ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ВЫЧИСЛЕНИЯ В НАУКЕ, ТЕХНИКЕ И ОБРАЗОВАНИИ

УДК 004.4 : 378.1

СУПЕРКОМПЬЮТЕРНАЯ СИСТЕМА КАК ФАКТОР ВЛИЯНИЯ НА ОБРАЗОВАТЕЛЬНЫЙ ПРОЦЕСС

С. Д. Кургалин, С. В. Борзунов

Воронежский государственный университет

Аннотация. В статье рассмотрены отличительные черты организации учебного процесса в вузе, в котором установлена высокопроизводительная компьютерная система. Будущие специалисты в области информационных технологий должны обладать компетенциями, связанными с применением высокопроизводительных вычислений для решения разнообразных практических задач. Указано на важность создания специализированных учебных и учебно-методических комплексов для методической поддержки учебных дисциплин, связанных с освоением суперкомпьютерных технологий.
Ключевые слова: суперкомпьютер, высокопроизводительная компьютерная система, информационные технологии, образовательный процесс, учебное пособие.

Введение

В настоящее время отличительной чертой организации учебного процесса в вузах является необходимость поддержания высокого уровня цифровых технологий и информационно-телекоммуникационных возможностей. В области компьютерных наук непрерывные количественные и, в ряде случаев, качественные изменения характерных параметров программно-аппаратных комплексов, используемых при решении задач математического моделирования, прогнозирования в различных сферах деятельности (промышленной, экономической, научной и других), обработки больших массивов данных, происходят в течение относительно коротких временных интервалов, соизмеримых с периодом подготовки выпускника бакалавриата, специалитета или магистратуры. Будущие специалисты в области информационных технологий должны обладать компетенциями, связанными с применением высокопроизводительных вычислений для решения разнообразных практических задач. Эти компетенции вырабатываются в вузах в процессе их обучения по программам общих и специальных курсов. Достижению указанных компетенций способствует наличие доступной аппаратной базы и программных средств, которые должны отвечать вполне определенным стандартам производительности. Наличие в крупном университете высокопроизводительной вычислительной системы — суперкомпьютера — постепенно становится непременным условием организации как современного образовательного процесса, так и научных исследований фундаментального и прикладного характера.

1. Основная часть

Суперкомпьютерные технологии сегодня позволяют решать многие фундаментальные и прикладные проблемы, моделирование и анализ которых требуют проведения масштабных вычислений. Значимость развития и практического использования суперкомпьютерных технологий ставит перед системой высшего образования важную задачу оперативной подготовки кадров в области таких технологий, отсутствие достаточного числа высококвалифицированных кадров в этой области становится серьезной проблемой. С целью разработки и обеспечения комплекса мероприятий, направленных на эффективное использование потенциала высшей школы для развития и внедрения суперкомпьютерных технологий в образование, науку

и промышленность создан Суперкомпьютерный консорциум университетов России. Однако в настоящее время сложилась такая ситуация, когда высококвалифицированные специалисты в области суперкомпьютерных технологий сосредоточены, в основном, в научно-исследовательских институтах и финансовом секторе. Как правило, такие специалисты ориентированы на работу с достаточно узкими классами вычислительных архитектур и прикладных задач. Вместе с тем, требуется массовая подготовка новых кадров высшей квалификации — универсальных специалистов для науки, промышленности и бизнеса, что возможно реализовать только посредством внедрения в вузах оперативно обновляемых систем и методик подготовки современных ИТ-специалистов. Суперкомпьютерные центры в вузах представляют собой, фактически, новый тип учебно-научных комплексов для обеспечения образовательного процесса и научных исследований.

Повышение эффективность работы высокопроизводительных компьютерных систем достигается за счет различных средств, среди которых можно выделить совершенствование архитектуры процессоров, применение графических ускорителей, а также использование ресурсов параллелизма — свойства вычислительных систем, при котором команды выполняются одновременно и при этом взаимодействуют друг с другом [1]. Следует отметить, что в подавляющем большинстве современных вычислительных систем заложены те или иные возможности для параллельного исполнения команд. Это достигается либо наличием некоторого числа ядер, способных обрабатывать несколько параллельных потоков данных, либо возможностью использовать целый набор одновременно работающих центральных процессоров. Создание программных продуктов для такого рода систем имеет свои особенности, и это значительно отличает процесс подготовки параллельной программы от ее последовательной версии. Идеи параллельной обработки данных активно используются при решении проблем в различных прикладных областях [2] и при моделировании разнообразных процессов [3–5], их начинают внедрять и в практические приложения теоретических курсов [6, 7].

Среди вопросов, требующих повышенного внимания со стороны разработчика, выделим методы координации исполняемых процессов, межпроцессорного обмена данными, распределения памяти и планирования операций. Важнейшее значение имеет организация сети обмена данными, оптимизированная в рамках минимизации времени отклика и увеличения пропускной способности передачи данных во время исполнения параллельной программы. Объединение нескольких вычислительных узлов в единую высокопроизводительную вычислительную систему требует междисциплинарного подхода, затрагивающего как методы и технологии из области архитектур процессоров, линий связи, так и программных решений для обеспечения согласованного взаимодействия, направленного на корректное и быстрое решение ресурсоемких задач.

Для методической поддержки учебных курсов, связанных с освоением высокопроизводительных компьютерных технологий, безусловно, нужно создавать специализированные учебные и учебно-методические комплексы, включающие циклы учебных пособий, в которых должны быть представлены особенности архитектуры суперкомпьютеров, применяемых в них операционных систем и других программных средств, языков программирования, принципов параллельной обработки данных, способов оценки эффективности алгоритмов и программ и т. д. На кафедре цифровых технологий Воронежского государственного университета (ВГУ) такой цикл учебных пособий был сформирован [9–13]. Он создавался сотрудниками кафедры на основе многолетнего опыта преподавания на факультете компьютерных наук (ФКН) указанных технологий, а также в процессе их применения в научных исследованиях [14]. Их отличает ориентация на практическое использование получаемых студентами знаний и на формирование необходимых компетенций для решения реальных вычислительных задач.

Указанный опыт формировался, начиная с 2002 года, когда на кафедре цифровых технологий был установлен первый в регионе суперкомпьютер [15, 16]. Он представлял собой одну из

лучших в то время высокопроизводительных кластерных систем (рис. 1) и занимал 26-е место в списке Top-50 России и стран СНГ, превосходя по мощности соответствующие центры во многих крупных городах России (Екатеринбург, Пермь, Самара, Томск, Казань, Владивосток и др.) (рис. 2).

