Profiling with Performance Schema

Mark Leith
Software Development Senior Manager

Program Agenda

- An Introduction to Performance Schema
- Performance Schema Configuration
- Profiling General Instance Activity
- Profiling Statement Activity

An Introduction to Performance Schema

 | Insert Information Protection Policy Classification from Slide 12

An Introduction to Performance Schema

- Performance Schema Overview

- A new default ***performance_schema*** schema within MySQL
- Tables use new ***PERFORMANCE_SCHEMA*** storage engine
  - Real engine, unlike INFORMATION_SCHEMA
- Records various run time statistics via in-built instrumentation points
- All recorded statistics are stored in fixed size ring buffers in memory
- Most instrumentation configuration can be done dynamically

An Introduction to Performance Schema

- Instrumentation Points
- At it's core, Performance Schema tracks ***latency*** for various ***events***
- Latency is exposed to ***picosecond*** precision (a *trillionth* of a second)
- In 5.5:
  - File I/O, Mutexes, Read/Write Locks, Conditions
- In 5.6:
  - Network I/O, Table I/O, Table Locks, Stages, Statements, Idle
- Also tracks other data as appropriate, like bytes, source position, etc.

An Introduction to Performance Schema

- Wait Events

Table Locks	wait/lock/table/%	(5.6)
Network IO	wait/io/socket/%	(5.6)
Table IO	wait/io/table/%	(5.6)
File IO	wait/io/file/%	(5.5)
Mutexes	wait/synch/mutex/%	(5.5)
Read/Write Locks	wait/synch/rwlock/%	(5.5)
Conditions	wait/synch/cond/%	(5.5)

An Introduction to Performance Schema

- Raw Wait Events – A Table IO Event Example

```
mysql> select * from events_waits_history where event_name like 'wait/io/table%'\G
*************************** 1. row ***************************
            THREAD_ID: 352950
             EVENT_ID: 913915
         END_EVENT_ID: 913915
           EVENT_NAME: wait/io/table/sql/handler
               SOURCE: handler.cc:2716
          TIMER_START: 1301573336396109228
            TIMER_END: 1301573336396938344
           TIMER_WAIT: 829116
                SPINS: NULL
        OBJECT_SCHEMA: mem30__quan
          OBJECT_NAME: normalized_statements_by_server_by_schema_data
           INDEX_NAME: PRIMARY
          OBJECT_TYPE: TABLE
OBJECT_INSTANCE_BEGIN: 360798712
     NESTING_EVENT_ID: 484888
   NESTING_EVENT_TYPE: STAGE
            OPERATION: fetch
      NUMBER_OF_BYTES: NULL
                FLAGS: NULL
```

(Event context, who, and in what level)

(Event type & origination)

(Timing – 829 nanoseconds)

(Database object the event was against)

(Object type, and individual instance)

(Whether the event was nested within another)

(Further info on the event, it's type, size, flags)

An Introduction to Performance Schema

- Statement and Stage Events (5.6+)
- Statements have two types of event
  - SQL Statements — statement/sql/%
  - COM Commands — statement/com/%
- Stages are the thread states (like SHOW PROFILE, but available across connections)
  - One form so far — stage/sql/%
  - Apart from... — stage/mysys/Waiting for table level lock

An Introduction to Performance Schema

- Raw Stage Events

```
mysql> select * from events_stages_history\G
*************************** 1. row ***************************
           THREAD_ID: 378012
            EVENT_ID: 273
        END_EVENT_ID: 273
          EVENT_NAME: stage/sql/preparing
              SOURCE: sql_optimizer.cc:483
         TIMER_START: 13801517561005622l4
           TIMER_END: 13801517561078456888
          TIMER_WAIT: 7283474
    NESTING_EVENT_ID: 249
NESTING_EVENT_TYPE: STATEMENT
*************************** 2. row ***************************
           THREAD_ID: 378012
            EVENT_ID: 274
        END_EVENT_ID: 274
          EVENT_NAME: stage/sql/executing
              SOURCE: sql_executor.cc:112
         TIMER_START: 13801517561078456888
           TIMER_END: 13801517561085078222
          TIMER_WAIT: 662134
    NESTING_EVENT_ID: 249
NESTING_EVENT_TYPE: STATEMENT
```

An Introduction to Performance Schema

- Raw Statement Events

```
mysql> select * from events_statements_history limit 1\G
*************************** 1. row ***************************
              THREAD_ID: 378012
               EVENT_ID: 249
           END_EVENT_ID: 288
             EVENT_NAME: statement/sql/select
                 SOURCE: mysqld.cc:907
            TIMER_START: 1380151755620846131
              TIMER_END: 1380151756820301872
             TIMER_WAIT: 1199455741
              LOCK_TIME: 0
               SQL_TEXT: select * from events_stages_history
                 DIGEST: f13b9e15a70df2108f6aa343860734fa
            DIGEST_TEXT: SELECT * FROM events_stages_history
         CURRENT_SCHEMA: performance_schema
            OBJECT_TYPE: NULL
          OBJECT_SCHEMA: NULL
            OBJECT_NAME: NULL
  OBJECT_INSTANCE_BEGIN: NULL
            MYSQL_ERRNO: 0
      RETURNED_SQLSTATE: NULL
           MESSAGE_TEXT: NULL
```

```
                  ERRORS: 0
                WARNINGS: 0
           ROWS_AFFECTED: 0
               ROWS_SENT: 125
           ROWS_EXAMINED: 125
CREATED_TMP_DISK_TABLES: 0
     CREATED_TMP_TABLES: 0
       SELECT_FULL_JOIN: 0
 SELECT_FULL_RANGE_JOIN: 0
           SELECT_RANGE: 0
     SELECT_RANGE_CHECK: 0
            SELECT_SCAN: 1
      SORT_MERGE_PASSES: 0
             SORT_RANGE: 0
              SORT_ROWS: 0
              SORT_SCAN: 0
          NO_INDEX_USED: 1
     NO_GOOD_INDEX_USED: 0
       NESTING_EVENT_ID: NULL
     NESTING_EVENT_TYPE: NULL
```

An Introduction to Performance Schema

Setup Tables

- Used to define certain configuration dynamically
- Can perform DML against these tables

```
mysql> SELECT table_name
    ->   FROM information_schema.tables
    ->  WHERE table_schema = 'performance_schema'
    ->    AND table_name LIKE 'setup%';
+-------------------+
| table_name        |
+-------------------+
| setup_actors      |
| setup_consumers   |
| setup_instruments |
| setup_objects     |
| setup_timers      |
+-------------------+
```

An Introduction to Performance Schema

Raw Data Tables

- Expose events, objects, or instances of instruments in a raw manner
- Allow seeing a (brief) history of raw event metrics as well

```
mysql> SELECT table_name
    ->   FROM information_schema.tables
    ->  WHERE table_schema = 'performance_schema'
    ->    AND table_name NOT LIKE 'setup%'
    ->    AND table_name NOT LIKE '%summary%';
+--------------------------------+
| table_name                     |
+--------------------------------+
| accounts                       |
| cond_instances                 |
| events_stages_current          |
| events_stages_history          |
| events_stages_history_long     |
| events_statements_current      |
| events_statements_history      |
| events_statements_history_long |
| events_waits_current           |
| events_waits_history           |
| events_waits_history_long      |
| file_instances                 |
| host_cache                     |
| hosts                          |
| mutex_instances                |
| performance_timers             |
| rwlock_instances               |
| session_account_connect_attrs  |
| session_connect_attrs          |
| socket_instances               |
| threads                        |
| users                          |
+--------------------------------+
```

An Introduction to Performance Schema

Summary Tables

- Summarize event information over various dimensions
- Useful for longer term monitoring of activity

```
mysql> SELECT table_name
    ->   FROM information_schema.tables
    ->  WHERE table_schema = 'performance_schema'
    ->    AND table_name LIKE '%summary%';
+----------------------------------------------------+
| table_name                                         |
+----------------------------------------------------+
| events_stages_summary_by_account_by_event_name     |
| events_stages_summary_by_host_by_event_name        |
| events_stages_summary_by_thread_by_event_name      |
| events_stages_summary_by_user_by_event_name        |
| events_stages_summary_global_by_event_name         |
| events_statements_summary_by_account_by_event_name |
| events_statements_summary_by_digest                |
| events_statements_summary_by_host_by_event_name    |
| events_statements_summary_by_thread_by_event_name  |
| events_statements_summary_by_user_by_event_name    |
| events_statements_summary_global_by_event_name     |
| 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          |
| file_summary_by_event_name                         |
| file_summary_by_instance                           |
| objects_summary_global_by_type                     |
| socket_summary_by_event_name                       |
| socket_summary_by_instance                         |
| table_io_waits_summary_by_index_usage              |
| table_io_waits_summary_by_table                    |
| table_lock_waits_summary_by_table                  |
+----------------------------------------------------+
```

Performance Schema Configuration

 | Insert Information Protection Policy Classification from Slide 12

Performance Schema Configuration

- Configuration Options
- There are three distinct types of configuration
- A way to configure ***how much memory to allocate*** for instruments
  - Set with various system variables in options files, not dynamic
- A way to ***enable/disable instrumentation at startup***
  - Again set within options files, only available as of 5.6
- A way to ***enable/disable instrumentation dynamically***
  - Set by updating various “setup” tables dynamically

