User Document

Running each component of the system:

PLEASE ENSURE THAT YOU HAVE PLACED THE CODE IN A FOLDER WHERE YOU HAVE WRITE ACCESS, since our file system is persistent.

Step 1: Compile the system using the compile script provided to generate all the .class files

./compile

Step 2: Run the coordinator in the first terminal window, giving its port number, NR, NW and N (total number of file servers) as command line arguments.

java -cp ".:./libthrift-0.9.3.jar:./slf4j-api-1.7.14.jar" CoordinatorServer <port number> <n> <nr> <nw>

Step 3: Start seven or more file servers (as many as needed) on different terminals, giving its port number and the IP and port of the coordinator as command line arguments.

java -cp ".:./libthrift-0.9.3.jar:./slf4j-api-1.7.14.jar" FileServer <port number> <coordinator IP> <coordinator port>

Step 3: Start the client using the following command.

java -cp ".:./libthrift-0.9.3.jar:./slf4j-api-1.7.14.jar" Client

At this point, *the system has been started* and you get a UI on the client asking you for input to run the system in a particular way. This has been explained in the next section.

To delete the folders and files created by the file system, there is a cleanup script provided. You can execute this script to delete all folders and the files inside them.

User input interface:

1) **Coordinator:** There is no input interface here as such, but you can see the synch operation printing out a message every time it runs.

2) **FileServer:** Every FileServer has an option to print out the map it stores which contains filename and version mappings.

These details can be seen by giving an input of 1 at any FileServer.

The details are printed as Filename – Version number for every file stored at the server.

3) **Client:** The client needs a lot of information before it starts the execution. The explanation of the client UI follows:

First, we need to input the number of clients we want to run. Ideally, this should be a sensible value (in the range of 1-10 or so)

After we input this value, the UI will ask for specific details needed for each client:

1) First, we need to input the IP address and the port of the file server that the client wants to connect to **ON SEPARATE LINES.**

2) We then input 0 if we want to do a read operation or 1 if we want to do a write operation for this client.

3) We then input the filename to be read or written.

4) Finally, we tell the client the number of such read or write requests it wants to send to the file server.

Once this 4 step process has been done for every client, the client will create n threads, where n is the number of clients input in the beginning, and then it will start sending the specified requests in an interleaved fashion through all these threads.