Distributed Library Management System using Java RMI

Amodh Akhelikar
Computer Engineering
Sardar Patel Institute of Technology
Email: amodh.akhelikar@spit.ac.in

Amandeep
Computer Engineering
Sardar Patel Institute of Technology
Email: amandeep.somdutt@spit.ac.in

Saumitra Chaskar
Computer Engineering
Sardar Patel Institute of Technology
Email: saumitra.chaskar@spit.ac.in

I. Introduction

Even in the age of the Internet and digital education, the big, old and dusty libraries still stand tall as the Parthenon of Knowledge. While you can fit hundreds and ten thousands of books on a computer hard drive but the libraries are sometimes constrained by the physical space or they are limited to a specific subject matter. Even though these libraries are limited individually and geographically distant, management, working and user interactions happening in this network of libraries can be handled through a single system with its managing components distributed all over the nodes of this network communicating with each other. Distributed systems in the field of computing are ideal for such kinds of coordinated and decentralized use cases. The application that we propose to build is one such Distributed Library Management System which will allow the library users and managers to access all kinds of literature not only from their local library but all the other libraries that are associated or networked with it. The users will be able to search the books, issue and return the book, while the library managers can add or remove a book and list availability of all the books in the library. The system will be based on the RMI (Remote Method Invocation) mechanism particularly its Java based implementation.

II. Literature Search

A Library management system is an application that is used to maintain the records of books in a certain library and make the load on the librarian lighter by providing an easier method to record the data about the books. A distributed library system called BRICKS [1] is a peer-to-peer architecture based system. Here they haven't talked about the method for communication being used and is more focused on the different types of simple and advanced queries the system can handle. To ensure the security of the library data such as data about the books [3] provides a blockchain solution. This solution adds an extra overhead on the processing time when considered for use in a distributed environment as more computation resources will be required. [2] talks about using a cloud computing approach where the servers will be hosted on enterprise cloud with a pay-per-use model. This increases our dependence on an external provider and increases the cost as the system scales. In distributed systems, Java RMI can be used as it provides an object-to-object communication method. The remote systems can invoke the methods located in these objects. [4] and [5] provides a detailed and well documented solution to use Java RMIs in a system.

III. Problem Statement

Managing many libraries which are geographically distant is very difficult for a library manager. Also the users face similar problems when they wish to access literature from libraries other than their local or college libraries. Facilitating access to these libraries by networking these libraries and providing a uniform interface for access to both the library managers and the users across these libraries.

IV. Methodology

The Distributed Library Management System is designed using the principle of Java RMI (Remote Method Invocation). The architecture is developed referring to the official Java RMI documentation [5] and the document [6].

Each server implements a server interface, the interface lists all the methods that a client can execute on the server. Server instances running in different libraries are registered on the RMI registry, a registry URL obtained from this registration is used as the endpoint by the clients to execute any methods or accessing any kind of data from the server.

UDP/IP Sockets are used for communicating between two server nodes. Also the requests from the users and managers for any kind of operations (borrow book, return book, list items etc.) are sent to the server in the form of a UDPRequest.

For executing any operation the client always requests its local library server. If the server is capable of handling the request it responds accordingly to the client, if not the server sends the request to the other appropriate server (server becomes client) and waits for its response to send it back to the client.

Threads are used for handling multiple client requests to be executed simultaneously. One thread is responsible for listening to the requests and new threads are created to cater one single client request.

V. System Overview

The Fig.1 shows the general architecture of the system that we have implemented. The system consists of three libraries COM (Computer Department Library), ITC (Information Technology Library) and ETX (Electronic Department Library). Each library has its own server

running locally. The person interacting with the system can either be the library user or the library manager. The user and his/her identity is associated with a single local library but he/she can issue books from any of the three libraries in the system. Each library has a single manager; he/she only controls the administration of that particular library. Both the user and the manager have their own interfaces for

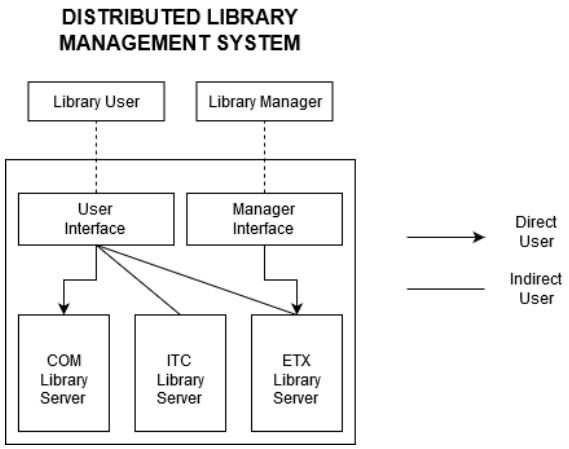

Fig.1. DLMS System Overview

interacting with the system. The users and managers have unique identities (ID Numbers) that they require to get themselves authenticated whenever they connect to the system. The users and managers are the direct users of their local libraries but the users are indirectly associated with the other libraries.

VI. Functional Overview of System

The system consisted of three departmental libraries. Each library had many books and their multiple copies. Java Hashmap was used to store the non-persistent data of books. The two major users of the library system are the library users (Students or Teachers) and the library manager (Library Administrator or Librarian). The users interact with the system through the users and manage interface. The library users and manager each have a set of operations that they can perform. The operations that the users and managers can perform are listed below.

