CSCI 5105, Spring 2022

(Introduction to Distributed Systems)

Programming Assignment 3

Submitted by:

Anish Shil

shil0037@umn.edu

Student ID: 5653115

Project User Document:

Status: Complete

Design Goal

The goal of this project is to design a implement a simple Distributed File System where multiple clients can share files together. The files would be replicated to multiple servers for increased performance and availability.

Working Components

Note: All implementations of the client, file server and coordinator nodes are done in Java

Coordinator

The coordinator node is a multithreaded file server which is a special server designated for handling other file server operations. The coordinator server is responsible for assembling a quorum of read and write servers based on a command line parameter specified at the time the coordinator is run. Once a consensus is reached within the quorum of read/write servers, a particular operation (read/write) can be carried out. The coordinator will impose sequential consistency to ensure writes done in a sequence are also read in the same sequence by clients.

File Server

A file server is a server which contains replicas of files. In a system, there are multiple file servers out of which one of them is designated to be the coordinator server, which gathers information to form quorum for reading and writing operations on arbitrarily selected file servers. In this project implementation example, 7 file servers have been used.

Client

A client is a simple Java class which is used to send out read/write requests to the group of servers assembled in the system. A client request can go to any random file server, but the request is acknowledged only when a certain subset of replicas (file servers) agree that they have most up-to-date data available in them. Multiple clients can concurrently try to read or write content to any of the file servers, but after all the operations the coordinator needs to ensure that up-to-date data is available on all replicas.

Assumption(s)

The following assumptions have been made in this assignment:

1. All components have been run on Walter Library 103 systems.
2. Port numbers from 9090 onwards are used to identify where the coordinator node and file servers would run. The coordinator is run on the hardcoded port 9091, whereas all file servers are run on port numbers taken as argument while running the ANT tasks.
3. The project java directory is /home/shil0037/Spring2022/CSCI5105/PA3/java
4. All build tasks are run from the above mentioned directory using ANT. The build file `build.xml` is placed here (details about running tasks using ANT described below).
5. The shared directory /export/scratch is used for sharing files for reading and writing among the clients. A new directory "shil0037" is created by the code for this purpose.

ANT Project

This project is built and run using Apache ANT. The build file contains targets for compiling the java classes, running each component (Java classes) with command line arguments, and cleaning the temp files and directories.

Here, the basic assumption is that all the tasks are to be run from the java directory (/home/shil0037/Spring2022/CSCI5105/PA3/java). The following is a brief description of the tasks which start the Client, Coordinator and the File servers respectively:

1. The task "client" is invoked as "ant client -DnumberOfOps=<opCount>" which in turn makes the following java call:

   java -classpath <pa3_classpath_entry_in_buildfile> Client <opcode> <opCount>

   The task itself runs the java class 4 times in parallel with different values for the opcode parameter to emulate different kinds of clients working concurrently to access the servers (read heavy, write heavy, balanced load, getting filesystem state).

2. The task "coordinator" is invoked as "ant coordinator -DnR=<nr> -DnW=<nw> -DnTotal=<nTotal>" which in turn makes the following java call:

   java -classpath <pa3_classpath_entry_in_buildfile> Coordinator <nr> <nw> <nTotal>

3. The task "fsnode" is invoked as "ant fsnode -Dport=<port_number>" which in turn makes the following java call:

   java -classpath <pa3_classpath_entry_in_buildfile> FileServer <port_number>

Towards the end of this document, multiple screenshots would be provided showing examples on how to invoke the ANT tasks, along with the state of the log files at different points and client execution sequences.

How to Test?

