

New Replication Features

MySQL 5.1 and MySQL 6.0/5.4

Dr. Lars Thalmann

Development Manager, Replication & Backup

lars@mysql.com

larsthalmann.blogspot.com

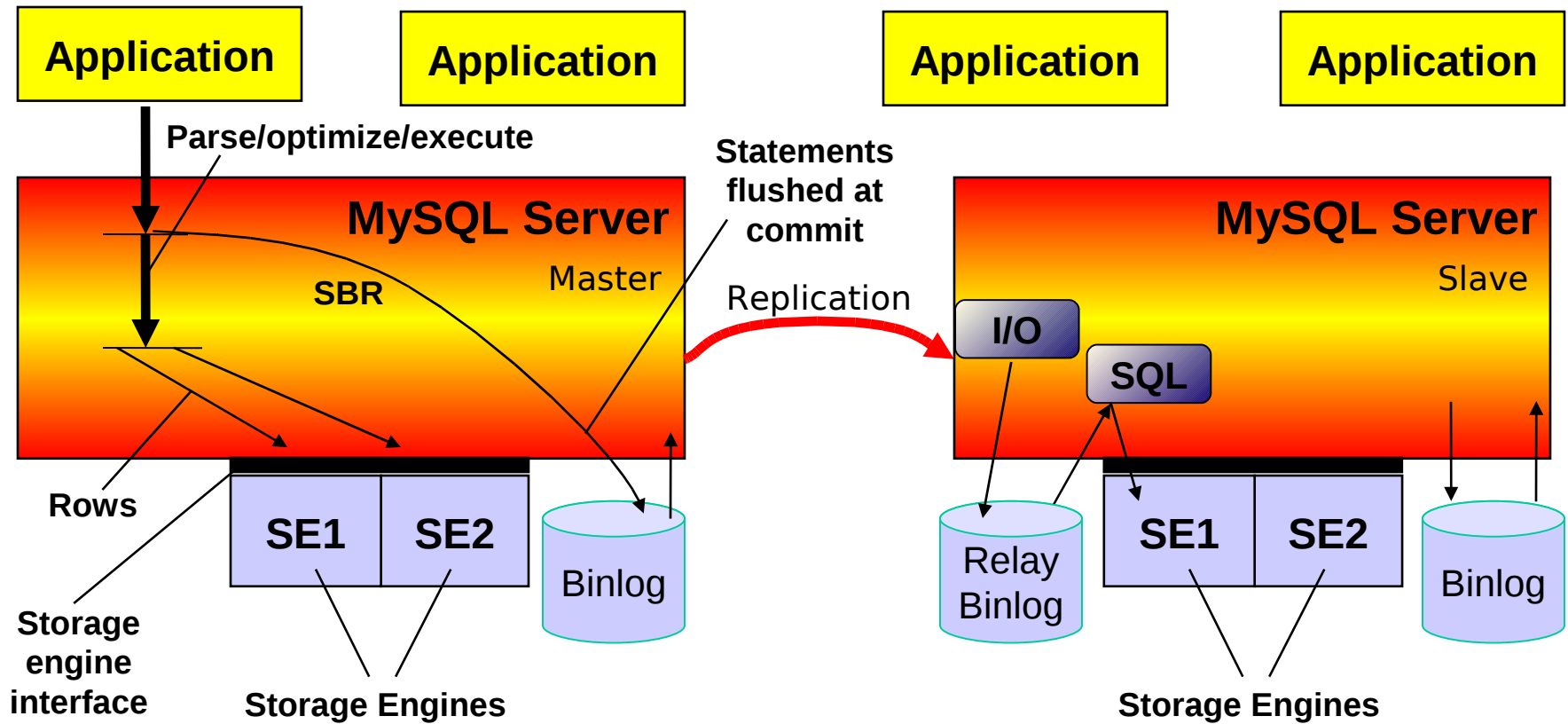
MySQL University

May 7th, 2009

Available in MySQL 5.1

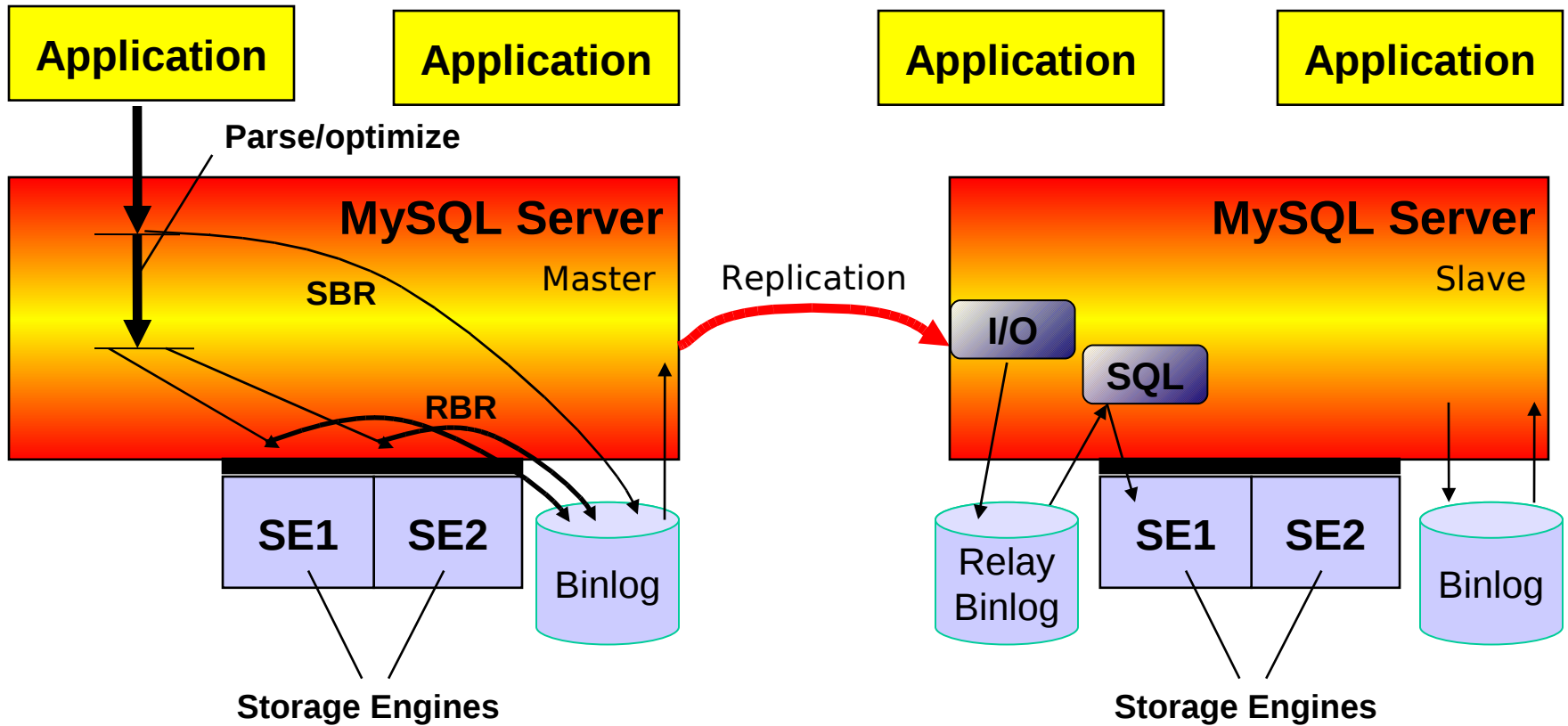
MySQL Replication Architecture

MySQL 4.0-5.0



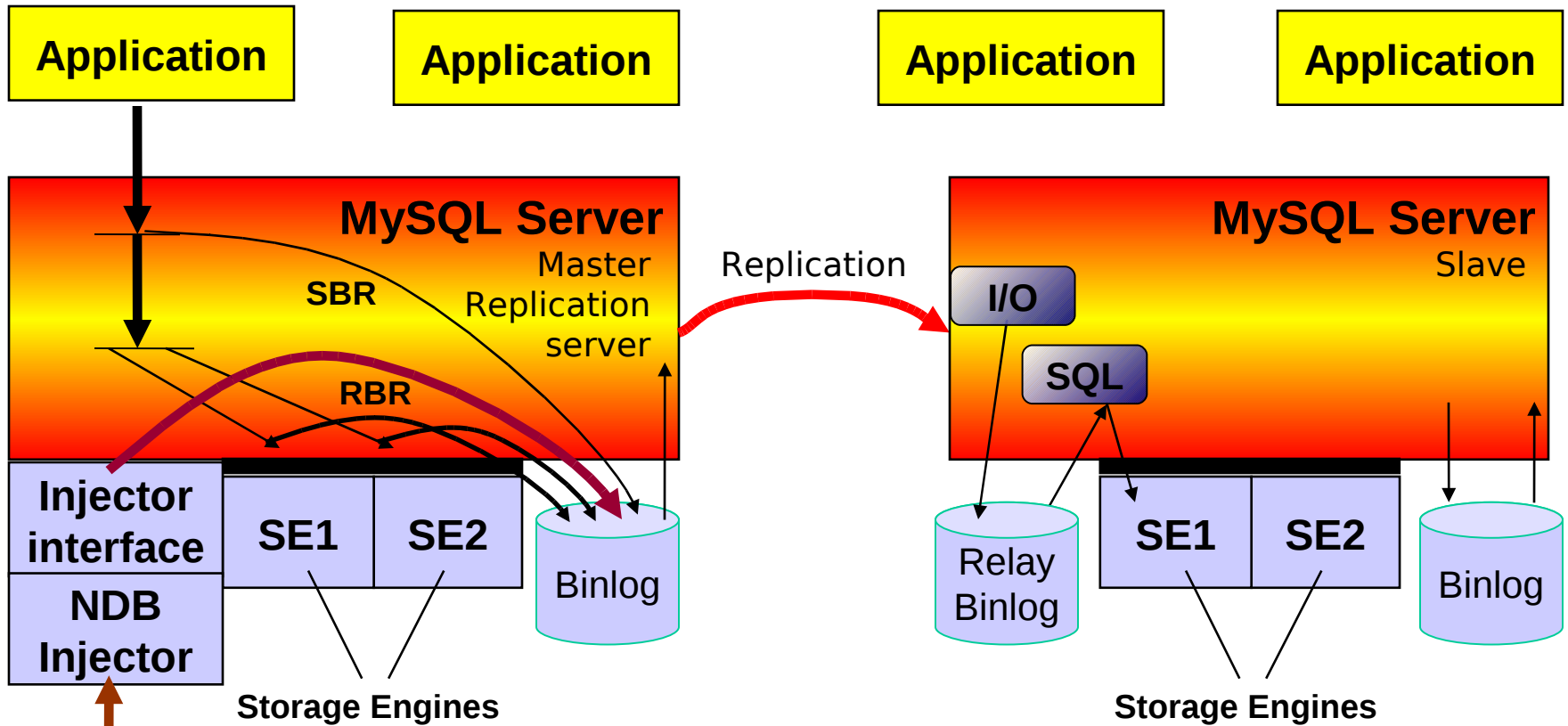
MySQL Replication Architecture

MySQL 5.1: Row-based replication (RBR)



MySQL Replication Architecture

MySQL 5.1: Injector interface/MySQL Cluster Replication



Row-based log from cluster data nodes

NUTS Test Driver

Writing MySQL tests in Perl

- “New Ubiquitous Testing Framework”
- Flexible, heterogeneous and (soon to be) distributed testbed for MySQL Replication.
- Infra-structure to conduct different system tests (stress, functional, configuration, performance and recovery tests).
- Perl is the core language used for extending the framework and writing test cases.
- Comprised of a test driver, executor, server deployer and test libraries.

