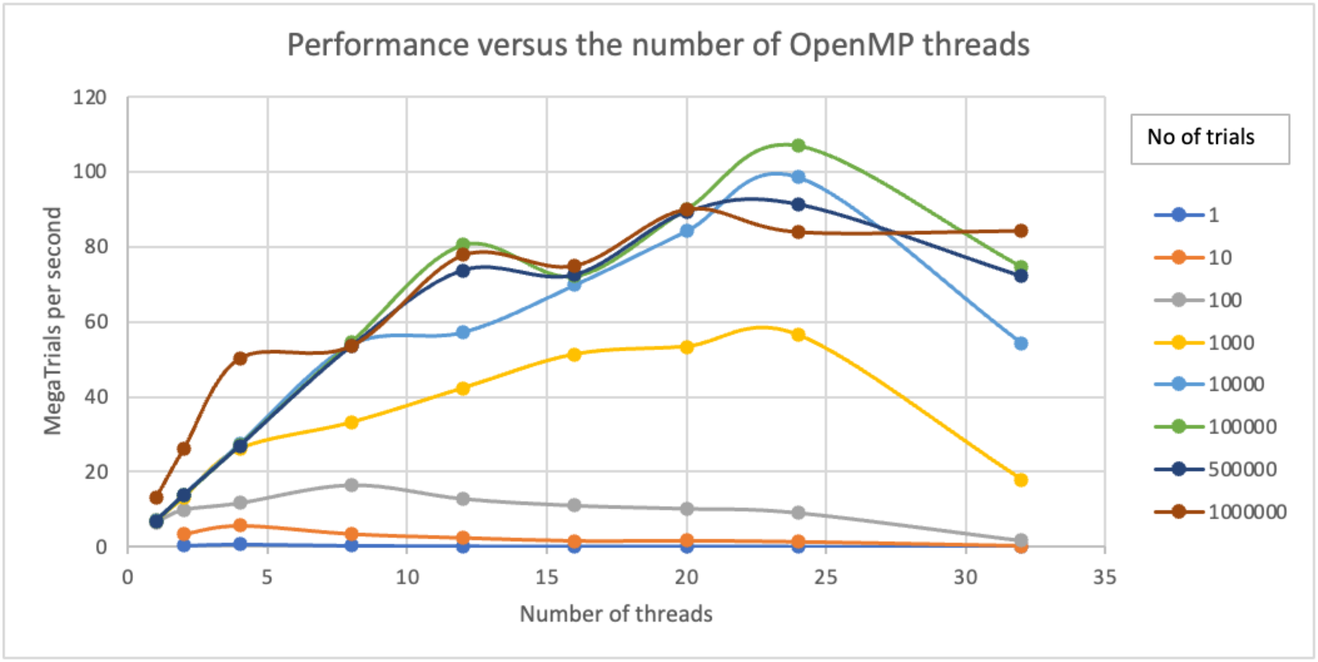
Name: Shreya Palit
Email: palits@oregonstate.edu
Project Name: OpenMP: Monte Carlo Simulation

