CS475 – Project #5

CUDA: Monte Carlo Simulation

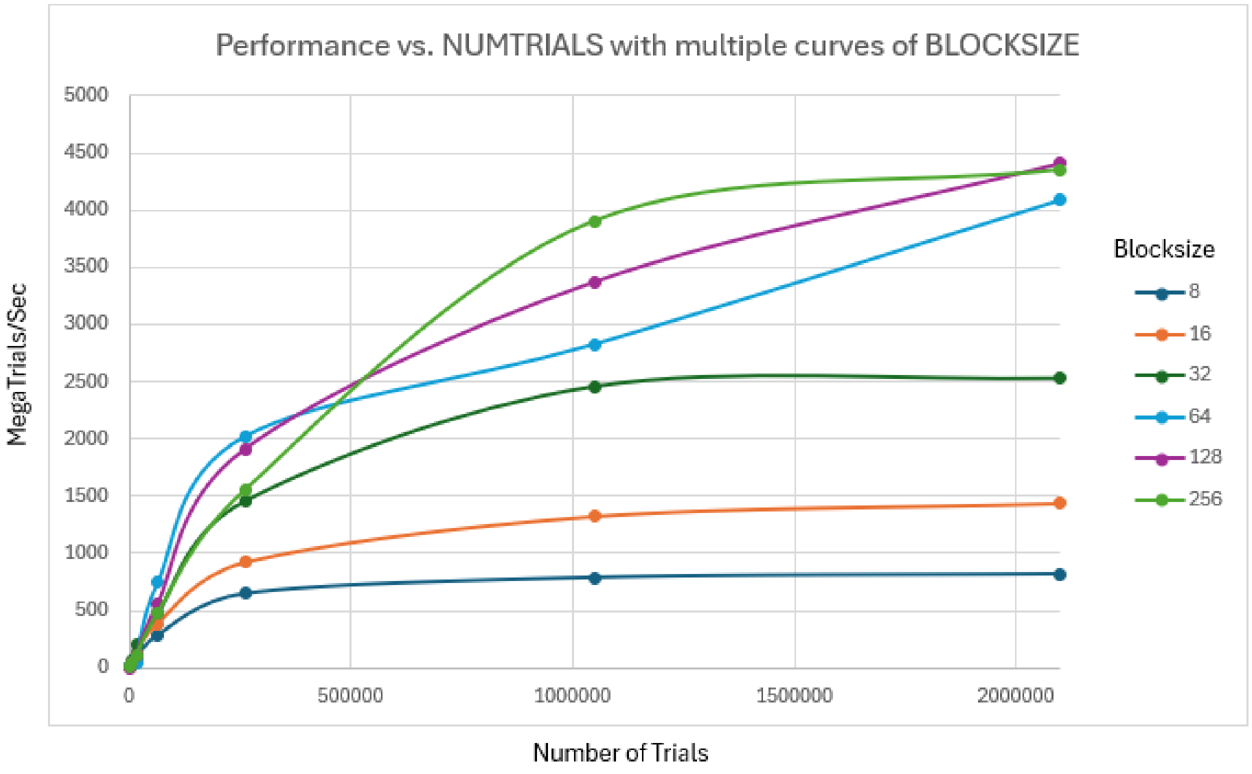
Richard Phan

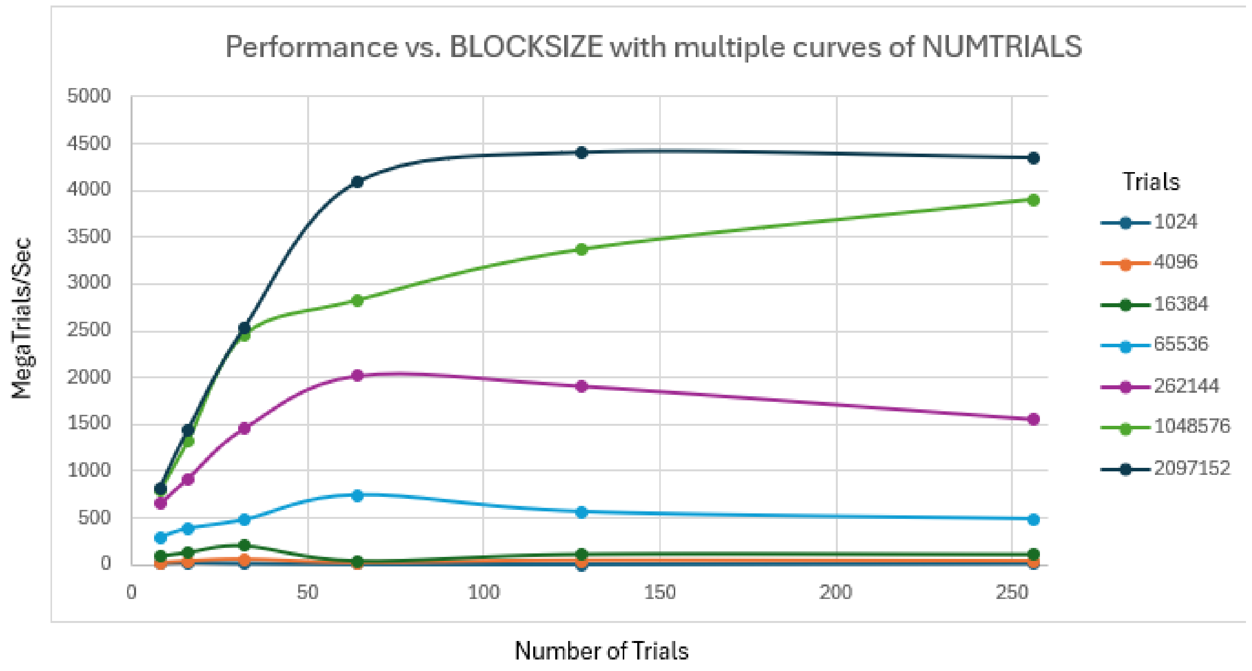
phanri@oregonstate.edu

Data Table

	8	16	32	64	128	256
1024	8.31	13.42	8.33	3.76	2.7	8.35
4096	20.1	34.05	51.22	22.21	40.69	35.86
16384	85.68	127.55	201.34	35.63	107.88	106.16
65536	288.17	380.88	475.17	742.03	562.33	484.28
262144	649.33	920.55	1458.17	2023.22	1912.68	1558.3
1048576	789.19	1325.46	2458.95	2828.49	3372.93	3906.07
2097152	819.31	1438.39	2529.96	4085.28	4407.26	4352.82

Graphs





Commentary

For this project, I used my personal desktop computer. To compile and run the CUDA Monte Carlo simulation, I connected to the Rabbit server provided by the school using Windows PowerShell with SSH. The server setup provided access to the appropriate GPU environment and CUDA compilers.

The estimated probability calculated by the simulation is approximately 8.75%

In terms of performance, I observed that increasing the number of trials consistently improves MegaTrials per second. Larger trial sizes allow better use of the GPU's parallel resources. Additionally, the BlockSize has a major impact on performance. The highest MegaTrials/sec values occurred with BlockSizes of 128 and 256. These sizes likely match well with the GPU's warp size and maximize occupancy.

BlockSize 8 consistently performed much worse than all others. This is likely because CUDA warps operate in groups of 32 threads. With only 8 threads, a warp is mostly idle, leading to inefficient execution and wasted GPU resources. This low utilization causes significantly reduced throughput.

Compared to Project #1, which used CPU parallelism (OpenMP), the CUDA version shows a dramatic speedup. In the OpenMP version, the best performance reached the low hundreds of MegaTrials/sec, while the CUDA version exceeds 4000 MegaTrials/sec. This shows the power of GPU computing for parallel tasks like Monte Carlo simulations.

Overall, this project shows how important it is to match your parallel algorithm to the underlying hardware. When done correctly, GPU computing can deliver huge performance benefits. The choice of BlockSize and the total number of trials are key factors in getting the most out of CUDA programs.