Percona Server Documentation

5.7.44-48 (2023-12-05)

Percona Technical Documentation Team

Table of contents

Download PDF

1. Percona Server for MySQL 5.7 Documentation

This documentation is for the latest release: Percona Server for MySQL 5.7.44-48 (Release Notes).

Percona Server for MySQL is a freely available, fully compatible, enhanced, and open source drop-in replacement for any MySQL database. It provides superior and optimized performance, greater scalability, and availability, enhanced backups, increased visibility and instrumentation.

Percona Server for MySQL is trusted by thousands of enterprises to provide better performance and concurrency for their most demanding workloads.

1.1 For Monitoring and Management

Percona Monitoring and Management (PMM)monitors and provides actionable performance data for MySQL variants, including Percona Server for MySQL, Percona XtraDB Cluster, Oracle MySQL Community Edition, Oracle MySQL Enterprise Edition, and MariaDB. PMM captures metrics and data for the InnoDB, XtraDB, and MyRocks storage engines, and has specialized dashboards for specific engine details.

Install PMM and connect your server instances to it.

Contact Us

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-05-17

2. Introduction

Download PDF

2.1 List of features available in Percona Server for MySQL releases

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

Percona Server for MySQL 5.7	Percona Server for MySQL 8.0
Misc. INFORMATION_SCHEMA Tables	Misc. INFORMATION_SCHEMA Tables
Extended Show Engine InnoDB Status	Extended Show Engine InnoDB Status
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
Page cleaner thread tuning	Page cleaner thread tuning
Statement Timeout	Statement Timeout
Extended SELECT INTO OUTFILE/DUMPFILE	Extended SELECT INTO OUTFILE/DUMPFILE
Per-query variable statement	Per-query variable statement
Extended mysqlbinlog	Extended mysqlbinlog
Slow Query Log Rotation and Expiration	Slow Query Log Rotation and Expiration
Metrics for scalability measurement	Feature not implemented
Audit Log	Audit Log
Backup Locks	Backup Locks
CSV engine mode for a standard-compliant quote and comma parsing	CSV engine mode for a standard-compliant quote and comma parsing
Super read-only	Super read-only

2.1.1 Other Reading

- What Is New in MySQL 5.7
- What Is New in MySQL 8.0

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

2.2 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 ability to tune, 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*.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-10-04

Download PDF

2.3 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 5.7.27	MySQL 5.7.27
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
Group Replication	Yes	Yes
MyRocks Storage Engine	Yes	No
TokuDB Storage Engine	Yes	No

Extra Features for Developers	Percona Server 5.7.27	MySQL 5.7.27
NoSQL Socket-Level Interface	Yes	Yes
X API Support	Yes	Yes
InnoDB Full-Text Search Improvements	Yes	No
Extra Hash/Digest Functions	Yes	No

Instrumentation and Troubleshooting Features	Percona Server 5.7.27	MySQL 5.7.27
INFORMATION_SCHEMA Tables	71	61
Global Performance and Status Counters	432	357
Per-Table Performance Counters	Yes	No
Per-Index Performance Counters	Yes	No
Per-User Performance Counters	Yes	No
Per-Client Performance Counters	Yes	No
Per-Thread Performance Counters	Yes	No
Global Query Response Time Statistics	Yes	No
Enhanced SHOW ENGINE INNODB STATUS	Yes	No
Undo Segment Information	Yes	No
Temporary Tables Information	Yes	No
Extended Slow Query Logging	Yes	No
User Statistics	Yes	No

Performance and Scalability Features	Percona Server 5.7.27	MySQL 5.7.27
Improved scalability by splitting mutexes	Yes	No
Improved MEMORY Storage Engine	Yes	No
Improved Flushing	Yes	No
Parallel Doublewrite Buffer	Yes	No
Configurable Page Sizes	Yes	Yes
Configurable Fast Index Creation	Yes	No
Per-column Compression for VARCHAR/BLOB and JSON	Yes	No
Compressed columns with Dictionaries	Yes	No

Security Features	Percona Server 5.7.27	MySQL 5.7.27
PAM Authentication Plugin	Yes	Enterprise-Only
Audit Logging Plugin	Yes	Enterprise-only

Encryption Features	Percona Server 5.7.27	MySQL 5.7.27
Encrypt InnodDB data	Yes	Yes
Encrypt InnoDB tablespaces	Experimental	No
Encrypt InnoDB logs	Experimental	No
Encrypt Binary logs	Experimental	No
Encrypt temporary files	Experimental	No
Key Rotation	Experimental	No
Scrubbing	Experimental	No
Enforce Encryption	Experimental	No
Storing Keyring in a file	Yes	Yes
Storing Keyring in a Hashicorp Vault	Yes	No

Operational Improvements	Percona Server 5.7.27	MySQL 5.7.27
Changed Page Tracking	Yes	Yes
Threadpool	Yes	Enterprise-only
Backup Locks	Yes	No
Extended SHOW GRANTS	Yes	No
Improved Handling of Corrupted Tables	Yes	No
Ability to kill Idle Transactions	Yes	No
Improvements to START TRANSACTION WITH CONSISTENT SNAPSHOT	Yes	No

Features for Running Database as a Service (DBaaS)	Percona Server 5.7.27	MySQL 5.7.27
Special Utility User	Yes	No
Enforce a Specific Storage Engine	Yes	No
Expanded Program Option Modifiers	Yes	No

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

2.4 Changed in Percona Server 5.7

Percona Server for MySQL 5.7 is based on MySQL 5.7 and incorporates many of the improvements found in Percona Server for MySQL 5.6.

2.4.1 Features removed from Percona Server for MySQL 5.7 that were available in Percona Server for MySQL 5.6

Note

Percona Server 5.7 won't be able to start if any of variables from these features are set in the server's configuration file.

Some features that were present in Percona Server for MySQL 5.6 have been removed in Percona Server for MySQL 5.7. These are:

- Handlersocket
  - This feature might be included in a future release if HandlerSocket starts supporting 5.7.
- Support for Fake Changes
  - Instead of replica prefetching using the fake changes, a 5.7 intra-schema parallel replication replica should be used.
- `SHOW ENGINE INNODB STATUS` no longer prints the count of active Read-Only transactions.
- InnoDB redo log archiving has been removed due to lack of user uptake of the feature.

2.4.2 Changes in Percona Server 5.6 features

- The minor Percona Server for MySQL version number ("y" in "5.a.b-x.y") has been dropped to simplify Percona Server for MySQL versioning.
- Performance Schema memory instrumentation support has been added to the

  Audit Log Plugin, Metrics for scalability measurement, and PAM Authentication Plugin, and to the core server to track memory used by User Statistics, Per-query variable statement, XtraDB changed page tracking, and Thread Pool features.
- Audit Log Plugin now produces diagnostics in a format consistent with the rest of the server.
- The performance_schema.metadata_locks table now displays `backup` and `binlog` lock information too. The `object_type` column has two new valid values: `backup`, and `binlog`.
- INFORMATION_SCHEMA.XTRADB_RSEG table schema has been changed to support new possible InnoDB page sizes. The `zip_size` column has been removed and replaced by new columns `physical_page_size`, `logical_page_size`, and `is_compressed`.
- XTRADB_READ_VIEW table no longer contains the `READ_VIEW_UNDO_NUMBER` column, which was associated with unused code and always contained zero.
- Interaction between `--hidden-` option modifier and session_track_system_variables has been implemented as follows: any variables with `--hidden-` modifier become hidden from the latter variable too. Thus, they should not be present there. Even if you never set `session_track_system_variables`, care must be taken if a variable contained in its default value (i.e. autocommit) is hidden.

- Nested `SET STATEMENT ... FOR SET STATEMENT ... FOR ...` statements have a different effect in the innermost clause when the nested clauses set the same variables: in 5.6, the innermost assignment is effective, but, in 5.7, the outermost assignment is effective.
- Utility user is treated as a `SUPER` user for the purposes of offline mode: utility user connections are not dropped if server switches to offline mode and new utility user connections can be established to such server.
- The server will abort startup with an error message if conflicting enforce_storage_engine and disabled_storage_engines option values are specified, that is, if the enforced storage engine is in the list of disabled storage engines.

2.4.3 Features available in Percona Server for MySQL 5.6 that have been replaced with MySQL 5.7 features

Note

Percona Server 5.7 won't be able to start if any of variables from these features are set in the server's configuration file.

Some Percona Server for MySQL 5.6 features have been replaced by similar or equivalent MySQL 5.7 features, so we now keep the MySQL 5.7 implementations in Percona Server for MySQL 5.7. These are:

- Lock-Free SHOW SLAVE STATUS NONBLOCKING has been replaced by a regular `SHOW SLAVE STATUS` implementation . Oracle implementation forbids calling it from a stored function.
- Behavior corresponding to slow_query_log_timestamp_precision set

  to `microsecond` is now the default, the variable itself and the behavior corresponding to the variable's `second` value is removed.
- Behavior corresponding to slow_query_log_timestamp_always set to `TRUE`

  is now the default, the variable itself and the behavior corresponding to the variable's `FALSE` value is removed.
- Statement timeout feature has been replaced by Oracle Server-side SELECT statement timeouts implementation. Differences: the Oracle variable is named max_statement_time instead of max_statement_time; variable have_statement_timeout variable has been removed; the timeouts only apply for read-only SELECT .
- Atomic write support on fusionIO devices with NVMFS has been replaced by Oracle implementation. It is no longer required to enable innodb_use_atomic_writes variable, and this variable has been removed. The atomic write support will be enabled, and the doublewrite buffer disabled, on supporting devices automatically. The Oracle implementation does not silently adjust innodb_flush_method to `O_DIRECT` if it has a different value. The user must set it to `O_DIRECT` explicitly, or atomic writes will not be enabled.
- Online GTID migration patch has been replaced by an upstream variable gtid_mode made dynamic.
- The Error Code Compatibility has been replaced by the multiple start-error-number directive in `sql/share/errmsg-utf8.txt` support.
- Ignoring missing tables in mysqldump with `--ignore-create-error` option has been replaced by the more general upstream option –ignore-error option.
- innodb_log_block_size has been replaced by innodb_log_write_ahead_size variable. To avoid read on write when the storage block size is not equal to 512 bytes, the latter should be set to the same value the former was. If innodb_log_block_size was set to non-default values, new log files must be created during the upgrade. This can be done by cleanly shutting down the service and removing the variable from `my.cnf` configuration and removing the old logs and starting the service again before doing the upgrade.
- Extended secure-file-priv server option , which was used to disable `LOAD DATA INFILE`, `SELECT INTO OUTFILE` statements, and `LOAD_FILE()` function completely, has been replaced by upstream introducing `NULL` as a possible value to this variable. To migrate, any value-less settings must be replaced by `NULL` .
- innodb_sched_priority_cleaner variable has been removed, as the effect of setting it to 39 (corresponding to nice value of -20), is now enabled by default.
- innodb_adaptive_hash_index_partitions has been replaced by innodb_adaptive_hash_index_parts .
- In the default server setup (with InnoDB being the only one XA-capable storage engine), `--tc-heuristic-recover=COMMIT` is silently converted to `ROLLBACK`. If TokuDB or another XA-supporting 3rd party storage engine is installed, `--tc-heuristic-recover=ROLLBACK` option is unavailable. The default value of `tc-heuristic-recover` option in Percona Server for MySQL 5.6 but not in MySQL 5.6 was `NONE` as a result of fix for upstream bug #70860. Since Oracle fixed the same bug in 5.7, the default value is `OFF` now.
- innodb_log_checksum_algorithm feature has been replaced by innodb_log_checksums option. In particular, to get the effect of setting the innodb_log_checksum_algorithm to `crc32`, innodb_log_checksums should be set to `ON`, which is a default setting for this variable.
- innodb_buffer_pool_populate server option and numa_interleave `mysql_safe.sh` option have been replaced by innodb_numa_interleave server option. Note that flush_caches option still remains.
- Ability to change database for mysqlbinlog implementation has been replaced from MariaDB one with MySQL rewrite-db one. The feature is mostly identical with two differences: 1) multiple rewrite rules must be given as separate options, and the ability to list them in a single rule, separated by commas, is lost. That is, any `--rewrite-db='a->b,c->d'` occurrences must be replaced with `--rewrite-db='a->b' --rewrite-db='c->d'`. 2) Whitespace around database names is not ignored.

- INFORMATION_SCHEMA.PROCESSLIST.TID column has been replaced by PERFORMANCE_SCHEMA.THREADS.THREAD_OS_ID column . If running under thread pool, `THREAD_OS_ID` column will always be `NULL`, whereas in the 5.6 implementation `TID` column showed either `NULL` or the assigned worker thread id at the moment.
- innodb_foreground_preflush server variable has been removed as the upstream implemented a similar feature without a controlling option.
- Log All Client Commands (syslog) feature has been replaced by Oracle mysql Logging implementation.
- Support for Multiple user level locks per connection has been replaced by Oracle implementation, which is based on the same contributed patch by *Kostja Osipov*.
- super-read-only option has been replaced by Oracle super_read_only variable implementation.
- Mutex names in `SHOW ENGINE INNODB MUTEX` have been replaced by Oracle mutex name implementation.
- Percona Server for MySQL now uses packaging similar to the upstream MySQL version. Most important change is that for *Debian*/*Ubuntu* upgrades you now need to run `mysql_upgrade` manually.

2.4.4 List of status variables that are no longer available in Percona Server for MySQL 5.7

Following status variables available in Percona Server for MySQL 5.6 are no longer present in Percona Server for MySQL 5.7:

Status Variables	Replaced by
Com_purge_archived	InnoDB redo log archiving has been removed due to lack of user uptake of the feature.
Com_purge_archived_before_date	InnoDB redo log archiving has been removed due to lack of user uptake of the feature.
read_views_memory	transaction descriptors replaced by the upstream implementation
descriptors_memory	transaction descriptors replaced by the upstream implementation
innodb_mem_total	This variable was always zero in 5.6 with the default innodb_use_sys_malloc setting
innodb_deadlocks	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (lock_deadlocks)
Innodb_ibuf_merges	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges)
Innodb_ibuf_merged_deletes	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_delete)
Innodb_ibuf_merged_delete_marks	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_delete_mark)
Innodb_ibuf_discarded_deletes	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_discard_delete)
Innodb_ibuf_discarded_delete_marks	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_discard_delete_mark)
Innodb_ibuf_discarded_inserts	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_discard_insert)
Innodb_ibuf_merged_inserts	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_merges_insert)
Innodb_ibuf_size	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (ibuf_size)
Innodb_s_lock_os_waits	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_s_os_waits)
Innodb_s_lock_spin_rounds	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_s_spin_rounds)
Innodb_s_lock_spin_waits	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_s_spin_waits)
Innodb_x_lock_os_waits	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_x_os_waits)

Status Variables	Replaced by
Innodb_x_lock_spin_rounds	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_x_spin_rounds)
Innodb_x_lock_spin_waits	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (innodb_rwlock_x_spin_waits)
Innodb_current_row_locks	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (lock_row_lock_current_waits)
Innodb_history_list_length	Information now available in INFORMATION_SCHEMA.INNODB_METRICS table (trx_rseg_history_len)
Innodb_mutex_os_waits	SHOW ENGINE INNODB MUTEX presents the same information, but per-mutex instead of whole system aggregation
Innodb_mutex_spin_rounds	SHOW ENGINE INNODB MUTEX presents the same information, but per-mutex instead of whole system aggregation
Innodb_mutex_spin_waits	SHOW ENGINE INNODB MUTEX presents the same information, but per-mutex instead of whole system aggregation

2.4.5 List of system variables that are no longer available in Percona Server for MySQL 5.7

Following system variables available in Percona Server for MySQL 5.6 are no longer present in Percona Server for MySQL 5.7:

Waring

Percona Server for MySQL 5.7 won't be able to start if some of these variables are set in the server's configuration file.

System Variables	Feature Comment
gtid_deployment_step	Replaced by an upstream variable gtid_mode made dynamic.
innodb_fake_changes	Instead of replica prefetching using the fake changes, a 5.7 intra-schema parallel replication replica should be used.
innodb_locking_fake_changes	Instead of replica prefetching using the fake changes, a 5.7 intra-schema parallel replication replica should be used.
innodb_log_archive	InnoDB redo log archiving has been removed due to lack of user uptake of the feature.
innodb_log_arch_dir	InnoDB redo log archiving has been removed due to lack of user uptake of the feature.
innodb_log_arch_expire_sec	InnoDB redo log archiving has been removed due to lack of user uptake of the feature.
innodb_log_block_size	Replaced by upstream innodb_log_write_ahead_size variable. To avoid read on write when the storage block size is not equal to 512 bytes, the latter should be set to the same value the former was. If innodb_log_block_size was set to non-default values, new log files must be created during the upgrade. This can be done by cleanly shutting down the service and removing the variable from my.cnf configuration and removing the old logs and starting the service again before doing the upgrade.
max_statement_time	Replaced by upstream max_execution_time variable in Server-side SELECT statement timeouts implementation.
have_statement_timeout	Variable has been removed due to upstream feature implementation
innodb_use_atomic_writes	Variable has been removed due to upstream feature implementation
innodb_adaptive_hash_index_partitions	Replaced by upstream variable innodb_adaptive_hash_index_parts

2.4.6 Features ported from Percona Server for MySQL 5.6 to Percona Server for MySQL 5.7

Following features were ported from Percona Server for MySQL 5.6 to Percona Server for MySQL 5.7:

Feature Ported	Version
Improved Buffer Pool Scalability	Percona Server for MySQL 5.7.10-1
Improved InnoDB I/O Scalability	Percona Server for MySQL 5.7.10-1
Query Cache Enhancements	Percona Server for MySQL 5.7.10-1
Improved NUMA support	Percona Server for MySQL 5.7.10-1
Thread Pool	Percona Server for MySQL 5.7.10-1
XtraDB Performance Improvements for I/O-Bound Highly-Concurrent Workloads	Percona Server for MySQL 5.7.10-1
Suppress Warning Messages	Percona Server for MySQL 5.7.10-1
Improved MEMORY Storage Engine	Percona Server for MySQL 5.7.10-1
Restricting the number of binlog files	Percona Server for MySQL 5.7.10-1
Extended SELECT INTO OUTFILE/DUMPFILE	Percona Server for MySQL 5.7.10-1
Per-query variable statement	Percona Server for MySQL 5.7.10-1
Extended mysqlbinlog	Percona Server for MySQL 5.7.10-1
Slow Query Log Rotation and Expiration	Percona Server for MySQL 5.7.10-1
CSV engine mode for standard-compliant quote and comma parsing	Percona Server for MySQL 5.7.10-1
Support for PROXY protocol	Percona Server for MySQL 5.7.10-1
Per-session server-id	Percona Server for MySQL 5.7.10-1
Too Many Connections Warning	Percona Server for MySQL 5.7.10-1
Handle Corrupted Tables	Percona Server for MySQL 5.7.10-1
Percona Toolkit UDFs	Percona Server for MySQL 5.7.10-1
Kill Idle Transactions	Percona Server for MySQL 5.7.10-1
Enforcing Storage Engine	Percona Server for MySQL 5.7.10-1
Utility user	Percona Server for MySQL 5.7.10-1

Feature Ported	Version
Expanded Program Option Modifiers	Percona Server for MySQL 5.7.10-1
XtraDB changed page tracking	Percona Server for MySQL 5.7.10-1
PAM Authentication Plugin	Percona Server for MySQL 5.7.10-1
Expanded Fast Index Creation	Percona Server for MySQL 5.7.10-1
Backup Locks	Percona Server for MySQL 5.7.10-1
Audit Log Plugin	Percona Server for MySQL 5.7.10-1
Start transaction with consistent snapshot	Percona Server for MySQL 5.7.10-1
Extended SHOW GRANTS	Percona Server for MySQL 5.7.10-1
User Statistics	Percona Server for MySQL 5.7.10-1
Slow Query Log	Percona Server for MySQL 5.7.10-1
Extended Show Engine InnoDB Status	Percona Server for MySQL 5.7.10-1
Show Storage Engines	Percona Server for MySQL 5.7.10-1
Process List	Percona Server for MySQL 5.7.10-1
Misc. INFORMATION_SCHEMA Tables	Percona Server for MySQL 5.7.10-1
Thread Based Profiling	Percona Server for MySQL 5.7.10-1
Metrics for scalability measurement	Percona Server for MySQL 5.7.10-1
Response Time Distribution	Percona Server for MySQL 5.7.10-1

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

2.5 Understand version numbers

A version number identifies the product release. The product contains the latest Generally Available (GA) features at the time of that release.

5.7.39	**-42.**	**2**
Base version	Minor build version	Custom build

Percona uses semantic version numbering, which follows the pattern of base version, minor build version, and an optional custom build. Percona assigns unique, non-negative integers in increasing order for each minor build release. The version number combines the base MySQL 5.7 version number, the minor build version, and the custom build version, if needed.

For example, the version numbers for Percona Server for MySQL 5.7.39-42.2 define the following information:

- Base version - the leftmost numbers indicate MySQL 5.7 version used as a base. An increase in base version resets the minor build version to 0.
- Minor build version - an internal number that increases by one every time Percona Server for MySQL is released.
- Custom build version - an optional number assigned to custom builds used for bug fixes. The software features, unless they're included in the bug fix, don't change.

Percona Server for MySQL 5.7.18-14, 5.7.18-15, and 5.7.18-16 are multiple releases based on MySQL 5.7.18.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-05-04

3. Installation

Download PDF

3.1 Installing Percona Server for MySQL 5.7

Before installing, you might want to read the Percona Server for MySQL 5.7 Release notes.

We gather Telemetry data in the Percona packages and Docker images.

3.1.1 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.

- Installing Percona Server for MySQL on *Debian* and *Ubuntu*
- Installing Percona Server for MySQL on Red Hat Enterprise Linux and CentOS

3.1.2 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-5.7.10-3.tar.gz
$ cd percona-server-5.7.10-3
```

Copy the debian packaging into the debian directory:

```
$ 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 "5.7.10-3-1" "Update to 5.7.10-3"
```

Build the Debian source package:

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

Use `sbuild` to build the binary package in a `chroot`:

```
$ sbuild -d sid percona-server-5.7_5.7.10_3-1.dsc
```

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

Note

The PAM Authentication Plugin is not built with the server by default. In order to build the Percona Server with the PAM plugin, add the `-DWITH_PAM=ON` option.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-11-27

Download PDF

3.2 Installing Percona Server for MySQL 5.7 on Debian and Ubuntu

Note

The following instructions install Percona Server for MySQL 5.7. The instructions to install Percona Server for MySQL 8.0 are available at this location.

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.

We gather Telemetry data in the Percona packages and Docker images.

3.2.1 What's in each DEB package?

The `percona-server-server-5.7` package contains the database server itself, the `mysqld` binary and associated files.

The `percona-server-common-5.7` package contains files common to the server and client.

The `percona-server-client-5.7` package contains the command line client.

The `percona-server-5.7-dbg` package contains debug symbols for the server.

The `percona-server-test-5.7` package contains the database test suite.

The `percona-server-source-5.7` package contains the server source.

The `libperconaserverclient20-dev` package contains header files needed to compile software to use the client library.

The `libperconaserverclient20` 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.

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

1. Update package repositories:

   ```
   shell
   $ sudo apt update
   ```

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

   ```
   shell
   $ sudo apt install gnupg2
   ```

3. Fetch the repository packages from Percona web:

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

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

```
shell
$ 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-original-release.list` file.

5. Remember to update the local cache:

```
shell
$ sudo apt update
```

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.

6. After that you can install the server package:

```
shell
$ sudo apt install percona-server-server-5.7
```

Note

Percona Server for MySQL 5.7 comes with the TokuDB storage engine and MyRocks storage engine. These storage engines are installed as plugin.

For information on how to install and configure TokuDB, refer to the TokuDB Installation guide.

For information on how to install and configure MyRocks, refer to the Percona MyRocks Installation Guide guide.

The Percona Server for MySQL distribution contains several useful User Defined Functions (UDF) from Percona Toolkit. After the installation completes, run the following commands to create these functions:

```
$ 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'"
```

For more details on the UDFs, see Percona Toolkit UDFS.

Percona `apt` Testing repository

Percona offers pre-release builds from the testing repository. To enable it, run **percona-release** with the `testing` argument. Run this command as root or by using the **sudo** command.

```
$ sudo percona-release enable original testing
```

Apt-Pinning the packages

In some cases, you might need to pin the selected packages to avoid upgrades from the distribution repositories. Create a new file `/etc/apt/preferences.d/00percona.pref` and add the following lines:

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

3.2.3 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 Percona Server for MySQL 5.7.10-3 release packages for *Debian* 8.0:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.10-3/binary/
debian/jessie/x86_64/Percona-Server-5.7.10-3-r63dafaf-jessie-x86_64-bundle.tar
```

You should then unpack the bundle to get the packages:

```
$ tar xvf Percona-Server-5.7.10-3-r63dafaf-jessie-x86_64-bundle.tar
```

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

```
$ ls *.deb
```

The output could be this:

```
libperconaserverclient20-dev_5.7.10-3-1.jessie_amd64.deb
libperconaserverclient20_5.7.10-3-1.jessie_amd64.deb
percona-server-5.7-dbg_5.7.10-3-1.jessie_amd64.deb
percona-server-client-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-common-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-server-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-source-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-test-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-tokudb-5.7_5.7.10-3-1.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 (`libperconaserverclient20_5.7.10-3-1.jessie_amd64.deb`, `percona-server-client-5.7_5.7.10-3-1.jessie_amd64.deb`, `percona-server-common-5.7_5.7.10-3-1.jessie_amd64.deb`, and `percona-server-server-5.7_5.7.10-3-1.jessie_amd64.deb`. Optionally you can install `percona-server-tokudb-5.7_5.7.10-3-1.jessie_amd64.deb` if you want TokuDB storage engine).

Note

Percona Server for MySQL 5.7 comes with the TokuDB storage engine. You can find more information on how to install and enable the TokuDB storage in the TokuDB Installation guide.

Warning

When installing packages manually like this, you'll need to make sure to resolve all the dependencies and install missing packages yourself. Following packages will need to be installed before you can manually install Percona Server: `mysql-common`, `libjemalloc1`, `libaio1` and `libmecab2`

AppArmor settings

AppArmor is a kernel-integrated system which controls how applications access the file system by creating application profiles. If the installation of MySQL adds an AppArmor profile, you can find the profile in the following locations:

- /etc/apparmor.d/usr.sbin.mysqld
- /etc/apparmor.d/local/usr.sbin.mysqld

The `local` version contains only comments. Add any changes specific for the server to the `local` file.

The `usr.sbin.mysqld` file has the following settings:

```
#include <tunables/global>

/usr/sbin/mysqld {
  ...
  # Allow data dir access
  /var/lib/mysql/ r,
  /var/lib/mysql/** rwk,

  # Allow data files dir access
    /var/lib/mysql-files/ r,
    /var/lib/mysql-files/** rwk,

  # Allow keyring dir access
    /var/lib/mysql-keyring/ r,
    /var/lib/mysql-keyring/** rwk,

  # Allow log file access
    /var/log/mysql/ r,
    /var/log/mysql/** rw,
  ...
}
```

The settings govern how the files are accessed. For example, the data file directory access gives read (r) access to a directory and read, write, and lock access (rwk) to all directories and files underneath `/mysql/`.

You should download the apparmor-utils package when you are working with existing AppArmor profiles. The utilities allow you to edit a profile without stopping AppArmor or removing the profile.

Before you edit a profile, change the profile to `complain` mode:

```
$ aa-complain /usr/sbin/mysqld
```

The output could be the following:

```
setting /usr/sbin/mysqld to complain mode
```

In complain mode, you can edit the profile to add settings because you have relocated the data directory: `/<volume>/dev/percona/data`:

```
    /<volume>/percona/data/ r,
    /<volume>/percona/data/** rwk,


You may need to reload AppArmor or reload the specific AppArmor profile to apply the changes.
```

You can also modify the `/etc/apparmor.d/tunables/alias` file as follows:

```
$ alias /var/lib/mysql -/volume/percona/data/
```

To reload one profile, run the following command:

```
$ sudo apparmor_parser -r /etc/apparmor.d/usr.sbin.mysqld
```

Restart AppArmor with the following command:

```
$ sudo systemctl restart apparmor
```

You can also disable AppArmor, but this action is not recommended. For earlier Ubuntu systems, prior to 16.04, use the following command:

```
$ sudo systemctl stop apparmor
$ sudo update-rc.d -f apparmor remove
```

For later Ubuntu systems, use the following:

```
$ sudo sudo systemctl stop apparmor
$ sudo systemctl disable apparmor
```

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

Note

Debian and *Ubuntu* installation does not automatically create a special `debian-sys-maint` user which can be used by the control scripts to control the Percona Server for MySQL `mysqld` and `mysqld_safe` services like it was the case with previous Percona Server for MySQL versions. If you still require this user you must create the user manually.

3.2.4 Running Percona Server for MySQL

The following procedure runs the Percona Server for MySQL:

1. Starting the service

   Percona Server for MySQL starts automatically after installation unless the server encounters errors during the installation process. You can also manually start it by running the following command:

   ```
   shell
   $ sudo service mysql start
   ```

2. Confirming the service is running

   You can verify the service status by running the following command:

   ```
   shell
   $ service mysql status
   ```

3. Stopping the service

   You can stop the service by running the following command:

   ```
   shell
   S sudo service mysql stop
   ```

4. Restarting the service

   You can restart the service by running the following command:

   ```
   shell
   S sudo service mysql restart
   ```

Note

Debian 8.0 (jessie) and *Ubuntu* 16.04(Xenial) 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.

3.2.5 Uninstalling Percona Server for MySQL

To uninstall Percona Server for MySQL, you must remove all of the installed packages.

You have the following options:

- Removing packages with **apt remove** leaves the configuration and data files.
- Removing the packages with **apt purge** removes 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

```
shell
$ sudo service mysql stop
```

2. Remove the packages

   - Remove the packages. This option does not delete the configuration or data files. If you do not require these files, you must delete each file manually.

```
shell
$ sudo apt remove 'percona-server*'
```

   - Purge the packages. This option deletes packages, configuration, and data files. The option does not delete any configuration or data files stored in your home directory. You may need to delete some files manually.

```
shell
$ sudo apt purge 'percona-server*'
$ sudo apt autoremove -y
$ sudo apt autoclean
$ sudo rm -rf /etc/mysql
```

Note

In a regular expression, the `\*` (asterisk) matches zero or more of the preceding item. The single quotes prevent the shell from misinterpreting the asterisk as a shell command.

If you do not plan to upgrade, run the following commands to remove the data directory location:

```
$ rm -rf /var/lib/mysql
$ rm -rf /var/log/mysql
$ sudo apt purge percona-server*
```

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-11-27

Download PDF

3.3 Installing Percona Server for MySQL 5.7 on Red Hat Enterprise Linux and CentOS

Note

The following instructions install Percona Server for MySQL 5.7 using the Yum repository. The instructions to install Percona Server for MySQL 8.0 using the Yum repository are available at this location.

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.

We gather Telemetry data in the Percona packages and Docker images.

Note

The RPM packages for Red Hat Enterprise Linux 7 (and compatible derivatives) do not support TLSv1.3, because TLSv1.3 requires OpenSSL 1.1.1, which is currently not available on this platform.

3.3.1 What's in each RPM package?

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

- The `Percona-Server-server-57` package contains the server itself (the `mysqld` binary).
- The `Percona-Server-57-debuginfo` package contains debug symbols for the server.
- The `Percona-Server-client-57` package contains the command line client.
- The `Percona-Server-devel-57` package contains the header files needed to compile software using the client library.
- The `Percona-Server-shared-57` package includes the client shared library.
- The `Percona-Server-shared-compat` package includes shared libraries for software compiled against older versions of the client library. The following libraries are included in this package: `libmysqlclient.so.12`, `libmysqlclient.so.14`, `libmysqlclient.so.15`, `libmysqlclient.so.16`, and `libmysqlclient.so.18`.

  This package is not included in downloads for Red Hat Enterprise Linux 9 and derivatives.
- The `Percona-Server-test-57` package includes the test suite for Percona Server for MySQL.

3.3.2 Installing from the Percona YUM repository

Note

RHEL 8 and other EL8 systems enable the MySQL module by default. This module hides the Percona-provided packages and the module must be disabled to make these packages visible. The following command disables the module:

```
shell
$ sudo yum module disable mysql
```

1. Install the Percona yum repository by running the following command as a `root` user or with `sudo`:

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

2. Enable the Percona Server 5.7 repository:

   ```
   shell
   $ sudo percona-release setup ps57
   ```

   The output should resemble the following:

   ```
   text
   * Disabling all Percona Repositories
   * Enabling the Percona Server 5.7 repository
   * Enabling the Percona XtraBackup 2.4 repository
   ```

3. Test the repository. Make sure packages are available from the repository by executing the `yum list` command. We filter the results by the version number:

   ```
   $ yum list | grep 5.7.38-41.1
   ```

   The output should be similar to the following:

   ```
   text
   ...
   Percona-Server-57-debuginfo.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   Percona-Server-57-debugsource.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   Percona-Server-client-57-debuginfo.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   Percona-Server-rocksdb-57.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   Percona-Server-rocksdb-57-debuginfo.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   Percona-Server-server-57-debuginfo.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   Percona-Server-shared-57-debuginfo.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   Percona-Server-shared-compat-57.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   Percona-Server-test-57-debuginfo.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   Percona-Server-tokudb-57.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   Percona-Server-tokudb-57-debuginfo.x86_64
   5.7.38-41.1.el8                                 percona-release-x86_64
   ...
   ```

4. Install the packages. You can install *Percona Server for MySQL* by running the following command:

   ```
   shell
   $ yum install Percona-Server-server-57
   ```

Note

Percona Server for MySQL 5.7 comes with the TokuDB storage engine. You can find more information on how to install and enable the TokuDB storage in the TokuDB Installation guide.

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 must install the Percona repository first if this operation has not been done already. See installing from the Percona YUM repository

3.3.3 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. The following example downloads the Percona Server for MySQL 5.7.31-34 release packages for *CentOS* 7:

   ```
   $ wget https://www.percona.com/downloads/Percona-Server-5.
   7/Percona-Server-5.7.31-34/binary/redhat/7/x86_64/Percona-Server-5.7.31-34-r2e68637-el7-x86_64-
   bundle.tar
   ```

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

   ```
   $ tar xvf Percona-Server-5.7.31-34-r2e68637-el7-x86_64-bundle.tar
   ```

   You should see the following packages:

   ```
   shell
   ```

   `$ ls *.rpm` The output should be similar to the following:

   ```
   text
   Percona-Server-57-debuginfo-5.7.31-34.1.el7.x86_64.rpm
   Percona-Server-client-57-5.7.31-34.1.el7.x86_64.rpm
   Percona-Server-devel-57-5.7.31-34.1.el7.x86_64.rpm
   Percona-Server-rocksdb-57-5.7.31-34.1.el7.x86_64.rpm
   Percona-Server-server-57-5.7.31-34.1.el7.x86_64.rpm
   Percona-Server-shared-57-5.7.31-34.1.el7.x86_64.rpm
   Percona-Server-shared-compat-57-5.7.31-34.1.el7.x86_64.rpm
   Percona-Server-test-57-5.7.31-34.1.el7.x86_64.rpm
   Percona-Server-tokudb-57-5.7.31-34.1.el7.x86_64.rpm
   ```

3. Run the following command to Percona Server for MySQL 5.7:

   ```
   shell
   $ rpm -ivh Percona-Server-server-57-5.7.31-34.1.el7.x86_64.rpm \
   Percona-Server-client-57-5.7.31-34.1.el7.x86_64.rpm \
   Percona-Server-shared-57-5.7.31-34.1.el7.x86_64.rpm
   ```

This command only installs the packages required to run the Percona Server for MySQL 5.7.

Optionally, you can install either the TokuDB storage engine, adding `Percona-Server-tokudb-57-5.7.31-34.1.el7.x86_64.rpm` or the MyRocks storage engine, adding `Percona-Server-rocksdb-57-5.7.31-34.1.el7.x86_64.rpm` to the install command.

You can find more information on how to install and enable the TokuDB storage in the TokuDB Installation guide.

You can find more information on how to install and enable the MyRocks storage engine in Percona MyRocks Installation.

To install all the packages (for debugging, testing, etc.) run the following command:

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

Note

When installing packages manually, you must resolve all dependencies and install any missing packages.

