Replication Fundamentals

7.1 Replication Architecture and Concepts

What is Replication?

Replication is the process of copying data from one MySQL server (Master) to one or more MySQL servers (Slaves) in real-time.

Benefits of Replication

1. **High Availability**: Failover to slave if master fails
2. **Scalability**: Distribute read queries to slaves
3. **Disaster Recovery**: Off-site replica for backup
4. **Analytics**: Run reports on slave without impacting production
5. **Testing**: Clone production environment for testing

Replication Process

```
Master Server
  ↓
Binary Log (stores all changes)
  ↓
  ├──→ Slave 1 (reads from binary log)
  ├──→ Slave 2
  └──→ Slave N
```

Replication Topology

Master-Slave (One-to-Many):

```
     Master
      /      |      \
    /        |        \
  Slave1   Slave2   Slave3
```

Master-Master (Multi-Master):

$$\begin{tikzcd}
\text{Master1} \arrow[r, leftrightarrow] \arrow[d] & \text{Master2} \arrow[d] \\
\text{Slave1} & \text{Slave2}
\end{tikzcd}$$

Cascading (Hierarchical):

$$\begin{tikzcd}
\text{Master} \arrow[d] & \\
\text{Slave1} \arrow[d] \arrow[dr] & \\
\text{Slave2} & \text{Slave3}
\end{tikzcd}$$

7.2 Setting Up Master-Slave Replication with GTID

GTID (Global Transaction ID)

GTID uniquely identifies each transaction in a replication topology:

GTID Format: server_uuid:transaction_id

Example: a1b2c3d4-e5f6-7890-abcd-ef1234567890:1

Advantages of GTID:

- Automatic failover to correct slave
- Crash-safe replication
- Simplifies slave setup (no need for binary log file/position)

Master Configuration

1. Configure Master Server

# Edit /etc/mysql/mysql.conf.d/mysqld.cnf

sudo vim /etc/mysql/mysql.conf.d/mysqld.cnf

Add:

```
[mysqld]
# Master Replication Configuration
server_id = 1
log_bin = /var/log/mysql/mysql-bin
binlog_format = ROW
binlog_expire_logs_seconds = 864000

# GTID Configuration
gtid_mode = ON
enforce_gtid_consistency = ON

# Other settings
max_binlog_size = 100M
```

2. Restart Master MySQL

```
sudo systemctl restart mysql
```

3. Verify Master Configuration

```
SHOW VARIABLES LIKE 'server_id';
SHOW VARIABLES LIKE 'gtid_mode';
SHOW VARIABLES LIKE 'enforce_gtid_consistency';
SHOW MASTER STATUS;
```

Create Replication User on Master

```
CREATE USER 'repl_user'@'192.168.1.101' IDENTIFIED BY 'ReplPass123!';
GRANT REPLICATION SLAVE ON *.* TO 'repl_user'@'192.168.1.101';
GRANT REPLICATION CLIENT ON *.* TO 'repl_user'@'192.168.1.101';
```

```
FLUSH PRIVILEGES;

-- Verify
SHOW GRANTS FOR 'repl_user'@'192.168.1.101';
```

Slave Configuration

1. Configure Slave Server

```
sudo vim /etc/mysql/mysql.conf.d/mysqld.cnf
```

Add:

```
[mysqld]
# Slave Replication Configuration
server_id = 2
relay-log = /var/log/mysql/mysql-relay-bin

# GTID Configuration
gtid_mode = ON
enforce_gtid_consistency = ON

# Read-only mode (recommended for slaves)
read_only = ON
super_read_only = ON

# Replication filters (optional)
replicate_do_db = production_db
```

2. Restart Slave MySQL

```
sudo systemctl restart mysql
```

Configure Slave to Connect to Master

```
CHANGE MASTER TO
    MASTER_HOST = '192.168.1.100',
    MASTER_USER = 'repl_user',
    MASTER_PASSWORD = 'ReplPass123!',
    MASTER_PORT = 3306,
    MASTER_AUTO_POSITION = 1;  -- GTID-based replication

-- For non-GTID replication (binary log position):
-- CHANGE MASTER TO
--     MASTER_HOST = '192.168.1.100',
--     MASTER_USER = 'repl_user',
--     MASTER_PASSWORD = 'ReplPass123!',
--     MASTER_LOG_FILE = 'mysql-bin.000001',
--     MASTER_LOG_POS = 154;
```

Start Replication

```
-- Start slave I/O and SQL threads
START SLAVE;

-- Alternative MySQL 8.0 syntax
START REPLICA;

-- Check replication status
SHOW SLAVE STATUS\G
```

-- Key fields to check:

-- Slave_IO_Running: Yes (reads from master)

-- Slave_SQL_Running: Yes (applies changes)

-- Seconds_Behind_Master: 0 (no lag)

-- Last_Errno: 0 (no errors)

Test Replication

On Master:

CREATE DATABASE test_replication;

USE test_replication;

CREATE TABLE test_table (id INT PRIMARY KEY, data VARCHAR(255));

INSERT INTO test_table VALUES (1, 'Test data 1');

INSERT INTO test_table VALUES (2, 'Test data 2');

SELECT * FROM test_table;

On Slave:

-- Wait a moment for replication

SELECT * FROM test_replication.test_table;

-- Should see the same data

7.3 Semi-Synchronous Replication

Concept

In asynchronous replication (default):

- Master writes data
- Master returns immediately to client
- Slave reads from binary log

Risk: If master crashes before slave reads, data is lost.

