Gregory Douglas Stula
CS 475
2020-05-20

Project 05

Hardware

This experiment was ran on DGX server which has
- 16 Nvidia Tesla V100 GPUs,
- 28TB of SSD storage,
- Two 24 core Intel Xeon 8166 Platinum CPUs @ 2.7Ghz
- 1.5TB of DDR4 2666 Memory

Experiment

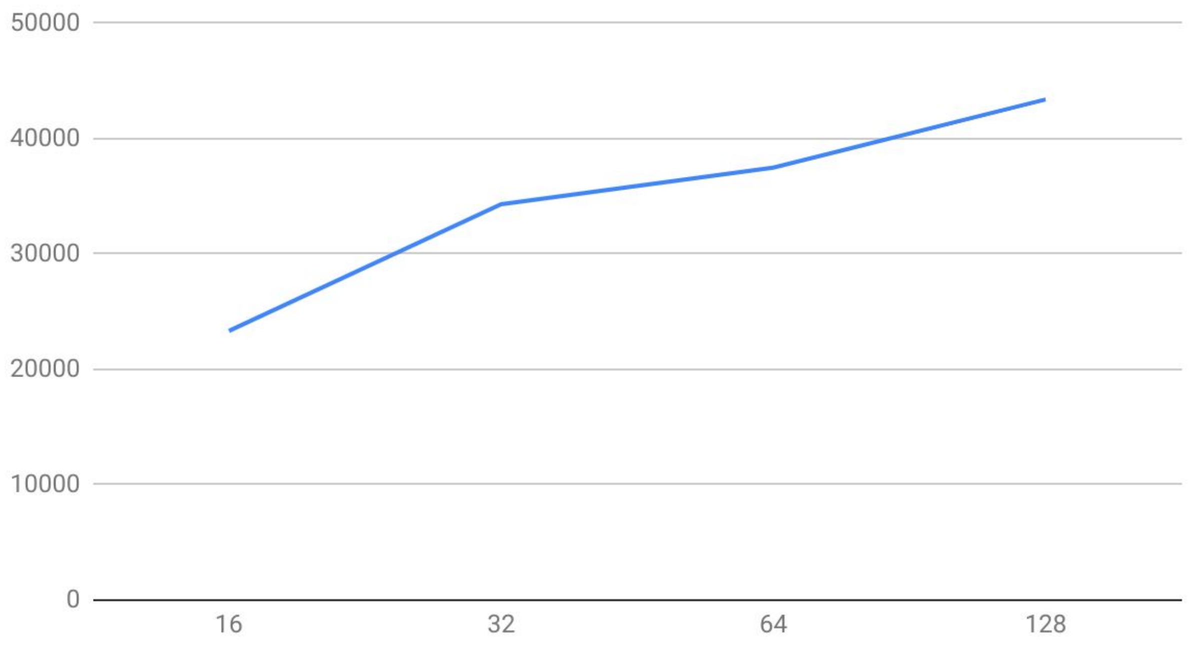
The test was to run a Monte Carlo experiment regarding a laser hitting a randomly changing sized plate in a C++/CUDA program compiled with nvidia cuda compiler. The number of trials used were 16*1024, 32*1024, 64*1024, 128*1024, 256*1024, 512*1024, and 1000*1024 times. This was done on a single thread, on the DGX server. Performance was measured in Mega Trials per Second (MT/s). The different block sizes used were 16 ,32, 64, and 128. The experiment was run partially within the C++/CUDA program and with a python3 script which redirects the output of the C++ program to a CSV file which was uploaded to google drive for analysis in Google sheets.
The block sizes were added manually to the google sheet after the upload. This was simple to do as they were in intervals of 16, 32, 64, and 128.

The python3 program can be called with ./run.py which is how it can be tested on the rabbit server. The included bash script can be submitted to slurm for running the experiment on the DGX server.

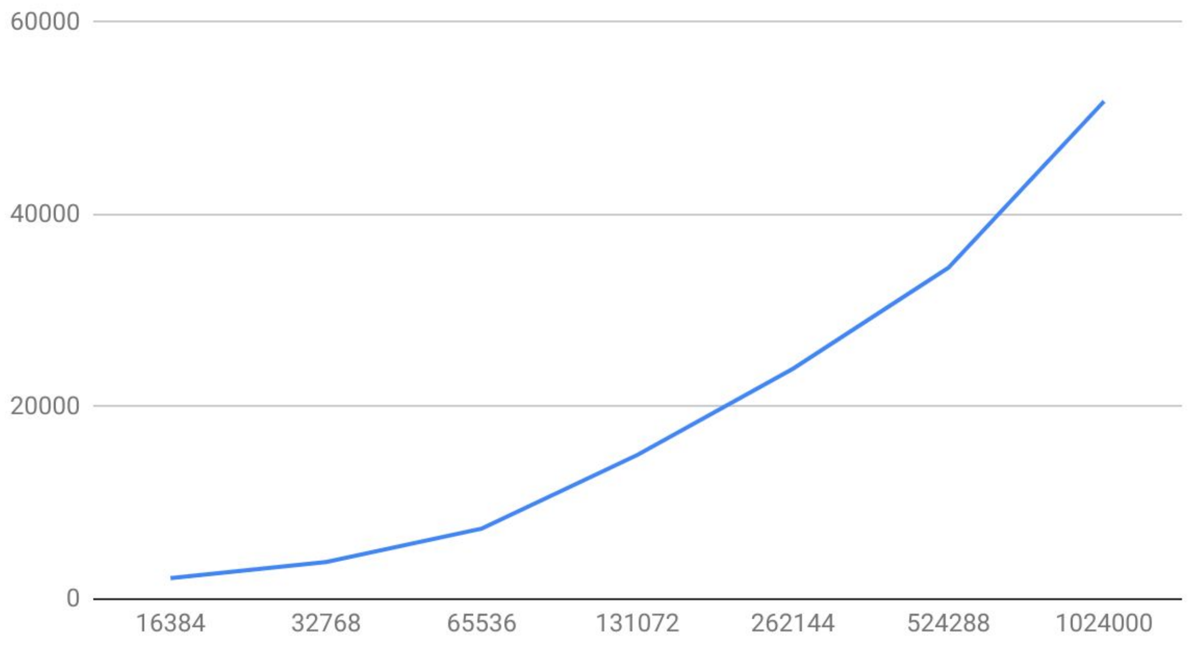
Results

number of trials	Block Size	megatrials per second	probability
16384	16	444.444429	42.51709
16384	32	571.428564	41.387939
16384	64	571.428564	42.041016
16384	128	592.592592	41.662598
32768	16	966.037701	41.85791
32768	32	1041.709017	42.01355
32768	64	1084.74573	41.412354
32768	128	751.83553	41.555786
65536	16	1597.503921	42.604065
65536	32	1740.016995	42.063904
65536	64	1954.198571	41.720581
65536	128	2035.785335	41.854858
131072	16	2959.537691	41.74118
131072	32	4063.492096	42.338562
131072	64	3938.461528	42.075348
131072	128	4031.495853	41.98761
262144	16	4366.73782	41.960526
262144	32	5927.640903	41.984558
262144	64	6787.075428	41.928482
262144	128	6866.722527	42.105103
524288	16	6173.323396	41.989899
524288	32	8573.521719	42.05246
524288	64	8614.090217	42.041779
524288	128	11115.33207	42.007828
1024000	16	6828.852153	42.054787
1024000	32	12393.49292	42.034473
1024000	64	14538.84601	42.054298
1024000	128	18007.87818	42.037792

Block Size and megatrials per second



number of trials and megatrials per second



Patterns in the curves

There is almost an exponential growth in megatrials per second as the number of trials increases. This shows the immense benefits of parallelization on the GPU and how well it scales. There is a less straightforward pattern when we look at performance vs blocksize. There is a large increase in performance from 16 to 32 but a much smaller increase from 32 to 64. Then from 64 to 138 there is slightly more slope.

