# Using MySQL 5.6 Performance Schema to Troubleshoot Typical Workload Bottlenecks

Peter Zaitsev,
CEO, Percona
Percona Technical Webinars
May 15, 2013

## About Presentation

- Introduction to Performance Schema
    - Focus on MySQL 5.6
- Performance Schema configuration
- Examples of using MySQL Performance Schema

## Acknowledgements

- Two people helped me greatly with this presentation:
  - **Marc Alff,** Performance Schema Architect, Oracle
  - **Mark Leith**, Development Manager, MySQL Enterprise Tools, Oracle
    - Author of **ps_helper**

# Performance Analyses

- Before MySQL 5.5
- STATUS variables (SESSION and GLOBAL)
  - Mostly non timed data
- SHOW INNODB STATUS
- SHOW PROFILES
- MySQL Slow Query Log
  - Does not include timing details
  - Some timing details in Percona Server

## PERFORMANCE_SCHEMA

- Provide Details about Query execution in structured way
- Include Timing
- Make Accessible through SQL
- Inspired by Oracle Wait Interface
    - Design started way before Oracle acquired MySQL

# Performance Schema Basics

- **PERFORMANCE_SCHEMA** Storage Engine
  - Only used for special tables in *performance_schema* database
- Platform Independent
- Monitor Server “events”
  - Statements, Stages, Waits
- Probes are placed in “Instrumentation Points” in the Server
- Focus on Low Overhead/Fast Collection
  - Time Measured Picoseconds
  - Operates in Fixed Memory
  - Per thread Event IDs
- 545 “instruments” in MySQL 5.6.11

# History and the Future

- MySQL 5.5
  - File I/O, Mutexes, RW Locks etc
  - Mainly helpful for Server Developers
- MySQL 5.6
  - Network I/O, Table I/O, Stages, Statements, Idle time
  - Tracks position, IO sizes etc
  - Hierarchy of Events
  - A lot more useful for DBAs
- MySQL 5.7
  - Work on improving Performance Schema continues.

# Performance Schema Tables

- 52 tables in *performance_schema*
  - No views shipped with server
- Mix of configuration tables and data tables
- Configuration Tables
- Object Tables
- Current Tables
- History Tables
- Summary Tables
- Other Tables

## Configuration Tables

- **setup_instruments**
  - Which instrumentation points are enabled
- **setup_consumers**
  - Which aggregation tables are maintained
  - Watch out for hierarchy !
- **setup_actors**
  - Define which users will be instrumented
- **setup_objects**
  - Which objects need to be instrumented
- **Threads**
  - Define which threads are instrumented

# Object Tables

- **cond_instances**
  - Identifies Conditions
- **file_instances**
  - Identifies Files
- **mutex_instances**
  - Identifies Mutexes
- **rwlock_instances**
  - Identifies rw_locks
- **socket_instances**
  - Identifies sockets
- **threads**
- **users**

# Current Tables

- Show what is currently happening
  - **events_stages_current**
  - **events_statements_current**
  - **events_waits_current**

```
mysql [localhost] {msandbox} (performance_schema) > select *
from events_stages_current \G
*************************** 1. row ***************************
         THREAD_ID: 59
          EVENT_ID: 1740786
      END_EVENT_ID: NULL
        EVENT_NAME: stage/sql/Sending data
            SOURCE: sql_executor.cc:187
       TIMER_START: 288480284583320000
         TIMER_END: NULL
        TIMER_WAIT: NULL
  NESTING_EVENT_ID: 1740772
NESTING_EVENT_TYPE: STATEMENT
1 row in set (0.00 sec)
```

## History Tables

- Two tables for each event type
  - **events_waits_history**
  - **events_waits_history_long**
- Same data structure
- Table  "_long" expires data globaly
- Table without "_long" by each thread separately

