CS342

Operating Systems

Project 2

04.04.2018

Sabit Gökberk Karaca	21401862	Section 2
Kaan Sancak	21502708	Section 1

1. Timing Experiments

Table 1: Experiment with 4000 lines

	2 Clients	4 Clients	6 Clients	8 Clients	10 Client
Client 1	0.003	0.002	0.003	0.007	0.003
Client 2	0.003	0.003	0.004	0.008	0.004
Client 3		0.004	0.005	0.01	0.013
Client 4		0.005	0.014	0.011	0.013
Client 5			0.014	0.015	0.014
Client 6			0.014	0.017	0.015
Client 7				0.017	0.016
Client 8				0.014	0.017
Client 9					0.019
Client 10					0.017
Average	**0.003**	**0.0035**	**0.009**	**0.012375**	**0.0131**

Table 2: Experiment with 8000 lines

	2 Clients	4 Clients	6 Clients	8 Clients	10 Client
Client 1	0.003	0.004	0.004	0.006	0.005
Client 2	0.003	0.006	0.005	0.006	0.007
Client 3		0.004	0.005	0.008	0.007
Client 4		0.007	0.007	0.007	0.017
Client 5			0.015	0.011	0.017
Client 6			0.015	0.011	0.018
Client 7				0.011	0.023
Client 8				0.015	0.015
Client 9					0.026
Client 10					0.027
Average	**0.003**	**0.00525**	**0.0085**	**0.009375**	**0.0162**

Table 3: Experiment with 12000 lines

	2 Clients	4 Clients	6 Clients	8 Clients	10 Client
Client 1	0.004	0.004	0.005	0.004	0.007
Client 2	0.004	0.005	0.006	0.007	0.007
Client 3		0.005	0.006	0.008	0.013
Client 4		0.007	0.007	0.01	0.017
Client 5			0.012	0.01	0.022
Client 6			0.015	0.011	0.022
Client 7				0.013	0.023
Client 8				0.013	0.023
Client 9					0.023
Client 10					0.02
Average	**0.004**	**0.00525**	**0.0085**	**0.0095**	**0.0177**

Table 4: Experiment with 16000 lines

	2 Clients	4 Clients	6 Clients	8 Clients	10 Client
Client 1	0.005	0.005	0.009	0.007	0.01
Client 2	0.005	0.007	0.01	0.008	0.01
Client 3		0.008	0.01	0.017	0.012
Client 4		0.009	0.01	0.018	0.018
Client 5			0.011	0.017	0.017
Client 6			0.013	0.023	0.021
Client 7				0.023	0.025
Client 8				0.026	0.027
Client 9					0.026
Client 10					0.034
Average	**0.005**	**0.00725**	**0.0105**	**0.017375**	**0.02**

