Gregory Douglas Stula
CS 475
2020-04-17

Project 01 - Monte Carlo Experiment

Hardware

This experiment was ran on Gentoo Linux with kernel 5.4.28 on a desktop pc with the following hardware configuration:

CPU: AMD Ryzen 7 3800X 8- (16) @ 3.900GHz
GPU: NVIDIA GeForce GTX 1080
Memory: 32078MiB DDR4

Performance Results

The test was to run a Monte Carlo experiment regarding a laser hitting a randomly changing sized plate in a C++ program compiled with gcc. The trial was run 10, 100, 1000, and 10000 times. This was done on a single thread, 2, threads, and incremented by 2 up to the 16 threads available to the AMD Ryzen 7 CPU. Performance was measured in Mega Trials per Second (MT/s). The experiment was run partially within the C++ program and with a python3 script wich redirects the output of the C++ program to a CSV file which was uploaded to google drive for analysis in Google sheets.

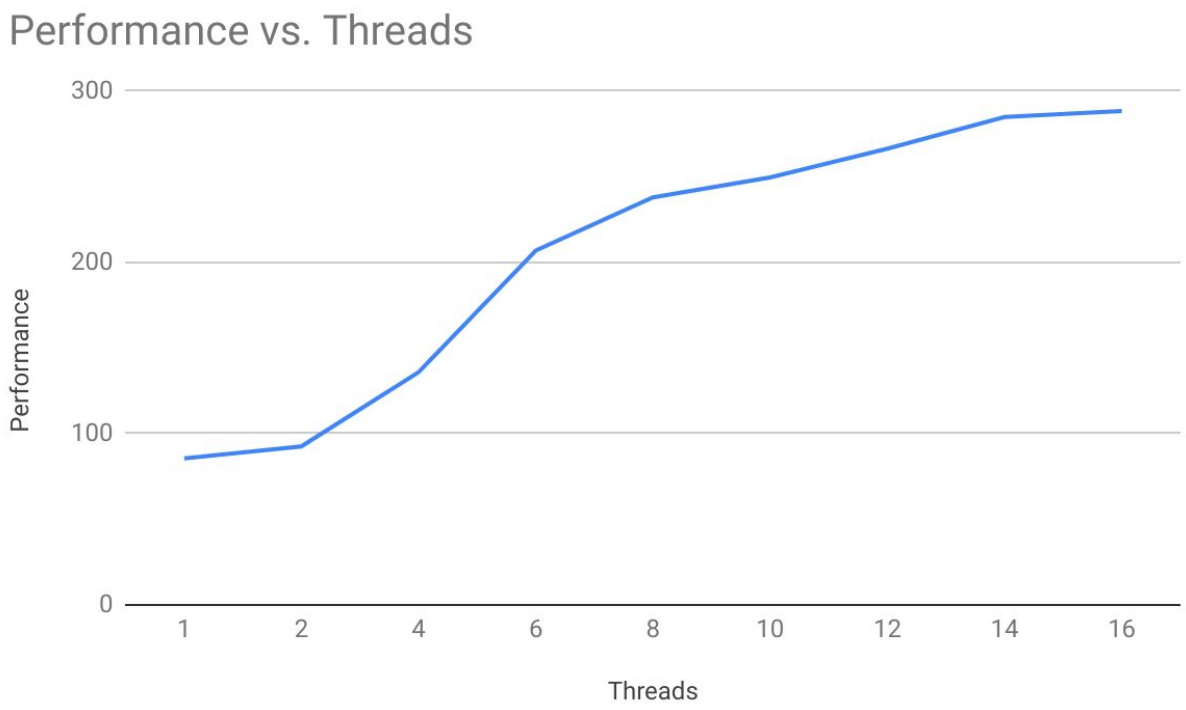
The python3 program can be called with ./run.py
The script compiles and executes the C++ program for each number of trial and thread combinations. The C++ program prints the threads, number of trials, probability of the laser hitting the plate, and the performance in a comma separated format to be redirected to a CSV file when executed. The script handles the redirects and multiple runs.

