ORACLE®

MySQL Replication Update

MySQL 5.5 (GA) & MySQL 5.6.2 (Dev. Milestone)

Lars Thalmann
Development Director
MySQL Replication, Backup & Connectors

O'Reilly MySQL Users Conference, April 2011

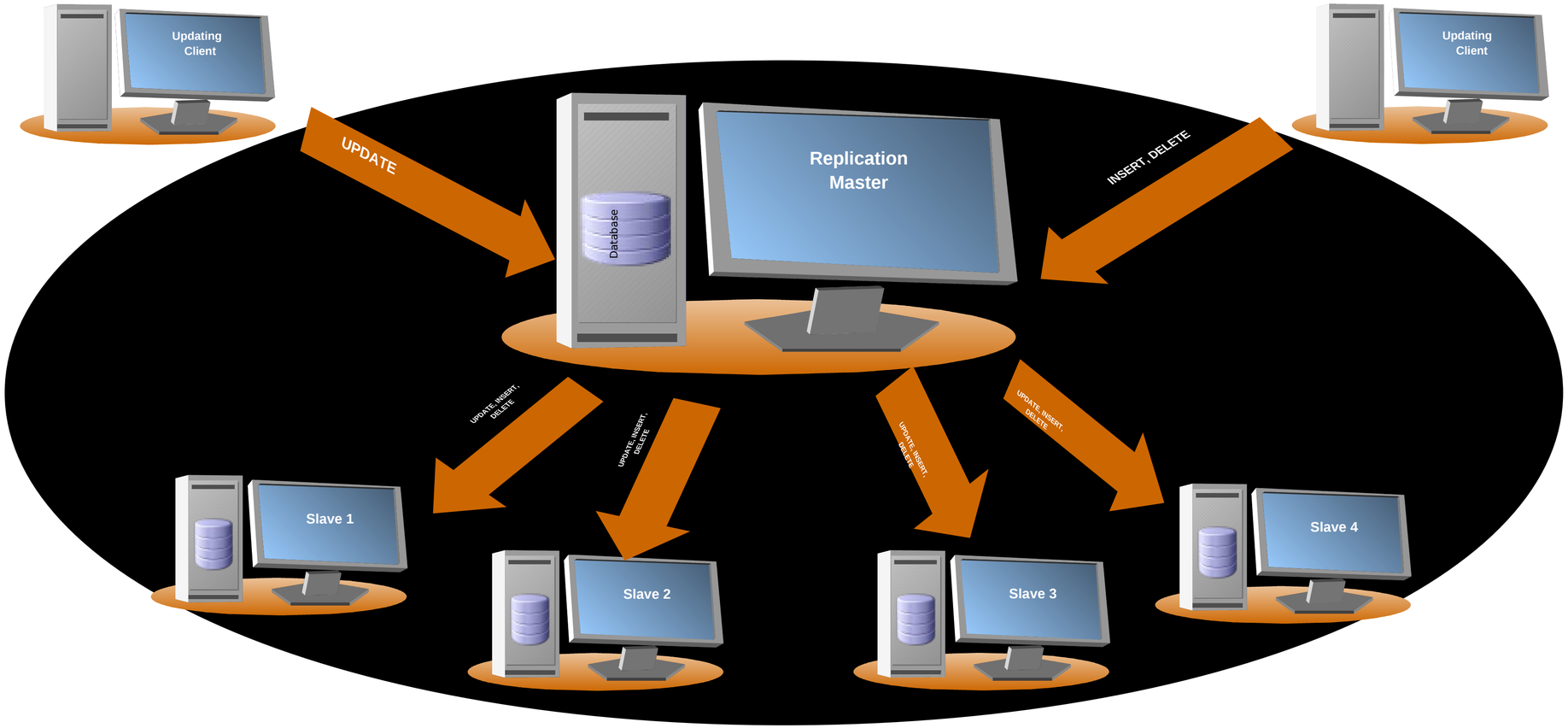
MySQL Releases

- **MySQL 5.1** - *Generally Available, November 2008*
- **MySQL 5.5** - *Generally Available, December 2010*
- **MySQL 5.6.2** - *Development Milestone Release, April 2011*

What is MySQL Replication?

