ДОКЛАДЫ РОССИЙСКОЙ АКАДЕМИИ НАУК. МАТЕМАТИКА, ИНФОРМАТИКА, ПРОЦЕССЫ УПРАВЛЕНИЯ, 2024, том 516, с. 103–112

ИНФОРМАТИКА

УДК 004.74

ПРИМЕНЕНИЕ МЕТОДОВ С МАШИННЫМ ОБУЧЕНИЕМ ДЛЯ УПРАВЛЕНИЯ СЕТЕВОЙ ВЫЧИСЛИТЕЛЬНОЙ ИНФРАСТРУКТУРОЙ

© 2024 г. Член-корреспондент РАН Р. Л. Смелянский[1, *], Е. П. Степанов[1, **]

Поступило 14.11.2023 г.
После доработки 20.03.2024 г.
Принято к публикации 26.03.2024 г.

В статье рассмотрено применение методов машинного обучения для оптимального управления ресурсами сетевой вычислительной инфраструктурой – вычислительной инфраструктурой нового поколения. Рассмотрена связь между предлагаемой вычислительной инфраструктурой и концепцией GRID. Показано, как методы машинного обучения в управлении сетевой вычислительной инфраструктуре позволяют решить проблемы управления вычислительной инфраструктурой, которые не позволили реализовать концепцию GRID в полной мере. В качестве примера рассмотрено применение метода многоагентной оптимизации в комбинации с методом машинного обучения с подкреплением для управления сетевыми ресурсами. Показано, что использование многоагентных методов машинного обучения позволяет повысить скорость распределения транспортных потоков и обеспечить оптимальную загрузку сетевых каналов вычислительной инфраструктуры по критерию равномерности распределения нагрузки и что такое управление сетевыми ресурсами эффективнее централизованного подхода.

Ключевые слова: методы обучения с подкреплением, многоагентные методы, сетевая вычислительная инфраструктура

DOI: 10.31857/S2686954324020176, **EDN:** XHPDZH

1. ВВЕДЕНИЕ

Организация вычислений – одна из основ современной цивилизации. Ее эффективность зависит от устройства вычислительной инфраструктуры. Поэтому важно понимать основные тенденции и направление ее развития, основной движущей силой которой являются требования приложений пользователей. Под термином *Вычислительная инфраструктура* далее будем понимать комплекс локальных вычислительных установок (центров обработки данных, высокопроизводительных вычислителей, серверных кластеров), связанных сетью передачи данных, у которого есть самостоятельные распределенные контуры динамического управления/распределения вычислительной нагрузкой и динамического управления потоками данных между компонентами комплекса. Вычислительная инфраструктура должна обладать всеми свойствами открытых систем и позволять неограниченно подключать к существующему комплексу вычислительных установок новые локации без остановки или нарушения функционирования существующего комплекса. Она должна отвечать всем требованиям безопасной обработки, передачи и хранения данных. Обеспечивать бесперебойный обмен данными как внутри существующего комплекса, так и с подключаемыми внешними компонентами в соответствии с требованиями приложений. Взаимодействующие компоненты такой вычислительной инфраструктуры должны позволять их модернизацию без остановки ее функционирования.

В [1] представлен анализ широкого спектра приложений из различных прикладных областей таких, как умный город, умный дом, здравоохранение (особенно такие направления как телемедицина, хирургия), транспорт, сельское хозяйство, научные междисциплинарные исследования. Отмечено, что все современные приложения чувствительны к времени отклика. Расстояние между точками обработки, хранения данных и получения результатов вычислений влияет на задержку транспортировки данных,

[1] *Московский государственный университет им. М.В. Ломоносова, факультет Вычислительной математики и кибернетики, кафедра Автоматизации систем вычислительных комплексов, Москва, Россия*

[*] *E-mail: smel@cs.msu.su*

[**] *E-mail: estepanov@lvk.cs.msu.ru*

и неправильный выбор локальной вычислительной установки может нарушить требования приложения и даже блокировать его работу. Это означает, что допустимое расстояние между местом обработки данных (приложением или его компонентами) и их источником ограничено. Еще один важный вывод, который был сделан в результате этого анализа, состоял в том, что привязка приложений или их компонентов к определенным географических местам неэффективна. Неэффективность проявляется, во-первых, в том, что ограничиваются мобильность как источника данных, так и потребителя результата вычислений. Во-вторых, время нахождения запроса на вычисления в вычислительной инфраструктуре (далее время заявки в системе), т.е. сумма времен на передачу данных к месту обработки, нахождение заявки на обработку данных в очереди, обработка данных (вычисления), передачу промежуточных результатов между компонентами приложения, которые могут выполняться в разных местах, и время на доставку результата, может оказаться больше, чем время нахождения этой же заявки в системе, если бы вычислительная инфраструктура могла динамически в ходе работы сама выбирать, где будет происходить обработка тех или иных данных. Этот вывод подтверждают результаты исследования [2]. Из практики хорошо известно, что в очереди к суперкомпьютеру задача может простоять несколько дней. В то же время эта же задача могла бы быть выполнена в облачной среде близлежащего центра обработки данных. Время выполнения в облачной среде наверняка будет больше, чем время выполнения в среде суперкомпьютера, но по критерию время нахождения заявки в очереди облачная среда может оказаться эффективнее. Сказанное позволяет говорить о том, что управлять распределением приложений в вычислительной инфраструктуре, потоками данных должна вычислительная инфраструктура, а не пользователь.

