Организация эффективной многопользовательской работы гибридных вычислительных систем*

С. И. Смагин[1], А. А. Сорокин[1], С. И. Мальковский[1], С. П. Королёв[1,*], О. А. Лукьянова[1], О. Ю. Никитин[1], В. А. Кондрашев[2], В. Ю. Черных[1,3]

[1]Вычислительный центр ДВО РАН, Хабаровск, Россия

[2]ФИЦ "Информатика и управление" РАН, Москва, Россия

[3]Институт горного дела ДВО РАН, Хабаровск, Россия

*Контактный e-mail: serejk@febras.net

Исследуются вопросы организации многопользовательской работы гибридных вычислительных систем. На примере кластера Центра коллективного пользования "Центр данных ДВО РАН", построенного на архитектуре OpenPOWER, рассмотрены особенности функционирования систем подобного класса и предложены решения для организации их работы. С использованием механизма виртуальных узлов проведена адаптация системы диспетчеризации заданий PBS Professional, позволяющая организовать эффективное распределение аппаратных ресурсов кластера между пользовательскими задачами. Реализованное программное окружение кластера с системой комплексного планирования заданий рассчитано на работу с широким перечнем компьютерных приложений, включая программы, построенные с использованием различных технологий параллельного программирования. Для эффективного исполнения в данной среде решений на основе машинного обучения, глубокого обучения и искусственного интеллекта применены технологии виртуализации. С использованием возможностей среды контейнеризации Singularity сформирован специализированный стек программного обеспечения и реализован особый режим его работы в формате единой вычислительной цифровой платформы.

Ключевые слова: гибридный вычислительный кластер, сопроцессор, многопользовательский режим работы, архитектура компьютера, система управления заданиями, виртуализация, контейнер.

Библиографическая ссылка: Смагин С.И., Сорокин А.А., Мальковский С.И., Королёв С.П., Лукьянова О.А., Никитин О.Ю., Кондрашев В.А., Черных В.Ю. Организация эффективной многопользовательской работы гибридных вычислительных систем // Вычислительные технологии. 2019. Т. 24, № 5. С. 49–60.
DOI: 10.25743/ICT.2019.24.5.005.

Введение

В последнее время подходы к созданию высокопроизводительных вычислительных систем быстро меняются. Доминировавшие 5–7 лет назад "монолитные" системы, вычислительная мощность которых обеспечивалась большим количеством многоядерных

*Title translation and abstract in English can be found on page 60.

центральных процессоров, замещаются гибридными [1, 2]. В таких системах основная производительность на операциях с плавающей точкой обеспечивается разного рода ускорителями, а центральные процессоры играют скорее координирующую и интегрирующую роль. Подтверждением этого факта является то, что в 2019 г. (по данным на июнь 2019 г.) число гибридных суперкомпьютеров, входящих в список TOP500, увеличилось до 133, а в первой двадцатке этого рейтинга — до 10.

Среди различных типов ускорителей, применяемых в гибридных вычислительных системах, ключевое место занимают графические сопроцессоры (GPU), которые можно использовать не только для обработки компьютерной графики, как это было ранее, но и для выполнения неспециализированных вычислений. Каждый из них содержит тысячи сравнительно простых вычислительных ядер, способных одновременно выполнять одну задачу. Такая архитектура позволяет проводить определенные виды расчетов более эффективно по сравнению со стандартными средствами вычислений. Графические сопроцессоры обеспечивают в десятки раз более высокую производительность, чем традиционные процессоры, имея при этом относительно низкую стоимость в пересчете на FLOPS.

Одновременно с развитием рассматриваемой аппаратной базы начали активно развиваться и алгоритмы обработки данных, которые легли в основу новых технологий параллельного программирования, а также средств разработки, позволяющих максимально эффективно использовать ресурсы гибридных вычислительных систем. К ним можно отнести: технологии CUDA [3], OpenCL [4], OpenMP [5], компилятор PGI CUDA Fortran [6] и др. Это привело к расширению сфер применения GPU в решении различных вычислительных задач, среди которых наибольший прирост производительности получили проекты, связанные с машинным обучением (ML — Machine Learning), глубоким обучением (DL — Deep Learning) и системами искусственного интеллекта (AI — Artificial Intelligence) [7].

