REPORT

1. INTRODUCTION AND PROBLEM STATEMENT:

Giants like Google, Microsoft and so on, store their data on multiple servers (similar to RAID) using cloud storage systems. This data distributed as blocks (block storage) can be accessed by multiple clients around the world.Thus, emulating the abstraction of a block storage system is important and it finds its applications such as Solid state drives, SSDs or disk drives. Data is stored in the form of block in a block storage system. This data can be lost due to disasters or corrupted due to human errors which emphasizes the need for data recovery. One way to recover lost data is redundancy which is achieved in this project by storing data on different disks or servers with RAID 5 distribution. Another way to achieve redundancy is by storing parity which gives information if a bit has flipped or not while transferring data which helps to decide whether data needs to be discarded or repaired and retained.

RAID 4 and RAID 5 makes use of multiple disks for redundancy (data replication or multiple copies) but behave as if there is just one logical disk.

RAID 4 distributes data in a round robin fashion and makes use of a separate single parity disk whereas RAID 5 distributes data and parity in a round robin fashion (no separate disk for parity alone) which is achieved in this project.

Failures are common is every system. But these failures if not caught and corrected can lead to a chain of failures and errors. The 2 most common failures dealt with in this project are fail stop (server crash) and corrupt block.

Fault Tolerance helps to tolerate the faults mentioned above which is also achieved in this project.

One other notable difference between RAID 4 and RAID 5 is that RAID 4 requires as many as N (number of servers) ‘gets’ to proceed with repair and retrieve lost data while RAID 5 requires only two gets or reads (old data and parity) only apart from new data.

Drawback of RAID 4 is everytime data needs to be written to the data disks, even the parity disk needs to be updated which increases the congestion on the memory bus and turns out to be the bottleneck of the system. Hence the better version which is RAID 5 is implemented.

2. DESIGN AND IMPLEMENTATION:

The starter code provided, works with multiple clients and a single server. In this task, the implementation has been extended for a single client to work with multiple servers (or disks), ns denoted by N where N ranges from a minimum of 4 disks to a maximum of 8. Note in this document, the term server and disk are used interchangeably.

In this project, the client-server system that was initially designed for RAID 4 distribution of data and parity blocks is updated to RAID 5 block distributions- depicted in the figure below:

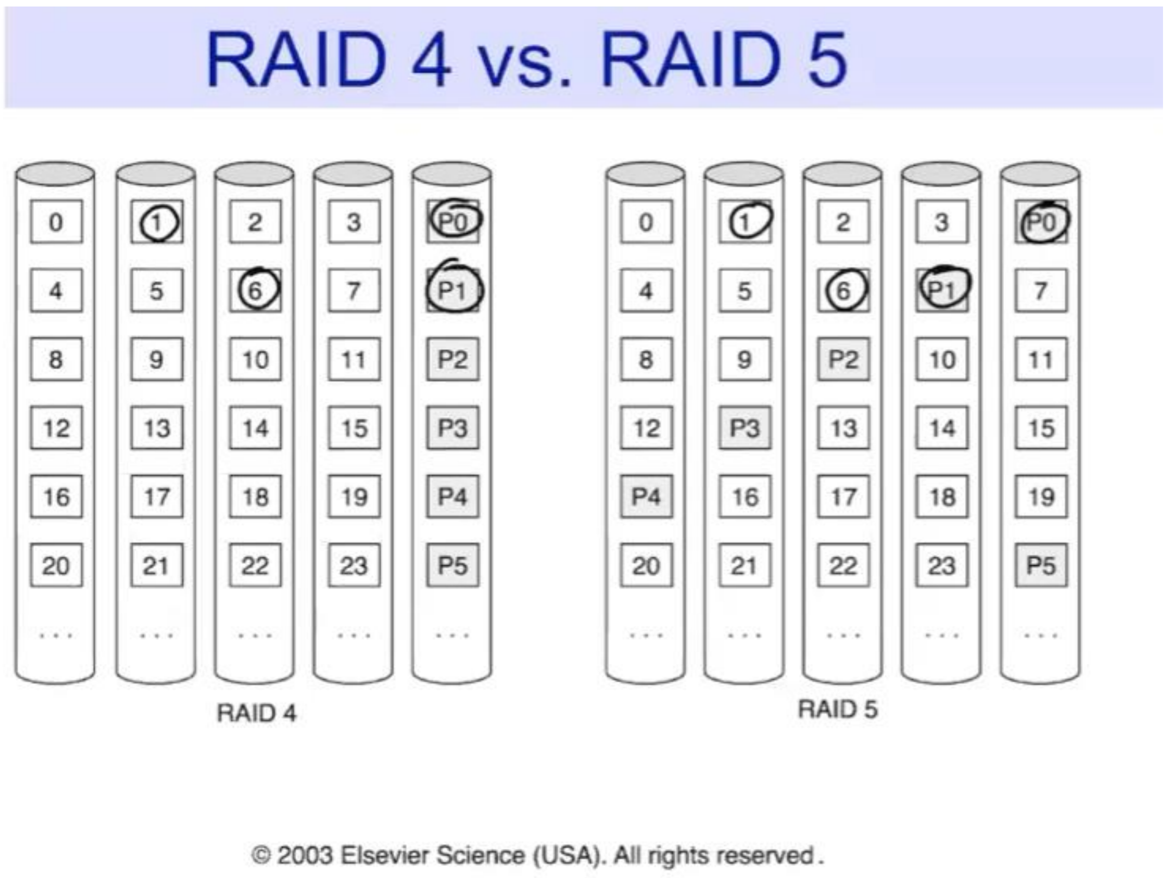


Figure RAID 4 vs RAID Virtual block distribution across Physical Blocks in the respective disk, Source:Lecture Notes