Как уже было отмечено, основной движущей силой развития инфраструктуры для вычислений, ее операционной среды, языков программирования и инструментов всегда были потребности приложений. Именно потребности приложений, а не возможности аппаратных средств, инженерии программирования, базового программного обеспечения. Набор свойств современных приложений [3] включает в себя: распределенность, самодостаточность, режим реального времени, эластичность, кроссплатформенность и другие. Описания этих терминов можно найти в [3]. Для облегчения понимания дальнейшего поясним термин самодостаточность, который определен в [3] как: “приложение уже не представляется только кодом и исходными данными, оно сопровождается описанием структуры приложения, взаимосвязи его компонентов (далее сервисов приложений), заданием требуемого уровня их производительности/времени реакции, явно сформулированными требованиями к вычислительным и сетевым ресурсам, ресурсам хранения данных и доступу к ним, предполагаемых временных ограничений на время вычислений и передачу данных в виде соглашения об уровне обслуживания (Service Level Agreement – далее SLA), процедуры запуска приложений. Для таких описаний сейчас активно развиваются специальные языки. Ярким примером прообраза такого языка может служить язык TOSCA [4] (далее такое описание будет называться Спецификацией Функционирования Приложения (Application Operation Specification – AOS))”.

Приложение со свойствами, сформулированном выше, предъявляет следующие требования к вычислительной инфраструктуре [3]: детерминированность поведения, безопасность, доступность, надежность и отказоустойчивость, эффективность и справедливость при распределении ресурсов, повсеместная виртуализация всех видов ресурсов, масштабируемость, бессерверность. Детерминированность поведения – это возможность заранее прогнозировать количественные параметры функционирования такие, как время исполнения, задержка на передачу данных. Бессерверная технология в [3] трактовалась, следуя [5], следующим образом: “инфраструктура должна автоматически размещать компоненты приложения таким образом, чтобы они могли взаимодействовать в соответствии с требованиями SLA приложения и так, чтобы гарантировать экономное использование ресурсов инфраструктуры”.

Чтобы вычислительная инфраструктура могла соответствовать требованиям эффективности и детерминированности и служить вычислительной инфраструктурой для приложений в вышесказанном смысле, ее функционирование должно соответствовать таким требованиям, как: “

• предсказуемость времени выполнения компонентов приложения и времени их взаимодействия (передачи данных) согласно AOS;

• наличие методов оценки времени выполнения компонентов приложения, инвариантных

к архитектуре вычислителей, входящих в состав сетевой вычислительной инфраструктуры;

• детерминированность характеристик передачи данных между компонентами приложения по оверлейным каналам (далее Quality of Service – QoS канала);

• надежная изоляция контура управления и контура передачи данных (далее просто контура данных) в сети передачи данных (СПД) вычислительной инфраструктуры от ошибок в сетевом оборудовании, а также изоляция потоков данных от вредоносных воздействий на этих контурах". [3]

Для обеспечения детерминированности параметров передачи данных между компонентами приложения необходимо: "

• устанавливать и гарантировать диапазоны изменения сквозной задержки и джиттера в СПД;

• гарантировать вероятность потери пакетов в СПД на уровне, соответствующем SLA приложения;

• сделать оперативно управляемым использование доступной пропускной способности каналов СПД;

• исключить непредсказуемые задержки при передаче данных, вызванные задержкой пакетов из-за отказа в соединении, повторной передачи, обратной связи по перегрузке, использования широковещательной рассылки и т.д." [3]

Вычислительную инфраструктуру, удовлетворяющую перечисленным выше требованиям, мы будем называть *сетевой вычислительной инфраструктурой* (Network Powered by Computing, далее NPC).

Рассматривая ретроспективу развития вычислительной инфраструктуры следует признать, что концепция GRID вычислительной инфраструктуры, предложенной Яном Фостером и Карлом Кессельманом в 1999 году [6], наиболее соответствует сформулированным выше требованиям. Вот как сами авторы определяют основные свойства GRID инфраструктуры (цитируется по [7]):

1. "Координирует ресурсы, которые не подлежат централизованному контролю... (Grid объединяет и координирует ресурсы и пользователей, которые живут в разных доменах управления – например, рабочий стол пользователя или центральный компьютер; разные административные подразделения одной и той же компании; или разных компаний; и решает вопросы безопасности, политики, оплаты, членства и т.д., которые возникают в этих условиях. В противном случае мы имеем дело с локальной системой управления.)

2. Использует стандартные, открытые протоколы и интерфейсы общего назначения... (GRID построен из многоцелевых протоколов и интерфейсов, которые решают такие фундаментальные проблемы, как аутентификация, авторизация, обнаружение ресурсов и доступ к ресурсам. ... [Это] важно, чтобы эти протоколы и интерфейсы были стандартными и открытыми. В противном случае мы имеем дело с системой, специфичной для приложения.)

3. Предоставляет нетривиальные качества обслуживания. (GRID позволяет скоординированно использовать составляющие его ресурсы для предоставления услуг различного качества, касающихся, например, времени отклика, пропускной способности, доступности и безопасности и/или совместного размещения нескольких типов ресурсов для удовлетворения сложных требований пользователей, поэтому полезность объединенной системы значительно больше, чем полезность суммы ее частей".

