MySQL Replication

Aaron Brown

https://github.com/9minutesnooze/mysql_replication_talk

MySQL Replication

replicates **changes** between servers

Replication is

asynchronous by default

single threaded in execution

subscriber based

How Replication Works

Master

1. Transaction is committed by client
2. MySQL writes SQL (SBR) or rows (RBR) to the **binary log**

Binary Log

1. **Serialized** representation of all write operations executed by server
2. Includes metadata for some non-deterministic queries (e.g. NOW(), RANDOM(), auto-increment values)

Binary Log

1. STATEMENT based replication (SBR) logs the SQL statement executed
2. ROW based replication (RBR) logs the changed rows
3. MIXED logs switches between SBR and RBR as necessary
4. Set with the binlog_format variable

Binary Log

Statement Based Replication (SBR)

```
 $ mysqlbinlog mysql-bin.000001
...
# at 21603
#130308  2:16:19 server id 1  end_log_pos 21672 Query thread_id=40 exec_time=0 error_code=0
SET TIMESTAMP=1362708979/*!*/;
BEGIN
/*!*/;
# at 21672
#130308  2:16:19 server id 1  end_log_pos 21700     Intvar
SET INSERT_ID=86/*!*/;
# at 21700
#130308  2:16:19 server id 1 end_log_pos 21831  Query thread_id=40 exec_time=0 error_code=0
SET TIMESTAMP=1362708979/*!*/;
INSERT INTO phrases (phrase) VALUES ('unflavored sustaining_pedal')
/*!*/;
# at 21831
#130308  2:16:19 server id 1  end_log_pos 21858     Xid = 198
COMMIT/*!*/;
# at 21858
...
```

Binary Log

Row Based Replication (RBR)

```
 $ mysqlbinlog -v mysql-bin.000001
# at 6872
#130318 13:31:26 server id 1  end_log_pos 6941 Query thread_id=48   exec_time=0 error_code=0
SET TIMESTAMP=1363613486/*!*/;
BEGIN
/*!*/;
# at 6941
# at 6991
#130318 13:31:26 server id 1  end_log_pos 6991      Table_map: `stuff`.`phrases` mapped to
number 43
#130318 13:31:26 server id 1  end_log_pos 7042      Write_rows: table id 43 flags: STMT_END_F
BINLOG '
LhdHURMBAAAAMgAAAE8bAAAAACsAAAAAAAEABXN0dWZmAAdwaHJhc2VzAAIDDwL9AgI=
LhdHURcBAAAAMwAAAIIbAAAAACsAAAAAAAEAAv/8Qp4AAA8Ac29saWQgcmFkaWF0aW9u
'/*!*/;
### INSERT INTO stuff.phrases
### SET
###   @1=40514
###   @2='solid radiation'
# at 7042
#130318 13:31:26 server id 1  end_log_pos 7069      Xid = 164
COMMIT/*!*/;
# at 7069
```

Statement Based Replication

1. Can cause inconsistencies for certain types of queries
   - UUID(), RAND(), USER(), ...
   - UPDATE/DELETE ... LIMIT without ORDER BY
2. Requires additional locks
3. Easy to read
4. CPU & I/O intensive on the slave

Row Based Replication

1. Safest - all changes can be replicated
2. Difficult to read the binary log
3. Larger binary logs: more disk & bandwidth
4. Fewer locks on both master and slave
5. Better for concurrency

How Replication Works

Slave

1. Two threads Involved
    1. I/O Thread
    2. SQL Thread

How Replication Works

Slave I/O Thread

- Connects to master as a REPLICATION SLAVE
- Reads binary log from master at the requested position (offset)
- Writes binary log entries from master into local **relay log**

How Replication Works

Slave SQL Thread

- Reads local **relay log** and executes SQL (SBR) or applies row changes (RBR)

How Replication Works

Relay Log

```
 $ mysqlbinlog mysql-relay.003023
...
# at 1397
#130312 6:50:29 server id 62751 end_log_pos 1316 Query thread_id=17276602 exec_time=0 error_code=0
SET TIMESTAMP=1363085429/*!*/;
BEGIN
/*!*/;
# at 1461
#130312 6:50:29 server id 62751 end_log_pos 1555 Query thread_id=17276602 exec_time=0 error_code=0
SET TIMESTAMP=1363085429/*!*/;
UPDATE phrases SET phrase = ‘foo bar’ WHERE `id` = 1149
/*!*/;
# at 1700
#130312  6:50:29 server id 62751  end_log_pos 1582      Xid = 441138693
COMMIT/*!*/;
...
```

How Replication Works

Slave

1. Slave I/O Thread reads master log, writes relay log
2. Slave SQL Thread reads relay log, executes/applies changes

SHOW SLAVE STATUS

Lag

```
       Slave_IO_State: Waiting for master to send event
          Master_Host: db1.example.com
....
      Master_Log_File: mysql-bin-log.002831       # I/O Thread master log file
  Read_Master_Log_Pos: 704483683                  # I/O Thread position in master log
       Relay_Log_File: mysql-relay-log.011455     # I/O Thread writing to this relay log
        Relay_Log_Pos: 180195940                  # I/O Thread writing at this relay log position
Relay_Master_Log_File: mysql-bin-log.002831       # SQL Thread caught up to this master file
     Slave_IO_Running: Yes                        # Is the slave I/O thread running?
    Slave_SQL_Running: Yes                        # Is the slave SQL thread running?
....
           Last_Errno: 0
           Last_Error:
         Skip_Counter: 0
  Exec_Master_Log_Pos: 704483683                  # SQL thread caught up to master log position
      Relay_Log_Space: 704484093                  # Disk space used by relay logs
      Until_Condition: None
       Until_Log_File:
        Until_Log_Pos: 0
...
Seconds_Behind_Master: 0                          # lag (in seconds) between SQL and IO threads
        Last_IO_Errno: 0
        Last_IO_Error:
       Last_SQL_Errno: 0
       Last_SQL_Error:
```

