



Diagnosing Failures in MySQL Replication

Devananda van der Veen
Percona Live London 2011

Introduction

- About Me
 - Sr Consultant at Percona since 2009
 - Worked with MySQL replication since 2004
 - Live in NW Washington
- About Percona
 - Support, Consulting, Training, Development
 - We create & maintain Percona-Server w/ XtraDB and XtraBackup
 - Many other tools too: ~~maatkit & aspersa~~ percona-toolkit

Table of Contents

- Introduce the Problems
- Replication Architecture Review
- Masters, Slaves, and the many files they keep
- Different ways things break
- How to diagnose the problems
- Knowing when to skip it, when and how to fix it, and when to just rebuild it.
- Questions (if we have time)

Replication is Ubiquitous

Who *doesn't* use MySQL Replication for...

- Load distribution
- Redundancy (local or geographic)
- Master-Master fail-over
- Taking backups from slave
- Archiving & Data Warehousing
- Many other things too

Ubiquitous... but not perfect

Google “mysql replication failure” → 130,000+ results



Ubiquitous... but not perfect

MySQL lists **51** Active bugs in Replication
as of October 15, 2011

26945	Explicit TEMP TABLE causes replication failure if mysqld restarted
48062	Date arithmetic in Stored Procedure breaks replication
43457	replicate-ignore-db behaves differently with different binlog formats
58637	INSERT..ON DUP KEY UPDATE unsafe if more than one unique key
62557	SHOW SLAVE STATUS gives wrong output with master-master and using SET uservars

<http://bit.ly/MysqlRepBugs>

Most problems have a Solution

Here are some specific problems that I'll discuss
(time permitting)

- Duplicate key errors
- Temp table does not exist after slave restart
- Binary and relay log corruption
- Slave reads from wrong position after crash
- And more...

But first ...

We need to know a little about how MySQL replication works

- Master & Slave threads
- Binary & Relay log files
- Status files (or status *tables* in 5.6)

If you want to know more, see Lars Thalmann's presentations

<http://bit.ly/LarsRepTalk>

<http://bit.ly/LarsRepTalk1>

A Brief Review

Master Host		
Write to	<code>master-bin.xxx</code>	(binary file)
Keeps list at	<code>master-bin.index</code>	(plain-text file)

Slave Host			
Composed of two threads: IO + SQL			
IO_Thread		SQL_Thread	
Status file	<code>master.info</code>	Status file	<code>relay.info</code>
Reads from	<code>master-bin.xxx</code>	Reads from	<code>relay-bin.xxx</code>
Writes to	<code>relay-bin.xxx</code>		
Local list	<code>relay-bin.index</code>		

<http://bit.ly/5-5-slave-log-status>

The Index Files

```
# cat master-bin.index      Master binary log index file
./master-bin.0000001
./master-bin.0000002
```

```
# cat relay-bin.index      Slave relay log index file
./relay-bin.0000001
./relay-bin.0000002
```

You sometimes need to edit these files manually
(but only in dire situations)

The Info Files

```
Slave:$ cat master.info
```

```
15  
master-bin.000001  
1818  
127.0.0.1  
repl_user  
repl_pass ←(plain-text!!)  
19501  
60  
...
```

```
# of lines in file  
Master_Log_File  
Read_Master_Log_Pos  
Master_Host  
Master_User  
Master_Password  
Master_Port  
Connect_Retry  
...
```

current
position
of io_thd
on master

The Info Files (2)

```
Slave:$ cat relay-log.info
```

```
./relay-bin.000002  
1392  
master-bin.000001  
1818
```

```
Relay_Log_File  
Relay_Log_Pos  
Relay_Master_Log_File  
Exec_Master_Log_Pos
```

local read position
of sql_thd



remote read position
of sql_thd



These positions correspond to the `end_log_pos` of the last statement executed by `sql_thread`

mysql> show slave status\G

```
Slave_IO_State: Waiting for master  
Slave_IO_Running: Yes  
Slave_SQL_Running: Yes
```

```
Master_Log_File: master-bin.000001  
Read_Master_Log_Pos: 1818
```

io_thread
remote pos

```
Relay_Log_File: relay-bin.000001  
Relay_Log_Pos: 251
```

sql_thread
local pos

```
Relay_Master_Log_File: master-bin.000001  
Exec_Master_Log_Pos: 1818
```