Однако, по мнению авторов статьи [7], ни одна из многочисленных попыток реализовать концепцию GRID до сих пор не увенчалась успехом. Ни одна из перечисленных в этой работе инфраструктур не удовлетворила сформулированные выше требования, не позволила автоматически минимизировать время выполнения приложения в вычислительной инфраструктуре с учетом его временных ограничений за счет эффективного автоматического управления распределением вычислительной нагрузки между доступными вычислителями, "нетривиального качества обслуживания", эффективного управления передачей данных между компонентами приложения, возможности неограниченного наращивания вычислительных ресурсов по требованию [7].

Связано это с тем, что производительность вычислительной техники, доступная скорость передачи данных и доступные математические методы управления ресурсами вычислительной инфраструктуры в начале 2000-х были недостаточны для достижения этих свойств. Однако, согласно [8], производительность серверов с 2014 по 2024 возросла примерно в 7 раз, производительность супервычислителей выросла на 4 порядка [9], а максимальная скорость передачи данных за этот же период возросла с нескольких Tbps до 1.1 Pbps [10].

Эти достижения стимулировали прорыв в математических методах оптимизации на основе машинного обучения. Дело в том, что, принимая во внимание отмеченные выше скорости вычислений и передачи данных и то, что сетевые сервисы и компоненты приложений легко масштабируются и работают в режиме реального времени, в NPC для оптимизации управления ресурсами, потоками данных и вычислениями требуются алгоритмы с низкой временной сложностью, т.к. временные задержки на принятие решений становятся критическими для эффективного функционирования инфраструктуры. Классические методы оптимизации [11] для этого не подходят, т.к. предполагают централизованное принятие решения, когда проводится централизованно комбинаторный поиск вариантов решения по чётко определенному (детерминированному) алгоритму, позволяющему найти наилучшее решение проблемы. Такой подход кроме вычислительной сложности требует больших накладных расходов на сбор, обработку и передачу данных между компонентами вычислительной инфраструктуры. В этих условиях предпочтительнее становятся методы распределенной, мультиагентной оптимизации (МАО). В мультиагентных методах решение задачи получается в ходе самоорганизации распределенного множества алгоритмов-агентов, способных к конкуренции и/или к кооперации, и имеющих собственные критерии, предпочтения и ограничения. Решение считается найденным, когда в ходе своих недетерминированных взаимодействий агенты достигают неулучшаемого консенсуса (временного равновесия или баланса интересов), который и принимается за решение задачи.

Развитие методов МАО существенно стимулировали достижения за указанный период времени в области математических методов распределенной оптимизации, позволяющие сократить число обменов и объем передаваемых данных при распределенной оптимизации в отдельных случаях до 240 раз, а в среднем – до 4–5 раз [12, 13].

Идея тензорного поезда [14] в случае d-мерных массивов, обладающих специальной структурой, позволила снизить требования к объему памяти для их хранения с экспоненциальных, т.е. $O(N^d)$ ячеек памяти, до почти линейных, т.е. $O(d * N * r^2)$, где r – максимальный из рангов тензорного поезда. При этом вычислительная сложность построения функционально-порожденных тензоров тензорного поезда всего лишь за $O(d * N * r^3)$ вычислений их элементов. Идеи тензорных разложений нашли ряд применений в задачах со сжатием нейросетевых моделей, где позволили сократить требования к памяти в десятки раз без существенных потерь в качестве их работы [15, 16]. В настоящее время за рубежом появляется интерес к пробным реализациям данных алгоритмов на реальных чипах [17], призванных решать задачи искусственного интеллекта.

Методы машинного обучения, идеи тензорных разложений нашли применение в задачах, связанных с методами оценки ожидаемого времени выполнения программ, инвариантных к особенностям архитектуры вычислителя [3]. Задача предсказания состоит в том, чтобы спрогнозировать ожидаемое время работы программы на вычислителе, на котором эта программа с определенным набором параметров до этого не выполнялась. Эти методы можно использовать для задач оптимального распределения компонентов приложения между вычислительными узлами, оптимального распределения потоков данных между наложенными каналами.

Ниже мы кратко рассмотрим основные положения архитектуры новой вычислительной инфраструктуры NPC и методы управления ею. В этой статье в качестве примера применения математических методов распределённого машинного обучения мы рассмотрим задачу управления потоками передаваемых данных (далее сетевым трафиком) в NPC, суть которой состоит в распределении трафика по каналам сети передачи данных так, чтобы не вызывать перегрузки и минимизировать задержки при передаче. Главной причиной выбора этой задачи в качестве примера является то, что её решение есть комбинация метода распределенной мультиагентной оптимизации и методов машинного обучения с подкреплением. Такая комбинация позволяет принимать решения по управлению быстрее по сравнению с централизованным подходом за счет распределенности агентов и подстраиваться в динамике под состояние сетевой вычислительной инфраструктуры за счет обучения с подкреплением.