Экономика подобных проектов имеет кардинально иную структуру затрат. Ранее, на этапе разработки компьютерных систем, были необходимы сравнительно небольшие вычислительные ресурсы, требования к которым резко масштабировались при их промышленной эксплуатации и увеличении числа пользователей и/или задач. В случае информационных систем, созданных с использованием методов и средств ML/DL/AI, напротив, требуются большие аппаратные ресурсы при разработке и обучении модели и кратно меньшие объемы операционных затрат при эксплуатации уже готового компьютерного продукта. Это делает перечисленные технологии более привлекательными с экономической точки зрения в решении как исследовательских, так и прикладных задач. С другой стороны, необходимость наличия больших вычислительных мощностей повышает значимость суперкомпьютерных центров коллективного пользования, которые могут предоставить заинтересованным лицам требуемые ресурсы и оказать соответствующую квалифицированную поддержку.

Важно отметить, что функционирующие в России вычислительные центры лишь начинают переоснащаться гибридными системами либо используют вычислительные кластеры, имеющие только центральные процессоры. Поэтому применяемые в них программно-аппаратные комплексы не всегда позволяют эффективно проводить исследования с использованием технологий ML/DL/AI. Еще одной проблемой является обеспечение эффективной модели управления и совместного использования ресурсов гибридных вычислительных систем (CPU, GPU, RAM). Она связана с необходимостью адаптации системного и прикладного программного обеспечения, реализации методов и

алгоритмов диспетчеризации задач в рамках единой вычислительной цифровой платформы. Попытки ее решения предпринимались различными коллективами исследователей [8–10]. Их результаты направлены на исследование отдельных задач в этой области, однако комплексного и эффективного решения по созданию HPC-среды для приложений с применением GPU до сих пор не предложено.

Одним из направлений работ, требующих дополнительного исследования, является возможность выполнения задач ML/DL/AI в многопользовательской и многозадачной вычислительной среде. С одной стороны, это связано с тем, что до настоящего времени такие задачи решались в основном на персональных компьютерах или виртуальных машинах в интерактивном однопользовательском режиме, что несовместимо с концепцией центров коллективного пользования. С другой стороны, используемое в работе прикладное программное обеспечение представляет собой совокупность большого количества различных программных компонентов (интерпретатор языка Python, модули языка Python, фреймворки ML/DL/AI и др.) и библиотек, которые могут требовать окружение, отличное от функционирующего на узлах вычислительного кластера. Управление подобными разнородными наборами подсистем, запуск и контроль выполнения задач в рамках гибридной платформы представляют определенные сложности и требуют проведения отдельных научных изысканий.

В статье представлены подходы по организации эффективной работы гибридной высокопроизводительной вычислительной системы на примере кластера Центра коллективного пользования "Центр данных ДВО РАН" (далее — ЦКП) (`http://lits.ccfebras.ru`), построенного на архитектуре OpenPOWER. Рассмотрены вопросы адаптации системы диспетчеризации заданий кластера и некоторые аспекты использования технологий виртуализации для эффективного запуска приложений ML/DL/AI на гибридных вычислительных кластерах.

1. Выбор и адаптация системы управления ресурсами гибридной вычислительной среды

Высокопроизводительные вычислительные кластеры представляют собой группы компьютеров, объединенных высокоскоростными каналами связи (InfiniBand, Omni-Path и т. д.). При помощи специализированного программного обеспечения все компьютеры этой группы (или часть из них) способны работать над одной или несколькими пользовательскими задачами. Для обеспечения высокой вычислительной плотности в качестве отдельных компьютеров применяются многопроцессорные серверы общего назначения. В гибридных вычислительных кластерах эти серверы оснащаются сопроцессорами, которые выполняют бóльшую часть вычислительной работы.

