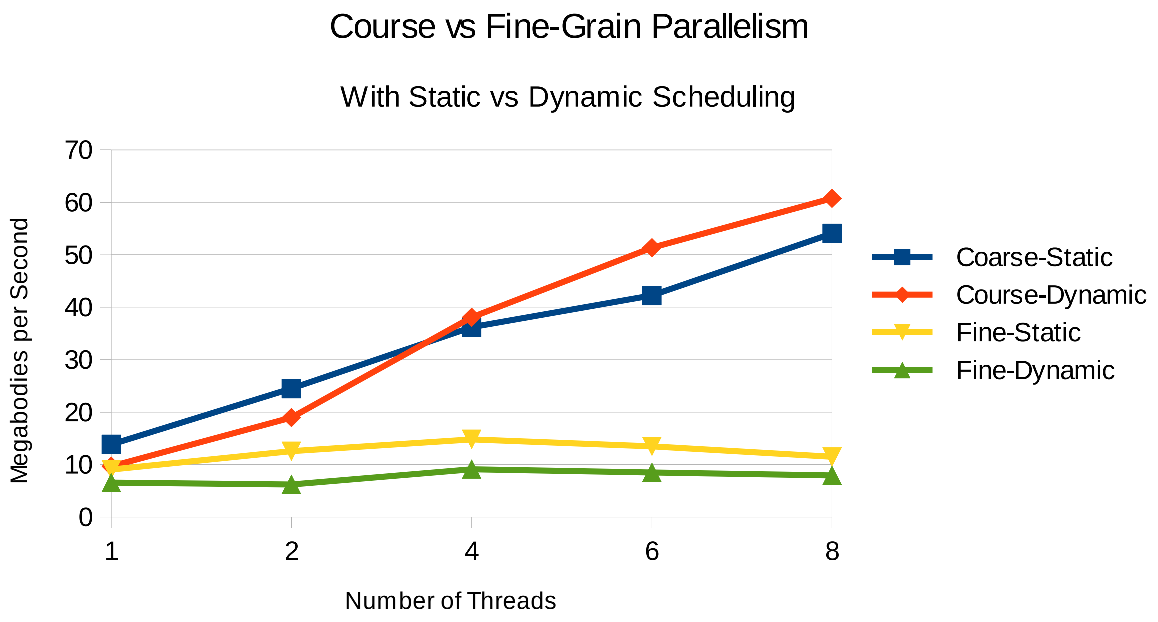
Travis Robinson
CS475
Spring 2016
Project 2

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

This project was run on Oregon States flip servers at access.engr.oregonstate.edu.

Coarse vs Fine Grained Parallelism
With Static vs Dynamic Scheduling

Threads	Coarse-Static	Course-Dynamic	Fine-Static	Fine-Dynamic
1	13.84	9.66	9.04	6.55
2	24.48	18.95	12.53	6.22
4	36.19	38.08	14.79	9.11
6	42.24	51.35	13.47	8.49
8	54.06	60.76	11.46	7.94



For this project we see that at lower thread counts, static performs better than dynamic scheduling. However, it also appears that the dynamic scheduling improves its calculations per second at a faster rate than static, so that it eventually surpasses static in the case of coarse-grain parallelism, and while it doesn't surpass static in fine-grain parallelism, we can see that it does improve at a faster rate and after reaching a peak falls at a slower rate.

We also see that coarse grain-parallelism has significantly better results than fine-grain parallelism. We also see that while fine-grain reaches a peak within the numbe of threads used, coarse grain does not.

The reason that static outperforms dynamic at lower thread counts but ends up being surpassed by dynamic is because of what each scheduling method is doing. Static is assigning particular tasks to each available thread, while dynamic is assigning them on-the-fly, as each thread becomes available. This means that with smaller thread counts, it's easier for static to assign a more-or-less even workload between the processors; this, combined with the fact that the program will need to use processing power for assigning work to threads as the program executes under dynamic scheduling, means that with lower thread amounts static will outperform.

As we see, dynamic shows higher rates of improvement for more threads and eventually surpasses static scheduling performance with higher thread counts. This is because with more threads, there is going to be more dead time with threads completing their assigned duties at varying times under static; with dynamic however, when a thread finishes its tasks it is given another, meaning it is there is no dead time for threads so calculations are completed faster.

The reason that coarse-grain parallelism does so much better than fine-grain is because with coarse-grain is because in this particular project, fine grain has too much communication. There's a balancing act in program coarseness between computation and communication. In this project, with the fine-grain parallelism, there's too much communication between threads resulting in a slow down. This is also why there's a peak for fine-grain at 4 threads. Beyond this the amount of communication slows things down the more threads there are due to the amount of communication between threads. This is not a problem with coarse-grain for this project, having a better ratio between computation and communication.