ORACLE

MySQL Performance Schema: Die Wunderwaffe

Mario Beck
Principal Sales Consultant – MySQL
http://mablomy.blogspot.com

Program Agenda

- Intro to Performance Schema
- Performance Schema in MySQL 5.5
  - Examples
- Performance Schema in MySQL 5.6
  - Examples
- Tools to simplify for Performance Schema

Safe Harbor Statement

The following 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 decision. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

DRIVING MySQL INNOVATION

MySQL Enterprise Monitor 2.2
MySQL Cluster 7.1
MySQL Cluster Manager 1.0
MySQL Workbench 5.2
MySQL Database 5.5
MySQL Enterprise Backup 3.5
MySQL Enterprise Monitor 2.3
MySQL Cluster Manager 1.1

All GA!

2010

MySQL Enterprise Backup 3.7
Oracle VM Template for MySQL Enterprise Edition
MySQL Enterprise Oracle Certifications
MySQL Windows Installer
MySQL Enterprise Security
MySQL Enterprise Scalability

All GA!

MySQL Database 5.6 DMR*
MySQL Cluster 7.2 DMR

MySQL Labs!
("early and often")
2011

MySQL Cluster 7.2
MySQL Cluster Manager 1.4
MySQL Utilities 1.0.6
MySQL Migration Wizard
MySQL Enterprise Backup 3.8
MySQL Enterprise Audit

All GA!

MySQL Cluster 7.3 DMR
MySQL Database 5.6 GA!

Available Now!

A BETTER MySQL

Q1-Q3 2012

***Development Milestone Release**

Introduction to P_S

A first Example: Watching a session

```
mysql> SELECT name,type,processlist_user AS user,instrumented FROM threads;
+----------------------------------------+------------+-------+--------------+
| name                                   | type       | user  | instrumented |
+----------------------------------------+------------+-------+--------------+
| thread/sql/main                        | BACKGROUND | NULL  | YES          |
| thread/innodb/io_handler_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/io_handler_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/io_handler_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/io_handler_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/io_handler_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/io_handler_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/io_handler_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/io_handler_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/io_handler_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/io_handler_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/srv_error_monitor_thread | BACKGROUND | NULL  | YES          |
| thread/innodb/srv_monitor_thread       | BACKGROUND | NULL  | YES          |
| thread/innodb/srv_master_thread        | BACKGROUND | NULL  | YES          |
| thread/innodb/srv_purge_thread         | BACKGROUND | NULL  | YES          |
| thread/innodb/srv_lock_timeout_thread  | BACKGROUND | NULL  | YES          |
| thread/innodb/page_cleaner_thread      | BACKGROUND | NULL  | YES          |
| thread/sql/signal_handler              | BACKGROUND | NULL  | YES          |
| thread/sql/one_connection              | FOREGROUND | root  | YES          |
| thread/sql/one_connection              | FOREGROUND | mario | YES          |
+----------------------------------------+------------+-------+--------------+
```

Performance Schema Overview

- A new schema called "performance_schema"
- Implemented in storage engine PERFORMANCE_SCHEMA
- Records various run time statistics via instrumentation points
- At it’s core, P_S tracks latency for various events
- Latency in picosecond precision (a trillionth of a second)
- Also tracks other data as appropriate
  - bytes
  - source position
  - …

Design Goals and Implications

- Access to P_S is lock-free
  - SHOW PROCESSLIST locks the server!
- P_S uses only static memory and arrays
  - avoid malloc() and pointers -> no need for locks
  - All RAM is allocated at server boot
- P_S is configured dynamically (except for memory ;-)

A word about memory

- Configure variables “performance_schema%” in my.cnf
- Watch status variables “performance_schema%” for lost data
- TRUNCATE P_S tables to reset data collection
- See cost with SHOW ENGINE PERFORMANCE_SCHEMA STATUS;

```
+---------------------+-----------------------------+-----------+
| Type                | Name                        | Status    |
+---------------------+-----------------------------+-----------+
...
| performance_schema  | threads.row_size            | 1856      |
| performance_schema  | threads.row_count           | 224       |
| performance_schema  | threads.memory              | 415744    |
...
| performance_schema  | performance_schema.memory   | 51869696  |
+---------------------+-----------------------------+-----------+
```

Instrumentation

Instrumentation is any measurement point inside the MySQL Server code. Examples are:

- File I/O
- Wait for Mutex
- SQL Statement Execution (5.6)
- MySQL 5.6 offers 500+ instruments

Setting up Instruments

- Setup instruments dynamically
- UPDATE setup_instruments ... WHERE ...

```
mysql> SELECT * FROM setup_instruments LIMIT 15;
+-----------------------------------------------------------+---------+-------+
| NAME                                                      | ENABLED | TIMED |
+-----------------------------------------------------------+---------+-------+
| wait/synch/mutex/sql/PAGE::lock                           | NO      | NO    |
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_sync               | NO      | NO    |
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_active             | NO      | NO    |
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_pool               | NO      | NO    |
| wait/synch/mutex/sql/LOCK_des_key_file                    | NO      | NO    |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_commit           | YES     | YES   |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_commit_queue     | YES     | YES   |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_done             | YES     | YES   |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_flush_queue      | YES     | YES   |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_index            | YES     | YES   |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_log              | YES     | YES   |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_sync             | YES     | YES   |
| wait/synch/mutex/sql/MYSQL_BIN_LOG::LOCK_sync_queue       | YES     | YES   |
| wait/synch/mutex/sql/MYSQL_RELAY_LOG::LOCK_commit         | NO      | NO    |
| wait/synch/mutex/sql/MYSQL_RELAY_LOG::LOCK_commit_queue   | NO      | NO    |
+-----------------------------------------------------------+---------+-------+
```

Consumer

Any table or postprocessing of raw instrument data is called a „consumer“. Consumers can be individually activated. Examples:

- Summaries of event wait times
- Statement Digests

Setting up Consumers

- Setup consumers dynamically
- UPDATE setup_consumers ... WHERE ...

```
mysql> SELECT * FROM setup_consumers;
+--------------------------------+---------+
| NAME                           | ENABLED |
+--------------------------------+---------+
| events_stages_current          | YES     |
| events_stages_history          | YES     |
| events_stages_history_long     | YES     |
| events_statements_current      | YES     |
| events_statements_history      | YES     |
| events_statements_history_long | YES     |
| events_waits_current           | YES     |
| events_waits_history           | YES     |
| events_waits_history_long      | YES     |
| global_instrumentation         | YES     |
| thread_instrumentation         | YES     |
| statements_digest              | YES     |
+--------------------------------+---------+
```

Setting Timers in performance_schema

```
mysql>SELECT * FROM setup_timers;
+-----------+-------------+
| NAME      | TIMER_NAME  |
+-----------+-------------+
| idle      | MICROSECOND |
| wait      | CYCLE       |
| stage     | NANOSECOND  |
| statement | NANOSECOND  |
+-----------+-------------+
4 rows in set (0,00 sec)

mysql> SELECT * FROM performance_timers;
+-------------+-----------------+------------------+----------------+
| TIMER_NAME  | TIMER_FREQUENCY | TIMER_RESOLUTION | TIMER_OVERHEAD |
+-------------+-----------------+------------------+----------------+
| CYCLE       |      2519154228 |                1 |             38 |
| NANOSECOND  |      1000000000 |                1 |            104 |
| MICROSECOND |         1000000 |                1 |            133 |
| MILLISECOND |            1037 |                1 |            152 |
| TICK        |             104 |                1 |           3591 |
+-------------+-----------------+------------------+----------------+
```

- Different timers incur different costs
- Timers can be selected

P_S in MySQL 5.5

Events

- All events are WAIT events for
  - File I/O: wait/io/file/%
  - Mutexes: wait/synch/mutex/%
  - RW-Locks: wait/synch/rwlock/%
  - Conditions: wait/synch/cond/%

- Data exposed in ringbuffers:
  - events_waits_current (all currently existing wait events)
  - events_waits_history (last 10 events per connection)
  - events_waits_history_long (last 1000 events in server)

Wait Events

```
mysql> SELECT * FROM events_waits_history_long WHERE event_id=300\G
*************************** 1. row ***************************
            THREAD_ID: 21
             EVENT_ID: 300
         END_EVENT_ID: 300
           EVENT_NAME: wait/io/file/myisam/dfile
               SOURCE: mi_delete_table.c:58
          TIMER_START: 4681805148058769
            TIMER_END: 4681805394910987
           TIMER_WAIT: 246852218
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: /private/var/tmp/#sql26f_2_4e.MYD
           INDEX_NAME: NULL
          OBJECT_TYPE: FILE
OBJECT_INSTANCE_BEGIN: 4314275520
     NESTING_EVENT_ID: NULL
   NESTING_EVENT_TYPE: NULL
            OPERATION: delete
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
```

