Project #1 - OpenMP: Monte Carlo Simulation

Junhyeok Jeong
jeongju@oregonstate.edu
CS 475 - 001- S2020

Tell me what machine ran this on

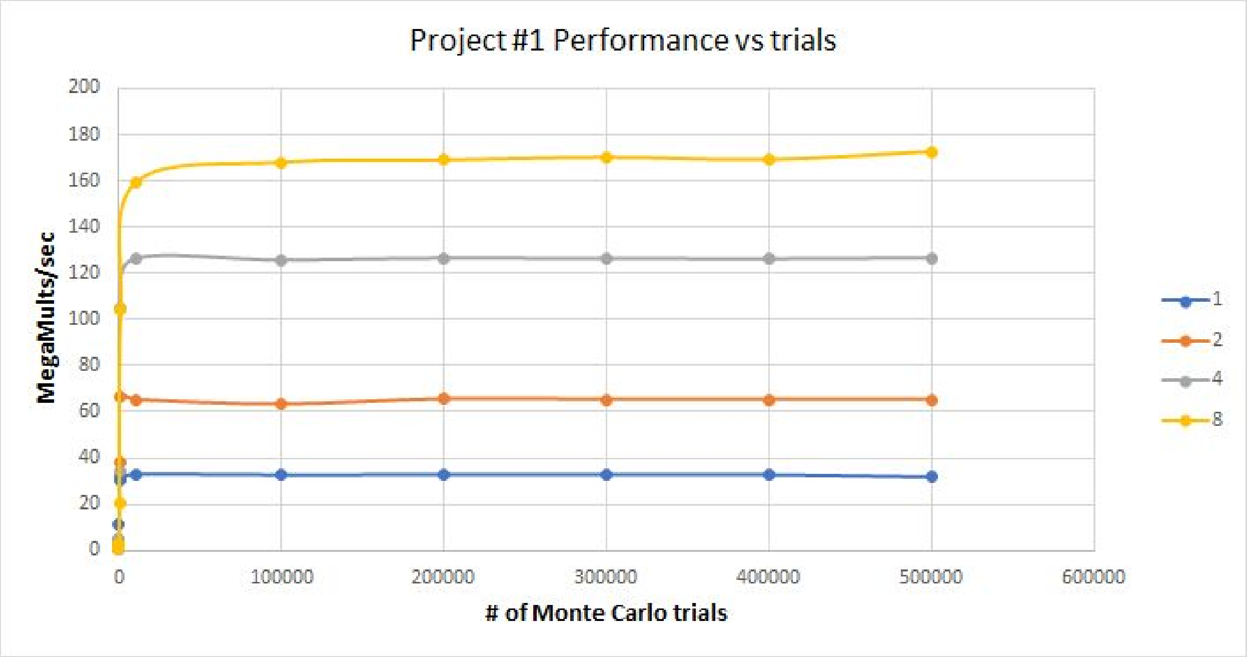
- **I ran my program on OSU flip3 server.**

Result Data chart

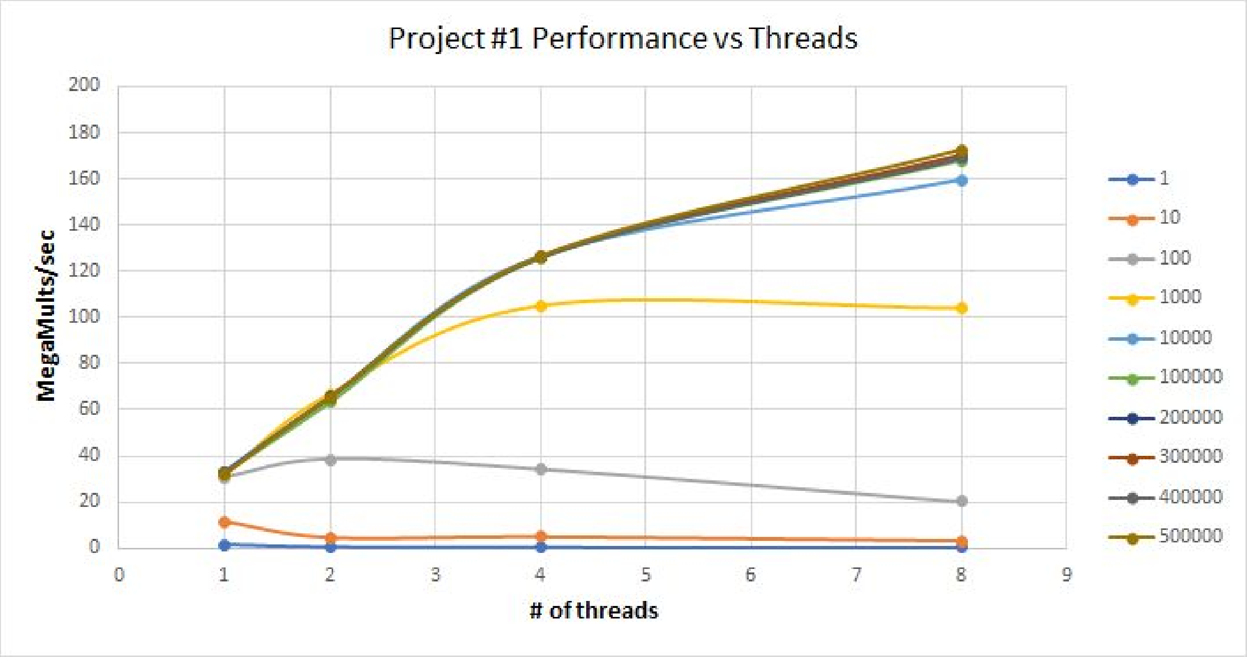
Threads	**Trials**	**Probability**	**Performance**	**Execution time**
1	1	0	1.64181	6.13E-07
1	10	0.2	11.4016	8.77E-07
1	100	0.17	30.7795	3.25E-06
1	1000	0.151	31.749	3.15E-05
1	10000	0.1296	32.8282	0.000305
1	100000	0.13044	32.5782	0.00307
1	200000	0.1301	32.663	0.006123
1	300000	0.13036	32.637	0.009197
1	400000	0.13021	32.6135	0.01227
1	500000	0.131364	31.6282	0.015859
2	1	0	0.758694	1.38E-06
2	10	0.1	4.55748	2.46E-06
2	100	0.12	38.3856	2.62E-06
2	1000	0.113	66.7646	1.51E-05
2	10000	0.1312	65.1428	0.000164
2	100000	0.13267	63.055	0.001596
2	200000	0.130155	65.3996	0.003076

