Percona Server for MySQL Documentation

Release 5.6.51-93.0

Dec 01, 2022

CONTENTS

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

- 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 for MySQL* 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.

Part I

Introduction

CHAPTER

ONE

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 for MySQL* and *MariaDB*.

CHAPTER

TWO

LIST OF FEATURES AVAILABLE IN *PERCONA SERVER FOR MYSQL* RELEASES

Percona Server for MySQL 5.6	*Percona Server for MySQL* 5.7	*Percona Server for MySQL* 8.0
Improved Buffer Pool Scalability	Improved Buffer Pool Scalability	Improved Buffer Pool Scalability
Improved InnoDB I/O Scalability	Improved InnoDB I/O Scalability	Improved InnoDB I/O Scalability
Multiple Adaptive Hash Search Partitions	Multiple Adaptive Hash Search Partitions	Multiple Adaptive Hash Search Partitions
Atomic write support for Fusion-io devices	Atomic write support for Fusion-io devices	Atomic write support for Fusion-io devices
Query Cache Enhancements	Query Cache Enhancements	Feature not implemented
Improved NUMA support	Improved NUMA support	Feature not implemented
Thread Pool	Thread Pool	Thread Pool
Suppress Warning Messages	Suppress Warning Messages	Suppress Warning Messages
Ability to change database for mysqlbinlog	Ability to change database for mysqlbinlog	Ability to change database for mysqlbinlog
Fixed Size for the Read Ahead Area	Fixed Size for the Read Ahead Area	Fixed Size for the Read Ahead Area
Improved MEMORY Storage Engine	Improved MEMORY Storage Engine	Improved MEMORY Storage Engine
Restricting the number of binlog files	Restricting the number of binlog files	Restricting the number of binlog files
Ignoring missing tables in mysqldump	Ignoring missing tables in mysqldump	Ignoring missing tables in mysqldump
Too Many Connections Warning	Too Many Connections Warning	Too Many Connections Warning
Handle Corrupted Tables	Handle Corrupted Tables	Handle Corrupted Tables
Lock-Free SHOW SLAVE STATUS	Lock-Free SHOW SLAVE STATUS	Lock-Free SHOW SLAVE STATUS
Expanded Fast Index Creation	Expanded Fast Index Creation	Expanded Fast Index Creation
Percona Toolkit UDFs	Percona Toolkit UDFs	Percona Toolkit UDFs
Support for Fake Changes	Support for Fake Changes	Support for Fake Changes
Kill Idle Transactions	Kill Idle Transactions	Kill Idle Transactions
XtraDB changed page tracking	XtraDB changed page tracking	XtraDB changed page tracking
Enforcing Storage Engine	Enforcing Storage Engine	Replaced with upstream implementation
Utility user	Utility user	Feature not implemented
Extending the secure-file-priv server option	Extending the secure-file-priv server option	Extending the secure-file-priv server option
Expanded Program Option Modifiers	Expanded Program Option Modifiers	Feature not implemented
PAM Authentication Plugin	PAM Authentication Plugin	PAM Authentication Plugin

Continued on next page

Table 2.1 – continued from previous page

Percona Server for MySQL 5.6	*Percona Server for MySQL* 5.7	*Percona Server for MySQL* 8.0
Log Archiving for XtraDB	Log Archiving for XtraDB	Log Archiving for XtraDB
User Statistics	User Statistics	User Statistics
Slow Query Log	Slow Query Log	Slow Query Log
Count InnoDB Deadlocks	Count InnoDB Deadlocks	Count InnoDB Deadlocks
Log All Client Commands (syslog)	Log All Client Commands (syslog)	Log All Client Commands (syslog)
Response Time Distribution	Response Time Distribution	Feature not implemented
Show Storage Engines	Show Storage Engines	Show Storage Engines
Show Lock Names	Show Lock Names	Show Lock Names
Process List	Process List	Process List
Misc. INFORMATION_SCHEMA Tables	Misc. INFORMATION_SCHEMA Tables	Misc. INFORMATION_SCHEMA Tables
Extended Show Engine InnoDB Status	Extended Show Engine InnoDB Status	Extended Show Engine InnoDB Status
Thread Based Profiling	Thread Based Profiling	Thread Based Profiling
XtraDB Performance Improvements for I/O-Bound Highly-Concurrent Workloads	XtraDB Performance Improvements for I/O-Bound Highly-Concurrent Workloads	XtraDB Performance Improvements for I/O-Bound Highly-Concurrent Workloads
Page cleaner thread tuning	Page cleaner thread tuning	Page cleaner thread tuning
Statement Timeout	Statement Timeout	Statement Timeout
Extended SELECT INTO OUTFILE/DUMPFILE	Extended SELECT INTO OUTFILE/DUMPFILE	Extended SELECT INTO OUTFILE/DUMPFILE
Per-query variable statement	Per-query variable statement	Per-query variable statement
Extended mysqlbinlog	Extended mysqlbinlog	Extended mysqlbinlog
Slow Query Log Rotation and Expiration	Slow Query Log Rotation and Expiration	Slow Query Log Rotation and Expiration
Metrics for scalability measurement	Metrics for scalability measurement	Feature not implemented
Audit Log	Audit Log	Audit Log
Backup Locks	Backup Locks	Backup Locks
CSV engine mode for standard-compliant quote and comma parsing	CSV engine mode for standard-compliant quote and comma parsing	CSV engine mode for standard-compliant quote and comma parsing
Super read-only	Super read-only	Super read-only

2.1 Other Reading

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

CHAPTER

THREE

PERCONA SERVER FOR MYSQL FEATURE COMPARISON

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

- 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.

We provide these benefits by significantly enhancing *Percona Server for MySQL* 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[1]
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 `secure-file-priv` option	Yes	
Enforcing the Specific Storage Engine	Yes	

[1] Feature available in Enterprise version only

CHAPTER

FOUR

CHANGED IN PERCONA SERVER 5.6

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

4.1 Features removed from *Percona Server for MySQL* 5.6 that were available in *Percona Server for MySQL* 5.5

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

- `SHOW [GLOBAL] TEMPORARY TABLES` functionality is now only available via the `INFORMATION_SCHEMA` tables **:table:'TEMPORARY_TABLES'** and **:table:'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.

4.2 Replaced features that were present in *Percona Server for MySQL* 5.5

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

- `SHOW ENGINE INNODB STATUS` section "OLDEST VIEW" has been replaced by the **:table:'XTRADB_READ_VIEW'** INFORMATION_SCHEMA table.
- `SHOW ENGINE INNODB STATUS` sections on memory usage for InnoDB/XtraDB hash tables has been replaced by the **:table:'XTRADB_INTERNAL_HASH_TABLES'** INFORMATION_SCHEMA table.
- The **:table:'INNODB_RSEG'** table has been renamed to **:table:'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 for MySQL* versions).
- *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 **: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 **:table:'QUERY_RESPONSE_TIME'** table; - command `FLUSH QUERY_RESPONSE_TIME;` has been replaced with **:variable:'query_response_time_flush'** variable.

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

Some *Percona Server for MySQL* 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 for MySQL* 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 for MySQL* 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 for MySQL* 5.5.28-29.2 because the **:variable:'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 dumped and imported, since it will fail to start on data files created when **:variable:'innodb_fast_checksum'** was enabled. As an alternative you can use **innochecksum** from *MySQL* 5.7 as described in this blogpost.
- Handle BLOB End of Line feature has been replaced by *MySQL* 5.6 binary-mode configuration option.
- *Percona Server for MySQL* 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 for MySQL* `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.

4.4 Features ported from *Percona Server for MySQL* 5.5 to *Percona Server for MySQL* 5.6

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

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

4.5 List of status variables that are no longer available in *Percona Server for MySQL* 5.6

Following status variables available in *Percona Server for MySQL* 5.5 are no longer present in *Percona Server for MySQL* 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 **:table:'TEMPORARY_TABLES'** and **:table:'GLOBAL_TEMPORARY_TABLES'**
Flashcache_enabled	information if the Flashcache support has been enabled has not been ported to *Percona Server for MySQL* 5.6
Innodb_adaptive_hash_cells	this variable has not been ported to *Percona Server for MySQL* 5.6
Innodb_adaptive_hash_heap_buffers	this variable has not been ported to *Percona Server for MySQL* 5.6
Innodb_adaptive_hash_hash_searches	replaced by adaptive_hash_searches counter in INFORMATION_SCHEMA INNODB_METRICS table
Innodb_adaptive_hash_non_hash_searches	replaced by adaptive_hash_searches_btree counter in INFORMATION_SCHEMA INNODB_METRICS table
Innodb_checkpoint_target_age	replaced by MySQL 5.6 flushing implementation
Innodb_dict_tables	InnoDB Data Dictionary Size Limit feature has been replaced by the new MySQL 5.6 table_definition_cache implementation
Innodb_master_thread_1_second_loops	new *InnoDB* master thread behavior makes this variable redundant
Innodb_master_thread_10_second_loops	new *InnoDB* master thread behavior makes this variable redundant
Innodb_master_thread_background_loops	new *InnoDB* master thread behavior makes this variable redundant
Innodb_master_thread_main_flush_loops	new *InnoDB* 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	Binary Log Group Commit feature has been replaced with the *MySQL* 5.6 implementation that doesn't have this status variable.
binlog_group_commits	Binary Log Group Commit feature has been replaced with the *MySQL* 5.6 implementation that doesn't have this status variable.

4.6 List of system variables that are no longer available in *Percona Server for MySQL* 5.6

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

Warning: *Percona Server for MySQL* 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
:variable:'fast_index_creation'	replaced by using MySQL's ALGORITHM option
:variable:'have_flashcache'	Information if the server has been compiled with the Flashcache support has not been ported to *Percona Server for MySQL* 5.6
:variable:'have_response_time_distribution'	Response Time Distribution feature has been ported to *Percona Server for MySQL* 5.6 without this variable
:variable:'innodb_adaptive_flushing_method'	replaced by MySQL 5.6 flushing implementation
:variable:'innodb_blocking_buffer_pool_restore'	variable doesn't have direct replacement in *MySQL* 5.6. Feature will be implemented in a future *Percona Server for MySQL* 5.6 release
:variable:'innodb_buffer_pool_restore_at_startup'	replaced by innodb_buffer_pool_load_at_startup
:variable:'innodb_buffer_pool_shm_checksum'	variable has been deprecated and removed in *Percona Server for MySQL* 5.5
:variable:'innodb_buffer_pool_shm_key'	variable has been deprecated and removed in *Percona Server for MySQL* 5.5
:variable:'innodb_checkpoint_age_target'	replaced by MySQL 5.6 flushing implementation
:variable:'innodb_dict_size_limit'	replaced by *MySQL* 5.6 new table_definition_cache implementation
:variable:'innodb_doublewrite_file'	Configuration of the Doublewrite Buffer feature containing this variable has not been ported to *Percona Server for MySQL* 5.6
:variable:'innodb_fast_checksum'	replaced by innodb_checksum_algorithm
:variable:'innodb_flush_neighbor_pages'	replaced by innodb_flush_neighbors
:variable:'innodb_ibuf_accel_rate'	Configurable Insert Buffer feature containing this variable has not been ported to *Percona Server for MySQL* 5.6
:variable:'innodb_ibuf_active_contract'	Configurable Insert Buffer feature containing this variable has not been ported to *Percona Server for MySQL* 5.6
:variable:'innodb_ibuf_max_size'	Configurable Insert Buffer feature containing this variable has not been ported to *Percona Server for MySQL* 5.6
:variable:'innodb_import_table_from_xtrabackup'	replaced by MySQL transportable tablespaces
:variable:'innodb_lazy_drop_table'	variable has been deprecated and removed in *Percona Server for MySQL* 5.5
:variable:'innodb_merge_sort_block_size'	replaced by innodb_sort_buffer_size
:variable:'innodb_page_size'	replaced by innodb_page_size
:variable:'innodb_read_ahead'	replaced by MySQL Read-Ahead Algorithm implementation, innodb_random_read_ahead
:variable:'innodb_recovery_stats'	InnoDB Recovery Stats feature containing this variable has not been ported to *Percona Server for MySQL* 5.6
:variable:'innodb_recovery_update_relay_log'	replaced by relay-log-recovery
:variable:'innodb_stats_auto_update'	replaced by innodb_stats_auto_recalc

Continued on next page

Table 4.2 – continued from previous page

System Variables	Feature Comment
:variable:‘innodb_stats_update_need_lock‘	variable has not been ported to *Percona Server for MySQL* 5.6
:variable:‘innodb_thread_concurrency_timer_based‘	InnoDB timer-based Concurrency Throttling feature containing this variable has not been ported to *Percona Server for MySQL* 5.6
:variable:‘innodb_use_sys_stats_table‘	variable has been replaced by Persistent Optimizer Statistics implementation in *MySQL* 5.6
:variable:‘log_slow_admin_statements‘	the upstream variable has the same functionality
:variable:‘log_slow_slave_statements‘	the upstream variable has the same functionality
:variable:‘optimizer_fix‘	this variable has been deprecated and removed in *Percona Server for MySQL* 5.5
:variable:‘query_response_time_range_base‘	Response Time Distribution feature containing this variable has been ported to *Percona Server for MySQL* 5.6, but requires plugin installation in order to work. More information can be found in *Response Time Distribution* documentation.
:variable:‘query_response_time_stats‘	Response Time Distribution feature containing this variable has been ported to *Percona Server for MySQL* 5.6, but requires plugin installation in order to work. More information can be found in *Response Time Distribution* documentation.

Part II

Installation

CHAPTER

FIVE

INSTALLING *PERCONA SERVER FOR MYSQL* 5.6

This page provides the information on how to you can install *Percona Server for MySQL*. Following options are available:

- *Installing Percona Server for MySQL from Repositories* (recommended)
- Installing *Percona Server for MySQL* from Downloaded *rpm* or *apt* Packages
- *Installing Percona Server for MySQL from a Binary Tarball*
- *Installing Percona Server for MySQL from a Source Tarball*
- *Installing Percona Server for MySQL from the Git Source Tree*
- *Compiling Percona Server for MySQL from Source*
- *Running Percona Server in a Docker Container*

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

5.1 Installing *Percona Server for MySQL* from Repositories

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 for MySQL*, *Percona XtraBackup*, and *Percona Toolkit*. This makes it easy to install and update your software and its dependencies through your operating system's package manager. This is the recommended way of installing where possible.

Following guides describe the installation process for using the official Percona repositories for `.deb` and `.rpm` packages.

5.1.1 Installing *Percona Server for MySQL* on *Debian* and *Ubuntu*

Ready-to-use packages are available from the *Percona Server for MySQL* software repositories and the download page.

Specific information on the supported platforms, products, and versions is described in Percona Software and Platform Lifecycle.

What's in each DEB package?

The `percona-server-server-5.6` package contains the database server itself, the `mysqld` binary and associated files.

The `percona-server-common-5.6` package contains files common to the server and client.

The `percona-server-client-5.6` package contains the command line client.

The `percona-server-5.6-dbg` package contains debug symbols for the server.

The `percona-server-test-5.6` package contains the database test suite.

The `percona-server-source-5.6` package contains the server source.

The `libperconaserverclient18.1-dev` package contains header files needed to compile software to use the client library.

The `libperconaserverclient18.1` package contains the client shared library. The `18.1` 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 its source code modified.

Installing *Percona Server for MySQL* from Percona `apt` repository

1. Install `GnuPG`, the GNU Privacy Guard:

```
$ sudo apt-get install gnupg2
```

2. Fetch the repository packages from Percona web:

```
$ wget https://repo.percona.com/apt/percona-release_latest.$(lsb_release -sc)_all.
↪deb
```

3. Install the downloaded package with **dpkg**. To do that, run the following commands as root or with **sudo**:

```
$ sudo dpkg -i percona-release_latest.$(lsb_release -sc)_all.deb
```

   Once you install this package the Percona repositories should be added. You can check the repository setup in the `/etc/apt/sources.list.d/percona-release.list` file.

4. Remember to update the local cache:

```
$ sudo apt-get update
```

5. After that you can install the server package:

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

Percona `apt` Testing repository

Percona offers pre-release builds from the testing repository. To enable it add the just uncomment the testing repository lines in the Percona repository definition in your repository file (default `/etc/apt/sources.list.d/percona-release.list`). It should looks like this (in this example `VERSION` is the name of your distribution):

```
$ sudo percona-release enable original 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.

Installing *Percona Server for MySQL* using downloaded deb packages

Download the packages of the desired series for your architecture from the download page. The easiest way is to download bundle which contains all the packages. Following example will download *Percona Server for MySQL* 5.6.25-73.1 release packages for *Debian* 8.0:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.6/Percona-Server-5.
↪6.25-73.1/binary/debian/jessie/x86_64/Percona-Server-5.6.25-73.1-r07b797f-
↪jessie-x86_64-bundle.tar
```

You should then unpack the bundle to get the packages:

```
$ tar xvf Percona-Server-5.6.25-73.1-r07b797f-jessie-x86_64-bundle.tar
```

After you unpack the bundle you should see the following packages:

```
$ ls *.deb
libperconaserverclient18.1-dev_5.6.25-73.1-1.jessie_amd64.deb
libperconaserverclient18.1_5.6.25-73.1-1.jessie_amd64.deb
percona-server-5.6-dbg_5.6.25-73.1-1.jessie_amd64.deb
percona-server-client-5.6_5.6.25-73.1-1.jessie_amd64.deb
percona-server-client_5.6.25-73.1-1.jessie_amd64.deb
percona-server-common-5.6_5.6.25-73.1-1.jessie_amd64.deb
percona-server-server-5.6_5.6.25-73.1-1.jessie_amd64.deb
percona-server-server_5.6.25-73.1-1.jessie_amd64.deb
percona-server-source-5.6_5.6.25-73.1-1.jessie_amd64.deb
percona-server-test-5.6_5.6.25-73.1-1.jessie_amd64.deb
percona-server-tokudb-5.6_5.6.25-73.1-1.jessie_amd64.deb
```

Now you can install *Percona Server for MySQL* by running:

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

This will install all the packages from the bundle. Another option is to download/specify only the packages you need for running *Percona Server for MySQL* installation (`libperconaserverclient18.1_5.6.25-73.1-1.jessie_amd64.deb`, `percona-server-client-5.6_5.6.25-73.1-1.jessie_amd64.deb`, `percona-server-common-5.6_5.6.25-73.1-1.jessie_amd64.deb`, and `percona-server-server-5.6_5.6.25-73.1-1.jessie_amd64.deb`).

Note: When installing packages manually like this, you'll need to make sure to resolve all the dependencies and install missing packages yourself.

Automating the Install *Percona Server for MySQL* using a non-interactive script

You can install *Percona Server for MySQL* with a non-interactive script using the following options:

- `debconf` - The Debian package configuration system

- `DEBIAN_FRONTEND` - an interface variable for `debconf`
- debconf-set-selections - inserts values into the debconf database

Note: If needed, you can return the contents of the debconf database with the following statement:

The following example script installs the server and secures the installation.

```
#!/bin/bash

# variable for the root password
dbpass="root"

# Install the OS updates
apt-get update && apt-get upgrade -y

# Set the timezone to CST
echo "America/Chicago" > /etc/timezone

dpkg-reconfigure -f noninteractive tzdata

# Install needed packages
apt-get install gnupg2
apt-get install debconf-utils

# Install noninteractive
export DEBIAN_FRONTEND=noninteractive

# Fetch the Percona repository
wget https://repo.percona.com/apt/percona-release_latest.$(lsb_release -sc)_all.deb

# Install the downloaded package with dpkg.
dpkg -i percona-release_latest.$(lsb_release -sc)_all.deb

# Update the local cache
apt-get update

# Install essential packages
apt-get -y install zsh htop

# Install MySQL Server in a Non-Interactive mode. Default root password will be "root"
debconf-set-selections <<< "percona-server-server-5.6 percona-server-server/root_
↪password password root"
debconf-set-selections <<< "percona-server-server-5.6 percona-server-server/root_
↪password_again password root"

apt-get -y install percona-server-server-5.6

# SQL statements to secure the installation
mysql -uroot -p"$dbpass"<< EOF_MYSQL

UPDATE mysql.user SET Password = PASSWORD("$dbpass") WHERE USER='root';
DELETE FROM mysql.user WHERE User='';
DELETE FROM mysql.user WHERE User='root' AND Host NOT IN ('localhost', '127.0.0.1',
↪'::1');
DROP DATABASE IF EXISTS test;
DELETE FROM mysql.db WHERE Db='test' OR Db='test\\_%';
FLUSH PRIVILEGES;
```

```
EOF_MYSQL

service mysql stop
service mysql start
```

The following table lists the default locations for files:

Files	Location
mysqld server	`/usr/sbin`
Configuration	`/etc/mysql/my.cnf`
Data directory	`/var/lib/mysql`
Logs	`/var/log/mysql`

Running *Percona Server for MySQL*

Percona Server for MySQL stores the data files in `/var/lib/mysql/` by default. You can find the configuration file that is used to manage *Percona Server for MySQL* in `/etc/mysql/my.cnf`. *Debian* and *Ubuntu* installation automatically creates a special `debian-sys-maint` user which is used by the control scripts to control the *Percona Server for MySQL* `mysqld` and `mysqld_safe` services. Login details for that user can be found in `/etc/mysql/debian.cnf`.

1. Starting the service

   Percona Server for MySQL is started automatically after it gets installed unless it encounters errors during the installation process. You can also manually start it by running:

   ```
   $ sudo service mysql start
   ```

2. Confirming that service is running

   You can check the service status by running:

   ```
   $ service mysql status
   ```

3. Stopping the service

   You can stop the service by running:

   ```
   $ sudo service mysql stop
   ```

4. Restarting the service

   You can restart the service by running:

   ```
   $ sudo service mysql restart
   ```

Note: *Debian* 8.0 (jessie) and *Ubuntu* 15.04 (vivid) come with systemd as the default system and service manager so you can invoke all the above commands with `sytemctl` instead of `service`. Currently both are supported.

Uninstalling *Percona Server for MySQL*

To uninstall *Percona Server for MySQL* you'll need to remove all the installed packages. Removing packages with **`apt-get remove`** will leave the configuration and data files. Removing the packages with **`apt-get purge`** will

remove all the packages with configuration files and data files (all the databases). Depending on your needs you can choose which command better suits you.

1. Stop the *Percona Server for MySQL* service

```
$ sudo service mysql stop
```

2. Remove the packages

   (a) Remove the packages. This will leave the data files (databases, tables, logs, configuration, etc.) behind. In case you don't need them you'll need to remove them manually.

```
$ sudo apt-get remove percona-server*
```

   (a) Purge the packages. **NOTE**: This will remove all the packages and delete all the data files (databases, tables, logs, etc.)

```
$ sudo apt-get purge percona-server*
```

5.1.2 Installing *Percona Server for MySQL* on Red Hat Enterprise Linux and CentOS

Ready-to-use packages are available from the *Percona Server for MySQL* software repositories and the download page. 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.

Specific information on the supported platforms, products, and versions are described in Percona Software and Platform Lifecycle.

What's in each RPM package?

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

The `Percona-Server-server-56` package contains the server itself (the `mysqld` binary).

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

The `Percona-Server-client-56` package contains the command line client.

The `Percona-Server-devel-56` package contains the header files needed to compile software using the client library.

The `Percona-Server-shared-56` 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. Following libraries are included in this package: `libmysqlclient.so.12`, `libmysqlclient.so.14`, `libmysqlclient.so.15`, and `libmysqlclient.so.16`.

The `Percona-Server-test-56` package includes the test suite for *Percona Server for MySQL*.

Installing *Percona Server for MySQL* from Percona `yum` repository

1. Install the Percona repository

   You can install Percona yum repository by running the following command as a `root` user or with sudo:

```
$ yum install https://repo.percona.com/yum/percona-release-latest.noarch.rpm
```

Output example

```
Loaded plugins: fastestmirror, langpacks
Examining /var/tmp/yum-root-6bcbFR/percona-release-latest.noarch.rpm: percona-
↪release-1.0-15.noarch
Marking /var/tmp/yum-root-6bcbFR/percona-release-latest.noarch.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package percona-release.noarch 0:1.0-15 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package            Arch      Version   Repository                          Size
================================================================================
Installing:
 percona-release    noarch    1.0-15    /percona-release-latest.noarch      21 k

Transaction Summary
================================================================================
Install  1 Package

Total size: 21 k
Installed size: 21 k
```

To install *Percona Server for MySQL* with SELinux policies, you also need the **`Percona-Server-selinux-*.noarch.rpm`** package:

```
$ yum install http://repo.percona.com/centos/7/RPMS/x86_64/Percona-Server-selinux-
↪56-5.6.42-rel84.2.el7.noarch.rpm
```

2. Testing the repository

   Make sure packages are now available from the repository, by executing the following command:

```
yum list | grep percona
```

   You should see output similar to the following:

```
...
Percona-Server-56-debuginfo.x86_64          5.6.25-rel73.1.el6          @percona-
↪release-x86_64
Percona-Server-client-56.x86_64             5.6.25-rel73.1.el6          @percona-
↪release-x86_64
Percona-Server-devel-56.x86_64              5.6.25-rel73.1.el6          @percona-
↪release-x86_64
Percona-Server-server-56.x86_64             5.6.25-rel73.1.el6          @percona-
↪release-x86_64
Percona-Server-shared-56.x86_64             5.6.25-rel73.1.el6          @percona-
↪release-x86_64
Percona-Server-test-56.x86_64               5.6.25-rel73.1.el6          @percona-
↪release-x86_64
Percona-Server-shared-compat.x86_64         5.1.68-rel14.6.551.rhel6    percona-
↪release-x86_64
```

```
...
```

3. Install the packages

   You can now install *Percona Server for MySQL* by running:

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

Percona *yum* Testing repository

Percona offers pre-release builds from our testing repository. To subscribe to the testing repository, you'll need to enable the testing repository in `/etc/yum.repos.d/percona-release.repo`. To do so, set both `percona-testing-$basearch` and `percona-testing-noarch` to `enabled = 1` (Note that there are 3 sections in this file: release, testing and experimental - in this case it is the second section that requires updating). **NOTE:** You'll need to install the Percona repository first (ref above) if this hasn't been done already.

Installing *Percona Server for MySQL* using downloaded rpm packages

1. Download the packages of the desired series for your architecture from the download page. The easiest way is to download bundle which contains all the packages. Following example will download *Percona Server for MySQL* 5.6.25-73.1 release packages for *CentOS* 6:

```
wget https://www.percona.com/downloads/Percona-Server-5.6/Percona-Server-5.6.25-
↪73.1/binary/redhat/6/x86_64/Percona-Server-5.6.25-73.1-r07b797f-el6-x86_64-
↪bundle.tar
```

2. You should then unpack the bundle to get the packages:

```
tar xvf Percona-Server-5.6.25-73.1-r07b797f-el6-x86_64-bundle.tar
```

   After you unpack the bundle you should see the following packages:

```
$ ls *.rpm

Percona-Server-56-debuginfo-5.6.25-rel73.1.el6.x86_64.rpm
Percona-Server-client-56-5.6.25-rel73.1.el6.x86_64.rpm
Percona-Server-devel-56-5.6.25-rel73.1.el6.x86_64.rpm
Percona-Server-server-56-5.6.25-rel73.1.el6.x86_64.rpm
Percona-Server-shared-56-5.6.25-rel73.1.el6.x86_64.rpm
Percona-Server-test-56-5.6.25-rel73.1.el6.x86_64.rpm
```

3. Now you can install *Percona Server for MySQL* by running:

```
rpm -ivh Percona-Server-server-56-5.6.25-rel73.1.el6.x86_64.rpm \
Percona-Server-client-56-5.6.25-rel73.1.el6.x86_64.rpm \
Percona-Server-shared-56-5.6.25-rel73.1.el6.x86_64.rpm
```

This will install only packages required to run the *Percona Server for MySQL*. To install all the packages (for debugging, testing, etc.) you should run:

```
$ rpm -ivh *.rpm
```

Note: When installing packages manually like this, you'll need to make sure to resolve all the dependencies and install missing packages yourself.

Install an Older Version of *Percona Server for MySQL*

A DBA may require the same MySQL version for all installed database instances. The required version may not be the most recent one. The following methods may be used to install an required version:

- Download the specific version packages and install the packages manually
- Create a custom local repository mirror and upgrade from that mirror with yum
- Connect to a cloud instance with the required database and software
- Point the default package-management utility to a specific version

Add the Percona repository . To verify, run the following `yum` command:

```
$ yum -q list available --showduplicates Percona-Server-server-56.x86_64
```

Output example

The query result has been edited for clarity:

```
Available Packages
Percona-Server-server-56.x86_64   5.6.20-rel68.0.el7      percona-release-
↪x86_64
Percona-Server-server-56.x86_64   5.6.21-rel69.0.el7      percona-release-
↪x86_64
Percona-Server-server-56.x86_64   5.6.21-rel70.0.el7      percona-release-
↪x86_64
Percona-Server-server-56.x86_64   5.6.21-rel70.1.el7      percona-release-
↪x86_64
Percona-Server-server-56.x86_64   5.6.22-rel71.0.el7      percona-release-
↪x86_64
Percona-Server-server-56.x86_64   5.6.22-rel72.0.el7      percona-release-
↪x86_64
...
Percona-Server-server-56.x86_64   5.6.37-rel82.2.el7      percona-release-
↪x86_64
...
```

Use yum to install an older version. You must rearrange the package name to list the package, version, and CPU family. Remove the ".x86_64" suffix from the package name.

For example, with Percona-Server 5.6.37, the name for the server in the package repository is:

Repository Name	CPU Family
Percona-Server-server-56.x86_64	5.6.37-rel82.2.el7

For installation, convert the package name to the following:

```
Percona-Server-server-56-5.6.37-rel82.2.el7
```

The following code installs the 5.6.37 version:

```
$ yum -q install Percona-Server-server-56-5.6.37-rel82.2.el7
Percona-Server-client-56-5.6.37-rel82.el7 Percona-Server-shared-56-5.6.37-rel82.2.el7
```

Output example

The results are the following:

```
===============================================================================================================
 Package                                Arch                    Version                                    ␣
↪           Repository                             Size
===============================================================================================================
Installing:
 Percona-Server-client-56               x86_64                  5.6.37-rel82.2.el7                         ␣
↪           percona-release-x86_64                5.6 M
 Percona-Server-server-56               x86_64                  5.6.37-rel82.2.el7                         ␣
↪           percona-release-x86_64                 18 M
 Percona-Server-shared-56               x86_64                  5.6.37-rel82.2.el7                         ␣
↪           percona-release-x86_64                618 k
     replacing  mariadb-libs.x86_64 1:5.5.64-1.el7

Transaction Summary
===============================================================================================================
Install  3 Packages

Is this ok [y/d/N]:
```

After you have installed the version, if you must revert to an earlier version, you can use **Yum** to also do that:

```
$ yum -q downgrade Percona-Server-server-56.x86_64
Percona-Server-client-56.x86_64 Percona-Server-sharing-56.x86_64

===============================================================================================================
Package                                 Arch                    Version                                    ␣
↪           Repository                            size
===============================================================================================================
Downgrading:
 Percona-Server-client-56               x86_64                  5.6.36-rel82.1.el7                         ␣
↪           percona-release-x86_64                5.7 M
 Percona-Server-server-56               x86_64                  5.6.36-rel82.1.el7                         ␣
↪           percona-release-x86_64                 18 M

Transaction Summary
===============================================================================================================
Downgrade  2 Packages

Is this ok [y/d/N]:
```

You can also downgrade to specific older version. In this example, you are downgrading to the 5.6.30 version. Run the following command:

```
$ yum -q downgrade Percona-Server-server-56-5.6.30-rel76.3.el7
Percona-Server-client-56-5.6.30-rel76.3.el7 Percona-Server-sharing-56-5.6.30-rel76.3.
↪el7

===============================================================================================================
 Package                                Arch                    Version                                    ␣
↪           Repository                             Size
```

```
================================================================================
Downgrading:
 Percona-Server-client-56               x86_64               5.6.30-rel76.3.el7
↪           percona-release-x86_64               5.6 M
 Percona-Server-server-56               x86_64               5.6.30-rel76.3.el7
↪           percona-release-x86_64                18 M

Transaction Summary
================================================================================
Downgrade  2 Packages

Is this ok [y/d/N]:
```

You can also use the `Yum shell` to install versions. The advantage of this method is a version is removed and a different version is installed in a single YUM transaction instead of manually downloading and installing each package. The transaction also does not break any package dependencies when you uninstall one version and install another.

```
$ yum shell
Loaded plugins: fastestmirror, langpacks
> remove Percona-Server-shared-56 Percona-Server-client-56 Percona-Server-server-56
> install Percona-Server-server-56-5.6.30-rel76.3.el7 Percona-Server-client-56-5.6.30-
↪rel76.3.el7 Percona-Server-shared-56-5.6.30-rel76.3.el7
Loading mirror speeds from cached hostfile
 * base: mirrors.raystedman.org
 * epel: fedora-epel.mirror.lstn.net
 * extras: mirrors.raystedman.org
 * updates: mirrors.raystedman.org
> run
--> Running transaction check
---> Package Percona-Server-client-56.x86_64 0:5.6.30-rel76.3.el7 will be installed
---> Package Percona-Server-client-56.x86_64 0:5.6.37-rel82.2.el7 will be erased
---> Package Percona-Server-server-56.x86_64 0:5.6.30-rel76.3.el7 will be installed
---> Package Percona-Server-server-56.x86_64 0:5.6.37-rel82.2.el7 will be erased
---> Package Percona-Server-shared-56.x86_64 0:5.6.30-rel76.3.el7 will be installed
---> Package Percona-Server-shared-56.x86_64 0:5.6.37-rel82.2.el7 will be erased
--> Finished Dependency Resolution

================================================================================
 Package                          Arch                 Version
↪          Repository                          Size
================================================================================
Installing:
 Percona-Server-client-56               x86_64               5.6.30-rel76.3.el7
↪          percona-release-x86_64               5.6 M
 Percona-Server-server-56               x86_64               5.6.30-rel76.3.el7
↪          percona-release-x86_64                18 M
 Percona-Server-shared-56               x86_64               5.6.30-rel76.3.el7
↪          percona-release-x86_64               619 k
Removing:
 Percona-Server-client-56               x86_64               5.6.37-rel82.2.el7
↪          @percona-release-x86_64               33 M
 Percona-Server-server-56               x86_64               5.6.37-rel82.2.el7
↪          @percona-release-x86_64               88 M
 Percona-Server-shared-56               x86_64               5.6.37-rel82.2.el7
↪          @percona-release-x86_64              3.4 M

Transaction Summary
================================================================================
```

```
Install  3 Packages
Remove   3 Packages

Total download size: 24 M
Is this ok [y/d/N]:
```

When you run package update, you can prevent an inadvertant set of updates to the latest version. You must have `yum-plugin-versionlock` installed and you can lock the packages to a specific version.

```
$ yum versionlock Percona-Server-server-56 Percona-Server-client-56 Percona-Server-
↪shared-56
Loaded plugins: fastestmirror, langpacks, versionlock
Adding versionlock on: 0:Percona-Server-server-56-5.6.30-rel76.3.el7
Adding versionlock on: 0:Percona-Server-client-56-5.6.30-rel76.3.el7
Adding versionlock on: 0:Percona-Server-shared-56-5.6.30-rel76.3.el7
versionlock added: 3
$ yum update
Loaded plugins: fastestmirror, langpacks, versionlock
Loading mirror speeds from cached hostfile
 * base: mirrors.raystedman.org
 * epel: pubmirror1.math.uh.edu
 * extras: mirrors.raystedman.org
 * updates: mirrors.raystedman.org
Excluding 3 updates due to versionlock (use "yum versionlock status" to show them)
No packages marked for update
$ yum -q versionlock list
0:Percona-Server-server-56-5.6.30-rel76.3.el7.*
0:Percona-Server-client-56-5.6.30-rel76.3.el7.*
0:Percona-Server-shared-56-5.6.30-rel76.3.el7.*
```

To update the packages to a different version, run the following command to unlock the versions before you run update:

```
$ yum versionlock clear
```

Note: The command can clear all version locks or a specific version lock.

Files	Location
mysqld server	`/usr/bin`
Configuration	`/etc/my.cnf`
Data directory	`/var/lib/mysql`
Logs	`/var/log/mysqld.log`

You can use the following command to locate the Data directory:

```
grep datadir /etc/my.cnf

datadir=/var/lib/mysql
```

Running *Percona Server for MySQL*

Percona Server for MySQL stores the data files in `/var/lib/mysql/` by default. You can find the configuration file that is used to manage *Percona Server for MySQL* in `/etc/my.cnf`.

1. Starting the service

Percona Server for MySQL does not start automatically on *RHEL* and *CentOS* after the installation. You should start the server by running:

```
service mysql start
```

2. Confirming that service is running

   You can check the service status by running:

```
service mysql status
```

3. Stopping the service

   You can stop the service by running:

```
service mysql stop
```

4. Restarting the service

   You can restart the service by running:

```
service mysql restart
```

Note: *RHEL* 7 and *CentOS* 7 come with systemd as the default system and service manager so you can invoke all the above commands with `sytemctl` instead of `service`. Currently both are supported.

Uninstalling *Percona Server for MySQL*

To completely uninstall *Percona Server for MySQL* you'll need to remove all the installed packages and data files.

1. Stop the *Percona Server for MySQL* service

```
service mysql stop
```

2. Remove the packages

```
yum remove Percona-Server*
```

3. Remove the data and configuration files

```
rm -rf /var/lib/mysql
rm -f /etc/my.cnf
```

Warning: This will remove all the packages and delete all the data files (databases, tables, logs, etc.), you might want to take a backup before doing this in case you need the data.

5.2 Installing *Percona Server for MySQL* from a Binary Tarball

In *Percona Server for MySQL* **:rn:'5.6.24-72.2'** and newer, the single binary tarball was replaced with multiple tarballs depending on the *OpenSSL* library available in the distribution:

OpenSSL version	Linux Distributions
ssl101	For *CentOS* 6 and *CentOS* 7 (“libssl.so.10 => /usr/lib64/libssl.so.10 “);
ssl102	For *Debian* 9 and *Ubuntu* versions starting from 14.04 (`libssl.so.1.1 => /usr/lib/libssl.so.1.1`);
ssl1:111	For *CentOS* 8 and *RedHat* 8 (`libssl.so.1.1 => /usr/lib64/libssl.so.1.1.1b`);

Note: The list is a guide. To find which libssl.so files are available on your system you should run the following in Ubuntu or Debian:

```
$ locate libssl | grep "^/usr/lib/"
```

In CentOS, run the following command:

```
$ ldconfig -p | grep ssl
```

Download the appropriate binary tarball from the `Linux - Generic` section on the download page.

Fetch and extract the correct binary tarball. For example for *Debian Stretch*:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.6/Percona-Server-5.6.45-86.
↪1/binary/tarball/Percona-Server-5.6.45-rel86.1-Linux.x86_64.ssl102.tar.gz
```

5.3 Installing *Percona Server for MySQL* from a Source Tarball

Fetch and extract the source tarball. For example:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.6/Percona-Server-5.6.49-89.
↪0/binary/tarball/Percona-Server-5.6.49-rel89.0-Linux.x86_64.ssl102.tar.gz
$ tar xfz Percona-Server-5.6.49-rel89.0-Linux.x86_64.ssl102.tar.gz
```

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

5.4 Installing *Percona Server for MySQL* from the Git Source Tree

Percona uses the Github revision control system for development. To build the latest *Percona Server for MySQL* from the source tree you will need `git` installed on your system.

You can now fetch the latest *Percona Server for MySQL* 5.6 sources.

```
$ git clone https://github.com/percona/percona-server.git
$ cd percona-server
$ git checkout 5.6
$ git submodule init
$ git submodule update
```

If you are going to be making changes to *Percona Server for MySQL* 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 for MySQL from Source* below.

5.5 Compiling *Percona Server for MySQL* 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 -DWITH_EMBEDDED_SERVER=OFF
```

Now, compile using make

```
$ make
```

Install:

```
$ make install
```

Percona Server 5.6 will now be installed on your system.

5.6 Building *Percona Server for MySQL* 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 for MySQL from the Git Source Tree*).

Extract the source tarball:

```
$ tar xfz 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.

5.6.1 Running Percona Server in a Docker Container

Docker images of Percona Server are hosted publicly on Docker Hub at https://hub.docker.com/r/percona/percona-server/.

For more information about using Docker, see the Docker Docs.

Note: Make sure that you are using the latest version of Docker. The ones provided via `apt` and `yum` may be outdated and cause errors.

Note: By default, Docker will pull the image from Docker Hub if it is not available locally.

Using the Percona Server Images

The following procedure describes how to run and access Percona Server 5.6 using Docker.

Starting a Percona Server Instance in a Container

To start a container named `ps` running the latest version in the Percona Server 5.6 series, with the root password set to `root`:

```
[root@docker-host] $ docker run -d \
  --name ps \
  -e MYSQL_ROOT_PASSWORD=root \
  percona/percona-server:5.6
```

Note: `root` is not a secure password.

Note: The *docker stop* command sends a *TERM* signal. Docker waits 10 seconds and sends a *KILL* signal. Very large instances cannot dump the data from memory to disk in 10 seconds. If you plan to run a very large instance, add the following option to the *docker run* command.

–stop-timeout 600

Accessing the Percona Server Container

To access the shell in the container:

```
[root@docker-host] $ docker exec -it ps /bin/bash
```

From the shell, you can view the error log:

```
[mysql@ps] $ more /var/log/mysql/error.log
2017-08-29T04:20:22.190474Z 0 [Warning] 'NO_ZERO_DATE', 'NO_ZERO_IN_DATE' and 'ERROR_
↪FOR_DIVISION_BY_ZERO' sql modes should be used with strict mode. They will be␣
↪merged with strict mode in a future release.
2017-08-29T04:20:22.190520Z 0 [Warning] 'NO_AUTO_CREATE_USER' sql mode was not set.
...
```

You can also run the MySQL command-line client to access the database directly:

```
[mysql@ps] $ mysql -uroot -proot
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.7.19-17 Percona Server (GPL), Release '17', Revision 'e19a6b7b73f'

Copyright (c) 2009-2017 Percona LLC and/or its affiliates
Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other␣
↪names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Accessing Percona Server from Application in Another Container

The image exposes the standard MySQL port 3306, so container linking makes Percona Server instance available from other containers. To link a container running your application (in this case, from image named `app/image`) with the Percona Server container, run it with the following command:

```
[root@docker-host] $ docker run -d \
  --name app \
  --link ps \
  app/image:latest
```

This application container will be able to access the Percona Server container via port 3306.

Environment Variables

When running a Docker container with Percona Server, you can adjust the configuration of the instance by passing one or more environment variables with the `docker run` command.

Note: These variables will not have any effect if you start the container with a data directory that already contains a database: any pre-existing database will always remain untouched on container startup.

The variables are optional, except that you must specify at least one of the following:

- **:variable:‘MYSQL_ALLOW_EMPTY_PASSWORD‘**: least secure, use only for testing.
- **:variable:‘MYSQL_ROOT_PASSWORD‘**: more secure, but setting the password on the command line is not recommended for sensitive production setups.

- **:variable:‘MYSQL_RANDOM_ROOT_PASSWORD‘**: most secure, recommended for production.

Note: To further secure your instance, use the **:variable:‘MYSQL_ONETIME_PASSWORD‘** variable if you are running version 5.6 or later.

Storing Data

There are two ways to store data used by applications that run in Docker containers:

- Let Docker manage the storage of your data by writing the database files to disk on the host system using its own internal volume management.
- Create a data directory on the host system (outside the container on high performance storage) and mount it to a directory visible from inside the container. This places the database files in a known location on the host system, and makes it easy for tools and applications on the host system to access the files. The user should make sure that the directory exists, and that permissions and other security mechanisms on the host system are set up correctly.

For example, if you create a data directory on a suitable volume on your host system named `/local/datadir`, you run the container with the following command:

```
[root@docker-host] $ docker run -d \
  --name ps \
  -e MYSQL_ROOT_PASSWORD=root \
  -v /local/datadir:/var/lib/mysql \
  percona/percona-server:5.6
```

The `-v /local/datadir:/var/lib/mysql` option mounts the `/local/datadir` directory on the host to `/var/lib/mysql` in the container, which is the default data directory used by Percona Server.

Note: If you the Percona Server container instance with a data directory that already contains data (the `mysql` subdirectory where all our system tables are stored), the **:variable:‘MYSQL_ROOT_PASSWORD‘** variable should be omitted from the `docker run` command.

Note: If you have SELinux enabled, assign the relevant policy type to the new data directory, so that the container will be allowed to access it:

```
[root@docker-host] $ chcon -Rt svirt_sandbox_file_t /local/datadir
```

Port Forwarding

Docker allows mapping ports on the container to ports on the host system using the `-p` option. If you run the container with this option, you can connect to the database by connecting your client to a port on the host machine. This can greatly simplify consolidating many instances to a single host.

To map the standard MySQL port 3306 to port 6603 on the host:

```
[root@docker-host] $ docker run -d \
 --name ps \
 -e MYSQL_ROOT_PASSWORD=root \
```

```
 -p 6603:3306 \
 percona/percona-server:5.6
```

Passing Options to Percona Server

You can pass options to Percona Server when running the container by appending them to the `docker run` command. For example, to start run Percona Server with UTF-8 as the default setting for character set and collation for all databases:

```
[root@docker-host] $ docker run -d \
 --name ps \
 -e MYSQL_ROOT_PASSWORD=root \
 percona/percona-server:5.6 \
 --character-set-server=utf8 \
 --collation-server=utf8_general_ci
```

CHAPTER

SIX

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 for MySQL* 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.

Note: In place upgrade is not possible if Fast InnoDB Checksum feature has been used. If this feature was enabled, turning it off before the upgrade requires table(s) to be dumped and imported, since the server will fail to start on data files created when **:variable:'innodb_fast_checksums'** was enabled. As an alternative you can use **`innochecksum`** from *MySQL* 5.7 as described in this blogpost.

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*

6.1.1 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 for MySQL* 5.6.

6.1.2 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 fourth, `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 for MySQL* 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
```

6.2 Upgrading using Standalone Packages

6.2.1 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`

Open Percona Server 5.6 Downloads and select the following:

- Version number
- Software - the operating system or source code

You can download all packages together or download the packages separately.

For example, if you required all of the packages for 5.6.51-91.0 for *Ubuntu* 18.04, a way of doing this is:

```
$ wget -r -l 1 -nd -A deb -R "dev" \
https://downloads.percona.com/downloads/Percona-Server-5.6/Percona-Server-5.6.51-91.0/
↪binary/debian/bionic/x86_64/Percona-Server-5.6.51-91.0-rb59139e-bionic-x86_64-
↪bundle.tar
```

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.

6.2.2 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 for MySQL* 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*" \
https://downloads.percona.com/downloads/Percona-Server-5.6/Percona-Server-5.6.51-91.0/
↪binary/redhat/7/x86_64/Percona-Server-5.6.51-91.0-rb59139e-el7-x86_64-bundle.tar
```

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
```

Performing a Distribution Upgrade in-place on a System with Percona Packages Installed ——————————————————————————

The recommended process for performing a distribution upgrade on a system with the Percona packages installed is:

1. Record which Percona packages are installed
2. Backup the data and configurations
3. Uninstall the Percona packages without removing the configurations or

data

4. Perform the upgrade by following the distribution upgrade instructions
5. Reboot the system
6. Install the Percona packages intended for the new version of the

distribution

Part III

Scalability Improvements

CHAPTER

SEVEN

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.

7.1 Version Specific Information

- **:rn:‘5.6.13-60.6‘** - Feature ported from *Percona Server for MySQL* 5.5

7.2 Other Information

7.2.1 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
[...]
17 Mutex spin waits 0, rounds 452650, OS waits 22573
18 RW-shared spins 27550, OS waits 13682; RW-excl spins 0, OS waits 0
```

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.

CHAPTER

EIGHT

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.

8.1 Version Specific Information

- **:rn:‘5.6.11-60.3‘**
  - Feature ported from *Percona Server for MySQL* 5.5
- **:rn:‘5.6.14-62.0‘**
  - New **:variable:‘innodb_log_checksum_algorithm‘** variable introduced.

8.2 System Variables

This variable enables or disables the effect of the per-session value of the *innodb_flush_log_at_trx_commit* variable.

If the global variable *innodb_use_global_flush_log_at_trx_commit* is set to `1`, the session will always use the current global value of *innodb_flush_log_at_trx_commit*. This is the upstream compatible mode. If the user attempts to change the *innodb_flush_log_at_trx_commit* value for a session, the session value is ignored.

If the global variable *innodb_use_global_flush_log_at_trx_commit* is set to `0`, a user can modify the `innodb_flush_log_at_trx_commit` per-session using the following command:

```
SET SESSION innodb_flush_log_at_trx_commit=0
```

This modification only affects the transactions in that session. Other sessions, if they have not been individually modified, continue to use the global *innodb_use_flush_log_at_trx_commit* value.

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 **:variable:‘innodb_log_block_size‘** to be changed. **Note:** This feature implementation is considered Experimental quality.

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 and parallel doublewrite files, but does not use the `fsync()` system call to flush the data files, log files, and parallel doublewrite files. This option isn't suitable for *XFS* file system.
- `ALL_O_DIRECT`: use O_DIRECT to open data files, log files, and parallel doublewrite files and use `fsync()` to flush the data files but not the log files or parallel doublewrite files. This option is recommended when *InnoDB* log files are big (more than 8GB), otherwise, there may be performance degradation. **Note**: When using this option on *ext4* filesystem variable **: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.

This variable is used to specify how log checksums are generated and verified. Behavior of **:variable:‘innodb_log_checksum_algorithm‘** depending on its value is mostly identical to **:variable:‘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 for MySQL* 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 **:variable:‘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. To ensure that any log data written using the previous algorithm is fully overwritten before strictness becomes effective following migration procedure to a strict log block checksum should be used:
  - note the current LSN;
  - set the log block checksum algorithm to the non-strict version of the desired algorithm;
  - wait until the current LSN advances to at least the previous LSN + log capacity;
  - set the log block checksum algorithm to the strict version of the desired algorithm.

8.2.1 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
...
```

8.3 Other Reading

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

CHAPTER

NINE

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 equal-sized partitions to avoid such problems.

The number of adaptive hash partitions specified by the variable **:variable:'innodb_adaptive_hash_index_partitions'** are created, and each hash index is assigned to a particular partition based on `index_id`. Thus the effect from the AHI partitioning is greatest when the AHI accesses are uniformly spread over a large number of indexes (table primary and secondary keys). However, if there are certain few hot indexes, then their corresponding AHI partitions will be hot as well, while others might be completely unused.

9.1 Version Specific Information

- **:rn:'5.6.13-60.6'** - Feature ported from *Percona Server for MySQL* 5.5

9.1.1 System Variables

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.

9.1.2 Other reading

- Index lock and adaptive search
- *MySQL* bug #62018

CHAPTER

TEN

MULTIPLE USER LEVEL LOCKS PER CONNECTION

Percona Server for MySQL supports multiple user level locks per connection as of version **:rn:'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.

10.1 Version Specific Information

- **:rn:'5.6.19-67.0'** - Feature implemented in *Percona Server for MySQL* 5.6

10.2 Other Information

- Author / Origin: Kostja Osipov

10.3 Other Reading

- MySQL: multiple user level locks per connection

Part IV

Performance Improvements

CHAPTER

ELEVEN

MULTIPLE PAGE ASYNCHRONOUS I/O REQUESTS

The I/O unit size in *InnoDB* is only one page, even if the server doing read ahead. A 16KB I/O unit size is small for sequential reads, and less efficient than a larger I/O unit size. *InnoDB* uses Linux asynchronous I/O (`aio`) by default. By submitting multiple consecutive 16KB read requests at once, Linux internally can merge requests and reads can be done more efficiently. This feature can submit multiple page I/O requests and works in the background.

You can manage the feature with the linear read-ahead technique. The technique adds pages to the buffer pool based on the buffer pool pages being accessed sequentially. The configuration parameter, `innodb_read_ahead_threshold` controls this process.

On a HDD RAID 1+0 environment, more than 1000MB/s disk reads can be achieved by submitting 64 consecutive pages requests at once, while only 160MB/s disk reads is shown by submitting single page request.

11.1 Version Specific Information

- **:rn:‘5.6.38-83.0‘**- Feature ported from the *Facebook MySQL* patch.

11.2 Status Variables

This variable shows the total number of submitted buffered asynchronous I/O requests. The variable is updated after the submission request to the kernel and is a counter, which always increases.

The following is an example of a variable call:

```
mysql> SHOW GLOBAL STATUS like "innodb_buffered_aio_submitted";

innodb_buffered_aio_submitted 12439
```

11.3 Other Reading

- Optimizing full table scans in InnoDB
- Bug #68659 InnoDB Linux native aio should submit more i/o requests at once

CHAPTER

TWELVE

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.

12.1 Version Specific Information

- **:rn:'5.6.11-60.3'** `Atomic write support for Fusion-io` feature implemented. This feature was ported from *MariaDB*.

12.2 System Variables

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

12.3 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

CHAPTER

THIRTEEN

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

13.1 Diagnosing contention more easily

This features 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 appear 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 from t where id > 0;
4       root    localhost       test    Query   1       Waiting on query cache mutex         ␣
↪SELECT number from t where id > 0;
5       root    localhost       test    Query   0       NULL
```

13.2 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.

13.3 System Variables

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

13.3.1 Other Reading

- MySQL general thread states
- Query cache freezes

CHAPTER

FOURTEEN

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-robin fashion over the available node. This can be done by using the mysqld_safe **:variable:'numa_interleave'** option. **NOTE:** In order for this feature to work correctly `mysqld_safe` needs to be started as `root`.

Another improvement implemented is preallocating the pages in the buffer pool on startup with **:variable:'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.

14.1 Version Specific Information

- **:rn:'5.6.11-60.3'** Improved `NUMA` support implemented. This feature was ported from Twitter's *MySQL* patches.
- **:rn:'5.6.27-75.0'** Variables **:variable:'innodb_buffer_pool_populate'** and **:variable:'numa_interleave'** have been mapped to the upstream implementation of the new innodb_numa_interleave variable.
- **:rn:'5.6.40-83.2'** Variables **:variable:'innodb_buffer_pool_populate'** and **:variable:'numa_interleave'** were reverted to their original implementation due to upstream variant being less effective in memory allocation. Now buffer pool is allocated with MAP_POPULATE, forcing NUMA interleaved allocation at the buffer pool initialization time.

14.2 System Variables

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.

14.3 Command-line Options for mysqld_safe

When enabled (set to `1`) 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.

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

14.4 Other Reading

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

CHAPTER

FIFTEEN

HANDLERSOCKET

15.1 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 for MySQL* with *XtraDB*, see below.

15.2 Version Specific Information

- **:rn:'5.6.17-66.0'** Full functionality available.

Other Information

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

15.3 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';
```

15.4 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         ␣
↪                             | Info             |
+----+-------------+-----------------+---------------+---------+------+---------------
↪-----------------------------+------------------+
|  1 | system user | connecting host | NULL          | Connect | NULL |␣
↪handlersocket: mode=rd, 0 conns, 0 active | NULL             |
|  2 | system user | connecting host | NULL          | Connect | NULL |␣
↪handlersocket: mode=rd, 0 conns, 0 active | NULL             |
...
| 16 | system user | connecting host | NULL          | Connect | NULL |␣
↪handlersocket: mode=rd, 0 conns, 0 active | NULL             |
| 17 | system user | connecting host | handlersocket | Connect | NULL |␣
↪handlersocket: mode=wr, 0 conns, 0 active | NULL             |
```

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 |
+----+------------+
```

15.4.1 Configuration options

HandlerSocket has many configuration options that are detailed here.

15.5 Other Reading

- Yoshinori Matsunobu's blog post describing HandlerSocket
- Percona Server now both SQL and NOSQL

CHAPTER

SIXTEEN

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.

16.1 Version Specific Information

- **:rn:‘5.6.13-60.5‘** : *Percona Server for MySQL* 5.5 feature re-implemented

16.2 Other Information

- Bugs fixed: #606811

16.3 Other Reading

- BUF_READ_AHEAD_AREA Bottleneck

CHAPTER

SEVENTEEN

THREAD POOL

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

The *thread pool* feature introduces a dynamic *thread pool* that enables the server to keep the top performance even with a large number of client connections. With *thread pool* enabled, the server decreases the number of threads, which then reduces the context switching and hot locks contentions. Using the *thread pool* has the most effect on `OLTP` workloads (relatively short CPU-bound queries).

To enable the *thread pool* feature, the **:variable:'thread_handling'** variable should be set to the `pool-of-threads` value. This can be done by adding the following to the *MySQL* configuration file `my.cnf`:

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

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

Important: The current implementation of the *thread pool* feature is built into 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 differences, this implementation is not compatible with the upstream implementation.

17.1 Priority connection scheduling

In *Percona Server for MySQL* **:rn:'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 **:variable:'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 **:variable:‘thread_pool_high_prio_mode‘** variable has been introduced in *Percona Server for MySQL* **:rn:‘5.6.15-63.0‘**.

17.1.1 Low priority queue throttling

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

In this case, 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 **:variable:‘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 **:variable:‘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 transactions and release the lock.

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.

17.2 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 for MySQL* **:rn:‘5.6.15-63.0‘**.

17.3 Version Specific Information

- **:rn:‘5.6.10-60.2‘** `Thread Pool` feature implemented. This feature was ported from *MariaDB*.
- **:rn:‘5.6.11-60.3‘** Implemented priority connection scheduling and introduced new variable **:variable:‘thread_pool_high_prio_tickets‘** to the original implementation introduced in *MariaDB*.
- **:rn:‘5.6.15-63.0‘** Default value for **:variable:‘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.

- **:rn:'5.6.15-63.0'** Default value for **:variable:'thread_pool_high_prio_tickets'** was changed from `0` to `4294967295` to enable the priority connection scheduling by default.
- **:rn:'5.6.15-63.0'** Implemented new **:variable:'thread_pool_high_prio_mode'** variable.
- **:rn:'5.6.15-63.0'** Implemented *Low priority queue throttling*.
- **:rn:'5.6.15-63.0'** Ported *MariaDB* patch MDEV-156 to improve Thread Pool behavior when network wait times are not insignificant.

17.4 System Variables

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

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 **:variable:'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 **:variable:'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).

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` disables the high priority queue.

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.

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

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

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.

This variable can be used to specify an additional port that *Percona Server for MySQL* 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 the following command can be used:

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

This variable can be used to specify the maximum allowed number of connections plus one extra `SUPER` users connection on the **:variable:'extra_port'**. This can be used with the **:variable:'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.

17.5 Status Variables

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

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

Note: When *thread pool* is enabled, the value of the **:variable:'thread_cache_size'** variable is ignored. The **:variable:'Threads_cached'** status variable contains `0` in this case.

17.6 Other Reading

- Thread pool in MariaDB 5.5
- Thread pool implementation in Oracle MySQL

CHAPTER

EIGHTEEN

PAGE CLEANER THREAD TUNING

Percona Server for MySQL has implemented page cleaner thread improvements in **:rn:‘5.6.13-61.0‘** release.

18.1 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 for MySQL* 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 (**:variable:‘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.

18.2 Furious Flushing

In certain situations it makes sense for InnoDB to flush pages as fast as possible instead of abiding to **:variable:‘innodb_io_capacity‘** setting and, starting with 5.6, **:variable:‘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 **:variable:‘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 **:variable:‘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.

18.3 Timeouts

Percona Server for MySQL 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.

18.4 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 for MySQL* has implemented new LSN age factor formula for page cleaner adaptive flushing. that can be controlled with **:variable:‘innodb_cleaner_lsn_age_factor‘** variable.

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.

18.5 Tuning Variables

Percona Server for MySQL 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.

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

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

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

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

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.

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.

18.6 Other reading

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

CHAPTER

NINETEEN

XTRADB PERFORMANCE IMPROVEMENTS FOR I/O-BOUND HIGHLY-CONCURRENT WORKLOADS

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

19.1 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 became 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 for MySQL* has implemented priority refill for the buffer pool buffer pool free list in **:rn:‘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 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 **:variable:‘innodb_empty_free_list_algorithm‘**.

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

19.2 LRU manager thread

Percona Server for MySQL **:rn:‘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.

19.3 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 **:variable:‘innodb_foreground_preflush‘**.

19.4 Relative Thread Scheduling Priorities for XtraDB

Percona Server for MySQL has implemented Relative Thread Scheduling Priorities for *XtraDB* in **:rn:‘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 **:variable:‘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.

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 for MySQL has introduced several options, only available in builds compiled with `UNIV_PERF_DEBUG` C preprocessor define.

19.5 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 for MySQL* has introduced several tuning options, only available in builds compiled with `UNIV_PERF_DEBUG` C preprocessor define.

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

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

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

When buffer pool cleaner thread acquire shared resources with priority.

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

19.6 Other Reading

- *Page cleaner thread tuning*

CHAPTER

TWENTY

PREFIX INDEX QUERIES OPTIMIZATION

Percona Server for MySQL has ported Prefix Index Queries Optimization feature from Facebook patch for *MySQL*.

Prior to this *InnoDB* would always fetch the clustered index for all prefix columns in an index, even when the value of a particular record was smaller than the prefix length. This implementation optimizes that case to use the record from the secondary index and avoid the extra lookup.

20.1 Status Variables

This variable shows the number of times secondary index lookup triggered cluster lookup.

This variable shows the number of times prefix optimization avoided triggering cluster lookup.

Part V

Flexibility Improvements

CHAPTER

TWENTYONE

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. Please note that warning code needs to be a string:

```
mysql> SET GLOBAL log_warnings_suppress = '1592';
Query OK, 0 rows affected (0.00 sec)
```

21.1 Version Specific Information

- **:rn:‘5.6.11-60.3‘**: Variable **:variable:‘log_warnings_suppress‘** ported from *Percona Server for MySQL* 5.5.

21.2 System Variables

It is intended to provide a more general mechanism for disabling warnings than existed previously with variable **: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.

21.3 Related Reading

- MySQL bug 42851
- MySQL InnoDB replication
- InnoDB Startup Options and System Variables
- InnoDB Error Handling

CHAPTER

TWENTYTWO

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* (DFR) introduced by the mysql-heap-dynamic-rows patch.

DFR 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 DFR 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 DFR implementation has two caveats regarding to ordering and indexes.

22.1 Caveats

22.1.1 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.

22.1.2 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.
```

22.2 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 `DRF`.

This sets two restrictions to tables:

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

22.2.1 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.

22.2.2 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).

22.3 Limitations

MyISAM tables are still used for query optimizer internal temporary tables where the `MEMORY` tables could be used now instead: for temporary tables containing large `VARCHAR``s, ``BLOB`, and `TEXT` columns.

22.4 Setting Row Format

Taking the restrictions into account, the *Improved MEMORY Storage Engine* will choose `DRF` 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.

22.4.1 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;
```

22.4.2 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 then 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`.

22.5 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_length    Data_free       Auto_increment  Create_time     Update_
↪time     Check_time      Collation       Checksum        Create_options  Comment
t1    MEMORY  10      Dynamic 0       X       0       X       0       0       NULL    ␣
↪NULL    NULL    NULL    latin1_swedish_ci       NULL    row_format=DYNAMIC KEY_
↪BLOCK_SIZE=124
```

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_length    Data_free       Auto_increment  Create_time     Update_
↪time     Check_time      Collation       Checksum        Create_options  Comment
t1    MEMORY  10      Fixed   0       X       0       X       0       0       NULL
↪NULL    NULL    NULL    latin1_swedish_ci       NULL    row_format=DYNAMIC KEY_
↪BLOCK_SIZE=124
```

22.6 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*)`.

22.7 See Also

- Dynamic row format for MEMORY tables

CHAPTER

TWENTYTHREE

RESTRICTING THE NUMBER OF BINLOG FILES

Maximum number of binlog files can now be restricted in *Percona Server for MySQL* with **:variable:‘max_binlog_files‘**. When variable **: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 **:variable:‘max_binlog_size‘** variable to limit the disk usage of the binlog files. If **:variable:‘max_binlog_size‘** is set to 1G and **:variable:‘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 **:variable:‘max_binlog_size‘** * **:variable:‘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.

23.1 Version Specific Information

- 5.6.11-60.3: Variable **:variable:‘max_binlog_files‘** introduced.

23.2 System Variables

23.3 Example

Number of the binlog files before setting this variable

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

Variable **: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
```

CHAPTER

TWENTYFOUR

EXTENDED MYSQLDUMP

24.1 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.

24.2 Backup Locks support

In *Percona Server for MySQL* **:rn:'5.6.16-64.0'** `mysqldump` has 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.

More information can be found on the *Backup Locks* feature documentation.

24.3 Compressed Columns support

In *Percona Server for MySQL* **:rn:'5.6.33-79.0'** **mysqldump** has been extended to support *Compressed columns with dictionaries* feature. More information about the new options can be found on the *Compressed columns with dictionaries* feature page.

24.4 Taking backup by descending primary key order

In **:rn:'5.6.35-81.0'** new `--order-by-primary-desc` has been implemented. This feature tells `mysqldump` to take the backup by descending primary key order (`PRIMARY KEY DESC`) which can be useful if storage engine is using reverse order column for a primary key.

24.5 Version Specific Information

- **:rn:'5.6.5-60.0'** **mysqldump** option `--ignore-create-error` introduced
- **:rn:'5.6.16-64.0'** **mysqldump** has been extended with *Backup Locks* support options
- **:rn:'5.6.33-79.0'** **mysqldump** has been extended with *Compressed columns with dictionaries* support options
- **:rn:'5.6.35-81.0'** **mysqldump** option `--order-by-primary-desc` introduced

CHAPTER

TWENTYFIVE

EXTENDED `SELECT INTO OUTFILE/DUMPFILE`

Percona Server for MySQL 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.

25.1 Version Specific Information

- **:rn:‘5.6.13-61.0‘** - Feature Implemented

25.2 Other Reading

- *MySQL* bug: #44835

CHAPTER

TWENTYSIX

PER-QUERY VARIABLE STATEMENT

Percona Server for MySQL has implemented per-query variable statement support in **:rn:'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>;
```

26.1 Examples

If we want to increase the **:variable:'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;
```

26.2 Version Specific Information

- **:rn:'5.6.14-62.0'** Feature implemented

26.3 Other Reading

- WL#681: Per query variable settings

CHAPTER

TWENTYSEVEN

EXTENDED `MYSQLBINLOG`

Percona Server for MySQL has implemented compression support for **`mysqlbinlog`** in **:rn:‘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 for MySQL 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.

27.1 Version Specific Information

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

CHAPTER

TWENTYEIGHT

SLOW QUERY LOG ROTATION AND EXPIRATION

Note: This feature is currently considered BETA quality.

Percona has implemented two new variables, **:variable:‘max_slowlog_size‘** and **:variable:‘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 **:variable:‘max_binlog_files‘** variable used for controlling the binary log.

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

28.1 Version Specific Information

- **:rn:‘5.6.16-64.0‘**: Variable **:variable:‘max_slowlog_size‘** introduced.
- **:rn:‘5.6.16-64.0‘**: Variable **:variable:‘max_slowlog_files‘** introduced.

28.2 System Variables

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 **:variable:‘slow_query_log_file‘**.000001.

Maximum number of slow query log files. Used with **:variable:‘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.

CHAPTER

TWENTYNINE

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".

29.1 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, @@session.autocommit=1/*!*/;
```

```
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/*!*/;
```

29.2 Version Specific Information

- **:rn:‘5.6.16-64.0‘** Full functionality.

