Introduction:

This file serves as a guide on how to run/replicate the tests which have been performed on a VM which mounts a FUSE file system showing basic FS calls, directory operations, optional operations along with test cases and benchmarking results using fio.

This file acts as a report (answering the questions asked), contains the examples as well as the different test cases and finally the comparison results.

Note: Based on the feedback for last assignment, this time I have added the bash script containing all the commands which I ran in VM. Please do not run the whole script altogether but only parts (which are required) which will be covered below in the report (follow the sequence). The FUSE agent in my case is run on the GCP VM.

Submission Requirements:

Bash script file (just contains the bash code) –
Test cases – attached in the file below.
File benchmarking results – attached in the file below.
Main code – testfuse5.py
GCP bucket ssh key - prashul-kumar-fall2023-23ecf366ca39.json
Log file – log_file.txt containing the detailed log of all the operations

File Structure –

testfuse5.py – Should place this in praskumainstance2
my_mount_point – create a directory in the GCP VM which should be mounted and serve as the interface for GCS bucket. Give full access to this bucket
prashul-kumar-fall2023-23ecf366ca39.json – place this in praskumainstance2. This file is mandatory and serves as the connect (has private key) between mounted directory and the bucket praskumabucket

```
prashulkumar@praskumainstance2:~$ ls
my_mount_point  prashul-kumar-fall2023-23ecf366ca39.json  testfuse5.py
prashulkumar@praskumainstance2:~$
```

Architecture –

Components –

The GCS FUSE File System contains the following:

FUSE - The FUSE library provides a bridge between user-space and kernel-space. It intercepts file system calls and redirects them to the GCS File System.

GCS_Bucket_FS - The GCS_Bucket_FS class implements the file system operations using the FUSE library. It interacts with Google Cloud Storage to manage files and directories.
This class also maintains in-memory data structures to track file metadata and content.

Google Cloud Storage - The GCS component stores the actual data. The GCS File System interacts with GCS to upload, download, and manage files and directories.

File system operations supported are -
chmod: Change file permissions.
chown: Change file ownership.
create: Create a new file.
open: Open an existing file.
close: Close a file.
getattr: Retrieve file or directory attributes.
getxattr: Retrieve extended attributes (placeholder implementation).
read: Read file content.
truncate: Truncate a file to a specified length.
write: Write data to a file.
mkdir: Create a new directory.
rmdir: Remove a directory.
readdir: List directory contents.
opendir: Open a directory.
flush: Flush cached data for a file.
fsync: Synchronize file data to storage.
rename: Rename a file or directory.
link: Create a hard link.
unlink: Delete a file (unlink).

Design Discussion –

File and Directory Operations

Creation and Deletion – To support for creation and deletion of files and directories - When these operations are performed, they are accurately represented within my ‘self.files’ data

structure. Deletion of files and directories is also efficiently handled, with associated GCS objects being removed.

Reading and Writing - The read operation allows users to retrieve content from GCS objects, while the write operation permits them to update these objects. This design enables users to read from and write to GCS objects as if they were local files.

Error Handling - In my design, I raised error codes defined in the 'errno' module (e.g., ENOENT, EROFS, ENOTEMPTY, EEXIST) when errors occur. These error codes convey specific information about the nature of the error, aiding users in identifying and resolving issues effectively which are put in place based on the logs messages generated.

Directory Listing - I wanted to ensure that users could easily list the contents of directories, including subdirectories and files. The 'readdir' operation enables the users to list directory contents, files and subdirectories.

Performance Optimization - My system incorporates a caching mechanism using a dictionary data structure. The **self.data** dictionary stores file content, enhancing read performance by serving data from the cache when available. This design effectively reduces the need for frequent network requests and enhances overall performance.

Logging and Debugging - In my system, I have integrated the Python 'logging' library to log operations, errors, and informational messages. This design allows users to adjust the logging level for debugging purposes.

Handling Read-Only Files - In my system, I track read-only files and raise an error (EROFS) when users attempt to modify them. This design choice ensures that read-only and read-write objects are handled correctly, maintaining data integrity.

Open and Close Operations - My system keeps track of file descriptors (fd) to manage open files. Users can open and close files as needed, and the 'close' operation updates timestamps to reflect changes.

Existing File and Directory Population- I also provided the functionality of populating existing files and directories from the GCS bucket when initialized. This design choice ensures that the local file system reflects the structure and contents of the GCS bucket, making it easy for users to work with their data.

Placeholder functions – There are quite a few places which I have implemented as just a placeholder, for example – flush, fsync, etc. The reason being, these operations are already being called by the FUSE system in the backend as can be seen through the logs (which will be discussed in the below examples). Since these functions are already being called in the backend, hence I just created as placeholder(return 0) and put a message to print showing the this

function has been called and the same can be verified through the logs (the callback print messages are displayed there).

Initial Setup steps/design –

First of all, the VM's created during previous assignments were only reused. Hence, just needed to start the VM's first using –

gcloud compute instances start praskumainstance2

prashulkumar@Prashuls-MacBook-Air google-cloud-sdk % gcloud compute instances list

```
prashulkumar@Prashuls-MacBook-Air google-cloud-sdk % gcloud compute instances list
NAME: praskumainstance1
ZONE: us-east1-c
MACHINE_TYPE: n1-standard-1
PREEMPTIBLE:
INTERNAL_IP: 10.142.0.2
EXTERNAL_IP: 34.23.176.23
STATUS: RUNNING

NAME: praskumainstance2
ZONE: us-east1-c
MACHINE_TYPE: t2d-standard-2
PREEMPTIBLE:
INTERNAL_IP: 10.142.0.3
EXTERNAL_IP: 34.139.87.207
STATUS: RUNNING
prashulkumar@Prashuls-MacBook-Air google-cloud-sdk %
```

Now the files have to be uploaded to the VM instance using –

gcloud compute scp /Users/prashulkumar/Documents/SEM-3/ECC/Assignment-3/testfuse5.py praskumainstance2:pythonproject/

gcloud compute scp /Users/prashulkumar/Documents/SEM-3/ECC/Assignment-3/prashul-kumar-fall2023-23ecf366ca39.json praskumainstance2:pythonproject/

Once the files have been uploaded, then we have to create a folder my_mount_point inside the VM and give it the appropriate access and also install fuse in VM –

sudo apt-get install fuse
mkdir my_mount_point
chmod 777 my_mount_point/

Now our initial setup is ready –

```
prashulkumar@praskumainstance2:~$ ls
my_mount_point  prashul-kumar-fall2023-23ecf366ca39.json  testfuse5.py
prashulkumar@praskumainstance2:~$
```

Now we should have another terminal open in which we will mount the directory created and in this terminal we will run our fuse file system code.

