
ORACLE®

ORACLE®

MySQL & PHP

Johannes Schlüter
MySQL Engineering

Johannes

Selbsternanntes Urgestein der Münchner PHP UG

```
From:       Johannes Schlueter <php@schlueters.de>
To:         phpuser-Muenchen
            <phpuser-muenchen@kbx7.de>
Subject:    [phpuser-muenchen] Re: Weihnachtsfeier
            php User München
Date:       Fri, 13 Dec 2002 14:13:41 +0100
```

Johannes?

```
$ php -r 'phpcredits();' | grep Johannes
```

CLI => Edin Kadribasic, Marcus Boerger, Johannes Schlueter, Moriyoshi Koizumi, Xinchen Hui

MySQL driver for PDO => George Schlossnagle, Wez Furlong, Ilia Alshanetsky, Johannes Schlueter

MySQLnd => Andrey Hristov, Ulf Wendel, Georg Richter, Johannes Schlüter

Reflection => Marcus Boerger, Timm Friebe, George Schlossnagle, Andrei Zmievski, Johannes Schlueter

tokenizer => Andrei Zmievski, Johannes Schlueter

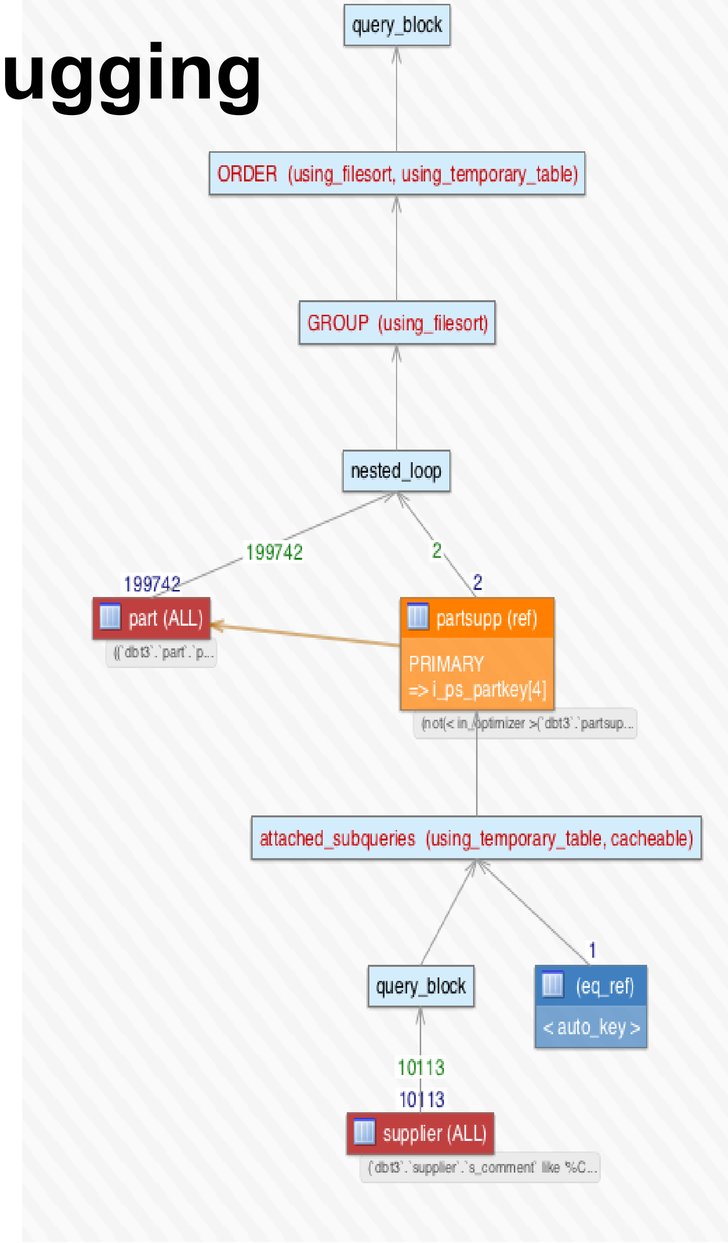
MySQL

MySQL Optimizer

- Subquery Optimizations
- File sort optimizations with small limit
  - 4X better execution time – 40s to 10s
- Index Condition Pushdown
  - 160X Better execution time – 15s to 90ms
- Postpone Materialization of views/subqueries in FROM
  - 240X better execution time for EXPLAIN – 8m to 2s
- Batched Key Access and Multi Range Read
  - 280X Better execution time – 2800s to 10s

MySQL Optimizer – Diagnostics and Debugging

- EXPLAIN
  - INSERT, UPDATE, and DELETE
  - JSON format for better readability
- Persistent Optimizer Statistics - InnoDB
- Optimizer Traces



Performance Schema Improvements

- Statements/Stages
  - Most resource intensive queries? Where do they spend time?
- Table/Index I/O, Table Locks
  - Which application tables/indexes cause the most load or contention?
- Users/Hosts/Accounts
  - Which application users/hosts/apps consume the most resources?
- Network I/O
  - Network loaded? How long do sessions idle?
- Summaries
  - Aggregate stats grouped by thread, user, host, account or object

dbahelper

- Views and stored procedures to work with INFORMATION_SCHEMA and PERFORMANCE_SCHEMA
- Maintained by Mark Leith

```
git clone git@github.com:MarkLeith/dbahelper.git
cd dbahelper
mysql -uroot -p < ./ps_helper_56.sql
```

- Also for MySQL 5.5 and 5.7

```
mysql> select * from wait_classes_global_by_avg_latency
       where event_class != 'idle';
+------------------+--------+-----------+-----------+-----------+------------+
| event_class      | events | tot_lat   | min_lat   | avg_lat   | max_lat    |
+------------------+--------+-----------+-----------+-----------+------------+
| wait/io/file     | 543123 | 44.60 s   | 19.44 ns  | 82.11 µs  | 4.21 s     |
| wait/io/table    | 22002  | 766.60 ms | 148.72 ns | 34.84 µs  | 44.97 ms   |
| wait/io/socket   | 79613  | 967.17 ms | 0 ps      | 12.15 µs  | 27.10 ms   |
| wait/lock/table  | 35409  | 18.68 ms  | 65.45 ns  | 527.51 ns | 969.88 µs  |
| wait/synch/rwlock| 37935  | 4.61 ms   | 21.38 ns  | 121.61 ns | 34.65 µs   |
| wait/synch/mutex | 390622 | 18.60 ms  | 19.44 ns  | 47.61 ns  | 10.32 µs   |
+------------------+--------+-----------+-----------+-----------+------------+
```

(output slightly modified to fit on slide)

```
mysql> select * from innodb_buffer_stats_by_table;
+---------------+--------------------+-----------+-----------+-------+--------+---- -+--------+
| object_schema | object_name        | allocated | data      | pages | hashed | old   | cached |
+---------------+--------------------+-----------+-----------+-------+--------+------+--------+
| InnoDB System | tinytiny/ttrss_e... | 41.95 MiB | 33.15 MiB |  2685 |   2685 | 2685 |  28595 |
| InnoDB System | tinytiny/ttrss_u... |  6.88 MiB |  4.67 MiB |   440 |    440 |  440 |  27510 |
| InnoDB System | piwik/piwik_arch... |  2.66 MiB |  1.36 MiB |   170 |    170 |  170 |   3505 |
| InnoDB System | piwik/piwik_arch... |  2.61 MiB | 764.9 KiB |   167 |    167 |  167 |   2201 |
```

