МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Лабораторная работа №*1*
по курсу «*Программирование графических процессоров»*

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

Выполнил: *Филиппов Владимир Михайлович*
Группа: *М8О-410Б-22*
Преподаватели: А.Ю. Морозов,
Е.Е. Заяц

Москва, *2025*

Условие

1. Ознакомление и установка программного обеспечения для работы с программно-аппаратной архитектурой параллельных вычислений (CUDA). Реализация одной из примитивных операций над векторами.
2. Поэлементное нахождение максимума векторов.

Программное и аппаратное обеспечение

GPU.

- Compute Capability: 8.9
- Графическая память (Global memory): 8 GB GDDR6 (8188 MiB — видно в nvidia-smi)
- Shared memory per SM (мультипроцессор): до 100 KB (может конфигурироваться: 64 KB shared + 32 KB L1 или наоборот)
- Shared memory per block: до 48 KB
- Constant memory: 64 KB (аппаратно выделено)
- Регистры на потоковый мультипроцессор (SM): 65,536
- Максимум регистров на блок: 32K (зависит от конфигурации)
- Максимальное число потоков на блок: 1024
- Максимальное число блоков в сетке (grid): 2^31 - 1 по X, 2^16 - 1 по Y и Z
- Количество мультипроцессоров (SM): 24 (у RTX 4060)
- Макс. потоков на SM: 2048
- Итого потоков на GPU: 24 × 2048 = 49,152 одновременно резидентных

CPU.

- Архитектура. x86_64
- Модель. AMD Ryzen 5 3600 6-Core Processor
- Количество физических ядер. 6
- Потоков. 12 (2 потока на ядро)
- Частота (BogoMIPS). 7200.05
- Кэш L1 (данные/инструкции). 192 KiB / 192 KiB
- Кэш L2. 3 MiB
- Кэш L3. 16 MiB

RAM.

- Общий объём. 15 GiB
- Swap. 4 GiB (свободен)

SSD.

- Объем. 1 TB
- Использовано. 156 GB

ПО.

- OS. Windows 11 + WSL 2.0.
- IDE. Visual studio code.
- NVCC. 11.5 (Build V11.5.119)

Метод решения

Вычисляем индекс и смещение, согласно которым каждый поток будет обрабатывать свою часть массива. Далее для каждого элемента массива потоки будут находить максимум из двух векторов.

