



P E R C O N A

www.percona.com

Percona Server Documentation

Release 5.6.21-69.0

Percona LLC and/or its affiliates 2009-2014

October 07, 2014

1	Introduction	3
1.1	The <i>Percona XtraDB</i> Storage Engine	3
1.2	List of features available in <i>Percona Server</i> releases	3
1.3	<i>Percona Server</i> Feature Comparison	5
1.4	Changed in <i>Percona Server 5.6</i>	7
2	Installation	13
2.1	Installing <i>Percona Server 5.6</i> from Binaries	13
2.2	Installing <i>Percona Server</i> from a Source Tarball	17
2.3	Installing <i>Percona Server</i> from the Bazaar Source Tree	17
2.4	Compiling <i>Percona Server</i> from Source	18
2.5	Building <i>Percona Server</i> Debian/Ubuntu packages	18
2.6	<i>Percona Server</i> In-Place Upgrading Guide: From 5.5 to 5.6	19
3	Scalability Improvements	25
3.1	Improved Buffer Pool Scalability	25
3.2	Improved <i>InnoDB</i> I/O Scalability	26
3.3	Multiple Adaptive Hash Search Partitions	29
3.4	Multiple user level locks per connection	30
4	Performance Improvements	31
4.1	Atomic write support for Fusion-io devices	31
4.2	Query Cache Enhancements	32
4.3	Improved NUMA support	33
4.4	<i>HandlerSocket</i>	35
4.5	Fixed Size for the Read Ahead Area	36
4.6	Thread Pool	37
4.7	Page cleaner thread tuning	42
4.8	<i>XtraDB</i> Performance Improvements for I/O-Bound Highly-Concurrent Workloads	44
5	Flexibility Improvements	51
5.1	Suppress Warning Messages	51
5.2	Improved MEMORY Storage Engine	52
5.3	Restricting the number of binlog files	56
5.4	Ignoring missing tables in <code>mysqldump</code>	57
5.5	Extended <code>SELECT INTO OUTFILE/DUMPFILE</code>	57
5.6	Per-query variable statement	57
5.7	Extended <code>mysqlbinlog</code>	58
5.8	Slow Query Log Rotation and Expiration	59
5.9	Ability to change database for <code>mysqlbinlog</code>	60

6	Reliability Improvements	63
6.1	Too Many Connections Warning	63
6.2	Error Code Compatibility	63
6.3	Handle Corrupted Tables	64
6.4	Lock-Free SHOW SLAVE STATUS	65
7	Management Improvements	67
7.1	<i>Percona Toolkit</i> UDFs	67
7.2	Support for Fake Changes	68
7.3	Kill Idle Transactions	69
7.4	Enforcing Storage Engine	70
7.5	Utility user	71
7.6	Extending the <code>secure-file-priv</code> server option	73
7.7	Expanded Program Option Modifiers	74
7.8	XtraDB changed page tracking	76
7.9	PAM Authentication Plugin	78
7.10	Expanded Fast Index Creation	80
7.11	Log Archiving for XtraDB	82
7.12	Statement Timeout	84
7.13	Backup Locks	85
7.14	Audit Log Plugin	88
7.15	Start transaction with consistent snapshot	95
8	Diagnostics Improvements	97
8.1	User Statistics	97
8.2	Slow Query Log	103
8.3	Extended Show Engine <i>InnoDB</i> Status	109
8.4	Count <i>InnoDB</i> Deadlocks	119
8.5	Log All Client Commands (<code>syslog</code>)	120
8.6	Show Storage Engines	121
8.7	Show Lock Names	122
8.8	Process List	122
8.9	Misc. INFORMATION_SCHEMA Tables	123
8.10	Thread Based Profiling	125
8.11	Metrics for scalability measurement	125
8.12	Response Time Distribution	127
9	TokuDB	133
9.1	TokuDB Introduction	133
9.2	TokuDB Installation	133
9.3	Using TokuDB	136
9.4	TokuDB Compression	136
9.5	Multiple Clustering Keys	137
10	Reference	139
10.1	List of upstream <i>MySQL</i> bugs fixed in <i>Percona Server 5.6</i>	139
10.2	List of variables introduced in <i>Percona Server 5.6</i>	151
10.3	Development of <i>Percona Server</i>	154
10.4	Trademark Policy	158
10.5	Index of INFORMATION_SCHEMA Tables	159
10.6	Frequently Asked Questions	160
10.7	Copyright and Licensing Information	160
10.8	<i>Percona Server 5.6</i> Release notes	161
10.9	Glossary	177

Percona Server is an enhanced drop-in replacement for *MySQL*. With *Percona Server*,

- Your queries will run faster and more consistently.
- You will consolidate servers on powerful hardware.
- You will delay sharding, or avoid it entirely.
- You will save money on hosting fees and power.
- You will spend less time tuning and administering.
- You will achieve higher uptime.
- You will troubleshoot without guesswork.

Does this sound too good to be true? It's not. *Percona Server* offers breakthrough performance, scalability, features, and instrumentation. Its self-tuning algorithms and support for extremely high-performance hardware make it the clear choice for companies who demand the utmost performance and reliability from their database server.

INTRODUCTION

1.1 The *Percona XtraDB* Storage Engine



Percona XtraDB is an enhanced version of the *InnoDB* storage engine, designed to better scale on modern hardware, and including a variety of other features useful in high performance environments. It is fully backwards compatible, and so can be used as a drop-in replacement for standard *InnoDB*.

Percona XtraDB includes all of *InnoDB* 's robust, reliable ACID-compliant design and advanced MVCC architecture, and builds on that solid foundation with more features, more tunability, more metrics, and more scalability. In particular, it is designed to scale better on many cores, to use memory more efficiently, and to be more convenient and useful. The new features are especially designed to alleviate some of *InnoDB* 's limitations. We choose features and fixes based on customer requests and on our best judgment of real-world needs as a high-performance consulting company.

Percona XtraDB engine will not have further binary releases, it is distributed as part of *Percona Server* and *MariaDB*.

1.2 List of features available in *Percona Server* releases

<i>Percona Server 5.1</i>	<i>Percona Server 5.5</i>	<i>Percona Server 5.6</i>
<i>Improved Buffer Pool Scalability</i>	<i>Improved Buffer Pool Scalability</i>	<i>Improved Buffer Pool Scalability</i>
<i>Configurable Insert Buffer</i>	<i>Configurable Insert Buffer</i>	Feature not implemented
<i>Improved InnoDB I/O Scalability</i>	<i>Improved InnoDB I/O Scalability</i>	<i>Improved InnoDB I/O Scalability</i>
<i>More Concurrent Transactions Available</i>	<i>More Concurrent Transactions Available</i>	Replaced by the upstream implementation
Feature not implemented	<i>Multiple Adaptive Hash Search Partitions</i>	<i>Multiple Adaptive Hash Search Partitions</i>
<i>Dedicated Purge Thread</i>	Replaced by the upstream implementation ²	Replaced by the upstream implementation
<i>Drop table performance</i>	<i>Drop table performance</i>	Replaced by the upstream fix ³
Feature not implemented	<i>Atomic write support for Fusion-io devices</i>	<i>Atomic write support for Fusion-io devices</i>
<i>Configuration of the Doublewrite Buffer</i>	<i>Configuration of the Doublewrite Buffer</i>	Feature not implemented
<i>Query Cache Enhancements</i>	<i>Query Cache Enhancements</i>	<i>Query Cache Enhancements</i>

¹ Feature has been deprecated after *Percona Server 5.5.11-20.2*. It has replaced by the upstream implementation of `innodb_undo_logs` in *MySQL 5.6.3*.

² Feature has not been ported from *Percona Server 5.1* version. It has been replaced by the upstream *Improved Purge Scheduling* implementation.

³ Feature has been removed and its controlling variable `innodb_lazy_drop_table` has been deprecated from *Percona Server 5.5.30-30.2*. Feature has been removed because the upstream `DROP TABLE` implementation has been improved when bugs #56332 and #51325 were fixed.

Table 1.1 – continued from previous page

<i>Percona Server 5.1</i>	<i>Percona Server 5.5</i>	<i>Percona Server 5.6</i>
<i>Fast InnoDB Checksum</i> ⁴	<i>Fast InnoDB Checksum</i> ⁴	Replaced by the upstream implementation
<i>Reduced Buffer Pool Mutex Contention</i>	<i>Reduced Buffer Pool Mutex Contention</i>	Feature not implemented
<i>InnoDB timer-based Concurrency Throttling</i>	<i>InnoDB timer-based Concurrency Throttling</i>	Replaced by the upstream implementation
<i>HandlerSocket</i>	<i>HandlerSocket</i>	Feature not implemented ⁶
Feature not implemented	<i>Improved NUMA support</i>	<i>Improved NUMA support</i>
Feature not implemented	<i>Thread Pool</i>	<i>Thread Pool</i>
Feature not implemented	<i>Binary Log Group Commit</i>	Replaced by the upstream implementation
<i>Support of Multiple Page Sizes</i> ⁸	<i>Support of Multiple Page Sizes</i> ⁸	Replaced by the upstream implementation
<i>Suppress Warning Messages</i>	<i>Suppress Warning Messages</i>	<i>Suppress Warning Messages</i>
<i>Handle BLOB End of Line</i>	<i>Handle BLOB End of Line</i>	Replaced by the upstream implementation
<i>Ability to change database for mysqlbinlog</i>	<i>Ability to change database for mysqlbinlog</i>	<i>Ability to change database for mysqlbinlog</i>
<i>Replication Stop Recovery</i>	Feature not implemented	Feature not implemented
<i>Fixed Size for the Read Ahead Area</i>	<i>Fixed Size for the Read Ahead Area</i>	<i>Fixed Size for the Read Ahead Area</i>
<i>Fast Shutdown</i>	Feature not implemented	Feature not implemented
Feature not implemented	<i>Improved MEMORY Storage Engine</i>	<i>Improved MEMORY Storage Engine</i>
Feature not implemented	<i>Restricting the number of binlog files</i>	<i>Restricting the number of binlog files</i>
<i>Ignoring missing tables in mysqldump</i>	<i>Ignoring missing tables in mysqldump</i>	<i>Ignoring missing tables in mysqldump</i>
<i>Too Many Connections Warning</i>	<i>Too Many Connections Warning</i>	<i>Too Many Connections Warning</i>
<i>Error Code Compatibility</i>	<i>Error Code Compatibility</i>	<i>Error Code Compatibility</i>
<i>Handle Corrupted Tables</i>	<i>Handle Corrupted Tables</i>	<i>Handle Corrupted Tables</i>
<i>Crash-Resistant Replication</i>	<i>Crash-Resistant Replication</i>	Replaced by the upstream implementation
<i>Lock-Free SHOW SLAVE STATUS</i>	<i>Lock-Free SHOW SLAVE STATUS</i>	<i>Lock-Free SHOW SLAVE STATUS</i>
<i>Fast InnoDB Recovery Process</i>	<i>Fast InnoDB Recovery Stats</i>	Feature not implemented
<i>InnoDB Data Dictionary Size Limit</i>	<i>InnoDB Data Dictionary Size Limit</i>	Replaced by the upstream implementation
<i>Expand Table Import</i>	<i>Expand Table Import</i>	Replaced by the upstream implementation
<i>Dump/Restore of the Buffer Pool</i>	<i>Dump/Restore of the Buffer Pool</i>	Replaced by the upstream implementation
<i>Fast Index Creation</i>	<i>Fast Index Creation</i>	Replaced by the upstream implementation
<i>Expanded Fast Index Creation</i>	<i>Expanded Fast Index Creation</i>	<i>Expanded Fast Index Creation</i>
<i>Prevent Caching to FlashCache</i>	<i>Prevent Caching to FlashCache</i>	Feature not implemented
<i>Percona Toolkit UDFs</i>	<i>Percona Toolkit UDFs</i>	<i>Percona Toolkit UDFs</i>
<i>Support for Fake Changes</i>	<i>Support for Fake Changes</i>	<i>Support for Fake Changes</i>
<i>Kill Idle Transactions</i>	<i>Kill Idle Transactions</i>	<i>Kill Idle Transactions</i>
<i>XtraDB changed page tracking</i>	<i>XtraDB changed page tracking</i>	<i>XtraDB changed page tracking</i>
Feature not implemented	<i>Enforcing Storage Engine</i>	<i>Enforcing Storage Engine</i>
Feature not implemented	<i>Utility user</i>	<i>Utility user</i>
Feature not implemented	<i>Extending the secure-file-priv server option</i>	<i>Extending the secure-file-priv server option</i>
Feature not implemented	<i>Expanded Program Option Modifiers</i>	<i>Expanded Program Option Modifiers</i>
Feature not implemented	<i>PAM Authentication Plugin</i>	<i>PAM Authentication Plugin</i>

⁴ Feature has been deprecated after *Percona Server 5.1.66-14.2* and *Percona Server 5.5.28-29.2*. It has been replaced by the upstream `innodb_checksum_algorithm` implementation released in *MySQL 5.6.3*.

⁵ Feature has been replaced by the upstream implementation `innodb-performance-thread_concurrency` in *MySQL 5.6*

⁶ Feature will be implemented in one of the future *Percona Server 5.6* releases.

⁷ Binary Log Group Commit feature has been replaced with the *MySQL 5.6* implementation.

⁸ Feature has been deprecated in the *Percona Server 5.1.68-14.6* and *Percona Server 5.5.30-30.2*. It has been replaced by the upstream `innodb_page_size` version released in *MySQL 5.6.4*.

⁹ Feature has been replaced by the *MySQL 5.6* binary-mode configuration option.

¹⁰ Feature has been replaced by the *MySQL 5.6* `relay-log-recovery` implementation.

¹¹ Feature has been replaced by the *MySQL 5.6* `table_definition_cache` implementation.

¹² Feature has been replaced by the *MySQL 5.6* Improved Tablespace Management implementation.

¹³ Feature has been replaced by the *MySQL 5.6* preloading the InnoDB buffer pool at startup implementation.

¹⁴ Feature has been replaced by the 5.6' `ALGORITHM=` option implementation.

Table 1.1 – continued from previous page

<i>Percona Server 5.1</i>	<i>Percona Server 5.5</i>	<i>Percona Server 5.6</i>
Feature not implemented	Feature not implemented	<i>Log Archiving for XtraDB</i>
<i>InnoDB Statistics</i>	<i>InnoDB Statistics</i>	Replaced by the upstream implement
<i>User Statistics</i>	<i>User Statistics</i>	<i>User Statistics</i>
<i>Slow Query Log</i>	<i>Slow Query Log</i>	<i>Slow Query Log</i>
<i>Count InnoDB Deadlocks</i>	<i>Count InnoDB Deadlocks</i>	<i>Count InnoDB Deadlocks</i>
<i>Log All Client Commands (syslog)</i>	<i>Log All Client Commands (syslog)</i>	<i>Log All Client Commands (syslog)</i>
<i>Response Time Distribution</i>	<i>Response Time Distribution</i>	<i>Response Time Distribution</i>
<i>Show Storage Engines</i>	<i>Show Storage Engines</i>	<i>Show Storage Engines</i>
<i>Show Lock Names</i>	<i>Show Lock Names</i>	<i>Show Lock Names</i>
<i>Process List</i>	<i>Process List</i>	<i>Process List</i>
Misc. INFORMATION_SCHEMA Tables	<i>Misc. INFORMATION_SCHEMA Tables</i>	<i>Misc. INFORMATION_SCHEMA</i>
Feature not implemented	<i>Extended Show Engine InnoDB Status</i>	<i>Extended Show Engine InnoDB St</i>
Feature not implemented	<i>Thread Based Profiling</i>	<i>Thread Based Profiling</i>
Feature not implemented	Feature not implemented	<i>XtraDB Performance Improvemen</i>
Feature not implemented	Feature not implemented	<i>Page cleaner thread tuning</i>
Feature not implemented	Feature not implemented	<i>Statement Timeout</i>
Feature not implemented	<i>Extended SELECT INTO OUTFILE/DUMPFIL</i>	<i>Extended SELECT INTO OUTFIL</i>
Feature not implemented	Feature not implemented	<i>Per-query variable statement</i>
Feature not implemented	<i>Extended mysqlbinlog</i>	<i>Extended mysqlbinlog</i>
Feature not implemented	<i>Slow Query Log Rotation and Expiration</i>	<i>Slow Query Log Rotation and Exp</i>
Feature not implemented	<i>Metrics for scalability measurement</i>	<i>Metrics for scalability measureme</i>
Feature not implemented	<i>Audit Log</i>	<i>Audit Log</i>
Feature not implemented	Feature not implemented	<i>Backup Locks</i>

1.2.1 Other Reading

- *Changed in Percona Server 5.6*
- *Percona Server In-Place Upgrading Guide: From 5.5 to 5.6*
- *Percona Sever In-Place Upgrading Guide: From 5.5 to 5.6*
- *Upgrading from MySQL 5.1 to 5.5*
- *What Is New in MySQL 5.5*
- *What Is New in MySQL 5.6*

1.2.2 Footnotes

1.3 Percona Server Feature Comparison

Percona Server is an enhanced drop-in replacement for *MySQL*. With *Percona Server*,

- Your queries will run faster and more consistently.
- You will consolidate servers on powerful hardware.
- You will delay sharding, or avoid it entirely.
- You will save money on hosting fees and power.

¹⁵ Feature has been replaced by the *MySQL* 5.6 Persistent Optimizer Statistics for InnoDB Tables implementation.

- You will spend less time tuning and administering.
- You will achieve higher uptime.
- You will troubleshoot without guesswork.

We provide these benefits by significantly enhancing *Percona Server* as compared to the standard *MySQL* database server:

Features	Percona Server 5.6.13	MySQL 5.6.13
Open source	Yes	Yes
ACID Compliance	Yes	Yes
Multi-Version Concurrency Control	Yes	Yes
Row-Level Locking	Yes	Yes
Automatic Crash Recovery	Yes	Yes
Table Partitioning	Yes	Yes
Views	Yes	Yes
Subqueries	Yes	Yes
Triggers	Yes	Yes
Stored Procedures	Yes	Yes
Foreign Keys	Yes	Yes
GTID Replication	Yes	Yes

Extra Features for Developers	Percona Server 5.6.13	MySQL 5.6.13
NoSQL Socket-Level Interface	Yes	Yes
Extra Hash/Digest Functions	Yes	

Extra Diagnostic Features	Percona Server 5.6.13	MySQL 5.6.13
INFORMATION_SCHEMA Tables	70	59
Global Performance and Status Counters	398	341
Per-Table Performance Counters	Yes	
Per-Index Performance Counters	Yes	
Per-User Performance Counters	Yes	
Per-Client Performance Counters	Yes	
Per-Thread Performance Counters	Yes	
Global Query Response Time Statistics	Yes	
InnoDB Data Dictionary as I_S Tables	Yes	
Access to InnoDB Data Statistics	Yes	
Enhanced SHOW ENGINE INNODB STATUS	Yes	
Enhanced Mutex Diagnostics	Yes	
Undo Segment Information	Yes	

Performance & Scalability Enhancements	Percona Server 5.6.13	MySQL 5.6.13
Fine-Grained Mutex Locking	Yes	
Lock-Free Algorithms	Yes	
Improved MEMORY Storage Engine	Yes	
Partitioned Adaptive Hash Search	Yes	
Support for FlashCache	Yes	
Read-Ahead Improvements	Yes	

Extra Features for DBA/Operations Staff	Percona Server 5.6.13	MySQL 5.6.13
Configurable Page Sizes	Yes	Yes
Configurable Insert Buffer Size	Yes	
Error/Warning Logging Enhancements	Yes	
Configurable Fast Index Creation	Yes	
Support for Fake Changes	Yes	
Changed Page Tracking	Yes	
PAM Authentication	Yes	Yes ¹⁷
Threadpool	Yes	Yes ¹
Log Archiving	Yes	

Running Database as a Service	Percona Server 5.6.13	MySQL 5.6.13
Special Utility User	Yes	
Expanded Program Option Modifiers	Yes	
Extended <code>secure-file-priv</code> option	Yes	
Enforcing the Specific Storage Engine	Yes	

1.4 Changed in Percona Server 5.6

Percona Server 5.6 is based on *MySQL 5.6* and incorporates many of the improvements found in *Percona Server 5.5*.

1.4.1 Features removed from *Percona Server 5.6* that were available in *Percona Server 5.5*

Some features that were present in *Percona Server 5.5* have been removed in *Percona Server 5.6*. These are:

- `SHOW [GLOBAL] TEMPORARY TABLES` functionality is now only available via the `INFORMATION_SCHEMA` tables `TEMPORARY_TABLES` and `GLOBAL_TEMPORARY_TABLES`.
- InnoDB timer-based Concurrency Throttling
- InnoDB Recovery Stats
- `Rows_read` counters in *Slow Query Log* and `SHOW PROCESSLIST` had a very fuzzy meaning so they were removed.

1.4.2 Replaced features that were present in *Percona Server 5.5*

Some features that were present in *Percona Server 5.5* have been replaced by a different implementation of the same/similar functionality in *Percona Server 5.6*. These are:

- `SHOW ENGINE INNODB STATUS` section “OLDEST VIEW” has been replaced by the `XTRADB_READ_VIEW` `INFORMATION_SCHEMA` table.
- `SHOW ENGINE INNODB STATUS` sections on memory usage for InnoDB/XtraDB hash tables has been replaced by the `XTRADB_INTERNAL_HASH_TABLES` `INFORMATION_SCHEMA` table.
- The `INNODB_RSEG` table has been renamed to `XTRADB_RSEG`.
- *Fixed Size for the Read Ahead Area* has been implemented differently. Buffer read-ahead area size is now precalculated once per buffer pool instance initialization instead of hardcoding it at 64MB (like it was done in previous *Percona Server* versions).

¹⁶Feature available in Enterprise version only

¹⁷Feature available in Enterprise version only

- *Response Time Distribution* feature has been implemented as a plugin. It has the following changes from the 5.5 implementation: - the plugin requires installation before the feature can be used; - variable `have_response_time_distribution` has been removed. The presence of the feature can be determined by querying `SHOW PLUGINS` instead; - replication updates performed by the slave SQL threads are not tracked; - command `SHOW QUERY_RESPONSE_TIME`; has been removed in favor of `QUERY_RESPONSE_TIME` table; - command `FLUSH QUERY_RESPONSE_TIME`; has been replaced with `query_response_time_flush` variable.

1.4.3 Features available in *Percona Server 5.5* that have been replaced with *MySQL 5.6* features

Some *Percona Server 5.5* features have been replaced by similar or equivalent *MySQL 5.6* features, so we now keep the *MySQL 5.6* implementations in *Percona Server 5.6*. These are:

- *Crash-Resistant Replication* has been replaced by *MySQL* crash safe replication
- *Improved InnoDB I/O Scalability* patches have been replaced by improvements and changes in *MySQL 5.6*, although *Percona* may make improvements in the future
- *InnoDB Data Dictionary Size Limit* has been replaced by *MySQL 5.6* using the existing `table-definition-cache` variable to limit the size of the *InnoDB* data dictionary.
- *Expand Table Import* has been replaced by *MySQL* “*InnoDB* transportable tablespaces”
- The *InnoDB* data dictionary `INFORMATION_SCHEMA` tables have been superseded by the *MySQL* implementations
- *XtraDB* `SYS_STATS` persistent table and index statistics has been replaced by the *MySQL 5.6* implementation
- *Dump/Restore of the Buffer Pool* is now available in *MySQL 5.6*, so we have replaced the *Percona Server* implementation with the *MySQL* one. The upstream implementation doesn’t have the periodic dump feature, but it’s possible to set it up by using the *event scheduler* and the new `innodb_buffer_pool_dump_now` variable. The following example shows how to implement a periodic buffer pool dump every hour:

```
mysql> CREATE EVENT automatic_bufferpool_dump
ON SCHEDULE EVERY 1 HOUR
DO
SET global innodb_buffer_pool_dump_now=ON;
```

- *fast_index_creation* (replaced by *MySQL 5.6*’s `ALGORITHM=` option).
- *Fast InnoDB Checksum* has been deprecated after *Percona Server 5.5.28-29.2* because the `innodb_checksum_algorithm` variable in *MySQL 5.6* makes it redundant. If this feature was enabled, turning it off before the upgrade requires table(s) to be dump and imported, since it will fail to start on data files created when `innodb_fast_checksums` was enabled.
- *Handle BLOB End of Line* feature has been replaced by *MySQL 5.6* `binary-mode` configuration option.
- *Percona Server 5.5* implemented `utf8_general50_ci` and `ucs2_general50_ci` collations as a fix for the upstream bug: [#27877](#). These are now being replaced by *MySQL 5.6* `utf8_general_mysql500_ci` and `ucs2_general_mysql500_ci` collations.
- *Percona Server* `INFORMATION_SCHEMA` `_STATS` tables in 5.5 have been replaced by new tables in *MySQL 5.6*: `INNODB_SYS_TABLES`, `INNODB_SYS_INDEXES`, `INNODB_SYS_COLUMNS`, `INNODB_SYS_FIELDS`, `INNODB_SYS_FOREIGN`, `INNODB_SYS_FOREIGN_COLS`, `INNODB_SYS_TABLESTATS` (although *MySQL 5.6* does not have `MYSQL_HANDLES_OPENED`, instead it has `REF_COUNT`). Following tables haven’t been implemented in *MySQL 5.6* but information is

available in other tables: INNODB_SYS_STATS - use MYSQL.INNODB_(INDEX|TABLE)_STATS instead, INNODB_TABLE_STATS - use INNODB_SYS_TABLESTATS or MYSQL.INNODB_TABLE_STATS instead, and INNODB_INDEX_STATS - use MYSQL.INNODB_INDEX_STATS instead.

1.4.4 Features ported from *Percona Server 5.5* to *Percona Server 5.6*

Following features were ported from *Percona Server 5.5* to *Percona Server 5.6*:

Feature Ported	Version
<i>Thread Pool</i>	5.6.10-60.2
<i>Atomic write support for Fusion-io devices</i>	5.6.11-60.3
<i>Improved InnoDB I/O Scalability</i>	5.6.11-60.3
<i>Improved NUMA support</i>	5.6.11-60.3
<i>Suppress Warning Messages</i>	5.6.11-60.3
<i>Improved MEMORY Storage Engine</i>	5.6.11-60.3
<i>Restricting the number of binlog files</i>	5.6.11-60.3
<i>Too Many Connections Warning</i>	5.6.11-60.3
<i>Error Code Compatibility</i>	5.6.11-60.3
<i>Lock-Free SHOW SLAVE STATUS</i>	5.6.11-60.3
<i>Percona Toolkit UDFs</i>	5.6.11-60.3
<i>Support for Fake Changes</i>	5.6.11-60.3
<i>Kill Idle Transactions</i>	5.6.11-60.3
<i>Enforcing Storage Engine</i>	5.6.11-60.3
<i>Utility user</i>	5.6.11-60.3
<i>Extending the secure-file-priv server option</i>	5.6.11-60.3
<i>Expanded Program Option Modifiers</i>	5.6.11-60.3
<i>XtraDB changed page tracking</i>	5.6.11-60.3
<i>PAM Authentication Plugin</i>	5.6.11-60.3
<i>User Statistics</i>	5.6.11-60.3
<i>Slow Query Log</i>	5.6.11-60.3
<i>Extended Show Engine InnoDB Status</i>	5.6.11-60.3
<i>Count InnoDB Deadlocks</i>	5.6.11-60.3
<i>Log All Client Commands (syslog)</i>	5.6.11-60.3
<i>Show Storage Engines</i>	5.6.11-60.3
<i>Thread Based Profiling</i>	5.6.11-60.3
<i>Fixed Size for the Read Ahead Area</i>	5.6.13-60.5
<i>Improved Buffer Pool Scalability</i>	5.6.13-60.6
<i>Multiple Adaptive Hash Search Partitions</i>	5.6.13-60.6
<i>HandlerSocket</i>	5.6.17-66.0
<i>Response Time Distribution</i>	5.6.21-69.0

1.4.5 List of status variables that are no longer available in *Percona Server 5.6*

Following status variables available in *Percona Server 5.5* are no longer present in *Percona Server 5.6*:

Status Variables	Replaced by
Com_show_temporary_tables	This variable has been removed together with the “SHOW [GLOBAL] TEMPORARY TABLES” statement, whose call number it was counting. The information about temporary tables is available via the INFORMATION_SCHEMA tables <code>TEMPORARY_TABLES</code> and <code>GLOBAL_TEMPORARY_TABLES</code>
Flashcache_enabled	information if the Flashcache support has been enabled has not been ported to <i>Percona Server 5.6</i>
Innodb_adaptive_hash_index	this variable has not been ported to <i>Percona Server 5.6</i>
Innodb_adaptive_hash_index	this variable has not been ported to <i>Percona Server 5.6</i>
Innodb_adaptive_hash_index	replaced by adaptive_hash_searches counter in INFORMATION_SCHEMA.INNODB_METRICS table
Innodb_adaptive_hash_index	replaced by adaptive_hash_searches_btree counter in INFORMATION_SCHEMA.INNODB_METRICS table
Innodb_checkpoint_target_size	replaced by MySQL 5.6 flushing implementation
Innodb_dict_tables	<i>InnoDB Data Dictionary Size Limit</i> feature has been replaced by the new MySQL 5.6 <code>table_definition_cache</code> implementation
Innodb_master_thread_sleeps	new <i>InnoDB</i> master thread behavior makes this variable redundant
Innodb_master_thread_sleeps	new <i>InnoDB</i> master thread behavior makes this variable redundant
Innodb_master_thread_sleeps	new <i>InnoDB</i> master thread behavior makes this variable redundant
Innodb_master_thread_sleeps	new <i>InnoDB</i> master thread behavior makes this variable redundant
Innodb_master_thread_sleeps	replaced by innodb_master_thread_sleeps counter in INFORMATION_SCHEMA.INNODB_METRICS table
binlog_commits	<i>Binary Log Group Commit</i> feature has been replaced with the MySQL 5.6 implementation that doesn't have this status variable.
binlog_group_commits	<i>Binary Log Group Commit</i> feature has been replaced with the MySQL 5.6 implementation that doesn't have this status variable.

1.4.6 List of system variables that are no longer available in *Percona Server 5.6*

Following system variables available in *Percona Server 5.5* are no longer present in *Percona Server 5.6*:

Warning: *Percona Server 5.6* won't be able to start if some of these variables are set in the server's configuration file.

System Variables	Feature Comment
fast_index_creation	replaced by using MySQL's <code>ALGORITHM</code> option
have_flashcache	Information if the server has been compiled with the Flashcache support has not been ported to <i>Percona Server 5.6</i>
have_response_time_distribution	<i>Response Time Distribution</i> feature has been ported to <i>Percona Server 5.6</i>
innodb_adaptive_flushing_method	replaced by MySQL 5.6 flushing implementation
innodb_blocking_buffer_pool_restore	variable doesn't have direct replacement in MySQL 5.6. Feature will be implemented in MySQL 5.7
innodb_buffer_pool_restore_at_startup	replaced by <code>innodb_buffer_pool_load_at_startup</code>
innodb_buffer_pool_shm_checksum	variable has been deprecated and removed in <i>Percona Server 5.5</i>
innodb_buffer_pool_shm_key	variable has been deprecated and removed in <i>Percona Server 5.5</i>
innodb_checkpoint_age_target	replaced by MySQL 5.6 flushing implementation
innodb_dict_size_limit	replaced by MySQL 5.6 new <code>table_definition_cache</code> implementation
innodb_doublewrite_file	<i>Configuration of the Doublewrite Buffer</i> feature containing this variable has not been ported to <i>Percona Server 5.6</i>
innodb_fast_checksum	replaced by <code>innodb_checksum_algorithm</code>
innodb_flush_neighbor_pages	replaced by <code>innodb_flush_neighbors</code>
innodb_ibuf_accel_rate	<i>Configurable Insert Buffer</i> feature containing this variable has not been ported to <i>Percona Server 5.6</i>
innodb_ibuf_active_contract	<i>Configurable Insert Buffer</i> feature containing this variable has not been ported to <i>Percona Server 5.6</i>
innodb_ibuf_max_size	<i>Configurable Insert Buffer</i> feature containing this variable has not been ported to <i>Percona Server 5.6</i>

System Variables	Feature Comment
<code>innodb_import_table_from_xtrabackup</code>	replaced by MySQL transportable tablespaces
<code>innodb_lazy_drop_table</code>	variable has been deprecated and removed in <i>Percona Server 5.5</i>
<code>innodb_merge_sort_block_size</code>	replaced by <code>innodb_sort_buffer_size</code>
<code>innodb_page_size</code>	replaced by <code>innodb_page_size</code>
<code>innodb_read_ahead</code>	replaced by MySQL Read-Ahead Algorithm implementation, <code>innodb_random_read_ahead</code>
<code>innodb_recovery_stats</code>	<i>InnoDB Recovery Stats</i> feature containing this variable has not been ported
<code>innodb_recovery_update_relay_log</code>	replaced by <code>relay-log-recovery</code>
<code>innodb_stats_auto_update</code>	replaced by <code>innodb_stats_auto_recalc</code>
<code>innodb_stats_update_need_lock</code>	variable has not been ported to <i>Percona Server 5.6</i>
<code>innodb_thread_concurrency_timer_based</code>	<i>InnoDB timer-based Concurrency Throttling</i> feature containing this variable
<code>innodb_use_sys_stats_table</code>	variable has been replaced by Persistent Optimizer Statistics implementation
<code>log_slow_admin_statements</code>	the upstream variable has the same functionality
<code>log_slow_slave_statements</code>	the upstream variable has the same functionality
<code>optimizer_fix</code>	this variable has been deprecated and removed in <i>Percona Server 5.5</i>
<code>query_response_time_range_base</code>	<i>Response Time Distribution</i> feature containing this variable has been ported
<code>query_response_time_stats</code>	<i>Response Time Distribution</i> feature containing this variable has been ported

INSTALLATION

2.1 Installing *Percona Server 5.6* from Binaries

Before installing, you might want to read the *Percona Server 5.6 Release notes*.

Ready-to-use binaries are available from the *Percona Server* [download page](#), including:

- RPM packages for *RHEL 5* and *RHEL 6*
- *Debian* packages for *Debian* and *Ubuntu*
- Generic `.tar.gz` packages

2.1.1 Using Percona Software Repositories

Percona apt Repository

Debian and *Ubuntu* packages from *Percona* are signed with a key. Before using the repository, you should add the key to **apt**. To do that, run the following commands as root:

```
$ apt-key adv --keyserver keys.gnupg.net --recv-keys 1C4CBDCDCD2EFD2A
```

Add this to `/etc/apt/sources.list`, replacing `VERSION` with the name of your distribution:

```
deb http://repo.percona.com/apt VERSION main
deb-src http://repo.percona.com/apt VERSION main
```

Remember to update the local cache:

```
$ apt-get update
```

After that you can install the server and client packages

```
$ apt-get install percona-server-server-5.6 percona-server-client-5.6
```

Supported Platforms

- x86
- x86_64 (also known as amd64)

Supported Releases

Debian

- 6.0 (squeeze)
- 7.0 (wheezy)

Ubuntu

- 10.04LTS (lucid)
- 12.04LTS (precise)
- 13.10 (saucy)
- 14.04LTS (trusty)

Percona *apt* Testing repository

Percona offers pre-release builds from the testing repository. To enable it add the following lines to your `/etc/apt/sources.list`, replacing `VERSION` with the name of your distribution:

```
deb http://repo.percona.com/apt VERSION main testing
deb-src http://repo.percona.com/apt VERSION main testing
```

Apt-Pinning the packages

In some cases you might need to “pin” the selected packages to avoid the upgrades from the distribution repositories. You’ll need to make a new file `/etc/apt/preferences.d/00percona.pref` and add the following lines in it:

```
Package: *
Pin: release o=Percona Development Team
Pin-Priority: 1001
```

For more information about the pinning you can check the official [debian wiki](#).

Percona *yum* Repository

The *Percona yum* repository supports popular *RPM*-based operating systems, including the *Amazon Linux AMI*.

The easiest way to install the *Percona Yum* repository is to install an *RPM* that configures **yum** and installs the [Percona GPG key](#).

Automatic Installation

Execute the following command as a root user:

```
$ yum install http://www.percona.com/downloads/percona-release/redhat/0.1-3/percona-release-0.1-3.noarch.rpm
```

You should see some output such as the following:

```
Retrieving http://www.percona.com/downloads/percona-release/redhat/0.1-3/percona-release-0.1-3.noarch.rpm
Preparing... ##### [100%]
 1:percona-release ##### [100%]
```

Testing The Repository

Make sure packages are downloaded from the repository, by executing the following command as root:

```
yum list | grep percona
```

You should see output similar to the following:

```
percona-release.noarch           0.1-3                @/percona-release-0.1-3.noarch
...
Percona-Server-client-56.x86_64  5.6.15-rel63.0.519.rhel6  percona
Percona-Server-devel-56.x86_64  5.6.15-rel63.0.519.rhel6  percona
Percona-Server-server-56.x86_64  5.6.15-rel63.0.519.rhel6  percona
Percona-Server-shared-56.x86_64  5.6.15-rel63.0.519.rhel6  percona
Percona-Server-test-56.x86_64    5.6.15-rel63.0.519.rhel6  percona
...
percona-xtrabackup.x86_64        2.2.4-5004.el6         percona
```

Supported Platforms

- x86_64
- i386

Supported Releases

The *CentOS* repositories should work well with *Red Hat Enterprise Linux* too, provided that **yum** is installed on the server.

- *CentOS 5* and *RHEL 5*
- *CentOS 6* and *RHEL 6* (Current Stable) ¹
- *CentOS 7* and *RHEL 7*
- *Amazon Linux AMI* (works the same as *CentOS 5*)

Percona yum Testing repository

Percona offers pre-release builds from the testing repository. To subscribe to the testing repository, you'll need to enable the testing repository in `/etc/yum.repos.d/percona-release.repo` (both `$basearch` and `noarch`). **NOTE:** You'll need to install the Percona repository first if this hasn't been done already.

Percona provides repositories for **yum** (RPM packages for *Red Hat*, *CentOS* and *Amazon Linux AMI*) and **apt** (`.deb` packages for *Ubuntu* and *Debian*) for software such as *Percona Server*, *XtraDB*, *XtraBackup*, and *Percona Toolkit*.

¹ "Current Stable": We support only the current stable RHEL6/CentOS6 release, because there is no official (i.e. RedHat provided) method to support or download the latest OpenSSL on RHEL/CentOS versions prior to 6.5. Similarly, and also as a result thereof, there is no official Percona way to support the latest Percona Server builds on RHEL/CentOS versions prior to 6.5. Additionally, many users will need to upgrade to OpenSSL 1.0.1g or later (due to the [Heartbleed vulnerability](#)), and this OpenSSL version is not available for download from any official RHEL/CentOS repository for versions 6.4 and prior. For any officially unsupported system, `src.rpm` packages may be used to rebuild Percona Server for any environment. Please contact our [support service](#) if you require further information on this.

This makes it easy to install and update your software and its dependencies through your operating system's package manager.

This is the recommend way of installing where possible.

YUM-Based Systems

Once the repository is set up, use the following commands:

```
$ yum install Percona-Server-client-56 Percona-Server-server-56
```

DEB-Based Systems

Once the repository is set up, use the following commands:

```
$ sudo apt-get install percona-server-server-5.6
```

2.1.2 Using Standalone Packages

RPM-Based Systems

Download the packages of the desired series for your architecture from [here](#).

For example, at the moment of writing, a way of doing this is:

```
$ wget -r -l 1 -nd -A rpm -R "*devel*,*debuginfo*" \
http://www.percona.com/downloads/Percona-Server-5.6/LATEST/RPM/rhel6/x86_64/
```

Install them in one command:

```
$ rpm -ivh Percona-Server-shared-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm \
Percona-Server-client-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm \
Percona-Server-server-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm
```

If you don't install all "at the same time", you will need to do it in a specific order - shared, client, server:

```
$ rpm -ivh Percona-Server-shared-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm
$ rpm -ivh Percona-Server-client-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm
$ rpm -ivh Percona-Server-server-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm
```

Otherwise, the dependencies won't be met and the installation will fail.

What's in each RPM?

Each of the *Percona Server* RPM packages have a particular purpose.

The `Percona-Server-server` package contains the server itself (the `mysqld` binary).

