Programming Assignment 1: Book Finder System on a DHT

CSCI 5105: Introduction to Distributed Systems

Spring 2019

Instructor: Abhishek Chandra

April 5, 2019

Team:

Harshit Jain (jain0149@umn.edu)

Vaybhav Shaw (shaw0162@umn.edu)

Table of Contents

1. Overview

The goal of this project is to implement a simple Book Finder System using Thrift and Java. The project helped us to understand RPC communications, Map Reduce operation and other distributed systems fundamentals.

The stack used for analysis:

1. **Thrift** - To build distributed services.
2. **Java** - For coding
3. **CSE Lab Machines** - To host server and worker nodes

2. Components Implemented

In this section, we present a brief overview of the various components used.

2.1 Client (Client.java)

The client will be responsible for setting book titles and genres to the system as well as getting a genre from the system with a book title from a sample file available with the client. The client also gets an arbitary node from SuperNode which then goes on to resolve the location of the book using DHT.

2.2 Node (Node.java)

The Node receives requests from Client to find book genre and set book title and genre. The node also sends request to SuperNode to join and be part of DHT.

2.3 Super Node (SuperNode.java)

SuperNode receives requests from Node to join the DHT. The SuperNode in turn returns an arbitrary node from DHT to Node for him to join the DHT. The SuperNode also returns an arbitary node to Client which it uses for book finding and storing.

2.4 SuperNodeHandler (SuperNodeHandler.java)

The SuperNode Handler implements the functionality of SuperNode. It first sends an arbitrary node to Node to join. It also implements Post join after which only new node can join the

network. And after the end it sends an Update DHT to all the notice to all the nodes to update their finger tables.

2.5 NodeHandler (NodeHandler.java)

The NodeHandler implements the functionality of Node including finding the successor, finding predecessor, updating finger tables, updating finger tables of other nodes, finding books, setting book etc.

2.6 ClientNodeInterface(ClientNodeInterface.java)

It provides relevant methods for Node interface with Client and SuperNode.

2.7 SuperNodeInterface (SuperNodeInterface.java)

SuperNodeInterface provides methods for SuperNode interface with Client and Node.

3. Workflow

We created two thrift files and generated two services for communication between Client, SuperNode and Node. We then created the handlers for each SuperNode and Node.

Detailed Workflow:

1. The Node sends a request to SuperNode to join the DHT.
2. The DHT returns an arbitrary node to Node which it uses to initialise its finger tables and predecessor.

3. After initialising its finger tables it than recursively calls other Nodes to update their finger tables and successor and predecessor.
4. After it has joined the network it calls the Post Join to tell that it has successfully joined the DHT and then makes the request to SuperNode to send an Update DHT request.
5. Once the DHT is updated the Client is now ready to populate the DHTs by sending book details.
6. The Books are then hashed into the DHT by using the SHA1.
7. The client makes a request to SuperNode. The SuperNode sends an arbitrary node to the Node. The client then makes a call to the Node to set the books. It also can use the arbitrary node to get the book.

4. Running the Code

We created a config.file which specifies the following:

1. First line mentions the number of bits (set to 5 initially, spans 0-31). Limitation of int: bits can not be more than 32 (spans 0 to 2^32-1).
2. Second line mentions the IP port number of SuperNode. (eg: **localhost 9099**) which is separated by a single space

To run the code, we need to follow the following steps:

(Everything needs to be run from the directory in which the files are present)

1. We first compile the project using the following syntax :
   javac -cp ".:/usr/local/Thrift/*" *.java -d .
2. We then run the SuperNode using the following command
   java -cp ".:/usr/local/Thrift/*" SuperNode
3. We then run each of the Node of the DHT, for eg:
   java -cp ".:/usr/local/Thrift/*" Node localhost 8080
   java -cp ".:/usr/local/Thrift/*" Node localhost 9085

java -cp ".:/usr/local/Thrift/*" Node localhost 9095

(Note: We pass two parameters, one ip and other port in the command line)

