CS 575

Project #2

Name: Chi Wen
ID: 933-276-677
Email: wench@oregonstate.edu

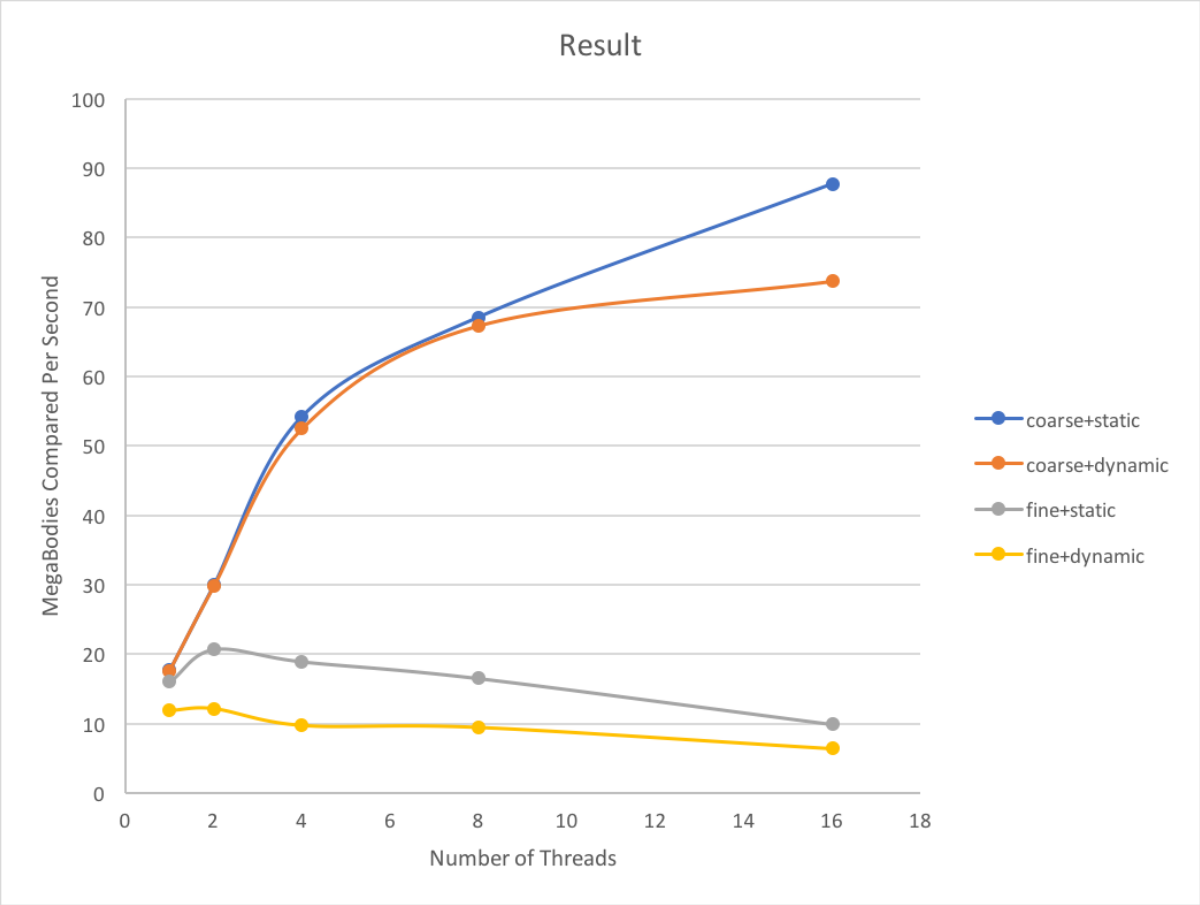
1. **Tell what machine you ran this on**

   School server, flip2.

2. **Create a table with your results.**

	1 thread	2 threads	4 threads	8 threads	16 threads
coarse+static	17.66	29.94	54.23	68.51	87.67
coarse+dynamic	17.64	29.77	52.5	67.32	73.73
fine+static	15.99	20.68	18.92	16.5	9.89
fine+dynamic	11.86	12.12	9.74	9.45	6.43

3. **Draw a graph. The X axis will be the number of threads. The Y axis will be the performance in whatever units you sensibly choose. On the same graph, plot 4 curves: coarse+static, coarse+dynamic, fine+static, and fine+dynamic**



1. **What patterns are you seeing in the speeds?**

   For coarse-grained parallelism, in both static and dynamic scheduling, when the number of threads increases, it will show an increase of speed. Between static and dynamic scheduling, static scheduling went up way faster than dynamic scheduling, and it is very obvious when the threads is bigger than 8.
   As for fine-grained parallelism, in both static and dynamic scheduling, when we use multiple threads, it will lead to a decrease of speed. So that if we divide it to two threads, it is will have the fastest speed.

2. **Why do you think it is behaving this way?**

   In the graph, no matter it is static scheduling or dynamic scheduling, coarse-grained parallelism are always better than fine-grained parallelism, due to coarse-grained parallelism is the i loop, and it can break the task at once; however for fine-grained parallelism, it will have to break the task into a lots of small tasks many times.

   In this program, static scheduling is always faster than dynamic scheduling, due to the reason of each of the task's workload is alike, in this situation, dynamic scheduling can't show its advantage. Dividing the total number of tasks for each thread to do is much effective than dynamic assigning task to the available thread.