Written by Yuting Chen

Agora Live Auction Report

Summary

For this final project, I have implemented a distributed auction platform in which users are able to create and bid on auctions. This platform allows multiple users to bid on items simultaneously. The standout feature of this project is in the distributed server cluster. Data are replicated on multiple servers that are synchronized in real time through a distributed consensus algorithm. The server cluster is able to survive failure of its member servers as long as the majority of the servers are still alive. When new servers rejoin a cluster after a failure, they will automatically get synchronized to the latest data in the cluster.

Architecture Overview

- **Communication Protocol**
    - Apache Thrift is used for the communication between server instances
    - Apache Thrift is used for the communication between client and servers
    - ThriftPaxos.thrift defines the data structures and methods used for Thrift
        - Files generated by compiling ThriftPaxos.thrift with thrift compiler are:
            - PaxosData.java
                - Main class for storing Paxos related data
            - Value.java
                - Additional data storage class used by PaxosData class
            - Bid.java
                - Class that contains information about a bid being made
            - User.java

        - Class that contains information about the client (the “user”) such as name, universal unique ID (UUID)
    - Item.java
        - Class that contains information about the item being sold
    - Auction.java
        - Class that represents the auction instance of item being sold. Keeps track of highest bidder, price, whether auction is open or closed
    - ThriftPaxos.java
        - Class that contains the Thrift RPC interface for the servers and the clients to interact with

- **Server**
    - Server.java class
        - This is a wrapper class for ThriftPaxosImpl.java class that implements the interface defined in ThriftPaxos.thrift file.
        - This class will create a server instance with 10 threads.
    - ThriftPaxosImpl.java class
        - This implements interface defined in ThriftPaxos.thrift file.
        - This class contains the logic behind Paxos distributed consensus algorithm.
        - Clients invoke the methods defined in this class to list, bid, create, delete, close auctions.
- **Client**
    - Client.java class
        - The main class for Client

        - This is a user interactive client that reads from the STDIN
        - Client is able to list auctions, delete/close auctions that it created, and bid on auctions
        - Which client has an UUID for unique identification (important for determining ownership of auction)

Client-Server Functionality

A client has the option of carrying out these 5 actions on the **command line**:

- LIST
    - List all the auctions and their information.
    - Use the itemName from here as an argument to BID, CREATE, CLOSE, DELETE
- BID *itemName bidPrice*
    - Bid for an item
    - Only non-negative integers are allowed
    - A bid is successful if the bid price is higher than the previous bid
- CREATE *itemName startPrice*
    - Create an auction item and set its starting price
    - Only non-negative integers are allowed
    - The itemName must not already exist
- CLOSE *itemName*
    - End the auction for the itemName
    - Only the original creator of this auction has the permission to close an auction
        - This permission is implemented using UUID
- DELETE *itemName*

- Delete the auction for the itemName
- Only the original creator of this auction has the permission to delete this auction
    - This permission is implemented using UUID (universal unique identifier)

Key Algorithms Used

- **Group communication**
    - The group communication is done through updating a shared state of auctions (e.g. by making a successful bid, or the creator of an auction closing that particular auction). The latest information about all the auctions can be viewed by any user at any time by typing the command “LIST” on the client.
- **Replicated Data Management**
    - Auction states, which are kept as a Java Map, are replicated across the server cluster. Any updates made to these Auction states (such as creating an auction, successfully bidding for an auction, deleting an auction, closing an auction) are instantly propagated across the server cluster.
- **Fault Tolerance**
    - If a server fails and rejoins the server cluster, the rejoined server will automatically receive the latest copy of the data from the cluster. Thus, as long as the majority of the servers in the cluster is alive, the individual servers are free to leave and rejoin. For example, in a five-server cluster, 2 servers can fail and optionally rejoin without having impact for the clients.
- **Distributed MutEx.**
    - Once an Auction is ended (in a “closed” state), no user is able to bid on that particular item. This is implemented as a Boolean shared state across the servers that determines whether an Auction instance is able to be updated or not.

- **Distributed Consensus**
    - I use the **Paxos** algorithm for the distributed consensus across multiple server instances. My Paxos is implemented using Apache Thrift and follows the algorithm described in *Paxos Made Simple* by *Lamport (2001)*.

Workflow of Using Client and Server

Please refer to README.txt for the example workflow using 5 servers and 3 clients, including tests for fault tolerance.