Bilkent University

Department of Computer Engineering

CS 342 – Operating Systems

Project 2

04.04.2018

İrem Ural
21502278
Hakan Sarp Aydemir
21501331

Course Instructor: İbrahim Körpeoğlu

INTRODUCTION

In this project, interprocess communication is used via shared memory with server client architecture. During this procedure, as several clients try to access the server in order to send a request or to wait the results from the client, they use the shared memory segment. Due to the fact that shared memory segment can be accessed through several clients at the same time or client and server might want to access the shared data at the same time, they should all get the most recent updated result without any error. It requires the synchronization of processes therefore semaphores are used in order to prevent any misleading results and to synchronize the processes while accessing the shared memory segment. Additionally, in the project during the implementation of client-server design, for each request from the client a new thread is created and each request is handled by a single thread.

In this report timing experiments are done in order to see the effects of three variables:

1. The number of lines that contain the keyword in the input file
2. The number of clients
3. The number of lines in the input file to search the keyword

These three variables are tested as the first one effect the number of times the result queue semaphores are used by the server while sending the results to the client. The second is used as the number of threads created by the server will be changed, hence the parallelization will change. Last one will affect the execution time of the thread hence the elapsed time of the client as it will wait the result.

The execution times of the programs can change according to running environment. It is executed on virtual machine (with64 bit OS) with a 10 GB dynamically allocated memory. However it was in an external hard disk hence the execution times of the programs can be slower. The processes running at the background can affect the execution time as well.

Time Experiment 1:

In this experiment, the number of lines in the input file is kept same in this part of experimentation. It contains 1000 lines, each line containing 150 characters. The keyword is set to “ali123”. The number of client is changed, while keeping the number of lines containing the keyword same. Secondly the number of client is kept same while increasing the number of occurrences of keyword in the text file. The experiment is done for M(Number of lines that contain keyword) = {10, 100, 1000} , for N(number of clients) = {1, 5, 10 ,15}.

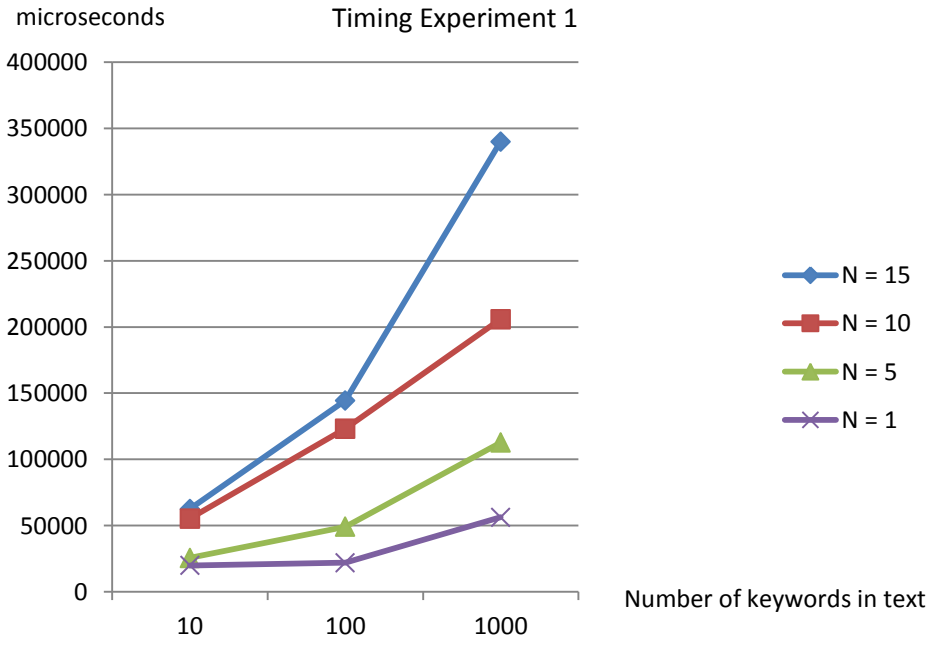
Table 1:

N = Number of clients , M = Number of Lines that Contain Keyword

N\M	10	100	1000
1	19914	21980	56285
5	25544	49156	112673
10	55243	123209	205863
15	62539	144494	340014

The results are in microseconds.

