ВЕСТНИК РОССИЙСКОЙ АКАДЕМИИ НАУК, 2022, том 92, № 1, с. 46–56

ИЗ РАБОЧЕЙ ТЕТРАДИ ИССЛЕДОВАТЕЛЯ

MC2E – МЕТАОБЛАЧНАЯ ВЫЧИСЛИТЕЛЬНАЯ СРЕДА ДЛЯ МЕЖДИСЦИПЛИНАРНЫХ ИССЛЕДОВАНИЙ

© 2022 г. Р. Л. Смелянский

Московский государственный университет имени М.В. Ломоносова, Москва, Россия

E-mail: smel@cs.msu.ru

Поступила в редакцию 29.06.2021 г.
После доработки 06.07.2021 г.
Принята к публикации 18.08.2021 г.

Метаоблачная вычислительная среда (Meta Claud Computing Environment – MC2E) – российско-китайский проект, посвящённый изучению методов и средств построения информационно-вычислительной среды для научных вычислительных экспериментов и междисциплинарных исследований, который завершился в 2020 г. Работы в этой области были сфокусированы на федеративном принципе организации вычислений и управления данными, предусматривающем создание специализированной неоднородной экосистемы из центров обработки данных с виртуализированной инфраструктурой и высокопроизводительных вычислителей, объединённых телекоммуникационными ресурсами – сетью передачи данных. Одна из ключевых проблем построения такой экосистемы, ставшая центральной в проекте MC2E, – интеграция высокопроизводительных вычислителей и центров обработки данных – облачных вычислительных сред на базе кластеров серверов. В статье представлен краткий обзор основных результатов проекта.

Ключевые слова: высокопроизводительные вычисления, суперкомпьютер, облако, центр обработки данных.

DOI: 10.31857/S086958732201008X

Научные междисциплинарные исследования в различных областях науки предполагают сотрудничество нескольких географически разделённых исследовательских групп, доступ к научным коллекциям данных и высокопроизводительным вычислительным ресурсам, для чего необходима информационная, вычислительная и коммуникационная инфраструктура, учитывающая специфику каждого проекта. Её построение традиционным способом (локальный кластер из серверов, или центр обработки данных – ЦОД со средствами виртуализации вычислителей[1] и сетью передачи данных) связано с рядом проблем:

СМЕЛЯНСКИЙ Руслан Леонидович – член-корреспондент РАН, заведующий кафедрой автоматизации систем вычислительных комплексов факультета вычислительной математики и кибернетики МГУ имени М.В. Ломоносова.

• это требует значительных финансовых и материальных вложений и высококвалифицированных IT-специалистов. Проблема усложняется, если в проекте эксперименты ведут независимые исследовательские группы, которые используют разные методы работы, часто имеют разные внутренние бизнес-процессы, аппаратные и программные предпочтения и могут располагаться далеко друг от друга; кроме того, эксперименты в разных предметных областях могут занимать разное время;

• на начальной стадии требования к инфраструктуре проекта известны приближённо и, как правило, завышены, что ведёт к потере эффективности инвестиций;

• работа усложняется, когда научные данные распределены и используются разными командами одновременно;

[1] Далее локальный кластер из серверов, или ЦОД со средствами виртуализации, будет обозначаться аббревиатурой HPC-C – High Performance Computing Cloud.

• данные, необходимые одной команде, могут принадлежать другой – без специализированной системы управления инфраструктурой вопросы правообладания данными регулировать сложно;

• разные группы исследователей одного и того же проекта могут использовать разные инструменты, имеющееся программное обеспечение для обработки, сбора и хранения данных. Создание или освоение нового инструментария для них обычно неприемлемо, поэтому необходимо предоставить возможность привносить в информационно-вычислительную среду проекта уже реализованные разработки.

Основным инструментом для численных экспериментов и моделирования всегда были высокопроизводительные вычисления (High Performance Computing – HPC). Вычислительные ресурсы для них предоставляются суперкомпьютерами и специализированными кластерами серверов. Однако анализ данных, представленных на сайте супервычислителей TOP500.org [1], позволяет говорить о том, что количество научных приложений растёт быстрее, чем количество суперкомпьютеров и установок высокопроизводительных вычислений[2]. В то же время мы видим быстрый рост популярности облачных вычислений, использование сетей центров обработки данных (Data Center Network – DCN) для увеличения вычислительной мощности платформ облачных вычислений. Хорошим примером здесь может служить Ассоциация EGI [2].

Суперкомпьютеры (HPC-S) и облачные среды ЦОД, или кластер серверов HPC (HPC-C), обладают разными вычислительными возможностями и отличаются по своей загрузке. Большинство приложений будет работать на суперкомпьютере быстрее, чем на кластере серверов в ЦОД. Однако общее время получения результата вычислительного эксперимента, то есть ожидание в очереди плюс время выполнения на суперкомпьютере HPC-S, может оказаться больше, чем время выполнения плюс время ожидания в очереди в облачной среде на кластере серверов HPC-C. Мы будем называть эту общую задержку временем программы в системе (критерий PTS).

Миссия проекта MC2E заключалась в том, чтобы исследовать, как должна быть организована среда, которая позволяет создавать эффективную по критерию PTS информационно-вычислительную инфраструктуру, отвечающую перечисленным особенностям конкретного междисциплинарного проекта. Одна из пока малоизученных проблем этой миссии – интеграция двух довольно разных сред высокопроизводительных вычислений – суперкомпьютеров (HPC-S) и облачных сред (HPC-C), различающихся по многим параметрам: уровню и способам управления ресурсами, технике виртуализации, составу параметров и форме спецификации запроса на выполнение программы, планированию и политике выделения ресурсов. Так, для выделения ресурсов в среде HPC-S характерно использование методов резервирования и изоляции, то есть когда разные вычисления не могут разделять один и тот же физический ресурс. Для сред HPC-C характерно выделение ресурсов по требованию и совместное использование несколькими программами одного и того же физического ресурса. Облачная среда предлагает бóльшую гибкость и удобство для работы с ресурсами по сравнению с HPC-S, предоставляя виртуализированные ресурсы, настроенные для конкретных целей. Перечисленные выше различия платформ HPC-S и HPC-C затрудняют автоматическое переключение задач между ними, если какая-либо из них становится сильно загруженной. Таким образом, чтобы изменить целевую платформу, исследователям необходимо потратить время и ресурсы на настройку своего программного обеспечения.

