Distributed Key-Value Store

Rushikesh Pharate

Overview

Consistency model defines the valid sequence of operations in a given distributed system. There is a wide range of consistency models available today. The main ones that we are going to implement are

1) Linearizability
2) Sequential consistency
3) Eventual consistency
4) Causal consistency

Algorithm/control flow for each consistency level

1. **Linearizability:**
   - The client issues GET/SET request to any of the servers.
   - After receiving the client request, the server, if not primary
     - Creates a new thread and makes an RPC call to primary and sends the GET/SET request details
     - Maintains a connection to the client **(Blocking request)**
     - Waits for a message from the primary
   - Upon receiving the message from any server, the primary server
     - Maintains a queue for incoming requests using lock
     - Acquires lock, and broadcasts (ACTION, VARIABLE, VALUE) message
     - When all the messages return, releases the lock
   - After receiving the broadcast message, the server
     - Updates its local store with the received variable details, if its a SET request
     - If its a GET requests doesn't do anything
     - If the requesting server, then it reads the value of the variable and sends it back to client via the maintained connection and closes the client connection

2. **Sequential Consistency:**
   - The client issues GET/SET request to any of the servers.
   - After receiving the client request, the server, if not primary
     - If its a SET request
       - Creates a new thread and makes an RPC call to primary
       - Sends the SET request details
       - Maintains a connection to the client (**Blocking request**)

        - If its a GET request
            - Reads the value of the variable from the local store.
            - Return the value immediately if the variable is present in the store else return None.
            - Close the client connection (**Non-Blocking request**)
    - Upon receiving the message from any server, the primary server
        - Maintains a queue for incoming requests using lock
        - Acquires lock, and broadcasts (ACTION, VARIABLE, VALUE) message. (Note: ACTION is always SET in sequential consistency because we only broadcast SET requests)
        - When all the messages return, releases the lock
    - After receiving the broadcast message, the server
        - Updates its local store with the received variable details.
        - If the requesting server, then it closes the client connection

3. **Eventual Consistency:**
    - The client issues GET/SET request to any of the servers.
    - After receiving the client request, the server, if not primary
        - If its a SET request
            - Creates a new thread and makes an RPC call to primary and sends the SET request details
            - In the main thread, closes the connection to the client (**Non-Blocking request**)
        - If its a GET request
            - Reads the value of the variable from the local store.
            - Return the value immediately if the variable is present in the store else return None.
            - Close the client connection (**Non-Blocking request**)
    - Upon receiving the message from any server, the primary server
        - Maintains a queue for incoming requests using lock
        - Acquires lock, and broadcasts (ACTION, VARIABLE, VALUE) message. (Note: ACTION is always SET in eventual consistency because we only broadcast SET requests)
        - When all the messages return, releases the lock
    - After receiving the broadcast message, the server
        a. Updates its local store with the received variable details.
        b. Does not care about the client connection as its already closed.

4. Causal Consistency:

- Every client maintains a logical clock maintaining the version of each variable that it deals with
- Similarly each server also keeps track of each variable logical clock
- Every request from client contains a minimum acceptable logical clock value for that variable along with the GET/SET request details
- Every response from server contains the Updated/Latest logical value for that variable along with the GET/SET response

Now, the algorithm:

- The client issues GET/SET request to any of the servers alog with minimum acceptable logical clock value.
- After receiving the client request, the server, if not primary
    - If its a SET request
        - Creates a new thread and makes an RPC call to primary and sends the SET request details
        - In the main thread, waits to get an update on the logical clock from primary
        - When the server gets a message from primary, it updates it logical clock value, local store and sends the latest clock value to the client as a response and closes the connection.
    - If its a GET request
        - Compares the request logical clock with its own logical clock for that variable
        - If the requests logical clock value is greater than its own then it essentially means that this message is from the future and it needs to wait for some time to respond to this.
        - In the meantime, if it receives the update via broadcast form primary, it updates its logical clock. As soon as its logical clock becomes equal to or greater than the client requests logical clock, it reads the value for that variable from the local store and returns it to the client along with the latest logical clock value for that variable
        - Closes the connection to client
- Upon receiving the message from any server, the primary server
    - Maintains a queue for incoming requests using lock
    - Acquires lock, increments logical clock for the VAR and broadcasts (ACTION, VARIABLE, VALUE) message. (Note: ACTION is always SET in causal consistency because we only broadcast SET requests)
    - When all the messages return, releases the lock

- After receiving the broadcast message, other servers
    a. Updates their local store with the received variable details.
    b. Updates the logical counter for that variable.

Please refer to the below table for a comparison of the nature of GET/SET requests for each consistency model

Consistency Model	**Linearizability**	**Sequential**	**Eventual**	**Causal**
SET request	Blocking	Blocking	Non-Blocking	Somewhat Non-Blocking
GET request	Blocking	Non-Blocking	Non-Blocking	Somewhat Non-Blocking

Table Blocking/non-Blocking

Steps to Run Distributed Key-Value Store:

- **Prerequisites**
  - You have the submission folder
  - Your base directory is the Dist-KV-Store folder

- **Starting the Servers:**
  - Open *config.json* file and change *NO_OF_SERVERS, PORT_START_RANGE*, or any other parameter as per your convenience.
  - Open a new terminal and run ***python3 main.py*** command
  - After this logs will be generated informing you about the servers and their accessibility on ports.

- **Starting the Client:**
  - *client.py* file contains a code to make a *GET/SET* request to any server.
  - There are 2 ways to run an client:
    - Run ***python3 client.py [server address] [server port number] [command] [var] [value]*** command. Here, command can be either GET/SET. *[Value]* should only be passed for SET requests.
      1. ex. SET request → *python3 client.py localhost 6486 SET x 5*
      2. ex GET request → *python3 client.py localhost 6486 GET x*
    - Run *bash run_clients.py* command after making commenting/uncommenting required details in the *run_clients.py* file. This will in turn call the above command but you can change this shell file to make concurrent requests to the same or different servers.
  - Running client for causal consistency is a bit different as we need to contact multiple servers. Make sure *NO_OF_SERVERS* is set 10 in *config.json* file as the ports are hardcoded for causal client.
    - Run *python3 client_causal.py localhost* or navigate to *test-cases/[consistency model]* and run *bash run_clients.py*

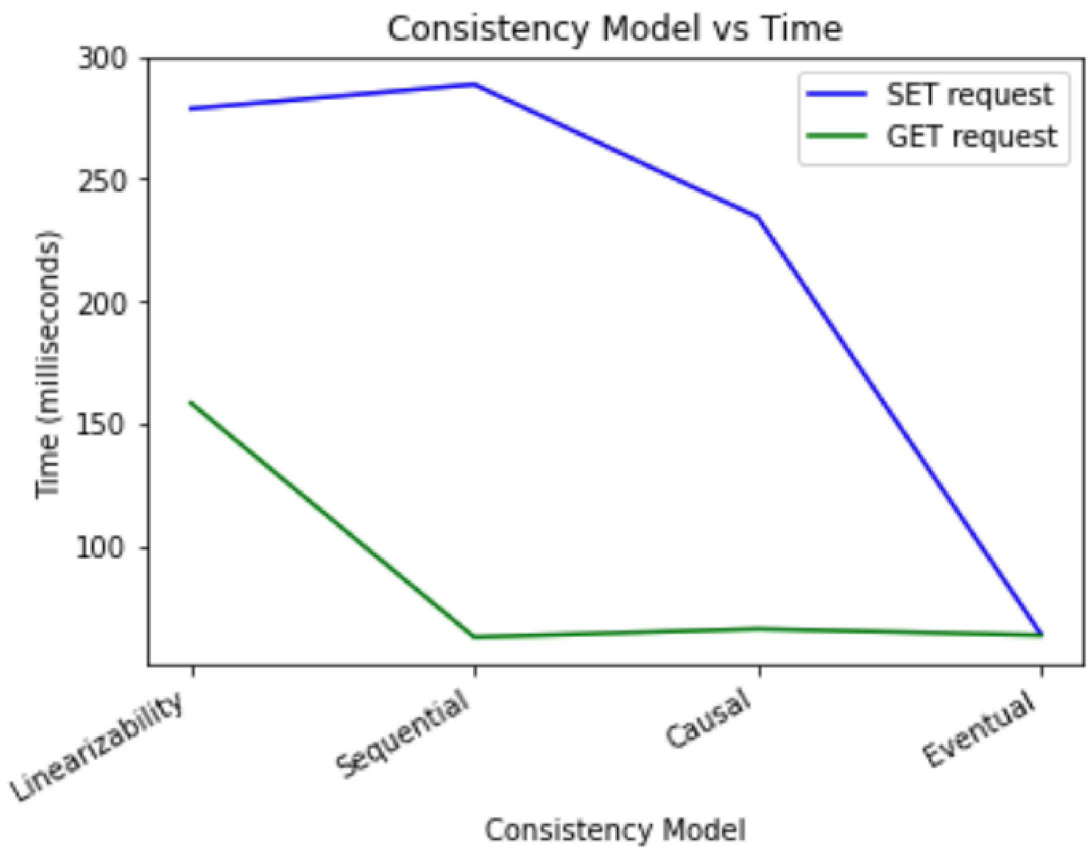
- When a client is started, the logs will be generated in the server terminal and you can see the logic flow

- **Run Test Cases:**
  - The submission folder has a *test cases* folder which contains bash scripts for testing different consistency levels
  - Navigate to *test cases/[consistency model]* folder
  - Run *bash run_clients.sh* command
  - This should start a test case for a particular consistency model. You can always change the shell script based on what you wish to test.
  - Bash scripts for performance testing are also there in the respective folders.

