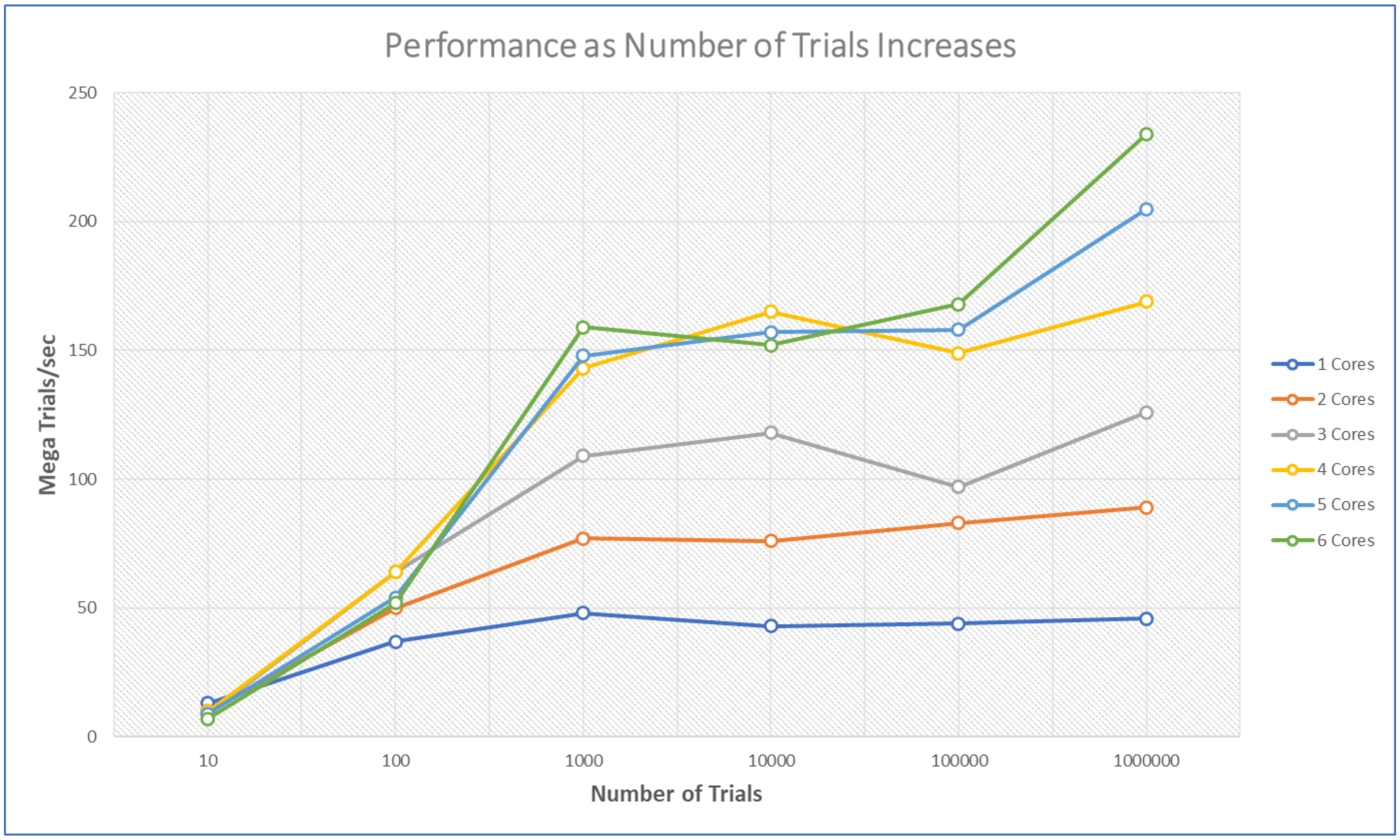
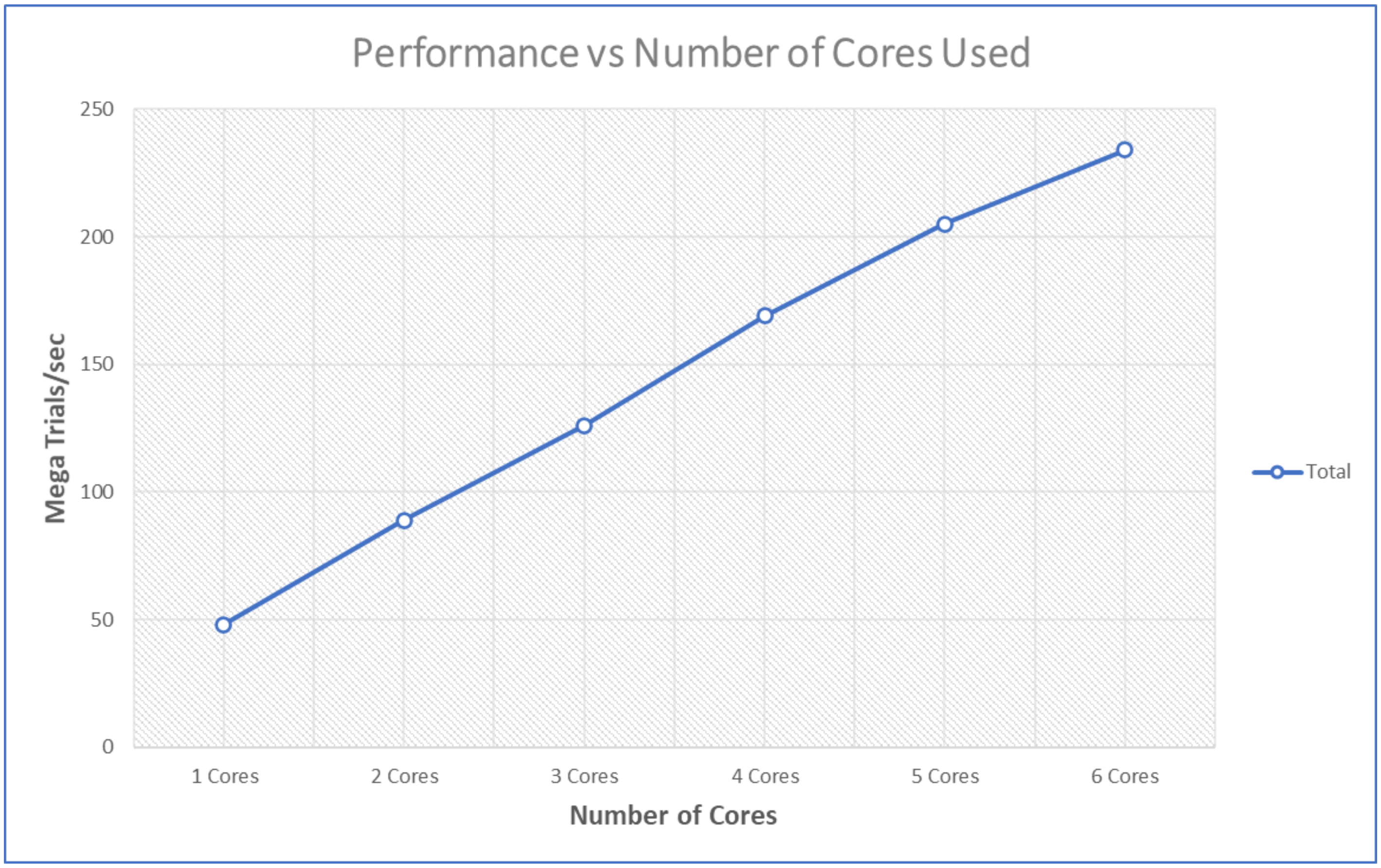
Clinton Hawkes
hawkesc@oregonstate.edu
CS475-400
Project #1

