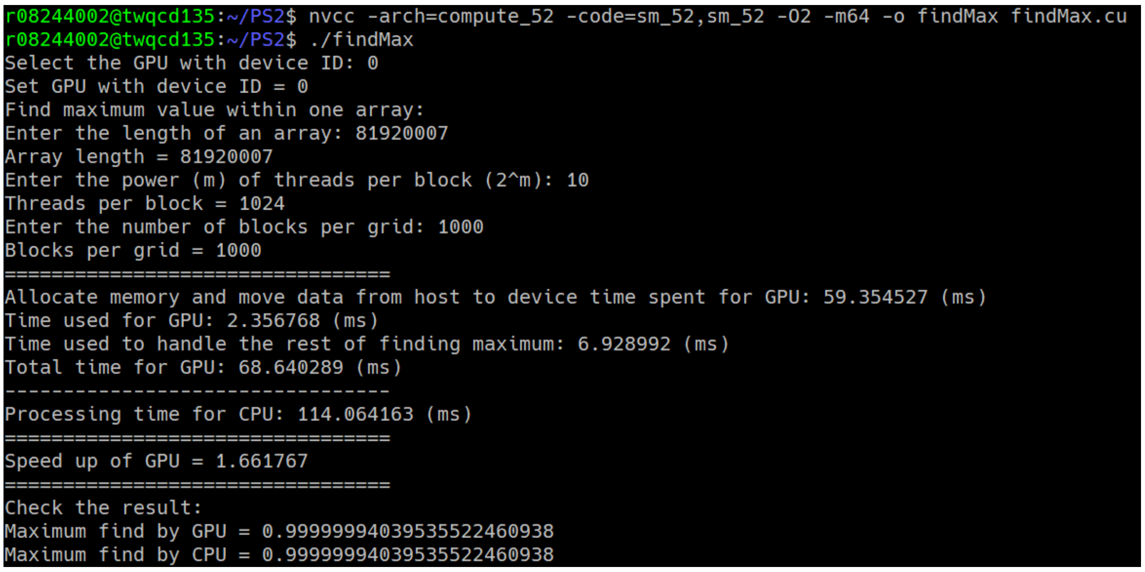
CUDA Parallel Programming Problem Set2

R08244002 Shin-Rong, Tsai

- **Finding maximum value in an array**

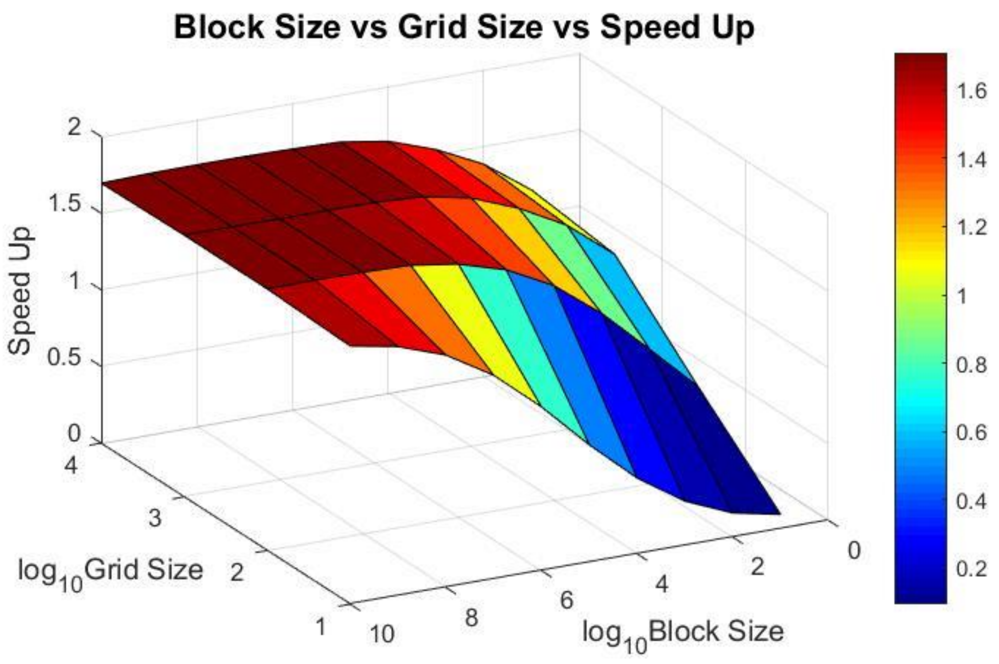
I. Result

Slightly change the code from *vecAdd.cu*, and save it as *findMax.cu*. Which gives:



```
r08244002@twqcd135:~/PS2$ nvcc -arch=compute_52 -code=sm_52,sm_52 -O2 -m64 -o findMax findMax.cu
r08244002@twqcd135:~/PS2$ ./findMax
Select the GPU with device ID: 0
Set GPU with device ID = 0
Find maximum value within one array:
Enter the length of an array: 81920007
Array length = 81920007
Enter the power (m) of threads per block (2^m): 10
Threads per block = 1024
Enter the number of blocks per grid: 1000
Blocks per grid = 1000
=================================
Allocate memory and move data from host to device time spent for GPU: 59.354527 (ms)
Time used for GPU: 2.356768 (ms)
Time used to handle the rest of finding maximum: 6.928992 (ms)
Total time for GPU: 68.640289 (ms)
---------------------------------
Processing time for CPU: 114.064163 (ms)
=================================
Speed up of GPU = 1.661767
=================================
Check the result:
Maximum find by GPU = 0.999999940395355224609375
Maximum find by CPU = 0.999999940395355224609375
```

In order to optimize the block size and grid size, loop through block size from $2^1 \sim 2^{10}$ and grid size from $10^1 \sim 10^4$, save the code in *findMax_Optimize.cu*. Then plot the output with MATLAB.



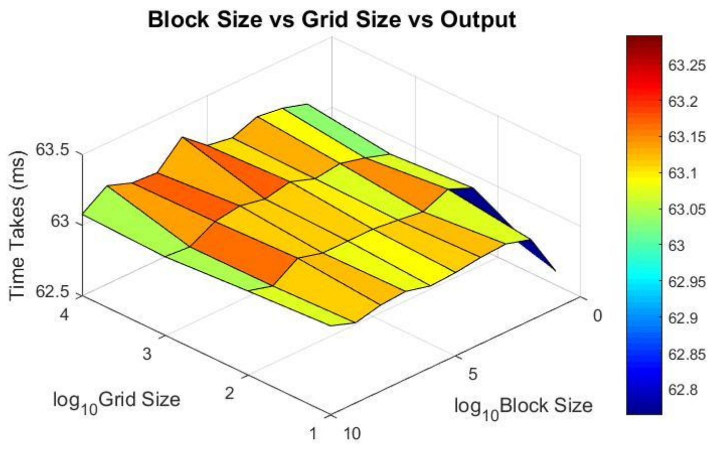
II. Discussion

