A20354704 Ashwin Chati

Design Document

Distributed Hash Table

A20354704 Ashwin Chati

Problem Statement:

Implement a distributed hash table such as ZHT. Each peer should be both server and a client. As a server, it should accept queries from other peers, check for matches against its local hash table, and respond with corresponding results. As a client, it should provide interfaces through which users can issue queries and view search results.

System should perform below operations:

1. put
   - the put operation should return success or failure.
   - Key should be of type string
   - value should be of type string

2. get
   - the get operation should return the value or null
   - Key should be of type string
   - value should be of type string

3. delete
   - the delete operation should return success or failure
   - Key should be of type string

System Architecture:

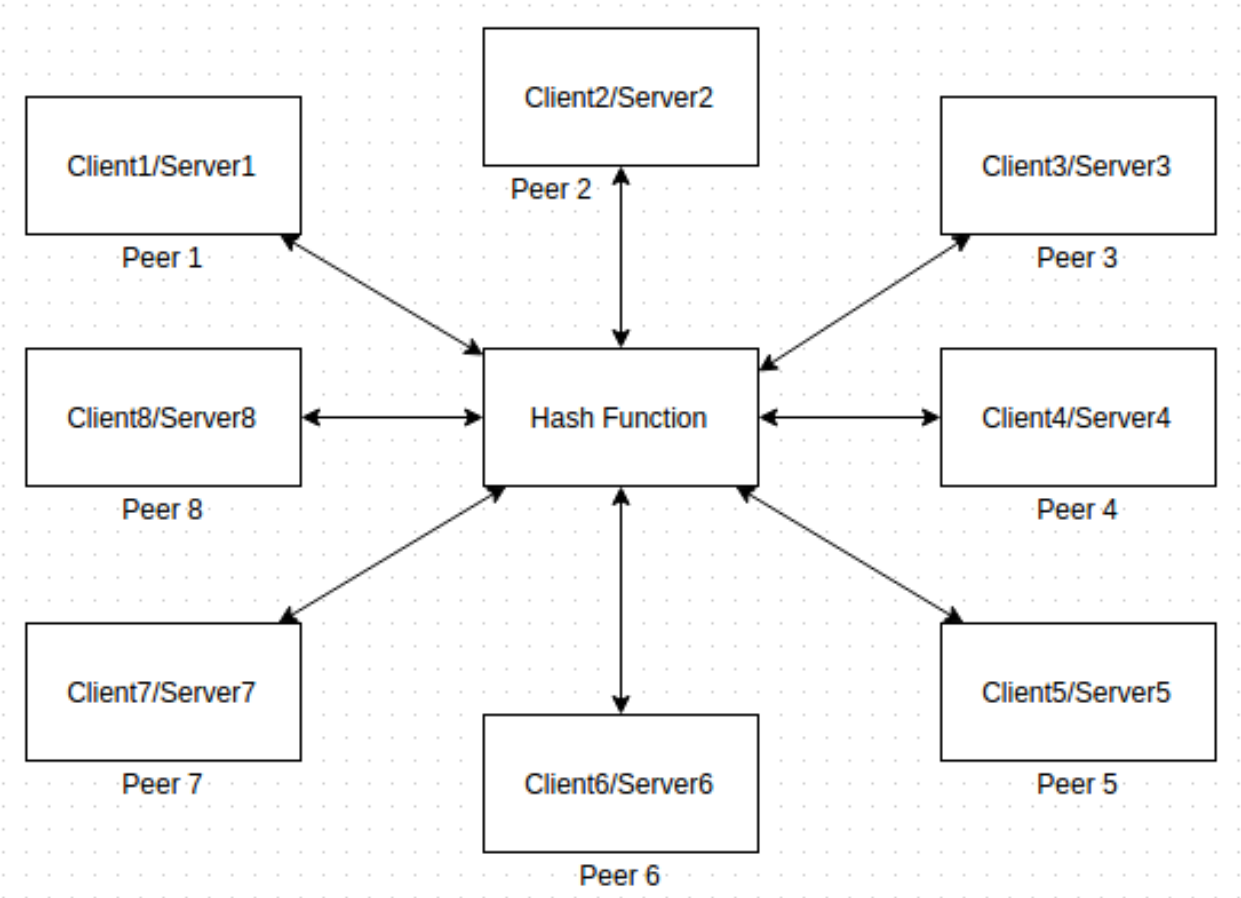
The high level architecture is shown in the given figure.

Each peer should be both server and a client.

As a server: It should accept queries from other peers, check for matches against its local hash table, and respond with corresponding results.

As a client: It should provide interfaces through which users can issue queries and view search results.

Hash Function: A hash function maps keys to small integers (buckets). Hash function is implemented to decide which peer to connect to for performing the operations.



A20354704 Ashwin Chati

Functionality:

Peer can perform below three operations:
1. put value
2. get value
3. delete value

1. Put:
As a client: It takes input in the form of key and value are taken from user. It then decides on which peer to connect using hash function based on key value and sends the key and value to the particular peer.

