Parallel Programming CS - 575

Project 5 -CUDA: Monte Carlo Simulation

Submitted by

Rahul Kumar Nalubandhu

Onid : nalubanr

Email : nalubanr@oregonstate.edu

1. Tell what machine you ran this on.

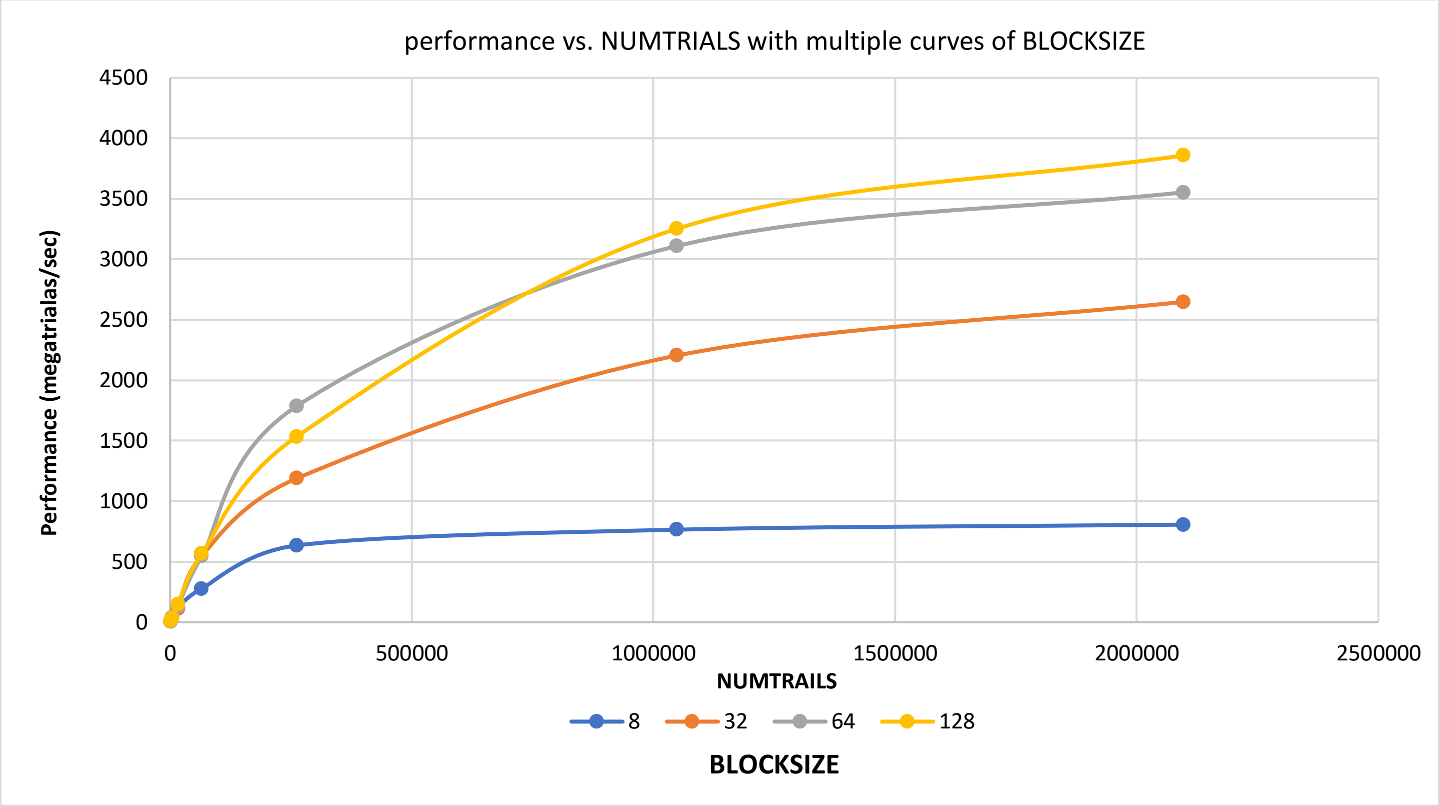
   I executed this project on rabbit server.