The project can be tested manually. The steps given below can be followed to do that:

1. Open about 9 terminal tabs (one for starting the coordinator, 7 for starting the file servers, and one for running the client).
2. Set the following env variable on all the open terminal windows:

   export THRIFT_LIB_PATH="/home/shil0037/Spring2022/CSCI5105/thrift-0.15.0/lib/java/build/libs" (adjust this to wherever your local thrift installation exists)

3. Compile all java files by running "ant compile" from the proj_dir directory.
4. Invoke the ant calls for all but the client (described in the previous section) in the open terminals. When the ant calls for the Coordinator and File servers would be run, the shell prompt would not come back since these commands start the servers, and the ant call does not end with an & (which would allow the terminal to become interactive again).
5. Verify the log files in {proj_dir}/build/log (proj_dir is the directory in Assumptions section #3)
6. Invoke the ant call for the client. The terminal should show the client initiate multiple parallel requests for read and write operations based on the opcode parameter. It would also show the state of the system after the specified number of operations have been carried out, along with the time elapsed in doing so.

Clean-up

Temporary log files and .class files can be cleaned up by running the "clean" target of the ANT build file. Run the following command to remove the aforementioned files/folder(s):

```
ant clean
```

All the files created by the execution of the client code for reads and writes would reside in /export/scratch/shil0037. This directory needs to be manually deleted after the project code has been run and tested.

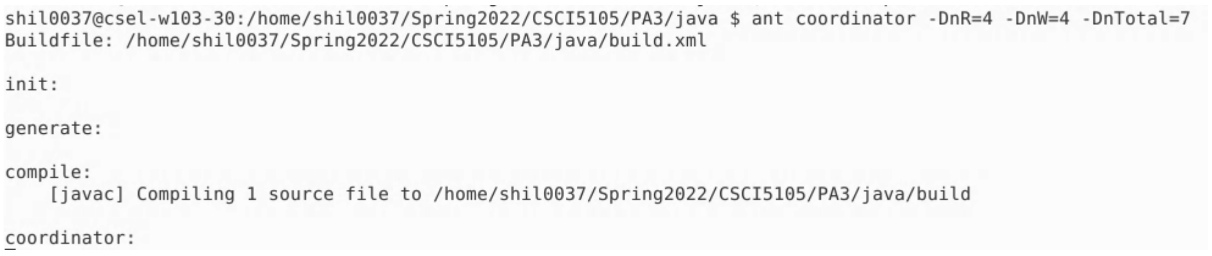
Test-Cases Attempted

Multiple clients were run concurrently with both read-heavy and write-heavy configurations. Four clients were executed in total - one being a read-heavy client, one being a write-heavy client, one being a client applying a balanced load (almost equal number of reads and writes), and one client fetching the state of the system after all reads and writes have been executed on the systems. The screenshots for these test cases have been included in the next section.

Screenshots for different execution instances

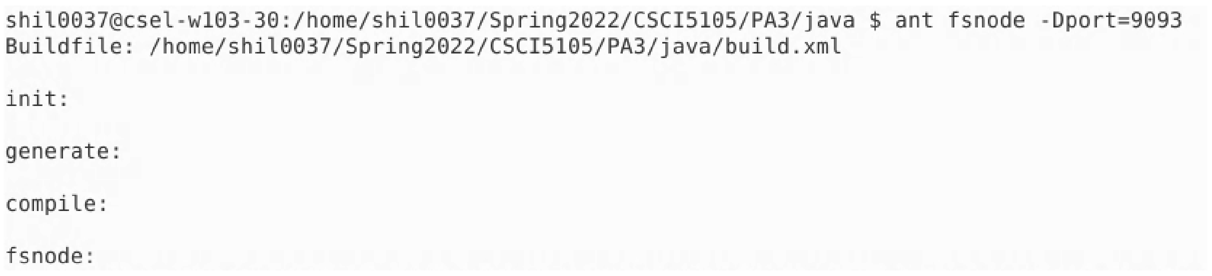
The following are a couple of screenshots from different execution points, such as starting the coordinator or file servers, log file contents, client executions for different load, etc.

