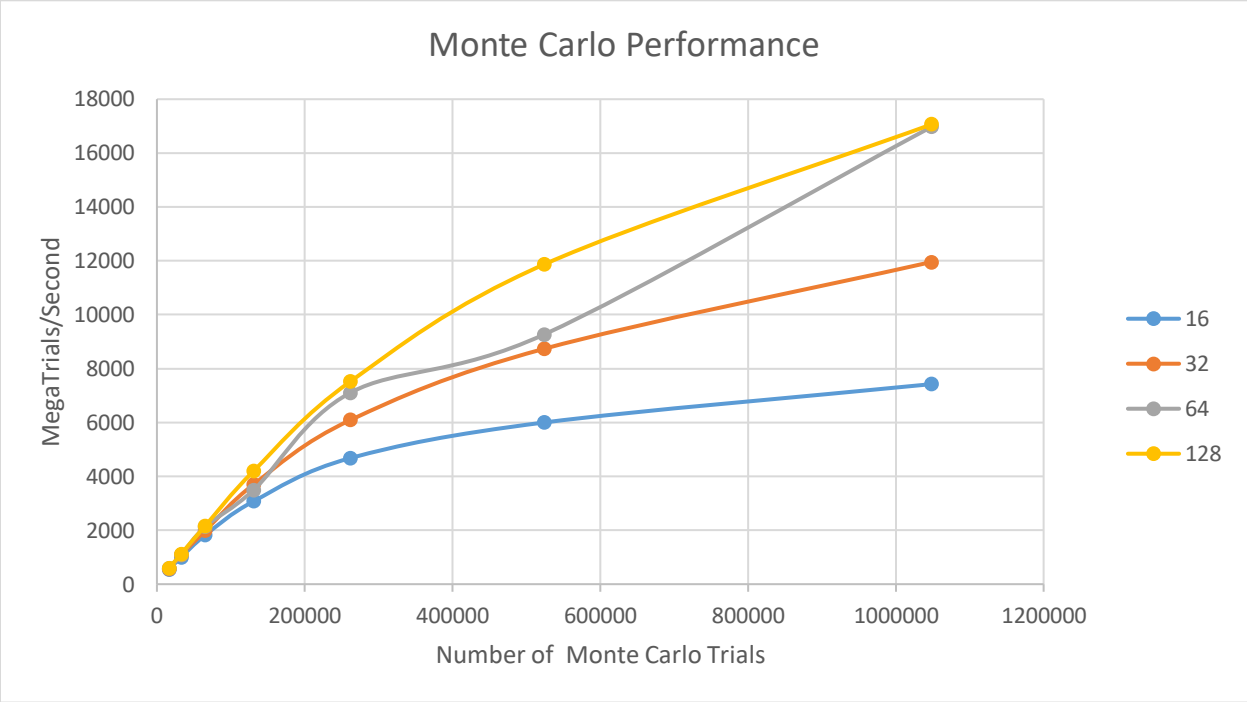
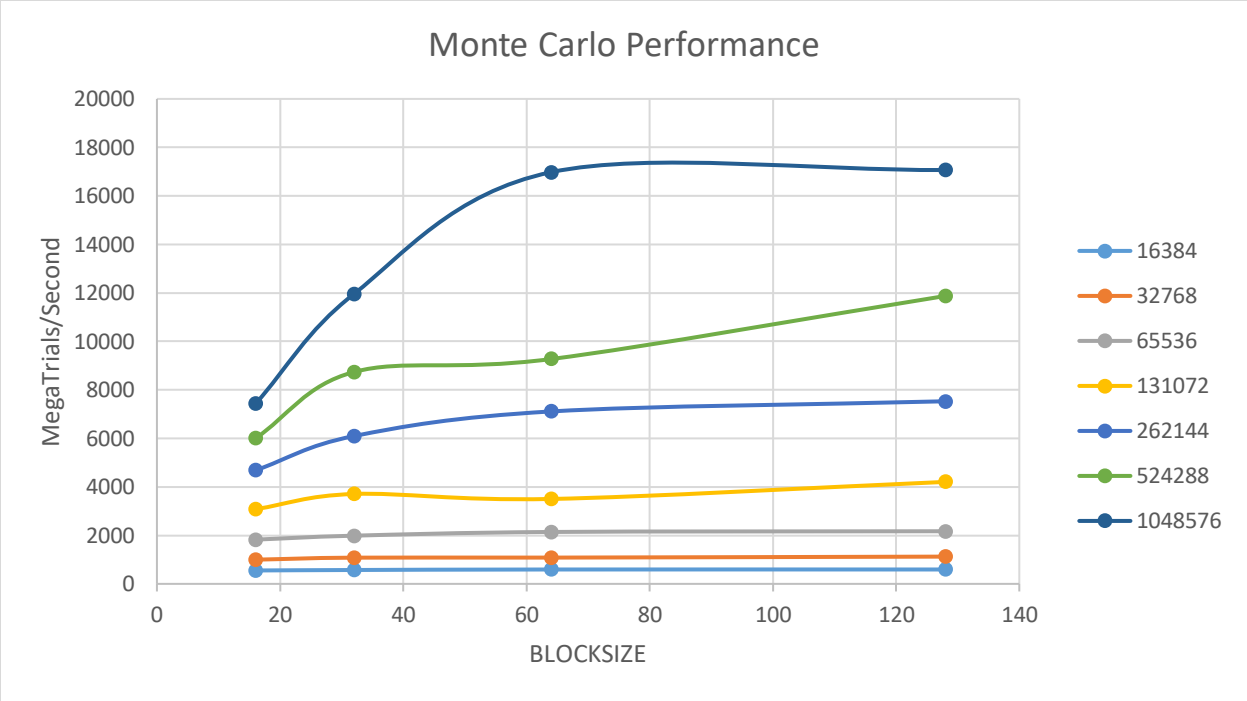
liuchenx@oregonstate.edu Chenxing Liu 2020 Spring