Table 5: Experiment with 16000 lines

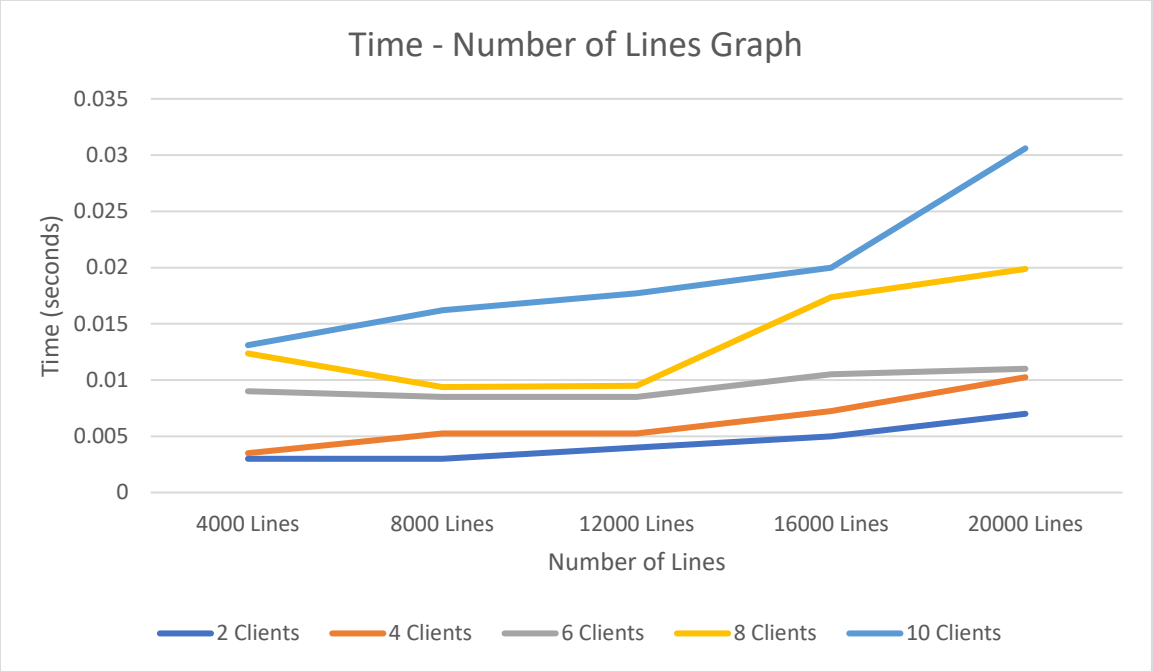
	2 Clients	4 Clients	6 Clients	8 Clients	10 Client
Client 1	0.007	0.007	0.005	0.006	0.08
Client 2	0.007	0.007	0.007	0.015	0.015
Client 3		0.009	0.008	0.017	0.018
Client 4		0.018	0.008	0.019	0.02
Client 5			0.014	0.017	0.022
Client 6			0.024	0.022	0.024
Client 7				0.029	0.026
Client 8				0.034	0.03
Client 9					0.033
Client 10					0.038
Average	**0.007**	**0.01025**	**0.011**	**0.019875**	**0.0306**

Table 6: Averages

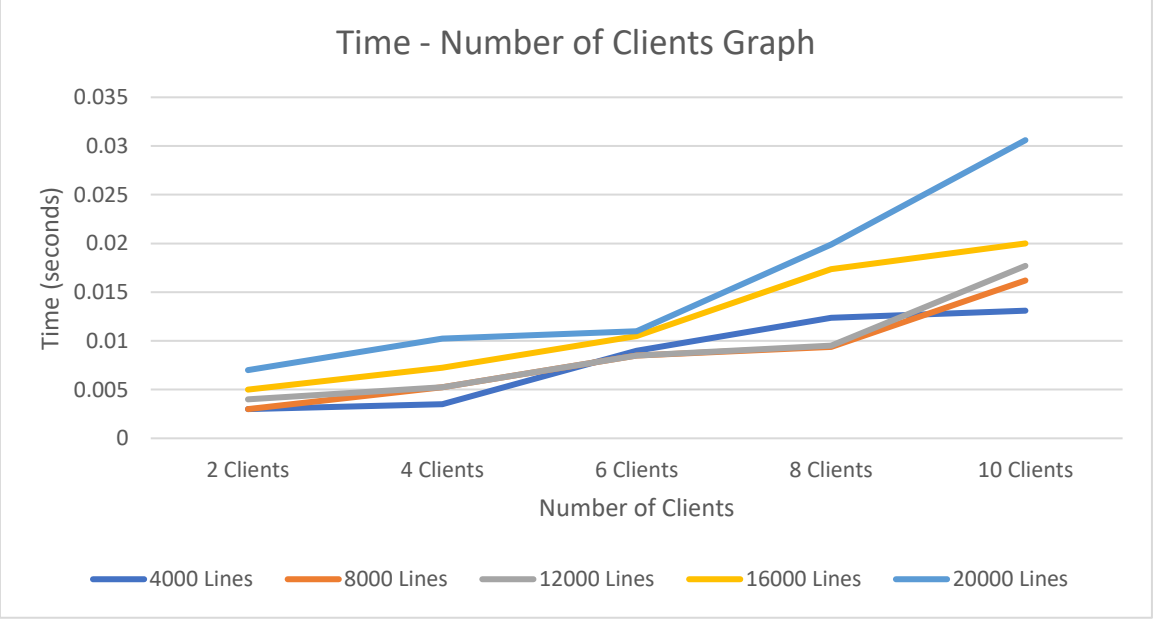
	2 Clients	4 Clients	6 Clients	8 Clients	10 Clients
4000 Lines	0.003	0.0035	0.009	0.012375	0.0131
8000 Lines	0.003	0.00525	0.0085	0.009375	0.0162
12000 Lines	0.004	0.00525	0.0085	0.0095	0.0177
16000 Lines	0.005	0.00725	0.0105	0.017375	0.02
20000 Lines	0.007	0.01025	0.011	0.019875	0.0306