Events Summaries

- Summary tables do not discard events but consolidate event data
- Many (fixed size) aggregations available
  - events_waits_summary_by_account_by_event_name
  - events_waits_summary_by_host_by_event_name
  - events_waits_summary_by_instance
  - events_waits_summary_by_thread_by_event_name
  - events_waits_summary_by_user_by_event_name
  - events_waits_summary_global_by_event_name
- TRUNCATE <...summary...> will reset the data collection

Example: Top File I/O

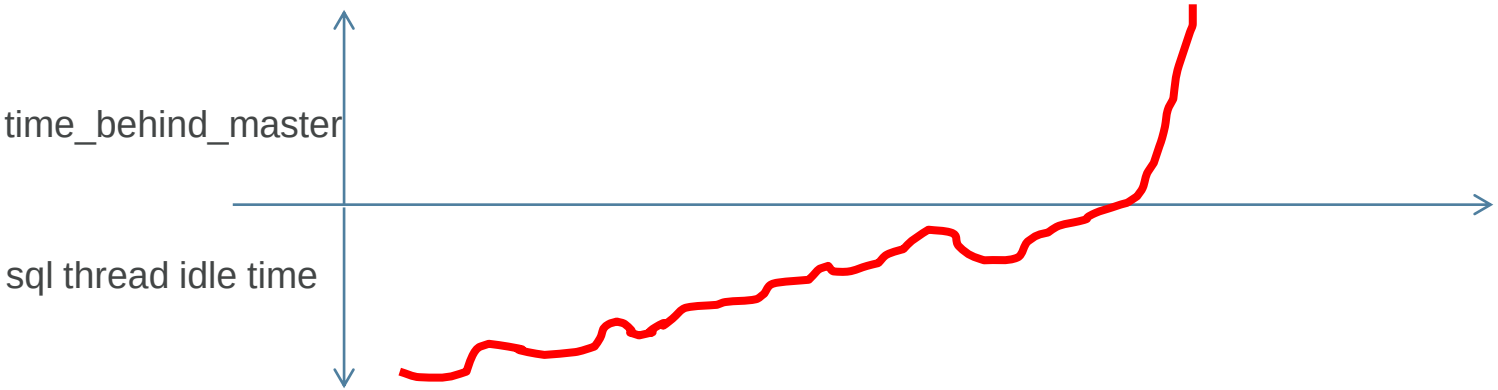
```
SELECT file_name AS file,
       count_read,
       sum_number_of_bytes_read AS total_read,
       IFNULL(sum_number_of_bytes_read / count_read, 0) AS avg_read,
       count_write,
       sum_number_of_bytes_write AS total_written,
       IFNULL(sum_number_of_bytes_write / count_write, 0.00) AS avg_write,
       (sum_number_of_bytes_read + sum_number_of_bytes_write) AS total,
       IFNULL(ROUND(100-((sum_number_of_bytes_read/(
              sum_number_of_bytes_read+sum_number_of_bytes_write)
              )*100), 2), 0.00) AS write_pct
  FROM performance_schema.file_summary_by_instance
 ORDER BY sum_number_of_bytes_read + sum_number_of_bytes_write DESC;
```

Example: Top File I/O

```
*************************** 1. row ***************************
         file: /usr/local/mysql/data/ibdata1
   count_read: 153
   total_read: 4571136
     avg_read: 29876.7059
  count_write: 3
total_written: 49152
    avg_write: 16384.0000
        total: 4620288
    write_pct: 1.06
*************************** 2. row ***************************
         file: /usr/local/mysql/data/mysql/innodb_index_stats.ibd
   count_read: 5
   total_read: 81920
     avg_read: 16384.0000
  count_write: 0
total_written: 0
    avg_write: 0.0000
        total: 81920
    write_pct: 0.00
```