Assertions, ie, no diff-based test execution verification.
Execution report based on TAP output.
Tests are simple perl modules.
Platform independent (relies on perl VM).
MySQL Version independent tests.

```
$ ./bin/nuts.pl --verbose --test replication

1..7
ok 1 - executing command CREATE DATABASE IF NOT EXISTS test
ok 2 - executing command USE test
ok 3 - executing command CREATE TABLE t (a int)
ok 4 - executing command INSERT INTO t VALUES (10)
not ok 5 - compare master slave contents

# Failed test 'compare master slave contents'
# at /home/acorreia/workspace.sun/repository.mysql/nuts-0.1/suites/samples/replication.pm line 34.
ok 6 - executing command DROP TABLE IF EXISTS t
ok 7 - executing command DROP DATABASE IF EXISTS test
Failed 1/7 subtests
[06:39:06]

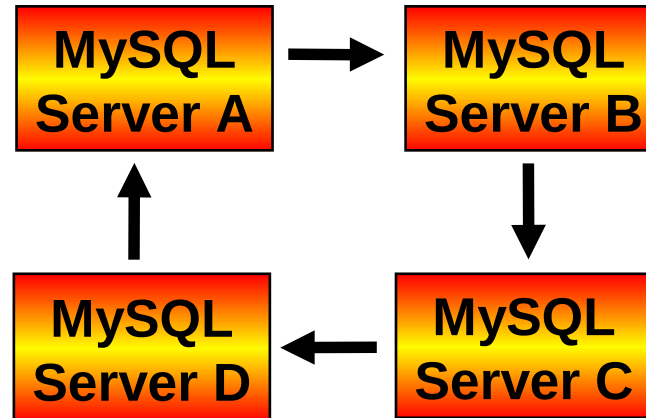
Test Summary Report
-----
samples::replication (Wstat: 0 Tests: 7 Failed: 1)
Failed test: 5
Files=1, Tests=7, 37 wallclock secs ( 0.24 usr 0.12 sys + 1.98 cusr 1.22 csys = 3.56 CPU)
Result: FAIL
```

Other Replication Changes

- Over **250 replication bugs** closed
- Refactored **position handling** on the slave (skipping events are now skipping complete transactions)
- MIXED binlogging mode safely replicating **unsafe (non-deterministic) statements** in row-based format
- **Warnings provided for unsafe statement execution** in STATEMENT binlog format
- Automatic **engine capability control** of logging formats
- **Event definitions** replicated as **slave-side-disabled**
- **Deprecated configuration options** “--master-host” etc

**Available in MySQL 6.0
and
Planned for MySQL 5.4**

Ignoring Servers in Circular Replication

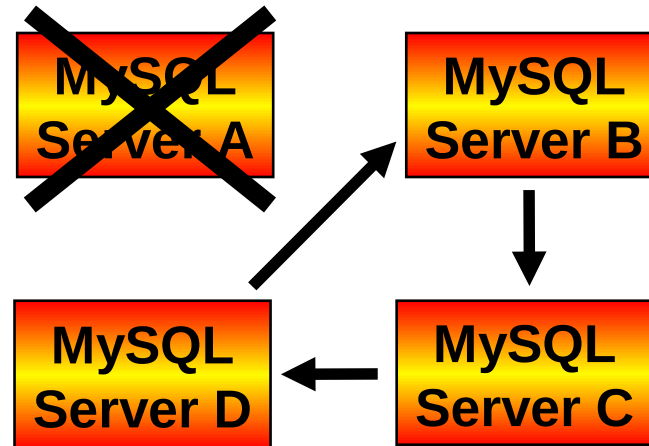


Circular replication

The originating server acts as the terminator of its own events:

When an event from **A** reaches **A** again, it is removed.

Ignoring Servers in Circular Replication



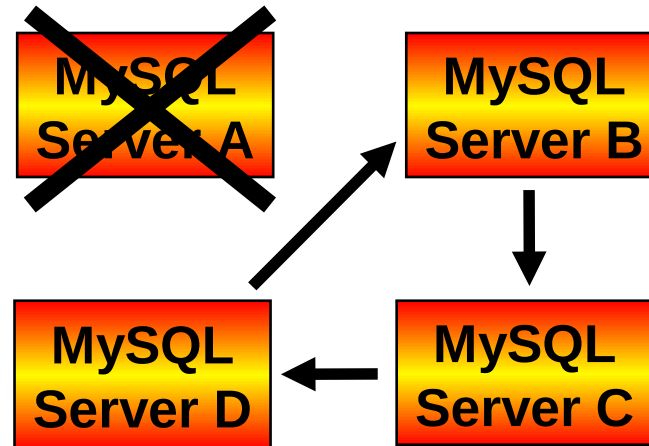
If server **A** is removed from the circle, then there can be events from **A** circulating indefinitely.

“Recently I had an interesting issue crop up. Due to an unfortunate migration incident in which involved master/master replication and not checking to see if replication was caught up, we ended up with an infinite replication loop of a number of SQL statements.”