Ещё одна проблема на пути к интеграции HPC-S и HPC-C заключается в том, как в гетерогенной интегрированной среде из двух платформ правильно выбрать вычислитель для выполнения MPI-программы[3], использующей библиотеку MPI (Message Passing Interface), – средство для взаимодействия между параллельными ветками вычислителей одной и той же программы. Другими словами, для каждой MPI-программы, находящейся в очереди, необходимо принять решение, где её выполнение будет более эффективно с точки зрения критерия PTS. Занимаясь проблемой интеграции, необходимо обосновать гипотезу о том, что совместное использование физических ресурсов в среде HPC-C несколькими MPI-программами одновременно сократит общее время их выполнения, то есть это время будет меньше, чем сумма времён последовательного выполнения каждой.

Другая трудность заключается в способности среды агрегировать ресурсы сети передачи данных между ЦОДами в DCN (Data Communication Network). Здесь ключевая проблема состоит в реализации сервиса выделения канала для передачи данных по требованию (Bandwidth on Demand – BoD). Этот сервис должен выделять по запросу каналы между двумя или более ЦОДами с соответствующим качеством, то есть способных передать определённое количество данных за определённый интервал времени через сеть передачи данных, например, Интернет. Следует подчеркнуть, что такой сервис не предполагает наличие

[2] Далее этот класс высокопроизводительных вычислителей для краткости будем обозначать HPC-S.

[3] Далее для краткости вместо MPI-программа будем использовать термин “программа”.

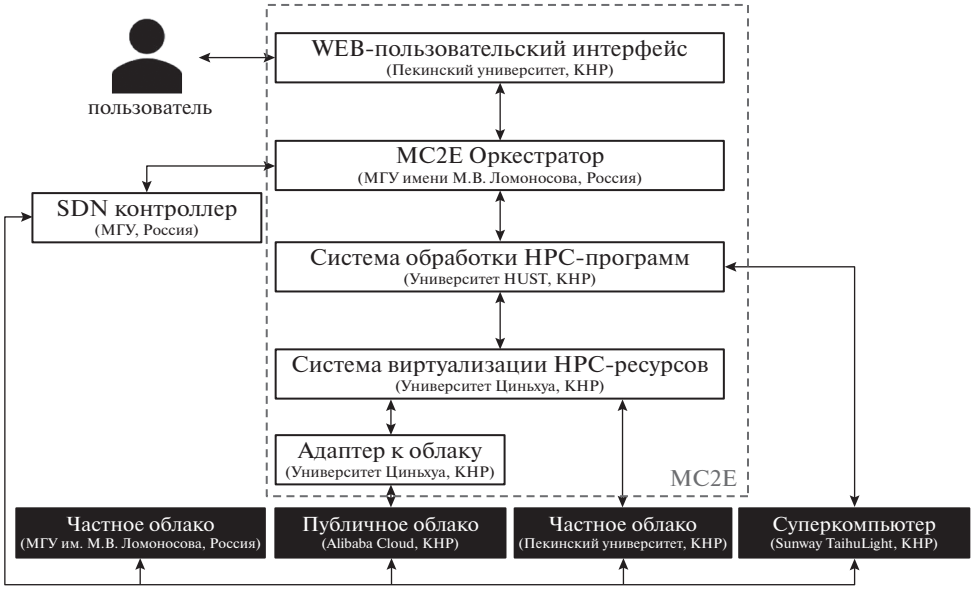

Рис 1. Структура MC2E

выделенного канала между взаимодействующими вычислителями. Более того, он должен создаваться динамически, путём объединения существующих сетевых ресурсов (линии передачи, коммутаторов и т.п.).

После вводных тезисов представим основные результаты проекта MC2E: принципы построения структуры среды MC2E; экспериментальное исследование влияния сети в ЦОД на совместное использование процессоров в облаках и возможность совместного применения этих ресурсов MPI-программами; новые подходы к прогнозированию времени выполнения программ на определённом наборе вычислителей для определения наиболее эффективного места выполнения программы; подходы к реализации сервиса BoD. С подробным отчётом по проекту можно ознакомиться в работе [3].

СТРУКТУРА MC2E

Перечислим основные принципы организации среды MC2E:

• инфраструктура представляет собой объединение локаций, оснащённых вычислителями, хранилищами и сетевыми ресурсами, называемых федератами;

• федерация управляет всеми ресурсами (центральный процессор, память, сеть передачи данных, программное обеспечение), предоставляемыми федератами;

• ресурсы одного федерата могут одновременно использовать разные проекты;

• ресурсы федерата могут быть виртуализированными;

• ресурсы на уровне пользователя имеют высокий уровень абстракции и их использование не должно подразумевать высокую квалификацию пользователя;

• результаты экспериментов всегда должны сохраняться и могут использоваться другими для воспроизведения или продолжения эксперимента;

• федерация предоставляет сервисы по обработке данных, другими словами, федерация – это компьютер.

На рисунке 1 показана организация среды MC2E с распределением ответственности между международными участниками проекта.

