CS 475 Parallel Computing
Instructor: Matthew Meyn
Student: Khuong Luu

Project 2
Performance Comparison of Static/Dynamic Coarse/Fine parallization methods

1. **Tell what machine you ran this on**
The computer I ran this experiment on has:
- Operating System: Ubuntu 16.04
- Kernel: 4.13.0-37-generic
- Processor: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, 4 cores
- Memory: 16GB, 2400Mhz
- **gcc version 5.4.0 20160609, OpenMP 3.1**

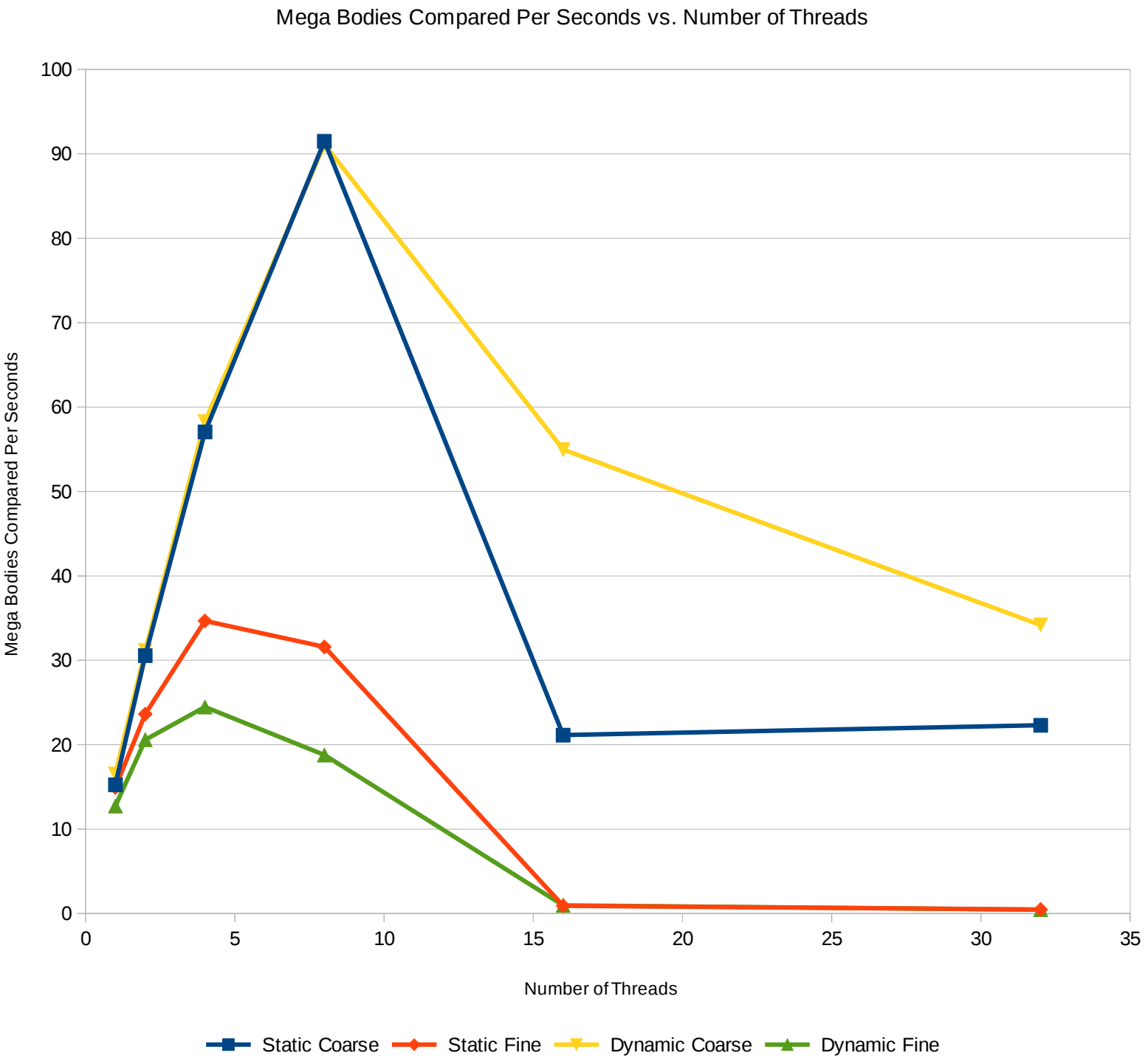
2. Results

Performance unit is in **Mega Bodies Compared Per Seconds**

Number of Threads	Static Coarse	Static Fine	Dynamic Coarse	Dynamic Fine
1	15.2589	14.934	16.53	12.7232
2	30.5632	23.6361	31.1812	20.5869
4	57.0418	34.6663	58.2681	24.4714
8	91.4892	31.5823	91.1095	18.7721
16	21.1354	0.918647	54.9533	0.976772
32	22.3023	0.487383	34.1899	0.460931

3. Graph

Performance Comparison of Static/Dynamic Coarse/Fine parallization methods



4. Patterns I observe and explanations

4.1. Coarse-grained parallelism vs. Fine-grained parallelism

1. Coarse-grained parallelism performance significantly better than fine-grained parallelism in this problem.

2. Coarse-grained parallelism's performance decreases later than fine-grained parallelism performance. Specifically, performance starts decreasing after NUM_THREADS = 4 with fine-grained parallelism but coarse-grained parallelism, the performance only starts decreasing after NUM_THREADS = 8.

Explanation:

1. This is because in coarse-grained parallelism in this problem, each thread has both for-loops and thus "own" more data and has more to compute on their own than each thread in fine-grained parallelism. Therefore, as the number of threads grows larger, threads in coarse-grained parallelism exchange data less frequent and compute more, thus perform better than threads in fine-grained parallelism.

2. Data exchange between threads is also the reason the performance of fine-grained parallelism decays earlier than of coarse-grained parallelism – the more frequent of data exchange, the earlier the cost of it overweight the benefits of parallelism and make the performance start to go down earlier as the number of threads grows up.

4.2. Static scheduling vs. Dynamic scheduling

1. With coarse-grained parallelism, dynamic scheduling has nearly the same performance as static scheduling before their performances start decreasing. In some of my runs, dynamic scheduling sometimes even has worse performance then static scheduling.

2. Dynamic scheduling only show notable better performance over static scheduling in coarse-grained parallelism when the number of threads become large (> 8 threads)

3. However, with fine-grained parallelism, it's the other way around - static scheduling performs clearly better than static scheduling for all number of threads.

Explanation:

1. With coarse-grained parallelism, the workload between iterations in both loops observably differ just little bit. Thus, dynamic scheduling is sometimes very slightly better than static scheduling, but sometimes slightly worse, but both cases are not really notable.

2. Dynamic scheduling only show notable better performance over static scheduling in coarse-grained parallelism when the number of threads become large (>8) in term of slower performance decrease because with larger number of threads, the workload of each threads decrease and the communication between threads occurs more frequent.

3. With fine-grained parallelism, the workload of each iteration of the inner loop (in this algorithm) take nearly the same amount of time, thus the benefit of dynamic scheduling is not much and the communication overhead of dynamic scheduling overweight its little benefit, and so dynamic scheduling performance moderately worse than static scheduling.