-- Sheeri Cabral, Oct 2008,

<http://www.pythian.com/news/1299/mysqlbinlog-server-id-before-mysql-51-awk-to-the-rescue>

Ignoring Servers in Circular Replication

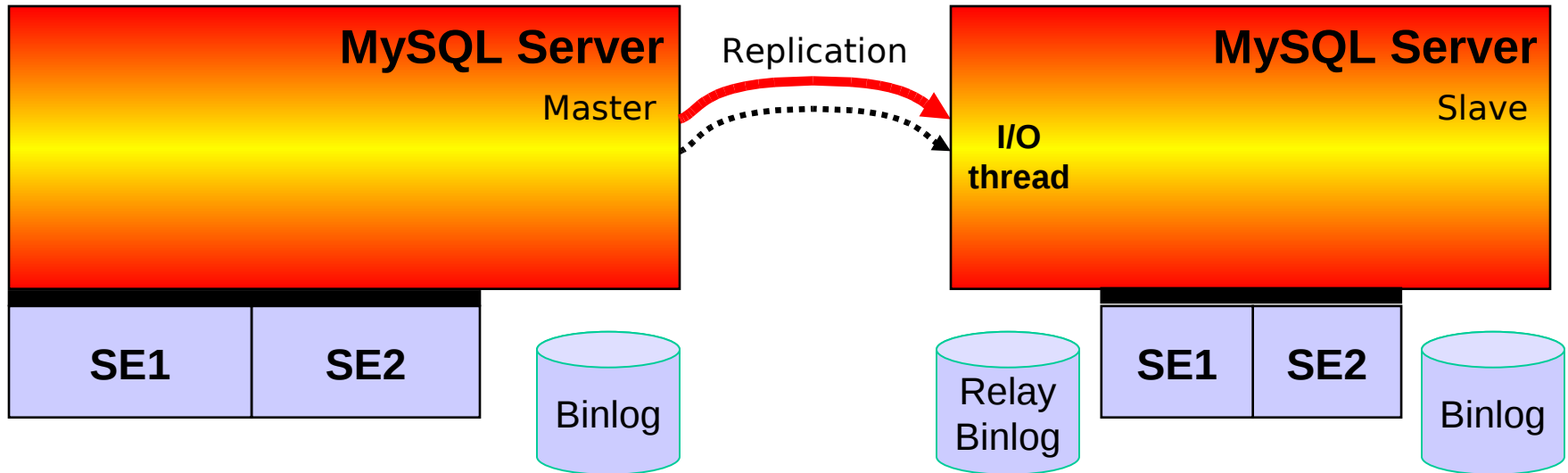


Circular replication

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=C ...  
          IGNORE_SERVER_IDS=(A)
```

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'
```

Slave Position Synchronization

If **sync_relay_log_info** = $X > 0$,
then the slave synchronizes its relay-log.info file to disk
after every X transactions.

A value of 1 is the generally the best choice.

Default is 0, which does not force any synchronization to disk by the MySQL server — in this case, the server relies on the operating system to flush the relay-log.info file's contents from time to time as for any other file.

Slave Position Synchronization

- **sync_master_info.** Slave synchronizes its master.info file to disk. Default is 0 (recommended in most situations).
- **sync_relay_log.** Slave synchronizes its relay log to disk.
 - Default value is 0, which means that the OS is responsible for syncing.
 - A value of 1 is the **safest choice**, because in the event of a crash you lose at most one statement or transaction from the relay log.
 - A value of 1 is also the **slowest choice** (unless the disk has a battery-backed cache, which makes synchronization very fast).

Automatic Relay Log Recovery

relay_log_recovery

Replication slave discards all unprocessed relay logs (and retrieves them from the replication master).

This should be used following a crash on the replication slave to insure that no possibly corrupted relay logs are processed.

The default value is 0 (disabled).

