Key-Value Store

Kapil (16305R010) , Abhishek(163059005)

September 2, 2017

1 Problem Statement

Design a key-value store, with a stateless front-end, and the key-value pairs distributed across one or more back-end servers. The front-end server redirects the request to one of the back-ends depending on the value of the key

2 Application Architecture

We designed a 3-tier key value system with multiple back-end key-value stores to store the key values. We have a front end server to manage the client requests and select the appropriate key store on the basis of key.The client provides the basic functionalities like creating and authenticating users using front-end server, adding, updating , deleting a pair and fetching the value of key supplied by the user from key-value server using client and front-end server.

We have divided application into 3 tiers

1. Client Layer
2. Fronted Server
3. Back-end Key-Value Store

2.1 Client Layer

Client layer acts as an interface between user and the front-end server , this layer submit user request to front-end server and the shows response to client . This layer consists of

1. User Authentication
2. CRUD operation on Key-Value Store

For this we have made a console base user interface where user can enter his choices

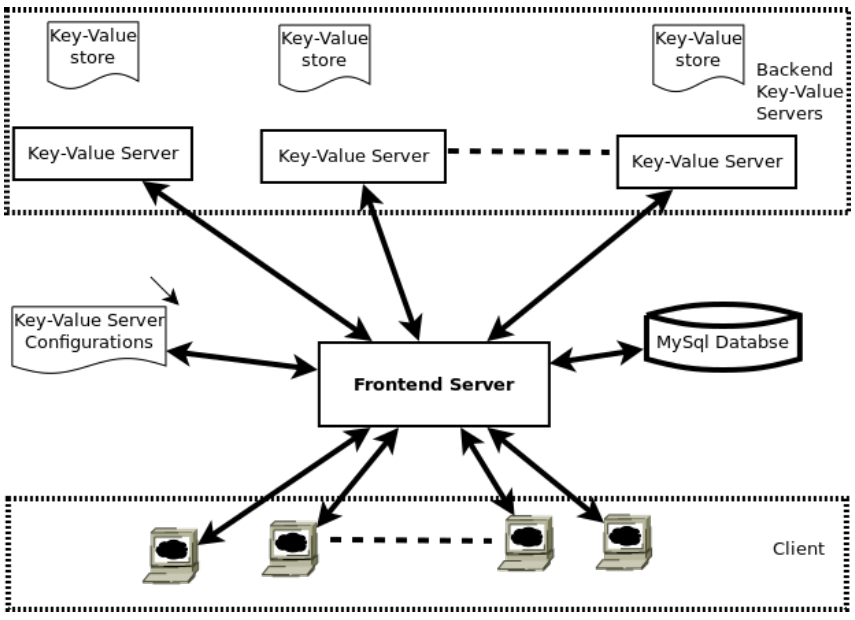

Figure 2.1: High Level Architecture

2.2 Front-end Server

Front-end server follows the multi-process architecture. For each user session there is a separate process on front-end server.This tier acts for user authentications using MySQL database and after authentication distributes the user request to appropriate key-value store server using the configuration of available Key-Value server in configuration file.

2.3 Key-Value Server

Key-Value server follows multi-threaded architecture.For each new request from front-end server, a thread is created to fulfill the request .There can be n numbers of Key-Value server. Each server stores key within a given range that can be configured once the key-value servers are started. Server stores the disjoint range of values so that one request is mapped to one key-value server.

Basically it use in-memory hash-table. For persistent storage of the key-value pairs, the server serializes the data to database.db file available on each key-value server that acts as key value store and deserialize it on start up .

Server allows multiple clients to read at a time but allow only single user to write (insert, update and delete)(multiple user can do write on different server in parallel- one user per key-value server)

3 Working and Messages

The application allows multiple clients to search simultaneously so read is not exclusive but while performing insert ,update and delete operations it requires that only one thread progress and other thread should block.

Above requirement is similar to reader writer problem where multiple readers can read simultaneously but write are exclusive.

The client sends various types of requests to front-end server like

- User Authentication
- User Registration
- Insert Key-Value pair
- Update Key-Value pair
- Delete Key-Value pair
- Fetch Key-Value pair based on key

The Front-end server forks a separate process for every user. It does two things. Authenticate the client using password authentication. Read the server.conf file to figure out which key-store to pass the request. And pass the following messages to key-value servers and send the result to client.

- Insert Key-Value pair
- Update Key-Value pair
- Delete Key-Value pair
- Fetch Key-Value pair based on key