29.3 Client Command Line Parameter

rewrite-db

Cli Yes

Conf Yes

Scope Global

Dyn No

Vartype String

Default Off

29.4 Related Reading

- WL #36

CHAPTER

THIRTY

CSV ENGINE MODE FOR STANDARD-COMPLIANT QUOTE AND COMMA PARSING

MySQL CSV Storage Engine is non-standard with respect to embedded " and , character parsing. Fixing this issue unconditionally would break *MySQL* CSV format compatibility for any pre-existing user tables and for data exchange with other *MySQL* instances, but it would improve compatibility with other CSV producing/consuming tools.

To keep both *MySQL* and other tool compatibility, a new dynamic, global/session server variable **:variable:'csv_mode'** has been implemented. This variable allows an empty value (the default), and `IETF_QUOTES`.

If `IETF_QUOTES` is set, then embedded commas are accepted in quoted fields as-is, and a quote character is quoted by doubling it. In legacy mode embedded commas terminate the field, and quotes are quoted with a backslash.

30.1 Example

Table:

```
mysql> CREATE TABLE albums (
  `artist` text NOT NULL,
  `album` text NOT NULL
  ) ENGINE=CSV DEFAULT CHARSET=utf8
  ;
```

Following example shows the difference in parsing for default and `IETF_QUOTES` **:variable:'csv_quotes'**.

```
mysql> INSERT INTO albums VALUES ("Great Artist", "Old Album"),
 ("Great Artist", "Old Album \"Limited Edition\"");
```

If the variable **:variable:'csv_mode'** is set to empty value (default) parsed data will look like:

```
"Great Artist","Old Album"
"Great Artist","\"Limited Edition\",Old Album"
```

If the variable **:variable:'csv_mode'** is set to `IETF_QUOTES` parsed data will look like as described in CSV rules:

```
"Great Artist","Old Album"
"Great Artist","""Limited Edition"",Old Album"
```

Parsing the CSV file which has the proper quotes (shown in the previous example) can show different results:

With **:variable:'csv_mode'** set to empty value, parsed data will look like:

```
mysql> SELECT * FROM albums;
+--------------+--------------------+
| artist       | album              |
+--------------+--------------------+
| Great Artist | Old Album          |
| Great Artist | ""Limited Edition" |
+--------------+--------------------+
2 rows in set (0.02 sec)
```

With **:variable:‘csv_mode‘** set to `IETF_QUOTES` parsed data will look like:

```
mysql> SET csv_mode = 'IETF_QUOTES';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT * FROM albums;
+--------------+----------------------------+
| artist       | album                      |
+--------------+----------------------------+
| Great Artist | Old Album                  |
| Great Artist | "Limited Edition",Old Album |
+--------------+----------------------------+
```

30.2 Version Specific Information

- **:rn:‘5.6.21-70.0‘**: Variable **:variable:‘csv_mode‘** implemented

30.3 System Variables

Setting this variable is to `IETF_QUOTES` will enable the standard-compliant quote parsing: commas are accepted in quoted fields as-is, and quoting of `"` is changed from `\"` to `""`. If the variable is set to empty value (the default), then the old parsing behavior is kept.

30.4 Related Reading

- MySQL bug #71091

CHAPTER

THIRTYONE

ONLINE GTID DEPLOYMENT

Percona Server for MySQL now supports Online GTID deployment. This enables GTID to be deployed on existing replication setups without making the master read-only and stopping all the slaves. This feature was ported from the *Facebook* branch. Before this feature was implemented deploying the GTID replication on already existing replication setups required making a master **:variable:‘read_only‘**, shutting down all *MySQL* instances in the replica set simultaneously at the same position, enabling the **:variable:‘gtid_mode‘** variable in `my.cnf`, and then starting all of the instances. **NOTE:** This feature does not provide means to disable GTIDs online. Disabling GTIDs online is available in *Percona Server for MySQL* 5.7.

With **:variable:‘gtid_deployment_step‘** enabled, a host cannot generate GTID values on its own, but if GTID logged events are received through replication stream from master, they will be logged.

31.1 Performing the online GTID deployment

The online GTID deployment procedure can be done with the following steps:

1. replicas: **:variable:‘gtid_mode‘** = `ON` and **:variable:‘gtid_deployment_step‘** = `ON`

   master: **:variable:‘gtid_mode‘** = `OFF` and **:variable:‘gtid_deployment_step‘** = `OFF`

On each replica, one at a time, restart the *MySQL* server to enable **:variable:‘gtid_mode‘** and **:variable:‘gtid_deployment_step‘**. Afterward, we are in a state where every replica has **:variable:‘gtid_mode‘** set to `ON` and **:variable:‘gtid_deployment_step‘** set to `ON`, but the master still has **:variable:‘gtid_mode‘** set to `OFF` and **:variable:‘gtid_deployment_step‘** set to `OFF`. **NOTE:** in order to successfully restart the slaves **:variable:‘enforce-gtid-consistency‘** needs to be enabled as well.

2. new master: **:variable:‘gtid_mode‘** = `ON` and **:variable:‘gtid_deployment_step‘** = `OFF`

   rest of the replicas: **:variable:‘gtid_mode‘** = `ON` and **:variable:‘gtid_deployment_step‘** = `ON`

   old master: **:variable:‘gtid_mode‘** = `OFF` and **:variable:‘gtid_deployment_step‘** = `OFF`

Perform a master promotion as normal, i.e. set the **:variable:‘gtid_mode‘** to `ON` and **:variable:‘gtid_deployment_step‘** to `OFF`, re-pointing the replicas and original master to the new master. The original master's replication will intentionally break when started, since it still has the variable **:variable:‘gtid_mode‘** set to `OFF` and **:variable:‘gtid_deployment_step‘** to `OFF`.

3. new master: **:variable:‘gtid_mode‘** = `ON` and **:variable:‘gtid_deployment_step‘** = `OFF`

   rest of the replicas: **:variable:‘gtid_mode‘** = `ON`

   old master: **:variable:‘gtid_mode‘** = `ON`

Restart the original master to enable **:variable:‘gtid_mode‘**. It will now be able to replicate from the new master, and the entire replica set now has **:variable:‘gtid_mode‘** set to `ON`. You can now set the **:variable:‘gtid_deployment_step‘** to `OFF`.

31.2 Version Specific Information

- **:rn:‘5.6.22-72.0‘**: Feature ported from the *Facebook* branch

31.3 System Variables

When this variable is enabled, a host cannot generate GTIDs on its own, but if GTID logged events are received through replication stream from the master, they will be logged.

The conditions for dynamic switching the **:variable:‘gtid_deployment_step‘** off are the same as for read_only variable:

1. If you attempt to enable **:variable:‘gtid_deployment_step‘** while you have any explicit locks (acquired with `LOCK TABLES`) or have a pending transaction, an error occurs.
2. If you attempt to enable **:variable:‘gtid_deployment_step‘** while other clients hold explicit table locks or have pending transactions, the attempt blocks until the locks are released and the transactions end. While the attempt to enable **:variable:‘gtid_deployment_step‘** is pending, requests by other clients for table locks or to begin transactions also block until **:variable:‘gtid_deployment_step‘** has been set.

31.4 Related Reading

- Lessons from Deploying MySQL GTID at Scale
- *MySQL* GTID documentation

CHAPTER

THIRTYTWO

SUPPORT FOR PROXY PROTOCOL

The proxy protocol transports connection information in a safe way to an intermediate proxy server (for example, HAProxy) between a server and a client (i.e., mysql client, etc.). Since the proxy protocol is a way to spoof the client address, the proxy protocol is disabled by default. The protocal can be enabled on a per-host or a per-network basis for the trusted source addresses where trusted proxy servers are known to run.

Unproxied connections are not allowed from these source addresses.

Note: You need to ensure proper firewall Access Control List (ACL) is in place when this feature is enabled.

Proxying is supported for TCP over IPv4 and IPv6 connections only. UNIX socket connections can not be proxied and do not fall under the effect of proxy-protocol-networks='*'.

As a special exception, it is forbidden for the proxied IP address to be either `127.0.0.1` or `::1`.

32.1 Version Specific Information

- **:rn:'5.6.25-73.0'**: Initial implementation

32.2 System Variables

This variable is a global-only, read-only variable, which is either a `*` (to enable proxying globally, a non-recommended setting), or a list of comma-separated IPv4 and IPv6 network and host addresses, for which proxying is enabled. Network addresses are specified in CIDR notation, i.e. `192.168.0.0/24`. To prevent source host spoofing, the setting of this variable must be as restrictive as possible to include only trusted proxy hosts.

Note: If the *proxy_protocol_networks* is set to a value that is not `*`, you must add `bind_address` with the MySQL server IP in my.cnf.

If you set the proxy_protocol_networks to an IPv4-mapped address, the variable works without `bind_address`.

32.3 Related Reading

- PROXY protocol specification

CHAPTER

THIRTYTHREE

PER-SESSION SERVER-ID

Variable **:variable:‘server_id‘** is a global variable. In multi-master replication setups or for external replication, **:variable:‘server_id‘** can be useful as a session variable. In that case a session replaying a binary log from another server would set it to that server's id. That way binary log has the ultimate source server id attached to it no matter how many hosts it passes, and it would provide loop detection for multi-master replication.

This was implemented by introducing the new **:variable:‘pseudo_server_id‘** variable. This variable, when set to non-zero value, will cause all binary log events in that session to have that **:variable:‘server_id‘** value. A new variable was introduced instead of converting **:variable:‘server_id‘** to have both global and session scope in order to preserve compatibility.

You should use this option at your own risk because it is very easy to break replication when using **:variable:‘pseudo_server_id‘**. One special case is circular replication which definitely will be broken if you set **:variable:‘pseudo_server_id‘** to a value not assigned to any participating server (ie., setup is `1->2->3->4->1`, and **:variable:‘pseudo_server_id‘** is set to `5`). It is also possible to create a temporary table `foo`, then change **:variable:‘pseudo_server_id‘** and from now `foo` will not be visible by this session until **:variable:‘pseudo_server_id‘** gets restored.

33.1 Version Specific Information

- **:rn:‘5.6.26-74.0‘** Feature implemented and **:variable:‘pseudo_server_id‘** variable has been introduced

33.2 System Variables

When this variable is set to `0` (default), it will use the global **:variable:‘server_id‘** value. **Note:** this is different from the setting the global **:variable:‘server_id‘** to `0` which disables replication. Setting this variable to non-zero value will cause binary log events in that session to have it as **:variable:‘server_id‘** value. Setting this variable requires `SUPER` privileges.

33.3 Other Reading

- MDEV-500 - Session variable for server_id
- Upstream bug #35125 - allow the ability to set the server_id for a connection for logging to binary log

CHAPTER

THIRTYFOUR

SUPPORT FOR TLS V1.1 AND V1.2

Percona Server for MySQL has implemented TLS v1.1 and v1.2 protocol support and at the same time disabled TLS v1.0 support (support for TLS v1.0 can be enabled by adding the `TLSv1` to **:variable:'tls_version'** variable). Support for TLS v1.1 and v1.2 protocols has been implemented by porting the **:variable:'tls_version'** variable from 5.7 server. TLS v1.0 protocol has been disabled because it will no longer be viable for PCI after June 30th 2016. Variable default has been changed from `TLSv1,TLSv1.1,TLSv1.2` to `TLSv1.1,TLSv1.2` to disable the support for TLS v1.0 by default.

The client-side has the ability to make TLSv1.1 and 1.2 connections, but the option to allow only some protocol versions (`--tls-version`, `MYSQL_OPT_TLS_VERSION` in C API) has not been backported due to compatibility concerns and relatively easy option to use 5.7 clients instead if needed. **Note:** `MASTER_TLS_VERSION` clause of `CHANGE MASTER TO` statement has not been backported.

34.1 Version Specific Information

- **:rn:'5.6.31-77.0'**: Implemented support for TLS v1.1 and TLS v1.2 protocols

34.2 System Variables

This variable defines protocols permitted by the server for encrypted connections.

This server variable is set to `ON` if the server has been compiled with a SSL library providing TLSv1.2 support.

CHAPTER

THIRTYFIVE

COMPRESSED COLUMNS WITH DICTIONARIES

In **:rn:‘5.6.33-79.0‘** *Percona Server for MySQL* has been extended with a new per-column compression feature. It is a data type modifier, independent from user-level SQL and *InnoDB* data compression, that causes the data stored in the column to be compressed on writing to storage and decompressed on read. For all other purposes the data type is identical to the one without modifier, i.e. no new data types are created. Compression is done by using the `zlib` library.

Additionally it is possible to pre-define a set of strings for each compressed column to achieve a better compression ratio on a relatively small individual data items.

This feature provides:

- a better compression ratio for text data which consist of a large number of predefined words (e.g. JSON or XML) using compression methods with static dictionaries,
- in contrast to *InnoDB* row compression method it provides a way to select which columns in the table must be compressed.

This feature is based on a patch provided by Weixiang Zhai.

35.1 Specifications

The feature is limited to InnoDB/XtraDB storage engine and to:

- `BLOB` (including `TINYBLOB`, `MEDIUMBLOB`, `LONGBLOG`),
- `TEXT` (including `LONGTEXT`),
- `VARCHAR` (including `NATIONAL VARCHAR`), and
- `VARBINARY` columns.

The syntax to declare a compressed column is using an extension of an existing `COLUMN_FORMAT` modifier: `COLUMN_FORMAT COMPRESSED`. If this modifier is applied on an unsupported column type or storage engine, an error is returned.

The compression can be specified:

- at the table create time: `CREATE TABLE ... (..., foo BLOB COLUMN_FORMAT COMPRESSED, ...);`
- or a table can be modified to compress/decompress a column later: `ALTER TABLE ... MODIFY [COLUMN] ... COLUMN_FORMAT COMPRESSED`, or `ALTER TABLE ... CHANGE [COLUMN] ... COLUMN_FORMAT COMPRESSED`.

To decompress a column, a `COLUMN_FORMAT` value any other than `COMPRESSED` can be specified: `FIXED`, `DYNAMIC`, or `DEFAULT`. If there is a column compression/decompression request in an `ALTER TABLE`, it is forced to the `COPY` algorithm.

Two new variables: **:variable:'innodb_compressed_columns_zip_level'** and **:variable:'innodb_compressed_columns_threshold'** have been implemented.

35.2 Compression dictionary support

To achieve better compression ration on relatively small individual data items, it is possible to pre-define compression dictionary, which is a set of strings for each compressed column.

Compression dictionaries can be represented as a list of words in a form of a string (comma or any other character can be used as a delimiter although not required). In other words, `a,bb,ccc`, `a bb ccc` and `abbccc` will have the same effect. However, the latter is more space-efficient. Quote symbol quoting is handled by regular SQL quoting. Maximum supported dictionary length is 32506 bytes (`zlib` limitation).

The compression dictionary is stored in a new system *InnoDB* table. As this table is of the data dictionary kind, concurrent reads are allowed, but writes are serialized, and reads are blocked by writes. Table read through old read views are unsupported, similarly to *InnoDB* internal DDL transactions.

35.2.1 Example

In order to use the compression dictionary you'll need first to create it. This can be done by running:

```
mysql> SET @dictionary_data = 'one' 'two' 'three' 'four';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE COMPRESSION_DICTIONARY numbers (@dictionary_data);
Query OK, 0 rows affected (0.00 sec)
```

To create a table that has both compression and compressed dictionary support you should run:

```
mysql> CREATE TABLE t1(
        id INT,
        a BLOB COLUMN_FORMAT COMPRESSED,
        b BLOB COLUMN_FORMAT COMPRESSED WITH COMPRESSION_DICTIONARY numbers
      ) ENGINE=InnoDB;
```

Following example shows how to insert a sample of JSON data into the table:

```
SET @json_value =
 '[\n'
 ' {\n'
 ' "one" = 0,\n'
 ' "two" = 0,\n'
 ' "three" = 0,\n'
 ' "four" = 0\n'
 ' },\n'
 ' {\n'
 ' "one" = 0,\n'
 ' "two" = 0,\n'
 ' "three" = 0,\n'
 ' "four" = 0\n'
 ' },\n'
```

```
 ' {\n'
 ' "one" = 0,\n'
 ' "two" = 0,\n'
 ' "three" = 0,\n'
 ' "four" = 0\n'
 ' },\n'
 ' {\n'
 ' "one" = 0,\n'
 ' "two" = 0,\n'
 ' "three" = 0,\n'
 ' "four" = 0\n'
 ' }\n'
 ']\n'
;
```

```
mysql> INSERT INTO t1 VALUES(0, @json_value, @json_value);
Query OK, 1 row affected (0.01 sec)
```

35.3 INFORMATION_SCHEMA Tables

This feature implemented two new `INFORMATION_SCHEMA` tables.

This table provides a view over the internal compression dictionary table. `SUPER` privilege is required to query it.

This table provides a view over the internal table that stores the mapping between the compression dictionaries and the columns using them. `SUPER` privilege is require to query it.

35.4 Limitations

Compressed columns cannot be used in indices (neither on their own nor as parts of composite keys).

Note: `CREATE TABLE t2 AS SELECT * FROM t1` will create a new table with a compressed column, whereas `CREATE TABLE t2 AS SELECT CONCAT(a,'') AS a FROM t1` will not create compressed columns.

At the same time, after executing `CREATE TABLE t2 LIKE t1` statement, `t2.a` will have `COMPRESSED` attribute.

`ALTER TABLE ... DISCARD/IMPORT TABLESPACE` is not supported for tables with compressed columns. To export and import tablespaces with compressed columns, you need to uncompress them first with: `ALTER TABLE ... MODIFY ... COLUMN_FORMAT DEFAULT`.

35.5 mysqldump command line parameters

By default, with no additional options, `mysqldump` will generate a *MySQL* compatible SQL output.

All `/*!50633 COLUMN_FORMAT COMPRESSED */` and `/*!50633 COLUMN_FORMAT COMPRESSED WITH COMPRESSION_DICTIONARY <dictionary> */` won't be in the dump.

When a new option `enable-compressed-columns` is specified, all `/*!50633 COLUMN_FORMAT COMPRESSED */` will be left intact and all `/*!50633 COLUMN_FORMAT COMPRESSED WITH COMPRESSION_DICTIONARY <dictionary> */` will be transformed into `/*!50633 COLUMN_FORMAT COMPRESSED */`. In this mode the dump will contain the necessary SQL statements to create compressed columns, but without dictionaries.

When a new `enable-compressed-columns-with-dictionaries` option is specified, dump will contain all compressed column attributes and compression dictionary.

Moreover, the following dictionary creation fragments will be added before `CREATE TABLE` statements which are going to use these dictionaries for the first time.

```
/*!50633 DROP COMPRESSION_DICTIONARY IF EXISTS <dictionary>; */
/*!50633 CREATE COMPRESSION_DICTIONARY <dictionary>(...); */
```

Two new options `add-drop-compression-dictionary` and `skip-add-drop-compression-dictionary` will control if `/*!50633 DROP COMPRESSION_DICTIONARY IF EXISTS <dictionary> */` part from previous paragraph will be skipped or not. By default, `add-drop-compression-dictionary` mode will be used.

When both `enable-compressed-columns-with-dictionaries` and `--tab=<dir>` (separate file for each table) options are specified, necessary compression dictionaries will be created in each output file using the following fragment (regardless of the values of `add-drop-compression-dictionary` and `skip-add-drop-compression-dictionary` options).

```
/*!50633 CREATE COMPRESSION_DICTIONARY IF NOT EXISTS <dictionary>(...); */
```

35.6 Downgrade scenario

If it is necessary to perform *Percona Server for MySQL* downgrade from a version **:rn:'5.6.33-79.0'** (or newer) to a version older than **:rn:'5.6.33-79.0'** and if user databases have one or more table with compressed columns, there are two options to do this safely:

1. Use `mysqldump` in compatible mode (no compressed columns extensions must be specified).
2. Manually remove the `COMPRESSED` attribute from all columns which have it via `ALTER TABLE ... MODIFY ... COLUMN_FORMAT DEFAULT` before updating server binaries. In this case, the downgraded server can start safely with old data files.

35.7 Version Specific Information

- **:rn:'5.6.33-79.0'** Feature implemented in *Percona Server for MySQL* 5.6

35.8 System Variables

This variable is used to specify the compression level used for compressed columns. Specifying `0` will use no compression, `1` the fastest and `9` the best compression. Default value is `6`.

By default a value being inserted will be compressed if its length exceeds **:variable:'innodb_compressed_columns_threshold'** bytes. Otherwise, it will be stored in raw (uncompressed) form.

Please also notice that because of the nature of some data, its compressed representation can be longer than the original value. In this case it does not make sense to store such values in compressed form as *Percona Server for MySQL* would have to waste both memory space and CPU resources for unnecessary decompression. Therefore, even if the length of such non-compressible values exceeds **:variable:'innodb_compressed_columns_threshold'**, they will be stored in an uncompressed form (however, an attempt to compress them will still be made).

This parameter can be tuned in order to skip unnecessary attempts of data compression for values that are known in advance by the user to have bad compression ratio of their first N bytes.

35.9 Other reading

- How to find a good/optimal dictionary for zlib 'setDictionary' when processing a given set of data?

CHAPTER

THIRTYSIX

BINLOGGING AND REPLICATION IMPROVEMENTS

Due to continuous development, *Percona Server for MySQL* incorporated a number of improvements related to replication and binary logs handling. This resulted in replication specifics, which distinguishes it from *MySQL*.

36.1 Temporary tables and mixed logging format

36.1.1 Summary of the fix:

As soon as some statement involving a temporary table was met when using the MIXED binlog format, *MySQL* was switching to the row-based logging of all statements till the end of the session or until all temporary tables used in this session were dropped. It is inconvenient in the case of long lasting connections, including replication-related ones. *Percona Server for MySQL* fixes the situation by switching between statement-based and row-based logging as necessary.

36.1.2 Version Specific Information

- **:rn:'5.6.21-70.1'** Fix implemented in *Percona Server for MySQL* 5.6

36.1.3 Details:

The *mixed* binary logging format supported by *Percona Server for MySQL* means that server runs in statement-based logging by default, but switches to row-based logging when replication would be unpredictable - in the case of a nondeterministic SQL statement that may cause data divergence if reproduced on a slave server. The switch is done upon any condition from the long list, and one of these conditions is the use of temporary tables.

Temporary tables are **never** logged using row-based format, but any statement, that touches a temporary table, is logged in row mode. This way all the side effects that temporary tables may produce on non-temporary ones are intercepted.

There is no need to use row logging format for any other statements solely because of the temp table presence. However *MySQL* was undertaking such an excessive precaution: once some statement with temporary table had appeared and the row-based logging was used, *MySQL* logged unconditionally all subsequent statements in row format.

Percona Server have implemented more accurate behavior: instead of switching to row-based logging until the last temporary table is closed, the usual rules of row vs statement format apply, and presence of currently opened temporary tables is no longer considered. This change was introduced with the fix of a bug #151 (upstream #72475).

36.2 Temporary table drops and binloging on GTID-enabled server

36.2.1 Summary of the fix:

MySQL logs DROP statements for all temporary tables irrelative of the logging mode under which these tables were created. This produces binlog writes and errand GTIDs on slaves with row and mixed logging. *Percona Server for MySQL* fixes this by tracking the binlog format at temporary table create time and using it to decide whether a DROP should be logged or not.

36.2.2 Version Specific Information

- **:rn:'5.6.38-83.0'** Fix implemented in *Percona Server for MySQL* 5.6

36.2.3 Details:

Even with `read_only` mode enabled, the server permits some operations, including ones with temporary tables. With the previous fix, temporary table operations are not binlogged in row or mixed mode. But *MySQL* doesn't track what was the logging mode when temporary table was created, and therefore unconditionally logs `DROP` statements for all temporary tables. These `DROP` statements receive `IF EXISTS` addition, which is intended to make them harmless.

Percona Server for MySQL have fixed this with the bug fixes #964, upstream #83003, and upstream #85258. Moreover, after all the binlogging fixes discussed so far nothing involving temporary tables is logged to binary log in row or mixed format, and so there is no need to consider `CREATE/DROP TEMPORARY TABLE` unsafe for use in stored functions, triggers, and multi-statement transactions in row/mixed format. Therefore an additional fix was introduced to mark creation and drop of temporary tables as unsafe inside transactions in statement-based replication only (bug fixed #1816, upstream #89467)).

36.3 Safety of statements with a `LIMIT` clause

36.3.1 Summary of the fix:

MySQL considers all `UPDATE/DELETE/INSERT ... SELECT` statements with `LIMIT` clause to be unsafe, no matter wether they are really producing non-deterministic result or not, and switches from statement-based logging to row-based one. *Percona Server for MySQL* is more accurate, it acknowledges such instructions as safe when they include `ORDER BY PK` or `WHERE` condition. This fix has been ported from the upstream bug report #42415 (#44).

36.3.2 Version Specific Information

- **:rn:'5.6.13-60.5'** Fix ported from *Percona Server for MySQL* 5.5

36.4 Writing `FLUSH` Commands to the Binary Log

`FLUSH` commands, such as `FLUSH SLOW LOGS`, are not written to the binary log if the system variable **:variable:'binlog_skip_flush_commands'** is set to **ON**.

In addition, the following changes were implemented in the behavior of `read_only` and **|super-read-only|** modes:

- When `read_only` is set to **ON**, any `FLUSH ...` command executed by a normal user (without the `SUPER` privilege) are not written to the binary log regardless of the value of the **|bsfc|** variable.
- When **|super-read-only|** is set to **ON**, any `FLUSH ...` command executed by any user (even by those with the `SUPER` privilege) are not written to the binary log regardless of the value of the **|bsfc|** variable.

An attempt to run a `FLUSH` command without either `SUPER` or `RELOAD` privileges results in the `ER_SPECIFIC_ACCESS_DENIED_ERROR` exception regardless of the value of the **|bsfc|** variable.

When **|bsfc|** is set to **ON**, `FLUSH ...` commands are not written to the binary log. See *Writing FLUSH Commands to the Binary Log* for more information about what else affects the writing of `FLUSH` commands to the binary log.

Note: `FLUSH LOGS`, `FLUSH BINARY LOGS`, `FLUSH TABLES WITH READ LOCK`, and `FLUSH TABLES ... FOR EXPORT` are not written to the binary log no matter what value the **|bsfc|** variable contains. The `FLUSH` command is not recorded to the binary log and the value of **|bsfc|** is ignored if the `FLUSH` command is run with the `NO_WRITE_TO_BINLOG` keyword (or its alias `LOCAL`).

See also:

***MySQL* Documentation: FLUSH Syntax** https://dev.mysql.com/doc/refman/5.6/en/flush.html

---

Part VI

Reliability Improvements

CHAPTER

THIRTYSEVEN

TOO MANY CONNECTIONS WARNING

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

37.1 Version-Specific Information

- **:rn:'5.6.11-60.3'**: Feature ported from *Percona Server for MySQL* 5.5.

CHAPTER

THIRTYEIGHT

HANDLE CORRUPTED TABLES

When a server subsystem tries to access a corrupted table, the server may crash. If this outcome is not desirable when a corrupted table is encountered, set the new system **:variable:‘innodb_corrupt_table_action‘** variable to a value which allows the ongoing operation to continue without crashing the server.

The server error log registers attempts to access corrupted table pages.

Interacting with the :variable:‘innodb_force_recovery‘ variable

The **:variable:‘innodb_corrupt_table_action‘** variable may work in conjunction with the **:variable:‘innodb_force_recovery‘** variable which considerably reduces the effect of *InnoDB* subsystems running in the background.

If the **:variable:‘innodb_force_recovery‘** variable is set to a low value and you expect the server to crash, it may still be running due to a non-default value of the **:variable:‘innodb_corrupt_table_action‘** variable.

For more information about the **:variable:‘innodb_force_recovery‘** variable, see Forcing InnoDB Recovery from the MySQL Reference Manual.

This feature adds a new system variable.

38.1 Version Specific Information

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

38.2 System Variables

- With the default value, `assert`, *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".

CHAPTER

THIRTYNINE

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 features 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 for MySQL* originally used `SHOW SLAVE STATUS NOLOCK` syntax for this feature. As of **:rn:'5.6.20-68.0'** release, *Percona Server for MySQL* implements `SHOW SLAVE STATUS NONBLOCKING` syntax, which comes from early *MySQL* 5.7. Current *MySQL* 5.7 does not have this syntax and regular `SHOW SLAVE STATUS` is non-blocking.

39.1 Status Variables

The **:variable:'Com_show_slave_status_nolock'** statement counter variable indicates the number of times the statement `SHOW SLAVE STATUS NOLOCK` has been executed.

39.2 Version Specific Information

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

- **:rn:‘5.6.27-76.0‘**: `SHOW SLAVE STATUS NOLOCK` syntax in 5.6 has been undeprecated. Both `SHOW SLAVE STATUS NOLOCK` and `SHOW SLAVE STATUS NONBLOCKING` are now supported.

Part VII

Management Improvements

CHAPTER

FORTY

PERCONA TOOLKIT UDFS

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

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

40.1 Version Specific Information

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

40.2 Other Information

- Author / Origin: Baron Schwartz

40.3 Installation

These UDFs are part of the *Percona Server for MySQL* 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.

40.4 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 object file: No such file or directory)
```

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`.

40.5 Other Reading

- *Percona Toolkit* documentation

CHAPTER

FORTYONE

SUPPORT FOR FAKE CHANGES

Note: This feature implementation is considered *BETA* quality.

Unless multi-threaded slaves are used, replication is single threaded in nature, and it's important from the standpoint of performance to make sure that the queries executed by the replication thread or the events applied should be executed as fast as possible. A single event taking too long to apply is going to cause entire replication to stall, slowing down the rate at which replication catches up. This is especially painful when the slave server is restarted because with cold buffer pool individual events take far too long to complete. The slave is also generally I/O bound because of the difference of workload on master and the slave, and the biggest problem with single replication thread is that it has to read data to execute queries and most of the time is spent reading data then actually updating it.

41.1 Concept of Replication Prefetching

The process can be sped up by having prefetch threads to warm the server: replay statements and then rollback at commit. Prefetching works on a simple principle that if the data needed by the slave to apply events is already read then the application of events will be very fast as the data would already be cached. Replication is made up of two independent processes, an I/O thread that receives events from the master and writes to the relay log, and a SQL thread that reads the relay logs and applies those events. If the events in the relay log can be read in advance before the SQL thread reads them then the data that is needed by the SQL thread to apply the event would already be in the buffer pool and hence random disk I/O would be avoided, which would drastically improve the performance of SQL thread.

41.2 Prefetching with InnoDB Fake Changes

The way prefetching can be implemented without *Support for Fake Changes*, in most of the cases is by replaying the statements from the relay log but then manually converting all `COMMITs` to `ROLLBACKs` . This has the caveat of introducing the extra locking that is caused by the locks that are taken by the statements which are being replayed. The locks taken by statements executed by the process which is doing the prefetching can also cause lock contention with events that the SQL thread is trying to apply. Another issue with doing rollback is that, when a transaction changes data, old versions of the data are written to the undo log buffer. During the rollback phase *InnoDB* then has to read old versions of the data corresponding to what it was before the change from the undo log buffer and move it back to the *InnoDB* data page. In case of large transactions, or a transaction that updates a lot of data, the rollback can be costly and can generate significant amount of I/O.

Keeping in view the need of prefetching and the current caveats the **:variable:‘innodb_fake_changes‘** variable was implemented. The **:variable:‘innodb_fake_changes‘** variable enables an option for the server-side which allows for prefetching to work in a more performant manner. What enabling this option really does is that *InnoDB* reads the data needed by the `DML` queries but does not actually update the records, and hence no undo log record is generated, as nothing has changed, which means that rollback is instantaneous, and *InnoDB* doesn't have to do any additional

work on rollback. However, the problem of locking contention is not completely solved, when the records are read, SHARED locks are taken on the records, so this can still cause contention with data changes that SQL thread needs to make. *Percona Server for MySQL* does have a variable **:variable:'innodb_locking_fake_changes'** to make fake changes implementation completely lock-less. Because the fake changes implementation is not ready for lock-less operation for all workloads this variable is not safe to use and that is why it is disabled by default.

The **:variable:'innodb_fake_changes'** option, by enabling rollbacks on `COMMITs`, enables prefetching tools to use it. It's by no way a tool that does prefetching of data. It merely provides a feature that is needed by prefetching tools to work in a performant manner. There is no prefetching that is transparently done by the slave when **:variable:'innodb_fake_changes'** is enabled, i.e., there is no change in slave behavior, there is no separate thread that is started to prefetch events. A separate utility is needed that runs with the session **:variable:'innodb_fake_changes'** variable enabled and that reads events from the relay log.

41.3 Caveats

Warning: This feature is only safe to use with an InnoDB-only server, because it is implemented in *InnoDB* only. Using it with any other storage engine, such as *MyISAM* or *TokuDB*, will cause data inconsistencies because `COMMITs` will not be rolled back on those storage engines.

41.3.1 `DML` operations are supported

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

41.3.2 `DDL` operations are not supported

`DDL` operations **are not supported**, i.e. `ALTER TABLE` and `TRUNCATE TABLE`. Running the `DDL` operations with **:variable:'innodb_fake_changes'** enabled would return an error and the subsequent `DML` operations may fail (from missing column etc.).

41.3.3 Explicit `COMMIT` will lead to an error

There are two types of transactions, implicit and explicit. Implicit transactions are ones that are created automatically by *InnoDB* to wrap around statements that are executed with autocommit enabled. For example, an `UPDATE` query that is not enclosed by `START TRANSACTION` and `COMMIT`, when autocommit is enabled will be automatically treated as a single statement transaction. When *MySQL* writes events to the binary log, the events corresponding to the implicit transactions are automatically wrapped by `BEGIN` and `COMMIT`.

When a session has the **:variable:'innodb_fake_changes'** option enabled, all the `COMMITs` will lead to an error, and nothing will be committed, this is in line with the implementation of **:variable:'innodb_fake_changes'** option, which guarantees that data is not left in an inconsistent state. Hence the option **:variable:'innodb_fake_changes'** would not be needed to be enabled at the `GLOBAL` level, rather the option **:variable:'innodb_fake_changes'** will only be enabled at the `SESSION` level by the utility that you would use to read and replay the relay logs. Enabling **:variable:'innodb_fake_changes'** only for the session that is used by the utility will ensure that the utility can safely execute DML queries without the actual data getting modified.

41.4 How to use InnoDB Fake Changes

A separate tool would be needed to read the relay log and replay the queries, the only purpose of **:variable:'innodb_fake_changes'** is to prevent actual data modifications. There are two different tools developed by Facebook that rely on **:variable:'innodb_fake_changes'** and can be used for the purpose of slave prefetching:

- One tool is built using python and is named prefetch .
- Second tool is built in C and is named faker.

Both the tools rely on the *Percona Server for MySQL* **:variable:'innodb_fake_changes'** option.

Any other utility that can read the relay logs and replay them using multiple threads, would achieve what the above two tools achieve. Making sure that data is not modified by the tool would be done by enabling **:variable:'innodb_fake_changes'** option, but only on the `SESSION` level.

41.5 System Variables

41.6 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.

41.7 Related Reading

- on MySQL replication prefetching
- replication prefetching revisited

CHAPTER

FORTYTWO

KILL IDLE TRANSACTIONS

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

In *Percona Server for MySQL* **:rn:‘5.6.35-80.0‘** this feature has been re-implemented by setting a connection socket read timeout value instead of periodically scanning the internal *InnoDB* transaction list.

42.1 Version Specific Information

- **:rn:‘5.6.11-60.3‘:** Feature ported from *Percona Server for MySQL* 5.5
- **:rn:‘5.6.35-80.0‘:** Feature re-implemented using socket timeouts

42.2 System Variables

CHAPTER

FORTYTHREE

ENFORCING STORAGE ENGINE

Percona Server for MySQL 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 **:variable:'enforce_storage_engine'** with engine that isn't available, system will not start.

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

43.1 Version Specific Information

- **:rn:'5.6.11-60.3'** Variable **:variable:'enforce_storage_engine'** implemented.

43.2 System Variables

Note: This variable is not case sensitive.

43.3 Example

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

```
enforce_storage_engine=InnoDB
```

CHAPTER

FORTYFOUR

UTILITY USER

Percona Server for MySQL 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 increase the results in the *SHOW STATUS* for *Connections*, *Threads_connected* and *Threads_created*.
- Utility user will not appear in **:table:‘USER_STATISTICS‘**, **:table:‘CLIENT_STATISTICS‘** or **:table:‘THREAD_STATISTICS‘** tables or in any performance_schema 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 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 **:variable:‘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 **:variable:'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 **:variable:'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 **:variable:'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"

44.1 System Variables

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

Specifies the password required for the utility user.

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

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`

CHAPTER

FORTYFIVE

EXTENDING THE `secure-file-priv` SERVER OPTION

Percona Server for MySQL 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.

In *Percona Server for MySQL* **:rn:'5.6.34-79.1'** empty `secure-file-priv` became an alias for `NULL` value: both disable `LOAD_FILE()`, `LOAD DATA INFILE`, and `SELECT INTO OUTFILE`. With this change it is no longer possible to disable security checks by omitting the option as that would take the default value (`/var/lib/mysql-files/` for `.deb` and `.rpm` and `NULL` for `.tar.gz` packages. Instead, `--secure-file-priv=''` (or `=/`) should be used.

45.1 Version Specific Information

- **:rn:'5.6.11-60.3'** Variable **:variable:'secure-file-priv'** extended behavior implemented.
- **:rn:'5.6.34-79.1'** Default value for **:variable:'secure-file-priv'** has been changed from `NULL` to `/var/lib/mysql-files/` when installed from `.deb` and `.rpm` packages.

45.2 System Variables

CHAPTER

FORTYSIX

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 commad line or specifying maximum-query_cache_size=4M in the `my.cnf` will prevent any client from setting the **:variable:'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. This option modifier does not work for non-numeric system variables.
- 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. This option modifier does not work for non-numeric system variables.
- 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.

46.1 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>
```

46.2 Version Specific Information

- **:rn:'5.6.11-60.3'** Expanded program option modifiers implemented

46.3 Examples

Adding the following option to the `my.cnf` will set the minimum limit on **:variable:'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 **:variable:'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 **:variable:‘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 **:variable:‘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
```

CHAPTER

FORTYSEVEN

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 **: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 **:variable:'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-backuped LSN range.

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

47.1 User statements for handling the XtraDB changed page bitmaps

In *Percona Server for MySQL* **:rn:'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.

47.2 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.

47.3 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.

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 **:variable:'innodb_max_changed_pages'**.

47.4 System Variables

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

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

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

CHAPTER

FORTYEIGHT

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/libmysqlclient` 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 for MySQL*.

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.

48.1 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 ␣
↪   |
```

48.2 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
```

48.3 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.

48.4 Supplementary groups support

Percona Server for MySQL 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.

48.5 Known issues

Default mysql stack size is not enough to handle `pam_ecryptfs` module. Workaround is to increase the *MySQL* stack size by setting the thread-stack variable to at least `512KB` or by increasing the old value by `256KB`.

PAM authentication can fail with `mysqld: pam_unix(mysqld:account): Fork failed: Cannot allocate memory` error in the `/var/log/secure` even when there is enough memory available. Current workaround is to set vm.overcommit_memory to `1`:

```
echo 1 > /proc/sys/vm/overcommit_memory
```

and by adding the `vm.overcommit_memory = 1` to `/etc/sysctl.conf` to make the change permanent after reboot. Authentication of internal (i.e. non PAM) accounts continues to work fine when `mysqld` reaches this memory utilization level. *NOTE:* Setting the `vm.overcommit_memory` to `1` will cause kernel to perform no memory overcommit handling which could increase the potential for memory overload and invoking of OOM killer.

48.6 Version Specific Information

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

CHAPTER

FORTYNINE

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. 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.

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

49.1 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.

49.2 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.

49.3 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`.

49.4 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 **:variable:‘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 **:variable:‘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.

49.4.1 Version Specific Information

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

49.4.2 System Variables

49.4.3 Other Reading

- Improved InnoDB fast index creation
- Thinking about running OPTIMIZE on your InnoDB Table? Stop!

CHAPTER

FIFTY

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 **:variable:'innodb_log_arch_expire_sec'** variable.

This feature can be used to create incremental backups with *Percona XtraBackup* as described in this guide.

50.1 User statements for handling the XtraDB log archiving

New statements have been introduced in *Percona Server for MySQL* 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.

50.2 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.

50.3 Version Specific Information

- **:rn:‘5.6.11-60.3‘**: Feature implemented

50.4 System Variables

This variable is used to enable or disable log archiving.

This variable is used to specify the log archiving directory.

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

50.5 Session Variables

The **:variable:‘Com_purge_archived‘** statement counter variable indicates the number of times the statement `PURGE ARCHIVED LOGS TO` has been executed.

The **:variable:‘Com_purge_archived_before_date‘** statement counter variable indicates the number of times the statement `PURGE ARCHIVED LOGS BEFORE` has been executed.

CHAPTER

FIFTYONE

STATEMENT TIMEOUT

Note: This feature implementation is considered ALPHA quality.

Percona Server for MySQL has implemented a statement timeout feature. This feature can be used to limit the query execution time by specifying the timeout value in the **:variable:‘max_statement_time‘** variable. After the specified number of miliseconds 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
```

51.1 Version Specific Information

- **:rn:‘5.6.13-61.0‘**: Statement timeout implemented. This feature with some changes was ported from Twitter *MySQL* patches.

51.2 System Variables

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

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

51.3 Status Variables

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

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

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.

CHAPTER

FIFTYTWO

BACKUP LOCKS

Percona Server for MySQL has implemented this feature in **:rn:'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`.

52.1 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. If there is an active `LOCK TABLES FOR BACKUP` lock then all DDL statements and all updates to MyISAM, CSV, MEMORY, ARCHIVE, *TokuDB*, and *MyRocks* tables will be blocked in the `Waiting for backup lock` status, visible in `PERFORMANCE_SCHEMA` or `PROCESSLIST`.

`LOCK TABLES FOR BACKUP` has no effect on `SELECT` queries for all mentioned storage engines. Against *InnoDB*, Blackhole and Federated tables, the `LOCK TABLES FOR BACKUP` is not applicable to the `INSERT`, `REPLACE`, `UPDATE`, `DELETE` statements: Blackhole tables obviously have no relevance to 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.
```

52.2 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`. Starting with *Percona Server for MySQL* **:rn:'5.6.26-74.0'** `LOCK TABLES FOR BACKUP` flushes the current binary log coordinates to *InnoDB*. Thus, under active

LOCK TABLES FOR BACKUP, the binary log coordinates in *InnoDB* are consistent with its redo log and any non-transactional updates (as the latter are blocked by LOCK TABLES FOR BACKUP). It is planned that this change will enable *Percona XtraBackup* to avoid issuing the more invasive LOCK BINLOG FOR BACKUP command under some circumstances.

52.3 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
```

52.4 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.

52.5 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. **:variable:'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.

52.6 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 **: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 **:variable:'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 **:variable:'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 **:variable:'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 INSERTs 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.

52.7 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 has been removed in *Percona Server for MySQL* **:rn:'5.6.17-66.0'** by implementing *Start transaction with consistent snapshot* feature.

52.7.1 System Variables

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`.

This is a server variable implemented to help other utilities decide if `LOCK BINLOG FOR BACKUP` can be avoided in some cases. When the necessary server-side functionality is available, this server system variable exists and its value is always `YES`.

52.7.2 Status Variables

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

52.7.3 Client Command Line Parameter

`lock-for-backup`

Version 5.6.16-64.0 Implemented

Cli Yes

Scope Global

Dyn No

Vartype String

Default 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.

CHAPTER

FIFTYTHREE

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=mysql --log-error=/var/log/mysql/error.log --pid-file=/var/run/
↪mysqld/mysqld.pid --socket=/var/run/mysqld/mysqld.sock --port=3306"
 "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 are retrieved from the `com_status_vars` array in the `sql/mysqld.cc` file in a MySQL source distribution. Examples include `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` field for this event include `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"=""
 />
```

53.1 Installation

Audit Log plugin is shipped with *Percona Server for MySQL*, 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␣
↪    |
+-------------------------------+----------+--------------------+--------------+-----
↪----+
```

53.2 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 **:variable:‘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 UTC","command_class":"show_databases","connection_id":"37","status
↪":0,"sqltext":"show databases","user":"root[root] @ localhost []","host":"localhost
↪","os_user":"","ip":""}}
```

Example of the `CSV` format:

```
"Query","49284_2014-08-27T10:47:11","2014-08-27T10:47:23 UTC","show_databases","37",0,
↪"show databases","root[root] @ localhost []","localhost","",""
```

53.3 Streaming the audit log to syslog

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

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

53.4 Filtering by user

In **:rn:‘5.6.32-78.0‘** *Percona Server for MySQL* has implemented filtering by user. This was implemented by adding two new global variables: **:variable:‘audit_log_include_accounts‘** and **:variable:‘audit_log_exclude_accounts‘** to specify which user accounts should be included or excluded from audit logging.

Warning: Only one of these variables can contain a list of users to be either included or excluded, while the other needs to be `NULL`. If one of the variables is set to be not `NULL` (contains a list of users), the attempt to set another one will fail. Empty string means an empty list.

Note: Changes of **:variable:‘audit_log_include_accounts‘** and **:variable:‘audit_log_exclude_accounts‘** do not apply to existing server connections.

53.4.1 Example

Following example shows adding users who will be monitored:

```
mysql> SET GLOBAL audit_log_include_accounts = 'user1@localhost,root@localhost';
Query OK, 0 rows affected (0.00 sec)
```

If you you try to add users to both include and exclude lists server will show you the following error:

```
mysql> SET GLOBAL audit_log_exclude_accounts = 'user1@localhost,root@localhost';
ERROR 1231 (42000): Variable 'audit_log_exclude_accounts' can't be set to the value␣
↪of 'user1@localhost,root@localhost'
```

To switch from filtering by included user list to the excluded one or back, first set the currently active filtering variable to `NULL`:

```
mysql> SET GLOBAL audit_log_include_accounts = NULL;
Query OK, 0 rows affected (0.00 sec)

mysql> SET GLOBAL audit_log_exclude_accounts = 'user1@localhost,root@localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> SET GLOBAL audit_log_exclude_accounts = "'user'@'host'";
Query OK, 0 rows affected (0.00 sec)

mysql> SET GLOBAL audit_log_exclude_accounts = '''user''@''host''';
Query OK, 0 rows affected (0.00 sec)

mysql> SET GLOBAL audit_log_exclude_accounts = '\'user\'@\'host\'';
Query OK, 0 rows affected (0.00 sec)
```

To see what users are currently in the on the list you can run:

```
mysql> SELECT @@audit_log_exclude_accounts;
+-------------------------------+
| @@audit_log_exclude_accounts |
+-------------------------------+
| 'user'@'host'                 |
```

```
+-----------------------------+
1 row in set (0.00 sec)
```

Account names are the ones that are logged in the audit log. For example when you create a user:

```
mysql> CREATE USER 'user1'@'%' IDENTIFIED BY '111';
Query OK, 0 rows affected (0.00 sec)
```

This is what you'll see when `user1` connected from `localhost`:

```
<AUDIT_RECORD
  NAME="Connect"
  RECORD="21_2016-06-30T09:29:51"
  TIMESTAMP="2016-06-30T09:35:04 UTC"
  CONNECTION_ID="6"
  STATUS="0"
  USER="user1" ;; this is a 'user' part of account in 5.6
  PRIV_USER="user1"
  OS_LOGIN=""
  PROXY_USER=""
  HOST="localhost" ;; this is a 'host' part of account in 5.6
  IP=""
  DB=""
/>
```

To exclude `user1` from logging in *Percona Server for MySQL* 5.6 you must set:

```
SET GLOBAL audit_log_exclude_accounts = 'user1@localhost';
```

The value can be `NULL` or comma separated list of accounts in form `user@host` or `'user'@'host'` (if user or host contains comma).

53.5 Filtering by SQL command type

In **:rn:'5.6.32-78.0'** *Percona Server for MySQL* has implemented filtering by SQL command type. This was implemented by adding two new global variables: **:variable:'audit_log_include_commands'** and **:variable:'audit_log_exclude_commands'** to specify which command types should be included or excluded from audit logging.

Warning: Only one of these variables can contain a list of command types to be either included or excluded, while the other needs to be `NULL`. If one of the variables is set to be not `NULL` (contains a list of command types), the attempt to set another one will fail. Empty string means an empty list.

Note: If both **:variable:'audit_log_exclude_commands'** and **:variable:'audit_log_include_commands'** are `NULL` all commands will be logged.

53.5.1 Example

The available command types can be listed by running:

```
mysql> SELECT name FROM performance_schema.setup_instruments WHERE name LIKE
↪"statement/sql/%" ORDER BY name;
+---------------------------------------+
| name                                  |
+---------------------------------------+
| statement/sql/alter_db                |
| statement/sql/alter_db_upgrade        |
| statement/sql/alter_event             |
| statement/sql/alter_function          |
| statement/sql/alter_procedure         |
| statement/sql/alter_server            |
| statement/sql/alter_table             |
| statement/sql/alter_tablespace        |
| statement/sql/alter_user              |
| statement/sql/analyze                 |
| statement/sql/assign_to_keycache      |
| statement/sql/begin                   |
| statement/sql/binlog                  |
| statement/sql/call_procedure          |
| statement/sql/change_db               |
| statement/sql/change_master           |
...
| statement/sql/xa_rollback             |
| statement/sql/xa_start                |
+---------------------------------------+
145 rows in set (0.00 sec)
```

You can add commands to the include filter by running:

```
mysql> SET GLOBAL audit_log_include_commands= 'set_option,create_db';
```

If you now create a database:

```
mysql> CREATE DATABASE world;
```

You'll see it the audit log:

```
<AUDIT_RECORD
  NAME="Query"
  RECORD="10724_2016-08-18T12:34:22"
  TIMESTAMP="2016-08-18T15:10:47 UTC"
  COMMAND_CLASS="create_db"
  CONNECTION_ID="61"
  STATUS="0"
  SQLTEXT="create database world"
  USER="root[root] @ localhost []"
  HOST="localhost"
  OS_USER=""
  IP=""
  DB=""
/>
```

To switch command type filtering type from included type list to excluded one or back, first reset the currently-active list to `NULL`:

```
mysql> SET GLOBAL audit_log_include_commands = NULL;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SET GLOBAL audit_log_exclude_commands= 'set_option,create_db';
Query OK, 0 rows affected (0.00 sec)
```

Note: Invocation of stored procedures have command type `call_procedure`, and all the statements executed within the procedure have the same type `call_procedure` as well.

53.6 System Variables

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 **:variable:‘audit_log_handler‘** is set to `FILE`.

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.

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

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

This variable is used to specify the list of users for which *Filtering by user* is applied. The value can be `NULL` or comma separated list of accounts in form `user@host` or `'user'@'host'` (if user or host contains comma). If this variable is set, then **:variable:‘audit_log_include_accounts‘** must be unset, and vice versa.

This variable is used to specify the list of commands for which *Filtering by SQL command type* is applied. The value can be `NULL` or comma separated list of commands. If this variable is set, then **:variable:‘audit_log_include_commands‘** must be unset, and vice versa.

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.

This variable is used to specify the list of users for which *Filtering by user* is applied. The value can be `NULL` or comma separated list of accounts in form `user@host` or `'user'@'host'` (if user or host contains comma). If this variable is set, then **:variable:‘audit_log_exclude_accounts‘** must be unset, and vice versa.

This variable is used to specify the list of commands for which *Filtering by SQL command type* is applied. The value can be `NULL` or comma separated list of commands. If this variable is set, then **:variable:‘audit_log_exclude_commands‘** must be unset, and vice versa.

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

This variable specifies the maximum size of the audit log file. Upon reaching this size, the audit log will be rotated. The rotated log files are present in the same directory as the current log file. The sequence number is appended to the log file name upon rotation. For this variable to take effect, set the **:variable:‘audit_log_handler‘** variable to `FILE` and the **:variable:‘audit_log_rotations‘** variable to a value greater than zero.

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

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 **:variable:‘audit_log_file‘** variable. If it is set to `SYSLOG`, the audit log will be written to syslog.

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.

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.

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.

53.7 Status Variables

The number of times an audit log entry was either dropped or written directly to the file due to its size being bigger than **:variable:‘audit_log_buffer_size‘** variable.

53.8 Version Specific Information

- **:rn:‘5.6.17-65.0‘** Audit Log plugin has been implemented in *Percona Server for MySQL*.
- **:rn:‘5.6.20-68.0‘** *Percona Server for MySQL* *Audit Log Plugin* now supports `JSON` and `CSV` log formats.
- **:rn:‘5.6.20-68.0‘** *Percona Server for MySQL* *Audit Log Plugin* now supports *streaming the audit log to syslog*.
- **:rn:‘5.6.32-78.0‘** *Percona Server for MySQL* *Audit Log Plugin* now supports filtering by *user* and *sql_command*.
- **:rn:‘5.6.44-85.0‘** **:variable:‘Audit_log_buffer_size_overflow‘** variable implemented

CHAPTER

FIFTYFOUR

START TRANSACTION WITH CONSISTENT SNAPSHOT

Percona Server for MySQL, in **:rn:'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: **:variable:'Binlog_snapshot_file'** and **:variable:'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.

54.1 Snapshot Cloning

The *Percona Server for MySQL* 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.

54.2 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.

54.3 System Variables

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`.

54.4 Status Variables

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

54.5 Other Reading

- MariaDB Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT

CHAPTER

FIFTYFIVE

SUPER-READ-ONLY OPTION

Percona Server for MySQL has ported a new option, **:variable:‘super_read_only‘** from *WebScaleSQL*, which is like the read-only option, but affecting users with `SUPER` privileges as well. This means that server will not allow any updates even for the users that have `SUPER` privilege. If slave replication repositories are tables (when **:variable:‘master-info-repository‘** variable is set to `TABLE` and/or **:variable:‘relay-log-info-repository‘** variable is set to `TABLE`), any implicit updates to them (as performed by `CHANGE MASTER`, `START SLAVE`, or `STOP SLAVE`) are allowed regardless of **:variable:‘super_read_only‘** setting. Enabling **:variable:‘super_read_only‘** implies regular `read-only` as well.

Interaction with **:variable:‘read_only‘** variable:

- Turning **:variable:‘read_only‘** off also turns off **:variable:‘super_read_only‘**.
- Turning **:variable:‘super_read_only‘** on also turns **:variable:‘read_only‘** on.
- All other changes to either one of these have no affect on the other.

55.1 System Variables

When enabled the server will not allow any updates, beside updating the replication repositories if slave status logs are tables, even for the users that have `SUPER` privilege.

CHAPTER

FIFTYSIX

EXTENDED SHOW GRANTS

Prior to *Percona Server for MySQL* **:rn:'5.6.23-72.1'**, `SHOW GRANTS` displays only the privileges granted explicitly to the named account. Other privileges might be available to the account, but they are not displayed. For example, if an anonymous account exists, the named account might be able to use its privileges, but `SHOW GRANTS` will not display them. In **:rn:'5.6.23-72.1'** `SHOW GRANTS` command was extended to display all the effectively available privileges to the account.

56.1 Example

If we create the following users:

```
mysql> CREATE USER grantee@localhost IDENTIFIED BY 'grantee1';
Query OK, 0 rows affected (0.50 sec)

mysql> CREATE USER grantee IDENTIFIED BY 'grantee2';
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE DATABASE db2;
Query OK, 1 row affected (0.20 sec)

mysql> GRANT ALL PRIVILEGES ON db2.* TO grantee WITH GRANT OPTION;
Query OK, 0 rows affected (0.12 sec)
```

- `SHOW GRANTS` output before the change:

```
mysql> SHOW GRANTS;
+-----------------------------------------------------------------------------------------
↪----------------------------+
| Grants for grantee@localhost
↪                            |
+-----------------------------------------------------------------------------------------
↪----------------------------+
| GRANT USAGE ON *.* TO 'grantee'@'localhost' IDENTIFIED BY PASSWORD
↪'*9823FF338D44DAF02422CF24DD1F879FB4F6B232' |
+-----------------------------------------------------------------------------------------
↪----------------------------+
1 row in set (0.04 sec)
```

Although the grant for the `db2` database isn't shown, `grantee` user has enough privileges to create the table in that database:

```
user@trusty:~$ mysql -ugrantee -pgrantee1 -h localhost
```

```
mysql> CREATE TABLE db2.t1(a int);
Query OK, 0 rows affected (1.21 sec)
```

- `SHOW GRANTS` output after the change shows all the privileges for the `grantee` user:

```
mysql> SHOW GRANTS;
+-----------------------------------------------------------------------------------------------
↪----------------------------+
| Grants for grantee@localhost
↪                            |
+-----------------------------------------------------------------------------------------------
↪----------------------------+
| GRANT USAGE ON *.* TO 'grantee'@'localhost' IDENTIFIED BY PASSWORD
↪'*9823FF338D44DAF02422CF24DD1F879FB4F6B232' |
| GRANT ALL PRIVILEGES ON `db2`.* TO 'grantee'@'%' WITH GRANT OPTION
↪                            |
+-----------------------------------------------------------------------------------------------
↪----------------------------+
2 rows in set (0.00 sec)
```

56.1.1 Version-Specific Information

- **:rn:'5.6.23-72.1'**: Feature implemented

56.1.2 Other reading

- #53645 - `SHOW GRANTS` not displaying all the applicable grants

CHAPTER

FIFTYSEVEN

SSL IMPROVEMENTS

Percona Server for MySQL enables transmitting data in the encrypted form by using the TLSv1.2 protocol. By default, *Percona Server for MySQL* disables TLSv1.0 and deprecates TLSv1.1.

Implemented Using OpenSSL

Percona Server for MySQL binaries link with the OpenSSL library to implement the support of TLS protocols. *Percona Server for MySQL* supports OpenSSL version 1.1.

Percona Server for MySQL binaries do not link with the YASSL EMBEDDED SSL LIBRARY as *MySQL* Community Edition does. The yaSSL library only supports TSLv1.0 and TSL1.1 protocols which are less secure than TLSv1.2. You may still link with yaSSL when building *Percona Server for MySQL* from source.

As part of its implementation, *Percona Server for MySQL* offers correct diagnostic messages in cases like ciphers on the client and the server mismatching, the required TLS version not enabled, and so on. For more information, see #75311 Error for SSL cipher is unhelpful.

Important: As *Percona Server for MySQL* does not use yaSSL, yaSSL security advisories are not applicable to *Percona Server for MySQL*. System administrators should track the security advisories relevant to OpenSSL and upgrade their operating system promptly.

See also:

More information about YASSL EMBEDDED SSL LIBRARY https://www.wolfssl.com/products/yassl/

***MySQL* Documentation: OpenSSL Versus yaSSL** https://dev.mysql.com/doc/refman/5.6/en/openssl-versus-yassl.html

***MySQL* Bug System (solved for *Percona Server for MySQL*):** #75311 Error for SSL cipher is unhelpful

By default, *Percona Server for MySQL* passes elliptic curve crypto-based ciphers to OpenSSL, such as ECDHE-RSA-AES128-GCM-SHA256.

Note: Although documented as supported, elliptic curve crypto-based ciphers do not work with *MySQL*.

See also:

***MySQL* Bug System (solved for *Percona Server for MySQL*):** #82935 Cipher ECDHE-RSA-AES128-GCM-SHA256 listed in man/Ssl_cipher_list, not supported

---

57.1 Multi-Domain Certificates

Percona Server for MySQL supports multi-domain certificates (SAN (Subject Alternative Name)). This feature is useful to help manage the storage better, for building high availability clusters, or as part of a backup solution.

See also:

***Percona* Blog Post: When would you use SAN with MySQL?** https://www.percona.com/blog/2009/03/09/when-would-you-use-san-with-mysql/

***MySQL* Bug System (solved for *Percona Server for MySQL*):** #68052 SSL Certificate Subject ALT Names with IPs not respected with –ssl-verify-serve

57.2 Compatibility Matrix

Feature	YaSSL	OpenSSL < 1.0.2	OpenSSL >= 1.0.2
Validation of SSL certificate common name	Yes	Yes	Yes
Validation of SAN	No	Yes	Yes
Support for wildcard names	No	No	Yes

57.3 SSL Improvements in `mysqlbinlog`

Percona Server for MySQL extends **`mysqlbinlog`** to accept the `SSL` connection options as all the other client programs.

See also:

How *Percona Server for MySQL* extends the functionality of `mysqlbinlog` *Extended mysqlbinlog*

***MySQL* Bug System (solved for *Percona Server for MySQL*):** #41975 Support for SSL options not included in mysqlbinlog

Part VIII

Diagnostics Improvements

CHAPTER

FIFTYEIGHT

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.

58.1 Version Specific Information

- **:rn:‘5.6.11-60.3‘:** Feature ported from *Percona Server for MySQL* 5.5.

58.2 Other Information

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

58.3 System Variables

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.

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 **:variable:‘userstat‘** needs to be enabled as well in order for thread statistics to be collected.

58.4 INFORMATION_SCHEMA Tables

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
```

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.

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.

In order for this table to be populated with statistics, additional variable **:variable:'thread_statistics'** should be set to `ON`.

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
```

58.5 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.

58.6 Status Variables

The **:variable:'Com_show_client_statistics'** statement counter variable indicates the number of times the statement SHOW CLIENT_STATISTICS has been executed.

The **:variable:'Com_show_index_statistics'** statement counter variable indicates the number of times the statement SHOW INDEX_STATISTICS has been executed.

The **:variable:'Com_show_table_statistics'** statement counter variable indicates the number of times the statement SHOW TABLE_STATISTICS has been executed.

The **:variable:'Com_show_thread_statistics'** statement counter variable indicates the number of times the statement SHOW THREAD_STATISTICS has been executed.

The **:variable:'Com_show_user_statistics'** statement counter variable indicates the number of times the statement SHOW USER_STATISTICS has been executed.

CHAPTER

FIFTYNINE

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.

59.1 Version Specific Information

- **:rn:'5.6.11-60.3':**
  - Feature ported from *Percona Server for MySQL* 5.5.
- **:rn:'5.6.13-60.6':**
  - New **:variable:'slow_query_log_always_write_time'** variable introduced
- **:rn:'5.6.22-71.0':**
  - Implemented improved slow log reporting for queries in stored procedures.

59.2 Other Information

- Author / Origin: Maciej Dobrzanski

59.3 System Variables

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`.

Specifies semantic of **:variable:'log_slow_rate_limit'** - `session` or `query`.

Behavior of this variable depends from **:variable:'log_slow_rate_type'**.

Specifies that only a fraction of `session/query` should be logged. Logging is enabled for every nth `session/query`. By default, n is 1, so logging is enabled for every `session/query`. Please note: when **:variable:'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 **:variable:'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 **:variable:'log_slow_rate_limit'** in `session` mode, you should reconnect for get effect.
- When **:variable:'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) **:variable:'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/`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 for MySQL* **:rn:'5.6.13-60.6'** information about the **:variable:'log_slow_rate_limit'** has been added to the slow query log. This means that if the **:variable:'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 **:rn:'5.6.17-65.0'** implementation of the **:variable:'log_slow_rate_type'** set to `query` with **:variable:'log_slow_rate_limit'** feature would log every nth query deterministically. With the current implementation each query has a non-deterministic probability of 1/n to get logged.

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

Prior to :rn:'5.6.22-71.0' implementation of logging stored procedures was logging the stored procedure `CALL`s themselves alon

- Each query from a stored procedure is now logged to the slow query log individually
- `CALL` itself isn't logged to the slow query log anymore as this would be counting twice for the same query which would lead to incorrect results
- Queries that were called inside of stored procedures are annotated in the slow query log with the stored procedure name in which they run.

Example of the improved stored procedure slow query log entry:

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE improved_sp_log()
       BEGIN
        SELECT * FROM City;
        SELECT * FROM Country;
       END//
mysql> DELIMITER ;
mysql> CALL improved_sp_log();
```

When we check the slow query log after running the stored procedure ,with variable:*log_slow_sp_statements* set to `TRUE`, it should look like this:

```
# Time: 150109 11:38:55
# User@Host: root[root] @ localhost []
# Thread_id: 40  Schema: world  Last_errno: 0  Killed: 0
# Query_time: 0.012989  Lock_time: 0.000033  Rows_sent: 4079  Rows_examined: 4079 ␣
↪Rows_affected: 0  Rows_read: 4079
# Bytes_sent: 161085
# Stored routine: world.improved_sp_log
SET timestamp=1420803535;
SELECT * FROM City;
# User@Host: root[root] @ localhost []
# Thread_id: 40  Schema: world  Last_errno: 0  Killed: 0
# Query_time: 0.001413  Lock_time: 0.000017  Rows_sent: 4318  Rows_examined: 4318 ␣
↪Rows_affected: 0  Rows_read: 4318
# Bytes_sent: 194601
# Stored routine: world.improved_sp_log
SET timestamp=1420803535;
```

If variable **:variable:‘log_slow_sp_statements‘** is set to `FALSE`:

- Entry is added to a slow-log for a `CALL` statement only and not for any of the individual statements run in that stored procedure
- Execution time is reported for the `CALL` statement as the total execution time of the `CALL` including all its statements

If we run the same stored procedure with the variable **:variable:‘log_slow_sp_statements‘** is set to `FALSE` slow query log should look like this:

```
# Time: 150109 11:51:42
# User@Host: root[root] @ localhost []
# Thread_id: 40  Schema: world  Last_errno: 0  Killed: 0
# Query_time: 0.013947  Lock_time: 0.000000  Rows_sent: 4318  Rows_examined: 4318 ␣
↪Rows_affected: 0  Rows_read: 4318
# Bytes_sent: 194612
SET timestamp=1420804302;
CALL improved_sp_log();
```

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

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,query_plan`.
- `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`.