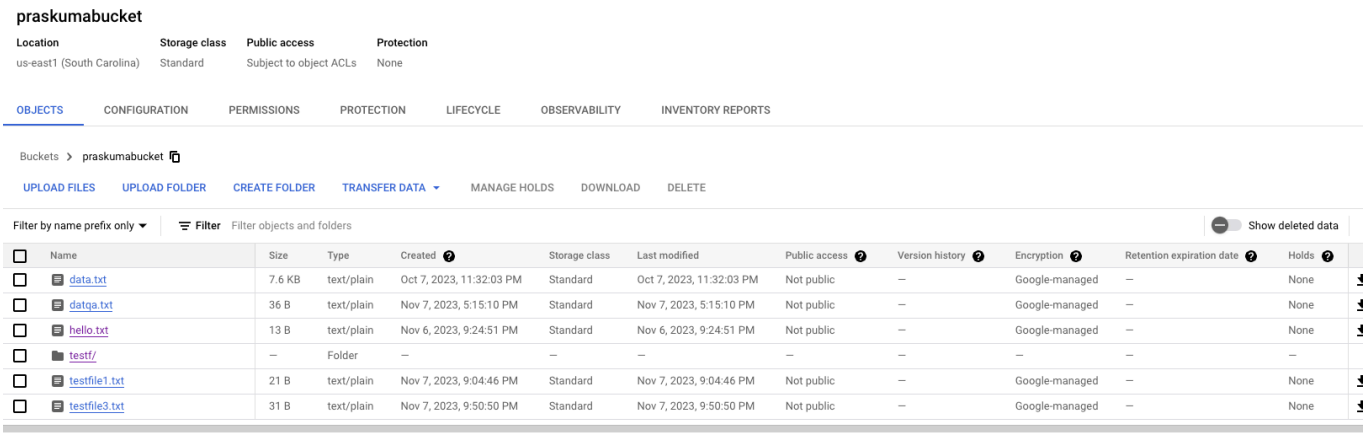
To run the fuse file system –

python3 testfuse5.py /home/prashulkumar/my_mount_point

This code starts the FUSE file system and mounts the created directory

```
prashulkumar@praskumainstance2:~$ python3 testfuse5.py /home/prashulkumar/my_mount_point
Connected to the bucket successfully
DEBUG:urllib3.util.retry:Converted retries value: 3 -> Retry(total=3, connect=None, read=None, redirect=None, status=None)
DEBUG:google.auth.transport.requests:Making request: POST https://oauth2.googleapis.com/token
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): oauth2.googleapis.com:443
DEBUG:urllib3.connectionpool:https://oauth2.googleapis.com:443 "POST /token HTTP/1.1" 200 None
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): storage.googleapis.com:443
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /storage/v1/b/praskumabucket/o?projection=noAcl&prettyPrint=false HTTP/1.1" 200 4781
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/data.txt?generation=1696735923726523&alt=media HTTP/1.1" 200 7776
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/datqa.txt?generation=1699395310241809&alt=media HTTP/1.1" 200 36
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/hello.txt?generation=1699323891832368&alt=media HTTP/1.1" 200 13
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testf%2F?generation=1699382196286225&alt=media HTTP/1.1" 200 0
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testf%2Fsample.txt?generation=1699382254877407&alt=media HTTP/1.1" 200 0
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testfile1.txt?generation=1699409086150234&alt=media HTTP/1.1" 200 21
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testfile3.txt?generation=1699411850807204&alt=media HTTP/1.1" 200 31
DEBUG:fuse.log-mixin:-> init / ()
DEBUG:fuse.log-mixin:<- init None
```

My gcs bucket already has some exisiting files (listed above)



Now from second terminal, ssh into praskumainstance2 and go to the mounted directory –
prashulkumar@praskumainstance2:~$ cd my_mount_point/
prashulkumar@praskumainstance2:**~/my_mount_point**$

Test cases and Examples –

Test Case1: This test case shows a simple loop combining various operations –

Mount->create->open->write->read->close loop

Now remember my folder is already in mounted state. (i.e. my_mount_point) –

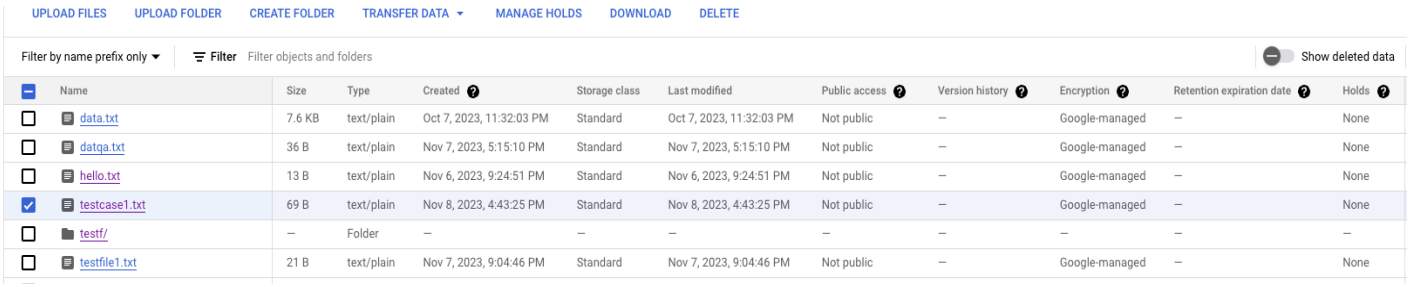
Sequence of commands –
prashulkumar@praskumainstance2:**~/my_mount_point**$ touch testcase1.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ cat > testcase1.txt
this is some content
this is another content
^C
prashulkumar@praskumainstance2:**~/my_mount_point**$ cat testcase1.txt
this is some content
this is another content
prashulkumar@praskumainstance2:**~/my_mount_point**$ echo "append to the same file" >> testcase1.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ cat testcase1.txt
this is some content
this is another content
append to the same file
prashulkumar@praskumainstance2:~/my_mount_point$ cd ..
prashulkumar@praskumainstance2:~$ umount my_mount_point

Logs generated (here I am only attaching the significant logs to show callbacks or what is actually happening in the backend) Please refer the log_file.txt to view the complete log –

DEBUG:fuse.log-mixin:-> getattr /testcase1.txt (None,)
DEBUG:fuse.log-mixin:-> create /testcase1.txt (33188,)
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "POST /upload/storage/v1/b/praskumabucket/o?uploadType=multipart HTTP/1.1" 200 748
DEBUG:fuse.log-mixin:<- create 1
DEBUG:fuse.log-mixin:-> getattr /testcase1.txt (1,)
DEBUG:fuse.log-mixin:<- getattr {'st_mode': 33188, 'st_ctime': 1699479626.1594315, 'st_mtime': 1699479626.1594317, 'st_atime': 1699479626.1594317, 'st_nlink': 1, 'st_size': 0, 'is_read_only': False}
DEBUG:fuse.log-mixin:-> flush /testcase1.txt (1,)
flush called for path: /testcase1.txt
DEBUG:fuse.log-mixin:-> open /testcase1.txt (32769,)
DEBUG:fuse.log-mixin:<- open 2
DEBUG:fuse.log-mixin:-> truncate /testcase1.txt (0,)

