Project 2: OpenMP: N-body Problem – Coarse vs Fine and Static vs Dynamic

Parallel Programming

April 27, 2018
Spring Term

Prepared for

Oregon State University

Matthew Meyn

Prepared by

Jeremy Fischer

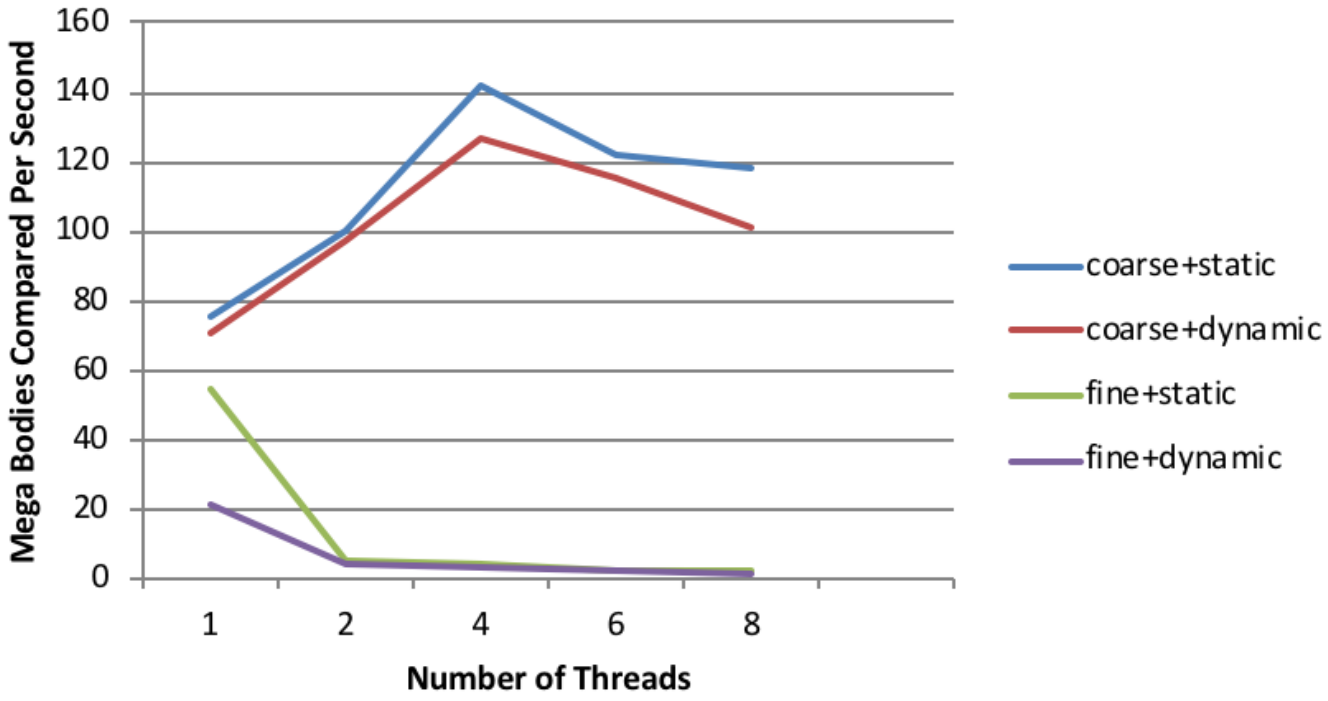
1 Tell what machine you ran this on

I ran this test on a 2015 Macbook Pro, 2.2 GHz Intel Core i7, 16 GB 1600 MHz DDR3.

2 Results

The below values are in mega (10^6) bodies compared per second.

Num Threads	*coarse+static*	*coarse+dynamic*	*fine+static*	*fine+dynamic*
1	75.7	70.98	54.67	21.13
2	100.5	97.69	5.07	4.66
4	141.49	127.2	3.88	3.07
6	122.2	115.81	2.6	2.06
8	118.28	101.57	2.21	1.54



3 What patterns are you seeing in the speeds?

The coarse-grained parallelism for both static and dynamic scheduling increases from one to four threads, and then decreases after. The fine-grained parallelism decreases dramatically as the number of threads increases. In both pairs, the static scheduling is faster than the dynamic scheduling.

4 Why do you think it is behaving this way?

I think the coarse-grained parallelism with static scheduling is faster than coarse-grained parallelism with dynamic scheduling, because this situation is better suited for static scheduling. In static scheduling, OpenMP assigns each thread a set of loop iterations to execute when it enters the parallel block. In dynamic scheduling, OpenMP assigns one iteration to each thread and when it finishes it gets assigned another. This results in dynamic scheduling having more overhead. Dynamic scheduling is better suited for a loop where each iteration may vary in execution time. Since this isn't in the case in this assignement, the dyanmic scheduling is adding unnecessary overhead. The overhead can be seen in the gap between the coarse+static and coarse+dynamic curves in the figure above.

As for why the speed decreases after four threads, I'd assume that there isn't enough work to be done in the for-loops that the overhead of managing more than four threads doesn't pay off in terms of performance. I'd also assume that the coarse+dynamic curve decreases in speed due to additional threads adding to the overhead discussed above.

I think that the speed in the fine-grained parallelism decreases dramatically with an increase in the number of threads because of the repetitive setup and destroy functionality taken on the thread pool. On each iteration the outer for-loop initiates the thread pool, and then destroys them at the end, and there isn't enough work done in the inner for-loop to make that overhead worthwhile.