Performance Evaluation:

Consistency Model	Linearizability (milliseconds)	Sequential (milliseconds)	Eventual (milliseconds)	Causal (milliseconds)
SET request (avg of 100 requests and 10 servers)	278.77	288.72	63.96	234.40
GET request (avg of 100 requests and 10 servers)	158.36	62.62	63.42	66.01

Please Note: For causal consistency, the GET requests are conditionally blocking so ideally the latency should be higher than 66.01 milliseconds. As I was not able to create an circumstance for 100 requests to wait, this is just normal Non-Blocking GET request



Fig

As you can clearly see from the graph, overall Eventual consistency model is the fastest. The second fastest is the Causal and after that comes Sequential model. The slowest amongst 4 is the Linearizability.

Challenges:

- The most challenging part of the assignment of testing the behavior of each consistency model and making sure its correct. I used logging for this and have used custom delays in some places to demonstrate in the test cases.
- Another challenging part was developing the primary-based broadcast algorithm. I had used a similar algorithm for Distributed Sequencer algorithm so that helped.
- Causal consistency Implementation as the algorithm provided in the slides is not clear.

Limitations and Future Improvement

1) Current algorithm is very centered around primary and this will create performance issues and primary is doing most of the heavy lifting. To overcome this we can use a total-order multicast algorithm that will solve the performance issue but has its own complexity issues.
2) If one of the server fails, the whole system might crash. This can be avoided if we use some kind of fault tolerance.

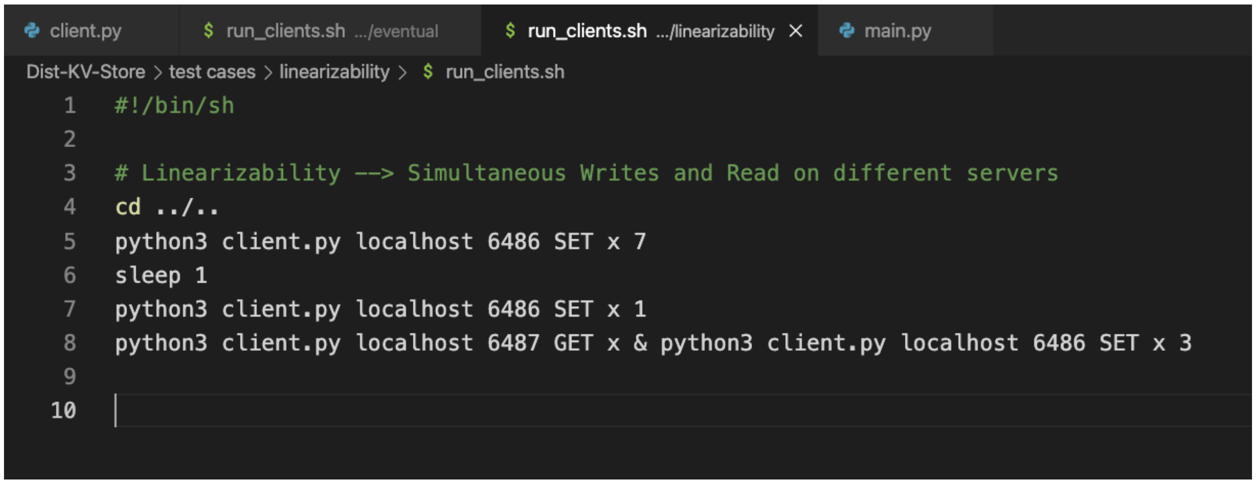
<u>Test Cases:</u>

Linearizability:

Servers 1 to 3 are running on 6485 to 6487 ports respectively

Sequence of request:

1) ***SET x 7*** on server 2
2) ***SET x 1*** on server 2
3) ***GET x*** on server 3 & ***SET x 3*** on server 2 → simultaneous requests


client.py
run_clients.sh .../eventual
run_clients.sh .../linearizability
main.py
Dist-KV-Store > test cases > linearizability > run_clients.sh

```
#!/bin/sh

# Linearizability --> Simultaneous Writes and Read on different servers
cd ../..
python3 client.py localhost 6486 SET x 7
sleep 1
python3 client.py localhost 6486 SET x 1
python3 client.py localhost 6487 GET x & python3 client.py localhost 6486 SET x 3

```

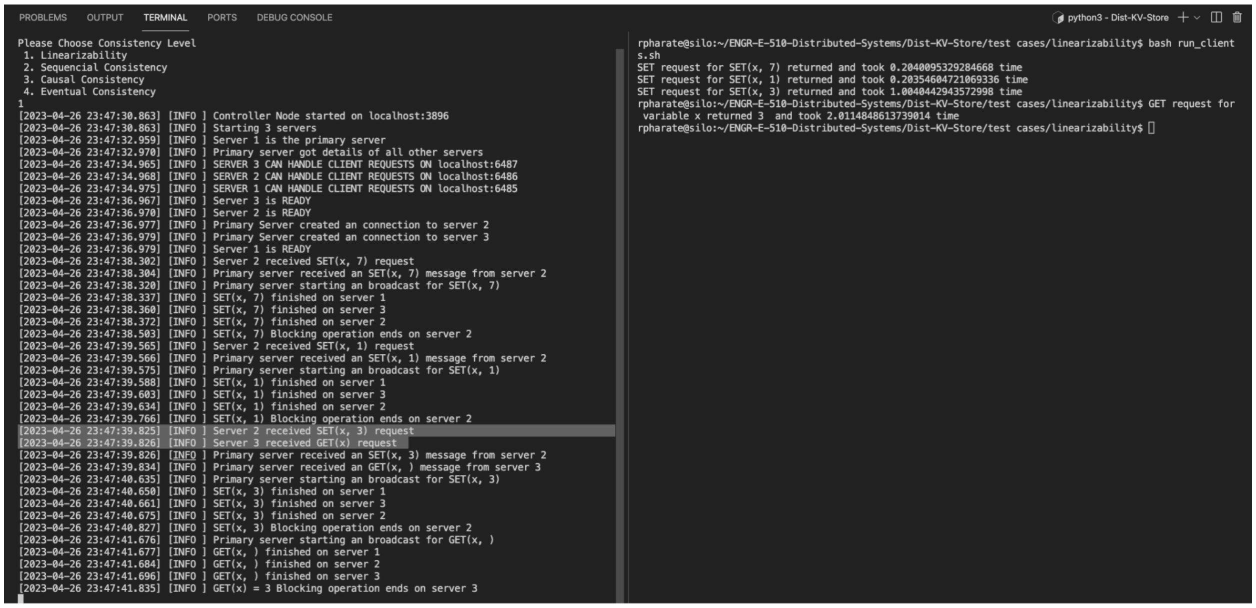
Server Logs:

Please Choose Consistency Level
 1. Linearizability
 2. Sequential Consistency
 3. Causal Consistency
 4. Eventual Consistency
1
[2023-04-26 23:47:30.863] [INFO] Controller Node started on localhost:3896
[2023-04-26 23:47:30.863] [INFO] Starting 3 servers
[2023-04-26 23:47:32.959] [INFO] Server 1 is the primary server
[2023-04-26 23:47:32.970] [INFO] Primary server got details of all other servers
[2023-04-26 23:47:34.965] [INFO] SERVER 3 CAN HANDLE CLIENT REQUESTS ON localhost:6487
[2023-04-26 23:47:34.968] [INFO] SERVER 2 CAN HANDLE CLIENT REQUESTS ON localhost:6486
[2023-04-26 23:47:34.975] [INFO] SERVER 1 CAN HANDLE CLIENT REQUESTS ON localhost:6485

```
[2023-04-26 23:47:36.967] [INFO ] Server 3 is READY
[2023-04-26 23:47:36.970] [INFO ] Server 2 is READY
[2023-04-26 23:47:36.977] [INFO ] Primary Server created an connection to server 2
[2023-04-26 23:47:36.979] [INFO ] Primary Server created an connection to server 3
[2023-04-26 23:47:36.979] [INFO ] Server 1 is READY
[2023-04-26 23:47:38.302] [INFO ] Server 2 received SET(x, 7) request
[2023-04-26 23:47:38.304] [INFO ] Primary server received an SET(x, 7) message from server 2
[2023-04-26 23:47:38.320] [INFO ] Primary server starting an broadcast for SET(x, 7)
[2023-04-26 23:47:38.337] [INFO ] SET(x, 7) finished on server 1
[2023-04-26 23:47:38.360] [INFO ] SET(x, 7) finished on server 3
[2023-04-26 23:47:38.372] [INFO ] SET(x, 7) finished on server 2
[2023-04-26 23:47:38.503] [INFO ] SET(x, 7) Blocking operation ends on server 2
[2023-04-26 23:47:39.565] [INFO ] Server 2 received SET(x, 1) request
[2023-04-26 23:47:39.566] [INFO ] Primary server received an SET(x, 1) message from server 2
[2023-04-26 23:47:39.575] [INFO ] Primary server starting an broadcast for SET(x, 1)
[2023-04-26 23:47:39.588] [INFO ] SET(x, 1) finished on server 1
[2023-04-26 23:47:39.603] [INFO ] SET(x, 1) finished on server 3
[2023-04-26 23:47:39.634] [INFO ] SET(x, 1) finished on server 2
[2023-04-26 23:47:39.766] [INFO ] SET(x, 1) Blocking operation ends on server 2
[2023-04-26 23:47:39.825] [INFO ] Server 2 received SET(x, 3) request
[2023-04-26 23:47:39.826] [INFO ] Server 3 received GET(x) request
[2023-04-26 23:47:39.826] [INFO ] Primary server received an SET(x, 3) message from server 2
[2023-04-26 23:47:39.834] [INFO ] Primary server received an GET(x, ) message from server 3
[2023-04-26 23:47:40.635] [INFO ] Primary server starting an broadcast for SET(x, 3)
[2023-04-26 23:47:40.650] [INFO ] SET(x, 3) finished on server 1
[2023-04-26 23:47:40.661] [INFO ] SET(x, 3) finished on server 3
[2023-04-26 23:47:40.675] [INFO ] SET(x, 3) finished on server 2
[2023-04-26 23:47:40.827] [INFO ] SET(x, 3) Blocking operation ends on server 2
[2023-04-26 23:47:41.676] [INFO ] Primary server starting an broadcast for GET(x, )
[2023-04-26 23:47:41.677] [INFO ] GET(x, ) finished on server 1
[2023-04-26 23:47:41.684] [INFO ] GET(x, ) finished on server 2
[2023-04-26 23:47:41.696] [INFO ] GET(x, ) finished on server 3
[2023-04-26 23:47:41.835] [INFO ] GET(x) = 3 Blocking operation ends on server 3
```

