Trevor Stahl

Project 6

CUDA Monte Carlo

CS 475e

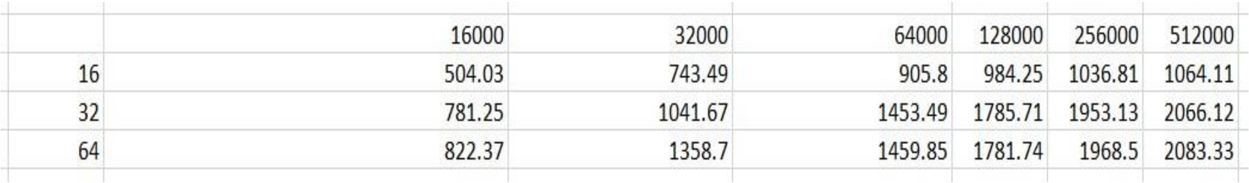
Professor Bailey

I ran the code on my own personal machine. I have Windows 10 up to date as of 5/30/2019 with an AMD 2700x, 8 core hyper threaded CPU at 4.1 Ghz right now. The GPU is an Nvidia 1080ti with a mediocre stock cooler. 16.3 GBs of RAM. Background CPU process kept below 5% and background GPU kept below 10%.

I am using Visual studio 2017. I needed to retarget the solution and change the SDK version.

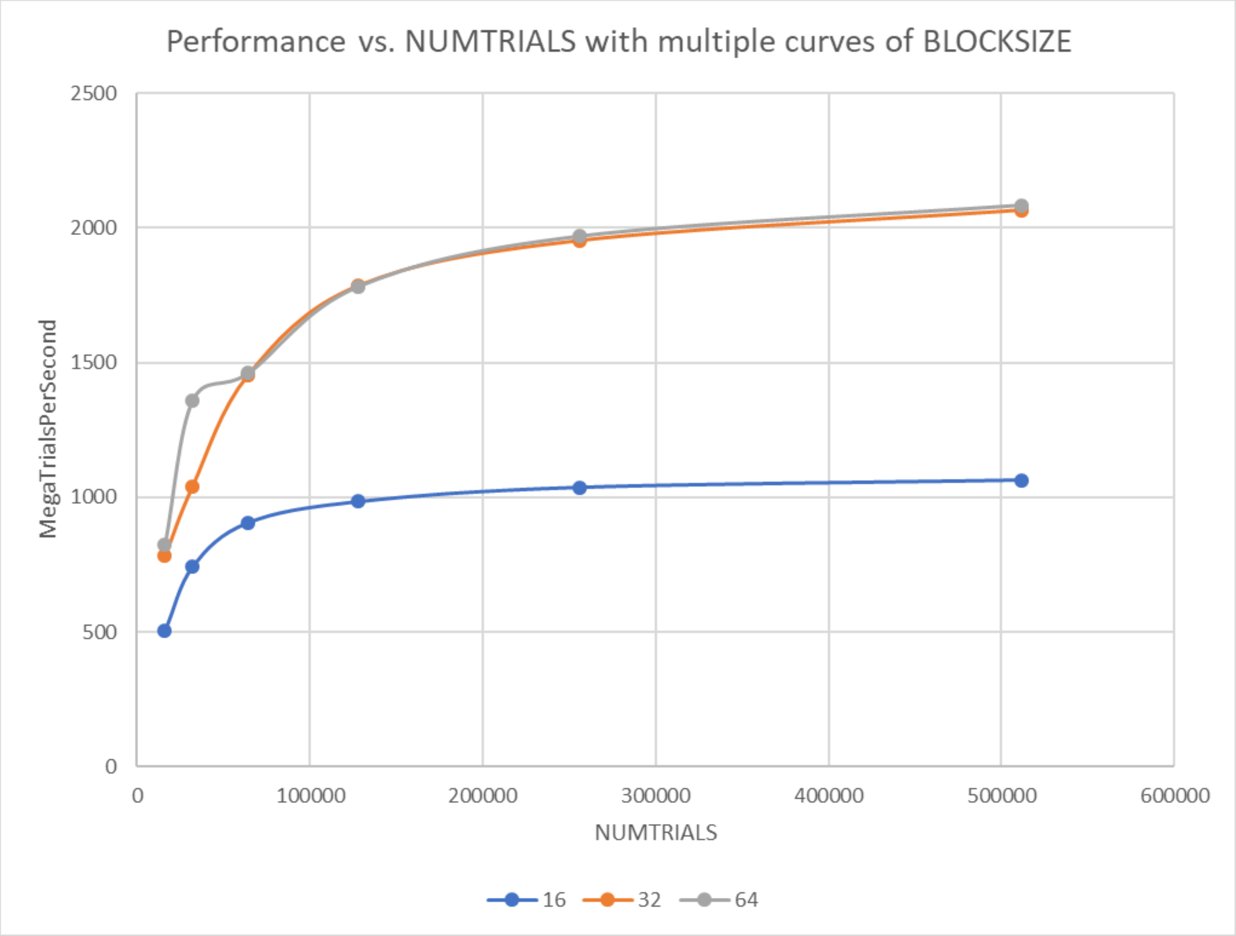
“GPU Device 0: "GeForce GTX 1080 Ti" with compute capability 6.1”

Table:

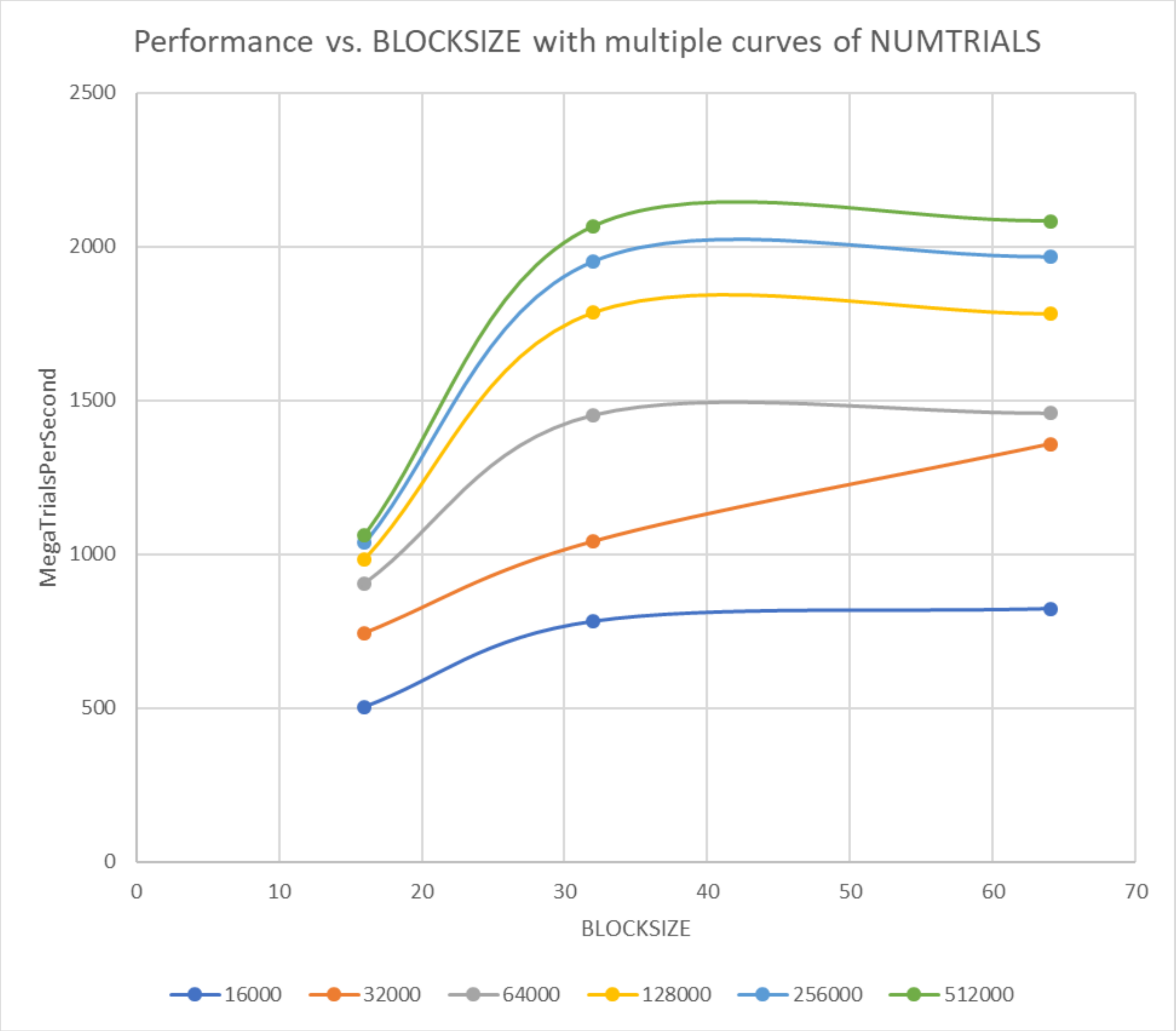


	16000	32000	64000	128000	256000	512000
16	504.03	743.49	905.8	984.25	1036.81	1064.11
32	781.25	1041.67	1453.49	1785.71	1953.13	2066.12
64	822.37	1358.7	1459.85	1781.74	1968.5	2083.33

Graph of performance vs. NUMTRIALS with multiple curves of BLOCKSIZE:



Graph of performance vs. BLOCKSIZE with multiple curves of NUMTRIALS:



Patterns seen:

- A block size of 16 is consistently much worse than 32 or 64
- A block size of 64 performs significantly better than 32 only at NUMTRIALS = 32000
- A block size of 64 performs very similarly compared to a block size of 32 at all of the other points
- A NUMTRIALS of 32000 seems to produce a nearly straight line across 16, 32, 64 BLOCKSIZE values
- There is occasionally a very slight decrease in performance when holding NUMTRIALS constant but going from 32 to 64 blocksize. Most times 32 to 64 provides an increase

Why might these patterns be:

- I am not at all sure why 64 BLOCKSIZE only performs significantly better than 32 at a low NUMTRIALS value once
- Because a block size of 64 and 32 both provide warp sizes of 32 threads, a block size of 64 and 32 would perform similarly. (I think)
- A NUMTRIALS of 32000 probably has a straight line only by chance and the graph dimensions
- A block size of 64 might perform slightly worse because it might lead to having 2 thread blocks commute unnecessarily compared to 32 block size (Maybe)

Why is a BLOCKSIZE of 16 so much worse?

- A block size of 16 might be much worse because it does not provide for the ideal warp size and causes a doubling of the need for grids/blocks

What this means for proper use of GPU parallel computing:

- This all suggests to me that one needs to use a block size evenly divisible by 32, including 32. Again, I think this is because of the warps.

Probability:

The average probability for all of the different runs was 41.89%