The `Percona-Server-56-debuginfo` package contains debug symbols for use debugging the database server.

The `Percona-Server-client` package contains the command line client.

The `Percona-Server-devel` package contains the header files needed to compile software using the client library.

The `Percona-Server-shared` package includes the client shared library.

The `Percona-Server-shared-compat` package includes shared libraries for software compiled against old versions of the client library.

The `Percona-Server-test` package includes the test suite for *Percona Server*.

DEB-Based Systems

Download the packages of the desired series for your architecture from [here](#).

For example, at the moment of writing, for *Ubuntu* 12.04LTS on `x86_64`, a way of doing this is:

```
$ wget -r -l 1 -nd -A deb -R "*dev*" \  
http://www.percona.com/downloads/Percona-Server-5.6/LATEST/deb/precise/x86_64/
```

Install them in one command:

```
$ sudo dpkg -i *.deb
```

The installation won't succeed as there will be missing dependencies. To handle this, use:

```
$ apt-get -f install
```

All dependencies will be installed and the *Percona Server* installation will be finished by **apt**.

What's in each DEB package?

The `percona-server-server` package contains the database server itself, the `mysqld` binary and associated files.

The `percona-server-common` package contains files common to the server and client.

The `percona-server-client` package contains the command line client.

The `percona-server-dfsg` package contains...

The `libperconaserverclient-dev` package contains header files needed to compile software to use the client library.

The `libperconaserverclient18` package contains the client shared library. The 18 is a reference to the version of the shared library. The version is incremented when there is a ABI change that requires software using the client library to be recompiled or their source code modified.

2.2 Installing *Percona Server* from a Source Tarball

Fetch and extract the source tarball. For example:

```
$ wget http://www.percona.com/edir/downloads/Percona-Server-5.6/LATEST/source/Percona-Server-5.6.6-alpha60.1.tar.gz  
$ tar xzf Percona-Server-5.6.6-alpha60.1.tar.gz
```

Next, follow the instructions in *Compiling Percona Server from Source* below.

2.3 Installing *Percona Server* from the Bazaar Source Tree

Percona uses the [Bazaar](#) revision control system for development. To build the latest Percona Server from the source tree you will need Bazaar installed on your system.

Good practice is to use a shared repository, create one like this:

```
$ bzip init-repo ~/percona-server
```

You can now fetch the latest *Percona Server* 5.6 sources. In the future, we will provide instructions for fetching each specific *Percona Server* version and building it, but currently we will just talk about building the latest *Percona Server* 5.6 development tree.

```
$ cd ~/percona-server
$ bzip branch lp:percona-server/5.6
```

Fetching all the history of *Percona Server* 5.6 may take a long time, up to 20 or 30 minutes is not uncommon.

If you are going to be making changes to *Percona Server* 5.6 and wanting to distribute the resulting work, you can generate a new source tarball (exactly the same way as we do for release):

```
$ cmake .
$ make dist
```

Next, follow the instructions in *Compiling Percona Server from Source* below.

2.4 Compiling *Percona Server* from Source

After either fetching the source repository or extracting a source tarball (from *Percona* or one you generated yourself), you will now need to configure and build *Percona Server*.

First, run `cmake` to configure the build. Here you can specify all the normal build options as you do for a normal *MySQL* build. Depending on what options you wish to compile *Percona Server* with, you may need other libraries installed on your system. Here is an example using a configure line similar to the options that *Percona* uses to produce binaries:

```
$ cmake . -DCMAKE_BUILD_TYPE=RelWithDebInfo -DBUILD_CONFIG=mysql_release -DFEATURE_SET=community -DW
```

Now, compile using `make`

```
$ make
```

Install:

```
$ make install
```

Percona Server 5.6 will now be installed on your system.

2.5 Building *Percona Server* Debian/Ubuntu packages

If you wish to build your own *Percona Server* Debian/Ubuntu (dpkg) packages, you first need to start with a source tarball, either from the *Percona* website or by generating your own by following the instructions above (*Installing Percona Server from the Bazaar Source Tree*).

Extract the source tarball:

```
$ tar xzf percona-server-5.6.15-62.0.tar.gz
$ cd percona-server-5.6.15-62.0
```

Put the debian packaging in the directory that Debian expects it to be in:


```
$ cp -ap build-ps/debian debian
```

Update the changelog for your distribution (here we update for the unstable distribution - sid), setting the version number appropriately. The trailing one in the version number is the revision of the Debian packaging.

```
$ dch -D unstable --force-distribution -v "5.6.15-62.0-1" "Update to 5.6.15-62.0"
```

Build the Debian source package:

```
$ dpkg-buildpackage -S
```

Use sbuild to build the binary package in a chroot:

```
$ sbuild -d sid percona-server-5.6_5.6.15_62.0-1.dsc
```

You can give different distribution options to dch and sbuild to build binary packages for all Debian and Ubuntu releases.

Note: *PAM Authentication Plugin* is not built with the server by default. In order to build the Percona Server with PAM plugin, additional option `-DWITH_PAM=ON` should be used.

2.6 Percona Server In-Place Upgrading Guide: From 5.5 to 5.6

In-place upgrades are those which are done using the existing data in the server. Generally speaking, this is stopping the server, installing the new server and starting it with the same data files. While they may not be suitable for high-complexity environments, they may be adequate for many scenarios.

The following is a summary of the more relevant changes in the 5.6 series. For more details, see

- *Percona Server documentation*
- *Changed in Percona Server 5.6*
- *Upgrading from MySQL 5.5 to 5.6*

Warning: Upgrade from 5.5 to 5.6 on a crashed instance is not recommended. If the server instance has crashed, crash recovery should be run before proceeding with the upgrade.

Note: Upgrading the from older *Percona Server* version that doesn't have default (16k) *InnoDB* page size is not recommended. This could happen if the variable `innodb_page_size` was set to non-default value.

2.6.1 Upgrading using the Percona repositories

The easiest and recommended way of installing - where possible - is by using the *Percona* repositories.

Instructions for enabling the repositories in a system can be found in:

- *Percona APT Repository*
- *Percona YUM Repository*

DEB-based distributions

Having done the full backup (or dump if possible), stop the server:

```
$ sudo /etc/init.d/mysqld stop
```

and proceed to do the modifications needed in your configuration file, as explained at the beginning of this guide.

Then install the new server with:

```
$ sudo apt-get install percona-server-server-5.6
```

The installation script will run automatically **mysql_upgrade** to migrate to the new grant tables, rebuild the indexes where needed and then start the server.

Note that this procedure is the same for upgrading from *MySQL 5.5* or *5.6* to *Percona Server 5.6*.

RPM-based distributions

Having done the full backup (and dump if possible), stop the server:

```
$ /sbin/service mysql stop
```

and check your installed packages with:

```
$ rpm -qa | grep Percona-Server
Percona-Server-client-55-5.5.29-rel29.4.401.rhel6.x86_64.rpm
Percona-Server-server-55-5.5.29-rel29.4.401.rhel6.x86_64.rpm
Percona-Server-shared-55-5.5.29-rel29.4.401.rhel6.x86_64.rpm
```

You may have a forth, `shared-compat`, which is for compatibility purposes.

After checking, proceed to remove them without dependencies:

```
$ rpm -qa | grep Percona-Server | xargs rpm -e --nodeps
```

It is important that you remove it without dependencies as many packages may depend on these (as they replace `mysql`) and will be removed if omitted.

Note that this procedure is the same for upgrading from *MySQL 5.5* or *5.6* to *Percona Server 5.6*: just `grep '^mysql-'` instead of `Percona-Server` and remove them.

You will have to install the following packages:

- `Percona-Server-server-56`
- `Percona-Server-client-56`

```
$ yum install Percona-Server-server-56 Percona-Server-client-56
```

Once installed, proceed to modify your configuration file - `my.cnf` - and recompile the plugins if necessary, as explained at the beginning of this guide.

As the schema of the grant table has changed, the server must be started without reading them:

```
$ /usr/sbin/mysqld --skip-grant-tables --user=mysql &
```

and use `mysql_upgrade` to migrate to the new grant tables, it will rebuild the indexes needed and do the modifications needed:

```
$ mysql_upgrade
...
OK
```

Once this is done, just restart the server as usual:

```
$ /sbin/service mysql restart
```

If it can't find the PID file, kill the server and start it normally:

```
$ killall /usr/sbin/mysqld
$ /sbin/service mysql start
```

2.6.2 Upgrading using Standalone Packages

DEB-based distributions

Having done the full backup (and dump if possible), stop the server:

```
$ sudo /etc/init.d/mysqld stop
```

and remove the the installed packages with their dependencies:

```
$ sudo apt-get autoremove percona-server-server-55 percona-server-client-55
```

Once removed, proceed to do the modifications needed in your configuration file, as explained at the beginning of this guide.

Then, download the following packages for your architecture:

- percona-server-server-5.6
- percona-server-client-5.6
- percona-server-common-5.6
- libperconaserverclient18

At the moment of writing this guide, for *Ubuntu* 12.04LTS on x86_64, a way of doing this is:

```
$ wget -r -l 1 -nd -A deb -R "*dev*" \
http://www.percona.com/downloads/Percona-Server-5.6/LATEST/deb/precise/x86_64/
```

Install them in one command:

```
$ sudo dpkg -i *.deb
```

The installation won't succeed as there will be missing dependencies. To handle this, use:

```
$ apt-get -f install
```

and all dependencies will be handled by **apt**.

The installation script will run automatically **mysql_upgrade** to migrate to the new grant tables and rebuild the indexes where needed.

RPM-based distributions

Having done the full backup (and dump if possible), stop the server:

```
$ /sbin/service mysql stop
```

and check your installed packages:

```
$ rpm -qa | grep Percona-Server
```

```
Percona-Server-client-55-5.5.29-rel29.4.401.rhel6.x86_64.rpm
Percona-Server-server-55-5.5.29-rel29.4.401.rhel6.x86_64.rpm
Percona-Server-shared-55-5.5.29-rel29.4.401.rhel6.x86_64.rpm
```

You may have a forth, `shared-compat`, which is for compatibility purposes.

After checked that, proceed to remove them without dependencies:

```
$ rpm -qa | grep Percona-Server | xargs rpm -e --nodeps
```

It is important that you remove it without dependencies as many packages may depend on these (as they replace `mysql`) and will be removed if omitted.

Note that this procedure is the same for upgrading from *MySQL 5.5* to *Percona Server 5.6*, just `grep '^mysql-'` instead of `Percona-Server` and remove them.

Download the following packages for your architecture:

- `Percona-Server-server-56`
- `Percona-Server-client-56`
- `Percona-Server-shared-56`

At the moment of writing this guide, a way of doing this is:

```
$ wget -r -l 1 -nd -A rpm -R "*devel*,*debuginfo*" \
http://www.percona.com/downloads/Percona-Server-5.6/LATEST/RPM/rhel6/x86_64/
```

Install them in one command:

```
$ rpm -ivh Percona-Server-shared-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm \
Percona-Server-client-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm \
Percona-Server-server-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm
```

If you don't install all "at the same time", you will need to do it in a specific order - `shared`, `client`, `server`:

```
$ rpm -ivh Percona-Server-shared-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm
$ rpm -ivh Percona-Server-client-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm
$ rpm -ivh Percona-Server-server-56-5.6.6-alpha60.1.285.rhel6.x86_64.rpm
```

Otherwise, the dependencies won't be met and the installation will fail.

Once installed, proceed to modify your configuration file - `my.cnf` - and recompile the plugins if necessary, as explained at the beginning of this guide.

As the schema of the grant table has changed, the server must be started without reading them:

```
$ /usr/sbin/mysqld --skip-grant-tables --user=mysql &
```

and use `mysql_upgrade` to migrate to the new grant tables, it will rebuild the indexes needed and do the modifications needed:

```
$ mysql_upgrade
```

After this is done, just restart the server as usual:

```
$ /sbin/service mysql restart
```

If it can't find the pid file, kill the server and start it normally:

```
$ killall /usr/sbin/mysqld  
$ /sbin/service mysql start
```


SCALABILITY IMPROVEMENTS

3.1 Improved Buffer Pool Scalability

The *InnoDB* buffer pool is a well known point of contention when many queries are executed concurrently. In *XtraDB*, the global mutex protecting the buffer pool has been split into several mutexes to decrease contention.

This feature splits the single global InnoDB buffer pool mutex into several mutexes:

Name	Protects
flush_state_mutex	flushing state of dirty blocks
LRU_list_mutex	LRU lists of blocks in buffer pool
flush_list_mutex	flush list of dirty blocks to flush
free_list_mutex	list of free blocks in buffer pool
zip_free_mutex	lists of free area to treat compressed pages
zip_hash_mutex	hash table to search compressed pages

The goal of this change is to reduce mutex contention, which can be very impacting when the working set does not fit in memory.

3.1.1 Version Specific Information

- 5.6.13–60.6 - Feature ported from *Percona Server 5.5*

3.1.2 Other Information

Detecting Mutex Contention

You can detect when you suffer from mutex contention in the buffer pool by reading the information provided in the SEMAPHORES section of the output of SHOW ENGINE INNODB STATUS:

Under normal circumstances this section should look like this:

SEMAPHORES

```
-----  
OS WAIT ARRAY INFO: reservation count 50238, signal count 17465  
Mutex spin waits 0, rounds 628280, OS waits 31338  
RW-shared spins 38074, OS waits 18900; RW-excl spins 0, OS waits 0
```

If you have a high-concurrency workload this section may look like this:

```
1 -----
2 SEMAPHORES
3 -----
4 OS WAIT ARRAY INFO: reservation count 36255, signal count 12675
5 --Thread 10607472 has waited at buf/buf0rea.c line 420 for 0.00 seconds the semaphore:
6 Mutex at 0x358068 created file buf/buf0buf.c line 597, lock var 0
7 waiters flag 0
8 --Thread 3488624 has waited at buf/buf0buf.c line 1177 for 0.00 seconds the semaphore:
9 Mutex at 0x358068 created file buf/buf0buf.c line 597, lock var 0
10 waiters flag 0
11 --Thread 6896496 has waited at btr/btr0cur.c line 442 for 0.00 seconds the semaphore:
12 S-lock on RW-latch at 0x8800244 created in file buf/buf0buf.c line 547
13 a writer (thread id 14879600) has reserved it in mode exclusive
14 number of readers 0, waiters flag 1
15 Last time read locked in file btr/btr0cur.c line 442
16 Last time write locked in file buf/buf0buf.c line 1797
[...]
```

Note that in the second case you will see indications that threads are waiting for a mutex created in the file `buf/buf0buf.c` (lines 5 to 7 or 8 to 10). Such an indication is a sign of buffer pool contention.

3.2 Improved *InnoDB* I/O Scalability

Because *InnoDB* is a complex storage engine it must be configured properly in order to perform at its best. Some points are not configurable in standard *InnoDB*. The goal of this feature is to provide a more exhaustive set of options for *XtraDB*, like ability to change the log block size.

3.2.1 Version Specific Information

- 5.6.11-60.3
 - Feature ported from *Percona Server 5.5*
- 5.6.14-62.0
 - New `innodb_log_checksum_algorithm` variable introduced.

3.2.2 System Variables

variable `innodb_use_global_flush_log_at_trx_commit`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Type Boolean

Default Value True

Range True/False

This variable is used to control the ability of the user to set the value of the global *MySQL* variable `innodb_flush_log_at_trx_commit`.

If `innodb_use_global_flush_log_at_trx_commit=0` (False), the client can set the global *MySQL* variable, using:

```
SET innodb_use_global_flush_log_at_trx_commit=N
```

If `innodb_use_global_flush_log_at_trx_commit=1` (True), the user session will use the current value of `innodb_flush_log_at_trx_commit`, and the user cannot reset the value of the global variable using a `SET` command.

variable `innodb_log_block_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 512

Units Bytes

This variable changes the size of transaction log records. The default size of 512 bytes is good in most situations. However, setting it to 4096 may be a good optimization with SSD cards. While settings other than 512 and 4096 are possible, as a practical matter these are really the only two that it makes sense to use. Clean restart and removal of the old logs is needed for the variable `innodb_log_block_size` to be changed. **Note:** This feature implementation is considered BETA quality.

variable `innodb_flush_method`

Version Info

- 5.6.13-61.0 - Ported from *Percona Server 5.5*

Command Line Yes

Config File Yes

Scope Global

Dyn No

Variable Type Enumeration

Default Value `fdatasync`

Allowed Values `fdatasync`, `O_DSYNC`, `O_DIRECT`, `O_DIRECT_NO_FSYNC`, `ALL_O_DIRECT`

This is an existing *MySQL 5.6* system variable that has a new allowed value `ALL_O_DIRECT`. It determines the method *InnoDB* uses to flush its data and log files. (See `innodb_flush_method` in the *MySQL 5.6 Reference Manual*).

The following values are allowed:

- `fdatasync`: use `fsync()` to flush both the data and log files.
- `O_SYNC`: use `O_SYNC` to open and flush the log files; use `fsync()` to flush the data files.
- `O_DIRECT`: use `O_DIRECT` to open the data files and `fsync()` system call to flush both the data and log files.

- `O_DIRECT_NO_FSYNC`: use `O_DIRECT` to open the data files but don't use `fsync()` system call to flush both the data and log files. This option isn't suitable for *XFS* file system.
- `ALL_O_DIRECT`: use `O_DIRECT` to open both data and log files, and use `fsync()` to flush the data files but not the log files. This option is recommended when *InnoDB* log files are big (more than 8GB), otherwise there might be even a performance degradation. **Note:** When using this option on *ext4* filesystem variable `innodb_log_block_size` should be set to 4096 (default log-block-size in *ext4*) in order to avoid the unaligned AIO/DIO warnings.

variable `innodb_log_checksum_algorithm`

Version Info

- 5.6.14-62.0 - Variable introduced

Command Line Yes

Config File Yes

Scope Global

Dyn No

Variable Type Enumeration

Default Value `innodb`

Allowed Values `none`, `innodb`, `crc32`, `strict_none`, `strict_innodb`, `strict_crc32`

This variable is used to specify how log checksums are generated and verified. Behavior of `innodb_log_checksum_algorithm` depending on its value is mostly identical to `innodb_checksum_algorithm`, except that the former applies to log rather than page checksums. **NOTE:** this feature is currently considered experimental.

The following values are allowed:

- `none`: means that a constant value will be written to log blocks instead of calculated checksum values and no checksum validation will be performed on *InnoDB*/*XtraBackup* recovery, or changed page tracking (if enabled).
- `innodb`: (the default) means the default *InnoDB* behavior – a custom and inefficient algorithm is used to calculate log checksums, but logs created with this option are compatible with upstream *MySQL* and earlier *Percona Server* or *Percona XtraBackup* versions that do not support other log checksum algorithms.
- `crc32`: will use CRC32 for log block checksums. Checksums will also benefit from hardware acceleration provided by recent Intel CPUs.
- `strict_*`: Normally, *XtraDB* or *Percona XtraBackup* will tolerate checksums created with other algorithms than is currently specified with the `innodb_log_checksum_algorithm` option. That is, if checksums don't match when reading the redo log on recovery, the block is considered corrupted only if no algorithm produces the value matching the checksum stored in the log block header. This can be disabled by prepending the value with the `strict_` suffix, e.g. `strict_none`, `strict_crc32` or `strict_innodb` will only accept checksums created using the corresponding algorithms, but not the other ones.

Status Variables

The following information has been added to `SHOW ENGINE INNODB STATUS` to confirm the checkpointing activity:

The max checkpoint age

The current checkpoint age target

The current age of the oldest page modification which has not been flushed to disk yet.

The current age of the last checkpoint

```

...
---
LOG
---
Log sequence number 0 1059494372
Log flushed up to   0 1059494372
Last checkpoint at  0 1055251010
Max checkpoint age  162361775
Checkpoint age target 104630090
Modified age        4092465
Checkpoint age      4243362
0 pending log writes, 0 pending chkp writes
...

```

3.2.3 Other Reading

- For Fusion-IO devices-specific tuning, see *Atomic write support for Fusion-io devices* documentation.

3.3 Multiple Adaptive Hash Search Partitions

The *InnoDB* adaptive hash index can have contention issues on multi-core systems when you run a mix of read and write queries that need to scan secondary indexes. This feature splits the adaptive hash index across several partitions to avoid such problems.

The number of adaptive hash partitions specified by the variable `innodb_adaptive_hash_index_partitions` are created, and hash indexes are assigned to each one based on `index_id`. This should help to solve contention problems in the adaptive hash search process when they occur.

3.3.1 Version Specific Information

- 5.6.13-60.6 - Feature ported from *Percona Server 5.5*

System Variables

variable `innodb_adaptive_hash_index_partitions`

Version Info

- **5.6.13-60.6** – Ported from *Percona Server 5.6*

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Def 1

Range 1-64, (on 32-bit platform 1-32)

Specifies the number of partitions to use in the adaptive hash search process.

When set to one, no extra partitions are created and the normal process is in effect. When greater than one, the specified number of partitions are created across which to perform the adaptive search.

Other reading

- [Index lock and adaptive search](#)
- [MySQL bug #62018](#)

3.4 Multiple user level locks per connection

Percona Server supports multiple user level locks per connection as of version 5.6.19-67.0. The `GET_LOCK()` function in upstream *MySQL* allows a connection to hold at most one user level lock. Taking a new lock automatically releases the old lock, if any.

The limit of one lock per session existed since early versions of *MySQL* didn't have a deadlock detector for SQL locks. [Metadata Locking](#) introduced in *MySQL* 5.5 added a deadlock detector, so starting from 5.5 it became possible to take multiple locks in any order. Deadlock would be detected when occurred, and an error would returned to the client which closed the wait chain.

3.4.1 Version Specific Information

- 5.6.19-67.0 - Feature implemented in *Percona Server* 5.6

3.4.2 Other Information

- Author / Origin: Kostja Osipov

3.4.3 Other Reading

- [MySQL: multiple user level locks per connection](#)

PERFORMANCE IMPROVEMENTS

4.1 Atomic write support for Fusion-io devices

Note: This feature implementation is considered BETA quality.

DirectFS filesystem on Fusion-io devices supports atomic writes. Atomic writes can be used instead of *InnoDB* doublewrite buffer to guarantee that the *InnoDB* data pages will be written to disk entirely or not at all. When atomic writes are enabled the device will take care of protecting the data against partial writes. In case the doublewrite buffer is enabled it will be disabled automatically. This will improve the write performance, because data doesn't need to be written twice anymore, and make the recovery simpler.

4.1.1 Version Specific Information

- **5.6.11–60.3** Atomic write support for Fusion-io feature implemented. This feature was ported from *MariaDB*.

4.1.2 System Variables

variable innodb_use_atomic_writes

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Default Value 0 (OFF)

This variable can be used to enable or disable atomic writes instead of the doublewrite buffer. When this option is enabled (set to 1), doublewrite buffer will be disabled on *InnoDB* initialization and the file flush method will be set to `O_DIRECT` if it's not `O_DIRECT` or `O_DIRECT_NO_FSYNC` already.

Warning: `innodb_use_atomic_writes` should only be enabled on supporting devices, otherwise *InnoDB* will fail to start.

4.1.3 Other Reading

- For general *InnoDB* tuning *Improved InnoDB I/O Scalability* documentation is available.
- FusionIO DirectFS atomic write support in *MariaDB*
- Atomic Writes Accelerate MySQL Performance

4.2 Query Cache Enhancements

This page describes the enhancements for the query cache. At the moment three features are available:

- Disabling the cache completely
- Diagnosing contention more easily
- Ignoring comments

4.2.1 Diagnosing contention more easily

This feature provides a new thread state - `Waiting on query cache mutex`. It has always been difficult to spot query cache bottlenecks because these bottlenecks usually happen intermittently and are not directly reported by the server. This new thread state appears in the output of `SHOW PROCESSLIST`, easing diagnostics.

Imagine that we run three queries simultaneously (each one in a separate thread):

```
> SELECT number from t where id > 0;
> SELECT number from t where id > 0;
> SELECT number from t where id > 0;
```

If we experience query cache contention, the output of `SHOW PROCESSLIST` will look like this:

```
> SHOW PROCESSLIST;
Id      User      Host          db          Command Time      State                                     Info
2       root     localhost    test       Sleep    2          NULL
3       root     localhost    test       Query   2          Waiting on query cache mutex          SELECT number i
4       root     localhost    test       Query   1          Waiting on query cache mutex          SELECT number i
5       root     localhost    test       Query   0          NULL
```

4.2.2 Ignoring comments

This feature adds an option to make the server ignore comments when checking for a query cache hit. For example, consider these two queries:

```
/* first query */ select name from users where users.name like 'Bob%';
/* retry search */ select name from users where users.name like 'Bob%';
```

By default (option off), the queries are considered different, so the server will execute them both and cache them both.

If the option is enabled, the queries are considered identical, so the server will execute and cache the first one and will serve the second one directly from the query cache.

4.2.3 System Variables

variable `query_cache_strip_comments`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value Off

Makes the server ignore comments when checking for a query cache hit.

Other Reading

- [MySQL general thread states](#)
- [Query cache freezes](#)

4.3 Improved NUMA support

In cases where the buffer pool memory allocation was bigger than size of the node, system would start swapping already allocated memory even if there is available memory on other node. This is would happen if the default *NUMA* memory allocation policy was selected. In that case system would favor one node more than other which caused the node to run out of memory. Changing the allocation policy to interleaving, memory will be allocated in round-robing fashion over the available node. This can be done by using the `mysqld_safe` `numa_interleave` option.

Another improvement implemented is preallocating the pages in the buffer pool on startup with `innodb_buffer_pool_populate` variable. This forces NUMA allocation decisions to be made immediately while the buffer cache is clean.

It is generally recommended to enable all of the options together to maximize the performance effects on the NUMA architecture.

4.3.1 Version Specific Information

- **5.6.11-60.3** Improved NUMA support implemented. This feature was ported from Twitter's *MySQL* patches.

4.3.2 System Variables

variable `innodb_buffer_pool_populate`

Command Line Yes

Config File No

Location `mysqld`

Scope Global

Dynamic No

Variable Type Boolean

Default Value OFF

Range ON/OFF

When this variable is enabled, *InnoDB* preallocates pages in the buffer pool on startup to force NUMA allocation decisions to be made immediately while the buffer cache is clean.

4.3.3 Command-line Options

variable `flush_caches`

Command Line No

Config File Yes

Location `mysqld_safe`

Dynamic No

Variable Type Boolean

Default Value OFF

Range ON/OFF

When enabled this will flush and purge buffers/caches before starting the server to help ensure NUMA allocation fairness across nodes. This option is useful for establishing a consistent and predictable behavior for normal usage and/or benchmarking.

variable `numa_interleave`

Command Line No

Config File Yes

Location `mysqld_safe`

Dynamic No

Variable Type Boolean

Default Value OFF

Range ON/OFF

When this option is enabled, `mysqld` will run with its memory interleaved on all NUMA nodes by starting it with `numactl --interleave=all`. In case there is just 1 CPU/node, allocations will be “interleaved” between that node.

4.3.4 Other Reading

- The MySQL “swap insanity” problem and the effects of the NUMA architecture
- A brief update on NUMA and MySQL

4.4 HandlerSocket

4.4.1 Description

HandlerSocket is a *MySQL* plugin that implements a NoSQL protocol for *MySQL*. This allows applications to communicate more directly with *MySQL* storage engines, without the overhead associated with using SQL. This includes operations such as parsing and optimizing queries, as well as table handling operations (opening, locking, unlocking, closing). As a result, using *HandlerSocket* can provide much better performance for certain applications that using normal SQL application protocols.

Complete documentation on the *HandlerSocket* plugin, including installation and configuration options, is located in the [project repository](#).

The plugin is disabled by default. To enable it in *Percona Server* with *XtraDB*, see below.

4.4.2 Version Specific Information

- 5.6.17–66.0 Full functionality available.

Other Information

Author/Origin Akira Higuchi, DeNA Co., Ltd.

4.4.3 Enabling the Plugin

Once *HandlerSocket* has been downloaded and installed on your system, there are two steps required to enable it.

First, add the following lines to the [mysqld] section of your *my.cnf* file:

```
loose_handlersocket_port = 9998
# the port number to bind to for read requests
loose_handlersocket_port_wr = 9999
# the port number to bind to for write requests
loose_handlersocket_threads = 16
# the number of worker threads for read requests
loose_handlersocket_threads_wr = 1
# the number of worker threads for write requests
open_files_limit = 65535
# to allow handlersocket to accept many concurrent
# connections, make open_files_limit as large as
# possible.
```

Second, log in to mysql as root, and execute the following query:

```
mysql> install plugin handlersocket soname 'handlersocket.so';
```

4.4.4 Testing the Plugin installation

If *handlersocket.so* was successfully installed, it will begin accepting connections on ports 9998 and 9999. Executing a `SHOW PROCESSLIST` command should show *HandlerSocket* worker threads:

```
mysql> SHOW PROCESSLIST;
```

```
+----+-----+-----+-----+-----+-----+
| Id | User          | Host                | db          | Command | Time | State |
+----+-----+-----+-----+-----+-----+-----+
```

```
| 1 | system user | connecting host | NULL | Connect | NULL | handlersocket: mode=rd, 0 cor
| 2 | system user | connecting host | NULL | Connect | NULL | handlersocket: mode=rd, 0 cor
...
| 16 | system user | connecting host | NULL | Connect | NULL | handlersocket: mode=rd, 0 cor
| 17 | system user | connecting host | handlersocket | Connect | NULL | handlersocket: mode=wr, 0 cor
```

To ensure *HandlerSocket* is working as expected, you can follow these steps:

Create a new table:

```
mysql> CREATE TABLE t (
  id int(11) NOT NULL,
  col varchar(20) NOT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB;
```

Insert a row with *HandlerSocket* (fields are separated by comma):

```
$ telnet 127.0.0.1 9999
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
P      1      test      t      PRIMARY id,col
0      1
1      +      2      1      test value
0      1
```

And check in SQL that the row has been written:

```
mysql> SELECT * FROM t;
+----+-----+
| id | col      |
+----+-----+
| 1 | test value |
+----+-----+
```

Configuration options

HandlerSocket has many configuration options that are detailed [here](#).

4.4.5 Other Reading

- Yoshinori Matsunobu's blog post describing [HandlerSocket](#)
- [Percona Server now both SQL and NOSQL](#)

4.5 Fixed Size for the Read Ahead Area

InnoDB dynamically calculates the size of the read-ahead area in case it has to trigger its read-ahead algorithm. When the workload involves heavy I/O operations, this size is computed so frequently that it has a non-negligible impact on the CPU usage. With this fix, buffer read-ahead area size is precalculated once per buffer pool instance initialization, and this value is used each time read-ahead is invoked. This implementation should remove a bottleneck experienced by some users.

4.5.1 Version Specific Information

- 5.6.13-60.5 : *Percona Server 5.5* feature re-implemented

4.5.2 Other Information

- Bugs fixed: #606811

4.5.3 Other Reading

- [BUF_READ_AHEAD_AREA Bottleneck](#)

4.6 Thread Pool

Note: This feature implementation is considered BETA quality.

MySQL executes statements using one thread per client connection. Once the number of connections increases past a certain point performance will degrade.

This feature enables the server to keep the top performance even with large number of client connections by introducing a dynamic thread pool. By using the thread pool server would decrease the number of threads, which will then reduce the context switching and hot locks contentions. Using the thread pool will have the most effect with OLTP workloads (relatively short CPU-bound queries).

In order to enable the thread pool variable `thread_handling` should be set up to `pool-of-threads` value. This can be done by adding:

```
thread_handling=pool-of-threads
```

to the *MySQL* configuration file `my.cnf`.

Although the default values for the thread pool should provide good performance, additional [tuning](#) can be performed with the dynamic system variables described below.

Note: Current implementation of the thread pool is built in the server, unlike the upstream version which is implemented as a plugin. Another significant implementation difference is that this implementation doesn't try to minimize the number of concurrent transactions like the *MySQL Enterprise Threadpool*. Because of these things this implementation isn't compatible with the upstream one.

4.6.1 Priority connection scheduling

In *Percona Server 5.6.11-60.3* priority connection scheduling for thread pool has been implemented. Even though thread pool puts a limit on the number of concurrently running queries, the number of open transactions may remain high, because connections with already started transactions are put to the end of the queue. Higher number of open transactions has a number of implications on the currently running queries. To improve the performance new `thread_pool_high_prio_tickets` variable has been introduced.

This variable controls the high priority queue policy. Each new connection is assigned this many tickets to enter the high priority queue. Whenever a query has to be queued to be executed later because no threads are available, the thread pool puts the connection into the high priority queue if the following conditions apply:

1. The connection has an open transaction in the server.

2. The number of high priority tickets of this connection is non-zero.

If both the above conditions hold, the connection is put into the high priority queue and its tickets value is decremented. Otherwise the connection is put into the common queue with the initial tickets value specified with this option.

Each time the thread pool looks for a new connection to process, first it checks the high priority queue, and picks connections from the common queue only when the high priority one is empty.

The goal is to minimize the number of open transactions in the server. In many cases it is beneficial to give short-running transactions a chance to commit faster and thus deallocate server resources and locks without waiting in the same queue with other connections that are about to start a new transaction, or those that have run out of their high priority tickets.

The default thread pool behavior is to always put events from already started transactions into the high priority queue, as we believe that results in better performance in vast majority of cases.

With the value of 0, all connections are always put into the common queue, i.e. no priority scheduling is used as in the original implementation in *MariaDB*. The higher is the value, the more chances each transaction gets to enter the high priority queue and commit before it is put in the common queue.

In some cases it is required to prioritize all statements for a specific connection regardless of whether they are executed as a part of a multi-statement transaction or in the autocommit mode. Or vice versa, some connections may require using the low priority queue for all statements unconditionally. To implement this new `thread_pool_high_prio_mode` variable has been introduced in *Percona Server 5.6.15-63.0*.

Low priority queue throttling

One case that can limit thread pool performance and even lead to deadlocks under high concurrency is a situation when thread groups are oversubscribed due to active threads reaching the oversubscribe limit, but all/most worker threads are actually waiting on locks currently held by a transaction from another connection that is not currently in the thread pool.

What happens in this case is that those threads in the pool that have marked themselves inactive are not accounted to the oversubscribe limit. As a result, the number of threads (both active and waiting) in the pool grows until it hits `thread_pool_max_threads` value. If the connection executing the transaction which is holding the lock has managed to enter the thread pool by then, we get a large (depending on the `thread_pool_max_threads` value) number of concurrently running threads, and thus, suboptimal performance as a result. Otherwise, we get a deadlock as no more threads can be created to process those transaction(s) and release the lock(s).

Such situations are prevented by throttling the low priority queue when the total number of worker threads (both active and waiting ones) reaches the oversubscribe limit. That is, if there are too many worker threads, do not start new transactions and create new threads until queued events from the already started transactions are processed.

4.6.2 Handling of Long Network Waits

Certain types of workloads (large result sets, BLOBs, slow clients) can have longer waits on network I/O (socket reads and writes). Whenever server waits, this should be communicated to the Thread Pool, so it can start new query by either waking a waiting thread or sometimes creating a new one. This implementation has been ported from *MariaDB* patch *MDEV-156* in *Percona Server 5.6.15-63.0*.

4.6.3 Version Specific Information

- **5.6.10-60.2** Thread Pool feature implemented. This feature was ported from *MariaDB*.
- **5.6.11-60.3** Implemented priority connection scheduling and introduced new variable `thread_pool_high_prio_tickets` to the original implementation introduced in *MariaDB*.

- **5.6.15–63.0** Default value for `thread_pool_max_threads` was changed from 500 to 100 000. This change was introduced because limiting the total number of threads in the *Thread Pool* can result in deadlocks and uneven distribution of worker threads between thread groups in case of stalled connections.
- **5.6.15–63.0** Default value for `thread_pool_high_prio_tickets` was changed from 0 to 4294967295 to enable the priority connection scheduling by default.
- **5.6.15–63.0** Implemented new `thread_pool_high_prio_mode` variable.
- **5.6.15–63.0** Implemented *Low priority queue throttling*.
- **5.6.15–63.0** Ported *MariaDB* patch [MDEV-156](#) to improve Thread Pool behavior when network wait times are not insignificant.

4.6.4 System Variables

variable `thread_pool_idle_timeout`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 60 (seconds)

This variable can be used to limit the time an idle thread should wait before exiting.

variable `thread_pool_high_prio_mode`

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Variable Type String

Default Value `transactions`

Allowed Values `transactions, statements, none`

This variable is used to provide more fine-grained control over high priority scheduling either globally or per connection.

The following values are allowed:

- `transactions` (the default). In this mode only statements from already started transactions may go into the high priority queue depending on the number of high priority tickets currently available in a connection (see `thread_pool_high_prio_tickets`).
- `statements`. In this mode all individual statements go into the high priority queue, regardless of connection's transactional state and the number of available high priority tickets. This value can be used to prioritize AUTOCOMMIT transactions or other kinds of statements such as administrative ones for specific connections. Note that setting this value globally essentially disables high priority scheduling, since in this case all statements from all connections will use a single queue (the high priority one)

- `none`. This mode disables high priority queue for a connection. Some connections (e.g. monitoring) may be insensitive to execution latency and/or never allocate any server resources that would otherwise impact performance in other connections and thus, do not really require high priority scheduling. Note that setting `thread_pool_high_prio_mode` to `none` globally has essentially the same effect as setting it to `statements` globally: all connections will always use a single queue (the low priority one in this case).

variable `thread_pool_high_prio_tickets`

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Variable Type Numeric

Default Value 4294967295

This variable controls the high priority queue policy. Each new connection is assigned this many tickets to enter the high priority queue. Setting this variable to 0 will disable the high priority queue.

variable `thread_pool_max_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 100000

This variable can be used to limit the maximum number of threads in the pool. Once this number is reached no new threads will be created.

variable `thread_pool_oversubscribe`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 3

The higher the value of this parameter the more threads can be run at the same time, if the value is lower than 3 it could lead to more sleeps and wake-ups.

variable `thread_pool_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value Number of processors

This variable can be used to define the number of threads that can use the CPU at the same time.

variable thread_pool_stall_limit

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 500 (ms)

The number of milliseconds before a running thread is considered stalled. When this limit is reached thread pool will wake up or create another thread. This is being used to prevent a long-running query from monopolizing the pool.

variable extra_port

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 0

This variable can be used to specify additional port *Percona Server* will listen on. This can be used in case no new connections can be established due to all worker threads being busy or being locked when `pool-of-threads` feature is enabled. To connect to the extra port following command can be used:

```
mysql --port='extra-port-number' --protocol=tcp
```

variable extra_max_connections

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 1

This variable can be used to specify the maximum allowed number of connections on the extra port. This can be used with the `extra_port` variable to access the server in case no new connections can be established due to all worker threads being busy or being locked when `pool-of-threads` feature is enabled.

4.6.5 Status Variables

variable Threadpool_idle_threads

Command Line Yes

Variable Type Numeric

This status variable shows the number of idle threads in the pool.

variable `Threadpool_threads`

Command Line Yes

Variable Type Numeric

This status variable shows the number of threads in the pool.

4.6.6 Other Reading

- [Thread pool in MariaDB 5.5](#)
- [Thread pool implementation in Oracle MySQL](#)

4.7 Page cleaner thread tuning

Percona Server has implemented page cleaner thread improvements in 5.6.13-61.0 release.

4.7.1 Pseudo-parallel flushing

Usage of the of multiple buffer pool instances is **not uniform**. The non-uniform buffer pool instance use means non-uniform free list depletion levels, which in turn causes transient stalls or single page LRU flushes for the query threads, increasing latency, and LRU/free list mutex contention. *Percona Server* has added heuristics that reduce the occurrence of depleted free lists and associated mutex contentions. Instead of issuing all the chunk-size LRU flush requests for the 1st instance, then for the 2nd instance, etc, the requests are issued to all instances in a pseudo-parallel manner: the 1st chunk for the 1st instance, the 1st chunk for the 2nd instance, etc., the 2nd chunk for the 1st instance, the 2nd chunk for the 2nd instance, etc. Moreover, if a particular instance has a nearly depleted free list (currently defined as <10% of `(innodb_lru_scan_depth)`), might be changed in the future), then the server keeps on issuing requests for that instance until it's not depleted anymore, or the flushing limit for it has been reached.

