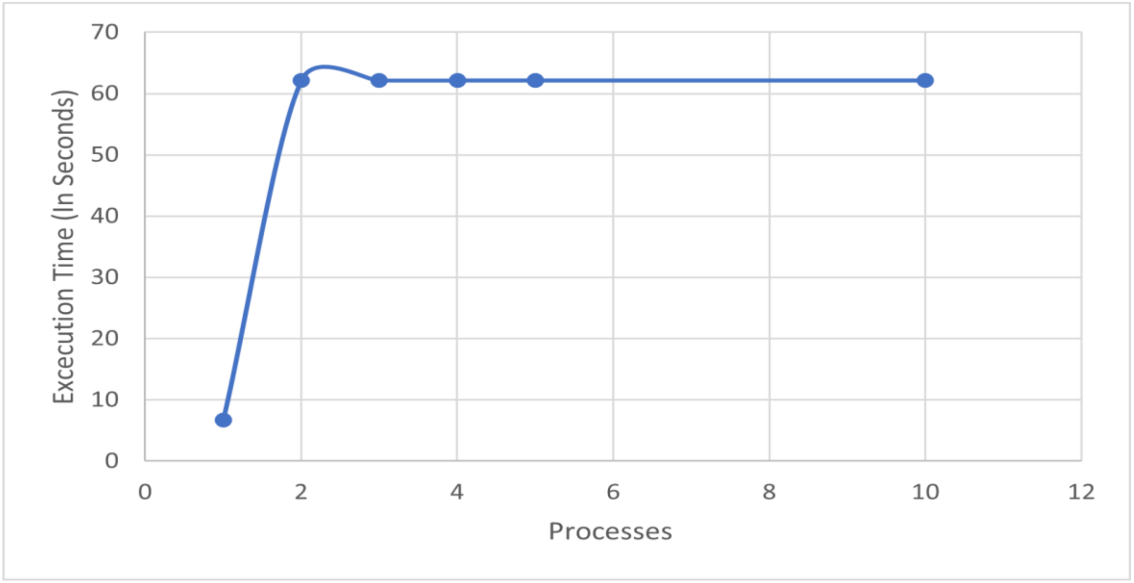
1.) *In your own words, briefly explain the purpose of the experiments and the experimental setup. Be sure to clearly state on which machine you ran the experiments, and exactly what your command line arguments were, so that we can reproduce your work in case of any confusion.*

In essence, the purpose of the experiments is to create an image of the Mandelbrot set all while utilizing multiple threads and command-line arguments. In the aspect of experimental setup, I connected to FIU's server via Ocelot (ocelot.aul.fiu.edu) and utilized PUTTY and Filezilla in order modify the given source code and generate the image.

2.) *Measure and graph the execution time of mandelmovie for each of 1, 2, 3, 4, 5, and 10 processes running simultaneously. Because each of these will be fairly long running, it will be sufficient to measure each configuration only once. Make sure that you write all of your output files somewhere in /tmp to avoid any timing problems with the network file system.*



This graph shows the processes (1,2,3,4,5,10) and their execution times (in seconds). Below is the process and exact time.

Process 1: 61.710 seconds

Process 2: 65.817 seconds

Process 3: 60.799 seconds

Process 4: 59.429 seconds

Process 5: 67.394 seconds

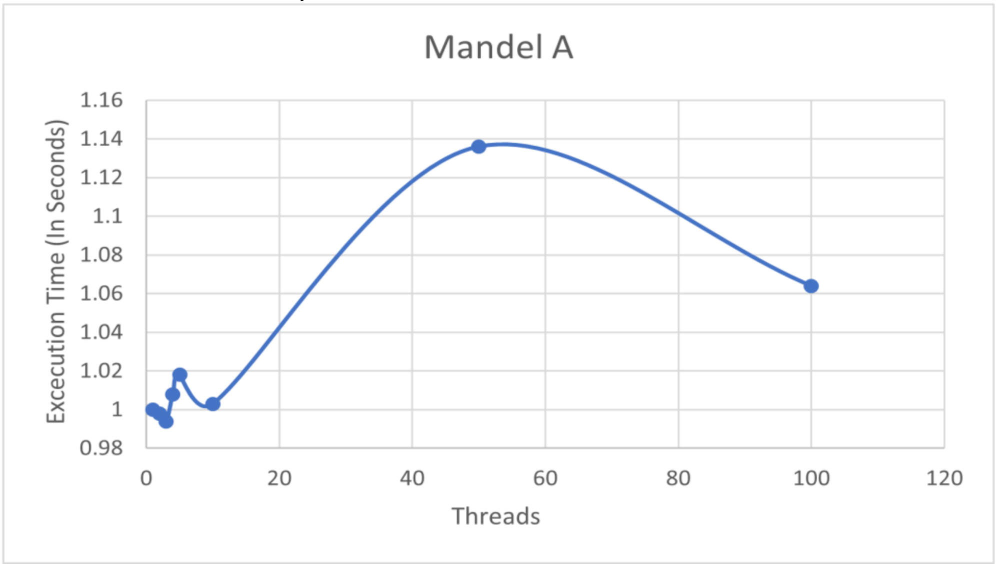
Process 10: 62.136 seconds

3.) Explain the shape of the curve. What is the optimal number of processes? Why? Is it possible to have too many processes? Why?

