CSCI 5105

Introduction to Distributed Systems

Spring - 2020

Design Document

Bulletin Board

Date: 04/05/2020

Team:

Soumya Agrawal (agraw184)
Garima Mehta (mehta250)
Bhaargav Sriraman (srira048)

TABLE OF CONTENTS

Project Implementation:

The aim of the project was to develop a bulletin board wherein a client can post/reply/read/choose articles. Our implementation consists of three packages: Client, Server and Coordinator. The github repo link: https://github.umn.edu/agraw184/BullletinBoard

Client:

The client side functionality is implemented in the package named, main. It has Client.java and ClientHelper.java as main files. Client.java has the functionality for the client to connect with the Server to carry out several functionalities that include, Post (A new article), Read (List of existing articles), Choose (An article to reply to an article), Reply (To an existing article). The client creates a Socket (TCP Connection) using the class SocketConnection.java which also creates ObjectInputStream, ObjectOutputStream for the same socket. ClientHelper.java has helper functions to facilitate processing of the message sent from client, the connection to the server (Sending and receiving messages), and displaying the list of articles when the client chooses to read the articles

Server:

The server side functionality is implemented in the Server package, with three core files, Server.java, ClientResponder.java, and ServerHelper.java. This package helps in connecting the Client to Server and Server to Coordinator (primary server). Server.java creates two sockets, one for Coordinator and the other for Server itself. Whenever a new client comes to a server, it creates a thread for its functionality using the ClientResponder class, so that it does not block the I/O and queues further requests from either Client or Coordinator. The first message received by the server is from the Coordinator, which sends the type of consistency to be used for the run of the program. The server checks the type of consistency and then runs specific methods from the ClientResponder class. ServerHelper. Java facilitates the sending and receiving of the messages from Client and Coordinator, both. It also helps in synching all the servers in Quorum consistency, sending the desired response from server to client depending on the functionality the client has chosen.

Coordinator:

The coordinator is essentially a server which acts as a primary server. It is a part of the Coordinator package which includes, Coordinator.java, ServerResponder.java, CoordinateHelper.java. Coordinator.java is the point of entry for the primary server and it asks

the user about the consistency that is to be used for the particular run of the program. Once the consistency is established by the user, it sends the type of consistency to all the servers as well and starts the ServerResponder, which has a thread for each of the servers so that it does not block the I/O. The ServerResponder calls specific methods depending on the type of consistency. CoordinatorHelper.java class helps in establishing the relationship between server and coordinator by sending, receiving and processing the requests received from the server. It also helps in determining the read and write servers for Quorum Consistency.

Serverlist.properties file has the IP Address and port number for all the servers and coordinator. It also has the

How to run:

This project is built to run in localhost with multiple ports

1. Run the make command to compile java files
2. Now, first we run the coordinator,using java Coordinator
3. Run the servers using, java Server <port>
   Ports can be found in serverList.properties, where server is running
   Note: We would suggest you to start the servers in the same order as present in the serverList.properties.
4. Run the clients using java Client <port>, here the port is where the server is running and client can connect to that server.
5. Run the TestClient to see the functionality of the operations that can be performed by the clients.

About the Consistencies:

1. **Sequential Consistency**:

   For sequential consistency, we are using the primary-backup protocol.
   a. POST/REPLY → Whenever the client chooses to post a new article or reply to an existing article:
      i. The client takes the Post/Reply request to the Server.
      ii. The server then contacts the Coordinator and gets the ID for the article, and then updates the dependencyList/ArticleList for itself.
      iii. Once the local update for Coordinator is complete, it broadcasts the changes to all the servers.
      iv. The server updates the local dependencyList/ArticleList and then the blocked I/O for client is opened, and client is able to choose the next set of actions.

b. READ → Whenever the client chooses to get the list of articles:
    i. The client takes the Read request to the server.
    ii. The server reads the data from its local copy of ArticleList/DependencyList and returns the list of articles in the required format (Post as a new article, Reply to the post with a tab below the specified parent of the reply)
c. CHOOSE → Whenever the client chooses to select a particular article to read:
    i. The client takes the Choose request to the server.
    ii. The server reads the data from its local copy of ArticleList and returns the particular article to the client.

2. Quorum Consistency:

For Quorum Consistency, we have to choose the number of read and write servers from a given pool of servers. It should adhere to the 2 conditions to satisfy the number of read and number of write servers, the sum of read and write servers should exceed the total number of servers and the number of write servers should be greater than the half of total number of servers. The number of read and write servers is specified in the ServerList.properties file.
After an interval of 15 seconds, a Sync thread is run, which is started at the beginning of the run of the program for Quorum.

a. POST/REPLY → Whenever the client chooses to post a new article or reply to an existing article:
    i. The client takes the Post/Reply request to the Server.
    ii. The Server then connects with the Coordinator to get the ID for the article.
    iii. Once the ID is established, the Coordinator takes the list of write servers and broadcasts the message to all the servers.
    iv. It uses the Sync function to periodically update all the servers in Quorum by getting the local copy from the servers and then passing the delta (difference/discrepancy in the local articleList/dependencyList of the servers) to the servers.
b. READ → Whenever the client chooses to get the list of articles:
    i. The client takes the Read request to the Server.
    ii. The Server then starts the sync process again by using the list of read servers.
    iii. It gets the latest ID from the coordinator then finds the differences in the local copies of the servers and updates the global map.
    iv. It gets the entire list of articles and dependencies from the global map and sends it back to the client.

c. CHOOSE → Whenever the client chooses to read a particular article:
    i. The client takes the request for choose to the Server.
    ii. The server then starts the sync process by using the list of read servers.
    iii. It gets the latest ID from the coordinator and then finds the differences in the local copies of the servers and updates the global map.
    iv. Once the global map is updated, it chooses the article ID given by the client from the global article list and then sends that article to the client.

