Design Document

A Simple Distributed Hash Table

Programming Assignment 2

TABLE OF CONTENTS

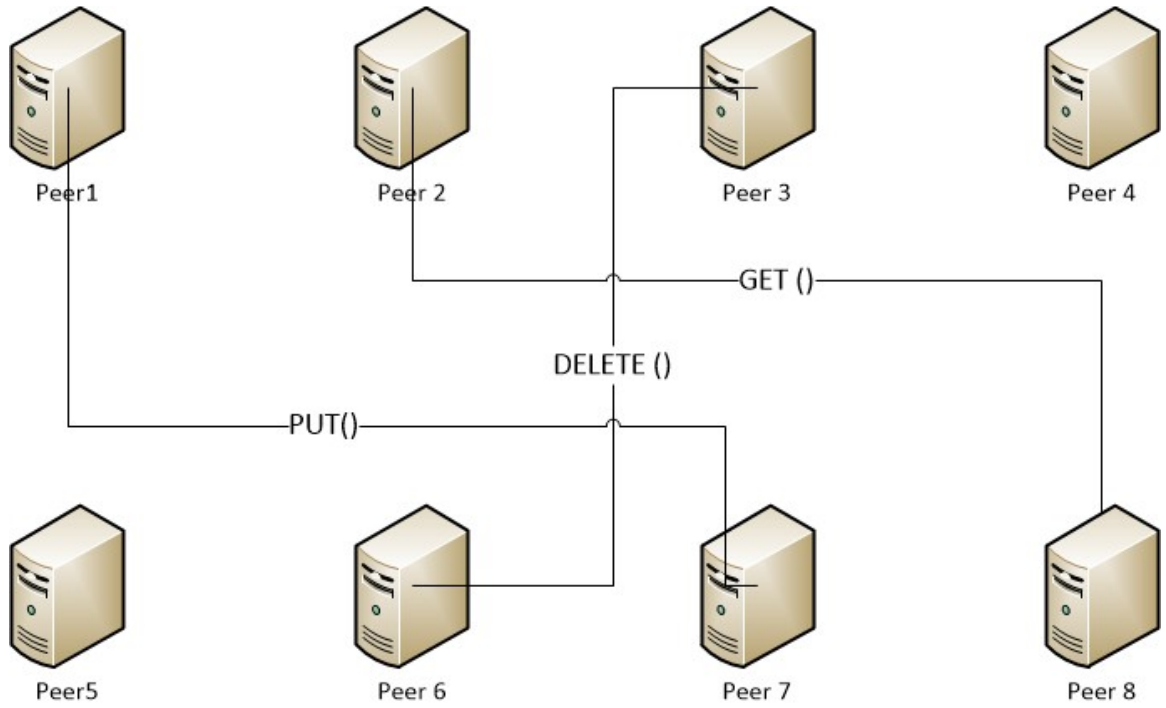
1. **Introduction**
    a. **Purpose**
        - It is possible for single server to hold Hash Table and to handle all requests.
        - Purpose of the simple Distributed Hash Table System is to high performance computing, scalability, fault tolerance, persistence.
        - The main advantage of hash tables over other table data structures is speed.
    b. **Scope**
        - This system is designed to share the Hash Table across multiple peers (Client/Server) connected over network. There are mainly 3 functions
        - **put**- Each peer (Client/Server) can call put function and make Key-Value pair entry on any server. Server will get selected by result of hash function
        - **get**- Each peer can search for other peer's Key and Value with the help of this function.
        - **delete** – Each peer can call delete functionality, to remove the particular Key-Value pair from distributed Hash Table holding that pair.
    c. **Acronyms, Abbreviations and Definition**
        - **Peer** – is Machine on server as client or server.
    d. **References**
        - http://datasys.cs.iit.edu/projects/ZHT/
        - https://en.wikipedia.org/wiki/Hash_table
        - To understand the Distributed System Architecture and its original intent

2. **System Overview**
    a. **Functionality**
    1. In Distributed Hash Table System, peers use to communicate with each other for multiple reasons. Like file downloading.
    2. Firstly Server in our case peer (Client/Server) gets start and put itself in ready state to accept request from other peer for 3 reasons either to **Put** or to **Get** or Delete the Key-Value pair.
    3. In **Put** call, peer sends a Key-Value pair to the hash function and which will return the addresses of server, where that Key-Value pair will get stored.
    4. After receiving **Put** call server stores the Key and Value in Hash Table in the format of that is <Key, Value> pair.
    5. If same Key is present in Hash Table Server simply overwrite the value associated with that key with new value.
    6. In **Get** call, Peer will send the Key to hash function, then hash function again will decide which server has that key accordingly request will route to that server. Server will search for specific Key send by peer on successful search, Server returns the value associated with that Key otherwise unsuccessful search message to client.
    7. In **Delete** call, peer send the key to has function, then hash function will decide which server can hold that Key-Value. Accordingly request get routed to that server.

8. For delete operation, server will lookup into hash table. If that hash table contain key send by peer then server will simply delete that Key-Value from hash table, otherwise will return the unsuccessful error message back to peer.