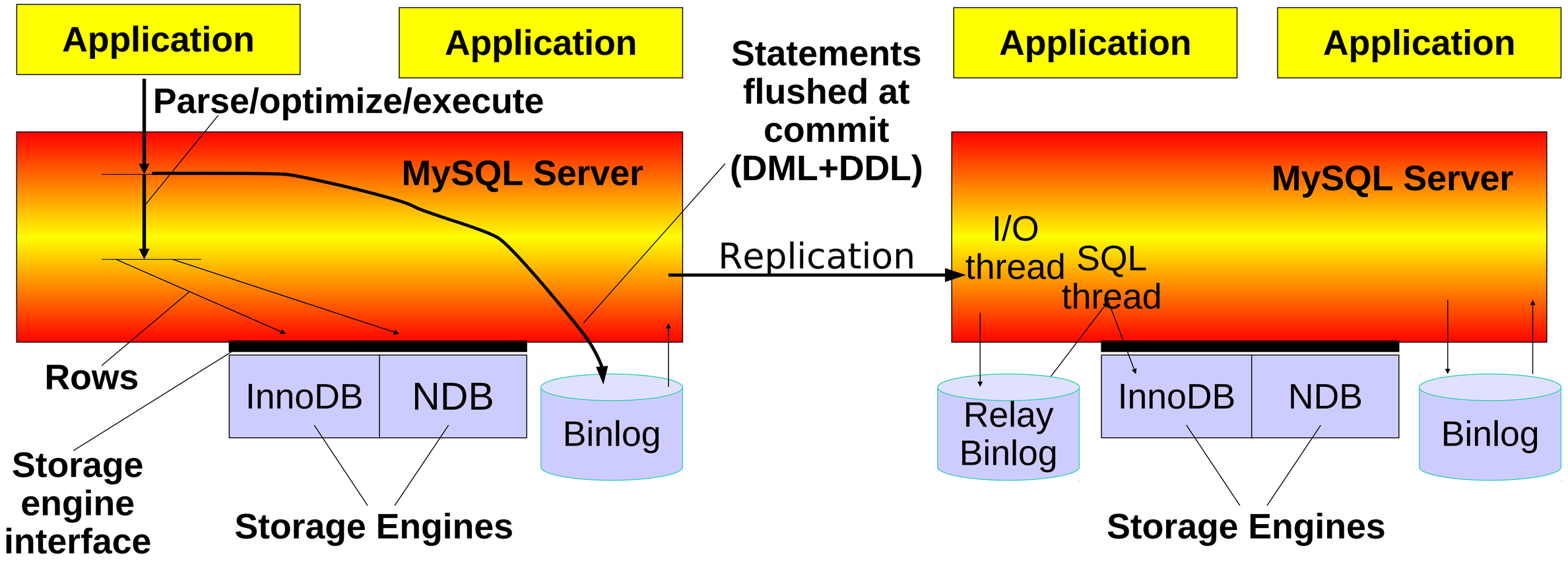
MySQL Replication

Asynchronous replication for maximum performance



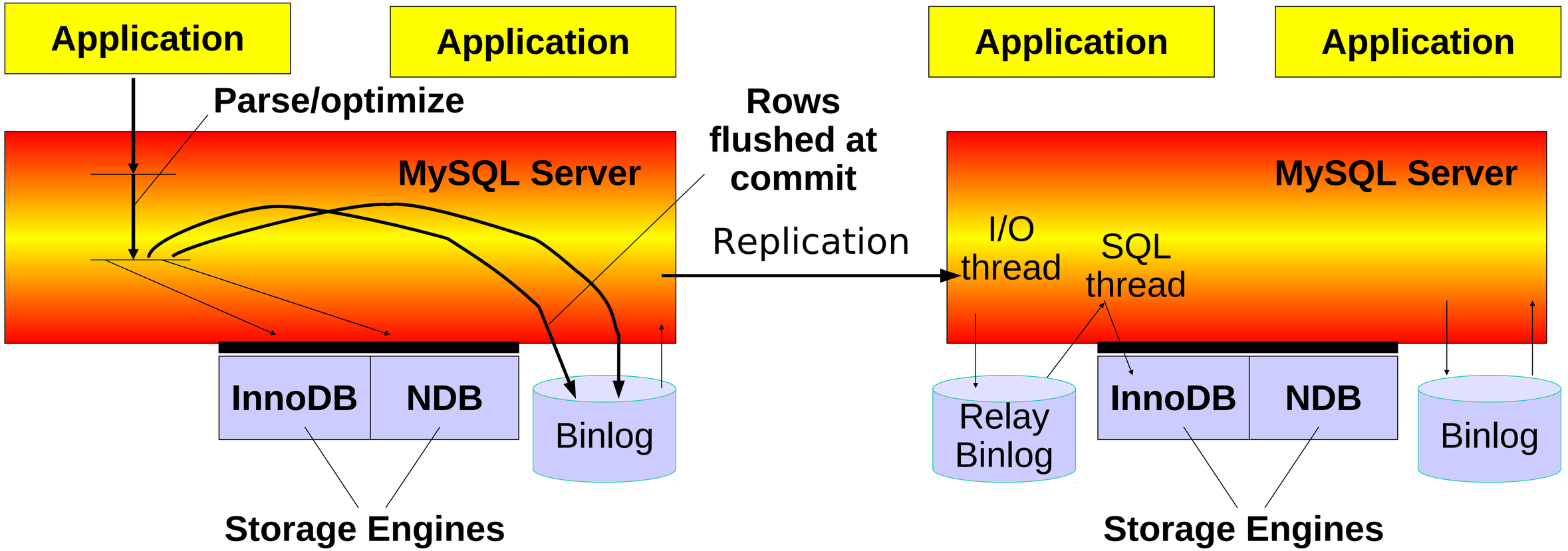
MySQL Replication Architecture

Statement-based replication



MySQL Replication Architecture

MySQL 5.1: Row-based replication



MySQL 5.5 (GA)

MySQL Replication users wanted

High Availability Enhancements

- Be sure that **slave has received the updates** from master
- **Tune replication** for maximum performance or safeness
- Get a crashed slave to **automatically recover** the relay log
- Immediately **detect if replication is not working**