DEBUG:fuse.log-mixin:<- truncate None
DEBUG:fuse.log-mixin:-> write /testcase1.txt (b'this is some content\n', 0, 2)
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testcase1.txt?alt=media HTTP/1.1" 200 0
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "POST /upload/storage/v1/b/praskumabucket/o?uploadType=multipart HTTP/1.1" 200 749
DEBUG:fuse.log-mixin:-> write /testcase1.txt (b'this is another content\n', 21, 2)
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testcase1.txt?alt=media HTTP/1.1" 200 21
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "POST /upload/storage/v1/b/praskumabucket/o?uploadType=multipart HTTP/1.1" 200 749
DEBUG:fuse.log-mixin:<- write 24
DEBUG:fuse.log-mixin:-> release /testcase1.txt (2,)
DEBUG:fuse.log-mixin:<- release 0
DEBUG:fuse.log-mixin:-> opendir / ()
opendir accessed for path: /
DEBUG:fuse.log-mixin:<- opendir 0
.....
DEBUG:fuse.log-mixin:<- write 24
DEBUG:fuse.log-mixin:-> flush /testcase1.txt (4,)
flush called for path: /testcase1.txt
DEBUG:fuse.log-mixin:-> open /testcase1.txt (32768,)
DEBUG:fuse.log-mixin:<- open 5
DEBUG:fuse.log-mixin:-> read /testcase1.txt (4096, 0, 5)
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testcase1.txt?alt=media HTTP/1.1" 200 69
DEBUG:fuse.log-mixin:<- read b'this is some content\nthis is another content\nappend to the same file\n'
DEBUG:fuse.log-mixin:-> destroy / ()
DEBUG:fuse.log-mixin:<- destroy None

The file is visible in GCS bucket –



Through this test case we can see the correctness of the following functions –
Mount, create, open, write, read, close, umount, opendir, flush (last 2 through callbacks visible in logs)

Test Case2: To verify if my populate_existing_files function is working fine or not, let's try to just connect to the GCS bucket and try reading existing files in the bucket

Run –

Terminal1 -
prashulkumar@praskumainstance2:~$ python3 testfuse5.py /home/prashulkumar/my_mount_point

```
prashulkumar@praskumainstance2:~$ python3 testfuse5.py /home/prashulkumar/my_mount_point
Connected to the bucket successfully
DEBUG:urllib3.util.retry:Converted retries value: 3 -> Retry(total=3, connect=None, read=None, redirect=None, status=None)
DEBUG:google.auth.transport.requests:Making request: POST https://oauth2.googleapis.com/token
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): oauth2.googleapis.com:443
DEBUG:urllib3.connectionpool:https://oauth2.googleapis.com:443 "POST /token HTTP/1.1" 200 None
DEBUG:urllib3.connectionpool:Starting new HTTPS connection (1): storage.googleapis.com:443
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /storage/v1/b/praskumabucket/o?projection=noAcl&prettyPrint=false HTTP/1.1" 200 5461
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/data.txt?generation=1696735923726523&alt=media HTTP/1.1" 200 7776
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/datqa.txt?generation=1699395310241809&alt=media HTTP/1.1" 200 36
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/hello.txt?generation=1699323891832368&alt=media HTTP/1.1" 200 13
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testcase1.txt?generation=1699479805274469&alt=media HTTP/1.1" 200 69
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testf%2F?generation=1699382196286225&alt=media HTTP/1.1" 200 0
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testf%2Fsample.txt?generation=1699382254877407&alt=media HTTP/1.1" 200 0
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testfile1.txt?generation=1699409086150234&alt=media HTTP/1.1" 200 21
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/testfile3.txt?generation=1699411850807204&alt=media HTTP/1.1" 200 31
DEBUG:fuse.log-mixin:-> init / ()
DEBUG:fuse.log-mixin:<- init None
```

Terminal2-
prashulkumar@praskumainstance2:~$ cd my_mount_point/
prashulkumar@praskumainstance2:**~/my_mount_point**$ cat data.txt

```
prashulkumar@praskumainstance2:~/my_mount_point$ cat data.txt
c1-10 89
c1-1 80
c1-12 32
c1-20 67
c1-16 90
c1-14 92
c1-4 56
c1-10 28
c1-4 34
c1-4 33
c1-6 15
c1-20 31
c1-11 18
c1-14 98
c1-19 79
c1-7 37
c1-8 94
c1-8 18
c1-16 44
c1-3 30
c1-7 92
c1-5 93
c1-6 90
c1-9 83
c1-19 90
c1-1 12
c1-6 93
c1-15 61
c1-14 67
c1-11 23
c1-18 97
c1-3 13
```

Test Case3: To verify the functionality of directory-related operations, including creating directories, opening directories, listing directory contents, and removing directories.

Terminal1:
prashulkumar@praskumainstance2:~$ python3 testfuse5.py /home/prashulkumar/my_mount_point

Terminal2:
prashulkumar@praskumainstance2:~$ cd my_mount_point/
prashulkumar@praskumainstance2:**~/my_mount_point**$ mkdir testfolder/
prashulkumar@praskumainstance2:**~/my_mount_point**$ cd testfolder/
prashulkumar@praskumainstance2:**~/my_mount_point/testfolder**$ mkdir subtestfolder/
prashulkumar@praskumainstance2:**~/my_mount_point/testfolder**$ touch file1.txt
prashulkumar@praskumainstance2:**~/my_mount_point/testfolder**$ touch file2.txt
prashulkumar@praskumainstance2:**~/my_mount_point/testfolder**$ ls
file1.txt file2.txt **subtestfolder**
prashulkumar@praskumainstance2:**~/my_mount_point/testfolder**$ rmdir subtestfolder/
prashulkumar@praskumainstance2:**~/my_mount_point/testfolder**$ ls
file1.txt file2.txt
prashulkumar@praskumainstance2:**~/my_mount_point/testfolder**$ cd ..
prashulkumar@praskumainstance2:**~/my_mount_point**$ ls
data.txt datqa.txt hello.txt testcase1.txt testf testfile1.txt testfile3.txt testfolder
prashulkumar@praskumainstance2:**~/my_mount_point**$ rmdir testfolder/
rmdir: failed to remove 'testfolder/': Directory not empty
prashulkumar@praskumainstance2:~/my_mount_point$ cd ..
prashulkumar@praskumainstance2:~$ umount my_mount_point

