General description:

We tested the functionality and the performance of our program by starting different number of servers using three computers. Then we applied the `put` operation several times and every time we measured the performance of our implementation. We then measured the performance of our `get` function by calling it for the inserted keys and we divided the results in three groups for the three computers we were using. Then we also measured the update and delete performances of our program. We performed our measurements for the different cache strategies.

Detailed description:

First, we started with three servers, one on each computer. Then we executed the `put` operation, with cache strategy `FIFO`, several times. We have noticed that the performance of the program is much better, when no server received a `SERVER-NOT_RESPONSIBLE` acknowledgement. In this case the program needs between 5 and 130 milliseconds to put the <key, value> pair on one of the servers. In case that one server receives a `SERVER_NOT_RESPONSIBLE` message, the program needs around 130 milliseconds, when the meta data table has only one element and between 200 and 400 milliseconds when there is more than one element in the meta data table.

We repeated the same tests for the other cache strategies (`LRU, LFU`) and we didn't see much of a difference in the final results. Again, the <key, value> tuples, for which we did not receive a `SERVER_NOT_RESPONSIBLE` acknowledgement were being saved in the appropriate server much faster, then those, for which we received one.

With the <key, value> tuples, that we already had stored in our three servers we tested the performance of the `get` function. The interesting thing here was, that the first access to any of the servers was always much more expensive than all the following.

When we tested the performance of the `update` and `delete` functions, we've noticed the same pattern – the first access to any server was always the most expensive one.

After that we started nine servers (three servers per computer) and repeated our performance measurements for the `put` function. We've noticed that the performance is worse than before. The reason behind that is most likely that we have more elements in the meta data table and the program needs more time to find the server that we need. Otherwise our observations from before (the performance tests with three servers) yielded the same results: when we don't have a `SERVER_NOT_RESPONSIBLE` acknowledgement the performance is more than two times better.

We again used the inserted <key, value> pairs to test the performance of our `get` function with nine running servers. The program needed more than two times more time to return the value that we are looking for. That is mainly because we have more servers and respectively more elements in the meta data table. Once more we establish that the first access to a specific server is much more expensive than any following accesses to the same server.

We came to the exact same conclusions for the performance of the **`delete`** and **`update`** functions.

For our biggest and final test we started 21 servers (seven servers per computer) and again we used the **`put`** operation a couple of times. There was a noticeable difference in the performance between these tests (with 21 running servers) and the previous ones (with nine or three running servers). That is of course logical due to the fact that the meta data table has much more elements.

The exact results from our tests can be found in the following table:

Number of servers	Operation	Time (in milliseconds)		Cache only
3	put	**NP**	5,130,3,96, 12	20 FIFO
		SNR	230, 395	
3	put	**NP**	110, 2, 149, 3, 2	20 LRU
		SNR	215, 276	
3	put	**NP**	103, 2, 2, 83, 25	20 LFU
		SNR	255, 301	
9	put	**NP**	134, 145, 61, 37, 8	20 FIFO
		SNR	364, 617, 555, 103	
9	put	**NP**	8, 9, 25, 113, 60, 82	20 LRU
		SNR	404, 185, 269, 220	
21	put	**NP**	165, 39, 70, 80, 192, 150, 163	20 FIFO
		SNR	605, 515, 390, 443, 456	
3	get	**Local**	30, 12, 3, 3	20 FIFO
		PC1 over network	7, 7	
		PC2 over network	91, 28, 19, 30	
9	get	**Local**	356, 23	20 FIFO
		PC1 over network	55, 6, 19, 10, 10	
		PC2 over network	180, 88, 19, 44, 36, 11	
3	update	**Local**	226, 4	20 FIFO
		PC1 over network	98, 7, 4, 5	
		PC2 over network	198, 12, 45, 9	
9	update	**Local**	339,7	20 FIFO
		PC1 over network	101, 6, 6, 8	
		PC2 over network	211, 10, 15, 64	
9	delete	**Local**	368, 369	20 FIFO
		PC1 over network	108, 37, 21	
		PC2 over network	84, 39, 13	

*NP = Normal put (without SERVER_NOT_RESPONSIBLE)
* SNR = put with SERVER_NOT_RESPONSIBLE