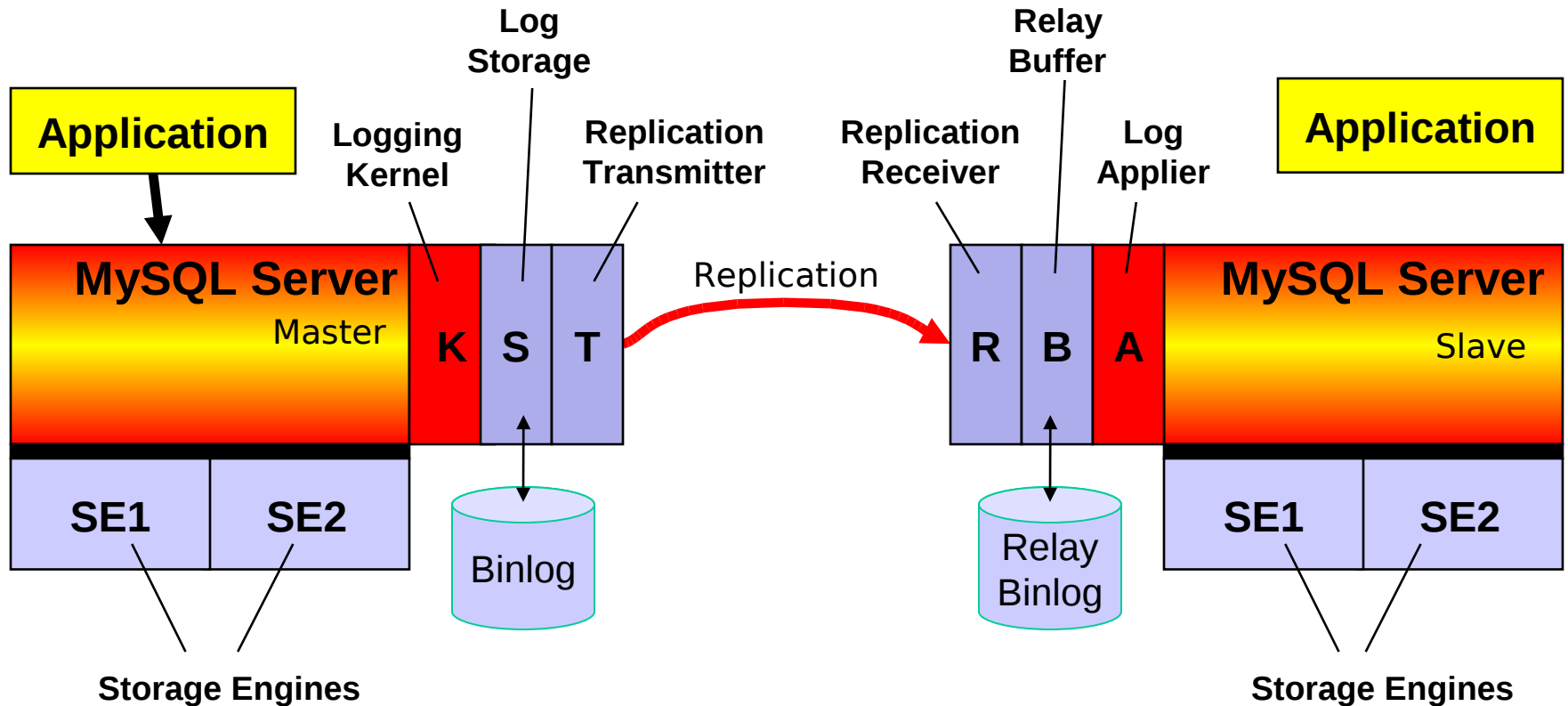
Binlog Performance Optimization

Mutex was taken on binary log ... even when writing to cache

- Not locking of mutex when writing event to cache
- Always write transactions to transaction cache
- Always write non-transactional statement to cache
- Flushed to disk at commit of transaction or statement

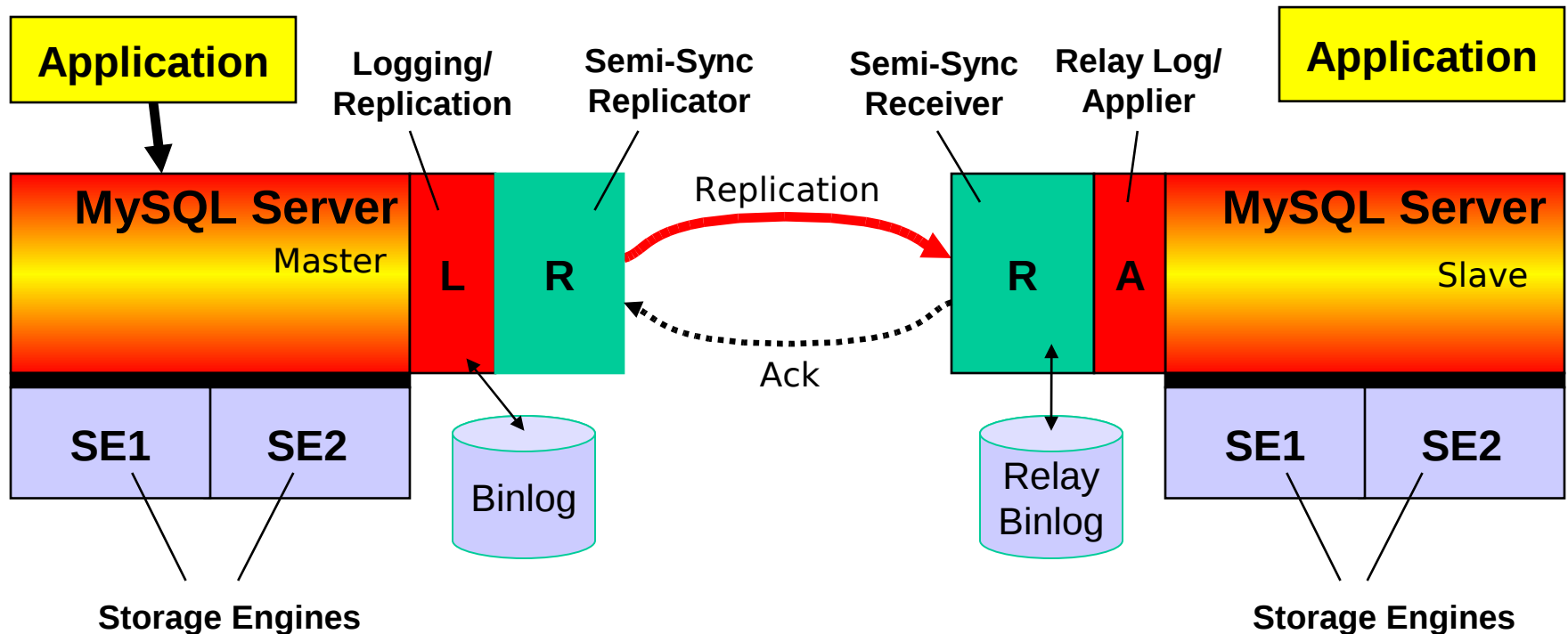
Preliminary benchmarks shows 5-14% performance improvement when binlog synchronization is on.

Pluggable Replication Architecture



Semi-synchronous Replication

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



***Available as two separate loadable components
for the master and the slave***

Why Semi-sync Replication?

- MySQL replication is asynchronous:
 - transactions can be lost forever if master cannot be recovered
 - when master failed, it may have transactions not replicated to slave, in which case we cannot fail-over to slave
- Provide HA by having slave acknowledge transaction before returning to user
- Fully synchronous replication would bring in more delay
- Semi-sync is a compromise solution

Design of Google Patch

- Make sure at least one slave receives the replication events before return from a transaction
- Wait for reply with a timeout
- Automatically disables semi-sync if timeout, and enables semi-sync when slave catch up