First, the command line arguments were accepted using ap.add_argument()

- In server.py: sid (server_id), cblk (optional argument corrupt block number)
- In memoryfs_shell_rpc.py: port0 to port7 (if allotted), ns

Major changes in rpc.py

1)A command-line argument "ns" specifying N (the number of servers).

```
ap.add_argument('-ns', '--number_of_servers', type=int, help='an integer value')
```

2) Command-line arguments port0, port1, ... port7 specifying the port of the (up to 8) servers.

```
 ap.add_argument('-port0', '--port0', type=int, help='an integer value')
 ap.add_argument('-port1', '--port1', type=int, help='an integer value')
 ap.add_argument('-port2', '--port2', type=int, help='an integer value')
 ap.add_argument('-port3', '--port3', type=int, help='an integer value')
 ap.add_argument('-port4', '--port4', type=int, help='an integer value')
 ap.add_argument('-port5', '--port5', type=int, help='an integer value')
 ap.add_argument('-port6', '--port6', type=int, help='an integer value')
```

```
    ap.add_argument('-port7', '--port7', type=int, help='an integer value')
```

3)

```
elif splitcmd[0] == "repair":
        if len(splitcmd) != 2:
          print ("Error: repair requires 1 argument")
        else:
          self.repair(splitcmd[1])
```

4)

```
def repair(self, sid):
    i = self.FileObject.RawBlocks.Repair(int(sid))
    if i == -1:
      print ("Error: cannot repair server\n")
      return -1
    return 0
```

Note: Since memoryfs_client.py and memoryfs_shell_rpc.py had too many changes, code not included in the report.
N servers are then created and are set to listen at the respective ports. For example: Server sid 0 listens at port0 8000, Server sid 1 listens at port1 8001 and so on.

Once the client-server connection is established, data is distributed in a round robin fashion across the multiple disks to distribute the load evenly which improves performance of the system.

Checksum is calculated on every block and the client stores to the server- the data blocks and checksum blocks. Parity is also calculated and stored. 128bit =16byte MD5 Checksums for each block is calculated using hashlib function in the XMLRPC package and then stored on a separate data structure for each server using class DiskChecksums.

RAID 5 Parity was calculated by performing an XOR on the contents of –

i)new data (to be stored on the specific disk)

ii)old data (already stored on that specific disk) ->requires one read

iii)old parity(parity calculated for old data) ->requires one read

This is an advantage of RAID 5 over RAID 4 as RAID 4 Parity calculation would have required a read on every server for the same physical block number in order for the contents to be XORed.

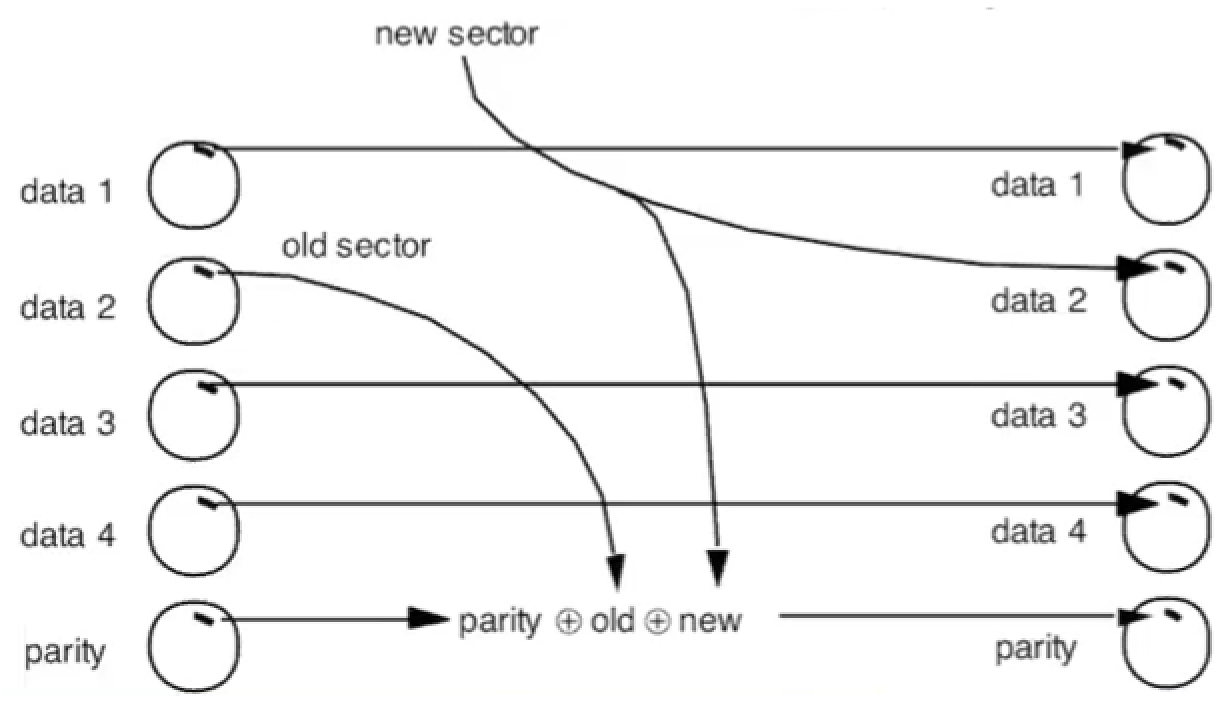


Figure Source: Lecture notes

2.1 DATA CORRUPTION:

At the server side, it is checked if the parameter for corrupt block number (**args.cblk** stored in CORRUPT_BLOCK_NUM**)** has been entered. If yes, then it checks if the present block to be Put() matched the CORRUPT_BLOCK_NUM which on evaluating successfully, the contents stored on the particular block in that server is set to corrupted contents (on every Put to the corrupt block).

