Introduction to Distributed Hash Table

1. **What is a Distributed Hash Table?**

   A **distributed hash table** (**DHT**) is a class of a decentralized distributed system that provides a lookup service similar to a hash table: (key, value) pairs are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption. This allows a DHT to scale to extremely large numbers of nodes and to handle continual node arrivals, departures, and failures.

2. **ZHT(Zero-Hop Distributed Table)**

   ZHT aims to be a building block for future distributed systems, such as parallel and distributed file systems, distributed job management systems, and parallel programming systems.

3. **Operations Provided**

   The program provides 3 key features

   - Put(Insert)
   - Get (fetch)
   - Delete (remove)

Software Architecture

- The System consists of a Peer which functions like a server and a client.
- The Client sends request to the server to perform the specific operations of get,put,delete

Client-Side Architecture

- The User selects an operation of get,put,delete.
- There is a ArrayList called **ArrayList_ServerList** which has the IP's of all the servers the client can connect to. The client also has a table for storing sockets whenever there is a connection established with the server. Whenever a key value is entered by the user the client sends that value to the Hash_Function method which converts the key into a byte array and divides the total by the total number of servers and returns the modulus value. Depending on the modulus value the server is chosen if the client is making a first time connection with the server then the IP is fetched from the ArrayList_ServerList and the connected socket is stored in **HashStorageSocket** and if the server-client connection is made previously then the socket is fetched from the **HashStorageSocket** depending on the value returned by Hash_Function.

Hash Function

- The Hash_Function is used to get the server where the value is stored/present.
- Whenever the key is passed to the hash function it converts it into a byte array and adds the total value.
- The total value is divided by the Total Server count and the modulo value is the server number where the value of the specific key is stored or needs to be stored.

Data Passing

- Message String(Key:10bytes Value:90bytes) is used for passing data from client to server
- The String consist of the operation to perform(get,put,delete) with the key value pair.
- The connection established between client and server is broken at the end(when the client shuts down permanently)
- On adding/deleting a key server returns a Boolean value of success or a Boolean failure on and for Get it returns the value associated with the specific key.

Server Architecture

- The server is a multi threaded server uses Sockets to establish client connections
- The Server has 3 methods:
    - boolean Put(String key,String Value)
    - boolean Del(String key)
    - String Get(String key)

The Put method inserts the value in the server hashtable.Del is used for deletion and Get ,fetches the value associated with the key.

Conclusion and Future Scope

The Hash Table provides an immediate lookup depending on the key present in it and is implemented on the server side to provide high speed computing and scalable systems. Using this system the performance is evaluated and measured upto 10k operations using 16 servers and clients on amazaon EC2.

Future Scope

- The Future Scope includes adding a removing servers dynamically
- It also includes more robust Hash Functions
- Improving and having a robust network topology
- Making the system more efficient to 1M and 10M node operations
- Having a broad cast primitive

References and Research Material

http://datasys.cs.iit.edu/projects/ZHT