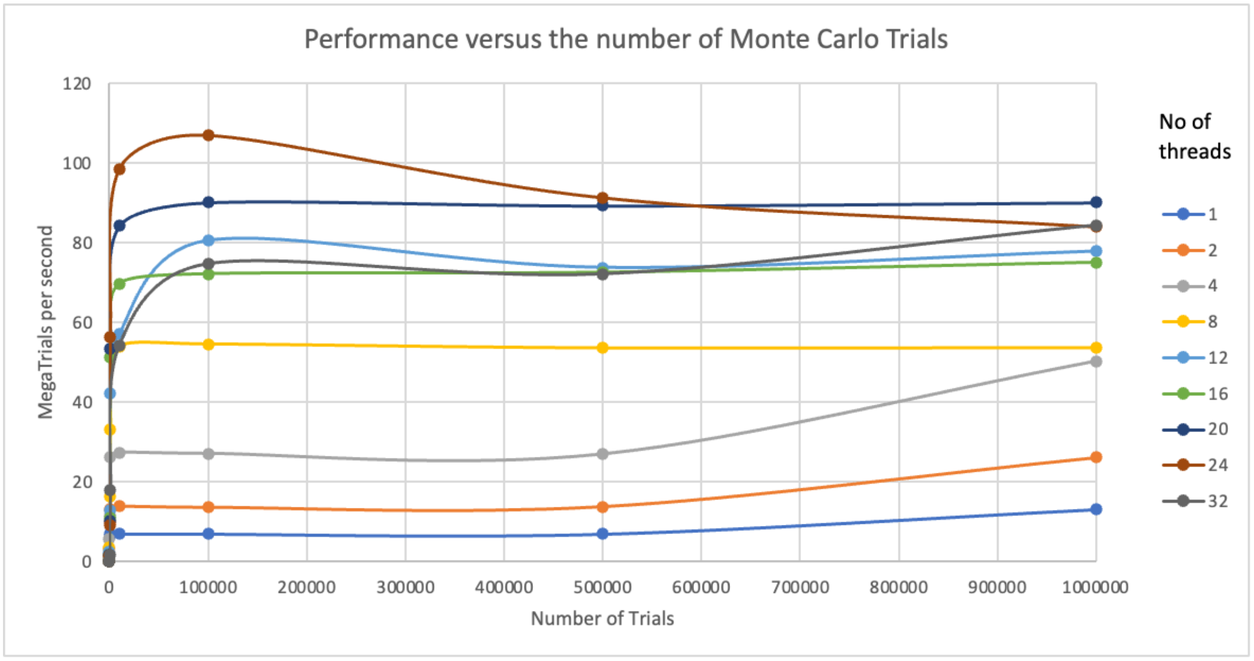
CS 575 - Project #1

Data Table:

	1	10	100	1000	10000	100000	500000	1000000
1			6.52	6.93	6.92	6.89	6.87	13.02
2	0.35	3.35	9.82	13.19	13.87	13.61	13.76	26.16
4	0.63	5.68	11.62	26.12	27.38	27.07	26.92	50.26
8	0.33	3.4	16.39	33.14	53.89	54.53	53.55	53.61
12	0.22	2.35	12.74	42.29	57.19	80.52	73.75	77.83
16	0.16	1.52	10.94	51.29	69.78	72.18	72.55	75.04
20	0.17	1.58	10.13	53.34	84.19	90.11	89.3	90.08
24	0.13	1.28	9	56.41	98.57	107.04	91.33	84.04
32	0.02	0.17	1.63	17.93	54.3	74.67	72.12	84.33

Two Performance Graphs:





Estimating Probability:

For all the threads - 1, 2, 4, 8, 12, 16, 20, 24, 32, I chose the probability result for the largest number of trials which is 1,000,000.

So the average probability = (26.84 + 26.89 + 26.89 + 26.9 + 26.92 + 26.82 + 26.92 + 26.86 + 26.82)/9 = 26.87

So, a **good estimate of the probability** is **26.9**.

Estimate of the Parallel Fraction:

In the largest number of trials 1,000,000,

1 thread performance = 13.02 MegaTrials/Sec

2 thread performance = 26.16 MegaTrials/Sec

4 thread performance = 50.26 MegaTrials/Sec

8 thread performance = 53.61 MegaTrials/Sec

12 thread performance = 77.83 MegaTrials/Sec

16 thread performance = 75.04 MegaTrials/Sec

20 thread performance = 90.08 MegaTrials/Sec

24 thread performance = 84.04 MegaTrials/Sec

32 thread performance = 84.33 MegaTrials/Sec

Now, Speedup = Performance with # of threads/ Performance with 1 thread

$Speedup_2$ = 26.16 / 13.02 = 2.01

$Speedup_4$ = 50.26 / 13.02 = 3.86

$Speedup_8$ = 53.61 / 13.02 = 4.12

$Speedup_{12}$ = 77.83 / 13.02 = 5.98

$Speedup_{16}$ = 75.04 / 13.02 = 5.76

$Speedup_{20}$ = 90.08 / 13.02 = 6.92

$Speedup_{24}$ = 84.04 / 13.02 = 6.45

$Speedup_{32}$ = 84.33 / 13.02 = 6.48

Now, $F_{Parallel}$ = n/(n-1)* (1- 1/Speedup) where n is the number of threads

$F_{Parallel2}$ = 2/1 * (1 - 1 / 2.01) = 1.00

$F_{Parallel4}$ = 4/3 * (1 - 1 / 3.86) = 0.99

$F_{Parallel8}$ = 8/7 * (1 - 1 / 4.12) = 0.87

$F_{Parallel12}$ = 12/11 * (1 - 1 / 5.98) = 0.91

$F_{Parallel16}$ = 16/15 * (1 - 1 / 5.76) = 0.88

$F_{Parallel20}$ = 20/19 * (1 - 1 / 6.92) = 0.90

$F_{Parallel24}$ = 24/23 * (1 - 1 / 6.45) = 0.88

$F_{Parallel32}$ = 32/31 * (1 - 1 / 6.48) = 0.87

So, a **good estimate of parallel fraction** is **0.91.**

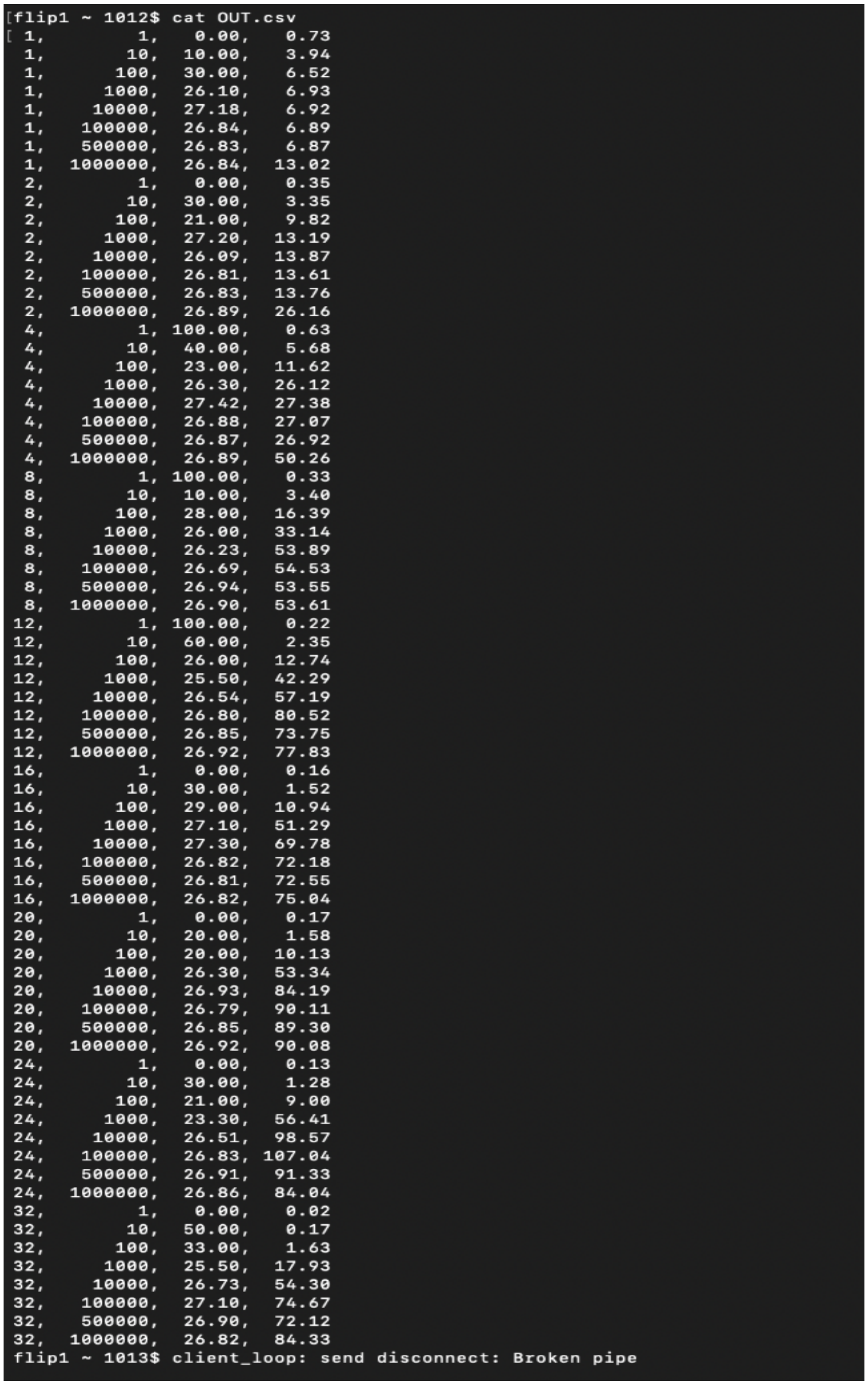
Commentary:

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

For the graph **Performance vs no of OpenMP threads**, as a general trend we can say that the performance generally increases as the number of trials and threads increase. However, for the number of trials: 1, 10 and 100, they do not have any increase in performance (megatrials/sec). Also it can be seen that after 24 threads, the performance started decreasing. Increasing the number of threads after 24 didn't have any effect on the performance. Out of all the number of trials for 24 threads, 100,000 trials performed the best and had a peak performance of 107.04 megatrials/sec.Till 20 threads, 100000, 500000 and 1000000 trials had almost same performance after which they increased and reached the peak but at varied levels. This graph shows that increasing the number of threads indefinitely will not have an indefinite increase in performance but instead it will start decreasing after reaching a peak.

