МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ
Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

Обеспечение доступа к кластеру с помощью веб-приложения для моделирования взаимодействия N тел

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 273 группы
направления 01.04.02 Прикладная математика и информатика
факультета компьютерных наук и информационных технологий
Акимова Артемия Андреевича

Научный руководитель
доцент, к.ф.-м.н. _____ К.П. Вахлаева

Зав.кафедрой
к.ф.-м.н. _____ М.В. Огнева

Саратов 2019

Актуальность темы. На протяжении последнего десятилетия графические ускорители стремительно наращивали производительность, чтобы удовлетворить непрерывно растущие запросы разработчиков графических приложений и компьютерных игр. Кроме того, за последние несколько лет изменились некоторые фундаментальные принципы разработки графической аппаратуры, в результате чего она стала более программируемой, чем когда-либо ранее. Сегодня графический ускоритель – это гибко программируемый массивно-параллельный процессор для обработки чисел в формате с плавающей точкой, возможности которого могут быть востребованы для решения целого ряда вычислительно трудоемких задач.

Графический процессор – это гибко программируемый массивно-параллельный процессор для обработки чисел в формате с плавающей точкой, возможности которого могут быть востребованы для решения целого ряда вычислительно трудоемких задач [1]. Одной из таких задач является задача N тел, которая представляет собой расчет поведения системы тел, взаимодействующих при помощи дальнодействующих сил. [2]. Примером является астрофизическое моделирование, в котором каждое тело представляет собой галактику или отдельную звезду, и тела притягиваются друг к другу через гравитационную силу. Различные варианты задачи возникают и во многих других проблемах вычислительной науки. [3 – 4].

На практике широкое применение получили процессоры от двух основных производителей: Nvidia и AMD [5 – 6]. Моделирование турбулентного потока жидкости и вычисление глобальной освещенности в компьютерной графике являются также примерами проблем, которые используют моделирование задачи гравитирующих N-тел [7].

Параллельное моделирование взаимодействия N тел может быть выполнено удаленно на вычислительном кластере с использованием технологии Nvidia CUDA. Для организации такого удаленного доступа к вычислительному кластеру необходимо знать его структуру и особенности

[8], что позволит в дальнейшем разработать веб-приложение, для удаленной работы с графическим процессором на кластере [9-10].

Цель работы заключается в реализации веб-приложения для работы на вычислительном кластере mpi-m1.sgu.ru, обеспечение безопасности данного веб-приложения и его тестирование на реализованном алгоритме всех пар решения задачи взаимодействия N гравитирующих тел. Поставленная цель определила следующие задачи работы:

1. рассмотреть математическую постановку задачи взаимодействия N тел по гравитационному закону;
2. изучить архитектуру графических процессоров и Nvidia CUDA;
3. изучить технологию NodeJS для работы с файловой системой, позволяющей заменить консоль при работе с кластером на веб-приложение;
4. реализовать последовательный алгоритм всех пар решения задачи N тел и его параллельную версию на основе OpenMP;
5. реализовать параллельные алгоритмы всех пар и Барнса-Хата решения задачи N тел с использованием Nvidia CUDA;
6. выполнить вычислительные эксперименты на суперкомпьютере «Ломоносов» для алгоритма всех пар, реализованного с использованием Nvidia CUDA.
7. реализовать веб-приложение для работы с кластером mpi-m1.sgu.ru и протестировать его работу на параллельном алгоритме всех пар решения задачи N тел с использованием OpenMP.

Методологические основы для работы представлены в работах Адинец, А. В., Кривова, М. А. [1]; Барнса, Дж [2, 3]; Калгина, К.Ю [4]. документации по Nvidia CUDA [5 – 8] и NodeJS [9, 10].

Теоретическая и практическая значимость магистерской работы.

В ходе выполнения магистерской работы были решены все поставленные задачи, что позволило достигнуть заявленные цели –

реализация веб-приложения для работы на вычислительном кластере mpi-m1.sgu.ru, обеспечение безопасности данного веб-приложения и его тестирование на реализованном алгоритме всех пар решения задачи взаимодействия N гравитирующих тел.

