CS-342 Operating Systems
Project-1 Report

Group Members:

- Uğur Can Altun / 22002701 / Section-02
- Utku Boran Torun / 21901898 / Section-02

Instructor: İbrahim Körpeoğlu

Part C - Experiments

Section 1 (Various N Values with Fixed K)

Proctopk

Test 1
For this test N = 2 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 408 milliseconds

Test 2
For this test N = 4 | K = 300| Number Of Words In File = 250000
The time that takes the program to execute = 695 milliseconds

Test 3
For this test N = 6 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 1120 milliseconds

Test 4
For this test N = 8 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 1571 milliseconds

Test 5
For this test N = 10 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 2356 milliseconds

Threadtopk

Test 1
For this test N = 2 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 428 milliseconds

Test 2
For this test N = 4 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 734 milliseconds

Test 3
For this test N = 6 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 1188 milliseconds

Test 4
For this test N = 8 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 1553 milliseconds

Test 5
For this test N = 10 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 1917 milliseconds

Graph Of Results

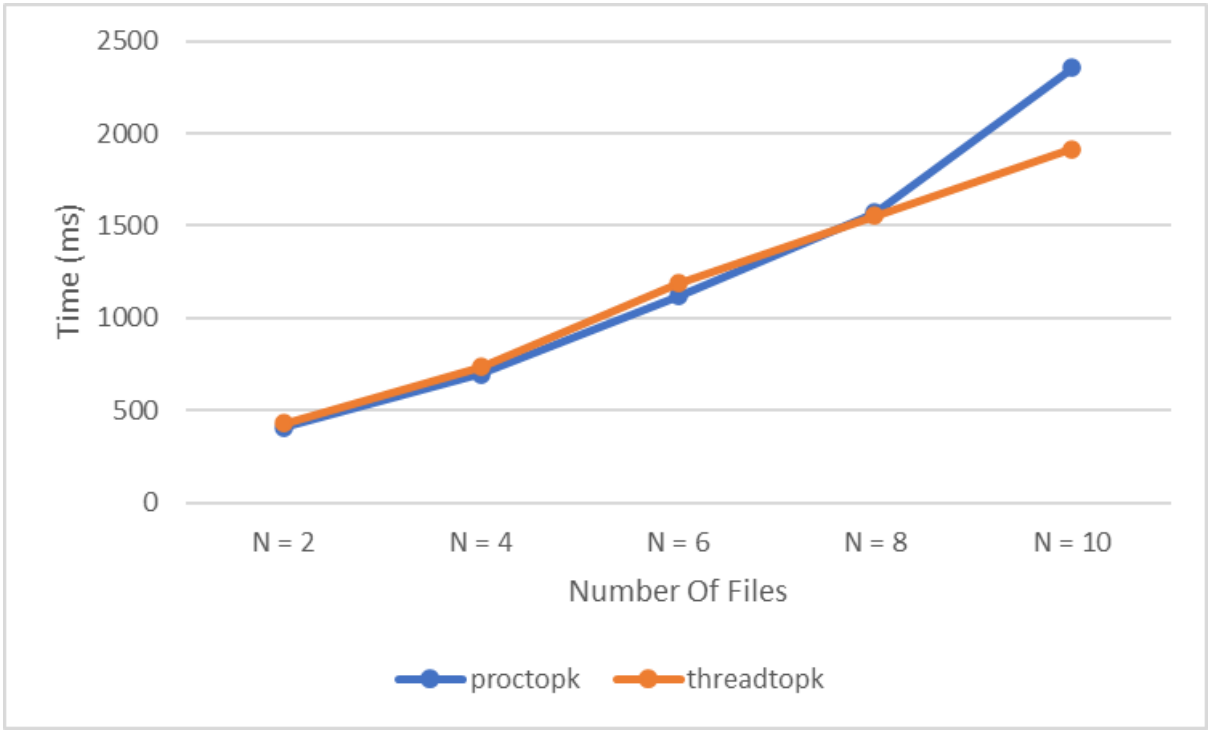

***Figure 1**: K fixed Proctopk and Threadtopk*

Section 2 (Various K Values with Fixed N)

Proctopk

Test 1
For this test N = 10 | K = 50 | Number Of Words In File = 250000
The time that takes the program to execute = 2365 milliseconds

Test 2
For this test N = 10 | K = 80 | Number Of Words In File = 250000
The time that takes the program to execute = 2462 milliseconds

Test 3
For this test N = 10 | K = 120 | Number Of Words In File = 250000
The time that takes the program to execute = 2491 milliseconds

Test 4
For this test N = 10 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 2641 milliseconds

Test 5
For this test N = 10 | K = 1000 | Number Of Words In File = 250000
The time that takes the program to execute = 2963 milliseconds

Threadtopk

Test 1
For this test N = 10 | K = 50 | Number Of Words In File = 250000
The time that takes the program to execute = 1937 milliseconds

Test 2
For this test N = 10 | K = 80 | Number Of Words In File = 250000
The time that takes the program to execute = 2391 milliseconds

Test 3
For this test N = 10 | K = 120 | Number Of Words In File = 250000
The time that takes the program to execute = 2577 milliseconds

Test 4
For this test N = 10 | K = 300 | Number Of Words In File = 250000
The time that takes the program to execute = 2615 milliseconds