Looking at Slave Threads

```
mysql> SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST WHERE User = 'system user'\G
*************************** 1. row ***************************
     ID: 955
   USER: system user
   HOST: 
     DB: NULL
COMMAND: Connect
   TIME: 0
  STATE: Has read all relay log; waiting for the slave I/O thread to update it
   INFO: NULL
TIME_MS: 11
*************************** 2. row ***************************
     ID: 954
   USER: system user
   HOST: 
     DB: NULL
COMMAND: Connect
   TIME: 3833897
  STATE: Waiting for master to send event
   INFO: NULL
TIME_MS: 3833897211
```

Looking at Slave Threads

```
mysql> pager grep -e _IO_ -e _SQL_

mysql> SHOW SLAVE STATUS\G
               Slave_IO_State: Waiting for master to send event
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:

mysql> nopager
```

Controlling Slave Threads

```
mysql> pager grep -e Running
mysql> SHOW SLAVE STATUS\G
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes

mysql> STOP SLAVE IO_THREAD;
mysql> SHOW SLAVE STATUS\G
             Slave_IO_Running: No
            Slave_SQL_Running: Yes

mysql> STOP SLAVE SQL_THREAD;
mysql> SHOW SLAVE STATUS\G
             Slave_IO_Running: No
            Slave_SQL_Running: No

mysql> START SLAVE;
mysql> SHOW SLAVE STATUS\G
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes

mysql> nopager
```

SQL Thread Gotchas

1. SQL thread acts just like a normal MySQL session
2. Stopping the SQL thread destroys explicit temporary tables (CREATE TEMPORARY TABLE)

Safely Stop Slave

1. Stop the I/O Thread

```
slave> STOP SLAVE IO_THREAD;
```

2. Watch SHOW SLAVE STATUS & wait until Read_Master_Log_Pos == Exec_Master_Log_Pos

```
slave> SHOW SLAVE STATUS\G
              Master_Log_File: mysql-bin.000002
          Read_Master_Log_Pos: 2264157
        Relay_Master_Log_File: mysql-bin.000002
          Exec_Master_Log_Pos: 2264157
```

Safely Stop Slave

3. If Slave_open_temp_tables > 0 start the I/O thread for a bit, then repeat until Slave_open_temp_tables == 0

```
slave> SHOW GLOBAL VARIABLES LIKE ‘Slave_open_temp_tables’;
| Slave_open_temp_tables | 0      |
```

4. Stop the slave SQL Thread

```
slave> STOP SLAVE SQL_THREAD;
```

Setting Up Replication

Master

1. Set up /etc/mysql/my.cnf on master and slave
2. Create user w/ REPLICATION SLAVE privilege on master
3. Create **consistent** backup from master and record binary log positions

Setting Up Replication

Slave

4. Restore backup on slave
5. Execute CHANGE MASTER command on slave
6. START SLAVE on slave

Setting Up Replication

Master

/etc/mysql/my.cnf

```
server-id = 1                # must be unique
log_bin = mysql-bin
relay_log = mysql-relay
log_slave_updates
expire_logs_days = 10
binlog_format = MIXED
```

required for master

good to specify

Setting Up Replication

Slave

```
/etc/mysql/my.cnf

server-id = 2                 # must be unique
log_bin = mysql-bin
relay_log = mysql-relay
log_slave_updates
expire_logs_days = 10
binlog_format = MIXED
```

required for slave

good to specify

Replication User

Master

```
mysql1> GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%'
IDENTIFIED BY ‘replication’;
```

Consistent Backup

1. mysqldump
2. Percona Xtrabackup
3. Filesystem Snapshots

mysqldump

1. Only good for small datasets
2. Creates a logical backup of SQL statements
3. Need to run in a transaction (InnoDB) or lock tables (MyISAM) to make it consistent
4. Consistent as of mysqldump **start** time

mysqldump

```
$ mysqldump --all-databases --single-transaction \
            --skip-lock-tables --master-data=2 \
            -u root -p > dump.sql

$ grep 'CHANGE MASTER' dump.sql
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.
000001', MASTER_LOG_POS=23110;
```

Percona Xtrabackup

1. Runs a separate read-only copy of InnoDB
2. Creates a binary backup that can be copied into the MySQL data directory
3. Consistent as of backup **completion** time
4. Can perform incremental backups

Percona Xtrabackup

```
mysql1> GRANT SELECT, RELOAD, LOCK TABLES,
REPLICATION CLIENT ON *.* TO 'backup'@'localhost'
IDENTIFIED BY ‘password’;

# from Percona repository
$ apt-get install xtrabackup

# take a backup
$ innobackupex --user=$USER --password=$PASSWORD
backupdir

# prepare it for use
$ innobackupex --apply-log --redo-only backupdir/
2013-03-08_20-21-18
```

