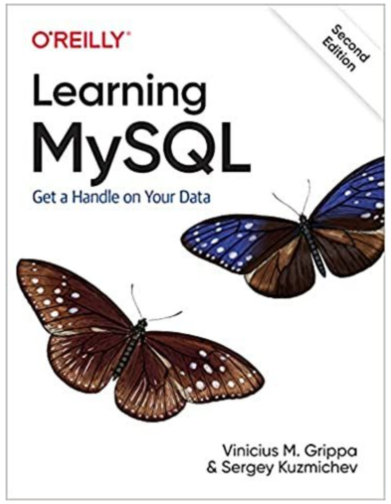
What we can know with Performance Schema in MySQL 8.0

Vinicius Grippa

Database Engineer at Percona

About me

- Percona Database Engineer for 5 years
- Working with databases for 18 years
- Co-author of the book Learning MySQL


O'REILLY
Second Edition
Learning MySQL
Get a Handle on Your Data
Vinicius M. Grippa & Sergey Kuzmichev

Agenda

- Starting from the beginning...
  - What is P_S?
  - Characteristics
  - What are events and instruments?
  - How does it work?
- What can we do with P_S?
- Questions

Starting from the beginning…

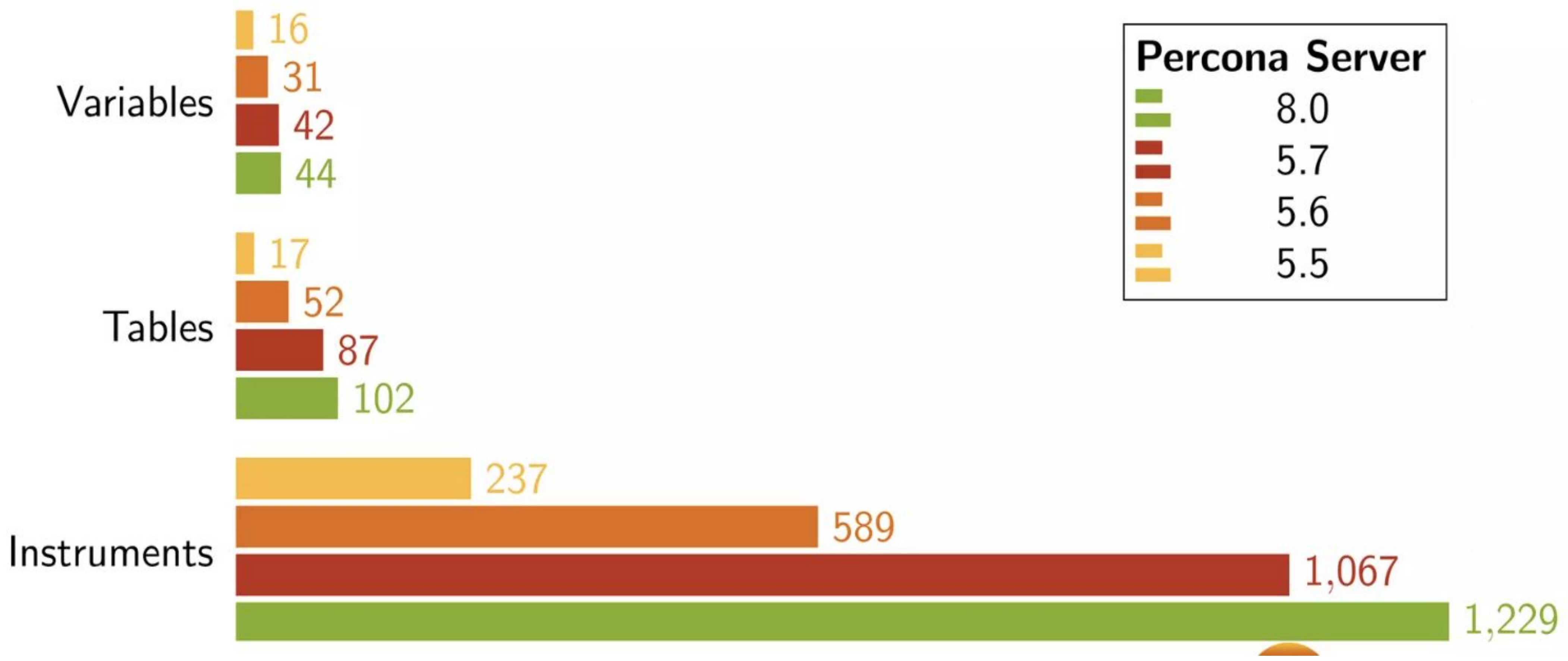
What is Performance Schema?

