Cole Pickford
2/13/20

Lab 3 Report

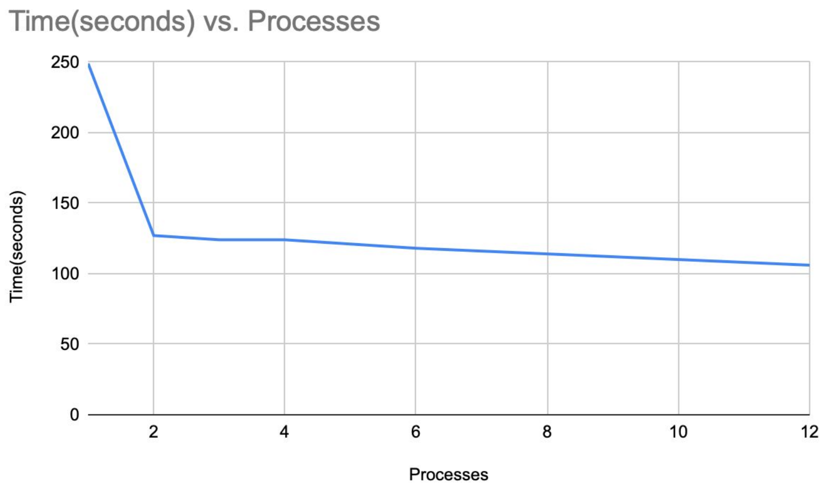
Machine: Docker Container
Command line arguments: ./mandel -x -0.5024 -y -0.603 -s 0.0001 -m 1000 -H 1024 -W 1300

The purpose of this experiment is for us to get more practice writing code for multiprocessing and multithreading in order to increase the efficiency of our programs. It also shows us the diminishing returns associated with adding more threads and processes so that we also avoid inefficient code. There is even a little bit of comparison between processes and threads in this experiment though it isn't as definitive because the commands are different.

Mandelmovie:

- Command: time ./mandelmovie <number of processes>

Processes	Time(seconds)
1	249
2	127
3	124
4	124
6	118
12	106

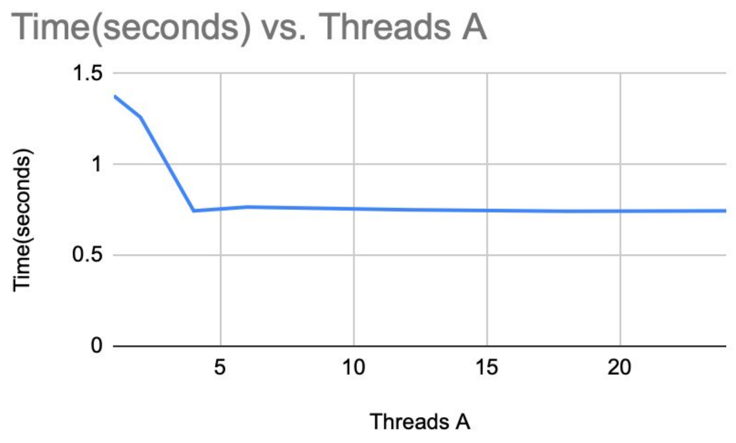


The shape of the curve is a steep dropoff in the beginning to a gradual decline. This is because of the diminishing returns associated with running multiple processes. The optimal number of processes is 2 because you are much faster than one, but you are not using as much of the CPU. As soon as it starts to flatten out it is less efficient to add processes because there isn't a significant change in the times. It is possible to have too many processes because eventually there will be so many that the time associated with contact switching will reduce efficiency and once again grow the execution time.

Mandel:

- A (mandel -x -.5 -y .5 -s 1 -m 2000):

Threads A	Time(seconds)
1	1.375
2	1.257
4	0.742
6	0.763
12	0.748
18	0.74
24	0.742

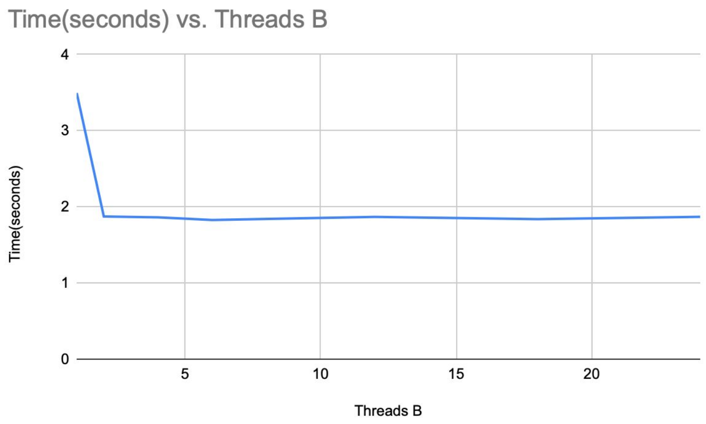


The curve shows that there is only a slight difference in time between one and two threads, but there is a significant drop in time when there are four threads. Afterwards, the added threads don't have much affect on time reduction and the curve levels off. The optimal number of threads for command A is four threads because it is faster and leaves more memory available. It becomes less efficient to add more threads..

- B (mandel -x 0.2869325 -y 0.0142905 -s .000001 -W 1024 -H 1024 -m 1000):

Threads B	Time(seconds)
1	3.495

2	1.873
4	1.862
6	1.827
12	1.868
18	1.838
24	1.869



The shape of this curve shows a significant drop in time between one thread and two threads. Afterwards, the added threads don't have much affect on time reduction and the curve levels off. The optimal number of threads is two threads because it is faster and leaves more memory available. It becomes less efficient to add more threads.

The difference in the shape between the two graphs is caused by the size and locations of the images they are generating. B has a much smaller scale and much larger image so it is going to be slower than A.