Big Data Project Report

Title: Web-Based Reservation System using Apache Cassandra and Flask Authors: Wojciech Bogacz 156034, Krzysztof Skrobała 156039

Introduction

In this project, We designed and implemented a basic reservation system as a web application. The system utilizes Apache Cassandra for data storage, offering a scalable and distributed database solution suitable for Big Data use cases. For the frontend, We developed a Flask-based web application with templating to provide a simple and responsive user interface.

Objectives

1. To understand the integration of NoSQL databases like Cassandra in modern applications.
2. To build a lightweight reservation system using a web framework (Flask).
3. To implement a GUI for users to create and view reservations.
4. To gain hands-on experience with Cassandra's data modeling and query mechanisms.

Tools and Technologies

1. Database: Apache Cassandra
2. Backend: Python 3 with Flask
3. Frontend: HTML, CSS, JavaScript
4. Other Tools: Cassandra Query Language (CQL), cqlsh CLI, Bootstra(optional)

System Design

1. Database Schema

A single table was used to store reservation data in Cassandra. The schema was designed for fast inserts and queries by reservation ID.

```
CREATE TABLE IF NOT EXISTS reservation (
    seat_id INT,
    user TEXT,
    PRIMARY KEY (seat_id)
)
```

2. Data Modeling Considerations

Since Cassandra is optimized for denormalized and query-based modeling, the schema is designed to support quick lookups using seat_id. Further optimization (e.g., using composite keys or clustering columns) can be implemented for more complex queries.

Web Application

1. Flask Backend

   Routes:

   ```
   / - Home page with form to create reservation

   /book - POST endpoint to save reservation to Cassandra

   /book - PUT endpoint to update reservation to Cassandra

   /cancel - delete endpoint to cancel reservation to Cassandra
   ```

   The backend uses cassandra-driver to connect and interact with the Cassandra database.

2. Implementation Overview

   Set up Cassandra locally and created the keyspace and table.

   Developed Flask App with route logic to handle form submissions and display data.

   Connected Flask to Cassandra using the official Python driver.

   Tested the app by creating and retrieving multiple reservations.

3. Results

The web application was successfully able to:

```
Store reservation data into Cassandra.

Display existing reservations from the database.

Handle multiple user inputs with minimal delay.
```

Cassandra's write-optimized architecture provided high-speed inserts, and the web app remained responsive under repeated usage.

8. Challenges Faced:

- Initial configuration of Cassandra for local development.
- Ensuring consistent communication between Flask and the database.

9. Future Improvements:

- Add user authentication for personalized reservations.
- Use clustering columns to enable queries by customer or date.
- Deploy the app to the cloud (e.g., AWS, Heroku).

10. Conclusion:

This project provided practical experience in building a scalable, web-based system using Big Data principles. By integrating Flask and Apache Cassandra, I gained insights into NoSQL data modeling, web development, and the interaction between frontend and backend systems.

LInk to Github repository: GitHub Repository