Note the sequence of operations for simultaneous requests is same on all the servers First *SET x 3* is broadcasted to all the 3 servers and then *GET x* is broadcasted.

Logs **Screenshot:**



```
Please Choose Consistency Level
 1. Linearizability
 2. Sequencial Consistency
 3. Causal Consistency
 4. Eventual Consistency
1
[2023-04-26 23:47:30.863] [INFO ] Controller Node started on localhost:3896
[2023-04-26 23:47:30.863] [INFO ] Starting 3 servers
[2023-04-26 23:47:32.959] [INFO ] Server 1 is the primary server
[2023-04-26 23:47:32.970] [INFO ] Primary server got details of all other servers
[2023-04-26 23:47:34.965] [INFO ] SERVER 3 CAN HANDLE CLIENT REQUESTS ON localhost:6487
[2023-04-26 23:47:34.968] [INFO ] SERVER 2 CAN HANDLE CLIENT REQUESTS ON localhost:6486
[2023-04-26 23:47:34.975] [INFO ] SERVER 1 CAN HANDLE CLIENT REQUESTS ON localhost:6485
[2023-04-26 23:47:36.967] [INFO ] Server 3 is READY
[2023-04-26 23:47:36.970] [INFO ] Server 2 is READY
[2023-04-26 23:47:36.977] [INFO ] Primary Server created an connection to server 2
[2023-04-26 23:47:36.979] [INFO ] Primary Server created an connection to server 3
[2023-04-26 23:47:36.979] [INFO ] Server 1 is READY
[2023-04-26 23:47:38.302] [INFO ] Server 2 received SET(x, 7) request
[2023-04-26 23:47:38.304] [INFO ] Primary server received an SET(x, 7) message from server 2
[2023-04-26 23:47:38.320] [INFO ] Primary server starting an broadcast for SET(x, 7)
[2023-04-26 23:47:38.337] [INFO ] SET(x, 7) finished on server 1
[2023-04-26 23:47:38.360] [INFO ] SET(x, 7) finished on server 3
[2023-04-26 23:47:38.372] [INFO ] SET(x, 7) finished on server 2
[2023-04-26 23:47:38.503] [INFO ] SET(x, 7) Blocking operation ends on server 2
[2023-04-26 23:47:39.565] [INFO ] Server 2 received SET(x, 1) request
[2023-04-26 23:47:39.566] [INFO ] Primary server received an SET(x, 1) message from server 2
[2023-04-26 23:47:39.575] [INFO ] Primary server starting an broadcast for SET(x, 1)
[2023-04-26 23:47:39.588] [INFO ] SET(x, 1) finished on server 1
[2023-04-26 23:47:39.603] [INFO ] SET(x, 1) finished on server 3
[2023-04-26 23:47:39.634] [INFO ] SET(x, 1) finished on server 2
[2023-04-26 23:47:39.766] [INFO ] SET(x, 1) Blocking operation ends on server 2
[2023-04-26 23:47:39.825] [INFO ] Server 2 received SET(x, 3) request
[2023-04-26 23:47:39.826] [INFO ] Server 3 received GET(x) request
[2023-04-26 23:47:39.826] [INFO ] Primary server received an SET(x, 3) message from server 2
[2023-04-26 23:47:39.834] [INFO ] Primary server received an GET(x, ) message from server 3
[2023-04-26 23:47:40.635] [INFO ] Primary server starting an broadcast for SET(x, 3)
[2023-04-26 23:47:40.650] [INFO ] SET(x, 3) finished on server 1
[2023-04-26 23:47:40.661] [INFO ] SET(x, 3) finished on server 3
[2023-04-26 23:47:40.675] [INFO ] SET(x, 3) finished on server 2
[2023-04-26 23:47:40.827] [INFO ] SET(x, 3) Blocking operation ends on server 2
[2023-04-26 23:47:41.676] [INFO ] Primary server starting an broadcast for GET(x, )
[2023-04-26 23:47:41.677] [INFO ] GET(x, ) finished on server 1
[2023-04-26 23:47:41.684] [INFO ] GET(x, ) finished on server 2
[2023-04-26 23:47:41.696] [INFO ] GET(x, ) finished on server 3
[2023-04-26 23:47:41.835] [INFO ] GET(x) = 3 Blocking operation ends on server 3
```

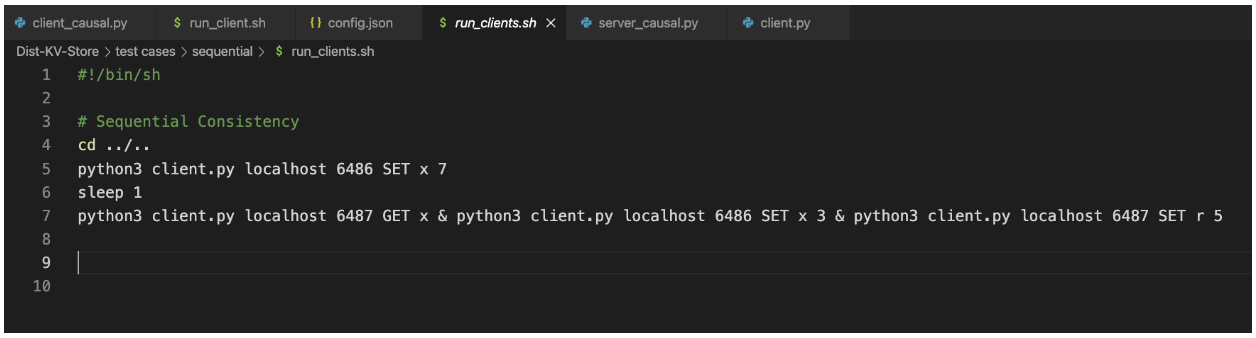
```
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/linearizability$ bash run_client
s.sh
SET request for SET(x, 7) returned and took 0.2040095329284668 time
SET request for SET(x, 1) returned and took 0.20354604721069336 time
SET request for SET(x, 3) returned and took 1.0040442943572998 time
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/linearizability$ GET request for
 variable x returned 3  and took 2.0114848613739014 time
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/linearizability$ 
```

Sequential Consistency:

Servers 1 to 3 are running on 6485 to 6487 ports respectively

Sequence of request:

1) ***SET x 7*** on server 2
2) ***GET x*** on server 3 & ***SET x 3*** on server 2 & ***SET r 5*** on server 3 → Concurrent requests



```
#!/bin/sh

# Sequential Consistency
cd ../..
python3 client.py localhost 6486 SET x 7
sleep 1
python3 client.py localhost 6487 GET x & python3 client.py localhost 6486 SET x 3 & python3 client.py localhost 6487 SET r 5