The following table lists the default locations for files:

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
```

The output should resemble the following:

```
datadir=/var/lib/mysql
```

3.3.4 Running Percona Server for MySQL

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.

1. Start the service. Percona Server for MySQL does not start automatically on *RHEL* and *CentOS* after the installation. Start the server by running the following command:

   `shell`
   `$ service mysql start`

2. Confirm that service is running by running the following command:

   `shell`
   `$ service mysql status`

3. Stop the service by running the following command:

   `shell`
   `$ service mysql stop`

4. Restart the service by running the following command:

   `shell`
   `$ service mysql restart`

Note

The *RHEL* 8 distributions and derivatives have added system-wide cryptographic policies component. This component allows the configuration of cryptographic subsystems.

3.3.5 Uninstalling Percona Server for MySQL

To completely uninstall Percona Server for MySQL you must remove all the installed packages and data files.

1. Stop the Percona Server for MySQL service

```
shell
$ service mysql stop
```

2. Remove the packages

```
shell
$ yum remove Percona-Server*
```

3. Remove the data and configuration files:

Warning

This command removes all the packages and deletes all the data files (databases, tables, logs, etc.). Take a backup in case you need the data.

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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-11-27

Download PDF

3.4 Installing *Percona Server for MySQL* 5.7 from a Binary Tarball

Note

The following instructions install Percona Server for MySQL 5.7. The instructions to install Percona Server for MySQL 8.0 are available at this location.

In *Percona Server for MySQL* 5.7.31-34 and later, the multiple binary tarballs are replaced with the following:

Type	Name	Description
Full	Percona-Server--Linux.x86_64.glibc2.12.tar.gz	Contains binaries, libraries, test files, and debug symbols
Minimal	Percona-Server--Linux.x86_64.glibc2.12-minimal.tar.gz	Contains binaries, and libraries but does not include test files, or debug symbols.

Select the *Percona Server for MySQL* 5.7 version number and type of tarball for your installation. Both binary tarballs support all distributions.

In *Percona Server for MySQL* before 5.7.31-34, multiple tarballs are provided based on the *OpenSSL* library available in the distribution:

- ssl100 - for *Debian* prior to 9 and *Ubuntu* prior to 14.04 versions (`libssl.so.1.0.0 => /usr/lib/x86_64-linux-gnu/libssl.so.1.0.0`);
- 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`);

You can download the binary tarballs from the `Linux - Generic` section on the download page.

Fetch the correct binary tarball. The example fetches Percona Server for MySQL 5.7.38-41 for Debian 10:

```
$ wget https://downloads.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.38-41/
binary/debian/buster/x86_64/Percona-Server-5.7.38-41-rda46e5474f9-buster-x86_64-bundle.tar
```

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

3.5 Installing *Percona Server for MySQL* 5.7 from a Source Tarball

Note

The following instructions install Percona Server for MySQL 5.7 from a source tarball. The instructions to install Percona Server for MySQL 8.0 from a source tarball are available at this location.

Fetch and extract the source tarball from Percona Downloads. The following example downloads and extracts Percona Server for MySQL 5.7. 38-41 on Ubuntu 22.04:

```
$ wget https://downloads.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.38-41/
binary/debian/focal/x86_64/Percona-Server-5.7.38-41-rda46e5474f9-focal-x86_64-bundle.tar
```

The output lists the downloaded file:

```
...
Saving to: 'Percona-Server-5.7.38-41-rda46e5474f9-focal-x86_64-bundle.tar'
...
```

Extract the tar file:

```
$ tar xvf Percona-Server-5.7.38-41-rda46e5474f9-focal-x86_64-bundle.tar
```

The output lists the files:

```
libperconaserverclient20_5.7.38-41-1.focal_amd64.deb
libperconaserverclient20-dev_5.7.38-41-1.focal_amd64.deb
percona-server-5.7-dbg_5.7.38-41-1.focal_amd64.deb
percona-server-client-5.7_5.7.38-41-1.focal_amd64.deb
percona-server-common-5.7_5.7.38-41-1.focal_amd64.deb
percona-server-rocksdb-5.7_5.7.38-41-1.focal_amd64.deb
percona-server-server-5.7_5.7.38-41-1.focal_amd64.deb
percona-server-source-5.7_5.7.38-41-1.focal_amd64.deb
percona-server-test-5.7_5.7.38-41-1.focal_amd64.deb
percona-server-tokudb-5.7_5.7.38-41-1.focal_amd64.deb
```

Follow the instructions in Compiling Percona Server for MySQL 5.7 from Source to complete the installation.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

3.6 Compiling Percona Server for MySQL 5.7 from Source

Note

The following instructions compile Percona Server for MySQL 5.7. The instructions on how to compile Percona Server for MySQL 8.0 are available at this location.

After either fetching the source repository or extracting a source tarball (from Percona or one you generated yourself), you must configure and build Percona Server. Do the following:

1. Run `cmake` to configure the build. Specify the build options like you would for a MySQL build. You may require other options on your sever.

   This example configures *Percona Server for MySQL* with similar options to what Percona uses to produce the binaries:

   ```
   shell
   $ cmake . -DCMAKE_BUILD_TYPE=RelWithDebInfo -DBUILD_CONFIG=mysql_release -DFEATURE_SET=community -DWITH_EMBEDDED_SERVER=OFF
   ```

2. compile using `make`:

   ```
   shell
   $ make
   ```

3. install the compiled file:

   ```
   shell
   $ make install
   ```
   Percona Server 5.7 is installed on your system.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

3.7 Installing Percona Server for MySQL 5.7 from the Git Source Tree

Note

The following instructions install Percona Server for MySQL 5.7 from the Git Source tree. The instruction to install the Percona Server for MySQL 8.0 from a Git Source tree are available in this location.

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

You can now fetch the latest *Percona Server for MySQL* 5.7 sources.

```
$ git clone https://github.com/percona/percona-server.git
$ cd percona-server
$ git checkout 5.7
$ git submodule init
$ git submodule update
```

If you are going to be making changes to *Percona Server for MySQL* 5.7 and wanting to distribute the resulting work, generate a new source tarball, which is process we follow for release:

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

Follow the instructions in Compiling Percona Server for MySQL from Source.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

4. Run in Docker

Download PDF

4.1 Running Percona Server for MySQL 5.7 in a Docker Container

Note

The following instructions run Percona Server for MySQL 5.7 in a Docker container. The instructions on how to run Percona Server for MySQL 8.0 in a Docker container are available at this location.

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.

We gather Telemetry data in the Percona packages and Docker images.

4.1.1 Using the Percona Server Images

The following procedure describes how to run and access Percona Server 5.7 using Docker.

Starting a Percona Server Instance in a Container

Note

By default, Docker pulls the image from Docker Hub if it is not available locally.

To start a container named `ps` running the latest version in the Percona Server 5.7 series, with the root password set to `root`:

```
[root@docker-host] $ docker run -d \
  --name ps \
  -e MYSQL_ROOT_PASSWORD=root \
  percona/percona-server:5.7
```

Warning

`root` is not a secure password. The word is used in the example for illustrative purposes only. Do not use this example in production.

Note

The docker stop command sends a TERM signal. Docker waits 10 seconds and sends a KILL signal. A very large instance 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`

4.1.2 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
```

The output may be similar to the following:

```
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>
```

4.1.3 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.

4.1.4 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:

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

Note

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

4.1.5 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:5.7
```

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 have the Percona Server container instance with a data directory that already contains data (the `mysql` subdirectory where all our system tables are stored), the 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:

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

4.1.6 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:5.7
```

4.1.7 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:5.7 \
 --character-set-server=utf8 \
 --collation-server=utf8_general_ci
```

5. Upgrade

Download PDF

5.1 Upgrade from earlier versions

An upgrade is supported from 5.6 to 5.7. We recommend that you upgrade to the latest version of 5.6 before upgrading to 5.7. You cannot skip versions, for example, you cannot upgrade from 5.5 to 5.7.

The following upgrade docs are available:

- Upgrading guide from 5.6 to 5.7
- Upgrade in-place with install Percona packages
- Upgrade using Percona repositories
- Upgrade using standalone packages

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-03-06

Download PDF

5.2 Performing a Distribution upgrade in-place on a System with installed Percona packages

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

1. Record the installed Percona packages
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 upgraded version of the distribution

Last update: 2022-09-27

Download PDF

5.3 Upgrading using the Percona repositories

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

5.3.1 `DEB` -based distributions

Note

Following commands will need to be run either as a root user or with **sudo**.

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

```
$ service mysql stop
```

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

Note

If you're running *Debian*/*Ubuntu* system with systemd as the default system and service manager you can invoke the above command with **systemctl** instead of **service**. Currently, both are supported.

Then install the new server with:

```
$ apt install percona-server-server-5.7
```

If you're using Percona Server for MySQL 5.6 with TokuDB you'll need to specify the TokuDB package as well:

```
$ apt install percona-server-server-5.7 percona-server-tokudb-5.7
```

The installation script will *NOT* run automatically **mysql_upgrade** as it was the case in previous versions. You'll need to run the command manually and restart the service after it's finished.

```
$ mysql_upgrade
```

The output should be similar to the following:

```
Checking if update is needed.
Checking server version.
Running queries to upgrade MySQL server.
Checking system database.
mysql.columns_priv                                OK
mysql.db                                          OK
mysql.engine_cost                                 OK
...
Upgrade process completed successfully.
Checking if update is needed.
```

Restart the service.

```
$ service mysql restart
```

Note that this procedure is the same for upgrading from MySQL 5.6 or 5.7 to Percona Server for MySQL 5.7.

5.3.2 `RPM`-based distributions

Note

Following commands will need to be run either as a root user or with **sudo**.

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

Note

If you're running *RHEL/CentOS* system with systemd as the default system and service manager you can invoke the above command with **systemctl** instead of **service**. Currently, both are supported.

```
$ service mysql stop
```

Check your installed packages with:

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

The output should be similar to the following:

```
Percona-Server-shared-56-5.6.28-rel76.1.el7.x86_64
Percona-Server-server-56-5.6.28-rel76.1.el7.x86_64
Percona-Server-devel-56-5.6.28-rel76.1.el7.x86_64
Percona-Server-client-56-5.6.28-rel76.1.el7.x86_64
Percona-Server-test-56-5.6.28-rel76.1.el7.x86_64
Percona-Server-56-debuginfo-5.6.28-rel76.1.el7.x86_64
```

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 packages, since they replace `mysql`, and will be removed if omitted.

Note that this procedure is the same for upgrading from MySQL 5.6 or 5.7 to Percona Server for MySQL 5.7: just grep `'^mysql-'` instead of `Percona-Server` and remove them.

Install `Percona-Server-server-57`:

```
$ yum install Percona-Server-server-57
```

Percona Server for MySQL 5.6 with TokuDB, specify the TokuDB package as well when doing the upgrade:

```
$ yum install Percona-Server-server-57 Percona-Server-tokudb-57
```

Once installed, proceed to modify your configuration file - `my.cnf` - and reinstall the plugins if necessary.

Note

If you're using TokuDB storage engine you'll need to comment out all the TokuDB specific variables in your configuration file(s) before starting the server, otherwise server won't be able to start. *RHEL/CentOS* 7 automatically backs up the previous configuration file to `/etc/my.cnf.rpmsave` and installs the default `my.cnf`. After upgrade/install process completes you can move the old configuration file back (after you remove all the unsupported system variables).

You can now start the `mysql` service:

```
$ service mysql start
```

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

Note

If you're using TokuDB storage engine, re-enable the storage engine plugin by running the: `ps-admin --enable-tokudb` before running `mysql_upgrade` otherwise you'll get errors.

```
$ mysql_upgrade
```

The output should be similar to the following:

```
Checking if update is needed.
Checking server version.
Running queries to upgrade MySQL server.
Checking system database.
mysql.columns_priv                                OK
mysql.db                                          OK
...
Upgrade process completed successfully.
Checking if update is needed.
```

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

```
$ service mysql restart
```

After the service has been successfully restarted you can use the new Percona Server for MySQL 5.7.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

5.4 Upgrading using Standalone Packages

You can also upgrade to *Percona Server for MySQL* using standalone packages.

5.4.1 DEB-based distributions

After taking a full backup and dump, if possible, stop the server:

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

Remove the installed packages with their dependencies:

```
$ sudo apt autoremove percona-server-server-5.6 percona-server-client-5.6
```

Once removed, proceed to do the modifications needed in your configuration file.

Then, download the following packages for your architecture:

- `percona-server-server-5.7`
- `percona-server-client-5.7`
- `percona-server-common-5.7`
- `libperconaserverclient20`

Following example will download Percona Server for MySQL Percona Server for MySQL 5.7.10-3 release packages for *Debian* 8.0:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.10-3/binary/
debian/jessie/x86_64/Percona-Server-5.7.10-3-r63dafaf-jessie-x86_64-bundle.tar
```

You should then unpack the bundle to get the packages:

```
$ tar xvf Percona-Server-5.7.10-3-r63dafaf-jessie-x86_64-bundle.tar
```

After you unpack the bundle, run `ls` to view the available packages:

```
$ ls *.deb
```

The output should be similar to the following:

```
libperconaserverclient20-dev_5.7.10-3-1.jessie_amd64.deb
libperconaserverclient20_5.7.10-3-1.jessie_amd64.deb
percona-server-5.7-dbg_5.7.10-3-1.jessie_amd64.deb
percona-server-client-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-common-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-server-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-source-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-test-5.7_5.7.10-3-1.jessie_amd64.deb
percona-server-tokudb-5.7_5.7.10-3-1.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 (`libperconaserverclient20_5.7.10-3-1.jessie_amd64.deb`, `percona-server-client-5.7_5.7.10-3-1.jessie_amd64.deb`, `percona-server-common-5.7_5.7.10-3-1.jessie_amd64.deb`, and `percona-server-server-5.7_5.7.10-3-1.jessie_amd64.deb`. Optionally you can install `percona-server-tokudb-5.7_5.7.10-3-1.jessie_amd64.deb` if you want TokuDB storage engine).

Note

Percona Server for MySQL 5.7 comes with the TokuDB storage engine. You can find more information on how to install and enable the TokuDB storage in the TokuDB Installation guide.

Warning

When installing packages manually, resolve all the dependencies and install any missing packages. The following packages should be installed before installing Percona Server for MySQL 5.7: `libmecab2`, `libjemalloc1`, `zlib1g-dev`, and `libaio1`.

The installation script will not run automatically **mysql_upgrade**, so you'll need to run it yourself and restart the service afterwards.

RPM-based distributions

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

```
$ service mysql stop
```

and check your installed packages:

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

The output should be similar to the following:

```
Percona-Server-shared-56-5.6.28-rel76.1.el6.x86_64
Percona-Server-server-56-5.6.28-rel76.1.el6.x86_64
Percona-Server-client-56-5.6.28-rel76.1.el6.x86_64
Percona-Server-tokudb-56-5.6.28-rel76.1.el6.x86_64
```

You may have a fourth, `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 5.6 to Percona Server for MySQL 5.7, just grep `'^mysql-'` instead of `Percona-Server` and remove them.

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 5.7.10-3 release packages for *CentOS* 7:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.7/Percona-Server-5.7.10-3/binary/
redhat/7/x86_64/Percona-Server-5.7.10-3-r63dafaf-el7-x86_64-bundle.tar
```

You should then unpack the bundle to get the packages:

```
$ tar xvf Percona-Server-5.7.10-3-r63dafaf-el7-x86_64-bundle.tar
```

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

```
$ ls *.rpm
```

The output should be similar to the following:

```
Percona-Server-57-debuginfo-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-client-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-devel-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-server-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-shared-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-shared-compat-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-test-57-5.7.10-3.1.el7.x86_64.rpm
Percona-Server-tokudb-57-5.7.10-3.1.el7.x86_64.rpm
```

Now you can install Percona Server for MySQL 5.7 by running:

```
rpm -ivh Percona-Server-server-57-5.7.10-3.1.el7.x86_64.rpm \
Percona-Server-client-57-5.7.10-3.1.el7.x86_64.rpm \
Percona-Server-shared-57-5.7.10-3.1.el7.x86_64.rpm
```

This will install only packages required to run the Percona Server for MySQL 5.7. Optionally you can install TokuDB storage engine by adding the `Percona-Server-tokudb-57-5.7.10-3.1.el7.x86_64.rpm` to the command above. You can find more information on how to install and enable the TokuDB storage in the TokuDB Installation guide.

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.

Once installed, proceed to modify your configuration file - `my.cnf` - and install the plugins if necessary. If you're using TokuDB storage engine you'll need to comment out all the TokuDB specific variables in your configuration file(s) before starting the server, otherwise server won't be able to start. *RHEL*/*CentOS* 7 automatically backs up the previous configuration file to `/etc/my.cnf.rpmsave` and installs the default `my.cnf`. After upgrade/install process completes you can move the old configuration file back (after you remove all the unsupported system variables).

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

```
$ service mysql start
```

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

Note

If you're using TokuDB storage engine you'll need re-enable the storage engine plugin by running the: `ps-admin --enable-tokudb` before running `mysql_upgrade` otherwise you'll get errors.

```
$ mysql_upgrade
```

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

```
$ service mysql restart
```

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

5.5 Percona Server In-Place Upgrading Guide: From 5.6 to 5.7

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 5.7 series. It's strongly recommended to that you read the following guides as they contain the list of incompatible changes that could cause automatic upgrade to fail:

- Changed in Percona Server 5.7
- Upgrading MySQL
- Upgrading from MySQL 5.6 to 5.7

Warning

Upgrade from 5.6 to 5.7 on a crashed instance is not recommended. If the server instance has crashed, crash recovery should be run before proceeding with the upgrade.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

6. Post-Installation

Download PDF

6.1 Post-Installation steps for Percona Server for MySQL 5.7

After you have installed *Percona Server for MySQL* 5.7, you may need to do the following:

Task	Description
Initialize the data directory	The source distribution or generic binary distribution installation does not automatically initialize the data directory
Update the root password	The CentOS/RedHat installations set up a temporary root password.
Securing the Installation	The mysql_secure_installation script improves the security of the installation.
Checking the server status	Verify the server returns information
Configuring the Server to Start at Startup	Common method to start the server automatically
Testing the server	Verify the server installation
Populating the time zone tables	The time zone tables are created but are not populated.
Exclude Buffer Pool Pages from core files	Reduce the size of the core files by excluding the buffer pool

6.1.1 Initializing the Data Directory

If you install the server using either the source distribution or generic binary distribution files, the data directory is not initialized, and you must run the initialization process after installation.

Run mysqld with the –initialize option or the initialize-insecure option.

Executing mysqld with either option does the following:

- Verifies the existence of the data directory
- Initializes the system tablespace and related structures
- Creates system tables including grant tables, time zone tables, and server-side help tables
- Creates `root@localhost`

You should run the following steps with the `mysql` login.

1. Navigate to the MySQL directory. The example uses the default location.

   ```
   shell
   $ cd /usr/local/mysql
   ```

2. Create a directory for the MySQL files. The secure_file_priv uses the directory path as a value.

   ```
   shell
   $ mkdir mydata
   ```

   The `mysql` user account should have the `drwxr-x---` permissions. Four sections define the permissions; file or directory, User, Group, and Others.

   The first character designates if the permissions are for a file or directory. The first character is `d` for a directory.

   The rest of the sections are specified in three-character sets.

Permission	User	Group	Other
Read	Yes	Yes	No
Write	Yes	No	No
Execute	Yes	Yes	No

3. Run the command to initialize the data directory.

   ```
   shell
   $ bin/mysqld --initialize
   ```

6.1.2 Updating the `root` password

The RedHat and derivative distributions set up a temporary password when MySQL is installed. To reset the password, you must start MySQL with the `--skip-grant-tables` option and update the `user` table.

The initial password is located using the following command:

```
$ grep 'temporary password' /var/log/mysqld.log
```

Follow this procedure to reset the root password:

1. Stop MySQL.

```
shell
$ sudo systemctl stop mysqld
```

2. Set `--skip-grant-tables` as an environment option. This method lets you specify the option without modifying configuration files.

```
shell
$ sudo systemctl set-environment MYSQLD_OPTS="--skip-grant-tables
--skip-networking"
```

3. Restart MySQL to make the option change effective.

```
shell
$ sudo systemctl restart mysqld
```

4. Access MySQL as `root`.

```
shell
$ mysql -u root
```

5. Change the root password.

```
sql
mysql> FLUSH PRIVILEGES;
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY ('NewPassword');
mysql> FLUSH PRIVILEGES;
mysql> exit
```

6. Stop MySQL

```
shell
$ sudo systemctl stop mysqld
```

7. Reset the environment options.

```
shell
$ sudo systemctl unset-environment MYSQLD_OPTS
```

8. Start MySQL

```
shell
$ sudo systemctl start mysqld
```

9. Log in to MySQL using the root password.

```
shell
$ mysql -u root -p
```

if you have trouble logging in after following the steps, repeat the procedure but, instead of using the `ALTER USER` statement, modify the `user` table.

```
sql
mysql> UPDATE mysql.user SET authentication_string=PASSWORD
('NewPassword'), password_expired='N'
WHERE User='root' AND Host='localhost';
mysql>FLUSH PRIVILEGES;
```

6.1.3 Securing the Installation

The mysql_secure_installation script improves the security of the installation.

Running the script does the following:

- Changes the `root` password
- Disallows remote login for `root` accounts
- Removes anonymous users
- Removes the `test` database
- Reloads the privilege tables

The following statement runs the script:

```
$ mysql_secure_installation
```

6.1.4 Checking the server status

After a generic binary installation, the server starts. The following command checks the server status:

```
$ sudo service mysql status
```

Access the server with the following command:

```
$ mysql -u root -p
```

6.1.5 Configuring the Server to Start at Startup

You can manage the server with systemd. If you have installed the server from a generic binary distribution on an operating system that uses systemd, you can manually configure systemd support.

The following commands start, check the status, and stop the server:

```
$ sudo systemctl start mysql
$ sudo systemctl status mysql
$ sudo systemctl stop mysql
```

Enabling the server to start at startup, run the following:

```
$ sudo systemctl enable mysql
```

6.1.6 Testing the Server

After you have initialized the data directory, and the server is started, you can run tests on the server.

This section assumes you have used the default installation settings. If you have modified the installation, navigate to the installation location. You can also add the location by Setting the Environment Variables.

You can use the mysqladmin client to access the server.

If you have issues connecting to the server, you should use the `root` user and the root account password.

```
$ sudo mysqladmin -u root -p version
```

The output should be similar to the following:

```
Enter password:

mysql Ver 8.0.19-10 for debian-linux-gnu on x86_64 (Percona Server (GPL), Release '10',
Revision 'f446c04')
...
Server version      8.0.19-10
Protocol version    10
Connection          Localhost via UNIX socket
UNIX socket         /var/run/mysqld/mysqld.sock
Uptime:             4 hours 58 min 10 section

Threads:    2 Questions:    16 Slow queries: 0 Opens: 139 Flush tables: 3
Open tables: 59  Queries per second avg: 0.0000
```

Use mysqlshow to display database and table information.

```
$ sudo mysqlshow -u root -p
```

The output should be similar to the following:

```
Enter password:

+--------------------+
|      Databases     |
+====================+
| information_schema |
+--------------------+
| mysql              |
+--------------------+
| performance_schema |
+--------------------+
| sys                |
+--------------------+
```

6.1.7 Populating the time zone tables

The time zone system tables are the following:

- `time_zone`
- `time_zone_leap_second`
- `time_zone_name`
- `time_zone_transition`
- `time_zone_transition_type`

If you install the server using either the source distribution or generic binary distribution files, the installation creates the time zone tables, but the tables are not populated.

The mysql_tzinfo_to_sql program populates the tables from the `zoneinfo` directory data available in Linux.

A common method to populate the tables is to add the zoneinfo directory path to `mysql_tzinfo_to_sql` and then send the output into `mysql`.

The example assumes you are running the command with the `root` account. You must use an account with the privileges able to modify MySQL system tables.

```
$ mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root -p rootpassword
```

6.1.8 Excluding Buffer Pool Pages from Core files

Implemented in *Percona Server for MySQL* 5.7.33-36, you can use the innodb_buffer_pool_in__core_file to reduce the size of the core file.

Buffer pools can produce large core files because the buffer pool is located in main memory. If the main memory is dumped to a core file, the buffer pool increases the size of the dump.

Having a large core file can have the following issues:

- Requires more time to write
- Consume disk space
- Reading the file

To exclude the buffer pool, run the following command at startup or use a `SET` statement:

```
mysqld> SET GLOBAL innodb_buffer_pool_in__core_file=OFF;
```

7. Diagnostics Improvements

Download PDF

7.1 User Statistics

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

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

7.1.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6.

7.1.2 Other Information

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

7.1.3 System Variables

`userstat`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	BOOLEAN
Default	OFF
Range	ON/OFF

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

`thread_statistics`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	BOOLEAN
Default	OFF
Range	ON/OFF

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

7.1.4 INFORMATION_SCHEMA Tables

`INFORMATION_SCHEMA.CLIENT_STATISTICS`

Column Name	Description
'CLIENT'	'The IP address or hostname from which the connection originated.'
'TOTAL_CONNECTIONS'	'The number of connections created for this client.'
'CONCURRENT_CONNECTIONS'	'The number of concurrent connections for this client.'
'CONNECTED_TIME'	'The cumulative number of seconds elapsed while there were connections from this client.'
'BUSY_TIME'	'The cumulative number of seconds there was activity on connections from this client.'
'CPU_TIME'	'The cumulative CPU time elapsed, in seconds, while servicing this client's connections.'
'BYTES_RECEIVED'	'The number of bytes received from this client's connections.'
'BYTES_SENT'	'The number of bytes sent to this client's connections.'
'BINLOG_BYTES_WRITTEN'	'The number of bytes written to the binary log from this client's connections.'
'ROWS_FETCHED'	'The number of rows fetched by this client's connections.'
'ROWS_UPDATED'	'The number of rows updated by this client's connections.'
'TABLE_ROWS_READ'	'The number of rows read from tables by this client's connections. (It may be different from ROWS_FETCHED.)'
'SELECT_COMMANDS'	'The number of SELECT commands executed from this client's connections.'
'UPDATE_COMMANDS'	'The number of UPDATE commands executed from this client's connections.'
'OTHER_COMMANDS'	'The number of other commands executed from this client's connections.'
'COMMIT_TRANSACTIONS'	'The number of COMMIT commands issued by this client's connections.'
'ROLLBACK_TRANSACTIONS'	'The number of ROLLBACK commands issued by this client's connections.'
'DENIED_CONNECTIONS'	'The number of connections denied to this client.'
'LOST_CONNECTIONS'	'The number of this client's connections that were terminated uncleanly.'
'ACCESS_DENIED'	'The number of times this client's connections issued commands that were denied.'
'EMPTY_QUERIES'	'The number of times this client's connections sent empty queries to the server.'

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

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.CLIENT_STATISTICS\G
```

The output could be similar to the following:

```
*************************** 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
```

7.1.5 INFORMATION_SCHEMA Tables

`INFORMATION_SCHEMA.INDEX_STATISTICS`

Column Name	Description
'TABLE_SCHEMA'	'The schema (database) name.'
'TABLE_NAME'	'The table name.'
'INDEX_NAME'	'The index name (as visible in SHOW CREATE TABLE).'
'ROWS_READ'	'The number of rows read from this index.'

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

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

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.INDEX_STATISTICS
   WHERE TABLE_NAME='tables_priv';
```

The output could be similar to the following:

```
+--------------+-----------------------+--------------------+-----------+
| TABLE_SCHEMA | TABLE_NAME            | INDEX_NAME         | ROWS_READ |
+--------------+-----------------------+--------------------+-----------+
| mysql        | tables_priv           | PRIMARY            |         2 |
+--------------+-----------------------+--------------------+-----------+
```

Note

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

`INFORMATION_SCHEMA.TABLE_STATISTICS`

Column Name	Description
'TABLE_SCHEMA'	'The schema (database) name.'
'TABLE_NAME'	'The table name.'
'ROWS_READ'	'The number of rows read from the table.'
'ROWS_CHANGED'	'The number of rows changed in the table.'
'ROWS_CHANGED_X_INDEXES'	'The number of rows changed in the table, multiplied by the number of indexes changed.'

This table is similar in function to the `INDEX_STATISTICS` table.

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.TABLE_STATISTICS
   WHERE TABLE_NAME=``tables_priv``;
```

The output could be similar to the following:

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

`INFORMATION_SCHEMA.THREAD_STATISTICS`

Column Name	Description
'THREAD_ID'	'Thread ID'
'TOTAL_CONNECTIONS'	'The number of connections created from this thread.'
'CONCURRENT_CONNECTIONS'	'Always zero, will be removed in a future version.'
'CONNECTED_TIME'	'The cumulative number of seconds elapsed while there were connections from this thread.'
'BUSY_TIME'	'The cumulative number of seconds there was activity from this thread.'
'CPU_TIME'	'The cumulative CPU time elapsed while servicing this thread.'
'BYTES_RECEIVED'	'The number of bytes received from this thread.'
'BYTES_SENT'	'The number of bytes sent to this thread.'
'BINLOG_BYTES_WRITTEN'	'The number of bytes written to the binary log from this thread.'
'ROWS_FETCHED'	'The number of rows fetched by this thread.'
'ROWS_UPDATED'	'The number of rows updated by this thread.'
'TABLE_ROWS_READ'	'The number of rows read from tables by this tread.'
'SELECT_COMMANDS'	'The number of SELECT commands executed from this thread.'
'UPDATE_COMMANDS'	'The number of UPDATE commands executed from this thread.'
'OTHER_COMMANDS'	'The number of other commands executed from this thread.'
'COMMIT_TRANSACTIONS'	'The number of COMMIT commands issued by this thread.'
'ROLLBACK_TRANSACTIONS'	'The number of ROLLBACK commands issued by this thread.'
'DENIED_CONNECTIONS'	'The number of connections denied to this thread.'
'LOST_CONNECTIONS'	'The number of thread connections that were terminated uncleanly.'
'ACCESS_DENIED'	'The number of times this thread issued commands that were denied.'
'EMPTY_QUERIES'	'The number of times this thread sent empty queries to the server.'
'TOTAL_SSL_CONNECTIONS'	'The number of thread connections that used SSL.'

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

`INFORMATION_SCHEMA.USER_STATISTICS`

Column Name	Description
'USER'	'The username. The value #mysql_system_user# appears when there is no username (such as for the replica SQL thread).'
'TOTAL_CONNECTIONS'	'The number of connections created from this user.'
'CONCURRENT_CONNECTIONS'	'The number of concurrent connections for this user.'
'CONNECTED_TIME'	'The cumulative number of seconds elapsed while there were connections from this user.'
'BUSY_TIME'	'The cumulative number of seconds there was activity on connections from this user.'
'CPU_TIME'	'The cumulative CPU time elapsed, in seconds, while servicing this user's connections.'
'BYTES_RECEIVED'	'The number of bytes received from this user's connections.'
'BYTES_SENT'	'The number of bytes sent to this user's connections.'
'BINLOG_BYTES_WRITTEN'	'The number of bytes written to the binary log from this user's connections.'
'ROWS_FETCHED'	'The number of rows fetched by this user's connections.'
'ROWS_UPDATED'	'The number of rows updated by this user's connections.'
'TABLE_ROWS_READ'	'The number of rows read from tables by this user's connections. (It may be different from ROWS_FETCHED.)'
'SELECT_COMMANDS'	'The number of SELECT commands executed from this user's connections.'
'UPDATE_COMMANDS'	'The number of UPDATE commands executed from this user's connections.'
'OTHER_COMMANDS'	'The number of other commands executed from this user's connections.'
'COMMIT_TRANSACTIONS'	'The number of COMMIT commands issued by this user's connections.'
'ROLLBACK_TRANSACTIONS'	'The number of ROLLBACK commands issued by this user's connections.'
'DENIED_CONNECTIONS'	'The number of connections denied to this user.'
'LOST_CONNECTIONS'	'The number of this user's connections that were terminated uncleanly.'
'ACCESS_DENIED'	'The number of times this user's connections issued commands that were denied.'
'EMPTY_QUERIES'	'The number of times this user's connections sent empty queries to the server.'

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

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

Example:

```
mysql> SELECT * FROM INFORMATION_SCHEMA.USER_STATISTICS\G
```

The output should be similar to the following:

```
*************************** 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
```

7.1.6 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.

7.1.7 Status Variables

`Com_show_client_statistics`

Option	Description
Scope	Global/Session
Data type	Numeric

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

`Com_show_index_statistics`

Option	Description
Scope	Global/Session
Data type	Numeric

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

`Com_show_table_statistics`

Option	Description
Scope	Global/Session
Data type	Numeric

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

`Com_show_thread_statistics`

Option	Description
Scope	Global/Session
Data type	Numeric

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

`Com_show_user_statistics`

Option	Description
Scope	Global/Session
Data type	Numeric

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

Last update: 2022-09-27

Download PDF

7.2 Slow Query Log

This feature adds microsecond time resolution and additional statistics to the slow query log output. It lets you enable or disable the slow query log at runtime, adds logging for the replica 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.

7.2.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6.

7.2.2 System Variables

`log_slow_filter`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global, Session
Dynamic	Yes

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

- `qc_miss`: The query was not found in the query cache.
- `full_scan`: The query performed a full table scan.
- `full_join`: The query performed a full join (a join without indexes).
- `tmp_table`: The query created an implicit internal temporary table.
- `tmp_table_on_disk`: The query's temporary table was stored on disk.
- `filesort`: The query used a filesort.
- `filesort_on_disk`: The filesort was performed on disk.

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

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

`log_slow_rate_type`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Enumerated
Default	session
Range	session, query

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

`log_slow_rate_limit`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global, session
Dynamic	Yes
Default	1
Range	1-1000

Behavior of this variable depends from log_slow_rate_type.

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

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

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

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

- if `log_slow_rate_limit` is 1 - log every query
- If `log_slow_rate_limit` > 1 - randomly log every 1/ `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* information about the log_slow_rate_limit has been added to the slow query log. This means that if the log_slow_rate_limit is effective it will be reflected in the slow query log for each written query. Example of the output looks like this:

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

`log_slow_sp_statements`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	TRUE
Range	TRUE/FALSE

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

Percona Server for MySQL implemented improvements for logging of stored procedures to the slow query log:

- 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 `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 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 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.

`log_slow_verbosity`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global, session
Dynamic	Yes

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

- `microtime`: Log queries with microsecond precision.
- `query_plan`: Log information about the query's execution plan.
- `innodb`: Log InnoDB statistics.
- `minimal`: Equivalent to enabling just `microtime`.
- `standard`: Equivalent to enabling `microtime,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`.

`slow_query_log_use_global_control`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Default	None

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

`slow_query_log_always_write_time`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Default	10

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

7.2.3 Other Information

Changes to the Log Format

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

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

Another example (log_slow_verbosity `=profiling`):

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

Notice that the `Killed: \`` keyword is followed by zero when the query successfully completes. If the query was killed, the ``Killed:` keyword is followed by a number other than zero:

Killed Numeric Code	Exception
0	NOT_KILLED
1	KILL_BAD_DATA
1053	ER_SERVER_SHUTDOWN (see MySQL Documentation)
1317	ER_QUERY_INTERRUPTED (see MySQL Documentation)
3024	ER_QUERY_TIMEOUT (see MySQL Documentation)
Any other number	KILLED_NO_VALUE (Catches all other cases)

Connection and Schema Identifier

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

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

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

Microsecond Time Resolution and Extra Row Information

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

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

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

Values and context:

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

Memory Footprint

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

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

Values and context:

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

Query Plan Information

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

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

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

InnoDB Usage Information

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

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

Values:

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

If the query did not use InnoDB tables, that information is written into the log instead of the above statistics.

7.2.4 Related Reading

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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

7.3 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):
  - `innodb_show_verbose_locks` - Whether to show locked records
  - `innodb_show_locks_held` - The number of locks held to print for each InnoDB transaction
- Added extended information about the InnoDB internal hash table sizes, in bytes, in the `BUFFER POOL AND MEMORY` section, also added a buffer pool size in bytes.
- Added additional LOG section information.

7.3.1 Version changes

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6.

7.3.2 Other Information

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

7.3.3 System Variables

`innodb_show_verbose_locks`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	ULONG
Default	0
Range	0 - 1

Specifies to show records locked in `SHOW ENGINE INNODB STATUS`. The default is `0`, which means only the higher-level information about the lock, for example, which table and index is locked, etc., is printed. If set to `1` enables the traditional InnoDB behavior. The locked records are dumped into the output.

`innodb_show_locks_held`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	ULONG
Default	10
Range	0 - 1000

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

`innodb_print_lock_wait_timeout_info`

Implemented in Percona Server 5.7.20-18.

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	OFF

Makes InnoDB 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. The query string can be obtained from the `EVENTS_STATEMENTS_CURRENT` table, based on the `PROCESSLIST_ID` field, which corresponds to `thread_id` from the log output.

Taking into account that blocking a transaction is often a multiple statement one, the following 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;
```

Note

The `PROCESSLIST_ID` in this example is exactly the thread id from the error log output.

7.3.4 Status Variables

The status variables here contain information available in the output of `SHOW ENGINE INNODB STATUS`, organized by the sections `SHOW ENGINE INNODB STATUS` displays. If you are familiar with the output of `SHOW ENGINE INNODB STATUS`, you will probably already recognize the information these variables contain.

BACKGROUND THREAD

The following variables contain information in the `BACKGROUND THREAD` section of the output from `SHOW ENGINE INNODB STATUS`. An example of that output is:

```
-----------------
BACKGROUND THREAD
-----------------
srv_master_thread loops: 1 srv_active, 0 srv_shutdown, 11844 srv_idle
srv_master_thread log flush and writes: 11844
```

InnoDB has a source thread that performs background tasks depending on the server state, once per second. If the server is under workload, the source 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, and makes a checkpoint.

`Innodb_master_thread_active_loops`

Option	Description
Scope	Global
Data type	Numeric

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

`Innodb_master_thread_idle_loops`

Option	Description
Scope	Global
Data type	Numeric

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

`Innodb_background_log_sync`

Option	Description
Scope	Global
Data type	Numeric

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

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
```

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
```

`Innodb_ibuf_free_list`

Option	Description
Scope	Global
Data type	Numeric

`Innodb_ibuf_segment_size`

Option	Description
Scope	Global
Data type	Numeric

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
```

`Innodb_lsn_current`

Option	Description
Scope	Global
Data type	Numeric

This variable shows the current log sequence number.

`Innodb_lsn_flushed`

Option	Description
Scope	Global
Data type	Numeric

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

`Innodb_lsn_last_checkpoint`

Option	Description
Scope	Global
Data type	Numeric

This variable shows the LSN of the latest completed checkpoint.

`Innodb_checkpoint_age`

Option	Description
Scope	Global
Data type	Numeric

This variable shows the current InnoDB checkpoint age, for example, the difference between the current LSN and the LSN of the last completed checkpoint.

`Innodb_checkpoint_max_age`

Option	Description
Scope	Global
Data type	Numeric

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

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
```

`Innodb_mem_adaptive_hash`

Option	Description
Scope	Global
Data type	Numeric

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

`Innodb_mem_dictionary`

Option	Description
Scope	Global
Data type	Numeric

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

`Innodb_mem_total`

Option	Description
Scope	Global
Data type	Numeric

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

`Innodb_buffer_pool_pages_LRU_flushed`

Option	Description
Scope	Global
Data type	Numeric

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

`Innodb_buffer_pool_pages_made_not_young`

Option	Description
Scope	Global
Data type	Numeric

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

`Innodb_buffer_pool_pages_made_young`

Option	Description
Scope	Global
Data type	Numeric

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.

