming of possibly initially naturally sequential codes is required, but knowledge of the likely performance behavior at the BLAS level is also useful. In general, one gets the feeling that the use of tensor cores of these Nvidia GPUs for traditional HPC tasks in the FP64 format with conventional dense matrices remains to be studied — GPU hardware has developed too quickly, with an initial orientation not at all towards HPC.

A number of studies are devoted to the use of tensor cores and simply from the point of view of the applicability of mixed precision. Thus, in [**219**] various issues of such arithmetic were studied on V100 and A100, including from the point of view of rounding modes.

Work on software for matrix multiplication using tensor cores and their effective use continues. In [**220**], the possibility of using multiword arithmetic was explored — when matrices are represented as the unevaluated sum of two or more lower-precision matrices, and the product of the matrices is calculated by multiplying their lower-precision constituents. This was done, for example, on tensor cores of A100 and V100 for the input matrices **A** and **B** in formula (1) with FP16 numbers, and the accumulative matrices with FP32 numbers. Rounding errors were studied and the use of algorithms that reduce rounding errors was proposed. The purpose of this is to improve the trade-off between performance and precision [**220**].

For AI, it is relevant to work with matrices that have a specific sparsity. Data on such work with tensor cores for V100, A100 and H100 are available in [**221**].

The following is some of the information about the performance of the A100, which the author considers most relevant, especially for HPC tasks. The more high performance of the A100 relative to the V100 is obvious and confirmed in many articles. This is primarily due to FP64 support in tensor cores, higher peak performance, larger L2 cache size, and higher memory bandwidth in the A100. But this does not mean that the gain will be obtained in any code (especially not optimized), when using any version of CUDA, with any task dimensions, and so on. Thus, in [**222**] on some codes given by Nvidia as examples of using CUDA [**223**], the V100 outperformed the A100.

Well, as a kind of starting integral estimate of the acceleration of the A100 relative to the V100, we will use the data from the SPEC ACCEL v1.2 benchmark (using 19 different applications) [**224**], where the maximum results obtained by Lenovo for the A100-PCIe-40GB relative to the V100S-PCIe-32GB are better than approximately 1.7 times. Another integral assessment of the performance of the A100 is provided by data from the SPEChpc 2021 benchmarks, for which modern supercomputers with GPUs are also used [**225**]. However, the official results of these benchmarks [**226**] (as of June 14, 2023) do not contain data for computing systems with the same number of A100 and V100 GPUs. Comparative data on the performance of servers with multiple A100 is presented below in Section 4.2.6 on H100 performance.

Microbenchmarks and low-level performance measurements, as close as possible to the A100 hardware. Several publications can be roughly classified into this group of performance tests. In [227] stydied the performance impact of using asynchronous copy at the microbenchmark level and in the higher-level modernized integral Rodinia benchmarks suite (for non-AI applications at the University of Virginia).

In [79], microbenchmarks were used to measure the latency and bandwidth (performance of executing instructions) on tensor cores (at PTX level). In [228], the performance of kernels for memory-bound iterative methods for solving systems of linear equations was studied on the V100 and A100.

In [229], a low-level synthetic benchmark was developed and applied to determine the performance and power consumption of the DGX A100 server with 8 A100 GPUs and the DGX-2 server with 16 V100 GPUs using dynamic frequency and voltage scaling. To measure performance, it was used the Mandelbrot floating-point benchmark [230], implemented on CUDA PTX, which gives the closest approximation to peak performance due to high data parallelization, virtually no execution delays due to memory access and branches. These servers use 64-core EPYC 7742 due to their support for PCIe-4.0, which was not available in Intel Xeon. At optimal for power efficiency configuration, the A100 achieved power efficiency of 51 GFLOPS/W and 91 GFLOPS/W for the FP64 format when operating without and using tensor cores, respectively [229].

To ensure high application scalability, communication between GPUs in a multi-GPU server and between cluster nodes of such servers is important. A study [231] focused on Deep Learning Recommendation Models (DLRMs—widely used by Internet companies to predict what consumers might like when multiple options are available) obtained data on the bandwidth of such communications using NCCL and MPI for systems containing A100 and V100.

Memory bandwidth benchmarks. Data on traditional performance benchmarks can start with bandwidth benchmarks — it very often limits performance and is important for other benchmarks discussed below. Here we must keep in mind that due to the different programming models used, different numerical results may be obtained for the same benchmark. A notable example of this is *BabelStream*[URL] [232], which is a modification of the widely used STREAM benchmark for CPUs (see, for example, data for ARM processors [5]).

BabelStream does not take into account the transfer time between the host and the device, and it has implementations in almost all programming models used for GPUs. Compared to the classic standard STREAM benchmark [**233**] BabelStream has added a kernel with the scalar product of vectors, and some other modifications have been made [**232**].

In [**234**, **235**], the obtained dandwidth data for all BabelStream kernels for different array sizes show a generally significant speedup of A100 relative to V100. For an 8.2 GB array, the V100 achieves 800 to 840 GB/s in these kernels, while the A100 achieves 1.33 to 1.4 TB/s. Although for small array sizes the A100 has lower bandwidth than the V100, for larger array sizes the A100 is about 1.7 times faster than the V100 for most kernels [**234**]. It should be borne in mind that these data were obtained shortly after the appearance of the A100 and obviously refer to the first model A100-40GB, and the CUDA 11 version was used.

Similar bandwidth data for all BabelStream kernels using different array sizes for the A100 and V100 are presented in [**173**].

In [**42**], data were obtained for the bandwidth (with the usual FP64 format) of all 5 BabelStream kernels: copy (a[i] = b[i]), mul (a[i] = b * c[i]), add (a[i] = b[i] + c[i]), triad (a[i] = b[i] + d * c[i]), dot sum (= sum + a[i] * b[i]) — and for each of them on 5 different programming models, for A100 (on a server with two 64-core EPYC 7H12 + 4 A100-40GB) and for V100 (on a server with two 20-core Xeon Gold 6230 + 4 V100 -32GB). One GPU was used in this benchmark. As a "unit of reference" that gives the maximum bandwidth of A100 and V100, the BabelStream results obtained with the CUDA version of this benchmark was taken.

Our analysis of data from [**42**] shows that in all BabelStream kernels using OpenMP and Kokkos on the A100 the "closing" to the CUDA values in percentage is more than on the V100 (with the exception of add and triad in Kokkos), and the achieved level of "close to the CUDA indicators" looks acceptable.

In many ways, BabelStream benchmark provide a convenient opportunity to compare the effectiveness of the implementation of different programming models of different GPUs. For example, in [**236**] a Fortran implementation of BabelStream was made using DO CONCURRENT tools from Fortran 2008 applicable to GPUs, which is in addition to other capabilities for working with GPUs in Fortran — using OpenMP, OpenACC or CUDA Fortran tools. Comparative performance data of different BabelStream implementations was obtained for A100-40GB and A100-80GB. Using NVHPC 22.7 for A100-80GB in C++ and Fortran, bandwidth of 1.7-1.8 TB/s was obtained in all 5 BabelStream kernels.

At the same time, the achieved bandwidth values on OpenMP and CUDA variants for C++ and Fortran practically coincided (the maximum difference is about one percent), with the exception of the dot kernel, where CUDA C++ was 13% slower than CUDA Fortran (after changing of the tuning parameter BabelStream, the C++ lag has decreased to one percent).

This review does not systematically examine the effectiveness of various GPU programming models, and these data were presented to illustrate the opinion of the authors [236] supported by the author of this review, about the lack of activity in the development of programming models using Fortran tools for heterogeneous architectures. The significantly higher results obtained in [236] for the A100 in BabelStream compared to [234, 235] are primarily associated with the use in [236] of a more modern A100 model (the A100-80GB has higher memory bandwidth; In [237] BabelStream triad kernel gave 1732 GB/s for A100-80GB versus 1399 GB/s for A100-40GB.

Achievable memory bandwidth data is no less important for AI than for HPC. In [231], a set of tests for DLRM tasks is proposed, which includes a memory bandwidth test specialized for DLRM, where it is evaluated when working with FP32 and FP16 formats. For the A100-40GB, a throughput of about 1.4 TB/s was achieved for FP32, for FP16 it was slightly lower. For V100 with FP32 more than 800 MB/s is achieved, and with FP16 — less than 800 MB/s [231].

Considering mixed-precision work in tensor cores, the corresponding memory bandwidth indicators are used to optimize the performance of the actively developed Ginkgo sparse linear algebra library portable to GPUs from different manufacturers [173]. The authors of [238] reports the performance of the A100 on a mixbench benchmark that uses kernels with mixed computational intensities (the ratio of floating point operations performed to bytes transferred) for different data formats and with different programming models [239], which gives and bandwidth estimates. This is actual mainly for AI and is not discussed here.

Performance of Linear Algebra Codes (BLAS). As with other GPU benchmarks, linear algebra tasks introduce a number of GPU-usable features that are different from traditional CPU use. This is mainly due to the frequent use of lower precision formats and support for operations with small matrices in tensor cores, which is primarily actual for AI. This led to BLAS with lower precision, as well as the development of batch linear algebra (solving many linear algebra tasks with small independent matrices; The batch size here is the number of matrices in it) [240].

Accordingly, linear algebra libraries specialized for GPUs have appeared (here we primarily mean not libraries from GPU development companies, but libraries focused on working with GPUs from different manufacturers) — and articles are appearing that study the performance of working with the tools of these libraries.

One thing to note here is the Ginkgo library for sparse linear algebra mentioned above; Data on the achieved performance on the A100 using it are available in [**173**, **234**, **235**, **237**, **238**]. For batch linear algebra, the well-known MAGMA library is applicable, the modules of which showed performance higher than the corresponding modules of the GPU manufacturers' libraries. Data on the performance of this library when running on the A100 are presented, for example, in [**234**] and [**241**]. For exascale computing, the libCEED library of the CEED Center (Center for Efficient Exascale Discretizations) of the US Department of Energy has appeared, where performance close to peak values is achieved on tensor cores of Nvidia GPUs [**242**], but the corresponding data in this work were obtained not for the A100, but for the V100.

It's natural to start analyzing performance for dense linear algebra problems with GEMM, since this is the only way to achieve maximum performance when working with A100 tensor cores. But due to the fundamental importance of GEMM and working with tensor cores, the general features of this for A100 have already been analyzed above at the beginning of Section 4.1.4 Publications that provide numerical estimates of the performance of GEMM with different data formats are reviewed here.

Benchmarks [**222**] reports the performance obtained for matrix multiplication of different data formats on the A100 and V100 for example CUDA kernels from Nvidia. There was no optimization goal here, small matrix dimensions atypical for traditional HPC problems were used, and this only illustrates the fact that you can use matrix multiplication code that will run faster on the V100 than on the A100.

Nvidia's tutorial on matrix multiplication for deep learning tasks [**217**] shows how GEMM's performance from FP16 on V100 increased dramatically when moving from cuBLAS v10 to cuBLAS v11, illustrating the rapid progress even in stable Nvidia software. This guide also shows typical performance dependences of the A100-SXM4-80GB for such GEMMs on matrix dimensions.

In [**231**] the benchmarks set for DLRM includes GEMM benchmark and uses GemmEx calls from cuBLAS. For V100 with FP32, this benchmark applies to working with FP32 vector cores, and for mixed precision with

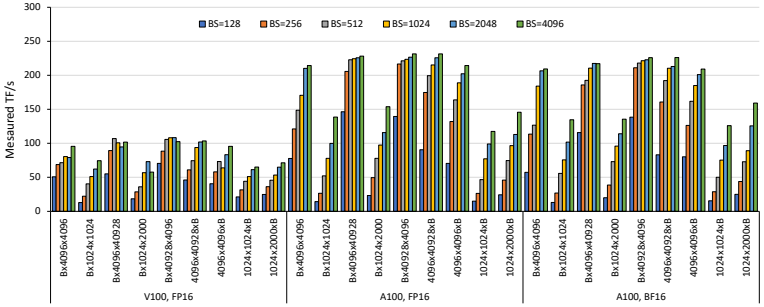


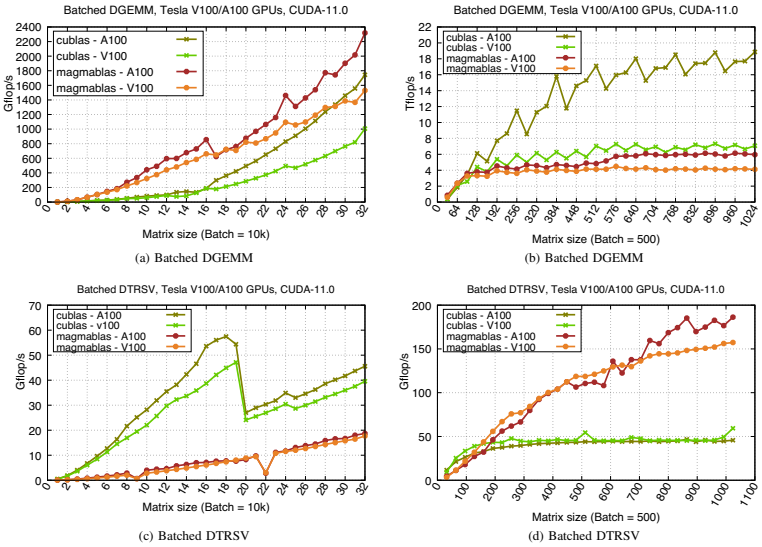
FIGURE 8. Performance of the GEMM batch implementation — with GemmEx calls from cuBLAS (Figure from [231])

FP16 — with tensor cores. For the A100-40GB, the benchmark with tensor cores additionally includes work with TF32 and BF16. But the achieved performance values given in [231] refer to the batch version of GEMM (see Figure 8; BS in this figure is the package size), which is due to the small (indicated in the figure) sizes of different matrices. This figure shows that the performance of the A100 with FP16/BF16 is several times greater than that of the V100 with FP16, which corresponds to the main focus of modern GPUs primarily on AI.

But the performance of batch variants of GEMM was also studied for the FP64 format. Data showing significant performance gains when running DGEMM in batch mode on a GPU are available in [242]. Batch variants of DGEMM for typical HPC tasks can be said to have not been used before — in the sense that relevant research has only begun to appear in the last ten years. Here it is worth noting the solution of the equation of hydrodynamics of compressible fluids with high-order finite elements [243] (see also [244, 245]).

In computational chemistry, the packaged DGEMM was used in the well-known CP2K software package for calculations using the quantum chemical DFT method with periodic boundary conditions in the basis of plane waves [246]. The effectiveness of batch matrix multiplication for calculating the matrix of pairwise distances used in molecular dynamics tasks was noted in [247].

The appearance of FP64 support by tensor cores in the A100 can significantly increase the relevance of batch DGEMM. But the use of these tools now largely relates to the development stage, figuring out where it can be used, which is also reflected in modern works. So, in [248] it was concluded that it is advisable to use batch DGEMM in the Nektar++ CFD application.



(a) Batched DGEMM (b) Batched DGEMM

(c) Batched DTRSV (d) Batched DTRSV

FIGURE 9. Performance of MAGMA and cuBLAS on DGEMM and DTRSV (Figure from [234])

In [249], calculations using the quantum Monte Carlo method were performed on nodes of the Vega supercomputer (at the Institute of Informatics, Slovenia), containing two CPUs (64-core EPYC 7H12) and four A100. Although the operation time of the batch GEMM was lower, in the Monte Carlo approach proposed in [249] using MPI parallelization of GEMM tasks, the overall performance of the entire simulation was much higher than that of the batch DGEMM. But in this work, tensor cores were not used.

Batch support for DGEMM is available in various libraries for the A100. In [234], the performance of batch DGEMM achieved on A100 was studied using MAGMA and cuBLAS (CUDA 11.0) tools. For small size workloads, MAGMA's batch DGEMM achieves 2.4/1.6 TFLOPS for A100/V100. This is 33/60% faster than cuBLAS, which is due to special optimization for small matrices in MAGMA. At the same time, the batch implementation of DGEMM in MAGMA did not use tensor cores (unlike cuBLAS). But on larger tasks, cuBLAS achieved 18/6 TFLOPS for the A100/V100 (for more details, see Figure 9, which also shows the performance of another BLAS routine, DTRSV); In general, on batch DGEMM, A100 is up to 1.5 times faster than V100 in MAGMA and up to 3 times in cuBLAS [234]. It can be seen that in the cuBLAS variant a good closing to peak tensor performance of the A100 for FP64 is achieved.

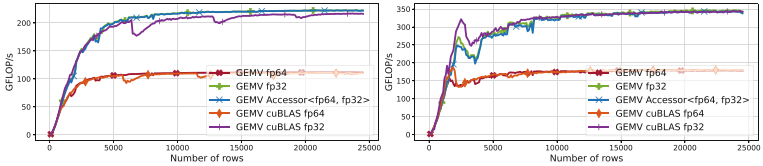


Figure 10. A100 performance for matrix vector multiplication (Figure from [238])

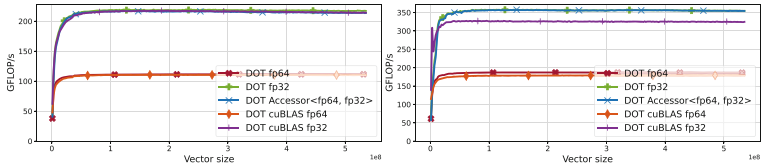


Figure 11. A100 performance with scalar product of vectors (Figure from [238])

In [234] also provides data on the performance of the A100 and V100 on MAGMA and cuBLAS implementations for batch BLAS DTRSV and LU and QR factorization (in LU and QR on the A100, MAGMA outperformed cuBLAS in all dimensions, and in DTRSV only in the middle ones).

To complete the discussion here of performance in batch linear algebra tasks on the A100, we should provide performance data for the QR factorization of dense matrices used in FP64 and FP32 formats [241]. The highest performance data, strongly superior to all alternatives, was achieved here using the MAGMA library. The asymptotic performance achieved (more than 2.6 TFLOPS for square FP64 matrices) is far from the theoretical peak values because the matrices for batch GEMM are not large enough.

From other publications regarding BLAS on A100, it is necessary to note the data [238] for matrix vector multiplication (GEMV kernel), which may be even more common (than GEMM) in HPC and for the scalar product of vectors (DOT kernel) — see Figure 10 and Figure 11. Here, the achieved performance for FP64 and FP32 using special memory access tools for different data formats of the Ginkgo library (marked as accessor in the figures) for A100 and V100 is compared with cuBLAS data. Here you can see how significantly the A100 outperforms the V100 in terms of performance across different BLAS implementations. Similar results were obtained in [238] for another matrix vector multiplication module from BLAS, TRSV.

As for working with sparse matrices, there are a number of publications for the A100 with data on the performance of sparse matrix vector multiplication (SpMV in BLAS), which is actual for various applications. [**173**, **234**, **235**] and [**238**] provide performance data on the A100 and V100 for SpMV in FP64 format. But the result here depends on the chosen sparse representation format, and on the specific matrices selected in the test, and, naturally, on the library used (cuSPARSE and Ginkgo were used here), and at the "macro level" it is less informative. Let us only note the maximum achieved performance of 220 GFLOPS and 135 GFLOPS on the A100 and V100, respectively [**173**], which is close to the upper limits was expected by article authors (230 GFLOPS and 140 GFLOPS, respectively—probably in roofline modeling).

Since choosing the optimal sparse matrix storage formats is also a non-trivial problem, machine learning was proposed to solve it on the A100 and V100 in [**250**].

The fast ourier transform (FFT). Important data on the performance of the A100 is provided by the achieved FFT performance, which is actual for various tasks, both HPC and AI. [**251**] demonstrates the performance of A100 using the developed SYCL FFT library compared to cuFFT. Although SYCL FFT is clearly inferior to cuFFT in terms of performance, the goal of the SYCL FFT developers is portability to GPUs from different manufacturers. At the same time, the lag of SYCL FFT is largely due to big overhead of the SYCL runtime (especially kernel dispatch), which is less important for large-scale problem.

Another FFT library aimed at working with GPUs from various developers is the open source VkFFT library, whose performance data on the A100-40GB is available in [**252**]. VkFFT supports double, single, and half precision data, and the performance achieved on the A100 is typically greater than with cuFFT [**252**].

In [**253**] developed the tcFFT library for one- and two-dimensional FFT using tensor cores. When running on FP16, tcFFT outperforms cuFFT (CUDA 11.0 was used) in 1D and 2D FFT on both the A100 and V100 (with the exception of low-dimensional 1D FFT). A significant superiority in performance in A100 relative to V100 for one- and two-dimensional FFT is shown; These data are illustrated in Figure 12 for a 2D FFT [**253**].

Computational chemistry:

(*A*) *Molecular dynamics.* Molecular dynamics problems began to act as classic problems, one of the first demonstrated examples of GPU

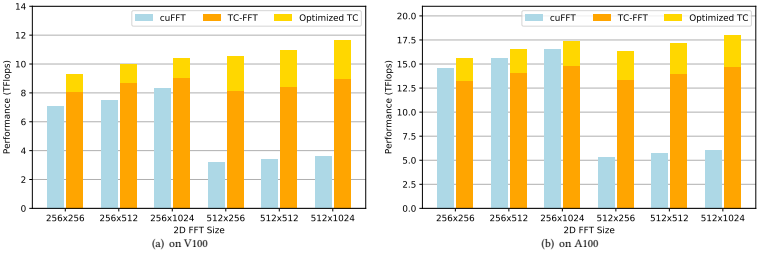


FIGURE 12. Performance of cuFFT and tcFFT on V100 and A100 (Figure from [253])

performance growth by their developers. Thus, in [190] an increase in performance in well-known molecular dynamics applications AMBER, GROMACS, NAMD and LAMMPS is indicated by 1.5-1.9 times on the A100 compared to the V100 (1.9 times applies to LAMMPS).

A detailed and in-depth study on the performance of LAMMPS in calculations of various molecular systems using different potentials and different GPUs, including the A100, is [254]. Here was used a cluster of multi-GPU servers (HPE/Cray EX on the Airforce Weather computing system, AFW HPC11). The A100-based nodes used one 64-core EPYC 7713 and four A100-40GB. LAMMPS uses Kokkos software to support code portability, and the paper aims to achieve the optimal choice of their parameters, and only the corresponding relative performance data is provided.

(B) *Molecular docking.* In [42], data were obtained on the performance of the miniBUDE mini-application [255] based on the BUDE (Bristol University Docking Engine) application targeted towards molecular docking for potential drug discovery tasks (prediction of the favored orientation of a ligand to protein at the form their stable complex. There, the change in the Gibbs free energy for the ligand-protein bond is considered extremely simple, empirically. BUDE was originally GPU-focused, and miniBUDE effectively ports to a wide variety of SDK for GPUs. The calculations here use FP32 and the A100 is about 30% faster than the V100.

Another molecular docking program, AutoDock, was upgraded to AutoDock-GPU in [256] to run on Nvidia GPUs. There, the A100 was found to achieve higher performance when using CUDA than when working with SYCL.

(C) *Quantum chemistry.* In [198] a radical modernization by the authors of the open source software package for computational chemistry (QUICK) is described, which makes it possible to carry out calculations

using quantum chemical methods HF and DFT (the most common in the world today) in a Gaussian basis sets on multi-GPU servers, including c A100. A feature of calculations of all modern quantum chemical methods in such a basis set is the need to calculate $O(N^4)$ integrals (N-dimension of the basis), which cannot be reduced to widespread mathematical methods.

These calculations (as well as the corresponding gradients) require major calculation time using the HF or DFT method. For parallelization, CUDA C and OpenMPI tools were used here. Each of the above integrals from contracted basis functions is calculated by one thread at the PTX level, which made it possible to achieve high performance for such a complex task. Parallelization in QUICK is possible from the level of a single GPU to a cluster of multi-GPU servers. All calculations are performed in the FP64 format required for quantum chemistry; Tensor cores are not used due to the lack of need for matrix multiplication in these methods.

The presented results of run times for various molecules containing from several tens to several hundred atoms in an Infiniband HDR cluster with DGX nodes (4 V100-SXM2 with two Xeon Gold 6248 per node) showed high parallel efficiency and almost linear performance scalability from 1 to 16 GPUs. On a DGX A100 server with 8 A100, parallel efficiency is slightly lower than with V100 (V100 has less SMs), but the calculation time is much lower [**198**]. QUICK is developing rapidly [**257**–**259**], and the achieved very high acceleration in DFT calculations relative to conventional dual-processor x86-64 servers makes it attractive for fairly large-scale calculations.

An alternative approach to providing highly efficient Gaussian DFT calculations using GPUs was proposed in [**260**] and was implemented on the A100. In [**260**] new algorithms were developed for calculating the Coulomb and exchange contributions to the matrix of the one-electron DFT Hamiltonian, and a program was created that generates CUDA codes (the kernels are different for different angular momenta). On molecular systems ranging in size from a few hundred to over a thousand atoms, they achieved fairly good performance scaling using up to 128 A100 GPUs on the Perlmutter supercomputer.

Another demonstration of the ability to achieve a high level of parallelization scaling in a cluster of multi-GPU servers with A100 on quantum chemistry problems is work [**261**], where calculations were performed at the Samsung Electronics supercomputer center (on the SSC-21 supercomputer, ranked 20th in the Top500), the nodes of which used HPE servers with two 32-core EPYC 7543/2.8 GHz, 1 TB memory and 8 A100-80GB (nodes connected via Infiniband HDR200).

Here, using the TDDFT method (extended in comparison with DFT, with time dependence for calculating exited states), using Gaussian basis functions, calculations of a protein molecule containing over 40 thousand atoms were performed using up to 256 A100. The parallelization program used OpenMPI, OpenMP and CUDA Fortran tools (from NVHPC SDK 22.3). One MPI rank per node was used, where the processors (they are connected to the A100 via PCIe) generate a number of OpenMP threads equal to the number of GPUs in the node, and NVLink is used to communicate between the A100. The resulting acceleration does not deviate much from linear scaling up to 256 A100; Even with 256 GPUs, the parallelization efficiency exceeds 80% [**261**].

Another work carried out within the framework of the ECP project [**262**] presents the calculation times for the exchange-correlation component of the one-electron Hamiltonian of the quantum chemical DFT method using Gaussian basis functions using the new NWChemEx software package (developed on the basis of the well-known NWChem software package). The corresponding part of the code was done on CUDA. In the calculation of a protein (ubiquitin, more than a thousand atoms), the dependence of the calculation time on the number of GPUs used in parallelization on the Perlmutter (with 4 A100 in each node) and Summit (with 6 V100 in each node) supercomputers was studied. To optimize the calculation, code was developed at the PTX level, and NCCL/NVSHMEM tools were used to optimize the use of distributed memory. The calculation time for all program components decreases almost linearly as the number of GPUs increases to 32, and the deteriorating deviations from linearity when using V100 are slightly larger. And these times themselves on one A100 are one and a half to two times less than on the V100.

Computational fluid dynamics (CFD). Data on the performance of CFD applications are as "typical" in reports from GPU manufacturers as data on molecular dynamics, and accordingly, many publications with such data for the A100 have already appeared.

Thus, in [**263**], the calculation using the OpenCFD-SCU program on the A100 required 3.036 seconds per calculation step compared to 4.702 seconds on the V100.

In [**264**] provides performance data on the well-known Nek5000/RS code using the specialized OCCA (Open Library Concurrent Compute Abstraction) library for different types of GPUs. The kernels on the A100 there support 2.1–2.2 TFLOPS (FP64) for the Poisson operator and 3.1–3.8 TFLOPS (FP64) for the advection operator (higher numbers in ranges refer

to higher dimensions). In the pressure preconditioner, the forward Poisson operator at coarser multigrid levels realizes 2.5–3.9 TFLOPS (FP32), and the Schwarz smoother supports 2.5–5.1 TFLOPS (FP32). The comparable values on the V100 are one and a half times lower.

Similar data on the higher performance of A100 relative to V100 by 1.55 times when running NekRS are available in [**265**].

In [**190**], an increase in productivity of 1.7 times is indicated on the A100 relative to the V100 of the famous NASA software package, FUN3D [**266**]. It is clear that this requires additional information about specifications of calculations.

FUN3D is a software package for solving CFD problems with an unstructured grid, originally written in Fortran, part of the source code of which was transferred to C++ CUDA in the form of kernels of the FLUDA library, and even a mini-application was developed. Calculations here are performed in FP64 format for most variables, and with mixed precision FP32/FP64 for linear algebra problems [**267**].

At the 2023 AIAA Science and Technology orum, there were a number of presentations that provided data on the A100's performance in various CFD tasks, including in comparison to the V100. In [**268**], it was found that three quarters of the total execution time of FUN3D using the A100-SXM-40GB on a 3.7 million point grid was used by memory-bound kernels. Achievable performance correlates with bandwidth, and calculation on the A100 by the NASA common research model (CRM) in transonic conditions using the Reynolds-averaged Navier–Stokes equations was 1.67 times faster than the V100-SXM-16GB calculation.

In [**269**], in an aerodynamic analysis for electric VTOL aircraft using a large eddy simulation program on a system with 8 A100, the calculation time using FP32 was 1.4 hours versus 2.9 hours on a system with 8 V100.

In [**270**], large eddy simulations of flow instability were performed on multi-GPU systems with 4 V100 and 8 A100. Performance vs. number of GPUs data for A100 and V100 systems shows that with the same number of GPUs (up to 4), A100 performance is several times higher when working with both FP64 and FP32, and good performance scaling up to 8 A100 is achieved.

Other CFD performance data for A100 and V100 servers at this forum is presented in [**271**], and in [**272**] performance data is obtained for systems with 4 A100 or V100 GPUs.

Report [273] presents data on the performance of solving CFD problems using the lattice Boltzmann method on the A100 and V100. The work [274] implements the lattice Boltzmann methods improved by the authors, compares the performance on the A100-40GB and V100 using FP32 and FP64 and, naturally, shows a significant increase in the performance of the A100 compared to the V100.

In [275] provides data on the performance achieved within the URANOS program when using the A100 or V100. URANOS is designed for compressible flow solver for high-fidelity modeling of compressible wall flows (solving the system of Navier-Stokes equations in a three-dimensional Cartesian system from low to high Mach and Reynolds numbers), is implemented in Fortran 90 and parallelized using OpenACC and MPI. The calculations were performed on two different Italian supercomputers with V100 in nodes, and on DGX-A100, where maximum performance was obtained. In all computing systems, maximum performance is obtained when using not the standard MPI implementation, but the supporting GPU from Nvidia, which uses direct data exchange between devices without accessing the host.

And in [276], for the tasks of accurate modeling of high-speed flows in various solution schemes, the efficiency of different Nvidia GPUs, including the A100 and V100, was compared using not only the FP64 format as the main one, but also FP32. Simulating supersonic jet noise (13 million cells, 400 000 iterations) on a single A100 using CUDA gave an run time of 34.5 hours. We calculated the ratios of various calculation times on A100 and V100 given in [276], and obtained a range of accelerations of A100 relative to V100 from 1.4 to 1.9 times.

Artificial intelligence. Regarding the A100's performance for AI problems, only two more important data sources (from the author's point of view) are considered here. Firstly, this is, of course, data on the performance of machine learning — MLPerf training benchmarks [277]; At the time of writing the review, the data was current for version 2.1 [278]. The more comparable "closed" section of these benchmarks is discussed below. Within it, possible data is divided into available cloud, available on-premise and preliminary. Table 14 shows locally available performance data for servers running the A100According to the rules, results can be provided for individual benchmarks from the general list. But in our review, the goal was maximum comparison, and the individual maximum performance values (obtained by different companies and achieved for individual of the benchmarks), which were not accompanied by many corresponding data for other benchmarks, are not shown in the table.

TABLE 14. Machine learning performance in MLPerf Training v2.1 benchmarks [278]. Data: Available on-premise

Task	Type and number of GPUs	Calculation time, minutes	Submitter
Image classification	A100-SXM-80GB, 4	54.956	ASUSTek[1]
	A100-PCIe-80GB, 8	30.756	ASUSTek[2]
	A100-SXM-80GB, 4	54.231	Dell[3]
Image segmentation (medical)	A100-SXM-80GB, 4	49.446	ASUSTek[1]
	A100-PCIe-80GB, 8	25.855	ASUSTek[2]
	A100-SXM-80GB, 4	47.253	Dell[3]
Object detection, light-weight	A100-SXM-80GB, 4	161.699	ASUSTek[1]
	A100-PCIe-80GB, 8	89.137	ASUSTek[2]
	A100-SXM-80GB, 4	222.199	Dell[3]
Object detection, heavy-weight	A100-SXM-80GB, 4	78.535	ASUSTek[1]
	A100-PCIe-80GB, 8	43.081	ASUSTek[2]
	A100-SXM-80GB, 4	83.712	Dell[3]
Speech recognition	A100-SXM-80GB, 4	59.989	ASUSTek[1]
	A100-PCIe-80GB, 8	32.534	ASUSTek[2]
	A100-SXM-80GB, 4	55.086	Dell[3]
Natural languages processing	A100-SXM-80GB, 4	33.431	ASUSTek[1]
	A100-PCIe-80GB, 8	24.186	ASUSTek[2]
	A100-SXM-80GB, 4	32.792	Dell[3]
Recommendation	A100-SXM-80GB, 4	3.147	ASUSTek[1]
	A100-SXM-80GB, 4	4.309	Dell[3]
Reinforcement Learning	A100-PCIe-80GB, 8	161.647	ASUSTek[2]

[1] with EPYC 7773X, GPU TDP 400 W;
[2] with 2×EPYC 7763, GPU TDP 300 W;
[3] with 2×EPYC 7763, GPU TDP 500 W;

The table contains only data whose submitter provided results for 7 benchmarks out of 8 available in MLPerf training 2.1. There are no V100 performance data for version 2.1, but the strong acceleration of A100 relative to V100 can be indirectly estimated from MLPerf training HPC 2.0 data [279]. The resulting benchmark performance may depend not only on the GPU characteristics itself, but also on the specific computing system and software used.

This is not discussed further here, but below, in Section 4.2.6, we present data on the performance of the A100 in the new versions of the MLPerf benchmarks — training v.3.0 and another set of benchmarks for machine learning inference, inference datacenter v.3.1. There, the performance of the A100 is compared with the H100 (this data became available after the practical completion of the review).

Another publication with important information on A100 performance metrics for various AI domains [231] is narrowly focused on DLRM. It has been noted that these models have atypical requirements compared to other types of deep learning models. In [231], using A100 and V100, various performance data were obtained, including on a cluster of GPU servers, which is actual for large, highly scalable data centers, and recommendations were made on possible improvements to DLRMs.

All of the above performance data naturally and unambiguously indicates the advantage of the A100 over the V100, including in performance, which should be most pronounced in HPC applications that require FP64 matrix multiplication and/or large memory capacity. The V100 lag is smaller in HPC applications with FP64 that do not use matrix multiplication in tensor cores: both GPUs have the same number of FP64 vector cores per one SM.

4.2. New Nvidia H100 GPUs

4.2.1. *H100 — microarchitectural implementations of Hopper*

The main metrics that characterize the H100 versus the A100 are shown in Table 10, and Figure 6 illustrates the overall SM structure in the H100, providing a framework for understanding the H100 architecture and scaling performance with the number of SMs. In addition to the H100 models discussed in this review, Nvidia began producing H800 GPUs, which also have Hopper architecture and are not subject to US sanctions against China due to the reduced performance level in the H800 to an acceptable level. These GPUs are not covered in this review; information about them is available, for example, in [185].

All information provided later in this section about the Hopper microarchitecture and its implementation in the H100 is based on a general description by Nvidia [78], and additionally, if more detailed information is needed, references are provided to relevant sources.

Thanks to the above discussion of the A100 and the Ampere architecture, the H100 analysis can only focus on the improvements in the H100 relative to the A100, since the overall build of the H100 and A100, as well as their SMs, are roughly the same. The general hierarchy of GPU construction from the SM level to the full GPU level is no different for the H100 and A100, and the terminology used does not change: from two SMs a TPC cluster is formed, and from a set of TPCs GPC clusters are formed, of which there are 8 on the GPU. But different H100 GPUs in this hierarchies differ in quantitative indicators.

Like the A100, the H100 has the highest available metrics supported by the Hopper architecture (these are specified for the GH100). In fact, Nvidia

TABLE 15. Macro-level characteristics of GH100 and H100 models

GPUs	GH100	H100-SXM5	H100-PCIe
Number of SMs	144	132	114
Number of TPCs	72	66	57
Number of GPCs	8	8	8
L2 cache	60 MB	50 MB	50 MB
Number of HBM stacks	6 (HBM2E or HBM3)	5 HBM3	5 HBM2E

offers two models (with different form factors) that implement the Hopper architecture, and their quantitative specifications in this hierarchy are shown in Table 15.

The GH100 has 9 TPCs in each GPC, but this is not required in real H100 models. The GH100, H100-SXM5 and H100-PCIe each use 10 memory controllers (512 bits each), giving a total memory width of 5120 bits (see Table 10). The most important thing is that H100 models with different form factors differ significantly in performance simply due to the different number of SMs, as well as due to different memory. This can be seen in more detail not in Table 15, but in Tables 10 and 12. Peak performance in the H100 can be calculated in the same way as was done above for the A100 — the number of SMs has simply increased in the H100, and every of 4 partitions of SMs in the H100 has twice the number of FP64, FP32 and INT32 vector devices than in A100 (see Figure 6).

Compared to the A100, in addition to the quantitative increase in the H100 in the number of available SMs and the number of vector cores contained in each SM, and at the general level of the H100 — the size of the L2 cache and other specifications, a number of very important improvements in the H100 hardware are closely related to CUDA and will be discussed further in the analysis of CUDA extensions for H100. This applies to both asynchronous execution and a new module, the Tensor Memory Accelerator (TMA), for efficiently transferring large blocks of data between global and shared memory (see Figure 6).

Finally, the H100 introduces or improves upon the A100 with a very large number of new hardware enhancements that are beyond the scope of this review in the narrow sense of its HPC focus and with limited consideration of AI. Even listing new such tools can take more than one indent — these are numerous improvements in MIG technology (especially actual for working with cloud technology), security tools, and so on. New instructions have been added to ISA, for example, for current genomics problems — this has a very important, but narrow meaning. A lot of what has been added relates to video and image processing and AI. Of the latter, we note only the appearance of hardware for transformer models used for NLP. As with the A100 above, interconnect hardware is covered in a separate section on H100 servers and computing systems.

