CS342
Operating Systems
Fall 2022

Project #1
Processes, IPC, and Threads

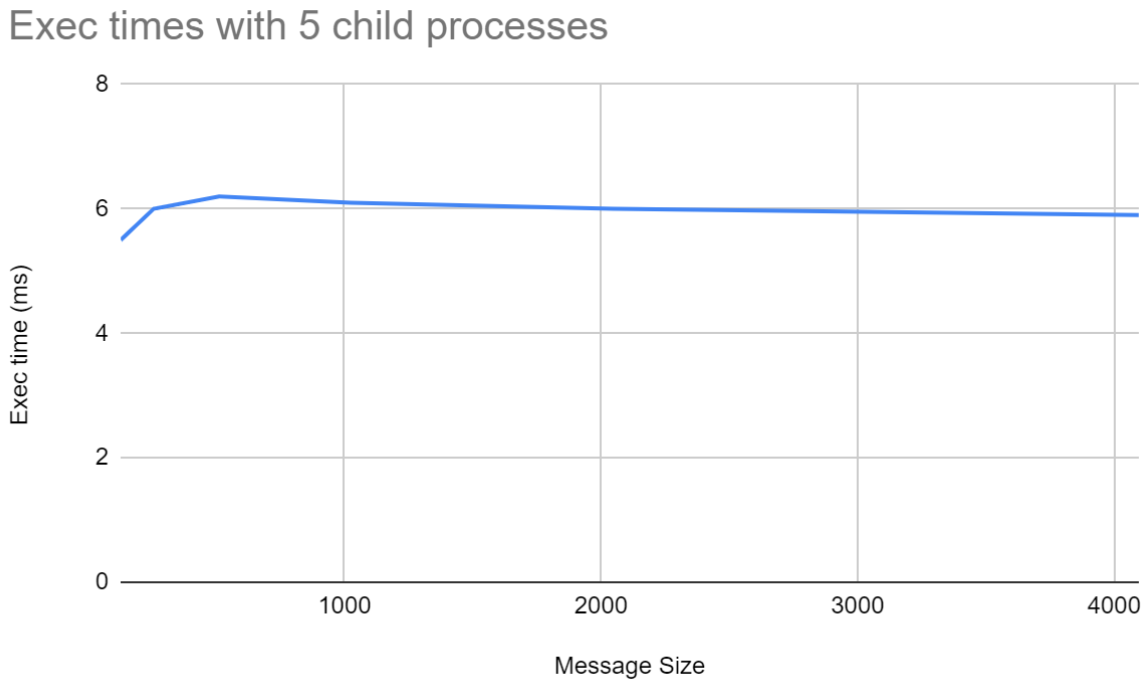
Arda Önal 21903350
Eren Polat 21902615

Experiments

In the experiments conducted, word count has been fixed and words of similar length have been used for all files to create a controlled setup. It is made sure that threads/child processes are fed with a similar amount (char count) of data. We have experimented with different number of processes/threads (N=1,2…5) by fixing message size, and with different message sizes but fixing N. We have also taken records of each sample for the same value multiples of times, as the results are not deterministic; average of multiple tries are taken to measure the average exec time (ms). Max number of messages is also fixed for a message queue used to communicate with children processes.

Processes

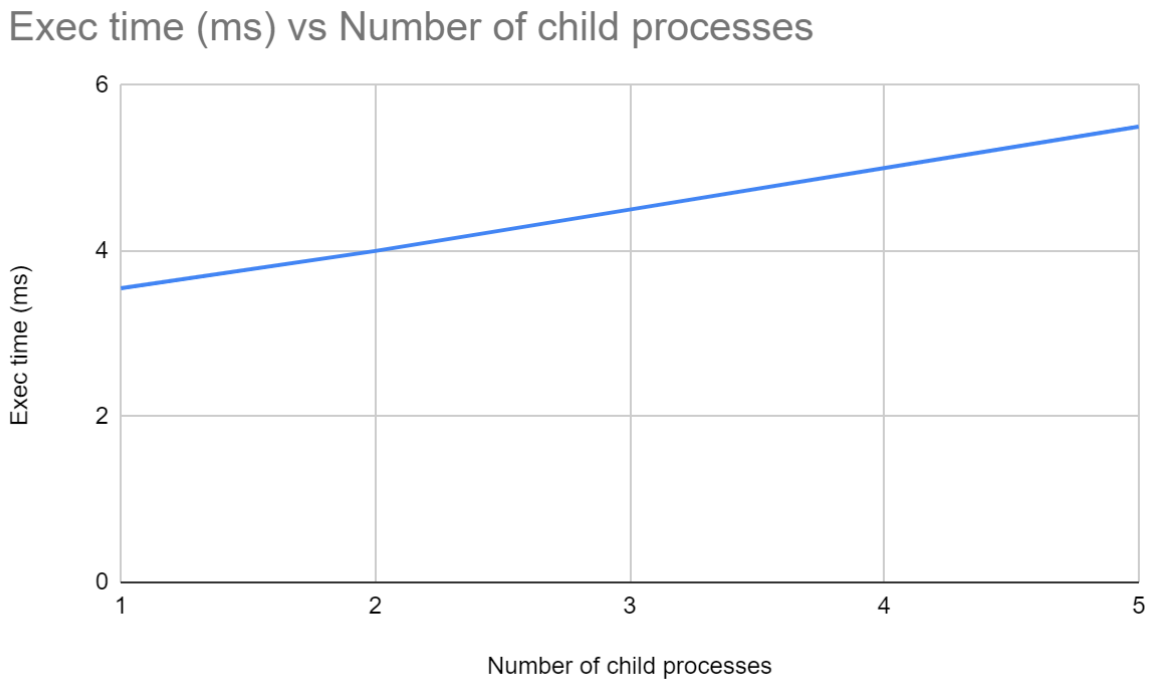
Message Size (bytes)	Exec time (ms)
128	5.5
256	6
512	6.2
1024	6.1
2048	6
4096	5.9



