Result of the Monte Carlo experiment with multiple threads.

Metta Saketh Srinivasa Rao

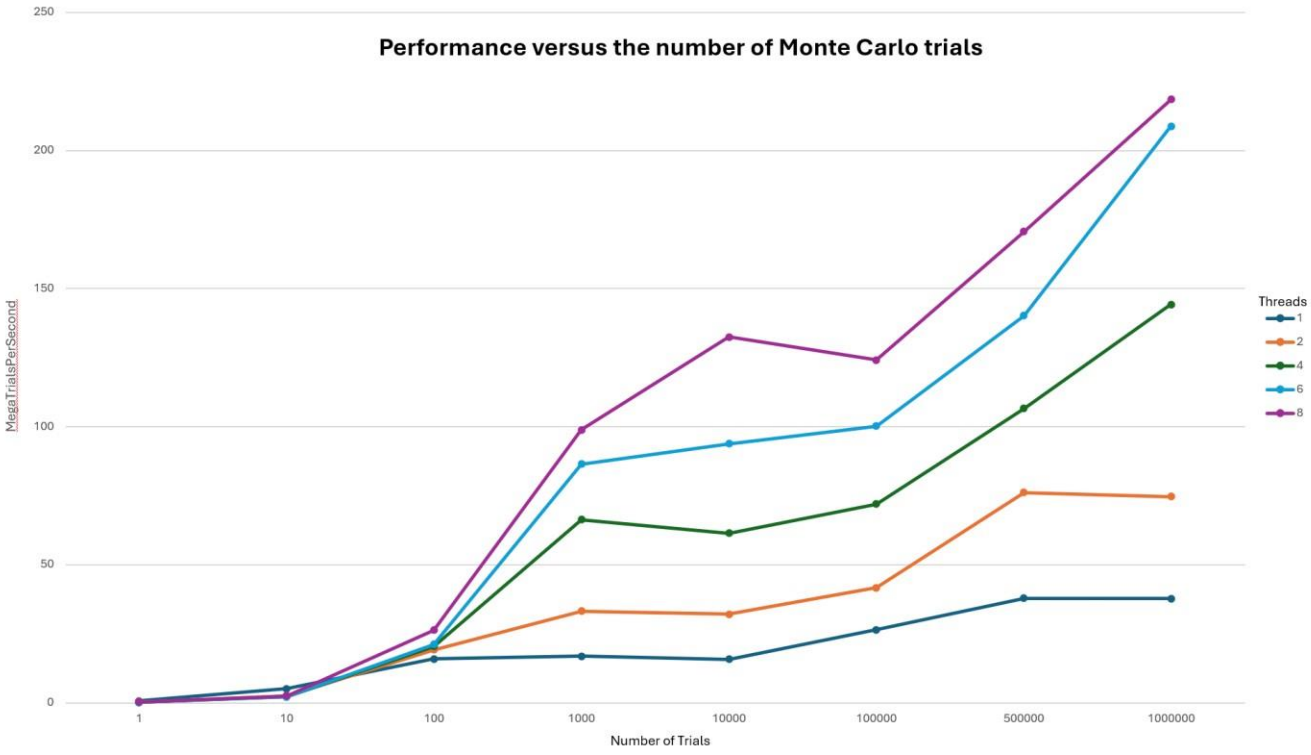
CS 575

mettas@oregonstate.edu

Below shows the table which contains the mega trails per second according to the number of threads and trials.