3. System Architecture

a. High Level Architecture



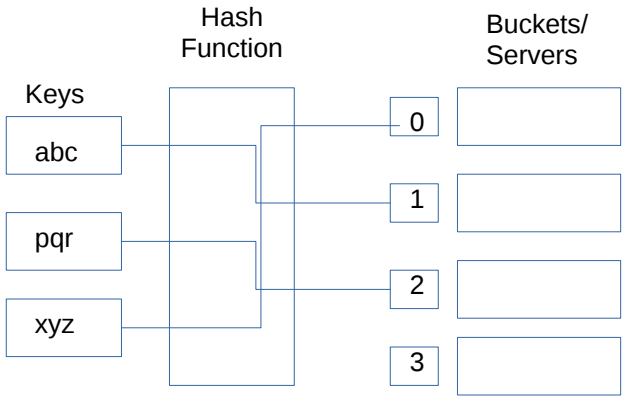
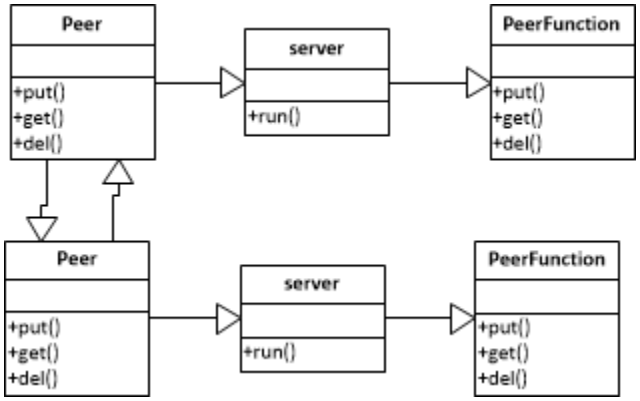


fig. Hash function working

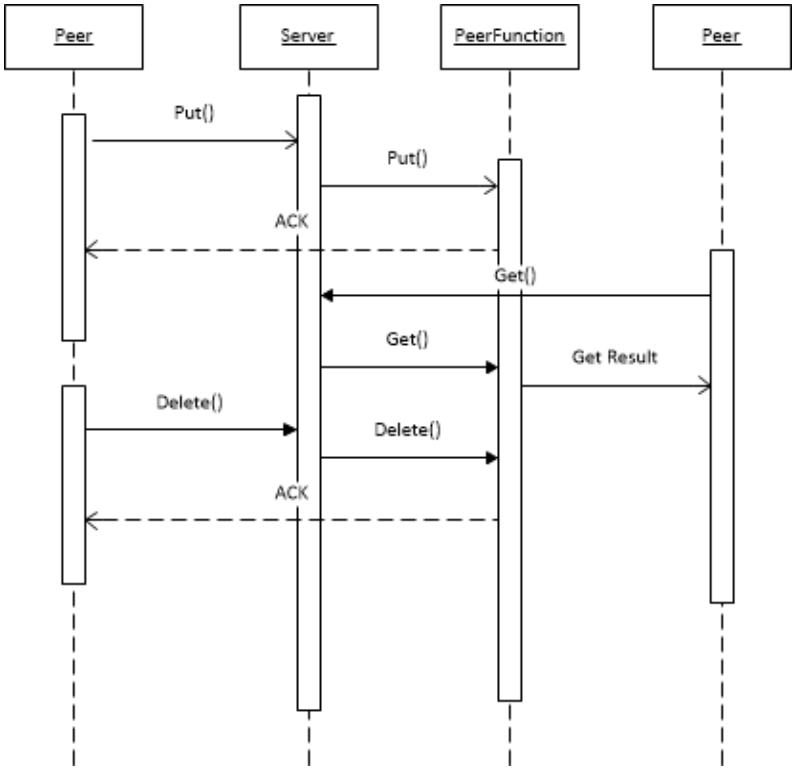
As shown in diagram Distributed Hash Table system is consisting of below components.

1. **Peer** – Requesting for storing Key-Value pair on server. Requesting value associated with Key. Request Delete Key-Value pair from server.
2. **Hash Function** – Peer enter the Key and value to store on server. Hash function on local machine finalize the server where that Key-Value is going to be saved. Similarly hash will help in searching and in delete operation to get Server address which could contain that possible Key-Values.
3. **Server/Bucket** – Server is system who store the value and Key associated with value in Hash Table.

b. Class Diagram –



c. Sequence Diagram –



4. **Technical Overview**
    a. Server
        - Server is a server side class. It shall always listing for new requests from client/peer.
        - As soon as clients get connected with this server it will create new thread for requesting client. And again go on listing for new client request.
    b. Peer/Client
        - Peer class could be client requesting for Put, Get and Delete functionality.

5. **Component Description**
    There are 2 main components in Distributed Hash Table System
    a. Server
    b. Peer

6. **Software Requirement**
    a. **Java** installed on machine
    b. **Ant** for building Java Application

7. **Future Scope**
    a. There should be a fault tolerance, if we kill any server client should be bale to locate the replica.
    b. Persistence- There should be way like file. To store/capture the current state of Hash Table and saved them for future use. If server goes down and if we make it start again it should have last working hash table details.