Примером вычислительного ресурса федерата может быть высокопроизводительный вычислитель либо ЦОД. У каждого федерата есть своя собственная политика, которая регулирует распределение ресурсов между участниками проектов. Идея построения информационно-вычислительной инфраструктуры как федерации разнородных вычислительных установок уже использовалась во многих действующих проектах. Некоторые из них предназначались для проведения экспериментов в области компьютерных сетей. Например, проект GENI – Global Environment for Network Innovation [4], инициированный Национальным научным фондом США (NSF), представляет собой виртуальную лабораторию для сетевых экспериментов в международном масштабе. Сегодня более 200 университетов США подключены к среде GENI. Другой поддержанный NSF проект – FABRIC [5] – предлагает адаптивную программируемую инфраструктуру для исследований в области компьютерных наук. К аналогич-

ным, но менее известным проектам относятся Ophelia [6] и Fed4Fire [7]. Первый реализован при поддержке ЕС, второй поддержан 17 компаниями из 8 европейских стран. Есть и другие, которые обеспечивают среду для вычислительных экспериментов независимо от прикладной области [8–10]. Однако у всех этих проектов есть несколько существенных недостатков:

• слабая интеграция между HPC-S и HPC-C, не позволяющая автоматически выбирать эффективный вычислительный ресурс;

• невозможность строить цепочки сервисов для проведения экспериментов;

• нельзя автоматически переносить уже существующее программное обеспечение в новую среду;

• при планировании ресурсов не учитывается масштабирование производительности сервисов;

• отсутствие служб контроля и управления DCN, таких как мониторинг и службы BoD.

Базовыми технологиями для проекта MC2E при разработке виртуальной инфраструктуры для междисциплинарных исследований служили программно-конфигурируемые сети (SDN) и виртуализация сетевых функций (NFV) [11]. Эти технологии позволяют повысить уровень абстракции ресурсов, обеспечить их согласованную оптимизацию и автоматизацию управления инфраструктурой. Вместо отдельных ресурсов пользователи получают виртуальные инфраструктуры, полностью укомплектованные необходимыми ресурсами – вычислительными мощностями, каналами связи и системами хранения – с гарантированной производительностью и качеством обслуживания (QoS) на основе соглашения об уровне сервиса (SLA).

Фактически, в проекте MC2E использовались две облачные среды: Docklet [12] и Cloud Conductor (C2) [13]. Предназначение Docklet – предоставить индивидуально настроенное рабочее пространство в облачной среде [14]. Основа Docklet – LXC виртуальный кластер [15]. Пользователи Docklet взаимодействуют непосредственно в своей рабочей области, используя браузер для разработки, отладки, тестирования своего программного обеспечения с помощью инструментов, которые предоставляет эта среда. Docklet позволяет строить индивидуальные небольшие виртуальные центры обработки данных, создавая виртуализированные кластеры, а затем предоставляя экспериментаторам настроенное рабочее пространство в облаке. Пользователям нужен только современный браузер для доступа в свою рабочую область, расположенную в сети федерата, из любого места в Интернете и в любое время.

Архитектура платформы C2 основана на эталонной реализации модели ETSI NFV MANO [13]. C2 обеспечивает полную поддержку жизненного цикла виртуализированных сервисов (ВС) – инициализацию, конфигурацию, выполнение и деинициализацию, которая осуществляется с помощью так называемых шаблонов на языке TOSCA [13]. Всё, что нужно для установки и запуска ВС, – шаблон TOSCA, включающий описание структуры облачного приложения, политики управления приложениями, образа операционной системы и сценариев запуска, остановки и настройки приложения, реализующего ВС. TOSCA формирует администратор сервиса в виде архива zip или tar.

Такая среда, построенная на основе федерации, имеет следующие преимущества:

• простота и скорость выделения, настройки и масштабирования ресурсов;

• разработка приложения как цепочки из сервисов на основе технологии NFV;

• объединение инфраструктур разных исследовательских групп на основе единой политики доступа;

• автоматизированное планирование ресурсов для выполнения запросов пользователей на основе политики доступа и требований SLA;

• язык шаблонов для описания приложений, позволяющий абстрагироваться от низкоуровневых системных деталей;

• децентрализованная система учёта ресурсов для взаиморасчётов между участниками проекта;

• широкие возможности для отслеживания и мониторинга экспериментов;

• повышенная эффективность виртуализации сети благодаря технологии программно-конфигурируемых сетей (ПКС), что позволяет настраивать виртуализированные сетевые каналы для каждого конкретного эксперимента;

• общий язык спецификаций, необходимый для переноса существующего исследовательского программного обеспечения (ПО) в среду MC2E.

Перечислим основные компоненты или подсистемы среды MC2E:

• мета-облако – оркестровка пользовательских приложений, распределение и планирование их между федератами;

• интерфейс – предоставляет пользователям унифицированный API для отправки своих приложений и федеративного администратора с целью управления и контроля ресурсов федерации;

• сеть – регулирует использование сетевых ресурсов и предоставляет услугу BoD;

• монитор – выполняет мониторинг и учёт потребления ресурсов всех федератов в MC2E;

• качество обслуживания, административный контроль и управление – обеспечивают соблюдение политики использования ресурсов на основе

требований пользователя (SLA) и гарантируют доступность ресурсов.

Эти компоненты подробно описаны в работе [16]. Общий процесс выполнения запроса пользователя в среде MC2E выглядит следующим образом:

1. С помощью единого интерфейса MC2E пользователь отправляет своё приложение и данные на интерфейсный сервер.

2. Интерфейсный сервер вызывает планировщик и монитор мета-облака, чтобы выбрать федерата для выполнения приложения.

3. Мета-облако анализирует очередь и прогнозирует время выполнения приложения и время передачи данных для всех доступных федератов.

4. Основываясь на прогнозе, мета-облако выбирает федерата, который минимизирует общее время пребывания приложения в системе (критерий PTS – время ожидания в очереди + время выполнения).

5. Мета-облако обращается к контуру управления сетью для прокладки канала к выбранному на шаге 4 федерату и отправляет туда приложение и его данные.