Performance Schema Configuration

- Memory Configuration
- Memory usage falls in to two distinct categories
- How many “classes” and ”instances” of event types to track
  - Affects all data available in performance schema
- How much summary and history data you want to track
  - Affects how long you can see data for

Performance Schema Configuration

Class / Instance Config

- Classes count instrument implementations - “wait/io/file/sql/binlog”
- Instances are the different runtime instances of them - “/data/mysql/binlog.000001”

```
mysql> SELECT variable_name, variable_value
    ->   FROM information_schema.global_variables
    ->  WHERE variable_name LIKE 'perf%classes'
    ->     OR variable_name LIKE 'perf%instances'
    ->  ORDER BY variable_name;
+-----------------------------------------+----------------+
| variable_name                           | variable_value |
+-----------------------------------------+----------------+
| PERFORMANCE_SCHEMA_MAX_COND_CLASSES     | 80             |
| PERFORMANCE_SCHEMA_MAX_COND_INSTANCES   | 723            |
| PERFORMANCE_SCHEMA_MAX_FILE_CLASSES     | 50             |
| PERFORMANCE_SCHEMA_MAX_FILE_INSTANCES   | 1556           |
| PERFORMANCE_SCHEMA_MAX_MUTEX_CLASSES    | 200            |
| PERFORMANCE_SCHEMA_MAX_MUTEX_INSTANCES  | 3112           |
| PERFORMANCE_SCHEMA_MAX_RWLOCK_CLASSES   | 30             |
| PERFORMANCE_SCHEMA_MAX_RWLOCK_INSTANCES | 1667           |
| PERFORMANCE_SCHEMA_MAX_SOCKET_CLASSES   | 10             |
| PERFORMANCE_SCHEMA_MAX_SOCKET_INSTANCES | 123            |
| PERFORMANCE_SCHEMA_MAX_STAGE_CLASSES    | 150            |
| PERFORMANCE_SCHEMA_MAX_STATEMENT_CLASSES| 169            |
| PERFORMANCE_SCHEMA_MAX_TABLE_INSTANCES  | 445            |
| PERFORMANCE_SCHEMA_MAX_THREAD_CLASSES   | 50             |
| PERFORMANCE_SCHEMA_MAX_THREAD_INSTANCES | 167            |
+-----------------------------------------+----------------+
```

Performance Schema Configuration

Class / Instance Config

- Monitor the ***_lost** status variables opposite to see whether these need tweaking
- You will generally not need to modify classes – but instances could need tuning

```
mysql> SELECT variable_name, variable_value
    ->   FROM information_schema.global_status
    ->  WHERE variable_name LIKE 'perf%classes_lost'
    ->     OR variable_name LIKE 'perf%instances_lost';
+-------------------------------------------+----------------+
| variable_name                             | variable_value |
+-------------------------------------------+----------------+
| PERFORMANCE_SCHEMA_COND_CLASSES_LOST      | 0              |
| PERFORMANCE_SCHEMA_COND_INSTANCES_LOST    | 0              |
| PERFORMANCE_SCHEMA_FILE_CLASSES_LOST      | 0              |
| PERFORMANCE_SCHEMA_FILE_INSTANCES_LOST    | 438056         |
| PERFORMANCE_SCHEMA_MUTEX_CLASSES_LOST     | 0              |
| PERFORMANCE_SCHEMA_MUTEX_INSTANCES_LOST   | 0              |
| PERFORMANCE_SCHEMA_RWLOCK_CLASSES_LOST    | 0              |
| PERFORMANCE_SCHEMA_RWLOCK_INSTANCES_LOST  | 0              |
| PERFORMANCE_SCHEMA_SOCKET_CLASSES_LOST    | 0              |
| PERFORMANCE_SCHEMA_SOCKET_INSTANCES_LOST  | 0              |
| PERFORMANCE_SCHEMA_STAGE_CLASSES_LOST     | 0              |
| PERFORMANCE_SCHEMA_STATEMENT_CLASSES_LOST | 0              |
| PERFORMANCE_SCHEMA_TABLE_INSTANCES_LOST   | 0              |
| PERFORMANCE_SCHEMA_THREAD_CLASSES_LOST    | 0              |
| PERFORMANCE_SCHEMA_THREAD_INSTANCES_LOST  | 0              |
+-------------------------------------------+----------------+
```

Performance Schema Configuration

History / Summary Sizes

- Define total rows per type
- The *history_size variables are per-thread
- The *history_long_size is total rows
- Only settable in my.cnf/ini

```
mysql> SELECT variable_name, variable_value
    ->   FROM information_schema.global_variables
    ->  WHERE variable_name LIKE 'perf%'
    ->    AND variable_name NOT LIKE '%instances'
    ->    AND variable_name NOT LIKE '%classes'
    ->  ORDER BY variable_name;
+-------------------------------------------------------+----------------+
| variable_name                                         | variable_value |
+-------------------------------------------------------+----------------+
| PERFORMANCE_SCHEMA                                    | ON             |
| PERFORMANCE_SCHEMA_ACCOUNTS_SIZE                      | 100            |
| PERFORMANCE_SCHEMA_DIGESTS_SIZE                       | 10000          |
| PERFORMANCE_SCHEMA_EVENTS_STAGES_HISTORY_LONG_SIZE    | 10000          |
| PERFORMANCE_SCHEMA_EVENTS_STAGES_HISTORY_SIZE         | 10             |
| PERFORMANCE_SCHEMA_EVENTS_STATEMENTS_HISTORY_LONG_SIZE| 10000          |
| PERFORMANCE_SCHEMA_EVENTS_STATEMENTS_HISTORY_SIZE     | 10             |
| PERFORMANCE_SCHEMA_EVENTS_WAITS_HISTORY_LONG_SIZE     | 10000          |
| PERFORMANCE_SCHEMA_EVENTS_WAITS_HISTORY_SIZE          | 10             |
| PERFORMANCE_SCHEMA_HOSTS_SIZE                         | 100            |
| PERFORMANCE_SCHEMA_MAX_FILE_HANDLES                   | 32768          |
| PERFORMANCE_SCHEMA_MAX_TABLE_HANDLES                  | 256            |
| PERFORMANCE_SCHEMA_SESSION_CONNECT_ATTRS_SIZE         | 512            |
| PERFORMANCE_SCHEMA_SETUP_ACTORS_SIZE                  | 100            |
| PERFORMANCE_SCHEMA_SETUP_OBJECTS_SIZE                 | 100            |
| PERFORMANCE_SCHEMA_USERS_SIZE                         | 100            |
+-------------------------------------------------------+----------------+
```

Performance Schema Configuration

setup_timers

- Define the granularity of timing
- Set different timers for different types of instrumentation

```
mysql> SELECT * FROM setup_timers;
+-------------+-------------+
| NAME        | TIMER_NAME  |
+-------------+-------------+
| idle        | MICROSECOND |
| wait        | CYCLE       |
| stage       | MICROSECOND |
| statement   | MICROSECOND |
+-------------+-------------+
4 rows in set (0.00 sec)

mysql> UPDATE setup_timers
    ->    SET timer_name = 'millisecond'
    ->  WHERE name = 'idle';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM setup_timers;
+-------------+-------------+
| NAME        | TIMER_NAME  |
+-------------+-------------+
| idle        | MILLISECOND |
| wait        | CYCLE       |
| stage       | MICROSECOND |
| statement   | MICROSECOND |
+-------------+-------------+
4 rows in set (0.00 sec)
```

Performance Schema Configuration

performance_timers

- Show which timers are available
- Show the performance characteristics of the timer on the platform MySQL installed on

```
mysql> SELECT * FROM performance_timers\G
*************************** 1. row ***************************
     TIMER_NAME: CYCLE
 TIMER_FREQUENCY: 3090445149
TIMER_RESOLUTION: 1
  TIMER_OVERHEAD: 21
*************************** 2. row ***************************
     TIMER_NAME: NANOSECOND
 TIMER_FREQUENCY: NULL
TIMER_RESOLUTION: NULL
  TIMER_OVERHEAD: NULL
*************************** 3. row ***************************
     TIMER_NAME: MICROSECOND
 TIMER_FREQUENCY: 3020537
TIMER_RESOLUTION: 1
  TIMER_OVERHEAD: 21
*************************** 4. row ***************************
     TIMER_NAME: MILLISECOND
 TIMER_FREQUENCY: 1035
TIMER_RESOLUTION: 15
  TIMER_OVERHEAD: 151
*************************** 5. row ***************************
     TIMER_NAME: TICK
 TIMER_FREQUENCY: 959
TIMER_RESOLUTION: 1
  TIMER_OVERHEAD: 9223372036854775807
5 rows in set (0.00 sec)
```

(Good Choice)

(Not Available)

(Good Choice)

(High Overhead)

(Just... Don't.)

