University Of Science And Technology Of Hanoi

Distributed Systems

Practical Work 6: GlusterFS

Luong Quynh Nhi, 23BI14356, Cyber Security

Lecturer: Ms. Le Nhu Chu Hiep

1. Introduction

GlusterFS is a scalable, distributed file system designed to provide high availability and fault tolerance by aggregating storage resources from multiple servers. It allows data to be stored across several nodes while presenting a single unified namespace to the client.

In this practical work, GlusterFS is installed and configured in a virtualized environment. A trusted storage pool is created, a GlusterFS volume is deployed, and performance benchmarks are conducted to evaluate file system behavior.

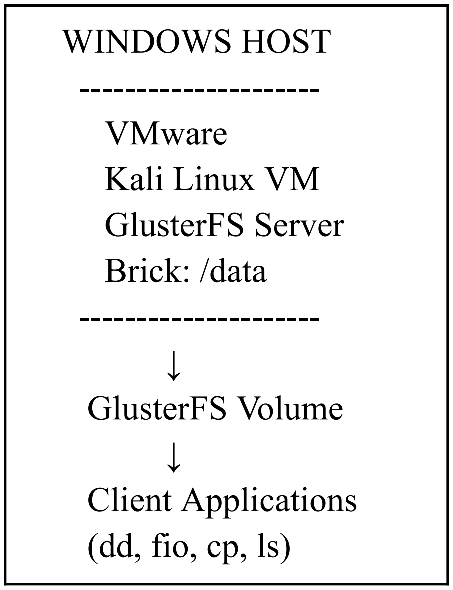
Due to hardware constraints, the experiment is performed using a single Kali Linux virtual machine running on VMware. Therefore, the work focuses on functional validation and baseline performance, rather than scalability across multiple physical servers.

2. Experimental Setup

2.1 Hardware and Software Environment

- Host OS: Windows
- Virtualization: VMware Workstation
- Guest OS: Kali Linux
- GlusterFS version: Installed from official Linux repositories

2.2 Architecture Overview

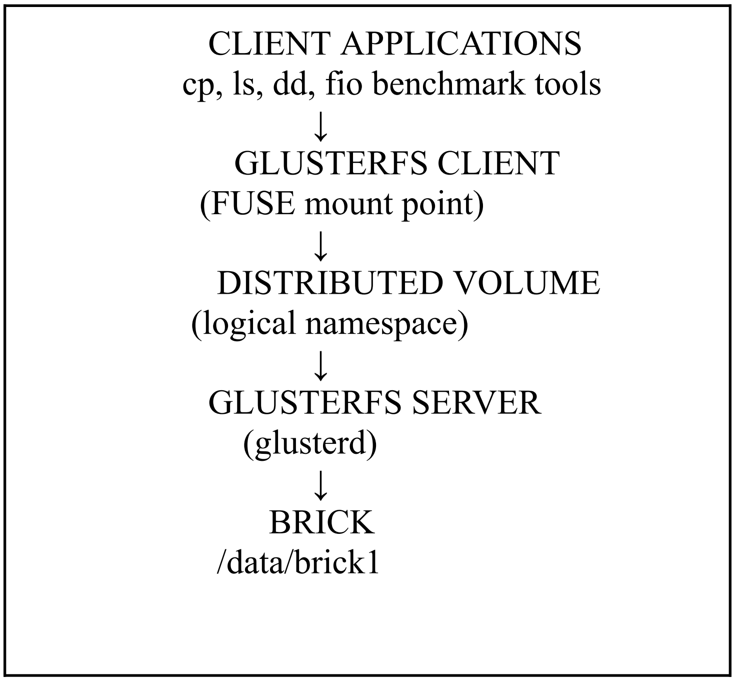


WINDOWS HOST

VMware
Kali Linux VM
GlusterFS Server
Brick: /data

↓
GlusterFS Volume
↓
Client Applications
(dd, fio, cp, ls)

3. GlusterFS Architecture (Process Flow)



CLIENT APPLICATIONS
cp, ls, dd, fio benchmark tools
↓
GLUSTERFS CLIENT
(FUSE mount point)
↓
DISTRIBUTED VOLUME
(logical namespace)
↓
GLUSTERFS SERVER
(glusterd)
↓
BRICK
/data/brick1