1. Probability of hitting plate = .13
   I got this figure by averaging all the outputs from running my program script. I had a range of values from .09 to .2, but the figure converged on .13 as the NUMTRIALS grew larger.
2. As the number of trials increases, we can see that the performance also increases. This is valid for any number of cores greater than 1. If I continued to increase the number of trials, the max performance observed would eventually level off and remain constant. This is due to the following:
   a. Gustafson's observation shows that Fp will approach 1 as the size of the dataset becomes extremely large. (as N -> infinity)
   b. If Fp approaches 1, then the Speedup when using n cores = 1 / ((F/n)+(1-F)) => 1 / (F/n) => n. The max performance in Mega Trials/sec for a computer with n cores should stabilize at n*(max performance of computer with 1 core)

   This, of course, is only valid for extremely large datasets. Most people aren't going to use dataset this big, so this really only give us a ceiling for the performance that can be attained. All we can really say is that performance increase as the size of the dataset increase. If somebody wants to know the exact performance that they can gain, they must use the data from each individual problem to calculate the gain.

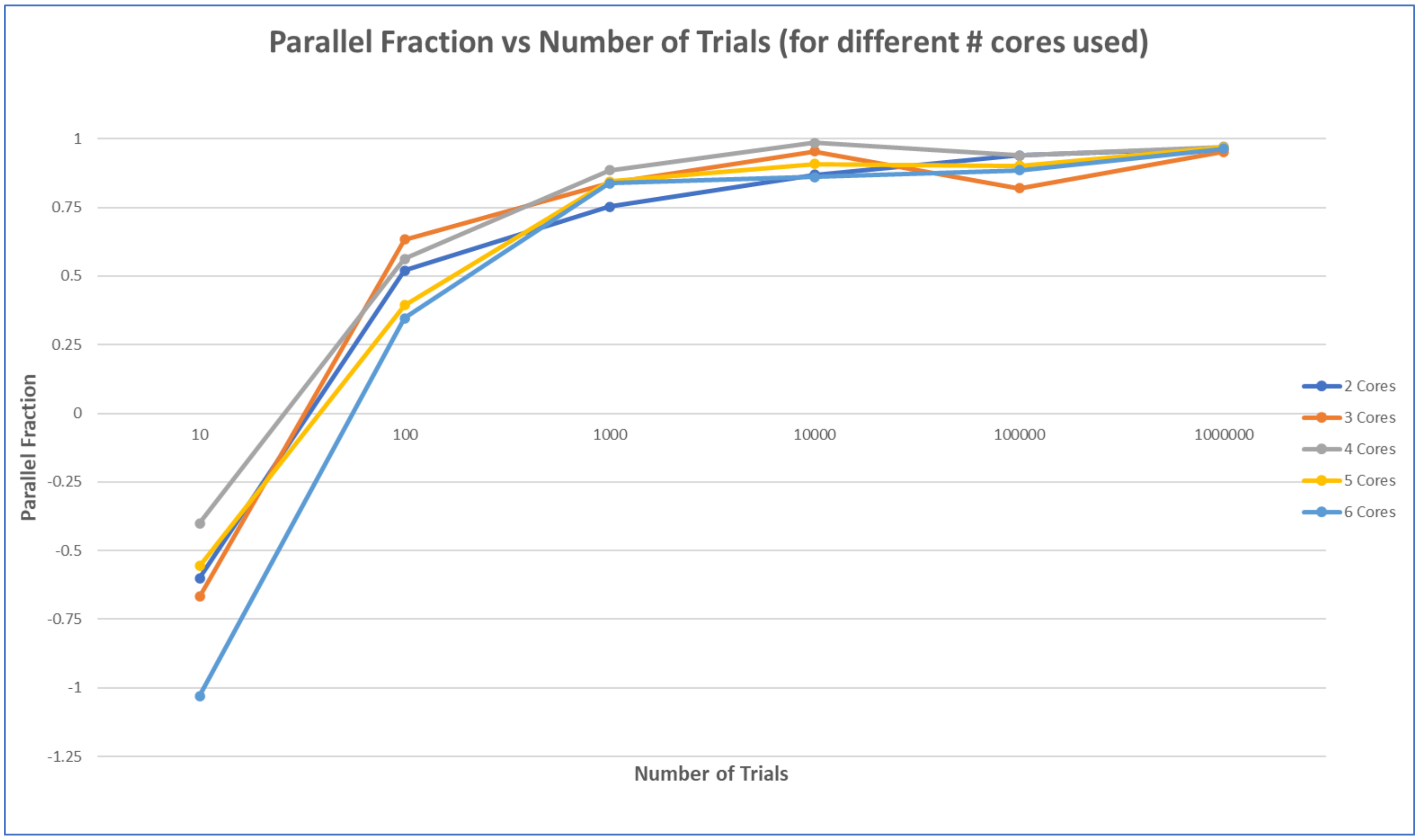


3. Looking at the graph provided below, you can see that the performance gained by using additional cores increases in a linear fashion. I obtained these measurements with NUMTRIALS = 1000000 and NUMTRIES = 100. Here are some of my observations:
   a. The max slope that the line can ever have is 1. This is only the case when Fp if very high. (approaching 1) This is only possible if the size of the dataset is extremely large. It is unlikely that a program can have an Fp that high with a small dataset.
   b. As the dataset gets smaller and smaller, the line is likely to get flatter. This is due to the synchronous portion of the code running a greater percent of the time when the dataset is small. (multi-processing gains can't be reaped when synchronous code runs)



4. The parallel fraction I calculated with NUMTRIALS = 1000000 and NUMTRIES = 10 using the information for 6 cores was: S = 234/46 = 5.09 -then- Fp = (6/5)*(1-(1/5.09)) = .96
   I wasn't sure which parameters to use when calculating this, so I ended up plotting them all. Look at the graphs below. I used excel to calculate the speedup for every core/NUMTRIALS pair and then plugged that into the function Fp = (n/n-1)*(1-(1/S)) for every pair. Excel makes these types of calculations very easy. Here are a couple of my observations:
   a. When NUMTRIALS was 10, the Fp became negative. This indicates that I lost performance by using parallel processing in my program when the dataset was this small. This is likely due to the program taking more time to implement parallel processing than it would have taken a single thread to do all the calculations.

b. This is a good visualization of Gustafson's observation. As the dataset grows larger, the Fp grows larger as well. This validates the notion that performance gained from parallel processing increases as the size of the dataset increases.



Final note on graphs:

Some of the data I gathered did not adhere to the patterns I had expected. I did not rerun the program, because I figured this variation is expected. I could have run my script a few times and averaged all the outputs, but I figured my commentary was more important than getting perfect looking graphs. I hope this aligns with your expectations. Thanks.