Project #1 – OpenMP: Monte Carlo Simulation

Hyun-taek Oh
ohhyun@oregonstate.edu
16 April 2024

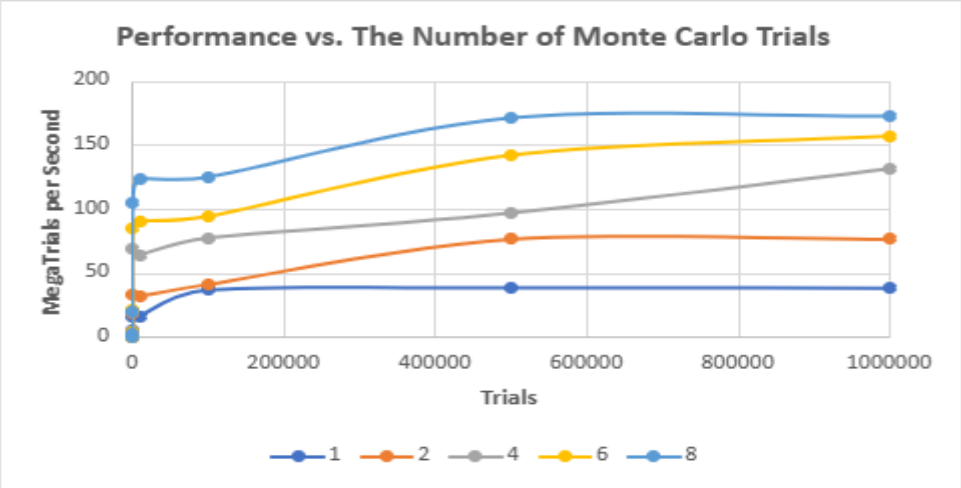
1. Data table of the performance numbers as a function of NUMT and NUMTRIALS.

Trials \ Threads	1	2	4	6	8
1	0.55	0.26	0.35	0.24	0.21
10	5.5	3.29	2.46	3.53	2.53
100	18.84	18.99	20.24	20.62	19.75
1000	15.89	32.52	69.51	84.22	104.23
10000	15.91	31.69	64.14	90.31	123.4
100000	36.49	40.76	77.14	94.04	124.76
500000	38.2	76.19	96.65	141.96	171.04
1000000	38.02	76.25	131.22	156.9	172.49

< **Figure 1.** The correlation between the number of trials and threads >

The data table above shows that the correlation between the number of threads and trials is related to the performance, which is measured by megatrials per second, indicating the increase in the number of threads results in better performance at the same number of the trials.

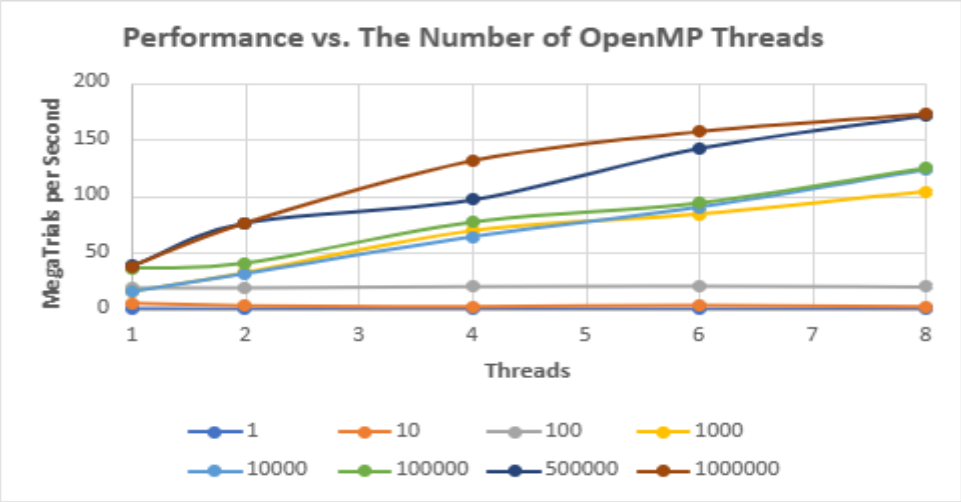
2. Performance versus the number of Monte Carlo trials, with the colored lines being the number of OpenMP threads.