2. Result Analysis

Graph 1: Time – Number of Lines Graph



Graph 2: Time – Number of Clients Graph



3. Conclusion

In this project, we have studied the synchronization problems, multi-threaded applications and inter-process communication. We have implemented a server program and a client program. Server was responsible for getting requests from different clients and processing them concurrently. In order to achieve this, we implemented a server that creates a thread for each request and assigns the task to this thread. We used a shared memory to implement the communication between client programs and the server program. The problem in this implementation was the synchronization of different threads. Since all of the threads was writing to and reading from the same shared memory, application was not reliable. Result queue statuses array and request queue were used commonly by client programs, request queue and result queues were used commonly by client and server programs. So, there were race conditions both between client programs and between server and client program. We achieved to solve the problem by using semaphores. Both the client and server programs lock the critical sections before they use, in that way this part of the shared memory can be modified by just one thread and the program becomes reliable.

After completing the project, we have designed a timing experiment to observe how the changing parameters affect the running time of the application. In this experiment, 5 sample values for number of lines of the input file and 5 sample values for the number of clients are chosen and they are changed one by one. One of the parameters was the size of input file, we used 5 different input files that have linearly increasing number of lines. We used the same text repetitively to get longer files, so it was guaranteed that number of found keywords was also increased for each file. We chose to use this parameter because when the number of found keywords is increased, each number of reads and writes to the shared memory is also increased. The other parameter was number of clients. Since all of the clients were running concurrently, it was important to see how they affect the running time of each other.

For each pair of parameters, average running time of the program is calculated by calculating the average running time of each client. Detailed information about the test results can be found in the related tables (See Tables 1-5). When all the test results are collected, average values are used to create a new table that helps to the analysis of this experiment (See Table 6). Number of lines is fixed and number of clients is increased in each row. Similarly, number of clients is fixed and number of clients is increased in each column. Collected data is visualized after creating the tables because data visualization is a better way of interpreting the test results. By using two parameters, two different tables are created.

When Time – Number of Lines graph is investigated, it can be said that running time of a client is increasing. This is probably a result of increasing number of found keywords. As mentioned before, both the client and the server programs try to modify the shared memory when the keyword is found in a new line. Therefore, it takes more time for the server to process the requests and each client terminates slower when compared to smaller input sizes.

Similarly, Time – Number of Clients graph also indicates that number of clients and running time of clients are proportional. This can be explained by two arguments. When number of clients are increased, number of requests also increases. Therefore, it takes more time for the server to process the requests again. Moreover, when a higher number of clients is used, it means that number of clients that take part in the race condition to modify the shared memory also increases and therefore, running time of each client also increases.

These experimental results were expected before the experiment and they are also consistent with the theoretical knowledge. Therefore, it can be said that this experiment successfully shows the relationship between running time of each client and selected parameters.