CS 575 Parallel Programming

Project #1

OpenMP: Monte Carlo Simulation

Name: Rachel Xing (#934229189)

Email Address: xingru@oregonstate.edu

Description: This project uses OpenMP to parallelize a Monte Carlo simulation of hitting the castle on the top of a cliff from the cannon controlled by an amateur band of mercenaries. The iterations for each set of trials are parallelized, excluding macro definitions, parameter initialization, and the repetition of the simulation itself. The program runs on the OSU Flip3 machine. To ensure reliable performance data, each trial set is run 30 times to record peak performance. The program is tested with thread counts of 1, 2, 4, 6, and 8, and trial sizes of 1, 10, 100, 1,000, 10,000, 100,000, and 500,000.

Commentary:

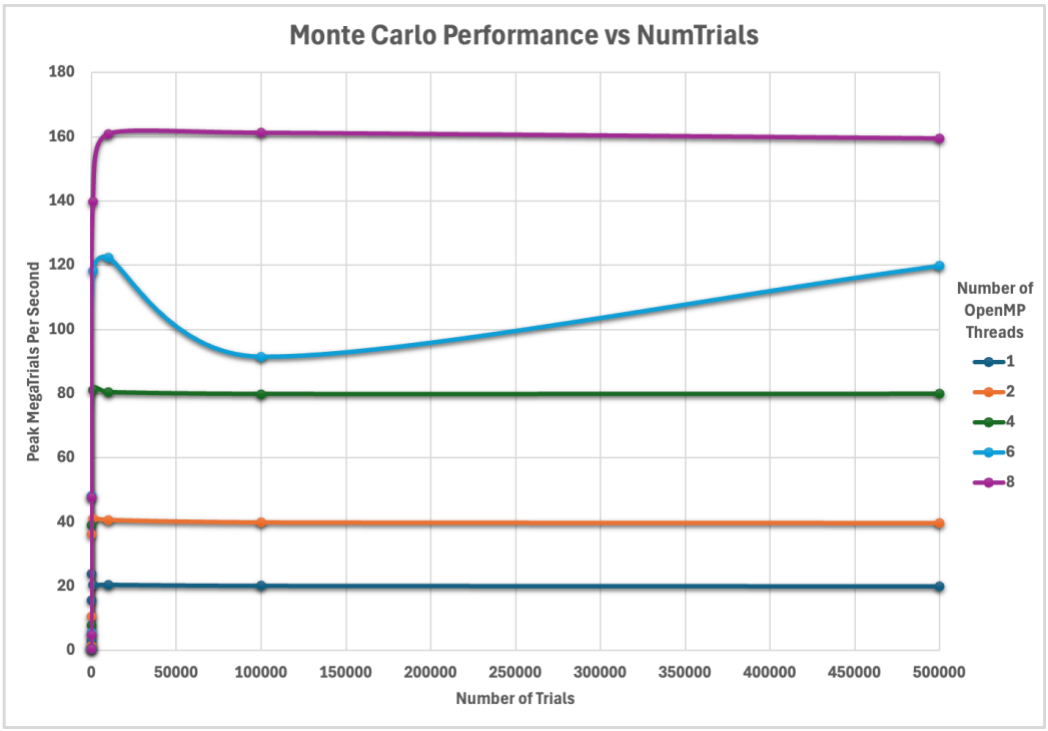
1. **Produce a rectangular table**

Performance Table (megatrials/sec)							
Trials# / Threads#	1	10	100	1000	10000	100000	500000
1	3.12	15.75	23.82	20.53	20.5	20.2	20.03
2	1.43	10.68	36.36	41.02	40.56	39.83	39.61
4	0.85	7.87	39.11	81.25	80.45	79.89	79.93
6	0.61	5.64	48.1	118.36	122.42	91.57	119.84
8	0.53	4.82	47.62	139.91	160.94	161.27	159.51