(output slightly modified to fit, more or less, on slide)

InnoDB

INNODB

ORACLE

InnoDB in MySQL 5.5

- Performance and Scalability
  - Multiple buffer pool instances
  - Multiple rollback segments
  - Improved purge scheduling
  - Extended change buffering with delete buffering and purge buffering
  - Native async I/O support on Linux
  - Improved log sys mutex
  - Separate flush list mutex
  - Windows performance improvements
  - Performance schema for InnoDB

InnoDB in MySQL 5.6

- Performance and Scalability
  - Split the kernel mutex
  - Multi threaded purge
  - Use rw_locks for page_hash
  - Add 'page_cleaner' thread to flush dirty pages
  - Ibuf merge rate improvement
  - Configurable data dictionary cache
  - InnoDB persistent optimizer statistics
  - Read Only Transactions

InnoDB in MySQL 5.6

- Monitoring & Diagnostics
  - InnoDB Information Schema Metrics Table
  - Information schema system tables for InnoDB
  - Information schema table for InnoDB buffer pool
  - InnoDB: report all deadlocks

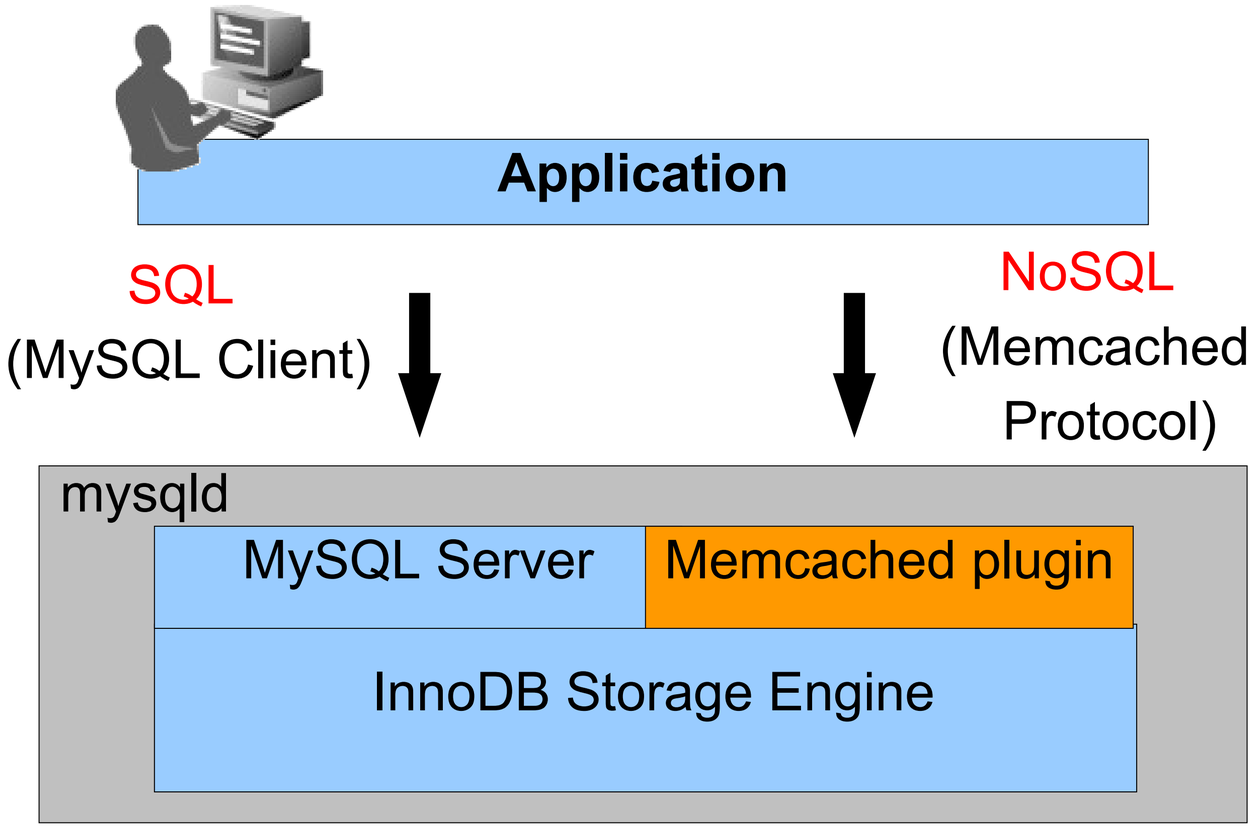
InnnoDB Disk Improvements

- SSD optimizations
- Configurable page size (4K, 8K)
- Export/Import per table tablespaces
- Separate InnoDB Undo tablespaces
- Read Only media support
- Compression improvements
- Large (over 4GB) redo logs support

Online ALTER TABLE

- ADD [FULLTEXT] INDEX
- DROP INDEX
- ADD/DROP FOREIGN KEY
- ADD/DROP COLUMN

Key-value Access to InnoDB Data



- Fast, simple access to InnoDB
  - via Memcached API
  - Use existing Memcached clients
  - Bypasses SQL parsing
- NotOnlySQL access
  - For key-value operations
  - SQL for rich queries, JOINs, FKs, etc.
- Implementation
  - Memcached plug-in to mysqld
  - Memcached mapped to native InnoDB API
  - Shared process for ultra-low latency

InnoDB FullText Search

- Support all query types supported by MyISAM:
  - Natural language search
  - Query expansion
  - Boolean search
- Plus
  - Proximity search
  - Create full-text index with parallel tokenization and parallel sort

InnoDB FT: Parallel Indexing and Tokenization

Data Size : Approx 2.7GB, 1 million row , 238 million word
Platform: Linux x86 64 bit, 8 cores , 32 GB RAM

	innodb_ft_sort_pll_degree			
	1	4	8	16
InnoDB	7 min 48.12 sec	5 min 12.06 sec	4 min 10.95 sec	3 min 33.40 sec
MyISAM	Around 11 min 30 sec (innodb_ft_sort_pll_degree does not affect MyISAM)			

Automatic Recovery

Crash-safe master – automatic binary log file trimming

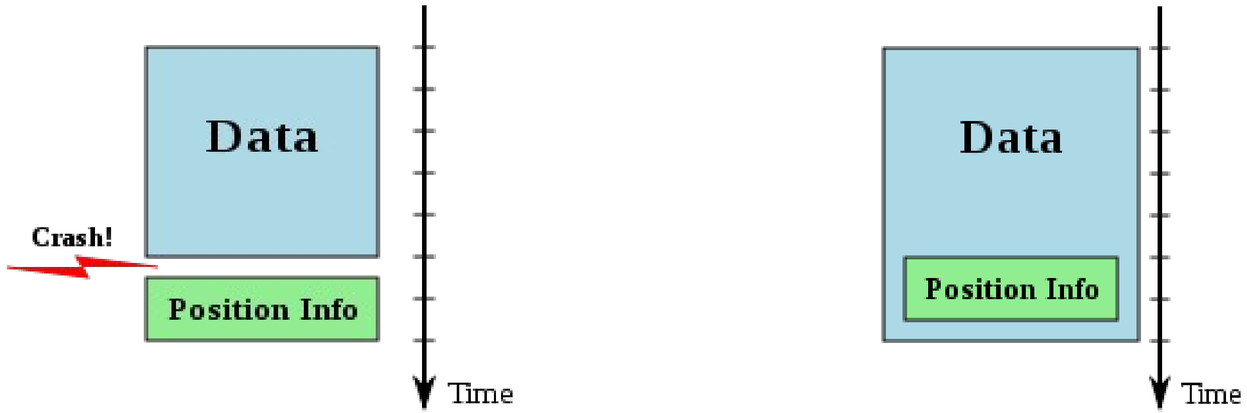
- If the master crashes, binary log will recover automatically from incomplete file flushes.
- On recovery:
  - The active binary log is scanned and any log corruption is detected;
  - Invalid portion of the binary log file is discarded and the file is trimmed.