Graph 1:



Result and Discussions 1:

While testing the results, it is observed that when there are multiple clients each client's execution time varies. The maximum of the execution time of clients is written in the table to show the relation. However, this variability occurs as a result of the usage of shared memory. While accessing the shared memory or modifying it, this section of code can be defined as critical section. One thread should access the critical section to prevent inconsistencies. There shouldn't be concurrent accesses in that critical section. Hence to prevent it, semaphores are used. When several processes, clients, try to access the queue state to find an available index, they check the mutex semaphore first. If it is used by another client at the same time, the process is put into waiting state, it is blocked. Hence by these calls, the critical section is executed serially by one process at a time. When there is a signal, one by one they start to execute it. For the result queue the semaphores are used to synchronize the access of client and the server to the same result queue. The client should get the right written data by server. The semaphores for the request queue are used at the same way, to synchronize the access of client and server. The utilization of semaphore blocks when the value of the semaphore is 0, as a result of these, the execution time of the client changes.

When the number of keywords increase in the input file this results in more access to result queue by both server and client, as there will be more lines send by this shared memory segment from server to client. Server will execute signaling and waiting more as it will find more keywords; it will slow down the threads as this section is executed sequentially. Because of that, client will wait more until it gets all the elements and it will perform wait and signaling operations of semaphore more. Therefore, the system performance will slow down; it will take more time than before.

Lastly, as an observation when there 15 clients are active and the number of keywords are 1000, the server could not serve all of them some of them could not find a place in the queue state and it prints "too many clients started". This occurs as none of the clients couldn't finish its execution, before other one tries to access the queue state and searches for available location. In the previous cases, when there is less keyword, the client waits less therefore; it could finish its execution.

Time Experiment 2:

The number of keywords in the input text is kept constant, 100 keywords are placed in the txt file. Each line contains 150 characters and the keyword is set to “ali123” during this experiment. This time, the number of lines to search the keyword is incremented while keeping the client number fixed. Additionally, the effect of change in the number of client to the execution time is seen by keeping the number of lines stable. The experiment is done for M(Number of lines in the text) = {1000, 5000, 10000} , for N(number of clients) = {1, 5, 10 ,15}.

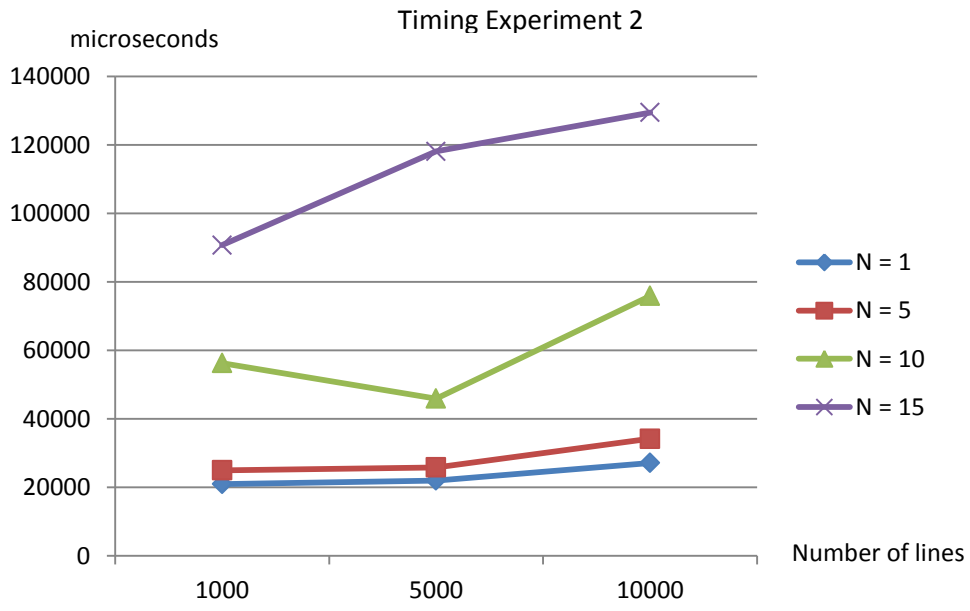
Table 2:

N = Number of clients, M = Number of lines in the text

N\M	1000	5000	10000
1	20989	22016	27125
5	25009	25833	34198
10	56272	45907	75899
15	90724	118062	129446

The results are in microseconds.

Graph 2:



Results and Discussions 2:

As the number of clients increment the execution time of client increments as well, though it does not increment as much, as there is parallelization provided by server by using threads. The client programs are running concurrently as well, however the CPU scheduling can cause differences between execution times. At the graph though the number of lines increment for 10 clients, in 5000 we can see a drop, the previous result (1000) should be smaller. It might be caused by the CPU exhaustion. Lastly, As the number of lines in the text file increases the execution time increases less than previous experiments. The change is caused by the loop statement in the thread and it causes the client to wait the result however server will access the semaphore at the same amount though number of lines change, because the keywords are located at the same place in the text file at the same amount. The client will use semaphores more but it will not wait after 100 keywords are accessed because the server will not access it anymore.