```

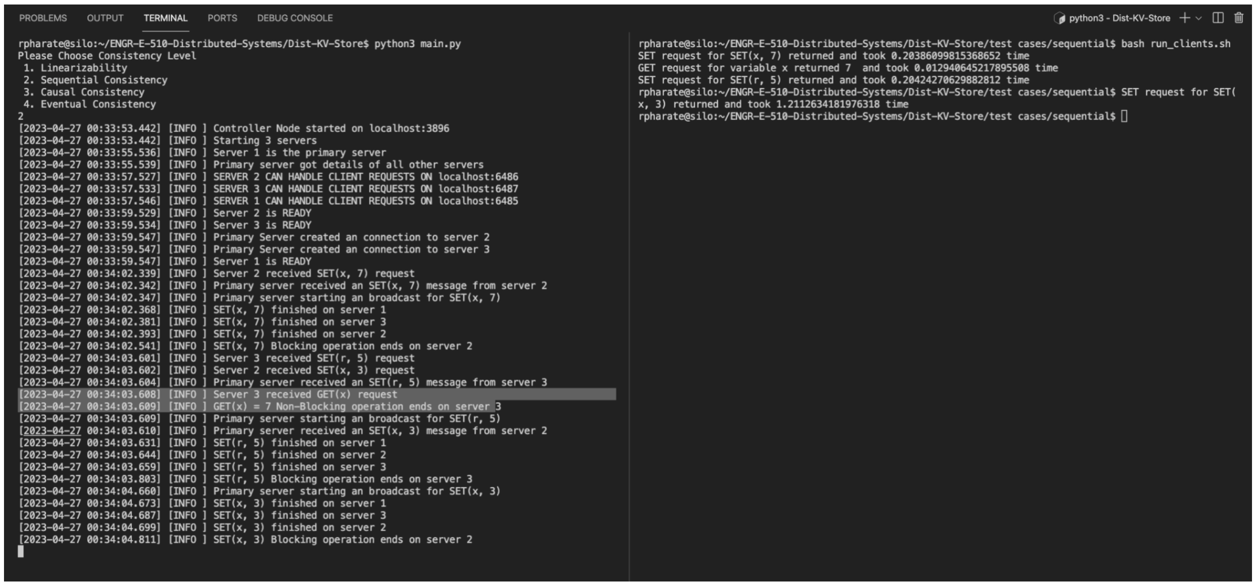
Logs:

```
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store$ python3 main.py
Please Choose Consistency Level
 1. Linearizability
 2. Sequential Consistency
 3. Causal Consistency
 4. Eventual Consistency
2
[2023-04-27 00:33:53.442] [INFO ] Controller Node started on localhost:3896
[2023-04-27 00:33:53.442] [INFO ] Starting 3 servers
[2023-04-27 00:33:55.536] [INFO ] Server 1 is the primary server
[2023-04-27 00:33:55.539] [INFO ] Primary server got details of all other servers
[2023-04-27 00:33:57.527] [INFO ] SERVER 2 CAN HANDLE CLIENT REQUESTS ON
localhost:6486
[2023-04-27 00:33:57.533] [INFO ] SERVER 3 CAN HANDLE CLIENT REQUESTS ON
localhost:6487
[2023-04-27 00:33:57.546] [INFO ] SERVER 1 CAN HANDLE CLIENT REQUESTS ON
localhost:6485
[2023-04-27 00:33:59.529] [INFO ] Server 2 is READY
[2023-04-27 00:33:59.534] [INFO ] Server 3 is READY
[2023-04-27 00:33:59.547] [INFO ] Primary Server created an connection to server 2
[2023-04-27 00:33:59.547] [INFO ] Primary Server created an connection to server 3
[2023-04-27 00:33:59.547] [INFO ] Server 1 is READY
[2023-04-27 00:34:02.339] [INFO ] Server 2 received SET(x, 7) request
[2023-04-27 00:34:02.342] [INFO ] Primary server received an SET(x, 7) message from server 2
[2023-04-27 00:34:02.347] [INFO ] Primary server starting an broadcast for SET(x, 7)
[2023-04-27 00:34:02.368] [INFO ] SET(x, 7) finished on server 1
[2023-04-27 00:34:02.381] [INFO ] SET(x, 7) finished on server 3
[2023-04-27 00:34:02.393] [INFO ] SET(x, 7) finished on server 2
[2023-04-27 00:34:02.541] [INFO ] SET(x, 7) Blocking operation ends on server 2
[2023-04-27 00:34:03.601] [INFO ] Server 3 received SET(r, 5) request
[2023-04-27 00:34:03.602] [INFO ] Server 2 received SET(x, 3) request
[2023-04-27 00:34:03.604] [INFO ] Primary server received an SET(r, 5) message from server 3
[2023-04-27 00:34:03.608] [INFO ] Server 3 received GET(x) request
[2023-04-27 00:34:03.609] [INFO ] GET(x) = 7 Non-Blocking operation ends on server 3
[2023-04-27 00:34:03.609] [INFO ] Primary server starting an broadcast for SET(r, 5)
[2023-04-27 00:34:03.610] [INFO ] Primary server received an SET(x, 3) message from server 2
[2023-04-27 00:34:03.631] [INFO ] SET(r, 5) finished on server 1
[2023-04-27 00:34:03.644] [INFO ] SET(r, 5) finished on server 2
[2023-04-27 00:34:03.659] [INFO ] SET(r, 5) finished on server 3
[2023-04-27 00:34:03.803] [INFO ] SET(r, 5) Blocking operation ends on server 3
[2023-04-27 00:34:04.660] [INFO ] Primary server starting an broadcast for SET(x, 3)
```

[2023-04-27 00:34:04.673] [INFO] SET(x, 3) finished on server 1
[2023-04-27 00:34:04.687] [INFO] SET(x, 3) finished on server 3
[2023-04-27 00:34:04.699] [INFO] SET(x, 3) finished on server 2
[2023-04-27 00:34:04.811] [INFO] SET(x, 3) Blocking operation ends on server 2

Note that *GET x* is a Non-Blocking operation and does not make an broadcast. Its basically and local read and returns immediately. In the above example, *GET x* (server 3), *SET r 5* (server 3) & *SET x 3* (server 2) are concurrent requests. Even though the *SET x 3* request is received prior (few miliseconds difference) before *GET x* request, its on server 2 and the broadcast for that request has not yet received on server 3 and hence the current value of x is 7 on server 3. So, *GET x* will return x=7 immediately.

Logs screenshot:



```
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store$ python3 main.py
Please Choose Consistency Level
 1. Linearizability
 2. Sequential Consistency
 3. Causal Consistency
 4. Eventual Consistency
2
[2023-04-27 00:33:53.442] [INFO ] Controller Node started on localhost:3896
[2023-04-27 00:33:53.442] [INFO ] Starting 3 servers
[2023-04-27 00:33:55.536] [INFO ] Server 1 is the primary server
[2023-04-27 00:33:55.539] [INFO ] Primary server got details of all other servers
[2023-04-27 00:33:57.527] [INFO ] SERVER 2 CAN HANDLE CLIENT REQUESTS ON localhost:6486
[2023-04-27 00:33:57.533] [INFO ] SERVER 3 CAN HANDLE CLIENT REQUESTS ON localhost:6487
[2023-04-27 00:33:57.546] [INFO ] SERVER 1 CAN HANDLE CLIENT REQUESTS ON localhost:6485
[2023-04-27 00:33:59.529] [INFO ] Server 2 is READY
[2023-04-27 00:33:59.534] [INFO ] Server 3 is READY
[2023-04-27 00:33:59.547] [INFO ] Primary Server created an connection to server 2
[2023-04-27 00:33:59.547] [INFO ] Primary Server created an connection to server 3
[2023-04-27 00:33:59.547] [INFO ] Server 1 is READY
[2023-04-27 00:34:02.339] [INFO ] Server 2 received SET(x, 7) request
[2023-04-27 00:34:02.342] [INFO ] Primary server received an SET(x, 7) message from server 2
[2023-04-27 00:34:02.347] [INFO ] Primary server starting an broadcast for SET(x, 7)
[2023-04-27 00:34:02.368] [INFO ] SET(x, 7) finished on server 1
[2023-04-27 00:34:02.381] [INFO ] SET(x, 7) finished on server 3
[2023-04-27 00:34:02.393] [INFO ] SET(x, 7) finished on server 2
[2023-04-27 00:34:02.541] [INFO ] SET(x, 7) Blocking operation ends on server 2
[2023-04-27 00:34:03.601] [INFO ] Server 3 received SET(r, 5) request
[2023-04-27 00:34:03.602] [INFO ] Server 2 received SET(x, 3) request
[2023-04-27 00:34:03.604] [INFO ] Primary server received an SET(r, 5) message from server 3
[2023-04-27 00:34:03.608] [INFO ] Server 3 received GET(x) request
[2023-04-27 00:34:03.609] [INFO ] GET(x) = 7 Non-Blocking operation ends on server 3
[2023-04-27 00:34:03.609] [INFO ] Primary server starting an broadcast for SET(r, 5)
[2023-04-27 00:34:03.610] [INFO ] Primary server received an SET(x, 3) message from server 2
[2023-04-27 00:34:03.631] [INFO ] SET(r, 5) finished on server 1
[2023-04-27 00:34:03.644] [INFO ] SET(r, 5) finished on server 2
[2023-04-27 00:34:03.659] [INFO ] SET(r, 5) finished on server 3
[2023-04-27 00:34:03.803] [INFO ] SET(r, 5) Blocking operation ends on server 3
[2023-04-27 00:34:04.660] [INFO ] Primary server starting an broadcast for SET(x, 3)
[2023-04-27 00:34:04.673] [INFO ] SET(x, 3) finished on server 1
[2023-04-27 00:34:04.687] [INFO ] SET(x, 3) finished on server 3
[2023-04-27 00:34:04.699] [INFO ] SET(x, 3) finished on server 2
[2023-04-27 00:34:04.811] [INFO ] SET(x, 3) Blocking operation ends on server 2
```

