PERFORMANCE_SCHEMA
and sys schema

What can we do with it?

FOSDEM 2016, January 30th, Brussels

Oli Sennhauser
Senior MySQL Consultant at FromDual GmbH
oli.sennhauser@fromdual.com
About FromDual GmbH

Support
MySQL
GALERA CLUSTER
remote-DBA
SOUg
MariaDB
Consulting
Open Source Business Alliance
Training
DOAG
Contents

PERFORMANCE_SCHEMA and sys schema

- Database Profiling
- PERFORMANCE_SCHEMA (P_S)
  - Installation, configuration, instrumentation, etc.
- sys schema
  - Installation, configuration, etc.
- Use cases
Database Profiling

• Where has my time gone?
• Where have my resources gone?

```javascript
function x() {
    start = current_time();
    count[x]++;
    ...
    end = current_time();
    duration[x] += (end - start);
}
```

<table>
<thead>
<tr>
<th>function</th>
<th>count</th>
<th>time</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>123</td>
<td>156.25</td>
<td>0.8%</td>
</tr>
<tr>
<td>y</td>
<td>19</td>
<td>827.30</td>
<td>4.1%</td>
</tr>
<tr>
<td>z</td>
<td>2</td>
<td>19280.00</td>
<td>95.1%</td>
</tr>
<tr>
<td>Total</td>
<td>144</td>
<td>20263.55</td>
<td>100.0%</td>
</tr>
</tbody>
</table>
MySQL Profiler

- Since MySQL 5.0
- `SET profiling = 1; SELECT ...; SHOW PROFILE;`
- Deprecated in 5.6
- So what now?

<table>
<thead>
<tr>
<th>Status</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>starting</td>
<td>0.000059</td>
</tr>
<tr>
<td>checking permissions</td>
<td>0.000008</td>
</tr>
<tr>
<td>Opening tables</td>
<td>0.000017</td>
</tr>
<tr>
<td>init</td>
<td>0.000023</td>
</tr>
<tr>
<td>System lock</td>
<td>0.000009</td>
</tr>
<tr>
<td>optimizing</td>
<td>0.000006</td>
</tr>
<tr>
<td>statistics</td>
<td>0.000014</td>
</tr>
<tr>
<td>preparing</td>
<td>0.000011</td>
</tr>
<tr>
<td>Creating tmp table</td>
<td>0.000023</td>
</tr>
<tr>
<td>Sorting result</td>
<td>0.000005</td>
</tr>
<tr>
<td>executing</td>
<td>0.000003</td>
</tr>
<tr>
<td>Sending data</td>
<td>1.253803</td>
</tr>
<tr>
<td>Creating sort index</td>
<td>0.000049</td>
</tr>
<tr>
<td>end</td>
<td>0.000005</td>
</tr>
<tr>
<td>removing tmp table</td>
<td>0.000019</td>
</tr>
<tr>
<td>end</td>
<td>0.000006</td>
</tr>
<tr>
<td>query end</td>
<td>0.000009</td>
</tr>
<tr>
<td>closing tables</td>
<td>0.000012</td>
</tr>
<tr>
<td>freeing items</td>
<td>0.000019</td>
</tr>
<tr>
<td>logging slow query</td>
<td>0.000004</td>
</tr>
<tr>
<td>cleaning up</td>
<td>0.000011</td>
</tr>
</tbody>
</table>
PERFORMANCE_SCHEMA (P_S)

- Discussed at least since 2006
- Introduced in MySQL 5.5 (2010)
  - Disabled by default
  - Not really useful yet
- Improved in MySQL 5.6 (2011/12)
  - Enabled by default
  - Became useful and interesting
- More improvements in MySQL 5.7 (2013-15)
  - More probes, more information:
    - Memory, Transactions, Replication, Session/Status variables
- Overhead 2% - 200%
  - Depends on how you do it → careful!
Goal of P_S

- Monitoring MySQL server execution at a low level
- Inspect internal execution of the server at runtime

- How: Server is instrumented to collect timing information
- Output: SQL queries on normal MySQL ring-buffer tables

- Events is anything the server does:
  - Function call
  - Wait for the O/S
  - SQL statement, stages
  - Memory
P_S installation & configuration

- Installed by default
  - `mysql_upgrade` → do restart afterwards
- Configuration in `my.cnf`
  - 5.5 off
  - `performance_schema = 1`
  - 5.6 and later on
- Tuning (`my.cnf`)
  - `performance_schema_events_waits_long = 10000`
- `SHOW GLOBAL VARIABLES LIKE 'PERF%';`
- `SHOW GLOBAL STATUS LIKE 'PERF%';`
Online enabling / disabling

- **setup_* tables**
  
  ```sql
  UPDATE setup_instruments
  SET ENABLED = 'YES', TIMED = 'YES'
  WHERE name = 'wait/synch/mutex/innodb/autoinc_mutex';
  ```

- **Actors**
  - Which user and host is profiled (default %)

- **Consumers**
  - Which P_S tables are filled with data (e.g. events_state_current, some are off!)

