Kristin Schaefer

Project #5 : CUDA Monte Carlo

Machine: DGX / NVIDIA V100 graphics card

Performance vs. Number of Trials and Block Size

BLOCKSIZE \ NUMTRIALS	16384	32768	65536	131072	262144	524288
16	533.33	972.46	1833.48	2584.23	4558.71	6061.41
32	571.43	1065.56	1988.35	3637.66	6045.76	9163.31
64	571.43	968.78	2066.60	4063.49	7025.73	10382.76
128	571.43	1141.58	2235.81	4129.03	7684.80	10893.62

Performance : MegaTrials/Second

Max Performance: 10893.62 MegaTrials/Second with 128 block size and 524,288 number of trials

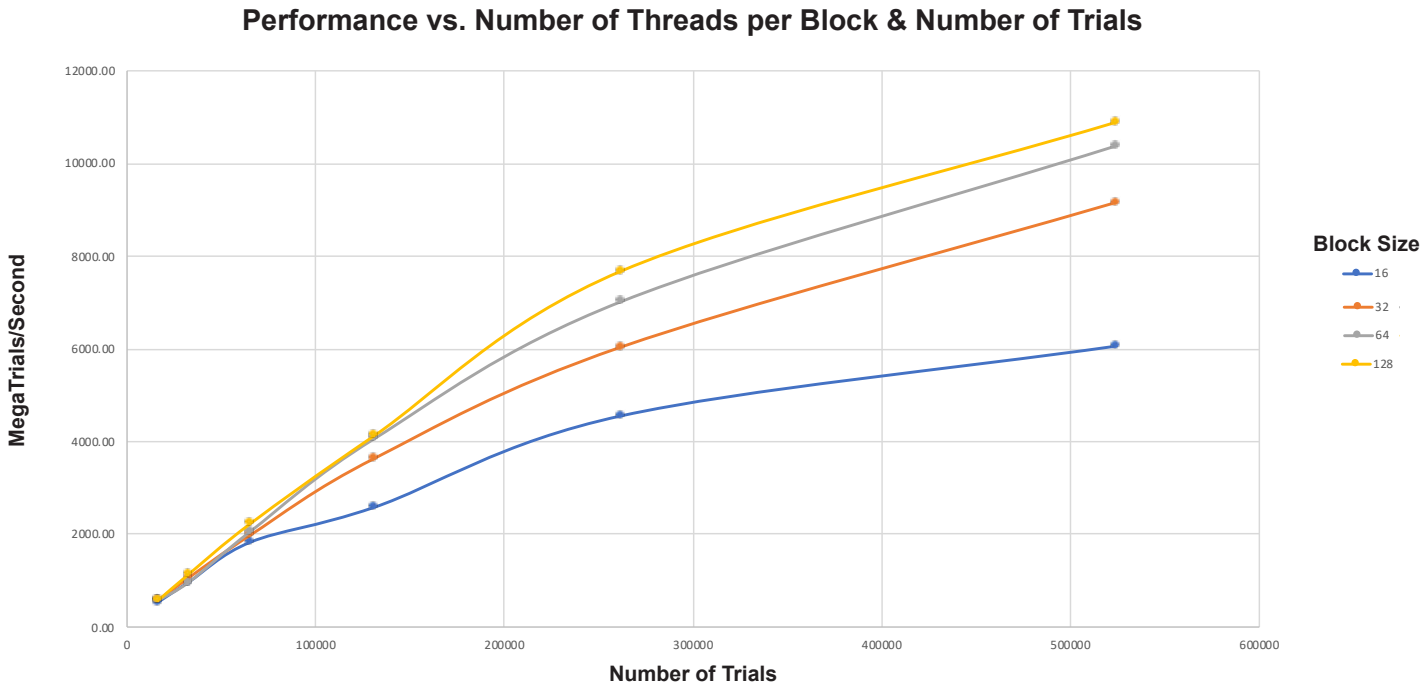
Probability vs. Number of Trials and Block Size

