SIMPLE FILE SYSTEM IN DHT

TEAM MEMBERS

- **AKSHAY KULKARNI [kulka182@umn.edu]**
- **ROHAN SADALE [sadal005@umn.edu]**

DESCRIPTION

In this project we have implemented simple file system using distributed hash table(DHT) based on the chord protocol. With the help of this system, the client can write and read file using the DHT. Chord provides support for one operation: given a key, it maps the key onto a node. Chord is scalable, with communication cost and the state maintained by each node scaling logarithmically with number of chord nodes. Each node maintains a finger-table which stores successor of the current node. The trick here is that number of neighbors stored by each node is proportional to log of number of nodes in DHT system. This allows for fast search (O (log N)) of key across the system.

Following are the major components of the project: -

- **Super Peer: -** Purpose of this component is to maintain information about all the participating nodes in DHT. When a new node wants to join DHT, it first contacts Super Peer, then Super Peer registers that node and the node can move forward by setting and updating its finger table. Basically, Super Peer does the job of book-keeping.

  Functions exposed by Super Peer[JoinProtocolHandler.java]:
  - **Join: -** This function takes IP address and Port of the node that wants to join DHT. If Super-Peer is busy, then it returns -1 else return Node-Id for the node that wants to join DHT.

  - **addToDHT: -** Super Peer when becomes available facilitates the joining of new node into DHT. It adds details of the node to the list of active nodes that are currently in DHT. The function Node object and returns list of all the nodes including current node that are currently part of DHT.

  - **PostJoin: -** After node joins DHT and all other nodes have updated their finger-table. Super Peer changes its state from busy to available.

- **GetNode: -** This function randomly selects a node from DHT and returns Node object.

- **Nodes:** - They are the actual components that form DHT. Nodes in DHT are laid out in circular network. When a node joins DHT, it receives list of all other nodes that are currently in DHT by Super Peer. Based on this, node updates its finger table. After nodes is done with updating its own finger-table, it broadcasts its presence to all other nodes in DHT. When other nodes become aware about new node in DHT, they also update their own finger-table. These nodes are responsible for storing files locally.

  Functions exposed by Node[OperationHandler.java]
  - **Write: -** This function four parameters
    - **Filename: -** The name of the file which is to be written.
    - **Content: -** Content to be written in the file
    - **shouldWrite: -** should file be written on current node.
    - **basePath: -** In which directory file should be written

    and decides whether file should be written on current node or not. If yes, file is written on the current node else request is transferred to next node whose Id is smaller or equal to id of the file that is being written.

  - **Read: -** This function takes three parameters
    - **Filename: -** The name of the file which is to be read.
    - **shouldWrite: -** should file be read from current node.
    - **basePath: -** From which directory file should be read.

    and decides whether file should be read from current node or not. If yes, file is read from current node else request is transferred to next node whose Id is smaller or equal to id of the file that is being read.

  - **UpdateDHT: -** This function takes list of nodes that are currently part of DHT and for each node it is checked whether addition of new node to DHT updates its finger-table or not. If we are required to update finger-table, then it is updated and in this correct information about the system is maintained.

- **Clients**: - They are responsible for reading and writing files into DHT. Client will first contact Super Peer for IP/Port of any random node in DHT. After obtaining this information, client will connect to that node and will check if that node is responsible for the file which is client is going to read or write. If yes, then client proceeds with its task else requests is recursively forwarded to other nodes in DHT.

  Client Scripts
  - **WriteClient.java: -** This program reads all files from given directory and distributes all files uniformly across the network.
  - **ReadClient.java: -** This program reads all filename from given directory and searches for the file on DHT, reads the content and displays the file content on standard output.

- **Configuration**
  Config.txt is the file which contains all configuration parameters. This file follows the pattern of **key:value .** Following are the parameters that can be configured: -
  - SuperPeer => IP Address of Super Peer
  - Port => Port on which Super Peer is listening to requests.
  - ReadDirectory => Directory that stores files to be distributed over DHT
  - WriteDirectory=> Directory in which files are to be written.
  - Verbose => Shows complete log (visited nodes) for the request.

HOW TO RUN COMPONENTS?

- thrift --gen java node.thrift
  thrift --gen java path.thrift
  thrift --gen java join.thrift
  thrift --gen java operation.thrift

  Running above four commands will generate Stub files in gen-java folder and we will need to include gen-java in class path.

- javac -cp .:../jars/libthrift-0.9.1.jar:../jars/slf4j-api-1.7.14.jar:gen-java/ *.java
  This will compile all the java files and create .class files. This assumes jars required to compile these files are placed in jars folder. If not, one should modify and provide appropriate path for jar.

- java -cp .:../jars/libthrift-0.9.1.jar:../jars/slf4j-api-1.7.14.jar:gen-java/ SuperPeer config.txt
  This will run SuperPeer and it is now ready to receive requests. SuperPeer will read file config.txt and set Maximum number of nodes that could be present in DHT.

- java -cp .:../jars/libthrift-0.9.1.jar:../jars/slf4j-api-1.7.14.jar:gen-java/ DHTNode config.txt Port-Number
  This will start new node that wants to join DHT. All of nodes that form DHT listen on port 9091 by default if no command line parameter is provided. All configuration parameters are picked from config.txt. Running multiple instances of this file on different node or on same machine with different port will constitute DHT.

By running above three commands, we have basically established DHT network and now the system is ready to receive and process the requests.

- java -cp .:../jars/libthrift-0.9.1.jar:../jars/slf4j-api-1.7.14.jar:gen-java/ WriteClient config.txt
  This will read all the files from directory mentioned in config.txt and distribute all files in that directory on DHT network.

- java -cp .:../jars/libthrift-0.9.1.jar:../jars/slf4j-api-1.7.14.jar:gen-java/ ReadClient config.txt
  This will read all the filenames from directory mentioned in config.txt and read content of those files from DHT.

TESTCASES

- Before every request is processed, a random node from DHT is selected that acts as starting node from which requests are processed. Let **Node-Id** be Id of the node selected and **File-Id** be Id of the file that is to processed on DHT. Following scenarios are possible
  - **Node-Id** is immediate successor of **File-Id.** In this cases infinite loop can be avoided by keeping tracking of previous node for each.
  - **Node-Id** is immediate predecessor of **File-Id.** In this case file is processed on next –node i.e. immediate successor of **Node-Id.**
  - **Node-Id** is same is as **File-Id.** In this case file is processed on current node.

    - In other cases, requests are forwarded to the node whose Id was just small or equal to File-Id and this recursive process continues until we reach target node.

- The system was tested when only single node was present in DHT. System worked correctly.
- The system was tested when one host machine deployed multiple nodes on different ports. Again, all test cases worked correctly.
- Also, the tried to read the file that was **not present** in DHT. In this case "NIL" was returned was given output.
- Also tried to test maximum number of nodes limit in DHT i.e. if maximum number of nodes in DHT is N, then we also ensured that (N+1) th cannot join the system and appropriate message is displayed.