Practical Work 6 – GlusterFS Replicated Volume Benchmark

Nguyen Tien Dung – 23BI14116

December 6, 2025

1 Introduction

The goal of this practical work is to deploy a distributed file system using GlusterFS, configure a replicated volume and evaluate its performance compared to a local file system. The experiment focuses on:

- Functional setup of GlusterFS (bricks, volume, mount).
- Measuring performance for **small files** (operations per second).
- Measuring performance for **large files** (MB/s).

Due to hardware constraints, all bricks are hosted on a single virtual machine. Logically, GlusterFS still manages a replicated volume with multiple bricks, but physically the bricks reside on the same host. This limitation is discussed in the analysis section.

2 Environment

2.1 Hardware and OS

- Single VM running Kali Linux (x86_64).
- Disk: virtual disk provided by the hypervisor (ext4 filesystem).
- All tests are executed on this VM, both local and GlusterFS benchmarks.

2.2 Software

- GlusterFS server and client:

```
sudo apt update
sudo apt install -y glusterfs-server glusterfs-client

sudo systemctl enable --now glusterd
sudo systemctl status glusterd # Active (running)
```

- Shell: `zsh` on Kali.
- Benchmark tools:
  - `dd` (for large file throughput).
  - Custom shell script using `date` and `bc` (for small file operations/s).

3 GlusterFS setup

3.1 Brick layout

Even though there is only a single VM, three bricks are created on the same host to emulate three GlusterFS storage units:

```
sudo mkdir -p /data/brick1/gv0
sudo mkdir -p /data/brick2/gv0
sudo mkdir -p /data/brick3/gv0
```

Each directory (`/data/brickX/gv0`) serves as a brick path. The host name of the machine is:

```
hostname
# kali
```

3.2 Volume creation

A replicated volume with three bricks is created and started:

```
sudo gluster volume create gv0 replica 3 transport tcp \
  kali:/data/brick1/gv0 \
  kali:/data/brick2/gv0 \
  kali:/data/brick3/gv0 force

sudo gluster volume start gv0
sudo gluster volume info gv0
```

The relevant part of `volume info` is:

- `Volume Name: gv0`
- `Type: Replicate`
- `Number of Bricks: 1 x 3 = 3`
- `Status: Started`

3.3 Mounting the volume

The volume `gv0` is mounted locally at `/mnt/gv0`:

```
sudo mkdir -p /mnt/gv0
sudo mount -t glusterfs kali:/gv0 /mnt/gv0

df -h | grep gv0
# kali:/gv0 ... /mnt/gv0
```

Because the mount is owned by `root`, permissions are adjusted to allow the normal user to run benchmarks:

```
sudo chown -R suiikawaii:suiikawaii /mnt/gv0
```

A quick functional test confirms that the volume is writable and readable:

```
cd /mnt/gv0
echo "hello from gluster" > test.txt
cat test.txt
# hello from gluster
```

4 Benchmark methodology

Two kinds of benchmarks are considered:

1. **Small files**: creating and reading many small files, measuring operations per second (ops/s).

2. **Large file**: writing and reading a 1 GB file, measuring throughput (MB/s).

All benchmarks are run on:

- **Local ext4 baseline**: directory `~/local_test`.
- **GlusterFS replicate-3 volume**: mount point `/mnt/gv0`.

4.1 Small file benchmark

A shell script `smallfile_bench.sh` is used:

```
#!/bin/bash
MNT=${1:-/mnt/gv0}
N=${2:-1000}
DIR=$MNT/smalltest_$RANDOM

mkdir -p "$DIR"

echo "Target dir: $DIR"
echo "Writing $N small files..."
t1=$(date +%s.%N)
for i in $(seq 1 $N); do
  echo "hello_$i" > "$DIR/file_$i"
done
sync
t2=$(date +%s.%N)

echo "Reading $N small files..."
t3=$(date +%s.%N)
for i in $(seq 1 $N); do
  cat "$DIR/file_$i" > /dev/null
done
t4=$(date +%s.%N)

w_time=$(echo "$t2-$t1" | bc)
r_time=$(echo "$t4-$t3" | bc)

w_ops=$(echo "scale=2; $N / $w_time" | bc)
r_ops=$(echo "scale=2; $N / $r_time" | bc)

echo "Write time: $w_time s => $w_ops ops/s"
echo "Read time: $r_time s => $r_ops ops/s"
```

The number of files is set to $N = 2000$. The script is executed as:

```
mkdir -p ~/local_test

# Local ext4 baseline
./smallfile_bench.sh ~/local_test 2000

# GlusterFS gv0
./smallfile_bench.sh /mnt/gv0 2000
```

4.2 Large file benchmark

For large files, `dd` is used to write and read a 1 GB file (1024×1 MB blocks). The built-in shell `time` command is used to report elapsed time and `dd` prints MB/s.

