Library Reservation System - Project Report

Dmytro Romaniv 151958, Patryk Maciejewski 151960

Introduction

The Library Reservation System is a web application that helps users manage book reservations in a library. It allows users to browse available books, make reservations, view their reservations, update reservation dates, and cancel reservations. The system uses Flask for the backend, Cassandra for the database, and Bootstrap for the frontend. It also supports stress testing to check how well the system performs under heavy use.

Database Schema

The database schema for the Library Reservation System consists of the following key tables:

- **books**:
  - `id` (UUID): Unique identifier for each book.
  - `title` (text): Title of the book.
  - `author` (text): Author(s) of the book.
  - `published_date` (date): Publication date of the book.
  - `pages` (int): Number of pages in the book.
- **reservations**:
  - `id` (UUID): Unique identifier for each reservation.
  - `user_id` (UUID): Identifier for the user making the reservation.
  - `book_id` (UUID): Identifier for the reserved book.
  - `reservation_date` (timestamp): Date and time of the reservation.
  - `status` (text): Status of the reservation (e.g., active, cancelled).

The tables are designed to ensure efficient data retrieval and storage, leveraging Cassandra's distributed architecture to provide high availability and fault tolerance.

Requirements

The project has the following requirements:

- A graphical user interface (GUI) accessible by at least 2 users simultaneously.
- A minimum of 3-4 nodes in the Cassandra cluster.
- The application must be operable from any node within the cluster.
- Users must be able to make, update, view, and cancel reservations. Each reservation must have a unique ID and should show who made it.
- Robust error handling and minimal delays in data processing.
- The ability to handle high load scenarios through stress testing.

Stress Testing

Stress testing is critical to ensure that the system can handle high loads and maintain performance under pressure. The following stress tests were conducted:

1. **Repeated Requests**: A single client makes the same request 10,000 times rapidly to test the system's ability to handle high-frequency requests.
2. **Concurrent Requests**: Multiple clients (two or more) make various requests randomly, totaling 10,000 requests to simulate concurrent usage.
3. **Reservation Contention**: Two clients attempt to reserve as many books as possible simultaneously to test the system's handling of contention and fairness.
4. **Continuous Cancellation and Reservation**: The same seat is continuously reserved and canceled 10,000 times to test the system's consistency and performance under repetitive operations.
5. **Bulk Update**: 1,000 reservations are updated in quick succession to evaluate the system's update performance and consistency.

These tests are essential for identifying bottlenecks, ensuring data integrity, and verifying that the system can scale effectively.

Problems Encountered

During the development of the Library Reservation System, several challenges were faced:

- **Token Collisions in Cassandra Cluster:** When setting up the Cassandra cluster, there were issues with token collisions, where two nodes had the same token. This was fixed by setting the `CASSANDRA_NUM_TOKENS` environment variable to ensure each node had unique tokens.
- **Data Filtering Issues:** Cassandra has limitations on data filtering, which required the use of `ALLOW FILTERING` in some queries. This can lead to unpredictable performance, so it was important to optimize these queries carefully.
- **UI Responsiveness:** Displaying a large number of books without pushing other content down was challenging. Adding scrollable containers for the book lists solved this problem and kept the interface clean and user-friendly.
- **Stress Testing:** Creating stress tests to simulate heavy use was necessary to check the system's robustness. The tests included rapid reservation requests, concurrent requests from multiple users, and constant updates and cancellations.

Conclusion

The Library Reservation System is an effective tool for managing book reservations in a library. Despite the challenges faced during development, including database setup and performance optimization, the system was successfully implemented. The combination of Flask, Cassandra, and Bootstrap provided a strong foundation for building a scalable and easy-to-use application.