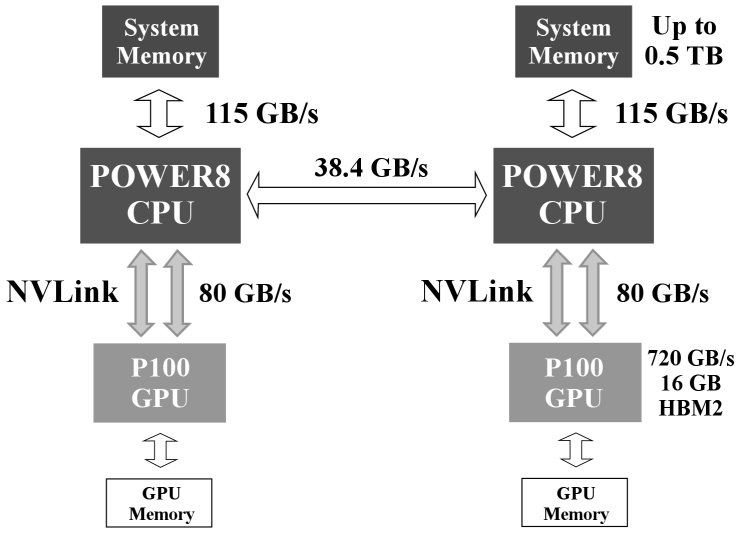
Наиболее широкое распространение среди архитектур построения многопроцессорных серверов общего назначения получила архитектура NUMA (Non-Uniform Memory Access, архитектура с неравномерным доступом к памяти) [11]. Это связано с тем, что она позволяет улучшить масштабируемость многопроцессорных вычислительных систем, оснастив каждый из процессоров своей локальной оперативной памятью. За счет этого значительно уменьшается число конфликтов обращения процессоров к памяти, что повышает общую производительность вычислительной системы. С другой стороны, это приводит к тому, что скорость доступа к оперативной памяти в системах с архитектурой NUMA зависит от ее расположения по отношению к процессору. Она максималь-

на при обращении процессора к своей локальной памяти и минимальна при обращении к памяти другого процессора.

Особенности NUMA-архитектуры серверов, из которых формируются гибридные вычислительные кластеры, накладывают дополнительные требования на используемую систему диспетчеризации заданий. Она должна не только обеспечивать эффективное распределение пользовательских задач между вычислительными узлами кластера и вести учет доступных ресурсов (в том числе и сопроцессоров), но и осуществлять дополнительную привязку процессов данной задачи к центральным процессорам и сопроцессорам в пределах отдельных вычислительных узлов. Это позволяет обеспечить максимальную производительность их подсистемы памяти.

Обоснование выбора системы диспетчеризации заданий и ее адаптацию для работы с ресурсами гибридных вычислительных систем рассмотрим на примере кластера ЦКП, построенного на архитектуре OpenPOWER [12]. Он состоит из одного управляющего и четырех вычислительных узлов. Все они имеют одинаковую конфигурацию, за исключением того, что узел управления оснащен контроллером Fibre Channel с пропускной способностью в 16 Гбит/с, применяемым для подключения к внешнему дисковому хранилищу. В качестве сети управления используется сеть, построенная по технологии Gigabit Ethernet, в качестве сети передачи данных — сеть EDR InfiniBand с пропускной способностью в 100 Гбит/с.

Управляющий и все вычислительные узлы представляют собой серверы Sitonica PW22LC (IBM Power Systems S822LC 8335-GTB). Они содержат по два процессора IBM POWER8 [13] с максимальной частотой 4.023 ГГц, два сопроцессора NVIDIA Tesla P100 GPU [14], 256 ГБ DDR4 ОЗУ, контроллер EDR InfiniBand и два жестких диска Seagate ST1000NX0313 1 ТБ 7200RPM. Эти серверы построены на архитектуре NUMA. Каждый из них включает в себя два NUMA-узла, состоящих из одного центрального процессора и его локальной оперативной памяти объемом 128 ГБ (см. рисунок). Доступ к оперативной памяти соседнего NUMA-узла осуществляется посредством шины X-Bus, соединяющей процессоры. К каждому центральному процессору по шине NVLink [15] подключен сопроцессор. В такой конфигурации каждый NUMA-узел сервера содержит по одному графическому сопроцессору.



Архитектура вычислительного узла

В качестве системы диспетчеризации заданий на гибридном вычислительном кластере выбрана открытая версия решения PBS Professional. Это связано как с большим опытом ее использования в ВЦ ДВО РАН, так и с возможностью расширения функциональности, позволяющей реализовать гибкое и эффективное распределение ресурсов гибридных вычислительных систем между пользовательскими задачами. Рассмотрим особенности конфигурации и адаптации PBS Professional.