Percona Xtrabackup

```
$ ls backupdir/2013-03-08_20-21-18
backup-my.cnf
ibdata1
mysql/
performance_schema/
stuff/
xtrabackup_binary
xtrabackup_binlog_info
xtrabackup_binlog_pos_innodb
xtrabackup_checkpoints
xtrabackup_logfile

$ cat xtrabackup_binlog_pos_innodb
./mysql-bin.000001 23110
```

Filesystem Snapshot

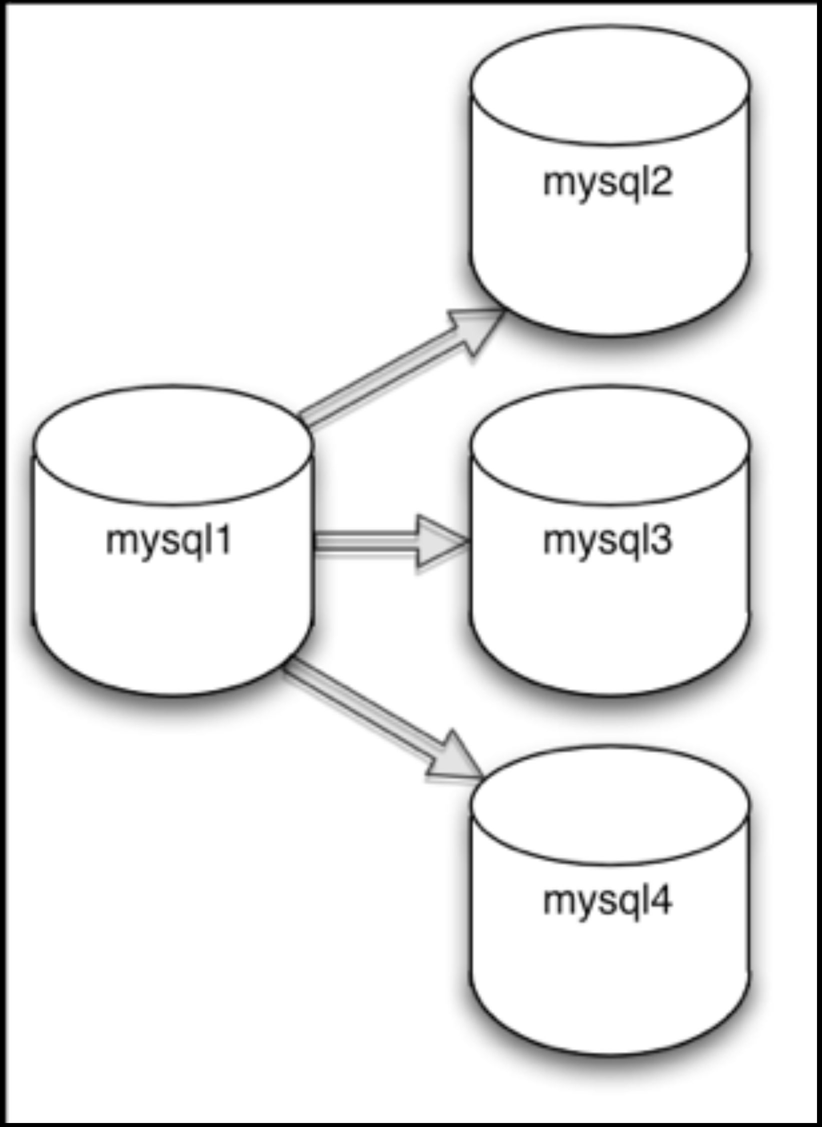
1. LVM or EBS snapshots
2. EBS snapshots: ec2-consistent-snapshot
3. Requires a brief lock on all tables w/ FLUSH TABLES WITH READ LOCK
4. Best with XFS filesystems: xfs_freeze

Setting Up Replication

Master/Slave

1. Restore backup of master **onto slave**

```
mysql2$ mysql -u root -p
< dump.sql
```



Setting Up Replication

Master/Slave

2. Use CHANGE MASTER to point to the correct master log file and log position

```
mysql2$ grep CHANGE dump.sql
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.
000002', MASTER_LOG_POS=149279;

mysql2> CHANGE MASTER TO MASTER_HOST='mysql1',
MASTER_USER='repl', MASTER_PASSWORD='replication',
MASTER_LOG_FILE='mysql-bin.000002',
MASTER_LOG_POS=149279;
```

Setting Up Replication

Master/Slave

3. Start the I/O Thread

```
    mysql> START SLAVE IO_THREAD;
    mysql> SHOW SLAVE STATUS\G
                Slave_IO_Running: Yes
             Exec_Master_Log_Pos: 149279   # Static until SQL Thread starts
                 Relay_Log_Space: 220021
                   Last_IO_Errno: 0
                   Last_IO_Error:
```