Performance Schema Configuration

- **performance_timers**
  - Show which timers are available
  - Show the performance characteristics of the timer on the platform MySQL installed on

```
mysql> SELECT * FROM performance_timers\G
*************************** 1. row ***************************
     TIMER_NAME: CYCLE
 TIMER_FREQUENCY: 3090445149
TIMER_RESOLUTION: 1
  TIMER_OVERHEAD: 21
*************************** 2. row ***************************
     TIMER_NAME: NANOSECOND
 TIMER_FREQUENCY: NULL
TIMER_RESOLUTION: NULL
  TIMER_OVERHEAD: NULL
*************************** 3. row ***************************
     TIMER_NAME: MICROSECOND
 TIMER_FREQUENCY: 3020537
TIMER_RESOLUTION: 1
  TIMER_OVERHEAD: 21
*************************** 4. row ***************************
     TIMER_NAME: MILLISECOND
 TIMER_FREQUENCY: 1035
TIMER_RESOLUTION: 15
  TIMER_OVERHEAD: 151
*************************** 5. row ***************************
     TIMER_NAME: TICK
 TIMER_FREQUENCY: 959
TIMER_RESOLUTION: 1
  TIMER_OVERHEAD: 9223372036854775807
5 rows in set (0.00 sec)
```

(Good Choice)

(Not Available)

(Good Choice)

(High Overhead)

(Just... Don't.)

Performance Schema Configuration

setup_instruments

- Turn on/off individual instruments, and timing, dynamically
- Use UPDATE to modify

```
mysql> SELECT * FROM setup_instruments LIMIT 20;
+-----------------------------------------------------+---------+-------+
| NAME                                                | ENABLED | TIMED |
+-----------------------------------------------------+---------+-------+
| wait/synch/mutex/sql/PAGE::lock                     | YES     | YES   |
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_sync         | YES     | YES   |
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_active       | YES     | YES   |
| wait/synch/mutex/sql/TC_LOG_MMAP::LOCK_pool         | YES     | YES   |
| wait/synch/mutex/sql/LOCK_des_key_file              | YES     | YES   |
| wait/synch/mutex/sql/BINARY_LOG::LOCK_commit        | YES     | YES   |
| wait/synch/mutex/sql/BINARY_LOG::LOCK_commit_queue  | YES     | YES   |
| wait/synch/mutex/sql/BINARY_LOG::LOCK_done          | YES     | YES   |
| wait/synch/mutex/sql/BINARY_LOG::LOCK_flush_queue   | YES     | YES   |
| wait/synch/mutex/sql/BINARY_LOG::LOCK_index         | YES     | YES   |
| wait/synch/mutex/sql/BINARY_LOG::LOCK_log           | YES     | YES   |
| wait/synch/mutex/sql/BINARY_LOG::LOCK_sync          | YES     | YES   |
| wait/synch/mutex/sql/BINARY_LOG::LOCK_sync_queue    | YES     | YES   |
| wait/synch/mutex/sql/RELAY_LOG::LOCK_commit         | YES     | YES   |
| wait/synch/mutex/sql/RELAY_LOG::LOCK_commit_queue   | YES     | YES   |
| wait/synch/mutex/sql/RELAY_LOG::LOCK_done           | YES     | YES   |
| wait/synch/mutex/sql/RELAY_LOG::LOCK_flush_queue    | YES     | YES   |
| wait/synch/mutex/sql/RELAY_LOG::LOCK_index          | YES     | YES   |
| wait/synch/mutex/sql/RELAY_LOG::LOCK_log            | YES     | YES   |
| wait/synch/mutex/sql/RELAY_LOG::LOCK_sync           | YES     | YES   |
+-----------------------------------------------------+---------+-------+
```

Performance Schema Configuration

setup_consumers

- Define how much to record in history (again, UPDATE to modify)
- ***Statement Digests*** also enabled

```
mysql> 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     |
+--------------------------------+---------+
```

Performance Schema Configuration

- **setup_actors (New in 5.6)**
  - Define exactly which connections to monitor
  - By default monitors all connections (% is a wildcard)
  - Modifications only apply to new connections

```
mysql> SELECT * FROM setup_actors;
+------+------+------+
| HOST | USER | ROLE |
+------+------+------+
| %    | %    | %    |
+------+------+------+
1 row in set (0.00 sec)

mysql> DELETE FROM setup_actors;
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO setup_actors
    -> VALUES ('%', 'mark', '%');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM setup_actors;
+------+------+------+
| HOST | USER | ROLE |
+------+------+------+
| %    | mark | %    |
+------+------+------+
1 row in set (0.00 sec)
```

Performance Schema Configuration

- **setup_objects (New in 5.6)**
  - Define exactly which database objects to monitor
  - Filters standard databases by default
  - Allows explicit enable/disable, matches on most specific

```
mysql> 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)

mysql> INSERT INTO setup_objects
    -> VALUES ('TABLE', 'myschema', 'logging', 'NO', 'NO');
Query OK, 1 row affected (0.00 sec)

mysql> 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   |
| TABLE       | myschema           | logging     | NO      | NO    |
+-------------+--------------------+-------------+---------+-------+
5 rows in set (0.00 sec)
```

Performance Schema Configuration

- Altering Enabled Instrumentation At Startup (5.6)
- Enable/Disable instruments at startup
  - performance_schema_instrument='instrument_name=value'
    - Value = [on | true | 1] / [off | false | 0] / [counted] (no timing)
    - Instruments can include wildcards "wait/synch/mutex/%"
- Enable/Disable consumers at startup
  - performance_schema_consumer_***consumer_name***=*value*
  - *performance_schema_consumer_events_stages_current=on*

Performance Schema Configuration

- Finally, grab ps_helper!
- A collection of functions / views / procedures to help with P_S
- I will be using the functions in the examples following for formatting
- Versions available for both 5.5 and 5.6
- http://www.markleith.co.uk/ps_helper/

Profiling General Instance Activity

Profiling General Instance Activity

- Defining what you want/need to monitor
- It's easy to just enable everything, but there are overhead concerns
  - On busy systems mutexes can be locked ***millions of times a second***
- For busy systems, it's best to just pick higher latency event types
  - Statements, Stages, Table IO, File IO, maybe Network IO
  - It's good to enable all *current consumers, and statement/stage history
- For concurrency issues toggle other instruments on as needed basis

Profiling General Instance Activity

- Different Ways of Monitoring
- Once you've narrowed down what you are interested in, there are two ways to start monitoring
- View raw data in the summary views
  - This gives you an overall picture of usage on the instance
- Snapshot data, and compute deltas over time
  - This gives you an idea of the rates of changes for events
- Let's start with viewing raw summary data..

Profiling General Instance Activity

- Top Waits by Latency – events_waits_summary_global_by_event_name

```
mysql> SELECT event_name AS event,
    ->        count_star AS total_events,
    ->        format_time(sum_timer_wait) AS total_latency,
    ->        format_time(avg_timer_wait) AS avg_latency,
    ->        format_time(max_timer_wait) AS max_latency
    ->   FROM performance_schema.events_waits_summary_global_by_event_name
    ->  WHERE event_name != 'idle'
    ->  ORDER BY sum_timer_wait DESC LIMIT 5;
+--------------------------------------+--------------+---------------+-------------+-------------+
| event                                | total_events | total_latency | avg_latency | max_latency |
+--------------------------------------+--------------+---------------+-------------+-------------+
| wait/io/file/myisam/dfile            |   3623721056 | 00:47:49.09   | 791.70 ns   | 312.96 ms   |
| wait/io/table/sql/handler            |     69879369 | 00:44:55.81   | 38.58 us    | 879.49 ms   |
| wait/io/file/innodb/innodb_log_file  |     28286631 | 00:37:57.13   | 80.50 us    | 476.00 ms   |
| wait/io/socket/sql/client_connection |    201937035 | 00:18:44.59   | 5.57 us     | 1.27 s      |
| wait/io/file/innodb/innodb_data_file |      2835077 | 00:08:15.34   | 174.72 us   | 455.22 ms   |
+--------------------------------------+--------------+---------------+-------------+-------------+
```

Only using InnoDB, this must be from temp tables

Profiling General Instance Activity

- Analyzing Global Waits
- Some mutex events that can affect global throughput (if high in list):
  - wait/synch/mutex/innodb/buf_pool_mutex
    - ***Increase innodb_buffer_pool_instances***
  - wait/synch/mutex/sql/Query_cache::structure_guard_mutex
    - ***Disable the Query Cache***
  - wait/synch/mutex/myisam/MYISAM_SHARE::intern_lock
    - ***Use InnoDB ;)***

Profiling General Instance Activity

- Analyzing Global Waits
- Some File IO events to watch for:
  - wait/io/file/sql/FRM
    - ***Tune table_open_cache / table_definition_cache***
  - wait/io/file/sql/file_parser (View definition parsing)
    - ***If high on 5.5, upgrade to 5.6 (which caches these)***
  - wait/io/file/sql/query_log / wait/io/file/sql/slow_log
    - ***Disable General / Slow query logs***

