

Com S 454/554 Fall 2020 Assignment 5

Requirement

How the data store servers work

You are required to develop a distributed data store replicated at one primary server and multiple backup servers. For simplicity, the data store maintains just one integer variable with initial value of 0; the servers will run on the same computer as different processes.

The primary server

When the primary server starts, it takes one integer argument: **PrimaryPort**. Then, it (1) sets up an integer variable (i.e., the primary replica of the data store), (2) creates a TCP server socket with port **PrimaryPort**, and (3) waits at the port for requests from clients or backup servers. It needs to handle the following types of requests coming to the port:

- **READ** request from client – When receiving this request, it should return the current value of its replica of the data store variable.
- **WRITE:newValue** request from client – When receiving this request, it should (1) update its own replica of the data store to **newValue**, (2) request each of the backup server to update its replica of the data store and get acknowledgement from it, and then (3) reply to the requesting client. Note that, if multiple WRITE requests are received, they should be executed sequentially for sequential consistency.
- **JOIN:backupPort** request from a backup server – When receiving this request, it should (1) record the requesting backup server's server socket port number (i.e., **backupPort**), and then (2) send acknowledgement to the requesting backup server.
- **UPDATE:newValue** request from a backup server -- When receiving this request, it should first act the same as receiving **WRITE:newValue** to have all the data store replicas to update to **newValue**. Then, it acknowledges the requesting backup server of the completion of update. Note that, if multiple WRITE/UPDATE requests are received, they should be executed sequentially for sequential consistency.

Each backup server

When a backup server starts, it takes two integer argument: **BackupPort** and **PrimaryPort**. Then, it (1) sets up an integer variable (i.e., a backup replica of the data store), (2) sends a **JOIN:BackupPort** request to the primary server's server socket at port **PrimaryPort**, creates a TCP server socket at port **BackupPort**, and (3) waits at port **BackupPort** for requests from clients or the primary server. It needs to handle the following types of requests coming to the port:

- **READ** request from client – When receiving this request, it should return the current value of its replica of the data store variable.
- **WRITE:newValue** request from client – When receiving this request, it should send **UPDATE:newValue** request to the primary server, and waits for its acknowledgement. Then, it sends acknowledge to the requesting client. Note that, when multiple WRITE requests are received, they should be executed sequentially for sequential consistency.

- **UPDATE:newValue** request from the primary server -- When receiving this request, it should update its replica of data store to newValue and then reply acknowledgement to the primary server.

The Client

A client program name Client.java is provided (find it at the attachment) to provide a command line interface to interact with the above servers. Once started, it accepts and passes commands you input to the servers accordingly. Examples:

- If you type command:
5000 READ
The client sends request **READ** to the server at port 5000, gets it executed, and prints the returned value.
- If you type command:
5000 WRITE:10
The client sends request **WRITE:10** to the server at port 5000, gets it executed, and prints the acknowledgement from the server.

Note that, you can run multiple instances of the program to act as multiple clients sending commands concurrently.

Environment setup

Can run program by Eclipse.

Summary

Program is fully functional and multiple clients are tested.

The main differences between backup server and primary server are:

- Different operation when receive "WRITE" and "UPDATE" command.
- Backup server needs to send its port number to primary server and primary server needs to record all the backup servers for "UPDATE" function.

