2022-2023 SPRING

CS 342 - OPERATING SYSTEMS PROJECT 01

Processes, IPC, and Threads

21.03.2023

Group Members:

Emre Karataş - 22001641

İpek Öztaş - 22003250

1.EXPERIMENTAL RESULTS

1.1 proctopk.c

1.1.1 *Case 1:*

K = 5;
N = 1,2,3,4,5;

N values	1	2	3	4	5
Real	0m0,008 s	0m0,009 s	0m0,024 s	0m0,176 s	0m0,261 s
User	0m0,001 s	0m0,005 s	0m0,003 s	0m0,005 s	0m0,005 s
sys	0m0,003 s	0m0,000 s	0m0,002 s	0m0,003 s	0m0,002 s

Table 1

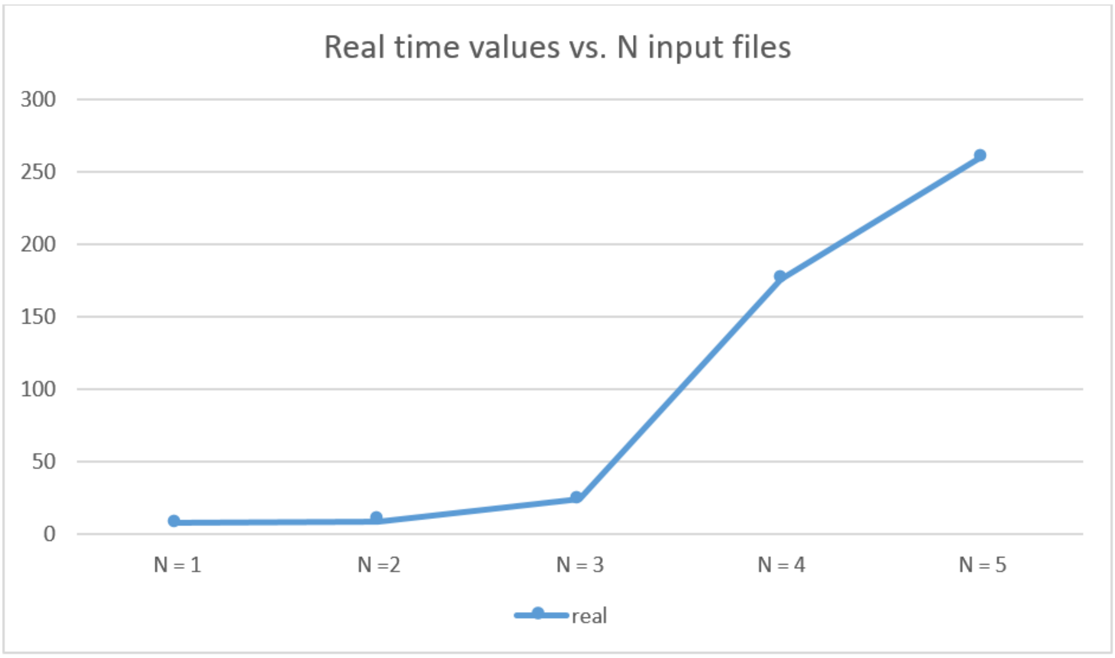

Figure 1

Figure 1 shows the real time values depending on the different number of input files. There is not a significant change when N goes up from 1 to 3. However, when N value increases to 4 from 3, the jump in real time value is higher. This is caused because the virtual machine that we used has three physical cores.

1.1.2 *Case 2:*
K = 5,10,15,20,25;
N = 2;

K values	5	10	15	20	25
Real	0m0,030 s	0m0,273 s	0m0,303 s	0m0,391 s	0m0,406 s
User	0m0,004 s	0m0,003 s	0m0,003 s	0m0,004 s	0m0,003 s
sys	0m0,002 s	0m0,002 s	0m0,005 s	0m0,004 s	0m0,006 s

