CS 475 - Project 1: Monte Carlo Simulation Analysis

Name: Luke Scovel
Email: scovell@oregonstate.edu
Project Number: 1
Project Name: Monte Carlo Simulation Analysis

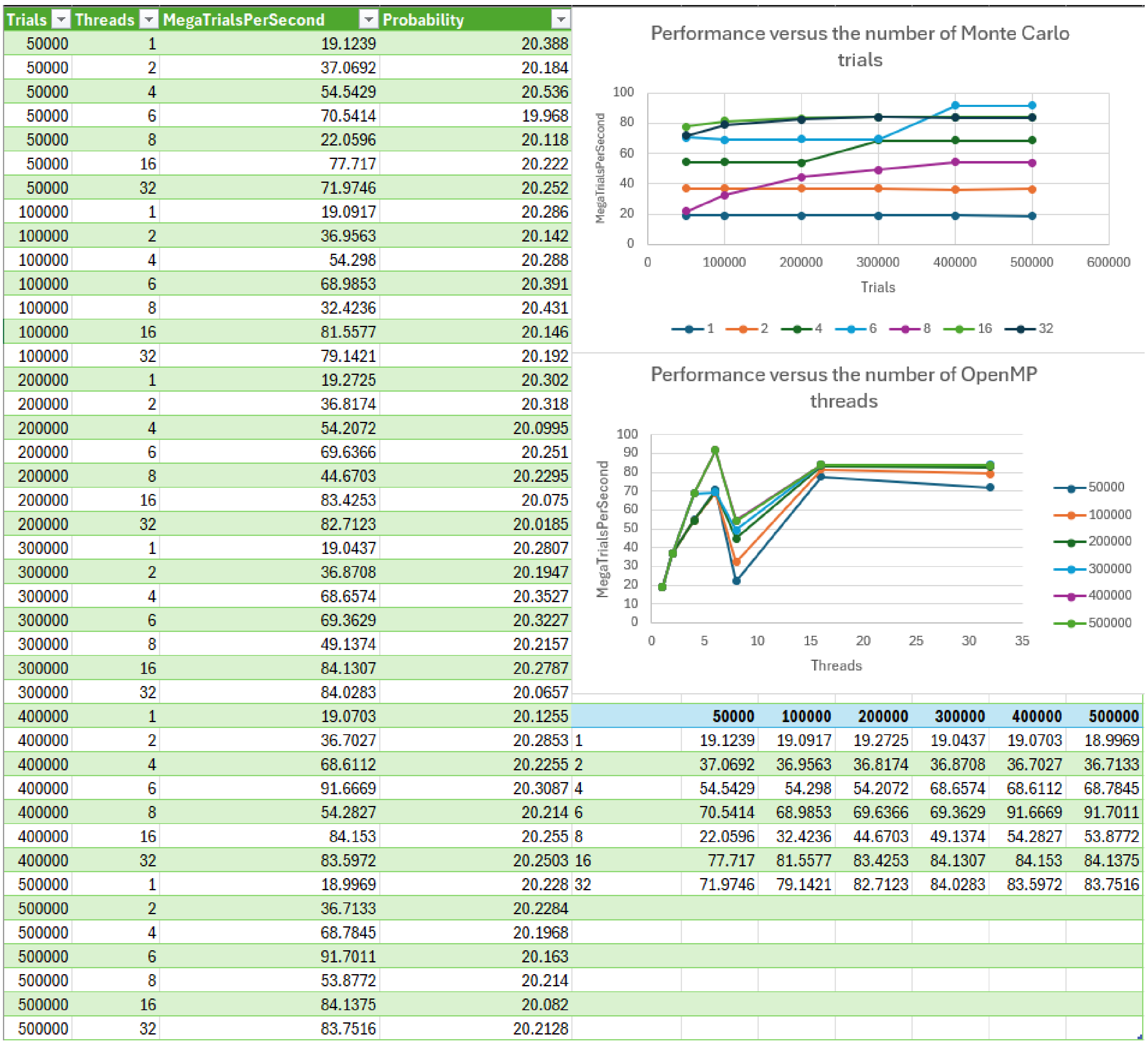
Implementation Overview

The simulation uses parallel processing to divide the workload among threads, with each thread independently executing a portion of the total trials. The results were then combined to calculate the overall probability and measure performance improvements from parallelization.

