Taylor Kirkpatrick

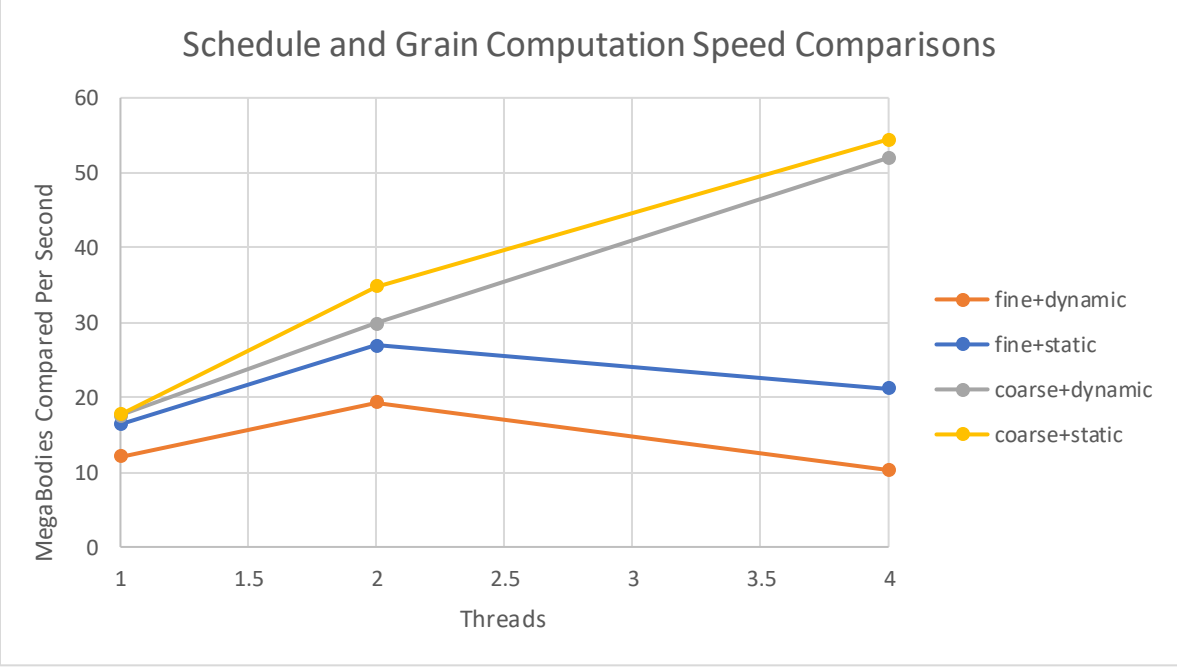
Project 2

1. I am running this program on flip, a server with 24 processors and 6 cores per processor.

2.

Threads	MegaBodies Compared Per Second	Schedule	Grain
1	17.617515	dynamic	coarse
1	17.693551	static	coarse
1	12.067398	dynamic	fine
1	16.385929	static	fine
2	29.821052	dynamic	coarse
2	34.80668	static	coarse
2	19.237397	dynamic	fine
2	26.977616	static	fine
4	51.994519	dynamic	coarse
4	54.462344	static	coarse
4	10.268036	dynamic	fine
4	21.209545	static	fine

3.



4. Coarse-grained speeds improve with more cores, as normal. The small loss of speed-per-core found in the coarse+static line is likely due to server load rather than an accurate representation of the speed changes, but it still shows the proper pattern. The pattern is broken by the fine-grained runs, however. While speed increases normally with a change from 1 to 2 cores, speed drops from 2 to 4 cores. Not only does it not increase as fast, but it actually computes the same amount of data slower. In both fine and coarse cases, dynamic scheduling lags behind static scheduling to varying degrees, though more so in fine-grain than in coarse-grain.
5. One more simple explanation for why coarse-grain is faster than fine-grain is simply that it runs more code in parallel. However this would not explain the odd drop from 2 to 4 threads noticed with the fine-grained computations. I suspect that this is because the Compute:Communicate ratio is changing. Though the grain isn't changing the number of threads, it is changing how much those threads need to communicate with resources that other threads also use. Coarse-grain sharing closes off much more of the program per-thread, allowing threads to avoid this issue. False sharing and other issues that result from this are probably less present in coarse-grained than in fine-grained. As for why this pattern is not present in the switch from 1 core to 2, I suspect this is because 2 cores are always sharing resources, and it is much easier for them to avoid complications like false sharing.