Несомненным достоинством данной работы является то, что она позволяет использовать разработанный алгоритмы ходе последующего обучения студентов технологиям Nvidia CUDA в рамках курса по параллельному и распределнному программированию. Данный проект не имеет некоммерческих аналогов, программный код которого является открытым программным обеспечением в данный момент. Реализованный алгоритм всех пар для взаимодействия N гравитирующих тел достаточно прост для изучения технологи Nvidia CUDA, но в то же время наглядным образом показывает его состоятельность с точки зрения вычислительной эффективности. С другой же стороны реализованное веб-приложение позволяет значительно упростить работу с кластерной системой, упрощая организацию доступа к нему. Это обусловлено заменой консольного приложения, на изучение которого тратилась большая часть курса обучения.

Структура и объём работы. Магистерская работа состоит из введения, шести разделов, заключения, списка использованных источников и 6 приложений. Общий объем работы – 105 страниц, из них 65 страниц – основное содержание, включая 24 рисунка, 3 таблицы, список использованных источников информации – 30 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Задача взаимодействия N гравитирующих тел» посвящен описанию основной задачи взаимодействия N тел.

Задача N тел является задачей расчета эволюции поведения системы из N тел, которые взаимодействуют при помощи некоторых сил. Эта задача возникает при численном моделировании процессов в различных областях науки: молекулярной динамике, астрономии, жидкостной динамике, электростатике. При этом силы, действующие между частицами в модели, как правило, имеют дальнодействующий характер. Если количество частиц N, то количество взаимодействий между ними, которые необходимо учитывать, растет как $O(N^2)$. Этим обуславливается высокая вычислительная сложность задачи. Помимо тривиального алгоритма взаимодействия, существуют различные приближенные схемы. В среднем они позволяют снизить сложность с $O(N^2)$ до $O(N \log N)$, или даже $O(N)$, однако имеют худшие показатели точности. Не для всех имеются оценки ошибки сверху. Более того, для расчета взаимодействия в скоплениях близко расположенных частиц часть этих методов все равно использует взаимодействие типа каждый-с-каждым внутри скопления. А поскольку общее число частиц может быть достаточно большим, размер среднего скопления также оказывается большим. Что также требует много вычислительных ресурсов для расчета $O(N^2)$ взаимодействий, где N в этом случае является размером скопления. Таким образом, для решения задачи N тел в любом случае требуются большие вычислительные ресурсы. В настоящее время, существует целый ряд высокопроизводительных вычислительных архитектур. Прежде всего, это традиционные центральные процессоры, а также построенные на их основе вычислительные кластеры. Далее рассмотрена реализация наиболее ресурсоемкой части задачи N тел, расчета взаимодействия частиц «каждый-с-каждым», на центральном и графическом процессорах. Задача N тел позволяет также оценить эффективность компилятора для данного процессора на вычислительно ёмких вычислениях.

В подразделе «Постановка задачи взаимодействия N тел» приводится фактическое описание задачи, математическое обоснование данной задачи.

В задаче N тел взаимодействие вычисляется отдельно между парами частиц, а сила, действующая на каждую частицу (i-частицу) является суммой вкладов отдельных частиц (j-частиц), которые на нее действуют. Каждая частица характеризуется набором параметров, таких как масса или заряд. Кроме того, каждая частица характеризуется скоростью $\vec{v_i}$ и положением $\vec{r_i}$. Задачей ядра вычисления взаимодействия является расчет ускорения, которые частица получает в результате влияния на нее других частиц

Подраздел «Описание алгоритма всех пар и стратегия распараллеливания» содержит теоретическое описание алгоритма, стратегию работы алгоритма в параллельной версии.

Второй раздел «Основные теоретические сведения о графических процессорах» посвящен сравнительному анализу различных графических устройств, краткому описанию их и исторической подоплеке.

Современные графические процессорные устройства (ГПУ) являются мощными специализированными вычислительными устройствами, которые могут быть использованы для ускорения решения целого ряда вычислительно сложных задач. К таким задачам относятся матричные операции, сеточные методы, задачи обработки изображений, машинного зрения, финансовых расчетов и т.д. На таких задачах удается получить значения реальной производительности ГПУ до 300 Гфлопс на одной машине, обеспечивая ускорение порядка 10 раз по сравнению с использованием ресурсов только центрального процессора. Помимо высокой производительности, ГПУ обладают рядом других преимуществ: низкое энергопотребление (около 0.5 Вт/ГФлопс), низкая стоимость (около $0.4/ГФлопс), высокая доступность.

