МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии 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: А.В. Куликов
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2020

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 7. Поэлементное вычисление модуля вектора.

Программное и аппаратное обеспечение

Видеокарта	GeForce GT 545
Compute capability	2.1
Графическая память	3004 Мб
Разделяемая память	48 Кб
Константная память	64 Кб
Количество регистров на блок	32768
Максимальное кол-во блоков	65535*65535*65535
Максимальное кол-во нитей в блоке	1024
Кол-во мультипроцессоров	3
Ядер CUDA	144

Процессор	Intel Core i7-3770
ОЗУ	16 Гб
ЖД	

Операционная система	Ubuntu 16.04.6 LTS
IDE	VS Code
Компилятор	nvcc V7.5.17

Метод решения

Пробегаем по вектору, заменяя каждый элемент его модулем.

Описание программы

В файле main.cu находится основная логика программы, включая ввод/вывод, работа с памятью, а также само ядро для вычисления на ГПУ.

В файле error.h находится функционал для обработки ошибок CUDA API и хендлеры для стандартных сигналов SIGSEGV, SIGABRT.

В файле benchmark.h находится функционал для измерения времени исполнения кода.

Поставленную задачу решает ядро kernel. В нем вычисляется индекс нити idx, на основе которого, каждой нити «поручается» обработка своих элементов массива. Если

количество элементов в массиве меньше количества запускаемых нитей, то каждой из них достанется по одному элементу, а некоторые будут простаивать. Если число элементов в массиве превышает число нитей, то некоторым из них придется обработать по нескольку элементов. Для этого и предусмотрен цикл с шагом offset. Таким образом при проходе по массиву каждый элемент заменяется своим абсолютным значением.

Результаты

Тестирование ядер с различными конфигурациями

Маленький файл: вектор из 100 чисел

---	32 блок	64 блок	128 блок	256 блок	512 блок	1024 блок
1 грид	0.025728	0.023936	0.022496	0.027552	0.023136	0.022272
32 грид	0.020320	0.020320	0.018944	0.018944	0.019712	0.021952
128 грид	0.019136	0.018720	0.019520	0.019584	0.022144	0.038720
256 грид	0.023200	0.022496	0.022592	0.023232	0.028320	0.052256
512 грид	0.024608	0.024224	0.024160	0.026432	0.041312	0.088544
1024 грид	0.030720	0.030496	0.047168	0.036320	0.064832	0.159392

Лучший результат: 0.018720 мс при параметрах <128, 64>

Средний файл: вектор из 10000 чисел

---	32 блок	64 блок	128 блок	256 блок	512 блок	1024 блок
1 грид	0.118080	0.069408	0.045696	0.032576	0.026400	0.023136
32 грид	0.030976	0.026720	0.022240	0.021280	0.021856	0.026080
128 грид	0.022496	0.021088	0.021568	0.021984	0.027488	0.037472
256 грид	0.022912	0.023872	0.033408	0.029408	0.034432	0.058816
512 грид	0.028384	0.026880	0.028416	0.030688	0.042048	0.089344
1024 грид	0.034784	0.040096	0.036096	0.046208	0.067392	0.159968

Лучший результат: 0.021088 мс при параметрах <128, 64>

Большой файл: вектор из 1000000 чисел

---	32 блок	64 блок	128 блок	256 блок	512 блок	1024 блок
1 грид	17.54214	8.773216	4.412256	2.234112	1.122944	0.586176
32 грид	1.115392	0.585088	0.345568	0.250016	0.253696	0.280416
128 грид	0.849600	0.457152	0.288128	0.256288	0.250976	0.289920
256 грид	0.783392	0.425280	0.275392	0.248800	0.247904	0.330624
512 грид	0.783584	0.424736	0.275488	0.246176	0.250368	0.385440
1024 грид	0.774400	0.423744	0.274144	0.247616	0.265184	0.450912

Лучший результат: 0.246176 мс при параметрах <512, 256>

Сравнение с CPU
Маленький файл: вектор из 100 чисел
Результат: 0.000509 мс

Средний файл: вектор из 10000 чисел
Результат: 0.005086 мс

Большой файл: вектор из 1000000 чисел
Результат: 0.304705 мс

Выводы

Данный алгоритм может быть использован в качестве подсобного в более сложной программе.

Особых проблем в решении задачи не возникало. Как видно из результатов тестирования, при обработке незначительного объема данных особого смысла прибегать к использованию GPU нет, но при возрастании объема обрабатываемых данных GPU начинает превосходить CPU. То, что это происходит не сразу, вероятно, объясняется оверхедом на создание нитей на GPU.