6. Федерат запускает приложение и возвращает результаты пользователю.

7. В случае сбоя подсистема мониторинга и управления качеством обслуживания перенаправляет приложение средствами мета-облака в другой федерат.

ОБЛАКО КАК СРЕДА ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Несмотря на то, что скорость вычисления в облачных средах ниже, чем в специализированных серверных кластерах или суперкомпьютерах [17], такая технология становится всё более популярной и выбирается в качестве платформы для высокопроизводительных вычислений из-за низкой стоимости и простоты доступа. В нескольких статьях [18, 19] показано, что одно из основных узких мест производительности HPC-C связано с задержками в сети передачи данных (СПД) в ЦОД. Суперкомпьютеры используют быстрые СПД-сети на основе специализированных средств [20, 21]. HPC-C в основном полагаются на обычный Ethernet. Задержки в СПД могут привести к недозагрузке процессоров (ЦП) приложениями с высокой интенсивностью обмена данными, поскольку в них могут возникать длительные “паузы” в вычислениях из-за ожидания передачи данных по сети между ветвями приложения.

Один из важных результатов проекта MC2E – экспериментальное обоснование гипотезы о том, что производительность HPC-приложений может незначительно деградировать при совместном использовании ядер ЦП. Эта гипотеза была проверена в рамках проекта MC2E с помощью теста HPC – NAS Parallel Benchmarks (NPB) [22] в облачной среде мини-ЦОД с гипервизором QEMU/KVM с 64 виртуальными машинами (ВМ) (Ubuntu 16.04, 1 vCPU, 1024 Мб ОЗУ), MPI версия 3.2. На головном сервере было 16 ВМ, на остальных – по 8 ВМ на каждом. Средняя задержка между разными виртуальными машинами – около 400 мкс. Пропускная способность составляла на одном сервере 18.2 Гб/с, на разных серверах – 5.86 Гб/с.

Для изучения влияния пропускной способности сети на загрузку ЦП была проведена серия экспериментов. Последовательно 5 NPB MPI-программ с 2, 4, 8, 16, 32, 64 процессами на каждой виртуальной машине запускались в сети с тремя пропускными способностями каналов: 100 Мб/с, 1 и 10 Гб/с соответственно. Эксперименты показали почти линейную зависимость деградации загрузки ЦП от роста числа MPI-процессов. Это происходило потому, что разные MPI-процессы выполнялись на разных ВМ, а данные передавались между разными серверами. Загрузка ЦП также снижалась, когда программа выполнялась на одном физическом сервере (2, 4 и 8 ЦП). Сокращение загрузки ЦП как раз и позволяет использовать ядра одного и того же ЦП между разными программами. Подробно эта серия экспериментов с графиками представлена в работе [16].

В другой серии экспериментов изучалась способность различных HPC-приложений совместно использовать ядра одного и того же ЦП. Методология эксперимента заключалась в следующем: последовательно запускались 5 пар идентичных NPB MPI-программ с одинаковым числом (2, 4, 8, 16, 32, 64) процессов в каждой. Для оценки способности программы совместно использовать ресурсы ЦП применялась метрика:

$$Queue\ metric = \frac{T^1_{pure} + T^2_{pure}}{\max(T^1_{sharing}, T^2_{sharing})},$$

где T^i_{pure} $(i = 1.2)$ – время выполнения без разделения ресурсов; $T^i_{sharing}$ $(i = 1.2)$ – время выполнения, когда две программы использовали одни и те же ядра одного и того же ЦП.

Нетрудно увидеть, что если значение метрики больше 1, то выполнение двух программ, запускаемых одновременно, займёт меньше времени, чем при их последовательном запуске. Эксперименты показали, что даже в среде с медленной сетью (100 Мб/с) можно получить до 20% ускорения выполнения. Однако не все программы могут эффективно разделять физические ресурсы. Подробно эта серия экспериментов описана в работе [16].

ПРОГНОЗИРОВАНИЕ ВРЕМЕНИ ВЫПОЛНЕНИЯ ПРОГРАММ

Напомним, что один из основных критериев эффективности среды облачных вычислений для HPC-приложений – время, проведённое программой в этой среде (критерий PTS). Данная величина зависит от алгоритмов распределения ресурсов в облачной среде (отображение виртуальных вычислителей на физические) и дисциплины обслуживания очереди программ с учётом неоднородности физических вычислителей.

Из анализа процесса выполнения запроса пользователя, представленного в конце раздела "Структура MC2E", видно, что этапы 3 (метапланирование) и 4 (локальное планирование) – точки оптимизации по критерию PTS. Важной частью проекта MC2E стали исследование и разработка алгоритма, выбор наиболее эффективного вычислителя в федерации для определённой программы на основе критерия минимального времени выполнения – важного компонента критерия PTS. Для этого специалисты изучили и разработали несколько алгоритмов прогнозирования времени выполнения программы на определённом наборе вычислителей с учётом истории выполнения программы на разных вычислителях (подробное описание алгоритмов см. [23]).

Задача прогнозирования времени выполнения программы на определённом вычислителе хорошо известна и относится к классическим. Например, время выполнения программы на конкретном вычислителе, а также время ожидания в очереди можно спрогнозировать, имея историю её запусков на этом вычислителе [24–27], для чего могут быть использованы многие алгоритмы экстраполяции, например [28, 29] или регрессии [30], или более сложные алгоритмы, в частности, ансамбль деревьев принятия решений (рандомный лес) [31]. Основной недостаток этих алгоритмов заключается в том, что их можно применять только к одному и тому же вычислителю. Но суть этой задачи в MC2E заключалась в прогнозировании времени выполнения программы на определённом *наборе* вычислителей. Конечно, упомянутые алгоритмы можно использовать для оценки времени выполнения программ на нескольких вычислителях. Однако для этого нужна история запуска каждой программы на каждом вычислителе из набора. Эта информация недоступна.