Results

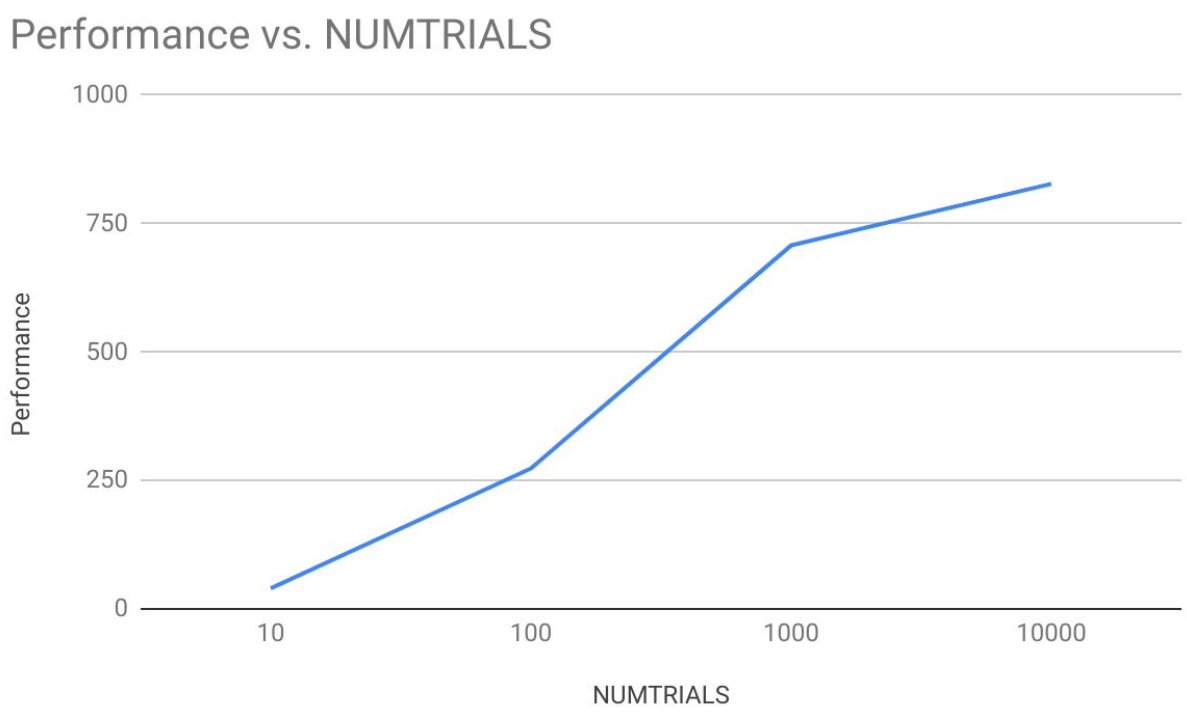
Threads	NUMTRIALS	Probability	Performance
1	10	0.1	8.116873
1	100	0.15	16.358587
1	1000	0.122	31.174988
1	10000	0.133	29.940121
2	10	0.2	5.783689
2	100	0.05	27.048948
2	1000	0.119	30.346251
2	10000	0.133	29.296732
4	10	0.3	5.467487
4	100	0.17	29.239717
4	1000	0.133	47.812569
4	10000	0.1261	53.320255
6	10	0.1	4.056796
6	100	0.12	39.447704
6	1000	0.127	80.09613
6	10000	0.1301	83.277817
8	10	0.1	4.226537
8	100	0.08	35.549213
8	1000	0.115	95.996902
8	10000	0.1296	102.213951
10	10	0	3.456619
10	100	0.12	33.311104
10	1000	0.12	94.286224
10	10000	0.1276	118.519928
12	10	0.1	3.45781
12	100	0.12	33.87532
12	1000	0.116	106.906219
12	10000	0.1327	122.146362
14	10	0.1	2.483856
14	100	0.19	30.202314
14	1000	0.116	112.523911

14	10000	0.136	139.721405
16	10	0.3	2.915456
16	100	0.17	28.256538
16	1000	0.128	108.401115
16	10000	0.1308	148.749756
Prob 10K	**0.130775**		
Min 10k	0.1261		
Max 10k	0.136		
Prob 100	0.13625		
Min 100	0.05		
Max 100	0.17		

Performance (Mega Trials per Second) vs Threads (at 1000k Trials)



Performance (Mega Trials per Second) vs



Trials (Aggregate)

Parallel Fraction

The single thread speed for 1000 trials was **29.940121 MT/s**
The multi thread speed (at 16 threads) for 1000 trials was **148.749756 MT/S**
S = Multi Thread Speed / Single Thread Speed = 148.749756 / 29.940121
Therefore we can calculate the Speedup at 16 threads vs single threaded
to be **S = 4.968241645**

The parallel fraction can be calculated with (N/ N-1)*(1 - (1/S))
Which is (16/15)*(1 - (1/4.968241645))
Which is (16/15) * (0.7987215455) = 0.8519696485
And that gives us **FP = 0.85**

Probability and Commentary

We will use the 10000 or 10K trail runs to determine our probability. The MIN probability in the 10K trail runs that was found was 12.6% and the MAX probability found was 13.6%. Taking the

average of the runs with 10000 trials we get a **probability of 13.07%** that the laser will hit the randomly sized plate.

If we look at runs of the program that only used a small amount of trials, like 100, we see that the probabilities are much more varied across the different threaded runs. These runs range from 6% to 17%. However the average probability of these runs is still closer to our previously stated value at 13.6% which gives us confidence in our 13.12% average based on the 1000 trial runs. The variance in these runs can be accounted for simply in the fact that 100 trials is not enough for a good estimate. By averaging the results of 100 trials across the 10 different thread runs we come closer to an accurate estimate found in the larger trial runs because we have 1000 trials in total.

We can see that as threads increased the Mega Trials per second increased as the number of threads increased. This shows that the Monte Carlo simulation for this scenario scaled nicely. We had close to a 5x speed up when increasing to 16 threads. The performance also increased across the aggregate number of trials which shows the scalability of the monte carlo simulation.