- **Instruments**
  - Which probes are activated (expensive ones are disabled)

- **Objects**
  - Database objects (table, trigger, procedure, function, event) to profile (on for user obj.)

- **Timers**
  - Default is in nanoseconds ($10^{-9}$ s)
Instrumentation depth

- Enable probes:
  - events_transactions_* → unknown
    - events_statements_* → moderate expensive
      - events_stages_* → expensive!
        - events_waits_* → very expensive!
      - end
    - events_memory_* → very expensive?
      - end
  - end
  - end
- End
- Can have a performance impact!
Consumer types and hierarchies

- `events_*_current` → one per thread
- `events_*_history` → most recent n events per thread (default 10)
- `events_*_history_long` → most recent n events (10000)

- `events_*_summary_global_by_event_name`
  - `events_*_summary_by_account_by_event_name` → (host + user)
    - `events_*_summary_by_host_by_event_name` → e.g. 127.0.0.1
    - `events_*_summary_by_user_by_event_name` → e.g. root
      - `events_*_summary_by_thread_by_event_name` → thread (connection)

- `events_*_summary_by_digest` → normalized query: WHERE a = ?
- `events_*_summary_by_program` → procedure, function, trigger, event
- `events_*_summary_by_instance` → file, mutex, prep stmt, rwlock, socket, condition

- Can be truncate with TRUCATE TABLE ...
sys schema

- Originally “ps_helper” in 5.6 (2012)
- By Mark Leith
  - Senior SW Dev Mgr at Oracle
- Make it easier to use the P_S!
- A database schema (sys)
- Consisting of tables, views, functions, procedures and triggers.
- To give human readable insight into a MySQL database.
- Based on PERFORMANCE_SCHEMA and INFORMATION_SCHEMA
- Current release v1.5 (2016-01)
- Download: https://github.com/mysql/mysql-sys
sys schema installation

- Since MySQL 5.7.7 installed by default!
  - Can be skipped
- In MySQL 5.6:
  
  ```bash
  wget https://github.com/mysql/mysql-sys/archive/master.tar.gz
  tar xf master.tar.gz
  cd mysql-sys-master
  Mysql -user=root --password < ./sys_56.sql
  ```

- `mysql_upgrade` installs/upgrades sys
  - Can be skipped
sys view/table types

- 2 types, e.g.:
  - `io_by_thread_by_latency` → human understandable numbers
  - `x$io_by_thread_by_latency` → base tables in picoseconds \((10^{-12}s)\)

- Topics:
  - `host_*` → Activities grouped by host
  - `innodb_*` → InnoDB Information
  - `io_*` → I/O consumers grouped by file, bytes, latency
  - `memory_*` → Memory usage grouped by host, thread, user, type
  - `schema_*` → Various information about schema
  - `statement_*` → Statistics about statements
  - `user_*` → Information per user
  - `waits_*` → Wait event informations
Configure P_S with sys

• Reset defaults:
  • CALL sys.ps_setup_reset_to_default(TRUE);

• Change setup tables:
  • CALL sys.ps_setup_enable_instrument('wait');
  • CALL sys.ps_setup_enable_instrument('stage');
  • CALL sys.ps_setup_enable_instrument('statement');
  • CALL sys.ps_setup_enable_consumer('current');
  • CALL sys.ps_setup_enable_consumer('history_long');
Use cases

- The following use cases can be found at:
SSL encryption used or not

```sql
SELECT variable_value AS tls_version, processlist_user AS user,
       processlist_host AS host
FROM performance_schema.status_by_thread AS sbt
JOIN performance_schema.threads AS t
ON t.thread_id = sbt.thread_id
WHERE variable_name = 'Ssl_version'
ORDER BY tls_version;
```

<table>
<thead>
<tr>
<th>tls_version</th>
<th>user</th>
<th>host</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLSv1.1</td>
<td>root</td>
<td>localhost</td>
</tr>
</tbody>
</table>

www.fromdual.com
Accounts not properly closing connections

```sql
SELECT ess.user, ess.host
 , (a.total_connections - a.current_connections) - ess.count_star as not_closed
 , ((a.total_connections - a.current_connections) - ess.count_star) * 100 /
   (a.total_connections - a.current_connections) as pct_not_closed
FROM performance_schema.events_statements_summary_by_account_by_event_name ess
JOIN performance_schema.accounts a on (ess.user = a.user and ess.host = a.host)
WHERE ess.event_name = 'statement/com/quit'
  AND (a.total_connections - a.current_connections) > ess.count_star
;

<table>
<thead>
<tr>
<th>user</th>
<th>host</th>
<th>not_closed</th>
<th>pct_not_closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>localhost</td>
<td>1</td>
<td>5.0000</td>
</tr>
</tbody>
</table>
```
SELECT DISTINCT m_u.user, m_u.host
    FROM mysql.user m_u
    LEFT JOIN performance_schema.accounts ps_a
        ON m_u.user = ps_a.user AND m_u.host = ps_a.host
    WHERE ps_a.user IS NULL
ORDER BY m_u.user
;

+----------+-----------+
| user     | host      |
|-----------+-----------|
| focmm     | master    |
| mysql.sys | localhost |
| oli       | localhost |
| root      | %         |
+----------+-----------+
Bad SQL queries by user

```sql
SELECT user, host, event_name,
    , sum_created_tmp_disk_tables AS tmp_disk_tables,
    , sum_select_full_join AS full_join
FROM performance_schema.events_statements_summary_by_account_by_event_name
WHERE sum_created_tmp_disk_tables > 0
    OR sum_select_full_join > 0
ORDER BY sum_sort_merge_passes DESC
LIMIT 10
;
```

<table>
<thead>
<tr>
<th>user</th>
<th>host</th>
<th>event_name</th>
<th>tmp_disk_tables</th>
<th>full_join</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>localhost</td>
<td>statement/sql/select</td>
<td>134</td>
<td>204</td>
</tr>
<tr>
<td>root</td>
<td>localhost</td>
<td>statement/sql/show_fields</td>
<td>3607</td>
<td>0</td>
</tr>
<tr>
<td>root</td>
<td>localhost</td>
<td>statement/sql/show_triggers</td>
<td>1792</td>
<td>0</td>
</tr>
<tr>
<td>root</td>
<td>localhost</td>
<td>statement/com/Field List</td>
<td>28</td>
<td>0</td>
</tr>
</tbody>
</table>

www.fromdual.com
Top long running queries

---

**UPDATE** `performance_schema.setup_consumers` **SET** enabled = 1
WHERE name = 'events_statements_history_long';

**SELECT** `left(digest_text, 64)`
, ROUND(SUM(`timer_end`-`timer_start`)/1000000000, 1) **AS** `tot_exec_ms`
, ROUND(SUM(`timer_wait`)/1000000000, 1) **AS** `tot_wait_ms`
, ROUND(SUM(`lock_time`)/1000000000, 1) **AS** `tot_lock_ms`
, MIN(LEFT(DATE_SUB(NOW(), INTERVAL `isgs.VARIABLE_VALUE` - `TIMER_START`*10e-13) second), 19)) **AS** `first_seen`
, MAX(LEFT(DATE_SUB(NOW(), INTERVAL `isgs.VARIABLE_VALUE` - `TIMER_START`*10e-13) second), 19)) **AS** `last_seen`
, COUNT(*) **as** `cnt`
FROM `performance_schema.events_statements_history_long`
JOIN `performance_schema.global_status` **AS** `isgs`
WHERE `isgs.variable_name` = 'UPTIME'
GROUP BY `left(digest_text,64)`
ORDER BY `tot_exec_ms` DESC
;

