Jesse Thoren

April 17, 2017

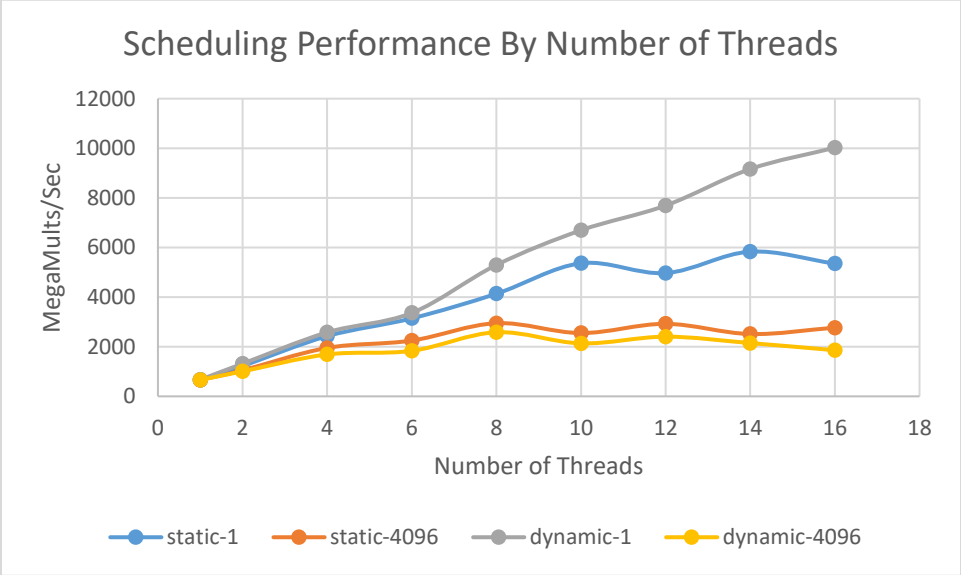
CS 475 – Parallel Programming

Project 2 – Written Commentary

1. My program was run on the OSU Flip server on April 17, 2017.
2. This is a table of Scheduling Performance By Number of Threads. The units on the values in the table are MegaMults/sec, and numbers displayed are the average value of 10 runs with a given schedule and number of threads.

	static-1	static-4096	dynamic-1	dynamic-4096
1	653.1	660.52	670.78	660.94
2	1222.62	1049.68	1315.63	1005.03
4	2426.45	1952.43	2580.01	1689.49
6	3142.93	2248.48	3361.68	1833.83
8	4136.73	2948.37	5295.05	2583.66
10	5365.22	2550.75	6695.1	2135.48
12	4968.64	2922.78	7690.93	2405.08
14	5832.88	2512.1	9162.71	2147.92
16	5350.61	2767.83	10022.56	1858.93

3. Graphed below is the data from number 2.



4. The dynamic test for low chunksize considerably outperforms the static test for low chunksize, especially for larger numbers of threads. In fact, the dynamic test for lower chunksize dominates performance across the board.
   The static test for large chunksize marginally outperforms the dynamic test for large chunksize. In both cases, a large chunksize has significantly worse performance than a small chunksize.

5. Performance suffers in the large chunksize considerably. This is likely because there is a lot of time where the threads are idling, because some of the chunks assigned to threads by the for loop are considerably shorter to complete than others. (The inner for loop varies between 1 multiplication and 32*1024 multiplications).
   With a chunk size of 1, the maximum "worst case scenario" is that one thread gets a command with 32767 fewer multiplications. With a chunk size of 4096, this is on the order of 4096*(32767), which takes a significant extra amount of time to complete.

   Furthermore, you don't see a lot of variation between static-4096 and dynamic-4096 after 8 threads because at 8 threads, the entire load is allocated already. (32*1024/4096 = 8 => 8 threads get a single workload, and the rest of the threads idle).

6. Dynamic allocation should be better than static in this scenario because the workload changes with each iteration of the loop. Rather than statically dividing up the workload with a chunksize of 1, we hand out new assignments as each thread completes its current workload.

   I think the reason this makes a difference is that the static division is always going to favor lower numbered threads. In the case with 8 threads, thread 0 will get chunks with i = 0, 8, 16, … while thread 7 gets chunks with i = 7, 15, 23, … In every case, thread 0 will have a task that is 7 multiplications easier than 7. In the dynamic case, the "hardest question" role gets passed around over time and creates less wait time for the other threads.