2	300000	0.130133	65.2369	0.004599
2	400000	0.130242	65.2913	0.006144
2	500000	0.130974	65.2595	0.007664
4	1	0	0.540519	2.14E-06
4	10	0	4.81715	2.27E-06
4	100	0.16	34.0735	3.17E-06
4	1000	0.14	105.163	9.79E-06
4	10000	0.1285	126.17	8.39E-05
4	100000	0.13219	125.614	0.000798
4	200000	0.131505	126.375	0.001588
4	300000	0.131337	126.194	0.002379
4	400000	0.131268	126.088	0.003192
4	500000	0.131088	126.48	0.003954
8	1	0	0.33888	2.96E-06
8	10	0.2	3.10263	3.99E-06
8	100	0.18	20.1	4.98E-06
8	1000	0.127	104.123	9.79E-06
8	10000	0.1355	159.333	6.34E-05
8	100000	0.13087	168.033	0.0006
8	200000	0.130415	169.037	0.001183
8	300000	0.13026	170.244	0.001768
8	400000	0.13064	169.232	0.002368
8	500000	0.13092	172.64	0.002896

1. **Do a table and two graphs showing performance versus trials and threads.**



- The “Performance vs trials” graph shows that adding more threads increased the performance indicator (MegaMults/sec). Although the increase rate between single and quad threads was consistent (about 30 -> 60 -> 120), the rate dropped significantly after quad threads.



- When I compared to the above graph, the “Performance vs Threads” graph made me see the threads scheduling overhead reduce the performance increment rate.

2. **Choosing one of the runs (the one with the maximum number of trials would be good), tell me what you think the actual probability is.**

- I chose 500000 trials (maximum number of trials) for this question. I believe that the actual probability of 500000 trials for all threads is **about 13%** because I calculated the probability that the beam hits the plate by tweaking code based on the conditions from the project description. First, it said “If d is less than 0., then the circle was completely missed. (Case A) Continue on to the next trial in the for-loop.” I put below code before Case B:

```
            if (d < 0.){
                continue;
            }
```

 Next, it said “If tmin is less than 0., then the circle completely engulfs the laser pointer. (Case B) Continue on to the next trial in the for-loop. I put below code before Case C:

```
            if (tmin < 0.){
                continue;
            }
```

Next, it said “If tt is less than 0., then the reflected beam went up instead of down. Continue on to the next trial in the for-loop. Otherwise, this beam hit the infinite plate. (Case D) Increment the number of hits and continue on to the next trial in the for-loop.” I put below code to accumulate the number of plate hits:

```
            if (t < 0.){
                continue;
            }
            numHits = numHits + 1;
```

Lastly, the code said

```
currentProb = (float)numHits/(float)NUMTRIALS;
```

This line will calculate the probability with the division between accumulated number of plate hits and the number of trials (In my case, 1 10 100 1000 10000 100000 200000 300000 400000 500000).

Most of all, the reduction flag of loop in the code:

```
#pragma omp parallel for default(none) shared(xcs,ycs,rs,tn)
reduction(+:numHits)
```

Will check the plate hit validation.

3. **Compute Fp, the Parallel Fraction, for this computation.**

- To compute Parallel Fraction, I use the speed-up from 1 to 8 threads with maximum trials (500000) because I tested threads from 1 to 8 sequentially.

Threads	Performance
1	31.6282
2	65.2595
4	126.48
8	172.64

- SpeedUp = (Performance with 8 threads) / (Performance with 1 thread) = 172.64 / 31.6282 = 5.45842001758 = **about 5.46**
- According to Amdahl's Law,

$$SpeedUp = \frac{T1}{Tn} = \frac{1}{\frac{Fparallel}{500000} + (1 - Fparallel)}$$

$$Fparallel = \frac{4.46}{5.46} * \frac{500000}{499999} = about\ 0.82$$

$$MAXSpeedUp = \frac{1}{1 - Fparallel} = \frac{1}{1 - 0.82} = about\ 5.56$$