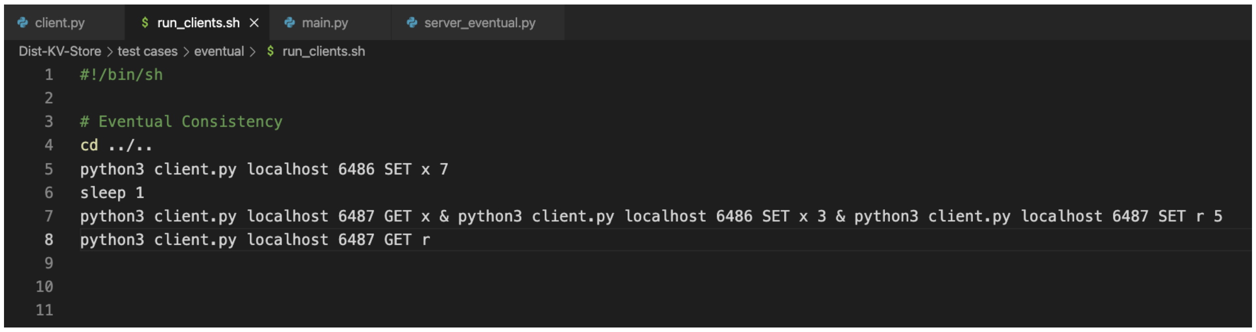
```
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/sequential$ bash run_clients.sh
SET request for SET(x, 7) returned and took 0.20386099815368652 time
GET request for variable x returned 7  and took 0.012940645217895508 time
SET request for SET(r, 5) returned and took 0.20424270629882812 time
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/sequential$ SET request for SET(
x, 3) returned and took 1.2112634181976318 time
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/sequential$ 
```

Eventual Consistency:

Servers 1 to 3 are running on 6485 to 6487 ports respectively

Sequence of requests:

1) ***SET x 7*** on server 2
2) ***GET x*** on server 3 & ***SET x 3*** on server 2 & ***SET r 5*** on server 3 → Concurrent requests
3) ***GET r*** on server 3



```
#!/bin/sh

# Eventual Consistency
cd ../..
python3 client.py localhost 6486 SET x 7
sleep 1
python3 client.py localhost 6487 GET x & python3 client.py localhost 6486 SET x 3 & python3 client.py localhost 6487 SET r 5
python3 client.py localhost 6487 GET r
```

Logs:

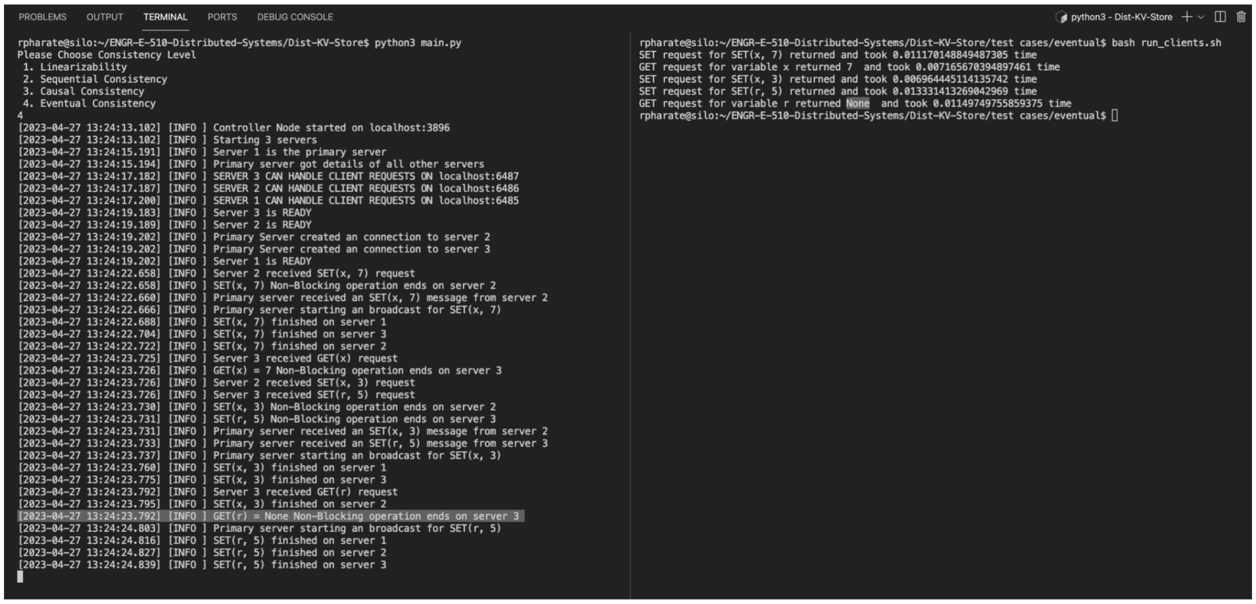
```
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store$ python3 main.py
Please Choose Consistency Level
 1. Linearizability
 2. Sequential Consistency
 3. Causal Consistency
 4. Eventual Consistency
4
[2023-04-27 13:24:13.102] [INFO ] Controller Node started on localhost:3896
[2023-04-27 13:24:13.102] [INFO ] Starting 3 servers
[2023-04-27 13:24:15.191] [INFO ] Server 1 is the primary server
[2023-04-27 13:24:15.194] [INFO ] Primary server got details of all other servers
[2023-04-27 13:24:17.182] [INFO ] SERVER 3 CAN HANDLE CLIENT REQUESTS ON localhost:6487
[2023-04-27 13:24:17.187] [INFO ] SERVER 2 CAN HANDLE CLIENT REQUESTS ON localhost:6486
[2023-04-27 13:24:17.200] [INFO ] SERVER 1 CAN HANDLE CLIENT REQUESTS ON localhost:6485
[2023-04-27 13:24:19.183] [INFO ] Server 3 is READY
[2023-04-27 13:24:19.189] [INFO ] Server 2 is READY
[2023-04-27 13:24:19.202] [INFO ] Primary Server created an connection to server 2
```

```
[2023-04-27 13:24:19.202] [INFO ] Primary Server created an connection to server 3
[2023-04-27 13:24:19.202] [INFO ] Server 1 is READY
[2023-04-27 13:24:22.658] [INFO ] Server 2 received SET(x, 7) request
[2023-04-27 13:24:22.658] [INFO ] SET(x, 7) Non-Blocking operation ends on server 2
[2023-04-27 13:24:22.660] [INFO ] Primary server received an SET(x, 7) message from server 2
[2023-04-27 13:24:22.666] [INFO ] Primary server starting an broadcast for SET(x, 7)
[2023-04-27 13:24:22.688] [INFO ] SET(x, 7) finished on server 1
[2023-04-27 13:24:22.704] [INFO ] SET(x, 7) finished on server 3
[2023-04-27 13:24:22.722] [INFO ] SET(x, 7) finished on server 2
[2023-04-27 13:24:23.725] [INFO ] Server 3 received GET(x) request
[2023-04-27 13:24:23.726] [INFO ] GET(x) = 7 Non-Blocking operation ends on server 3
[2023-04-27 13:24:23.726] [INFO ] Server 2 received SET(x, 3) request
[2023-04-27 13:24:23.726] [INFO ] Server 3 received SET(r, 5) request
[2023-04-27 13:24:23.730] [INFO ] SET(x, 3) Non-Blocking operation ends on server 2
[2023-04-27 13:24:23.731] [INFO ] SET(r, 5) Non-Blocking operation ends on server 3
[2023-04-27 13:24:23.731] [INFO ] Primary server received an SET(x, 3) message from server 2
[2023-04-27 13:24:23.733] [INFO ] Primary server received an SET(r, 5) message from server 3
[2023-04-27 13:24:23.737] [INFO ] Primary server starting an broadcast for SET(x, 3)
[2023-04-27 13:24:23.760] [INFO ] SET(x, 3) finished on server 1
[2023-04-27 13:24:23.775] [INFO ] SET(x, 3) finished on server 3
[2023-04-27 13:24:23.792] [INFO ] Server 3 received GET(r) request
[2023-04-27 13:24:23.795] [INFO ] SET(x, 3) finished on server 2
[2023-04-27 13:24:23.792] [INFO ] GET(r) = None Non-Blocking operation ends on server 3
[2023-04-27 13:24:24.803] [INFO ] Primary server starting an broadcast for SET(r, 5)
[2023-04-27 13:24:24.816] [INFO ] SET(r, 5) finished on server 1
[2023-04-27 13:24:24.827] [INFO ] SET(r, 5) finished on server 2
[2023-04-27 13:24:24.839] [INFO ] SET(r, 5) finished on server 3
```

As you can see in the logs both GET and SET operations are Non-Blocking. GET request immediately return the local value of the variable and SET requests also returns immediately. The broadcast message can starts after the SET request is terminated as you can see in these logs.

Server 3 receives *SET r 5* request and it return immediately before the broadcast. The broadcast for this requests starts later. Now, when another server receives *GET r* requests it returns None immediately (local read) as that variable doesn't exist in any of the server as the broadcast for it has not happened.