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

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 **:variable:'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]
```

Specifies which variables have global scope instead of local. For such variables, the global variable value is used in the current session, but without copying this value to the session value. Value is a "flag" variable - you can specify multiple values separated by commas

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

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, **:variable:'log_slow_rate_limit'** will not apply to it.

59.4 Other Information

59.4.1 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 (**:variable:'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';
```

59.4.2 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
```

59.4.3 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`

59.4.4 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

59.4.5 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 **:variable:‘log_slow_filter‘** option.

59.4.6 *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.

59.5 Related Reading

- Impact of logging on MySQL's performance
- log_slow_filter Usage
- Added microseconds to the slow query log event time

CHAPTER

SIXTY

EXTENDED SHOW ENGINE *INNODB* STATUS

This feature reorganizes the output of `SHOW ENGINE INNODB STATUS` for 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:

- Added two variables to control `SHOW ENGINE INNODB STATUS` information presented (bugfix for upstream bug #29126):
  - **:variable:‘innodb_show_verbose_locks‘** - Whether to show records locked
  - **:variable:‘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.

60.1 Version Specific Information

- **:rn:‘5.6.11-60.3‘**:

  Feature ported from *Percona Server for MySQL* 5.5.

60.2 Other Information

- Author / Origin: Baron Schwartz, http://lists.mysql.com/internals/35174

60.3 System Variables

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.

Specifies the number of locks held to print for each *InnoDB* transaction in `SHOW ENGINE INNODB STATUS`.

Makes *InnoDB* to write information about all lock wait timeout errors into the log file.

This allows to find out details about the failed transaction, and, most importantly, the blocking transaction. Query string can be obtained from **:table:‘performance_schema.events_statements_current‘** table, based on the `PROCESSLIST_ID` field, which corresponds to `thread_id` from the log output.

Taking into account that blocking transaction is often a multiple statement one, folowing query can be used to obtain blocking thread statements history:

```
SELECT s.SQL_TEXT FROM performance_schema.events_statements_history s
INNER JOIN performance_schema.threads t ON t.THREAD_ID = s.THREAD_ID
WHERE t.PROCESSLIST_ID = %d
UNION
SELECT s.SQL_TEXT FROM performance_schema.events_statements_current s
INNER JOIN performance_schema.threads t ON t.THREAD_ID = s.THREAD_ID
WHERE t.PROCESSLIST_ID = %d;
```

(PROCESSLIST_ID in this example is exactly the thread id from error log output).

60.4 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.

60.4.1 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
```

InnoDB has a master thread which performs background tasks depending on the server state, once per second. If the server is under workload, the master thread runs the following: performs background table drops; performs change buffer merge, adaptively; flushes the redo log to disk; evicts tables from the dictionary cache if needed to satisfy its size limit; makes a checkpoint. If the server is idle: performs background table drops, flushes and/or checkpoints the redo log if needed due to the checkpoint age; performs change buffer merge at full I/O capacity; evicts tables from the dictionary cache if needed; makes a checkpoint; and purges archived logs if needed.

This variable shows the number of times the above one-second loop was executed for active server states.

This variable shows the number of times the above one-second loop was executed for idle server states.

This variable shows the number of times the *InnoDB* master thread has written and flushed the redo log.

60.4.2 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
```

60.4.3 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
```

60.4.4 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
```

This variable shows the current log sequence number.

This variable shows the current maximum LSN that has been written and flushed to disk.

This variable shows the LSN of the latest completed checkpoint.

This variable shows the current *InnoDB* checkpoint age, i.e., the difference between the current LSN and the LSN of the last completed checkpoint.

This variable shows the maximum allowed checkppoint age above which the redo log is close to full and a checkpoint must happen before any further redo log writes.

60.4.5 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
```

This variable shows the current size, in bytes, of the adaptive hash index.

This variable shows the current size, in bytes, of the *InnoDB* in-memory data dictionary info.

This variable shows the total amount of memory, in bytes, *InnoDB* has allocated in the process heap memory.

This variable shows the total number of buffer pool pages which have been flushed from the LRU list, i.e., too old pages which had to be flushed in order to make buffer pool room to read in new data pages.

This variable shows the number of times a buffer pool page was not marked as accessed recently in the LRU list because of **:variable:‘innodb_old_blocks_time‘** variable setting.

This variable shows the number of times a buffer pool page was moved to the young end of the LRU list due to its access, to prevent its eviction from the buffer pool.

This variable shows the total number of buffer pool pages which are considered to be old according to the Making the Buffer Pool Scan Resistant manual page.

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.

This status variable shows the total amount of memory allocated for the *InnoDB* read views (in bytes).

60.4.6 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
```

This variable shows the length of the *InnoDB* history list.

This variable shows the next free transaction id number.

This variable shows the highest transaction id, above which the current oldest open read view does not see any transaction changes. Zero if there is no open view.

This variable shows the oldest transaction id whose records have not been purged yet.

This variable shows the number of undo log records for the oldest transactions that has not been purged yet.

60.5 INFORMATION_SCHEMA Tables

The following table contains information about the oldest active transaction in the system.

The following table contains information about the memory usage for InnoDB/XtraDB hash tables.

60.6 Other reading

- SHOW INNODB STATUS walk through
- Table locks in SHOW INNODB STATUS

CHAPTER

SIXTYONE

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 (**: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_deadlocks';
+----------------+
| 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.

61.1 Version Specific Information

- **:rn:‘5.6.11-60.3‘**: Full functionality available.

61.2 Status Variables

One new status variable was introduced by this feature.

61.3 Related Reading

- Original post by Eric Bergen

CHAPTER

SIXTYTWO

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.

62.1 Version Specific Information

- **:rn:'5.6.11-60.3'**: Feature ported from *Percona Server for MySQL* 5.5.

62.2 Other Information

- Author / Origin: Percona

62.3 Client Variables

The variable enables (ON)/disables (OFF) logging to syslog.

62.4 Other Reading

- http://en.wikipedia.org/wiki/Syslog
- http://tools.ietf.org/html/rfc5424

CHAPTER

SIXTYTHREE

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                                                        ␣
↪  | Transactions | XA   | Savepoints |
+------------+---------+----------------------------------------------------------------
↪--+--------------+------+------------+
| InnoDB     | YES     | Supports transactions, row-level locking, and foreign keys  ␣
↪  | YES          | YES  | YES        |
...
+------------+---------+----------------------------------------------------------------
↪--+--------------+------+------------+
```

After the Change:

```
mysql> show storage engines;
+------------+---------+----------------------------------------------------------------
↪--------------+--------------+------+------------+
| Engine     | Support | Comment                                                        ␣
↪              | Transactions |   XA | Savepoints |
+------------+---------+----------------------------------------------------------------
↪--------------+--------------+------+------------+
| InnoDB     | YES     | Percona-XtraDB, Supports transactions, row-level locking,␣
↪and foreign keys |          YES | YES  | YES        |
...
+------------+---------+----------------------------------------------------------------
↪--------------+--------------+------+------------+
```

63.1 Version-Specific Information

- **:rn:‘5.6.11-60.3‘**: Full functionality available.

CHAPTER

SIXTYFOUR

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  |
.....
```

CHAPTER

SIXTYFIVE

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:

- **:table:'PROCESSLIST'** table:
  - added column `TIME_MS`

65.1 Version Specific Information

- **:rn:'5.6.5-60.0'**:
  - Added column `TIME_MS` to table `PROCESSLIST`.
- **:rn:'5.6.11-60.3'**:
  - Added `ROWS_SENT` and `ROWS_EXAMINED` columns to table `PROCESSLIST`.'
- **:rn:'5.6.27-75.0'**
  - Added `TID` column to table `PROCESSLIST`.

65.2 INFORMATION_SCHEMA Tables

65.3 Example Output

Table **:table:'PROCESSLIST'**:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST;

+----+------+-----------+--------------------+---------+------+-----------+-----------
↪----------------+---------+-----------+---------------+
| ID | USER | HOST      | DB                 | COMMAND | TIME | STATE     | INFO      ␣
↪                | TIME_MS | ROWS_SENT | ROWS_EXAMINED |
+----+------+-----------+--------------------+---------+------+-----------+-----------
↪----------------+---------+-----------+---------------+
| 12 | root | localhost | information_schema | Query   |    0 | executing | select *␣
↪from processlist |       0 |         0 |             0 |
+----+------+-----------+--------------------+---------+------+-----------+-----------
↪----------------+---------+-----------+---------------+
```

CHAPTER

SIXTYSIX

MISC. INFORMATION_SCHEMA TABLES

This page lists the `INFORMATION_SCHEMA` tables added to standard *MySQL* by *Percona Server for MySQL* that don't exist elsewhere in the documentation.

66.1 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.

This table holds information on the temporary tables existing for all connections. You don't need the `SUPER` privilege to query this table.

This table holds information on the temporary tables existing for the running connection.

66.2 Multiple Rollback Segments

Percona Server for MySQL, in addition to the upstream multiple rollback segment implementation, provides the additional Information Schema table: `INFORMATION_SCHEMA.XTRADB_RSEG`.

66.3 `INFORMATION_SCHEMA` Tables

This feature provides the following table:

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)
```

CHAPTER

SIXTYSEVEN

THREAD BASED PROFILING

Percona Server for MySQL 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.

67.1 Version Specific Information

- **:rn:'5.6.11-60.3'**: Thread based profiling ported from *Percona Server for MySQL* 5.5.

CHAPTER

SIXTYEIGHT

METRICS FOR SCALABILITY MEASUREMENT

Note: This feature has been deprecated in *Percona Server for MySQL* **:rn:‘5.6.34-79.1‘**. Users who have installed this plugin but are not using its capability are advised to uninstall the plugin due to known crashing bugs.

Percona Server for MySQL has implemented extra scalability metrics in **:rn:‘5.6.17-65.0‘**. These metrics allow using Little’s Law, queueing theory, and Universal Scalability Law to gain insights into server performance. This feature is implemented as a plugin.

68.1 Installation

Scalability Metrics plugin is shipped with *Percona Server for MySQL*, 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     |
+-------------------------------+----------+--------------------+--------------------
↪----+---------+
```

68.2 System Variables

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.

68.3 Status Variables

This status variable shows total time elapsed, in microseconds, since metrics collection was started.

This status variable shows number of completed queries since metrics collection was started.

This status variable shows number of queries currently executed.

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).

This counter accounts the non-idle server time, that is, time when at least one query was executing.

68.4 Version Specific Information

- **:rn:‘5.6.17-65.0‘** Scalability Metrics plugin has been implemented in *Percona Server for MySQL*.
- **:rn:‘5.6.34-79.1‘** Feature has been deprecated.

68.5 Other Reading

- Fundamental performance and scalability instrumentation
- Forecasting MySQL Scalability with the Universal Scalability Law Whitepaper

CHAPTER

SIXTYNINE

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 query execution time begins after the initial locks are acquired. 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; positive infinity]
```

or

```
(0; 0.000001], (0.000001; 0.000010], (0.000010; 0.000100], ..., (0.100000; 1.0]; (1.0;
↪ 10.0]...(1000000; positive infinity]
```

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; positive infinity]
```

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 = 0.000094 seconds

* 4301 queries with 0.000010 < query execution time < = 0.000100 seconds; total␣
↪execution time of the 4301 queries = 0.236555 seconds

* 1499 queries with 0.000100 < query execution time < = 0.001000 seconds; total␣
↪execution time of the 1499 queries = 0.824450 seconds

* 14851 queries with 0.001000 < query execution time < = 0.010000 seconds; total␣
↪execution time of the 14851 queries = 81.680502 seconds

* 8066 queries with 0.010000 < query execution time < = 0.100000 seconds; total␣
↪execution time of the 8066 queries = 443.635693 seconds

* 1 query with 10.000000 < query execution time < = 100.0000 seconds; total execution␣
↪time of the 1 query = 55.937094 seconds
```

69.1 Logging the queries in separate `READ` and `WRITE` tables

Percona Server for MySQL **:rn:'5.6.22-72.0'** is now able to log the queries response times into separate `READ` and `WRITE` `INFORMATION_SCHEMA` tables. The two new tables are named **:table:'QUERY_RESPONSE_TIME_READ'** and **:table:'QUERY_RESPONSE_TIME_WRITE'** respectively. The decision on whether a query is a `read` or a `write` is based on the type of the command. Thus, for example, an `UPDATE ... WHERE <condition>` is always logged as a `write` query even if `<condition>` is always false and thus no actual writes happen during its execution.

Following SQL commands will be considered as `WRITE` queries and will be logged into the **:table:'QUERY_RESPONSE_TIME_WRITE'** table: `CREATE_TABLE`, `CREATE_INDEX`, `ALTER_TABLE`, `TRUNCATE`, `DROP_TABLE`, `LOAD`, `CREATE_DB`, `DROP_DB`, `ALTER_DB`, `RENAME_TABLE`, `DROP_INDEX`, `CREATE_VIEW`, `DROP_VIEW`, `CREATE_TRIGGER`, `DROP_TRIGGER`, `CREATE_EVENT`, `ALTER_EVENT`, `DROP_EVENT`, `UPDATE`, `UPDATE_MULTI`, `INSERT`, `INSERT_SELECT`, `DELETE`, `DELETE_MULTI`, `REPLACE`, `REPLACE_SELECT`, `CREATE_USER`, `RENAME_USER`, `DROP_USER`, `ALTER_USER`, `GRANT`, `REVOKE`, `REVOKE_ALL`, `OPTIMIZE`, `CREATE_FUNCTION`, `CREATE_PROCEDURE`, `CREATE_SPFUNCTION`,

DROP_PROCEDURE, DROP_FUNCTION, ALTER_PROCEDURE, ALTER_FUNCTION, INSTALL_PLUGIN, and UNINSTALL_PLUGIN. Commands not listed here are considered as READ queries and will be logged into the **:table:'QUERY_RESPONSE_TIME_READ'** table.

69.2 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 (**:table:'QUERY_RESPONSE_TIME'**) to output gathered statistics.

```
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME_READ SONAME 'query_response_time.so';
```

This plugin provides the interface (**:table:'QUERY_RESPONSE_TIME_READ'**) to output gathered statistics. **NOTE:** Available in *Percona Server for MySQL* **:rn:'5.6.22-72.0'** or later.

```
mysql> INSTALL PLUGIN QUERY_RESPONSE_TIME_WRITE SONAME 'query_response_time.so';
```

This plugin provides the interface (**:table:'QUERY_RESPONSE_TIME_WRITE'**) to output gathered statistics. **NOTE:** Available in *Percona Server for MySQL* **:rn:'5.6.22-72.0'** or later.

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     |
| QUERY_RESPONSE_TIME_READ    | ACTIVE   | INFORMATION SCHEMA | query_response_time.
↪so | GPL     |
| QUERY_RESPONSE_TIME_WRITE   | ACTIVE   | INFORMATION SCHEMA | query_response_time.
↪so | GPL     |
+-----------------------------+----------+--------------------+----------------------
↪-+---------+
```

69.3 Usage

To start collecting query time metrics, **:variable:'query_response_time_stats'** should be enabled:

```
SET GLOBAL query_response_time_stats = on;
```

And to make it persistent, add the same to `my.cnf`:

```
[mysqld]
query_response_time_stats = on
```

69.3.1 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_region_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 **:variable:‘query_response_time_stats‘** is ON, the execution times for these two `SELECT` queries will also be collected.

69.3.2 FLUSH

Flushing can be done by setting the **:variable:‘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 **:table:‘QUERY_RESPONSE_TIME‘**, **:table:‘QUERY_RESPONSE_TIME_READ‘**, and **:table:‘QUERY_RESPONSE_TIME_WRITE‘** tables
- Reads the value of **:variable:‘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.

69.3.3 Stored procedures

Stored procedure calls count as a single query.

69.3.4 Collect time point

Time is collected after query execution completes (before clearing data structures).

69.4 Version Specific Information

- **:rn:‘5.6.21-69.0‘**: Feature ported from *Percona Server for MySQL* 5.5 as a plugin
- **:rn:‘5.6.22-72.0‘**: Implemented query logging based on `READ` and `WRITE` queries.

69.5 System Variables

Setting this variable to `ON` will flush the statistics and re-read the **:variable:‘query_response_time_range_base‘**.

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;
```

This variable enables and disables collection of query times.

69.6 INFORMATION_SCHEMA Tables

CHAPTER

SEVENTY

INNODB PAGE FRAGMENTATION COUNTERS

InnoDB page fragmentation is caused by random insertion or deletion from a secondary index. This means that the physical ordering of the index pages on the disk is not same as the index ordering of the records on the pages. As a consequence this means that some pages take a lot more space and that queries which require a full table scan can take a long time to finish.

To provide more information about the *InnoDB* page fragmentation *Percona Server for MySQL* now provides the following counters as status variables: **:variable:'Innodb_scan_pages_contiguous'**, **:variable:'Innodb_scan_pages_disjointed'**, **:variable:'Innodb_scan_data_size'**, **:variable:'Innodb_scan_deleted_recs_size'**, and **:variable:'Innodb_scan_pages_total_seek_distance'**.

70.1 Version Specific Information

- **:rn:'5.6.38-83.0'**: Feature Implemented

70.2 Status Variables

This variable shows the number of contiguous page reads inside a query.

This variable shows the number of disjointed page reads inside a query.

This variable shows the size of data in all *InnoDB* pages read inside a query (in bytes) - calculated as the sum of `page_get_data_size(page)` for every page scanned.

This variable shows the size of deleted records (marked as `deleted` in `page_delete_rec_list_end()`) in all *InnoDB* pages read inside a query (in bytes) - calculated as the sum of `page_header_get_field(page, PAGE_GARBAGE)` for every page scanned.

This variable shows the total seek distance when moving between pages.

70.3 Related Reading

- InnoDB: look after fragmentation
- Defragmenting a Table

Part IX

TokuDB

CHAPTER

SEVENTYONE

TOKUDB INTRODUCTION

TokuDB is a highly scalable, zero-maintenance downtime MySQL storage engine that delivers indexing-based query acceleration, improved replication performance, unparalleled compression, and live schema modification. 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.

Percona Server for MySQL is fully compatible with the separately available *TokuDB* storage engine package. The *TokuDB* engine must be separately downloaded and then enabled as a plug-in component. This package can be installed alongside standard *Percona Server for MySQL* 5.6 releases starting with **:rn:‘5.6.19-67.0‘** and does not require any specially adapted version of *Percona Server for MySQL*.

Warning: Only the Percona supplied *TokuDB* engine should be used with *Percona Server for MySQL* 5.6. A *TokuDB* engine downloaded from other sources is not compatible. *TokuDB* file formats are not the same across *MySQL* variants. Migrating from one variant to any other variant requires a logical data dump and reload.

Additional features unique to *TokuDB* include:

- Up to 25x Data Compression
- Fast Inserts
- Eliminates Slave Lag with *Read Free Replication*
- Hot Schema Changes
- Hot Index Creation - *TokuDB* tables support insertions, deletions and queries with no down time while indexes are being added to that table
- Hot column addition, deletion, expansion, and rename - *TokuDB* tables support insertions, deletions and queries without down-time when an alter table adds, deletes, expands, or renames columns
- On-line Backup

For more information on installing and using *TokuDB* click on the following links:

71.1 TokuDB Installation

Percona Server for MySQL is compatible with the separately available *TokuDB* storage engine package. 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 for MySQL* 5.6 releases starting with **:rn:‘5.6.19-67.0‘** and does not require any specially adapted version of *Percona Server for MySQL*.

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.

Warning: Only the Percona supplied *TokuDB* engine should be used with *Percona Server for MySQL* 5.6. A *TokuDB* engine downloaded from other sources is not compatible. *TokuDB* file formats are not the same across *MySQL* variants. Migrating from one variant to any other variant requires a logical data dump and reload.

TokuDB is currently supported only for 64-bit Linux distributions. It is not available for Debian 6.0 (squeeze) due to conflicts with the **gcc** version required by *TokuDB*.

71.1.1 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 for MySQL* 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 **:variable:'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
```

71.1.2 Installation

TokuDB storage engine for *Percona Server for MySQL* is currently available in our *apt* and *yum* repositories.

You can install the *Percona Server for MySQL* 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
```

71.1.3 Enabling the TokuDB Storage Engine

Once the *TokuDB* server package has been installed following output will be shown:

```
* This release of Percona Server is distributed with TokuDB storage engine.
   * Run the following script to enable the TokuDB storage engine in Percona Server:

    ps_tokudb_admin --enable -u <mysql_admin_user> -p[mysql_admin_pass] [-S <socket>]␣
↪[-h <host> -P <port>]

   * See http://www.percona.com/doc/percona-server/5.6/tokudb/tokudb_installation.
↪html for more installation details

   * See http://www.percona.com/doc/percona-server/5.6/tokudb/tokudb_intro.html for␣
↪an introduction to TokuDB
```

Percona Server for MySQL **:rn:'5.6.22-72.0'** has implemented `ps_tokudb_admin` script to make the enabling the *TokuDB* storage engine easier. This script will automatically disable Transparent huge pages, if they're enabled, and install and enable the *TokuDB* storage engine with all the required plugins. You need to run this script as root or with **sudo**. The script should only be used for local installations and should not be used to install TokuDB to a remote server. After you run the script with required parameters:

```
ps_tokudb_admin --enable -uroot -pPassw0rd
```

Following output will be displayed:

```
Checking if Percona server is running with jemalloc enabled...
>> Percona server is running with jemalloc enabled.

Checking transparent huge pages status on the system...
>> Transparent huge pages are currently disabled on the system.

Checking if thp-setting=never option is already set in config file...
>> Option thp-setting=never is not set in the config file.
>> (needed only if THP is not disabled permanently on the system)

Checking TokuDB plugin status...
>> TokuDB plugin is not installed.
```

```
Adding thp-setting=never option into /etc/mysql/my.cnf
>> Successfuly added thp-setting=never option into /etc/mysql/my.cnf

Installing TokuDB engine...
>> Successfuly installed TokuDB plugin.
```

If the script returns no errors, *TokuDB* storage engine should be successfully enabled on your server. You can check it out by running:

```
mysql> SHOW ENGINES;
...
 | TokuDB | YES | Tokutek TokuDB Storage Engine with Fractal Tree(tm) Technology |␣
↪YES | YES | YES |
...
```

71.1.4 Enabling the TokuDB Storage Engine Manually

If you're running *Percona Server for MySQL* **:rn:'5.6.22-71.0'** this storage engine requires manual installation.

```
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';
INSTALL PLUGIN tokudb_background_job_status 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 ␣
↪    |
| TokuDB_background_job_status  | ACTIVE   | INFORMATION SCHEMA | ha_tokudb.so | GPL ␣
↪    |
```

```
...
```

71.1.5 TokuDB Version

TokuDB storage engine version can be checked with:

```
mysql> SELECT @@tokudb_version;
+------------------+
| @@tokudb_version |
+------------------+
| 5.6.27-76.0      |
+------------------+
1 row in set (0.00 sec)
```

Note: *TokuDB* storage engine has the same version as *Percona Server for MySQL* after **:rn:‘5.6.26-74.0‘** release.

71.1.6 Upgrade

Installing the *TokuDB* package is compatible with existing server setup and databases.

71.1.7 Version Specific Information

- **:rn:‘5.6.17-66.0‘** TokuDB storage engine available as a separate *Percona Server for MySQL* package.
- **:rn:‘5.6.19-67.0‘** TokuDB storage engine is considered GA quality.
- **:rn:‘5.6.22-72.0‘** Implemented `ps_tokudb_admin` script to make the installation *easier*

71.2 Using TokuDB

Warning: Do not move or modify any *TokuDB* files. You will break the database, and need to recover the database from a backup.

71.2.1 Fast Insertions and Richer Indexes

TokuDB’s fast indexing enables fast queries through the use of rich indexes, such as covering and clustering indexes. It’s worth investing some time to optimize index definitions to get the best performance from *MySQL* and *TokuDB*. Here are some resources to get you started:

- “Understanding Indexing” by Zardosht Kasheff (video)
- Rule of Thumb for Choosing Column Order in Indexes
- Covering Indexes: Orders-of-Magnitude Improvements
- Introducing Multiple Clustering Indexes
- Clustering Indexes vs. Covering Indexes

- How Clustering Indexes Sometimes Helps UPDATE and DELETE Performance
- *High Performance MySQL, 3rd Edition* by Baron Schwartz, Peter Zaitsev, Vadim Tkachenko, Copyright 2012, O'Reilly Media. See Chapter 5, *Indexing for High Performance*.

71.2.2 Clustering Secondary Indexes

One of the keys to exploiting TokuDB's strength in indexing is to make use of clustering secondary indexes.

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 for MySQL* 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 ...
```

To define a secondary index as clustering, simply add the word `CLUSTERING` before the key definition. For example:

```
CREATE TABLE foo (
  column_a INT,
  column_b INT,
  column_c INT,
  PRIMARY KEY index_a (column_a),
  CLUSTERING KEY index_b (column_b)) ENGINE = TokuDB;
```

In the previous example, the primary table is indexed on *column_a*. Additionally, there is a secondary clustering index (named *index_b*) sorted on *column_b*. Unlike non-clustered indexes, clustering indexes include all the columns of a table and can be used as covering indexes. For example, the following query will run very fast using the clustering *index_b*:

```
SELECT column_c
  FROM foo
  WHERE column_b BETWEEN 10 AND 100;
```

This index is sorted on *column_b*, making the `WHERE` clause fast, and includes *column_c*, which avoids lookups in the primary table to satisfy the query.

TokuDB makes clustering indexes feasible because of its excellent compression and very high indexing rates. For more information about using clustering indexes, see Introducing Multiple Clustering Indexes.

71.2.3 Hot Index Creation

TokuDB enables you to add indexes to an existing table and still perform inserts and queries on that table while the index is being created.

The `ONLINE` keyword is not used. Instead, the value of the **:variable:'tokudb_create_index_online'** client session variable is examined.

Hot index creation is invoked using the `CREATE INDEX` command after setting **:variable:'tokudb_create_index_online'** to `on` as follows:

```
mysql> SET tokudb_create_index_online=on;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE INDEX index ON foo (field_name);
```

Alternatively, using the `ALTER TABLE` command for creating an index will create the index offline (with the table unavailable for inserts or queries), regardless of the value of **:variable:'tokudb_create_index_online'**. The only way to hot create an index is to use the `CREATE INDEX` command.

Hot creating an index will be slower than creating the index offline, and progress depends how busy the mysqld server is with other tasks. Progress of the index creation can be seen by using the `SHOW PROCESSLIST` command (in another client). Once the index creation completes, the new index will be used in future query plans.

If more than one hot `CREATE INDEX` is issued for a particular table, the indexes will be created serially. An index creation that is waiting for another to complete will be shown as *Locked* in `SHOW PROCESSLIST`. We recommend that each `CREATE INDEX` be allowed to complete before the next one is started.

71.2.4 Hot Column Add, Delete, Expand, and Rename (HCADER)

TokuDB enables you to add or delete columns in an existing table, expand `char`, `varchar`, `varbinary`, and `integer` type columns in an existing table, or rename an existing column in a table with little blocking of other updates and queries. HCADER typically blocks other queries with a table lock for no more than a few seconds. After that initial short-term table locking, the system modifies each row (when adding, deleting, or expanding columns) later, when the row is next brought into main memory from disk. For column rename, all the work is done during the seconds of downtime. On-disk rows need not be modified.

To get good performance from HCADER, observe the following guidelines:

- The work of altering the table for column addition, deletion, or expansion is performed as subsequent operations touch parts of the Fractal Tree, both in the primary index and secondary indexes.

  You can force the column addition, deletion, or expansion work to be performed all at once using the standard syntax of `OPTIMIZE TABLE X`, when a column has been added to, deleted from, or expanded in table X. It is important to note that as of *TokuDB* version 7.1.0, `OPTIMIZE TABLE` is also hot, so that a table supports updates and queries without blocking while an `OPTIMIZE TABLE` is being performed. Also, a hot `OPTIMIZE TABLE` does not rebuild the indexes, since *TokuDB* indexes do not age. Rather, they flush all background work, such as that induced by a hot column addition, deletion, or expansion.

- Each hot column addition, deletion, or expansion operation must be performed individually (with its own SQL statement). If you want to add, delete, or expand multiple columns use multiple statements.

- Avoid adding, deleting, or expanding a column at the same time as adding or dropping an index.

- The time that the table lock is held can vary. The table-locking time for HCADER is dominated by the time it takes to flush dirty pages, because MySQL closes the table after altering it. If a checkpoint has happened recently, this operation is fast (on the order of seconds). However, if the table has many dirty pages, then the flushing stage can take on the order of minutes.
- Avoid dropping a column that is part of an index. If a column to be dropped is part of an index, then dropping that column is slow. To drop a column that is part of an index, first drop the indexes that reference the column in one alter table statement, and then drop the column in another statement.
- Hot column expansion operations are only supported to `char`, `varchar`, `varbinary`, and `integer` data types. Hot column expansion is not supported if the given column is part of the primary key or any secondary keys.
- Rename only one column per statement. Renaming more than one column will revert to the standard MySQL blocking behavior. The proper syntax is as follows:

```
ALTER TABLE table
  CHANGE column_old column_new
  DATA_TYPE REQUIRED_NESS DEFAULT
```

  Here's an example of how that might look:

```
ALTER TABLE table
  CHANGE column_old column_new
  INT(10) NOT NULL;
```

Notice that all of the column attributes must be specified. `ALTER TABLE table CHANGE column_old column_new;` induces a slow, blocking column rename.

- Hot column rename does not support the following data types: `TIME`, `ENUM`, `BLOB`, `TINYBLOB`, `MEDIUMBLOB`, `LONGBLOB`. Renaming columns of these types will revert to the standard MySQL blocking behavior.
- Temporary tables cannot take advantage of HCADER. Temporary tables are typically small anyway, so altering them using the standard method is usually fast.

71.2.5 Compression Details

TokuDB offers different levels of compression, which trade off between the amount of CPU used and the compression achieved. Standard compression uses less CPU but generally compresses at a lower level, high compression uses more CPU and generally compresses at a higher level. We have seen compression up to 25x on customer data.

Compression in *TokuDB* occurs on background threads, which means that high compression need not slow down your database. Indeed, in some settings, we've seen higher overall database performance with high compression.

Note: We recommend that users use standard compression on machines with six or fewer cores, and high compression on machines with more than six cores.

The ultimate choice depends on the particulars of how a database is used, and we recommend that users use the default settings unless they have profiled their system with high compression in place.

Compression is set on a per-table basis and is controlled by setting row format during a `CREATE TABLE` or `ALTER TABLE`. For example:

```
CREATE TABLE table (
  column_a INT NOT NULL PRIMARY KEY,
```

```
  column_b INT NOT NULL) ENGINE=TokuDB
  ROW_FORMAT=row_format;
```

If no row format is specified in a `CREATE TABLE`, the table is compressed using whichever row format is specified in the session variable **:variable:'tokudb_row_format'**. If no row format is set nor is **:variable:'tokudb_row_format'**, the zlib compressor is used.

:variable:'row_format' and **:variable:'tokudb_row_format'** variables accept the following values:

- `TOKUDB_DEFAULT`: This sets the compression to the default behavior. As of TokuDB 7.1.0, the default behavior is to compress using the zlib library. In the future this behavior may change.
- `TOKUDB_FAST`: This sets the compression to use the quicklz library.
- `TOKUDB_SMALL`: This sets the compression to use the lzma library.

In addition, you can choose a compression library directly, which will override previous values. The following libraries are available:

- `TOKUDB_ZLIB`: Compress using the zlib library, which provides mid-range compression and CPU utilization.
- `TOKUDB_QUICKLZ`: Compress using the quicklz library, which provides light compression and low CPU utilization.
- `TOKUDB_LZMA`: Compress using the lzma library, which provides the highest compression and high CPU utilization.
- `TOKUDB_SNAPPY` - This compression is using snappy library and aims for very high speeds and reasonable compression.
- `TOKUDB_UNCOMPRESSED`: This setting turns off compression and is useful for tables with data that cannot be compressed.

71.2.6 Changing Compression of a Table

Modify the compression used on a particular table with the following command:

```
ALTER TABLE table
  ROW_FORMAT=row_format;
```

Note: Changing the compression of a table only affects newly written data (dirtied blocks). After changing a table's compression you can run `OPTIMZE TABLE` to rewrite all blocks of the table and its indexes.

71.2.7 Read Free Replication

TokuDB slaves can be configured to perform significantly less read IO in order to apply changes from the master. By utilizing the power of Fractal Tree indexes:

- insert/update/delete operations can be configured to eliminate read-modify-write behavior and simply inject messages into the appropriate Fractal Tree indexes
- update/delete operations can be configured to eliminate the IO required for uniqueness checking

To enable Read Free Replication, the servers must be configured as follows:

- On the replication master:
  - Enable row based replication: set `BINLOG_FORMAT=ROW`

- On the replication slave(s):
  - The slave must be in read-only mode: set `read_only=1`
  - Disable unique checks: set `tokudb_rpl_unique_checks=0`
  - Disable lookups (read-modify-write): set `tokudb_rpl_lookup_rows=0`

Note: You can modify one or both behaviors on the slave(s).

Note: As long as the master is using row based replication, this optimization is available on a *TokuDB* slave. This means that it's available even if the master is using *InnoDB* or *MyISAM* tables, or running non-TokuDB binaries.

Warning: *TokuDB* Read Free Replication will not propagate `UPDATE` and `DELETE` events reliably if *TokuDB* table is missing the primary key which will eventually lead to data inconsistency on the slave.

71.2.8 Transactions and ACID-compliant Recovery

By default, *TokuDB* checkpoints all open tables regularly and logs all changes between checkpoints, so that after a power failure or system crash, *TokuDB* will restore all tables into their fully ACID-compliant state. That is, all committed transactions will be reflected in the tables, and any transaction not committed at the time of failure will be rolled back.

The default checkpoint period is every 60 seconds, and this specifies the time from the beginning of one checkpoint to the beginning of the next. If a checkpoint requires more than the defined checkpoint period to complete, the next checkpoint begins immediately. It is also related to the frequency with which log files are trimmed, as described below. The user can induce a checkpoint at any time by issuing the `FLUSH LOGS` command. When a database is shut down normally it is also checkpointed and all open transactions are aborted. The logs are trimmed at startup.

71.2.9 Managing Log Size

TokuDB keeps log files back to the most recent checkpoint. Whenever a log file reaches 100 MB, a new log file is started. Whenever there is a checkpoint, all log files older than the checkpoint are discarded. If the checkpoint period is set to be a very large number, logs will get trimmed less frequently. This value is set to 60 seconds by default.

TokuDB also keeps rollback logs for each open transaction. The size of each log is proportional to the amount of work done by its transaction and is stored compressed on disk. Rollback logs are trimmed when the associated transaction completes.

71.2.10 Recovery

Recovery is fully automatic with *TokuDB*. *TokuDB* uses both the log files and rollback logs to recover from a crash. The time to recover from a crash is proportional to the combined size of the log files and uncompressed size of rollback logs. Thus, if there were no long-standing transactions open at the time of the most recent checkpoint, recovery will take less than a minute.

71.2.11 Disabling the Write Cache

When using any transaction-safe database, it is essential that you understand the write-caching characteristics of your hardware. *TokuDB* provides transaction safe (ACID compliant) data storage for *MySQL*. However, if the underlying operating system or hardware does not actually write data to disk when it says it did, the system can corrupt your database when the machine crashes. For example, *TokuDB* can not guarantee proper recovery if it is mounted on an NFS volume. It is always safe to disable the write cache, but you may be giving up some performance.

For most configurations you must disable the write cache on your disk drives. On ATA/SATA drives, the following command should disable the write cache:

```
$ hdparm -W0 /dev/hda
```

There are some cases when you can keep the write cache, for example:

- Write caching can remain enabled when using XFS, but only if XFS reports that disk write barriers work. If you see one of the following messages in /var/log/messages, then you must disable the write cache:
  - `Disabling barriers, not supported with external log device`
  - `Disabling barriers, not supported by the underlying device`
  - `Disabling barriers, trial barrier write failed`

  XFS write barriers appear to succeed for single disks (with no LVM), or for very recent kernels (such as that provided by Fedora 12). For more information, see the XFS FAQ.

In the following cases, you must disable the write cache:

- If you use the ext3 filesystem
- If you use LVM (although recent Linux kernels, such as Fedora 12, have fixed this problem)
- If you use Linux's software RAID
- If you use a RAID controller with battery-backed-up memory. This may seem counter-intuitive. For more information, see the XFS FAQ

In summary, you should disable the write cache, unless you have a very specific reason not to do so.

71.2.12 Progress Tracking

TokuDB has a system for tracking progress of long running statements, thereby removing the need to define triggers to track statement execution, as follows:

- Bulk Load: When loading large tables using `LOAD DATA INFILE` commands, doing a `SHOW PROCESSLIST` command in a separate client session shows progress. There are two progress stages. The first will state something like `Inserted about 1000000 rows`. After all rows are processed like this, the next stage tracks progress by showing what fraction of the work is done (e.g. `Loading of data about 45% done`)
- Adding Indexes: When adding indexes via `ALTER TABLE` or `CREATE INDEX`, the command `SHOW PROCESSLIST` shows progress. When adding indexes via `ALTER TABLE` or `CREATE INDEX`, the command `SHOW PROCESSLIST` will include an estimation of the number of rows processed. Use this information to verify progress is being made. Similar to bulk loading, the first stage shows how many rows have been processed, and the second stage shows progress with a fraction.
- Commits and Aborts: When committing or aborting a transaction, the command `SHOW PROCESSLIST` will include an estimate of the transactional operations processed.

71.2.13 Migrating to TokuDB

To convert an existing table to use the *TokuDB* engine, run `ALTER TABLE... ENGINE=TokuDB`. If you wish to load from a file, use `LOAD DATA INFILE` and not `mysqldump`. Using `mysqldump` will be much slower. To create a file that can be loaded with `LOAD DATA INFILE`, refer to the `INTO OUTFILE` option of the SELECT Syntax.

Note: Creating this file does not save the schema of your table, so you may want to create a copy of that as well.

71.3 Getting Started with TokuDB

71.3.1 System and Hardware Requirements

Operating Systems: *TokuDB* is currently supported on 64-bit Linux only.

Memory: *TokuDB* Requires at least 1GB of main memory but for best results, we recommend to run with at least 2GB of main memory.

Disk space and configuration: Please make sure to allocate enough disk space for data, indexes and logs. In our users' experience, *TokuDB* achieves up to 25x space savings on data and indexes over *InnoDB* due to high compression.

71.3.2 Creating Tables and Loading Data

Creating TokuDB Tables

TokuDB tables are created the same way as other tables in *MySQL* by specifying `ENGINE=TokuDB` in the table definition. For example, the following command creates a table with a single column and uses the *TokuDB* storage engine to store its data:

```
CREATE TABLE table (
  id INT(11) NOT NULL) ENGINE=TokuDB;
```

Loading Data

Once *TokuDB* tables have been created, data can be inserted or loaded using standard *MySQL* insert or bulk load operations. For example, the following command loads data from a file into the table:

```
LOAD DATA INFILE file
  INTO TABLE table;
```

Note: For more information about loading data, see the *MySQL* 5.6 reference manual.

Migrating Data from an Existing Database

Use the following command to convert an existing table for the *TokuDB* storage engine:

```
ALTER TABLE table
  ENGINE=TokuDB;
```

71.3.3 Bulk Loading Data

The *TokuDB* bulk loader imports data much faster than regular *MySQL* with *InnoDB*. To make use of the loader you need flat files in either comma separated or tab separated format. The *MySQL* `LOAD DATA INFILE ...` statement will invoke the bulk loader if the table is empty. Keep in mind that while this is the most convenient and, in most cases, the fastest way to initialize a *TokuDB* table, it may not be replication safe if applied to the master

For more information, see the *MySQL* 5.6 Reference Manual: LOAD DATA INFILE.

To obtain the logical backup and then bulk load into *TokuDB*, follow these steps:

1. Create a logical backup of the original table. The easiest way to achieve this is using `SELECT ... INTO OUTFILE`. Keep in mind that the file will be created on the server.

   ```
   SELECT * FROM table
     INTO OUTFILE 'file.csv';
   ```

2. The output file should either be copied to the destination server or the client machine from which you plan to load it.

3. To load the data into the server use `LOAD DATA INFILE`. If loading from a machine other than the server use the keyword `LOCAL` to point to the file on local machine. Keep in mind that you will need enough disk space on the temporary directory on the server since the local file will be copied onto the server by the MySQL client utility.

   ```
   LOAD DATA [LOCAL] INFILE 'file.csv';
   ```

It is possible to create the CSV file using either **mysqldump** or the *MySQL* client utility as well, in which case the resulting file will reside on a local directory. In these 2 cases you have to make sure to use the correct command line options to create a file compatible with `LOAD DATA INFILE`.

The bulk loader will use more space than normal for logs and temporary files while running, make sure that your file system has enough disk space to process your load. As a rule of thumb, it should be approximately 1.5 times the size of the raw data.

Note: Please read the original *MySQL* documentation to understand the needed privileges and replication issues needed around `LOAD DATA INFILE`.

71.3.4 Considerations to Run TokuDB in Production

In most cases, the default options should be left in-place to run *TokuDB*, however it is a good idea to review some of the configuration parameters.

Memory allocation

TokuDB will allocate 50% of the installed RAM for its own cache (global variable **:variable:‘tokudb_cache_size‘**). While this is optimal in most situations, there are cases where it may lead to memory over allocation. If the system tries to allocate more memory than is available, the machine will begin swapping and run much slower than normal.

It is necessary to set the **:variable:‘tokudb_cache_size‘** to a value other than the default in the following cases:

- **Running other memory heavy processes on the same server as TokuDB**: In many cases, the database process needs to share the system with other server processes like additional database instances, http server, application server, e-mail server, monitoring systems and others. In order to properly configure TokuDB's memory consumption, it's important to understand how much free memory will be left and assign a sensible value for *TokuDB*. There is no fixed rule, but a conservative choice would be 50% of available RAM while all the other processes are running. If the result is under 2 GB, you should consider moving some of the other processes to a different system or using a dedicated database server.

  :variable:'tokudb_cache_size' is a static variable, so it needs to be set before starting the server and cannot be changed while the server is running. For example, to set up TokuDB's cache to 4G, add the following line to your `my.cnf` file:

```
tokudb_cache_size = 4G
```

- **System using InnoDB and TokuDB**: When using both the *TokuDB* and *InnoDB* storage engines, you need to manage the cache size for each. For example, on a server with 16 GB of RAM you could use the following values in your configuration file:

```
innodb_buffer_pool_size = 2G
tokudb_cache_size = 8G
```

- **Using TokuDB with Federated or FederatedX tables**: The Federated engine in *MySQL* and FederatedX in *MariaDB* allow you to connect to a table on a remote server and query it as if it were a local table (please see the *MySQL* documentation: 14.11. The FEDERATED Storage Engine for details). When accessing the remote table, these engines could import the complete table contents to the local server to execute a query. In this case, you will have to make sure that there is enough free memory on the server to handle these remote tables. For example, if your remote table is 8 GB in size, the server has to have more than 8 GB of free RAM to process queries against that table without going into swapping or causing a kernel panic and crash the MySQL process. There are no parameters to limit the amount of memory that the Federated or FederatedX engine will allocate while importing the remote dataset.

Specifying the Location for Files

As with *InnoDB*, it is possible to specify different locations than the default for TokuDB's data, log and temporary files. This way you may distribute the load and control the disk space. The following variables control file location:

- **:variable:'tokudb_data_dir'**: This variable defines the directory where the *TokuDB* tables are stored. The default location for TokuDB's data files is the *MySQL* data directory.
- **:variable:'tokudb_log_dir'**: This variable defines the directory where the *TokuDB* log files are stored. The default location for TokuDB's log files is the *MySQL* data directory. Configuring a separate log directory is somewhat involved and should be done only if absolutely necessary. We recommend to keep the data and log files under the same directory.
- **:variable:'tokudb_tmp_dir'**: This variable defines the directory where the *TokuDB* bulk loader stores temporary files. The bulk loader can create large temporary files while it is loading a table, so putting these temporary files on a disk separate from the data directory can be useful. For example, it can make sense to use a high-performance disk for the data directory and a very inexpensive disk for the temporary directory. The default location for TokuDB's temporary files is the MySQL data directory.

Table Maintenance

Overview

The fractal tree provides fast performance by inserting small messages in the buffers in the fractal trees instead of requiring a potential IO for an update on every row in the table as required by a B-tree. Additional background

information on how fractal trees operate can be found here. For tables whose workload pattern is a high number of sequential deletes, it may be beneficial to flush these delete messages down to the basement nodes in order to allow for faster access. The way to perform this operation is via the `OPTIMIZE` command.

The following extensions to the `OPTIMIZE` command have been added in *TokuDB* version 7.5.5:

- **Hot Optimize Throttling**

  By default, table optimization will run with all available resources. To limit the amount of resources, it is possible to limit the speed of table optimization. The **:variable:'tokudb_optimize_throttle'** session variable determines an upper bound on how many fractal tree leaf nodes per second are optimized. The default is 0 (no upper bound) with a valid range of [0,1000000]. For example, to limit the table optimization to 1 leaf node per second, use the following setting:

```
SET tokudb_optimize_throttle=1;
```

- **Optimize a Single Index of a Table**

  To optimize a single index in a table, the **:variable:'tokudb_optimize_index_name'** session variable can be set to select the index by name. For example, to optimize the primary key of a table:

```
SET tokudb_optimize_index_name='primary';
OPTIMIZE TABLE t;
```

- **Optimize a Subset of a Fractal Tree Index**

  For patterns where the left side of the tree has many deletions (a common pattern with increasing id or date values), it may be useful to delete a percentage of the tree. In this case, it is possible to optimize a subset of a fractal tree starting at the left side. The **:variable:'tokudb_optimize_index_fraction'** session variable controls the size of the sub tree. Valid values are in the range [0.0,1.0] with default 1.0 (optimize the whole tree). For example, to optimize the leftmost 10% of the primary key:

```
SET tokudb_optimize_index_name='primary';
SET tokudb_optimize_index_fraction=0.1;
OPTIMIZE TABLE t;
```

71.4 TokuDB Variables

Like all storage engines, *TokuDB* has variables to tune performance and control behavior. Fractal Tree algorithms are designed for near optimal performance and TokuDB's default settings should work well in most situations, eliminating the need for complex and time consuming tuning in most cases.

71.4.1 TokuDB Server Variables

Name	Cmd-Line	Option File	Var Scope	Dynamic
:variable:'tokudb_alter_print_error'	Yes	Yes	Session, Global	Yes

Continued on next page

Table 71.1 – continued from previous page

Name	Cmd-Line	Option File	Var Scope	Dynamic
:variable:‘tokudb_analyze_delete_fraction‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_analyze_in_background‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_analyze_mode‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_analyze_throttle‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_analyze_time‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_auto_analyze‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_backup_allowed_prefix‘	No	Yes	Global	No
:variable:‘tokudb_backup_dir‘	No	Yes	Session	No
:variable:‘tokudb_backup_exclude‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_backup_last_error‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_backup_last_error_string‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_backup_plugin_version‘	No	No	Global	No
:variable:‘tokudb_backup_throttle‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_backup_version‘	No	No	Global	No
:variable:‘tokudb_block_size‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_bulk_fetch‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_cache_size‘	Yes	Yes	Global	No
:variable:‘tokudb_cachetable_pool_threads‘	Yes	Yes	Global	No
:variable:‘tokudb_cardinality_scale_percent‘	Yes	Yes	Global	Yes
:variable:‘tokudb_check_jemalloc‘	Yes	Yes	Global	No
:variable:‘tokudb_checkpoint_lock‘	Yes	Yes	Global	No
:variable:‘tokudb_checkpoint_on_flush_logs‘	Yes	Yes	Global	Yes
:variable:‘tokudb_checkpoint_pool_threads‘	Yes	Yes	Global	Yes
:variable:‘tokudb_checkpointing_period‘	Yes	Yes	Global	Yes
:variable:‘tokudb_cleaner_iterations‘	Yes	Yes	Global	Yes
:variable:‘tokudb_cleaner_period‘	Yes	Yes	Global	Yes
:variable:‘tokudb_client_pool_threads‘	Yes	Yes	Global	No
:variable:‘tokudb_commit_sync‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_compress_buffers_before_eviction‘	Yes	Yes	Global	No
:variable:‘tokudb_create_index_online‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_data_dir‘	Yes	Yes	Global	No
:variable:‘tokudb_debug‘	Yes	Yes	Global	Yes
:variable:‘tokudb_dir_per_db‘	Yes	Yes	Global	Yes
:variable:‘tokudb_directio‘	Yes	Yes	Global	No

Continued on next page

Table 71.1 – continued from previous page

Name	Cmd-Line	Option File	Var Scope	Dynamic
:variable:‘tokudb_disable_hot_alter‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_disable_prefetching‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_disable_slow_alter‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_disable_slow_update‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_disable_slow_upsert‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_empty_scan‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_enable_fast_update‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_enable_fast_upsert‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_enable_partial_eviction‘	Yes	Yes	Global	No
:variable:‘tokudb_fanout‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_fs_reserve_percent‘	Yes	Yes	Global	No
:variable:‘tokudb_fsync_log_period‘	Yes	Yes	Global	Yes
:variable:‘tokudb_hide_default_row_format‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_killed_time‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_last_lock_timeout‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_load_save_space‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_loader_memory_size‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_lock_timeout‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_lock_timeout_debug‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_log_dir‘	Yes	Yes	Global	No
:variable:‘tokudb_max_lock_memory‘	Yes	Yes	Global	No
:variable:‘tokudb_optimize_index_fraction‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_optimize_index_name‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_optimize_throttle‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_pk_insert_mode‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_prelock_empty‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_read_block_size‘	Yes	Yes	Session, Global	Yes

Continued on next page

Table 71.1 – continued from previous page

Name	Cmd-Line	Option File	Var Scope	Dynamic
:variable:‘tokudb_read_buf_size‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_read_status_frequency‘	Yes	Yes	Global	Yes
:variable:‘tokudb_row_format‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_rpl_check_readonly‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_rpl_lookup_rows‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_rpl_lookup_rows_delay‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_rpl_unique_checks‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_rpl_unique_checks_delay‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_strip_frm_data‘	Yes	Yes	Global	No
:variable:‘tokudb_support_xa‘	Yes	Yes	Session, Global	Yes
:variable:‘tokudb_tmp_dir‘	Yes	Yes	Global	No
:variable:‘tokudb_version‘	No	No	Global	No
:variable:‘tokudb_write_status_frequency‘	Yes	Yes	Global	Yes

When set to `ON` errors will be printed to the client during the `ALTER TABLE` operations on *TokuDB* tables.

This variables controls whether or not deleted rows in the fractal tree are reported to the client and to the *MySQL* error log during an `ANALYZE TABLE` operation on a *TokuDB* table. When set to `1`, nothing is reported. When set to `0.1` and at least 10% of the rows scanned by `ANALYZE` were deleted rows that are not yet garbage collected, a report is returned to the client and the *MySQL* error log.

This system-level variable restricts the location of the destination directory where the backups can be located. Attempts to backup to a location outside of the directory this variable points to or its children will result in an error.

The default is NULL, backups have no restricted locations. This read only variable can be set in the `my.cnf` configuration file and displayed with the `SHOW VARIABLES` command when *Percona TokuBackup* plugin is loaded.

```
mysql> SHOW VARIABLES LIKE 'tokudb_backup_allowed_prefix';
+-------------------------------+-----------+
| Variable_name                 | Value     |
+-------------------------------+-----------+
| tokudb_backup_allowed_prefix  | /dumpdir  |
+-------------------------------+-----------+
```

When enabled, this session level variable serves two purposes, to point to the destination directory where the backups will be dumped and to kick off the backup as soon as it is set. For more information see *Percona TokuBackup*.

Use this variable to set a regular expression that defines source files excluded from backup. For example, to exclude all `lost+found` directories, use the following command:

```
mysql> set tokudb_backup_exclude='/lost\\+found($|/)';
```

For more information see *Percona TokuBackup*.

This session variable will contain the error number from the last backup. `0` indicates success. For more information see *Percona TokuBackup*.

This session variable will contain the error string from the last backup. For more information see *Percona TokuBackup*.

This read-only server variable documents the version of the *TokuBackup* plugin. For more information see *Percona TokuBackup*.

This variable specifies the maximum number of bytes per second the copier of a hot backup process will consume. Lowering its value will cause the hot backup operation to take more time but consume less I/O on the server. The default value is `18446744073709551615` which means no throttling. For more information see *Percona TokuBackup*.

This read-only server variable documents the version of the hot backup library. For more information see *Percona TokuBackup*.

This variable controls the maximum size of node in memory before messages must be flushed or node must be split.

Changing the value of **:variable:'tokudb_block_size'** only affects subsequently created tables and indexes. The value of this variable cannot be changed for an existing table/index without a dump and reload.

This variable determines if our bulk fetch algorithm is used for `SELECT` statements. `SELECT` statements include pure `SELECT ...` statements, as well as `INSERT INTO table-name ... SELECT ...`, `CREATE TABLE table-name ... SELECT ...`, `REPLACE INTO table-name ... SELECT ...`, `INSERT IGNORE INTO table-name ... SELECT ...`, and `INSERT INTO table-name ... SELECT ... ON DUPLICATE KEY UPDATE`.

This variable configures the size in bytes of the *TokuDB* cache table. The default cache table size is 1/2 of physical memory. Percona highly recommends using the default setting if using buffered I/O, if using direct I/O then consider setting this parameter to 80% of available memory.

Consider decreasing **:variable:'tokudb_cache_size'** if excessive swapping is causing performance problems. Swapping may occur when running multiple *MySQL* server instances or if other running applications use large amounts of physical memory.

This variable defines the number of threads for the cachetable worker thread pool. This pool is used to perform node prefetches, and to serialize, compress, and write nodes during cachetable eviction. The default value of 0 calculates the pool size to be `num_cpu_threads * 2`.

This variable enables/disables startup checking if jemalloc is linked and correct version and that transparent huge pages are disabled. Used for testing only.

Disables checkpointing when true. Session variable but acts like a global, any session disabling checkpointing disables it globally. If a session sets this lock and disconnects or terminates for any reason, the lock will not be released. Special purpose only, do **not** use this in your application.

When enabled forces a checkpoint if we get a flush logs command from the server.

This defines the number of threads for the checkpoint worker thread pool. This pool is used to serialize, compress and write nodes cloned during checkpoint. Default of `0` uses old algorithm to set pool size to `num_cpu_threads/4`.

This variable specifies the time in seconds between the beginning of one checkpoint and the beginning of the next. The default time between *TokuDB* checkpoints is 60 seconds. We recommend leaving this variable unchanged.

This variable specifies how many internal nodes get processed in each **:variable:'tokudb_cleaner_period'** period. The default value is `5`. Setting this variable to `0` turns off cleaner threads.

This variable specifies how often in seconds the cleaner thread runs. The default value is `1`. Setting this variable to `0` turns off cleaner threads.

This variable defines the number of threads for the client operations thread pool. This pool is used to perform node maintenance on over/undersized nodes such as message flushing down the tree, node splits, and node merges. Default of `0` uses old algorithm to set pool size to `1 * num_cpu_threads`.

Session variable **:variable:‘tokudb_commit_sync‘** controls whether or not the transaction log is flushed when a transaction commits. The default behavior is that the transaction log is flushed by the commit. Flushing the transaction log requires a disk write and may adversely affect the performance of your application.

To disable synchronous flushing of the transaction log, disable the **:variable:‘tokudb_commit_sync‘** session variable as follows:

```
SET tokudb_commit_sync=OFF;
```

Disabling this variable may make the system run faster. However, transactions committed since the last checkpoint are not guaranteed to survive a crash.

Warning: By disabling this variable and/or setting the **:variable:‘tokudb_fsync_log_period‘** to non-zero value you have effectively downgraded the durability of the storage engine. If you were to have a crash in this same window, you would lose data. The same issue would also appear if you were using some kind of volume snapshot for backups.

When this variable is enabled it allows the evictor to compress unused internal node partitions in order to reduce memory requirements as a first step of partial eviction before fully evicting the partition and eventually the entire node.

This variable controls whether indexes created with the `CREATE INDEX` command are hot (if enabled), or offline (if disabled). Hot index creation means that the table is available for inserts and queries while the index is being created. Offline index creation means that the table is not available for inserts and queries while the index is being created.

Note: Hot index creation is slower than offline index creation.

This variable configures the directory name where the *TokuDB* tables are stored. The default value is `NULL` which uses the location of the *MySQL* data directory. For more information check *TokuDB files and file types* and *TokuDB file management*.

This variable enables mysqld debug printing to `STDERR` for *TokuDB*. Produces tremendous amounts of output that is nearly useless to anyone but a *TokuDB* developer, not recommended for any production use at all. It is a mask value `ULONG`:

```
#define TOKUDB_DEBUG_INIT                   (1<<0)
#define TOKUDB_DEBUG_OPEN                   (1<<1)
#define TOKUDB_DEBUG_ENTER                  (1<<2)
#define TOKUDB_DEBUG_RETURN                 (1<<3)
#define TOKUDB_DEBUG_ERROR                  (1<<4)
#define TOKUDB_DEBUG_TXN                    (1<<5)
#define TOKUDB_DEBUG_AUTO_INCREMENT         (1<<6)
#define TOKUDB_DEBUG_INDEX_KEY              (1<<7)
#define TOKUDB_DEBUG_LOCK                   (1<<8)
#define TOKUDB_DEBUG_CHECK_KEY              (1<<9)
#define TOKUDB_DEBUG_HIDE_DDL_LOCK_ERRORS   (1<<10)
#define TOKUDB_DEBUG_ALTER_TABLE            (1<<11)
#define TOKUDB_DEBUG_UPSERT                 (1<<12)
#define TOKUDB_DEBUG_CHECK                  (1<<13)
#define TOKUDB_DEBUG_ANALYZE                (1<<14)
#define TOKUDB_DEBUG_XA                     (1<<15)
#define TOKUDB_DEBUG_SHARE                  (1<<16)
```

When this variable is set to `ON` all new tables and indices will be placed within their corresponding database directory within the **:variable:‘tokudb_data_dir‘** or system datadir. Existing table files will not be automatically relocated

to their corresponding database directory. If you rename a table, while this variable is enabled, the mapping in the *Percona FT* directory file will be updated and the files will be renamed on disk to reflect the new table name. For more information check *TokuDB files and file types* and *TokuDB file management*.

When enabled, *TokuDB* employs Direct I/O rather than Buffered I/O for writes. When using Direct I/O, consider increasing **:variable:‘tokudb_cache_size‘** from its default of 1/2 physical memory.

This variable is used specifically for testing or to disable hot alter in case there are bugs. Not for use in production.

TokuDB attempts to aggressively prefetch additional blocks of rows, which is helpful for most range queries but may create unnecessary I/O for range queries with `LIMIT` clauses. Prefetching is `ON` by default, with a value of `0`, it can be disabled by setting this variable to `1`.

This variable is used specifically for testing or to disable hot alter in case there are bugs. Not for use in production. It controls whether slow alter tables are allowed. For example, the following command is slow because `HCADER` does not allow a mixture of column additions, deletions, or expansions:

```
ALTER TABLE table
ADD COLUMN column_a INT,
DROP COLUMN column_b;
```

By default, **:variable:‘tokudb_disable_slow_alter‘** is disabled, and the engine reports back to *MySQL* that this is unsupported resulting in the following output:

```
ERROR 1112 (42000): Table 'test_slow' uses an extension that doesn't exist in this␣
↪MySQL version
```

This variable is used specifically for testing or to disable slow update in case there are bugs. Not for use in production.

This variable is used specifically for testing or to disable slow upsert in case there are bugs. Not for use in production.

Defines direction to be used to perform table scan to check for empty tables for bulk loader.

Toggles the fast updates ON/OFF for the `UPDATE` statement. Fast update is an experimental feature that involves queries optimization to avoid random reads during their execution.

Toggles the fast updates ON/OFF for the `INSERT` statement. Fast update is an experimental feature that involves queries optimization to avoid random reads during their execution.

This variable is used to control if partial eviction of nodes is enabled or disabled.

This variable controls the Fractal Tree fanout. The fanout defines the number of pivot keys or child nodes for each internal tree node. Changing the value of :variable:tokudb_fanout only affects subsequently created tables and indexes. The value of this variable cannot be changed for an existing table/index without a dump and reload.

This variable controls the percentage of the file system that must be available for inserts to be allowed. By default, this is set to `5`. We recommend that this reserve be at least half the size of your physical memory. See *Full Disks* for more information.

This variable controls the frequency, in milliseconds, for `fsync()` operations. If set to `0` then the `fsync()` behavior is only controlled by the **:variable:‘tokudb_commit_sync‘**, which can be `ON` or `OFF`.

This variable is used to hide the `ROW_FORMAT` in `SHOW CREATE TABLE`. If `zlib` compression is used, row format will show as `DEFAULT`.

This variable is used to specify frequency in milliseconds for lock wait to check to see if the lock was killed.

This variable contains a JSON document that describes the last lock conflict seen by the current *MySQL* client. It gets set when a blocked lock request times out or a lock deadlock is detected.

The **:variable:‘tokudb_lock_timeout_debug‘** session variable must have bit `0` set for this behavior, otherwise this session variable will be empty.

This session variable changes the behavior of the bulk loader. When it is disabled the bulk loader stores intermediate data using uncompressed files (which consumes additional CPU), whereas `ON` compresses the intermediate files.

Note: The location of the temporary disk space used by the bulk loader may be specified with the **:variable:'tokudb_tmp_dir'** server variable.

If a `LOAD DATA INFILE` statement fails with the error message `ERROR 1030 (HY000): Got error 1` from storage engine, then there may not be enough disk space for the optimized loader, so disable **:variable:'tokudb_prelock_empty'** and try again. More information is available in *Known Issues*.

This variable limits the amount of memory (in bytes) that the *TokuDB* bulk loader will use for each loader instance. Increasing this value may provide a performance benefit when loading extremely large tables with several secondary indexes.

Note: Memory allocated to a loader is taken from the *TokuDB* cache, defined in **:variable:'tokudb_cache_size'**, and may impact the running workload's performance as existing cached data must be ejected for the loader to begin.

This variable controls the amount of time that a transaction will wait for a lock held by another transaction to be released. If the conflicting transaction does not release the lock within the lock timeout, the transaction that was waiting for the lock will get a lock timeout error. The units are milliseconds. A value of `0` disables lock waits. The default value is 4000 (four seconds).

If your application gets a `lock wait timeout` error (-30994), then you may find that increasing the **:variable:'tokudb_lock_timeout'** may help. If your application gets a `deadlock found` error (-30995), then you need to abort the current transaction and retry it.

The following values are available:

- `0`: No lock timeouts or lock deadlocks are reported.
- `1`: A JSON document that describes the lock conflict is stored in the **:variable:'tokudb_last_lock_timeout'** session variable
- `2`: A JSON document that describes the lock conflict is printed to the MySQL error log.

  In addition to the JSON document describing the lock conflict, the following lines are printed to the MySQL error log:

  - A line containing the blocked thread id and blocked SQL
  - A line containing the blocking thread id and the blocking SQL.
- `3`: A JSON document that describes the lock conflict is stored in the **:variable:'tokudb_last_lock_timeout'** session variable and is printed to the MySQL error log.

  In addition to the JSON document describing the lock conflict, the following lines are printed to the MySQL error log:

  - A line containing the blocked thread id and blocked SQL
  - A line containing the blocking thread id and the blocking SQL.

This variable specifies the directory where the *TokuDB* log files are stored. The default value is `NULL` which uses the location of the *MySQL* data directory. Configuring a separate log directory is somewhat involved. Please contact Percona support for more details. For more information check *TokuDB files and file types* and *TokuDB file management*.

Warning: After changing *TokuDB* log directory path, the old *TokuDB* recovery log file should be moved to new directory prior to start of *MySQL* server and log file's owner must be the `mysql` user. Otherwise server will fail to initialize the *TokuDB* store engine restart.

This variable specifies the maximum amount of memory for the PerconaFT lock table.

For patterns where the left side of the tree has many deletions (a common pattern with increasing id or date values), it may be useful to delete a percentage of the tree. In this case, it's possible to optimize a subset of a fractal tree starting at the left side. The **:variable:'tokudb_optimize_index_fraction'** session variable controls the size of the sub tree. Valid values are in the range [0.0,1.0] with default 1.0 (optimize the whole tree).

This variable can be used to optimize a single index in a table, it can be set to select the index by name.

By default, table optimization will run with all available resources. To limit the amount of resources, it is possible to limit the speed of table optimization. This determines an upper bound on how many fractal tree leaf nodes per second are optimized. The default `0` imposes no limit.

This variable controls the behavior of primary key insertions with the command `REPLACE INTO` and `INSERT IGNORE` on tables with no secondary indexes and on tables whose secondary keys whose every column is also a column of the primary key.

For instance, the table `(column_a INT, column_b INT, column_c INT, PRIMARY KEY (column_a,column_b), KEY (column_b))` is affected, because the only column in the key of `column_b` is present in the primary key. *TokuDB* can make these insertions really fast on these tables. However, triggers may not work and row based replication definitely will not work in this mode. This variable takes the following values, to control this behavior. This only applies to tables described above, using the command `REPLACE INTO` or `INSERT IGNORE`. All other scenarios are unaffected.

- `0`: Insertions are fast, regardless of whether triggers are defined on the table. `REPLACE INTO` and `INSERT IGNORE` statements fail if row based replication is enabled. This value has been deprecated in *Percona Server for MySQL* **:rn:'5.6.30-76.3'**. Trying to set variable to this value on *Percona Server for MySQL* **:rn:'5.6.30-76.3'** or newer will generate a warning and the mode will be set back to default (`1`).
- `1` (default): Insertions are fast, if there are no triggers defined on the table. *TokuDB* storage engine with **:variable:'tokudb_pk_insert_mode'** set to `1` is safe to use in all conditions. On `INSERT IGNORE` or `REPLACE INTO`, it tests to see if triggers exist on the table, or replication is active with `!BINLOG_FORMAT_STMT` before it allows the optimization. If either of these conditions are met, then it falls back to the "safe" operation of looking up the target row first.
- `2`: Insertions are slow, all triggers on the table work, and row based replication works on `REPLACE INTO` and `INSERT IGNORE` statements.

