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: Днепров И.С.
Группа: 8О-407Б
Преподаватели: Крашенинников К.Г.,
Морозов А.Ю.

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

Device: GeForce GT 545
Размер глобальной памяти: 3150381056
Размер константной памяти : 65536
Размер разделяемой памяти: 49152
Регистров на блок: 32768
Максимум потоков на блок: 1024
Количество мультипроцессоров : 3
OS: Linux Ubuntu 16.04.6
Редактор: Atom

Метод решения

Поэлементно пробежать по вектору и домножить отрицательные числа на -1.

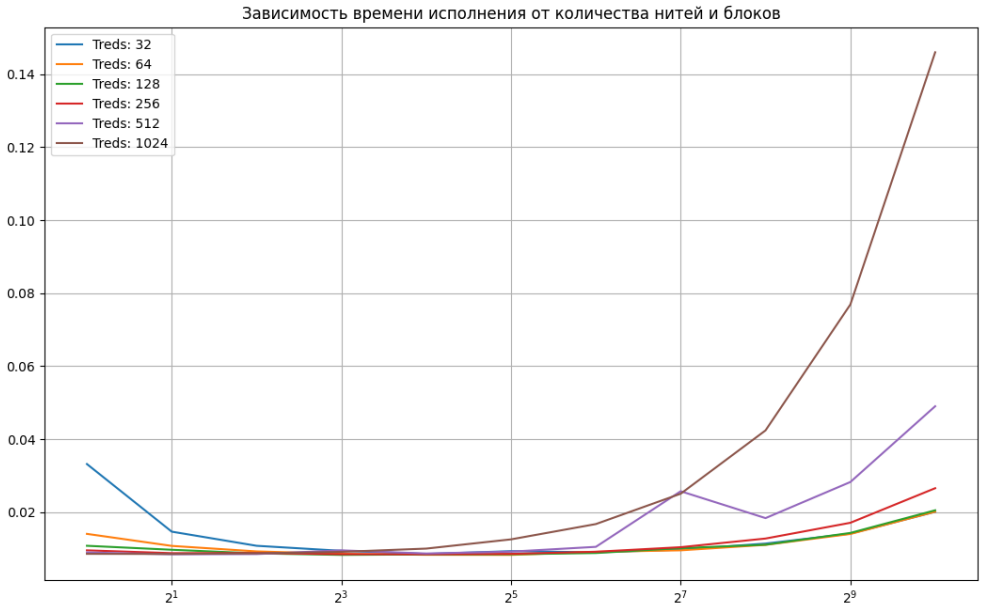
Описание программы

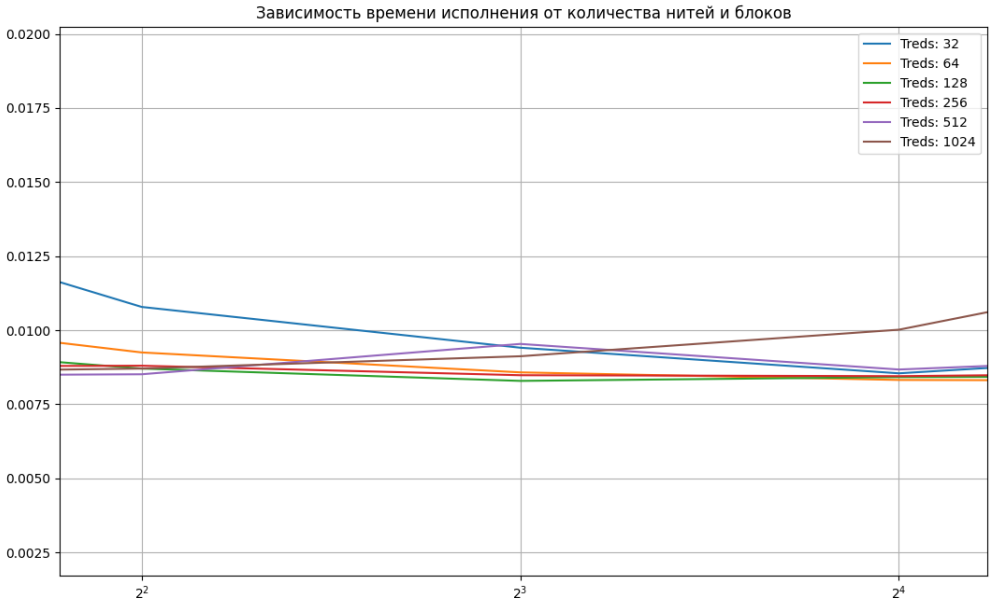
Чтобы найти поэлементный модуль вектора на GPU, нужно этот вектор скопировать в память GPU, после чего отрицательные числа до множить на -1 и записать их на то же место в векторе. Потом вектор можно скопировать в основную память и вывести.

Сравнение времени исполнения

Я решил сравнить время работы в зависимости от количества нитей и блоков, чтобы понять, какой вариант оптимален для моего случая. Сравнения проводились на тестовом наборе из 1024 чисел.

На CPU результат был получен за 0.017669, что более чем в два раза медленнее лучшего результата.





Видно, что для разного количества нитей, оптимальное количество блоков разное и находится оно в пределах 4 и 16, и чем больше нитей, тем меньше блоков требуется для достижения максимальной скорости, что логично. Но после определенного числа блоков, затраты на распараллеливание превышают выхлоп от этой операции. И видно, что конфигурация 1024 на 1024 работает гораздо медленнее, чем 1 на 32.
В моем случае оптимальный результат составил 0.008288 и был достигнут сразу на двух конфигурациях <<<64, 32 >>>, <<<128, 8>>>. Что-то странное происходит в конфигурации <<<128, 512>>>, похоже на то, что в это время на кмпьютере происходили еще какие-т о вычисления, которые повлияли на время выполнения.

Выводы

Задание было простым, но довольно интересно было его выполнять и анализировать скорость выполнения в зависимости от разного числа поток и блоков.