4.7.2 Furious Flushing

In certain situations it makes sense for InnoDB to flush pages as fast as possible instead of abiding to `innodb_io_capacity` setting and, starting with 5.6, `innodb_lru_scan_depth`. This is known as “furious flushing”. Oracle *MySQL* 5.6 page cleaner flushes may perform furious flushing for the flush list up to `innodb_io_capacity_max` I/Os per second. We have extended this flush list flushing so that page cleaner thread sleep is skipped whenever the checkpoint age is in the sync preflush zone, allowing issuing more than `innodb_io_capacity_max` per second if needed.

For the LRU flushing, Oracle *MySQL* 5.6 does not have furious flushing, which may cause single page LRU flushes in the case of empty free lists. We have implemented the LRU list furious flushing by allowing the page cleaner thread to sleep less if the free lists are nearly depleted. The sleep time is determined as follows: if free list is filled less than 1% over all buffer pool instances: no sleep; less than 5%: 50ms shorter sleep time than the previous iteration; between 5% and 20%: no change; more than 20%: 50ms longer sleep time.

4.7.3 Timeouts

Percona Server has implemented time limits for LRU and flush list flushes in a single page cleaner thread iteration. The thread assumes that one iteration of its main loop (LRU and flush list flushes) complete under 1 second, but under heavy load we have observed iterations taking up to 40 seconds. Such situations confuse the heuristics, and

an LRU or a flush taking a long time prevents the other kind of flush from running, which in turn may cause query threads to perform sync preflushes or single page LRU flushes depending on the starved flush type. If a LRU flush timeout happens, the current flushing pass over all buffer pool instances is still completed in order to ensure that all the instances have received at least a bit of flushing. In order to implement this for flush list flushes, the flush requests for each buffer pool instance were broken up to chunks too.

4.7.4 Adaptive Flushing Tuning

With the tuned page cleaner heuristics, adaptive flushing may become too aggressive and maintain a consistently lower checkpoint age than a similarly-configured stock server. This results in a performance loss due to reduced write combining. To address this *Percona Server* has implemented new LSN age factor formula for page cleaner adaptive flushing. that can be controlled with `innodb_cleaner_lsn_age_factor` variable.

variable `innodb_cleaner_lsn_age_factor`

Version Info

- **5.6.13-61.0** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Values legacy, high_checkpoint

Default Value high_checkpoint

This variable is used to specify which algorithm should be used for page cleaner adaptive flushing. When `legacy` option is set, server will use the upstream algorithm and when the `high_checkpoint` is selected, *Percona* implementation will be used.

4.7.5 Tuning Variables

Percona Server has introduced several tuning options, only available in builds compiled with `UNIV_PERF_DEBUG` or `UNIV_DEBUG` C preprocessor define. These options are experimental, thus their name, allowed values, their semantics, and `UNIV_PERF_DEBUG` presence may change at any future release. Their default values are used for the corresponding variables in regular (that is, no `UNIV_PERF_DEBUG` defined) builds.

variable `innodb_cleaner_max_lru_time`

Version Info

- **5.6.13-61.0** – Introduced.

Default Value 1000 (milliseconds)

This variable is used to specify the timeout for the LRU flush of one page cleaner thread iteration.

variable `innodb_cleaner_max_flush_time`

Version Info

- **5.6.13-61.0** – Introduced.

Default Value 1000 (milliseconds)

This variable is used to specify the timeout for the flush list flush.

variable `innodb_cleaner_lru_chunk_size`

Version Info

- 5.6.13-61.0 – Introduced.

Default Value 100

This variable replaces the hardcoded 100 constant as a chunk size for the LRU flushes.

variable innodb_cleaner_flush_chunk_size

Version Info

- 5.6.13-61.0 – Introduced.

Default Value 100

This variable is used for specifying the chunk size for the flush list flushes.

variable innodb_cleaner_free_list_lwm

Version Info

- 5.6.13-61.0 – Introduced.

Default Value 10

Values 0-100

This variable is used to specify the percentage of free list length below which LRU flushing will keep on iterating on the same buffer pool instance to prevent empty free list.

variable innodb_cleaner_eviction_factor

Version Info

- 5.6.13-61.0 – Introduced.

Variable Type Boolean

Values ON/OFF

Default Value OFF

This variable is used for choosing between flushed and evicted page counts for LRU flushing heuristics. If enabled, makes LRU tail flushing to use evicted instead of flushed page counts for its heuristics.

4.7.6 Other reading

- *XtraDB Performance Improvements for I/O-Bound Highly-Concurrent Workloads*
- #68481 - InnoDB LRU flushing for MySQL 5.6 needs work

4.8 XtraDB Performance Improvements for I/O-Bound Highly-Concurrent Workloads

In *Percona Server 5.6.13-61.0* a number of *XtraDB* performance improvements have been implemented for high-concurrency scenarios.

4.8.1 Priority refill for the buffer pool free list

In highly-concurrent I/O-bound workloads the following situation may happen:

1. Buffer pool free lists are used faster than they are refilled by the LRU cleaner thread.
2. Buffer pool free lists become empty and more and more query and utility (i.e. purge) threads stall, checking whether a buffer pool free list has become non-empty, sleeping, performing single-page LRU flushes.
3. The number of buffer pool free list mutex waiters increases.
4. When the LRU manager thread (or a single page LRU flush by a query thread) finally produces a free page, it is starved from putting it on the buffer pool free list as it must acquire the buffer pool free list mutex too. However, being one thread in up to hundreds, the chances of a prompt acquisition are low.

To avoid this *Percona Server* has implemented priority refill for the buffer pool free list in 5.6.13-61.0. This implementation adjusts the buffer pool free list producer to always acquire the mutex with high priority and buffer pool free list consumer to always acquire the mutex with low priority. The implementation makes use of *thread priority lock framework*.

Even the above implementation does not fully resolve the buffer pool free list mutex contention, as the mutex is still being acquired needlessly whenever the buffer pool free list is empty. This is addressed by delegating all the LRU flushes to the LRU manager thread, never attempting to evict a page or perform a LRU single page flush by a query thread, and introducing a backoff algorithm to reduce buffer pool free list mutex pressure on empty buffer pool free lists. This is controlled through a new system variable `innodb_empty_free_list_algorithm`.

variable `innodb_empty_free_list_algorithm`

Version Info

- 5.6.13-61.0 – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Values legacy, backoff

Default Value backoff

When `legacy` option is set, server will use the upstream algorithm and when the `backoff` is selected, *Percona* implementation will be used.

4.8.2 LRU manager thread

Percona Server 5.6.16-64.0 has split a new LRU manager thread out of the *InnoDB* page cleaner thread, that performs LRU flushes and evictions to refill the buffer pool free lists. Before this implementation this was done by the `page_cleaner` thread together with the flush list flushes.

Downsides of that approach were:

- The cleaner thread sleep time has to take into account both LRU and flush list needs, by using the shorter sleep time of the times requested by both. This meant that the code paths for the other one is called at times needlessly.
- LRU/flush list flushes in the cleaner are serialized, which may cause transient stalls (up to LRU/flush list flush timeout values, which are 1 second by default) for user threads in the cases of empty buffer pool free lists and large checkpoint age at the same time.

4.8.3 Backoff for sync preflushes

Currently if a log-writing thread finds that the checkpoint age is in the sync preflush zone, it will attempt to advance the checkpoint itself by issuing a flush list flush batch unless one is running already. After the page cleaner tuning, in some cases, this feature hinders more than helps: the cleaner thread knows that the system is in sync preflush state and will perform furious flushing itself. The thread doing its own flushes will only contribute to mutex pressure and use CPU. In such cases it is better for the query threads to wait for any required flushes to complete instead. Whenever a query thread needs to perform a sync preflush to proceed, two options are now available:

1. the query thread may issue a flush list batch itself and wait for it to complete. This is also used whenever the page cleaner thread is not running.
2. alternatively the query thread may wait until the flush list flush is performed by the page cleaner thread. The wait is implemented using a tweaked exponential backoff: the thread sleeps for a random progressively-increasing time waiting for the flush list flush to happen. The sleep time counter is periodically reset to avoid runaway sleeps. This algorithm may change in the future.

The behavior is controlled by a new system variable `innodb_foreground_preflush`.

variable `innodb_foreground_preflush`

Version Info

- **5.6.13-61.0** – Introduced.

Command Line Yes**Config File** Yes**Scope** Global**Dynamic** Yes**Values** `sync_preflush`, `exponential_backoff`**Default Value** `exponential_backoff`

4.8.4 Relative Thread Scheduling Priorities for XtraDB

Percona Server has implemented Relative Thread Scheduling Priorities for *XtraDB* in 5.6.13-61.0. This feature was implemented because whenever a high number of query threads is running on the server, the cleaner thread and other utility threads must receive more CPU time than a fair scheduling would allocate. New `innodb_sched_priority_cleaner` option has been introduced that corresponding to Linux `nice` values of -20 . . 19, where 0 is 19 (lowest priority) and 39 is -20 (highest priority). When new values are set server will attempt to set the thread nice priority for the specified thread type and return a warning with an actual priority if the attempt failed.

Note: This feature implementation is Linux-specific.

variable `innodb_sched_priority_cleaner`

Version Info

- **5.6.13-61.0** – Introduced.

Command Line Yes**Config File** Yes**Scope** Global**Dynamic** Yes

Values 1-39**Default Value** 19

This variable is used to set a thread scheduling priority. Values correspond to Linux `nice` values of `-20..19`, where 0 is 19 (lowest priority) and 39 is -20 (highest priority). This priority value affects both cleaner and LRU manager threads.

Percona Server has introduced several options, only available in builds compiled with `UNIV_PERF_DEBUG` C pre-processor define.

variable innodb_sched_priority_purge**Version Info**

- **5.6.13-61.0** – Introduced.

Command Line Yes**Config File** Yes**Scope** Global**Dynamic** Yes**Variable Type** Boolean**variable innodb_sched_priority_io****Version Info**

- **5.6.13-61.0** – Introduced.

Command Line Yes**Config File** Yes**Scope** Global**Dynamic** Yes**Variable Type** Boolean**variable innodb_sched_priority_master****Version Info**

- **5.6.13-61.0** – Introduced.

Command Line Yes**Config File** Yes**Scope** Global**Dynamic** Yes**Variable Type** Boolean

4.8.5 Thread Priority Locks

The *InnoDB* worker threads compete for the shared resource accesses with the query threads. Performance experiments show that under high concurrency the worker threads must acquire the shared resources with priority. To this end, a priority mutex and a priority RW lock locking primitives have been implemented, that use the existing sync array code to wake up any high-priority waiting threads before any low-priority waiting threads, as well as reduce any low-priority thread spinning if any high-priority waiters are already present for a given sync object. The following mutexes have been converted to be priority mutexes: `dict_sys`, buffer pool LRU list, buffer pool free list, `rseg`, `log_sys`, and

internal hash table sync object array mutexes. The following RW locks have been converted to priority RW locks: `fsp`, `page_hash`, `AHI`, `index`, and `purge`. To specify which threads are high-priority for shared resource acquisition, *Percona Server* has introduced several tuning options, only available in builds compiled with `UNIV_PERF_DEBUG` C preprocessor define.

variable `innodb_priority_purge`

Version Info

- **5.6.13-61.0** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

When this option is enabled purge coordinator and worker threads acquire shared resources with priority.

variable `innodb_priority_io`

Version Info

- **5.6.13-61.0** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

When this option is enabled I/O threads acquire shared resources with priority.

variable `innodb_priority_cleaner`

Version Info

- **5.6.13-61.0** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

When this option is enabled buffer pool page cleaner and LRU manager threads acquire shared resources with priority.

variable `innodb_priority_master`

Version Info

- **5.6.13-61.0** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

When buffer pool cleaner thread acquire shared resources with priority.

Note: These variables are intended for performance experimenting and not regular user tuning.

4.8.6 Other Reading

- *Page cleaner thread tuning*

FLEXIBILITY IMPROVEMENTS

5.1 Suppress Warning Messages

This feature is intended to provide a general mechanism (using `log_warnings_silence`) to disable certain warning messages to the log file. Currently, it is only implemented for disabling message #1592 warnings. This feature does not influence warnings delivered to a client.

5.1.1 Version Specific Information

- 5.6.11-60.3: Variable `log_warnings_suppress` ported from *Percona Server 5.5*.

5.1.2 System Variables

variable `log_warnings_suppress`

Version Info

- 5.6.11-60.3 – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type SET

Default Value (empty string)

Range (empty string),1592

It is intended to provide a more general mechanism for disabling warnings than existed previously with variable `suppress_log_warning_1592`. When set to the empty string, no warnings are disabled. When set to 1592, warning #1592 messages (unsafe statement for binary logging) are suppressed. In the future, the ability to optionally disable additional warnings may also be added.

5.1.3 Related Reading

- MySQL bug 42851
- MySQL InnoDB replication

- [InnoDB Startup Options and System Variables](#)
- [InnoDB Error Handling](#)

5.2 Improved MEMORY Storage Engine

As of MySQL 5.5.15, a *Fixed Row Format* (FRF) is still being used in the MEMORY storage engine. The fixed row format imposes restrictions on the type of columns as it assigns on advance a limited amount of memory per row. This renders a VARCHAR field in a CHAR field in practice and makes impossible to have a TEXT or BLOB field with that engine implementation.

To overcome this limitation, the *Improved MEMORY Storage Engine* is introduced in this release for supporting **true** VARCHAR, VARBINARY, TEXT and BLOB fields in MEMORY tables.

This implementation is based on the *Dynamic Row Format* (DRF) introduced by the `mysql-heap-dynamic-rows` patch.

DRF is used to store column values in a variable-length form, thus helping to decrease memory footprint of those columns and making possible BLOB and TEXT fields and real VARCHAR and VARBINARY.

Unlike the fixed implementation, each column value in DRF only uses as much space as required. This is, for variable-length values, up to 4 bytes is used to store the actual value length, and then only the necessary number of blocks is used to store the value.

Rows in DRF are represented internally by multiple memory blocks, which means that a single row can consist of multiple blocks organized into one set. Each row occupies at least one block, there can not be multiple rows within a single block. Block size can be configured when creating a table (see below).

This DRF implementation has two caveats regarding to ordering and indexes.

5.2.1 Caveats

Ordering of Rows

In the absence of `ORDER BY`, records may be returned in a different order than the previous MEMORY implementation.

This is not a bug. Any application relying on a specific order without an `ORDER BY` clause may deliver unexpected results. A specific order without `ORDER BY` is a side effect of a storage engine and query optimizer implementation which may and will change between minor *MySQL* releases.

Indexing

It is currently impossible to use indexes on BLOB columns due to some limitations of the *Dynamic Row Format*. Trying to create such an index will fail with the following error:

```
BLOB column '<name>' can't be used in key specification with the used table type.
```

5.2.2 Restrictions

For performance reasons, a mixed solution is implemented: the fixed format is used at the beginning of the row, while the dynamic one is used for the rest of it.

The size of the fixed-format portion of the record is chosen automatically on `CREATE TABLE` and cannot be changed later. This, in particular, means that no indexes can be created later with `CREATE INDEX` or `ALTER TABLE` when the dynamic row format is used.

All values for columns used in indexes are stored in fixed format at the first block of the row, then the following columns are handled with DFR.

This sets two restrictions to tables:

- the order of the fields and therefore,
- the minimum size of the block used in the table.

Ordering of Columns

The columns used in fixed format must be defined before the dynamic ones in the `CREATE TABLE` statement. If this requirement is not met, the engine will not be able to add blocks to the set for these fields and they will be treated as fixed.

Minimum Block Size

The block size has to be big enough to store all fixed-length information in the first block. If not, the `CREATE TABLE` or `ALTER TABLE` statements will fail (see below).

5.2.3 Setting Row Format

Taking the restrictions into account, the *Improved MEMORY Storage Engine* will choose DFR over FRF at the moment of creating the table according to following criteria:

- There is an implicit request of the user in the column types **OR**
- There is an explicit request of the user **AND** the overhead incurred by DFR is beneficial.

Implicit Request

The implicit request by the user is taken when there is at least one `BLOB` or `TEXT` column in the table definition. If there are none of these columns and no relevant option is given, the engine will choose FRF.

For example, this will yield the use of the dynamic format:

```
mysql> CREATE TABLE t1 (f1 VARCHAR(32), f2 TEXT, PRIMARY KEY (f1)) ENGINE=HEAP;
```

While this will not:

```
mysql> CREATE TABLE t1 (f1 VARCHAR(16), f2 VARCHAR(16), PRIMARY KEY (f1)) ENGINE=HEAP;
```

Explicit Request

The explicit request is set with one of the following options in the `CREATE TABLE` statement:

- `KEY_BLOCK_SIZE = <value>`
 - Requests the DFR with the specified block size (in bytes)
- `ROW_FORMAT = DYNAMIC`
 - Requests the dynamic format with the default block size (256 bytes)

Despite its name, the `KEY_BLOCK_SIZE` option refers to a block size used to store data rather than indexes. The reason for this is that an existing `CREATE TABLE` option is reused to avoid introducing new ones.

The *Improved MEMORY Engine* checks whether the specified block size is large enough to keep all key column values. If it is too small, table creation will abort with an error.

After DRF is requested explicitly and there are no `BLOB` or `TEXT` columns in the table definition, the *Improved MEMORY Engine* will check if using the dynamic format provides any space saving benefits as compared to the fixed one:

- if the fixed row length is less than the dynamic block size (plus the dynamic row overhead - platform dependent)
OR
- there isn't any variable-length columns in the table or `VARCHAR` fields are declared with length 31 or less,

the engine will revert to the fixed format as it is more space efficient in such case. The row format being used by the engine can be checked using `SHOW TABLE STATUS`.

5.2.4 Examples

On a 32-bit platform:

```
mysql> CREATE TABLE t1 (f1 VARCHAR(32), f2 VARCHAR(32), f3 VARCHAR(32), f4 VARCHAR(32),
PRIMARY KEY (f1)) KEY_BLOCK_SIZE=124 ENGINE=HEAP ROW_FORMAT=DYNAMIC;
```

```
mysql> SHOW TABLE STATUS LIKE 't1';
```

Name	Engine	Version	Row_format	Rows	Avg_row_length	Data_length	Max_data_length	Index_...
t1	MEMORY	10	Dynamic	0	X	0	0	NULL NULL NULL

On a 64-bit platform:

```
mysql> CREATE TABLE t1 (f1 VARCHAR(32), f2 VARCHAR(32), f3 VARCHAR(32), f4 VARCHAR(32),
PRIMARY KEY (f1)) KEY_BLOCK_SIZE=124 ENGINE=HEAP ROW_FORMAT=DYNAMIC;
```

```
mysql> SHOW TABLE STATUS LIKE 't1';
```

Name	Engine	Version	Row_format	Rows	Avg_row_length	Data_length	Max_data_length	Index_...
t1	MEMORY	10	Fixed	0	X	0	0	NULL NULL NULL

5.2.5 Implementation Details

MySQL MEMORY tables keep data in arrays of fixed-size chunks. These chunks are organized into two groups of `HP_BLOCK` structures:

- `group1` contains indexes, with one `HP_BLOCK` per key (part of `HP_KEYDEF`),
- `group2` contains record data, with a single `HP_BLOCK` for all records.

While columns used in indexes are usually small, other columns in the table may need to accommodate larger data. Typically, larger data is placed into `VARCHAR` or `BLOB` columns.

The *Improved MEMORY Engine* implements the concept of dataspace, `HP_DATASPACE`, which incorporates the `HP_BLOCK` structures for the record data, adding more information for managing variable-sized records.

Variable-size records are stored in multiple “chunks”, which means that a single record of data (a database “row”) can consist of multiple chunks organized into one “set”, contained in `HP_BLOCK` structures.

In variable-size format, one record is represented as one or many chunks depending on the actual data, while in fixed-size mode, one record is always represented as one chunk. The index structures would always point to the first chunk in the chunkset.

Variable-size records are necessary only in the presence of variable-size columns. The *Improved Memory Engine* will be looking for BLOB or VARCHAR columns with a declared length of 32 or more. If no such columns are found, the table will be switched to the fixed-size format. You should always put such columns at the end of the table definition in order to use the variable-size format.

Whenever data is being inserted or updated in the table, the *Improved Memory Engine* will calculate how many chunks are necessary.

For INSERT operations, the engine only allocates new chunksets in the recordspace. For UPDATE operations it will modify the length of the existing chunkset if necessary, unlinking unnecessary chunks at the end, or allocating and adding more if a larger length is needed.

When writing data to chunks or copying data back to a record, fixed-size columns are copied in their full format, while VARCHAR and BLOB columns are copied based on their actual length, skipping any NULL values.

When allocating a new chunkset of N chunks, the engine will try to allocate chunks one-by-one, linking them as they become allocated. For allocating a single chunk, it will attempt to reuse a deleted (freed) chunk. If no free chunks are available, it will try to allocate a new area inside a HP_BLOCK.

When freeing chunks, the engine will place them at the front of a free list in the dataspace, each one containing a reference to the previously freed chunk.

The allocation and contents of the actual chunks varies between fixed and variable-size modes:

- Format of a fixed-size chunk:
 - uchar[]
 - * With `sizeof=chunk_dataspace_length`, but at least `sizeof(uchar*)` bytes. It keeps actual data or pointer to the next deleted chunk, where `chunk_dataspace_length` equals to full record length
 - uchar
 - * Status field (1 means “in use”, 0 means “deleted”)
- Format of a variable-size chunk:
 - uchar[]
 - * With `sizeof=chunk_dataspace_length`, but at least `sizeof(uchar*)` bytes. It keeps actual data or pointer to the next deleted chunk, where `chunk_dataspace_length` is set according to table’s `key_block_size`
 - uchar*
 - * Pointer to the next chunk in this chunkset, or NULL for the last chunk
 - uchar
 - * Status field (1 means “first”, 0 means “deleted”, 2 means “linked”)

Total chunk length is always aligned to the next `sizeof(uchar*)`.

5.2.6 See Also

- [Dynamic row format for MEMORY tables](#)

5.3 Restricting the number of binlog files

Maximum number of binlog files can now be restricted in *Percona Server* with `max_binlog_files`. When variable `max_binlog_files` is set to non-zero value, the server will remove the oldest binlog file(s) whenever their number exceeds the value of the variable.

This variable can be used with the existing `max_binlog_size` variable to limit the disk usage of the binlog files. If `max_binlog_size` is set to 1G and `max_binlog_files` to 20 this will limit the maximum size of the binlogs on disk to 20G. The actual size limit is not necessarily `max_binlog_size * max_binlog_files`. Server restart or `FLUSH LOGS` will make the server start a new log file and thus resulting in log files that are not fully written in these cases limit will be lower.

5.3.1 Version Specific Information

- 5.6.11-60.3: Variable `max_binlog_files` introduced.

5.3.2 System Variables

variable `max_binlog_files`

Version Info

- **5.6.11-60.3** – Introduced.

Command Line Yes**Config File** Yes**Scope** Global**Dynamic** Yes**Variable Type** ULONG**Default Value** 0 (unlimited)**Range** 0-102400

5.3.3 Example

Number of the binlog files before setting this variable

```
$ ls -l mysql-bin.0* | wc -l
26
```

Variable `max_binlog_files` is set to 20:

```
max_binlog_files = 20
```

In order for new value to take effect `FLUSH LOGS` needs to be run. After that the number of binlog files is 20

```
$ ls -l mysql-bin.0* | wc -l
20
```

5.4 Ignoring missing tables in mysqldump

In case table name was changed during the **mysqldump** process taking place, **mysqldump** would stop with error:

```
Couldn't execute 'show create table testtable'
Table 'testdb.tabletest' doesn't exist (1146)\n")
```

This could happen if **mysqldump** was taking a backup of a working slave and during that process table name would get changed. This error happens because **mysqldump** takes the list of the tables at the beginning of the dump process but the `SHOW CREATE TABLE` happens just before the table is being dumped.

With this option **mysqldump** will still show error to `stderr`, but it will continue to work and dump the rest of the tables.

5.4.1 Version Specific Information

- 5.6.5-60.0 **mysqldump** option `--ignore-create-error` introduced

5.5 Extended `SELECT INTO OUTFILE/DUMPFILE`

Percona Server has extended the `SELECT INTO ... OUTFILE` and `SELECT INTO DUMPFILE` commands to add the support for UNIX sockets and named pipes. Before this was implemented the database would return an error for such files.

This feature allows using `LOAD DATA LOCAL INFILE` in combination with `SELECT INTO OUTFILE` to quickly load multiple partitions across the network or in other setups, without having to use an intermediate file which wastes space and I/O.

5.5.1 Version Specific Information

- 5.6.13-61.0 - Feature Implemented

5.5.2 Other Reading

- *MySQL* bug: #44835

5.6 Per-query variable statement

Percona Server has implemented per-query variable statement support in 5.6.14-62.0. This feature provides the ability to set variable values only for a certain query, after execution of which the previous values will be restored. Per-query variable values can be set up with the following command:

```
mysql> SET STATEMENT <variable=value> FOR <statement>;
```

5.6.1 Examples

If we want to increase the `sort_buffer_size` value just for one specific sort query we can do it like this:

```
mysql> SET STATEMENT sort_buffer_size=100000 FOR SELECT name FROM name ORDER BY name;
```

This feature can also be used with *Statement Timeout* to limit the execution time for a specific query:

```
mysql> SET STATEMENT max_statement_time=1000 FOR SELECT name FROM name ORDER BY name;
```

We can provide more than one variable we want to set up:

```
mysql> SET STATEMENT sort_buffer_size=100000, max_statement_time=1000 FOR SELECT name FROM name ORDER BY name;
```

5.6.2 Version Specific Information

- 5.6.14-62.0 Feature implemented

5.6.3 Other Reading

- WL#681: Per query variable settings

5.7 Extended mysqlbinlog

Percona Server has implemented compression support for **mysqlbinlog** in 5.6.15-63.0. This is similar to support that both `mysql` and `mysqldump` programs include (the `-C`, `--compress` options “Use compression in server/client protocol”). Using the compressed protocol helps reduce the bandwidth use and speed up transfers.

Percona Server has also implemented support for SSL. **mysqlbinlog** now accepts the SSL connection options as all the other client programs. This feature can be useful with `--read-from-remote-server` option. Following SSL options are now available:

- `--ssl` - Enable SSL for connection (automatically enabled with other flags).
- `--ssl-ca=name` - CA file in PEM format (check OpenSSL docs, implies `--ssl`).
- `--ssl-capath=name` - CA directory (check OpenSSL docs, implies `--ssl`).
- `--ssl-cert=name` - X509 cert in PEM format (implies `--ssl`).
- `--ssl-cipher=name` - SSL cipher to use (implies `--ssl`).
- `--ssl-key=name` - X509 key in PEM format (implies `--ssl`).
- `--ssl-verify-server-cert` - Verify server’s “Common Name” in its cert against hostname used when connecting. This option is disabled by default.

5.7.1 Version Specific Information

- 5.6.15-63.0 **mysqlbinlog** option `--compress` introduced
- 5.6.15-63.0 **mysqlbinlog** now has all SSL connection options as the rest of the *MySQL* client programs.

5.8 Slow Query Log Rotation and Expiration

Note: This feature is currently considered BETA quality.

Percona has implemented two new variables, `max_slowlog_size` and `max_slowlog_files` to provide users with ability to control the slow query log disk usage. These variables have the same behavior as upstream variable `max_binlog_size` and `max_binlog_files` variable used for controlling the binary log.

Warning: For this feature to work variable `slow_query_log_file` needs to be set up manually and without the `.log` suffix. The slow query log files will be named using `slow_query_log_file` as a stem, to which a dot and a sequence number will be appended.

5.8.1 Version Specific Information

- 5.6.16-64.0: Variable `max_slowlog_size` introduced.
- 5.6.16-64.0: Variable `max_slowlog_files` introduced.

5.8.2 System Variables

variable `max_slowlog_size`

Version Info

- **5.6.16-64.0** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type numeric

Default Value 0 (unlimited)

Range 4096 - 1073741824

Slow query log will be rotated automatically when its size exceeds this value. The default is 0, don't limit the size. When this feature is enabled slow query log file will be renamed to `slow_query_log_file.000001`.

variable `max_slowlog_files`

Version Info

- **5.6.16-64.0** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type numeric

Default Value 0 (unlimited)

Range 0 - 102400

Maximum number of slow query log files. Used with `max_slowlog_size` this can be used to limit the total amount of slow query log files. When this number is reached server will create a new slow query log file with increased sequence number. Log file with the lowest sequence number will be deleted.

5.9 Ability to change database for mysqlbinlog

Sometimes there is a need to take a binary log and apply it to a database with a different name than the original name of the database on binlog producer.

New option `rewrite-db` has been added to the `mysqlbinlog` utility that allows the changing names of the used databases in both Row-Based and Statement-Based replication. This was possible before by using tools like `grep`, `awk` and `sed` but only for SBR, because with RBR database name is encoded within the BINLOG ‘...’ statement.

Option `rewrite-db` of `mysqlbinlog` utility allows to setup rewriting rule “from->to”.

5.9.1 Example

mysqlbinlog output before rewrite-db

```
$ mysqlbinlog mysql-bin.000005
...
# at 175
#120517 13:10:00 server id 2  end_log_pos 203  Intvar
SET INSERT_ID=4083/*!*/;
# at 203
#120517 13:10:00 server id 2  end_log_pos 367  Query  thread_id=88  exec_time=0  error_code=0
use world/*!*/;
SET TIMESTAMP=1337253000/*!*/;
insert into City (Name, CountryCode, District, Population) values ("New City", "ZMB", "TEX", 111000)
/*!*/;
# at 367
#120517 13:10:00 server id 2  end_log_pos 394  Xid = 1414
COMMIT/*!*/;
DELIMITER ;
```

mysqlbinlog output when the new option is used:

```
$ mysqlbinlog --rewrite-db='world->new_world' mysql-bin.000005
...
# at 106
use new_world/*!*/;
#120517 13:10:00 server id 2  end_log_pos 175  Query  thread_id=88  exec_time=0  error_code=0
SET TIMESTAMP=1337253000/*!*/;
SET @@session.pseudo_thread_id=88/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=1, @@session.unique_checks=1, @@sessio
SET @@session.sql_mode=0/*!*/;
SET @@session.auto_increment_increment=1, @@session.auto_increment_offset=1/*!*/;
/*!!\C latin1 *//*!*/;
SET @@session.character_set_client=8,@@session.collation_connection=8,@@session.collation_server=8/*
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
BEGIN
/*!*/;
# at 175
```

```
#120517 13:10:00 server id 2  end_log_pos 203  Intvar
SET INSERT_ID=4083/*!*/;
# at 203
#120517 13:10:00 server id 2  end_log_pos 367  Query  thread_id=88  exec_time=0  error_code=0
SET TIMESTAMP=1337253000/*!*/;
insert into City (Name, CountryCode, District, Population) values ("New City", "ZMB", "TEX", 111000)
/*!*/;
# at 367
#120517 13:10:00 server id 2  end_log_pos 394  Xid = 1414
COMMIT/*!*/;
```

5.9.2 Version Specific Information

- 5.6.16-64.0 Full functionality.

5.9.3 Client Command Line Parameter

option rewrite-db

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type String

Default Value Off

5.9.4 Related Reading

- [WL #36](#)

RELIABILITY IMPROVEMENTS

6.1 Too Many Connections Warning

This feature issues the warning `Too many connections` to the log, if `log_warnings` is enabled.

6.1.1 Version-Specific Information

- 5.6.11-60.3: Feature ported from *Percona Server 5.5*.

6.2 Error Code Compatibility

Percona Server with XtraDB has error code incompatibilities with *MySQL 5.6*. It is important to maintain compatibility in the error codes used by the servers. For example, scripts that may be run on both servers could contain references to error codes.

The reasons for the current incompatibilities are:

- *Percona Server with XtraDB* contains features that have been backported from *MySQL 5.6*. Some of the *MySQL 5.6* features added new error codes.
- Some *Percona Server with XtraDB* features have added new error codes.

The solution to the first problem is to preserve *MySQL 5.6* error codes in the *Percona Server*. An example of where this has been done is *Percona Server* feature Query Cache Enhancements. This feature adds error `ER_QUERY_CACHE_DISABLED` to the *Percona Server*, which is defined as error code 1651 in *MySQL 5.6*.

After migrating *Percona Server / XtraDB* to *MySQL 5.6*, users might experience troubles because of this.

The solution to the second problem is to insure that unique error codes are chosen, when adding new ones to *Percona Server*, that will never be duplicated during *MySQL* development.

For example, *MySQL* has a tool `comp_err` that generates:

- `errmsg.sys` files
- header file `include/mysqld_error.h`
- header file `include/mysqld_ename.h`

from the file `errmsg.txt`.

To keep error numbers consistent, we should add some fictive errors to `errmsg.txt`, because `comp_err` assigns error code numbers sequentially, without gaps.

I propose patch to `comp_err`.

This patch allows usage of a new syntax, with prefix `PADD`, for example:

```
PADD_QUERY_CACHE_DISABLED 1651
  eng "ER_QUERY_CACHE_DISABLED padding to 1651 error"
ER_QUERY_CACHE_DISABLED
  eng "Query cache is disabled; restart the server with query_cache_type=1 to enable it"
```

`comp_err` with my patch pads empty intervals (from last error code number to 1651) by error message `ER_QUERY_CACHE_DISABLED padding to 1651 error`, i.e. and `ER_QUERY_CACHE_DISABLED` now has error code 1651 (as desired). I propose to use this patch for Percona errors, for example:

```
PADD_PERCONA_NEW_ERROR_CODE 4000
  end "Padd empty space to error code number 4000 (Percona error codes)"
...some percona error codes...
```

Patch only adds prefix `PADD_` and pads error in sys files. All other *MySQL* code (load*.sys files, my_error, etc) works as old one.

6.2.1 Version-Specific Information

- 5.6.11-60.3 Feature ported from *Percona Server 5.5*.

6.3 Handle Corrupted Tables

Instead of crashing the server as they used to do, corrupted *InnoDB* tables are simply disabled, so that the database remains available while the corruption is being fixed.

This feature adds a new system variable.

6.3.1 Version Specific Information

- 5.6.10-60.2: Feature ported from *Percona Server 5.5*

6.3.2 System Variables

variable `innodb_corrupt_table_action`

Version Info

- **5.6.10-60.2** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Range assert, warn, salvage

- With the default value *XtraDB* will intentionally crash the server with an assertion failure as it would normally do when detecting corrupted data in a single-table tablespace.

- If the `warn` value is used it will pass corruption of the table as `corrupt table` instead of crashing itself. For this to work `innodb_file_per_table` should be enabled. All file I/O for the datafile after detected as corrupt is disabled, except for the deletion.
- When the option value is `salvage`, *XtraDB* allows read access to a corrupted tablespace, but ignores corrupted pages”.

6.4 Lock-Free SHOW SLAVE STATUS

The `STOP SLAVE` and `SHOW SLAVE STATUS` commands can conflict due to a global lock in the situation where one thread on a slave attempts to execute a `STOP SLAVE` command, while a second thread on the slave is already running a command that takes a long time to execute.

If a `STOP SLAVE` command is given in this situation, it will wait and not complete execution until the long-executing thread has completed its task. If another thread now executes a `SHOW SLAVE STATUS` command while the `STOP SLAVE` command is waiting to complete, the `SHOW SLAVE STATUS` command will not be able to execute while the `STOP SLAVE` command is waiting.

This feature modifies the `SHOW SLAVE STATUS` syntax to allow:

```
SHOW SLAVE STATUS NONBLOCKING
```

This will display the slave’s status as if there were no lock, allowing the user to detect and understand the situation that is occurring.

Note: The information given when `NONBLOCKING` is used may be slightly inconsistent with the actual situation while the lock is being held.

Note: *Percona Server* originally used `SHOW SLAVE STATUS NOLOCK` syntax for this feature. As of 5.6.20-68.0 release, *Percona Server* implements `SHOW SLAVE STATUS NONBLOCKING` syntax, which comes from *MySQL 5.7*. The `NOLOCK` one has been deprecated and will be removed in *Percona Server 5.7*.

6.4.1 Version Specific Information

- 5.6.11-60.3: Feature ported from *Percona Server 5.5*.
- 5.6.20-68.0: *Percona Server* implemented the `NONBLOCKING` syntax from *MySQL 5.7* and deprecated the `NOLOCK` syntax.

MANAGEMENT IMPROVEMENTS

7.1 *Percona Toolkit* UDFs

Three *Percona Toolkit* UDFs that provide faster checksums are provided:

- `libfnv1a_udf`
- `libfnv_udf`
- `libmurmur_udf`

7.1.1 Version Specific Information

- 5.6.11-60.3: Began distributing `libfnv1a_udf`, `libfnv_udf`, and `libmurmur_udf`.

7.1.2 Other Information

- Author / Origin: Baron Schwartz

7.1.3 Installation

These UDFs are part of the *Percona Server* packages. To install one of the UDFs into the server, execute one of the following commands, depending on which UDF you want to install:

```
mysql -e "CREATE FUNCTION fnv1a_64 RETURNS INTEGER SONAME 'libfnv1a_udf.so'"
mysql -e "CREATE FUNCTION fnv_64 RETURNS INTEGER SONAME 'libfnv_udf.so'"
mysql -e "CREATE FUNCTION murmur_hash RETURNS INTEGER SONAME 'libmurmur_udf.so'"
```

Executing each of these commands will install its respective UDF into the server.

7.1.4 Troubleshooting

If you get the error:

```
ERROR 1126 (HY000): Can't open shared library 'fnv_udf.so' (errno: 22 fnv_udf.so: cannot open shared
```

Then you may need to copy the `.so` file to another location in your system. Try both `/lib` and `/usr/lib`. Look at your environment's `$LD_LIBRARY_PATH` variable for clues. If none is set, and neither `/lib` nor `/usr/lib` works, you may need to set `LD_LIBRARY_PATH` to `/lib` or `/usr/lib`.

7.1.5 Other Reading

- *Percona Toolkit* documentation

7.2 Support for Fake Changes

Note: This feature implementation is considered ALPHA quality.

Restarting a slave server in a replication environment or setting up new slave server can cause a replication reads slower. This is happening because replication in *MySQL* is single-threaded and because it needs to read the data before it can execute the queries. The process can be sped up by having prefetch threads to warm the server: replay statements and then rollback at commit.

That makes prefetch simple but has high overhead from locking rows only to undo changes at rollback.

Using this approach, support for *Fake Changes* have been implemented in order to remove the overhead and make it faster.

By reading the rows for `INSERT`, `UPDATE` and `DELETE` statements but not updating them (*Fake Changes*), the rollback is very fast as in most cases there is nothing to do.

7.2.1 Caveats

DML operations are supported

Currently only DML operations **are supported**, i.e. `UPDATE`, `INSERT`, `REPLACE` and `DELETE` (set deleted flag).

DDL operations are not supported

DDL operations **are not supported**, i.e. `ALTER TABLE` and `TRUNCATE TABLE`. *Fake Changes* should be disabled temporarily if DDL statements are going to be executed. Otherwise, data may be lost.

Explicit `COMMIT` will lead to an error

From the viewpoint of transactional RDBMS, `COMMIT` should not be “fake” anytime. `ROLLBACK` must be used to terminate the fake transaction.

7.2.2 System Variables

variable `innodb_fake_changes`

Version Info

- **5.6.11-60.3** – Introduced

Scope GLOBAL

Type BOOLEAN

Dynamic YES

Default Value FALSE

This variable enables the *Fake Changes* feature.

variable `innodb_locking_fake_changes`

Version Info

- **5.6.11-60.3** – Introduced

Scope GLOBAL

Type BOOLEAN

Dynamic YES

Default Value TRUE

When this variable is set to `FALSE`, it makes fake transactions not to take any row locks. This feature was implemented because, although fake change transactions downgrade the requested exclusive (X) row locks to shared (S) locks, these S locks prevent X locks from being taken and block the real changes. However, this option is not safe to set to `FALSE` by default, because the fake changes implementation is not ready for lock-less operation for all workloads. Namely, if a real transaction will remove a row that a fake transaction is doing a secondary index maintenance for, the latter will fail. This option is considered experimental and might be removed in the future if lockless operation mode fixes are implemented.