<table>
<thead>
<tr>
<th><code>left(digest_text, 64)</code></th>
<th><code>tot_exec_ms</code></th>
<th><code>tot_wait_ms</code></th>
<th><code>tot_lock_ms</code></th>
<th><code>cnt</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>INSERT INTO <code>history</code> (<code>itemid</code>, <code>clock</code>, <code>ns</code>, <code>VALUE</code>) VALUES</td>
<td>12.3</td>
<td>12.3</td>
<td>2.4</td>
<td>6</td>
</tr>
<tr>
<td>SELECT <code>i</code>. <code>itemid</code>, <code>i</code>. <code>state</code>, <code>i</code>. <code>delta</code>, <code>i</code>. <code>m</code></td>
<td>10.4</td>
<td>10.4</td>
<td>1.9</td>
<td>6</td>
</tr>
<tr>
<td>SELECT <code>i</code>. <code>itemid</code>, <code>i</code>. <code>key</code>, <code>h</code>. <code>host</code>, <code>i</code>. <code>typ</code></td>
<td>7.5</td>
<td>7.5</td>
<td>0.0</td>
<td>1</td>
</tr>
<tr>
<td>SELECT <code>h</code>. <code>hostid</code>, <code>h</code>. <code>host</code>, <code>h</code>. <code>name</code>, <code>t</code>. <code>htt</code></td>
<td>3.2</td>
<td>3.2</td>
<td>1.1</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td>2.0</td>
<td>0.9</td>
<td>4</td>
</tr>
</tbody>
</table>
Unused indexes

```sql
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
    AND index_name != 'PRIMARY'
ORDER BY object_schema, object_name
;
```

```
+-----------------+-----------------+-----------------+
| object_schema   | object_name     | index_name      |
+-----------------+-----------------+-----------------+
| a               | audit           | data_2          |
| a               | audit           | data            |
| a               | c1              | c2_id           |
+-----------------+-----------------+-----------------+

SELECT *
FROM sys.schema_unused_indexes
;
```
# Redundant indexes

```sql
SELECT table_schema, table_name,
    redundant_index_columns, dominant_index_columns
FROM sys/schema_redundant_indexes;
```

<table>
<thead>
<tr>
<th>table_schema</th>
<th>table_name</th>
<th>redundant_index_columns</th>
<th>dominant_index_columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>crm</td>
<td>accounts</td>
<td>id,deleted</td>
<td>id</td>
</tr>
<tr>
<td>crm</td>
<td>accounts_bugs</td>
<td>account_id</td>
<td>account_id,bug_id</td>
</tr>
<tr>
<td>crm</td>
<td>acl_actions</td>
<td>id,deleted</td>
<td>id</td>
</tr>
<tr>
<td>crm</td>
<td>acl_roles</td>
<td>id,deleted</td>
<td>id</td>
</tr>
<tr>
<td>crm</td>
<td>acl_roles_actions</td>
<td>role_id</td>
<td>role_id,action_id</td>
</tr>
</tbody>
</table>
Statements with errors or warnings

```sql
SELECT db, query, exec_count, errors, warnings
FROM sys.statements_with_errors_or_warnings;
```

+-----------------+-----------------------------+--------------+---------+---------+
| db              | query                      | exec_count   | errors  | warnings |
| mysql           | UPDATE `setup_consumers... | 1            | 1       | 0       |
| mysql           | SELECT LEFT (`digest_t... | 2            | 1       | 0       |
| sys             | SELECT * FROM `setup_co... | 1            | 1       | 0       |
| performance_schema | SELECT * FROM `user_sum... | 1            | 1       | 0       |
| test            | SELECT * FROM `test` LI... | 6            | 1       | 0       |
```
### Tables with full table scan

```sql
SELECT *
FROM sys.schema_tables_with_full_table_scans
;
```

<table>
<thead>
<tr>
<th>object_schema</th>
<th>object_name</th>
<th>rows_full_scanned</th>
<th>latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>zabbix</td>
<td>triggers</td>
<td>1436</td>
<td>5.05 ms</td>
</tr>
<tr>
<td>zabbix</td>
<td>hosts</td>
<td>34</td>
<td>41.81 us</td>
</tr>
<tr>
<td>zabbix</td>
<td>interface</td>
<td>6</td>
<td>4.33 ms</td>
</tr>
<tr>
<td>zabbix</td>
<td>expressions</td>
<td>4</td>
<td>648.98 us</td>
</tr>
<tr>
<td>zabbix</td>
<td>config</td>
<td>1</td>
<td>3.62 ms</td>
</tr>
<tr>
<td>zabbix</td>
<td>globalmacro</td>
<td>1</td>
<td>2.61 ms</td>
</tr>
<tr>
<td>zabbix</td>
<td>media</td>
<td>1</td>
<td>548.01 us</td>
</tr>
</tbody>
</table>
## I/O by user