Выбор вычислителя осуществляется следующим образом. Одним из хорошо известных алгоритмов [24–27] оценивается время выполнения программы по историям её проведения на каждом вычислителе из определённого набора. Примером такой истории может быть трасса выполнения программы [32]. Основываясь на полученных данных, можно либо построить расписание для группы программ, либо, следуя жадной стратегии, отправлять каждую программу на вычислитель, где у неё минимальное время выполнения. Однако необходимы все истории выполнения всех программ на каждом из вычислителей в рассматриваемом наборе. Такой информации, как правило, нет.

Один из важных и новых результатов проекта MC2E – разработка метода прогнозирования времени выполнения программы, который позволяет ослабить требование данных обо всех историях выполнения всех программ на каждом из вычислителей в рассматриваемом наборе: для прогнозирования времени выполнения программы на вычислителях из определённого набора достаточно только историй выполнения этой программы на некоторых из них (подробное описание метода см. [23]). Другими словами, нет необходимости запускать каждую программу на каждом вычислителе.

Основная идея нового подхода к задаче прогнозирования времени выполнения программы заключалась в том, что рассматриваемая проблема очень похожа на решаемую в системах рекомендаций, или рекомендательных системах (РС) [33]. РС относится к подклассам системы фильтрации информации, стремящейся предсказать рейтинг или предпочтение пользователя некоторого объекта [34]. Примерами такого объекта могут быть фильмы, книги или любые другие товары. Таким образом, рекомендательная система восстанавливает, прогнозирует отношения между пользователями и товарами на основе отдельных пользовательских оценок предпочтения.

У РС есть рейтинговая матрица, в которой строки (или столбцы) соответствуют фильмам, книгам или товарам, а столбцы (или строки) соответствуют пользователям. Эта матрица часто бывает разреженной, поскольку в списке присутствует множество пользователей и элементов, поскольку пользователи не могут физически оценить все рассматриваемые элементы. Система пытается предсказать предпочтения каждого пользователя для каждого элемента на основе индивидуальных оценок пользователей для некоторых элементов. Другими словами, РС должна заполнить пустые ячейки в рейтинговой матрице. В этих терминах рассмотрим следующую аналогию: пользователи – это вычислители, элементы – это программы, а оценки пользователей – времена выполнения программ. Таким образом, вычислители "оценивают" программы, при этом чем меньше рейтинг (время выполнения), тем лучше.

В результате задача прогнозирования времени выполнения программы была сведена к задаче заполнения пустых ячеек в матрице "Программы–Вычислители", построенной для заданного набора программ и заданного набора вычислителей. В ячейках этой матрицы указано время выполне-

ния конкретной программы с конкретными наборами данных, соответствующих конкретному вычислителю.

Должно быть ясно, что точность прогноза зависит от количества известных историй выполнения программ на вычислителе из определённого набора. Нами было исследовано два подхода к этой задаче. Первый базировался на группировке вычислителей на основе корреляции Пирсона [35] и показал [23], что данный подход целесообразно применять в случае плотно заполненной (не менее 95% ячеек) матрицы “Программы–Вычислители”.

Второй подход, разработанный для разреженной матрицы “Программы–Вычислители”, подразумевал её разложение на векторные представления вычислителей и программ – так называемые embedding’и. Embedding – это элементы относительно малоразмерного пространства, в которое можно преобразовывать векторы бóльшей размерности. Техника embedding упрощает машинное обучение для случая больших данных, таких как разреженные векторы, представляющие слова. В идеале она частично отражает семантику данных, помещая семантически похожие объекты близко друг к другу в пространстве embedding’ов [36]. В статье [23] подробно показано, как использовать embedding программ и embedding вычислителей для прогнозирования времени выполнения программы на конкретном вычислителе. Техника декомпозиции матрицы “Программы–Вычислители” для расчёта embedding дана в работе [37].

Предложенный подход к прогнозированию времени выполнения программы требует минимальных знаний о программе, которые обычно собирают на всех современных вычислителях. Другое важное преимущество подхода – возможность декомпозиции матрицы “Программы–Вычислители” на embedding и для программ, и для вычислителей размерности 1. Этот факт позволяет тотально упорядочить как вычислители, так и программы, что существенно упрощает подбор вычислителя с эффективным временем выполнения. Подробнее о тестировании методов прогнозирования см. [23].

СЕРВИС “КАНАЛ ПО ТРЕБОВАНИЮ” (BoD)

Как было отмечено в начале статьи, одной из исследованных в проекте MC2E была проблема динамического создания канала между ЦОДами по требованию с заданным качеством связи. Среда, предназначенная для междисциплинарных научных исследований, должна иметь гибкие и мощные механизмы распределения, планирования и администрирования сетевых ресурсов. В противном случае накладные расходы на сетевые ресурсы в сети ЦОДов будут очень высокими, так как нагрузка на этот ресурс носит спорадический характер. Естественно, возникал вопрос о возможности создания сервиса BoD, способного формировать между ЦОДами канал надлежащего качества по требованию, то есть передавать определённое количество данных за определённый интервал времени через транспортную сеть TCP/IP. Следует подчеркнуть, что такой сервис не предполагает выделенного канала между взаимодействующими сторонами. Более того, он должен создаваться динамически путём агрегирования существующих сетевых ресурсов.

В статье [38] подробно рассмотрены протоколы агрегации сетевых ресурсов и алгоритмы для создания сервиса BoD. Здесь представлена лишь логическая схема подхода, предложенного в цитируемой статье. Создание сервиса BoD можно разделить на две основные части: идентификация и агрегация маршрутов в сети ЦОДов и распределение потоков данных между ними согласно требованиям качества сервиса BoD для каждого потока.