Profiling General Instance Activity

- Top Files by Total IO – file_summary_by_instance

```
mysql> SELECT format_path(file_name) AS file,
    ->        count_read,
    ->        format_bytes(sum_number_of_bytes_read) AS total_read,
    ->        format_bytes(IFNULL(sum_number_of_bytes_read / count_read, 0)) AS avg_read,
    ->        count_write,
    ->        format_bytes(sum_number_of_bytes_write) AS total_written,
    ->        format_bytes(IFNULL(sum_number_of_bytes_write / count_write, 0.00)) AS avg_write,
    ->        format_bytes(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 LIMIT 5;
+-----------------------------------------------------------------+-----------+------------+-----------+-------------+---------------+-----------+-----------+-----------+
| file                                                            | count_read | total_read | avg_read  | count_write | total_written | avg_write | total     | write_pct |
+-----------------------------------------------------------------+-----------+------------+-----------+-------------+---------------+-----------+-----------+-----------+
| @@datadir/ibdata1                                               |       801 | 14.48 MiB  | 18.52 KiB |    46197285 | 1.66 TiB      | 38.66 KiB | 1.66 TiB  |    100.00 |
| @@datadir/mem__events/events.ibd                                |       293 | 4.58 MiB   | 16.00 KiB |     8066205 | 123.42 GiB    | 16.04 KiB | 123.43 GiB|    100.00 |
| @@datadir/mem__quan/normalized_statements_by_server_by_schema_data.ibd |     5 | 80.00 KiB  | 16.00 KiB |     4536985 | 70.61 GiB     | 16.32 KiB | 70.61 GiB |    100.00 |
| @@datadir/ib_logfile0                                           |         6 | 68.00 KiB  | 11.33 KiB |    67011933 | 64.79 GiB     | 1.01 KiB  | 64.79 GiB |    100.00 |
| @@datadir/ib_logfile1                                           |         0 | 0 bytes    | 0 bytes   |    66779384 | 64.70 GiB     | 1.02 KiB  | 64.70 GiB |    100.00 |
+-----------------------------------------------------------------+-----------+------------+-----------+-------------+---------------+-----------+-----------+-----------+
```

High IO on per-tablespace tables are candidates for a separate mountpoint / disk with “DATA DIRECTORY”

Profiling General Instance Activity

- Analyzing User Activity
- All event_* summaries are give a number of dimensions to view data
- To analyze connection activity you can do this in 3 ways
  - By User
  - By Host
  - By Account (user@host)
- The following examples are by user, but could be replaced with the host, or account, summary views

Profiling General Instance Activity

- Top Users by IO Latency (or any other event class really) - events_waits_summary_by_user_by_event_name

```
mysql> SELECT user, SUM(count_star) AS count,
    ->        format_time(SUM(sum_timer_wait)) as total_latency
    ->   FROM performance_schema.events_waits_summary_by_user_by_event_name
    ->  WHERE event_name LIKE 'wait/io/file/%'
    ->    AND user IS NOT NULL
    ->  GROUP BY user
    ->  ORDER BY SUM(sum_timer_wait) DESC;
+-----------------+------------+---------------+
| user            | count      | total_latency |
+-----------------+------------+---------------+
| mem3            | 3046722930 | 00:41:57.51   |
| service_manager |    7131542 | 00:09:08.22   |
| mark            |        234 | 1.43 ms       |
+-----------------+------------+---------------+
```

Replace with other event classes here

Profiling General Instance Activity

- Top Users by Statement Latency - events_statements_summary_by_user_by_event_name

```
mysql> SELECT user,
    ->        SUM(count_star) AS total_statements,
    ->        format_time(SUM(sum_timer_wait)) AS total_latency,
    ->        format_time(SUM(sum_timer_wait) / SUM(count_star)) AS avg_latency
    ->   FROM performance_schema.events_statements_summary_by_user_by_event_name
    ->  WHERE user IS NOT NULL
    ->  GROUP BY user
    ->  ORDER BY SUM(sum_timer_wait) DESC
    ->  LIMIT 5;
+-----------------+------------------+---------------+-------------+
| user            | total_statements | total_latency | avg_latency |
+-----------------+------------------+---------------+-------------+
| mem3            |       1167229105 | 193.17h       | 595.78 us   |
| root            |             2104 | 00:06:11.61   | 176.62 ms   |
| service_manager |           313171 | 00:05:05.65   | 975.97 us   |
| agent_limit     |           495270 | 00:03:51.77   | 467.96 us   |
| mark            |              227 | 801.68 ms     | 3.53 ms     |
+-----------------+------------------+---------------+-------------+
```

Profiling General Instance Activity

- Top Users by Connections – users, hosts, accounts tables

```
mysql> SELECT * FROM users WHERE user IS NOT NULL ORDER BY current_connections DESC;
+-----------------+---------------------+--------------------+
| USER            | CURRENT_CONNECTIONS | TOTAL_CONNECTIONS |
+-----------------+---------------------+--------------------+
| mem3            |                  24 |               5956 |
| service_manager |                   7 |              10772 |
| mark            |                   1 |                  1 |
+-----------------+---------------------+--------------------+
3 rows in set (0.07 sec)

mysql> SELECT * FROM accounts WHERE user IS NOT NULL ORDER BY current_connections DESC;
+-----------------+-----------------------------------------------------------+---------------------+--------------------+
| USER            | HOST                                                      | CURRENT_CONNECTIONS | TOTAL_CONNECTIONS |
+-----------------+-----------------------------------------------------------+---------------------+--------------------+
| mem3            | [redacted].oracle.com                                     |                  24 |               5956 |
| service_manager | [redacted].oracle.com                                     |                   6 |              10771 |
| service_manager | localhost                                                 |                   1 |                  1 |
| mark            | [redacted].vpn.oracle.com                                 |                   1 |                  1 |
+-----------------+-----------------------------------------------------------+---------------------+--------------------+
```

Profiling General Instance Activity

- Analyzing Table Activity
- 3 summary views have been added in 5.6, to compliment Table IO
- objects_summary_global_by_type
  - A high level aggregate view of database object latency
- table_io_waits_summary_by_table
  - A detailed aggregate by table, breaking down reads, writes, etc.
- table_io_waits_summary_by_index_usage
  - Further breaking down usage per index

Profiling General Instance Activity

- Top Tables by Latency - objects_summary_global_by_type

```
mysql> SELECT object_schema AS db_name,
    ->        object_name AS table_name,
    ->        count_star AS total_events,
    ->        format_time(sum_timer_wait) AS total_latency,
    ->        format_time(avg_timer_wait) AS avg_latency,
    ->        format_time(max_timer_wait) AS max_latency
    ->   FROM performance_schema.objects_summary_global_by_typ
    ->  ORDER BY sum_timer_wait DESC LIMIT 5;
+---------------+-------------------------------+-------------+---------------+-------------+-------------+
| db_name       | table_name                    | total_events | _latency     | avg_latency | max_latency |
+---------------+-------------------------------+-------------+---------------+-------------+-------------+
| mem23tyr55    | inventory_instance_attributes |    65321697 | 00:22:36.90   | 20.77 us    | 879.49 ms   |
| mem23tyr55    | dc_p_string                   |    86563785 | 00:13:50.52   | 9.59 us     | 324.01 ms   |
| mem23tyr55    | dc_p_long                     |    18174843 | 00:09:44.27   | 32.15 us    | 35.07 ms    |
| mem__inventory| MysqlServer                   |      734370 | 23.70 s       | 32.27 us    | 111.50 ms   |
| mem23tyr55    | dc_p_double                   |      717954 | 21.09 s       | 29.37 us    | 34.84 ms    |
+---------------+-------------------------------+-------------+---------------+-------------+-------------+
```

Less events, yet higher latency is a sign that there could be concurrency issues

Profiling General Instance Activity

- Table Breakdown by Usage - table_io_waits_summary_by_table

```
mysql> SELECT object_schema, object_name,
    ->        count_fetch as selects, format_time(sum_timer_fetch) as select_latency,
    ->        count_insert as inserts, format_time(sum_timer_insert) as insert_latency,
    ->        count_update as updates, format_time(sum_timer_update) as update_latency,
    ->        count_delete as deletes, format_time(sum_timer_delete) as delete_latency
    ->   FROM performance_schema.table_io_waits_summary_by_table
    ->  ORDER BY sum_timer_wait DESC LIMIT 5;
+---------------+--------------------+---------+---------------+---------+---------------+---------+---------------+---------+---------------+
| object_schema | object_name        | selects | select_latency | inserts | insert_latency | updates | update_latency | deletes | delete_latency |
+---------------+--------------------+---------+---------------+---------+---------------+---------+---------------+---------+---------------+
| mem__inventory | MysqlServer        | 1251460 | 43.41 s       |      51 | 22.44 ms      |  167136 | 15.02 s       |       0 | 0 ps          |
| mem__events    | events             |  293364 | 5.38 s        |   16128 | 2.79 s        |  277428 | 20.23 s       |       0 | 0 ps          |
| mem__events    | subjects           | 1968949 | 16.38 s       |     322 | 2.17 s        |       0 | 0 ps          |       0 | 0 ps          |
| mem__inventory | Os                 | 1093048 | 12.70 s       |      30 | 4.51 ms       |   42419 | 2.84 s        |       0 | 0 ps          |
| mem__inventory | Network_v4Addresses |  137505 | 916.01 ms     |   63768 | 9.42 s        |       0 | 0 ps          |   63665 | 1.97 s        |
+---------------+--------------------+---------+---------------+---------+---------------+---------+---------------+---------+---------------+
```

Profiling General Instance Activity

Tables with Full Scans

- Search in the "table_io_waits_summary_by_index_usage" table **where index_name is null**
- This is a catch all row for rows read without indexes

```
mysql> SELECT object_schema,
    ->        object_name,
    ->        count_read AS rows_full_scanned
    ->   FROM performance_schema.table_io_waits_summary_by_index_usage
    ->  WHERE index_name IS NULL
    ->    AND count_read > 0
    ->  ORDER BY count_read DESC LIMIT 5;
+---------------+-----------------+------------------+
| object_schema | object_name     | rows_full_scanned |
+---------------+-----------------+------------------+
| mem23tyr55    | rule_alarms     |          5207641 |
| mem23tyr55    | dc_p_string     |          3989361 |
| mem_inventory | MysqlServer     |           153463 |
| mem_advisors  | advisor_schedules |          99974 |
| mem_inventory | Agent           |            43464 |
+---------------+-----------------+------------------+
```