Flexibility Enhancements

- **Filter events from particular servers**
- **Flush logs independently**
- **Correctly convert data** when master/slave use different data types

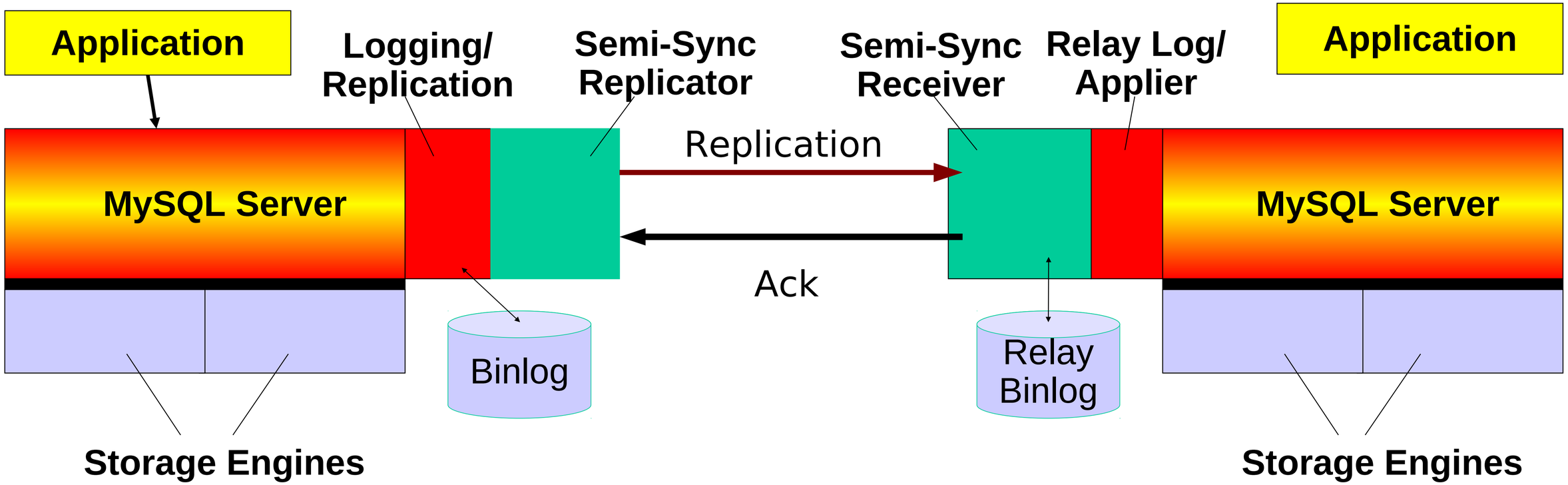
This is included in MySQL 5.5

MySQL 5.5 Replication Features

1. **Semisynchronous replication**
   Improved resilience by having master wait for slave to persist events.
2. **Slave fsync tuning & Automatic relay log recovery**
   Tune fsyncs so corruption is less likely on slave crashes.
   Let the slave recover from corrupted relay logs.
3. **Replication Heartbeat**
   Have a more precise failure detection mechanism.
   Avoid spurious relay log rotation when the master is idle.
4. **Per server replication filtering**
   Instruct slave to discard events from a master with a specific server id
5. **Precise Slave Type Conversions**
   Use different types on master and slave
   Get automatic type promotion and demotion when using RBR
6. **Individual Log Flushing**
   Selectively flush server logs when using 'FLUSH LOGS'

1. Semisynchronous Replication

Originally developed by Mark Callaghan and Wei Li, Google
Modularized, tested, and bug fixed by Zhenxing He, MySQL



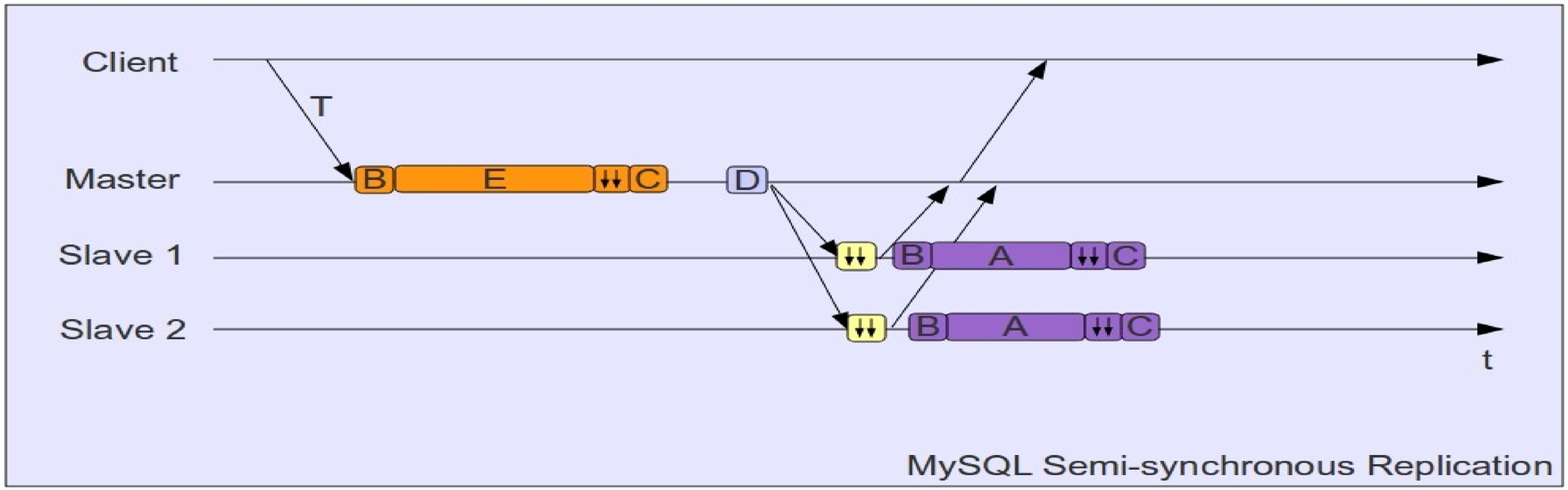
Available as two separate loadable components for the master and the slave
Slave acknowledge relay logging each transaction