By assigning the RawBlock contents with a bytearray - "Corrupted data stream" encoded in 'utf-8' format left justified with a set over zeroes appended to the end, the data is corrupted. This happens on when a cblk or corrupt block number(which corresponds to physical block number) is entered at the server side in order to corrupt or decay the contents of specified block.

For example: python3 memoryfs_server.py -bs 128 -nb 256 -port 8002 -sid 2 -cblk 3

If parameter **args.cblk** wasn't found, or the following command was entered

for example: python3 memoryfs_server.py -bs 128 -nb 256 -port 8002 -sid 2

then the contents entered by the client are marshalled and stored at the respective blocks on the respective server on every Put().

At the Server side itself, on every Get() the checksums (a kind of parity with multiple bits which stores the hashed value of the data) is verified for correctness.

2.2 VIRTUAL-TO-PHYSICAL BLOCK TRANSLATION

If one server has 256 physical blocks (0 to 255) then N=4 servers can stores 4*256=768 blocks. But the client cannot view individual blocks as 0-255 on each server, instead it can view the blocks stacked from 0 to 767. Hence these 767 virtual block numbers need to be mapped to respective physical blocks in the respective servers. This is done using the code below which is self-explanatory.

2.2.1 Code:

```
def Mapping_VB_To_PB(self, block_number): #block_number is the Virtual block no
    Physical_Block_Number = block_number//(N-1) #N=no. of servers
    Parity_Server_ID = (N-1)-Physical_Block_Number%N
    Server_ID = block_number%(N-1)
    if Server_ID>=Parity_Server_ID:
      Server_ID+=1
    return Server_ID, Physical_Block_Number, Parity_Server_ID
```

2.3 FAILURES:

Only two types of errors that tolerated by the system implemented are:

i)Fail Stop: Any one of the server crashes (emulated using ‘Ctrl+C’ on the specific server terminal)

ii)Corrupt Block: A particular block on a server is forced to decay or store corrupted contents

2.4 FAULT TOLERANCE:

i)Handling Fail Stop or server crash:

If a server crashes, the client must be able to continue to work as if no error occurred. For this a repair procedure needs to be performed on the server that crashes.

At the client manually type the command **repair sid,** where sid is an integer that denotes the server no that needs to be repaired.

The repair() function in the command line is accepted at the memoryfs_shell_rpc.py file using:

```
elif splitcmd[0] == "repair":
    if len(splitcmd) != 2:
      print ("Error: repair requires 1 argument")
    else:
      self.repair(splitcmd[1])
```

which invokes a function repair (defined in memoryfs_shell_rpc.py) that further invoked the Repair function (defined in the client file).This Repair function stores the data received from RepairBlock() invoked in a try - except ConnectionRefusedError block as part of error handling code.

ConnectionRefusedError is an exception raised by XMLRPC when the client-server connection has failed.

RepairBlock() is a function that performs a get() on the blocks stored on every server for a given physical block number (index). For example, if physical block 3 is needed on server 2 then a get is performed by the client on each of the remaining servers (0,1,3) to fetch their respective physical block 3. These contents are then XORed at the client side. This procedure is performed for all the physical blocks (from 0 to 255 in each server) and the XORed values are stored back into the server that had crashed once the server is restored.

ii)Handling Corrupt Block:

When the Client requests a block from the server , the fresh checksum of the block requested is computed using **hashlib.md5(data).digest()** the checksum of the same block is fetched and compared.If the two match, then the block of data stored on the server is returned. If not, it means the block was corrupted. So, the RepairBlock() procedure is implemented and the new contents are stored in the block that was corrupted.

3. EVALUATION AND REPRODUCIBILITY

Start 5 server terminals, enter the directory where the 3 files are stored(current working directory on the system).Run the command **python3 memoryfs_server.py -bs 128 -nb 256 -port 8000 -sid 0** .Using respective sid and port no from the table below, in the command above, 4 server terminals are started.

