PERCONA

MySQL Replication Best Practices

Francisco Bordenave
MySQL Consultant
May 13th, 2015

Agenda

- Replication in a nutshell.
- Setting up variables and replication.
- Monitoring replication.
- Checking data consistency.
- Synchronizing data.
- GTID overview.
- GTID online implementation.

Replication in a nutshell

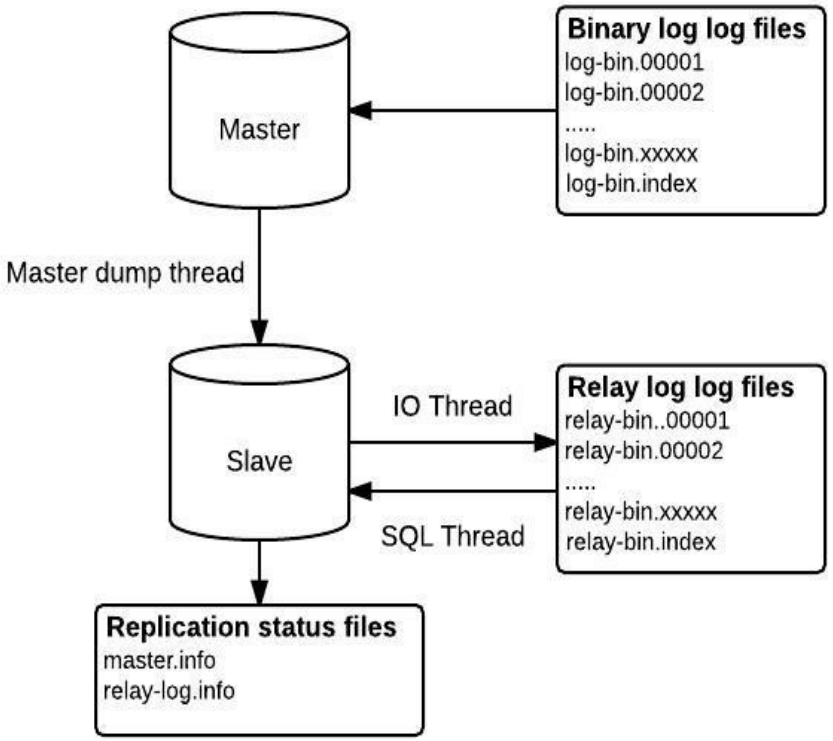
Replication in a nutshell

- Replication components
- Binary log types
- Replication threads

Replication in a nutshell

Replication components

- Master server
  - Binlog dump thread(s)
- Slave(s) server(s)
  - Replication threads
- Binary logs
- Relay logs
- Information files



Replication in a nutshell

Binary log types

Statement format

```
# at 237
#150416 18:30:21 server id 1  end_log_pos 316 CRC32 0x54b3fd07      Query   thread_id=3    exec_time=0    error_code=0
SET TIMESTAMP=1429209021/*!*/;
BEGIN
/*!*/;
# at 316
#150416 18:30:21 server id 1  end_log_pos 440 CRC32 0x23bfcad7      Query   thread_id=3    exec_time=0    error_code=0
SET TIMESTAMP=1429209021/*!*/;
insert into t1 (f1,f2) values(17650, 'hohdeal3lo')
/*!*/;
# at 440
#150416 18:30:21 server id 1  end_log_pos 471 CRC32 0x569c6fe3      Xid = 19
COMMIT/*!*/;
DELIMITER ;
```

Row format

```
# at 311
#130419 16:27:17 server id 1  end_log_pos 385 CRC32 0x572186a8      Rows_query
# insert into t1 (f1,f2) values(17650, 'hohdeal3lo') <- 5.6 only - binlog_rows_query_log_events needs to be ON
# at 385
#130419 16:27:17 server id 1  end_log_pos 433 CRC32 0x538010c4      Table_map: `test`.`t1` mapped to number 70
# at 433
#130419 16:27:17 server id 1  end_log_pos 484 CRC32 0x44f810d4      Write_rows: table id 70 flags: STMT_END_F
BINLOG '
tbZxUR0BAAAASgAAAIEBAACAADJpbnNlcnQgaW50byB0MSAoZjEsZjIpIHZhbHVlcygxNzY1MCwg
J2hvaGRlYWwzbG8nKaiGIVc=
tbZxURMBAAAAMAAAALEBAAAAAEYAAAAAAAAEABHRlc3QAAnQxAAIDDwItAADEEIBT
tbZxUR4BAAAAMwAAAOQBAAAAAEYAAAAAAAEAAgAC//zyRAAACmhvaGRlYWwzbG/UEPhE
'/*!*/;
### INSERT INTO `test`.`t1`
### SET
###   @1=17650 /* INT meta=0 nullable=0 is_null=0 */
###   @2='hohdeal3lo' /* VARSTRING(45) meta=45 nullable=0 is_null=0 */
# at 484
```

Replication in a nutshell

Replication threads

Master threads

- One thread per slave connected that reads and send binlog events.

