MySQL Performance Schema

in 1 hour

March, 21, 2019 Sveta Smirnova



Table of Contents

- •Why Performance Schema?
- •How it Works?
- •How to Tune?
- •How to Ease?



Why Performance Schema?



Metadata vs Runtime

Metadata

Schema
Supported charsets etc.

Runtime

Resource usage Acquired locks Running queries

Difference



Metadata vs Runtime

Metadata

Always existed

- SHOW commands
- Information Schema

Runtime

Was not perfect

- Status variables
- Extensions for Information Schema

History



Metadata vs Runtime

Metadata
Information Schema

Runtime
Performance Schema

Now



- No design issues like in Information Schema
 - No locking reads



Almost



- No design issues like in Information Schema
 - No locking reads
- Optimized
 - It took one and a half major versions



- No design issues like in Information Schema
 - No locking reads
- Optimized
 - It took one and a half major versions
- Consistently extending
 - New features in every release



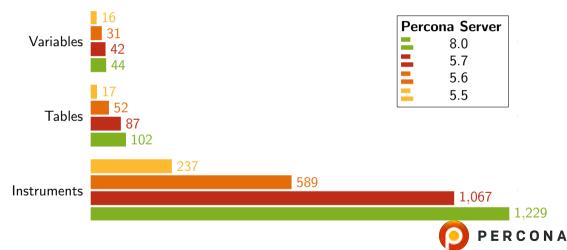
Easy to add your own



- No design issues like in Information Schema
 - No locking reads
- Optimized
 - It took one and a half major versions
- Consistently extending
 - New features in every release
- Easy to add your own
- Being mature



Performance Schema Over Years



5.5: Server insights

```
$ iostat -p nvmeOn1 5
          %user
                  %nice %system %iowait
                                          %steal
                                                    %idle
avg-cpu:
          73.02
                   0.00
                           12.03
                                    6.37
                                            0.00
                                                     8.59
Device
                           kB_read/s
                                        kB_wrtn/s
                                                      kB_read
                                                                 kB_wrtn
                   tps
               1235.80
nvmeOn1
                                5.60
                                         16662.40
                                                           28
                                                                   83312
nvme0n1p3
               1158.00
                                5.60
                                         16662.40
                                                           28
                                                                   83312
```



5.5: Server insights

thread_id EVENT_NAME	sum_kb
10 wait/io/file/innodb/innodb_parallel_dblwrite_file	1312.0000
117 wait/io/file/sql/binlog	3.4023
113 wait/io/file/sql/binlog	1.7012
58 wait/io/file/sql/query_log	0.2031

⁴ rows in set (0.00 sec)



5.5: Server insights

5.6: Statements



- 5.5: Server insights
- 5.6: Statements
- 5.6: Stages

ERCONA

5.7: Prepared statements

```
mysql> select SQL_TEXT, sum(COUNT_EXECUTE) from prepared_statements_instances group by SQL_TEXT;
 SQL TEXT
                                                            sum(COUNT EXECUTE)
 COMMIT
                                                                         11035
 UPDATE sbtest1 SET k=k+1 WHERE id=?
                                                                         11872
 UPDATE sbtest1 SET c=? WHERE id=?
                                                                         11606
 DELETE FROM sbtest1 WHERE id=?
                                                                         11336
  INSERT INTO sbtest1 (id, k, c, pad) VALUES (?, ?, ?, ?)
                                                                         11076
5 rows in set (0.22 sec)
```



5.7: Stored routines



5.7: Stored routines

```
mysql> call sp_test(NULL);
Query OK, 1 row affected (0.07 sec)
mysql> select thread_id, event_name, sql_text from events_statements_history
    -> where event_name like 'statement/sp%';
I thread id I event name
                                       I sal text
         24 | statement/sp/hpush_jump | NULL
         24 | statement/sp/stmt
                                      | INSERT INTO t1 VALUES(val)
         24 | statement/sp/stmt
                                        INSERT IGNORE INTO t1 VALUES('Som...
         24 | statement/sp/stmt
                                        GET STACKED DIAGNOSTICS CONDITION...
         24 | statement/sp/stmt
                                        GET STACKED DIAGNOSTICS CONDITION...
         24 | statement/sp/hreturn
                                       I NULL
         24 | statement/sp/hpop
                                       I MULT.
```