By default *TokuDB* preemptively grabs an entire table lock on empty tables. If one transaction is doing the loading, such as when the user is doing a table load into an empty table, this default provides a considerable speedup.

However, if multiple transactions try to do concurrent operations on an empty table, all but one transaction will be locked out. Disabling **:variable:'tokudb_prelock_empty'** optimizes for this multi-transaction case by turning off preemptive pre-locking.

Note: If this variable is set to `OFF`, fast bulk loading is turned off as well.

Fractal tree leaves are subdivided into read blocks, in order to speed up point queries. This variable controls the target uncompressed size of the read blocks. The units are bytes and the default is 65,536 (64 KB). A smaller value favors read performance for point and small range scans over large range scans and higher compression. The minimum value of this variable is 4096.

Changing the value of **:variable:‘tokudb_read_block_size‘** only affects subsequently created tables. The value of this variable cannot be changed for an existing table without a dump and reload.

This variable controls the size of the buffer used to store values that are bulk fetched as part of a large range query. Its unit is bytes and its default value is 131,072 (128 KB).

A value of `0` turns off bulk fetching. Each client keeps a thread of this size, so it should be lowered if situations where there are a large number of clients simultaneously querying a table.

This variable controls in how many reads the progress is measured to display `SHOW PROCESSLIST`. Reads are defined as `SELECT` queries.

For slow queries, it can be helpful to set this variable and **:variable:‘tokudb_write_status_frequency‘** to `1`, and then run `SHOW PROCESSLIST` several times to understand what progress is being made.

This controls the default compression algorithm used to compress data when no row format is specified in the `CREATE TABLE` command. For more information on compression algorithms see *Compression Details*.

The *TokuDB* replication code will run row events from the binary log with *Read Free Replication* when the slave is in read-only mode. This variable is used to disable the slave read only check in the *TokuDB* replication code.

This allows Read-Free-Replication to run when the slave is NOT read-only. By default, **:variable:‘tokudb_rpl_check_readonly‘** is enabled (check that slave is read-only). Do **NOT** change this value unless you completely understand the implications!

When disabled, *TokuDB* replication slaves skip row lookups for `delete row` log events and `update row` log events, which eliminates all associated read I/O for these operations.

Warning: *TokuDB Read Free Replication* will not propagate `UPDATE` and `DELETE` events reliably if *TokuDB* table is missing the primary key which will eventually lead to data inconsistency on the slave.

Note: Optimization is only enabled when **:variable:‘read_only‘** is set to `1` and **:variable:‘binlog_format‘** is `ROW`.

This variable allows for simulation of long disk reads by sleeping for the given number of microseconds prior to the row lookup query, it should only be set to a non-zero value for testing.

When disabled, *TokuDB* replication slaves skip uniqueness checks on inserts and updates, which eliminates all associated read I/O for these operations.

Note: Optimization is only enabled when **:variable:‘read_only‘** is set to `1` and **:variable:‘binlog_format‘** is `ROW`.

This variable allows for simulation of long disk reads by sleeping for the given number of microseconds prior to the row lookup query, it should only be set to a non-zero value for testing.

When this variable is set to `ON` during the startup server will check all the status files and remove the embedded `.frm` metadata. This variable can be used to assist in *TokuDB* data recovery. **WARNING:** Use this variable only if you know what you’re doing otherwise it could lead to data loss.

This variable defines whether or not the prepare phase of an XA transaction performs an `fsync()`.

This variable specifies the directory where the *TokuDB* bulk loader stores temporary files. The bulk loader can create large temporary files while it is loading a table, so putting these temporary files on a disk separate from the data directory can be useful.

:variable:‘tokudb_load_save_space‘ determines whether the data is compressed or not. The error message `ERROR 1030 (HY000): Got error 1 from storage engine` could indicate that the disk has run out of space.

For example, it can make sense to use a high-performance disk for the data directory and a very inexpensive disk for the temporary directory. The default location for temporary files is the *MySQL* data directory.

For more information check *TokuDB files and file types* and *TokuDB file management*.

This read-only variable documents the version of the *TokuDB* storage engine.

This variable controls in how many writes the progress is measured to display `SHOW PROCESSLIST`. Writes are defined as `INSERT`, `UPDATE` and `DELETE` queries.

For slow queries, it can be helpful to set this variable and **:variable:‘tokudb_read_status_frequency‘** to 1, and then run `SHOW PROCESSLIST` several times to understand what progress is being made.

71.5 Percona TokuBackup

Percona *TokuBackup* is an open-source hot backup utility for *MySQL* servers running the *TokuDB* storage engine (including *Percona Server for MySQL* and *MariaDB*). It does not lock your database during backup. The *TokuBackup* library intercepts system calls that write files and duplicates the writes to the backup directory.

71.5.1 Installing From Binaries

TokuBackup is included with *Percona Server for MySQL* **:rn:‘5.6.26-74.0‘** and later versions. Installation can be performed with the `ps_tokudb_admin` script.

To install *Percona TokuBackup*:

1. Run `ps_tokudb_admin --enable-backup` to add the `preload-hotbackup` option into **[mysqld_safe]** section of `my.cnf`.

```
$ sudo ps_tokudb_admin --enable-backup
Checking SELinux status...
INFO: SELinux is disabled.

Checking if preload-hotbackup option is already set in config file...
INFO: Option preload-hotbackup is not set in the config file.
```

```
Checking TokuBackup plugin status...
INFO: TokuBackup plugin is not installed.

Adding preload-hotbackup option into /etc/my.cnf
INFO: Successfully added preload-hotbackup option into /etc/my.cnf
PLEASE RESTART MYSQL SERVICE AND RUN THIS SCRIPT AGAIN TO FINISH␣
↪INSTALLATION!
```

2. Restart mysql service

```
$ sudo service mysql restart
```

3. Run `ps_tokudb_admin --enable-backup` again to finish installation of *TokuBackup* plugin

```
$ sudo ps_tokudb_admin --enable-backup
Checking SELinux status...
INFO: SELinux is disabled.

Checking if preload-hotbackup option is already set in config file...
INFO: Option preload-hotbackup is set in the config file.

Checking TokuBackup plugin status...
INFO: TokuBackup plugin is not installed.

Checking if Percona Server is running with libHotBackup.so preloaded...
INFO: Percona Server is running with libHotBackup.so preloaded.

Installing TokuBackup plugin...
INFO: Successfully installed TokuBackup plugin.
```

71.5.2 Making a Backup

To run *Percona TokuBackup*, the backup destination directory must exist, be writable and owned by the same user under which *MySQL* server is running (usually `mysql`) and empty. Once this directory is created, the backup can be run using the following command:

```
mysql> set tokudb_backup_dir='/path_to_empty_directory';
```

Note: Setting the **:variable:‘tokudb_backup_dir‘** variable automatically starts the backup process to the specified directory. Percona TokuBackup will take full backup each time, currently there is no incremental backup option

71.5.3 Restoring From Backup

Percona TokuBackup does not have any functionality for restoring a backup. You can use **`rsync`** or **`cp`** to restore the files. You should check that the restored files have the correct ownership and permissions.

Note: Make sure that the datadir is empty and that *MySQL* server is shut down before restoring from backup. You can’t restore to a datadir of a running mysqld instance (except when importing a partial backup).

The following example shows how you might use the **`rsync`** command to restore the backup:

```
$ rsync -avrP /data/backup/ /var/lib/mysql/
```

Since attributes of files are preserved, in most cases you will need to change their ownership to *mysql* before starting the database server. Otherwise, the files will be owned by the user who created the backup.

```
$ chown -R mysql:mysql /var/lib/mysql
```

If you have changed default *TokuDB* data directory (**:variable:'tokudb_data_dir'**) or *TokuDB* log directory (**:variable:'tokudb_log_dir'**) or both of them, you will see separate folders for each setting in backup directory after taking backup. You'll need to restore each folder separately:

```
$ rsync -avrP /data/backup/mysql_data_dir/ /var/lib/mysql/
$ rsync -avrP /data/backup/tokudb_data_dir/ /path/to/original/tokudb_data_dir/
$ rsync -avrP /data/backup/tokudb_log_dir/ /path/to/original/tokudb_log_dir/
$ chown -R mysql:mysql /var/lib/mysql
$ chown -R mysql:mysql /path/to/original/tokudb_data_dir
$ chown -R mysql:mysql /path/to/original/tokudb_log_dir
```

71.5.4 Advanced Configuration

Monitoring Progress

TokuBackup updates the *PROCESSLIST* state while the backup is in progress. You can see the output by running `SHOW PROCESSLIST` or `SHOW FULL PROCESSLIST`.

Excluding Source Files

You can exclude certain files and directories based on a regular expression set in the **:variable:'tokudb_backup_exclude'** session variable. If the source file name matches the excluded regular expression, then the source file is excluded from backup.

For example, to exclude all `lost+found` directories from backup, use the following command:

```
mysql> SET tokudb_backup_exclude='/lost\\+found($|/)';
```

Throttling Backup Rate

You can throttle the backup rate using the **:variable:‘tokudb_backup_throttle‘** session-level variable. This variable throttles the write rate in bytes per second of the backup to prevent TokuBackup from crowding out other jobs in the system. The default and max value is 18446744073709551615.

```
mysql> SET tokudb_backup_throttle=1000000;
```

Restricting Backup Target

You can restrict the location of the destination directory where the backups can be located using the **:variable:‘tokudb_backup_allowed_prefix‘** system-level variable. Attempts to backup to a location outside of the specified directory or its children will result in an error.

The default is `null`, backups have no restricted locations. This read-only variable can be set in the `my.cnf` configuration file and displayed with the `SHOW VARIABLES` command:

```
mysql> SHOW VARIABLES LIKE 'tokudb_backup_allowed_prefix';
+-------------------------------+-----------+
| Variable_name                 | Value     |
+-------------------------------+-----------+
| tokudb_backup_allowed_prefix | /dumpdir  |
+-------------------------------+-----------+
```

Reporting Errors

Percona TokuBackup uses two variables to capture errors. They are **:variable:‘tokudb_backup_last_error‘** and **:variable:‘tokudb_backup_last_error_string‘**. When *TokuBackup* encounters an error, these will report on the error number and the error string respectively. For example, the following output shows these parameters following an attempted backup to a directory that was not empty:

```
mysql> SET tokudb_backup_dir='/tmp/backupdir';
ERROR 1231 (42000): Variable 'tokudb_backup_dir' can't be set to the value of '/tmp/
↪backupdir'

mysql> SELECT @@tokudb_backup_last_error;
+-----------------------------+
| @@tokudb_backup_last_error |
+-----------------------------+
|                          17 |
+-----------------------------+

mysql> SELECT @@tokudb_backup_last_error_string;
+-------------------------------------------------+
| @@tokudb_backup_last_error_string               |
+-------------------------------------------------+
| tokudb backup couldn't create needed directories. |
+-------------------------------------------------+
```

Create a Backup with a Timestamp

If you plan to store more than one backup in a location, you should add a timestamp to the backup directory name.

A sample Bash script has this information:

```
#!/bin/bash

tm=$(date "+%Y-%m-%d-%H-%M-%S");
backup_dir=$PWD/backup/$tm;
mkdir -p $backup_dir;
bin/mysql -uroot -e "set tokudb_backup_dir='$backup_dir'"
```

Using TokuDB Hot Backup for Replication

TokuDB Hot Backup makes a transactionally consistent copy of the TokuDB files while applications read and write to these files. The TokuDB hot backup library intercepts certain system calls that writes files and duplicates the writes on backup files while copying files to the backup directory. The copied files contain the same content as the original files.

TokuDB Hot Backup also has an API. This API includes the `start capturing` and `stop capturing` commands. The "capturing" command starts the process, when a portion of a file is copied to the backup location, and this portion is changed, these changes are also applied to the backup location.

Replication often uses backup replication to create slaves. You must know the last executed global transaction identifier (GTID) or binary log position both for the slave and master configuration.

To lock tables, use `FLUSH TABLE WITH READ LOCK` or use the smart locks like `LOCK TABLES FOR BACKUP` or `LOCK BINLOG FOR BACKUP`.

During the copy process, the binlog is flushed, and the changes are copied to backup by the "capturing" mechanism. After everything has been copied, and the "capturing" mechanism is still running, use the `LOCK BINLOG FOR BACKUP`. After this statement is executed, the binlog is flushed, the changes are captured, and any queries that could change the binlog position or executed GTID are blocked.

After this command, we can stop capturing and retrieve the last executed GTID or binlog log position and unlock the binlog.

After a backup is taken, there are the following files in the backup directory:

- tokubackup_slave_info
- tokubackup_binlog_info

These files contain information for slave and master. You can use this information to start a new slave from the master or slave.

The `SHOW MASTER STATUS` and `SHOW SLAVE STATUS` commands provide the information.

In specific binlog formats, a binary log event can contain statements that produce temporary tables on the slave side, and the result of further statements may depend on the temporary table content. Typically, temporary tables are not selected for backup because they are created in a separate directory. A backup created with temporary tables created by binlog events can cause issues when restored because the temporary tables are not restored. The data may be inconsistent.

The following system variables **:variable:'–tokudb-backup-safe-slave'**, which enables or disables the safe-slave mode, and **:variable:'–tokudb-backup-safe-slave-timeout'**, which defines the maximum amount of time in seconds to wait until temporary tables disappear. The `safe-slave` mode, when used with `LOCK BINLOG FOR BACKUP`, the slave SQL thread is stopped and checked to see if temporary tables produced by the slave exist or do not exist. If temporary tables exist, the slave SQL thread is restarted until there are no temporary tables or a defined timeout is reached.

You should not use this option for group-replication.

71.5.5 Limitations and known issues

- You must disable *InnoDB* asynchronous IO if backing up *InnoDB* tables with *TokuBackup*. Otherwise you will have inconsistent, unrecoverable backups. The appropriate setting is `innodb_use_native_aio=0`.
- To be able to run Point-In-Time-Recovery you'll need to manually get the binary log position.
- Transactional storage engines (*TokuDB* and *InnoDB*) will perform recovery on the backup copy of the database when it is first started.
- Tables using non-transactional storage engines (*MyISAM*) are not locked during the copy and may report issues when starting up the backup. It is best to avoid operations that modify these tables at the end of a hot backup operation (adding/changing users, stored procedures, etc.).
- The database is copied locally to the path specified in `/path/to/backup`. This folder must exist, be writable, be empty, and contain enough space for a full copy of the database.
- *TokuBackup* always makes a backup of the *MySQL* **:variable:'datadir'** and optionally the **:variable:'tokudb_data_dir'**, **:variable:'tokudb_log_dir'**, and the binary log folder. The latter three are only backed up separately if they are not the same as or contained in the *MySQL* **:variable:'datadir'**. None of these three folders can be a parent of the *MySQL* **:variable:'datadir'**.
- No other directory structures are supported. All *InnoDB*, *MyISAM*, and other storage engine files must be within the *MySQL* **:variable:'datadir'**.
- *TokuBackup* does not follow symbolic links.
- *TokuBackup* does not backup *MySQL* configuration file(s).
- *TokuBackup* does not backup tablespaces if they are out of **:variable:'datadir'**.
- Due to upstream bug #80183, *TokuBackup* can't recover backed-up table data if backup was taken while running `OPTIMIZE TABLE` or `ALTER TABLE ... TABLESPACE`.
- *TokuBackup* doesn't support incremental backups.

71.6 TokuDB Troubleshooting

71.6.1 Known Issues

Replication and binary logging: *TokuDB* supports binary logging and replication, with one restriction. *TokuDB* does not implement a lock on the auto-increment function, so concurrent insert statements with one or more of the statements inserting multiple rows may result in a non-deterministic interleaving of the auto-increment values. When running replication with these concurrent inserts, the auto-increment values on the slave table may not match the auto-increment values on the master table. Note that this is only an issue with Statement Based Replication (SBR), and not Row Based Replication (RBR).

For more information about auto-increment and replication, see the *MySQL* Reference Manual: AUTO_INCREMENT handling in InnoDB.

In addition, when using the `REPLACE INTO` or `INSERT IGNORE` on tables with no secondary indexes or tables where secondary indexes are subsets of the primary, the session variable **:variable:'tokudb_pk_insert_mode'** controls whether row based replication will work.

Uninformative error message: The `LOAD DATA INFILE` command can sometimes produce `ERROR 1030 (HY000): Got error 1 from storage engine`. The message should say that the error is caused by insufficient disk space for the temporary files created by the loader.

Transparent Huge Pages: *TokuDB* will refuse to start if transparent huge pages are enabled. Transparent huge page support can be disabled by issuing the following as root:

```
# echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

Note: The previous command needs to be executed after every reboot, because it defaults to `always`.

XA behavior vs. InnoDB: *InnoDB* forces a deadlocked XA transaction to abort, *TokuDB* does not.

Disabling the unique checks: For tables with unique keys, every insertion into the table causes a lookup by key followed by an insertion, if the key is not in the table. This greatly limits insertion performance. If one knows by design that the rows being inserted into the table have unique keys, then one can disable the key lookup prior to insertion.

If your primary key is an auto-increment key, and none of your secondary keys are declared to be unique, then setting `unique_checks=OFF` will provide limited performance gains. On the other hand, if your primary key has a lot of entropy (it looks random), or your secondary keys are declared unique and have a lot of entropy, then disabling unique checks can provide a significant performance boost.

If **:variable:'unique_checks'** is disabled when the primary key is not unique, secondary indexes may become corrupted. In this case, the indexes should be dropped and rebuilt. This behavior differs from that of *InnoDB*, in which uniqueness is always checked on the primary key, and setting **:variable:'unique_checks'** to off turns off uniqueness checking on secondary indexes only. Turning off uniqueness checking on the primary key can provide large performance boosts, but it should only be done when the primary key is known to be unique.

71.6.2 Lock Visualization in TokuDB

TokuDB uses key range locks to implement serializable transactions, which are acquired as the transaction progresses. The locks are released when the transaction commits or aborts (this implements two phase locking).

TokuDB stores these locks in a data structure called the lock tree. The lock tree stores the set of range locks granted to each transaction. In addition, the lock tree stores the set of locks that are not granted due to a conflict with locks granted to some other transaction. When these other transactions are retired, these pending lock requests are retried. If a pending lock request is not granted before the lock timer expires, then the lock request is aborted.

Lock visualization in *TokuDB* exposes the state of the lock tree with tables in the information schema. We also provide a mechanism that may be used by a database client to retrieve details about lock conflicts that it encountered while executing a transaction.

The `TOKUDB_TRX` table

The **:table:'TOKUDB_TRX'** table in the `INFORMATION_SCHEMA` maps *TokuDB* transaction identifiers to *MySQL* client identifiers. This mapping allows one to associate a *TokuDB* transaction with a *MySQL* client operation.

The following query returns the *MySQL* clients that have a live *TokuDB* transaction:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_TRX,
  INFORMATION_SCHEMA.PROCESSLIST
  WHERE trx_mysql_thread_id = id;
```

The TOKUDB_LOCKS table

The **:table:‘tokudb_locks‘** table in the information schema contains the set of locks granted to *TokuDB* transactions.

The following query returns all of the locks granted to some *TokuDB* transaction:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;
```

The following query returns the locks granted to some *MySQL* client:

```
SELECT id FROM INFORMATION_SCHEMA.TOKUDB_LOCKS,
  INFORMATION_SCHEMA.PROCESSLIST
  WHERE locks_mysql_thread_id = id;
```

The TOKUDB_LOCK_WAITS table

The **:table:‘tokudb_lock_waits‘** table in the information schema contains the set of lock requests that are not granted due to a lock conflict with some other transaction.

The following query returns the locks that are waiting to be granted due to a lock conflict with some other transaction:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCK_WAITS;
```

The :variable:‘tokudb_lock_timeout_debug‘ session variable

The **:variable:‘tokudb_lock_timeout_debug‘** session variable controls how lock timeouts and lock deadlocks seen by the database client are reported.

The following values are available:

0 No lock timeouts or lock deadlocks are reported.

1 A JSON document that describes the lock conflict is stored in the **:variable:‘tokudb_last_lock_timeout‘** session variable

2 A JSON document that describes the lock conflict is printed to the *MySQL* error log.

Supported since 7.5.5: In addition to the JSON document describing the lock conflict, the following lines are printed to the MySQL error log:

- A line containing the blocked thread id and blocked SQL
- A line containing the blocking thread id and the blocking SQL.

3 A JSON document that describes the lock conflict is stored in the **:variable:‘tokudb_last_lock_timeout‘** session variable and is printed to the *MySQL* error log.

Supported since 7.5.5: In addition to the JSON document describing the lock conflict, the following lines are printed to the *MySQL* error log:

- A line containing the blocked thread id and blocked SQL
- A line containing the blocking thread id and the blocking SQL.

The :variable:‘tokudb_last_lock_timeout‘ session variable

The **:variable:‘tokudb_last_lock_timeout‘** session variable contains a JSON document that describes the last lock conflict seen by the current *MySQL* client. It gets set when a blocked lock request times out or a lock deadlock is detected. The **:variable:‘tokudb_lock_timeout_debug‘** session variable should have bit `0` set (decimal `1`).

Example

Suppose that we create a table with a single column that is the primary key.

```
mysql> SHOW CREATE TABLE table;

Create Table: CREATE TABLE `table` (
`id` int(11) NOT NULL,
PRIMARY KEY (`id`)) ENGINE=TokuDB DEFAULT CHARSET=latin1
```

Suppose that we have 2 *MySQL* clients with ID's 1 and 2 respectively. Suppose that *MySQL* client 1 inserts some values into `table`. *TokuDB* transaction 51 is created for the insert statement. Since autocommit is disabled, transaction 51 is still live after the insert statement completes, and we can query the **:table:‘tokudb_locks‘** table in information schema to see the locks that are held by the transaction.

```
mysql> SET AUTOCOMMIT=OFF;
mysql> INSERT INTO table VALUES (1),(10),(100);

Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;

+--------------+-----------------------+---------------+----------------+-------------
↪----+--------------------+------------------+-----------------------------+
| locks_trx_id | locks_mysql_thread_id | locks_dname   | locks_key_left | locks_key_
↪right | locks_table_schema | locks_table_name | locks_table_dictionary_name |
+--------------+-----------------------+---------------+----------------+-------------
↪----+--------------------+------------------+-----------------------------+
|           51 |                     1 | ./test/t-main | 0001000000     | 0001000000 ␣
↪    | test               | t                | main                        |
|           51 |                     1 | ./test/t-main | 000a000000     | 000a000000 ␣
↪    | test               | t                | main                        |
|           51 |                     1 | ./test/t-main | 0064000000     | 0064000000 ␣
↪    | test               | t                | main                        |
+--------------+-----------------------+---------------+----------------+-------------
↪----+--------------------+------------------+-----------------------------+

mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCK_WAITS;

Empty set (0.00 sec)
```

The keys are currently hex dumped.

Now we switch to the other *MySQL* client with ID 2.

```
mysql> INSERT INTO table VALUES (2),(20),(100);
```

The insert gets blocked since there is a conflict on the primary key with value 100.

The granted *TokuDB* locks are:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;

+--------------+----------------------+--------------+----------------+-------------
↪----+--------------------+------------------+-----------------------------+
| locks_trx_id | locks_mysql_thread_id | locks_dname    | locks_key_left | locks_key_
↪right | locks_table_schema | locks_table_name | locks_table_dictionary_name |
+--------------+----------------------+--------------+----------------+-------------
↪----+--------------------+------------------+-----------------------------+
|           51 |                    1 | ./test/t-main | 0001000000     | 0001000000 ␣
↪    | test               | t                | main                        |
|           51 |                    1 | ./test/t-main | 000a000000     | 000a000000 ␣
↪    | test               | t                | main                        |
|           51 |                    1 | ./test/t-main | 0064000000     | 0064000000 ␣
↪    | test               | t                | main                        |
|           51 |                    1 | ./test/t-main | 0002000000     | 0002000000 ␣
↪    | test               | t                | main                        |
|           51 |                    1 | ./test/t-main | 0014000000     | 0014000000 ␣
↪    | test               | t                | main                        |
+--------------+----------------------+--------------+----------------+-------------
↪----+--------------------+------------------+-----------------------------+
```

The locks that are pending due to a conflict are:

```
SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCK_WAITS;

+-------------------+-----------------+------------------+----------------------+------
↪----------------+-----------------------+--------------------+------------------+---
↪-------------------------+
| requesting_trx_id | blocking_trx_id | lock_waits_dname | lock_waits_key_left | lock_
↪waits_key_right | lock_waits_start_time | locks_table_schema | locks_table_name |␣
↪locks_table_dictionary_name |
+-------------------+-----------------+------------------+----------------------+------
↪----------------+-----------------------+--------------------+------------------+---
↪-------------------------+
|                62 |              51 | ./test/t-main    | 0064000000           |␣
↪0064000000           |         1380656990910 | test               | t                ␣
↪ | main                      |
+-------------------+-----------------+------------------+----------------------+------
↪----------------+-----------------------+--------------------+------------------+---
↪-------------------------+
```

Eventually, the lock for client 2 times out, and we can retrieve a JSON document that describes the conflict.

```
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction

mysql> SELECT @@TOKUDB_LAST_LOCK_TIMEOUT;

+-----------------------------------------------------------------------------------------
↪--------------------------+
| @@tokudb_last_lock_timeout
↪                          |
+-----------------------------------------------------------------------------------------
↪--------------------------+
| "mysql_thread_id":2, "dbname":"./test/t-main", "requesting_txnid":62, "blocking_
↪txnid":51, "key":"0064000000" |
+-----------------------------------------------------------------------------------------
↪--------------------------+
```

```
ROLLBACK;
```

Since transaction 62 was rolled back, all of the locks taken by it are released.

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TOKUDB_LOCKS;

+--------------+-----------------------+--------------+----------------+-------------
↪----+--------------------+------------------+-----------------------------+
| locks_trx_id | locks_mysql_thread_id | locks_dname    | locks_key_left | locks_key_
↪right | locks_table_schema | locks_table_name | locks_table_dictionary_name |
+--------------+-----------------------+--------------+----------------+-------------
↪----+--------------------+------------------+-----------------------------+
|           51 |                     1 | ./test/t-main | 0001000000     | 0001000000 ␣
↪    | test               | t                | main                        |
|           51 |                     1 | ./test/t-main | 000a000000     | 000a000000 ␣
↪    | test               | t                | main                        |
|           51 |                     1 | ./test/t-main | 0064000000     | 0064000000 ␣
↪    | test               | t                | main                        |
|           51 |                     2 | ./test/t-main | 0002000000     | 0002000000 ␣
↪    | test               | t                | main                        |
|           51 |                     2 | ./test/t-main | 0014000000     | 0014000000 ␣
↪    | test               | t                | main                        |
+--------------+-----------------------+--------------+----------------+-------------
↪----+--------------------+------------------+-----------------------------+
```

71.6.3 Engine Status

Engine status provides details about the inner workings of *TokuDB* and can be useful in tuning your particular environment. The engine status can be determined by running the following command:

```
SHOW ENGINE tokudb STATUS;
```

The following is a reference of table status statements:

disk free space: This is a gross estimate of how much of your file system is available. Possible displays in this field are:

- More than twice the reserve (“more than 10 percent of total file system space”)
- Less than twice the reserve
- Less than the reserve
- File system is completely full

time of environment creation: This is the time when the *TokuDB* storage engine was first started up. Normally, this is when `mysqld` was initially installed with *TokuDB*. If the environment was upgraded from *TokuDB* 4.x (4.2.0 or later), then this will be displayed as “Dec 31, 1969” on Linux hosts.

time of engine startup: This is the time when the *TokuDB* storage engine started up. Normally, this is when `mysqld` started.

time now: Current date/time on server.

db opens: This is the number of times an individual PerconaFT dictionary file was opened. This is a not a useful value for a regular user to use for any purpose due to layers of open/close caching on top.

db closes: This is the number of times an individual PerconaFT dictionary file was closed. This is a not a useful value for a regular user to use for any purpose due to layers of open/close caching on top.

num open dbs now: This is the number of currently open databases.

max open dbs: This is the maximum number of concurrently opened databases.

period, in ms, that recovery log is automatically fsynced: `fsync()` frequency in milliseconds.

dictionary inserts: This is the total number of rows that have been inserted into all primary and secondary indexes combined, when those inserts have been done with a separate recovery log entry per index. For example, inserting a row into a table with one primary and two secondary indexes will increase this count by three, if the inserts were done with separate recovery log entries.

dictionary inserts fail: This is the number of single-index insert operations that failed.

dictionary deletes: This is the total number of rows that have been deleted from all primary and secondary indexes combined, if those deletes have been done with a separate recovery log entry per index.

dictionary deletes fail: This is the number of single-index delete operations that failed.

dictionary updates: This is the total number of rows that have been updated in all primary and secondary indexes combined, if those updates have been done with a separate recovery log entry per index.

dictionary updates fail: This is the number of single-index update operations that failed.

dictionary broadcast updates: This is the number of broadcast updates that have been successfully performed. A broadcast update is an update that affects all rows in a dictionary.

dictionary broadcast updates fail: This is the number of broadcast updates that have failed.

dictionary multi inserts: This is the total number of rows that have been inserted into all primary and secondary indexes combined, when those inserts have been done with a single recovery log entry for the entire row. (For example, inserting a row into a table with one primary and two secondary indexes will normally increase this count by three).

dictionary multi inserts fail: This is the number of multi-index insert operations that failed.

dictionary multi deletes: This is the total number of rows that have been deleted from all primary and secondary indexes combined, when those deletes have been done with a single recovery log entry for the entire row.

dictionary multi deletes fail: This is the number of multi-index delete operations that failed.

dictionary updates multi: This is the total number of rows that have been updated in all primary and secondary indexes combined, if those updates have been done with a single recovery log entry for the entire row.

dictionary updates fail multi: This is the number of multi-index update operations that failed.

le: max committed xr: This is the maximum number of committed transaction records that were stored on disk in a new or modified row.

le: max provisional xr: This is the maximum number of provisional transaction records that were stored on disk in a new or modified row.

le: expanded: This is the number of times that an expanded memory mechanism was used to store a new or modified row on disk.

le: max memsize: This is the maximum number of bytes that were stored on disk as a new or modified row. This is the maximum uncompressed size of any row stored in *TokuDB* that was created or modified since the server started.

le: size of leafentries before garbage collection (during message application): Total number of bytes of leaf nodes data before performing garbage collection for non-flush events.

le: size of leafentries after garbage collection (during message application): Total number of bytes of leaf nodes data after performing garbage collection for non-flush events.

le: size of leafentries before garbage collection (outside message application): Total number of bytes of leaf nodes data before performing garbage collection for flush events.

le: size of leafentries after garbage collection (outside message application): Total number of bytes of leaf nodes data after performing garbage collection for flush events.

checkpoint: period: This is the interval in seconds between the end of an automatic checkpoint and the beginning of the next automatic checkpoint.

checkpoint: footprint: Where the database is in the checkpoint process.

checkpoint: last checkpoint began: This is the time the last checkpoint began. If a checkpoint is currently in progress, then this time may be later than the time the last checkpoint completed.

Note: If no checkpoint has ever taken place, then this value will be `Dec 31, 1969` on Linux hosts.

---

checkpoint: last complete checkpoint began: This is the time the last complete checkpoint started. Any data that changed after this time will not be captured in the checkpoint.

checkpoint: last complete checkpoint ended: This is the time the last complete checkpoint ended.

checkpoint: time spent during checkpoint (begin and end phases): Time (in seconds) required to complete all checkpoints.

checkpoint: time spent during last checkpoint (begin and end phases): Time (in seconds) required to complete the last checkpoint.

checkpoint: last complete checkpoint LSN: This is the Log Sequence Number of the last complete checkpoint.

checkpoint: checkpoints taken: This is the number of complete checkpoints that have been taken.

checkpoint: checkpoints failed: This is the number of checkpoints that have failed for any reason.

checkpoint: waiters now: This is the current number of threads simultaneously waiting for the checkpoint-safe lock to perform a checkpoint.

checkpoint: waiters max: This is the maximum number of threads ever simultaneously waiting for the checkpoint-safe lock to perform a checkpoint.

checkpoint: non-checkpoint client wait on mo lock: The number of times a non-checkpoint client thread waited for the multi-operation lock.

checkpoint: non-checkpoint client wait on cs lock: The number of times a non-checkpoint client thread waited for the checkpoint-safe lock.

checkpoint: checkpoint begin time: Cumulative time (in microseconds) required to mark all dirty nodes as pending a checkpoint.

checkpoint: long checkpoint begin time: The total time, in microseconds, of long checkpoint begins. A long checkpoint begin is one taking more than 1 second.

checkpoint: long checkpoint begin count: The total number of times a checkpoint begin took more than 1 second.

checkpoint: checkpoint end time: The time spent in checkpoint end operation in seconds.

checkpoint: long checkpoint end time: The time spent in checkpoint end operation in seconds.

checkpoint: long checkpoint end count: This is the count of end_checkpoint operations that exceeded 1 minute.

cachetable: miss: This is a count of how many times the application was unable to access your data in the internal cache.

cachetable: miss time: This is the total time, in microseconds, of how long the database has had to wait for a disk read to complete.

cachetable: prefetches: This is the total number of times that a block of memory has been prefetched into the database's cache. Data is prefetched when the database's algorithms determine that a block of memory is likely to be accessed by the application.

cachetable: size current: This shows how much of the uncompressed data, in bytes, is currently in the database's internal cache.

cachetable: size limit: This shows how much of the uncompressed data, in bytes, will fit in the database's internal cache.

cachetable: size writing This is the number of bytes that are currently queued up to be written to disk.

cachetable: size nonleaf: This shows the amount of memory, in bytes, the current set of non-leaf nodes occupy in the cache.

cachetable: size leaf: This shows the amount of memory, in bytes, the current set of (decompressed) leaf nodes occupy in the cache.

cachetable: size rollback: This shows the rollback nodes size, in bytes, in the cache.

cachetable: size cachepressure: This shows the number of bytes causing cache pressure (the sum of buffers and work done counters), helps to understand if cleaner threads are keeping up with workload. It should really be looked at as more of a value to use in a ratio of cache pressure / cache table size. The closer that ratio evaluates to 1, the higher the cache pressure.

cachetable: size currently cloned data for checkpoint: Amount of memory, in bytes, currently used for cloned nodes. During the checkpoint operation, dirty nodes are cloned prior to serialization/compression, then written to disk. After which, the memory for the cloned block is returned for re-use.

cachetable: evictions: Number of blocks evicted from cache.

cachetable: cleaner executions: Total number of times the cleaner thread loop has executed.

cachetable: cleaner period: *TokuDB* includes a cleaner thread that optimizes indexes in the background. This variable is the time, in seconds, between the completion of a group of cleaner operations and the beginning of the next group of cleaner operations. The cleaner operations run on a background thread performing work that does not need to be done on the client thread.

cachetable: cleaner iterations: This is the number of cleaner operations that are performed every cleaner period.

cachetable: number of waits on cache pressure: The number of times a thread was stalled due to cache pressure.

cachetable: time waiting on cache pressure: Total time, in microseconds, waiting on cache pressure to subside.

cachetable: number of long waits on cache pressure: The number of times a thread was stalled for more than 1 second due to cache pressure.

cachetable: long time waiting on cache pressure: Total time, in microseconds, waiting on cache pressure to subside for more than 1 second.

cachetable: client pool: number of threads in pool: The number of threads in the client thread pool.

cachetable: client pool: number of currently active threads in pool: The number of currently active threads in the client thread pool.

cachetable: client pool: number of currently queued work items: The number of currently queued work items in the client thread pool.

cachetable: client pool: largest number of queued work items: The largest number of queued work items in the client thread pool.

cachetable: client pool: total number of work items processed: The total number of work items processed in the client thread pool.

cachetable: client pool: total execution time of processing work items: The total execution time of processing work items in the client thread pool.

cachetable: cachetable pool: number of threads in pool: The number of threads in the cachetable thread pool.

cachetable: cachetable pool: number of currently active threads in pool: The number of currently active threads in the cachetable thread pool.

cachetable: cachetable pool: number of currently queued work items: The number of currently queued work items in the cachetable thread pool.

cachetable: cachetable pool: largest number of queued work items: The largest number of queued work items in the cachetable thread pool.

cachetable: cachetable pool: total number of work items processed: The total number of work items processed in the cachetable thread pool.

cachetable: cachetable pool: total execution time of processing work items: The total execution time of processing work items in the cachetable thread pool.

cachetable: checkpoint pool: number of threads in pool: The number of threads in the checkpoint thread pool.

cachetable: checkpoint pool: number of currently active threads in pool: The number of currently active threads in the checkpoint thread pool.

cachetable: checkpoint pool: number of currently queued work items: The number of currently queued work items in the checkpoint thread pool.

cachetable: checkpoint pool: largest number of queued work items: The largest number of queued work items in the checkpoint thread pool.

cachetable: checkpoint pool: total number of work items processed: The total number of work items processed in the checkpoint thread pool.

cachetable: checkpoint pool: total execution time of processing work items: The total execution time of processing work items in the checkpoint thread pool.

locktree: memory size: The amount of memory, in bytes, that the locktree is currently using.

locktree: memory size limit: The maximum amount of memory, in bytes, that the locktree is allowed to use.

locktree: number of times lock escalation ran: Number of times the locktree needed to run lock escalation to reduce its memory footprint.

locktree: time spent running escalation (seconds): Total number of seconds spent performing locktree escalation.

locktree: latest post-escalation memory size: Size of the locktree, in bytes, after most current locktree escalation.

locktree: number of locktrees open now: Number of locktrees currently open.

locktree: number of pending lock requests: Number of requests waiting for a lock grant.

locktree: number of locktrees eligible for the STO: Number of locktrees eligible for "Single Transaction Optimizations". `STO` optimization are behaviors that can happen within the locktree when there is exactly one transaction active within the locktree. This is a not a useful value for a regular user to use for any purpose.

locktree: number of times a locktree ended the STO early: Total number of times a "single transaction optimization" was ended early due to another trans- action starting.

locktree: time spent ending the STO early (seconds): Total number of seconds ending "Single Transaction Optimizations". `STO` optimization are behaviors that can happen within the locktree when there is exactly one transaction active within the locktree. This is a not a useful value for a regular user to use for any purpose.

locktree: number of wait locks: Number of times that a lock request could not be acquired because of a conflict with some other transaction.

locktree: time waiting for locks: Total time, in microseconds, spend by some client waiting for a lock conflict to be resolved.

locktree: number of long wait locks: Number of lock waits greater than 1 second in duration.

locktree: long time waiting for locks: Total time, in microseconds, of the long waits.

locktree: number of lock timeouts: Count of the number of times that a lock request timed out.

locktree: number of waits on lock escalation: When the sum of the sizes of locks taken reaches the lock tree limit, we run lock escalation on a background thread. The clients threads need to wait for escalation to consolidate locks and free up memory. This counter counts the number of times a client thread has to wait on lock escalation.

locktree: time waiting on lock escalation: Total time, in microseconds, that a client thread spent waiting for lock escalation to free up memory.

locktree: number of long waits on lock escalation: Number of times that a client thread had to wait on lock escalation and the wait time was greater than 1 second.

locktree: long time waiting on lock escalation: Total time, in microseconds, of the long waits for lock escalation to free up memory.

ft: dictionary updates: This is the total number of rows that have been updated in all primary and secondary indexes combined, if those updates have been done with a separate recovery log entry per index.

ft: dictionary broadcast updates: This is the number of broadcast updates that have been successfully performed. A broadcast update is an update that affects all rows in a dictionary.

ft: descriptor set: This is the number of time a descriptor was updated when the entire dictionary was updated (for example, when the schema has been changed).

ft: messages ignored by leaf due to msn: The number of messages that were ignored by a leaf because it had already been applied.

ft: total search retries due to TRY AGAIN Total number of search retries due to TRY AGAIN. Internal value that is no use to anyone other than a developer debugging a specific query/search issue.

ft: searches requiring more tries than the height of the tree: Number of searches that required more tries than the height of the tree.

ft: searches requiring more tries than the height of the tree plus three Number of searches that required more tries than the height of the tree plus three.

ft: leaf nodes flushed to disk (not for checkpoint): Number of leaf nodes flushed to disk, not for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) (bytes): Number of bytes of leaf nodes flushed to disk, not for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) (uncompressed bytes): Number of bytes of leaf nodes flushed to disk, not for checkpoint.

ft: leaf nodes flushed to disk (not for checkpoint) (seconds): Number of seconds waiting for IO when writing leaf nodes flushed to disk, not for checkpoint.

ft: nonleaf nodes flushed to disk (not for checkpoint): Number of non-leaf nodes flushed to disk, not for checkpoint.

ft: nonleaf nodes flushed to disk (not for checkpoint) (bytes): Number of bytes of non-leaf nodes flushed to disk, not for checkpoint.

ft: nonleaf nodes flushed to disk (not for checkpoint) (uncompressed bytes): Number of uncompressed bytes of non-leaf nodes flushed to disk, not for checkpoint.

ft: nonleaf nodes flushed to disk (not for checkpoint) (seconds): Number of seconds waiting for I/O when writing non-leaf nodes flushed to disk, not for checkpoint.

ft: leaf nodes flushed to disk (for checkpoint): Number of leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (for checkpoint) (bytes): Number of bytes of leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (for checkpoint) (uncompressed bytes): Number of uncompressed bytes of leaf nodes flushed to disk for checkpoint.

ft: leaf nodes flushed to disk (for checkpoint) (seconds) Number of seconds waiting for IO when writing leaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (for checkpoint): Number of non-leaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (for checkpoint) (bytes): Number of bytes of non-leaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (for checkpoint) (uncompressed bytes): Number of uncompressed bytes of non-leaf nodes flushed to disk for checkpoint.

ft: nonleaf nodes flushed to disk (for checkpoint) (seconds): Number of seconds waiting for IO when writing non-leaf nodes flushed to disk for checkpoint.

ft: uncompressed / compressed bytes written (leaf): Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for leaf nodes.

ft: uncompressed / compressed bytes written (nonleaf): Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for non-leaf nodes.

ft: uncompressed / compressed bytes written (overall): Ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for all nodes.

ft: nonleaf node partial evictions: The number of times a partition of a non-leaf node was evicted from the cache.

ft: nonleaf node partial evictions (bytes): The number of bytes freed by evicting partitions of non-leaf nodes from the cache.

ft: leaf node partial evictions: The number of times a partition of a leaf node was evicted from the cache.

ft: leaf node partial evictions (bytes): The number of bytes freed by evicting partitions of leaf nodes from the cache.

ft: leaf node full evictions The number of times a full leaf node was evicted from the cache.

ft: leaf node full evictions (bytes): The number of bytes freed by evicting full leaf nodes from the cache.

ft: nonleaf node full evictions (bytes): The number of bytes freed by evicting full non-leaf nodes from the cache.

ft: nonleaf node full evictions: The number of times a full non-leaf node was evicted from the cache.

ft: leaf nodes created: Number of created leaf nodes .

ft: nonleaf nodes created: Number of created non-leaf nodes.

ft: leaf nodes destroyed: Number of destroyed leaf nodes.

ft: nonleaf nodes destroyed: Number of destroyed non-leaf nodes.

ft: bytes of messages injected at root (all trees): Amount of messages, in bytes, injected at root (for all trees).

ft: bytes of messages flushed from h1 nodes to leaves Amount of messages, in bytes, flushed from `h1` nodes to leaves.

ft: bytes of messages currently in trees (estimate): Amount of messages, in bytes, currently in trees (estimate).

ft: messages injected at root: Number of messages injected at root node of a tree.

ft: broadcast messages injected at root: Number of broadcast messages injected at root node of a tree.

ft: basements decompressed as a target of a query: Number of basement nodes decompressed for queries.

ft: basements decompressed for prelocked range: Number of basement nodes decompressed by queries aggressively.

ft: basements decompressed for prefetch: Number of basement nodes decompressed by a prefetch thread.

ft: basements decompressed for write: Number of basement nodes decompressed for writes.

ft: buffers decompressed as a target of a query: Number of buffers decompressed for queries.

ft: buffers decompressed for prelocked range: Number of buffers decompressed by queries aggressively.

ft: buffers decompressed for prefetch: Number of buffers decompressed by a prefetch thread.

ft: buffers decompressed for write: Number of buffers decompressed for writes.

ft: pivots fetched for query: Number of pivot nodes fetched for queries.

ft: pivots fetched for query (bytes): Number of bytes of pivot nodes fetched for queries.

ft: pivots fetched for query (seconds): Number of seconds waiting for I/O when fetching pivot nodes for queries.

ft: pivots fetched for prefetch: Number of pivot nodes fetched by a prefetch thread.

ft: pivots fetched for prefetch (bytes): Number of bytes of pivot nodes fetched by a prefetch thread.

ft: pivots fetched for prefetch (seconds): Number seconds waiting for I/O when fetching pivot nodes by a prefetch thread.

ft: pivots fetched for write: Number of pivot nodes fetched for writes.

ft: pivots fetched for write (bytes): Number of bytes of pivot nodes fetched for writes.

ft: pivots fetched for write (seconds): Number of seconds waiting for I/O when fetching pivot nodes for writes.

ft: basements fetched as a target of a query: Number of basement nodes fetched from disk for queries.

ft: basements fetched as a target of a query (bytes): Number of basement node bytes fetched from disk for queries.

ft: basements fetched as a target of a query (seconds): Number of seconds waiting for IO when fetching basement nodes from disk for queries.

ft: basements fetched for prelocked range: Number of basement nodes fetched from disk aggressively.

ft: basements fetched for prelocked range (bytes): Number of basement node bytes fetched from disk aggressively.

ft: basements fetched for prelocked range (seconds): Number of seconds waiting for I/O when fetching basement nodes from disk aggressively.

ft: basements fetched for prefetch: Number of basement nodes fetched from disk by a prefetch thread.

ft: basements fetched for prefetch (bytes): Number of basement node bytes fetched from disk by a prefetch thread.

ft: basements fetched for prefetch (seconds): Number of seconds waiting for I/O when fetching basement nodes from disk by a prefetch thread.

ft: basements fetched for write: Number of basement nodes fetched from disk for writes.

ft: basements fetched for write (bytes): Number of basement node bytes fetched from disk for writes.

ft: basements fetched for write (seconds): Number of seconds waiting for I/O when fetching basement nodes from disk for writes.

ft: buffers fetched as a target of a query: Number of buffers fetched from disk for queries.

ft: buffers fetched as a target of a query (bytes): Number of buffer bytes fetched from disk for queries.

ft: buffers fetched as a target of a query (seconds): Number of seconds waiting for I/O when fetching buffers from disk for queries.

ft: buffers fetched for prelocked range: Number of buffers fetched from disk aggressively.

ft: buffers fetched for prelocked range (bytes): Number of buffer bytes fetched from disk aggressively.

ft: buffers fetched for prelocked range (seconds): Number of seconds waiting for I/O when fetching buffers from disk aggressively.

ft: buffers fetched for prefetch: Number of buffers fetched from disk by a prefetch thread.

ft: buffers fetched for prefetch (bytes): Number of buffer bytes fetched from disk by a prefetch thread.

ft: buffers fetched for prefetch (seconds): Number of seconds waiting for I/O when fetching buffers from disk by a prefetch thread.

ft: buffers fetched for write: Number of buffers fetched from disk for writes.

ft: buffers fetched for write (bytes): Number of buffer bytes fetched from disk for writes.

ft: buffers fetched for write (seconds): Number of seconds waiting for I/O when fetching buffers from disk for writes.

ft: leaf compression to memory (seconds): Total time, in seconds, spent compressing leaf nodes.

ft: leaf serialization to memory (seconds): Total time, in seconds, spent serializing leaf nodes.

ft: leaf decompression to memory (seconds): Total time, in seconds, spent decompressing leaf nodes.

ft: leaf deserialization to memory (seconds): Total time, in seconds, spent deserializing leaf nodes.

ft: nonleaf compression to memory (seconds): Total time, in seconds, spent compressing non leaf nodes.

ft: nonleaf serialization to memory (seconds): Total time, in seconds, spent serializing non leaf nodes.

ft: nonleaf decompression to memory (seconds): Total time, in seconds, spent decompressing non leaf nodes.

ft: nonleaf deserialization to memory (seconds): Total time, in seconds, spent deserializing non leaf nodes.

ft: promotion: roots split: Number of times the root split during promotion.

ft: promotion: leaf roots injected into: Number of times a message stopped at a root with height `0`.

ft: promotion: h1 roots injected into: Number of times a message stopped at a root with height `1`.

ft: promotion: injections at depth 0: Number of times a message stopped at depth `0`.

ft: promotion: injections at depth 1: Number of times a message stopped at depth `1`.

ft: promotion: injections at depth 2: Number of times a message stopped at depth `2`.

ft: promotion: injections at depth 3: Number of times a message stopped at depth `3`.

ft: promotion: injections lower than depth 3: Number of times a message was promoted past depth `3`.

ft: promotion: stopped because of a nonempty buffer: Number of times a message stopped because it reached a nonempty buffer.

ft: promotion: stopped at height 1 Number of times a message stopped because it had reached height `1`.

ft: promotion: stopped because the child was locked or not at all in memory: Number of times promotion was stopped because the child node was locked or not at all in memory. This is a not a useful value for a regular user to use for any purpose.

ft: promotion: stopped because the child was not fully in memory: Number of times promotion was stopped because the child node was not at all in memory. This is a not a useful value for a normal user to use for any purpose.

ft: promotion: stopped anyway, after locking the child: Number of times a message stopped before a child which had been locked.

ft: basement nodes deserialized with fixed-keysize: The number of basement nodes deserialized where all keys had the same size, leaving the basement in a format that is optimal for in-memory workloads.

ft: basement nodes deserialized with variable-keysize: The number of basement nodes deserialized where all keys did not have the same size, and thus ineligible for an in-memory optimization.

ft: promotion: succeeded in using the rightmost leaf shortcut: Rightmost insertions used the rightmost-leaf pin path, meaning that the Fractal Tree index detected and properly optimized rightmost inserts.

ft: promotion: tried the rightmost leaf shortcut but failed (out-of-bounds): Rightmost insertions did not use the rightmost-leaf pin path, due to the insert not actually being into the rightmost leaf node.

ft: promotion: tried the rightmost leaf shortcut but failed (child reactive): Rightmost insertions did not use the rightmost-leaf pin path, due to the leaf being too large (needed to split).

ft: cursor skipped deleted leaf entries: Number of leaf entries skipped during search/scan because the result of message application and reconciliation of the leaf entry MVCC stack reveals that the leaf entry is deleted in the current transactions view. It is a good indicator that there might be excessive garbage in a tree if a range scan seems to take too long.

ft flusher: total nodes potentially flushed by cleaner thread: Total number of nodes whose buffers are potentially flushed by cleaner thread.

ft flusher: height-one nodes flushed by cleaner thread: Number of nodes of height one whose message buffers are flushed by cleaner thread.

ft flusher: height-greater-than-one nodes flushed by cleaner thread: Number of nodes of height > 1 whose message buffers are flushed by cleaner thread.

ft flusher: nodes cleaned which had empty buffers: Number of nodes that are selected by cleaner, but whose buffers are empty.

ft flusher: nodes dirtied by cleaner thread: Number of nodes that are made dirty by the cleaner thread.

ft flusher: max bytes in a buffer flushed by cleaner thread: Max number of bytes in message buffer flushed by cleaner thread.

ft flusher: min bytes in a buffer flushed by cleaner thread: Min number of bytes in message buffer flushed by cleaner thread.

ft flusher: total bytes in buffers flushed by cleaner thread: Total number of bytes in message buffers flushed by cleaner thread.

ft flusher: max workdone in a buffer flushed by cleaner thread: Max workdone value of any message buffer flushed by cleaner thread.

ft flusher: min workdone in a buffer flushed by cleaner thread: Min workdone value of any message buffer flushed by cleaner thread.

ft flusher: total workdone in buffers flushed by cleaner thread: Total workdone value of message buffers flushed by cleaner thread.

ft flusher: times cleaner thread tries to merge a leaf: The number of times the cleaner thread tries to merge a leaf.

ft flusher: cleaner thread leaf merges in progress: The number of cleaner thread leaf merges in progress.

ft flusher: cleaner thread leaf merges successful: The number of times the cleaner thread successfully merges a leaf.

ft flusher: nodes dirtied by cleaner thread leaf merges: The number of nodes dirtied by the "flush from root" process to merge a leaf node.

ft flusher: total number of flushes done by flusher threads or cleaner threads: Total number of flushes done by flusher threads or cleaner threads.

ft flusher: number of in memory flushes: Number of in-memory flushes.

ft flusher: number of flushes that read something off disk: Number of flushes that had to read a child (or part) off disk.

ft flusher: number of flushes that triggered another flush in child: Number of flushes that triggered another flush in the child.

ft flusher: number of flushes that triggered 1 cascading flush: Number of flushes that triggered 1 cascading flush.

ft flusher: number of flushes that triggered 2 cascading flushes: Number of flushes that triggered 2 cascading flushes.

ft flusher: number of flushes that triggered 3 cascading flushes: Number of flushes that triggered 3 cascading flushes.

ft flusher: number of flushes that triggered 4 cascading flushes: Number of flushes that triggered 4 cascading flushes.

ft flusher: number of flushes that triggered 5 cascading flushes: Number of flushes that triggered 5 cascading flushes.

ft flusher: number of flushes that triggered over 5 cascading flushes: Number of flushes that triggered more than 5 cascading flushes.

ft flusher: leaf node splits: Number of leaf nodes split.

ft flusher: nonleaf node splits: Number of non-leaf nodes split.

ft flusher: leaf node merges: Number of times leaf nodes are merged.

ft flusher: nonleaf node merges: Number of times non-leaf nodes are merged.

ft flusher: leaf node balances: Number of times a leaf node is balanced.

hot: operations ever started: This variable shows the number of hot operations started (`OPTIMIZE TABLE`). This is a not a useful value for a regular user to use for any purpose.

hot: operations successfully completed: The number of hot operations that have successfully completed (`OPTIMIZE TABLE`). This is a not a useful value for a regular user to use for any purpose.

hot: operations aborted: The number of hot operations that have been aborted (`OPTIMIZE TABLE`). This is a not a useful value for a regular user to use for any purpose.

hot: max number of flushes from root ever required to optimize a tree: The maximum number of flushes from the root ever required to optimize a tree.

txn: begin: This is the number of transactions that have been started.

txn: begin read only: Number of read only transactions started.

txn: successful commits: This is the total number of transactions that have been committed.

txn: aborts: This is the total number of transactions that have been aborted.

logger: next LSN: This is the next unassigned Log Sequence Number. It will be assigned to the next entry in the recovery log.

logger: writes: Number of times the logger has written to disk.

logger: writes (bytes): Number of bytes the logger has written to disk.

logger: writes (uncompressed bytes): Number of uncompressed the logger has written to disk.

logger: writes (seconds): Number of seconds waiting for I/O when writing logs to disk.

logger: number of long logger write operations: Number of times a logger write operation required 100ms or more.

indexer: number of indexers successfully created: This is the number of times one of our internal objects, a indexer, has been created.

indexer: number of calls to toku_indexer_create_indexer() that failed: This is the number of times a indexer was requested but could not be created.

indexer: number of calls to indexer->build() succeeded: This is the total number of times that indexes were created using a indexer.

indexer: number of calls to indexer->build() failed: This is the total number of times that indexes were unable to be created using a indexer

indexer: number of calls to indexer->close() that succeeded: This is the number of indexers that successfully created the requested index(es).

indexer: number of calls to indexer->close() that failed: This is the number of indexers that were unable to create the requested index(es).

indexer: number of calls to indexer->abort(): This is the number of indexers that were aborted.

indexer: number of indexers currently in existence: This is the number of indexers that currently exist.

indexer: max number of indexers that ever existed simultaneously: This is the maximum number of indexers that ever existed simultaneously.

loader: number of loaders successfully created: This is the number of times one of our internal objects, a loader, has been created.

loader: number of calls to toku_loader_create_loader() that failed: This is the number of times a loader was requested but could not be created.

loader: number of calls to loader->put() succeeded: This is the total number of rows that were inserted using a loader.

loader: number of calls to loader->put() failed: This is the total number of rows that were unable to be inserted using a loader.

loader: number of calls to loader->close() that succeeded: This is the number of loaders that successfully created the requested table.

loader: number of calls to loader->close() that failed: This is the number of loaders that were unable to create the requested table.

loader: number of calls to loader->abort(): This is the number of loaders that were aborted.

loader: number of loaders currently in existence: This is the number of loaders that currently exist.

loader: max number of loaders that ever existed simultaneously: This is the maximum number of loaders that ever existed simultaneously.

memory: number of malloc operations: Number of calls to `malloc()`.

memory: number of free operations: Number of calls to `free()`.

memory: number of realloc operations: Number of calls to `realloc()`.

memory: number of malloc operations that failed: Number of failed calls to `malloc()`.

memory: number of realloc operations that failed: Number of failed calls to `realloc()`.

memory: number of bytes requested: Total number of bytes requested from memory allocator library.

memory: number of bytes freed: Total number of bytes allocated from memory allocation library that have been freed (used - freed = bytes in use).

memory: largest attempted allocation size: Largest number of bytes in a single successful `malloc()` operation.

memory: size of the last failed allocation attempt: Largest number of bytes in a single failed `malloc()` operation.

memory: number of bytes used (requested + overhead): Total number of bytes allocated by memory allocator library.

memory: estimated maximum memory footprint: Maximum memory footprint of the storage engine, the max value of (used - freed).

memory: mallocator version: Version string from in-use memory allocator.

memory: mmap threshold: The threshold for malloc to use mmap.

filesystem: ENOSPC redzone state: The state of how much disk space exists with respect to the red zone value. Redzone is space greater than **:variable:‘tokudb_fs_reserve_percent‘** and less than full disk.

Valid values are:

0 Space is available

1 Warning, with 2x of redzone value. Operations are allowed, but engine status prints a warning.

2 In red zone, insert operations are blocked

3 All operations are blocked

filesystem: threads currently blocked by full disk: This is the number of threads that are currently blocked because they are attempting to write to a full disk. This is normally zero. If this value is non-zero, then a warning will appear in the “disk free space” field.

filesystem: number of operations rejected by enospc prevention (red zone): This is the number of database inserts that have been rejected because the amount of disk free space was less than the reserve.

filesystem: most recent disk full: This is the most recent time when the disk file system was entirely full. If the disk has never been full, then this value will be `Dec 31, 1969` on Linux hosts.

filesystem: number of write operations that returned ENOSPC: This is the number of times that an attempt to write to disk failed because the disk was full. If the disk is full, this number will continue increasing until space is available.

filesystem: fsync time: This the total time, in microseconds, used to fsync to disk.

filesystem: fsync count: This is the total number of times the database has flushed the operating system’s file buffers to disk.

filesystem: long fsync time: This the total time, in microseconds, used to fsync to disk when the operation required more than 1 second.

filesystem: long fsync count: This is the total number of times the database has flushed the operating system's file buffers to disk and this operation required more than 1 second.

context: tree traversals blocked by a full fetch: Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a full fetch.

context: tree traversals blocked by a partial fetch: Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a partial fetch.

context: tree traversals blocked by a full eviction Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a full eviction.

context: tree traversals blocked by a partial eviction Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a partial eviction.

context: tree traversals blocked by a message injection: Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of message injection.

context: tree traversals blocked by a message application Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of message application (applying fresh ancestors messages to a basement node).

context: tree traversals blocked by a flush: Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a buffer flush from parent to child.

context: tree traversals blocked by a the cleaner thread: Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of a cleaner thread.

context: tree traversals blocked by something uninstrumented: Number of times node `rwlock` contention was observed while pinning nodes from root to leaf because of something uninstrumented.

context: promotion blocked by a full fetch (should never happen): Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a full fetch.

context: promotion blocked by a partial fetch (should never happen): Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a partial fetch.

context: promotion blocked by a full eviction (should never happen): Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a full eviction.

context: promotion blocked by a partial eviction (should never happen): Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a partial eviction.

context: promotion blocked by a message injection: Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of message injection.

context: promotion blocked by a message application: Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of message application (applying fresh ancestors messages to a basement node).

context: promotion blocked by a flush: Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a buffer flush from parent to child.

context: promotion blocked by the cleaner thread: Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of a cleaner thread.

context: promotion blocked by something uninstrumented: Number of times node `rwlock` contention was observed within promotion (pinning nodes from root to the buffer to receive the message) because of something uninstrumented.

context: something uninstrumented blocked by something uninstrumented: Number of times node `rwlock` contention was observed for an uninstrumented process because of something uninstrumented.

handlerton: primary key bytes inserted: Total number of bytes inserted into all primary key indexes.

71.7 Frequently Asked Questions

This section contains frequently asked questions regarding *TokuDB* and related software.

71.7.1 Transactional Operations

What transactional operations does TokuDB support?

TokuDB supports `BEGIN TRANSACTION`, `END TRANSACTION`, `COMMIT`, `ROLLBACK`, `SAVEPOINT`, and `RELEASE SAVEPOINT`.

71.7.2 TokuDB and the File System

How can I determine which files belong to the various tables and indexes in my schemas?

The **:table:'tokudb_file_map'** plugin lists all Fractal Tree Indexes and their corresponding data files. The `internal_file_name` is the actual file name (in the data folder).

```
mysql> SELECT * FROM information_schema.tokudb_file_map;

+-------------------------+-----------------------------------------+---------------+--
↪-----------+------------------------+
| dictionary_name         | internal_file_name                      | table_schema  |␣
↪table_name  | table_dictionary_name  |
+-------------------------+-----------------------------------------+---------------+--
↪-----------+------------------------+
| ./test/tmc-key-idx_col2  | ./_test_tmc_key_idx_col2_a_14.tokudb  | test          |␣
↪tmc         | key_idx_col2           |
| ./test/tmc-main          | ./_test_tmc_main_9_14.tokudb          | test          |␣
↪tmc         | main                   |
| ./test/tmc-status        | ./_test_tmc_status_8_14.tokudb        | test          |␣
↪tmc         | status                 |
+-------------------------+-----------------------------------------+---------------+--
↪-----------+------------------------+
```

71.7.3 Full Disks

What happens when the disk system fills up?

The disk system may fill up during bulk load operations, such as `LOAD DATA IN FILE` or `CREATE INDEX`, or during incremental operations like `INSERT`.

In the bulk case, running out of disk space will cause the statement to fail with `ERROR 1030 (HY000): Got error 1 from storage engine`. The temporary space used by the bulk loader will be released. If this happens, you can use a separate physical disk for the temporary files (for more information, see **:variable:'tokudb_tmp_dir'**). If server runs out of free space *TokuDB* will assert the server to prevent data corruption to existing data files.

Otherwise, disk space can run low during non-bulk operations. When available space is below a user-configurable reserve (5% by default) inserts are prevented and transactions that perform inserts are aborted. If the disk becomes completely full then *TokuDB* will freeze until some disk space is made available.

Details about the disk system:

- There is a free-space reserve requirement, which is a user-configurable parameter given as a percentage of the total space in the file system. The default reserve is five percent. This value is available in the global variable **:variable:'tokudb_fs_reserve_percent'**. We recommend that this reserve be at least half the size of your physical memory.

  TokuDB polls the file system every five seconds to determine how much free space is available. If the free space dips below the reserve, then further table inserts are prohibited. Any transaction that attempts to insert rows will be aborted. Inserts are re-enabled when twice the reserve is available in the file system (so freeing a small amount of disk storage will not be sufficient to resume inserts). Warning messages are sent to the system error log when free space dips below twice the reserve and again when free space dips below the reserve.

  Even with inserts prohibited it is still possible for the file system to become completely full. For example this can happen because another storage engine or another application consumes disk space.

- If the file system becomes completely full, then *TokuDB* will freeze. It will not crash, but it will not respond to most SQL commands until some disk space is made available. When *TokuDB* is frozen in this state, it will still respond to the following command:

```
SHOW ENGINE TokuDB STATUS;
```

  Make disk space available will allow the storage engine to continue running, but inserts will still be prohibited until twice the reserve is free.

  Note: Engine status displays a field indicating if disk free space is above twice the reserve, below twice the reserve, or below the reserve. It will also display a special warning if the disk is completely full.

- In order to make space available on this system you can:
  - Add some disk space to the filesystem.
  - Delete some non-TokuDB files manually.
  - If the disk is not completely full, you may be able to reclaim space by aborting any transactions that are very old. Old transactions can consume large volumes of disk space in the recovery log.
  - If the disk is not completely full, you can drop indexes or drop tables from your *TokuDB* databases.
  - Deleting large numbers of rows from an existing table and then closing the table may free some space, but it may not. Deleting rows may simply leave unused space (available for new inserts) inside *TokuDB* data files rather than shrink the files (internal fragmentation).

The fine print:

- The *TokuDB* storage engine can use up to three separate file systems simultaneously, one each for the data, the recovery log, and the error log. All three are monitored, and if any one of the three falls below the relevant threshold then a warning message will be issued and inserts may be prohibited.
- Warning messages to the error log are not repeated unless available disk space has been above the relevant threshold for at least one minute. This prevents excess messages in the error log if the disk free space is fluctuating around the limit.
- Even if there are no other storage engines or other applications running, it is still possible for *TokuDB* to consume more disk space when operations such as row delete and query are performed, or when checkpoints are taken. This can happen because *TokuDB* can write cached information when it is time-efficient rather than when inserts are issued by the application, because operations in addition to insert (such as delete) create log entries, and also because of internal fragmentation of *TokuDB* data files.
- The **:variable:‘tokudb_fs_reserve_percent‘** variable can not be changed once the system has started. It can only be set in `my.cnf` or on the mysqld command line.

71.7.4 Backup

How do I back up a system with TokuDB tables?

Taking backups with Percona TokuBackup

TokuDB is capable of performing online backups with *Percona TokuBackup*. To perform a backup, execute `backup to '/path/to/backup';`. This will create backup of the server and return when complete. The backup can be used by another server using a copy of the binaries on the source server. You can view the progress of the backup by executing `SHOW PROCESSLIST;`. *TokuBackup* produces a copy of your running *MySQL* server that is consistent at

the end time of the backup process. The thread copying files from source to destination can be throttled by setting the **:variable:'tokudb_backup_throttle'** server variable. For more information check *Percona TokuBackup*.

The following conditions apply:

- Currently, *TokuBackup* only supports tables using the *TokuDB* storage engine and the *MyISAM* tables in the `mysql` database.

> **Warning:** You must disable *InnoDB* asynchronous IO if backing up *InnoDB* tables via *TokuBackup* utility. Otherwise you will have inconsistent, unrecoverable backups. The appropriate setting is **:variable:'innodb_use_native_aio'** to `0`.

