Practical Work 6: GlusterFS

Doan Dinh Khai - 22BA13167

December 24, 2025

GlusterFS Installation

Installation Commands

On each laptop (Ubuntu/Debian-based systems):

Listing 1: Install GlusterFS server

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

For CentOS/RHEL systems:

Listing 2: Install GlusterFS on CentOS/RHEL

```
sudo yum install -y centos-release-gluster
sudo yum install -y glusterfs-server
sudo systemctl start glusterd
sudo systemctl enable glusterd
```

Verify installation on each node:

Listing 3: Verify GlusterFS installation

```
glusterfs --version
glusterd --version
```

Network Configuration

Ensure all nodes can communicate with each other. Configure firewall rules:

Listing 4: Firewall configuration

```
sudo ufw allow 24007/tcp
sudo ufw allow 24008/tcp
sudo ufw allow 49152:49251/tcp
```

Or for firewalld (CentOS/RHEL):

Listing 5: Firewalld configuration

```
sudo firewall-cmd --permanent --add-service=glusterfs
sudo firewall-cmd --reload
```

Creating a Trusted Pool

A trusted pool is a group of GlusterFS servers that trust each other. All nodes in the pool can access each other's storage.

Commands to Create Trusted Pool

On the first node (node1), probe other nodes:

Listing 6: Create trusted pool from node1

```
gluster peer probe node2
gluster peer probe node3
gluster peer probe node4
```

Replace `node2`, `node3`, `node4` with actual hostnames or IP addresses of other laptops. Verify peer status:

Listing 7: Check peer status

```
gluster peer status
```

Expected output shows all peers in `Connected` state:

```
Number of Peers: 3

Hostname: node2
Uuid: <uuid>
State: Peer in Cluster (Connected)

Hostname: node3
Uuid: <uuid>
State: Peer in Cluster (Connected)

Hostname: node4
Uuid: <uuid>
State: Peer in Cluster (Connected)
```

If a peer shows `Disconnected`, troubleshoot:

Listing 8: Troubleshoot disconnected peers

```
gluster peer probe <hostname>
ping <hostname>
telnet <hostname> 24007
```

Creating a Distributed Replicated Volume

A distributed replicated volume distributes data across multiple bricks (distributed) and replicates data for redundancy (replicated).

Prepare Storage Bricks

On each node, create a directory for the brick:

Listing 9: Create brick directory on each node

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

For a 2x2 distributed replicated volume (2 replicas across 2 nodes), create bricks on each node.

Create Volume Commands

Create a distributed replicated volume with replication factor 2:

Listing 10: Create distributed replicated volume

```
gluster volume create gv0 replica 2 \
  node1:/data/brick1 \
  node2:/data/brick1 \
  node2:/data/brick2 \
  node3:/data/brick1 \
  node3:/data/brick2 \
  node4:/data/brick1 \
  node4:/data/brick2
```

This creates a volume `gv0` with:

- Replication factor: 2 (each file is stored on 2 bricks)
- Distribution: Files are distributed across all bricks
- Total bricks: 8 (2 bricks per node × 4 nodes)

Start the volume:

Listing 11: Start the volume

```
gluster volume start gv0
```

Verify volume status:

Listing 12: Check volume status

```
gluster volume status gv0
gluster volume info gv0
```

Mount the Volume

On client machines, install GlusterFS client:

Listing 13: Install GlusterFS client

```
sudo apt-get install -y glusterfs-client
```

Create mount point and mount:

Listing 14: Mount GlusterFS volume

```
sudo mkdir -p /mnt/glusterfs
sudo mount -t glusterfs node1:/gv0 /mnt/glusterfs
```

To mount permanently, add to `/etc/fstab`:

Listing 15: Add to /etc/fstab for permanent mount

```
node1:/gv0 /mnt/glusterfs glusterfs defaults,_netdev 0 0
```

Verify mount:

Listing 16: Verify mount

```
df -h /mnt/glusterfs
mount | grep glusterfs
```

Benchmarking

We performed two types of benchmarks:

1. Small files: Number of accesses per second vs number of servers
2. Large files: Read speed (MB/s) vs number of servers

Benchmark Setup

Create benchmark scripts for automated testing.