- a.k.a. P_S
- Introduced in MySQL 5.5
- It is a feature that monitors MySQL at a low level, checking the events happening in the database. It is intended to provide information from MySQL at runtime.
- Focus on performance data (different from `Information_Schema` that provides information about metadata)

P_S Characteristics

- It is in-memory tables that use no persistent on-disk storage.
- It uses its own engine (`ENGINE=PERFORMANCE_SCHEMA`)
- It is fast(most of the times) and flexible (you can define what events you want to monitor).
- We can extract information using SQL.
- It collects events using instrumentation points.
- Consistently extending the instruments on each release.

P_S Characteristics



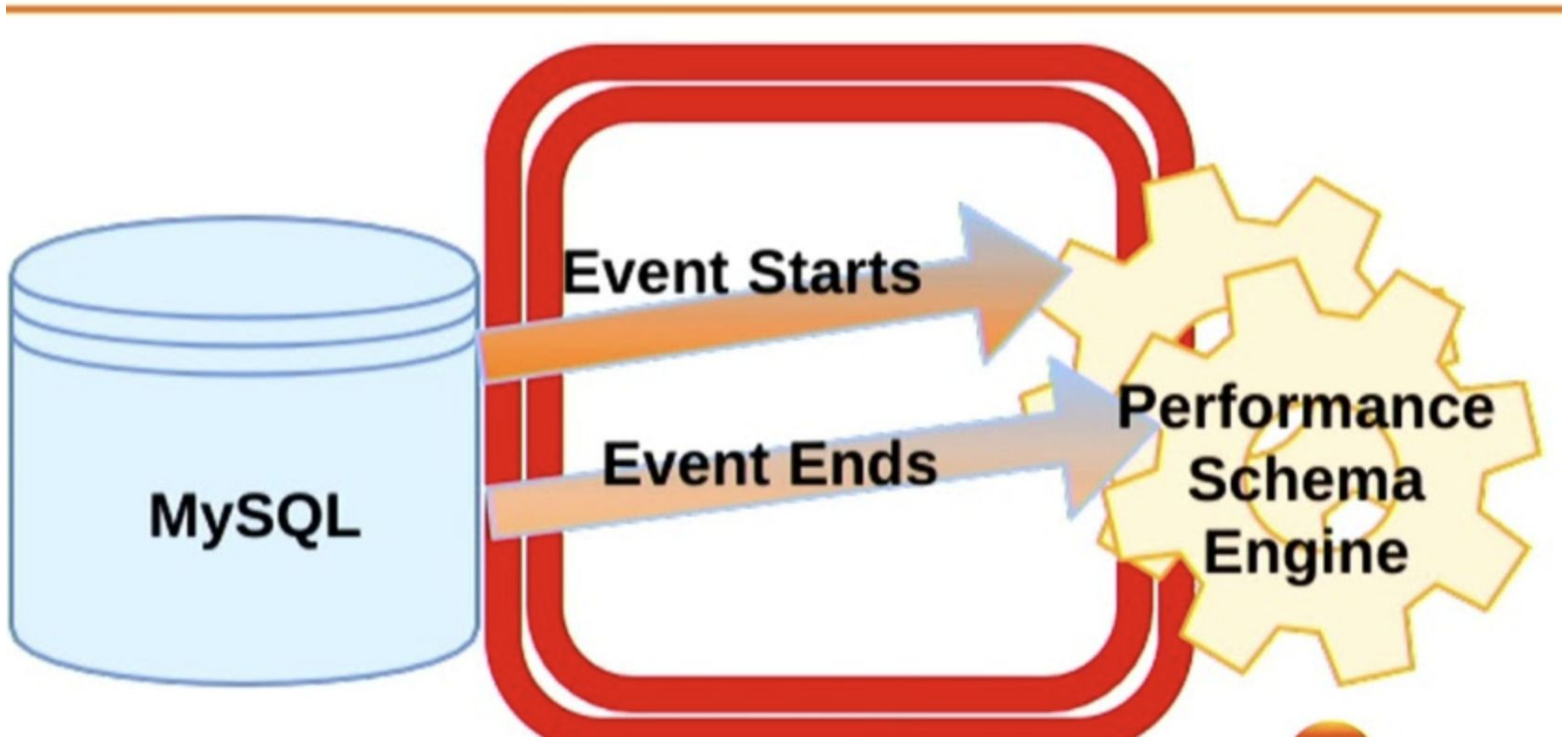
P_S Characteristics

- 8.0.32 (Latest Community GA)
