FACULTY OF ENGINEERING

Assignment 01

Distributed Systems

Presented by
1- Omar Ahmed Fouad 40
2- Mostafa Fathy 67
3- Youmna Dwidar 73

5th April 2020

Table of Contents

1 Problem Definition

In this assignment , we were asked to design a client server system that consists of multiple nodes. The communication is based on RMI. The client invokes a method in the server using the remote method invocation.

The server provides the service of updating and querying the shortest path on the changing graph. Using this problem was made on the importance of its applications such as GPS navigations , routing schemes in computer networks .. etc . First, We will have an initial graph. The graph is pre processed and stored preparing for the coming requests. After finishing processing the graph, a workload is issued to the server by multiple clients at the same time and their requests should be answered or served as fast as possible.

Requirements :

- design a server that reads a graph from a file , processes it and implements the 3 methods of querying shortest path , adding a new edge or deleting an existing edge .
- The server methods should handle the consistency problems and have locks to maintain them.
- No two write operations can be performed simultaneously.
- The server object must be a remote object in order to be called by RMI.
- design and implement a client that invokes the methods in the server by sending requests.
- measure the performance of the server with multiple numbers of operations , clients and write percentage in those operations .

2 Algorithms

The server Design :

- The server consists of a remote interface **GraphInterface** and a remote class **Graph** that implements this interface.
    - **GraphInterface :**
        - consists of three methods query , add , delete.
        - each takes 2 attributes which are node1 and node2
    - **Graph class :**
        - implements the graph interface methods with the help of some other private methods .
        - read the initial graph from either the standard input (console) or a file.
            - readFromFile
            - readFomStandardInput
        - Getting the query result is done by the usage of the BFS algorithm as the graph is unweighted so the shortest path can be obtained by BFS.
        - delete and add are using the two methods **getIndex** and **createIndex** who are responsible of addressing the graph and each node index to easily get its neighbors from the graph.
        - for consistency issues :
            - we are using locks in the add and delete methods (writes)
            - for the reading , we couldn't iterate on a changing data structure , so we used java concurrent Data structures such as **ConsurrentHashMap** and **copyOnWriteArrayLists**
    - **Main Class :**
        - the main class initializes a local registry with a specified port.
        - rebind an object of the graph to a name to be identified by in the registry.

The client Design :

- The client is very simple.
- It gets the remote object after getting the registry of the server by host ip address and port number.
- Each operation is randomly generated with the given percentage of writing given at start time.
- the operations can be random generated or can be given as a file format.

4 Implementation

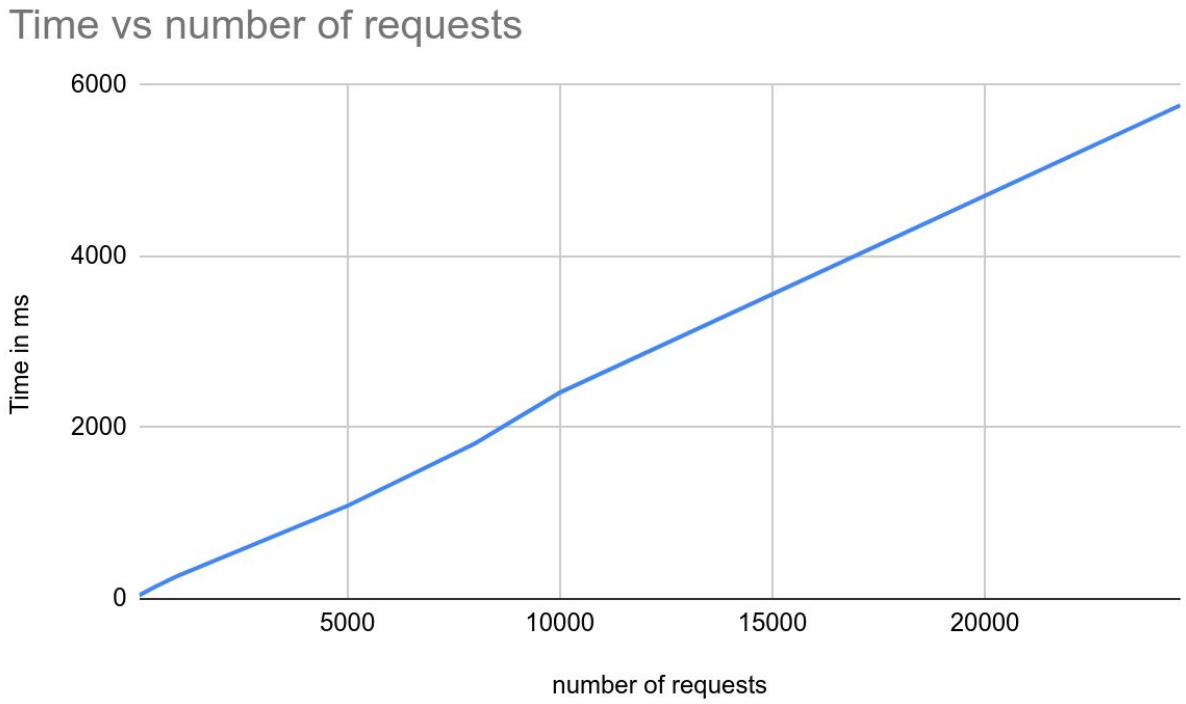
We tested the client and the server on the same machine. Port forwarding is also used to make the server public so that we can use the internet in sending the client requests to the server. We didn't test it, afraid to play with router settings :)