< **Figure 2.** The performance versus the number of trials, with threads >

As shown in Figure 2, when the number of trials increases largely, each thread has a different performance. The case of 8 threads displays the best performance, which has 172.49 megatrials per second.

3. **Performance versus the number of OpenMP threads, with the colored lines being the number of Monte Carlo trials.**



< **Figure 3.** The performance versus the number of threads, with trials >

As shown in Figure 3, each thread number has a different performance, and the best result is manifested by the case of the 8 threads, indicating the more the number of trials increase, the better the threads perform.

4. **Estimate of the Probability.**

Threads \ Trials	1	10	100	1000	10000	100000	500000	1000000
1	0	0	51	56.2	56.62	57.13	57.06	57.03
2	100	40	51	54.1	55.86	57.03	57.07	57.03
4	100	60	64	59.3	56.75	56.99	56.95	57.01
6	100	40	62	57.9	56.32	57.12	57.07	57
8	100	20	40	52.7	56.32	57	57.01	57.01

< **Figure 4.** The estimated probability >

The table above shows the estimated probability, which is calculated by the division between the number of success and trials. The actual probability in the case of each thread in 1,000,000 trials is approximately 57%, meaning that the estimated probability may not be affected by the number of threads because the number of threads is not for increase in the probability of success but for the performance of data processing.

5. Estimate of the Parallel Fraction.

Before the calculation of the Parallel Fraction (P_F), we need to know the Speedup of each core. The Speedup equation is below.

$$Speedup_n = \frac{P_n}{P_1}$$

P_n is performance using n cores, and P_1 is performance using 1 core. So, we can get each core of $Speedup$. We focus on the $Speedup$ in the case of the 1,000,000 trials.

$$Speedup_1 = \frac{P_1}{P_1} = 1$$

$$Speedup_2 = \frac{P_2}{P_1} = \frac{76.25}{38.02} \approx 2.0$$

$$Speedup_4 = \frac{P_4}{P_1} = \frac{131.22}{38.02} \approx 3.5$$

$$Speedup_6 = \frac{P_6}{P_1} = \frac{156.9}{38.02} \approx 4.1$$

$$Speedup_8 = \frac{P_8}{P_1} = \frac{172.49}{38.02} \approx 4.5$$

According to the Lecture Note, the equation of the Parallel Fraction (P_F) is below.

$$P_F = \frac{n}{n-1} \times \frac{(Speedup - 1)}{Speedup}$$

If the number of cores is one, P_F is equal to zero because of denominators.

If the number of cores is two,

$$P_F = \frac{2}{2-1} \cdot \frac{(2.0 - 1)}{2.0} = 1$$

If the number of cores is four,

$$P_F = \frac{4}{4-1} \cdot \frac{(3.5-1)}{3.5} \approx 0.95$$

If the number of cores is six,

$$P_F = \frac{6}{6-1} \cdot \frac{(4.1-1)}{4.1} \approx 0.91$$

If the number of cores is eight,

$$P_F = \frac{8}{8-1} \cdot \frac{(4.5-1)}{4.5} \approx 0.89$$

When it comes to the results of the Parallel Fraction calculation, the number of cores and the Parallel Fraction have trade off relationships. It might be affected by the number of trials, but the actual value of P_F decreases.

6. Why do the graphs look the way they do? What are they telling you?

It is obvious that the increasing number of cores gives better performance when the number of trials is going to be large enough because the cores share the huge works and decrease execution time. We can assume that the increase in the number of cores leads to better computing processes linearly. However, the results of the experiment show that the number of cores does not always guarantee the best performance and execution time. In fact, there is a limitation on multi-core processors that cannot solve the sequential portion of the program. According to Amdahl's Law, the sequential portion of the program does not go away, and it also does not get any smaller. For example, at certain points, the performance of cores levels off. This may occur because of the sequential portion.