`Innodb_buffer_pool_pages_old`

Option	Description
Scope	Global
Data type	Numeric

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.

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
```

`Innodb_max_trx_id`

Option	Description
Scope	Global
Data type	Numeric

This variable shows the next free transaction id number.

`Innodb_oldest_view_low_limit_trx_id`

Option	Description
Scope	Global
Data type	Numeric

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.

`Innodb_purge_trx_id`

Option	Description
Scope	Global
Data type	Numeric

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

`Innodb_purge_undo_no`

Option	Description
Scope	Global
Data type	Numeric

7.3.5 INFORMATION_SCHEMA Tables

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

`INFORMATION_SCHEMA.XTRADB_READ_VIEW`

Column Name	Description
`READ_VIEW_LOW_LIMIT_TRX_NUMBER`	This is the highest transactions number at the time the view was created.
`READ_VIEW_UPPER_LIMIT_TRX_ID`	This is the highest transactions ID at the time the view was created. This means that it should not see newer transactions with IDs bigger than or equal to that value.
`READ_VIEW_LOW_LIMIT_TRX_ID`	This is the latest committed transaction ID at the time the oldest view was created. This means that it should see all transactions with IDs smaller than or equal to that value.

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

`INFORMATION_SCHEMA.XTRADB_INTERNAL_HASH_TABLES`

Column Name	Description
`INTERNAL_HASH_TABLE_NAME`	Hash table name
`TOTAL_MEMORY`	Total amount of memory
`CONSTANT_MEMORY`	Constant memory
`VARIABLE_MEMORY`	Variable memory

7.3.6 Other reading

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

Last update: 2022-09-27

Download PDF

7.4 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;
```

The output could be similar to the following:

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

The output could be similar to the following:

```
+------------+---------
+-------------------------------------------------------------------------------+--------------
+------+------------+
| Engine     | Support |
Comment                                                                | Transactions
|   XA | Savepoints |
+------------+---------
+-------------------------------------------------------------------------------+--------------
+------+------------+
| InnoDB     | YES     | Percona-XtraDB, Supports transactions, row-level locking, and
foreign keys |          YES | YES  | YES        |
...
+------------+---------
+-------------------------------------------------------------------------------+--------------
+------+------------+
```

7.4.1 Version-Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

7.5 Process List

This page describes Percona changes to both the standard MySQL `SHOW PROCESSLIST` command and the standard MySQL `INFORMATION_SCHEMA` table `PROCESSLIST`.

7.5.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

7.5.2 INFORMATION_SCHEMA Tables

`INFORMATION_SCHEMA.PROCESSLIST`

This table implements modifications to the standard MySQL `INFORMATION_SCHEMA` table `PROCESSLIST`.

Column Name	Description
'ID'	'The connection identifier.'
'USER'	'The MySQL user who issued the statement.'
'HOST'	'The host name of the client issuing the statement.'
'DB'	'The default database, if one is selected, otherwise NULL.'
'COMMAND'	'The type of command the thread is executing.'
'TIME'	'The time in seconds that the thread has been in its current state.'
'STATE'	'An action, event, or state that indicates what the thread is doing.'
'INFO'	'The statement that the thread is executing, or NULL if it is not executing any statement.'
'TIME_MS'	'The time in milliseconds that the thread has been in its current state.'
'ROWS_EXAMINED'	'The number of rows examined by the statement being executed (NOTE: This column is not updated for each examined row so it does not necessarily show an up-to-date value while the statement is executing. It only shows a correct value after the statement has completed.).'
'ROWS_SENT'	'The number of rows sent by the statement being executed.'
'TID'	'The Linux Thread ID. For Linux, this corresponds to light-weight process ID (LWP ID) and can be seen in the ps -L output. In case when Thread Pool is enabled, "TID" is not null for only currently executing statements and statements received via "extra" connection.'

7.5.3 Example Output

Table PROCESSLIST:

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

The output could be similar to the following:

```
+----+------+-----------+--------------------+---------+------+-----------
+---------------------------+---------+-----------+---------------+
```

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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

7.6 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.

7.6.1 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.

`INFORMATION_SCHEMA.GLOBAL_TEMPORARY_TABLES`

Percona Server for MySQL 5.7.10-1 - Feature ported from *Percona Server for MySQL* 5.6

Column Name	Description
`SESSION_ID`	'MySQL connection id'
`TABLE_SCHEMA`	'Schema in which the temporary table is created'
`TABLE_NAME`	'Name of the temporary table'
`ENGINE`	'Engine of the temporary table'
`NAME`	'Internal name of the temporary table'
`TABLE_ROWS`	'Number of rows of the temporary table'
`AVG_ROW_LENGTH`	'Average row length of the temporary table'
`DATA_LENGTH`	'Size of the data (Bytes)'
`INDEX_LENGTH`	'Size of the indexes (Bytes)'
`CREATE_TIME`	'Date and time of the creation of the temporary table'
`UPDATE_TIME`	'Date and time of the latest update of the temporary table'

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

`INFORMATION_SCHEMA.TEMPORARY_TABLES`

Percona Server for MySQL 5.7.10-1 - Feature ported from *Percona Server for MySQL* 5.6

Column Name	Description
`SESSION_ID`	'MySQL connection id'
`TABLE_SCHEMA`	'Schema in which the temporary table is created'
`TABLE_NAME`	'Name of the temporary table'
`ENGINE`	'Engine of the temporary table'
`NAME`	'Internal name of the temporary table'
`TABLE_ROWS`	'Number of rows of the temporary table'
`AVG_ROW_LENGTH`	'Average row length of the temporary table'
`DATA_LENGTH`	'Size of the data (Bytes)'
`INDEX_LENGTH`	'Size of the indexes (Bytes)'
`CREATE_TIME`	'Date and time of the creation of the temporary table'
`UPDATE_TIME`	'Date and time of the latest update of the temporary table'

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

7.6.2 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`.

7.6.3 `INFORMATION_SCHEMA` Tables

This feature provides the following table:

`INFORMATION_SCHEMA.XTRADB_RSEG`

Column Name	Description
`rseg_id`	'rollback segment id'
`space_id`	'space where the segment placed'
`physical_page_size`	'physical page size'
`logical_page_size`	'logical page size'
`is_compressed`	'is the page compressed'
`page_no`	'page number of the segment header'
`max_size`	'max size in pages'
`curr_size`	'current size in pages'

This table shows information about all the rollback segments (the default segment and the extra segments).

Here is an example of output with `innodb_rollback_segments = 8`:

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

The output resembles the following:

```
+---------+----------+--------------------+-------------------+---------------+---------
+------------+-----------+
| rseg_id | space_id | physical_page_size | logical_page_size | is_compressed | page_no |
max_size   | curr_size |
+---------+----------+--------------------+-------------------+---------------+---------
+------------+-----------+
|       0 |        0 |              16384 |             16384 |             0 |       6 |
4294967294 |         2 |
|       1 |       24 |              16384 |             16384 |             0 |       3 |
4294967294 |         1 |
|       2 |       24 |              16384 |             16384 |             0 |       4 |
4294967294 |         1 |
|       3 |       24 |              16384 |             16384 |             0 |       5 |
4294967294 |         1 |
|       4 |       24 |              16384 |             16384 |             0 |       6 |
4294967294 |         1 |
|       5 |       24 |              16384 |             16384 |             0 |       7 |
4294967294 |         1 |
|       6 |       24 |              16384 |             16384 |             0 |       8 |
4294967294 |         1 |
|       7 |       24 |              16384 |             16384 |             0 |       9 |
4294967294 |         1 |
|       8 |       24 |              16384 |             16384 |             0 |      10 |
4294967294 |         1 |
+---------+----------+--------------------+-------------------+---------------+---------
+------------+-----------+
9 rows in set (0.00 sec)
```

Download PDF

7.7 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.

7.7.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

Download PDF

7.8 Metrics for scalability measurement

Note

This feature has been deprecated in *Percona Server for MySQL* Percona Server for MySQL 5.7.16-10. 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. These metrics allow using Little's Law, queuing theory, and Universal Scalability Law to gain insights into server performance. This feature is implemented as a plugin.

7.8.1 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     |
+--------------------------------+----------+--------------------+-----------------------
+---------+
```

7.8.2 System Variables

`scalability_metrics_control`

Option	Description
Command-line	Yes
Scope	Global
Dynamic	Yes
Data type	String
Default	OFF
Values	OFF, ON, RESET

This variable can be used to enable and disable the collection of metrics for scalability measurement. By setting the value to `RESET` all counters will be reset while continuing to count metrics.

7.8.3 Status Variables

`scalability_metrics_elapsedtime`

Option	Description
Data type	Numeric

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

`scalability_metrics_queries`

Option	Description
Data type	Numeric

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

`scalability_metrics_concurrency`

Option	Description
Data type	Numeric

This status variable shows number of queries currently executed.

`scalability_metrics_totaltime`

Option	Description
Data type	Numeric

This status variable shows total execution time of all queries, including the in-progress time of currently executing queries, in microseconds (ie. if two queries executed with 1 second of response time each, the result is 2 seconds).

`scalability_metrics_busytime`

Option	Description
Data type	Numeric

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

7.8.4 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6
- Percona Server for MySQL 5.7.16-10: Feature has been deprecated.

7.8.5 Other Reading

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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

7.9 Response Time Distribution

The slow query log provides exact information about queries that take a long time to execute. Sometimes there are a large number of queries that each take a 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 afterthe initial locks are acquired. The user can define time intervals that divide the range from 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 replica 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
```

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

Percona Server for MySQL is now able to log the queries response times into separate `READ` and `WRITE` `INFORMATION_SCHEMA` tables. The two new tables are named INFORMATION_SCHEMA.QUERY_RESPONSE_TIME_READ and INFORMATION_SCHEMA.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 INFORMATION_SCHEMA.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 INFORMATION_SCHEMA.QUERY_RESPONSE_TIME_READ table.

7.9.2 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 (INFORMATION_SCHEMA.QUERY_RESPONSE_TIME) to output gathered statistics.

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

This plugin provides the interface (INFORMATION_SCHEMA.QUERY_RESPONSE_TIME_READ) to output gathered statistics.

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

This plugin provides the interface (INFORMATION_SCHEMA.QUERY_RESPONSE_TIME_WRITE) to output gathered statistics.

You can check if plugins are installed correctly by running:

```
mysql> SHOW PLUGINS;
```

The output could be similar to the following:

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

7.9.3 Usage

To start collecting query time metrics, 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
```

SELECT

You can get the distribution using the query:

```
mysql> SELECT * from INFORMATION_SCHEMA.QUERY_RESPONSE_TIME;
```

The output could be similar to the following:

```
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 query_response_time_stats is ON, the execution times for these two `SELECT` queries will also be collected.

FLUSH

Flushing can be done by setting the query_response_time_flush to `ON` (or `1`):

```
mysql> SET GLOBAL query_response_time_flush='ON';
```

`FLUSH` does two things:

- Clears the collected times from the INFORMATION_SCHEMA.QUERY_RESPONSE_TIME, INFORMATION_SCHEMA.QUERY_RESPONSE_TIME_READ, and INFORMATION_SCHEMA.QUERY_RESPONSE_TIME_WRITE tables
- Reads the value of query_response_time_range_base and uses it to set the range base for the table

Note

The execution time for the `FLUSH` query will also be collected.

Stored procedures

Stored procedure calls count as a single query.

Collect time point

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

7.9.4 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

7.9.5 System Variables

`query_response_time_flush`

Option	Description
Command-line	Yes
Config file	No
Scope	Global
Dynamic	No
Data type	Boolean
Default	OFF
Range	OFF/ON

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

`query_response_time_range_base`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	10
Range	2-1000

Sets up the logarithm base for the scale.

Note

The variable takes effect only after this command has been executed:

```
mysql> SET GLOBAL query_response_time_flush=1;
```

`query_response_time_stats`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	OFF
Range	ON/OFF

This global variable enables and disables collection of query times.

`query_response_time_session_stats`

Option	Description
Command-line	No
Config file	No
Scope	Session
Dynamic	Yes
Data type	Text
Default	GLOBAL
Range	ON/OFF/GLOBAL

This variable enables and disables collection of query times on session level, thus customizing QRT behavior for individual connections. By default, its value is GLOBAL, which means that its value is taken from the query_response_time_stats variable.

7.9.6 INFORMATION_SCHEMA Tables

`INFORMATION_SCHEMA.QUERY_RESPONSE_TIME`

Column Name	Description
'VARCHAR TIME'	'Interval range in which the query occurred'
'INT(11) COUNT'	'Number of queries with execution times that fell into that interval'
'VARCHAR TOTAL'	'Total execution time of the queries '

`INFORMATION_SCHEMA.QUERY_RESPONSE_TIME_READ`

Column Name	Description
'VARCHAR TIME'	'Interval range in which the query occurred'
'INT(11) COUNT'	'Number of queries with execution times that fell into that interval'
'VARCHAR TOTAL'	'Total execution time of the queries '

`INFORMATION_SCHEMA.QUERY_RESPONSE_TIME_WRITE`

Column Name	Description
'VARCHAR TIME'	'Interval range in which the query occurred'
'INT(11) COUNT'	'Number of queries with execution times that fell into that interval'
'VARCHAR TOTAL'	'Total execution time of the queries '

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

7.10 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 the same as the index ordering of the records on the pages. Some pages take more space and full table scan queries can take more time to finish.

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

7.10.1 Version Specific Information

- Percona Server 5.7.20-18: Feature Implemented

7.10.2 Status Variables

`Innodb_scan_pages_contiguous`

Option	Description
Scope	Session
Data type	Numeric

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

`Innodb_scan_pages_disjointed`

Option	Description
Scope	Session
Data type	Numeric

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

`Innodb_scan_data_size`

Option	Description
Scope	Session
Data type	Numeric

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.

`Innodb_scan_deleted_recs_size`

Option	Description
Scope	Session
Data type	Numeric

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.

`Innodb_scan_pages_total_seek_distance`

Option	Description
Scope	Session
Data type	Numeric

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

7.10.3 Related Reading

- InnoDB: look after fragmentation
- Defragmenting a Table

Last update: 2022-09-27

Download PDF

7.11 Using the libcoredumper

This feature was implemented in *Percona Server for MySQL* 5.7.31-34 and has been tested against the supported operating systems for that version.

Operating Systems	Versions
Debian	9, 10
Red Hat Enterprise Linux and derivatives	6, 7, 8
Ubuntu	16.04, 18.04, 20.04

The documented moment of a computer when either the computer or an application crashed is a core dump file. Developers examine the dump as one of the tasks when searching for the cause of a failure.

The `libcoredumper` is a free and Open Source fork of `google-coredumper`, enhanced to work on newer Linux versions, and GCC and CLANG.

Test this tool in a safe environment before using in production.

7.11.1 Enable the libcoredumper

Enable core dumps for troubleshooting purposes.

To enable the `libcoredumper`, add the `coredumper` variable to the `mysqld` section of `my.cnf`. This variable is independent of the older `core-file` variable.

The variable can have the following possible values:

Value	Description
Blank	The core dump is saved under MySQL datadir and named `core`.
A path ending with `/`	The core dump is saved under the specified directory and named `core`.
Full path with a filename	The core dump is saved under the specified directory and with the specified filename

Restart the server.

7.11.2 Verify the tool is active

MySQL writes to the log when generating a core file and delegates the core dump operation to the Linux kernel. An example of the log message is the following:

```
Writing a core file
```

MySQL using the `libcoredumper` to generate the file creates the following message in the log:

```
Writing a core file using lib coredumper
```

Every core file adds a crash timestamp instead of a PID for the following reasons:

- Correlates the core file with a crash. MySQL prints a UTC timestamp on the crash log.

```
text
10:02:09 UTC - mysqld got signal 11;
```

- Keeps multiple core files.

  For example, the operators and containers run as PID 1. If the process ID identified the core file, each container crash generates a core dump that overwrites the previous core file.

Disable the libcoredumper

You can disable the libcoredumper. A core file may contain sensitive data and takes disk space.

To disable the `libcoredumper`, do the following:

1. In the `mysqld` section of my.cnf, remove the `libcoredumper` variable.
2. Restart the server.

Download PDF

7.12 Stack Trace

Developers use the stack trace in the debug process, either an interactive investigation or during the post-mortem. No configuration is required to generate a stack trace.

Implemented in *Percona Server for MySQL* 5.7.31-34, the stack trace adds the following:

Name	Description
Prints binary BuildID	The Strip utility removes unneeded sections and debugging information to reduce the size. This method is standard with containers where the size of the image is essential. The BuildID lets you resolve the stack trace when the Strip utility removes the binary symbols table.
Print the server version information	The version information establishes the starting point for analysis. Some applications, such as MySQL, only print this information to a log on startup, and when the crash occurs, the size of the log may be large, rotated, or truncated.

Last update: 2022-09-27

8. Flexibility Improvements

Download PDF

8.1 Suppress Warning Messages

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

```
mysql> SET GLOBAL log_warnings_suppress = '1592';
```

The output could be similar to the following:

```
Query OK, 0 rows affected (0.00 sec)
```

8.1.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Variable log_warnings_suppress ported from *Percona Server for MySQL* 5.6.
- Percona Server for MySQL 5.7.11-4: Feature has been removed from *Percona Server for MySQL* 5.7 because MySQL 5.7.11 has implemented a new system variable, log_statements_unsafe_for_binlog, which implements the same effect.

8.1.2 System Variables

`innodb_ft_ignore_stopwords`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	SET
Default	`(empty string)`
Range	`(empty string)`, `1592`

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

8.1.3 Related Reading

- MySQL bug 42851

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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

8.2 Improved MEMORY Storage Engine

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

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

This implementation is based on the *Dynamic Row Format* (`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.

8.2.1 Caveats

Ordering of Rows

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

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

Indexing

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

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

8.2.2 Restrictions

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

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

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

This sets two restrictions to tables:

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

Ordering of Columns

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

Minimum Block Size

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

8.2.3 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.

8.2.4 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.

Implicit Request

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

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

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

While this will not:

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

Explicit Request

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

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

Despite its name, the `KEY_BLOCK_SIZE` option refers to a block size used to store data rather 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`.

8.2.5 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';
```

The output should be simmilar to the following:

```
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';
```

The output should be simmilar to the following:

```
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
```

8.2.6 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\*)`.

8.2.7 See Also

- Dynamic row format for MEMORY tables

Last update: 2022-09-27

Download PDF

8.3 Restricting the number of binlog files

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

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

8.3.1 Example

Number of the binlog files before setting this variable

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

The output could be the following:

```
26
```

Variable max_binlog_files is set to 20:

```
max_binlog_files = 20
```

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

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

The output could be the following:

```
20
```

8.3.2 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Variable max_binlog_files ported from *Percona Server for MySQL* 5.6.
- Percona Server 5.7.23-23: Variable max_binlog_files is deprecated and replaced with binlog_space_limit.

8.3.3 System Variables

max_binlog_files

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	ULONG
Default	0 (unlimited)
Range	0-102400

binlog_space_limit

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	ULONG
Default	0 (unlimited)
Range	0-102400

This option places an upper limit on the total size in bytes of all binary logs. A value of `0` means "no limit". This is useful for a server host that has limited disk space.

When the limit is reached, oldest binary logs are purged until the total size is under the limit or only active log is remaining.

Note

You should not set `--binlog-space-limit` to less or equal than the value of `--max-binlog-size` because after the max-binlog-size limit will be reached, logs will be rotated and immediately pruned by binlog-space-limit.

Last update: 2022-09-27

Download PDF

8.4 Extended mysqldump

8.4.1 Backup Locks support

In *Percona Server for MySQL* Percona Server for MySQL 5.7.10-1 `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.

8.4.2 Compressed Columns support

In *Percona Server for MySQL* Percona Server for MySQL 5.7.17-11 **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.

8.4.3 Taking backup by descending primary key order

In Percona Server for MySQL 5.7.17-12 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.

8.4.4 RocksDB support

mysqldump will now detect when MyRocks is installed and available by seeing if there is a session variable named rocksdb_skip_fill_cache and setting it to `1` if it exists.

mysqldump will now automatically enable session variable rocksdb_bulk_load if it is supported by target server.

8.4.5 Version Specific Information

- Percona Server for MySQL 5.7.10-1: **mysqldump** has been extended with Backup Locks support options
- Percona Server for MySQL 5.7.17-11: **mysqldump** has been extended with Compressed columns with dictionaries support options
- Percona Server for MySQL 5.7.17-12: **mysqldump** option `--order-by-primary-desc` introduced

Last update: 2022-09-27

Download PDF

8.5 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.

8.5.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

8.5.2 Other Reading

- MySQL bug: #44835

Last update: 2022-09-27

Download PDF

8.6 Per-query variable statement

Percona Server for MySQL has implemented per-query variable statement support. 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>;
```

8.6.1 Examples

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

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

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

```
mysql> SET STATEMENT max_execution_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;
```

8.6.2 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

8.6.3 Other Reading

- WL#681: Per query variable settings

Download PDF

8.7 Extended mysqlbinlog

Percona Server for MySQL has implemented protocol compression support for the **mysqlbinlog** command.

You can request protocol compression when connecting to a remote server to transfer binary log files. The protocol compression helps reduce the bandwidth use and improves the transfer speed.

In the mysqlbinlog utility add either the `--compress` or `-C` flag to the command-line options.

```
mysqlbinlog [--compress|-C] --remote-server
```

8.7.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

8.8 Slow Query Log Rotation and Expiration

Note

This feature is currently considered BETA quality.

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

Warning

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

8.8.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

8.8.2 System Variables

`max_slowlog_size`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	numeric
Default	0 (unlimited)
Range	4096 - 1073741824

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

`max_slowlog_files`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	numeric
Default	0 (unlimited)
Range	0 - 102400

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

Download PDF

8.9 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 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.

8.9.1 Example

Table:

```
> 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` csv_quotes.

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

If the 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 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 csv_mode set to empty value, parsed data will look like:

```
> SELECT * FROM albums;
```

The output could be similar to the following:

```
+--------------+--------------------+
| artist       | album              |
+--------------+--------------------+
| Great Artist | Old Album          |
| Great Artist | ""Limited Edition" |
```

```
+--------------+--------------------+
2 rows in set (0.02 sec)
```

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

```
mysql> SET csv_mode = 'IETF_QUOTES';
```

The output could be similar to the following:

```
Query OK, 0 rows affected (0.00 sec)
```

```
> SELECT * FROM albums;
```

The output could be similar to the following:

```
+--------------+----------------------------+
| artist       | album                      |
+--------------+----------------------------+
| Great Artist | Old Album                  |
| Great Artist | "Limited Edition",Old Album |
+--------------+----------------------------+
```

8.9.2 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

8.9.3 System Variables

csv_mode

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global, Session
Dynamic	Yes
Data type	SET
Default	`(empty string)`
Range	`(empty string)`, `IETF_QUOTES`

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.

8.9.4 Related Reading

- MySQL bug #71091

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

8.10 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 protocol 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`.

8.10.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

8.10.2 System Variables

`proxy_protocol_networks`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Default	`(empty string)`

This variable is a global-only, read-only variable. The available values are:

- Empty string, which is the default
- List of comma-separated IPv4 network and host addresses, or IPv6 network and host addresses. Network addresses are specified in CIDR notation, i.e. `192.168.0.0/24`.
- An `\*` (asterisk) allows the proxy headers from any account. This setting is not recommended because this setting may compromise security.

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`.

8.10.3 Related Reading

- PROXY protocol specification

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

8.11 Per-session server-id

Variable server_id is a global variable. In multi-source replication setups or for external replication, 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-source replication.

This was implemented by introducing the new pseudo_server_id variable. This variable, when set to non-zero value, will cause all binary log events in that session to have that server_id value. A new variable was introduced instead of converting 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 pseudo_server_id. One special case is circular replication which definitely will be broken if you set pseudo_server_id to a value not assigned to any participating server (ie., setup is `1->2->3->4->1`, and pseudo_server_id is set to `5`). It is also possible to create a temporary table `foo`, then change pseudo_server_id and from now `foo` will not be visible by this session until pseudo_server_id gets restored.

8.11.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

8.11.2 System Variables

`pseudo_server_id`

Option	Description
Command-line	Yes
Config file	No
Scope	Session
Dynamic	Yes
Default	0

When this variable is set to `0` (default), it will use the global server_id value. **Note:** this is different from the setting the global 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 server_id value. Setting this variable requires `SUPER` privileges.

8.11.3 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

Last update: 2022-09-27

Download PDF

8.12 Compressed columns with dictionaries

In Percona Server for MySQL 5.7.17-11 *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 reading. For all other purposes, the data type is identical to the one without the 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 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
- a way to select columns in the table to compress (in contrast to the InnoDB row compression method)

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

8.12.1 Specifications

The feature is limited to InnoDB/XtraDB storage engine and to columns of the following data types:

- `BLOB` (including `TINYBLOB`, `MEDIUMBLOB`, `LONGBLOG`)
- `TEXT` (including `TINYTEXT`, `MEDUUMTEXT`, `LONGTEXT`)
- `VARCHAR` (including `NATIONAL VARCHAR`)
- `VARBINARY`
- `JSON`

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 to an unsupported column type or storage engine, an error is returned.

The compression can be specified:

- when creating a table: `CREATE TABLE ... (..., foo BLOB COLUMN_FORMAT COMPRESSED, ...);`
- when altering a table and modifying a column to the compressed format: `ALTER TABLE ... MODIFY [COLUMN] ... COLUMN_FORMAT COMPRESSED`, or `ALTER TABLE ... CHANGE [COLUMN] ... COLUMN_FORMAT COMPRESSED`.

Unlike Oracle MySQL, compression is applicable to generated stored columns. Use this syntax extension as follows:

```
mysql> CREATE TABLE t1(
        id INT,
        a BLOB,
        b JSON COLUMN_FORMAT COMPRESSED,
        g BLOB GENERATED ALWAYS AS (a) STORED COLUMN_FORMAT COMPRESSED WITH
COMPRESSION_DICTIONARY numbers
      ) ENGINE=InnoDB;
```

To decompress a column, specify a value other than `COMPRESSED` to `COLUMN_FORMAT`: `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: innodb_compressed_columns_zip_level and innodb_compressed_columns_threshold have been implemented.

8.12.2 Compression dictionary support

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

Compression dictionaries can be represented as a list of words in the 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.

Example

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

```
mysql> SET @dictionary_data = 'one' 'two' 'three' 'four';
```

The output should be similar to the following:

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> CREATE COMPRESSION_DICTIONARY numbers (@dictionary_data);
```

The output should be similar to the following:

```
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;
```

The 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);
```

The output should be similar to the following:

```
Query OK, 1 row affected (0.01 sec)
```

8.12.3 INFORMATION_SCHEMA Tables

This feature implemented two new `INFORMATION_SCHEMA` tables.

`INFORMATION_SCHEMA.XTRADB_ZIP_DICT`

Column Name	Description
'BIGINT(21)_UNSIGNED id'	'dictionary ID'
'VARCHAR(64) name'	'dictionary name'
'BLOB zip_dict'	'compression dictionary string'

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

`INFORMATION_SCHEMA.XTRADB_ZIP_DICT_COLS`

Column Name	Description
'BIGINT(21)_UNSIGNED table_id'	'table ID from `INFORMATION_SCHEMA.INNODB_SYS_TABLES`'
'BIGINT(21)_UNSIGNED column_pos'	'column position (starts from `0` as in `INFORMATION_SCHEMA.INNODB_SYS_COLUMNS`)'
'BIGINT(21)_UNSIGNED dict_id'	'dictionary ID'

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.

8.12.4 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`.

8.12.5 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>(...); */
```

8.12.6 Downgrade scenario

If it is necessary to perform *Percona Server for MySQL* downgrade from a version Percona Server for MySQL 5.7.17-11 (or newer) to a version older than Percona Server for MySQL 5.7.17-11 and if user databases have one or more table with compressed columns, there are two options to do this safely:

- Use `mysqldump` in compatible mode (no compressed columns extensions must be specified).
- 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.

8.12.7 Version Specific Information

- Percona Server for MySQL 5.7.17-11: Feature implemented in *Percona Server for MySQL* 5.7

8.12.8 System Variables

`innodb_compressed_columns_zip_level`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	6
Range	`0` - `9`

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`.

`innodb_compressed_columns_threshold`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	96
Range	`1` - `2^64-1` (or `2^32-1` for 32-bit release)

By default a value being inserted will be compressed if its length exceeds 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 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.

8.12.9 Other reading

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

Last update: 2022-09-27

Download PDF

8.13 InnoDB Full-Text Search improvements

8.13.1 Ignoring Stopword list

By default all Full-Text Search indexes check the stopwords list, to see if any indexed elements *contain* one of the words on that list.

Using this list for n-gram indexes isn't always suitable, as an example, any item that contains `a` or `i` will be ignored. Another word that can't be searched is `east`, this one will find no matches because `a` is on the FTS stopword list.

To resolve this issue, in *Percona Server for MySQL* Percona Server 5.7.20-18 a new innodb_ft_ignore_stopwords variable has been implemented which controls whether InnoDB Full-Text Search should ignore stopword list.

Although this variable is introduced to resolve n-gram issues, it affects all Full-Text Search indexes as well.

Being a stopword doesn't just mean to be a one of the predefined words from the list. Tokens shorter than innodb_ft_min_token_size or longer than innodb_ft_max_token_size are also considered stopwords. Therefore, when innodb_ft_ignore_stopwords is set to `ON` even for non-ngram FTS, `innodb_ft_min_token_size` / `innodb_ft_max_token_size` will be ignored meaning that in this case very short and very long words will also be indexed.

System Variables

`innodb_ft_ignore_stopwords`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global, Session
Dynamic	Yes
Data type	Boolean
Default	`OFF`

When enabled, this variable will instruct InnoDB Full Text Search parser to ignore the stopword list when building/updating an FTS index.

8.13.2 Punctuation Marks in Full-Text Search

By default, full text search is unable to find words with various punctuation characters in boolean search mode, although those characters are indexed with ngram parser. A new variable ft_query_extra_word_chars was introduced in *Percona Server for MySQL* Percona Server 5.7.21-20 to solve this issue.

When it's enabled, all the non-whitespace symbols are considered to be word symbols by FTS query parser, except for the boolean search syntax symbols (which are specified by ft_boolean_syntax variable). The latter ones are also considered to be word symbols inside double quotes. This only applies for the query tokenizer, and the indexing tokenizer is not changed in any way. Because of this, the double quote symbol

itself is never considered a word symbol, as no existing indexing tokenizer does so, thus searching for it would never return documents.

System Variables

`ft_query_extra_word_chars`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global, Session
Dynamic	Yes
Data type	Boolean
Default	`OFF`

When enabled, this variable will make all non-whitespace symbols (including punctuation marks) to be treated as word symbols in full-text search queries.

Last update: 2022-09-27

Download PDF

8.14 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.

8.14.1 Temporary tables and mixed logging format

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.

Version Specific Information

- Percona Server for MySQL 5.7.10-1: Fix ported from *Percona Server for MySQL* 5.6

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 replica 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).

8.14.2 Temporary table drops and binloging on GTID-enabled server

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 replicas 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.

Version Specific Information

- Percona Server for MySQL 5.7.17-11: Fix ported from *Percona Server for MySQL* 5.6

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)).

8.14.3 Safety of statements with a `LIMIT` clause

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).

Version Specific Information

- 5.7.10.1: Fix ported from *Percona Server for MySQL* 5.6

8.14.4 Performance improvement on relay log position update

Summary of the fix

MySQL always updated relay log position in multi-source replications setups regardless of whether the committed transaction has already been executed or not. Percona Server omitts relay log position updates for the already logged GTIDs.

Version Specific Information

- Percona Server for MySQL 5.7.18-14: Fix implemented in *Percona Server for MySQL* 5.7

Details

Particularly, such unconditional relay log position updates caused additional fsync operations in case of `relay-log-info-repository=TABLE`, and with the higher number of channels transmitting such duplicate (already executed) transactions the situation became proportionally worse. Bug fixed #1786 (upstream #85141).

8.14.5 Performance improvement on source and connection status updates

Summary of the fix

Replica nodes configured to update source status and connection information only on log file rotation did not experience the expected reduction in load. MySQL was additionaly updating this information in case of multi-source replication when replica had to skip the already executed GTID event.

Version Specific Information

- Percona Server 5.7.20-19: Fix implemented in *Percona Server for MySQL* 5.7

Details

The configuration with `master_info_repository=TABLE` and `sync_master_info=0` makes replica to update source status and connection information in this table on log file rotation and not after each sync_master_info event, but it didn't work on multi-source replication setups. Heartbeats sent to the replica to skip GTID events which it had already executed previously, were evaluated as relay log rotation events and reacted with `mysql.slave_master_info` table sync. This inaccuracy could produce huge (up to 5 times on some setups) increase in write load on the replica, before this problem was fixed in *Percona Server for MySQL*. Bug fixed #1812 (upstream #85158).

8.14.6 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 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 binlog_skip_flush_commands 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 binlog_skip_flush_commands 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 binlog_skip_flush_commands variable.

`binlog_skip_flush_commands`

Introduced in 5.6.43-84.3.

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Default	OFF

When binlog_skip_flush_command 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 binlog_skip_flush_commands variable contains. The `FLUSH` command is not recorded to the binary log and the value of binlog_skip_flush_commands is ignored if the `FLUSH` command is run with the `NO_WRITE_TO_BINLOG` keyword (or its alias `LOCAL`).

8.14.7 Limitations

Do not use one or more dot characters (.) when defining the values for the following variables:

- log_bin
- log_bin_index

MySQL and *XtraBackup* handle the value in different ways and this difference causes unpredictable behavior.

9. Management Improvements

Download PDF

9.1 Percona Toolkit UDFs

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

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

9.1.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

9.1.2 Other Information

- Author / Origin: Baron Schwartz

9.1.3 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.

9.1.4 Troubleshooting

If you get the error:

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

9.1.5 Other Reading

- *Percona Toolkit* documentation

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

9.2 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* Percona Server for MySQL 5.7.17-11 this feature has been re-implemented by setting a connection socket read timeout value instead of periodically scanning the internal InnoDB transaction list.

9.2.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6.
- Percona Server for MySQL 5.7.17-11: Feature re-implemented using socket timeouts.

9.2.2 System Variables

`innodb_kill_idle_transaction`

Option	Description
Config file	`YES`
Scope	`GLOBAL`
Dynamic	`YES`
Data type	`INTEGER`
Default	0 (disabled)
Units	Seconds

Starting with Percona Server for MySQL 5.7.17-11 this variable is an alias of kill_idle_transaction. To enable this feature, set this variable to the desired seconds wait until the transaction is killed.

NOTE: This variable has been deprecated and it will be removed in a future major release.

`kill_idle_transaction`

Option	Description
Config file	`YES`
Scope	`GLOBAL`
Dynamic	`YES`
Data type	`INTEGER`
Default	0 (disabled)
Units	Seconds

The variable has been implemented in Percona Server for MySQL 5.7.17-11. If non-zero, any idle transaction will be killed after being idle for this many seconds.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

9.3 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 enforce_storage_engine with engine that isn't available, system will not start.

Note

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

9.3.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

9.3.2 System Variables

enforce_storage_engine

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	String
Default	NULL

Note

This variable is not case sensitive.

9.3.3 Example

Adding following option to source/glossary.rst`my.cnf` will start the server with InnoDB as enforced storage engine.

```
enforce_storage_engine=InnoDB
```

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

9.4 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 pre-pending the name of the modifier and a dash "-" before the actual configuration option name. For example specifying –maximum-query_cache_size=4M on the mysqld command line or specifying maximum-query_cache_size=4M in the `my.cnf` will prevent any client from setting the query_cache_size value larger than 4MB.

Currently MySQL supports five existing option modifiers:

```
* disable [disable-<option_name>] disables or ignores option_name.

* enable [enable-<option_name>] enables option_name.

* loose [loose-<option_name>] - mysqld will not exit with an error if it does not recognize
option_name, but instead it will issue only a warning.

* maximum [maximum-<option_name>=<value>] indicates that a client can not set the value of
option_name greater than the limit specified. If the client does attempt to set the value of
option_name greater than the limit, the option_name will simply be set to the defined limit.
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 by *Percona Server for MySQL*:

```
* 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.
```

9.4.1 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>
```

9.4.2 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

9.4.3 Examples

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

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

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

Initial query_cache_size size:

```
mysql> SHOW variables LIKE 'query_cache_size';
```

The output should be similar to the following:

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

The output should be similar to the following:

```

Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SHOW variables LIKE 'query_cache_size';
```

The output should be similar to the following:

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

The output should be similar to the following:

```
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
mysql> SHOW warnings;
```

The output should be similar to the following:

```
+---------+------+-----------------------------------------------------+
| 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';
```

The output should be similar to the following:

```
+------------------+---------+
| Variable_name    | Value   |
+------------------+---------+
| query_cache_size | 4194304 |
+------------------+---------+
1 row in set (0.00 sec)
```

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

```
hidden-query_cache_size=1
```

```
mysql> SHOW variables LIKE 'query_cache%';
```

The output should be similar to the following:

```
+------------------------------+---------+
| Variable_name                | Value   |
+------------------------------+---------+
| query_cache_limit            | 1048576 |
| query_cache_min_res_unit     | 4096    |
| query_cache_strip_comments   | OFF     |
| query_cache_type             | ON      |
| query_cache_wlock_invalidate | OFF     |
+------------------------------+---------+
5 rows in set (0.00 sec)
```

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

```
readonly-query_cache_size=1
```

Trying to change the variable value will result in error:

```
mysql> SHOW variables LIKE 'query_cache%';
```

The output should be similar to the following:

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

The output should be similar to the following:

```
ERROR 1238 (HY000): Variable 'query_cache_size' is a read only variable
```

Download PDF

9.5 XtraDB changed page tracking

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

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

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

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

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

9.5.1 User statements for handling the XtraDB changed page bitmaps

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.

