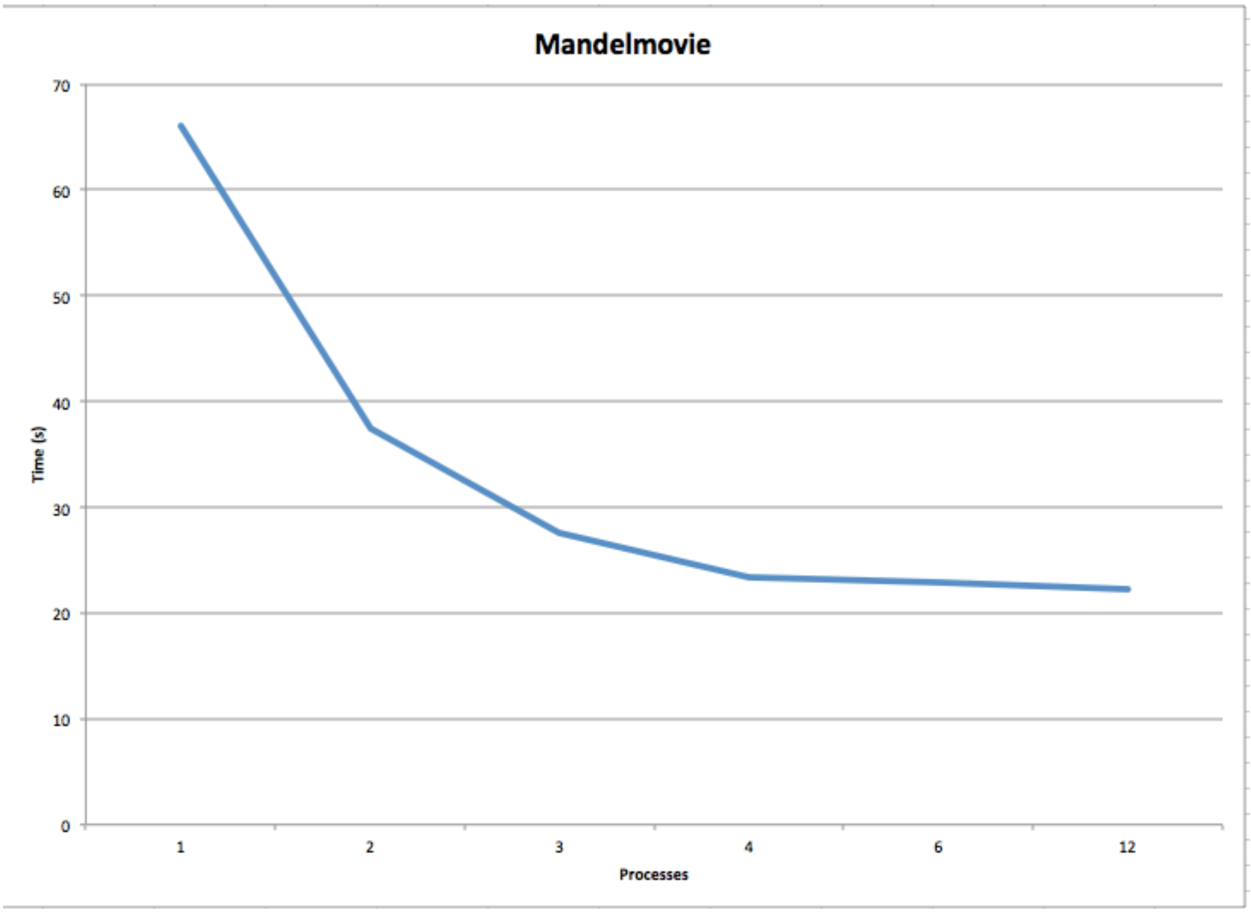
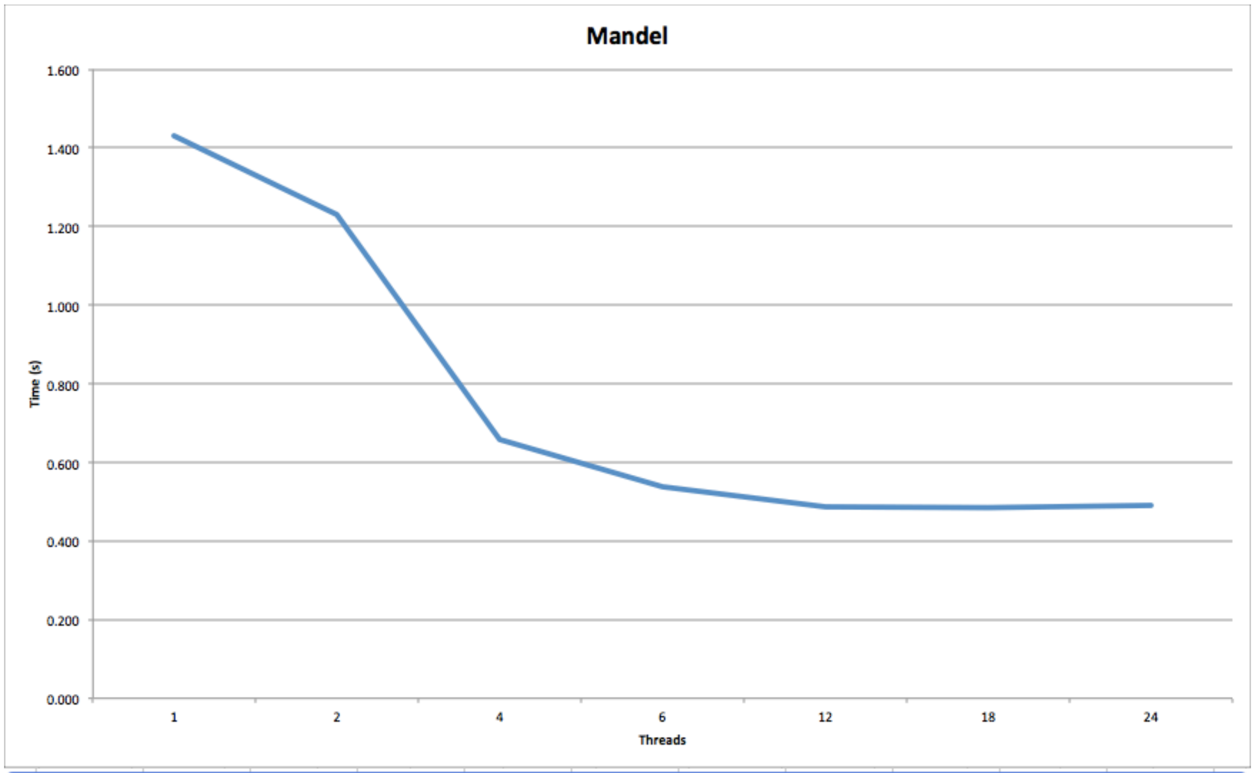
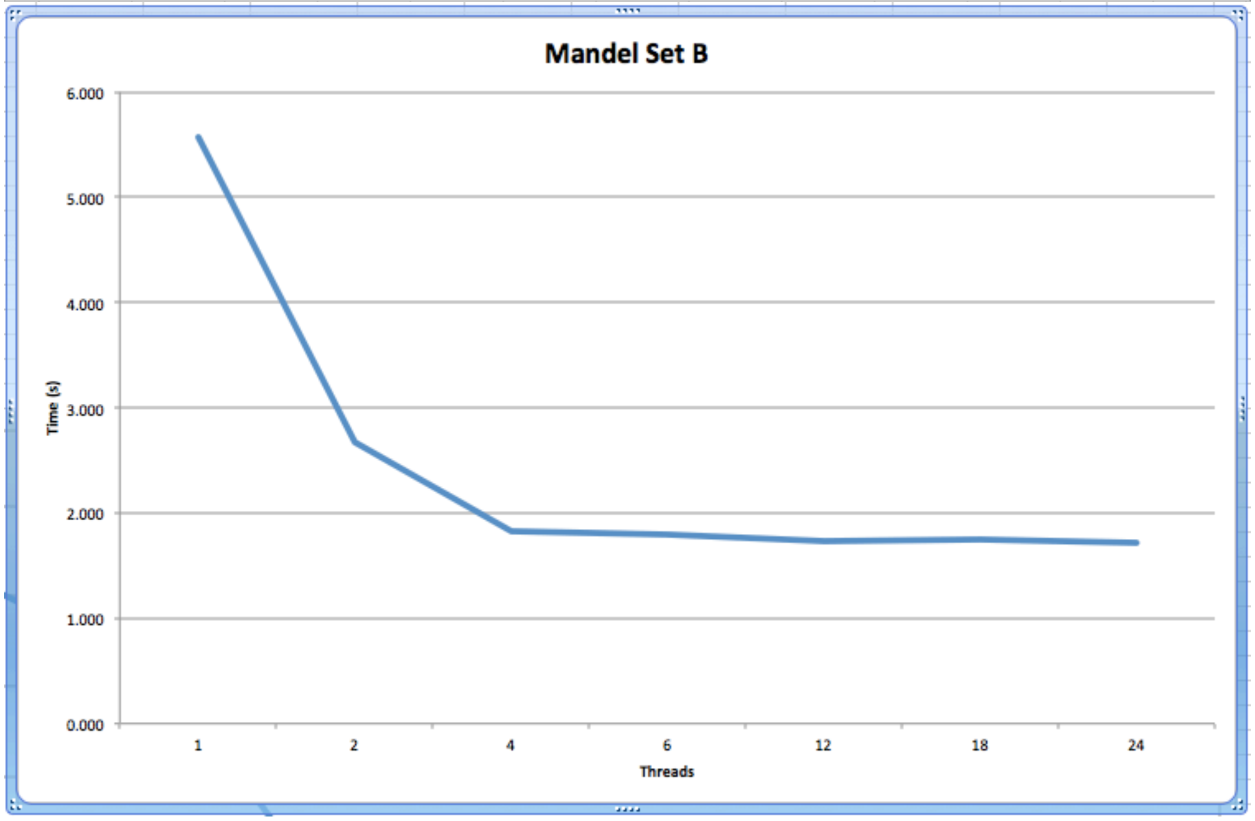
Grace Bushong
February 24, 2017

The purpose of the experiments was to see if adding multiple processes or threads actually improves the speed of the program. If you have too many processes or threads you can spend more time creating them than the time they save diving the work, so there is often an optimal number of processes or threads to create to divide the work well.
I ran the experiments on my personal laptop, but I tested my programs on student00.cse.nd.edu. I ran mandelmovie using ./mandelmovie 1… 12 and I ran mandel using both of the specified tests {A and B} with –n <threads> added on to the end.



This graph indicates that the optimal number of processes is 3-4. Once you get past this the time advantages drastically decrease per process added. Having more processes than that wastes time in making them.





These graphs indicate that the optimal number of threads is around 4, because that's where the time saving levels off. These graphs have a different shape because different commands are being run. Set A has less of a time decrease from 1 to 2 threads. Set B levels off almost completely at the end but decreases drastically at the beginning. Splitting the work between two threads helped more in Set B than it did in Set A, indicating that Set A had more overhead that had to be taken care of outside of the threading.