представлением. Компонент View фреймворка MVC – это в основном представление DOM. Это просто, когда пишется код, который взаимодействует с DOM, но фреймворку очень сложно обрабатывать различные манипуляции с DOM. Традиционные представления MVC обычно включают в себя много сложного пользовательского интерфейса. Если данные изменили крошечный элемент, то в конечном итоге приложение снова отобразится, и цикл продолжится. Причина этого заключается в том, что, как правило, большинство известных MVC-фреймворков используют двустороннюю привязку данных.

Таким образом, React Native подходит для быстрой разработки мобильных приложений и проверки гипотез**,** при любой квалификации пользователя, даже если он не является мобильным разработчиком. React Native постоянно развивается, появляются новые функции и библиотеки.

Литература

1. React Native для Мобильной разработки [Электронный ресурс] / Computer & IT Ebooks Centre. – Режим доступа: https://dl.ebooksworld.ir/motoman/Apress. React.Native.for.Mobile.Development.2nd.Edition.www.EBooksWorld.ir.pdf. – Дата доступа: 03.04.2023.

А. Н. Харлан
(ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **Е. А. Дей**, канд. физ.-мат. наук, доцент

ИЗУЧЕНИЕ ЭФФЕКТИВНОСТИ ОБРАБОТКИ МАТРИЦ С ИСПОЛЬЗОВАНИЕМ ВИДЕОКАРТЫ

Современная видеокарта (GPU, Graphics Processing Unit) – это специализированное устройство, которое используется для обработки графической информации и выполнения вычислений. Она содержит множество вычислительных блоков, называемых ядрами, которые работают параллельно и обрабатывают данные одновременно [1, 2].

CUDA (Compute Unified Device Architecture) и OpenCL (Open Computing Language) – это языки программирования, которые используются для программирования на видеокартах. Они используются для написания программ, которые могут выполняться на множестве ядер видеокарты с реализацией одновременных (параллельных) вычислений [1].

CUDA является языком программирования, разработанным компанией Nvidia. Он используется для программирования видеокарт, которые работают на архитектуре Nvidia. Одно из главных преимуществ языка CUDA заключается в том, что он предоставляет высокоуровневый API (Application Programming Interface) – программный интерфейс приложения, то есть, набор способов и правил, по которым различные программы общаются между собой и обмениваются данными. Это облегчает написание программ, реализующих параллельные вычисления. Кроме того, CUDA имеет богатый набор инструментов, таких как наборы библиотек, которые упрощают разработку приложений [1, 2].

OpenCL, с другой стороны, является языком программирования, который может использоваться на различных устройствах, таких как процессоры, FPGA (Field-Programmable Gate Array), то есть программируемая логическая матрица (ПЛМ), и видеокарты. Он является стандартом открытого кода и поддерживается множеством производителей устройств. Это позволяет разработчикам создавать кроссплатформенные приложения, которые могут выполняться на разных устройствах.

Задача данной работы состояла в освоении технологии вычислений на GPU с использованием CUDA, разработке программы, реализующей решение тестовой задачи по обработке матриц и проведении вычислительных экспериментов с целью исследования эффективности использования видеокарт для реализации большого объема вычислений.

Разработанная программа выполняет транспонирование матриц заданных размеров с использованием центрального процессора (CPU) и видеокарты (GPU). Результатом программы являются (рисунок 1)

– вывод вычисленных матриц в размере 10x10;
– вывод времени выполнения транспонирования для CPU и GPU;
– вывод информации об используемых CPU и GPU;
– вывод таблицы полученных данных, а также их отношение.

```
Информация об используемых устройствах :
Процессор : QuadCore Intel Core i5-7300HQ, 3200 MHz
Видеокарта : nVIDIA GeForce GTX 1060 with Max-Q Design

Размер матрицы:     | 10x10       | 50x50       | 100x100     | 500x500     | 1000x1000   | 5000x5000     | 10000x10000
Время CPU:          | 0,000800ms  | 0,014700ms  | 0,033400ms  | 1,059300ms  | 3,980500ms  | 200,573800ms  | 901,038400ms
Время GPU:          | 0,010240ms  | 0,009216ms  | 0,014336ms  | 0,114688ms  | 0,427008ms  | 11,326304ms   | 58,617855ms
Отношение CPU/GPU:  | 0,078125    | 1,595052    | 2,329799    | 9,236363    | 9,321839    | 17,708671     | 15,371398
```

Рисунок 1 – Пример вывода результатов одного из расчетов

Для графического представления данных построим график зависимости времени счета от размера используемых матриц (рисунок 2).