Crash-safe slave - Slave Info Tables

- Transactional positioning – InnoDB based by default.
- Protection against slave crashes:
  - Automatic recovery.
- Possibility to issue SELECTs on slave information.
  - Possibility to code multi-source replication in pure SQL.
- Automatic conversion between files and tables on startup.

Crash-safe slave - Slave Info Tables

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



Automatic Switchover/Failover

Automatic Switchover/Failover: Global Trans Ids

- Promote any slave to be the new master.
- Automatically pick replication stream correctly:
  - Slave auto-positioning;
  - Slave automatically skips transactions that already exist in the server.

Automatic Switchover/Failover: Global Trans Ids

- Two major parts:
  1. Logical Identifier: (*Server UUID, Transaction Number);*
  2. Server State: set of applied transactions.

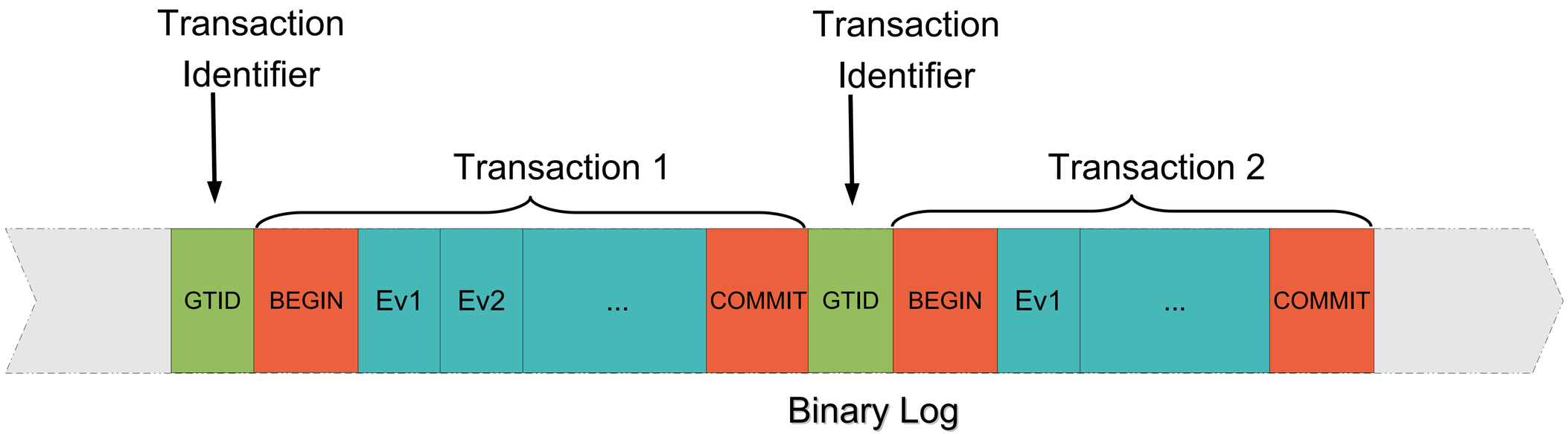
- Designed to work with transactional engines (InnoDB).

Automatic Switchover/Failover: Global Trans Ids

- The identifier:
  - Abstracts physical positions into logical identifiers – human friendly;
  - It is uniquely assigned when the transaction executes;
  - It is preserved when the transaction is re-applied.
- The server state:
  - Enables slave auto-positioning;
  - Failover facilitator – reduced administration overhead;
  - Protects against undesirable transaction re-execution.

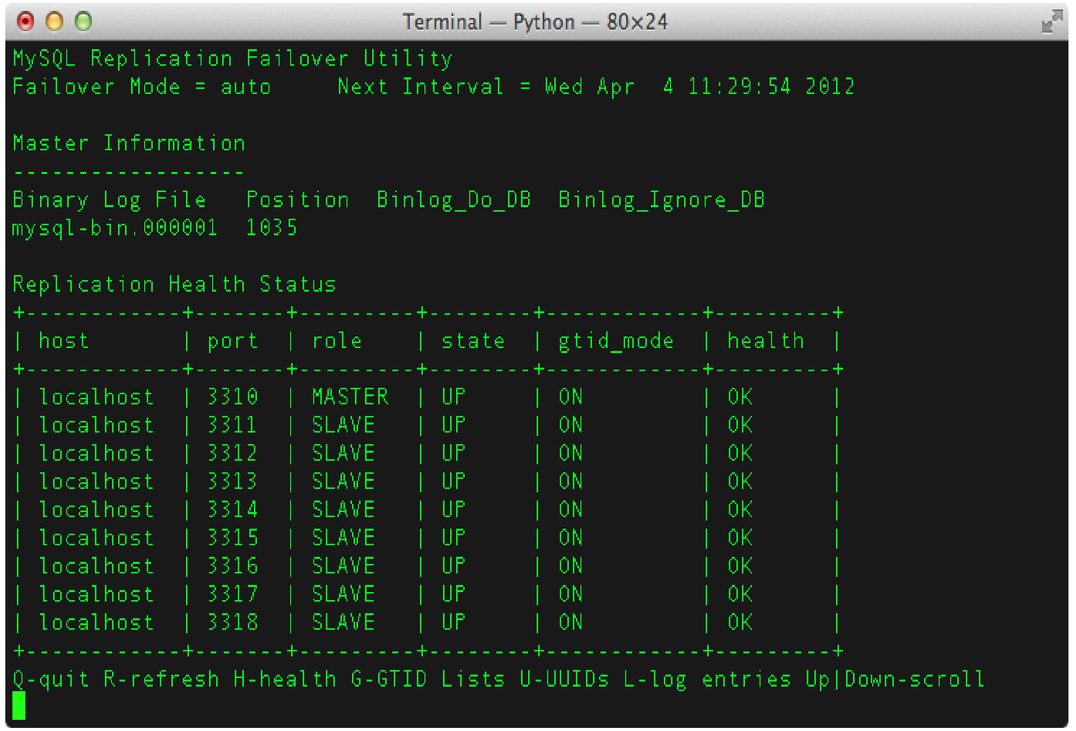
Automatic Switchover/Failover: Global Trans Ids

- Persistence:
  - It is written to the binary log.
  - Precedes a collection of events that comprise a transaction.



