Assignment #5: Report

Zhen Guan (202191382, zguan@mun.ca)

July 27, 2022

1 Introduction

This report compares the GPGPU implementation to the original implementation for the previous assignment (Assignment # 4 PGM image filter) in terms of both the code design and structure as well as the speed of execution.

My Implementation is as followed:

	Framework	Language	Subject
Assignment #5	CUDA	C++	PGM Image Filter
Assignment #4	MPI	C	PGM Image Filter

2 Code & Data Structure

I have 3 parts of code in both implementation: the **Convolution Module, PGM IO Module, and main logics**. The CUDA version has some more modules including **Timing Module**.

2.1 Convolution Module

The biggest difference between the CUDA version and the MPI version is on their convolution modules. Although the convolution algorithm does not change, they have different parallel implementations: the CUDA version uses **block**s and **thread**s, while the MPI version uses multi-process + socket. The calculation unit is also different: the CUDA version uses the independent GPU hardware from NVIDIA, while the MPI version uses CPU:

	Compute Unit	Communication Method
CUDA	GPU	Video Memory
MPI	CPU	Shared Memory + Socket

Because of these differences, the way to use memory is different:

2.1.1 MPI

See next page.

```
    // The 0 process (rank=0) allocates memory by
    MPI_Win_allocate_shared(...);

    // The other processes (rank!=0) claims the memory by
    MPI_Win_shared_query(...);
```

2.1.2 CUDA

```
    cudaMalloc(...);

    // And cudaMemcpy is also required to
    // copy the data from the host to the device:
    cudaMemcpy(..., cudaMemcpyHostToDevice);

    // or vice versa:
    cudaMemcpy(..., cudaMemcpyDeviceToHost);
```

2.2 PGM IO Module

The PGM IO module of CUDA version is the same as the MPI version.

2.3 Main Logics

The design of the data structure of convolution kernel is slightly different from the MPI version. The CUDA version uses 1D array to store the kernel, while the MPI version uses 2D:

```
    // MPI:
    kernel[x][y];

    // CUDA:
    kernel[x * width + y];
```

This is because "cudaMemcpy" is to copy flat arrays. Because the convolution kernel is a configurable matrix, I am not putting this difference in the "Convolution Module" section.

2.4 Timing Module

I don't have a timing method in MPI version since at that time I didn't saw the content of Assgiment #5. The timing method in CUDA version is as follows:

```
    float time_milliseconds;
    cudaEvent_t start;
    cudaEvent_t stop;

    cudaEventCreate(&start);
```

```
    cudaEventCreate(&stop);
    cudaEventRecord(start);

    timeConsumingCode();

    cudaEventRecord(stop);
    cudaEventSynchronize(stop);
    cudaEventElapsedTime(&time, start, stop);
    cudaEventDestroy(start);
    cudaEventDestroy(stop);

    printf("Time elapsed: %.0fms\n", time);
```

To measure the time cost of the MPI version, I will modify the MPI version and use timespec structure that comes from time.h.

3 Speed of Execution

I have ran 10 times for both the CUDA and MPI version on a 256x256 PGM image.

Image Size: 256x256	**CUDA (1 block 16 threads)** NVIDIA RTX 3060 Laptop 130W	**MPI (16 workers)** Apple M1 Pro 31W
1	15 ms	1,018 ms
2	14 ms	1,018 ms
3	16 ms	1,019 ms
4	14 ms	1,021 ms
5	14 ms	1,018 ms
6	15 ms	1,019 ms
7	17 ms	1,017 ms
8	14 ms	1,018 ms
9	14 ms	1,017 ms
10	15 ms	1,018 ms

Image Size: 256x256	**CUDA (1 block 1 thread)** NVIDIA RTX 3060 Laptop 130W	**MPI (1 worker)** Apple M1 Pro 31W
1	131 ms	1,024 ms
2	132 ms	1,024 ms
3	132 ms	1,024 ms
4	131 ms	1,024 ms
5	131 ms	1,024 ms
6	131 ms	1,024 ms
7	131 ms	1,023 ms
8	131 ms	1,023 ms
9	131 ms	1,024 ms
10	131 ms	1,024 ms

Since the data size is so small, it can not show the advantages of muliti-threading and multi-processing.

To show the advantage of parallel compute, I created a 30000x30000 PGM image, first run on multi-thread, then single-thread.

Image Size: 30000x30000	**CUDA (1 block 900 thread)** NVIDIA RTX 3060 Laptop 130W	**MPI (25 worker)** Apple M1 Pro 31W
1	21,069 ms	32,847 ms

The single-thread result:

Image Size: 30000x30000	**CUDA (1 block 1 thread)** NVIDIA RTX 3060 Laptop 130W	**MPI (1 worker)** Apple M1 Pro 31W
1	**Couldn't finish in reasonable time**	252,100 ms (4 min 12 sec)

The single-thread performance of GPU is even worse than the CPU, this is as expected. The power of GPU is not single SP's performance, but the ability to keep a big amount of threads running simultaneously.

4 Problems Encountered

In Windows platform please don't generate a very big PGM image with the ".pgm" suffix, otherwise the explorer will try to preview it. This stucked my hard drive and I had to unmount the disk partition to remove that huge PGM file. (Killing the explorer process didn't work)

Changing the suffix ".pgm" to something else solved this problem.