Система диспетчеризации заданий позволяет распределять пользовательские задачи по всем пяти узлам вычислительного кластера. Для обеспечения тонкой привязки пользовательских задач к отдельным NUMA-узлам предложено использовать механизмы виртуальных узлов и "хуков". Первый из них позволяет представить каждый из вычислительных узлов в системе диспетчеризации заданий в виде двух виртуальных узлов, содержащих по одному процессору POWER8 (10 процессорных ядер или 80 аппаратных потоков), одному сопроцессору и 128 ГБ оперативной памяти. Для учета доступных сопроцессоров введен дополнительный ресурс ngpus, по одной единице которого (соответствует одному сопроцессору) назначается каждому виртуальному узлу. При запуске пользовательских заданий число доступных ngpus в вычислительной системе уменьшается на число единиц данного ресурса, заказанных пользователем. Такое решение позволяет избежать запуска нескольких пользовательских заданий на одном сопроцессоре.

Второй из используемых механизмов позволяет привязать различные действия (выполнение команд на узлах кластера, модификацию содержимого скриптов описания заданий и т. д.) к событиям, связанным с изменением статуса пользовательского задания (постановка задания в очередь, начало исполнения задания, завершение исполнения задания и т. д.). Для выделения пользовательским процессам, запускаемым на некотором NUMA-узле, подключенного к нему сопроцессора разработан специальный хук (скрипт, написанный на языке Python), выполняющийся при запуске пользовательского задания. Он выставляет значение переменной окружения CUDA_VISIBLE_DEVICES, равное номеру виртуального узла, выделенного этому заданию. Указанная информация используется драйвером GPU для определения списка доступных пользователю сопроцессоров. Например, при запуске пользовательского задания на нулевом виртуальном узле значение указанной переменной будет выставлено равным "0" (нумерация виртуальных узлов, NUMA-узлов и ускорителей начинается с нуля). После этого сам скрипт пользовательского задания запускается при помощи команды numactl, а параметры --cpunodebind и --membind выставляются равными номеру виртуального узла. Предложенный механизм позволяет изолировать пользовательские задания в рамках отдельных NUMA-узлов, что уменьшает число обращений к нелокальной оперативной памяти и повышает эффективность использования вычислительной системы, в том числе и сопроцессоров. Пример скрипта описания задания, выполняющегося на одном NUMA-узле с использованием одного ядра центрального процессора (8 аппаратных потоков) и одного сопроцессора, представлен в листинге 1.

При запуске пользовательских заданий, требующих для своего исполнения ресурсов более чем одного NUMA-узла, система управления выделяет для них необходимое число вычислительных узлов, которые не разделяются на виртуальные. Привязка пользовательских процессов к используемым ресурсам осуществляется при помощи штатных способов, реализованных в рамках отдельных технологий параллельного программирования (MPI, OpenMP и т. д.).

```
#PBS -k oe
#PBS -l select=1:ncpus=8:mem=4gb:ngpus=1
#PBS -l place=shared
#PBS -r n
#PBS -M user@mail.com
#PBS -m abe
#PBS -q workq
#PBS -N gpu_job
#!/bin/sh

cd /home/user/gpu_job && ./solver
```

Листинг 1. Пример скрипта описания задания, выполняющегося на одном NUMA-узле

Реализованное программное окружение кластера с системой диспетчеризации заданий рассчитано на работу с широким перечнем прикладного программного обеспечения, включая приложения, построенные с использованием технологий OpenMP и MPI. Для эффективного исполнения в данной среде приложений ML/DL/AI применены технологии виртуализации, описание которых представлено в следующем разделе статьи.

2. Использование технологий виртуализации для эффективного исполнения приложений ML/DL/AI на гибридных вычислительных кластерах

Технологии виртуализации находят широкое применение при организации высокопроизводительных вычислений. Наибольший интерес в данной области представляет виртуализация приложений, предназначенная для их выполнения в специальным образом сформированном окружении (системное программное обеспечение, библиотеки и т. д.), изолированном от окружения приложений, работающих непосредственно в операционной системе (ОС) вычислительной машины. С ее помощью решаются следующие основные задачи:

- выполнение на высокопроизводительных системах (ВС) программного обеспечения, предназначенного для работы на иных версиях ОС Linux;
- облегчение переносимости программ вместе с их окружением с одной ВС на другую;
- обеспечение воспроизводимости вычислительных экспериментов.