Google Patch

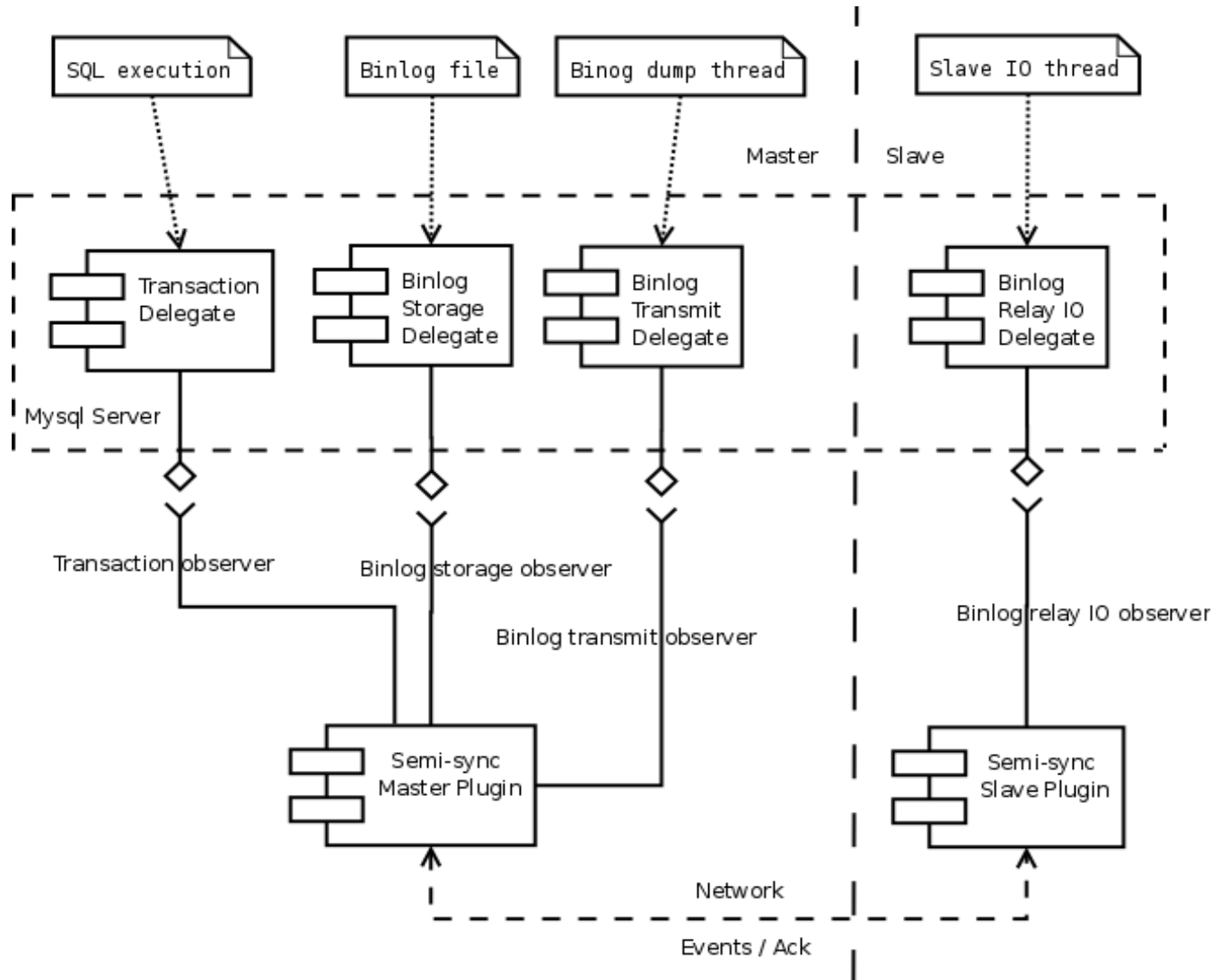
- Written for MySQL 5.0.37
- Implemented directly in the server source
- Dependent on InnoDB

Works Done on Google Patch

- Merged Google patch to MySQL 6.0
- Removed dependency on InnoDB
- Defined semi-sync replication interfaces
- Implemented semi-sync as replication plug-ins

Semi-sync Replication Architecture

A first step towards Pluggable Replication



Install Plug-ins and Play

- **On master**
 - `INSTALL PLUGIN 'rpl_semi_sync_master' SONAME 'libsemisync_master.so';`
 - `SET rpl_semi_sync_master_enabled=1;`
 - `SET rpl_semi_sync_master_timeout=1000;` (1s, default 10ms)
- **On slave**
 - `INSTALL PLUGIN 'rpl_semi_sync_slave' SONAME 'libsemisync_slave.so';`
 - `SET rpl_semi_sync_slave_enabled=1;`
 - `START SLAVE;`