The port no above signifies the port at which the server is listening or waiting for the client to connect.

sid 0	port 8000
sid 1	port 8001
sid 2	port 8002
sid 3	port 8003

```
python3 memoryfs_server.py -bs 128 -nb 256 -port 8000 -sid 0

python3 memoryfs_server.py -bs 128 -nb 256 -port 8001 -sid 1

python3 memoryfs_server.py -bs 128 -nb 256 -port 8002 -sid 2

python3 memoryfs_server.py -bs 128 -nb 256 -port 8003 -sid 3
```

Start the client on another terminal using the command **python3 memoryfs_shell_rpc.py -ns 4 -port0 8000 -port1 8001 -port2 8002 -port3 8003 -nb 768 -bs 128 -is 16 -ni 16 -cid 0** (use client_ID or cid and a port number at which the client is open to connect to the server)

Evaluation of load distribution:

Say client needs to perform put() or store virtual block no or block_number=7 to one of the 4 servers.

```
Physical_Block_Number = block_number//(N-1)   #7//(4-1)  ->  7//3  ->2
# Physical_Block_Number =2
Parity_Server_ID = (N-1)-Physical_Block_Number%N   #(4-1)-2%4  ->3-2  ->1
# Parity_Server_ID =1
Server_ID = block_number%(N-1)  #7%(4-1)  ->7%3  ->1
# Server_ID =1
if Server_ID>=Parity_Server_ID:   # if 1>0    evaluates to true
  Server_ID+=1                     # Server_ID =2
```

These values are returned from the Mapping_VB_To_PB function and are then used in respective put and get functions in the client file to index the physical block contents distributed across N servers.

List of commands to execute at the client terminal to test the working of the implementation:

Python memoryfs_shell_rpc.py -ns 4 -port0 8000 -port1 8001 -port2 8002 -port3 8003 -nb 768-bs 128 -is 16 -ni 16 -cid 0

ls

mkdir dir1

mkdir dir2

ls

cd dir1

create file1

create file2

append file1 abcdefghi

ls

cat file1

ls

ls

After Crashing server 3(sid 2 as index starts from 0) using Ctrl+C (at server side) issue the following commands at the client side:

create file3

at terminal for sid 2 restart server using Python memoryfs_shell_rpc.py -ns 4 -port0 8000 -port1 8001 -port2 8002 -port3 8003 -nb 768-bs 128 -is 16 -ni 16 -cid 0

at the client side issue: **repair 2**

create file4

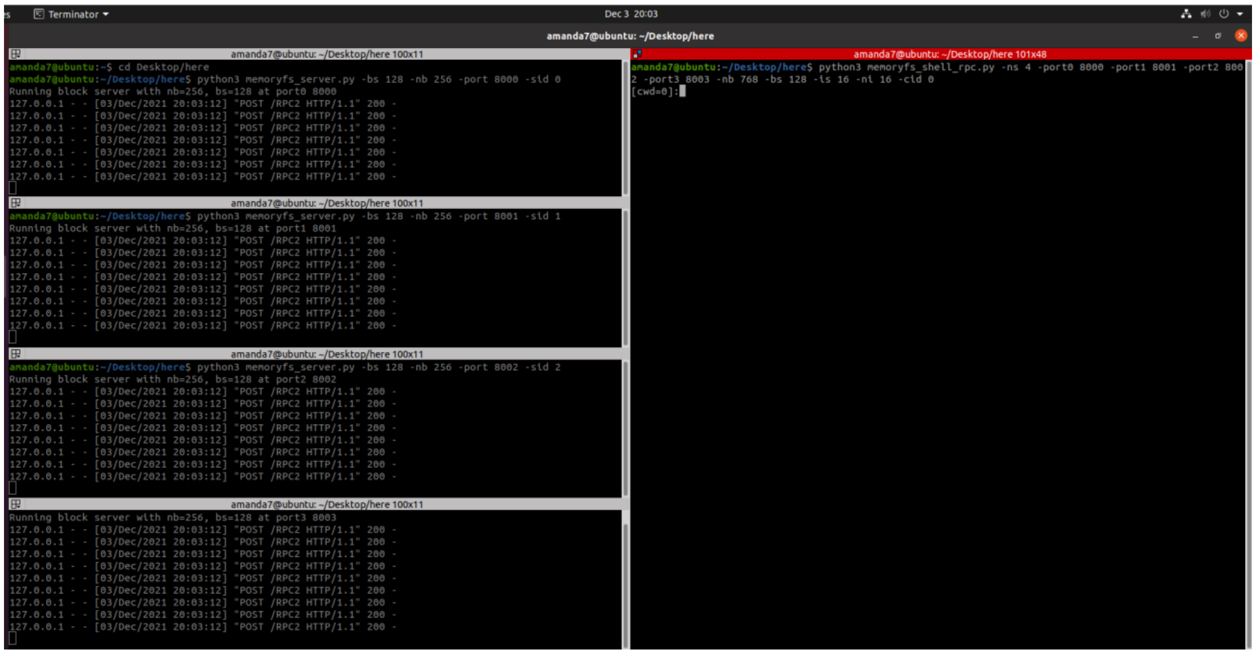
ls

4. CONCLUSION

The block storage system runs with RAID 5 distribution of blocks and can tolerate two types of errors -fail stop and corrupt block. The repair procedure successfully repairs the corrupted block or restores the crashed server's contents when the server is restored and repair in invoked in the client terminal.