Описание программы

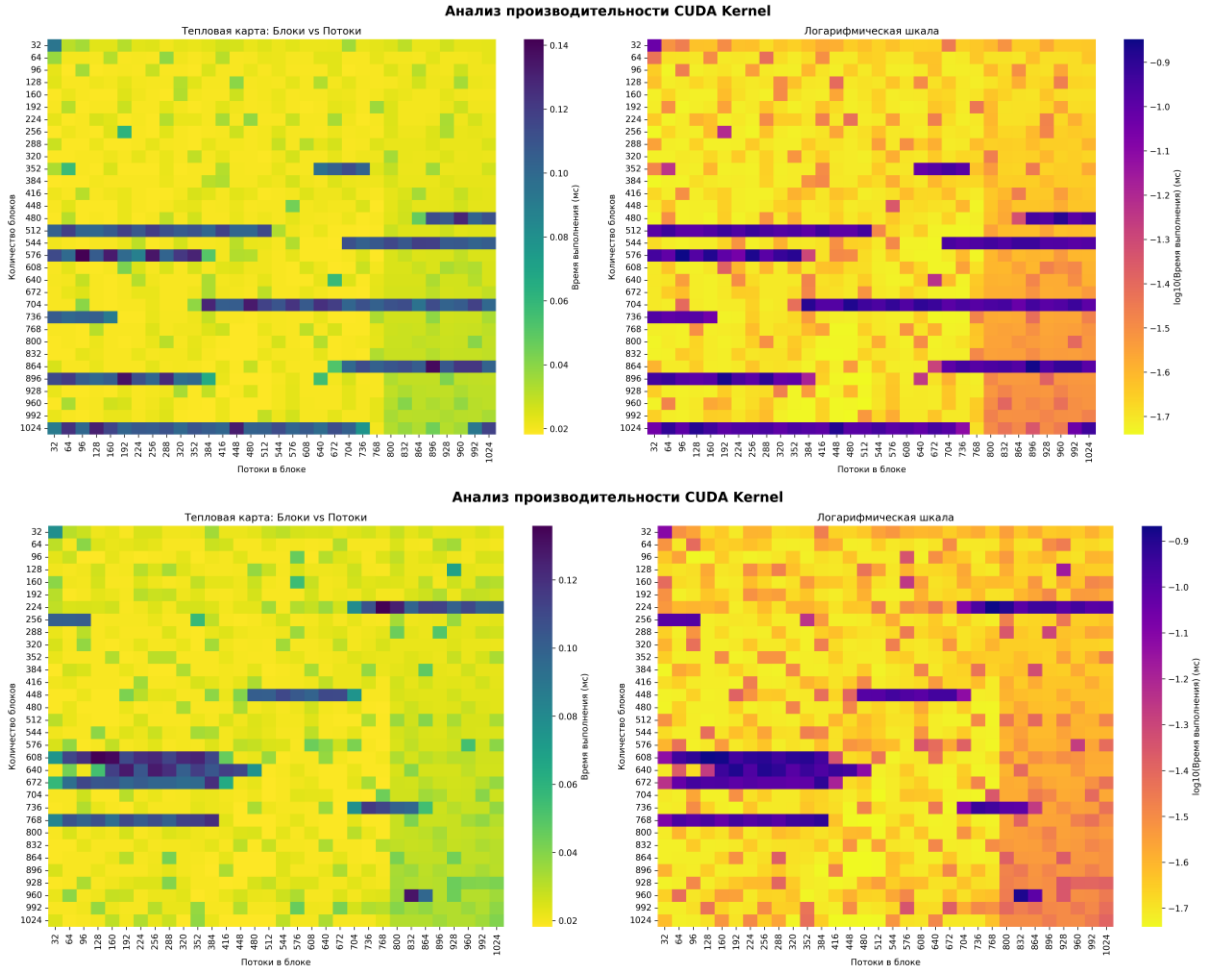
Программа написана в одном файле main.cu. Используется ядро для нахождения при помощи GPU поэлементного максимума двух массивов.
Разделение по файлам, описание основных типов данных и функций. Обязательно описать реализованные ядра.

Результаты

В рамках тестирования, я написал CPU реализацию заданной функции и сравнил время выполнения двух вариаций программ. Для GPU я запускал для каждого количества блоков и потоков по 20 тестов, чтобы усреднить результат. После чего я выбирал минимальное среднее время выполнения. Время в таблице представлено в миллисекундах. Не вооруженным взглядом видно, что программа на GPU работает намного быстрее однопоточной реализации на CPU.

	CPU	GPU
1000	0,017836	0,003736
100000	1,78813	0,005043
1000000	17,3159	0,018571
10000000	212,356	0,991584

Также я попробовал усредненное тестирование для разных конфигураций ядра.
Получил вот такие heatmapы.



Вообще довольно интересные картинки получились. Сразу бросается в глаза, что периодически случаются выбросы из обычного выполнения, у которых по странному стечению обстоятельств какое-то очень короткое время выполнения. Поскольку это происходит «линиями», будем считать, что в какой-то момент нагрузка на GPU со стороны сторонних приложений падает и производительность резко возрастает.
Другой интересной особенностью является то, что начиная с 800 потоков в блоке, производительность начинает расти очень резко, затухая «к верху» при уменьшении количества самих блоков.

Выводы

Реализованный алгоритм может быть использован, например, в компьютерной графике или зрении, для объединения двух изображений или применения маски.
Но на самом деле, больше было интересно погрузиться в CUDA и в мир исполнения программ на GPU. Для меня это ценный опыт, который я буду использовать в будущем.