Project 3

Osamah Alzacko & Anurag Rana

# Server Implementation

The server is working in a multi-threaded architecture as follows:

* Server main socket listener
  + Keep listening to new connections and reply to any incoming command.
* Recovery thread
  + A separate thread that handles the connection to two servers before the host and two servers after, and handles saving the stream of keys from each server.
* Command thread
  + A separate thread that handles user commands directly on the server like read a value, write a value, print all key:value in the server, recover key store from other servers, and a kill switch to quit the application.

## Features

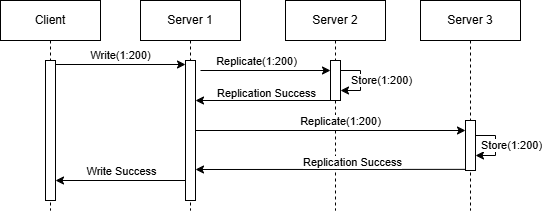
1. Recovery: The server supports recovery on startup, each host will try to connect to the 2 predecessor servers and 2 successor servers in the ring and will get all key:value store that is related to the host using the Hash function to determine if the key should be stored in the host.

A diagram of a server

AI-generated content may be incorrect.

Server recovery sequence.

1. Replication: Servers support replication of write commands and each write would be executed in the host server and on 2 successor servers for replication. When a write command is executed, the connection with the client is blocked until the write command is executed and replicated (if the replication was successful with 1 or 2 replications) then write will be executed on the server and an acknowledgement is sent to the client.



