Nicholas Newton
2/17/22

OS Report

Section 1: Experiment Purpose

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.

The purpose of the experiment was to apply the understanding of processes and threads with regards to execution time. The setup was designed to implement execution of a program through multiple processes and through multiple threads. The results of the experiment were designed to see the optimal number of processes or threads to minimize execution time and to see if there is a limit to the number of processes/threads. If a limit is found, there is no need to have more overhead with multiple processes/threads for a fraction of a second in improved efficiency. The programs were compiled and run on the SVS OS class docker. For multiple processes, the command ./mandelmovie was used to call 50 instances of the mandel.c program. The mandel.c program was modified to add multithreading and color schemes to the patterns found. Mandel can be run with multiple flag combinations. My pattern was created using the command: `./mandel -x -0.57 -y -0.5 -s 0.00015 -m 200 -W 2000 -H 2000`. Multithreading was achieved with the -n flag:

```
./mandel -x 0.2869325 -y 0.0142905 -s .000001 -W 1024 -H 1024 -m 1000 -n 12 -o test
```

Section 2: Multiple Processes mandelmovie Analysis

Measure and graph the execution time of mandelmovie for each of 1, 2, 3, 4, 6, and 12 processes running simultaneously. Because each of these will be fairly long running, it will be sufficient to measure each configuration only once.

Processes Running	Execution Time (s)
1	139.7015045
2	76.91539645
3	58.93387485
4	50.81170082
6	48.17397761
12	45.53066587

Table 1. Processes vs Execution Time

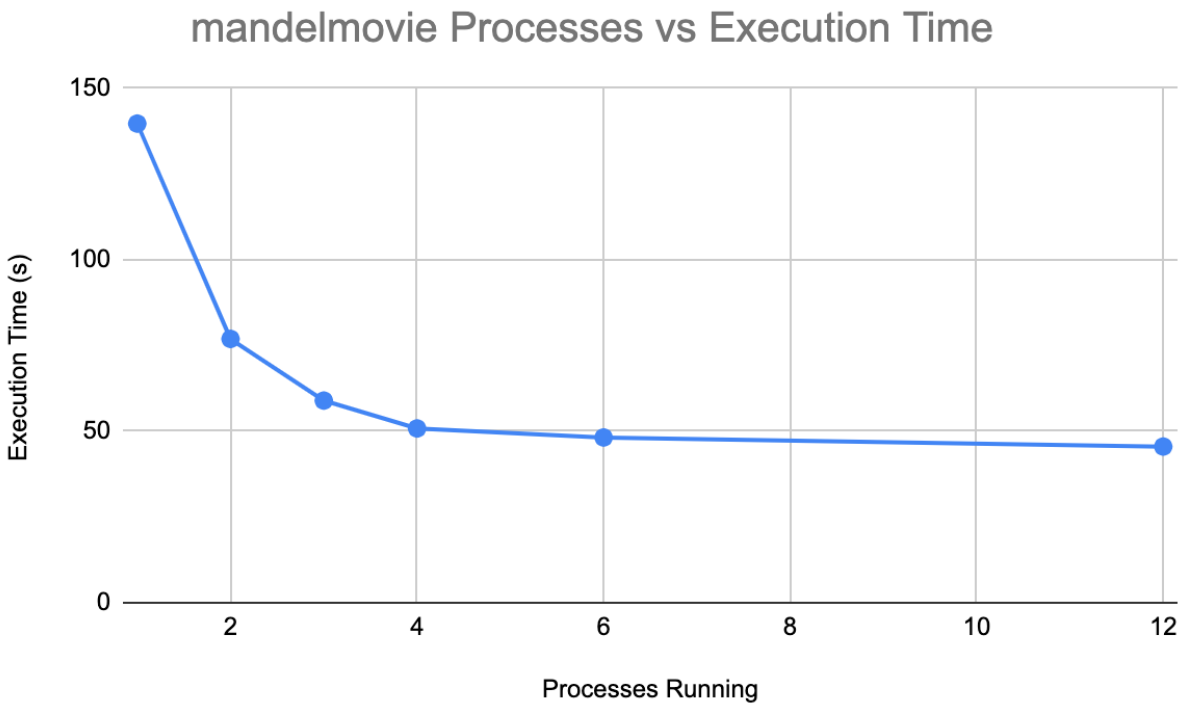


Figure 1. Graph of Processes vs Execution Time

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

The multi-process execution followed a logarithmic curve. As the process number increases, the execution time decreases making the program finish faster. However, after about 4 processes, the execution time levels out. This limit at 4 processes means it would not be efficient to increase the process amount after this; more overhead for minimal improvement in execution time.