9.5.2 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.

9.5.3 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.

`INFORMATION_SCHEMA.INNODB_CHANGED_PAGES`

Column Name	Description
'INT(11) space_id'	'space id of modified page'
'INT(11) page_id'	'id of modified page'
'BIGINT(21) start_lsn'	'start of the interval'
'BIGINT(21) end_lsn'	'end of the interval '

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

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

9.5.4 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

9.5.5 System Variables

`innodb_max_changed_pages`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	1000000
Range	1 - 0 (unlimited)

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

`innodb_track_changed_pages`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Boolean
Default	0 - False
Range	0-1

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

`innodb_max_bitmap_file_size`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	104857600 (100 MB)
Range	4096 (4KB) - 18446744073709551615 (16EB)

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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

9.6 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, expand_fast_index_creation may also optimize index access for subsequent DML statements because using it results in much less fragmented indexes.

9.6.1 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.

9.6.2 `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.

9.6.3 `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`.

9.6.4 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 expand_fast_index_creation. It is a session variable, so you can temporarily switch it off if you are short on tmpdir space and/or don't want this optimization to be used for a specific table.

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

```
* `UNIQUE` indexes in `ALTER TABLE` are ignored to enforce uniqueness where necessary when
copying the data to a temporary table;
```

```
* `ALTER TABLE` and `OPTIMIZE TABLE` always process tables containing foreign keys as if
expand_fast_index_creation is OFF to avoid dropping keys that are part of a FOREIGN KEY
constraint;


* **mysqldump --innodb-optimize-keys** ignores foreign keys because InnoDB requires a full
table rebuild on foreign key changes. So adding them back with a separate `ALTER TABLE`
after restoring the data from a dump would actually make the restore slower;


* **mysqldump --innodb-optimize-keys** ignores indexes on `AUTO_INCREMENT` columns, because
they must be indexed, so it is impossible to temporarily drop the corresponding index;


* **mysqldump --innodb-optimize-keys** ignores the first UNIQUE index on non-nullable
columns when the table has no `PRIMARY KEY` defined, because in this case InnoDB picks such
an index as the clustered one.
```

Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

System Variables

expand_fast_index_creation

Option	Description
Command-line	Yes
Config file	No
Scope	Local/Global
Dynamic	Yes
Data type	Boolean
Default	ON/OFF

Other Reading

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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

9.7 Backup Locks

Percona Server for MySQL has implemented this feature 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`.

9.7.1 `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.
```

9.7.2 `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. source binary log coordinates corresponding to the current SQL thread state on a replication replica) 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 replica 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`.

`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.

9.7.3 `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
```

9.7.4 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`.

9.7.5 Interaction with other global locks

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

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

9.7.6 MyISAM index and data buffering

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

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

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

MyISAM may also cache data for bulk inserts, e.g. when executing multi-row 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.

9.7.7 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.

Percona XtraBackup provides the –backup-locks option. If you disable this option, `Flush Table with Read Lock` is used on the backup stage.

Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

System Variables

`have_backup_locks`

Option	Description
Command-line	Yes
Config file	No
Scope	Global
Dynamic	No
Data type	Boolean
Default	YES

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

`have_backup_safe_binlog_info`

Option	Description
Command-line	Yes
Config file	No
Scope	Global
Dynamic	No
Data type	Boolean
Default	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`.

Status Variables

`Com_lock_tables_for_backup`

Option	Description
Scope	Global/Session
Data type	Numeric

`Com_lock_binlog_for_backup`

Option	Description
Scope	Global/Session
Data type	Numeric

`Com_unlock_binlog`

Option	Description
Scope	Global/Session
Data type	Numeric

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

Client Command Line Parameter

`lock-for-backup`

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	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.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

9.8 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.

Audit logging documents the database usage. You can use the log for troubleshooting.

9.8.1 Installation

Audit Log plugin is installed with *Percona Server for MySQL*, but is not enabled by default. You can verify if the plugin is enabled by running the following commands:

```
mysql> SELECT * FROM information_schema.PLUGINS WHERE PLUGIN_NAME LIKE '%audit%';
```

The output should be similar to the following;

```
Empty set (0.00 sec)
```

```
mysql> SHOW variables LIKE 'audit%';
```

The output should be similar to the following;

```
Empty set (0.01 sec)
```

```
mysql> SHOW variables LIKE 'plugin%';
```

The output should be similar to the following;

```
+---------------+-------------------------+
| Variable_name | Value                   |
+---------------+-------------------------+
| plugin_dir    | /usr/lib/mysql/plugin/ |
+---------------+-------------------------+
1 row in set (0.00 sec)
```

Note

The location of the MySQL plugin directory depends on the operating system and may be different.

The following command enables the plugin:

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

Run the following command to verify if the plugin was installed correctly:

```
mysql> SELECT * FROM information_schema.PLUGINS WHERE PLUGIN_NAME LIKE '%audit%'\G
```

The output should be similar to the following;

```
*************************** 1. row ***************************
           PLUGIN_NAME: audit_log
        PLUGIN_VERSION: 0.2
        PLUGIN_STATUS: ACTIVE
           PLUGIN_TYPE: AUDIT
  PLUGIN_TYPE_VERSION: 4.1
        PLUGIN_LIBRARY: audit_log.so
PLUGIN_LIBRARY_VERSION: 1.7
        PLUGIN_AUTHOR: Percona LLC and/or its affiliates.
    PLUGIN_DESCRIPTION: Audit log
        PLUGIN_LICENSE: GPL
           LOAD_OPTION: ON
1 row in set (0.00 sec)
```

You can review the audit log variables with the following command:

```
mysql> SHOW variables LIKE 'audit%';
```

The output should be similar to the following;

```
+-------------------------------+---------------+
| Variable_name                 | Value         |
+-------------------------------+---------------+
| audit_log_buffer_size         | 1048576       |
| audit_log_exclude_accounts    |               |
| audit_log_exclude_commands    |               |
| audit_log_exclude_databases   |               |
| audit_log_file                | audit.log     |
| audit_log_flush               | OFF           |
| audit_log_format              | OLD           |
| audit_log_handler             | FILE          |
| audit_log_include_accounts    |               |
| audit_log_include_commands    |               |
| audit_log_include_databases   |               |
| audit_log_policy              | ALL           |
| audit_log_rotate_on_size      | 0             |
| audit_log_rotations           | 0             |
| audit_log_strategy            | ASYNCHRONOUS  |
| audit_log_syslog_facility     | LOG_USER      |
| audit_log_syslog_ident        | percona-audit |
| audit_log_syslog_priority     | LOG_INFO      |
+-------------------------------+---------------+
18 rows in set (0.00 sec)
```

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_2021-06-30T11:56:53"
  TIMESTAMP="2021-06-30T11:56:53 UTC"
  MYSQL_VERSION="5.7.34-37"
```

```
  STARTUP_OPTIONS="--daemonize --pid-file=/var/run/mysqld/mysqld.pid"
  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 the following:

- `CONNECTION_ID`
- `STATUS`
- `USER`
- `PRIV_USER`
- `OS_LOGIN`
- `PROXY_USER`
- `HOST`
- `IP`

The value for `STATUS` is `0` for successful logins and non-zero for failed logins.

Example of the Disconnect event:

```
<AUDIT_RECORD
  NAME="Quit"
  RECORD="5_2021-06-29T19:33:03"
  TIMESTAMP="2021-06-29T19:34:38Z"
  CONNECTION_ID="14"
  STATUS="0"
  USER="root"
  PRIV_USER="root"
  OS_LOGIN=""
  PROXY_USER=""
  HOST="localhost"
  IP=""
  DB=""
/>
```

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

Note

The `statement/sql/%` populates the audit log command_class field, for example, the `SELECT name FROM performance_schema.setup_instruments WHERE name LIKE "statement/sql/%"` query.

The `%statement/com%` entry populates the audit log command_class field as lowercase text, for example, the `SELECT name FROM performance_schema.setup_instruments WHERE name LIKE '%statement/com%'` query. If you run a 'Ping' command, then the command_class field is 'ping', and for 'Init DB', the command_class field is 'init db'.

Example of the Query event:

```
<AUDIT_RECORD
  NAME="Query"
  RECORD="4_2021-06-29T19:33:03"
  TIMESTAMP="2021-06-29T19:33:34Z"
  COMMAND_CLASS="show_variables"
  CONNECTION_ID="14"
  STATUS="0"
  SQLTEXT="show variables like 'audit%'"
  USER="root[root] @ localhost []"
  HOST="localhost"
  OS_USER=""
  IP=""
  DB=""
/>
```

9.8.2 Log Format

The audit log plugin supports the following log formats: `OLD`, `NEW`, `JSON`, and `CSV`. The `OLD\`format and the``NEW `format are based on XML. The OLD format defines each log record with XML attributes. The NEW` format defines each log record with XML tags. The information logged is the same for all four formats. The audit_log_format variable controls the log format choice.

An example of the `OLD` format:

```
<AUDIT_RECORD
  NAME="Query"
  RECORD="3_2021-06-30T11:56:53"
  TIMESTAMP="2021-06-30T11:57:14 UTC"
  COMMAND_CLASS="select"
  CONNECTION_ID="3"
  STATUS="0"
  SQLTEXT="select * from information_schema.PLUGINS where PLUGIN_NAME like '%audit%'"
  USER="root[root] @ localhost []"
  HOST="localhost"
  OS_USER=""
  IP=""
  DB=""
/>
```

An example of the `NEW` format:

```
<AUDIT_RECORD>
  <NAME>Query</NAME>
  <RECORD>16684_2021-06-30T16:07:41</RECORD>
  <TIMESTAMP>2021-06-30T16:08:06 UTC</TIMESTAMP>
  <COMMAND_CLASS>select</COMMAND_CLASS>
  <CONNECTION_ID>2</CONNECTION_ID>
  <STATUS>0</STATUS>
  <SQLTEXT>select id, holder from one</SQLTEXT>
  <USER>root[root] @ localhost []</USER>
  <HOST>localhost</HOST>
  <OS_USER></OS_USER>
  <IP></IP>
  <DB></DB>
```

An example of the `JSON` format:

```
{"audit_record":
{"name":"Query","record":"13149_2021-06-30T15:03:11","timestamp":"2021-06-30T15:07:58
UTC","command_class":"show_databases","connection_id":"2","status":0,"sqltext":"show
databases","user":"root[root] @ localhost
[]","host":"localhost","os_user":"","ip":"","db":""}}
```

An example of the `CSV` format:

```
"Query","22567_2021-06-30T16:10:09","2021-06-30T16:19:00 UTC","select","2",0,"select
count(*) from one","root[root] @ localhost []","localhost","","",""
```

9.8.3 Streaming the audit log to syslog

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

Note

The actions for the variables: audit_log_strategy, audit_log_buffer_size, audit_log_rotate_on_size, audit_log_rotations are captured only with `FILE` handler.

9.8.4 Filtering by user

In Percona Server for MySQL 5.7.14-7 *Percona Server for MySQL* has implemented filtering by user. This was implemented by adding two new global variables: audit_log_include_accounts and 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 audit_log_include_accounts and audit_log_exclude_accounts do not apply to existing server connections.

Example

Following example shows adding users who will be monitored:

```
mysql> SET GLOBAL audit_log_include_accounts = 'user1@localhost,root@localhost';
```

The output should be similar to the following;

```
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';
```

The output should be similar to the following;

```
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;
```

The output should be similar to the following;

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SET GLOBAL audit_log_exclude_accounts = 'user1@localhost,root@localhost';
```

The output should be similar to the following;

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SET GLOBAL audit_log_exclude_accounts = "'user'@'host'";
```

The output should be similar to the following;

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SET GLOBAL audit_log_exclude_accounts = '''user''@''host''';
```

The output should be similar to the following;

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SET GLOBAL audit_log_exclude_accounts = '\'user\'@\'host\'';
```

The output should be similar to the following;

```
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;
```

The output should be similar to the following;

```
+-----------------------------+
| @@audit_log_exclude_accounts |
+-----------------------------+
| 'user'@'host'               |
+-----------------------------+
1 row in set (0.00 sec)
```

Account names from mysql.user table are the one that are logged in the audit log. For example when you create a user:

```
mysql> CREATE USER 'user1'@'%' IDENTIFIED BY '111';
```

The output should be similar to the following;

```
Query OK, 0 rows affected (0.00 sec)
```

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

```
<AUDIT_RECORD
  NAME="Connect"
  RECORD="2_2021-06-30T11:56:53"
  TIMESTAMP="2021-06-30T11:56:53 UTC"
  CONNECTION_ID="6"
  STATUS="0"
  USER="user1" ;; this is a 'user' part of account in 5.7
  PRIV_USER="user1"
  OS_LOGIN=""
  PROXY_USER=""
  HOST="localhost" ;; this is a 'host' part of account in 5.7
  IP=""
  DB=""
/>
```

To exclude `user1` from logging in *Percona Server for MySQL* 5.7 you must set:

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

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

9.8.5 Filtering by SQL command type

In Percona Server for MySQL 5.7.14-7 *Percona Server for MySQL* has implemented filtering by SQL command type. This was implemented by adding two new global variables: audit_log_include_commands and 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 audit_log_exclude_commands and audit_log_include_commands are `NULL` all commands will be logged.

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;
```

The output should be similar to the following;

```
+----------------------------------------+
| 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';
```

When you create a database with the following command:

```
mysql> CREATE DATABASE sample;
```

The action is captured in the audit log:

```
<AUDIT_RECORD>
  <NAME>Query</NAME>
  <RECORD>24320_2021-06-30T17:44:46</RECORD>
  <TIMESTAMP>2021-06-30T17:45:16 UTC</TIMESTAMP>
  <COMMAND_CLASS>create_db</COMMAND_CLASS>
  <CONNECTION_ID>2</CONNECTION_ID>
  <STATUS>0</STATUS>
  <SQLTEXT>CREATE DATABASE sample</SQLTEXT>
```

```
  <USER>root[root] @ localhost []</USER>
  <HOST>localhost</HOST>
  <OS_USER></OS_USER>
  <IP></IP>
  <DB></DB>
</AUDIT_RECORD>
```

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;
```

The output should be similar to the following;

```
Query OK, 0 rows affected (0.00 sec)
```

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

The output should be similar to the following;

```
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.

9.8.6 Filtering by database

In Percona Server for MySQL 5.7.14-7 *Percona Server for MySQL* has implemented filtering by SQL database. This was implemented by adding two new global variables: audit_log_include_databases and audit_log_exclude_databases to specify which databases should be included or excluded from audit logging.

Warning

Only one of these variables can contain a list of databases 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 databases), the attempt to set another one will fail. Empty string means an empty list.

If query is accessing any of databases listed in `audit_log_include_databases`, the query will be logged. If query is accessing only databases listed in `audit_log_exclude_databases`, the query will not be logged. `CREATE TABLE` statements are logged unconditionally.

Note

Changes of audit_log_include_databases and audit_log_exclude_databases do not apply to existing server connections.

Example

To add databases to be monitored you should run:

```
mysql> SET GLOBAL audit_log_include_databases = 'test,mysql,db1';
```

The output should be similar to the following;

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SET GLOBAL audit_log_include_databases= 'db1,db3';
```

The output should be similar to the following;

```
Query OK, 0 rows affected (0.00 sec)
```

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

```
mysql> SET GLOBAL audit_log_exclude_databases = 'test,mysql,db1';
```

The output should be similar to the following;

```
ERROR 1231 (42000): Variable 'audit_log_exclude_databases can't be set to the value of
'test,mysql,db1'
```

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

```
mysql> SET GLOBAL audit_log_include_databases = NULL;
```

The output should be similar to the following;

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SET GLOBAL audit_log_exclude_databases = 'test,mysql,db1';
```

The output should be similar to the following;

```
Query OK, 0 rows affected (0.00 sec)
```

9.8.7 System Variables

audit_log_strategy

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	String
Default	ASYNCHRONOUS
Allowed values	ASYNCHRONOUS, PERFORMANCE, SEMISYNCHRONOUS, SYNCHRONOUS

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

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

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

audit_log_file

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	String
Default	audit.log

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

audit_log_flush

Option	Description
Command-line	Yes
Scope	Global
Dynamic	Yes
Data type	String
Default	OFF

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

audit_log_buffer_size

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	Numeric
Default	1 Mb

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

audit_log_exclude_accounts

Option	Description
Command-line	Yes
Scope	Global
Dynamic	Yes
Data type	String

The variable has been implemented in Percona Server for MySQL 5.7.14-7. 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 audit_log_include_accounts must be unset, and vice versa.

audit_log_exclude_commands

Option	Description
Command-line	Yes
Scope	Global
Dynamic	Yes
Data type	String

The variable has been implemented in Percona Server for MySQL 5.7.14-7. 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 audit_log_include_commands must be unset, and vice versa.

audit_log_exclude_databases

Option	Description
Command-line	Yes
Scope	Global
Dynamic	Yes
Data type	String

The variable has been implemented in Percona Server for MySQL 5.7.14-7. This variable is used to specify the list of commands for which Filtering by database is applied. The value can be `NULL` or comma separated list of commands. If this variable is set, then audit_log_include_databases must be unset, and vice versa.

audit_log_format

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	String
Default	OLD
Allowed values	OLD, NEW, CSV, JSON

Implemented in Percona Server for MySQL 5.7.14-7.

audit_log_include_accounts

Option	Description
Command-line	Yes
Scope	Global
Dynamic	Yes
Data type	String

The variable has been implemented in Percona Server for MySQL 5.7.14-7. 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 audit_log_exclude_accounts must be unset, and vice versa.

audit_log_include_commands

Option	Description
Command-line	Yes
Scope	Global
Dynamic	Yes
Data type	String

The variable has been implemented in Percona Server for MySQL 5.7.14-7. 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 audit_log_exclude_commands must be unset, and vice versa.

audit_log_include_databases

Option	Description
Command-line	Yes
Scope	Global
Dynamic	Yes
Data type	String

The variable has been implemented in Percona Server for MySQL 5.7.14-7. This variable is used to specify the list of commands for which Filtering by database is applied. The value can be `NULL` or comma separated list of commands. If this variable is set, then audit_log_exclude_databases must be unset, and vice versa.

audit_log_policy

Option	Description
Command-line	Yes
Scope	Global
Dynamic	Yes
Data type	String
Default	ALL
Allowed values	ALL, LOGINS, QUERIES, NONE

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

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

audit_log_rotate_on_size

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	Numeric
Default	0 (don't rotate the log file)

This variable is measured in bytes and 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 audit_log_handler variable to `FILE` and the audit_log_rotations variable to a value greater than zero.

audit_log_rotations

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	Numeric
Default	0

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

audit_log_handler

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	String
Default	FILE
Allowed values	FILE, SYSLOG

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

audit_log_syslog_ident

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	String
Default	percona-audit

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

audit_log_syslog_facility

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	String
Default	LOG_USER

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

audit_log_syslog_priority

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	String
Default	LOG_INFO

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

9.8.8 Status Variables

audit_log_buffer_size_overflow

Option	Description
Scope	Global
Data type	Numeric

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

9.8.9 Version Specific Information

- Percona Server for MySQL 5.7.10-1 Feature ported from *Percona Server for MySQL* 5.6
- Percona Server for MySQL 5.7.14-7 *Percona Server for MySQL* Audit Log Plugin now supports filtering by user, sql_command, and databases.
- Percona Server for MySQL 5.7.26-29 Audit_log_buffer_size_overflow variable implemented

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

9.9 Start transaction with consistent snapshot

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

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

9.9.1 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.

9.9.2 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.

9.9.3 System Variables

`have_snapshot_cloning`

Option	Description
Command-line	Yes
Config file	No
Scope	Global
Dynamic	No
Data type	Boolean

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

9.9.4 Status Variables

`Binlog_snapshot_file`

Option	Description
Scope	Global
Data type	String

`Binlog_snapshot_position`

Option	Description
Scope	Global
Data type	Numeric

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

9.9.5 Other Reading

- MariaDB Enhancements for START TRANSACTION WITH CONSISTENT SNAPSHOT

Download PDF

9.10 Extended SHOW GRANTS

In Oracle MySQL `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 *Percona Server for MySQL* `SHOW GRANTS` command was extended to display all the effectively available privileges to the account.

9.10.1 Example

If we create the following users:

```
mysql> CREATE USER grantee@localhost IDENTIFIED BY 'grantee1';
```

The output should be similar to the following:

```
Query OK, 0 rows affected (0.50 sec)
```

```
mysql> CREATE USER grantee IDENTIFIED BY 'grantee2';
```

The output should be similar to the following:

```
Query OK, 0 rows affected (0.09 sec)
```

```
mysql> CREATE DATABASE db2;
```

The output should be similar to the following:

```
Query OK, 1 row affected (0.20 sec)
```

```
mysql> GRANT ALL PRIVILEGES ON db2.* TO grantee WITH GRANT OPTION;
```

The output should be similar to the following:

```
Query OK, 0 rows affected (0.12 sec)
```

- `SHOW GRANTS` output before the change:

```
mysql> SHOW GRANTS;
```

The output should be similar to the following:

```
+-----------------------------------------------------------------------------------------------
+
| 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);
```

The output should be similar to the following:

```
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;
```

The output should be similar to the following:

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

Version-Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

Other reading

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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

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

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

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

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

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

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

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

Example: `--utility_user_password` =`Passw0rD`

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

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

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

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

9.11.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

9.11.2 System Variables

`utility_user`

Option	Description
Command-line	Yes
Config file	utility_user=user@host
Scope	Global
Dynamic	No
Data type	String
Default	NULL

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

`utility_user_password`

Option	Description
Command-line	Yes
Config file	utility_user_password=
Scope	Global
Dynamic	No
Data type	String
Default	NULL

Specifies the password required for the utility user.

`utility_user_schema_access`

Option	Description
Command-line	Yes
Config file	utility_user_schema_access=„
Scope	Global
Dynamic	No
Data type	String
Default	NULL

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

`utility_user_privileges`

Option	Description
Command-line	Yes
Config file	utility_user_privileges=„
Scope	Global
Dynamic	No
Data type	String
Default	NULL

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

Last update: 2022-09-27

Download PDF

9.12 PS-Admin script

You can use the ps-admin script allows Enabling the TokuDB Storage Engine and Percona TokuBackup. If the TokuDB storage engine enables the transparent huge pages, the script adds the thp-setting=never option to my.cnf to disable transparent huge pages on runtime.

An example of enabling the TokuDB plugin follows:

```
$ sudo ps-admin --enable-tokudb -u root -pPassw0rd
```

The following example enables the TokuBackup.

```
$ sudo ps-admin --enable-tokubackup
```

You are able to Enable MyRocks with ps-admin and disable and uninstall the MyRocks storage engine.

An example of the enabling and disabling the MyRocksDB plugin follows:

```
$ sudo ps-admin --enable-rocksdb -u root -pPassw0rd
```

```
$ sudo ps-admin --disable-rocksdb -u root -pPassw0rd
```

The ps-admin script can also enable or disable the following:

- Audit Log plugin
- PAM Authentication Plugin
- Query Reponse Time plugin
- MYSQLX plugin

An example of enabling the PAM Authentication plugin follows:

```
$ sudo ps-admin --enable-pam -u root -pPassw0rd
```


Last update: 2022-09-27

10. Percona MyRocks

Download PDF

10.1 Percona MyRocks Introduction

MyRocks is a storage engine for MySQL based on RocksDB, an embeddable, persistent key-value store. *Percona MyRocks* is an implementation for Percona Server.

The RocksDB store is based on the log-structured merge-tree (or LSM tree). It is optimized for fast storage and combines outstanding space and write efficiency with acceptable read performance. As a result, MyRocks has the following advantages compared to other storage engines, if your workload uses fast storage, such as SSD:

- Requires less storage space
- Provides more storage endurance
- Ensures better IO capacity
- Percona MyRocks Installation Guide
- MyRocks Limitations
- Differences between Percona MyRocks and Facebook MyRocks
- MyRocks Server Variables
- MyRocks Information Schema Tables

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

10.2 Percona MyRocks Installation Guide

Percona MyRocks is distributed as a separate package that can be enabled as a plugin for Percona Server 5.7 and later versions.

Note

File formats across different MyRocks variants may not be compatible. Percona Server supports only *Percona MyRocks*. Migrating from one variant to another requires a logical data dump and reload.

10.2.1 Installing Percona MyRocks

It is recommended to install Percona software from official repositories:

1. Configure Percona repositories as described in Percona Software Repositories Documentation.
2. Install Percona MyRocks using the corresponding package manager:
   - For Debian or Ubuntu:

```
shell
$ sudo apt install percona-server-rocksdb-5.7
```

   - For RHEL or CentOS:

```
shell
$ sudo yum install Percona-Server-rocksdb-57.x86_64
```

After you install the Percona MyRocks package, you should see the following output:

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

       ps-admin --enable-rocksdb -u <mysql_admin_user> -p[mysql_admin_pass] [-S <socket>] [-
h <host> -P <port>]
```

Enable MyRocks with ps-admin

Run the `ps-admin` script as system root user or with **sudo** and provide the MySQL root user credentials to properly enable the RocksDB (MyRocks) storage engine:

```
$ sudo ps-admin --enable-rocksdb -u root -pPassw0rd
```

You should see the following output:

```
Checking if RocksDB plugin is available for installation ...
INFO: ha_rocksdb.so library for RocksDB found at /usr/lib64/mysql/plugin/ha_rocksdb.so.

Checking RocksDB engine plugin status...
INFO: RocksDB engine plugin is not installed.

Installing RocksDB engine...
INFO: Successfully installed RocksDB engine plugin.
```

Note

Running the `ps-admin` script to enable Percona MyRocks also installs and enables the RocksDB plugin.

If the script returns no errors, Percona MyRocks should be successfully enabled on the server. You can verify it as follows:

```
mysql> SHOW ENGINES;
```

The output could be the following:

```
+---------+---------
+-------------------------------------------------------------------------------+--------------
+------+------------+
| Engine  | Support |
Comment                                                                         | Transactions |
XA   | Savepoints |
+---------+---------
+-------------------------------------------------------------------------------+--------------
+------+------------+
| ROCKSDB | YES     | RocksDB storage
engine                                                  | YES          | YES  |
YES        |
...
| InnoDB  | DEFAULT | Percona-XtraDB, Supports transactions, row-level locking, and foreign
keys | YES          | YES  | YES        |
+---------+---------
+-------------------------------------------------------------------------------+--------------
+------+------------+
10 rows in set (0.00 sec)
```

Note that the RocksDB engine is not set to be default, new tables will still be created using the InnoDB (XtraDB) storage engine. To make RocksDB storage engine default, set `default-storage-engine=rocksdb` in the `[mysqld]` section of `my.cnf` and restart Percona Server.

Alternatively, you can add `ENGINE=RocksDB` after the `CREATE TABLE` statement for every table that you create.

Installing MyRocks Plugins

You can install MyRocks manually with a series of INSTALL PLUGIN statements. You must have the `INSERT` privilege for the `mysql.plugin` system table.

The following statements install MyRocks:

```
INSTALL PLUGIN ROCKSDB SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_CFSTATS SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_DBSTATS SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_PERF_CONTEXT SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_PERF_CONTEXT_GLOBAL SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_CF_OPTIONS SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_GLOBAL_INFO SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_COMPACTION_STATS SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_DDL SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_INDEX_FILE_MAP SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_LOCKS SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_TRX SONAME 'ha_rocksdb.so';
INSTALL PLUGIN ROCKSDB_DEADLOCK SONAME 'ha_rocksdb.so';
```

10.2.2 Removing Percona MyRocks

It will not be possible to access tables created using the RocksDB engine with another storage engine after you remove Percona MyRocks. If you need this data, alter the tables to another storage engine. For example, to alter the `City` table to InnoDB, run the following:

```
mysql> ALTER TABLE City ENGINE=InnoDB;
```

To disable and uninstall the RocksDB engine plugins, use the `ps-admin` script as follows:

```
$ sudo ps-admin --disable-rocksdb -u root -pPassw0rd
```

You should see the following output:

```
Checking RocksDB engine plugin status...
INFO: RocksDB engine plugin is installed.

Uninstalling RocksDB engine plugin...
INFO: Successfully uninstalled RocksDB engine plugin.
```

After the engine plugins have been uninstalled, remove the Percona MyRocks package:

- For Debian or Ubuntu:

```
$ sudo apt remove percona-server-rocksdb-5.7
```

- For RHEL or CentOS:

```
$ sudo yum remove Percona-Server-rocksdb-57.x86_64
```

Finally, remove all the MyRocks Server Variables from the configuration file (`my.cnf`) and restart Percona Server.

Uninstall MyRocks Plugins

You can uninstall the plugins for MyRocks. You must have the `DELETE` privilege for the `mysql.plugin` system table.

The following statements remove the MyRocks plugins:

```
UNINSTALL PLUGIN ROCKSDB;
UNINSTALL PLUGIN ROCKSDB_CFSTATS;
UNINSTALL PLUGIN ROCKSDB_DBSTATS;
UNINSTALL PLUGIN ROCKSDB_PERF_CONTEXT;
UNINSTALL PLUGIN ROCKSDB_PERF_CONTEXT_GLOBAL;
UNINSTALL PLUGIN ROCKSDB_CF_OPTIONS;
UNINSTALL PLUGIN ROCKSDB_GLOBAL_INFO;
UNINSTALL PLUGIN ROCKSDB_COMPACTION_STATS;
UNINSTALL PLUGIN ROCKSDB_DDL;
UNINSTALL PLUGIN ROCKSDB_INDEX_FILE_MAP;
UNINSTALL PLUGIN ROCKSDB_LOCKS;
UNINSTALL PLUGIN ROCKSDB_TRX;
UNINSTALL PLUGIN ROCKSDB_DEADLOCK;
```

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

10.3 MyRocks Limitations

The MyRocks storage engine lacks the following features compared to InnoDB:

- Online DDL
- ALTER TABLE ... EXCHANGE PARTITION
- SAVEPOINT
- Transportable tablespace
- Foreign keys
- Spatial indexes
- Fulltext indexes
- Gap locks
- Group Replication
- Generated Columns
- Partial Update of LOB in InnoDB

You should also consider the following:

- `\*_bin` (e.g. `latin1_bin`) or binary collation should be used on `CHAR` and `VARCHAR` indexed columns. The following binary collations are supported: `binary`, `latin1_bin`, and `utf8_bin`. By default, MyRocks prevents creating indexes with non-binary collations (including `latin1`). You can optionally use it by setting rocksdb_strict_collation_exceptions to `t1` (table names with regex format), but non-binary covering indexes other than `latin1` (excluding `german1`) still require a primary key lookup to return the `CHAR` or `VARCHAR` column.
- Either `ORDER BY DESC` or `ORDER BY ASC` is slow. This is because of "Prefix Key Encoding" feature in RocksDB. See http://www.slideshare.net/matsunobu/myrocks-deep-dive/58 for details. By default, ascending scan is faster and descending scan is slower. If the "reverse column family" is configured, then descending scan will be faster and ascending scan will be slower. Note that InnoDB also imposes a cost when the index is scanned in the opposite order.
- MyRocks does not support operating as either a source or a replica in any replication topology that is not exclusively row-based. Statement-based and mixed-format binary logging is not supported. For more information, see Replication Formats.
- When converting from large MyISAM/InnoDB tables, either by using the `ALTER` or `INSERT INTO SELECT` statements it's recommended that you check the Data loading documentation and create MyRocks tables as below (in case the table is sufficiently big it will cause the server to consume all the memory and then be terminated by the OOM killer):

```
SET session sql_log_bin=0;
SET session rocksdb_bulk_load=1;
ALTER TABLE large_myisam_table ENGINE=RocksDB;
SET session rocksdb_bulk_load=0;
```

You should see the following output:

```
.. warning::

 If you are loading large data without enabling :ref:`rocksdb_bulk_load`
 or :ref:`rocksdb_commit_in_the_middle`, please make sure transaction
```

```
size is small enough. All modifications of the ongoing transactions are
kept in memory.
```

- The`XA protocol <https://dev.mysql.com/doc/refman/5.7/en/xa.html>`_ support, which allows distributed transactions combining multiple separate transactional resources, is an experimental feature in MyRocks: the implementation is less tested, it may lack some functionality and be not as stable as in case of InnoDB.
- MySQL has spatial data types . These data types are not supported by MyRocks.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

10.4 Differences between Percona MyRocks and Facebook MyRocks

The original MyRocks was developed by Facebook and works with their implementation of MySQL. *Percona MyRocks* is a branch of MyRocks for Percona Server and includes the following differences from the original implementation:

- The behavior of the `START TRANSACTION WITH CONSISTENT SNAPSHOT` statement depends on the transaction isolation level.

Storage Engine	Transaction isolation level	
	`READ COMMITTED`	`REPEATABLE READ`
InnoDB	Success	Success
Facebook MyRocks	Fail	Success (MyRocks engine only; read-only, as all MyRocks engine snapshots)
Percona MyRocks	Fail with any DML which would violate the read-only snapshot constraint	Success (read-only snapshots independent of the engines in use)

- Percona MyRocks includes the `lz4` and `zstd` statically linked libraries.

Compression

The supported compression algorithms are the following:

Compression Algorithm	Percona MyRocks	Facebook MyRocks
Zlib	Yes	Yes
LZ4	Yes	Yes
ZStd	Yes	Yes
None	Yes	Yes
Snappy	No	Yes
Bzip	No	Yes

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

10.5 MyRocks Server Variables

The MyRocks server variables expose configuration of the underlying RocksDB engine. There several ways to set these variables:

- For production deployments, you should have all variables defined in the configuration file.
- *Dynamic* variables can be changed at runtime using the `SET` statement.
- If you want to test things out, you can set some of the variables when starting `mysqld` using corresponding command-line options.

If a variable was not set in either the configuration file or as a command-line option, the default value is used.

Also, all variables can exist in one or both of the following scopes:

- *Global* scope defines how the variable affects overall server operation.
- *Session* scope defines how the variable affects operation for individual client connections.