7.2.3 Implementation Details

- The fake session is used as a prefetch of the replication, it should not affect to later replication SQL execution.
- The effective unit is each transaction. The behavior is decided at the start of the each one and never changed during the transaction
- `INSERT` operations doesn't use the `INSERT BUFFER`, it always causes the reading of the page actually for the option. `DELETE` also doesn't use the `INSERT BUFFER`.
- It never acquires `X_LOCK` from tables or records, only `S_LOCK`.
- The auto increment values behaves as usual.
- It reserves free pages as usual.
- Existed only `root ~ leaf` pages, which are accessed in the DML operation.
- It will not prefetch allocate/free, split/merge, `INODE`, `XDES` or other management pages. The same is for extern pages, i.e. large `BLOB` s).
- Foreign key constraints are checked (for causing IO), but passed always.

7.2.4 Related Reading

- on MySQL replication prefetching

7.3 Kill Idle Transactions

This feature limits the age of idle *XtraDB* transactions. If a transaction is idle for more seconds than the threshold specified, it will be killed. This prevents users from blocking purge by mistake.

7.3.1 System Variables

variable `innodb_kill_idle_transaction`

Version Info

- 5.6.11-60.3 – Introduced

Scope GLOBAL

Config YES

Dynamic YES

Variable Type INTEGER

Default Value 0 (disabled)

Units Seconds

To enable this feature, set this variable to the desired seconds wait until the transaction is killed.

7.4 Enforcing Storage Engine

Percona Server has implemented variable which can be used for enforcing the use of a specific storage engine.

When this variable is specified and a user tries to create a table using an explicit storage engine that is not the specified enforced engine, he will get either an error if the `NO_ENGINE_SUBSTITUTION SQL` mode is enabled or a warning if `NO_ENGINE_SUBSTITUTION` is disabled and the table will be created anyway using the enforced engine (this is consistent with the default *MySQL* way of creating the default storage engine if other engines aren't available unless `NO_ENGINE_SUBSTITUTION` is set).

In case user tries to enable `enforce_storage_engine` with engine that isn't available, system will not start.

Note: If you're using `enforce_storage_engine`, you must either disable it before doing `mysql_upgrade` or perform `mysql_upgrade` with server started with `--skip-grants-tables`.

7.4.1 Version Specific Information

- 5.6.11-60.3 Variable `enforce_storage_engine` implemented.

7.4.2 System Variables

variable `enforce_storage_engine`

Command Line No

Config File Yes

Scope Global

Dynamic No

Variable Type String

Default Value NULL

7.4.3 Example

Adding following option to *my.cnf* will start the server with InnoDB as enforced storage engine.

```
enforce_storage_engine=InnoDB
```

7.5 Utility user

Percona Server has implemented ability to have a *MySQL* user who has system access to do administrative tasks but limited access to user schema. This feature is especially useful to those operating *MySQL* As A Service.

This user has a mixed and special scope of abilities and protection:

- Utility user will not appear in the `mysql.user` table and can not be modified by any other user, including root.
- Utility user will not appear in `USER_STATISTICS`, `CLIENT_STATISTICS` or `THREAD_STATISTICS` tables.
- Utility user's queries may appear in the general and slow logs.
- Utility user doesn't have the ability create, modify, delete or see any schemas or data not specified (except for `information_schema`).
- Utility user may modify all visible, non read-only system variables (see *Expanded Program Option Modifiers* functionality).
- Utility user may see, create, modify and delete other system users only if given access to the `mysql` schema.
- Regular users may be granted proxy rights to the utility user but any attempt to impersonate the utility user will fail. The utility user may not be granted proxy rights on any regular user. For example running: `GRANT PROXY ON utility_user TO regular_user;` will not fail, but any actual attempt to impersonate as the utility user will fail. Running: `GRANT PROXY ON regular_user TO utility_user;` will fail when `utility_user` is an exact match or is more specific than the utility user specified.

When the server starts, it will note in the log output that the utility user exists and the schemas that it has access to.

In order to have the ability for a special type of *MySQL* user, which will have a very limited and special amount of control over the system and can not be see or modified by any other user including the root user, three new options have been added.

Option `utility_user` specifies the user which the system will create and recognize as the utility user. The host in the utility user specification follows conventions described in the *MySQL manual*, i.e. it allows wildcards and IP masks. Anonymous user names are not permitted to be used for the utility user name.

This user must not be an exact match to any other user that exists in the `mysql.user` table. If the server detects that the user specified with this option exactly matches any user within the `mysql.user` table on start up, the server will report an error and shut down gracefully. If host name wildcards are used and a more specific user specification is identified on start up, the server will report a warning and continue.

Example: `--utility_user =frank@%` and `frank@localhost` exists within the `mysql.user` table.

If a client attempts to create a *MySQL* user that matches this user specification exactly or if host name wildcards are used for the utility user and the user being created has the same name and a more specific host, the creation attempt will fail with an error.

Example: `--utility_user =frank@%` and `CREATE USER 'frank@localhost';`

As a result of these requirements, it is strongly recommended that a very unique user name and reasonably specific host be used and that any script or tools test that they are running within the correct user by executing `'SELECT CURRENT_USER()'` and comparing the result against the known utility user.

Option `utility_user_password` specifies the password for the utility user and MUST be specified or the server will shut down gracefully with an error.

Example: `--utility_user_password='Passw0rD';`

Option `utility_user_schema_access` specifies the name(s) of the schema(s) that the utility user will have access to read write and modify. If a particular schema named here does not exist on start up it will be ignored. If a schema by the name of any of those listed in this option is created after the server is started, the utility user will have full access to it.

Example: `--utility_user_schema_access=schema1,schema2,schema3 ;`

Option `utility_user_privileges` allows a comma-separated list of extra access privileges to grant to the utility user.

Example: `--utility-user-privileges="CREATE, DROP, LOCK TABLES"`

7.5.1 System Variables

variable `utility_user`

Version Info

- 5.6.11-60.3 – Implemented

Command Line Yes

Config File `utility_user=<user@host>`

Scope Global

Dynamic No

Variable Type String

Default Value NULL

Specifies a MySQL user that will be added to the internal list of users and recognized as the utility user.

variable `utility_user_password`

Version Info

- 5.6.11-60.3 – Implemented

Command Line Yes

Config File `utility_user_password=<password>`

Scope Global

Dynamic No

Variable Type String

Default Value NULL

Specifies the password required for the utility user.

variable `utility_user_schema_access`

Version Info

- 5.6.11-60.3 – Implemented

Command Line Yes

Config File `utility_user_schema_access=<schema>,<schema>,<schema>`

Scope Global**Dynamic** No**Variable Type** String**Default Value** NULL

Specifies the schemas that the utility user has access to in a comma delimited list.

variable `utility_user_privileges`**Version Info**

- 5.6.13-60.6 – Implemented

Command Line Yes**Config File** `utility_user_privileges=<privilege1>,<privilege2>,<privilege3>`**Scope** Global**Dynamic** No**Variable Type** String**Default Value** NULL

This variable can be used to specify a comma-separated list of extra access privileges to grant to the utility user. Supported values for the privileges list are: `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `DROP`, `RELOAD`, `SHUTDOWN`, `PROCESS`, `FILE`, `GRANT`, `REFERENCES`, `INDEX`, `ALTER`, `SHOW DATABASES`, `SUPER`, `CREATE TEMPORARY TABLES`, `LOCK TABLES`, `EXECUTE`, `REPLICATION SLAVE`, `REPLICATION CLIENT`, `CREATE VIEW`, `SHOW VIEW`, `CREATE ROUTINE`, `ALTER ROUTINE`, `CREATE USER`, `EVENT`, `TRIGGER`, `CREATE TABLESPACE`

7.6 Extending the `secure-file-priv` server option

Percona Server has extended `secure-file-priv` server option. When used with no argument, the `LOAD_FILE()` function will always return `NULL`. The `LOAD DATA INFILE` and `SELECT INTO OUTFILE` statements will fail with the following error: “The MySQL server is running with the `-secure-file-priv` option so it cannot execute this statement”. `LOAD DATA LOCAL INFILE` is not affected by the `-secure-file-priv` option and will still work when it’s used without an argument.

7.6.1 Version Specific Information

- 5.6.11-60.3 Variable `secure-file-priv` extended behavior implemented.

7.6.2 System Variables

variable `secure-file-priv`**Command Line** No**Config File** Yes**Scope** Global**Dynamic** No**Variable Type** String

Default Value NULL

7.7 Expanded Program Option Modifiers

MySQL has the concept of **options modifiers** which is a simple way to modify either the way that *MySQL* interprets an option or the way the option behaves. Option modifiers are used by simply prepending the name of the modifier and a dash “-” before the actual configuration option name. For example specifying `--maximum-query_cache_size=4M` on the `mysqld` command line or specifying `maximum-query_cache_size=4M` in the `my.cnf` will prevent any client from setting the `query_cache_size` value larger than 4MB.

Currently MySQL supports five existing option modifiers:

- `disable` [`disable-<option_name>`] disables or ignores `option_name`.
- `enable` [`enable-<option_name>`] enables `option_name`.
- `loose` [`loose-<option_name>`] - `mysqld` will not exit with an error if it does not recognize `option_name`, but instead it will issue only a warning.
- `maximum` [`maximum-<option_name>=<value>`] indicates that a client can not set the value of `option_name` greater than the limit specified. If the client does attempt to set the value of `option_name` greater than the limit, the `option_name` will simply be set to the defined limit.
- `skip` [`skip-<option_name>`] skips or ignores `option_name`.

In order to offer more control over option visibility, access and range limits, the following new option modifiers have been added:

- `minimum` [`minimum-<option_name>=<value>`] indicates that clients can not set the value of `option_name` to less than the limit specified. If the client does attempt to set the value of `option_name` lesser than the limit, the `option_name` will simply be set to the defined limit.
- `hidden` [`hidden-<option_name>=<TRUE/FALSE>`] indicates that clients can not see or modify the value of `option_name`.
- `readonly` [`readonly-<option_name>=<TRUE/FALSE>`] indicates that clients can see the value of `option_name` but can not modify the value.

7.7.1 Combining the options

Some of the option modifiers may be used together in the same option specification, example:

```
--skip-loose-<option_name>
--loose-readonly-<option_name>=<T/F>
--readonly-<option_name>=<T/F>
--hidden-<option_name>=<T/F>
```

7.7.2 Version Specific Information

- 5.6.11-60.3 Expanded program option modifiers implemented

7.7.3 Examples

Adding the following option to the `my.cnf` will set the minimum limit on `query_cache_size`


```
minimum-query_cache_size = 4M
```

Trying to set up bigger value will work correctly, but if we try to set it up with smaller than the limit, defined minimum limit will be used and warning (1292) will be issued:

Initial `query_cache_size` size:

```
mysql> show variables like 'query_cache_size';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| query_cache_size | 8388608 |
+-----+-----+
1 row in set (0.00 sec)
```

Setting up bigger value:

```
mysql> set global query_cache_size=16777216;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> show variables like 'query_cache_size';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| query_cache_size | 16777216 |
+-----+-----+
1 row in set (0.00 sec)
```

Setting up smaller value:

```
mysql> set global query_cache_size=1048576;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
mysql> show warnings;
+-----+-----+-----+-----+
| Level | Code | Message |
+-----+-----+-----+-----+
| Warning | 1292 | Truncated incorrect query_cache_size value: '1048576' |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> show variables like 'query_cache_size';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| query_cache_size | 4194304 |
+-----+-----+
1 row in set (0.00 sec)
```

Adding following option to `my.cnf` will make `query_cache_size` hidden.

```
hidden-query_cache_size=1
```

```
mysql> show variables like 'query_cache%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| query_cache_limit | 1048576 |
| query_cache_min_res_unit | 4096 |
| query_cache_strip_comments | OFF |
+-----+-----+
```

```
| query_cache_type           | ON       |
| query_cache_wlock_invalidate | OFF      |
+-----+-----+
5 rows in set (0.00 sec)
```

Adding following option to `my.cnf` will make `query_cache_size` read-only

```
readonly-query_cache_size=1
```

Trying to change the variable value will result in error:

```
mysql> show variables like 'query_cache%';
+-----+-----+
| Variable_name           | Value    |
+-----+-----+
| query_cache_limit       | 1048576  |
| query_cache_min_res_unit | 4096     |
| query_cache_size        | 8388608  |
| query_cache_strip_comments | OFF      |
| query_cache_type        | ON       |
| query_cache_wlock_invalidate | OFF      |
+-----+-----+
6 rows in set (0.00 sec)

mysql> set global query_cache_size=16777216;
ERROR 1238 (HY000): Variable 'query_cache_size' is a read only variable
```

7.8 XtraDB changed page tracking

XtraDB now tracks the pages that have changes written to them according to the redo log. This information is written out in special changed page bitmap files. This information can be used to speed up incremental backups using [Percona XtraBackup](#) by removing the need to scan whole data files to find the changed pages. Changed page tracking is done by a new *XtraDB* worker thread that reads and parses log records between checkpoints. The tracking is controlled by a new read-only server variable `innodb_track_changed_pages`.

Bitmap filename format used for changed page tracking is `ib_modified_log_<seq>_<startlsn>.xdb`. The first number is the sequence number of the bitmap log file and the *startlsn* number is the starting LSN number of data tracked in that file. Example of the bitmap log files should look like this:

```
ib_modified_log_1_0.xdb
ib_modified_log_2_1603391.xdb
```

Sequence number can be used to easily check if all the required bitmap files are present. Start LSN number will be used in *XtraBackup* and `INFORMATION_SCHEMA` queries to determine which files have to be opened and read for the required LSN interval data. The bitmap file is rotated on each server restart and whenever the current file size reaches the predefined maximum. This maximum is controlled by a new `innodb_max_bitmap_file_size` variable.

Old bitmap files may be safely removed after a corresponding incremental backup is taken. For that there are server *User statements for handling the XtraDB changed page bitmaps*. Removing the bitmap files from the filesystem directly is safe too, as long as care is taken not to delete data for not-yet-backupped LSN range.

This feature will be used for implementing faster incremental backups that use this information to avoid full data scans in *Percona XtraBackup*.

7.8.1 User statements for handling the XtraDB changed page bitmaps

In *Percona Server 5.6.11-60.3* new statements have been introduced for handling the changed page bitmap tracking. All of these statements require `SUPER` privilege.

- `FLUSH CHANGED_PAGE_BITMAPS` - this statement can be used for synchronous bitmap write for immediate catch-up with the log checkpoint. This is used by innobackupex to make sure that XtraBackup indeed has all the required data it needs.
- `RESET CHANGED_PAGE_BITMAPS` - this statement will delete all the bitmap log files and restart the bitmap log file sequence.
- `PURGE CHANGED_PAGE_BITMAPS BEFORE <lsn>` - this statement will delete all the change page bitmap files up to the specified log sequence number.

7.8.2 Additional information in SHOW ENGINE INNODB STATUS

When log tracking is enabled, the following additional fields are displayed in the `LOG` section of the `SHOW ENGINE INNODB STATUS` output:

- “Log tracked up to:” displays the LSN up to which all the changes have been parsed and stored as a bitmap on disk by the log tracking thread
- “Max tracked LSN age:” displays the maximum limit on how far behind the log tracking thread may be.

7.8.3 INFORMATION_SCHEMA Tables

This table contains a list of modified pages from the bitmap file data. As these files are generated by the log tracking thread parsing the log whenever the checkpoint is made, it is not real-time data.

table `INFORMATION_SCHEMA.INNODB_CHANGED_PAGES`

Columns

- `space_id` (*INT(11)*) – space id of modified page
- `page_id` (*INT(11)*) – id of modified page
- `start_lsn` (*BIGINT(21)*) – start of the interval
- `end_lsn` (*BIGINT(21)*) – end of the interval

The `start_lsn` and the `end_lsn` columns denote between which two checkpoints this page was changed at least once. They are also equal to checkpoint LSNs.

Number of records in this table can be limited by using the variable `innodb_max_changed_pages`.

7.8.4 System Variables

variable `innodb_max_changed_pages`

Version Info

- **5.6.11-60.3** – Variable `innodb_max_changed_pages` introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes
Variable Type Numeric
Default Value 1000000
Range 1 - 0 (unlimited)

This variable is used to limit the result row count for the queries from `INNODB_CHANGED_PAGES` table.

variable innodb_track_changed_pages

Version Info

- **5.6.11-60.3** – Variable introduced

Command Line Yes
Config File Yes
Scope Global
Dynamic No
Variable Type Boolean
Default Value 0 - False
Range 0-1

This variable is used to enable/disable *XtraDB changed page tracking* feature.

variable innodb_max_bitmap_file_size

Version Info

- **5.6.11-60.3** – Variable introduced

Command Line Yes
Config File Yes
Scope Global
Dynamic Yes
Variable Type Numeric
Default Value 104857600 (100 MB)
Range 4096 (4KB) - 18446744073709551615 (16EB)

This variable is used to control maximum bitmap size after which the file will be rotated.

7.9 PAM Authentication Plugin

Percona PAM Authentication Plugin is a free and Open Source implementation of the *MySQL*'s authentication plugin. This plugin acts as a mediator between the *MySQL* server, the *MySQL* client, and the PAM stack. The server plugin requests authentication from the PAM stack, forwards any requests and messages from the PAM stack over the wire to the client (in cleartext) and reads back any replies for the PAM stack.

PAM plugin uses dialog as its client side plugin. Dialog plugin can be loaded to any client application that uses `libperconaserverclient/libperconaserverclient` library.

Here are some of the benefits that Percona dialog plugin offers over the default one:

- It correctly recognizes whether PAM wants input to be echoed or not, while the default one always echoes the input on the user’s console.
- It can use the password which is passed to *MySQL* client via “-p” parameter.
- Dialog client [installation bug](#) has been fixed.
- This plugin works on *MySQL* and *Percona Server*.

Percona offers two versions of this plugin:

- Full PAM plugin called *auth_pam*. This plugin uses *dialog.so*. It fully supports the PAM protocol with arbitrary communication between client and server.
- Oracle-compatible PAM called *auth_pam_compat*. This plugin uses *mysql_clear_password* which is a part of Oracle MySQL client. It also has some limitations, such as, it supports only one password input. You must use `-p` option in order to pass the password to *auth_pam_compat*.

These two versions of plugins are physically different. To choose which one you want used, you must use *IDENTIFIED WITH ‘auth_pam’* for *auth_pam*, and *IDENTIFIED WITH ‘auth_pam_compat’* for *auth_pam_compat*.

7.9.1 Installation

This plugin requires manual installation because it isn’t installed by default.

```
mysql> INSTALL PLUGIN auth_pam SONAME 'auth_pam.so';
```

After the plugin has been installed it should be present in the plugins list. To check if the plugin has been correctly installed and active

```
mysql> SHOW PLUGINS;
...
...
| auth_pam                | ACTIVE    | AUTHENTICATION    | auth_pam.so | GPL    |
```

7.9.2 Configuration

In order to use the plugin, authentication method should be configured. Simple setup can be to use the standard UNIX authentication method (`pam_unix`).

Note: To use `pam_unix`, `mysql` will need to be added to the shadow group in order to have enough privileges to read the `/etc/shadow`.

A sample `/etc/pam.d/mysqld` file:

```
auth        required    pam_unix.so
account    required    pam_unix.so
```

For added information in the system log, you can expand it to be:

```
auth        required    pam_warn.so
auth        required    pam_unix.so audit
account    required    pam_unix.so audit
```

7.9.3 Creating a user

After the PAM plugin has been configured, users can be created with the PAM plugin as authentication method

```
mysql> CREATE USER 'newuser'@'localhost' IDENTIFIED WITH auth_pam;
```

This will create a user `newuser` that can connect from `localhost` who will be authenticated using the PAM plugin. If the `pam_unix` method is being used user will need to exist on the system.

7.9.4 Supplementary groups support

Percona Server has implemented PAM plugin support for supplementary groups. Supplementary or secondary groups are extra groups a specific user is member of. For example user `joe` might be a member of groups: `joe` (his primary group) and secondary groups `developers` and `dba`. A complete list of groups and users belonging to them can be checked with `cat /etc/group` command.

This feature enables using secondary groups in the mapping part of the authentication string, like “`mysql, developers=joe, dba=mark`”. Previously only primary groups could have been specified there. If user is a member of both `developers` and `dba`, PAM plugin will map it to the `joe` because `developers` matches first.

7.9.5 Version Specific Information

- 5.6.11-60.3 PAM authentication plugin has been integrated with *Percona Server*.
- 5.6.12-60.4 Implemented PAM support for supplementary groups.

7.10 Expanded Fast Index Creation

Note: This feature implementation is considered BETA quality.

Percona has implemented several changes related to *MySQL*'s fast index creation feature. This feature expands the `ALTER TABLE` command by adding a new clause that provides online index renaming capability, that is renaming indexes without rebuilding the whole table.

7.10.1 Enabling Expanded Fast Index Creation

Fast index creation was implemented in *MySQL* as a way to speed up the process of adding or dropping indexes on tables with many rows. However, cases have been found in which fast index creation creates an inconsistency between *MySQL* and *InnoDB* data dictionaries.

This feature implements a session variable that enables extended fast index creation. Besides optimizing DDL directly, `expand_fast_index_creation` may also optimize index access for subsequent DML statements because using it results in much less fragmented indexes.

mysqldump

A new option, `--innodb-optimize-keys`, was implemented in **mysqldump**. It changes the way *InnoDB* tables are dumped, so that secondary and foreign keys are created after loading the data, thus taking advantage of fast index creation. More specifically:

- KEY, UNIQUE KEY, and CONSTRAINT clauses are omitted from CREATE TABLE statements corresponding to *InnoDB* tables.
- An additional ALTER TABLE is issued after dumping the data, in order to create the previously omitted keys.

ALTER TABLE

When ALTER TABLE requires a table copy, secondary keys are now dropped and recreated later, after copying the data. The following restrictions apply:

- Only non-unique keys can be involved in this optimization.
- If the table contains foreign keys, or a foreign key is being added as a part of the current ALTER TABLE statement, the optimization is disabled for all keys.

OPTIMIZE TABLE

Internally, OPTIMIZE TABLE is mapped to ALTER TABLE ... ENGINE=innodb for *InnoDB* tables. As a consequence, it now also benefits from fast index creation, with the same restrictions as for ALTER TABLE.

Caveats

InnoDB fast index creation uses temporary files in tmpdir for all indexes being created. So make sure you have enough tmpdir space when using `expand_fast_index_creation`. It is a session variable, so you can temporarily switch it off if you are short on tmpdir space and/or don't want this optimization to be used for a specific table.

There's also a number of cases when this optimization is not applicable:

- UNIQUE indexes in ALTER TABLE are ignored to enforce uniqueness where necessary when copying the data to a temporary table;
- ALTER TABLE and OPTIMIZE TABLE always process tables containing foreign keys as if `expand_fast_index_creation` is OFF to avoid dropping keys that are part of a FOREIGN KEY constraint;
- `mysqldump --innodb-optimize-keys` ignores foreign keys because *InnoDB* requires a full table rebuild on foreign key changes. So adding them back with a separate ALTER TABLE after restoring the data from a dump would actually make the restore slower;
- `mysqldump --innodb-optimize-keys` ignores indexes on AUTO_INCREMENT columns, because they must be indexed, so it is impossible to temporarily drop the corresponding index;
- `mysqldump --innodb-optimize-keys` ignores the first UNIQUE index on non-nullable columns when the table has no PRIMARY KEY defined, because in this case *InnoDB* picks such an index as the clustered one.

7.10.2 Version Specific Information

- 5.6.10-60.2 Variable `expand_fast_index_creation` implemented. This variable is controlling whether fast index creation optimizations made by Percona are used.

7.10.3 System Variables

variable `expand_fast_index_creation`**Command Line** Yes**Config File** No**Scope** Local/Global**Dynamic** Yes**Variable Type** Boolean**Default Value** OFF**Range** ON/OFF

7.10.4 Other Reading

- [Improved InnoDB fast index creation](#)
- [Thinking about running OPTIMIZE on your InnoDB Table? Stop!](#)

7.11 Log Archiving for XtraDB

Note: This feature implementation is considered BETA quality.

XtraDB and *InnoDB* write to the redo log files in a cyclic manner, so that the oldest log data is overwritten with the newest one. This feature makes copies of the old log files before they are overwritten, thus saving all the redo log for a write workload.

When log archiving is enabled, it duplicates all redo log writes in a separate set of files in addition to normal redo log writing, creating new files as necessary.

Archived log file name format is `ib_log_archive_<startlsn>`. The start LSN marks the log sequence number when the archive was started. An example of the archived log files should look like this:

```
ib_log_archive_00000000010145937920
ib_log_archive_00000000010196267520
```

The oldest archived logs can be removed automatically by setting up the `innodb_log_arch_expire_sec` variable.

This feature can be used to create incremental backups with *Percona XtraBackup* as described in this [guide](#).

7.11.1 User statements for handling the XtraDB log archiving

New statements have been introduced in *Percona Server* for handling the *XtraDB* log archiving. Both of these statements require SUPER privilege.

- `PURGE ARCHIVED LOGS BEFORE <datetime>` - this will delete archived logs modified before datetime. Archive which is currently in progress will not be deleted.
- `PURGE ARCHIVED LOGS TO <log_filename>` - this will delete all archived logs up to the 'log_filename' (including 'log_filename' too). Archive which is currently in progress will not be deleted.

7.11.2 Limitations

When log archiving is enabled both redo and archived logs need to be written to disk, which can have some IO performance impact. It can also lead to more aggressive flushing because less space is available in the redo log.

7.11.3 Version Specific Information

- 5.6.11-60.3: Feature implemented

7.11.4 System Variables

variable `innodb_log_archive`

Version Info

- 5.6.11-60.3 – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Values ON/OFF

Default Value OFF

This variable is used to enable or disable log archiving.

variable `innodb_log_arch_dir`

Version Info

- 5.6.11-60.3 – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Type Text

Default Value ./

This variable is used to specify the log archiving directory.

variable `innodb_log_arch_expire_sec`

Version Info

- 5.6.11-60.3 – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Type Numeric

Default Value 0

Number of seconds since last modification after which archived log should be deleted.

7.12 Statement Timeout

Note: This feature implementation is considered ALPHA quality.

Percona Server has implemented a statement timeout feature. This feature can be used to limit the query execution time by specifying the timeout value in the `max_statement_time` variable. After the specified number of milliseconds is reached the server will attempt to abort the statement and return the following error to the client:

```
ERROR 1877 (70101): Query execution was interrupted, max_statement_time exceeded
```

7.12.1 Version Specific Information

- 5.6.13-61.0: Statement timeout implemented. This feature with some changes was ported from [Twitter MySQL](#) patches.

7.12.2 System Variables

variable `max_statement_time`

Version Info

- 5.6.13-61.0 – Introduced.

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Variable Type Numeric

Values 0 - 18446744073709551615

Default Value 0 (no timeout is enforced)

Units milisecond

This system variable is used to specify the maximum execution time for any statement. After specified number of milliseconds is reached server will attempt to abort the statement.

variable `have_statement_timeout`

Version Info

- 5.6.13-61.0 – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Values YES/NO

This system variable shows if the feature is supported for the current operating system.

7.12.3 Status Variables

variable `Max_statement_time_exceeded`

Version Info

- 5.6.13-61.0 – Introduced.

Scope Global

Type Numeric

This status variable shows the number of statements that exceeded execution time limit.

variable `Max_statement_time_set`

Version Info

- 5.6.13-61.0 – Introduced.

Scope Global

Type Numeric

This status variable shows the number of statements for which execution time limit was set.

variable `Max_statement_time_set_failed`

Version Info

- 5.6.13-61.0 – Introduced.

Scope Global

Type Numeric

This status variable shows the number of statements for which execution time limit could not be set, that can happen if some OS-related limits were exceeded.

7.13 Backup Locks

Percona Server has implemented this feature in 5.6.16-64.0 to be a lightweight alternative to `FLUSH TABLES WITH READ LOCK` for both physical and logical backups. Three new statements are now available: `LOCK TABLES FOR BACKUP`, `LOCK BINLOG FOR BACKUP` and `UNLOCK BINLOG`.

7.13.1 LOCK TABLES FOR BACKUP

`LOCK TABLES FOR BACKUP` uses a new MDL lock type to block updates to non-transactional tables and DDL statements for all tables. More specifically, if there's an active `LOCK TABLES FOR BACKUP` lock, all DDL statements and all updates to MyISAM, CSV, MEMORY and ARCHIVE tables will be blocked in the `Waiting for backup lock` status as visible in `PERFORMANCE_SCHEMA` or `PROCESSLIST`. `SELECT` queries for all tables and `INSERT/REPLACE/UPDATE/DELETE` against *InnoDB*, Blackhole and Federated tables are not affected by `LOCK`

TABLES FOR BACKUP. Blackhole tables obviously have no relevance for backups, and Federated tables are ignored by both logical and physical backup tools.

Unlike FLUSH TABLES WITH READ LOCK, LOCK TABLES FOR BACKUP does not flush tables, i.e. storage engines are not forced to close tables and tables are not expelled from the table cache. As a result, LOCK TABLES FOR BACKUP only waits for conflicting statements to complete (i.e. DDL and updates to non-transactional tables). It never waits for SELECTs, or UPDATEs to *InnoDB* tables to complete, for example.

If an “unsafe” statement is executed in the same connection that is holding a LOCK TABLES FOR BACKUP lock, it fails with the following error:

```
ERROR 1880 (HY000): Can't execute the query because you have a conflicting backup lock
UNLOCK TABLES releases the lock acquired by LOCK TABLES FOR BACKUP.
```

7.13.2 LOCK BINLOG FOR BACKUP

LOCK BINLOG FOR BACKUP uses another new MDL lock type to block all operations that might change either binary log position or `Exec_Master_Log_Pos` or `Exec_Gtid_Set` (i.e. master binary log coordinates corresponding to the current SQL thread state on a replication slave) as reported by `SHOW MASTER/SLAVE STATUS`. More specifically, a commit will only be blocked if the binary log is enabled (both globally, and for connection with `sql_log_bin`), or if commit is performed by a slave thread and would advance `Exec_Master_Log_Pos` or `Executed_Gtid_Set`. Connections that are currently blocked on the global binlog lock can be identified by the `Waiting for binlog lock status` in `PROCESSLIST`.

7.13.3 UNLOCK BINLOG

UNLOCK BINLOG releases the LOCK BINLOG FOR BACKUP lock, if acquired by the current connection. The intended use case for *Percona XtraBackup* is:

```
LOCK TABLES FOR BACKUP
... copy .frm, MyISAM, CSV, etc. ...
LOCK BINLOG FOR BACKUP
UNLOCK TABLES
... get binlog coordinates ...
... wait for redo log copying to finish ...
UNLOCK BINLOG
```

7.13.4 Privileges

Both LOCK TABLES FOR BACKUP and LOCK BINLOG FOR BACKUP require the RELOAD privilege. The reason for that is to have the same requirements as FLUSH TABLES WITH READ LOCK.

7.13.5 Interaction with other global locks

Both LOCK TABLES FOR BACKUP and LOCK BINLOG FOR BACKUP have no effect if the current connection already owns a FLUSH TABLES WITH READ LOCK lock, as it's a more restrictive lock. If FLUSH TABLES WITH READ LOCK is executed in a connection that has acquired LOCK TABLES FOR BACKUP or LOCK BINLOG FOR BACKUP, FLUSH TABLES WITH READ LOCK fails with an error.

If the server is operating in the read-only mode (i.e. `read_only` set to 1), statements that are unsafe for backups will be either blocked or fail with an error, depending on whether they are executed in the same connection that owns LOCK TABLES FOR BACKUP lock, or other connections.

7.13.6 MyISAM index and data buffering

MyISAM key buffering is normally write-through, i.e. by the time each update to a *MyISAM* table is completed, all index updates are written to disk. The only exception is delayed key writing feature which is disabled by default.

When the global system variable `delay_key_write` is set to `ALL`, key buffers for all *MyISAM* tables are not flushed between updates, so a physical backup of those tables may result in broken *MyISAM* indexes. To prevent this, `LOCK TABLES FOR BACKUP` will fail with an error if `delay_key_write` is set to `ALL`. An attempt to set `delay_key_write` to `ALL` when there's an active backup lock will also fail with an error.

Another option to involve delayed key writing is to create *MyISAM* tables with the `DELAY_KEY_WRITE` option and set the `delay_key_write` variable to `ON` (which is the default). In this case, `LOCK TABLES FOR BACKUP` will not be able to prevent stale index files from appearing in the backup. Users are encouraged to set `delay_key_writes` to `OFF` in the configuration file, `my.cnf`, or repair *MyISAM* indexes after restoring from a physical backup created with backup locks.

MyISAM may also cache data for bulk inserts, e.g. when executing multi-row `INSERT`s or `LOAD DATA` statements. Those caches, however, are flushed between statements, so have no effect on physical backups as long as all statements updating *MyISAM* tables are blocked.

7.13.7 mysqldump

`mysqldump` has also been extended with a new option, `lock-for-backup` (disabled by default). When used together with the `--single-transaction` option, the option makes `mysqldump` issue `LOCK TABLES FOR BACKUP` before starting the dump operation to prevent unsafe statements that would normally result in an inconsistent backup.

When used without the `single-transaction` option, `lock-for-backup` is automatically converted to `lock-all-tables`.

Option `lock-for-backup` is mutually exclusive with `lock-all-tables`, i.e. specifying both on the command line will lead to an error.

If the backup locks feature is not supported by the target server, but `lock-for-backup` is specified on the command line, `mysqldump` aborts with an error.

If `master-data` is used together with `single-transaction`, `lock-for-backup` does not have any effect, i.e. `FLUSH TABLES WITH READ LOCK` will still be used to get the binary log coordinates. This limitation will be removed after implementing the following [blueprint](#).

System Variables

variable `have_backup_locks`

Version Info

- 5.6.16-64.0 – Implemented

Command Line Yes

Config File No

Scope Global

Dynamic No

Variable Type Boolean

Default Value YES

This is a server variable implemented to help other utilities decide what locking strategy can be implemented for a server. When available, the backup locks feature is supported by the server and the variable value is always `YES`.

Status Variables

variable `Com_lock_tables_for_backup`

Version Info

- **5.6.16-64.0** – Implemented

Variable Type Numeric

variable `Com_lock_binlog_for_backup`

Version Info

- **5.6.16-64.0** – Implemented

Variable Type Numeric

variable `Com_unlock_binlog`

Version Info

- **5.6.16-64.0** – Implemented

Variable Type Numeric

These status variables indicate the number of times the corresponding statements have been executed.

Client Command Line Parameter

option `lock-for-backup`

Version Info

- **5.6.16-64.0** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value Off

When used together with the `--single-transaction` option, the option makes `mysqldump` issue `LOCK TABLES FOR BACKUP` before starting the dump operation to prevent unsafe statements that would normally result in an inconsistent backup.

7.14 Audit Log Plugin

Percona Audit Log Plugin provides monitoring and logging of connection and query activity that were performed on specific server. Information about the activity will be stored in the XML log file where each event will have its `NAME` field, its own unique `RECORD_ID` field and a `TIMESTAMP` field. This implementation is alternative to the MySQL Enterprise Audit Log Plugin

Audit Log plugin produces the log of following events:

- **Audit** - Audit event indicates that audit logging started or finished. NAME field will be `Audit` when logging started and `NoAudit` when logging finished. Audit record also includes server version and command-line arguments.

Example of the Audit event:

```
<AUDIT_RECORD
  "NAME"="Audit"
  "RECORD"="1_2014-04-29T09:29:40"
  "TIMESTAMP"="2014-04-29T09:29:40 UTC"
  "MYSQL_VERSION"="5.6.17-65.0-655.trusty"
  "STARTUP_OPTIONS"="--basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib/mysql/plugin --user=
  "OS_VERSION"="x86_64-debian-linux-gnu",
/>
```

- **Connect/Disconnect** - Connect record event will have NAME field `Connect` when user logged in or login failed, or `Quit` when connection is closed. Additional fields for this event are `CONNECTION_ID`, `STATUS`, `USER`, `PRIV_USER`, `OS_LOGIN`, `PROXY_USER`, `HOST`, and `IP`. STATUS will be 0 for successful logins and non-zero for failed logins.

Example of the Disconnect event:

```
<AUDIT_RECORD
  "NAME"="Quit"
  "RECORD"="24_2014-04-29T09:29:40"
  "TIMESTAMP"="2014-04-29T10:20:13 UTC"
  "CONNECTION_ID"="49"
  "STATUS"="0"
  "USER"=""
  "PRIV_USER"=""
  "OS_LOGIN"=""
  "PROXY_USER"=""
  "HOST"=""
  "IP"=""
  "DB"=""
/>
```

- **Query** - Additional fields for this event are: `COMMAND_CLASS` (values come from the `com_status_vars` array in the `sql/mysqld.cc` file in a MySQL source distribution. Examples are `select`, `alter_table`, `create_table`, etc.), `CONNECTION_ID`, `STATUS` (indicates error when non-zero), `SQLTEXT` (text of SQL-statement), `USER`, `HOST`, `OS_USER`, `IP`. Possible values for the NAME name field for this event are `Query`, `Prepare`, `Execute`, `Change user`, etc.

Example of the Query event:

```
<AUDIT_RECORD
  "NAME"="Query"
  "RECORD"="23_2014-04-29T09:29:40"
  "TIMESTAMP"="2014-04-29T10:20:10 UTC"
  "COMMAND_CLASS"="select"
  "CONNECTION_ID"="49"
  "STATUS"="0"
  "SQLTEXT"="SELECT * from mysql.user"
  "USER"="root[root] @ localhost []"
  "HOST"="localhost"
  "OS_USER"=""
  "IP"=""
/>
```

7.14.1 Installation

Audit Log plugin is shipped with *Percona Server*, but it is not installed by default. To enable the plugin you must run the following command:

```
INSTALL PLUGIN audit_log SONAME 'audit_log.so';
```

You can check if the plugin is loaded correctly by running:

```
SHOW PLUGINS;
```

Audit log should be listed in the output:

```
+-----+-----+-----+-----+-----+
| Name                | Status | Type                | Library          | License |
+-----+-----+-----+-----+-----+
...
| audit_log           | ACTIVE | AUDIT               | audit_log.so    | GPL     |
+-----+-----+-----+-----+-----+
```

7.14.2 Log Format

The audit log plugin supports four log formats: OLD, NEW, JSON, and CSV. OLD and NEW formats are based on XML, where the former outputs log record properties as XML attributes and the latter as XML tags. Information logged is the same in all four formats. The log format choice is controlled by `audit_log_format` variable.

Example of the OLD format:

```
<AUDIT_RECORD
  "NAME"="Query"
  "RECORD"="2_2014-04-28T09:29:40"
  "TIMESTAMP"="2014-04-28T09:29:40 UTC"
  "COMMAND_CLASS"="install_plugin"
  "CONNECTION_ID"="47"
  "STATUS"="0"
  "SQLTEXT"="INSTALL PLUGIN audit_log SONAME 'audit_log.so'"
  "USER"="root[root] @ localhost []"
  "HOST"="localhost"
  "OS_USER"=""
  "IP"=""
/>
```

Example of the NEW format:

```
<AUDIT_RECORD>
<NAME>Quit</NAME>
<RECORD>10902_2014-04-28T11:02:54</RECORD>
<TIMESTAMP>2014-04-28T11:02:59 UTC</TIMESTAMP>
<CONNECTION_ID>36</CONNECTION_ID>
<STATUS>0</STATUS>
<USER></USER>
<PRIV_USER></PRIV_USER>
<OS_LOGIN></OS_LOGIN>
<PROXY_USER></PROXY_USER>
<HOST></HOST>
<IP></IP>
<DB></DB>
</AUDIT_RECORD>
```


Example of the JSON format:

```
{"audit_record":{"name":"Query","record":"4707_2014-08-27T10:43:52","timestamp":"2014-08-27T10:44:19"}}
```

Example of the CSV format:

```
"Query","49284_2014-08-27T10:47:11","2014-08-27T10:47:23 UTC","show_databases","37",0,"show databases"
```

7.14.3 Streaming the audit log to syslog

Ability to stream the audit log to syslog was implemented in *Percona Server 5.6.20-68.0*. To stream the audit log to syslog you'll need to set `audit_log_handler` variable to `SYSLOG`. To control the syslog file handler, the following variables can be used: `audit_log_syslog_ident`, `audit_log_syslog_facility`, and `audit_log_syslog_priority`. These variables have the same meaning as appropriate parameters described in the `syslog(3)` manual.