Section 3: Multiple Threads

For the following two configurations, measure and graph the execution time of multithreaded mandel using 1, 2, 4, 6, 12, 18, and 24 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: mandel -x -.5 -y .5 -s 1 -m 2000
B: mandel -x 0.2869325 -y 0.0142905 -s .000001 -W 1024 -H 1024 -m 1000

Threads	Case A	Case B
1	1.271593094	3.028554678
2	0.7600178719	1.789578199
4	0.687939167	1.188545942
6	0.5007810593	1.179382086
12	0.450936079	1.190354824
18	0.4301526546	1.173285723
24	0.4400920868	1.169609785

Table 2. Case A vs Case B

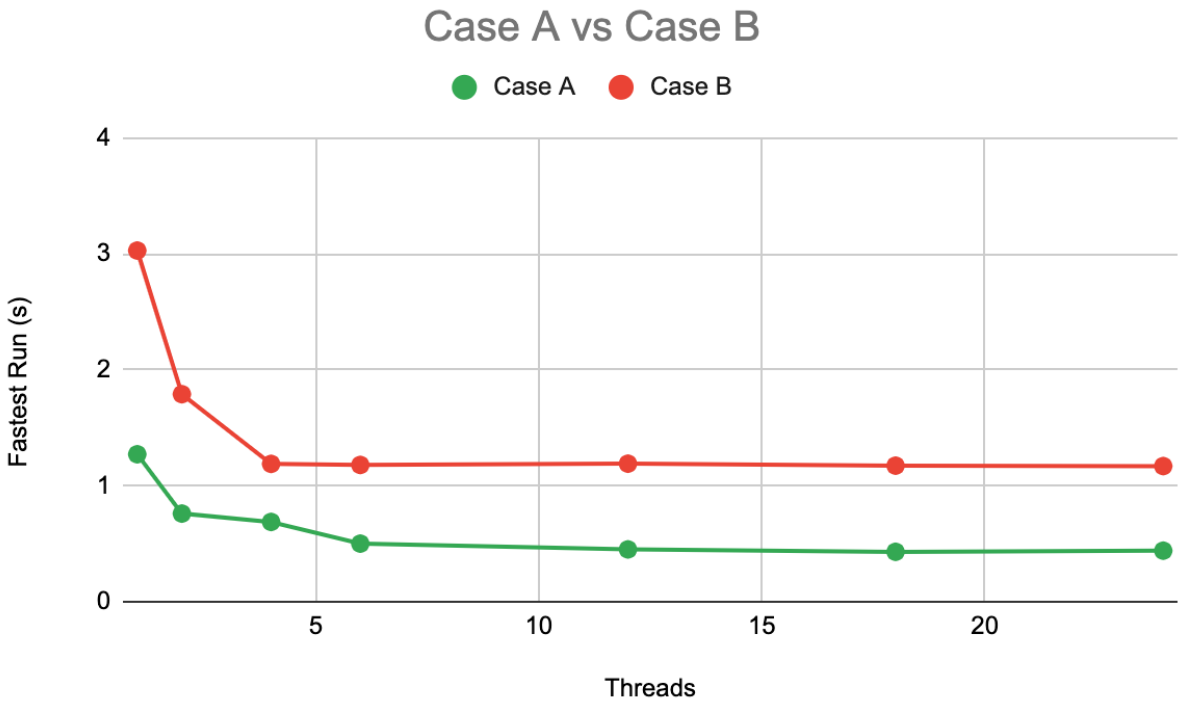


Figure 2. Case A vs Case B Graph

Threads	Fastest Run (s)
1	1.271593094
2	0.7600178719
4	0.687939167
6	0.5007810593
12	0.450936079
18	0.4301526546
24	0.4400920868