The test data is generated randomly at each run of the client. You provide the number of requests and the percentage of the write requests.

5 Results

One Client running :

number of requests	Time
100	40.415223
500	147.419697
1000	266.662164
5000	1086.61492
8000	1813.560378
10000	2409.334904
24600	5764.007286

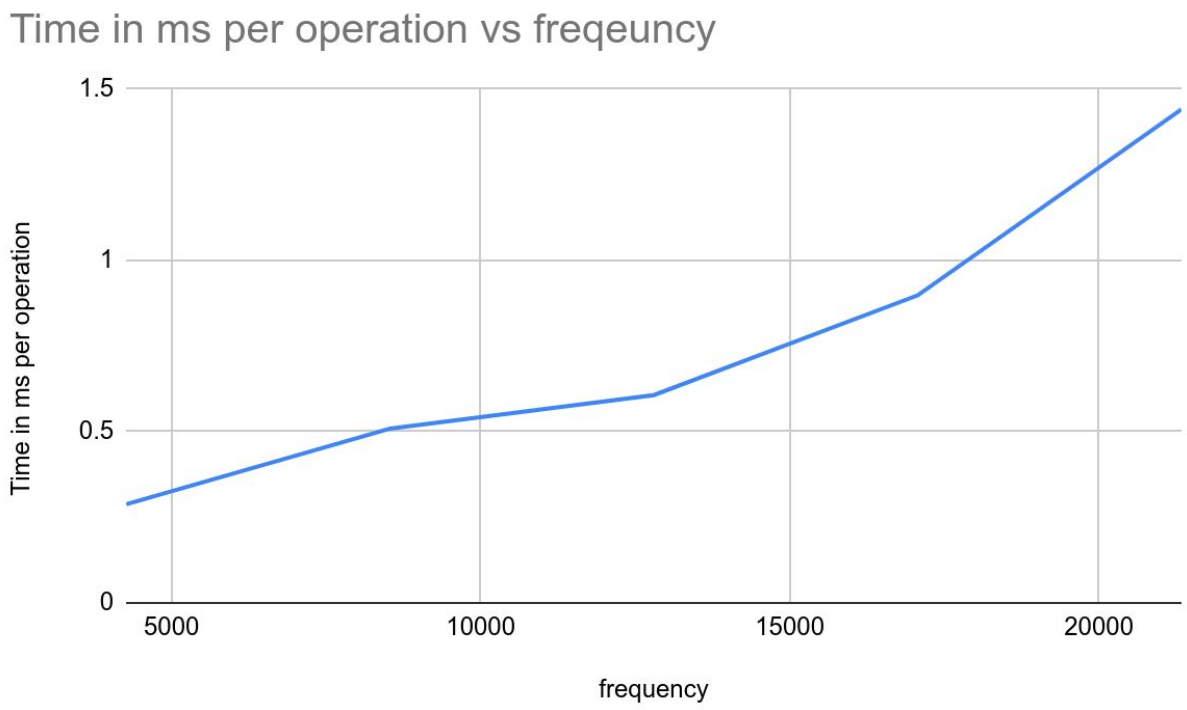


We can get the frequency of requests in 1 client. It will be used in the next graphs.