- Starting the coordinator



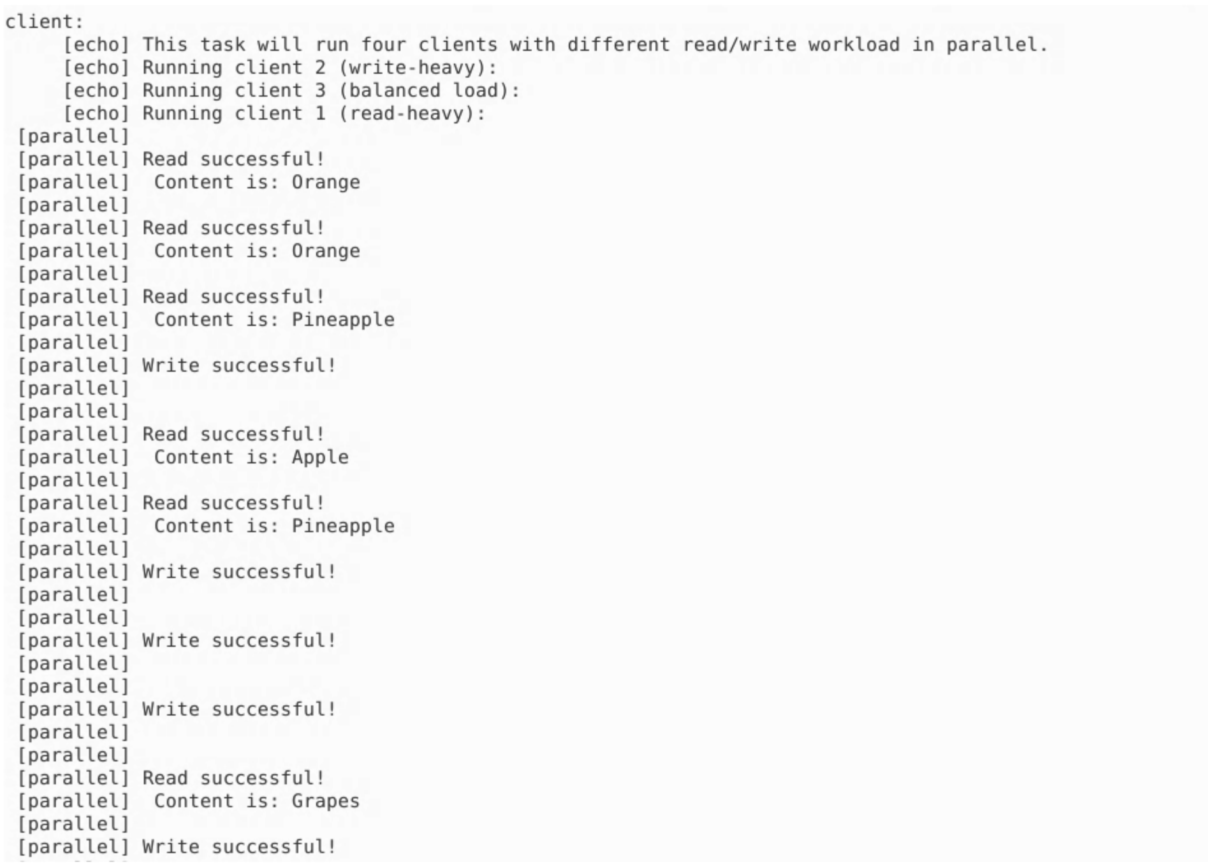
```
shil0037@csel-w103-30:/home/shil0037/Spring2022/CSCI5105/PA3/java $ ant coordinator -DnR=4 -DnW=4 -DnTotal=7
Buildfile: /home/shil0037/Spring2022/CSCI5105/PA3/java/build.xml

init:

generate:

compile:
    [javac] Compiling 1 source file to /home/shil0037/Spring2022/CSCI5105/PA3/java/build

coordinator:
```

