Bilkent University
Department of Computer Engineering

Operating Systems
CS 342
Project 1

Nisa Yılmaz - 22001849 - Section 1

Barış Yıldırım - 22003175 - Section 2

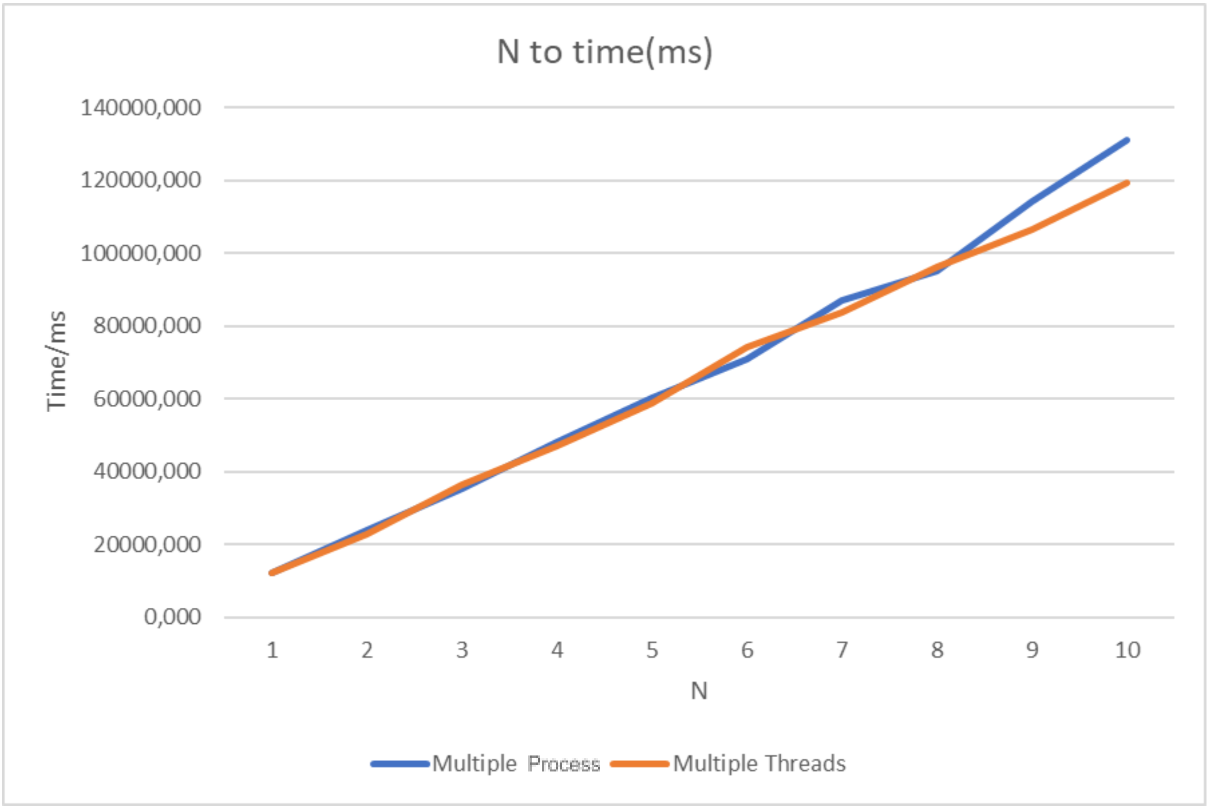

Fig.1 N to Time Graph for proctopk and threadtopk

To measure how the programs proctopk and threadtopk are affected by the number of input files we experimented with ten different n values and k fixed to 100 words. All the files contained 300000 words that were selected from a dictionary of size 10000. As a trivial result, we observed that as the value of n gets higher, both the programs took longer to execute and the increase in time was almost linear for both of them. Although both of the lines look very similar, we can see that the blue line is slightly above the orange line in the majority of the graph. This is because the blue line's (multiple processes) overhead is higher than the orange line's (multiple threads). In the program we have implemented, the child had to talk with the parent which required additional handling in the case of the multiprocess program. This was achieved by using shared memory. The creation of the shared memory, reading, and writing to it all caused additional overhead. Also, the code and variables are all copied into the child process when the fork() is called. Whereas in a multithreaded approach, both the data and the global variables are shared among all threads so the mentioned additional overhead is eliminated making the multithreaded approach more suitable for a program like this.