5. RESULTS

RESULTS TO EMULATE FAIL STOP TOLERANCE



```
amanda7@ubuntu:~$ cd Desktop/here
amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_server.py -bs 128 -nb 256 -port 8000 -sid 0
Running block server with nb=256, bs=128 at port0 8000
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_server.py -bs 128 -nb 256 -port 8001 -sid 1
Running block server with nb=256, bs=128 at port1 8001
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
```

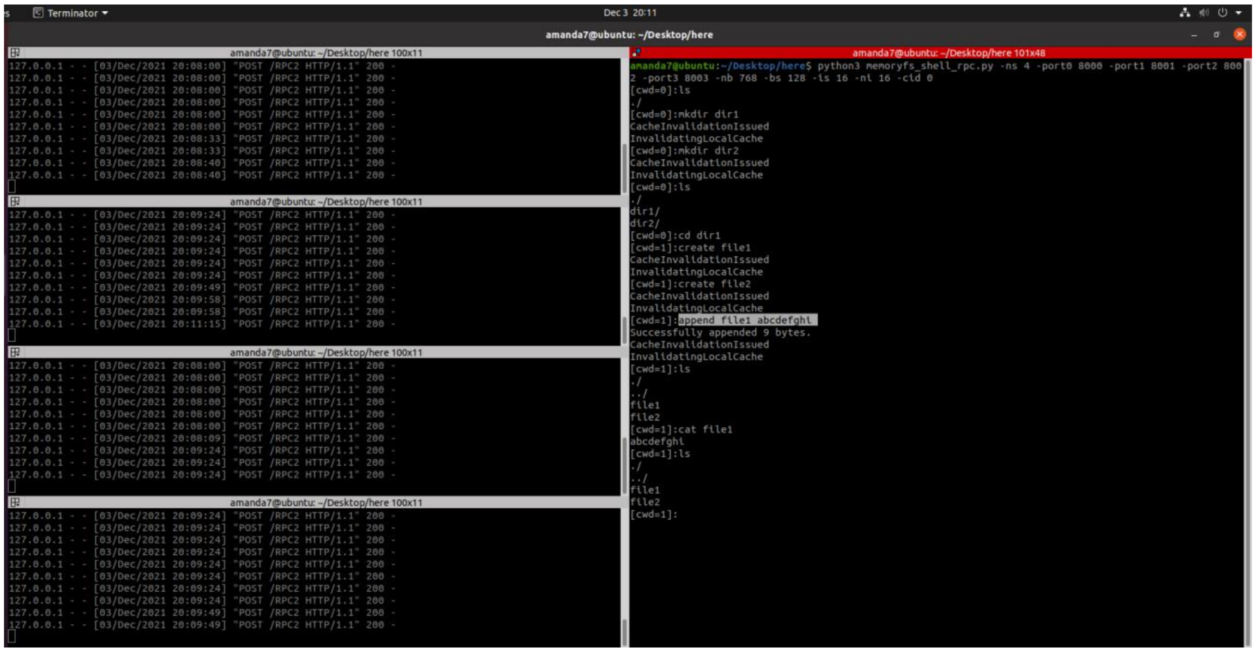
```
amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_server.py -bs 128 -nb 256 -port 8002 -sid 2
Running block server with nb=256, bs=128 at port2 8002
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
```

```
Running block server with nb=256, bs=128 at port3 8003
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:03:12] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_shell_rpc.py -ns 4 -port0 8000 -port1 8001 -port2 800
2 -port3 8003 -nb 768 -bs 128 -is 16 -ni 16 -cid 0
[cwd=0]:
```

Terminator
Dec 3 20:03
amanda7@ubuntu: ~/Desktop/here
amanda7@ubuntu: ~/Desktop/here 100x11
amanda7@ubuntu: ~/Desktop/here 100x11
amanda7@ubuntu: ~/Desktop/here 100x11
amanda7@ubuntu: ~/Desktop/here 100x11
amanda7@ubuntu: ~/Desktop/here 101x48

Figure: Initial commands to be run



```
amanda7@ubuntu: ~/Desktop/here 100x11
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:33] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:33] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:40] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:40] "POST /RPC2 HTTP/1.1" 200 -
```

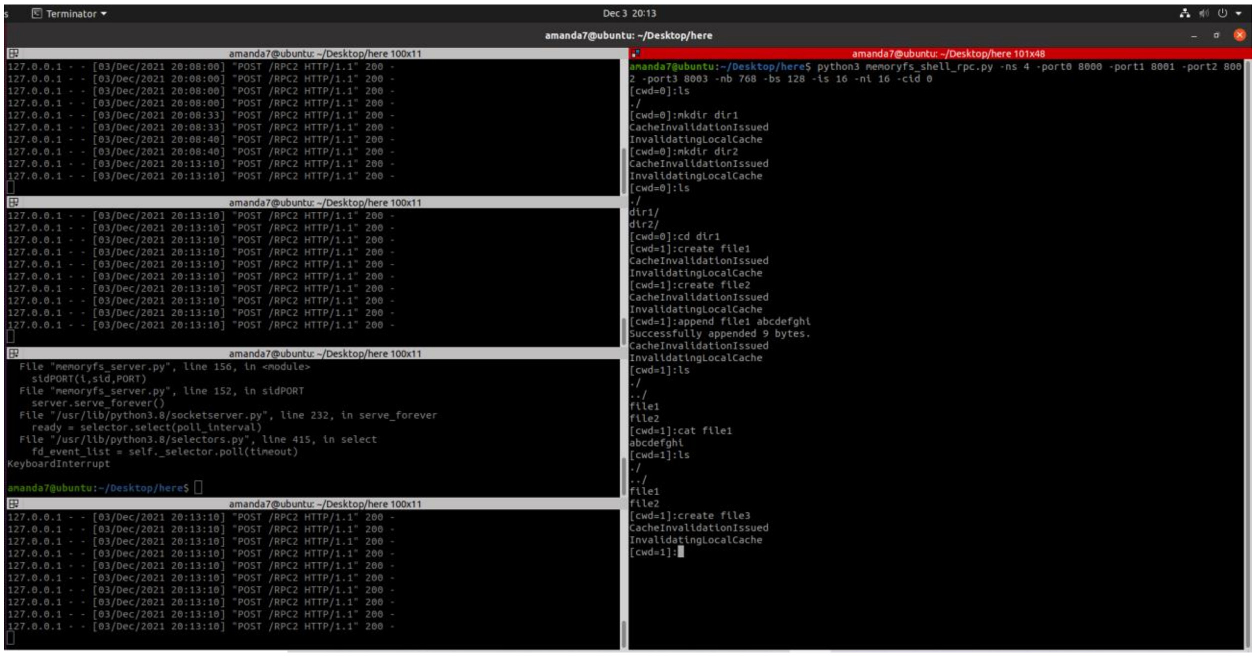
```
amanda7@ubuntu: ~/Desktop/here 100x11
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:49] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:58] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:58] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:11:15] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 100x11
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 100x11
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:24] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:49] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:09:49] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 101x48
amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_shell_rpc.py -ns 4 -port0 8000 -port1 8001 -port2 800
2 -port3 8003 -nb 768 -bs 128 -is 16 -ni 16 -cid 0
[cwd=0]:ls
./
[cwd=0]:mkdir dir1
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=0]:mkdir dir2
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=0]:ls
./
dir1/
dir2/
[cwd=0]:cd dir1
[cwd=1]:create file1
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:create file2
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:append file1 abcdefghi
Successfully appended 9 bytes.
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:ls
./
../
file1
file2
[cwd=1]:cat file1
abcdefghi
[cwd=1]:ls
./
../
file1
file2
[cwd=1]:
```