    - 45 (44) variables
    - 111 (102) tables
    - 1241 (1229) Instruments

QUIZ

- What is the latest P_S system variable added to MySQL 8.0?

What are events and instruments?



What are events and instruments?

- Example:
  - Instruments wraps the diagnosed code
  - https://github.com/mysql/mysql-server/blob/1bfe02bdad6604d54913c62614bde57a055c8332/storage/innobase/log/log0buf.cc#L501

What are events and instruments?

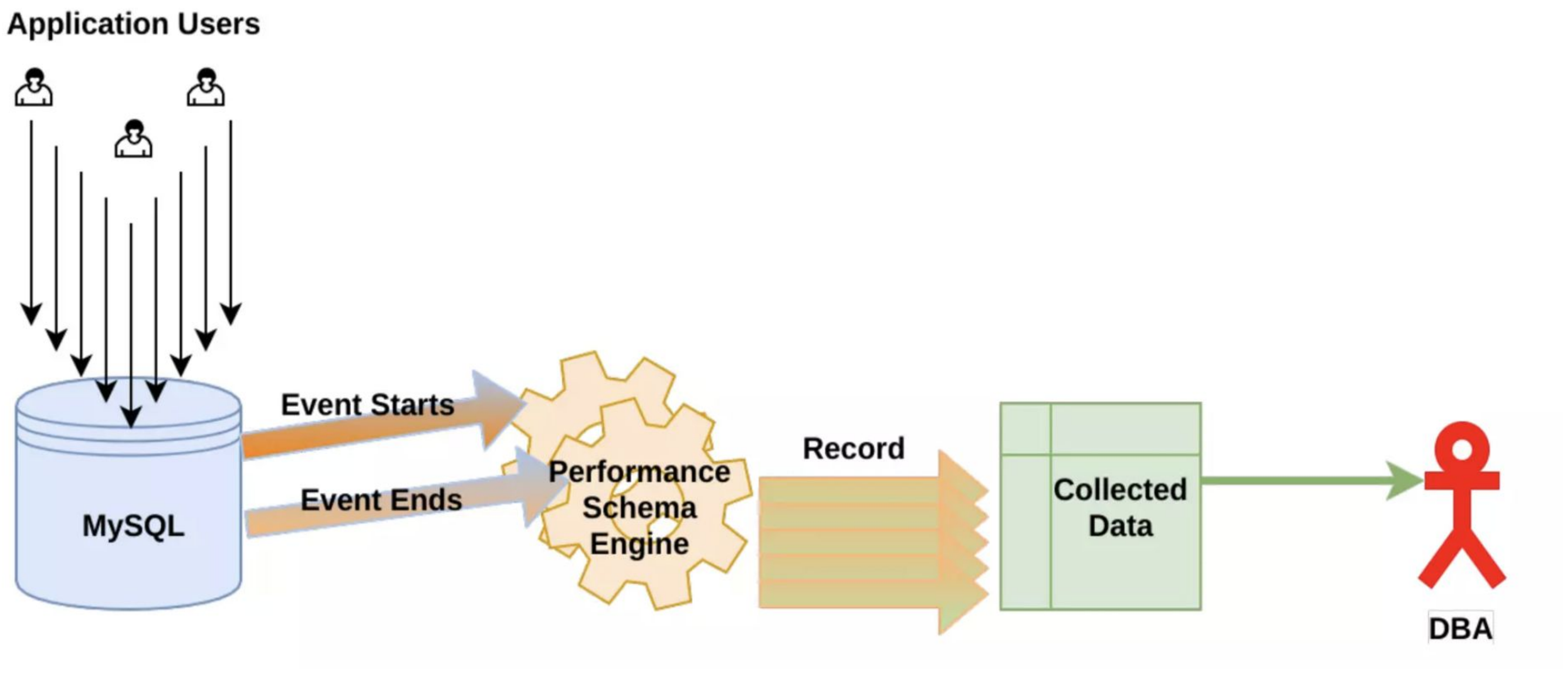
```
#ifdef UNIV_PFS_RWLOCK
[...]
      /* Instrumented to inform we are acquiring a shared rwlock */
      locker = PSI_RWLOCK_CALL(start_rwlock_rdwait)(
          &state, log.pfs_psi, PSI_RWLOCK_SHAREDLOCK, __FILE__,
          static_cast<uint>(__LINE__));
[...]
       if (locker != nullptr) {
         PSI_RWLOCK_CALL(end_rwlock_rdwait)(locker, 0);
       }
```

