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»

Выполнил: *Ермаков Ярослав Валерьевич*
Группа: *М8О-407Б-22*
Преподаватели: А.Ю. Морозов,
Е.Е. Заяц

Москва, *2025*

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 8. Реверс вектора.

Программное и аппаратное обеспечение

Аппаратная и программная конфигурация использовалась на облачных серверах Google Collab. Использованная для выполнения программа включала графический ускоритель NVIDIA Tesla T4 с объемом видеопамяти 15360 МБ. Ошибки памяти ECC отсутствовали. Драйвер графического ускорителя имел версию 550.54.15, а установленная версия CUDA составляла 12.4.

Характеристики CUDA-устройства (Tesla T4):

1. Computecapability: 7.5 (sm_75)
2. Количество SM: 40
3. Размер варпа: 32
4. Макс. потоков на блок: 1024
5. Макс. потоков на SM: 1024
6. Shared memory на блок: 49 152 байт
7. Shared memory на SM: 65 536 байт
8. Регистров на блок: 65 536
9. Макс. размеры сетки (grid): 2 147 483 647 × 65 535 × 65 535
10. Макс. размеры блока (block dims): 1024 × 1024 × 64
11. Компиляция и модель запуска:
12. Компилятор: nvcc 12.4
13. Ключи сборки:-O3-arch=sm_75

Распределение вычислений: сетка потоков, данные параметров классов размещаются в константной памяти устройства.

Метод решения

Задача: реализовать реверс вектора — то есть перестановку его элементов в обратном порядке.

Алгоритм в общих чертах:

1. На CPU (хосте) читаются входные данные: число элементов и сам вектор. Память под массив выделяется как на хосте, так и на GPU (девайсе).
2. Вектор копируется из хоста в память GPU.
3. Запускается ядро CUDA (reverse_kernel), где каждому потоку назначается свой индекс i.
    a. Для индекса i вычисляется элемент out[i] = in[n - 1 - i].
    b. Чтобы покрыть вектор любой длины при фиксированном числе потоков, используется grid-stride loop — каждый поток с шагом stride = blockDim.x * gridDim.x обрабатывает несколько элементов.
4. После завершения вычислений результат копируется обратно на хост.

5. На CPU результат выводится в требуемом формате (%.10e).

Описание программы

Программа реализует реверс элементов вектора вещественных чисел типа double с использованием технологии CUDA. Ввод осуществляется из стандартного потока: сначала задаётся количество элементов n, затем последовательность чисел. После считывания входных данных выполняется выделение памяти на GPU, копирование входного массива, запуск CUDA-ядра reverse_kernel, в котором каждый поток переставляет один или несколько элементов массива в обратном порядке с использованием grid-stride цикла. Результаты возвращаются в оперативную память CPU и выводятся в стандартный поток вывода в формате научной нотации с десятью знаками после запятой.

Основные компоненты программы: макрос CSC для контроля ошибок вызовов CUDA API, ядро reverse_kernel для параллельного реверса массива, а также функция main, обеспечивающая управление вводом/выводом, взаимодействие CPU и GPU и освобождение ресурсов. Программа предназначена для эффективной обработки больших массивов данных за счёт параллельных вычислений на графическом процессоре.

Результаты

Для оценки производительности были проведены эксперименты с различными размерами входного вектора (n) и разными конфигурациями CUDA-сетки. Время измерялось с помощью событий CUDA (cudaEventElapsedTime) и функции clock() для CPU. Единицы измерения — миллисекунды (ms).

Сравнение времени работы CUDA-ядра при разных конфигурациях <<<grid,block>>>

Тестовый вектор: n = 10^7 элементов (double).

Конфигурация ядер	Время GPU (ms)
<<<1, 32>>>	15.4
<<<1, 256>>>	5.8
<<<32, 256>>>	1.2
<<<128, 512>>>	0.9
<<<1024, 1024>>>	0.7

Вывод: при маленьком числе потоков GPU используется неэффективно; при увеличении числа блоков и потоков достигается почти линейное ускорение, после чего время стабилизируется (достигается предел пропускной способности памяти).

Сравнение CPU и GPU

Реверс вектора был реализован и на CPU (однопоточная версия без CUDA).

Размер вектора n	CPU (ms)	GPU (<<<128, 512>>>) (ms)	Ускорение
(10^5)	0.6	0.3	2.0×
(10^6)	6.1	0.7	8.7×
(10^7)	61.2	0.9	68.0×

Вывод: при малых размерах массива выигрыш от GPU невелик, так как значительное время занимает передача данных между CPU и GPU. Однако при увеличении размера массива достигается значительное ускорение (десятки раз).

Выводы

В ходе выполнения лабораторной работы №1 была реализована программа реверса элементов вектора на CUDA. Реверс является базовой операцией, применимой в алгоритмах обработки массивов, структур данных и потоков информации. Эксперименты показали, что при малых размерах данных ускорение GPU минимально из-за накладных расходов, однако на больших входных массивах использование CUDA даёт значительный прирост производительности.