Frequency = 4267.86414024 nearly 4267 request/second

Frequency ver average response time per operation : Fixed percentage of writes (50%)

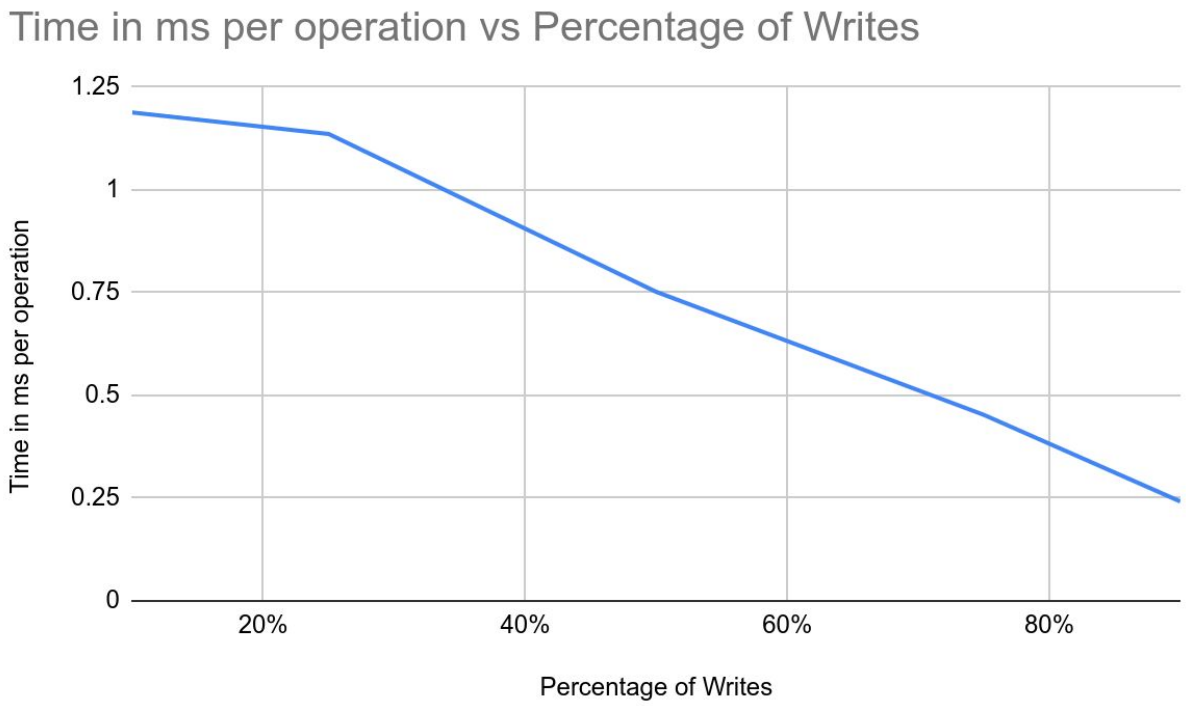
frequency	Time in ms per operation
4267	0.287874433
8534	0.508983412
12801	0.606517243
17068	0.897950779
21335	1.440870882



response time Vs percentage of write operations (one client)

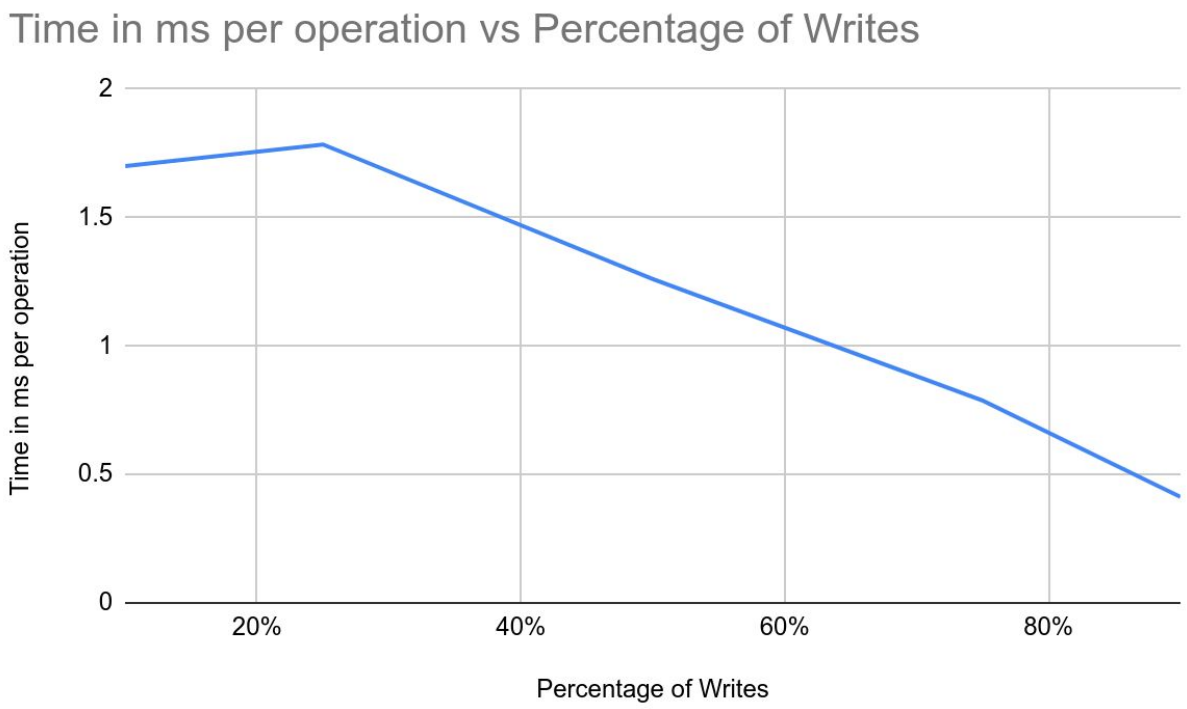
Percentage of Writes	Time in ms per operation
10%	1.1882
25%	1.13609
50%	0.75185
75%	0.4524
90%	0.241869