Profiling General Instance Activity

Tables with Unused Indexes

- Search for indexes in the "table_io_waits_summary_by_index_usage" **where count_star = 0**
- *Should ensure that you have a representative time frame before using this!*

```
mysql> SELECT object_schema,
    ->        object_name,
    ->        index_name
    ->   FROM performance_schema.table_io_waits_summary_by_index_usage
    ->  WHERE index_name IS NOT NULL
    ->    AND count_star = 0
    ->  ORDER BY object_schema, object_name limit 5;
+---------------+-------------+------------+
| 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 |
+---------------+-------------+------------+
```

Profiling General Instance Activity

- **host_cache (New in 5.6)**
  - Exposes hosts cached for DNS
  - Counts errors, per error type
  - Shows when errors started happening
  - Compare *sum_connect_errors* to *max_connect_errors*

```
mysql> DESC host_cache;
+-----------------------------------------+------------------+
| Field                                   | Type             |
+-----------------------------------------+------------------+
| IP                                      | varchar(64)      |
| HOST                                    | varchar(255)     |
| HOST_VALIDATED                          | enum('YES','NO') |
| SUM_CONNECT_ERRORS                      | bigint(20)       |
| COUNT_HOST_BLOCKED_ERRORS               | bigint(20)       |
| COUNT_NAMEINFO_TRANSIENT_ERRORS         | bigint(20)       |
| COUNT_NAMEINFO_PERMANENT_ERRORS         | bigint(20)       |
| COUNT_FORMAT_ERRORS                     | bigint(20)       |
| COUNT_ADDRINFO_TRANSIENT_ERRORS         | bigint(20)       |
| COUNT_ADDRINFO_PERMANENT_ERRORS         | bigint(20)       |
| COUNT_FCRDNS_ERRORS                     | bigint(20)       |
| COUNT_HOST_ACL_ERRORS                   | bigint(20)       |
| COUNT_NO_AUTH_PLUGIN_ERRORS             | bigint(20)       |
| COUNT_AUTH_PLUGIN_ERRORS                | bigint(20)       |
| COUNT_HANDSHAKE_ERRORS                  | bigint(20)       |
| COUNT_PROXY_USER_ERRORS                 | bigint(20)       |
| COUNT_PROXY_USER_ACL_ERRORS             | bigint(20)       |
| COUNT_AUTHENTICATION_ERRORS             | bigint(20)       |
| COUNT_SSL_ERRORS                        | bigint(20)       |
| COUNT_MAX_USER_CONNECTIONS_ERRORS       | bigint(20)       |
| COUNT_MAX_USER_CONNECTIONS_PER_HOUR_ERRORS | bigint(20)    |
| COUNT_DEFAULT_DATABASE_ERRORS           | bigint(20)       |
| COUNT_INIT_CONNECT_ERRORS               | bigint(20)       |
| COUNT_LOCAL_ERRORS                      | bigint(20)       |
| COUNT_UNKNOWN_ERRORS                    | bigint(20)       |
| FIRST_SEEN                              | timestamp        |
| LAST_SEEN                               | timestamp        |
| FIRST_ERROR_SEEN                        | timestamp        |
| LAST_ERROR_SEEN                         | timestamp        |
+-----------------------------------------+------------------+
29 rows in set (0.02 sec)
```

Profiling General Instance Activity

- Computing Rates of Change

- Looking at the summary data gives a great idea of overall loads
- Looking at the rates of change for certain events can give you a better idea of ***utilization*** in certain cases
- For example, the ***wait/synch/cond/sql/MYSQL_RELAY_LOG::update_cond*** event tracks the time spent waiting in the replication SQL thread for new events to be written to the relay log – inversely, this is the SQL thread idle time

Profiling General Instance Activity

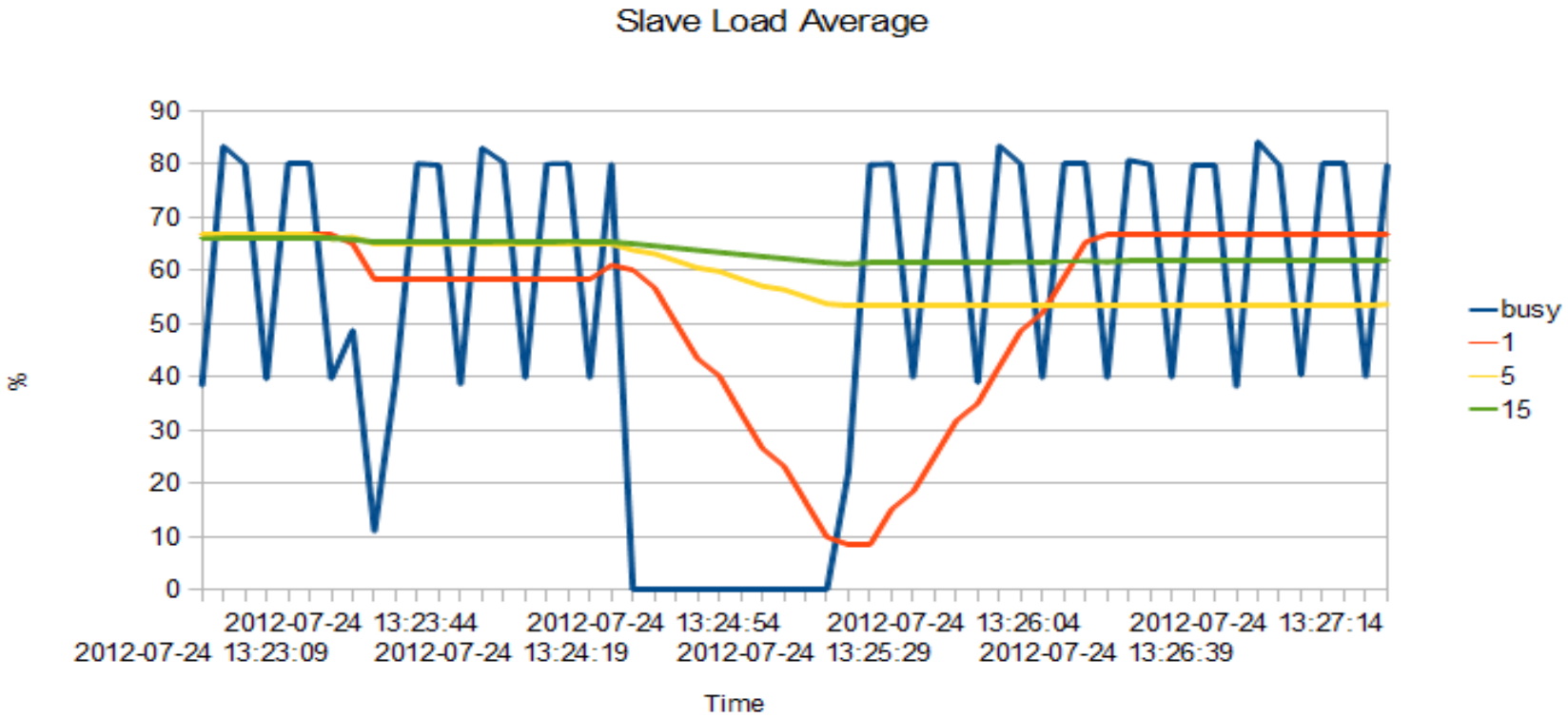
Slave Load Average

- An example of monitoring this over time
- Given the history, we can also compute *moving averages*

```
mysql> select tstamp, busy_pct, one_min_avg, five_min_avg, fifteen_min_avg
    ->   from slave_sql_load_average
    ->  order by tstamp desc limit 10;
+---------------------+----------+-------------+-------------+----------------+
| tstamp              | busy_pct | one_min_avg | five_min_avg | fifteen_min_avg |
+---------------------+----------+-------------+-------------+----------------+
| 2012-07-24 14:00:29 |    79.94 |       67.10 |       66.92 |          66.25 |
| 2012-07-24 14:00:24 |    39.97 |       67.10 |       66.93 |          66.15 |
| 2012-07-24 14:00:19 |    79.92 |       67.11 |       66.92 |          66.34 |
| 2012-07-24 14:00:14 |    84.31 |       67.09 |       66.93 |          66.24 |
| 2012-07-24 14:00:09 |    38.62 |       66.98 |       66.93 |          66.11 |
| 2012-07-24 14:00:04 |    83.26 |       67.03 |       66.95 |          66.31 |
| 2012-07-24 13:59:59 |    79.30 |       66.73 |       66.90 |          66.19 |
| 2012-07-24 13:59:54 |    39.97 |       66.81 |       66.91 |          66.09 |
| 2012-07-24 13:59:49 |    79.94 |       66.81 |       66.91 |          66.28 |
| 2012-07-24 13:59:44 |    79.95 |       66.80 |       66.91 |          66.18 |
+---------------------+----------+-------------+-------------+----------------+
```

http://www.markleith.co.uk/2012/07/24/a-mysql-replication-load-average-with-performance-schema/

Profiling Statement Activity



Profiling General Instance Activity

- Computing Rates of Change

- To look at how things change over time, record the events you are interested in within a **persistent** history table
- Get current stats from performance schema for a specific thread, the last row of history, compute the ***time delta*** and ***event count*** delta
- If interested in calculating percentages et al, copy the approach shown in my replication load average – you have to take in to account waiting 100% on other events, or just this event, etc.

