Member-1: Neelkumar Shah
UFID: 1584-6803

Member-2: Jeshalraj Thakaria
UFID: 8321-3912

HW-4 Part-B Distributed Remote File System

Report

The problem statement i.e. to build a distributed remote file system was divided into two major modules: building a Metaserver and building *N* Dataservers.

1. Metaserver

1.1. Design for implementation of metaserver

1.1.1. Initial tests for server connection, and to-and-fro data transfer to server using xmlrpclib

- Build a server: python simpleht.py --port 2222
- Test Program (in terminal window):

```
import pickle, xmlrpclib
from xmlrpclib import Binary
rpc = xmlrpclib.ServerProxy("http://localhost:2000/")
d1 = {'a':'text1', 'b':'text2'}
path1 = 'dir1/dir2/abc'
sData = pickle.dumps(d1)                    #Marshall Data
rpc.put(Binary(path1), Binary(sData))       #add to server
rd1 = rpc.get(Binary(path1))                #get from server
rd1 = pickle.loads(rd1.data)                #Unmarshall Data
print rd1
```

1.1.2. Changes to HierarchicalBlockFS.py to create distributedFS.py to store metadata to metaserver

- Initialized an instance of metaserver, *rpc_met*, in __init__(self).
- Modification to the traverseparent() method to return an additional data i.e. path of parent.
- In order to keep track of subdirectories and files, datatype of *files* was changed to list and was operated when needed.
- In main(), port number for metaserver was fetched from argument given in command

```
m_port = argv[2]
```

1.1.3. Changes to simpleht.py to create metaserver.py

- Added a new method to delete a value from hashtable in metaserver using .pop() method

```
def delete_k(self, key):
    self.data.pop(key.data)
    return True
```

1.2. Tests for verifying the functionality of metaserver

- Directories created on multiple levels.
- All the functions related to directories and metadata such as chmod, create, getxattr, listxattr, mkdir, rmdir, readdir, getattr, removexattr and utimens were tested on multilevel directories.

2. Dataserver

2.1. Design for implementation of metaserver

2.1.1. Additional modifications to distributedFS.py to store data to 'N' dataserver

- Initialized a list of instances of dataserver, *rpc_dat*, in __init__(self).
- Developed a hashfunction named *hashit.* The key of the table is path of directory or file, while the value is the data itself.
- Logic to assign dataserver to a particular data.
  Hashfunction considers ASCII value of individual characters in the path and adds it.
  IndexOfDataServer = Sum(ASCII(path)) % NumberOfDataServers
- In main(), port number for dataservers were fetched from argument given in command

```
d_port = list()
    for x in range(3,len(argv)):
     d_port.append(argv[x])
```

2.1.2. Changes to simpleht.py to create dataserver.py

- Added a new method to delete a value from hashtable in metaserver using .pop() method
- Modification to .get() method. if rv={}: return False

2.2. Tests for verifying the functionality of dataserver

- Hashfunction was tested for its equal load distribution to number of servers ranging from 2 to 5.
- All the file related functions such as, write, unlink, symlink, readlink, read, truncate were tested for number of text files ranging from a few Bytes to 104 KB.
- All kind of file operation were performed, like truncate, append, overwrite, etc at different levels of directories.
- Hardlink and symlinks were also tested on multiple levels and were working successfully.

Tests for rename() method:

- Large blocks of files could be successfully moved and copied using rename method.
- Tested by using *mv* command to move the files at multiple levels of directories.
- Tested by renaming the files at same level as well as different levels in the file system hierarchy.
- Copied files using cp command at same level as well as different levels in the file system hierarchy.

➢ Common approaches in methods.

- Getattr()

  Get the metadata of file from meta server and check if the key exists at server. If key doesn't exist raise error, else return attributes.

- Write ()

  Steps:

  1. Calculate size and no of blocks from metadata.
  2. Get all the file blocks make changes and upload it back.