Figure: Commands issued before emulating a crash



```
amanda7@ubuntu: ~/Desktop/here 100x11
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:00] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:33] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:33] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:40] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:08:40] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
```

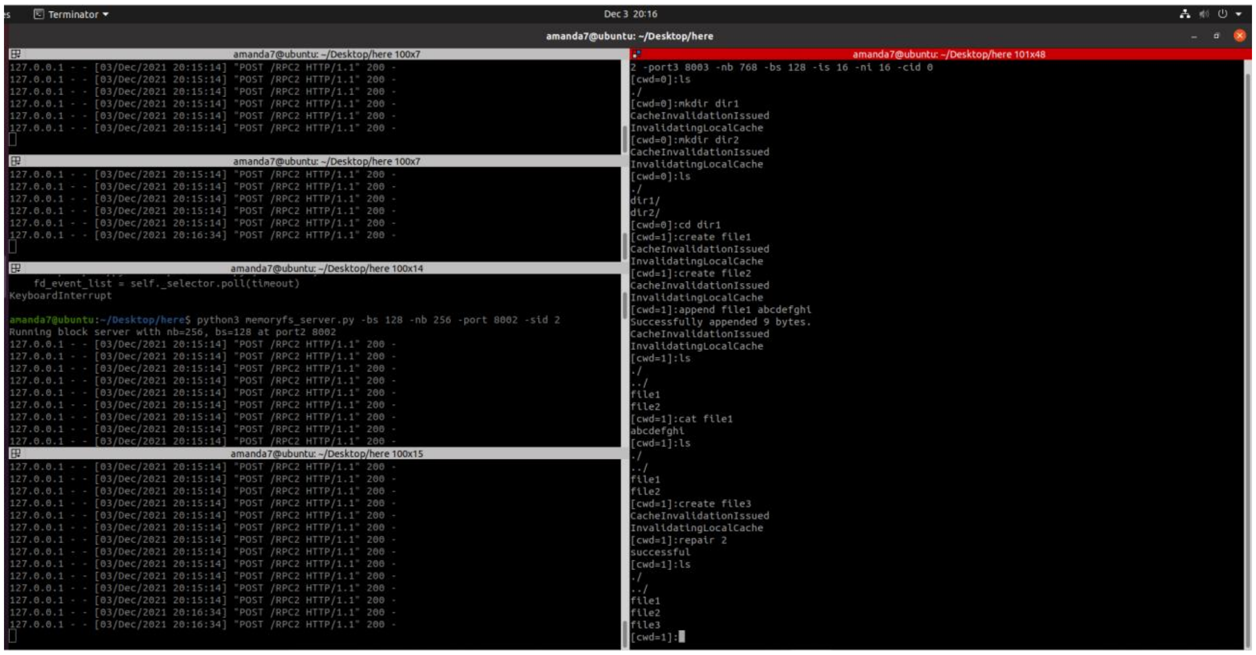
```
amanda7@ubuntu: ~/Desktop/here 100x11
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 100x11
  File "memoryfs_server.py", line 156, in <module>
    sidPORT(i,sid,PORT)
  File "memoryfs_server.py", line 152, in sidPORT
    server.serve_forever()
  File "/usr/lib/python3.8/socketserver.py", line 232, in serve_forever
    ready = selector.select(poll_interval)
  File "/usr/lib/python3.8/selectors.py", line 415, in select
    fd_event_list = self._selector.poll(timeout)
KeyboardInterrupt

amanda7@ubuntu:~/Desktop/here$
```

```
amanda7@ubuntu: ~/Desktop/here 100x11
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:13:10] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 101x48
amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_shell_rpc.py -ns 4 -port0 8000 -port1 8001 -port2 800
2 -port3 8003 -nb 768 -bs 128 -is 16 -ni 16 -cid 0
[cwd=0]:ls
./
[cwd=0]:mkdir dir1
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=0]:mkdir dir2
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=0]:ls
./
dir1/
dir2/
[cwd=0]:cd dir1
[cwd=1]:create file1
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:create file2
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:append file1 abcdefghi
Successfully appended 9 bytes.
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:ls
./
../
file1
file2
[cwd=1]:cat file1
abcdefghi
[cwd=1]:ls
./
../
file1
file2
[cwd=1]:create file3
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:
```