```sql
SELECT *
    FROM sys.user_summary_by_file_io
;
```

<table>
<thead>
<tr>
<th>user</th>
<th>ios</th>
<th>io_latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>6926</td>
<td>8.52 s</td>
</tr>
<tr>
<td>zabbix</td>
<td>738</td>
<td>127.52 ms</td>
</tr>
<tr>
<td>root</td>
<td>101</td>
<td>6.57 ms</td>
</tr>
<tr>
<td>fpmmm</td>
<td>363</td>
<td>644.00 us</td>
</tr>
</tbody>
</table>
```
Latency on file I/O

```sql
SELECT event_name, total, total_latency, read_latency, write_latency, misc_latency
FROM sys.io_global_by_wait_by_latency
LIMIT 5
;
```

<table>
<thead>
<tr>
<th>event_name</th>
<th>total</th>
<th>tot_latency</th>
<th>r_latency</th>
<th>w_latency</th>
<th>misc_latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>innodb/innodb_log_file</td>
<td>1323</td>
<td>8.84 s</td>
<td>29.08 us</td>
<td>25.08 ms</td>
<td>8.81 s</td>
</tr>
<tr>
<td>innodb/innodb_data_file</td>
<td>6299</td>
<td>8.81 s</td>
<td>429.56 ms</td>
<td>660.95 ms</td>
<td>7.72 s</td>
</tr>
<tr>
<td>sql/binlog_index</td>
<td>40</td>
<td>35.20 ms</td>
<td>4.07 us</td>
<td>0 ps</td>
<td>35.19 ms</td>
</tr>
<tr>
<td>sql/binlog</td>
<td>610</td>
<td>26.09 ms</td>
<td>14.46 us</td>
<td>4.12 ms</td>
<td>21.96 ms</td>
</tr>
<tr>
<td>sql/relaylog</td>
<td>13</td>
<td>18.77 ms</td>
<td>1.97 us</td>
<td>5.56 us</td>
<td>18.77 ms</td>
</tr>
</tbody>
</table>
### Memory by user

```sql
UPDATE performance_schema.setup_instruments
    SET enabled = 'YES'
    WHERE name LIKE 'memory%'
;

SELECT user, current_allocated, current_avg_alloc, current_max_alloc, total_allocated
    FROM sys.memory_by_user_by_current_bytes
;
```

<table>
<thead>
<tr>
<th>user</th>
<th>curr_alloc</th>
<th>curr_avg_alloc</th>
<th>curr_max_alloc</th>
<th>tot_alloc</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>348.35 KiB</td>
<td>9.95 KiB</td>
<td>248.04 KiB</td>
<td>8.54 MiB</td>
</tr>
<tr>
<td>zabbix</td>
<td>44.72 KiB</td>
<td>2.48 KiB</td>
<td>32.02 KiB</td>
<td>11.87 MiB</td>
</tr>
<tr>
<td>fpmmm</td>
<td>43.56 KiB</td>
<td>4.36 KiB</td>
<td>43.56 KiB</td>
<td>1.37 MiB</td>
</tr>
<tr>
<td>background</td>
<td>9.85 KiB</td>
<td>1.97 KiB</td>
<td>9.64 KiB</td>
<td>1.42 MiB</td>
</tr>
</tbody>
</table>
SELECT, INSERT, UPDATE and DELETE per table

```
SELECT object_type, object_schema, object_name
  , count_star, count_read, count_write
  , count_insert
FROM performance_schema.table_io_waits_summary_by_table
WHERE count_star > 0
ORDER BY count_read DESC
LIMIT 5
;
```

