Introduction to CUDA Parallel Programing Homework 2: Result and Discussion

Kuan-Chia Chiu
Department of Physics, National Taiwan University, Taipei 10617, Taiwan

In HW2, I will show the result for my code finding the maximum element in a vector, which is based on the example "vecDot.cu" from the folder "/work/cuda_lecture/vecDotProduct". The original results "Output.txt "are shown in the my folder "hw2".

RESULT

The Output file is attached in the folder "hw2". I only shows three sets of results here:

```
------------------------------------
Find maximun value of vector:
Set GPU with device ID = 0
Enter the size of the vector: 40960007
Enter the number (2^m) of threads per block: 1024
Enter the number of blocks per grid: 64
Input time for GPU: 30.703680 (ms)
Processing time for GPU: 0.401536 (ms)
GPU Gflops: 204.016615
Output time for GPU: 6.793600 (ms)
Total time for GPU: 37.898819 (ms)
Processing time for CPU: 64.174461 (ms)
CPU Gflops: 1.276520
Speed up of GPU = 1.693310
Check result:
h_CPU_Max =1.000000000000000e+00
h_GPU_Max =9.999773502349854e-01
The difference =2.264976501464844e-05
------------------------------------
------------------------------------
Find maximun value of vector:
Set GPU with device ID = 0
Enter the size of the vector: 40960007
Enter the number (2^m) of threads per block: 1024
Enter the number of blocks per grid: 10
Input time for GPU: 30.631071 (ms)
Processing time for GPU: 0.398304 (ms)
GPU Gflops: 205.672090
Output time for GPU: 6.785984 (ms)
Total time for GPU: 37.815361 (ms)
Processing time for CPU: 63.905823 (ms)
CPU Gflops: 1.281887
Speed up of GPU = 1.689943
Check result:
h_CPU_Max =1.000000000000000e+00
h_GPU_Max =9.999693632125854e-01
The difference =3.063678741455078e-05
------------------------------------
------------------------------------
Find maximun value of vector:
Set GPU with device ID = 0
Enter the size of the vector: 40960007
Enter the number (2^m) of threads per block: 1024
Enter the number of blocks per grid: 4
Input time for GPU: 30.512512 (ms)
Processing time for GPU: 0.929824 (ms)
GPU Gflops: 88.102709
Output time for GPU: 6.793696 (ms)
Total time for GPU: 38.236034 (ms)
Processing time for CPU: 63.987392 (ms)
CPU Gflops: 1.280252
Speed up of GPU = 1.673484
Check result:
h_CPU_Max =9.999997615814209e-01
h_GPU_Max =9.986919164657593e-01
The difference =1.307845115661621e-03
------------------------------------
```

In these results, I kept the block size fixed and tried to tune the grid size(from 64 to 4). Next, I changed the block size(512, 256, 128, 64). **It is noticed that the max values found by GPU and CPU are not really the same. The difference is around** $10^{-5} \sim 10^{-3}$**.**

DISCUSSION

From the result, I made some summary here:

As Fig 1 shown, there are some local minimum at (Block size **1024**, Grid size **16**) and (Block size **512**, Grid size **16**). Except these two points, the GPU Gflops is increasing as grid size and block size.

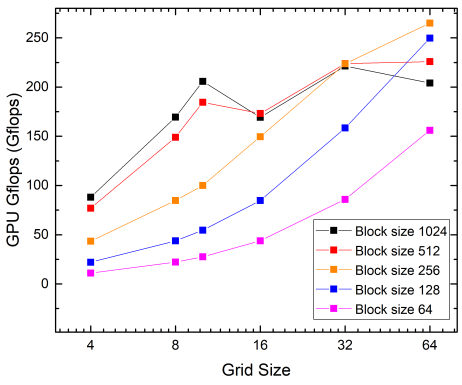

FIG. 1: GPU Gflops for different block and grid sizes.