- Starting the file server on a given port number



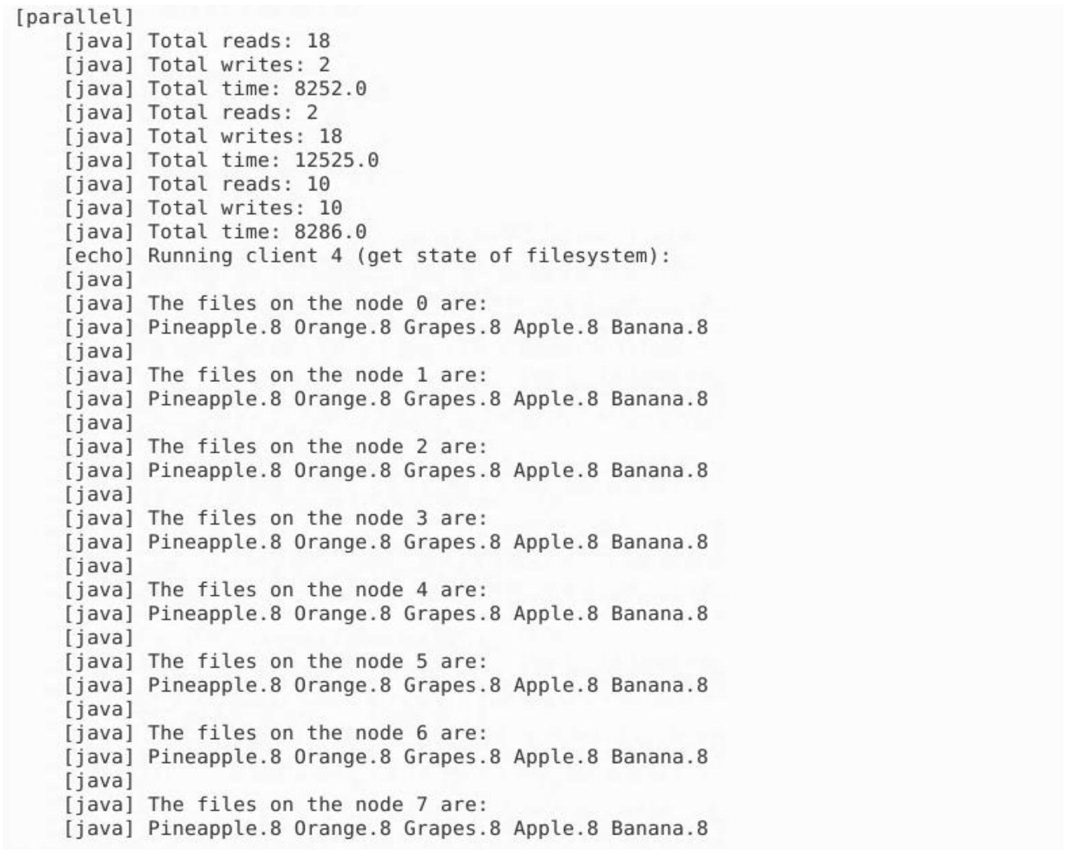
```
shil0037@csel-w103-30:/home/shil0037/Spring2022/CSCI5105/PA3/java $ ant fsnode -Dport=9093
Buildfile: /home/shil0037/Spring2022/CSCI5105/PA3/java/build.xml

init:

generate:

compile:

fsnode:
```