Слова “построение и агрегация маршрутов” означают, что для передачи потока данных между ЦОДами одновременно используют разные маршруты и агрегируют несколько физических линий в одну логическую. Есть много протоколов и технологий, которые позволяют сделать это [38]. Однако все они оставляют открытой проблему качества обслуживания.

Для построения маршрутов необходимо решить несколько задач.

Первая: сколько и какие маршруты необходимы, чтобы удовлетворить требования качества сервиса BoD? Маршруты должны иметь минимальные пересечения по линиям связи и коммутаторам/маршрутизаторам. Это ограничение проистекает из особенностей работы алгоритмов управления перегрузкой на транспортном уровне в сети. Количество рёберно-непересекающихся путей между двумя вершинами в графе определяется теоремой Менгера [39], которая утверждает, что наибольшее количество непересекающихся по рёбрам маршрутов от вершины u к вершине v в графе равно наименьшему количеству рёбер в разрезе <u, v> этого графа.

Отсутствие пересечения маршрутов не всегда относится к критическим моментам. Например, если физическая линия на пересечении имеет достаточную пропускную способность, чтобы удовлетворить требованиям качества сервиса для всех проходящих через неё потоков, то эта линия, будучи пересечением маршрутов, критическим местом не является. Возможно, что в топологии сети нет альтернативных непересекающихся маршрутов между источником и пунктом назначения. Однако при наличии физических каналов

с высокой пропускной способностью можно преобразовать граф топологии сети так, чтобы ребро, соответствующее такой линии, было заменено несколькими рёбрами с меньшей пропускной способностью. Альтернативным решением может быть поиск маршрутов с наименьшим количеством пересечений, как это делает алгоритм Min Cost Max Flow – MCMF [40].

После нахождения значения k – количества непересекающихся маршрутов, граф топологии сети обрабатывают специальным алгоритмом для определения k-маршрутов между источником и пунктом назначения. Оказалось, что для этого подходит далеко не каждый алгоритм. Например, жадный алгоритм [41] не решит эту проблему. Правильным выбором оказался алгоритм MCMF [42]. Он сводит исходную задачу к поиску максимального потока в сети. В результате набор маршрутов с ресурсами, достаточными для предоставления сервиса BoD, будет идентифицирован.

Вторая задача: как распределить ресурсы этих маршрутов при передаче прикладного потока, то есть как решить проблему AFLD (Applicaton Flow Load Distribution)? Она была разделена на три части: оценка ресурсов, распределение ресурсов и реализация рассчитанного распределения между найденными маршрутами. Решение первой даёт ответ на вопрос, достаточно ли доступной пропускной способности идентифицированных маршрутов для обеспечения нужного качества сервиса BoD? Если это так, то решение второй части проблемы даёт ответ на вопрос, как распределить нагрузку потока данных приложения между этими маршрутами. При решении третьей части приходится иметь дело с многопоточными протоколами, то есть использующими несколько маршрутов одновременно. Эти протоколы разделяют прикладной поток на несколько транспортных подпотоков, каждый из которых использует свой маршрут.

Существуют два основных подхода к многопоточной маршрутизации на транспортном уровне: статический и динамический. MPTCP – это статический подход [42], предполагающий априорное выделение определённого количества транспортных подпотоков (то есть маршрутов), между которыми распределяются сегменты прикладного потока. Динамический подход, например FDMP [43], использует динамическое выделение маршрута для подпотока по запросу транспортного агента в зависимости от соответствия общей пропускной способности текущего множества подпотоков требованию качества сервиса.

Для решения AFLD-проблемы была разработана математическая модель многопоточной передачи данных по требованию между ЦОДами при следующих предположениях:

• чтобы воспользоваться сервисом BoD, пара ЦОДов должна заключить контракт с оператором сети передачи данных, к которой подключены ЦОДы, где оговариваются максимально допустимые объёмы передаваемых данных и максимально допустимое для этого время, качество связи и т.д.;

• по одному и тому же контракту невозможно одновременное появление двух и более запросов на сервис BoD;

• вероятность распределения потоков запросов на сервис по каждому контракту известна.

На основе этой модели проблема AFLD была сформулирована как задача целочисленного линейного программирования (ЦЛП) с дискретным временем [16]. Её решение в форме ЦЛП дало ответы на следующие вопросы: достаточно ли имеющейся пропускной способности на выявленных маршрутах для удовлетворения всех потоков запросов при заданном наборе контрактов на определённом интервале времени? Как должна быть распределена пропускная способность на каждом маршруте между потоками, возникающими по требованию?

На базе маршрутизаторов Juniper VMX был построен прототип предложенных выше подходов к реализации сервиса BoD на основе технологии VPN, апробированный между ЦОДами на факультете вычислительной математики и кибернетики МГУ имени М.В. Ломоносова и ЦОДами Пекинского университета [3]. В апреле 2021 г. успешно завершился пилотный проект реализации сервиса BoD между ЦОДами в Москве и Новосибирске [44].

* * *

Международный проект MC2E нацелен на исследование методов построения среды для академических междисциплинарных исследований. MC2E-среда была простроена как федерация локальных вычислительных сред, называемых федератами. Каждый федерат представляет собой высокопроизводительный кластер – либо ЦОД, либо суперкомпьютер. К преимуществам предложенного метода можно отнести:

• высокий уровень управления ресурсами и гибкие возможности для определения виртуальных сред;

• возможность использовать имеющееся программное обеспечение исследователей;

• высокое качество планирования и эффективность использования ресурсов по критерию PST;

• освобождение пользователя от рутинных задач системного администрирования, а также определение единого способа описания жизнен-

ного цикла виртуализированного сервиса в ЦОД (или в HPC-кластере).

В ходе проекта экспериментально обоснована гипотеза о том, что можно получить сокращение среднего времени выполнения программ в облаке. Эксперименты продемонстрировали, что на определённых классах приложений можно получить до 20% сокращения времени.