2. СЕТЕВАЯ ВЫЧИСЛИТЕЛЬНАЯ ИНФРАСТРУКТУРА

Функциональная архитектура NPC (рис. 4) была представлена в [3]. Основу построения NPC составляют технологии программно-кон-

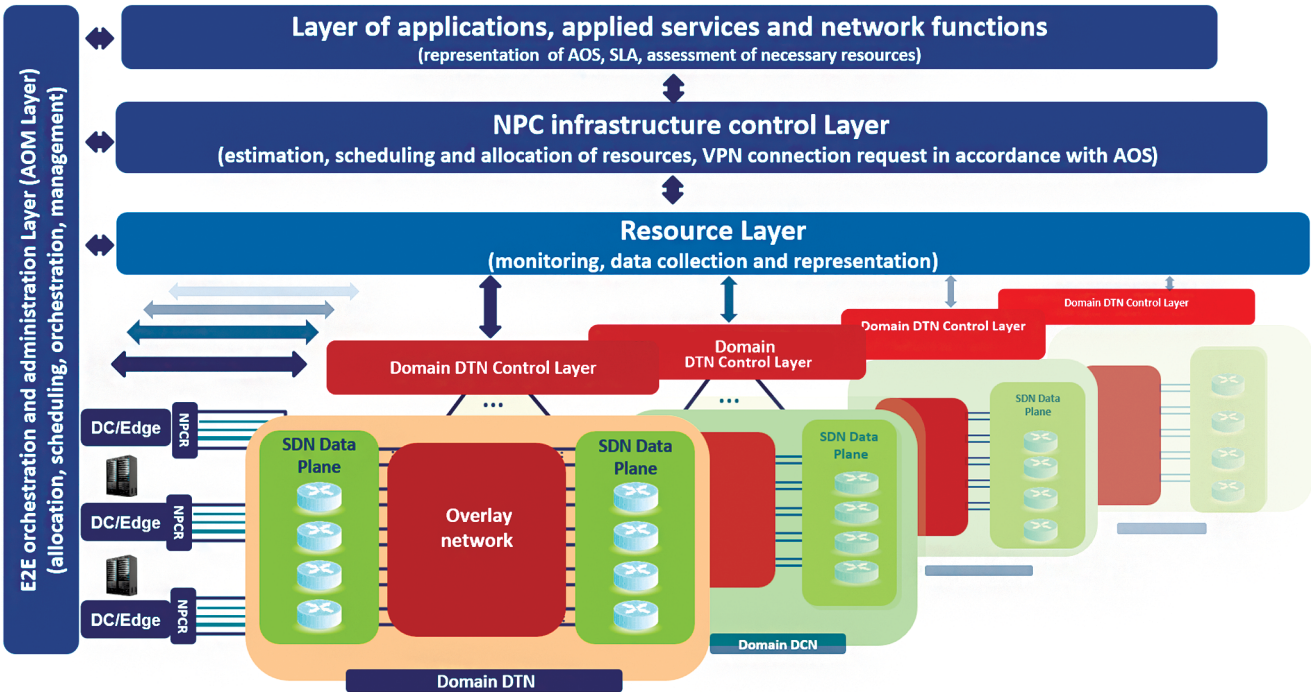

Рис. 1. Функциональная архитектура NPC.

фигурируемых сетей (SDN) и виртуализации сетевых функций (NFV) [18]. Кратко опишем ее организацию, основанную на федеративном принципе [6]. В ведении каждого федерата находится определенное количество вычислительных ресурсов, телекоммуникационных ресурсов и ресурсов хранения данных. Управление перечисленными ресурсами в NPC организовано в виде нескольких контуров: контур приложения, сервисов приложений и сетевых функций (ASNF), контур управления инфраструктурой NPC (NPCIC), контур вычислительных, сетевых ресурсов и ресурсов хранения данных (NPCRL), контур оркестрации, администрирования и управления (OAM).

Контур ASNF является распределенным и отвечает за представление приложения: его код и его данные, спецификацию работы приложения (AOS), представление сервисов приложения и виртуализированных сетевых функций VNF, необходимых для работы приложения, спецификацию сети передачи данных между компонентами приложения. Спецификация сети должна включать топологию сети передачи данных между компонентами приложения, требования к качеству обслуживания каналов (QoS), которые должны включать такие характеристики, как доступная пропускная способность, допустимая задержка, допустимая вероятность потери пакетов, допустимый диапазон изменения джиттера. На основе этой информации и SLA для приложения в целом контур ASNF рассчитывает SLA для каждого сервиса приложения, входящего в его AOS спецификацию.

Контур NPCIC также является распределенным. Его компоненты, координируя свои действия, обеспечивают планирование и назначение компонентам приложения ресурсов NPC в соответствии с AOS спецификацией и прогнозируемым временем для вычислений и передачи данных, соблюдая требования приложения SLA на основе текущего состояния ресурсов. Для обеспечения взаимодействия компонентов приложения контур NPCIC создает и поддерживает наложенную сеть между надлежащими вычислительными локациями в NPC, где были размещены компоненты приложения в соответствии с AOS спецификацией. Именно в этом контуре определяется, какие VNF потребуются и где в сети передачи данных.

Контур NPCRL отвечает за унифицированное представление состояния разнородных ресурсов NPC и мониторинг их текущего состояния, и прогнозирование их состояния на ближайшее время. Прогнозирование является краеугольным камнем, позволяющим сделать поведение NPC предсказуемым.

Контур OAM организует согласованное взаимодействие компонентов приложения в соответствии с AOS, собирает данные о потреблении ресурсов для каждого компонента приложения, обеспечивает управление безопасностью и администрирование NPC.

В представленной архитектуре есть важный для дальнейшего изложения компонент – устройство, ответственное за объединение ресурсов обработки данных, сети передачи данных и внешних источников запросов на вычислительные услуги. Назовем такое устройство NPC маршрутизатором (NPCR, изображен на рис. 4 справа от DC/Edge). NPCR доступно несколько наложенных каналов – каналов, состоящих из физических линий передачи данных, возможно, разной природы: каоксиал, витая пара, оптическая линия, радиоканал. Из-за разной природы наложенные каналы предоставляют разное качество сервиса, поэтому лишь некоторые из них могут быть использованы для передачи данных до определенной точки назначения с соблюдением требований SLA.

