



PERCONA
Performance Consulting Experts

Using MySQL 5.5 Performance Schema

Apr 11-14, 2011

MySQL Conference and Expo

Santa Clara, CA

by Peter Zaitsev, Percona Inc

Intro to Performance Schema

- New feature of MySQL 5.5
- Gathering detailed diagnostic information
- Focused on “waits” - locks, file IO etc
- Information stored in tables
 - Accessible via SQL Interface
- Designed with low overhead in mind
 - Lock-less itself
 - Overhead can range from 0 to 20%+
 - Workload and Settings Related
- Being improved. MySQL 5.6 has more cool stuff

Origins of Performance Schema

- Historically MySQL Instrumentation was poor
 - Designed MyISAM, Single CPU, Simple Queries in mind
 - Status variables, slow query log...
- We had issues understanding causes of performance issues
- Miracle Conference 2003, Denmark
 - Oracle and MS Focused conference with MySQL “for fun”
 - Cary Millsap, Jonathan Lewis as speakers
 - “Oracle Wait Interface” de-facto standard for serious analyses
 - Exists in Oracle since version 7.0 (released in 1992)
 - I claimed we get it in MySQL before version 7 :)

Development

- I could not get any development done
- MySQL was pushing ahead “Enterprise” features
 - Getting SAP/R3 to run on MySQL was the deal
- 2005 Renewed interest of Performance Schema in relationship to “Merlin” project
- 2008 Development started by Marc Alff
 - I was long gone by that time
- April 2009 - Initial Release of source code
- December 2010 - Released in MySQL 5.5 GA

Performance Schema Design

- Tracking Events
 - In 5.5 limited to “wait” events
- Events reported by “Instruments”
 - Example: wait/io/file/innodb/innodb_data_file
- Multiple Instances of same instrument
 - Different mutexes, files, etc
- Events are processed by “Consumers”
 - Stored as a history or aggregated
 - No “Logging” available at this point
- Can setup which instruments and consumers to enable **dynamically**

Timing

- Timing can be enabled or disabled on per instrument basics
 - See “**setup_instruments**” table
- Different timers are available
 - Default **CYCLE** - lowest overhead
 - See **performance_timers**, **setup_timers** tables
- Events times are stored in picoseconds
 - Can be approximate due to variable frequency etc

Enabling MySQL P.S

- Ensure it is compiled in, check “**SHOW ENGINES**”
 - Enabled in binary MySQL and Percona Server distros
- Make sure to run `mysql_upgrade`
 - To create `performance_schema` database and tables
- Enable it in configuration **--performance-schema**
 - Startup only, can't be set online
- Check it is ON:
 - Check `mysql` error log file if it failed to start

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

PS Variables

- Performance Schema Variables
 - Startup only !

```
mysql> show variables like "perf%";
```

Variable_name	Value
performance_schema	ON
performance_schema_events_waits_history_long_size	10000
performance_schema_events_waits_history_size	10
performance_schema_max_cond_classes	80
performance_schema_max_cond_instances	1000
performance_schema_max_file_classes	50
performance_schema_max_file_handles	32768
performance_schema_max_file_instances	10000
performance_schema_max_mutex_classes	200
performance_schema_max_mutex_instances	1000000
performance_schema_max_rwlock_classes	30
performance_schema_max_rwlock_instances	1000000
performance_schema_max_table_handles	100000
performance_schema_max_table_instances	50000
performance_schema_max_thread_classes	50
performance_schema_max_thread_instances	1000

```
16 rows in set (0.00 sec)
```

PS Engine Status

- Shows object sizes and memory usage

```
mysql> show engine performance_schema status ;
```

Type	Name	Status
performance_schema	events_waits_current.row_size	136
performance_schema	events_waits_current.row_count	3000
performance_schema	events_waits_history.row_size	120
performance_schema	events_waits_history.row_count	10000
performance_schema	events_waits_history.memory	1200000
performance_schema	events_waits_history_long.row_size	120
performance_schema	events_waits_history_long.row_count	10000
performance_schema	events_waits_history_long.memory	1200000
performance_schema	(pfs_mutex_class).row_size	248
performance_schema	(pfs_mutex_class).row_count	200
...		
performance_schema	(pfs_table).memory	7200000
performance_schema	performance_schema.memory	394468704

```
51 rows in set (0.00 sec)
```

