PROJECT 6: CUDA Monte Carlo

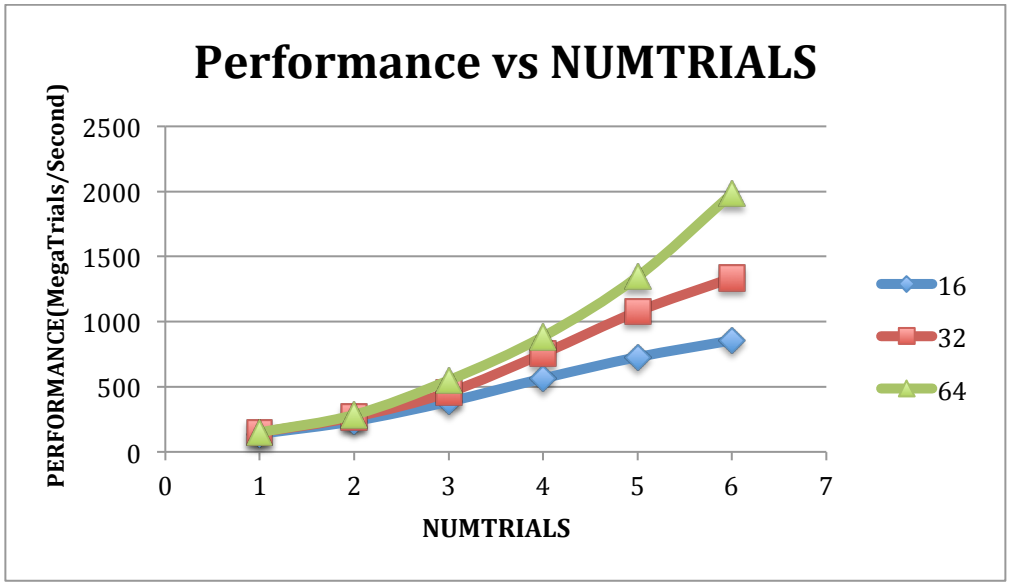
1. **Tell what machine you ran this on.**

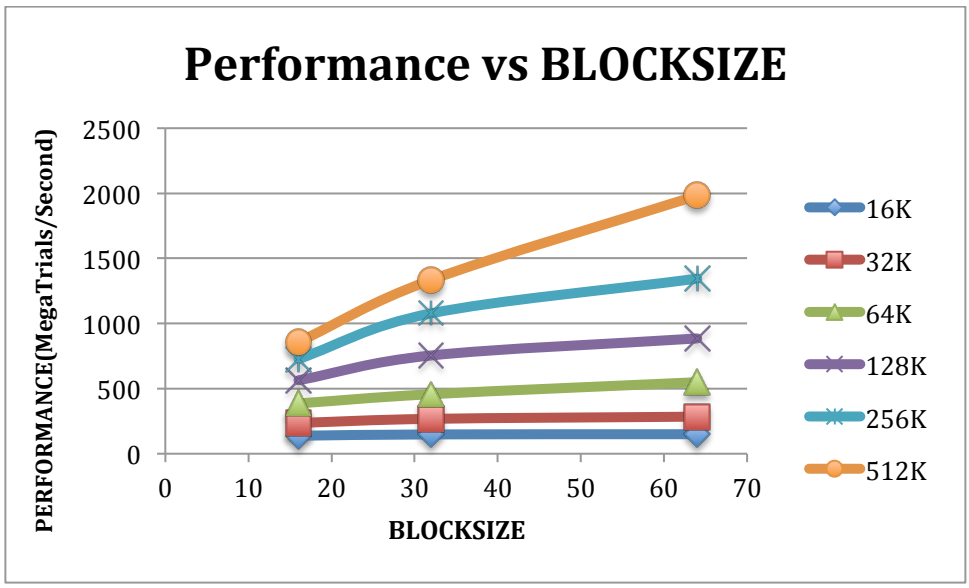
   I ran this on rabbit.

2. **Show the tables and graphs**

BLOCKSIZE	NUMTRIALS					
	16K	32K	64K	128K	256K	512K
16	137.82	236.33	385.25	563.95	727.66	856.05
32	148.19	267.78	456.94	754.61	1078.04	1335.29
64	149.58	284.52	549.21	885.24	1344.49	1982.1

Probability reported: 0.4198





3. **What patterns are you seeing in the performance curves?**

   The performance seems to improve as the numtrials increases. Blocksize 64 performs much better than the blocksize 16. Also, as seen above, the performance tends to improve as blocksize increases (graph 2). With higher numtrials the performance is much better than that with lower numtrials.

4. **Why do you think the patterns look this way?**

   All threads in a Warp execute the same instruction at any given time, on different data. Each SM needs a gang of Warps to work on so that something is always ready to run. Hence, with higher blocksize (threads per block with multiples of the Warp size 32) the performance tends to improve, as all threads will execute the same instruction parallely on different data.

5. **Why is a BLOCKSIZE of 16 so much worse than the other two?**

   Multiples of the Warp size 32 is good number of threads per block. A “Warp” is a group of 32 threads that are simultaneously executing the same instruction on different pieces of data. Hence, the performance of blocksize 16 is much worse than a blocksize of 64.

6. **What does that mean for the proper use of GPU parallel computing?**

   It means that by experimenting, we can maybe find a good combination of number of numtrials and blocksizes for optimally utilizing the GPU parallel computing. After observation, it seems like having large data items distributed between large number of threads(blocksize) will give better performance.