NPCR выполняет функции сразу несколько устройств – диспетчер потока задач в среде NPC, маршрутизатор трафика и задач в среде NPC, VPN шлюз для вычислительных мощностей, находящихся за ним (на рис. 4 они обозначены как DC/Edge), CPE устройство для вычислительных мощностей, находящихся за ним, и предоставляет следующую функциональность:

1. Распределение сервисов приложения (AS) / виртуальных сетевых функций (VNF) между вычислительными узлами (ВУ) контура NPCRL;

2. Принятие решения: стоит выполнять определенный AS/VNF на ВУ, подсоединенном к этому NPCR или нет;

3. Перенаправление AS / VNF, которые не были приняты для выполнения на текущем ВУ, на другие NPCR, чьи вычислительные ресурсы будут предпочтительнее с точки зрения эффективности выполнения приложения как единого целого или сбалансированности загрузки ресурсов NPC среды;

4. Оптимальная маршрутизация трафика как между сервисами приложения, так и между VNF, используемых конкретным приложением;

5. Обеспечение требуемого SLA для транспортного соединения.

Некоторые из NPCR являются точками доступа к ресурсам среды NPC для внешних источников данных и приложений. Далее мы рассмотрим алгоритмы функционирования NPCR, обеспечивающие эффективное распределение потоков данных между доступными каналам и передачи данных в соответствии с требованиями AOS спецификации.

3. УПРАВЛЕНИЕ СЕТЕВЫМ ТРАФИКОМ ПРИ ПОМОЩИ МЕТОДОВ ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Последние статьи [19, 20, 21] по методам балансировки нагрузки показали, что основное внимание сейчас уделяется методам машинного обучения, особенно многоагентному обучению с подкреплением (MARL). Как уже было отмечено, масштабы и скорость передачи данных в современных сети не позволяют получить оптимальное по скорости и точности балансировки трафика решение классическими математическими методами. Вот почему исследования продолжают фокусироваться на методах машинного обучения. Существует три подхода к мультиагентной оптимизации: централизованный, децентрализованный с взаимодействием между агентами и полностью децентрализованный. Во всех этих подходах предполагается, что агенты формируют свое локальное состояние на основе наблюдений за окружающей средой. Однако поведение агентов в этих подходах различно:

• централизованный подход предполагает наличие в среде одного объекта (центр управления), который координирует работу каждого агента. Этот центр управления собирает данные о локальных состояниях от всех агентов и определяет действия для каждого из них на основе решения задачи оптимизации;

• децентрализованный подход с взаимодействием между агентами (или просто децентрализованный подход) предполагает, что каждый агент обменивается с другими агентами своим локальным состоянием. На основе собранной информации агент выбирает оптимальное действие;

• полностью децентрализованный подход не предполагает какого-либо общения между агентами. Каждый агент принимает решение только на основе анализа истории своего поведения.

Авторы настоящей статьи в работе [22] предложили новый метод Multi-Agent Routing using Hashing (MAROH). Метод MAROH основан на объединении идей децентрализованного мультиагентного обучения с подкреплением и

