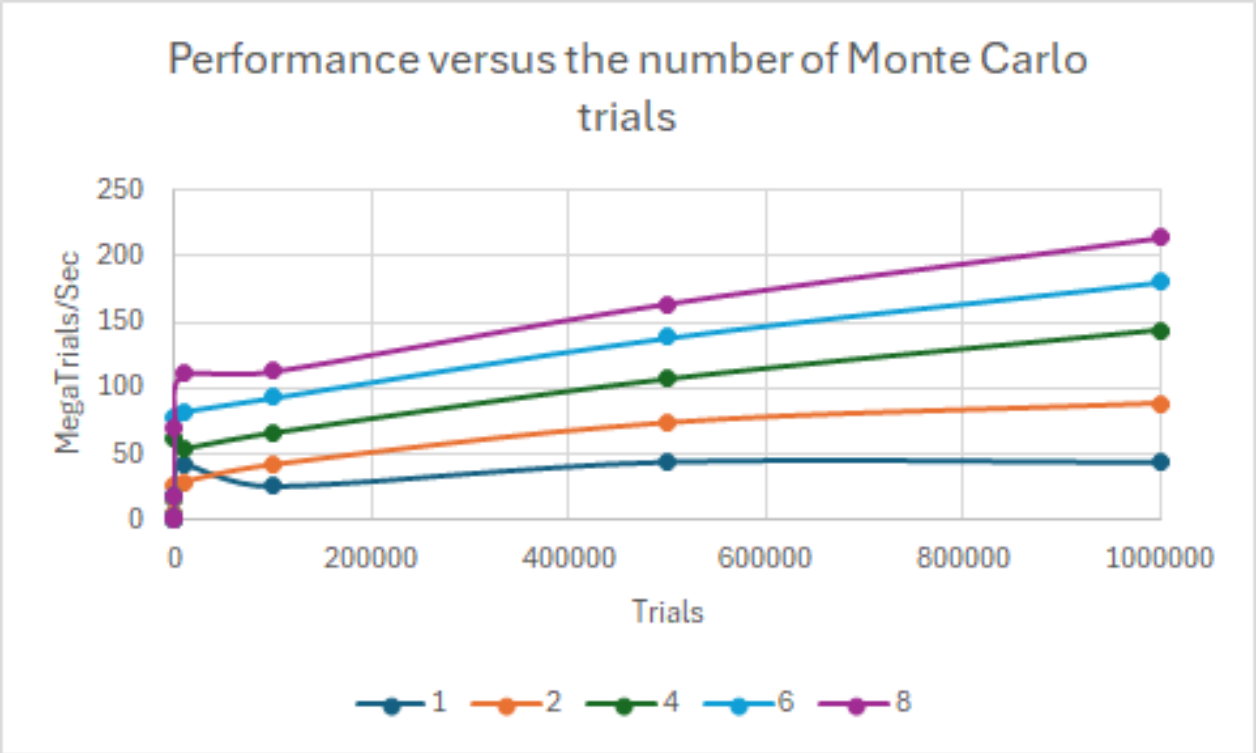
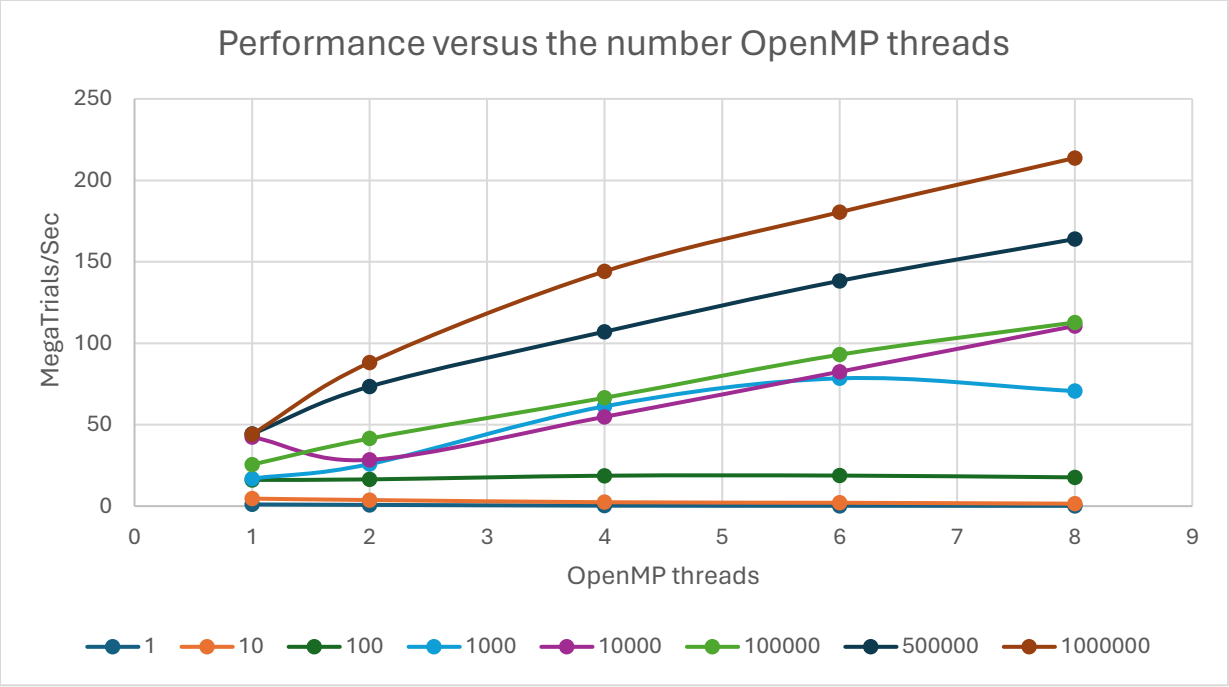
CS 475 – Parallel Programming

Jenny Zhong - zhongje@oregonstate.edu

Project #1 - OpenMP: Monte Carlo Simulation

	1	2	4	6	8
1	1.03	0.78	0.3	0.21	0.19
10	4.62	3.74	2.4	2.07	1.52
100	16.09	16.43	18.68	18.76	17.67
1000	16.91	25.81	61.26	78.44	70.62
10000	42.43	28.4	54.83	82.52	110.58
100000	25.51	41.49	66.5	92.97	112.77
500000	44.27	73.39	107.08	138.3	163.94
1000000	44.15	88.11	144.09	180.46	213.71





- Your estimate of the Probability. Based on a run with 6 cores/threads and 1000000 trials, the actual probability is likely 57%.
- Your estimate of the Parallel Fraction (*show your work!*).
  - $S(Speedup) = \frac{(Performance\ with\ six\ threads)}{(Performance\ with\ one\ thread)} = \frac{180.46\ MegaTrials/Sec}{44.15\ MegaTrials/Sec} = 4.09$
  - $F_p(Parallel\ Fraction) = \frac{n}{n-1} * \frac{Speedup-1}{Speedup} = \frac{6}{5} * \frac{4.09-1}{4.09} = 0.91$
  - Given this Parallel Fraction, the maximum speedup would be 100 even if we used hundreds of cores. The parallel fraction puts an upper bound on how much benefit you can get from adding more cores, according to Amdahl's Law.
- Your commentary: why do the graphs look the way they do? What are they telling you?
  - As the number of trials increases, an increase in number of cores yields better performance. The number of threads/cores and the number of trials has a positive correlation. Under Gustafson-Baris Observation, when using the same parallel program on larger datasets, the parallel fraction F_p increases.