See Fig 2: However, the CPU Gflops is much smaller than GPU's. Of course, it would not change much when I changed the block and grid size.

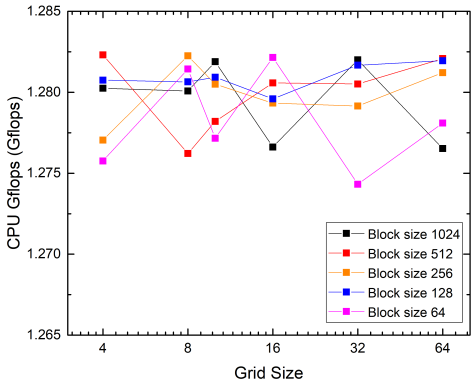

FIG. 2: CPU Gflops for different block and grid sizes.

From Fig 3, the input time of loading the data to GPU is not much increasing as the grid size increasing except (Block size **512**, Grid size **64**).

On the other hand (See Fig 4), the output time for GPU is varying between 4.5 ms to 7 ms. The range is larger than input time some times.

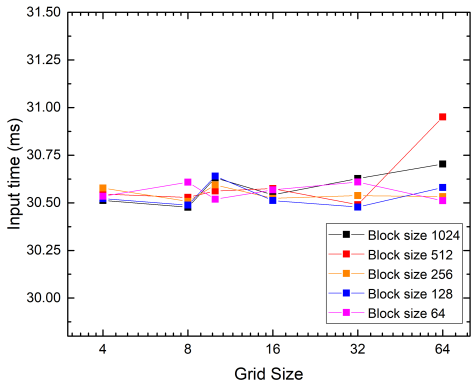

FIG. 3: Input time for GPU

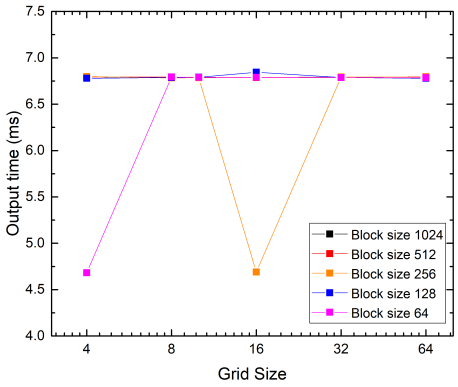

FIG. 4: Output time for GPU

Check the total time (See Fig 5), we could find there is a local minimum (Block size **256**, Grid size **16**) in this range of block and grid size.

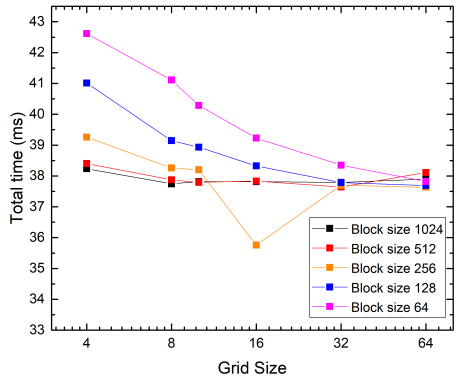

FIG. 5: Total time for GPU

In the end (See Fig 6), we found the values of speed up for GPU are consistent from the result of total time for GPU (which should be inverse of total time).

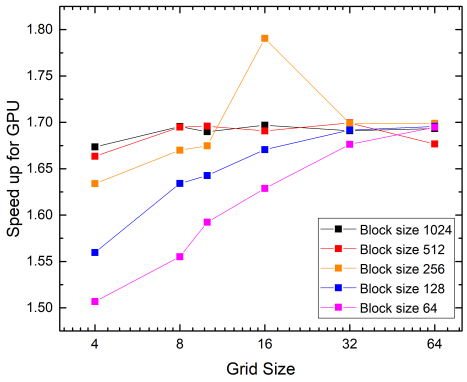

FIG. 6: Speed up for GPU