Bryce Graves

Professor Mike Bailey

CS 475

2020 4 15 (ISO 8601)

Project: 1

Bryce Graves
gravebry@oregonstate.edu

```
Running trials with 1 threads
Running trials with 2 threads
Running trials with 3 threads
Running trials with 4 threads
Results!
threads,trials,probability,performance
1,1000000,0.130932,25.395866
1,2000000,0.131213,25.131031
1,4000000,0.130921,25.256504
1,8000000,0.130872,25.131388
1,16000000,0.131063,25.155672
2,1000000,0.130824,49.135174
2,2000000,0.130825,49.139328
2,4000000,0.131023,49.059277
2,8000000,0.131131,49.021507
2,16000000,0.131120,49.014931
3,1000000,0.130159,54.641247
3,2000000,0.130826,54.526138
3,4000000,0.130671,55.919823
3,8000000,0.130859,56.489685
3,16000000,0.130916,58.021381
4,1000000,0.130838,72.605057
4,2000000,0.131380,72.363167
4,4000000,0.131337,72.390884
4,8000000,0.130984,72.555214
4,16000000,0.131125,72.531715


Cleaning data for export

Data clean. Exporting to csv...

Data exported.
Calculating average probability from all tests
Average probability for entire test set: 0.13095094999999998

Calculating FP...
FP: 0.8669587298857158
 .../CS-475/project-1  master  2
```

program which documents it's progress by the current iteration's thread usage. It then outputs the data to the terminal before saving it to a csv file seen in figure 2. But I digress. We are here for the data. This specific set of runs showed an average probability of 13.095%. And an Fp of 0.86696.

To simplify the handling and manipulation of the generated data I decided to create a Ruby program the system commands required to run C++ program with the desired flags and gather the results to export to csv and analyze within the program. I have included this with my other files. It's not my best work, but it does chop up and prepare the data, like a semi competent Itamae, for both graphing and outputting the required calculations.

The first figure is the output of the

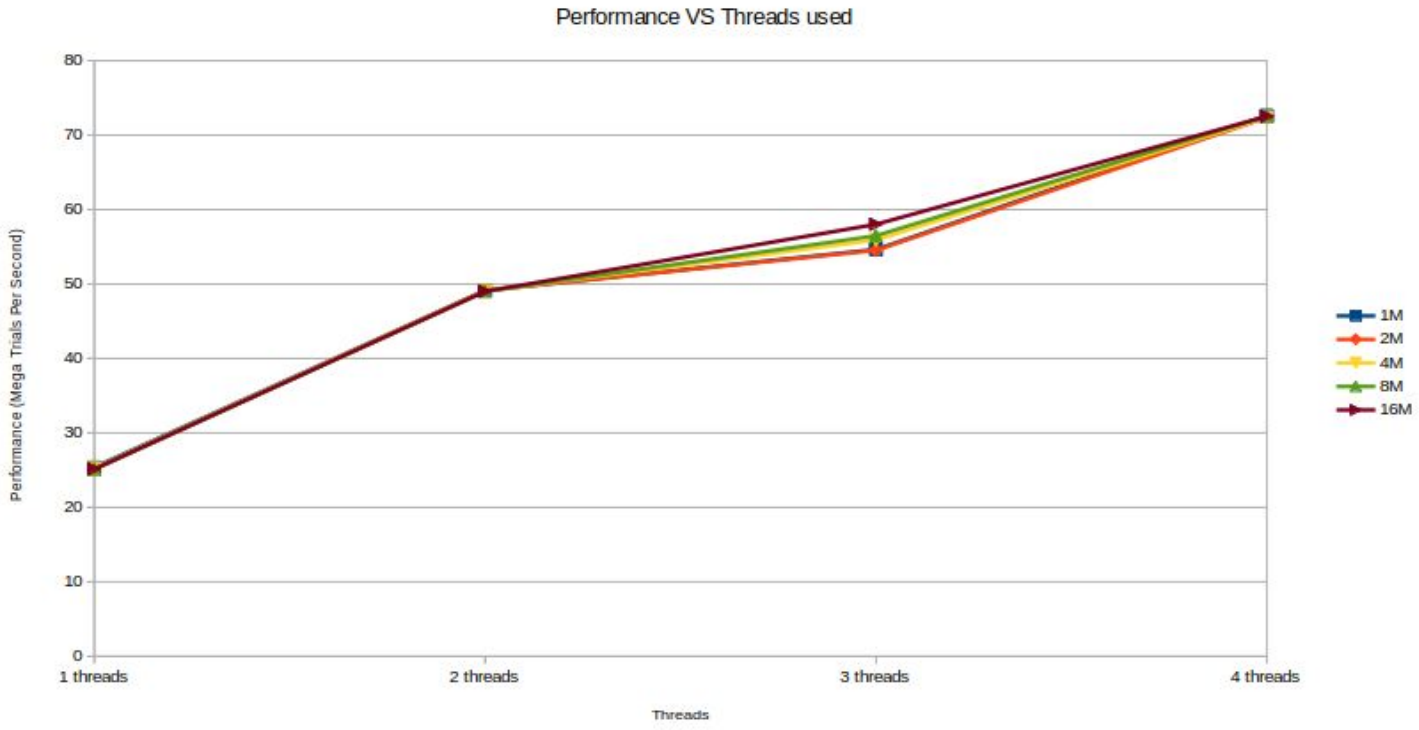
	A	B	C	D	E	F
1	threads	trials	probability	performance		
2	1	1000000	0.130932	25.395866		
3	1	2000000	0.131213	25.131031		
4	1	4000000	0.130921	25.256504		
5	1	8000000	0.130872	25.131388		
6	1	16000000	0.131063	25.155672		
7	2	1000000	0.130824	49.135174		
8	2	2000000	0.130825	49.139328		
9	2	4000000	0.131023	49.059277		
10	2	8000000	0.131131	49.021507		
11	2	16000000	0.13112	49.014931		
12	3	1000000	0.130159	54.641247		
13	3	2000000	0.130826	54.526138		
14	3	4000000	0.130671	55.919823		
15	3	8000000	0.130859	56.489685		
16	3	16000000	0.130916	58.021381		
17	4	1000000	0.130838	72.605057		
18	4	2000000	0.13138	72.363167		
19	4	4000000	0.131337	72.390884		
20	4	8000000	0.130984	72.555214		
21	4	16000000	0.131125	72.531715		
22						
23		1M	2M	4M	8M	16M
24	1 threads	25.395866	25.131031	25.256504	25.131388	25.155672
25	2 threads	49.13517	49.139328	49.059277	49.021507	49.014931
26	3 threads	54.64124	54.526138	55.919823	56.489685	58.021381
27	4 threads	72.605057	72.363167	72.390884	72.555214	72.531715
28						
29		1 thread	2 threads	3 threads	4 threads	
30	1M	25.395866	49.135174	54.641247	72.605057	
31	2M	25.131031	49.139328	54.526138	72.363167	
32	4M	25.256504	49.059277	55.919823	72.390884	
33	8M	25.131388	49.021507	56.489685	72.555214	
34	16M	25.155672	49.014931	58.021381	72.531715	