1. From the figure, we can see that GPU with proper block size and grid size greatly speed up the process. The bigger the grid size and block size are, the more it speeds up.

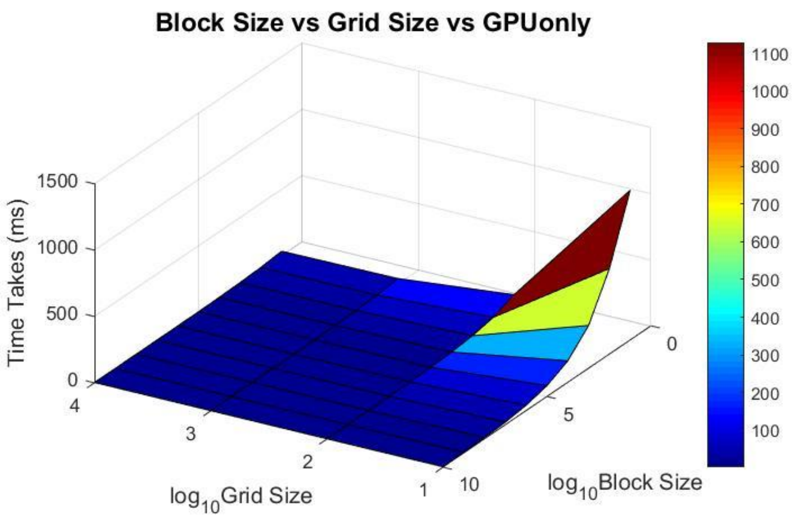
   There are some causes that affect this result, discuss in later section:

↑ : the more ↓ : the less		↑ : faster ↓ : slower	
Grid size	↓	Output	↑
Block size	↑	Parallel reduction	↓
Block size × Grid size	↑	Loop through all the array	↑

2. From the point of view of moving data between host and device, the input data is basically the same for all of them, but the output data length is depending on the grid size. The bigger the grid size is, the slower it is.
   We can see that it slightly tilts according the grid size, but it doesn't affect the efficiency that much.



3. From the point of view of processing parallel reduction, the bigger the block size is, the more time it takes with the same grid size. But from the point of view of looping through all the elements in the array, the more the threads (block size × grid size) are, the less time it needs to loop through, which means faster.
   Since looping through all the elements require accessing the global memory, and parallel reduction requires accessing the shared memory, the looping time dominant the results.
   So we can see that with the same grid size, the more the block size is, the less time it takes.


Block Size vs Grid Size vs GPUonly
Time Takes (ms)
1500
1000
500
0
4
3
2
1
$\log_{10}$Grid Size
0
5
10
$\log_{10}$Block Size
1100
1000
900
800
700
600
500
400
300
200
100