1. Semisynchronous Replication

Write to binary log
Write to relay log
Write to binary log
B B Begin
C C Commit

T – transaction
t – time

A Applier - SQL thread
D Distribution - Dump threads
E Execution – Session Thread
Collector - IO Thread



2. Slave fsync tuning

Three new variables: **sync_relay_log_info**, **sync_master_info**, **sync_relay_log** for fsync of replication meta data and log.

sync_relay_log_info
Synchronize relay-log.info file to disk after that many transactions

sync_master_info
Slave synchronize master info after that many events.

sync_relay_log
Slave synchronizes the relay after this many events.

2. Automatic Relay Log Recovery

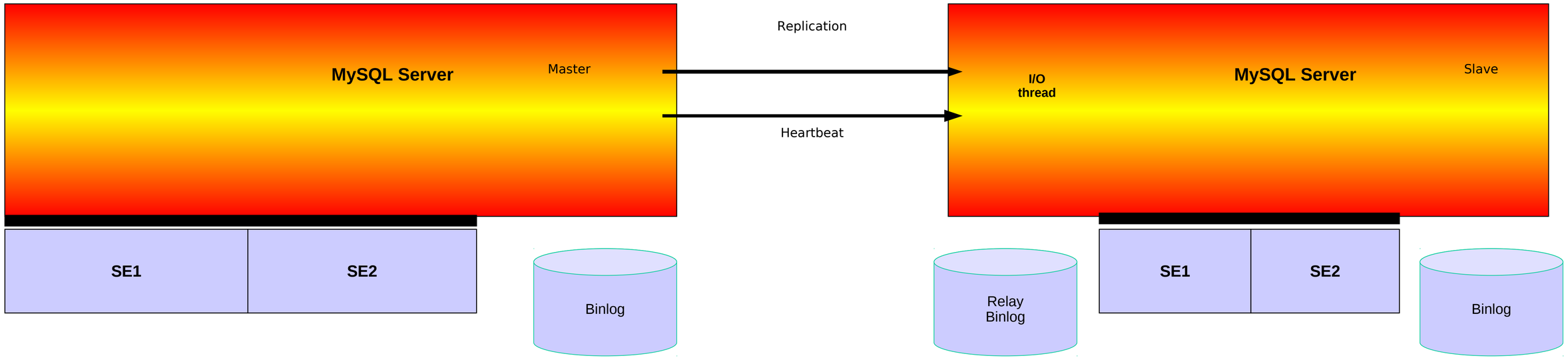
relay_log_recovery = 1

On restart, slave discards all unprocessed relay logs (and retrieves them from master).

This can be used after a slave crash to ensure that potentially corrupted relay logs are not processed.

The default value is 0 (disabled).

3. Replication Heartbeat



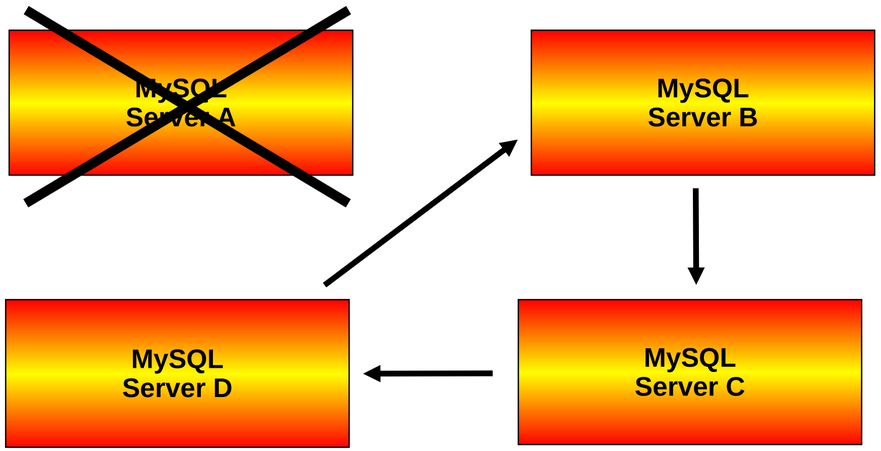
Automatic checking of connection status

No more relay log rotates when the master is idle

Detection of master/slave disconnect configurable in millisecs

CHANGE MASTER SET master_heartbeat_period= val;
SHOW STATUS like 'slave_heartbeat period'
SHOW STATUS like 'slave_received_heartbeats'

