Cassandra Project Report

Team Details:

- Sagar Jha: 110100024
- Rohan Gyani: 110040001
- Abhishek Gupta: 110040067

Introduction

Cassandra is a distributed storage system for managing very large amounts of structured data spread across many servers, while providing highly available service with no single point of failure.

A table in Cassandra is a distributed multidimensional map indexed by a key. The row key in a table is a string with no size restrictions. Columns are grouped together into sets called column families. Cassandra exposes 2 types of column families, Simple Columns and Super Columns.

Typically a read/write request for a key gets routed to any node in the Cassandra cluster. The node then determines the replicas for this particular key. For writes, the system routes the requests to the replicas and waits for a quorum of replicas to acknowledge the completion of the writes. For reads, based on the consistency guarantees required by the client, the system either routes the requests to the closest replica or routes the requests to all replicas and waits for a quorum of responses.

Cassandra is used in Facebook Inbox Search. It supports 2 types of search features: (a) Interaction and (b) Term Search. For (a), user id is the key and words that make up the message become the super column. For (b), user id is key and recipients' id is the super-columns.

Code Files

- Adder.java: Remote Interface of the Appliction
- AdderRemote.java: Remote Interface Implementation of the Application
- App.java: Hosts rmi services provided in above 2 files in App process (Application Layer)
- Client.java: It provides the user interface with features such as login, sending a message to another user, searching a keyword in the chat, opening a chat history with another person, logout

Implementation Details

We use the concept of chord to partition the data across nodes. Each node maintains a finger table which is used to locate the node storing the data item or the node wherein the data needs to be inserted.

When a node is inserted, it takes as parameters (self_node_id, helper_node_id, rmi_service_url). helper_node_id helps the new node setup its finger table. Currently, we are not handling transfer of data from an existing node to a new node, hence, all nodes need to be inserted at the start of the experiment.

Each node maintains 2 tables: Interaction, Term. Interaction table has key of row as sender, supercolumn as receiver and messages exchanged between them as columns. Term table has sender as key of row, each word in the message as the supercolumn and messages exchanged between them as columns.

.

We provide 3 functionalities:

1. send(username, message, username2)
2. search_chat(username, username2)
3. search_keyword(username, keyword)

In the 1st functionality, user1 sends message to user2 (interface provided by Client.java). Client makes a rmi call to ‘send’ of App.java. This inturn inserts into both the tables Interface and Term. In the Term table, multiple supercolumns are added depending on the words in the message. In Interaction, new supercolumns are added if the sender had not messaged the recipient ever before.

In the 2nd functionality, user1 searches chat history with user2. This request is forwarded to the appropriate node, depending on the finger table. In the Interaction table of that node, the columns corresponding to the supercolumn of username2 are returned in an array of strings.

In the 3rd functionality, user1 searches a particular keyword. This request is forwarded to the appropriate node, depending on the finger table. In the Term table of that node, the columns corresponding to the supercolumn of keyword are returned in an array of strings.

Storage Representation: We maintain a directory structure as follows:
Table->Key->Sueprcolumn files
In the files, we store messages along with length of the message sequentially.

Thank You!