Screenshot depicting the same –

```
prashulkumar@praskumainstance2:~$ cd my_mount_point/
prashulkumar@praskumainstance2:~/my_mount_point$ mkdir testfolder/
prashulkumar@praskumainstance2:~/my_mount_point$ cd testfolder/
prashulkumar@praskumainstance2:~/my_mount_point/testfolder$ mkdir subtestfolder/
prashulkumar@praskumainstance2:~/my_mount_point/testfolder$ touch file1.txt
prashulkumar@praskumainstance2:~/my_mount_point/testfolder$ touch file2.txt
prashulkumar@praskumainstance2:~/my_mount_point/testfolder$ ls
file1.txt  file2.txt  subtestfolder
prashulkumar@praskumainstance2:~/my_mount_point/testfolder$ rmdir subtestfolder/
prashulkumar@praskumainstance2:~/my_mount_point/testfolder$ ls
file1.txt  file2.txt
prashulkumar@praskumainstance2:~/my_mount_point/testfolder$ cd ..
prashulkumar@praskumainstance2:~/my_mount_point$ ls
data.txt  datqa.txt  hello.txt  testcase1.txt  testf  testfile1.txt  testfile3.txt  testfolder
prashulkumar@praskumainstance2:~/my_mount_point$ rmdir testfolder/
rmdir: failed to remove 'testfolder/': Directory not empty
prashulkumar@praskumainstance2:~/my_mount_point$
```

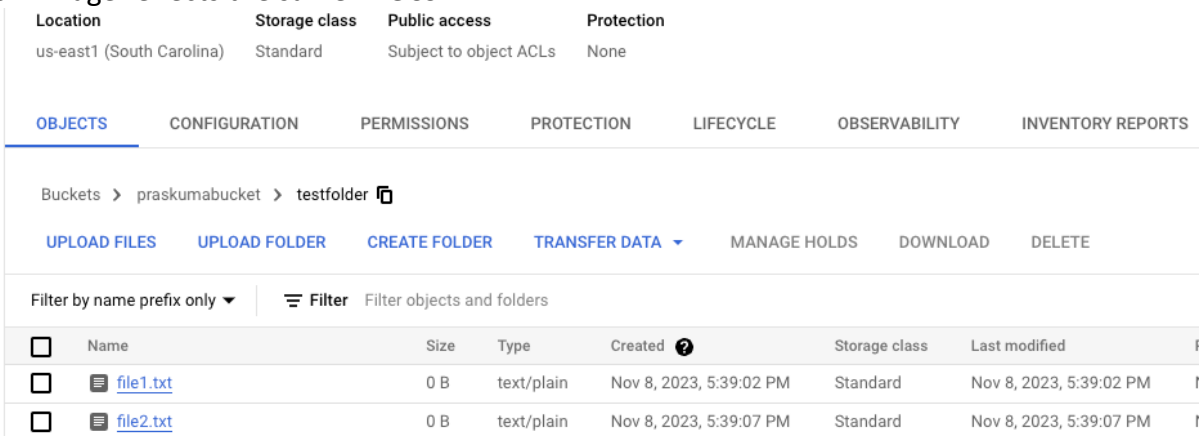
Now let's look at some significant portion of log (full log in log file)–

DEBUG:fuse:FUSE operation getattr raised a <class 'fuse.FuseOSError'>, returning errno 2.
  raise FuseOSError(ENOENT) # File or directory does not exist
DEBUG:fuse.log-mixin:-> mkdir /testfolder (493,)
…
…
DEBUG:fuse.log-mixin:-> mkdir /testfolder/subtestfolder (493,)
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "POST
…
…
DEBUG:fuse.log-mixin:-> create /testfolder/file1.txt (33188,)
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "POST
DEBUG:fuse.log-mixin:-> flush /testfolder/file1.txt (1,)
flush called for path: /testfolder/file1.txt
…
…
DEBUG:fuse.log-mixin:-> opendir /testfolder ()
opendir accessed for path: /testfolder
DEBUG:fuse.log-mixin:-> getattr /testfolder (None,)
DEBUG:fuse.log-mixin:<- getattr **{'st_mode': 16877, 'st_ctime': 1699483023.2785578, 'st_mtime': 1699483023.2785578, 'st_atime': 1699483023.2785578, 'st_nlink': 3, 'st_size': 0, 'is_read_only': False}**
…
…
DEBUG:fuse.log-mixin:-> rmdir /testfolder/subtestfolder ()
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "DELETE /storage/v1/b/praskumabucket/o/testfolder%2Fsubtestfolder%2F?prettyPrint=false HTTP/1.1" 204 0
…
…
DEBUG:fuse.log-mixin:-> rmdir /testfolder ()
Not Empty
DEBUG:fuse.log-mixin:<- rmdir '[Errno 39] Directory not empty'
DEBUG:fuse:FUSE operation rmdir raised a <class 'fuse.FuseOSError'>, returning errno 39.

Now if we see above, that we are able to successfully create a directory (**mkdir**) inside the mounted directory and then 'ls' (**readdir**) command also works successfully and able to list the files/directories. Moreover we also created another directory inside the created directory and also able to test the '**rmdir'** function to remove the empty directory. The logs have been attached which confirm the same (notice the st_nlink). Notice, that we cannot remove the testfolder directory and I have handled in the code to raise this FUSEError, incase the directory

is not empty. This mimics the normal OS behaviour, where we cannot remove a directory which is not empty. But can remove an empty directory using the same.

Below image reflects the same in GCS –



Test Case4: To verify the functionality of the fsync operation, ensuring that data is correctly synchronized to storage.

Terminal1:
prashulkumar@praskumainstance2:~$ python3 testfuse5.py /home/prashulkumar/my_mount_point

Terminal2:
prashulkumar@praskumainstance2:~$ cd my_mount_point/
prashulkumar@praskumainstance2:**~/my_mount_point**$ ls
data.txt datqa.txt hello.txt testcase1.txt testf testfile1.txt testfile3.txt testfolder
prashulkumar@praskumainstance2:**~/my_mount_point**$ touch file10.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ ls
data.txt datqa.txt file10.txt **hello.txt testcase1.txt testf testfile1.txt testfile3.txt testfolder**
prashulkumar@praskumainstance2:**~/my_mount_point**$ nano file10.txt
prashulkumar@praskumainstance2:~$ umount my_mount_point

```
prashulkumar@praskumainstance2:~$ cd my_mount_point/
prashulkumar@praskumainstance2:~/my_mount_point$ ls
data.txt  datqa.txt  hello.txt  testcase1.txt  testf  testfile1.txt  testfile3.txt  testfolder
prashulkumar@praskumainstance2:~/my_mount_point$ touch file10.txt
prashulkumar@praskumainstance2:~/my_mount_point$ ls
data.txt  datqa.txt  file10.txt  hello.txt  testcase1.txt  testf  testfile1.txt  testfile3.txt  testfolder
prashulkumar@praskumainstance2:~/my_mount_point$ nano file10.txt
prashulkumar@praskumainstance2:~/my_mount_point$
```