4. Start the SQL Thread

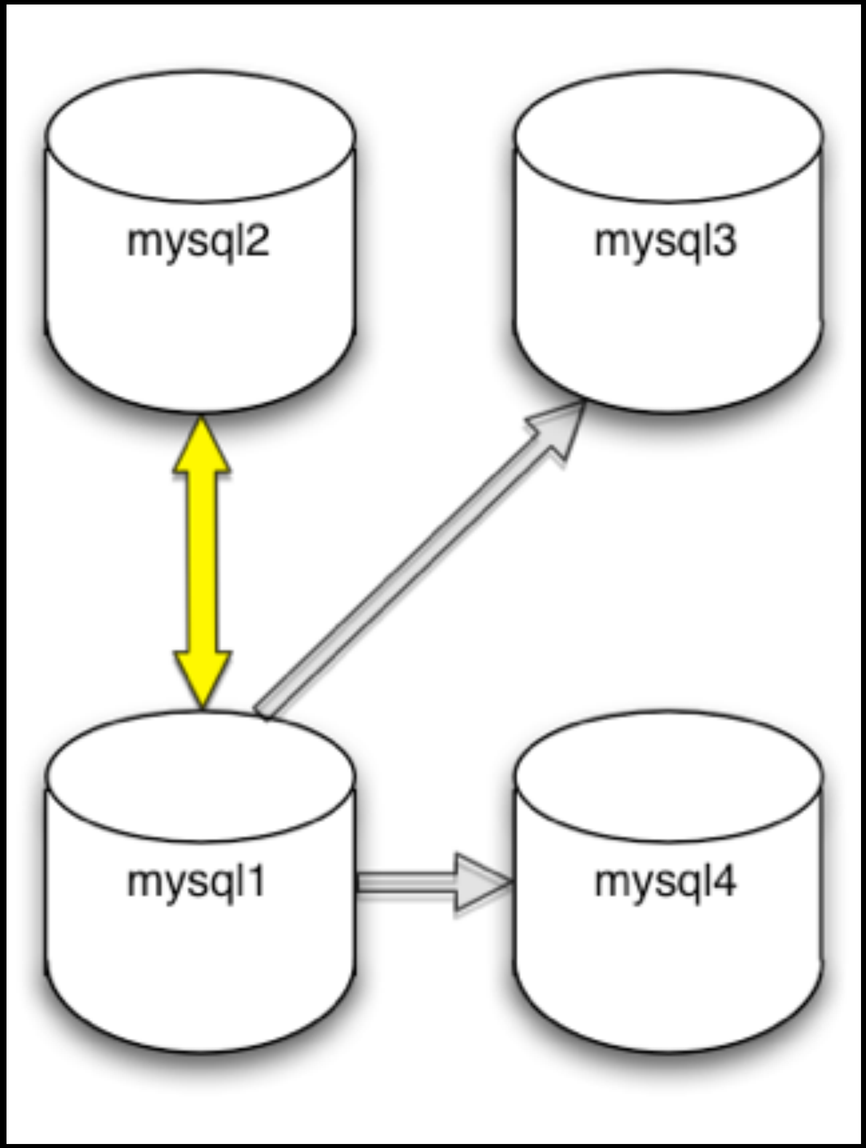
```
mysql> START SLAVE SQL_THREAD;
    mysql> SHOW SLAVE STATUS\G
               Slave_SQL_Running: Yes
             Exec_Master_Log_Pos: 486298   # Starts moving
                 Relay_Log_Space: 220021
                  Last_SQL_Errno: 0
                  Last_SQL_Error:
```

Setting Up Replication

Master/Master

1. Master/Master replication a variant of Master/Slave

2. Each master is a slave of the other master



Setting Up Replication

Master/Master

1. Stop the SQL thread on the slave

```
mysql2> STOP SLAVE SQL_THREAD\G
```

2. Get the slave's binary log coordinates

```
mysql2> SHOW MASTER STATUS\G
              File: mysql-bin.000002
          Position: 882394
```

Setting Up Replication

Master/Master

3. Execute CHANGE MASTER on the first master (mysql1)

```
mysql1> CHANGE MASTER TO MASTER_HOST=‘mysql2’,
MASTER_USER=‘repl’, MASTER_PASSWORD=‘replication’,
MASTER_LOG_FILE=’mysql-bin.000002’, MASTER_LOG_POS=
882394;
```

4. Start the slave on both mysql1 and mysql2

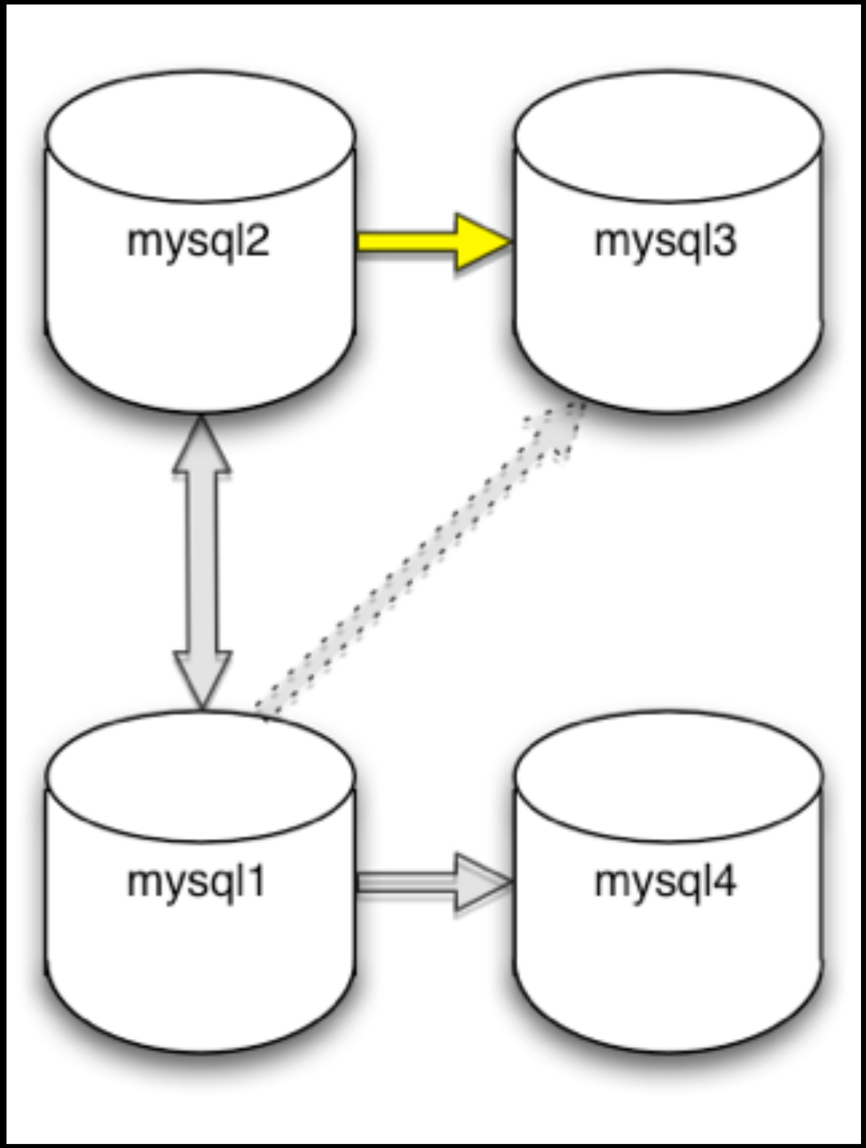
```
mysql1> START SLAVE;

mysql2> START SLAVE;
```

