Xiaoying Li
lixiaoyi@oregonstate.edu
Project #5 CUDA Monte Carlo

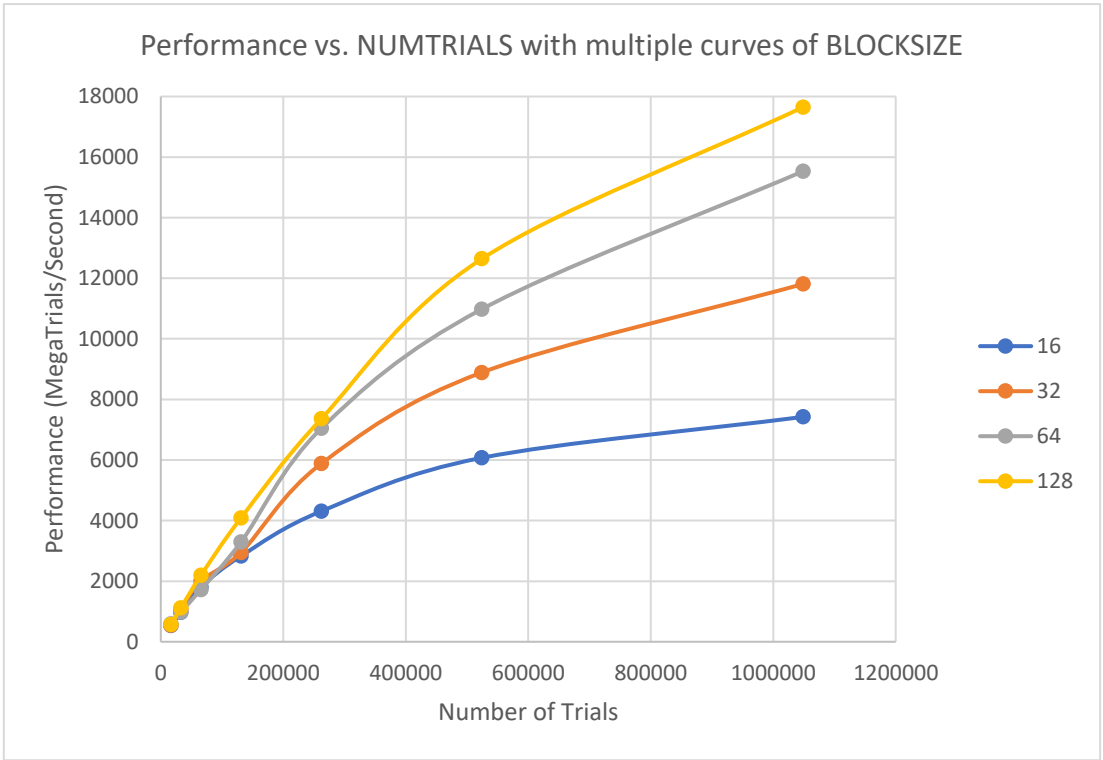
- **Machine:**
  I ran the project on OSU's DGX system.

- **Probability that the beam hits the plate:**
  about 42%

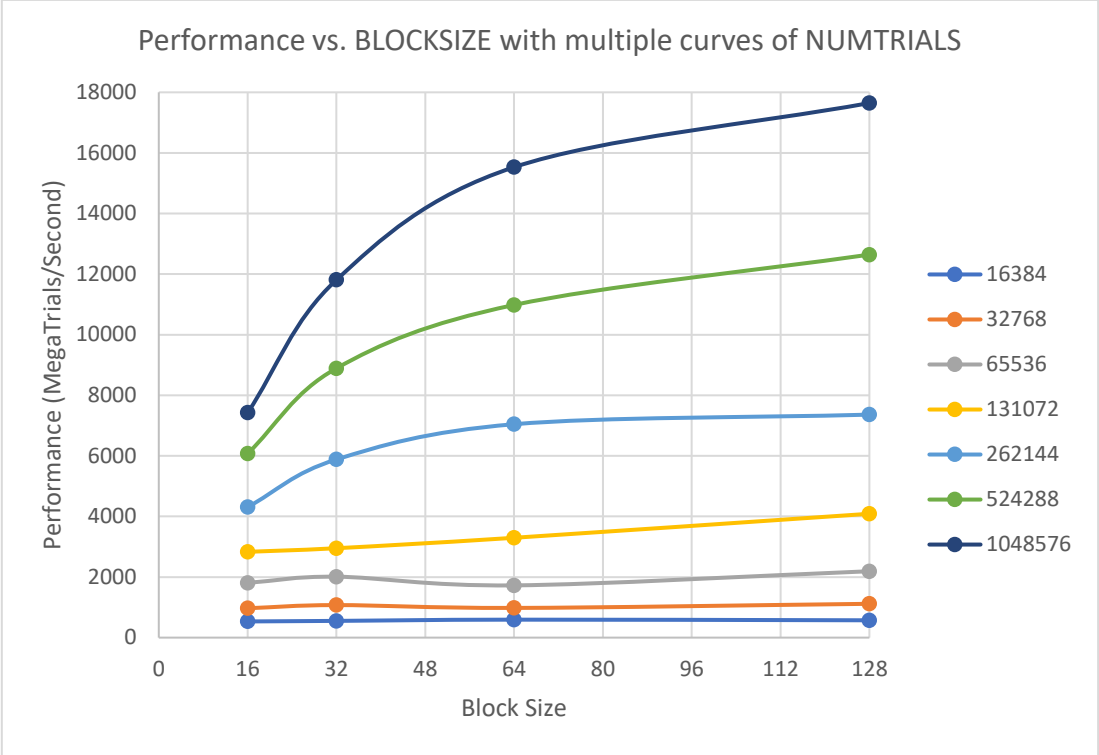
- **Table of Performance vs. NUMTRIALS and BLOCKSIZE:**