# Summary Tables

- Aggregation for event types and objects:
  - **events_stages_summary_by_thread_by_event_name**
  - **events_waits_summary_by_thread_by_event_name**
  - **file_summary_by_instance**
  - **table_io_waits_summary_by_index_usage**
  - **table_io_waits_summary_by_table**
  - **table_lock_waits_summary_by_table**

```
*************************** 1. row ***************************
                FILE_NAME: /mnt/data/sandboxes/msb_5_6_11/data/ib_logfile0
               EVENT_NAME: wait/io/file/innodb/innodb_log_file
    OBJECT_INSTANCE_BEGIN: 140459772958272
               COUNT_STAR: 982665
           SUM_TIMER_WAIT: 839809863881042
           MIN_TIMER_WAIT: 1202400
           AVG_TIMER_WAIT: 854624504
           MAX_TIMER_WAIT: 47269445190
               COUNT_READ: 6
…
```

# Other Tables

- Various other tables added as needed
  - **performance_timers**
  - **host_cache**
  - **session_connect_attrs**

```
mysql [localhost] {msandbox} (performance_schema) > select * from session_connect_attrs;
+----------------+-----------------+----------------+------------------+
| PROCESSLIST_ID | ATTR_NAME       | ATTR_VALUE     | ORDINAL_POSITION |
+----------------+-----------------+----------------+------------------+
|             40 | _os             | linux-glibc2.5 |                0 |
|             40 | _client_name    | libmysql       |                1 |
|             40 | _pid            | 22210          |                2 |
|             40 | _client_version | 5.6.11         |                3 |
|             40 | _platform       | x86_64         |                4 |
|             40 | program_name    | mysql          |                5 |
+----------------+-----------------+----------------+------------------+
6 rows in set (0.00 sec)
```

# Configuring Performance Schema

- Is better in MySQL 5.6 but still is a pain
- Enabled by default in MySQL 5.6
  - **skip_performance_schema** to disable
- Limits have to be set statically
  - **performance_schema_events_stages_history_long_size=10000**
- Check "lost" values for Performance Schema in **SHOW STATUS**
- Check Performance Schema Memory usage with **SHOW ENGINE PERFORMACE_SCHEMA STATUS**

## Instruments and Consumers

- Can be configured at startup in MySQL 5.6
  - **--performance-schema-instrument='wait/synch/cond/%=counted‘**
  - **--performance-schema-consumer-events-waits-history=on**

# Instruments

- Can be enabled through SQL
  - Changes lost on restart
  - Use **–init-file=ps.sql** for configuration
- Can enable counting and timing

```
mysql [localhost] {msandbox} (performance_schema) > select * from setup_instruments where name
like "%file%" limit 10;
+--------------------------------------------------+---------+-------+
| NAME                                             | ENABLED | TIMED |
+--------------------------------------------------+---------+-------+
| wait/synch/mutex/sql/LOCK_des_key_file           | NO      | NO    |
| wait/synch/mutex/innodb/file_format_max_mutex    | NO      | NO    |
| wait/synch/mutex/innodb/srv_dict_tmpfile_mutex   | NO      | NO    |
| wait/synch/mutex/innodb/srv_misc_tmpfile_mutex   | NO      | NO    |
| wait/synch/mutex/innodb/srv_monitor_file_mutex   | NO      | NO    |
| wait/io/file/sql/map                             | YES     | YES   |
| wait/io/file/sql/binlog                          | YES     | YES   |
| wait/io/file/sql/binlog_index                    | YES     | YES   |
| wait/io/file/sql/relaylog                        | YES     | YES   |
| wait/io/file/sql/relaylog_index                  | YES     | YES   |
+--------------------------------------------------+---------+-------+
10 rows in set (0.00 sec)
```

# Consumers

- Which “tables” are populated
- Watch out for hierarchy
    - http://bit.ly/127jZvU

```
mysql [localhost] {msandbox} (performance_schema) >
select * from setup_consumers;
+---------------------------------+---------+
| NAME                            | ENABLED |
+---------------------------------+---------+
| events_stages_current           | NO      |
| events_stages_history           | NO      |
| events_stages_history_long      | NO      |
| events_statements_current       | YES     |
| events_statements_history       | NO      |
| events_statements_history_long  | NO      |
| events_waits_current            | NO      |
| events_waits_history            | NO      |
| events_waits_history_long       | NO      |
| global_instrumentation          | YES     |
| thread_instrumentation          | YES     |
| statements_digest               | YES     |
+---------------------------------+---------+
12 rows in set (0.00 sec)
```

