Project 2

Brandon Lee

# Machine Specifications

I ran this project on the OSU’s flip server through my 2015 MacBook Pro.

# Create a table with your results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | static-1 | static-4096 | dynamic-1 | dynamic-4096 |
| 1 thread | 219.91 | 264.88 | 265.23 | 245.38 |
| 2 threads | 442.97 | 362.67 | 512.21 | 402.72 |
| 4 threads | 1024.14 | 706.17 | 1046.81 | 750.95 |
| 6 threads | 749.22 | 619.86 | 1497.84 | 637.2 |
| 8 threads | 1290.52 | 690.36 | 1517.58 | 729.31 |
| 10 threads | 1586.95 | 890.32 | 1797.52 | 889.64 |
| 12 threads | 1368.16 | 687.24 | 2180.88 | 735.17 |
| 14 threads | 1369.22 | 687.02 | 1989.56 | 631.52 |
| 16 threads | 2580.99 | 771.15 | 2856.86 | 717.07 |

\*\*\* Values are represented in MegaMults per Second

\*\*\* Uptime is 18:22:03 up 117 days, 20:08, 104 users, load average: 2.64, 2.86, 3.39

# Create a graph with your results

What patterns are you seeing in the speeds?

I notice that in my test trials, the smaller chunk sizes appear to perform much better than the larger chunk sizes as the number of threads increase. This may be due to the fact that smaller chunk sizes allow for a more uniform distribution of work among the various threads acting on the loop. The dynamic chunk size 1 thread appears to perform the best per number of threads. My hypothesis for this is due to the nature of dynamic scheduling, once one thread is done with its share of work, it is able to help out any struggling threads with their workloads. It is observed that the performance and thread number relationship appears to oscillate as the number of threads gradually increases. I notice that in every third thread (3, 6, 9), performance appears to increase from the previous thread count regardless of scheduling or chunk size.

Why does chunksize 1 vs. 4096 matter like this?

Large and small chunk sizes make a significant impact on this experiment as you increase the chunk size, the more work a single thread has to do before the next thread is able to iterate. This may cause some threads to be slowed down by one another in large chunk sizes as some threads might have to do some heavy lifting in each of their rotations in the for loop. Conversely, having a small chunk size allows for a uniform distribution of iterations per thread as the amount of work per thread is minuscule in comparison.

Why does static vs. dynamic matter like this?

Static and dynamic scheduling provides a difference in performance as static scheduling sets fixed workloads with threads while dynamic scheduling allows for threads with lesser workloads to assist fellow threads with more work. Not all iterations through a loop take the same amount of time. Allowing threads with less work to assist those with more speeds up performance. With the benefit of dynamic scheduling, unutilized threads are delegated work such that more threads are simultaneously assisting in processing the task. This is observable in the graph above. Of iterations with the same block size, dynamic threads have near same performance at worst and exceeding performance at best.