Project #1 – OpenMP: Monte Carlo Simulation

Young-Joon Park

parky8@oregonstate.edu

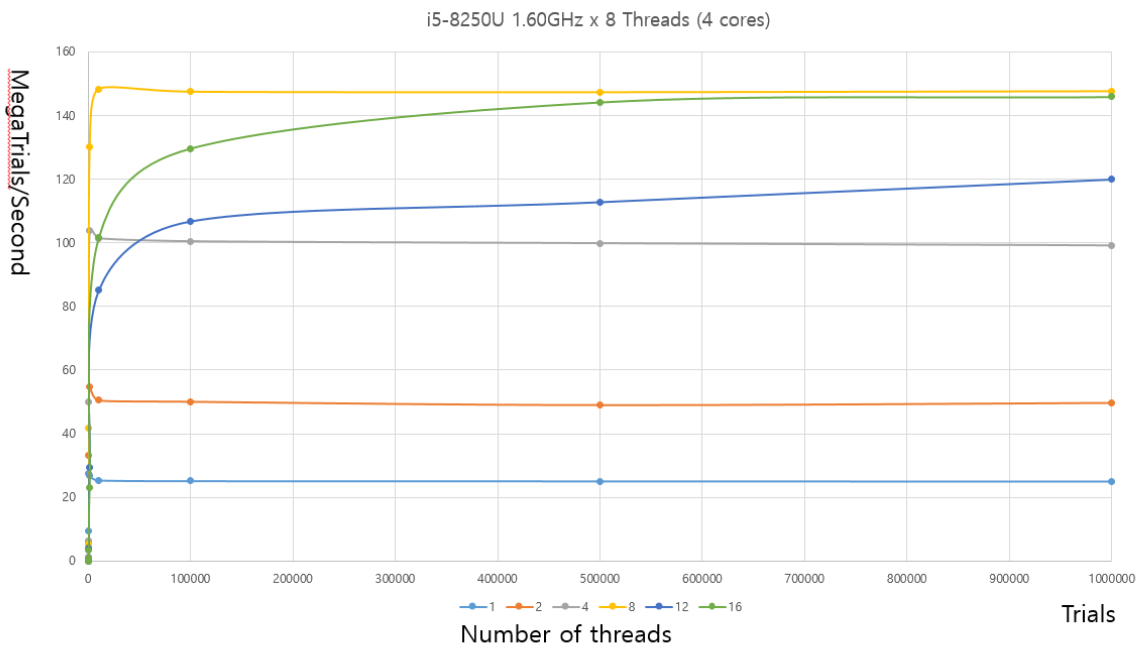
Data table of the performance numbers as a function of threads and NUMTRIALS:

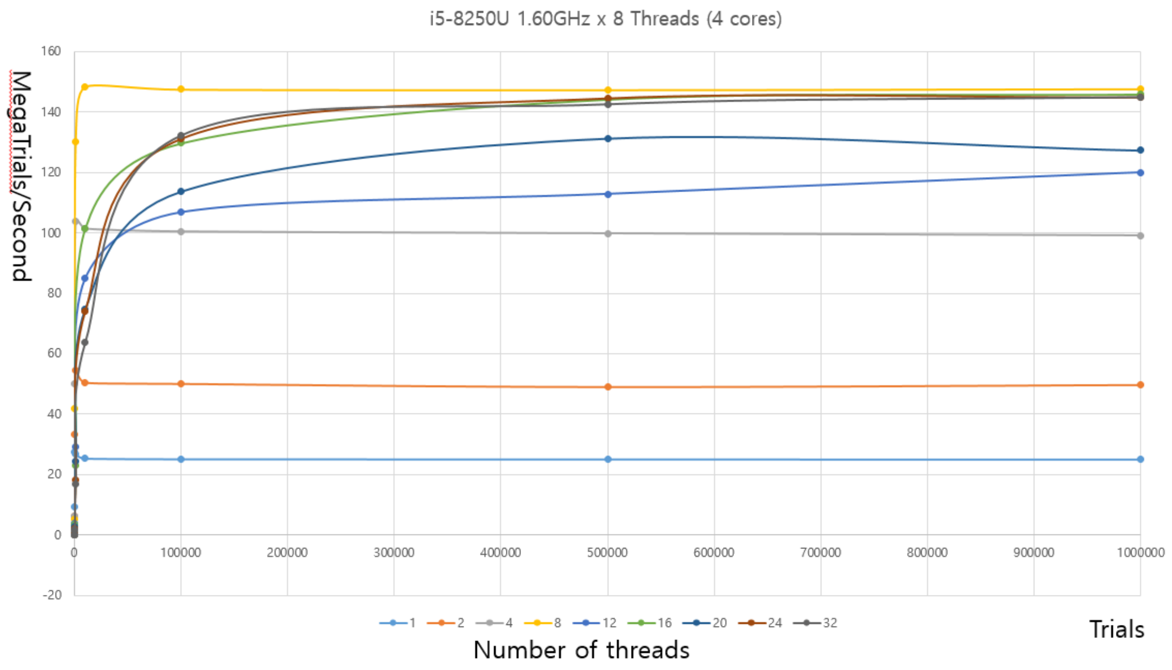
Threads\Num Trials	1	10	100	1000	10000	100000	500000	1000000
1	1.22	9.38	27.37	26.76	25.33	25.11	25.05	24.99
2	0.79	6.27	33.23	54.61	50.54	50.13	49.08	49.76
4	0.65	6.25	50	103.91	101.35	100.42	99.8	99.12
8	0.6	5.14	41.86	130.12	148.41	147.52	147.37	147.69
12	0.04	0.59	3.95	29.27	85.07	106.82	112.87	120
16	0.04	0.38	3.49	22.95	101.57	129.73	144.25	145.95
20	0.03	0.26	2.85	24.26	74.83	113.78	131.33	127.37
24	0.03	0.27	2.14	18.07	74.21	131.19	144.55	144.87
32	0.02	0.21	1.67	16.72	63.84	132.37	142.67	145.04

The 2 performance graphs.

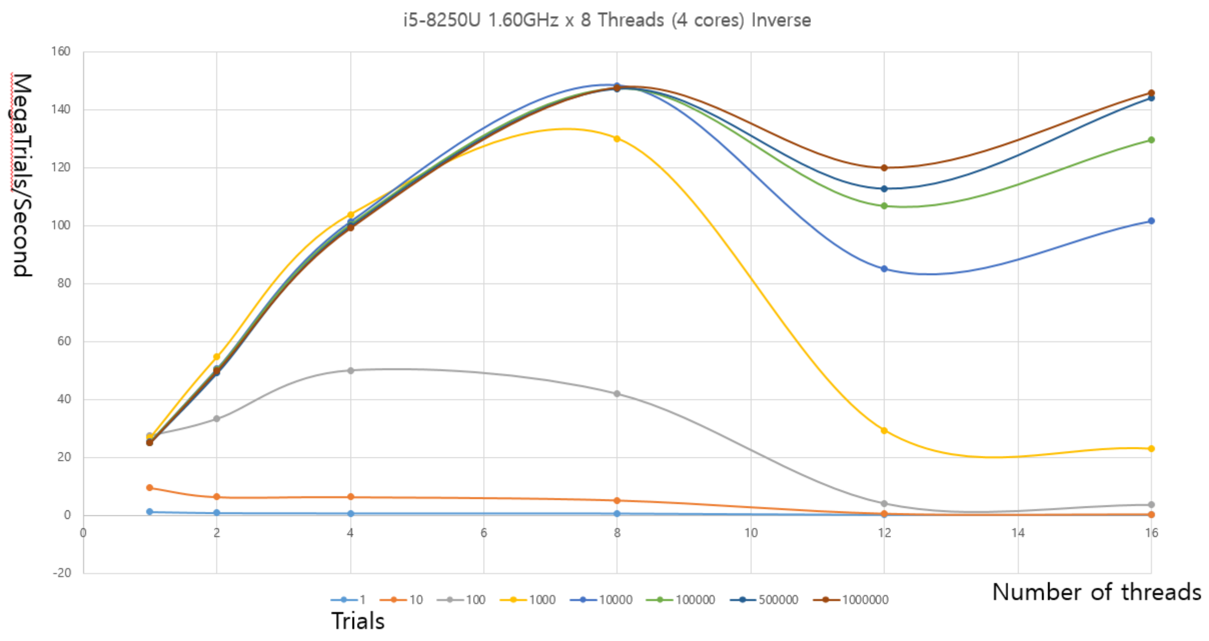
1. Performance versus the number of Monte Carlo trials, with the colored lines being the number of OpenMP threads.