# Configuring Threads

- Can enable/disable instrumentation for any thread
  - Both user and system

```
*************************** 21. row ***************************
          THREAD_ID: 23
               NAME: thread/sql/one_connection
               TYPE: FOREGROUND
     PROCESSLIST_ID: 4
   PROCESSLIST_USER: msandbox_rw
   PROCESSLIST_HOST: localhost
     PROCESSLIST_DB: sbtest
PROCESSLIST_COMMAND: Query
   PROCESSLIST_TIME: 0
  PROCESSLIST_STATE: statistics
   PROCESSLIST_INFO: SELECT c from sbtest where id between 503759 and 503858 order by c
   PARENT_THREAD_ID: 1
               ROLE: NULL
       INSTRUMENTED: YES
21 rows in set (0.00 sec)
```

# Configuring “actors”

- By default all users from all hosts are profiled
  - We can change that as needed

```
mysql [localhost] {msandbox} (performance_schema) >
select * from setup_actors;
+------+------+------+
| HOST | USER | ROLE |
+------+------+------+
| %    | %    | %    |
+------+------+------+
1 row in set (0.00 sec)
```

# Configuring Objects

- Object means *Table* for now
- Skips instrumentation
    - Table IO
    - Lock Information

```
mysql [localhost] {msandbox} (performance_schema) > select * from setup_objects;
+-------------+--------------------+-------------+---------+-------+
| OBJECT_TYPE | OBJECT_SCHEMA      | OBJECT_NAME | ENABLED | TIMED |
+-------------+--------------------+-------------+---------+-------+
| TABLE       | mysql              | %           | NO      | NO    |
| TABLE       | performance_schema | %           | NO      | NO    |
| TABLE       | information_schema | %           | NO      | NO    |
| TABLE       | %                  | %           | YES     | YES   |
+-------------+--------------------+-------------+---------+-------+
4 rows in set (0.00 sec)
```

## Getting Incremental Data

- What have been top statements for last 5 minutes ?
    - Pull the data in the separate table and compute the difference
    - Use **TRUNCATE TABLE** to flush statistics

# Overhead

- Can vary significantly on workload and configuration
- CPU bound, heavy on contention – worse overhead
- Mark Callaghan results
  - http://bugs.mysql.com/bug.php?id=68413
  - 3% overhead for having PS compiled in
  - 11% overhead with default settings
- Dimitri Kravchuk investigation
  - http://bit.ly/14obY7v
- My results (**sysbench** read only)
  - 10% overhead PS OFF->ON
  - 24% overhead PS OFF-> ALL ON
- Recognize the gains as well!

## Things which pain me

- Complexity
    - Percona offers wonderful Support Contracts ☺
- Overhead
    - Can we simplify overhead configuration ? Reduce it ?
- Support for Prepared Statements
- Lack of Per statement wait event aggregation
- No Resource Usage (CPU time, Memory)

## PS_Helper

- A great tool by Mark Leith to make **PERFORMANCE_SCHEMA** easier to use
  - http://bit.ly/Sw8AmE
- Implemented as set of Views and Stored Procedures
- Integrates data from **PERFORMANCE_SCHEMA** and **INFORMATION_SCHEMA** where possible

## STATEMENTS

- Note: Can’t order by “total_latency”

```
mysql [localhost] {msandbox} (ps_helper) > select * from statement_analysis
order by exec_count desc limit 1 \G
*************************** 1. row ***************************
        query: SELECT c FROM sbtest WHERE id = ?
    full_scan: 
   exec_count: 590402
    err_count: 0
   warn_count: 0
total_latency: 00:21:54.47
  max_latency: 1.15 s
  avg_latency: 2.23 ms
    rows_sent: 590470
rows_sent_avg: 1
 rows_scanned: 590519
       digest: 88dbb114cd63f49039275d1129fc8646
1 row in set (0.00 sec)
```

## TEMP TABLES

- Would be good to track tmp table sizes in memory and on disk

```
mysql [localhost] {msandbox} (ps_helper) > select * from
statements_with_temp_tables order by exec_count desc limit 1 \G
*************************** 1. row ***************************
                   query: SELECT DISTINCTROW c FROM sbte ... id BETWEEN ? AND ?
ORDER BY c
              exec_count: 211797
       memory_tmp_tables: 211802
         disk_tmp_tables: 0
avg_tmp_tables_per_query: 1
  tmp_tables_to_disk_pct: 0
                  digest: 51cd1a1d76fcec29235fa3303af8af0e
1 row in set (0.00 sec)
```

