Nicholas Skinner

CS 475 Campus

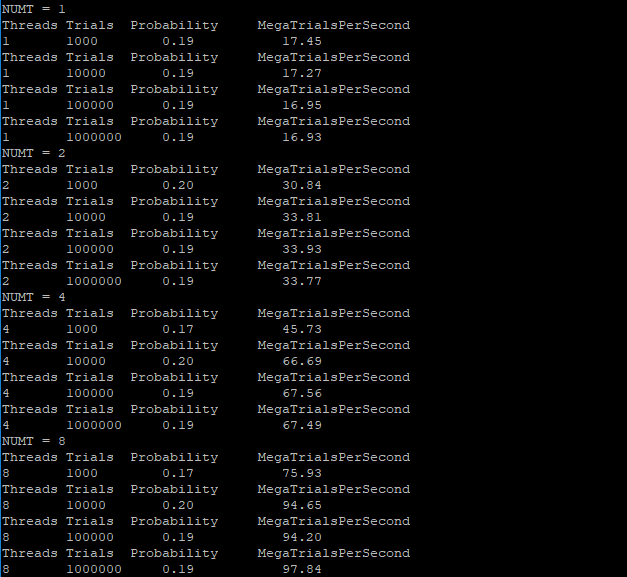
**Project 1 - Simple OpenMP Experiment**

**Platform:**

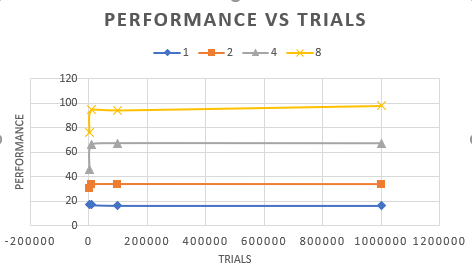
Tests for this project were conducted on the Oregon State University Flip Server.

**Probability:**

According to my executions with larger numbers of runs, the probability narrows down to somewhere in the **19.0%** range, this applies to all my scenarios, and can be seen in the images below where the (1) the number of threads, (2) the number of trials, (3) the probability of hitting the plate, and (4) the MegaTrialsPerSecond are printed out and tabbed to be seperate:

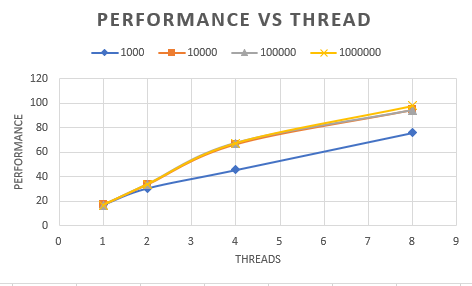


**Performance VS Trials**

****

The above chart is the performance vs the number of trials, it shows a bit of the setup cost to each additional thread that is added. The cost gets steeper per additional thread, and through the 8 thread line, we can see that performance is not quite peaked.

**Performance VS Threads**

****

This chart displays the performance vs threads for each of my number of trials. 1000 trials seemed to scale linearly, while the others took a slight logarithmic growth pattern.

**Parallel Fractions:**

The parallel fraction for this setup

Speedup S = (results 4 threads / results 1 thread)

Amdahl’s Law states:

(n/(1-n)) \* (1-(1/S))

Where n = number of threads

And S = speedup

**1000 Trials:**

Speedup for 2 threads:

S = x/x

S=~1.767

Fp = (2/(1-2)) \* (1-(1/S))

Fp = (2/1) \* (1-.5658)

Fp = 2 \* .4342

**Fp = .8684**

Speedup for 4 threads:

S = x/x

S=2.6206

Fp = (4/(1-4)) \* (1-(1/S))

Fp = (4/3)\* (.6184)

**Fp = .8245**

Speedup for 8 threads:

S = x/x

S = 4.3513

Fp = (8/(1-8)) \* (1-(1/S))

Fp = (8/7) \* (1-0.2298169366521797)

Fp = (1.142857) \* (0.7701830633478203)

Fp = **.8802**

**10000 Trials:**

Speedup for 2 threads:

S = x/x

S = 1.9577

Fp = (2/(1-2)) \* (1-(1/S))

Fp = (2) \* (1-.5108)

**Fp =** **0.9784**

Speedup for 4 threads:

S = x/x

S = 3.861609727851766

Fp = (4/(1-4)) \* (1-(1/S))

Fp = (4/3) \* (1-.258959)

**Fp =** **0.9881**

Speedup for 8 threads:

S = x/x

S = 5.480602200347423

Fp = (8/(1-8)) \* (1-(1/S))

Fp = (8/7)\*(1-0.1824617010036978)

**Fp = 0.9343**

**100000 Trials:**

Speedup for 2 threads:

S = x/x

S = 2.001769911504425

Fp = (2/(1-2)) \* (1-(1/S))

Fp = (2)\* (1-0.4995579133510167)

**Fp = 1.0008** -- this may be due to variance on the machine outside of my control

Speedup for 4 threads:

S = x/x

S = 3.985840707964602

Fp = (4/(1-4)) \* (1-(1/S))

Fp = (4/3) \* (1- 0.2508880994671403)

**Fp = 0.9988**

Speedup for 8 threads:

S = x/x

S = 5.5575

Fp = (8/(1-8)) \* (1-(1/S))

Fp = (8/7) \* (1-0.1799363057324841)

**Fp = 0.9372**

**1000000 Trials:**

Speedup for 2 threads:

S = x/x

S = 1.994683992911991

Fp = (2/(1-2)) \* (1-(1/S))

Fp = (2)\*(0.4986674563221796)

**Fp = 0.9973**

Speedup for 4 threads:

S = x/x

S = 3.990549320732428

Fp = (4/(1-4)) \* (1-(1/S))

Fp = (4/3) \* (1-0.2505920663114269)

Fp = (1.33)\* (0.7494079336885731)

**Fp = 0.9992**

Speedup for 8 threads:

S = x/x

S = 5.779090372120496

Fp = (8/(1-8)) \* (1-(1/S))

Fp = (8/7) \* (1-0.1730376124284546)

Fp = (1.142857142857143) \* (0.8269623875715454)

**Fp = 0.9451**

**Notes:**

I noticed when running these calculations, my megatrials per second for 1 thread went down over time instead of being consistent, this caused my Fp to fluctuate strangely. I believe I do have the formula correct though, and applied it to all of my results.