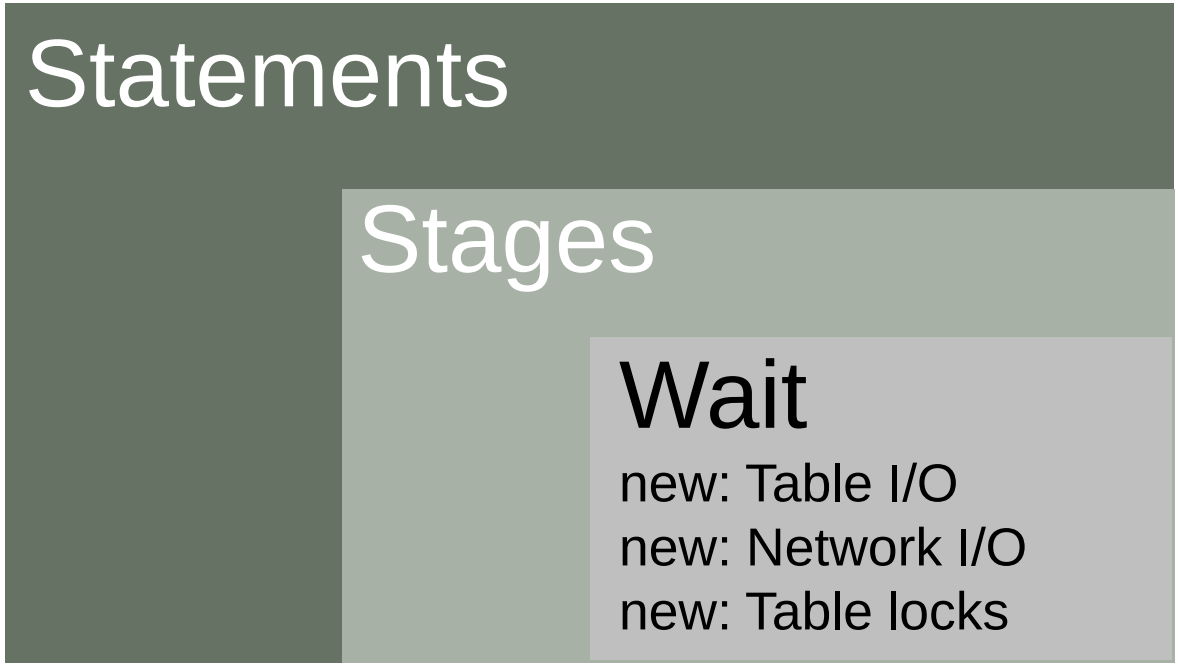
Example: Slave Load

- slave_behind_master is not sufficient
  - Will show only if already in trouble
  - reactive monitoring only
- We need PROactive monitoring – load of SQL thread
- How long is SQL thread idle?
- Watch **wait/synch/cond/sql/MYSQL_RELAY_LOG::update_cond**



P_S in MySQL 5.6

Nested Events



Statement Digest

A “digest” is a hash of a normalized query.

Digests are helpful to find similar queries and consolidate statistics for the same kind of query.

Example: ‘SELECT prod,rev FROM revenue WHERE prod = ?

Finer grained filters

- Filter for users, hosts, roles

```
mysql> SELECT * FROM setup_actors;
+------+------+------+
| HOST | USER | ROLE |
+------+------+------+
| %    | %    | %    |
+------+------+------+
```

- Filter for individual tables

```
mysql> SELECT * FROM setup_objects;
+-------------+--------+---------------+---------+-------+
| OBJECT_TYPE | SCHEMA | OBJECT_NAME   | ENABLED | TIMED |
+-------------+--------+---------------+---------+-------+
| TABLE       | test   | %             | NO      | NO    |
| TABLE       | %      | %             | YES     | YES   |
+-------------+--------+---------------+---------+-------+
```

- Combine filters *„What statements is user x executing on table t?“*

Example: Query Analyzer

```
SELECT DIGEST_TEXT AS query,
       IF(SUM_NO_GOOD_INDEX_USED > 0 OR SUM_NO_INDEX_USED > 0, '*', ''),
       COUNT_STAR AS exec_count,
       SUM_ERRORS AS err_count,
       SUM_WARNINGS AS warn_count,
       SUM_TIMER_WAIT AS total_latency,
       MAX_TIMER_WAIT AS max_latency,
       AVG_TIMER_WAIT AS avg_latency,
       SUM_ROWS_SENT AS rows_sent,
       ROUND(SUM_ROWS_SENT / COUNT_STAR) AS rows_sent_avg,
       SUM_ROWS_EXAMINED AS rows_scanned,
       DIGEST AS digest
  FROM events_statements_summary_by_digest
ORDER BY SUM_TIMER_WAIT DESC;
```