Test 5
For this test N = 10 | K = 1000 | Number Of Words In File = 250000
The time that takes the program to execute = 2689 milliseconds

Graph Of Results

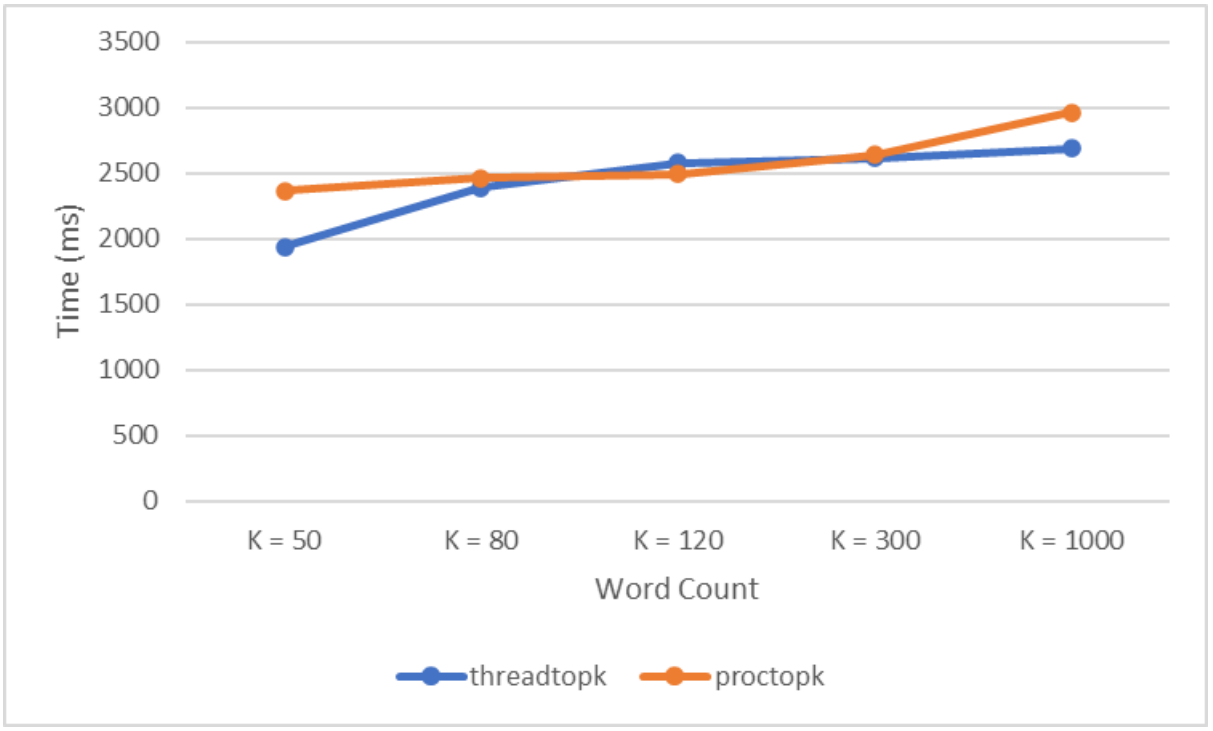

***Figure 2**: N fixed Proctopk and Threadtopk*

Discussion Of Results

Section 1 (Various N Values with Fixed K)

For the first section of the experiment, we fixed K to 300 and incremented N by 2 up to 10. The number of words in each file was 250000. After doing these we measured the execution time of the programs proctopk.c and threadtopk.c. Theoretically, processes are slower than threads as they have isolated memory and threads share the memory with their parent processes and other threads within the process. Although the program uses shared memory for inter-communication between a parent process and child processes, it can be said that it would take longer to transfer data than the threads approach as threads use the exact same memory as it is specified above. If we have 10 files to be scanned, the program creates 10 different threads that run concurrently. In the case of processes, a new memory location is allocated for child processes that initially contain the same values and size as the parent process and it is more costly compared to the threads approach. Furthermore, multiple processes cannot run concurrently based on the number of cores. As a result of this, processes take more time than threads as we can observe from the graph (See Figure 1). Initially, both threads and processes approach approximately the same amount to execute due to relatively few file numbers and less k value as the processing power of the computer/virtual machine is too powerful compared to the process executed. After a while, as it can be observed from the graph, proctopk execution time surpasses threadtopk execution time when N is greater than 8 due to the virtual machine's core usage. The number of cores available to the processes starts to be insufficient for the program after that point and the advantages of threads described above can be observed at that part of the graph.

Section 2 (Various K Values with Fixed N)

For the second section of the experiment, we fixed N to 10 (which is the maximum) and used various K that varied from 50 to 1000 (which is the maximum). As the K value is not correlated with the thread or child process creation but correlates with the amount of memory that is allocated for a data structure that will store the results of the threads/child processes and can be accessed from all components of the program, it is not expected to cause any significant difference between the execution time of the threadtopk and proctopk. When we increased K, there wasn't too much difference between execution times. This shows that the increase of K does not affect execution time compared to the effect of the increase of N on execution times (See Figure 2). Generally, threadtopk execution time is relatively smaller than proctopk execution time because threads run concurrently and it is less costly to create threads than child processes. Therefore, it can be observed that threadtopk generally has less execution time than proctopk throughout the graph as a fixed amount of child processes or thread, which is N, are created throughout the experiment.