Особую актуальность виртуализация приложений приобрела при решении задач, связанных с ML/DL/AI на вычислительных кластерах. Это связано с тем, что используемое для их решения прикладное программное обеспечение представляет собой совокупность определенных версий нескольких компонентов: интерпретатора языка Python, фреймворка ML/DL/AI, вспомогательных модулей языка Python и различных специализированных библиотек. В связи с этим подготовка такого окружения очень часто становится невозможной в версии ОС Linux, установленной на узлах вычислительного кластера ввиду недоступности некоторых программных компонентов. Применение виртуализации приложений позволяет решить эту проблему. Рассмотрим основные сложности, возникающие при использовании данной технологии на вычислительных кластерах, а также возможные пути их решения.

Могут применяться различные технологии виртуализации приложений: программная, аппаратная и виртуализация на уровне ОС. С точки зрения минимизации накладных расходов наиболее перспективной при использовании в области высокопроизводительных вычислений является технология виртуализации на уровне ОС. При ее использовании ядро операционной системы поддерживает несколько изолированных экземпляров пространства пользователя, которые с точки зрения выполняющегося в них приложения полностью идентичны отдельной копии ОС. Эти экземпляры пространства пользователя называют контейнерами, а саму технологию — технологией контейнеризации. Для систем на базе UNIX и Linux технологию контейнеризации можно рассматривать как улучшенную реализацию механизма chroot.

Среди всех реализаций технологий контейнеризации, применяемых в настоящее время, можно выделить контейнеры Docker [16], Singularity [17] и Shifter [18]. Сформулируем основные требования, предъявляемые к рассматриваемым решениям при их использовании на вычислительных кластерах.

- Возможность интеграции решения в систему диспетчеризации заданий вычислительного кластера, предназначенную для распределения вычислительных ресурсов между пользователями ВС и учета их потребления пользовательскими заданиями.
- Отсутствие возможности эскалации привилегий со стороны произвольного пользовательского кода, выполняющегося в контейнере.
- Прямой доступ к оборудованию ВС (графические и иные сопроцессоры, хост-адаптеры высокоскоростной сети передачи данных и т. д.).
- Поддержка прозрачной работы с MPI-приложениями.
- Отсутствие необходимости в развертывании окружения, использующегося для работы с образами контейнеров.

В таблице приведены характеристики рассматриваемых технологий контейнеризации согласно [17] с дополнениями. Из нее следует, что Docker меньше всего удовлетворяет указанным требованиям. Основной причиной этого являются его архитектурные особенности, а именно то, что для управления контейнерами (запуск, остановка и т. д.) используется привилегированная системная служба. Это затрудняет реализацию работы с такими контейнерами из-под систем диспетчеризации заданий кластера, так как для их полноценного функционирования необходимо, чтобы контейнеры запускались как дочерние процессы пользователя. Еще один недостаток этого решения — возможность эскалации привилегий до уровня привилегий сервиса Docker со стороны пользовательского приложения, выполняющегося в контейнере. Работа запущенных в контейнере приложений с графическими сопроцессорами поддерживается, но эксплуатация такого решения затруднена необходимостью установки в контейнер драйвера сопроцессора, версия которого должна совпадать с версией драйвера, установленной в операционной системе ВС. Возможным решением данной проблемы может быть использование nvidia-docker, который позволяет задействовать драйвер операционной системы ВС.

В решении контейнеризации Shifter отсутствуют некоторые из недостатков, свойственных контейнерам Docker. Так, контейнеры Shifter запускаются с правами непривилегированного пользователя, поддерживается их интеграция с системами диспетчеризации заданий. Приложение, запущенное в таком контейнере, имеет прямой доступ к оборудованию ВС. Также при использовании этих контейнеров возможен запуск MPI-приложений. Единственный, но весомый недостаток Shifter — необходимость

Основные характеристики технологии контейнеризации

Характеристика	Singularity	Shifter	Docker
Модель предоставления привилегий	SUID/UserNS	SUID	Привилегированный сервис
Наличие привилегированных или доверенных сервисов	Нет	Нет	Да
Наличие инфраструктуры для работы с контейнерами	Нет	Да	Да
Дополнительные сетевые настройки	Нет	Нет	Да
Доступ к файловой системе хоста	Да	Да	Да, снижение безопасности
Нативная поддержка GPU	Да	Нет	Нет
Нативная поддержка InfiniBand	Да	Да	Да
Нативная поддержка MPI	Да	Да	Да
Поддержка основных систем диспетчеризации заданий	Да	Да	Нет
Разработано для научных расчетов	Да	Да	Нет
Контейнеры не модифицируются при использовании	Да	Нет	Да
Простая установка на HPC-кластерах	Да	Нет	Да
Ограничение доступной пользователю функциональности	Да	Да	Нет

развертывания инфраструктуры Docker, предназначенной для работы с образами контейнеров.