Example: Query Analyzer

```
*******************   ******** 1. row ************************
         query: SELECT * FROM tables WHERE TABLE_NAME LIKE ?
     full_scan: *
    exec_count: 6
     err_count: 0
    warn_count: 0
total_latency: 11409222000
  max_latency: 2015101000
  avg_latency: 1901537000
     rows_sent: 180
rows_sent_avg: 30
 rows_scanned: 180
       digest: b4df4004ec83f72f1721383fbd8f4e4d
```

Example: Queries of user “root”

```
TRUNCATE TABLE setup_actors;
INSERT INTO setup_actors VALUES ("%","root","%");
SELECT * FROM setup_actors;
       +------+------+------+
       | HOST | USER | ROLE |
       +------+------+------+
       | %    | root | %    |
       +------+------+------+
TRUNCATE TABLE events_statements_summary_by_digest;
...

SELECT DIGEST_TEXT AS query, COUNT_STAR AS exec_count,
       FROM events_statements_summary_by_digest;
```

Example: Who created tmp tables?

```
SELECT DIGEST_TEXT AS query,
       COUNT_STAR AS exec_count,
       SUM_CREATED_TMP_TABLES AS memory_tmp_tables,
       SUM_CREATED_TMP_DISK_TABLES AS disk_tmp_tables,
       ROUND(SUM_CREATED_TMP_TABLES / COUNT_STAR,
       ROUND((SUM_CREATED_TMP_DISK_TABLES /
                    SUM_CREATED_TMP_TABLES) * 100),
       DIGEST AS digest
  FROM events_statements_summary_by_digest
 WHERE SUM_CREATED_TMP_TABLES > 0
ORDER BY SUM_CREATED_TMP_DISK_TABLES DESC DESC;
```

Example: Who created tmp tables?

```
*************************** 1. row ***************************
                  query: SELECT * FROM columns ORDER BY …
             exec_count: 62
       memory_tmp_tables: 62
         disk_tmp_tables: 12
avg_tmp_tables_per_query: 1
  tmp_tables_to_disk_pct: 20
                 digest: c84fcc3cb43d92ea62efc5012ec0ff82
```

Example: Find unused Indexes

```
SELECT object_schema,
       object_name,
       index_name
  FROM table_io_waits_summary_by_index_usage
 WHERE index_name IS NOT NULL
   AND count_star = 0
 ORDER BY object_schema, object_name;
```

Example: Find unused Indexes

```
+---------------+-------------+-----------------+
| object_schema | object_name | index_name      |
+---------------+-------------+-----------------+
| mem           | dc_p_double | PRIMARY         |
| mem           | dc_p_double | end_time        |
| mem           | dc_p_long   | PRIMARY         |
| mem           | dc_p_long   | end_time        |
| mem           | dc_p_string | begin_time      |
| mem           | dc_p_string | end_time        |
```