1st graph, but 2 versions, 2nd one with more threads

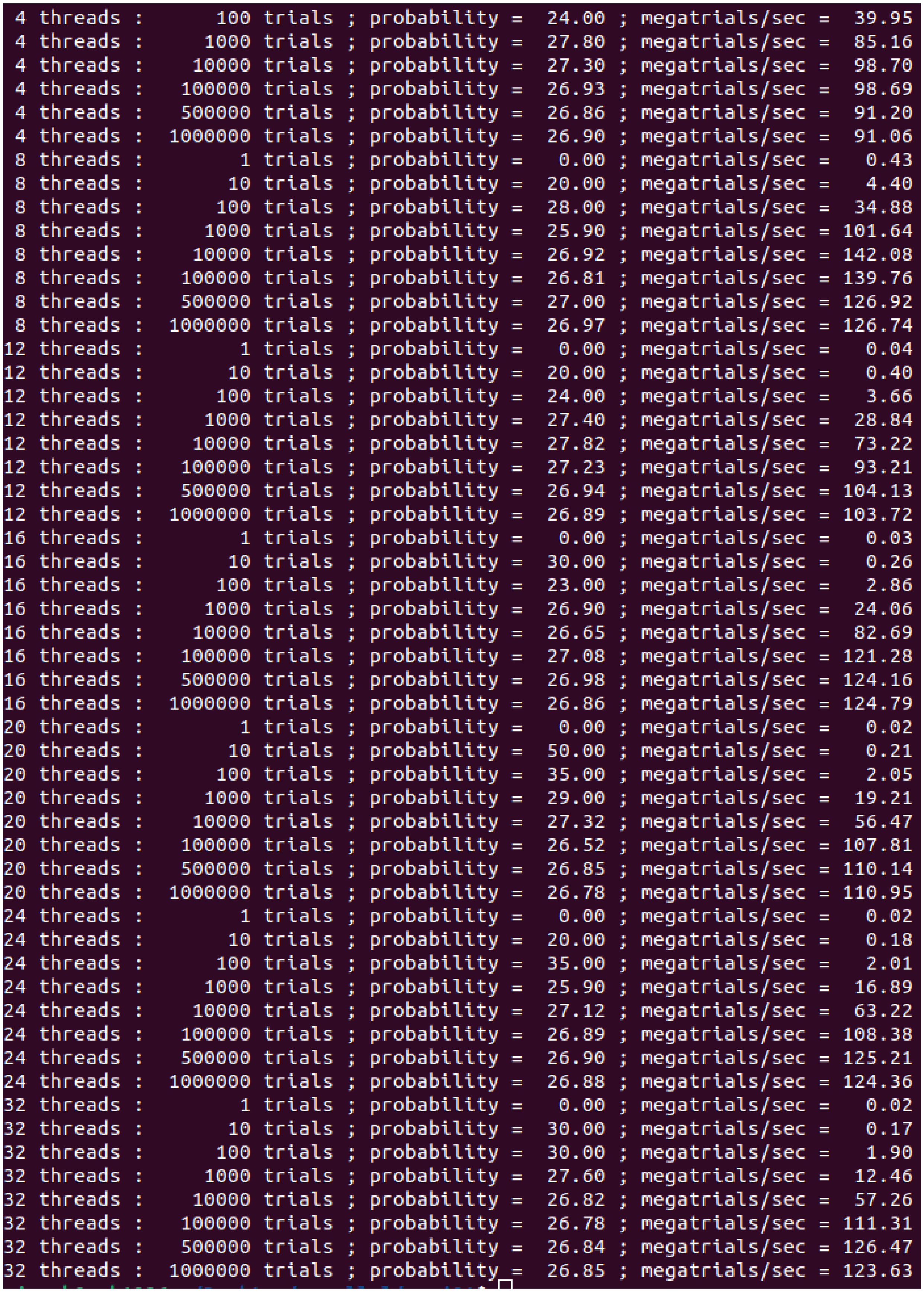




2. Performance versus the number OpenMP threads, with the colored lines being the number of Monte Carlo trials.



Estimate of the Probability.



```
 4 threads :      100 trials ; probability =  24.00 ; megatrials/sec =  39.95
 4 threads :     1000 trials ; probability =  27.80 ; megatrials/sec =  85.16
 4 threads :    10000 trials ; probability =  27.30 ; megatrials/sec =  98.70
 4 threads :   100000 trials ; probability =  26.93 ; megatrials/sec =  98.69
 4 threads :   500000 trials ; probability =  26.86 ; megatrials/sec =  91.20
 4 threads :  1000000 trials ; probability =  26.90 ; megatrials/sec =  91.06
 8 threads :        1 trials ; probability =   0.00 ; megatrials/sec =   0.43
 8 threads :       10 trials ; probability =  20.00 ; megatrials/sec =   4.40
 8 threads :      100 trials ; probability =  28.00 ; megatrials/sec =  34.88
 8 threads :     1000 trials ; probability =  25.90 ; megatrials/sec = 101.64
 8 threads :    10000 trials ; probability =  26.92 ; megatrials/sec = 142.08
 8 threads :   100000 trials ; probability =  26.81 ; megatrials/sec = 139.76
 8 threads :   500000 trials ; probability =  27.00 ; megatrials/sec = 126.92
 8 threads :  1000000 trials ; probability =  26.97 ; megatrials/sec = 126.74
12 threads :        1 trials ; probability =   0.00 ; megatrials/sec =   0.04
12 threads :       10 trials ; probability =  20.00 ; megatrials/sec =   0.40
12 threads :      100 trials ; probability =  24.00 ; megatrials/sec =   3.66
12 threads :     1000 trials ; probability =  27.40 ; megatrials/sec =  28.84
12 threads :    10000 trials ; probability =  27.82 ; megatrials/sec =  73.22
12 threads :   100000 trials ; probability =  27.23 ; megatrials/sec =  93.21
12 threads :   500000 trials ; probability =  26.94 ; megatrials/sec = 104.13
12 threads :  1000000 trials ; probability =  26.89 ; megatrials/sec = 103.72
16 threads :        1 trials ; probability =   0.00 ; megatrials/sec =   0.03
16 threads :       10 trials ; probability =  30.00 ; megatrials/sec =   0.26
16 threads :      100 trials ; probability =  23.00 ; megatrials/sec =   2.86