3. Read Your Write Consistency:

The read your write consistency is implemented using the local protocol, wherein if a server writes in one of the servers and then connects to another server, it can see the changes made to the

a. POST/REPLY → Whenever the client chooses to post a new article or reply to an existing article:
    i. The client takes the Post/Reply request to the Server.
    ii. The Server then connects with the Coordinator to get the ID for the article.
    iii. Once the local update for Coordinator is complete, it broadcasts the changes to all the servers.
    iv. The server updates the local dependencyList/ArticleList and then the blocked I/O for client is opened, and client is able to choose the next set of actions.
b. READ → Whenever the client chooses to get the list of articles:
    i. The client takes the Read request to the Server.
    ii. The Server then starts the sync process by using the list of servers.
    iii. It gets the latest ID from the coordinator then finds the differences in the local copies of the servers and updates the global map.
    iv. It gets the entire list of articles and dependencies from the global map and sends it back to the client.
c. CHOOSE → Whenever the client chooses to read a particular article:
    i. The client takes the request for choose to the Server.
    ii. The server then starts the sync process by using the list of read servers.
    iii. It gets the latest ID from the coordinator and then finds the differences in the local copies of the servers and updates the global map.
    iv. Once the global map is updated, it chooses the article ID given by the client from the global article list and then sends that article to the client.

About the Classes:

1. Server:

   This is the base class for Server where the threads for ServerSocket and CoordinatorSocket are started.

2. ServerHelper:

   This is the helper class for communication between Client and Servers and Server and Coordinator.

3. QuorumSyncThread:

   This is the thread for syncing the local databases of all the servers.

4. ClientResponder:

   This class creates a thread for each of the clients that join and calls specific methods depending on the type of Consistency.

5. Coordinator:

   This is the base class for Coordinator where it asks the user for the type of consistency that needs to be implemented for the current run of the program.

6. ServerResponder:

   This class creates a thread for each of the servers that connects with the coordinator and calls specific methods depending on the type of consistency.

7. CoordinatorHelper:

   This is the helper class for Coordinator to receive, process and send messages to the Server. It also validates the read and write servers for Quorum and then curates the list randomly.

8. Consistency:

   This is an enum which has all the consistency types

9. Client:

   The is the main class to start a client and runs the terminal UI.

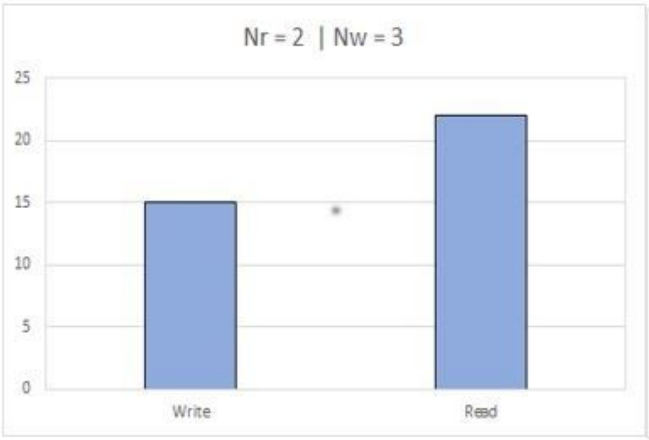
10. ClientHelper:

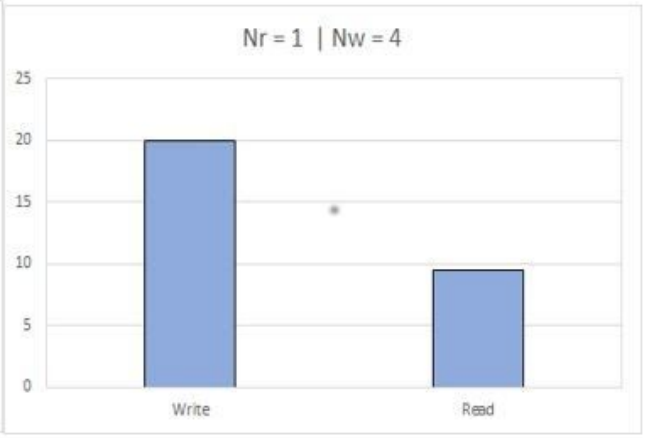
    All the helper methods are present in this class that are used by the Client.

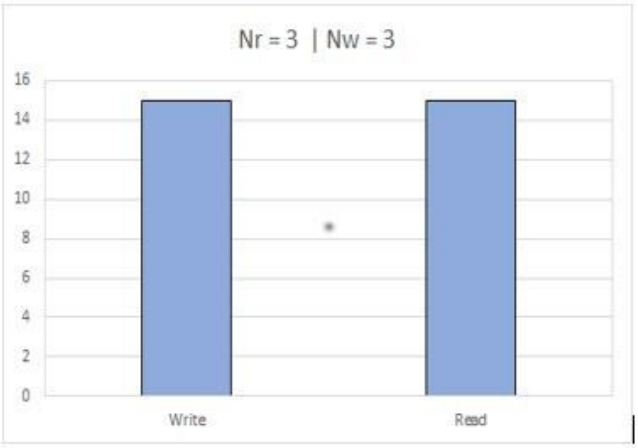
11. SocketConnection:

    This is the class which binds TCP connection between different entities.

Analysis:







We ran the Bulletin Board System using 3 different combinations of Nr (Number of Read Servers) and Nw (Number of Write Servers) using Quorum Consistency. The results were as expected. The time take to read from Nr = 1 is the least (9.5 secs) and the time taken to read from Nr = 3 is the maximum. Similarly, the time taken to write from Nw = 4 is maximum (20 secs) and the time taken from Nw = 3 is lesser but not significantly less.