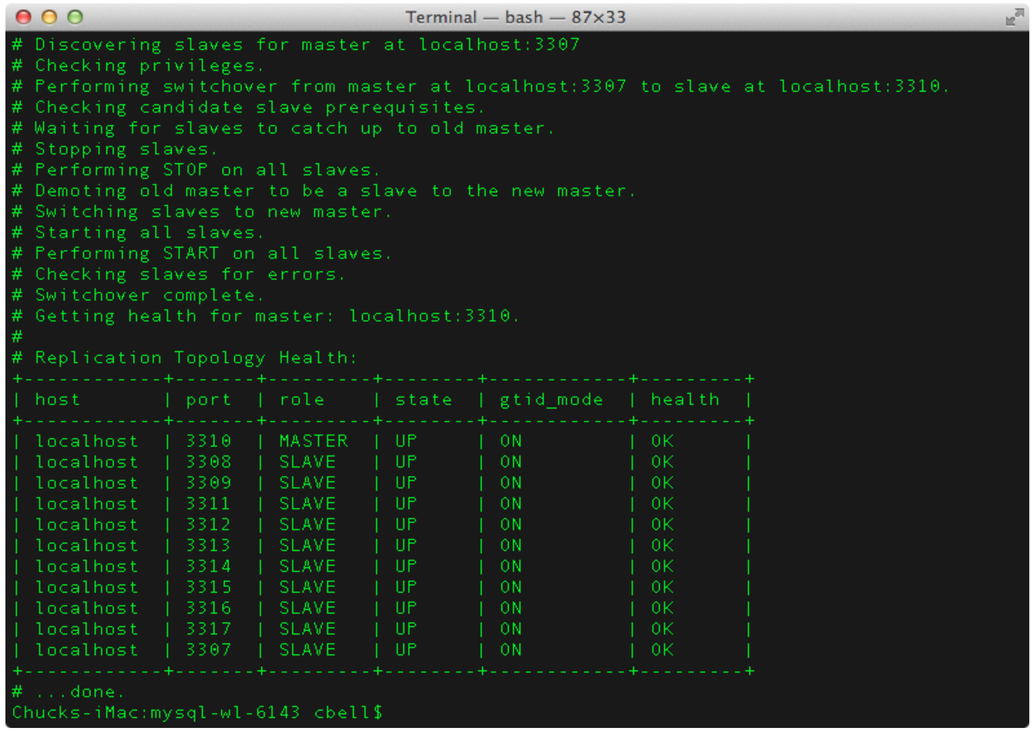
Automatic Switchover/Failover: mysqlfailover

- Check and report health at specific intervals in seconds.
- Automatic failover.



Automatic Switchover/Failover: mysqlrpladmin

- General replication administration utility: start, stop, topology health, elect, failover, switchover, gtid.
- On-demand failover or switchover.



MySQL Utitlities

- A collection of Python utilities for managing MySQL databases
- Originally MySQL Workbench Plugin
- Available under the GPLv2 license
- Written in Python
- Python library to grow solutions for common administrative problems

MySQL Utilities

- Easily administer MySQL servers from the command line
  - mysqldbcompare – compare databases
  - mysqldbcopy – copy databases between servers
  - mysqlfailover – Automatic fail-over
  - mysqlrpladmin – General replication administration utility
  - mysqlrplshow – show a graph of your topology
  - mysqlreplicate – setup replication
  - mysqlrplcheck – check replication configuration
  - ...
- Build your own tools on top of the core of the library

MySQL Fabric

An integrated environment for managing a farm of MySQL server supporting high-availability and sharding.

http://labs.mysql.com

MySQL Fabric

- “Farm” Management System
- Distributed
- High-Availability
- Sharding
- Procedure Executor
- Extensible

- Written in Python
- Early alpha
  - Long road ahead
- You can participate
  - Suggest features
  - Report bugs
  - Contribute patches
- MySQL 5.6 is focus

MySQL Fabric

- Decision logic in connector
  - Reducing network load
- Support Transactions
  - API to provide sharding key
- Global Updates
  - Global Tables
  - Schema updates
- Procedure Executor
- Shard Multiple Tables
  - Using same key
- Sharding Functions
  - Range
  - (Consistent) Hash
- Shard Operations
  - Using built-in executor
  - Shard move
  - Shard split

PHP: mysqlnd_ms

- Plugin for PHP for doing load-balancing
- ms originally meant “master / slave”
- http://dev.mysql.com/doc/refman/5.5/en/apis-php-book.mysqlnd-ms.html
- Uses GTIDs to ensure sending users to a slave which received data we wrote before
- 1.6.0-alpha added MySQL Fabric Support

mysqlnd_ms 1.6.0 configuration

```
{
  "test" : {
    "fabric":{
       "hosts": [ {
         "host": "localhost",
         "port": 8080 } ]
    }
  }
}
```

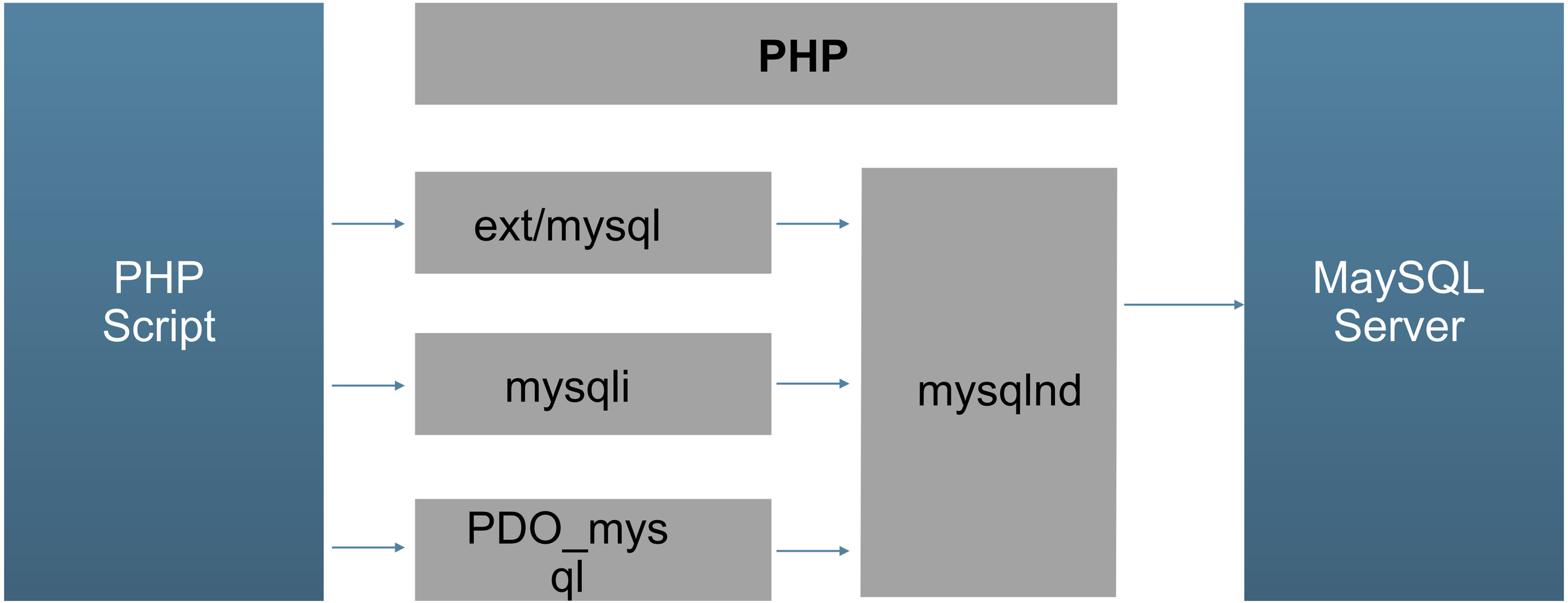
mysqlnd_ms 1.6.0

```
<?php
$c = new mysqli("test", "root", "", "test");

echo "Creating global table:\n";
mysqlnd_ms_fabric_select_global($c, "test.fabrictest");
$c->query("CREATE TABLE fabrictest (id INT NOT NULL)");

echo "Inserting with ID 10:\n";
mysqlnd_ms_fabric_select_shard($c, "test.fabrictest", 10);
$c->query("INSERT INTO fabrictest VALUES (10)");
?>
```

