Berk Saltuk Yılmaz - 21903419 - Sec01

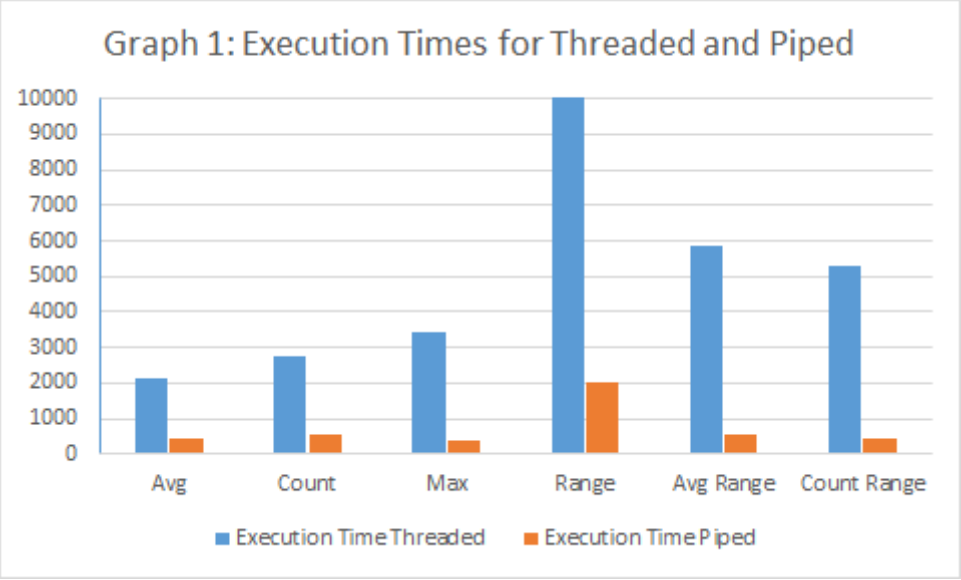
CS342 Operating Systems - Project 1 Report

C) I have calculated the execution times of threads and child processes using the clock() method.

In my first experiment, I have calculated how much it takes to execute each command with threads and with child processes when the set of input files are the same. I have created graphs and tables for this experiment.

N = 3, file size = 10000	Execution time for threaded program (microseconds)	Execution time for piped program (microseconds)
Avg	2106.00	458.00
Count	2769.00	550.00
Max	3442.00	410.00
Range 2000 5000 150	485989.00	2008.00
Avg 2000 5000	5865.00	532.00
Count 2000 5000	5300.00	447.00

Table 1: Different Commands' Execution Times for Threaded vs Piped Programs

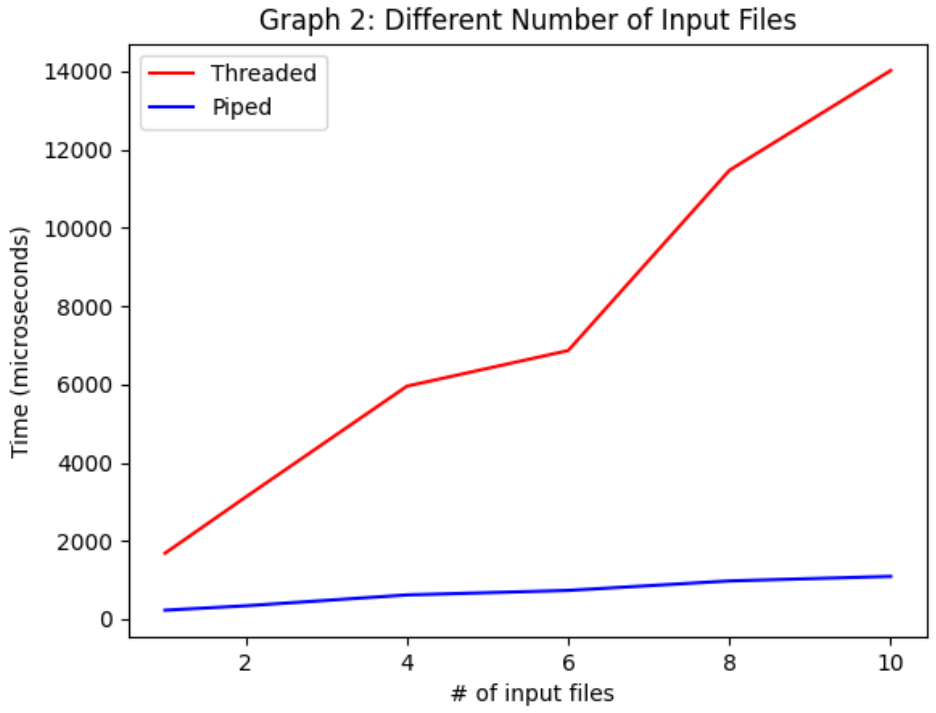


In my second experiment, I have calculated how much time it takes to execute the same command while changing the number of input files with the same input file (and of course

varying the number of threads and child processes). I have created graphs and tables for this experiment. For this experiment I have kept the input size as 10000 integers and I have executed the avg command.