Table 3. Multithread Case A

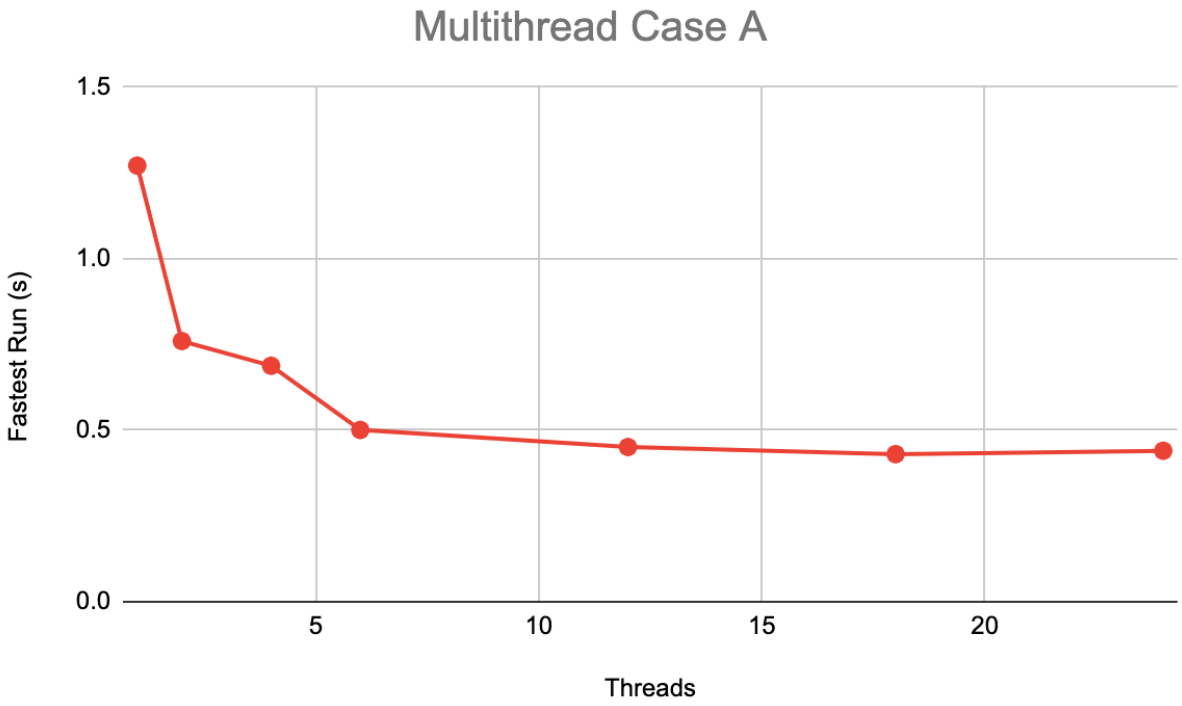
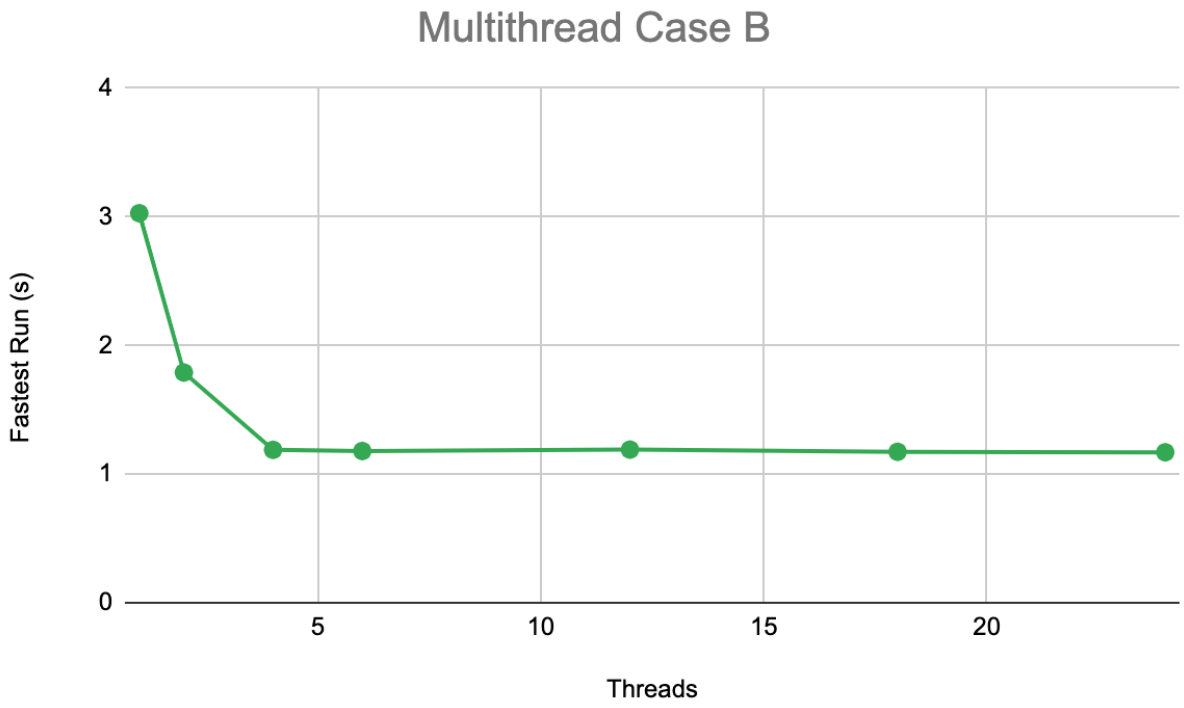


Figure 3. Multithread Case A Graph

Threads	Fastest Run (s)
1	3.028554678
2	1.789578199
4	1.188545942
6	1.179382086
12	1.190354824
18	1.173285723
24	1.169609785

Table 4. Multithread Case B



Graph 4. Multithread Case B

Explain the shape of the two curves. What is the optimal number of threads? Why do curves A and B have a different shape?

Both of the two curves follow a logarithmic curve like the processes graph. The optimal thread number is again around 4. After this point, the graph levels off. Curves A and B have a different shape because curve B has to do more processing per thread. There are more specifications in the command line call that delay the execution of the program. Case B, however, experiences a sharper decrease during the initial threading, which suggests that the threads benefit case B more than case A. This outcome could be because there is more overhead and computation needed for case B, so splitting this into threads helped the program execute faster.