7 rows in set (0.00 sec)



5.7: Metadata Locks

mysql> select id, db, state, info, time from information_schema.processlist; I id I db lstate info l time sbtest NUL.L. 65 I NULT. | Waiting on empty queue I NULL. I 5451 I performance_schema | executing | select id, db, state, i... | sbtest | Waiting for table metadata lock | alter table sbtest1 add... |

4 rows in set (0.00 sec)



5.7: Metadata Locks

```
mysql> select processlist_id, object_type, lock_type, lock_status, source
    -> from metadata_locks join threads on (owner_thread_id=thread_id)
    -> where object_schema='sbtest' and object_name='sbtest1';
| processlist_id | object_type | lock_type
                                                   | lock_status | source
             96 | TABLE
                               | SHARED_READ
                                                   GRANTED
                                                                 | sql_parse.cc:5850
             95 | TABLE
                                SHARED_UPGRADABLE | GRANTED
                                                                  sql_parse.cc:5850
             95 | TABLE
                                EXCLUSIVE
                                                   | PENDING
                                                                 | mdl.cc:3685
```

3 rows in set (0.00 sec)



5.7: Memory usage



5.7: Memory usage

```
mysql> select event_name, count_alloc, count_free,
```

- -> CURRENT_NUMBER_OF_BYTES_USED/1024/1024 as used_mb
- -> from memory_summary_global_by_event_name where CURRENT_NUMBER_OF_BYTES_USED > 0
- -> order by used_mb desc limit 8;

event_name	count_alloc	count_free	used_mb		
memory/sql/JOIN_CACHE	15	l 6	9216.00000000		
memory/innodb/ut0link_buf	1 2	1 0	24.00004578		
memory/innodb/ut0new	1 6	1 0	16.07953835		
memory/innodb/lock0lock	6241	0	10.92063904		
memory/mysys/KEY_CACHE	1 3	0	8.00205994		
memory/innodb/buf0dblwr	378	370	5.80024719		
memory/innodb/memory	15686935	15682985	4.90249634		
memory/innodb/ut0pool	1	0	4.000022		

8 rows in set (0.04 sec)

5.7: Replication

```
mysql> show slave status \G
Slave_IO_State: Waiting for master to send event
        Master_Host: 127.0.0.1
        Master_User: root
        Master Port: 13000
       Connect_Retry: 60
     Master_Log_File: master-bin.000002
 Read_Master_Log_Pos: 63810611
      Relay_Log_File: slave-relay-bin-master@002d1.000004
      Relay_Log_Pos: 1156
Relay_Master_Log_File: master-bin.000001
    Slave_IO_Running: Yes
   Slave_SQL_Running: No
     Replicate_Do_DB:
 Replicate_Ignore_DB:
```



5.7: Replication

- 10 Thread
 - replication_connection_status
 - replication_connection_status

SQL Thread

- replication_applier_configuration
- replication_applier_filters
- replication_applier_global_filters
- replication_applier_status
- replication_applier_status_by_coordinator
- replication_applier_status_by_worker

Group replication

- replication_group_member_stats
- replication_group_members



5.7: Variables

18 rows in set (0.06 sec)

```
mysql> select * from variables_by_thread where variable_name='join_buffer_size';
 THREAD_ID | VARIABLE_NAME
                              | VARIABLE_VALUE
        129 | join_buffer_size | 262144
        132 | join_buffer_size | 262144
        133 | join_buffer_size | 262144
        134 | join_buffer_size | 262144
        135 | join_buffer_size | 262144
        144 | join_buffer_size | 262144
        145 | join_buffer_size |
                                262144
        146 | join_buffer_size | 262144
        147 | join_buffer_size | 262144
        148 | join_buffer_size | 1073741824
                                                                             PERCONA
```