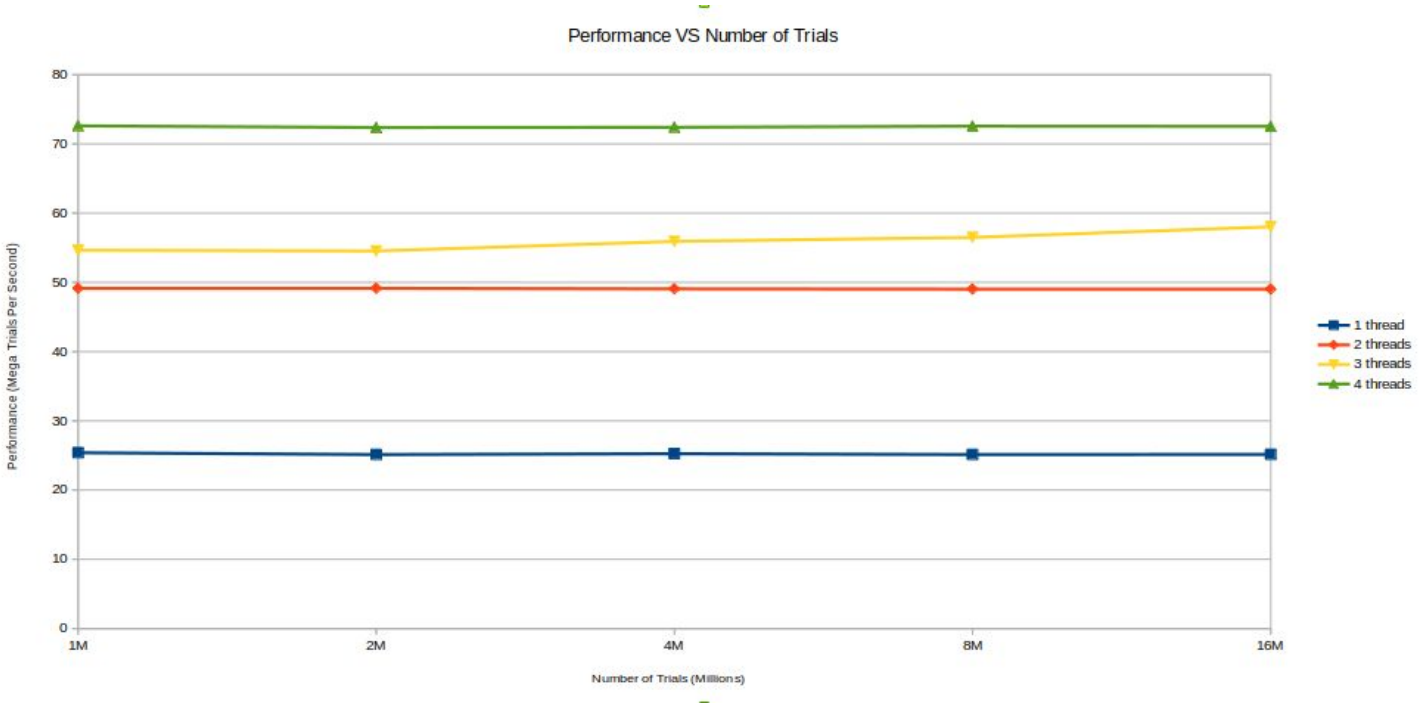
```ruby
one_thread_max = data.select do |line|
  line.first == '1'
end.map(&:last).map(&:to_f).max

four_thread_max = data.select do |line|
  line.first == '4'
end.map(&:last).map(&:to_f).max      You, 15 minutes ago • Uncommitted changes

puts "FP: #{((4.0 / 3.0) * (1.0 - (1.0 / (four_thread_max / one_thread_max))))}"
```

Here are the calculations that generate the previously shown FP output. And below are the graphs of the performance in respect to the increase in trial size and threads used.





Conclusion: Due to the calculations behaving a lot like those from the previous lab, they are able to be done in parallel quite easily with surprisingly little overhead (maybe this is due to the much larger sizes?). The speed up times for these calculations look very similar to those from the previous lab. Overall the performance increases were very consistent likely due to the “smallest” set of trials being 1 million. Maybe if this was pushed lower the earlier smaller tests would be more varied.