## SORTING

- Accounting average sort space used would help

```
mysql [localhost] {msandbox} (ps_helper) > select * from
ps_helper.statements_with_sorting order by exec_count desc limit 1 \G
*************************** 1. row ***************************
            query: SELECT c FROM sbtest WHERE id BETWEEN ? AND ? ORDER BY c
       exec_count: 281347
sort_merge_passes: 281357
  avg_sort_merges: 1
sorts_using_scans: 0
 sort_using_range: 281357
      rows_sorted: 28135800
  avg_rows_sorted: 100
           digest: 7cba2ddcbeaca5d0912a514d5cdc614b
1 row in set (0.00 sec)
```

# TABLE_STATISTICS

- Even more stats than famous Google’s USER_STATISTICS patch

```
mysql [localhost] {msandbox} (ps_helper) > select * from schema_table_statistics where
table_schema='sbtest' limit 1 \G
*************************** 1. row ***************************
     table_schema: sbtest
       table_name: sbtest
     rows_fetched: 158764154
    fetch_latency: 1.37h
    rows_inserted: 378901
   insert_latency: 00:07:17.38
     rows_updated: 1136714
   update_latency: 00:45:40.08
     rows_deleted: 378902
   delete_latency: 00:03:00.34
 io_read_requests: 636003
          io_read: 9.70 GiB
  io_read_latency: 00:28:12.01
io_write_requests: 203925
         io_write: 3.11 GiB
 io_write_latency: 17.26 s
 io_misc_requests: 2449
  io_misc_latency: 3.87 s
1 row in set (3.25 sec)
```

## ..with Buffer Pool Information

```
mysql [localhost] {msandbox} (ps_helper) > select * from
schema_table_statistics_with_buffer where table_schema='sbtest' limit 1 \G
*************************** 1. row ***************************
              table_schema: sbtest
                table_name: sbtest
              rows_fetched: 152462125
             fetch_latency: 1.31h
             rows_inserted: 363850
            insert_latency: 00:06:59.73
              rows_updated: 1091562
            update_latency: 00:43:51.35
              rows_deleted: 363852
            delete_latency: 00:02:53.92
….
innodb_buffer_allocated: 110.41 MiB
        innodb_buffer_data: 97.63 MiB
       innodb_buffer_pages: 7066
innodb_buffer_pages_hashed: 7066
   innodb_buffer_pages_old: 7066
 innodb_buffer_rows_cached: 593628
1 row in set (24.82 sec)
```

# Index Usage

- Can also find unused indexes with **schema_unused_indexes** view

```
mysql [localhost] {msandbox} (ps_helper) > select * from
schema_index_statistics limit 1 \G
*************************** 1. row ***************************
  table_schema: sbtest
    table_name: sbtest
    index_name: PRIMARY
 rows_selected: 222005091
select_latency: 1.91h
 rows_inserted: 0
insert_latency: 0 ps
  rows_updated: 1589497
update_latency: 1.07h
  rows_deleted: 529831
delete_latency: 0 ps
1 row in set (0.00 sec)
```

# Active File IO

```
mysql [localhost] {msandbox} (ps_helper) > select * from
top_io_by_file limit 1 \G
*************************** 1. row ***************************
         file: @@datadir/sbtest/sbtest.ibd
   count_read: 1535779
   total_read: 23.43 GiB
     avg_read: 16.00 KiB
  count_write: 461491
total_written: 7.05 GiB
  avg_written: 16.01 KiB
        total: 30.48 GiB
    write_pct: 23.12
1 row in set (0.00 sec)
```

