Trevor Stahl
stahltr@oregonstate.edu

Project #0

Simple OpenMP Experiment

I conducted my experimentation on the flip3 and flip2 servers.
I compiled using the following command:
gcc -o proj project0.cpp -lm -fopenmp

I used an ARRAYSIZE of 200000 and NUMTRIES of 10

The uptime load average recorded before starting tests was:
1.04, 1.28, 1.39
The uptime load average recorded after completing tests was:
1.19, 1.75, 1.69

I ran tests for 1, 2, 4, and 8 threads

I recorded timing precision, peak performance and average performance for each run.

One Thread
Precision: 0.000000000000
Average Performance: 161.75 MegaMults per Second
Peak Performance: 170.66 MegaMults per Second
Average Total Execution Time: 0.001292575395 Seconds

Two Threads
Precision: 0.000000000000
Average Performance: 315.46 MegaMults per Second
Peak Performance: 336.32 MegaMults per Second
Average Total Execution Time: 0.000669329893 Seconds

Four Threads
Precision: 0.000000000000
Average Performance: 630.68 MegaMults per Second
Peak Performance: 675.95 MegaMults per Second
Average Total Execution Time: 0.000345858594 Seconds

Eight Threads
Precision: 0.000000000000
Average Performance: 1218.28 MegaMults per Second
Peak Performance: 1338.21 MegaMults per Second
Average Total Execution Time: 0.000199652393 Seconds

The 4-thread to 1-thread speed up I obtained from these results is:
S = 0.001292575395/0.000345858594 = ~3.73729

The speed up, S, is less than four. This makes sense because although the program uses 4 times as much power by using 4 threads, the ‘overhead’ or firing up the threads means that no speed up will ever be equal to simply the ratio of threads.

Parallel Fraction for 4 thread to 1 thread speed up:
Fp = (4.0/3.0) * (1.0 - (1.0/S)) = (4.0/3.0) * (1.0 - 0.26757324589) = …
= (4.0/3.0) * (0.7324267541) = 0.97656900547

I will not calculate the speed up for 4 to 8, 2 to 4 etc. The values between them seem to be consistent and near what should be expected though which is good, and I would expect Fp values similar for each.