5.7: Variables



5.7: Variables



8.0: Errors

mysql> select ERROR_NUMBER NUM, ERROR_NAME, SUM_ERROR_RAISED SUM, FIRST_SEEN, LAST_SEEN

-> from events_errors_summary_global_by_error where SUM_ERROR_RAISED > 0

-> order by SUM_ERROR_RAISED desc:

	·		. ,								
NUM		ERROR_NAME	SUM			FIRST_SEEN		1	LAST_SEEN		†
1213		ER_LOCK_DEADLOCK	1798	86	+- 	2019-01-20	20:51:34	1	2019-01-20	23:05:43	Ī
1287	1	ER_WARN_DEPRECATED_SYNTAX	1 20	7		2019-01-20	20:51:41	1	2019-01-20	23:03:29	1
1158	- 1	ER_NET_READ_ERROR	9	96		2019-01-20	20:57:36	1	2019-01-20	23:05:44	1
1205	- 1	ER_LOCK_WAIT_TIMEOUT	1 7	2		2019-01-20	20:52:19	1	2019-01-20	23:04:57	1
1295	- 1	ER_UNSUPPORTED_PS	1 6	34		2019-01-20	20:51:33	1	2019-01-20	23:03:26	1
3719	-	ER_DEPRECATED_UTF8_ALIAS	1 3	33		2019-01-20	20:50:33	1	2019-01-20	20:50:34	1
1064	- 1	ER_PARSE_ERROR	1	2		2019-01-20	21:11:33	1	2019-01-20	21:11:33	
1317	1	ER_QUERY_INTERRUPTED	I	2	1	2019-01-20	21:24:23	1	2019-01-20	21:29:37	
1054	- 1	ER_BAD_FIELD_ERROR	1	1	1	2019-01-20	22:14:57	1	2019-		
1066	-	ER_NONUNIQ_TABLE	1	1		2019-01-20	21:23:45	1	2019	PERC	0
1160	- 1	ER NET ERROR ON WRITE	1	1	1	2019-01-20	21:51:36	1	2019-		

- 5.7: Prepared statements
- 5.7: Stored routines
- 5.7: Metadata Locks
- 5.7: Memory usage
- 5.7: Replication
- 5.7: Variables
- 8.0: Errors
 - More



How it Works? PERCONA

Gets the Data

Wraps the diagnosed code

```
locker = PSI_RWLOCK_CALL(start_rwlock_wrwait)(
    &state, lock->pfs_psi, PSI_RWLOCK_TRYEXCLUSIVELOCK,
    file_name, static_cast<uint>(line));

ret = rw_lock_x_lock_func_nowait(lock, file_name, line);

if (locker != NULL) {
    PSI_RWLOCK_CALL(end_rwlock_wrwait)(locker, static_cast<int>(ret));
}
```



Stores the Data

```
CREATE TABLE 'events NAME current' (
'[OWNER_]THREAD_ID' bigint(20) unsigned NOT NULL,
'[OWNER_]EVENT_ID' bigint(20) unsigned NOT NULL,
'END_EVENT_ID' bigint(20) unsigned DEFAULT NULL,
'EVENT NAME' varchar(128) NOT NULL.
'SOURCE' varchar(64) DEFAULT NULL,
'TIMER_START' bigint(20) unsigned DEFAULT NULL,
'TIMER_END' bigint(20) unsigned DEFAULT NULL.
'TIMER_WAIT' bigint(20) unsigned DEFAULT NULL.
'OBJECT SCHEMA' varchar(64) DEFAULT NULL.
'OBJECT_NAME' varchar(512) DEFAULT NULL.
'OBJECT TYPE' varchar(64) DEFAULT NULL.
'OBJECT_INSTANCE_BEGIN' bigint(20) unsigned NOT NULL,
'NESTING_EVENT_ID' bigint(20) unsigned DEFAULT NULL,
'NESTING_EVENT_TYPE' enum('TRANSACTION', 'STATEMENT', 'STAGE', 'WAIT') DEFAULT NULL,
EVENT-SPECIFIC-FIELDS
) ENGINE=PERFORMANCE SCHEMA
```