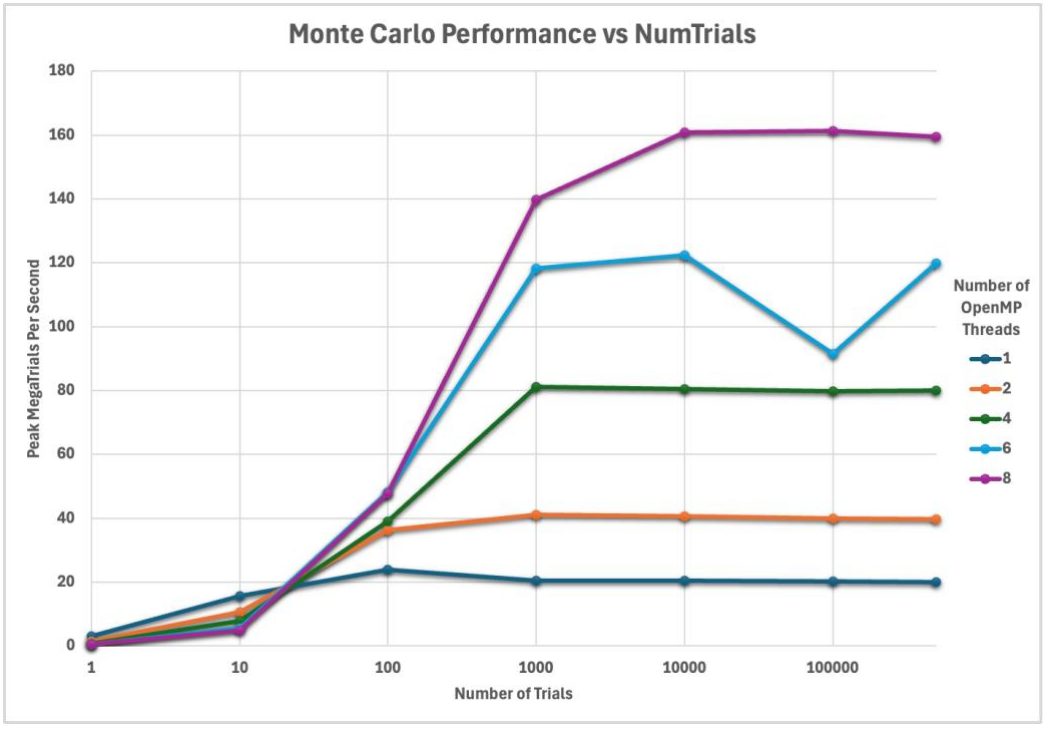
Probability Table (%)							
Trials# / Threads#	1	10	100	1000	10000	100000	500000
1	0	20	22	24.1	23	23.21	23.13
2	0	10	26	23.6	23.6	23.11	23.11
4	0	40	17	23.9	23.52	23.32	23.2
6	100	30	28	22	22.44	22.92	23.12
8	0	20	24	23	22.75	23.3	23.12

2. **Produce a graph of performance versus the number of Monte Carlo trials, with the colored lines being the number of OpenMP threads.**

Using a linear Scale for number of trials:



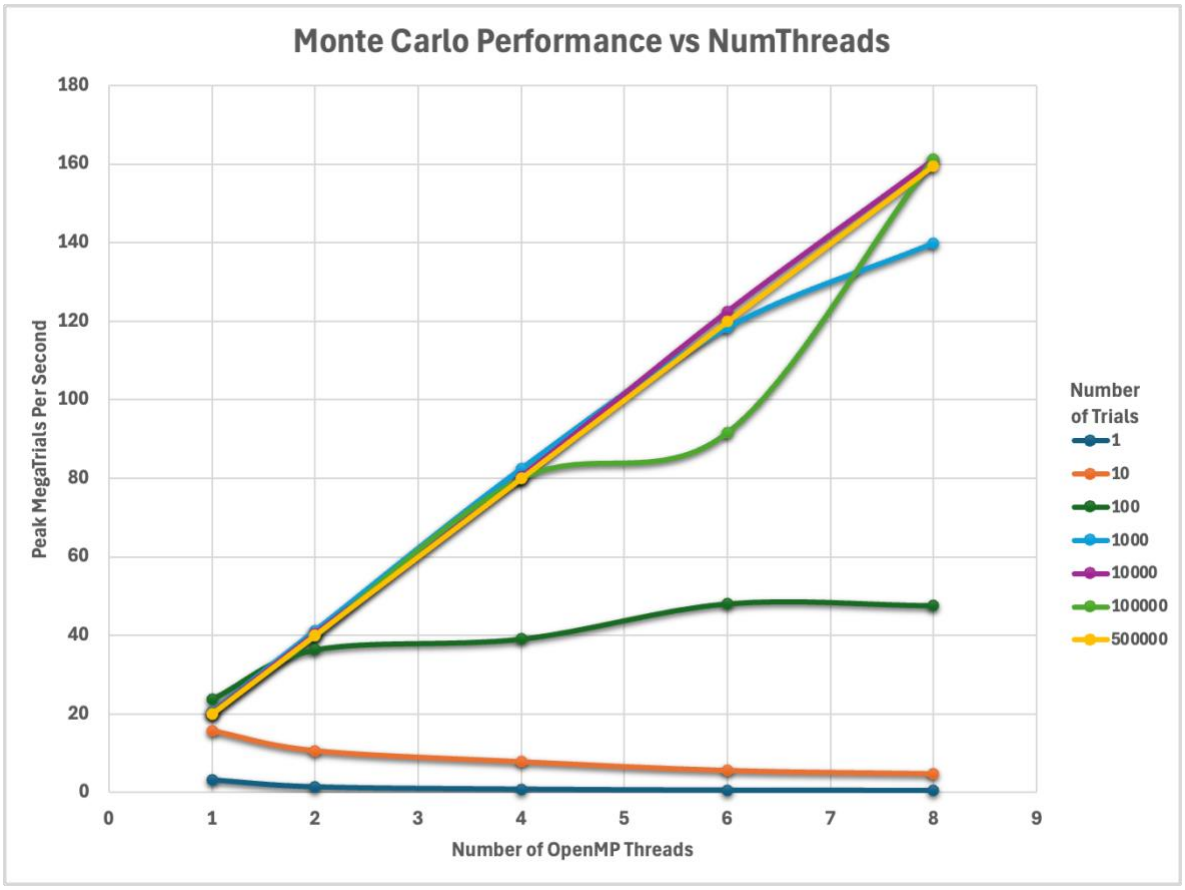
Using a logarithmic Scale for number of trials: I used a scatter plot with a straight line for this chart because the smoothing curve in Excel may unintentionally introduce concave-ups or concave-downs on the connecting curve that might be misleading.