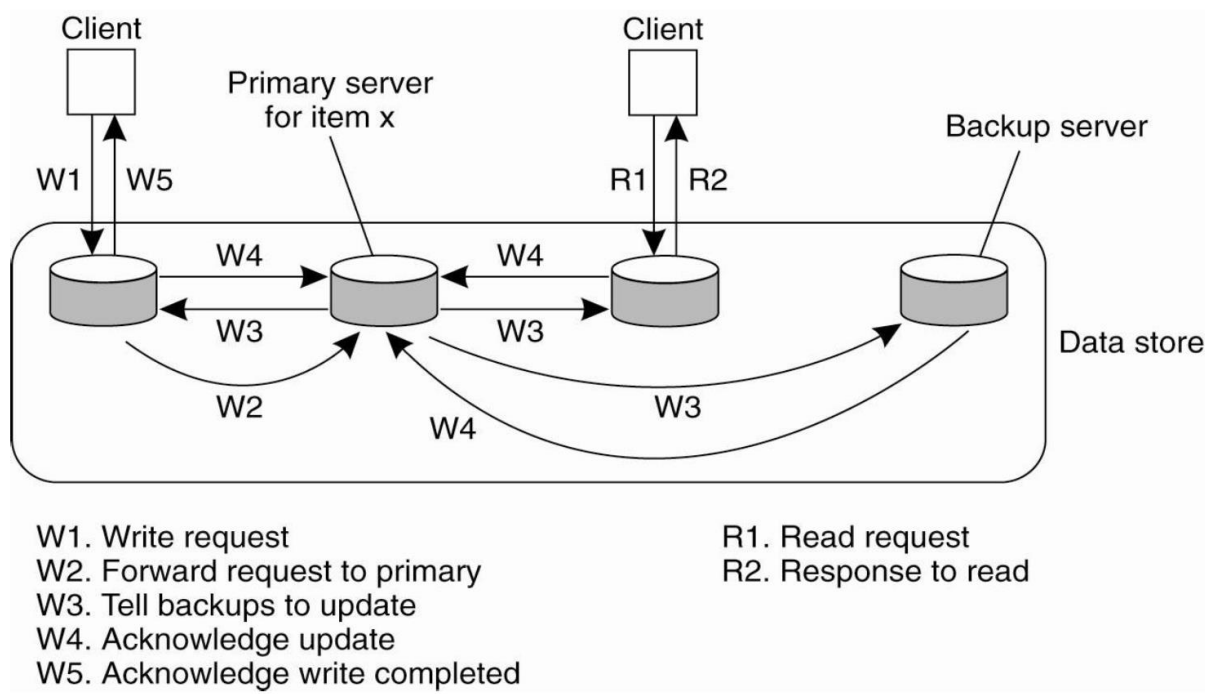
Difficulties 1: in the server side, the incoming message can come from client, primary server, or other backup server. The message identity is not clear. However, the command is clear enough to make execution. We don't have to know where the message from, the only thing matters is that we can always reply back to the sender through TCP. Since as long as the TCP channel is built by socket, the communication is two-way.

Difficulties 2: for the backup server, when it gets command "WRITE", it only needs to send "UPDATE" message to the primary server. For the primary server, it needs to broadcast to all the backup servers when it receives "WRITE" or "UPDATE" command. Since we cannot make sure the identity of the sender,

we simply broadcast “UPDATE” messages to all recorded backup server. Arraylist is used to keep track of all confirmed backup servers’ port.

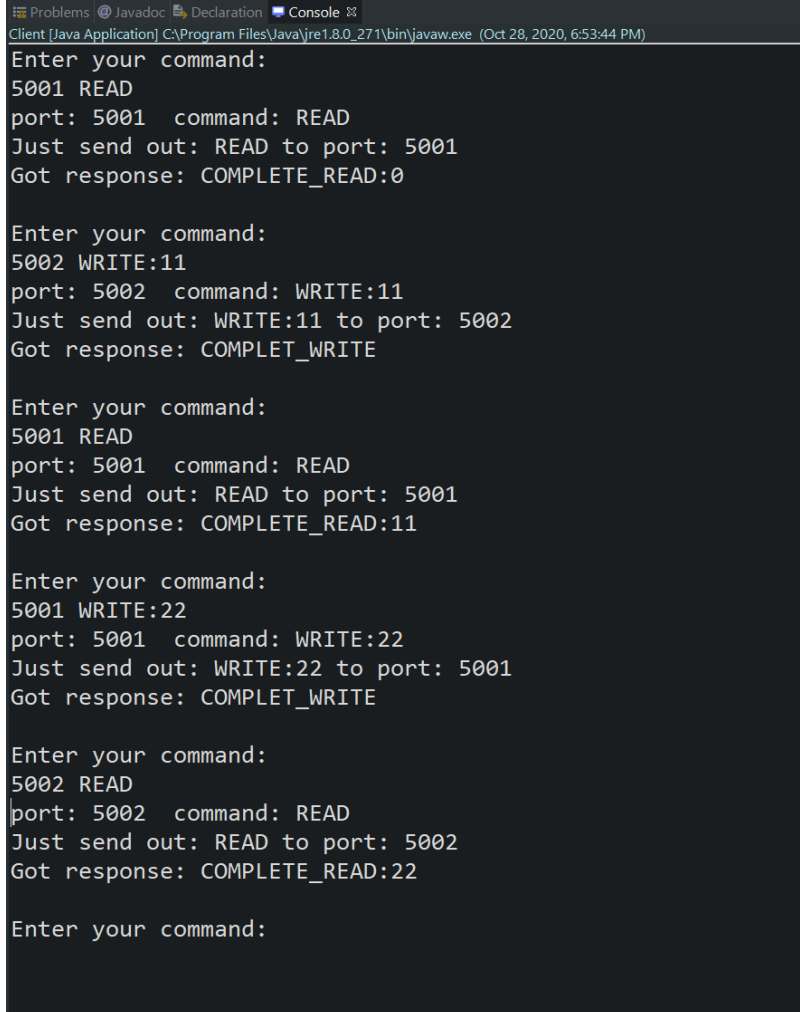
Difficulties 3: it is tricky sometimes that we have to make sure primary server responds to backup server immediately before broadcasting to other backup servers to avoid the situation that two servers are waiting for each other then the program gets stuck. This is the side effect of that server does not the identity of the sender.

