

Programming Assignment 1: Simple File System on a DHT

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.

COMPONENTS

Following are the major components of the project: -

SuperPeer

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 are as follows

- 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*)

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. This is the location on client side from which files will be picked up for reading or writing to DHT.
 - Ex:- If ReadDirectory is /home/kulka182/ and filename is world.txt then required file name is /home/kulka182/world.txt and this file will be stored on DHT.
- WriteDirectory: Directory in which files are to be written. This is the path DHT Nodes from which given file name will be read or written.
 - Ex:- if WriteDirectory is /export/scratch/ and file name is world.txt then file will be stored in the location /export/scratch/ with name world.txt
- Verbose: Shows complete log (visited nodes) for the request.

HOW TO RUN COMPONENTS?

Put components on Unix

- scp all the files on the server.
- The file structure on server should be -
 - Project Folder
 - jars (folder)
 - slf4j-api-1.7.14.jar
 - libthrift-0.9.1.jar
 - files (folder)

- all files

Generate Stub files

- *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.

Compile files

```
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.

Make sure that you start SuperNode, each node and Client in separate sessions(different terminal tabs)

Start Super-Peer

```
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.

Join a node to 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 commands, we have basically established DHT network and now the system is ready to receive and process the requests.

Write Files

```
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.

Read Files

```
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.

Read/Write Files One by One

```
java -cp ../jars/libthrift-0.9.1.jar../jars/slf4j-api-1.7.14.jar:gen-java/ Workers config.txt  
<OType> <filename>
```

example:-

- java -cp ../jars/libthrift-0.9.1.jar../jars/slf4j-api-1.7.14.jar:gen-java/ Worker config.txt 0 Upright.txt
- java -cp ../jars/libthrift-0.9.1.jar../jars/slf4j-api-1.7.14.jar:gen-java/ Worker config.txt 1 Upright.txt

OType -> 1 for writing the file **filename** to DHT.

OType-> 0 for reading the file **filename** to 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 be processed on DHT.
- Following scenarios are possible
 - **Node-Id** is same as **File-Id**. In this case file is processed on current node.
 - **Node-Id** is immediate successor of **File-Id**. In this case 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**.
 - 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. All test cases worked correctly.
- Tried to read the file that was **not present** in DHT. In this case “NIL” was returned as output.

Below are the screenshots for the test cases tested and logs generated.

LOG GENERATED

Log is generated on each terminal where the session is open. For example, on the terminal where Super Node is running will keep on printing its status such as active node list in the DHT and any operation performed by it. Similar is the case for each node and client.

A sample log at SuperNode

```

Requesting for connection .....
Acknowledging after joining DHT ....
Currently connected Node in DHT ...
-----
      HostName                Port      NodeId
-----
csel-x29-06.cselabs.umn.edu   9091       17
-----
csel-x29-04.cselabs.umn.edu   9091       21
-----
csel-x29-03.cselabs.umn.edu   9091       23
-----
csel-x29-02.cselabs.umn.edu   9091       25
-----