Replication process

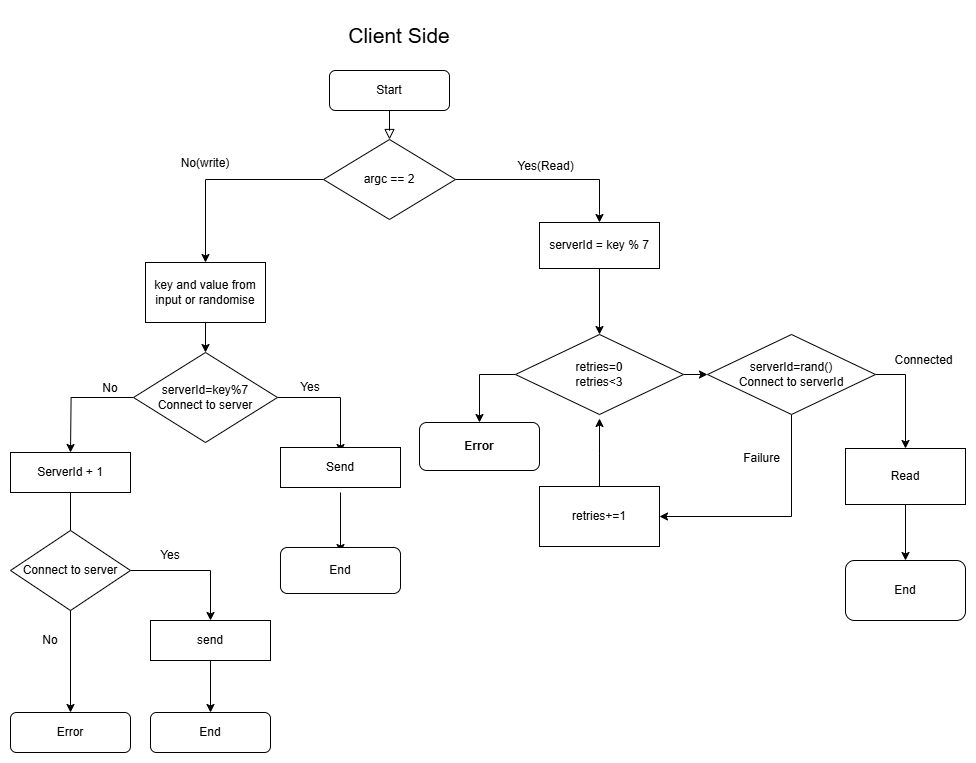
# Client Implementation

The client has sequential architecture as follows: -

* Commands for read and write:
* Read: - ./client [message key]
* Write: - ./client [message key] [message value]
* All parameters of read and write commands are numeric and write parameters of [message key] and [message value] are optional.
* For write if we don’t provide params, we assign randomized [message key] and [message value]
* Connection request:
* For write, server ID to which the connection request should be sent is decided based on [message key], by modulo of 7.
* For read, server ID is derived from [message key] using modulo of 7, then the connection request is sent randomly to anyone of the servers among between server ID, server ID.
* Read:
* Read request is sent to the designated server.
* If the read request is sent successfully, we receive the value of that specific key from that server and print it on client side, if no value is found 0 is received.
* Write:
* On write, we send the key and message, which is either randomized or received as parameters of write command.
* If write request is sent successfully, we receive an acknowledgement if write is successful on server end else we receive 0 if write is failed on server end.
* Disconnect Socket:
* Respective socket is closed when we receive response from the server, on both read and write operation.

Note: - Client-side operation can handle only one read, or one write operation at a time, which gives more flexibility, regarding how many messages are to be sent or if we need to perform individual read. Multiple messages are sent using iterations in bash command itself, which prevents any hardcoded values inside the code.

On client-side, failures are displayed in Red, requests are displayed in Yellow, success responses are displayed in Green.



Client-Side flow chart

# Experiments, Results and Outputs

Experiment 1:- All servers are active, and client is sending read and write requests

Result:-

* On write:
* message is successfully sent and written to all 3 servers namely, [serverID], [serverID + 1], [serverID + 2] and acknowledgement is sent back.

A screenshot of a computer program

AI-generated content may be incorrect.

Write done on 3 servers

* On read:
* Request with key is sent to any one of the 3 servers namely, [serverID], [serverID + 1], [serverID + 2], and server sends back the value for given key.

A screenshot of a computer program

AI-generated content may be incorrect.

Read happening from live servers

Experiment 2:- Few servers are down, and client is sending read and write requests

Result:-

* On write:
* For write, if 1 server is down, write was performed on other 2, but if 2 servers are down, an error is sent back to client and write is not performed.

A screenshot of a computer program

AI-generated content may be incorrect.

Write happens when 2 servers are alive out of 3

A screenshot of a computer screen

AI-generated content may be incorrect.

Error is thrown if 2 or more servers are down, on write

* On read:
* For read, if 1 server is down , request is then sent randomly to remaining 2 and if 2 are down , request is sent to last server, and if all are down error is sent back to client.

A screenshot of a computer program

AI-generated content may be incorrect.

Read Success if at least one server is alive

A screenshot of a computer program

AI-generated content may be incorrect.

Read Failed when all three servers are down

Experiment 3:- Clients sending messages concurrently, but total ordering is maintained

Result:- Total Ordering is ensured as all incoming requests from client goes into the listen queue and server handles it one by one in order , so as connection between server is TCP, 1st message propagated by the server is also the 1st message delivered at other 2 servers.

Experiment 4:- Read request is sent randomly to any of the 3 associated servers

Result:- Based on key we decide the server and its related servers and read request is sent to any one of them randomly.

A screenshot of a computer program

AI-generated content may be incorrect.

Server to read form is randomly selected from 3 designated servers

Experiment 5:- Write an existing key will overwrite the old value instead of making new entry

Result:- If in the write command we add 2 params key and value, if the key exists on the server derived from moduling key by 7, then its value will be replaced by new value passed as param in bash command.

A screenshot of a computer program

AI-generated content may be incorrect.

When key and value are sent in params, client updates if the value exists on servers

Experiment 6:- Recovery of crashed servers

Result:- On recovery, the server goes up again ready to accept requests, to be noted server also accepts commands and if we give recover command , it will send requests to 2 servers behind and ahead of it to retrieve the key and value related to itself.

A screenshot of a computer

AI-generated content may be incorrect.

Before Recovery

A screenshot of a computer program

AI-generated content may be incorrect.

After recovery, all keys related to crash servers are retrieved from related servers