The following server variables are available:

Variable Name
rocksdb_access_hint_on_compaction_start
rocksdb_advise_random_on_open
rocksdb_allow_concurrent_memtable_write
rocksdb_allow_to_start_after_corruption
rocksdb_allow_mmap_reads
rocksdb_allow_mmap_writes
rocksdb_alter_column_default_inplace
rocksdb_base_background_compactions
rocksdb_blind_delete_primary_key
rocksdb_block_cache_size
rocksdb_block_restart_interval
rocksdb_block_size
rocksdb_block_size_deviation
rocksdb_bulk_load
rocksdb_bulk_load_allow_sk
rocksdb_bulk_load_allow_unsorted
rocksdb_bulk_load_size
rocksdb_bytes_per_sync
rocksdb_cache_dump
rocksdb_cache_index_and_filter_blocks
rocksdb_checksums_pct
rocksdb_collect_sst_properties
rocksdb_commit_in_the_middle
rocksdb_commit_time_batch_for_recovery
rocksdb_compact_cf
rocksdb_compaction_readahead_size
rocksdb_compaction_sequential_deletes
rocksdb_compaction_sequential_deletes_count_sd
rocksdb_compaction_sequential_deletes_file_size
rocksdb_compaction_sequential_deletes_window
rocksdb_concurrent_prepare
rocksdb_create_checkpoint
rocksdb_create_if_missing
rocksdb_create_missing_column_families
rocksdb_datadir

Variable Name
rocksdb_db_write_buffer_size
rocksdb_deadlock_detect
rocksdb_deadlock_detect_depth
rocksdb_debug_optimizer_no_zero_cardinality
rocksdb_debug_ttl_ignore_pk
rocksdb_debug_ttl_read_filter_ts
rocksdb_debug_ttl_rec_ts
rocksdb_debug_ttl_snapshot_ts
rocksdb_default_cf_options
rocksdb_delayed_write_rate
rocksdb_delete_cf
rocksdb_delete_obsolete_files_period_micros
rocksdb_disable_file_deletions
rocksdb_enable_bulk_load_api
rocksdb_enable_insert_with_update_caching
rocksdb_enable_iterate_bounds
rocksdb_enable_native_partition
rocksdb_enable_pipelined_write
rocksdb_enable_remove_orphaned_dropped_cfs
rocksdb_enable_ttl
rocksdb_enable_ttl_read_filtering
rocksdb_enable_thread_tracking
rocksdb_enable_write_thread_adaptive_yield
rocksdb_error_if_exists
rocksdb_error_on_suboptimal_collation
rocksdb_flush_log_at_trx_commit
rocksdb_flush_memtable_on_analyze
rocksdb_force_compute_memtable_stats
rocksdb_force_compute_memtable_stats_cachetime
rocksdb_force_flush_memtable_and_lzero_now
rocksdb_force_flush_memtable_now
rocksdb_force_index_records_in_range
rocksdb_hash_index_allow_collision
rocksdb_ignore_unknown_options
rocksdb_index_type

Variable Name
rocksdb_info_log_level
rocksdb_is_fd_close_on_exec
rocksdb_keep_log_file_num
rocksdb_large_prefix
rocksdb_lock_scanned_rows
rocksdb_lock_wait_timeout
rocksdb_log_file_time_to_roll
rocksdb_manifest_preallocation_size
rocksdb_manual_compaction_bottommost_level
rocksdb_manual_wal_flush
rocksdb_master_skip_tx_api
rocksdb_max_background_compactions
rocksdb_max_background_flushes
rocksdb_max_background_jobs
rocksdb_max_bottom_pri_background_compactions
rocksdb_max_latest_deadlocks
rocksdb_max_log_file_size
rocksdb_max_manifest_file_size
rocksdb_max_open_files
rocksdb_max_row_locks
rocksdb_max_subcompactions
rocksdb_max_total_wal_size
rocksdb_merge_buf_size
rocksdb_merge_combine_read_size
rocksdb_merge_tmp_file_removal_delay_ms
rocksdb_new_table_reader_for_compaction_inputs
rocksdb_no_block_cache
rocksdb_no_create_column_family
rocksdb_override_cf_options
rocksdb_paranoid_checks
rocksdb_partial_index_sort_max_mem
rocksdb_pause_background_work
rocksdb_perf_context_level
rocksdb_persistent_cache_path
rocksdb_persistent_cache_size_mb

Variable Name
rocksdb_pin_l0_filter_and_index_blocks_in_cache
rocksdb_print_snapshot_conflict_queries
rocksdb_rate_limiter_bytes_per_sec
rocksdb_read_free_rpl
rocksdb_read_free_rpl_tables
rocksdb_records_in_range
rocksdb_reset_stats
rocksdb_rollback_on_timeout
rocksdb_rpl_skip_tx_api
rocksdb_seconds_between_stat_computes
rocksdb_signal_drop_index_thread
rocksdb_sim_cache_size
rocksdb_skip_bloom_filter_on_read
rocksdb_skip_fill_cache
rocksdb_skip_locks_if_skip_unique_check
rocksdb_sst_mgr_rate_bytes_per_sec
rocksdb_stats_dump_period_sec
rocksdb_stats_level
rocksdb_stats_recalc_rate
rocksdb_store_row_debug_checksums
rocksdb_strict_collation_check
rocksdb_strict_collation_exceptions
rocksdb_table_cache_numshardbits
rocksdb_table_stats_background_thread_nice_value
rocksdb_table_stats_max_num_rows_scanned
rocksdb_table_stats_recalc_threshold_count
rocksdb_table_stats_recalc_threshold_pct
rocksdb_table_stats_sampling_pct
rocksdb_table_stats_use_table_scan
rocksdb_tmpdir
rocksdb_two_write_queues
rocksdb_trace_block_cache_access
rocksdb_trace_queries
rocksdb_trace_sst_api
rocksdb_track_and_verify_wals_in_manifest

Variable Name
`rocksdb_unsafe_for_binlog`
`rocksdb_update_cf_options`
`rocksdb_use_adaptive_mutex`
`rocksdb_use_default_sk_cf`
`rocksdb_use_direct_io_for_flush_and_compaction`
`rocksdb_use_direct_reads`
`rocksdb_use_fsync`
`rocksdb_validate_tables`
`rocksdb_verify_row_debug_checksums`
`rocksdb_wal_bytes_per_sync`
`rocksdb_wal_dir`
`rocksdb_wal_recovery_mode`
`rocksdb_wal_size_limit_mb`
`rocksdb_wal_ttl_seconds`
`rocksdb_whole_key_filtering`
`rocksdb_write_batch_max_bytes`
`rocksdb_write_disable_wal`
`rocksdb_write_ignore_missing_column_families`
`rocksdb_write_policy`

`rocksdb_access_hint_on_compaction_start`

Option	Description
Command-line	--rocksdb-access-hint-on-compaction-start
Dynamic	No
Scope	Global
Data type	String or numeric
Default	NORMAL or 1

The variable has been implemented in Percona Server 5.7.19-17. Specifies the file access pattern once compaction is started, and applied to all input files of compaction. Possible values are:

- `0` = `NONE`
- `1` = `NORMAL` (default)
- `2` = `SEQUENTIAL`
- `3` = `WILLNEED`

`rocksdb_advise_random_on_open`

Option	Description
Command-line	--rocksdb-advise-random-on-open
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to hint at the underlying file system that the file access pattern is random, when a data file is opened. Enabled by default.

`rocksdb_allow_concurrent_memtable_write`

Option	Description
Command-line	--rocksdb-allow-concurrent-memtable-write
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to allow multiple writers to update memtables in parallel. Disabled by default.

`rocksdb_allow_to_start_after_corruption`

Option	Description
Command-line	--rocksdb_allow_to_start_after_corruption
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to allow the server to restart once MyRocks reported data corruption. Disabled by default.

Once corruption is detected server writes a marker file (named ROCKSDB_CORRUPTED) in the data directory and aborts. If a marker file exists, then mysqld exits on startup with an error message. The restart failure will continue until the problem is solved or until mysqld is started with this variable turned on in the command line.

Note

Not all memtables support concurrent writes.

`rocksdb_allow_mmap_reads`

Option	Description
Command-line	--rocksdb-allow-mmap-reads
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to allow the OS to map a data file into memory for reads. Disabled by default. If you enable this, make sure that rocksdb_use_direct_reads is disabled.

`rocksdb_allow_mmap_writes`

Option	Description
Command-line	--rocksdb-allow-mmap-writes
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to allow the OS to map a data file into memory for writes. Disabled by default.

`rocksdb_alter_column_default_inplace`

Option	Description
Command-line	--rocksdb-alter-column-default-inplace
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server for MySQL 5.7.35-38. Allow inplace alter for the alter column default operation.

`rocksdb_base_background_compactions`

Option	Description
Command-line	--rocksdb-base-background-compactions
Dynamic	No
Scope	Global
Data type	Numeric
Default	1

The variable has been implemented in Percona Server 5.7.19-17. This variable has been replaced in Percona Server 5.7.20-18 by rocksdb_max_background_jobs, which automatically decides how many threads to allocate towards flush/compaction. Specifies the suggested number of concurrent background compaction jobs, submitted to the default LOW priority thread pool in RocksDB. Default is `1`. The allowed range of values is from `-1` to `64`. Maximum depends on the rocksdb_max_background_compactions variable.

`rocksdb_blind_delete_primary_key`

Option	Description
Command-line	--rocksdb-blind-delete-primary-key
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server for MySQL 5.7.30-33. Skips verifying if rows exist before executing deletes. The following conditions must be met:

- The variable is enabled
- Only a single table is listed in the `DELETE` statement
- The table has only a primary key with no secondary keys

`rocksdb_block_cache_size`

Option	Description
Command-line	--rocksdb-block-cache-size
Dynamic	No
Scope	Global
Data type	Numeric
Default	536870912

The variable has been implemented in Percona Server 5.7.19-17. Specifies the size of the LRU block cache for RocksDB. This memory is reserved for the block cache, which is in addition to any filesystem caching that may occur.

The minimum value is `1024` because that's the size of one block.

The default value is `536870912`.

The maximum value is `9223372036854775807`.

`rocksdb_block_restart_interval`

Option	Description
Command-line	--rocksdb-block-restart-interval
Dynamic	No
Scope	Global
Data type	Numeric
Default	16

The variable has been implemented in Percona Server 5.7.19-17. Specifies the number of keys for each set of delta encoded data. The default value is `16`. The allowed range is from `1` to `2147483647`.

`rocksdb_block_size`

Option	Description
Command-line	--rocksdb-block-size
Dynamic	No
Scope	Global
Data type	Numeric
Default	4096

The variable has been implemented in Percona Server 5.7.19-17. The minimum value has changed from `0` to `1024` in Percona Server 5.7.20-18. This variable specifies the size of the data block for reading RocksDB data files. The default value is `4096`. The allowed range is from `1024` to `18446744073709551615`.

`rocksdb_block_size_deviation`

Option	Description
Command-line	--rocksdb-block-size-deviation
Dynamic	No
Scope	Global
Data type	Numeric
Default	10

The variable has been implemented in Percona Server 5.7.19-17. Specifies the threshold for free space allowed in a data block (see rocksdb_block_size). If there is less space remaining, close the block (and write to new block). Default value is `10`, meaning that the block is not closed until there is less than 10 bits of free space remaining.

The allowed range is from `1` to `2147483647`.

`rocksdb_bulk_load_allow_sk`

Option	Description
Command-line	--rocksdb-bulk-load-allow-sk
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.23-23. Enabling this variable allows secondary keys to be added using the bulk loading feature. This variable can be toggled only when the bulk load is disabled, for example, when rocksdb_bulk_load is `OFF`.

`rocksdb_bulk_load_allow_unsorted`

Option	Description
Command-line	--rocksdb-bulk-load-allow-unsorted
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.20-18. By default, the bulk loader requires its input to be sorted in the primary key order. If enabled, unsorted inputs are allowed too, which are then sorted by the bulkloader itself, at a performance penalty.

`rocksdb_bulk_load`

Option	Description
Command-line	--rocksdb-bulk-load
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to use bulk load: MyRocks will ignore checking keys for uniqueness or acquiring locks during transactions. Disabled by default. Enable this only if you are certain that there are no row conflicts, for example, when setting up a new MyRocks instance from a MySQL dump.

When the rocksdb_bulk_load variable is enabled, it behaves as if the variable rocksdb_commit_in_the_middle is enabled, even if the variable rocksdb_commit_in_the_middle is disabled.

`rocksdb_bulk_load_size`

Option	Description
Command-line	--rocksdb-bulk-load-size
Dynamic	Yes
Scope	Global, Session
Data type	Numeric
Default	1000

The variable has been implemented in Percona Server 5.7.19-17. Specifies the number of keys to accumulate before committing them to the storage engine when the bulk load is enabled (see rocksdb_bulk_load). The default value is `1000`, which means that a batch can contain up to 1000 records before they are implicitly committed. The allowed range is from `1` to `1073741824`.

`rocksdb_bytes_per_sync`

Option	Description
Command-line	--rocksdb-bytes-per-sync
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17 and changed to dynamic in Percona Server 5.7.21-20. Specifies how often should the OS sync files to disk as they are being written, asynchronously, in the background. This operation can be used to smooth out write I/O over time. The default value is `0` meaning that files are never synced. The allowed range is up to `18446744073709551615`.

`rocksdb_cache_dump`

Option	Description
Command-line	--rocksdb-cache-dump
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server for MySQL 5.7.30-33. Includes RocksDB block cache content in a core dump. This variable is enabled by default.

`rocksdb_cache_index_and_filter_blocks`

Option	Description
Command-line	--rocksdb-cache-index-and-filter-blocks
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether RocksDB should use the block cache for caching the index and bloomfilter data blocks from each data file. Enabled by default. If you disable this feature, RocksDB will allocate additional memory to maintain these data blocks.

`rocksdb_checksums_pct`

Option	Description
Command-line	--rocksdb-checksums-pct
Dynamic	Yes
Scope	Global, Session
Data type	Numeric
Default	100

The variable has been implemented in Percona Server 5.7.19-17. Specifies the percentage of rows to be checksummed. The default value is `100` (checksum all rows). The allowed range is from `0` to `100`.

`rocksdb_collect_sst_properties`

Option	Description
Command-line	--rocksdb-collect-sst-properties
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to collect statistics on each data file to improve optimizer behavior. Enabled by default.

`rocksdb_commit_in_the_middle`

Option	Description
Command-line	--rocksdb-commit-in-the-middle
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to commit rows implicitly when a batch contains more than the value of rocksdb_bulk_load_size. This option should only be enabled at the time of data import because it may cause locking errors.

This variable is disabled by default.

When the rocksdb_bulk_load variable is enabled, it behaves as if the variable rocksdb_commit_in_the_middle is enabled, even if the variable rocksdb_commit_in_the_middle is disabled.

`rocksdb_commit_time_batch_for_recovery`

Option	Description
Command-line	--rocksdb-commit-time-batch-for-recovery
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.23-23. Specifies whether to write the commit time write batch into the database or not.

Note

If the commit time write batch is only useful for recovery, then writing to WAL is enough.

`rocksdb_compact_cf`

Option	Description
Command-line	--rocksdb-compact-cf
Dynamic	Yes
Scope	Global
Data type	String
Default	

The variable has been implemented in Percona Server 5.7.19-17. Specifies the name of the column family to compact.

`rocksdb_compaction_readahead_size`

Option	Description
Command-line	--rocksdb-compaction-readahead-size
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the size of reads to perform ahead of compaction. The default value is `0`. Set this to at least 2 megabytes (`16777216`) when using MyRocks with spinning disks to ensure sequential reads instead of random. The maximum allowed value is `18446744073709551615`.

Note

If you set this variable to a non-zero value, rocksdb_new_table_reader_for_compaction_inputs is enabled.

`rocksdb_compaction_sequential_deletes`

Option	Description
Command-line	--rocksdb-compaction-sequential-deletes
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the threshold to trigger compaction on a file if it has more than this number of sequential delete markers. The default value is `0` meaning that compaction is not triggered regardless of the number of delete markers. The maximum allowed value is `2000000` (two million delete markers).

Note

Depending on workload patterns, MyRocks can potentially maintain large numbers of delete markers, which increases the latency of queries. This compaction feature will reduce latency, but may also increase the MyRocks write rate. Use this variable together with rocksdb_compaction_sequential_deletes_file_size to only perform compaction on large files.

`rocksdb_compaction_sequential_deletes_count_sd`

Option	Description
Command-line	--rocksdb-compaction-sequential-deletes-count-sd
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to count single deletes as delete markers recognized by rocksdb_compaction_sequential_deletes. Disabled by default.

`rocksdb_compaction_sequential_deletes_file_size`

Option	Description
Command-line	--rocksdb-compaction-sequential-deletes-file-size
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the minimum file size required to trigger compaction on it by rocksdb_compaction_sequential_deletes. The default value is `0`, meaning that compaction is triggered regardless of file size. The allowed range is from `-1` to `9223372036854775807`.

`rocksdb_compaction_sequential_deletes_window`

Option	Description
Command-line	--rocksdb-compaction-sequential-deletes-window
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the size of the window for counting delete markers by rocksdb_compaction_sequential_deletes. The default value is `0`. The allowed range is up to `2000000` (two million).

`rocksdb_concurrent_prepare`

Option	Description
Command-line	--rocksdb-concurrent_prepare
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been deprecated in the *Percona Server for MySQL* 5.7.21-20, as it has been renamed upstream to `rocksdb_two_write_queues`.

The variable has been implemented in Percona Server 5.7.20-18. When enabled this variable allows/encourages threads that are using two-phase commit to `prepare` in parallel.

`rocksdb_create_checkpoint`

Option	Description
Command-line	--rocksdb-create-checkpoint
Dynamic	Yes
Scope	Global
Data type	String
Default	

The variable has been implemented in Percona Server 5.7.19-17. Specifies the directory where MyRocks should create a checkpoint. Empty by default.

`rocksdb_create_if_missing`

Option	Description
Command-line	--rocksdb-create-if-missing
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether MyRocks should create its database if it does not exist. Enabled by default.

`rocksdb_create_missing_column_families`

Option	Description
Command-line	--rocksdb-create-missing-column-families
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether MyRocks should create new column families if they do not exist. Disabled by default.

`rocksdb_datadir`

Option	Description
Command-line	--rocksdb-datadir
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies the location of the MyRocks data directory. By default, it is created in the current working directory.

`rocksdb_db_write_buffer_size`

Option	Description
Command-line	--rocksdb-db-write-buffer-size
Dynamic	No
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the maximum size of all memtables used to store writes in MyRocks across all column families. When this size is reached, the data is flushed to persistent media. The default value is `0`. The allowed range is up to `18446744073709551615`.

`rocksdb_deadlock_detect`

Option	Description
Command-line	--rocksdb-deadlock-detect
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether MyRocks should detect deadlocks. Disabled by default.

`rocksdb_deadlock_detect_depth`

Implemented in Percona Server 5.7.20-18.

Option	Description
Command-line	--rocksdb-deadlock-detect-depth
Dynamic	Yes
Scope	Global, Session
Data type	Numeric
Default	50

The variable has been implemented in Percona Server 5.7.20-18. Specifies the number of transactions deadlock detection will traverse through before assuming deadlock.

`rocksdb_debug_optimizer_no_zero_cardinality`

Option	Description
Command-line	--rocksdb-debug-optimizer-no-zero-cardinality
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether MyRocks should prevent zero cardinality by always overriding it with some value.

`rocksdb_debug_ttl_ignore_pk`

Option	Description
Command-line	--rocksdb-debug-ttl-ignore-pk
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.20-18, for debugging purposes only. If true, compaction filtering will not occur on Primary Key TTL data. This variable is a no-op in non-debug builds.

`rocksdb_debug_ttl_read_filter_ts`

Option	Description
Command-line	--rocksdb_debug-ttl-read-filter-ts
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.20-18. For debugging purposes only. Overrides the TTL read filtering time to time + debug_ttl_read_filter_ts. A value of `0` denotes that the variable is not set. This variable is a no-op in non-debug builds.

`rocksdb_debug_ttl_rec_ts`

Option	Description
Command-line	--rocksdb-debug-ttl-rec-ts
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.20-18. For debugging purposes only. Overrides the TTL of records to `now()` + debug_ttl_rec_ts. The value can be +/- to simulate a record inserted in the past vs a record inserted in the “future”. A value of `0` denotes that the variable is not set. This variable is a no-op in non-debug builds.

`rocksdb_debug_ttl_snapshot_ts`

Option	Description
Command-line	--rocksdb-debug-ttl-snapshot-ts
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.20-18. For debugging purposes only. Sets the snapshot during compaction to `now()` + rocksdb_debug_set_ttl_snapshot_ts. The value can be +/- to simulate a snapshot in the past vs a snapshot created in the "future". A value of `0` denotes that the variable is not set. This variable is a no-op in non-debug builds.

`rocksdb_default_cf_options`

Option	Description
Command-line	--rocksdb-default-cf-options
Dynamic	No
Scope	Global
Data type	String
Default	

The variable has been implemented in Percona Server 5.7.19-17. Specifies the default column family options for MyRocks. Empty by default.

`rocksdb_delayed_write_rate`

Option	Description
Command-line	--rocksdb-delayed-write-rate
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	16777216

The variable has been implemented in Percona Server 5.7.19-17. Specifies the write rate in bytes per second, which should be used if MyRocks hits a soft limit or threshold for writes. Default value is `16777216` (16 MB/sec). The allowed range is from `0` to `18446744073709551615`.

`rocksdb_delete_cf`

Option	Description
Command-line	--rocksdb-delete-cf
Dynamic	Yes
Scope	Global
Data type	String
Default	“”

The variable has been implemented in Percona Server for MySQL 5.7.30-33. Deletes the column family by name. The default value is “”, an empty string.

For example:

```
SET @@global.ROCKSDB_DELETE_CF = 'cf_primary_key';
```

`rocksdb_delete_obsolete_files_period_micros`

Option	Description
Command-line	--rocksdb-delete-obsolete-files-period-micros
Dynamic	No
Scope	Global
Data type	Numeric
Default	21600000000

The variable has been implemented in Percona Server 5.7.19-17. Specifies the period in microseconds to delete obsolete files regardless of files removed during compaction. Default value is `21600000000` (6 hours). Allowed range is up to `9223372036854775807`.

`rocksdb_enable_bulk_load_api`

Option	Description
Command-line	--rocksdb-enable-bulk-load-api
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to use the `SSTFileWriter` feature for bulk loading. This feature bypasses the memtable, but requires keys to be inserted into the table in either ascending or descending order. Enabled by default. If disabled, bulk loading uses the normal write path via the memtable and does not require keys to be inserted in any order.

`rocksdb_enable_insert_with_update_caching`

Option	Description
Command-line	--rocksdb-enable-insert-with-update-caching
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server for MySQL 5.7.30-33. Specifies whether to enable optimization where the read is cached from a failed insertion attempt in INSERT ON DUPLICATE KEY UPDATE.

`rocksdb_enable_iterate_bounds`

Option	Description
Command-line	--rocksdb-enable-iterate-bounds
Dynamic	Yes
Scope	Global, Local
Data type	Boolean
Default	TRUE

The variable has been implemented in Percona Server for MySQL 5.7.30-33. Enables the rocksdb iterator upper bounds and lower bounds in read options.

Implemented in Percona Server for MySQL 5.7.35-38.

`rocksdb_enable_native_partition`

Option	Description
Command-line	--rocksdb-enable-native-partition
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

This variable is **experimental** and should not be used in production.

This variable enables native partitioning and may be used when upgrading to 8.0.

`rocksdb_enable_pipelined_write`

Option	Description
Command-line	--rocksdb-enable-pipelined-write
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

DBOptions::enable_pipelined_write for RocksDB.

The variable has been implemented in Percona Server for MySQL 5.7.35-38.

If `enable_pipelined_write` is `true`, a separate write thread is maintained for WAL write and memtable write. A write thread first enters the WAL writer queue and then the memtable writer queue. A pending thread on the WAL writer queue only waits for the previous WAL write operations but does not wait for memtable write operations. Enabling the feature may improve write throughput and reduce latency of the prepare phase of a two-phase commit.

`rocksdb_enable_remove_orphaned_dropped_cfs`

Option	Description
Command-line	--rocksdb-enable-remove-orphaned-dropped-cfs
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	TRUE

The variable has been implemented in Percona Server for MySQL 5.7.30-33. Enables the removal of dropped column families (cfs) from metadata if the cfs do not exist in the cf manager.

The default value is `TRUE`.

`rocksdb_enable_ttl`

Option	Description
Command-line	--rocksdb-enable-ttl
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to keep expired TTL records during compaction. Enabled by default. If disabled, expired TTL records will be dropped during compaction.

`rocksdb_enable_ttl_read_filtering`

Option	Description
Command-line	--rocksdb-enable-ttl-read-filtering
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.20-18. For tables with TTL, expired records are skipped/filtered out during processing and in query results. Disabling this will allow these records to be seen, but as a result, rows may disappear in the middle of transactions as they are dropped during compaction. **Use with caution.**

`rocksdb_enable_thread_tracking`

Option	Description
Command-line	--rocksdb-enable-thread-tracking
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to enable tracking the status of threads accessing the database. Disabled by default. If enabled, thread status will be available via `GetThreadList()`.

`rocksdb_enable_write_thread_adaptive_yield`

Option	Description
Command-line	--rocksdb-enable-write-thread-adaptive-yield
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether the MyRocks write batch group leader should wait up to the maximum allowed time before blocking on a mutex. Disabled by default. Enable it to increase throughput for concurrent workloads.

`rocksdb_error_if_exists`

Option	Description
Command-line	--rocksdb-error-if-exists
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to report an error when a database already exists. Disabled by default.

`rocksdb_error_on_suboptimal_collation`

Option	Description
Command-line	--rocksdb-error-on-suboptimal-collation
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.23-23. Specifies whether to report an error instead of a warning if an index is created on a char field where the table has a sub-optimal collation (case insensitive). Enabled by default.

`rocksdb_flush_log_at_trx_commit`

Option	Description
Command-line	--rocksdb-flush-log-at-trx-commit
Dynamic	Yes
Scope	Global, Session
Data type	Numeric
Default	1

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to sync on every transaction commit, similar to innodb_flush_log_at_trx_commit. Enabled by default, which ensures ACID compliance.

Possible values:

- `0`: Do not sync on transaction commit. This provides better performance but may lead to data inconsistency in case of a crash.
- `1`: Sync on every transaction commit. This is set by default and recommended as it ensures data consistency, but reduces performance.
- `2`: Sync every second.

`rocksdb_flush_memtable_on_analyze`

Option	Description
Command-line	--rocksdb-flush-memtable-on-analyze
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17 and removed in Percona Server 5.7.21-20. Specifies whether to flush the memtable when running `ANALYZE` on a table. Enabled by default. This ensures accurate cardinality by including data in the memtable for calculating stats.

`rocksdb_force_compute_memtable_stats`

Option	Description
Command-line	--rocksdb-force-compute-memtable-stats
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether data in the memtables should be included for calculating index statistics used by the query optimizer. Enabled by default. This provides better accuracy but may reduce performance.

`rocksdb_force_compute_memtable_stats_cachetime`

Option	Description
Command-line	--rocksdb-force-compute-memtable-stats-cachetime
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	60000000

The variable has been implemented in Percona Server 5.7.20-18. Specifies for how long the cached value of memtable statistics should be used instead of computing it every time during the query plan analysis.

`rocksdb_force_flush_memtable_and_lzero_now`

Option	Description
Command-line	--rocksdb-force-flush-memtable-and-lzero-now
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Works similar to force_flush_memtable_now but also flushes all L0 files.

`rocksdb_force_flush_memtable_now`

Option	Description
Command-line	--rocksdb-force-flush-memtable-now
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Forces MyRocks to immediately flush all memtables out to data files.

Warning

Use with caution! Write requests will be blocked until all memtables are flushed.

`rocksdb_force_index_records_in_range`

Option	Description
Command-line	--rocksdb-force-index-records-in-range
Dynamic	Yes
Scope	Global, Session
Data type	Numeric
Default	1

The variable has been implemented in Percona Server 5.7.19-17. Specifies the value used to override the number of rows returned to query optimizer when `FORCE INDEX` is used. The default value is `1`. The allowed range is from `0` to `2147483647`. Set to `0` if you do not want to override the returned value.

`rocksdb_hash_index_allow_collision`

Option	Description
Command-line	--rocksdb-hash-index-allow-collision
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether hash collisions are allowed. Enabled by default, which uses less memory. If disabled, the full prefix is stored to prevent hash collisions.

`rocksdb_ignore_unknown_options`

Option	Description
Command-line	
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.20-18. When enabled, it allows RocksDB to receive unknown options and not exit.

`rocksdb_index_type`

Option	Description
Command-line	--rocksdb-index-type
Dynamic	No
Scope	Global
Data type	Enum
Default	kBinarySearch

The variable has been implemented in Percona Server 5.7.19-17. Specifies the type of indexing used by MyRocks:

- `kBinarySearch`: Binary search (default).
- `kHashSearch`: Hash search.

`rocksdb_info_log_level`

Option	Description
Command-line	--rocksdb-info-log-level
Dynamic	Yes
Scope	Global
Data type	Enum
Default	error_level

The variable has been implemented in Percona Server 5.7.19-17. Specifies the level for filtering messages written by MyRocks to the `mysqld` log.

- `debug_level`: Maximum logging (everything including debugging log messages)
- `info_level`
- `warn_level`
- `error_level` (default)
- `fatal_level`: Minimum logging (only fatal error messages logged)

`rocksdb_is_fd_close_on_exec`

Option	Description
Command-line	--rocksdb-is-fd-close-on-exec
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether child processes should inherit open file handles. Enabled by default.

`rocksdb_large_prefix`

Option	Description
Command-line	--rocksdb-large-prefix
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.20-18. When enabled, this option allows index key prefixes longer than 767 bytes (up to 3072 bytes). This option mirrors the innodb_large_prefix The values for rocksdb_large_prefix should be the same between source and replica.

`rocksdb_keep_log_file_num`

Option	Description
Command-line	--rocksdb-keep-log-file-num
Dynamic	No
Scope	Global
Data type	Numeric
Default	1000

The variable has been implemented in Percona Server 5.7.19-17. Specifies the maximum number of info log files to keep. The default value is `1000`. The allowed range is from `1` to `18446744073709551615`.

`rocksdb_lock_scanned_rows`

Option	Description
Command-line	--rocksdb-lock-scanned-rows
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to hold the lock on rows that are scanned during `UPDATE` and not actually updated. Disabled by default.

`rocksdb_lock_wait_timeout`

Option	Description
Command-line	--rocksdb-lock-wait-timeout
Dynamic	Yes
Scope	Global, Session
Data type	Numeric
Default	1

The variable has been implemented in Percona Server 5.7.19-17. Specifies the number of seconds MyRocks should wait to acquire a row loc before aborting the request. The default value is `1`. The allowed range is up to `1073741824`.

`rocksdb_log_file_time_to_roll`

Option	Description
Command-line	--rocksdb-log-file-time-to-roll
Dynamic	No
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the period (in seconds) for rotating the info log files. The default value is `0`, meaning that the log file is not rotated. The allowed range is up to `18446744073709551615`.

`rocksdb_manifest_preallocation_size`

Option	Description
Command-line	--rocksdb-manifest-preallocation-size
Dynamic	No
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the number of bytes to preallocate for the MANIFEST file used by MyRocks to store information about column families, levels, active files, etc. The default value is `0`. The allowed range is up to `18446744073709551615`.

Note

A value of `4194304` (4 MB) is reasonable to reduce random I/O on XFS.

`rocksdb_manual_compaction_bottommost_level`

Option	Description
Command-line	--rocksdb-manual-compaction-bottommost-level
Dynamic	Yes
Scope	Global, Session
Data type	Enum
Default	kForceOptimized

The variable has been implemented in Percona Server for MySQL 5.7.35-38.

Option for skipping bottommost level compaction during manual compaction. The values are the following:

- `kSkip` - Skip bottommost level compaction
- `kIfHaveCompactionFilter` - Only compact the bottommost level if there is a compaction filter
- `kForce` - Always compact the bottommost level
- `kForceOptimized` - The default value. Always compact the bottommost level but in the bottommost level avoid double-compacting files created

`rocksdb_manual_wal_flush`

Option	Description
Command-line	--rocksdb-manual-wal-flush
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.20-18. This variable can be used to disable automatic/timed WAL flushing and instead rely on the application to do the flushing.

`rocksdb_master_skip_tx_api`

Option	Description
Command-line	
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server for MySQL 5.7.30-33. When enabled, uses the WriteBatch API, which is faster. The session does not hold any lock-on-row access. This variable is not effective on replicas.

Note

Due to the disabled row locks, improper use of the variable can cause data corruption or inconsistency.

`rocksdb_max_background_compactions`

Option	Description
Command-line	--rocksdb-max-background-compactions
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	-1

The variable has been implemented in Percona Server 5.7.19-17. This variable has been replaced in Percona Server 5.7.20-18 by rocksdb_max_background_jobs, which automatically decides how many threads to allocate towards flush/compaction. This variable has been re-implemented in Percona Server for MySQL 5.7.31-34.

Sets DBOptions:: max_background_compactions for RocksDB. The default value is `-1`. The allowed range is up to `64`.

`rocksdb_max_background_flushes`

Option	Description
Command-line	--rocksdb-max-background-flushes
Dynamic	No
Scope	Global
Data type	Numeric
Default	-1

The variable has been implemented in Percona Server 5.7.19-17. This variable has been replaced in Percona Server 5.7.20-18 by rocksdb_max_background_jobs, which automatically decides how many threads to allocate towards flush/compaction. This variable has been re-implemented in Percona Server for MySQL 5.7.31-34.

Sets DBOptions:: max_background_flushes for RocksDB. The default value is `-1`. The allowed range is up to `64`.

`rocksdb_max_background_jobs`

Option	Description
Command-line	--rocksdb-max-background-jobs
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	2

This variable has been implemented in Percona Server 5.7.20-18 to replace rocksdb_base_background_compactions, rocksdb_max_background_compactions, and rocksdb_max_background_flushes variables. This variable specifies the maximum number of background jobs. It automatically decides how many threads to allocate towards flush/compaction. It was implemented

to reduce the number of (confusing) options for users and can tweak and push the responsibility down to RocksDB level.

`rocksdb_max_bottom_pri_background_compactions`

Option	Description
Command-line	--rocksdb_max_bottom_pri_background_compactions
Dynamic	No
Data type	Unsigned integer
Default	0

This variable has been implemented in Percona Server for MySQL 5.7.31-34. Creates a specified number of threads, sets a lower CPU priority, and lets compactions use them. The maximum compaction concurrency is capped by `rocksdb_max_background_compactions` or `rocksdb_max_background_jobs`

The minimum value is `0` and the maximum value is `64`.

`rocksdb_max_latest_deadlocks`

Option	Description
Command-line	--rocksdb-max-latest-deadlocks
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	5

This variable has been implemented in Percona Server 5.7.20-18. Specifies the maximum number of recent deadlocks to store.

`rocksdb_max_log_file_size`

Option	Description
Command-line	--rocksdb-max-log-file-size
Dynamic	No
Scope	Global
Data type	Numeric
Default	0

This variable has been implemented in Percona Server 5.7.19-17. Specifies the maximum size for info log files, after which the log is rotated. The default value is `0`, meaning that only one log file is used. The allowed range is up to `18446744073709551615`.

Also see rocksdb_log_file_time_to_roll.

`rocksdb_max_manifest_file_size`

Option	Description
Command-line	--rocksdb-manifest-log-file-size
Dynamic	No
Scope	Global
Data type	Numeric
Default	18446744073709551615

This variable has been implemented in Percona Server 5.7.19-17. Specifies the maximum size of the MANIFEST data file, after which it is rotated. The default value is also the maximum, making it practically unlimited: only one manifest file is used.

`rocksdb_max_open_files`

Option	Description
Command-line	--rocksdb-max-open-files
Dynamic	No
Scope	Global
Data type	Numeric
Default	1000

This variable has been implemented in Percona Server 5.7.19-17. Default value has changed to `1000` in Percona Server 5.7.19-17. Specifies the maximum number of file handles opened by MyRocks. Values in the range between `0` and `open_files_limit` are taken as they are. If rocksdb_max_open_files value is greater than `open_files_limit`, it will be reset to 1/2 of `open_files_limit`, and a warning will be emitted to the `mysqld` error log. A value of `-2` denotes auto-tuning: just sets rocksdb_max_open_files value to 1/2 of `open_files_limit`. Finally, `-1` means no limit, i.e. an infinite number of file handles.

Warning

Setting rocksdb_max_open_files to `-1` is dangerous, as the server may quickly run out of file handles in this case.

`rocksdb_max_row_locks`

Option	Description
Command-line	--rocksdb-max-row-locks
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	1048576

This variable has been implemented in Percona Server 5.7.19-17. The default value has changed from `1073741824` to `1048576` in Percona Server 5.7.21-21. The scope has changed to `Global` in Percona Server for MySQL 5.7.32-35. Specifies the limit on the maximum number of row locks a transaction can have before it

fails. The default value is also the maximum, making it practically unlimited: transactions never fail due to row locks.

`rocksdb_max_subcompactions`

Option	Description
Command-line	--rocksdb-max-subcompactions
Dynamic	No
Scope	Global
Data type	Numeric
Default	1

The variable has been implemented in Percona Server 5.7.19-17. Specifies the maximum number of threads allowed for each compaction job. A default value of `1` means no subcompactions (one thread per compaction job). The allowed range is up to `64`.

`rocksdb_max_total_wal_size`

Option	Description
Command-line	--rocksdb-max-total-wal-size
Dynamic	No
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the maximum total size of WAL (write-ahead log) files, after which memtables are flushed. The default value is `0`: the WAL size limit is chosen dynamically. The allowed range is up to `9223372036854775807`.

`rocksdb_merge_buf_size`

Option	Description
Command-line	--rocksdb-merge-buf-size
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	67108864

The variable has been implemented in Percona Server 5.7.19-17. Specifies the size (in bytes) of the merge-sort buffers used to accumulate data during secondary key creation. New entries are written directly to the lowest level in the database, instead of updating indexes through the memtable and L0. These values are sorted using merge-sort, with buffers set to 64 MB by default (`67108864`). The allowed range is from `100` to `18446744073709551615`.

`rocksdb_merge_combine_read_size`

Option	Description
Command-line	--rocksdb-merge-combine-read-size
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	1073741824

The variable has been implemented in Percona Server 5.7.19-17. Specifies the size (in bytes) of the merge-combine buffer used for the merge-sort algorithm as described in rocksdb_merge_buf_size. The default size is 1 GB (`1073741824`). The allowed range is from `100` to `18446744073709551615` .

`rocksdb_merge_tmp_file_removal_delay_ms`

Option	Description
Command-line	--rocksdb_merge_tmp_file_removal_delay_ms
Dynamic	Yes
Scope	Global, Session
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.20-18. Fast secondary index creation creates merge files when needed. After finishing secondary index creation, merge files are removed. By default, the file removal is done without any sleep, so removing GBs of merge files within <1s may happen, which will cause trim stalls on Flash. This variable can be used to rate limit the delay in milliseconds.

`rocksdb_new_table_reader_for_compaction_inputs`

Option	Description
Command-line	--rocksdb-new-table-reader-for-compaction-inputs
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether MyRocks should create a new file descriptor and table reader for each compaction input. Disabled by default. Enabling this may increase memory consumption, but will also allow pre-fetch options to be specified for compaction input files without impacting table readers used for user queries.

`rocksdb_no_block_cache`

Option	Description
Command-line	--rocksdb-no-block-cache
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to disable the block cache for column families. Variable is disabled by default, meaning that using the block cache is allowed.

`rocksdb_no_create_column_family`

Option	Description
Command-line	--rocksdb-no-create-column-family
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.23-24. Specifies whether column families can be created implicitly via an index comment. If this variable is set to `ON`, then column families must already exist or must be present within the rocksdb_override_cf_options for a user to assign and index to a column family.

`rocksdb_override_cf_options`

Option	Description
Command-line	--rocksdb-override-cf-options
Dynamic	No
Scope	Global
Data type	String
Default	

The variable has been implemented in Percona Server 5.7.19-17. Specifies option overrides for each column family. Empty by default.

`rocksdb_paranoid_checks`

Option	Description
Command-line	--rocksdb-paranoid-checks
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether MyRocks should re-read the data file as soon as it is created to verify correctness. Enabled by default.

`rocksdb_pause_background_work`

Option	Description
Command-line	--rocksdb-pause-background-work
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether MyRocks should pause all background operations. Disabled by default. There is no practical reason for a user to ever use this variable because it is intended as a test synchronization tool for the MyRocks MTR test suites.