В подразделе «Архитектура ГПУ и система C$» описывается основной подход к созданию графических устройств, к их программированию а работе на аппаратном уровне.

В подразделе «Этапы трансляции на ГПУ, их особенности» приводится описание цикла работы ГПУ с данными и обработка функциональных операций.

Третий раздел «Реализация алгоритмов для решения задачи взаимодействия N гравитирующих тел» посвящен описанию алгоритмов всех пар и Барнса-Хата, моделирование динамической системы, на которой их последующая реализация на различных вычислительных устройствах.

В качестве модельной системы используется набор из 800 точечных тел (частиц), расположенных в узлах прямоугольной сетки и вращающихся вокруг центра этой сетки. Массы частиц изменяются в диапазоне от 100 до 199 расчетных единиц; на приведенных рисунках более массивные частицы представлены в виде кругов с большим радиусом. В качестве тестовых значений, которые в были использованы для проверки правильности различных модификаций исходной программы, выводятся значения координаты начальной частицы после 100 итераций.

При моделировании системы (с учетом наложенного ограничения на максимальное значение силы) можно достаточно произвольным образом выбирать такие характеристики, как массы тел, их скорости, расстояния между ними и значение гравитационной постоянной.

В подразделе «Реализация алгоритма всех пар» приводится последовательная версия алгоритма всех пар.

В задаче N тел взаимодействие вычисляется отдельно между парами частиц, а сила, действующая на каждую частицу (i-частицу) является суммой вкладов отдельных частиц (j-частиц), которые на нее действуют. Каждая частица характеризуется набором параметров, таких как масса или заряд. Кроме того, каждая частица характеризуется скоростью $\vec{v_i}$ и положением $\vec{r_i}$. Задачей ядра вычисления взаимодействия является расчет ускорения, которые частица получает в результате влияния на нее других частиц.

В подразделе «Распараллеливание алгоритма всех пар с использованием OpenMP» приводится модификация предыдущей реализации.

Подраздел «Распараллеливание алгоритма всех пар с использованием Nvidia CUDA» посвящен основным теоретическим основам Nvidia CUDA, а так же практическому применению полученных знаний для реализации параллельной версии алгоритма всех пар при помощи Nvidia CUDA.

Compute Unified Device Architecture (CUDA) – это программная модель, включающая описание вычислительного параллелизма и иерархичной структуры памяти непосредственно в язык программирования. С точки зрения программного обеспечения, реализация CUDA представляет собой кроссплатформенную систему компиляции и исполнения программ, части которых работают на CPU и GPU. CUDA предназначена для разработки GPGPU-приложений без привязки к графическим API и поддерживается всеми GPU Nvidia, начиная с серии GeForce8.

Концепция CUDA отводит GPU роль массивно-параллельного сопроцессора. В литературе о CUDA основная система, к которой подключен GPU, для краткости называется термином хост (host), аналогично сам GPU по отношению к хосту часто называется просто устройством (device). CUDA-программа задействует как CPU, так и GPU, на CPU выполняется последовательная часть кода и подготовительные стадии для GPU-вычислений. Параллельные участки кода могут быть перенесены на GPU, где будут одновременно выполняться большим множеством нитей (threads). Важно отметить ряд принципиальных различий между обычными потоками CPU и нитями GPU:

- Нить GPU чрезвычайно легковесна, ее контекст минимален, регистры распределены заранее;
- Для эффективного использования ресурсов GPU программе необходимо задействовать тысячи отдельных нитей, в то время как на многоядерном CPU максимальная эффективность обычно

достигается при числе потоков, равном или в несколько раз большем количества ядер.

В целом работа нитей на GPU соответствует принципу SIMD, однако есть существенное различие. Только нити в пределах одной группы (для GPU архитектуры Fermi – 32 нити), называемой варпом (warp) выполняются физически одновременно. Нити различных варпов могут находиться на разных стадиях выполнения программы. Такой метод обработки данных обозначается термином SIMT (Single Instruction – Multiple Threads). Управление работой варпов производится на аппаратном уровне.

Подраздел «Реализация алгоритма Барнса-Хата при помощи Nvidia CUDA» посвящен реализации самого быстрого с точки зрения вычислительной сложности алгоритма для сравнения с более простыми, но менее производительными алгоритмами.

