Ryan Loizzo, Nathan Rao
3/1/2018
Operating Systems

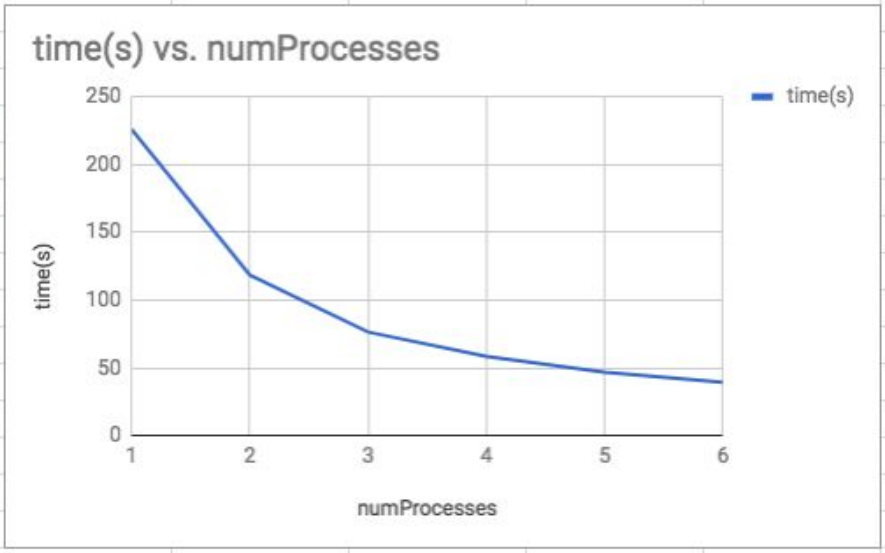
Project 3 Report

The Mandelbrot Parallelism project explored the use of processes and threads to help the speedup and performance of our code. Different numbers of processes and threads yield vastly different usage statistics and effect how, using fork and pthreads, the Mandelbrot set is visualized. For our CPU, operating system, memory, and model, we assume this is asking about our local machine, which is a MacBook Air running macOS High Sierra 10.13.3 with 1.4 GHz i5 on 4GB RAM. We ran all of our tests on the student04 machine.

Evaluation of processes on mandelmovie:

./mandel -x -.354632 -y .634567 -s .0001 -m 2000 -H 1000 -W 1000 -n <num_threads>

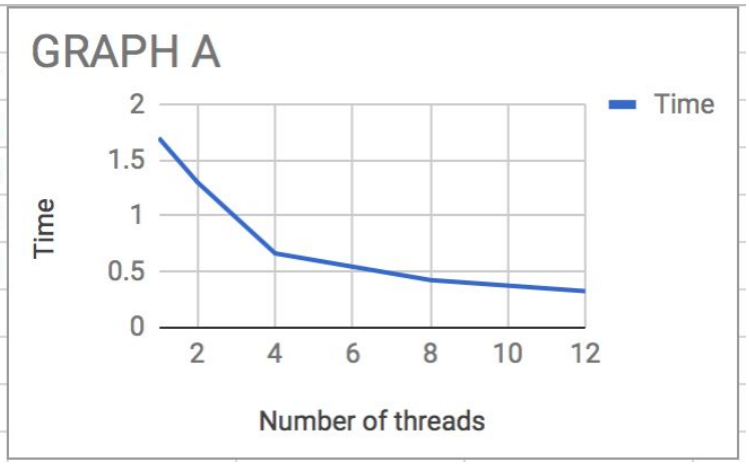
numProcesses	time(s)
1	225.94
2	118.53
3	76.46
4	58.46
5	46.78
6	39.39