Slave threads

- **IO thread**
  - connects to master and get binlog contents.
  - dump contents into relay logs.
  - updates read master position to get status of IO events.
- **SQL thread**
  - reads relay log events.
  - execute events (either via statement or row updates)
  - update execute master position to get status of latest SQL executed.
  - with parallel replication you can have many SQL appliers.

Replication variables

Replication variables

Master

- `server_id` an integer value that identifies each server, it needs to be unique in all servers
- `log_bin` if present in my.cnf file enables the binary logging which is necessary for replication.

Is this enough? Well yes but we may want to have some extra variables set.

- `binlog_format` defines which format we want to use: STATEMENT, MIXED or ROW
- `log_bin=basename` to setup different names to binary log files instead of default host_name.bin
- `binlog_rows_query_log_events` enables extra verbosity in ROW format, stores queries as comments.

Controlling what is stored and what is not in binlogs

- `binlog-do-db=db_name` only if we want to store and replicate certain Dbs
- `binlog-ignore-db=db_name` used when we don't want to replicate certain DBs

Replication variables

Slave

- `server_id` this is mandatory in all servers within a replication environment.
- `log_bin` in a slave this is useful to allow a replication chain.

Some extra variables we can configure

- `binlog_format`
- `log_bin=basename`
- `log-slave-updates` instruct slave to log events ran by SQL thread.
- `relay_log=file_name` to name relay log files
- `expire_log_days` amount of days for binary log retention

Filtering binlogs

- `replicate-do-db=db_name` only if we want to store and replicate certain DBs
- `replicate-ignore-db=db_name` used when we don't want to replicate certain DBs
- `replicate_wild_do_table` used to replicate tables based on wildcard patterns
- `replicate_wild_ignore_table` used to ignore tables in replication based on wildcard patterns

Replication variables

Best practices

Enable binary log on slaves

- `log-bin`
- `log-slave-updates` makes MySQL log events ran by SQL Thread (replication thread)
- `expire_log_days` you don't want to keep binlogs in your master forever.

Create crash-safe slaves

- `master-info-repository=TABLE` store information about master binary log files and position into table.
- `relay-log-info-repository=TABLE` store information about relay log files and position into table.
- `relay-log-recovery=1` enables automatic recovery of relay log files after crash

Filter replication at slave side

- Use `replicate-ignore-db` and `replicate_wild_ignore_table` to filter data instead of `binlog-ignore-db` in master.

Be always on the safe side

- `read-only` avoid users to write in slaves (only users with SUPER privileges can write).
- `skip-slave-start` avoid replication to start after a crash or server restart.

Replication Setup I

Configuration files

Master my.cnf

```
[mysqld]
server_id=1
log_bin
binlog_format=ROW
expire_log_days=7
```

Slave my.cnf

```
[mysqld]
server_id=2
read-only
skip-slave-start
log_bin
log-slave-updates
binlog_format=ROW
master-info-repository=TABLE
relay-log-info-repository=TABLE
relay-log-recovery=1
```

Replication Setup II

Create replication user on master

```
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'slave_IP' identified by 'password';
```

Create a backup from master

Using mysqldump:

```
mysqldump -uroot -p --master-info=2 > backup.sql
```
this stores binary log position to dump file.

Using xtrabackup:

```
innobackupex --user=root --password=PootPass /path/to/dest/
innobackupex --apply-log /path/to/dest/
```

Restore backup on slave and get replication information:

Using mysqldump:

```
mysql -uroot -p < backup.sql
head -25 backup.sql |grep "CHANGE MASTER"
   -- CHANGE MASTER TO MASTER_LOG_FILE='mysqld-bin.000001', MASTER_LOG_POS=120;
```

Using xtrabackup:

```
/etc/init.d/mysql stop
rm -rf /var/lib/mysql/*
cd /var/lib/mysql
mv /backup/* .
chown -R mysql:mysql /var/lib/mysql
/etc/init.d/mysql start
cat /var/lib/mysql/xtrabackup_binlog_info
   MASTER_LOG_FILE='mysqld-bin.000021', MASTER_LOG_POS=120;
```

Replication Setup III

Configure slave to start replication

```
CHANGE MASTER TO MASTER_HOST='master_IP', MASTER_USER='repl', MASTER_PASSWORD='password',
MASTER_LOG_FILE='mysqld-bin.000021', MASTER_LOG_POS=120;
START SLAVE;
```

Check replication is working properly

```
SHOW SLAVE STATUS\G
 ...
 Slave_IO_Running: Yes
 Slave_SQL_Running: Yes
 ...
 Seconds_Behind_Master: 13
 ...
```

Congratulations, you have setup a slave properly!