```

Sample logs at a DHT Node

```
sadal005@csel-x29-06:~/DS1$ java -cp ../jars/libthrift-0.9.1.jar:../jars/slf4j-api-1.7.14.jar:gen-java/ DHTNode config.txt
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Waiting for connection .....
Finger Table for Node with HostName :- csel-x29-06.cselabs.umn.edu
-----
      HostName      Port      NodeId
-----
csel-x29-04.cselabs.umn.edu    9091        21
-----
csel-x29-04.cselabs.umn.edu    9091        21
-----
csel-x29-03.cselabs.umn.edu    9091        23
-----
csel-x29-06.cselabs.umn.edu    9091        17
-----
csel-x29-06.cselabs.umn.edu    9091        17
-----
Previous Node:-    csel-x29-02.cselabs.umn.edu    9091    25

Contacting node with IP csel-x29-04.cselabs.umn.edu and nodeId 21
Updated finger table for node with IP csel-x29-04.cselabs.umn.edu and nodeId 21

Contacting node with IP csel-x29-03.cselabs.umn.edu and nodeId 23
Updated finger table for node with IP csel-x29-03.cselabs.umn.edu and nodeId 23

Contacting node with IP csel-x29-02.cselabs.umn.edu and nodeId 25
Updated finger table for node with IP csel-x29-02.cselabs.umn.edu and nodeId 25

Starting thrift server at host csel-x29-06.cselabs.umn.edu
Starting the simple server...
```

```
Reading file Icicle.txt with hash value 25 from node csel-x29-02.cselabs.umn.edu
Reading file Affront.txt with hash value 25 from node csel-x29-02.cselabs.umn.edu
Reading file Staunch.txt with hash value 25 from node csel-x29-02.cselabs.umn.edu
Finger Table for Node with HostName :- csel-x29-02.cselabs.umn.edu
-----
      HostName      Port      NodeId
-----
csel-x29-06.cselabs.umn.edu    9091        17
-----
csel-x29-06.cselabs.umn.edu    9091        17
-----
csel-x29-06.cselabs.umn.edu    9091        17
-----
csel-x29-06.cselabs.umn.edu    9091        17
-----
csel-x29-06.cselabs.umn.edu    9091        17
-----
Previous Node:-    csel-x29-03.cselabs.umn.edu    9091    23
```

Sample log at Client

```

Initial Connection to csel-x29-02.cselabs.umn.edu on port 9091
Reading file Disable.txt FROM DHT
File Content :- Disable
1. csel-x29-02.cselabs.umn.edu:9091
2. csel-x29-04.cselabs.umn.edu:9091 [File Read from this Node]

Initial Connection to csel-x29-02.cselabs.umn.edu on port 9091
Reading file Carriage.txt FROM DHT
File Content :- Carriage
1. csel-x29-02.cselabs.umn.edu:9091
2. csel-x29-04.cselabs.umn.edu:9091
3. csel-x29-03.cselabs.umn.edu:9091 [File Read from this Node]

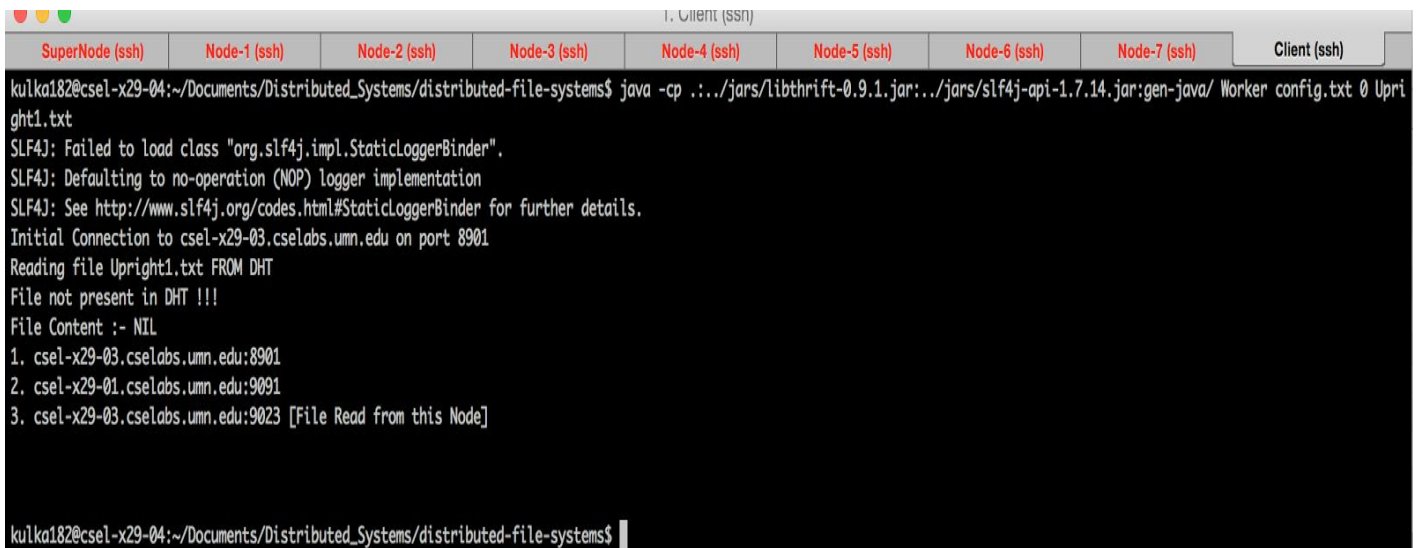
sadal005@csel-x29-05:~/DS1$