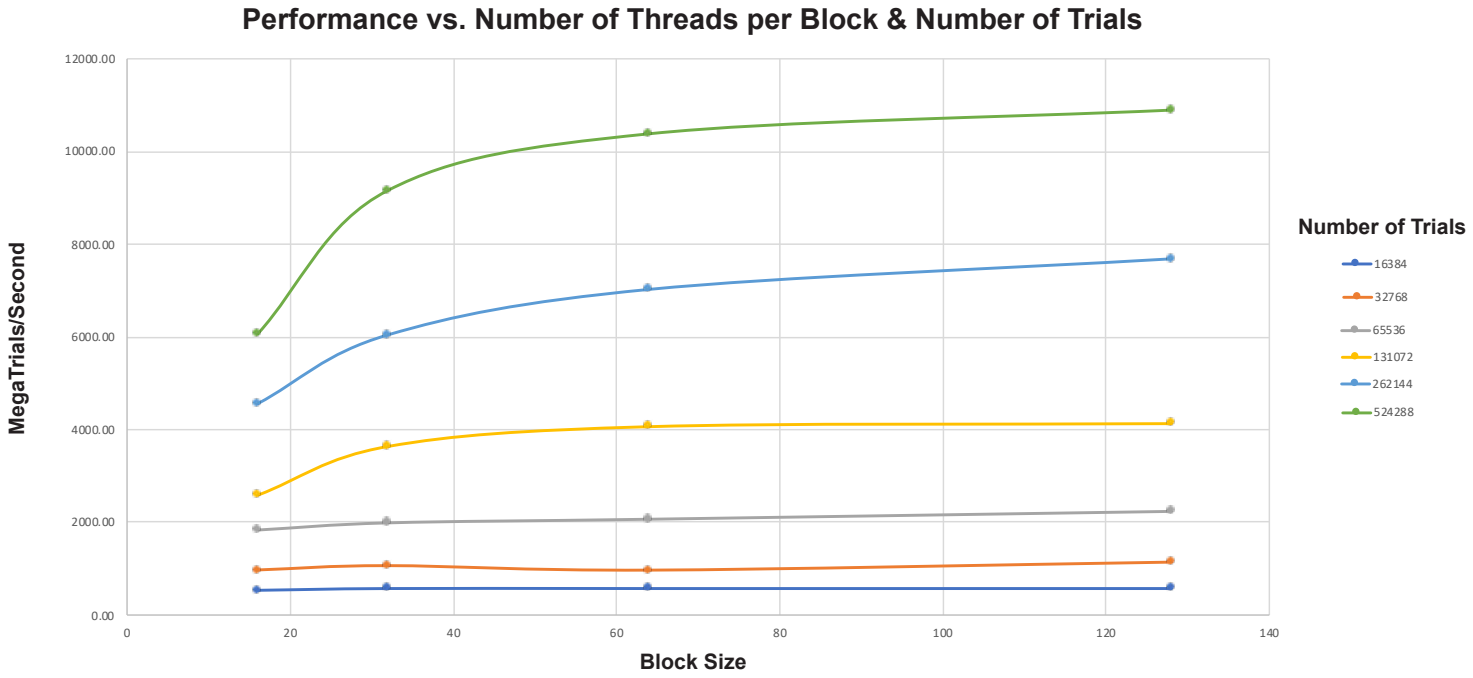
BLOCKSIZE \ NUMTRIALS	16384	32768	65536	131072	262144	524288
16	42.38	42.02	41.74	42.09	42.09	42.16
32	41.28	41.89	42.36	41.84	42.01	42.07
64	42.16	42.26	41.95	42.17	41.86	42.06
128	41.70	41.82	42.10	41.96	41.86	41.97

Average probability: 41.99%

Kristin Schaefer

Project #5 : CUDA Monte Carlo





Kristin Schaefer

Project #5 : CUDA Monte Carlo

What patterns are you seeing in the performance curves? Why do you think the patterns look this way? Why is BLOCKSIZE of 16 so much worse than the other two:

As the number of trials and the block size increase, the performance increases consistently. I think the patterns look this way for two reasons. The first reason being that with data parallelism on the GPU, as the data set size increases, performance generally increases. The second reason is that generally (but not always) the greater the number of threads per workgroup there are, the greater the performance will be. The performance is significantly worse with BLOCKSIZE of 16 because the number of threads in a warp is 32. If you use 16 threads, then you are leaving half of the threads on the table.

How do these performances results compare with what you got in Project #1? Why:

The maximum performance of Project #1 was 192.66 MegaTrials/Second with 16 threads and 1,000,000 number of trials. The maximum performance of Project #5 was 10893.62 MegaTrials/Second with 128 block size and 524,288 number of trials (1,000,000 was not tested). This is a performance increase greater than 50x, which is very significant.

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

The results lead to the conclusion that you should have a large data set size, perhaps greater than 200,000, to consider using CUDA. You should also have a minimum of 32 work items per workgroup. Then you should run benchmarks to see if adding more work items (in multiples of 32) to the work group increases the performance. If you have the opportunity to transform your code to use CUDA or OpenMP (CPU parallelism), you should make it your first priority to use CUDA, as it offers a significantly greater performance increase than OpenMP.