Четвертый раздел «Технология NodeJS» посвящен описанию технологии NodeJS, базовым принципам работы с ним и основным паттернам, используемым при разработке серверного приложения при помощи данной библиотеки для организации удаленного доступа к кластерным системам.

NodeJS рассчитан на задачи, имеющим веб-инфраструктуру и мобильные приложения, в back-end которых надо в режиме real-time вносить изменения, используя архитектуру, построенную на базе микросервисов. NodeJS способен существенно сократить время на разработку приложения, не меняя при этом логику приложения.

NodeJS продолжает развиваться динамически и амбициозно. Так например, в течении нескольких последних лет разработчиками было добавлено около 200 000 модулей для NodeJS, это превышает в несколько раз темпы развития уже устоявшихся серверных языков, таких как Perl у которого количество модулей в репозитории меньше. Так же нужно подчеркнуть что технология NodeJS набирает обороты и используется

такими компаниями, как Yahoo, Microsoft, PayPal и LinkedIn, не говоря уже про Google.

В подразделе «Установка NodeJS» приводится подробное описание подготовки окружения к непосредственной разработке на NodeJS.

Благодаря тому, что NodeJS имеет отличный инструмент, NPM — менеджер пакетов, с его помощью можно управлять модулями и зависимостями. Его легко использовать и масштабировать в рамках серверного окружения. Так, например, используя NodeJS для нескольких проектов, можно установить пакеты/модули как глобально, так и локально.

Так же есть ряд дополнительных инструментов для комфортной работы с NodeJS. Так, например, для поддержания процессов используют утилиты: forever или supervisor. Первая устанавливается из менеджера пакетов NPM и служит только для поддержания процессов NodeJS, в то время как второй умеет работать и с другими утилитами такими как RabbitMQ, Bash Scripts и тем самым является более универсальной. Так же существует модуль supervisor который устанавливается как пакет для NodeJS и играет роль наблюдателя за изменениями в скрипте и автоматического перезапуска без утечки памяти ОЗУ и без очистки межмодульных зависимостей.

В подразделе «Работа с удаленным сервером при помощи NodeJS» описываются особенности и возможности технологии NodeJS.

Пятый раздел «Вычислительные кластеры СГУ и МГУ» посвящен описанию кластерных систем СГУ и МГУ и полученных результатов работы описанных выше алгоритмов на данных кластерных системах.

В подразделе «Вычислительный кластер СГУ» приводится подробное описание возможностей кластера СГУ.

Вычислительный кластер СГУ состоит из 5 узлов, один из которых является головным (s9.cluster.sgu.ru), и четырех счетных узлов. Узлы кластера полностью однородные (как hardware, так и software).

Краткая характеристика узла кластера:

- RAM 16 Гб,

- Intel(R) Xeon(R) CPU E5345 @ 2.33ГГц (8 процессоров на узле),
- OS Fedora 10,
- /home монтируемая по NFS директория, 1Тб,
- /storage директория для хранения временных файлов на счетных узлах кластера, 1Тб,
- Сеть 1000 Mbit,
- gcc version 4. 3. 2.
- Адрес головного узла: s9.cluster.sgu.ru
- Адреса вычислительных узлов: s5... s8.cluster.sgu.ru
- Раздел /home экспортируется на счетные узлы кластера с головного узла (s9). Это означает, что задачи, выполняющие чтение/запись на счетных узлах, пересылают данные по сети с/на узел s9. В случае, если счетные задачи интенсивно работают с жестким диском, следует использовать разделы на счетных узлах, и для получения доступа к этим разделам следует обратиться к администратору кластера.

В подразделе «Вычислительный кластер СГУ» приводится подробное описание возможностей кластера МГУ.

В ноябре 2011 года Nvidia добавила Московский Государственный Университет имени М.В. Ломоносова к списку CUDA Center of Excellence. Статус CUDA Center of Excellence – это самая высокая награда, которую может получить институт за передовые достижения, полученные с использованием графических процессоров Nvidia и технологии Nvidia CUDA.

Для реализации разработанного алгоритма и был использован данный кластер.

