CUDA: Monte Carlo Simulation

Jai Reddy Kukunuru

1. **Tell what machine you ran this on**

   Nvidia DGX system

2. **What do you think this new probability is?**

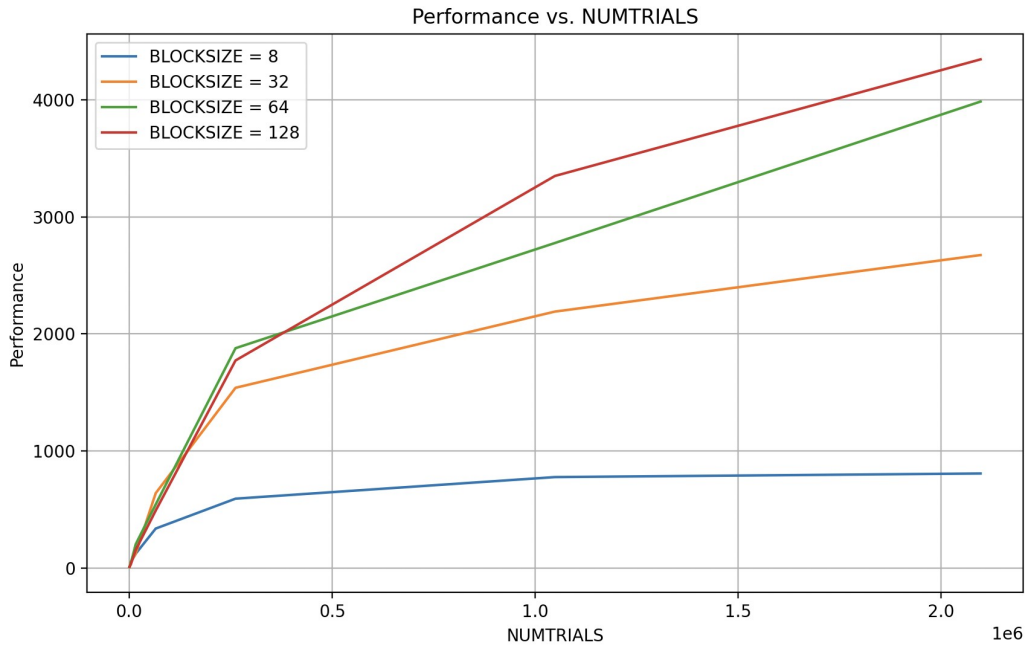
   74.56%

3. **Show the table and the two graphs**

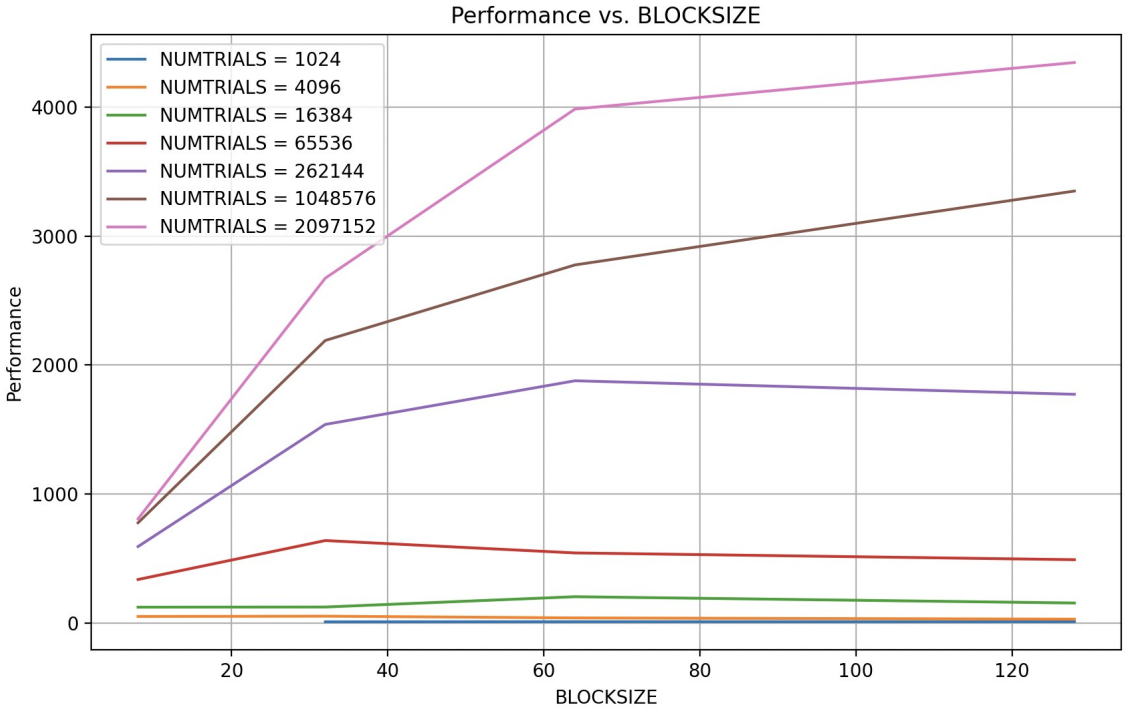
NumTrials	Blocksize	Performance	Probability
1024	32	8.4589	73.14
1024	64	9.4451	76.37
1024	128	9.409	74.12
4096	8	49.8637	74.76
4096	32	52.0114	74.66
4096	64	38.659	74.61
4096	128	28.907	74.93
16384	8	121.9048	74.83
16384	32	123.1361	75.1
16384	64	203.094	74.32
16384	128	154.2169	74.55
65536	8	336.5653	74.31
65536	32	638.8023	74.58