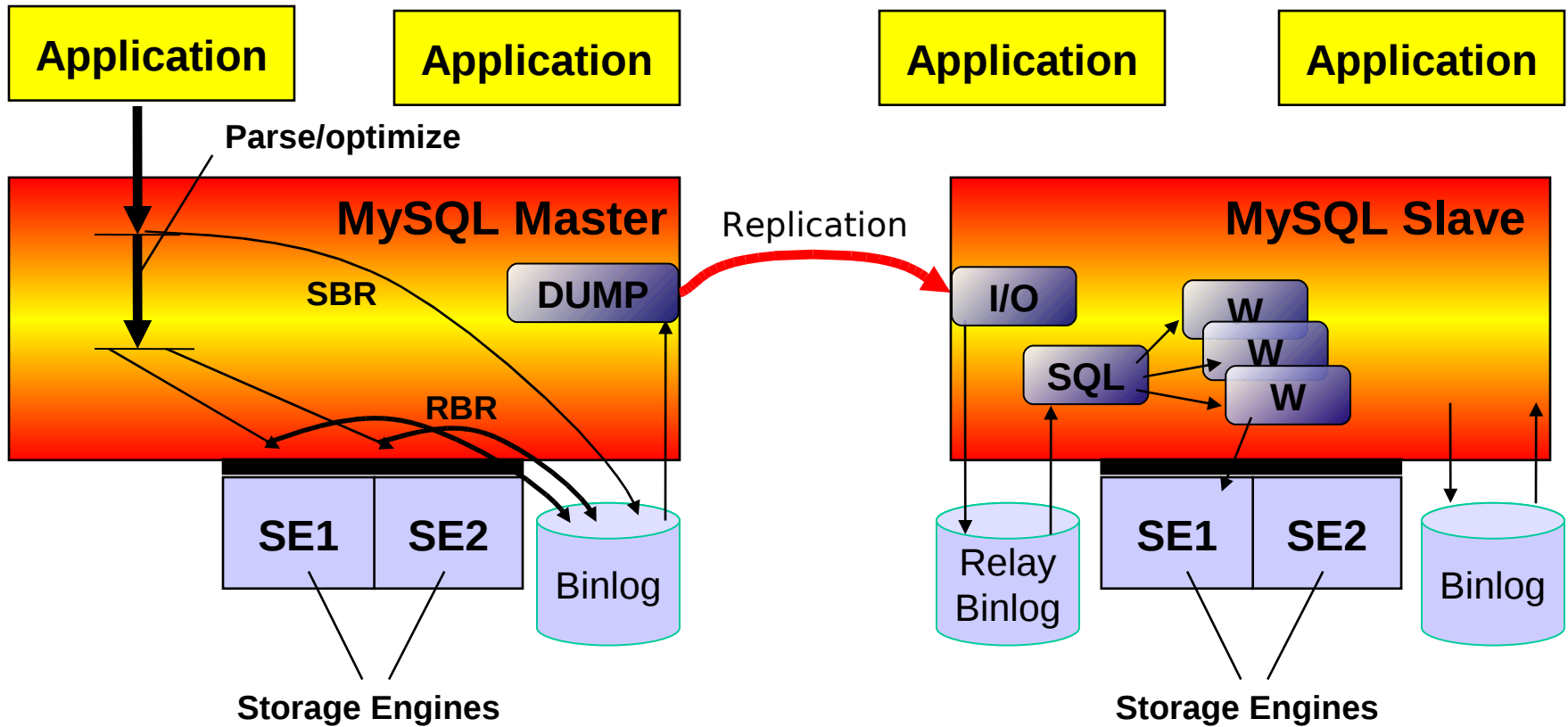
Checking semi-sync status

- **On master**
 - Rpl_semi_sync_master_status
 - Rpl_semi_sync_master_clients
 - Rpl_semi_sync_master_yes_tx
 - Rpl_semi_sync_master_no_tx
- **On Slave**
 - Rpl_semi_sync_slave_status

Feature Previews and Ongoing Projects

MySQL Replication Architecture

Multi-threaded Slave



Available as feature preview:

http://forge.mysql.com/wiki/Category:Software_Preview

Feature Preview

Time-delayed replication

<http://forge.mysql.com/wiki/ReplicationFeatures/DelayedReplication>

Provides:

Execution of replication events on slave is to be N seconds behind master

User interface:

CHANGE MASTER TO MASTER_DELAY= N

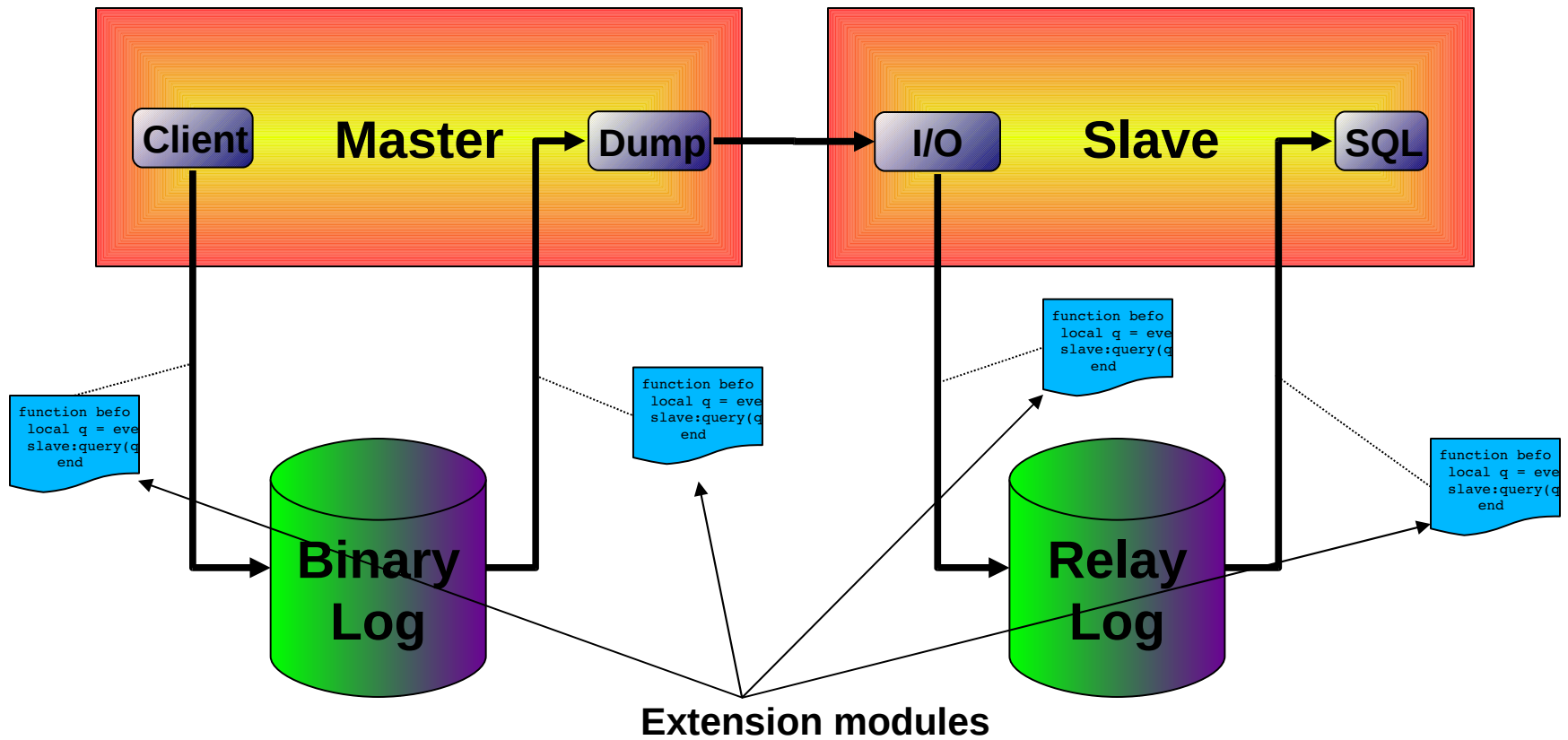
N is a non-negative less than MAX_ULONGLONG number of seconds to wait by the slave. (Attempts to set higher is rejected with error.)