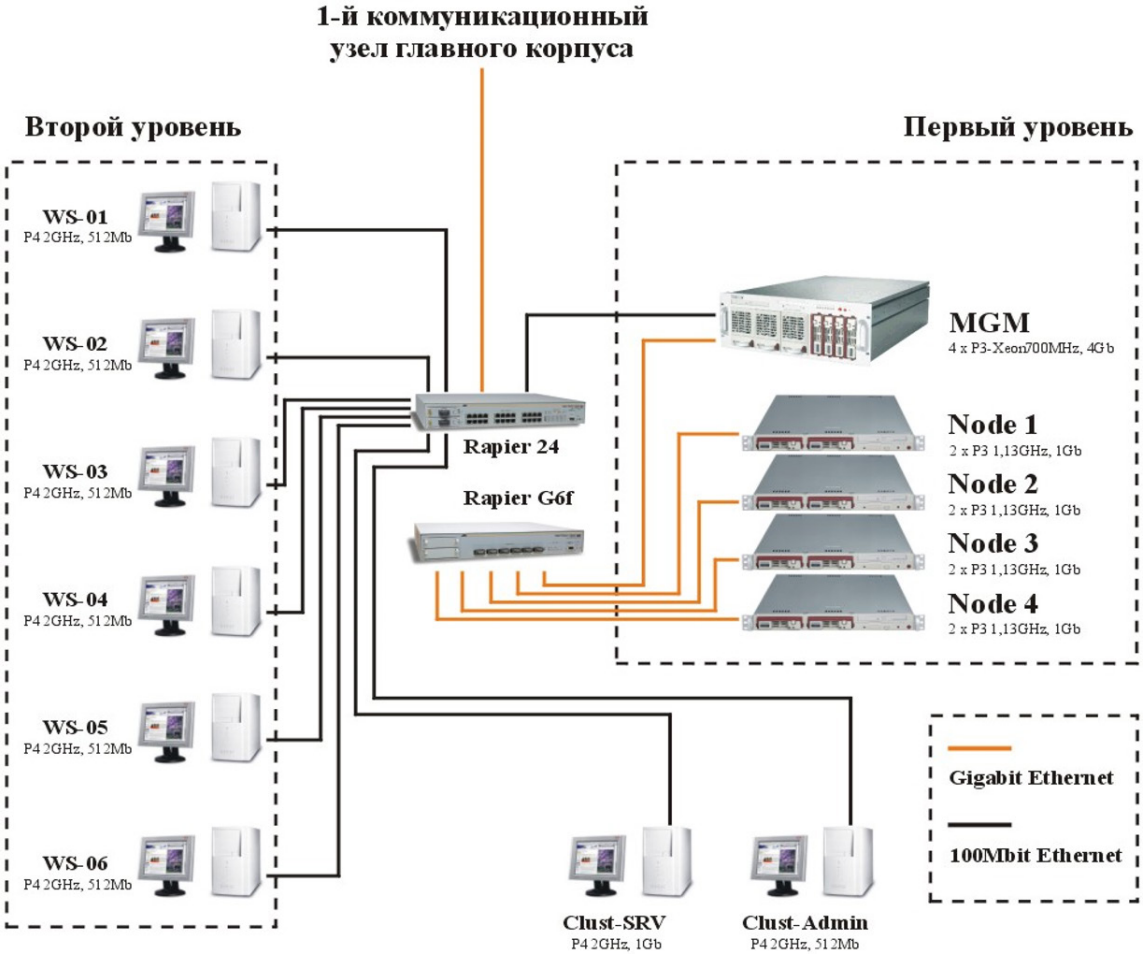

Рис. 1. Схема архитектуры первого высокопроизводительного компьютерного кластера в ВГУ

В настоящее время в Суперкомпьютерном центре факультета компьютерных наук работает суперкомпьютер с пиковой производительностью 39 Tflops (рис. 3), заменивший в 2016 году предшествующую параллельную компьютерную систему меньшей производительности. Он состоит из 10 узлов, каждый из которых имеет по два 12-ядерных процессора (всего 240 процессорных ядер), в каждом по 128 Гбайт оперативной памяти и по SSD диску в 256 Гбайт. Узлы содержат энергоэффективные математические сопроцессоры: семь узлов — по 2 ускорителя Intel Xeon Phi, а три остальных – по 2 ускорителя Nvidia Tesla. Так, сопроцессор Nvidia Tesla A100 включает в себя 6912 потоковых ядер CUDA, а Intel Xeon Phi позволяет производить вычисления с использованием до 240 логических ядер. Все узлы объединены высокоскоростной сетью InfiniBand 40 Гбит/с. Кроме того, в его состав входит управляющий узел, который также является и сервером для хранения файлов. Последний имеет два 6-ядерных процессора, 64 Гбайт оперативной памяти и дисковую подсистему объемом 48 Тбайт.

На базе этого суперкомпьютера организован универсальный учебно-научный комплекс, в котором ведутся учебные занятия, осуществляется научная работа, разрабатывается программное обеспечение для проведения исследований, формируются новые подходы к организации учебного процесса, создаются различные методы и средства, обогащающие содержание и методическое обеспечение учебных дисциплин.

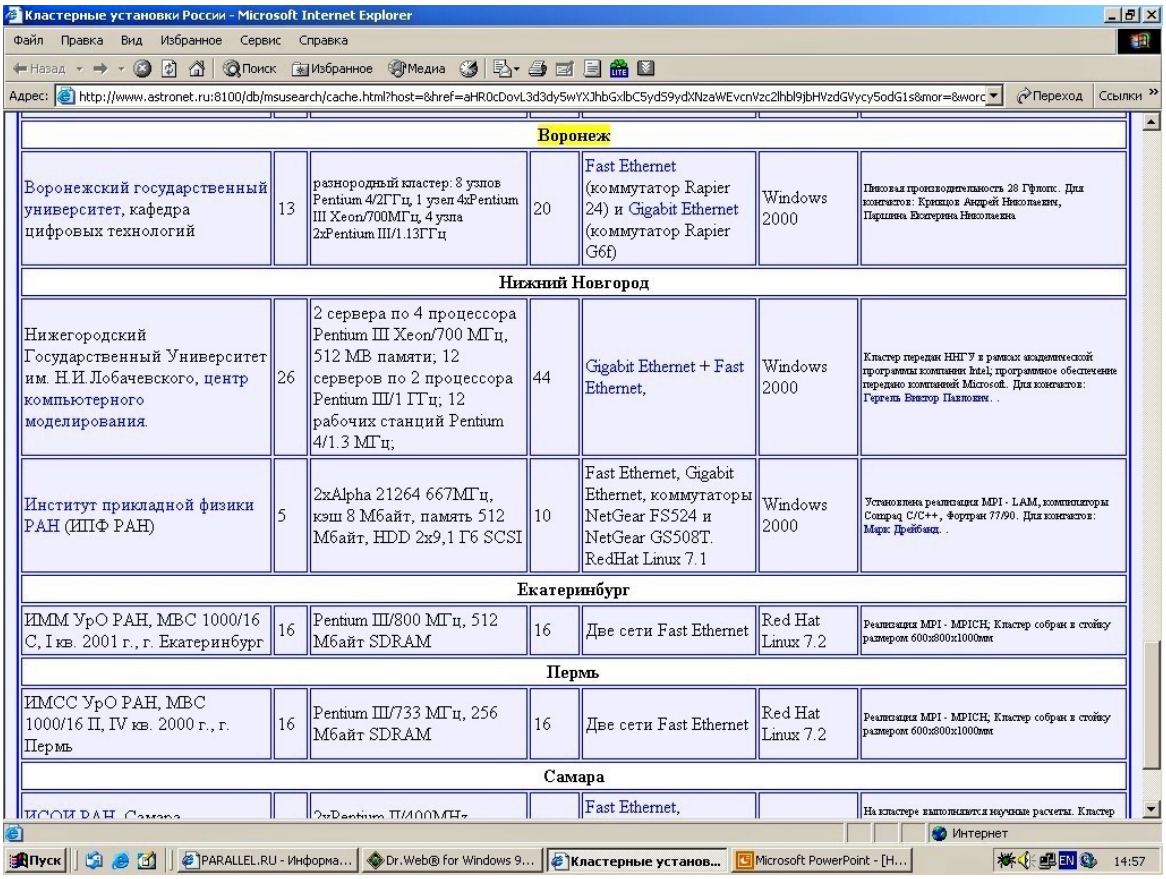

Рис. 2. Высокопроизводительная кластерная система кафедры цифровых технологий в списке Top-50 России и стран СНГ