PERCONA

Replication Monitoring

Replication Monitoring

What to monitor?

Slave threads are running and replication lag.

Basic manual check

```
SHOW SLAVE STATUS\G
 ...
 Slave_IO_Running: Yes
 Slave_SQL_Running: Yes
 ...
 Seconds_Behind_Master: 13
 ...
```

Advanced manual check using pt-hearbeat

In master:

```
pt-heartbeat -D test --update -h master-server --daemonize
```

In Slave:

```
pt-heartbeat -D test --monitor -h slave-server
```

Gazillions of free monitoring tools

- Nagios
- Cacti
- Munin
- etc

Checking data consistency

Checking data consistency

How to check if slave has consistent data?
pt-table-checksum can do the work for you

```
pt-table-checksum --replicate=test.cksums --create-replicate-table
       TS ERRORS         DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
     04-30T11:31:50      0      0   633135       8       0   5.400 sakila.actor
     04-30T11:31:52      0      0   290859       1       0   2.692 sakila.film
     Checksumming sakila.film:  16% 02:27 remain
     Checksumming sakila.film:  55% 01:58 remain
     Checksumming sakila.film:  86% 00:24 remain
     04-30T11:34:38      2      0 22187768     126       0 165.216 sakila.film
...
     04-30T11:38:09      0      0        0       1       0   0.033 mysql.time_zone_name
     04-30T11:38:09      0      0        0       1       0   0.052 mysql.time_zone_transition
     04-30T11:38:09      0      0        0       1       0   0.054 mysql.time_zone_transition_type
     04-30T11:38:09      0      0        8       1       0   0.064 mysql.user
...
```

Too long output? Don't worry use `--replicate-check-only` flag

```
pt-table-checksum --replicate-check-only --replicate=test.cksums
Differences on h=127.0.0.1
TABLE CHUNK CNT_DIFF CRC_DIFF CHUNK_INDEX LOWER_BOUNDARY UPPER_BOUNDARY
sakila.film 1 0 1 PRIMARY 1 100
```

PERCONA

Syncing data

Syncing data

Using pt-table-sync to re-sync data

Use `--print` and `--verbose` flags because we want to make sure what tool is going to do:

```
perl pt-table-sync --print --verbose --replicate=test.cksums --sync-to-master slave_server
# Syncing via replication h=slave_server
# DELETE REPLACE INSERT UPDATE ALGORITHM START    END      EXIT DATABASE.TABLE
REPLACE INTO `sakila`.`film`(`film_id`, `title`, `description`, `release_year`, `language_id`,
`original_language_id`, `rental_duration`, `rental_rate`, `length`, `replacement_cost`, `rating`,
`special_features`, `last_update`) VALUES ('1', 'ACADEMY DINOSAUR', 'A Epic Drama of a Feminist And a Mad
Scientist who must Battle a Teacher in The Canadian Rockies', '2006', '1', NULL, '6', '0.99', '86', '20.99',
'PG', 'Deleted Scenes,Behind the Scenes', '2006-02-15 05:03:42') /*percona-toolkit src_db:sakila src_tbl:film
src_dsn:P=3306,h=192.168.70.2 dst_db:sakila dst_tbl:film dst_dsn:h=192.168.70.3 lock:1 transaction:1
changing_src:test.cksums replicate:test.cksums bidirectional:0 pid:10109 user:root
host:localhost.localdomain*/;
REPLACE INTO `sakila`.`film`(`film_id`, `title`, `description`, `release_year`, `language_id`,
`original_language_id`, `rental_duration`, `rental_rate`, `length`, `replacement_cost`, `rating`,
`special_features`, `last_update`) VALUES ('2', 'ACE GOLDFINGER', 'A Astounding Epistle of a Database
Administrator And a Explorer who must Find a Car in Ancient China', '2006', '1', NULL, '3', '4.99', '48',
'12.99', 'G', 'Trailers,Deleted Scenes', '2006-02-15 05:03:42') /*percona-toolkit src_db:sakila src_tbl:film
src_dsn:P=3306,h=192.168.70.2 dst_db:sakila dst_tbl:film dst_dsn:h=192.168.70.3 lock:1 transaction:1
changing_src:test.cksums replicate:test.cksums bidirectional:0 pid:10109 user:root
host:localhost.localdomain*/;
...
#      0      10      0      0 Chunk     19:58:50 19:58:50 2    sakila.film
```