Controlling of a delayed setup is via SHOW SLAVE STATUS "Seconds_behind_master"

Kudos: Kay Röpke, original patch; Sven, modifications

Scriptable Replication

<http://forge.mysql.com/wiki/ReplicationFeatures/ScriptableReplication>



Scriptable Replication

Paul Tuckfield's (Google/YouTube)
The "oracle" algorithm

```
module(..., package.seeall); require "luasql.mysql"

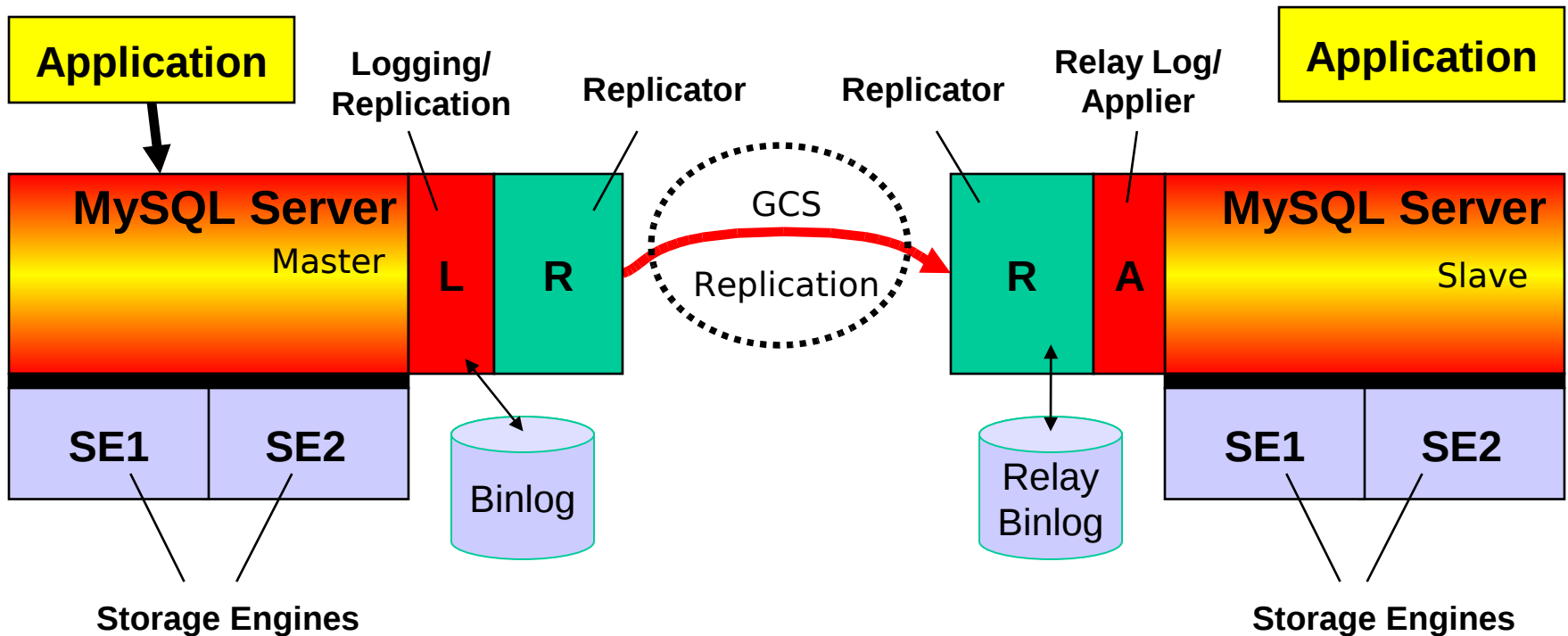
pattern = {
  ["UPDATE%s+(%w+).*%s(WHERE.*)"] = "SELECT * FROM %1 %2",
  ["DELETE%s+FROM%s+(%w+).*%s(WHERE.*)"] = "SELECT * FROM %1 %2",
}

env = luasql.mysql()
con = env:connect("test", "root", "", "localhost", mysql.port)

function before_write(event)
  local line = event.query
  if not line then return end
  for pat, repl in pairs(pattern) do
    local str = string.gsub(line, pat, repl)
    if str then con:execute(str); breakend
  end
end
```

Pluggable Replication Architecture

Synchronous Replication



More Information

<http://dev.mysql.com/replication/>

Dr. Lars Thalmann

Development Manager, Replication & BackupTechnology
lars@mysql.com

Bonus Slides

Semi-sync Master Plug-in

- Transaction
 - Wait for semi-sync slave ACK after commit but before return to user
- Binlog storage
 - report the binlog file name and position after the binlog events has been flushed to disk
- Binlog transmit
 - Flag the event that needs a reply, ACK from slave are done by a separate connection

Semi-sync Slave Plug-in

- Binlog relay IO
 - Send ACK to master if event is flagged after written event to relay log

Using Semi-sync Preview Build

- Install the latest MySQL 6.0
- Download corresponding preview build, only Linux supported now
 - http://downloads.mysql.com/forge/replication_preview
- Unpack it to the `lib/mysql/plugin` directory of your MySQL installation

Using the Semi-Sync Source Code

- Install the latest MySQL 6.0
- Branch the code with bzip
 - <https://code.launchpad.net/~hezx/mysql-server/semi-sync-replication>
- Run `./configure --prefix=<path to mysql installation>`
- Run `make`
- Run `make install`