МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки 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: И.А. Мариничев
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников
А.Ю. Морозов

Москва, 2022

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

Compute capability	:	2.1
Name	:	GeForce GT 545
Total Global Memory	:	3150381056
Shared memory per block	:	49152
Registers per block	:	32768
Warp size	:	32
Max threads per block	:	(1024, 1024, 64)
Max block	:	(65535, 65535, 65535)
Total constant memory	:	65536
Multiprocessors count	:	3
Processor	:	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
RAM	:	16 GB
Drive	:	349G
OS	:	Ubuntu 16.04.6 LTS
IDE	:	Visual Studio Code
Compiler	:	NVIDIA (R) Cuda compiler driver V7.5.17

Метод решения

Считываем вектор (CPU). Копируем наш исходный вектор на CUDA и создаем там же еще один вектор, куда будем сохранять реверсированный вектор. В ядре каждый поток будет в idx-ый элемент реверс вектора ставить (n - 1 - idx)-ый элемент исходного вектора, если потоков меньше, чем размер вектора, то каждый поток будет еще делать шаг, равный числу потоков, где n – размер вектора, idx – абсолютный номер потока. Затем копируем реверс вектор в исходный и выводим результат.

Описание программы

```
// выполняет обращение вектора
__global__ void parallel_reverse(double *vec, double *vec_reverse, int n)
```

Результаты

Конфигурация	Размер теста (время указано в ms)					
	O(10^2)	O(10^3)	O(10^4)	O(10^5)	O(10^6)	O(10^7)
<<< 1, 32 >>>	0.034464	0.065632	0.635392	6.264896	62.582527	624.777039
<<< 32, 32 >>>	0.031200	0.035584	0.057088	0.436224	4.209984	42.059040
<<< 32, 256 >>>	0.034176	0.029504	0.032896	0.185888	1.844608	19.482719
<<< 256, 256 >>>	0.038144	0.036992	0.033728	0.176416	1.658688	18.557600
<<< 256, 512 >>>	0.041120	0.041824	0.039168	0.176416	1.632256	17.948032
<<< 512, 512 >>>	0.050752	0.051712	0.048672	0.177632	1.611936	17.269823
<<< 512, 1024 >>>	0.097248	0.097600	0.096832	0.244640	1.686848	16.509727
<<< 1024, 1024 >>>	0.164256	0.164640	0.164928	0.310208	1.742656	16.433216
CPU	0.0	3.0	25.0	173.0	2629.0	31244.0

Выводы

Данный алгоритм может найти применение в численных методах, в которых присутствует независимая работа со строчками/столбцами или возможно при операции отражения изображений, если опять же рассматривать вектор как часть матрицы. При работе на CPU данный алгоритм имеет сложность O(n), это видно и по таблице. Один варп также имеет линейную сложность, но уже на два порядка ниже. После 32x32 потоков время перестает значительно падать и приобретает один порядок и постепенно уменьшается с увеличением потоков, это больше всего заметно на предельной длине вектора.