Project 1

Trevor Stahl

stahltr@oregonstate.edu

CS 475 Spring 2019

I ran my code on on my personal computer, 2017 MacBook Air

The uptime load averages before and after my runs were in the mid 1.XXs

I compiled with the following command:

```
g++-8 -o proj1 project1.cpp -lm -fopenmp
```

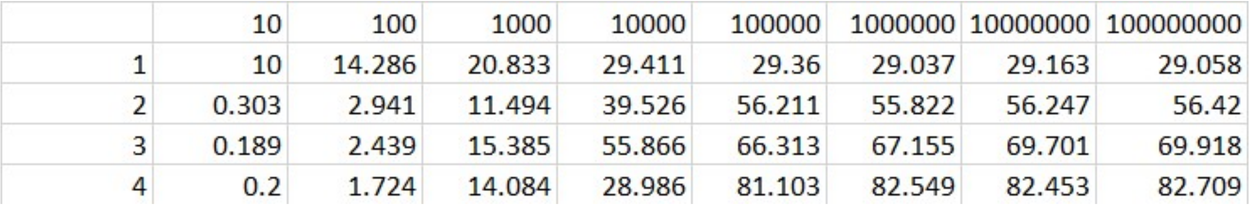
g++-8 is my prefix for gcc 8.3.0

- Run for some combination of trials and threads. Run each with 10 NUMTRIES, record only peak performance.

NUMT values: 1, 2, 3, 4
NUMTRIALS values: 100000000, 10000000, 1000000, 100000, 10000, 1000, 100, 10