Profiling General Instance Activity

- Computing Rates of Change Caveats

- Note, this method can not be used generally, you have to be able to correlate it to specific thread(s)
- As latency is per thread, and you have to compare to a known time delta, summary views do not give this to you (you don't get thread counts per interval)
- However, when tied to a specific threads such as the SQL thread, you can compute other events in the same way – file IO latency for example

Profiling Statement Activity

Profiling Statement Activity

- Statement Profiling Options
- The new instrumentation in 5.6 gives many options
- See statements running currently with events_statements_current
- Summary views by user, host, account
- Statement history with events_statements_history*
  - And link to stages, and waits, with "***nested events***"
- ***A normalized view with events_statements_summary_by_digest***

Profiling Statement Activity

- **events_statements_current**

- Exposes the currently executing, or last executed (if no current) statement per thread – executing = "timer_end IS NULL"
- Stats are incremented "live" for running statements

```
mysql> desc events_statements_current;
+---------------------------+-----------------------------------+
| Field                     | Type                              |
+---------------------------+-----------------------------------+
| THREAD_ID                 | int(11)                           |
| EVENT_ID                  | bigint(20) unsigned               |
| END_EVENT_ID              | bigint(20) unsigned               |
| EVENT_NAME                | varchar(128)                      |
| SOURCE                    | varchar(64)                       |
| TIMER_START               | bigint(20) unsigned               |
| TIMER_END                 | bigint(20) unsigned               |
| TIMER_WAIT                | bigint(20) unsigned               |
| LOCK_TIME                 | bigint(20) unsigned               |
| SQL_TEXT                  | longtext                          |
| DIGEST                    | varchar(32)                       |
| DIGEST_TEXT               | longtext                          |
| CURRENT_SCHEMA            | varchar(64)                       |
| OBJECT_TYPE               | varchar(64)                       |
| OBJECT_SCHEMA             | varchar(64)                       |
| OBJECT_NAME               | varchar(64)                       |
| OBJECT_INSTANCE_BEGIN     | bigint(20) unsigned               |
| MYSQL_ERRNO               | int(11)                           |
| RETURNED_SQLSTATE         | varchar(5)                        |
| MESSAGE_TEXT              | varchar(128)                      |
| ERRORS                    | bigint(20) unsigned               |
| WARNINGS                  | bigint(20) unsigned               |
| ROWS_AFFECTED             | bigint(20) unsigned               |
| ROWS_SENT                 | bigint(20) unsigned               |
| ROWS_EXAMINED             | bigint(20) unsigned               |
| CREATED_TMP_DISK_TABLES   | bigint(20) unsigned               |
| CREATED_TMP_TABLES        | bigint(20) unsigned               |
| SELECT_FULL_JOIN          | bigint(20) unsigned               |
| SELECT_FULL_RANGE_JOIN    | bigint(20) unsigned               |
| SELECT_RANGE              | bigint(20) unsigned               |
| SELECT_RANGE_CHECK        | bigint(20) unsigned               |
| SELECT_SCAN               | bigint(20) unsigned               |
| SORT_MERGE_PASSES         | bigint(20) unsigned               |
| SORT_RANGE                | bigint(20) unsigned               |
| SORT_ROWS                 | bigint(20) unsigned               |
| SORT_SCAN                 | bigint(20) unsigned               |
| NO_INDEX_USED             | bigint(20) unsigned               |
| NO_GOOD_INDEX_USED        | bigint(20) unsigned               |
| NESTING_EVENT_ID          | bigint(20) unsigned               |
| NESTING_EVENT_TYPE        | enum('STATEMENT','STAGE','WAIT')  |
+---------------------------+-----------------------------------+
```

Profiling Statement Activity

- Generating a Detailed Processlist by JOINing to threads

```
mysql> SELECT pps.thread_id AS thd_id,
    ->        pps.processlist_id AS conn_id,
    ->        IF(pps.name = 'thread/sql/one_connection',
    ->           CONCAT(pps.processlist_user, '@', pps.processlist_host),
    ->           REPLACE(pps.name, 'thread/', '')) user,
    ->        pps.processlist_db AS db,
    ->        pps.processlist_command AS command,
    ->        pps.processlist_state AS state,
    ->        pps.processlist_time AS time,
    ->        format_statement(pps.processlist_info) AS current_statement,
    ->        IF(esc.timer_wait IS NOT NULL,
    ->           format_statement(esc.sql_text),
    ->           NULL) AS last_statement,
    ->        IF(esc.timer_wait IS NOT NULL,
    ->           format_time(esc.timer_wait),
    ->           NULL) as last_statement_latency,
    ->        format_time(esc.lock_time) AS lock_latency,
    ->        esc.rows_examined,
    ->        esc.rows_sent,
    ->        esc.rows_affected,
    ->        esc.created_tmp_tables AS tmp_tables,
    ->        esc.created_tmp_disk_tables as tmp_disk_tables,
    ->        IF(esc.no_good_index_used > 0 OR esc.no_index_used > 0,
    ->           'YES', 'NO') AS full_scan,
    ->        ewc.event_name AS last_wait,
    ->        IF(ewc.timer_wait IS NULL AND ewc.event_name IS NOT NULL,
    ->           'Still Waiting',
    ->           format_time(ewc.timer_wait)) last_wait_latency,
    ->        ewc.source
    ->   FROM performance_schema.threads AS pps
    ->   LEFT JOIN performance_schema.events_waits_current AS ewc USING (thread_id)
    ->   LEFT JOIN performance_schema.events_statements_current as esc USING (thread_id)
    -> ORDER BY pps.processlist_time DESC, last_wait_latency DESC;
```

```
*************************** 24. row ***************************
               thd_id: 71454
              conn_id: 71435
                 user: mem3@[redacted].oracle.com
                   db: mysql
              command: Query
                state: executing
                 time: 2
    current_statement: SELECT page_type,         IF(t ...
       last_statement: NULL
last_statement_latency: NULL
         lock_latency: 274.00 us
        rows_examined: 0
            rows_sent: 0
        rows_affected: 0
           tmp_tables: 3
      tmp_disk_tables: 1
            full_scan: YES
            last_wait: wait/io/file/myisam/dfile
    last_wait_latency: 531.34 us
               source: mf_iocache.c:1783
```

Writing temp table to disk

Profiling Statement Activity

- History of completed statements (*last 10 per session by default!*)

```
mysql> select * from events_statements_history limit 1\G
*************************** 1. row ***************************
              THREAD_ID: 378012
               EVENT_ID: 249
           END_EVENT_ID: 288
             EVENT_NAME: statement/sql/select
                 SOURCE: mysqld.cc:907
            TIMER_START: 1380151755620846131
              TIMER_END: 1380151756820301872
             TIMER_WAIT: 1199455741
              LOCK_TIME: 0
               SQL_TEXT: select * from events_stages_history
                 DIGEST: f13b9e15a70df2108f6aa343860734fa
            DIGEST_TEXT: SELECT * FROM events_stages_history
         CURRENT_SCHEMA: performance_schema
            OBJECT_TYPE: NULL
          OBJECT_SCHEMA: NULL
            OBJECT_NAME: NULL
  OBJECT_INSTANCE_BEGIN: NULL
            MYSQL_ERRNO: 0
      RETURNED_SQLSTATE: NULL
           MESSAGE_TEXT: NULL
```

```
                ERRORS: 0
              WARNINGS: 0
         ROWS_AFFECTED: 0
             ROWS_SENT: 125
         ROWS_EXAMINED: 125
CREATED_TMP_DISK_TABLES: 0
     CREATED_TMP_TABLES: 0
       SELECT_FULL_JOIN: 0
 SELECT_FULL_RANGE_JOIN: 0
           SELECT_RANGE: 0
     SELECT_RANGE_CHECK: 0
            SELECT_SCAN: 1
      SORT_MERGE_PASSES: 0
             SORT_RANGE: 0
              SORT_ROWS: 0
              SORT_SCAN: 0
          NO_INDEX_USED: 1
     NO_GOOD_INDEX_USED: 0
       NESTING_EVENT_ID: NULL
     NESTING_EVENT_TYPE: NULL
```

Profiling Statement Activity

- Statement Summaries Per User

```
mysql> SELECT *
    ->   FROM performance_schema.events_statements_summary_by_user_by_event_name
    ->  WHERE user = 'mem3'
    ->    AND sum_timer_wait > 0
    ->  ORDER BY sum_timer_wait DESC LIMIT 1\G
*************************** 1. row ***************************
                      USER: mem3
                EVENT_NAME: statement/sql/select
                COUNT_STAR: 46110097
            SUM_TIMER_WAIT: 56305193025555000
            MIN_TIMER_WAIT: 17858000
            AVG_TIMER_WAIT: 1221103000
            MAX_TIMER_WAIT: 47860948487000
             SUM_LOCK_TIME: 4326090581000000
                SUM_ERRORS: 26122
              SUM_WARNINGS: 3194
         SUM_ROWS_AFFECTED: 32
             SUM_ROWS_SENT: 59436762
         SUM_ROWS_EXAMINED: 1953654145
SUM_CREATED_TMP_DISK_TABLES: 14262
      SUM_CREATED_TMP_TABLES: 60178
        SUM_SELECT_FULL_JOIN: 5926
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 17880
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 204539
      SUM_SORT_MERGE_PASSES: 0
           SUM_SORT_RANGE: 0
            SUM_SORT_ROWS: 150801
            SUM_SORT_SCAN: 3073
        SUM_NO_INDEX_USED: 190598
     SUM_NO_GOOD_INDEX_USED: 0
```