Move a Slave

1. Goal: Move mysql3 to be a slave of mysql2.

2. Need to get mysql2 and mysql3 stopped at the same position relative to mysql1



Move a Slave

1. Stop the slave on both mysql2 (new parent) and mysql3*

```
mysql2> STOP SLAVE;
mysql3> STOP SLAVE;
```

* Actually, use the safe stop slave procedure from earlier instead of this.

Move a Slave

2. Let server with the earlier log position catch up to the other

```
mysql2> SHOW SLAVE STATUS\G
        Relay_Master_Log_File: mysql-bin.000002
          Exec_Master_Log_Pos: 2268287
mysql3> SHOW SLAVE STATUS\G
        Relay_Master_Log_File: mysql-bin.000002
          Exec_Master_Log_Pos: 2271176

mysql2> START SLAVE UNTIL MASTER_LOG_FILE='mysql-bin.000002',
MASTER_LOG_POS=2271176;

mysql2> SHOW SLAVE STATUS\G
        Relay_Master_Log_File: mysql-bin.000002
          Exec_Master_Log_Pos: 2271176
```

Move a Slave

3. Get the master log position of the new parent - mysql2

```
mysql2> SHOW MASTER STATUS\G
            File: mysql-bin.000002
        Position: 2071838
```

4. Change mysql3 to use mysql2's binlog positions

```
mysql3> CHANGE MASTER TO MASTER_HOST='mysql2',
MASTER_LOG_FILE='mysql-bin.000002',
MASTER_LOG_POS=2071838;
```

Move a Slave

5. Start the slaves

```
mysql3> START SLAVE;
mysql2> START SLAVE;
```

Slave Lag

Seconds_Behind_Master

1. Difference between the timestamps of the query last **read** by the IO thread and the query last **executed** by the SQL thread.
2. Not very accurate
3. Represents only one level in tiered replication

Slave Lag

pt-heartbeat

1. Inserts/updates a row in a table on the ultimate master with timestamp
2. Accurate representation of slave lag across entire replication tree

Slave Lag
pt-heartbeat

```
user@mysql1$ pt-heartbeat --create-table --update --user root --
password root --database percona --table heartbeat

mysql2> SELECT * FROM percona.heartbeat\G
*************************** 1. row ***************************
                   ts: 2013-03-15T19:17:28.001870
            server_id: 1
                 file: mysql-bin.000002
             position: 2910464
relay_master_log_file: mysql-bin.000002
  exec_master_log_pos: 2680108

mysql2> SELECT (UNIX_TIMESTAMP(UTC_TIMESTAMP()) - UNIX_TIMESTAMP(ts))
as secs_behind FROM percona.heartbeat;
+-------------+
| secs_behind |
+-------------+
|          72 |
+-------------+
```

Master/Slave Sync

1. Replication is **asynchronous**
2. No guarantees that a change will execute the same on both master and slave
3. master and slave can get out of sync

pt-table-checksums

1. Performs checksum calculation on chunks of rows
2. Populates a table with checksum results
3. Each table must have a unique key

pt-table-checksums

```
# intentionally mess up the slave
mysql4> UPDATE phrases SET phrase = 'whoops' WHERE phrase LIKE 'a%';
Query OK, 1562 rows affected (0.03 sec)
Rows matched: 1562  Changed: 1562  Warnings: 0

user@mysql1$ pt-table-checksum --replicate percona.checksums --databases
stuff --user root --pass root --no-check-binlog-format
            TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
03-15T19:37:52      0      2    16672       4       0   0.311 stuff.phrases

mysql4> SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks
FROM percona.checksums WHERE (master_cnt <> this_cnt OR master_crc <>
this_crc  OR ISNULL(master_crc) <> ISNULL(this_crc)) GROUP BY db, tbl;
+-------+---------+------------+--------+
| db    | tbl     | total_rows | chunks |
+-------+---------+------------+--------+
| stuff | phrases |      17176 |      2 |
+-------+---------+------------+--------+
```