For the graph **Performance vs no of Monte Carlo trials**, as a general trend it can be seen that beyond 100,000 trials all the performance lines for the various number of threads become stagnant. For 1, 10, 100 and 1000 trials it can be seen that the performance is concentrated around 0-20 megatrials per second. This might be due to the fact that for a small number of trials they have similar performance even if the number of threads increases, however for more trials, the performances among all the different threads are distinct. For threads 1,2,4 after 100,000 trials the performance has an upwards trend. For thread 8 the performance is stagnant from 100,000 trials to 1,000,000 trials. As can be seen from the graph thread 24 peaked at 100,000 trials with performance 107.04 megatrials/sec. This tells us that for small trials and number of threads there is overhead required so it's not worth the parallelism to work efficiently.

Screenshot of outcome:



```
[flip1 ~ 1012$ cat OUT.csv
[ 1,         1,   0.00,   0.73
  1,        10,  10.00,   3.94
  1,       100,  30.00,   6.52
  1,      1000,  26.10,   6.93
  1,     10000,  27.18,   6.92
  1,    100000,  26.84,   6.89
  1,    500000,  26.83,   6.87
  1,   1000000,  26.84,  13.02
  2,         1,   0.00,   0.35
  2,        10,  30.00,   3.35
  2,       100,  21.00,   9.82
  2,      1000,  27.20,  13.19
  2,     10000,  26.09,  13.87
  2,    100000,  26.81,  13.61
  2,    500000,  26.83,  13.76
  2,   1000000,  26.89,  26.16
  4,         1, 100.00,   0.63
  4,        10,  40.00,   5.68
  4,       100,  23.00,  11.62
  4,      1000,  26.30,  26.12
  4,     10000,  27.42,  27.38
  4,    100000,  26.88,  27.07
  4,    500000,  26.87,  26.92
  4,   1000000,  26.89,  50.26
  8,         1, 100.00,   0.33
  8,        10,  10.00,   3.40
  8,       100,  28.00,  16.39
  8,      1000,  26.00,  33.14
  8,     10000,  26.23,  53.89
  8,    100000,  26.69,  54.53
  8,    500000,  26.94,  53.55
  8,   1000000,  26.90,  53.61
 12,         1, 100.00,   0.22
 12,        10,  60.00,   2.35
 12,       100,  26.00,  12.74
 12,      1000,  25.50,  42.29
 12,     10000,  26.54,  57.19
 12,    100000,  26.80,  80.52
 12,    500000,  26.85,  73.75
 12,   1000000,  26.92,  77.83
 16,         1,   0.00,   0.16
 16,        10,  30.00,   1.52
 16,       100,  29.00,  10.94
 16,      1000,  27.10,  51.29
 16,     10000,  27.30,  69.78
 16,    100000,  26.82,  72.18
 16,    500000,  26.81,  72.55
 16,   1000000,  26.82,  75.04
 20,         1,   0.00,   0.17
 20,        10,  20.00,   1.58
 20,       100,  20.00,  10.13
 20,      1000,  26.30,  53.34
 20,     10000,  26.93,  84.19
 20,    100000,  26.79,  90.11
 20,    500000,  26.85,  89.30
 20,   1000000,  26.92,  90.08
 24,         1,   0.00,   0.13
 24,        10,  30.00,   1.28
 24,       100,  21.00,   9.00
 24,      1000,  23.30,  56.41
 24,     10000,  26.51,  98.57
 24,    100000,  26.83, 107.04
 24,    500000,  26.91,  91.33
 24,   1000000,  26.86,  84.04
 32,         1,   0.00,   0.02
 32,        10,  50.00,   0.17
 32,       100,  33.00,   1.63
 32,      1000,  25.50,  17.93
 32,     10000,  26.73,  54.30
 32,    100000,  27.10,  74.67
 32,    500000,  26.90,  72.12
 32,   1000000,  26.82,  84.33
 flip1 ~ 1013$ client_loop: send disconnect: Broken pipe
```