As a server: It takes key and value pair sent by the other peer and inserts the values into the hashmap.

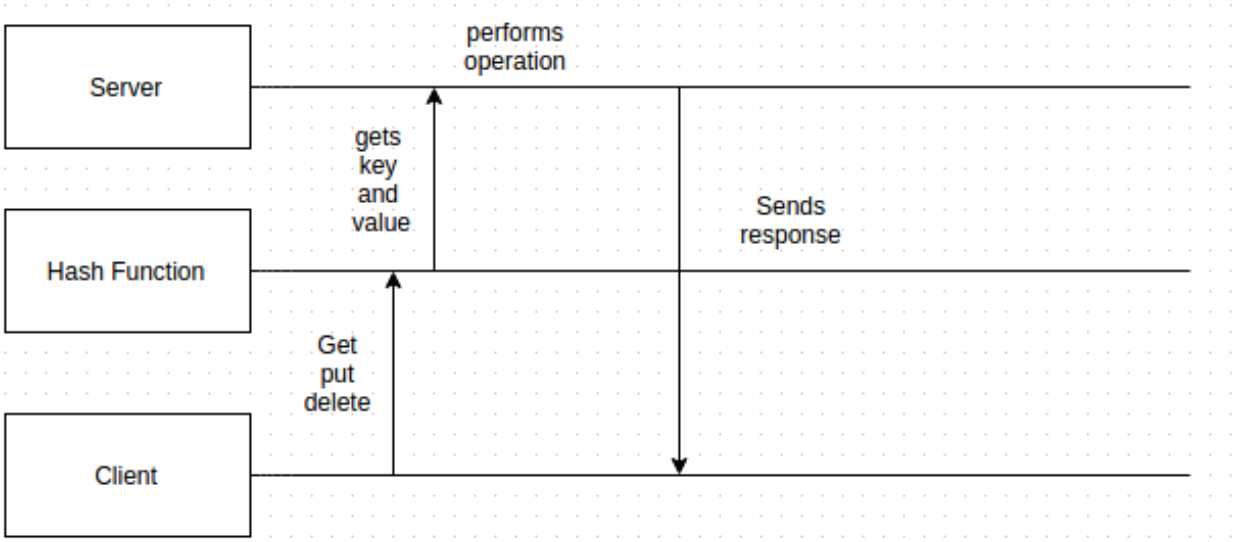
2. Get:
As a client: It takes input in the form of key for which the value needs to be find. It then decides which peer to connect using hash function based on the key and sends the key to that particular peer.

As a Server: It takes key sent by other peer for which the value needs to be searched. Then the value for that particular key value is searched and the response is sent back to the requesting peer.

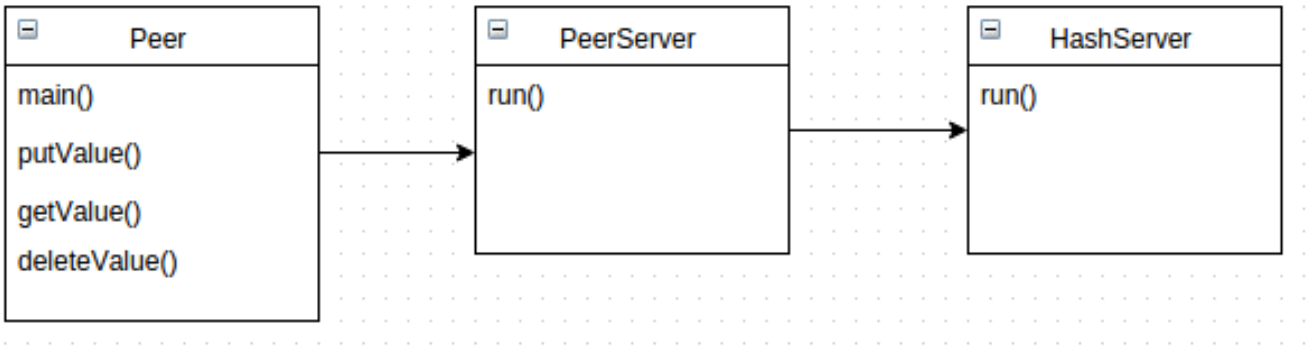
3. Delete:

As a client: It takes input in the form of key which the user wants to delete . It then decides which peer to connect using hash function based on the key and sends the key to that particular peer.

As a Server: It takes key sent by other peer for which the value needs to be deleted. Then the value for that particular key value is deleted and the response is sent back to the requesting peer.


Server
Hash Function
Client
performs operation
gets key and value
Get put delete
Sends response

Class Diagram:



Peer: This is the main peer class. When it up and running it keeps listening for the peer request to connect. When the peer gets connected it generates a new thread and keeps listening for other peers.

PeerServer: This class is used to process the threads initiated by the main Peer class. Put, get, and delete requests are processed by this class.

HashServer: This class is used to process the thread form the peers for get, put and delete.

System Requirement:

1. Operating System: Windows/Linux
2. Softwares: a. java
   b. Ant

Future Scope:

1. Resilience can be added for key, value pairs.