pt-table-sync

1. Examines output of pt-table-checksum and generates SQL to fix master/slave sync issues

pt-table-sync

Find out of sync rows

```
user@mysql4$ pt-table-sync --print --sync-to-master
--replicate percona.checksums --wait 0 u=root,p=root
```

```
...
REPLACE INTO `stuff`.`phrases`(`id`, `phrase`) VALUES ('17', 'addlepated
montego_bay') /*percona-toolkit src_db:stuff src_tbl:phrases
src_dsn:P=3306,h=mysql1,p=...,u=root dst_db:stuff dst_tbl:phrases
dst_dsn:p=...,u=root lock:0 transaction:1 changing_src:percona.checksums
replicate:percona.checksums bidirectional:0 pid:9321 user:vagrant host:mysql4*/;
...
```

pt-table-sync

Fix It!

```
user@mysql4$ pt-table-sync --execute --print --sync-to-master --replicate
percona.checksums --wait 0 u=root,p=root > fixed.sql

user@mysql1$ pt-table-checksum --replicate percona.checksums --databases
stuff --user root --pass root --no-check-binlog-format

            TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
03-15T19:52:13      0      0    18365       4       0   0.324 stuff.phrases

mysql4> SELECT db, tbl, SUM(this_cnt) AS total_rows, COUNT(*) AS chunks
FROM percona.checksums WHERE (master_cnt <> this_cnt  OR master_crc <>
this_crc  OR ISNULL(master_crc) <> ISNULL(this_crc)) GROUP BY db, tbl;
Empty set (0.00 sec)
```

Slave Errors

Breaking It

```
 mysql4> INSERT INTO phrases (phrase) VALUES ('Watch this, Mom!');

 mysql4> SHOW SLAVE STATUS\G
                   Last_Errno: 1062
                   Last_Error: Could not execute Write_rows event on
table stuff.phrases; Duplicate entry '30888' for key 'PRIMARY',
Error_code: 1062; handler error HA_ERR_FOUND_DUPP_KEY; the event's
master log mysql-bin.000002, end_log_pos 7202272
                 Skip_Counter: 0
               Last_SQL_Errno: 1062
               Last_SQL_Error: Could not execute Write_rows event on
table stuff.phrases; Duplicate entry '30888' for key 'PRIMARY',
Error_code: 1062; handler error HA_ERR_FOUND_DUPP_KEY; the event's
master log mysql-bin.000002, end_log_pos 7202272
```

Slave Errors

Get Things Going Again

```
mysql4> SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1;

mysql4> SHOW SLAVE STATUS\G
                 Skip_Counter: 1

mysql4> START SLAVE;
mysql4> SHOW SLAVE STATUS\G
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
```

Slave Errors

Fix The Problem

```
user@mysql1$ pt-table-checksum --replicate percona.checksums --databases
stuff --user root --pass root --no-check-binlog-format
            TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
03-17T15:03:51      0      1    31912       4       0   0.334 stuff.phrases

user@mysql4$ pt-table-sync --print --sync-to-master --replicate
percona.checksums --wait 0 u=root,p=root
REPLACE INTO `stuff`.`phrases`(`id`, `phrase`) VALUES ('30888', 'weightless astronaut') /*percona-toolkit
src_db:stuff src_tbl:phrases src_dsn:P=3306,h=33.33.33.11,p=...,u=root dst_db:stuff dst_tbl:phrases
dst_dsn:p=...,u=root lock:0 transaction:1 changing_src:percona.checksums replicate:percona.checksums
bidirectional:0 pid:9515 user:vagrant host:mysql4*/;

user@mysql4$ pt-table-sync --execute --sync-to-master --replicate
percona.checksums --wait 0 u=root,p=root

user@mysql1$ pt-table-checksum --replicate percona.checksums --databases
stuff --user root --pass root --no-check-binlog-format
            TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
03-17T15:06:40      0      0    32243       4       0   0.327 stuff.phrases
```

Slave Errors

Oops, I Broke It Again

```
mysql4> INSERT INTO phrases (phrase) VALUES ('pockmarked convex_polyhedron'),
('pearlescent inflorescence'), ('raffish khufu'), ('thoracic sash_cord'),
('schizophrenic representative_sampling');

mysql4> SHOW SLAVE STATUS\G
                   Last_Errno: 1062
                   Last_Error: Could not execute Write_rows event on table
stuff.phrases; Duplicate entry '32522' for key 'PRIMARY', Error_code: 1062;
handler error HA_ERR_FOUND_DUPP_KEY; the event's master log mysql-bin.000002,
end_log_pos 7544676

user@mysql4$ pt-slave-restart --user root --pass root
2013-03-17T15:15:43 p=...,u=root mysql-relay.000002     7395466 1062
2013-03-17T15:15:43 p=...,u=root mysql-relay.000002     7395677 1062
2013-03-17T15:15:43 p=...,u=root mysql-relay.000002     7395881 1062
2013-03-17T15:15:44 p=...,u=root mysql-relay.000002     7396096 1062
2013-03-17T15:15:44 p=...,u=root mysql-relay.000002     7396294 1062
^C
```