Table 16. Nvidia GPUs Interconnect Specifications according to [78, 280]

GPUs	V100	A100	H100
Number of lines (differential pairs) per NVLink port	8	4	2
Single/bidirectional bandwidth of NVLink link, GB/s	25/50	25/50	25/50
Number of NVLink links per GPU	6	12	18
Total bandwidth of NVLink links	300	600	900
Total aggregate NVSwitch bandwidth, TB/s	2.4	4.8	7.2

4.2.2. *Computing systems with H100*

This section describes Nvidia's H100-based computing systems (from servers to supercomputers) using x86-64 server processors, as well as the H100 interconnects and Nvidia's Grace ARM processors that will be used in H100-based servers.

NVLink network for H100. To scale performance with the growing number of GPUs used in servers (which has become a current characteristic trend) and quickly exchange GPU data with the CPU and with other GPUs, interconnects are critical. From the author's point of view, here Nvidia managed to achieve a very striking breakthrough (see Table 16).

NVLink4 in H100, as a high-speed, low-latency interconnect with support of fault-tolerant features, provides a bandwidth of 900 GB/s — 1.5 times more than NVLink3 [78]. It is important that this bandwidth is ideally combined with other bandwithes of the new Nvidia superchips using ARM architecture, which will be discussed below.

NVLink4 uses two differential pairs in each direction (two times less than there were in the NVLink3 channel with the same bandwidth) to form a single channel with an effective bandwidth of 25 GB/s in each direction. Therefore, the H100 now has 18 NVLink4 channels versus 12 channels in the A100 [78].

But more importantly, NVLink4 now supports the NVLink network, allowing up to 256 H100s in different nodes (servers) to securely communicate with each other using a fat tree topology. The corresponding cluster has 32 nodes with 8 H100s each. The NVLink network has a network address space and uses address translation hardware in the H100, ensuring isolation of the network address space and the separate GPUs address spaces (previously on the NVLink network all GPUs share a common address space [78].

The NVSwitch switch initially provided the ability to combine the memory of several GPUs. Powered by the H100, the NVSwitch3 provides hardware-accelerated multicast collective operations, delivering up to two times the bandwidth and reduced latency of NCCL on the A100 for small block size collectives. This significantly reduces the load on SM during collective communications [78].

Using the NVLink network and NVSwitch3 allows you to create large-scale NVLink Switch System networks with very high levels of bandwidth. Nodes in such a network are connected through a second ("external" to the nodes) level of NVSwitches, which reside in switch modules outside the nodes and connect multiple nodes together. Connected nodes are capable of delivering 57.6 TB/s of all-to-all bandwidth [78].

This results in building H100-contained NUMA systems with very high memory scaling levels achievable, in line with the trends of today's large AI training models.

Servers and clusters with H100. H100 GPUs can be hosted on a server using appropriate Nvidia modules. Nvidia supplies HGX modules (roughly speaking, an analogue of a graphics processor board) that contain the H100 and can be used by other companies to create a server containing the H100. HGX H100 is a unit containing 4 or 8 H100. The 4 H100 HGX configuration has fully interconnected point-to-point connections, while the 8 H100 configuration provides full bandwidth between the H100s via the NVSwitch.

Another type of module with H100 from Nvidia is the H100 CNX, where in addition to the H100 there are Nvidia ConnectX-7 SmartNIC capabilities that provide 400 Gb/s throughput in the Infiniband NDR400 or Ethernet 400 variant [78]. It is clear that the HGX modules are primarily focused on AI, while the CNX modules are primarily focused on HPC.

The server configuration as it relates to H100 is determined by these modules. Therefore, here we will note only the DGX H100 servers that have already been supplied for use in supercomputers, focused primarily on AI (although various companies have already announced their servers with H100-PCIe and HGX, for example, Supermicro and Gigabyte).

The DGX H100 with 8 H100 system contains two independent data processing units (DPU) Nvidia Bluefild-3 and 8 ConnectX-7 adapters [78].The DGX H100, in addition to being almost completely ready to solve AI problems, is ideal for working with AI in cloud technology, support for which starts at the level of one H100.

In the DGX H100 SuperPOD cluster (minimum configuration — 32 DGX H100 nodes), compared to the SuperPOD A100, instead of two levels of Infiniband switches, one level of NVSwitch3 switches is used, and the maximum cable length from switch to switch is increased from 5 meters for DGX A100 to 8 meters for DGX H100.

The June 2023 version of the Top500 list has 5 supercomputers using H100-PCIe in x86-64 based servers. The highest performance among these supercomputers (14th place in the Top500) was achieved in the predecessor of the EOS supercomputer announced by Nvidia [**281**]. This cluster uses 128 DGX SuperPOD nodes. The DGX H100 servers run Ubuntu 22.04 and contains a 56-core Xeon Platinum 8480/2 GHz (Intel Sapphire Rapids, fourth generation Xeon Scalable CPU) and NVidia ConnectX-7 for Infiniband NDR.

Other Top500 supercomputers with H100 are in the second hundred of the Top500 list and beyond. They use servers from other companies and other server CPUs, including AMD EPYC. Of these supercomputers, it is worth noting Henri, which ranks 255th and also tops the similar June Green500 list. Henri uses Infiniband HDR and the nodes use 32-core Xeon Platinum 8362/2.8 GHz (Intel Ice Lake) processors. It should also be noted the expected Kestrel supercomputer [**282**].

The very fact that the DGX H100 is being used today in a supercomputer suggests its likely focus on AI and working with cloud technologies.

4.2.3. *Nvidia ARM hardware to work with H100*

The use of ARM processors in high-performance servers with the H100 GPU should be a vivid real-life illustration of ARM's success in competition with traditional x86-64 server processors, which were used in Nvidia servers with the H100 shipped at the time of writing this review. But the use of ARM in servers with Nvidia GPUs began earlier: multi-core ARM processors are already used in servers with the A100 (for example, from Gigabyte [**283**]; these servers are now supplied to Russia and Uruguay, and Gigabyte already offers servers with the H100).

In 2023, Nvidia began trial deliveries of modules containing ARM processors to work with the H100. Information on the architecture and performance of Nvidia's ARM CPU (Grace) for running the H100 is still very limited; Available benchmark and application level performance estimates for Nvidia's Grace were more about expectations.

Therefore, the discussion of Nvidia's ARM hardware (the Grace superchip module, which has begun trial deliveries, and the Grace Hopper superchip, where the Grace superchip is integrated with the H100) is very brief here.

TABLE 17. Main technical characteristics of the Grace superchip

Architecture	Neoverse V2 (ARM 9.0-A) with 4×SVE2(128 bit)
Number of ARM cores	144 (two Grace processors)
Peak performance	7.1 TFLOPS (FP64)
L1 cache	64 KB I-cache and 64 KB D-cache (per core)
L2 cache	1 MB (per core)
L3 cache	2×x117 MB
Memory type	LPDDR5X (with ECC)
Memory bandwidth	up to 1 TB/s
Memory capacity	240/480/960 GB
NVLink-C2C bandwidth (bidirectional)	900 GB/s
PCIe links	8×PCIe-v5 ×16 with splitting capability
Their total (full duplex bandwidth	1 TB/s
TDP	500 W (with memory)

Data from: [284, 285]

ARM Grace superchips. The Grace superchip is essentially an implementation of an SoC containing two ARM processors and memory, with PCIe support. As noted in [13], it was created specifically for supercomputers and HPC.

The Grace data analyzed below is based on [284], and additional references are provided only for information from other sources. The main technical characteristics of Grace are collected in Table 17.

The first thing to note is that Grace is based on the latest ARM cores for HPC, Neoverse V2 [286], which were announced in the fall of 2022. These are 64-bit ARM cores with OoO instruction execution, supporting SVE2 instructions with 128-bit vectors.

The Grace superchip consists of two 72-core Grace processors connected by an NVLink-C2C channel supporting memory coherence with a throughput of 900 GB/s, which eliminates interaction between sockets of conventional servers as a bottleneck [13]. At the same time, the peak performance (FP64) of the superchip, 7.1 TFLOPS, is not much lower than that of the A100 without the use of tensor cores.

It should be noted that in modern Neoverse V2 cores, aimed at CPUs with a large number of cores, the L1 I-cache is protected by parity checking, and the L1 D-cache is protected by ECC codes. The L2 D-cache of ARM Neoverse V2 cores is also protected by ECC codes and can have a capacity of 1 or 2 MB [286]; in Grace, Nvidia chose the 1 MB option. The L3 cache is naturally also protected by ECC codes; or other details of the Neoverse V2 microarchitecture, see [286].

An interesting and very important feature of the Grace superchip developed by Nvidia is the Scalable Coherency Fabric (SCF), which has

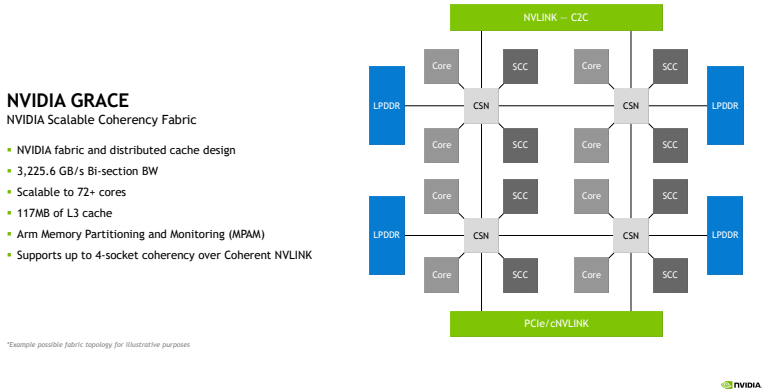


Figure 13. General construction of the Grace microarchitecture (Figure from [13])

a mesh structure over which the cores and L3 cache partitions (SCCs in Figure 13) are distributed, and ensures data flow between the cores, NVLink-C2C, memory and system I/O with a total half-bandwidth of more than 3.2 TB/s. This is illustrated in Figure 13. SCF is designed to scale beyond a single Grace CPU to form a super chip with 144 cores. Cores and SCCs are distributed across the mesh, and CSN (Cache Switch Nodes) act as the interface between the CPU cores, the cache, and the rest of the system [13].

The Grace superchip also has special tools that support partitioning of hardware resources, which can be effectively used when working in a cloud environment [284].

Nvidia developers explain their choice of 32-channel LPDDR5X memory technology as the "golden mean" between HBM2E and DDR5 in terms of capacity, bandwidth, cost and power consumption [13]. The maximum theoretical bandwidth of LPDDR5X in the Grace superchip (1 TB/s) is slightly higher than the bandwidth of NVLink-C2C (see Table 17).

It should also be noted that Grace supports four PCIe-v5 ×16 channels (in addition, for various tasks there are two more low-speed PCIe-v5 x2 channels) [13]. This immediately makes this superchip the leader among CPUs in terms of I/O performance.

But the TDP of the Grace superchip (this is a dual-processor SoC) is comparable to the TDP of modern GPUs; this is higher than the A100 and H100 PCIe (but lower than the H100 SXM). To work with the Grace superchip, air or liquid cooling can be used [284].

Nvidia in [284] also provides expected estimates of some of the Grace benchmarks. These relate to STREAM test data (up to 400

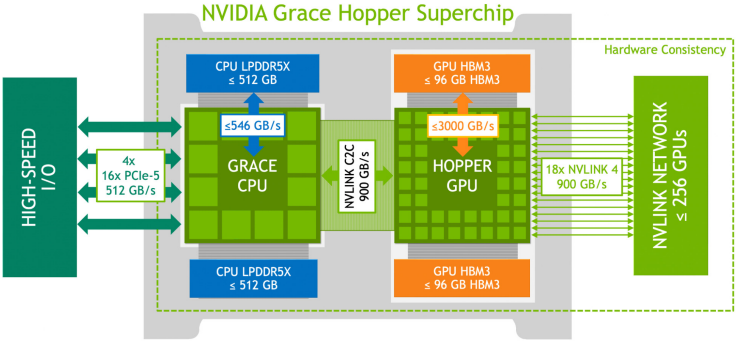


FIGURE 14. Grace Hopper with support for working with NVLink4 (Figure from [287])

GB/s for one Grace CPU and up to 800 GB/s for the entire superchip), SPECrate2017_int_base (370 and 740 for one Grace CPU and superchip, respectively), and significant speedup values in some applications (including areas CFD and weather forecasting) compared to a dual-processor server with AMD EPYC 7763 (Milan). Other initial data have appeared, but they are still completely insufficient.

Nvidia talks about a twofold increase in power efficiency compared to traditional CPUs used in data centers [284]. Plans to create supercomputers based on Grace were announced by the US Los Alamos National Laboratory and the Swiss National Computing Center [14].

4.2.4. *Grace Hopper superchips*

In addition to Grace, Nvidia also announced Grace Hopper, in which Grace (two 72-core processors) is integrated with the H100. The main basic indicators of this superchip are approximately the sum of the indicators of the H100 and Grace discussed above, and depend on which variants of each of these two integrated components is used in the Grace Hopper.

In version 1.01 of the document [287], on which further analysis of Grace Hopper superchips is based, it is indicated that the LPDDR5X size for Grace in the Grace Hopper superchip is up to 512 GB, size of HBM3 memory for H100 is up to 96 GB, and about the presence of two options Grace Hopper superchip — with support for working with NVLink4 (see Figure 14) and without it (then systems with such superchips are connected via Infiniband NDR). It is clear that the integration of two CPUs, H100 and memory gives a high TDP, 1000 W is specified in [287].

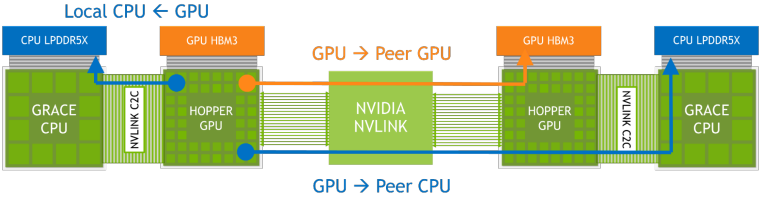


Figure 15. General microarchitecture of Grace Hopper (Figure from [287])

After completing work on this section of the review at the same URL [287], Nvidia posted new versions of this document (the latest version known to the author is 1.11), where the superchips in question are now called GH200 Grace Hopper. The most important changes there are associated with the emergence of new GH200 models, in which instead of HBM3, HBM3E with a capacity of 141 GB with a bandwidth of 4.8 TB/s will be used (versus 4 TB/s when working with HBM3). The main characteristics of the GH200 Grace Hopper architecture, of course, have not changed. But since due to the rapid development of GH200 and related computing systems in 2023, new modifications appeared in later versions of this document, and sometimes even the names used were modernized, the analysis of GH200 in this review was carried out in a limited scope and is further based on version 1.01 of the document.

As the interconnect between the Grace processor and the H100 uses NVLink-C2C, which provides high bandwidth and low latency (see Figure 15) [287]. NVLink-C2C provides memory coherence and hardware support for system-wide atomic operations, which improves the performance of synchronization primitives. Memory coherence allows developers to transfer only the data they need (rather than entire pages of memory) from Grace to H100 and back again, and naturally simplifies programming heterogeneous applications. The performance of non-local memory accesses also improves (for example, when CPU and H100 threads access memory located on another device) [287].

A key feature of the GH200 superchip is the use of extended GPU memory (EGM). Each GPU from a multi-node computing system with an NVLink network based on GH200 has access to both the LPDDR5X memory of all Grace processors and the HBM memory of all GPUs, i.e. The total memory capacity for the GPU is dramatically expanded. A unified

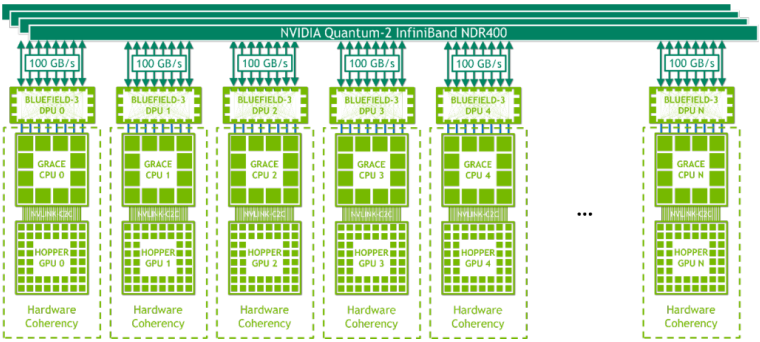


FIGURE 16. General construction of interconnections GH200 with NVLink support (Figure from [287])

memory is formed with common page tables, a common virtual address space. The speed of GPU access to local memory is set by NVLink-C2C, and to remote memory by NVlink4 [287] (see Figure 14 and Figure 15).

Servers with GH200 are already offered, for example, by Gigabyte [288]. Nvidia offered computing systems with the GH200 — the MGX GH200 and DGX GH200 platforms. The MGX GH200 modular platform (there are options with HBM3 or HBM3e) is as similar as possible to a server with one GH200 superchip, without support for the NVLink switching system. Nodes with MGX GH200 can form traditional clusters, where Infiniband or Ethernet can be used for communication, working through the DPU.

DGX GH200 is a supercomputer for AI; there, 256 superchips are combined using the NVLink interconnect, giving a memory address space of up to 144 TB. A summary of these Nvidia AI-focused platforms is provided in version 1.11 of the document [287].

From the point of view of the possibility of building the most powerful supercomputers, the basis for ultra-high levels of performance scaling is provided by the use of NVSwitch3 with NVLink4 channels in conjunction with the GH200, since this provides Nvidia's original two-level scaling. Its capabilities are illustrated in Figure 16, which relates to working with the GH200 variant with NVLink support [287].

At the first level, it's possible to combine a set of GH200 superchips into a single domain, connected through the NVSwitch3 switching system (with a total bandwidth of all GPU threads in the domain up to 900 GB/s per superchip). The use of these interconnects makes it possible to form

a common memory of the entire domain, including both the LPDDR5X memory of the Grace processors and the HBM3 memory of the H100 [287]. In late 2023, Nvidia announced another single-rack compute system (GH200 NLV32) — with a much smaller domain than the top-end DGX GH200 configuration. Already at this level a very high level of performance is achieved; Nvidia also calls this system a supercomputer (see, for example, [289]).

At the second level, clustering of domains is possible using the most modern, widely used interconnects. For this, PCIe-v5 channels supported in Grace are used. Figure 16 shows a possible variant of such formation of a cluster of domains using the Bluefild-3 DPU. For communication with Grace, this DPU has 32 PCIe-v5 lanes, which has a total bidirectional bandwidth of 256 GB/s (this DPU has another 32 GB of own memory). And to form a cluster of domains, the DPU has 1 or 2 additional ports with speeds up to 400 Gbps (they can work with Infiniband NDR400 or Ethernet 400) [290], as shown in Figure 16. The use of DPU removes the Grace need to manage transferring data over the cluster interconnect.

It is clear that scaling on such a system will be a difficult task even for AI. But the author does not know, even through the announcement, about the possibility of such an "option" and the level of scaling of systems with GPUs from other companies. But in general, it is clear that Nvidia with the GH200 is focused on delivering more work-ready computing systems.

4.2.5. *CUDA Extensions for H100*

Everything stated below in this section is based on the description of the H100 hardware [78]. But first, it is advisable here to point out the general structure of the CUDA platform in its part related to compilers, since over time new programming languages appear that can also be used to obtain kernels executed on Nvidia GPUs. At the same time, the CUDA platform provides different compilers with a unified software stack, and creating a compiler for a new programming language basically comes down to writing only the front end part of the compiler.

These compilers are based on LLVM (in [78] the use of its 7th version is indicated). In addition to the capabilities of the C++ and Fortran ISO standards used for working on GPUs, as well as their CUDA extensions, back in 2021 in CUDA 11.4 Nvidia introduced CUDA-Python [291], and there are also developments for Julia and other programming languages.

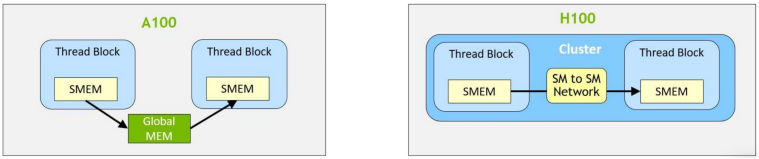


FIGURE 17. Working with DSMEM in A100 and H100 (Figure from [78])

The corresponding front-end components of these compilers generate an intermediate representation (IR) called NVVM IR [292], which is based on the famous LLVM IR. Next, using the libNVVM library, a PTX code is generated, from the virtual ISA of which a kernel is obtained that runs on the GPU [78].

The most important thing for achieving high performance of GPGPU is a provision of data locality, which gives high-speed access to nearby memory levels, and asynchronous execution, when the calculations themselves are performed independently (simultaneously) with data transfer.

Previously (and for the A100), the CUDA programming model included blocks of threads, from which a grid of threads was formed. IIn H100, the number of SMs has increased significantly and another level has been added to the overall thread hierarchy, a thread block cluster, which includes a set of thread blocks spanning multiple SMs. All this is reflected in the H100 hardware implementation with a new larger level of memory localization, since clusters of thread blocks in the H100 work simultaneously on SMs inside the GPC, ensuring fast data exchange between threads in the cluster.

Cluster threads can directly access the shared memory of other SMs using load, store, and atomic operations (which cannot be partially executed). For this purpose, DSMEM (Distributed Shared Memory) is formed. Figure 17 shows how data is exchanged between thread blocks in A100 and in H100 with DSMEM. A thread block cluster has access to more shared memory capacity than a single thread block—the virtual address space of "shared" memory. Compared to using global memory, DSMEM speeds up data exchange between blocks of threads by approximately 7 times. [78].

On the other hand, from the author's point of view, the appearance of a of thread block cluster on a new GPU model is a clear demonstration that widely used programming models for GPUs, which require the highest possible performance, automatically become poorly portable to other GPU models.

Other notable extensions in CUDA for H100 relate to asynchronous execution, allowing for overlapping (simultaneous execution) of data movement, computation, and synchronization [78]. To achieve this, the H100 has a new asynchronous memory copy module TMA and a new asynchronous transaction barrier. TMA can transfer large (up to the shared memory capacity) blocks of data and multidimensional tensors from global memory to shared memory and back.

The TMA operation is asynchronous and uses shared memory-based asynchronous barriers like the A100. In addition, one thread in a warp can be selected to perform an asynchronous operation, and then multiple threads can wait for the data transfer to complete using a barrier. TMA frees threads to do other independent work, since after a thread creates a so-called copy handle before running TMA, TMA itself performs all other functions as part of the H100 hardware.

Asynchronous barriers first appeared in the A100, and the H100 introduced a new primitive for asynchronous memory copies, the asynchronous transaction barrier. The write command to shared memory transfers not only the data, but also the number of transactions. The asynchronous transaction barrier blocks threads on a wait command until all producer threads have completed the "Arrive-On" operation and the sum of all transaction counters has reached the expected value. See [78] for more details. These capabilities are naturally used to work with thread block clusters.

4.2.6. *H100 Initial Performance Data*

Data on the H100's performance in benchmarks and HPC applications at the time of the June 2023 Top500 list was virtually non-existent, except for the starting data in the Hot Chips 34 report [293]. We will also note the available (as of June 14, 2023) data from the tiny suite of the SPEChpc 2021 benchmark [226], where results for the Lenovo ThinkSystem SR655 V3 server are presented with one and two H100-PCIe-80GB (with EPYC 9654P). We compared these data with Lenovo's SPEChpc 2021 results for one and two A100-PCIe-80GB on another ThinkSystem SR670 V2 server (with Xeon Platinum 8380). For the base variant of the test (the peak variant for the A100 was not carried out), the resulting performance on one H100 was 1.33 times greater than that of the A100, and on two H100s the same acceleration was 1.51 times. But should be kept in mind that for parallelization, OpenACC tools (plus MPI) were used here.

Other interesting data on the performance of H100 were obtained for the well-known molecular dynamics application Amber version 22 [**294**]. There, for typical test molecular systems for this application, calculations were carried out that gave performance estimates (the number of simulated nanoseconds per day of calculation) on different graphics processors. The relative performance calculated from these values shows the acceleration of H100 relative to A100 by 1.11-1.28 times for dihydrofolate reductase (23 558 atoms), 1.27 times for nucleosomes (25 095 atoms), 1.42-1.43 times for a protein of 90 906 atoms, 1.40 times for cellulose with a large number of macromolecules (408 609 atoms), 1.35 times for a satellite of the tobacco mosaic virus (1 067 095 atoms).

In calculations of smaller molecular systems, the accelerations obtained on H100 were also and greater than those indicated above, but we do not present these data here. But it is useful to note that for all the molecular systems calculated in [**294**], the performance of the Nvidia RTX 3090 is close to the A100, and the Nvidia RTX 4090 is close to the H100.

There is also other data on Amber22 performance on the H100 compared to performance on the A100, but these are discussed below in Section 5.3.2, with comparison to performance on the MI250.

But first of all, new generation GPUs are usually focused on solving AI problems, so the initial performance data appears here. For the H100, first became available Nvidia's preliminary results for one of the classic AI (machine learning) benchmarks suites, MLCommon — MLPerf Training in version 2.1 [**278**].

The calculation times presented in [**278**] for H100 (in the "closed"/"preliminary" section of MLPerf Training) were obtained on the DGX H100 system, containing, in addition to 8 H100, two Xeon processors (56 cores) and running the Ubuntu 20 distribution. Data analysis for the DGX A100 system with 8×A100-SXM-80GB in benchmarks that give comparable results to the H100 shows that the calculation time on the H100 is about two times less than on the A100, with the exception of the NLP benchmark, where the H100 is 3.8 times faster.

At the end of June 2023, data on the performance of H100 and A100 appeared on the new version of MLPerf Training v3.0 [**295**], and in September — on the new version of MLPerf inference datacenter v.3.1 [**298**], which are illustrated in Table 18 and Table 19.

Table 18. Performance data for H100 and GH200 in MLPerf inference datacenter v. 3.1 benchmarks (all data in the category "closed"/"available"); data have been selected for maximum comparability for 99% accuracy and rounded to 2 significant digits

GPUs	1	2	4	8
Image Classification/ResNet				
H100-PCIe-80GB	47(55)[1]	106(115)[2]	206(200)[2]	368(443)[1]
H100-SXM-80GB	73(89)[3]		312(354)[3]	584(704)[4]
GH200-96GB	77(93)			
A100-PCIe-80GB			147(158)[5]	
A100-SXM-80GB				305(340)[6]; 290(326)[7]
NLP/BERT				
H100-PCIe-80GB	4.6(5.7)[1]	9.1(12)[2]	18(23)[2]	35(46)[1]
H100-SXM-80GB	7.3(8.8)[3]		29(36)[3]	56(70)[4]
GH200-96GB	7.7(10)			
A100-PCIe-80GB			12(13)[5]	
A100-SXM-80GB				25(28)[6]; 25(28)[7]

The performance data (number of querues in the server scenario and number of samples - in parentheses - in the offline scenario) is **reported per millisecond instead than per second**.

All calculations were carried out using TensorRT 9.0.0 and CUDA 12.2.

[1] Data from Nvidia on Gigabyte G482-Z54, with 2×EPYC 7742;
[2] Data from Dell on Dell PowerEdge R760x with 2×Xeon Platinum 8480+
[3] Data from Dell on Dell PowerEdge XE9640 with 2×Xeon Platinum 8468
[4] Data from Nvidia on DGX H100 with 2×Xeon Platinum 8480C
[5] Data from Dell on Dell PowerEdge R750x with 2×Xeon Gold 6338
[6] Data from HPE on HPE ProLiant XL675d Gen10 Plus with 2×EPYC 7763
[7] Data from Oracle on Oracle BM.GPU.A100-v2.8 with 2×EPYC 7J13

Data for these tables were selected to provide the greatest possible comparison of performance, but since they were actually obtained after the completion of data selection for review, they are not analyzed in detail here.

Table 18 shows selected data from the MLPerf inference datacenter version 3.1 benchmarks suite. The data in this table shows the H100's clear performance gains over the A100, the H100-SXM's more subtle performance gains over the H100-PCIe, and the 96GB GH200's increased performance over the H100-SXM (and in more so over the H100-PCIe). This demonstrates the increase in performance for AI area when working with a single H100 GPU in the Grace-integrated GH200 variant with 96 GB HBM-memory. In addition, these data show good scalability at the number of GPUs used in the server increases from 1 to 8.

TABLE 19. Machine learning performance in MLPerf Training v3.0 benchmarks [295] in minutes

GPUs	2	4	8
Image classification			
A100-PCIe		58[1]; 61[2]	32[3]
A100-SXM		54[4]	27[5]
H100-PCIe	82[6]	45[2]; 39[7]	20[8]; 21[9]
H100-SXM		27[10]; 26[11]	13[12]; 13[13]
Image segmentation (medical)			
A100-PCIe		47[1]; 48[2]	
A100-SXM		47[4]	23[5]
H100-PCIe	67[6]	32[2]; 31[7]	19[8]; 18[9]
H100-SXM		22[10]; 22[11]	12[12]; 12[13]
Object detection, light-weight			
A100-PCIe		171[1]; 176[2]	
A100-SXM		222[4]	79[5]
H100-PCIe		114[2]; 107[7]	54[8]; 56[9]
H100-SXM		72[10]; 72[11]	37[12]; 37[13]
Object detection, heavy-weight			
A100-PCIe		86[1]; 81[2]	47[3]
A100-SXM		83[4]	38[5]
H100-PCIe		62[2]; 55[7]	28[8]; 28[9]
H100-SXM		40[10]	19[12]; 20[13]
Speech recognition			
A100-PCIe		63[1]; 64[2]	
A100-SXM		55[4]	29[5]
H100-PCIe	77[6]	51[2]; 45[7]	23[8]; 28[9]
H100-SXM		27[10]; 27[11]	19[12]; 19[13]
Natural languages processing			
A100-PCIe		45[1]; 51[2]	
A100-SXM		32[4]	15
H100-PCIe	43[6]		10[8]; 10[9]
H100-SXM		11[10]; 11[11]	5.4[12]; 5.4[13]
Recommendation (DLRM)			
A100-SXM			8.4[5]
H100-SXM		8.8[10]	4.3[14]; 4.3[12]

[1] Data on ESC4000-E11 with 2×Xeon Platinum 8462Y+
[2] Data on R750x with 2×Xeon Gold 6338
[3] Data on ThinkSystem SR670 V2 Server with 2×Xeon Platinum 8360Y
[4] Data on XE8545 with 2×EPYC 7763
[5] Data on XE9680 with 2×Xeon Platinum 8480+
[6] Data on D54Q-2U with 2×Xeon Gold 6430
[7] Data on R760x with 2×Xeon Platinum 8480+
[8] Data on ESC8000A-E12 with 2×EPYC 9654
[9] Data on AS-4125GS-TNRT with 2×EPYC 9554
[10] Data on SYS-421GU-TNX with 2×Xeon Platinum 8460H
[11] Data on XE8640 with 2×Xeon Platinum 8468
[12] Data on XE9680 with 2×Xeon Platinum 8470
[13] Data on AS-8125GS-TNHR with 2×EPYC 9634
[14] Data on G593-SD0 with 2×Xeon Platinum 8480+

All data in the table refers to "available on-premise" and was obtained on a GPUs with a memory capacity of 80 GB and **rounded to two significant digits**. For information about the AI models used in the benchmarks and the underlying datasets used, see [295].

Table 19 shows a sample of approximately one-sixth of the MLPerf Training v3.0 benchmark data table presented in [**295**], where are, for example, also results for a larger number of GPUs used. Such a large amount of data presented in this table allows to compare the performance of the H100-SXM, H100-PCIe and A100-SXM/PCIe with different numbers of GPUs used in the server, using different software and different hosts.

The data in this table shows that possible reasonable variations in the host hardware (of course, CPUs containing dozens of cores were used here) and software chosen by server manufacturers for these benchmarks do not have a strong impact on the achieved performance and are therefore not analyzed here.

The data in this table in almost all benchmarks shows good scaling with the number of GPUs in the server up to 8. As a zero approximation for comparing performance, we can assume that the dependence of performance on the number of GPUs is linear, although this is not always the case — for example, when moving from 4 to 8 H100-PCIe speedup in the medical image segmentation benchmark was only 70 percent.

The table data shows that noticeably higher performance is achieved when working with the H100-SXM compared to working with the H100-PCIe (for example, one and a half times more for 4 GPUs in image classification and image segmentation benchmarks); or that the performance of the H100-SXM is twice as high as that of the A100 on the same benchmarks with the same number of GPUs; or that H100-PCIe is one and a half times faster than A100 under the same conditions, and so on. It also happens that scaling with the number of GPUs is rather insufficiently high for AI (for example, when moving from 4 to 8 H100-PCIe in the medical image segmentation benchmark), but such an analysis, for obvious reasons, is not carried out here.

4.3. Nvidia GPUs Summary

Despite the emergence of alternative Nvidia GPUs from AMD and Intel, there are no signs of weakening in the position of Nvidia GPUs in hardware and software terms — their use in quantitative terms will continue to develop rapidly. Software tools for Nvidia GPUs are the most widely available, and new software tools are usually released for them earlier than for competitive GPUs. New generation GPU architectures from Nvidia provide a high level of compatibility with earlier GPU models. Accordingly, the portability of the software to newer models is significantly higher than that achieved when switching to work with competitor GPUs.

While Nvidia's GPU market share could theoretically fall in the global market due to the introduction of AMD and Intel GPUs, the global increase in AI efforts will boost the adoption of Nvidia GPUs. The possible performance advantages of Intel and AMD GPUs need to be proven in real applications. From an HPC perspective, it's worth paying attention to Nvidia's increasing focus mainly on AI area.

5. Next generation GPUs from AMD

The new generation GPUs in this review include GPUs that appeared after the basic Nvidia V100 and A100 GPUs. AMD has come a long way in recent years not only in the field of successful competition with Intel x86 processors (AMD EPYC is pushing aside Intel Xeon not only in the market of traditional servers, but also in the market of servers with GPUs). The introduction of the MI100 [**296**] at the end of 2020, and the MI200 family the following year [**297**, **299**] showed that competition with Nvidia GPUs is starting. It was the GPU MI100 and MI200 that became the first representatives of GPUs produced, classified in this review as a new generation.

The MI100 was sometimes seen as a precursor to AMD GPUs in upcoming supercomputers (the Spock cluster with MI100 in nodes is a predecessor to Frontier [**254**]; the MI100 began to be used in the nodes of LUMI, which took third place in the June Top500 list, when its nodes already began using the MI250X [**42**]). The MI100 architecture has already been discussed in publications — see, for example, the review [**21**]. By analogy with the V100, here we will limit ourselves only to the summary technical characteristics of the MI100 (see Table 20) and Table 21, but we will analyze data on the achieved performance of this GPU, including in comparison with other GPUs considered in the review.

The relevance of the analysis of GPUs from the MI200 family is increased by the fact that they have now been used to build not only the No. 1 supercomputer in the Top500, which for the first time in the world crossed the 1 EFLOPS barrier (Frontier), and the third in the Top500 (LUMI) list, which belongs to the European Union (since LUMI is installed in Finland — this also demonstrates significant European success in the supercomputing field): MI250X GPUs are used in 2% of supercomputers from the Top500, which also reflects another success of the manufacturer, HPE/Cray, in whose hardware the MI250X was placed.

Table 20. Specifications of modern AMD and Nvidia GPUs

GPUs	MI100	MI210	MI250	MI250X	A100 with PCIe	A100 with SXM4	H100 with PCIe	H100 with SXM5
TSMC technology, nm	7	6			7		4	
Number of active cores (stream processors)[1]	7680	6656	13312	14080	6912	6912	14592	16896
Number of active CUs[2]	120	104	2 × 104	2 × 110	108	108	114	132
GPU base/boost clock (MHz)	1000/1502	1000/1700			765/1410 or 1065/1410	1095/1410 or 1275/1410	1095/17554	1590/19804
Peak performance: FP64 (TFLOPS)	11.5	22.6	45.3	47.9	9.7	9.7	25.6	33.5
FP64 with tensor cores (TFLOPS)	No	45.3	90.5	95.7	19.5	19.5	51.2	66.9
FP32 (TFLOPS)	23.1	22.6	45.3	47.9	19.5	19.5	51.2	66.9
FP32 with tensor cores (TFLOPS)	46.1	45.3	90.5	95.7	No support: lower precision inputs			
FP16/BF16 (TFLOPS)	184.6/ 92.3	181	362.1	383	78/312[3]		205[3]	268[3]
INT8 (TOPS)	184.6	181	362.1	383	624		1513	1978.9

[1] Low-level SIMD cores with support for multiply-and-add commands are also called CUDA processors by Nvidia (matrix/tensor cores are not taken into account here);
[2] for Nvidia — streaming multiprocessors (SM);
[3] when using sparsity — twice as high; Performance with FP16, BF16 and INT8 relates to tensor kernels.

Performance with FP16, BF16 and INT8 refers to tensor cores.

TABLE 21. Specifications of modern AMD and Nvidia GPUs (continuation)

GPUs	MI100	MI210	MI250	MI250X	A100 with PCIe	A100 with SXM4	H100 with PCIe	H100 with SXM5
GPU memory type	HBM2	HBM2E						HBM3
Memory bus, bit	4096		8192		5120			
Memory capacity, GB	32	64	2 × 64		40 or 80		80[5]	
Memory clock, MHz	1200	1600[6]			1215 or 1512	1215 or 1593	1593	1313
Its bandwidth, GB/s	1229	1638	3277 (2 × 1638.4)		1555 or 1935	1555 or 2039	2039	1681
Interconnect of GPU with GPU or with CPU	Infinity Fabric 2.0	Infinity Fabric 3.0			with CPU: PCIe v4	NVLink-3.0	with CPU: PCIe v5	NVLink-4.0
Its bandwidth, GB/s	3 × 92	3 × 100	6 × 100		with CPU: 64	600	with CPU: 128	900
L1 cache, KB	16 on CU				192 on SM		256 on SM	
L2 cache, MB	8	16			40 or 80	40	50	
TDP, W[4]	; 300 PSU: 700	; 300 PSU: 700	500; 560 PSU: 900	500; 560 PSU: 900	250 or 300 PSU: 700	400 PSU:800	350 PSU: 750	700 PSU: 1100

[4] There is a model with HBM3/96 GB and a different GPU clock.

[5] for tensor operations with reduced (less than FP32) precision, the boost clock is lower;

[6] TDP: the number after the semicolon for AMD is peak, PSU: suggested power for Power Supply Unit;

This table data are taken from the database [185] and from manufacturers websites.

