МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Лабораторная работа № 1
по курсу «Программирование графических процессов»

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

Выполнил: И.В. Сектименко
Группа: М8О-410Б-22
Преподаватели: А.Ю. Морозов,
Е.Е. Заяц

Москва, 2025

Условие

Кратко описывается задача:

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 1. Сложение векторов.
   Входные данные. На первой строке задано число n – размер векторов. В следующих 2-ух строках записано по n вещественных чисел – элементы векторов. Ограничение: $n < 2^{25}$.
   Выходные данные. Необходимо вывести n чисел – результат сложения исходных векторов. Все результаты выводить с относительной точностью 10^{-10}.

Программное и аппаратное обеспечение

Характеристики:

— графического процессора:

compute capability:	7.5
графическая память:	15828320256
разделяемая память на блок:	49152
константная память:	65536
количество регистров на блок:	65536
максимальное количество блоков:	(2147483647, 65535, 65535)
максимальное количество нитей на блок:	(1024, 1024, 64)
количество мультипроцессоров:	40

— процессора: Intel(R) Xeon(R) CPU @ 2.00GHz;

— оперативной памяти: 12Гб общей памяти;

— жесткого диска: 256 Гб общей памяти.

Во время выполнения работы использовалась IDE Google Colab. В нее встроена ОС Ubuntu 22.04.4 LTS с видеокартой NVIDIA-SMI 550.54.15.

Метод решения

Задача достаточно тривиальная: необходимо пройтись по двум массивам и поэлементно их сложить. Результат я решила сохранить в отдельный третий массив. Однако при решении необходимо применить технологию CUDA. Так сначала надо перекопировать данные из памяти процессора в память графического процессора, распараллелить программу, чтобы нагрузка на все потоки была равномерной и результат вычислялся быстрее, после чего полученный массив суммы двух массивов необходимо обратно перекопировать из памяти графического процессора в память центрального.

Описание программы

Программа состоит из 1 файла.

Сначала считывается n – количество чисел в массивах. Затем резервируется память под два массива с вещественными числами, и считываются сами массивы. Аллоцируется память на графическом процессоре под этих два массива, и они туда перекопируются. Вычисляется программа-ядро на графическом процессоре. Резервируется память под

результат на центральном процессоре, и туда перекопируется полученный на графическом процессоре результат. Этот результат выводится в стандартный поток вывода. В конце все резервируемые ресурсы освобождаются.
Рассмотрим подробнее описание реализованного ядра.
Там происходит процесс сложения двух векторов поэлементно и записи полученного результата в третий. Важно отметить, что каждый поток выполняет какую-то свою часть задачи. Для этого характеристики – абсолютный порядковый номер и общее количество потоков. Он выполняет работу в массиве с индексами кратными его порядковому номеру.

Результаты

Ниже приведена сравнительная таблица. В спроках – программа выполняется ядрами с разными конфигурациями или на центральном процессоре, в столбцах – разные объемы данных.

	100 элем.	100 000 элем.	10 000 000 элем.
<<<1, 32>>>	0,1 мс	1,55 мс	194,13 мс
<<<1, 256>>>	0,098 мс	0,29 мс	27,05 мс
<<<1, 1024>>>	0,096 мс	0,21 мс	10,38 мс
<<<32, 32>>>	0,096 мс	0,15 мс	6,88 мс
<<<32, 256>>>	0,095 мс	0,12 мс	1,41 мс
<<<32, 1024>>>	0,094 мс	0,1 мс	1,15 мс
<<<256, 32>>>	0,095 мс	0,12 мс	1,4 мс
<<<256, 256>>>	0,093 мс	0,1 мс	1,11 мс
<<<256, 1024>>>	0,095 мс	0,12 мс	1,05 мс
<<<1024, 32>>>	0,094 мс	0,11 мс	1,18 мс
<<<1024, 256>>>	0,095 мс	0,12 мс	1,17 мс
<<<1024, 1024>>>	0,12 мс	0,12 мс	1,15 мс
CPU	0,002 мс	1,11 мс	124,65 мс

Выводы

Данное задание является учебным, обучающим, поэтому чаще всего его можно встретить в школьных заданиях или вузовских лабораторных работах. Особых проблем не возникло. Большую часть времени заняло не выполнение самого задания, а составление отчета.
В ходе составления отчета было выявлено несколько тенденций. Во-первых, на маленьком объеме данных не выгодно использовать GPU, ведь на выделение блоков и нитей и перекопирование данных в память графического процессора тоже уходит время. Во-вторых, есть некоторая «точка», после которой увеличивать конфигурацию ядра бессмысленно, более того это приводит к проигрышу по времени на несколько сотых миллисекунд. В-третьих, по мере приближения к самой оптимальной конфигурации оптимизация происходит по экспоненте. В-четвертых, небольшое количество потоков хоть и распараллеливает программу и, казалось бы, должно оптимизировать ее время работы, но проигрывает по времени выполнению той же самой программы на центральном процессоре.