In semi-synchronous replication:

- Master writes data
- At least one slave acknowledges receipt
- Master returns to client
- Safer but slightly higher latency

Installation

On Master:

```
INSTALL PLUGIN rpl_semi_sync_master SONAME 'semisync_master.so';

SHOW PLUGINS LIKE 'rpl_semi_sync%';

SET GLOBAL rpl_semi_sync_master_enabled = ON;
SET GLOBAL rpl_semi_sync_master_timeout = 10000;  -- 10 seconds

SHOW VARIABLES LIKE 'rpl_semi_sync_master%';
```

On Slave:

```
INSTALL PLUGIN rpl_semi_sync_slave SONAME 'semisync_slave.so';

SET GLOBAL rpl_semi_sync_slave_enabled = ON;

SHOW VARIABLES LIKE 'rpl_semi_sync_slave%';
```

Restart Slave I/O Thread

```
-- On Slave
STOP SLAVE IO_THREAD;
```

```
START SLAVE IO_THREAD;

SHOW SLAVE STATUS\G
```

Monitor Semi-Synchronous Replication

```
-- On Master
SHOW STATUS LIKE 'Rpl_semi_sync_master%';
-- Rpl_semi_sync_master_yes_tx: Transactions acknowledged by slave
-- Rpl_semi_sync_master_no_tx: Transactions not acknowledged (timeout)

-- On Slave
SHOW STATUS LIKE 'Rpl_semi_sync_slave%';
```

7.4 Replication Monitoring and Troubleshooting

Monitor Replication Status

```
-- On Slave
SHOW SLAVE STATUS\G

-- Key fields:
-- Slave_IO_Running: Should be Yes
-- Slave_SQL_Running: Should be Yes
-- Seconds_Behind_Master: Should be 0 or low
-- Last_Errno: Should be 0
-- Last_Error: Should be empty
```

Common Replication Issues

Issue 1: Slave I/O Thread Stopped

```
-- On Slave
SHOW SLAVE STATUS\G
-- Slave_IO_Running: No

-- Possible causes:
-- 1. Network connectivity issue
-- 2. Wrong credentials
-- 3. Master not responding

-- Solution:
SHOW SLAVE STATUS\G  -- Check Last_IO_Error
-- Fix issue and restart
STOP SLAVE;
START SLAVE;
```

Issue 2: Replication Lag

```
-- On Slave
SHOW SLAVE STATUS\G
-- Seconds_Behind_Master: High number

-- Causes: Slow queries, heavy load, network lag

-- Solutions:
-- 1. Optimize slow queries
-- 2. Increase slave resources
-- 3. Use parallel replication (MySQL 5.7+)
```

Issue 3: Replication Error

```
-- On Slave
SHOW SLAVE STATUS\G
-- Last_Errno: Non-zero
-- Last_Error: Error message

-- Example: Duplicate key error on slave
-- Slave_SQL_Running: No
-- Last_Error: Duplicate entry '1' for key 'PRIMARY'

-- Solution options:
-- 1. Skip the error (if safe)
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1;
STOP SLAVE;
START SLAVE;

-- 2. Rebuild slave from backup
-- 3. Use pt-table-sync to sync data
```

Monitoring Tools

Percona Toolkit:

```
# Check table consistency
pt-table-checksum \
  --host=master \
  --user=repl_user \
  --password=ReplPass123!
```

```
# Sync differences
pt-table-sync --execute \
  h=master,u=repl_user,p=ReplPass123! \
  h=slave,u=repl_user,p=ReplPass123!

# Check slave status
pt-slave-restart \
  --host=slave \
  --user=repl_user \
  --password=ReplPass123!
```

Performance Schema:

```
-- View replication configuration
SELECT * FROM performance_schema.replication_connection_configuration\G

-- View connection status
SELECT * FROM performance_schema.replication_connection_status\G

-- View applier status
SELECT * FROM performance_schema.replication_applier_status\G
```

7.5 Summary: Key Takeaways

1. **Replication Basics**: Master writes data, slaves read from binary log
2. **GTID**: Uniquely identifies transactions, simplifies replication
3. **Master Setup**: server_id, log_bin, gtid_mode, create replication user
4. **Slave Setup**: server_id (unique), CHANGE MASTER TO, START SLAVE
5. **Semi-Synchronous**: Master waits for slave acknowledgment (safer but slower)

6. **Monitoring**: SHOW SLAVE STATUS, Seconds_Behind_Master, Last_Error
7. **Troubleshooting**: Check network, credentials, replication errors, lag