4. Per server replication filtering



If server A is removed from the circle, server B can be set to terminate A's events in the new circle

Server B> **CHANGE MASTER TO MASTER_HOST=D ... IGNORE_SERVER_IDS=(A)**

5. Precise Slave Type Conversions

- **Example, MySQL 5.5 row-based**
  SLAVE_TYPE_CONVERSIONS = 'ALL_LOSSY':

  master> CREATE TABLE foo (a INT);

  slave> CREATE TABLE foo (a TINYINT);

  master> INSERT INTO foo VALUES (1);

  slave> <<<success>>>

  Example, MySQL 5.5 row-based
  SLAVE_TYPE_CONVERSIONS = '':

  master> CREATE TABLE foo (a INT);

  slave> CREATE TABLE foo (a TINYINT);

  master> INSERT INTO foo VALUES (1);

  slave> <<<error>>>

6. Individual log flushing

Flush of individual logs:

FLUSH <log_type> LOGS;

Examples:

FLUSH ERROR LOGS, RELAY LOGS;

FLUSH BINARY LOGS, ENGINE LOGS, SLOW LOGS;

Log types supported:

- **SLOW** - close & reopen the slow query log file.
- **ERROR** - close & reopen the error log file.
- **BINARY** - close & reopen the binary log files.
- **ENGINE** - close & reopen any flushable logs for installed storage engines
- **GENERAL** - close & reopen the general query log file
- **RELAY** - close & reopen the relay log files

MySQL 5.6.2 (Development Milestone)

MySQL 5.6.2 Development Milestone Replication Features

1. Crash-safe slave – replication info tables
2. Crash-safe master – binary log recovery
3. Replication event checksums
4. Time delayed replication
5. Optimized row-based logging
6. Informational log events
7. Remote backup of binary logs
8. Server UUIDs – Replication topology detection

1. Crash-safe slave - Slave Info Tables

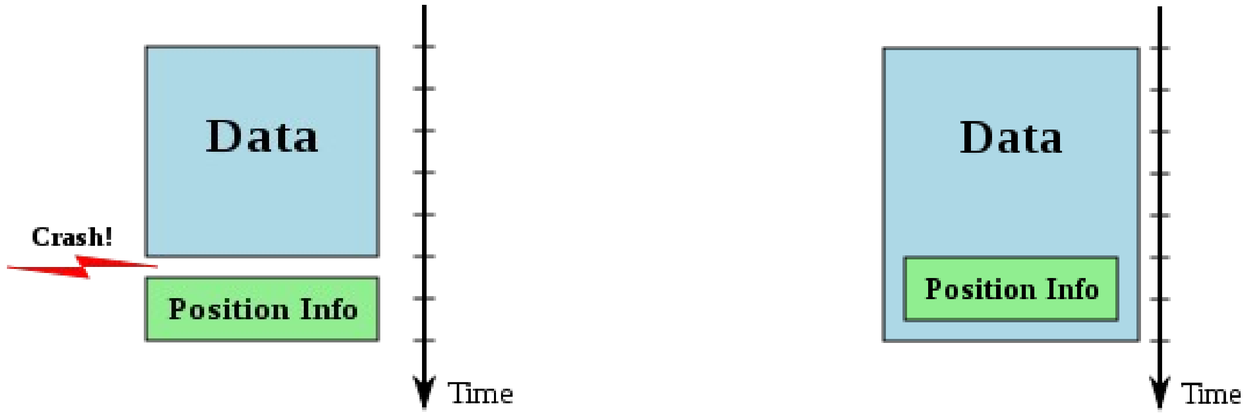
5.6.2 DM

- Protection against slave crashes
  - Automatic recovery
  - Engine agnostic
- Possibility to do SELECT of slave information
  - Possibility to code multi-source replication in pure SQL
- Automatic conversion between files and tables on startup

1. Crash-safe slave - Slave Info Tables

5.6.2 DM

- System tables:
  - slave_master_info (master.info)
  - slave_relay_log_info (relay-log.info)
- Positional info transactionally stored with the data in tables



2. Crash-safe master

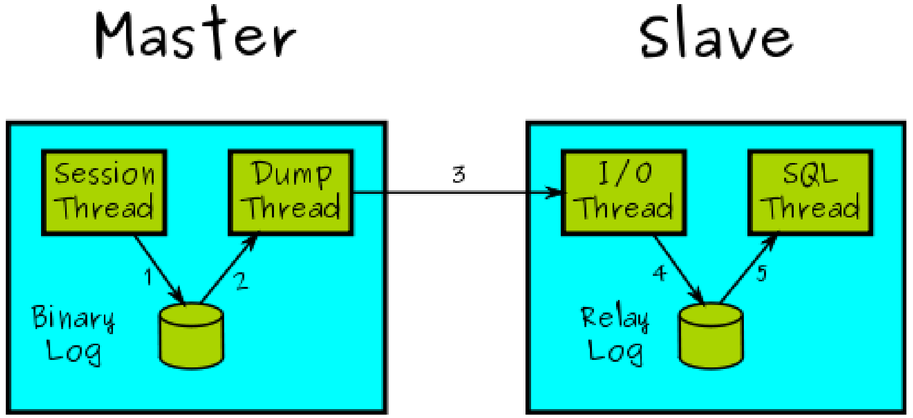
5.6.2 DM

- Server can cope with binary log corruption in the event of a crash
- On restart
  - The active binary log is scanned and any log corruption is detected
  - Invalid portion of the binary log file is discarded and the file is trimmed