- Multiple clients execution showing concurrent reads and writes



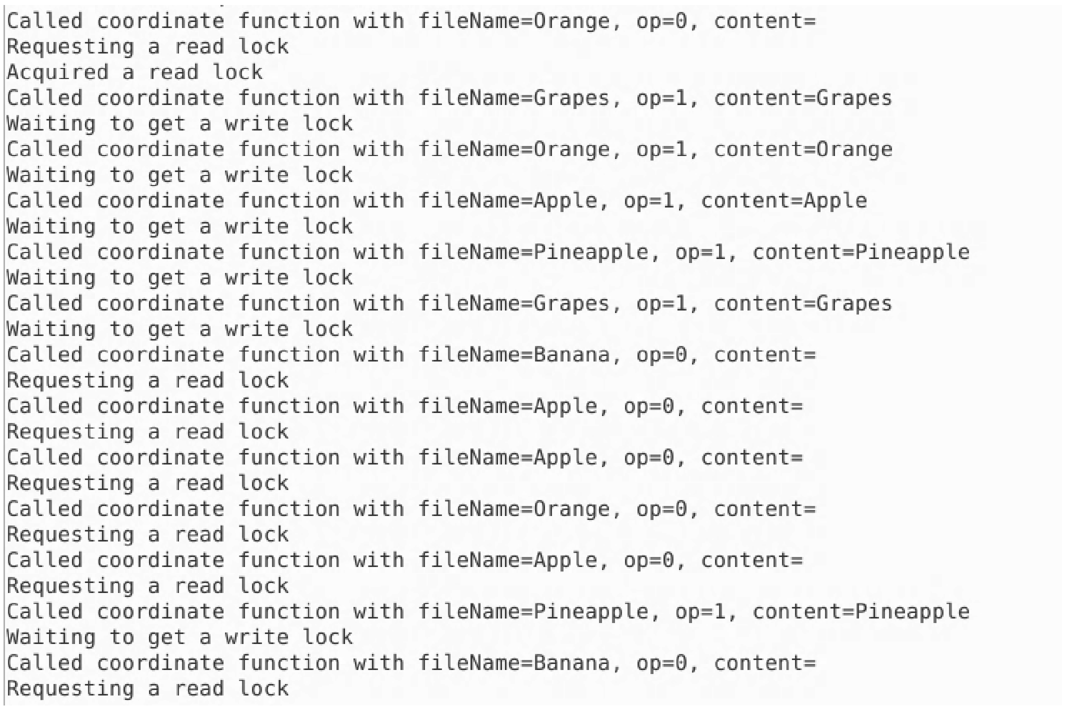
```
client:
     [echo] This task will run four clients with different read/write workload in parallel.
     [echo] Running client 2 (write-heavy):
     [echo] Running client 3 (balanced load):
     [echo] Running client 1 (read-heavy):
 [parallel]
 [parallel] Read successful!
 [parallel]  Content is: Orange
 [parallel]
 [parallel] Read successful!
 [parallel]  Content is: Orange
 [parallel]
 [parallel] Read successful!
 [parallel]  Content is: Pineapple
 [parallel]
 [parallel] Write successful!
 [parallel]
 [parallel]
 [parallel] Read successful!
 [parallel]  Content is: Apple
 [parallel]
 [parallel] Read successful!
 [parallel]  Content is: Pineapple
 [parallel]
 [parallel] Write successful!
 [parallel]
 [parallel]
 [parallel] Write successful!
 [parallel]
 [parallel]
 [parallel] Write successful!
 [parallel]
 [parallel]
 [parallel] Read successful!
 [parallel]  Content is: Grapes
 [parallel]
 [parallel] Write successful!
```

- State of file system on each server and time taken for all reads/writes (20 operations)



