CSCI/ECEN 5673: Distributed Systems Spring 2024
MarketPlace, Assignment-1

Team Members:

1. Tarun Annapareddy
2. Surya Rajendran

Components:

1. **Server:**

We have a single Server Running. But this server will keep accepting the connections from the client and assign the work to the thread so they can communicate using specific buffers and each socket connection work can be handled in parallel

2. **Client:**

Our Client connects to the server with the socket connection and uses ObjectOutputStream and ObjectInputStream to communicate with the server. Once the communication is done the client closes the socket.

3. **Authentication:**

No API is allowed unless the customer creates an account/ Login to the system. So, the session ID is stored for each login and for write API we will be authenticating whether the person acting is authorized to do the task. For Example, the First time a seller login to the system we store the seller ID. When we receive a request to update the quantity of an item, we will check whether the person is the creator of the item

4. **Item KeyWords and Search:**

Each item has a set of keywords that we store in the db. When users search for a product with a certain set of keys, we do an intersection between both sets and respond with the item if there is at least one common match between the sets

5. **DB Connection:**

The first time we want to interact with the DB socket connection is made through username, and password verification utilizing the JDBC java connector. All the client request handlers re-use the same connection for interacting with the DB. This will save the connection costs.

6. **Warning and LogingOut for Inactivity:**

We use Timer Demon for each client connection session. Once the timer ticks the 3-minute mark, a warning is sent to the Client. Once it ticks the 5-minute mark we will close the socket connection

1. The first time the session is created, a timer demon is created with 2 timeouts one for 3 minute warning and a 5minutes logout
2. Every time the server receives a request from the client within the session the timer will be reset
3. Once the 3min timeout is done server will write a warning to the client using OutputStream
4. Once a 5min timeout is done server will directly log out the client

Testing:

Testing Multiple Buyer and Seller Clients:

We Spin up a Single client code with creates 10/100 threads and each thread acts as a client, they create their own connections with the server and act as a separate client and make the request

Benchmark:

Single Buyer and Single Seller:

Buyer:
Average LOGIN time: 0.78 milliseconds
Average SEARCH_ITEM time: 1.9 milliseconds
Average RATE_ITEM time: 0.56 milliseconds
Average UPDATE_CART time: 1.16 milliseconds
Average RESET_CART time: 0.66 milliseconds
Seller:
Average SELLER_RATING time: 0.6 milliseconds
Average SELLER_ADD_ITEM time: 1.3 milliseconds
Average SELLER_UPDATE_ITEM time: 0.7 milliseconds
Average SELLER_REMOVE_ITEM time: 1.18 milliseconds

- In this scenario, there is a balance between read and write/update operations, and the response times for both are relatively low.
- The server appears to handle both types of operations efficiently, with read (SEARCH_ITEM, SELLER_RATING) and write/update (RATE_ITEM, UPDATE_CART, SELLER_ADD_ITEM, SELLER_UPDATE_ITEM, SELLER_REMOVE_ITEM) operations having low average times.

Throughput:
Buyer Throughput: 1231/sec
Seller Throughput: 1058/sec

10 Buyers and 10 Sellers:

Buyer:
Total Average LOGIN time: 4.09 milliseconds
Total Average SEARCH_ITEM time: 8.12 milliseconds
Total Average RATE_ITEM time: 2.58 milliseconds
Total Average UPDATE_CART time: 1.99 milliseconds
Total Average RESET_CART time: 1.61 milliseconds
Seller:
Total Average SELLER_RATING time: 1.37 milliseconds
Total Average SELLER_ADD_ITEM time: 6.81 milliseconds
Total Average SELLER_UPDATE_ITEM time: 2.63 milliseconds

Total Average SELLER_REMOVE_ITEM time: 3.9 milliseconds

- As the number of buyers and sellers increases, the total average times for both read and write/update operations increase.
- The write/update operations, such as RATE_ITEM, UPDATE_CART, SELLER_ADD_ITEM, SELLER_UPDATE_ITEM, and SELLER_REMOVE_ITEM, contribute to the higher total average times. This suggests that the server is experiencing increased load due to a higher number of write/update requests.

100 Buyers and 100 Sellers:

Buyer:
Total Average LOGIN time: 9.924 milliseconds
Total Average SEARCH_ITEM time: 46.452 milliseconds
Total Average RATE_ITEM time: 31.046 milliseconds
Total Average UPDATE_CART time: 12.828 milliseconds
Total Average RESET_CART time: 5.11 milliseconds

Seller:
Total Average SELLER_RATING time: 2.053 milliseconds
Total Average SELLER_ADD_ITEM time: 6.217 milliseconds
Total Average SELLER_UPDATE_ITEM time: 14.708 milliseconds
Total Average SELLER_REMOVE_ITEM time: 24.529 milliseconds

- In this scenario, there is a substantial increase in total average times for both read and write/update operations.
- The read operations (SEARCH_ITEM, SELLER_RATING) have significantly higher average times, indicating a notable impact on the overall system performance. This suggests that as the system is in stress. We should scale the system beyond the threads. We need to run multiple instances of server and do loadbalancing to improve the performance.