PHP's MySQL Architecture



API Choice

- mysqli
  - Support for all MySQL features
  - Best support / stability
  - Integration with existing applications / environments
- PDO_MYSQL
  - Simple applications supporting multiple databases
    - API-compatibility is often not enough, though
  - Integration with existing applications / environments

Three kinds of PHP Users

Users of existing Applications	Quick'n'Dirty Applications	Framework-based Applications
Take existing applications (i.e. Wordpress, phpBB, moodle, ...) Little customization only Often hard to scale	Ad-hoc developed applications Code often mixture of PHP with embedded SQL etc. Hard to scale/adapt to new situations	Custom applications built on i.e. Symfony, Zend Framework or CakePHP Built on DB abstraction Layers Abstractions hide full power of MysQL

On-Going Demands

Improve Security!

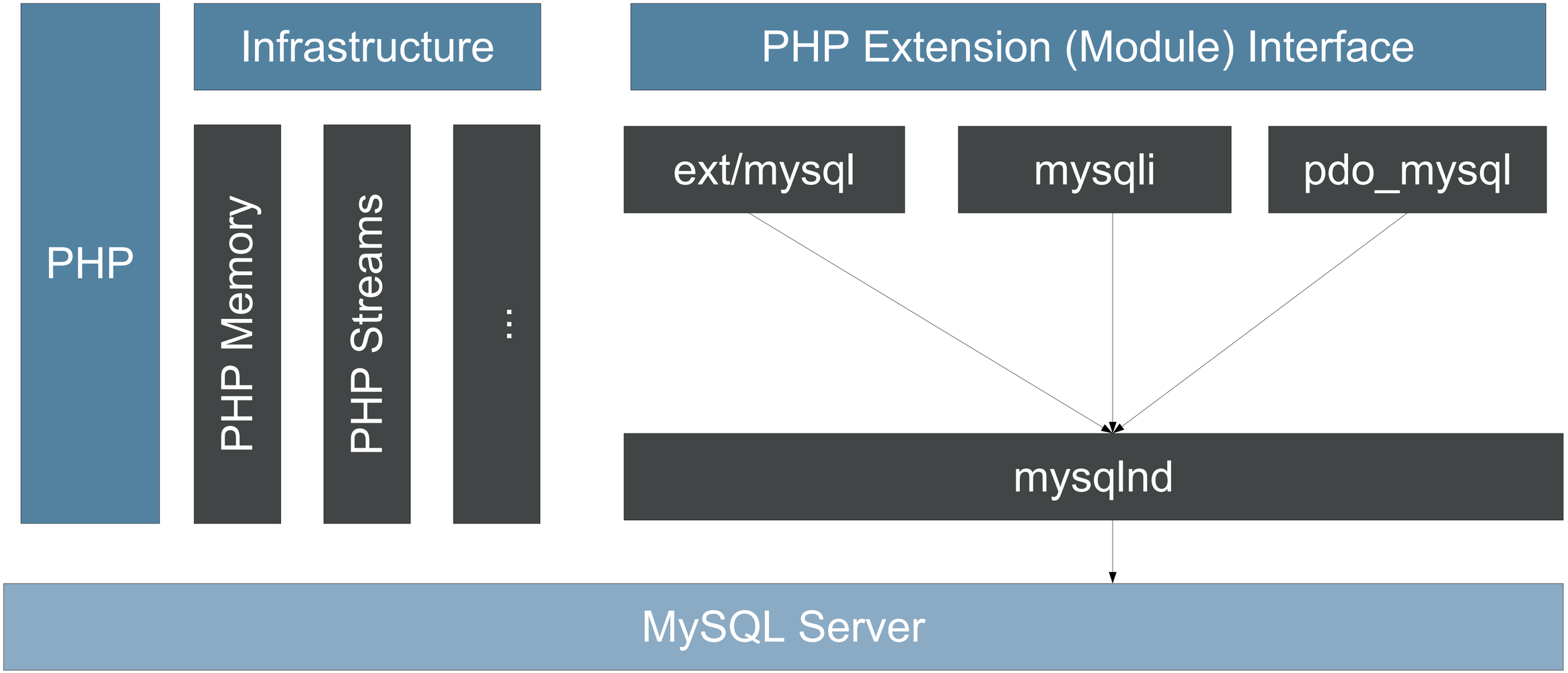
Better performance!

More Scalability!

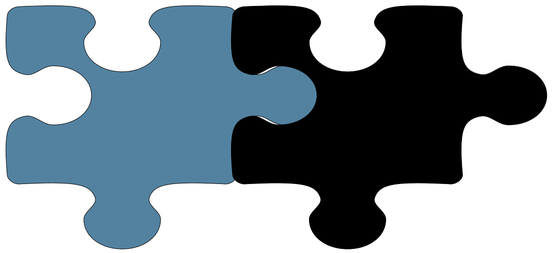
Higher Availability!

Refactor the application!