# Better PROCESSLIST

```
mysql [localhost] {msandbox} (ps_helper) > select * from processlist_full limit 1,1 \G
*************************** 1. row ***************************
                thd_id: 29
               conn_id: 10
                  user: msandbox_rw@localhost
                    db: sbtest
               command: Query
                 state: updating
                  time: 0
     current_statement: UPDATE sbtest set k=k+1 where id=593459
        last_statement: NULL
last_statement_latency: NULL
          lock_latency: 92.00 us
         rows_examined: 0
             rows_sent: 0
         rows_affected: 0
            tmp_tables: 0
       tmp_disk_tables: 0
             full_scan: NO
             last_wait: wait/io/file/innodb/innodb_data_file
     last_wait_latency: Still Waiting
                source: fil0fil.cc:5367
1 row in set (0.08 sec)
```

# What have been user up to ?

```
mysql [localhost] {msandbox} (ps_helper) > select * from user_summary_by_statement_type where user='msandbox_rw';
+-------------+-------------------+----------+---------------+-------------+--------------+-----------+---------------+---------------+------------+
| user        | statement         | count    | total_latency | max_latency | lock_latency | rows_sent | rows_examined | rows_affected | full_scans |
+-------------+-------------------+----------+---------------+-------------+--------------+-----------+---------------+---------------+------------+
| msandbox_rw | select            | 15762627 | 18.31h        | 1.95 s      | 00:37:45.28  | 295144775 |     690492895 |             0 |          0 |
| msandbox_rw | update            |  3377656 | 5.38h         | 2.01 s      | 00:11:59.34  |         0 |       3377656 |       3377656 |          0 |
| msandbox_rw | commit            |  1125879 | 2.27h         | 711.67 ms   | 0 ps         |         0 |             0 |             0 |          0 |
| msandbox_rw | insert            |  1125882 | 00:52:13.03   | 1.05 s      | 00:02:53.33  |         0 |             0 |       1125882 |          0 |
| msandbox_rw | delete            |  1125882 | 00:42:12.21   | 987.01 ms   | 00:02:43.17  |         0 |       1125882 |       1125882 |          0 |
| msandbox_rw | begin             |  1125911 | 00:08:02.31   | 87.85 ms    | 0 ps         |         0 |             0 |             0 |          0 |
| msandbox_rw | show_table_status |        1 | 47.66 ms      | 47.66 ms    | 342.00 us    |         1 |             1 |             0 |          1 |
| msandbox_rw | Quit              |        1 | 11.45 us      | 11.45 us    | 0 ps         |         0 |             0 |             0 |          0 |
+-------------+-------------------+----------+---------------+-------------+--------------+-----------+---------------+---------------+------------+
8 rows in set (0.01 sec)
```

## Lets Get Hands Dirty

- Bottlenecks with Disk IO
- Excessive Mutex Contention
- Row locks and Meta Data Locks

## Types of Performance Problems

- “Whole server” overload problems
  - PERFORMANCE_SCHEMA is very good for it
- Query Performance Problem
  - Good as well
- Specific Query **instance** Performance Problem
  - Has ways to go still

# DISK IO

- Remember to sort by SUM_TIMER_WAIT
- Get information about given **thread** io bottleneck
- Can get aggregated data from **file_summary_by_instance**
    - With file names but no thread_id information

```
mysql [localhost] {msandbox} (performance_schema) > select * from events_waits_summary_by_thread_by_event_name where thread_id=50 order by
sum_timer_wait desc limit 5;
+-----------+-------------------------------------+------------+------------------+----------------+----------------+----------------+
| THREAD_ID | EVENT_NAME                          | COUNT_STAR | SUM_TIMER_WAIT   | MIN_TIMER_WAIT | AVG_TIMER_WAIT | MAX_TIMER_WAIT |
+-----------+-------------------------------------+------------+------------------+----------------+----------------+----------------+
|        50 | wait/io/table/sql/handler           |   20723427 | 1274640121436475 |         105525 |       61507005 |  1672119329305 |
|        50 | wait/io/file/innodb/innodb_data_file |      93185 |  239475455286555 |        2751020 |     2569892535 |   377507798250 |
|        50 | idle                                |    1026400 |  171223633000000 |        2000000 |      166000000 |    88556000000 |
|        50 | wait/io/file/innodb/innodb_log_file  |      67840 |   65499501825285 |        7024950 |      965499480 |   369105432770 |
|        50 | wait/lock/table/sql/handler         |    1857294 |   11156965686975 |         283745 |        6006885 |    77666892450 |
+-----------+-------------------------------------+------------+------------------+----------------+----------------+----------------+
5 rows in set (0.16 sec)
```