Table 2

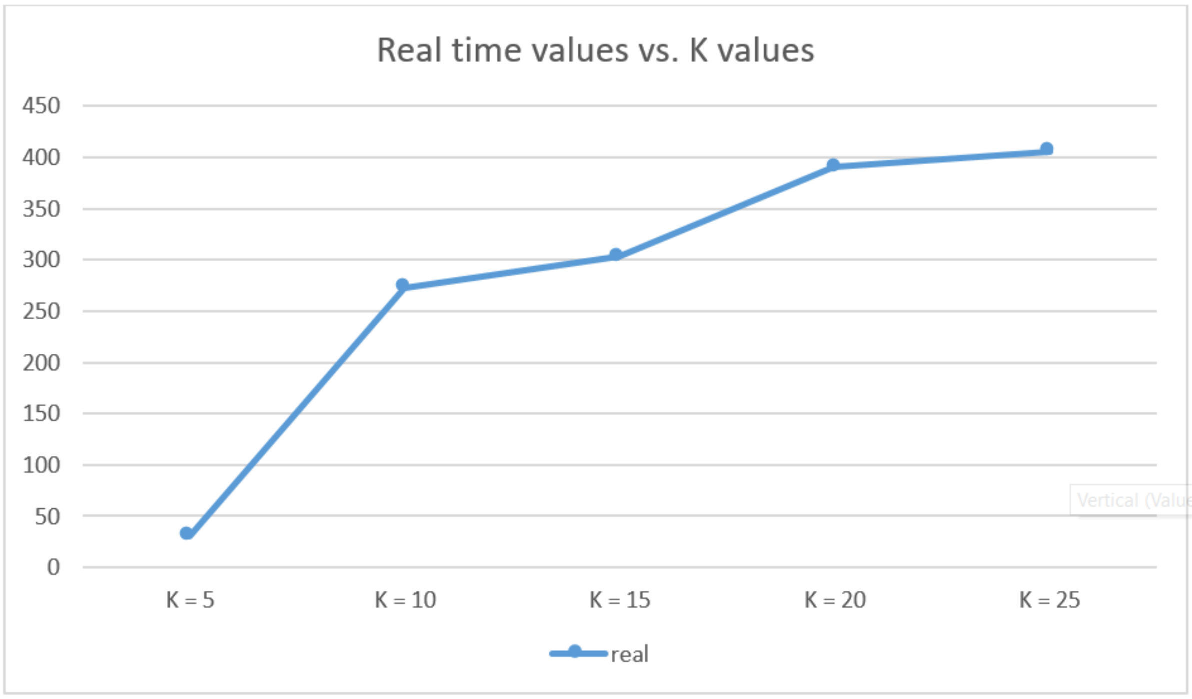

Figure 2

1.2 threadtopk.c

1.2.1 *Case 1:*

K = 5;
N = 1,2,3,4,5;

N values	1	2	3	4	5
Real	0m0,002 s	0m0,002 s	0m0,003 s	0m0,007 s	0m0,009 s
User	0m0,002 s	0m0,002 s	0m0,000 s	0m0,003 s	0m0,000 s
sys	0m0,000 s	0m0,000 s	0m0,003 s	0m0,000 s	0m0,005 s

Table 3

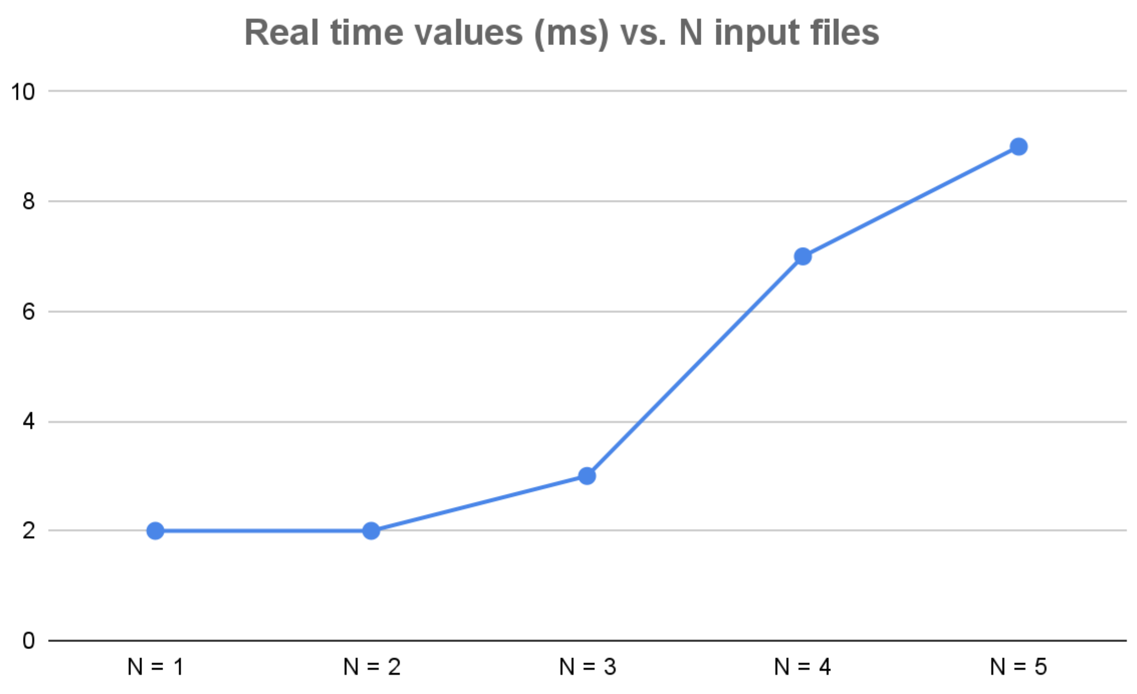

Figure 3

Figure 3 shows the real time values depending on the different number of input files. There is not a significant change when N goes up from 1 to 3. However, when N value increases to 4 from 3, the jump in real time value is higher. This is caused because the virtual machine that we used has three physical cores.