Summarized by statement type, I.e: statement/sql/select statement/sql/update

Profiling Statement Activity

- Stage Summaries Per User

```
mysql> SELECT user, event_name AS stage,
    ->        count_star AS count,
    ->        format_time(sum_timer_wait) as total_latency,
    ->        format_time(avg_timer_wait) as avg_latency,
    ->        format_time(max_timer_wait) as max_latency
    ->   FROM performance_schema.events_stages_summary_by_user_by_event_name
    ->  WHERE user = 'mem3'
    ->    AND sum_timer_wait > 0
    ->  ORDER BY sum_timer_wait DESC;
+------+---------------------------------------+---------+---------------+-------------+-------------+
| user | stage                                 | count   | total_latency | avg_latency | max_latency |
+------+---------------------------------------+---------+---------------+-------------+-------------+
| mem3 | stage/sql/Sending data                |  742258 | 00:09:57.41   | 804.85 us   | 40.43 s     |
| mem3 | stage/sql/init                        | 2499216 | 00:02:23.17   | 57.29 us    | 192.33 ms   |
| mem3 | stage/sql/statistics                  |  742397 | 44.26 s       | 59.61 us    | 93.55 ms    |
| mem3 | stage/sql/freeing items               | 1503235 | 19.32 s       | 12.85 us    | 4.13 ms     |
| mem3 | stage/sql/updating                    |  205527 | 19.13 s       | 93.08 us    | 163.45 ms   |
| mem3 | stage/sql/Opening tables              | 1006357 | 13.69 s       | 13.60 us    | 14.62 ms    |
| mem3 | stage/sql/System lock                 |  997725 | 11.80 s       | 11.82 us    | 2.39 ms     |
| mem3 | stage/sql/preparing                   |  742361 | 6.22 s        | 8.38 us     | 2.08 ms     |
| mem3 | stage/sql/closing tables              | 1503135 | 5.81 s        | 3.87 us     | 4.10 ms     |
| mem3 | stage/sql/optimizing                  |  744163 | 4.29 s        | 5.77 us     | 4.97 ms     |
| mem3 | stage/sql/update                      |   97458 | 3.98 s        | 40.83 us    | 175.50 ms   |
| mem3 | stage/sql/checking permissions        | 1303885 | 3.16 s        | 2.43 us     | 2.53 ms     |
```

Profiling Statement Activity

- Tracing Statements

- By taking all available data in the "***_history_long**" tables, we can build a complete picture of what sessions and statements are doing
- We can **link** the histories of Statements, Stages and Waits using the "**event_id**" and "**nesting_event_id**" columns, which define hierarchy
- *This is only really effective on development/test systems, production systems with concurrency age history very very quickly, and turning the *_history_long tables on with all instrumentation on is probably not recommend*

Profiling Statement Activity

```
mysql> select event_id, event_name, timer_wait, nesting_event_id
    ->   from events_waits_history_long where thread_id = 20;
+----------+---------------------------------------+------------+------------------+
| event_id | event_name                            | timer_wait | nesting_event_id |
+----------+---------------------------------------+------------+------------------+
|     2547 | idle                                  | 9264793230 |             2516 |
|     2548 | wait/io/socket/sql/client_connection  |    4974480 |             2516 |
|     2551 | wait/synch/mutex/sql/THD::LOCK_thd_data |     472500 |             2550 |
|     2552 | wait/synch/mutex/sql/THD::LOCK_thd_data |      90720 |             2550 |
|     2554 | wait/synch/rwlock/sql/LOCK_grant      |     623700 |             2553 |
|     2556 | wait/synch/mutex/sql/MDL_map::mutex   |     181440 |             2555 |
|     2557 | wait/synch/rwlock/sql/MDL_lock::rwlock |     128520 |             2555 |
+----------+---------------------------------------+------------+------------------+
```

```
mysql> select event_id, event_name, timer_wait, nesting_event_id from events_stages_history_long where event_id = 2555;
+----------+--------------------------+------------+------------------+
| event_id | event_name               | timer_wait | nesting_event_id |
+----------+--------------------------+------------+------------------+
|     2555 | stage/sql/Opening tables |  203212000 |             2549 |
+----------+--------------------------+------------+------------------+
```

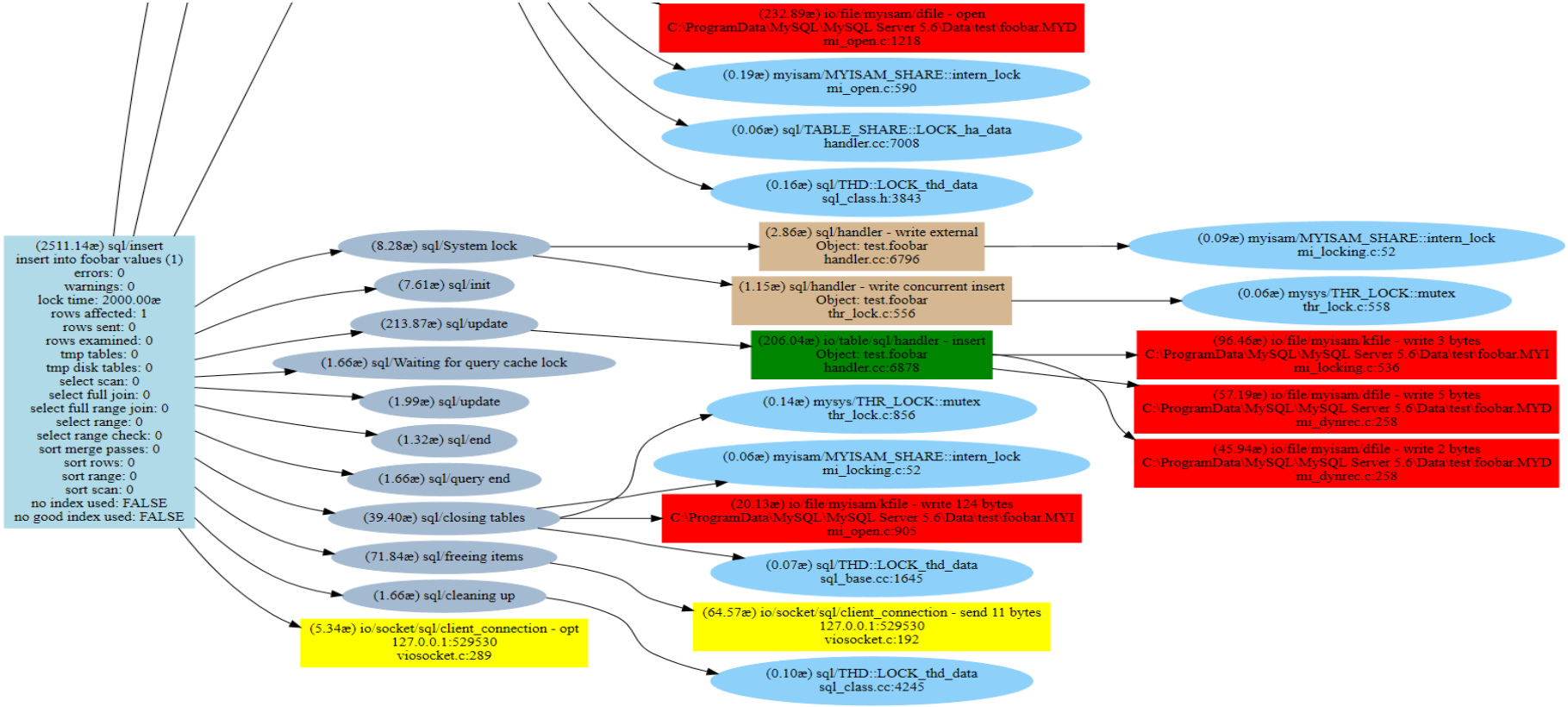
```
mysql> select event_id, event_name, timer_wait, substring(sql_text, 1, 40) as sql_text
    ->   from events_statements_history_long where event_id = 2549;
+----------+---------------------+------------+------------------------------------------+
| event_id | event_name          | timer_wait | sql_text                                 |
+----------+---------------------+------------+------------------------------------------+
|     2549 | statement/sql/select |  309041000 | select event_id, event_name, timer_wait, |
+----------+---------------------+------------+------------------------------------------+
```

This means we can graph their relationships!