4. Next, we then run the Client on the local node to set the book details using:

   java -cp ".:/usr/local/Thrift/*" Client set shakespeares.txt

   (Note: We pass two parameters, one *set* and other the <fileName> which has the book details, separated by a space)

5. Next, we run the Client on the local node to get the book genre using:

   java -cp ".:/usr/local/Thrift/*" Client get Venus and Adonis

   (Note: We pass two parameters, one *get* and other the <bookName>)

5. Testing

Scenario 1 :

1. **When we join the Nodes to the SuperNode, we get the updated finger tables for all nodes:**

```
Node Id: 113
IP and Port Details: localhost:3654
Predecessor of the Node: localhost:9010
Successor of the Node: localhost:9036
Printing the Finger Table :
start: 114      interval_start: 114     interval_end: 114   successor: localhost:9036       successorId: 153
start: 115      interval_start: 115     interval_end: 116   successor: localhost:9036       successorId: 153
start: 117      interval_start: 117     interval_end: 120   successor: localhost:9036       successorId: 153
start: 121      interval_start: 121     interval_end: 128   successor: localhost:9036       successorId: 153
start: 129      interval_start: 129     interval_end: 144   successor: localhost:9036       successorId: 153
start: 145      interval_start: 145     interval_end: 176   successor: localhost:9036       successorId: 153
start: 177      interval_start: 177     interval_end: 240   successor: localhost:9095       successorId: 45
start: 241      interval_start: 241     interval_end: 112   successor: localhost:9095       successorId: 45
```

Scenario 2 :

When we run the set command from the client, its sets the <title, genre> on all nodes

```
Node Id: 153
IP and Port Details: localhost:9036
Predecessor of the Node: localhost:3654
Successor of the Node: localhost:9095
Printing the Finger Table :
start: 154      interval_start: 154     interval_end: 154successor: localhost:9095       successorId: 45
start: 155      interval_start: 155     interval_end: 156successor: localhost:9095       successorId: 45
start: 157      interval_start: 157     interval_end: 160successor: localhost:9095       successorId: 45
start: 161      interval_start: 161     interval_end: 168successor: localhost:9095       successorId: 45
start: 169      interval_start: 169     interval_end: 184successor: localhost:9095       successorId: 45
start: 185      interval_start: 185     interval_end: 216successor: localhost:9095       successorId: 45
start: 217      interval_start: 217     interval_end: 24successor: localhost:9095        successorId: 45
start: 25       interval_start: 25      interval_end: 152successor: localhost:9095       successorId: 45
Book Title: As You Like It      Hash_Of_The_Book :132
Book Title: Love's Labor's Lost Hash_Of_The_Book :128
Book Title: The Two Gentlemen of Verona Hash_Of_The_Book :120
Book Title: The Two Noble Kinsmen       Hash_Of_The_Book :134
Book Title: Henry V     Hash_Of_The_Book :114
Book Title: King John   Hash_Of_The_Book :132
Book Title: Hamlet      Hash_Of_The_Book :151
```