Note: Variables: `audit_log_strategy`, `audit_log_buffer_size`, `audit_log_rotate_on_size`, `audit_log_rotations` have effect only with `FILE` handler.

7.14.4 System Variables

variable `audit_log_strategy`

Version Info

- **5.6.17-65.0** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value ASYNCHRONOUS

Allowed values ASYNCHRONOUS, PERFORMANCE, SEMISYNCHRONOUS, SYNCHRONOUS

This variable is used to specify the audit log strategy, possible values are:

- `ASYNCHRONOUS` - (default) log using memory buffer, do not drop messages if buffer is full
- `PERFORMANCE` - log using memory buffer, drop messages if buffer is full
- `SEMISYNCHRONOUS` - log directly to file, do not flush and sync every event
- `SYNCHRONOUS` - log directly to file, flush and sync every event

This variable has effect only when `audit_log_handler` is set to `FILE`.

variable `audit_log_file`

Version Info

- **5.6.17-65.0** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value audit.log

This variable is used to specify the filename that's going to store the audit log. It can contain the path relative to the datadir or absolute path.

variable audit_log_flush

Version Info

- 5.6.17-65.0 – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value OFF

When this variable is set to ON log file will be closed and reopened. This can be used for manual log rotation.

variable audit_log_buffer_size

Version Info

- 5.6.17-65.0 – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 4096

This variable can be used to specify the size of memory buffer used for logging, used when `audit_log_strategy` variable is set to ASYNCHRONOUS or PERFORMANCE values. This variable has effect only when `audit_log_handler` is set to FILE.

variable audit_log_format

Version Info

- 5.6.17-65.0 – Original implementation
- 5.6.20-68.0 – Implemented support for CSV and JSON log formats

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value OLD

Allowed values OLD, NEW, CSV, JSON

This variable is used to specify the audit log format. The audit log plugin supports four log formats: OLD, NEW, JSON, and CSV. OLD and NEW formats are based on XML, where the former outputs log record properties as XML attributes and the latter as XML tags. Information logged is the same in all four formats.

variable audit_log_policy

Version Info

- 5.6.17-65.0 – Implemented

Command Line Yes**Scope** Global**Dynamic** Yes**Variable Type** String**Default Value** ALL**Allowed values** ALL, LOGINS, QUERIES, NONE

This variable is used to specify which events should be logged. Possible values are:

- ALL - all events will be logged
- LOGINS - only logins will be logged
- QUERIES - only queries will be logged
- NONE - no events will be logged

variable `audit_log_rotate_on_size`**Version Info**

- 5.6.17-65.0 – Implemented

Command Line Yes**Scope** Global**Dynamic** No**Variable Type** Numeric**Default Value** 0 (don't rotate the log file)

This variable is used to specify the maximum audit log file size. Upon reaching this size the log will be rotated. The rotated log files will be present in the same same directory as the current log file. A sequence number will be appended to the log file name upon rotation. This variable has effect only when `audit_log_handler` is set to FILE.

variable `audit_log_rotations`**Version Info**

- 5.6.17-65.0 – Implemented

Command Line Yes**Scope** Global**Dynamic** No**Variable Type** Numeric**Default Value** 0

This variable is used to specify how many log files should be kept when `audit_log_rotate_on_size` variable is set to non-zero value. This variable has effect only when `audit_log_handler` is set to FILE.

variable `audit_log_handler`**Version Info**

- 5.6.20-68.0 – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value FILE

Allowed values FILE, SYSLOG

This variable is used to configure where the audit log will be written. If it is set to `FILE`, the log will be written into a file specified by `audit_log_file` variable. If it is set to `SYSLOG`, the audit log will be written to `syslog`.

variable `audit_log_syslog_ident`

Version Info

- **5.6.20-68.0** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value `percona-audit`

This variable is used to specify the `ident` value for `syslog`. This variable has the same meaning as the appropriate parameter described in the [syslog\(3\) manual](#).

variable `audit_log_syslog_facility`

Version Info

- **5.6.20-68.0** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value `LOG_USER`

This variable is used to specify the `facility` value for `syslog`. This variable has the same meaning as the appropriate parameter described in the [syslog\(3\) manual](#).

variable `audit_log_syslog_priority`

Version Info

- **5.6.20-68.0** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value `LOG_INFO`

This variable is used to specify the `priority` value for `syslog`. This variable has the same meaning as the appropriate parameter described in the `syslog(3)` manual.

7.14.5 Version Specific Information

- 5.6.17-65.0 Audit Log plugin has been implemented in *Percona Server*.
- 5.6.20-68.0 *Percona Server Audit Log Plugin* now supports JSON and CSV log formats.
- 5.6.20-68.0 *Percona Server Audit Log Plugin* now supports *streaming the audit log to syslog*.

7.15 Start transaction with consistent snapshot

Percona Server, in 5.6.17-66.0, has ported *MariaDB* enhancement for `START TRANSACTION WITH CONSISTENT SNAPSHOTS` feature to *MySQL* 5.6 group commit implementation. This enhancement makes binary log positions consistent with *InnoDB* transaction snapshots.

This feature is quite useful to obtain logical backups with correct positions without running a `FLUSH TABLES WITH READ LOCK`. Binary log position can be obtained by two newly implemented status variables: `Binlog_snapshot_file` and `Binlog_snapshot_position`. After starting a transaction using the `START TRANSACTION WITH CONSISTENT SNAPSHOT`, these two variables will provide you with the binlog position corresponding to the state of the database of the consistent snapshot so taken, irrespectively of which other transactions have been committed since the snapshot was taken.

7.15.1 Snapshot Cloning

The *Percona Server* implementation extends the `START TRANSACTION WITH CONSISTENT SNAPSHOT` syntax with the optional `FROM SESSION` clause:

```
START TRANSACTION WITH CONSISTENT SNAPSHOT FROM SESSION <session_id>;
```

When specified, all participating storage engines and binary log instead of creating a new snapshot of data (or binary log coordinates), create a copy of the snapshot which has been created by an active transaction in the specified session. `session_id` is the session identifier reported in the `Id` column of `SHOW PROCESSLIST`.

Currently snapshot cloning is only supported by *XtraDB* and the binary log. As with the regular `START TRANSACTION WITH CONSISTENT SNAPSHOT`, snapshot clones can only be created with the `REPEATABLE READ` isolation level.

For *XtraDB*, a transaction with a cloned snapshot will only see data visible or changed by the donor transaction. That is, the cloned transaction will see no changes committed by transactions that started after the donor transaction, not even changes made by itself. Note that in case of chained cloning the donor transaction is the first one in the chain. For example, if transaction A is cloned into transaction B, which is in turn cloned into transaction C, the latter will have read view from transaction A (i.e. the donor transaction). Therefore, it will see changes made by transaction A, but not by transaction B.

7.15.2 mysqldump

`mysqldump` has been updated to use new status variables automatically when they are supported by the server and both `--single-transaction` and `--master-data` are specified on the command line. Along with the `mysqldump` improvements introduced in *Backup Locks* there is now a way to generate `mysqldump` backups that are guaranteed to be consistent without using `FLUSH TABLES WITH READ LOCK` even if `--master-data` is requested.

7.15.3 System Variables

variable `have_snapshot_cloning`

Version Info

- 5.6.16-64.0 – Implemented

Command Line Yes

Config File No

Scope Global

Dynamic No

Variable Type Boolean

This server variable is implemented to help other utilities detect if the server supports the `FROM SESSION` extension. When available, the snapshot cloning feature and the syntax extension to `START TRANSACTION WITH CONSISTENT SNAPSHOT` are supported by the server, and the variable value is always `YES`.

7.15.4 Status Variables

variable `Binlog_snapshot_file`

Version Info

- 5.6.17-66.0 – Implemented

Variable Type Numeric

variable `Binlog_snapshot_position`

Version Info

- 5.6.17-66.0 – Implemented

Variable Type Numeric

These status variables are only available when the binary log is enabled globally.

7.15.5 Other Reading

- [MariaDB Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT](#)

DIAGNOSTICS IMPROVEMENTS

8.1 User Statistics

This feature adds several `INFORMATION_SCHEMA` tables, several commands, and the `userstat` variable. The tables and commands can be used to understand the server activity better and identify the source of the load.

The functionality is disabled by default, and must be enabled by setting `userstat` to `ON`. It works by keeping several hash tables in memory. To avoid contention over global mutexes, each connection has its own local statistics, which are occasionally merged into the global statistics, and the local statistics are then reset to 0.

8.1.1 Version Specific Information

- **5.6.11-60.3:** Feature ported from *Percona Server 5.5*.

8.1.2 Other Information

- **Author/Origin:** *Google*; *Percona* added the `INFORMATION_SCHEMA` tables and the `userstat` variable.

8.1.3 System Variables

variable `userstat`

Version Info

- **5.6.11-60.3** – Variable introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type BOOLEAN

Default Value OFF

Range ON/OFF

Enables or disables collection of statistics. The default is `OFF`, meaning no statistics are gathered. This is to ensure that the statistics collection doesn't cause any extra load on the server unless desired.

variable `thread_statistics`

Version Info

- **5.6.11-60.3** – Feature ported from *Percona Server 5.5*

Command Line Yes**Config File** Yes**Scope** Global**Dynamic** Yes**Variable Type** BOOLEAN**Default Value** OFF**Range** ON/OFF

Enables or disables collection of thread statistics. The default is `OFF`, meaning no thread statistics are gathered. This is to ensure that the statistics collection doesn't cause any extra load on the server unless desired. Variable `userstat` needs to be enabled as well in order for thread statistics to be collected.

8.1.4 INFORMATION_SCHEMA Tables

table INFORMATION_SCHEMA.CLIENT_STATISTICS**Columns**

- **CLIENT** – The IP address or hostname from which the connection originated.
- **TOTAL_CONNECTIONS** – The number of connections created for this client.
- **CONCURRENT_CONNECTIONS** – The number of concurrent connections for this client.
- **CONNECTED_TIME** – The cumulative number of seconds elapsed while there were connections from this client.
- **BUSY_TIME** – The cumulative number of seconds there was activity on connections from this client.
- **CPU_TIME** – The cumulative CPU time elapsed, in seconds, while servicing this client's connections.
- **BYTES_RECEIVED** – The number of bytes received from this client's connections.
- **BYTES_SENT** – The number of bytes sent to this client's connections.
- **BINLOG_BYTES_WRITTEN** – The number of bytes written to the binary log from this client's connections.
- **ROWS_FETCHED** – The number of rows fetched by this client's connections.
- **ROWS_UPDATED** – The number of rows updated by this client's connections.
- **TABLE_ROWS_READ** – The number of rows read from tables by this client's connections. (It may be different from `ROWS_FETCHED`.)
- **SELECT_COMMANDS** – The number of `SELECT` commands executed from this client's connections.
- **UPDATE_COMMANDS** – The number of `UPDATE` commands executed from this client's connections.
- **OTHER_COMMANDS** – The number of other commands executed from this client's connections.

- **COMMIT_TRANSACTIONS** – The number of COMMIT commands issued by this client’s connections.
- **ROLLBACK_TRANSACTIONS** – The number of ROLLBACK commands issued by this client’s connections.
- **DENIED_CONNECTIONS** – The number of connections denied to this client.
- **LOST_CONNECTIONS** – The number of this client’s connections that were terminated uncleanly.
- **ACCESS_DENIED** – The number of times this client’s connections issued commands that were denied.
- **EMPTY_QUERIES** – The number of times this client’s connections sent empty queries to the server.

This table holds statistics about client connections. The Percona version of the feature restricts this table’s visibility to users who have the SUPER or PROCESS privilege.

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.CLIENT_STATISTICS\G
***** 1. row *****
      CLIENT: 10.1.12.30
    TOTAL_CONNECTIONS: 20
CONCURRENT_CONNECTIONS: 0
      CONNECTED_TIME: 0
        BUSY_TIME: 93
        CPU_TIME: 48
    BYTES_RECEIVED: 5031
    BYTES_SENT: 276926
BINLOG_BYTES_WRITTEN: 217
    ROWS_FETCHED: 81
    ROWS_UPDATED: 0
    TABLE_ROWS_READ: 52836023
    SELECT_COMMANDS: 26
    UPDATE_COMMANDS: 1
    OTHER_COMMANDS: 145
    COMMIT_TRANSACTIONS: 1
    ROLLBACK_TRANSACTIONS: 0
    DENIED_CONNECTIONS: 0
    LOST_CONNECTIONS: 0
      ACCESS_DENIED: 0
    EMPTY_QUERIES: 0
```

table INFORMATION_SCHEMA. **INDEX_STATISTICS**

Columns

- **TABLE_SCHEMA** – The schema (database) name.
- **TABLE_NAME** – The table name.
- **INDEX_NAME** – The index name (as visible in SHOW CREATE TABLE).
- **ROWS_READ** – The number of rows read from this index.

This table shows statistics on index usage. An older version of the feature contained a single column that had the TABLE_SCHEMA, TABLE_NAME and INDEX_NAME columns concatenated together. The *Percona* version of the feature separates these into three columns. Users can see entries only for tables to which they have SELECT access.

This table makes it possible to do many things that were difficult or impossible previously. For example, you can use it to find unused indexes and generate DROP commands to remove them.

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INDEX_STATISTICS
      WHERE TABLE_NAME='tables_priv';
+-----+-----+-----+-----+
| TABLE_SCHEMA | TABLE_NAME          | INDEX_NAME          | ROWS_READ |
+-----+-----+-----+-----+
| mysql         | tables_priv          | PRIMARY             |          2 |
+-----+-----+-----+-----+
```

Note: Current implementation of index statistics doesn't support partitioned tables.

table INFORMATION_SCHEMA.TABLE_STATISTICS

Columns

- **TABLE_SCHEMA** – The schema (database) name.
- **TABLE_NAME** – The table name.
- **ROWS_READ** – The number of rows read from the table.
- **ROWS_CHANGED** – The number of rows changed in the table.
- **ROWS_CHANGED_X_INDEXES** – The number of rows changed in the table, multiplied by the number of indexes changed.

This table is similar in function to the INDEX_STATISTICS table.

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TABLE_STATISTICS
      WHERE TABLE_NAME='tables_priv';
+-----+-----+-----+-----+-----+
| TABLE_SCHEMA | TABLE_NAME          | ROWS_READ | ROWS_CHANGED | ROWS_CHANGED_X_INDEXES |
+-----+-----+-----+-----+-----+
| mysql         | tables_priv          |          2 |             0 |                       0 |
+-----+-----+-----+-----+-----+
```

Note: Current implementation of table statistics doesn't support partitioned tables.

table INFORMATION_SCHEMA.THREAD_STATISTICS

Columns

- **THREAD_ID** – int(21)
- **TOTAL_CONNECTIONS** – int(21)
- **CONCURRENT_CONNECTIONS** – int(21)
- **CONNECTED_TIME** – int(21)
- **BUSY_TIME** – int(21)
- **CPU_TIME** – int(21)
- **BYTES_RECEIVED** – int(21)
- **BYTES_SENT** – int(21)
- **BINLOG_BYTES_WRITTEN** – int(21)

- **ROWS_FETCHED** – int(21)
- **ROWS_UPDATED** – int(21)
- **TABLE_ROWS_READ** – int(21)
- **SELECT_COMMANDS** – int(21)
- **UPDATE_COMMANDS** – int(21)
- **OTHER_COMMANDS** – int(21)
- **COMMIT_TRANSACTIONS** – int(21)
- **ROLLBACK_TRANSACTIONS** – int(21)
- **DENIED_CONNECTIONS** – int(21)
- **LOST_CONNECTIONS** – int(21)
- **ACCESS_DENIED** – int(21)
- **EMPTY_QUERIES** – int(21)

table INFORMATION_SCHEMA.USER_STATISTICS

Columns

- **USER** – The username. The value #mysql_system_user# appears when there is no username (such as for the slave SQL thread).
- **TOTAL_CONNECTIONS** – The number of connections created for this user.
- **CONCURRENT_CONNECTIONS** – The number of concurrent connections for this user.
- **CONNECTED_TIME** – The cumulative number of seconds elapsed while there were connections from this user.
- **BUSY_TIME** – The cumulative number of seconds there was activity on connections from this user.
- **CPU_TIME** – The cumulative CPU time elapsed, in seconds, while servicing this user's connections.
- **BYTES_RECEIVED** – The number of bytes received from this user's connections.
- **BYTES_SENT** – The number of bytes sent to this user's connections.
- **BINLOG_BYTES_WRITTEN** – The number of bytes written to the binary log from this user's connections.
- **ROWS_FETCHED** – The number of rows fetched by this user's connections.
- **ROWS_UPDATED** – The number of rows updated by this user's connections.
- **TABLE_ROWS_READ** – The number of rows read from tables by this user's connections. (It may be different from ROWS_FETCHED.)
- **SELECT_COMMANDS** – The number of SELECT commands executed from this user's connections.
- **UPDATE_COMMANDS** – The number of UPDATE commands executed from this user's connections.
- **OTHER_COMMANDS** – The number of other commands executed from this user's connections.

- **COMMIT_TRANSACTIONS** – The number of COMMIT commands issued by this user’s connections.
- **ROLLBACK_TRANSACTIONS** – The number of ROLLBACK commands issued by this user’s connections.
- **DENIED_CONNECTIONS** – The number of connections denied to this user.
- **LOST_CONNECTIONS** – The number of this user’s connections that were terminated uncleanly.
- **ACCESS_DENIED** – The number of times this user’s connections issued commands that were denied.
- **EMPTY_QUERIES** – The number of times this user’s connections sent empty queries to the server.

This table contains information about user activity. The *Percona* version of the patch restricts this table’s visibility to users who have the SUPER or PROCESS privilege.

The table gives answers to questions such as which users cause the most load, and whether any users are being abusive. It also lets you measure how close to capacity the server may be. For example, you can use it to find out whether replication is likely to start falling behind.

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.USER_STATISTICS\G
***** 1. row *****
      USER: root
TOTAL_CONNECTIONS: 5592
CONCURRENT_CONNECTIONS: 0
   CONNECTED_TIME: 6844
      BUSY_TIME: 179
      CPU_TIME: 72
   BYTES_RECEIVED: 603344
   BYTES_SENT: 15663832
BINLOG_BYTES_WRITTEN: 217
   ROWS_FETCHED: 9793
   ROWS_UPDATED: 0
TABLE_ROWS_READ: 52836023
SELECT_COMMANDS: 9701
UPDATE_COMMANDS: 1
   OTHER_COMMANDS: 2614
COMMIT_TRANSACTIONS: 1
ROLLBACK_TRANSACTIONS: 0
   DENIED_CONNECTIONS: 0
   LOST_CONNECTIONS: 0
   ACCESS_DENIED: 0
   EMPTY_QUERIES: 0
```

8.1.5 Commands Provided

- FLUSH CLIENT_STATISTICS
- FLUSH INDEX_STATISTICS
- FLUSH TABLE_STATISTICS
- FLUSH THREAD_STATISTICS
- FLUSH USER_STATISTICS

These commands discard the specified type of stored statistical information.

- SHOW CLIENT_STATISTICS
- SHOW INDEX_STATISTICS
- SHOW TABLE_STATISTICS
- SHOW THREAD_STATISTICS
- SHOW USER_STATISTICS

These commands are another way to display the information you can get from the INFORMATION_SCHEMA tables. The commands accept WHERE clauses. They also accept but ignore LIKE clauses.

8.2 Slow Query Log

This feature adds microsecond time resolution and additional statistics to the slow query log output. It lets you enable or disable the slow query log at runtime, adds logging for the slave SQL thread, and adds fine-grained control over what and how much to log into the slow query log.

You can use *Percona-Toolkit's* `pt-query-digest` tool to aggregate similar queries together and report on those that consume the most execution time.

8.2.1 Version Specific Information

- 5.6.11-60.3:
 - Feature ported from *Percona Server 5.5*.
- 5.6.13-60.4:
 - New `slow_query_log_always_write_time` variable introduced

8.2.2 Other Information

- Author / Origin: Maciej Dobrzanski

8.2.3 System Variables

variable `log_slow_filter`

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Filters the slow log by the query's execution plan. The value is a comma-delimited string, and can contain any combination of the following values:

- `qc_miss`: The query was not found in the query cache.
- `full_scan`: The query performed a full table scan.
- `full_join`: The query performed a full join (a join without indexes).

- `tmp_table`: The query created an implicit internal temporary table.
- `tmp_table_on_disk`: The query's temporary table was stored on disk.
- `filesort`: The query used a filesort.
- `filesort_on_disk`: The filesort was performed on disk.

Values are OR'ed together. If the string is empty, then the filter is disabled. If it is not empty, then queries will only be logged to the slow log if their execution plan matches one of the types of plans present in the filter.

For example, to log only queries that perform a full table scan, set the value to `full_scan`. To log only queries that use on-disk temporary storage for intermediate results, set the value to `tmp_table_on_disk, filesort_on_disk`.

variable `log_slow_rate_type`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Enumerated

Default Value `session`

Range `session, query`

Specifies semantic of `log_slow_rate_limit` - `session` or `query`.

variable `log_slow_rate_limit`

Command Line Yes

Config File Yes

Scope Global, session

Dynamic Yes

Default Value 1

Range 1-1000

Behavior of this variable depends from `log_slow_rate_type`.

Specifies that only a fraction of `session/query` should be logged. Logging is enabled for every `n`th `session/query`. By default, `n` is 1, so logging is enabled for every `session/query`. Please note: when `log_slow_rate_type` is `session` rate limiting is disabled for the replication thread.

Logging all queries might consume I/O bandwidth and cause the log file to grow large.

- When `log_slow_rate_type` is `session`, this option lets you log full sessions, so you have complete records of sessions for later analysis; but you can rate-limit the number of sessions that are logged. Note that this feature will not work well if your application uses any type of connection pooling or persistent connections. Note that you change `log_slow_rate_limit` in `session` mode, you should reconnect for get effect.
- When `log_slow_rate_type` is `query`, this option lets you log just some queries for later analysis. For example, if you set the value to 100, then one percent of queries will be logged.

Note that every query has global unique `query_id` and every connection can has it own (session) `log_slow_rate_limit`. Decision "log or no" calculated in following manner:

- if `log_slow_rate_limit` is 1 - log every query

- If `log_slow_rate_limit > 1` - randomly log every $1/\text{log_slow_rate_limit}$ query.

This allows flexible setup logging behavior.

For example, if you set the value to 100, then one percent of `sessions/queries` will be logged. In *Percona Server 5.6.13-60.6* information about the `log_slow_rate_limit` has been added to the slow query log. This means that if the `log_slow_rate_limit` is effective it will be reflected in the slow query log for each written query. Example of the output looks like this:

```
Log_slow_rate_type: query Log_slow_rate_limit: 10
```

Prior to 5.6.17-65.0 implementation of the `log_slow_rate_type` set to `query` with `log_slow_rate_limit` feature would log every `n`th query deterministically. With the current implementation each query has a non-deterministic probability of $1/n$ to get logged.

variable `log_slow_sp_statements`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value TRUE

Range TRUE/FALSE

If TRUE, statements executed by stored procedures are logged to the slow if it is open.

Note: Support for logging stored procedures doesn't involve triggers, so they won't be logged even if this feature is enabled.

variable `log_slow_verbosity`

Command Line Yes

Config File Yes

Scope Global, session

Dynamic Yes

Specifies how much information to include in your slow log. The value is a comma-delimited string, and can contain any combination of the following values:

- `microtime`: Log queries with microsecond precision.
- `query_plan`: Log information about the query's execution plan.
- `innodb`: Log *InnoDB* statistics.
- `minimal`: Equivalent to enabling just `microtime`.
- `standard`: Equivalent to enabling `microtime, innodb`.
- `full`: Equivalent to all other values OR'ed together without the `profiling` and `profiling_use_getrusage` options.
- `profiling`: Enables profiling of all queries in all connections.
- `profiling_use_getrusage`: Enables usage of the `getrusage` function.

Values are OR'ed together.

For example, to enable microsecond query timing and *InnoDB* statistics, set this option to `microtime`, `innodb` or `standard`. To turn all options on, set the option to `full`.

variable `slow_query_log_timestamp_always`

Command Line Yes
Config File Yes
Scope Global
Dynamic Yes
Variable Type Boolean
Default Value FALSE
Range TRUE/FALSE

If TRUE, a timestamp is printed on every slow log record. Multiple records may have the same time.

variable `slow_query_log_timestamp_precision`

Command Line Yes
Config File Yes
Scope Global
Dynamic Yes
Variable Type Enumerated
Default Value `second`
Range `second, microsecond`

Normally, entries to the slow query log are in seconds precision, in this format:

```
# Time: 090402 9:23:36 # User@Host: XXX @ XXX [10.X.X.X]
```

If `slow_query_log_timestamp_precision=microsecond`, entries to the slow query log are in microsecond precision, in this format:

```
# Time: 090402 9:23:36.123456 # User@Host: XXX @ XXX [10.X.X.X]
```

variable `slow_query_log_use_global_control`

Command Line Yes
Config File Yes
Scope Global
Dynamic Yes
Default Value None

Specifies which variables have global scope instead of local. Value is a “flag” variable - you can specify multiple values separated by commas

- `none`: All variables use local scope
- `log_slow_filter`: Global variable `log_slow_filter` has effect (instead of local)
- `log_slow_rate_limit`: Global variable `log_slow_rate_limit` has effect (instead of local)
- `log_slow_verbosity`: Global variable `log_slow_verbosity` has effect (instead of local)

- `long_query_time`: Global variable `long_query_time` has effect (instead of local)
- `min_examined_row_limit`: Global variable `min_examined_row_limit` has effect (instead of local)
- all Global variables has effect (instead of local)

variable `slow_query_log_always_write_time`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value 10

This variable can be used to specify the query execution time after which the query will be written to the slow query log. It can be used to specify an additional execution time threshold for the slow query log, that, when exceeded, will cause a query to be logged unconditionally, that is, `log_slow_rate_limit` will not apply to it.

8.2.4 Other Information

Changes to the Log Format

The feature adds more information to the slow log output. Here is a sample log entry:

```
# Time: 130601 8:01:06.058915
# User@Host: root[root] @ localhost [] Id: 42
# Schema: imdb Last_errno: 0 Killed: 0
# Query_time: 7.725616 Lock_time: 0.000328 Rows_sent: 4 Rows_examined: 1543720 Rows_affected: 0
# Bytes_sent: 272 Tmp_tables: 0 Tmp_disk_tables: 0 Tmp_table_sizes: 0
# QC_Hit: No Full_scan: Yes Full_join: No Tmp_table: No Tmp_table_on_disk: No
# Filesort: No Filesort_on_disk: No Merge_passes: 0
SET timestamp=1370073666;
SELECT id,title,production_year FROM title WHERE title = 'Bambi';
```

Another example (`log_slow_verbosity=profiling`):

```
# Time: 130601 8:03:20.700441
# User@Host: root[root] @ localhost [] Id: 43
# Schema: imdb Last_errno: 0 Killed: 0
# Query_time: 7.815071 Lock_time: 0.000261 Rows_sent: 4 Rows_examined: 1543720 Rows_affected: 0
# Bytes_sent: 272
# Profile_starting: 0.000125 Profile_starting_cpu: 0.000120
Profile_checking_permissions: 0.000021 Profile_checking_permissions_cpu: 0.000021
Profile_Opening_tables: 0.000049 Profile_Opening_tables_cpu: 0.000048 Profile_init: 0.000048
Profile_init_cpu: 0.000049 Profile_System_lock: 0.000049 Profile_System_lock_cpu: 0.000048
Profile_optimizing: 0.000024 Profile_optimizing_cpu: 0.000024 Profile_statistics: 0.000036
Profile_statistics_cpu: 0.000037 Profile_preparing: 0.000029 Profile_preparing_cpu: 0.000029
Profile_executing: 0.000012 Profile_executing_cpu: 0.000012 Profile_Sending_data: 7.814583
Profile_Sending_data_cpu: 7.811634 Profile_end: 0.000013 Profile_end_cpu: 0.000012
Profile_query_end: 0.000014 Profile_query_end_cpu: 0.000014 Profile_closing_tables: 0.000023
Profile_closing_tables_cpu: 0.000023 Profile_freeing_items: 0.000051
Profile_freeing_items_cpu: 0.000050 Profile_logging_slow_query: 0.000006
Profile_logging_slow_query_cpu: 0.000006
# Profile_total: 7.815085 Profile_total_cpu: 7.812127
SET timestamp=1370073800;
SELECT id,title,production_year FROM title WHERE title = 'Bambi';
```

Connection and Schema Identifier

Each slow log entry now contains a connection identifier, so you can trace all the queries coming from a single connection. This is the same value that is shown in the Id column in `SHOW FULL PROCESSLIST` or returned from the `CONNECTION_ID()` function.

Each entry also contains a schema name, so you can trace all the queries whose default database was set to a particular schema.

```
# Id: 43 Schema: imdb
```

Microsecond Time Resolution and Extra Row Information

This is the original functionality offered by the `microslow` feature. `Query_time` and `Lock_time` are logged with microsecond resolution.

The feature also adds information about how many rows were examined for `SELECT` queries, and how many were analyzed and affected for `UPDATE`, `DELETE`, and `INSERT` queries,

```
# Query_time: 0.962742 Lock_time: 0.000202 Rows_sent: 4 Rows_examined: 1543719 Rows_affected: 0
```

Values and context:

- `Rows_examined`: Number of rows scanned - `SELECT`
- `Rows_affected`: Number of rows changed - `UPDATE`, `DELETE`, `INSERT`

Memory Footprint

The feature provides information about the amount of bytes sent for the result of the query and the number of temporary tables created for its execution - differentiated by whether they were created on memory or on disk - with the total number of bytes used by them.

```
# Bytes_sent: 8053 Tmp_tables: 1 Tmp_disk_tables: 0 Tmp_table_sizes: 950528
```

Values and context:

- `Bytes_sent`: The amount of bytes sent for the result of the query
- `Tmp_tables`: Number of temporary tables created on memory for the query
- `Tmp_disk_tables`: Number of temporary tables created on disk for the query
- `Tmp_table_sizes`: Total Size in bytes for all temporary tables used in the query

Query Plan Information

Each query can be executed in various ways. For example, it may use indexes or do a full table scan, or a temporary table may be needed. These are the things that you can usually see by running `EXPLAIN` on the query. The feature will now allow you to see the most important facts about the execution in the log file.

```
# QC_Hit: No Full_scan: Yes Full_join: No Tmp_table: No Tmp_table_on_disk: No  
# Filesort: No Filesort_on_disk: No Merge_passes: 0
```

The values and their meanings are documented with the `log_slow_filter` option.

InnoDB Usage Information

The final part of the output is the *InnoDB* usage statistics. *MySQL* currently shows many per-session statistics for operations with `SHOW SESSION STATUS`, but that does not include those of *InnoDB*, which are always global and shared by all threads. This feature lets you see those values for a given query.

```
# InnoDB_IO_r_ops: 6415 InnoDB_IO_r_bytes: 105103360 InnoDB_IO_r_wait: 0.001279
# InnoDB_rec_lock_wait: 0.000000 InnoDB_queue_wait: 0.000000
# InnoDB_pages_distinct: 6430
```

Values:

- `innodb_IO_r_ops`: Counts the number of page read operations scheduled. The actual number of read operations may be different, but since this can be done asynchronously, there is no good way to measure it.
- `innodb_IO_r_bytes`: Similar to `innodb_IO_r_ops`, but the unit is bytes.
- `innodb_IO_r_wait`: Shows how long (in seconds) it took *InnoDB* to actually read the data from storage.
- `innodb_rec_lock_wait`: Shows how long (in seconds) the query waited for row locks.
- `innodb_queue_wait`: Shows how long (in seconds) the query spent either waiting to enter the *InnoDB* queue or inside that queue waiting for execution.
- `innodb_pages_distinct`: Counts approximately the number of unique pages the query accessed. The approximation is based on a small hash array representing the entire buffer pool, because it could take a lot of memory to map all the pages. The inaccuracy grows with the number of pages accessed by a query, because there is a higher probability of hash collisions.

If the query did not use *InnoDB* tables, that information is written into the log instead of the above statistics.

8.2.5 Related Reading

- [Impact of logging on MySQL's performance](#)
- [log_slow_filter Usage](#)
- [Blueprint in Launchpad](#)

8.3 Extended Show Engine *InnoDB* Status

This feature reorganizes the output of `SHOW ENGINE INNODB STATUS` for a better readability and prints the amount of memory used by the internal hash tables. In addition, new variables are available to control the output.

This feature modified the `SHOW ENGINE INNODB STATUS` command as follows:

- `TRANSACTION` section was moved to the end of the output, so that important information is not overlooked when there is a large amount of it.
- Added two variables to control `SHOW ENGINE INNODB STATUS` information presented (bugfix for #29123):
 - `innodb_show_verbose_locks` - Whether to show records locked
 - `innodb_show_locks_held` - Number of locks held to print for each *InnoDB* transaction
- Added extended information about *InnoDB* internal hash table sizes (in bytes) in the `BUFFER POOL AND MEMORY` section; also added buffer pool size in bytes.
- Added additional `LOG` section information.

8.3.1 Version Specific Information

- 5.6.11-60.3:
Feature ported from *Percona Server 5.5*.

8.3.2 Other Information

- Author / Origin: Baron Schwartz, <http://lists.mysql.com/internals/35174>

8.3.3 System Variables

variable `innodb_show_verbose_locks`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 0

Range 0 - 1

Specifies to show records locked in `SHOW ENGINE INNODB STATUS`. The default is 0, which means only the higher-level information about the lock (which table and index is locked, etc.) is printed. If set to 1, then traditional *InnoDB* behavior is enabled: the records that are locked are dumped to the output.

variable `innodb_show_locks_held`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 10

Range 0 - 1000

Specifies the number of locks held to print for each *InnoDB* transaction in `SHOW ENGINE INNODB STATUS`.

8.3.4 Status Variables

The status variables here contain information available in the output of `SHOW ENGINE INNODB STATUS`, organized by the sections `SHOW ENGINE INNODB STATUS` displays. If you are familiar with the output of `SHOW ENGINE INNODB STATUS`, you will probably already recognize the information these variables contain.

BACKGROUND THREAD

The following variables contain information in the BACKGROUND THREAD section of the output from SHOW ENGINE INNODB STATUS. An example of that output is:

```
-----
BACKGROUND THREAD
-----
srv_master_thread loops: 1 srv_active, 0 srv_shutdown, 11844 srv_idle
srv_master_thread log flush and writes: 11844
```

variable `Innodb_master_thread_active_loops`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_master_thread_idle_loops`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_background_log_sync`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

SEMAPHORES

The following variables contain information in the SEMAPHORES section of the output from SHOW ENGINE INNODB STATUS. An example of that output is:

```
-----
SEMAPHORES
-----
OS WAIT ARRAY INFO: reservation count 9664, signal count 11182
Mutex spin waits 20599, rounds 223821, OS waits 4479
RW-shared spins 5155, OS waits 1678; RW-excl spins 5632, OS waits 2592
Spin rounds per wait: 10.87 mutex, 15.01 RW-shared, 27.19 RW-excl
```

variable `Innodb_mutex_os_waits`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_mutex_spin_rounds`

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_mutex_spin_waits

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_s_lock_os_waits

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_s_lock_spin_rounds

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_s_lock_spin_waits

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_x_lock_os_waits

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_x_lock_spin_rounds

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_x_lock_spin_waits

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

INSERT BUFFER AND ADAPTIVE HASH INDEX

The following variables contain information in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the output from SHOW ENGINE INNODB STATUS. An example of that output is:

```
-----
INSERT BUFFER AND ADAPTIVE HASH INDEX
-----
Ibuf: size 1, free list len 6089, seg size 6091,
44497 inserts, 44497 merged recs, 8734 merges
0.00 hash searches/s, 0.00 non-hash searches/s
```

variable `Innodb_ibuf_discarded_delete_marks`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_discarded_deletes`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_discarded_inserts`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_free_list`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_merged_delete_marks`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_ibuf_merged_deletes

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_ibuf_merged_inserts

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_ibuf_merges

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_ibuf_segment_size

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_ibuf_size

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

LOG

The following variables contain information in the LOG section of the output from SHOW ENGINE INNODB STATUS. An example of that output is:

```
LOG
---
Log sequence number 10145937666
Log flushed up to   10145937666
Pages flushed up to 10145937666
Last checkpoint at  10145937666
Max checkpoint age  80826164
Checkpoint age target 78300347
Modified age        0
Checkpoint age      0
0 pending log writes, 0 pending chkp writes
```



```

9 log i/o's done, 0.00 log i/o's/second
Log tracking enabled
Log tracked up to 10145937666
Max tracked LSN age 80826164

```

variable InnoDB_lsn_current**Version Info**

- **5.6.11-60.3** – Introduced.

Variable Type Numeric**Scope** Global**variable InnoDB_lsn_flushed****Version Info**

- **5.6.11-60.3** – Introduced.

Variable Type Numeric**Scope** Global**variable InnoDB_lsn_last_checkpoint****Version Info**

- **5.6.11-60.3** – Introduced.

Variable Type Numeric**Scope** Global**variable InnoDB_checkpoint_age****Version Info**

- **5.6.11-60.3** – Introduced.

Variable Type Numeric**Scope** Global**variable InnoDB_checkpoint_max_age****Version Info**

- **5.6.11-60.3** – Introduced.

Variable Type Numeric**Scope** Global**BUFFER POOL AND MEMORY**

The following variables contain information in the BUFFER POOL AND MEMORY section of the output from SHOW ENGINE INNODB STATUS. An example of that output is:

```

-----
BUFFER POOL AND MEMORY
-----
Total memory allocated 137363456; in additional pool allocated 0
Total memory allocated by read views 88
Internal hash tables (constant factor + variable factor)

```

```
Adaptive hash index 2266736          (2213368 + 53368)
Page hash           139112 (buffer pool 0 only)
Dictionary cache    729463 (554768 + 174695)
File system         824800 (812272 + 12528)
Lock system         333248 (332872 + 376)
Recovery system     0          (0 + 0)
Dictionary memory allocated 174695
Buffer pool size    8191
Buffer pool size, bytes 134201344
Free buffers        7481
Database pages      707
Old database pages  280
Modified db pages   0
Pending reads 0
Pending writes: LRU 0, flush list 0 single page 0
Pages made young 0, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 707, created 0, written 1
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
No buffer pool page gets since the last printout
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 707, unzip_LRU len: 0
```

variable `InnoDB_mem_adaptive_hash`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `InnoDB_mem_dictionary`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `InnoDB_mem_total`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `InnoDB_buffer_pool_pages_LRU_flushed`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `InnoDB_buffer_pool_pages_made_not_young`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `InnoDB_buffer_pool_pages_made_young`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `InnoDB_buffer_pool_pages_old`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

variable `InnoDB_descriptors_memory`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

This status variable shows the current size of the descriptors array (in bytes). The descriptor array is an *XtraDB* data structure that contains the information on currently running transactions.

variable `InnoDB_read_views_memory`

Version Info

- **5.6.11-60.3** – Introduced.

Variable Type Numeric

Scope Global

This status variable shows the total amount of memory allocated for the *InnoDB* read view (in bytes).

TRANSACTIONS

The following variables contain information in the TRANSACTIONS section of the output from `SHOW INNODB STATUS`. An example of that output is:

```
-----
TRANSACTIONS
-----
Trx id counter F561FD
Purge done for trx's n:o < F561EB undo n:o < 0
History list length 19
LIST OF TRANSACTIONS FOR EACH SESSION:
---TRANSACTION 0, not started, process no 993, OS thread id 140213152634640
mysql thread id 15933, query id 32109 localhost root
show innodb status
```

```
---TRANSACTION F561FC, ACTIVE 29 sec, process no 993, OS thread id 140213152769808 updating or deleting  
mysql tables in use 1, locked 1
```

variable Innodb_history_list_length

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_max_trx_id

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_oldest_view_low_limit_trx_id

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_purge_trx_id

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_purge_undo_no

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

variable Innodb_current_row_locks

Version Info

- 5.6.11-60.3 – Introduced.