По нашему мнению, решением, наиболее полно удовлетворяющим всем обозначенным требованиям, является Singularity, которое изначально создавалось для работы на вычислительных кластерах. Контейнеры Singularity представляют собой отдельные исполняемые файлы, запускаемые пользователем либо из своей программной оболочки, либо через любую из существующих систем диспетчеризации заданий (поддерживается их полная интеграция, не требующая каких-либо дополнительных настроек). Порождаемые при этом процессы наследуют права пользователя, а потому отсутствует опасность эскалации привилегий. Приложения, запущенные в контейнере, имеют полный доступ к оборудованию ВС. Также ими полностью поддерживаются различные версии библиотеки MPI. Так как контейнеры Singularity распространяются в виде обычных файлов, отсутствует необходимость в настройке и поддержке какой-либо дополнительной вспомогательной управляющей программной инфраструктуры.

Использование Singularity на гибридном вычислительном кластере ЦКП позволило исключить проблемы, связанные с применением разных версий ОС, а также обеспечить централизованный контроль и управление ресурсами. Например, на кластере, работающем под управлением операционной системы CentOS версии 7, было установлено решение IBM PowerAI версии 4.0 (включающее набор фреймворков ML/DL/AI, оптимизированных для архитектуры кластера), требующее для своей работы операционную систему Ubuntu версии 16[1].

[1]Версии программ и требования к ним приведены на момент выполнения исследований.

Процесс развертывания перечисленного прикладного программного обеспечения заключался в его установке в контейнер Singularity, а также в создании для каждого из фреймворков специальных конфигурационных файлов (модулей) утилиты Environment Modules [19]. Контейнер сконфигурирован таким образом, чтобы при его запуске:

- анализировались переменные окружения, устанавливаемые предварительно загружаемым модулем;
- выполнялась активация соответствующего фреймворка, после чего управление передавалось интерпретатору языка Python, содержащемуся в контейнере.

Так как созданный контейнер является обычным исполняемым файлом, его запуск через систему диспетчеризации заданий ничем не отличается от запуска любой другой программы. В листинге 2 приведен пример скрипта описания задания, использующего фреймворк TensorFlow из состава IBM PowerAI. В данном случае исполняемый файл python-pai является файлом контейнера Singularity.

```
#!/bin/sh
#PBS -k oe
#PBS -l select=1:ncpus=16:ngpus=2:mem=10gb
#PBS -r n
#PBS -M user@mail.com
#PBS -m abe
#PBS -q workq
#PBS -N cifar10

# Load TensorFlow module
module add tensorflow/1.1.0
# Switch to working directory
cd /home/user/tf_working_dir
# Start neural network training
python-pai ./cifar10/cifar10_multi_gpu_train.py
# Analyze training results
python-pai ./cifar10/cifar10_eval.py
```

Листинг 2. Пример скрипта описания задания, использующего TensorFlow из состава PowerAI

Как видно, процесс использования программного обеспечения, установленного в контейнере, абсолютно прозрачен для пользователя, при этом соблюдается условие централизованного управления ресурсами кластера. Таким образом, можно сделать вывод, что система контейнеризации Singularity может успешно применяться для виртуализации приложений на гибридном вычислительном кластере с целью их выполнения в многопользовательской вычислительной среде.

Непрерывное совершенствование прикладных программных систем, их адаптация под разные виды вычислительных архитектур и ОС постепенно решает сформулированные выше системные проблемы. Однако заложенные в технологиях виртуализации возможности, безусловно, будут востребованы в дальнейшем при проведении расчетов, в которых в формате единой вычислительной среды необходимо реализовать особый стек программного обеспечения и специальный режим его работы.

Заключение

Появление современных вычислительных систем, построенных с использованием графических ускорителей, а также создание сопутствующей программной экосистемы позволило сделать существенный шаг вперед в решении многих ресурсоемких вычислительных задач. Однако многие вопросы эффективной эксплуатации гибридных вычислительных кластеров остаются до сих пор нерешенными.

Авторами на примере ресурсов ЦКП рассмотрены особенности функционирования подобного класса систем и предложены решения для организации их эффективной многопользовательской работы. В частности, сформулированы подходы по эффективному управлению ресурсами гибридной вычислительной среды, а также использованию в ее работе технологий виртуализации.