3. Replication Events Checksums

5.6.2 DM

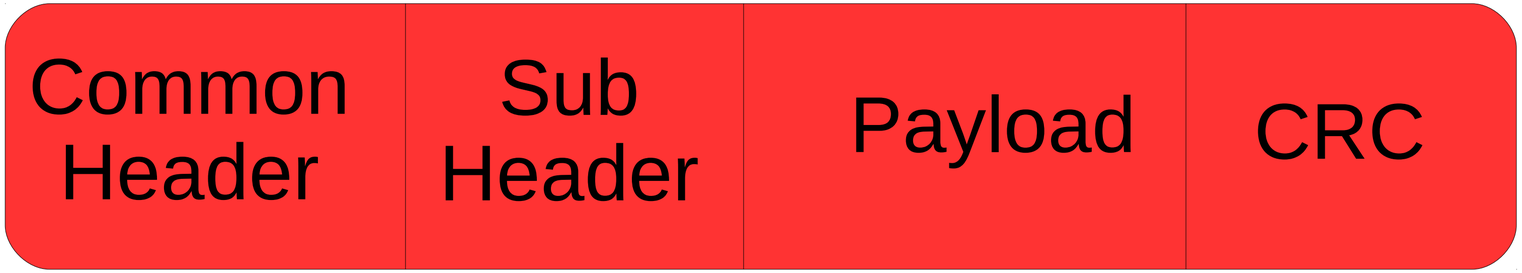
1. **Create checksum in session thread**
2. *Check in dump thread*
3. *Check when reading from network*
4. **Create before writing to Relay Log (if there is none)**
5. *Check when reading Relay Log*



3. Replication Events Checksums

5.6.2 DM

- Algorithm: CRC32. CRC appended at end of event:



- New configuration options:

  --binlog-checksum = NONE,CRC32 (default: NONE)

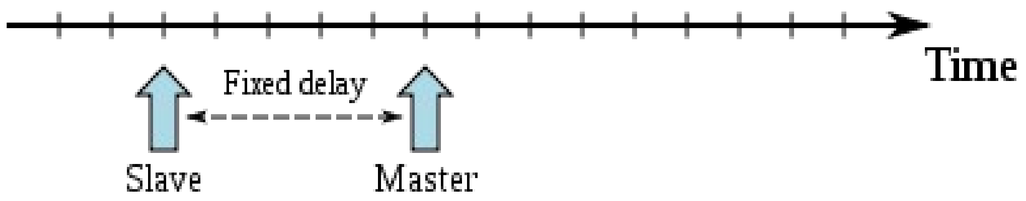
  --master-verify-checksum=0,1 (default: 0)

  --slave-sql-verify-checksum=0,1 (default: 1)

4. Time-Delayed Replication

5.6.2 DM

- Make replication slave lag behind the master
  - Protects against user mistakes
  - Test how lagging affects replication
- Slave waits a given number of seconds before applying the changes
  - Delays configured per slave
  - Implemented in the SQL thread layer



