Brad Sherman
Professor Thain
Operating Systems
Project 3 Report

Purpose:

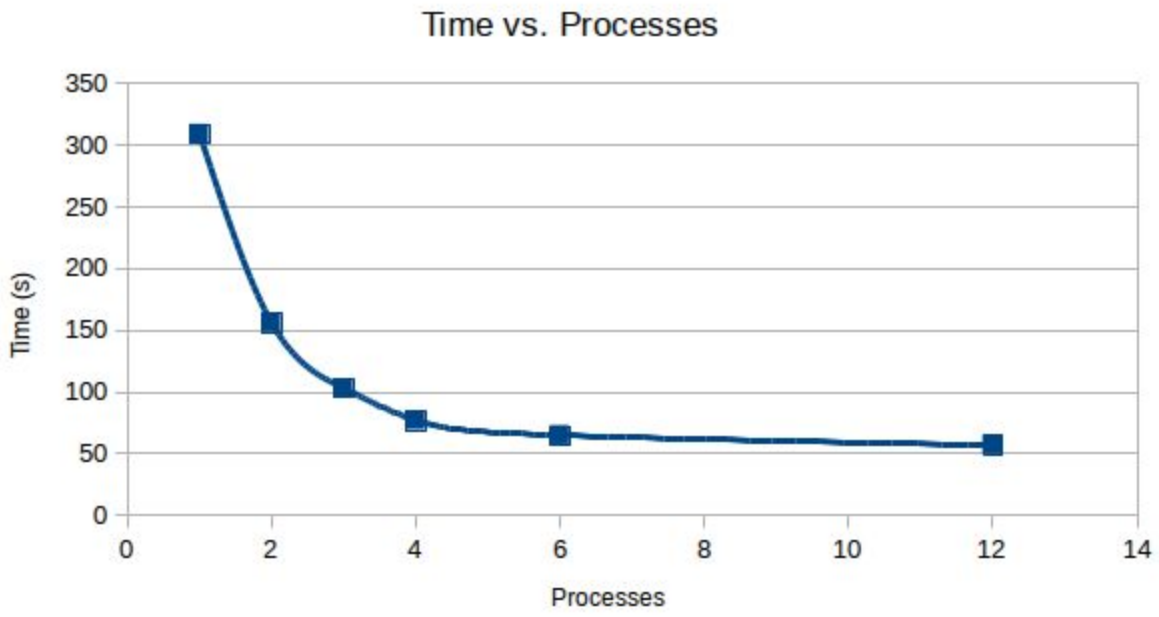
The purpose of this experiment was to investigate the benefits/tradeoffs of using multiple processes/threads to perform expensive computation. It is obvious that splitting the work up to be done by multiple processes/threads will generally decrease runtime, but we want to find out where the threshold is and if one can split a task up into too many processes or threads. My experiments were performed on my laptop, a Thinkpad W540 with an Intel Core i7 processor and 8GB memory. I store my created pictures in “pics directory” in the current working directory. In my mandelmovie.c, the exact command I run is:

```
./mandel -x -0.542900 -y 0.599587 -s .000001 -m 2000 -W 1000 -H 1000
```

Where the scale starts at 2 and goes down to .000001.

Mandelmovie:

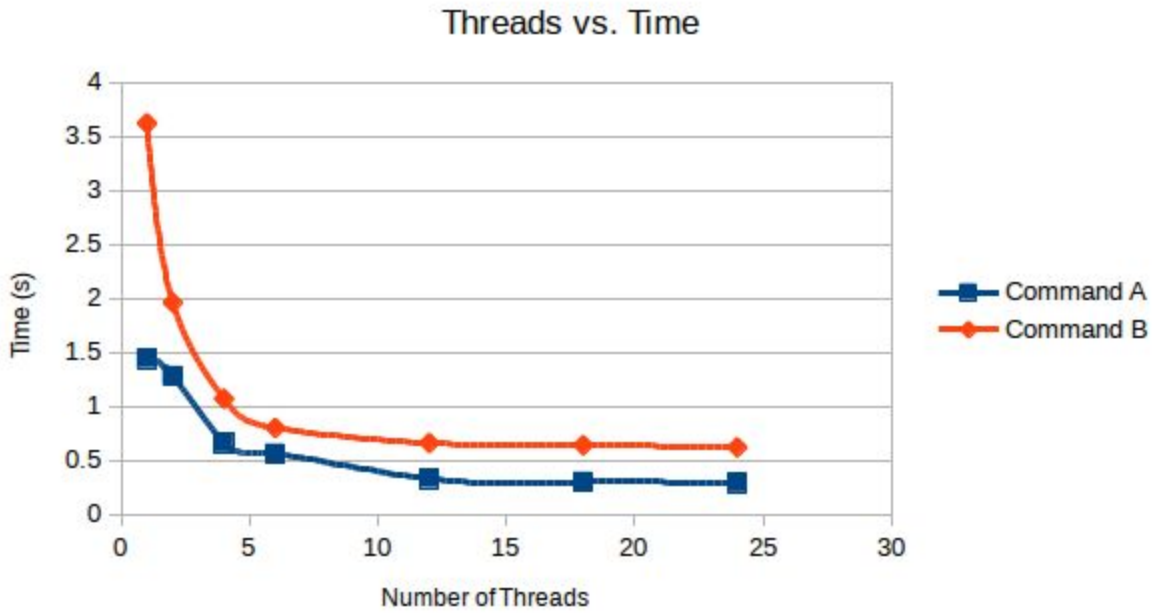
After running mandelmovie with the specified parameters, I created this chart:



I think that the optimal number of processes is 6, because we can still see a slightly lower time from 4 to 6, but when we look at the difference between 6 and 12, the difference in time is essentially negligible. I also think that because of the shape of this graph, we can see that we can have too many processes. Once we get to 6 processes, the amount of time gained from each extra process is negligible, so the overhead of setting up more processes isn’t even worth it. The graph basically becomes a flat line after 6-8 processes.

Mandel:

After running mandel with the two specified configurations I created this chart:



Both of these curves have a similar shape, but Command B took significantly more time to complete than Command A. Presumably this is because it is a larger image and at a much smaller scale. It seems that the optimal number of threads is around 10. As was the case for using multiple processes, the benefit of adding more threads after 10 is negligible, so I would say 10 is the maximum amount of threads we should use to create an optimal image.