Solving these requests



mysqlnd in depth



mysqlnd_result:fetch_row

mysqlnd:query

mysqlnd:prepare

mysqlnd

mnd::allocate

mysqlnd_net:read_result

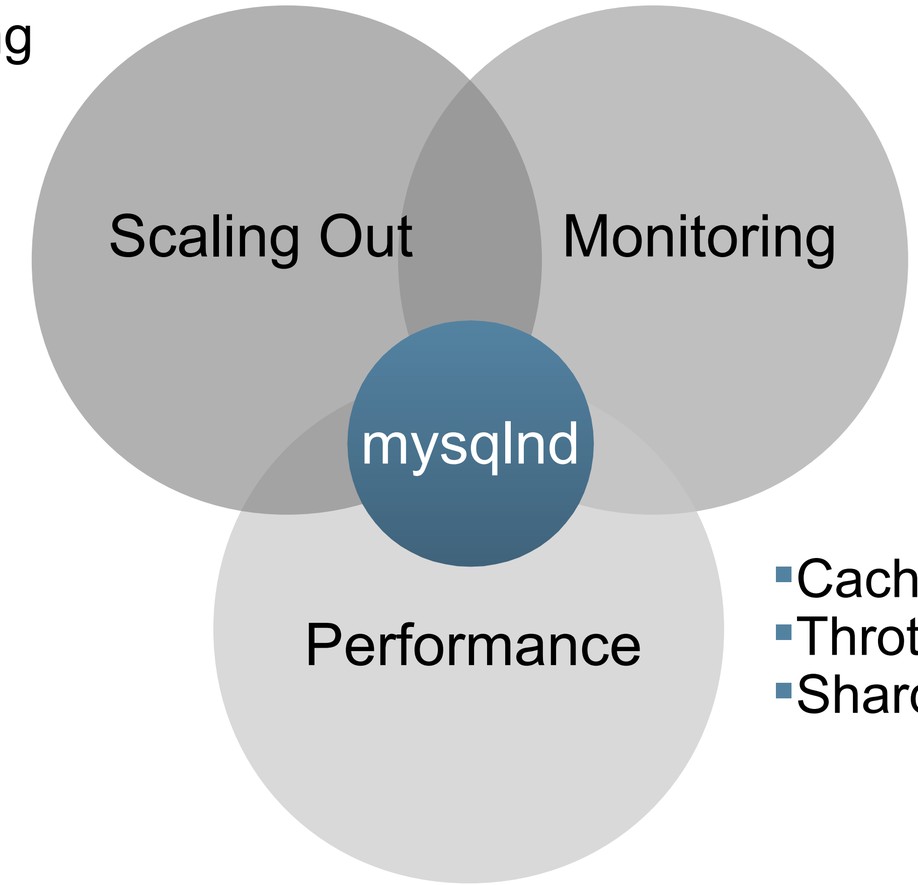
mysqlnd_net:send

myslqnd Plugins

- Loaded as regular PHP extension
  - Written in C
- Hook into mysqlnd in any level
  - From high-level APIs (query, prepare fetch_row, etc)
  - To low-level (network IO, memory allocation)
- Can be 100% transparent
- Can provide userspace APIs which can work with ext/mysql, mysqli and PDO_mysql instances

What can Plugins be used for?

- Read/Write Splitting
- Failover
- Round-Robin

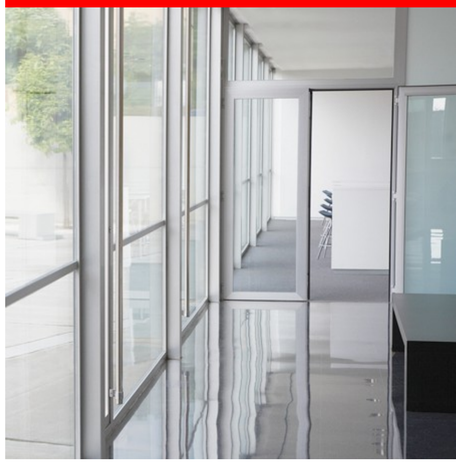


- Query Logging
- Query Analysis
- Query Auditing

- Caching
- Throttling
- Sharding

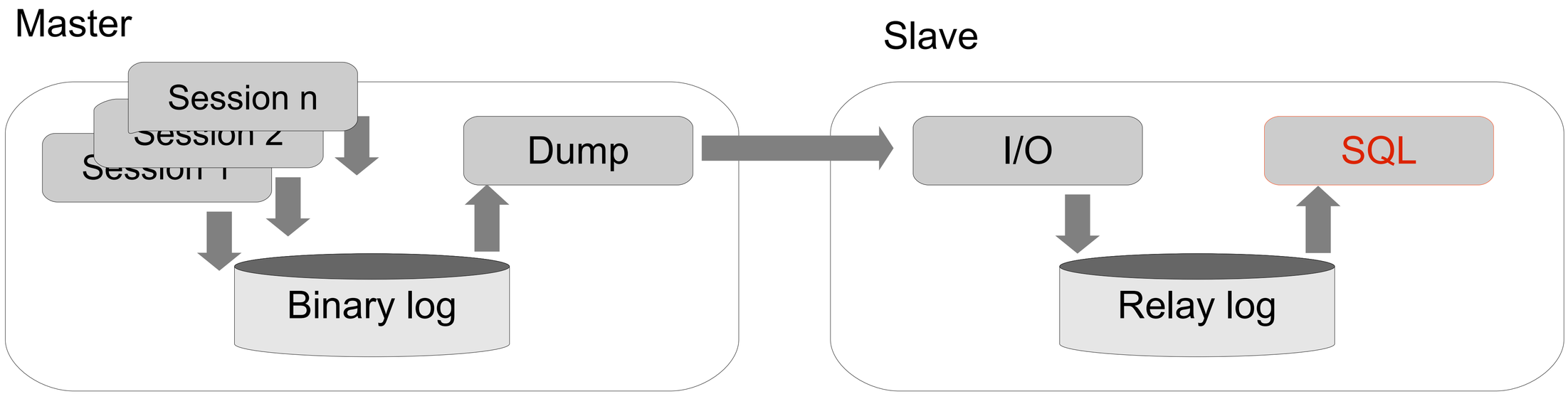
Existing Plugins

- Stable open source:
  - mysqlnd_uh – Userspace Handler
  - mysqlnd_ms – Repliation and load balancing
  - mysqlnd_qc – Client Side Query Cache
  - mysqlnd_memcache – memache mapping
- Experimental openn source:
  - mysqlnd_mc – multi connect
  - mysqlnd_pscache – prepared statement cache
  - mysqlnd_sip – SQL injection protection
- Commercial
  - MySQL Enterprise Query Analyzer