- Transactional storage engines (*TokuDB* and *InnoDB*) will perform recovery on the backup copy of the database when it is first started.
- Tables using non-transactional storage engines (*MyISAM*) are not locked during the copy and may report issues when starting up the backup. It is best to avoid operations that modify these tables at the end of a hot backup operation (adding/changing users, stored procedures, etc.).
- The database is copied locally to the path specified in `/path/to/backup`. This folder must exist, be writable, be empty, and contain enough space for a full copy of the database.
- *TokuBackup* always makes a backup of the *MySQL* `datadir` and optionally the **:variable:'tokudb_data_dir'**, **:variable:'tokudb_log_dir'**, and the binary log folder. The latter three are only backed up separately if they are not the same as or contained in the *MySQL* `datadir`. None of these three folders can be a parent of the *MySQL* `datadir`.
- A folder is created in the given backup destination for each of the source folders.
- No other directory structures are supported. All *InnoDB*, *MyISAM*, and other storage engine files must be within the *MySQL* `datadir`.
- *TokuBackup* does not follow symbolic links.

Other options for taking backups

TokuDB tables are represented in the file system with dictionary files, log files, and metadata files. A consistent copy of all of these files must be made during a backup. Copying the files while they may be modified by a running *MySQL* may result in an inconsistent copy of the database.

LVM snapshots may be used to get a consistent snapshot of all of the *TokuDB* files. The LVM snapshot may then be backed up at leisure.

The `SELECT INTO OUTFILE` statement or **`mysqldump`** application may also be used to get a logical backup of the database.

References

The MySQL 5.5 reference manual describes several backup methods and strategies. In addition, we recommend reading the backup and recovery chapter in the following book:

High Performance MySQL, 3rd Edition, by Baron Schwartz, Peter Zaitsev, and Vadim Tkachenko, Copyright 2012, O'Reilly Media.

Cold Backup

When *MySQL* is shut down, a copy of the *MySQL* data directory, the *TokuDB* data directory, and the *TokuDB* log directory can be made. In the simplest configuration, the *TokuDB* files are stored in the *MySQL* data directory with all of other *MySQL* files. One merely has to back up this directory.

Hot Backup using mylvmbackup

The **mylvmbackup** utility, located on Launchpad, works with *TokuDB*. It does all of the magic required to get consistent copies of all of the *MySQL* tables, including *MyISAM* tables, *InnoDB* tables, etc., creates the LVM snapshots, and backs up the snapshots.

Logical Snapshots

A logical snapshot of the databases uses a SQL statements to retrieve table rows and restore them. When used within a transaction, a consistent snapshot of the database can be taken. This method can be used to export tables from one database server and import them into another server.

The `SELECT INTO OUTFILE` statement is used to take a logical snapshot of a database. The `LOAD DATA INFILE` statement is used to load the table data. Please see the *MySQL* 5.6 reference manual for details.

Note: Please do not use the :program‘mysqlhotcopy‘ to back up *TokuDB* tables. This script is incompatible with *TokuDB*.

71.7.5 Missing Log Files

What do I do if I delete my logs files or they are otherwise missing?

You’ll need to recover from a backup. It is essential that the log files be present in order to restart the database.

71.7.6 Isolation Levels

What is the default isolation level for TokuDB?

It is repeatable-read (MVCC).

How can I change the isolation level?

TokuDB supports repeatable-read, serializable, read-uncommitted and read-committed isolation levels (other levels are not supported). *TokuDB* employs pessimistic locking, and aborts a transaction when a lock conflict is detected.

To guarantee that lock conflicts do not occur, use repeatable-read, read-uncommitted or read- committed isolation level.

71.7.7 Lock Wait Timeout Exceeded

Why do my |MySQL| clients get lock timeout errors for my update queries? And what should my application do when it gets these errors?

Updates can get lock timeouts if some other transaction is holding a lock on the rows being updated for longer than the *TokuDB* lock timeout. You may want to increase the this timeout.

If an update deadlocks, then the transaction should abort and retry.

For more information on diagnosing locking issues, see *Lock Visualization in TokuDB*.

71.7.8 Query Cache

Does TokuDB support the query cache?

Yes, you can enable the query cache in the `my.cnf` file. Please make sure that the size of the cache is set to something larger than `0`, as this, in effect, disables the cache.

71.7.9 Row Size

What is the maximum row size?

The maximum row size is 32 MiB.

71.7.10 NFS & CIFS

Can the data directories reside on a disk that is NFS or CIFS mounted?

Yes, we do have customers in production with NFS & CIFS volumes today. However, both of these disk types can pose a challenge to performance and data integrity due to their complexity. If you're seeking performance, the switching infrastructure and protocols of a traditional network were not conceptualized for low response times and can be very difficult to troubleshoot. If you're concerned with data integrity, the possible data caching at the NFS level can cause inconsistencies between the logs and data files that may never be detected in the event of a crash. If you are thinking of using a NFS or CIFS mount, we would recommend that you use synchronous mount options, which are available from the NFS mount man page, but these settings may decrease performance. For further discussion please look here.

71.7.11 Using Other Storage Engines

Can the MyISAM and InnoDB Storage Engines be used?

MyISAM and *InnoDB* can be used directly in conjunction with *TokuDB*. Please note that you should not overcommit memory between *InnoDB* and *TokuDB*. The total memory assigned to both caches must be less than physical memory.

Can the Federated Storage Engines be used?

The Federated Storage Engine can also be used, however it is disabled by default in *MySQL*. It can be enabled by either running mysqld with `--federated` as a command line parameter, or by putting `federated` in the `[mysqld]` section of the `my.cnf` file.

For more information see the *MySQL* 5.6 Reference Manual: FEDERATED Storage Engine.

71.7.12 Using MySQL Patches with TokuDB

Can I use MySQL source code patches with TokuDB?

Yes, but you need to apply Percona patches as well as your patches to *MySQL* to build a binary that works with the Percona Fractal Tree library.

71.7.13 Truncate Table vs Delete from Table

Which is faster, TRUNCATE TABLE or DELETE FROM TABLE?

Please use `TRUNCATE TABLE` whenever possible. A table truncation runs in constant time, whereas a `DELETE FROM TABLE` requires a row-by-row deletion and thus runs in time linear to the table size.

71.7.14 Foreign Keys

Does TokuDB enforce foreign key constraints?

No, *TokuDB* ignores foreign key declarations.

71.7.15 Dropping Indexes

Is dropping an index in TokuDB hot?

No, the table is locked for the amount of time it takes the file system to delete the file associated with the index.

71.8 Removing TokuDB storage engine

In case you want remove the TokuDB storage engine from *Percona Server for MySQL* without causing any errors following is the recommended procedure:

71.8.1 Change the tables from TokuDB to InnoDB

If you still need the data in the TokuDB tables you'll need to alter the tables to other supported storage engine i.e., *InnoDB*:

```
mysql> ALTER TABLE City ENGINE=InnoDB;
```

Note: In case you remove the TokuDB storage engine before you've changed your tables to other supported storage engine you won't be able to access that data without re-installing the TokuDB storage engine.

71.8.2 Removing the plugins

If you're using *Percona Server for MySQL* **:rn:'5.6.22-72.0'** or later you can use the `ps_tokudb_admin` script to remove the plugins:

```
ps_tokudb_admin --disable -uroot -pPassw0rd
```

Script output should look like this:

```
Checking if Percona server is running with jemalloc enabled...
>> Percona server is running with jemalloc enabled.

Checking transparent huge pages status on the system...
>> Transparent huge pages are currently disabled on the system.
```

```
Checking if thp-setting=never option is already set in config file...
>> Option thp-setting=never is set in the config file.

Checking TokuDB plugin status...
>> TokuDB plugin is installed.

Removing thp-setting=never option from /etc/mysql/my.cnf
>> Successfuly removed thp-setting=never option from /etc/mysql/my.cnf

Uninstalling TokuDB plugin...
>> Successfuly uninstalled TokuDB plugin.
```

Prior to *Percona Server for MySQL* **:rn:'5.6.22-72.0'** TokuDB storage engine requires manual removal:

```
UNINSTALL PLUGIN tokudb;
UNINSTALL PLUGIN tokudb_file_map;
UNINSTALL PLUGIN tokudb_fractal_tree_info;
UNINSTALL PLUGIN tokudb_fractal_tree_block_map;
UNINSTALL PLUGIN tokudb_trx;
UNINSTALL PLUGIN tokudb_locks;
UNINSTALL PLUGIN tokudb_lock_waits;
```

After the engine and the plugins have been uninstalled you can remove the TokuDB package by using the apt/yum commands:

```
[root@centos ~]# yum remove Percona-Server-tokudb-56.x86_64
```

or

```
root@wheezy:~# apt-get remove percona-server-tokudb-5.6
```

Note: Make sure you've removed all the TokuDB specific variables from your configuration file (`my.cnf`) before you restart the server, otherwise server could show errors or warnings and won't be able to start.

71.9 Getting the Most from TokuDB

Compression: *TokuDB* compresses all data on disk, including indexes. Compression lowers cost by reducing the amount of storage required and frees up disk space for additional indexes to achieve improved query performance. Depending on the compressibility of the data, we have seen compression ratios up to 25x for high compression. Compression can also lead to improved performance since less data needs to be read from and written to disk.

Fast Insertions and Deletions: TokuDB's Fractal Tree technology enables fast indexed insertions and deletions. Fractal Trees match B-trees in their indexing sweet spot (sequential data) and are up to two orders of magnitude faster for random data with high cardinality.

Eliminates Slave Lag: *TokuDB* replication slaves can be configured to process the replication stream with virtually no read IO. Uniqueness checking is performed on the *TokuDB* master and can be skipped on all *TokuDB* slaves. Also, row based replication ensures that all before and after row images are captured in the binary logs, so the *TokuDB* slaves can harness the power of Fractal Tree indexes and bypass traditional read-modify-write behavior. This "Read Free Replication" ensures that replication slaves do not fall behind the master and can be used for read scaling, backups, and disaster recovery, without sharding, expensive hardware, or limits on what can be replicated.

Hot Index Creation: *TokuDB* allows the addition of indexes to an existing table while inserts and queries are being performed on that table. This means that *MySQL* can be run continuously with no blocking of queries or insertions while indexes are added and eliminates the down-time that index changes would otherwise require.

Hot Column Addition, Deletion, Expansion and Rename: *TokuDB* allows the addition of new columns to an existing table, the deletion of existing columns from an existing table, the expansion of `char`, `varchar`, `varbinary`, and `integer` type columns in an existing table, and the renaming of an existing column while inserts and queries are being performed on that table.

Online (Hot) Backup: The *TokuDB* can create backups of online database servers without downtime.

Fast Indexing: In practice, slow indexing often leads users to choose a smaller number of sub-optimal indexes in order to keep up with incoming data rates. These sub-optimal indexes result in disproportionately slower queries, since the difference in speed between a query with an index and the same query when no index is available can be many orders of magnitude. Thus, fast indexing means fast queries.

Clustering Keys and Other Indexing Improvements: *TokuDB* tables are clustered on the primary key. *TokuDB* also supports clustering secondary keys, providing better performance on a broader range of queries. A clustering key includes (or clusters) all of the columns in a table along with the key. As a result, one can efficiently retrieve any column when doing a range query on a clustering key. Also, with *TokuDB*, an auto-increment column can be used in any index and in any position within an index. Lastly, *TokuDB* indexes can include up to 32 columns.

Less Aging/Fragmentation: *TokuDB* can run much longer, likely indefinitely, without the need to perform the customary practice of dump/reload or `OPTIMIZE TABLE` to restore database performance. The key is the fundamental difference with which the Fractal Tree stores data on disk. Since, by default, the Fractal Tree will store data in 4MB chunks (pre-compression), as compared to InnoDB's 16KB, *TokuDB* has the ability to avoid "database disorder" up to 250x better than InnoDB.

Bulk Loader: *TokuDB* uses a parallel loader to create tables and offline indexes. This parallel loader will use multiple cores for fast offline table and index creation.

Full-Featured Database: *TokuDB* supports fully ACID-compliant transactions, MVCC (Multi-Version Concurrency Control), serialized isolation levels, row-level locking, and XA. *TokuDB* scales with high number of client connections, even for large tables.

Lock Diagnostics: *TokuDB* provides users with the tools to diagnose locking and deadlock issues. For more information, see *Lock Visualization in TokuDB*.

Progress Tracking: Running `SHOW PROCESSLIST` when adding indexes provides status on how many rows have been processed. Running `SHOW PROCESSLIST` also shows progress on queries, as well as insertions, deletions and updates. This information is helpful for estimating how long operations will take to complete.

Fast Recovery: *TokuDB* supports very fast recovery, typically less than a minute.

CHAPTER

SEVENTYTWO

TOKUDB FILES AND FILE TYPES

The *TokuDB* file set consists of many different files that all serve various purposes.

If you have any *TokuDB* data your data directory should look similar to this:

```
root@server:/var/lib/mysql# ls -lah
...
-rw-rw----  1 mysql mysql  76M Oct 13 18:45 ibdata1
...
-rw-rw----  1 mysql mysql  16K Oct 13 15:52 tokudb.directory
-rw-rw----  1 mysql mysql  16K Oct 13 15:52 tokudb.environment
-rw-------  1 mysql mysql    0 Oct 13 15:52 __tokudb_lock_dont_delete_me_data
-rw-------  1 mysql mysql    0 Oct 13 15:52 __tokudb_lock_dont_delete_me_environment
-rw-------  1 mysql mysql    0 Oct 13 15:52 __tokudb_lock_dont_delete_me_logs
-rw-------  1 mysql mysql    0 Oct 13 15:52 __tokudb_lock_dont_delete_me_recovery
-rw-------  1 mysql mysql    0 Oct 13 15:52 __tokudb_lock_dont_delete_me_temp
-rw-rw----  1 mysql mysql  16K Oct 13 15:52 tokudb.rollback
...
```

This document lists the different types of *TokuDB* and *Percona Fractal Tree* files, explains their purpose, shows their location and how to move them around.

72.1 tokudb.environment

This file is the root of the *Percona FT* file set and contains various bits of metadata about the system, such as creation times, current file format versions, etc.

Percona FT will create/expect this file in the directory specified by the *MySQL* datadir.

72.2 tokudb.rollback

Every transaction within *Percona FT* maintains its own transaction rollback log. These logs are stored together within a single *Percona FT* dictionary file and take up space within the *Percona FT* cachetable (just like any other *Percona FT* dictionary).

The transaction rollback logs will `undo` any changes made by a transaction if the transaction is explicitly rolled back, or rolled back via recovery as a result of an uncommitted transaction when a crash occurs.

Percona FT will create/expect this file in the directory specified by the *MySQL* datadir.

72.3 tokudb.directory

Percona FT maintains a mapping of a dictionary name (example: `sbtest.sbtest1.main`) to an internal file name (example: `_sbtest_sbtest1_main_xx_x_xx.tokudb`). This mapping is stored within this single *Percona FT* dictionary file and takes up space within the *Percona FT* cachetable just like any other *Percona FT* dictionary.

Percona FT will create/expect this file in the directory specified by the *MySQL* datadir.

72.4 Dictionary files

TokuDB dictionary (data) files store actual user data. For each *MySQL* table there will be:

- One `status` dictionary that contains metadata about the table.
- One `main` dictionary that stores the full primary key (an imaginary key is used if one was not explicitly specified) and full row data.
- One `key` dictionary for each additional key/index on the table.

These are typically named: `_<database>_<table>_<key>_<internal_txn_id>.tokudb`

Percona FT creates/expects these files in the directory specified by **:variable:‘tokudb_data_dir‘** if set, otherwise the *MySQL* `datadir` is used.

72.5 Recovery log files

The *Percona FT* recovery log records every operation that modifies a *Percona FT* dictionary. Periodically, the system will take a snapshot of the system called a checkpoint. This checkpoint ensures that the modifications recorded within the *Percona FT* recovery logs have been applied to the appropriate dictionary files up to a known point in time and synced to disk.

These files have a rolling naming convention, but use: `log<log_file_number>.tokulog<log_file_format_version>`.

Percona FT creates/expects these files in the directory specified by **:variable:‘tokudb_log_dir‘** if set, otherwise the *MySQL* datadir is used.

Percona FT does not track what log files should or shouldn't be present. Upon startup, it discovers the logs in the log directory, and replays them in order. If the wrong logs are present, the recovery aborts and possibly damages the dictionaries.

72.6 Temporary files

Percona FT might need to create some temporary files in order to perform some operations. When the bulk loader is active, these temporary files might grow to be quite large.

As different operations start and finish, the files will come and go.

There are no temporary files left behind upon a clean shutdown,

Percona FT creates/expects these files in the directory specified by **:variable:‘tokudb_tmp_dir‘** if set. If not, the **:variable:‘tokudb_data_dir‘** is used if set, otherwise the *MySQL* datadir is used.

72.7 Lock files

Percona FT uses lock files to prevent multiple processes from accessing and writing to the files in the assorted *Percona FT* functionality areas. Each lock file will be in the same directory as the file(s) that it is protecting.

These empty files are only used as semaphores across processes. They are safe to delete/ignore as long as no server instances are currently running and using the data set.

`__tokudb_lock_dont_delete_me_environment`

`__tokudb_lock_dont_delete_me_recovery`

`__tokudb_lock_dont_delete_me_logs`

`__tokudb_lock_dont_delete_me_data`

`__tokudb_lock_dont_delete_me_temp`

Percona FT is extremely pedantic about validating its data set. If a file goes missing or unfound, or seems to contain some nonsensical data, it will assert, abort or fail to start. It does this not to annoy you, but to try to protect you from doing any further damage to your data.

CHAPTER

SEVENTYTHREE

TOKUDB FILE MANAGEMENT

As mentioned in the *TokuDB files and file types* *Percona FT* is extremely pedantic about validating its data set. If a file goes missing or can't be accessed, or seems to contain some nonsensical data, it will assert, abort or fail to start. It does this not to annoy you, but to try to protect you from doing any further damage to your data.

This document contains examples of common file maintenance operations and instructions on how to safely execute these operations.

Beginning in Percona Server **:rn:'5.6.33-79.0'** a new server option was introduced called **:variable:'tokudb_dir_per_db'**. This feature addressed two shortcomings the *renaming of data files* on table/index rename, and the ability to *group data files together* within a directory that represents a single database. This feature is disabled by default.

In *Percona Server for MySQL* **:rn:'5.6.36-82.0'** new **:variable:'tokudb_dir_cmd'** variable has been implemented that can be used to edit *TokuDB* files.

73.1 Moving TokuDB data files to a location outside of the default MySQL datadir

TokuDB uses the location specified by the **:variable:'tokudb_data_dir'** variable for all of its data files. If the **:variable:'tokudb_data_dir'** variable is not explicitly set, *TokuDB* will use the location specified by the servers datadir for these files.

The *TokuDB* data files are protected from concurrent process access by the `__tokudb_lock_dont_delete_me_data` file that is located in the same directory as the *TokuDB* data files.

TokuDB data files may be moved to other locations with symlinks left behind in their place. If those symlinks refer to files on other physical data volumes, the **:variable:'tokudb_fs_reserve_percent'** monitor will not traverse the symlink and monitor the real location for adequate space in the file system.

To safely move your TokuDB data files:

1. Shut the server down cleanly.
2. Change the **:variable:'tokudb_data_dir'** in your `my.cnf` configuration file to the location where you wish to store your *TokuDB* data files.
3. Create your new target directory.
4. Move your `*.tokudb` files and your `__tokudb_lock_dont_delete_me_data` from the current location to the new location.
5. Restart your server.

73.2 Moving TokuDB temporary files to a location outside of the default MySQL datadir

TokuDB will use the location specified by the **:variable:‘tokudb_tmp_dir‘** variable for all of its temporary files. If **:variable:‘tokudb_tmp_dir‘** variable is not explicitly set, *TokuDB* will use the location specified by the **:variable:‘tokudb_data_dir‘** variable. If the **:variable:‘tokudb_data_dir‘** variable is also not explicitly set, *TokuDB* will use the location specified by the servers datadir for these files.

TokuDB temporary files are protected from concurrent process access by the `__tokudb_lock_dont_delete_me_temp` file that is located in the same directory as the *TokuDB* temporary files.

If you locate your *TokuDB* temporary files on a physical volume that is different from where your *TokuDB* data files or recovery log files are located, the **:variable:‘tokudb_fs_reserve_percent‘** monitor will not monitor their location for adequate space in the file system.

To safely move your *TokuDB* temporary files:

1. Shut the server down cleanly. A clean shutdown will ensure that there are no temporary files that need to be relocated.
2. Change the **:variable:‘tokudb_tmp_dir‘** variable in your `my.cnf` configuration file to the location where you wish to store your new *TokuDB* temporary files.
3. Create your new target directory.
4. Move your `__tokudb_lock_dont_delete_me_temp` file from the current location to the new location.
5. Restart your server.

73.3 Moving TokuDB recovery log files to a location outside of the default MySQL datadir

TokuDB will use the location specified by the **:variable:‘tokudb_log_dir‘** variable for all of its recovery log files. If the **:variable:‘tokudb_log_dir‘** variable is not explicitly set, *TokuDB* will use the location specified by the servers datadir for these files.

The *TokuDB* recovery log files are protected from concurrent process access by the `__tokudb_lock_dont_delete_me_logs` file that is located in the same directory as the *TokuDB* recovery log files.

TokuDB recovery log files may be moved to another location with symlinks left behind in place of the **:variable:‘tokudb_log_dir‘**. If that symlink refers to a directory on another physical data volume, the **:variable:‘tokudb_fs_reserve_percent‘** monitor will not traverse the symlink and monitor the real location for adequate space in the file system.

To safely move your *TokuDB* recovery log files:

1. Shut the server down cleanly.
2. Change the **:variable:‘tokudb_log_dir‘** in your `my.cnf` configuration file to the location where you wish to store your *TokuDB* recovery log files.
3. Create your new target directory.
4. Move your `log*.tokulog*` files and your `__tokudb_lock_dont_delete_me_logs` file from the current location to the new location.
5. Restart your server.

73.4 Improved table renaming functionality

When you rename a *TokuDB* table via SQL, the data files on disk keep their original names and only the mapping in the *Percona FT* directory file is changed to map the new dictionary name to the original internal file names. This makes it difficult to quickly match database/table/index names to their actual files on disk, requiring you to use the **:table:'INFORMATION_SCHEMA.TOKUDB_FILE_MAP'** table to cross reference.

Beginning with *Percona Server for MySQL* **:rn:'5.6.33-79.0'** a new server option was introduced called **:variable:'tokudb_dir_per_db'** to address this issue.

When **:variable:'tokudb_dir_per_db'** is enabled (`OFF` by default), this is no longer the case. When you rename a table, the mapping in the *Percona FT* directory file will be updated and the files will be renamed on disk to reflect the new table name.

73.5 Improved directory layout functionality

Many users have had issues with managing the huge volume of individual files that *TokuDB* and *Percona FT* use.

Beginning with *Percona Server for MySQL* **:rn:'5.6.33-79.0'** a new server option was introduced called **:variable:'tokudb_dir_per_db'** to address this issue.

When **:variable:'tokudb_dir_per_db'** variable is enabled (`OFF` by default), all new tables and indices will be placed within their corresponding database directory within the `tokudb_data_dir` or server datadir.

If you have **:variable:'tokudb_data_dir'** variable set to something other than the server datadir, *TokuDB* will create a directory matching the name of the database, but upon dropping of the database, this directory will remain behind.

Existing table files will not be automatically relocated to their corresponding database directory.

You can easily move a tables data files into the new scheme and proper database directory with a few steps:

```
mysql> SET GLOBAL tokudb_dir_per_db=true;
mysql> RENAME TABLE <table> TO <tmp_table>;
mysql> RENAME TABLE <tmp_table> TO <table>;
```

Note: Two renames are needed because *MySQL* doesn't allow you to rename a table to itself. The first rename, renames the table to the temporary name and moves the table files into the owning database directory. The second rename sets the table name back to the original name. Tables can also be renamed/moved across databases and will be placed correctly into the corresponding database directory.

Warning: You must be careful with renaming tables in case you have used any tricks to create symlinks of the database directories on different storage volumes, the move is not a simple directory move on the same volume but a physical copy across volumes. This can take quite some time and prevent access to the table being moved during the copy.

73.6 Editing *TokuDB* files with :variable:'tokudb_dir_cmd'

Note: This feature is currently considered *Experimental*.

In *Percona Server for MySQL* **:rn:'5.6.36-82.0'** new **:variable:'tokudb_dir_cmd'** variable has been implemented that can be used to edit *TokuDB* files. **WARNING:** Use this variable only if you know what you're doing otherwise it **WILL** lead to data loss.

This method can be used if any kind of system issue causes the loss of specific `.tokudb` files for a given table, because the *TokuDB* tablespace file mapping will then contain invalid (nonexistent) entries, visible in **:table:'INFORMATION_SCHEMA.TokuDB_file_map'** table.

This variable is used to send commands to edit directory file. The format of the command line is the following:

```
command arg1 arg2 .. argn
```

I.e, if we want to execute some command the following statement can be used:

```
SET tokudb_dir_cmd = "command arg1 ... argn"
```

Currently the following commands are available:

- `attach dictionary_name internal_file_name` - attach internal_file_name to a dictionary_name, if the dictionary_name exists override the previous value, add new record otherwise
- `detach dictionary_name` - remove record with corresponding dictionary_name, the corresponding internal_file_name file stays untouched
- `move old_dictionary_name new_dictionary_name` - rename (only) dictionary_name from old_dictionary_name to new_dictionary_name

Information about the dictionary_name and internal_file_name can be found in the **:table:'TokuDB_file_map'** table:

```
mysql> SELECT dictionary_name, internal_file_name FROM INFORMATION_SCHEMA.TokuDB_file_
↪map;
+-----------------------------+-------------------------------------------------------
↪---+
| dictionary_name             | internal_file_name                                    ␣
↪    |
+-----------------------------+-------------------------------------------------------
↪---+
| ./world/City-key-CountryCode | ./_world_sql_340a_39_key_CountryCode_12_1_1d_B_1.
↪tokudb |
| ./world/City-main           | ./_world_sql_340a_39_main_12_1_1d_B_0.tokudb          ␣
↪    |
| ./world/City-status         | ./_world_sql_340a_39_status_f_1_1d.tokudb             ␣
↪    |
+-----------------------------+-------------------------------------------------------
↪---+
```

73.6.1 System Variables

This variable is used to send commands to edit *TokuDB* directory files.

> **Warning:** Use this variable only if you know what you're doing otherwise it **WILL** lead to data loss.

73.6.2 Status Variables

This variable contains the error number of the last executed command by using the **:variable:'tokudb_dir_cmd'** variable.

This variable contains the error string of the last executed command by using the **:variable:‘tokudb_dir_cmd‘** variable.

CHAPTER

SEVENTYFOUR

TOKUDB BACKGROUND ANALYZE TABLE

Prior to **:rn:'5.6.27-76.0'** release there was no mechanism to optionally and automatically trigger a background analysis or gathering of cardinality statistics on a *TokuDB* tables. These statistics are used for `JOIN` optimizations and helping the *MySQL* optimizer choose the appropriate index for a query. If a table statistics or index cardinality becomes outdated, you might see queries which previously performed well suddenly running much slower until statistics are updated again. For *TokuDB* cardinality statistics were only updated when an explicit `ANALYZE TABLE` was issued.

Although this operation was an online in the sense that `ANALYZE TABLE` within *TokuDB* didn't explicitly lock anything *MySQL* itself would place a read lock on a table being analyzed, preventing writes while the analysis is in progress.

There was also no throttling or scope limiting mechanism for *TokuDB* `ANALYZE TABLE` other than time limit (**:variable:'tokudb_analyze_time'** 5 second default) that it spends on each index in a table.

In *Percona Server for MySQL* **:rn:'5.6.27-76.0'** new behavior and variables have been implemented to have the tables automatically analyzed in the background based on a measured change in data. This has been done by implementing the new background job manager that can perform operations on a background thread.

74.1 Background Jobs

Background jobs and schedule are transient in nature and are not persisted anywhere. Any currently running job will be terminated on shutdown and all scheduled jobs will be forgotten about on server restart. There can't be two jobs on the same table scheduled or running at any one point in time. If you manually invoke an `ANALYZE TABLE` that conflicts with either a pending or running job, the running job will be canceled and the users task will run immediately in the foreground. All the scheduled and running background jobs can be viewed by querying the **:table:'TOKUDB_BACKGROUND_JOB_STATUS'** table.

New **:variable:'tokudb_analyze_in_background'** variable has been implemented in order to control if the `ANALYZE TABLE` will be dispatched to the background process or if it will be running in the foreground. To control the function of `ANALYZE TABLE` a new **:variable:'tokudb_analyze_mode'** variable has been implemented. This variable offers options to cancel any running or scheduled job on the specified table (`TOKUDB_ANALYZE_CANCEL`), use existing analysis algorithm (`TOKUDB_ANALYZE_STANDARD`), or to recount the logical rows in table and update persistent count (`TOKUDB_ANALYZE_RECOUNT_ROWS`).

`TOKUDB_ANALYZE_RECOUNT_ROWS` is a new mechanism that is used to perform a logical recount of all rows in a table and persist that as the basis value for the table row estimate. This mode was added for tables that have been upgraded from an older version of *TokuDB* that only reported physical row counts and never had a proper logical row count. Newly created tables/partitions will begin counting logical rows correctly from their creation and should not need to be recounted unless some odd edge condition causes the logical count to become inaccurate over time. This analysis mode has no effect on the table cardinality counts. It will take the currently set session values for **:variable:'tokudb_analyze_in_background'**, and **:variable:'tokudb_analyze_throttle'**. Changing the global or session instances of these values after scheduling will have no effect on the job.

Any background job, both pending and running, can be canceled by setting the **:variable:'tokudb_analyze_mode'** to `TOKUDB_ANALYZE_CANCEL` and issuing the `ANALYZE TABLE` on the table for which you want to cancel all the jobs for.

74.2 Auto analysis

To implement the background analysis and gathering of cardinality statistics on a *TokuDB* tables new `delta` value is now maintained in memory for each *TokuDB* table. This value is not persisted anywhere and it is reset to `0` on a server start. It is incremented for each `INSERT/UPDATE/DELETE` command and ignores the impact of transactions (rollback specifically). When this delta value exceeds the **:variable:'tokudb_auto_analyze'** percentage of rows in the table an analysis is performed according to the current session's settings. Other analysis for this table will be disabled until this analysis completes. When this analysis completes, the delta is reset to `0` to begin recalculating table changes for the next potential analysis.

Status values are now reported to server immediately upon completion of any analysis (previously new status values were not used until the table has been closed and re-opened). Half-time direction reversal of analysis has been implemented, meaning that if a **:variable:'tokudb_analyze_time'** is in effect and the analysis has not reached the half way point of the index by the time **:variable:'tokudb_analyze_time'**/2 has been reached: it will stop the forward progress and restart the analysis from the last/rightmost row in the table, progressing leftwards and keeping/adding to the status information accumulated from the first half of the scan.

For small ratios of `table_rows` / **:variable:'tokudb_auto_analyze'**, auto analysis will be run for almost every change. The trigger formula is: `if (table_delta >= ((table_rows * tokudb_auto_analyze) / 100))` then run `ANALYZE TABLE`. If a user manually invokes an `ANALYZE TABLE` and **:variable:'tokudb_auto_analyze'** is enabled and there are no conflicting background jobs, the users `ANALYZE TABLE` will behave exactly as if the delta level has been exceeded in that the analysis is executed and delta reset to `0` upon completion.

74.3 System Variables

When this variable is set to `ON` it will dispatch any `ANALYZE TABLE` job to a background process and return immediately, otherwise `ANALYZE TABLE` will run in foreground/client context.

This variable is used to control the function of `ANALYZE TABLE`. Possible values are:

- `TOKUDB_ANALYZE_CANCEL` - Cancel any running or scheduled job on the specified table.
- `TOKUDB_ANALYZE_STANDARD` - Use existing analysis algorithm. This is the standard table cardinality analysis mode used to obtain cardinality statistics for a tables and its indexes. It will take the currently set session values for **:variable:'tokudb_analyze_time'**, **:variable:'tokudb_analyze_in_background'**, and **:variable:'tokudb_analyze_throttle'** at the time of its scheduling, either via a user invoked `ANALYZE TABLE` or an auto schedule as a result of **:variable:'tokudb_auto_analyze'** threshold being hit. Changing the global or session instances of these values after scheduling will have no effect on the scheduled job.
- `TOKUDB_ANALYZE_RECOUNT_ROWS` - Recount logical rows in table and update persistent count. This is a new mechanism that is used to perform a logical recount of all rows in a table and persist that as the basis value for the table row estimate. This mode was added for tables that have been upgraded from an older version of *TokuDB*/PerconaFT that only reported physical row counts and never had a proper logical row count. Newly created tables/partitions will begin counting logical rows correctly from their creation and should not need to be recounted unless some odd edge condition causes the logical count to become inaccurate over time. This analysis mode has no effect on the table cardinality counts. It will take the currently set session values for **:variable:'tokudb_analyze_in_background'**, and **:variable:'tokudb_analyze_throttle'**. Changing the global or session instances of these values after scheduling will have no effect on the job.

This variable is used to define maximum number of keys to visit per second when performing `ANALYZE TABLE` with either a `TOKUDB_ANALYZE_STANDARD` or `TOKUDB_ANALYZE_RECOUNT_ROWS`.

This session variable controls the number of seconds an analyze operation will spend on each index when calculating cardinality. Cardinality is shown by executing the following command:

```
SHOW INDEXES FROM table_name;
```

If an analyze is never performed on a table then the cardinality is `1` for primary key indexes and unique secondary indexes, and `NULL` (unknown) for all other indexes. Proper cardinality can lead to improved performance of complex SQL statements.

Percentage of table change as `INSERT/UPDATE/DELETE` commands to trigger an `ANALYZE TABLE` using the current session **:variable:‘tokudb_analyze_in_background‘**, **:variable:‘tokudb_analyze_mode‘**, **:variable:‘tokudb_analyze_throttle‘**, and **:variable:‘tokudb_analyze_time‘** settings. If this variable is enabled and **:variable:‘tokudb_analyze_in_background‘** variable is set to `OFF`, analysis will be performed directly within the client thread context that triggered the analysis. **NOTE:** *InnoDB* enabled this functionality by default when they introduced it. Due to the potential unexpected new load it might place on a server, it is disabled by default in *TokuDB*.

Percentage to scale table/index statistics when sending to the server to make an index appear to be either more or less unique than it actually is. *InnoDB* has a hard coded scaling factor of 50%. So if a table of 200 rows had an index with 40 unique values, InnoDB would return 200/40/2 or 2 for the index. The new TokuDB formula is the same but factored differently to use percent, for the same table.index (200/40 * **:variable:‘tokudb_cardinality_scale‘**) / 100, for a scale of 50% the result would also be 2 for the index.

74.4 INFORMATION_SCHEMA Tables

Note: If you’re upgrading an existing *TokuDB* installation prior to **:rn:‘5.6.27-76.0‘**, the *TokuDB* plugin must be disabled and re-enabled via **`ps_tokudb_admin`** to register this new table. Alternately the plugin can be enabled by running:

```
INSTALL PLUGIN tokudb_background_job_status SONAME 'ha_tokudb.so'
```

This table holds the information on scheduled and running background `ANALYZE TABLE` jobs for *TokuDB* tables.

74.5 Version Specific Information

- **:rn:‘5.6.27-76.0‘**: Feature implemented

CHAPTER

SEVENTYFIVE

TOKUDB STATUS VARIABLES

TokuDB status variables provide details about the inner workings of *TokuDB* storage engine and they can be useful in tuning the storage engine to a particular environment.

You can view these variables and their values by running:

```
mysql> SHOW STATUS LIKE 'tokudb%';
```

75.1 TokuDB Status Variables Summary

The following global status variables are available:

Name	Var Type
:variable:‘Tokudb_DB_OPENS‘	integer
:variable:‘Tokudb_DB_CLOSES‘	integer
:variable:‘Tokudb_DB_OPEN_CURRENT‘	integer
:variable:‘Tokudb_DB_OPEN_MAX‘	integer
:variable:‘Tokudb_LEAF_ENTRY_MAX_COMMITTED_XR‘	integer
:variable:‘Tokudb_LEAF_ENTRY_MAX_PROVISIONAL_XR‘	integer
:variable:‘Tokudb_LEAF_ENTRY_EXPANDED‘	integer
:variable:‘Tokudb_LEAF_ENTRY_MAX_MEMSIZE‘	integer
:variable:‘Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_IN‘	integer
:variable:‘Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_OUT‘	integer
:variable:‘Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_IN‘	integer
:variable:‘Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_OUT‘	integer
:variable:‘Tokudb_CHECKPOINT_PERIOD‘	integer
:variable:‘Tokudb_CHECKPOINT_FOOTPRINT‘	integer
:variable:‘Tokudb_CHECKPOINT_LAST_BEGAN‘	datetime
:variable:‘Tokudb_CHECKPOINT_LAST_COMPLETE_BEGAN‘	datetime
:variable:‘Tokudb_CHECKPOINT_LAST_COMPLETE_ENDED‘	datetime
:variable:‘Tokudb_CHECKPOINT_DURATION‘	integer
:variable:‘Tokudb_CHECKPOINT_DURATION_LAST‘	integer
:variable:‘Tokudb_CHECKPOINT_LAST_LSN‘	integer
:variable:‘Tokudb_CHECKPOINT_TAKEN‘	integer
:variable:‘Tokudb_CHECKPOINT_FAILED‘	integer
:variable:‘Tokudb_CHECKPOINT_WAITERS_NOW‘	integer
:variable:‘Tokudb_CHECKPOINT_WAITERS_MAX‘	integer
:variable:‘Tokudb_CHECKPOINT_CLIENT_WAIT_ON_MO‘	integer
:variable:‘Tokudb_CHECKPOINT_CLIENT_WAIT_ON_CS‘	integer

Continued on next page

Table 75.1 – continued from previous page

Name	Var Type
:variable:‘Tokudb_CHECKPOINT_BEGIN_TIME‘	integer
:variable:‘Tokudb_CHECKPOINT_LONG_BEGIN_TIME‘	integer
:variable:‘Tokudb_CHECKPOINT_LONG_BEGIN_COUNT‘	integer
:variable:‘Tokudb_CHECKPOINT_END_TIME‘	integer
:variable:‘Tokudb_CHECKPOINT_LONG_END_TIME‘	integer
:variable:‘Tokudb_CHECKPOINT_LONG_END_COUNT‘	integer
:variable:‘Tokudb_CACHETABLE_MISS‘	integer
:variable:‘Tokudb_CACHETABLE_MISS_TIME‘	integer
:variable:‘Tokudb_CACHETABLE_PREFETCHES‘	integer
:variable:‘Tokudb_CACHETABLE_SIZE_CURRENT‘	integer
:variable:‘Tokudb_CACHETABLE_SIZE_LIMIT‘	integer
:variable:‘Tokudb_CACHETABLE_SIZE_WRITING‘	integer
:variable:‘Tokudb_CACHETABLE_SIZE_NONLEAF‘	integer
:variable:‘Tokudb_CACHETABLE_SIZE_LEAF‘	integer
:variable:‘Tokudb_CACHETABLE_SIZE_ROLLBACK‘	integer
:variable:‘Tokudb_CACHETABLE_SIZE_CACHEPRESSURE‘	integer
:variable:‘Tokudb_CACHETABLE_SIZE_CLONED‘	integer
:variable:‘Tokudb_CACHETABLE_EVICTIONS‘	integer
:variable:‘Tokudb_CACHETABLE_CLEANER_EXECUTIONS‘	integer
:variable:‘Tokudb_CACHETABLE_CLEANER_PERIOD‘	integer
:variable:‘Tokudb_CACHETABLE_CLEANER_ITERATIONS‘	integer
:variable:‘Tokudb_CACHETABLE_WAIT_PRESSURE_COUNT‘	integer
:variable:‘Tokudb_CACHETABLE_WAIT_PRESSURE_TIME‘	integer
:variable:‘Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_COUNT‘	integer
:variable:‘Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_TIME‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS_ACTIVE‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_QUEUE_SIZE‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_MAX_QUEUE_SIZE‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_ITEMS_PROCESSED‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_EXECUTION_TIME‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS_ACTIVE‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_QUEUE_SIZE‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_MAX_QUEUE_SIZE‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_ITEMS_PROCESSED‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_EXECUTION_TIME‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS_ACTIVE‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_QUEUE_SIZE‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_MAX_QUEUE_SIZE‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_ITEMS_PROCESSED‘	integer
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_EXECUTION_TIME‘	integer
:variable:‘Tokudb_LOCKTREE_MEMORY_SIZE‘	integer
:variable:‘Tokudb_LOCKTREE_MEMORY_SIZE_LIMIT‘	integer
:variable:‘Tokudb_LOCKTREE_ESCALATION_NUM‘	integer
:variable:‘Tokudb_LOCKTREE_ESCALATION_SECONDS‘	numeric
:variable:‘Tokudb_LOCKTREE_LATEST_POST_ESCALATION_MEMORY_SIZE‘	integer
:variable:‘Tokudb_LOCKTREE_OPEN_CURRENT‘	integer

Continued on next page

Table 75.1 – continued from previous page

Name	Var Type
:variable:‘Tokudb_LOCKTREE_PENDING_LOCK_REQUESTS‘	integer
:variable:‘Tokudb_LOCKTREE_STO_ELIGIBLE_NUM‘	integer
:variable:‘Tokudb_LOCKTREE_STO_ENDED_NUM‘	integer
:variable:‘Tokudb_LOCKTREE_STO_ENDED_SECONDS‘	numeric
:variable:‘Tokudb_LOCKTREE_WAIT_COUNT‘	integer
:variable:‘Tokudb_LOCKTREE_WAIT_TIME‘	integer
:variable:‘Tokudb_LOCKTREE_LONG_WAIT_COUNT‘	integer
:variable:‘Tokudb_LOCKTREE_LONG_WAIT_TIME‘	integer
:variable:‘Tokudb_LOCKTREE_TIMEOUT_COUNT‘	integer
:variable:‘Tokudb_LOCKTREE_WAIT_ESCALATION_COUNT‘	integer
:variable:‘Tokudb_LOCKTREE_WAIT_ESCALATION_TIME‘	integer
:variable:‘Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_COUNT‘	integer
:variable:‘Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_TIME‘	integer
:variable:‘Tokudb_DICTIONARY_UPDATES‘	integer
:variable:‘Tokudb_DICTIONARY_BROADCAST_UPDATES‘	integer
:variable:‘Tokudb_DESCRIPTOR_SET‘	integer
:variable:‘Tokudb_MESSAGES_IGNORED_BY_LEAF_DUE_TO_MSN‘	integer
:variable:‘Tokudb_TOTAL_SEARCH_RETRIES‘	integer
:variable:‘Tokudb_SEARCH_TRIES_GT_HEIGHT‘	integer
:variable:‘Tokudb_SEARCH_TRIES_GT_HEIGHTPLUS3‘	integer
:variable:‘Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT‘	integer
:variable:‘Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_BYTES‘	integer
:variable:‘Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_UNCOMPRESSED_BYTES‘	integer
:variable:‘Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_SECONDS‘	numeric
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT‘	integer
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_BYTES‘	integer
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_UNCOMPRESSE‘	integer
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_SECONDS‘	numeric
:variable:‘Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT‘	integer
:variable:‘Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_BYTES‘	integer
:variable:‘Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_UNCOMPRESSED_BYTES‘	integer
:variable:‘Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_SECONDS‘	numeric
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT‘	integer
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_BYTES‘	integer
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_UNCOMPRESSED_BY‘	integer
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_SECONDS‘	numeric
:variable:‘Tokudb_LEAF_NODE_COMPRESSION_RATIO‘	numeric
:variable:‘Tokudb_NONLEAF_NODE_COMPRESSION_RATIO‘	numeric
:variable:‘Tokudb_OVERALL_NODE_COMPRESSION_RATIO‘	numeric
:variable:‘Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS‘	numeric
:variable:‘Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS_BYTES‘	integer
:variable:‘Tokudb_LEAF_NODE_PARTIAL_EVICTIONS‘	integer
:variable:‘Tokudb_LEAF_NODE_PARTIAL_EVICTIONS_BYTES‘	integer
:variable:‘Tokudb_LEAF_NODE_FULL_EVICTIONS‘	integer
:variable:‘Tokudb_LEAF_NODE_FULL_EVICTIONS_BYTES‘	integer
:variable:‘Tokudb_NONLEAF_NODE_FULL_EVICTIONS‘	integer
:variable:‘Tokudb_NONLEAF_NODE_FULL_EVICTIONS_BYTES‘	integer
:variable:‘Tokudb_LEAF_NODES_CREATED‘	integer
:variable:‘Tokudb_NONLEAF_NODES_CREATED‘	integer

Continued on next page

Table 75.1 – continued from previous page

Name	Var Type
:variable:‘Tokudb_LEAF_NODES_DESTROYED‘	integer
:variable:‘Tokudb_NONLEAF_NODES_DESTROYED‘	integer
:variable:‘Tokudb_MESSAGES_INJECTED_AT_ROOT_BYTES‘	integer
:variable:‘Tokudb_MESSAGES_FLUSHED_FROM_H1_TO_LEAVES_BYTES‘	integer
:variable:‘Tokudb_MESSAGES_IN_TREES_ESTIMATE_BYTES‘	integer
:variable:‘Tokudb_MESSAGES_INJECTED_AT_ROOT‘	integer
:variable:‘Tokudb_BROADCASE_MESSAGES_INJECTED_AT_ROOT‘	integer
:variable:‘Tokudb_BASEMENTS_DECOMPRESSED_TARGET_QUERY‘	integer
:variable:‘Tokudb_BASEMENTS_DECOMPRESSED_PRELOCKED_RANGE‘	integer
:variable:‘Tokudb_BASEMENTS_DECOMPRESSED_PREFETCH‘	integer
:variable:‘Tokudb_BASEMENTS_DECOMPRESSED_FOR_WRITE‘	integer
:variable:‘Tokudb_BUFFERS_DECOMPRESSED_TARGET_QUERY‘	integer
:variable:‘Tokudb_BUFFERS_DECOMPRESSED_PRELOCKED_RANGE‘	integer
:variable:‘Tokudb_BUFFERS_DECOMPRESSED_PREFETCH‘	integer
:variable:‘Tokudb_BUFFERS_DECOMPRESSED_FOR_WRITE‘	integer
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_QUERY‘	integer
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_QUERY_BYTES‘	integer
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_QUERY_SECONDS‘	numeric
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_PREFETCH‘	integer
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_BYTES‘	integer
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_SECONDS‘	numeric
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_WRITE‘	integer
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_WRITE_BYTES‘	integer
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_WRITE_SECONDS‘	numeric
:variable:‘Tokudb_BASEMENTS_FETCHED_TARGET_QUERY‘	integer
:variable:‘Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_BYTES‘	integer
:variable:‘Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_SECONDS‘	numeric
:variable:‘Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE‘	integer
:variable:‘Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_BYTES‘	integer
:variable:‘Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_SECONDS‘	numeric
:variable:‘Tokudb_BASEMENTS_FETCHED_PREFETCH‘	integer
:variable:‘Tokudb_BASEMENTS_FETCHED_PREFETCH_BYTES‘	integer
:variable:‘Tokudb_BASEMENTS_FETCHED_PREFETCH_SECONDS‘	numeric
:variable:‘Tokudb_BASEMENTS_FETCHED_FOR_WRITE‘	integer
:variable:‘Tokudb_BASEMENTS_FETCHED_FOR_WRITE_BYTES‘	integer
:variable:‘Tokudb_BASEMENTS_FETCHED_FOR_WRITE_SECONDS‘	numeric
:variable:‘Tokudb_BUFFERS_FETCHED_TARGET_QUERY‘	integer
:variable:‘Tokudb_BUFFERS_FETCHED_TARGET_QUERY_BYTES‘	integer
:variable:‘Tokudb_BUFFERS_FETCHED_TARGET_QUERY_SECONDS‘	numeric
:variable:‘Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE‘	integer
:variable:‘Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_BYTES‘	integer
:variable:‘Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_SECONDS‘	numeric
:variable:‘Tokudb_BUFFERS_FETCHED_PREFETCH‘	integer
:variable:‘Tokudb_BUFFERS_FETCHED_PREFETCH_BYTES‘	integer
:variable:‘Tokudb_BUFFERS_FETCHED_PREFETCH_SECONDS‘	numeric
:variable:‘Tokudb_BUFFERS_FETCHED_FOR_WRITE‘	integer
:variable:‘Tokudb_BUFFERS_FETCHED_FOR_WRITE_BYTES‘	integer
:variable:‘Tokudb_BUFFERS_FETCHED_FOR_WRITE_SECONDS‘	integer
:variable:‘Tokudb_LEAF_COMPRESSION_TO_MEMORY_SECONDS‘	numeric

Continued on next page

Table 75.1 – continued from previous page

Name	Var Type
:variable:‘Tokudb_LEAF_SERIALIZATION_TO_MEMORY_SECONDS‘	numeric
:variable:‘Tokudb_LEAF_DECOMPRESSION_TO_MEMORY_SECONDS‘	numeric
:variable:‘Tokudb_LEAF_DESERIALIZATION_TO_MEMORY_SECONDS‘	numeric
:variable:‘Tokudb_NONLEAF_COMPRESSION_TO_MEMORY_SECONDS‘	numeric
:variable:‘Tokudb_NONLEAF_SERIALIZATION_TO_MEMORY_SECONDS‘	numeric
:variable:‘Tokudb_NONLEAF_DECOMPRESSION_TO_MEMORY_SECONDS‘	numeric
:variable:‘Tokudb_NONLEAF_DESERIALIZATION_TO_MEMORY_SECONDS‘	numeric
:variable:‘Tokudb_PROMOTION_ROOTS_SPLIT‘	integer
:variable:‘Tokudb_PROMOTION_LEAF_ROOTS_INJECTED_INTO‘	integer
:variable:‘Tokudb_PROMOTION_H1_ROOTS_INJECTED_INTO‘	integer
:variable:‘Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_0‘	integer
:variable:‘Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_1‘	integer
:variable:‘Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_2‘	integer
:variable:‘Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_3‘	integer
:variable:‘Tokudb_PROMOTION_INJECTIONS_LOWER_THAN_DEPTH_3‘	integer
:variable:‘Tokudb_PROMOTION_STOPPED_NONEMPTY_BUFFER‘	integer
:variable:‘Tokudb_PROMOTION_STOPPED_AT_HEIGHT_1‘	integer
:variable:‘Tokudb_PROMOTION_STOPPED_CHILD_LOCKED_OR_NOT_IN_MEMORY‘	integer
:variable:‘Tokudb_PROMOTION_STOPPED_CHILD_NOT_FULLY_IN_MEMORY‘	integer
:variable:‘Tokudb_PROMOTION_STOPPED_AFTER_LOCKING_CHILD‘	integer
:variable:‘Tokudb_BASEMENT_DESERIALIZATION_FIXED_KEY‘	integer
:variable:‘Tokudb_BASEMENT_DESERIALIZATION_VARIABLE_KEY‘	integer
:variable:‘Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_SUCCESS‘	integer
:variable:‘Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_FAIL_POS‘	integer
:variable:‘Tokudb_RIGHTMOST_LEAF_SHORTCUT_FAIL_REACTIVE‘	integer
:variable:‘Tokudb_CURSOR_SKIP_DELETED_LEAF_ENTRY‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_TOTAL_NODES‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_H1_NODES‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_HGT1_NODES‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_EMPTY_NODES‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_NODES_DIRTIED‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_MAX_BUFFER_SIZE‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_MIN_BUFFER_SIZE‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_SIZE‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_MAX_BUFFER_WORKDONE‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_MIN_BUFFER_WORKDONE‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_WORKDONE‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_STARTED‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_RUNNING‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_COMPLETED‘	integer
:variable:‘Tokudb_FLUSHER_CLEANER_NUM_DIRTIED_FOR_LEAF_MERGE‘	integer
:variable:‘Tokudb_FLUSHER_FLUSH_TOTAL‘	integer
:variable:‘Tokudb_FLUSHER_FLUSH_IN_MEMORY‘	integer
:variable:‘Tokudb_FLUSHER_FLUSH_NEEDED_IO‘	integer
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES‘	integer
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_1‘	integer
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_2‘	integer
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_3‘	integer
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_4‘	integer

Continued on next page

Table 75.1 – continued from previous page

Name	Var Type
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_5‘	integer
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_GT_5‘	integer
:variable:‘Tokudb_FLUSHER_SPLIT_LEAF‘	integer
:variable:‘Tokudb_FLUSHER_SPLIT_NONLEAF‘	integer
:variable:‘Tokudb_FLUSHER_MERGE_LEAF‘	integer
:variable:‘Tokudb_FLUSHER_MERGE_NONLEAF‘	integer
:variable:‘Tokudb_FLUSHER_BALANCE_LEAF‘	integer
:variable:‘Tokudb_HOT_NUM_STARTED‘	integer
:variable:‘Tokudb_HOT_NUM_COMPLETED‘	integer
:variable:‘Tokudb_HOT_NUM_ABORTED‘	integer
:variable:‘Tokudb_HOT_MAX_ROOT_FLUSH_COUNT‘	integer
:variable:‘Tokudb_TXN_BEGIN‘	integer
:variable:‘Tokudb_TXN_BEGIN_READ_ONLY‘	integer
:variable:‘Tokudb_TXN_COMMITS‘	integer
:variable:‘Tokudb_TXN_ABORTS‘	integer
:variable:‘Tokudb_LOGGER_NEXT_LSN‘	integer
:variable:‘Tokudb_LOGGER_WRITES‘	integer
:variable:‘Tokudb_LOGGER_WRITES_BYTES‘	integer
:variable:‘Tokudb_LOGGER_WRITES_UNCOMPRESSED_BYTES‘	integer
:variable:‘Tokudb_LOGGER_WRITES_SECONDS‘	numeric
:variable:‘Tokudb_LOGGER_WAIT_LONG‘	integer
:variable:‘Tokudb_LOADER_NUM_CREATED‘	integer
:variable:‘Tokudb_LOADER_NUM_CURRENT‘	integer
:variable:‘Tokudb_LOADER_NUM_MAX‘	integer
:variable:‘Tokudb_MEMORY_MALLOC_COUNT‘	integer
:variable:‘Tokudb_MEMORY_FREE_COUNT‘	integer
:variable:‘Tokudb_MEMORY_REALLOC_COUNT‘	integer
:variable:‘Tokudb_MEMORY_MALLOC_FAIL‘	integer
:variable:‘Tokudb_MEMORY_REALLOC_FAIL‘	integer
:variable:‘Tokudb_MEMORY_REQUESTED‘	integer
:variable:‘Tokudb_MEMORY_USED‘	integer
:variable:‘Tokudb_MEMORY_FREED‘	integer
:variable:‘Tokudb_MEMORY_MAX_REQUESTED_SIZE‘	integer
:variable:‘Tokudb_MEMORY_LAST_FAILED_SIZE‘	integer
:variable:‘Tokudb_MEM_ESTIMATED_MAXIMUM_MEMORY_FOOTPRINT‘	integer
:variable:‘Tokudb_MEMORY_MALLOCATOR_VERSION‘	string
:variable:‘Tokudb_MEMORY_MMAP_THRESHOLD‘	integer
:variable:‘Tokudb_FILESYSTEM_THREADS_BLOCKED_BY_FULL_DISK‘	integer
:variable:‘Tokudb_FILESYSTEM_FSYNC_TIME‘	integer
:variable:‘Tokudb_FILESYSTEM_FSYNC_NUM‘	integer
:variable:‘Tokudb_FILESYSTEM_LONG_FSYNC_TIME‘	integer
:variable:‘Tokudb_FILESYSTEM_LONG_FSYNC_NUM‘	integer

This variable shows the number of times an individual PerconaFT dictionary file was opened. This is a not a useful value for a regular user to use for any purpose due to layers of open/close caching on top.

This variable shows the number of times an individual PerconaFT dictionary file was closed. This is a not a useful value for a regular user to use for any purpose due to layers of open/close caching on top.

This variable shows the number of currently opened databases.

This variable shows the maximum number of concurrently opened databases.

This variable shows the maximum number of committed transaction records that were stored on disk in a new or modified row.

This variable shows the maximum number of provisional transaction records that were stored on disk in a new or modified row.

This variable shows the number of times that an expanded memory mechanism was used to store a new or modified row on disk.

This variable shows the maximum number of bytes that were stored on disk as a new or modified row. This is the maximum uncompressed size of any row stored in *TokuDB* that was created or modified since the server started.

This variable shows the total number of bytes of leaf nodes data before performing garbage collection for non-flush events.

This variable shows the total number of bytes of leaf nodes data after performing garbage collection for non-flush events.

This variable shows the total number of bytes of leaf nodes data before performing garbage collection for flush events.

This variable shows the total number of bytes of leaf nodes data after performing garbage collection for flush events.

This variable shows the interval in seconds between the end of an automatic checkpoint and the beginning of the next automatic checkpoint.

This variable shows at what stage the checkpointer is at. It's used for debugging purposes only and not a useful value for a normal user.

This variable shows the time the last checkpoint began. If a checkpoint is currently in progress, then this time may be later than the time the last checkpoint completed. If no checkpoint has ever taken place, then this value will be `Dec 31, 1969` on Linux hosts.

This variable shows the time the last complete checkpoint started. Any data that changed after this time will not be captured in the checkpoint.

This variable shows the time the last complete checkpoint ended.

This variable shows time (in seconds) required to complete all checkpoints.

This variable shows time (in seconds) required to complete the last checkpoint.

This variable shows the last successful checkpoint LSN. Each checkpoint from the time the PerconaFT environment is created has a monotonically incrementing LSN. This is not a useful value for a normal user to use for any purpose other than having some idea of how many checkpoints have occurred since the system was first created.

This variable shows the number of complete checkpoints that have been taken.

This variable shows the number of checkpoints that have failed for any reason.

This variable shows the current number of threads waiting for the `checkpoint safe` lock. This is a not a useful value for a regular user to use for any purpose.

This variable shows the maximum number of threads that concurrently waited for the `checkpoint safe` lock. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of times a non-checkpoint client thread waited for the multi-operation lock. It is an internal `rwlock` that is similar in nature to the *InnoDB* kernel mutex, it effectively halts all access to the PerconaFT API when write locked. The `begin` phase of the checkpoint takes this lock for a brief period.

This variable shows the number of times a non-checkpoint client thread waited for the checkpoint-safe lock. This is the lock taken when you `SET tokudb_checkpoint_lock=1`. If a client trying to lock/postpone the checkpointer has to wait for the currently running checkpoint to complete, that wait time will be reflected here and summed. This is not a useful metric as regular users should never be manipulating the checkpoint lock.

This variable shows the cumulative time (in microseconds) required to mark all dirty nodes as pending a checkpoint.

This variable shows the cumulative actual time (in microseconds) of checkpoint `begin` stages that took longer than 1 second.

This variable shows the number of checkpoints whose `begin` stage took longer than 1 second.

This variable shows the time spent in checkpoint end operation in seconds.

This variable shows the total time of long checkpoints in seconds.

This variable shows the number of checkpoints whose `end_checkpoint` operations exceeded 1 minute.

This variable shows the number of times the application was unable to access the data in the internal cache. A cache miss means that date will need to be read from disk.

This variable shows the total time, in microseconds, of how long the database has had to wait for a disk read to complete.

This variable shows the total number of times that a block of memory has been prefetched into the database's cache. Data is prefetched when the database's algorithms determine that a block of memory is likely to be accessed by the application.

This variable shows how much of the uncompressed data, in bytes, is currently in the database's internal cache.

This variable shows how much of the uncompressed data, in bytes, will fit in the database's internal cache.

This variable shows the number of bytes that are currently queued up to be written to disk.

This variable shows the amount of memory, in bytes, the current set of non-leaf nodes occupy in the cache.

This variable shows the amount of memory, in bytes, the current set of (decompressed) leaf nodes occupy in the cache.

This variable shows the rollback nodes size, in bytes, in the cache.

This variable shows the number of bytes causing cache pressure (the sum of buffers and work done counters), helps to understand if cleaner threads are keeping up with workload. It should really be looked at as more of a value to use in a ratio of cache pressure / cache table size. The closer that ratio evaluates to 1, the higher the cache pressure.

This variable shows the amount of memory, in bytes, currently used for cloned nodes. During the checkpoint operation, dirty nodes are cloned prior to serialization/compression, then written to disk. After which, the memory for the cloned block is returned for re-use.

This variable shows the number of blocks evicted from cache. On its own this is not a useful number as its impact on performance depends entirely on the hardware and workload in use. For example, two workloads, one random, one linear for the same starting data set will have two wildly different eviction patterns.

This variable shows the total number of times the cleaner thread loop has executed.

TokuDB includes a cleaner thread that optimizes indexes in the background. This variable is the time, in seconds, between the completion of a group of cleaner operations and the beginning of the next group of cleaner operations. The cleaner operations run on a background thread performing work that does not need to be done on the client thread.

This variable shows the number of cleaner operations that are performed every cleaner period.

This variable shows the number of times a thread was stalled due to cache pressure.

This variable shows the total time, in microseconds, waiting on cache pressure to subside.

This variable shows the number of times a thread was stalled for more than one second due to cache pressure.

This variable shows the total time, in microseconds, waiting on cache pressure to subside for more than one second.

This variable shows the number of threads in the client thread pool.

This variable shows the number of currently active threads in the client thread pool.

This variable shows the number of currently queued work items in the client thread pool.

This variable shows the largest number of queued work items in the client thread pool.

This variable shows the total number of work items processed in the client thread pool.

This variable shows the total execution time of processing work items in the client thread pool.

This variable shows the number of threads in the cachetable threadpool.

This variable shows the number of currently active threads in the cachetable thread pool.

This variable shows the number of currently queued work items in the cachetable thread pool.

This variable shows the largest number of queued work items in the cachetable thread pool.

This variable shows the total number of work items processed in the cachetable thread pool.

This variable shows the total execution time of processing work items in the cachetable thread pool.

This variable shows the number of threads in the checkpoint threadpool.

This variable shows the number of currently active threads in the checkpoint thread pool.

This variable shows the number of currently queued work items in the checkpoint thread pool.

This variable shows the largest number of queued work items in the checkpoint thread pool.

This variable shows the total number of work items processed in the checkpoint thread pool.

This variable shows the total execution time of processing work items in the checkpoint thread pool.

This variable shows the amount of memory, in bytes, that the locktree is currently using.

This variable shows the maximum amount of memory, in bytes, that the locktree is allowed to use.

This variable shows the number of times the locktree needed to run lock escalation to reduce its memory footprint.

This variable shows the total number of seconds spent performing locktree escalation.

This variable shows the locktree size, in bytes, after most current locktree escalation.

This variable shows the number of locktrees that are currently opened.

This variable shows the number of requests waiting for a lock grant.

This variable shows the number of locktrees eligible for `Single Transaction optimizations`. STO optimization are behaviors that can happen within the locktree when there is exactly one transaction active within the locktree. This is a not a useful value for a regular user to use for any purpose.

This variable shows the total number of times a `Single Transaction Optimization` was ended early due to another transaction starting. STO optimization are behaviors that can happen within the locktree when there is exactly one transaction active within the locktree. This is a not a useful value for a regular user to use for any purpose.

This variable shows the total number of seconds ending the `Single Transaction Optimizations`. STO optimization are behaviors that can happen within the locktree when there is exactly one transaction active within the locktree. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of times that a lock request could not be acquired because of a conflict with some other transaction. PerconaFT lock request cycles to try to obtain a lock, if it can not get a lock, it sleeps/waits and times out, checks to get the lock again, repeat. This value indicates the number of cycles it needed to execute before it obtained the lock.

This variable shows the total time, in microseconds, spent by client waiting for a lock conflict to be resolved.

This variable shows number of lock waits greater than one second in duration.

This variable shows the total time, in microseconds, of the long waits.

This variable shows the number of times that a lock request timed out.

When the sum of the sizes of locks taken reaches the lock tree limit, we run lock escalation on a background thread. The clients threads need to wait for escalation to consolidate locks and free up memory. This variables shows the number of times a client thread had to wait on lock escalation.

This variable shows the total time, in microseconds, that a client thread spent waiting for lock escalation to free up memory.

This variable shows number of times that a client thread had to wait on lock escalation and the wait time was greater than one second.

This variable shows the total time, in microseconds, of the long waits for lock escalation to free up memory.

This variable shows the total number of rows that have been updated in all primary and secondary indexes combined, if those updates have been done with a separate recovery log entry per index.

This variable shows the number of broadcast updates that have been successfully performed. A broadcast update is an update that affects all rows in a dictionary.

This variable shows the number of time a descriptor was updated when the entire dictionary was updated (for example, when the schema has been changed).

This variable shows the number of messages that were ignored by a leaf because it had already been applied.

Internal value that is no use to anyone other than a developer debugging a specific query/search issue.

Internal value that is no use to anyone other than a developer debugging a specific query/search issue.

Internal value that is no use to anyone other than a developer debugging a specific query/search issue.

This variable shows the number of leaf nodes flushed to disk, not for checkpoint.

This variable shows the size, in bytes, of leaf nodes flushed to disk, not for checkpoint.

This variable shows the size, in bytes, of uncompressed leaf nodes flushed to disk not for checkpoint.

This variable shows the number of seconds waiting for I/O when writing leaf nodes flushed to disk, not for checkpoint

This variable shows the number of non-leaf nodes flushed to disk, not for checkpoint.

This variable shows the size, in bytes, of non-leaf nodes flushed to disk, not for checkpoint.

This variable shows the size, in bytes, of uncompressed non-leaf nodes flushed to disk not for checkpoint.

This variable shows the number of seconds waiting for I/O when writing non-leaf nodes flushed to disk, not for checkpoint

This variable shows the number of leaf nodes flushed to disk, for checkpoint.

This variable shows the size, in bytes, of leaf nodes flushed to disk, for checkpoint.

This variable shows the size, in bytes, of uncompressed leaf nodes flushed to disk for checkpoint.

This variable shows the number of seconds waiting for I/O when writing leaf nodes flushed to disk for checkpoint

This variable shows the number of non-leaf nodes flushed to disk, for checkpoint.

This variable shows the size, in bytes, of non-leaf nodes flushed to disk, for checkpoint.

This variable shows the size, in bytes, of uncompressed non-leaf nodes flushed to disk for checkpoint.

This variable shows the number of seconds waiting for I/O when writing non-leaf nodes flushed to disk for checkpoint

This variable shows the ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for leaf nodes.

This variable shows the ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for non-leaf nodes.

This variable shows the ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for all nodes.

This variable shows the number of times a partition of a non-leaf node was evicted from the cache.