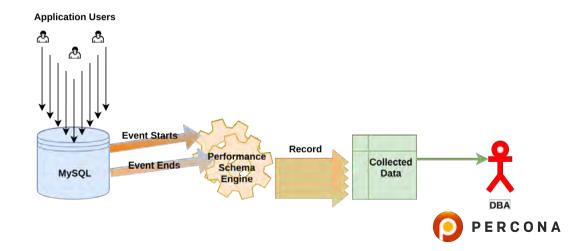


Returns the Data

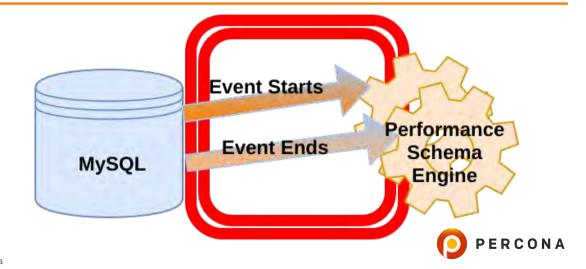
- SQL
- Same basic structure for events_* tables
- Others are unified whenever possible



How Does Performance Schema Works?



Instruments: Get the Data



Performance Note

 Every instrumented call is wrapped into two Performance Schema calls



Performance Note

- Every instrumented call is wrapped into two Performance Schema calls
- For rarely used calls no performance impact
 - Memory allocation
 - Statement-related data



Performance Note

- Every instrumented call is wrapped into two Performance Schema calls
- For rarely used calls no performance impact
 - Memory allocation
 - Statement-related data
- Significant impact for often used



- Table of 2048 rows
- *select * from joinit;
- Test preparation

```
mysql> truncate performance_schema.events_waits_history_long;
Query OK, 0 rows affected (0.02 sec)
-- OR
mysql> truncate performance_schema.events_statements_history_long;
Query OK, 0 rows affected (0.02 sec)
```



- Table of 2048 rows
- *select * from joinit;
- The test

```
mysql> \P sha1sum
PAGER set to 'sha1sum'
mysql> select * from joinit;
7f188e9c9fd628613eab85366f0729604b8b5e3e -
2048 rows in set (0.09 sec)
```



- Table of 2048 rows
- *select * from joinit;
- events_waits*

```
mysql> select event_name, count(*) from performance_schema.events_waits_history_long
    -> where thread_id=42 group by EVENT_NAME order by 'count(*)' desc;
```



. . .

- Table of 2048 rows
- *select * from joinit;
- events_statements*

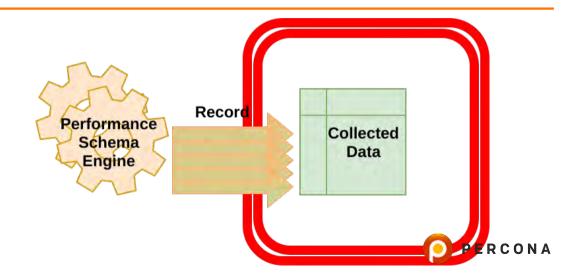
```
mysql> select event_name, count(*) from performance_schema.events_statements_history_long
    -> where thread_id=42 group by EVENT_NAME;
```



- Table of 2048 rows
- *select * from joinit;
- Depends on the instrument!



Consumers: Store the Data



Performance Schema Limitations

- Must be supported by the component
- Collects data only after they are enabled
- Never frees allocated memory



How to Tune? PERCONA

Performance Schema Defaults

- Defaults
 - ON
- **5.7:** Only global, thread, statements and transactions instrumentation
- 8.0: Memory and MDL



Performance Schema Defaults

- Defaults
 - ON
- **5.7:** Only global, thread, statements and transactions instrumentation
- 8.0: Memory and MDL
- All other instruments/consumers disabled



Two Places

- Configuration options
- setup_* tables



Startup Configuration

• performance_schema = ON



Startup Configuration

- performance_schema = ON
- Limits of table size and number of instrumented instances
 - Not dynamic!
 - Examples
 - performance_schema_accounts_size
 - performance_schema_max_thread_instances
 - $\color{red} \blacksquare \hspace{0.1cm} \texttt{performance_schema_max_thread_classes}$