Рис. 3. Внешний вид суперкомпьютера (слева — общий план, справа — детальный вид передней панели)

Заключение

В настоящее время становится очевидным взаимодействие и взаимовлияние установленных в вузах высокопроизводительных компьютерных систем (суперкомпьютеров) и осущест-

вляемого учебного процесса. С одной стороны, суперкомпьютеры позволяют перейти к преподаванию учебных дисциплин, находящихся на переднем крае компьютерных наук, обучать перспективным информационным технологиям, которые, как ожидается, будут все более востребованы на рынке труда. С другой стороны, учебный процесс предъявляет свои требования к высокопроизводительным компьютерным системам, заставляя их непрерывно совершенствоваться, обновлять программные средства и устанавливать те их них, которые дадут необходимые компетенции выпускаемым специалистам, повышать квалификацию персонала суперкомпьютерных центров. Это дает возможность перейти на новый уровень педагогической и научной деятельности с применением обновленных методик для повышения интереса студентов к преподаваемым дисциплинам, повысить процент усвоения учебного предмета и стимулировать его углубленное изучение.

Необходимо максимально развивать и использовать эти возможности.

Литература

1. *Таненбаум Э.* Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стеен. – Санкт-Петербург : Питер, 2003. – 877 с.

2. *Туровский Я. А.* Выбор анализирующих вейвлетов для системы с параллельной обработкой биомедицинских данных / Я. А. Туровский, С. Д. Кургалин, А. В. Максимов // Вестник Воронежского гос. ун-та. Серия: Системный анализ и информационные технологии. – 2011. – № 2. – С. 74–79.

3. *Туровский Я. А.* Моделирование процесса выделения частотных локальных максимумов в сигналах электроэнцефалограмм / Я. А. Туровский, С. Д. Кургалин, А. В. Максимов // Вестник Тамбовского гос. технического ун-та. – 2012. – Т. 18, № 4. – С. 827–833.

4. *Туровский Я. А.* Моделирование нейрокомпьютерного интерфейса на основе активностной парадигмы / Я. А. Туровский, С. Д. Кургалин, А. В. Максимов // Системы управления и информационные технологии. – 2012. – № 1(47). – С. 99–103.

5. *Туровский Я. А.* Моделирование выделения и анализа цепочек локальных максимумов вейвлет-спектров на примере сигналов с известными свойствами / Я. А. Туровский, С. Д. Кургалин, А. Г. Семенов // Системы управления и информационные технологии. – 2013. – № 2(52). – С. 24–29.

6. *Kurgalin S.* The Discrete Math Workbook: A Companion Manual for Practical Study / S. Kurgalin, S. Borzunov. – Springer International Publishing, 2018. – 485 p. – (Series: Texts in Computer Science).

7. *Kurgalin S.* Algebra and Geometry with Python / S. Kurgalin, S. Borzunov. – Springer, 2021. – xvi, 425 p.

8. *Борзунов С. В.* Языки программирования. Python: решение сложных задач : учебное пособие для высшего образования / С. В. Борзунов, С. Д. Кургалин. – Санкт-Петербург : Лань, 2023. – 192 с.

9. *Kurgalin S.* A Practical Approach to High-Performance Computing / S. Kurgalin, S. Borzunov. – Springer, 2019. – xi, 206 p.

10. *Борзунов С. В.* Практикум по параллельному программированию / С. В. Борзунов, С. Д. Кургалин, А. В. Флегель. – Санкт-Петербург : БХВ, 2017. – 236 с. – (Учеб. литература для вузов).

11. *Борзунов С. В.* Суперкомпьютерные вычисления : практический подход / С. В. Борзунов, С. Д. Кургалин. – Санкт-Петербург : БХВ, 2019. – 256 с. – (Учеб. литература для вузов).

12. *Борзунов С. В.* Параллельное программирование: задачи и решения : учебное пособие / С. В. Борзунов, С. Д. Кургалин. – Воронеж : Издательский дом ВГУ, 2018. – 113 с.

13. *Борзунов С. В.* Практикум по параллельному программированию : учебное пособие / С. В. Борзунов, С. Д. Кургалин, М. В. Куцов. – Воронеж : Издательский дом ВГУ, 2016. – 79 с.

14. *Кургалин С. Д.* Using the resources of the Supercomputer Center of Voronezh State University in learning processes and scientific researches / С. Д. Кургалин, С. В. Борзунов // Труды международной конференции «Суперкомпьютерные дни в России». – Москва : Изд-во МГУ, 2018. – С. 972–977.

15. *Кургалин С. Д.* Высокопроизводительный вычислительный кластер Воронежского госуниверситета / С. Д. Кургалин, В. С. Александров, С. В. Побежимов // Высокопроизводительные параллельные вычисления на кластерных системах : материалы 2-го Международного научно-практического семинара. – Нижний Новгород, 2002. – С. 174–176.

16. *Кургалин С. Д.* Суперкомпьютерные технологии в Воронежском государственном университете / С. Д. Кургалин, С. В. Борзунов // Вестник Воронежского гос. ун-та. Серия: Проблемы высшего образования. – 2018. – № 3. – С. 183–187.

УДК 004.023

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ПЛАТФОРМ ДЛЯ ХРАНЕНИЯ И ОБРАБОТКИ ДАННЫХ DATABRICKS И SNOWFLAKE

В. В. Нестругина

Воронежский государственный университет

Аннотация. Рассматриваются критические различия между хранилищами данных и озерами данных, между платформами Databricks и Snowflake.
Ключевые слова: Databricks, Snowflake, хранилище данных, озеро данных.

Введение

Традиционно корпорации использовали хранилища данных для хранения данных различных типов, генерируемых из разных источников. Хранилища данных предназначены для поддержки принятия решений с помощью аналитики, извлеченной из данных. Однако с развитием технологий потребности в данных изменились из-за увеличения скорости, объема и достоверности данных. На рынке довольно много соответствующих систем, но мы обсудим две из них: Databricks и Snowflake.

Databricks стремится объединить хранилища данных и озера данных в рамках единой платформы, в то время как Snowflake продвигает хранилище данных с предложением программного обеспечения как услуги (SaaS), которое отличается меньшими затратами на обслуживание и масштабируемостью.

В данной статье будут рассмотрены критические различия между хранилищами данных и озерами данных, между Databricks и Snowflake, а также альтернативы этим технологиям.

1. Сравнение хранилища данных и озера данных

Хранилище данных хранит исторические данные о бизнесе, что позволяет анализировать и извлекать ценную информацию. Оно не хранит текущую информацию и не обновляется в режиме реального времени и в основном используется с реляционными базами данных, где данные хранятся в отношениях (таблицах) со схемой, оптимизированной для быстрого запроса и анализа.

Хранилище данных хранит данные в структурированном формате, доступном через запросы SQL. Структурированными данными легче манипулировать, чем неструктурированными или полуструктурированными данными, поскольку схема всех данных известна заранее [3].

В отличие от хранилищ данных, озера данных могут содержать неструктурированные данные. [4] Озера данных позволяют хранить данные в различных форматах в облачном объектном хранилище (S3, ADLS или облачном хранилище Google) с отдельным уровнем обработки. Они дополняют ограничения хранилищ данных. Хранение и обработка децентрализованы, и вместо использования больших таблиц данные разбиваются на более мелкие файлы и распределяются по нескольким узлам. Разделенное хранилище позволяет озеру данных масштабироваться независимо [2].