Observation:

- We can observe in the chart with a linear scale that parallel performance increases in a logarithmic trend from 1 to 1,000 trials and then stabilizes. This aligns with Amdahl's Law that increasing the parallelizable portion of the simulations by increasing the number of trials yields significant gains in performance up to a point. Once the parallel fraction becomes saturated, the additions of trials will not result in any more significant improvement in performance.
- In the lower range of trials (1 to 100), performance gain tends to decrease as the number of threads increases. This may suggest a potential issue of false sharing. The false sharing occurs in our code because multiple threads are attempting to access the same cache line even though they are accessing different elements of the same parameter array. Since we haven't discussed false sharing in-depth, this is just a preliminary observation, and no solution is available at this time.
- There is an outlier, which is a sudden drop in the stabilized performance when the number of threads is 6 and the number of trials is 100,000.

3. Produce a graph of performance versus the number OpenMP threads, with the colored lines being the number of Monte Carlo trials.



Observation:

- We can observe that performance gains increase with the number of threads when the number of trials is $\geqslant 100$. That is because, as the number of trials increases, the parallelizable workload becomes large enough to take full advantage of parallelism, leading to more significant performance gains. By contrast, with a low number of trials, there is little performance gain or even performance loss because only a small portion of the workload is parallelizable. In these cases, the overhead of parallelism can exceed the gain, resulting in an unexpected decrease in overall performance as the number of threads increases.

4. **Choose one of the runs (the one with the maximum number of trials would be good), tell me what you think the actual probability is.**

   With more trials in the simulation, the result becomes closer to the real-world probability according to the statistics. There are five simulations with the largest number of trials (50,000 trials), each with a different number of threads. Their probability results range from a minimum of 23.11% to a maximum of 23.20%. Based on this, I have chosen **the run with 50,000 trials and 1 thread**, and I believe its probability of **23.13% should reflect the actual probability** because it is the closest to the average of the five runs ($\frac{23.13\%+23.11\%+23.2\%+23.12\%+23.12\%}{5} = 23.136\%$).

5. **Compute Fp, the Parallel Fraction, for this computation.**

   Using Amdahl's law, we can obtain this equation for parallel fraction (Fp) based on the speedup and the number of cores (n):

$$(Amdahl's\ law)\ Speedup_n = \frac{1}{\frac{F}{n} + (1-F)} \Longrightarrow F_p = \frac{n}{n-1} * (1 - \frac{1}{Speedup_n})$$

   After substituting $Speedup_n = \frac{T_1}{T_n} = \frac{P_n}{P_1}$, we get:

$$F_{p,n} = \frac{n}{n-1} * (1 - \frac{1}{P_n/P_1}) = \frac{n}{n-1} * (1 - \frac{P_1}{P_n})$$

   [Since performance gains from parallel programming start to show when the number of trials ≥ 100, I will exclude the runs with 1 and 10 trials from the calculation of the parallel fraction.]

