PROJECT Testing and Execution

Execution:

Simpleht.py has been modified, make sure the simpleht.py attached is used

1. Launch the servers:

./launch_procs.py 2 or python launch_procs.py 2

Where python launch_procs.py <num_servers>

num_servers represents the number of servers within each server bank

2. **python refresh_v1.py fusemount 'http://localhost:8000' 2**

Python refresh_v1.py <mount point> <start port http addr> <num of servers>

Make sure the same number of servers is used in both launch_procs.py and refresh_v1.py. Run this in the directory where the fuse library is stored.

The ports are hardcoded in the code as port number 8000, 7000 and 9000. When you specify the number of servers, eg say 3: then port numbers 7000, 7001, 7002 , 8000, 8001, 8002, 9000, 9001, 9002 get executed.

3. To exit, use the kill_servers.sh

./kill_servers.sh

This will kill all the servers which are running.

Testing:

The following commands were tested for normal functioning of the file system

Regular File System Functions

Cd fusemount
Echo "test" > test
Cat test
Ls
Ln –s test file
Unlink file
Rm test

If the file is deleted before the symlink, then the symlink becomes red in color! Does not have any value !

Mkdir d
Echo "Hi" > hi
Mv hi li
Cp li ji
Ls –ltr
Touch hi

Chmod 0777 hi
Ls –ltr

Case 1: Server Down :

Kill any one server, or upto one cluster of servers (eg, 8000, 8001, 8002)

Access the contents in fusemount, the file system works using the majority voter.

Corrupting data in any of the remaining two servers, results in a failure

Ps ax
Kill <pid> ##**Kill the process which runs python simpleht.py --port = 8000 and not xterm –e**

Now return to the fusemount directory and type

ls *#Contents will be displayed*

Case 2: Server UP:

Now get the servers up

python simpleht.py --port 8000
python simpleht.py --port 8001

Type ls in the fusemount directory or

Wait for 60s till the system refreshes

Accessing server directly

The contents of the server can be viewed by opening terminal and python interpreter

python
import xmlrpclib
from xmlrpclib import Binary
s = xmlrpclib.ServerProxy('http://localhost:8000')
s.print_content() #this lists the contents in the particular server window, the current terminal will just display True

Corruption of Data in the file System:

Open a new terminal, keeping the file system running and corrupt the data by proxying into one of the servers

Open Terminal

python
import xmlrpclib
from xmlrpclib import Binary
s = xmlrpclib.ServerProxy('http://localhost:8000')
s.print_content() #this lists the contents in the particular server
From the print_content(), look for the file names and corrupt one of the files, or corrupt '/' file, which will be present in 8000,7000,9000
s.put(Binary('/').Binary('Corruption'),1000)

Type ls in fusemount or wait for 60s until refresh is done.

When the server refreshes every 60s, or when the particular server is accessed, the contents will be restored from the remaining two copies

Any new file written into any server through ServerProxy will not be reflected in the File system even if it has a majority (Corruption in multiple servers)

Install iozone:

sudo apt-get install iozone3

cd fusemount

iozone -a -+u -n 4k -g 16k -b ../file.xls

SNAPSHOTS:

The descriptions are written below the figures.