2. Databricks

Databricks — это облачная платформа, специализирующаяся на анализе данных в любом масштабе, независимо от их местоположения. Это платформа данных и аналитики, которая

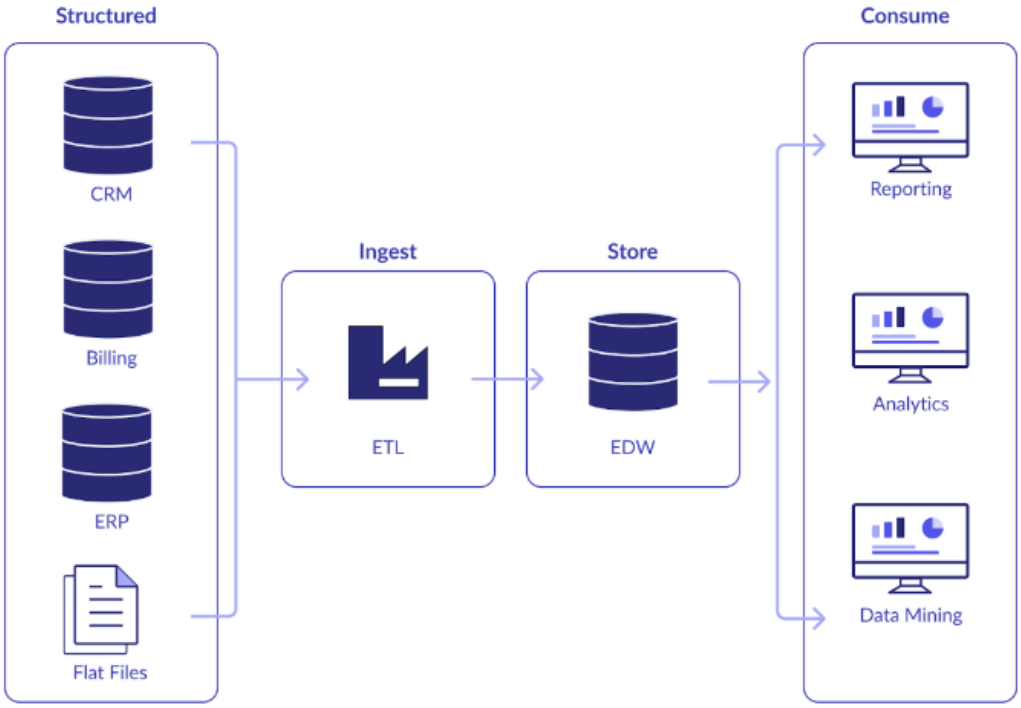

Рис. 1. Архитектура хранилища данных [1]

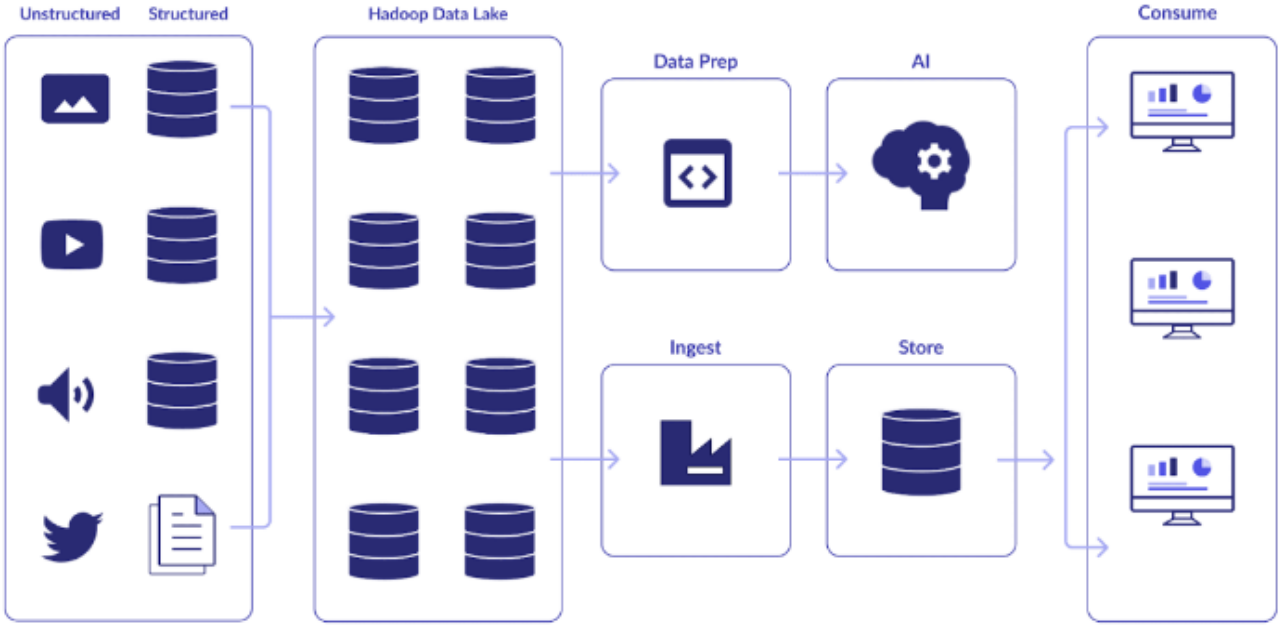

Рис. 2. Архитектура озера данных [1]

помогает предприятиям извлекать бизнес-аналитику из данных. Он также предоставляет полноценное рабочее пространство для анализа данных со средой выполнения машинного обучения, управляемым MLflow и блокнотами для совместной работы.

Databricks хорошо известен своей способностью обрабатывать большие объемы данных. Кроме того, он поддерживает несколько языков, что делает его более эффективным, поскольку вы можете интегрировать библиотеки из любой экосистемы языков программирования. В результате крупные предприятия часто используют Databricks для производственных операций в таких отраслях, как здравоохранение, финансовые технологии, развлечения и других [2].

3. Snowflake

Snowflake — это облачное хранилище данных, которое обеспечивает все функции хранилища данных с помощью одного инструмента без различных системных интеграций. Начать работу с ним относительно легко, он достаточно рентабелен и быстро масштабируется по сравнению с устаревшим хранилищем данных. Разделенное хранилище и вычисления обеспечивают совместное использование и масштабирование данных, а Snowflake абстрагирует сложности облака и позволяет клиентам загружать, интегрировать, обрабатывать, анализировать и делиться своими данными [2].

В следующей таблице приведены общие различия между Databricks и Snowflake.

Таблица

Сравнение Databricks и Snowflake

	Databricks	Snowflake
Модель обслуживания	PaaS	SaaS
Поддержка основных облачных платформ	Azure, AWS, Google	Azure, AWS, Google
Масштабируемость	Автомасштабирование	Автомасштабирование до 128 нод
Привязка к поставщику	Нет	Да
Удобство использования	Кривая обучения	Прост в освоении
Структуры данных	Все типы данных (неструктурированные, аудио, видео, логи, текст, и т.д.)	Полуструктурированные и структурированные данные
Сервисы	Большие данные, data science, анализ данных, машинное обучение	Управление базами данных и хранилищем данных
Машинное обучение	Встроенный и унифицированный инструмент для любого типа разработки	Доступно только через сторонние интеграции
Стоимость	Оплата по факту использования	Оплата по факту использования
Интерфейс запросов	SQL, Spark DataFrame, Koalas	SQL
Оптимизация запросов	Векторизация и стоимостная оптимизация	Векторизация и стоимостная оптимизация
Различные типы нод	Да	Нет

4. Архитектура Databricks и Snowflake

Архитектура озер данных отличает их от обычных хранилищ данных разделением хранения и вычислений. Databricks имеет отдельный уровень для хранения и вычислений.

