Brian Bowles

CS 3100

Assignment 3 Paper

January 21, 2015

When running a program on multiple threads you usually experience a decrease in runtime because each thread is doing something, but the threads run at the same time. For an 800 x 800 image of the Mandelbrot with one thread it took nearly two seconds to generate the Mandelbrot and decreased in time by nearly logarithmic amount for each thread added. Two threads took only 1 second, which is a major decrease while the others aren’t so much due to the fact that I’m using a dual core processor, four took 0.9 seconds, and eight took about 0.6 - 0.7 seconds, sixteen took anywhere from 0.5 to 0.6 seconds, so it was usually faster than using eight threads, but not always. Because the threads work in parallel they make for a much quicker execution of a program, unless too many threads are used, then it starts to increase in time again because of the scheduling of the computer.

Threads seem to be very useful when programming, but you also must be very careful in your use of threads because if there are multiple threads accessing the same memory and at least one writes to it that causes a data race, which is extremely bad. To avoid data races the programmer can use const everywhere possible and make sure the threads are never accessing the same memory by creating a mutex to lock data while it’s in use by a thread. The concept of threads involved in this assignment is pretty simple, but it required a lot of work to implement those threads to make the program run a little faster. The best part about this assignment was when I used two different computers to test the time and the ones in the TSC gave me 1731 ms, 857ms, 811ms, 530ms, and 358ms respectively with 1, 2, 4, 8, and 16 threads while the computers in the CS tutor lab gave me the ones listed above; it’s really cool to see how different machines perform in comparison.