Khoa Nguyen - Lab 5 - Lab Report - CS360 - 03/04/2022

In the lab5-prime problem, the behaviors of each schedule were mostly expected. In general, default type had the slowest runtime, followed by static type, then dynamic type, and lastly guided type. One exception was that when the chunk size was 1, the runtimes of static were unexpectedly the lowest. The reason was because when using schedule (static, 1), the number of cores being used was always less than the amount of threads the program asked for. Also, the runtimes when using 16 threads were approximately two times faster than that of running with only 8 threads, which was expected. The results were recorded in the table below:

Primes (3, 50,000,000)	Default	Static	Dynamic	Guided
Thread 8 Chunk 1	14	20	11	9
Thread 8 Chunk 50	14	10	10	10
Thread 8 Chunk 100	14	10	10	10
Thread 8 Chunk 1000	14	10	10	10
Thread 16 Chunk 1	7	10	6	6
Thread 16 Chunk 50	7	5	5	5
Thread 16 Chunk 100	7	5	5	5
Thread 16 Chunk 1000	7	5	5	5

In the lab5-curve problem, for chunk size of 1, the dynamic schedule had a slower-than-expected and the slowest runtime (13 seconds vs an average of 8 seconds by other schedule options). Unfortunately, I could not find the exact reason for that behavior. When I ran the program, I checked the cores and it seemed like all cores I asked for were being used. I hypothesized that the process of waiting for each core to complete their current job and assign a single new job with a chunk size of one would not be as efficient as assigning jobs before runtime. As the chunk size got bigger, this process became less significant. In all other cases, the runtimes of each schedule option were similar to each other, which was expected since the time to calculate each trapezoid should also be similar regardless of the input values. Overall, the runtimes with 16 threads were two times faster than that of 8 threads, which was expected. The results were recorded in the table below:

Curve (0, 1000000) n = 100,000,000	Default	Static	Dynamic	Guided
Thread 8 Chunk 1	8	9	13	8
Thread 8 Chunk 50	8	8	8	8
Thread 8 Chunk 100	8	9	8	8
Thread 8 Chunk 1000	8	9	8	8
Thread 16 Chunk 1	4	5	13	5
Thread 16 Chunk 50	5	4	4	5
Thread 16 Chunk 100	5	4	5	4
Thread 16 Chunk 1000	5	4	4	5