16 threads :     1000 trials ; probability =  26.90 ; megatrials/sec =  24.06
16 threads :    10000 trials ; probability =  26.65 ; megatrials/sec =  82.69
16 threads :   100000 trials ; probability =  27.08 ; megatrials/sec = 121.28
16 threads :   500000 trials ; probability =  26.98 ; megatrials/sec = 124.16
16 threads :  1000000 trials ; probability =  26.86 ; megatrials/sec = 124.79
20 threads :        1 trials ; probability =   0.00 ; megatrials/sec =   0.02
20 threads :       10 trials ; probability =  50.00 ; megatrials/sec =   0.21
20 threads :      100 trials ; probability =  35.00 ; megatrials/sec =   2.05
20 threads :     1000 trials ; probability =  29.00 ; megatrials/sec =  19.21
20 threads :    10000 trials ; probability =  27.32 ; megatrials/sec =  56.47
20 threads :   100000 trials ; probability =  26.52 ; megatrials/sec = 107.81
20 threads :   500000 trials ; probability =  26.85 ; megatrials/sec = 110.14
20 threads :  1000000 trials ; probability =  26.78 ; megatrials/sec = 110.95
24 threads :        1 trials ; probability =   0.00 ; megatrials/sec =   0.02
24 threads :       10 trials ; probability =  20.00 ; megatrials/sec =   0.18
24 threads :      100 trials ; probability =  35.00 ; megatrials/sec =   2.01
24 threads :     1000 trials ; probability =  25.90 ; megatrials/sec =  16.89
24 threads :    10000 trials ; probability =  27.12 ; megatrials/sec =  63.22
24 threads :   100000 trials ; probability =  26.89 ; megatrials/sec = 108.38
24 threads :   500000 trials ; probability =  26.90 ; megatrials/sec = 125.21
24 threads :  1000000 trials ; probability =  26.88 ; megatrials/sec = 124.36
32 threads :        1 trials ; probability =   0.00 ; megatrials/sec =   0.02
32 threads :       10 trials ; probability =  30.00 ; megatrials/sec =   0.17
32 threads :      100 trials ; probability =  30.00 ; megatrials/sec =   1.90
32 threads :     1000 trials ; probability =  27.60 ; megatrials/sec =  12.46
32 threads :    10000 trials ; probability =  26.82 ; megatrials/sec =  57.26
32 threads :   100000 trials ; probability =  26.78 ; megatrials/sec = 111.31
32 threads :   500000 trials ; probability =  26.84 ; megatrials/sec = 126.47
32 threads :  1000000 trials ; probability =  26.85 ; megatrials/sec = 123.63
```