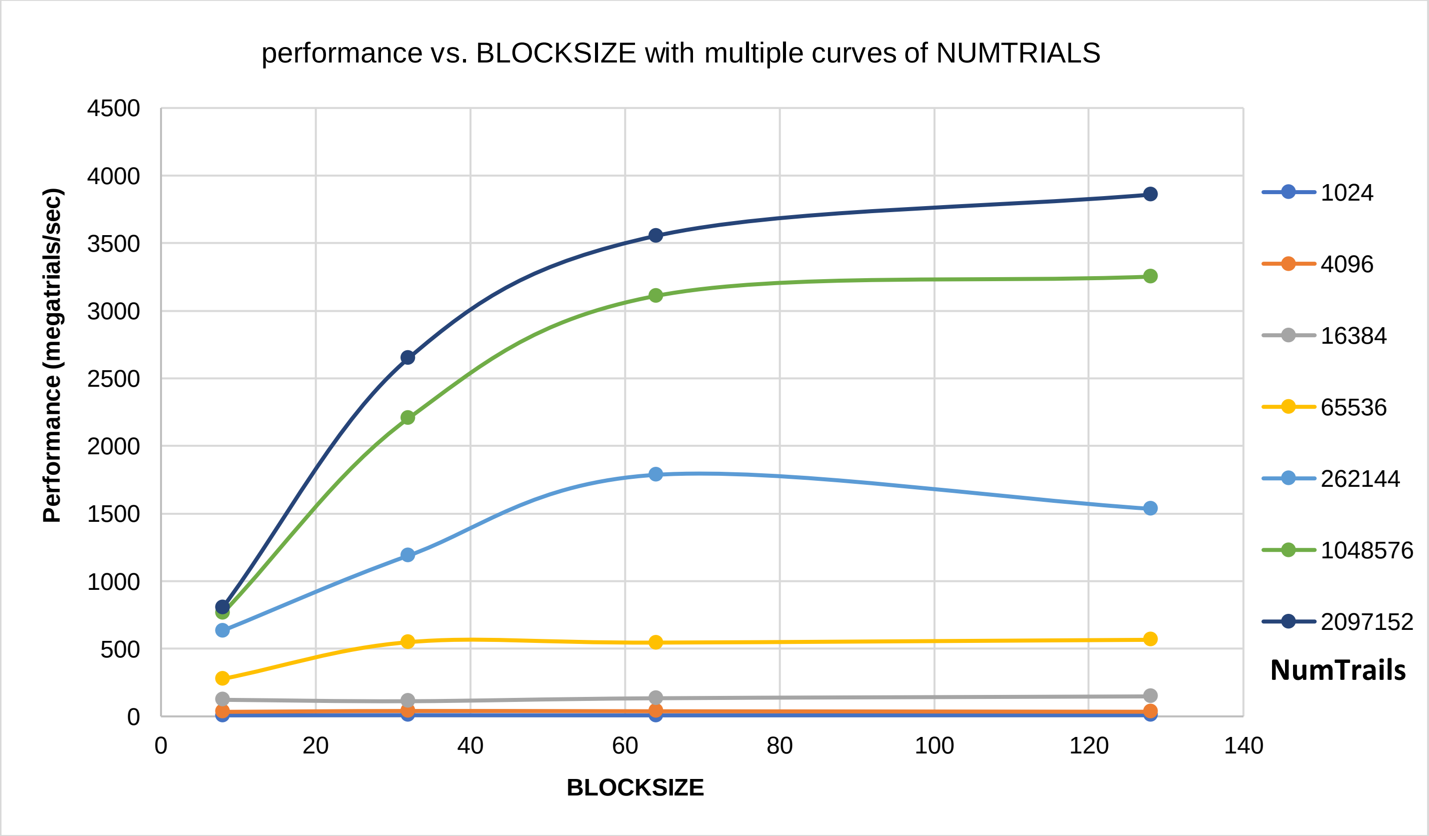
2. What do you think this new probability is?