# Mutex Contention

- Can use same table to see Waits for the thread you’re concerned about
  - Can get the portion of the time easily
- … or look at the global picture

```
mysql [localhost] {msandbox} (performance_schema) > select * from events_waits_summary_global_by_event_name where event_name like "%synch%"
order by sum_timer_wait  desc limit 5;
+---------------------------------------------+------------+------------------+----------------+----------------+----------------+
| EVENT_NAME                                  | COUNT_STAR | SUM_TIMER_WAIT   | MIN_TIMER_WAIT | AVG_TIMER_WAIT | MAX_TIMER_WAIT |
+---------------------------------------------+------------+------------------+----------------+----------------+----------------+
| wait/synch/mutex/mysys/THR_LOCK::mutex      |   21979626 | 5189584490777066 |         115230 |      236108608 |   348080318762 |
| wait/synch/mutex/sql/THD::LOCK_thd_data     |   58362822 |   39901276018172 |         107882 |         683364 |   239671140510 |
| wait/synch/mutex/innodb/trx_mutex           |    5647053 |    2940507577268 |          42418 |         520706 |    50872912740 |
| wait/synch/mutex/innodb/trx_undo_mutex      |    2852288 |    1660141672366 |          45090 |         581828 |    42721021500 |
| wait/synch/rwlock/innodb/index_tree_rw_lock |          2 |           593518 |         245490 |         296592 |         348028 |
+---------------------------------------------+------------+------------------+----------------+----------------+----------------+
5 rows in set (0.01 sec)
```

# Row Level Lock waits

- Might be better diagnosed using **INFORMATION_SCHEMA**

```
mysql [localhost] {msandbox} (performance_schema) > select *
from information_schema.innodb_trx limit 5 \G
*************************** 1. row ***************************
                    trx_id: 36751490
                 trx_state: LOCK WAIT
               trx_started: 2013-05-15 08:36:37
     trx_requested_lock_id: 36751490:6:5:19
          trx_wait_started: 2013-05-15 08:36:37
                trx_weight: 2
       trx_mysql_thread_id: 861
                 trx_query: select * from sbtest where id=18 for
update
       trx_operation_state: starting index read
         trx_tables_in_use: 1
         trx_tables_locked: 1
          trx_lock_structs: 2
     trx_lock_memory_bytes: 376
           trx_rows_locked: 1
         trx_rows_modified: 0
   trx_concurrency_tickets: 0
       trx_isolation_level: REPEATABLE READ
         trx_unique_checks: 1
    trx_foreign_key_checks: 1
trx_last_foreign_key_error: NULL
 trx_adaptive_hash_latched: 0
 trx_adaptive_hash_timeout: 10000
          trx_is_read_only: 0
trx_autocommit_non_locking: 0
```

```
mysql [localhost] {msandbox}
(performance_schema) > select * from
information_schema.INNODB_LOCK_WAITS limit 5
\G
*************************** 1. row
***************************
requesting_trx_id: 36751490
requested_lock_id: 36751490:6:5:19
  blocking_trx_id: 36751489
 blocking_lock_id: 36751489:6:5:19
1 row in set (0.00 sec)
```

# … Data in PERFORMANCE_SCHEMA

```
mysql [localhost] {msandbox} (performance_schema) > select * from events_waits_current where
thread_id=880 \G
*************************** 1. row ***************************
            THREAD_ID: 880
             EVENT_ID: 124
         END_EVENT_ID: NULL
           EVENT_NAME: wait/io/table/sql/handler
               SOURCE: handler.cc:2722
          TIMER_START: 345708272369645 70
            TIMER_END: NULL
           TIMER_WAIT: NULL
                SPINS: NULL
        OBJECT_SCHEMA: sbtest
          OBJECT_NAME: sbtest
           INDEX_NAME: PRIMARY
          OBJECT_TYPE: TABLE
OBJECT_INSTANCE_BEGIN: 140169480812144
     NESTING_EVENT_ID: 123
   NESTING_EVENT_TYPE: STAGE
            OPERATION: fetch
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
1 row in set (0.00 sec)
```