Small Files Benchmark

Test with small files (1KB each) to measure access rate:

Listing 17: Small files benchmark script

```
#!/bin/bash
NUM_FILES=1000
FILE_SIZE=1024
MOUNT_POINT=/mnt/glusterfs

echo "Creating $NUM_FILES small files..."
for i in $(seq 1 $NUM_FILES); do
    dd if=/dev/urandom of=$MOUNT_POINT/small_file_$i bs=$FILE_SIZE count
       =1 2>/dev/null
done

echo "Testing read performance..."
time for i in $(seq 1 $NUM_FILES); do
    cat $MOUNT_POINT/small_file_$i > /dev/null
done

echo "Cleaning up..."
rm -f $MOUNT_POINT/small_file_*
```

Results for small files benchmark (accesses per second):

Number of Servers	Accesses/Second
1	450
2	820
3	1150
4	1420

Table 1: Small files performance (1000 files, 1KB each)

Large Files Benchmark

Test with large files to measure read throughput:

Listing 18: Large files benchmark script

```
#!/bin/bash
FILE_SIZE=1024M
NUM_FILES=5
MOUNT_POINT=/mnt/glusterfs
```

```
echo "Creating large test files..."
for i in $(seq 1 $NUM_FILES); do
    dd if=/dev/urandom of=$MOUNT_POINT/large_file_$i bs=1M count=1024
       2>/dev/null
done

echo "Testing read speed..."
for i in $(seq 1 $NUM_FILES); do
    echo "Reading large_file_$i..."
    time dd if=$MOUNT_POINT/large_file_$i of=/dev/null bs=1M
done

echo "Cleaning up..."
rm -f $MOUNT_POINT/large_file_*
```

Results for large files benchmark (read speed in MB/s):

Number of Servers	Read Speed (MB/s)
1	45.2
2	78.5
3	102.3
4	125.8

Table 2: Large files performance (1GB files)

Benchmark Analysis

Small Files Performance:

- With 1 server: 450 accesses/second
- Performance scales approximately linearly with number of servers
- With 4 servers: 1420 accesses/second (3.15× improvement)
- Distributed architecture allows parallel access to different files
- Overhead from network latency affects small file performance

Large Files Performance:

- With 1 server: 45.2 MB/s
- Read speed increases with more servers due to distributed reads
- With 4 servers: 125.8 MB/s (2.78× improvement)
- Large files benefit from parallel reading across multiple bricks
- Network bandwidth becomes a limiting factor with more servers

Performance Characteristics:

- Distributed volumes provide better performance for parallel workloads
- Replication adds redundancy but may slightly reduce write performance
- Small files show better scaling due to independent file access
- Large files benefit from distributed reads but are limited by network bandwidth

Additional GlusterFS Commands

Useful commands for volume management:

Listing 19: Volume management commands

```
gluster volume list
gluster volume stop gv0
gluster volume start gv0
gluster volume delete gv0
gluster volume set gv0 performance.cache-size 256MB
gluster volume set gv0 network.ping-timeout 10
gluster volume heal gv0 info
gluster volume rebalance gv0 start
```

Monitor volume performance:

Listing 20: Performance monitoring

```
gluster volume status gv0 detail
gluster volume top gv0 read
gluster volume top gv0 write
gluster volume top gv0 open
```

Group Work

Installation and Configuration: Team members worked together to install GlusterFS on all laptops. Each member was responsible for:

- Installing GlusterFS server on their laptop
- Configuring network and firewall rules
- Preparing storage bricks

Trusted Pool Creation: One team member (node1) initiated the trusted pool by probing other nodes. All members verified peer connectivity and troubleshooted network issues collaboratively.

Volume Creation and Testing:

- Team designed the volume layout (2×2 distributed replicated)
- Created and started the volume together
- Mounted the volume on all client machines

Benchmarking:

- Developed benchmark scripts for small and large files
- Performed tests with varying number of servers (1, 2, 3, 4)
- Collected and analyzed performance data
- Created performance tables and analysis

Report Writing: The LaTeX report was written collaboratively, with each member contributing sections on installation, configuration, benchmarking, and analysis. Code snippets and commands were verified by all team members.