Logs screenshot:



```
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store$ python3 main.py
Please Choose Consistency Level
 1. Linearizability
 2. Sequential Consistency
 3. Causal Consistency
 4. Eventual Consistency
4
[2023-04-27 13:24:13.102] [INFO ] Controller Node started on localhost:3896
[2023-04-27 13:24:13.102] [INFO ] Starting 3 servers
[2023-04-27 13:24:15.191] [INFO ] Server 1 is the primary server
[2023-04-27 13:24:15.194] [INFO ] Primary server got details of all other servers
[2023-04-27 13:24:17.182] [INFO ] SERVER 3 CAN HANDLE CLIENT REQUESTS ON localhost:6487
[2023-04-27 13:24:17.187] [INFO ] SERVER 2 CAN HANDLE CLIENT REQUESTS ON localhost:6486
[2023-04-27 13:24:17.200] [INFO ] SERVER 1 CAN HANDLE CLIENT REQUESTS ON localhost:6485
[2023-04-27 13:24:19.183] [INFO ] Server 3 is READY
[2023-04-27 13:24:19.189] [INFO ] Server 2 is READY
[2023-04-27 13:24:19.202] [INFO ] Primary Server created an connection to server 2
[2023-04-27 13:24:19.202] [INFO ] Primary Server created an connection to server 3
[2023-04-27 13:24:19.202] [INFO ] Server 1 is READY
[2023-04-27 13:24:22.658] [INFO ] Server 2 received SET(x, 7) request
[2023-04-27 13:24:22.658] [INFO ] SET(x, 7) Non-Blocking operation ends on server 2
[2023-04-27 13:24:22.660] [INFO ] Primary server received an SET(x, 7) message from server 2
[2023-04-27 13:24:22.666] [INFO ] Primary server starting an broadcast for SET(x, 7)
[2023-04-27 13:24:22.688] [INFO ] SET(x, 7) finished on server 1
[2023-04-27 13:24:22.704] [INFO ] SET(x, 7) finished on server 3
[2023-04-27 13:24:22.722] [INFO ] SET(x, 7) finished on server 2
[2023-04-27 13:24:23.725] [INFO ] Server 3 received GET(x) request
[2023-04-27 13:24:23.726] [INFO ] GET(x) = 7 Non-Blocking operation ends on server 3
[2023-04-27 13:24:23.726] [INFO ] Server 2 received SET(x, 3) request
[2023-04-27 13:24:23.726] [INFO ] Server 3 received SET(r, 5) request
[2023-04-27 13:24:23.730] [INFO ] SET(x, 3) Non-Blocking operation ends on server 2
[2023-04-27 13:24:23.731] [INFO ] SET(r, 5) Non-Blocking operation ends on server 3
[2023-04-27 13:24:23.731] [INFO ] Primary server received an SET(x, 3) message from server 2
[2023-04-27 13:24:23.733] [INFO ] Primary server received an SET(r, 5) message from server 3
[2023-04-27 13:24:23.737] [INFO ] Primary server starting an broadcast for SET(x, 3)
[2023-04-27 13:24:23.760] [INFO ] SET(x, 3) finished on server 1
[2023-04-27 13:24:23.775] [INFO ] SET(x, 3) finished on server 3
[2023-04-27 13:24:23.792] [INFO ] Server 3 received GET(r) request
[2023-04-27 13:24:23.795] [INFO ] SET(x, 3) finished on server 2
[2023-04-27 13:24:23.792] [INFO ] GET(r) = None Non-Blocking operation ends on server 3
[2023-04-27 13:24:24.803] [INFO ] Primary server starting an broadcast for SET(r, 5)
[2023-04-27 13:24:24.816] [INFO ] SET(r, 5) finished on server 1
[2023-04-27 13:24:24.827] [INFO ] SET(r, 5) finished on server 2
[2023-04-27 13:24:24.839] [INFO ] SET(r, 5) finished on server 3
```

```
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/eventual$ bash run_clients.sh
SET request for SET(x, 7) returned and took 0.011170148849487305 time
GET request for variable x returned 7  and took 0.007165670394897461 time
SET request for SET(x, 3) returned and took 0.006964445114135742 time
SET request for SET(r, 5) returned and took 0.013331413269042969 time
GET request for variable r returned None  and took 0.01149749755859375 time
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/eventual$
```

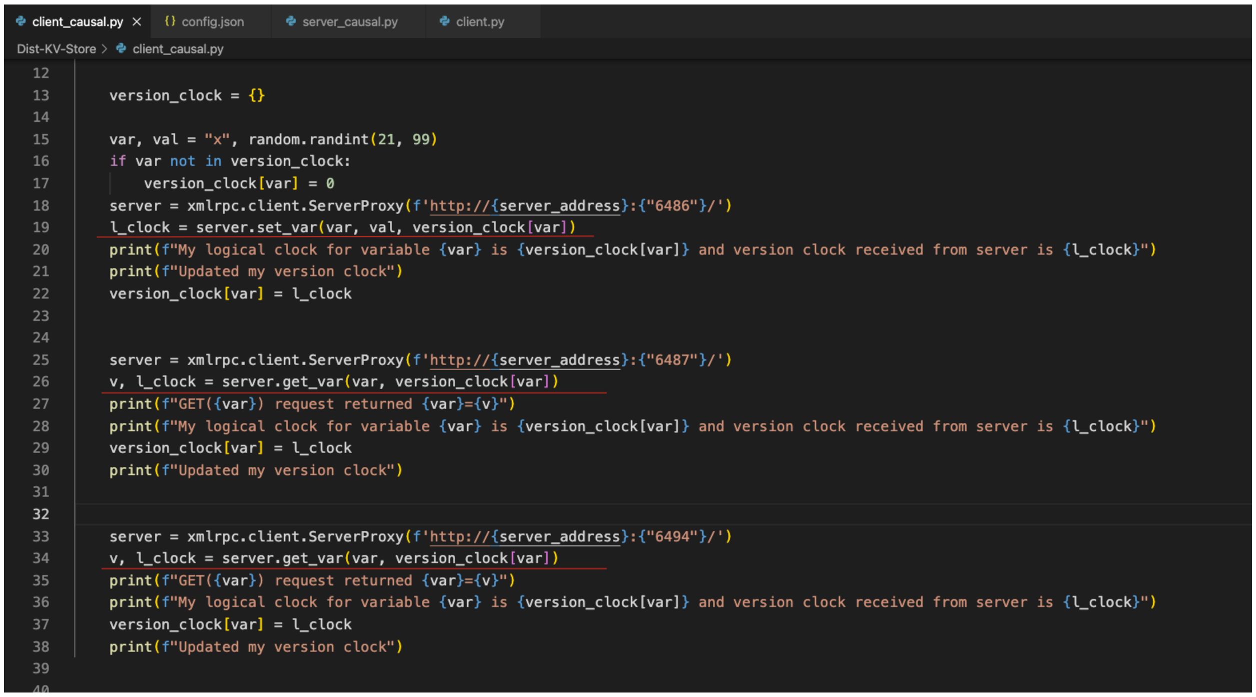
Causal Consistency:

I have used 10 servers for this case. It was necessary to demonstrate the causal consistency behavior. Also, I have used some sleep statement in order to log events as needed and create a scenario that will differentiate between causal consistency and sequential consistency. Also, note that the client is different than the rest of the models implementation as we need to maintain version vectors (logical clocks) on the client side as well.

Servers 1 to 10 are running on 6485 to 6494 ports respectively

Sequence of requests:

4) ***SET x [random value]*** on server 2
5) ***GET x*** on server 3
6) ***GET r*** on server 10
7)



```python

    version_clock = {}

    var, val = "x", random.randint(21, 99)
    if var not in version_clock:
        version_clock[var] = 0
    server = xmlrpc.client.ServerProxy(f'http://{server_address}:{"6486"}/')
    l_clock = server.set_var(var, val, version_clock[var])
    print(f"My logical clock for variable {var} is {version_clock[var]} and version clock received from server is {l_clock}")
    print(f"Updated my version clock")
    version_clock[var] = l_clock


    server = xmlrpc.client.ServerProxy(f'http://{server_address}:{"6487"}/')
    v, l_clock = server.get_var(var, version_clock[var])
    print(f"GET({var}) request returned {var}={v}")
    print(f"My logical clock for variable {var} is {version_clock[var]} and version clock received from server is {l_clock}")
    version_clock[var] = l_clock
    print(f"Updated my version clock")


    server = xmlrpc.client.ServerProxy(f'http://{server_address}:{"6494"}/')
    v, l_clock = server.get_var(var, version_clock[var])
    print(f"GET({var}) request returned {var}={v}")
    print(f"My logical clock for variable {var} is {version_clock[var]} and version clock received from server is {l_clock}")
    version_clock[var] = l_clock
    print(f"Updated my version clock")