NUMTRIALS / BLOCKSIZE	16384	32768	65536	131072	262144	524288	1048576
16	533.33	968.78	1807.59	2830.68	4309.31	6072.65	7421.97
32	551.72	1073.38	2009.81	2951.01	5880.83	8889.85	11808.29
64	592.59	979.91	1723.91	3295.25	7043.85	10981.23	15529.86
128	571.43	1115.47	2190.37	4087.82	7360.29	12641.98	17645.66

- **Graph of Performance vs. NUMTRIALS with multiple curves of BLOCKSIZE:**



- **Graph of Performance vs. BLOCKSIZE with multiple curves of NUMTRIALS:**



- **Patterns in the performance curves and why:**

For the same number of trials, when the number of trials is small (less than 100K), the change of block size almost has no effect on the performance. But when the number of trials gets larger, the bigger the block size is, the higher the performance is. Because as the block size gets bigger, the overheads also get bigger. And when the number of trials is small, the increased overheads override the increased speed. Thus, when the number of trials is small (less than 100K), bigger block size doesn't bring better performance. But when the number of trials gets larger, the overheads has less and less influence on the speed. And the threads in a thread block can cooperate with each other by synchronizing their execution and efficiently sharing data through a low latency shared memory. Thus, larger block size means more threads can cooperate with each other in the same time, which makes the performance results better.

For the same block size, the bigger the number of trials is, the higher the performance is. Because the GPU's workload is divided into a grid of blocks, each block's workload is divided into a grid of threads. And GPU programs expect to not just have a few threads, but to have thousands of them. Thus, larger number of trials makes better use of GPU's computing capacity, which makes the performance results better.

- **Reason for a BLOCKSIZE of 16 so much worse than the others:**

Because for the multithreaded Warp scheduler, there needs to be a bunch of Warps to work on so that something is always ready to run. And to help this, any particular grid of threads should be multiple of 32. Every 32 threads constitute a Warp. Each thread in a Warp

simultaneously executes the same instruction on different pieces of data. So, anything less than 32 is wasting Warp. Because the Warp is trying to execute 32 units at a time, but when a BLOCKSIZE is only 16, there are 16 units that could be computing useful information but aren't, which makes the execution go idle. Therefore, the performance of a BLOCKSIZE of 16 is much worse than the others.

- **Compare these performance results with performance results in Project #1 and why:**

These performance results are much, much better than the performance results in Project #1. Even the worst performance number in this project is much larger than the best performance number in Project #1.

Because this CUDA Monte Carlo simulation is a very intense data parallel application, which exploits GPU's computing power. GPU programs have thousands of threads. Each thread executes the same program, but operates on a different small piece of the overall data. Thus, many, many threads, all wake up at about the same time, all execute the same kernel program, all hope to work on a small piece of the overall problem. But in Project #1, we use CPU multicores, which only has very, very small numbers of threads. Thus, less work can be done at the same time. Therefore, the performance results in this GPU project are much better than the performance results in Project #1.

- **The meaning for the proper use of GPU parallel computing:**

GPU is good at data parallel programming. In order to efficiently use GPU's parallel computing power, the problem should be break into many, many small pieces.