makes sense as the delete operation and the add is O(1) and the query is O(V+E) and all performed sequentially as they are from one client.



We will try again but with 3 clients .

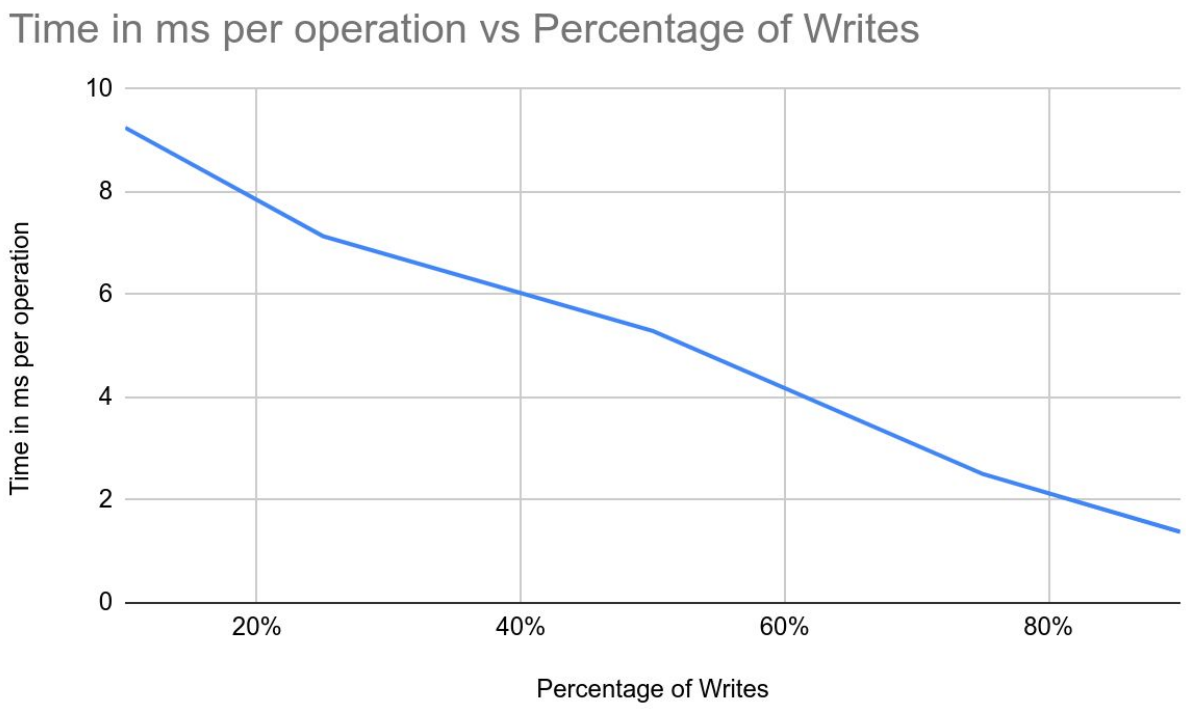
Percentage of Writes	Time in ms per operation
10%	1.7
25%	1.784346763
50%	1.260717832
75%	0.787528395
90%	0.412469568



Even with multiple clients, If we have 8000 writes by every client performed sequentially will also result in less time than any of them with higher query percentage.

even with using 10 threads .

Percentage of Writes	Time in ms per operation
10%	9.25
25%	7.137
50%	5.293
75%	2.508
90%	1.381



This means that we may need to optimize the query more.