При проведении исследования использовано оборудование Центра коллективного пользования “Центр данных ДВО РАН” (ВЦ ДВО РАН, г. Хабаровск) [20].

Благодарности. Исследование выполнено при частичной финансовой поддержке РФФИ (грант № 18-29-03196). Адаптация отдельных прикладных программ и алгоритмов производилась в рамках гранта ДВО РАН № 18-5-100.

Список литературы / References

[1] **Strohmaier, E., Meuer, H.W., Dongarra, J., Simon, H.D.** The TOP500 List and Progress in High-Performance Computing // Computer. 2015. Vol. 48, No. 11. P. 42–49. DOI: 10.1109/MC.2015.338.

[2] **Keckler, S.W., Dally, W.J., Khailany, B., Garland, M., Glasco, D.** GPUs and the Future of Parallel Computing // IEEE Micro. 2011. Vol. 31, No. 5. P. 7–17. DOI: 10.1109/MM.2011.89.

[3] **Garland, M., Le Grand, S., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., Volkov, V.** Parallel computing experiences with CUDA // IEEE Micro. 2008. Vol. 28, No. 4. P. 13–27. DOI: 10.1109/MM.2008.57.

[4] **Stone, J.E., Gohara, D., Shi, G.** OpenCL: A parallel programming standard for heterogeneous computing systems // Computing in Science & Engineering. 2010. Vol. 12, No. 3. P. 66–72. DOI: 10.1109/MCSE.2010.69.

[5] **Liao, C., Yan, Y., de Supinski, B.R., Quinlan, D.J., Chapman, B.** Early experiences with the OpenMP accelerator model // Lecture Notes in Computer Science. 2013. Vol. 8122. P. 84–98. DOI: 10.1007/978-3-642-40698-0_7.

[6] **Ruetsch, G., Fatica, M.** CUDA Fortran for scientists and engineers: Best practices for efficient CUDA Fortran Programming (1st ed.). San Francisco: Morgan Kaufmann Publ., 2013. 338 p.

[7] **Steinkraus, D., Buck, I., Simard, P.Y.** Using GPUs for machine learning algorithms // Proc. of Eighth Intern. Conf. on Document Analysis and Recognition (ICDAR’05). Seoul, 2005. Vol. 2. P. 1115–1120. DOI: 10.1109/ICDAR.2005.251.

[8] **Reuther, A., Byun, C., Arcand, W. et al.** Scalable system scheduling for HPC and big data // J. of Parallel and Distributed Computing. 2018. Vol. 111. P. 76–92. DOI: 10.1016/j.jpdc.2017.06.009.

[9] **Sterling, T., Anderson, M., Brodowicz, M.** Chapter 5 — the essential resource management // High Performance Computing. 2018. P. 141–190. DOI: 10.1016/B978-0-12-420158-3.00005-8.

[10] **Quintero, D., de Souza Casali, D., Luis Cerdas Moya, E. et al.** IBM spectrum computing solutions. First Edition. USA: Intern. Business Machines Corporation, 2017. 214 p.

[11] **Lameter, C.** NUMA (Non-Uniform Memory Access): An overview // Queue. 2013. Vol 11, No. 7. P. 1–12. DOI: 10.1145/2508834.2513149.

[12] **Gschwind, M.** OpenPOWER: Reengineering a server ecosystem for large-scale data centers // Proc. of IEEE Hot Chips 26 Symp. (HCS). Cupertino, CA, USA, 2014. Accession Number: 16123547. DOI: 10.1109/HOTCHIPS.2014.7478829.

[13] **Sinharoy, B., Van Norstrand, J.A., Eickemeyer, R.J. et al.** IBM POWER8 processor core microarchitecture // IBM J. of Research and Development. 2015. Vol. 59, No. 1. P. 2:1–2:21. DOI: 10.1147/JRD.2014.2376112.

[14] **Foley, D., Danskin, J.** Ultra-performance Pascal GPU and NVLink interconnect // IEEE Micro. 2017. Vol. 37, No. 2. P. 7–17. DOI: 10.1109/MM.2017.37.

[15] **Appelhans, D., Walkup, B.** Leveraging NVLINK and asynchronous data transfer to scale beyond the memory capacity of GPUs // Proc. of the 8th Works. on Latest Advances in Scalable Algorithms for Large-Scale Systems. Denver, CO, USA, 2017. Article No. 5. DOI: 10.1145/3148226.3148232.

