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.

Выполнила: Е.И. Усачева
Группа: 8О-407Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2020

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 задания:** 7. Поэлементное вычисление модуля вектора.

Программное и аппаратное обеспечение

Лабораторная работа выполнялась в среде Google Collab с использованием встроенного аппаратного ускорителя GPU в облаке.

Свойства GPU:

Название:	Tesla T4
Compute capability:	7.5
Глобальная память:	15812263936
Разделяемая память (на блок):	49152
Константная память:	65536
Количество регистров на блок:	65536
Максимальное количество блоков:	2147483647*65535*65535
Максимальное количество нитей:	1024*1024*64
Количество мультипроцессоров:	40

nvcc: NVIDIA (R) Cuda compiler driver, release 9.2, V9.2.88

Процессор:	1,8 GHz 2−ядерный процессор Intel Core i5
Оперативная память:	8 ГБ 1600 MHz DDR3
Жесткий диск:	SSD-накопитель PCIe
OS:	macOS Catalina v10.15.4

Метод решения

Программа принимает на вход вектор, состоящий из *N* элементов. С помощью спецификатора функции *__global__* запускаем ядро из CPU, выполняется оно на GPU. Т.к. вектор по условию задачи одномерный, будем использовать только x-измерение. Рассчитав предварительно расположение нити и смещения, приступаем к вычислению модуля каждого элемента вектора с помощью функции *abs()*. Важно отметить, что

встроенная переменная *threadIdx* и её поле x позволяют задать соответствие между расчетом элемента вектора и нитью в блоке. То есть происходит расчет каждого элемента вектора в отдельной нити.

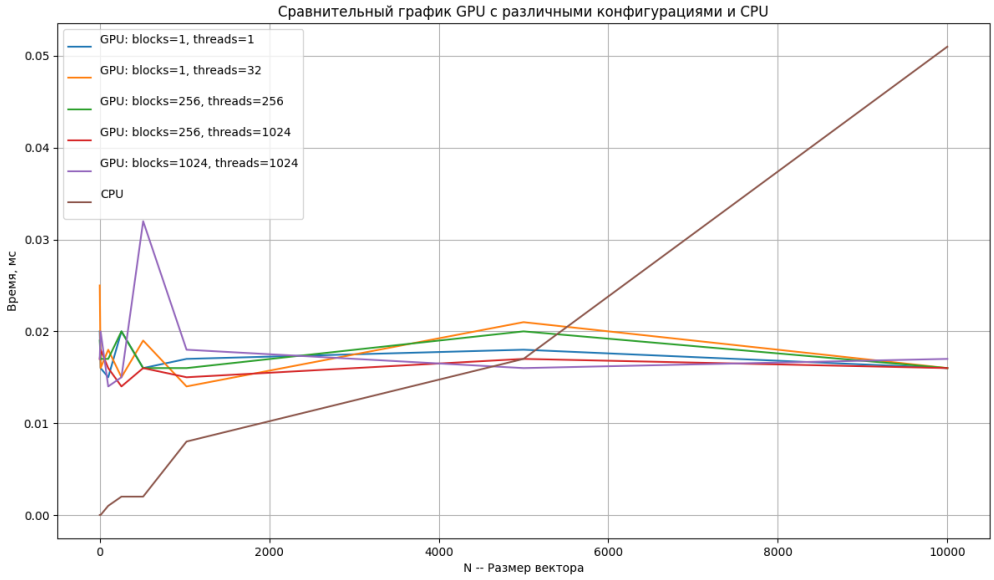
Хороший цикл статей про CUDA и устройство GPU: habr

Описание программы

Программа принимает на вход через потоковый ввод целое число *N* - размер вектора. Затем выделяем память на CPU для вектора типа *double* размера *N* и заполняем его так же через потоковый ввод. Далее создаем копию этого вектора и выделяем память под него на GPU с помощью функции *cudaMalloc()*. Копируем данные из исходного вектора в копию на GPU через функцию *cudaMemcpy()*.
Запускаем ядро. Для этого вызываем функцию *kernel()* (в моем случае) со спецификатором вызова функции *__global__*, который определяет, что функция вызывается из CPU, а выполняется на GPU. В качестве спецификаторов запуска ядра я указала *<<<gridSize=256, blockSize=256>>>* (на этапе написания кода лабораторной работы я просто взяла пример с семинара). Важно отметить, что x-размерность блока на данном GPU составляет 1024. Это значит, что за один раз возможно вычислить модуль каждого элемента вектора, если его размер *N <= 1024* элементов.
Затем, уже после запуска ядра, в функции *kernel()* рассчитываем *id* текущей нити *index* и смещение *offset*. В цикле запускаем расчет модуля элемента вектора. Таким образом получается, что на расчет каждого элемента вектора выделяется отдельная нить.
Когда расчеты получены, копируем обратно данные из GPU вектора в вектор CPU с помощью функции *cudaMemcpy()* и освобождаем используемую на GPU память функцией *cudaFree()*. Выводим получившийся вектор на печать. Освобождаем с помощью *free()* память на CPU.

Результаты

На графике ниже изображена зависимость время исполнения расчетов от размера вектора для GPU с различными конфигурациями и CPU. Для центрального процессора зависимость, можно сказать, практически линейная. На примере с GPU видно небольшие «выбросы», однако в целом можно заметить, что для *N=0* и *N=10000* время выполнения примерно одинаковое. Следовательно, мы наблюдаем константную зависимость.
Однако важно заметить, что для векторов маленькой размерности (кол-во элементов) целесообразнее использовать CPU. Это видно из того же графика. К тому же, не потребуется время для копирования входных-выходных данных.



Выводы

В данной лабораторной работе я познакомилась с технологией CUDA и устройством GPU. Также я написала простейшую программу для вычисления поэлементного модуля вектора с использованием вычислительных мощностей графического процессора.
Данный способ задействования возможностей GPU оказался очень эффективным и отлично подходит не только для 3D-моделирования и игровой графики, но и для сложных математических вычислений.
Сложностей в программировании данная задача у меня не вызвала. Однако, были проблемы с настройкой облачного сервиса Google Collab, которые решились с помощью статей по данной теме из интернета.