Making it easier

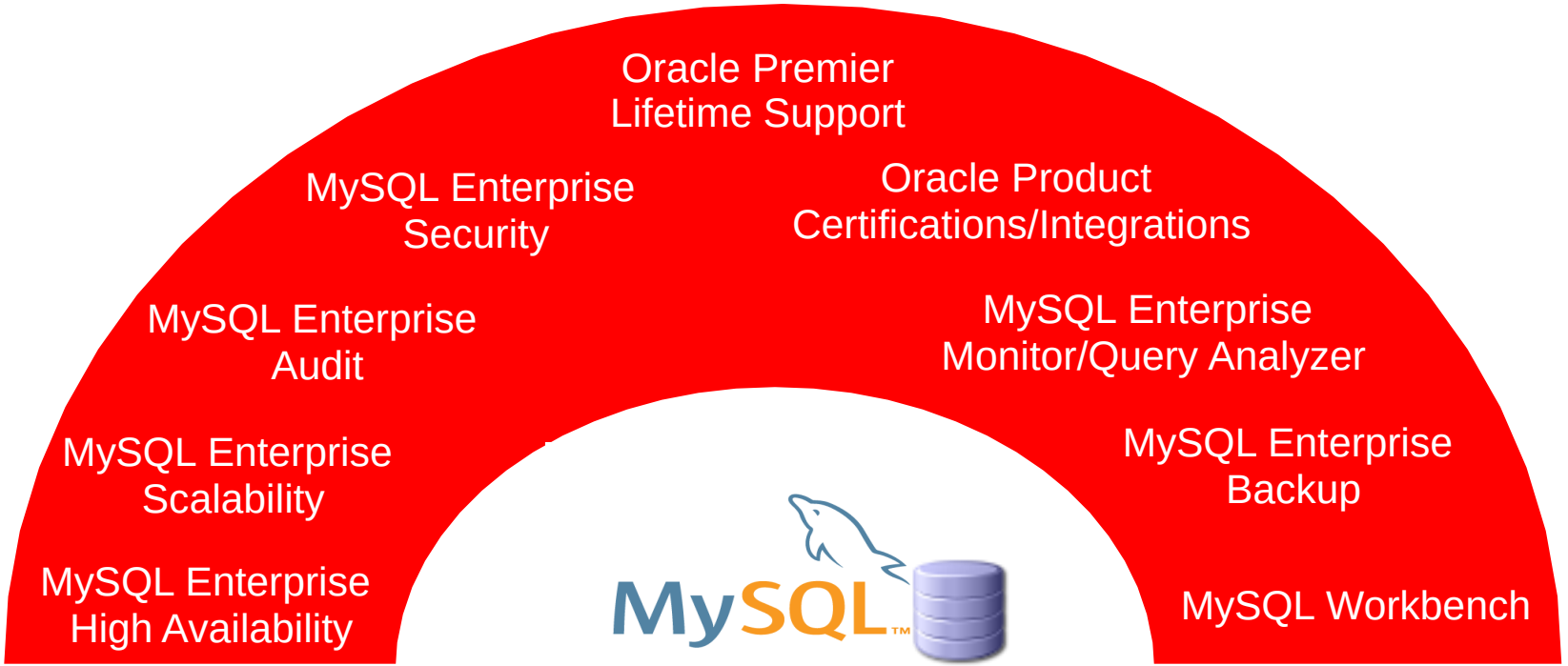
ps_helper

http://www.markleith.co.uk/ps_helper/

- A set of functions for formatting data
  - e.g. format_bytes(23487234) = “22.40 MiB“
- A set of views to prepare most common queries
  - All shown examples are views in ps_helper
- A set of procedures for common tasks
- Versions available for both 5.5 and 5.6