PS Status Variables

- Performance Schema Status Variables
 - Mainly used for tracking lost information

```
mysql> show status like "perf%";
```

Variable_name	Value
Performance_schema_cond_classes_lost	0
Performance_schema_cond_instances_lost	0
Performance_schema_file_classes_lost	0
Performance_schema_file_handles_lost	0
Performance_schema_file_instances_lost	1
Performance_schema_locker_lost	0
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_table_handles_lost	0
Performance_schema_table_instances_lost	0
Performance_schema_thread_classes_lost	0
Performance_schema_thread_instances_lost	0

```
14 rows in set (0.00 sec)
```

Instrumented Threads

- All threads are instrumented, including background threads

```
mysql> select * from threads;
```

THREAD_ID	PROCESSLIST_ID	NAME
0	0	thread/sql/main
552	535	thread/sql/one_connection
7	0	thread/innodb/io_handler_thread
5	0	thread/innodb/io_handler_thread
6	0	thread/innodb/io_handler_thread
3	0	thread/innodb/io_handler_thread
557	540	thread/sql/one_connection
...		
550	533	thread/sql/one_connection
547	530	thread/sql/one_connection
548	531	thread/sql/one_connection
546	529	thread/sql/one_connection
15	0	thread/innodb/srv_master_thread

```
34 rows in set (0.00 sec)
```

events_waits_current

- Information about latest wait for thread
 - TIMER_END is NULL for event not completed
- Very helpful for “sampling” form of profiling

```
mysql> select * from events_waits_current where thread_id=556 \G
***** 1. row *****
      THREAD_ID: 556
      EVENT_ID: 5269821
      EVENT_NAME: wait/io/file/innodb/innodb_log_file
      SOURCE: fil0fil.c:5290
      TIMER_START: 1286162214787611
      TIMER_END: NULL
      TIMER_WAIT: NULL
      SPINS: NULL
      OBJECT_SCHEMA: NULL
      OBJECT_NAME: /var/lib/mysql/ib_logfile0
      OBJECT_TYPE: FILE
      OBJECT_INSTANCE_BEGIN: 139804768874064
      NESTING_EVENT_ID: NULL
      OPERATION: sync
      NUMBER_OF_BYTES: NULL
      FLAGS: 0
1 row in set (0.00 sec)
```

events_waits_history

- Same structure as events_waits_current
- Stores `performance_schema_events_waits_history_size` last wait entries per thread
 - 10 by default
- Can be seen to see limited event history for every thread

events_waits_history_long

- Another history table
 - Same structure as events_waits_current
- Stores `performance_schema_events_waits_history_long_size` events
 - But globally not per thread
 - 10000 by default
- Can be used to see the limited global history of the system
 - In my tests 10.000 events history was enough for 50ms worth of time

events_waits_summary_by_instance

- Waits summary per instance
- OBJECT_INSTANCE_BEGIN is memory address of the object
- What instances are responsible for most waits ?
 - Note some object have one instance other have many

```
mysql> select * from events_waits_summary_by_instance order by sum_timer_wait
desc limit 1 \G
***** 1. row *****
      EVENT_NAME:
wait/synch/cond/sql/Query_cache::COND_cache_status_changed
OBJECT_INSTANCE_BEGIN: 17187440
      COUNT_STAR: 6145437
      SUM_TIMER_WAIT: 1660879298836863
      MIN_TIMER_WAIT: 0
      AVG_TIMER_WAIT: 270262195
      MAX_TIMER_WAIT: 18446744073709550140
1 row in set (0.02 sec)
```

events_waits_summary_by_thread_by_event_name

- This table name is the longest one in performance_schema if you wonder
- Helpful to see what given thread was waiting for the most

```
mysql> select event_name,count_star,SUM_TIMER_WAIT from
events_waits_summary_by_thread_by_event_name where thread_id=556 order by sum_timer_wait desc
limit 10;
```

event_name	count_star	SUM_TIMER_WAIT
wait/io/file/innodb/innodb_log_file	10526	128411207222319
wait/synch/cond/sql/Query_cache::COND_cache_status_changed	448354	120991275783630
wait/synch/mutex/sql/Query_cache::structure_guard_mutex	2985682	19251903642138
wait/synch/rwlock/innodb/dict_table_stats	820895	3091942331559
wait/synch/mutex/mysys/THR_LOCK::mutex	4036024	2159774524917
wait/synch/mutex/innodb/log_sys_mutex	642894	1936040908401
wait/synch/mutex/innodb/kernel_mutex	702090	1444982387301
wait/synch/mutex/sql/THD::LOCK_thd_data	3950787	1402166395908
wait/synch/rwlock/innodb/buf_pool_page_hash_latch	4606523	1188302684751
wait/synch/mutex/sql/LOCK_open	1009025	747047442366