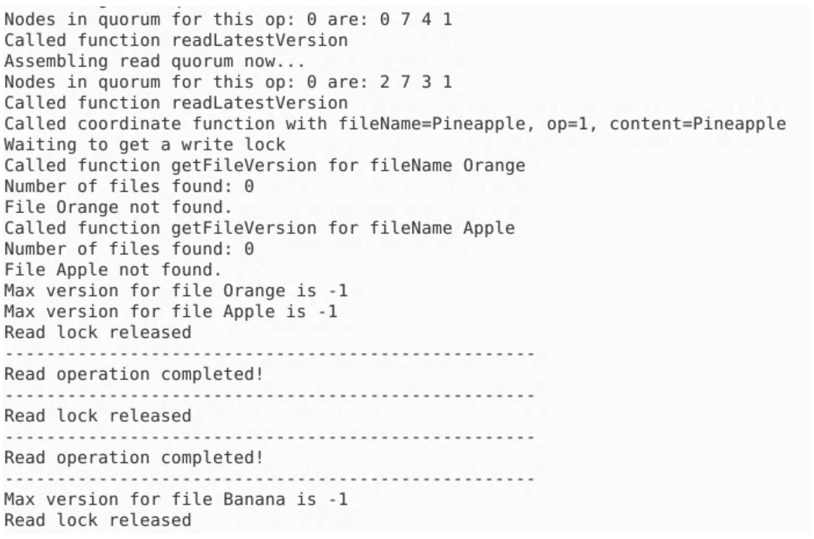
```
[parallel]
    [java] Total reads: 18
    [java] Total writes: 2
    [java] Total time: 8252.0
    [java] Total reads: 2
    [java] Total writes: 18
    [java] Total time: 12525.0
    [java] Total reads: 10
    [java] Total writes: 10
    [java] Total time: 8286.0
    [echo] Running client 4 (get state of filesystem):
    [java]
    [java] The files on the node 0 are:
    [java] Pineapple.8 Orange.8 Grapes.8 Apple.8 Banana.8
    [java]
    [java] The files on the node 1 are:
    [java] Pineapple.8 Orange.8 Grapes.8 Apple.8 Banana.8
    [java]
    [java] The files on the node 2 are:
    [java] Pineapple.8 Orange.8 Grapes.8 Apple.8 Banana.8
    [java]
    [java] The files on the node 3 are:
    [java] Pineapple.8 Orange.8 Grapes.8 Apple.8 Banana.8
    [java]
    [java] The files on the node 4 are:
    [java] Pineapple.8 Orange.8 Grapes.8 Apple.8 Banana.8
    [java]
    [java] The files on the node 5 are:
    [java] Pineapple.8 Orange.8 Grapes.8 Apple.8 Banana.8
    [java]
    [java] The files on the node 6 are:
    [java] Pineapple.8 Orange.8 Grapes.8 Apple.8 Banana.8
    [java]
    [java] The files on the node 7 are:
    [java] Pineapple.8 Orange.8 Grapes.8 Apple.8 Banana.8
```

- Log file showing coordinator trying to acquire locks for read/write operations



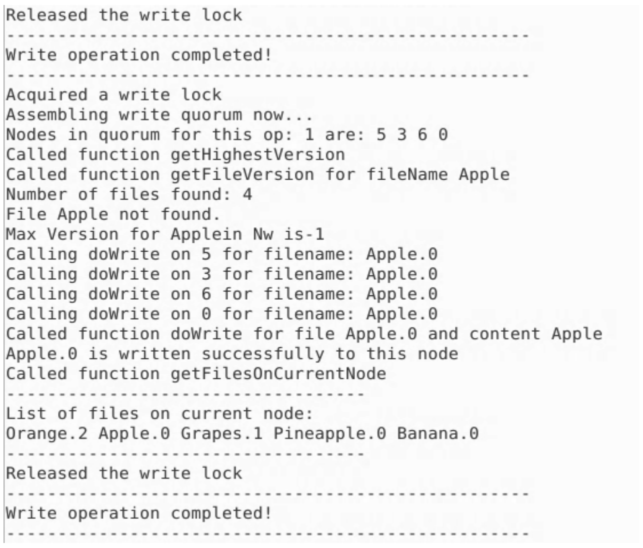
```
Called coordinate function with fileName=Orange, op=0, content=
Requesting a read lock
Acquired a read lock
Called coordinate function with fileName=Grapes, op=1, content=Grapes
Waiting to get a write lock
Called coordinate function with fileName=Orange, op=1, content=Orange
Waiting to get a write lock
Called coordinate function with fileName=Apple, op=1, content=Apple
Waiting to get a write lock
Called coordinate function with fileName=Pineapple, op=1, content=Pineapple
Waiting to get a write lock
Called coordinate function with fileName=Grapes, op=1, content=Grapes
Waiting to get a write lock
Called coordinate function with fileName=Banana, op=0, content=
Requesting a read lock
Called coordinate function with fileName=Apple, op=0, content=
Requesting a read lock
Called coordinate function with fileName=Apple, op=0, content=
Requesting a read lock
Called coordinate function with fileName=Orange, op=0, content=
Requesting a read lock
Called coordinate function with fileName=Apple, op=0, content=
Requesting a read lock
Called coordinate function with fileName=Pineapple, op=1, content=Pineapple
Waiting to get a write lock
Called coordinate function with fileName=Banana, op=0, content=
Requesting a read lock
```

- Log file showing the coordinator assemble an initial read quorum when no file is written. The read lock acquired earlier before forming the quorum is released when the file version is determined.