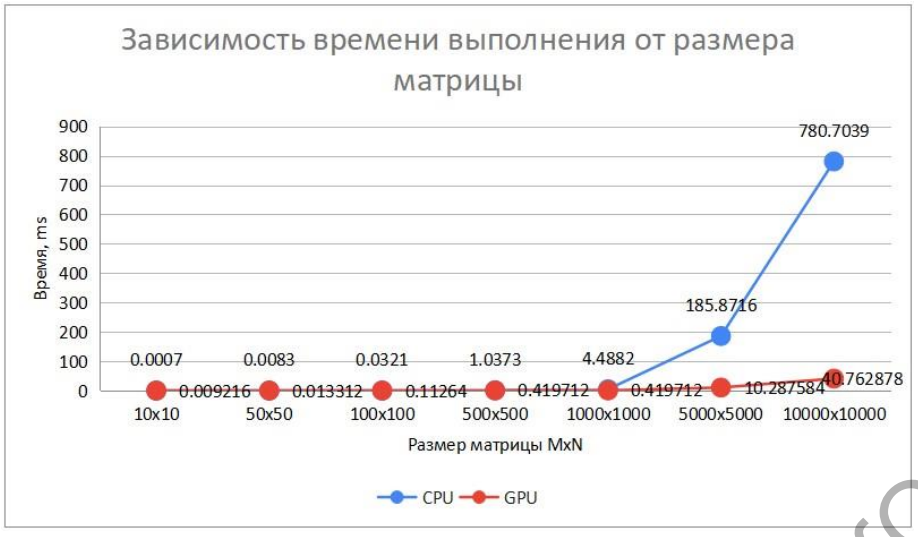

Рисунок 2 – Зависимость времени выполнения от размера используемых матриц

По ходу графику видно, что время выполнения при малых размерах матриц имеет малые различия для CPU и GPU. Однако, при переходе размера матрицы от 1000x1000 до 10000x10000 наблюдается сильное возрастание требуемого времени выполнения для CPU. Причём отношение времени CPU к GPU на промежутке от 5000x5000 до 10000x10000 близко к двум десяткам (рисунок 3).

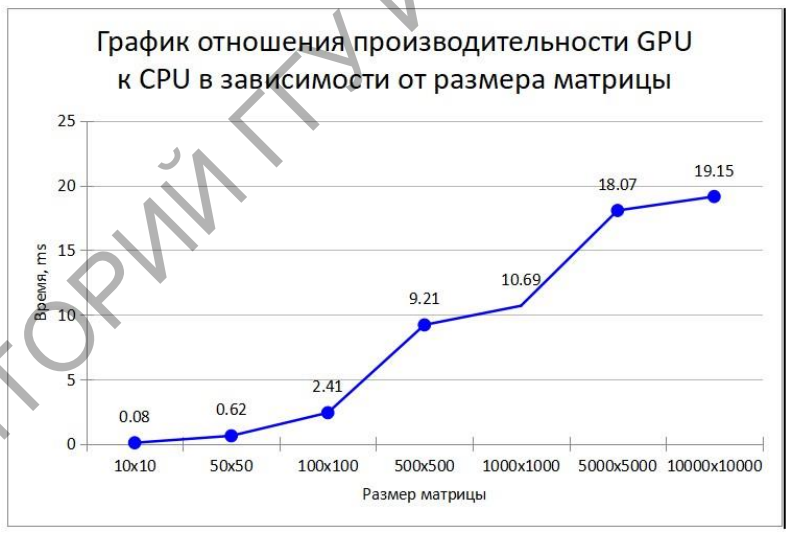

Рисунок 3 – Коэффициент эффективности GPU в зависимости от размера матриц

Проанализировав график можем сделать следующие выводы:

– При достаточно малых размерах матрицы (≤ 50x50) вычисления на GPU показывают большие временные затраты, в сравнении с вычислениями на CPU;

– Начиная с размера матрицы 100x100, в которой скорость выполнения задачи с использованием GPU примерно в 2.5 раза быстрее, производительность начинает резко расти, вплоть до конечного результата с разницей в 19 раз в пользу GPU.

Таким образом, использование GPU для вычислений не обеспечивает большей производительности при малых объёмах данных, а наоборот, требует больше времени на выполнение, чем CPU. Но при использовании больших объёмов данных вычисления на GPU показывают сильное превосходство над использованием CPU.

Литература

1. Боресков, А. В. Параллельные вычисления на GPU.Архитектура и программная модель CUDA / А. В. Боресков [и др.]. – 2-е издание. – М. : Изд-во МГУ, 2015. – 336 с.

2. Хабр [Электронныйресурс] Вычисления на GPU – зачем, когда и как. – Режим доступа: https://habr.com/ru/companies/dbtc/articles/498374/. – Дата доступа: 19.03.2023.

М. М. Хартон
(БГУ, Минск)
Науч. рук. **Г. В. Крылова**, канд. физ.-мат. наук

ЭЛЕКТРОННАЯ СТРУКТУРА ЭЛЕКТРОСТАТИЧЕСКИ УДЕРЖИВАЕМОЙ ГРАФЕНОВОЙ КВАНТОВОЙ ТОЧКИ НА ПОВЕРХНОСТИ ОДНОСТЕННОЙ УГЛЕРОДНОЙ НАНОТРУБКИ

Введение

В настоящее время, углеродные нанотрубки являются перспективным наноматериалом для разработок нанооптоэлектронных устройств на квантовых эффектах и сенсоров. Интерес к таким наноматериалам обусловлен чрезвычайно высокой подвижностью их носителей заряда [1]. Одностенная углеродная нанотрубка (ОУНТ) имеет цилиндрическую форму и представляет собой свернутый графеновый лист. Диаметр ОУНТ варьируется от 0,6 до 1 нм. В зависимости от направления сворачивания, характеризуемого таким параметром как хиральность, нанотрубки могут быть металлического или полупроводникового типа. Под воздействием электромагнитного импульса на графеновую поверхность и приложении электрического напряжения к подложке, на