Disabling CMake Options

- It is possible to compile MySQL with instrumentation disabled.

Formats	Description	Default
DISABLE_PSI_COND	Exclude Performance Schema condition instrumentation	OFF
DISABLE_PSI_DATA_LOCK	Exclude the performance schema data lock instrumentation	OFF
DISABLE_PSI_ERROR	Exclude the performance schema server error instrumentation	OFF
DISABLE_PSI_FILE	Exclude Performance Schema file instrumentation	OFF
DISABLE_PSI_IDLE	Exclude Performance Schema idle instrumentation	OFF

How it works?


Application Users
MySQL
Event Starts
Event Ends
Performance Schema Engine
Record
Collected Data
DBA

What can we do with P_S?

What can we do with P_S?

- Statements
- Memory Usage
- Locks
- Replication (Async and Group Replication)
- Variables and MySQL information

P_S Defaults

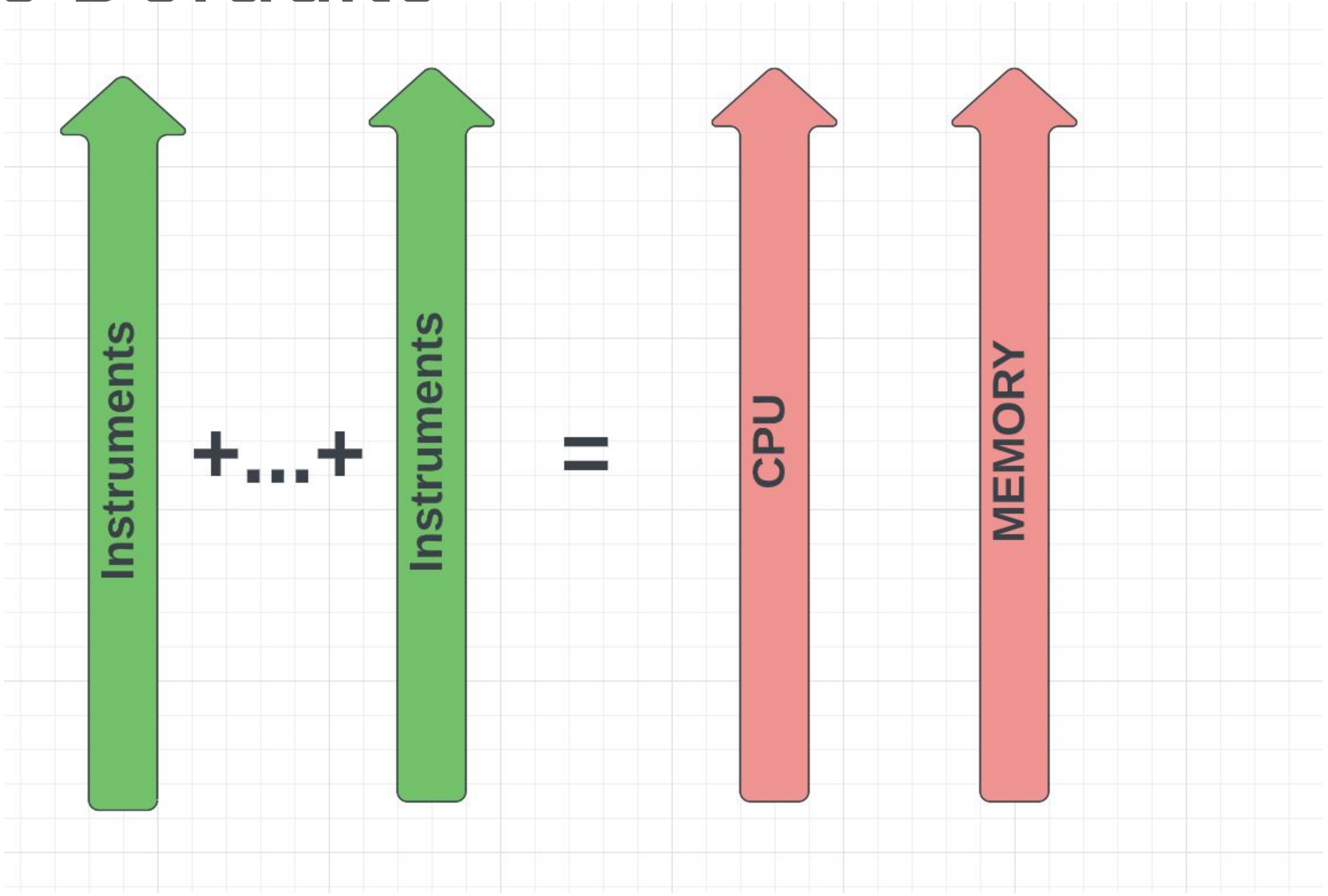
```
mysql> > select * from setup_consumers where enabled like 'YES';
```