   When the number of trials = 100:

$$F_{p,n=2} = \frac{2}{2-1} * \left(1 - \frac{P_1}{P_2}\right) = \frac{2}{1} * \left(1 - \frac{23.82\ megatrials/sec}{36.36\ megatrials/sec}\right) = 0.68977$$

$$F_{p,n=4} = \frac{4}{4-1} * \left(1 - \frac{P_1}{P_4}\right) = \frac{4}{3} * \left(1 - \frac{23.82\ megatrials/sec}{39.11\ megatrials/sec}\right) = 0.52126$$

$$F_{p,n=6} = \frac{6}{6-1} * \left(1 - \frac{P_1}{P_4}\right) = \frac{6}{5} * \left(1 - \frac{23.82\ megatrials/sec}{48.10\ megatrials/sec}\right) = 0.60574$$

$$F_{p,n=8} = \frac{8}{8-1} * \left(1 - \frac{P_1}{P_8}\right) = \frac{8}{7} * \left(1 - \frac{23.82\ megatrials/sec}{47.62\ megatrials/sec}\right) = 0.57119$$

When the number of trials = 1,000:

$$F_{p,n=2} = \frac{2}{2-1} * \left(1 - \frac{P_1}{P_2}\right) = \frac{2}{1} * \left(1 - \frac{20.53\ megatrials/sec}{41.02\ megatrials/sec}\right) = 0.99902$$

$$F_{p,n=4} = \frac{4}{4-1} * \left(1 - \frac{P_1}{P_4}\right) = \frac{4}{3} * \left(1 - \frac{20.53\ megatrials/sec}{81.25\ megatrials/sec}\right) = 0.99643$$

$$F_{p,n=6} = \frac{6}{6-1} * \left(1 - \frac{P_1}{P_4}\right) = \frac{6}{5} * \left(1 - \frac{20.53\ megatrials/sec}{118.36\ megatrials/sec}\right) = 0.99186$$

$$F_{p,n=8} = \frac{8}{8-1} * \left(1 - \frac{P_1}{P_8}\right) = \frac{8}{7} * \left(1 - \frac{20.53\ megatrials/sec}{139.91\ megatrials/sec}\right) = 0.97516$$

When the number of trials = 10,000:

$$F_{p,n=2} = \frac{2}{2-1} * \left(1 - \frac{P_1}{P_2}\right) = \frac{2}{1} * \left(1 - \frac{20.50\ megatrials/sec}{40.56\ megatrials/sec}\right) = 0.98915$$

$$F_{p,n=4} = \frac{4}{4-1} * \left(1 - \frac{P_1}{P_4}\right) = \frac{4}{3} * \left(1 - \frac{20.50\ megatrials/sec}{80.45\ megatrials/sec}\right) = 0.99358$$

$$F_{p,n=6} = \frac{6}{6-1} * \left(1 - \frac{P_1}{P_4}\right) = \frac{6}{5} * \left(1 - \frac{20.50\ megatrials/sec}{122.42\ megatrials/sec}\right) = 0.99905$$

$$F_{p,n=8} = \frac{8}{8-1} * \left(1 - \frac{P_1}{P_8}\right) = \frac{8}{7} * \left(1 - \frac{20.50\ megatrials/sec}{160.94\ megatrials/sec}\right) = 0.99728$$

When the number of trials = 100,000:

$$F_{p,n=2} = \frac{2}{2-1} * \left(1 - \frac{P_1}{P_2}\right) = \frac{2}{1} * \left(1 - \frac{20.20\ megatrials/sec}{39.83\ megatrials/sec}\right) = 0.98569$$

$$F_{p,n=4} = \frac{4}{4-1} * \left(1 - \frac{P_1}{P_4}\right) = \frac{4}{3} * \left(1 - \frac{20.20\ megatrials/sec}{79.89\ megatrials/sec}\right) = 0.99620$$

$$F_{p,n=6} = \frac{6}{6-1} * \left(1 - \frac{P_1}{P_4}\right) = \frac{6}{5} * \left(1 - \frac{20.20\ megatrials/sec}{91.57\ megatrials/sec}\right) = 0.93528$$

$$F_{p,n=8} = \frac{8}{8-1} * \left(1 - \frac{P_1}{P_8}\right) = \frac{8}{7} * \left(1 - \frac{20.20 megatrials/sec}{161.27\ megatrials/sec}\right) = 0.99971$$