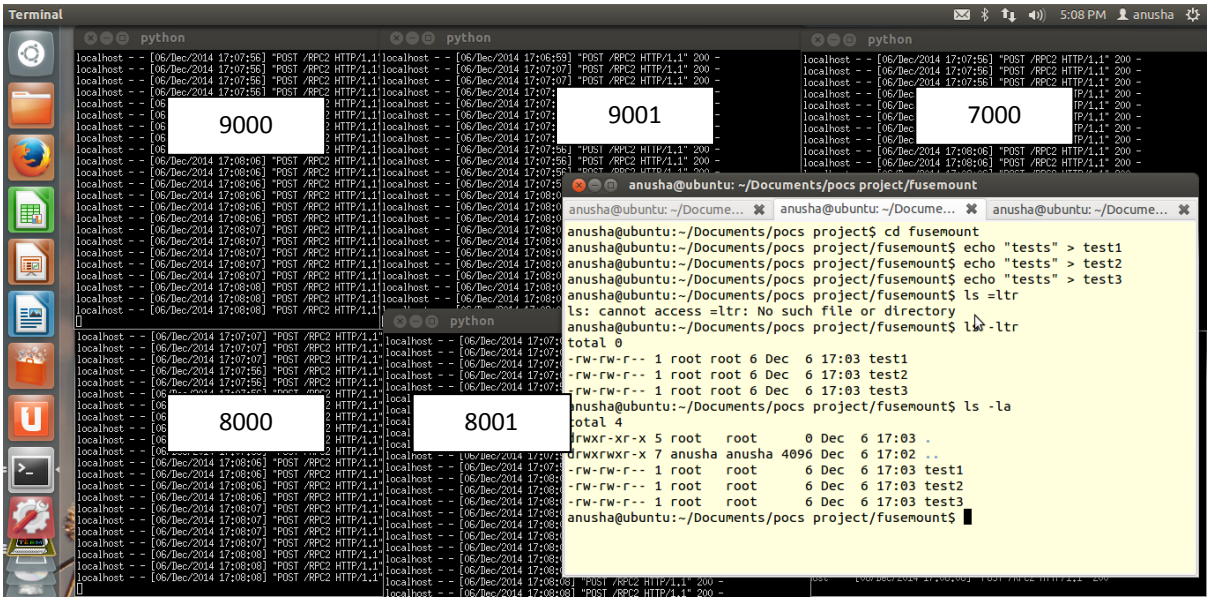

Fig1 : All 6 servers up and running.

Fig 1 displays all the 5 servers are running, and the mounted file system has 3 files saved in it.

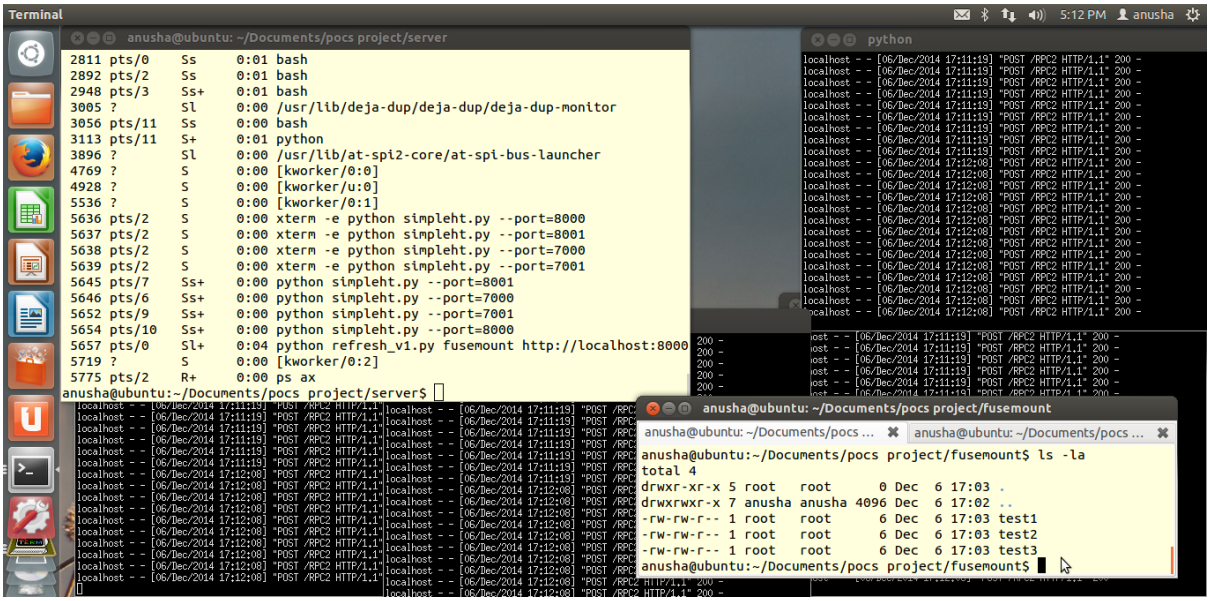

Fig 2: Both servers down

When one cluster of servers are down, the majority still exists and the file system functions normally.