Fig: After server sid2 crashed and then create file3 issued on client, client still runs as if nothing happened



```
amanda7@ubuntu: ~/Desktop/here 100x7
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 100x7
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:16:34] "POST /RPC2 HTTP/1.1" 200 -
```

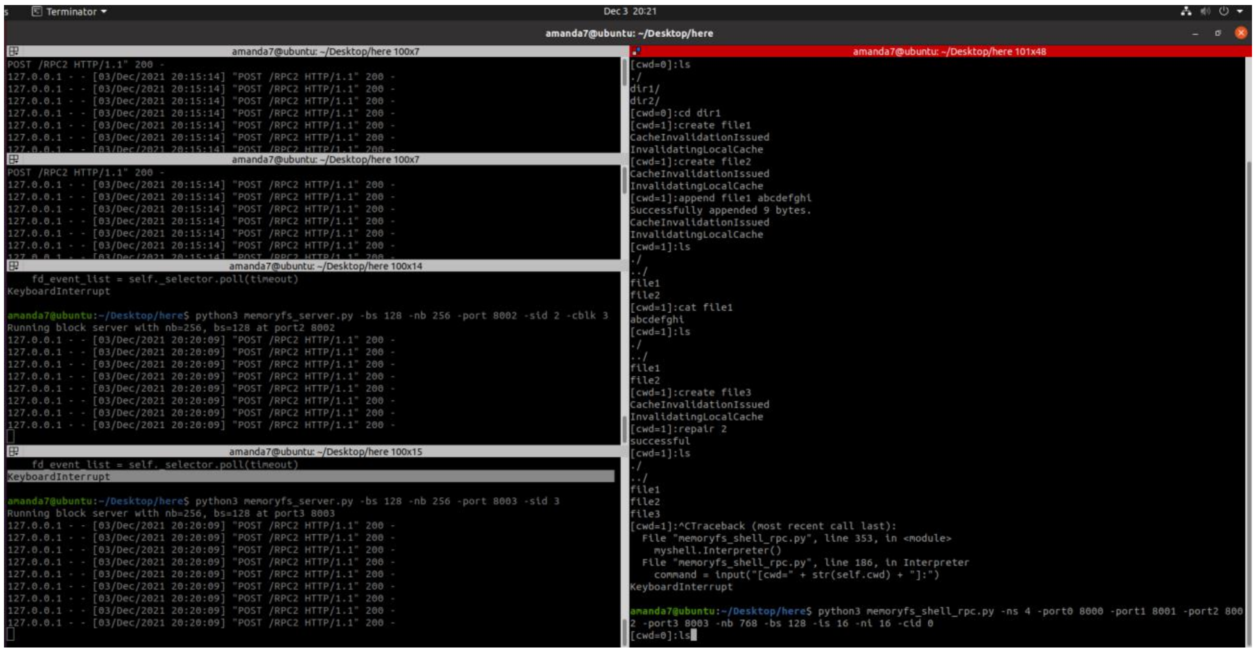
```
amanda7@ubuntu: ~/Desktop/here 100x14
    fd_event_list = self._selector.poll(timeout)
KeyboardInterrupt

amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_server.py -bs 128 -nb 256 -port 8002 -sid 2
Running block server with nb=256, bs=128 at port2 8002
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 100x15
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:16:34] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:16:34] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 101x48
2 -port3 8003 -nb 768 -bs 128 -is 16 -ni 16 -cid 0
[cwd=0]:ls
./
[cwd=0]:mkdir dir1
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=0]:mkdir dir2
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=0]:ls
./
dir1/
dir2/
[cwd=0]:cd dir1
[cwd=1]:create file1
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:create file2
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:append file1 abcdefghi
Successfully appended 9 bytes.
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:ls
./
../
file1
file2
[cwd=1]:cat file1
abcdefghi
[cwd=1]:ls
./
../
file1
file2
[cwd=1]:create file3
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:repair 2
successful
[cwd=1]:ls
./
../
file1
file2
file3
[cwd=1]:
```

Figure : Sid 2 was restarted and Repair successfully invoked on sid 2 from client terminal.