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

 |

Profiling Statement Activity



 | |

Profiling Statement Activity

- Statement Digests

- A new summary view that presents aggregated statistics for ***normalized statements***
- Can be used to drill in to your problem statements instead of using the Slow Query Log, online (*and it has more info*)
- Each statement gets an MD5 “DIGEST”, that is also available to link in to events_statements_current / events_statements_history etc. for raw per query statistics, if wanted

Profiling Statement Activity

```
mysql> desc performance_schema.events_statements_summary_by_digest;
+-----------------------------+---------------------+------+-----+---------------------+
| Field                       | Type                | Null | Key | Default             |
+-----------------------------+---------------------+------+-----+---------------------+
| DIGEST                      | varchar(32)         | YES  |     | NULL                |
| DIGEST_TEXT                 | longtext            | YES  |     | NULL                |
| COUNT_STAR                  | bigint(20) unsigned | NO   |     | NULL                |
| SUM_TIMER_WAIT              | bigint(20) unsigned | NO   |     | NULL                |
| MIN_TIMER_WAIT              | bigint(20) unsigned | NO   |     | NULL                |
| AVG_TIMER_WAIT              | bigint(20) unsigned | NO   |     | NULL                |
| MAX_TIMER_WAIT              | bigint(20) unsigned | NO   |     | NULL                |
| SUM_LOCK_TIME               | bigint(20) unsigned | NO   |     | NULL                |
| SUM_ERRORS                  | bigint(20) unsigned | NO   |     | NULL                |
| SUM_WARNINGS                | bigint(20) unsigned | NO   |     | NULL                |
| SUM_ROWS_AFFECTED           | bigint(20) unsigned | NO   |     | NULL                |
| SUM_ROWS_SENT               | bigint(20) unsigned | NO   |     | NULL                |
| SUM_ROWS_EXAMINED           | bigint(20) unsigned | NO   |     | NULL                |
| SUM_CREATED_TMP_DISK_TABLES | bigint(20) unsigned | NO   |     | NULL                |
| SUM_CREATED_TMP_TABLES      | bigint(20) unsigned | NO   |     | NULL                |
| SUM_SELECT_FULL_JOIN        | bigint(20) unsigned | NO   |     | NULL                |
| SUM_SELECT_FULL_RANGE_JOIN  | bigint(20) unsigned | NO   |     | NULL                |
| SUM_SELECT_RANGE            | bigint(20) unsigned | NO   |     | NULL                |
| SUM_SELECT_RANGE_CHECK      | bigint(20) unsigned | NO   |     | NULL                |
| SUM_SELECT_SCAN             | bigint(20) unsigned | NO   |     | NULL                |
| SUM_SORT_MERGE_PASSES       | bigint(20) unsigned | NO   |     | NULL                |
| SUM_SORT_RANGE              | bigint(20) unsigned | NO   |     | NULL                |
| SUM_SORT_ROWS               | bigint(20) unsigned | NO   |     | NULL                |
| SUM_SORT_SCAN               | bigint(20) unsigned | NO   |     | NULL                |
| SUM_NO_INDEX_USED           | bigint(20) unsigned | NO   |     | NULL                |
| SUM_NO_GOOD_INDEX_USED      | bigint(20) unsigned | NO   |     | NULL                |
| FIRST_SEEN                  | timestamp           | NO   |     | 0000-00-00 00:00:00 |
| LAST_SEEN                   | timestamp           | NO   |     | 0000-00-00 00:00:00 |
+-----------------------------+---------------------+------+-----+---------------------+
```

Profiling Statement Activity

- Statement Normalization
- Normalization folds certain constructs of statements
  - Strip whitespace / comments
  - Replace literals with ?, such as “WHERE foo = 1” becomes “WHERE foo = ?”
  - Folds lists of things, such as “IN (1,2,3)” becomes “IN (…)”
  - Folds multi-row inserts, such as “VALUES (1), (2)” becomes “VALUES (?) /*, … */”

Profiling Statement Activity

```
mysql> SELECT *
    ->   FROM performance_schema.events_statements_summary_by_digest
    ->  ORDER BY sum_timer_wait DESC LIMIT 1\G
*************************** 1. row ***************************
                    DIGEST: 19b2442e880ef7d9a21db222bae81bdd
               DIGEST_TEXT: SELECT page_type , IF ( TABLE_NAME IS NULL OR INSTR ( TA
, - ? ) ) AS TABLE_NAME , IF ( index_name IS NULL , page_type , index_name ) AS inde
iables WHERE variable_name = ? ) , ? ) , compressed_size ) ) AS total_bytes , SUM ( n
ER_PAGE GROUP BY page_type , TABLE_NAME , index_name ORDER BY NULL
                COUNT_STAR: 1312
            SUM_TIMER_WAIT: 52002971063293000
            MIN_TIMER_WAIT: 31717292272000
            AVG_TIMER_WAIT: 39636410871000
            MAX_TIMER_WAIT: 47860948487000
             SUM_LOCK_TIME: 372147000000
                SUM_ERRORS: 1
              SUM_WARNINGS: 2622
         SUM_ROWS_AFFECTED: 0
             SUM_ROWS_SENT: 609197
         SUM_ROWS_EXAMINED: 2064202829
SUM_CREATED_TMP_DISK_TABLES: 2624
      SUM_CREATED_TMP_TABLES: 3936
        SUM_SELECT_FULL_JOIN: 0
 SUM_SELECT_FULL_RANGE_JOIN: 0
           SUM_SELECT_RANGE: 0
     SUM_SELECT_RANGE_CHECK: 0
            SUM_SELECT_SCAN: 2624
      SUM_SORT_MERGE_PASSES: 0
           SUM_SORT_RANGE: 0
            SUM_SORT_ROWS: 0
            SUM_SORT_SCAN: 0
        SUM_NO_INDEX_USED: 1312
   SUM_NO_GOOD_INDEX_USED: 0
               FIRST_SEEN: 2012-09-25 11:50:59
                LAST_SEEN: 2012-09-26 14:00:36
```

Has errors/warnings

More rows scanned Than returned

Lots of temp tables

Doing full table scans

Profiling Statement Activity

- Statements with Temporary Tables

```
mysql> SELECT format_statement(DIGEST_TEXT) AS query,
    ->        COUNT_STAR AS exec_count,
    ->        SUM_CREATED_TMP_TABLES AS in_memory,
    ->        SUM_CREATED_TMP_DISK_TABLES AS on_disk,
    ->        ROUND(SUM_CREATED_TMP_TABLES / COUNT_STAR) AS avg_per_query,
    ->        ROUND((SUM_CREATED_TMP_DISK_TABLES / SUM_CREATED_TMP_TABLES) * 100) AS to_disk_pct
    ->   FROM performance_schema.events_statements_summary_by_digest
    ->  WHERE SUM_CREATED_TMP_TABLES > 0
    -> ORDER BY SUM_CREATED_TMP_DISK_TABLES DESC, SUM_CREATED_TMP_TABLES DESC LIMIT 5;
+-----------------------------------------------------------------+------------+-----------+---------+---------------+-------------+
| query                                                           | exec_count | in_memory | on_disk | avg_per_query | to_disk_pct |
+-----------------------------------------------------------------+------------+-----------+---------+---------------+-------------+
| SELECT * FROM ( SELECT digest  ... irstSeen` , unix_timestamp ... |       1299 |      9093 |    3897 |             7 |          43 |
| SELECT plugin_name FROM inform ... tus = ? ORDER BY plugin_name |       2597 |      2597 |    2597 |             1 |         100 |
| SELECT page_type , IF ( TABLE_ ... E , index_name ORDER BY NULL |       1298 |      3894 |    2596 |             3 |          67 |
| SELECT COUNT ( * ) AS `num_loc ... ocesslist` WHERE `state` = ? |       1567 |      1567 |    1567 |             1 |         100 |
| SELECT COUNT ( * ) AS `num_lon ... g_query_time AND `state` = ? |       1567 |      1567 |    1567 |             1 |         100 |
+-----------------------------------------------------------------+------------+-----------+---------+---------------+-------------+
```

Profiling Statement Activity

- Statements with Full Table Scans

```
mysql> SELECT format_statement(DIGEST_TEXT) AS query,
    ->        COUNT_STAR AS exec_count,
    ->        SUM_NO_INDEX_USED AS no_index_used_count,
    ->        ROUND((SUM_NO_INDEX_USED / COUNT_STAR) * 100) no_index_used_pct,
    ->        SUM_ROWS_EXAMINED AS rows_scanned
    ->   FROM performance_schema.events_statements_summary_by_digest
    ->  WHERE DIGEST_TEXT LIKE 'SELECT%'
    ->    AND (SUM_NO_INDEX_USED > 0
    ->     OR SUM_NO_GOOD_INDEX_USED > 0)
    -> ORDER BY no_index_used_pct DESC, exec_count DESC LIMIT 5;
+-----------------------------------------------------------------+------------+--------------------+------------------+--------------+
| query                                                           | exec_count | no_index_used_count | no_index_used_pct | rows_scanned |
+-----------------------------------------------------------------+------------+--------------------+------------------+--------------+
| SELECT `os0_` . `hid` AS `hid4 ... , `os0_` . `hasVersion` AS ... |      93459 |              93459 |              100 |      2947520 |
| SELECT `os0_` . `hid` AS `hid4 ... , `os0_` . `hasVersion` AS ... |      39532 |              39532 |              100 |      1369092 |
| SELECT CAST ( SUM_NUMBER_OF_BY ... WHERE EVENT_NAME = ? LIMIT ?   |       6550 |               6550 |              100 |       201740 |
| SELECT `os0_` . `hid` AS `hid4 ... , `os0_` . `hasVersion` AS ... |       5956 |               5956 |              100 |       113031 |
| SELECT `os0_` . `hid` AS `hid3 ... sVersion23_39_` , `os0_` . ... |       5104 |               5104 |              100 |       122016 |
+-----------------------------------------------------------------+------------+--------------------+------------------+--------------+
```

Wrapping Up

- This is really just scratching the surface, there are 533 instrument types within 5.6, and many more columns that can be used to filter for different views
- There are many other ways to look at the data that I haven't presented here
- I hope this gives you a helping start in how to start looking at the data though!
- All of these views, and more, are available within ps_helper on my site!