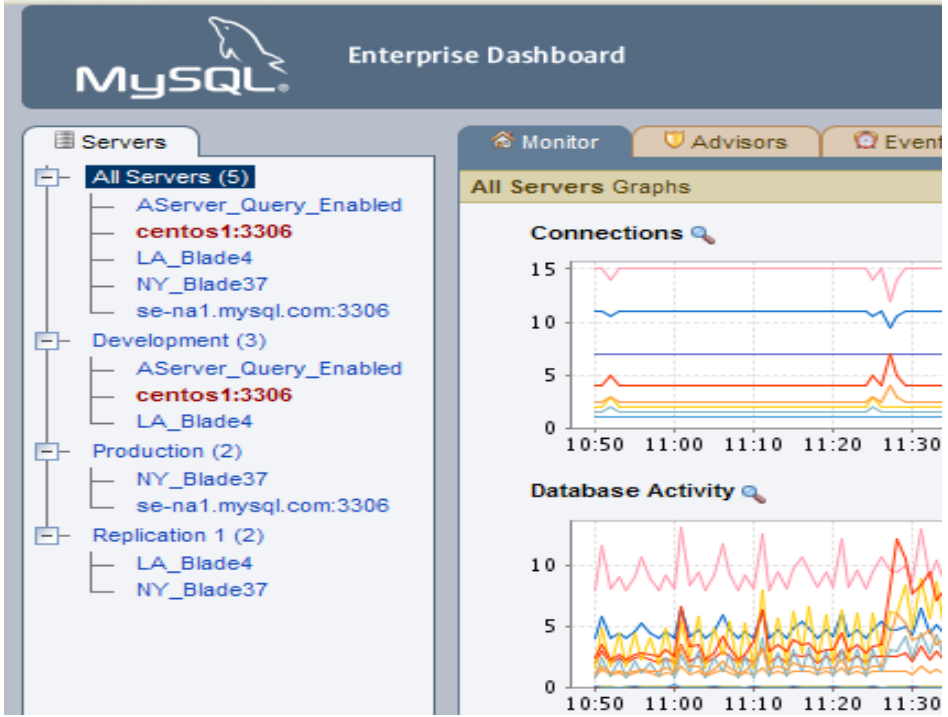
MySQL Enterprise Edition

Highest Levels of Security, Performance and Availability



MySQL Enterprise Monitor

- Web-based, global view of MySQL/Cluster applications (on-premise and Cloud deployments)
- Automated, rules-based monitoring and alerts (SMTP, SNMP enabled)
- Query capture, monitoring, analysis and tuning, correlated with Monitor graphs
- Real-time Replication Monitor with auto-discovery of master-slave topologies
- Integrated with Oracle Support

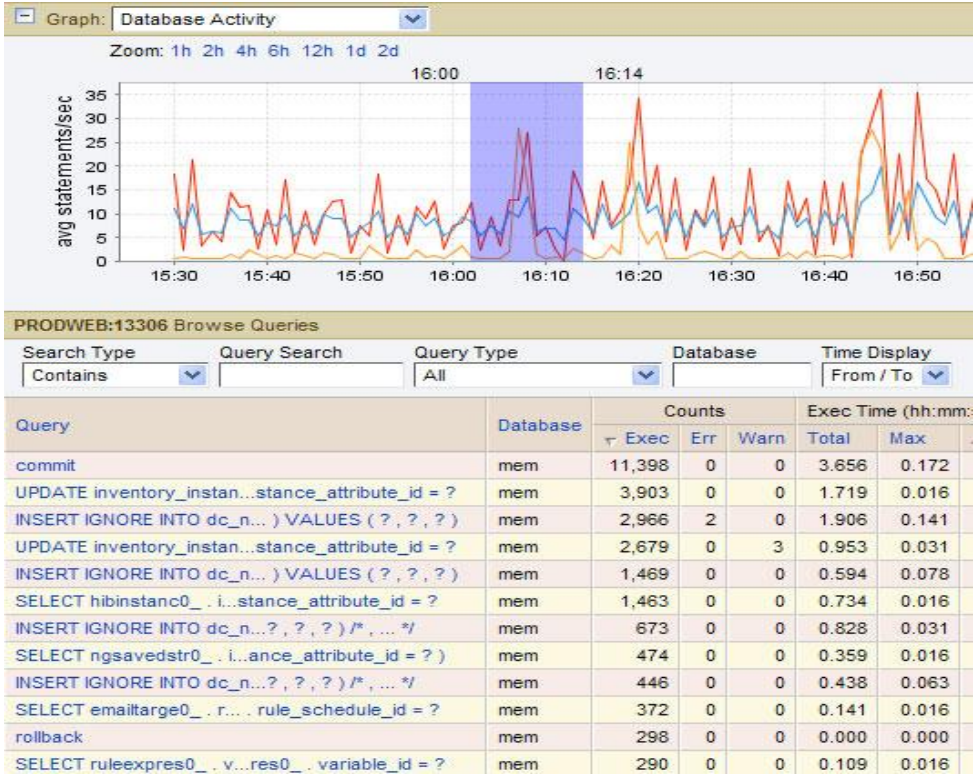


A Virtual MySQL DBA Assistant!

MySQL Query Analyzer

- Centralized monitoring of queries without Slow Query Log, SHOW PROCESSLIST;
- Aggregated view of query execution counts, time, and rows
- Visual “grab and go” correlation with Monitor graphs
- Enabled via Connectors (PHP, JDBC, .Net) or MySQL Proxy