Хотя Snowflake является управляемым сервисом и его архитектура прозрачна для конечных пользователей, он также имеет отдельный уровень хранения и обработки.

Snowflake вдохновлен устаревшей архитектурой хранилищ данных, но модернизирован. Внутри он имеет разделение хранения и обработки и может масштабироваться независимо, сохраняя при этом оба уровня.

Напротив, Databricks полностью отделил уровни хранения и обработки. Он позволяет хранить данные где угодно, в любом формате и форме. Он фокусируется на уровне обработки и предлагает свободу выбора механизма обработки при интеграции сторонних решений [2].

5. Структуры данных

Snowflake поддерживает полуструктурированные и структурированные данные. Данные можно загружать непосредственно в Snowflake, минуя процесс ETL.

Однако Databricks позволяет хранить все типы данных в любом формате и типе, поскольку его уровень хранения не зависит от уровня обработки. Databricks может работать как инструмент ETL для добавления структуры к неструктурированным данным [2].

6. Масштабируемость

Обе платформы используют облачные вычисления для масштабирования без значительных накладных расходов. Databricks можно масштабировать настолько, насколько вы можете инвестировать в инфраструктуру, но Snowflake ограничен 128 нодами. Кроме того, Snowflake предлагает варианты хранилищ фиксированного размера, при которых конечный пользователь не может изменять размер нодов, но может изменять размер кластеров. Кроме того, Snowflake предлагает автоматическое масштабирование и автоматическую приостановку, что позволяет запускать и останавливать кластеры во время простоя или занятости. Databricks позволяет предоставлять различные типы узлов на разных уровнях масштабирования, это немного сложнее и потребует необходимых технических знаний для масштабирования кластера Databricks [2].

7. Производительность

Snowflake подходит для высокопроизводительных запросов, поскольку у него уже есть структурированные данные, подходящие для сценария использования в бизнесе. И Snowflake, и Databricks реализуют стоимостные оптимизацию и векторизацию. Кроме того, Databricks предлагает интеграцию хеша для ускорения агрегации запросов.

Snowflake имеет низкую производительность при работе с полуструктурированными данными, поскольку ему может потребоваться загрузить все данные в оперативную память и выполнить полное сканирование. Databricks, напротив, позволяет оптимизировать задания по обработке данных для выполнения высокопроизводительных запросов.

8. Машинное обучение

Snowflake не имеет библиотек машинного обучения, но предлагает коннекторы для интеграции различных инструментов машинного обучения. Кроме того, он предоставляет доступ к своему уровню хранения или результатам экспортных запросов, которые можно использовать для обучения и тестирования моделей.

Пользователь имеет возможность разрабатывать модели машинного обучения и проводить аналитику с помощью Snowflake, но это требует интеграции с другими решениями из-за отсутствия таких услуг «из коробки». Вместо этого Snowflake предоставляет драйверы для интеграции с другими библиотеками или модулями платформы и доступа к данным [5].

9. Варианты использования Databricks и Snowflake

Snowflake подходит для бизнес-аналитики на основе SQL благодаря своему дизайну и архитектуре. Databricks тоже подходит для бизнес-аналитики на основе SQL, а также предлагает поддержку более широкого спектра вариантов использования, таких как механизмы рекомендаций или обнаружение вторжений. Оба продукта поддерживают информационные панели для отчетности и аналитики.

Databricks можно масштабировать для удовлетворения высоких требований к пропускной способности любой системы большого объема, но производительность запросов для аналитики будет низкой[6]. С другой стороны, Snowflake имеет ограниченную поддержку непрерывной записи и параллелизма, но может превзойти производительность Databricks [6].

10. Различия между Databricks и Snowflake

Snowflake — это полностью управляемый сервис, поэтому его развертывание и масштабирование упрощается. Большая часть операций скрыта от конечного пользователя, поэтому возможностей для тонкой настройки немного. С другой стороны, Databricks требует гораздо большего администрирования и развертывания; для оптимизации запросов, выполняемых в механизме озера данных, требуется опыт.

Обе платформы зависят от облачных платформ из-за их уровней хранения. Однако им угрожают провайдеры облачных услуг, которые предлагают аналогичные платформы и имеют лучшую интеграцию в свои экосистемы [2].

Заключение

Хотя Databricks и Snowflake имеют некоторые общие черты, один из них ориентирован на озера данных, а другой — на хранилища данных. Snowflake лучше всего подходит для SQL-подобных приложений бизнес-аналитики и обеспечивает более высокую производительность. С другой стороны, Databricks предлагает поддержку нескольких языков программирования. Snowflake также немного проще использовать для разработчиков по сравнению с Databricks, который требует длительного обучения. Из-за различных компромиссов многим компаниям приходится использовать их вместе: Databricks для ETL и Snowflake для хранилища данных.

Разработчику крайне важно иметь глубокое понимание технологического ландшафта, независимо от того, создает ли он приложения или оценивает отраслевые решения. Достигнув полного понимания этих технологических решений, он сможете принимать обоснованные решения и эффективно использовать подходящее решение для удовлетворения конкретных требований к SaaS, озерам данных или хранилищам данных.

Литература

1. *McCalley E.* Snowflake vs Databricks: Where Should You Put Your Data? / E. McCalley // Datagrom : [сайт]. – 2023. – URL: https://www.datagrom.com/data-science-machine-learning-ai-blog/snowflake-vs-databricks (дата обращения: 01.11.2023).

2. Databricks vs Snowflake // Macrometa : [сайт]. – 2023. – URL: https://www.macrometa.com/event-stream-processing/databricks-vs-snowflake (дата обращения: 01.11.2023).

3. What is the difference between structured and unstructured data? // Macrometa : [сайт]. – 2023. – URL: https://www.macrometa.com/articles/structured-vs-unstructured-data (дата обращения: 01.11.2023).

4. What is a Data Lake? // Macrometa : [сайт]. – 2023. – URL: https://www.macrometa.com/articles/what-is-data-lake (дата обращения: 01.11.2023).

5. Machine learning and AI // Macrometa : [сайт]. – 2023. – URL: https://www.macrometa.com/topics/machine-learning-ai (дата обращения: 01.11.2023).

6. *Phaujdar A.* Databricks vs Snowflake: 9 critical differences / A. Phaujdar // Hevodata : [сайт]. – 2023. – URL: https://hevodata.com/learn/databricks-vs-snowflake/#d3 (дата обращения: 01.11.2023).

УДК 378.1 : 681.3

ИСПОЛЬЗОВАНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON В РАЗРАБОТКЕ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ МАТЕМАТИЧЕСКИХ КОДОВ

А. В. Романов, С. Д. Кургалин, К. О. Петрищев

Воронежский государственный университет

Аннотация. Представлен опыт использования языка программирования Python в решении задач высокопроизводительных вычислений на примере программы молекулярной динамики для суперкомпьютера Воронежского государственного университета. Показано, что код на этом языке является более доступным для разработки программного обеспечения в условиях необходимости переноса части вычислений на математические сопроцессоры Intel или Nvidia.
Ключевые слова: высокопроизводительные вычисления, суперкомпьютер, компьютерный кластер, язык программирования Python, MPI, pyMIC, pyCUDA.