Picking out 32 threads, 1,000,000 trials, it can be estimated that the probability is around 27 percent.

Estimate of the Parallel Fraction (*show your work!*).

From the noteset,

Fp = (n / (n – 1)) * ((T1 - Tn) / T1)

Take the 1,000,000 trial runs:

1 thread: 24.99 MegaTrials/Sec = 24.99 * 10^6 Trials/Sec

T1 = 1,000,000 Trials / 24.99 * 10^6 Trials / Sec = 0.0400 Sec

8 threads: 147.69 MegaTrials/Sec = 147.69 * 10^6 Trials/Sec

Tn = 1,000,000 Trials / 147.69 * 10^6 Trials / Sec = 0.0068 Sec

Fp = (n / (n – 1)) * ((T1 - Tn) / T1)

= (8 / (8 – 1)) * ((0.0400s – 0.0068s) / 0.0400s)

= (8 / 7) * (0.0332s / 0.0400s)

= 0.9486

Fp = 94.86%

Commentary: why do the graphs look the way they do? What are they telling you?

1. When the dataset is small, multi-thread parallelism is not in its peak performance. This is because there is a certain overhead in multithreading. The overhead holds a larger share of the possible performance when there is not much to process. However, as the amount of data to process increases, the performance improves and asymptotes out, because the overhead eventually becomes negligible. This overhead issue is much clearer in the third graph – the

smaller datasets can be found at lower levels of performance, while the bigger datasets tend to enjoy higher levels of performance.

2. It is also interesting that the third graph dips after 8 threads. I think there are several factors that affect the graph. The first is that the old Linux laptop I used for the trials had a 4-core 8 thread CPU. Hence, 8 threads were optimized for performance. When it comes to 12-thread trials, each CPU had to handle 3 threads which do not divide cleanly in binary, hence (I think, along with the fact that the CPU is optimized for 8 threads) there is a dip in performance. When there are 16 threads, performance increases relative to 12 threads but not as much as 8 threads. I think this is because 16 threads cleanly divides in binary (hence, better than 12), but each of the hyperthreaded 8 threads in the CPU must each handle 2 threads to make up 16 (hence, there is a need to store and load registers from memory, which is sub-optimal compared to 8 threads).