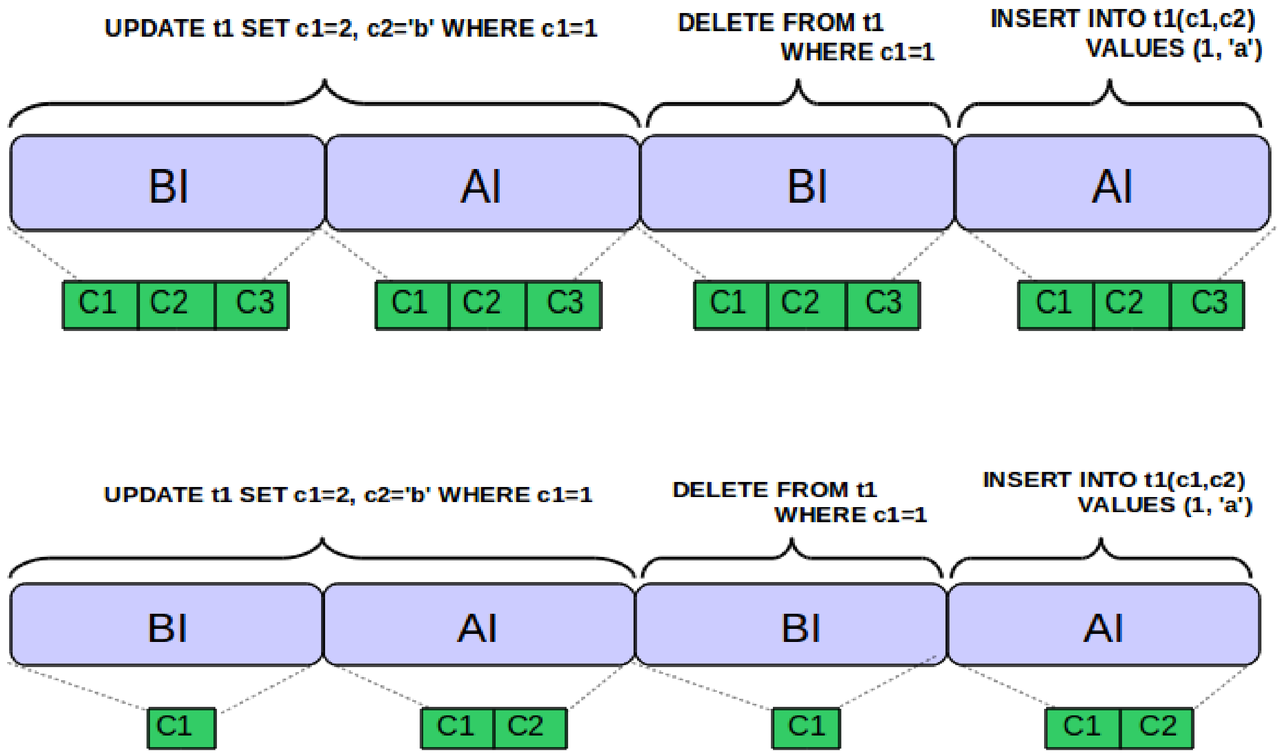
5. Row-based optimized logging

5.6.2 DM

- Server dynamically choose which columns to log for DELETE, UPDATE and INSERT row events:
  - **Minimal** – Primary key for BI and changed columns for AI
  - **Noblob** – No blobs columns when not needed
  - **Full** – All columns always

5. Row-based optimized logging

5.6.2 DM



6. Informational Log Events

5.6.2 DM

- Logs the query that originated the subsequent rows changes
- Shows up in mysqlbinlog and SHOW SLAVE STATUS output
- New option:

  --binlog-rows-query-log-events= ON|OFF
- New server variable:

  --binlog_rows_query_log_events= ON|OFF

7. Remote Binary Log Backup

5.6.2 DM

- mysqlbinlog can now retrieve and dump a remote MySQL binary log
- No need for remote login to retrieve master's binary logs, e.g. to setup a slave (no need for SSH access to MySQL host machine)

8. Server UUIDs

5.6.2 DM

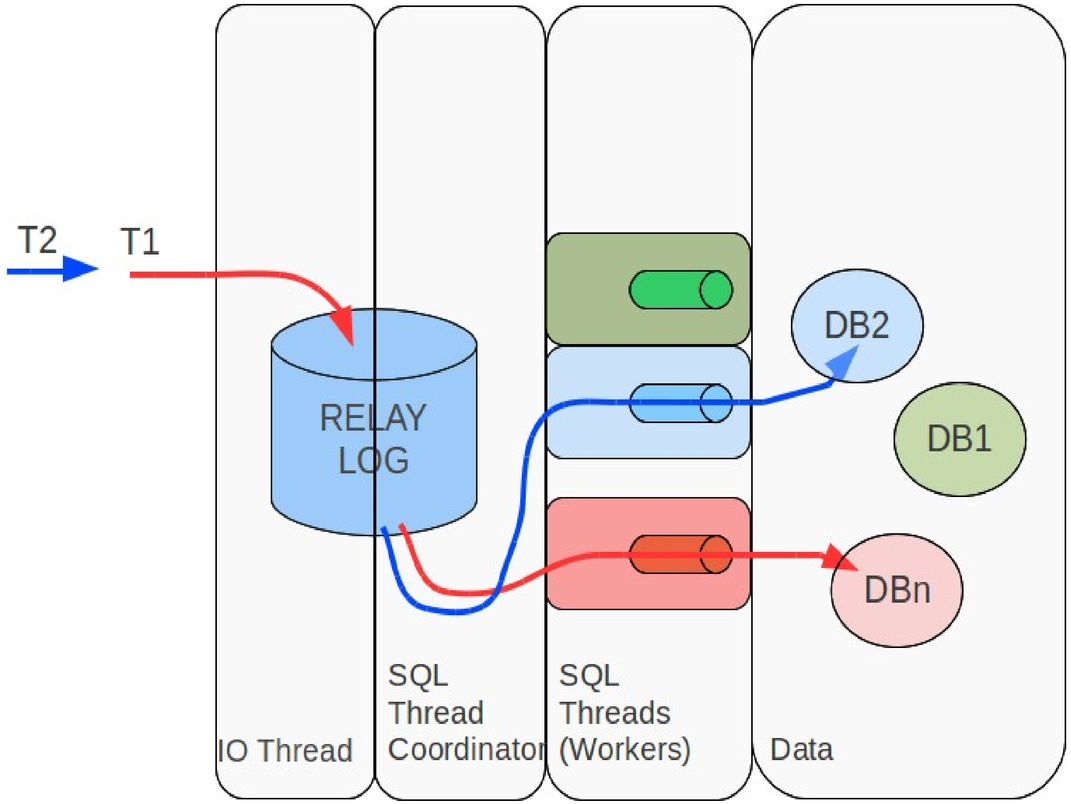
- Servers generate their own UUIDs and include them in the replication setup handshake protocol
- The UUIDs are exposed to the end user, enabling automatic tools, such as MySQL Enterprise Monitor, able to easily and reliably:
  - Replication topology auto-discovery
  - Topology reconfiguration auto-discovery, e.g. during fail-overs

labs.mysql.com

Multi-Threaded Slave

MySQL Labs

- Increased slave performance
- Workload applied in parallel:
  - Changes to each database are applied and committed independently
  - Automatic (serialized) recovery at restart
- Download from labs.mysql.com



