CS 575

Project #5

CUDA Monte Carlo

Xuming Wu

wuxum@oregonstate.edu

Code was run on the DGX Systems. Got results as:

1. Performance table

NUMTRIALS\BLOCKSIZE	16	32	64	128
2048	58.823528	51.282053	62.5	62.5
4096	125	121.21212	117.647057	121.21212
8192	250.000001	250.000001	250.000001	200.000005
16384	484.848481	484.848481	484.848481	400.00001
32768	886.580127	891.209734	969.696962	994.174716
65536	1391.304317	1680.065671	1836.771314	1802.816845
131072	2717.982837	2850.382744	3442.016862	3442.016862
262144	4218.331611	5326.398082	6335.653763	6455.476752
524288	5620.583041	7710.117228	9351.598425	8775.575726
1048576	7031.759516	10614.83645	13018.67264	13802.86485

2. Performance vs. NUMTRIALS with multiple curves of BLOCKSIZE

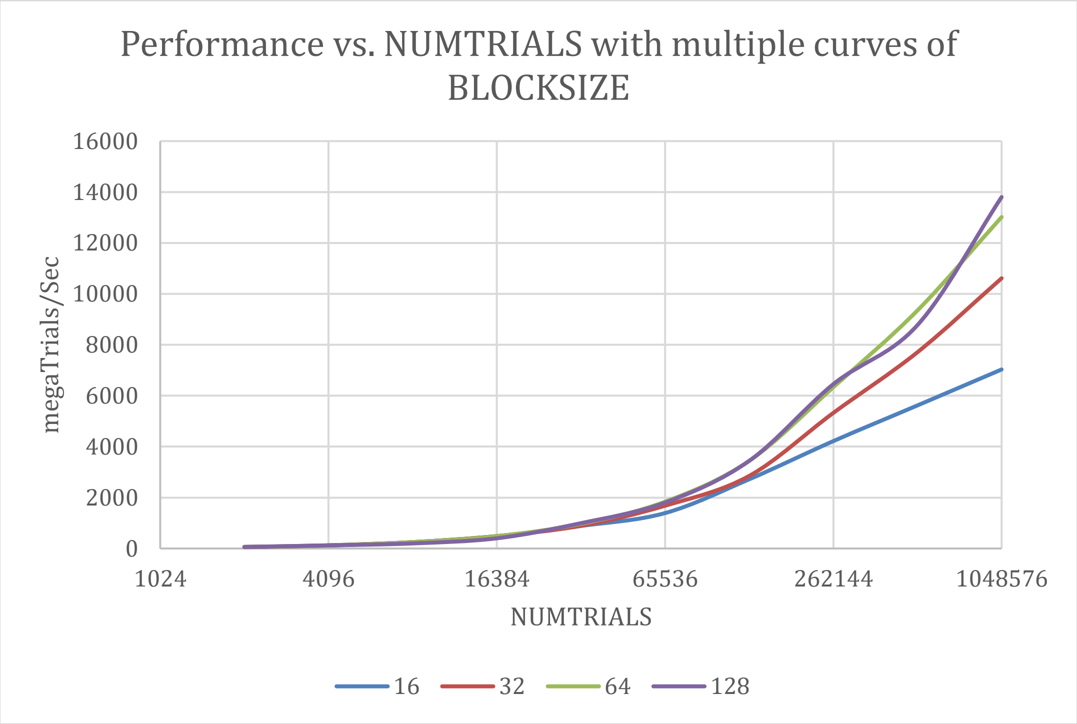

Figure 1

3. Performance vs. BLOCKSIZE with multiple curves of NUMTRIALS

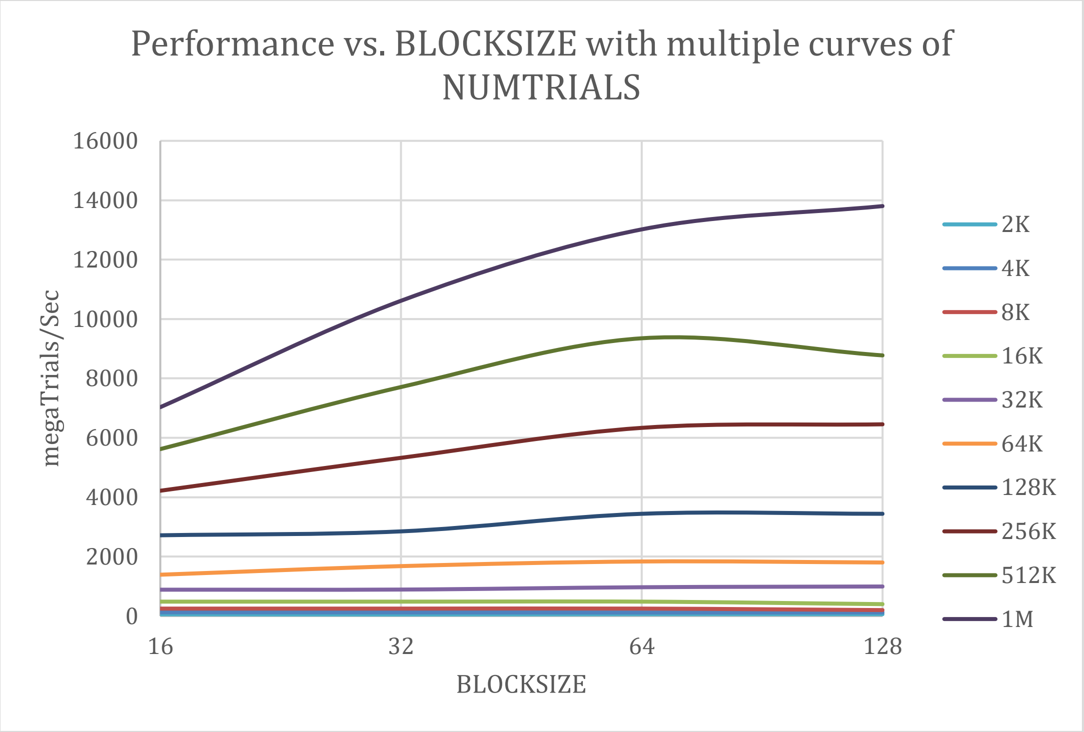

Figure 2

Commentary:

From the Figure 1 and 2 we can observe that the number of threads per block does not obviously affect the performance when using smaller dataset (2K ~ 128K Monte Carlo trials in this project). When the dataset is larger than 128K, the benefit of using more threads becomes obvious, which is getting a higher performance. I think it is because the CUs on the graphics card have great ability of computing such that small dataset can not fully exploit the performance of the CUs. Then for the same reason, a BLOCKSIZE of 16 is much worse than others when the NUMTRIALS gets larger. Compared to performance results in Project #1, the performance in this project has higher boundaries since GPUs have stronger compute capability than CPUs. This means that GPU has better performance in floating-point computation and parallel computing.