EVALUATION REPORT

INTRODUCTION

This experiment is to briefly demonstrate how a single process can be run in two different ways. One way is by creating multiple process and the other way is by creating multiple threads for a single process.

PART I

The program `mandelseries.c` generates 50 images using the function `fork()` while varying the value of scale down to a desired value to get a desired image. The command line argument for this is given as an integer varying from 2 to 50 which represents maximum number of process to be created at the same time. If there is no command line argument given, the program assumes the default value 1. The purpose of this program is to generate multiple images using multiple processes.

PART II

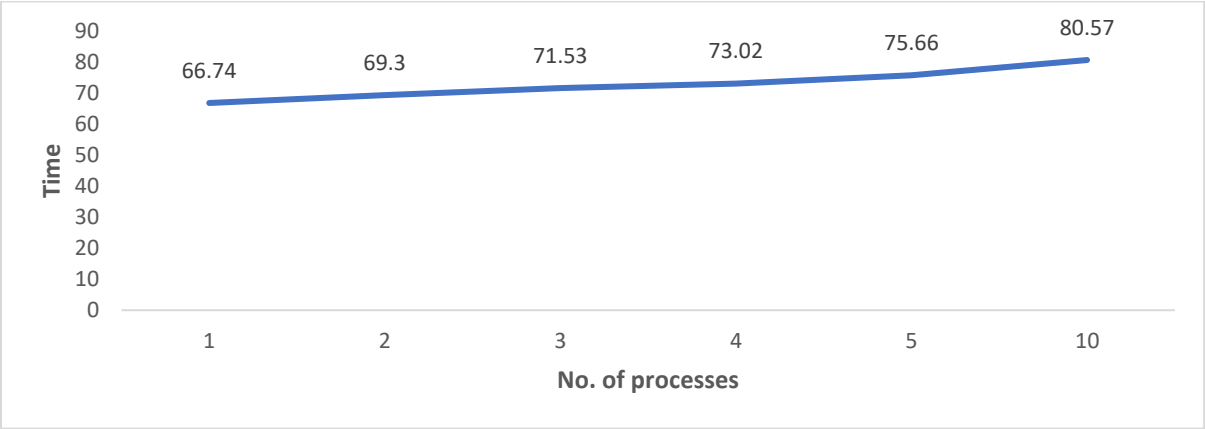
The program `mandel.c` generates a single image by the use threads which is created by the function `pthread_create()`. The command line argument for this is given as `-n` which represents the number of threads to be created to create a certain section of image. The value of `-n` varies from 1 to 50. If the value of the argument is 2, then `thread0` computes image from pixels 0 to 250. While `thread1` computes image from pixel 251 to 500 for image that is 500 pixels high. The threads are joined using the function `pthread_join()` function. The purpose of this program is to create a single image using multiple threads.

MULTIPLE PROCESS

The `mandelseries.c` program is run for different values of argument. The time taken for the multiple process to complete is noted and tabled as,

NUMBER OF PROCESS TO RUN AT SAME TIME (n)	TIME CONSUMED (Sec)
1	66.74
2	69.30
3	71.53
4	73.02
5	75.66
10	80.57

The graph is given as,



The time taken increases for very increase in number of process to start at the same time. It is because with each process created must wait for the previous process to complete. So, creating more than one process at a single time, makes the next process wait for a while till the previous process is completed. The optimal number of processes that can be created would be 5 since running more than increases the time consumed by the process steeply. It is not possible to too many processes because the time consumed for the nth process to wait for the previous processes to complete might result in waste of memory allocated for that process to wait.

MULTIPLE THREADS

The `mandel.c` program is run for different values of argument. The program is repeated for five times. The time taken for the multiple threads to complete is computed and tabled as,

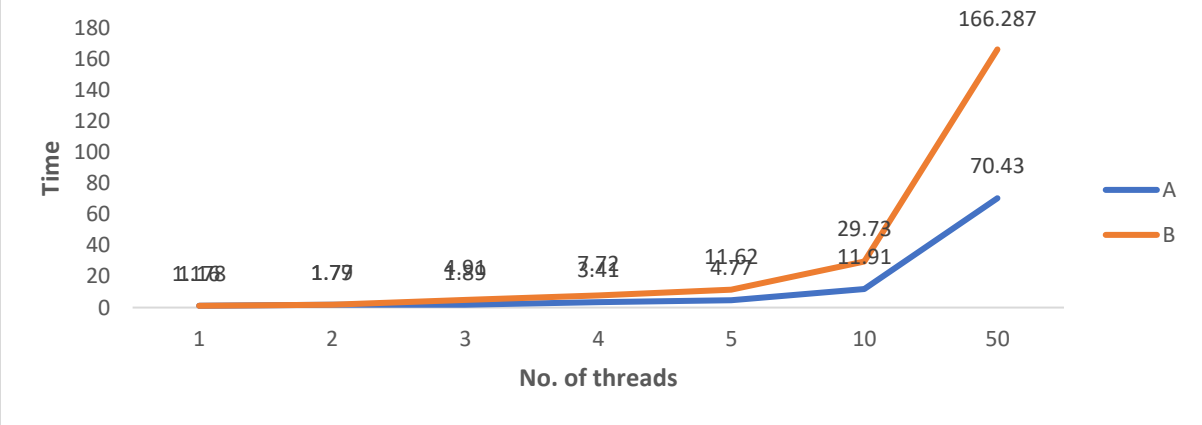
A:

NUMBER OF THREADS TO RUN AT SAME TIME (n)	TIME CONSUMED (Sec)
1	1.178
2	1.79
3	1.89
4	3.41
5	4.77
10	11.91
50	70.43

B:

NUMBER OF THREADS TO RUN AT SAME TIME (n)	TIME CONSUMED (Sec)
1	1.16
2	1.77
3	4.91
4	7.72
5	11.62
10	29.73
50	166.287

The graph is given as,



The time taken increases for very increase in number of threads to start at the same time. It is because with each thread created the time is consumed in joining the thread. The optimal number of threads that can be created would be 2 since running more than 2 increases the time consumed by the threads steeply. It is not possible to too many threads because too many threads might give way to race condition and synchronization of threads needs to be implemented so that all the threads join properly. The curves A and B differ in shape because of the change in the height and width of the input.