<table>
<thead>
<tr>
<th>object_type</th>
<th>object_schema</th>
<th>object_name</th>
<th>count_star</th>
<th>count_read</th>
<th>count_write</th>
<th>count_insert</th>
</tr>
</thead>
<tbody>
<tr>
<td>TABLE</td>
<td>zabbix</td>
<td>history_uint</td>
<td>12791</td>
<td>92</td>
<td>12699</td>
<td>12699</td>
</tr>
<tr>
<td>TABLE</td>
<td>zabbix</td>
<td>history</td>
<td>3374</td>
<td>10</td>
<td>3364</td>
<td>3364</td>
</tr>
<tr>
<td>TABLE</td>
<td>zabbix</td>
<td>history_str</td>
<td>2003</td>
<td>48</td>
<td>1955</td>
<td>1955</td>
</tr>
<tr>
<td>TABLE</td>
<td>zabbix</td>
<td>trends_uint</td>
<td>930</td>
<td>0</td>
<td>930</td>
<td>930</td>
</tr>
<tr>
<td>TABLE</td>
<td>zabbix</td>
<td>trends</td>
<td>200</td>
<td>0</td>
<td>200</td>
<td>200</td>
</tr>
</tbody>
</table>
```
How to proceed?

1. Define the question to answer:
   “Which transaction was locked by which other transaction?”

2. Can sys schema answer the question?

   ```sql
   SELECT * FROM sys.innodb_lock_waits;
   +----------------------------------+
   | wait_started | wait_age | locked_table  | locked_index | locked_type   |
   +----------------------------------+
   | 16-01-29 17:53:09 | 00:00:02 | `test`.`test` | PRIMARY      | RECORD        |
   +----------------------------------+
   ```

3. If not: Can P_S or I_S answer the question?

   ```sql
   SELECT * FROM information_schema.innodb_locks;
   +----------------------------------+
   | lock_trx_id | lock_type | lock_table  | lock_index | lock_rec   |
   +----------------------------------+
   | 27514604    | RECORD    | `test`.`test` | PRIMARY    | 2          |
   +----------------------------------+
   ```

Only recent, no history yet! :(
SHOW PROFILE VS P_S

SET PROFILING = 1;
SELECT ...;
SHOW PROFILES;

+-------+----------+-----------------------------+
| Query_ID | Duration  | Query                       |
|---------+----------+-----------------------------+
| 1       | 2.26217725 | select * from test          |

SHOW PROFILE FOR QUERY 1;

+-----------+----------+
| Status     | Duration  |
|-----------+----------+
| starting  | 0.000060 |
| checking permissions | 0.000009 |
| Opening tables     | 0.00020  |
| init      | 0.000020 |
| System lock | 0.000012 |
| optimizing | 0.000006 |
| statistics | 0.000014 |
| preparing | 0.000012 |
| executing | 0.000003 |
| Sending data | 2.261963  |
| end       | 0.000010 |
| query end | 0.000008 |
| closing tables | 0.000013 |
| freeing items | 0.000013 |
| cleaning up | 0.000015 |

UPDATE performance_schema.setup_instruments SET ENABLED = 'YES', TIMED = 'YES' WHERE NAME LIKE '%statement/%';
UPDATE performance_schema.setup_instruments SET ENABLED = 'YES', TIMED = 'YES' WHERE NAME LIKE '%stage/%';
UPDATE performance_schema.setup_consumers SET ENABLED = 'YES' WHERE NAME LIKE '%events_statements_%';
UPDATE performance_schema.setup_consumers SET ENABLED = 'YES' WHERE NAME LIKE '%events_stages_%';
SELECT ...;
SELECT eshl.event_id AS Query_ID, TRUNCATE(eshl.timer_wait/1000000000000, 6) as Duration, LEFT(eshl.sql_text, 120) AS Query
FROM performance_schema.events_statements_history_long AS eshl
JOIN performance_schema.events_statements_history_long
WHERE nesting_event_id = 12;

+-------+-------+---------------------------------------------+
| Query_ID | Duration  | Query                                      |
|---------+----------+---------------------------------------------+
| 12      | 13.560737 | select * from test.test                   |

SELECT SUBSTR(event_name, 11) AS Stage,
TRUNCATE(timer_wait/1000000000000, 6) AS Duration
FROM performance_schema.events_stages_history_long
WHERE nesting_event_id = 12;

+-------+----------+
| Stage  | Duration  |
|-------+----------+
| starting | 0.000043  |
| checking permissions | 0.000004 |
| Opening tables | 0.002700 |
| init      | 0.000025  |
| System lock | 0.000009 |
| optimizing | 0.000002 |
| statistics | 0.000014 |
| preparing | 0.000013  |
| executing | 0.000000  |
| Sending data | 13.557683 |
| end      | 0.000002  |
| query end | 0.000008  |
| closing tables | 0.000006 |
| freeing items | 0.000215 |
| cleaning up | 0.000001 |
Q & A

Questions ?
Discussion?

We have time for some face-to-face talks...

- FromDual provides neutral and independent:
  - Consulting
  - Remote-DBA
  - Support for MySQL, Galera, Percona Server and MariaDB
  - Training

www.fromdual.com/presentations