NAME	ENABLED
events_statements_current	YES
events_statements_history	YES
events_transactions_current	YES
events_transactions_history	YES
global_instrumentation	YES
thread_instrumentation	YES
statements_digest	YES

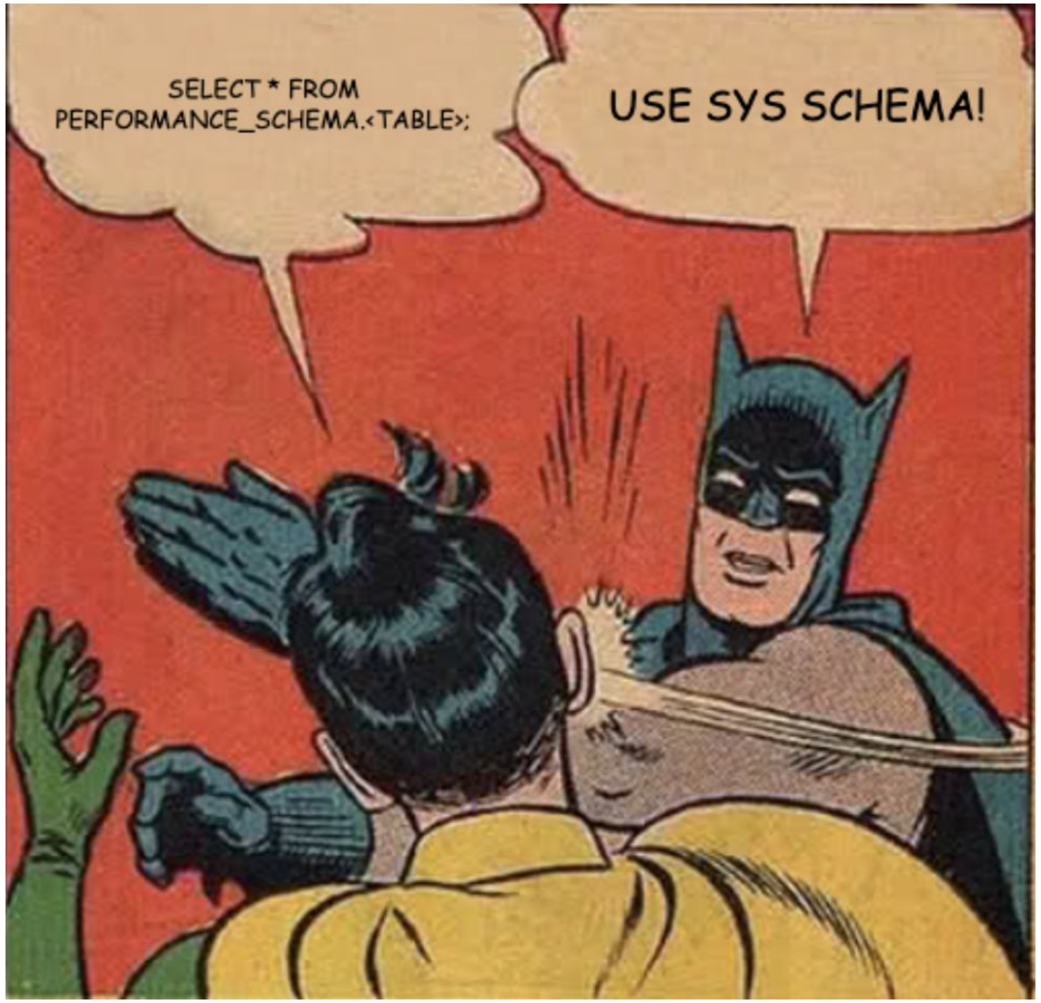
P_S enable/disable

```
mysql>  UPDATE setup_instruments SET ENABLED = 'YES', TIMED = 'YES'

        WHERE NAME LIKE 'statement/%';




mysql>  UPDATE setup_consumers SET ENABLED = 'YES'

        WHERE NAME LIKE '%statements%';
```