Slave Errors

Fix The Problem (Again!)

```
user@mysql1$ pt-table-checksum --replicate percona.checksums --databases
stuff --user root --pass root --no-check-binlog-format
            TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
03-17T15:16:55      0      1    33006       4       0   0.333 stuff.phrases

user@mysql4$ pt-table-sync --print --sync-to-master --replicate
percona.checksums --wait 0 u=root,p=root
REPLACE INTO `stuff`.`phrases`(`id`, `phrase`) VALUES ('32522', 'suppliant epistle_of_jeremiah')
REPLACE INTO `stuff`.`phrases`(`id`, `phrase`) VALUES ('32523', 'tantrik specialization')
REPLACE INTO `stuff`.`phrases`(`id`, `phrase`) VALUES ('32524', 'uncensored absorption_coefficient')
REPLACE INTO `stuff`.`phrases`(`id`, `phrase`) VALUES ('32525', 'worth silverside')
REPLACE INTO `stuff`.`phrases`(`id`, `phrase`) VALUES ('32526', 'face picture_rail')

user@mysql4$ pt-table-sync --execute --sync-to-master --replicate
percona.checksums --wait 0 u=root,p=root

user@mysql1$ pt-table-checksum --replicate percona.checksums --databases
stuff --user root --pass root --no-check-binlog-format
            TS ERRORS  DIFFS     ROWS  CHUNKS SKIPPED    TIME TABLE
03-17T15:21:30      0      0    33568       4       0   0.335 stuff.phrases
```

Catastrophe

Recent backup + binary logs can be used to recover from some errors

http://www.flickr.com/photos/slava/496607907

Frack!

```
mysql1> DELETE FROM phrases;
Query OK, 35205 rows affected
(0.10 sec)
```

Rescue Us, Binary Logs

1. SHOW MASTER STATUS immediately
2. Get the master log position from a recent backup
3. Restore recent backup to a recovery server

Rescue Us, Binary Logs

1. Error happened between MASTER_LOG_POS 6360 and 264428 in mysql-bin.000004

```
mysql1> SHOW MASTER STATUS\G
            File: mysql-bin.000004
        Position: 264428

user@mysql1$ grep 'CHANGE MASTER' dump-20130317.sql
-- CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000004',
MASTER_LOG_POS=6360;
```

Rescue Us, Binary Logs

1. Restore backup

```
user@recovery$ mysql -u root -p < dump-20130317.sql
```

2. Set up as slave at backup log position

```
recovery> CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000004',
MASTER_LOG_POS=6360, MASTER_HOST='mysql1', MASTER_USER='repl',
MASTER_PASSWORD='replication';
```

3. Don’t start the slave yet!

Find Position of Error

```
root@mysql1# mysqlbinlog mysql-bin.000004 --start-position 6360 --stop-position 264428 --server-id 1 | grep -C10 -i DELETE
```

```
#130317 15:35:03 server id 1  end_log_pos 24807       Xid = 45974
COMMIT/*!*/;
# at 24807
#130317 15:35:03 server id 1  end_log_pos 24876       Query  thread_id=163      exec_time=0  error_code=0
SET TIMESTAMP=1363534503/*!*/;
BEGIN
/*!*/;
# at 24876
#130317 15:35:03 server id 1  end_log_pos 24959       Query  thread_id=163      exec_time=0  error_code=0
SET TIMESTAMP=1363534503/*!*/;
DELETE FROM phrases
/*!*/;
# at 24959
#130317 15:35:03 server id 1  end_log_pos 24986       Xid = 45973
COMMIT/*!*/;
# at 24986
#130317 15:35:04 server id 1  end_log_pos 25055       Query  thread_id=160      exec_time=0  error_code=0
SET TIMESTAMP=1363534504/*!*/;
BEGIN
/*!*/;
# at 25055
```

Skip the Error

```
recovery> START SLAVE UNTIL MASTER_LOG_POS=24807,
MASTER_LOG_FILE='mysql-bin.000004';

recovery> STOP SLAVE;

recovery> CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000004',
MASTER_LOG_POS=24986;

recovery> SELECT COUNT(*) FROM stuff.phrases\G
count(*): 40165

mysql1> SELECT COUNT(*) FROM stuff.phrases\G
count(*): 5207
```

