Shopping Lists on the Cloud

Afonso Abreu
up202008552

João Malva
up202006605

Sofia Moura
up201907201

1 Introduction

This report approaches the development of a local-first shopping list application, blending local data persistence with a cloud-based collaboration. Each list is assigned a unique identifier, such as a URL, that can be shared between different users, allowing them to add or delete items. These items can have specific attributes assigned, such as flags to denote their acquisition and target quantities that dynamically update as the items are obtained. To support its ambitious goal of serving a massive user base, this project will use a consistent hashing approach inspired by the Amazon Dynamo paper, ensuring that the shopping lists operate independently and manage data access. For this milestone, we will delve into the details of the application's architecture, data management methods and the technologies employed to develop the described shopping list application.

2 Plan

The project plan involves the development of an application with local storage, initially featuring an interface for convenient development and testing. For this, we decided to use Java for the application implementation. The application aims to implement a cloud-based key-value database using consistent hashing for efficient data management. Additionally, the project includes the integration of replication and data synchronization, alongside a load-balancer to ensure optimal performance. Conflict resolution will be implemented through the 'Last-Write-Wins' approach initially, with a subsequent migration to a more suitable Conflict-free Replicated Data Type (CRDT). The plan also emphasizes the enhancement of the user experience with a focus on creating a more engaging and interactive interface.

3 Implementation

3.1 Consistent Hashing

The data storage mechanism within the system is built upon the principles of consistent hashing, enabling the distribution of data across a set servers. This method leverages a circular space where each server is assigned a position on the hash ring, and data is mapped onto the ring using a hash function. The hashing of keys to integers determines the specific server responsible for storing the data. This approach ensures a balanced distribution of data, minimizing the need for extensive reorganization when new servers are added or existing servers are removed. By adopting this consistent hashing approach, the system can effectively manage data storage and retrieval.

3.2 Message Protocol

The protocol of a message in distributed systems defines the rules and conventions for exchanging messages between different components. This way, we will specify how the messages will be formatted, transmitted and interpreted so that users can perform actions just as create and delete shopping lists and add or delete items in a certain list. The message format, which specifies how data is serialized, was decided to be in Java, due its portability and rich libraries. Each message will contain a header in order to identify its type and a body with the actual data. Since we are dealing with shopping lists and adding/deleting items some messages will contain counters and flags which indicate if an item was correctly placed, given an unique identifier.

3.3 Replication and Consistency

The data replication strategy ensures that data is stored in the server receiving the request and in the subsequent two servers, emphasizing fault tolerance and data redundancy within the network. The system enforces the requirement that any operation must be accepted by at least two out of the three designated servers responsible for data storage, enhancing data consistency and resilience. During data retrieval, the system prioritizes fetching data from the server processing the request and from at least one of the other two servers.

3.4 Conflict-free Replicated Data Types

One of the main goals of this application is the ability to create and share shopping lists efficiently while ensuring data persistence and availability. Given the need for high availability, initially it will be employed a Last-Writer-Wins strategy with local clocks to manage the lists, where the first thing written is the what will be stored. However, seeking for a scaling of the application to serve millions of users, there's a need for a more robust method and this is when CRDT's becomes relevant. CRDT's guarantee that updates made in an offline environment can be merged with the cloud-hosted version without conflicts. In addition, there will be a local copy of the data in a file, allowing the users to edit their lists whether in online or offline mode.

4 Conclusion

In conclusion, this project is based on a local-first shopping list application with offline capabilities, cloud collaboration and efficient data synchronization methods. To accommodate millions of users, it will be used a cloud approach, including data sharding and inspiration from the Amazon Dynamo paper.

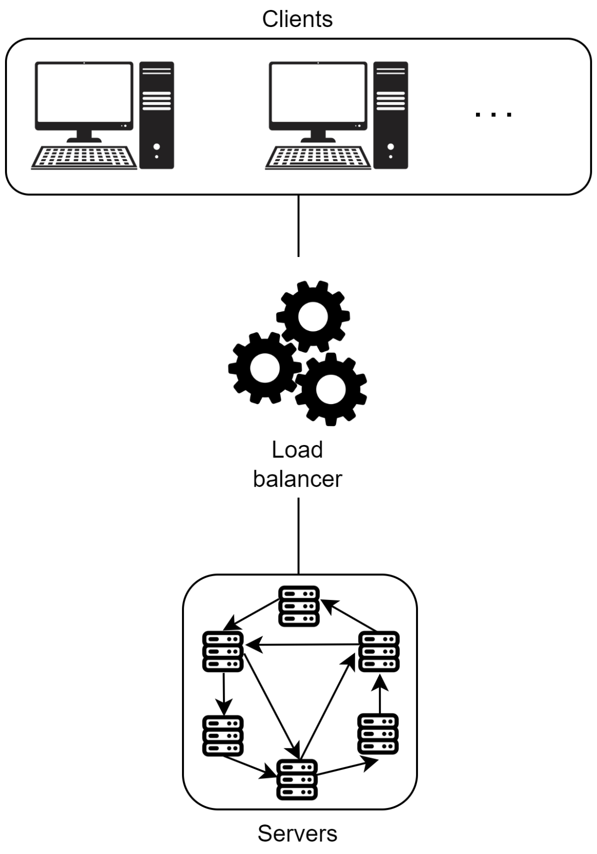

Figure 1: Application Architecture