Project #5 CUDA: Monte Carlo Simulation

Erick Branner, brannere@oregonstate.edu | 18 May 2022

1 Tell what machine you ran this on

Rabbit

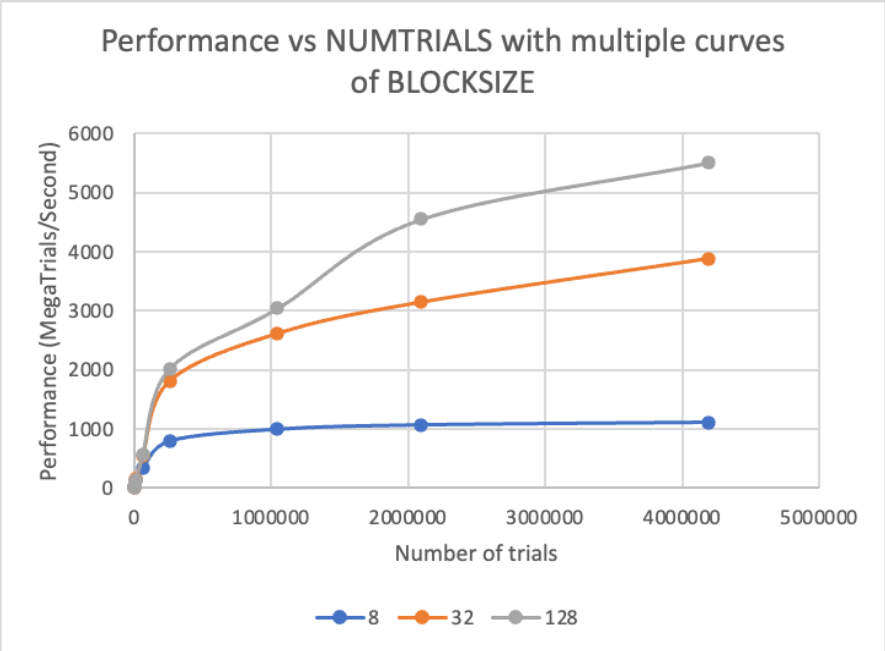
2 Show the table and the two graphs

Monte Carlo performance table

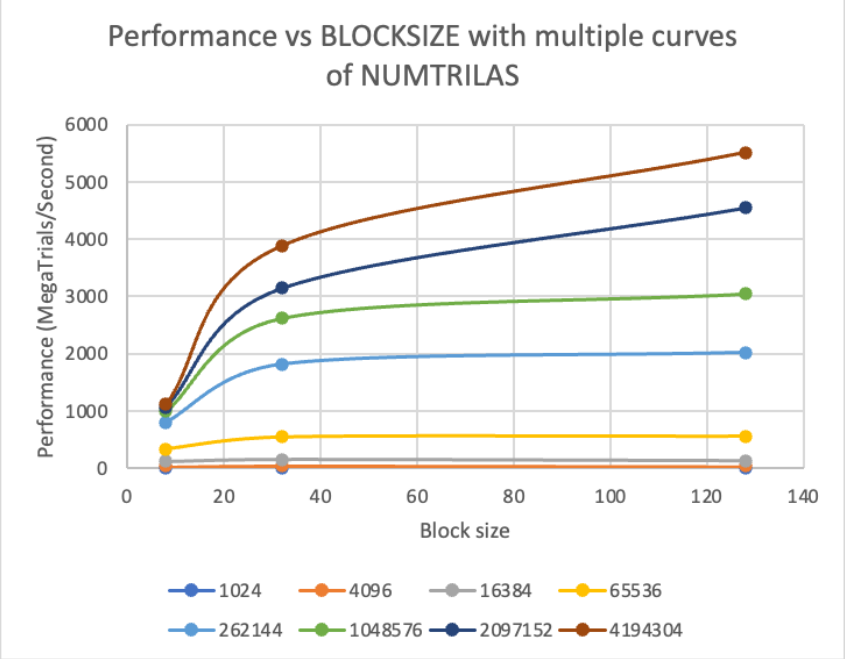
number of trials	blocksize	MegaTrials/Second	probability
1024	8	9.2996	23.05%
1024	32	8.665	22.27%
1024	128	8.5242	19.43%
4096	8	36.5923	23.07%
4096	32	39.3725	23.90%
4096	128	37.5257	22.68%
16384	8	124.0611	22.54%
16384	32	147.1687	22.56%
16384	128	134.1368	22.42%
65536	8	334.4219	22.78%
65536	32	544.102	22.45%
65536	128	558.647	22.78%
262144	8	798.9077	22.47%
262144	32	1817.617	22.57%
262144	128	2023.2157	22.35%
1048576	8	996.3513	22.58%
1048576	32	2617.4614	22.47%
1048576	128	3041.3961	22.47%
2097152	8	1068.7715	22.45%
2097152	32	3146.3825	22.48%
2097152	128	4552.3757	22.52%

4194304	8	1113.2704	22.47%
4194304	32	3881.5449	22.51%
4194304	128	5509.3102	22.49%

Graph of performance vs. NUMTRIALS with multiple curves of BLOCKSIZE



Graph of performance vs. BLOCKSIZE with multiple curves of NUMTRIALS



3 What patterns are you seeing in the performance curves?

Performance increases with larger block sizes *and* number of trials. Performance is much lower when the number of trials is in range [1024, 65536], as well as block size of 8 (*commented on in section 5*).

4 Why do you think the patterns look this way?

The performance jump from BLOCKSIZE 32 to 128 is because a Warp may stop at some point waiting for memory access – execution is paused. With BLOCKSIZE 128, there are four total Warps. When there's more than one Warp of threads, another Warp can jump in when one is blocked due to accessing memory.

5 Why is a BLOCKSIZE of 8 so much worse than the others?

A full Warp is not being utilized with block size of 8. With a BLOCKSIZE of 8, there are 24 (32-8) potential executions not being applied to data.

6 How do these formance results compare with what you got in Project #1? Why?

The performance of CUDA is *much* faster than OpenMP. CUDA has faster performance because instructions in data blocks are done at the same time. OpenMP threads must sequentially compute their own data chunk. For example, a data size of 32 in CUDA can have the same instruction performed each data bit synchronously; OpenMP would have to execute the instruction on data 1, then data 2, data 3, etc. The latter takes longer.

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

Given the answers in 4, 5 and 6, you should use at least one Warp to maximize your performance.