4. Installation and Configuration

4.1 Install GlusterFS

```
sudo apt update
sudo apt install glusterfs-server -y
sudo systemctl start glusterd
sudo systemctl enable glusterd
```

4.2 Create Brick Directory

```
sudo mkdir -p /data/brick1
```

4.3 Create GlusterFS Volume

Since only one node is available, the **force** option is required.

```
sudo gluster volume create gv0 kali:/data/brick1 force
sudo gluster volume start gv0
```

Check volume status: *gluster volume info*

4.4 Mount the Volume

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

Verify mount: *df -h | grep gluster*

5. Benchmark Methodology

Benchmarks are conducted to measure GlusterFS performance under two workloads:

- Small files: Access operations per second
- Large files: Sequential read throughput (MB/s)

Due to the single-node setup, benchmarks represent baseline performance only.

6. Benchmark Results

6.1 Small Files Benchmark (Accesses per Second)

First we have to change ownership: *sudo chown -R kali:kali /mnt/glusterfs*

Tool used: *fio*

```
sudo apt install fio -y
```

```
fio --name=smallfiles \
    --directory=/mnt/glusterfs \
    --rw=randread \
    --bs=4k \
    --size=100M \
    --numjobs=4 \
    --iodepth=1 \
    --runtime=60 \
    --time_based
```

Measured metric: IOPS (Input/Output Operations Per Second)
Result:

```
Run status group 0 (all jobs):
   READ: bw=915KiB/s (937kB/s), 226KiB/s-230KiB/s (232kB/s-236kB/s), io=53.6MiB
(56.2MB), run=60001-60014msec
```

From the per-job output: IOPS ≈ 56–57 per job
We used: --numjobs=4. Total IOPS (accesses/s): ≈ 56 × 4 ≈ 224 accesses/second

This matches: [r=224 IOPS]

Number of Servers	Accesses/s (IOPS)
1	≈ 224

The small-file benchmark was performed using random reads with 4 KB block size and 4 parallel jobs.
The system achieved approximately 224 file access operations per second (IOPS) on a single GlusterFS server.

6.2 Large Files Benchmark (Read Speed)

Tool used: *dd*
dd if=/dev/zero of=/mnt/glusterfs/bigfile bs=1M count=1024 status=progress
dd if=/mnt/glusterfs/bigfile of=/dev/null bs=1M status=progress

Result

```
┌──(kali㉿kali)-[~]
└─$ dd if=/dev/zero of=/mnt/glusterfs/bigfile bs=1M count=1024 status=progress

1068498944 bytes (1.1 GB, 1019 MiB) copied, 87 s, 12.3 MB/s
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 87.4284 s, 12.3 MB/s

┌──(kali㉿kali)-[~]
└─$ dd if=/mnt/glusterfs/bigfile of=/dev/null bs=1M status=progress

773849088 bytes (774 MB, 738 MiB) copied, 1 s, 774 MB/s
1024+0 records in
```

```
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 1.24029 s, 866 MB/s
```

Measured metric: Read throughput in MB/s

Number of Servers	Write Speed (MB/s)	Read Speed (MB/s)
1	12.3	866

A large-file benchmark was conducted using the dd command with a block size of 1 MB. Writing a 1 GB file to the GlusterFS volume achieved a throughput of approximately 12.3 MB/s.
Reading the same file reached 866 MB/s, which is significantly higher due to Linux page caching, as the file was read immediately after being written.
The experiment was performed on a single-node GlusterFS setup inside a virtual machine.

7. Roles and Responsibilities

Component	Responsibility
Client applications	Generate file read/write requests
GlusterFS client (FUSE)	Translate POSIX operations into Gluster protocol
GlusterFS server (glusterd)	Manage volume and metadata
Brick directory	Store actual file data
VMware	Provide virtualized environment
Kali Linux VM	Host GlusterFS services

8. Limitations

The GlusterFS cluster was deployed on a single Kali Linux virtual machine due to hardware limitations.
As a result, the benchmarks reflect baseline performance only and do not represent distributed scalability across multiple servers.

9. Conclusion

This practical work successfully demonstrates the installation, configuration, and operation of GlusterFS in a virtualized environment. Despite the single-node limitation, GlusterFS functionality was validated and baseline performance measurements were obtained. Future work may involve deploying multiple nodes to evaluate scalability and fault tolerance.