5.1. GPUs AMD MI200

5.1.1. *Microarchitecture and technical characteristics of different MI200 models*

The MI200 family includes three different models — MI210 [**300**], MI250 [**301**] and MI250X [**302**], of which the top model MI250X appeared and was the first to be supplied to the Frontier supercomputer [**303**], the leader of the Top500 list.

As AMD transitioned from producing its traditional Radeon Instinct graphics processors to next-generation GPUs, the architecture also changed. If earlier AMD graphics processors used the GCN (Graphics Core Next) architecture, then CDNA (Compute DNA) was developed for MI100 (Compute DNA) [**304**], and in MI200 it was upgraded to the second version CDNA 2 [**305**]. When these architectures change, naturally, a certain continuity is maintained, primarily in the computing units, the improvement of which largely occurs in technological and quantitative indicators.

For the analogue of the basic computing unit SM in Nvidia GPUs for AMD GPUs, the term CU is used (see Table 1), which is also used in BR100. Sufficiently detailed information on the construction of the CU and its main blocks for CDNA is available in [**304**]. The CU in CDNA 2 contains, in particular, a dispatcher for working with threads, files of ordinary and vector registers, cores (including matrix ones, see below), L1 cache and LDS (Local Data Share) memory [**305**]. LDS shared memory (often also called LSM, Local Share Memory) is used by threads inside a warp (wavefront in AMD terminology, see Table 1) [**305**].

The L1 cache capacities for the MI100 and various MI200 models, as well as other important indicators of these AMD GPUs, in comparison with the A100 and H100 are shown in Table 20 and Table 21. Taking into account the delays in PVC supplies and the possible freeze in production of the BR100, the greatest interest in this review and for comparison purposes GPUs in general currently represent the GPUs from this table.

But first of all, we should point out the technological indicators. MI100 began to be produced using TSMC 7 nm technology (like the A100), and MI200 — using 6 nm technology (only in the latest H100 that Nvidia introduced, used 4 nm technology). The MI100 contains 25.6 billion transistors with die size of 750 mm^2 — and the MI250X already has 58.2 billion (with a smaller die size 724 mm^2) [**185**]. These indicators are important, among other things, because they are associated with possible cost indicators and TDP values.

These data show the similarity of such indicators for MI100 and V100 (see also Table 10). At the same time, in MI250X the number of transistors is slightly larger than in A100 (there are 54.2 billion of them), and the MI250X die size is noticeably lower (in A100 — 826 mm^2). Accordingly, ideas arise about the possible comparability of MI100 and V100, as well as MI250X and A100 — which has been tested in a number of publications with performance data. As for the comparison of the technological indicators of the MI250X and A100, it requires fundamental clarification, which will be discussed below.

To compare the MI250X with the A100, we should add a comparison with the H100, which will appear in 2023. Here, the new 4 nm technology level allowed Nvidia to place 80 billion transistors with a reduced die size (compared to the A100) 814 mm^2.

The main indicators of GPU MI100, MI200, A100 and H100 are given for comparison in Table 20, 21, and Figure 18 shows a higher (than CU) level of computing power scaling in CDNA 2 — GCD, which also has 4 Compute Engine's (CE), each containing 28 CUs (two columns of 14 CUs each in the figure [**305**]. Two CUs of the 112 physical CUs are disabled [**41**] — accordingly, Table 20 indicates 110 CUs.

The GCD die is of fundamental importance for AMD technology: the top MI250 and MI250X models each contain by two GCDs [**305**], connected as a chiplet [**299**].

As GCD interconnect the Infinity Fabric 3.0 (discussed in the next section of the review) is used, but its bandwidth is much lower than the GPU memory bandwidth (see Table 21). When programmed, MI250 and MI250X will be treated as two different GPUs [**305**]. Accordingly, returning to the technological comparison with the A100 and H100 GPUs, MI250/MI250X can also be perceived as GPU pairs. And per one GCD in the MI250X, the number of transistors in it, roughly speaking, is 2 times less than in the A100.

If we move from the CDNA 2 architecture to the supplied MI200 models, they differ in the number of active CUs (see Table 20). According to [**305**], each CU contains 64 cores (64 shader cores or stream processors in AMD terminology, equivalent to four SIMD blocks with vectors of 16 numbers) — these are some analogues of CUDA cores in Nvidia GPUs. Each such core can perform multiply-and-add commands with FP64 numbers. Then the peak GPU performance (FLOPS) is twice the product of the number of such GPU cores and the clock frequency. According to the established tradition, manufacturers use the accelerated frequency in calculations of peak performance — such numbers are given in this table.

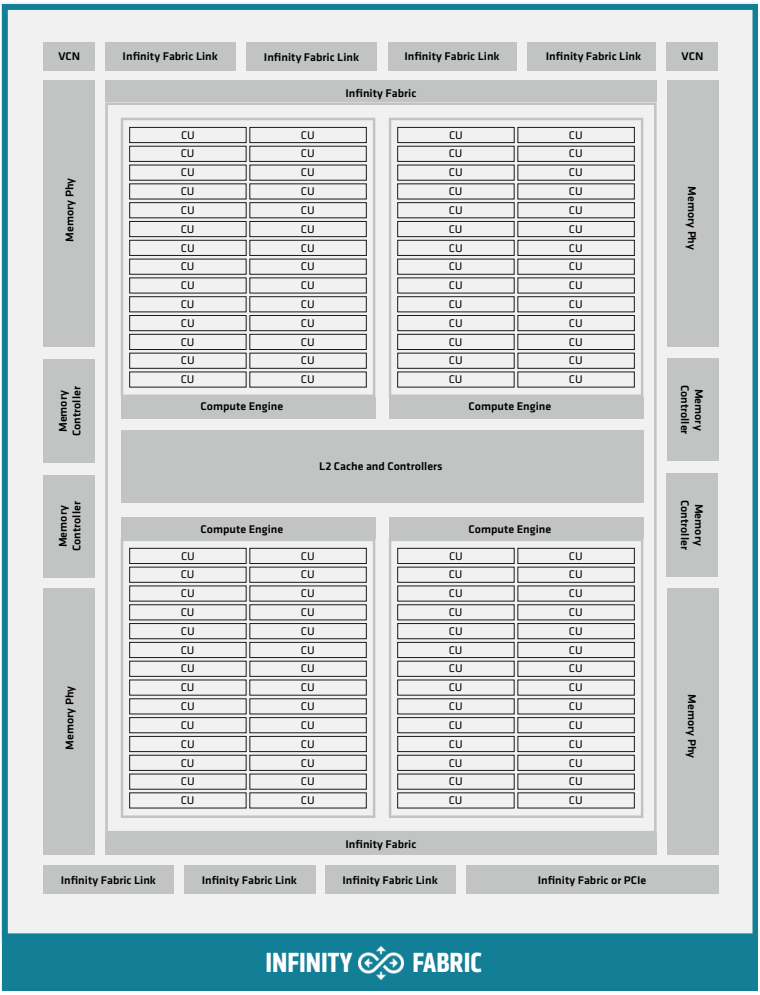


FIGURE 18. General construction of a GCD die (Figure from [305])

In CDNA2, the ALUs located in the CU became 64-bit, and it became possible to pack two FP32 numbers in place of one FP64, and work with them, for example, in multiply-and-add instructions [305]. The MI250X's peak performance with packaged FP32s doubles to 95.7 TFLOPS without using matrix cores.

In CDNA (in CUs), matrix cores have appeared (analogues of tensor

TABLE 22. The number of results obtained per clock cycle on one CU in CDNA (MI100) and CDNA 2 (MI200)

Format	By vector units		By matrix units	
	CDNA/MI100	CDNA 2/MI200	CDNA/MI100	CDNA 2/MI200
FP64	64 FLOPS	128 FLOPS	—	256 FLOPS
FP32	128 FLOPS	128 FLOPS	256 FLOPS	256 FLOPS
FP16	—	—	1024 FLOPS	1024 FLOPS
BF16	—	—	512 FLOPS	1024 FLOPS
INT8	—	—	1024 numbers	1024 numbers

Data from [307]. It also lists the number of clock cycles required to perform each type of MFMA operation. Running mixed format MFMA with FP32 results gives 1024 FLOPS per clock cycle.

cores in Nvidia GPUs), which can execute MFMA (Matrix-Fused-Multiply-Add) commands — "multiply-and-add" on small-sized matrices using mixed precision, but there the maximum precision is FP32 [304], as in the V100. And in CDNA 2 — and accordingly in MI200 (similar to A100) MFMA can also work with the FP64 format. Acceptable sizes of matrices with FP64 ($M \times N \times K$ in formula (1)): $16 \times 16 \times 4$ or $4 \times 4 \times 4$ [299].

It is useful to note here the peculiarity of the MFMA command set (it is a set, since for each combination of number formats used in different matrices there is its own version of the MFMA command) and similar WMMA commands in Nvidia GPUs. They perform the operation expressed by formula (1) with $\alpha = \beta = 1$, but the result is placed in another matrix **D** instead of **C** on the left side of this formula. There is no support for working with specific sparse matrices for AI area (as in the A100) in CDNA 2. ISA documentation for CDNA 2 is available at [306].

To calculate the peak performance of MI100 and MI200, you need to know the number of results obtained per clock cycle in one CU; this data is presented in Table 22. For example, each CU in CDNA 2 has 4 matrix cores, each of which produces 64 FP64 results per clock cycle [305], which after multiplying the total number of matrix cores in the GPU by 64 and by the clock frequency gives the peak performance shown in Table 20 when working with matrix cores.

In general, if we compare the peak performance data presented in Table 20, we can see that when working with vector cores, the MI100 is ahead of the A100 (and naturally the V100, see Table 12) for FP64 and FP32 in this indicator, giving this an potential advantage for all (working with vectors) HPC applications (unless performance is limited by matrix multiplication). There is no support for FP64 in the MI100 matrix cores, and with an accuracy below FP32, the MI100 is competitive with the V100 in this performance indicator, being much inferior to the A100.

On vector cores, the MI250 and MI250X models containing two GCDs each outperform the A100 in peak floating point performance. The same occurs when working with matrix/tensor cores (with the exception of FP32). However, A100 for the field of AI, when using special sparsity of matrices with tensor cores when working with reduced precision relative to FP64, is ahead in terms of peak performance.

The actual performance achieved fundamentally also depends on other factors, primarily the memory hierarchy — but first you need to decide how the comparison is made: logically, the MI250 and MI250X models are presented as two GPUs. If we divide the peak performance of these two models presented in Table 20 by two, then for FP64 all MI200 models are still significantly ahead of the A100, and for FP32 their advantage is small (for tensor cores the A100 also shows higher performance, but the original data are not supported in standard FP32 format).

When compared with one GCD, the newly released H100 already surpasses the MI200 in peak performance with FP64, and at lower precisions this superiority is large. It should be noted that the peak performance of one GCD MI250X for the FP64 format is very close to that of the H100-PCIe (see Table 20). A systematic comparison of these GPUs based on the actual achieved performance during the preparation of the review was impossible due to the lack of relevant publications, and in practical terms, a comparison of the H100/GH200 with the expected MI300 may become relevant in the near future.

There are other execution units in CDNA 2 that are relevant to working with images and video, but they are not discussed here — for example, GCD has two VCN blocks for machine learning tasks in working with images and video (see Figure 18).

Now we should turn to the memory hierarchy — the second main component that determines GPU performance. The most detailed data on the entire MI200 memory hierarchy is presented in [**299**]. The L1 cache here has a structure traditional for AMD GPUs, including not only division into an instruction cache and data cache, but also division of the latter into a vector cache (for working with a vector register file) and a scalar cache. And it is impossible to directly compare the capacity of the L1 D-cache in CDNA 2 with the capacity of such a cache in the A100 due to the fact that CDNA 2 also has a separate LDS shared memory with a capacity of 64 KB per CU [**41**]. And in V100 and A100, the corresponding cache (the size of which is given above in Table 21) is unified, and includes L1 and shared memory.

The L2 cache located in the GCD has a capacity of 8 MB and is a 16-way set-associative cache. The L2 cache in GCD has a total data transfer rate with lower level caches of 4096 KB per clock [**299**]. More precisely, the L2 cache in GCD consists of 32 slices, each of which transfers 128 bytes/cycle, or a total of 6.96 TB/s [**41**]. Through the 3rd generation Infinity Fabric interconnect, L2 cache data can communicate with HBM2E memory at a speed of 2 KB per clock. Attached to a single GCD, the HBM2E memory has a capacity of 64 GB with a bandwidth of 1.6 TB/s [**299**].

Models MI250 and MI250X each have two GCDs. Accordingly, at the level of the entire model, the indicators listed in the previous paragraph are doubled, as shown in Table 21. And Infinity Fabric in CDNA 2 began to ensure cache coherence between GCDs, and the HBM2E memory, as well as the L2 cache of both GCDs, are shared [**299**]. If the mapping to the A100 is done relative to a single GCD, then the capacity and throughput of HBM2E in CDNA 2 is higher than that of the A100-40GB. But Nvidia then began producing A100 models with double the capacity of the HBM2E (80GB), which have more capacity and bandwidth than the single GCD in the MI250/MI250X.

Comparing the listed memory indicators in Table 21, we can say that they are comparable in different AMD and Nvidia GPU models (MI100 and V100, MI200 and A100; H100 is not taken into account here). However, the striking difference is in the capacity of the L1 and L2 caches: in the A100 they are much larger than in the GCD, which can have an important impact on the performance achieved. The MI250/MI250X L2 cache capacity per GCD remains the same as the MI100 (8 MB), slightly larger than the V100 (6 MB) [**185**]. But in the A100 this capacity was sharply increased and became several times larger than that of the MI200. As for the L1 cache, in the MI100 its capacity was only 16 KB (plus 64 KB LDS) on the CU [**262**] versus 128 KB on the SM in the V100 (where this memory is partially used as shared memory) [**185**]. The MI200 CU also has the same L1 cache capacity as the MI100 (and also has a separate LDS memory).

But the capacity of the L1 cache is 8 times less than that of the unified V100 cache, and additional LDS may not compensate for this — even after allocating part of the capacity of the unified V100 cache to shared memory, the remaining capacity for the L1 cache may be much larger than in MI100 and MI200. This has already caused performance problems (see Subsection 5.3.2.2 on MI250/MI250X performance further on this), so compared to the A100, the MI200 cache is potentially bottlenecked place.

In any case, the performance publications discussed below have already noted the shortcomings of the cache memory in the MI100 and MI200. It is possible that to optimize applications for MI200, it is advisable to use not a simple roofline model, but an empirical one taking into account cache memory.

If we compare the TDP (see Table 21), then for the MI200 GPUs compared (to the A100), these indicators are also quite close, although the TDP of the A100-SXM models is higher. It should also be noted that the numbers indicated in this table in the TDP line for AMD GPUs refer, as indicated in documents [**300**–**302**], to TBP (Total Board Power).

In general, everything already discussed above suggests that at the hardware level, the performance indicators of the MI100 are comparable to the V100, and those of the MI200 (per GCD) are basically comparable to the A100, which makes it interesting and relevant to compare the corresponding actually achieved indicators of their productivity.

The last thing to discuss in this section is PCIe support in the MI100 and MI200 GPUs. Previously, this bus was used for communication between the device and the host, and its limited bandwidth could become a bottleneck for GPU performance. When multi-GPU servers began to be offered, this bus could also be used for communication between GPUs, which would also become a bottleneck in the server. In the GPUs discussed in this section of the review, they all have a special high-speed interconnect for communication between GPUs (Infinity Fabric for AMD, NVLink for NVidia). But to communicate between a host and a device, the use of such a specialized interconnect requires its hardware support by the host processor.

This has always been done on AMD GPUs since Infinity Fabric was originally focused on communication between AMD CPUs. This is not the case for NVLink — and required the use of PCIe to communicate with conventional x86-64 server processors. Therefore, the advantage of IBM Power9 was its support for an interface with NVLink for communication with the V100, thanks to which Power9 is also used in such famous supercomputers as Summit and Sierra. All GPUs compared in the Table 21 (except H100) have support for PCIe-v4 ×16 with a bandwidth of 64 GB/s.

But here there is a fundamental difference between these GPUs from AMD and from Nvidia: PCIe in AMD GPUs is not used for communication with the EPYC CPU, but is a commonly used PCIe interface to which network cards are connected, for example, for communication between

cluster nodes. The use of Infinity Fabric in AMD GPUs gives them an advantage in connection bandwidth to the CPU, including compared to the A100-PCIe. And the PCIe interface available in the MI100/MI200 is aimed at using it in conjunction with MPI versions that support these GPUs for exchanging messages between servers without involving the CPU — this is used, for example, in the SLATE dense linear algebra library, focused on similar systems with distributed memory [309].

The consideration of GPU interconnects in this review is carried out in sections devoted to GPU-based computing systems and is considered as additional special hardware for the GPU, as well as possibly special processors that support the corresponding interface. Infinity Fabric is therefore discussed in the next section.

5.1.2. *Computing systems based on MI200*

First of all, an analysis of the Infinity Fabric interconnect is required here. It was originally close to PCIe and is currently used for communication between EPYC processors in servers. Like NVLink, new versions of Infinity Fabric appeared as development progressed; in CDNA 2 this is already the third generation [299].

The MI200 has different Infinity Fabric connection options, selected for optimal work in different physical conditions (for example, for short connections between GCDs within the same GPU, a special interconnect is used). One Infinity Fabric channel is 16 bits wide in one direction (this was also the case in MI100). The bandwidth of one such channel in CDNA 2 is 50 or 100 GB/s in one or two directions, and between two GCDs in one GPU 4 channels are used with a total bandwidth of 400 GB/s [299] (see Figure 19 with communication topology in the nodes of the LUMI supercomputer [41]).

For communication between GCDs from different GPUs, a multi-GPU server uses one or two Infinity Fabric channels with bidirectional bandwidth of 100 or 200 GB/s, respectively. Finally, to communicate the MI250X with EPYC CPU, uses another Infinity Fabric variant with 72 GB/s bidirectional bandwidth to communicate with each GCD; In addition, host interconnects such as the Cray Slingshot-11 [41, 305] (see Figure 19).

The MI200-containing servers themselves may use a single or dual EPYC CPU option, and different interconnect topologies are possible there [305]. Figure 19 [41] shows the topology for a server with 4 MI250X and one EPYC — this corresponds to the LUMI [41] and Frontier [303] supercomputer nodes used.

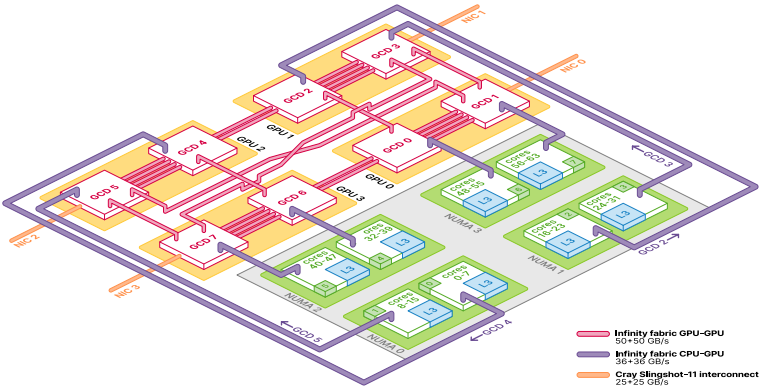


Figure 19. Using Infinity Fabric in a LUMI supercomputer node with 4 MI250X (Figure from [41])

Figure 19 corresponds to the flagship node topology for HPC; for it, [299] reports a total Infinity Fabric bandwidth of 1.54 TB/s. [305] presents another server topology with 4 MI250s coupled to two EPYC CPUs via a PCIe switch as a "mainstream" for HPC and AI, and illustrates another topology for a flagship server for AI with 8 MI250s and two EPYCs.

The MI210, which has one GCD, has its own modifications to work with Infinity Fabric. The MI210 provides 64 GB/s of bandwidth to the CPU without the need for PCIe switches, and the MI210-MI210 interconnect can use 3 Infinity Fabric lanes for a total of 300 GB/s of bandwidth. Finally, for a multi-GPU server with MI210, flagship and “mainstream” options are also offered [305].

Different topologies can also occur in Nvidia multi-GPU servers. However, the presence of different types of Infinity Fabric connections can complicate the optimization of highly scalable programming in multi-GPU servers with MI200. However, in [308], which examined various Infinity Fabric interconnect options on a server with 4 MI250X GPUs, no significant NUMA effects were identified on the CPU-GPU communication bandwidth at the HIP API level.

The Frontier supercomputer nodes use one 64-core EPYC 7A53 processor (operating at 2 GHz) with 4 MI250X. This CPU, codenamed Trento, was created specifically to working with the MI250X in Frontier nodes. The L3 cache capacity of this CPU is 32 MB for each of the eight 8-core groups (256 MB total per CPU). The CPU is configured as 4 NUMA

nodes, each of which is connected to 128 GB of memory, so its capacity in the server is 512 GB (8 channels of DDR4-3200) [41, 299, 303]. The possible topology for using Infinity Fabric in multi-GPU servers with MI250X was discussed above (see Figure 19); [41] points out the importance of proper NUMA node binding to the GPU, which can be critical to optimizing application performance. Frontier was built by Cray/HPE and uses the Slingshot-11 interconnect to connect the server nodes [299].

The third-place Top500 supercomputer, LUMI, uses the same hardware as Frontier [41, 42]. Here we must also point out the Crusher system, which is actively used in research, containing 192 of the same nodes as Frontier (9408 nodes) and LUMI (2560 nodes), and is used for testing and development. And even earlier, the predecessor of Frontier was considered the Spock cluster, which had 36 nodes containing by 4 MI100 [310] (see Table 23 below).

Naturally, almost all the world's leading server manufacturers supply models containing the MI200 GPU. Their use in modules of the standard OAM (OCP Accelerator Module) form factor [305] contributes to the rapid and widespread adoption of such servers. Servers are available in a variety of sizes from 1U to 4U, as well as 6U and 10U, containing from 1 to 8 GPUs (as well as 10 and 20) MI100, MI210 and MI250. These servers can have 1–2 AMD EPYC CPUs of Zen 2 and Zen 3 architectures, as well as 2nd and 3rd generation Intel Xeon Scalable processors. A list of such servers is available at [311].

To achieve maximum performance when working with the MI250/MI250X GPU, liquid cooling is required (without it, it is usually impossible to have more than 500 W). A classic example of this is the HPE Cray accelerator blade EX235a [46], used in related supercomputers.

5.2. SDK for MI100 and MI200

The discussion of SDKs in this section of the review is very limited, since most of the corresponding components from AMD are direct analogues of the SDKs from Nvidia described above, the names of which are also often very similar. AMD's SDK for the GPUs in question is called ROCm (Radeon Open Compute, and m stands for "multi-GPU computing" at the heart of these tools [312]). One advantage of AMD's SDK is that it is available in source code [312], while Nvidia's SDK is simply freely available.

If in 2020, when MI100 appeared, version ROCm 4.0 was available [312], then by mid-2023, before the appearance of the June Top500 list, version ROCm 5.5.1 was already available [313]. If the basis of NVidia's SDK should be considered the CUDA API, then the basis of AMD is the HIP API[58]. HIP as a programming model is extremely close to CUDA, and similar terms are used there (see Table 1). In HIP, when working with AMD GPUs, a warp (wavefront in AMD terminology) is used with a size of 64 threads [313], and not 32 (as for SM in Nvidia), which is related to the CU architecture.

But programs from Nvidia GPUs began to be portabled to AMD GPUs regardless of the appearance of the MI100. Thus, information on the achieved performance when porting various software systems from a popular area of application of modern GPUs, classical molecular dynamics, is presented in [314].

As an advantage of HIP over CUDA, we can point out the possibility of using HIP when working on Nvidia GPUs, which makes it possible to port software for AMD GPUs to Nvidia hardware. But here two important clarifying points arise. Firstly, we need data on the comparison of the performance achieved when running HIP relative to that achieved using CUDA (this will be discussed below). Since HIP uses certain CUDA backends when working with Nvidia GPUs, this eliminates possible problems.

It is generally believed that using HIP causes almost no performance degradation compared to direct CUDA encoding [42]. Data actually demonstrating this for one of the algorithms for solving a CFD problem with an unstructured mesh are available in [168]. But the new version of CUDA for H100 has extensions, the most striking of which can be considered a change in the hierarchy of the used sets of threads (a cluster of thread blocks has appeared). Accordingly, potential HIP performance issues for Nvidia GPUs may remain.

The ROCm software stack at the lowest level includes drivers. Currently, ROCm support is provided on Linux (drivers are included in the kernel module, which can be installed on Ubuntu, RHEL and SLES distributions). It is clear that ROCm includes all the typical components for an SDK. There is a compiler, now called ROCmCC (with support for HIP, OpenMP and OpenCL), which is based on Clang/LLVM [313]. Naturally, there is a profiler, debugger and performance tracing library.

Math libraries come with the prefix roc or hip. The roc prefix is used in the names of libraries optimized for AMD GPUs, and the hip prefix is used for libraries where tools for Nvidia GPUs are used as a back-end [313]. Given the prefix the full names of these libraries make their purpose obvious and are not given here. There are also communication libraries similar to the corresponding tools from Nvidia. For example, Nvidia's ROCm counterpart to NCCL is RCCL.

Even due to the syntactic similarity between HIP and CUDA, it seems natural that ROCm would have a conversion tool from CUDA to HIP, HIPIFY [313]. But in the general case, this should be considered as the first semi-automatic step towards such a transformation, which may require further manual modification, if only because not all CUDA APIs are supported in HIP (see, for example, [42]). Here we should point out a certain analogy between HIPIFY and Intel DPCT tools.

It is important to point out here the open source hipSYCL project (now called *AdaptiveCpp*) at the University of Heidelberg (Germany), an implementation of SYCL targeting CPUs and GPUs from different manufacturers. And in [165] an attempt has already been made to implement oneAPI without the DPC++ compiler, using hipSYCL. A comparison of the performance of MI250X using different implementations of SYCL, DPC++, as well as HIP and OpenMP on different applications is carried out in [176].

Another important project for working with AMD GPUs is the GPUFORT project [315]. These tools transform one source code into another source code. There are two possible options for such transformation of the source text. First, from CUDA Fortran (or possibly OpenACC) to a Fortran variant with OpenMP version 4.5 directives. The resulting text can then be compiled using AOMP (a Clang/LLVM-based compiler with a Fortran front end). Secondly, it is possible to use HIPFORT tools, which provide a Fortran and HIP runtime interface and access to math libraries. If compiled to run a program on an Nvidia GPU, HIPFORT provides an interface to the CUDA runtime tools and associated math libraries [316].

As for Fortran, it should be noted that there is support for the AMD GPUs under consideration outside of ROCm — in the famous HPE Cray Fortran, which also has support for OpenACC, which is missing in ROCm.

It is clear that the operation of the more traditional parallelization tools MPI and SHMEM is also ensured. In [317], the performance of a number of MPI variants that support working with CDNA and CDNA 2 — OpenMPI, Cray MPICH, MVAPICH2-GDR (as well as RCCL tools) was studied, including when using the Cray Slingshot-10 interconnect in the famous Spock cluster (it has nodes with 64-core EPYC 7662 and four MI100).

We're not talking about the AI frameworks included with ROCm here, since they're AI-specific and the review is primarily focused on HPC — but ROCm is naturally integrated with the major frameworks [313].

In conclusion of this section, it can be noted that both the hardware and software of the GPU MI100 and MI200 do not show such a strong focus primarily on AI (and on HPC — rather secondary) as in the GPU H100 (and partly in the A100). At Nvidia, this it shows up not only in documentation, but also in the new supercomputers that are emerging: unlike the more "traditionally oriented" Frontier with AMD MI250X, the H100 is used in supercomputers from the Top500, focused on AI. Nvidia produces AI-ready, high-performance computing systems up to the supercomputing level.

5.3. MI100 and MI200 performance data in benchmarks and applications

By the time the June Top500 list appeared, many articles had become available that examined the performance of these GPUs in different benchmarks and on different applications, including performance comparisons with respect to the V100 and A100. In most cases, when there were sufficient publications for the specific AMD GPU models under consideration, this review prioritized HPC-related works for analysis (but many data on AI are also provided). Among these publications, we also selected articles related to performance data on widespread and relevant mathematics problems. From publications on applications in this review, traditional areas known in HPC, for example, computational chemistry or CFD problems, were also selected, with priority given to world-known applications.

But here it is necessary to highlight the work on the ECP project and the summary data of a review nature on more than 20 applications (more precisely, projects) specifically focused on exascale — for example, [**262**]. There is a large number of performance comparison data for the MI100, MI250X, V100 and A100. Since performance estimates in the limiting case are interesting for a specific application or at least an application in the relevant domain, in many cases it is useful to look at the data in [**262**], where performance is also presented at the single GPU level.

However, it must be borne in mind that the ECP does not mainly use globally used applications, but their special versions (possibly parts) made to focus on exascale, or new developments. And this may be completely ineffective and does not provide accurate estimates for performance in more typical applications, for small computing systems, or for calculations of not so complex objects. In addition, this is just the initial data from ongoing work on various projects using early versions of ROCm in AMD GPUs, and the results will clearly be improved (some clarifying articles related to specific projects have already appeared). Accordingly, a very large number of criticisms appeared here regarding many ROCm components of different versions from 4.2.0 to 4.5.0, which AMD tried to quickly fix

Since much of the following data on the performance of the MI100 and MI200 GPUs was obtained using well-known supercomputers (including those from among the leaders in the Top500), brief information about such computing systems (including those used for comparison with Nvidia GPUs) is summarized in Table 23.

5.3.1. *Performance of MI100*

Comparative performance data of the MI100 compared to the V100 and A100 will be discussed here (everything relative to the MI200 is discussed below).

As for comparing the performance of GPU MI100 and A100, although the data in Table 20 shows slight advantages of MI100 in terms of peak performance (for example, with FP64), limited attention should be paid to these indicators for GPUs — no less often performance is correlated with memory bandwidth, and more precisely, more important is to look at the roofline model data. The available data from articles generally clearly indicate large (often several times) performance advantages of the A100 relative to the MI100, although there are exceptions.

Table 23. Computing systems whose nodes were actively used in publications on GPUs performance cited in this review; for supercomputers from the Top500, their number in the list is given in parentheses

Computing system	Number and type of GPUs in the node	Number and type of CPUs in the node	Interconnect (between nodes)[1]
Frontier (1) [303]	4×MI250X	1×EPYC 7A53(Trento)	4×HPE Slingshot-11
LUMI (3) [318]	4×MI250X	1×EPYC 7A53(Trento)	4×HPE Slingshot-11
Crusher [319]	4×MI250X	1×EPYC 7A53(Trento)	4×HPE Slingshot-11
Spock [320]	4×MI100	1×EPYC 7662(Rome)	1×HPE Slingshot-10
Leonardo (4) [205]	4×A100-40GB	1×Xeon Platinum 8358	4×Nvidia Infiniband HDR
Perlmutter (8) [206]	4×A100-40GB	1×EPYC 7763	4×HPE Slingshot-11
ThetaGPU [321]	8×A100-40GB	2×EPYC (Rome)	8×Infiniband HDR
Polaris (19) [322]	4×A100-40GB	1×EPYC 7543P (Milan)	Slingshot-10
Summit (5) [323]	6×V100-16GB	2×Power9	2×Mellanox Infiniband EDR

[1] The interconnects used in the publications cited in the review are indicated.

Therefore, we will present here only a few confirmations of this, mainly in an integral sense (with a wide scope of applications) — and not based on individual specific examples. For example, in [45] data corresponding to the above are presented on 6 different applications and mini-applications (from different fields of science) included in the ECP project or related mini-applications. According to data from [45] we calculated how many times the achieved performance of A100 was higher than MI100: for AMR-Wind (CFD part of the ExaWind, Exascale Predictive Wind Plant Flow Physics Modeling project) — on three different kernels in 1.7-5.3 times; in the quantum chemical mini-application GAMESS RI-MP2 — 5.8 times; in the TestSNAP mini-application for the SNAP quantum potential, which is then used in the LAMMPS molecular dynamics application (from the EXAALT project for nuclear fusion problems) on three different kernels — 2.2-7.9 times, and so on.

As another example of performance data covering even more projects and applications from ECP, we point out slightly more recent publication [262] (see also [324]); comments about [262] are given above. Our calculations of relative performance based on data from [262] show that the A100 was about two times faster on CFD problems (NekRS), and three times faster

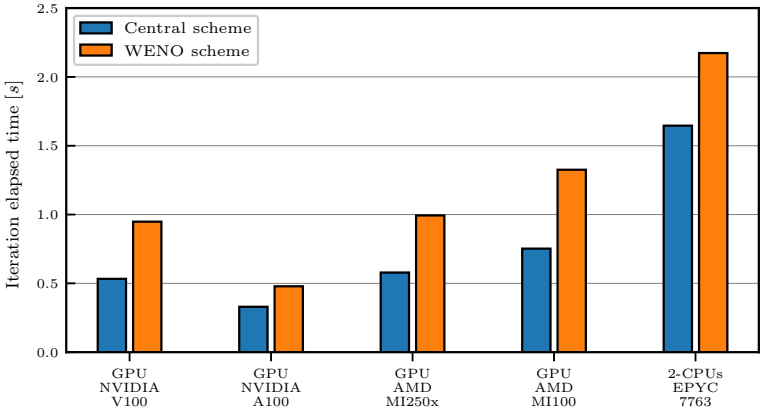


FIGURE 20. Time per iteration (seconds) for STREAmS-2 with a $420 \times 250 \times 320$ grid for two calculation schemes: the central flow estimation scheme and the WENO scheme (figure from [**325**])

on quantum chemistry (Gamess, Fock matrix construction) and molecular dynamics problems (LAMMPS). For TestSNAP, the A100 was 2.8 times faster than the MI100.

A newer version of ROCm 4.5.2 on another CFD application, STREAmS-2 [**325**], also produced similar MI100 performance lag values (see Figure 20) [**325**]. These data are discussed in more detail below in Subsection 5.3.2.2. It is unlikely that progress in new versions of ROCm can completely eliminate such a lag.

In [**225**] performance using MI100, A100, and V100 was examined on SPEChpc 2021, which contains components from applications in different HPC fields, but the data for MI100 was obtained in a different "small" suite of SPEChpc than was not used for other GPUs.

Application performance is the most practically relevant, but corresponding data on the performance of MI100 relative to A100, showing similar data above, is also available for specific mathematical methods—for example, for FFT [**251**], for QR decomposition of square matrices using the MAGMA library for single and double accuracy [**241**], for BabelStream bandwidth benchmarks [**42**]. There is also other data on BabelStream benchmarks for their implementation using DO CONCURRENT Fortran tools [**236**], where the memory bandwidth of MI100 was slightly different from the C++ version of BabelStream (sometimes Fortran gave better results). The bandwidth for both A100-80GB and A100-40GB in [**236**], was naturally found to be much greater than that of MI100.

Modern GPUs, their SDKs, and the applications that use them are complex applications require different parameters for optimization for different GPUs, and there are exceptions to all rules. Thus, in [241] data on the same QR decomposition with rocBLAS show the advantage of MI100 over cuBLAS with A100. And in [326] within the AnySeq/GPU code for bioinformatics tasks using FP32 cores in MI100, V100 and A100, the performance (in trillions cell updates per second, TCUPS) was 3.8, 1.7 and 3.3 TCUPS, respectively. But in general, in publications, the strong lag between the MI100 and the A100 in terms of performance is shown quite clearly.

But comparing the performance of MI100 with V100 seems more relevant, including as a possible addition or basis for comparing A100 and MI200. Here the achieved performance is indeed quite comparable, and the peak FP64 performance of the MI100 is still noticeably higher than that of the V100. This gives the MI100 a chance to achieve higher performance for compute-intensive applications (in roofline model terminology). Let us immediately give a striking example of this in [262], where, using the quantum chemical application NWChemEX in calculations with the computationally intensive method of coupled clusters (CCSD), a comparison of the performance of Spock nodes (4 MI100 each) and Summit (6 V100 each) showed the advantage of Spock. NWChemEx used CUDA on Summit and HIP (ROCm 4.3.0) on Spock.

The acceleration values of Spock relative to Summit that we calculated (according to Table 11 in [262]) show that on one Spock node the total time of CCSD calculation is 1.5 times less than on the Summit node, and the time of an additional more complex calculation of the correction T (for triple excitations) are 1.9 times fewer than on the Summit node.

As the number of nodes increases to 4, the performance advantage of Spock at an equal number of nodes quickly decreases, and at 4 nodes Spock is only faster in terms of computation time of T [262]. It must be said that probably most modern GPU applications are more memory-bound, and the V100 is less behind the MI100 in memory bandwidth.

The article [262] provides comparative data on the performance of deep learning for MI100 and V100 within the framework of two ECP projects — CANDLE (precision medicine for oncology) and ExaLearn (the goals of the project are clear from its name). In CANDLE, BERT performance is moderately greater on the MI100 than on the V100, but not to the extent that the relative performance of these GPUs would indicate (obviously in [262] referring to peak hardware performance). And for ExaLearn, [262] indicates a much lower performance of the MI100.

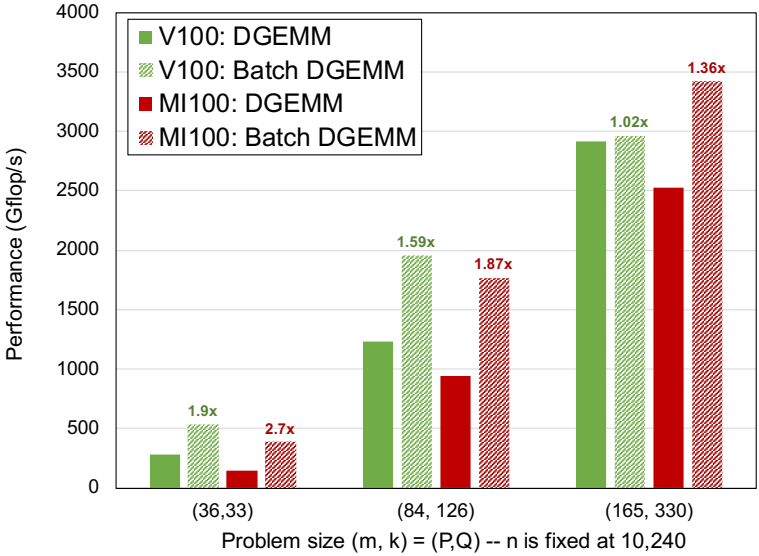


FIGURE 21. Performance of MI100 and V100 on DGEMM (Figure from [242])

It is important that in addition to higher peak performance, MI100 showed higher memory bandwidth than V100 on all components of the BabelStream test using a wide range of different programming models [42]. In the same work, the performance of the miniBUDE (molecular docking) application on MI100 turned out to be higher than on V100-32GB and almost coincided with the performance of A100-40GB.