65536	64	542.8041	74.57
65536	128	490.3041	74.7
262144	8	592.1214	74.58
262144	32	1539.8496	74.6
262144	64	1878.4683	74.59
262144	128	1773.5441	74.72
1048576	8	776.5297	74.69
1048576	32	2191.2532	74.77
1048576	64	2777.4199	74.71
1048576	128	3350.5113	74.77
2097152	8	807.2427	74.67
2097152	32	2674.7204	74.67
2097152	64	3986.8597	74.69
2097152	128	4347.6185	74.68

Graph1: Performance vs Numtrials



Graph 2: Performance vs Blocksize



4. **What patterns are you seeing in the performance curves?**

Performance tends to increase as the number of trials (NUMTRIALS) increases: Looking at each block size (e.g., 8, 32, 64, 128), as the NUMTRIALS value increases (from 1024 to 2097152), the performance generally improves. This pattern is evident in most cases, where larger NUMTRIALS values lead to higher performance values (measured in megaTrials/second).

Performance varies with the block size (BLOCKSIZE): For a fixed NUMTRIALS value, the performance differs depending on the block size used. In general, larger block sizes tend to yield higher performance. However, the relationship between block size and performance is not strictly linear or consistent across all cases. There are instances where a smaller block size achieves higher performance than a larger block size.

Optimal performance is achieved at certain combinations of NUMTRIALS and BLOCKSIZE: By comparing the performance values for each combination of NUMTRIALS and BLOCKSIZE, we can identify some configurations that yield higher performance. For example, in the given data, the combination of NUMTRIALS=16384 and BLOCKSIZE=32 achieves a relatively high performance of 123.1361 megaTrials/second. Similarly, NUMTRIALS=1048576 and BLOCKSIZE=32 achieves a performance of 2191.2532 megaTrials/second.

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

Workload distribution: The Monte Carlo simulation involves executing a large number of independent trials. By increasing the number of trials (NUMTRIALS), more work is available for parallel execution, allowing the GPU to utilize its processing power more effectively. As a result, the performance tends to increase with higher NUMTRIALS values.

Thread utilization: The performance of CUDA programs often depends on how effectively the GPU threads are utilized. Larger block sizes (BLOCKSIZE) allow for more parallelism by launching more threads per block. This can lead to better utilization of the GPU's computational resources, resulting in improved

performance. However, excessively large block sizes may cause inefficient memory access patterns or resource contention, leading to suboptimal performance.

Memory access patterns: The efficiency of memory access patterns can impact the overall performance. In CUDA programs, accessing global memory incurs higher latency compared to shared memory or registers. When the workload is divided among threads within a block, the memory access patterns can influence the memory coalescing and caching efficiency. Optimal memory access patterns can lead to improved performance, while inefficient access patterns can result in increased memory latency and reduced performance.

GPU architecture: The underlying architecture of the GPU plays a crucial role in determining the performance characteristics. Factors such as the number of streaming multiprocessors (SMs), the number of cores per SM, memory hierarchy, and memory bandwidth can influence the performance. Different GPU architectures may exhibit varying performance trends based on their architectural features and capabilities.

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

Due to following reasons:

1. **Thread Divergence**: A smaller block size increases the likelihood of threads taking different execution paths, leading to inefficiencies and serialization of computation.
2. **Memory Latency**: Accessing global memory is slower than accessing shared memory or registers. With fewer threads actively accessing memory in a smaller block, the impact of memory latency becomes more significant.
3. **Occupancy and Resource Utilization**: Smaller block sizes may underutilize the available computational resources, resulting in lower overall performance due to insufficient thread occupancy.
4. **Overhead**: Managing and coordinating smaller blocks incurs additional overhead, including synchronization, scheduling, and memory allocation, which can become relatively more significant.

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

GPU parallel computing offers superior performance and scalability compared to CPU parallel computing for highly parallelizable tasks. GPUs with their large number of cores excel at processing massive data sets and complex simulations, providing faster execution times. GPU parallel computing requires optimizing parameters specific to its architecture, while CPU parallel computing focuses on thread scheduling and cache utilization. Hybrid approaches combining GPU and CPU parallel computing can further optimize performance.