CS 475

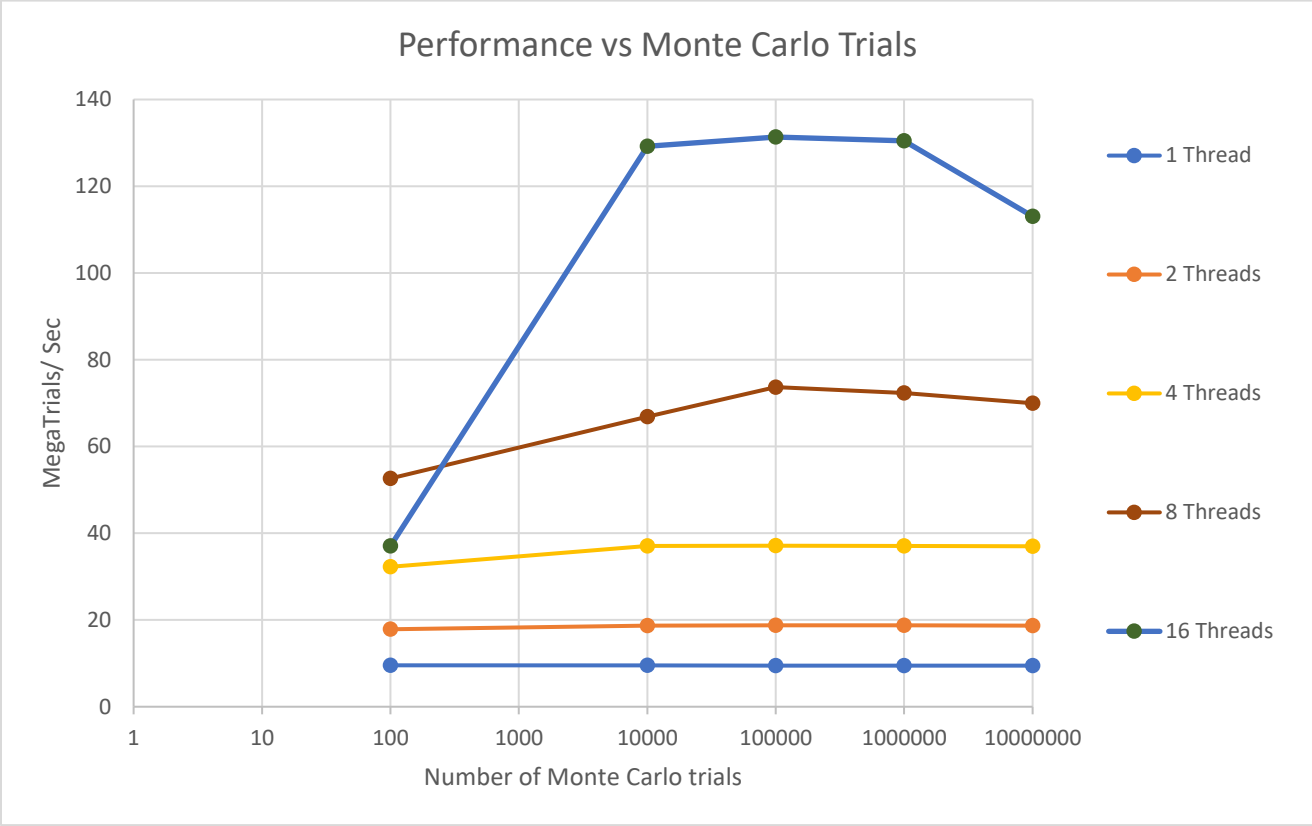
4/15/2021

Kyle Huang

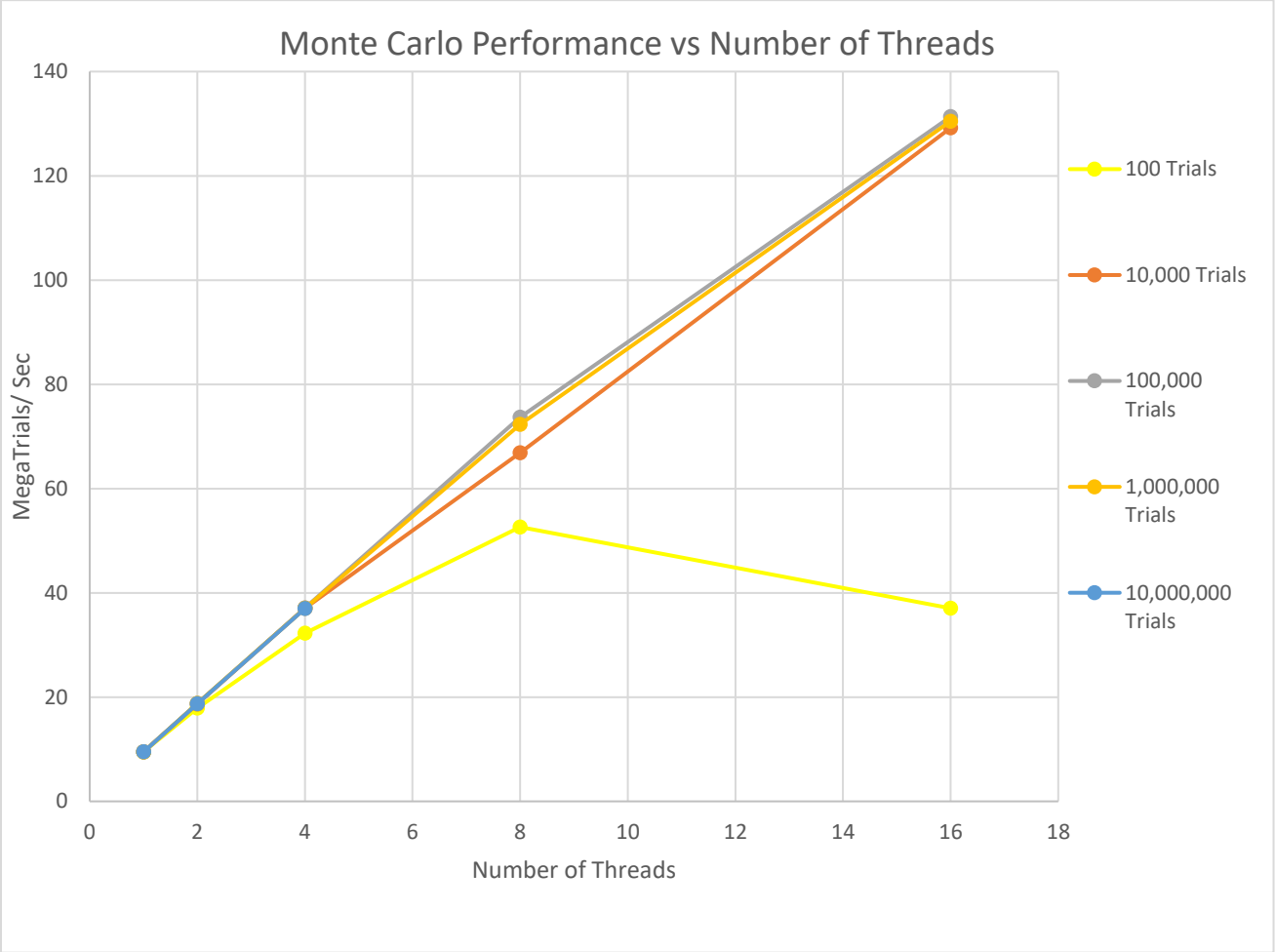
Huangky@oregonstate.edu

Project 1

1.



2.



Tables:

Probability:

	Trials				
Number of threads	100	10000	100000	1000000	10000000
1	4%	6.11%	6.66%	6.93%	6.57%
2	4%	6.08%	6.62%	6.57%	6.51%
4	4%	5.93%	6.44%	6.37%	6.33%
8	3%	6.53%	6.37%	6.3%	6.34%
16	4%	1.85%	5.83%	5.97%	5.77%

MegaTrials/ Sec:

	Trials				
Number of threads	100	10000	100000	1000000	10000000
1	9.52	9.53	9.49	9.47	9.48
2	17.86	18.69	18.77	18.74	18.71
4	32.26	37.05	37.12	37.06	36.98
8	52.63	66.89	73.69	72.33	69.97
16	37.04	129.2	131.34	130.48	113.06

3. Compute actual probability.
   I am calculating for 16 threads and 1,000,000 trials.

   Speed up, S, 1 thread to 16 threads.

   S = $\frac{130.48\ Mt/s}{9.47\ Mt/S}$ = 13.7782

   $F_p = \frac{16}{15}(1 - \frac{1}{S})$

   F_p = 0.989

   Explanation:

   The first thing I did to get the program to work was to look over the program, specifically at the '???' locations. I noticed that 'omp parallel for' needed to have something at the end, I decided to use 'shared()' containing all variables for calculations and 'numHits'. I used 'shared()' since all of these variables will need to be shared between the threads and when I tried 'default(none)' I got an error. Next, I looked over the project specs and saw I needed to solve some equations with respect to time. I then solved equations that needed to be set in terms of time. I also plugged equations into the program as I was going. I put the equations in their respective variable and used the equations consecutively. All of these equations were given in the respective order needed to implement them. These all work in this order since you first check if the ball will travel far enough; then, check if it will clear the cliff. The last thing I added was 'numHits++' in the case where the ball hits the castle. This works since it is only being

called by one thread and can be shared. I also added an extra print to ‘stdout’ so I can capture the output with a python script.

One thing I noticed while analyzing the results, is that an extremely large number of trials (10MM) would cause a slower time. I think this is due to a caching issue where the arrays are too big to fit into cache, therefore the computer will have to look in memory for the data. Another factor that affected it was that when I was doing 10MM trials, my CPU was maxed out and dropped clock speed down to 4.8GHz from 5.1GHz on all other trials. Another thing I noticed is that the probability is low for arrays with less elements. I think this is due to the law of large numbers and will come closer to an accurate prediction as there are more test cases. The last thing I noticed is that there is significant over-head in setting up more threads to run the trials, on smaller datasets.