Notice that the change in message size (the size of the message sent over the message queue) remains around the same for 128, 256, 512, 1024, 2048, 4096 bytes. This is because the amount of work done for different message sizes remains the same, that we read the same amount of words, and the runtime operations for counting the frequencies for it remain the same. Number of child processes is fixed to N = 5 for this, but we have also tried it with other N's and achieved similar results, that message size doesn't change the exec time for fixed N.

When message size is 128

Number of child processes	Exec time (ms)
1	3.55
2	4
3	4.5
4	5
5	5.5

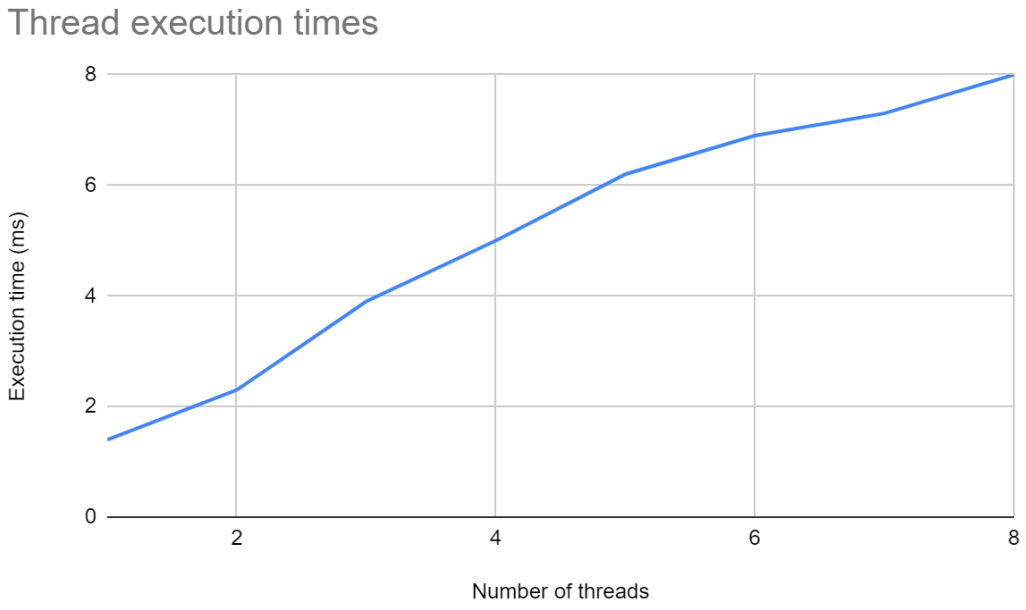


We have fixed message size to 128 bytes and measured the execution time for multiple values of number of child processes (N). The average of multiple tries are taken for different values of N. Observe that the execution time increases as we increase the number of child processes. This is because although processes may seem parallel, they are actually executed in a single CPU sequentially, one at a time. Therefore, we gain no time in creating more processes to read/find frequencies unless we have multiple CPU's. The results we

have achieved show a linear rise of exec time with respect to N, as the number of child processes increase, the number of files they read increases, and the amount of work that should be done in the CPU increases since we have **more data** to process (the number of words read/found frequencies increases). Also, context switches in processes are expensive, contributing to why exec time increases as we increase N.

Thread

Number of Threads	Execution time (ms)
1	1.397
2	2.3
3	3.9
4	5
5	6.2
6	6.9
7	7.3
8	8



For the same reason exec time increases as we increase the number of child processes, exec time also increases as the number of threads increases. Since there are N input files each processed by N threads, the workload increases with more no. of threads. Furthermore, since there is only one CPU and only one thread's instructions are executed sequentially at each burst, the total time it takes for all words to be processed increases. Again since there is no parallelization of more than one GPU, we don't gain time as we increase the number of threads.

Notice that threads' exec time is less than processes in general for the same number of input files (N). This is because context switches between threads is faster, creation time of a thread is faster, fewer resources are used, and the ability to use shared memory because all threads are essentially segments of the same process. With child processes, they need to use IPC methods, in this case they send a message to a message queue, which causes more time to be spent in the execution. Context switches between different processes are also more expensive than the ones in threads. For these reasons, execution with threads with same N was recorded to be faster