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: В. В. Бирюков
Группа: 8О-407Б
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: 8. Реверс вектора.

Программное и аппаратное обеспечение

Характеристики графического процессора:

- Наименование: GeForce GT 545
- Compute capability: 2.1
- Графическая память: 3150381056 Б
- Разделяемая память на блок: 49152 Б
- Количество регистров на блок: 32768
- Максимальное количество потоков на блок: (1024, 1024, 64)
- Максимальное количество блоков: (65535, 65535, 65535)
- Константная память: 65536 Б
- Количество мультипроцессоров: 3

Характеристики системы:

- Процессор: Intel(R) Core(TM) i7-3770 CPU 3.40GHz
- Память: 15 ГБ
- HDD: 500 ГБ

Программное обеспечение:

- ОС: Ubuntu 16.04.6 LTS
- IDE: Visual Studio Code 1.71
- Компилятор: nvcc 7.5.17

Метод решения

Алгоритм реверса вектора, реализованный в лабораторной работе, осуществляет разворот переданного вектора inplace, результат функции записывается в тот же вектор. Для этого каждый элемент вектора необходимо поменять местами с симметричным относительно середины вектора. Индекс симметричного элемента вычисляется по формуле: *длина вектора - индекс элемента - 1*. Очевидно, что такие замены нужны для всех элементов вектора вплоть до его середины. Сложность алгоритма по времени — $O(n)$, по памяти — $O(n)$. Каждый поток обрабатывает элемент вектора со своим индексом, затем все элементы с шагом, равным суммарному числу потоков.

Описание программы

В файле `lab1.cu` расположен основной код программы. Функция `reverse` — вычислительное ядро — осуществляет описанный выше алгоритм и принимает два аргумента: указатель на массив данных и размер массива. Файл `error_checkers.hpp` содержит вспомогательные функции и макросы для обработки ошибок CUDA.

Результаты

Конфигурация	**Время выполнения, мс**				
CPU	0.014637	0.138480	1.133743	6.507774	64.007620
1, 32	0.029933	0.118758	1.006740	9.971686	99.352540
32, 32	0.023328	0.024077	0.082394	0.738733	7.400250
64, 64	0.028032	0.024160	0.056173	0.478918	4.720102
128, 128	0.024128	0.024826	0.053414	0.478854	4.711308
256, 256	0.029030	0.025178	0.051539	0.475379	4.709266
512, 512	0.044480	0.041453	0.068550	0.465312	4.689190
1024, 1024	0.174150	0.169146	0.197504	0.562067	4.611514
Размер теста	10^3	10^4	10^5	10^6	10^7

Выводы

В ходе выполнения лабораторной работы я познакомился с основами технологии CUDA. Алгоритмы из данной лабораторной могут использоваться, например, при анализе больших объемов данных. Программирование для GPU оказалось легче чем я ожидал, в простейших случаях достаточно определить взаимно-однозначное отображение между элементами данных и потоками и реализовать обработку этих элементов.

По результатам выполнения видно, что небольшое число потоков заметно проигрывает однопоточному подходу. Вероятно это связано с тем, что графические ядра по отдельности слабее ядер CPU. При увеличении числа потоков, их преимущества становятся более заметны, оптимальной конфигурацией для данного графического процессора является 256, 256.