Variable Type Numeric

Scope Global

8.3.5 INFORMATION_SCHEMA Tables

The following table contains information about the oldest active transaction in the system.

table INFORMATION_SCHEMA.XTRADB_READ_VIEW

Columns

- **READ_VIEW_UNDO_NUMBER** –
- **READ_VIEW_LOW_LIMIT_TRX_NUMBER** – This is the highest transactions number at the time the view was created.
- **READ_VIEW_UPPER_LIMIT_TRX_ID** – This is the highest transactions ID at the time the view was created. This means that it should not see newer transactions with IDs bigger than or equal to that value.
- **READ_VIEW_LOW_LIMIT_TRX_ID** – This is the latest committed transaction ID at the time the oldest view was created. This means that it should see all transactions with IDs smaller than or equal to that value.

The following table contains information about the memory usage for InnoDB/XtraDB hash tables.

table INFORMATION_SCHEMA.XTRADB_INTERNAL_HASH_TABLES

Columns

- **INTERNAL_HASH_TABLE_NAME** – Hash table name
- **TOTAL_MEMORY** – Total amount of memory
- **CONSTANT_MEMORY** – Constant memory
- **VARIABLE_MEMORY** – Variable memory

8.3.6 Other reading

- [SHOW INNODB STATUS walk through](#)
- [Table locks in SHOW INNODB STATUS](#)

8.4 Count *InnoDB* Deadlocks

When running a transactional application you have to live with deadlocks. They are not problematic as long as they do not occur too frequently. The standard `SHOW INNODB STATUS` gives information on the latest deadlocks but it is not very useful when you want to know the total number of deadlocks or the number of deadlocks per unit of time.

This change adds a status variable that keeps track of the number of deadlocks since the server startup, opening the way to a better knowledge of your deadlocks.

This feature was provided by Eric Bergen under BSD license (see [InnoDB Deadlock Count Patch](#)).

It adds a new global status variable (`innodb_deadlocks`) showing the number of deadlocks.*

You can use it with `SHOW GLOBAL STATUS`, e.g.:

```
mysql> SHOW GLOBAL STATUS LIKE 'innodb_deadlocks';
+-----+-----+
| Variable_name | Value |
+-----+-----+
```

```
| innodb_deadlocks | 323 |
+-----+-----+
```

or with INFORMATION_SCHEMA, e.g.:

```
mysql> SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS WHERE VARIABLE_NAME = 'innodb_dea
+-----+
| VARIABLE_VALUE |
+-----+
| 323            |
+-----+
```

A deadlock will occur when at least two transactions are mutually waiting for the other to finish, thus creating a circular dependency that lasts until something breaks it. *InnoDB* is quite good at detecting deadlocks and generally returns an error instantly. Most transactional systems have no way to prevent deadlocks from occurring and must be designed to handle them, for instance by retrying the transaction that failed.

8.4.1 Version Specific Information

- 5.6.11-60.3: Full functionality available.

8.4.2 Status Variables

One new status variable was introduced by this feature.

variable Innodb_deadlocks

Variable Type LONG

Scope Global

8.4.3 Related Reading

- [Original post by Eric Bergen](#)

8.5 Log All Client Commands (syslog)

When enabled, this feature causes all commands run by the command line client to be logged to syslog. If you want to enable this option permanently, add it to the [mysql] group in my.cnf.

8.5.1 Version Specific Information

- 5.6.11-60.3: Feature ported from *Percona Server 5.5*.

8.5.2 Other Information

- Author / Origin: Percona

8.5.3 Client Variables

variable `syslog`

Command Line Yes

Config File Yes

Server No

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value OFF

Range ON/OFF

The variable enables (ON)/disables (OFF) logging to syslog.

8.5.4 Other Reading

- <http://en.wikipedia.org/wiki/Syslog>
- <http://tools.ietf.org/html/rfc5424>

8.6 Show Storage Engines

This feature changes the comment field displayed when the `SHOW STORAGE ENGINES` command is executed and *XtraDB* is the storage engine.

Before the Change:

```
mysql> show storage engines;
+-----+-----+-----+-----+
| Engine      | Support | Comment                                     | Transaction |
+-----+-----+-----+-----+
| InnoDB      | YES     | Supports transactions, row-level locking, and foreign keys | YES         |
...
+-----+-----+-----+-----+
```

After the Change:

```
mysql> show storage engines;
+-----+-----+-----+-----+
| Engine      | Support | Comment                                     | Transaction |
+-----+-----+-----+-----+
| InnoDB      | YES     | Percona-XtraDB, Supports transactions, row-level locking, and foreign keys | YES         |
...
+-----+-----+-----+-----+
```

8.6.1 Version-Specific Information

- 5.6.11-60.3: Full functionality available.

8.7 Show Lock Names

This feature is currently undocumented except for the following example.

Example:

```
mysql> SHOW ENGINE INNODB MUTEX;
+-----+-----+-----+
| Type   | Name                               | Status           |
+-----+-----+-----+
| InnoDB | &rseg->mutex                       | os_waits=210    |
| InnoDB | &dict_sys->mutex                   | os_waits=3      |
| InnoDB | &trx_doublewrite->mutex           | os_waits=1      |
| InnoDB | &log_sys->mutex                    | os_waits=1197   |
| InnoDB | &LRU_list_mutex                   | os_waits=2      |
| InnoDB | &fil_system->mutex                 | os_waits=5      |
| InnoDB | &kernel_mutex                     | os_waits=242    |
| InnoDB | &new_index->lock                   | os_waits=2      |
| InnoDB | &new_index->lock                   | os_waits=415    |
.....
```

8.8 Process List

This page describes Percona changes to both the standard *MySQL* `SHOW PROCESSLIST` command and the standard *MySQL* `INFORMATION_SCHEMA` table `PROCESSLIST`.

The changes that have been made as of version 5.6 of the server are:

- `PROCESSLIST` table:
 - added column `TIME_MS`

8.8.1 Version Specific Information

- 5.6.5-60.0:
 - Added column `TIME_MS` to table `PROCESSLIST`.
- 5.6.11-60.3:
 - Added `ROWS_SENT` and `ROWS_EXAMINED` columns to table `PROCESSLIST`.

8.8.2 INFORMATION_SCHEMA Tables

table `INFORMATION_SCHEMA.PROCESSLIST`

This table implements modifications to the standard *MySQL* `INFORMATION_SCHEMA` table `PROCESSLIST`.

Columns

- **ID** – The connection identifier.
- **USER** – The *MySQL* user who issued the statement.
- **HOST** – The host name of the client issuing the statement.
- **DB** – The default database, if one is selected, otherwise `NULL`.
- **COMMAND** – The type of command the thread is executing.

- **TIME** – The time in seconds that the thread has been in its current state.
- **STATE** – An action, event, or state that indicates what the thread is doing.
- **INFO** – The statement that the thread is executing, or NULL if it is not executing any statement.
- **TIME_MS** – The time in milliseconds that the thread has been in its current state.
- **ROWS_SENT** – The number of rows query sent.
- **ROWS_EXAMINED** – The number of rows scanned for a query.

8.8.3 Example Output

Table `PROCESSLIST`:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST;
```

ID	USER	HOST	DB	COMMAND	TIME	STATE	INFO
12	root	localhost	information_schema	Query	0	executing	select * from processlist

8.9 Misc. INFORMATION_SCHEMA Tables

This page lists the `INFORMATION_SCHEMA` tables added to standard *MySQL* by *Percona Server* that don't exist elsewhere in the documentation.

8.9.1 Temporary tables

Note: This feature implementation is considered ALPHA quality.

Only the temporary tables that were explicitly created with `CREATE TEMPORARY TABLE` or `ALTER TABLE` are shown, and not the ones created to process complex queries.

table `INFORMATION_SCHEMA.GLOBAL_TEMPORARY_TABLES`

Version Info

- **5.6.5-60.0** – Feature introduced

Columns

- **SESSION_ID** – *MySQL* connection id
- **TABLE_SCHEMA** – Schema in which the temporary table is created
- **TABLE_NAME** – Name of the temporary table
- **ENGINE** – Engine of the temporary table
- **NAME** – Internal name of the temporary table
- **TABLE_ROWS** – Number of rows of the temporary table
- **AVG_ROW_LENGTH** – Average row length of the temporary table

- **DATA_LENGTH** – Size of the data (Bytes)
- **INDEX_LENGTH** – Size of the indexes (Bytes)
- **CREATE_TIME** – Date and time of creation of the temporary table
- **UPDATE_TIME** – Date and time of the latest update of the temporary table

This table holds information on the temporary tables existing for all connections. You don't need the `SUPER` privilege to query this table.

table `INFORMATION_SCHEMA.TEMPORARY_TABLES`

Version Info

- **5.6.5-60.0** – Feature introduced

Columns

- **SESSION_ID** – *MySQL* connection id
- **TABLE_SCHEMA** – Schema in which the temporary table is created
- **TABLE_NAME** – Name of the temporary table
- **ENGINE** – Engine of the temporary table
- **NAME** – Internal name of the temporary table
- **TABLE_ROWS** – Number of rows of the temporary table
- **AVG_ROW_LENGTH** – Average row length of the temporary table
- **DATA_LENGTH** – Size of the data (Bytes)
- **INDEX_LENGTH** – Size of the indexes (Bytes)
- **CREATE_TIME** – Date and time of creation of the temporary table
- **UPDATE_TIME** – Date and time of the latest update of the temporary table

This table holds information on the temporary tables existing for the running connection.

8.9.2 Multiple Rollback Segments

Percona Server, in addition to the upstream multiple rollback segment implementation, provides the additional Information Schema table: `INFORMATION_SCHEMA.XTRADB_RSEG`.

8.9.3 INFORMATION_SCHEMA Tables

This feature provides the following table:

table `INFORMATION_SCHEMA.XTRADB_RSEG`

Columns

- **rseg_id** – rollback segment id
- **space_id** – space where the segment placed
- **zip_size** – compressed page size in bytes if compressed otherwise 0
- **page_no** – page number of the segment header
- **max_size** – max size in pages

- **curr_size** – current size in pages

This table shows information about all the rollback segments (the default segment and the extra segments).

Here is an example of output with `innodb_extra_rsegments = 8`

```
mysql> select * from information_schema.innodb_rseg;
+-----+-----+-----+-----+-----+-----+
| rseg_id | space_id | zip_size | page_no | max_size | curr_size |
+-----+-----+-----+-----+-----+-----+
|      0 |      0 |      0 |      6 | 4294967294 |      1 |
|      1 |      0 |      0 |     13 | 4294967294 |      2 |
|      2 |      0 |      0 |     14 | 4294967294 |      1 |
|      3 |      0 |      0 |     15 | 4294967294 |      1 |
|      4 |      0 |      0 |     16 | 4294967294 |      1 |
|      5 |      0 |      0 |     17 | 4294967294 |      1 |
|      6 |      0 |      0 |     18 | 4294967294 |      1 |
|      7 |      0 |      0 |     19 | 4294967294 |      1 |
|      8 |      0 |      0 |     20 | 4294967294 |      1 |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)
```

8.10 Thread Based Profiling

Percona Server now uses thread based profiling by default, instead of process based profiling. This was implemented because with process based profiling, threads on the server, other than the one being profiled, can affect the profiling information.

Thread based profiling is using the information provided by the kernel `getrusage` function. Since the 2.6.26 kernel version, thread based resource usage is available with the **RUSAGE_THREAD**. This means that the thread based profiling will be used if you're running the 2.6.26 kernel or newer, or if the **RUSAGE_THREAD** has been ported back.

This feature is enabled by default if your system supports it, in other cases it uses process based profiling.

8.10.1 Version Specific Information

- 5.6.11-60.3: Thread based profiling ported from *Percona Server 5.5*.

8.11 Metrics for scalability measurement

Percona Server has implemented extra scalability metrics in 5.6.17-65.0. These metrics allow using Little's Law, queuing theory, and Universal Scalability Law to gain insights into server performance. This feature is implemented as a plugin.

8.11.1 Installation

Scalability Metrics plugin is shipped with *Percona Server*, but it is not installed by default. To enable the plugin you must run the following command:

```
INSTALL PLUGIN scalability_metrics SONAME 'scalability_metrics.so';
```

You can check if the plugin is loaded correctly by running:

```
SHOW PLUGINS;
```

The plugin should be listed in the output:

Name	Status	Type	Library	License
...				
scalability_metrics	ACTIVE	AUDIT	scalability_metrics.so	GPL

8.11.2 System Variables

variable `scalability_metrics_control`

Version Info

- 5.6.17-65.0 – Implemented

Command Line Yes

Scope Global

Dynamic Yes

Variable Type String

Default Value OFF

Values OFF, ON, RESET

This variable can be used to enable and disable the collection of metrics for scalability measurement. By setting the value to `RESET` all counters will be reset while continuing to count metrics.

8.11.3 Status Variables

variable `scalability_metrics_elapsedtime`

Version Info

- 5.6.17-65.0 – Implemented

Variable Type Numeric

This status variable shows total time elapsed since metrics collection was started.

variable `scalability_metrics_queries`

Version Info

- 5.6.17-65.0 – Implemented

Variable Type Numeric

This status variable shows number of completed queries since metrics collection was started.

variable `scalability_metrics_concurrency`

Version Info

- 5.6.17-65.0 – Implemented

Variable Type Numeric

This status variable shows number of queries currently executed.

variable scalability_metrics_totalltime

Version Info

- 5.6.17-65.0 – Implemented

Variable Type Numeric

This status variable shows total execution time of all queries, including the in-progress time of currently executing queries, in microseconds (ie. if two queries executed with 1 second of response time each, the result is 2 seconds).

variable scalability_metrics_busytime

Version Info

- 5.6.17-65.0 – Implemented

Variable Type Numeric

This counter accounts the non-idle server time, that is, time when at least one query was executing.

8.11.4 Version Specific Information

- 5.6.17-65.0 Scalability Metrics plugin has been implemented in *Percona Server*.

8.11.5 Other Reading

- [Fundamental performance and scalability instrumentation](#)
- [Forecasting MySQL Scalability with the Universal Scalability Law Whitepaper](#)

8.12 Response Time Distribution

The slow query log provides exact information about queries that take a long time to execute. However, sometimes there are a large number of queries that each take a very short amount of time to execute. This feature provides a tool for analyzing that information by counting and displaying the number of queries according to the length of time they took to execute. The user can define time intervals that divide the range 0 to positive infinity into smaller intervals and then collect the number of commands whose execution times fall into each of those intervals.

Note that in a replication environment, the server will not take into account *any* queries executed by the slave SQL threads (whether they are slow or not) for the time distribution.

Each interval is described as:

```
(range_base ^ n; range_base ^ (n+1)]
```

The range_base is some positive number (see Limitations). The interval is defined as the difference between two nearby powers of the range base.

For example, if the range base=10, we have the following intervals:

```
(0; 10 ^ -6], (10 ^ -6; 10 ^ -5], (10 ^ -5; 10 ^ -4], ..., (10 ^ -1; 10 ^ 1], (10^1; 10^2]... (10^7; p
```

or

```
(0; 0.000001], (0.000001; 0.000010], (0.000010; 0.000100], ..., (0.100000; 1.0]; (1.0; 10.0]... (1000
```

For each interval, a count is made of the queries with execution times that fell into that interval.

You can select the range of the intervals by changing the range base. For example, for base range=2 we have the following intervals:

```
(0; 2 ^ -19], (2 ^ -19; 2 ^ -18], (2 ^ -18; 2 ^ -17], ..., (2 ^ -1; 2 ^ 1], (2 ^ 1; 2 ^ 2]...(2 ^ 25;
```

or

```
(0; 0.000001], (0.000001, 0.000003], ..., (0.25; 0.5], (0.5; 2], (2; 4]...(8388608; positive infinity)
```

Small numbers look strange (i.e., don't look like powers of 2), because we lose precision on division when the ranges are calculated at runtime. In the resulting table, you look at the high boundary of the range.

For example, you may see:

time	count	total
0.000001	0	0.000000
0.000010	17	0.000094
0.000100	4301	0.236555
0.001000	1499	0.824450
0.010000	14851	81.680502
0.100000	8066	443.635693
1.000000	0	0.000000
10.000000	0	0.000000
100.000000	1	55.937094
1000.000000	0	0.000000
10000.000000	0	0.000000
100000.000000	0	0.000000
1000000.000000	0	0.000000
TOO LONG QUERY	0	0.000000

This means there were:

- * 17 queries with 0.000001 < query execution time <= 0.000010 seconds; total execution time of the 17 queries is 0.000094 seconds
- * 4301 queries with 0.000010 < query execution time <= 0.000100 seconds; total execution time of the 4301 queries is 0.236555 seconds
- * 1499 queries with 0.000100 < query execution time <= 0.001000 seconds; total execution time of the 1499 queries is 0.824450 seconds
- * 14851 queries with 0.001000 < query execution time <= 0.010000 seconds; total execution time of the 14851 queries is 81.680502 seconds
- * 8066 queries with 0.010000 < query execution time <= 0.100000 seconds; total execution time of the 8066 queries is 443.635693 seconds
- * 1 query with 10.000000 < query execution time <= 100.0000 seconds; total execution time of the 1 query is 55.937094 seconds

8.12.1 Installing the plugins

In order to enable this feature you'll need to install the necessary plugins:

```
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME_AUDIT SONAME 'query_response_time.so';
```

This plugin is used for gathering statistics.

```
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME SONAME 'query_response_time.so';
```

This plugin provides the interface (`QUERY_RESPONSE_TIME`) to output gathered statistics.

You can check if plugins are installed correctly by running:

```
mysql> SHOW PLUGINS;
```

```
...
| QUERY_RESPONSE_TIME          | ACTIVE | INFORMATION SCHEMA | query_response_time.so | GPL |
| QUERY_RESPONSE_TIME_AUDIT    | ACTIVE | AUDIT               | query_response_time.so | GPL |
+-----+-----+-----+-----+-----+

```

8.12.2 Usage

SELECT

You can get the distribution using the query:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME
time          count  total
0.000001      0      0.000000
0.000010      0      0.000000
0.000100      1      0.000072
0.001000      0      0.000000
0.010000      0      0.000000
0.100000      0      0.000000
1.000000      0      0.000000
10.000000     8      47.268416
100.000000    0      0.000000
1000.000000   0      0.000000
10000.000000  0      0.000000
100000.000000 0      0.000000
1000000.000000 0      0.000000
TOO LONG QUERY 0      0.000000
```

You can write a complex query like:

```
SELECT c.count, c.time,
  (SELECT SUM(a.count) FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME as a WHERE a.count != 0) as query_count,
  (SELECT COUNT(*) FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME as b WHERE b.count != 0) as not_zero_count,
  (SELECT COUNT(*) FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME) as region_count
FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME as c WHERE c.count > 0;
```

Note: If `query_response_time_stats` is ON, the execution times for these two SELECT queries will also be collected.

FLUSH

Flushing can be done by setting the `query_response_time_flush` to ON (or 1):

```
mysql> SET GLOBAL query_response_time_flush='ON';
```

FLUSH does two things:

- Clears the collected times from the `QUERY_RESPONSE_TIME` table
- Reads the value of `query_response_time_range_base` and uses it to set the range base for the table

Note: The execution time for the FLUSH query will also be collected.

Stored procedures

Stored procedure calls count as a single query.

Collect time point

Time is collected after query execution completes (before clearing data structures).

8.12.3 Version Specific Information

- 5.6.21-69.0: Feature ported from *Percona Server 5.5* as a plugin

8.12.4 System Variables

variable `query_response_time_flush`

Version Info

- **5.6.21-69.0** – Introduced.

Scope Global

Dynamic No

Variable Type Boolean

Default Value OFF

Range OFF/ON

Setting this variable to ON will flush the statistics and re-read the `query_response_time_range_base`.

variable `query_response_time_range_base`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 10

Range 2-1000

Sets up the logarithm base for the scale.

NOTE: The variable takes effect only after this command has been executed:

```
mysql> SET GLOBAL query_response_time_flush=1;
```

variable `query_response_time_stats`

Version Info

- **5.6.21-69.0** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value OFF

Range ON/OFF

This variable enables and disables collection of query times.

8.12.5 INFORMATION_SCHEMA Tables

table INFORMATION_SCHEMA.QUERY_RESPONSE_TIME

Columns

- **TIME** (*VARCHAR*) – Interval range in which the query occurred
- **COUNT** (*INT(11)*) – Number of queries with execution times that fell into that interval
- **TOTAL** (*VARCHAR*) – Total execution time of the queries

9.1 TokuDB Introduction

Percona Server is fully compatible with the separately available TokuDB storage engine package. TokuDB engine is not packaged with any *Percona Server 5.6* download. The TokuDB engine must be separately downloaded and then enabled as a plug-in component. This package can be installed alongside standard *Percona Server 5.6* releases starting with 5.6.19–67.0 and does not require any specially adapted version of *Percona Server*.

The TokuDB storage engine is a scalable, ACID and MVCC compliant storage engine that provides indexing-based query improvements, offers online schema modifications, and reduces slave lag for both hard disk drives and flash memory. This storage engine is specifically designed for high performance on write-intensive workloads which is achieved with Fractal Tree indexing.

Key TokuDB features available in the current production release:

- *TokuDB Compression*
- Multiple Clustering Indexes
- Hot Table Optimization
- Prelocking index and range scans

Some features that were available in older alpha releases have since been discontinued:

- Fast Updates with NOAR
- TokuDB AUTOINCREMENT implementation

TokuDB is released by Tokutek Inc. under a modified form of the GNU GPL v2 license. You can view a copy of the modified license [here](#).

9.2 TokuDB Installation

Percona Server is technically compatible with the separately available TokuDB storage engine package. TokuDB engine is not packaged with any *Percona Server 5.6* download. The TokuDB engine must be separately downloaded and then enabled as a plug-in component. This package can be installed alongside with standard *Percona Server 5.6* releases starting with 5.6.19–67.0 and does not require any specially adapted version of *Percona Server*.

The TokuDB storage engine is a scalable, ACID and MVCC compliant storage engine that provides indexing-based query improvements, offers online schema modifications, and reduces slave lag for both hard disk drives and flash memory. This storage engine is specifically designed for high performance on write-intensive workloads which is achieved with Fractal Tree indexing. To learn more about Fractal Tree indexing, you can visit the following [Wikipedia page](#).

Only the [Percona supplied](#) TokuDB engine should be used with *Percona Server 5.6*. A TokuDB engine downloaded from other sources may not be compatible.

The TokuDB engine is released by Tokutek Inc. under a modified form of the GNU GPL v2 license. You can view a copy of the modified license [here](#).

TokuDB Hot Backup is a proprietary, commercial product available for download only from [Tokutek.com](#). It is not available for download from Percona.

TokuDB is currently supported only for 64-bit Linux distributions. It is not available for Debian 6.0 (squeeze) and Ubuntu 10.04 (lucid) due to conflicts with the gcc version required by TokuDB.

9.2.1 Prerequisites

libjemalloc library

TokuDB storage engine requires libjemalloc library 3.3.0 or greater. If the version in the distribution repository is lower than that you can use one from [Percona Software Repositories](#) or download it from somewhere else.

If the libjemalloc wasn't installed and enabled before it will be automatically installed when installing the TokuDB storage engine package by using the apt or yum package manager, but *Percona Server* instance should be restarted for libjemalloc to be loaded. This way libjemalloc will be loaded with LD_PRELOAD. You can also enable libjemalloc by specifying malloc-lib variable in the [mysqld_safe] section of the my.cnf file:

```
[mysqld_safe]
malloc-lib= /path/to/jemalloc
```

Transparent huge pages

TokuDB won't be able to start if the transparent huge pages are enabled. [Transparent huge pages](#) is feature available in the newer kernel versions. You can check if the Transparent huge pages are enabled with:

```
$ cat /sys/kernel/mm/transparent_hugepage/enabled

[always] madvise never
```

If transparent huge pages are enabled and you try to start the TokuDB engine you'll get the following message in you error.log:

```
Transparent huge pages are enabled, according to /sys/kernel/mm/redhat_transparent_hugepage/enabled
Transparent huge pages are enabled, according to /sys/kernel/mm/transparent_hugepage/enabled
```

You can [disable](#) transparent huge pages permanently by passing transparent_hugepage=never to the kernel in your bootloader (**NOTE:** For this change to take an effect you'll need to reboot your server).

You can disable the transparent huge pages by running the following command as root (**NOTE:** Setting this will last only until the server is rebooted):

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

9.2.2 Installation

TokuDB storage engine for *Percona Server* is currently available in our [apt](#) and [yum](#) repositories.

You can install the *Percona Server* with TokuDB engine by using the apt/yum commands:

```
[root@centos ~]# yum install Percona-Server-tokudb-56.x86_64
```

or

```
root@wheezy:~# apt-get install percona-server-tokudb-5.6
```

9.2.3 Enabling the TokuDB Storage Engine

This storage engine requires manual installation if there is a root password already set up during the new installation or upgrade.

```
INSTALL PLUGIN tokudb SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_file_map SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_fractal_tree_info SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_fractal_tree_block_map SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_trx SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_locks SONAME 'ha_tokudb.so';
INSTALL PLUGIN tokudb_lock_waits SONAME 'ha_tokudb.so';
```

After the engine has been installed it should be present in the engines list. To check if the engine has been correctly installed and active:

```
mysql> SHOW ENGINES;
...
| TokudB | YES | Tokutek TokudB Storage Engine with Fractal Tree(tm) Technology | YES | YES | YES |
...

```

To check if all the TokuDB plugins have been installed correctly you should run:

```
mysql> SHOW PLUGINS;
...
| TokudB | ACTIVE | STORAGE ENGINE | ha_tokudb.so | GPL |
| TokudB_file_map | ACTIVE | INFORMATION SCHEMA | ha_tokudb.so | GPL |
| TokudB_fractal_tree_info | ACTIVE | INFORMATION SCHEMA | ha_tokudb.so | GPL |
| TokudB_fractal_tree_block_map | ACTIVE | INFORMATION SCHEMA | ha_tokudb.so | GPL |
| TokudB_trx | ACTIVE | INFORMATION SCHEMA | ha_tokudb.so | GPL |
| TokudB_locks | ACTIVE | INFORMATION SCHEMA | ha_tokudb.so | GPL |
| TokudB_lock_waits | ACTIVE | INFORMATION SCHEMA | ha_tokudb.so | GPL |
...

```

9.2.4 TokuDB Version

TokuDB storage engine version can be checked with:

```
mysql> SELECT @@tokudb_version;
+-----+
| @@tokudb_version |
+-----+
| tokudb-7.1.7-rc7 |
+-----+
1 row in set (0.00 sec)
```

9.2.5 Upgrade

Installing the TokuDB package is compatible with existing server setup and databases.

9.2.6 Version Specific Information

- 5.6.17–66.0 TokuDB storage engine available as a separate *Percona Server* package.
- 5.6.19–67.0 TokuDB storage engine is considered GA quality.

9.2.7 Other Reading

- [Official TokuDB Documentation](#)

9.3 Using TokuDB

TokuDB storage engine can be used as the rest of the already available *MySQL* engines. You can create the TokuDB tables by specifying TokuDB when creating the tables:

```
mysql> CREATE TABLE `City` (  
  `ID` int(11) NOT NULL AUTO_INCREMENT,  
  `Name` char(35) NOT NULL DEFAULT '',  
  `CountryCode` char(3) NOT NULL DEFAULT '',  
  `District` char(20) NOT NULL DEFAULT '',  
  `Population` int(11) NOT NULL DEFAULT '0',  
  PRIMARY KEY (`ID`),  
  KEY `CountryCode` (`CountryCode`)  
) ENGINE=TokuDB
```

You can also convert existing tables to TokuDB by running ALTER TABLE:

```
mysql> ALTER TABLE City ENGINE=TokuDB;
```

9.3.1 Other Reading

- [InnoDB vs TokuDB comparison](#)
- [TokuDB vs InnoDB in timeseries INSERT benchmark](#)
- [Considering TokuDB as an engine for timeseries data](#)
- [Benchmarking Percona Server TokuDB vs InnoDB](#)

9.4 TokuDB Compression

TokuDB storage engine provides new compression options for TokuDB tables.

These options are defined on per-table basis by setting ROW_FORMAT during a CREATE TABLE or ALTER TABLE. If no value is specified for ROW_FORMAT zlib compression is used by default. Currently available formats are:

- TOKUDB_ZLIB - This compression is using zlib library and provides mid-range compression with medium CPU utilization.

- TOKUDB_QUICKLZ - This compression is using quicklz library and provides light compression with low CPU utilization.
- TOKUDB_LZMA - This compression is using lzma library and provides the highest compression with high CPU utilization.
- TOKUDB_UNCOMPRESSED - This option disables the compression.

9.4.1 Example

In order to create a new table with the highest compression you need to specify the TOKUDB_LZMA as the ROW_FORMAT:

```
mysql> CREATE TABLE `City` (
  > `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `CountryCode` (`CountryCode`)
) ENGINE=TokuDB, ROW_FORMAT=TOKUDB_LZMA;
```

In order to change the compression for this table you need to change the ROW_FORMAT with ALTER TABLE command:

```
mysql> ALTER TABLE City ROW_FORMAT=TOKUDB_ZLIB;
```

Note: When the compression format is changed it will affect only newly written data. In order to change the old data you'll need to run OPTIMIZE TABLE on that table.

```
mysql> OPTIMIZE TABLE City;
```

9.5 Multiple Clustering Keys

TokuDB allows a secondary key to be defined as a clustering key. This means that all of the columns in the table are clustered with the secondary key. *Percona Server* parser and query optimizer support Multiple Clustering Keys when TokuDB engine is used. This means that the query optimizer will avoid primary clustered index reads and replace them by secondary clustered index reads in certain scenarios.

The parser has been extended to support following syntax:

```
CREATE TABLE ... ( ..., CLUSTERING KEY identifier (column list), ...
CREATE TABLE ... ( ..., UNIQUE CLUSTERING KEY identifier (column list), ...
CREATE TABLE ... ( ..., CLUSTERING UNIQUE KEY identifier (column list), ...
CREATE TABLE ... ( ..., CONSTRAINT identifier UNIQUE CLUSTERING KEY identifier (column list), ...
CREATE TABLE ... ( ..., CONSTRAINT identifier CLUSTERING UNIQUE KEY identifier (column list), ...

CREATE TABLE ... (... column type CLUSTERING [UNIQUE] [KEY], ...)
CREATE TABLE ... (... column type [UNIQUE] CLUSTERING [KEY], ...)

ALTER TABLE ..., ADD CLUSTERING INDEX identifier (column list), ...
ALTER TABLE ..., ADD UNIQUE CLUSTERING INDEX identifier (column list), ...
ALTER TABLE ..., ADD CLUSTERING UNIQUE INDEX identifier (column list), ...
ALTER TABLE ..., ADD CONSTRAINT identifier UNIQUE CLUSTERING INDEX identifier (column list), ...
```

```
ALTER TABLE ..., ADD CONSTRAINT identifier CLUSTERING UNIQUE INDEX identifier (column list), ...  
CREATE CLUSTERING INDEX identifier ON ...
```

9.5.1 Examples

To create clustered index in TokuDB table, you need to add `CLUSTERING` keyword in front of any index that is to be created.

- To create a clustered index while creating a new table:

```
CREATE TABLE city (city_id int PRIMARY KEY, name varchar(50), CLUSTERING KEY (city_id)) engine=toku
```

- To add a clustered index on already existing table:

```
ALTER TABLE city ADD CLUSTERING INDEX clstr_key(city_id);
```

or

```
CREATE CLUSTERING INDEX clstr_key ON city(city_id);
```

9.5.2 Version Specific Information

- 5.6.17–65.0 - *Percona Server* implemented support for Multiple Clustering Keys

9.5.3 Other Reading

- [Introducing Multiple Clustering Indexes](#)
- [Clustering indexes vs. Covering indexes](#)

REFERENCE

10.1 List of upstream *MySQL* bugs fixed in *Percona Server 5.6*

<p>Upstream bug #70860 - --tc-heuristic-recover option values are broken Launchpad bug #1334330 Upstream state Open (checked on 2014-10-06) Fix Released 5.6.20-68.0 Upstream fix N/A</p>
<p>Upstream bug #73418 - Add --manual-lldb option to mysql-test-run.pl Launchpad bug #1328482 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.20-68.0 Upstream fix N/A</p>
<p>Upstream bug #67806 - Multiple user level lock per connection Launchpad bug N/A Upstream state Closed Fix Released 5.6.19-67.0 Upstream fix N/A</p>
<p>Upstream bug #1118 - Allow multiple concurrent locks with GET_LOCK() Launchpad bug N/A Upstream state Closed Fix Released 5.6.19-67.0 Upstream fix N/A</p>
<p>Upstream bug #72615 - MTR --mysqld=--default-storage-engine=foo incompatible w/ dynamically... Launchpad bug #1318537 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.17-66.0 Upstream fix N/A</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #72163 - Rev 5774 broke rpl_plugin_load Launchpad bug #1299688 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.17-65.0 Upstream fix N/A</p>
<p>Upstream bug #60782 - Audit plugin API: no MYSQL_AUDIT_GENERAL_LOG notifications with general... Launchpad bug #1182535 Upstream state Closed Fix Released 5.6.17-65.0 Upstream fix N/A</p>
<p>Upstream bug #71250 - Bison 3 breaks mysql build Launchpad bug #1262439 Upstream state Closed Fix Released 5.6.17-65.0 Upstream fix N/A</p>
<p>Upstream bug #71374 - Slave IO thread won't attempt auto reconnect to the master/error-code 1159 Launchpad bug #1268729 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.16-64.1 Upstream fix N/A</p>
<p>Upstream bug #71988 - page_cleaner: aggressive background flushing Launchpad bug #1238039 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.16-64.0 Upstream fix N/A</p>
<p>Upstream bug #71624 - printf size_t results in a fatal warning in 32-bit debug builds Launchpad bug #1277505 Upstream state Can't repeat (checked on 2014-10-06) Fix Released 5.6.16-64.0 Upstream fix N/A</p>
<p>Upstream bug #71094 - ssl.cmake related warnings Launchpad bug #1274411 Upstream state Closed Fix Released 5.6.16-64.0 Upstream fix N/A</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #71089 - CMake warning when generating Makefile Launchpad bug #1274827 Upstream state Closed Fix Released 5.6.16-64.0 Upstream fix 5.6.18</p>
<p>Upstream bug #71708 - 70768 fix perf regression: high rate of RW lock creation and destruction Launchpad bug #1279671 Upstream state Closed Fix Released 5.6.16-64.0 Upstream fix 5.6.19</p>
<p>Upstream bug #54430 - innodb should retry partial reads/writes where errno was 0 Launchpad bug #1262500 Upstream state Closed Fix Released 5.6.16-64.0 Upstream fix N/A</p>
<p>Upstream bug #70854 - Tc_log_page_size should be unflushable or server crashes if 2 XA SEs ... Launchpad bug #1255551 Upstream state Closed Fix Released 5.6.16-64.0 Upstream fix N/A</p>
<p>Upstream bug #47134 - Crash on startup when XA support functions activated by a second engine Launchpad bug #1255549 Upstream state Closed Fix Released 5.6.16-64.0 Upstream fix N/A</p>
<p>Upstream bug #71270 - Failures to end bulk insert for partitioned tables handled incorrectly Launchpad bug #1204871 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.16-64.0 Upstream fix N/A</p>
<p>Upstream bug #71217 - Threadpool - add thd_wait_begin/thd_wait_end to the network IO functions Launchpad bug #1159743 Upstream state Open (checked on 2014-10-06) Fix Released 5.6.15-63.0 Upstream fix N/A</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #41975 - Support for SSL options not included in mysqlbinlog Launchpad bug #1258154 Upstream state Closed Fix Released 5.6.15-63.0 Upstream fix N/A</p>
<p>Upstream bug #71092 - InnoDB FTS introduced new mutex sync level in 5.6.15, broke UNIV_SYNC ... Launchpad bug #1258154 Upstream state Closed Fix Released 5.6.15-63.0 Upstream fix 5.6.12</p>
<p>Upstream bug #71411 - buf_flush_LRU() does not return correct number in case of compressed pages Launchpad bug #1231918 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.13-61.0 Upstream fix N/A</p>
<p>Upstream bug #70417 - rw_lock_x_lock_func_nowait() calls os_thread_get_curr_id() mostly ... Launchpad bug #1230220 Upstream state Closed Fix Released 5.6.13-61.0 Upstream fix 5.6.16</p>
<p>Upstream bug #70490 - Suppression is too strict on some systems Launchpad bug #1205196 Upstream state No Feedback (checked on 2014-10-06) Fix Released 5.6.13-61.0 Upstream fix N/A</p>
<p>Upstream bug #64556 - Interrupting a query inside InnoDB causes an unrelated warning to be ... Launchpad bug #1115158 Upstream state Closed Fix Released 5.6.13-61.0 Upstream fix 5.6.14</p>
<p>Upstream bug #70500 - Page cleaner should perform LRU flushing regardless of server activity Launchpad bug #1234562 Upstream state Open (checked on 2014-10-06) Fix Released 5.6.13-61.0 Upstream fix N/A</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #60682 - deadlock from thd_security_context Launchpad bug #1115048 Upstream state Closed Fix Released 5.6.13-61.0 Upstream fix N/A</p>
<p>Upstream bug #70489 - Crash when using AES_ENCRYPT on empty string Launchpad bug #1201033 Upstream state Unsupported Fix Released 5.6.13-61.0 Upstream fix N/A</p>
<p>Upstream bug #68481 - InnoDB LRU flushing for MySQL 5.6 needs work Launchpad bug #1232406 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.13-61.0 Upstream fix N/A</p>
<p>Upstream bug #70453 - Add hard timeouts to page cleaner flushes Launchpad bug #1232101 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.13-61.0 Upstream fix N/A</p>
<p>Upstream bug #69170 - buf_flush_LRU is lazy Launchpad bug #1231918 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.13-61.0 Upstream fix N/A</p>
<p>Upstream bug #68555 - thread convoys from log_checkpoint_margin with innodb_buffer_pool_inst... Launchpad bug #1236884 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.13-61.0 Upstream fix N/A</p>
<p>Upstream bug #70228 - Is buf_LRU_free_page() really supposed to make non-zip block sticky at ... Launchpad bug #1220544 Upstream state Closed Fix Released 5.6.13-60.6 Upstream fix N/A</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #69617 - 5.6.12 removed UNIV_SYNC_DEBUG from UNIV_DEBUG Launchpad bug #1216815 Upstream state Closed Fix Released 5.6.13-60.6 Upstream fix 5.6.16</p>
<p>Upstream bug #69258 - does buf_LRU_buf_pool_running_out need to lock buffer pool mutexes Launchpad bug #1219842 Upstream state Not a Bug Fix Released 5.6.13-60.6 Upstream fix N/A</p>
<p>Upstream bug #70216 - Unnecessary overhead from persistent adaptive hash index latches Launchpad bug #1218347 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.13-60.6 Upstream fix N/A</p>
<p>Upstream bug #62018 - innodb adaptive hash index mutex contention Launchpad bug #1216804 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.13-60.6 Upstream fix N/A</p>
<p>Upstream bug #54814 - make BUF_READ_AHEAD_AREA a constant Launchpad bug #1186974 Upstream state Closed Fix Released 5.6.13-60.5 Upstream fix N/A</p>
<p>Upstream bug #69179 - accessing information_schema.partitions causes plans to change Launchpad bug #1192354 Upstream state Closed Fix Released 5.6.13-60.5 Upstream fix 5.6.14</p>
<p>Upstream bug #42415 - UPDATE/DELETE with LIMIT clause unsafe for SBL even with ORDER BY PK ... Launchpad bug #1132194 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.13-60.5 Upstream fix N/A</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #69639 - mysql failed to build with dtrace Sun D 1.11 Launchpad bug #1196460 Upstream state Open (checked on 2014-10-06) Fix Released 5.6.13-60.5 Upstream fix N/A</p>
<p>Upstream bug #69524 - Some tests for table cache variables fail if open files limit is too low Launchpad bug #1182572 Upstream state Closed Fix Released 5.6.12-60.4 Upstream fix N/A</p>
<p>Upstream bug #68354 - Server crashes on update/join FEDERATED + local table when only 1 local... Launchpad bug #1182572 Upstream state N/A Fix Released 5.6.12-60.4 Upstream fix N/A</p>
<p>Upstream bug #69856 - mysql_install_db does not function properly in 5.6 for debug builds Launchpad bug #1179359 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.12-60.4 Upstream fix N/A</p>
<p>Upstream bug #71603 - file name is not escaped in binlog for LOAD DATA INFILE statement Launchpad bug #1277351 Upstream state N/A Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #71183 - os_file_fsync() should handle fsync() returning EINTR Launchpad bug #1262651 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #63451 - atomic/x86-gcc.h:make_atomic_cas_body64 potential miscompilation bug Launchpad bug #878022 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix 5.6.16</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #70277 - last argument of LOAD DATA ... SET ... statement repeated twice in binlog</p> <p>Launchpad bug #1223196</p> <p>Upstream state Closed</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix 5.6.15</p>
<p>Upstream bug #69252 - All the parts.partition_max* tests are broken with MTR -parallel</p> <p>Launchpad bug #1180481</p> <p>Upstream state Closed</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix 5.6.15</p>
<p>Upstream bug #69265 - -DBUILD_CONFIG=mysql_release -DWITH_DEBUG=ON fails 4 and skips 27 MTR ...</p> <p>Launchpad bug #1163135</p> <p>Upstream state Closed</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix N/A</p>
<p>Upstream bug #68714 - Remove literal statement digest values from perfschema tests</p> <p>Launchpad bug #1157078</p> <p>Upstream state Verified (checked on 2014-10-06)</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix N/A</p>
<p>Upstream bug #68635 - Doc: Multiple issues with performance_schema_max_statement_classes</p> <p>Launchpad bug #1157075</p> <p>Upstream state Closed</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix N/A</p>
<p>Upstream bug #68800 - client doesn't read plugin-dir from my.cnf set by MYSQL_READ_DEFAULT_FILE</p> <p>Launchpad bug #1155859</p> <p>Upstream state Closed</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix 5.6.12</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #69124 - Incorrect truncation of long SET expression in LOAD DATA can cause SQL ...</p> <p>Launchpad bug #1175519</p> <p>Upstream state N/A</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix N/A</p>
<p>Upstream bug #68970 - fsp_reserve_free_extents switches from small to big tblspace handling ...</p> <p>Launchpad bug #1169494</p> <p>Upstream state Verified (checked on 2014-10-06)</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix N/A</p>
<p>Upstream bug #68713 - create_duplicate_weedout_tmp_table() leaves key_part_flag uninitialized</p> <p>Launchpad bug #1157037</p> <p>Upstream state Verified (checked on 2014-10-06)</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix N/A</p>
<p>Upstream bug #68490 - slave_max_allowed_packet Not Honored on Slave IO Connect</p> <p>Launchpad bug #1135097</p> <p>Upstream state Closed</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix 5.6.12</p>
<p>Upstream bug #68999 - SSL_OP_NO_COMPRESSION not defined</p> <p>Launchpad bug #1183610</p> <p>Upstream state No Feedback (checked on 2014-10-06)</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix N/A</p>
<p>Upstream bug #68845 - Unnecessary log_sys->mutex reacquisition in mtr_log_reserve_and_write()</p> <p>Launchpad bug #1163439</p> <p>Upstream state Verified (checked on 2014-10-06)</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix N/A</p>
<p>Upstream bug #62578 - mysql client aborts connection on terminal resize</p> <p>Launchpad bug #925343</p> <p>Upstream state Closed</p> <p>Fix Released 5.6.11-60.3</p> <p>Upstream fix 5.6.12</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #49169 - read_view_open_now is inefficient with many concurrent sessions Launchpad bug #1131187 and #1131189 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #63144 - CREATE TABLE IF NOT EXISTS metadata lock is too restrictive Launchpad bug #1127008 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix 5.6.13</p>
<p>Upstream bug #68477 - Suboptimal code in skip_trailing_space() Launchpad bug #1132351 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #68476 - Suboptimal code in my_strnxfrm_simple() Launchpad bug #1132350 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #67504 - Duplicate error in replication with slave triggers and auto increment Launchpad bug #1068210 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #67974 - Server crashes in add_identifier on concurrent ALTER TABLE and SHOW ENGINE Launchpad bug #1017192 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix 5.6.12</p>
<p>Upstream bug #68045 - security vulnerability CVE-2012-4414 Launchpad bug #1049871 Upstream state N/A Fix Released 5.6.11-60.3 Upstream fix N/A</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #66550 - security vulnerability CVE-2012-4414 Launchpad bug #1049871 Upstream state N/A Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #67685 - security vulnerability CVE-2012-5611 Launchpad bug #1083377 Upstream state N/A Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #66237 - Temporary files created by binary log cache are not purged after transa... Launchpad bug #1070856 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #54430 - innodb should retry partial reads/writes where errno was 0 Launchpad bug #1079596 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #62856 - Check for “stack overrun” doesn’t work with gcc-4.6, server crashes Launchpad bug #1042517 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #61180 - korr/store macros in my_global.h assume the argument to be a char pointer Launchpad bug #1042517 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #61178 - Incorrect implementation of intersect(ulonglong) in non-optimized Bitmap.. Launchpad bug #1042517 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.11-60.3 Upstream fix N/A</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #54127 - mysqld segfaults when built using <code>--with-max-indexes=128</code> Launchpad bug #1042517 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #64800 - mysqldump with <code>--include-master-host-port</code> putting quotes around port no. Launchpad bug #1013432 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #66301 - INSERT ... ON DUPLICATE KEY UPDATE + innodb_autoinc_lock_mode=1 is broken Launchpad bug #1035225 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix 5.6.12</p>
<p>Upstream bug #60743 - typo in cmake/dtrace.cmake Launchpad bug #1013455 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix 5.6.13</p>
<p>Upstream bug #64663 - Segfault when adding indexes to InnoDB temporary tables Launchpad bug #999147 Upstream state N/A Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #64432 - Bug #54330 (Broken fast index creation) was never fixed in 5.5 Launchpad bug #939485 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #25007 - memory tables with dynamic rows format Launchpad bug #1148822 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.11-60.3 Upstream fix N/A</p>
Continued on next page