```

Logs:

```
rpharate@silo:~/ENGR-E-510-Distributed-Systems$ cd Dist-KV-Store/
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store$ python3 main.py
Please Choose Consistency Level
 1. Linearizability
 2. Sequential Consistency
 3. Causal Consistency
 4. Eventual Consistency
3
[2023-04-29 22:55:56.731] [INFO ] Controller Node started on localhost:3896
[2023-04-29 22:55:56.731] [INFO ] Starting 10 servers
[2023-04-29 22:55:58.889] [INFO ] Server 1 is the primary server
[2023-04-29 22:55:58.915] [INFO ] Primary server got details of all other servers
[2023-04-29 22:56:00.858] [INFO ] SERVER 2 CAN HANDLE CLIENT REQUESTS ON localhost:6486
[2023-04-29 22:56:00.871] [INFO ] SERVER 3 CAN HANDLE CLIENT REQUESTS ON localhost:6487
[2023-04-29 22:56:00.879] [INFO ] SERVER 5 CAN HANDLE CLIENT REQUESTS ON localhost:6489
[2023-04-29 22:56:00.886] [INFO ] SERVER 6 CAN HANDLE CLIENT REQUESTS ON localhost:6490
[2023-04-29 22:56:00.895] [INFO ] SERVER 8 CAN HANDLE CLIENT REQUESTS ON localhost:6492
[2023-04-29 22:56:00.900] [INFO ] SERVER 10 CAN HANDLE CLIENT REQUESTS ON localhost:6494
[2023-04-29 22:56:00.903] [INFO ] SERVER 4 CAN HANDLE CLIENT REQUESTS ON localhost:6488
[2023-04-29 22:56:00.910] [INFO ] SERVER 9 CAN HANDLE CLIENT REQUESTS ON localhost:6493
[2023-04-29 22:56:00.919] [INFO ] SERVER 1 CAN HANDLE CLIENT REQUESTS ON localhost:6485
[2023-04-29 22:56:00.913] [INFO ] SERVER 7 CAN HANDLE CLIENT REQUESTS ON localhost:6491
[2023-04-29 22:56:02.859] [INFO ] Server 2 is READY
[2023-04-29 22:56:02.871] [INFO ] Server 3 is READY
```

```
[2023-04-29 22:56:02.881] [INFO ] Server 5 is READY
[2023-04-29 22:56:02.887] [INFO ] Server 6 is READY
[2023-04-29 22:56:02.897] [INFO ] Server 8 is READY
[2023-04-29 22:56:02.902] [INFO ] Server 10 is READY
[2023-04-29 22:56:02.904] [INFO ] Server 4 is READY
[2023-04-29 22:56:02.912] [INFO ] Server 9 is READY
[2023-04-29 22:56:02.915] [INFO ] Server 7 is READY
[2023-04-29 22:56:02.921] [INFO ] Primary Server created an connection to server 2
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 3
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 4
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 5
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 6
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 7
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 8
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 9
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 10
[2023-04-29 22:56:02.922] [INFO ] Server 1 is READY
[2023-04-29 22:56:46.599] [INFO ] Server 2 received SET(x, 51) request with logical clock value 0
[2023-04-29 22:56:46.599] [INFO ] Current logical clock value for variable x on server 2 is 0
[2023-04-29 22:56:46.602] [INFO ] Primary server received an SET(x, 51) message from server 2
[2023-04-29 22:56:46.617] [INFO ] Primary server starting an broadcast for SET(x, 51)
[2023-04-29 22:56:46.674] [INFO ] SET(x, 51) finished on server 1
[2023-04-29 22:56:46.674] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 1
[2023-04-29 22:56:47.689] [INFO ] SET(x, 51) finished on server 2
[2023-04-29 22:56:47.689] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 2
[2023-04-29 22:56:48.738] [INFO ] SET(x, 51) finished on server 3
[2023-04-29 22:56:48.738] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 3
[2023-04-29 22:56:48.802] [INFO ] SET(x, 51) operation ends on server 2 with logical clock value 1
[2023-04-29 22:56:48.804] [INFO ] Server 3 received GET(x) request with logical clock value 1
[2023-04-29 22:56:48.808] [INFO ] Current logical clock value for variable x on server 3 is 1
[2023-04-29 22:56:48.808] [INFO ] GET(x) = 51 operation ends on server 3 with logical clock value 1
[2023-04-29 22:56:48.818] [INFO ] Server 10 received GET(x) request with logical clock value 1
[2023-04-29 22:56:48.825] [INFO ] Current logical clock value for variable x on server 10 is 0
[2023-04-29 22:56:48.825] [INFO ] Waiting for server 10's logical clock: 0 to catch up to clients
logical clock: 1 for variable x
[2023-04-29 22:56:48.825] [INFO ] Blocking execution of GET(x) request
[2023-04-29 22:56:49.752] [INFO ] SET(x, 51) finished on server 4
[2023-04-29 22:56:49.752] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 4
[2023-04-29 22:56:50.765] [INFO ] SET(x, 51) finished on server 5
[2023-04-29 22:56:50.765] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 5
[2023-04-29 22:56:51.788] [INFO ] SET(x, 51) finished on server 6
[2023-04-29 22:56:51.788] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 6
[2023-04-29 22:56:52.802] [INFO ] SET(x, 51) finished on server 7
[2023-04-29 22:56:52.802] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 7
[2023-04-29 22:56:53.814] [INFO ] SET(x, 51) finished on server 8
[2023-04-29 22:56:53.814] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 8
[2023-04-29 22:56:54.828] [INFO ] SET(x, 51) finished on server 9
[2023-04-29 22:56:54.828] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 9
[2023-04-29 22:56:55.841] [INFO ] SET(x, 51) finished on server 10
```

```
[2023-04-29 22:56:55.841] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 10
[2023-04-29 22:56:56.033] [INFO ] Finally catched up..... Server 10's logical clock: 1, Get requests logical clock: 1 for variable x
[2023-04-29 22:56:56.033] [INFO ] GET(x) = 51 operation ends on server 10 with logical clock value 1
```

As you can see in the logs, first, server 2 receives SET(x,51) request.

- **Servers 2's logical clock is 0 at this moment.**
- **Server 2 forwards this write request to primary**
- **Primary server increments the logical clock for this variable and sends this to server 2**
- **Server 2 updates its counter for variable x and then returns this value to client.**
- **Primary server broadcasts this update along with this new logical clock value to all other servers.**
- **Upon receiving the updated logical clock value from server 2 clients updates its clock for variable x**

Now, we send an GET(x) request to server 3 along with logical clock of 1 (updated from servers response to SET request)

- **Server 3 has already received the broadcast message and hence it immediately return the value of x which is 51**

Now, we send the next GET(x) request to server 10 along with the updated logical clock value.

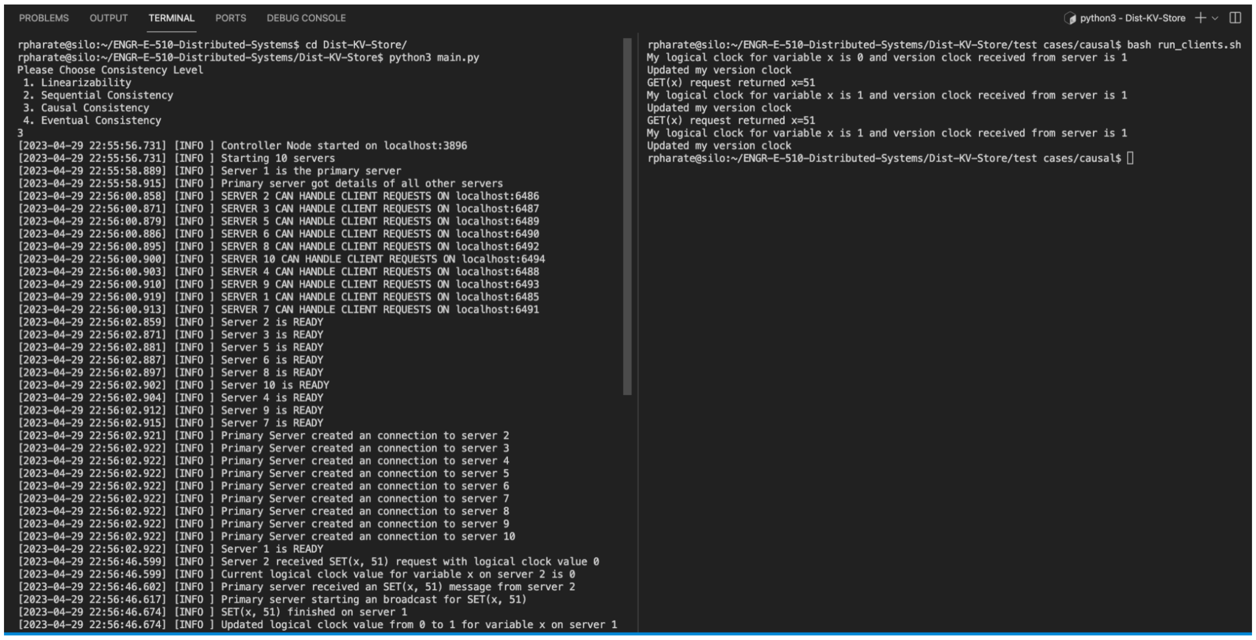
- **When this GET request comes to server 10, it does not process it and waits as its own logical clock for variable x is at 0.**
- **The GET request comes with logical clock of 1 which is from future.**
- **The logical clock of server 10 is at 0 because the broadcast for SET(x,51) has not yet reached.**
- **When server 10 receives this broadcast message it updates its logical clock for variable x to 1 and now can process the GET(x) request.**

As, you can see in the logs, this Blocking behavior is highlighted in yellow.