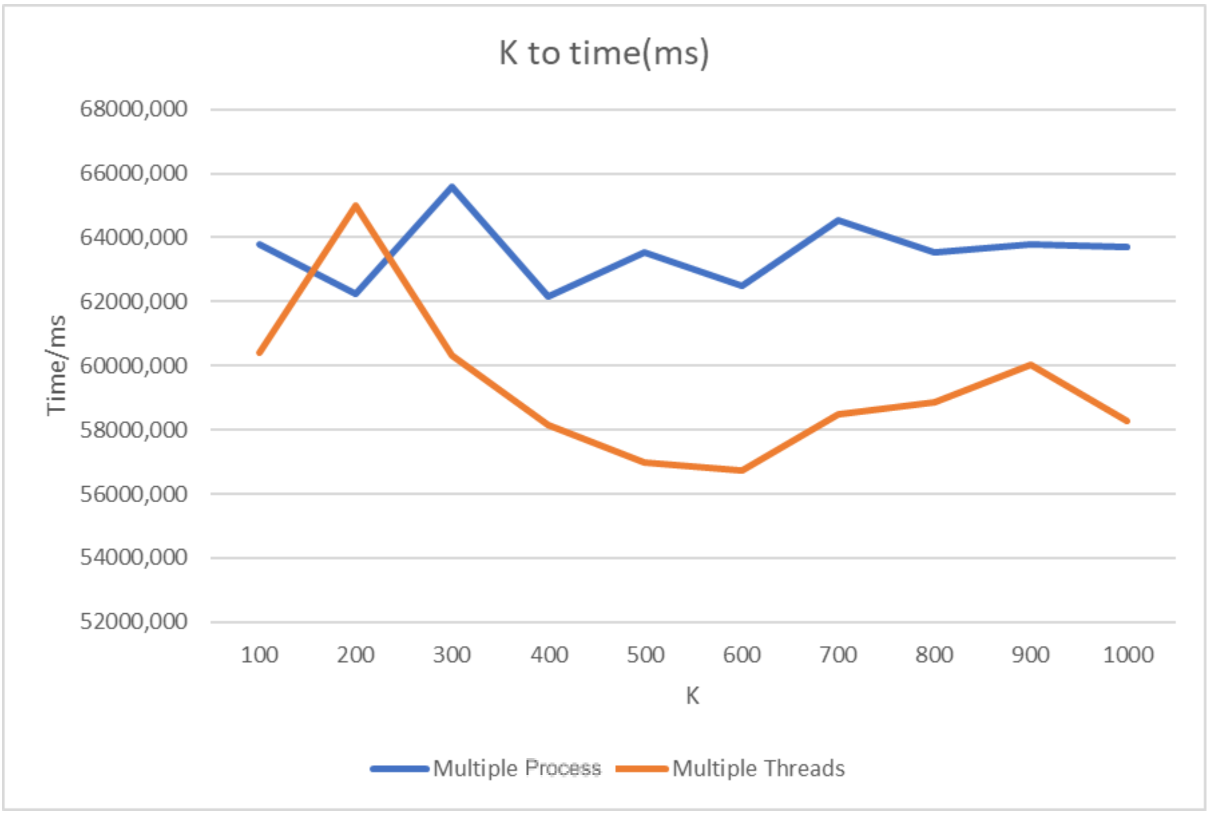

Fig.2 K to Time Graph for proctopk and threadtopk

In the second experiment, we fixed the value of n to five and gave k values between 100 and 1000 for both programs. When looking at the graph we can see that there is no uniform change in either of the graphs. For both of the lines, the movement is rather fluctuating movement. We believe that this is caused by the intervals we have given to the graph and looking at the graph in a larger interval, the lines seem more or less straight, except the peak of threadtopk. This peak at 200 words is a pattern that we could not evaluate. When we consider the average value for each program, it is clearly seen that threadtopk lasts less than proctopk. This again supports the points mentioned in the first experiment. Since values don't change directly proportional to k as they did to n, we can conclude that most of the overhead is caused by the creation and copying of a child process. Here because of this overhead, the overall values of the blue line are higher than the orange one which is implemented using threads.