Разработано новое решение проблемы выбора подходящего вычислителя MPI-программы в гетерогенной среде на основе критерия PST. Для этого был предложен новый подход к прогнозированию времени выполнения программы MPI на вычислителе, даже если она на нём никогда не выполнялась. Построены и проанализированы два алгоритма: базирующийся на группировке вычислителей с использованием корреляции Пирсона (для плотной матрицы “Программа–Вычислитель”) и основанный на технике разложения такой матрицы, позволяющий получать векторные представления (embedding) программы и вычислителя (для разреженной матрицы “Программа–Вычислитель”).

Следует подчеркнуть, что предложенный подход к прогнозированию времени выполнения программы требует минимального набора данных об истории выполнения программ, который обычно доступен. Другое его важное преимущество заключается в том, что в результате матричной декомпозиции embedding предоставляется и для программ, и для вычислителей размерности 1. Этот факт позволяет установить тотальное упорядочение как на заданном множестве вычислителей, так и на заданном наборе программ, что существенно упрощает выбор вычислителя с эффективным временем выполнения. В качестве гипотезы в проекте сформулировано предположение, что данный метод прогнозирования времени выполнения может быть применён не только к MPI-программам.

Кроме того, в ходе проекта MC2E предложено и исследовано решение для построения сервиса BoD. Этот подход разделён на проблемы агрегации маршрутов и распределения прикладных потоков. Изучены разные варианты решения задачи агрегации маршрутов в зависимости от топологии сети и возможностей сетевого оборудования. Вопрос о распределении прикладных потоков между агрегированными маршрутами сформулирован и решён в виде задач ЦЛП.

Было бы наивно полагать, что представленные результаты полностью охватывают все проблемы, возникающие при создании информационно-вычислительных сред для междисциплинарных исследований. Перечислим лишь несколько из оставшихся за рамками проекта MC2E: автоматизация запуска программ на различных вычислителях в среде, где есть ресурсы HPC-C и HPC-S; скоординированное управление данными и сетевыми ресурсами в реальном времени; мониторинг и аналитика, администрирование и безопасность в таких средах [45]; учёт потребляемых ресурсов; взаиморасчёты между участниками федерации; использование технологий периферийных вычислителей [46].

БЛАГОДАРНОСТИ

Автор выражает огромную благодарность всем участникам совместной работы в проекте MC2E: заведующему лабораторией, старшему научному сотруднику Виталию Антоненко, ведущему научному сотруднику Анатолию Бахмурову, аспирантам Ивану Петрову, Андрею Чупахину и Алексею Колосову, магистранту Глебу Ишелеву (Московский государственный университет имени М.В. Ломоносова, Россия); профессору (главному исследователю) Сянцюнь Чен, профессору Чен Мин, профессору-исследователю Дунган Цао, докторанту Цзюньмин Ма (Пекинский университет, Китай); исследователю Вэньлай Чжао (Университет Цинхуа, Китай); профессору Мин Чен (Хуачжунский университет науки и технологий, Китай).

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Работа выполнялась при поддержке гранта Министерства науки и высшего образования России № 05.613.21.0088 (уникальный идентификатор RFMEFI61318X0088) и Национальной ключевой программы исследований и разработок Китая (2017YFE0123600).

ЛИТЕРАТУРА

1. *Meuer H. et al.* The Top500 project. 2019. http://www.top500.org/
2. *Kranzlmüller D., de Lucas J.M., Öster P.* The European grid initiative (EGI) // Remote instrumentation and virtual laboratories. Boston, MA, USA: Springer, 2010. P. 61–66.
3. Отчёт НИР “Исследование и разработка метаоблачной вычислительной среды”. Регистрационный номер AAAA-A19-119011190172-9. https://rosrid.ru
4. *Hwang T.* NSF GENI cloud enabled architecture for distributed scientific computing // 2017 IEEE Aerospace Conference. March 4–11, 2017, Big Sky, MT, US. IEEE. P. 1–8.
5. *Baldin I.N., Griffioen A., Monga J. et al.* FABRIC: A National-Scale Programmable Experimental Network Infrastructure // IEEE Internet Computing. 2019. V. 23. № 6. P. 38–47.
6. *Dewar R.G., MacKinnon L.M., Pooley R.J. et al.* The OPHELIA Project: Supporting Software Development in a Distributed Environment // ICWI. P. 568–571.
7. Fed4Fire project. 2019. https://www.fed4fire.eu/the-project/
8. *Grossman R.L., Gu Y., Mambretti J. et al.* An overview of the open science data cloud // Proceedings of the

19th ACM International Symposium on High Performance Distributed Computing. ACM. P. 377–384.