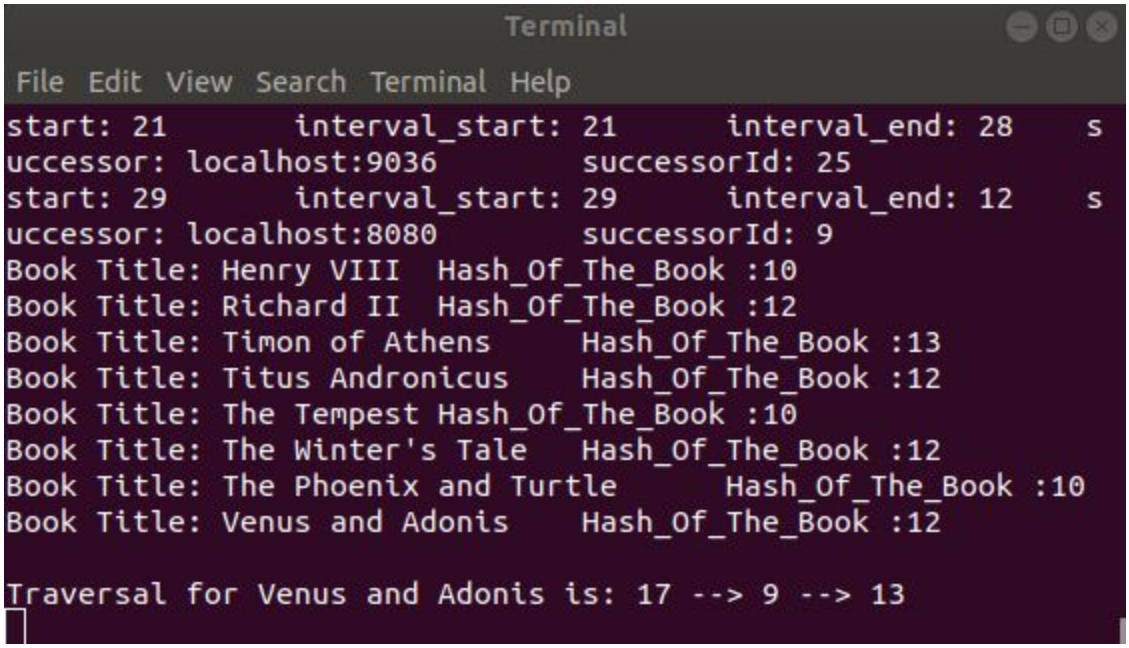
Scenario 3 :

- *When we run the get command from the client, its gets the genre from the respective node*

```
jain0149@csel-kh4250-05:/home/jain0149/PA2_init_n $ java -c
p ".:/usr/local/Thrift/*" Client get Venus and Adonis
localhost 9099
Genre: Poems
```

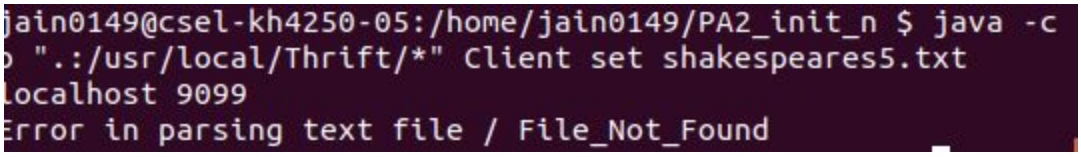
Scenario 4 :

- *When we run the get command from the client, its displays the hop-path on the node from which it is get - ting the value*



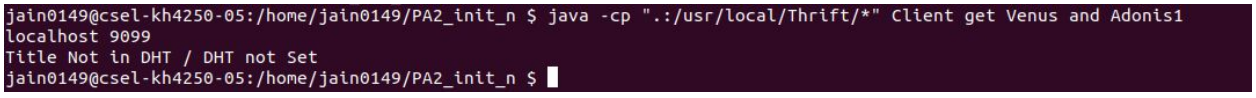
Scenario 5 (Negative Test Case) :

If the filename is incorrect in the client command line, it throws an error



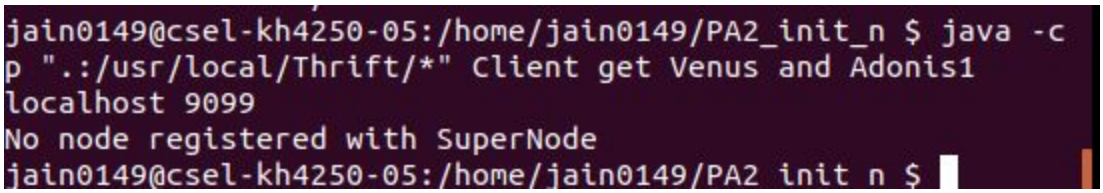
Scenario 6 (Negative Test Case):

When the title is not present in the DHT or the get is called before set, it shows a graceful message



Scenario 7 (Negative Test Case):

When no node is connected to the SuperNode and the client requests for a random node, it shows a message



Scenario 8 (Negative Test Case):

When the port number entered is not a valid integer, it shows an error