Table 10.1 – continued from previous page

<p>Upstream bug #61595 - mysql-test/include/wait_for_slave_param.inc timeout logic is incorrect Launchpad bug #800035 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #54160 - InnoDB should retry on failed read or write, not immediately panic Launchpad bug #764395 Upstream state Closed Fix Released 5.6.11-60.3 Upstream fix N/A</p>
<p>Upstream bug #68116 - InnoDB monitor may hit an assertion error in buf_page_get_gen in debug ... Launchpad bug #1100178 Upstream state Closed Fix Released 5.6.10-60.2 Upstream fix 5.6.22</p>
<p>Upstream bug #65946 - Sid_map::Sid_map calls DBUG which may have uninitialized THR_KEY_mysys and.. Launchpad bug #1050758 Upstream state Duplicate/Closed Fix Released 5.6.5-60.0 Upstream fix 5.6.15</p>
<p>Upstream bug #20001 - Support for temp-tables in INFORMATION_SCHEMA Launchpad bug N/A Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.5-60.0 Upstream fix N/A</p>
<p>Upstream bug #69146 - Optimization in buf_pool_get_oldest_modification if srv_buf_pool_instances Launchpad bug #1176496 Upstream state Verified (checked on 2014-10-06) Fix Released 5.6.5-60.0 Upstream fix N/A</p>

10.2 List of variables introduced in Percona Server 5.6

10.2.1 System Variables

- `enforce_storage_engine`

- `expand_fast_index_creation`
- `extra_max_connections`
- `extra_port`
- `innodb_buffer_pool_populate`
- `innodb_corrupt_table_action`
- `innodb_fake_changes`
- `innodb_kill_idle_transaction`
- `innodb_locking_fake_changes`
- `innodb_log_arch_dir`
- `innodb_log_arch_expire_sec`
- `innodb_log_archive`
- `innodb_log_block_size`
- `innodb_max_bitmap_file_size`
- `innodb_max_changed_pages`
- `innodb_show_locks_held`
- `innodb_show_verbose_locks`
- `innodb_track_changed_pages`
- `innodb_use_atomic_writes`
- `innodb_use_global_flush_log_at_trx_commit`
- `log_slow_filter`
- `log_slow_rate_limit`
- `log_slow_rate_type`
- `log_slow_sp_statements`
- `log_slow_verbosity`
- `log_warnings_suppress`
- `max_binlog_files`
- `query_cache_strip_comments`
- `slow_query_log_timestamp_always`
- `slow_query_log_timestamp_precision`
- `slow_query_log_use_global_control`
- `thread_pool_high_prio_tickets`
- `thread_pool_idle_timeout`
- `thread_pool_max_threads`
- `thread_pool_oversubscribe`
- `thread_pool_size`
- `thread_pool_stall_limit`

- `thread_statistics`
- `userstat`

10.2.2 Status Variables

- `Com_purge_archived`
- `Com_purge_archived_before_date`
- `Com_show_client_statistics`
- `Com_show_index_statistics`
- `Com_show_slave_status_nolock`
- `Com_show_table_statistics`
- `Com_show_thread_statistics`
- `Com_show_user_statistics`
- `Innodb_background_log_sync`
- `Innodb_buffer_pool_pages_LRU_flushed`
- `Innodb_buffer_pool_pages_made_not_young`
- `Innodb_buffer_pool_pages_made_young`
- `Innodb_buffer_pool_pages_old`
- `Innodb_checkpoint_age`
- `Innodb_checkpoint_max_age`
- `Innodb_deadlocks`
- `Innodb_history_list_length`
- `Innodb_ibuf_discarded_delete_marks`
- `Innodb_ibuf_discarded_deletes`
- `Innodb_ibuf_discarded_inserts`
- `Innodb_ibuf_free_list`
- `Innodb_ibuf_merged_delete_marks`
- `Innodb_ibuf_merged_deletes`
- `Innodb_ibuf_merged_inserts`
- `Innodb_ibuf_merges`
- `Innodb_ibuf_segment_size`
- `Innodb_ibuf_size`
- `Innodb_lsn_current`
- `Innodb_lsn_flushed`
- `Innodb_lsn_last_checkpoint`
- `Innodb_master_thread_active_loops`
- `Innodb_master_thread_idle_loops`

- `Innodb_max_trx_id`
- `Innodb_mem_adaptive_hash`
- `Innodb_mem_dictionary`
- `Innodb_mem_total`
- `Innodb_mutex_os_waits`
- `Innodb_mutex_spin_rounds`
- `Innodb_mutex_spin_waits`
- `Innodb_oldest_view_low_limit_trx_id`
- `Innodb_purge_trx_id`
- `Innodb_purge_undo_no`
- `Innodb_current_row_locks`
- `Innodb_read_views_memory`
- `Innodb_descriptors_memory`
- `Innodb_s_lock_os_waits`
- `Innodb_s_lock_spin_rounds`
- `Innodb_s_lock_spin_waits`
- `Innodb_x_lock_os_waits`
- `Innodb_x_lock_spin_rounds`
- `Innodb_x_lock_spin_waits`
- `Threadpool_idle_threads`
- `Threadpool_threads`

10.3 Development of Percona Server

Percona Server is an open source project to produce a distribution of the *MySQL* server with improved performance, scalability and diagnostics.

10.3.1 Submitting Changes

We keep trunk in a constant state of stability to allow for a release at any time and to minimize wasted time by developers due to broken code from somebody else interfering with their day.

You should also be familiar with our *Jenkins* setup.

Overview

At Percona we use *Bazaar* for source control and *launchpad* for both code hosting and release management.

Changes to our software projects could be because of a new feature (blueprint) or fixing a bug (bug). Projects such as refactoring could be classed as a blueprint or a bug depending on the scope of the work.

Blueprints and bugs are targeted to specific milestones (releases). A milestone is part of a series - e.g. 1.6 is a series in Percona XtraBackup and 1.6.1, 1.6.2 and 1.6.3 are milestones in the 1.6 series.

Code is proposed for merging in the form of merge requests on launchpad.

Some software (such as Percona Xtrabackup) we maintain both a development branch and a stable branch. For example: Xtrabackup 1.6 is the current stable series, and changes that should make it into bugfix releases of 1.6 should be proposed for the 1.6 tree. However, most new features or more invasive (or smaller) bug fixes should be targeted to the next release, currently 1.7. If submitting something to 1.6, you should also propose a branch that has these changes merged to the development release (1.7). This way somebody else doesn't have to attempt to merge your code and we get to run any extra tests that may be in the tree (and check compatibility with all platforms).

For Percona Server, we have two current bzd branches on which development occurs: 5.1 and 5.5. As Percona Server is not a traditional project, instead being a set of patches against an existing product, these two branches are not related. That is, we do not merge from one to the other. To have your changes in both, you must propose two branches: one for 5.1 version of patch and one for 5.5.

Making a change to a project

In this case we're going to use percona-xtrabackup as an example. workflow is similar for Percona Server, but patch will need to be modified both in 5.1 and 5.5 branches.

- `bzd branch lp:percona-xtrabackup featureX` (where 'featureX' is a sensible name for the task at hand)
- (developer makes changes in featureX, testing locally)
- Developer pushes to `lp:~username/percona-xtrabackup/featureX`
- When the developer thinks the branch may be ready to be merged, they will run the branch through param build.
- If there are any build or test failures, developer fixes them (in the case of failing tests in trunk... no more tests should fail. Eventually all tests will pass in trunk)
- Developer can then submit a merge proposal to `lp:percona-xtrabackup`, referencing URL for the param build showing that build and test passes
- Code undergoes review
- Once code is accepted, it can be merged (see other section)

If the change also applies to a stable release (e.g. 1.6) then changes should be made on a branch of 1.6 and merged to a branch of trunk. In this case there should be two branches run through param build and two merge proposals (one for 1.6 and one with the changes merged to trunk). This prevents somebody else having to guess how to merge your changes.

Merging approved branches

Before code hits trunk, it goes through a "staging" branch, where some extra tests may be run (e.g. valgrind) along with testing that all branches behave well together (build and test) before pushing to trunk.

To ensure quality, **DO NOT push directly to trunk!** everything must go through adequate testing first. This ensures that at any point trunk is in a releasable state.

Please note that **ALL changes must go through staging first** This is to ensure that several approved merge requests do not interact badly with each other.

- Merge captain (for lack of a better term for the person merging approved code into trunk) may collate several approved branches that have individually passed param-build as run by the original developers.
 - Workflow would look something like this:

- * `bzr branch lp:percona-xtrabackup staging`
 - * `bzr merge lp:~user/percona-xtrabackup/featureX`
 - * `bzr commit -m "merge feature X"`
 - * `bzr merge lp:~user/percona-xtrabackup/featureY`
 - * `bzr commit -m "merge feature Y"`
 - * `bzr push --overwrite lp:percona-xtrabackup/staging'`
 - * Run `lp:percona-xtrabackup/staging` through param build (in future, we'll likely have a Jenkins job specifically for this)
 - * If build succeeds, `bzr push lp:percona-server` (and branches will be automatically marked as 'merged'.. although bug reports will need to be manually changed to 'Fix Released')
 - * If build or test fails, attempt to find which branch may be the cause, and repeat process but without that branch.
- Any failing branch will be set to 'Work in Progress' with a 'Needs fixing' review with the URL of the build in jenkins where the failure occurred. This will allow developers to fix their code.

Resubmitting a merge request

In the event of a merge request being marked as 'Work In Progress' due to build/test failures when merging, the developer should fix up the branch, run through param build and then 'Resubmit' the merge proposal.

There is a link on launchpad to resubmit the merge proposal, this means it appears in the list of merge requests to review again rather than off in the "work in progress" section.

Percona Server

The same process for Percona Server, but we have different branches (and merge requests) for 5.1 and 5.5 series.

Upgrading MySQL base version

- Same process as other modifications.
- create local branch
- make changes
- param build
- merge request

We will need some human processes to ensure that we do not merge extra things during the time when base MySQL version is being updated to avoid making life harder for the person doing the update.

10.3.2 Making a release

- `bzr branch lp:project release-project-VERSION`
- build packages
- perform any final tests (as we transition, this will already have been done by jenkins)
- `bzr tag project-version`

- merge request back to lp:project including the tag (TODO: write exact bzd commands for this)

This way anybody can easily check out an old release by just using bzd to branch the specific tag.

10.3.3 Jenkins

Our Jenkins instance uses a mixture of VMs on physical hosts that Percona runs and Virtual Machines in Amazon EC2 that are launched on demand.

Basic Concepts

We have some jobs that are activated based on source control changes (new commits in a bzd repository). We have some that are “param build” - that is, a user specifies parameters for the build (e.g. the bzd tree). A param-build allows developers to ensure their branch compiles and passes tests on all supported platforms *before* submitting a merge request. This helps us maintain the quality of the main bzd branches and not block other developers work.

Jenkins is a Master/Slave system and the jenkins master schedules the builds across available machines (and may launch new VMs in EC2 to meet demand).

Most of our jobs are what’s known as “matrix builds”. That is, a job that will be run with several different configurations of the project (e.g. release, debug) across several platforms (e.g. on a host matching the label of “centos5-32” and a host matching label of “ubuntu-natty-32bit”). Matrix builds show a table of lights to indicate their status. Clicking “build now” on one of these queues up builds for all of the combinations.

We have some integration of our regression test suites (currently xtrabackup) with Jenkins ability to parse JUnitXML, presenting a nice user interface to any test failures.

Because building some projects is non-trivial, in order to not duplicate the list of compile instructions for each job, we use template builds. You’ll see builds such as percona-xtrabackup-template which is a disabled job, but all current xtrabackup jobs point to it for the commands to build and run the test suite.

Percona Xtrabackup

<http://jenkins.percona.com/view/Percona%20Xtrabackup/>

We currently build both xtrabackup 1.6 and xtrabackup trunk (will become 1.7).

There are param-builds for 1.6 and trunk too. These should be run for each merge request (and before any collection of merged branches is pushed to trunk)

Percona Server

We have separate jobs for Percona Server 5.1 and Percona Server 5.5 due to the different build systems that MySQL 5.1 and 5.5 use.

The `mysql-test-run.pl` test suite is integrated with Jenkins through `subunit` and `subunit2junitxml` allowing us to easily see which tests passed/failed on any particular test run.

Percona Server 5.1

<http://jenkins.percona.com/view/Percona%20Server%205.1/>

We have trunk and param jobs. We also have a valgrind job that will run after a successful trunk build.

Percona Server 5.5

<http://jenkins.percona.com/view/Percona%20Server%205.5/>

Similar to 5.1, but for PS5.5 instead.

MySQL Builds

<http://jenkins.percona.com/view/MySQL/>

I've set up a few jobs in Jenkins that should help us predict the future for Percona Server. Namely, if upstream MySQL may cause us any problems.

I wanted to see if some test failures were possibly upstream, so I set up two jobs:

<http://jenkins.percona.com/view/MySQL/job/mysql-5.1-url-param/> <http://jenkins.percona.com/view/MySQL/job/mysql-5.5-url-param/>

both of which ask for a URL to a MySQL source tarball and then do a full build and test across the platforms we have in jenkins.

But my next thought was that we could try and do this *before* the source tarballs come out - hopefully then being able to have MySQL release source tarballs that do in fact pass build and test everywhere where we're wanting to support Percona Server.

<http://jenkins.percona.com/view/MySQL/job/mysql-5.1-trunk/> <http://jenkins.percona.com/view/MySQL/job/mysql-5.5-trunk/>

are scheduled to just try once per week (we can change the frequency if we want to) to build and test from the MySQL bzt trees.

I also have a valgrind build (same configuration as for Percona Server) to help us see if there's any new valgrind warnings (or missed suppressions).

I'm hoping that these jobs will help us catch any future problems before they become our problem. (e.g. we can easily see that the sporadic test failures we see in Percona Server are actually in upstream MySQL).

10.4 Trademark Policy

This Trademark Policy is to ensure that users of Percona-branded products or services know that what they receive has really been developed, approved, tested and maintained by Percona. Trademarks help to prevent confusion in the marketplace, by distinguishing one company's or person's products and services from another's.

Percona owns a number of marks, including but not limited to Percona, XtraDB, Percona XtraDB, XtraBackup, Percona XtraBackup, Percona Server, and Percona Live, plus the distinctive visual icons and logos associated with these marks. Both the unregistered and registered marks of Percona are protected.

Use of any Percona trademark in the name, URL, or other identifying characteristic of any product, service, website, or other use is not permitted without Percona's written permission with the following three limited exceptions.

First, you may use the appropriate Percona mark when making a nominative fair use reference to a bona fide Percona product.

Second, when Percona has released a product under a version of the GNU General Public License ("GPL"), you may use the appropriate Percona mark when distributing a verbatim copy of that product in accordance with the terms and conditions of the GPL.

Third, you may use the appropriate Percona mark to refer to a distribution of GPL-released Percona software that has been modified with minor changes for the sole purpose of allowing the software to operate on an operating system

or hardware platform for which Percona has not yet released the software, provided that those third party changes do not affect the behavior, functionality, features, design or performance of the software. Users who acquire this Percona-branded software receive substantially exact implementations of the Percona software.

Percona reserves the right to revoke this authorization at any time in its sole discretion. For example, if Percona believes that your modification is beyond the scope of the limited license granted in this Policy or that your use of the Percona mark is detrimental to Percona, Percona will revoke this authorization. Upon revocation, you must immediately cease using the applicable Percona mark. If you do not immediately cease using the Percona mark upon revocation, Percona may take action to protect its rights and interests in the Percona mark. Percona does not grant any license to use any Percona mark for any other modified versions of Percona software; such use will require our prior written permission.

Neither trademark law nor any of the exceptions set forth in this Trademark Policy permit you to truncate, modify or otherwise use any Percona mark as part of your own brand. For example, if XYZ creates a modified version of the Percona Server, XYZ may not brand that modification as “XYZ Percona Server” or “Percona XYZ Server”, even if that modification otherwise complies with the third exception noted above.

In all cases, you must comply with applicable law, the underlying license, and this Trademark Policy, as amended from time to time. For instance, any mention of Percona trademarks should include the full trademarked name, with proper spelling and capitalization, along with attribution of ownership to Percona Inc. For example, the full proper name for XtraBackup is Percona XtraBackup. However, it is acceptable to omit the word “Percona” for brevity on the second and subsequent uses, where such omission does not cause confusion.

In the event of doubt as to any of the conditions or exceptions outlined in this Trademark Policy, please contact trademarks@percona.com for assistance and we will do our very best to be helpful.

10.5 Index of INFORMATION_SCHEMA Tables

This is a list of the INFORMATION_SCHEMA TABLES that exist in *Percona Server* with *XtraDB*. The entry for each table points to the page in the documentation where it’s described.

- `CLIENT_STATISTICS`
- `GLOBAL_TEMPORARY_TABLES`
- `INDEX_STATISTICS`
- `INNODB_CHANGED_PAGES`
- `QUERY_RESPONSE_TIME`
- `TABLE_STATISTICS`
- `TEMPORARY_TABLES`
- `THREAD_STATISTICS`
- `USER_STATISTICS`
- `XTRADB_INTERNAL_HASH_TABLES`
- `XTRADB_READ_VIEW`
- `XTRADB_RSEG`

10.6 Frequently Asked Questions

10.6.1 Q: Will *Percona Server with XtraDB* invalidate our *MySQL* support?

A: We don't know the details of your support contract. You should check with your *Oracle* representative. We have heard anecdotal stories from *MySQL* Support team members that they have customers who use *Percona Server with XtraDB*, but you should not base your decision on that.

10.6.2 Q: Will we have to *GPL* our whole application if we use *Percona Server with XtraDB*?

A: This is a common misconception about the *GPL*. We suggest reading the *Free Software Foundation*'s excellent reference material on the [GPL Version 2](#), which is the license that applies to *MySQL* and therefore to *Percona Server with XtraDB*. That document contains links to many other documents which should answer your questions. *Percona* is unable to give legal advice about the *GPL*.

10.6.3 Q: Do I need to install *Percona* client libraries?

A: No, you don't need to change anything on the clients. *Percona Server* is 100% compatible with all existing client libraries and connectors.

10.6.4 Q: When using the *Percona XtraBackup* to setup a replication slave on Debian based systems I'm getting: "ERROR 1045 (28000): Access denied for user 'debian-sys-maint'@'localhost' (using password: YES)"

A: In case you're using init script on Debian based system to start `mysqld`, be sure that the password for `debian-sys-maint` user has been updated and it's the same as that user's password from the server that the backup has been taken from. Password can be seen and updated in `/etc/mysql/debian.cnf`. For more information on how to set up a replication slave using *Percona XtraBackup* see [this how-to](#).

10.7 Copyright and Licensing Information

10.7.1 Documentation Licensing

This software documentation is (C)2009-2013 Percona LLC and/or its affiliates and is distributed under the [Creative Commons Attribution-ShareAlike 2.0 Generic](#) license.

10.7.2 Software License

Percona Server is built upon *MySQL* from *Oracle*. Along with making our own modifications, we merge in changes from other sources such as community contributions and changes from *MariaDB*.

The original `SHOW USER/TABLE/INDEX` statistics code came from *Google*.

Percona does not require copyright assignment.

See the `COPYING` files accompanying the software distribution.

10.8 Percona Server 5.6 Release notes

10.8.1 Percona Server 5.6.21-69.0

Percona is glad to announce the release of *Percona Server 5.6.21-69.0* on October 7th, 2014 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on [MySQL 5.6.21](#), including all the bug fixes in it, *Percona Server 5.6.21-69.0* is the current GA release in the *Percona Server 5.6* series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the [5.6.21-69.0 milestone](#) at Launchpad.

New Features

Response Time Distribution feature has been from *Percona Server 5.5* as a plugin.

TokuDB storage engine package has been updated to version [7.5.1](#).

Bugs Fixed

Backup Locks did not guarantee consistent `SHOW SLAVE STATUS` information with binary log disabled. Bug fixed [#1358836](#).

Audit Log Plugin would rotate the audit log in middle of an audit message. Bug fixed [#1363370](#).

When the binary log is enabled on a replication slave, `SHOW SLAVE STATUS` performed under an active BINLOG lock could lead to a deadlock. Bug fixed [#1372806](#).

Fixed a memory leak in *Metrics for scalability measurement*. Bug fixed [#1334570](#).

Fixed a memory leak if *secure-file-priv* option was used with no argument. Bug fixed [#1334719](#).

`LOCK TABLES FOR BACKUP` is now incompatible with `LOCK TABLES`, `FLUSH TABLES WITH READ LOCK`, and `FLUSH TABLES FOR EXPORT` in the same connection. Bug fixed [#1360064](#).

Other bugs fixed: [#1361568](#).

10.8.2 Percona Server 5.6.20-68.0

Percona is glad to announce the release of *Percona Server 5.6.20-68.0* on August 29th, 2014 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on [MySQL 5.6.20](#), including all the bug fixes in it, *Percona Server 5.6.20-68.0* is the current GA release in the *Percona Server 5.6* series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the [5.6.20-68.0 milestone](#) at Launchpad.

New Features

Percona Server has implemented the [MySQL 5.7](#) `SHOW SLAVE STATUS NONBLOCKING` syntax for *Lock-Free SHOW SLAVE STATUS* feature. The existing `SHOW SLAVE STATUS NOLOCK` is kept as a deprecated alias and will be removed in *Percona Server 5.7*. There were no functional changes for the feature.

Percona Server Audit Log Plugin now supports JSON and CSV formats. The format choice is controlled by `audit_log_format` variable.

Percona Server Audit Log Plugin now supports *streaming the audit log to syslog*.

TokuDB storage engine package has been updated to version 7.1.8.

Bugs Fixed

Querying `INNODB_CHANGED_PAGES` with a range condition `START_LSN > x AND END_LSN < y` would lead to a server crash if the range was empty with `x` greater than `y`. Bug fixed #1202252 (*Jan Lindström* and *Sergei Petrunia*).

SQL statements of other connections were missing in the output of `SHOW ENGINE INNODB STATUS`, in `LATEST DETECTED DEADLOCK` and `TRANSACTIONS` sections. This bug was introduced by *State-ment Timeout* patch in *Percona Server 5.6.13-61.0*. Bug fixed #1328824.

Some of TokuDB distribution files were missing in the TokuDB binary tarball. Bug fixed #1338945.

With *XtraDB changed page tracking* feature enabled, queries from the `INNODB_CHANGED_PAGES` could read the bitmap data whose write was in still progress. This would cause the query to fail with an `ER_CANT_FIND_SYSTEM_REC` and a warning printed to the server error log. The workaround is to add an appropriate `END_LSN`-limiting condition to the query. Bug fixed #1193332.

`mysqld-debug` was missing from *Debian* packages. This regression was introduced in *Percona Server 5.6.16-64.0*. Bug fixed #1290087.

Fixed a memory leak in *Slow Query Log Rotation and Expiration*. Bug fixed #1314138.

The audit log plugin would write log with XML syntax errors when `OLD` and `NEW` formats were used. Bug fixed #1320879.

Combination of *Log Archiving for XtraDB*, *XtraDB changed page tracking*, and small *InnoDB* logs could hang the server on the bootstrap shutdown. Bug fixed #1326379.

`--tc-heuristic-recover` option values were broken. Bug fixed #1334330 (upstream #70860).

If the bitmap directory has a bitmap file sequence with a start LSN of one file less than a start LSN of the previous file, a debug build would assert when queries were run on `INNODB_CHANGED_PAGES` table. Bug fixed #1342494.

Other bugs fixed: #1337247, #1350386, #1208371, #1261341, #1151723, #1182050, #1182068, #1182072, #1184287, #1280875, #1338937, #1334743, #1349394, #1182046, #1182049, and #1328482 (upstream #73418).

10.8.3 Percona Server 5.6.19-67.0

Percona is glad to announce the release of *Percona Server 5.6.19-67.0* on June 30th, 2014 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on *MySQL 5.6.19*, including all the bug fixes in it, *Percona Server 5.6.19-67.0* is the current GA release in the *Percona Server 5.6* series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the *5.6.19-67.0* milestone at [Launchpad](#).

NOTE: There was no *Percona Server 5.6.18* release because there was no *MySQL Community Server 5.6.18* release. That version number was used for *MySQL Enterprise Edition* release to address the OpenSSL “Heartbleed” issue.

New Features

Percona has merged a contributed patch by Kostja Osipov implementing the *Multiple user level locks per connection* feature. This feature fixes the upstream bugs: #1118 and #67806.

TokuDB Storage engine, which is available as a separate package that can be *installed* along with the *Percona Server 5.6.19-67.0*, is now considered GA quality.

Percona Server now supports the MTR `--valgrind` option for a server that is either statically or dynamically linked with `jemalloc`.

Bugs Fixed

The `libperconaserverclient18.1` package was missing the library files. Bug fixed #1329911.

Percona Server introduced a regression in 5.6.17-66.0 when support for TokuDB storage engine was implemented. This regression caused spurious “wrong table structure” errors for `PERFORMANCE_SCHEMA` tables. Bug fixed #1329772.

Race condition in group commit code could lead to a race condition in PFS instrumentation code resulting in a server crash. Bug fixed #1309026 (upstream #72681).

Other bugs fixed: #1326348 and #1167486.

10.8.4 *Percona Server* 5.6.17-66.0

Percona is glad to announce the release of *Percona Server* 5.6.17-66.0 on June 11th, 2014 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on [MySQL 5.6.17](#), including all the bug fixes in it, *Percona Server* 5.6.17-66.0 is the current GA release in the *Percona Server* 5.6 series. All of *Percona*’s software is open-source and free, all the details of the release can be found in the 5.6.17-66.0 milestone at [Launchpad](#).

New Features

MySQL benchmarks (`sql-bench` directory in the *MySQL* distribution) has been made compatible with the TokuDB storage engine.

Percona Server has ported *MariaDB* code enhancement for *start transaction with consistent snapshot*. This enhancement makes binary log positions consistent with *InnoDB* transaction snapshots.

Percona Server has implemented ability to *clone a snapshot* created by `START TRANSACTION WITH CONSISTENT SNAPSHOT` in another session.

TokuDB Storage engine is now available as a separate package that can be *installed* along with the *Percona Server* 5.6.17-66.0. This feature is currently considered Release Candidate quality.

Percona Server version 5.6 now includes *HandlerSocket* in addition to *Percona Server* 5.5.

Bugs Fixed

Fix for #1225189 introduced a regression in *Percona Server* 5.6.13-61.0 which could lead to an error when running `mysql_install_db`. Bug fixed #1314596.

InnoDB could crash if workload contained writes to compressed tables. Bug fixed #1305364.

GUI clients such as *MySQL Workbench* could not authenticate with a user defined with `auth_pam_compat` plugin. Bug fixed #1166938.

Help in *Percona Server* 5.6 command line client was linking to *Percona Server* 5.1 manual. Bug fixed #1198775.

InnoDB redo log resizing could crash when *XtraDB changed page tracking* was enabled. Bug fixed #1204072.

Audit Log Plugin wasn't parsing escape characters correctly in the `OLD` format. Bug fixed #1313696.

Percona Server version was reported incorrectly in *Debian/Ubuntu* packages. Bug fixed #1319670.

Other bugs fixed: #1318537 (upstream #72615), #1318874, #1285618, #1272732, #1314568, #1271178, and #1323014.

10.8.5 Percona Server 5.6.17-65.0

Percona is glad to announce the release of *Percona Server* 5.6.17-65.0 on May 6th, 2014 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on [MySQL 5.6.17](#), including all the bug fixes in it, *Percona Server* 5.6.17-65.0 is the current GA release in the *Percona Server* 5.6 series. All of *Percona's* software is open-source and free, all the details of the release can be found in the 5.6.17-65.0 milestone at [Launchpad](#).

New Features

Percona Server now supports *Metrics for scalability measurement*.

Percona Server now supports *Audit Log Plugin*.

Percona Server parser and query optimizer now support *Multiple Clustering Keys* when TokuDB engine is used.

Storage engine handler interface has been extended with new calls to notify the storage engine of imminent table or index scan. The calls are used by TokuDB to improve performance of these operations.

Percona Server packages are now available for *Ubuntu* 14.04.

Bugs Fixed

Percona Server couldn't be built with *Bison* 3.0. Bug fixed #1262439, upstream #71250.

Fixed the inadequate background LRU flushing for write workloads with *InnoDB* compression that could lead to lower performance. Bug fixed #1295268.

Percona Server debug packages were not built for the previous releases. Bug fixed #1298352.

Queries that no longer exceed `long_query_time` were written to the slow query log if they matched the previous `long_query_time` value when `slow_query_log_use_global_control` variable was set to `all`. Bug fixed #1016991.

When writing audit plugins it was not possible to get notifications for general-log events without enabling the general-log. Bug fixed #1182535 (upstream #60782).

`mysqld_safe` did not correctly parse `flush_caches` and `numa_interleave` options. Bug fixed #1231110.

Thread Pool would handle a new client connection without notifying Audit Plugin. Bug fixed #1282008.

Fixed a performance issue in extending tablespaces if running under `fusionIO` with *atomic writes* enabled. Bug fixed #1286114 (*Jan Lindström*).

Previous implementation of the `log_slow_rate_type` set to `query` with `log_slow_rate_limit` feature would log every `n`th query deterministically instead of each query having a `1/n` probability to get logged. Fixed by randomly selecting the queries to be logged instead of logging every `n`th query. Bug fixed #1287650.

Percona Server source files were referencing *Maatkit* instead of *Percona Toolkit*. Bug fixed #1174779.

Maximum allowed value for `log_slow_rate_limit` was `ULONG_MAX` (ie. either 4294967295 or 18446744073709551615, depending on the platform). As it was unreasonable to configure the slow log for every four billionth session/query, new maximum allowed value is set to 1000. Bug fixed #1290714.

Other bugs fixed #1295523, #1299688 (upstream #72163) and #1272732.

10.8.6 Percona Server 5.6.16-64.2-tokudb with TokuDB engine

Percona is glad to announce the first **BETA** release of *Percona Server 5.6.16-64.2-tokudb* with TokuDB engine on March 27th, 2014. Downloads are available [here](#) and from the *Percona Software Repositories*.

Based on *Percona Server 5.6.16-64.2* including all the features and bug fixes in it, and on TokuDB 7.1.5-rc.4, *Percona Server 5.6.16-64.2-tokudb* is the first **BETA** release in the *Percona Server 5.6* with TokuDB engine series. All of *Percona's* software is open-source and free, all the details of the release can be found in the [5.6.16-64.2 milestone](#) at Launchpad.

More information on how to *install* and *use* TokuDB can be found in the documentation. This feature is currently considered BETA quality.

New Features

Installer will now automatically install TokuDB engine on new installations.

10.8.7 Percona Server 5.6.16-64.2

Percona is glad to announce the release of *Percona Server 5.6.16-64.2* on March 25th, 2014 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on [MySQL 5.6.16](#), including all the bug fixes in it, *Percona Server 5.6.16-64.2* is the current GA release in the *Percona Server 5.6* series. All of *Percona's* software is open-source and free, all the details of the release can be found in the [5.6.16-64.2 milestone](#) at Launchpad.

Bugs Fixed

The upgrade to *Percona Server 5.6.16-64.1* would silently comment out any options in `my.cnf` that have paths specified that contain `share/mysql`. Bug fixed #1293867.

Percona Server could fail to start after upgrade if the `lc-messages-dir` option was set in the `my.cnf` configuration file. Bug fixed #1294067.

Dependency on `mysql-common` package, introduced in *Percona Server 5.6.16-64.0* could lead to wrongly chosen packages for upgrade, spurious removes and installs with some combination of packages installed which use the `mysql` libraries. Bug fixed #1294211.

These three bugs were fixed by removing the dependency on `mysql-common` package.

Percona Toolkit UDFs were missing from *Debian/Ubuntu* packages, this regression was introduced in *Percona Server 5.6.16-64.0*. Bug fixed #1296416.

Percona Server installer will create the symlinks from `libmysqlclient` to `libperconaserverclient` during the installation on *CentOS*. This was implemented in order to provide the backwards compatibility after the `libmysqlclient` library has been renamed to `libperconaserverclient`.

10.8.8 Percona Server 5.6.16-64.1

Percona is glad to announce the release of *Percona Server 5.6.16-64.1* on March 17th, 2014 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on [MySQL 5.6.16](#), including all the bug fixes in it, *Percona Server 5.6.16-64.1* is the current GA release in the *Percona Server 5.6* series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the [5.6.16-64.1 milestone](#) at Launchpad.

Bugs Fixed

After installing the `auth_socket` plugin any local user might get root access to the server. If you're using this plugin upgrade is advised. This is a regression, introduced in *Percona Server 5.6.11-60.3*. Bug fixed [#1289599](#)

The new client and server packages included files with paths that were conflicting with the ones in `mysql-libs` package on *CentOS*. Bug fixed [#1278516](#).

A clean installation of `Percona-Server-server-55` on *CentOS* would fail due to a typo in `mysql_install_db` call. Bug fixed [#1291247](#).

`libperconaserverclient18.1` *Debian/Ubuntu* packages depended on `multiarch-support`, which is not available on all the supported distribution versions. Bug fixed [#1291628](#).

The *InnoDB* file system mutex was being locked incorrectly if *Atomic write support for Fusion-io devices* was enabled. Bug fixed [#1287098](#).

Slave I/O thread wouldn't attempt to automatically reconnect to the master after a network time-out (error: 1159). Bug fixed [#1268729](#) (upstream [#71374](#)).

Renaming the `libmysqlclient` to `libperconaserverclient`

This release fixes some of the issues caused by the `libmysqlclient` rename to `libperconaserverclient` in *Percona Server 5.6.16-64.0*. The old name was conflicting with the upstream `libmysqlclient`.

Except for packaging, `libmysqlclient` and `libperconaserverclient` of the same version do not have any differences. Users who previously compiled software against Percona-provided `libmysqlclient` will either need to install the corresponding package of their distribution, such as `mysql-lib` for *CentOS* and `libmysqlclient18` for *Ubuntu/Debian* or recompile against `libperconaserverclient`. Another workaround option is to create a symlink from `libperconaserverclient.so.18.0.0` to `libmysqlclient.so.18.0.0`.

10.8.9 Percona Server 5.6.16-64.0

Percona is glad to announce the release of *Percona Server 5.6.16-64.0* on March 10th, 2014 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on [MySQL 5.6.16](#), including all the bug fixes in it, *Percona Server 5.6.16-64.0* is the current GA release in the *Percona Server 5.6* series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the [5.6.16-64.0 milestone](#) at Launchpad.