1.2.2 *Case 2:*

K = 5,10,15,20,25;
N = 2;

K values	5	10	15	20	25
Real	0m0,003 s	0m0,002 s	0m0,002 s	0m0,002 s	0m0,002 s
User	0m0,000 s	0m0,000 s	0m0,003 s	0m0,003 s	0m0,002 s
sys	0m0,003 s	0m0,003 s	0m0,000 s	0m0,000 s	0m0,000 s

Table 4

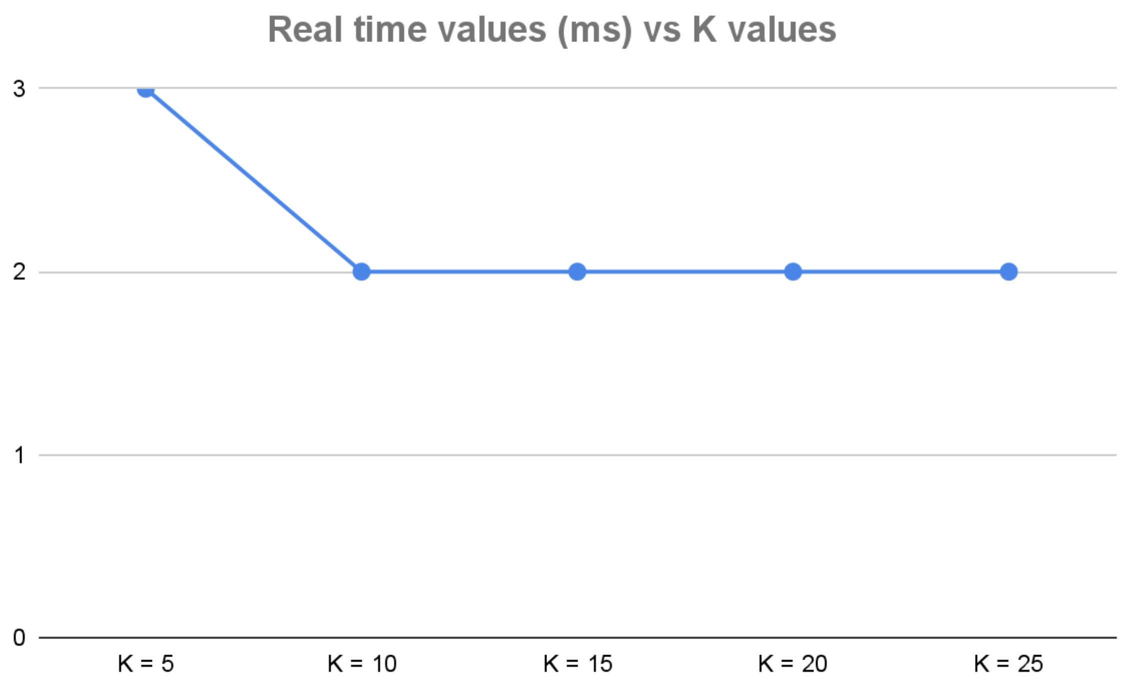

Figure 4

2. COMPARISONS AND COMMENTS

From the results in Figure 2 and Figure 4, it can be observed that the program using threads (threadtopk.c) completes faster than the program using child processes (proctopk.c). This indicates that the use of threads is more efficient in this particular scenario, counting top K words in the given input files.

One reason for this could be that threads are lighter weight than child processes, as they share the same memory space and do not require inter-process communication. This means that the overhead involved in creating and managing threads is lower compared to that of child processes. Additionally, context switching between threads is faster than between processes, as the former involves only a change in the program counter and stack pointer, while the latter requires saving and restoring the entire process state.

Another factor that may be contributing to the performance difference is the amount of data being processed. In the case of this program, the amount of data may not be large enough to take full advantage of the parallelism offered by child processes.With threads, the overhead of creating and synchronising threads is much lower, and the threads can be scheduled more efficiently, leading to better performance.

Overall, the results suggest that in this particular scenario, the use of threads is a better choice for achieving faster execution times. However, it is important to note that the choice between threads and processes depends on the specific requirements of the application, as each has its own strengths and weaknesses, and may be better suited for different use cases.