10 rows in set (0.00 sec)

events_waits_summary_global_by_event_name order

- Summary by Event Name
 - All Instances are event are summed together
- Helpful to see what waits are most impacting the system

```
mysql> select event_name,count_star,sum_timer_wait from
events_waits_summary_global_by_event_name order by SUM_TIMER_WAIT desc limit 5;
```

event_name	count_star	sum_timer_wait
wait/io/file/innodb/innodb_log_file	205438	2552133070220355
wait/synch/cond/sql/Query_cache::COND_cache_status_changed	8405382	2259497326493034
wait/synch/mutex/sql/Query_cache::structure_guard_mutex	55769435	361568224932147
wait/io/file/innodb/innodb_data_file	62423	347302500600411
wait/synch/rwlock/innodb/dict_table_stats	15330162	53005067680923

```
5 rows in set (0.00 sec)
```

file_summary_by_event_name

- I/O Statistics by file Type
- Only Number provided, no timing information

```
mysql> select * from file_summary_by_event_name order by count_read+count_write desc limit 5;
```

EVENT_NAME	COUNT_READ	COUNT_WRITE	SUM_NUMBER_OF_BYTES_READ	SUM_NUMBER_OF_BYTES_WRITE
wait/io/file/innodb/innodb_log_file	12	151264	139264	1531636736
wait/io/file/innodb/innodb_data_file	5450	61747	92241920	2903113728
wait/io/file/myisam/kfile	4414	46926	964363	915379
wait/io/file/myisam/dfile	2186	48884	268169685	267934556
wait/io/file/sql/FRM	703	138	107947	23535

```
5 rows in set (0.00 sec)
```

file_summary_by_instance

- Provides file IO statistics by individual file
- Once again no IO latencies reported
- Can be used to watch per table physical io with **innodb_file_per_table**

```
mysql> select * from file_summary_by_instance order by count_read+count_write desc limit 1 \G
***** 1. row *****
      FILE_NAME: /var/lib/mysql/ib_logfile1
      EVENT_NAME: wait/io/file/innodb/innodb_log_file
      COUNT_READ: 4
      COUNT_WRITE: 88440
      SUM_NUMBER_OF_BYTES_READ: 132096
      SUM_NUMBER_OF_BYTES_WRITE: 896974848
1 row in set (0.00 sec)
```

mutex_instances,rwlock_instances

- Status for mutexes and RW-Locks
- Contains information about who holds the mutex
 - Often more helpful than who is waiting for it

```
mysql> select * from rwlock_instances where WRITE_LOCKED_BY_THREAD_ID is not null \G
***** 1. row *****
      NAME: wait/synch/rwlock/sql/Query_cache_query::lock
OBJECT_INSTANCE_BEGIN: 139804720739992
WRITE_LOCKED_BY_THREAD_ID: 552
  READ_LOCKED_BY_COUNT: 0
1 row in set (0.02 sec)
```

Benefits

- Great insight in MySQL Operation
- Good Level of Details
 - No need to know server code just architecture
- Information accessible with SQL
 - The interface well known to DBAs
- Clear what is instrumented and what is not
 - **setup_instruments** table
- Actively Developed
 - MySQL 5.6 has number of improvements already

Drawbacks

- Potentially high overhead for some production use
- Not everything is instrumented yet
 - And impossible to tell how much time is not instrumented
- No per statement aggregation
- No file IO latencies and operations
 - Are reads ? Writes ? Or fsyncs ? Slow
- No logging
 - Or simple way to stream data from history table to disk
- Limited documentation
 - What does each of the instruments mean ?

Percona Live, May 26, New York



PERCONA
LIVE

www.percona.com/live

The End

- Thank you !
- Send your feedback to pz@percona.com
- Percona Does MySQL Support, Consulting, Training
- We're creators of

