System Description:

This project is an implementation of Causal Consistency model between the servers.

The project has a Server class that takes the port to be run on as the argument.

It next requests for the number of other servers/datacentres that would be run. Upon prompt please enter the ports of those datacentre/ servers.

Assumption is made that this POC will be running on a single machine. Hence the IP addresses are not input to the program.

The server now starts and listens to incoming client connections and spawns a thread for each client connected.

The Client class takes as argument the Server's port number to which it has to connect.

Valid commands from client are:

1. read "element"

2. write "element" "value"

The client program takes these inputs from the user and passes it on to the server. Server does a replicated write to all the other servers in the network with the dependency and value.

Server spawns another thread to connect to the other server, in which case it is treated as a client by the server. When The servers sees a connection with message "serverwriteconnect" it does the dependency check and either commit entry or push it to queue for later processing.

This thread exits and returns result to the main server.

A read action causes to add dependency of the read element on to the client dependency list.

Write causes to send the dependency to other servers, clear dependency, write value and update the new dependency to the client dependency list.

The same happens on each server.

We use two maps, one to store the data and other to store dependency. No commit is done without checking the dependency.

When a replicate write happens, we iteratively check the queue to process all pending write requests.

If unseen keys are encountered system indicates key not found. Minimal NullPointerExceptions.

Simulation:

To simulate the first example, a check is added to induce sleep when data key is b and server is 3.

Run the Server.class file with arguments "port" "serverid"

You will be prompted to enter number of servers and their ports:

~$: Server 8585 1

Enter the number of servers

2

Enter the server List

8585

8888

Server Started ....on port: 8585

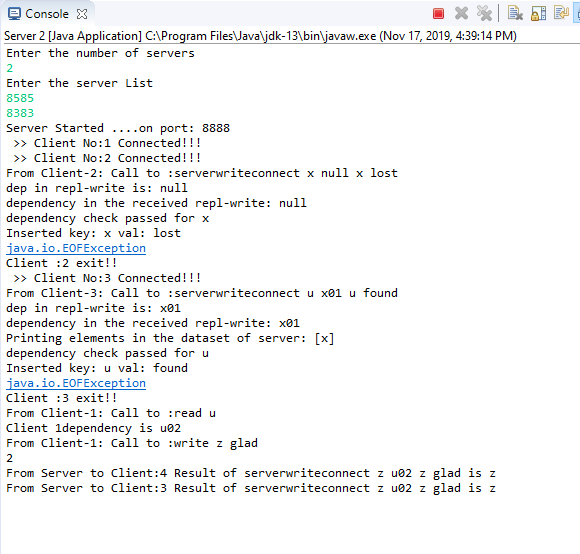
Run the client.class file with server port as argument:

~$: Client 8585

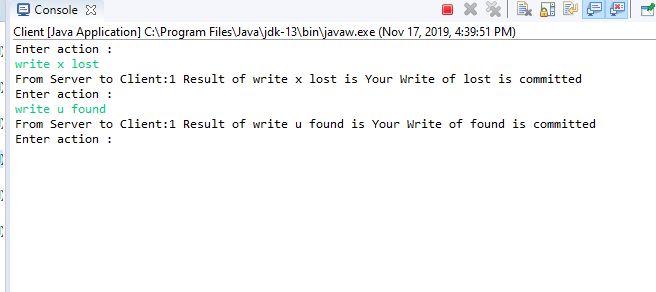
Enter action :

Sample outputs are attached:

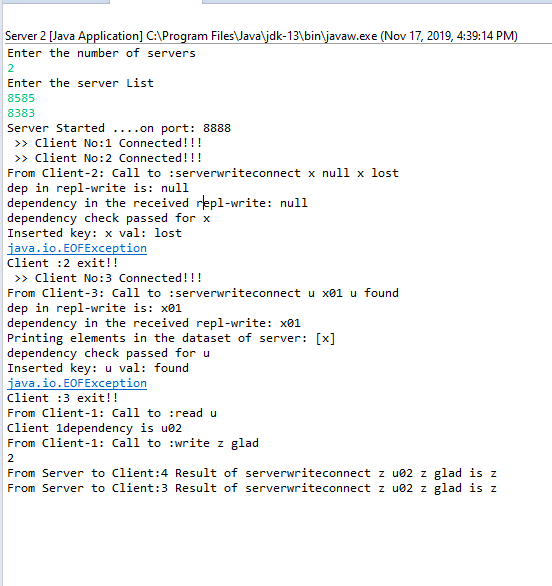
Server1 :port 8585



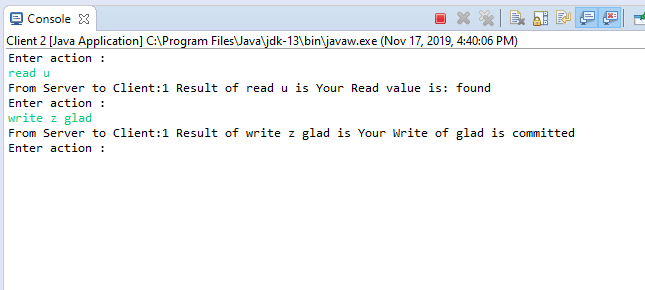
Client 1: Server1



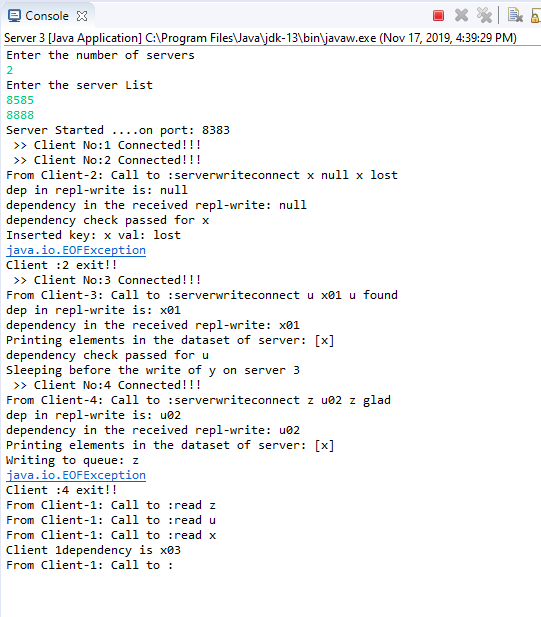
Server 2: port 8888



Client 2: Server 2

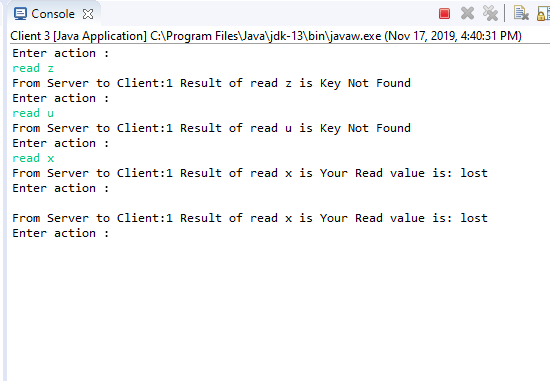


Server 3: port 8383



Client 3 doesnot commit z because of dependency.

Client 3: Server 3



We can see that the element is added to queue for later processing. The key u is not entered yet due to sleep induced.

Once the sleep is interrupted, u will be entered and z will be resumed from queue, as shown next.

Server 3:



Client 3:

