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: Г.Н. Хренов
Группа: 8О-407Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2021

Условие

1. Цель работы: ознакомление и установка программного обеспечения для работы с программно-аппаратной архитектурой параллельных вычислений(CUDA). Реализация одной из примитивных операций над векторами.

2. Вариант 6: поэлементное возведение в квадрат вектора.
   Входные данные: на первой строке задано число n - размер векторов. Наследующей строке записано n вещественных чисел n элементы вектора. Выходные данные: необходимо вывести n чисел - результат поэлементного возведения в квадрат исходного вектора.

Программное и аппаратное обеспечение

GPU name: NVIDIA GeForce RTX 2060
compute capability 7:5
totalGlobalMem: 6442450944
sharedMemPerBlock: 49152
totalConstMem: 65536
regsPerBlock: 65536
maxThreadsDim: 1024 1024 64
maxGridSize: 2147483647 65535 65535
multiProcessorCount: 30
CPU name: AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx
MaxClockSpeed: 2300
NumberOfCourse: 4
RAM: 8
SSD: 256, HDD: 1024
OS: Windows10
Compiler: nvcc

Метод решения

Передаем входной вектор с CPU на GPU, там выполняем вычисления, применяя распараллеливание, затем передаем вектор обратно на CPU.

Описание программы

kernel.cu:

__global__ void kernel(double* arr, int n): выполняет возведение элементов вектора в квадрат. Каждый поток считает элементы с шагом, равным общему количеству потоков.

int main():

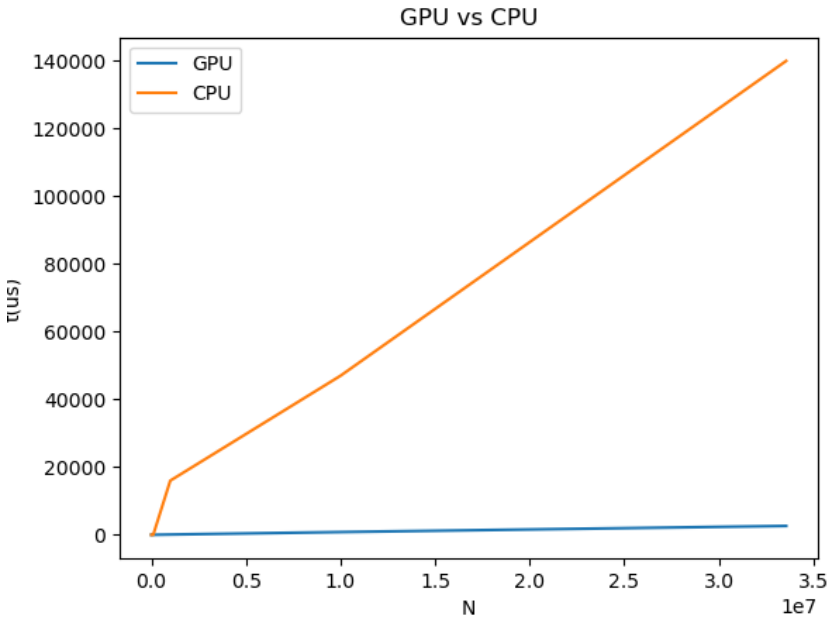
cudaMemcpy(dev_arr, arr, sizeof(double) * n, cudaMemcpyHostToDevice): копирование вектора размера n из arr в dev_arr с CPU(host) на GPU(device).

kernel<<<256,256>>>(dev_arr, n): вызов ядра с 256 блоками по 256 потока в каждом.

Результаты

	<<< 1, 32 >>>	<<< 64, 128 >>>	<<< 256, 512 >>>	<<< 1024, 1024 >>>
1000	13.184	3.616	4.896	20.488
10000	90.144	4.064	5.216	20.736
100000	863.81	7.744	7.968	24.228
1000000	13468	88.48	58.24	70.401
10000000	132450	821.19	555.81	536.887
33554431(max)	340130	2629.4	1858.8	1796.7

(в таблице указано время работы ядер в us)



Выводы

При решении задач, в которых можно распараллелить вычисления, производительность GPU во много раз превосходит CPU, поэтому сейчас GPU имеет множества областей применения – везде, где нужны высокопроизводительные вычисления. Как показала лабораторная работа, для каждой задачи необходимо правильно подбирать параметры ядра: число блоков и число потоков в этих блоках, так как принцип “чем больше, тем лучше” не всегда оптимален.