New Features

Percona Server has implemented *Backup Locks* that can be used as a lightweight alternative to `FLUSH TABLES WITH READ LOCK` for taking physical and logical backups.

Percona Server has split a new *LRU manager thread* out of the InnoDB cleaner thread, that performs LRU flushes and evictions to refill the free lists.

Percona Server now has server-side support for *TokuDB Compression* option syntax.

Percona Server has implemented *Slow Query Log Rotation and Expiration* feature to provide users with better control of slow query log disk space usage.

Ability to change database for mysqlbinlog has been ported from *Percona Server 5.1*.

In order to comply with Linux distribution packaging standards *Percona's* version of `libmysqlclient` has been renamed to `libperconaserver`. The old name was conflicting with the upstream `libmysqlclient`. Except for packaging, `libmysqlclient` and `libperconaserverclient` of the same version do not have any differences. Users wishing to continue using `libmysqlclient` will have to install the corresponding package of their distribution, such as `mysql-lib` for *CentOS* and `libmysqlclient18` for *Ubuntu/Debian*. Users wishing to build software against `libperconaserverclient` should install `libperconaserverclient-dev` package. An old version of *Percona-built libmysqlclient* will be available for [download](#).

Bugs Fixed

The *XtraDB* version number in `univ.i` was incorrect. Bug fixed #1277383.

Percona Toolkit UDFs were only shipped with RPM packages. Bug fixed #1159625.

A debug server build will crash if, while performing a bulk insert to a partitioned table, one of the partitions will return a failure for `end_bulk_insert` handler call. Bug fixed #1204871 (upstream #71270).

Percona Server 5.6 installation on *Debian* would fail due to default config reference to `/etc/mysql/conf.d` which didn't exist. Bug fixed #1206648.

Due to a regression in the buffer pool mutex split, a combination of *InnoDB* compression, write workload, and multiple active purge threads could lead to a server crash. Bug fixed #1247021.

Server would crash on startup when XA support functions were activated by a second storage engine. Fix for this bug was ported from *MariaDB*. Bug fixed #1255549 (upstream #47134).

`FLUSH STATUS` could cause a server crash on the next transaction commit if two XA-supporting storage engines are in use. Fix for this bug was ported from *MariaDB*. Bugs fixed #1255551 (upstream #70854).

`auth_pam.so` shared library needed for *PAM Authentication Plugin* was missing from RPM packages. Bug fixed #1268246.

Fix for bug #1227581, a buffer pool mutex split regression, was not complete, thus a combination of write workload to *InnoDB* compressed table and a tablespace drop could crash the server. Bug fixed #1269352.

Binary RPM packages couldn't be built from source tarballs on *Fedora 19*. Bug fixed #1229598.

Percona Server compilation with Performance Schema turned off would fail due to regression introduced by the 5.6 priority mutex framework. Bug fixed #1272747.

The *InnoDB* page cleaner thread could have incorrectly decided whether the server is busy or idle on some of its iterations and consequently issue a too big flush list flushing request on a busy server, causing performance instabilities. Bug fixed #1238039 (upstream #71988).

Percona Server had different server version value when installing from Source and from Binary/RPM. Bug fixed #1244178.

InnoDB did not handle the cases of asynchronous and synchronous I/O requests completing partially or being interrupted. Bugs fixed #1262500 (upstream #54430), and #1263087 (*Andrew Gaul*).

The fix for upstream bug #70768 may cause a high rate of RW lock creations and destructions, resulting in a performance regression on some workloads. Bug fixed #1279671 (upstream #71708).

Debian and *Ubuntu* packaging has been reworked to meet the packaging standards. Bug fixed #1281261.

Fixed the CMake warnings that were happening when Makefile was generated. Bugs fixed #1274827, upstream bug fixed #71089 and #1274411 (upstream #71094).

On *Ubuntu* Precise multiple architecture versions of `libmysqlclient18` couldn't be installed side by side. Bug fixed #1052636.

Percona Server source tree has been reorganized to match the *MySQL* source tree layout closer. Bug fixed #1014477.

Performance schema autosizing heuristics have been updated to account for *Percona Server*-specific `wait/synch/mutex/sql/THD::LOCK_temporary_tables` mutex. Bug fixed #1264952.

Database administrator password could be seen in plain text if when `debconf-get-selections` was executed. Bug fixed #1018291.

Other bugs fixed: #1276445, #1005787, #1285064, #1229598, and #1277505 (upstream #71624).

10.8.10 *Percona Server* 5.6.16-64.0-tokudb with TokuDB engine

Percona is glad to announce the first **ALPHA** release of *Percona Server* 5.6.16-64.0-tokudb with TokuDB engine on March 3rd, 2014. Downloads are available [here](#) and from the *Percona Software Repositories*.

Based on *Percona Server* 5.6.16-64.0 including all the features and bug fixes in it, and on TokuDB 7.1.5-rc.3, *Percona Server* 5.6.16-64.0-tokudb is the first **ALPHA** release in the *Percona Server* 5.6 with TokuDB series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.16-64.0 milestone at [Launchpad](#).

New Features

Percona Server now supports TokuDB storage engine. More information on how to *install* and *use* TokuDB can be found in the documentation. This feature is currently considered ALPHA quality.

Available TokuDB features:

- *TokuDB Compression*

Currently available ALPHA features ¹:

- Multiple Clustering Indexes
- Fast Updates with NOAR
- Hot Table Optimization
- TokuDB AUTOINCREMENT implementation
- Prelocking index and range scans

10.8.11 *Percona Server* 5.6.15-63.0

Percona is glad to announce the release of *Percona Server* 5.6.15-63.0 on December 19th, 2013 (Downloads are available [here](#) and from the *Percona Software Repositories*).

¹ These features are available only in ALPHA *Percona Server* with TokuDB builds. They might change or even disappear in a future release.

Based on MySQL 5.6.15, including all the bug fixes in it, *Percona Server* 5.6.15-63.0 is the current GA release in the *Percona Server* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.15-63.0 milestone at Launchpad.

New Features

Thread Pool high priority scheduling is now enabled by default by changing the default `thread_pool_high_prio_tickets` value from 0 to 4294967295.

Percona Server now supports *Low priority queue throttling*. This feature should improve *Thread Pool* performance under high concurrency in a situation when thread groups are oversubscribed.

Introduced new `thread_pool_high_prio_mode` to provide more fine-grained control over high priority scheduling either globally or per connection in *Thread Pool*.

Percona Server has *extended mysqlbinlog* to provide SSL and compression support.

Percona Server has reduced the performance overhead of the *User Statistics* feature.

Bugs Fixed

INSTALL PLUGIN statement would crash server if *User Statistics* were enabled. Bug fixed #1011047.

`innodb_log_checksum_algorithm` did not have any effect when set from `my.cnf` or `mysql` command line, it would have effect only if set dynamically. Bug fixed #1248505.

Server would crash on shutdown if *Atomic write support for Fusion-io devices* feature is enabled. Bug fixed #1255628 (*Jan Lindström*).

Percona Server would crash when data was select from `XTRADB_RSEG` table when *InnoDB* system table space was initialized with lower then default number of rollback segments. Bug fixed #1260259.

Certain types of workloads (large result sets, blobs, slow clients) can have longer waits on network I/O (socket reads and writes). Whenever server waits, this should be communicated to the *Thread Pool*, so it can start new query by either waking a waiting thread or sometimes creating a new one. Ported *MariaDB* patch MDEV-156, bug fixed #1159743.

`mysqldump --innodb-optimize-keys` was generating incorrect CREATE TABLE statements for partitioned tables. Bug fixed #1233841.

Fixed errors when server was compiled with `-DWITH_LIBWRAP=ON` option. Bug fixed #1240442.

If `innobase_atomic_writes` was used on separate undo files that do not exists would lead to operating system error. Bug fixed #1255638 (*Jan Lindström*).

Default value for `thread_pool_max_threads` has been changed from 500 to 100 000 (the maximum supported number of connections), because limiting the total number of threads in the *Thread Pool* can result in deadlocks and uneven distribution of worker threads between thread groups in case of stalled connections. Bug fixed #1258097.

`PURGE CHANGED_PAGE_BITMAPS BEFORE` statement would delete the changed page data after the specified LSN and up to the start of the next bitmap file. If this data were to be used for fast incremental backups, its absence would cause *Percona XtraBackup* to fall back to the full-scan incremental backup mode. Bug fixed #1260035 (*Andrew Gaul*).

Server performance could degrade under heavy load or it could deadlock on shutdown while performing purge. Bug fixed #1236696.

Server could crash under heavy load if *InnoDB* compression was used. Bug fixed #1240371.

Redo log checksum mismatches would be diagnosed using the data page checksum algorithm setting instead of redo log checksum algorithm one. Bug fixed #1250148.

Other bugs fixed: bug #1082333, bug #1260945, bug #1248046, bug #1243067, bug #1238563, bug #1258154 (upstream bug #71092), bug #1242748, bug #1239062, bug #1200788, bug #1193319, bug #1240044.

10.8.12 Percona Server 5.6.14-62.0

Percona is glad to announce the release of *Percona Server* 5.6.14-62.0 on October 24th, 2013 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on *MySQL* 5.6.14, including all the bug fixes in it, *Percona Server* 5.6.14-62.0 is the current GA release in the *Percona Server* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.14-62.0 milestone at [Launchpad](#).

New Features

Percona Server has implemented more efficient log block checksums with new `innodb_log_checksum_algorithm` variable.

Percona Server has implemented support for *Per-query variable statement*.

Bugs Fixed

Percona Server could crash while accessing BLOB or TEXT columns in *InnoDB* tables if *Support for Fake Changes* was enabled. Bug fixed #1188168.

Expanded Program Option Modifiers did not deallocate memory correctly. Bug fixed #1167487.

Some *Expanded Program Option Modifiers* didn't have an effect if they were specified in non-normalized way (`innodb_io_capacity` vs `innodb-io-capacity`). Bug fixed #1233294.

Building *Percona Server* with `-DHAVE_PURIFY` option would result in an error. Fixed by porting the `close_socket` function from *MariaDB*. Bug fixed #1203567.

Enabling *Enforcing Storage Engine* feature could lead to error on *Percona Server* shutdown. Bug fixed #1233354.

Storage engine enforcement (`enforce_storage_engine`) is now ignored when the server is started in either bootstrap or skip-grant-tables mode. Bug fixed #1236938.

When installed *Percona Server* 5.6 GA release was still showing RC instead of GA on Debian-based systems. Bug fixed #1239418.

Other bugs fixed: bug fixed #1238008, bug fixed #1190604, bug fixed #1200162, bug fixed #1188172, and bug fixed #1214727.

10.8.13 Percona Server 5.6.13-61.0

Percona is glad to announce the first GA (Generally Available) release of *Percona Server* 5.6.13-61.0 on October 7th, 2013 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on *MySQL* 5.6.13, including all the bug fixes in it, *Percona Server* 5.6.13-61.0 is the first GA release in the *Percona Server* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.13-61.0 milestone at [Launchpad](#).

New Features

Percona Server has implemented a number of *XtraDB performance improvements for I/O-bound high-concurrency workloads*. This feature fixes the upstream bug: #68555 (#1236884).

Percona Server has implemented a number of performance improvements for *Page cleaner thread tuning*. This feature fixes the upstream bugs: #69170 (#1231918), #70453 (#1232101) and #68481 (#1232406).

ALL_O_DIRECT method for `innodb_flush_method` has been ported from *Percona Server 5.5*.

Statement Timeout feature has been ported from the Twitter branch.

Percona Server has *extended* the `SELECT INTO ... OUTFILE` and `SELECT INTO DUMPFILE` to add the support for UNIX sockets and named pipes.

Bugs Fixed

Due to an incompatible upstream change that went in unnoticed, the log tracker thread would attempt to replay any file operations it encountered. In most cases this were a no-op, but there were race conditions for certain DDL operations that would have resulted in server crash. Bug fixed #1217002.

`apt-get upgrade` of *Percona Server* would fail in post-installation step if server failed to start. Bug fixed #1002500.

Percona Server 5.6 now ships with memcached plugins. Bug fixed #1159621.

Fixed the `libssl.so.6` dependency issues in binary tarballs releases. Bug fixed #1172916.

Error in `install_layout.cmake` could cause that some library files, during the build, end up in different directories on `x86_64` environment. Bug fixed #1174300.

Server would crash if empty string was passed to `AES_ENCRYPT` when older OpenSSL version was used. Upstream bug fixed #70489, bug fixed #1201033.

Kill Idle Transactions feature didn't work correctly if *Thread Pool* was enabled. Bug fixed #1201440.

Percona Server 5.6.12-60.4 would crash if server was started with *Thread Pool* feature enabled. Bugs fixed #1201681, #1194097 and #1201442.

Memory leak was introduced by the fix for bug #1132194. Bug fixed #1204873.

A server could have crashed under a heavy I/O-bound workload involving compressed InnoDB tables. Bug fixed #1224432.

A potential deadlock, involving `DDL`, `SELECT`, `SHOW ENGINE INNODB STATUS`, and `KILL`, has been fixed. Fixed the upstream bug #60682, bug fixed #1115048.

A memory leak in *Utility user* feature has been fixed. Bug fixed #1166638.

A server could crash due to a race condition between a `INNODB_CHANGED_PAGES` query and bitmap file delete by `PURGE CHANGED_PAGE_BITMAP` or directly on the file system. Bug fixed #1191580.

Percona Server could not be built with *Thread Pool* feature and `-DWITH_PERFSHEMA_ENGINE=OFF` option. Bug fixed #1196383.

Page cleaner should perform LRU flushing regardless of server activity. Fixed the upstream bug #70500, bug fixed #1234562.

Fixed the upstream bug #64556 which could cause an unrelated warning to be raised if a query inside *InnoDB* was interrupted. Bug fixed #1115158.

Other bugs fixed: bug fixed #1131949, bug fixed #1191589, bug fixed #1229583, upstream bug fixed #70490 bug fixed #1205196, upstream bug fixed #70417 bug fixed #1230220.

10.8.14 Percona Server 5.6.13-60.6

Percona is glad to announce the fourth Release Candidate release of *Percona Server* 5.6.13-60.6 on September 20th, 2013 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on [MySQL 5.6.13](#), including all the bug fixes in it, *Percona Server* 5.6.13-60.6 is the fourth RC release in the *Percona Server* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.13-60.6 milestone at [Launchpad](#).

New Features

Improved Buffer Pool Scalability feature has been ported from *Percona Server* 5.5. This feature splits the single global InnoDB buffer pool mutex into several mutexes. The goal of this change is to reduce mutex contention, which can be very impacting when the working set does not fit in memory.

Multiple Adaptive Hash Search Partitions feature has been ported from *Percona Server* 5.5. This feature splits the adaptive hash index across several partitions and decreases the AHI latch contention. This feature fixes the upstream bug [#62018](#) ([#1216804](#)).

Utility user feature has been extended by adding a new `utility_user_privileges` that allows a comma separated value list of extra access privileges that can be granted to the utility user.

Percona Server now provides additional information in the slow query log when `log_slow_rate_limit` variable is enabled.

A new variable `slow_query_log_always_write_time` has been introduced. It can be used to specify an additional execution time threshold for the slow query log, that, when exceeded, will cause a query to be logged unconditionally, that is, `log_slow_rate_limit` will not apply to it.

Bugs Fixed

The unnecessary overhead from persistent InnoDB adaptive hash index latching has been removed, potentially improving stability of the *Multiple Adaptive Hash Search Partitions* feature as well. Upstream bug fixed [#70216](#), bug fixed [#1218347](#).

Adaptive hash index memory size was incorrectly calculated in `SHOW ENGINE INNODB STATUS` and `XTRADB_INTERNAL_HASH_TABLES`. Bug fixed [#1218330](#).

An unnecessary buffer pool mutex acquisition has been removed, potentially improving performance. Upstream bug fixed [#69258](#), bug fixed [#1219842](#).

Fixed the build warnings caused by *User Statistics* code on non-Linux platforms. Bug fixed [#711817](#).

Adaptive hash indexing partitioning code has been simplified, potentially improving performance. Bug fixed [#1218321](#).

Other bugs fixed: upstream bug fixed [#69617](#) bug fixed [#1216815](#), upstream bug fixed [#70228](#) bug fixed [#1220544](#).

10.8.15 Percona Server 5.6.13-60.5

Percona is glad to announce the third Release Candidate release of *Percona Server* 5.6.13-60.5 on August 28th, 2013 (Downloads are available [here](#) and from the *Percona Software Repositories*).

Based on [MySQL 5.6.13](#), including all the bug fixes in it, *Percona Server* 5.6.13-60.5 is the third RC release in the *Percona Server* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.13-60.5 milestone at [Launchpad](#).

Ported Features

Fixed Size for the Read Ahead Area has been ported from *Percona Server 5.5*

Bug Fixes

If binary log was enabled, *Fake Changes* transactions were binlogged. This could lead to data corruption issues with deeper replication topologies. Bug fixed #1190580.

Querying `INFORMATION_SCHEMA.PARTITIONS` could cause key distribution statistics for partitioned tables to be reset to those corresponding to the last partition. Fixed the upstream bug #69179. Bug fixed #1192354.

Changes made to the RPM scripts for previous *Percona Server* version caused installer to fail if there were different *datadir* options in multiple configuration files. Bug fixed #1201036.

Fixed the upstream bug #42415 that would cause `UPDATE/DELETE` statements with the `LIMIT` clause to be unsafe for Statement Based Replication even when `ORDER BY` primary key was present. Fixed by implementing an algorithm to do more elaborate analysis on the nature of the query to determine whether the query will cause uncertainty for replication or not. Bug fixed #1132194.

When an upgrade was performed between major versions (e.g. by uninstalling a 5.1 RPM and then installing a 5.5 one), `mysql_install_db` was still called on the existing data directory which lead to re-creation of the `test` database. Bug fixed #1169522.

Fixed the upstream bug #69639 which caused compile errors for *Percona Server* with `DTrace` version `Sun D 1.11` provided by recent `SmartOS` versions. Bug fixed #1196460.

Fixed a regression introduced in *Percona Server 5.6.12-60.4*, where server wouldn't be able to start if *Atomic write support for Fusion-io devices* was enabled. Bug fixed #1214735.

Other bugs fixed: bug fixed #1188162, bug fixed #1203308 and bug fixed #1189743.

10.8.16 Percona Server 5.6.12-60.4

Percona is glad to announce the second Release Candidate release of *Percona Server 5.6.12-60.4* on June 27th, 2013 (Downloads are available [here](#) and from the [Percona Software Repositories](#)).

Based on [MySQL 5.6.12](#), including all the bug fixes in it, *Percona Server 5.6.12-60.4* is the second RC release in the *Percona Server 5.6* series. All of *Percona's* software is open-source and free, all the details of the release can be found in the [5.6.12-60.4 milestone](#) at [Launchpad](#).

New Features

Percona Server has implemented support for supplementary groups for *PAM Authentication Plugin*.

Bug Fixes

`mysql_install_db` did not work properly in debug and *Valgrind* builds. Bug fixed #1179359.

Fixed yum dependencies that were causing conflicts in CentOS 6.3 during installation. Bug fixed #1051874.

The RPM installer script had the *datadir* hardcoded to `/var/lib/mysql` instead of using `my_print_defaults` function to get the correct *datadir* info. Bug fixed #1181753.

Fixed the upstream bug #68354 that could cause server to crash when performing update or join on Federated and MyISAM tables with one row, due to incorrect interaction between Federated storage engine and the optimizer. Bug fixed #1182572.

Fixed the compiler warnings caused by *Atomic write support for Fusion-io devices* when building *Percona Server* on non-Linux platforms. Bug fixed #1189429.

Other bugs fixed: bug fixed #1188516, bug fixed #1130731 and bug fixed #1133266.

10.8.17 Percona Server 5.6.11-60.3

Percona is glad to announce the first Release Candidate release of *Percona Server 5.6.11-60.3* on June 3rd, 2013 (Downloads are available [here](#) and from the [Percona Software Repositories](#)).

Based on [MySQL 5.6.11](#), including all the bug fixes in it, *Percona Server 5.6.11-60.3* is the first RC release in the *Percona Server 5.6* series. All of *Percona's* software is open-source and free, all the details of the release can be found in the [5.6.11-60.3 milestone](#) at [Launchpad](#).

This release contains all the bug fixes from latest *Percona Server 5.5* release (currently *Percona Server 5.5.31-30.3*).

New Features

Percona Server has implemented *Log Archiving for XtraDB*. Currently this feature implementation is considered *ALPHA*.

Ported Features

Percona Server has ported priority connection scheduling for the *Thread Pool* from *Percona Server 5.5*.

Percona Server has ported the *Atomic write support for Fusion-io devices* patch from *MariaDB*. This feature adds atomic write support for `directFS` filesystem on `Fusion-io` devices. This feature implementation is considered *BETA* quality.

Percona Server has ported `innodb_read_views_memory` and `innodb_descriptors_memory` status variables in the *Extended Show Engine InnoDB Status* to improve *InnoDB* memory usage diagnostics.

Improved InnoDB I/O Scalability has been ported from *Percona Server 5.5*

Improved NUMA support has been ported from *Percona Server 5.5*

Suppress Warning Messages has been ported from *Percona Server 5.5*

Improved MEMORY Storage Engine has been ported from *Percona Server 5.5*

Restricting the number of binlog files has been ported from *Percona Server 5.5*

Too Many Connections Warning has been ported from *Percona Server 5.5*

Error Code Compatibility has been ported from *Percona Server 5.5*

Lock-Free SHOW SLAVE STATUS has been ported from *Percona Server 5.5*

Percona Toolkit UDFs has been ported from *Percona Server 5.5*

Support for Fake Changes has been ported from *Percona Server 5.5*

Kill Idle Transactions has been ported from *Percona Server 5.5*

Enforcing Storage Engine has been ported from *Percona Server 5.5*

Utility user has been ported from *Percona Server 5.5*

Extending the secure-file-priv server option has been ported from *Percona Server 5.5*

Expanded Program Option Modifiers has been ported from *Percona Server 5.5*

XtraDB changed page tracking has been ported from *Percona Server 5.5*

PAM Authentication Plugin has been ported from *Percona Server 5.5*

User Statistics has been ported from *Percona Server 5.5*

Slow Query Log has been ported from *Percona Server 5.5*

Extended Show Engine InnoDB Status has been ported from *Percona Server 5.5*

Count InnoDB Deadlocks has been ported from *Percona Server 5.5*

Log All Client Commands (syslog) has been ported from *Percona Server 5.5*

Show Storage Engines has been ported from *Percona Server 5.5*

Thread Based Profiling has been ported from *Percona Server 5.5*

Bug Fixes

Transaction objects are now allocated calling `calloc()` directly instead of using *InnoDB* heap allocation. This may improve write performance for high levels of concurrency. Bug fixed #1185686.

Under very rare circumstances, deleting a zero-size bitmap file at the right moment would make server stop with an I/O error if changed page tracking is enabled. Bug fixed #1184517.

Missing path separator between the directory and file name components in a bitmap file name could stop the server starting if the `innodb_data_home_dir` variable didn't have the path separator at the end. Bug fixed #1181887.

Changed page tracking used to hold the log system mutex for the log reads needlessly, potentially limiting performance on write-intensive workloads. Bug fixed #1171699.

Incorrect schema definition for the *User Statistics* tables in `INFORMATION_SCHEMA` (`CLIENT_STATISTICS`, `INDEX_STATISTICS`, `TABLE_STATISTICS`, `THREAD_STATISTICS`, and `USER_STATISTICS`) led to the maximum counter values being limited to 32-bit signed integers. Fixed so that these values can be 64-bit unsigned integers now. Bug fixed #714925.

Server would crash if an `INNODB_CHANGED_PAGES` query is issued that has an empty LSN range and thus does not need to read any bitmap files. Bug fixed #1184427.

Query to the `INNODB_CHANGED_PAGES` table would cause server to stop with an I/O error if a bitmap file in the middle of requested LSN range was missing. Bug fixed #1179974.

A warning is now returned if a bitmap file I/O error occurs after an `INNODB_CHANGED_PAGES` query started returning data to indicate an incomplete result set. Bug fixed #1185040.

The `INNODB_CHANGED_PAGES` table couldn't be queried if the log tracker wasn't running. Bug fixed #1185304.

Fixed the upstream bug #68970 that, in *Percona Server*, would cause small tablespaces to expand too fast around 500KB tablespace size. Bug fixed #1169494.

Fixed the RPM package dependencies issues. Bug fixed #1186831.

Reduced the overhead from *Handle Corrupted Tables* check as it was missing branch predictor annotations. Bug fixed #1176864.

Other bugs fixed: bug fixed #1184695, bug fixed #1184512, bug fixed #1183585, bug fixed #1178606, bug fixed #1177356, bug fixed #1160895, bug fixed #1182876, bug fixed #1180481, bug fixed #1163135, bug fixed #1157078, bug fixed #1182889, bug fixed #1133926, bug fixed #1165098, bug fixed #1182793, bug fixed #1157075, bug fixed #1183625, bug fixed #1155475, bug fixed #1157037, bug fixed #1182065, bug fixed #1182837, bug fixed #1177780, bug fixed #1154954.

10.8.18 Percona Server 5.6.10-60.2

Percona is glad to announce the Alpha release of *Percona Server 5.6.10-60.2* on March 11, 2013 (Downloads are available [here](#) and from the [EXPERIMENTAL Percona Software Repositories](#)).

Based on [MySQL 5.6.10](#), including all the bug fixes in it, *Percona Server 5.6.10-60.2* is the third ALPHA release in the *Percona Server 5.6* series. All of *Percona's* software is open-source and free, all the details of the release can be found in the [5.6.10-60.2 milestone at Launchpad](#).

New Features

Expanded Fast Index Creation has been ported from *Percona Server 5.5*

Ported the *Thread Pool* patch from *MariaDB*. This feature enables the server to keep the top performance even with the increased number of client connections.

Handle Corrupted Tables has been ported from *Percona Server 5.5*

Bug Fixes

Fixed the upstream [#68116](#) that caused the server crash with assertion error when *InnoDB* monitor with verbose lock info was used under heavy load. This bug is affecting only `-debug` builds. Bug fixed [#1100178](#) (*Laurynas Biveinis*).

Fixed the extra `libtinfo` package dependency in *Percona Server* tarballs. Bug fixed [#1153950](#) (*Ignacio Nin*).

10.8.19 Percona Server 5.6.6-60.1

Percona is glad to announce the ALPHA release of *Percona Server 5.6.6-60.1* on August 27, 2012 (Downloads are available [here](#) and from the [EXPERIMENTAL Percona Software Repositories](#)).

Based on [MySQL 5.6.6](#), including all the bug fixes in it, *Percona Server 5.6.6-60.1* is the second ALPHA release in the *Percona Server 5.6* series. All of *Percona's* software is open-source and free, all the details of the release can be found in the [5.6.6-60.1 milestone at Launchpad](#).

This release does NOT include all the features and fixes in *Percona Server 5.5*. A list of features included is provided below.

Features removed

- `fast_index_creation` variable has been removed, it is replaced by the MySQL 5.6's `ALGORITHM=` option to `ALTER TABLE`
- `HandlerSocket` has been removed from the tree. It was not built with the `5.6.5-60.0` binaries and may return when `HandlerSocket` supports MySQL 5.6.

- `SHOW [GLOBAL] TEMPORARY TABLES` functionality is now only available via the `TEMPORARY_TABLES` and `GLOBAL_TEMPORARY_TABLES` in the `INFORMATION_SCHEMA` database.

10.8.20 Percona Server 5.6.5-60.0

Percona is glad to announce the ALPHA release of *Percona Server 5.6.5-60.0* on August 15, 2012 (Downloads are available [here](#) and from the [EXPERIMENTAL Percona Software Repositories](#)).

Based on [MySQL 5.6.5](#), including all the bug fixes in it, *Percona Server 5.6.5-60.0* is the first ALPHA release in the 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the [5.6.5-60.0 milestone](#) at [Launchpad](#).

This release does NOT include all the features and fixes in *Percona Server 5.5*. A list of features included is provided below.

Features Included

- Fixed upstream *MySQL* bug [#49336](#), `mysqlbinlog` couldn't handle `stdin` when "I" was used. Bug fixed: [#933969](#) (*Sergei Glushchenko*).
- Fixed upstream *MySQL* bug [#65946](#), which could cause the server to segfault if using `libeatmydata` (and possibly other `LD_PRELOADs`). Bug fixed: [#933969](#) (*Stewart Smith*).
- Added the `TIME_MS` column in the `PROCESSLIST` table
- *Ignoring missing tables in mysqldump* feature has been ported from *Percona Server 5.5*
- `fast_index_creation` feature has been ported from *Percona Server 5.5*
- *Temporary tables* information have been added to the `INFORMATION_SCHEMA` database

Features removed

- The `optimizer_fix` variable from *Percona Server 5.5* will not be present in 5.6

10.9 Glossary

ACID Set of properties that guarantee database transactions are processed reliably. Stands for *Atomicity*, *Consistency*, *Isolation*, *Durability*.

Atomicity Atomicity means that database operations are applied following a "all or nothing" rule. A transaction is either fully applied or not at all.

Consistency Consistency means that each transaction that modifies the database takes it from one consistent state to another.

Durability Once a transaction is committed, it will remain so.

Foreign Key A referential constraint between two tables. Example: A purchase order in the `purchase_orders` table must have been made by a customer that exists in the `customers` table.

Isolation The Isolation requirement means that no transaction can interfere with another.

InnoDB A *Storage Engine* for *MySQL* and derivatives (*Percona Server*, *MariaDB*) originally written by Innobase Oy, since acquired by Oracle. It provides *ACID* compliant storage engine with *foreign key* support. As of *MySQL* version 5.5, InnoDB became the default storage engine on all platforms.

Jenkins *Jenkins* is a continuous integration system that we use to help ensure the continued quality of the software we produce. It helps us achieve the aims of:

- no failed tests in trunk on any platform,
- aid developers in ensuring merge requests build and test on all platforms,
- no known performance regressions (without a damn good explanation).

LSN Log Serial Number. A term used in relation to the *InnoDB* or *XtraDB* storage engines.

MariaDB A fork of *MySQL* that is maintained primarily by Monty Program AB. It aims to add features, fix bugs while maintaining 100% backwards compatibility with MySQL.

my.cnf The file name of the default MySQL configuration file.

MyISAM A *MySQL Storage Engine* that was the default until MySQL 5.5.

MySQL An open source database that has spawned several distributions and forks. MySQL AB was the primary maintainer and distributor until bought by Sun Microsystems, which was then acquired by Oracle. As Oracle owns the MySQL trademark, the term MySQL is often used for the Oracle distribution of MySQL as distinct from the drop-in replacements such as *MariaDB* and *Percona Server*.

Percona Server Percona's branch of *MySQL* with performance and management improvements.

Storage Engine A *Storage Engine* is a piece of software that implements the details of data storage and retrieval for a database system. This term is primarily used within the *MySQL* ecosystem due to it being the first widely used relational database to have an abstraction layer around storage. It is analogous to a Virtual File System layer in an Operating System. A VFS layer allows an operating system to read and write multiple file systems (e.g. FAT, NTFS, XFS, ext3) and a Storage Engine layer allows a database server to access tables stored in different engines (e.g. *MyISAM*, *InnoDB*).

XtraDB Percona's improved version of *InnoDB* providing performance, features and reliability above what is shipped by Oracle in *InnoDB*.

- *genindex*
- *modindex*
- *search*

Symbols

5.6.10-60.2 (release notes), 176
 5.6.11-60.3 (release notes), 174
 5.6.12-60.4 (release notes), 173
 5.6.13-60.5 (release notes), 172
 5.6.13-60.6 (release notes), 171
 5.6.13-61.0 (release notes), 170
 5.6.14-62.0 (release notes), 170
 5.6.15-63.0 (release notes), 168
 5.6.16-64.0 (release notes), 166
 5.6.16-64.0-tokudb (release notes), 168
 5.6.16-64.1 (release notes), 165
 5.6.16-64.2 (release notes), 165
 5.6.16-64.2-tokudb (release notes), 165
 5.6.17-65.0 (release notes), 164
 5.6.17-66.0 (release notes), 163
 5.6.19-67.0 (release notes), 162
 5.6.20-68.0 (release notes), 161
 5.6.21-69.0 (release notes), 161
 5.6.5-60.0 (release notes), 177
 5.6.6-60.1 (release notes), 176

A

ACID, 177
 Atomicity, 177
 audit_log_buffer_size (variable), 92
 audit_log_file (variable), 91
 audit_log_flush (variable), 92
 audit_log_format (variable), 92
 audit_log_handler (variable), 93
 audit_log_policy (variable), 92
 audit_log_rotate_on_size (variable), 93
 audit_log_rotations (variable), 93
 audit_log_strategy (variable), 91
 audit_log_syslog_facility (variable), 94
 audit_log_syslog_ident (variable), 94
 audit_log_syslog_priority (variable), 94

B

Binlog_snapshot_file (variable), 96
 Binlog_snapshot_position (variable), 96

C

CLIENT_STATISTICS (table), 98
 Com_lock_binlog_for_backup (variable), 88
 Com_lock_tables_for_backup (variable), 88
 Com_unlock_binlog (variable), 88
 Consistency, 177

D

Durability, 177

E

enforce_storage_engine (variable), 70
 expand_fast_index_creation (variable), 82
 extra_max_connections (variable), 41
 extra_port (variable), 41

F

flush_caches (variable), 34
 Foreign Key, 177

G

GLOBAL_TEMPORARY_TABLES (table), 123

H

have_backup_locks (variable), 87
 have_snapshot_cloning (variable), 96
 have_statement_timeout (variable), 84

I

INDEX_STATISTICS (table), 99
 InnoDB, 177
 innodb_adaptive_hash_index_partitions (variable), 29
 Innodb_background_log_sync (variable), 111
 Innodb_buffer_pool_pages_LRU_flushed (variable), 116
 Innodb_buffer_pool_pages_made_not_young (variable), 116
 Innodb_buffer_pool_pages_made_young (variable), 117
 Innodb_buffer_pool_pages_old (variable), 117
 innodb_buffer_pool_populate (variable), 33
 INNODB_CHANGED_PAGES (table), 77
 Innodb_checkpoint_age (variable), 115

InnoDB_checkpoint_max_age (variable), 115
innodb_cleaner_eviction_factor (variable), 44
innodb_cleaner_flush_chunk_size (variable), 44
innodb_cleaner_free_list_lwm (variable), 44
innodb_cleaner_lru_chunk_size (variable), 43
innodb_cleaner_lsn_age_factor (variable), 43
innodb_cleaner_max_flush_time (variable), 43
innodb_cleaner_max_lru_time (variable), 43
innodb_corrupt_table_action (variable), 64
InnoDB_current_row_locks (variable), 118
InnoDB_deadlocks (variable), 120
InnoDB_descriptors_memory (variable), 117
innodb_empty_free_list_algorithm (variable), 45
innodb_fake_changes (variable), 68
innodb_flush_method (variable), 27
innodb_foreground_preflush (variable), 46
InnoDB_history_list_length (variable), 118
InnoDB_ibuf_discarded_delete_marks (variable), 113
InnoDB_ibuf_discarded_deletes (variable), 113
InnoDB_ibuf_discarded_inserts (variable), 113
InnoDB_ibuf_free_list (variable), 113
InnoDB_ibuf_merged_delete_marks (variable), 113
InnoDB_ibuf_merged_deletes (variable), 113
InnoDB_ibuf_merged_inserts (variable), 114
InnoDB_ibuf_merges (variable), 114
InnoDB_ibuf_segment_size (variable), 114
InnoDB_ibuf_size (variable), 114
innodb_kill_idle_transaction (variable), 70
innodb_locking_fake_changes (variable), 69
innodb_log_arch_dir (variable), 83
innodb_log_arch_expire_sec (variable), 83
innodb_log_archive (variable), 83
innodb_log_block_size (variable), 27
innodb_log_checksum_algorithm (variable), 28
InnoDB_lsn_current (variable), 115
InnoDB_lsn_flushed (variable), 115
InnoDB_lsn_last_checkpoint (variable), 115
InnoDB_master_thread_active_loops (variable), 111
InnoDB_master_thread_idle_loops (variable), 111
innodb_max_bitmap_file_size (variable), 78
innodb_max_changed_pages (variable), 77
InnoDB_max_trx_id (variable), 118
InnoDB_mem_adaptive_hash (variable), 116
InnoDB_mem_dictionary (variable), 116
InnoDB_mem_total (variable), 116
InnoDB_mutex_os_waits (variable), 111
InnoDB_mutex_spin_rounds (variable), 111
InnoDB_mutex_spin_waits (variable), 112
InnoDB_oldest_view_low_limit_trx_id (variable), 118
innodb_priority_cleaner (variable), 48
innodb_priority_io (variable), 48
innodb_priority_master (variable), 48
innodb_priority_purge (variable), 48
InnoDB_purge_trx_id (variable), 118
InnoDB_purge_undo_no (variable), 118
InnoDB_read_views_memory (variable), 117
InnoDB_s_lock_os_waits (variable), 112
InnoDB_s_lock_spin_rounds (variable), 112
InnoDB_s_lock_spin_waits (variable), 112
innodb_sched_priority_cleaner (variable), 46
innodb_sched_priority_io (variable), 47
innodb_sched_priority_master (variable), 47
innodb_sched_priority_purge (variable), 47
innodb_show_locks_held (variable), 110
innodb_show_verbose_locks (variable), 110
innodb_track_changed_pages (variable), 78
innodb_use_atomic_writes (variable), 31
innodb_use_global_flush_log_at_trx_commit (variable), 26
InnoDB_x_lock_os_waits (variable), 112
InnoDB_x_lock_spin_rounds (variable), 112
InnoDB_x_lock_spin_waits (variable), 112
Isolation, **177**

J

Jenkins, **178**

L

lock-for-backup (option), 88
log_slow_filter (variable), 103
log_slow_rate_limit (variable), 104
log_slow_rate_type (variable), 104
log_slow_sp_statements (variable), 105
log_slow_verbosity (variable), 105
log_warnings_suppress (variable), 51
LSN, **178**

M

MariaDB, **178**
max_binlog_files (variable), 56
max_slowlog_files (variable), 59
max_slowlog_size (variable), 59
max_statement_time (variable), 84
Max_statement_time_exceeded (variable), 85
Max_statement_time_set (variable), 85
Max_statement_time_set_failed (variable), 85
my.cnf, **178**
MyISAM, **178**
MySQL, **178**

N

numa_interleave (variable), 34

P

Percona Server, **178**
PROCESSLIST (table), 122

Q

query_cache_strip_comments (variable), 33
 QUERY_RESPONSE_TIME (table), 131
 query_response_time_flush (variable), 130
 query_response_time_range_base (variable), 130
 query_response_time_stats (variable), 130

R

rewrite-db (option), 61

S

scalability_metrics_busytime (variable), 127
 scalability_metrics_concurrency (variable), 126
 scalability_metrics_control (variable), 126
 scalability_metrics_elapsedtime (variable), 126
 scalability_metrics_queries (variable), 126
 scalability_metrics_totaltime (variable), 127
 secure-file-priv (variable), 73
 slow_query_log_always_write_time (variable), 107
 slow_query_log_timestamp_always (variable), 106
 slow_query_log_timestamp_precision (variable), 106
 slow_query_log_use_global_control (variable), 106
 Storage Engine, **178**
 syslog (variable), 121

T

TABLE_STATISTICS (table), 100
 TEMPORARY_TABLES (table), 124
 thread_pool_high_prio_mode (variable), 39
 thread_pool_high_prio_tickets (variable), 40
 thread_pool_idle_timeout (variable), 39
 thread_pool_max_threads (variable), 40
 thread_pool_oversubscribe (variable), 40
 thread_pool_size (variable), 40
 thread_pool_stall_limit (variable), 41
 THREAD_STATISTICS (table), 100
 thread_statistics (variable), 97
 Threadpool_idle_threads (variable), 41
 Threadpool_threads (variable), 42

U

USER_STATISTICS (table), 101
 userstat (variable), 97
 utility_user (variable), 72
 utility_user_password (variable), 72
 utility_user_privileges (variable), 73
 utility_user_schema_access (variable), 72

X

XtraDB, **178**
 XTRADB_INTERNAL_HASH_TABLES (table), 119
 XTRADB_READ_VIEW (table), 119
 XTRADB_RSEG (table), 124