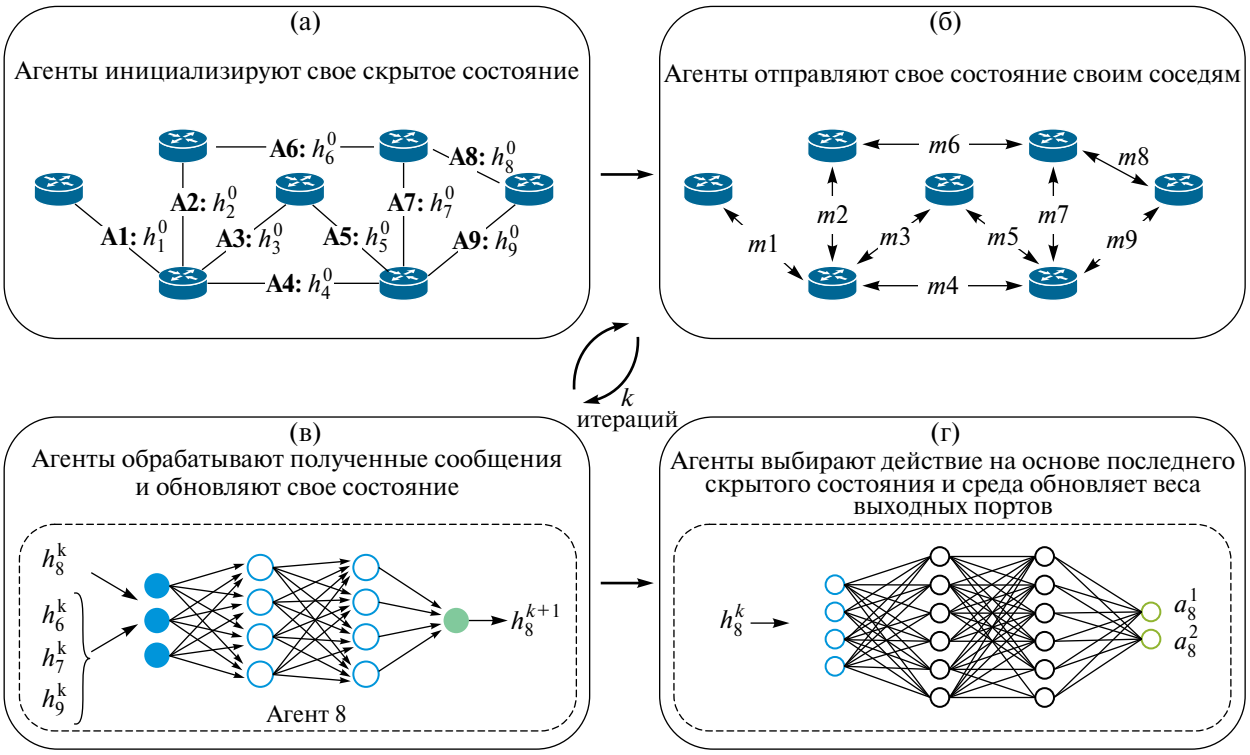

Рис. 2. Схема работы MAROH.

алгоритмов консистентного хеширования. Эта комбинация позволила обеспечить справедливое распределение трафика между сетевыми ресурсами и эффективное их использование. Предложенный метод в каждый момент времени каждому выходному порту NPCR присваивает определенный вес, который потом используется консистентной хэш-функцией для распределения потоков данных между наложенными каналами. Таким образом, ключевыми проблемами являются то, как присвоить веса выходным портам NPCR и как построить надлежащую хеш-функцию.

Чтобы избежать появление циклов на маршрутах, для выбора порта до надлежащего места назначения в методе MAROH был разработан специальный алгоритм построения маршрутов, называемый выбором следующего перехода (NHS) [22]. Согласно этому алгоритму, строится два ациклических направленных остовных подграфа топологии NPC и лишь один раз можно перейти между этими подграфами при построении маршрута, что гарантирует отсутствие циклов.

Метод MAROH состоит из трех частей:

1. Вычисление набора выходных портов алгоритмом NHS.

2. Расчет весов для вычисленных портов алгоритмом MARL.

3. Распределение потоков данных между каналами, инцидентными вычисленным портам в соответствии с рассчитанными весами.

Напомним, что MAROH – это алгоритм управления трафиком в NPC агентах, расположенными по одному на каждом NPCR.

Общая схема работы агента в предложенном методе представлена на рис. 4. В MAROH используется метод, называемый нейронной сетью передачи сообщений (Message Passing Neural Network – MPNN). MPNN предназначен для построения оптимальной передачи сообщений между агентами в сети, представленной неориентированным графом. Для TCP/IP сети эти сообщения будут являться сообщениями протокола внутридоменной маршрутизации IGP (например, OSPF) с обновлениями состояний сети, а для MAROH агента – сообщения о состоянии агента. Заметим, что для контура NPCRL могут быть адаптированы существующие протоколы IGP.

Метод MPNN основан на итеративном алгоритме передачи сообщений между выбранными элементами графа – вершинами, где располагаются агенты. Передаваемые сообщения обеспечивают обновление состояний агентов, которое в методе MPNN называется скрытым состоянием (hidden state). В нашем случае скрытое состо-

яние представлено тройкой – занятой полосой пропускания канала $b_{i,j}$, пропускной способностью канала $c_{i,j}$ и текущими весами $r_{i,j}$. Скрытое состояние может быть инициализировано случайно выбранными значениями либо рассчитанными заранее на основе уже собранной статистики (рис. 4а). Затем агенты отправляют свое скрытое состояние всем своим соседям в графе наложенной сети (рис. 4б). На основе полученных сообщений от соседей и своего собственного скрытого состояния каждый NPCR обновляет свое скрытое состояние при помощи нейронной сети (рис. 4в). Процесс отправки сообщений соседям и обновления скрытого состояния повторяется заранее заданное количество раз. Это обеспечивает распространение состояния агента в заранее определённой его окрестности. Размер этой окрестности определяет количество итераций отправки сообщений соседям и обновления скрытого состояния. Экспериментальное исследование показало, что количество обменов не должно превышать диаметр графа, иначе скрытые состояния у всех агентов становятся почти неразличимыми между собой и агенты будут применять произвольные действия вместо тех, которые подходят текущему состоянию сети. После завершения процесса обновления состояния специальная нейронная сеть на агенте NPCR рассчитывает оценку ценности для каждого возможного действия агента (рис. 4г). Под действиями агента будем понимать операции по изменению веса канала: сложение с константой, умножение на константу или отсутствие изменения веса. Агент NCPR выбирает действие согласно выданной оценке. В результате изменяется вес наложенного канала, а после работы хеш-функции и распределение потоков данных в сети.

Под оптимальным распределением трафика в сети будем понимать равномерную загрузку всех наложенных каналов. Для достижения этой цели будем минимизировать функцию $\Phi = \frac{1}{|E|}\Sigma_{(u,v)\in E}\left(\frac{b_{u,v}}{c_{u,v}} - \mu'\right)^2$, где $|E|$ – число наложенных каналов, а μ' – средняя загрузка каналов, равная $\frac{1}{|E|}\Sigma_{(u,v)\in E}\frac{b_{u,v}}{c_{u,v}}$. В терминах задачи оптимизации MARL Φ является целевой функцией.