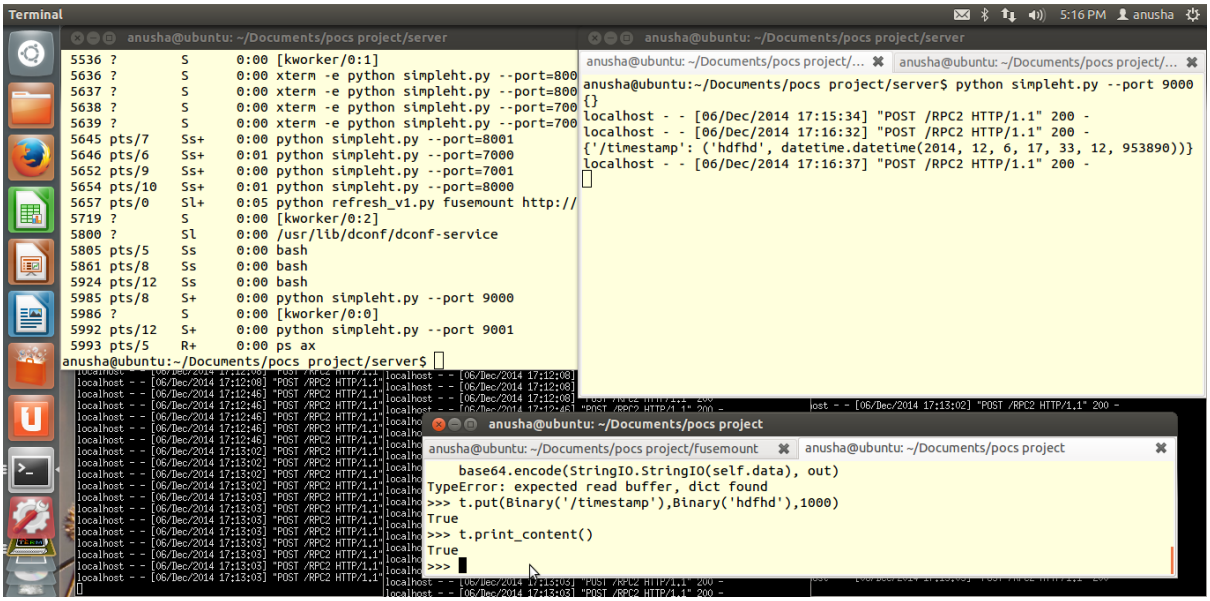

Fig 3: the servers 9000 and 9001 are up

The servers 9000 and 9001 are up and have no files in them. Either a file access or a periodic refresh restores the contents of the servers. In this case, the servers are restored by the refresh function. The figure shows an external entry added to create a time stamp when the server is pulled back up. The next image displays the data after refresh gets executed. When this is updated the file metadata is maintained to be same as the metadata on the other servers, i.e. it does not get altered.