Remote-Write Protocols



Screenshot

Client console:



```
Problems Javadoc Declaration Console
Client [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Oct 28, 2020, 6:53:44 PM)
Enter your command:
5001 READ
port: 5001 command: READ
Just send out: READ to port: 5001
Got response: COMPLETE_READ:0

Enter your command:
5002 WRITE:11
port: 5002 command: WRITE:11
Just send out: WRITE:11 to port: 5002
Got response: COMPLET_WRITE

Enter your command:
5001 READ
port: 5001 command: READ
Just send out: READ to port: 5001
Got response: COMPLETE_READ:11

Enter your command:
5001 WRITE:22
port: 5001 command: WRITE:22
Just send out: WRITE:22 to port: 5001
Got response: COMPLET_WRITE

Enter your command:
5002 READ
port: 5002 command: READ
Just send out: READ to port: 5002
Got response: COMPLETE_READ:22

Enter your command:
```

Primary server:

```
Problems Javadoc Declaration Console
PrimaryServer [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Oct 28, 2020, 6:53:06 PM)
Enter primary port number:5000
Waiting for connetion .....

Got a new request
The request is: JOIN:backupPort 5001
JOIN:backupPort operation done! and port is: 5001

Got a new request
The request is: JOIN:backupPort 5002
JOIN:backupPort operation done! and port is: 5002

Got a new request
The request is: UPDATE:11
Just send out: UPDATE:11 to port: 5001
Got response: COMPLET_UPDATE

Just send out: UPDATE:11 to port: 5002
Got response: COMPLET_UPDATE

command operation done!, new value is: 11

Got a new request
The request is: UPDATE:22
Just send out: UPDATE:22 to port: 5001
Got response: COMPLET_UPDATE

Just send out: UPDATE:22 to port: 5002
Got response: COMPLET_UPDATE

command operation done!, new value is: 22
```

Backup server 1:

```
Problems Javadoc Declaration Console
BackupServer [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Oct 28, 2020, 6:53:11 PM)
Enter this backup server port number followed by primary server port number: 5001 5000
Just send out: JOIN: 5001 to port: 5000
Got acknowledgement: COMPLETE_JOIN
Backup Server is listening on port..... 5001

Got a new request
The request is: READ
READ operation done!

Got a new request
The request is: UPDATE:11
command operation done!, new value is: 11

Got a new request
The request is: READ
READ operation done!

Got a new request
The request is: WRITE:22
Just send out: UPDATE:22 to port: 5000
Got response: COMPLET_UPDATE

WRITE operation done!, new value is: 22

Got a new request
The request is: UPDATE:22
command operation done!, new value is: 22
```

Backup server 2:

```
Problems Javadoc Declaration Console
BackupServer [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe (Oct 28, 2020, 6:53:19 PM)
Enter this backup server port number followed by primary server port number: 5002 5000
Just send out: JOIN: 5002 to port: 5000
Got acknowledgement: COMPLETE_JOIN
Backup Server is listening on port..... 5002
|
Got a new request
The request is: WRITE:11
Just send out: UPDATE:11 to port: 5000
Got response: COMPLET_UPDATE

WRITE operation done!, new value is: 11

Got a new request
The request is: UPDATE:11
command operation done!, new value is: 11

Got a new request
The request is: UPDATE:22
command operation done!, new value is: 22

Got a new request
The request is: READ
READ operation done!
```