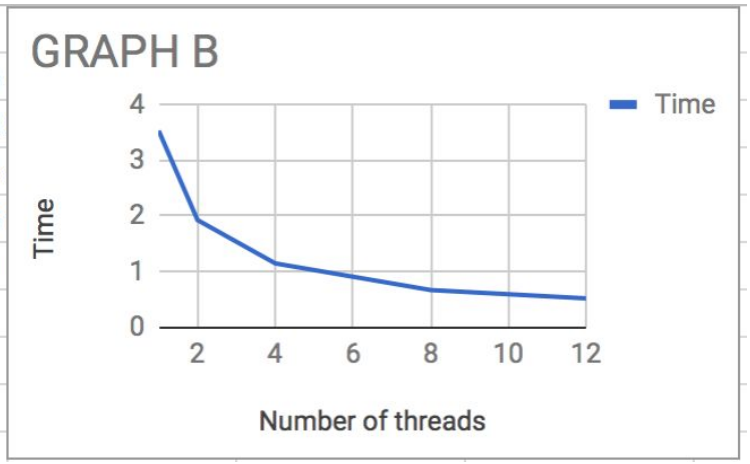
This graph of mandelmovie, a negative exponential curve, directly follows the predicted action of the code. As more processes are added, the amount of time to complete the creation of all 50 images reduces. The reason the graph is a negative exponential is because at some point adding another process does not significantly reduce the amount of time - a threshold or limit. The optimal number of processes appears to be around 5 or 6, as only a few seconds are saved by adding those processes, and even less additional time will be saved when adding more than those. Having too many processes, additionally,

Evaluation of threads on mandel:

Number of threads	Time
1	1.7
2	1.3
4	0.66
8	0.42
12	0.32



Number of threads	Time
1	3.53
2	1.92
4	1.14
8	0.66
12	0.51

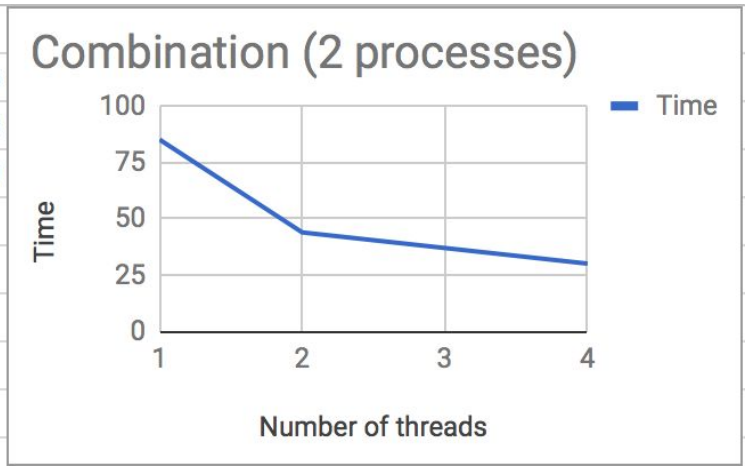


These graphs of mandel, which both generate a negative exponential curve, also directly follows the predicted action. As the number of threads increases, the time should decrease since these threads are all working at the same time. At some point, however, the return on the number of threads in negligible, which is why the graph is negative exponential. Graph A shows a significantly less steep curve. This is because the output of graph B is much larger and takes more time to compute than graph A, so the effect of threading has a more noticeable effect.

Evaluation of the hybrid approach on mandelmovie:

Change the number of threads within mandelmovie.cpp by changing myargs[14] to the desired number of threads.
After compiling with the Makefile, run mandelmovie with 2 processes by using the command ./mandelmovie 2

Number of threads	Time
1	85.06
2	43.85
4	29.99



This graph also shows a negative exponential curve, as expected. While a decrease in time was easily expected, I was expecting the speedup to be more significant. If I increased the thread count to 8, I would expect the time to still decrease, but by a pretty small margin. Based on the marginally small decrease in time as the number of threads, I would say that increasing the number of processes is more advantageous than increasing the number of threads. There is more overhead that goes into creating threads, while creating more processes does not require this overhead. Thus, adding processes has a more significant speedup than adding threads.