This variable shows the amount, in bytes, of memory freed by evicting partitions of non-leaf nodes from the cache.

This variable shows the number of times a partition of a leaf node was evicted from the cache.

This variable shows the amount, in bytes, of memory freed by evicting partitions of leaf nodes from the cache.

This variable shows the number of times a full leaf node was evicted from the cache.

This variable shows the amount, in bytes, of memory freed by evicting full leaf nodes from the cache.

This variable shows the number of times a full non-leaf node was evicted from the cache.

This variable shows the amount, in bytes, of memory freed by evicting full non-leaf nodes from the cache.

This variable shows the number of created leaf nodes.

This variable shows the number of created non-leaf nodes.

This variable shows the number of destroyed leaf nodes.

This variable shows the number of destroyed non-leaf nodes.

This variable shows the size, in bytes, of messages injected at root (for all trees).

This variable shows the size, in bytes, of messages flushed from `h1` nodes to leaves.

This variable shows the estimated size, in bytes, of messages currently in trees.

This variables shows the number of messages that were injected at root node of a tree.

This variable shows the number of broadcast messages dropped into the root node of a tree. These are things such as the result of `OPTIMIZE TABLE` and a few other operations. This is not a useful metric for a regular user to use for any purpose.

This variable shows the number of basement nodes decompressed for queries.

This variable shows the number of basement nodes aggressively decompressed by queries.

This variable shows the number of basement nodes decompressed by a prefetch thread.

This variable shows the number of basement nodes decompressed for writes.

This variable shows the number of buffers decompressed for queries.

This variable shows the number of buffers decompressed by queries aggressively.

This variable shows the number of buffers decompressed by a prefetch thread.

This variable shows the number of buffers decompressed for writes.

This variable shows the number of pivot nodes fetched for queries.

This variable shows the number of bytes of pivot nodes fetched for queries.

This variable shows the number of seconds waiting for I/O when fetching pivot nodes for queries.

This variable shows the number of pivot nodes fetched by a prefetch thread.

This variable shows the number of bytes of pivot nodes fetched for queries.

This variable shows the number seconds waiting for I/O when fetching pivot nodes by a prefetch thread.

This variable shows the number of pivot nodes fetched for writes.

This variable shows the number of bytes of pivot nodes fetched for writes.

This variable shows the number of seconds waiting for I/O when fetching pivot nodes for writes.

This variable shows the number of basement nodes fetched from disk for queries.

This variable shows the number of basement node bytes fetched from disk for queries.

This variable shows the number of seconds waiting for I/O when fetching basement nodes from disk for queries.

This variable shows the number of basement nodes fetched from disk aggressively.

This variable shows the number of basement node bytes fetched from disk aggressively.

This variable shows the number of seconds waiting for I/O when fetching basement nodes from disk aggressively.

This variable shows the number of basement nodes fetched from disk by a prefetch thread.

This variable shows the number of basement node bytes fetched from disk by a prefetch thread.

This variable shows the number of seconds waiting for I/O when fetching basement nodes from disk by a prefetch thread.

This variable shows the number of buffers fetched from disk for writes.

This variable shows the number of buffer bytes fetched from disk for writes.

This variable shows the number of seconds waiting for I/O when fetching buffers from disk for writes.

This variable shows the number of buffers fetched from disk for queries.

This variable shows the number of buffer bytes fetched from disk for queries.

This variable shows the number of seconds waiting for I/O when fetching buffers from disk for queries.

This variable shows the number of buffers fetched from disk aggressively.

This variable shows the number of buffer bytes fetched from disk aggressively.

This variable shows the number of seconds waiting for I/O when fetching buffers from disk aggressively.

This variable shows the number of buffers fetched from disk aggressively.

This variable shows the number of buffer bytes fetched from disk by a prefetch thread.

This variable shows the number of seconds waiting for I/O when fetching buffers from disk by a prefetch thread.

This variable shows the number of buffers fetched from disk for writes.

This variable shows the number of buffer bytes fetched from disk for writes.

This variable shows the number of seconds waiting for I/O when fetching buffers from disk for writes.

This variable shows the total time, in seconds, spent compressing leaf nodes.

This variable shows the total time, in seconds, spent serializing leaf nodes.

This variable shows the total time, in seconds, spent decompressing leaf nodes.

This variable shows the total time, in seconds, spent deserializing leaf nodes.

This variable shows the total time, in seconds, spent compressing non leaf nodes.

This variable shows the total time, in seconds, spent serializing non leaf nodes.

This variable shows the total time, in seconds, spent decompressing non leaf nodes.

This variable shows the total time, in seconds, spent deserializing non leaf nodes.

This variable shows the number of times the root split during promotion.

This variable shows the number of times a message stopped at a root with height `0`.

This variable shows the number of times a message stopped at a root with height `1`.

This variable shows the number of times a message stopped at depth `0`.

This variable shows the number of times a message stopped at depth `1`.

This variable shows the number of times a message stopped at depth `2`.

This variable shows the number of times a message stopped at depth `3`.

This variable shows the number of times a message was promoted past depth `3`.

This variable shows the number of times a message stopped because it reached a nonempty buffer.

This variable shows the number of times a message stopped because it had reached height `1`.

This variable shows the number of times a message stopped because it could not cheaply get access to a child.

This variable shows the number of times a message stopped because it could not cheaply get access to a child.

This variable shows the number of times a message stopped before a child which had been locked.

This variable shows the number of basement nodes deserialized where all keys had the same size, leaving the basement in a format that is optimal for in-memory workloads.

This variable shows the number of basement nodes deserialized where all keys did not have the same size, and thus ineligible for an in-memory optimization.

This variable shows the number of times a message injection detected a series of sequential inserts to the rightmost side of the tree and successfully applied an insert message directly to the rightmost leaf node. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of times a message injection detected a series of sequential inserts to the rightmost side of the tree and was unable to follow the pattern of directly applying an insert message directly to the rightmost leaf node because the key does not continue the sequence. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of times a message injection detected a series of sequential inserts to the rightmost side of the tree and was unable to follow the pattern of directly applying an insert message directly to the rightmost leaf node because the leaf is full. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of leaf entries skipped during search/scan because the result of message application and reconciliation of the leaf entry MVCC stack reveals that the leaf entry is `deleted` in the current transactions view. It is a good indicator that there might be excessive garbage in a tree if a range scan seems to take too long.

This variable shows the total number of nodes potentially flushed by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of height `1` nodes that had messages flushed by flusher or cleaner threads, i.e., internal nodes immediately above leaf nodes. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of nodes with height greater than `1` that had messages flushed by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of nodes cleaned by flusher or cleaner threads which had empty message buffers. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of nodes dirtied by flusher or cleaner threads as a result of flushing messages downward. This is a not a useful value for a regular user to use for any purpose.

This variable shows the maximum bytes in a message buffer flushed by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the minimum bytes in a message buffer flushed by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the total bytes in buffers flushed by flusher and cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the maximum bytes worth of work done in a message buffer flushed by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the minimum bytes worth of work done in a message buffer flushed by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the total bytes worth of work done in buffers flushed by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of times flusher and cleaner threads tried to merge two leafs. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of flusher and cleaner threads leaf merges in progress. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of successful flusher and cleaner threads leaf merges. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of nodes dirtied by flusher or cleaner threads performing leaf node merges. This is a not a useful value for a regular user to use for any purpose.

This variable shows the total number of flushes done by flusher threads or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of in memory flushes (required no disk reads) by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of flushes that read something off disk by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of flushes that triggered a flush in child node by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of flushes that triggered one cascading flush by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of flushes that triggered two cascading flushes by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of flushes that triggered three cascading flushes by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of flushes that triggered four cascading flushes by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of flushes that triggered five cascading flushes by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of flushes that triggered more than five cascading flushes by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the total number of leaf node splits done by flusher threads or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the total number of non-leaf node splits done by flusher threads or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the total number of leaf node merges done by flusher threads or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the total number of non-leaf node merges done by flusher threads or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of times two adjacent leaf nodes were rebalanced or had their content redistributed evenly by flusher or cleaner threads. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of hot operations started (`OPTIMIZE TABLE`). This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of hot operations completed (`OPTIMIZE TABLE`). This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of hot operations aborted (`OPTIMIZE TABLE`). This is a not a useful value for a regular user to use for any purpose.

This variable shows the maximum number of flushes from root ever required to optimize trees. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of transactions that have been started.

This variable shows the number of read-only transactions started.

This variable shows the total number of transactions that have been committed.

This variable shows the total number of transactions that have been aborted.

This variable shows the recovery logger next LSN. This is a not a useful value for a regular user to use for any purpose.

This variable shows the number of times the logger has written to disk.

This variable shows the number of bytes the logger has written to disk.

This variable shows the number of uncompressed bytes the logger has written to disk.

This variable shows the number of seconds waiting for IO when writing logs to disk.

This variable shows the number of times a logger write operation required 100ms or more.

This variable shows the number of times one of our internal objects, a loader, has been created.

This variable shows the number of loaders that currently exist.

This variable shows the maximum number of loaders that ever existed simultaneously.

This variable shows the number of `malloc` operations by PerconaFT.

This variable shows the number of `free` operations by PerconaFT.

This variable shows the number of `realloc` operations by PerconaFT.

This variable shows the number of `malloc` operations that failed by PerconaFT.

This variable shows the number of `realloc` operations that failed by PerconaFT.

This variable shows the number of bytes requested by PerconaFT.

This variable shows the number of bytes used (requested + overhead) by PerconaFT.

This variable shows the number of bytes freed by PerconaFT.

This variable shows the largest attempted allocation size by PerconaFT.

This variable shows the size of the last failed allocation attempt by PerconaFT.

This variable shows the maximum memory footprint of the storage engine, the max value of (used - freed).

This variable shows the version of the memory allocator library detected by PerconaFT.

This variable shows the `mmap` threshold in PerconaFT, anything larger than this gets `mmap'ed`.

This variable shows the number of threads that are currently blocked because they are attempting to write to a full disk. This is normally zero. If this value is non-zero, then a warning will appear in the `disk free space` field.

This variable shows the total time, in microseconds, used to `fsync` to disk.

This variable shows the total number of times the database has flushed the operating system's file buffers to disk.

This variable shows the total time, in microseconds, used to `fsync` to dis k when the operation required more than one second.

This variable shows the total number of times the database has flushed the operating system's file buffers to disk and this operation required more than one second.

CHAPTER

SEVENTYSIX

TOKUDB FRACTAL TREE INDEXING

Fractal Tree indexing is the technology behind TokuDB and is protected by multiple patents. This type of index enhances the tradional B-tree data structure used in other database engines, and optimizes performance for modern hardware and data sets.

76.1 Background

The B-tree data structure was optimized for large blocks of data but the performance is limited by I/O bandwidth. The size of a production database generally exceeds available main memory. Most leaves in a tree are stored on disk, not in RAM. If a leaf is not in main memory inserting information requires a disk I/O operation. Continually adding RAM to keep pace with data's growth is too expensive.

76.2 Buffers

Like a B-tree structure, a fractal tree index is a tree data structure, but each node has buffers that allow messages to be stored. Insertions, deletions, and updates are inserted into the buffers as messages. Buffers let each disk operation be more efficient by writing large amounts of data. Buffers also avoid the common B-tree scenario when disk writes change only a small amount of data.

In fractal tree indexes, non-leaf (internal) nodes have child nodes. The number of child nodes is variable and based on a pre-defined range. When data is inserted or deleted from a node, the number of child nodes changes. Internal nodes may join or split to maintain the defined range. When the buffer is full, the mesages are flushed to children nodes.

Fractal tree index data structure involves the same algorithmic complexity as B-tree queries. There is no data loss because the queries follow the path from the root to leaf and pass through all messages. A query knows the current state of data even if changes have not been propagated to the corresponding leaves.

Each message is stamped with a unique message sequence number (MSN) when the message is stored in a non-leaf node message buffer. The MSN maintains the order of messages and ensures the messages are only applied once to leaf nodes when the leaf node is updated by messages.

Buffers are also serialized to disk, messages in internal nodes are not lost in the case of a crash or outage. If a write happened after a checkpoint, but before a crash, recovery replays the operation from the log.

CHAPTER

SEVENTYSEVEN

TOKUDB PERFORMANCE SCHEMA INTEGRATION

In *Percona Server for MySQL* **:rn:‘5.6.38-83.0‘** has implemented *TokuDB* integration with Performance Schema

This integration can be used for profiling additional *TokuDB* operations.

TokuDB instruments available in Performance Schema can be seen in **:table:‘PERFORMANCE_SCHEMA.SETUP_INSTRUMENTS‘** table:

```
mysql> SELECT * FROM performance_schema.setup_instruments WHERE NAME LIKE "%/fti/%";
+-----------------------------------------------------------------+---------+-------+
| NAME                                                            | ENABLED | TIMED |
+-----------------------------------------------------------------+---------+-------+
| wait/synch/mutex/fti/kibbutz_mutex                              | NO      | NO    |
| wait/synch/mutex/fti/minicron_p_mutex                           | NO      | NO    |
| wait/synch/mutex/fti/queue_result_mutex                         | NO      | NO    |
| wait/synch/mutex/fti/tpool_lock_mutex                           | NO      | NO    |
| wait/synch/mutex/fti/workset_lock_mutex                         | NO      | NO    |
| wait/synch/mutex/fti/bjm_jobs_lock_mutex                        | NO      | NO    |
| wait/synch/mutex/fti/log_internal_lock_mutex                    | NO      | NO    |
| wait/synch/mutex/fti/cachetable_ev_thread_lock_mutex            | NO      | NO    |
| wait/synch/mutex/fti/cachetable_disk_nb_mutex                   | NO      | NO    |
| wait/synch/mutex/fti/safe_file_size_lock_mutex                  | NO      | NO    |
| wait/synch/mutex/fti/cachetable_m_mutex_key                     | NO      | NO    |
| wait/synch/mutex/fti/checkpoint_safe_mutex                      | NO      | NO    |
| wait/synch/mutex/fti/ft_ref_lock_mutex                          | NO      | NO    |
| wait/synch/mutex/fti/ft_open_close_lock_mutex                   | NO      | NO    |
| wait/synch/mutex/fti/loader_error_mutex                         | NO      | NO    |
| wait/synch/mutex/fti/bfs_mutex                                  | NO      | NO    |
| wait/synch/mutex/fti/loader_bl_mutex                            | NO      | NO    |
| wait/synch/mutex/fti/loader_fi_lock_mutex                       | NO      | NO    |
| wait/synch/mutex/fti/loader_out_mutex                           | NO      | NO    |
| wait/synch/mutex/fti/result_output_condition_lock_mutex         | NO      | NO    |
| wait/synch/mutex/fti/block_table_mutex                          | NO      | NO    |
| wait/synch/mutex/fti/rollback_log_node_cache_mutex              | NO      | NO    |
| wait/synch/mutex/fti/txn_lock_mutex                             | NO      | NO    |
| wait/synch/mutex/fti/txn_state_lock_mutex                       | NO      | NO    |
| wait/synch/mutex/fti/txn_child_manager_mutex                    | NO      | NO    |
| wait/synch/mutex/fti/txn_manager_lock_mutex                     | NO      | NO    |
| wait/synch/mutex/fti/treenode_mutex                             | NO      | NO    |
| wait/synch/mutex/fti/locktree_request_info_mutex                | NO      | NO    |
| wait/synch/mutex/fti/locktree_request_info_retry_mutex_key      | NO      | NO    |
| wait/synch/mutex/fti/manager_mutex                              | NO      | NO    |
| wait/synch/mutex/fti/manager_escalation_mutex                   | NO      | NO    |
| wait/synch/mutex/fti/db_txn_struct_i_txn_mutex                  | NO      | NO    |
| wait/synch/mutex/fti/manager_escalator_mutex                    | NO      | NO    |
| wait/synch/mutex/fti/indexer_i_indexer_lock_mutex               | NO      | NO    |
```

```
| wait/synch/mutex/fti/indexer_i_indexer_estimate_lock_mutex | NO       | NO    |
| wait/synch/mutex/fti/fti_probe_1                           | NO       | NO    |
| wait/synch/rwlock/fti/multi_operation_lock                 | NO       | NO    |
| wait/synch/rwlock/fti/low_priority_multi_operation_lock    | NO       | NO    |
| wait/synch/rwlock/fti/cachetable_m_list_lock               | NO       | NO    |
| wait/synch/rwlock/fti/cachetable_m_pending_lock_expensive  | NO       | NO    |
| wait/synch/rwlock/fti/cachetable_m_pending_lock_cheap      | NO       | NO    |
| wait/synch/rwlock/fti/cachetable_m_lock                    | NO       | NO    |
| wait/synch/rwlock/fti/result_i_open_dbs_rwlock             | NO       | NO    |
| wait/synch/rwlock/fti/checkpoint_safe_rwlock               | NO       | NO    |
| wait/synch/rwlock/fti/cachetable_value                     | NO       | NO    |
| wait/synch/rwlock/fti/safe_file_size_lock_rwlock           | NO       | NO    |
| wait/synch/rwlock/fti/cachetable_disk_nb_rwlock            | NO       | NO    |
| wait/synch/cond/fti/result_state_cond                      | NO       | NO    |
| wait/synch/cond/fti/bjm_jobs_wait                          | NO       | NO    |
| wait/synch/cond/fti/cachetable_p_refcount_wait             | NO       | NO    |
| wait/synch/cond/fti/cachetable_m_flow_control_cond         | NO       | NO    |
| wait/synch/cond/fti/cachetable_m_ev_thread_cond            | NO       | NO    |
| wait/synch/cond/fti/bfs_cond                               | NO       | NO    |
| wait/synch/cond/fti/result_output_condition                | NO       | NO    |
| wait/synch/cond/fti/manager_m_escalator_done               | NO       | NO    |
| wait/synch/cond/fti/lock_request_m_wait_cond               | NO       | NO    |
| wait/synch/cond/fti/queue_result_cond                      | NO       | NO    |
| wait/synch/cond/fti/ws_worker_wait                         | NO       | NO    |
| wait/synch/cond/fti/rwlock_wait_read                       | NO       | NO    |
| wait/synch/cond/fti/rwlock_wait_write                      | NO       | NO    |
| wait/synch/cond/fti/rwlock_cond                            | NO       | NO    |
| wait/synch/cond/fti/tp_thread_wait                         | NO       | NO    |
| wait/synch/cond/fti/tp_pool_wait_free                      | NO       | NO    |
| wait/synch/cond/fti/frwlock_m_wait_read                    | NO       | NO    |
| wait/synch/cond/fti/kibbutz_k_cond                         | NO       | NO    |
| wait/synch/cond/fti/minicron_p_condvar                     | NO       | NO    |
| wait/synch/cond/fti/locktree_request_info_retry_cv_key     | NO       | NO    |
| wait/io/file/fti/tokudb_data_file                          | YES      | YES   |
| wait/io/file/fti/tokudb_load_file                          | YES      | YES   |
| wait/io/file/fti/tokudb_tmp_file                           | YES      | YES   |
| wait/io/file/fti/tokudb_log_file                           | YES      | YES   |
+------------------------------------------------------------+----------+-------+
```

For *TokuDB*-related objects, following clauses can be used when querying Performance Schema tables:

- `WHERE EVENT_NAME LIKE '%fti%'` or
- `WHERE NAME LIKE '%fti%'`

For example, to get the information about *TokuDB* related events you can query **:table:'PERFORMANCE_SCHEMA.events_waits_summary_global_by_event_name'** like:

```
mysql> SELECT * FROM performance_schema.events_waits_summary_global_by_event_name␣
↪WHERE EVENT_NAME LIKE '%fti%';

+----------------------------------------+-----------+---------------+-------------
↪---+---------------+----------------+
| EVENT_NAME                             | COUNT_STAR | SUM_TIMER_WAIT | MIN_TIMER_
↪WAIT | AVG_TIMER_WAIT | MAX_TIMER_WAIT |
+----------------------------------------+-----------+---------------+-------------
↪---+---------------+----------------+
| wait/synch/mutex/fti/kibbutz_mutex     |          0 |              0 |           ␣
↪ 0 |             0 |              0 |
```

```
| wait/synch/mutex/fti/minicron_p_mutex    |          0 |               0 |
↪ 0 |              0 |              0 |
| wait/synch/mutex/fti/queue_result_mutex |          0 |               0 |
↪ 0 |              0 |              0 |
| wait/synch/mutex/fti/tpool_lock_mutex    |          0 |               0 |
↪ 0 |              0 |              0 |
| wait/synch/mutex/fti/workset_lock_mutex |          0 |               0 |
↪ 0 |              0 |              0 |
...
| wait/io/file/fti/tokudb_data_file        |         30 |       179862410 |
↪ 0 |        5995080 |       68488420 |
| wait/io/file/fti/tokudb_load_file        |          0 |               0 |
↪ 0 |              0 |              0 |
| wait/io/file/fti/tokudb_tmp_file         |          0 |               0 |
↪ 0 |              0 |              0 |
| wait/io/file/fti/tokudb_log_file         |       1367 |  2925647870145 |
↪ 0 |     2140195785 |    12013357720 |
+------------------------------------------+------------+-----------------+-------------
↪---+----------------+----------------+
71 rows in set (0.02 sec)
```

Part X

Reference

CHAPTER

SEVENTYEIGHT

LIST OF UPSTREAM *MYSQL* BUGS FIXED IN *PERCONA SERVER FOR MYSQL* 5.6

Upstream bug #94121 - Enable hardware CRC32 under Valgrind
JIRA bug #5388
Upstream state Verified (checked on 2019-05-15)
Fix Released **:rn:‘5.6.44-85.0‘**
Upstream fix N/A

Upstream bug #93917 - Wrong binlog entry for BLOB on a blackhole intermediary master
JIRA bug #5353
Upstream state Verified (checked on 2019-05-15)
Fix Released **:rn:‘5.6.44-85.0‘**
Upstream fix N/A

Upstream bug #93708 - Page Cleaner will sleep for long time if clock changes
JIRA bug #5221
Upstream state Verified (checked on 2019-05-15)
Fix Released **:rn:‘5.6.44-85.0‘**
Upstream fix N/A

Upstream bug #93649 - STOP SLAVE SQL_THREAD deadlocks if done while holding LOCK INSTANCE ...
JIRA bug #4758
Upstream state Closed
Fix Released **:rn:‘5.6.43-84.3‘**
Upstream fix N/A

Upstream bug #92809 - Inconsistent ResultSet for different Execution Plans
JIRA bug #4907
Upstream State Closed
Fix Released **:rn:‘5.6.43-84.3‘**
Upstream fix N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream bug #92227 - SQL injection on slave due to non-quoting in binlogged ROLLBACK TO ... **JIRA bug** #4791 **Upstream state** N/A **Fix Released** **:rn:‘5.6.43-84.3‘** **Upstream fix** N/A
Upstream bug #90351 - GLOBAL STATUS variables drift after rollback **JIRA bug** #3951 **Upstream State** Closed **Fix Released** **:rn:‘5.6.40-84.0‘** **Upstream Fix** 5.6.44
Upstream Bug #90264 - Some file operations in mf_iocache2.c are not instrumented **JIRA bug** #3937 **Upstream State** Closed **Fix Released** **:rn:‘5.6.40-84.0‘** **Upstream Fix** N/A
Upstream Bug #90238 - Comparison of uninitailized memory in log_in_use **JIRA bug** #3925 **Upstream State** Closed **Fix Released** **:rn:‘5.6.40-84.0‘** **Upstream Fix** N/A
Upstream Bug #90111 - Incorrect enum comparisons **JIRA bug** #3893 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.40-84.0‘** **Upstream Fix** N/A
Upstream Bug #89766 - a typo in *cmake/plugin.cmake* prevents *MYSQL_SERVER* to be defined ... **JIRA bug** #3871 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.40-84.0‘** **Upstream Fix** N/A
Upstream bug #88720 - Inconsistent and unsafe FLUSH behavior in terms of replication **JIRA bug** #1827 **Upstream state** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.44-85.0‘** **Upstream fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #88057 - Intermediary slave does not log master changes with... **JIRA bug** #1119 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.39-83.1'** **Upstream Fix** N/A
Upstream Bug #87065 - Release lock on table statistics after query plan created **JIRA bug** #2503 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.38-83.0'** **Upstream Fix** N/A
Upstream Bug #86260 - Assert on KILL'ing a stored routine invocation **JIRA bug** #1091 **Upstream State** Closed **Fix Released** **:rn:'5.6.36-82.1'** **Upstream Fix** 5.6.40
Upstream Bug #86209 - audit plugin + MB collation connection + PREPARE stmt parse error crash... **JIRA bug** #1089 **Upstream State** N/A **Fix Released** **:rn:'5.6.36-82.0'** **Upstream Fix** N/A
Upstream Bug #85838 - rpl_diff.inc in 5.7 does not compare data from different servers **JIRA bug** #2257 **Upstream State** Closed **Fix Released** **:rn:'5.6.36-82.0'** **Upstream Fix** N/A
Upstream Bug #85678 - field-t deletes Fake_TABLE objects through base TABLE pointer w/o ... **JIRA bug** #2253 **Upstream State** Closed **Fix Released** **:rn:'5.6.36-82.0'** **Upstream Fix** 5.6.37
Upstream Bug #85671 - segfault-t failing under recent AddressSanitizer **JIRA bug** #2252 **Upstream State** Closed **Fix Released** **:rn:'5.6.36-82.0'** **Upstream Fix** 5.6.37

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #85258 - DROP TEMPORARY TABLE creates a transaction in binary log on read only... **JIRA bug** #1785 **Upstream State** Closed **Fix Released** **:rn:‘5.6.36-82.0‘** **Upstream Fix** N/A
Upstream Bug #84415 - slave don’t report Seconds_Behind_Master when running ... **JIRA bug** #1770 **Upstream State** Closed **Fix Released** **:rn:‘5.6.36-82.0‘** **Upstream Fix** N/A
Upstream Bug #84366 - InnoDB index dives do not detect concurrent tree changes, return bogus... **JIRA bug** #1743 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.35-80.0‘** **Upstream Fix** N/A
Upstream Bug #83814 - Add support for OpenSSL 1.1 **JIRA bug** #1105 **Upstream State** Closed **Fix Released** **:rn:‘5.6.36-82.1‘** **Upstream Fix** N/A
Upstream Bug #83648 - Assertion failure in thread x in file fts0que.cc line 3659 **JIRA bug** #1023 **Upstream State** N/A **Fix Released** **:rn:‘5.6.35-80.1‘** **Upstream Fix** N/A
Upstream Bug #83232 - replication breaks after bug #74145 happens in master **JIRA bug** #1017 **Upstream State** Closed **Fix Released** **:rn:‘5.6.42-84.2‘** **Upstream Fix** N/A
Upstream Bug #83124 - Bug 81657 fix merge to 5.6 broken **JIRA bug** #1750 **Upstream State** Closed **Fix Released** **:rn:‘5.6.33-79.0‘** **Upstream Fix** 5.6.35

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #83073 - GCC 5 and 6 miscompile mach_parse_compressed **JIRA bug** #1745 **Upstream State** Closed **Fix Released** **:rn:‘5.6.33-79.0‘** **Upstream Fix** 5.6.35
Upstream Bug #83003 - Using temporary tables on slaves increases GTID sequence number **JIRA bug** #964 **Upstream State** Closed **Fix Released** **:rn:‘5.6.35-80.0‘** **Upstream Fix** N/A
Upstream Bug #82980 - Multi-threaded slave leaks worker threads in case of thread create ... **JIRA bug** #2193 **Upstream State** Closed **Fix Released** **:rn:‘5.6.33-79.0‘** **Upstream Fix** 5.6.38
Upstream Bug #82935 - Cipher ECDHE-RSA-AES128-GCM-SHA256 listed in man/Ssl_cipher_list, not ... **JIRA bug** #1737 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.33-79.0‘** **Upstream Fix** N/A
Upstream Bug #82886 - Server may crash due to a glibc bug in handling short-lived detached ... **JIRA bug** #1006 **Upstream State** Closed **Fix Released** **:rn:‘5.6.33-79.0‘** **Upstream Fix** 5.6.35
Upstream Bug #82798 - Small buffer pools might be too small for rseg init during crash recovery **JIRA bug** #3525 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.33-79.0‘** **Upstream Fix** N/A
Upstream Bug #82019 - Is client library supposed to retry EINTR indefinitely or not **JIRA bug** #1720 **Upstream State** Closed **Fix Released** **:rn:‘5.6.32-78.0‘** **Upstream Fix** 5.6.33

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #81714 - mysqldump get_view_structure does not free MYSQL_RES in one error path **JIRA bug** #2152 **Upstream State** Closed **Fix Released** **:rn:‘5.6.31-77.0‘** **Upstream Fix** 5.6.38
Upstream Bug #81675 - mysqlbinlog does not free the existing connection before opening new ... **JIRA bug** #1718 **Upstream State** Closed **Fix Released** **:rn:‘5.6.31-77.0‘** **Upstream Fix** 5.6.33
Upstream Bug #81674 - LeakSanitizer-enabled build fails to bootstrap server for MTR **JIRA bug** #3486 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.32-78.0‘** **Upstream Fix** N/A
Upstream Bug #81657 - DBUG_PRINT in THD::decide_logging_format prints incorrectly, access ... **JIRA bug** #2150 **Upstream State** Closed **Fix Released** **:rn:‘5.6.31-77.0‘** **Upstream Fix** N/A
Upstream Bug #81467 - innodb_fts.sync_block test unstable due to slow query log nondeterminism **JIRA bug** #2232 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.35-80.1‘** **Upstream Fix** N/A
Upstream Bug #80014 - mysql build fails, memory leak in gen_lex_hash, clang address sanitizer **JIRA bug** #3433 **Upstream State** Closed **Fix Released** **:rn:‘5.6.30-76.3‘** **Upstream Fix** 5.6.35
Upstream Bug #79703 - Spin rounds per wait will be negative in InnoDB status if spin waits ... **JIRA bug** #1684 **Upstream State** Closed **Fix Released** **:rn:‘5.6.28-76.1‘** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #79610 - Failed DROP DATABASE due FK constraint on master breaks slave **JIRA bug** #1683 **Upstream State** Closed **Fix Released** **:rn:'5.6.32-78.0'** **Upstream Fix** N/A
Upstream Bug #79185 - Innodb freeze running REPLACE statements **JIRA bug** #945 **Upstream State** Closed **Fix Released** **:rn:'5.6.27-76.0'** **Upstream Fix** 5.6.30
Upstream Bug #79117 - "change_user" command should be aware of preceding "error" command **JIRA bug** #659 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #78223 - memory leak in mysqlbinlog **JIRA bug** #3440 **Upstream State** Closed **Fix Released** **:rn:'5.6.31-77.0'** **Upstream Fix** N/A
Upstream Bug #78050 - Crash on when XA functions activated by a storage engine **JIRA bug** #742 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.16-64.0'** **Upstream Fix** N/A
Upstream Bug #77684 - DROP TABLE IF EXISTS may brake replication if slave has replication filter **JIRA bug** #1639 **Upstream State** Closed **Fix Released** **:rn:'5.6.26-74.0'** **Upstream Fix** 5.6.30
Upstream Bug #77637 - mysql 5.6.25 compiled warning **JIRA bug** #3632 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.39-83.1'** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #77591 - ALTER TABLE does not allow to change NULL/NOT NULL if foreign key exists **JIRA bug** #1635 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.26-74.0‘** **Upstream Fix** N/A
Upstream Bug #77399 - Deadlocks missed by INFORMATION_SCHEMA.INNODB_METRICS lock_deadlocks ... **JIRA bug** #1632 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.31-77.0‘** **Upstream Fix** N/A
Upstream Bug #77275 - Newest RHEL/CentOS openssl update breaks mysql DHE ciphers **JIRA bug** #906 **Upstream State** Closed **Fix Released** **:rn:‘5.6.25-73.0‘** **Upstream Fix** 5.6.26
Upstream Bug #76927 - Duplicate UK values in READ-COMMITTED (again) **JIRA bug** #1494 **Upstream State** Closed **Fix Released** **:rn:‘5.6.25-73.0‘** **Upstream Fix** 5.6.27
Upstream Bug #76418 - Server crashes when querying partitioning table MySQL_5.7.14 **JIRA bug** #1050 **Upstream State** N/A **Fix Released** **:rn:‘5.6.36-82.1‘** **Upstream Fix** N/A
Upstream Bug #76349 - memory leak in add_derived_key() **JIRA bug** #826 **Upstream State** Closed **Fix Released** **:rn:‘5.6.24-72.2‘** **Upstream Fix** 5.6.27
Upstream Bug #76142 - InnoDB tablespace import fails when importing table w/ different datadir **JIRA bug** #1697 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.31-77.0‘** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #75642 - Extend valid range of dummy certificates ni mysql-test/std_data **JIRA bug** #1605 **Upstream State** Closed **Fix Released** **:rn:‘5.6.22-72.0‘** **Upstream Fix** 5.6.23
Upstream Bug #75595 - Compute InnoDB redo log block checksums faster **Launchpad BP** https://blueprints.launchpad.net/percona-server/+spec/more-efficient-log-block-checksums **Upstream State** Closed **Fix Released** 5.6.14-62.0 **Upstream Fix** 5.6.25
Upstream Bug #75534 - Solve buffer pool mutex contention by splitting it **JIRA bug** *Improved Buffer Pool Scalability* **Upstream State** Closed **Fix Released** **:rn:‘5.6.13-60.6‘** **Upstream Fix** N/A
Upstream Bug #75480 - Selecting wrong pos with SHOW BINLOG EVENTS causes a potentially ... **JIRA bug** #1600 **Upstream State** N/A **Fix Released** **:rn:‘5.6.25-73.0‘** **Upstream Fix** N/A
Upstream Bug #75311 - Error for SSL cipher is unhelpful **JIRA bug** #1779 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.35-80.1‘** **Upstream Fix** N/A
Upstream Bug #75235 - Optimize ibuf merge when reading a page from disk **JIRA bug** #2484 **Upstream State** Closed **Fix Released** **:rn:‘5.6.33-79.0‘** **Upstream Fix** N/A
Upstream Bug #75189 - engines suite tests depending on InnoDB implementation details **JIRA bug** #2103 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.22-71.0‘** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #74987 - mtr failure on Ubuntu Utopic, mysqlhotcopy fails with wrong error(255) **JIRA bug** #2102 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.22-71.0‘** **Upstream Fix** N/A
Upstream Bug #74842 - Incorrect attribute((nonnull)) for btr_cur_ins_lock_and_undo callees **JIRA bug** #385 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.21-70.1‘** **Upstream Fix** N/A
Upstream Bug #74644 - A query on empty table with BLOBs may crash server **JIRA bug** #176 **Upstream State** N/A **Fix Released** **:rn:‘5.6.22-71.0‘** **Upstream Fix** N/A
Upstream Bug #74637 - make dirty page flushing more adaptive **Launchpad BP** Split LRU ... **Upstream State** Verified (checked on 2019-05-15) **Fix Released** 5.6.16-64.0 **Upstream Fix** N/A
Upstream Bug #74440 - mysql_install_db not handling mysqld startup failure **JIRA bug** #1553 **Upstream State** Won’t fix **Fix Released** **:rn:‘5.6.21-70.0‘** **Upstream Fix** N/A
Upstream Bug #73979 - wrong stack size calculation leads to stack overflow in pinbox allocator **JIRA bug** #807 **Upstream State** Closed **Fix Released** **:rn:‘5.6.22-71.0‘** **Upstream Fix** N/A
Upstream Bug #73736 - Missing testcase sync in rpl_err_ignoredtable **JIRA bug** #2081 **Upstream State** Closed **Fix Released** **:rn:‘5.6.21-69.0‘** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #73689 - Zero can be a valid InnoDB checksum, but validation will fail later **JIRA bug** #PS-909 **Upstream State** Closed **Fix Released** **:rn:'5.6.25-73.0'** **Upstream Fix** 5.6.22
Upstream Bug #73418 - Add –manual-lldb option to mysql-test-run.pl **JIRA bug** #2448 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.20-68.0'** **Upstream Fix** N/A
Upstream Bug #73066 - Replication stall with multi-threaded replication **JIRA bug** #1511 **Upstream State** Closed **Fix Released** **:rn:'5.6.21-70.0'** **Upstream Fix** N/A
Upstream Bug #72615 - MTR –mysqld=–default-storage-engine=foo incompatible w/ dynamically... **JIRA bug** #2071 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.17-66.0'** **Upstream Fix** N/A
Upstream Bug #72475 - Binlog events with binlog_format=MIXED are unconditionally logged in ROW.. **JIRA bug** #151 **Upstream State** Closed **Fix Released** **:rn:'5.6.21-70.1'** **Upstream Fix** N/A
Upstream Bug #72466 - More memory overhead per page in the InnoDB buffer pool **JIRA bug** #1689 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.30-76.3'** **Upstream Fix** N/A
Upstream Bug #72457 - Replication with no tmpdir space can break replication **JIRA bug** #1107 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.42-84.2'** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #72163 - Rev 5774 broke rpl_plugin_load **JIRA bug** #2068 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.17-65.0‘** **Upstream Fix** N/A
Upstream Bug #72108 - Hard to read history file **JIRA bug** #2066 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.24-72.2‘** **Upstream Fix** N/A
Upstream Bug #71988 - page_cleaner: aggressive background flushing **JIRA bug** #1437 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.16-64.0‘** **Upstream Fix** N/A
Upstream Bug #71761 - ANALYZE TABLE should remove its table from background stat processing ... **JIRA bug** #1749 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.33-79.0‘** **Upstream Fix** N/A
Upstream Bug #71759 - memory leak with string thread variable that set memalloc flag **JIRA bug** #1006 **Upstream State** Closed **Fix Released** **:rn:‘5.6.33-79.0‘** **Upstream Fix** N/A
Upstream Bug #71708 - 70768 fix perf regression: high rate of RW lock creation and destruction **JIRA bug** #1474 **Upstream State** Closed **Fix Released** **:rn:‘5.6.16-64.0‘** **Upstream Fix** 5.6.19
Upstream Bug #71624 - printf size_t results in a fatal warning in 32-bit debug builds **JIRA bug** #760 **Upstream State** Can’t Repeat **Fix Released** **:rn:‘5.6.16-64.0‘** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #71603 - file name is not escaped in binlog for LOAD DATA INFILE statement **JIRA bug** #3092 **Upstream State** N/A **Fix Released** **:rn:‘5.6.11-60.3‘** **Upstream Fix** N/A
Upstream Bug #71411 - buf_flush_LRU() does not return correct number in case of compressed pages **JIRA bug** #2430 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.13-61.0‘** **Upstream Fix** N/A
Upstream Bug #71374 - Slave IO thread won’t attempt auto reconnect to the master/error-code 1159 **JIRA bug** #1470 **Upstream State** N/A **Fix Released** **:rn:‘5.6.16-64.1‘** **Upstream Fix** N/A
Upstream Bug #71270 - Failures to end bulk insert for partitioned tables handled incorrectly **JIRA bug** #700 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.16-64.0‘** **Upstream Fix** N/A
Upstream Bug #71250 - Bison 3 breaks mysql build **JIRA bug** #376 **Upstream State** Closed **Fix Released** **:rn:‘5.6.17-65.0‘** **Upstream Fix** N/A
Upstream Bug #71217 - Threadpool - add thd_wait_begin/thd_wait_end to the network IO functions **JIRA bug** #1343 **Upstream State** Open (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.15-63.0‘** **Upstream Fix** N/A
Upstream Bug #71183 - os_file_fsync() should handle fsync() returning EINTR **JIRA bug** #1461 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.11-60.3‘** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #71094 - ssl.cmake related warnings **JIRA bug** #2058 **Upstream State** Closed **Fix Released** **:rn:‘5.6.16-64.0‘** **Upstream Fix** N/A
Upstream Bug #71092 - InnoDB FTS introduced new mutex sync level in 5.6.15, broke UNIV_SYNC ... **JIRA bug** #1393 **Upstream State** Closed **Fix Released** **:rn:‘5.6.15-63.0‘** **Upstream Fix** 5.6.12
Upstream Bug #71091 - CSV engine does not properly process `""`, in quotes **JIRA bug** #153 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.21-70.0‘** **Upstream Fix** N/A
Upstream Bug #71089 - CMake warning when generating Makefile **JIRA bug** #2059 **Upstream State** Closed **Fix Released** **:rn:‘5.6.16-64.0‘** **Upstream Fix** 5.6.18
Upstream Bug #70860 - –tc-heuristic-recover option values are broken **JIRA bug** #1514 **Upstream State** Closed **Fix Released** **:rn:‘5.6.20-68.0‘** **Upstream Fix** N/A
Upstream Bug #70854 - Tc_log_page_size should be unflushable or server crashes if 2 XA SEs ... **JIRA bug** #743 **Upstream State** Closed **Fix Released** **:rn:‘5.6.16-64.0‘** **Upstream Fix** N/A
Upstream Bug #70500 - Page cleaner should perform LRU flushing regardless of server activity **JIRA bug** #1428 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.13-61.0‘** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #70490 - Suppression is too strict on some systems **JIRA bug** #2038 **Upstream State** Closed **Fix Released** **:rn:‘5.6.13-61.0‘** **Upstream Fix** N/A
Upstream Bug #70489 - Crash when using AES_ENCRYPT on empty string **JIRA bug** #689 **Upstream State** Unsupported **Fix Released** **:rn:‘5.6.13-61.0‘** **Upstream Fix** N/A
Upstream Bug #70453 - Add hard timeouts to page cleaner flushes **JIRA bug** #2431 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.13-61.0‘** **Upstream Fix** N/A
Upstream Bug #70417 - rw_lock_x_lock_func_nowait() calls os_thread_get_curr_id() mostly ... **JIRA bug** #2429 **Upstream State** Closed **Fix Released** **:rn:‘5.6.13-61.0‘** **Upstream Fix** 5.6.16
Upstream Bug #70277 - last argument of LOAD DATA ... SET ... statement repeated twice in binlog **JIRA bug** #3020 **Upstream State** Closed **Fix Released** **:rn:‘5.6.11-60.3‘** **Upstream Fix** 5.6.15
Upstream Bug #70228 - Is buf_LRU_free_page() really supposed to make non-zip block sticky at ... **JIRA bug** #1415 **Upstream State** Closed **Fix Released** **:rn:‘5.6.13-60.6‘** **Upstream Fix** N/A
Upstream Bug #70216 - Unnecessary overhead from persistent adaptive hash index latches **JIRA bug** #715 **Upstream State** Closed **Fix Released** **:rn:‘5.6.13-60.6‘** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #69991 - MySQL client is broken without readline **JIRA bug** #1467 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:‘5.6.24-72.2‘** **Upstream Fix** N/A
Upstream Bug #69856 - mysql_install_db does not function properly in 5.6 for debug builds **JIRA bug** #359 **Upstream State** Won’t fix **Fix Released** **:rn:‘5.6.12-60.4‘** **Upstream Fix** N/A
Upstream Bug #69639 - mysql failed to build with dtrace Sun D 1.11 **JIRA bug** #1392 **Upstream State** Unsupported **Fix Released** **:rn:‘5.6.13-60.5‘** **Upstream Fix** N/A
Upstream Bug #69617 - 5.6.12 removed UNIV_SYNC_DEBUG from UNIV_DEBUG **JIRA bug** #1411 **Upstream State** Closed **Fix Released** **:rn:‘5.6.13-60.6‘** **Upstream Fix** 5.6.16
Upstream Bug #69524 - Some tests for table cache variables fail if open files limit is too low **JIRA bug** #96 **Upstream State** Closed **Fix Released** **:rn:‘5.6.12-60.4‘** **Upstream Fix** N/A
Upstream Bug #69396 - Can’t set query_cache_type to 0 when it is already 0 **JIRA bug** #3563 **Upstream State** Closed **Fix Released** **:rn:‘5.6.33-79.0‘** **Upstream Fix** N/A
Upstream Bug #69265 - -DBUILD_CONFIG=mysql_release -DWITH_DEBUG=ON fails 4 and skips 27 MTR ... **JIRA bug** #1345 **Upstream State** Closed **Fix Released** **:rn:‘5.6.11-60.3‘** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #69258 - does buf_LRU_buf_pool_running_out need to lock buffer pool mutexes **JIRA bug** #1414 **Upstream State** Not a bug **Fix Released** **:rn:'5.6.13-60.6'** **Upstream Fix** N/A
Upstream Bug #69252 - All the parts.partition_max* tests are broken with MTR –parallel **JIRA bug** #1364 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** 5.6.15
Upstream Bug #69179 - accessing information_schema.partitions causes plans to change **JIRA bug** #680 **Upstream State** Duplicate **Fix Released** **:rn:'5.6.13-60.5'** **Upstream Fix** 5.6.14
Upstream Bug #69170 - buf_flush_LRU is lazy **JIRA bug** #2430 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.13-61.0'** **Upstream Fix** N/A
Upstream Bug #69146 - Optimization in buf_pool_get_oldest_modification if srv_buf_pool_instances **JIRA bug** #2418 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.5-60.0'** **Upstream Fix** N/A
Upstream Bug #69124 - Incorrect truncation of long SET expression in LOAD DATA can cause SQL ... **JIRA bug** #663 **Upstream State** N/A **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #69059 - GTIDs lack a reasonable deployment strategy **Launchpad BP** GTID deploy... **Upstream State** Closed **Fix Released** 5.6.22-72.0 **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #68999 - SSL_OP_NO_COMPRESSION not defined **JIRA bug** #362 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** 5.6.25
Upstream Bug #68970 - fsp_reserve_free_extents switches from small to big tblspace handling ... **JIRA bug** #656 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #68845 - Unnecessary log_sys->mutex reacquisition in mtr_log_reserve_and_write() **JIRA bug** #1347 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #68800 - client doesn't read plugin-dir from my.cnf set by MYSQL_READ_DEFAULT_FILE **JIRA bug** #82 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** 5.6.12
Upstream Bug #68714 - Remove literal statement digest values from perfschema tests **JIRA bug** #1340 **Upstream State** Not a bug **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #68713 - create_duplicate_weedout_tmp_table() leaves key_part_flag uninitialized **JIRA bug** #644 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #68659 - InnoDB Linux native aio should submit more i/o requests at once **JIRA bug** *Multiple page asynchronous I/O requests* **Upstream State** Analyzing (checked on 2019-05-15) **Fix Released** **:rn:'5.6.38-83.0'** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #68635 - Doc: Multiple issues with performance_schema_max_statement_classes **JIRA bug** #1339 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #68555 - thread convoys from log_checkpoint_margin with innodb_buffer_pool_inst... **JIRA bug** #2434 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.13-61.0'** **Upstream Fix** N/A
Upstream Bug #68490 - slave_max_allowed_packet Not Honored on Slave IO Connect **JIRA bug** #49 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** 5.6.12
Upstream Bug #68481 - InnoDB LRU flushing for MySQL 5.6 needs work **JIRA bug** #2432 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.13-61.0'** **Upstream Fix** N/A
Upstream Bug #68477 - Suboptimal code in skip_trailing_space() **JIRA bug** #1321 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #68476 - Suboptimal code in my_strnxfrm_simple() **JIRA bug** #1320 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #68354 - Server crashes on update/join FEDERATED + local table when only 1 local... **JIRA bug** #96 **Upstream State** N/A **Fix Released** **:rn:'5.6.12-60.4'** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #68116 - InnoDB monitor may hit an assertion error in buf_page_get_gen in debug ... **JIRA bug** #616 **Upstream State** Closed **Fix Released** **:rn:‘5.6.10-60.2‘** **Upstream Fix** 5.6.22
Upstream Bug #68052 - SSL Certificate Subject ALT Names with IPs not respected with –ssl-ver... **JIRA bug** #1076 **Upstream State** Closed **Fix Released** **:rn:‘5.6.36-82.1‘** **Upstream Fix** N/A
Upstream Bug #68045 - security vulnerability CVE-2012-4414 **JIRA bug** #348 **Upstream State** N/A **Fix Released** **:rn:‘5.6.11-60.3‘** **Upstream Fix** N/A
Upstream Bug #67974 - Server crashes in add_identifier on concurrent ALTER TABLE and SHOW ENGINE **JIRA bug** #344 **Upstream State** N/A **Fix Released** **:rn:‘5.6.11-60.3‘** **Upstream Fix** 5.6.12
Upstream Bug #67879 - Slave deadlock caused by stop slave, show slave status and global read... **Launchpad BP** *Lock-Free SHOW SLAVE STATUS* **Upstream State** Closed **Fix Released** 5.6.11-60.3 **Upstream Fix** 5.6.23
Upstream Bug #67806 - Multiple user level lock per connection **JIRA bug** *Multiple user level locks per connection* **Upstream State** Closed **Fix Released** **:rn:‘5.6.19-67.0‘** **Upstream Fix** N/A
Upstream Bug #67685 - security vulnerability CVE-2012-5611 **JIRA bug** #350 **Upstream State** N/A **Fix Released** **:rn:‘5.6.11-60.3‘** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #67504 - Duplicate error in replication with slave triggers and auto increment **JIRA bug** #34 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #66779 - innochecksum does not work with compressed tables **JIRA bug** #1302 **Upstream State** Closed **Fix Released** **:rn:'5.6.25-73.0'** **Upstream Fix** N/A
Upstream Bug #66550 - security vulnerability CVE-2012-4414 **JIRA bug** #348 **Upstream State** N/A **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #66301 - INSERT ... ON DUPLICATE KEY UPDATE + innodb_autoinc_lock_mode=1 is broken **JIRA bug** #576 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** 5.6.12
Upstream Bug #66237 - Temporary files created by binary log cache are not purged after transa... **JIRA bug** #599 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #65946 - Sid_map::Sid_map calls DBUG which may have unitialized THR_KEY_mysys and.. **JIRA bug** #585 **Upstream State** Duplicate **Fix Released** **:rn:'5.6.5-60.0'** **Upstream Fix** 5.6.15
Upstream Bug #64800 - mysqldump with –include-master-host-port putting quotes around port no. **JIRA bug** #1923 **Upstream State** Verified (checked on 2019-05-15) **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A

Continued on next page

Table 78.1 – continued from previous page

Upstream Bug #64663 - Segfault when adding indexes to InnoDB temporary tables **JIRA bug** #557 **Upstream State** N/A **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #64556 - Interrupting a query inside InnoDB causes an unrelated warning to be ... **JIRA bug** #1967 **Upstream State** Closed **Fix Released** **:rn:'5.6.13-61.0'** **Upstream Fix** 5.6.14
Upstream Bug #64432 - Bug #54330 (Broken fast index creation) was never fixed in 5.5 **JIRA bug** #544 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** N/A
Upstream Bug #63451 - atomic/x86-gcc.h:make_atomic_cas_body64 potential miscompilation bug **JIRA bug** #508 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** 5.6.16
Upstream Bug #63144 - CREATE TABLE IF NOT EXISTS metadata lock is too restrictive **JIRA bug** #40 **Upstream State** Closed **Fix Released** **:rn:'5.6.11-60.3'** **Upstream Fix** 5.6.13
Upstream Bug #63130 - CMake-based check for the presence of a system readline library is not... **JIRA bug** #1467 **Upstream State** Can't Repeat (checked on 2019-05-15) **Fix Released** **:rn:'5.6.24-72.2'** **Upstream Fix** N/A

>>>>>>> bc8f2e5d09b4369c1633f76a6fd143146ae2abe4 **|:Upstream Bug: :mysqlbug:'62856' - Check for "stack overrun" doesn't work with gcc-4.6, server crashes | |:JIRA bug: :psbug:'2795' | |:Upstream State: Closed | |:Fix Released: :rn:'5.6.11-60.3' | |:Upstream Fix: N/A | +——————————————————————————————-+ |:Upstream Bug: :mysqlbug:'62578' - mysql client aborts connection on terminal resize | |:JIRA bug: :psbug:'84' | |:Upstream State: Won't fix | |:Fix Released: :rn:'5.6.11-60.3' | |:Upstream Fix: 5.6.12 | +——————————————————————————————-+ |:Upstream Bug: :mysqlbug:'62018' - innodb adaptive hash index mutex contention | |:JIRA bug: :psbug:'1410' | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: :rn:'5.6.13-60.6' | |:Upstream Fix: N/A | +——————————————————————————————-+ |:Upstream**

**Bug: :mysqlbug:‘61595‘ - mysql-test/include/wait_for_slave_param.inc timeout logic is incorrect | |:JIRA bug: :psbug:‘485‘ | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: :rn:‘5.6.11-60.3‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: :mysqlbug:‘61180‘ - korr/store macros in my_global.h assume the argument to be a char pointer | |:JIRA bug: :psbug:‘2795‘ | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.11-60.3‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: :mysqlbug:‘61178‘ - Incorrect implementation of intersect(ulonglong) in non-optimized Bitmap..| |:JIRA bug: :psbug:‘2795‘ | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: :rn:‘5.6.11-60.3‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: :mysqlbug:‘60782‘ - Audit plugin API: no MYSQL_AUDIT_GENERAL_LOG notifications with general...| |:JIRA bug: #1369 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.17-65.0‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #60743 - typo in cmake/dtrace.cmake | |:JIRA bug: #1924 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.11-60.3‘ | |:Upstream Fix: 5.6.13 | +——————————————————————————————————-+ |:Upstream Bug: #60682 - deadlock from thd_security_context | |:JIRA bug: #1310 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.13-61.0‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #57583 - fast index create not used during “alter table foo engine=innodb” | |:JIRA bug: #2619 | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: :rn:‘5.6.5-60.0‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #57430 - query optimizer does not pick covering index for some “order by” queries | |:JIRA bug: #1587 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.22-71.0‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #56676 - ‘show slave status’ ,’show global status’ hang when ‘stop slave’ takes... | |:Launchpad BP: *Lock-Free SHOW SLAVE STATUS* | |:Upstream State: Closed | |:Fix Released: 5.6.11-60.3 | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #54814 - make BUF_READ_AHEAD_AREA a constant | |:JIRA bug: #668 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.13-60.5‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #54430 - innodb should retry partial reads/writes where errno was 0 | |:JIRA bug: #1948 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.11-60.3‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #54430 - innodb should retry partial reads/writes where errno was 0 | |:JIRA bug: #1948 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.11-60.3‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #54160 - InnoDB should retry on failed read or write, not immediately panic | |:JIRA bug: #2628 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.11-60.3‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #54127 - mysqld segfaults when built using –with-max-indexes=128 | |:JIRA bug: #2795 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.11-60.3‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #53645 - SHOW GRANTS not displaying all the applicable grants | |:JIRA bug: #1467 | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: :rn:‘5.6.23-72.1‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #53588 - Blackhole: Specified key was too long; max key length is 1000 bytes | |:JIRA bug: #1126 | |:JIRA bug: #2619 | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: :rn:‘5.6.39-83.1‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #49169 - read_view_open_now is inefficient with many concurrent sessions | |:JIRA bug: #636 and #637 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.11-60.3‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #49120 - mysqldump should have flag to delay creating indexes for innodb plugin | |:JIRA bug: #2619 | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: :rn:‘5.6.5-60.0‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #47134 - Crash on startup when XA support functions activated by a second engine | |:JIRA bug: #742 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.16-64.0‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #45679 - KILL QUERY not behaving consistently and will hang in some cases | |:JIRA bug: #3551 | |:Upstream State: Closed | |:Fix Released: :rn:‘5.6.33-79.0‘ | |:Upstream Fix: N/A | +——————————————————————————————————-+ |:Upstream Bug: #42415 - UPDATE/DELETE with LIMIT clause unsafe for SBL even with ORDER BY PK ... | |:JIRA bug: #44 |

|:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: **:rn:‘5.6.13-60.5‘** | |:Upstream Fix: N/A | +———-+ |:Upstream Bug: #41975 - Support for SSL options not included in mysqlbinlog | |:JIRA bug: #1393 | |:Upstream State: Closed | |:Fix Released: **:rn:‘5.6.15-63.0‘** | |:Upstream Fix: N/A | +——-+ |:Upstream Bug: #39833 - CREATE INDEX does full table copy on TEMPORARY table | |:JIRA bug: N/A | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: **:rn:‘5.6.10-60.2‘** | |:Upstream Fix: N/A | +———————————————————————————————————————-+ |:Upstream Bug: #35125 - Allow the ability to set the server_id for a connection for logging to... | |:Launchpad Bug: Blueprint | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: 5.6.26-74.0 | |:Upstream Fix: N/A | +——-+ |:Upstream Bug: #25007 - memory tables with dynamic rows format | |:JIRA bug: #2407 | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: **:rn:‘5.6.11-60.3‘** | |:Upstream Fix: N/A | +——-+ |:Upstream Bug: #20001 - Support for temp-tables in INFORMATION_SCHEMA | |:JIRA bug: *Temporary tables* | |:Upstream State: Verified (checked on 2019-05-15) | |:Fix Released: **:rn:‘5.6.5-60.0‘** | |:Upstream Fix: N/A | +———-+ |:Upstream Bug: #1118 - Allow multiple concurrent locks with GET_LOCK() | |:Launchpad BP: *Multiple user level locks per connection* | |:Upstream State: Closed | |:Fix Released: 5.6.19-67.0 | |:Upstream Fix: N/A | +——-+

CHAPTER

SEVENTYNINE

LIST OF VARIABLES INTRODUCED IN PERCONA SERVER 5.6

79.1 System Variables

Name	Cmd-Line	Option File	Var Scope	Dynamic
:variable:‘csv_mode‘	Yes	Yes	Both	Yes
:variable:‘enforce_storage_engine‘	Yes	Yes	Global	No
:variable:‘expand_fast_index_creation‘	Yes	No	Both	Yes
:variable:‘extra_max_connections‘	Yes	Yes	Global	Yes
:variable:‘extra_port‘	Yes	Yes	Global	No
:variable:‘gtid_deployment_step‘	Yes	Yes	Global	Yes
:variable:‘have_backup_locks‘	Yes	No	Global	No
:variable:‘have_backup_safe_binlog_info‘	Yes	No	Global	No
:variable:‘have_snapshot_cloning‘	Yes	No	Global	No
:variable:‘have_statement_timeout‘	Yes	No	Global	No
:variable:‘innodb_adaptive_hash_index_partitions‘	Yes	Yes	Global	No
:variable:‘innodb_buffer_pool_populate‘	Yes	Yes	Global	No
:variable:‘innodb_cleaner_lsn_age_factor‘	Yes	Yes	Global	Yes
:variable:‘innodb_corrupt_table_action‘	Yes	Yes	Global	Yes
:variable:‘innodb_empty_free_list_algorithm‘	Yes	Yes	Global	Yes
:variable:‘innodb_fake_changes‘	Yes	Yes	Both	Yes
:variable:‘innodb_foreground_preflush‘	Yes	Yes	Global	Yes
:variable:‘innodb_kill_idle_transaction‘	Yes	Yes	Global	Yes
:variable:‘innodb_locking_fake_changes‘	Yes	Yes	Both	Yes
:variable:‘innodb_log_arch_dir‘	Yes	Yes	Global	No
:variable:‘innodb_log_arch_expire_sec‘	Yes	Yes	Global	Yes
:variable:‘innodb_log_archive‘	Yes	Yes	Global	Yes
:variable:‘innodb_log_block_size‘	Yes	Yes	Global	No
:variable:‘innodb_log_checksum_algorithm‘	Yes	Yes	Global	Yes
:variable:‘innodb_max_bitmap_file_size‘	Yes	Yes	Global	Yes
:variable:‘innodb_max_changed_pages‘	Yes	Yes	Global	Yes
:variable:‘innodb_sched_priority_cleaner‘	Yes	Yes	Global	Yes
:variable:‘innodb_show_locks_held‘	Yes	Yes	Global	Yes
:variable:‘innodb_show_verbose_locks‘	Yes	Yes	Global	Yes
:variable:‘innodb_track_changed_pages‘	Yes	Yes	Global	No
:variable:‘innodb_use_atomic_writes‘	Yes	Yes	Global	No
:variable:‘innodb_use_global_flush_log_at_trx_commit‘	Yes	Yes	Global	Yes
:variable:‘log_slow_filter‘	Yes	Yes	Both	Yes

Continued on next page

Table 79.1 – continued from previous page

Name	Cmd-Line	Option File	Var Scope	Dynamic
:variable:‘log_slow_rate_limit‘	Yes	Yes	Both	Yes
:variable:‘log_slow_rate_type‘	Yes	Yes	Global	Yes
:variable:‘log_slow_sp_statements‘	Yes	Yes	Global	Yes
:variable:‘log_slow_verbosity‘	Yes	Yes	Both	Yes
:variable:‘log_warnings_suppress‘	Yes	Yes	Global	Yes
:variable:‘max_binlog_files‘	Yes	Yes	Global	Yes
:variable:‘max_slowlog_files‘	Yes	Yes	Global	Yes
:variable:‘max_slowlog_size‘	Yes	Yes	Global	Yes
:variable:‘max_statement_time‘	Yes	Yes	Both	Yes
:variable:‘proxy_protocol_networks‘	Yes	Yes	Global	No
:variable:‘pseudo_server_id‘	Yes	No	Session	Yes
:variable:‘query_cache_strip_comments‘	Yes	Yes	Global	Yes
:variable:‘slow_query_log_always_write_time‘	Yes	Yes	Global	Yes
:variable:‘slow_query_log_timestamp_always‘	Yes	Yes	Global	Yes
:variable:‘slow_query_log_timestamp_precision‘	Yes	Yes	Global	Yes
:variable:‘slow_query_log_use_global_control‘	Yes	Yes	Global	Yes
:variable:‘super_read_only‘	Yes	Yes	Global	Yes
:variable:‘thread_pool_high_prio_mode‘	Yes	Yes	Both	Yes
:variable:‘thread_pool_high_prio_tickets‘	Yes	Yes	Both	Yes
:variable:‘thread_pool_idle_timeout‘	Yes	Yes	Global	Yes
:variable:‘thread_pool_max_threads‘	Yes	Yes	Global	Yes
:variable:‘thread_pool_oversubscribe‘	Yes	Yes	Global	Yes
:variable:‘thread_pool_size‘	Yes	Yes	Global	Yes
:variable:‘thread_pool_stall_limit‘	Yes	Yes	Global	No
:variable:‘thread_statistics‘	Yes	Yes	Global	Yes
:variable:‘tokudb_alter_print_error‘				
:variable:‘tokudb_analyze_delete_fraction‘				
:variable:‘tokudb_analyze_in_background‘	Yes	Yes	Both	Yes
:variable:‘tokudb_analyze_mode‘	Yes	Yes	Both	Yes
:variable:‘tokudb_analyze_throttle‘	Yes	Yes	Both	Yes
:variable:‘tokudb_analyze_time‘	Yes	Yes	Both	Yes
:variable:‘tokudb_auto_analyze‘	Yes	Yes	Both	Yes
:variable:‘tokudb_block_size‘				
:variable:‘tokudb_bulk_fetch‘				
:variable:‘tokudb_cache_size‘				
:variable:‘tokudb_cachetable_pool_threads‘	Yes	Yes	Global	No
:variable:‘tokudb_cardinality_scale_percent‘	Yes	Yes	Global	Yes
:variable:‘tokudb_check_jemalloc‘				
:variable:‘tokudb_checkpoint_lock‘				
:variable:‘tokudb_checkpoint_on_flush_logs‘				
:variable:‘tokudb_checkpoint_pool_threads‘	Yes	Yes	Global	No
:variable:‘tokudb_checkpointing_period‘				
:variable:‘tokudb_cleaner_iterations‘				
:variable:‘tokudb_cleaner_period‘				
:variable:‘tokudb_client_pool_threads‘	Yes	Yes	Global	No
:variable:‘tokudb_commit_sync‘				
:variable:‘tokudb_compress_buffers_before_eviction‘	Yes	Yes	Global	No
:variable:‘tokudb_create_index_online‘				

Continued on next page

Table 79.1 – continued from previous page

Name	Cmd-Line	Option File	Var Scope	Dynamic
:variable:‘tokudb_data_dir‘				
:variable:‘tokudb_debug‘				
:variable:‘tokudb_directio‘				
:variable:‘tokudb_disable_hot_alter‘				
:variable:‘tokudb_disable_prefetching‘				
:variable:‘tokudb_disable_slow_alter‘				
:variable:‘tokudb_disable_slow_update‘				
:variable:‘tokudb_disable_slow_upsert‘				
:variable:‘tokudb_empty_scan‘				
:variable:‘tokudb_enable_partial_eviction‘	Yes	Yes	Global	No
:variable:‘tokudb_fanout‘	Yes	Yes	Both	Yes
:variable:‘tokudb_fs_reserve_percent‘				
:variable:‘tokudb_fsync_log_period‘				
:variable:‘tokudb_hide_default_row_format‘				
:variable:‘tokudb_killed_time‘				
:variable:‘tokudb_last_lock_timeout‘				
:variable:‘tokudb_load_save_space‘				
:variable:‘tokudb_loader_memory_size‘				
:variable:‘tokudb_lock_timeout‘				
:variable:‘tokudb_lock_timeout_debug‘				
:variable:‘tokudb_log_dir‘				
:variable:‘tokudb_max_lock_memory‘				
:variable:‘tokudb_optimize_index_fraction‘				
:variable:‘tokudb_optimize_index_name‘				
:variable:‘tokudb_optimize_throttle‘				
:variable:‘tokudb_pk_insert_mode‘				
:variable:‘tokudb_prelock_empty‘				
:variable:‘tokudb_read_block_size‘				
:variable:‘tokudb_read_buf_size‘				
:variable:‘tokudb_read_status_frequency‘				
:variable:‘tokudb_row_format‘				
:variable:‘tokudb_rpl_check_readonly‘				
:variable:‘tokudb_rpl_lookup_rows‘				
:variable:‘tokudb_rpl_lookup_rows_delay‘				
:variable:‘tokudb_rpl_unique_checks‘				
:variable:‘tokudb_rpl_unique_checks_delay‘				
:variable:‘tokudb_strip_frm_data‘	Yes	Yes	Global	No
:variable:‘tokudb_support_xa‘				
:variable:‘tokudb_tmp_dir‘				
:variable:‘tokudb_version‘				
:variable:‘tokudb_write_status_frequency‘				
:variable:‘userstat‘	Yes	Yes	Global	Yes
:variable:‘version_comment‘	Yes	Yes	Global	Yes
:variable:‘version_suffix‘	Yes	Yes	Global	Yes

