Contents

1. System Introduction

Distributed Hash Table ((hereinafter referred to as “DHT”) System simulates the behavior of Hash Table Data Structure across network by taking advantages of the Distributed Systems. This concept will help to achieve scalability if resources are increased based on the demand. DHT concept can be used as File Indexing Servers, Domain Name Servers, etc. It utilizes the advantages of decentralized distribution system by providing Dictionary like interface. Nodes of this Dictionary like interface can be spread across the network. It provides:

- A distributed decentralized database where users can store the data using Key – Value pair concepts
- DHT Client can perform Put, Get and Delete operations on DHT database
- Log File at Client and Server ends will record the operations done on each of the nodes

Some of the features available in the current System are,

- **Easy to use console based interface**

  A console based interface allows user to select through various operations (Put, Get, and Delete). Depending upon the operation selected by user, System prompts for the required inputs (like Key Value, Key Path, etc.).

- **Highly parameterized system**

  The “.properties” file allows administrator or user to define the various options (like DHT Server IP Address, Port, No. of Servers in DHT Environment, No. of Data Points for Testing purpose, etc.) for keeping the System scalable and parameterized.

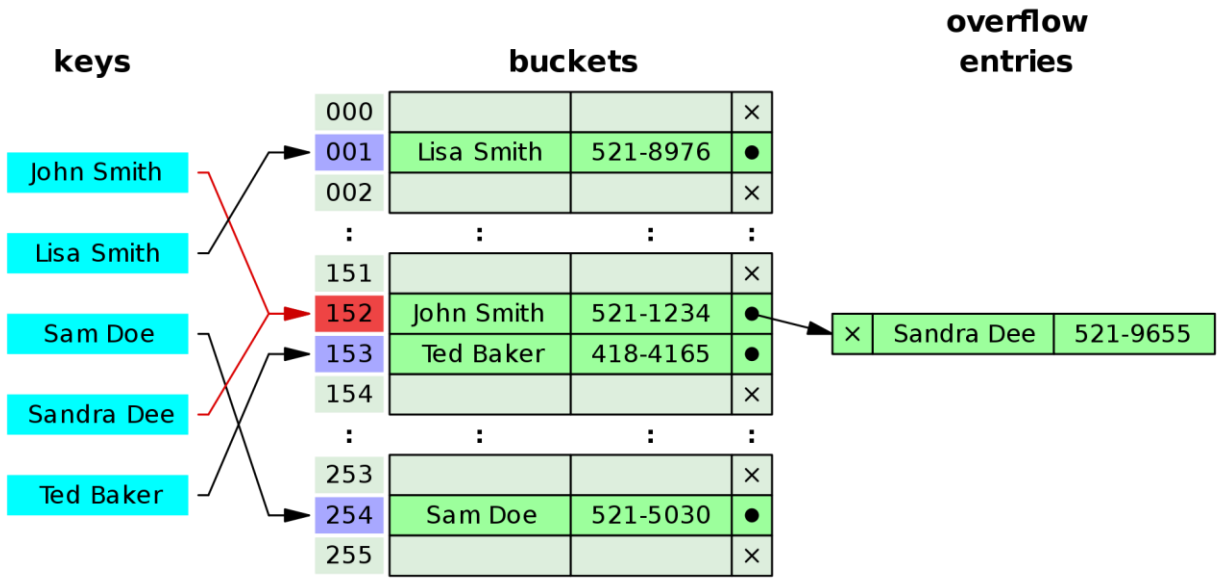
- **TCP Support**

  System uses TCP / IP Protocol to ensure that the data is downloaded with consistency. The reasons for choosing TCP / IP over UDP are listed below.

  - TCP is connection oriented – once a connection is established, data can be sent bidirectional.
  - It is also widely used by other protocols like HTTP, HTTPS, FTP, SMTP, and Telnet which makes this protocol widely popular for Net Traffic.
  - TCP is more reliable since it manages message acknowledgment and retransmissions in case of lost parts. Thus there is absolutely no missing data. UDP does not ensure that communication has reached receiver since concepts of acknowledgment, time out and retransmission are not present.

- TCP transmissions are sent in a sequence and they are received in the same sequence. In the event of data segments arriving in wrong order, TCP reorders and delivers application. In the case of UDP, sent message sequence may not be maintained when it reaches receiving application. There is absolutely no way of predicting the order in which message will be received.
- It has its own Error checking mechanism ensuring the data order and consistency.

2. Architecture



(courtesy: Internet)

The image above clarifies the architecture for the Distributed Hash Table. DHT Server can be identified applying Hash Function on Key. Once the required DHT Server is identified then operations can be performed by applying same or different Hash Function on the identified DHT Server.

3. DHT Server

a. Process

DHT Server accepts the data from various DHT Clients using Socket and Multithreading concepts. As soon as DHT Client connects to DHT Server for the first time, there is a new thread created on DHT Server and this newly created thread is then taken over by the DHT Server thread handler in order to fulfill DHT Client's request. The connection between DHT Client and DHT Server is released only after all the client operations are completed or client process is terminated. This helps to reduce the overhead cost of opening and closing the connection for every small chunks of data.

DHT Server uses ConceurrentHashMap structure from Java to simulate Distributed Hash Table data structure behavior. The Initial Capacity defined while creating ConcurrentHashMap serves as number of Buckets in Hash Table.

Program uses inbuilt hashCode function from Java and Key data to identify a destination DHT Server. Once the destination DHT Server is identified then the data is sent to the server using Sockets and Multithreading concept. After receiving the Key and Value / just Key data, server computes Hash Code using inbuilt hashCode function from Java and Key data. This Hash Code is divided by the Initial Capacity of ConcurrentHashMap structure and remainder is considered as Hash Table Bucket Number to put, get or delete the data.

It performs the operations like Put, Get, and Delete on DHT database based on the inputs received from DHT Client.

b. Tradeoffs made

- Have used ConcurrentHashMap instead of SynchronizedMap. The former one locks a particular bucket or segment to which it is writing whereas the later one locks the complete Map object while writing or reading. But the later one ensures 100% data consistency in case of highly concurrent operations but lacks speed. It is exactly reverse case with ConcurrentHashMap

c. Possible Improvements

- Usage of Thread Pool

- Better exception handling
- More focus on Synchronization in order to avoid Data or output inconsistency due to use of ConcurrentHashMap
- Data Resilience
- More focus on enhancing program to have better throughput in real life Systems and data
- More focus on enhancing program to optimize Get performance

4. DHT Client

1. Process

DHT Client either accepts the data from user or generates random Key – Value pairs. It then performs various operations accordingly. In addition, it also prompts to user for additional inputs depending upon the operation selected. DHT Client uses Socket to communicate with DHT Server.

It performs the operations like Put, Get, and Delete on DHT Server database based on the inputs received from DHT Client.

2. Tradeoffs made

- User console based input is disabled for automation of Testing

3. Possible Improvements

- Better exception handling

5. Additional features

Following are some of the additional features Distributed Hash Table implemented.

a. Client based log file creation
b. Server based log file creation
c. Console based interface
d. Usage of Properties file to read various parameter