Multi-Threaded Slaves

The problem



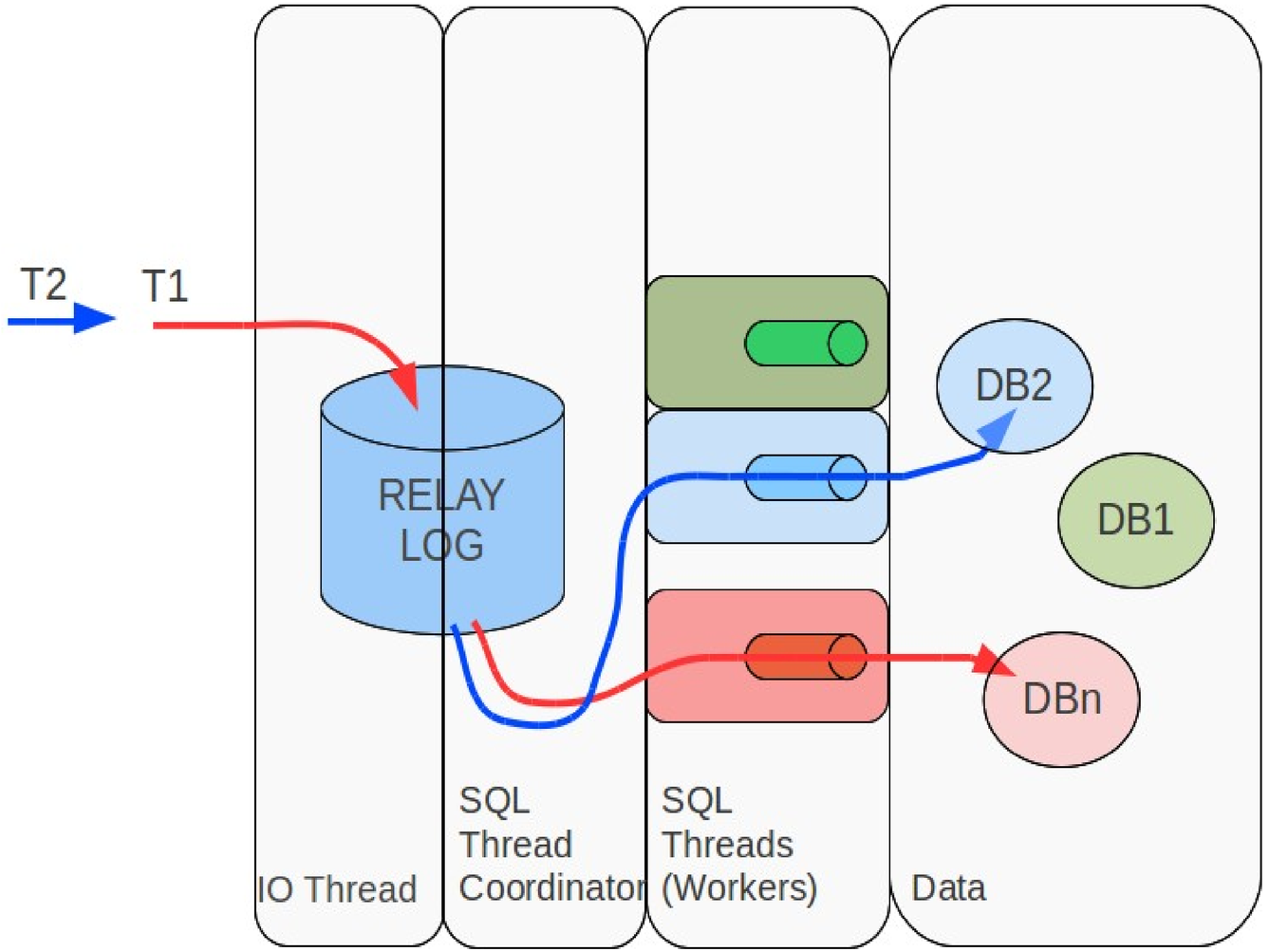
Replication system architecture: multi-threaded master vs single threaded slave applier

The problem

- Multicore on the Master and the Slave
- Recent progress of scaling-up the Innodb engine
- Slave side needs to match an upcoming solution for group commit to the binary log to overcome deficiency of the current architecture

Facilities & Prerequisites

- The user application is logically partitioned per database
- The user application is content with local database consistency



Requirements

- Arbitrary manually configurable number of Worker threads
- All binary log formats supported
- All storage engines supported
- Automatic fallback to sequential execution for statements that do not comply with parallel execution
- Automatic recovery in face of crashes in the case of the Innodb engine

Overview of architecture

- SQL thread is split into Coordinator and Worker threads communicating in style of the producer-consumer model
- Coordinator activities include:
  read replication event, handle Skip-Until-Delay options and find an appropriate Worker to execute the rest.
- Worker activities include:
  apply the event including the commit of transaction.
- Coordinator and Worker cooperate on recovery provision

MySQL 5.7
Development Milestones

MySQL 5.7

Development Milestone Release 2

500,000 Queries/Second

Download Now »