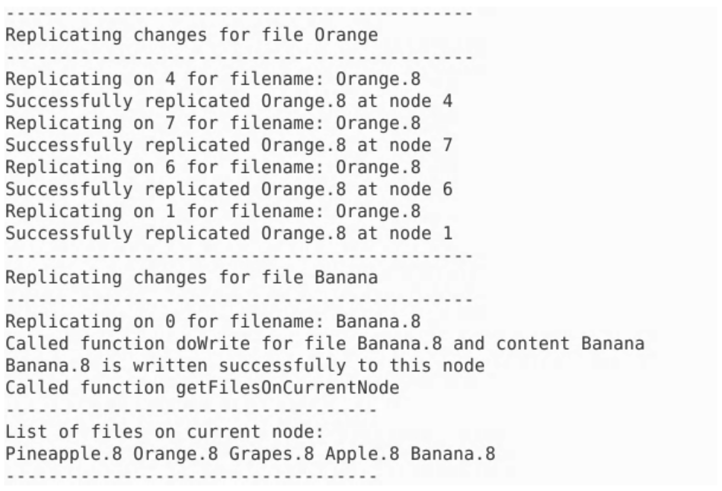


```
Nodes in quorum for this op: 0 are: 0 7 4 1
Called function readLatestVersion
Assembling read quorum now...
Nodes in quorum for this op: 0 are: 2 7 3 1
Called function readLatestVersion
Called coordinate function with fileName=Pineapple, op=1, content=Pineapple
Waiting to get a write lock
Called function getFileVersion for fileName Orange
Number of files found: 0
File Orange not found.
Called function getFileVersion for fileName Apple
Number of files found: 0
File Apple not found.
Max version for file Orange is -1
Max version for file Apple is -1
Read lock released
--------------------------------------------------
Read operation completed!
--------------------------------------------------
Read lock released
--------------------------------------------------
Read operation completed!
--------------------------------------------------
Max version for file Banana is -1
Read lock released
```

- Log files showing different versions of files on the servers after writing them, and replicating latest versions of files across different servers



```
Released the write lock
--------------------------------------------------
Write operation completed!
--------------------------------------------------
Acquired a write lock
Assembling write quorum now...
Nodes in quorum for this op: 1 are: 5 3 6 0
Called function getHighestVersion
Called function getFileVersion for fileName Apple
Number of files found: 4
File Apple not found.
Max Version for Applein Nw is-1
Calling doWrite on 5 for filename: Apple.0
Calling doWrite on 3 for filename: Apple.0
Calling doWrite on 6 for filename: Apple.0
Calling doWrite on 0 for filename: Apple.0
Called function doWrite for file Apple.0 and content Apple
Apple.0 is written successfully to this node
Called function getFilesOnCurrentNode
----------------------------------
List of files on current node:
Orange.2 Apple.0 Grapes.1 Pineapple.0 Banana.0
----------------------------------
Released the write lock
--------------------------------------------------
Write operation completed!
--------------------------------------------------
```



```
------------------------------------------
Replicating changes for file Orange
------------------------------------------
Replicating on 4 for filename: Orange.8
Successfully replicated Orange.8 at node 4
Replicating on 7 for filename: Orange.8
Successfully replicated Orange.8 at node 7
Replicating on 6 for filename: Orange.8
Successfully replicated Orange.8 at node 6
Replicating on 1 for filename: Orange.8
Successfully replicated Orange.8 at node 1
------------------------------------------
Replicating changes for file Banana
------------------------------------------
Replicating on 0 for filename: Banana.8
Called function doWrite for file Banana.8 and content Banana
Banana.8 is written successfully to this node
Called function getFilesOnCurrentNode
----------------------------------
List of files on current node:
Pineapple.8 Orange.8 Grapes.8 Apple.8 Banana.8
----------------------------------
```