sql_thread
remote pos

Fields re-ordered for display

Running Example

```
~/sandboxes$ make_rep1_sandbox --circular 2 5.1.55
```

Percona-Server-5.1.55-rel12.6-200-Linux-x86_64

Node1 = Active master | Node2 = Passive Master

```
node1 > create table foo ( id int not null
auto_increment primary key, v varchar(255) not null
default '', unique (v)) engine=innodb;
Query OK, 0 rows affected (0.01 sec)
```

```
node1 > insert into foo (v) values ('a'), ('b');
Query OK, 2 rows affected (0.01 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

mysqlbinlog on masters

at 1610

#110318 14:20:12 server id 101 end_log_pos 1683

Query thread_id=8 exec_time=0 error_code=0

SET TIMESTAMP=1300483212/*!*/;

BEGIN/*!*/;

at 1683

#110318 14:20:12 server id 101 end_log_pos 1791

Query thread_id=8 exec_time=0 error_code=0

SET TIMESTAMP=1300483212/*!*/;

INSERT INTO `foo` VALUES (1, 'a'), (2, 'b')/*!*/;

at 1791

#110318 14:20:12 server id 101 end_log_pos 1818

Xid = 32

COMMIT/*!*/;

mysqlbinlog on relay logs

```
# at 1184
#110318 14:20:12 server id 101  end_log_pos 1683
Query    thread_id=8      exec_time=0      error_code=0
SET TIMESTAMP=1300483212/*!*/;
BEGIN/*!*/;
```

end_log_pos is copied from master!



```
# at 1257
#110318 14:20:12 server id 101  end_log_pos 1791
Query    thread_id=8      exec_time=0      error_code=0
SET TIMESTAMP=1300483212/*!*/;
INSERT INTO `foo` VALUES (1, 'a'), (2, 'b')/*!*/;
```

```
# at 1365
#110318 14:20:12 server id 101  end_log_pos 1818
      Xid = 32
COMMIT/*!*/;
```

mysqlbinlog on slave binlog

```
# at 1610
#110318 14:20:12 server id 101  end_log_pos 1674
Query    thread_id=8      exec_time=15    error_code=0
SET TIMESTAMP=1300483212/*!*/;
BEGIN/*!*/;
```

Here, `exec_time` means “slave lag”!

```
# at 1674
#110318 14:20:12 server id 101  end_log_pos 1782
Query    thread_id=8      exec_time=15    error_code=0
SET TIMESTAMP=1300483212/*!*/;
INSERT INTO `foo` VALUES (1, 'a'), (2, 'b')/*!*/;
```

```
# at 1782
#110318 14:20:12 server id 101  end_log_pos 1809
      Xid = 35
COMMIT/*!*/;
```

End of the Review

Any Questions?
(find me after the talk)

Finally, the meat of the talk!



My Approach

When replication stops...

- **Why** did it stop?
 - This usually tells you how to fix it
 - Generally does not take much time
- **Where** did it break?
 - Determine extent of corruption / data loss
 - Were statements skipped or run twice?
- **What is the business impact?**
- Don't over-analyze. Estimate is probably OK at this step.

My Approach

Now that you know why it stopped...

- Choose your restore method
 - Do you prioritize *site availability* or *data integrity*?
- Validate slave consistency
 - If possible, before allowing traffic
 - if necessary, use tools to resync the slave
- Have a “plan B”
 - Sometimes you just need to rebuild

Understand What Failed

Where and **why** the failure happens matters a lot!

- Understand the basics.
- Know your architecture.

How you solve it is ...

- Learned from experience
- Based on common principles
- Similar even for different organizations

Two Times when Replication Fails

Server OK but replication stopped

- Writes on the slave
- File corruption in binary or relay-log
- Documented limitation in MySQL
- Un/known Bugs?

Crash caused replication to stop

- Hardware fails
- `mysqld` crash
- Kernel panic / OOM killer
- `kill -9 `pidof mysqld``
- Host recently restarted and “rolled back” status file

Two Times when Replication Fails

Server OK but replication stopped

Dedicated Monitoring!!

Seconds_Behind_Master lies.

Watch IO and SQL thread state and position.

Use a heartbeat table.

Crash caused replication to stop

skip-slave-start

innodb_overwrite_relay
_log_info

(only in Percona-Server)

5.6-labs features

<http://bit.ly/crash-safe-replication>

Common Situations

- Documented limitations & bugs
- Writes on the slave
- File Corruption
- Hardware Failures

Documented Limitations

Documented limitations

- Statement-Based Replication (SBR) has many limitations
 - Documentation: <http://bit.ly/5-5-rep-limits>
 - **33** subsections
 - Still the default format, even in 5.5
- Row-Based Replication (RBR) avoids most limitations
 - But not all: <http://bit.ly/5-5-rbr-limits>
 - All tables require a PRIMARY KEY
 - Can be more difficult to do “online alter table” than SBR

Documented limitations

- Get to know the limitations & open bugs
 - Some are “documented” as bug reports
- Avoid them like the plague
- Watch the patch notes like a hawk
(even if you don't upgrade)
- Change your application as necessary

Some limitations in SBR...

| Note | 1592 | Statement may not be safe to log
in statement format. |

- Non-Deterministic functions or statements
 - UUID, NOW, CURRENT_USER, FOUND_ROWS
 - UPDATE|DELETE with LIMIT but no ORDER BY
- TEMPORARY and MEMORY tables
- SET TX_ISOLATION = READ_COMMITTED
- Routines / Triggers with logical expressions
- Dynamic SQL
- And lots more ...

Some limitations in RBR...

- Can't read SQL in binary log
- Different DDL on master / slave breaks RBR (*)
- Slave calls fsync() for every row, not every transaction
(this is really a performance issue, not a limitation)
- Mixing InnoDB and MyISAM in one transaction ...
 - It's just not a good idea!
 - Order of binlog events changed multiple times in 5.1

(*) Percona is working to enable slaves with different table definitions to replay RBR events that would normally fail.

Writes on the Slave

- (1) When it's just a slave
- (2) When it's also a master

Write on Slave (1)

(when a slave is just a slave)