P_S Defaults



Great, everything is in P_S!



Statements

```
mysql (performance_schema) > show tables like '%statement%';
+----------------------------------------------------+
| Tables_in_performance_schema (%statement%)         |
+----------------------------------------------------+
| events_statements_current                          |
| events_statements_histogram_by_digest              |
| events_statements_histogram_global                 |
| events_statements_history                          |
[...]
| events_statements_summary_by_thread_by_event_name  |
| events_statements_summary_by_user_by_event_name    |
| events_statements_summary_global_by_event_name     |
| prepared_statements_instances                      |
+----------------------------------------------------+
13 rows in set (0.00 sec)
```

Statements

```
mysql (sys)> > SHOW TABLES LIKE 'statements%';
+---------------------------------------------+
| Tables_in_sys (statements%)                 |
+---------------------------------------------+
| statements_with_errors_or_warnings          |
| statements_with_full_table_scans            |
| statements_with_runtimes_in_95th_percentile |
| statements_with_sorting                     |
| statements_with_temp_tables                 |
+---------------------------------------------+
5 rows in set (0.01 sec)
```

Statements

```
mysql (sys)(sys) > show create table
statements_with_full_table_scans\G
*************************** 1. row ***************************
               View: statements_with_full_table_scans
        Create View: CREATE ALGORITHM=MERGE
DEFINER=`mysql.sys`@`localhost` SQL SECURITY INVOKER VIEW
`statements_with_full_table_scans`
(`query`,`db`,`exec_count`,`total_latency`,`no_index_used_count
`,`no_good_index_used_count`,`no_index_used_pct`,`rows_sent`,`r
ows_examined`,`rows_sent_avg`,`rows_examined_avg`,`first_seen`,
`last_seen`,`digest`) AS select
`sys`.`format_statement`(`performance_schema`.`events_statement
s_summary_by_digest`.`DIGEST_TEXT`) AS
`query`,`performance_schema`.`events_statements_summary_by_dige
st`.`SCHEMA_NAME` AS
`db`,`performance_schema`.`events_statements_summary_by_digest`
.`COUNT_STAR` AS
```

Statements

```
mysql (sys) > select * from statements_with_full_table_scans\G
*************************** 1. row ***************************
                   query: SELECT COUNT ( * ) FROM `joinit` WHERE `g` = ?
                      db: test
              exec_count: 1
           total_latency: 719.05 ms
     no_index_used_count: 1
no_good_index_used_count: 0
       no_index_used_pct: 100
               rows_sent: 1
           rows_examined: 2097152
           rows_sent_avg: 1
       rows_examined_avg: 2097152
              first_seen: 2023-02-03 03:49:14.992202
               last_seen: 2023-02-03 03:49:14.992202
                  digest: 
1dd5a3d060dcb43bb9fbd9934f2dfd050dd496cad58dff5e8f286c0ed1329db7
```