A very relevant comparison is the performance of MI100 and V100 on DGEMM, which was carried out in [242] for batch and conventional implementation of matrix multiplication. The corresponding data is presented in Figure 21 using ROCm 4.2 (hipBLAS) for MI100 and CUDA 11.2 (cuBLAS) for V100. The parameters m,n,k on the abscissa here correspond to the dimensions of the matrices M, N, K in formula (1). It can be seen that MI100 can be ahead of V100. [242] also provides data for CEED benchmarks from ECP, showing comparable performance between MI100 and V100, and performance on NekRS, where MI100 was 15% behind V100. The same 15% lag is indicated in [265].

In [327] calculations were carried out using the QUICK quantum chemical application in combination with the AMBER molecular dynamics application (according to the QM/MM scheme). In the starting tests, the MI100 gave performance two times less than the V100, due to the excessive

use of registers in AMD kernels. After improvements in QUICK, in the calculations of large-sized systems, the performance of MI100 and V100 turned out to be comparable (on small systems, V100 was much faster). The kernel, which calculates the gradient of the exchange-correlation potential, calculated much faster on the MI100 than on the V100 [**327**].

ECP-related work [**328**] compared the performance of MI100 (on Spock) and V100 (on Summit) on the well-known CFD benchmark HipBone (which is GPU-focused, based on the well-known C++-based NekBone benchmark, but does not cover all its functionality [**142**]), and compared the performance at different degrees of the polynomial of the kernel of the Poisson operator, which actually determines the performance of HipBone. HipBone uses FP64, but some preprocessing subtasks use FP32.

On one GPU, the performance of MI100 (with ROCm v4.5.0) obtained in [**328**] is approximately the same as the performance of V100 (with CUDA v11.0.3) at all degrees of the polynomial — sometimes MI100 is faster, and sometimes V100. The highest performance is achieved with the highest polynomial degree used (N=15): for V100 — 2101.4 GFLOPS, for MI100 — 2135.2 GFLOPS; good scalability was also shown on Spock and Summit nodes with the number of MPI ranks up to 32 and 48, respectively [**328**].

In report [**262**] based on data for some ECP projects, the comparability of the performance of MI100 and V100 is noted — this is said, for example, for AMR-Wind data (our estimate of the acceleration of V100 relative to MI100 according to Figure 20 in [**262**] is 1.4 times).

In the ExaSMR project (simulation of a small modular reactor core by combining two parts, neutron physics and CFD — NekRS is used for the latter), the Shift code with HIP (ROCm 4.2) for the first part was at first inferior to V100 with CUDA by a factor of two, but after optimization the lag was only 20%. In the NekRS part of ExaSMR, in some kernels, the MI100 was only a few percent behind the V100 on Summit [**262**]; our calculation according to tables 42 and 43 from [**262**] gives an acceleration of V100 relative to MI100 by 10–20%.

In other software components of other ECP projects, [**262**] indicated a more severe performance lag in MI100. On the QMCPACK application (quantum Monte Carlo calculations), nickel oxide supercell calculations achieved very different lags in different cases, but overall the MI100 gave significantly lower performance, and AMD compilers were criticized. In TestSNAP, MI100 in Spock lags behind V100 in Summit by approximately 2 times (our calculation using Table 19 in [**262**]). In the ECP project with the Gamess quantum chemical application, for ock matrix calculation

in different basis types with lengths from 4 to 12 thousand orbitals, MI100 is 1.4-2.6 times slower (our calculation according to Table 15 in [**262**]). Naturally, there is a lot of other data available in [**262**] comparing the performance of V100 and MI100.

Authors of [**329**] compared the performance of MI100 and V100 using mini-application, MiniMDock for molecular docking. In variants optimal for GPU performance (with HIP and CUDA), MiniMDock, when using medium and large ligands, MI100 (Spock, ROCm 4.5) was inferior in performance to V100 (Summit, NVHPC 21.11) by 32 and 39%, respectively, but on small ligands MI100 was slightly faster than V100.

Authors of [**330**] optimized some critical important kernels for the CFD application FUN3D, whose performance on the A100 was discussed above. For all of them except the viscous flow kernel, MI100 (with ROCm 4.2.0) was 18–53% slower than V100 (with CUDA 11.2).

Naturally, MI100 is used in a variety of areas, not only in HPC, but also for AI, for example, for deep learning in the DLRM model [**331**].

We can say that the MI100 is comparable in performance to the V100, although the MI100 often lagged behind. Much of the MI100 performance data reported above refers to preliminary results obtained shortly after the MI100 became available, and these results may still be quite significantly improved on new versions of ROCm (CUDA versions seem much more stable). But due to the clear advantages of all indicators of MI210 relative to MI100, not only those indicated in Table 20 and Table 21, but also all others known to the author at the time of writing the review, interest in MI100, in the author's opinion, relates to the use of MI100 as already purchased, and the acquisition of a new MI100 is not practical compared to MI210. Therefore, a more complete review of the available performance data for the MI100 is not provided here.

5.3.2. *Performance of MI200*

Looking at the data in Table 20 and Table 21, it is clear that on a per-GCD basis, the MI210 and MI250 models are almost the same, but the GCD in the MI250X has a slight increase in the number of CUs, which should provide a slight increase in performance. Therefore, although we will begin the analysis of the achieved performance with a number of current articles known to the author (at the time of preparation of the review) with data on the MI210, this may also be of interest for evaluations of two others models of the MI200 family.

5.3.2.1. Performance of MI210. As a basic indicator of the achieved performance of the MI210, we first point out the data Dell presented for its PowerEdge R750x servers — for double and single precision matrix multiplication (DGEMM and SGEMM) [**332**], including in comparison with data for the MI100. Dell's calculations on the MI210 used tensor (matrix cores and ROCm 5.1.3. A performance of 28.3 TFLOPS was achieved for DGEMM, and 32.2 TFLOPS for SGEMM. These data [**332**] indicate a large acceleration in MI210 relative to MI100 for DGEMM — 3.6 times (but in MI100 there is no FP64 support in tensor cores), and a low achieved percentage of the peak performance in MI210 (62.5%).

For SGEMM on MI210, performance was 9% higher than on MI100 [**332**]. But FP32 is supported in the MI100 tensor cores, which gives the MI100 even slightly higher peak single-precision performance than the MI210 (see Table 20 and Table 21).

In the HPL benchmark, the performance of MI210 reaches 18.2 TFLOPS per 1 GPU and grows almost linearly up to 4 GPUs (up to 72.6 TFLOPS), while the performance of MI100 per 1 GPU is several times less and grows more slowly with the number of GPUs in the server [**332**].

For HPL, there is also data from AMD itself, which allows us to demonstrate the possible impact of the ROCm version [**333**]. Here, using the newer ROCm 5.2.0 and rocHPL 6.0.0 on a single GPU, the HPL benchmark gave 21.07 TFLOPS (1.16 times faster than Dell), on 4 GPUs — 81.097 TFLOPS (1.12 times faster), on 8 GPUs — 159.73 TFLOPS. As noted in [**333**], performance may also vary depending on the driver version. We assume that the influence of differences in other components of the servers used in the benchmarks in [**333**] and [**332**] here is negligible.

On the HPL-AI benchmark with mixed precision FP16/FP32/FP64 on 4 GPUs in [**333**], a performance of 444.77 TFLOPS was obtained.

Dell [**332**] provides data on the performance of MI210 relative to MI100 in the LAMMPS molecular dynamics application for its several different types (including reactive molecular dynamics) and potentials — when working with FP64, MI210 is 18–30% faster, and with FP32 — by 12%. Data on the performance of LAMMPS for reactive molecular dynamics are also reported by AMD using a more newer version of ROCm [**333**].

The first articles with MI210 performance data began to appear for applications in various fields. For example, [**334**] provides performance data on the Uni-Dock molecular docking application developed in this paper, which is superior in performance to the well-known AutoDock-GPU application (although detailed Uni-Dock performance data is provided in [**334**] for V100 and MI100).

Among the works that compare the performance of MI210 and Nvidia GPUs, we point out CFD data [268] for the FUN3D application mentioned above in the section on A100 performance with its FLUDA library for GPUs. Based on the data in Table 4 in [268] we find that the A100-40GB is 1.44 times faster than the MI210 (and the MI210 is 1.29 times faster than the V100-16GB).

In [335], the performance of MI210 and A100-40GB was compared when running the PyTorch deep learning framework (PyTorch 2.0-20230102 was used) using ROCm 5.4.2 and CUDA 11.8 stacks. This article proposed TorchBench, a new set of benchmarks for the PyTorch software stack. In comparison, the classic MLPerf deep learning benchmark suite, whose GPU results were discussed earlier in this review, includes 8 deep learning models, while TorchBench has 48. TorchBench is open source and, thanks to its wide variety of representative models, allows for e.g. and identify situations with decreased performance when running PyTorch on a GPU.

Comparison made in TorchBench's 32-bit (default) configuration. The A100's TF32 support gives increased performance at the cost of reduced precision, so not all models can use it. For deep learning and inference of a trained neural network with FP32, the MI210 was faster in some models and the A100 in others, which does not give a clear performance advantage to one of these GPUs [335].

In general, MI210 immediately began to be used for tasks related to AI. For example, [336] developed a DLRM-oriented technique for overlapping communication and computation and created a kernel for MI210. And in [337], an improved algorithm was proposed and implemented on MI210 to improve the performance of a convolutional neural network (CNN), which, using the MIOpen deep learning library and ResNet50 source data, allowed to obtain 74% of the peak single-precision performance of MI210.

5.3.2.2. Performance of MI250/MI250X. As for the MI250/MI250X GPUs, each containing 2 GCDs, their potential competition with the A100 in terms of performance was demonstrated by the developers themselves. In June 2022, Nvidia cited data demonstrating the superior performance (and power efficiency) of the A100-SXM-80G relative to the MI250 in single- and quad-GPU servers (counting the MI250 as one GPU) on classic molecular dynamics applications AMBER, GROMACS, LAMMPS, NAMD, and OpenMM [338], and in August 2022, AMD provided opposite performance data for all of these applications except GROMACS — but instead of showing data for this program, it showed a performance advantage on the HPCG benchmark [339]. It is clear that such

a comparison requires additional data about the versions of applications and SDKs of the companies used, those used in the calculations of molecular systems and types of potentials, and so on. The performance analysis of the MI250/MI250X below also includes published data on molecular dynamics performance.

That the performance of the MI250X is high is evident from the success of AMD GPUs in the Top500, where Frontier and LUMI supercomputers take first and third places with HPL performance of 1194.0 and 309.1 PFLOPS, respectively, and HPCG performance of 14.1 and 3.4 PFLOPS respectively [4]. A fairly broad comparison of the performance of 8 different applications from different areas of HPC (mostly not from those widely used in the HPC world) on Frontier and Summit was carried out in [340], where showing the advantages of Frontier, a general starting comparative assessment of the performance of MI250X and V100 is given.

But since such successes on supercomputers are largely associated with a high level of scaling with the number of nodes, we present here just one more illustration based on a comparison of the performance of nodes with the A100 and with the MI250X — in the Perlmutter supercomputer and the Crusher cluster (brief data on these computing systems is available in Table 23). Their nodes contain 4 GPUs A100 and MI250X, respectively.

In [341] the performance of X-ray tracing code using Kokkos tools (as well as a version with CUDA) was studied on different numbers of Perlmutter and Crusher nodes (there, CUDA and HIP were used as back ends, respectively). Most of the computing time on the GPU there is taken up by the nanoBraggSpots kernel; Using Kokkos, nanoBraggSpots achieved over 60% performance improvement per Crusher node compared to Perlmutter [341]. The data in Figure 2 of this article shows that the performance of nanoBraggSpots with an equal number of nodes from 32 to 128 is several times greater on Crusher. And the use of Kokkos gave noticeably higher performance than the usual use of CUDA.

But then we need to keep in mind which comparison of Nvidia and AMD GPUs is most interesting. AMD MI250 and MI250X each have two logical GPUs (2 GCDs each), while the A100 is a single GPU. From a programming point of view, MI250 or MI250X are two GPUs, and it is interesting to compare the A100 with one GCD, which was often (but not always) done in publications. And then the question arises about costs — if they are comparable for the A100 and MI250X, it would be interesting to compare them with the whole MI250X. But here there is no comparison of cost indicators or TCO (it not only depends on where the GPU is purchased and used, but also changes over time).

Basic benchmarks and math libraries performance tests. The most important baseline benchmarks for the Frontier and Crusher nodes are available on the website of the US Oak Ridge National Laboratory, which owns both these supercomputers and Summit [342]. Some of this data also applies to the host CPU (EPYC 7A53). To work with MI250X (with one GCD), ROCm 5.3.0 is used there.

Article [342] provides information on the achieved bandwidth of MI250X memory, CPU-to-GPU and inter-GPU communications. It presents data on bandwidth in all five kernels of the BabelStream benchmark (classic version with FP64) using HIP and OpenCL, depending on the dimension of one-dimensional used in benchmarks arrays.

According to [342], the HBM bandwidth achieved in BabelStream is 77–86% of the peak value (82–90% is achieved for a CPU with its DRAM). The highest values were achieved here when running the copy (slightly less in mul) kernel of BabelStream using hipcc, a maximum of about 1.38 TB/s. But in some kernels of BabelStream, using OpenCL gave higher bandwidth than hipcc, and with the dot product kernel, hipcc (unlike OpenCL) gave abnormally low bandwidth.

The memory bandwidth of MI250X (one GCD) in [324] was also studied by executing SpMV from the Ginkgo library using different sparse matrix storage formats.

CPU-GPU bandwidth (measured by hipMemcpy) in [342] was 25.6 GB/s (71% of peak), and between GPUs in the osu_bw MPI test from OSU microbenchmarks [343] (using Cray MPICH 8.1.23) — from 37.6 GB/s (75.2% of peak) on one Infinity Fabric channel to 145.3 GB/s (72.7% of peak) on four Infinity Fabric channels [342]. In [342], the latest measurements are for one-way transfers—and Table 19 of that review gives a peak value of 100 GB/s per channel for two-way transfers.

Article [342] also provides data on the performance of MI250X on GEMM (using hipBLAS), including DGEMM. For GEMM with FP16, a specific test CORALGEMM [344] was used, the performance of which increased up to the highest matrix dimension of over 8 thousands. For FP32 and FP64, in addition to this test, another specific test gpu_xgemm [345] was used in [342] (see Figure 22).

The choice of one or another specific test does not give fundamental changes in performance here — it is mainly determined, of course, by the choice of FP32 or FP64. The use of hipBLAS does not give any outstanding results (such as achieving 90% or more of the peak value). But it's interesting whether the jumps in the performance curves depending on the matrix size are related to its "optimal" parity.

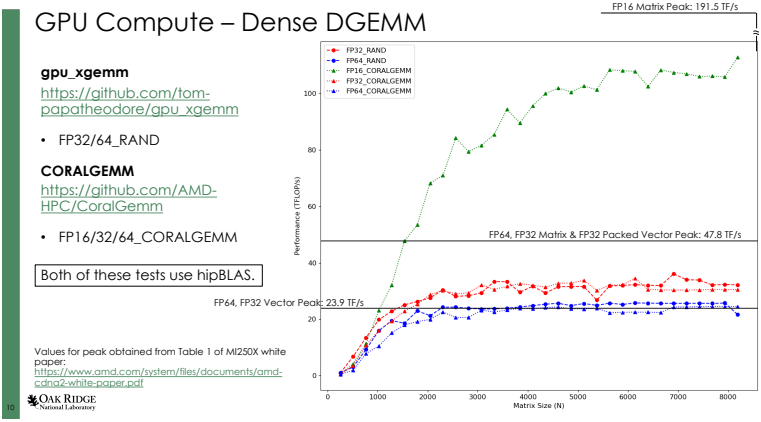


FIGURE 22. GEMM performance on one MI250X GCD (Figure from [342])

Article [342] also presents mixbench test data, but these results, as above in the section for A100, are not considered here, since they are more interesting for AI.

Benchmarks [333] presents data on the performance of MI250 (with ROCm 5.2.0) in the HPL (with rocHPL 6.0.0), HPL-AI (with HPL-AI-1.0.0) and HPCG 3.0 benchmarks. On one GPU the achieved performance was 40.45 TFLOPS (about 90% of the peak vector value), and on 4 GPUs it was 161.97 TFLOPS. Note that the results in [333] are for full GPUs with two GCDs. On HPL-AI (module with FP16/32/64) on 4 GPUs, performance reached 930.44 TFLOPS. On HPCG, 488.8 GFLOPS were obtained for one GPU and 1927.7 TFLOPS for 4 GPUs.

Computational Chemistry. Considering that AMD and Nvidia actively compared the performance of the MI250X and A100 on molecular dynamics problems, we will begin our presentation here with this area of computational chemistry. It should be noted right away that achieving high performance when porting molecular dynamics applications from Nvidia GPUs to AMD GPUs is not an easy task. The transfer of such software packages from CUDA to HIP is discussed in [314] (although the models analyzed in this review were not considered there).

AMD reports its LAMMPS 2022 molecular dynamics application performance across several famous of this application benchmarks on the MI250 and MI210 GPUs and compares them to results on the A100 [346].

A single MI250 outperforms the A100 on all benchmarks (while the MI210 containing a single GCD tends to lag behind the A100-PCIe-80GB). AMD has also demonstrated good performance scaling of LAMMPS up to 4 GPUs per server [**346**].

Authors of [**254**] conducted a detailed and in-depth study aimed at improving the performance of LAMMPS after porting it from Nvidia GPU hardware (Summit with V100 and AFW HPC11 computing system with A100-40GB used in the article) to MI100 GPUs (in the Spock cluster) and MI250X (in the Crusher cluster). LAMMPS uses the Kokkos library to support work with different hardware. To optimize the performance (studied for the normal FP64 format) of LAMMPS on Nvidia and AMD GPUs, roofline models were used for each kernel, and performance estimates involved 6 different potentials and calculations of liquid, metallic, granular, biological and polymer systems up to size 55 million atoms.

Support for LAMMPS operation on GPUs with long-range interaction-oriented PPPM potential was implemented in [**254**] using FFT, for which rocFFT tools were used on AMD GPUs. When running at ReaxFF potential on a 256K atoms molecular system, about 20% (5.6 TFLOPS) of the peak FP64 performance of a single GCD in Crusher was achieved.

In [**254**], despite a large amount of careful and extensive research done, the obtained performance estimates are indicated as preliminary, which is probably due to the active development of ROCm noted in this article (ROCm v4.5.0 was used in this work).

The pointed out different performance behavior when changing Kokkos parameters for AMD and Nvidia GPUs. This can be taken as an illustration of the difficulty of porting software from one type of GPU to another while maintaining efficient code optimization.

In [**262**] the performance of the SNAP quantum potential (as part of the EXAALT ECP project), then used as part of the work with LAMMPS, is analyzed. Although more performance information was obtained here for the MI100, which the MI250X was, of course, far ahead of, the GCD MI250X alone at the very beginning of 2021 was significantly behind even the V100 in this calculation, and much more behind the A100. This was due to the lack of L1 cache capacity in the MI250X CUs — the V100 and A100 used a capacity of 96 KB per SM for SNAP, 6 times more than the MI100 and MI200 per CU (see Table 20 and Table 21).

Table 24. Performance (ns/day) of AMBER 22 on different GPUs

Test	A100-PCIe performance[1]	MI250 performance[2]	H100-PCIe performance[1]
JAC Production NVE 4fs	1199.22	1871	1479.32
JAC Production NPT 4fs	1194.5	1794	1424.90
STMV Production NPT 4fs	52.02	80.65	70.15

[1] data from [349] (data as of 03/21/2023). Used Amber 22 Update 1, AmberTools 22 Update 1, Nvidia CUDA 11.4

[2] data from [333], used ROCm 5.2.0 and Amber container 22.amd_100

Work on optimizing the SNAP kernels continues [262], but although these were rather preliminary results, further progress is expected primarily in terms of the use of Kokkos here. [262] states that SNAP's L1 cache hit rate is (for the largest SNAP kernel) 66% for the MI250X versus 90% for the V100.

In [347], AMD demonstrated good scalability of the NAMD molecular dynamics application on the famous APOA1 and STMV NVE benchmarks using up to 8 MI250.

Above, at the very beginning of sec:5.3.2.2, it was noted that there was no data provided by AMD on GROMACS performance on MI250/MI250X in 2022, which was obviously due to temporary problems with porting the GROMACS code, and was quickly corrected.

In [333], there is data on the performance of MI250 on the most famous molecular dynamics applications AMBER, GROMACS, LAMMPS and NAMD. Let us give a number of examples. For GROMACS, in the well-known STMV (Satellite Tobacco Mosaic Virus) benchmark, a performance of 34.2 ns/day was obtained on one MI250X, and 89.26 ns/day on 4 MI250Xs. For NAMD 3.0, in the standard STMV NVE benchmark, a performance of 19.87 ns/day was obtained on one MI250X, and 77.132 ns/day on four MI250Xs.

The GROMACS 2023.1 version uses SYCL as a back-end for the MI200, what has shown good scaling of the achieved performance in the STMV benchmark when using up to 8 MI250X. It has been discovered that not all the features of CDNA 2, which can provide a significant increase in GROMACS performance, are yet used by the AMD compiler [348].

For AMBER22, here are performance data on 8 known AMBER benchmarks. Thus, on the Cellulose Production NPT 4fs benchmark, a performance of 227.2 ns/day was obtained. We have compared the data from [333] with the results available in [349], and summarized these data in Table 24.

But here we need to make important clarifications about comparing the performance data of different molecular dynamics applications on different GPUs. The calculation data provided relate primarily to the molecular system being calculated (where the total number of atoms is important), which is determined by the name of the benchmark itself. But the calculation time also depends on other parameters — the potential used, the time-step (often measured in femtoseconds, fs in Table 24) and the cutoff radius of interactions. Therefore, the data provided on the achieved performance may not be comparable, and the use of the same parameters is required, which are not even always given. The author is not aware of such a coincidence of parameters in different data sources of Table 24.

It should be borne in mind that in this table the data from [**349**] was obtained with a far from new version of CUDA; AMD itself compared in [**333**] with CUDA 11.6, the old version of CUDA may not be effective enough for the H100 — in the publications cited in this review, CUDA 12.2 was also used. But these data, as well as those presented above in this review, indicate that the MI250 has real competition in molecular dynamics performance with the A100 (and maybe with the H100), perhaps even ahead in performance, and the dependence on the versions of the SDKs used is obvious.

Another article [**350**], which examined the performance of MI250, is at the intersection of molecular dynamics and AI. In [**350**], using the V100, A100 and MI250 GPUs, a fairly large number of calculations were carried out using the DeePMD-kit software package for molecular dynamics simulation using machine learning potentials (instead of using the QM/MM method for this). DeepMD-kit allows these potentials to be integrated with LAMMPS, Amber, OpenMM and GROMACS applications. Based on Table IV from [**350**], we computed the performance speedups of A100-80GB over a single MI250 GCD. For LAMMPS molecular dynamics calculations for FP64 they ranged from 6% to 2.5 times. The lack of details in [**350**] about the software versions used (for example, about the LAMMPS version ported to the AMD GPU, used for molecular dynamics calculations) allows these values to be attributed rather to preliminary estimates, but still providing some information about the performance of the MI250.

The paper [**351**] can also be partly (in a certain mathematical sense) considered of interest for molecular dynamics problems — here, to calculate the Euclidean minimum spanning tree using an algorithm improved by the authors, A100 (with NVCC 11.5) and MI250X (1 GCD, with ROCm 4.5) and Calculations were carried out on 12 different data sets. Based on the obtained performance data in Figure 6 from [**351**], we calculated the relative speedup of A100 compared to 1 GCD, it is in the range of 1.5-1.7.

Table 25. Calculation time of correction, T (in seconds) on GPUs [352]

GPU	Programming model		
	SYCL	CUDA	HIP
MI250X (1 GCD)	17.41	—	15.56
MI250X (2 GCD)	8.97	—	8.12
A100	18.23	16.14	—

In general, it seems that stable reliable data on comparing the performance of the MI200 with Nvidia GPUs in molecular dynamics problems is still rather absent (in the sense that new and improved indicators are appearing).

In [352], data on the achieved performance within the framework of the NWChemEX project in ECP using MI250X and A100 in Crusher and Polaris supercomputer nodes, respectively, were studied (see Table 23). NWChemEX is also focused on calculations using the high-precision quantum chemical method CCSD(T), in which the main calculation time is spent on the correction due to triple excitations (T), which is a perturbation to CCSD, requiring $O(n^7)$ calculations, where n is the dimension of the basis. The kernel for calculating this correction was implemented in different versions using HIP, CUDA and DPC++; or this, the compilers clang-16, gcc-10.3.0 and CUDA-11.4.4/ROCm-5.1.0 were used (for Polaris/Crusher, respectively), and for parallelization between nodes, Cray-mpich 8.1.16 was used.

Table 25 shows the obtained times for calculating the correction T for the molecular fragment of ubiquitin (these are data from Table III in [352]) on one GPU. From these data it follows that one GCD in the MI250X was slightly faster than A100-40GB (both on HIP versus CUDA and using SYCL), and on two GCDs the performance of the MI250X increased by 1.9 times. At the same time, the performance achieved using DPC++ turned out to be quite close to that obtained using CUDA or HIP.

But in [352] on the A100 it was also found that in the SYCL implementation, the kernel for calculating T places very high demands on the L1 and L2 caches, and the generated PTX code showed a large number of loads and stores into local (private in SYCL terms) memory. Adding one clang key gave a 2.5 times increase in performance. Changing another #pragma option to clang increased performance by another 20%. But this PTX code did not use FMA multiply-and-add operations, and after adding another clang key, the additional performance improvement was about 20% [352]. The results in Table 25 include optimization.

In addition, scalability up to a large number of nodes in Crusher and Polaris was demonstrated, which is highly dependent on the basis types used.

TABLE 26. Strong and weak performance scaling (in TFLOPS) of the Dirac operator HISQ on Perlmutter and Frontier nodes

GPU in a node	Number of GPUs (for MI250X — number of GCDs)	Performance (strong scaling)	Performance (weak scaling)
A100-40GB	1	—	1.35746
	4	5.06892	4.53759
MI250X	1	—	0.92974
	2	1.63049	1.51439
	4	3.00675	2.89454
	8	4.69513	5.57654

For strong scaling, a global lattice was used 96^4; for weak scaling — local lattice 32^4. The table use data from [354].

In [353], using the still "pilot" state of the LUMI-G supercomputer (LUMI subcluster with MI250X GPU in the nodes), calculations were carried out using the well-known quantum chemical application CP2K with the authors' improved methods for the exchange part of the Hartree-ock method and correlation methods with periodic boundary conditions, but using a Gaussian basis, which makes it possible to use block-by-block sparsity. Good performance scalability and parallelization efficiency up to 32 LUMI-G nodes were obtained.

Lattice quantum chromodynamics (LQCD). In [354], the performance of the SIMULATeQCD application for quantum chromodynamics, available on GitHub, was studied when running on one or several nodes (multi-GPU servers) of cluster systems, including Perlmutter and Frontier (see Table 26). Although the MI250X performance data here, as in many other publications, is considered preliminary, the comparison made here with the performance of the A100-40GB is interesting (although it is clear that using more modern versions of ROCm and additional optimization of the program code the performance achieved may increase significantly in the coming times).

SIMULATeQCD uses CUDA or HIP codes as a backend. [354] provides performance data using up to 256 A100s and GCDs; We will limit ourselves here to data from benchmarks of the Dirac operator HISQ with scaling within one node (see Table 26).

Although the results for the MI250X were reported as preliminary, the performance achieved was lower than what the authors expected based on the MI250X specifications. In general, here, as a first approximation, we can say that one GCD is somewhat behind the A100 in terms of performance, while the full MI250X (2 GCDs) is somewhat ahead of it.

In [355], the performance on nodes of the Big Red 200 (4 A100-40GB per node) and Crusher (4 MI250X per node) supercomputers was compared using the well-known QUDA program for lattice quantum chromodynamics. Due to poor scaling with the number of nodes in both systems, we will only point out performance data for one node — a node with an MI250X (with 8 GCDs) was 16% faster than a node with an A100. The advantage of the MI250X in [355] was attributed to the memory-bound performance of QUDA (with the MI250X has 2×the memory bandwidth over that A100 model, see Table 21).

Computational fluid dynamics (CFD). We begin the analysis of data on the performance of MI250X and MI250 in CFD applications with AMD data [333] on the performance of MI250 in the standard (for OpenFOAM CFD application) HPC Motorbike benchmark (662.3 sec. on one MI250 and 209.84 sec. on four MI250).

In [356], for the numerical simulation of realistic combustion devices, a special PeleLMeX solver was used, the performance of which was studied on Crusher with MI250X and Summit with V100. With a small number of GPUs used, the performance of the MI250X was found to be one and a half times greater than that of the V100.

In [357], multi-particle collision dynamics calculations were carried out to particle-based description of hydrodynamic interactions using the Cabana 1.0-dev library based on Kokkos 3.5.00, using A100 and MI250. A comparison of multi-GPU servers containing 4 A100 or 4 MI250 showed that with smaller sizes of the calculated system, MI250 is 7% faster, and with a larger size, A100 is 19% faster (according to data from Figure 3 in [357]). This allows us to talk about the comparability of the performance of a full (with two GCD) MI250 and one A100.

A very interesting and relevant article comparing the performance of MI250X, MI100, A100 and V100 [325] concerns the solution of the Navier-Stokes equations for compressible flow using finite difference discretization — for wall bounded turbulent flows. There, the implementation of the STREAmS-2 application, portable to x86-64 CPUs, AMD and Nvidia GPUs, developed on the basis of the object-oriented (using Fortran 2008) application STREAmS-1, was considered.

To work with Nvidia GPUs, [325] uses CUDA Fortran tools (HPC SDK 22.11), and with AMD GPUs, HIPFORT (ROCm 4.5.2 on MI100 and ROCm 5.0.2 on MI250X). To port STREAmS code from CUDA Fortran to HIPFORT, [325] created its own PyconvertSTREAmS tools, since GPUFORT, according to the authors of [325], was rather at the research stage of development. But for optimization, the code obtained in PyconvertSTREAmS may also require subsequent manual modification.

Calculations with analysis of performance scaling on several nodes were carried out in [**325**] in *EuroHPC JU* clusters: for MI250X — on LUMI, for A100 — on Leonardo (see Table 23).

When comparing calculations on one GPU in MI250X, one GCD was used; data on the corresponding calculation times for two different calculation schemes in STREAmS-2 are shown above in Figure 20. The dimensions of the grid used in the calculations were chosen to be maximum for possible placement in the V100 memory capacity [**325**].

The data in Figure 20 indicates the similarity of the achieved performance between one GCD MI250X and V100; The MI100 was significantly slower than the V100, and the A100 was significantly faster than the GCD alone. As noted in [**325**], this does not correspond to the peak performance ratios (with FP64) of these GPUs, and there was a more thorough analysis of the execution times of individual kernels, which showed similar results for kernels with the A100 clearly leading in performance. Exmples of unexpected increases in GCD performance achieved with small changes in kernels are also given in [**325**].

An analysis in [**325**] using a roofline model (considering only the HBM) found that calculations on the GCD, as expected for CFD, were always memory-bound, and on the A100 were often in the computationally intensive region. Data movement for GCD was found to be significantly higher than for A100, meaning the A100 fetches registers and caches more often than GCD. In addition, [**325**] pointed out the large use of registers in the weighted essentially non-oscillatory (WENO) scheme used.

It is advisable to add additional comments to the above-described data from article [**325**]. As for the data on the higher performance of the V100 relative to the MI100, this seems rather natural (this was mentioned above in the section on MI100 performance). However, the V100 comes in two models [**185**] — with a memory capacity of 16 GB and (which appeared later) with 32 GB, and the V100 model used in [**325**] with 16 GB when working with a large grid size (but fits in 32 GB MI100) will simply be unacceptable.

Regarding the MI250X, firstly, the assumption of possibly weaker caching is consistent with the MI250X cache disadvantages noted above compared to the A100 (see also Table 20 and Table 21). Secondly, [**142**] points out a disadvantage of the AMD compiler (in ROCm v4.5.2) compared to the Nvidia compiler (in CUDA v11.0.3): CUDA uses significantly fewer registers per warp than the AMD compiler and provides much higher warp load on SM, which contributes to higher performance of Nvidia GPUs.

Table 27. NekRS performance data on a single GPU using CUDA and HIP for Nvidia and AMD GPUs respectively

System	Device	Relative Performance
Summit	V100-16GB	1.00
ThetaGPU	A100-40GB	1.57
Perlmutter	A100-40GB	1.62
Polaris	A100-40GB	1.62
Spock	MI100-32GB	0.84
Crusher	MI250X-64GB (1 GCD)	1.32

Thirdly, it is also actual to compare the performance of the full MI250X (the performance of the cheaper MI250 should probably be close) with the A100, which was not carried out in [**325**]. If we make a natural assumption about the good scalability of STREAmS-2 within the whole MI250X, then dividing the calculation time with MI250X in Figure 20 by two as an initial approximation, we will obtain calculation times comparable to the A100. In addition, one must keep in mind the lack of sufficient information about the level of optimization achieved when working with HIPORT. All this does not contradict the fact that STREAmS-2 achieves a higher percentage of the peak performance of the A100 compared to GCD.

Preprint [**325**] also obtained data showing the high scalability of STREAmS-2 performance — for example, efficiency of more than 80% when using up to 16 Leonardo nodes and up to 32 LUMI nodes, and good results on a larger number of nodes, giving for STREAmS-2 high potential for using in compressible fluid dynamics.

Argonne National Lab [**144**] conducted performance studies of the Nek5000/RS application on a range of supercomputers using MI250X, A100 and V100 GPUs on nodes, scaling from a single GPU to many nodes. Calculations were carried out using an upgraded version of Nek5000 for GPU operation (NekRS version 22.0) as part of the ECP ExaSMR project to generate virtual nuclear reactor simulation datasets with high-fidelity coupled physical models of reactor phenomena. Table 27 shows the performance of NekRS for singlerod simulation on a single GPU.

According to this table, a single GCD on a Crusher provides 1.32×the performance gain for solving the Navier-Stokes equation compared to a single V100 on a Summit. The A100 outperforms a single GCD, and overall, according to [**144**], the performance of a single GCD is about 85% of that of a single A100.

In [144], in particular, a comparison was made of performance scaling on Frontier (with ROCm 5.1.0 and Cray-mpich 8.1.17) and Crusher (with different versions of the SDK — ROCm 5.1.0, ROCm 5.2.0, Cray-mpich 8.1.16 and Cray-mpich 8.1.19). On Crusher, ROCm 5.1.0 gave performance 2–5% faster than ROCm 5.2.0, and performance on Frontier was better than on Crusher.

In [358], for large-eddy simulation problems, the performance of the NekRS and AMR-Wind codes for modeling flows in the atmospheric boundary layer using the Summit and Crusher supercomputers (with nodes containing V100 and MI250X, respectively) was studied in strong and weak scaling variants. For NekRS, a single GCD MI250X on a Crusher has been shown to provide performance comparable to a single V100 on a Summit.

In [142] presents the performance of the MI250X (compared to the MI100 and V100) using the NekBone-based HipBone benchmark (the HipBone proxy application was discussed above in the A100 and MI100 performance sections) using the libParanumal finite element library (developed within the ECP) with OCCA tools for abstraction between different parallel programming models — for example, OpenMP, OpenCL, CUDA, HIP and SYCL. HipBone, like Nekbone, uses a regular mesh of hexahedral elements.

The calculations in [142] were carried out on the Summit, Spock and Crusher computing systems (see Table 23) using CUDA 11.0.3, ROCm 4.5.0 and ROCm 4.5.2 in their nodes, respectively. Testing was conducted on a single GCD, allowing for potentially more than half of the MI250X's total compute capabilities and higher clock speeds than running the same workload on both GCDs simultaneously. However, there was no significant difference in performance between kernels running on one GCD or on both GCDs simultaneously in the MI250X. It was also found that the Nvidia compiler uses significantly fewer registers per warp compared to the AMD compiler, resulting in much higher warp utilization on the SM.

Article [142] also carried out a detailed study of strong and weak scaling in the used computing systems, which is not discussed here.

Report [359] indicates performance data on MI250X, MI100, V100 and A100 for the NekRS application and the HipBone proxy application. Computing systems Summit, Spock, Crusher and ThetaGPU were used for calculations (see Table 23).

The performance of the Poisson operator (as well as that of other operators in mesh methods — calculating the result of the operator's apply to a function at mesh points), which determines the calculation time of HipBone, for different degrees of the polynomial used for one GCD is 20–30% greater than for V100 (see Figure 43 in [359]).

All devices achieved the highest HipBone performance at the highest polynomial degree used (15) — 2101.4 GFLOPS in V100, 2135.2 GFLOPS in MI100, 2774.9 GFLOPS in a single GCD [142, 359]. For NekRS, one GCD on Crusher on different kernels gave from 57% to 88% of the performance of A100, in terms of total calculation time — 71% [359].

A number of publications have been discussed above regarding the performance of GPU-oriented software NekRS and subsequent NekBone and HipBone. All are focused on being portable across different types of GPUs, are heavily involved in ECP work, and use C++ tools (although NekBone has a version with CUDA Fortran [358], and all are based on Nek5000, which used Fortran).

But this direction using C++ has an alternative, which is also based on Nek5000 and is focused on portability to different types of accelerators, and it was also used in studies of the achieved performance using the GPUs discussed in the review. In [171], for the field of direct numerical simulations of turbulence with applications in sustainable shipping, a simulation of the flow around a lettner rotor (at Re = 30000) and its interaction with a turbulent boundary layer was carried out on clusters with multi-GPU servers in nodes using A100 and MI250X.

For this purpose, Neko software for working with unstructured meshes, also based on Nek5000, was used and modernized in [171]. Neko uses the object-oriented capabilities of modern Fortran standards to control memory allocation and provide multi-layered abstractions of the solver stack, enabling computation on a variety of architectures from conventional CPUs to different types of accelerators.