While in editor mode for file we write the following text to this file and then write out (while still being in editor mode only)

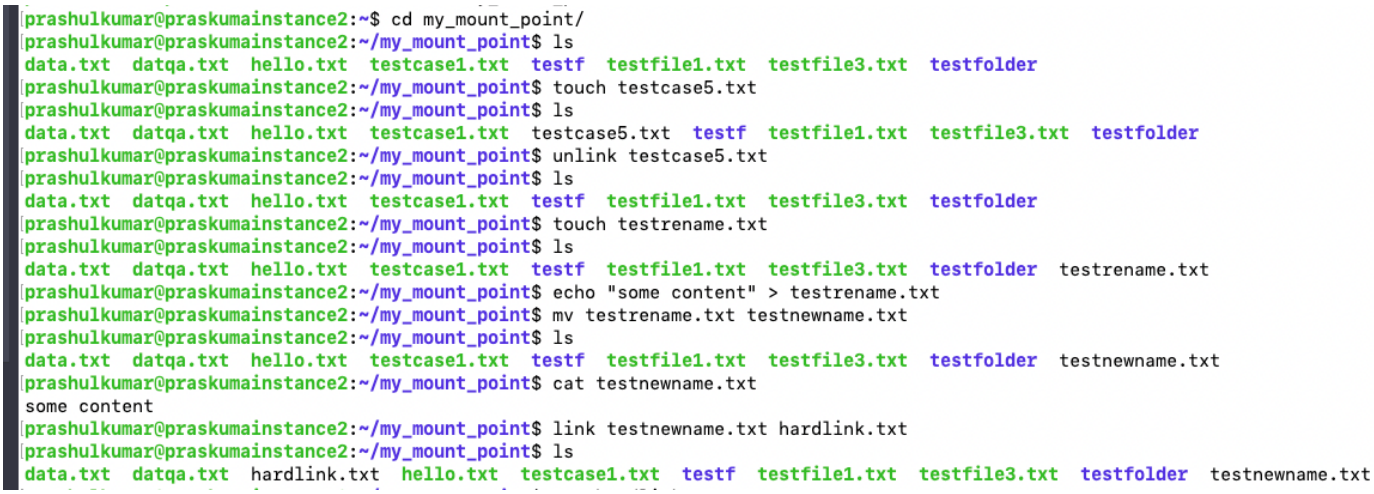
Significant logs –
DEBUG:fuse.log-mixin:-> open /file10.txt (32769,)
DEBUG:fuse.log-mixin:<- open 5
DEBUG:fuse.log-mixin:-> getxattr /file10.txt ('security.capability',)
…
…
DEBUG:fuse.log-mixin:-> write /file10.txt (b'abcd\npqrs\n\n', 0, 5)
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "GET /download/storage/v1/b/praskumabucket/o/file10.txt?alt=media HTTP/1.1" 200 0
DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "POST /upload/storage/v1/b/praskumabucket/o?uploadType=multipart HTTP/1.1" 200 737
DEBUG:fuse.log-mixin:<- write 11
DEBUG:fuse.log-mixin:-> fsync /file10.txt (0, 5)
fsync called for path: /file10.txt, fdatasync: 0, fh: 5
DEBUG:fuse.log-mixin:<- fsync 0
DEBUG:fuse.log-mixin:-> flush /file10.txt (5,)

Here again we can see, when in editor mode, we write out (while file still being open), the fsync is being called in the logs to sync the written data to the file, thus showing the function call as success.

Test Case5: To verify the functionality of link, rename and unlink operation in my file system.

Terminal1:
prashulkumar@praskumainstance2:**~**$ python3 testfuse5.py /home/prashulkumar/my_mount_point

Terminal2:
prashulkumar@praskumainstance2:**~**$ cd my_mount_point/
prashulkumar@praskumainstance2:**~/my_mount_point**$ touch testcase5.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ ls
data.txt datqa.txt hello.txt testcase1.txt testcase5.txt **testf testfile1.txt testfile3.txt testfolder**
prashulkumar@praskumainstance2:**~/my_mount_point**$ unlink testcase5.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ ls
data.txt datqa.txt hello.txt testcase1.txt testf testfile1.txt testfile3.txt testfolder
prashulkumar@praskumainstance2:**~/my_mount_point**$ touch testrename.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ ls
data.txt datqa.txt hello.txt testcase1.txt testf testfile1.txt testfile3.txt testfolder
testrename.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ echo "some content" > testrename.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ mv testrename.txt testnewname.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ ls
data.txt datqa.txt hello.txt testcase1.txt testf testfile1.txt testfile3.txt testfolder
testnewname.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ cat testnewname.txt
some content
prashulkumar@praskumainstance2:**~/my_mount_point**$ link testnewname.txt hardlink.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ ls
data.txt datqa.txt hardlink.txt **hello.txt testcase1.txt testf testfile1.txt testfile3.txt testfolder** testnewname.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ cd ..
prashulkumar@praskumainstance2:**~**$ umount my_mount_point



Significant logs –

DEBUG:fuse.log-mixin:-> create /testcase5.txt (33188,)

…

…

DEBUG:fuse.log-mixin:-> unlink /testcase5.txt ()

DEBUG:urllib3.connectionpool:https://storage.googleapis.com:443 "DELETE /storage/v1/b/praskumabucket/o/testcase5.txt?prettyPrint=false HTTP/1.1" 204 0

…

…

DEBUG:fuse.log-mixin:-> create /testrename.txt (33188,)

…

DEBUG:fuse.log-mixin:-> write /testrename.txt (b'some content\n', 0, 3)

…

…

DEBUG:fuse.log-mixin:-> rename /testrename.txt ('/testnewname.txt',)

…

…

DEBUG:fuse.log-mixin:-> open /testnewname.txt (32768,)

DEBUG:fuse.log-mixin:<- read b'some content\n'

…

…

DEBUG:fuse.log-mixin:-> link /hardlink.txt ('/testnewname.txt',)

DEBUG:fuse.log-mixin:-> getattr /hardlink.txt (None,)

DEBUG:fuse.log-mixin:<- getattr {'st_mode': 33188, 'st_ctime': 1699486815.6382048, 'st_mtime': 1699486815.6382048, 'st_atime': 1699486664.951957, **'st_nlink': 2**, 'st_size': 13, 'is_read_only': False}

Hence we can see from the images as well as the logs that our **‘unlink’**, then **‘rename’** (mv) and finally **‘link’** operation performs as intended and the same is confirmed through logs and screenshot above. Remember **‘flush’** implementation is already shown through logs earlier.