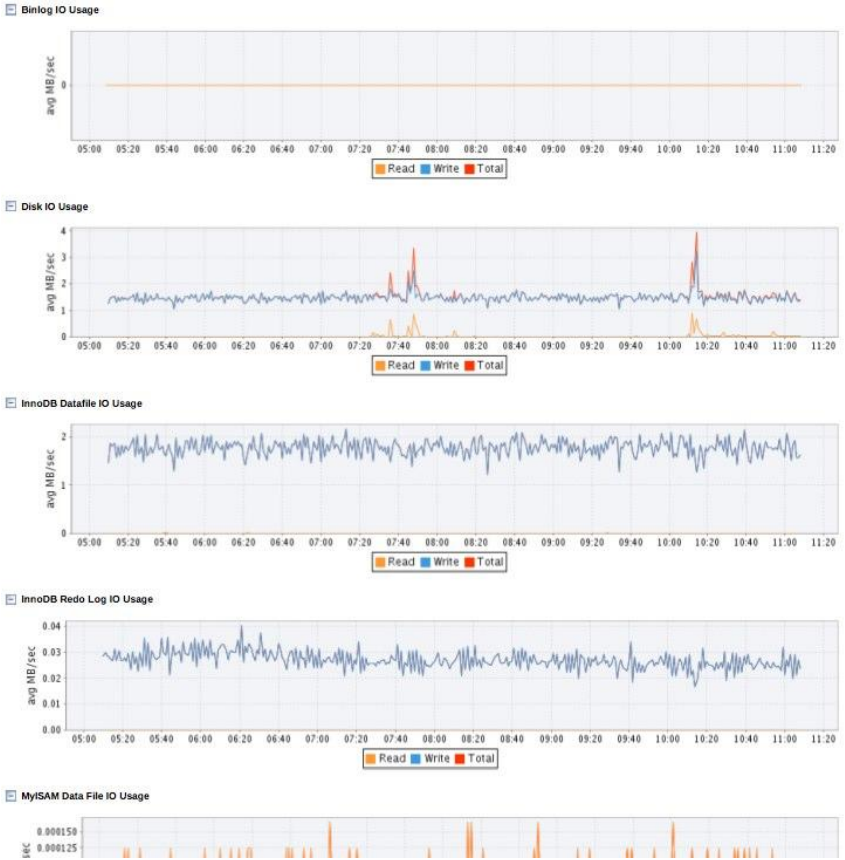
Saves you time parsing atomic executions from logs. Finds problems you cannot find yourself.



Query	Database	Counts			Exec Time (hh:mm:	
		Exec	Err	Warn	Total	Max
commit	mem	11,398	0	0	3.656	0.172
UPDATE inventory_instan...stance_attribute_id = ?	mem	3,903	0	0	1.719	0.016
INSERT IGNORE INTO dc_n...) VALUES (? , ? , ?)	mem	2,966	2	0	1.906	0.141
UPDATE inventory_instan...stance_attribute_id = ?	mem	2,679	0	3	0.953	0.031
INSERT IGNORE INTO dc_n...) VALUES (? , ? , ?)	mem	1,469	0	0	0.594	0.078
SELECT hibinstanc0_ . i...stance_attribute_id = ?	mem	1,463	0	0	0.734	0.016
INSERT IGNORE INTO dc_n...? , ? , ?) /* , ... */	mem	673	0	0	0.828	0.031
SELECT ngsavedstr0_ . i...ance_attribute_id = ?)	mem	474	0	0	0.359	0.016
INSERT IGNORE INTO dc_n...? , ? , ?) /* , ... */	mem	446	0	0	0.438	0.063
SELECT emailtarge0_ . r... . rule_schedule_id = ?	mem	372	0	0	0.141	0.016
rollback	mem	298	0	0	0.000	0.000
SELECT ruleexpres0_ . v...res0_ . variable_id = ?	mem	290	0	0	0.109	0.016

MySQL Enterprise Monitor and P_S

- Graphs show historical status
- If monitoring 5.5 with file instruments enabled
- Binlog I/O Usage
- InnoDB Datafile I/O Usage
- InnoDB Redo Log I/O Usage
- MyISAM Data File I/O Usage
- MyISAM Index File I/O Usage



Summary

How to continue with performance_schema

- P_S = Swiss Army Knife, ext.version
- Highly configurable, low overhead
- Play with it
- Follow blogs:
  - http://marcalff.blogspot.co.uk/
  - http://www.markleith.co.uk/
- Use ps_helper:
  - http://www.markleith.co.uk/ps_helper/

ORACLE

MySQL Performance Schema: Die Wunderwaffe

Fragen?

Learn More

- mysql.com
  - MySQL Products and Editions
  - TCO calculator
  - Customer use cases and success stories
- dev.mysql.com
  - Downloads, Documentation
  - Forums
  - PlanetMySQL
- eDelivery.oracle.com
  - Download and evaluate all MySQL products