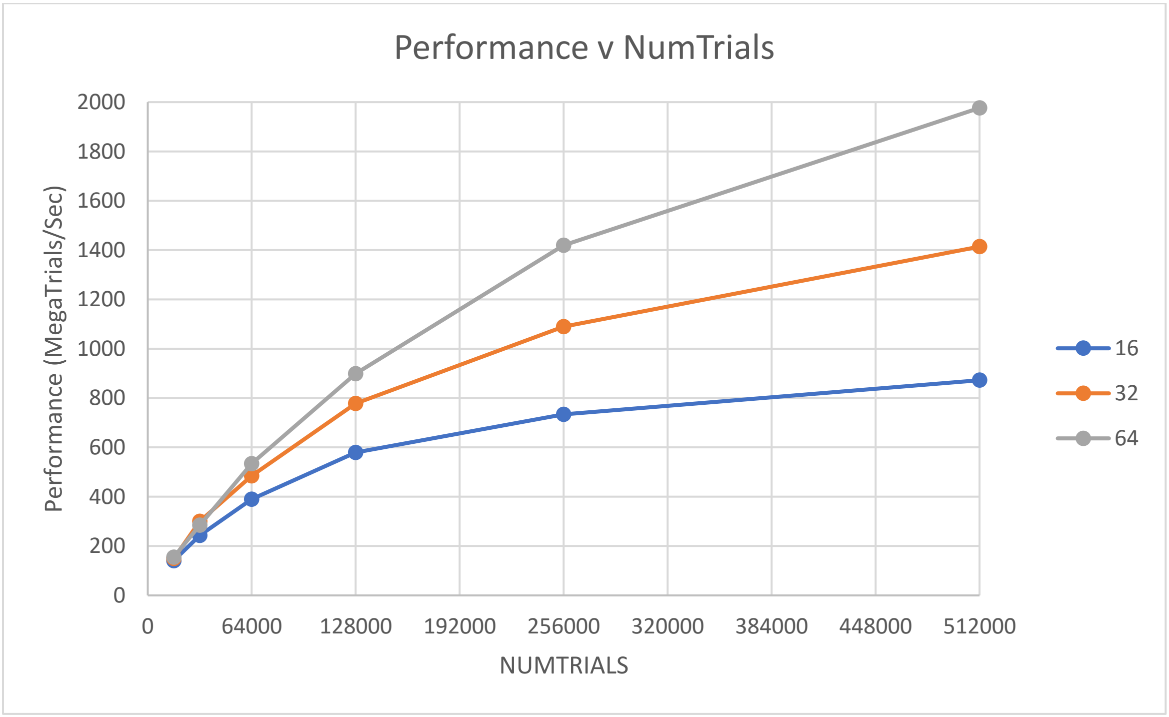
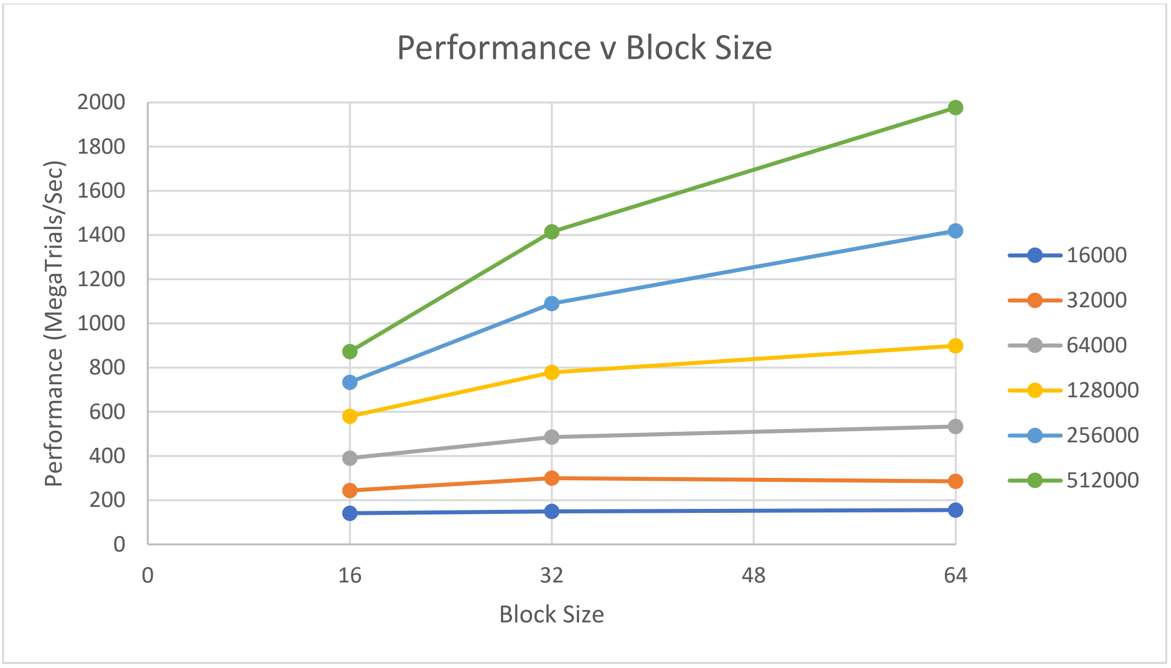
Amy Stockinger (stockina)
CS475 Project 6 – CUDA Monte Carlo
Due: 5/31/2019

I ran my program on *rabbit*.
My calculated probability of a hit given the X, Y and r ranges: about 42%
Here are my performance results:

Monte Carlo Performances (MegaTrials/sec)

		Trials					
		16000	**32000**	**64000**	**128000**	**256000**	**512000**
blocksize	**16**	141.24	243.9	390.7	579.96	734.15	872.51
	32	149.43	299.85	484.97	778.66	1090.36	1414.05
	64	155.47	285.06	533.62	899.08	1419.45	1976.53





What patterns are you seeing in the performance curves? Why do you think the patterns look this way? Both graphs show relatively clean curves, especially once the trials get to 128k. In the first graph, we start to see a clear separation of the three lines at 128k size. Likewise, in the second graph, 128k size (yellow line and above) is where we start to see a more defined slope in the curves. The graphs look this way because 16 block size is underpowering the GPU as compared to the massive performance gains seen by switching to a larger block size.

Why is BLOCKSIZE 16 so much worse than the other two? This block size seems to be too small compared to the number of blocks. As the size grows larger, the gap in performance between 16 block size and 32/64 block size increases by *a lot*. By size 256k, we can see that the 64 block size performance is nearly double that of the 16 block size, and by size 512k, the 32 block size also becomes nearly double that of 16 block size. Block size 16 is just not a good choice, especially as the trials increase.

What does this mean for proper use of GPU parallel computing? It is better to spread the threads over more blocks and get more done at one time rather than using more blocks with a lower number of threads.