Which Queries Do Not Use Indexes?

```
SELECT THREAD_ID TID, SUBSTR(SQL_TEXT, 1, 50) SQL_TEXT, ROWS_SENT,
   ROWS_EXAMINED RE, CREATED_TMP_TABLES, NO_INDEX_USED, NO_GOOD_INDEX_USED
    FROM performance_schema.events_statements_history
   WHERE NO_INDEX_USED=1 OR NO_GOOD_INDEX_USED=1\G
```

Memory Usage

```
mysql (sys) > show tables like '%memory%';
+-------------------------------------+
| Tables_in_sys (%memory%)            |
+-------------------------------------+
| memory_by_host_by_current_bytes     |
| memory_by_thread_by_current_bytes   |
| memory_by_user_by_current_bytes     |
| memory_global_by_current_bytes      |
| memory_global_total                 |
| x$memory_by_host_by_current_bytes   |
| x$memory_by_thread_by_current_bytes |
| x$memory_by_user_by_current_bytes   |
| x$memory_global_by_current_bytes    |
| x$memory_global_total               |
+-------------------------------------+
10 rows in set (0.00 sec)
```

Memory Usage – What I did in the past

```
 SELECT if(processlist_user is null,
           substring_index(t.name, '/', -2),
           processlist_user) AS user,
        t.processlist_db AS db,
        m.current_count_used AS curr_count,
        sys.format_bytes(current_number_of_bytes_used) curr_alloc,
        count_alloc,
        sys.format_bytes(sum_number_of_bytes_alloc) total_alloc,
        count_free,
        sys.format_bytes(sum_number_of_bytes_free) total_free
   FROM performance_schema.threads t
   JOIN performance_schema.memory_summary_by_thread_by_event_name m using
(thread_id)
  WHERE thread_id = 80
  ORDER BY current_number_of_bytes_used DESC;
```

Memory Usage - NOW()

```
mysql(sys) > SELECT * FROM x$memory_by_thread_by_current_bytes where
thread_id=479\G
*************************** 1. row ***************************
         thread_id: 479
              user: msandbox@localhost
current_count_used: 40643
 current_allocated: 14040184
 current_avg_alloc: 345.4515
 current_max_alloc: 7775192
   total_allocated: 1328003047
```

Locks

- Metadata Locks
- Data locks

Metadata Locks

Metadata locks are locks on the table itself to prevent concurrent changes to its structure.

Metadata Locks

```
mysql> SELECT * FROM sys.schema_table_lock_waits\G
*************************** 1. row ***************************
               object_schema: test
                 object_name: sbtest1
           waiting_thread_id: 82
                 waiting_pid: 43
             waiting_account: msandbox@localhost
           waiting_lock_type: EXCLUSIVE
       waiting_lock_duration: TRANSACTION
               waiting_query: alter table sbtest1 add column c_int INT
 [...]
          blocking_thread_id: 71
                blocking_pid: 32
 [...]
          blocking_lock_type: SHARED_WRITE
      blocking_lock_duration: TRANSACTION
     sql_kill_blocking_query: KILL QUERY 32
```

Data Locks

Unlike most Performance Schema data collection, there are no instruments for controlling whether data lock information is collected or system variables for controlling data lock table sizes. The Performance Schema collects information that is already available in the server, ***so there is no memory or CPU overhead to generate this information*** *or need for parameters that control its collection.*

Data Locks

```
mysql> SELECT * FROM performance_schema.data_locks\G
*************************** 1. row ***************************
               ENGINE: INNODB
       ENGINE_LOCK_ID: 140492108675664:1068:140492004192328
ENGINE_TRANSACTION_ID: 1777
            THREAD_ID: 55
             EVENT_ID: 27
        OBJECT_SCHEMA: test
          OBJECT_NAME: sbtest7
[..]
OBJECT_INSTANCE_BEGIN: 140492004192328
            LOCK_TYPE: TABLE
            LOCK_MODE: IX
          LOCK_STATUS: GRANTED
            LOCK_DATA: NULL
```