RESULTS FOR CORRUPT BLOCK TESTCASE



```
amanda7@ubuntu: ~/Desktop/here 100x7
POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
```

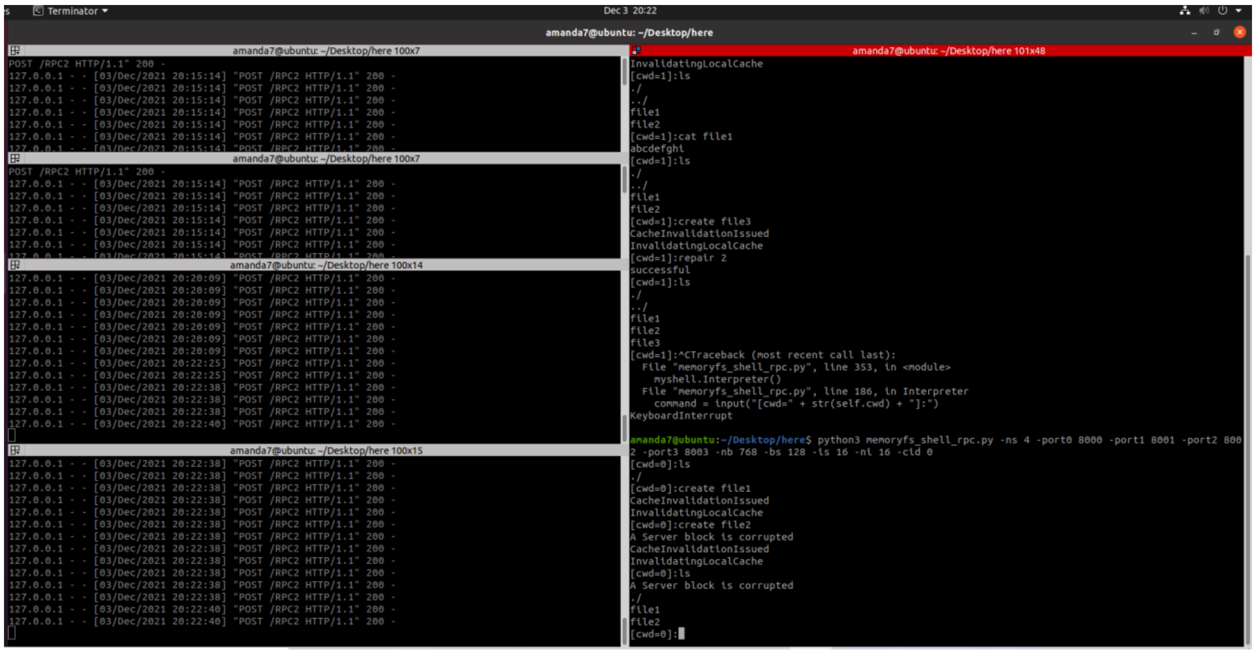
```
amanda7@ubuntu: ~/Desktop/here 100x7
POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 100x14
    fd_event_list = self._selector.poll(timeout)
KeyboardInterrupt

amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_server.py -bs 128 -nb 256 -port 8002 -sid 2 -cblk 3
Running block server with nb=256, bs=128 at port2 8002
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 100x15
    fd_event_list = self._selector.poll(timeout)
KeyboardInterrupt

amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_server.py -bs 128 -nb 256 -port 8003 -sid 3
Running block server with nb=256, bs=128 at port3 8003
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 101x48
[cwd=0]:ls
./
dir1/
dir2/
[cwd=0]:cd dir1
[cwd=1]:create file1
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:create file2
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:append file1 abcdefghi
Successfully appended 9 bytes.
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:ls
./
../
file1
file2
[cwd=1]:cat file1
abcdefghi
[cwd=1]:ls
./
../
file1
file2
[cwd=1]:create file3
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:repair 2
successful
[cwd=1]:ls
./
../
file1
file2
file3
[cwd=1]:^CTraceback (most recent call last):
  File "memoryfs_shell_rpc.py", line 353, in <module>
    myshell.Interpreter()
  File "memoryfs_shell_rpc.py", line 186, in Interpreter
    command = input("[cwd=" + str(self.cwd) + "]:")
KeyboardInterrupt

amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_shell_rpc.py -ns 4 -port0 8000 -port1 8001 -port2 800
2 -port3 8003 -nb 768 -bs 128 -is 16 -ni 16 -cid 0
[cwd=0]:ls
```

Figure: cblk 3 entered at sid 2



```
amanda7@ubuntu: ~/Desktop/here 100x7
POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 100x7
POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:15:14] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 100x14
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:20:09] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:25] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:25] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:40] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 100x15
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:38] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:40] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [03/Dec/2021 20:22:40] "POST /RPC2 HTTP/1.1" 200 -
```

```
amanda7@ubuntu: ~/Desktop/here 101x48
InvalidatingLocalCache
[cwd=1]:ls
./
../
file1
file2
[cwd=1]:cat file1
abcdefghi
[cwd=1]:ls
./
../
file1
file2
[cwd=1]:create file3
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=1]:repair 2
successful
[cwd=1]:ls
./
../
file1
file2
file3
[cwd=1]:^CTraceback (most recent call last):
  File "memoryfs_shell_rpc.py", line 353, in <module>
    myshell.Interpreter()
  File "memoryfs_shell_rpc.py", line 186, in Interpreter
    command = input("[cwd=" + str(self.cwd) + "]:")
KeyboardInterrupt

amanda7@ubuntu:~/Desktop/here$ python3 memoryfs_shell_rpc.py -ns 4 -port0 8000 -port1 8001 -port2 800
2 -port3 8003 -nb 768 -bs 128 -is 16 -ni 16 -cid 0
[cwd=0]:ls
./
[cwd=0]:create file1
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=0]:create file2
A Server block is corrupted
CacheInvalidationIssued
InvalidatingLocalCache
[cwd=0]:ls
A Server block is corrupted
./
file1
file2
[cwd=0]:
```

Figure : After running a few command, the client is notified of a corrupt block in the server, but the client still continues operating as usual.