Progress and Planning

Progress: Priority 1

1. **Options for writing full or partial row images in RBR**
   Optimize for performance, disk size, or functionality
2. **Replication-level checksums**
   Detect transmission or disk corruptions
3. **Transactional replication information**
   Automatically recover from a slave crash
4. **Informational events**
   Original statement for RBR, User and IP of statement executor, engine-dependent information
5. **Time-delayed replication**
   Protect against user mistakes
6. **Server UUIDs**
   Unique server ids making it easier to analyze replication topologies
7. **Remote backup of binary logs using mysqlbinlog tool**
   Retrieve the binary log from master
8. **Enhancements to Oracle Golden Gate Replication**
   Use Golden Gate to replicate MySQL to/from Oracle DBMS and other systems

Progress: Priority 1

1. ~~**Options for writing full or partial row images in RBR**~~ *MySQL 5.6*
   Optimize for performance, disk size, or functionality
2. ~~**Replication-level checksums**~~ *MySQL 5.6*
   Detect transmission or disk corruptions
3. ~~**Transactional replication information**~~ *MySQL 5.6*
   Automatically recover from a slave crash
4. ~~**Informational events**~~ *MySQL 5.6*
   Original statement for RBR, User and IP of statement executor, engine-dependent information
5. ~~**Time-delayed replication**~~ *MySQL 5.6*
   Protect against user mistakes
6. ~~**Server UUIDs**~~ *MySQL 5.6*
   Unique server ids making it easier to analyze replication topologies
7. ~~**Remote backup of binary logs using mysqlbinlog tool**~~ *MySQL 5.6*
   Retrieve the binary log from master
8. ~~**Enhancements to Oracle Golden Gate Replication**~~ *Golden Gate works with MySQL*
   Use Golden Gate to replicate MySQL to/from Oracle DBMS and other systems

Progress: Priority 2

9. **Multi-threaded slave for better performance** *labs.mysql.com*
   Faster slave since different threads apply different databases
10. **Performance schema for replication state**
    Possible to use queries instead of SHOW commands to read the state
11. **Preallocated binlog files**
    Improved performance by not having to append to files
12. **Group commit for the binary log (and some other scalability enhancements)**
    Improved performance by commit multiple transactions in one go
13. **Modular replication**
    Use different replication modules to replicate to/from a MySQL server
14. **Scriptable replication**
    Write your own plugin (e.g. replication filtering on data or statement type, extraction of data, pre-heating of caches)
15. **High resolution replication delay measurement**
    IO and SQL delay separately measured in milliseconds
16. **Universal Transaction ID (a.k.a. Global Transaction ID, Transactional Replication)**
    Identifiers enabling easy master failover

Other Developments

MySQL Workbench Utilities

- Easy-to-use command line solutions for administration and maintenance
  - Part of MySQL Workbench 5.2.31
  - Written in Python
  - Easily to extend using the supplied library

- How to get it
  - Download MySQL Workbench
    http://www.mysql.com/downloads/workbench/
  - Get the source
    https://launchpad/net/mysql-utilities

MySQL Enterprise Backup 3.5

- History and Progress tables
- Fully aligned with MySQL server development testing
- Easier installation out of box for all supported platforms

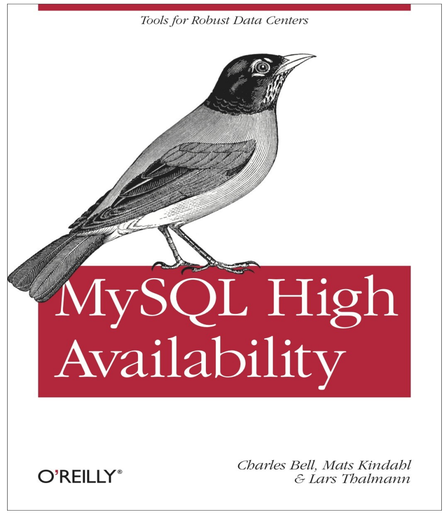
  (No Perl installation required)
- Optimized and reorganized internal code rewritten in C/C++

  (mysqlbackup)
- Fewer processes (No MySQL client process required)
- Improved error reporting

Tips

- **MySQL High Availability**
  Bell, Kindahl & Thalmann
  O'Reilly Media, July 2010

- **MySQL Support**
  www.mysql.com/contact



- *Book Signing, 12 Apr 3:50pm, O'Reilly booth in Expo hall*
- *MySQL Replication BOF, 13 Apr 6:00pm*

Lars Thalmann
Development Director, MySQL Replication, Backup & Connectors
lars.thalmann@oracle.com
www.larsthalmann.com

Disclaimer

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.


ORACLE®