Local ext4:

```
cd ~/local_test

sync
time dd if=/dev/zero of=largefile bs=1M count=1024 oflag=direct

sync
time dd if=largefile of=/dev/null bs=1M iflag=direct
```

GlusterFS gv0:

```
cd /mnt/gv0

sync
time dd if=/dev/zero of=largefile bs=1M count=1024 oflag=direct

sync
time dd if=largefile of=/dev/null bs=1M iflag=direct
```

5 Results

5.1 Small files: operations per second

For $N = 2000$ files:

- **Local ext4 (~/local_test)**:
  - Write: `Write time:  0.171241606 s => 11679.40 ops/s`
  - Read: `Read time:  3.035663430 s => 658.83 ops/s`
- **GlusterFS replica 3 (/mnt/gv0)**:
  - Write: `Write time:  7.080283163 s => 282.47 ops/s`
  - Read: `Read time:  4.928230655 s => 405.82 ops/s`

These values are summarized in Table 1.

Backend	Write time (s)	Write (ops/s)	Read time (s)	Read (ops/s)
Local ext4	0.171	11679.40	3.036	658.83
GlusterFS gv0	7.080	282.47	4.928	405.82

Table 1: Small file benchmark (2000 files).

5.2 Large file: throughput (MB/s)

For a 1 GB file (`bs=1M, count=1024`):

Local ext4:

- Write: `934 MB/s`
- Read: `1.1 GB/s` ($\approx$ 1100 MB/s)

GlusterFS gv0:

- Write: `176 MB/s`
- Read: `672 MB/s`

These values are summarized in Table 2.

Backend	Write throughput	Read throughput
Local ext4	934 MB/s	≈ 1100 MB/s
GlusterFS gv0	176 MB/s	672 MB/s

Table 2: Large file benchmark (1 GB).

6 Discussion

6.1 Small files

The small file benchmark shows a very high write rate on the local filesystem (≈ 11,679 ops/s) and moderately high read rate (≈ 659 ops/s). In contrast, GlusterFS delivers significantly lower write performance (≈ 282 ops/s) and slower reads (≈ 406 ops/s).

The reasons include:

- GlusterFS introduces extra layers: network protocol, replication logic, metadata updates and consistency checks.
- For each small file, the overhead of opening, replicating and closing the file is relatively large compared to the payload size.
- Even though all bricks are on the same VM, GlusterFS still behaves as a distributed system and does not optimize for "single-host" mode.

Overall, this is expected: distributed replicated filesystems are known to have higher overhead for many small file operations.

6.2 Large file

For large sequential I/O, the difference is smaller in relative terms but still significant:

- Local ext4 can write the 1 GB file at about 934 MB/s and read it at approximately 1.1 GB/s.
- GlusterFS achieves about 176 MB/s for writes and 672 MB/s for reads.

Writes are notably slower on GlusterFS because each block must be replicated to three bricks and metadata must be coordinated. Read performance is better than writes (672 MB/s), but still below the local filesystem due to the additional software layers and the fact that all three bricks are on the same physical disk, so no real parallelism is gained.

It is likely that on a real cluster with bricks on different physical machines and disks, read performance could benefit from parallelism, while write performance would still pay the cost of replication.

6.3 Limitations

The main limitation of this experiment is that all three bricks are hosted on the same VM, sharing the same underlying disk and CPU. As a result:

- The "number of servers" is logical (3 bricks) rather than physical (3 independent machines).
- There is no true network latency or parallel disk throughput.

However, the results still clearly demonstrate the overhead introduced by a replicated distributed file system compared to a local filesystem, and they are sufficient for illustrating the concepts requested by the practical work.

7 Personal work

For this practical, I performed the following tasks:

- Installed and configured GlusterFS server and client on a Kali Linux VM.
- Created three bricks on the same host and configured a replicated volume (`gv0`) with three bricks.
- Mounted the volume at `/mnt/gv0` and verified functional read/write operations.
- Implemented a small file benchmark script (`smallfile_bench.sh`) to measure operations per second for creating and reading many small files.
- Used `dd` to measure large file throughput (MB/s) on both the local filesystem and GlusterFS.
- Collected and summarized performance results and wrote this report comparing local vs. GlusterFS behavior, including a discussion of limitations and expected overhead.

8 Conclusion

This practical demonstrates how to set up a replicated GlusterFS volume and how to evaluate its performance using simple benchmarks. Even on a single VM with all bricks on the same host, the experiments show a clear trade-off:

- Local ext4 provides much higher performance for both small file operations and large sequential I/O.
- GlusterFS introduces overhead due to replication and distributed management, which significantly reduces throughput for small files and moderately impacts large file performance.

In a real multi-node deployment, GlusterFS would provide higher availability and potentially better scalability, but at the cost of increased complexity and resource usage. This lab helps to understand these trade-offs in practice.