Performance Testing Comparison b/w Native and Fuse(GCS):

Test1:

Initial setup in VM –

prashulkumar@praskumainstance2:~$ sudo apt-get install fio

Now let's make a new directory called test so that all the testing is done inside this folder only and then we can remove the directory after test completion –

First test for our native file system –

prashulkumar@praskumainstance2:~$ mkdir test/
prashulkumar@praskumainstance2:~$ chmod 777 test/
prashulkumar@praskumainstance2:~$ cd test/
prashulkumar@praskumainstance2:**~/test**$ fio --name=test --size=25k --filename=/home/prashulkumar/test/testfile

Results –

```
prashulkumar@praskumainstance2:~$ ls
my_mount_point  prashul-kumar-fall2023-23ecf366ca39.json  test  testfuse5.py  testfuseold.py
prashulkumar@praskumainstance2:~$ cd test/
prashulkumar@praskumainstance2:~/test$ fio --name=test --size=25k --filename=/home/prashulkumar/test/testfile
test: (g=0): rw=read, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=psync, iodepth=1
fio-3.25
Starting 1 process
test: Laying out IO file (1 file / 0MiB)

test: (groupid=0, jobs=1): err= 0: pid=32460: Thu Nov  9 00:15:03 2023
  read: IOPS=1000, BW=4000KiB/s (4096kB/s)(24.0KiB/6msec)
    clat (nsec): min=800, max=3786.6k, avg=868778.33, stdev=1535956.75
     lat (nsec): min=830, max=3787.0k, avg=868893.33, stdev=1536079.86
    clat percentiles (nsec):
     |  1.00th=[    804],  5.00th=[    804], 10.00th=[    804],
     | 20.00th=[    892], 30.00th=[    892], 40.00th=[   1160],
     | 50.00th=[   1160], 60.00th=[  14016], 70.00th=[1417216],
     | 80.00th=[1417216], 90.00th=[3784704], 95.00th=[3784704],
     | 99.00th=[3784704], 99.50th=[3784704], 99.90th=[3784704],
     | 99.95th=[3784704], 99.99th=[3784704]
  lat (nsec)   : 1000=33.33%
  lat (usec)   : 2=16.67%, 20=16.67%
  lat (msec)   : 2=16.67%, 4=16.67%
  cpu          : usr=0.00%, sys=0.00%, ctx=2, majf=0, minf=13
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwts: total=6,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
   READ: bw=4000KiB/s (4096kB/s), 4000KiB/s-4000KiB/s (4096kB/s-4096kB/s), io=24.0KiB (24.6kB), run=6-6msec

Disk stats (read/write):
  sda: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
prashulkumar@praskumainstance2:~/test$
```

test: (g=0): rw=read, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=psync, iodepth=1
fio-3.25

```
Starting 1 process
test: Laying out IO file (1 file / 0MiB)

test: (groupid=0, jobs=1): err= 0: pid=32460: Thu Nov  9 00:15:03 2023
  read: IOPS=1000, BW=4000KiB/s (4096kB/s)(24.0KiB/6msec)
    clat (nsec): min=800, max=3786.6k, avg=868778.33, stdev=1535956.75
     lat (nsec): min=830, max=3787.0k, avg=868893.33, stdev=1536079.86
    clat percentiles (nsec):
     |  1.00th=[    804],  5.00th=[    804], 10.00th=[    804],
     | 20.00th=[    892], 30.00th=[    892], 40.00th=[   1160],
     | 50.00th=[   1160], 60.00th=[  14016], 70.00th=[1417216],
     | 80.00th=[1417216], 90.00th=[3784704], 95.00th=[3784704],
     | 99.00th=[3784704], 99.50th=[3784704], 99.90th=[3784704],
     | 99.95th=[3784704], 99.99th=[3784704]
  lat (nsec)   : 1000=33.33%
  lat (usec)   : 2=16.67%, 20=16.67%
  lat (msec)   : 2=16.67%, 4=16.67%
  cpu          : usr=0.00%, sys=0.00%, ctx=2, majf=0, minf=13
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwts: total=6,0,0,0 short=0,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
   READ: bw=4000KiB/s (4096kB/s), 4000KiB/s-4000KiB/s (4096kB/s-4096kB/s), io=24.0KiB
(24.6kB), run=6-6msec

Disk stats (read/write):
  sda: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
```

Now let's go for our mounted gcs_fuse system –

prashulkumar@praskumainstance2:**~**$ python3 testfuse5.py
/home/prashulkumar/my_mount_point

prashulkumar@praskumainstance2:**~**$ cd my_mount_point/
prashulkumar@praskumainstance2:**~/my_mount_point**$ ls
data.txt datqa.txt hello.txt testcase1.txt testf testfile1.txt testfile3.txt testfolder testnewname.txt
prashulkumar@praskumainstance2:**~/my_mount_point**$ mkdir test/

prashulkumar@praskumainstance2:**~/my_mount_point**$ chmod 777 test
prashulkumar@praskumainstance2:**~/my_mount_point**$ ls
data.txt datqa.txt hello.txt test **testcase1.txt testf testfile1.txt testfile3.txt testfolder testnewname.txt**
prashulkumar@praskumainstance2:**~/my_mount_point**$ cd test/
prashulkumar@praskumainstance2:**~/my_mount_point/test**$ fio --name=test --size=25k --filename=/home/prashulkumar/my_mount_point/test/testfile

```
prashulkumar@praskumainstance2:~/my_mount_point$ ls
data.txt  datqa.txt  hello.txt  test  testcase1.txt  testf  testfile1.txt  testfile3.txt  testfolder  testnewname.txt
prashulkumar@praskumainstance2:~/my_mount_point$ cd test/
prashulkumar@praskumainstance2:~/my_mount_point/test$ fio --name=test --size=25k --filename=/home/prashulkumar/my_mount_point/test/testfile
test: (g=0): rw=read, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=psync, iodepth=1
fio-3.25
Starting 1 process
test: Laying out IO file (1 file / 0MiB)
fio: pid=32494, err=5/file:backend.c:479, func=full resid, error=Input/output error

test: (groupid=0, jobs=1): err= 5 (file:backend.c:479, func=full resid, error=Input/output error): pid=32494: Thu Nov  9 00:19:19 2023
  read: IOPS=138, BW=333KiB/s (341kB/s)(12.0KiB/36msec)
    clat (nsec): min=669, max=35460k, avg=11821201.00, stdev=20471535.49
     lat (nsec): min=760, max=35460k, avg=11821575.00, stdev=20471696.58
    clat percentiles (nsec):
     |  1.00th=[     668],  5.00th=[     668], 10.00th=[     668],
     | 20.00th=[     668], 30.00th=[     668], 40.00th=[    3248],
     | 50.00th=[    3248], 60.00th=[    3248], 70.00th=[35389440],
     | 80.00th=[35389440], 90.00th=[35389440], 95.00th=[35389440],
     | 99.00th=[35389440], 99.50th=[35389440], 99.90th=[35389440],
     | 99.95th=[35389440], 99.99th=[35389440]
  lat (nsec)   : 750=20.00%
  lat (usec)   : 4=20.00%
  lat (msec)   : 50=20.00%
  cpu          : usr=0.00%, sys=0.00%, ctx=2, majf=0, minf=21
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=28.6%, 4=71.4%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwts: total=5,0,0,0 short=1,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
   READ: bw=333KiB/s (341kB/s), 333KiB/s-333KiB/s (341kB/s-341kB/s), io=12.0KiB (12.3kB), run=36-36msec
prashulkumar@praskumainstance2:~/my_mount_point/test$
```