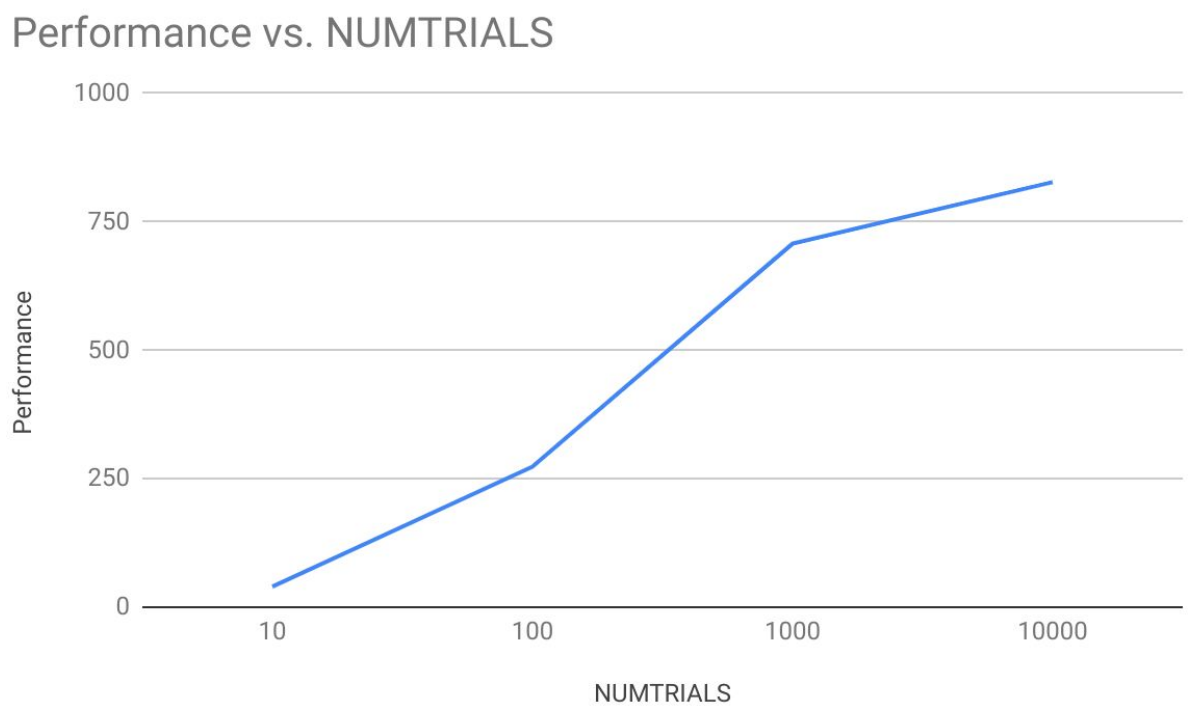
Why do the patterns look this way?

As the block size increases there is more room for parallel tasks as there are more threads per block. This is why megatrials per second increases with block size, although it is not a constant increase. There is a clear change in slope between each interval block size. The almost exponential size slope in the number of trials vs megatrials per second graph is representative of what kind of programs benefit from parallelization. The larger the data sets (in this case from repetitive trials) the more performance benefit there is parallelization.

Why is a block size of 16 so much worse than the others?

The block size of 16 is worse than the others because we are restricting the number of threads per block in CUDA to the point that it is bottlenecking the system and hindering the performance benefits we would otherwise see by parallelizing the trials in this simulation.

Compared to Project 01



Above is the graph from project one, as we can see the CPU is limited in the amount of parallelization benefits it can offer compared to the vast number of cores available on the GPU. The diminishing returns on parallelization were not reached on the GPU where as they were quickly reached on the CPU which could simply not scale with the problem evenly at even 10000. The CPU is much more limited in terms of what kind of parallel problems it can solved compared to an enterprise grade GPU like the Nvidia Tesla V100.

What does this mean for the proper use of GPU parallel computing?

This means that the GPU is much better suited for extremely large data sets whereas the CPU is suitable for much smaller data sets being run in parallel. The CPU is also more abundantly available than the GPU for most users, so extremely large datasets and problems of scale should be handled with the appropriate hardware if performance and time is important.