```

A sample of File missing

The test case can be tested by adding an additional file in data folder and running the Read command below. As we haven't wrote file to any node, we will get below error.

java -cp ../jars/libthrift-0.9.1.jar:../jars/slf4j-api-1.7.14.jar:gen-java/ ReadClient config.txt



```

1. Client (ssh)
SuperNode (ssh) Node-1 (ssh) Node-2 (ssh) Node-3 (ssh) Node-4 (ssh) Node-5 (ssh) Node-6 (ssh) Node-7 (ssh) Client (ssh)
kulka182@csel-x29-04:~/Documents/Distributed_Systems/distributed-file-systems$ java -cp ../jars/libthrift-0.9.1.jar:../jars/slf4j-api-1.7.14.jar:gen-java/ Worker config.txt 0 Upright1.txt
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Initial Connection to csel-x29-03.cselabs.umn.edu on port 8901
Reading file Upright1.txt FROM DHT
File not present in DHT !!!
File Content :- NIL
1. csel-x29-03.cselabs.umn.edu:8901
2. csel-x29-01.cselabs.umn.edu:9091
3. csel-x29-03.cselabs.umn.edu:9023 [File Read from this Node]

kulka182@csel-x29-04:~/Documents/Distributed_Systems/distributed-file-systems$

```


A sample of SuperNode not running

If we run a join node command without starting Super Node, we will get below error.

SuperNode (ssh)	Node-1 (ssh)	Node-2 (ssh)	Node-3 (ssh)	Node-4 (ssh)	Node-5 (ssh)	Node-6 (ssh)	Node-7 (ssh)	Client (ssh)
<pre>kulka182@csel-x29-01:~/Documents/Distributed_Systems/distributed-file-systems\$ java -cp ../jars/libthrift-0.9.1.jar:../jars/slf4j-api-1.7.14.jar:gen-java/ DHTNode config.txt SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder". SLF4J: Defaulting to no-operation (NOP) logger implementation SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details. ===== Unable to establish connection with SuperPeer ... Exiting ... ===== kulka182@csel-x29-01:~/Documents/Distributed_Systems/distributed-file-systems\$</pre>								

A sample of maximum limit

This test case can be tested by changing configuration in config.txt. We tested by setting the MOD value to 3 and finger table size to 3. Then we tried to add 4 nodes and it failed while adding 4th node.

```
iTerm Shell Edit View Profiles Toolbelt Window Help
1. Node-6 (ssh)
SuperNode (ssh) Node-1 (ssh) Node-2 (ssh) Node-3 (ssh) Node-4 (ssh) Node-5 (ssh) Node-6 (ssh) Node-7 (ssh) Client (ssh)
kulka182@csel-x29-03:~/Documents/Distributed_Systems/distributed-file-systems$ java -cp ../jars/libthrift-0.9.1.jar:../jars/slf4j-api-1.7.14.jar:gen-java/ DHTNode config.txt 8901
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Waiting for connection .....
DHT has reached its maximum capacity and hence no new node can join the network
kulka182@csel-x29-03:~/Documents/Distributed_Systems/distributed-file-systems$
```