test: (g=0): rw=read, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=psync, iodepth=1
fio-3.25
Starting 1 process
test: Laying out IO file (1 file / 0MiB)
fio: pid=32494, err=5/file:backend.c:479, func=full resid, error=Input/output error

test: (groupid=0, jobs=1): err= 5 (file:backend.c:479, func=full resid, error=Input/output error): pid=32494: Thu Nov 9 00:19:19 2023
  read: IOPS=138, BW=333KiB/s (341kB/s)(12.0KiB/36msec)
    clat (nsec): min=669, max=35460k, avg=11821201.00, stdev=20471535.49
     lat (nsec): min=760, max=35460k, avg=11821575.00, stdev=20471696.58
    clat percentiles (nsec):
     | 1.00th=[668], 5.00th=[668], 10.00th=[668],
     | 20.00th=[668], 30.00th=[668], 40.00th=[3248],
     | 50.00th=[3248], 60.00th=[3248], 70.00th=[35389440],

```
     | 80.00th=[35389440], 90.00th=[35389440], 95.00th=[35389440],
     | 99.00th=[35389440], 99.50th=[35389440], 99.90th=[35389440],
     | 99.95th=[35389440], 99.99th=[35389440]
  lat (nsec)   : 750=20.00%
  lat (usec)   : 4=20.00%
  lat (msec)   : 50=20.00%
  cpu          : usr=0.00%, sys=0.00%, ctx=2, majf=0, minf=21
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=28.6%, 4=71.4%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     issued rwts: total=5,0,0,0 short=1,0,0,0 dropped=0,0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=1

Run status group 0 (all jobs):
   READ: bw=333KiB/s (341kB/s), 333KiB/s-333KiB/s (341kB/s-341kB/s), io=12.0KiB (12.3kB),
run=36-36msec
```

Note: The fio test was run only for a size of 25Kb due to the limitations of the GCS object storage method. If I give the file size anything greater than 25Kb then the TooMany requests error of GCS bucket prevents me from doing test for file size > 25 Kb. Hence this was the limitation of using fio for testing in GCS mounted directory.

```
    self._process_response(result)
  File "/home/prashulkumar/.local/lib/python3.9/site-packages/google/resumable_media/_upload.py", line 125, in _process_response
    _helpers.require_status_code(response, (http.client.OK,), self._get_status_code)
  File "/home/prashulkumar/.local/lib/python3.9/site-packages/google/resumable_media/_helpers.py", line 108, in require_status_code
    raise common.InvalidResponse(
google.resumable_media.common.InvalidResponse: ('Request failed with status code', 429, 'Expected one of', <HTTPStatus.OK: 200>)

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/prashulkumar/.local/lib/python3.9/site-packages/fuse.py", line 734, in _wrapper
    return func(*args, **kwargs) or 0
  File "/home/prashulkumar/.local/lib/python3.9/site-packages/fuse.py", line 866, in write
    return self.operations('write', self._decode_optional_path(path), data,
  File "/home/prashulkumar/.local/lib/python3.9/site-packages/fuse.py", line 1251, in __call__
    ret = getattr(self, op)(path, *args)
  File "/home/prashulkumar/testfuse5.py", line 171, in write
    blob.upload_from_string(new_content)
  File "/home/prashulkumar/.local/lib/python3.9/site-packages/google/cloud/storage/blob.py", line 3056, in upload_from_string
    self.upload_from_file(
  File "/home/prashulkumar/.local/lib/python3.9/site-packages/google/cloud/storage/blob.py", line 2763, in upload_from_file
    self._prep_and_do_upload(
  File "/home/prashulkumar/.local/lib/python3.9/site-packages/google/cloud/storage/blob.py", line 2622, in _prep_and_do_upload
    _raise_from_invalid_response(exc)
  File "/home/prashulkumar/.local/lib/python3.9/site-packages/google/cloud/storage/blob.py", line 4778, in _raise_from_invalid_response
    raise exceptions.from_http_status(response.status_code, message, response=response)
google.api_core.exceptions.TooManyRequests: 429 POST https://storage.googleapis.com/upload/storage/v1/b/praskumabucket/o?uploadType=multipart: {
  "error": {
    "code": 429,
    "message": "The object praskumabucket/test/testfile exceeded the rate limit for object mutation operations (create, update, and delete). Please reduce your request rate. See https://cloud.google.com/storage/d
ocs/gcs429.",
    "errors": [
      {
        "message": "The object praskumabucket/test/testfile exceeded the rate limit for object mutation operations (create, update, and delete). Please reduce your request rate. See https://cloud.google.com/stora
ge/docs/gcs429.",
        "domain": "usageLimits",
        "reason": "rateLimitExceeded"
      }
```

Now if we compare the results –

Native FS Results:

- Read IOPS :1000
- Bandwidth: 4000 KiB/s (4096 KB/s)
- Average Read Latency: 868,778.33 ns
- Maximum Read Latency: 3,787,000 ns
- Standard Deviation of Read Latency: 1,535,956.75 ns

GCS Mounted Directory Results:

- Read IOPS: 138
- Bandwidth: 333 KiB/s (341 KB/s)
- Average Read Latency: 11,821,201.00 ns
- Maximum Read Latency: 35,460,000 ns
- Standard Deviation of Read Latency: 20,471,535.49 ns

In the Native FS scenario, the system demonstrated significantly higher read performance compared to the GCS Mounted Directory. The Native FS achieved 1000 IOPS with an average latency of 868,778.33 ns.
In contrast, the GCS Mounted Directory had a lower read performance, achieving 138 IOPS with an average latency of 11,821,201.00 ns.

Conclusion:
The Native FS exhibited superior read performance in terms of both IOPS and latency, making it the preferred choice for scenarios where high read performance is essential.
These results indicate that the Native FS is better suited for read-heavy workloads, while the GCS Mounted Directory would have performance limitations due to its interaction with cloud storage as well as the too-many read requests limit set on the GCS bucket.