User Operations:

1. *userLogin():* Before executing any operations on the system the user has to login to the system by providing a 8 character unique ID. For example: *COMU1234*. *"COM"* indicates the department,*"U"* indicates that the client is a user followed by a set of 4 unique digits for every user in the same department.
2. *borrowItem():* The user can borrow items from its local library or from any other library in the network.
3. *waitInQueue():* The user can wait in the queue for a particular item if it is not available or is yet to be added to the library.
4. *returnItem():* The user can return the borrowed item if he no longer needs it.
5. *findItem():* The user can look for an item's availability in his/her local library or the other libraries.

Manager Operations:

1. *managerLogin():* Before executing any operations on the system the user has to login to the system by providing a 8 character unique ID. For example: *COMU1234*. *"COM"* indicates the department,*"M"* indicates that the client is a manager followed by a set of 4 digits.
2. *addItem():* The manager can add a new item in the library. If the item is already present its quantity is increased.
3. *removeItem():* The manager can discontinue any particular item in the inventory.
4. *listItems():* The manager can list all the items in his/her local library along with their quantity.

The borrowLists, waitingLists for the items were stored as Java Hashmaps (Key,Value pairs). All the actions either by the manager or by the user are logged into a centralized log file. This can be used for debugging the system and performing error analysis.

VII. Result Analysis

The Distributed Library Management System we designed was thoroughly tested by simulating different scenarios consisting of both the user operations and the manager operations. The system was able to handle multiple requests at a time and remained responsive. The system always remained in a consistent state. All the operations were logged into a separate log file for each server, each manager and for each user.

```
>java DLMS.Server COM
RMI registry created at port 1234
Server registered.  Registry currently contains:
//localhost:1234/DLMS-COM
DLMS Server ready.
ManagerClient [COMM1111] log in successfully
 Manager COMM1111 list all of item success. All Items: COM1111 The Alchemist 4 , COM2222 Tuesdays
5 ,
```

Fig.2. Creating the Server for COM Library.

```
>java DLMS.ManagerClient
Enter managerID:
COMM1111
Lookup completed
Log in successfully

Please Select An Operation:
1: AddItem
2: RemoveItem
3: ListAllAvailability
4: Exit
3
 Manager [COMM1111] list all items in server ---> Success. All Items: COM1111 The Alchemist 4 , COM
2222 Tuesdays 5 ,
```

Fig.3. COM Library Manager adding a new book to the system.

```
>java DLMS.UserClient
Enter userID:
COMU1453
Lookup completed
Log in successfully

Please Select An Operation:
1: BorrowItem
2: FindItem
3: ReturnItem
4: ListBorrowedItem
5: Exit

1
Enter The ItemID
COM1111
 User [COMU1453] borrow item [COM1111] for [1] days ---> Success.
```

Fig.4. COM Library User borrowing a new book from the Library.

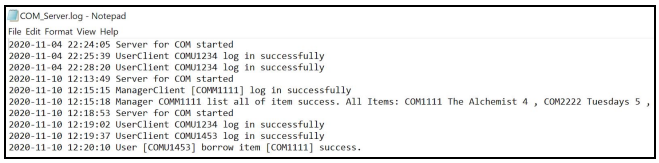

Fig.5. COM Library Server Log.

VIII. Conclusion

We were able to successfully build a completely functioning DLMS using Java RMI mechanism. The most challenging part was designing the server implementation as it was at the intersection of all the operations and the methods in the object were invoked numerous times.The system was a primitive one but can be easily extended. The Distributed approach of system design provides great flexibility to the designer particularly while designing large scale applications which are dynamic in nature (growing in terms of functionalities) and involve multiple processes which are dependent on each other and communicate with each other.

IX. References

[1] Aloia, Nicola & Concordia, Cesare & Meghini, Carlo. (2007). Implementing BRICKS, a Digital Library Management System.. 4-15.

[2] Balram, S. Yadav and K. K. Singh, "Application of Cloud Computing In Library Services," 2018 5th International Symposium on Emerging Trends and Technologies in Libraries and Information Services (ETTLIS), Noida, 2018, pp. 75-78, doi: 10.1109/ETTLIS.2018.8485214.

[3] X. Liu, "A Smart Book Management System Based on Blockchain Platform," 2019 International Conference on Communications, Information System and Computer Engineering (CISCE), Haikou, China, 2019, pp. 120-123, doi: 10.1109/CISCE.2019.00035.

[4] *Java Remote Method Invocation Distributed Computing for Java*. [Online]. Available: https://www.oracle.com/java/technologies/javase/remote-method-invocation-distributed-computing.html. [Accessed: 08-Nov-2020].

[5] "Java Remote Method Invocation (Java RMI)," JDK 6 Remote Method Invocation (RMI)-related APIs & Developer Guides. [Online]. Available: https://docs.oracle.com/javase/6/docs/technotes/guides/rmi/index.html. [Accessed: 08-Nov-2020].

[6] "Creating a Distributed System with RMI", CSE 486, Spring 2011, CSE Department, University at Buffalo.