In [171], a device abstraction layer is used to manage device memory, data transfer, and kernel execution from Fortran, and CUDA and HIP backends are developed. The calculations here were performed on an Alvis cluster having by 4 A100-SXM4 per node (CUDA version 11.1.1 was used) with a Mellanox ConnectX-6 interconnect (2×400 Gb/s) — using OpenMPI 4.0.5, and on an HPE Cray EX cluster, having by 4 MI250X per node (version ROCm 4.5.2 was used) with an HPE Slingshot 10 interconnect — using Cray MPICH 8.1.14. On the hosts in Alvis gcc 10.2 was used, in HPE — Cray cce 13.0.

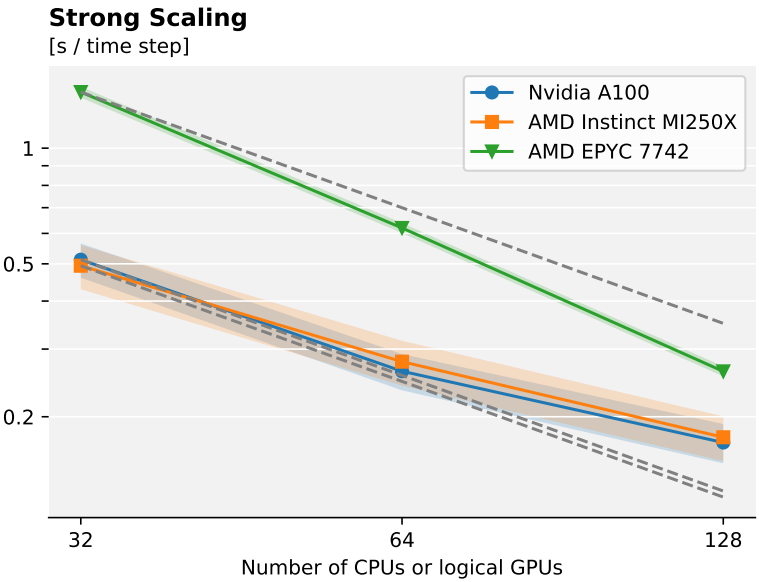


FIGURE 23. Performance in time per time step: shaded areas refer to standard deviation. The orange and blue lines represent GPU systems. There are two logical GPUs in each MI250X (picture from [171])

The hosts in both clusters had 512 GB of DDR4 memory (with two Intel Xeon Gold 6338 per Alvis node and an AMD EPYC 7A53 per HPE node).

Neko calculations were memory-bound, which the authors believe allows compute-intensive applications to benefit from the higher peak FP64 performance of the MI250X [171]. [171] concluded that the two GCDs in the MI250X match the two A100 in terms of performance. The average time per time step differs by less than 5% when comparing two A100 to one MI250X (see Figure 23). But this contrasts with data discussed above, for example [359], that a single MI250X GCD has performance closer to 71% of that of a single A100.

A number of publications that provide performance data for the MI250 or MI250X relate to magnetohydrodynamics. The article [360] presented a new Idefix code for nonrelativistic fluid dynamics and magnetohydrodynamics based on Kokkos, which in benchmarks for magnetohydrodynamics on a server with 4 MI250X showed performance 2.57 times faster than on a

server with 4 V100. Also [360] shows much higher power efficiency with the MI250 than with the CPU.

In [361], calculations were carried out using the Athena++ program (more precisely, the AthenaK code was used to work with the GPU) for general relativistic magnetohydrodynamics using Crusher nodes (with 4 MI250X per node) and Polaris (with 4 A100 per node). Compared to the same number of logical GPUs (i.e., comparing the GCD number for the MI250X to the A100 number), Polaris is, roughly speaking, a couple of times faster for any number of logical GPUs (see Figure 32 in [361]), and still scales good in both computing systems. This suggests that when comparing the A100 to the entire MI250X, they are competitive in performance.

Article [362] provides data on the performance of MI250X, MI100, A100-40GB and V100-16GB in cluster systems using PARTHENON-HYDRO — a mini-application (based on the astrophysical magnetohydrodynamics code Athena++), consisting of about 1.4 thousand lines of C++ for 1D, 2D and 3D compressible hydrodynamics on uniform and multilevel meshes.

The obtaining performance (here considered simply as some relative value) is 5.7 — for the MI250X (two GCDs, with ROCm 5.1.0); 4.2 — for A100 (with CUDA 11.5); 2.7 and 2.15 — for V100 and MI100, respectively. This means that the full MI250X is well ahead of the V100 and MI100 in performance, and faster than the A100 (although when calculated on a single GCD, the MI250X lags behind the A100). Data on the achieved strong and weak scalability in the systems used for calculations were also considered in [362], but this is not analyzed here.

Plasma physics. In principle, this area also belongs to CFD, but is highlighted here as a separate part, since there have been a number of publications in plasma physics that have used the MI250/MI250X and obtained interesting performance data.

Thus, in [363], a higher performance of the MI250 compared to the V100 was shown in solving the Vlasov kinetic equation for the dynamics of plasma charged particles.

In [364] studied the performance of the CGYRO code, which solves five-dimensional gyrokinetic-Maxwell equations describing the evolution of plasma microturbulence in magnetic fusion devices. To work with GPUs (MI250X in Frontier and A100 in Perlmutter), OpenACC tools were used (from HPE Cray Fortran for MI250X and from Nvidia Fortran for A100), as well as the cuFFT and hipFFT libraries for A100 and MI250X, respectively.

In the initial test, CGYRO on the MI250X node was significantly slower than on the A100 node, but after optimization it became faster with the MI250X. But in [364], calculations were compared with the same number of nodes and GPUs of these supercomputers; accordingly, the full MI250X of two GCDs turned out to be faster than the A100.

In [365], the calculation time of the HiPACE++ application for modeling plasma accelerators with a quasi-static particle-in-cell algorithm was compared for the A100-80GB and MI250X (one GCD). In this case, all the main data for the calculation is located entirely in the GPU memory during the calculation. CUDA 11.0 was used on the A100, and early test access to Crusher was used for calculations on GCD (probably with HIP). FFT was used in the calculations, and rocFFT, according to [365], then had poor performance at grid sizes other than a power of two. The calculation time on the GCD at two different types of calculation stages ranged from 0.7 to 1.6 and from 0.9 to 3.7 times compared to the time on the A100.

AMD cites the PIConGPU application, which also uses the particle-in-cell algorithm, as an example of a plasma physics application adapted to run on the MI200 GPU by using the ALPAKA (Abstraction Library for Parallel Kernel Acceleration) backend running on top of HIP/ROCm [366].

The particle-in-cell method is generally widely used in plasma physics applications, and it was also used in calculations using MI250X. In [367] 3D modeling of laser-matter interaction was studied on the massively parallel WarpX PIC code using GPUs on Frontier, Summit and Perlmutter (a similar study was also carried out in [368]).

Preprint [369] analyzed the performance of the linear iterative solver TFQMR using Kokkos (including a faster batch version) available in the PETSc (Portable Extensible Toolkit for Scientific Computing) library. This is interesting for plasma physics when working with the Landau collision operator. The performance of the MI250X and A100 here was evaluated using Perlmutter and Crusher cluster nodes.

The above-mentioned articles [365, 367, 368] are directly related to projects from ECP. Like many of the other publications used in Subsection 5.3.2.2, the latest data obtained in the newly appeared computing systems with GPU MI250/MI250X are used here; this data was often considered preliminary due to the possible rapid progress of new versions of the SDK from AMD.

Artificial intelligence tasks. Although AMD didn't put AI exactly first place of potential applications for the MI200, these GPUs were immediately use for AdI applications. Thus, in [28], the integration of HPC and deep learning was used: for the Monte Carlo application (for thermodynamic calculations in materials science), a set of surrogate deep learning models was developed and calculations were performed on many nodes of two supercomputers: Summit c V100 (using the CUDA stack) and Crusher with MI250X (using the ROCm stack). On different model sizes, one GCD was up to 15% faster than the V100 (and the whole MI250X was correspondingly faster by up to 2.3 times). Crusher also showed better performance scaling relative to Summit.

These GPUs have become used for AI tasks in a wide variety of fields that use AI. Thus, in [370], MI250X was used for mitochondrial segmentation tasks.

And after the deployment of the European supercomputer LUMI with MI250X, probably due to the use of different languages in Europe, publications on natural language processing on MI250X began to actively appear there, including using their own modified BERT models (see, for example, [371–374]).

5.4. AMD GPUs Summary

The above performance data for the MI200 GPUs — mostly the MI250 and MI250X — certainly suggests that these GPUs are somewhat competitive with the A100. Both AMD with ROCm and application developers are actively pushing forward to achieve higher performance.

It's clear that the performance data already reported often doesn't meet expectations based on the higher peak performance of these AMD GPUs compared to the A100 (meaning even a single GCD). Although in many cases the benchmarks were memory bound rather than compute-intensive in the roofline model, this still does not correlate with the many performance gaps relative to the A100, as well as the sometimes identifiable performance "sinks" of the MI250X (see, for example, data [328]).

In a number of publications, this is associated with the insufficiently large cache size (mainly L1) relative to the A100, and the use of too many registers in the codes generated by AMD compilers compared to Nvidia CUDA tools, although in some cases the reasons for insufficient performance compared to expected performance remain unclear. The existing comments regarding AMD's SDK software correlate with the clear indication in many of the above publications that the performance data obtained is preliminary, therefore suggesting improvements due to the rapid development of the SDK.

Since there are also data on higher performance of MI250/MI250X, the fact that the comparative performance of these GPUs relative to the A100 is not sufficiently predictable at this point in time should be considered important.

Here one cannot try to be based on a "statistical analysis" of comparative performance data, but must be based on data for specific applications and objects of study (which, to a first approximation, are also available in this review), as well as on similar (in purely mathematical sense) methods used, and take into account the already discovered shortcomings of MI250/MI250X and their SDK (the latter may possibly be corrected in new versions).

But it is necessary to keep in mind the possible opposite conclusions about the efficiency of the MI200 when taking into account comparative prices and TCO, and accordingly choose to compare the full MI250/MI250X or individual GCDs with the A100.

At the time of writing this review, the MI300 and the EI Capitans supercomputer are expected to appear (in 2023, at the Lawrence Livermore National Laboratory (USA)), with the integration of Zen 4 architecture CPUs into the MI300. This confirms AMD's likely progress in GPUs and supercomputers in the near future. **[375]** pointed out the continued small size of the L1 cache — this could be a potential weak point of the MI300.

In relation to HPC, it is worth pointing out AMD's great focus on this area compared to Nvidia's greater focus on AI, which is evident in the new supercomputers included in the Top500.

Conclusion

In this review, the advantages and possible disadvantages of GPUs were discussed above. As for specific performance data, a lot has been said about them above, and it is advisable to combine them with cost indicators (including energy consumption indicators), which are not discussed here. What follows is a mostly general summary of a slightly different, more general and/or briefer nature.

1. There is competition among modern GPU developers in terms of performance and other indicators, including not only from the USA, but possibly also from China; a European accelerator is also expected, which can also be used on exascale supercomputers. Accordingly, current Nvidia GPUs near-monopoly will likely diminish little by little.

2. The general technological direction of building GPUs is the use of multi-chip technologies (chiplets), which were probably first actively used by AMD [376]. This enables performance scalability that can be implemented in a cost-effective manner with currently negligible latency increases, including for interconnects between compute cores.
3. The general trend may be the integration of CPU and GPU (AMD MI300, Intel Falcon Shore, Nvidia GH200), as well as several GPUs within one model (as in AMD MI200 and Intel PVC).
4. The ultra-fast development of GPUs leads to the fact that attempts to standardize any hardware components are still being implemented to a limited extent. Thus, the use of the OAM form factor standard in MI200, PVC and BR100 is combined with Nvidia's use of its own SXM form factors, which looks quite understandable in the light of Nvidia's supply of its own ready-to-run computing systems — from multi-GPU servers to clusters.

   Standardization is even more difficult for GPU interconnects — all of the GPUs reviewed used their own, different selection from the competition. Here, the reason for the orientation not towards standards is rather the competitive struggle for leadership in performance.
5. The general direction is to expand the use of multi-GPU systems for AI and HPC tasks. In the corresponding servers, it is necessary to use two processors containing dozens of processor cores, which include not only Intel Xeon and AMD EPYC, but also ARM processors (the classic option would be Nvidia Grace or GH200). An alternative may be to use one server processor containing over fifty cores (both AMD and Intel offer such CPUs today).

   Traditional for HPC quantum chemistry problems previously used GPUs primarily for calculations in plane wave basis sets or computationally more complex methods taking into account electron correlation, where the bulk of the calculation time was spent on general mathematical methods. But for widespread quantum chemistry problems, performance is limited by the time it takes to calculate two-electron integrals in a Gaussian basis (in the HF and DFT methods). Here, high efficiency of GPU calculations, which gives good performance scalability when working with multiple GPUs, was achieved only very recently (the corresponding data is presented in the review). This makes it possible to more actively use multi-GPU systems also for quantum chemistry problems.

6. As for the BR100, they may be relevant primarily to software developers, including in the scientific field (since there are practically no applications running on the BR100) — and, perhaps, especially in countries where program developers and GPU consumers have significant financial restrictions. However, the absence of the FP64 format in the BR100, traditional in the classical HPC field, will likely lead to the BR100's main focus on AI tasks. But the prospects for the use of BR100 are unclear due to US sanctions.

7. Given certain development delays in Intel's semiconductor technology and delays in the start of shipments of Ponte Vecchio compared to Intel's original plans, and the emerging need to move to work with the DPC++ /oneAPI software model, it can be assumed that much faster progress with GPUs by Intel can be expected after the appearance of the Falcon Shores GPU or a subsequent x86-integrated device (rather after the implementation of the new promising 18A technology).

   Intel's most striking contribution to the development of GPU applications today seems to be the development and support of the DPC++ programming model.

8. There is no doubt that the adoption of Nvidia's H100 GPU will continue to expand rapidly (with faster growth of H100 using for AI tasks). Both the already released and the upcoming Nvidia H100 GPUs (GH200 also actually contain the H100) will be very relevant for AI and HPC tasks. But the most powerful leap forward in performance scalability seems to be the ability to build clusters with the H100 using NVLink networks.

   The advantage of Nvidia hardware primarily for AI tasks is the delivery of AI-ready DGX server systems and DGX SuperPOD clusters, which are at the supercomputing level. Although such systems can be used for HPC tasks, applications ready for them are needed. Another advantage of the H100 is its huge set of SDKs for HPC and AI that interact effectively and complement each other.

9. GPUs from AMD, as an alternative to Nvidia GPUs, seem to be most relevant at present, primarily for calculations on applications that

have already shown high performance on the AMD MI200, or are expected to do so in the very near future, as well as a field for the development of new software, including porting existing applications. In terms of potential application use in the near future, this could make sense for any HPC and AI fields.

The use of these GPUs will obviously expand primarily in science and in the activities of software developers. There is a certain degree of comparability in performance with the Nvidia GPUs reviewed, which means that price points data are of increased importance.

10. Taking into account current trends, it can be assumed that the use of universal software development tools on GPUs will expand, rather than those focused on a specific manufacturer's architecture.

References

[1] *Top500 the list*, The Green500 ranking, 61st edition, 2023. URL ↑307

[2] W. Tschudi, T. Xu, D. Sartor, J. Stein. *High-performance data centers. A research roadmap*, LBNL-53483, 2004, 53 pp. URL ↑307

[3] T. Maltenberger, I. Ilic, Tolovski I, T. Rab. "Evaluating multi-GPU sorting with modern interconnects", *SIGMOD'22: Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA, June 12–17, 2022), ACM, New York, 2022, ISBN 978-1-4503-9249-5, pp. 1795–1809. doi ↑307, 324, 355

[4] *Top500 the list*, The Top500 ranking, 61st edition, 2023. URL ↑308, 309, 421

[5] M. B. Kuzminsky. "Modern server ARM processors for supercomputers: A64FX and others. Initial data of benchmarks", *Program Systems: Theory and Applications*, **13**:1(52) (2022), pp. 131–194. doi URL ↑308, 312, 365

[6] J. Gao, F. Zheng, Qi F, Ding Y, H. Li, H. Lu, W. He, H. Wei, L. Jin, X. Liu, D. Gong, F. Wang, Y. Zheng, H. Sun, Z. Zhou, Y. Liu, H. You. "Sunway supercomputer architecture towards exscale computing: analysis and practice", *Science China Information Sciences*, **64**:4 (2021), id. 141101, 21 pp. doi URL ↑308

[7] J. Selig. *The cerebras software development kit: A technical overview*, Cerebras systems Inc., 2022, 8 pp. URL ↑308

[8] *Andromeda, a 13.5 Million Core AI Supercomputer*, a section on the Cerebras company site, 2024. URL ↑308

[9] *Top500 the list*, List Statistics of TOP500, 61st edition, 2023. URL URL ↑308

[10] T. P. Morgan. *Chip roadmaps unfold, crisscrossing and interconnecting, at AMD*, The Next Platform, Stackhouse Publishing, 2022. URL ↑308, 313

[11] A. Shah. *Intel Reiterates Plans to Merge CPU, GPU High-performance Chip Roadmaps*, Tabor network, HPCwire, 2022. URL ↑308

[12] T. P. Morgan. *The Increasingly Graphic Nature Of Intel Datacenter Compute*, The Next Platform, Stackhouse Publishing, 2022. URL ↑308

[13] J. Evans. "Nvidia Grace", *2022 IEEE Hot Chips 34 Symposium (HCS)* (Cupertino, CA, USA, 21–23 August 2022), IEEE, 2022, pp. 1–20. doi ↑308, 313, 314, 384, 385

[14] A. C. Elster, T. A. Haugdahl. "Nvidia Hopper GPU and Grace CPU Highlights", *Computing in Science and Engineering*, **24**:2 (2022), pp. 95–100. doi ↑308, 314, 386

[15] J. Evans. *Inside Grace*, Featured Playlists, GPU Technology Conference (GTC), Nvidia On-Demand, Nvidia, 2022. URL ↑308, 314

[16] *CUDA C++ Programming Guide*, Release 12.4, Nvidia, 2024, 544 pp. URL ↑309, 315, 316, 317, 347, 349, 350, 352, 359

[17] *Ampere Tuning Guide*, Release 12.4, Nvidia, 2024, 22 pp. URL ↑309, 319

[18] Z. Zhang, S. Jiao, J. Li, W. Wu, L. Wan, X. Qin, W. Hu, J. Yang. "KSSOLV-GPU: An efficient GPU-enabled MATLAB toolbox for solving the Kohn-Sham equations within density functional theory in plane-wave basis set", *Chinese Journal of Chemical Physics*, **34**:5 (2021), pp. 552–564. doi ↑310

[19] P. Giannozzi, O. Baseggio, P. Bonfà, D. Brunato, R. Car, I. Carnimeo, C. Cavazzoni, de Gironcoli S., P. Delugas, Ruffino . F., A. Ferretti, N. Marzari, I. Timrov, A. Urru, S. Baroni. "Quantum ESPRESSO toward the exascale", *The Journal of chemical physics*, **152**:15 (2020), id. 154105. doi ↑310

[20] He Lizhong. *The pace of GPU localization is accelerating, and emerging teams continue to emerge*, Industry Brief Report, Capital Securities, Beijing, 2022, 15 pp. (in Chinese). URL ↑311

[21] J. Bispo, J. Barbosa, P. Silva, C. Morales, M. Myllykoski, P. Ojeda-May, M. Bialczak, M. Uchroński, odarczyk A. Wł, P. Wauligmann, E. Krishnasamy, S. Varrette, S. Lührs. *Best Practice Guide: Modern Accelerators*, ed. Shoukourian H., PRACE, 2021, 111 pp. URL ↑311, 312, 396

[22] J. Finkelstein, J. S. Smith, S. M. Mniszewski, K. Barros, C. F. A. Negre, E. H. Rubensson, A. M. N. Niklasson. "Quantum-based molecular dynamics simulations using tensor cores", *Journal of Chemical Theory and Computation*, **17**:10 (2021), pp. 6180–6192. doi ↑311

[23] S. Posey, J. Luitjens, O. Hennigh, S. Oberlin. "GPU-based HPC and AI developments for CFD" (Maui, Hawaii, USA, July 11-15, 2022), 2022, id. ICCFD11-3803, 5 pp. URL ↑311

[24] R. Schade, T. Kenter, H. Elgabarty, M. Lass, O. Schütt, A. Lazzaro, H. Pabst, S. Mohr, J. Hutter, T. D. Kühne, C. Plessl. "Towards electronic structure-based *ab-initio* molecular dynamics simulations with hundreds of millions of atoms", *Parallel Computing*, **111** (July 2022), id. 102920, 11 pp. doi ↑311

[25] O. Terzo, J. Martinovič (eds.). *HPC, Big Data, and AI Convergence Towards Exascale: Challenge and Vision*, 1st ed., CRC Press, 2022, ISBN 9781003176664, 322 pp. ↑312

[26] M. Nowicki, Górski Ł., P. Bała. "PCJ Java library as a solution to integrate HPC, Big Data and Artificial Intelligence workloads", *Journal of Big Data*, **8**:1 (2021), pp. 1–21, id. 62. ↑312

[27] F. Yin, F. Shi. "A comparative survey of Big Data computing and HPC: from a parallel programming model to a cluster architecture", *International Journal of Parallel Programming*, **50**:1 (2022), pp. 27–64. ↑312

[28] J. Yin, F. Wang, M. Shankar. "Strategies for integrating deep learning surrogate models with HPC simulation applications", *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (Lyon, France, 2022), IEEE, 2022, ISBN 978-1-6654-9747-3, pp. 01–10. ↑312, 437

[29] S. R. Sukumar, J. A. Balma, C. D. Rickett, K. J. Maschhoff, J. Landman, C. R. Yates, A. G. Chittiboyina, Y. K. Peterson, A. Vose, K. Byler, J. Baudry, I. A. Khan. "The convergence of HPC, AI and Big Data in rapid-response to the COVID-19 pandemic", *Driving Scientific and Engineering Discoveries Through the Integration of Exeriment, Big Data, and Modeling and Simulation: 21st Smoky Mountains Computational Sciences and Engineering, SMC 2021, Virtual Event, October 18-20, 2021, Revised Selected Papers*, Communications in Computer and Information Science, vol. **1512**, 2022, ISBN 978-3-030-96497-9, pp. 157-172. ↑312

[30] J. Ejarque, R. M. Badia, L. Albertin, f, G. Aloisio, E. Baglione, Y. Becerra, S. Boschert, J. R. Berlin, A. D'Anca, D. Elia, F. Exrtier, S. Fiore, J. Flich, A. Folch, S. J. Gibbons, N. Koldunov, F. Lordan, S. Lorito, Løvholt F., J. Macías, M. Volpe. "Enabling dynamic and intelligent workflows for HPC, data analytics, and AI convergence", *Future Generation Computer Systems*, **134** (September 2022), pp. 414–429. ↑312

[31] N. Ihde, P. Marten, A. Eleliemy, G. Poerwawinata, P. Silva, I. Tolovski, Ciorba . M., T. Rabl. "A survey of Big Data, High Performance Computing, and Machine Learning benchmarks", Technology Conference on Performance Evaluation and Benchmarking, Lecture Notes in Computer Science, vol. **13169**, Springer, Cham, 2021, ISBN 978-3-030-94436-0, pp. 98–118. ↑312

[32] *High-Performance Deep Learning Project (HiDL)*, NOWLAB: Network Based Computing Lab, Ohio state university. ↑312

[33] *High-Performance Big Data Project (HiBD)*, NOWLAB: Network Based Computing Lab, Ohio state university. ↑312

[34] W. Jeon, G. Ko, J. Lee, H. Lee, D. Ha, W. W. Ro. "Deep learning with GPUs", *Advances in Computers*, **122** (2021), pp. 167–215. ↑312

[35] M. Hong, L. Xu. "Biren BR100 GPGPU: Accelerating Datacenter Scale AI Computing", 2022 IEEE Hot Chips 34 Symposium (HCS) (Cupertino, CA, USA), 2022, pp. 1–22. ↑312, 319, 321, 322, 323, 324, 325, 326

[36] A. Shilov. *Russian Company Taps China's Zhaoxin x86 CPU to Replace AMD, Intel CPUs*, Tom's Hardware, Future US, New York, 2022. ↑312

[37] H. Shang, F. Li, Y. Zhang, L. Zhang, Y. Fu, Y. Gao, Y. Wu, X. Duan, R. Lin, X. Liu, Y. Liu, D. Chen. "Exreme-scale *ab initio* quantum Raman spectra simulations on the leadership HPC system in China", *SC'21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ACM, New York, November 2021, ISBN 978-1-4503-8442-1, id. 6, 13 pp. ↑313

[38] D. Schneider. "The Exascale Era is Upon Us: The Frontier supercomputer may be the first to reach 1 000 000 000 000 000 000 operations per second", *IEEE Spectrum*, **59**:1 (January 2022), pp. 34–35. ↑313

[39] J. Dongarra, A. Geist. *Report on the Oak Ridge National Laboratory's Frontier System*, Technical Report ICL-UT-22-05, Oak Ridge National Laboratory, 2022, Accessed 15.10.2023. ↑313

[40] *Frontier Spec Sheet*, Oak Ridge National Laboratory, UT-Battelle, 2019, 4 pp. ↑313

[41] *GPU nodes — LUMI-G*, Hardware documentation, LUMI (Large Unified Modern Infrastructure) consortium. ↑313, 400, 403, 404, 406, 407, 408

[42] G. S. Markomanolis, A. Alpay, J. Young, M. Klemm, N. Malaya, A. Esposito, J. Heikonen, S. Bastrakov, A. Debus, T. Kluge, K. Steiniger, J. Stephan, R. Widera, M. Bussmann. "Evaluating GPU programming models for the LUMI supercomputer", *Supercomputing Frontiers*, Lecture Notes in Computer Science (Asian Conference on Supercomputing Frontiers), vol. **13214**, Springer, Cham, 2022, ISBN 978-3-031-10419-0, pp. 79–101. ↑313, 366, 373, 396, 408, 409, 410, 414, 416

[43] *Aurora*, Argonne Leadership Computing Facility, Argonne National Laboratory. ↑313

[44] O. Peckham. *LRZ announces new phase of SuperMUC-NG Supercomputer with Intels Ponte Vecchio GPU*, Tabor network, HPCwire, 2021. ↑313

[45] J. H. Kwack, J. Tramm, C. Bertoni, Y. Ghadar, B. Homerding, E. Rangel, C. Knight, S. Parker. "Evaluation of performance portability of applications and mini-apps across AMD, Intel and Nvidia GPUs", *2021 International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (14 November 2021, St. Louis, MO, USA), IEEE, 2021, ISBN 978-1-6654-2439-4, pp. 45–56. ↑313, 413

[46] *HPE Cray Supercomputing EX*, Hewlett Packard Enterprise Development LP, 2024. ↑313, 408

[47] C. Bertoni, S. Parker. *Aurora overvew*, ALCF SDL Workshop (October 6, 2022), 2022, 20 pp. ↑313, 337

[48] T. P. Morgan. *The NVSwitch Fabric That Is The Hub Of The DGX H100 SuperPOD*, The Next Platform, Stackhouse Publishing, 2022. ↑313

[49] A. Ishii, R. Wells. "The Nvlink-Network switch: Nvidia's switch chip for high communication-bandwidth superpods", *2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 Aug., 2022, Cupertino, CA, USA), IEEE, 2022, ISBN 978-1-6654-6028-6, pp. 1–23. ↑313

[50] A. Eassa, A. Ishii, R. Wells. *Upgrading Multi-GPU Interconnectivity with the Third-Generation Nvidia NVSwitch*, Technical blog, Nvidia developer, 2022. ↑313

[51] *BR100 series general purpose GPU chip*, Biren Technology, Shanghai, 2023. ↑314, 321, 322, 326

[52] M. Andersch, G. Palmer, R. Krashinsky, N. Stam, V. Mehta, G. Brito, S. Ramaswamy. *Nvidia Hopper Architecture In-Depth*, Technical blog, Nvidia developer, 2022. ↑314

[53] P. Alcorn. *From Opteron to Milan: Crusher Supercomputer Comes Online With New AMD CPUs and MI250X GPUs*, Tom's Hardware, Future US, New York, 2022. ↑314

[54] *Intel Xeon CPU Max series product overview*, Intel, 2023, Accessed 15.10.2023. ↑314

[55] *Accelerator Processor Stream*, European Processor Initiative, 2022. ↑314

[56] *EPI EPAC1.0 RISC-V test chip samples delivered*, News, European Processor Initiative, 2021. ↑314

[57] M. Kovač, P. Notton, D. Hofman, J. Knezović. "How Europe is preparing its core solution for exascale machines and a global, sovereign, advanced computing platform", *Mathematical and Computational Applications*, **25**:3 (2020), pp. 46. ↑314

[58] *HIP Programming Guide*, Version 5.0, 2023, Accessed 15.10.2023. ↑315, 316, 317, 358, 409

[59] *OpenMP Application Programming Interface*, Version 5.2, OpenMP Architecture Review Board, 2021, 669 pp. ↑315

[60] *Khronos OpenCL Registry*, Formatted specifications and other related documentation, Khronos Group. ↑315

[61] *SYCL 2020 Specification*, rev. 6, Khronos Group, 2022, 585 pp. ↑315

[62] *DPC++ Part 1: An Introduction to the New Programming Model*, Intel (Accessed 15.10.2023). ↑315

[63] N. N. Bavarsad, H. M. Makrani, H. Sayadi, L. Landis, S. Rafatirad, H. Homayoun. "HosNa: A DPC++ benchmark suite for heterogeneous architectures", *2021 IEEE 39th International Conference on Computer Design (ICCD)* (24–27 October 2021, Storrs, CT, USA), IEEE, 2021, ISBN 978-1-6654-3219-1, pp. 509–516. ↑315

[64] C. Trott, L. Berger-Vergiat, D. Poliakoff, S. Rajamanickam, D. Lebrun-Grandie, J. Madsen, N. Al Awar, M. Gligoric, G. Shipman, G. Womeldorff. "The Kokkos EcoSystem: comprehensive performance portability for high performance computingà", *Computing in Science & Engineering*, **23**:5 (2021), pp. 10–18. ↑315

[65] C. R. Trott, D. Lebrun-Grandié, D. Arndt, J. Ciesko, V. Dang, N. Ellingwood, R. Gayatri, E. Harvey, D. S. Hollman, D. Ibanez, N. Liber, J. Madsen, J. Miles, D. Poliakoff, A. Powell, S. Rajamanickam, M. Simberg, D. Sunderland, B. Turcksin, J. Wilke. "Kokkos 3: Programming model extensions for the exascale era", *IEEE Transactions on Parallel and Distributed Systems*, **33**:4 (2021), pp. 805–817. ↑315

[66] S. Moore. *The state of the LAMMPS KOKKOS package*, SAND2021-9785C, Sandia National Lab, Albuquerque, NM, 2021, Accessed 15.10.2023. ↑315

[67] Y. Ghadar, T. Applencourt, B. Homerding, K. Harms, J. Hammond. *SYCL Programming Model for Aurora*, 2020 ECP Annual Meeting, 2020. ↑316, 329, 338

[68] R. Van Oostrum, N. Chalmers, D. Mc Dougall, P. Bauman, N. Curtis, N. Malaya, N. Wolfe. *AMD GPU Hardware Basics*, Frontier Application Readiness Kick-Off Workshop, 2019, 55 pp. ↑316

[69] *Intel oneAPI GPU Optimization Guide Release 2022.3*, Intel (Accessed 15.10.2023). ↑317, 329, 331, 332, 333, 334, 335, 336, 338

[70] D. Khudia, J. Huang, P. Basu, S. Deng, H. Liu, J. Park, M. Smelyanskiy. *Fbgemm: Enabling high-performance low-precision deep learning inference*, 2021, 5 pp. arXiv 2101.05615 ↑318

[71] R. Carrasco, R. Vega, C. A. Navarro. "Analyzing GPU tensor core potential for fast reductions", *2018 37th International Conference of the Chilean Computer Science Society (SCCC)* (05–09 November 2018, Santiago, Chile), IEEE, 2018, ISBN 9781538692349, pp. 1–6. ↑318

[72] G. Gupta. *Using Tensor Cores for Mixed-Precision Scientific Computing*, Technical blog, Nvidia developer, 2019. ↑318

[73] *Nvidia A100 Tensor Core GPU Architecture*, V1.0, Nvidia, 2020, 82 pp. ↑318, 333, 344, 345, 346, 347, 350, 351, 352, 353, 355

[74] *754-2019 — IEEE Standard for Floating-Point Arithmetic*, IEEE Std 754-2019, Revision of IEEE 754-2008, 2019, 84 pp. ↑319

[75] D. Kalamkar, D. Mudigere, N. Mellempudi, D. Das, K. Banerjee, S. Avancha, D. T. Vooturi, N. Jammalamadaka, J. Huang, H. Yuen, J. Yang, J. Park, A. Heinecke, E. Georganas, S. Srinivasan, A. Kundu, M. Smelyanskiy, B. Kaul, P. Dubey. *A study of BFLOAT16 for deep learning training*, 2019, 10 pp. arXiv 1905.12322 ↑319

[76] D. Stosic, P. Micikevicius. *Accelerating AI Training with Nvidia TF32 Tensor Cores*, Technical blog, Nvidia developer, 2021. ↑319

[77] P. Micikevicius, D. Stosic, N. Burgess, M. Cornea, P. Dubey, R. Grisenthwaite, S. Ha, A. Heinecke, P. Judd, J. Kamalu, N. Mellempudi, S. Oberman, M. Shoeybi, M. Siu, H. Wu. *Fp8 formats for deep learning*, 2022, 9 pp. arXiv 2209.05433 ↑319

[78] *Nvidia H100 Tensor Core GPU Architecture*, Includes final GPU / memory clocks and final TFLOPS performance specs, V1.04, Nvidia, 2023, 71 pp. ↑319, 333, 346, 347, 348, 349, 350, 379, 381, 382, 389, 390, 391

[79] W. Sun, A. Li, T. Geng, S. Stuijk, H. Corporaal. "Dissecting tensor cores via microbenchmarks: latency, throughput and numerical behaviors", *IEEE Transactions on Parallel and Distributed Systems*, **34**:1 (2022), pp. 246–261. ↑319, 365

[80] M. Lehmann, M. J. Krause, G. Amati, M. Sega, J. Harting, S. Gekle. "Accuracy and performance of the lattice Boltzmann method with 64-bit, 32-bit, and customized 16-bit number formats", *Physical Review E*, **106**:1 (2022), id. 015308. ↑320

[81] J. Domke, K. Matsumura, M. Wahib, H. Zhang, K. Yashima, T. Tsuchikawa, Y. Tsuji, A. Podobas, S. Matsuoka. Double-precision FPUs in high-performance computing: an embarrassment of riches? *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (20–24 May 2019, Rio de Janeiro, Brazil), IEEE, 2019, ISBN 978-1-7281-1246-6, pp. 78–88. ↑320

[82] R. Schade, T. Kenter, H. Elgabarty, M. Lass, O. Schütt, A. Lazzaro, H. Pabst, S. Mohr, J. Hutter, T. D. Kühne, C. Plessl. "Towards electronic structure-based *ab-initio* molecular dynamics simulations with hundreds of millions of atoms", *Parallel Computing*, **111** (2022), id. 102920, 11 pp. ↑320

[83] R. Schade, T. Kenter, H. Elgabarty, M. Lass, T. D. Kühne, C. Plessl. *Breaking the exascale barrier for the electronic structure problem in* ab-initio *molecular dynamics*, 2022, 6 pp. arXiv 2205.12182 ↑320

[84] V. W. Yu, M. Govoni. *GPU acceleration of large-scale full-frequency GW calculations*, 2022, 54 pp. arXiv 2203.05623 ↑320

[85] J. J. Eriksen. "Efficient and portable acceleration of quantum chemical many-body methods in mixed floating point precision using OpenACC compiler directives", *Molecular Physics*, **115**:17–18 (2017), pp. 2086–2101. ↑320

[86] D. Ruda, S. Turek, D. Ribbrock, P. Zajac. *Very fast FEM Poisson solvers on lower precision accelerator hardware*, ECCOMAS Congress 2022 (5–9 June 2022, Oslo, Norway), 2022, 24 pp. URL ↑320

[87] H. Ootomo, R. Yokota. "Recovering single precision accuracy from Tensor Cores while surpassing the FP32 theoretical peak performance", *The International Journal of High Performance Computing Applications*, **36**:4 (2022), pp. 475–491. ↑320

[88] A. Jain, N. Sharma. "Accelerated AI inference at CNN-based machine vision in ASICs: A design approach", *ECS Transactions*, **107**:1 (2022), pp. 5165. ↑320

[89] B. Gallet, M. Gowanlock. *Computing double precision Euclidean distances using GPU tensor cores*, 2022, 10 pp. arXiv 2209.11287 ↑320, 350

[90] J. Domke, E. Vatai, A. Drozd, P. T. Chen, Y. Oyama, L. Zhang, S. Salaria, D. Mukunoki, A. Podobas, M. T. Wahib, S. Matsuoka. Matrix engines for high performance computing: A paragon of performance or grasping at straws? *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (17–21 May 2021, Portland, OR, USA), IEEE, 2021, ISBN 978-1-6654-4066-0, pp. 1056–1065. ↑320, 351, 362

[91] H. Tan, R. Yan, L. Yang, L. Huang, L. Xiao, Q. Yang. "Efficient multiple-precision and mixed-precision floating-point fused multiply-accumulate unit for HPC and AI applications", *Algorithms and Architectures for Parallel Processing*, 22nd International Conference ICA3PP 2022 (Copenhagen, Denmark, October 10–12, 2022), Lecture Notes in Computer Science, vol. **13777**, Springer Nature Switzerland, Cham, 2023, ISBN 978-3-031-22676-2, pp. 642–659. ↑321

[92] *Exlusive interview with Biren Technology executives: Deconstructing the company's first 7nm GPU*, Semiconductor Industry Observation by MooreElite.com (Hefei) Co., 2022 (in Chinese). ↑321, 324, 325, 326

[93] *Nvidia A100 Tensor Core GPU Datasheet*, V1.0, Nvidia, 2020, 3 pp. ↑323, 324, 325

[94] J. Choquette, E. Lee, R. Krashinsky, V. Balan, B. Khailany. "3.2 The A100 Datacenter GPU and Ampere Architecture", *2021 IEEE International Solid-State Circuits Conference (ISSCC)* (13–22 February 2021, San Francisco, CA, USA), IEEE, 2021, ISBN 9781728195506, pp. 48–50. ↑323, 324, 326, 351, 352, 353, 355