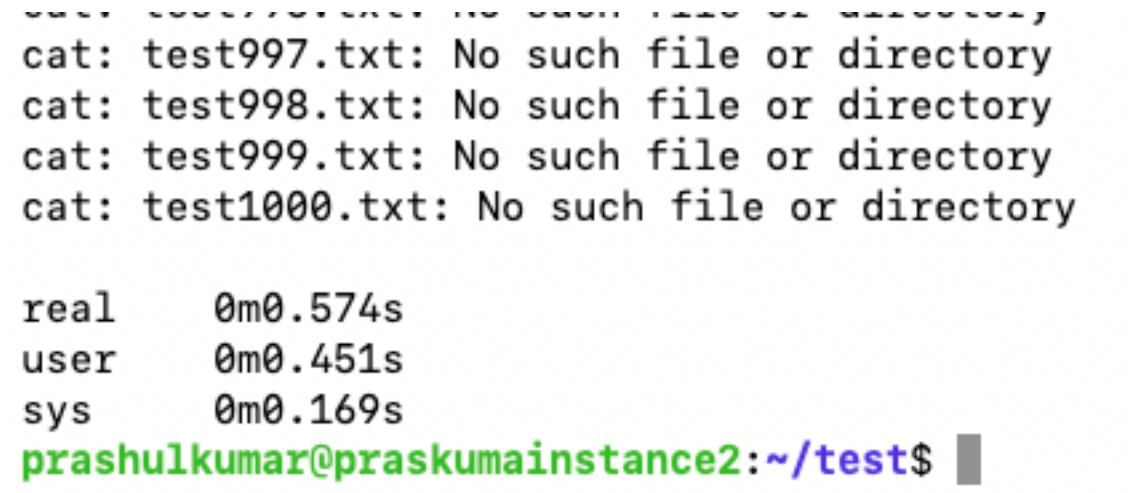
Now after cleanup (removing the test directory), lets try the second test –

Test2:

Native F/S:

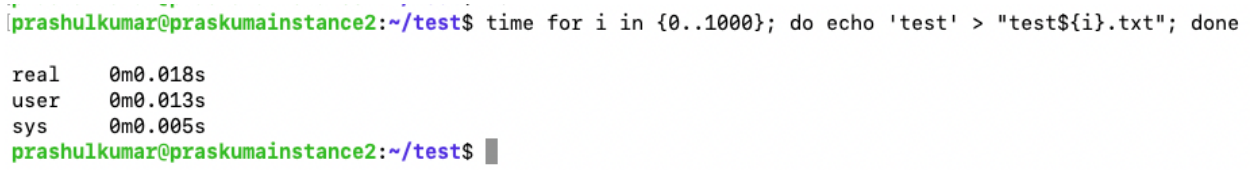
prashulkumar@praskumainstance2:**~**$ rm -rf test
prashulkumar@praskumainstance2:**~**$ ls
my_mount_point prashul-kumar-fall2023-23ecf366ca39.json testfuse5.py testfuseold.py
prashulkumar@praskumainstance2:**~**$ mkdir test/

prashulkumar@praskumainstance2:**~**$ chmod 777 test
prashulkumar@praskumainstance2:**~**$ ls
my_mount_point prashul-kumar-fall2023-23ecf366ca39.json test testfuse5.py testfuseold.py
prashulkumar@praskumainstance2:**~**$ cd test
prashulkumar@praskumainstance2:**~/test**$ ls
prashulkumar@praskumainstance2:**~/test**$ time for i in {0..1000}; do cat "test${i}.txt" > /dev/null; done



real 0m0.574s
user 0m0.451s
sys 0m0.169s

prashulkumar@praskumainstance2:~/test$ time for i in {0..1000}; do echo 'test' > "test${i}.txt"; done



real 0m0.018s
user 0m0.013s
sys 0m0.005s

GCS Mounted directory:

prashulkumar@praskumainstance2:**~/my_mount_point**$ mkdir test/
prashulkumar@praskumainstance2:**~/my_mount_point**$ chmod 777 test/
prashulkumar@praskumainstance2:**~/my_mount_point**$ cd test/
prashulkumar@praskumainstance2:**~/my_mount_point/test**$ time for i in {0..1000}; do cat "test${i}.txt" > /dev/null; done

```
cat: test997.txt: No such file or directory
cat: test998.txt: No such file or directory
cat: test999.txt: No such file or directory
cat: test1000.txt: No such file or directory

real    0m0.885s
user    0m0.514s
sys     0m0.163s
prashulkumar@praskumainstance2:~/my_mount_poin
```

real 0m0.885s
user 0m0.514s
sys 0m0.163s

prashulkumar@praskumainstance2:~/my_mount_point/test$ time for i in {0..1000}; do echo 'test' > "test${i}.txt"; done

```
prashulkumar@praskumainstance2:~/my_mount_point/test$ time for i in {0..1000}; do echo 'test' > "test${i}.txt"; done

real    1m45.396s
user    0m0.021s
sys     0m0.218s
prashulkumar@praskumainstance2:~/my_mount_point/test$
```

real 1m45.396s
user 0m0.021s
sys 0m0.218s

Summary –

Read Operation:

- **Native FS Result:**
  - Real Time: 0m0.574s
  - User Time: 0m0.451s
  - Sys Time: 0m0.169s
- **GCS Mounted Directory Result:**
  - Real Time: 0m0.885s
  - User Time: 0m0.514s
  - Sys Time: 0m0.163s

Write Operation:

- **Native FS Result:**
  - Real Time: 0m0.018s
  - User Time: 0m0.013s
  - Sys Time: 0m0.005s
- **GCS Mounted Directory Result:**
  - Real Time: 1m45.396s
  - User Time: 0m0.021s
  - Sys Time: 0m0.218s

Read Operation: The Native FS outperformed the GCS Mounted Directory in the read operation, with significantly lower real, user, and sys times. The GCS Mounted Directory showed slower read performance.

Write Operation: The Native FS demonstrated superior write performance, completing the operation much faster than the GCS Mounted Directory. The GCS Mounted Directory exhibited significantly higher real time, indicating a delay in write operations.

Conclusion: For both read and write operations, the Native FS consistently showed better performance compared to the GCS Mounted Directory. Hence, local file systems provide faster data access and modification compared to file systems interacting with cloud storage.

Future Improvements –

As of now, the major improvement I see are –

- Error Handling: The edge cases for the mounted file system has not been taken care of. Hence if there is any bad input or many other cases which can come in a file system, which have not been taken care of. The future scope is to handle the exits more gracefully for different types of errors.

- Concurrent writes: As of now, if I access the same file through multiple different terminals, I am able to write to the same file through each of them. There would be potential race conditions in this case eventually leading to data corruption or loss, which is not something we want. Hence, the second area of improvement is to implement file locks through threading module, so that, if 1 person has write access, then if the second person tries to write the same file, they may not be able to do so, due to the lock mechanism, thus making the file as read-only unless the first person completes their operation.

Reference:

https://docs.gitlab.com/ee/administration/operations/filesystem_benchmarking.html

https://github.com/fusepy/fusepy/blob/master/examples/memory.py

https://www.stavros.io/posts/python-fuse-filesystem/