Введение

Тенденции в развитии современных суперкомпьютерных комплексов указывают на то, что путь наращивания мощности топовых систем определен окончательно. Вычислительные комплексы, входящие в верхнюю часть списка Top500 суперкомпьютеров мира, создаются по гибридной схеме и имеют в своем составе не только универсальные процессоры, но и энергоэффективные математические сопроцессоры, такие как Nvidia Tesla, AMD Instinct или Intel Xeon Phi. Линейка последних была закрыта в 2018 году, тем не менее, машины с использованием ускорителей Intel Xeon Phi еще работают во многих научных учреждениях мира, в том числе в ОИЯИ (Дубна) и в Суперкомпьютерном центре Воронежского государственного университета [1, 2].

Изменение аппаратной базы современных суперкомпьютеров требует новых подходов к проектированию программ и подготовке квалифицированных специалистов, способных к реализации математического кода не только с использованием технологии MPI, но и с применением инструментов CUDA. Несмотря на принципиальную необходимость в наличии навыков программирования на языке C для реализации алгоритмов на ускорителях Nvidia, Intel и AMD, задачу можно существенно упростить переносом части операций верхнего уровня в код на языке Python [3,4].

1. Высокопроизводительные вычисления на сопроцессорах Intel Xeon Phi и Nvidia Tesla с использованием языка Python

Популярность языка Python в научной среде вызвана целым рядом факторов, сделавших его наиболее удобным для прикладных исследований. В частности, возможность создания всевозможных «оберток» над инструментами более низкого уровня, написанных на компилируемых языках, позволила прикладным специалистам не отвлекаться на тонкости взаимодействия с аппаратной частью, а профильным – существенно сократить время разработки конечного продукта.

К таким инструментам можно отнести модуль mpi4py, предоставляющий широкие возможность использования стандарта интерфейса передачи сообщений MPI [5], позволяя приложениям Python получать доступ к системным вычислительным ресурсам. Так как этот стандарт всегда являлся основной моделью межпроцессорного взаимодействия в суперкомпьютерах,

наличие подобного инструмента является существенным преимуществом использования языка Python при разработке параллельных программ.

Ниже представлен пример неблокирующей передачи сообщений с использованием модуля mpi4py [6].

```
from mpi4py import MPI
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
if rank == 0:
    data = {'x': 10, 'y': 20}
    req = comm.isend(data, dest=1, tag=12)
    req.wait()
elif rank == 1:
    req = comm.irecv(source=0, tag=12)
    data = req.wait()
```

Для работы с математическими сопроцессорами Python предоставляет два модуля PyMIC [7] и PyCUDA [8], позволяющие выполнять сопряжение «быстрого» кода на языке C внутри ядра CUDA или библиотеки Intel Xeon Phi. Принцип действия обоих модулей схож и заключаются в предоставлении интерфейса к основным операциям процессор/сопроцессор.

Одной из подлежащих распараллеливанию операций, в частности, является скалярное произведение векторов. Для разгрузки вычислений на сопроцессоры Tesla можно воспользоваться встроенной в pyCUDA функцией ReductionKernel.

Ниже представлен код функции с передачей массивов numpy на GPU [9]:

```
import numpy as np
import pycuda.autoinit
from pycuda import gpuarray
from pycuda.reduction import ReductionKernel
a = np.float32(np.random.random(3))
b = np.float32(np.random.random(3))
gpu_a = gpuarray.to_gpu(a)
gpu_b = gpuarray.to_gpu(b)
dot=(np.float32,   neutral   =»0»,   reduce_expr=»a+b»,   map_expr=»x[i]*y[i]»,
arguments=»float *x, float *y»)
result = dot(gpu_a, gpu_b).get()
```

Принцип работы ReductionKernel сходен с обходом цикла for в прагме openMP с последующим выполнением клаузы reduction.

Эту же операцию можно выполнить с использованием математического сопроцессора Intel Xeon Phi, а сходство архитектуры сопроцессора с универсальными процессорами фирм Intel и AMD позволяет в этом случае использовать уже известные приемы программирования. Далее представлен вариант исходного кода пользовательской библиотеки для интеграции с модулем PyMIC. Заметим, что код мало чем отличается от стандартной реализации скалярного произведения векторов на языке C [10].

```
#include <pymic_kernel.h>
#include <stdio.h>
PYMIC_KERNEL
float dot(float* a, float* b)
{
    float res = 0;
    #pragma omp parallel for reduction(+: res)
    for (int i = 0; i < 3; ++i)
    {
```

```
        res += a[i]*b[i];
    }
    return res;
}
```

Библиотека выполняет расчеты на сопроцессоре Intel Xeon Phi, получая данные из программы на языке Python, и возвращает обратно результат. Код программы с вызовом представленной ранее библиотеки (libdot.so) и генерацией векторов средствами модуля numpy выглядит следующим образом:

```
import pymic as mic
import numpy as np
device = mic.devices[0]
library = device.load_library(«libdot.so»)
stream = device.get_default_stream()
a = np.float32(np.random.random(3))
b = np.float32(np.random.random(3))
offl_a = stream.bind(a)
offl_b = stream.bind(b)
result = stream.invoke(library.dot, offl_a, offl_b)
stream.sync()
```

Таким образом, язык Python обладает всем необходимым инструментарием для организации параллельной обработки данных на суперкомпьютере. Коммуникацию между узлами кластера возможно осуществить силами модуля mpi4py, передачу данных на математические сопроцессоры можно выполнить с помощью модулей pyMIC и pyCUDA.

В рамках работы над выпускными квалификационными работами на кафедре цифровых технологий факультета компьютерных наук ведется разработка программного пакета для моделирования химических структур методом молекулярной динамики. Взаимодействие между узлами осуществляется с помощью модуля mpi4py, разгрузка части операций на ускорители выполняется через модули pyMIC и pyCUDA. В качестве межатомных потенциалов предполагается использовать функции Бехлера-Парринелло [11]. Задача в этом случае может быть легко распараллелена на сопроцессоры, так как сложность математического ядра невелика.

Заключение

В настоящей работе представлен опыт использования языка программирования Python для реализации задач высокопроизводительных вычислений на суперкомпьютере ВГУ. Показано, что с использованием модулей mpi4py, pyMIC и pyCUDA легко создавать и отлаживать программный код, а интерфейс для MPI во многом повторяет реализации MPI для языков программирования C и Fortran.

Для обучения специалистов работе на современных суперкомпьютерах целесообразно начинать обучения с использования более простых в программировании сопроцессоров фирмы Intel с дальнейшим переходом на сопроцессоры Nvidia, которые стали сегодня признанным стандартом в отрасли. При этом использование для операций верхнего уровня модулей PyMIC и PyCUDA позволяет существенно упростить работу с суперкомпьютером и предоставить «бесшовную» интеграцию с инструментами популярного языка программирования Python.

Литература

1. *Кургалин С. Д.* Суперкомпьютерные технологии в Воронежском государственном университете / С. Д. Кургалин, С. В. Борзунов // Вестник Воронежского гос. ун-та. Серия: Проблемы высшего образования. – 2018. – № 3. – С. 183–187.

2. *Кургалин С. Д.* Using the resources of the Supercomputer Center of Voronezh State University in learning processes and scientific researches / С. Д. Кургалин, С. В. Борзунов // Труды международной конференции «Суперкомпьютерные дни в России». – Москва : Изд-во МГУ, 2018. – С. 972–977.

3. *Борзунов С. В.* Языки программирования. Python: решение сложных задач : учебное пособие для высшего образования / С. В. Борзунов, С. Д. Кургалин. – Санкт-Петербург : Лань, 2023. – 192 с.

4. *Kurgalin S.* Algebra and Geometry with Python / S. Kurgalin, S. Borzunov. – Springer, 2021. – xvi, 425 p.