[95] *Nvidia A100 tensor core GPU architecture*, V1.0, Nvidia, 2020, 82 pp. ↑324

[96] M. Hassanpour, M. Riera, A. González. "A survey of near-data processing architectures for neural networks", *Machine Learning and Knowledge Extraction*, **4**:1 (2022), pp. 66–102. ↑324

[97] J. Gómez-Luna, Y. Guo, S. Brocard, J. Legriel, R. Cimadomo, G. F. Oliveira, G. Singh, O. Mutlu. *An experimental evaluation of machine learning training on a real processing-in-memory system*, 2022, 21 pp. arXiv 2207.07886 ↑324

[98] D. Niu, S. Li, Y. Wang, W. Han, Z. Zhang, Y. Guan, T. Guan, F. Sun, F. Xue, L. Duan, Y. Fang, H. Zheng, X. Jiang, S. Wang, F. Zuo, Y. Wang, B. Yu, Q. Ren, Y. Xie. "184QPS/W 64Mb/mm^{2}3D logic-to-DRAM hybrid bonding with process-near-memory engine for recommendation system", *IEEE International Solid-State Circuits Conference (ISSCC)* (20–26 February 2022, San Francisco, CA, USA), IEEE, 2022, pp. 1–3. ↑324

[99] *BiLi 106M*, Product details, Biren Technology, Shanghai, 2020–2023. ↑323, 325

[100] *BiLi 106B, 106C*, Product details, Biren Technology, Shanghai, 2020–2023. ↑323, 325

[101] R. Blankenship, M. Wagh. *Introducing the CXL 3.1 Specification*, Compute express link consortium, 2022, 27 pp. ↑324

[102] T. Coughlin. "Digital storage and memory", *Computer*, **55**:1 (2022), pp. 20–29. ↑324

[103] *Nvidia A100 Tensor Core GPU Datasheet*, Nvidia, 2021, 3 pp. ↑324

[104] *Ampere Tuning Guide*, Release 12.4, Nvidia, 2024, 22 pp. ↑324

[105] *Server/OAI*, Wiki page, Open computers project. ↑325

[106] *Nvidia DGX A100*, Datasheet, Nvidia, 2023, 2 pp. ↑325

[107] T. P. Morgan. *China launches the inevitable indigenous GPU*, The Next Platform, Stackhouse Publishing, 2022. URL ↑326

[108] *BIRENSUPA software development platform*, Product details, Biren Technology, Shanghai, 2023. URL ↑326

[109] *MLPerf inference: datacenter benchmark suite results*, MLCommons. URL ↑326, 327

[110] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, J. Carole-Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka, C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, Y. Zhou. "Mlperf inference benchmark", *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)* (30 May 2020–03 June 2020, Valencia, Spain), IEEE, 2020, ISBN 978-1-7281-4661-4, pp. 446–459. doi ↑326

[111] M. H. Saad, S. Hashima, W. Sayed, E. H. El-Shazly, A. H. Madian, M. M. Fouda. "Early diagnosis of COVID-19 images using optimal CNN hyperparameters", *Diagnostics*, **13**:1 (2023), id. 76. doi ↑326

[112] J. Devlin, W. Ming-Chang, K. Lee, K. Toutanova. "BERT: Pre-training of deep bidirectional transformers for language understanding", *Human Language Technology: Conference of the North American Chapter of the Association of Computational Linguistics*. V. 1, NAACL-HLT 2019 (June 2–June 7, 2019, Minneapolis, Minnesota, USA), ACL, 2019, ISBN 978-1-950737-13-0, pp. 4171–4186. URL ↑327

[113] *Nvidia TensorRT, an SDK for high-performance deep learning inference*, Web site, Nvidia developer, Nvidia. URL ↑328

[114] D. Blythe. "The X^e GPU architecture", *2020 IEEE Hot Chips 32 Symposium (HCS)* (16–18 August 2020, Palo Alto, CA, USA), IEEE, 2020, ISBN 978-1-7281-7129-6, pp. 1–27. doi ↑329, 330, 336

[115] D. Blythe. "X^{e}HPC Ponte Vecchio", *2021 IEEE Hot Chips 33 Symposium (HCS)* (22–24 August 2021, Palo Alto, CA, USA), IEEE, 2021, ISBN 978-1-6654-1397-8, pp. 1–34. doi ↑329, 331

[116] *Intel data center GPU Max series product brief*, Intel (Accessed 15.10.2023). URL ↑329, 330, 331, 335, 336, 337, 341

[117] *Intel data center GPU flex series product brief*, Intel (Accessed 15.10.2023). URL ↑329

[118] D. Dhote, C. Virmani, K. G. Krishna, S. Raghav. "The science of ray tracing", *International Journal of Computer Applications*, **176**:42 (2020), pp. 15–20. doi ↑329

[119] *Intel data center GPU Max series*, Intel (Accessed 15.10.2023). URL ↑330, 335

[120] H. Jiang. “Intel’s Ponte Vecchio GPU: architecture, systems and software”, *2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 August 2022, Cupertino, CA, USA), IEEE, 2022, ISBN 978-1-6654-6028-6, pp. 1–29. doi ↑329, 334, 335, 338, 341

[121] M. Sidorova, L. Gorbushin, N. Koneva. “Analytical review of electronic devices of modern supercomputing systems”, Proceedings of the International Russian Automation Conference, RusAutoCon2021 (September 5-11, 2021, Sochi, Russia), Lecture Notes in Electrical Engineering, vol. **857**, Springer, Cham, 2022, ISBN 978-3-030-94201-4, pp. 25–33. doi ↑329

[122] W. Tian, B. Li, Z. Li, H. Cui, J. Shi, Y. Wang, J. Zhao. “Using chiplet encapsulation technology to achieve processing-in-memory functions”, *Micromachines*, **13**:10 (2022), pp. 1790. doi ↑330, 331

[123] S. K. Moore. “3 paths to 3D processors”, *IEEE Spectrum*, **59**:6 (2022), pp. 24–29. doi ↑330

[124] S. Zhang, Z. Li, H. Zhou, R. Li, S. Wang, W. Kyung-Paik, P. He,. “Recent prospectives and challenges of 3D heterogeneous integration”, *e-Prime-Advances in Electrical Engineering, Electronics and Energy*, 2022, id. 100052. doi ↑330

[125] R. Hadidi, B. Asgari, B. A. Mudassar, S. Mukhopadhyay, S. Yalamanchili, H. Kim. “Demystifying the characteristics of 3D-stacked memories: A case study for Hybrid Memory Cube”, *2017 IEEE international symposium on Workload characterization (IISWC)* (01–03 October 2017, Seattle, WA, USA), IEEE, 2017, pp. 66–75. doi ↑330

[126] X. Ma, Y. Wang, Y. Wang, X. Cai, Y. Han. “Survey on chiplets: interface, interconnect and integration methodology”, *CCF Transactions on High Performance Computing*, 2022, no. 4, pp. 43–52. doi ↑330

[127] *Universal chiplet interconnect express specifications*, Universal Chiplet Interconnect Express, 2023. URL ↑330

[128] W. Gomes, A. Koker, P. Stover, D. Ingerly, S. Siers, S. Venkataraman, C. Pelto, T. Shah, A. Rao, O’ .,Mahony, E. Karl, L. Cheney, I. Rajwani, H. Jain, R. Cortez, A. Chandrasekhar, B. Kanthi, R. Koduri. “Ponte Vecchio: A multi-tile 3D stacked processor for exascale computing”, *2022 IEEE International Solid-State Circuits Conference (ISSCC)* (20–26 February 2022, San Francisco, CA, USA), IEEE, 2022, ISBN 978-1-6654-2800-2, pp. 42–44. doi ↑330, 331

[129] W. Gomes, A. Koker, P. Stover, D. Ingerly, S. Siers, S. Venkataraman, C. Pelto, T. Shah, A. Rao, F. O’Mahony, E. Karl, L. Cheney, I. Rajwani, H. Jain, R. Cortez, A. Chandrasekhar, B. Kanthi, R. Koduri. *Ponte Vecchio: A multi-tile 3D stacked processor for exascale computing*, HPC user forum, Accelerated Computing Systems and Graphics Group, 2021. URL ↑330, 331, 332, 333, 334, 335

[130] *Intel data center GPU Max series technical overview*, Intel, 2023 (Accessed 15.10.2023). URL ↑330, 331, 332, 333, 335, 336, 337

[131] S. K. Moore. *Behind Intel’s HPC chip that will pierce the exascale barrier*, Blog, IEEE Spectrum, IEEE, 2022. URL ↑330

[132] D. B. Ingerly, S. Amin, L. Aryasomayajula, A. Balankutty, D. Borst, A. Chandra, K. Cheemalapati, C. S. Cook, R. Criss, K. Enamul, W. Gomes, D. Jones, K. C. Kolluru, A. Kandas, G.-Kim S., H. Ma, D. Pantuso, C. F. Petersburg, M. Phen-givoni, A. M. Pillai, A. Sairam, P. Shekhar, P. Sinha, P. Stover, A. Telang, Z. Zell. "Foveros: 3D integration and the use of face-to-face chip stacking for logic devices", *2019 IEEE International Electron Devices Meeting (IEDM)* (07–11 December 2019, San Francisco, CA, USA), IEEE, 2019, ISBN 978-1-7281-4033-9, pp. 19.6.1-19.6.4. ↑331

[133] R. Mahajan, R. Sankman, N. Patel, W. Dae-Kim, K. Aygun, Z. Qian, Y. Mekonnen, I. Salama, S. Sharan, D. Iyengar, D. Mallik. "Embedded multi-die interconnect bridge (EMIB)–a high density, high bandwidth packaging interconnect", *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)* (31 May 2016–03 June 2016, Las Vegas, NV, USA), IEEE, 2016, pp. 557–565. ↑331

[134] S. Irani. *Hang SK Intel Ponte Vecchio compute accelerator OAM product and system*, 2021 OCP Global Summit, 2021. ↑331

[135] A. Tekin, A.Durak T., C. Piechurski, D. Kaliszan, F. A. Sungur, F. Robertsén, P. Gschwandtn. *State-of-the-art and trends for computing and interconnect network solutions for HPC and AI*, Partnership for Advanced Computing in Europe, PRACE, 2021, 38 pp. ↑332

[136] W. Sun, A. Li, T. Geng, S. Stuijk, H. Corporaal. "Dissecting tensor cores via microbenchmarks: latency, throughput and numerical behaviors", *IEEE Transactions on Parallel and Distributed Systems*, **34**:1 (2023), pp. 246–261. ↑332

[137] *Intel Products formerly Alchemist*, Intel (Accessed 15.10.2023). ↑333

[138] D. Watts. *Lenovo ThinkSystem and ThinkAgile GPU Summary*, Product Guide, Lenovo press, 2024, 71 pp. ↑333

[139] Zh. Liu. *Intel Axes Data Center GPU Max 1350, Preps New Max 1450 for 'Different Markets'*, Tom's Hardware, Future US, New York, 2023. ↑333

[140] R. Vuduc, A. Chandramowlishwaran, J. Choi, M. (E.) Guney, A. Shringarpure. "On the limits of GPU acceleration", *Proceedings of the 2nd USENIX conference on Hot topics in parallelism*, HotPar'10 (June 14–15, 2010, Berkeley, CA, USA), USENIX Association, Berkeley, 2010, id. 13, 6 pp. ↑334

[141] B. Hanindhito, D. Gourounas, A. Fathi, D. Trenev, A. Gerstlauer, L. K. John. "GAPS: GPU-acceleration of PDE solvers for wave simulation", *ICS '22: Proceedings of the 36th ACM International Conference on Supercomputing* (June 28–30, 2022, Virtual Event), ACM, NeW York, 2022, ISBN 978-1-4503-9281-5, id. 30, 13 pp. ↑334

[142] N. Chalmers, A. Mishra, D. Mc Dougall, T. Warburton. "HipBone: A performance-portable GPU-accelerated C++ version of the NekBone benchmark", *The International Journal of High Performance Computing Applications*, **37**:5 (2023), pp. 560-577. ↑334, 417, 430, 432, 433

[143] J.-L. Philippe. *Intel HW roadmap and architecture specifics*, OneAPI workshop with FocusCoE, 2022, 48 pp. URL ↑334, 336

[144] M. Min, H. Yu-Lan, P. Fischer, T. Rathnayake, J. Holmen. *Nek5000/RS Performance on Advanced GPU Architectures*, ANL-22/81, Argonne National Lab.(ANL), Argonne, IL, 2022., 30 pp. ↑337, 431, 432

[145] *oneAPI GPU Optimization Guide*, edition 2023.1, Intel, 2023, 411 pp. URL ↑337

[146] D. Blythe. "XeHPC ponte vecchio", 2021 IEEE hot chips 33 symposium (HCS), 2021, pp. 1–34. doi ↑336

[147] A. J. van der Steen, "Overview of recent supercomputers": J. J. Dongarra, A. J. Van der Steen, *High-performance computing systems: Status and outlook*, Acta Numerica, vol. **21**, 2012, pp. 379–474 . doi URL ↑337

[148] *Intel Xeon CPU Max Series*, Product Brief, Intel., 2023, 3 pp. URL ↑337

[149] G. M. Shipman, S. Swaminarayan, G. Grider, J. Lujan, R. J. Zerr. *Early performance results on 4th Gen Intel Xeon scalable processors with DDR and Intel Xeon processors, codenamed sapphire rapids with HBM*, 2022, 5 pp. arXiv 2211.05712 ↑337

[150] *SiPearl: collaboration with Intel to accelerate exascale supercomputing deployment in Europe*, Press release, The Silicon Pearl, 2 pp. URL ↑337

[151] S. Parker. *Future ALCF systems*, 2021 ALCF Computational Performance Workshop, Argonne National Laboratory, 2021, 25 pp. URL ↑338

[152] Y. Ghadar, T. Williams. *An overview of Aurora, Argonnes upcoming exascale system*, ALCF Developer Session (December 11 2019), 2020, 45 pp. URL ↑338

[153] *SYCL*, The SYCL main page, Khronos Group. URL ↑338

[154] Castaño G., Y. aqir-Rhazoui, C. García, M. Prieto-Matías. "Evaluation of Intel's DPC++ compatibility tool in heterogeneous computing", *Journal of Parallel and Distributed Computing*, **165** (2022), pp. 120–129. doi ↑338, 339

[155] *Intel oneAPI 2023 Release: Preview the Tools*, Intel (Accessed 15.10.2023). URL ↑338

[156] *Intel oneAPI plug-ins from Codeplay for Nvidia and AMD GPUs*, Intel (Accessed 15.10.2023). URL ↑338

[157] *oneAPI DPC++ Compiler documentation*, LLVM documentation, Intel, 2024. URL ↑338

[158] *Benchmarking the performance of oneAPI on heterogeneous computing platforms*, webinar slides, Moasys, Intel Software, 2021, 30 pp. URL ↑338

[159] *OneAPI Specifications*, Unified Acceleration Foundation, 2024. URL ↑338

[160] Z. Wang, Y. Plyakhin, C. Sun, Z. Zhang, Z. Jiang, A. Huang, H. Wang. "A source-to-source CUDA to SYCL code migration tool: Intel DPC++ Compatibility Tool", *IWOCL '22: Proceedings of the 10th International Workshop on OpenCL* (May 10–12, 2022, Bristol, United Kingdom), ACM, New York, 2022, ISBN 978-1-4503-9658-5, id. 17, 2 pp. doi ↑339

[161] A. Fortenberry, S. Tomov. "Extending MAGMA portability with OneAPI", *2022 Workshop on Accelerator Programming Using Directives (WACCPD)* (13–18 November, 2022, Dallas, TX, USA), IEEE, 2022, ISBN 978-1-6654-9019-1, pp. 22–31. ↑339

[162] D. J. Hardy, J. Choi, W. Jiang, E. Tajkhorshid. "Experiences porting NAMD to the Data Parallel C++ programming model", *IWOCL '22: Proceedings of the 10th International Workshop on OpenCL* (May 10–12, 2022, Bristol, United Kingdom), ACM, New York, 2022, ISBN 978-1-4503-9658-5, id. 15, 5 pp. ↑339

[163] A. Alekseenko, S. Páll, E. Lindahl. "Experiences with adding SYCL support to GROMACS", *IWOCL '21: Proceedings of the 9th International Workshop on OpenCL* (April 27–29, 2021, Munich, Germany), ACM, New York, ISBN 978-1-4503-9033-0, pp. 1, id. 17. ↑339

[164] *GROMACS Highlights*, GROMACS development team, 2023. ↑339

[165] A. Alpay, B. Soproni, H. Wünsche, V. Heuveline. "Exploring the possibility of a hipSYCL-based implementation of oneAPI", *IWOCL '22: Proceedings of the 10th International Workshop on OpenCL* (May 10–12, 2022, Bristol, United Kingdom), ACM, New York, 2022, ISBN 978-1-4503-9658-5, id. 10, 12 pp. ↑339, 410

[166] I. Sakiotis, K. Arumugam, M. Paterno, D. Ranjan, B. Terzić, M. Zubair, *High Performance Computing*, 38th International Conference ISC High Performance 2023 (May 21–25, 2023, Hamburg, Germany), Lecture Notes in Computer Science, vol. **13948**, Springer, Cham, 2023, ISBN 978-3-031-32040-8, pp. 339–358. ↑339

[167] I. Z. Reguly, A. M. B. Owenson, A. Powell, S. A. Jarvis, G. R. Mudalige. "Under the hood of SYCL — an initial performance analysis with an unstructured-mesh CFD application", *High Performance Computing*, 36th International Conference ISC High Performance 2021 (June 24–July 2, 2021, Virtual Event), Lecture Notes in Computer Science, vol. **12728**, Springer, Cham, 2021, ISBN 978-3-030-78712-7, pp. 391–410. ↑339

[168] A. C. Walden, M. Zubair, E. J. Nielsen. "Performance and portability of a linear solver across emerging architectures", *Accelerator Programming Using Directives*, 7th International Workshop WACCPD 2020 (November 20, 2020, Virtual Event), Lecture Notes in Computer Science, vol. **12655**, Springer, Cham, 2021, ISBN 978-3-030-74223-2, pp. 61–79. ↑339, 409

[169] M. Zubair, C. Stone, A. Walden, E. Nielsen. *Experiences in Moving CUDA-Optimized Kernels to Intel GPUs using oneAPI*, SC21, 2021, 22 pp. ↑339

[170] *Nvidia HPC Compilers User's Guide*, DU-09862-001-V2023, version 2023, 173 pp. ↑339

[171] M. Karp, D. Massaro, N. Jansson, A. Hart, J. Wahlgren, P. Schlatter, S. Markidis. "Large-scale direct numerical simulations of turbulence using GPUs and modern Fortran", *The International Journal of High Performance Computing Applications*, **37**:5 (2023), pp. 487–502. doi ↑340, 433, 434

[172] *OpenMP Compilers & Tools*, OpenMP, 2023. URL ↑340

[173] T. Cojean, Y. H. M. Tsai, H. Anzt. "Ginkgo—A math library designed for platform portability", *Parallel Computing*, **111** (2022), id. 102902. doi ↑340, 366, 367, 368, 372

[174] J. Fuentes, D. López, S. González. "Teaching heterogeneous computing using DPC++", *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (30 May 2022–03 June 2022, Lyon, France), IEEE, 2022, ISBN 978-1-6654-9747-3, pp. 354–360, id. 102902. doi ↑340

[175] Y. Fridman, G. Tamir, G. Oren. "Portability and scalability of OpenMP offloading on state-of-the-art accelerators", *High Performance Computing*, ISC High Performance 2023 International Workshops (May 21–25, 2023, Hamburg, Germany), Lecture Notes in Computer Science, vol. **13999**, Springer, Cham, 2023, ISBN 978-3-031-40842-7, pp. 378–390. doi ↑340

[176] I. Z. Reguly. "Evaluating the performance portability of SYCL across CPUs and GPUs on bandwidth-bound applications", *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W'23 (November 12–17, 2023, Denver, CO, USA), ACM, New York, 2023, ISBN 979-8-4007-0785-8, pp. 1038–1047. doi ↑340, 341, 344, 410

[177] *SPEC CPU2017 Results*, Standard Performance Evaluation Corporation. URL ↑341

[178] L. Solis-Vasquez, E. Mascarenhas, A. Koch. "Experiences migrating CUDA to SYCL: A molecular docking case study", *Proceedings of the 2023 International Workshop on OpenCL*, IWOCL'23 (April 18–20, 2023, Cambridge, United Kingdom), ACM, New York, 2023, ISBN 979-8-4007-0745-2, pp. 1–11, id. 15. doi ↑341, 342

[179] P. Nguyen, P. Nayak, H. Anzt, *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '23 (November 12–17, 2023, Denver, CO, USA), ACM, New York, 2023, ISBN 979-8-4007-0785-8, pp. 1048–1058. arXiv 2308.08417 doi ↑342

[180] T. P. Morgan. *One New Feature For Intel's HPC Compute Engines: Contrition*, The Next Platform, Stackhouse Publishing, 2022. URL ↑343

[181] T. P. Morgan. *Aurora In A Socket: What Intel's Falcon Shores XPU Might Do*, The Next Platform, Stackhouse Publishing, 2022. URL ↑343

[182] *Nvidia Parallel Thread Execution ISA*, Release 8.4, Nvidia, 2024, 598 pp. URL ↑345

[183] *Inline PTX Assembly in CUDA*, vol. **1**, Release 12.4, Nvidia, 2024, 16 pp. URL ↑345

[184] *Nvidia Ampere GA102 GPU Architecture*, Updated with Nvidia RTX A6000 and Nvidia A40 Information, V2.0, Nvidia, 2021, 53 pp. URL ↑345

[185] *GPU Specs Database*, A reference list of most graphics cards released in recent years, TechPowerUp. URL ↑345, 346, 350, 351, 379, 398, 399, 404, 430

[186] M. Osama, D. Merrill, C. Cecka, M. Garland, J. D. Owens, *Proceedings of the 28th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPoPP'23 (25 February 2023–1 March 2023, Montreal, QC, Canada), ACM, New York, 2023, ISBN 979-8-4007-0015-6, pp. 429–431. doi arXiv 2301.03598 ↑350

[187] *Nsight Compute*, The User Guide for Nsight Compute, v2024.1.1, Nvidia. URL ↑351

[188] A. Li, S. L. Song, M. Wijtvliet, A. Kumar, H. Corporaal. "SFU-driven transparent approximation acceleration on GPUs", *Proceedings of the 2016 International Conference on Supercomputing*, ICS'16 (June 1–3, 2016, Istanbul, Turkey), ACM, New York, 2016, ISBN 978-1-4503-4361-9, pp. 1–14, id. 15. doi ↑352

[189] Z. Jia, M. Maggioni, B. Staiger, D. P. Scarpazza. *Dissecting the Nvidia volta GPU architecture via microbenchmarking*, 2018, 66 pp. arXiv 1804.06826 ↑352

[190] J. Choquette, W. Gandhi, O. Giroux, N. Stam, R. Krashinsky. "Nvidia A100 tensor core GPU: performance and innovation", *IEEE Micro*, **41**:2 (2021), pp. 29–35. doi ↑352, 355, 373, 376

[191] *Multi-Instance GPU User Guide*, RN-08625-v2.0, Nvidia, 2024, iv+53 pp. URL ↑353

[192] *Nvidia A100 80GB PCIe GPU*, Product Brief, PB-10577-001_v03, Nvidia, 2022. URL ↑353

[193] A. Li, S. L. Song, J. Chen, J. Li, X. Liu, N. R. Tallent, K. J. Barker. "Evaluating modern GPU interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect", *IEEE Transactions on Parallel and Distributed Systems*, **31**:1 (2019), pp. 94–110. doi ↑354

[194] C. Lutz, S. Breß, S. Zeuch, T. Rabl, V. Markl. "Pump up the volume: Processing large data on GPUs with fast interconnects", *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD'20 (June 14–19, 2020, Portland, OR, USA), ACM, New York, 2020, ISBN 978-1-4503-6735-6, pp. 1633–1649. doi ↑354

[195] *Nvidia NVSwitch*, Technical Overview, Nvidia, 2018, 8 pp. URL ↑354

[196] M. Špeťko, O. Vysocký, B. Jansík, L. Říha. "DGX-A100 Face to Face DGX-2—performance, power and thermal behavior evaluation", *Energies*, **14**:2 (2021), id. 376, 18 pp. doi ↑355, 356

[197] Y. R. Choi, V. Nikolskiy, V. Stegailov. "Matrix-matrix multiplication using multiple GPUS connected by Nvlink", *2020 Global Smart Industry Conference (GloSIC)* (17–19 November 2020, Chelyabinsk, Russia), IEEE, 2020, ISBN 9781728180755, pp. 354–361. doi ↑356

[198] M. Manathunga, C. Jin, V. W. D. Cruzeiro, Y. Miao, D. Mu, K. Arumugam, K. Keipert, H. M. Aktulga, K. M. Merz jr, A. W. Götz. "Harnessing the power of multi-GPU acceleration into the quantum interaction computational kernel program", *Journal of Chemical Theory and Computation*, **17**:7 (2021), pp. 3955–3966. ↑356, 373, 374

[199] Y. R. Choi, V. Nikolskiy, V. Stegailov. "Matrix-matrix multiplication using multiple GPUS connected by Nvlink", *2020 Global Smart Industry Conference (GloSIC)* (17–19 November 2020, Chelyabinsk, Russia), IEEE, 2020, ISBN 9781728180755, pp. 354–361. ↑356

[200] Y. R. Choi, V. Stegailov. "Multi-GPU GEMM algorithm performance analysis for Nvidia and AMD GPUs connected by NVLink and PCIe", *Mathematical Modeling and Supercomputer Technologies: 22nd International Conference*, Revised Selected Papers, 22nd International Conference MMST 2022 (November 14–17, 2022, Nizhny Novgorod, Russia), Springer, Cham, 2022, ISBN 978-3-031-24144-4, pp. 281–292. ↑356

[201] *Nvidia DGX A100*, Datasheet, Nvidia, 2023, 2 pp. ↑356

[202] *Nvidia DGX A100*, User Guide, DU-09821-001_v01, Nvidia, 2023, 126 pp. ↑356

[203] *Nvidia DGX Platform*, Cloud & Data Center, Nvidia, 2024. ↑356

[204] *Nvidia DGX SuperPOD: Scalable infrastructure for AI leadership*, Reference Architecture, RA-09950-001, Nvidia, 2021, 30 pp. ↑356

[205] *Leonardo HPC system*, Technical info, Leonardo Pre-exascale Supercomputer, 2024. ↑357, 413

[206] *Perlmutter architecture*, NERSC documentation, NERSC. ↑357, 413

[207] *Nvidia HPC SDK*, Overview, Containers, Nvidia. ↑357

[208] *Nvidia HPC SDK Version 24.3 Documentation*, Nvidia, 2024. ↑357, 358

[209] *CUDA LLVM Compiler*, Nvidia Developer, Nvidia. ↑358

[210] S. Potluri, K. Hamidouche, A. Venkatesh, D. Bureddy, K. Panda. "Efficient inter-node MPI communication using GPUDirect RDMA for InfiniBand clusters with Nvidia GPUs", *2013 42nd International Conference on Parallel Processing* (01–04 October 2013, Lyon, France), IEEE, 2013, ISBN 978-0-7695-5117-3, pp. 80–89. ↑358

[211] *Nvidia CUDA Fortran programming guide*, HPC SDK documentation, version 24.3, Nvidia, 2024. ↑360, 361

[212] G. Ruetsch, M. Fatica. *CUDA Fortran for scientists and engineers. Best practices for efficient CUDA Fortran programming*, 1st ed., Morgan Kaufmann, 2013, ISBN 978-0-12-416970-8, 338 pp. ↑362

[213] S. Oyanagi. *HPE Cray MPI Update*, Slides, SC'21 ANL MPICH BOF (November 17, 2021), 6 pp. ↑362

[214] *Developing a Linux Kernel Module using GPUDirect RDMA*, v12.4, Nvidia, 2024, 48 pp. ↑362

[215] *MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, RoCE, and Slingshot*, User guide, Version 2.3.7, ed. Dhabaleswar K., NBCL, 2024. URL ↑362

[216] S. Bhalachandra, B. Austin, S. Williams, N. J. Wright. *Understanding the impact of input entropy on FPU, CPU, and GPU power*, 2022, 6 pp. arXiv 2212.08805 ↑363

[217] *Nvidia Docs. Matrix multiplication background*, User's Guide, Deep learning, DU-09799-001_v001, Nvidia, 2023, 17 pp. URL ↑363, 368

[218] D. Mukunoki, K. Ozaki, T. Ogita, T. Imamura. "DGEMM using tensor cores, and its accurate and reproducible versions", *High Performance Computing*, 35th International Conference, ISC High Performance 2020 (June 22–25, 2020, Frankfurt/Main, Germany), Lecture Notes in Computer Science, vol. **12151**, Springer, Cham, 2020, ISBN 978-3-030-50742-8, pp. 230–248. doi ↑363

[219] M. Fasi, N. J. Higham, M. Mikaitis, S. Pranesh. "Numerical behavior of Nvidia tensor cores", *PeerJ Computer Science*, 2021, id. 7e330. doi ↑364

[220] M. Fasi, N. J. Higham, F. Lopez, T. Mary, M. Mikaitis. "Matrix multiplication in multiword arithmetic: error analysis and application to GPU tensor cores", *SIAM Journal on Scientific Computing*, **45**:1 (2023), pp. C1–C19. doi ↑364

[221] S. Li, K. Osawa, T. Hoefler. *Efficient quantized sparse matrix operations on tensor cores*, 2022, 13 pp. arXiv 2209.06979 ↑364

[222] D. Tao, J. Tiuan. *Performance of Sample CUDA Benchmarks on Nvidia Ampere A100 vs Tesla V100*, GitHub Inc., 2021. URL ↑364, 368

[223] *CUDA Samples*, Reference Manual, TRM-06704-001_v11.2, 142 pp. URL ↑364

[224] *All ACCEL Results Published by SPEC*, Standard Performance Evaluation Corporation, 2024. URL ↑364

[225] H. Brunst, S. Chandrasekaran, F. Ciorba, N. Hagerty, R. Henschel, G. Juckeland, J. Li, V. G. M. Vergara, S. Wienke, M. Zavala, *2022 22nd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)* (16-19 May 2022, Taormina, Italy), 2022, ISBN 978-1-6654-9956-9, pp. 675–684. doi arXiv 2203.06751 ↑364, 414

[226] *All HPC2021 Results Published by SPEC*, SPEC, 2024. URL ↑364, 391

[227] M. Svedin, S. W. D. Chien, G. Chikafa, N. Jansson, A. Podobas. "Benchmarking the Nvidia GPU lineage: From early K80 to modern A100 with asynchronous memory transfers", *Proceedings of the 11th International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies* (June 21–23, 2021, Online Germany), ACM, New York, 2021, ISBN 978-1-4503-8549-7, id. 9, 6 pp. doi arXiv 2106.04979 ↑365

[228] L. Zhang, M. Wahib, P. Chen, J. Meng, X. Wang, T. Endo, S. Matsuoka. *Persistent kernels for iterative memory-bound GPU applications*, 2022. arXiv 2204.02064 ↑365

[229] M. Špeťko, O. Vysocký, B. Jansík, L. Říha. "DGX-A100 face to face DGX-2—performance, power and thermal behavior evaluation", *Energies*, **14**:2 (2021), pp. 376. ↑365

[230] B. Jansik. *Mandelbrot benchmark*, Accessed 15.10.2023. ↑365

[231] D. Mudigere, Y. Hao, J. Huang, Z. Jia, A. Tulloch, S. Sridharan, X. Liu, M. Ozdal, J. Nie, J. Park, L. Luo, J. A. Yang, L. Gao, D. Ivchenko, A. Basant, Y. Hu, J. Yang, E. K. Ardestani, X. Wang, R. Komuravelli, H. Ching-Chu, S. Yilmaz, H. Li, J. Qian, Z. Feng, Y. Ma, J. Yang, E. Wen, H. Li, L. Yang, C. Sun, W. Zhao, D. Melts, K. Dhulipala, R. KKishore, T. Graf, A. Eisenman, K. K. Matam, A. Gangidi, G. J. Chen, M. Krishnan, A. Nayak, K. Nair, B. Muthiah, M. Khorashadi, P. Bhattacharya, P. Lapukhov, M. Naumov, A. Mathews, L. Qiao, M. Smelyanskiy, B. Jia, V. Rao. "Software-hardware co-design for fast and scalable training of deep learning recommendation models", *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22 (June 18–22, 2022, New York, USA), ACM, New York, 2022, ISBN 978-1-4503-8610-4, pp. 993–1011. arXiv 2104.05158 ↑365, 367, 368, 369, 379

[232] T. Deakin, J. Price, M. Martineau, S. McIntosh-Smith. "Evaluating attainable memory bandwidth of parallel programming models via BabelStream", *International Journal of Computational Science and Engineering*, **17**:3 (2018), pp. 247–262. ↑365, 366

[233] J. D. McCalpin. *Memory bandwidth and machine balance in current high performance computers*, IEEE computer society technical committee on computer architecture (TCCA) newsletter, 1995, 7 pp. ↑366

[234] Y. M. Tsai, T. Cojean, H. Anzt. "Evaluating the performance of Nvidia's A100 Ampere GPU for sparse and batched computations", *2020 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)* (12 November 2020, GA, USA), IEEE, 2020, ISBN 978-0-7381-1048-6, pp. 26–38. arXiv 2008.08478 ↑366, 367, 368, 370, 371, 372

[235] Y. M. Tsai, T. Cojean, H. Anzt. *Evaluating the performance of Nvidia's A100 Ampere GPU for sparse linear algebra computations*, 2020, 9 pp. arXiv 2008.08478 ↑366, 367, 368, 372

[236] J. R. Hammond, T. Deakin, J. Cownie, S. McIntosh-Smith. "Benchmarking Fortran DO CONCURRENT on CPUs and GPUs using BabelStream", *2022 IEEE/ACM International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)* (13-18 November 2022, Dallas, TX, USA), IEEE, 2022, ISBN 978-1-6654-5185-7, pp. 82–99. ↑366, 367, 414

[237] C. J. Balos. "Reproduced computational results report for "Ginkgo": a modern linear operator algebra framework for high performance computing"", *ACM Transactions on Mathematical Software (TOMS)*, **48**:1 (2022), id. 3, 7 pp. ↑367, 368

[238] T. Grützmacher, H. Anzt, E. S. Quintana-Ortí. "Using Ginkgo's memory accessor for improving the accuracy of memorybound low precision BLAS", *Software: Practice and Experience*, **53**:1, Special Issue: New Trends in High-Performance Computing: Software Systems and Applications (2021), pp. 81–98. ↑367, 368, 371, 372

[239] *Mixbench*, A benchmark suite for GPUs on mixed operational intensity kernels, Openbenchmarking.org, Phoronix Media, 2024. ↑367

[240] T. Dong, A. Haidar, P. Luszczek, S. Tomov, A. Abdelfattah, J. Dongarra. "Magma batched: A batched blas approach for small matrix factorizations and applications on gpus", *Journal of Latex class files*, **14**:8 (2015), Accessed 15.10.2023. ↑367

[241] A. Abdelfattah, S. Tomov, J. Dongarra. "Batch QR factorization on GPUs: design, optimization, and tuning", Computational Science – ICCS 2022 (June 21–23, 2022, London, UK), Lecture Notes in Computer Science, vol. **13350**, Springer, Cham, 2022, ISBN 978-3-031-08750-9, pp. 60–74. ↑368, 371, 414, 415

[242] A. Abdelfattah, V. Barra, N. Beams, R. Bleile, J. Brown, S. Jean-Camier, R. Carson, N. Chalmers, V. Dobrev, Y. Dudouit, P. Fischer, A. Karakus, S. Kerkemeier, T. Kolev, H. Yu-Lan, E. Merzari, M. Min, M. Phillips, T. Rathnayake, R. Rieben, T. Stitt, A. Tomboulides, S. Tomov, V. Tomov, A. Vargas, T. Warburton, K. Weiss. "GPU algorithms for efficient exascale discretizations", *Parallel Computing*, **108** (2021), id. 102841, 10 pp. ↑368, 369, 416

[243] T. Dong, V. Dobrev, T. Kolev, R. Rieben, S. Tomov, J. Dongarra. "A step towards energy efficient computing: Redesigning a hydrodynamic application on CPU-GPU", *2014 IEEE 28th International Parallel and Distributed Processing Symposium* (19–23 May 2014, Phoenix, AZ, USA), IEEE, 2014, ISBN 978-1-4799-3800-1, pp. 972–981. ↑369

[244] A. Abdelfattah, M. Baboulin, V. Dobrev, J. Dongarra, C. Earl, J. Falcou, A. Haidar, I. Karlin, T. Kolev, I. Masliah, S. Tomov. "High-performance tensor contractions for GPUs", *Procedia Computer Science*, **80** (2016), pp. 108–118. ↑369

[245] A. Heinecke, G. Henry, M. Hutchinson, H. Pabst. "LIBXSMM: accelerating small matrix multiplications by runtime code generation", *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC'16 (13–18 November 2016, Salt Lake City, UT), IEEE, 2016, ISBN 978-1-4673-8815-3, pp. 981–991. ↑369

[246] I. Bethune, F. Reid, A. Lazzaro. *CP2K Performance from Cray XT3 to XC30*, Cray User Group (CUG), 2014, 11 pp. ↑369

[247] A. Sedova, A. Tharrington, B. Messer. *Portability in scientific computing: The molecular dynamics non-bonded forces calculation as a case study*, 2018, 25 pp. ↑369

[248] H. Waugh, S. McIntosh-Smith. "On the use of BLAS libraries in modern scientific codes at scale", *Driving Scientific and Engineering Discoveries Through the Convergence of HPC, Big Data and AI*, 17th Smoky Mountains Computational Sciences and Engineering Conference SMC 2020 (August 26–28, 2020, Oak Ridge, TN, USA), Communications in Computer and Information Science, vol. **1315**, Springer, Cham, 2020, ISBN 978-3-030-63392-9, pp. 67–79. ↑369

[249] N. Mijić, D. Davidović. "Batched matrix operations on distributed GPU's with application in theoretical physics", *2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO)* (23–27 May 2022, Opatija, Croatia), IEEE, 2022, ISBN 978-953-233-103-5, pp. 293–299. arXiv 2203.09353 ↑370

[250] C. Stylianou, M. Weiland. *Optimizing sparse linear algebra through automatic format selection and machine learning*, 2023, 10 pp. arXiv 2303.05098 ↑372

[251] V. R. Pascuzzi, M. Goli. "Benchmarking a proof-of-concept performance portable SYCL-based fast Fourier transformation Library", *Proceedings of the 10th International Workshop on OpenCL*, International Workshop on OpenCL IWOCL'22 (May 10–12, 2022, Bristol, United Kingdom), ACM, New York, 2022, ISBN 978-1-4503-9658-5, id. 20, 9 pp. arXiv 2203.09384 ↑372, 414

[252] D. Tolmachev. "VkFFT-a performant, cross-platform and open-source GPU FFT library", *IEEE Access*, **11** (2023), pp. 12039–12058. ↑372

[253] B. Li, S. Cheng, J. Lin. *tcFFT: Accelerating half-precision FFT through tensor cores*, 2021, 10 pp. arXiv 2104.11471 ↑372, 373

[254] N. Hagerty, Vergara V. Melesse, A. Tharrington. "Studying performance portability of LAMMPS across diverse GPU-based platforms", S2 World 2020. CUG 2021 & 2022. PN_HCP. HeteroPar 2022, *Concurency and computation. Practice and Experience*, **35**:28 (2023), id. e7895. ↑373, 396, 424

[255] A. Poenaru, W.-C. Lin, S. McIntosh-Smith. "A performance analysis of modern parallel programming models using a compute-bound application", *ISC High Performance 2021: High Performance Computing*, Lecture Notes in Computer Science, vol. **12728**, Springer, Cham, 2021, ISBN 978-3-030-78712-7, pp. 332–350. ↑373

[256] L. Solis-Vasquez, A. F. Tillack, D. Santos,-Martins, A. Koch, S. Le,Grand, S. Forli. "Benchmarking the performance of irregular computations in AutoDock-GPU molecular docking", *Parallel Computing*, **109** (2022), id. 102861, 12 pp. ↑373

[257] M. Manathunga, H. M. Aktulga, A. W. Götz, K. M. Merz. "Quantum mechanics/molecular mechanics simulations on Nvidia and AMD graphics processing units", *J. Chem. Inf. Model.*, **63**:3 (2023), pp. 711-717. ↑374

[258] V. W. D. Cruzeiro, M. Manathunga, K. M. Merz, A. W. Götz. "Open-source multi-GPU-accelerated QM/MM simulations with AMBER and QUICK", *J. Chem. Inf. Model.*, **61**:5 (2021), pp. 2109–2115. ↑374

[259] A. Shajan, M. Manathunga, A. W. Götz, K. M. Jr. Merz. "Geometry optimization: A comparison of different open-source geometry optimizers", *J. Chem. Theory Comput.*, **19**:21 (2023), pp. 7533–7541. doi ↑374

[260] D. B. Williams-Young, A. Asadchev, D. T. Popovici, D. Clark, J. Waldrop, T. Windus, E. F. Valeev, W. A. de Jong. "Distributed memory, GPU accelerated Fock construction for hybrid, Gaussian basis density functional theory", *The Journal of Chemical Physics*, **158**:23 (2023), id. 234104. doi ↑374

[261] I. Kim, D. Jeong, J. Won-Son, J. Hyung-Kim, Y. M. Rhee, Y. Jung, H. Choi, J. Yim, I. Jang, D. S. Kim. "Kohn-Sham time-dependent density functional theory on the massively parallel GPUs", *npj Computational Materials*, **9** (2023), id. 81, 12 pp. doi ↑374, 375

[262] A. Siegel, E. Draeger, J. Deslippe, T. M. Evans, M. Francois, T. Germann, D. Martin, W. Hart. *Application Results on Early Exascale Hardware*, No. ORNL/TM-2022/2437, Oak Ridge National Lab.(ORNL), Oak Ridge, TN (US), 2022. URL ↑375, 404, 412, 413, 415, 417, 418, 424, 425

[263] G. Dang, S. Liu, T. Guo, J. Dang, X. Li. "Direct numerical simulation of compressible turbulence accelerated by graphics processing unit: An open-source high accuracy accelerated computational fluid dynamic software", *Physics of Fluids*, **34**:12 (2022), id. 126106. doi ↑375

[264] M. Min, A. Tombouldies. *Simulating Atmospheric Boundary Layer Turbulence with Nek5000/RS*, No. ANL-22/79, Argonne National Lab.(ANL), Argonne, IL, 2022, 34 pp. ↑375

[265] P. Fischer, S. Kerkemeier, M. Min, H. Yu-Lan, M. Phillips, T. Rathnayake, E. Merzari, A. Tomboulides, A. Karakus, N. Chalmers, T. Warburton. "NekRS, a GPU-accelerated spectral element Navier–Stokes solver", *Parallel Computing*, **114** (2022), id. 102982, 13 pp. doi ↑376, 416

[266] *FUN3D is a Computational Fluid Dynamics (CFD) suite of tools actively developed at NASA that benefits Aeronautics, Space Technology, and Exloration by modeling fluid flow*, 14.0.2-16d1333, NASA Official: David P. Lockard. URL ↑376

[267] G. Nastac, A. Walden, E. J. Nielsen, K. Frendi. "Implicit thermochemical nonequilibrium flow simulations on unstructured grids using GPUs", AIAA Scitech 2021 Forum (11–15, 19-21 January 2021, virtual event), 2021, id. AIAA 2021-0159. doi ↑376

[268] G. Nastac, A. Walden, L. Wang, E. J. Nielsen, Y. Liu, M. Opgenorth, J. Orender, M. Zubair. "A multi-architecture approach for implicit computational fluid dynamics on unstructured grids", AIAA Scitech 2023 Forum (23–27 January 2023, National Harbor, MD & Online), 2023, id. AIAA 2023-1226. doi ↑376, 420

[269] V. Pasquariello, Y. Bunk, S. Eberhardt, H. Pei-Huang, J. Matheis, M. Ugolotti, S. Hickel. "GPU-accelerated simulations for eVTOL aerodynamic analysis", AIAA Scitech 2023 Forum (23–27 January 2023, National Harbor, MD & Online), 2023, id. AIAA 2023-2107. doi ↑376

[270] T. Regev, J. Nestmann, A. Garzuzi, D. Greenblatt, S. Frankel. "GPU-accelerated high-fidelity implicit large eddy simulations of coanda cylinder flow instabilities", AIAA Scitech 2023 Forum (23–27 January 2023, National Harbor, MD & Online), 2023, id. AIAA 2023-0272. doi ↑376

[271] H. C. V. Kakumani, A. S. Chamarthi, N. Hoffmann, S. H. Frankel. "GPU-accelerated numerical study of temperature effects in choked under-expanded supersonic jets", AIAA SCITECH 2023 Forum (23–27 January 2023, National Harbor, MD & Online), 2023, id. AIAA 2023-0976. doi ↑376

[272] J. Sitaraman, D. Jude. "Development of GPGPU capable multi-solver overset methods", AIAA SCITECH 2023 Forum (23–27 January 2023, National Harbor, MD & Online), 2023, id. AIAA 2023-0042. doi ↑376

[273] M. Mortazawy, M. Rao, J. Jilesen, D. Work, R. Shock. *Early Stage Vehicle Aerodynamics Development using a GPU Based LBM CFD Solver*, SAE Technical Paper No2023-01-0560, 2003, 7 pp. doi ↑377

[274] A. Kummerländer, M. Dorn, M. Frank, M. J. Krause. "Implicit propagation of directly addressed grids in lattice Boltzmann methods", *Concurrency and Computation: Practice and Experience*, **35**:8 (2023), id. e7509. doi ↑377

[275] De Vanna F., F. Avanzi, M. Cogo, S. Sandrin, M. Bettencourt, F. Picano, E. Benini. "URANOS: A GPU accelerated Navier-Stokes solver for compressible wall-bounded flows", *Computer Physics Communications*, 2023, id. 108717, 18 pp. doi ↑377

[276] H. Chandravamsi, A. S. Chamarthi, N. Hoffmann, S. H. Frankel. "On the application of gradient based reconstruction for flow simulations on generalized curvilinear and dynamic mesh domains", *Computers & Fluids*, 2023, id. 105859, 28 pp. doi ↑377

[277] P. Mattson, C. Cheng, C. Coleman, G. Diamos, P. Micikevicius, D. Patterson, H. Tang, Y. Gu-Wei, P. Bailis, V. Bittorf, D. Brooks, D. Chen, D. Dutta, U. Gupta, K. Hazelwood, A. Hock, X. Huang, A. Ike, B. Jia, D. Kang, D. Kanter, N. Kumar, J. Liao, G. Ma, D. Narayanan, T. Oguntebi, G. Pekhimenko, L. Pentecost, V. J. Reddi, T. Robie, T. S. John, T. Tabaru, J. Carole-Wu, L. Xu, M. Yamazaki, C. Young, M. Zaharia. "MLPerf training benchmark", *Proceedings of Machine Learning and Systems*. V. 2, MLSys 2020, eds. I. Dhillon, D. Papailiopoulos, V. Sze, 2020, pp. 336–349. URL arXiv:1910.01500 ↑377

[278] *MLPerf Training v.2.1 Results*, ML Commons. URL ↑377, 378, 392

[279] *MLPerf Training HPC v2.0 results*, ML Commons. URL ↑378

[280] *Nvidia NVLink and NVLink Switch*, Cloud & Data Center, Nvidia. URL ↑381

[281] *PRE-EOS 128 NODE DGX SuperPOD — Nvidia DGX H100, Xeon Platinum 8480C 56C 2GHZ, Nvidia H100 Tensor core GPUs, Nvidia ConnectX-7 NDR 400G Infiniband*, Top500.org, 2023. URL ↑383

[282] *Kestrel System Configuration*, National Renewable Energy Laboratory Computing Systems, Alliance for Sustainable Energy, 2023. URL ↑383

[283] *G242-P36 (rev. 100)*, Products, GIGA-BYTE Technology, 2024. URL ↑383

[284] *Nvidia Grace CPU Superchip Whitepaper*, V1.1, Nvidia, 2024, 20 pp. URL ↑384, 385, 386

[285] *Nvidia Grace CPU Superchip*, Datasheet, Nvidia, 2024, 3 pp. URL URL ↑384

[286] *Arm Neoverse V2 Core Technical Reference Manual*, Accessed 15.10.2023. URL ↑384

[287] *Nvidia GH200 Grace Hopper Siperchip Architecture*, V1.01, Nvidia, 2024, 39 pp. URL ↑386, 387, 388, 389

[288] *H263-V11*, Products, rev. LAW1, Giga-Byte Technology. URL ↑388

[289] H. Petty, I. Goldwasser, P. Desale. *One Giant Superchip for LLMs, Recommenders, and GNNs: Introducing Nvidia GH200 NVL32*, Technical blog, Nvidia developer, 2023. URL ↑389

[290] *Nvidia BlueField-3 networking platform*, Datasheet, Nvidia, 2023, 2 pp. URL ↑389

[291] *CUDA Python Manual*, v. 12.4.0, Nvidia, 2024. URL ↑389

[292] *Nvidia NVVM IR Specification*, v. 12.4, Nvidia, 2024, 80 pp. URL ↑390

[293] J. Choquette. "Nvidia Hopper GPU: scaling performance", *2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 August 2022, Cupertino, CA, USA), IEEE, 2022, ISBN 978-1-6654-6028-6, pp. 1–46. doi ↑391

[294] *Amber22: pmemd.cuda performance information*, The Amber project, ed. Kollman P., 2023. URL ↑392

[295] *MLPerf Training v.3.0 Results*, MLCommons, 2024. URL ↑392, 394, 395

[296] *AMD Instinct™ MI100 Accelerators*, Overview, AMD, 2020. URL ↑396

[297] *AMD Instinct™ Accelerators*, Products, AMD. URL ↑396

[298] *MLPerf Inference Datacenter v.3.1 Results*, MLCommons. URL ↑392

[299] A. Smith, N. James. "AMD Instinct™ MI200 series accelerator and node architectures", *2022 IEEE Hot Chips 34 Symposium (HCS)* (21–23 August 2022, Cupertino, CA, USA), IEEE, 2022, ISBN 978-1-6654-6028-6, 23 pp. doi ↑396, 400, 402, 403, 404, 406, 407, 408

[300] *AMD Instinct™ MI210 Accelerators*, AMD, 2022. URL ↑399, 405

[301] *AMD Instinct™ MI250 drivers & support*, Accessed 15.10.2023. URL ↑399, 405

[302] *AMD Instinct™ MI250X drivers & support*, Accessed 15.10.2023. URL ↑399, 405

[303] *Frontier user guide*, Oak Ridge National Laboratory, 2024. URL ↑399, 406, 408, 413

[304] *Introducing AMD CDNA architecture*, AMD, 2020, 11 pp. URL ↑399, 402

[305] *Introducing AMD CDNA 2 Architecture*, White paper, AMD Instinct MI200, Advanced Micro Device, 2021, 17 pp. URL ↑399, 400, 401, 402, 406, 407, 408

[306] *"AMD Instinct MI200" instruction set architecture*, Reference Guide, AMD Instinct MI200, Advanced Micro Devices, 2021, 275 pp. URL ↑402

[307] C. Sitaraman, N. Chalmers, N. Malaya, D. McDougal, O. O'Reilly, R. van Oostrum. *AMD matrix cores*, GPU open, AMD Labs notes, ed. Greathouse R., Advanced Micro Devices, 2023. ↑402

[308] C. Pearson. *Interconnect bandwidth heterogeneity on AMD MI250x and Infinity fabric*, 2023. arXiv 2302.14827 ↑407

[309] M. Gates, A. YarKhan, D. Sukkari, K. Akbudak, S. Cayrols, D. Bielich, A. Abdelfattah, M. A. Farhan, J. Dongarra. "Portable and efficient dense linear algebra in the beginning of the exascale era", *2022 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (13–18 November 2022, Dallas, TX, USA), IEEE, 2022, ISBN 978-1-6654-6021-7, pp. 36–46. ↑406

[310] V. G. Melesse Vergara, R. D. Budiardja, M. J. Davis, M. A. Ezell, J. A. Hanley, C. J. Zimmer, M. J. Brim, W. R. Elwasif, D. T. Dietz. "Approaching the final Frontier: lessons learned from the deployment of HPE/Cray EX Spock and Crusher supercomputers", *Cray User Group 2022 Proceedings*, CUG (May 2, 2022 – May 5, 2022), Oak Ridge National Lab. (ORNL), 2022. ↑408

[311] *AMD Instinct Accelerator Qualified Servers Q4 2022*, Reference Guide, AMD Instinct MI200, Advanced Micro Devices, 2022, 3 pp. ↑408

[312] *Welcome to AMD ROCm Platform*, Revision e2b73a17, Advanced Micro Devices, 2021. ↑408, 409

[313] *AMD ROCm documentation*, v. 6.1.1, 2024. ↑409, 410, 411

[314] N. Kondratyuk, V. Nikolskiy, D. Pavlov, V. Stegailov. "GPU-accelerated molecular dynamics: State-of-art software performance and porting from Nvidia CUDA to AMD HIP", *The International Journal of High Performance Computing Applications*, **35**:4 (2021), pp. 312–324. ↑409, 423

[315] D. Charrier et al.. *GPUFORT: S2S translation tool for CUDA Fortran and Fortran+X in the spirit of hipify*, AMD ROCm Software, GitHub Inc.. ↑410

[316] *AMD ROCm documentation*, AMD, 2024. ↑410

[317] K. S. Khorassani, C. Chen-Chen, B. Ramesh, A. Shafi, H. Subramoni, D. K. Panda. "High Performance MPI over the Slingshot Interconnect", *Journal of Computer Science and Technology*, **38**:1 (2023), pp. 128–145. ↑411

[318] *Welcome to the LUMI supercomputer user guide*, LUMI (Large Unified Modern Infrastructure) consortium. ↑413

[319] *Crusher Quick-Start Guide*, Oak Ridge National Laboratory, 2024. ↑413

[320] *Spock Quick-Start Guide*, Oak Ridge National Laboratory, 2023. ↑413

[321] *ThetaGPU Machine Overview*, Argonne National Laboratory, Accessed 15.10.2023. ↑413

[322] *Polaris Machine Overview*, Argonne National Laboratory, 2023. ↑413

[323] *Summit User Guide*, Alpine, Oak Ridge National Laboratory, 2023. ↑413

[324] M. A. Herou et al.x. *ECP software technology capability assessment report*, No. ORNL/TM-2022/2651, V3.0, Oak Ridge National Lab, 2022, 237 pp. URL ↑413, 422

[325] S. Sathyanarayana, M. Bernardini, D. Modesti, S. Pirozzoli, F. Salvadore. *High-speed turbulent flows towards the exascale: STREAmS-2 porting and performance*, 2023, 32 pp. arXiv 2304.05494 ↑414, 429, 430, 431

[326] A. Müller, B. Schmidt, R. Membarth, R. Leißa, S. Hack. *AnySeq/GPU: A novel approach for faster sequence alignment on GPUs*, 2022, 11 pp. arXiv 2205.07610 ↑415

[327] M. Manathunga, H. M. Aktulga, A. W. Götz, K. M. Merz jr. "Quantum mechanics/molecular mechanics simulations on Nvidia and AMD Graphics Processing Units", *Journal of Chemical Information and Modeling*, **63**:3 (2023), pp. 711–717. doi ↑416, 417

[328] T. Kolev, P. Fischer, A. P. Austin, A. T. Barker, N. Beams, J. Brown, S. Jean-Camier, N. Chalmers, V. Dobrev, Y. Dudouit, L. Ghaffary, S. Kerkemeier, H. Yu-Lan, E. Merzari, M. Min, W. Pazner, T. Rathnayake, M. S. Shephard, M. H. Siboni, C. W. Smith, J. L. Thompson, S. Tomov, T. Warburton. *High-order algorithmic developments and optimizations for large-scale GPU-accelerated simulations*, ECP Milestone Report: WBS 2.2.6.06, Milestone CEED-MS36, 2021, 51 pp. URL ↑417, 437

[329] M. Thavappiragasam, W. Elwasif, A. Sedova. *Portability for GPU-accelerated molecular docking applications for cloud and HPC: can portable compiler directives provide performance across all platforms?*, 2022, 10 pp. arXiv 2203.02096 ↑418

[330] C. P. Stone, A. Walden, M. Zubair, J. Nielsen. "Accelerating unstructured-grid CFD algorithms on Nvidia and AMD GPUs", *2021 IEEE/ACM 11th Workshop on Irregular Applications: Architectures and Algorithms (IA3)* (15 November 2021, St. Louis, MO, USA), 2021, ISBN 978-1-6654-1126-4, pp. 19–26. doi ↑418

[331] S. Pumma, A. Vishnu. "Semantic-aware lossless data compression for Deep Learning Recommendation Model (DLRM)", *2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)* (15 November 2021, St. Louis, MO, USA), IEEE, 2021, ISBN 978-1-6654-1124-0, pp. 1–8. doi ↑418

[332] F. Han, N. Kumar. *HPC Application Performance on Dell PowerEdge R750xa Servers with the AMD Instinct MI210 Accelerator*, Dell, 2022. URL ↑419

[333] *AMD Instinct MI200 Series Accelerators Benchmarks*, AMD, 2024. URL ↑419, 423, 425, 426, 429

[334] Y. Yu, C. Cai, J. Wang, Z. Bo, Z. Zhu, H. Zheng. "Uni-dock: Gpu-accelerated docking enables ultralarge virtual screening", *Journal of Chemical Theory and Computation*, **19**:11 (2023), pp. 3336–3345. doi ↑419

[335] Y. Hao, X. Zhao, B. Bao, D. Berard, W. Constable, A. Aziz, X. Liu. *TorchBench: benchmarking PyTorch with High API surface coverage*, 2023, 13 pp. arXiv 2304.14226 ↑420

[336] K. Punniyamurthy, B. M. Beckmann, K. Hamidouche. *Optimizing distributed ML communication with fused computation-collective operations*, 2023, 12 pp. arXiv 2305.06942 ↑420

[337] Y. Guo, L. Lu, S. Zhu. "Novel accelerated methods for convolution neural network with matrix core", *The Journal of Supercomputing*, **79**:17 (2023), pp. 19547–19573. ↑420

[338] A. Eassa, C. Porter. *Fueling high-performance computing with full-stack innovation*, Technical blog, Nvidia developer, 2022. ↑420

[339] *Driving the Industry into the Exascale Era with AMD Instinct Accelerators*, AMD Community, AMD, 2022. ↑420

[340] R. D. Budiardja, M. Berrill, M. Eisenbach, G. R. Jansen, W. Joubert, S. Nichols, D. M. Rogers, A. Tharrington, O. E. B. Messer,. "Ready for the Frontier: preparing applications for the world's first exascale system", *High Performance Computing*, ISC High Performance 2023, LNCS, 13948, Springer, Cham, 2023, ISBN 978-3-031-32040-8, pp. 182–201. ↑421

[341] F. Wittwer, N. K. Sauter, D. Mendez, B. K. Poon, A. S. Brewster, J. M. Holton, M. E. Wall, W. E. Hart, D. J. Bard, J. P. Blaschke. *Accelerating X-ray tracing for exascale systems using Kokkos*, 2022, 6 pp. arXiv 2205.07976 ↑421

[342] T. Papatheodore. *Frontier/Crusher node performance*, Frontier Training Workshop (February 16, 2023), Oak Ridge National Laboratory, 13 pp. ↑422, 423

[343] *MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, RoCE, and Slingshot*, NOWLAB: Network Based Computing Lab, NBCL. ↑422

[344] J. Kurzak, N. Malaya, M. Klemm, E. Hiew. *Matrix Multiply Stress Test*, GitHub inc., 2023. ↑422

[345] T. Papatheodore. *GPU XGEMM*, Benchmark, 2023. ↑422

[346] *Enhancing LAMMPS simulations with AMD instinct accelerators: unleashing performance and scalability*, Solution Brief, High Performance Computing, AMD, 2023, 4 pp. ↑423, 424

[347] *HPC Comes to Life with AMD Instinct GPUs and NAMD*, Solution Brief, High Performance Computing, AMD, 2022, 4 pp. ↑425

[348] S. Pall, A. Alekseenko. "GROMACS 2023: Readiness on the AMD GPU Heterogeneous Platform", *PDC Newsletters*, 2023, no. 1. ↑425

[349] *Molecular Dynamics. Nvidia GPU Benchmarks AMBER 22*, Blog, Exxact, 2023. ↑425, 426

[350] J. Zeng, D. Zhang, D. Lu, P. Mo, Z. Li, Y. Chen, M. Rynik, L. Huang, Z. Li, S. Shi, Y. Wang,; Ye H., P. Tuo, J. Yang, Y. Ding, Y. Li, D. Tisi, Q. Zeng, H. Bao, Y. Xia, J. Huang, K. Muraoka, Y. Wang, J. Chang, F. Yuan, S. L. Bore, C. Cai, Y. Lin, B. Wang, J. Xu, J.-X. Zhu, C. Luo, Y. Zhang, R. E. A. Goodall, W. Liang, A. K. Singh, S. Yao, J. Zhang, R. Wentzcovitch, J. Han, J. Liu, W. Jia, D. M. York, W. E, R. Car, L. Zhang, H. Wang. "DeePMD-kit v2: A software package for deep potential models", *The Journal of chemical physics*, **159**:5 (2023), id. 054801. ↑426

[351] A. Prokopenko, P. Sao, D. Lebrun-Grandie. "A single-tree algorithm to compute the Euclidean minimum spanning tree on GPUs", *ICPP '22: Proceedings of the 51st International Conference on Parallel Processing* (29 August 2022–1 September 2022, Bordeaux, France), ACM, New York, 2022, ISBN 978-1-4503-9733-9, id. 14, 10 pp. ↑426

[352] A. Bagusetty, A. Panyala, G. Brown, J. Kirk. "Towards cross-platform portability of coupled-cluster methods with perturbative triples using SYCL", *2022 IEEE/ACM International Workshop on Performance, Portability and Productivity in HPC (P3HPC)* (13–18 November 2022, Dallas, TX, USA), IEEE, 2022, ISBN 978-1-6654-6021-7, pp. 81–88. ↑427

[353] A. Bussy, O. Schütt, J. Hutter. "Sparse tensor based nuclear gradients for periodic Hartree–Fock and low-scaling correlated wave function methods in the CP2K software package: A massively parallel and GPU accelerated implementation", *The Journal of Chemical Physics*, **158**:16 (Apr 28 2023), id. 164109. ↑428

[354] L. Mazur, D. Bollweg, D. A. Clarke, L. Altenkort, O. Kaczmarek, R. Larsen, T. Hai-Shu, J. Goswami, P. Scior, H. Sandmeyer, M. Neumann, H. Dick, S. Ali, J. Kim, C. Schmidt, P. Petreczky, S. Mukherjee. "SIMULATeQCD: A simple multi-GPU lattice code for QCD calculations", *Computer Physics Communications*, **300** (July 2024), id. 109164. ↑428

[355] S. Gottlieb, H. Jeong, A. Strelchenko. *Two-link staggered quark smearing in QUDA*, 2023, 10 pp. arXiv 2301.05518 ↑429

[356] P. Mullowney, S. Thomas, A. K. Carr, K. Swirydowicz, M. Day, L. Esclapez. *Novel solver algorithms for nearly singular linear systems arising in combustion modelling*, NREL/PR-2C00-81907, 2022 SIAM Conference on Parallel Processing for Scientific Computing (February 23, 2022), National Renewable Energy Lab. (NREL), 2022. ↑429

[357] R. Halver, C. Junghans, G. Sutmann. "Using heterogeneous GPU nodes with a Cabana-based implementation of MPCD", *Parallel Computing*, **117** (September 2023), id. 103033. ↑429

[358] M. Min, M. Brazell, A. Tomboulides, M. Churchfield, P. Fischer, M. Sprague. *Towards exascale for wind energy simulations*, 2022, 16 pp. arXiv 2210.00904 ↑432, 433

[359] T. Kolev, P. Fischer, A. Abdelfattah, N. Beams, J. Brown, S. Jean-Camier, R. Carson, N. Chalmers, V. Dobrev, Y. Dudouit, L. Ghaffari, A. Y. Joshi, S. Kerkemeier, Y.-H. Lan, D. Mc Dougall, D. Medina, M. Min, A. Mishra, W. Pazner, M. Phillips, T. Ratnayaka, M. S. Shephard, M. H. Siboni, C. W. Smith, J. L. Thompson, A. Tomboulides, S. Tomov, V. Tomov, T. Warburton. *High-order algorithmic developments and optimizations for more robust exascale applications*, ECP Milestone Report. WBS 2.2.6.06, Milestone CEED-MS38, 2022, 76 pp. URL ↑432, 433, 434

[360] G. R. J. Lesur, S. Baghdadi, G. Wafflard-Ernandez, J. Mauxion, C. M. T. Robert, M. Van den Bossche. "IDEFIX: a versatile performance-portable Godunov code for astrophysical flows", *Astronomy and Asrtrophysics*, **677** (September 2023), id. A9, 17 pp. doi ↑434, 435

[361] C. J. White, P. D. Mullen, F. Yan-Jiang, S. W. Davis, J. M. Stone, V. Morozova, L. Zhang, *The Astrophysical Journal*, **949**:2 (2023), id. 103, 29 pp. doi ↑435

[362] P. Grete, J. C. Dolence, J. M. Miller, J. Brown, B. Ryan, A. Gaspar, F. Glines, S. Swaminarayan, J. Lippuner, C. J. Solomon, G. Shipman, C. Junghans, D. Holladay, J. M. Stone, L. F. Roberts. "Parthenon—a performance portable block-structured adaptive mesh refinement framework", *The International Journal of High Performance Computing Applications*, **37**:5 (2023), pp. 465–486. doi ↑435

[363] N. Schild, M. Räth, S. Eibl, K. Hallatschek, K. Kormann. "A performance portable implementation of the semi-Lagrangian algorithm in six dimensions", *Computer Physics Communications*, **295** (February 2024), id. 108973. doi ↑435

[364] I. Sfiligoi, E. A. Belli, J. Candy, R. D. Budiardja. "Optimization and Portability of a Fusion OpenACC-based Fortran HPC code from Nvidia to AMD GPUs", *PEARC '23: Practice and Experience in Advanced Research Computing* (Portland, OR, USA, July 23–27, 2023), ACM, New York, July 2023, ISBN 978-1-4503-9985-2, pp. 246–250. doi ↑435, 436

[365] S. Diederichs, C. Benedetti, A. Huebl, R. Lehe, A. Myers, A. Sinn, J.-Vay L., W. Zhang, M. Thévenet. "HiPACE++: a portable, 3D quasi-static particle-in-cell code", *Computer Physics Communications*, **278** (September 2022), id. 108421. doi ↑436

[366] *Breaking Barriers in Plasma Physics with PIConGPU and AMD Instinc MI250 GPU*, Solution Brief, High Performance Computing, AMD, 2023, 4 pp. URL ↑436

[367] L. Fedeli, A. Huebl, F. Boillod-Cerneux, T. Clark, K. Gott, C. Hillairet, S. Jaure, A. Leblanc, R. Lehe, A. Myers, C. Piechurski, M. Sato, N. Zaim, W. Zhang, L. Jean-Vay, H. Vincenti. "Pushing the Frontier in the design of laser-based electron accelerators with groundbreaking mesh-refined particle-in-cell simulations on exascale-class supercomputers", *2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)* (Dallas, Texas, USA, November 13–18, 2022), IEEE, 2022, id. 3, 12 pp. ↑436

[368] A. Huebl, R. Lehe, E. Zoni, O. Shapoval, R. T. Sandberg, M. Garten, A. Formenti, R. Jambunathan, P. Kumar, K. Gott, A. Myers, W. Zhang, A. Almgren, C. E. Mitchell, J. Qiang, D. Grote, A. Sinn, S. Diederichs, M. Thevenet, L. Fedeli, T. Clark, N. Zaim, H. Vincenti, L. Jean-Vay,. *From compact plasma particle sources to advanced accelerators with modeling at exascale*, 2023, 4 pp. arXiv 2303.12873 ↑436

[369] M. F. Adams, P. Wang, J. Merson, K. Huck, M. G. Knepley. *A performance portable, fully implicit Landau collision operator with batched linear solvers*, 2024, 20 pp. arXiv 2209.03228 ↑436

[370] O. Thawakar, R. M. Anwer, J. Laaksonen, O. Reiner, M. Shah, F. S. Khan. *3D mitochondria instance segmentation with spatio-temporal transformers*, 2023, 10 pp. arXiv 2303.12073 ↑437

[371] D. Samuel, A. Kutuzov, S. Touileb, E. Velldal, Øvrelid L., Rønningstad E., E. Sigdel, A. Palatkina. *NorBench–A benchmark for Norwegian language models*, 2023, 16 pp. arXiv 2305.03880 ↑437

[372] L. Yankovskaya, M. Tars, A. Tättar, M. Fishel. "Machine translation for low-resource Finno-Ugric languages", *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)* (Tórshavn, Faroe Islands, May 22–24, 2023), University of Tartu Library, 2023, ISBN 978-99-1621-999-7, pp. 762–771. ↑437

[373] L. Charpentier, S. Wold, D. Samuel, E. Rønningstad. "BRENT: Bidirectional retrieval enhanced Norwegian transformer", *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)* (Tórshavn, Faroe Islands, May 22–24, 2023), University of Tartu Library, 2023, ISBN 978-99-1621-999-7, pp. 202–214. ↑437

[374] D. Samuel, L. Øvrelid. "Tokenization with factorized subword encoding", *Findings of the Association for Computational Linguistics: ACL 2023* (Toronto, Canada, July 9–14, 2023), ACL, 2023, ISBN 9781959429623, pp. 14143–14161. ↑437

[375] *AMD Radeon Instinct MI300*, GPU Specs Database, TechPowerUp. ↑438

[376] S. Naffziger, N. Beck, T. Burd, K. Lepak, G. H. Loh, M. Subramony, S. White. "Pioneering chiplet technology and design for the AMD EPYC™ and Ryzen™ processor families: Industrial product", *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)* (Valencia, Spain, 14–18 June 2021), IEEE, 2021, ISBN 978-1-4503-9086-6, pp. 57–70. ↑439

Appendix. Abbreviations used in sections of the review (for GPUs from different manufacturers)

Abbreviations for hardware

ALU Arithmetic Logic Unit *1, 5.1*

ASIC Application-Specific Integrated Circuit *1*

CXL Compute Exress Link . . . . . *Introduction, 2, 3.1*

CPU Central Processing Unit . . . . . *Introduction, 1, 2, 3.1, 3.2, 3.3, 4.1, 4.2, 5, 5.1, 5.2, 5.3, 5.4*

ECC Error-correcting Code *4.1, 4.2*

ECP Exascale Compute Project *3.1, 4.1, 5.3*

EPI European Processor Initiative . . . . . *Introduction*

GPU Graphics Processing Unit . . . . . *Introduction, 1, 2, 3, 3.1, 3.2, 3.3, 4, 4.1, 4.2, 4.3, 5, 5.1, 5.2, 5.3, 5.4*

GPGPU General-purpose GPU . . . . . *Introduction, 3.1, 4.1, 4.2*

HBM High Bandwidth Memory *3.1, 4.1, 4.2, 5, 5.3*

ISA Instruction Set Architecture *1, 3.1, 4.1, 4.2, 5.1*

NUMA Non-Uniform Memory Access *2, 4.2, 5.1*

OoO Out-of-order *4.2*

PIM Processing-in-memory *2, 3.1, 4.1*

SIMT Single Instruction, Multiple Threads *1, 4.1*

SoC System-on-Chip *3.1, 4.2*

SVE2 Scalable Vector Extension 2 (ARM) *4.2*

TCO Total Cost of Ownership . . . . . *Introduction, 5.3, 5.4*

TDP Thermal Design Power *2, 3.1, 3.3, 4.1, 4.2, 5, 5.1*

Abbreviations for mathematical and programming fields

API Application Programming Interface . . . *1, 4.1, 5.1, 5.2*

BLAS Basic Linear Algebra Subprograms *1, 4.1, 5.3*

FFT Fast Fourier Transform *4.1, 5.3*

HPC High Performance Calculations . . . . . *Introduction, 1, 3.1, 3.2, 3.3, 4, 4.1, 4.2, 4.3, 5.1, 5.2, 5.3, 5.4*

HPCG High Performance Conjugate Gradients 5.3

HPL High Performance Linpack . . . . . Introduction, 3.2, 5.3

LLVM Low Level Virtual Machine (early abbreviation; now — a famous set of compilers and tools) 4.1, 4.2, 5.2

PGAS Partitioned Global Address Space 4.1

SDK Software Development Kit Introduction, 1, 3, 4.1, 5.2, 5.3, 5.4

Abbreviations for areas of use of GPUs and applications that use them

CFD Computational Fluid Dynamics . . . 3.2, 4.1, 4.2, 5.2, 5.3

DFT Density Functional Theory . . . . . Introduction, 4.1, 5.4

DLRM Deep Learning Recommendation Models . . 4.1, 4.2, 5.3

BERT Bidirectional Encoder Representations from Transformers 2, 4.2, 5.3

NLP Natural Language Processing 2, 4.2

Abbreviations common to GPUs from different companies

SU Special Function Units 4.1

OAM OpenCompute Accelerator Module . . . . . Introduction, 2, 3.1, 5.1, 5.3, 5.4

Abbreviations for Nvidia GPUs

CUDA Compute Unified Device Architecture . . . . . Introduction, 1, 2, 3.2, 3.3, 4.1, 4.2, 5, 5.1, 5.2, 5.3, 5.4

GPC GPU (*or* Graphics) Processing Clusters 2, 4.1, 4.2

GTC GPU Technology Conference . . . . . Introduction

MIG Multi-Instance GPU 2, 4.1, 4.2

NCCL Nvidia Collective Communications Library 4.1, 4.2, 5.2

PTX Parallel Threads execution 4.1, 4.2, 5.3

SM Streaming Multiprocessor 1, 3.1, 4, 4.1, 4.2, 5, 5.1, 5.2, 5.3

SXM Server PCI Exress Module 2, 3.1, 3.3, 4.1, 4.2, 5, 5.1, 5.3, 5.4

TPC Texture Processing Clusters 4.1, 4.2

Abbreviations for Intel GPUs

AIC Add-in Card *3.1*

DPC++ Data Parallel C++ *Introduction, 1, 3.2, 3.3, 4.1, 5.2, 5.3, 5.4*

DPCT DPC++ Compatibility Tool *3.2, 5.2*

PVC Ponte Vecchio *3, 3.1, 3.2, 3.3, 5.1, 5.4*

Abbreviations for AMD GPUs

CU Compute Unit *1, 2, 5, 5.1, 5.2, 5.3*

GCD Graphics Compute Die *3.1, 5.1, 5.3, 5.4*

HIP Heterogeneous Computing Interface for Portability . . . . . *Introduction, 1, 3.2, 3.3, 4.1, 5.1, 5.2, 5.3*

Abbreviations for GPU BR100

CU *1, 2, 5, 5.1, 5.2, 5.3*

SVI Secure Virtual Instance *2*

Received 16.10.2023;
approved after reviewing 24.01.2024;
accepted for publication 01.03.2024;
published online 28.06.2024.

Recommended by *prof. Sergey M. Abramov*

Information about the author:

Mikhail Borisovich Kuzminsky

Senior Researcher, Laboratory of Computer Software for Chemical Research, Candidate of Chemical Sciences, Institute of Organic Chemistry, Russian Academy of Sciences. The scientific interests are high-performance computing, computer hardware, computational chemistry.

0000-0002-3944-8203

e-mail: *kus@free.net*

The author declare no conflicts of interests.

RU EN ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ ISSN 2079-3316
научная статья ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В МЕДИЦИНЕ

УДК 61:007
10.25209/2079-3316-2024-15-2-475-492

Архитектура взаимодействия в медицинской экосистеме

Владимир Леонидович **Малых**[1]✉, Алексей Николаевич **Калинин**[2], Сергей Владимирович **Рудецкий**[3]

Аннотация. В сфере медицинской информатики наблюдается устойчивая тенденция к формированию сложной многокомпонентной экосистемы. На передний край выходят проблемы взаимодействия и интеграции компонентов экосистемы: медицинских, лабораторных, радиологических систем, ЕГИСЗ, ЕМИАС, МДЛП, различных регистров и сервисов, в том числе реализующих подходы ИИ к обработке данных и решению задач. Пациенты остро нуждаются в личных кабинетах, интегрирующих их медицинские данные, становятся активными участника экосистемы. Решать интеграционные задачи приходится в сильно неоднородной информационной среде, когда становится недостижимым обеспечение синхронного интерактивного взаимодействия между участниками экосистемы. Для отдельных приложений необходима гибкая возможность сочетания как синхронного, так и асинхронного взаимодействия, выбираемого ситуационно, исходя из конкретных временных задержек и характеристик взаимодействия.