Comparison with sequential consistency:

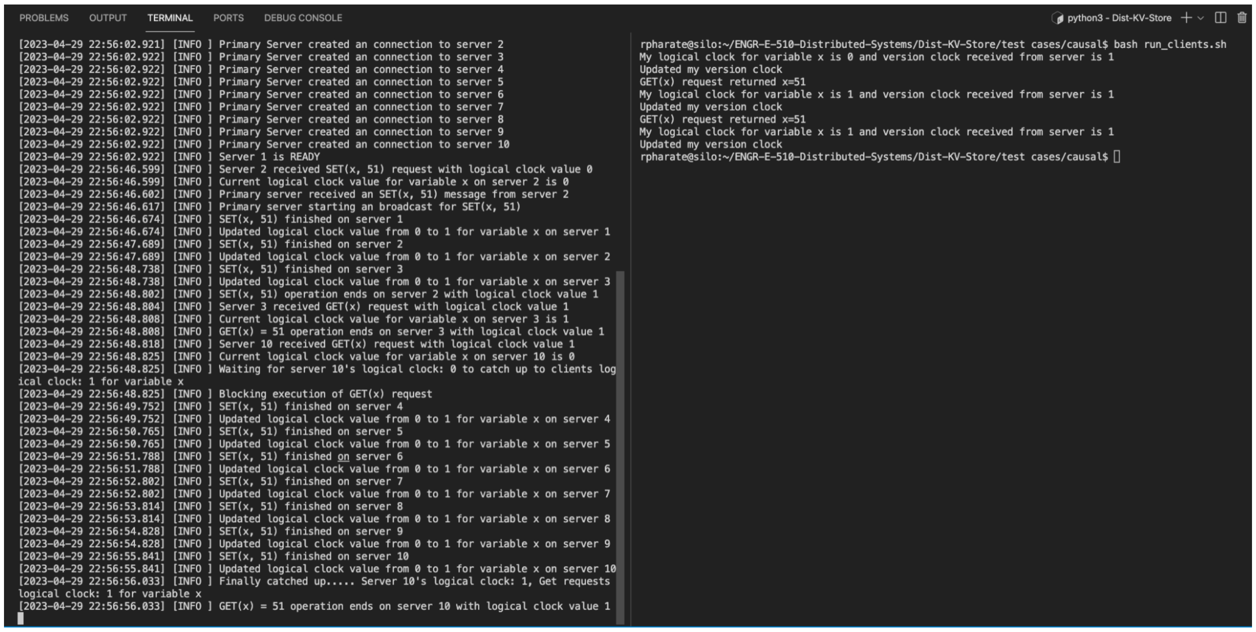
- In sequential consistency, the write operations are completely blocking meaning it will not return until all the servers have the same copy of the variable.
- So, in this case, the partial broadcast condition will never occur where causal consistency is effective.

Logs screenshot:



```
rpharate@silo:~/ENGR-E-510-Distributed-Systems$ cd Dist-KV-Store/
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store$ python3 main.py
Please Choose Consistency Level
 1. Linearizability
 2. Sequential Consistency
 3. Causal Consistency
 4. Eventual Consistency
3
[2023-04-29 22:55:56.731] [INFO ] Controller Node started on localhost:3896
[2023-04-29 22:55:56.731] [INFO ] Starting 10 servers
[2023-04-29 22:55:58.889] [INFO ] Server 1 is the primary server
[2023-04-29 22:55:58.915] [INFO ] Primary server got details of all other servers
[2023-04-29 22:56:00.858] [INFO ] SERVER 2 CAN HANDLE CLIENT REQUESTS ON localhost:6486
[2023-04-29 22:56:00.871] [INFO ] SERVER 3 CAN HANDLE CLIENT REQUESTS ON localhost:6487
[2023-04-29 22:56:00.879] [INFO ] SERVER 5 CAN HANDLE CLIENT REQUESTS ON localhost:6489
[2023-04-29 22:56:00.886] [INFO ] SERVER 6 CAN HANDLE CLIENT REQUESTS ON localhost:6490
[2023-04-29 22:56:00.895] [INFO ] SERVER 8 CAN HANDLE CLIENT REQUESTS ON localhost:6492
[2023-04-29 22:56:00.900] [INFO ] SERVER 10 CAN HANDLE CLIENT REQUESTS ON localhost:6494
[2023-04-29 22:56:00.903] [INFO ] SERVER 4 CAN HANDLE CLIENT REQUESTS ON localhost:6488
[2023-04-29 22:56:00.910] [INFO ] SERVER 9 CAN HANDLE CLIENT REQUESTS ON localhost:6493
[2023-04-29 22:56:00.919] [INFO ] SERVER 1 CAN HANDLE CLIENT REQUESTS ON localhost:6485
[2023-04-29 22:56:00.913] [INFO ] SERVER 7 CAN HANDLE CLIENT REQUESTS ON localhost:6491
[2023-04-29 22:56:02.859] [INFO ] Server 2 is READY
[2023-04-29 22:56:02.871] [INFO ] Server 3 is READY
[2023-04-29 22:56:02.881] [INFO ] Server 5 is READY
[2023-04-29 22:56:02.887] [INFO ] Server 6 is READY
[2023-04-29 22:56:02.897] [INFO ] Server 8 is READY
[2023-04-29 22:56:02.902] [INFO ] Server 10 is READY
[2023-04-29 22:56:02.904] [INFO ] Server 4 is READY
[2023-04-29 22:56:02.912] [INFO ] Server 9 is READY
[2023-04-29 22:56:02.915] [INFO ] Server 7 is READY
[2023-04-29 22:56:02.921] [INFO ] Primary Server created an connection to server 2
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 3
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 4
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 5
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 6
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 7
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 8
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 9
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 10
[2023-04-29 22:56:02.922] [INFO ] Server 1 is READY
[2023-04-29 22:56:46.599] [INFO ] Server 2 received SET(x, 51) request with logical clock value 0
[2023-04-29 22:56:46.599] [INFO ] Current logical clock value for variable x on server 2 is 0
[2023-04-29 22:56:46.602] [INFO ] Primary server received an SET(x, 51) message from server 2
[2023-04-29 22:56:46.617] [INFO ] Primary server starting an broadcast for SET(x, 51)
[2023-04-29 22:56:46.674] [INFO ] SET(x, 51) finished on server 1
[2023-04-29 22:56:46.674] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 1
```

```
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/causal$ bash run_clients.sh
My logical clock for variable x is 0 and version clock received from server is 1
Updated my version clock
GET(x) request returned x=51
My logical clock for variable x is 1 and version clock received from server is 1
Updated my version clock
GET(x) request returned x=51
My logical clock for variable x is 1 and version clock received from server is 1
Updated my version clock
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/causal$ 
```



```
[2023-04-29 22:56:02.921] [INFO ] Primary Server created an connection to server 2
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 3
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 4
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 5
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 6
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 7
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 8
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 9
[2023-04-29 22:56:02.922] [INFO ] Primary Server created an connection to server 10
[2023-04-29 22:56:02.922] [INFO ] Server 1 is READY
[2023-04-29 22:56:46.599] [INFO ] Server 2 received SET(x, 51) request with logical clock value 0
[2023-04-29 22:56:46.599] [INFO ] Current logical clock value for variable x on server 2 is 0
[2023-04-29 22:56:46.602] [INFO ] Primary server received an SET(x, 51) message from server 2
[2023-04-29 22:56:46.617] [INFO ] Primary server starting an broadcast for SET(x, 51)
[2023-04-29 22:56:46.674] [INFO ] SET(x, 51) finished on server 1
[2023-04-29 22:56:46.674] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 1
[2023-04-29 22:56:47.689] [INFO ] SET(x, 51) finished on server 2
[2023-04-29 22:56:47.689] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 2
[2023-04-29 22:56:48.738] [INFO ] SET(x, 51) finished on server 3
[2023-04-29 22:56:48.738] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 3
[2023-04-29 22:56:48.802] [INFO ] SET(x, 51) operation ends on server 2 with logical clock value 1
[2023-04-29 22:56:48.804] [INFO ] Server 3 received GET(x) request with logical clock value 1
[2023-04-29 22:56:48.808] [INFO ] Current logical clock value for variable x on server 3 is 1
[2023-04-29 22:56:48.808] [INFO ] GET(x) = 51 operation ends on server 3 with logical clock value 1
[2023-04-29 22:56:48.818] [INFO ] Server 10 received GET(x) request with logical clock value 1
[2023-04-29 22:56:48.825] [INFO ] Current logical clock value for variable x on server 10 is 0
[2023-04-29 22:56:48.825] [INFO ] Waiting for server 10's logical clock: 0 to catch up to clients log
ical clock: 1 for variable x
[2023-04-29 22:56:48.825] [INFO ] Blocking execution of GET(x) request
[2023-04-29 22:56:49.752] [INFO ] SET(x, 51) finished on server 4
[2023-04-29 22:56:49.752] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 4
[2023-04-29 22:56:50.765] [INFO ] SET(x, 51) finished on server 5
[2023-04-29 22:56:50.765] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 5
[2023-04-29 22:56:51.788] [INFO ] SET(x, 51) finished on server 6
[2023-04-29 22:56:51.788] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 6
[2023-04-29 22:56:52.802] [INFO ] SET(x, 51) finished on server 7
[2023-04-29 22:56:52.802] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 7
[2023-04-29 22:56:53.814] [INFO ] SET(x, 51) finished on server 8
[2023-04-29 22:56:53.814] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 8
[2023-04-29 22:56:54.828] [INFO ] SET(x, 51) finished on server 9
[2023-04-29 22:56:54.828] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 9
[2023-04-29 22:56:55.841] [INFO ] SET(x, 51) finished on server 10
[2023-04-29 22:56:55.841] [INFO ] Updated logical clock value from 0 to 1 for variable x on server 10
[2023-04-29 22:56:56.033] [INFO ] Finally catched up..... Server 10's logical clock: 1, Get requests 
logical clock: 1 for variable x
[2023-04-29 22:56:56.033] [INFO ] GET(x) = 51 operation ends on server 10 with logical clock value 1
```

```
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/causal$ bash run_clients.sh
My logical clock for variable x is 0 and version clock received from server is 1
Updated my version clock
GET(x) request returned x=51
My logical clock for variable x is 1 and version clock received from server is 1
Updated my version clock
GET(x) request returned x=51
My logical clock for variable x is 1 and version clock received from server is 1
Updated my version clock
rpharate@silo:~/ENGR-E-510-Distributed-Systems/Dist-KV-Store/test cases/causal$ 
```