Project #0

Simple OpenMP Experiment

Haoxiang Wang; Student ID: 932359049

1. Machine Running the Test

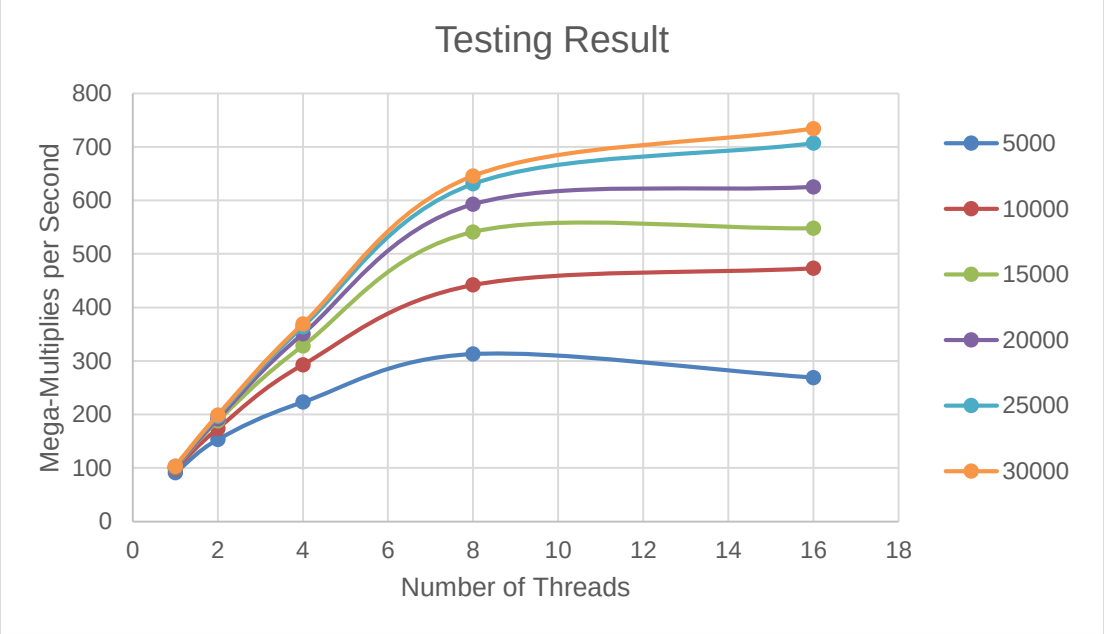
I used school's server "flip" to run the test. At the time my test is running, the "uptime" is less than 1.

2. Performance Results

I used Benchmarks to let the program running with different number of threads, and different size of the array. The number of thread I used for the test are 1, 2, 4, 8, and 16. The size of the array I used in the test are 5000, 100000, 15000, 20000, 25000, and 30000. The performance is evaluated by calculating mega-multiplies per second. The following form is the result I got from the test.

	5000	10000	15000	20000	25000	30000
1	91.19	98.15	100.7	101.94	102.91	103.39
2	153.3	173.64	188.06	191.69	196.68	199.11
4	223.22	293.02	328.34	350.78	364.12	369.24
8	313.02	442.07	541.12	592.98	631.03	645.65
16	269.09	473.36	547.9	625.4	707.16	734.19

In the form, the horizontal direction stands for the size of the array, the vertex direction stands for the number of the threads. In the middle is the original testing results I got from the test. To draw a graph of these results, the x axis will be number of threads, and the y axis will be the mega-multiplies per second. The graph is shown below.



3. Behavior Explanation

As it shows in the graph, each line stands for one particular array size, and each node on the line stands for the performance based on particular number of threads. The bigger value mega-multiplies per second has, the higher performance it shows.

So based on the graph, when the array size is fixed, the more threads the test uses, the higher performance it has. This really makes sense since the more threads work at the same time the shorter time they will use to complete the task. Also, when the thread number is fixed, the larger the array is, the higher “performance” it produces. I think this is because the threads have more work to do and haven’t reach their highest limits. There also exists a point on line 5000 which has lower performance than using less threads. I think this is the point that beyond the “comfort zone” of the test. At this point, I think used too many threads on a small size of array, which might affect the total performance of the test.