# Check out statement history

- Great to see what last statements given connection has ran!

```
mysql [localhost] {msandbox} (performance_schema) > select * from
events_statements_history where thread_id=880 \G
*************************** 1. row ***************************
              THREAD_ID: 880
               EVENT_ID: 109
           END_EVENT_ID: 144
             EVENT_NAME: statement/sql/select
                 SOURCE: mysqld.cc:923
            TIMER_START: 34586663679918000
              TIMER_END: 37065990748790000
             TIMER_WAIT: 2479327068872000
              LOCK_TIME: 141000000
               SQL_TEXT: select * from sbtest where id=18 for update
                 DIGEST: 16588172b60f779413ca98f5d620938a
            DIGEST_TEXT: SELECT * FROM `sbtest` WHERE `id` = ? FOR UPDATE
         CURRENT_SCHEMA: sbtest
…
ROWS_SENT: 1
          ROWS_EXAMINED: 1
NESTING_EVENT_ID: NULL
     NESTING_EVENT_TYPE: NULL
```

# Meta Data Locks

```
mysql [localhost] {msandbox} (performance_schema) > select * from events_waits_current where thread_id=880
\G
*************************** 1. row ***************************
            THREAD_ID: 880
             EVENT_ID: 260
         END_EVENT_ID: NULL
           EVENT_NAME: wait/synch/cond/sql/MDL_context::COND_wait_status
               SOURCE: mdl.cc:1306
          TIMER_START: 377081745071819 38
            TIMER_END: NULL
           TIMER_WAIT: NULL
                SPINS: NULL
        OBJECT_SCHEMA: NULL
          OBJECT_NAME: NULL
           INDEX_NAME: NULL
          OBJECT_TYPE: NULL
OBJECT_INSTANCE_BEGIN: 0
     NESTING_EVENT_ID: 259
   NESTING_EVENT_TYPE: STAGE
            OPERATION: timed_wait
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
1 row in set (0.00 sec)
```

# MDL Lock waits accounted !

```
*************************** 2. row ***************************
              THREAD_ID: 880
               EVENT_ID: 146
           END_EVENT_ID: 2012
             EVENT_NAME: statement/sql/truncate
                 SOURCE: mysqld.cc:923
            TIMER_START: 377054438143130000
              TIMER_END: 379881781928450000
             TIMER_WAIT: 282734378532000
              LOCK_TIME: 282345977000000
               SQL_TEXT: truncate sbtest
                 DIGEST: c36ce2ae8d78a3e3d79ec73e31142ca4
            DIGEST_TEXT: TRUNCATE `sbtest`
         CURRENT_SCHEMA: sbtest
            OBJECT_TYPE: NULL
          OBJECT_SCHEMA: NULL
            OBJECT_NAME: NULL
  OBJECT_INSTANCE_BEGIN: NULL
            MYSQL_ERRNO: 0
      RETURNED_SQLSTATE: 00000
           MESSAGE_TEXT: NULL
….
     NO_GOOD_INDEX_USED: 0
       NESTING_EVENT_ID: NULL
     NESTING_EVENT_TYPE: NULL
```

## Why use Summaries ?

- The “log” tables have best level of details
  - But they can “decay” way too quickly
  - **events_waits_history_long** set to hold 10000 events
  - Enough for 0.5 seconds for test workloads
    - Can be even less with heavy contention
    - “Stages” can be more verbose than waits for some workloads

# More on PERFORMANCE_SCHEMA

- MySQL Manual on Performance Schema
  - http://bit.ly/Uc7GIO
- Marc Alff's Blog
  - http://marcalff.blogspot.com/
- Mark Leith's Blog
  - http://www.markleith.co.uk/
- Presentations
  - MySQL Connect 2012
    - http://bit.ly/142DuIa
  - Percona Live 2013
    - http://bit.ly/12rZHwk

# More Resources

- Training from Percona
  - http://www.percona.com/training
- Percona Live, London,UK
  - Nov 11-12, 2013
  - http://www.percona.com/live/london-2013/home
- Percona Webinars
  - http://www.percona.com/webinars
- MySQL Performance Blog
  - http://www.mysqlperformanceblog.com/

## Thank You!

Peter Zaitsev
pz@percona.com