MySQL 5.7

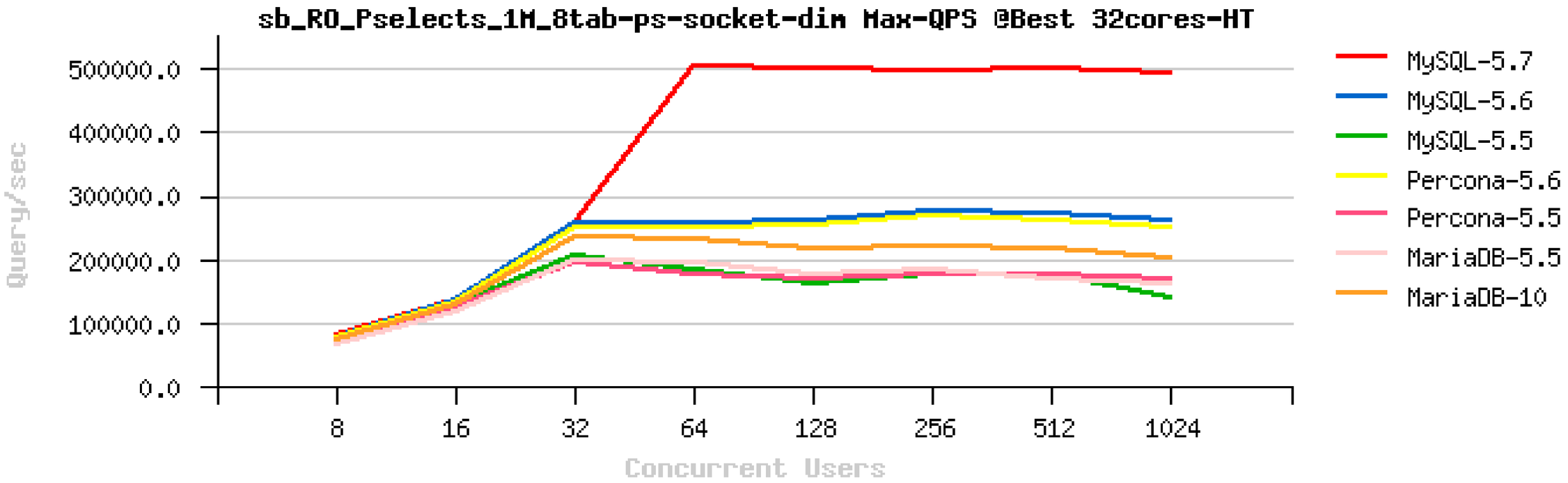
Development Milestone Release 3

1 Million Queries/Second

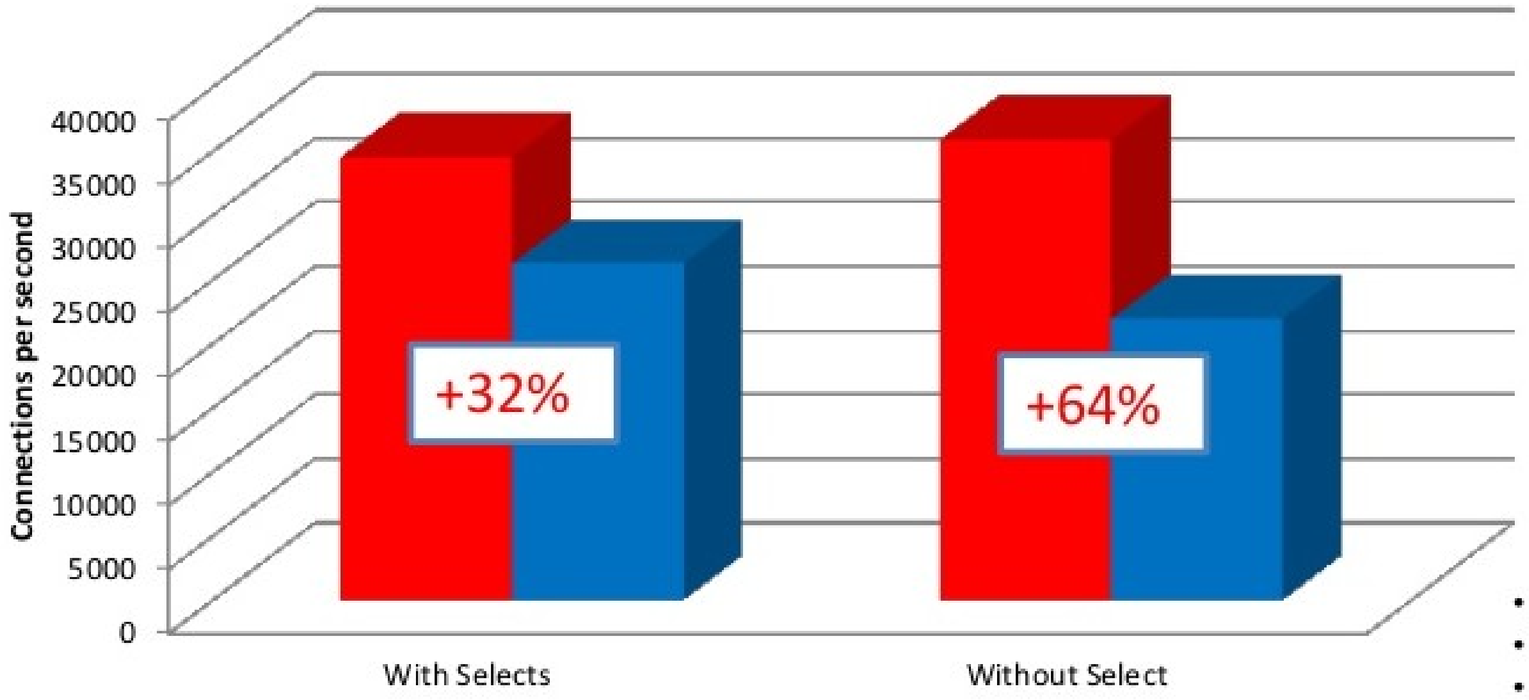
Download Now »

MySQL 5.7 – OLTP_RO Point-Selects 8-tables

- UNIX socket, sysbench 0.4.8

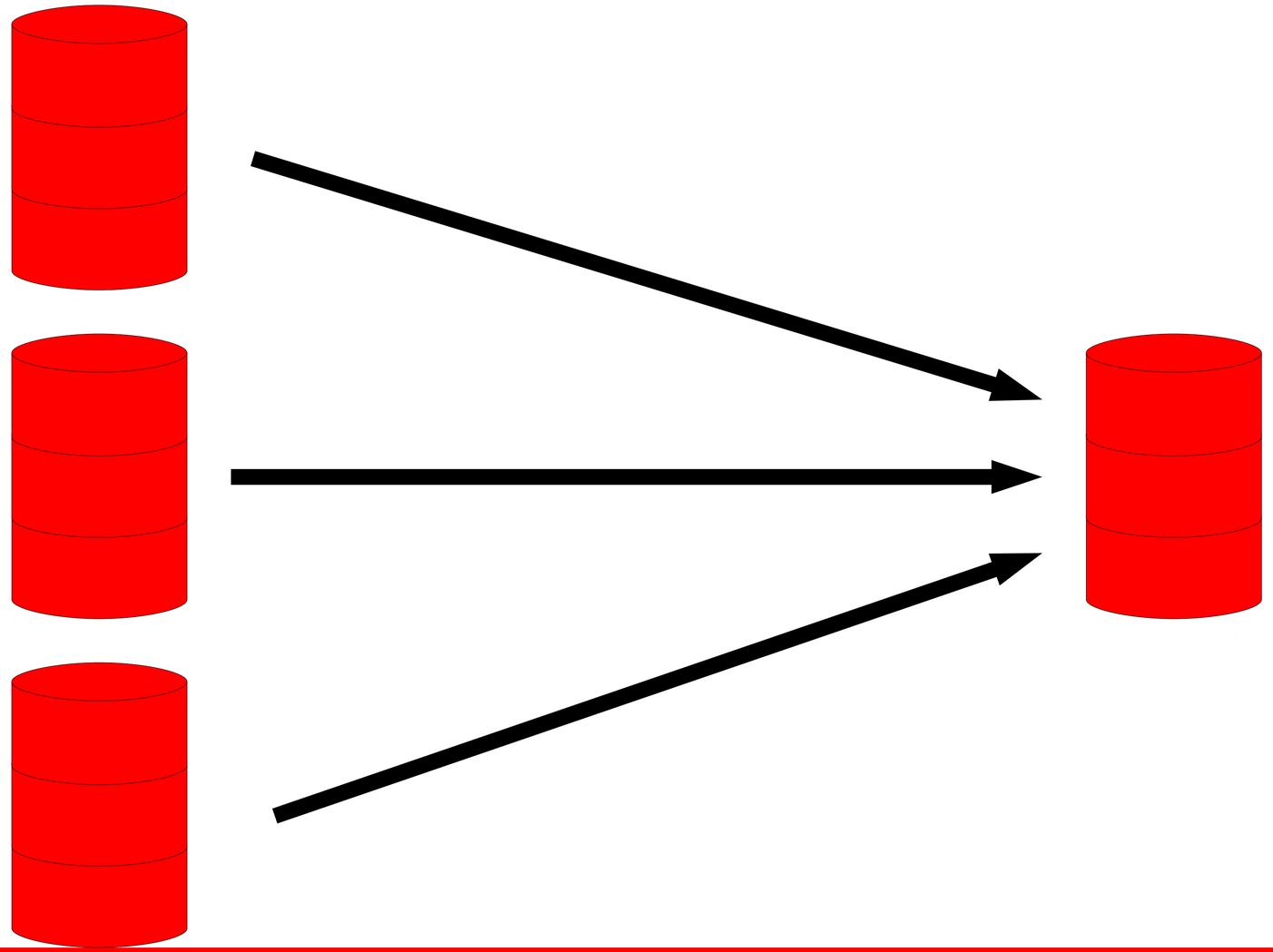


MySQL 5.7 – Connections per Second



- mysql-bench
- 25 concurrent client threads
- Executing connect/select/disconnect
- 100000 iterations each

MySQL 5.7 – Multi-Source-Replication



MSQL 5.7 – EXPLAIN

- Problem: A statement in a session is taking too long
- New option:
  - EXPLAIN FOR CONNECTION fromother session

```
EXPLAIN [FORMAT=(JSON|TRADITIONAL)] FOR CONNECTION id
```

MySQL 5.7 – InnoDB

- Improved Online ALTER TABLE
  - Online RENAME INDEX
  - Online change VARCHAR
- Improved InnoDB Temp Tables
  - New separate tablespace for temp tables
  - Improved CREATE/DROP performance for temp tables
- Optimized temp table DML
  - No REDO logging, no change bffering, no locking

Summary

With MySQL 5.6 and 5.7 we want to be

… more scalable

… faster

… better obsersvable

… more stable

… help operations

The world's most popular open source database

Resources

- Blog from MySQL Engineering VP Tomas Ulin
  http://insidemysql.com/
- Group Blog of the MySQL Server Team
  http://mysqlserverteam.com
- Ulfs Blog (mysqlnd and other things)
  http://blog.ulf-wendel.de
- Benchmark Details by DimitriK
  http://dimitrik.free.fr

Johannes Schlüter

johannes.schlueter@oracle.com
Twitter: @phperror
Blog: http://schlueters.de/blog

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.