```
node2 > insert into foo (v) values ('bad insert');
```

```
node2 > select * from foo;
```

1	a
2	b
3	bad insert

```
node1 > insert into foo (v) values ('c');
```

```
node2 > show slave status\G
```

```
Last_Errno: 1062
```

```
Last_Error: Error 'Duplicate entry '3' for key 'PRIMARY'' on query. Default database: 'test'.
```

```
Query: 'insert into foo (v) values ('c')'
```

Write on Slave (1)

(when a slave is just a slave)

Sample binlog from slave (node2)

```
#110328 13:55:52 server id 101  end_log_pos 640
  Query   thread_id=7      exec_time=0
error_code=0
INSERT INTO `foo` VALUES (1, 'a'), (2, 'b')
```

```
#110328 13:56:34 server id 102  end_log_pos 877
  Query   thread_id=7      exec_time=0
error_code=0
SET TIMESTAMP=1301345794/*!**/;
insert into foo (v) values ('bad insert')
```

Write on Slave (1)

(when a slave is just a slave)

- DELETE record from slave; optionally, insert to master
 - may preserve data integrity
 - insert to master allows you to keep the record
 - more time-consuming than SKIP_COUNTER
 - foreign keys and triggers make this very complicated!

Write on Slave (1)

(when a slave is just a slave)

- `SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1;`
 - easiest method
 - data inconsistency may propagate to other tables, eg. if you use `INSERT . . SELECT` or triggers
 - must sync later with `pt-table-sync`

Write on Slave (1)

(when a slave is just a slave)

- `auto_increment_increment` & `_offset`
 - would have prevented failure in this example
 - safe to do even if you “never” write to slave
- `master = even`, `slave = odd`
 - makes it trivial to identify bad writes - `id` will be odd

Write on Slave (2)

(when a slave is also a master)

```
node1 > insert into foo (v) values ('c');
node1 > SHOW SLAVE STATUS\G
Error 'Duplicate entry '3' for key 'PRIMARY'' on
query. Default database: 'test'. Query: 'insert
into foo (v) values ('bad insert')'
```