Warning

If someone were to set a rocksdb_force_flush_memtable_now to `1` while rocksdb_pause_background_work is set to `1`, the client that issued the `rocksdb_force_flush_memtable_now=1` will be blocked indefinitely until rocksdb_pause_background_work is set to `0`.

`rocksdb_perf_context_level`

Option	Description
Command-line	--rocksdb-perf-context-level
Dynamic	Yes
Scope	Global, Session
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the level of information to capture with the Perf Context plugins. The default value is `0`. The allowed range is up to `5`.

Value	Description
0	Unknown setting
1	Disable perf stats
2	Enable only count stats
3	Enable count stats and time stats except for mutexes
4	Enable count stats, time stats, except for wall time or CPU time for mutexes
5	Enable count and time stats

`rocksdb_persistent_cache_path`

Option	Description
Command-line	--rocksdb-persistent-cache-path
Dynamic	No
Scope	Global
Data type	String
Default	

The variable has been implemented in Percona Server 5.7.19-17. Specifies the path to the persistent cache. Set this together with rocksdb_persistent_cache_size_mb.

`rocksdb_persistent_cache_size_mb`

Option	Description
Command-line	--rocksdb-persistent-cache-size-mb
Dynamic	No
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the size of the persistent cache in megabytes. Default is `0` (persistent cache disabled). The allowed range is up to `18446744073709551615`. Set this together with rocksdb_persistent_cache_path.

`rocksdb_pin_l0_filter_and_index_blocks_in_cache`

Option	Description
Command-line	--rocksdb-pin-l0-filter-and-index-blocks-in-cache
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether MyRocks pins the filter and index blocks in the cache if rocksdb_cache_index_and_filter_blocks is enabled. Enabled by default.

`rocksdb_print_snapshot_conflict_queries`

Option	Description
Command-line	--rocksdb-print-snapshot-conflict-queries
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether queries that generate snapshot conflicts should be logged to the error log. Disabled by default.

`rocksdb_rate_limiter_bytes_per_sec`

Option	Description
Command-line	--rocksdb-rate-limiter-bytes-per-sec
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the maximum rate at which MyRocks can write to media via memtable flushes and compaction. The default value is `0` (the write rate is not limited). The allowed range is up to `9223372036854775807`.

`rocksdb_read_free_rpl`

Option	Description
Command-line	--rocksdb-read-free-rpl
Dynamic	Yes
Scope	Global
Data type	Enum
Default	OFF

The variable has been implemented in Percona Server for MySQL 5.7.30-33. se read-free replication, which allows no row lookup during replication, on the replica.

The options are the following:

- OFF - Disables the variable
- PK_SK - Enables the variable on all tables with a primary key
- PK_ONLY - Enables the variable on tables where the only key is the primary key

`rocksdb_read_free_rpl_tables`

Option	Description
Command-line	--rocksdb-read-free-rpl-tables
Dynamic	Yes
Scope	Global, Session
Data type	String
Default	

This variable is disabled in *Percona Server for MySQL* 5.7.30-33. We recommend that you use `rocksdb_read_free_rpl` instead of this variable.

The variable has been implemented in Percona Server 5.7.19-17 and disabled in Percona Server for MySQL 5.7.30-33. Lists tables (as a regular expression) that should use read-free replication on the replica (that is, replication without row lookups). Empty by default.

`rocksdb_records_in_range`

Option	Description
Command-line	--rocksdb-records-in-range
Dynamic	Yes
Scope	Global, Session
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the value to override the result of `records_in_range()`. The default value is `0`. The allowed range is up to `2147483647`.

`rocksdb_reset_stats`

Option	Description
Command-line	--rocksdb-reset-stats
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Resets MyRocks internal statistics dynamically (without restarting the server).

`rocksdb_rollback_on_timeout`

Option	Description
Command-line	--rocksdb-rollback-on-timeout
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server for MySQL 5.7.30-33. By default, only the last statement on a transaction is rolled back. If `--rocksdb-rollback-on-timeout=ON`, a transaction timeout causes a rollback of the entire transaction.

`rocksdb_rpl_skip_tx_api`

Option	Description
Command-line	--rocksdb-rpl-skip-tx-api
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17 and removed in Percona Server 5.7.20-19. The variable has been re-implemented in Percona Server 5.7.21-21. Specifies whether write batches should be used for replication thread instead of the transaction API. Disabled by default.

There are two conditions that are necessary to use it: row replication format and replica operating in super read-only mode.

`rocksdb_seconds_between_stat_computes`

Option	Description
Command-line	--rocksdb-seconds-between-stat-computes
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	3600

The variable has been implemented in Percona Server 5.7.19-17. Specifies the number of seconds to wait between recomputation of table statistics for the optimizer. During that time, only changed indexes are updated. The default value is `3600`. The allowed value is from `0` to `4294967295`.

`rocksdb_signal_drop_index_thread`

Option	Description
Command-line	--rocksdb-signal-drop-index-thread
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Signals the MyRocks drop index thread to wake up.

`rocksdb_sim_cache_size`

Option	Description
Command-line	--rocksdb-sim-cache-size
Dynamic	No
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.20-18. Enables the simulated cache, which allows us to figure out the hit/miss rate with a specific cache size without changing the real block cache.

`rocksdb_skip_bloom_filter_on_read`

Option	Description
Command-line	--rocksdb-skip-bloom-filter-on_read
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether bloom filters should be skipped on reads. Disabled by default (bloom filters are not skipped).

`rocksdb_skip_fill_cache`

Option	Description
Command-line	--rocksdb-skip-fill-cache
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF`

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to skip caching data on the read requests. Disabled by default (caching is not skipped).

`rocksdb_skip_locks_if_skip_unique_check`

Option	Description
Command-line	rocksdb_skip_locks_if_skip_unique_check
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server for MySQL 5.7.35-38. Skips row locking when unique checks are disabled.

`rocksdb_sst_mgr_rate_bytes_per_sec`

Option	Description
Command-line	--rocksdb-sst-mgr-rate-bytes-per-sec
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. The default value has changed from `67108864` to `0` in Percona Server 5.7.20-18. Specifies the maximum rate for writing to data files. The default value is `0`. This option is not effective on HDD. The allowed range is from `0` to `18446744073709551615`.

`rocksdb_stats_dump_period_sec`

Option	Description
Command-line	--rocksdb-stats-dump-period-sec
Dynamic	No
Scope	Global
Data type	Numeric
Default	600

The variable has been implemented in Percona Server 5.7.19-17. Specifies the period in seconds for performing a dump of the MyRocks statistics to the info log. The default value is `600`. The allowed range is up to `2147483647`.

`rocksdb_stats_level`

Option	Description
Command-line	--rocksdb-stats-level
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server for MySQL 5.7.30-33. Controls the RocksDB statistics level. The default value is “0” (kExceptHistogramOrTimers), which is the fastest level. The maximum value is “4”.

`rocksdb_stats_recalc_rate`

Option	Description
Command-line	--rocksdb-stats-recalc-rate
Dynamic	No
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.23-23. Specifies the number of indexes to recalculate per second. Recalculating index statistics periodically ensures it matches the actual sum from SST files. The default value is `0`. The allowed range is up to `4294967295`.

`rocksdb_store_row_debug_checksums`

Option	Description
Command-line	--rocksdb-store-row-debug-checksums
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to include checksums when writing index or table records. Disabled by default.

`rocksdb_strict_collation_check`

Option	Description
Command-line	--rocksdb-strict-collation-check
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to check and verify that table indexes have proper collation settings. Enabled by default.

`rocksdb_strict_collation_exceptions`

Option	Description
Command-line	--rocksdb-strict-collation-exceptions
Dynamic	Yes
Scope	Global
Data type	String
Default	

The variable has been implemented in Percona Server 5.7.19-17. Lists tables (as a regular expression) that should be excluded from verifying case-sensitive collation enforced by rocksdb_strict_collation_check. Empty by default.

`rocksdb_table_cache_numshardbits`

Option	Description
Command-line	--rocksdb-table-cache-numshardbits
Dynamic	No
Scope	Global
Data type	Numeric
Default	6

The variable has been implemented in Percona Server 5.7.19-17. Max value has been changed from `2147483647` to `19` in Percona Server 5.7.20-18. Specifies the number of table caches. The default value is `6`. The allowed range is from `0` to `19`.

`rocksdb_table_stats_background_thread_nice_value`

Option	Description
Command-line	--rocksdb-table-stats-background-thread-nice-value
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	19

The variable has been implemented in Percona Server for MySQL 5.7.30-33. The nice value for index stats. The minimum = -20 (THREAD_PRIO_MIN) The maximum = 19 (THREAD_PRIO_MAX)

`rocksdb_table_stats_max_num_rows_scanned`

Option	Description
Command-line	--rocksdb-table-stats-max-num-rows-scanned
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server for MySQL 5.7.30-33. The maximum number of rows to scan in a table scan is based on a cardinality calculation. The minimum is `0` (every modification triggers a stats recalculation). The maximum is `18,446,744,073,709,551,615`.

`rocksdb_table_stats_recalc_threshold_count`

Option	Description
Command-line	--rocksdb-table-stats-recalc-threshold-count
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	100

The variable has been implemented in Percona Server for MySQL 5.7.30-33. The number of modified rows to trigger a stats recalculation. This is a dependent variable for stats recalculation. The minimum is `0`. The maximum is `18,446,744,073,709,551,615`.

`rocksdb_table_stats_recalc_threshold_pct`

Option	Description
Command-line	--rocksdb-table-stats-recalc-threshold-pct
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	10

The variable has been implemented in Percona Server for MySQL 5.7.30-33. The percentage of the number of modified rows over the total number of rows to trigger stats recalculations. This is a dependent variable for stats recalculation. The minimum value is `0` The maximum value is `100` (RDB_TBL_STATS_RECALC_THRESHOLD_PCT_MAX).

`rocksdb_table_stats_sampling_pct`

Option	Description
Command-line	--rocksdb-table-stats-sampling-pct
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	10

The variable has been implemented in Percona Server 5.7.19-17. Specifies the percentage of entries to sample when collecting statistics about table properties. The default value is `10`. The allowed range is from `0` to `100`.

`rocksdb_table_stats_use_table_scan`

Option	Description
Command-line	--rocksdb-table-stats-use-table-scan
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	FALSE

The variable has been implemented in Percona Server for MySQL 5.7.30-33. Enables table-scan-based index calculations. The default value is `FALSE`.

`rocksdb_tmpdir`

Option	Description
Command-line	--rocksdb-tmpdir
Dynamic	Yes
Scope	Global, Session
Data type	String
Default	

The variable has been implemented in Percona Server 5.7.19-17. Specifies the path to the directory for temporary files during DDL operations.

`rocksdb_trace_block_cache_access`

Option	Description
Command-line	--rocksdb-trace-block-cache-access
Dynamic	Yes
Scope	Global
Data type	String
Default	""

The variable has been implemented in Percona Server for MySQL 5.7.30-33. Defines the block cache trace option string. The format is "sampling frequency: max_trace_file_size:trace_file_name." The sampling frequency value and max_trace_file_size value are positive integers. The block accesses are saved to the `rocksdb_datadir/block_cache_traces/trace_file_name`. The default value is an empty string ("").

`rocksdb_trace_queries`

Option	Description
Command-line	--rocksdb-trace-queries
Dynamic	Yes
Scope	Global
Data type	String
Default	""

The variable has been implemented in Percona Server for MySQL 5.7.35-38. This variable is a trace option string. The format is `sampling_frequency:max_trace_file_size:trace_file_name`. The `sampling_frequency` value and `max_trace_file_size` value are positive integers. The queries are saved to the rocksdb_datadir/queries_traces/trace_file_name.

The file size unit is measured in bytes.

The sampling frequency specifies that one request is sampled from `sampling_frequency` requests. If the request is `1`, all the requests are traced. If the request is `5`, then for every five requests, one request is traced.

`rocksdb_trace_sst_api`

Option	Description
Command-line	--rocksdb-trace-sst-api
Dynamic	Yes
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to generate trace output in the log for each call to `SstFileWriter`. Disabled by default.

`rocksdb_track_and_verify_wals_in_manifest`

Option	Description
Command-line	--rocksdb-track-and-verify-wals-in-manifest
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server for MySQL 5.7.35-38.

DBOptions::track_and_verify_wals_in_manifest for RocksDB

If true, the log numbers and sizes of the synced WALs are tracked in Manifest, then, during a DB recovery, if a synced WAL is missing from the disk, or the size of the WAL does not match the recorded size in Manifest, an error is reported adn the recovery is aborted.

Note

This option does not work with a secondary instance.

`rocksdb_two_write_queues`

Option	Description
Command-line	--rocksdb-track-and-verify-wals-in-manifest
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.21-20. When enabled this variable allows/encourages threads that are using two-phase commit to `prepare` in parallel.

`rocksdb_unsafe_for_binlog`

Option	Description
Command-line	--rocksdb-unsafe-for-binlog
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to allow statement-based binary logging which may break consistency. Disabled by default.

`rocksdb_update_cf_options`

Option	Description
Command-line	--rocksdb-update-cf-options
Dynamic	No
Scope	Global
Data type	String
Default	

The variable has been implemented in Percona Server 5.7.19-17. Specifies option updates for each column family. Empty by default.

`rocksdb_use_adaptive_mutex`

Option	Description
Command-line	--rocksdb-use-adaptive-mutex
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to use an adaptive mutex which spins in user space before resorting to the kernel. Disabled by default.

`rocksdb_use_default_sk_cf`

Option	Description
Command-line	--rocksdb-use-default-sk-cf
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server for MySQL 5.7.32-35. Use `default_sk` column family for secondary keys.

`rocksdb_use_direct_io_for_flush_and_compaction`

Option	Description
Command-line	--rocksdb-use-direct-io-for-flush-and-compaction
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to write to data files directly, without caches or buffers. Disabled by default.

`rocksdb_use_direct_reads`

Option	Description
Command-line	--rocksdb-use-direct-reads
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to read data files directly, without caches or buffers. Disabled by default. If you enable this, make sure that rocksdb_allow_mmap_reads is disabled.

`rocksdb_use_fsync`

Option	Description
Command-line	--rocksdb-use-fsync
Dynamic	No
Scope	Global
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether MyRocks should use `fsync` instead of `fdatasync` when requesting a sync of a data file. Disabled by default.

`rocksdb_validate_tables`

Option	Description
Command-line	--rocksdb-validate-tables
Dynamic	No
Scope	Global
Data type	Numeric
Default	1

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to verify that MySQL `.frm` files match MyRocks tables.

- `0`: do not verify.
- `1`: verify and fail on error (default).
- `2`: verify and continue with the error.

`rocksdb_verify_row_debug_checksums`

Option	Description
Command-line	--rocksdb-verify-row-debug-checksums
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to verify checksums when reading index or table records. Disabled by default.

`rocksdb_wal_bytes_per_sync`

Option	Description
Command-line	--rocksdb-wal-bytes-per-sync
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. The variable has been changed to dynamic in Percona Server 5.7.21-20. Specifies how often should the OS sync WAL (write-ahead log) files to the disk as they are being written, asynchronously, in the background. This operation can be used to smooth out write I/O over time. The default value is `0`, meaning that files are never synced. The allowed range is up to `18446744073709551615`.

`rocksdb_wal_dir`

Option	Description
Command-line	--rocksdb-wal-dir
Dynamic	No
Scope	Global
Data type	String
Default	

The variable has been implemented in Percona Server 5.7.19-17. Specifies the path to the directory where MyRocks stores WAL files.

`rocksdb_wal_recovery_mode`

Option	Description
Command-line	--rocksdb-wal-recovery-mode
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	2

Note

In version 5.7.31-34 and later, the default is changed from `1` to `2`.

The variable has been implemented in Percona Server 5.7.19-17. Specifies the level of tolerance when recovering write-ahead logs (WAL) files after a system crash.

The following are the options:

- `0`: if the last WAL entry is corrupted, truncate the entry and either start the server normally or refuse to start.
- `1`: if a WAL entry is corrupted, the server fails to start and does not recover from the crash.
- `2` (default): if a corrupted WAL entry is detected, truncate all entries after the detected corrupted entry. You can select this setting for replication replicas.
- `3`: If a corrupted WAL entry is detected, skip only the corrupted entry and continue the apply WAL entries. This option can be dangerous.

`rocksdb_wal_size_limit_mb`

Option	Description
Command-line	--rocksdb-wal-size-limit-mb
Dynamic	No
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the maximum size of all WAL files in megabytes before attempting to flush memtables and delete the oldest files. The default value is `0` (never rotated). The allowed range is up to `9223372036854775807`.

`rocksdb_wal_ttl_seconds`

Option	Description
Command-line	--rocksdb-wal-ttl-seconds
Dynamic	No
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.19-17. Specifies the timeout in seconds before deleting archived WAL files. The default is `0` (WAL files are not archived). The allowed range is up to `9223372036854775807`.

`rocksdb_whole_key_filtering`

Option	Description
Command-line	--rocksdb-whole-key-filtering
Dynamic	No
Scope	Global
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether the bloomfilter should use the whole key for filtering instead of just the prefix. Enabled by default. Make sure that lookups use the whole key for matching.

`rocksdb_write_batch_max_bytes`

Option	Description
Command-line	--rocksdb-write-batch-max-bytes
Dynamic	Yes
Scope	Global
Data type	Numeric
Default	0

The variable has been implemented in Percona Server 5.7.20-18. Specifies the maximum size of a RocksDB write batch in bytes. `0` means no limit. In case the user exceeds the limit following error is shown:

```
ERROR HY000: Status error 10 received from RocksDB: Operation aborted: Memory
limit reached.
```

`rocksdb_write_disable_wal`

Option	Description
Command-line	--rocksdb-write-disable-wal
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Lets you temporarily disable the writes to WAL files, which can be useful for bulk loading.

`rocksdb_write_ignore_missing_column_families`

Option	Description
Command-line	--rocksdb-write-ignore-missing-column-families
Dynamic	Yes
Scope	Global, Session
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.19-17. Specifies whether to ignore writes to column families that do not exist. Disabled by default (writes to non-existent column families are not ignored).

`rocksdb_write_policy`

Option	Description
Command-line	--rocksdb-write-policy
Dynamic	No
Scope	Global
Data type	String
Default	write_committed

The variable has been implemented in Percona Server 5.7.23-23. Specifies when two-phase commit data are actually written into the database. Allowed values are `write_committed`, `write_prepared`, and `write_unprepared`.

Default value is `write_committed` which means data are written at commit time. If the value is set to `write_prepared`, then data are written after the prepare phase of a two-phase transaction. If the value is set to `write_unprepared`, then data are written before the prepare phase.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-01-23

Download PDF

10.6 MyRocks status variables

MyRocks status variables provide details about the inner workings of the 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 'rocksdb%';
```

10.6.1 List of status variables

The following global status variables are available:

Name	Var Type
rocksdb_rows_deleted	Numeric
rocksdb_rows_inserted	Numeric
rocksdb_rows_read	Numeric
rocksdb_rows_updated	Numeric
rocksdb_rows_expired	Numeric
rocksdb_system_rows_deleted	Numeric
rocksdb_system_rows_inserted	Numeric
rocksdb_system_rows_read	Numeric
rocksdb_system_rows_updated	Numeric
rocksdb_memtable_total	Numeric
rocksdb_memtable_unflushed	Numeric
rocksdb_queries_point	Numeric
rocksdb_queries_range	Numeric
rocksdb_covered_secondary_key_lookups	Numeric
rocksdb_additional_compactions_trigger	Numeric
rocksdb_block_cache_add	Numeric
rocksdb_block_cache_add_failures	Numeric
rocksdb_block_cache_bytes_read	Numeric
rocksdb_block_cache_bytes_write	Numeric
rocksdb_block_cache_data_add	Numeric
rocksdb_block_cache_data_bytes_insert	Numeric
rocksdb_block_cache_data_hit	Numeric
rocksdb_block_cache_data_miss	Numeric
rocksdb_block_cache_filter_add	Numeric
rocksdb_block_cache_filter_bytes_evict	Numeric
rocksdb_block_cache_filter_bytes_insert	Numeric
rocksdb_block_cache_filter_hit	Numeric
rocksdb_block_cache_filter_miss	Numeric
rocksdb_block_cache_hit	Numeric
rocksdb_block_cache_index_add	Numeric
rocksdb_block_cache_index_bytes_evict	Numeric
rocksdb_block_cache_index_bytes_insert	Numeric
rocksdb_block_cache_index_hit	Numeric
rocksdb_block_cache_index_miss	Numeric
rocksdb_block_cache_miss	Numeric

Name	Var Type
rocksdb_block_cache_compressed_hit	Numeric
rocksdb_block_cache_compressed_miss	Numeric
rocksdb_bloom_filter_prefix_checked	Numeric
rocksdb_bloom_filter_prefix_useful	Numeric
rocksdb_bloom_filter_useful	Numeric
rocksdb_bytes_read	Numeric
rocksdb_bytes_written	Numeric
rocksdb_compact_read_bytes	Numeric
rocksdb_compact_write_bytes	Numeric
rocksdb_compaction_key_drop_new	Numeric
rocksdb_compaction_key_drop_obsolete	Numeric
rocksdb_compaction_key_drop_user	Numeric
rocksdb_flush_write_bytes	Numeric
rocksdb_get_hit_l0	Numeric
rocksdb_get_hit_l1	Numeric
rocksdb_get_hit_l2_and_up	Numeric
rocksdb_get_updates_since_calls	Numeric
rocksdb_iter_bytes_read	Numeric
rocksdb_memtable_hit	Numeric
rocksdb_memtable_miss	Numeric
rocksdb_no_file_closes	Numeric
rocksdb_no_file_errors	Numeric
rocksdb_no_file_opens	Numeric
rocksdb_num_iterators	Numeric
rocksdb_number_block_not_compressed	Numeric
rocksdb_number_db_next	Numeric
rocksdb_number_db_next_found	Numeric
rocksdb_number_db_prev	Numeric
rocksdb_number_db_prev_found	Numeric
rocksdb_number_db_seek	Numeric
rocksdb_number_db_seek_found	Numeric
rocksdb_number_deletes_filtered	Numeric
rocksdb_number_keys_read	Numeric
rocksdb_number_keys_updated	Numeric
rocksdb_number_keys_written	Numeric

Name	Var Type
rocksdb_number_merge_failures	Numeric
rocksdb_number_multiget_bytes_read	Numeric
rocksdb_number_multiget_get	Numeric
rocksdb_number_multiget_keys_read	Numeric
rocksdb_number_reseeks_iteration	Numeric
rocksdb_number_sst_entry_delete	Numeric
rocksdb_number_sst_entry_merge	Numeric
rocksdb_number_sst_entry_other	Numeric
rocksdb_number_sst_entry_put	Numeric
rocksdb_number_sst_entry_singledelete	Numeric
rocksdb_number_stat_computes	Numeric
rocksdb_number_superversion_acquires	Numeric
rocksdb_number_superversion_cleanups	Numeric
rocksdb_number_superversion_releases	Numeric
rocksdb_rate_limit_delay_millis	Numeric
rocksdb_row_lock_deadlocks	Numeric
rocksdb_row_lock_wait_timeouts	Numeric
rocksdb_snapshot_conflict_errors	Numeric
rocksdb_stall_l0_file_count_limit_slowdowns	Numeric
rocksdb_stall_locked_l0_file_count_limit_slowdowns	Numeric
rocksdb_stall_l0_file_count_limit_stops	Numeric
rocksdb_stall_locked_l0_file_count_limit_stops	Numeric
rocksdb_stall_pending_compaction_limit_stops	Numeric
rocksdb_stall_pending_compaction_limit_slowdowns	Numeric
rocksdb_stall_memtable_limit_stops	Numeric
rocksdb_stall_memtable_limit_slowdowns	Numeric
rocksdb_stall_total_stops	Numeric
rocksdb_stall_total_slowdowns	Numeric
rocksdb_stall_micros	Numeric
rocksdb_wal_bytes	Numeric
rocksdb_wal_group_syncs	Numeric
rocksdb_wal_synced	Numeric
rocksdb_write_other	Numeric
rocksdb_write_self	Numeric
rocksdb_write_timedout	Numeric

Name	Var Type
rocksdb_write_wal	Numeric

`rocksdb_rows_deleted`

This variable shows the number of rows that were deleted from MyRocks tables.

`rocksdb_rows_inserted`

This variable shows the number of rows that were inserted into MyRocks tables.

`rocksdb_rows_read`

This variable shows the number of rows that were read from MyRocks tables.

`rocksdb_rows_updated`

This variable shows the number of rows that were updated in MyRocks tables.

`rocksdb_rows_expired`

This variable shows the number of expired rows in MyRocks tables.

`rocksdb_rows_filtered`

This variable shows the number of rows that were filtered out for TTL in MyRocks tables.

`rocksdb_system_rows_deleted`

This variable shows the number of rows that were deleted from MyRocks system tables.

`rocksdb_system_rows_inserted`

This variable shows the number of rows that were inserted into MyRocks system tables.

`ocksdb_system_rows_read`

This variable shows the number of rows that were read from MyRocks system tables.

`rocksdb_system_rows_updated`

This variable shows the number of rows that were updated in MyRocks system tables.

`rocksdb_memtable_total`

This variable shows the memory usage, in bytes, of all memtables.

`rocksdb_memtable_unflushed`

This variable shows the memory usage, in bytes, of all unflushed memtables.

`rocksdb_queries_point`

This variable shows the number of single row queries.

`rocksdb_queries_range`

This variable shows the number of multi/range row queries.

`rocksdb_covered_secondary_key_lookups`

This variable shows the number of lookups via the secondary index that returned all fields requested directly from the secondary index.

`rocksdb_additional_compactions_trigger`

This variable shows the number of triggered additional compactions. MyRocks triggers an additional compaction if (number of deletions / number of entries) > (rocksdb_compaction_sequential_deletes / rocksdb_compaction_sequential_deletes_window) in the SST file.

`rocksdb_block_cache_add`

This variable shows the number of blocks added to block cache.

`rocksdb_block_cache_add_failures`

This variable shows the number of failures when adding blocks to block cache.

`rocksdb_block_cache_bytes_read`

This variable shows the number of bytes read from cache.

`rocksdb_block_cache_bytes_write`

This variable shows the number of bytes written into cache.

`rocksdb_block_cache_data_add`

This variable shows the number of data blocks added to block cache.

`rocksdb_block_cache_data_bytes_insert`

This variable shows the number of bytes of data blocks inserted into cache.

`rocksdb_block_cache_data_hit`

This variable shows the number of cache hits when accessing the data block from the block cache.

`rocksdb_block_cache_data_miss`

This variable shows the number of cache misses when accessing the data block from the block cache.

`rocksdb_block_cache_filter_add`

This variable shows the number of filter blocks added to block cache.

`rocksdb_block_cache_filter_bytes_evict`

This variable shows the number of bytes of bloom filter blocks removed from cache.

`rocksdb_block_cache_filter_bytes_insert`

This variable shows the number of bytes of bloom filter blocks inserted into cache.

`rocksdb_block_cache_filter_hit`

This variable shows the number of times cache hit when accessing filter block from block cache.

`rocksdb_block_cache_filter_miss`

This variable shows the number of times cache miss when accessing filter block from block cache.

`rocksdb_block_cache_hit`

This variable shows the total number of block cache hits.

`rocksdb_block_cache_index_add`

This variable shows the number of index blocks added to block cache.

`rocksdb_block_cache_index_bytes_evict`

This variable shows the number of bytes of index block erased from cache.

`rocksdb_block_cache_index_bytes_insert`

This variable shows the number of bytes of index blocks inserted into cache.

`rocksdb_block_cache_index_hit`

This variable shows the total number of block cache index hits.

`rocksdb_block_cache_index_miss`

This variable shows the number of times cache hit when accessing index block from block cache.

`rocksdb_block_cache_miss`

This variable shows the total number of block cache misses.

`rocksdb_block_cache_compressed_hit`

This variable shows the number of hits in the compressed block cache.

`rocksdb_block_cache_compressed_miss`

This variable shows the number of misses in the compressed block cache.

`rocksdb_bloom_filter_prefix_checked`

This variable shows the number of times bloom was checked before creating iterator on a file.

`rocksdb_bloom_filter_prefix_useful`

This variable shows the number of times the check was useful in avoiding iterator creation (and thus likely IOPs).

`rocksdb_bloom_filter_useful`

This variable shows the number of times bloom filter has avoided file reads.

`rocksdb_bytes_read`

This variable shows the total number of uncompressed bytes read. It could be either from memtables, cache, or table files.

`rocksdb_bytes_written`

This variable shows the total number of uncompressed bytes written.

`rocksdb_compact_read_bytes`

This variable shows the number of bytes read during compaction

`rocksdb_compact_write_bytes`

This variable shows the number of bytes written during compaction.

`rocksdb_compaction_key_drop_new`

This variable shows the number of key drops during compaction because it was overwritten with a newer value.

`rocksdb_compaction_key_drop_obsolete`

This variable shows the number of key drops during compaction because it was obsolete.

`rocksdb_compaction_key_drop_user`

This variable shows the number of key drops during compaction because user compaction function has dropped the key.

`rocksdb_flush_write_bytes`

This variable shows the number of bytes written during flush.

`rocksdb_get_hit_l0`

This variable shows the number of `Get()` queries served by L0.

`rocksdb_get_hit_l1`

This variable shows the number of `Get()` queries served by L1.

`rocksdb_get_hit_l2_and_up`

This variable shows the number of `Get()` queries served by L2 and up.

`rocksdb_get_updates_since_calls`

This variable shows the number of calls to `GetUpdatesSince` function. Useful to keep track of transaction log iterator refreshes

`rocksdb_iter_bytes_read`

This variable shows the number of uncompressed bytes read from an iterator. It includes size of key and value.

`rocksdb_memtable_hit`

This variable shows the number of memtable hits.

`rocksdb_memtable_miss`

This variable shows the number of memtable misses.

`rocksdb_no_file_closes`

This variable shows the number of time file were closed.

`rocksdb_no_file_errors`

This variable shows number of errors trying to read in data from an sst file.

`rocksdb_no_file_opens`

This variable shows the number of time file were opened.

`rocksdb_num_iterators`

This variable shows the number of currently open iterators.

`rocksdb_number_block_not_compressed`

This variable shows the number of uncompressed blocks.

`rocksdb_number_db_next`

This variable shows the number of calls to `next`.

`rocksdb_number_db_next_found`

This variable shows the number of calls to `next` that returned data.

`rocksdb_number_db_prev`

This variable shows the number of calls to `prev`.

`rocksdb_number_db_prev_found`

This variable shows the number of calls to `prev` that returned data.

`rocksdb_number_db_seek`

This variable shows the number of calls to `seek`.

`rocksdb_number_db_seek_found`

This variable shows the number of calls to `seek` that returned data.

`rocksdb_number_deletes_filtered`

This variable shows the number of deleted records that were not required to be written to storage because key did not exist.

`rocksdb_number_keys_read`

This variable shows the number of keys read.

`rocksdb_number_keys_updated`

This variable shows the number of keys updated, if inplace update is enabled.

`rocksdb_number_keys_written`

This variable shows the number of keys written to the database.

`rocksdb_number_merge_failures`

This variable shows the number of failures performing merge operator actions in RocksDB.

`rocksdb_number_multiget_bytes_read`

This variable shows the number of bytes read during RocksDB `MultiGet()` calls.

`rocksdb_number_multiget_get`

This variable shows the number `MultiGet()` requests to RocksDB.

`rocksdb_number_multiget_keys_read`

This variable shows the keys read via `MultiGet()`.

`rocksdb_number_reseeks_iteration`

This variable shows the number of times reseek happened inside an iteration to skip over large number of keys with same userkey.

`rocksdb_number_sst_entry_delete`

This variable shows the total number of delete markers written by MyRocks.

`rocksdb_number_sst_entry_merge`

This variable shows the total number of merge keys written by MyRocks.

`rocksdb_number_sst_entry_other`

This variable shows the total number of non-delete, non-merge, non-put keys written by MyRocks.

`rocksdb_number_sst_entry_put`

This variable shows the total number of put keys written by MyRocks.

`rocksdb_number_sst_entry_singledelete`

This variable shows the total number of single delete keys written by MyRocks.

`rocksdb_number_stat_computes`

This variable was removed in *Percona Server for MySQL* Percona Server 5.7.23-23.

`rocksdb_number_superversion_acquires`

This variable shows the number of times the superversion structure has been acquired in RocksDB, this is used for tracking all of the files for the database.

`rocksdb_number_superversion_cleanups`

`rocksdb_number_superversion_releases`

`rocksdb_rate_limit_delay_millis`

This variable was removed in *Percona Server for MySQL* Percona Server 5.7.23-23.

`rocksdb_row_lock_deadlocks`

This variable shows the total number of deadlocks that have been detected since the instance was started.

`rocksdb_row_lock_wait_timeouts`

This variable shows the total number of row lock wait timeouts that have been detected since the instance was started.

`rocksdb_snapshot_conflict_errors`

This variable shows the number of snapshot conflict errors occurring during write transactions that forces the transaction to rollback.

`rocksdb_stall_l0_file_count_limit_slowdowns`

This variable shows the slowdowns in write due to L0 being close to full.

`rocksdb_stall_locked_l0_file_count_limit_slowdowns`

This variable shows the slowdowns in write due to L0 being close to full and compaction for L0 is already in progress.

`rocksdb_stall_l0_file_count_limit_stops`

This variable shows the stalls in write due to L0 being full.

`rocksdb_stall_locked_l0_file_count_limit_stops`

This variable shows the stalls in write due to L0 being full and compaction for L0 is already in progress.

`rocksdb_stall_pending_compaction_limit_stops`

This variable shows the stalls in write due to hitting limits set for max number of pending compaction bytes.

`rocksdb_stall_pending_compaction_limit_slowdowns`

This variable shows the slowdowns in write due to getting close to limits set for max number of pending compaction bytes.

`rocksdb_stall_memtable_limit_stops`

This variable shows the stalls in write due to hitting max number of `memTables` allowed.

`rocksdb_stall_memtable_limit_slowdowns`

This variable shows the slowdowns in writes due to getting close to max number of memtables allowed.

`rocksdb_stall_total_stops`

This variable shows the total number of write stalls.

`rocksdb_stall_total_slowdowns`

This variable shows the total number of write slowdowns.

`rocksdb_stall_micros`

This variable shows how long (in microseconds) the writer had to wait for compaction or flush to finish.

`rocksdb_wal_bytes`

This variables shows the number of bytes written to WAL.

`rocksdb_wal_group_syncs`

This variable shows the number of group commit WAL file syncs that have occurred.

`rocksdb_wal_synced`

This variable shows the number of times WAL sync was done.

`rocksdb_write_other`

This variable shows the number of writes processed by another thread.

`rocksdb_write_self`

This variable shows the number of writes that were processed by a requesting thread.

`rocksdb_write_timedout`

This variable shows the number of writes ending up with timed-out.

`rocksdb_write_wal`

This variable shows the number of Write calls that request WAL.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-11-03

Download PDF

10.7 Gap locks detection

Percona Server for MySQL has implemented Gap locks detection in Percona Server for MySQL 5.7.18-14 based on a Facebook MySQL patch.

If a transactional storage engine does not support gap locks (for example MyRocks) and a gap lock is being attempted while the transaction isolation level is either `REPEATABLE READ` or `SERIALIZABLE`, the following SQL error will be returned to the client and no actual gap lock will be taken on the effected rows.