   The actual probability is 74.66%. This is by selecting the probabilities of thread 128 for maximum number of trails i.e., 2097152, hence the probability of pins fitting into the hole is 74.66%.

3. Show the table and the two graphs.

Number of Trials	Blocksize	MegaTrials/Second	Probability
1024	8	7.6702	75.20%
1024	32	9.2834	73.63%
1024	64	7.8412	74.32%
1024	128	9.305	76.56%
4096	8	33.7642	74.44%
4096	32	40.3023	75.51%
4096	64	38.152	75.20%
4096	128	34.5666	74.34%
16384	8	122.6641	75.73%
16384	32	112.2807	75.03%
16384	64	134.2423	74.07%
16384	128	147.9769	74.66%
65536	8	277.6196	74.77%
65536	32	549.0617	74.52%
65536	64	545.8422	74.78%
65536	128	566.5283	74.79%
262144	8	634.7435	74.72%
262144	32	1188.9695	74.75%
262144	64	1786.6957	74.60%
262144	128	1534.9448	74.57%
1048576	8	764.6963	74.76%
1048576	32	2204.2244	74.77%
1048576	64	3109.8034	74.71%
1048576	128	3252.7298	74.78%
2097152	8	806.5076	74.63%
2097152	32	2648.4542	74.72%
2097152	64	3553.8205	74.69%
2097152	128	3859.1449	74.66%


performance vs. NUMTRIALS with multiple curves of BLOCKSIZE
Performance (megatrialas/sec)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
500000
1000000
1500000
2000000
2500000
NUMTRAILS
8
32
64
128
BLOCKSIZE


performance vs. BLOCKSIZE with multiple curves of NUMTRIALS
Performance (megatrials/sec)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100
120
140
BLOCKSIZE
1024
4096
16384
65536
262144
1048576
2097152
NumTrails

4. What patterns are you seeing in the performance curves?
   a) **For the graph, performance vs. NUMTRIALS:**
      For each block size (8, 32, 64), there's a general upward trend in MegaTrials/Second as the number of trials rises. The performance for block size 8 improved up to a certain number of NumTrails, then remained steady for the rest of the execution. The other block sizes, 32, 64, and 128, saw performance improvements as the number of trails grew. Among them, block size 128 reached the highest performance, demonstrating that performance indeed escalates with an increase in block size.

   b) **For the graph, performance vs. BLOCKSIZE:**
      The performance of 1024 and 4096 Numtrials stays consistent throughout the process, whereas there's a slight improvement in performance for 16384 and 65536 Numtrials. However, for all the remaining Numtrials, performance significantly escalates as the number of blocks increases.

5. Why do you think the patterns look this way?

   As the quantity of trials escalates, each processor should be allocated a higher number of blocks. The block size remains constant, but as the data volume grows, the workload increases, leading to enhanced performance. Additionally, as the block size expands, so does the thread size, contributing to further performance improvement.

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

   Compared to the other two block sizes, the performance of block size 8 is noticeably lesser. This can be attributed to its size being smaller than 32, while the other three block sizes, 32, 64 and 128, are multiples of 32. This allows for maximum performance through full utilization with these larger block sizes.

7. How do these performance results compare with what you got in Project #1? Why?

   When comparing the current project's performance with that of Project #1, it's apparent that the present project is significantly greater. This project utilizes CUDA, a parallel computing platform and API model created by NVIDIA. This technology takes advantage of NVIDIA's own graphics processing units (GPUs) to enhance computational performance.

8. What does this mean for what you can do with GPU parallel computing?

   You can get the most out of GPU parallel computing when you're seeing peak performance. Having more data enhances performance since it allows full utilization of the threads and prevents them from just sitting idle. Now, the data we select is typically in multiples of 32, and this ensures that each block size gets its own full queue in a single instance. But there's a catch - the block size can only get so big. If it hits its maximum, pushing it further won't give any performance improvements.