All good, we will run some `REPLACE` commands so we are good to go with `--execute` flag

```
pt-table-sync --execute --verbose --replicate=test.cksums --sync-to-master 192.168.70.3
# Syncing via replication h=192.168.70.3
# DELETE REPLACE INSERT UPDATE ALGORITHM START    END      EXIT DATABASE.TABLE
#      0      10      0      0 Chunk     20:08:23 20:08:23 2    sakila.film
```

PERCONA

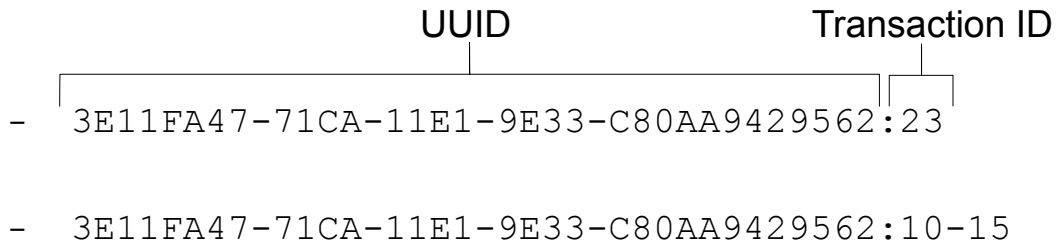
GTID Overview

GTID Overview

- What's GTID
- How to deploy it?

GTID Overview

- What's GTID
  - Global Transaction Identifier
    - Transaction has unique identifier in all servers
  - Server UUID:Transaction ID



  - Ensures transaction belongs to only one server
  - Ensures transactions are applied only once in replication stream

GTID Overview

Deployment

- `enforce_gtid_consistency` force MySQL to allow only transactional safe statements.
- `gtid_mode=ON` enable GTID

Is this enough? No, GTID can't be deployed in a running environment in MySQL community version. So there is a procedure to deploy GTID which needs downtime:

1.Stop applications, sync data and stop replication:

```
STOP SLAVE;
```

2.Stop both servers:

```
/etc/init.d/mysql stop
```

3.Restart both servers with described variables previously:

In my.cnf

```
skip-slave-start
gtid_mode=ON
enforce-gtid-consistency
log-bin
log-slave-updates
```

4.Reset slave to use GTID based positions:

```
CHANGE MASTER TO MASTER_HOST='master_IP', MASTER_USER='repl', MASTER_PASSWORD='password',
MASTER_AUTO_POSITION=1;
START SLAVE;
```

5.Resume applications.

GTID Online Implementation

GTID Online Implementation

- What's needed?
- How to deploy it?

GTID Online Implementation

- What's needed?
  - Percona Server > 5.6.22 (it will be available in MySQL 5.7 community)
  - Setup of GTID variables:

```
skip-slave-start
gtid_mode=ON
enforce-gtid-consistency
log-bin

log-slave-updates
gtid_deployment_step=ON
```

  - Slave promotion to master.
  - Ability to failover applications.

GTID Online Implementation

- How to deploy it?

  On slave:

1. Stop replication:
   ```
   STOP SLAVE;
   ```
2. Modify my.cnf and restart server:
   ```
   skip-slave-start
   gtid_mode=ON
   enforce-gtid-consistency
   log-bin

   log-slave-updates
   ```
3. Change following variable and start replication:
   ```
   set global gtid_deployment_step=ON;
   start slave;
   ```
4. Configure replication user to allow connections from old-master:
   ```
   GRANT REPLICATION SLAVE ON *.* TO 'repl'@'master_IP' identified by 'password';
   ```
5. Once replication is up to date, change applications to point to server.

GTID Online Implementation

- How to deploy it?

  On old-master:

1. Modify my.cnf and restart server:

```
skip-slave-start
gtid_mode=ON
enforce-gtid-consistency
log-bin

log-slave-updates
```

2. Change following variable:

```
set global gtid_deployment_step=ON;
```

3. Configure server to replicate from promoted master using GTID:

```
change master to master_host='slave_ip', master_user='rep', master_password='password',
master_auto_position=1;
start slave;
```

4. In both servers disable gtid_deployment_step variable

```
set global gtid_deployment_step=OFF;
```

PERCONA

Questions?