Был проведен экспериментально сравнительный анализ эффективности метода MAROH с другими распространенными методами балансировки – ECMP [23] и UCMP [24]. Методика проведения сравнительного анализа основывается на моделировании многоагентного управления потоками данных в NPC методом Монте-Карло. Для моделирования ECMP и UCMP использовался тот же подход с расчетом весов наложенного канала, как и в MAROH, только для ECMP использовались наложенные каналы равной протяженности и вес у всех каналов брался равным 1, а для UCMP вес наложенного канала принимался равным текущей загруженности канала. Здесь стоит подчеркнуть важное отличие MAROH от методов ECMP и UCMP – MAROH принимает решение на основе общения между агентами, в то время как ECMP и UCMP только на основе локальной информации. После каждого обновления весов в сеть пускалось 40 потоков с новыми идентификаторами, распределение которых меняло загрузку наложенных каналов. Так как в настоящий момент отсутствуют статистические данные по потокам в сетевой вычислительной инфраструктуре, каждый поток имел случайно выбранные узел источник и узел назначения. Запускалось 100 итераций обновлений весов для каждого из методов MAROH, ECMP, UCMP с одинаковыми входными данными (множество потоков на каждой итерации) на топологии с множеством вариантов альтернативных маршрутов, изображенной на рис. 4. В качестве критерия сравнения использовалась целевая функция Φ, отражающая равномерность загрузки наложенных каналов.

Также было проведено сравнение с полностью централизованным подходом по управлению агентами NPCR, в котором может достигаться оптимальное распределение трафика, так как присутствует информация о состоянии всей сети. Использовался генетический алгоритм для расчета весов наложенных каналов на каждой итерации. Результаты сравнительного анализа значения целевой функции Φ приведены на рис. 4 в виде коробчатой диаграммой для каждого метода. MAROH обозначен как Multi-agent Traffic Engineering (MA-TE). На рис. 4 хорошо видно, что MAROH показывает значительно лучше результаты по сравнению с ECMP и UCMP и похожие результаты по сравнению с централизованным подходом.

4. ЗАКЛЮЧЕНИЕ

Рост вычислительной и сетевой производительности, а также кардинальные изменения организации современных приложений требуют

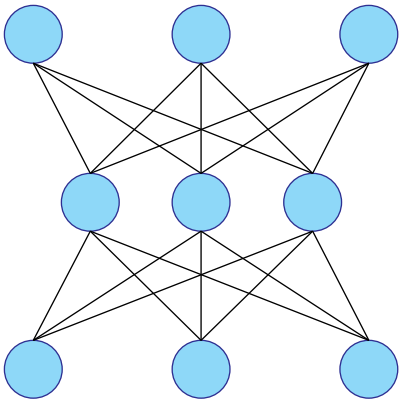

Рис. 3. Топология сети агентов NPCR в сравнительном анализе.

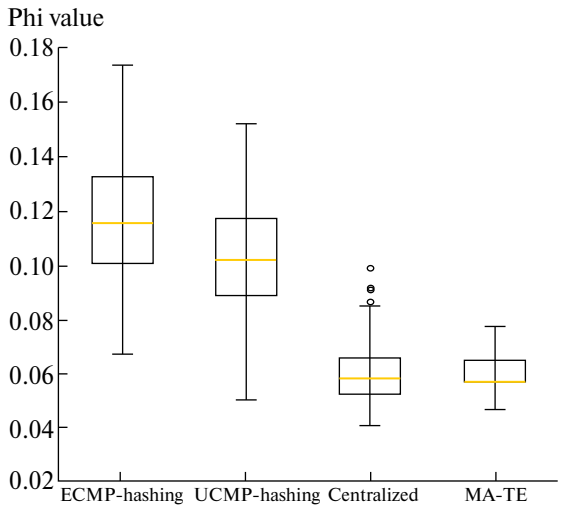

Рис. 4. Сравнение эффективности работы MAROH с ECMP, UCMP и централизованным подходом.

создания новой вычислительной инфраструктуры. В статье представлены архитектура новой Сетевой Вычислительной инфраструктуры и основные принципы ее организации. Режим работы реального времени для современных приложений, скорость передачи данных и обработки данных в современных сетях, размер масштабирования сервисов как сетевых, так и приложений на сегодня таков, что применение классических методов оптимизации управления ресурсами и потоками задач и данных невозможен из-за их вычислительной сложности. Использование методов машинного обучения позволяют эффективно решить разнообразные задачи от прогнозирования состояния вычислительной инфраструктуры до планирования потоков данных внутри наложенной сети. В статье представлен сравнительный анализ метода машинного обучения для управления трафиком в вычислительной среде нового поколения MAROH с широко распространенными методами ECMP и UCMP. Основное достоинство MAROH – распределенное принятие решений, когда каждый NPCR маршрутизатор выполняет балансировку проходящих через него потоков данных на основе своего локального состояния и данных о состоянии его соседей. Экспериментально было показано, что предложенный метод превосходит по критерию равномерности загрузки каналов ECMP и UCMP. Сказанное позволяет утверждать, что методы машинного обучения позволят сделать сетевую вычислительную инфраструктуру предсказуемой, безопасной, надежной, эффективной и масштабируемой.

БЛАГОДАРНОСТИ

Авторы считают своим приятным долгом поблагодарить академика РАН Е.Е. Тыртышникова и доцента С.А. Матвеева за консультации по последним достижениям в области математических методов факторизации и представления матриц, и профессора А.И. Гасникова, доцента А.Н. Безносикова за консультации по вопросу последних достижений в области математических методов распределенной оптимизации.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Работа выполнена при поддержке Программы развития МГУ, проект № 23-Ш03-03.