```
ERROR HY000: Using Gap Lock without full unique key in multi-table or multi-statement
transactions is not allowed. You need to either rewrite queries to use all unique key
columns in WHERE equal conditions, or rewrite to single-table, single-statement transaction.
```

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

10.8 Data Loading

By default, MyRocks configurations are optimized for short transactions, and not for data loading. MyRocks has a couple of special session variables to speed up data loading dramatically.

10.8.1 Sorted bulk loading

If your data is guaranteed to be loaded in primary key order, then this method is recommended. This method works by dropping any secondary keys first, loading data into your table in primary key order, and then restoring the secondary keys via Fast Secondary Index Creation.

Creating Secondary Indexes

When loading data into empty tables, it is highly recommended to drop all secondary indexes first, then loading data, and adding all secondary indexes after finishing loading data. MyRocks has a feature called `Fast Secondary Index Creation`. Fast Secondary Index Creation is automatically used when executing `CREATE INDEX` or `ALTER TABLE ... ADD INDEX`. With Fast Secondary Index Creation, the secondary index entries are directly written to bottommost RocksDB levels and bypassing compaction. This significantly reduces total write volume and CPU time for decompressing and compressing data on higher levels.

Loading Data

As described above, loading data is highly recommended for tables with primary key only (no secondary keys), with all secondary indexes added after loading data.

When loading data into MyRocks tables, there are two recommended session variables:

```
SET session sql_log_bin=0;
SET session rocksdb_bulk_load=1;
```

When converting from large MyISAM/InnoDB tables, either by using the `ALTER` or `INSERT INTO SELECT` statements it's recommended that you create MyRocks tables as below (in case the table is sufficiently big it will cause the server to consume all the memory and then be terminated by the OOM killer):

```
SET session sql_log_bin=0;
SET session rocksdb_bulk_load=1;
ALTER TABLE large_myisam_table ENGINE=RocksDB;
SET session rocksdb_bulk_load=0;
```

Using sql_log_bin=0 avoids writing to binary logs.

With rocksdb_bulk_load set to `1`, MyRocks enters special mode to write all inserts into bottommost RocksDB levels, and skips writing data into MemTable and the following compactions. This is very efficient way to load data.

The rocksdb_bulk_load mode operates with a few conditions:

- None of the data being bulk loaded can overlap with existing data in the table. The easiest way to ensure this is to always bulk load into an empty table, but the mode will allow loading some data into the table, doing other operations, and then returning and bulk loading addition data if there is no overlap between what is being loaded and what already exists.
- The data may not be visible until bulk load mode is ended (i.e. the rocksdb_bulk_load is set to zero again). The method that is used is building up SST files which will later be added as-is to the database. Until a particular SST has been added the data will not be visible to the rest of the system, thus issuing a `SELECT` on the table currently being bulk loaded will only show older data and will likely not show the most recently added rows. Ending the bulk load mode will cause the most recent SST file to be added. When bulk loading multiple tables, starting a new table will trigger the code to add the most recent SST file to the system – as a result, it is inadvisable to interleave `INSERT` statements to two or more tables during bulk load mode.

By default, the rocksdb_bulk_load mode expects all data be inserted in primary key order (or reversed order). If the data is in the reverse order (i.e. the data is descending on a normally ordered primary key or is ascending on a reverse ordered primary key), the rows are cached in chunks to switch the order to match the expected order.

Inserting one or more rows out of order will result in an error and may result in some of the data being inserted in the table and some not. To resolve the problem, one can either fix the data order of the insert, truncate the table, and restart.

10.8.2 Unsorted bulk loading

If your data is not ordered in primary key order, then this method is recommended. With this method, secondary keys do not need to be dropped and restored. However, writing to the primary key no longer goes directly to SST files, and are written to temporary files for sorted first, so there is extra cost to this method.

To allow for loading unsorted data:

```
SET session sql_log_bin=0;
SET session rocksdb_bulk_load_allow_unsorted=1;
SET session rocksdb_bulk_load=1;
...
SET session rocksdb_bulk_load=0;
SET session rocksdb_bulk_load_allow_unsorted=0;
```

Note that rocksdb_bulk_load_allow_unsorted can only be changed when rocksdb_bulk_load is disabled (set to `0`). In this case, all input data will go through an intermediate step that writes the rows to temporary SST files, sorts them rows in the primary key order, and then writes to final SST files in the correct order.

10.8.3 Other Approaches

If rocksdb_commit_in_the_middle is enabled, MyRocks implicitly commits every rocksdb_bulk_load_size records (default is `1,000`) in the middle of your transaction. If your data loading fails in the middle of the statement (`LOAD DATA` or bulk `INSERT`), rows are not entirely rolled back, but some of rows are stored in the table. To restart data loading, you'll need to truncate the table and loading data again.

Warning

If you are loading large data without enabling rocksdb_bulk_load or rocksdb_commit_in_the_middle, please make sure transaction size is small enough. All modifications of the ongoing transactions are kept in memory.

10.8.4 Other Reading

- Data Loading - this document has been used as a source for writing this documentation
- ALTER TABLE ... ENGINE=ROCKSDB uses too much memory

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

11. Performance Improvements

Download PDF

11.1 Multiple page asynchronous I/O requests

The I/O unit size in InnoDB is only one page, even if the server is 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 are more efficient. 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 an 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.

11.1.1 Version Specific Information

- Percona Server 5.7.20-18: Feature ported from the *Facebook MySQL* patch.

11.1.2 Status Variables

`Innodb_buffered_aio_submitted`

Implemented in Percona Server 5.7.20-18.

Option	Description
Data type	Numeric
Scope	Global

This variable shows the number of submitted buffered asynchronous I/O requests.

11.1.3 See also

Optimizing full table scans in InnoDB

Bug #68659 InnoDB Linux native aio should submit more i/o requests at once

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

11.2 Query Cache Enhancements

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

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

11.2.1 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):

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

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

```
> mysql> SHOW PROCESSLIST;
```

The output is similar to the following:

```
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
```

11.2.2 Ignoring comments

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

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

11.2.3 System Variables

`query_cache_strip_comments`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	Off

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

Other Reading

- MySQL general thread states
- Query cache freezes

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

11.3 Limiting the Estimation of Records in a Query

Availability: This feature is ***technical preview** quality.

This page describes an alternative when running queries against a large number of table partitions. When a query runs, InnoDB estimates the records in each partition. This process can result in more pages read and more disk I/O, if the buffer pool must fetch the pages from disk. This process increases the query time if there are a large number of partitions.

The addition of two variables make it possible to override records_in_range which effectively bypasses the process.

> **Warning**
>
> The use of these variables may result in improper index selection by the optimizer.

`innodb_records_in_range`

Option	Description
Command-line	`--innodb-records-in-range`
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	0

Availability: This feature is **technical preview** quality.

The variable provides a method to limit the number of records estimated for a query.

```
mysql> SET @@GLOBAL.innodb_records_in_range=100;
100
```

11.3.1 `innodb_force_index_records_in_range`

Option	Description
Command-line	--innodb-force-index-records-in-range
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	0

Availability: This feature is **technical preview** quality.

This variable provides a method to override the records_in_range result when a FORCE INDEX is used in a query.

```
mysql> SET @@GLOBAL.innodb_force_index_records_in_range=100;
100
```

11.3.2 Using the favor_range_scan optimizer switch

Availability: The feature is **technical preview** quality.

In specific scenarios, the optimizer chooses to scan a table instead of using a range scan. The conditions are the following:

- Table with an extremely large number of rows
- Compound primary keys made of two or more columns
- WHERE clause contains multiple range conditions

The optimizer_switch controls the behavior of the optimizer. The favor_range_scan switch arbitrarily lowers the cost of a range scan by a factor of 10.

The available values are:

- ON
- OFF (Default)
- DEFAULT

```
mysql> SET optimizer_switch='favor_range_scan=on';
```

Download PDF

11.4 Improved NUMA support

In cases where the buffer pool memory allocation is bigger than the size of the node, the system starts swapping already allocated memory, even if memory is available on another node. This happens if the default `NUMA` memory allocation policy is selected. In that case, the system favors one node more than another, which causes the node to run out of memory. If the allocation policy is interleaving, the memory is allocated in a round-robin fashion over the available node. This method uses the upstream innodb_numa_interleave. This feature extends the upstream implementation by implementing the flush_caches variable.

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

11.4.1 Version Specific Information

Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6

Percona Server 5.7.22-22: Feature reverted from the upstream implementation back to the one ported from *Percona Server for MySQL* 5.6, in which innodb_numa_interleave variable not only enables NUMA memory interleaving at InnoDB buffer pool allocation but allocates buffer pool with MAP_POPULATE, forcing interleaved allocation at the buffer pool initialization time.

11.4.2 Command-line Options for mysqld_safe

`flush_caches`

Option	Description
Command-line	Yes
Config file	Yes
Location	mysqld_safe
Dynamic	No
Data type	Boolean
Default	0 (OFF)
Range	0/1

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.

See also

The MySQL “swap insanity” problem and the effects of the NUMA architecture

A brief update on NUMA and MySQL

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

11.5 Thread Pool

Servers continually execute queries from multiple clients. In *MySQL*, for each connection query, the server creates a thread, processes the query, and then destroys the thread. This method can have disadvantages because the server must consume resources to create, process, and destroy the thread. Therefore, when the number of connections grows, the server performance drops. Too many active threads impact performance because of context switching, and thread contention.

A thread pool is distinct from connection pooling. A thread pool has the following advantages:

- Limits the number of threads running on the server
- Minimizes wasting resources by creating and then destroying threads

This feature ensures that multiple connections using a thread pool will not cause the server to churn through resources or cause a server exit when the server runs out of memory. A thread pool reuses threads and is most efficient for the short queries associated with transactions.

To enable the *thread-pool* feature, the 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 in System Variables.

11.5.1 Priority connection scheduling

Even though thread pool puts a limit on the number of concurrently running queries, the number of open transactions may remain high, because connections with already started transactions are put to the end of the queue. Higher number of open transactions has a number of implications on the currently running queries. To improve the performance new thread_pool_high_prio_tickets variable has been introduced.

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

- The connection has an open transaction in the server.
- 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 the vast majority of cases.

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

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

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 thread_pool_max_threads value. If the connection executing the transaction which is holding the lock has managed to enter the *thread-pool* by then, we get a large (depending on the thread_pool_max_threads value) number of concurrently running threads, and thus, suboptimal performance as a result. Otherwise, we get a deadlock as no more threads can be created to process those 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.

11.5.2 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.

11.5.3 Version Specific Information

- Percona Server for MySQL 5.7.10-1: `Thread Pool` feature ported from *Percona Server for MySQL* 5.6.

11.5.4 System Variables

`thread_handling`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	String
Default	one-thread-per-connection

This variable defines how the server handles threads for connections from the client.

Values	Description
one-thread-per-connection	One thread handles all requests for a connection
pool-of-threads	A thread pool handles requests for all connections
no-threads	A single thread for all connections for debugging mode

`thread_pool_idle_timeout`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	60 (seconds)

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

`thread_pool_high_prio_mode`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global, Session
Dynamic	Yes
Data type	String
Default	transactions
Allowed values	transactions, statements, none

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

The following values are allowed:

- `transactions` (the default). In this mode, only statements from already started transactions may go into the high priority queue depending on the number of high priority tickets currently available in a connection (see thread_pool_high_prio_tickets).
- `statements`. In this mode, all individual statements go into the high priority queue, regardless of connection's transactional state and the number of available high priority tickets. This value can be used to prioritize `AUTOCOMMIT` transactions or other kinds of statements such as administrative ones for specific connections. Note that setting this value globally essentially disables high priority scheduling, since in this case all statements from all connections will use a single queue (the high priority one)
- `none`. This mode disables high priority queue for a connection. Some connections (e.g. monitoring) may be insensitive to execution latency and/or never allocate any server resources that would otherwise impact performance in other connections and thus, do not really require high priority scheduling. Note that setting thread_pool_high_prio_mode to `none` globally has essentially the same effect as setting it to `statements` globally: all connections will always use a single queue (the low priority one in this case).

`thread_pool_high_prio_tickets`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global, Session
Dynamic	Yes
Data type	Numeric
Default	4294967295
This variable controls the high priority queue policy. Each new connection is assigned this many tickets to enter the high priority queue. Setting this variable to `0` disables the high priority queue.	

`thread_pool_max_threads`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	100000
This variable can be used to limit the maximum number of threads in the	
pool. Once this number is reached no new threads will be created.	

`thread_pool_oversubscribe`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	3
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.	

`thread_pool_size`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	Number of processors
This variable can be used to define the number of threads that can use the CPU	
at the same time.	

`thread_pool_stall_limit`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Numeric
Default	500 (ms)
The number of milliseconds before a running thread is considered stalled. When	
this limit is reached thread pool will wake up or create another thread. This is	
being used to prevent a long-running query from monopolizing the pool.	

`extra_port`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Numeric
Default	0

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
```

`extra_max_connections`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	1

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

11.5.5 Status Variables

`Threadpool_idle_threads`

Option	Description
Data type	Numeric
Scope	Global

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

`Threadpool_threads`

Option	Description
Data type	Numeric
Scope	Global

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

Note

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

11.5.6 Other Reading

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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

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

11.6.1 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.

This is addressed by delegating all the LRU flushes to the LRU manager thread, never attempting to evict a page or perform an LRU single page flush by a query thread, and introducing a backoff algorithm to reduce buffer pool free list mutex pressure on empty buffer pool free lists. This is controlled through a new system variable innodb_empty_free_list_algorithm.

11.6.2 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature partially ported from *Percona Server for MySQL* 5.6
- Percona Server for MySQL 5.7.10-3: Implemented support for multi-threaded LRU
- Percona Server for MySQL 5.7.11-4: Implemented support for parallel doublewrite buffer

`innodb_empty_free_list_algorithm`

Option	Description
Command-line	Yes
Config File	Yes
Scope	Global
Dynamic	Yes
Data type	legacy, backoff
Default	backoff

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

11.6.3 Multi-threaded LRU flusher

Percona Server for MySQL 5.7.10-3 has introduced a true multi-threaded LRU flushing. In this scheme, each buffer pool instance has its own dedicated LRU manager thread that is tasked with performing LRU flushes and evictions to refill the free list of that buffer pool instance. Existing multi-threaded flusher no longer does any LRU flushing and is tasked with flush list flushing only.

This has been done to address the shortcomings of the existing MySQL 5.7 multi-threaded flusher:

- All threads still synchronize on each coordinator thread iteration. If a particular flushing job is stuck on one of the worker threads, the rest will idle until the stuck one completes.
- The coordinator thread heuristics focus on flush list adaptive flushing without considering the state of free lists, which might be in need of urgent refill for a subset of buffer pool instances on a loaded server.
- LRU flushing is serialized with flush list flushing for each buffer pool instance, introducing the risk that the right flushing mode will not happen for a particular instance because it is being flushed in the other mode.

The following InnoDB metrics are no longer accounted, as their semantics do not make sense under the current LRU flushing design: `buffer_LRU_batch_flush_avg_time_slot`, `buffer_LRU_batch_flush_avg_pass`, `buffer_LRU_batch_flush_avg_time_thread`, `buffer_LRU_batch_flush_avg_time_est`.

The need for InnoDB recovery thread writer threads is also removed, consequently all associated code is deleted.

11.6.4 Parallel doublewrite buffer

The legacy doublewrite buffer is shared between all the buffer pool instances and all the flusher threads. It collects all the page write requests into a single buffer, and, when the buffer fills, writes it out to disk twice, blocking any new write requests until the writes complete. This becomes a bottleneck with increased flusher parallelism, limiting the effect of extra cleaner threads. In addition, single page flushes, if they are performed, are subject to above and also contend on the doublewrite mutex.

To address these issues *Percona Server for MySQL* Percona Server for MySQL 5.7.11-4 has introduced private doublewrite buffers for each buffer pool instance, for each batch flushing mode (LRU or flush list). For example, with four buffer pool instances, there will be eight doublewrite shards. Only one flusher thread can access any shard at a time, and each shard is added to and flushed completely independently of the rest. This does away with the mutex and the event wait does not block other threads from proceeding anymore, it only waits for the asynchronous I/O to complete. The only inter-thread synchronization is between the flusher thread and I/O completion threads.

The new doublewrite buffer is contained in a new file, where all the shards are contained, at different offsets. This file is created on startup, and removed on a clean shutdown. If it's found on a crashed instance startup, its contents are read and any torn pages are restored. If it's found on a clean instance startup, the server startup is aborted with an error message.

The location of the doublewrite file is governed by a new innodb_parallel_doublewrite_path global, read-only system variable. It defaults to `xb_doublewrite` in the data directory. The variable accepts both absolute and relative paths. In the latter case they are treated as relative to the data directory. The doublewrite file is not a tablespace from InnoDB internals point of view.

The legacy InnoDB doublewrite buffer in the system tablespace continues to address doublewrite needs of single page flushes, and they are free to use the whole of that buffer (128 pages by default) instead of the last eight pages as currently used. Note that single page flushes will not happen in *Percona Server for MySQL* unless innodb_empty_free_list_algorithm is set to `legacy` value.

The existing system tablespace is not touched in any way for this feature implementation, ensuring that cleanly-shutdown instances may be freely moved between different server flavors.

Interaction with innodb_flush_method

Regardless of innodb_flush_method setting, the parallel doublewrite file is opened with `O_DIRECT` flag to remove OS caching, then its access is further governed by the exact value set: if it's set to `O_DSYNC`, the parallel doublewrite is opened with `O_SYNC` flag too. Further, if it's one of `O_DSYNC`, `O_DIRECT_NO_FSYNC`, or

`ALL_O_DIRECT`, then the doublewrite file is not flushed after a batch of writes to it is completed. With other innodb_flush_method values the doublewrite buffer is flushed only if setting `O_DIRECT` has failed.

`innodb_parallel_doublewrite_path`

Option	Description
Command-line	Yes
Scope	Global
Dynamic	No
Data type	String
Default	xb_doublewrite

This variable is used to specify the location of the parallel doublewrite file. It accepts both absolute and relative paths. In the latter case they are treated as relative to the data directory.

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

`innodb_sched_priority_master`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Boolean

11.6.5 Other Reading

- Page cleaner thread tuning
- Bug #74637 - make dirty page flushing more adaptive
- Bug #67808 - in innodb engine, double write and multi-buffer pool instance reduce concurrency
- Bug #69232 - buf_dblwr->mutex can be split into two

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-10-14

Download PDF

11.7 Prefix Index Queries Optimization

Percona Server for MySQL 5.6 has ported the Prefix Index Queries Optimization feature from the 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.

11.7.1 Status Variables

`Innodb_secondary_index_triggered_cluster_reads`

Implemented in Percona Server for MySQL 5.7.18-14.

Option	Description
Data type	Numeric
Scope	Global

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

`Innodb_secondary_index_triggered_cluster_reads_avoided`

Implemented in Percona Server for MySQL 5.7.18-14.

Option	Description
Data type	Numeric
Scope	Global
This variable shows the number of times prefix optimization avoided triggering cluster lookup.	

12. Reliability Improvements

Download PDF

12.1 Too Many Connections Warning

This feature issues the warning `Too many connections` to the log, if log_error_verbosity is set to `2` or higher.

12.1.1 Version-Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

12.2 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 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.

12.2.1 Interacting with the innodb_force_recovery variable

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

If the innodb_force_recovery variable is set to a low value and you expect the server to crash, the server may continue to run due to a non-default value of the innodb_corrupt_table_action variable.

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

This feature adds a new system variable.

12.2.2 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6 5.6

12.2.3 System Variables

`innodb_corrupt_table_action`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	ULONG
Default	assert
Range	assert, warn, salvage

- 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". You must enable the innodb_file_per_table option.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

13. Scalability Improvements

Download PDF

13.1 Improved Buffer Pool Scalability

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

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

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

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

13.1.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from Percona Server for MySQL 5.6

13.1.2 Other Information

Detecting Mutex Contention

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

Under normal circumstances this section should look like this:

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

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

```
1 ----------
2 SEMAPHORES
3 ----------
4 OS WAIT ARRAY INFO: reservation count 36255, signal count 12675
5 --Thread 10607472 has waited at buf/buf0rea.c line 420 for 0.00 seconds the semaphore:
6 Mutex at 0x358068 created file buf/buf0buf.c line 597, lock var 0
7 waiters flag 0
```

```
8 --Thread 3488624 has waited at buf/buf0buf.c line 1177 for 0.00 seconds the semaphore:
9 Mutex at 0x358068 created file buf/buf0buf.c line 597, lock var 0
10 waiters flag 0
11 --Thread 6896496 has waited at btr/btr0cur.c line 442 for 0.00 seconds the semaphore:
12 S-lock on RW-latch at 0x8800244 created in file buf/buf0buf.c line 547
13 a writer (thread id 14879600) has reserved it in mode  exclusive
14 number of readers 0, waiters flag 1
15 Last time read locked in file btr/btr0cur.c line 442
16 Last time write locked in file buf/buf0buf.c line 1797
[...]
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.

Last update: 2022-09-27

Download PDF

13.2 Improved InnoDB I/O Scalability

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

13.2.1 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from Percona Server for MySQL 5.6

13.2.2 System Variables

`innodb_use_global_flush_log_at_trx_commit`

Option	Description
Command-line	Yes
Config File	Yes
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	True
Range	True/False

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 uses 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.

```
SET innodb_use_global_flush_log_at_trx_commit=1
```

`innodb_flush_method`

Option	Description
Command-line	Yes
Config File	Yes
Scope	Global
Dynamic	No
Data type	Enumeration
Default	fdatasync
Allowed values	fdatasync, O_DSYNC, O_DIRECT, O_DIRECT_NO_FSYNC, ALL_O_DIRECT

The variable was ported from Percona Server for MySQL 5.6 in Percona Server for MySQL 5.7.10-3. This is an existing MySQL 5.7 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 5.7 Reference Manual).

The following values are allowed:

- `fdatasync`: use `fsync()` to flush data, log, and parallel doublewrite files.
- `O_SYNC`: use `O_SYNC` to open and flush the log and parallel doublewrite files; use `fsync()` to flush the data files. Do not use `fsync()` to flush the parallel doublewrite file.
- `O_DIRECT`: use O_DIRECT to open the data files and `fsync()` system call to flush data, log, and parallel doublewrite 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**: On *ext4* filesystem, set `innodb_log_write_ahead_size`. This variable should match the filesystem's write-ahead block size and avoids the `unaligned AIO/DIO` warnings.

Status Variables

The following information has been added to `SHOW ENGINE INNODB STATUS` to confirm the checkpoint 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
...
```

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-05-08

14. Security Improvements

Download PDF

14.1 Data at Rest Encryption

Percona Server for MySQL enables data at rest encryption of the InnoDB (file-per-table) tablespace by encrypting the physical database files. The data is automatically encrypted prior to writing to storage and automatically decrypted when read. If unauthorized users access the data files, they cannot read the contents. Percona Server for MySQL 5.7 data at rest encryption is similar to the MySQL 5.7 data-at-rest encryption. Percona Server for MySQL 8.0 provides more encryption features and options which are not available in this version.

The following table lists the various features that are considered Generally Available (GA) or in **tech preview**. The **tech preview** features and variables are not recommended to be used in production. Features and variables marked as **deprecated** perform no action.

Feature	Status	GA Version	Tech Preview Version	Deprecated Version
Vault Keyring Plugin	Generally Available, supported	Percona Server 5.7.21-21		
Encrypting a File-Per-Table Tablespace	Generally Available, supported	Percona Server 5.7.21-21		
Encrypting a General Tablespace	Generally Available, supported	Percona Server 5.7.21-21		
Temporary file encryption	Generally Available, supported	Percona Server 5.7.22-22		
binlog_encrypt	Generally Available, supported	Percona Server 5.7.21-21		
InnoDB System Tablespace Encryption	Deprecated	Percona Server 5.7.23-24	Percona Server for MySQL 5.7.32-35	
Doublewrite buffer	Deprecated	Percona Server 5.7.23-24	Percona Server for MySQL 5.7.32-35	
InnoDB Undo Tablespace Encryption	Deprecated	Percona Server 5.7.23-24	Percona Server for MySQL 5.7.32-35	
Redo Log Encryption	Deprecated	Percona Server 5.7.23-24	Percona Server for MySQL 5.7.32-35	
Data Scrubbing	Deprecated	Percona Server 5.7.23-24	Percona Server for MySQL 5.7.32-35	

14.1.1 Architecture

The data at rest encryption uses a two-tier architecture with the following components:

Type	Description
Master key	The Master key is used to encrypt or decrypt the tablespace keys.
Tablespace key for each tablespace	The tablespace key encrypts the data pages and is written in the tablespace header.

When the server must access the data, the master key decrypts the tablespace key, the tablespace is decrypted and available for read or write operations.

The two separate keys architecture allows the master key to be rotated in a minimal operation. During the master key rotation, each tablespace key is re-encrypted with the new master key. Only the first page of the tablespace file (.ibd) is read and written during the rotation. An encrypted page is decrypted at the I/O layer, added to the buffer pool, and used to read and write the data. A buffer pool page is not encrypted. The I/O layer encrypts the page before the page is flushed to disk.

An encryption key in the tablespace header is required to encrypt or decrypt the tablespace. The Master key is stored in the keyring plugin.

Note

Percona XtraBackup version 2.4 supports the backup of encrypted general tablespaces.

14.2 Vault Keyring Plugin

To enable encryption, use either of the following plugins:

- keyring_file stores the keyring data locally in a flat file
- keyring_vault provides an interface for the database with a HashiCorp Vault server to store key and secure encryption keys.

Enable only one keyring plugin at a time. Enabling multiple keyring plugins is not supported and may result in data loss.

Note

The keyring_file plugin should not be used for regulatory compliance.

To install the selected plugin, follow the installing and uninstalling plugins instructions.

14.2.1 Loading the Keyring Plugin

Load the plugin at server startup with the early-plugin-load Option to enable the keyring. To make encrypted table recovery more efficient,load the plugin with the configuration file.

Run the following command to load the keyring_file plugin:

```
$ mysqld --early-plugin-load="keyring_file=keyring_file.so"
```

Note

To start a server with different early plugins to be loaded, the `--early-plugin-load` option can contain the plugin names in a double-quoted list with each plugin name separated by a semicolon. The use of double quotes ensures the semicolons do not create issues when the list is executed in a script.

To enable Master key vault encryption, the user must have SUPER privileges.

The following statements loads the keyring_vault plugin and the keyring_vault_config. The second statement provides the location to the keyring_vault configuration file.

```
[mysqld]
early-plugin-load="keyring_vault=keyring_vault.so"
loose-keyring_vault_config="/home/mysql/keyring_vault.conf"
```

Add the following statements to my.cnf:

```
[mysqld]
early-plugin-load="keyring_vault=keyring_vault.so"
loose-keyring_value_config="/home/mysql/keyring_vault.conf"
```

Restart the server.

Note

The keyring_vault extension, ".so", and the file location for the vault configuration should be changed to match your operating system's extension and operating system location.

14.2.2 Describing the keyring_vault_config file

The keyring_vault_config file has the following information:

- `vault_url` - the Vault server address
- `secret_mount_point` - where the keyring_vault stores the keys
- `secret_mount_point_version` - the `KV Secrets Engine version (kv or kv-v2)` used. Implemented in *Percona Server for MySQL* 5.7.33-36.
- `token` - a token generated by the Vault server
- `vault_ca [optional]` - if the machine does not trust the Vault's CA certificate, this variable points to the CA certificate used to sign the Vault's certificates.

The following is a configuration file example:

```
vault_url = https://vault.public.com:8202
secret_mount_point = secret
secret_mount_point_version = AUTO
token = 58a20c08-8001-fd5f-5192-7498a48eaf20
vault_ca = /data/keyring_vault_confs/vault_ca.crt
```

Warning

Each `secret_mount_point` must be used by only one server. Multiple servers using the same secret_mount_point may cause unpredictable behavior.

Create a backup of the keyring configuration file or data file immediately after creating the encrypted tablespace. If you are using Master key encryption, backup before master key rotation and after master key rotation.

The first time a key is retrieved from a keyring, the keyring_vault communicates with the Vault server to retrieve the key type and data.

Variables

keyring_vault_config

keyring_vault_timeout

14.2.3 Verifying the Keyring Plugin is Active

To verify the keyring plugin is active, run the SHOW PLUGINS statement or run a query on the INFORMATION_SCHEMA.PLUGINS table. You can also query the PLUGINS view.

```
SELECT plugin_name, plugin_status FROM INFORMATION_SCHEMA.PLUGINS WHERE plugin_name LIKE
'keyring%';
```

The output could be the following:

```
+---------------+----------------+
| plugin_name   | plugin_status  |
+===============+================+
| keyring_file  | ACTIVE         |
+---------------+----------------+
```

14.2.4 Encrypting a File-Per-Table Tablespace

The CREATE TABLESPACE statement is extended to allow the `ENCRYPTION=['Y/N']` option to encrypt a File-per-Table tablespace.

```
CREATE TABLE myexample (id INT mytext varchar(255)) ENCRYPTION='Y';
```

To enable encryption to an existing tablespace, add the `ENCRYPTION` option to the `ALTER TABLE` statement.

```
CREATE TABLE myexample ENCRYPTION='Y';
```

You must add the `ENCRYPTION` option to ALTER TABLE to change the table encryption state. Without the `ENCRYPTION` option, an encrypted table remains encrypted or an unencrypted table remains unencrypted.

To change the tablespace key, run the optimize table command.

```
mysql> optimize table t1;
```

14.2.5 Encrypting a General Tablespace

As of Percona Server 5.7.20-18, *Percona Server for MySQL* supports general tablespace encryption. You cannot partially encrypt the tables in a general tablespace. All of the tables must be encrypted or none of the tables are encrypted.

Automatically Encrypting Tablespaces

Add the `innodb_encrypt_tables` variable to my.cnf to automatically encrypt general tablespaces. The possible values for the variable are:

Value	Description
OFF	The default value which disables automatic encryption of new tables
ON	Enables automatic encryption for new tables
FORCE	New tables are automatically created with encryption.
Adding ENCRYPTION=NO to either a CREATE TABLE or ALTER TABLE statement results in a warning.	

The CREATE TABLESPACE statement is extended to allow the `ENCRYPTION=['Y/N']` option.

```
CREATE TABLE t1 (id INT) ENCRYPTION='Y';
```

To encrypt an existing table, add the ENCRYPTION option in the `ALTER TABLE` statement.

```
ALTER TABLE t1 ENCRYPTION='Y';
```

You can also disable encryption for a table, set the encryption to N.

```
ALTER TABLE t1 ENCRYPTION='N';
```

Note

The `ALTER TABLE` statement modifies the current encryption mode only if the `ENCRYPTION` clause is explicitly added.

System Variables

Note

You cannot change the tablespace key for tables in a general tablespace.

14.2.6 Encrypting Binary Logs

To start binlog encryption, start the server with `-encrypt-binlog=1`. This state requires `-master_verify_checksum` and `-binlog_checksum` to be `ON` and one of the keyring plugins loaded.

NOTE: These actions do not encrypt all binlogs in a replication schema. You must enable `encrypt-binlog` on each of the replica servers, even if they do not produce binlog files. Enabling encryption on replica servers enable relay log encryption.

You can rotate the encryption key used by *Percona Server for MySQL* by running the following statement:

```
SELECT rotate_system_key("percona_binlog");
```

Note

The `rotate_system_key("percona_binlog")` command is **Experimental** quality.

This command creates a new binlog encryption key in the keyring. The new key encrypts the next binlog file.

14.2.7 Temporary file encryption

Percona Server for MySQL supports the encryption of temporary file storage. Users enable the encryption with `encrypt-tmp_files`.

Enable the variable in the following command:

```
[mysqld]
...
encrypt-tmp-files=ON
...
```

14.2.8 Verifying the Encryption Setting

For single tablespaces, verify the ENCRYPTION option using INFORMATION_SCHEMA.TABLES and the CREATE OPTIONS settings.

```
SELECT TABLE_SCHEMA, TABLE_NAME, CREATE_OPTIONS FROM
       INFORMATION_SCHEMA.TABLES WHERE CREATE_OPTIONS LIKE '%ENCRYPTION%';
```

The output could be the following:

```
+----------------------+-------------------+------------------------------+
| TABLE_SCHEMA         | TABLE_NAME        | CREATE_OPTIONS               |
+----------------------+-------------------+------------------------------+
|sample                | t1                | ENCRYPTION="Y"               |
+----------------------+-------------------+------------------------------+
```

A `flag` field in the `INFORMATION_SCHEMA.INNODB_TABLESPACES` has the bit number 13 set if the tablespace is encrypted. This bit can be checked with the `flag & 8192` expression with the following method:

```
SELECT space, name, flag, (flag & 8192) != 0 AS encrypted FROM
INFORMATION_SCHEMA.INNODB_TABLESPACES WHERE name in ('foo', 'test/t2', 'bar',
'noencrypt');
```

The output could be the following:

```
  +-------+-----------+-------+-----------+
```

```
| space | name      | flag  | encrypted |
+-------+-----------+-------+-----------+
|    29 | foo       | 10240 |      8192 |
|    30 | test/t2   |  8225 |      8192 |
|    31 | bar       | 10240 |      8192 |
|    32 | noencrypt |  2048 |         0 |
+-------+-----------+-------+-----------+
4 rows in set (0.01 sec)
```

To allow for master Key rotation, you can encrypt an already encrypted InnoDB system tablespace with a new master key by running the following `ALTER INSTANCE` statement:

```
ALTER INSTANCE ROTATE INNODB MASTER KEY;
```

14.2.9 Rotating the Master Key

For security, you should rotate the Master key in a timely manner. Use the `ALTER INSTANCE` statement. To rotate the key, you must have `SUPER` privilege.

```
ALTER INSTANCE ROTATE INNODB MASTER KEY;
```

The statement cannot be run at the same time you run `CREATE TABLE ... ENCRYPTION` or `ALTER TABLE ENCRYPTION` statements. The `ALTER INSTANCE` statement uses locks to prevent conflicts. If a DML statement is running, that statement must complete before the `ALTER INSTANCE` statement begins.

When the Master key is rotated, the tablespace keys in that instance are re-encrypted. The operation does not re-encrypt the tablespace data.

The re-encryption for the tablespace keys must succeed for the key rotation to be successful. If the rotation is interrupted, for example, if there is a server failure, the operation rolls forward when the server restarts.

14.2.10 InnoDB System Tablespace Encryption

This feature was in **tech preview** from version 5.7.23-24 but is **deprecated** from version 5.7.32-35. This feature is not recommended to be used in production.

The InnoDB system tablespace is encrypted by using master key encryption. The server must be started with the `--bootstrap` option.

If the variable innodb_sys_tablespace_encrypt is set to ON and the server has been started in the bootstrap mode, you may create an encrypted table as follows:

```
mysql> CREATE TABLE ... TABLESPACE=innodb_system ENCRYPTION='Y'
```

Note

You cannot encrypt existing tables in the System tablespace.

It is not possible to convert the system tablespace from encrypted to unencrypted or vice versa. A new instance should be created and user tables must be transferred to the desired instance.

You can encrypt the already encrypted InnoDB system tablespace (key rotation) with a new master key by running the following `ALTER INSTANCE` statement:

```
mysql> ALTER INSTANCE ROTATE INNODB MASTER KEY
```

`innodb_sys_tablespace_encrypt`

Option	Description
Command-line	--innodb-sys-tablespace-encrypt
Scope	Global
Dynamic	No
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.23-24 and deprecated in Percona Server for MySQL 5.7.32-35. Enables the encryption of the InnoDB System tablespace. It is essential that the server is started with the `--bootstrap` option.

14.2.11 Doublewrite buffer

This feature was in **tech preview** from version Percona Server 5.7.23-24 but is **deprecated** from version Percona Server for MySQL 5.7.32-35. This feature is not recommended to be used in production.

The two types of doublewrite buffers used in *Percona Server for MySQL* are encrypted differently.

When the InnoDB system tablespace is encrypted, the `doublewrite buffer` pages are encrypted as well. The key which was used to encrypt the InnoDB system tablespace is also used to encrypt the doublewrite buffer.

Percona Server for MySQL encrypts the `parallel doublewrite buffer` with the respective tablespace keys. Only encrypted tablespace pages are written as encrypted in the parallel doublewrite buffer. Unencrypted tablespace pages will be written as unencrypted.

`innodb_parallel_dblwr_encrypt`

Option	Description
Command-line	--innodb-parallel-dblwr-encrypt
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.23-24 and deprecated in Percona Server for MySQL 5.7.32-35. Enables the encryption of the parallel doublewrite buffer. For encryption, uses the key of the tablespace where the parallel doublewrite buffer is used.

14.2.12 InnoDB Undo Tablespace Encryption

This feature was in **tech preview** from version Percona Server 5.7.23-24 but is **deprecated** from version Percona Server for MySQL 5.7.32-35. This feature is not recommended to be used in production.

The encryption of InnoDB Undo tablespaces is only available when using separate undo tablespaces. Otherwise, the InnoDB undo log is part of the InnoDB system tablespace.

14.2.13 System variables

`innodb_undo_log_encrypt`

Option	Description
Command-line	--innodb-undo-log-encrypt
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	Off

The variable has been implemented in Percona Server 5.7.23-24 and deprecated in Percona Server for MySQL 5.7.32-35. Enables the encryption of InnoDB Undo tablespaces. You can enable encryption and disable encryption while the server is running.

Note

If you enable undo log encryption, the server writes encryption information into the header. That information stays in the header during the life of the undo log. If you restart the server, the server will try to load the encryption key from the keyring during startup. If the keyring is not available, the server cannot start.

14.2.14 Redo Log Encryption

This feature was in **tech preview** from version 5.7.23-24 but is **deprecated** from version 5.7.32-35. This feature is not recommended to be used in production.

InnoDB redo log encryption is enabled by setting the variable innodb_redo_log_encrypt. This variable has three values: `MASTER_KEY`, `KEYRING_KEY` and `OFF` (set by default).

`MASTER_KEY` uses the InnoDB master key to encrypt with unique keys for each log file in the redo log header.

`KEYRING_KEY` uses the `percona_redo` versioned key from the keyring. When innodb_redo_log_encrypt is set to `KEYRING_KEY`, each new redo log file is encrypted with the latest `percona_redo` key from the keyring.

14.2.15 System variables

Implemented in version Percona Server for MySQL 5.7.27-30, the key rotation is redesigned to allow `SELECT rotate_system_key("percona_redo)`. The currently used key version is available in the innodb_redo_key_version status. The feature is **Experimental**.

14.2.16 Data Scrubbing

This feature was in **tech preview** from version Percona Server 5.7.23-24 but is **deprecated** from version Percona Server for MySQL 5.7.32-35. This feature is not recommended to be used in production.

While data encryption ensures that the existing data are not stored in plain form, the data scrubbing literally removes the data once the user decides they should be deleted. Compare this behavior with how the `DELETE` statement works which only marks the affected data as *deleted* - the space claimed by this data is overwritten with new data later.

Once enabled, data scrubbing works automatically on each tablespace separately. To enable data scrubbing, you need to set the following variables:

- innodb-background-scrub-data-uncompressed
- innodb-background-scrub-data-compressed

Uncompressed tables can also be scrubbed immediately, independently of key rotation or background threads. This can be enabled by setting the variable innodb-immediate-scrub-data-uncompressed. This option is not supported for compressed tables.

Note that data scrubbing is made effective by setting the innodb_online_encryption_threads variable to a value greater than **zero**.

14.2.17 System Variables

`innodb_background_scrub_data_compressed`

Option	Description
Command-line	--innodb-background-scrub-data-compressed
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.23-24 and deprecated in Percona Server for MySQL 5.7.32-35.

`innodb_background_scrub_data_uncompressed`

Option	Description
Command-line	--innodb-background-scrub-data-uncompressed
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.23-24 and deprecated in Percona Server for MySQL 5.7.32-35.

