CS 575 Project 5

Jeremy Dao

May 2020

1. The following results were gathered using a machine with a Nvidia Titan X Pascal GPU.
2. The following table displays the performance data for varying number of block sizes and number of trials. Note that all performance values are in units of MegaTrials per Second.

Number of Trials \ Block Size	16384	32768	65536	131072	262144	524288	1048576
16	1454.55	1340.31	1536.38	1697.47	1724.27	1959.57	1903.68
32	2285.71	1996.10	2585.86	2682.38	3112.46	3493.39	3239.55
64	2000.00	1906.89	2314.12	2620.60	3196.25	3472.66	3248.22
128	2000.00	1868.61	2354.02	2974.58	2943.59	3305.23	3157.75

The following graphs displays the performance data from the table above. The first graph shows block size versus performance for various number of trials. The second graphs is the inverse; it shows number of trials versus performance for various block sizes.

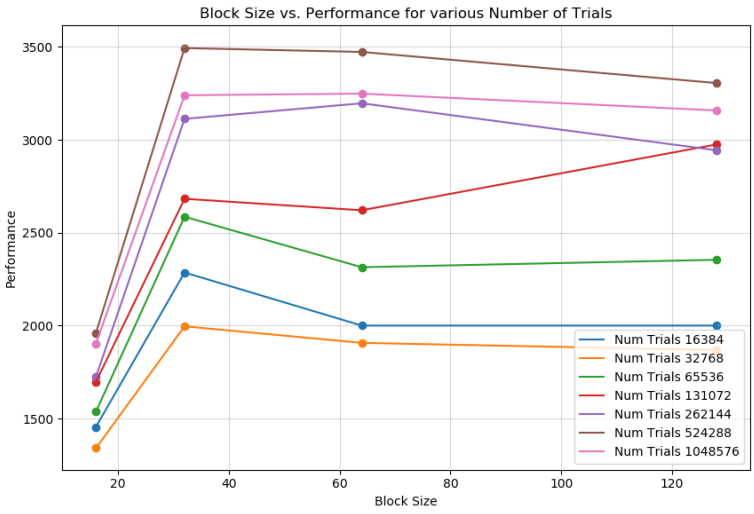

Figure 1: Block size vs. Array size for various number of trials

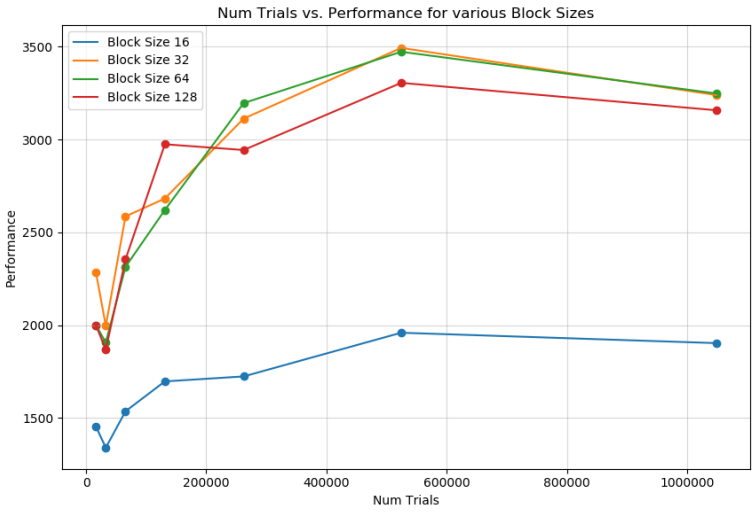

Figure 2: Number of Trials vs. Array size for various block sizes

3. We can see from the figure 1 that is general more trials results in higher performance. However, this only happens up to a limit. The largest number of trials tried (1024*1024) had lower performance than the next largest number of trials (512*1024). This could have happened because 1024*1024 trials was too much data for the GPU to handle, resulting in slowdown. Furthermore we can also see that increasing block size results in higher performance, up to a point. This makes sense, since more threads in each block means that there is more computation happening in parallel. However, there is a limit to how much performance can be gained from increasing the block size. As seen in figure 2 having a block size of 128 resulted in worse performance than other block sizes for almost all number of trials. This happens because there are too many threads in each block, resulting in too few blocks, so there are less blocks to switch between whenever a block's execution stops. This may cause execution to occasionally go idle, slowing down performance.

4. Looking at figure 2 we can see an expecially large gap in performance between block size of 16 and all other block sizes. In other words, a block size of 16 does significantly worse than all the other block sizes. This is because in CUDA each streaming multiprocessor (SM) always executes 32 threads at a time. So if each block only has 16 threads, the SM will still execute 32 threads, but only 16 of them will be meaningful data that we will use, essentially throwing away half of the computation done.

5. We do see somewhat similar results as in project 1 where we used CPU parallelism. In both cases, having too little data results in worse performance no matter the number of CPUs/block size. However, here we also see that increasing the block size does not always result in an increase in performance, unlike how in project 1 increasing the number of CPUs always resulted in an increase in performance (given that the data is too small). This is due to the difference in how threads in a GPU work versus multiple threads in a CPU. As explained in question 3, too many threads results in less blocks which can cause non-optimal switching between blocks in the streaming multiprocessor, causing computation to occasionally go idle. This is different from a CPU where each thread is sort of like it's own streaming multiprocessor, so increasing the number of threads does not result in more blocking of computation.

6. This means for proper GPU parallel computing, one must carefully choose the correct block size to use, which will depend on the size of the data. The size of the data will also contribute to how many blocks there are, so if the is a small amount of data a smaller block size might work better so there are enough blocks for the SM to switch between (though make sure that the block size is some multiple of 32 to avoid the warp issue discussed in 4). Furthermore, the proper block size will also depend on the specs of the GPU being used, which will dictate how many active blocks and threads each SM can have.