When the number of trials = 500,000:

$$F_{p,n=2} = \frac{2}{2-1} * \left(1 - \frac{P_1}{P_2}\right) = \frac{2}{1} * \left(1 - \frac{20.03\ megatrials/sec}{39.61\ megatrials/sec}\right) = 0.98864$$

$$F_{p,n=4} = \frac{4}{4-1} * \left(1 - \frac{P_1}{P_4}\right) = \frac{4}{3} * \left(1 - \frac{20.03\ megatrials/sec}{79.93\ megatrials/sec}\right) = 0.99921$$

$$F_{p,n=6} = \frac{6}{6-1} * \left(1 - \frac{P_1}{P_4}\right) = \frac{6}{5} * \left(1 - \frac{20.03\ megatrials/sec}{119.84\ megatrials/sec}\right) = 0.99943$$

$$F_{p,n=8} = \frac{8}{8-1} * \left(1 - \frac{P_1}{P_8}\right) = \frac{8}{7} * \left(1 - \frac{20.03\ megatrials/sec}{159.51\ megatrials/sec}\right) = 0.99935$$

6. Given this Fp, what is the maximum Speedup you could *ever* get, no matter how many cores you use?

According to the lecture, the theoretical maximum speedup can be estimated by the following formula:

$$\max(Speedup) = \lim_{n\to\infty} Speedup = \frac{1}{1 - F_p}$$

To correctly calculate it for each number of trials, I first average the F_p values obtained across different number of threads. The calculation excludes any outliers. This can ensure a more accurate estimation of F_p for each trial size to better estimate the maximum Speedup.

When the number of trials = 100:

$$\overline{F_p} = \frac{0.68977 + 0.52126 + 0.60574 + 0.57119}{4} = 0.59699$$

$$\mathbf{max(Speedup)_{\#\ of\ Trial=100}} = \frac{\mathbf{1}}{\mathbf{1 - 0.59699}} = \mathbf{2.48}$$

When the number of trials = 1,000:

$$\overline{F_p} = \frac{0.99902 + 0.99643 + 0.99186 + 0.97516}{4} = 0.99062$$

$$\mathbf{max(Speedup)_{\#\ of\ Trial=1000}} = \frac{\mathbf{1}}{\mathbf{1 - 0.99062}} = \mathbf{106.60}$$

When the number of trials = 10,000:

$$\overline{F_p} = \frac{0.98915 + 0.99358 + 0.99905 + 0.99728}{4} = 0.99477$$

$$\mathbf{max(Speedup)_{\#\ of\ Trial=10000}} = \frac{\mathbf{1}}{\mathbf{1 - 0.99477}} = \mathbf{191.20}$$

With the number of trials = 100,000:

$$\overline{F_p} = \frac{0.98569 + 0.99620 + 0.99971}{3} = 0.99387$$

$$\max(\boldsymbol{Speedup})_{\#\ of\ Trial=100000} = \frac{\mathbf{1}}{\mathbf{1 - 0.99387}} = \mathbf{163.13}$$

With the number of trials = 500,000:

$$\bar{F}_p = \frac{0.98864 + 0.99921 + 0.99943 + 0.99935}{4} = 0.99666$$

$$\max(\boldsymbol{Speedup})_{\#\ of\ Trial=500000} = \frac{\mathbf{1}}{\mathbf{1 - 0.99666}} = \mathbf{299.40}$$

Comment:

We can observe that as the number of trials increases, the parallel fraction Fp rises sharply first, from 0.597 at 100 trials to 0.991 at 1,000 trials, indicating that the workload has become highly parallel. After this point, Fp slowly approached 1.0 but never reached it as expected because the parallel fraction had become saturated. An exception occurs at 100,000 trials, where an unexpectedly decreased Fp appears. This outlier may be due to load imbalance because the CPU load average of the flip3 server was around 4.0 when I was running these simulations. Overall, the results are mostly consistent with the Gustafson-Barsis observation: (1) Fp grows with enlarged problem size (number of trials), and (2) the maximum speedup improves significantly as the simulation scale increases (from 2.48 at 100 trials to 299.40 at 500,000 trials).

[*Additional Graph*: 3D visualization of parallel performance with respect to the number of trials and threads in the Monte Carlo simulation]