In my graph, the ‘curve,’ seemed to go up and down meaning the times were relatively similar. As far as the optimal number of processes? Well can should be around 3 processes as the lower they are, the better results you’d get. Now if you begin to have more processes as time goes on then you are defiantly going to see huge discrepancies and dips in the program’s performance. So, yes, it is very possible to have too many processes.

4.) *For the following two configurations, measure and graph the execution time of multithreaded mandel using 1, 2, 3, 4, 5, 10, 50, and 100 threads. The execution time of these experiments may be upset by other things going on in the machine. So, repeat each measurement five times, and use the fastest time achieved.*

   a. ***A:** mandel -x -.5 -y .5 -s 1 -m 2000*
   b. ***B:** mandel -x 0.2869325 -y 0.0142905 -s .000001 -W 1024 -H 1024 -m 1000*



Below are the execution times and threads for Mandel A.

Mandel A:

Thread 1: 1 seconds

Thread 2: 0.998 seconds

Thread 3: 0.994 seconds

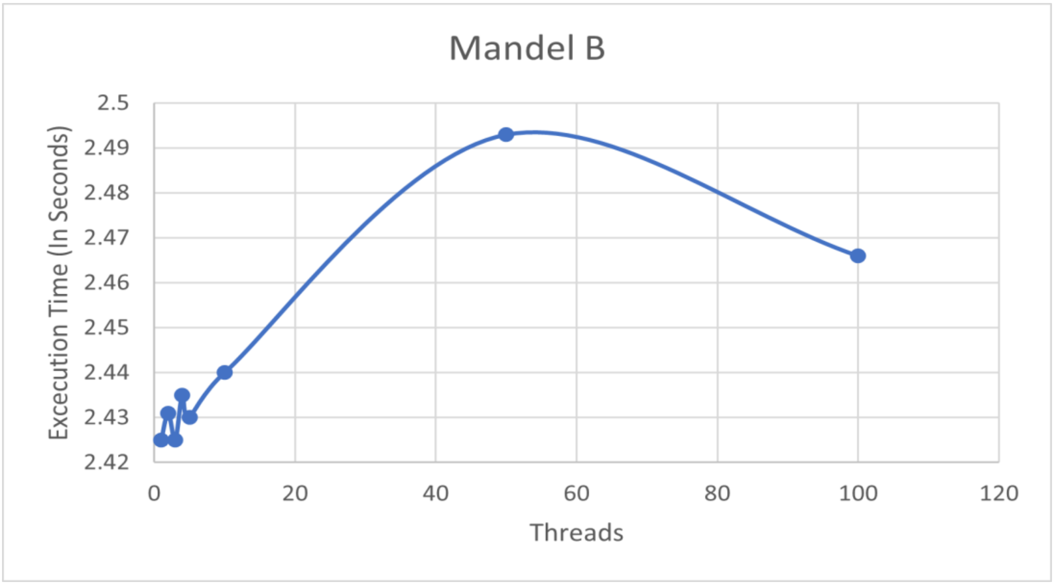
Thread 4: 1.008 seconds

Thread 5: 1.018 seconds

Thread 10: 1.003 seconds

Thread 50: 1.136 seconds

Thread 100: 1.064 seconds



Below are the execution times and threads for Mandel B.

Mandel B:

Thread 1: 2.425 seconds

Thread 2: 2.431 seconds

Thread 3: 2.425 seconds

Thread 4: 2.435 seconds

Thread 5: 2.430 seconds

Thread 10: 2.440 seconds

Thread 50: 2.493 seconds

Thread 100: 2.466 seconds

5.) *Explain the shape of the two curves. What is the optimal number of threads? Why do curves A and B have a different shape? (Hint: Modify your program to print a message when each thread starts and stops.)*
The two curves are representing the program's execution time per the number of threads given within our question (1,2,3,4,5,10,50,100). As per my execution times on my computer, Mandel A seems to stay the same with little to no drops in time within the number of increasing threads given. As for Mandel B, while time is rather similar as well, the difference in time is bigger than the time difference within Mandel A. Regardless, the number of threads is really reliant on the user's computer and the workload that's given to the computer. Curves A and B tend to have different curves simply because of the fact of the different parameters given to us which gives us vastly different times.