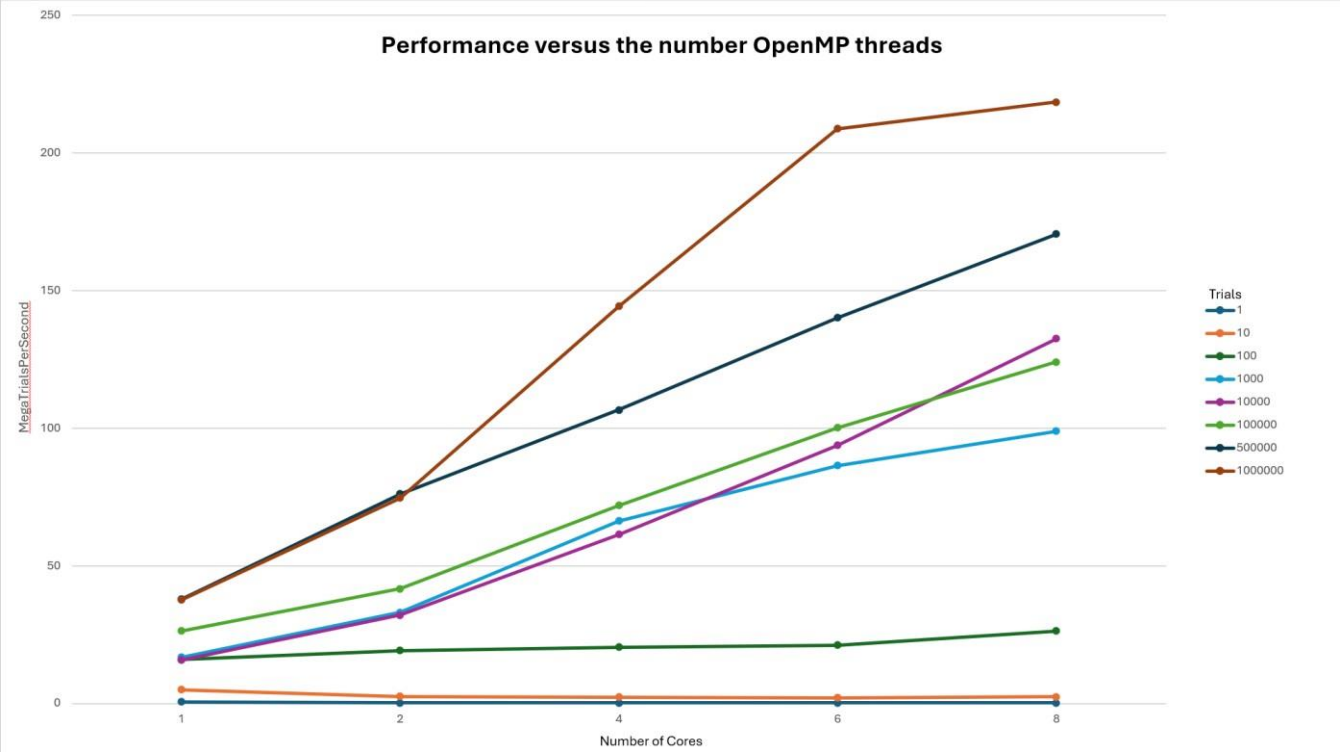
Number of Trials	Number of threads					
	1	2	4	6	8	Grand Total
1	0.64	0.26	0.24	0.21	0.26	1.61
10	5.09	2.57	2.33	2.14	2.48	14.61
100	15.93	19.21	20.44	21.2	26.33	103.11
1000	16.82	33.15	66.38	86.48	98.9	301.73
10000	15.81	32.14	61.46	93.79	132.48	335.68
100000	26.41	41.65	72.01	100.14	124.14	364.35
500000	37.9	76.14	106.63	140.2	170.6	531.47
1000000	37.7	74.71	144.25	208.81	218.43	683.9
Grand Total	**156.3**	**279.8**	**473.74**	**652.97**	**773.62**	**2336.46**

The below graph shows the performance of each of the trials with the number of threads according to the number of trails ran. Here The Y-axis represents the performance, and the X-axis represents the number of trials.



The above graph shows that the performance of all the cores is similar if not worse for all the trails that have the numbers 100 or less when compared with 1 thread, this is because of the overhead required when creating the different threads. But once we reach more than 100, we can see that the graphs diverge and show that the performance of 8 cores is better than 6 cores, and 6 cores is better than 4 cores and so on. We can also see that the performance of single core peaks at around 52 and just stays there and 2 threads peaks around 115. This shows the peak performance of both the 1-thread and 2thread parallel programming.

The below graph shows the performance of every trial that was run using different number of trials with respect to the number of threads. Here X-axis is the number of threads used and the Y-axis represents the performance.



From the above graph, we can see that number of Mega Trials decreases as we increase the number of threads when the number of trails is 100 or less. Only after 1000 trails or more does it show a better cost to performance ratio as we increase the number of threads. This is also due to the overhead required when creating new threads. We can see that for 1 and 10 trials, 1-thread has more efficiency and for 100 trails, all the parallel programs have the same performance. The increase in performance as we increase the number of threads is shown only after 1000 trials or more. This shows that sometimes, if the number of trials or calculations is less, then using less number of threads is more efficient than more threads.

From the different runs performed using varying number of threads and trails, the probability of the ball going into the hole is around 57.4 %.

The parallel function for the 8-thread run is calculated using the below formula:

Speedup = performance with 8-threads / performance with 1-thread = 281.22 / 56.48 = 4.97

Parallel Function = (8 / 7) * ((speedup - 1) / speedup) = (8 / 7) * ((4.97 - 1) / 4.97) = (8 / 7) * (3.97 / 4.97)
=
1.14 * 0.79 = 0.90

Parallel Function = 0.90

Using the Parallel Function, we can conclude that even if we throw hundreds of cores at this problem, we can only speed up the execution by up to 90%. As the rest 10% is the code that must be executed serially, and parallel programming cannot be used.