79.2 Status Variables

Name	Var Type	Var Scope
:variable:‘Binlog_snapshot_file‘	Numeric	Global
:variable:‘Binlog_snapshot_position‘	Numeric	Global
:variable:‘Com_lock_tables_for_backup‘	Numeric	Both
:variable:‘Com_lock_binlog_for_backup‘	Numeric	Both
:variable:‘Com_purge_archived‘	Numeric	Both
:variable:‘Com_purge_archived_before_date‘	Numeric	Both
:variable:‘Com_show_client_statistics‘	Numeric	Both
:variable:‘Com_show_index_statistics‘	Numeric	Both
:variable:‘Com_show_slave_status_nolock‘	Numeric	Both
:variable:‘Com_show_table_statistics‘	Numeric	Both
:variable:‘Com_show_thread_statistics‘	Numeric	Both
:variable:‘Com_show_user_statistics‘	Numeric	Both
:variable:‘Com_unlock_binlog‘	Numeric	Both
:variable:‘Innodb_background_log_sync‘	Numeric	Global
:variable:‘Innodb_buffer_pool_pages_LRU_flushed‘	Numeric	Global
:variable:‘Innodb_buffer_pool_pages_made_not_young‘	Numeric	Global
:variable:‘Innodb_buffer_pool_pages_made_young‘	Numeric	Global
:variable:‘Innodb_buffer_pool_pages_old‘	Numeric	Global
:variable:‘Innodb_checkpoint_age‘	Numeric	Global
:variable:‘Innodb_checkpoint_max_age‘	Numeric	Global
:variable:‘Innodb_deadlocks‘	Numeric	Global
:variable:‘Innodb_history_list_length‘	Numeric	Global
:variable:‘Innodb_ibuf_discarded_delete_marks‘	Numeric	Global
:variable:‘Innodb_ibuf_discarded_deletes‘	Numeric	Global
:variable:‘Innodb_ibuf_discarded_inserts‘	Numeric	Global
:variable:‘Innodb_ibuf_free_list‘	Numeric	Global
:variable:‘Innodb_ibuf_merged_delete_marks‘	Numeric	Global
:variable:‘Innodb_ibuf_merged_deletes‘	Numeric	Global
:variable:‘Innodb_ibuf_merged_inserts‘	Numeric	Global
:variable:‘Innodb_ibuf_merges‘	Numeric	Global
:variable:‘Innodb_ibuf_segment_size‘	Numeric	Global
:variable:‘Innodb_ibuf_size‘	Numeric	Global
:variable:‘Innodb_lsn_current‘	Numeric	Global
:variable:‘Innodb_lsn_flushed‘	Numeric	Global
:variable:‘Innodb_lsn_last_checkpoint‘	Numeric	Global
:variable:‘Innodb_master_thread_active_loops‘	Numeric	Global
:variable:‘Innodb_master_thread_idle_loops‘	Numeric	Global
:variable:‘Innodb_max_trx_id‘	Numeric	Global
:variable:‘Innodb_mem_adaptive_hash‘	Numeric	Global
:variable:‘Innodb_mem_dictionary‘	Numeric	Global
:variable:‘Innodb_mem_total‘	Numeric	Global
:variable:‘Innodb_mutex_os_waits‘	Numeric	Global
:variable:‘Innodb_mutex_spin_rounds‘	Numeric	Global
:variable:‘Innodb_mutex_spin_waits‘	Numeric	Global
:variable:‘Innodb_oldest_view_low_limit_trx_id‘	Numeric	Global
:variable:‘Innodb_purge_trx_id‘	Numeric	Global
:variable:‘Innodb_purge_undo_no‘	Numeric	Global
:variable:‘Innodb_current_row_locks‘	Numeric	Global
:variable:‘Innodb_read_views_memory‘	Numeric	Global

Continued on next page

Table 79.2 – continued from previous page

Name	Var Type	Var Scope
:variable:‘Innodb_descriptors_memory‘	Numeric	Global
:variable:‘Innodb_s_lock_os_waits‘	Numeric	Global
:variable:‘Innodb_s_lock_spin_rounds‘	Numeric	Global
:variable:‘Innodb_s_lock_spin_waits‘	Numeric	Global
:variable:‘Innodb_x_lock_os_waits‘	Numeric	Global
:variable:‘Innodb_x_lock_spin_rounds‘	Numeric	Global
:variable:‘Innodb_x_lock_spin_waits‘	Numeric	Global
:variable:‘Max_statement_time_exceeded‘	Numeric	Global
:variable:‘Max_statement_time_set‘	Numeric	Global
:variable:‘Max_statement_time_set_failed‘	Numeric	Global
:variable:‘Threadpool_idle_threads‘	Numeric	Global
:variable:‘Threadpool_threads‘	Numeric	Global
:variable:‘Tokudb_DB_OPENS‘		
:variable:‘Tokudb_DB_CLOSES‘		
:variable:‘Tokudb_DB_OPEN_CURRENT‘		
:variable:‘Tokudb_DB_OPEN_MAX‘		
:variable:‘Tokudb_LEAF_ENTRY_MAX_COMMITTED_XR‘		
:variable:‘Tokudb_LEAF_ENTRY_MAX_PROVISIONAL_XR‘		
:variable:‘Tokudb_LEAF_ENTRY_EXPANDED‘		
:variable:‘Tokudb_LEAF_ENTRY_MAX_MEMSIZE‘		
:variable:‘Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_IN‘		
:variable:‘Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_OUT‘		
:variable:‘Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_IN‘		
:variable:‘Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_OUT‘		
:variable:‘Tokudb_CHECKPOINT_PERIOD‘		
:variable:‘Tokudb_CHECKPOINT_FOOTPRINT‘		
:variable:‘Tokudb_CHECKPOINT_LAST_BEGAN‘		
:variable:‘Tokudb_CHECKPOINT_LAST_COMPLETE_BEGAN‘		
:variable:‘Tokudb_CHECKPOINT_LAST_COMPLETE_ENDED‘		
:variable:‘Tokudb_CHECKPOINT_DURATION‘		
:variable:‘Tokudb_CHECKPOINT_DURATION_LAST‘		
:variable:‘Tokudb_CHECKPOINT_LAST_LSN‘		
:variable:‘Tokudb_CHECKPOINT_TAKEN‘		
:variable:‘Tokudb_CHECKPOINT_FAILED‘		
:variable:‘Tokudb_CHECKPOINT_WAITERS_NOW‘		
:variable:‘Tokudb_CHECKPOINT_WAITERS_MAX‘		
:variable:‘Tokudb_CHECKPOINT_CLIENT_WAIT_ON_MO‘		
:variable:‘Tokudb_CHECKPOINT_CLIENT_WAIT_ON_CS‘		
:variable:‘Tokudb_CHECKPOINT_BEGIN_TIME‘		
:variable:‘Tokudb_CHECKPOINT_LONG_BEGIN_TIME‘		
:variable:‘Tokudb_CHECKPOINT_LONG_BEGIN_COUNT‘		
:variable:‘Tokudb_CHECKPOINT_END_TIME‘		
:variable:‘Tokudb_CHECKPOINT_LONG_END_TIME‘		
:variable:‘Tokudb_CHECKPOINT_LONG_END_COUNT‘		
:variable:‘Tokudb_CACHETABLE_MISS‘		
:variable:‘Tokudb_CACHETABLE_MISS_TIME‘		
:variable:‘Tokudb_CACHETABLE_PREFETCHES‘		
:variable:‘Tokudb_CACHETABLE_SIZE_CURRENT‘		

Continued on next page

Table 79.2 – continued from previous page

Name	Var Type	Var Scope
:variable:‘Tokudb_CACHETABLE_SIZE_LIMIT‘		
:variable:‘Tokudb_CACHETABLE_SIZE_WRITING‘		
:variable:‘Tokudb_CACHETABLE_SIZE_NONLEAF‘		
:variable:‘Tokudb_CACHETABLE_SIZE_LEAF‘		
:variable:‘Tokudb_CACHETABLE_SIZE_ROLLBACK‘		
:variable:‘Tokudb_CACHETABLE_SIZE_CACHEPRESSURE‘		
:variable:‘Tokudb_CACHETABLE_SIZE_CLONED‘		
:variable:‘Tokudb_CACHETABLE_EVICTIONS‘		
:variable:‘Tokudb_CACHETABLE_CLEANER_EXECUTIONS‘		
:variable:‘Tokudb_CACHETABLE_CLEANER_PERIOD‘		
:variable:‘Tokudb_CACHETABLE_CLEANER_ITERATIONS‘		
:variable:‘Tokudb_CACHETABLE_WAIT_PRESSURE_COUNT‘		
:variable:‘Tokudb_CACHETABLE_WAIT_PRESSURE_TIME‘		
:variable:‘Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_COUNT‘		
:variable:‘Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_TIME‘		
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS‘		
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS_ACTIVE‘		
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_QUEUE_SIZE‘		
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_MAX_QUEUE_SIZE‘		
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_ITEMS_PROCESSED‘		
:variable:‘Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_EXECUTION_TIME‘		
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS‘		
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS_ACTIVE‘		
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_QUEUE_SIZE‘		
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_MAX_QUEUE_SIZE‘		
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_ITEMS_PROCESSED‘		
:variable:‘Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_EXECUTION_TIME‘		
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS‘		
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS_ACTIVE‘		
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_QUEUE_SIZE‘		
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_MAX_QUEUE_SIZE‘		
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_ITEMS_PROCESSED‘		
:variable:‘Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_EXECUTION_TIME‘		
:variable:‘Tokudb_LOCKTREE_MEMORY_SIZE‘		
:variable:‘Tokudb_LOCKTREE_MEMORY_SIZE_LIMIT‘		
:variable:‘Tokudb_LOCKTREE_ESCALATION_NUM‘		
:variable:‘Tokudb_LOCKTREE_ESCALATION_SECONDS‘		
:variable:‘Tokudb_LOCKTREE_LATEST_POST_ESCALATION_MEMORY_SIZE‘		
:variable:‘Tokudb_LOCKTREE_OPEN_CURRENT‘		
:variable:‘Tokudb_LOCKTREE_PENDING_LOCK_REQUESTS‘		
:variable:‘Tokudb_LOCKTREE_STO_ELIGIBLE_NUM‘		
:variable:‘Tokudb_LOCKTREE_STO_ENDED_NUM‘		
:variable:‘Tokudb_LOCKTREE_STO_ENDED_SECONDS‘		
:variable:‘Tokudb_LOCKTREE_WAIT_COUNT‘		
:variable:‘Tokudb_LOCKTREE_WAIT_TIME‘		
:variable:‘Tokudb_LOCKTREE_LONG_WAIT_COUNT‘		
:variable:‘Tokudb_LOCKTREE_LONG_WAIT_TIME‘		
:variable:‘Tokudb_LOCKTREE_TIMEOUT_COUNT‘		

Continued on next page

Table 79.2 – continued from previous page

Name	Var Type	Var Scope
:variable:‘Tokudb_LOCKTREE_WAIT_ESCALATION_COUNT‘		
:variable:‘Tokudb_LOCKTREE_WAIT_ESCALATION_TIME‘		
:variable:‘Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_COUNT‘		
:variable:‘Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_TIME‘		
:variable:‘Tokudb_DICTIONARY_UPDATES‘		
:variable:‘Tokudb_DICTIONARY_BROADCAST_UPDATES‘		
:variable:‘Tokudb_DESCRIPTOR_SET‘		
:variable:‘Tokudb_MESSAGES_IGNORED_BY_LEAF_DUE_TO_MSN‘		
:variable:‘Tokudb_TOTAL_SEARCH_RETRIES‘		
:variable:‘Tokudb_SEARCH_TRIES_GT_HEIGHT‘		
:variable:‘Tokudb_SEARCH_TRIES_GT_HEIGHTPLUS3‘		
:variable:‘Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT‘		
:variable:‘Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_BYTES‘		
:variable:‘Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_UNCOMPRESSED_BYTES‘		
:variable:‘Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_SECONDS‘		
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT‘		
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_BYTES‘		
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_UNCOMPRESSE‘		
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_SECONDS‘		
:variable:‘Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT‘		
:variable:‘Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_BYTES‘		
:variable:‘Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_UNCOMPRESSED_BYTES‘		
:variable:‘Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_SECONDS‘		
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT‘		
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_BYTES‘		
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_UNCOMPRESSED_BY‘		
:variable:‘Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_SECONDS‘		
:variable:‘Tokudb_LEAF_NODE_COMPRESSION_RATIO‘		
:variable:‘Tokudb_NONLEAF_NODE_COMPRESSION_RATIO‘		
:variable:‘Tokudb_OVERALL_NODE_COMPRESSION_RATIO‘		
:variable:‘Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS‘		
:variable:‘Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS_BYTES‘		
:variable:‘Tokudb_LEAF_NODE_PARTIAL_EVICTIONS‘		
:variable:‘Tokudb_LEAF_NODE_PARTIAL_EVICTIONS_BYTES‘		
:variable:‘Tokudb_LEAF_NODE_FULL_EVICTIONS‘		
:variable:‘Tokudb_LEAF_NODE_FULL_EVICTIONS_BYTES‘		
:variable:‘Tokudb_NONLEAF_NODE_FULL_EVICTIONS‘		
:variable:‘Tokudb_NONLEAF_NODE_FULL_EVICTIONS_BYTES‘		
:variable:‘Tokudb_LEAF_NODES_CREATED‘		
:variable:‘Tokudb_NONLEAF_NODES_CREATED‘		
:variable:‘Tokudb_LEAF_NODES_DESTROYED‘		
:variable:‘Tokudb_NONLEAF_NODES_DESTROYED‘		
:variable:‘Tokudb_MESSAGES_INJECTED_AT_ROOT_BYTES‘		
:variable:‘Tokudb_MESSAGES_FLUSHED_FROM_H1_TO_LEAVES_BYTES‘		
:variable:‘Tokudb_MESSAGES_IN_TREES_ESTIMATE_BYTES‘		
:variable:‘Tokudb_MESSAGES_INJECTED_AT_ROOT‘		
:variable:‘Tokudb_BROADCASE_MESSAGES_INJECTED_AT_ROOT‘		
:variable:‘Tokudb_BASEMENTS_DECOMPRESSED_TARGET_QUERY‘		

Continued on next page

Table 79.2 – continued from previous page

Name	Var Type	Var Scope
:variable:‘Tokudb_BASEMENTS_DECOMPRESSED_PRELOCKED_RANGE‘		
:variable:‘Tokudb_BASEMENTS_DECOMPRESSED_PREFETCH‘		
:variable:‘Tokudb_BASEMENTS_DECOMPRESSED_FOR_WRITE‘		
:variable:‘Tokudb_BUFFERS_DECOMPRESSED_TARGET_QUERY‘		
:variable:‘Tokudb_BUFFERS_DECOMPRESSED_PRELOCKED_RANGE‘		
:variable:‘Tokudb_BUFFERS_DECOMPRESSED_PREFETCH‘		
:variable:‘Tokudb_BUFFERS_DECOMPRESSED_FOR_WRITE‘		
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_QUERY‘		
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_QUERY_BYTES‘		
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_QUERY_SECONDS‘		
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_PREFETCH‘		
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_BYTES‘		
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_SECONDS‘		
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_WRITE‘		
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_WRITE_BYTES‘		
:variable:‘Tokudb_PIVOTS_FETCHED_FOR_WRITE_SECONDS‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_TARGET_QUERY‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_BYTES‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_SECONDS‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_BYTES‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_SECONDS‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_PREFETCH‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_PREFETCH_BYTES‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_PREFETCH_SECONDS‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_FOR_WRITE‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_FOR_WRITE_BYTES‘		
:variable:‘Tokudb_BASEMENTS_FETCHED_FOR_WRITE_SECONDS‘		
:variable:‘Tokudb_BUFFERS_FETCHED_TARGET_QUERY‘		
:variable:‘Tokudb_BUFFERS_FETCHED_TARGET_QUERY_BYTES‘		
:variable:‘Tokudb_BUFFERS_FETCHED_TARGET_QUERY_SECONDS‘		
:variable:‘Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE‘		
:variable:‘Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_BYTES‘		
:variable:‘Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_SECONDS‘		
:variable:‘Tokudb_BUFFERS_FETCHED_PREFETCH‘		
:variable:‘Tokudb_BUFFERS_FETCHED_PREFETCH_BYTES‘		
:variable:‘Tokudb_BUFFERS_FETCHED_PREFETCH_SECONDS‘		
:variable:‘Tokudb_BUFFERS_FETCHED_FOR_WRITE‘		
:variable:‘Tokudb_BUFFERS_FETCHED_FOR_WRITE_BYTES‘		
:variable:‘Tokudb_BUFFERS_FETCHED_FOR_WRITE_SECONDS‘		
:variable:‘Tokudb_LEAF_COMPRESSION_TO_MEMORY_SECONDS‘		
:variable:‘Tokudb_LEAF_SERIALIZATION_TO_MEMORY_SECONDS‘		
:variable:‘Tokudb_LEAF_DECOMPRESSION_TO_MEMORY_SECONDS‘		
:variable:‘Tokudb_LEAF_DESERIALIZATION_TO_MEMORY_SECONDS‘		
:variable:‘Tokudb_NONLEAF_COMPRESSION_TO_MEMORY_SECONDS‘		
:variable:‘Tokudb_NONLEAF_SERIALIZATION_TO_MEMORY_SECONDS‘		
:variable:‘Tokudb_NONLEAF_DECOMPRESSION_TO_MEMORY_SECONDS‘		
:variable:‘Tokudb_NONLEAF_DESERIALIZATION_TO_MEMORY_SECONDS‘		

Continued on next page

Table 79.2 – continued from previous page

Name	Var Type	Var Scope
:variable:‘Tokudb_PROMOTION_ROOTS_SPLIT‘		
:variable:‘Tokudb_PROMOTION_LEAF_ROOTS_INJECTED_INTO‘		
:variable:‘Tokudb_PROMOTION_H1_ROOTS_INJECTED_INTO‘		
:variable:‘Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_0‘		
:variable:‘Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_1‘		
:variable:‘Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_2‘		
:variable:‘Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_3‘		
:variable:‘Tokudb_PROMOTION_INJECTIONS_LOWER_THAN_DEPTH_3‘		
:variable:‘Tokudb_PROMOTION_STOPPED_NONEMPTY_BUFFER‘		
:variable:‘Tokudb_PROMOTION_STOPPED_AT_HEIGHT_1‘		
:variable:‘Tokudb_PROMOTION_STOPPED_CHILD_LOCKED_OR_NOT_IN_MEMORY‘		
:variable:‘Tokudb_PROMOTION_STOPPED_CHILD_NOT_FULLY_IN_MEMORY‘		
:variable:‘Tokudb_PROMOTION_STOPPED_AFTER_LOCKING_CHILD‘		
:variable:‘Tokudb_BASEMENT_DESERIALIZATION_FIXED_KEY‘		
:variable:‘Tokudb_BASEMENT_DESERIALIZATION_VARIABLE_KEY‘		
:variable:‘Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_SUCCESS‘		
:variable:‘Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_FAIL_POS‘		
:variable:‘Tokudb_RIGHTMOST_LEAF_SHORTCUT_FAIL_REACTIVE‘		
:variable:‘Tokudb_CURSOR_SKIP_DELETED_LEAF_ENTRY‘		
:variable:‘Tokudb_FLUSHER_CLEANER_TOTAL_NODES‘		
:variable:‘Tokudb_FLUSHER_CLEANER_H1_NODES‘		
:variable:‘Tokudb_FLUSHER_CLEANER_HGT1_NODES‘		
:variable:‘Tokudb_FLUSHER_CLEANER_EMPTY_NODES‘		
:variable:‘Tokudb_FLUSHER_CLEANER_NODES_DIRTIED‘		
:variable:‘Tokudb_FLUSHER_CLEANER_MAX_BUFFER_SIZE‘		
:variable:‘Tokudb_FLUSHER_CLEANER_MIN_BUFFER_SIZE‘		
:variable:‘Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_SIZE‘		
:variable:‘Tokudb_FLUSHER_CLEANER_MAX_BUFFER_WORKDONE‘		
:variable:‘Tokudb_FLUSHER_CLEANER_MIN_BUFFER_WORKDONE‘		
:variable:‘Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_WORKDONE‘		
:variable:‘Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_STARTED‘		
:variable:‘Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_RUNNING‘		
:variable:‘Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_COMPLETED‘		
:variable:‘Tokudb_FLUSHER_CLEANER_NUM_DIRTIED_FOR_LEAF_MERGE‘		
:variable:‘Tokudb_FLUSHER_FLUSH_TOTAL‘		
:variable:‘Tokudb_FLUSHER_FLUSH_IN_MEMORY‘		
:variable:‘Tokudb_FLUSHER_FLUSH_NEEDED_IO‘		
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES‘		
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_1‘		
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_2‘		
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_3‘		
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_4‘		
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_5‘		
:variable:‘Tokudb_FLUSHER_FLUSH_CASCADES_GT_5‘		
:variable:‘Tokudb_FLUSHER_SPLIT_LEAF‘		
:variable:‘Tokudb_FLUSHER_SPLIT_NONLEAF‘		
:variable:‘Tokudb_FLUSHER_MERGE_LEAF‘		
:variable:‘Tokudb_FLUSHER_MERGE_NONLEAF‘		

Continued on next page

Table 79.2 – continued from previous page

Name	Var Type	Var Scope
:variable:'Tokudb_FLUSHER_BALANCE_LEAF'		
:variable:'Tokudb_HOT_NUM_STARTED'		
:variable:'Tokudb_HOT_NUM_COMPLETED'		
:variable:'Tokudb_HOT_NUM_ABORTED'		
:variable:'Tokudb_HOT_MAX_ROOT_FLUSH_COUNT'		
:variable:'Tokudb_TXN_BEGIN'		
:variable:'Tokudb_TXN_BEGIN_READ_ONLY'		
:variable:'Tokudb_TXN_COMMITS'		
:variable:'Tokudb_TXN_ABORTS'		
:variable:'Tokudb_LOGGER_NEXT_LSN'		
:variable:'Tokudb_LOGGER_WRITES'		
:variable:'Tokudb_LOGGER_WRITES_BYTES'		
:variable:'Tokudb_LOGGER_WRITES_UNCOMPRESSED_BYTES'		
:variable:'Tokudb_LOGGER_WRITES_SECONDS'		
:variable:'Tokudb_LOGGER_WAIT_LONG'		
:variable:'Tokudb_LOADER_NUM_CREATED'		
:variable:'Tokudb_LOADER_NUM_CURRENT'		
:variable:'Tokudb_LOADER_NUM_MAX'		
:variable:'Tokudb_MEMORY_MALLOC_COUNT'		
:variable:'Tokudb_MEMORY_FREE_COUNT'		
:variable:'Tokudb_MEMORY_REALLOC_COUNT'		
:variable:'Tokudb_MEMORY_MALLOC_FAIL'		
:variable:'Tokudb_MEMORY_REALLOC_FAIL'		
:variable:'Tokudb_MEMORY_REQUESTED'		
:variable:'Tokudb_MEMORY_USED'		
:variable:'Tokudb_MEMORY_FREED'		
:variable:'Tokudb_MEMORY_MAX_REQUESTED_SIZE'		
:variable:'Tokudb_MEMORY_LAST_FAILED_SIZE'		
:variable:'Tokudb_MEM_ESTIMATED_MAXIMUM_MEMORY_FOOTPRINT'		
:variable:'Tokudb_MEMORY_MALLOCATOR_VERSION'		
:variable:'Tokudb_MEMORY_MMAP_THRESHOLD'		
:variable:'Tokudb_FILESYSTEM_THREADS_BLOCKED_BY_FULL_DISK'		
:variable:'Tokudb_FILESYSTEM_FSYNC_TIME'		
:variable:'Tokudb_FILESYSTEM_FSYNC_NUM'		
:variable:'Tokudb_FILESYSTEM_LONG_FSYNC_TIME'		
:variable:'Tokudb_FILESYSTEM_LONG_FSYNC_NUM'		

CHAPTER

EIGHTY

DEVELOPMENT OF *PERCONA SERVER FOR MYSQL*

Percona Server for MySQL is an open source project to produce a distribution of the *MySQL* Server with improved performance, scalability and diagnostics.

80.1 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.

80.1.1 Overview

At Percona we use Git for source control, GitHub for code hosting, and Jira for release management.

We change our software to implement new features and/or to fix bugs. Refactoring could be classed either as a new feature or a bug depending on the scope of work.

New features and bugs are targeted to specific releases. A release is part of a series. For example, 2.4 is a series in Percona XtraBackup and 2.4.15, 2.4.16 and 2.4.17 are releases in this series.

Code is proposed for merging in the form of pull requests on GitHub.

For *Percona Server for MySQL*, we have several Git branches on which development occurs: 5.5, 5.6, 5.7, and 8.0. As *Percona Server for MySQL* is not a traditional project, instead of being a set of patches against an existing product, these branches are not related. In other words, we do not merge from one release branch to another. To have your changes in several branches, you must propose branches to each release branch.

80.1.2 Making a Change to a Project

In this case, we are going to use `percona-xtrabackup` as an example. The workflow is similar for *Percona Server for MySQL*, but the patch will need to be modified in all release branches of *Percona Server for MySQL*.

- `git branch https://github.com/percona/percona-xtrabackup/featureX` (where 'featureX' is a sensible name for the task at hand)
- (developer makes changes in featureX, testing locally)
- The Developer pushes to `https://github.com/percona/username/percona-xtrabackup/featureX`
- The developer can submit a pull request to https://github.com/percona/percona-xtrabackup,
- Code undergoes a review
- Once code is accepted, it can be merged

If the change also applies to a stable release (e.g. 2.4) then changes should be made on a branch of 2.4 and merged to a branch of trunk. In this case there should be two branches run through the param build and two merge proposals (one for the stable release and one with the changes merged to trunk). This prevents somebody else having to guess how to merge your changes.

80.1.3 *Percona Server for MySQL*

The same process for *Percona Server for MySQL*, but we have several different branches (and merge requests).

CHAPTER

EIGHTYONE

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.

CHAPTER

EIGHTYTWO

INDEX OF INFORMATION_SCHEMA TABLES

This is a list of the INFORMATION_SCHEMA TABLES that exist in *Percona Server for MySQL* with *XtraDB*. The entry for each table points to the page in the documentation where it's described.

- **:table:'CLIENT_STATISTICS'**
- **:table:'GLOBAL_TEMPORARY_TABLES'**
- **:table:'INDEX_STATISTICS'**
- **:table:'INNODB_CHANGED_PAGES'**
- **:table:'QUERY_RESPONSE_TIME'**
- **:table:'TABLE_STATISTICS'**
- **:table:'TEMPORARY_TABLES'**
- **:table:'THREAD_STATISTICS'**
- **:table:'USER_STATISTICS'**
- **:table:'XTRADB_INTERNAL_HASH_TABLES'**
- **:table:'XTRADB_READ_VIEW'**
- **:table:'XTRADB_RSEG'**
- **:table:'XTRADB_ZIP_DICT'**
- **:table:'XTRADB_ZIP_DICT_COLS'**

CHAPTER

EIGHTYTHREE

FREQUENTLY ASKED QUESTIONS

83.1 Q: Will *Percona Server for MySQL* 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 for MySQL* with *XtraDB*, but you should not base your decision on that.

83.2 Q: Will we have to *GPL* our whole application if we use *Percona Server for MySQL* 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 for MySQL* 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*.

83.3 Q: Do I need to install *Percona* client libraries?

A: No, you don't need to change anything on the clients. *Percona Server for MySQL* is 100% compatible with all existing client libraries and connectors.

83.4 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.

CHAPTER

EIGHTYFOUR

COPYRIGHT AND LICENSING INFORMATION

84.1 Documentation Licensing

84.2 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.

CHAPTER

EIGHTYFIVE

PERCONA SERVER FOR MYSQL 5.6 RELEASE NOTES

85.1 *Percona Server for MySQL* 5.6.51-93.0

Date December 21, 2022

Percona Server for MySQL 5.6.51-93.0 includes all the features and bug fixes available in MySQL 5.6.51 Community Edition in addition to enterprise-grade features developed by Percona.

85.1.1 Bugs Fixed

- PS-7856: A server exit when auto-incremented column was updated ina partitioned table.
- Bug #28142052: A *THD::m_query_string* could be allocated on the wrong MEM_ROOT, which allowed the prepared statement to make the string invalid by deleting *Prepared_statement::main_mem_root*.
- Bug #31933415: Fixed out of bounds read in ER().
- Bug#32803050: When an Update or Delete was executed with indexes on a table with a row-based replication setup and a federated table on the replica, the replica exited.
- PS-8129: Fixed a mutex hang in *threadpool_unix*.
- PS-7563: Fixed a Read buffer overflow in the *DBUG_PRINT* macro in mysqltest.
- PS-7769: Fixed a use-after-return error in *audit_log_exclude_account_validate*.
- PS-8204: The wrong length was specified for some XML escape rules and the terminating null symbol from the replacement rule was copied into the resulting string. This operation caused text truncation in the audit log file.

85.2 *Percona Server for MySQL* 5.6.51-92.0

Date November 11, 2021

Percona Server for MySQL 5.6.51-92.0 includes all the features and bug fixes available in MySQL 5.6.51 Community Edition in addition to enterprise-grade features developed by Percona.

85.2.1 Bugs Fixed

- Bug #32391415: The mysql_change_user() fails with unknown collation.
- Bug #31576731: The innodb_ft_total_cache_size was not capped when set to total memory.
- Bug#32620378, Bug#32620398: The innodb_memcached plugin code allowed integer underflow.

- BUG#30366310: Using a function to assign default values to two or more variables caused server exit.
- Bug #31674599: The udf_int() function could cause a server exit when encountering an incorrect name error.
- BUG#31774422: A transaction commit during the update of a system variable, which causes a binary log rotation, while concurrently reading variables from another connection may cause a deadlock situation.
- Bug #31899685: The Data Dictionary table instance open and close sequence may cause the InnoDB table share instance to be evicted from the dictionary cache.
- Bug #32316323: A buffer miscalculation in a Microsoft Windows 10 zipped archive could cause a server exit.
- PS-7746: Possible double call to *free_share()* in ha_innobase::open()

85.3 *Percona Server for MySQL* 5.6.51-91.0

Date February 8, 2021

Installation Installing Percona Server for MySQL

Percona Server for MySQL 5.6.51-91.0 includes all the features and bug fixes available in MySQL 5.6.51 Community Edition in addition to enterprise-grade features developed by Percona.

85.3.1 Bugs Fixed

- PS-1956: LP #1088929: Changed data type for some microsecond times for the slow query log to 64-bit

85.4 *Percona Server for MySQL* 5.6.50-90.0

Date November 2, 2020

Installation Installing Percona Server for MySQL

Percona Server for MySQL 5.6.50-90.0 includes all the features and bug fixes available in MySQL 5.6.50 Community Edition in addition to enterprise-grade features developed by Percona.

85.4.1 Bugs Fixed

- PS-7346: Correct the buffer calculation for the audit plugin used when large queries are executed(PS-5395).

85.5 *Percona Server for MySQL* 5.6.49-89.0

Date July 27, 2020

Installation Installing Percona Server for MySQL

Percona Server for MySQL 5.6.49-89.0 includes all the features and bug fixes available in MySQL 5.6.49 Community Edition in addition to enterprise-grade features developed by Percona.

85.5.1 Bugs Fixed

- PS-7043: Incorrect result when constant equality expression is used in LEFT JOIN condition (Upstream #99499)
- PS-6995: Optimizer not picking best execution plan for Primary Key lookup
- PS-5620: Modify Docker image to support supplying custom TLS certificates (Thanks to user agarner for reporting this issue)

85.6 *Percona Server for MySQL* 5.6.48-88.0

Date May 13, 2020

Installation Installing Percona Server for MySQL

Percona Server for MySQL 5.6.48-88.0 includes all the features and bug fixes available in MySQL 5.6.48 Community Edition in addition to enterprise-grade features developed by Percona.

85.6.1 Improvements

- PS-5391: RedHat-based distributions may set service configurations in a file saved in the '/etc/sysconfig' directory. The init script did not read any file in that directory. We have added the ability for the init.d script to read files in the 'etc/sysconfig' directory.

85.6.2 Bugs Fixed

- PS-6953: If the LOAD DATA process found a line with an escape character followed by UTF8 characters, then the process failed to load the line and any additional lines. This issue was a regression. (Upstream #98089)
- PS-6945: The tokubackup process exported an incorrect API. This API mismatch prevented large file backups.
- PS-6946: The address and thread sanitizers found memory use after free and other issues in the tokubackup software..
- PS-7020: In Ubuntu 20.04, only python3-mysqldb is available. We converted the MTR tests to support both Python2 and Python3. The scripts require Python2.6 or newer.
- PS-6954: In the tokudb-backup-plugin, there was a collision between -std=c++11 and -std=gnu++03.
- PS-6899: The tests, main.events_bugs and main.events_1, failed because 2020-01-01 was considered a future time. (Upstream #98860)
- PS-6110: The libHotBackup.so was missing from Percona-Server-5.6.46-rel86.2-Linux.x86_64.ssl101.tar.gz.
- PS-4649: TokuDB: Documented PerconaFT, which is fractal tree indexing. This type of indexing enhances the B-tree data structure.
- PS-6966: In Percona Server compilation, fixed clang-10 compilation issues. (Upstream #99221)

85.7 *Percona Server for MySQL* 5.6.47-87.0

Date January 21, 2020

Installation Installing Percona Server for MySQL

Percona Server for MySQL 5.6.47-87.0 includes all the features and bug fixes available in MySQL 5.6.47 Community Edition in addition to enterprise-grade features developed by Percona.

85.7.1 Bugs Fixed

- #1469: The Memory storage engine detects an incorrect "is full" condition stuck when the space contained reusable memory chunks and the space could be reused.
- #6023: Percona Server exits when a Kerberos password is changed after the password has expired.
- #5813: Setting 'none' value for **:variable:'slow_query_log_use_global_control'** throws error.
- #4541: Document that maximum- and minimum- option modifiers should not be used with non-numeric system variables
- #6750: Red Hat Enterprise Linux 8 had a file conflict when installing client packages
- #5675: wrong row is updated (Upstream #96578)
- #5591: mysql server crash after empty set sql query (Upstream #95236)
- #1673: **:variable:'ps_tokudb_admin'** cannot install tokudb remotely
- #5956: Root session must not be allowed to kill *Utility user* session
- #5952: *Utility user* is visible in performance_schema .threads
- #5642: page tracker thread does not exit if startup fails
- #4842: Documentation - **:variable:'innodb_corrupt_table_action'** does not list a default value
- #5521: Documentation - **:variable:'tokudb_cache_size'** definition missing

85.8 *Percona Server for MySQL* 5.6.46-86.2

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.46-86.2 on November 6, 2019 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.46, including all the bug fixes in it, *Percona Server for MySQL* 5.6.46-86.2 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free.

85.8.1 Bugs Fixed

- The Audit log filtering by user was not working. Bug fixed #5707.

85.8.2 Other Bugs Fixed

#5988, #5216, #5327, #5761, #5933, #5941, and #5997.

Find the release notes for Percona Server for MySQL 5.6.46-86.2 in our online documentation. Report bugs in the Jira bug tracker.

85.9 *Percona Server for MySQL* 5.6.45-86.1

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.45-86.1 on August 20, 2019 (Downloads are available here and from the *Percona Software Repositories*).

This release merges changes of MySQL 5.6.45, including all the bug fixes in it. Percona Server for MySQL 5.6.45-86.1 is now the current GA release in the 5.6 series. All of Percona's software is open-source and free.

85.9.1 Bugs Fixed

- The TokuDB hot backup library continually dumps TRACE information to the server error log. The user cannot enable or disable the dump of this information. Bug fixed #4850
- The TokuDBBackupPlugin is optional at cmake time. Bug fixed #5748.

85.9.2 Other Bugs Fixed

- #5531
- #5146
- #5638
- #5645
- #5669
- #5749
- #5752
- #5780
- #5725
- #5742
- #5743
- #5746

Find the release notes for Percona Server for MySQL 5.6.45-86.1 in online documentation. Report bugs in the Jira bug tracker.

85.10 *Percona Server for MySQL* 5.6.44-86.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.44-86.0 on May 24, 2019 (Downloads are available here and from the *Percona Software Repositories*).

This release merges changes of MySQL 5.6.44, including all the bug fixes in it. Percona Server for MySQL 5.6.44-86.0 is now the current GA release in the 5.6 series. All of Percona's software is open-source and free.

85.10.1 Bugs Fixed

Percona Server 5.6.44-85.0-1 Debian/Ubuntu packages would reset the root password to an empty string during the upgrade on Debian/Ubuntu. This fixes the CVE-2019-12301 issue. Bug fixed #5640.

As mentioned in the blogpost packages were replaced in the repositories after the issue was discovered. While CentOS/RHEL wasn't affected by the CVE, some users got the `[Errno -1] Package does not match intended download.` message when trying to install the packages once they were replaced.

Report bugs in the Jira bug tracker.

85.11 *Percona Server for MySQL* 5.6.44-85.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.44-85.0 on May 17, 2019 (Downloads are available here and from the *Percona Software Repositories*).

This release merges changes of MySQL 5.6.44, including all the bug fixes in it. Percona Server for MySQL 5.6.44-85.0 is now the current GA release in the 5.6 series. All of Percona's software is open-source and free.

85.11.1 New Features

- New **:variable:'Audit_log_buffer_size_overflow'** status variable has been implemented to track when an *Audit Log Plugin* entry was either dropped or written directly to the file due to its size being bigger than **:variable:'audit_log_buffer_size'** variable.

85.11.2 Bugs Fixed

- TokuDB storage engine would assert on load when used with jemalloc 5.x. Bug fixed #5406.
- `FLUSH` commands written to the binary log could cause errors in case of replication. Bug fixed #1827 (upstream #88720).
- TokuDB storage engine couldn't be enabled on Docker images. Bug fixed #5283.
- the `ACCESS_DENIED` field of the **:table:'information_schema.user_statistics'** table was not updated correctly. Bugs fixed #3956, #4996.
- the page cleaner could sleep for long time when the system clock was adjusted to an earlier point in time. Bug fixed #5221 (upstream #93708).
- executing `SHOW BINLOG EVENT` from an invalid position could result in a segmentation fault on 32bit machines. Bug fixed #5243.
- `BLOB` entries in the binary log could become corrupted in case when a database with `Blackhole` tables served as an intermediate binary log server in a replication chain. Bug fixed #5353 (upstream #93917).
- when *Audit Log Plugin* was enabled, the server could use a lot of memory when handling large queries. Bug fixed #5395.
- PerconaFT `locktree` library was re-licensed to Apache v2 license. Bug fixed #5501.

Other bugs fixed: #5512, #5550, #5578, #5388 (upstream #94121), and #5441.

This release also contains the fixes for the following security issues: CVE-2018-3123, CVE-2019-2683, CVE-2019-2627, and CVE-2019-2614.

Report bugs in the Jira bug tracker.

85.12 *Percona Server for MySQL* 5.6.43-84.3

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.43-84.3 on January 29, 2019 (Downloads are available here and from the *Percona Software Repositories*).

This release merges changes of MySQL 5.6.43, including all the bug fixes in it. *Percona Server for MySQL* 5.6.43-84.3 is now the current GA release in the 5.6 series. All of Percona's software is open-source and free.

85.12.1 Bugs Fixed

- A sequence of LOCK TABLES FOR BACKUP and STOP SLAVE SQL_THREAD could cause replication to be blocked and not possible to be restarted normally. Bug fixed #4758 (upstream #93649).
- **http** was replaced with **https** in http://bugs.percona.com in server crash messages. Bug fixed #4855.
- Wrong query results could be received in semi-join subqueries with materialization-scan that allowed inner tables of different semijoin nests to interleave. Bug fixed #4907 (upstream bug #92809).
- The audit logs could be corrupted when the `audit_log_rotations` option was changed at runtime. Bug fixed #4950.
- There was a typo in `mysqld_safe.sh`: **trottling** was replaced with **throttling**. Bug fixed #240. Thanks to Michael Coburn for the patch.

Other bugs fixed: #2477, #3535, #3568, #3672, #3673, #4791, #4989, #5100, #5118, #5163, #5268, #5270, #5271.

This release also contains fixes for the following CVE issues: CVE-2019-2534, CVE-2019-2529, CVE-2019-2482, CVE-2019-2455, CVE-2019-2503, CVE-2018-0734.

Find the release notes for *Percona Server for MySQL* 5.6.43-84.3 in our online documentation. Report bugs in the Jira bug tracker.

85.13 *Percona Server for MySQL* 5.6.42-84.2

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.42-84.2 on November 29, 2018 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.42, including all the bug fixes in it, *Percona Server for MySQL* 5.6.42-84.2 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free.

85.13.1 Improvements

- #4790: Improve user statistics accuracy

Other improvements

- #4881: Add LLVM/clang 7 to Travis-CI

85.13.2 Bugs Fixed

- Slave replication could break if upstream bug #74145 (*FLUSH LOGS improperly disables the logging if the log file cannot be accessed*) occurred in master. Bug fixed #1017 (Upstream #83232).

- The binary log could be corrupted when the disk partition used for temporary. files (`tmpdir` system variable) had little free space. Bug fixed #1107 (Upstream #72457).
- PURGE CHANGED_PAGE_BITMAPS did not work when the `innodb_data_home_dir` system variable was used. Bug fixed #4723.
- Setting the `tokudb_last_lock_timeout` variable via command line could cause server to stop working when the actual timeout took place. Bug fixed #4943.
- Dropping TokuDB table with non-alphanumeric characters could lead to a crash. Bug fixed #4979.

Other bugs fixed

- #4781: sql_yacc.yy uses SQLCOM_SELECT instead of SQLCOM_SHOW_XXXX_STATS
- #4529: MTR: index_merge_rocksdb2 inadvertently tests InnoDB instead of MyRocks
- #4746: Revert our fix for PS-3851 (Percona Ver 5.6.39-83.1 Failing assertion: sym_node->table != NULL)
- #4773: Percona Server sources can't be compiled without server
- #4785: Setting version_suffix to NULL leads to handle_fatal_signal (sig=11) in Sys_var_version::global_value_ptr
- #4813: Using flush_caches leads to SELinux denial errors

Find the release notes for Percona Server for MySQL 5.6.42-84.2 in our online documentation. Report bugs in the Jira bug tracker.

85.14 *Percona Server for MySQL* 5.6.41-84.1

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.41-84.1 on August 17, 2018 (Downloads are available here and from the *Percona Software Repositories*).

This release merges changes of MySQL 5.6.41, including all the bug fixes in it. Percona Server for MySQL 5.6.41-84.1 is now the current GA release in the 5.6 series. All of Percona's software is open-source and free.

85.14.1 Bugs Fixed

- A simple `SELECT` query on a table with CHARSET=euckr COLLATE=euckr_bin could return different results each time it was executed. Bug fixed #4513 (upstream #91091).
- Percona Server 5.6.39-83.1 could crash when altering an InnoDB table that has a full-text search index defined. Bug fixed #3851 (upstream #68987).

85.14.2 Other Bugs Fixed

- #3325: "online upgrade GTID cause binlog damage in high write QPS situation"
- #3976: "Errors in MTR tests main.variables-big, main.information_schema-big, innodb.innodb_bug14676111"
- #4506: "Backport fixes from 8.0 for InnoDB memcached Plugin"

Find the release notes for Percona Server for MySQL 5.6.41-84.1 in online documentation. Report bugs in the Jira bug tracker.

85.15 *Percona Server for MySQL* 5.6.40-84.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.40-84.0 on May 30, 2018 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.40, including all the bug fixes in it, *Percona Server for MySQL* 5.6.40-84.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free.

85.15.1 New Features

- A new string variable **:variable:'version_suffix'** allows to change suffix for the *Percona Server for MySQL* version string returned by the read-only **:variable:'version'** variable. This allows to append the version number for the server with a custom suffix to reflect some build or configuration specifics. Also **:variable:'version_comment'** (default value of which is taken from the CMake `COMPILATION_COMMENT` option) is converted from a global read-only to a global read-write variable and thereby it is now cutomizable.
- Query response time plugin now can be disabled at session level with use of a new variable **:variable:'query_response_time_session_stats'**.

85.15.2 Bugs Fixed

- Compilation warning was fixed for `-DWITH_QUERY_RESPONSE_TIME=ON` CMake compilation option, which makes QRT to be linked statically. Bug fixed #3841.
- A code clean-up was done to fix clang 6 specific compilation warnings and errors (bug fixed #3893, upstream #90111).
- Using `-DWITHOUT_<PLUGIN>=ON` CMake variable to exclude a plugin from the build didn't work for some plugins, including a number of storage engines. Bug fixed #3901.
- A clean-up in *Percona Server for MySQL* binlog-related code was made to avoid uninitialized memory comparison. Bug fixed #3925 (upstream #90238).
- Temporary file I/O was not instrumented for Performance Schema. Bug fixed #3937 (upstream #90264).
- A `key_block_size` value was set automatically by the Improved MEMORY Storage Engine, which resulted in warnings when changing the engine type to *InnoDB*, and constantly growing `key_block_size` during alter operations. Bugs fixed #3936, #3940, and #3943.
- *Percona Server for MySQL* Debian packages description included reference to `/etc/mysql/my.cnf` file, which is not actually present in these packages. Bug fixed #2046.
- Fixes were introduced to remove GCC 8 compilation warnings for the *Percona Server for MySQL* build, retaining compatibility with old compiler versions, including GCC 4.4. Bugs fixed #3950 and #4471.
- A typo in `plugin.cmake` file prevented to compile plugins statically into the server. Bug fixed #3871 (upstream #89766).
- `-DWITH_NUMA=ON` build option was silently ignored by CMake when NUMA development package was not installed, instead of exiting by error. Bug fixed #4487.
- Variables **:variable:'innodb_buffer_pool_populate'** and **:variable:'numa_interleave'** mapped to the upstream innodb_numa_interleave variable in **:rn:'5.6.27-75.0'** were reverted to their original implementation due to upstream variant being less effective in memory allocation. Now buffer pool is allocated with MAP_POPULATE, forcing NUMA interleaved allocation at the buffer pool initialization time. Bug fixed #3967.
- **:variable:'audit_log_include_accounts'** variable did not take effect if placed in `my.cnf` configuration file, while still working as intended if set dynamically. Bug fixed #3867.

- Synchronization between between **:variable:‘innodb_kill_idle_transaction‘** and **:variable:‘kill_idle_transaction‘** system variables was broken because of the regression in *Percona Server for MySQL* **:rn:‘5.6.40-83.2‘**. Bug fixed #3955.
- Executing the `SHOW GLOBAL STATUS` expression could cause “data drift” on global status variables in case of a query rollback: the variable, being by its nature a counter and allowing only an increase, could return to its previous value. Bug fixed #3951 (upstream #90351).
- `ALTER TABLE ... COMMENT = ...` statement caused *TokuDB* to rebuild the whole table, which is not needed, as only FRM metadata should be changed. The fix was provided as a contribution by Fungo Wang. Bugs fixed #4280 and #4292.
- A number of *Percona Server for MySQL* 8.0 *TokuDB* fixes have been backported to *Percona Server for MySQL* 5.6 in preparation for using MySQL 8.0. Bugs fixed #4379, #4380, #4387, #4378, #4383, #4384, #4386, #4382, #4391, #4390, #4392, and #4381.

85.15.3 TokuDB Changes and Fixes

- Two new variables, **:variable:‘tokudb_enable_fast_update‘** and **:variable:‘tokudb_enable_fast_upsert‘**, were introduced to facilitate the *TokuDB* fast updates feature, which involves queries optimization to avoid random reads during their execution. Bug fixed #4365.
- A data race was fixed in minicron utility of the PerconaFT, as a contribution by Rik Prohaska. Bug fixed #4281.

85.15.4 Other Bugs Fixed

- #3818 “Orphaned file mysql-test/suite/innodb/r/percona_innodb_kill_idle_trx.result”
- #3926 “Potentially truncated bitmap file name in log_online_open_bitmap_file_read_only() (storage/innobase/log/log0online.cc)”
- #2204 “Test main.audit_log_default_db is unstable”
- #3767 “Fix compilation warnings/errors with clang”
- #3773 “Incorrect key file for table frequently for tokudb”
- #3794 “MTR test main.percona_show_temp_tables_stress does not wait for events to start”
- #3798 “MTR test innodb.percona_extended_innodb_status fails if InnoDB status contains unquoted special characters”
- #3887 “TokuDB does not compile with -DWITH_PERFSCHEMA_STORAGE_ENGINE=OFF”
- #4388 “5.7 code still has TOKU_INCLUDE_OPTION_STRUCTS which is a MariaDB specific construct”
- #4265 “TDB-114 (Change use of MySQL HASH to unordered_map) introduces memory leak”
- #4277 “memory leaks in TDB-2 and TDB-89 tests”
- #4276 “Data race on cache table attributes detected by the thread sanitizer”
- #4451 “Implement better compression algo testing”
- #4469 “variable use out of scope bug in get_last_key test detected by ASAN in clang 6”
- #4470 “the cachetable-simple-pin-nonblocking-cheap test occasionally fails due to a locking conflict with the cachetable evictor”
- #1131 “User_var_log_event::User_var_log_event(const char*, uint, const Format_description_log_event*): Assertion ‘(bytes_read == (data_written - ((old_pre_checksum_fd || (description_event->checksum_alg == BINLOG_CHECKSUM_ALG_OFF)) ? 0 : 4))) || ((”.

85.16 *Percona Server for MySQL* 5.6.39-83.1

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.39-83.1 on February 13, 2018. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.6.39, including all the bug fixes in it, *Percona Server for MySQL* 5.6.39-83.1 is now the current stable release in the 5.6 series. All of *Percona*'s software is open-source and free.

85.16.1 Bugs Fixed

With **:variable:'innodb_large_prefix'** set to `1`, Blackhole storage engine was incompatible with InnoDB keys longer than 1000 bytes, thus adding new indexes would cause replication errors on the slave. Bug fixed #1126 (upstream #53588).

Intermediary slave with Blackhole storage engine couldn't record updates from master to its own binary log in case master has **:variable:'binlog_rows_query_log_events'** option enabled. Bug fixed #1119 (upstream #88057).

A build error on FreeBSD caused by fixing the bug #255 was present. Bug fixed #2284.

Server queries that contained JSON special characters and were logged by `audit_log` plugin in JSON format caused invalid output due to lack of escaping. Bug fixed #1115.

Compilation warnings fixed in sql_planner.cc module. Bug fixed #3632 (upstream #77637).

A memory leak fixed in PFS unit test. Bug fixed #1806 (upstream #89384).

A GCC 7 warning fix introduced regression in *Percona Server for MySQL* **:rn:'5.6.38-83.0'** that lead to a wrong SQL query built to access the remote server when Federated storage engine was used. Bug fixed #1134.

Percona Server for MySQL now uses *Travis CI* for additional tests. Bug fixed #3777.

Other bugs fixed: #257, #1090 (upstream #78048), #1127, and #2415.

This release also contains fixes for the following CVE issues: CVE-2018-2562, CVE-2018-2573, CVE-2018-2583, CVE-2018-2590, CVE-2018-2591, CVE-2018-2612, CVE-2018-2622, CVE-2018-2640, CVE-2018-2645, CVE-2018-2647, CVE-2018-2665, CVE-2018-2668, CVE-2018-2696, CVE-2018-2703, CVE-2017-3737.

85.16.2 TokuDB Changes

A memory leak was fixed in the PerconaFT library, caused by not destroying PFS key objects on shutdown. Bug fixed TDB-98.

A clang-format configuration was added to PerconaFT and TokuDB. Bug fixed TDB-104.

Other bugs fixed: TDB-48, TDB-78, TDB-93, and TDB-99.

85.17 *Percona Server for MySQL* 5.6.38-83.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.38-83.0 on December 8, 2017 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.38, including all the bug fixes in it, *Percona Server for MySQL* 5.6.38-83.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.38-83.0 milestone at Launchpad.

85.17.1 New Features

Percona Server for MySQL has implemented *InnoDB Page Fragmentation Counters*.

Percona Server for MySQL has implemented support for *Multiple page asynchronous I/O requests*. This feature was ported from a *Facebook MySQL* patch.

Percona Server for MySQL has implemented *TokuDB integration* with `PERFORMANCE_SCHEMA`.

Percona Server for MySQL packages are now available for *Ubuntu 17.10 (Artful)*.

85.17.2 Bugs Fixed

Dynamic row format feature to support `BLOB/VARCHAR` in `MEMORY` tables requires all the key columns to come before any `BLOB` columns. This requirement however was not enforced, allowing creating MEMORY tables in unsupported column configurations, which then crashed or lose data in usage. Bug fixed #1731483.

If an I/O syscall returned an error during the server shutdown with *Thread Pool* enabled, a mutex could be left locked. Bug fixed #1702330 (*Daniel Black*).

After fixing bug #1668602, bug #1539504, and bug #1313901, `CREATE/DROP TEMPORARY TABLE` statements were forbidden incorrectly in transactional contexts, including function and trigger calls, even when they required no binary logging at all. Bug fixed #1711781.

Running `ANALYZE TABLE` while a long-running query is accessing the same table in parallel could lead to a situation where new queries on the same table are blocked in a `Waiting for table flush` state. Fixed by stopping `ANALYZE TABLE` flushing affected *InnoDB* and *TokuDB* tables from the table definition cache. Bug fixed #1704195 (upstream #87065).

MyRocks additions to the 5.6 *mysqldump* output have been removed.

`CREATE TABLE ... LIKE ...` did not use source `row_format` on target *TokuDB* table. Bug fixed #76.

TokuDB would encode already encoded database name for a directory name. Bug fixed #74.

Other bugs fixed: #1670902 (upstream #85352), #1670902 (upstream #85352), #1729241, #83, #80, and #75.

85.18 Percona Server 5.6.37-82.2

Percona is glad to announce the release of Percona Server 5.6.37-82.2 on August 25, 2017. Downloads are available here and from the *Percona Software Repositories*.

This release is based on MySQL 5.6.37 and includes all the bug fixes in it. Percona Server 5.6.37-82.2 is now the current stable release in the 5.6 series. All software developed by Percona is open-source and free. Details of the release can be found in the 5.6.37-82.2 milestone on Launchpad.

Note: Red Hat Enterprise Linux 5 (including CentOS 5 and other derivatives), Ubuntu 12.04 and older versions are no longer supported by Percona software. The reason for this is that these platforms reached end of life, will not receive updates and are not recommended for use in production.

85.18.1 Bugs Fixed

- #1703105: Fixed overwriting of error log on server startup.
- #1705729: Fixed the `postinst` script to correctly locate the `datadir`.
- #1709834: Fixed the `mysqld_safe` script to correctly locate the `basedir`.
- Other fixes: #1706262

85.18.2 TokuDB Changes

- TDB-72: Fixed issue when renaming a table with non-alphanumeric characters in its name.

85.18.3 Platform Support

- Stopped providing packages for RHEL 5 (CentOS 5) and Ubuntu 12.04.

85.19 *Percona Server for MySQL* 5.6.36-82.1

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.36-82.1 on August 1, 2017 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.36, including all the bug fixes in it, *Percona Server for MySQL* 5.6.34-79.1 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.36-82.1 milestone at Launchpad.

85.19.1 New Features

Percona Server for MySQL can now be built with support of OpenSSL 1.1.

Percona Server for MySQL is now available on Debian 9 (stretch). The support only covers the `amd64` architecture.

TokuDB enables to kill a query that is awaiting an FT locktree lock.

85.19.2 Bugs Fixed

Row counts in *TokuDB* could be lost intermittently after restarts. Bug fixed #2.

In *TokuDB*, two races in the fractal tree lock manager could significantly affect transactional throughput for some applications that used a small number of concurrent transactions. These races manifested as transactions unnecessarily waiting for an available lock. Bug fixed #3.

TokuDB could assert when opening a dictionary with no useful information to error log. Bug fixed #23.

TokuDB could assert for various reasons deserializing nodes with no useful error output. Bug fixed #24.

Percona Server for MySQL could crash when running a query over a partitioned table that uses an index to read a range of rows if this range was not covered by any existing partition. Bug fixed #1657941 (upstream #76418).

With two client connections to a server (debug server build), the server could crash after one of the clients set the global option `userstat` and flushed the client statistics (`FLUSH CLIENT_STATISTICS`) and then both clients were closed. Bug fixed #1661488.

TokuDB did not pass cmake flags on to snappy cmake. Bug fixed #41. The progress status for partitioned TokuDB table ALTERs was misleading. Bug fixed #42.

When a client application connecting to the Aurora cluster end point using SSL (`--ssl-verify-server-cert` or `--ssl-mode=VERIFY_IDENTITY` option), wildcard and SAN enabled SSL certificates were ignored. See also *Compatibility Matrix*. Note that the `--ssl-verify-server-cert` option is deprecated in *Percona Server for MySQL* 5.7. Bug fixed #1673656 (upstream #68052).

Killing a stored procedure execution could result in an assert failure on a debug server build. Bug fixed #1689736 (upstream #86260).

It was not possible to build *Percona Server for MySQL* on Debian 9 (stretch) due to issues with OpenSSL 1.1. Bug fixed #1702903 (upstream #83814).

The `SET STATEMENT .. FOR` statement changed the global instead of the session value of a variable if the statement occurred immediately after the `SET GLOBAL` or `SHOW GLOBAL STATUS` command. Bug fixed #1385352.

The synchronization between the LRU manager and page cleaner threads was not done at shutdown. Bug fixed #1689552.

Other bugs fixed: #6, #44, #65, #1160986, #1676740, #1689989, #1689998, #1690012, #1699788, and #1684601 (upstream #86016).

85.19.3 Compatibility Matrix

Feature	YaSSL	OpenSSL < 1.0.2	OpenSSL >= 1.0.2
'commonName' validation	Yes	Yes	Yes
SAN validation	No	Yes	Yes
Wildcards support	No	No	Yes

85.20 *Percona Server for MySQL* 5.6.36-82.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.36-82.0 on May 12, 2017 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.36, including all the bug fixes in it, *Percona Server for MySQL* 5.6.34-79.1 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.36-82.0 milestone at Launchpad.

85.20.1 New Features

Percona Server for MySQL 5.6 packages are now available for Ubuntu 17.04 (*Zesty Zapus*).

Percona Server for MySQL now supports *Prefix Index Queries Optimization*.

:variable:'tokudb_dir_cmd' can now be used to *edit the TokuDB* files.

85.20.2 Bugs Fixed

Deadlock could occur in I/O-bound workloads when server was using several small buffer pool instances in combination with small redo log files and variable **:variable:'innodb_empty_free_list_algorithm'** set to `backoff` algorithm. Bug fixed #1651657.

Querying **:table:'TABLE_STATISTICS'** in combination with a stored function could lead to a server crash. Bug fixed #1659992.

`tokubackup_slave_info` file was created for a master server after taking the backup with *Percona TokuBackup*. Bug fixed #135.

Fixed a memory leak in *Percona TokuBackup*. Bug fixed #1669005.

Compressed columns with dictionaries could not be added to a partitioned table by using `ALTER TABLE`. Bug fixed #1671492.

Fixed a memory leak that happened in case of failure to create a multi-threaded slave worker thread. Bug fixed #1675716.

Combination of using any audit API-using plugin, like *Audit Log Plugin* and *Response Time Distribution*, with multi-byte collation connection and `PREPARE` statement with a parse error could lead to a server crash. Bug fixed #1688698 (upstream #86209).

Fix for a #1433432 bug in *Percona Server for MySQL* **:rn:'5.6.28-76.1'** caused a performance regression due to suboptimal LRU manager thread flushing heuristics. Bug fixed #1631309.

Creating *Compressed columns with dictionaries* in *MyISAM* tables by specifying partition engines would not result in error. Bug fixed #1631954.

It was not possible to configure basedir as a symlink. Bug fixed #1639735.

Replication slave did not report `Seconds_Behind_Master` correctly when running in multi-threaded slave mode. Bug fixed #1654091 (upstream #84415).

`DROP TEMPORARY TABLE` would create a transaction in binary log on a read-only server. Bug fixed #1668602 (upstream #85258).

Creating a compression dictionary with **:variable:'innodb_fake_changes'** enabled could lead to a server crash. Bug fixed #1629257.

Other bugs fixed: #1660828 (upstream #84786), #1664519 (upstream #84940), #1674299, #1683456, #1670588 (upstream #84173), #1672389, #1674507, #1674867, #1675623, #1650294, #1659224, #1660565, #1662908, #1669002, #1671473, #1673800, #1674284, #1676441, #1676705, #1676847 (upstream #85671), #1677130 (upstream #85678), #1677162, #1678692, #1678792, #1680510 (upstream #85838), #1683993, #1684012, #1684078, #1684264, and #1674281.

85.21 *Percona Server for MySQL* 5.6.35-81.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.35-81.0 on March 24th, 2017 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.35, including all the bug fixes in it, *Percona Server for MySQL* 5.6.35-80.1 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.35-81.0 milestone at Launchpad.

85.21.1 New Features

Percona Server for MySQL has implemented new **`mysqldump`** `--order-by-primary-desc` option. This feature tells `mysqldump` to take the backup by descending primary key order (`PRIMARY KEY DESC`) which can be useful if storage engine is using reverse order column for a primary key.

85.21.2 Bugs Fixed

When **:variable:'innodb_ft_result_cache_limit'** was exceeded by internal memory allocated by *InnoDB* during the FT scan not all memory was released which could lead to server assertion. Bug fixed #1634932 (upstream #83648).

Log tracking initialization did not find last valid bitmap data correctly, potentially resulting in needless redo log retracking or hole in the tracked LSN range. Bug fixed #1658055.

If *Audit Log Plugin* was unable to create file pointed by **:variable:'audit_log_file'**, server would crash during the startup. Bug fixed #1666496.

A `DROP TEMPORARY TABLE ...` for a table created by a `CREATE TEMPORARY TABLE ... SELECT ...` would get logged in the binary log on a disconnect with mixed mode replication. Bug fixed #1671013.

TokuDB did not use index with even if cardinality was good. Bug fixed #1671152.

Row-based replication events were not reflected in `Rows_updated` fields in the *User Statistics* `INFORMATION_SCHEMA` tables. Bug fixed #995624.

A long-running binary log commit would block `SHOW STATUS`, which in turn could block a number of of other operations such as client connects and disconnects. Bug fixed #1646100.

It was impossible to use column compression dictionaries with partitioned *InnoDB* tables. Bug fixed #1653104.

Diagnostics for OpenSSL errors have been improved. Bug fixed #1660339 (upstream #75311).

When `DuplicateWeedout` strategy was used for joins, use was not reported in the query plan info output extension for the slow query log. Bug fixed #1592694.

Other bugs fixed: #1650321, #1650322, #1654501, #1663251, #1666213, #1652912, #1659548, #1663452, #1670834, #1672871, #1626545, #1644174, #1658006, #1658021, #1659218, #1659746, #1660239, #1660243, #1660255, #1660348, #1662163 upstream (#81467), #1664219, #1664473, #1671076, and #1671123.

85.22 *Percona Server for MySQL* 5.6.35-80.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.35-80.0 on February 8th, 2017 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.35, including all the bug fixes in it, *Percona Server for MySQL* 5.6.35-80.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.35-80.0 milestone at Launchpad.

85.22.1 New Features

Kill Idle Transactions feature has been re-implemented by setting a connection socket read timeout value instead of periodically scanning the internal *InnoDB* transaction list. This makes the feature applicable to any transactional storage engine, such as *TokuDB*, and, in future, *MyRocks*. This re-implementation is also addressing some existing bugs, including server crashes: #1166744, #1179136, #907719, and #1369373.

85.22.2 Bugs Fixed

Logical row counts for *TokuDB* tables could get inaccurate over time. Bug fixed #1651844 (#732).

Repeated execution of `SET STATEMENT ... FOR <SELECT FROM view>` could lead to a server crash. Bug fixed #1392375.

`CREATE TEMPORARY TABLE` would create a transaction in binary log on a read only server. Bug fixed #1539504 (upstream #83003).

If temporary tables from `CREATE TABLE ... AS SELECT` contained compressed attributes it could lead to server crash. Bug fixed #1633957.

Using *Per-query variable statement* with subquery temporary tables could cause a memory leak. Bug fixed #1635927.

Fixed new compilation warnings with GCC 6. Bugs fixed #1641612 and #1644183.

A server could crash if a bitmap write I/O error happens in the background log tracking thread while a `FLUSH CHANGED_PAGE_BITMAPS` is executing concurrently. Bug fixed #1651656.

TokuDB was using wrong function to calculate free space in data files. Bug fixed #1656022 (#1033).

`CONCURRENT_CONNECTIONS` column in the **:table:'USER_STATISTICS'** table was showing incorrect values. Bug fixed #728082.

InnoDB index dives did not detect some of the concurrent tree changes, which could return bogus estimates. Bug fixed #1625151 (upstream #84366).

:table:'INFORMATION_SCHEMA.INNODB_CHANGED_PAGES' queries would needlessly read potentially incomplete bitmap data past the needed LSN range. Bug fixed #1625466.

Percona Server for MySQL `cmake` compiler would always attempt to build *RocksDB* even if `-DWITHOUT_ROCKSDB=1` argument was specified. Bug fixed #1638455.

Adding `COMPRESSED` attributes to *InnoDB* special tables fields (like `mysql.innodb_index_stats` and `mysql.innodb_table_stats`) could lead to server crashes. Bug fixed #1640810.

Lack of free pages in the buffer pool is not diagnosed with **:variable:'innodb_empty_free_list_algorithm'** set to `backoff` (which is the default). Bug fixed #1657026.

`mysqld_safe` now limits the use of `rm` and `chown` to avoid privilege escalation. `chown` can now be used only for `/var/log` directory. Bug fixed #1660265. Thanks to Dawid Golunski (https://legalhackers.com).

Renaming a *TokuDB* table to a non-existent database with **:variable:'tokudb_dir_per_db'** enabled would lead to a server crash. Bug fixed #1030.

Read Free Replication optimization could not be used for *TokuDB* partition tables. Bug fixed #1012.

Other bugs fixed: #1486747 (upstream #76872), #1633988, #1638198 (upstream #82823), #1638897, #1646384, #1647530, #1647741, #1651121, #1156772, #1644569, #1644583, #1648389, #1648737, #1650247, #1650256, #1650324, #1650450, #1655587, and #1647723.

85.23 *Percona Server for MySQL* 5.6.34-79.1

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.34-79.1 on November 23rd, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.34, including all the bug fixes in it, *Percona Server for MySQL* 5.6.34-79.1 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.34-79.1 milestone at Launchpad.

85.23.1 Deprecated Features

Metrics for scalability measurement feature is now deprecated. Users who have installed this plugin but are not using its capability are advised to uninstall the plugin due to known crashing bugs.

85.23.2 Bugs Fixed

When a stored routine would call an administrative command such as `OPTIMIZE TABLE`, `ANALYZE TABLE`, `ALTER TABLE`, `CREATE/DROP INDEX`, etc. the effective value of **:variable:'log_slow_sp_statements'** was overwritten by the value of **:variable:'log_slow_admin_statements'**. Bug fixed #719368.