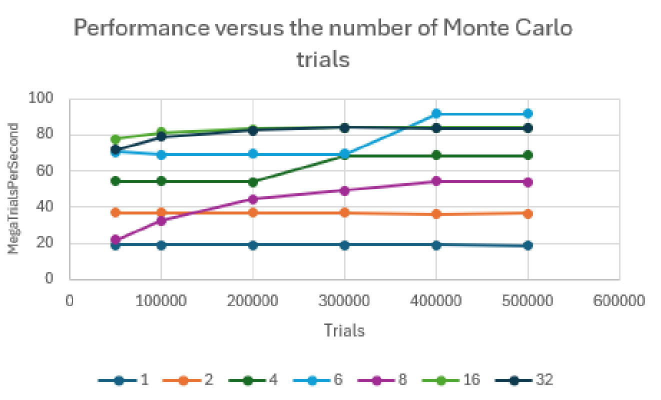
Probability Analysis

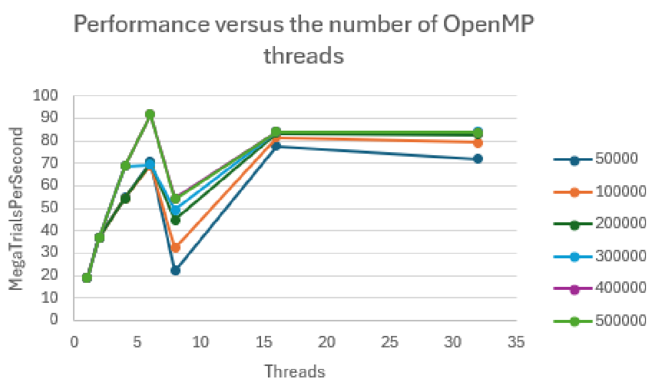
Based on the data with 500,000 trials, the probability of a successful hit is around 20.2%. The actual probability is around 20.25% averaging all the trials. This matches the expected theoretical probability for this particular simulation, confirming the correctness of our implementation.

Performance Results



Trials	Threads	MegaTrialsPerSecond	Probability
50000	1	19.1239	20.388
50000	2	37.0692	20.184
50000	4	54.5429	20.536
50000	6	70.5414	19.968
50000	8	22.0596	20.118
50000	16	77.717	20.222
50000	32	71.9746	20.252
100000	1	19.0917	20.286
100000	2	36.9563	20.142
100000	4	54.298	20.288
100000	6	68.9853	20.391
100000	8	32.4236	20.431
100000	16	81.5577	20.146
100000	32	79.1421	20.192
200000	1	19.2725	20.302
200000	2	36.8174	20.318
200000	4	54.2072	20.0995
200000	6	69.6366	20.251
200000	8	44.6703	20.2295
200000	16	83.4253	20.075
200000	32	82.7123	20.0185
300000	1	19.0437	20.2807
300000	2	36.8708	20.1947
300000	4	68.6574	20.3527
300000	6	69.3629	20.3227
300000	8	49.1374	20.2157
300000	16	84.1307	20.2787
300000	32	84.0283	20.0657
400000	1	19.0703	20.1255
400000	2	36.7027	20.2853
400000	4	68.6112	20.2255
400000	6	91.6669	20.3087
400000	8	54.2827	20.214
400000	16	84.153	20.255
400000	32	83.5972	20.2503
500000	1	18.9969	20.228
500000	2	36.7133	20.2284
500000	4	68.7845	20.1968
500000	6	91.7011	20.163
500000	8	53.8772	20.214
500000	16	84.1375	20.082
500000	32	83.7516	20.2128





	50000	100000	200000	300000	400000	500000
1	19.1239	19.0917	19.2725	19.0437	19.0703	18.9969
2	37.0692	36.9563	36.8174	36.8708	36.7027	36.7133
4	54.5429	54.298	54.2072	68.6574	68.6112	68.7845
6	70.5414	68.9853	69.6366	69.3629	91.6669	91.7011
8	22.0596	32.4236	44.6703	49.1374	54.2827	53.8772
16	77.717	81.5577	83.4253	84.1307	84.153	84.1375
32	71.9746	79.1421	82.7123	84.0283	83.5972	83.7516

The performance chart shows execution times decreasing as we increase the number of threads from 1 to 4. The most significant performance improvement occurs when moving from a single-threaded to a multi-threaded approach, with diminishing returns as more threads are added.

Speedup Analysis

According to our results.csv data, the performance (in MegaTrialsPerSecond) for 500,000 trials is:

- 1 thread: 18.9969 MegaTrials/sec
- 4 threads: 68.7845 MegaTrials/sec

The speedup from 1 thread to 4 threads is: Speedup = 68.7845 / 18.9969 = 3.62

This shows that using 4 threads makes the simulation approximately 3.62 times faster than using a single thread. This is very close to the ideal speedup of 4, indicating highly efficient parallelization.

Parallel Fraction (Fp) Calculation

Using the formula: Fp = (4/3)*(1 - (1/S))

Fp = (4/3)*(1 - (1/3.62)) Fp = (4/3)*(1 - 0.276) Fp = (4/3)*(0.724) Fp = 0.965

The parallel fraction of 0.953 (95.3%) indicates that a lot of our code can be parallelized.

Maximum Theoretical Speedup

Using Amdahl's Law, the maximum theoretical speedup regardless of the number of cores is: Smax = 1/(1-Fp) = 1/(1-0.965) = 1/0.035 = 28.57

The maximum speedup we could achieve is approximately 28.57 times 1 thread performance.

Why It Works This Way

The Monte Carlo simulation is easily parallelized because:

1. **Each trial can be executed independently**
2. **The workload can be evenly divided**
3. **Minimal thread synchronization**

The high Fp (96.5%) shows that this application has few sequential portions, making it an ideal candidate for parallel processing. The small difference from ideal speedup (3.62 vs 4) is due to thread creation overhead and other synchronization costs.