5. *Grama A.* Introduction to Parallel Computing / A. Grama, V. Kumar, A. Gupta, G. Karypis. – Second edition. – Addison-Wesley, 2003. – xx, 636 p.

6. *Palach J.* Parallel Programming with Python / J. Palach. – Packt Publishing,2014. – 103 p.

7. pymic: A python offload module for the intel(r) xeon phi(tm) coprocessor // https://software.intel.com URL: https://software.intel.com/en-us/articles/ pymic-a-python-offload-module-for-the-intelr-xeon-phitm-coprocessor (дата обращения: 10.03.2018).

8. pycuda 2022.2.2 documentation // https://documen.tician.de URL: https://documen.tician.de/pycuda/ (дата обращения: 12.05.2023).

9. *Тоуманен Б.* Программирование GPU при помощи Python и CUDA / Б. Тоуманен. – Москва : ДМК Пресс, 2020. – 253 с.

10. *Jeffers J.* Intel Xeon Phi Coprocessor High-Performance Programming / J. Jeffers, J. Reinders. – Elsevier, 2013. – 409 p.

11. *Behler J.* Atom-centered symmetry functions for constructing high-dimensional neural network potentials / J. Behler // Journal Chemical Physics. – 2011. – V. 134, № 7. – P. 074106.

УДК 004.9

МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ С ИСПОЛЬЗОВАНИЕМ GPU

А. А. Рябчиков

Воронежский государственный университет

Аннотация. Изначально созданные для улучшения графических возможностей в компьютерных системах, сегодня GPU, благодаря их уникальной архитектуре, стали фундаментом для решения сложных научных задач. В рамках этой статьи будет проведен детальный анализ основ архитектуры GPU, особенностей их интеграции в современные вычислительные системы и практическому применению GPU в различных научных областях, с акцентом на достигнутые результаты, преимущества и возможные ограничения.
Ключевые слова: графические процессоры (GPU), компьютерные игры, параллельная обработка, биоинформатика и геномика, машинное обучение, центральные процессоры (CPU), симуляции молекулярной динамики, глубокие нейронные сети, CUDA, OpenCL, программирование для GPU, иерархия памяти, пропускная способность.

Введение

В последние десятилетия мир вычислительных технологий претерпел настоящую революцию. Если ранее компьютеры и софт рассматривались в основном как инструменты для обработки текстовых данных и базовых вычислений, то сейчас они стали мощными устройствами, способными моделировать реальные физические процессы. В центре этой революции — графические процессорные устройства или GPU. Первоначально разработанные для улучшения качества изображений в видеоиграх и мультимедийных приложениях, эти устройства быстро нашли применение в более сложных и технически насыщенных сферах.

Процессоры GPU отличаются от обычных центральных процессоров (CPU) своей способностью одновременно обрабатывать большие объемы данных. Именно такая параллельная обработка позволила ускорить многие вычислительные процессы в разы. Согласно исследованиям, некоторые задачи, решаемые на GPU, могут выполняться в 10–100 раз быстрее, чем на традиционных CPU.

С применением GPU в научных исследованиях, ученые получили возможность моделировать более сложные системы и проводить эксперименты на компьютере, которые раньше были недоступны из-за ограничений производительности. Например, в области квантовой механики, где требуется обработка огромных матриц для предсказания поведения частиц, GPU позволили сократить время расчетов с недель до нескольких часов.

Этот обзор посвящен рассмотрению роли GPU в моделировании физических процессов. Мы подробно рассмотрим архитектурные и технические аспекты GPU, их преимущества по сравнению с традиционными вычислительными системами, а также представим конкретные примеры исследований, в которых использование GPU привело к настоящим прорывам.

1. История GPU

История развития графических процессорных устройств (GPU) началась в далеких 1970-х годах, периоде активного распространения персональных компьютеров. Первые графические адаптеры были созданы в ответ на растущий спрос пользователей на качественное графическое представление данных. Это было время, когда индустрия видеоигр начала свое восхождение, и спрос на высококачественную графику только увеличивался.

В 1980-х и 1990-х годах технологический прогресс в области производства полупроводников и архитектурные инновации способствовали быстрому росту производительности GPU. Они перестали быть простыми графическими адаптерами и превратились в сложные вычислительные системы, способные обрабатывать миллионы пикселей одновременно.

Однако настоящий революционный прорыв произошел в 2000-х, когда компания NVIDIA ввела архитектуру CUDA. Этот инновационный подход позволил GPU не просто обрабатывать графику, но и выполнять общие вычислительные функции. Таким образом, GPU стали полноценными участниками вычислительного процесса, наряду с центральными процессорами.

Введение архитектуры CUDA дало толчок разработке новых методов и алгоритмов, оптимизированных под GPU. Научные исследования, ранее считавшиеся чрезмерно ресурсоемкими, теперь могли быть выполнены в разы быстрее. Например, задачи, связанные с анализом генома, молекулярной динамикой или исследованиями в области климата, стали доступнее благодаря мощи GPU.

Так, в последующие годы наблюдался активный рост числа публикаций и исследований, в которых акцент делался на использовании GPU. Специфические вычислительные задачи, такие как глубокое обучение и искусственные нейронные сети, особенно выиграли от этой инновации.

В итоге, трансформация GPU из простых графических ускорителей в универсальные вычислительные системы открыла новые горизонты для научных исследований и индустрии в целом.

2. Архитектура и характеристики GPU

Современные графические процессорные устройства (GPU) представляют собой высокотехнологичные вычислительные комплексы, результат многолетних исследований и инноваций в области микроэлектроники. Главная особенность GPU заключается в их архитектуре: в отличие от традиционных центральных процессоров (CPU) с несколькими ядрами, GPU имеют сотни и даже тысячи ядер, способных работать параллельно.

Эта многопоточность GPU стала откликом на потребность рынка в более быстрой и эффективной графической обработке. Однако со временем исследователи поняли, что потенциал GPU можно использовать не только в графике. Такие характеристики, как высокая пропускная способность памяти и эффективное использование потоков, делают их идеальным инструментом для сложных вычислительных задач.

Для понимания масштаба: если рассмотреть такой GPU, как NVIDIA Tesla V100, он содержит 5120 CUDA ядер и имеет пропускную способность памяти до 900 GB/s. Такие характеристики позволяют ему обрабатывать сложные задачи, такие как глубокое обучение или трехмерное моделирование, в разы быстрее, чем традиционные CPU.

В контексте научного моделирования GPU предоставляют уникальную возможность для обработки огромных массивов данных. Будь то климатическое моделирование, квантовая механика или биоинформатика, возможность одновременной обработки тысяч и миллионов данных позволяет ученым получать результаты в реальном времени и с высокой степенью точности.

В заключение, можно утверждать, что архитектура и характеристики современных GPU предоставляют исследователям мощный инструмент, который может кардинально изменить подход к научным исследованиям и анализу данных.

3. Программные аспекты и инструменты

В современном мире графических процессорных устройств (GPU) критически важно обладать правильными инструментами для эффективной их эксплуатации. Поскольку архитекту-

ра GPU отличается от традиционных центральных процессоров (CPU), разработка программ для их эффективного использования требует специализированных знаний и инструментов.

CUDA (Compute Unified Device Architecture) — это параллельная платформа вычислений и программная модель, созданная компанией NVIDIA. Она предоставляет разработчикам прямой доступ к виртуальной инструкционной архитектуре GPU. С момента ее введения в 2007 году CUDA стала одной из ключевых платформ для глубокого обучения, искусственного интеллекта и научных вычислений. С помощью CUDA разработчики могут оптимизировать свои приложения для максимальной производительности на GPU, что в некоторых случаях позволяет достигать ускорения в сотни раз.

