МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра 806 «Вычислительная математика и программирование»

**Лабораторная работа №1**

**по курсу «Программирование графических процессоров»**

**Освоение программного обеспечения для работы с технологией CUDA. Примитивные операции над векторами.**

Выполнил: И. П. Моисеенков

Группа: М8О-408Б-19

Преподаватель: А.Ю. Морозов

Москва, 2022

**Условие**

**Цель работы**: ознакомление и установка программного обеспечения для

работы с программно-аппаратной архитектурой параллельных вычислений (CUDA), реализация одной из примитивных операций над векторами.

**Вариант 2**:

**Входные данные**. На первой строке задано число n - размер векторов. В

следующих 2-х строках, записано по n вещественных чисел - элементы векторов.

**Выходные данные**. Необходимо вывести n чисел - результат вычитания

исходных векторов.

**Программное и аппаратное обеспечение**

В качестве графического процессора использую видеокарту Nvidia GeForce GT 545, установленную на сервере преподавателя.

Compute capability : 2.1

Name : GeForce GT 545

Total Global Memory : 3150381056

Shared memory per block : 49152

Registers per block : 32768

Warp size : 32

Max threads per block : (1024, 1024, 64)

Max block : (65535, 65535, 65535)

Total constant memory : 65536

Multiprocessors count : 3

В качестве редактора кода использовался Visual Studio Code.

**Метод решения**

Вычитание векторов — это операция, которую можно легко распараллелить, т.к. выполняется она поэлементно. Поэтому на графическом процессоре будем производить вычитание поэлементно.

**Описание программы**

В программе создаются два динамических массива arr1, arr2. Они копируются на GPU. В функции ядра kernel происходит поэлементное вычитание элементов массива. Результат вычитания записывается в первый массив (чтобы не использовать лишнюю память).

В программе предусмотрена обработка ошибок, которые могут возникнуть при работе с cuda. При возникновении каких-то проблем программа выведет соответствующее сообщение с типом ошибки и завершит работу.

**Результаты**

Рассмотрим время работы программы на различных тестах при различных размерах сетки (и без использования графического процессора вообще). Будем замерять непосредственно время работы алгоритма. В качестве тестов используются вектора из дробных элементов различной длины. Результаты приведены в таблице ниже.

|  |  |  |  |
| --- | --- | --- | --- |
| Размер сетки ядра | 1000 элементов, мс | 100 000 элементов, мс | 1 000 000 элементов, мс |
| CPU | 0.005785 | 0.258287 | 2.41057 |
| <<<1, 32>>> | 0.043328 | 2.018464 | 19.880287 |
| <<<32, 32>>> | 0.033184 | 0.154048 | 1.412608 |
| <<<128, 128>>> | 0.032992 | 0.086624 | 0.718240 |
| <<<256, 256>>> | 0.035808 | 0.087296 | 0.712704 |
| <<<512, 512>>> | 0.051808 | 0.097600 | 0.708576 |
| <<<1024, 1024>>> | 0.169696 | 0.220704 | 0.803872 |

На небольших данных алгоритм на CPU справляется быстрее параллельного. Но чем больше данных в тесте, тем сильнее заметна разница между временем выполнения алгоритма на CPU и на графическом процессоре.

**Выводы**

В первой лабораторной работе я познакомился с технологией cuda и попробовал реализовать и запустить простой алгоритм с использованием вычислений на графическом процессоре. Я сравнил время работы алгоритма при реализации на CPU и при реализации на различных конфигурациях графического процессора. Заметил, что на больших объемах данных алгоритм на GPU работает до 3 раз быстрее! Поэтому видеокарты все чаще используют для выполнения математических задач (нейросети, криптовалюта итд).