1. I ran it on OSU's DGX system.

2.

BLOCKSIZE / TRIALS	16384	32768	65536	131072	262144	524288	1048576
16	551.724124	997.0789	1823.686	3079.699	4681.143	6001.465	7425.335
32	571.428564	1080.169	1986.421	3706.787	6095.238	8738.134	11959.12
64	592.592592	1082.452	2140.021	3500.855	7111.111	9272.213	16978.24
128	592.592592	1124.04	2169.491	4205.339	7529.412	11872.46	17066.67





3.

When the number of trials remains the same, the performance improves as the block size gets larger.

When the block size remains the same, the performance improves as the number of trials increases.

4.

Reason for improved performance with larger block size:

Every 32 threads constitute a “Warp”. It’s likely that a Warp’s execution will need to stop at some point. If we have larger block sizes, when one Warp is idle, other Warps can be swapped in. Thus, idle time is reduced and performance improved.

Reason for improved performance with increasing number of trials:

GPU chips are customized to handle streaming data and provide superior processing power than their CPU counterparts. We need a large amount of data to feed GPUs so that their computing power can be well utilized. We see performance constantly goes up as numbers of trials increase from 16384 to 1048576, probably because we haven’t reached the limit of GPU computing power.

5.

The way GPU chips compute is that 32 threads are organized into a Warp and synchronized when executing instructions. A block size of 16 is not enough to fill a Warp, thus leaving a lot of computing power on the table. Even a block size of 32 is not efficient, as a Warp may sit idle at some point. So performance for a block size of 16 is really bad.

6.

Compare 16 threads and 1000000 trials in project 1 with 1048576 trials, block size of 128 in project 5, we saw a tenfold increase in performance, from 180 to almost 1800.

Reason: GPUs are designed to handle streaming data and provide superior computing power than their CPU counterparts.

7. CPUs are for general purpose programming and can handle irregular data structures and flow control. GPUs are designed for data parallel programming and can only handle regular data structures and flow control. If we need to do repetitive and highly-parallel computing task, like in project 5, we have one instruction for many pieces of data, we should use GPUs, instead of CPUs, to get much better performance.