```
-----
node2 > insert into foo (v) values ('bad insert');
node2 > SHOW SLAVE STATUS\G
Error 'Duplicate entry '3' for key 'PRIMARY'' on
query. Default database: 'test'. Query: 'insert
into foo (v) values ('c')'
```

Note: I used SLAVE STOP; on both hosts to simulate simultaneous writes

Write on Slave (2)

(when a slave is also a master)

- Fixing master-master is same principle as master-slave...
- But you might accidentally corrupt your primary master!
- So be extra careful...

For example:

- DELETE record from secondary; optionally, insert to primary
 - Use `SET SESSION SQL_LOG_BIN = 0` for DELETE
 - But allow INSERT to replicate normally

Write on Slave (2)

(when a slave is also a master)

- `SET SQL_SLAVE_SKIP_COUNTER = 1`
 - On both masters
 - Then sync with `pt-table-sync`
- `auto_increment_increment & _offset`
 - A “must have” if you use `AUTO_INC`
 - + master-master replication
 - + fail-over

Common mistake re: SKIP

There is a real open source project that ...

“[W]ill not only check on the health of your MySQL Replication setup, but it can also **automatically heal and restart the MySQL Replication**. Self-Healing MySQL Replication is not a new idea.”

<http://bit.ly/replication-suicide>

... but it's a really bad idea!!

File Corruption

File Corruption

Slave may stop if any of these files become corrupt

Binary log | Relay log | Table data | Index data

Examples of SHOW SLAVE STATUS

Last_IO_Error: 1236

Last_IO_Error: Got fatal error 1236 from master
when reading data from binary log

Last_SQL_Errno: 1594

Last_SQL_Error: Relay log read failure: Could not
parse relay log event entry.

If Master Binlog is Corrupt...

- Verify it with `mysqlbinlog [--verbose --base64]`
- Find extent of corruption
- `CHANGE MASTER TO Master_Log_Pos = <good_pos>`
- Analyze bad section of log. Identify all affected tables.
- Use `pt-table-sync` when replication catches up.
- Resolve underlying (hardware) cause of the corruption!

If Slave Relay Log is Corrupt...

- Verify it with `mysqlbinlog <relay-log-file>`
- Check master binlog for corruption.
 - If it is, see previous slide.
- Re-fetch the corrupted relay log
`CHANGE MASTER TO`
`Master_Log_Pos = <Exec_master_log_pos>;`
- Very rare
- If this occurs frequently, check for network problems.

If a Table is Corrupt...

- Stop replication and stop all traffic to that server
- Mount the file system read-only
- Check mysql error log - it may contain more info
- Even after “fixing” the corruption, `pt-table-sync`
- Sometimes, only option is restore from a backup

Hardware Failures

“A SAN is just a bigger SPoF”
-unknown

HW Failure #1

Unplanned master restart → all slaves stop with error:
'Client requested master to start replication
from impossible position'

Process:

- Compare each SLAVE STATUS to master binlog
- Realize that Exec_Master_Log_Pos different on each slave, and *greater than master's log file size!*
- Panic for a minute...

HW Failure #1

Possible solutions:

- Promote slave that read the most
- Isolate “extra” events from slave binlog, replay on master
- Force slaves to resume from next binlog, then `pt-table-sync` to remove “extra” events

Prevent with:

- RAID BBU + HDD cache disabled
- `innodb_flush_log_at_trx_commit = 1`
- `sync_binlog = 1`

HW Failure #2

Replaced a failed disk in a replica.
After restart, replication fails with duplicate key error.

Process:

- Check error log... *there's no slave stop coordinates!*
- Realize file system in read-only mode before shut down
- Was master .info rolled back by file system journal?

HW Failure #2

Possible solutions:

- Guess where slave stopped, then `CHANGE MASTER`
But what if data files also rolled back?
- `SQL_SLAVE_SKIP_COUNTER` + prayers + `pt-table-sync`
- Rebuild the slave

Prevent with:

- Percona Server + `innodb_overwrite_relay_log_info`

War Stories

- Slave generates different `auto_inc` values for INSERT inside an `ON_INSERT TRIGGER`. This goes unnoticed for months, then starts causing problems.
- During MMM migration, a few uncommitted transactions get left behind and cause duplicate key errors on both nodes
- Major hardware failure in a geo-distributed three-node replication ring. Each host is written to by local processes. There are some tables only written on single host, but also some shared tables that all write to. How do you determine which of remaining two masters is “good”?

Thank You to Our Sponsors

Platinum Sponsor



Gold Sponsor



Silver Sponsors



Percona Live London Sponsors

Exhibitor Sponsors



Friends of Percona Sponsors

Couchbase



Monty Program



Tokutek



Media Sponsors



O'REILLY

Annual MySQL Users Conference

Presented by Percona Live

The Hyatt Regency Hotel, Santa Clara, CA

April 10th-12th, 2012

Featured Speakers

Mark Callaghan, Facebook

Jeremy Zawodny, Craigslist

Marten Mickos, Eucalyptus Systems

Sarah Novotny, Blue Gecko

Peter Zaitsev, Percona

Baron Schwartz, Percona

The Call for Papers is Now Open!

Visit www.percona.com/live/mysql-conference-2012/



deva@percona.com
devananda.vdv@gmail

We're Hiring!
percona.com/about-us/careers/



PERCONA
LIVE

www.percona.com/live