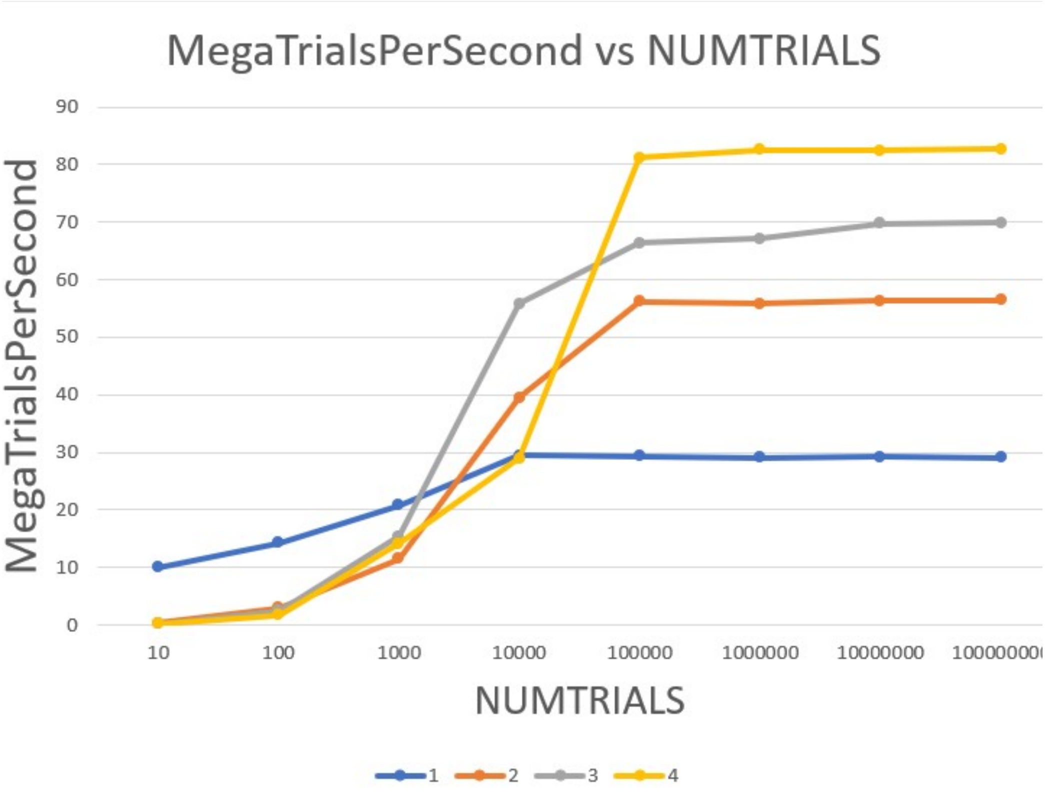
32 different runs

- Table of results



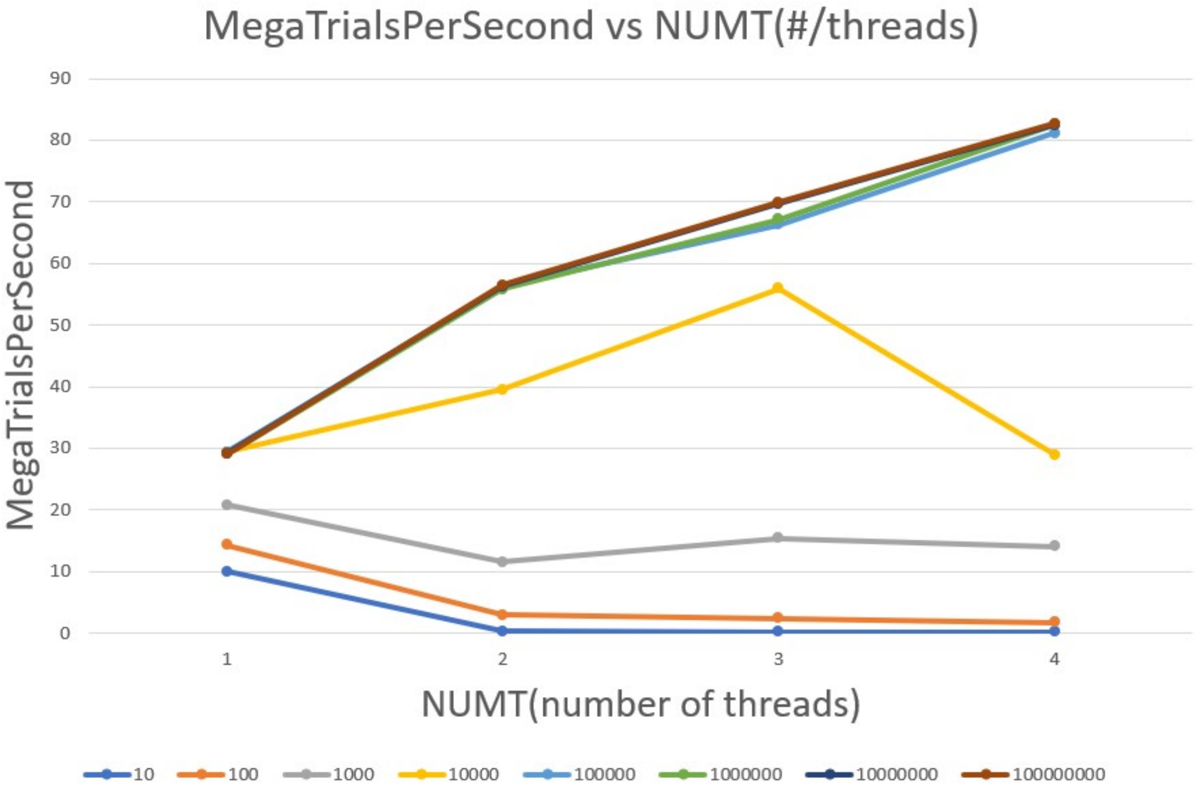
	10	100	1000	10000	100000	1000000	10000000	100000000
1	10	14.286	20.833	29.411	29.36	29.037	29.163	29.058
2	0.303	2.941	11.494	39.526	56.211	55.822	56.247	56.42
3	0.189	2.439	15.385	55.866	66.313	67.155	69.701	69.918
4	0.2	1.724	14.084	28.986	81.103	82.549	82.453	82.709

- Make a graph showing results. Axises with NUMT and NUMTRIALS. Content of MegaTrialsPerSecond.



COLORS ARE NUMBER OF THREADS

- Make a 2nd graph showing performance versus threads.



- Choose one of the runs, the one with the maximum number of trials, and compute probability.

    ~19% - Probability was ~19% for all runs

- Record speedup compared to one thread. I am using performance comparisons from data with the highest NUMTRIALS value. NUMT = 100000000

2 thread speedup:
S2 = (execution time of 1 thread)/(execution time of 2 threads) =
(Performance of 2 threads) / (Performance of 1 thread) = 56.420 / 29.058 = 1.942

4 thread speedup:
S4 = (execution time of 1 thread)/(execution time of 4 threads) =
(Performance of 4 threads) / (Performance of 1 thread) = 82.709 / 29.058 = 2.846

- Compute Fp, the parallel fraction for this computation. I am computing 4-to-1 thread parallel fraction.

The speed up I obtained is S = 2.846

This means Fp = (4.0/3.0)*(1.0 - (1.0 / S)) =
1.33333333333 * (1.0 - 0.3513703443) = 1.333333333333 * 0.6486286557 = 0.8648

Fp = 0.8648