MariaDB

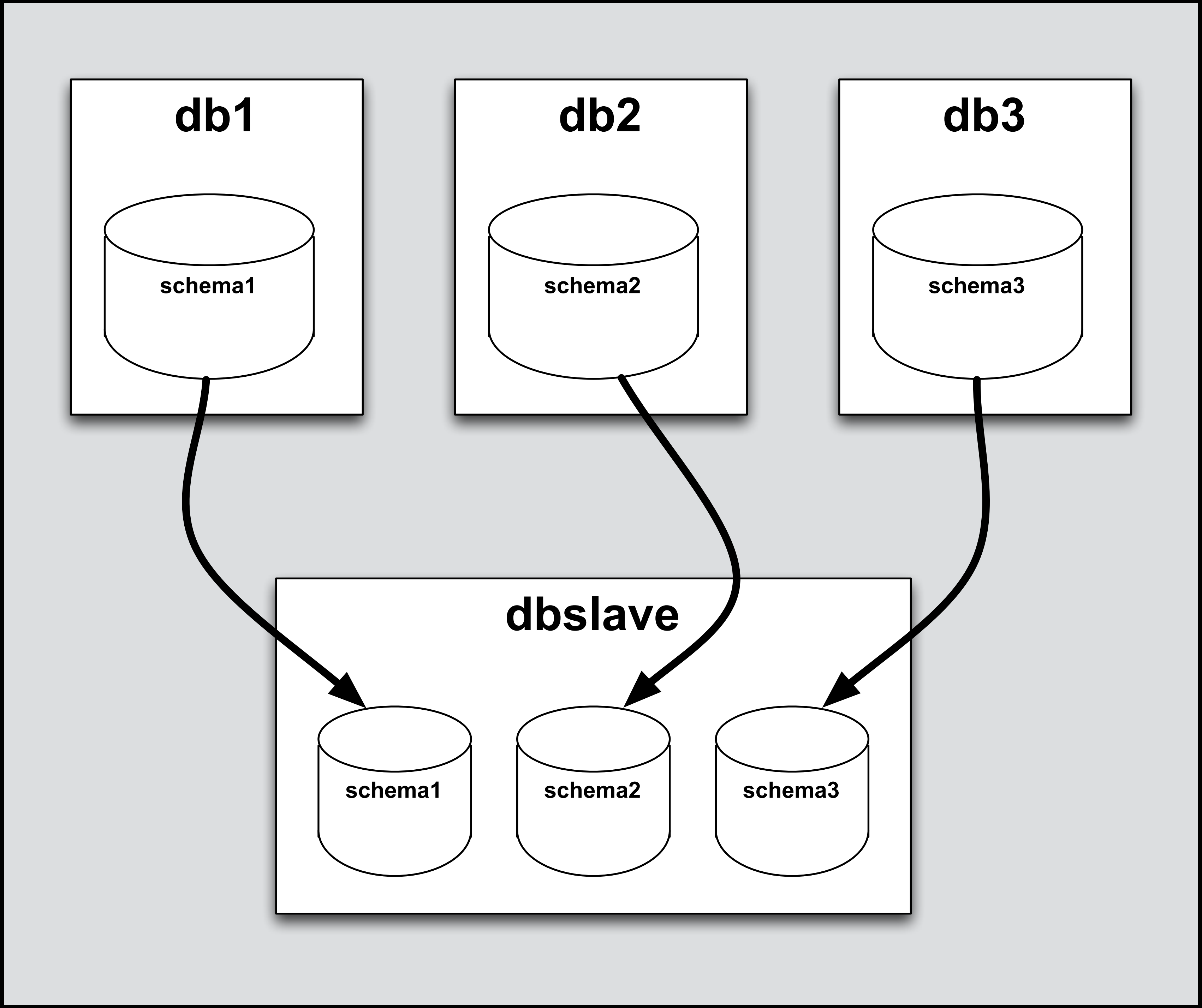
Multi-Source Replication

https://mariadb.com/kb/en/multi-source-replication/

MariaDB

1. Provides "Multi-Source Replication" in 10.x (beta)
2. https://mariadb.com/kb/en/multi-source-replication/

MariaDB


db1
schema1
db2
schema2
db3
schema3
dbslave
schema1
schema2
schema3

MariaDB

Replication Status

```
dbslave> SHOW SLAVE STATUS\G
Empty set (0.00 sec)
```

WAT!?

MariaDB

Replication Status

1. Use the connection name

```
dbslave> SHOW SLAVE “schema1” STATUS\G

           Connection_name: schema1
           Master_Log_File: mysql-binlog.000291
       Read_Master_Log_Pos: 25768737
          Slave_IO_Running: Yes
         Slave_SQL_Running: Yes
```

MariaDB

Replication Status

2. Set the default_master_connection

```
dbslave> SET @@default_master_connection=‘schema1’;

dbslave> SHOW SLAVE STATUS\G

          Connection_name: schema1
            Master_Log_File: mysql-binlog.000291
      Read_Master_Log_Pos: 25768737
          Slave_IO_Running: Yes
         Slave_SQL_Running: Yes
```

MariaDB

Replication Status

3. Look at all the things

```
dbslave> SHOW ALL SLAVES STATUS\G

*************************** 1. row ***************************
                Connection_name: schema1
…
*************************** 2. row ***************************
                Connection_name: schema2
…
*************************** 3. row ***************************
                Connection_name: schema3
…
```

MariaDB

Setting Up Replication

1. Use the connection name in the statement

```
dbslave> CHANGE MASTER “schema1” TO
MASTER_HOST=‘db1’, MASTER_USER=‘repl’,
MASTER_PASSWORD=‘replication’,
MASTER_LOG_FILE=’mysql-bin.000002’, MASTER_LOG_POS=
882394;

dbslave> START SLAVE “schema1”;
```

MariaDB

Setting Up Replication

2. Set the default_master_connection

```
dbslave> SET @@default_master_connection=‘schema1’;

dbslave> CHANGE MASTER TO MASTER_HOST=‘db1’,
MASTER_USER=‘repl’, MASTER_PASSWORD=‘replication’,
MASTER_LOG_FILE=’mysql-bin.000002’, MASTER_LOG_POS=
882394;

dbslave> START SLAVE;
```

MariaDB

Using Third Party Utilities

Most (nearly all) utilities don’t support MariaDB Multi-Source Replication

MariaDB

Using Third Party Utilities

1. Set the default_master_connection in the utility

```
user@dbslave$ pt-table-sync --print --sync-to-master
--replicate percona.checksums --wait 0 u=root,p=root
--set-vars=default_master_connection='schema1'
```

MariaDB

Using Third Party Utilities

2. Set default_master_connection in a .my.cnf file - Doesn’t work yet :(

```
mysql: unknown variable
'default_master_connection=schema1'
```

Stuff I Didn't Cover

1. MySQL 5.5+: Semi-Synchronous Replication - http://blog.9minutesnooze.com/performance-mysql-replication-high-latency/
2. MySQL 5.6: Global Transaction IDs & parallel replication

Questions?