[16] **Merkel, D.** Docker: lightweight Linux containers for consistent development and deployment // Linux J. 2014. No. 239. P. 76–90.

[17] **Kurtzer, G.M., Sochat, V., Bauer, M.W.** Singularity: Scientific containers for mobility of compute // PLOS ONE. 2017. Vol. 12, No. 5. Accession Number: e0177459. DOI: 10.1371/journal.pone.0177459.

[18] **Gerhardt, L., Bhimji, W., Canon, S. et al.** Shifter: Containers for HPC // J. of Physics: Conf. Series. 2017. Vol. 898. P. 082021. DOI: 10.1088/1742-6596/898/8/082021.

[19] **Furlani, J.L.** Modules: Providing a flexible user environment // Proc. of the Fifth Large Installation Systems Administration Conf. (LISA V). 1991. P. 141–152.

[20] **Sorokin, A.A., Makogonov, S.V., Korolev, S.P.** The information infrastructure for collective scientific work in the Far East of Russia // Scientific and Technical Information Proc. 2017. Vol. 44, No. 4. P. 302–304. DOI: 10.3103/S0147688217040153.

Поступила в редакцию 10 июля 2019 г.

The organization of effective multi-user operation of hybrid computing systems

Smagin, Sergey I.[1], Sorokin, Aleksei A.[1], Malkovsky, Sergey I.[1], Korolev, Sergey P.[1,*], Lukyanova, Olga A.[1], Nikitin, Oleg Yu.[1], Kondrashev, Vadim A.[2], Chernykh, Vladimir Yu.[1,3]

[1]Computing Center FEB RAS, Khabarovsk, 680000, Russia
[2]FRC CSC RAS, Moscow, 119333, Russia
[3]Mining Institute FEB RAS, Khabarovsk, 680000, Russia

*Corresponding author: Korolev, Sergey P., e-mail: serejk@febras.net

Purpose. Improving the technology of machine learning, deep learning and artificial intelligence plays an important role in acquiring new knowledge, technological modernization and the digital economy development. An important factor of the development

in these areas is the availability of an appropriate high-performance computing infrastructure capable of providing the processing of large amounts of data. The creation of co-processor-based hybrid computing systems, as well as new parallel programming technologies and application development tools allows partial solving this problem. However, many issues of organizing the effective multi-user operation of this class of systems require a separate study. The current paper addresses research in this area.

Methodology. Using the OpenPOWER architecture-based cluster in the Shared Services Center "The Data Center of the Far Eastern Branch of the Russian Academy of Sciences", the features of the functioning of hybrid computing systems are considered and solutions are proposed for organizing their work in a multi-user mode. Based on the virtual nodes concept, an adaptation of the PBS Professional job scheduling system was carried out, which provides an efficient allocation of cluster hardware resources among user tasks. Application virtualization technology was used for effective execution of machine learning and deep learning problems.

Findings. The implemented cluster software environment with the integrated task scheduling system is designed to work with a wide range of computer applications, including programs built using parallel programming technologies. The virtualization technologies were used in this environment for effective execution of the software, based on machine learning, deep learning and artificial intelligence. Having the capabilities of the container Singularity, a specialized software stack and its operation mode was implemented for execution machine learning, deep learning and artificial intelligence tasks on a unified computing digital platform.

Originality. The features of hybrid computing platforms functioning are considered, and the approach for their effective multi-user work mode is proposed. An effective resource manage model is developed, based on the virtualization technology usage.

Keywords: hybrid computing cluster, coprocessor, multi-user access mode, computer architecture, job scheduling system, workload manager, virtualization, container.

Cite: Smagin, S.I., Sorokin, A.A., Malkovsky, S.I., Korolev, S.P., Lukyanova, O.A., Nikitin, O.Yu., Kondrashev, V.A., Chernykh, V.Yu. The organization of effective multi-user operation of hybrid computing systems // Computational Technologies. 2019. Vol. 24, No. 5. P. 49–60. (In Russ.) DOI: 10.25743/ICT.2019.24.5.005.

Acknowledgements. The reported study was funded by RFBR according to the research project No. 18-29-03196. The adaptation of individual applications and algorithms is supported by FEB RAS (grant No. 18-5-100).

Received July 10, 2019