Kevin Bastoul

EE-451

04/11/17

PHW#4 Report

Running instructions:

1. Login to hpc
2. source /usr/usc/cuda/5.5/setup.sh
3. go to directory
4. nvcc –o p1 p1.cu (for p1)
5. nvcc –o p2 p2.cu (for p2)
6. qsub p1queue.pbs (for p1)
7. qsub p2queue.pbs (for q2)

Approach 1 – Non-optimized

Execution times:

* b=32 -> 202768138.000000 ns
* b=16 -> 129837857.000000 ns

Discussion:

I planned on executing b=8, 16, & 32 but the delay for waiting on HPC jobs (5-6 hours/submission) made me run out of time. From the two values that I was able to run it appears that b=32 more efficient than b=16. If this trend continues than b=32 may be optimal. From some of the reading I’ve done, it sounds like choosing the optimal block/grid size can be a very complex and researched topic. Based on the specific problem and hardware present there will be a different “sweet spot” of block/grid size, but figuring this out beyond experimentation is difficult. From this small test it seems like this problems sweet spot is close to 1024 threads/block

The maximum value b can be set to is 32 because the maximum threads per block supported by nvcc CUDA is 1024 (32\*32). Any b more than this will result in too many threads per block. In addition, with a grid size defined as 1024/b, a b greater than 1024 would result in a fractional grid size which is not supported.

Approach 1 -

Execution times:

* b=32 -> 13085101.000000 ns
* b=16 -> 19261175.000000 ns

Discussion:

I planned on executing b=8, 16, & 32 but the delay for waiting on HPC jobs (5-6 hours/submission) made me run out of time. Again, the “sweet spot” seems to be around 1024 threads/block. The maximum value of b is again 32 for the reasons described in pt1.

Note:

-I’ve included the p2 output files but not the p1 because I don’t have them in my directory currently and it would take too many hours to run again before the deadline