СПИСОК ЛИТЕРАТУРЫ

1. *Smeliansky R.* Hierarchical edge computing // Int. Conf. Modern Network Tech., MoNeTec-2018. Moscow, 2018. P. 97–105.
2. *Smeliansky R. et al.* On hpc & cloud environments integration. Chapter 1 // Performance evaluation models for distributed service networks. Springer: Springer Nature Switzerland AG Gewerbestrasse 11, 6330 Cham, Switzerland, 2020.
3. *Smeliansky R.* Network Powered by Computing: Next Generation of Computational Infrastructure // Edge Computing Technology, Management and Integration. IntechOpen, 2023. ISBN 978-1-83768-862-3. P. 47–70.
4. Topology and Orchestration Specification for Cloud Applications. http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os. html [Accessed: 2024-19-03]
5. https://www.itprotoday.com/serverless-computing/what-serverless-computing (accessed: March 19, 2024).
6. *Foster I., Kesselman C.* The Grid 2: Blueprint for a new computing infrastructure. Elsevier, 2003.

7. *Foster I., Kesselman. C.* The history of the grid // arXiv:2204.04312, 2022.
8. https://www.cpubenchmark.net/year-on-year.html (accessed: March 19, 2024).
9. https://www.visualcapitalist.com/cp/charted-history-exponential-growth-in-ai-computation/ (accessed: March 19, 2024).
10. https://habr.com/ru/companies/yota/articles/283220/ (accessed: March 19, 2024).
11. *Моисеев Н.Н., Иванилов Ю.П., Столярова Е.М.* Методы оптимизации. М.: Наука, 1978. 352 с.
12. *Karimireddy S.P. et al.* Scaffold: Stochastic controlled averaging for federated learning. International conference on machine learning. PMLR, 2020.
13. *Vogels T., Karimireddy SP., Jaggi M.* PowerSGD: Practical low-rank gradient compression for distributed optimization // Advances in Neural Information Processing Systems. 2019.
14. *Oseledets I., Tyrtyshnikov E.* TT-cross approximation for multidimensional arrays // Linear Algebra and its Applications. 2010. Т. 432. № 1. P. 70–88.
15. *Gusak J. et al.* Automated multi-stage compression of neural networks // Proceedings of the IEEE/CVF Int. Conf. on Computer Vision Workshops, 2019.
16. *Novikov A. et al.* Tensorizing neural networks // Advances in neural information processing systems. 2015. N 28.
17. *Gong Y. et al.* ETTE: Efficient tensor-train-based computing engine for deep neural networks // Proceedings of the 50th Ann. Int. Symp. on Computer Architecture. 2023. P. 1–13.
18. *Смелянский Р.Л., Антоненко В.А.* Концепции программного управления и виртуализации сетевых сервисов в современных сетях передачи данных. М.: Курс, 2019. 160 с.
19. *Bernardez G. et al.* Is machine learning ready for traffic engineering optimization? // 2021 IEEE29th International Conference on Network Protocols (ICNP). IEEE, 2021.
20. *You Xinyu et al.* Toward packet routing with fully distributed multiagent deep reinforcement learning. // IEEE Transactions on Systems, Man, and Cybernetics: Systems 52.2 (2020): 855–868.
21. *Mai Xuan, Quanzhi Fu and Yi Chen.* Packet routing with graph attention multi-agent reinforcement learning // 2021 IEEE Global Communications Conference (GLOBECOM). IEEE, 2021.
22. *Stepanov E. et al.* On Fair Traffic allocation and Efficient Utilization of Network Resources based on MARL // Preliminary on ResearchGate. Available from: https://www.researchgate.net/publication/371166584_On_Fair_Traffic_allocation_and_Efficient_Utilization_of_Network_Resources_based_on_MARL (accessed: November 14, 2023).
23. ECMP Load Balancing. Available from: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mp_l3_vpns/configuration/xe-3s/asr903/mp-l3-vpns-xe-3s-asr903-book/mp-l3-vpns-xe-3s-asr903-book_chapter_0100.pdf (accessed: November 14, 2023).
24. UCMP Load Balancing. Available from: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/mp_l3_vpns/configuration/xe-3s/asr903/17-1-1/b-mpls-l3-vpns-xe-17-1-asr900/m-ucmp.pdf (accessed: November 14, 2023).

ON ML METHODS FOR NETWORK POWERED BY COMPUTING INFRASTRUCTURE

Corresponding Member of the RAS R. L. Smeliansky[a, *], E. P. Stepanov[a, **]

[a]Lomonosov Moscow State University, Faculty of computational mathematics and cybernetics, Department of computing systems and automation, Moscow, Russia

The paper considers the application of machine learning methods for optimal resource management for Network Powered by Computing (NPC) – a new generation computing infrastructure. The relation between the proposed computing infrastructure and the GRID concept is considered. It is shown how machine learning methods applied to computing infrastructure management make it possible to solve the problems of computing infrastructure management that did not allow the GRID concept to be fully implemented. As an example, the application of multi-agent optimization methods with reinforcement learning for network resources management is considered. It is shown that the application of multi-agent machine learning methods makes it possible to increase the speed of distribution of transport flows and ensure optimal NPC network channel load according to the criterion of uniform load distribution, and that such management of network resources is more effective than a centralized approach.

Keywords: reinforcement learning, multi-agent methods, network powered by computing