Startup Configuration

- performance_schema = ON
- Instruments and consumers, enabled at startup
 - Pattern
 - --performance-schema-consumer-consumer_name=value
 - --performance-schema-instrument='instrument_name=value'

Examples

- --performance-schema-consumer-events_transactions_history_long=ON
- --performance-schema-instrument='%=0FF'
- --performance-schema-instrument='memory/innodb/%=ON'
- --performance-schema-instrument='wait/lock/metadata/sql/mdl=ON'



setup_* tables



- setup_* tables
- Actors: user connections

3 rows in set (0.00 sec)



- setup_* tables
- Threads: thread classes



- setup_* tables
- Threads: thread classes



- setup_* tables
- Objects
 - Types of events, their names and schema



- setup_* tables
- Consumers: what to store

```
mysql> select * from setup_consumers;
 NAME
                                      ENABLED
                                      YES
  events_stages_current
  events_stages_history
                                      YES
  events_stages_history_long
  events statements current
                                      YES
  events_statements_history
                                      YES
  events_statements_history_long
                                      NΩ
  events_transactions_current
                                      YES
  events_transactions_history
                                      YES
```

events_transactions_history_long



. . .

- setup_* tables
- Instruments: what to instrument

```
mysql> select NAME, ENABLED, TIMED, PROPERTIES, VOLATILITY from setup_instruments
   -> order by rand() limit 3;
```

NAME	ENABLED		PROPERTIES	VOLATILITY
wait/synch/mutex/pfs/LOCK_pfs_share_list	NO	NO	singleton	1
memory/csv/Transparent_file	YES	NULL		0
wait/synch/mutex/sql/THD::LOCK_thd_data	YES	YES		5

3 rows in set (0.02 sec)



How to Ease? PERCONA

sys Schema

- Views on Performance Schema tables
- Stored routines



sys Schema

- Views on Performance Schema tables
- Stored routines
- Easier configuration
- Shortcuts to typical use cases



sys Schema

- Views on Performance Schema tables
- Stored routines
- Easier configuration
- Shortcuts to typical use cases
- 5.7+: Installed by default
- Before: github.com/mysql/mysql-sys



Nicer Views

- statement_analysis
- statements_with_full_table_scans
- * statements_with_runtimes_in_95th_percentile
- * statements_with_sorting
- statements_with_temp_tables
- statements_with_errors_or_warnings
- memory_by_thread_by_current_bytes
- More



Compare!

- Performance Schema
 - SELECT THREAD_ID, DIGEST, ROWS_SENT RS, ROWS_EXAMINED. CREATED_TMP_TABLES. NO_INDEX_USED, NO_GOOD_INDEX_USED FR.OM performance_schema.events_statements_history WHERE NO INDEX USED=1 OR NO GOOD INDEX USED=1 PERCONA

Compare!

- sys schema
 - * select * from sys.statements_with_full_table_scans



Easier configuration

Enable

```
call sys.ps_setup_enable_consumer(YOUR_CONSUMER);
call sys.ps_setup_enable_instrument(YOUR_INSTRUMENT);
```



Easier configuration

Disable

```
call sys.ps_setup_disable_consumer(YOUR_CONSUMER);
call sys.ps_setup_disable_instrument(YOUR_INSTRUMENT);
```



Easier configuration

Examine

```
call sys.ps_setup_show_disabled_consumers();
call sys.ps_setup_show_disabled_instruments();
call sys.ps_setup_show_enabled_consumers();
call sys.ps_setup_show_enabled_instruments();
```



Performance Schema Dashboard



23:05:47



Performance Schema Dashboard



23:32:15



Performance Schema Dashboard



22:27:15



Performance Schema Dashboard



18:08:15



More information

- 🔯 Blog of MySQL developers team
- Mark Leith: author of sys schema
- Official reference manual
- Webinar ''Performance Schema for MySQL Troubleshooting''
- PMM Demo



Thank you!