Пиковая производительность	1,7 Пфлопс
Производительность на тесте Linpack	901.9 Тфлопс
Число вычислительных узлов x86	5 104
Число графических вычислительных узлов	1 065
Число вычислительных узлов PowerXCell	30
Число процессоров/ядер x86	12 346 / 52 168
Число графических ядер	954 240
Оперативная память	92 ТБ
Общий объем дисковой памяти вычислителя	1,75 ПБ
Основной тип процессора	Intel Xeon X5570/Intel Xeon 5670, Nvidia X2070
Число типов вычислительных узлов	8
Основной тип вычислительных узлов	ТВ2-XN
System/Servise/Management Network	QDR Infiniband 4x/10G Ethernet/Gigabit Ethernet
Система хранения данных	Параллельная файловая система Lustre, файловая система NFS, иерархическая файловая система StorNext, система резервного копирования и архивирования данных
Операционная система	Clustrx T-Platforms Edition
Занимаемая площадь	252 м^2
Потребление энергии	2,6 МВт
Вес всех составляющих	Более 75 тонн
Производитель	Т-Платформы
Год выпуска	2009 г.

Шестой раздел «Реализация веб-приложения» посвящен созданию веб-приложения для организации удаленного доступа к кластеру СГУ взамен существующему консольному приложению. Так же производится подробное руководство работы с ним и его графическое описание.

ЗАКЛЮЧЕНИЕ

Организация удаленного доступа к вычислительному кластеру в данной работе основана на технологии NodeJS, в связи с чем была выполнена практическая реализация тестового веб-приложения на NodeJS для работы с удаленным сервером, являющимся заменой консоли. Также были изучены основы архитектуры графических процессоров и технологии Nvidia CUDA, рассмотрена постановка задачи взаимодействия N тел по гравитационному закону для моделирования на графических процессорах, приведено описание технических характеристик вычислительного кластера. После реализации алгоритма всех пар и алгоритма Барнса-Хата при помощи Nvidia CUDA получены результаты, которые говорят о возможности и целесообразности распараллеливания алгоритма всех пар при помощи технологии Nvidia CUDA. Вычисления алгоритма всех пар были выполнены на суперкомпьютере «Ломоносов» и на кластере mpi-m1.sgu.ru. Для доступа к кластеру mpi-m1.sgu.ru через веб-интерфейс реализовано приложение, с помощью которого выполнялись расчеты параллельного алгоритма всех пар решения задачи N тел.

СПИСОК ОСНОВНЫХ ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Адинец, А. В., Кривов, М. А. Методы оптимизации программ для современных графических процессоров. Всероссийская научная конференция «Научный сервис в сети Интернет-2008: решение больших задач». 2008. С. 345 – 353.
2. Barnes J., Hut P., A hierarchical Onlogn/ force-calculation algorithm, Nature, 324, С. 446–449
3. Barnes J., A modified tree code: don’t laugh; it runs, J. Comput. Phys., С. 161 – 170.
4. Калгин, К.Ю., Иерархия памяти и эффективное программирование CUDA [Электронный ресурс]. URL: http://ssd.sscc.ru/sites/default/files/content/attach/332/cuda-4-mem.pdf, (дата обращения: 20.05.2019). Загл. с экр. Яз. рус.
5. Nvidia, CUDA Programming Guide, Version 2.2, Nvidia Corp., Santa Clara.
6. Samples for CUDA Developers which demonstrates features in CUDA Toolkit [Электронный ресурс]. URL: https://github.com/NVIDIA/cuda-samples, (дата обращения: 02.03.2019). Загл. с экр. Яз. англ.
7. Nvidia CUDA Programming Guide [Электронный ресурс]. URL: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (дата обращения: 12.05.2019). Загл. с экр. Яз. англ.
8. N-body simulations [Электронный ресурс]. URL: http://www.scholarpedia.org/article/N-body_simulations (дата обращения: 20.05.2019). Загл. с экр. Яз. англ.
9. NodeJS v11.1.0 Documentation [Электронный ресурс]. URL: https://nodejs.org/api/ (дата обращения: 28.04.2019). Загл. с экр. Яз. англ.
10. NodeJS server-side application [Электронный ресурс]. URL: https://blog.nodejitsu.com/keep-a-nodejs-server-up-with-forever/ (дата обращения: 01.05.2019). Загл. с экр. Яз. англ.