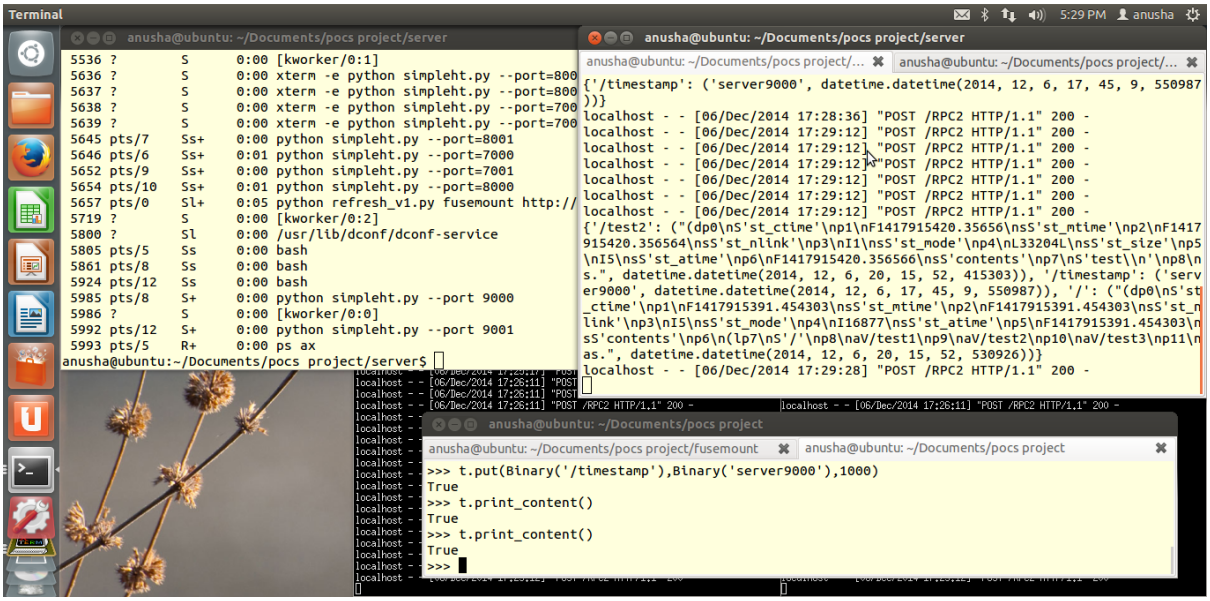

Fig 4 Servers after-Refresh

The servers 9000 and 9001 are refreshed and their contents are brought back up using the two unaltered copies which exist in two other server clusters(namely 8000 and 7000). The refresh also preserves any additional data that was written to the server (the timestamp entry in this example).

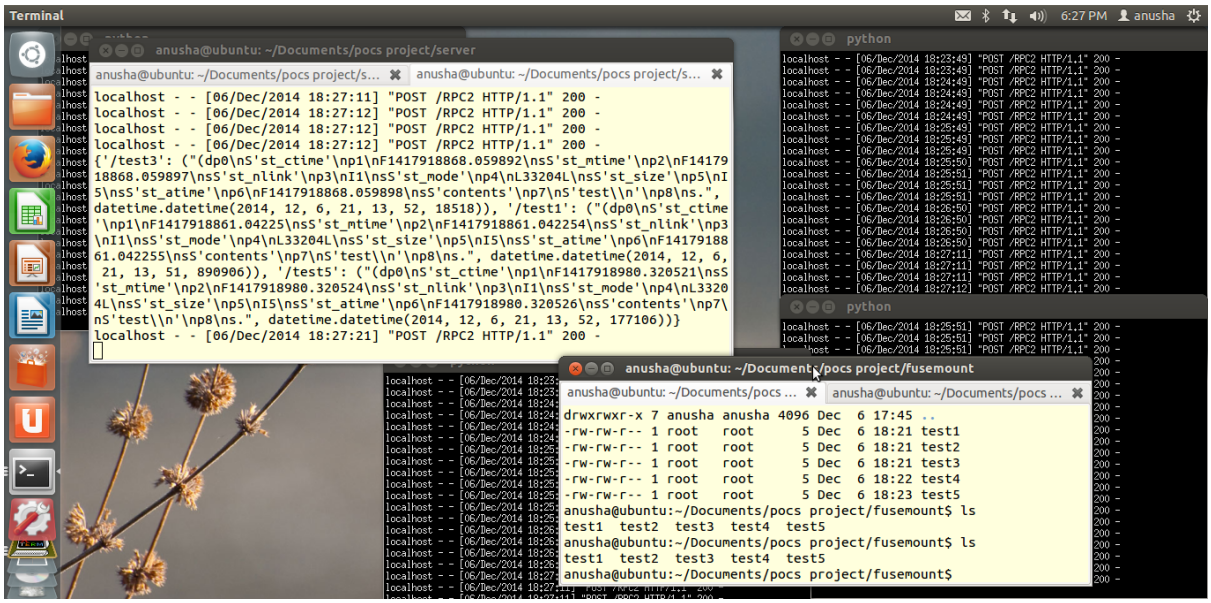

Fig 5 restore through access

Here, the contents of the server are restored through file access in the file system. When a file is accessed, two correct copies will exist in two servers. The status of the third server is checked and if it is up, the contents will be written on it.