Replication

- I/O Thread
  - replication_connection_status

- SQL thread
  - replication_applier_status
  - replication_applier_status_by_coordinator - MTS only
  - replication_applier_status_by_worker
  - replication_applier_global_filters
  - replication_applier_filters

- Group replication
  - replication_group_members
  - replication_group_member_stats

Replication - SHOW REPLICA STATUS\G

```
mysql > select * from replication_connection_configuration  join
replication_applier_configuration using (channel_name)\G
******************* 1. row ***************************
                CHANNEL_NAME:
                        HOST: 127.0.0.1
                        PORT: 47009
                        USER: rsandbox
           NETWORK_INTERFACE:
               AUTO_POSITION: 0
                 SSL_ALLOWED: NO
                 SSL_CA_FILE:
                 SSL_CA_PATH:

[...]
```

Replication

```
mysql(performance_schema) > select * from replication_group_members\G
*************************** 1. row ****************************
              CHANNEL_NAME: group_replication_applier
                 MEMBER_ID: 00049008-1111-1111-1111-111111111111
               MEMBER_HOST: 127.0.0.1
               MEMBER_PORT: 49008
              MEMBER_STATE: ONLINE
               MEMBER_ROLE: PRIMARY
            MEMBER_VERSION: 8.0.31
MEMBER_COMMUNICATION_STACK: XCom
[...]
```

Variables And MySQL Information

```
mysql (performance_schema) > show tables like '%variables%';
+----------------------------------------+
| Tables_in_performance_schema (%variables%) |
+----------------------------------------+
| global_variables                       |
| persisted_variables                    |
| session_variables                      |
| user_variables_by_thread               |
| variables_by_thread                    |
| variables_info                         |
+----------------------------------------+
6 rows in set (0.00 sec)
```

Variables And MySQL Information

```
mysql(performance_schema)> select * from variables_by_thread
where thread_id = 96 and variable_name like 'wait_timeout';
+-----------+---------------+----------------+
| THREAD_ID | VARIABLE_NAME | VARIABLE_VALUE |
+-----------+---------------+----------------+
|        96 | wait_timeout  | 1000           |
+-----------+---------------+----------------+

mysql> show global variables like 'wait_timeout';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| wait_timeout  | 28800 |
+---------------+-------+
1 row in set (0.00 sec)
```

Variables And MySQL Information

```
mysql> show global variables like
'performance_schema_show_processlist';
+-------------------------------------+-------+
| Variable_name                       | Value |
+-------------------------------------+-------+
| performance_schema_show_processlist | ON    |
+-------------------------------------+-------+
1 row in set (0.00 sec)
```

Variables And MySQL Information

```
mysql> select * from performance_schema.processlist\G
[...]
     ID: 23
   USER: root
   HOST: localhost
     DB: NULL
COMMAND: Query
   TIME: 0
  STATE: executing
   INFO: select * from performance_schema.processlist
2 rows in set (0.00 sec)
```

Questions?

Percona is a world-class open source database software, support, and services company focused on helping you scale and innovate with speed as you grow.

83M+
Software Downloads
TTM as of November 2022

100K+
Blog Views Per Month

800+
Customers

40%
YoY MRR Growth
As of Q4 2021

Trusted by...

- More than a third of the Fortune 50
- 4 of the top 6 Retailers
- 3 of the top 5 Healthcare Companies
- 9 of the top 10 Tech Companies
- 6 of the top 10 Gaming Companies
- 4 of the top 5 Manufacturing Companies

谢谢

Thank you

Grazie

Obrigado

Gracias

Percona is a world-class open source database software, support, and services company focused on helping you scale and innovate with speed as you grow.

83M+	100K+	800+	40%
Software Downloads	Blog Views Per Month	Customers	YoY MRR Growth
TTM as of November 2022			As of Q4 2021

Trusted by…

- More than a third of the Fortune 50
- 4 of the top 6 Retailers
- 3 of the top 5 Healthcare Companies
- 9 of the top 10 Tech Companies
- 6 of the top 10 Gaming Companies
- 4 of the top 5 Manufacturing Companies