9. *Bal H.E., Bhoedjang R., Hofman R. et al.* Performance evaluation of the Orca shared-object system // ACM Transactions on Computer Systems (TOCS). 1998. V. 16. № 1. P. 1–40.
10. *Brun R., Urban L., Carminati F. et al.* GEANT: detector description and simulation tool // CERN Program Library Long Writeup W5013, 1993.
11. *Смелянский Р.Л., Антоненко В.А.* Концепции программного управления и виртуализации сетевых сервисов в современных сетях передачи данных. М.: Курс, 2020.
12. *Cao D.-G., An B., Shi P.-C., Wang H-M.* Providing Virtual Cloud for Special Purposes on Demand in Joint-Cloud Computing Environment // Journal of Computer Science and Technology. 2017. V. 32. № 2. P. 211–218.
13. *Antonenko V., Smeliansky R., Ermilov A. et al.* C2: General Purpose Cloud Platform with NFV Life-cycle Management // 2017 IEEE 9th International Conference on Cloud Computing Technology and Science. December 11–14, 2017, Hong Kong, China. IEEE. P. 353–356.
14. *An B., Shan X., Cui Z. et al.* Workspace as a Service: an Online Working Environment for Private Cloud // 2017 IEEE Symposium on Service-Oriented System Engineering (SOSE). April 6–9, 2017, San Francisco, CA, USA. IEEE. P. 19–27.
15. *Cao D.-G., Liu P., Cui W. et al.* Cluster as a Service: a Resource Sharing Approach for Private Cloud // Tsinghua Science and Technology. 2016. V. 21. № 6. P. 610–619.
16. *Antonenko V., Chupakhin A., Kolosov A. et al.* On HPC & Cloud Environments Integration // Performance Evaluation Models for Distributed Service Networks / Ed. by G. Bocewicz, J. Pemoera, V. Toporkov. Springer Nature Switzerland AG, 2020. P. 159–185.
17. *Netto M.A., Calheiros R.N., Rodrigues E.R. et al.* HPC cloud for scientific and business applications: Taxonomy, vision, and research challenges // ACM Computing Surveys (CSUR). 2018. V. 51. № 1. P. 8.
18. *Gupta A., Faraboschi P., Gioachin F. et al.* Evaluating and improving the performance and scheduling of HPC applications in cloud // IEEE Transactions on Cloud Computing. 2016. V. 4. № 3. P. 307–321.
19. *Gupta A., Milojicic D.* Evaluation of HPC applications on cloud // Sixth Open Cirrus Summit. October 12–13, 2011, Atlanta, GA, USA. P. 22–26.
20. Infiniband in supercomputer systems. https://www.businesswire.com/news/home/20181112005379/en/Mellanox-InfiniBand-Ethernet-Solutions-Accelerate-Majority-TOP500
21. Gigabit Ethernet in supercomputer systems. https://www.mellanox.com/solutions/high-performance-computing/top500.php
22. NAS Parallel Benchmarks. https://www.nas.nasa.gov/publications/npb.html
23. *Chupakhin A., Kolosov A., Bahmurov A. et al.* Application of recommender systems approaches to the MPI program execution time prediction // IEEE Proceedings of 3rd International Conference "Modern Network Technologies-2020" (MoNeTec-2020). October 27–29, 2020, Moscow, Russia.
24. *Gibbons R.* A historical application profiler for use by parallel schedulers // Proceedings of the Job Scheduling Strategies for Parallel Processing. Springer, 1997.
25. *Kapadia N.H., Fortes J.A., Brodley C.E.* Predictive application performance modeling in a computational grid environment // Proceedings of the Eighth International Symposium on High Performance Distributed Computing. IEEE. 1999. P. 47–54.
26. *Li H., Groep D., Templon J., Wolters L.* Predicting job start times on clusters // CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid. 2004. P. 301–308.
27. *Mohr B., Wolf F.* Kojak – a tool set for automatic performance analysis of parallel programs // Euro-Par 2003 Parallel Processing. Springer, 2003. P. 1301–1304.
28. *Iverson M.A., Özgüner F., Potter L.* Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment // IEEE Trans. Comput. 1999. V. 48. № 12. P. 1374–1379.
29. *Liu X., Chen J., Liu K., Yang Y.* Forecasting duration intervals of scientific workflow activities based on time-series patterns // Proceedings of the IEEE Fourth International Conference on eScience. 2008. P. 23–30.
30. Ridge Regression. https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Ridge_Regression.pdf (accessed 21 Jun 2020).
31. Random forest algorithm. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm (accessed 21 Jun 2020).
32. *Smeliansky R.L.* Model of distributed computing system operation with // Programming and Computer Software. 2013. V. 39. № 5. P. 223–241.
33. https://www.coursera.org/specializations/recommender-systems
34. https://en.wikipedia.org/wiki/Recommender_system#:~:text=A%20recommender%20system%2C%20or%20a,would%20give%20to%20an%20item
35. Pearson's Correlation Coefficient. https://link.springer.com/referenceworkentry/10.1007%2F978-1-4020-5614-7_2569 (accessed 21 Jun 2020).
36. https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture
37. *Cheng C.M., Jin X.Q.* Matrix Decomposition // Encyclopedia of Social Network Analysis and Mining / Ed. by *R. Alhajj, J. Rokne*. NY: Springer, 2018.
38. *Stepanov E.P., Smeliansky R.L.* On bandwidth on demand problem // Proceedings of the 27th International Symposium Nuclear Electronics and Computing (NEC'2019). 30 September – 4 October 2019. CEUR-WS Budva, Montenegro. V. 2507. P. 402–407.
39. *Thomas B., Göring F., Harant J.* Menger's theorem // Journal of Graph Theory. 2001. V. 37. № 4. P. 35–36.
40. *Stepanov E., Smeliansky R.* On Analysis of Traffic Flow Demultiplexing Effectiveness // 2018 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTeC). IEEE, 2018.

41. *Kukreja N., Maier G., Alvizu R., Pattavina A.* SDN based automated testbed for evaluating multipath TCP // IEEE International Conference on Communication, ICC 2015. June 8–12, 2015, London, UK. Workshop Proceedings. 2016. P. 718–723.

42. *Raiciu C., Paasch C., Barre S. et al.* How hard can it be? Designing and implementing a deployable multipath ECP // Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI-12). April 25–27, 2012, San Jose, CA. USENIX. P. 399–412.

43. *Chemeritskiy E., Stepanov E., Smeliansky R.* Managing network resources with flow (de) multiplexing protocol // Mathematical and Computational Methods in Electrical Engineering. 2015. V. 53. P. 35–43.

44. Демонстрация приложения Bandwidth on Demand на базе контроллера RunOS. https://youtu.be/XjggLW1LKKg

45. *Burke J.* What is the role of machine learning in networking? https://searchnetworking.techtarget.com/answer/What-is-the-role-of-machine-learning-in-networking

46. *Smelyansky R.* Hierarchical edge computing // International conference proceedings "Modern Network Technologies", MoNeTec-2018. M., 2018. P. 97–105.