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: Лобанов О. А.
Группа: 8О-408Б-20
Преподаватель: А.Ю. Морозов

Москва, 2023

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 4. Поэлементное нахождение минимума векторов.

Программное и аппаратное обеспечение

Device: Tesla T4
Compute capability: 7.5
Total constant memory: 65536
Registers per block: 65536
Max threads per block: 1024
Multiprocessors count: 40
OS: Ubuntu 22.04.2 LTS
Redactor: colab google

Метод решения

Для нахождения поэлементного минимума двух векторов достаточно вызвать количество нитей равное размеру массивов и записать в качестве результата 2-ух соответствующих элементов массива по идентификатору в третий.

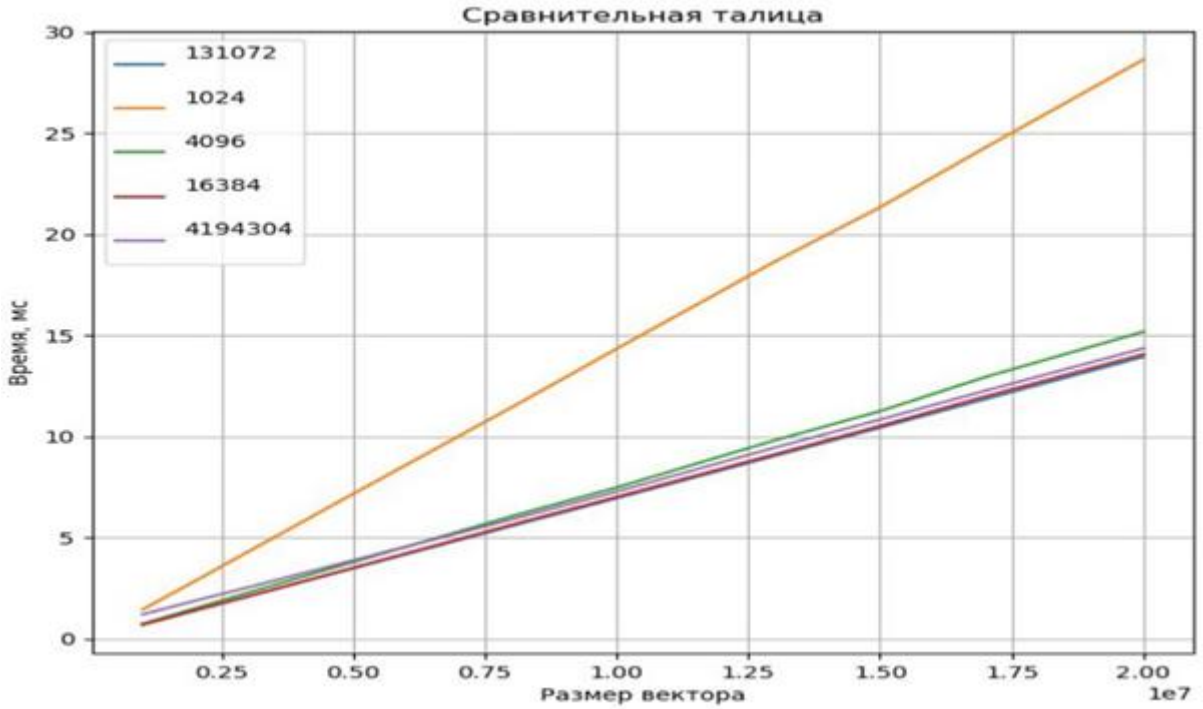
Описание программы

Для того, чтобы выполнить мой вариант я выделил 3 блока памяти на device. Первый - для 1-го вектора, второй для 2-го вектора и третий для результирующего. После этого, с помощью команды cudaMemcpy, я скопировал данные векторов в созданные массивы. После работы kernel я вернул результат в результирующий вектор с той же самой командой cudaMemcpy.
В kernel я вычисляю абсолютный индекс, который будет индексом в массиве. Далее выполняется операция нахождения минимума и запись его в третий массив:

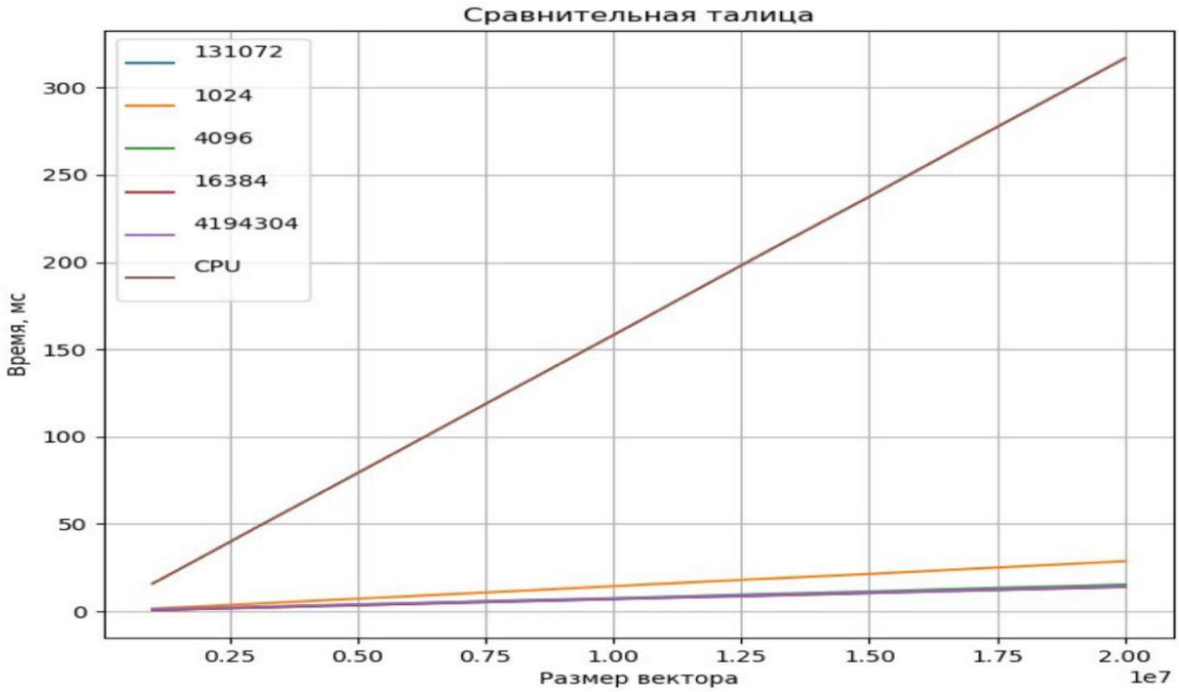
```
__global__ void kernel(double *array1, double *array2, double *result, int size_array) {
    int absolute_index = blockIdx.x * blockDim.x + threadIdx.x;
    int offset = gridDim.x * blockDim.x;
    while (absolute_index < size_array) {
        result[absolute_index] = array1[absolute_index] < array2[absolute_index] ? array1[absolute_index] : array2[absolute_index];
        absolute_index += offset;
    }
}
```

Результаты

Здесь видно, что при количестве потоков 1024, запуск получился успешнее на GPU чем на CPU.



GPU



CPU

Выводы

В ходе выполнения столкнулся с проблемой, что у меня не имеется видеокарты Nvidia. Поэтому я выполнил данную работу в Google Colab, где предоставляют карту Tesla T4. Выполнив данную работу, я убедился, что GPU дает большое преимущество над CPU.