Input file number	Execution time for threaded program (microseconds)	Execution time for piped program (microseconds)
1	1685.00	229.00
2	3120.00	341.00
4	5953.00	619.00
6	6863.00	734.00
8	11470.00	977.00
10	14019.00	1093.00

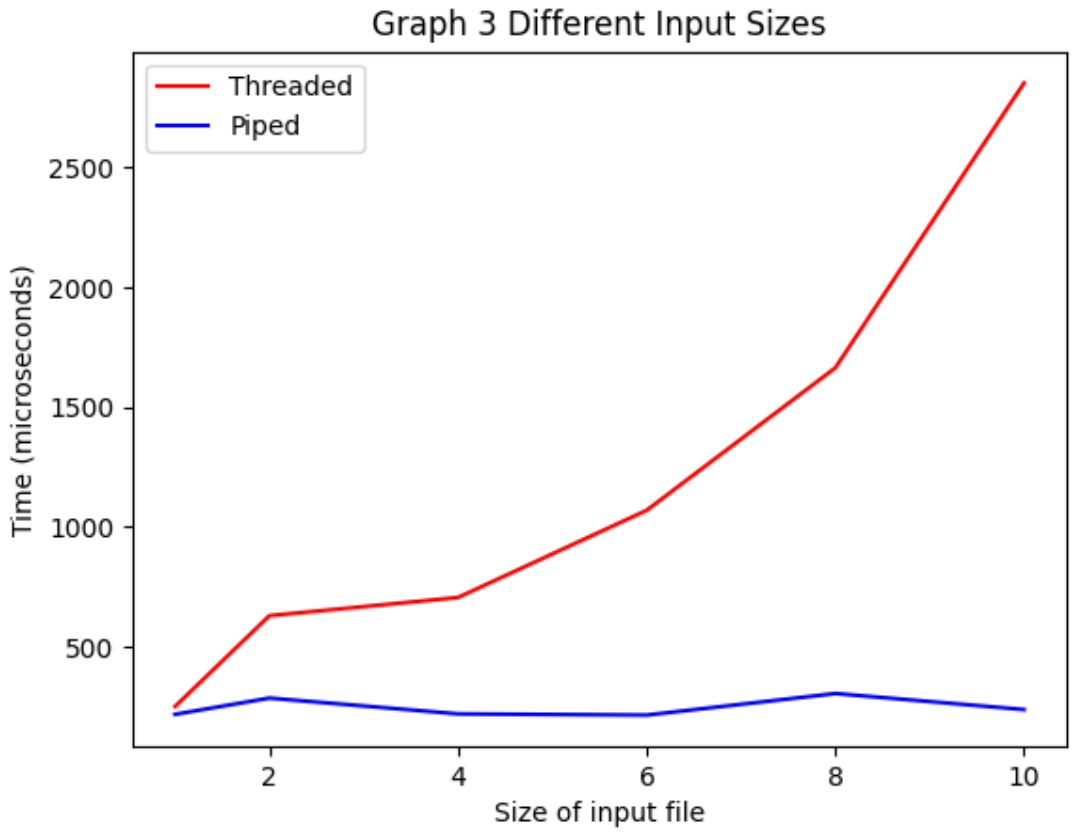
Table 2: Different Number of input files and execution times



In my final experiment, for the same number of threads and processes I have increased the length of input files (e.g. 100 to 20000 integers in a file). I have created graphs and tables for this experiment. For this experiment I have set N = 1 and executed the max command.

Input set size	Execution time for threaded program (microseconds)	Execution time for piped program (microseconds)
100	251.00	218.00
1000	630.00	286.00
2000	706.00	220.00
5000	1070.00	215.00
10000	1665.00	305.00
20000	2854.00	238.00

Table 3: Different size of input files with same number of thread or children



Conclusions:

I can conclude that for both solutions, execution times are increasing as the thread/child number increases as expected since the overhead is increasing with the new pipes and a new AS is created for each thread. Moreover, from the first experiment it can be seen that for a relatively large data set (10000 in this case), the multithreaded program is being executed much slower, approximately ten times longer. I was expecting the piped program would take longer to execute since pipes are creating an overhead. However, this result might be about the linked list implementation I have used in the threaded program which takes a lot of time while adding an item and retrieving items. Moreover, in the piped program I have directly sent the children's results with pipes to the parent and there was no overhead due to the linked list etc. In the final experiment it can be seen that for short input data sets (<1000) piped and threaded programs have similar execution times. However, there is approximately linear increase in threaded programs as the data set gets larger, and the piped program is not that affected by the increasing data set size. I believe that the same reasons are valid for the third experiment.