Thread Pool `thread limit reached` and `failed to create thread` messages are now printed on the first occurrence as well. Bug fixed #1636500.

:table:'INFORMATION_SCHEMA.TABLE_STATISTICS' and **:table:'INFORMATION_SCHEMA.INDEX_STATISTICS'** tables were not correctly updated for *TokuDB*. Bug fixed #1629448.

Other bugs fixed: #1633061, #1633430, and #1635184.

85.24 *Percona Server for MySQL* 5.6.33-79.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.33-79.0 on October 18th, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.33, including all the bug fixes in it, *Percona Server for MySQL* 5.6.33-79.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.33-79.0 milestone at Launchpad.

85.24.1 New Features

Percona Server for MySQL has implemented support for per-column `VARCHAR/BLOB` *compression* for the *XtraDB* storage engine. This also features compression dictionary support, to improve compression ratio for relatively short individual rows, such as JSON data.

A new *TokuDB* **:variable:'tokudb_dir_per_db'** option has been introduced to address two *TokuDB* shortcomings, the *renaming of data files* on table/index rename, and the ability to *group data files together* within a directory that represents a single database. This feature is disabled by default.

85.24.2 Bugs Fixed

After fixing bug #1540338, system table engine validation check is no longer skipped for tables that don't have an explicit `ENGINE` clause in a `CREATE TABLE` statement. If *MySQL* upgrade statements are replicated, and slave does not have the *MyISAM* set as a default storage engine, then the `CREATE TABLE mysql.server` statement would attempt to create an *InnoDB* table and fail because `mysql_system_tables.sql` script omitted explicit engine setting for this table. Bug fixed #1600056.

Audit Log Plugin malformed record could be written after **:variable:'audit_log_flush'** was set to `ON` in `ASYNC` and `PERFORMANCE` modes. Bug fixed #1613650.

A race condition between *HandlerSocket* and server shutdown could lead to a server stall if shutdown was issued immediately after *HandlerSocket* startup. Bug fixed #1617198.

HandlerSocket could access freed memory on startup. Bug fixed #1617998.

Workloads with statements that take non-transactional locks (`LOCK TABLES`, global read lock, and similar) could have caused deadlocks when running under *Thread Pool* with high priority queue enabled and **:variable:'thread_pool_high_prio_mode'** set to `transactions`. Fixed by placing such statements into the high priority queue even with the above **:variable:'thread_pool_high_prio_mode'** setting. Bugs fixed #1619559 and #1374930.

Fixed memory leaks in *Audit Log Plugin*. Bug fixed #1620152 (upstream #71759).

Server could crash due to a `glibc` bug in handling short-lived detached threads. Bug fixed #1621012 (upstream #82886).

`QUERY_RESPONSE_TIME_READ` and `QUERY_RESPONSE_TIME_WRITE` were returning `QUERY_RESPONSE_TIME` table data if accessed through a name that is not full uppercase. Bug fixed #1552428.

Fixed memory leaks in *HandlerSocket*. Bug fixed #1617949.

`KILL QUERY` was not behaving consistently and it would hang in some cases. Bug fixed #1621046 (upstream #45679).

Cipher `ECDHE-RSA-AES128-GCM-SHA256` was listed in the list of supported ciphers but it wasn't supported. Bug fixed #1622034 (upstream #82935).

Successful doublewrite recovery was showed as a warning in the error log. Bug fixed #1622985.

Variable **:variable:'query_cache_type'** couldn't be set to `0` if it was already set to that value. Bug fixed #1625501 (upstream #69396).

LRU manager thread could run too long on a server shutdown, causing a server crash. Bug fixed #1626069.

`tokudb_default` was not recognized by *Percona Server for MySQL* as a valid row format. Bug fixed #1626206.

InnoDB `ANALYZE TABLE` didn't remove its table from the background statistics processing queue. Bug fixed #1626441 (upstream #71761).

Upstream merge for #81657 to 5.6 was incorrect. Bug fixed #1626936 (upstream #83124).

Fixed multi-threaded slave thread leaks that happened in case of thread create failure. Bug fixed #1619622 (upstream #82980).

Shutdown waiting for a purge to complete was undiagnosed for the first minute. Bug fixed #1616785.

Unnecessary *InnoDB* change buffer merge attempts are now skipped upon reading disk pages of non-applying types. Bug fixed #1618393 (upstream #75235).

Other bugs fixed: #1614439, #1614949, #1616392 (upstream #82798), #1624993 (#736), #1613647, #1626002 (upstream #83073), #904714, #1610102, #1610110, #1613663, #1613728, #1613986, #1614885, #1615959, #1615970, #1616333, #1616404, #1616768, #1617150, #1617216, #1617267, #1618478, #1618811, #1618819, #1619547, #1619572, #1620583, #1622449, #1622456, #1622977, #1623011, #1624992 (#1014), #1625176, #1625187, #1626500, #1628417, #964, and #735.

85.25 *Percona Server for MySQL* 5.6.32-78.1

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.32-78.1 on September 21st, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.32, including all the bug fixes in it, *Percona Server for MySQL* 5.6.32-78.1 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.32-78.1 milestone at Launchpad.

85.25.1 Bugs Fixed

Limiting `ld_preload` libraries to be loaded from specific directories in `mysqld_safe` didn't work correctly for relative paths. Bug fixed #1624247.

Fixed possible privilege escalation that could be used when running `REPAIR TABLE` on a `MyISAM` table. Bug fixed #1624397.

The general query log and slow query log cannot be written to files ending in `.ini` and `.cnf` anymore. Bug fixed #1624400.

Implemented restrictions on symlinked files (`error_log`, `pid_file`) that can't be used with `mysqld_safe`. Bug fixed #1624449.

Other bugs fixed: #1553938.

85.26 *Percona Server for MySQL* 5.6.32-78.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.32-78.0 on August 22th, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.32, including all the bug fixes in it, *Percona Server for MySQL* 5.6.32-78.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.32-78.0 milestone at Launchpad.

85.26.1 New Features

Percona Server for MySQL *Audit Log Plugin* now supports filtering by *user* and *sql_command*.

Percona Server for MySQL now supports tree map file block allocation strategy for *TokuDB*.

85.26.2 Bugs Fixed

Fixed potential cardinality `0` issue for *TokuDB* tables if `ANALYZE TABLE` finds only deleted rows and no actual logical rows before it times out. Bug fixed #1607300 (#1006, #732).

TokuDB `database.table.index` names longer than 256 characters could cause server crash if *background analyze table* status was checked while running. Bug fixed #1005.

PAM Authentication Plugin would abort authentication while checking UNIX user group membership if there were more than a thousand members. Bug fixed #1608902.

If `DROP DATABASE` would fail to delete some of the tables in the database, the partially-executed command is logged in the binlog as `DROP TABLE t1, t2, ...` for the tables for which drop succeeded. A slave might fail to replicate such `DROP TABLE` statement if there exist foreign key relationships to any

of the dropped tables and the slave has a different schema from master. Fix by checking, on the master, whether any of the database to be dropped tables participate in a Foreign Key relationship, and fail the `DROP DATABASE` statement immediately. Bug fixed #1525407 (upstream #79610).

PAM Authentication Plugin didn't support spaces in the UNIX user group names. Bug fixed #1544443.

Due to security reasons `ld_preload` libraries can now only be loaded from the system directories (`/usr/lib64`, `/usr/lib`) and the MySQL installation base directory.

Percona Server for MySQL 5.6 could not be built with the `-DMYSQL_MAINTAINER_MODE=ON` option. Bug fixed #1590454.

In the client library, any EINTR received during network I/O was not handled correctly. Bug fixed #1591202 (upstream #82019).

The included `.gitignore` in the percona-server source distribution had a line `*.spec`, which means someone trying to check in a copy of the percona-server source would be missing the spec file required to build the RPMs. Bug fixed #1600051.

Audit Log Plugin did not transcode queries. Bug fixed #1602986.

LeakSanitizer-enabled build failed to bootstrap server for MTR. Bug fixed #1603978 (upstream #81674).

Fixed `MYSQL_SERVER_PUBLIC_KEY` connection option memory leak. Bug fixed #1604419.

The fix for bug #1341067 added a call to free some of the heap memory allocated by OpenSSL. This is not safe for repeated calls if OpenSSL is linked twice through different libraries and each is trying to free the same. Bug fixed #1604676.

If the changed page bitmap redo log tracking thread stops due to any reason, then shutdown will wait for a long time for the log tracker thread to quit, which it never does. Bug fixed #1606821.

Audit Log Plugin events did not report the default database. Bug fixed #1435099.

Canceling the *TokuDB Background ANALYZE TABLE* job twice or while it was in the queue could lead to server assertion. Bug fixed #1004.

Fixed various spelling errors in comments and function names. Bug fixed #728 (*Otto Kekäläinen*).

Implemented set of fixes to make PerconaFT build and run on the AArch64 (64-bit ARMv8) architecture. Bug fixed #726 (*Alexey Kopytov*).

Other bugs fixed: #1603073, #1604323, #1604364, #1604462, #1604774, #1606782, #1607224, #1607359, #1607606, #1607607, #1607671, #1608385, #1608437, #1608845, #1609422, #1610858, #1612084, #1612551, #1455430, #1455432, #1610242, #998, #1003, #729, and #730.

85.27 *Percona Server for MySQL* 5.6.31-77.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.31-77.0 on July 7th, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.31, including all the bug fixes in it, *Percona Server for MySQL* 5.6.31-77.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.31-77.0 milestone at Launchpad.

85.27.1 New Features

Percona Server for MySQL has implemented protocol support for *TLS 1.1 and TLS 1.2*. This implementation turns off TLS v1.0 support by default.

TokuDB MTR suite is now part of the default MTR suite in *Percona Server for MySQL* 5.6.

85.27.2 Bugs Fixed

Querying the **:table:'GLOBAL_TEMPORARY_TABLES'** table would cause server crash if temporary table owning threads would execute new queries. Bug fixed #1581949.

Audit Log Plugin would hang when trying to write log record of **:variable:'audit_log_buffer_size'** length. Bug fixed #1588439.

Audit log in `ASYNC` mode could skip log records which don't fit into log buffer. Bug fixed #1588447.

The **:variable:'innodb_log_block_size'** feature attempted to diagnose the situation where the logs have been created with a log block value that differs from the current **:variable:'innodb_log_block_size'** setting. But this diagnostics came too late, and a misleading error `No valid checkpoints found` was produced first, aborting the startup. Bug fixed #1155156.

Some transaction deadlocks did not increase the **:table:'INFORMATION_SCHEMA.INNODB_METRICS'** `lock_deadlocks` counter. Bug fixed #1466414 (upstream #77399).

InnoDB tablespace import would fail when trying to import a table with different data directory. Bug fixed #1548597 (upstream #76142).

Audit Log Plugin was truncating SQL queries to 512 bytes. Bug fixed #1557293.

Regular user extra port connection would fail if **:variable:'max_connections'** plus one `SUPER` user were already connected on the main port, even if it connecting would not violate the **:variable:'extra_max_connections'**. Bug fixed #1583147.

The error log warning `Too many connections` was only printed for connection attempts when **:variable:'max_connections'** plus one `SUPER` have connected. If the extra `SUPER` is not connected, the warning was not printed for a non-SUPER connection attempt. Bug fixed #1583553.

`mysqlbinlog` did not free the existing connection before opening a new remote one. Bug fixed #1587840 (upstream #81675).

Fixed memory leaks in `mysqltest`. Bugs fixed #1582718 and #1588318.

Fixed memory leaks in `mysqlcheck`. Bug fixed #1582741.

Fixed memory leak in `mysqlbinlog`. Bug fixed #1582761 (upstream #78223).

Fixed memory leaks in `mysqldump`. Bug fixed #1587873 and #1588845 (upstream #81714).

Fixed memory leak in non-existing defaults file handling. Bug fixed #1588344.

Fixed memory leak in `mysqlslap`. Bug fixed #1588361.

Transparent Huge Pages check will now only happen if **:variable:'tokudb_check_jemalloc'** option is set. Bugs fixed #939 and #713.

Logging in `ydb` environment validation functions now prints more useful context. Bug fixed #722.

Other bugs fixed: #1588386, #1529885, #1541698 (upstream #80261), #1582681, #1583589, #1587426 (upstream, #81657), #1589431, #956, and #964.

85.28 *Percona Server for MySQL* 5.6.30-76.3

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.30-76.3 on May 25th, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.30, including all the bug fixes in it, *Percona Server for MySQL* 5.6.30-76.3 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.30-76.3 milestone at Launchpad.

85.28.1 Bugs Fixed

When *Read Free Replication* was enabled for *TokuDB* and there was no explicit primary key for the replicated *TokuDB* table there could be duplicated records in the table on update operation. The fix disables *Read Free Replication* for tables without explicit primary key and does rows lookup for `UPDATE` and `DELETE` binary log events and issues warning. Bug fixed #1536663 (#950).

Attempting to execute a non-existing prepared statement with *Response Time Distribution* plugin enabled could lead to a server crash. Bug fixed #1538019.

TokuDB was using using different memory allocators, this was causing `safemalloc` warnings in debug builds and crashes because memory accounting didn't add up. Bug fixed #1546538 (#962).

Fixed heap allocator/deallocator mismatch in *Metrics for scalability measurement*. Bug fixed #1581051.

Percona Server for MySQL is now built with system `zlib` library instead of the older bundled one. Bug fixed #1108016.

Reduced the memory overhead per page in the *InnoDB* buffer pool. The fix was based on Facebook patch #91e979e. Bug fixed #1536693 (upstream #72466).

`CREATE TABLE ... LIKE ...` could create a system table with an unsupported enforced engine. Bug fixed #1540338.

Change buffer merge could throttle to 5% of I/O capacity on an idle server. Bug fixed #1547525.

`Slave_open_temp_tables` would fail to decrement on the slave with disabled binary log if master was killed. Bug fixed #1567361.

Server will now show more descriptive error message when *Percona Server for MySQL* fails with `errno == 22 "Invalid argument"`, if **:variable:'innodb_flush_method'** was set to `ALL_O_DIRECT`. Bug fixed #1578604.

Killed connection threads could get their sockets closed twice on shutdown. Bug fixed #1580227.

`AddressSanitizer` build with `LeakSanitizer` enabled was failing at `gen_lex_hash` invocation. Bug fixed #1580993 (upstream #80014).

`apt-cache show` command for `percona-server-client` was showing `innotop` included as part of the package. Bug fixed #1201074.

`mysql-systemd` would fail with PAM authentication and proxies due to regression introduced when fixing #1534825 in *Percona Server for MySQL* **:rn:'5.6.29-76.2'**. Bug fixed #1558312.

Upgrade logic for figuring if *TokuDB* upgrade can be performed from the version on disk to the current version was broken due to regression introduced when fixing #684 in *Percona Server for MySQL* **:rn:'5.6.27-75.0'**. Bug fixed #717.

If `ALTER TABLE` was run while **:variable:'tokudb_auto_analyze'** variable was enabled it would trigger auto-analysis, which could lead to a server crash if `ALTER TABLE DROP KEY` was used because it would be operating on the old table/key meta-data. Bug fixed #945.

The *TokuDB* storage engine with **:variable:'tokudb_pk_insert_mode'** set to `1` is safe to use in all conditions. On `INSERT IGNORE` or `REPLACE INTO`, it tests to see if triggers exist on the table, or replication is active with `!BINLOG_FORMAT_STMT` before it allows the optimization. If either of these conditions are met, then it falls back to the "safe" operation of looking up the target row first. Bug fixed #952.

Bug in *TokuDB* Index Condition Pushdown was causing `ORDER BY DESC` to reverse the scan outside of the *WHERE* bounds. This would cause query to hang in a `sending data` state for several minutes in some environments with large amounts of data (3 billion records) if the `ORDER BY DESC` statement was used. Bugs fixed #988, #233, and #534.

Other bugs fixed: #1399562 (upstream #75112), #1510564 (upstream #78981), #1496282 (#964), #1496786 (#956), #1566790, #1552673, #1567247, #1567869, #718, #914, #970, #971, #972, #976, #977, #981, #637, and #982.

85.29 *Percona Server for MySQL* 5.6.29-76.2

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.29-76.2 on March 7th, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.29, including all the bug fixes in it, *Percona Server for MySQL* 5.6.29-76.2 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.29-76.2 milestone at Launchpad.

85.29.1 Bugs Fixed

With *Expanded Fast Index Creation* enabled, DDL queries involving *InnoDB* temporary tables would cause later queries on the same tables to produce warnings that their indexes were not found in the index translation table. Bug fixed #1233431.

Package upgrade on *Ubuntu* would run `mysql_install_db` even though data directory already existed. Bug fixed #1457614.

Package upgrade on *Ubuntu* and *Debian* could reset the `GTID` number sequence when post-install script was restarting the service. Bug fixed #1507812.

Starting *MySQL* with `systemctl` would fail with timeout if the socket was specified with a custom path. Bug fixed #1534825.

Write-heavy workload with a small buffer pool could lead to a deadlock when free buffers are exhausted. Bug fixed #1521905.

`libjemalloc.so.1` was missing from binary tarball. Bug fixed #1537129.

`mysqldumpslow` script has been removed because it was not compatible with *Percona Server for MySQL* extended slow query log format. Please use pt-query-digest from *Percona Toolkit* instead. Bug fixed #856910.

When `cmake/make/make_binary_distribution` workflow was used to produce binary tarballs it would produce tarballs with `mysql-...` naming instead of `percona-server-...`. Bug fixed #1540385.

Cardinality of partitioned *TokuDB* tables became inaccurate after the changes introduced by *TokuDB Background ANALYZE TABLE* feature in *Percona Server for MySQL* **:rn:'5.6.27-76.0'**. Bug fixed #925.

Running the `TRUNCATE TABLE` while *TokuDB Background ANALYZE TABLE* is enabled could lead to a server crash once analyze job tries to access the truncated table. Bug fixed #938.

Added proper memory cleanup if for some reason a *TokuDB* table is unable to be opened from a dead closed state. This prevents an assertion from happening the next time the table is attempted to be opened. Bug fixed #917.

Other bugs fixed: #898, #1521120 and #1534246.

85.30 *Percona Server for MySQL* 5.6.28-76.1

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.28-76.1 on January 12th, 2016 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.28, including all the bug fixes in it, *Percona Server for MySQL* 5.6.28-76.1 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.28-76.1 milestone at Launchpad.

85.30.1 Bugs Fixed

Clustering secondary index could not be created on a partitioned *TokuDB* table. Bug fixed #1527730 (#720).

When enabled, *super-read-only option* could break statement-based replication while executing a multi-table update statement on a slave. Bug fixed #1441259.

Running `OPTIMIZE TABLE` or `ALTER TABLE` without the `ENGINE` clause would silently change table engine if **:variable:'enforce_storage_engine'** variable was active. This could also result in system tables being changed to incompatible storage engines, breaking server operation. Bug fixed #1488055.

Setting the **:variable:'innodb_sched_priority_purge'** (available only in debug builds) while purge threads were stopped would cause a server crash. Bug fixed #1368552.

Small buffer pool size could cause *XtraDB* buffer flush thread to spin at 100% CPU. Bug fixed #1433432.

Enabling *TokuDB* with `ps_tokudb_admin` script inside the Docker container would cause an error due to insufficient privileges even when running as root. In order for this script to be used inside docker containers this error has be been changed to a warning that a check is impossible. Bug fixed #1520890.

InnoDB status will start printing negative values for spin rounds per wait, if the wait number, even though being accounted as a signed 64-bit integer, will not fit into a signed 32-bit integer. Bug fixed #1527160 (upstream #79703).

Other bugs fixed: #1384595 (upstream #74579), #1384658 (upstream #74619), #1471141 (upstream #77705), #1179451, #1524763 and #1530102.

85.31 *Percona Server for MySQL* 5.6.27-76.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.27-76.0 on December 4th, 2015 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.27, including all the bug fixes in it, *Percona Server for MySQL* 5.6.27-76.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.27-76.0 milestone at Launchpad.

85.31.1 New Features

`SHOW SLAVE STATUS NOLOCK` syntax in 5.6 has been undeprecated. Both `SHOW SLAVE STATUS NOLOCK` and `SHOW SLAVE STATUS NONBLOCKING` are now supported. *Percona Server for MySQL* originally used `SHOW SLAVE STATUS NOLOCK` syntax for this feature. As of **:rn:'5.6.20-68.0'** release, *Percona Server for MySQL* implements `SHOW SLAVE STATUS NONBLOCKING` syntax, which comes from early *MySQL* 5.7. Current *MySQL* 5.7 does not have this syntax and regular `SHOW SLAVE STATUS` is non-blocking.

TokuDB tables can now be automatically *analyzed in the background* based on a measured change in data.

Percona Server for MySQL has implemented new **:variable:‘tokudb_strip_frm_data‘** variable which can be used to assist in *TokuDB* data recovery. **WARNING:** Use this variable only if you know what you’re doing otherwise it could lead to data loss.

85.31.2 Bugs Fixed

Setting the **:variable:‘tokudb_backup_last_error_string‘** and **:variable:‘tokudb_backup_last_error‘** values manually could cause server assertion. Bug fixed #1512464.

Fixed invalid memory accesses when **mysqldump** was running with `--innodb-optimize-keys` option. Bug fixed #1517444.

Fixed incorrect filename specified in `storage/tokudb/PerconaFT/buildheader/CMakeLists.txt` which could cause subsequent builds to fail. Bug fixed #1510085 (*Sergei Golubchik*).

Fixed multiple issues with *TokuDB* CMake scripts. Bugs fixed #1510092, #1509219 and #1510081 (*Sergei Golubchik*).

An upstream fix for upstream bug #76135 might cause server to stall or hang. Bug fixed #1519094 (upstream #79185).

ps_tokudb_admin now prevents *Percona TokuBackup* activation if there is no *TokuDB* storage engine on the server. Bug fixed #1520099.

Percona TokuBackup plugin now gets removed during the *TokuDB* storage engine uninstall process. Bug fixed #1520472.

New `--defaults-file` option has been implemented for **ps_tokudb_admin** to specify the *MySQL* configuration file if it’s not in the default location. Bug fixed #1517021.

Other bugs fixed: #1425480, #1517523, #1515741 and #1513178.

85.32 *Percona Server for MySQL* 5.6.27-75.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.27-75.0 on November 5th, 2015 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.27, including all the bug fixes in it, *Percona Server for MySQL* 5.6.27-75.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*‘s software is open-source and free, all the details of the release can be found in the 5.6.27-75.0 milestone at Launchpad.

85.32.1 New Features

Percona Server for MySQL is now available for *Ubuntu* 15.10 (Wily).

TokuDB MTR tests have been integrated into *Percona Server for MySQL*.

Linux thread ID is now available in the **:table:‘PROCESSLIST‘** table.

Percona Server for MySQL has now re-enabled savepoints in triggers and stored functions.

Variables **:variable:‘innodb_buffer_pool_populate‘** and **:variable:‘numa_interleave‘** have been mapped to the upstream implementation of the new innodb_numa_interleave option.

85.32.2 Bugs Fixed

Fixed transactional inconsistency with rollback in *TokuDB*. Rolling back a large transaction with many inserts/updates/deletes could result in some of the changes being committed rather than rolled back. Bug fixed #1512455.

Variable **:variable:‘tokudb_backup_exclude‘** was not excluding files correctly. Bug fixed #1512457.

TokuDB could crash under load if transaction-isolation level `READ-COMMITTED` was used. Bug fixed #1501633.

TokuDB thread pool names were missing in the `SHOW ENGINE tokudb STATUS` which caused duplicate entries. Bug fixed #1512452.

Manipulating the **:variable:‘innodb_track_redo_log_now‘** variable dynamically would crash the server if it was started without **:variable:‘innodb_track_changed_pages‘** enabled. This variable is available on debug builds only. Bug fixed #1368530.

If the user had duplicate pid-file options in config files when running `yum upgrade`, the upgrade would stop with error because it would think it found the duplicate pid while it was the same pid specified twice. Bug fixed #1454917.

On some filesystems server would not start if *XtraDB changed page tracking* feature was enabled and **:variable:‘innodb_flush_method‘** variable was set to `O_DIRECT`. Bugs fixed #1500720 and #1498891.

When *User Statistics* are enabled, executing any statement of the `SHOW` family with non-empty result, would bump **:table:‘USER_STATISTICS‘** `ROWS_FETCHED` column values erroneously. Bug fixed #1510953.

A write operation with **:variable:‘innodb_fake_changes‘** enabled could cause a server assertion if it followed the pessimistic B-tree update path internally. Bug fixed #1192898.

An online DDL operation could have caused server crash with fake changes enabled. Bug fixed #1226532.

Fixed the conflicting meta packages between 5.1, 5.5, and 5.6 release series in *Debian* and *Ubuntu* distributions. `percona-server-server` and `percona-server-client` meta packages now point to the latest 5.6 release. Bug fixed #1292517.

:table:‘INNODB_CHANGED_PAGES‘ table was unavailable with non-default **:variable:‘innodb_data_home_dir‘** setting if the variable had a trailing slash. Bug fixed #1364315.

Changing **:variable:‘innodb_fake_changes‘** variable value in the middle of a transaction would have an immediate effect, that is, making part of the transaction run with fake changes enabled and the rest with fake changes disabled, resulting in a broken transaction. Fixed by making any **:variable:‘innodb_fake_changes‘** value changes becoming effective at the start of the next transaction instead of the next statement. Bug fixed #1395579.

`UPDATE` statement could crash the server with *Support for Fake Changes* enabled. Bug fixed #1395706.

Startup would fail due to a small hard-coded timeout value in the init script for the pid file to appear. This has been fixed by creating default file for *Debian* init script timeout parameters in `etc/default/mysql`. Bug fixed #1434022.

`CMakeLists.txt` for `tokudb-backup-plugin` was missing Valgrind dependency. Bug fixed #1494283.

Percona Server for MySQL would fail to install on *CentOS* 7 if `mariadb-devel` package was already installed. Bug fixed #1499721.

Fixed suboptimal *Support for Fake Changes* handling in online `ALTER` storage engine API. Bug fixed #1204422.

The upstream bug #76627 was not fixed for the `ALL_O_DIRECT` case. Bug fixed #1500741.

Fixed multiple *TokuDB* clang build issues. Bug fixed #1512449.

Other bugs fixed: #1204443, #1384632, #1475117, #1512301, #1452397, #1160960, #1495965, and #1497942.

85.33 *Percona Server for MySQL* 5.6.26-74.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.26-74.0 on September 15th, 2015 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.26, including all the bug fixes in it, *Percona Server for MySQL* 5.6.26-74.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.26-74.0 milestone at Launchpad.

85.33.1 New Features

TokuDB storage engine source has been merged into the *Percona Server for MySQL* code. *TokuDB* storage engine **:variable:'tokudb_version'** variable now has the same value as the *Percona Server for MySQL* version variable **:variable:'version'**.

TokuDB Hot Backup has been renamed to *Percona TokuBackup* and it is now open source. Source code has been integrated into *Percona Server for MySQL* code as a git submodule. *TokuDB* Hot Backup plugin source has been merged into *Percona Server for MySQL* code.

Tokutek Fractal Tree has been renamed to *Percona FT* and its source code has been integrated into *Percona Server for MySQL* code as a git submodule.

TokuDB tests for *Percona Server for MySQL* 5.6 have been merged into *Percona Server for MySQL* 5.6 code.

Google SNAPPY compression/decompression algorithm is now available as *TokuDB Compression* table format.

Percona Server for MySQL now supports changing the **:variable:'server_id'** variable *per session*, by implementing the new **:variable:'pseudo_server_id'** variable. This feature is also fixing upstream bug #35125.

Percona Server for MySQL has temporarily disabled savepoints in triggers and stored functions. The reason is that even having fixed bug #1438990 and bug #1464468 we have found more cases where savepoints in triggers break binary logging and replication, resulting in server crashes and broken slaves. This feature will be disabled until the above issues are properly resolved.

`LOCK TABLES FOR BACKUP` now *flushes the current binary log coordinates* to *InnoDB*. Thus, under active `LOCK TABLES FOR BACKUP`, the binary log coordinates in *InnoDB* are consistent with its redo log and any non-transactional updates (as the latter are blocked by `LOCK BINLOG FOR BACKUP`). It is planned that this change will enable *Percona XtraBackup* to avoid issuing the more invasive `LOCK BINLOG FOR BACKUP` command under some circumstances.

`innodb_stress` has been added to the list of default MTR suites. For most supported systems satisfying the newly added dependencies is straightforward, but on *CentOS* 5, the default Python is too old. Thus `python26` and `python26-mysqldb` packages should be installed there and `python26` should be made the default python for the testsuite environment.

Three new *TokuDB* variables, **:variable:'tokudb_client_pool_threads'**, **:variable:'tokudb_cachetable_pool_threads'**, and **:variable:'tokudb_checkpoint_pool_threads'**, have been implemented to improve the controlling of thread pool size.

Percona Server for MySQL has implemented new **:variable:‘tokudb_enable_partial_eviction‘** option in *TokuDB* to allow disabling of partial eviction of nodes.

Percona Server for MySQL has implemented new **:variable:‘tokudb_compress_buffers_before_eviction‘** option in *TokuDB* which allows the evictor to compress unused internal node partitions in order to reduce memory requirements as a first step of partial eviction before fully evicting the partition and eventually the entire node.

85.33.2 Bugs Fixed

Querying **:table:‘GLOBAL_TEMPORARY_TABLES‘** table would crash threads working with internal temporary tables used by `ALTER TABLE`. Bug fixed #1113388.

Selecting from **:table:‘GLOBAL_TEMPORARY_TABLES‘** table while running an online `ALTER TABLE` on a partitioned table in parallel could lead to a server crash. Bug fixed #1193264.

Kill Idle Transactions feature could cause an assertion on a debug build due to a race condition. Bug fixed #1206008.

`libmylsqclient_16` symbols were missing in *Percona Server for MySQL* shared library package on *RHEL/CentOS* 7. Bug fixed #1420691.

Prepared statements in stored procedures could crash *Response Time Distribution* plugin. Bug fixed #1426345.

When variable **:variable:‘innodb_corrupt_table_action‘** is set to `Warn/Salvage` then server could crash on updating table statistics during query execution on affected tables. Bug fixed #1426610.

A sequence of failing `TRUNCATE TABLE`, then insert to that table, and `CHECK TABLE` would crash the server. Bug fixed #1433197.

When *InnoDB* change buffering was enabled and used, executing a `FLUSH TABLE ... FOR EXPORT` would cause a server hang and `SHOW PROCESSLIST` would show that table in a `System Lock` state. Bug fixed #1454441 (upstream #77011).

`FLUSH INDEX_STATISTICS / FLUSH CHANGED_PAGE_BITMAPS` and `FLUSH USER_STATISTICS / RESET CHANGE_PAGE_BITMAPS` pairs of commands were inadvertently joined, i.e. issuing either command had the effect of both. The first pair, besides flushing both index statistics and changed page bitmaps, had the effect of `FLUSH INDEX_STATISTICS` requiring `SUPER` instead of `RELOAD` privilege. The second pair resulted in `FLUSH USER_STATISTICS` destroying changed page bitmaps. Bug fixed #1472251.

Enabling **:variable:‘super_read_only‘** together with **:variable:‘read_only‘** in `my.cnf` would result in server crashing on startup. The workaround is to enable **:variable:‘super_read_only‘** dynamically on a running server.Bug fixed #1389935 (the fix was ported from Facebook patch #14d5d9).

Enabling **:variable:‘super_read_only‘** as a command line option would not enable **:variable:‘read_only‘**. Bug fixed #1389935 (the fix was ported from Facebook patch #14d5d9).

If a new connection thread was created while a `SHOW PROCESSLIST` command or a **:table:‘INFORMATION_SCHEMA.PROCESSLIST‘** query was in progress, it could have a negative TIME_MS value returned in the PROCESSLIST output. Bug fixed #1379582.

With *Support for Fake Changes* enabled, a write to an *InnoDB* table that would cause B-tree reorganization could lead to server assertion with `unknown error code 1000`. Bug fixed #1410410.

Running `ALTER TABLE ... DISCARD TABLESPACE` with *Support for Fake Changes* enabled would lead to a server assertion. Bug fixed #1372219.

`ALTER TABLE` did not allow to change a column to `NOT NULL` if the column was referenced in a foreign key. Bug fixed #1470677 (upstream #77591).

`DROP TABLE IF EXISTS` which fails due to a foreign key presence could break replication if slave had replication filters. Bug fixed #1475107 (upstream #77684).

Enabling *Log Archiving for XtraDB* when `--innodb-read-only` option was enabled would cause server to crash. Bug fixed #1484432.

LRU manager thread flushing was being accounted to `buffer_flush_background` *InnoDB* metrics which was wrong and redundant. Bug fixed #1491420.

Fixed a typo in the cleaner thread loop where `n_flushed` is added to, instead of reset, by the idle server flushing. This may cause a cleaner thread sleep skip on a non-idle server. Bug fixed #1491435.

Running *TokuDB* for a long time with lots of file open and close operations could lead to a server crash due to server incorrectly setting a reserved value. Bug fixed #690.

Fixed *TokuDB* memory leak due to data race in context status initialization. Bug fixed #697.

Removed unnecessary calls to `malloc_usable_size()` function in PerconaFT library to improve the performance. Bug fixed #682.

Other bugs fixed: #1370002, #1464468, #1287299, #1472256, #867, #870, #878, #658, #661, #663, #665, #687, #696, #698, #870, #685, and #878.

85.34 *Percona Server for MySQL* 5.6.25-73.1

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.25-73.1 on July 9th, 2015 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.25, including all the bug fixes in it, *Percona Server for MySQL* 5.6.25-73.1 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.25-73.1 milestone at Launchpad.

85.34.1 New Features

TokuDB storage engine package has been updated to version 7.5.8

New TokuDB Features

- Exposed ft-index fanout as TokuDB option tokudb_fanout, (default=16 range=2-16384).
- Tokuftdump can now provide a summary info with new –summary option.
- Fanout has been serialized in the ft-header and ft_header.fanout value has been exposed in tokuftdump.
- New checkpoint status variables have been implemented:
  - CP_END_TIME - checkpoint end time, time spend in checkpoint end operation in seconds,
  - CP_LONG_END_COUNT - long checkpoint end count, count of end_checkpoint operations that exceeded 1 minute,
  - CP_LONG_END_TIME - long checkpoint end time, total time of long checkpoints in seconds.
- "Engine" status variables are now visible as "GLOBAL" status variables.

TokuDB Bugs Fixed

- Fixed assertion with big transaction in `toku_txn_complete_txn`.
- Fixed assertion that was caused when a transaction had rollback log nodes orphaned in the blocktable.
- Fixed `ftcxx` tests failures that were happening when it was run in parallel.
- Fixed multiple test failures for Debian/Ubuntu caused by assertion on `setlocale()`.
- Status has been refactored to its own file/subsystem within ft-index code to make the it more accessible.

85.35 *Percona Server for MySQL* 5.6.25-73.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.25-73.0 on June 29th, 2015 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.25, including all the bug fixes in it, *Percona Server for MySQL* 5.6.25-73.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.25-73.0 milestone at Launchpad.

85.35.1 New Features

Percona Server for MySQL has implemented *Support for PROXY protocol*. The implementation is based on a patch developed by Thierry Fournier.

85.35.2 Bugs Fixed

Symlinks to `libmysqlclient` libraries were missing on *CentOS* 6. Bug fixed #1408500.

RHEL/CentOS 6.6 `openssl` package (`1.0.1e-30.el6_6.9`), containing a fix for CVE-2015-4000, changed the DH key sizes to a minimum of 768 bits. This caused an issue for *MySQL* as it uses 512 bit keys. Fixed by backporting an upstream 5.7 fix that increases the key size to 2048 bits. Bug fixed #1462856 (upstream #77275).

Some compressed *InnoDB* data pages could be mistakenly considered corrupted, crashing the server. Bug fixed #1467760 (upstream #73689) *Justin Tolmer*.

innochecksum would fail to check tablespaces in compressed format. The fix for this bug has been ported from *Facebook MySQL* 5.6 patch. Bug fixed #1100652 (upstream #66779).

Using concurrent `REPLACE`, `LOAD DATA REPLACE` or `INSERT ON DUPLICATE KEY UPDATE` statements in the `READ COMMITTED` isolation level or with the **:variable:'innodb_locks_unsafe_for_binlog'** option enabled could lead to a unique-key constraint violation. Bug fixed #1308016 (upstream #76927).

Issuing `SHOW BINLOG EVENTS` with an invalid starting binlog position would cause a potentially misleading message in the server error log. Bug fixed #1409652 (upstream #75480).

While using **:variable:'max_slowlog_size'**, the slow query log was rotated every time **:variable:'slow_query_log'** was enabled, not really checking if the current slow log is indeed bigger than **:variable:'max_slowlog_size'** or not. Bug fixed #1416582.

Fixed possible server assertions when *Backup Locks* are used. Bug fixed #1432494.

If **:variable:'query_response_time_range_base'** was set as a command line option or in a configuration file, its value would not take effect until the first flush was made. Bug fixed #1453277 (*Preston Bennes*).

`mysqld_safe` script is now searching for `libjemalloc.so.1` library, needed by TokuDB, in the `basedir` directory as well. Bug fixed #1462338.

Prepared XA transactions could cause a debug assertion failure during the shutdown. Bug fixed #1468326.

Variable **:variable:'log_slow_sp_statements'** now supports skipping the logging of stored procedures into the slow log entirely with new `OFF_NO_CALLS` option. Bug fixed #1432846.

TokuDB HotBackup library is now automatically loaded with `mysqld_safe` script. Bug fixed #1467443.

Other bugs fixed: #1457113, #1380895, and #1413836.

85.36 *Percona Server for MySQL* 5.6.24-72.2

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.24-72.2 on May 8th, 2015 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.24, including all the bug fixes in it, *Percona Server for MySQL* 5.6.24-72.2 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.24-72.2 milestone at Launchpad.

85.36.1 New Features

TokuDB storage engine package has been updated to version 7.5.7

85.36.2 Bugs Fixed

A server binary as distributed in binary tarballs could fail to load on different systems due to an unsatisfied `libssl.so.6` dynamic library dependency. This was fixed by replacing the single binary tarball with multiple tarballs depending on the *OpenSSL* library available in the distribution: 1) ssl100 - for all *Debian/Ubuntu* versions except *Squeeze/Lucid* (`libssl.so.1.0.0 => /usr/lib/x86_64-linux-gnu/libssl.so.1.0.0 (0x00007f2e389a5000)`); 2) ssl098 - only for *Debian Squeeze* and *Ubuntu Lucid* (`libssl.so.0.9.8 => /usr/lib/libssl.so.0.9.8 (0x00007f9b30db6000)`); 3) ssl101 - for *CentOS 6* and *CentOS 7* (`libssl.so.10 => /usr/lib64/libssl.so.10 (0x00007facbe8c4000)`); 4) ssl098e - to be used only for *CentOS 5* (`libssl.so.6 => /lib64/libssl.so.6 (0x00002aed5b64d000)`). Bug fixed #1172916.

Executing a stored procedure containing a subquery would leak memory. Bug fixed #1380985 (upstream #76349).

A slave server restart could cause a `1755` slave SQL thread error if multi-threaded slave was enabled. This is a regression introduced by fix for bug #1331586 in **:rn:'5.6.21-70.0'**. Bug fixed #1380985.

A string literal containing an invalid UTF-8 sequence could be treated as falsely equal to a UTF-8 column value with no invalid sequences. This could cause invalid query results. Bug fixed #1247218 by a fix ported from *MariaDB* (MDEV-7649).

Percona Server for MySQL `.deb` binaries were built without fast mutexes. Bug fixed #1433980.

Installing or uninstalling the *Audit Log Plugin* would crash the server if the **:variable:'audit_log_file'** variable was pointing to an inaccessible path. Bug fixed #1435606.

The **:variable:'audit_log_file'** would point to random memory area if the *Audit Log Plugin* was not loaded into server, and then installed with `INSTALL PLUGIN`, and `my.cnf` contained **:variable:'audit_log_file'** setting. Bug fixed #1437505.

A specific trigger execution on the master server could cause a slave assertion error under row-based replication. The trigger would satisfy the following conditions: 1) it sets a savepoint; 2) it declares a condition handler which releases this savepoint; 3) the trigger execution passes through the condition handler. Bug fixed #1438990 (upstream #76727).

Percona Server for MySQL client packages were built with with `EditLine` instead of `Readline`. This was causing history file produced by the client no longer easy to read. Further, a client built with `EditLine` could display incorrectly on *PuTTY* SSH client after its window resize. Bugs fixed #1266386, #1296192 and #1332822 (upstream #63130, upstream #72108 and #69991).

Unlocking a table while holding the backup binlog lock would cause an implicit erroneous backup lock release, and a subsequent server crash or hang at the later explicit backup lock release request. Bug fixed #1371827.

Initializing slave threads or executing `CHANGE MASTER TO` statement would crash a debug build if autocommit was disabled and at least one of slave info tables were configured as tables. Bug fixed #1393682.

Other bugs fixed: #1372263 (upstream #72080), #1436138 (upstream #76505), #1182949 (upstream #69453), #1111203 (upstream #68291), and #1384566 (upstream #74615).

85.37 *Percona Server for MySQL* 5.6.23-72.1

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.23-72.1 on March 4th, 2015 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.23, including all the bug fixes in it, *Percona Server for MySQL* 5.6.23-72.1 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.23-72.1 milestone at Launchpad.

85.37.1 New Features

TokuDB storage engine package has been updated to version 7.5.6.

85.37.2 Bugs Fixed

`RPM` pre-install script assumed that the `PID` file was always located in the **:variable:'datadir'**. If it was not, during installation, wrong assumption could be made if the server was running or not. Bug fixed #1201896.

`SHOW GRANTS` displayed only the privileges granted explicitly to the named account. Other effectively available privileges were not displayed. Fixed by implementing *Extended SHOW GRANTS* feature. Bug fixed #1354988 (upstream #53645).

InnoDB lock monitor output was printed even if it was not requested. Bug fixed #1418996.

The stored procedure key was made consistent with other keys in the *Slow Query Log* by replacing space with an underscore. Bug fixed #1419230.

Some `--big-test` MTR tests were failing for *Percona Server for MySQL* because they weren't updated. Bug fixed #1419827.

Other bugs fixed: #1408232, #1385036, #1386363, and #1420303.

85.38 *Percona Server for MySQL* 5.6.22-72.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.22-72.0 on February 6th, 2015 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.22, including all the bug fixes in it, *Percona Server for MySQL* 5.6.22-72.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.22-72.0 milestone at Launchpad.

85.38.1 New Features

Percona Server for MySQL is now able to log the query's response times into *separate* `READ` and `WRITE` `INFORMATION_SCHEMA` tables. Two new `INFORMATION_SCHEMA` tables **:table:'QUERY_RESPONSE_TIME_READ'** and **:table:'QUERY_RESPONSE_TIME_WRITE'** have been implemented for `READ` and `WRITE` queries correspondingly.

New `ps_tokudb_admin` script has been implemented to make the TokuDB Storage engine installation *easier*.

Percona Server for MySQL now supports *Online GTID deployment*. This enables GTID to be deployed on existing replication setups without making the master `read-only` and stopping all the slaves. This feature was ported from the Facebook branch.

85.38.2 Bugs Fixed

`SET STATEMENT ... FOR <statement>` would crash the server if it could not execute the `<statement>` due to: 1) if the `<statement>` was Read-Write in a Read-Only transaction (bug #1387951), 2) if the `<statement>` needed to re-open an already open temporary table and would fail to do so (bug #1412423), 3) if the `<statement>` needed to commit implicitly the ongoing transaction and the implicit commit would fail (bug #1418049).

TokuDB storage engine would fail to load after the upgrade on *CentOS* 5 and 6. Bug fixed #1413956.

Fixed a potential low-probability crash in *XtraDB* linear read-ahead code. Bug fixed #1417953.

Setting the **:variable:'max_statement_time'** per query had no effect. Bug fixed #1376934.

Other bugs fixed: #1407941, #1415843 (upstream #75642).

85.39 *Percona Server for MySQL* 5.6.22-71.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.22-71.0 on January 12th, 2015 (Downloads are available here and from the *Percona Software Repositories*).

Based on MySQL 5.6.22, including all the bug fixes in it, *Percona Server for MySQL* 5.6.22-71.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.22-71.0 milestone at Launchpad.

85.39.1 New Features

Percona Server for MySQL has implemented improved slow log reporting for queries in *stored procedures*.

TokuDB storage engine package has been updated to version 7.5.4. *Percona Server for MySQL* with an older version of TokuDB could hit an early scaling limit when the binary log was enabled. TokuDB

7.5.4 fixes this problem by using the hints supplied by the binary log group commit algorithm to avoid fsync'ing its recovery log during the commit phase of the 2 phase commit algorithm that *MySQL* uses for transactions when the binary log is enabled.

85.39.2 Bugs Fixed

Debian and *Ubuntu* init scripts no longer have a hardcoded server startup timeout. This has been done to accommodate situations where server startup takes a very long time, for example, due to a crash recovery or buffer pool dump restore. Bugs fixed #1072538 and #1328262.

A read-write workload on compressed *InnoDB* tables might have caused an assertion error. Bug fixed #1268656.

Selecting from **:table:'GLOBAL_TEMPORARY_TABLES'** table while running an online `ALTER TABLE` in parallel could lead to server crash. Bug fixed #1294190.

A wrong stack size calculation could lead to a server crash when Performance Schema tables were storing big amount of data or in case of server being under highly concurrent load. Bug fixed #1351148 (upstream #73979).

A query on an empty table with a `BLOB` column may crash the server. Bug fixed #1384568 (upstream #74644).

A read-write workload on compressed *InnoDB* tables might have caused an assertion error. Bug fixed #1395543.

If *HandlerSocket* was enabled, the server would hang during shutdown. Bug fixed #1397859.

The default *MySQL* configuration file, `my.cnf`, was not installed during a new installation on *CentOS*. Bug fixed #1405667.

The query optimizer did not pick a covering index for some `ORDER BY` queries. Bug fixed #1394967 (upstream #57430).

`SHOW ENGINE INNODB STATUS` was displaying two identical `TRANSACTIONS` sections. Bug fixed #1404565.

A race condition in *Multiple user level locks per connection* implementation could cause a deadlock. Bug fixed #1405076.

Other bugs fixed: #1394357, #1337251, #1399174, #1396330 (upstream #74987), #1401776 (upstream #75189).

85.40 *Percona Server for MySQL* 5.6.21-70.1

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.21-70.1 on November 24th, 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 for MySQL* 5.6.21-70.1 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.21-70.1 milestone at Launchpad.

85.40.1 Bugs Fixed

A slave replicating in RBR mode would crash, if a table definition between master and slave differs with an allowed conversion, and the binary log contains a table map event followed by two row log events. This bug is an upstream regression introduced by a fix for bug #72610. Bug fixed #1380010.

An incorrect source code function attribute would cause MySQL to crash on an InnoDB row write, if compiled with a recent GCC with certain compilation options. Bug fixed #1390695 (upstream #74842).

MTR tests for *Response Time Distribution* were not packaged in binary packages. Bug fixed #1387170.

The RPM packages provided for *CentOS* 5 were built using a debugging information format which is not supported in the `gdb` version included with *CentOS* 5.10. Bug fixed #1388972.

A session on a server in mixed mode binlogging would switch to row-based binlogging whenever a temporary table was created and then queried. This switch would last until the session end or until all temporary tables in the session were dropped. This was unnecessarily restrictive and has been fixed so that only the statements involving temporary tables were logged in the row-based format whereas the rest of the statements would continue to use the statement-based logging. Bug fixed #1313901 (upstream #72475).

Other bugs fixed: #1387227, and #1388001.

85.41 *Percona Server for MySQL* 5.6.21-70.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.21-70.0 on October 30th, 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 for MySQL* 5.6.21-70.0 is the current GA release in the *Percona Server for MySQL* 5.6 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.21-70.0 milestone at Launchpad.

85.41.1 New Features

Percona Server for MySQL has ported new *super-read-only option* from *WebScaleSQL*.

Percona Server for MySQL has implemented *CSV engine mode for standard-compliant quote and comma parsing*. This feature also fixes the bug #1316042 (upstream #71091).

Percona Server for MySQL now supports Read Free Replication for TokuDB storage engine.

TokuDB storage engine package has been updated to version 7.5.2.

85.41.2 Bugs Fixed

Values of `IP` and `DB` fields in the *Audit Log Plugin* were incorrect. Bug fixed #1379023.

Specifying the `--malloc-lib` during the server start would produce two `LD_PRELOAD` entries, if a system-wide jemalloc library was installed. Bug fixed #1382069.

In multi-threaded slave replication setup, an incomplete log event group (the one which doesn't end with `COMMIT`/`ROLLBACK`/`XID`) in a relay log could have caused a replication stall. An incomplete log event group might occur as a result of one of the following events: 1) slave crash; 2) `STOP SLAVE` or `FLUSH LOGS` command issued at a specific moment; 3) server shutdown at a specific moment. Bug fixed #1331586 (upstream #73066).

Purging bitmaps exactly up to the last tracked LSN would abort *XtraDB changed page tracking*. Bug fixed #1382336.

`mysql_install_db` script would silently ignore any mysqld startup failures. Bug fixed #1382782 (upstream #74440)

Other bugs fixed: #1369950, #1335590, #1067103, and #1282599.

85.42 *Percona Server for MySQL* 5.6.21-69.0

Percona is glad to announce the release of *Percona Server for MySQL* 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 for MySQL* 5.6.21-69.0 is the current GA release in the *Percona Server for MySQL* 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.

85.42.1 New Features

Response Time Distribution feature has been from *Percona Server for MySQL* 5.5 as a plugin.

TokuDB storage engine package has been updated to version 7.5.1.

85.42.2 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.

85.43 *Percona Server for MySQL* 5.6.20-68.0

Percona is glad to announce the release of *Percona Server for MySQL* 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 for MySQL* 5.6.20-68.0 is the current GA release in the *Percona Server for MySQL* 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.

85.43.1 New Features

Percona Server for MySQL 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 for MySQL* 5.7. There were no functional changes for the feature.

Percona Server for MySQL *Audit Log Plugin* now supports `JSON` and `CSV` formats. The format choice is controlled by **:variable:'audit_log_format'** variable.

Percona Server for MySQL *Audit Log Plugin* now supports *streaming the audit log to syslog*.

TokuDB storage engine package has been updated to version 7.1.8.

85.43.2 Bugs Fixed

Querying **:table:'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 *Statement Timeout* patch in *Percona Server for MySQL* **:rn:'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 **:table:'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 for MySQL* **:rn:'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 **:table:'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).

85.44 *Percona Server for MySQL* 5.6.19-67.0

Percona is glad to announce the release of *Percona Server for MySQL* 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 for MySQL* 5.6.19-67.0 is the current GA release in the *Percona Server for MySQL* 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 for MySQL* 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.

85.44.1 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 for MySQL* 5.6.19-67.0, is now considered GA quality.

Percona Server for MySQL now supports the MTR `--valgrind` option for a server that is either statically or dynamically linked with `jemalloc`.

85.44.2 Bugs Fixed

The `libperconaserverclient18.1` package was missing the library files. Bug fixed #1329911.

Percona Server for MySQL introduced a regression in **:rn:'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.

85.45 *Percona Server for MySQL* 5.6.17-66.0

Percona is glad to announce the release of *Percona Server for MySQL* 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 for MySQL* 5.6.17-66.0 is the current GA release in the *Percona Server for MySQL* 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.

85.45.1 New Features

MySQL benchmarks (`sql-bench` directory in the *MySQL* distribution) has been made compatible with the TokuDB storage engine.

Percona Server for MySQL has ported *MariaDB* code enhancement for *start transaction with consistent snapshot*. This enhancement makes binary log positions consistent with *InnoDB* transaction snapshots.

Percona Server for MySQL 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 for MySQL* 5.6.17-66.0. This feature is currently considered Release Candidate quality.

Percona Server for MySQL version 5.6 now includes *HandlerSocket* in addition to *Percona Server for MySQL* 5.5.

85.45.2 Bugs Fixed

Fix for #1225189 introduced a regression in *Percona Server for MySQL* **:rn:'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 for MySQL* 5.6 command line client was linking to *Percona Server for MySQL* 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 for MySQL version was reported incorrectly in *Debian/Ubuntu* packages. Bug fixed #1319670.

Other bugs fixed: #1318537 (upstream #72615), #1318874, #1285618, #1272732, #1314568, #1271178, and #1323014.

85.46 *Percona Server for MySQL* 5.6.17-65.0

Percona is glad to announce the release of *Percona Server for MySQL* 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 for MySQL* 5.6.17-65.0 is the current GA release in the *Percona Server for MySQL* 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.

85.46.1 New Features

Percona Server for MySQL now supports *Metrics for scalability measurement*.

Percona Server for MySQL now supports *Audit Log Plugin*.

Percona Server for MySQL parser and query optimizer now support *Clustering Secondary Indexes* 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 for MySQL packages are now available for *Ubuntu* 14.04.

85.46.2 Bugs Fixed

Percona Server for MySQL 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 for MySQL debug packages were not built for the previous releases. Bug fixed #1298352.

Queries that no longer exceed **:variable:'long_query_time'** were written to the slow query log if they matched the previous **:variable:'long_query_time'** value when **:variable:'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 **:variable:'flush_caches'** and **:variable:'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 **:variable:'log_slow_rate_type'** set to `query` with **:variable:'log_slow_rate_limit'** feature would log every nth 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 nth query. Bug fixed #1287650.

Percona Server for MySQL source files were referencing *Maatkit* instead of *Percona Toolkit*. Bug fixed #1174779.

Maximum allowed value for **:variable:'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.

85.47 *Percona Server for MySQL* 5.6.16-64.2-tokudb with TokuDB engine

Percona is glad to announce the first **BETA** release of *Percona Server for MySQL* 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 for MySQL* **:rn:'5.6.16-64.2'** including all the features and bug fixes in it, and on TokuDB 7.1.5-rc.4, *Percona Server for MySQL* 5.6.16-64.2-tokudb is the first **BETA** release in the *Percona Server for MySQL* 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.

85.47.1 New Features

Installer will now automatically install TokuDB engine on new installations.

85.48 *Percona Server for MySQL* 5.6.16-64.2

Percona is glad to announce the release of *Percona Server for MySQL* 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 for MySQL* 5.6.16-64.2 is the current GA release in the *Percona Server for MySQL* 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.

85.48.1 Bugs Fixed

The upgrade to *Percona Server for MySQL* **:rn:'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 for MySQL 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 for MySQL* **:rn:'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 for MySQL* **:rn:'5.6.16-64.0'**. Bug fixed #1296416.

Percona Server for MySQL 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` .

85.49 *Percona Server for MySQL* 5.6.16-64.1

Percona is glad to announce the release of *Percona Server for MySQL* 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 for MySQL* 5.6.16-64.1 is the current GA release in the *Percona Server for MySQL* 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.

85.49.1 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 for MySQL* **:rn:'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).

85.49.2 Renaming the `libmysqlclient` to `libperconaserverclient`

This release fixes some of the issues caused by the `libmysqlclient` rename to `libperconaserverclient` in *Percona Server for MySQL* **:rn:'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`.

85.50 *Percona Server for MySQL* 5.6.16-64.0

Percona is glad to announce the release of *Percona Server for MySQL* 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 for MySQL* 5.6.16-64.0 is the current GA release in the *Percona Server for MySQL* 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.

85.50.1 New Features

Percona Server for MySQL 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 for MySQL 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 for MySQL now has server-side support for *Compression Details* option syntax.

Percona Server for MySQL 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 for MySQL* 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.

85.50.2 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 for MySQL 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 for MySQL 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 for MySQL 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 for MySQL 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).

85.51 *Percona Server for MySQL* 5.6.16-64.0-tokudb with TokuDB engine

Percona is glad to announce the first **ALPHA** release of *Percona Server for MySQL* 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 for MySQL* **:rn:'5.6.16-64.0'** including all the features and bug fixes in it, and on TokuDB 7.1.5-rc.3, *Percona Server for MySQL* 5.6.16-64.0-tokudb is the first **ALPHA** release in the *Percona Server for MySQL* 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.

85.51.1 New Features

Percona Server for MySQL 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:

- *Compression Details*

Currently available ALPHA features[1]:

[1] These features are available only in ALPHA Percona Server with TokuDB builds. They might change or even disappear in a future release.

- Multiple Clustering Indexes
- Fast Updates with NOAR
- Hot Table Optimization
- TokuDB AUTOINCREMENT implementation
- Prelocking index and range scans

85.52 *Percona Server for MySQL* 5.6.15-63.0

Percona is glad to announce the release of *Percona Server for MySQL* 5.6.15-63.0 on December 19th, 2013 (Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.6.15, including all the bug fixes in it, *Percona Server for MySQL* 5.6.15-63.0 is the current GA release in the *Percona Server for MySQL* 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.

85.52.1 New Features

Thread Pool high priority scheduling is now enabled by default by changing the default **:variable:'thread_pool_high_prio_tickets'** value from `0` to `4294967295`.

Percona Server for MySQL 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 **:variable:'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 for MySQL has *extended* **`mysqlbinlog`** to provide `SSL` and compression support.

Percona Server for MySQL has reduced the performance overhead of the *User Statistics* feature.

85.52.2 Bugs Fixed

`INSTALL PLUGIN` statement would crash server if *User Statistics* were enabled. Bug fixed #1011047.

:variable:'innodb_log_checksum_algorithm' did not have any effect when set from `my.cnf` or mysqld 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 for MySQL would crash when data was select from **:table:'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 **:variable:‘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 **:variable:‘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.

85.53 *Percona Server for MySQL* 5.6.14-62.0

Percona is glad to announce the release of *Percona Server for MySQL* 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 for MySQL* 5.6.14-62.0 is the current GA release in the *Percona Server for MySQL* 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.

85.53.1 New Features

Percona Server for MySQL has implemented more efficient log block checksums with new **:variable:‘innodb_log_checksum_algorithm‘** variable.

Percona Server for MySQL has implemented support for *Per-query variable statement*.

85.53.2 Bugs Fixed

Percona Server for MySQL 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 (**:variable:‘innodb_io_capacity‘** vs **:variable:‘innodb-io-capacity‘**). Bug fixed #1233294.

Building *Percona Server for MySQL* 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 for MySQL* shutdown. Bug fixed #1233354.

Storage engine enforcement (**:variable:'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 for MySQL* 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.

85.54 *Percona Server for MySQL* 5.6.13-61.0

Percona is glad to announce the first GA (Generally Available) release of *Percona Server for MySQL* 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 for MySQL* 5.6.13-61.0 is the first GA release in the *Percona Server for MySQL* 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.

85.54.1 New Features

Percona Server for MySQL 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 for MySQL 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 **:variable:'innodb_flush_method'** has been ported from *Percona Server for MySQL* 5.5.

Statement Timeout feature has been ported from the Twitter branch.

Percona Server for MySQL has *extended* the `SELECT INTO ... OUTFILE` and `SELECT INTO DUMPFILE` to add the support for UNIX sockets and named pipes.

85.54.2 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 for MySQL* would fail in post-installation step if server failed to start. Bug fixed #1002500.

Percona Server for MySQL 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 for MySQL **:rn:'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 **:table:'INNODB_CHANGED_PAGES'** query and bitmap file delete by `PURGE CHANGED_PAGE_BITMAP` or directly on the file system. Bug fixed #1191580.

Percona Server for MySQL could not be built with *Thread Pool* feature and `-DWITH_PERFSCHEMA_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.

85.55 *Percona Server for MySQL* 5.6.13-60.6

Percona is glad to announce the fourth Release Candidate release of *Percona Server for MySQL* 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 for MySQL* 5.6.13-60.6 is the fourth RC release in the *Percona Server for MySQL* 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.

85.55.1 New Features

Improved Buffer Pool Scalability feature has been ported from *Percona Server for MySQL* 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 for MySQL* 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 **:variable:'utility_user_privileges'** that allows a comma separated value list of extra access privileges that can be granted to the utility user.

Percona Server for MySQL now provides additional information in the slow query log when **:variable:'log_slow_rate_limit'** variable is enabled.

A new variable **: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, **:variable:'log_slow_rate_limit'** will not apply to it.

85.55.2 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 **:table:'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.

85.56 *Percona Server for MySQL* 5.6.13-60.5

Percona is glad to announce the third Release Candidate release of *Percona Server for MySQL* 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 for MySQL* 5.6.13-60.5 is the third RC release in the *Percona Server for MySQL* 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.

85.56.1 Ported Features

Fixed Size for the Read Ahead Area has been ported from *Percona Server for MySQL* 5.5

85.56.2 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 **:table:'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 for MySQL* 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 for MySQL* with `DTrace version Sun D 1.11` provided by recent `SmartOS` versions. Bug fixed #1196460.

Fixed a regression introduced in *Percona Server for MySQL* **:rn:'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.

85.57 *Percona Server for MySQL* 5.6.12-60.4

Percona is glad to announce the second Release Candidate release of *Percona Server for MySQL* 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 for MySQL* 5.6.12-60.4 is the second RC release in the *Percona Server for MySQL* 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.

85.57.1 New Features

Percona Server for MySQL has implemented support for supplementary groups for *PAM Authentication Plugin*.

85.57.2 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 for MySQL* on non-Linux platforms. Bug fixed #1189429.

Other bugs fixed: bug fixed #1188516, bug fixed #1130731 and bug fixed #1133266.

85.58 *Percona Server for MySQL* 5.6.11-60.3

Percona is glad to announce the first Release Candidate release of *Percona Server for MySQL* 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 for MySQL* 5.6.11-60.3 is the first RC release in the *Percona Server for MySQL* 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 for MySQL* 5.5 release (currently *Percona Server for MySQL* 5.5.31-30.3).

85.58.1 New Features

Percona Server for MySQL has implemented *Log Archiving for XtraDB*. Currently this feature implementation is considered *ALPHA*.

85.58.2 Ported Features

Percona Server for MySQL has ported priority connection scheduling for the *Thread Pool* from *Percona Server for MySQL* 5.5.

Percona Server for MySQL 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 for MySQL has ported **:variable:‘innodb_read_views_memory‘** and **:variable:‘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 for MySQL* 5.5

Improved NUMA support has been ported from *Percona Server for MySQL* 5.5

Suppress Warning Messages has been ported from *Percona Server for MySQL* 5.5

Improved MEMORY Storage Engine has been ported from *Percona Server for MySQL* 5.5

Restricting the number of binlog files has been ported from *Percona Server for MySQL* 5.5

Too Many Connections Warning has been ported from *Percona Server for MySQL* 5.5

error_pad has been ported from *Percona Server for MySQL* 5.5

Lock-Free SHOW SLAVE STATUS has been ported from *Percona Server for MySQL* 5.5

Percona Toolkit UDFs has been ported from *Percona Server for MySQL* 5.5

Support for Fake Changes has been ported from *Percona Server for MySQL* 5.5

Kill Idle Transactions has been ported from *Percona Server for MySQL* 5.5

Enforcing Storage Engine has been ported from *Percona Server for MySQL* 5.5

Utility user has been ported from *Percona Server for MySQL* 5.5

Extending the secure-file-priv server option has been ported from *Percona Server for MySQL* 5.5

Expanded Program Option Modifiers has been ported from *Percona Server for MySQL* 5.5

XtraDB changed page tracking has been ported from *Percona Server for MySQL* 5.5

PAM Authentication Plugin has been ported from *Percona Server for MySQL* 5.5

User Statistics has been ported from *Percona Server for MySQL* 5.5

Slow Query Log has been ported from *Percona Server for MySQL* 5.5

Extended Show Engine InnoDB Status has been ported from *Percona Server for MySQL* 5.5

Count InnoDB Deadlocks has been ported from *Percona Server for MySQL* 5.5

Log All Client Commands (syslog) has been ported from *Percona Server for MySQL* 5.5

Show Storage Engines has been ported from *Percona Server for MySQL* 5.5

Thread Based Profiling has been ported from *Percona Server for MySQL* 5.5

85.58.3 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 **:variable:'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` (**:table:'CLIENT_STATISTICS'**, **:table:'INDEX_STATISTICS'**, **:table:'TABLE_STATISTICS'**, **:table:'THREAD_STATISTICS'**, and **:table:'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 **:table:'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 **:table:'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 **:table:'INNODB_CHANGED_PAGES'** query started returning data to indicate an incomplete result set. Bug fixed #1185040.

The **:table:'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 for MySQL*, 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.

85.59 *Percona Server for MySQL* 5.6.10-60.2

Percona is glad to announce the Alpha release of *Percona Server for MySQL* 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 for MySQL* 5.6.10-60.2 is the third ALPHA release in the *Percona Server for MySQL* 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.

85.59.1 New Features

Expanded Fast Index Creation has been ported from *Percona Server for MySQL* 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 for MySQL* 5.5

85.59.2 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 for MySQL* tarballs. Bug fixed #1153950 (*Ignacio Nin*).

85.60 *Percona Server for MySQL* 5.6.6-60.1

Percona is glad to announce the ALPHA release of *Percona Server for MySQL* 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 for MySQL* 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.

85.60.1 Features removed

- **:variable:'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 **:rn:'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 **:table:'TEMPORARY_TABLES'** and **:table:'GLOBAL_TEMPORARY_TABLES'** in the `INFORMATION_SCHEMA` database.

85.61 *Percona Server for MySQL* 5.6.5-60.0

Percona is glad to announce the ALPHA release of *Percona Server for MySQL* 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 for MySQL* 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.

85.61.1 Features Included

- Fixed upstream *MySQL* bug #49336, mysqlbinlog couldn't handle `stdin` when "|" was used. Bug fixed: #933969 (*Sergei Glushchenko*).
- Fixed upstream *MySQL* bug #65946, which could cause the server to segfault if if using `libeatmydata` (and possibly other LD_PRELOADs). Bug fixed: #933969 (*Stewart Smith*).
- Added the `TIME_MS` column in the **:table:'PROCESSLIST'** table
- *Ignoring missing tables in mysqldump* feature has been ported from *Percona Server for MySQL* 5.5
- **:variable:'fast_index_creation'** feature has been ported from *Percona Server for MySQL* 5.5
- *Temporary tables* information have been added to the `INFORMATION_SCHEMA` database

85.61.2 Features removed

- The **:variable:'optimizer_fix'** variable from *Percona Server for MySQL* 5.5 will not be present in 5.6

CHAPTER

EIGHTYSIX

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

INDEX