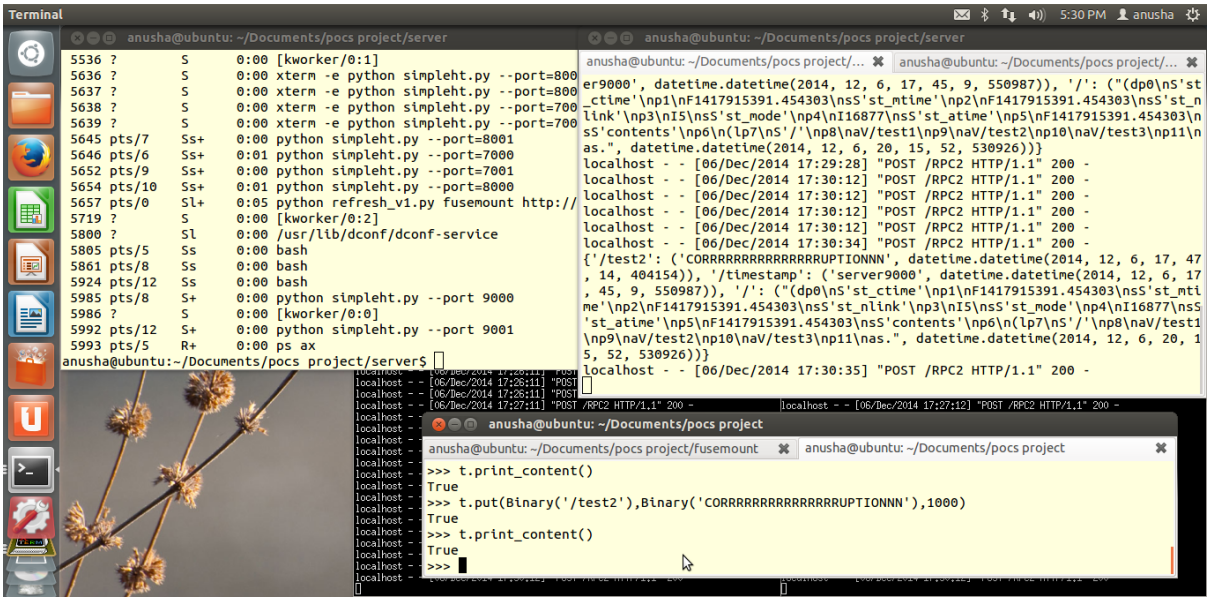

Fig 6 Corruption of a file in server 9000

A file '/test2' in server 9000 is corrupted. The contents and metadata of the server is corrupted. The corruption is restored either through file access or refresh.

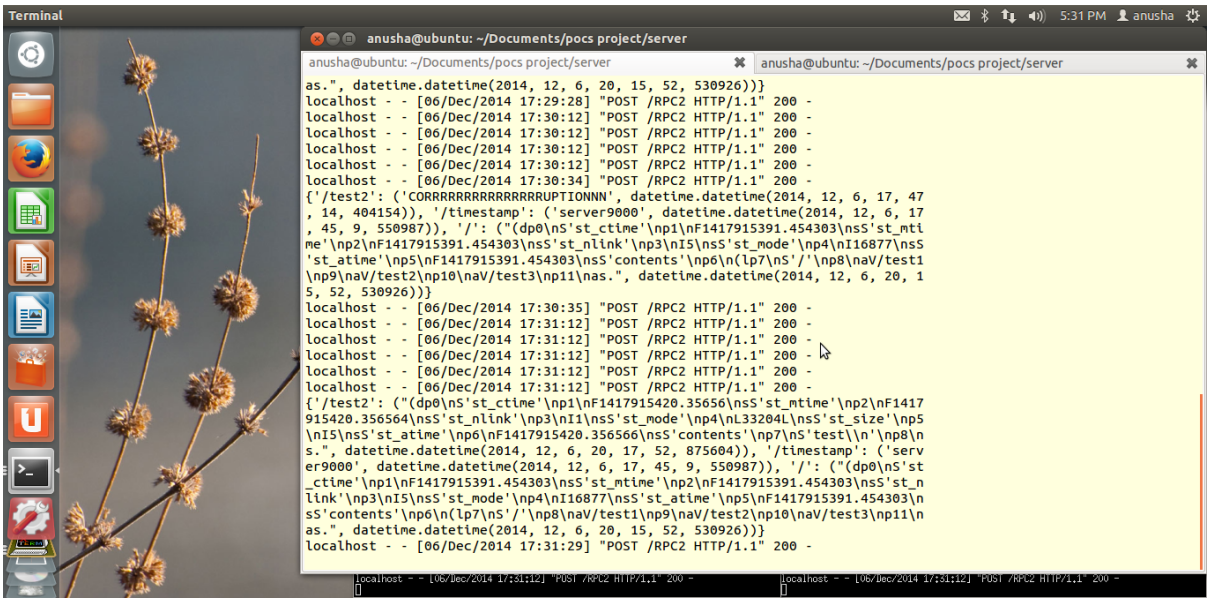

Fig 7 Corrupted data restored

The file '/test2' which was corrupted is restored through the refresh function.