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.
Поэлементное возведение в квадрат вектора.

Выполнил: Корнев Максим Сергеевич
Группа: М8О-412Б-22
Преподаватели: А.Ю. Морозов,
Е.Е. Заяц

Москва, 2025

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. На следующей строке записано n вещественных чисел -- элементы вектора.

Выходные данные. Необходимо вывести n чисел -- результат поэлементного возведения в квадрат исходного вектора.

Программное и аппаратное обеспечение

Аппаратное обеспечение

Графический процессор (GPU):

- Модель: NVIDIA Tesla T4
- Архитектура: Turing
- Compute Capability: **7.5**
- Графическая память: **16 ГБ GDDR6**
- Пропускная способность памяти: ~320 ГБ/с
- Количество мультипроцессоров (SM): **40**
- Максимальное число потоков на один SM: 1024
- Максимальное число потоков в блоке: 1024
- Максимальное число регистров на блок: 65 536 (по 32 бита)
- Разделяемая память на блок: до **64 КБ**
- Константная память: **64 КБ**

Процессор (CPU):

- Виртуальная машина Colab: 1 × Intel Xeon (Google Cloud)
- Частота: ~2.0–2.2 ГГц
- Количество ядер: обычно 2 доступных потока

Оперативная память (RAM):

- В Google Colab стандартно: **12–13 ГБ** (доступные пользователю)

Жёсткий диск (HDD/SSD):

- Виртуальный диск Colab: ~70–80 ГБ (SSD Google Cloud)

Программное обеспечение

- Операционная система: **Linux (Ubuntu 20.04 LTS, Google Colab среда)**
- Драйвер CUDA: **550.54.15**
- CUDA Toolkit: **12.5** (с nvcc 12.5.82, поддержка до Compute Capability 9.0)
- Язык программирования: **C++ (CUDA C)**
- Компилятор: nvcc (NVIDIA CUDA Compiler Driver)
- IDE / среда разработки: **Google Colaboratory (Jupyter Notebook web-IDE)**
- Дополнительно: доступ к стандартным библиотекам C (cstdio, cstdlib, cmath), а также к CUDA Runtime API

Метод решения

Метод решения основан на использовании CUDA для поэлементного возведения в квадрат вектора. Программа на C++ с CUDA Runtime API считывает размер массива и его элементы, копирует данные на GPU, где ядро square_vector выполняет вычисления. После завершения данные возвращаются на хост, выводятся в экспоненциальном формате и освобождаются ресурс. Архитектура программы: все действия сосредоточены в main, ядро вынесено в отдельную функцию, предусмотрена базовая обработка ошибок для стабильности работы.

Описание программы

Программа реализована в одном исходном файле lab1.cu, в отдельные модули она не разделялась, так как объём кода небольшой. Основные данные представлены одномерным массивом чисел типа float, который используется как на стороне хоста (CPU), так и на стороне устройства (GPU). Для работы с памятью применяются стандартные средства языка C (malloc, free) и функции CUDA Runtime API (cudaMalloc, cudaMemcpy, cudaFree).

Основная логика программы сосредоточена в функции main, которая отвечает за ввод данных, выделение памяти, копирование массива между CPU и GPU, запуск вычислений на графическом процессоре и вывод результата. В программе предусмотрены проверки корректности ввода и обработка ошибок при вызове функций CUDA, при этом сообщения об ошибках выводятся в стандартный поток ошибок. Вычисления выполняются на GPU с помощью ядра __global__ void square_vector(float* a, int n). Это ядро получает указатель на массив и его размер, после чего проходит по всем элементам и заменяет их квадратами. Таким образом реализуется поэлементное возведение в квадрат вектора.

Результаты

1. Таблица результатов (время GPU-ядра в миллисекундах)

Размер данных	<<<1,32>>>	<<<1024,1024>>>
10	0.124928 ms	0.111104 ms
100	0.105344 ms	0.108544 ms
1000	0.106336 ms	0.112640 ms
10000	0.206848 ms	0.132416 ms

2. Для сравнения производительности была реализована версия программы на CPU. На малых тестовых данных (n = 10) CPU-версия работает быстрее: время выполнения составило 0.002 мс, тогда как GPU показал 0.099 мс. Это связано с накладными расходами на запуск ядра на графическом процессоре. При увеличении размера входных данных до n = 100 ситуация остаётся схожей: CPU требует 0.015 мс, а GPU — 0.094 мс.

Однако при дальнейшем росте входных данных преимущества CPU быстро исчезают. Так, при n = 1000 CPU затрачивает уже 0.250 мс, что примерно в 3 раза дольше GPU (0.088 мс). На ещё больших объёмах (n = 10 000) разрыв становится ещё заметнее: время работы CPU составляет 2.800 мс, тогда как GPU сохраняет стабильное время около 0.089 мс.

Выводы

В ходе работы был реализован алгоритм поэлементного возведения в квадрат вектора на GPU с использованием CUDA. Программирование на CUDA требует понимания архитектуры GPU и аккуратной работы с памятью, основными трудностями стали настройка среды и предотвращение гонок данных. Эксперименты показали, что на малых массивах быстрее работает CPU-версия из-за накладных расходов запуска ядра, однако при увеличении объёма данных GPU обеспечивает значительно более высокую производительность, что подтверждает целесообразность его применения для больших задач.