OpenCL (Open Computing Language) — это открытый стандарт, предназначенный для написания программ, выполняемых на различных вычислительных платформах. В отличие от CUDA, который специфичен для продукции NVIDIA, OpenCL поддерживается многими производителями, включая AMD, Intel и даже Apple. Он предоставляет универсальный интерфейс для программирования GPU, CPU и даже FPGA.

Оба инструмента предоставляют библиотеки и фреймворки, которые помогают ученым и инженерам адаптировать и оптимизировать свои алгоритмы для выполнения на GPU. Например, в области физики частиц или биоинформатики, где требуется обработка больших объемов данных, использование этих инструментов может значительно сократить время расчетов.

Таким образом, эффективное использование GPU в научных исследованиях и индустриальных приложениях требует знания специализированных программных инструментов, таких как CUDA и OpenCL. Эти инструменты не только ускоряют вычисления, но и предоставляют возможность создания более сложных и детализированных моделей и алгоритмов.

4. Практические примеры использования GPU

Гидродинамика. GPU оказались революционным инструментом в гидродинамике, особенно при моделировании сложных водных потоков и динамики жидкостей и газов. Используя традиционные CPU-методы, исследователи часто сталкивались с проблемами длительных расчетов, особенно при больших объемах данных. Однако с внедрением GPU в эту сферу было замечено ускорение в десятки раз. Например, исследования, связанные с моделированием цунами или потоков рек, которые раньше занимали недели, теперь могут быть завершены за несколько дней или даже часов.

Квантовая механика. В области квантовой механики часто требуются расчеты больших и сложных матриц, что является исключительно трудоемкой задачей для стандартных вычислительных систем. С появлением GPU исследователи стали способными проводить эти расчеты значительно быстрее. Например, исследования взаимодействия квантовых частиц, которое раньше занимало месяцы расчетов на традиционных системах, с использованием GPU может быть выполнено всего за несколько дней, сохраняя при этом высокую степень точности.

Космические исследования. В астрофизике и космологии GPU предоставили исследователям инструменты для изучения явлений на новом уровне. Будь то моделирование формирования звезд, исследование космического излучения или динамика галактик, GPU позволили достигать более детализированных и точных результатов. Например, при моделировании столкновений галактик, ученые обнаружили, что с помощью GPU они могут более детально изучать влияние темной материи и темной энергии, что раньше было практически невозможно из-за ограничений производительности.

В целом, применение GPU в различных областях науки открыло новые горизонты для исследований, предоставив ученым мощные инструменты для решения сложных задач.

5. Преимущества и ограничения использования GPU

Графические процессорные устройства (GPU) уже давно перешли от их первоначального назначения — обработки графики — к более широкому спектру применения, особенно в научных исследованиях. Это стало возможным благодаря ряду преимуществ, которые они предоставляют:

Преимущества:

Высокая скорость обработки: GPU спроектированы так, чтобы быстро обрабатывать параллельные задачи, что делает их идеальными для научного моделирования, где одни и те же вычисления могут выполняться для многих данных одновременно.

Параллелизм: Сотни или даже тысячи ядер в GPU позволяют одновременно выполнять многие операции, что ускоряет процессы, такие как рендеринг изображений или сложные расчеты.

Эффективное использование ресурсов: Графические процессоры эффективно распределяют нагрузку, максимизируя использование каждого ядра.

Однако, как и любое технологическое новшество, GPU имеют свои ограничения и вызовы:

Ограничения:

Ограниченная универсальность: несмотря на то что GPU могут обрабатывать множество задач быстрее, не все задачи поддаются эффективной параллелизации. Некоторые последовательные задачи могут выполняться медленнее на GPU, чем на CPU.

Требования к кодированию: Написание эффективного кода для GPU требует знания специфических языков программирования, таких как CUDA или OpenCL. Это может создать порог вхождения для некоторых исследователей.

Проблемы с памятью: Память GPU обычно ограничена, и передача данных между основной памятью компьютера и GPU может стать узким местом.

Тепловые и энергетические ограничения: Высокая производительность GPU также означает большое тепловыделение, что требует эффективного охлаждения, и, в некоторых случаях, может привести к более высокому энергопотреблению.

В заключение, несмотря на ряд преимуществ, которые GPU предоставляют научному сообществу, важно также учитывать и их ограничения. Правильное понимание этих аспектов позволит исследователям наиболее эффективно интегрировать GPU в свои исследовательские проекты.

6. Перспективы развития

По мере того, как технологии продолжают развиваться, аппаратные возможности GPU продолжат улучшаться, что приведет к еще большему росту производительности. На основе текущих тенденций, существуют прогнозы, что будущие поколения GPU будут способны обрабатывать данные в десятки, а может быть и в сотни раз быстрее текущих моделей.

Следует также отметить, что программное обеспечение и инструменты, разработанные для GPU, будут продолжать улучшаться, становясь более дружелюбными для пользователя и более эффективными. Это обеспечит более простую интеграцию GPU в различные научные дисциплины, возможно, даже открыв новые области исследований, которые раньше считались непрактичными или невозможными.

Кроме того, с ростом интереса к квантовым вычислениям и их потенциальной интеграции с классическими системами, GPU могут играть ключевую роль в создании гибридных вычислительных систем, объединяющих лучшие черты обеих технологий.

Заключение

Графические процессоры (GPU) сыграли революционную роль в развитии современной научной и инженерной деятельности. Первоначально они были разработаны исключительно для обработки графики, особенно для улучшения графического представления в компьютерных играх. Однако по мере того как технология развивалась, возросла и сложность задач, которые можно было решать с использованием GPU. Сейчас GPU используются в самых разных областях, от биоинформатики и геномики до машинного обучения и искусственного интеллекта.

Основной причиной такого широкого использования GPU в научных исследованиях является их уникальная архитектура. Пока традиционные центральные процессоры (CPU) оптимизированы для последовательной обработки, GPU состоят из тысяч маленьких ядер, работающих параллельно. В эпоху, когда объемы данных увеличиваются с каждым годом, эта параллельная архитектура делает GPU идеальными для обработки больших массивов данных. Например, симуляции молекулярной динамики или тренировка глубоких нейронных сетей, требующие обработки огромных объемов информации, могут получать значительное ускорение при использовании GPU.

Однако переход к использованию GPU не лишен проблем. Программирование для GPU требует особого набора навыков и понимания их архитектуры. Для максимизации производительности необходимо учитывать многие факторы, такие как иерархия памяти, латентность и пропускная способность. И хотя существуют специализированные инструменты, такие как CUDA и OpenCL, чтобы упростить этот процесс, все равно требуется значительное вложение времени и ресурсов.

Тем не менее, даже учитывая эти проблемы, преимущества использования GPU в научных исследованиях делают их неотъемлемым инструментом для современного научного сообщества. С их помощью исследователи могут достигать результатов быстрее и проводить эксперименты, которые раньше казались невозможными.

Литература

1. *Smith A.* GPU Computing in Modern Physics // Journal of Computational Physics. – 2021.

2. NVIDIA. NVIDIA Tesla V100 Architecture // NVIDIA Whitepapers. – 2017.

3. Owens J.D. GPU Computing in Modern Research // High-Performance Computing Journal. – 2008.

4. *James L.* Parallel Processing in Quantum Simulations // Quantum Physics Reports. – 2016.