В статье предлагается специальная архитектура, позволяющая реализовывать синхронное и асинхронное взаимодействие между участниками экосистемы. Адаптация ПО, рассчитанного только на синхронное взаимодействие, под асинхронную архитектуру не требует радикальной переделки ПО. Подход отрабатывался на примере адаптации модуля МДЛП МИС Интерин к работе во внутренней защищенной сети медицинского центра. Предложенная архитектура может быть использована разработчиками ПО и в других сферах деятельности, где идёт активное развитие экосистем, сопровождающееся ростом интеграционных взаимодействий.

Ключевые слова и фразы: медицинская информатика, медицинские информационные системы, цифровая экосистема, интеграция, асинхронное взаимодействие, личный кабинет пациента

Благодарности:

Для цитирования: Малых В. Л., Калинин А. Н., Рудецкий С. В. *Архитектура взаимодействия в медицинской экосистеме* // Программные системы: теория и приложения. 2024. **Т. 15**. № 2(61). С. 475–492. https://psta.psiras.ru/read/psta2024_2_475-492.pdf

Введение

В системном обзоре цифрового здравоохранения США [1] отмечаются происходящие и ожидаемые революционные изменения в индивидуальном здравоохранении и медицинском менеджменте, стратегии здравоохранения для всего населения. «Такие разработки, как облачные вычисления, искусственный интеллект, машинное обучение, блокчейн, цифровая диагностика и лечение, телемедицина и ориентированные на потребителя мобильные приложения для здравоохранения, теперь регулярно используются в самоконтроле, здравоохранении и биомедицинской науке» (пер. Яндекс).

Речь идет о самой богатой стране мира, чьи расходы на здравоохранение в перспективе к 2028 году прогнозируются на уровне 20% от валового внутреннего продукта, стране, задающей тренды развития цифрового здравоохранения для всего мира. Однако отмечается и множество проблем: недостаточная совместимость цифровых технологий, изолированность и недоступность источников медицинских данных, низкая эффективность цифровых интерфейсов и инструментов для помощи пациентам за пределами клиник. Всё это тормозит реализацию концепции обучающейся системы здравоохранения, которая ведёт сбор фактических данных в режиме реального времени, связывает наборы данных и анализирует их с использованием искусственного интеллекта и машинного обучения.

На передний край в цифровой медицине выходит проблема интероперабельности. С этой целью в США разработаны стандарты: Health Level 7 Fast Healthcare Interoperability Resources (FHIR) (HL7 International, n.d.), SNOMED (SNOMED International, n.d.), RxNorm (NLM, 2022), United States Core Data for Interoperability (USCDI). И тем не менее, широкая совместимость информационных платформ здравоохранения является неполной во многом из-за неполной доступности медицинских записей, отсутствия стандартов терминологии и проблем обмена данными между системами здравоохранения.

Техническую интероперабельность (способность систем обмениваться данными) можно обеспечить стандартизацией данных, реализацией API, использованием стандартных сетевых протоколов. Но очень трудно, практически невозможно на данном этапе, обеспечить семантическую интероперабельность, когда система не только в состоянии принять данные в достаточно свободном формате, но и «понимает» их.

Можно привести поучительный пример когнитивной системы от IBM Doctor Watson. Дебютировав в 2011 Watson Health, обещала революционные изменения в здравоохранении. Декларировалась способность системы

к самообучению, усваиванию знаний из различных открытых источников данных, в том числе и из научных публикаций, свободное общение с системой на естественном языке, т. е. предполагалось достижение той самой семантической интероперабельности в сфере здравоохранения. Но именно для здравоохранения ничего из этого не сработало. В результате, по образному выражению некоторых западных аналитиков, система Watson Health была продана «по частям, на металлолом».

В настоящее время IBM переосмысливает применение технологий Watson, но уже без прежних амбиций для здравоохранения (Steve Lohr, The New York Times, July 17, 2021). Не следует возлагать слишком большие надежды и на применение технологий ChatGPT. Как отмечают специалисты, в 3–5% случаев эта технология будет выдавать на ваши вопросы совершенно неадекватные ответы, что совершенно не допустимо для сферы здравоохранения. Достаточно привести *пример из сети*, что GPT «с лёгкостью» создала описание операции удаления головы по Вишневскому с указанием показаний к операции и этапов операции. Поэтому на данном технологическом этапе развития цифрового здравоохранения и в данной статье мы сосредоточимся на технических аспектах взаимодействия систем и опускаем вопросы семантической интероперабельности.

«Проблемы кибербезопасности и конфиденциальности являются основными препятствиями на пути внедрения цифрового здравоохранения, продолжают подрывать доверие пациентов и усиливают нежелание систем здравоохранения делиться данными» … Эти критические вызовы требуют технологических, управленческих и юридических протоколов. Государственно-частное партнерство необходимо для разработки надстроечной структуры, обеспечивающей безопасность и защиту персональных данных в сфере цифрового здравоохранения» [1] (пер. Яндекс). Один из выводов авторов [1] – существующее взаимодействие информационных систем здравоохранения не обеспечивает адекватной поддержки оптимального долгосрочного оказания медицинской помощи и не способствует удовлетворению потребностей США в области здравоохранения.

Если посмотреть концептуальные работы [2, 3] по стратегии развития цифрового здравоохранения в других странах (Индия и ЮАР), то можно заметить, что их концепции хорошо согласуются с концепциями развития здравоохранения для США [1]. Вполне обоснованно можно сделать вывод, что возникло согласованное и разделяемое развитыми странами концептуальное представление о будущем мирового цифрового здравоохранения.

В современной терминологии для обозначения нового формирующегося цифрового здравоохранения используется термин «медицинская

экосистема» (healthcare ecosystems, digital healthcare ecosystems). Цифровое здравоохранение понимается широко, оно охватывает данные, собираемые электронным способом, техническую и коммуникационную инфраструктуру и приложения в экосистеме здравоохранения. Лаконичное определение медицинской экосистемы – это сложная сеть, объединяющая организации, людей и технологии, для предоставления пациентам медицинских услуг.

Россия также не остаётся в стороне от мирового тренда в здравоохранении. Ведётся концептуальная разработка отечественной экосистемы медицинской помощи [4]. Основная особенность медицинских экосистем – это множество разнотипных участников (key players): пациенты, врачи, медсёстры, различные лечебные и диагностические учреждения, лаборатории, различные государственные агентства (Росздравнадзор и др.), фармакологические компании, производители медицинских устройств, провайдеры различных медицинских услуг и сервисов, операторы ЭДО, отдельно отметим для нашей страны Федеральную государственную информационную систему в сфере здравоохранения ЕГИСЗ и Федеральную государственную информационную систему мониторинга движения лекарственных препаратов ФГИС МДЛП. Основными участниками медицинской экосистемы являются МИС [5].

В настоящее время в России происходят значительные изменения, связанные с формированием современной развитой цифровой среды. Облачный рынок в России сохранил динамику роста в 2023 году, по оценкам Linx рост составил 30%. По данным «Открытых систем» «стимулирующее влияние на облачный рынок по-прежнему будут оказывать такие факторы, как санкционное воздействие, стимулирующее спрос на отечественное ПО и оборудование, а также нарастающая цифровая трансформация предприятий в самых разных отраслях и интенсивное развитие технологий на базе искусственного интеллекта». Президент объявил о запуске национального проекта «Экономика данных». На формирование цифровых платформ во всех ключевых отраслях экономики и социальной сферы будет направлено не менее 700 млрд руб. Предполагается создание на технологии блокчейна аналога системы Свифт для стран членов БРИКС. Всё это даёт уверенность в сформировании в ближайшем времени подходящей цифровой среды для медицинских экосистем.

Всё ещё большой проблемой для медицинских экосистем остаётся формирование системы прав и привилегий участников. Кто и что может запросить у других участников, что и кому можно сообщить. Выше отмечалось, что для решения этой проблемы требуется разработка технологических, управленческих и юридических протоколов, требуется участие и поддержка государства. Эту проблему мы оставляем за рамками статьи.

Важна также и архитектура взаимодействия участников медицинской экосистемы, отдельному аспекту этой архитектуры и посвящена данная статья.

Взаимодействие в такой сложной неоднородной среде становится проблемой. Практически с каждым участником экосистемы приходится «говорить» на его «личном» языке, например ФГИС МДЛП имеет свой документированный API для взаимодействия. В самом информационном взаимодействии могут наблюдаться значительные временные задержки (latency). При этом задержки не сетевые, они связаны либо со значительным временем отработки запроса (построение статистических отчётов, обработка больших массивов данных, использование ресурсозатратных вычислительных методов, в том числе методов ИИ), либо с особенностями расположения и работы некоторых участников в защищенных закрытых сетях. Последнее свойственно для ведомственной медицины (Банк России, Федеральная таможенная служба, Министерство обороны и др.), с которой у авторов имеется многолетний опыт работы, и где выход во внешний мир может осуществляться через отдельные шлюзы, обеспечивающие безопасность информационного обмена. Можем привести пример такого обмена, когда гарантированное время доставки сообщения из внутренней сети в МДЛП через шлюз, работающий в пакетном режиме, составляло до 45 минут (на практике оказалось меньше, но всё равно могло быть значительным).

Следует также отметить, что появляются асинхронные виды оказания медицинской помощи, в частности асинхронные медицинские консультации, см. например *«Асинхронные телемедицинские услуги - преимущества и ограничения»*: «Асинхронная телемедицина или телемедицина с промежуточным хранением (метод коммутации “store-and-forward”), относится к типу общения между пациентами и поставщиками медицинских услуг, которое не ведётся в режиме реального времени. Этот метод предполагает безопасный обмен электронными сообщениями или отправку заранее записанной информации и документов, которые медицинские работники рассматривают позже».

Указанные особенности требуют построения особой архитектуры взаимодействия в медицинской экосистеме. В условиях плавающих временных задержек архитектура должна позволять пользователям работать как синхронно, так и асинхронно, настраиваясь ситуационно автоматически на моду информационного обмена. Одни и те же модули должны поддерживать оба режима работы.

Задача обеспечения взаимодействия в информационной среде не нова. Разработаны определённые классы ПО для её решения: Request Management Software, Service Request Management Software, Data Management

Tools, Business Intelligence, разрабатываются интеграционные платформы - Integration Platforms, Big Data Integration Platforms Interconnected, см. *обзор*. В многолетних разработках этих классов ПО участвуют ведущие западные компании: Amazon, Google, IBM, Microsoft, Oracle, SAP, SAS и др. Проблема использования западного ПО в нашей стране и необходимость отказа от него общеизвестна, см. работу [6]. Однако путь к импортозамещению оказался чрезвычайно трудным. Как отмечено в [7] нет отечественных аналогов зарубежных систем различных классов, в том числе и перечисленных выше.

Отечественная медицинская экосистема только начала формироваться, ведётся разработка отечественных базовых интеграционных платформенных решений для медицины (Нетрика, Ростелеком). В этих условиях в [7] предлагается двигаться поступательно с использованием решений Open Source, вести разработку собственных платформ автоматизации. Соглашаясь с [7], авторы видят обоснованной разработку принципов построения собственной интеграционной архитектуры для участников отечественной экосистемы медицинской помощи. Задача построения отечественной медицинской интеграционной платформы чрезвычайно важна и трудоёмка, и двигаться к её решению можно лишь постепенно, поэтому мы рассмотрим лишь отдельные аспекты этой проблемы и перейдём к рассмотрению одного из базовых принципов построения интеграционной архитектуры.

1. Базовый принцип интеграционной архитектуры

В не устаревшей работе [8] представлены широко известные принципы построения интеграционной архитектуры. Мы будем следовать архитектуре, основанной на асинхронном обмене сообщениями. Это зарекомендовавший себя подход, созданный специально для интеграции информационных систем. Недостатком такого подхода по мнению [8] является высокая цена разработки, необходимость реализовывать адаптеры между системой доставки и приложениями.

Будем описывать архитектуру с привлечением понятия интеграционной платформы взаимодействий (ИПВ). Концептуально задача ИПВ, получить от участника экосистемы сообщение (запрос), передать его адресату (другому участнику, иногда даже самому себе), получить ответное сообщение (результат), если оно предполагается, и вернуть ответ запрашивающей стороне. ИПВ должна решать проблемы временной невозможности доставок сообщений адресатам, например в силу неготовности адресата к приёму. Для этого нужно предусмотреть возможность откладывания доставки сообщений с дальнейшими попытками доставки

с использованием различных стратегий доставки отложенных сообщений. В дальнейшем изложении также будем использовать термин «запрос» как синоним к термину «сообщение».

Основная топология интеграционного узла может быть «хаб + спицы», где роль хаба исполняет ИПВ. Но не исключается и взаимодействие «точка - точка» в вариантах «ИПВ – участники взаимодействия» и «ИПВ – ИПВ». Ниже будет приведен пример апробации одного из вариантов такой архитектуры.

В условиях невозможности обеспечения гарантированного времени ответа от внешних систем для синхронной работы и возможных задержек в передаче данных между ИПВ и внутренней сетью следует предусмотреть базовую возможность работы платформы в асинхронном режиме. Подобная архитектура была разработана для многофункционального медицинского центра при реализации взаимодействия модуля МДЛП из внутренней сети через шлюз с внешней системой МДЛП.

Основные черты архитектуры:

(*1*) К ИПВ направляется типизированный запрос (сообщение, объект, документ, файл...) с указанием адресата (участника взаимодействия). Содержание запроса и его формат для общего описания архитектуры не важны.

(*2*) Каждый запрос получает идентификатор. Механизм назначения идентификаторов запросов описан ниже.

(*3*) Идентификатор запроса формируется как значение хеш-функции, основанной на содержании запроса, при этом хеш строится только по существенным атрибутам запроса, исключая незначимые для сути запроса "временные"компоненты (дату и время запроса, автора запроса и т. п.).

Например, пациент запрашивает по известному идентификатору заключение врача-диагноста. Запрашиваемый документ находится в терминальном состоянии (подписан, возможно, квалифицированной ЭЦП, изменение документа не предполагается). В хеш запроса включим тип и идентификатор запрашиваемого документа. Важно, чтобы хеш уникально характеризовал запрос, и при повторении запроса мы бы получили то же самое значение хеша.

(*4*) Запросы, имеющие одинаковое значение идентификатора (одинаковое значение хеш-функции), считаются идентичными.

Пункты 3) и 4) являются основой архитектуры. Они позволяют каждому запросу присваивать идентификатор – значение хеш-функции. Какие запросы считать идентичными, какие характеристики должны

быть включены в хеш решают авторы запросов, потому что только они знают и понимают семантику запросов. Поэтому прерогатива построения идентификатора (хеша) относится только к авторам и не затрагивает ИПВ. Например, документ целиком может войти в состав данных, по которым формируется хеш. В работе [9] отмечено использование хеш-функций в блокчейн-процессе передачи медицинских данных для решения проблемы дублирования записей о пациенте. Мы используем это полезное свойство хеш-функции для выделения идентичных запросов.

(5) Адресат по запросу может вернуть ИПВ результат.

(6) Возвращаемый результат связывается с запросом и передаётся ИПВ, где сохраняется в БД в хранилище результатов запросов.

(7) Каждый результат типизирован и ему сопоставляется время жизни (время актуальности результата).

Пользовательские интерфейсы, генерирующие запросы к другим участникам, работают в асинхронном режиме следующим образом:

(1) Перед отправкой запросу присваивается идентификатор, например вышеописанным методом вычисления хеш-функции;

(2) Запрос направляется к ИПВ;

(3) ИПВ проверит, есть ли в хранилище результатов актуальный результат для данного значения идентификатора; если результат есть, то он возвращается как ответ на запрос.

(4) Если результата для данного значения идентификатора нет, то делается проверка не является ли запрос повторным, если запрос является повторным и время прошедшее с момента отправки более раннего запроса не превысило время актуальности ожидаемого результата, то запрос игнорируется.

Время актуальности ожидаемого результата – это время прошедшее от времени оправки запроса до получения результата считающегося актуальным. Например, результат запроса подписанного и не подлежащего изменению документа может иметь неограниченное время актуальности. Результат запроса свободных талонов для записи к врачу очевидно имеет ограниченное время актуальности. Определение времени актуальности ожидаемого результата для различных запросов может вызывать затруднения. В этих случаях можно предоставить оценку актуальности результата пользователю. Результаты, помеченные пользователем как не актуальные, не будут влиять на прохождение идентичных запросов.

(5) Если запрос является повторным и время прошедшее с момента отправки более раннего запроса превысило время актуальности ожидаемого результата, то ИПВ отправляет запрос адресату повторно.

Пользовательский интерфейс, создающий запросы к внешним системам, работает следующим образом:

(1) Запрос отправляется ИПВ (менеджеру запросов) и ожидается ответ до стандартного таймаута.

(2) Если ответ за время таймаута не поступает, то пользователю (интерфейсу) выдается сообщение о переходе запроса в асинхронный режим ожидания ответа.

(3) Запрос, который не удалось отработать в синхронном режиме, должен быть повторен запрашивающей стороной (автором, участником взаимодействия) и при наличии актуального результата запрашивающей стороной будет получен ответ на запрос.

(4) Следует также предусмотреть «силовое» выполнение запроса (force request), когда пользователь имеет возможность отправить повторный запрос независимо от нахождения в хранилище актуального результата для этого запроса.

Соответственно в архитектуру ИПВ входят следующие основные компоненты:

(1) менеджер запросов;

(2) хранилище запросов и результатов;

(3) адаптеры, предназначенные для работы с различными участниками экосистемы.

Архитектура обеспечивает взаимодействие между участниками экосистемы с помощью ИПВ, см. рисунок 1. Архитектура позволяет

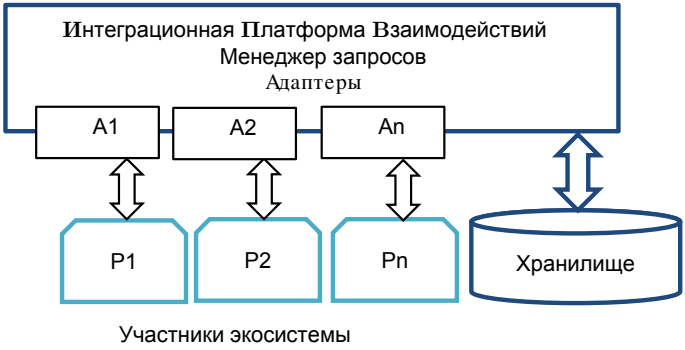


Рисунок 1. Архитектура интеграционного узла экосистемы

"бесшовно сшить"два режима работы синхронный и асинхронный, при отсутствии критических временных задержек архитектура будет работать в наиболее благоприятном для пользователей синхронном режиме.

ИПВ сама становится компонентой экосистемы. Экземпляры ИПВ могут общаться друг с другом напрямую, передавая сообщения участникам подключенным через адаптеры к экземплярам ИПВ. Экземпляры ИПВ должны обладать возможностями масштабироваться. Например, для крупных федеральных систем (участников) соответствующие экземпляры ИПВ должны обладать соответствующей производительностью обработки больших потоков заявок и ёмкостью хранилищ. Для решения локальных интеграционных задач медицинских учреждений могут применяться экземпляры ИПВ с более низкими характеристиками. Поскольку имеется полная аналогия архитектуры с обычной почтой, то разные по масштабу и характеристикам ИПВ можно ассоциировать с главпочтамтами и местными небольшими почтовыми отделениями. Каждый экземпляр ИПВ доставляет сообщения определённому набору участников и соответственно нуждается в определённом наборе адаптеров.

На рисунке 2 представлена общая интеграционная архитектура экосистемы, адаптеры и хранилища на рисунке опущены.

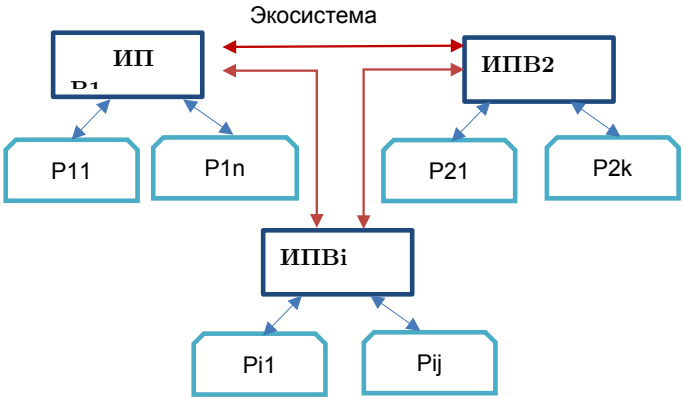


Рисунок 2. Общая интеграционная архитектура экосистемы

Проблемы адресации, маршрутизации доставки сообщений, защищенных и безопасных протоколов передачи данных в экосистеме могут быть решены уже имеющимися средствами сети Интернет.

2. Асинхронная архитектура личного кабинета пациента

Роль пациента в экосистеме оказания медицинской помощи чрезвычайно выросла. Пациент становится активным участником экосистемы, на передний план выходит заинтересованность пациента в сохранении и

поддержании собственного здоровья, принципы превентивной медицины. Экосистема должна предоставлять пациентам личные кабинеты, через которые они могут взаимодействовать с другими участниками экосистемы [10].

«Отдельные люди и семьи привыкли к мобильным и онлайн-инструментам в других аспектах своей жизни, таких как бронирование авиабилетов, автомобильные услуги и банковское дело. Развитие надежного партнерства между отдельными лицами, семьями и поставщиками медицинских услуг требует дальнейшего внедрения систем, которые функционируют так же, как и эти другие инструменты, предлагая ориентированную на пациента, простую и безопасную двустороннюю связь для записи на прием, самостоятельной регистрации и опросов для обратной связи...

Полное вовлечение людей в заботу об их здоровье и благополучии с помощью цифрового здравоохранения, удовлетворение общественного спроса на участие в растущей экосистеме цифрового здравоохранения... является приоритетом в достижении полностью реализованного будущего цифрового здравоохранения. Медицинские данные носят исключительно личный характер, и непреднамеренное или злонамеренное раскрытие этих данных может перевернуть жизнь человека. Использование всего потенциала цифрового здравоохранения потребует широкого доверия к системам здравоохранения и коммерческим предприятиям для защиты человека от негативных последствий» [1] (перевод Яндекс). Данные, обрабатываемые в МИС, относятся законодательством (ФЗ-152) к персональным данным специального характера, и подлежат защите по самому высокому классу.

Пациенты являются самыми многочисленными участниками экосистемы. Понятно, что у пациентов не будет «личных» адаптеров и все они будут решать свои интеграционные задачи через предусмотренные архитектурой универсальные адаптеры.

Основные проблемы, связанные с персональными медицинскими данными пациентов, сводятся к защите персональной информации, обеспечению права пациентов на получение своих персональных данных, идентификации пациентов, как участников экосистемы.

При проектировании платформ цифрового здравоохранения необходимо переходить на принципы проектирования в расчете на конечного пользователя, включая вовлеченность конечных пользователей в процесс разработки и в поддержание жизненного цикла ПО. «Помимо индивидуального управления данными о состоянии здоровья, вовлечение потребителей в их собственное здоровье и медицинскую помощь с помощью цифровых платформ потребует от разработчиков систем и руководителей

систем здравоохранения учета мнения клиентов при разработке, внедрении и оценке инструментов и платформ цифрового здравоохранения» [1] (перевод Яндекс).

Перейдем от общесистемных принципов построения цифровых ориентированных на пациентов экосистем здравоохранения к частной цели статьи и покажем возможности асинхронной архитектуры для реализации ЛК пациентов.

Ниже приводится описание принципов работы ЛК в асинхронном режиме применительно к многофункциональному медицинскому центру. ЛК через шлюз должен работать с АС МЦ, находящейся во внутренней защищенной сети. Особенности работы шлюза таковы, что запросы от ЛК могут приходить во внутреннюю сеть с большой задержкой, поэтому необходима возможность работы ЛК в асинхронном режиме.

(1) Из личного кабинета создается типизированный запрос на получение данных (консультации пациента, список результатов исследований, свободные талоны записи к специалистам, запрос документа, др.) к внутренней сети МЦ, обращаясь к ИПВ.
(2) Запрос хешируется, при этом хеш строится только по существенным атрибутам объекта, исключая незначимые для сути запроса “временные” компоненты (дату и время запроса, автора запроса и т.п.).
(3) Запросы, имеющие одинаковое значение хеш-функции, считаются идентичными.
(4) Далее ИПВ обращается к внутреннему шлюзу.
(5) Внутренний шлюз передает объект запроса к GraphQL серверу МИС, который обрабатывает только запросы, относящиеся к взаимодействию Личного кабинета и МИС.
(6) Возвращаемый результат связывается с запросом (значением хеш-функции) и передаётся через шлюз -> ИПВ - > в Личный кабинет, где сохраняется в БД в хранилище результатов запросов.
(7) Каждый результат типизирован и ему сопоставляется время жизни (время актуальности результата);

Пользовательские интерфейсы, генерирующие запросы из внешней сети к внутренней сети медицинской организации, работают в асинхронном режиме следующим образом:

(1) Перед отправкой запроса вычисляется его хеш-функция.
(2) ИПВ проверяет, есть ли в хранилище результатов актуальный результат для данного значения хеш-функции, если результат есть, то он возвращается как ответ на запрос.

(3) Если результата для данного значения хеш-функции нет, то делается проверка не является ли запрос повторным, если запрос является повторным и время прошедшее с момента отправки более раннего запроса не превысило время актуальности ожидаемого результата, то запрос игнорируется.

(4) Если запрос является повторным и время прошедшее с момента отправки более раннего запроса превысило время актуальности ожидаемого результата, то запрос отправляется через в ИПВ далее через шлюз в МЦ.

Пользовательский интерфейс, создающий запросы к внешним системам, работает следующим образом:

(1) Запрос отправляется менеджеру запросов и ожидается ответ до стандартного таймаута.

(2) Если ответ за время таймаута не поступает, то пользователю (интерфейсу) выдается сообщение «Ваша заявка принята, ожидайте ответа от МИС», далее в асинхронном режиме ответ будет получен и отобразится в разделе запроса информации.

(3) Запрос, который не удалось отработать в синхронном режиме, должен быть повторен и при наличии актуального результата будет получен ответ на запрос.

Если больших (больше таймаута пользовательского интерфейса) временных задержек при отработке запросов ЛК не будет, то ЛК будет работать в наиболее благоприятном для пользователей синхронном режиме.

Сайт МЦ может быть построен на тех же принципах, что и ЛК, и должен иметь возможность работы в асинхронном режиме.

3. Апробация архитектуры и программное решение

Масштаб задачи построения общей интеграционной архитектуры для экосистемы медицинской помощи накладывал значительные ограничения в части возможности полной детальной разработки архитектуры и её прототипирования исследовательским центром медицинской информатики ИПС РАН совместно с ООО «Интерин технологии». Было решено апробировать лишь отдельное архитектурное решение на локальной интеграционной задаче. Требовалось построить асинхронную архитектуру взаимодействия модуля МДЛП, находящегося во внутренней закрытой сети многофункционального медицинского центра, с ФГИС МДЛП. Между внутренней и внешней сетью МЦ данные передавались через особый шлюз, что не исключало значительные временные задержки при

доставке сообщений. Поэтому для решения интеграционной задачи как нельзя лучше подходили вышеописанные возможности архитектуры в части асинхронной работы.

Прототип ИПВ был реализован на языке Java на сервере приложения Java Apache Tomcat 8 с использованием OpenJDK8, хранилищем данных для ИПВ выступала СУБД Oracle. Клиентское ПО модуля МДЛП было реализовано на языке JavaScript в среде Интерин Promis Alpha [11] разработки ООО «Интерин технологии».

Адаптация модуля МДЛП под асинхронную работу была выполнена очень быстро и не потребовала серьёзных переделок модуля. Компонент ИПВ был реализован за 2 месяца. Вычислительные эксперименты полностью подтвердили принятую концепцию архитектуры. Модуль МДЛП позволял работать во внутренней защищённой сети как в асинхронном, так и в синхронном режиме, в зависимости от задержек при передаче сообщений через шлюз.

Заключение

В статье поднимается проблема разработки интеграционной архитектуры для отечественной экосистемы оказания медицинской помощи. Проблема сложная и многогранная. На данном этапе развития экосистемы целесообразно вести обмен идеями по созданию интеграционной архитектуры и проводить прототипирование отдельных архитектурных идей. В работе предложен апробированный принцип построения асинхронного взаимодействия между участниками экосистемы. Предложенное решение может быть использовано другими разработчиками, решающими интеграционные задачи в цифровых экосистемах любой природы, особенно в случае информационного обмена с задержками. Полученные результаты могут найти применение и в более сложных архитектурах будущего [12]. Работа может также повлиять на развитие отечественной цифровой медицинской экосистемы. Актуально привлечение к решению проблемы крупных отечественных разработчиков ПО.

Список использованных источников

[1] Abernethy A., Adams L., Barrett M., Bechtel C., Brennan P., Butte A., Faulkner J., Fontaine E., Friedhoff S., Halamka J., Howell M., Johnson K., Long P., McGraw D., Miller R., Lee P., Perlin J., Rucker D., Sandy L., Savage L., Stump L., Tang P., Topol E., Tuckson R., Valdes K. *The promise of digital health: then, now, and the future* // NAM Perspect.– Jun 27 2022.– id. 202206e. doi ↑476, 477, 485, 486

[2] Viswanadham N. *Ecosystem model for healthcare platform* // Sādhanā.– 2021.– Vol. **46**.– id. 188.– 13 pp. ↑477

[3] Chibuike M. C., Grobbelaar S. S., Botha A. *Overcoming challenges for improved patient-centric care: a scoping review of platform ecosystems in healthcare* // IEEE Access.– 2024.– Vol. **12**.– Pp. 14298–14313. ↑477

[4] Белышев Д. В., Гулиев Я. И., Михеев А. Е. *Цифровая экосистема медицинской помощи* // Врач и информационные технологии.– 2018.– № 5.– С. 4–17. ↑478

[5] Михеев А. Е. *МИС как бизнес-платформа цифровой экосистемы медицинской помощи* // Менеджер здравоохранения.– 2022.– № S1.– С. 5–22. ↑478

[6] Малых В. Л., Калинин А. Н. *Миграция с Oracle на PostgreSQL сильно связной МИС Интерин* // Программные системы: теория и приложения.– 2022.– Т. **13**.– № 4(55).– С. 77–91. ↑480

[7] Фадеев В. *Пять мифов корпоративной автоматизации* // Открытые системы. СУБД.– 2023.– № 02.– С. 41–43. ↑480

[8] Добровольский А. *Интеграция приложений: методы взаимодействия, топология, инструменты* // Открытые системы. СУБД.– 2006.– № 09.– С. 30–34. ↑480

[9] Abugabah A., Nizamuddin N., Alzubi A. A. *Decentralized telemedicine framework for a smart healthcare ecosystem* // IEEE Access.– 2020.– Vol. **8**.– Pp. 166575–166588. ↑482

[10] Казаков И. Ф., Гулиев Я. И., Бельченков А. А., Рудецкий С. В. *Развитие пациент-ориентированных ИТ-сервисов в медицинских организациях* // Менеджер здравоохранения.– 2022.– № S1.– С. 63–68. ↑485

[11] Гулиев Я. И., Белышев Д. В., Кочуров Е. В. *Медицинская информационная система "Интерин Promis Alpha-— новые горизонты* // Врач и информационные технологии.– 2016.– № 6.– С. 6–15. ↑488

[12] Прозоров А., Шнырев Р., Волков Д. *Архитектура цифровых платформ будущего* // Открытые системы. СУБД.– 2021.– № 02.– С. 24–28. ↑488

Поступила в редакцию 22.02.2024;
одобрена после рецензирования 08.04.2024;
принята к публикации 16.05.2024;
опубликована онлайн 29.06.2024.

Рекомендовал к публикации *к.т.н. Я. И. Гулиев*

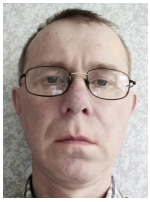
Информация об авторах:

Владимир Леонидович Малых

Зав. лабораторией Института программных систем им. А. К. Айламазяна, 30 лет работы в сфере медицинской информатики, архитектор семейства МИС Интерин. Научные интересы: Медицинские информационные системы, платформы для МИС, интеграционные профили и личные кабинеты участников медицинских экосистем.

0000-0002-0072-0724

e-mail: *mvl@interin.ru*



Алексей Николаевич Калинин

Ведущий инженер-программист ООО «Интерин технологии», ведущий разработчик семейства МИС Интерин. Медицинские информационные системы, компоненты разработки МИС на платформе Интерин Альфа, интеграционные профили участников медицинских экосистем.

0000-0002-3607-7400

e-mail: *ank@interin.ru*

Сергей Владимирович Рудецкий

Руководитель группы разработки ООО «Интерин технологии». Научные интересы: Медицинские информационные системы, платформы для МИС, интеграционные профили и личные кабинеты участников медицинских экосистем.

0000-0003-2986-3785

e-mail: *rsv@interin.ru*

Авторы внесли равный вклад в подготовку публикации.

Декларация об отсутствии личной заинтересованности: *благополучие авторов не зависит от результатов исследования.*

EN RU PROGRAM SYSTEMS: THEORY AND APPLICATIONS ISSN 2079-3316
Research Article INFORMATION TECHNOLOGY IN MEDICINE

UDC 61:007
10.25209/2079-3316-2024-15-2-475-492

Architecture of interaction in the digital medical ecosystem

Vladimir Leonidovich **Malykh**[1], Aleksey Nikolayevich **Kalinin**[2], Aleksey Nikolayevich **Rudetsky**[3]

Abstract. In the field of medical informatics, there is a steady trend towards the formation of a complex multicomponent ecosystem. The problems of interaction and integration of ecosystem components come to the forefront: medical, laboratory, radiological systems, EGISZ, EMIAS, MDLP, various registers and services, including those implementing AI approaches to data processing and problem solving. Patients are in dire need of personal offices that integrate their medical data, patients become active participants in the ecosystem. Integration tasks have to be solved in a highly heterogeneous information environment, when it becomes unattainable to ensure synchronous interactive interaction between ecosystem participants. For individual applications, a flexible combination of both synchronous and asynchronous interaction is required, selected situationally based on specific time delays and interaction characteristics.

The article proposes a special architecture that allows for synchronous and asynchronous interaction between ecosystem participants. Adapting software designed only for synchronous interaction to an asynchronous architecture does not require a radical redesign of the software. The approach was worked out using the example of adapting the MDLP MIS Interin module to work in the internal secure network of the multidisciplinary medical center of the Bank of Russia. The proposed architecture can be used by software developers in other fields of activity, where there is an active development of ecosystems, accompanied by an increase in integration interactions. (*In Russian*).

Key words and phrases: medical informatics, medical information systems, digital ecosystem, integration, asynchronous interaction, patient's personal office

2020 *Mathematics Subject Classification*: 94-04; 68P05; 92C50

Acknowledgments:

For citation: Vladimir L. Malykh, Aleksey N. Kalinin, Aleksey N. Rudetsky. *Architecture of interaction in the digital medical ecosystem*. Program Systems: Theory and Applications, 2024, **15**:2(61), pp. 475–492. (*In Russ.*). https://psta.psiras.ru/read/psta2024_2_475-492.pdf

References

[1] A. Abernethy, L. Adams, M. Barrett, C. Bechtel, P. Brennan, A. Butte, J. Faulkner, E. Fontaine, S. Friedhoff, J. Halamka, M. Howell, K. Johnson, P. Long, D. McGraw, R. Miller, P. Lee, J. Perlin, D. Rucker, L. Sandy, L. Savage, L. Stump, P. Tang, E. Topol, R. Tuckson, K. Valdes. "The promise of digital health: then, now, and the future", *NAM Perspect.*, 2022, id. 202206e.

[2] N. Viswanadham. "Ecosystem model for healthcare platform", *Sādhanā*, **46** (2021), id. 188, 13 pp.

[3] Chibuike M. C. , S. S. Grobbelaar, A. Botha. "Overcoming challenges for improved patient-centric care: a scoping review of platform ecosystems in healthcare", *IEEE Access*, **12** (2024), pp. 14298–14313.

[4] D. V. Belyshev, Ya. I. Guliev, A. E. Mixeev. "Digital healthcare ecosystem", *Vrach i informacionnye texnologii*, 2018, no. 5, pp. 4–17 (in Russian).

[5] A. E. Mixeev. "HIS as a buisiness platform of the digital ecosystem of medical care", *Menedzher zdravooxraneniya*, 2022, no. S1, pp. 5–22 (in Russian).

[6] V. L. Malyx, A. N. Kalinin. "Migration from Oracle to PostgreSQL", *Program Systems: Theory and Applications*, **13**:4(55) (2022), pp. 77–91 (in Russian).

[7] V. Fadeev. "Five myths of corporate automation", *Otkrytye sistemy. SUBD*, 2023, no. 02, pp. 41–43 (in Russian).

[8] A. Dobrovol'skij. "Application integration: interaction methods, topology, tools", *Otkrytye sistemy. SUBD*, 2006, no. 09, pp. 30–34 (in Russian).

[9] A. Abugabah, N. Nizamuddin, A. A. Alzubi. "Decentralized telemedicine framework for a smart healthcare ecosystem", *IEEE Access*, **8** (2020), pp. 166575–166588.

[10] I. F. Kazakov, Ya. I. Guliev, A. A. Bel'chenkov, S. V. Rudeckij. "Development of patient-centered services in medical organizations", *Menedzher zdravooxraneniya*, 2022, no. S1, pp. 63–68 (in Russian).

[11] Ya. I. Guliev, D. V. Belyshev, E. V. Kochurov. "Medical information system Interin Promis Alpha — new horizons", *Vrach i informacionnye texnologii*, 2016, no. 6, pp. 6–15 (in Russian).

[12] A. Prozorov, R. Shnyrev, D. Volkov. "Architecture of digital platforms of the future", *Otkrytye sistemy. SUBD*, 2021, no. 02, pp. 24–28 (in Russian).

Том **15** Выпуск 2(61) 2024 г.

Авторский указатель

Vol. 15 Issue 2(61) 2024

Author index

Vol. 15 Issue 2(61) 2024

Contents

Чтобы сменить язык страницы, кликните, пожалуйста, флаг в верхнем углу