14.2.18 Variables

keyring_vault_config - Defines the location of the Loading the Keyring Plugin configuration file.

Option	Description
Command line	--keyring-vault-config
Dynamic	Yes
Scope	Global
Variable Type	Text
Default	

keyring_vault_timeout - Set the duration in seconds for the Vault server connection timeout. The default value is `15`. The allowed range is from `0` to `86400`. To wait an infinite amount of time set the variable to `0`.

Option	Description
Command line	--keyring-vault-timeout
Dynamic	Yes
Scope	Global
Variable Type	Numeric
Default	15

`innodb_encrypt_tables`

Option	Description
Command-line	--innodb-encrypt-tables
Scope	Global
Dynamic	Yes
Data type	Text
Default	OFF

The variable has been implemented in Percona Server 5.7.21-21.

Note

This variable is **Experimental** quality.

`innodb_redo_log_encrypt`

Option	Description
Command-line	--innodb-redo-log-encrypt
Scope	Global
Dynamic	Yes
Data type	Text
Default	OFF

The variable has been implemented in Percona Server 5.7.23-24. Enables the encryption of the redo log.

`innodb_scrub_log`

Option	Description
Command-line	--innodb-scrub-log
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	OFF

The variable has been implemented in Percona Server 5.7.23-24. Specifies if data scrubbing should be automatically applied to the redo log.

`innodb_scrub_log_speed`

Option	Description
Command-line	--innodb-scrub-log-speed
Scope	Global
Dynamic	Yes
Data type	Text
Default	

The variable has been implemented in Percona Server 5.7.23-24. Specifies the velocity of data scrubbing (writing dummy redo log records) in bytes per second.

Last update: 2022-09-27

Download PDF

14.3 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.

14.3.1 Installation

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

```
mysqlINSTALL 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

```
mysqlSHOW PLUGINS;
```

The output should be include the following:

```
...
...
| auth_pam                      | ACTIVE   | AUTHENTICATION     | auth_pam.so | GPL     |
```

14.3.2 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
```

14.3.3 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.

14.3.4 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.

14.3.5 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.

14.3.6 Version Specific Information

- 8.0.12-1 Feature ported from *Percona Server for MySQL* 5.7.

Download PDF

14.4 SSL Improvements

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.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

14.5 Data Masking

This feature was implemented in *Percona Server for MySQL* version Percona Server for MySQL 5.7.32-35.

The Percona Data Masking plugin is a free and Open Source implementation of the MySQL's data masking plugin. Data Masking provides a set of functions to hide sensitive data with modified content.

Data masking can have either of the characteristics:

- Generation of random data, such as an email address
- De-identify data by transforming the data to hide content

Installing the plugin

The following command installs the plugin:

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

Data Masking functions

The data masking functions have the following categories:

- General purpose
- Special purpose
- Generating Random Data with Defined characteristics
- Using Dictionaries to Generate Random Data

General Purpose

The general purpose data masking functions are the following:

Parameter	Description	Sample

mask_inner(string, margin1, margin2 [, character])	Returns a result where only the inner part of a string is masked. An optional masking character can be specified.	`mysql> SELECT mask_inner('123456789', 1, 2);`
		`+----------------------------------+`
		`| mask_inner("123456789", 1, 2) |`
		`+----------------------------------+`
		`|1XXXXXX89 |`
		`+----------------------------------+`

mask_outer(string, margin1, margin2 [, character])	Masks the outer part of the string. The inner section is not masked.	`mysql> SELECT mask_outer('123456789', 2, 2);`
		`+----------------------------------+`
		`| mask_outer("123456789", 2, 2). |`
		`+----------------------------------+`
		`| XX34567XX |`
		`+----------------------------------+`

Special Purpose

The special purpose data masking functions are as follows:

Generating Random Data for Specific Requirements

The following functions generate random values for specific requirements:

Parameter	Description	Sample

gen_range(lower, upper)	Generates a random number based on a selected range and supports negative numbers.	`mysql> SELECT gen_range(10, 100) AS result;`
		`+-------------------------------------+`
		`| result |`
		`+-------------------------------------+`
		`| 56 |`
		`+-------------------------------------+`
		`mysql> SELECT gen_range(100,80);`
		`+-------------------------------------+`
		`| gen_range(100,80) |`
		`+-------------------------------------+`
		`| 91 |`
		`+-------------------------------------+`

gen_rnd_email()	Generates a random email address. The domain is `example.com`.	`mysql> SELECT gen_rnd_email();`
		`+---------------------------------------+`
		`| gen_rnd_email() |`
		`+---------------------------------------+`
		`| sma.jrts@example.com |`
		`+---------------------------------------+`

gen_rnd_pan([size in integer])	Generates a random primary account number. This function should only be used for test purposes.	`mysql> SELECT mask_pan(gen_rnd_pan());`
		`+-------------------------------------+`
		`| mask_pan(gen_rnd_pan()) |`
		`+-------------------------------------+`
		`| XXXXXXXXXXXX4444 |`
		`+-------------------------------------+`

gen_rnd_us_phone()	Generates a random U.S. phone number. The generated number adds the *1* dialing code and is in the *555* area code. The *555* area code is not valid for any U.S. phone number.	`mysql> SELECT gen_rnd_us_phone();`
		`+--------------------------------+`
		`| gen_rnd_us_phone() |`
		`+--------------------------------+`
		`| 1-555-635-5709 |`
		`+--------------------------------+`

gen_rnd_ssn()	Generates a random, non-legitimate US	`mysql> SELECT gen_rnd_ssn()`

Parameter	Description	Sample
	Social Security Number in an `AAA-BBB-CCCC` format. This function should only be used for test purposes.	+-----------------------------+ | gen_rnd_ssn() | +-----------------------------+ | 995-33-5656 | +-----------------------------+

Using Dictionaries to Generate Random Terms

Data masking returns a value from a range. To use a predefined file as the range to select a string value, load and use a dictionary. A dictionary supports only strings and is loaded from a file with the following characteristics:

- Plain text
- One term per line
- Must contain at least one entry

An example of a dictionary, which is a list of trees, located in /usr/local/mysql/dict-files/testdict

- Black Ash
- White Ash
- Bigtooth Aspen
- Quaking Aspen

The following table displays the commands for using dictionaries to generate random terms:

Parameter	Description	Sample

gen_range(lower, upper)	Generates a random number based on a selected range and supports negative numbers.	`mysql> SELECT gen_range(10, 100) AS result;`
		`+-------------------------------------+`
		`| result |`
		`+-------------------------------------+`
		`| 56 |`
		`+-------------------------------------+`
		`mysql> SELECT gen_range(100,80);`
		`+-------------------------------------+`
		`| gen_range(100,80) |`
		`+-------------------------------------+`
		`| 91 |`
		`+-------------------------------------+`

gen_rnd_email()	Generates a random email address. The domain is `example.com`.	`mysql> SELECT gen_rnd_email();`
		`+---------------------------------------+`
		`| gen_rnd_email() |`
		`+---------------------------------------+`
		`| sma.jrts@example.com |`
		`+---------------------------------------+`

gen_rnd_pan([size in integer])	Generates a random primary account number. This function should only be used for test purposes.	`mysql> SELECT mask_pan(gen_rnd_pan());`
		`+-------------------------------------+`
		`| mask_pan(gen_rnd_pan()) |`
		`+-------------------------------------+`
		`| XXXXXXXXXXXX4444 |`
		`+-------------------------------------+`

gen_rnd_us_phone()	Generates a random U.S. phone number. The generated number adds the *1* dialing code and is in the *555* area code. The *555* area code is not valid for any U.S. phone number.	`mysql> SELECT gen_rnd_us_phone();`
		`+-------------------------------+`
		`| gen_rnd_us_phone() |`
		`+-------------------------------+`
		`| 1-555-635-5709 |`
		`+-------------------------------+`

gen_rnd_ssn()	Generates a random, non-legitimate US	`mysql> SELECT gen_rnd_ssn()`

Parameter	Description	Sample
	Social Security Number in an `AAA-BBB-CCCC` format. This function should only be used for test purposes.	+-----------------------------+ | gen_rnd_ssn() | +-----------------------------+ | 995-33-5656 | +-----------------------------+

Uninstalling the plugin

The UNINSTALL PLUGIN statement disables and uninstalls the plugin.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

15. TokuDB

Download PDF

15.1 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 replica 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 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* 5.7 releases 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* 5.7. 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 Replica 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:

- TokuDB Installation
- Using TokuDB
- Getting Started with TokuDB
- TokuDB Variables
- Percona TokuBackup
- TokuDB Troubleshooting
- Frequently Asked Questions
- Removing TokuDB storage engine

15.1.1 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 Replica Lag: TokuDB replication replicas can be configured to process the replication stream with virtually no read IO. Uniqueness checking is performed on the TokuDB source and can be skipped on all TokuDB replicas. Also, row based replication ensures that all before and after row images are captured in the binary logs, so the TokuDB replicas can harness the power of Fractal Tree indexes and bypass traditional read-modify-write behavior. This "Read Free Replication" ensures that replication replicas do not fall behind the source 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.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

15.2 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* 5.7 releases 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 replica 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* 5.7. 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.

15.2.1 Prerequisites

`libjemalloc` library

TokuDB storage engine requires `libjemalloc` library 3.3.0 or greater. If the version in the distribution repository is lower than that you can use one from Percona Software Repositories or download it from somewhere else.

If the `libjemalloc` wasn't installed and enabled before it will be automatically installed when installing the TokuDB storage engine package by using the **apt`** or **yum** package manager, but *Percona Server 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 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
```

The output could be the following:

```
 [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

This setting lasts until the server is rebooted.

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

15.2.2 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-57.x86_64
```

or

```
root@wheezy:~# apt install percona-server-tokudb-5.7
```

15.2.3 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-admin --enable-tokudb -u <mysql_admin_user> -p[mysql_admin_pass] [-S <socket>] [-h
<host> -P <port>]

   * See http://www.percona.com/doc/percona-server/5.7/tokudb/tokudb_installation.html for
more installation details

   * See http://www.percona.com/doc/percona-server/5.7/tokudb/tokudb_intro.html for an
introduction to TokuDB
```

Percona Server for MySQL 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-admin --enable-tokudb -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;
```

The output could be the following:

```
...
 | TokuDB | YES | Tokutek TokuDB Storage Engine with Fractal Tree(tm) Technology | YES | YES
| YES |
...
```

15.2.4 Enabling the TokuDB Storage Engine Manually

If you don't want to use `ps-admin` script you'll need to manually install the storage engine ad required plugins.

```
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;
```

The output could be the following:

```
...
| 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;
```

The output could be the following:

```
...
| 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     |
...
```

15.2.5 TokuDB Version

TokuDB storage engine version can be checked with:

```
mysql> SELECT @@tokudb_version;
```

The output could be the following:

```
+-----------------+
| @@tokudb_version |
+-----------------+
| 5.7.10-1rc1     |
+-----------------+
1 row in set (0.00 sec)
```

15.2.6 Upgrade

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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

15.3 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.

15.3.1 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*.

15.3.2 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.

15.3.3 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 tokudb_create_index_online client session variable is examined.

Hot index creation is invoked using the `CREATE INDEX` command after setting tokudb_create_index_online to `on` as follows:

```
mysql> SET tokudb_create_index_online=on;
```

The output should resemble the following:

```
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 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.

15.3.4 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.

15.3.5 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 tokudb_row_format. If no row format is set nor is tokudb_row_format, the zlib compressor is used.

row_format and 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.

15.3.6 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.

15.3.7 Read Free Replication

TokuDB replicas can be configured to perform significantly less read IO in order to apply changes from the source. 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 source:
  - Enable row based replication: set `BINLOG_FORMAT=ROW`
- On the replication replica(s):
  - The replica 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 replica(s).

Note

As long as the source is using row based replication, this optimization is available on a TokuDB replica. This means that it's available even if the source 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 replica.

15.3.8 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.

15.3.9 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.

15.3.10 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.

15.3.11 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.

15.3.12 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.

15.3.13 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.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

15.4 Fast Updates with TokuDB

15.4.1 Introduction

Update intensive applications can have their throughput limited by the random read capacity of the storage system. The cause of the throughput limit is the read-modify-write algorithm that MySQL uses to process update statements (read a row from the storage engine, apply the updates to it, write the new row back to the storage engine).

To address this throughput limit, TokuDB provides an experimental fast update feature, which uses a different update algorithm. Update expressions of the SQL statement are encoded into tiny programs that are stored in an update Fractal Tree message. This update message is injected into the root of the Fractal Tree index. Eventually, these update messages reach a leaf node, where the update programs are applied to the row. Since messages are moved between Fractal Tree levels in batches, the cost of reading in the leaf node is amortized over many update messages.

This feature is available for `UPDATE` and `INSERT` statements, and can be turned ON/OFF separately for them with use of two variables. Variable tokudb_enable_fast_update variable toggles fast updates for the `UPDATE`, and tokudb_enable_fast_upsert does the same for `INSERT`.

15.4.2 Limitations

Fast updates are activated instead of normal MySQL read-modify-write updates if the executed expression meets the number of conditions.

- fast updates can be activated for a statement or a mixed replication,
- a primary key must be defined for the involved table,
- both simple and compound primary keys are supported, and `int`, `char` or `varchar` are the allowed data types for them,
- updated fields should have `Integer` or `char` data type,
- fields that are part of any key should be not updated,
- clustering keys are not allowed,
- triggers should be not involved,
- supported update expressions should belong to one of the following types:
    - `x = constant`
    - `x = x + constant`
    - `x = x - constant`
    - `x = if (x=0,0,x-1)`
    - `x = x + values`

15.4.3 Usage Specifics and Examples

Following example creates a table that associates event identifiers with their count:

```
CREATE TABLE t (
    event_id bigint unsigned NOT NULL PRIMARY KEY,
    event_count bigint unsigned NOT NULL
);
```

Many graph applications that map onto relational tables can use duplicate key inserts and updates to maintain the graph. For example, one can update the meta-data associated with a link in the graph using duplicate key insertions. If the affected rows is not used by the application, then the insertion or update can be marked and executed as a fast insertion or a fast update.

Insertion example

If it is not known if the event identifier (represented by event_id) already exists in the table, then `INSERT ... ON DUPLICATE KEY UPDATE ...` statement can insert it if not existing, or increment its event_count otherwise. Here is an example with duplicate key insertion statement, where `%id` is some specific event_id value:

```
INSERT INTO t VALUES (%id, 1)
  ON DUPLICATE KEY UPDATE event_count=event_count+1;
```

EXPLANATION

If the event id's are random, then the throughput of this application would be limited by the random read capacity of the storage system since each `INSERT` statement has to determine if this event_id exists in the table.

TokuDB replaces the primary key existence check with an insertion of an "upsert" message into the Fractal Tree index. This "upsert" message contains a copy of the row and a program that increments event_count. As the Fractal Tree buffer's get filled, this "upsert" message is flushed down the tree. Eventually, the message reaches a leaf node and gets executed there. If the key exists in the leaf node, then the event_count is incremented. Otherwise, the new row is inserted into the leaf node.

Update example

If event_id is known to exist in the table, then `UPDATE` statement can be used to increment its event_count (once again, specific event_id value is written here as `%id`):

```
UPDATE t SET event_count=event_count+1
WHERE event_id=%id;
```

EXPLANATION

TokuDB generates an "update" message from the `UPDATE` statement and its update expression trees, and inserts this message into the Fractal Tree index. When the message eventually reaches the leaf node, the increment program is extracted from the message and executed.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

15.5 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
```

The output should be similar to the following:

```
...
-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.

15.5.1 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 source/glossary.rst`datadir`.

15.5.2 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 source/glossary.rst`datadir`.

15.5.3 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 source/glossary.rst`datadir`.

15.5.4 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 tokudb_data_dir if set, otherwise the MySQL `datadir` is used.

15.5.5 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 tokudb_log_dir if set, otherwise the MySQL source/glossary.rst`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.

15.5.6 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 tokudb_tmp_dir if set. If not, the tokudb_data_dir is used if set, otherwise the MySQL source/glossary.rst`datadir` is used.

15.5.7 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.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

15.6 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 Percona Server for MySQL 5.7.15-9 a new server option was introduced called 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 enabled by default.

In *Percona Server for MySQL* Percona Server for MySQL 5.7.18-14 new tokudb_dir_cmd variable has been implemented that can be used to edit the contents of the TokuDB/PerconaFT directory map.

15.6.1 Moving TokuDB data files to a location outside of the default MySQL datadir

TokuDB uses the location specified by the tokudb_data_dir variable for all of its data files. If the tokudb_data_dir variable is not explicitly set, TokuDB will use the location specified by the servers source/glossary.rst`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 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 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.

15.6.2 Moving TokuDB temporary files to a location outside of the default MySQL datadir

TokuDB will use the location specified by the tokudb_tmp_dir variable for all of its temporary files. If tokudb_tmp_dir variable is not explicitly set, TokuDB will use the location specified by the tokudb_data_dir variable. If the tokudb_data_dir variable is also not explicitly set, TokuDB will use the location specified by the servers source/glossary.rst`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 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 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.

15.6.3 Moving TokuDB recovery log files to a location outside of the default MySQL datadir

TokuDB will use the location specified by the tokudb_log_dir variable for all of its recovery log files. If the tokudb_log_dir variable is not explicitly set, TokuDB will use the location specified by the servers source/glossary.rst`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 tokudb_log_dir. If that symlink refers to a directory on another physical data volume, the 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 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.

15.6.4 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 refTOKUDB_FILE_MAP table to cross reference.

Beginning with *Percona Server for MySQL* Percona Server for MySQL 5.7.15-9 a new server option was introduced called tokudb_dir_per_db to address this issue.

When tokudb_dir_per_db is enabled (`ON` 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.

15.6.5 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* Percona Server for MySQL 5.7.15-9 a new server option was introduced called tokudb_dir_per_db to address this issue.

When tokudb_dir_per_db variable is enabled (`ON` by default), all new tables and indices will be placed within their corresponding database directory within the `tokudb_data_dir` or server source/glossary.rst`datadir`.

If you have tokudb_data_dir variable set to something other than the server source/glossary.rst`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.

15.6.6 Editing TokuDB directory map with tokudb_dir_cmd

Note

This feature is currently considered *Experimental*.

In *Percona Server for MySQL* Percona Server for MySQL 5.7.18-14 new tokudb_dir_cmd variable has been implemented that can be used to edit the TokuDB directory map.

Warning

Use this variable only if you know what you're doing otherwise it will have 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 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 TokuDB_file_map table:

```
mysql> SELECT dictionary_name, internal_file_name FROM INFORMATION_SCHEMA.TokuDB_file_map;
```

The output should be similar to the following:

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

System Variables

`tokudb_dir_cmd`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	String

The variable has been implemented in Percona Server for MySQL 5.7.18-14. This variable is used to send commands to edit TokuDB directory map.

> **Warning**
>
> Use this variable only if you know what you're doing otherwise it **WILL** lead to data loss.

Status Variables

`tokudb_dir_cmd_last_error`

Option	Description
Scope	Global
Data type	Numeric

This variable contains the error number of the last executed command by using the tokudb_dir_cmd variable.

`tokudb_dir_cmd_last_error_string`

Option	Description
Scope	Global
Data type	Numeric

This variable contains the error string of the last executed command by using the tokudb_dir_cmd variable.

Download PDF

15.7 TokuDB Background ANALYZE TABLE

Percona Server for MySQL has an option to automatically analyze tables in the background based on a measured change in data. This has been done by implementing the background job manager that can perform operations on a background thread.

15.7.1 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 TOKUDB_BACKGROUND_JOB_STATUS table.

New 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 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 tokudb_analyze_in_background, and 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 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.

15.7.2 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 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 tokudb_analyze_time is in effect and the analysis has not reached the half way point of the index by the time 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` / 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 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.

15.7.3 System Variables

`tokudb_analyze_in_background`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global/Session
Dynamic	Yes
Data type	Boolean
Default	ON

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.

`tokudb_analyze_mode`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global/Session
Dynamic	Yes
Data type	ENUM
Default	TOKUDB_ANALYZE_STANDARD
Range	TOKUDB_ANALYZE_CANCEL, TOKUDB_ANALYZE_STANDARD, TOKUDB_ANALYZE_RECOUNT_ROWS

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 tokudb_analyze_time, tokudb_analyze_in_background, and tokudb_analyze_throttle at the time of its scheduling, either via a user invoked `ANALYZE TABLE` or an auto schedule as a result of 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 tokudb_analyze_in_background, and tokudb_analyze_throttle. Changing the global or session instances of these values after scheduling will have no effect on the job.

`tokudb_analyze_throttle`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global/Session
Dynamic	Yes
Data type	Numeric
Default	0

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`.

`tokudb_analyze_time`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global/Session
Dynamic	Yes
Data type	Numeric
Default	5

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.

`tokudb_auto_analyze`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global/Session
Dynamic	Yes
Data type	Numeric
Default	30

Percentage of table change as `INSERT/UPDATE/DELETE` commands to trigger an `ANALYZE TABLE` using the current session tokudb_analyze_in_background, tokudb_analyze_mode, tokudb_analyze_throttle, and tokudb_analyze_time settings. If this variable is enabled and 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.

`tokudb_cardinality_scale_percent`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global/Session
Dynamic	Yes
Data type	Numeric
Default	100
Range	0-100

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 * tokudb_cardinality_scale) / 100, for a scale of 50% the result would also be 2 for the index.

15.7.4 INFORMATION_SCHEMA Tables

`INFORMATION_SCHEMA.TOKUDB_BACKGROUND_JOB_STATUS`

Column Name	Description
'id'	'Simple monotonically incrementing job id, resets to 0 on server start.'
'database_name'	'Database name'
'table_name'	'Table name'
'job_type'	'Type of job, either TOKUDB_ANALYZE_STANDARD or TOKUDB_ANALYZE_RECOUNT_ROWS'
'job_params'	'Param values used by this job in string format. For example: TOKUDB_ANALYZE_DELETE_TIME=1.0; TOKUDB_ANALYZE_TIME=5; TOKUDB_ANALYZE_THROTTLE=2048;'
'scheduler'	'Either USER or AUTO to indicate if the job was explicitly scheduled by a user or if it was scheduled as an automatic trigger'
'scheduled_time'	'The time the job was scheduled'
'started_time'	'The time the job was started'
'status'	'Current job status if running. For example: ANALYZE TABLE standard db.tbl.idx 3 of 5 50% rows 10% time scanning forward'

This table holds the information on scheduled and running background `ANALYZE TABLE` jobs for TokuDB tables.

15.7.5 Version Specific Information

- Percona Server for MySQL 5.7.10-1: Feature ported from *Percona Server for MySQL* 5.6
- Percona Server for MySQL 5.7.11-4: tokudb_analyze_in_background is now set to `ON` by default and tokudb_auto_analyze is set to `30`

Download PDF

15.8 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.

15.8.1 TokuDB Server Variables

Name	Cmd-Line	Option File	Var Scope	Dynamic
tokudb_alter_print_error	Yes	Yes	Session, Global	Yes
tokudb_analyze_delete_fraction	Yes	Yes	Session, Global	Yes
tokudb_analyze_in_background	Yes	Yes	Session, Global	Yes
tokudb_analyze_mode	Yes	Yes	Session, Global	Yes
tokudb_analyze_throttle	Yes	Yes	Session, Global	Yes
tokudb_analyze_time	Yes	Yes	Session, Global	Yes
tokudb_auto_analyze	Yes	Yes	Session, Global	Yes
tokudb_backup_allowed_prefix	No	Yes	Global	No
tokudb_backup_dir	No	Yes	Session	No
tokudb_backup_exclude	Yes	Yes	Session, Global	Yes
tokudb_backup_last_error	Yes	Yes	Session, Global	Yes
tokudb_backup_last_error_string	Yes	Yes	Session, Global	Yes
tokudb_backup_plugin_version	No	No	Global	No
tokudb_backup_throttle	Yes	Yes	Session, Global	Yes
tokudb_backup_version	No	No	Global	No
tokudb_block_size	Yes	Yes	Session, Global	Yes
tokudb_bulk_fetch	Yes	Yes	Session, Global	Yes
tokudb_cachetable_pool_threads	Yes	Yes	Global	No
tokudb_cardinality_scale_percent	Yes	Yes	Global	Yes
tokudb_check_jemalloc	Yes	Yes	Global	No
tokudb_checkpoint_lock	Yes	Yes	Global	No
tokudb_checkpoint_on_flush_logs	Yes	Yes	Global	Yes
tokudb_checkpoint_pool_threads	Yes	Yes	Global	Yes
tokudb_checkpointing_period	Yes	Yes	Global	Yes
tokudb_cleaner_iterations	Yes	Yes	Global	Yes
tokudb_cleaner_period	Yes	Yes	Global	Yes
tokudb_client_pool_threads	Yes	Yes	Global	No

Name	Cmd-Line	Option File	Var Scope	Dynamic
tokudb_commit_sync	Yes	Yes	Session, Global	Yes
tokudb_compress_buffers_before_eviction	Yes	Yes	Global	No
tokudb_create_index_online	Yes	Yes	Session, Global	Yes
tokudb_data_dir	Yes	Yes	Global	No
tokudb_debug	Yes	Yes	Global	Yes
tokudb_dir_per_db	Yes	Yes	Global	Yes
tokudb_directio	Yes	Yes	Global	No
tokudb_disable_hot_alter	Yes	Yes	Session, Global	Yes
tokudb_disable_prefetching	Yes	Yes	Session, Global	Yes
tokudb_disable_slow_alter	Yes	Yes	Session, Global	Yes
tokudb_empty_scan	Yes	Yes	Session, Global	Yes
tokudb_enable_fast_update	Yes	Yes	Session, Global	Yes
tokudb_enable_fast_upsert	Yes	Yes	Session, Global	Yes
tokudb_enable_partial_eviction	Yes	Yes	Global	No
tokudb_fanout	Yes	Yes	Session, Global	Yes
tokudb_fs_reserve_percent	Yes	Yes	Global	No
tokudb_fsync_log_period	Yes	Yes	Global	Yes
tokudb_hide_default_row_format	Yes	Yes	Session, Global	Yes
tokudb_killed_time	Yes	Yes	Session, Global	Yes
tokudb_last_lock_timeout	Yes	Yes	Session, Global	Yes
tokudb_load_save_space	Yes	Yes	Session, Global	Yes
tokudb_loader_memory_size	Yes	Yes	Session, Global	Yes
tokudb_lock_timeout	Yes	Yes	Session, Global	Yes
tokudb_lock_timeout_debug	Yes	Yes	Session, Global	Yes
tokudb_log_dir	Yes	Yes	Global	No

Name	Cmd-Line	Option File	Var Scope	Dynamic
tokudb_max_lock_memory	Yes	Yes	Global	No
tokudb_optimize_index_fraction	Yes	Yes	Session, Global	Yes
tokudb_optimize_index_name	Yes	Yes	Session, Global	Yes
tokudb_optimize_throttle	Yes	Yes	Session, Global	Yes
tokudb_pk_insert_mode	Yes	Yes	Session, Global	Yes
tokudb_prelock_empty	Yes	Yes	Session, Global	Yes
tokudb_read_block_size	Yes	Yes	Session, Global	Yes
tokudb_read_buf_size	Yes	Yes	Session, Global	Yes
tokudb_read_status_frequency	Yes	Yes	Global	Yes
tokudb_row_format	Yes	Yes	Session, Global	Yes
tokudb_rpl_check_readonly	Yes	Yes	Session, Global	Yes
tokudb_rpl_lookup_rows	Yes	Yes	Session, Global	Yes
tokudb_rpl_lookup_rows_delay	Yes	Yes	Session, Global	Yes
tokudb_rpl_unique_checks	Yes	Yes	Session, Global	Yes
tokudb_rpl_unique_checks_delay	Yes	Yes	Session, Global	Yes
tokudb_strip_frm_data	Yes	Yes	Global	No
tokudb_support_xa	Yes	Yes	Session, Global	Yes
tokudb_tmp_dir	Yes	Yes	Global	No
tokudb_version	No	No	Global	No
tokudb_write_status_frequency	Yes	Yes	Global	Yes

`tokudb_alter_print_error`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global/Session
Dynamic	Yes
Data type	Boolean
Default	OFF

When set to `ON` errors will be printed to the client during the `ALTER TABLE` operations on TokuDB tables.

`tokudb_analyze_delete_fraction`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global/Session
Dynamic	Yes
Data type	Numeric
Default	1.000000
Range	0.0 - 1.000000

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.

`tokudb_backup_allowed_prefix`

Option	Description
Command-line	No
Config file	Yes
Scope	Global
Dynamic	No
Data type	String
Default	NULL

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';
```

The output could be:

```
+------------------------------+-----------+
| Variable_name                | Value     |
+------------------------------+-----------+
| tokudb_backup_allowed_prefix | /dumpdir  |
+------------------------------+-----------+
```

`tokudb_backup_dir`

Option	Description
Command-line	No
Config file	No
Scope	Session
Dynamic	Yes
Data type	String
Default	NULL

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.

`tokudb_backup_exclude`

Option	Description
Command-line	No
Config file	No
Scope	Session
Dynamic	Yes
Data type	String
Default	(mysqld_safe.pid)+

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:

```
mysqlset tokudb_backup_exclude='/lost\\+found(#|/)';
```

For more information see Percona TokuBackup.

`tokudb_backup_last_error`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	0

This session variable will contain the error number from the last backup. `0` indicates success. For more information see Percona TokuBackup.

`tokudb_backup_last_error_string`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	String
Default	NULL

This session variable will contain the error string from the last backup. For more information see Percona TokuBackup.

`tokudb_backup_plugin_version`

Option	Description
Command-line	No
Config file	No
Scope	Global
Dynamic	No
Data type	String

This read-only server variable documents the version of the TokuBackup plugin. For more information see Percona TokuBackup.

`tokudb_backup_throttle`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	18446744073709551615

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.

`tokudb_backup_version`

Option	Description
Command-line	No
Config file	No
Scope	Global
Dynamic	No
Data type	String

This read-only server variable documents the version of the hot backup library. For more information see Percona TokuBackup.

`tokudb_block_size`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	4194304
Range	4096 - 4294967295

This variable controls the maximum size of node in memory before messages must be flushed or node must be split.

Changing the value of 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.

`tokudb_bulk_fetch`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	ON

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`.

`tokudb_cache_size`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Numeric

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 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.

`tokudb_cachetable_pool_threads`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	0
Range	0 - 1024

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.

`tokudb_check_jemalloc`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Boolean
Default	ON

This variable enables/disables startup checking if jemalloc is linked and correct version and that transparent huge pages are disabled. Used for testing only.

`tokudb_checkpoint_lock`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	OFF

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.

`tokudb_checkpoint_on_flush_logs`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	OFF

When enabled forces a checkpoint if we get a flush logs command from the server.

`tokudb_checkpoint_pool_threads`

Option	Description
Command-line	Yes
Config file	Yes
Scope	
Dynamic	No
Data type	Numeric
Default	0
Range	0 - 1024

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`.

`tokudb_checkpointing_period`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	60
Range	0 - 4294967295

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.

`tokudb_cleaner_iterations`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	5
Range	0 - 18446744073709551615

This variable specifies how many internal nodes get processed in each tokudb_cleaner_period period. The default value is `5`. Setting this variable to `0` turns off cleaner threads.

`tokudb_cleaner_period`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	1
Range	0 - 18446744073709551615

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.

`tokudb_client_pool_threads`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Numeric
Default	0
Range	0 - 1024

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`.

`tokudb_commit_sync`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	ON

Session 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 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 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.

`tokudb_compress_buffers_before_eviction`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Boolean
Default	ON

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.

`tokudb_create_index_online`

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.

`tokudb_data_dir`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	String
Default	NULL

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.

`tokudb_debug`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	0
Range	0 - 18446744073709551615

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)
```

`tokudb_dir_per_db`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	ON

The variable has been implemented in Percona Server for MySQL 5.7.15-9. When this variable is set to `ON` all new tables and indices will be placed within their corresponding database directory within the tokudb_data_dir or system source/glossary.rst`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.

`tokudb_directio`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Boolean
Default	OFF

When enabled, TokuDB employs Direct I/O rather than Buffered I/O for writes. When using Direct I/O, consider increasing tokudb_cache_size from its default of 1/2 physical memory.

`tokudb_disable_hot_alter`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	OFF

This variable is used specifically for testing or to disable hot alter in case there are bugs. Not for use in production.

`tokudb_disable_prefetching`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	OFF

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`.

`tokudb_disable_slow_alter`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	OFF

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, 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
```

`tokudb_empty_scan`

Defines direction to be used to perform table scan to check for empty tables for bulk loader.

`tokudb_enable_fast_update`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global/Session
Dynamic	Yes
Data type	Boolean
Default	OFF

Toggles the fast updates feature ON/OFF for the `UPDATE` statement. Fast update involves queries optimization to avoid random reads during their execution.

`tokudb_enable_fast_upsert`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global/Session
Dynamic	Yes
Data type	Boolean
Default	OFF

Toggles the fast updates feature ON/OFF for the `INSERT` statement. Fast update involves queries optimization to avoid random reads during their execution.

`tokudb_enable_partial_eviction`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Boolean
Default	ON

This variable is used to control if partial eviction of nodes is enabled or disabled.

`tokudb_fanout`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	16
Range	2-16384

This variable controls the Fractal Tree fanout. The fanout defines the number of pivate keys or child nodes for each internal tree node. Changing the value of 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.

`tokudb_fs_reserve_percent`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Numeric
Default	5
Range	0-100

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.

`tokudb_fsync_log_period`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	0
Range	0-4294967295

This variable controls the frequency, in milliseconds, for `fsync()` operations. If set to `0` then the `fsync()` behavior is only controlled by the tokudb_commit_sync, which can be `ON` or `OFF`.

`tokudb_hide_default_row_format`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	ON

This variable is used to hide the `ROW_FORMAT` in `SHOW CREATE TABLE`. If `zlib` compression is used, row format will show as `DEFAULT`.

`tokudb_killed_time`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	4000
Range	0-18446744073709551615

This variable is used to specify frequency in milliseconds for lock wait to check to see if the lock was killed.

`tokudb_last_lock_timeout`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	String
Default	NULL

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 tokudb_lock_timeout_debug session variable must have bit `0` set for this behavior, otherwise this session variable will be empty.

`tokudb_load_save_space`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	ON

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 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 tokudb_prelock_empty and try again. More information is available in Known Issues.

`tokudb_loader_memory_size`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	100000000
Range	0-18446744073709551615

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 tokudb_cache_size, and may impact the running workload's performance as existing cached data must be ejected for the loader to begin.

`tokudb_lock_timeout`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	4000
Range	0-18446744073709551615

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 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.

`tokudb_lock_timeout_debug`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	1
Range	0-3

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 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
```

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.
```

`tokudb_log_dir`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	String
Default	NULL

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.

`tokudb_max_lock_memory`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	Numeric
Default	65560320
Range	0-18446744073709551615

This variable specifies the maximum amount of memory for the PerconaFT lock table.

`tokudb_optimize_index_fraction`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	1.000000
Range	0.000000 - 1.000000

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 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).

`tokudb_optimize_index_name`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	String
Default	NULL

This variable can be used to optimize a single index in a table, it can be set to select the index by name.

`tokudb_optimize_throttle`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	0
Range	0-18446744073709551615

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.

`tokudb_pk_insert_mode`

Removed in Percona Server for MySQL 5.7.12-5.

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	1
Range	0-3

Note

The tokudb_pk_insert_mode session variable has been removed in Percona Server for MySQL Percona Server for MySQL 5.7.12-5 and the behavior is now that of the former tokudb_pk_insert_mode set to `1`. The optimization will be used where safe and not used where not safe.

`tokudb_prelock_empty`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	ON

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 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.

`tokudb_read_block_size`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	65536 (64KB)
Range	4096 - 4294967295

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 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.

`tokudb_read_buf_size`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	131072 (128KB)
Range	0 - 1048576

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.

`tokudb_read_status_frequency`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	10000
Range	0 - 4294967295

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 tokudb_write_status_frequency to `1`, and then run `SHOW PROCESSLIST` several times to understand what progress is being made.

`tokudb_row_format`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	ENUM
Default	TOKUDB_ZLIB
Range	TOKUDB_DEFAULT, TOKUDB_FAST, TOKUDB_SMALL, TOKUDB_ZLIB, TOKUDB_QUICKLZ, TOKUDB_LZMA, TOKUDB_SNAPPY, TOKUDB_UNCOMPRESSED

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.

`tokudb_rpl_check_readonly`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	ON

The TokuDB replication code will run row events from the binary log with Read Free Replication when the replica is in read-only mode. This variable is used to disable the replica read only check in the TokuDB replication code.

This allows Read-Free-Replication to run when the replica is NOT read-only. By default, tokudb_rpl_check_readonly is enabled (check that replica is read-only). Do **NOT** change this value unless you completely understand the implications!

`tokudb_rpl_lookup_rows`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	ON

When disabled, TokuDB replication replicas 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 replica.

Note

Optimization is only enabled when read_only is set to `1` and binlog_format is `ROW`.

`tokudb_rpl_lookup_rows_delay`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	0
Range	0 - 18446744073709551615

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.

`tokudb_rpl_unique_checks`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	ON

When disabled, TokuDB replication replicas skip uniqueness checks on inserts and updates, which eliminates all associated read I/O for these operations.

Note

Optimization is only enabled when read_only is set to `1` and binlog_format is `ROW`.

`tokudb_rpl_unique_checks_delay`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Numeric
Default	0
Range	0 - 18446744073709551615

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.

`tokudb_strip_frm_data`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Boolean
Default	OFF

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.

`tokudb_support_xa`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Session, Global
Dynamic	Yes
Data type	Boolean
Default	ON

This variable defines whether or not the prepare phase of an XA transaction performs an `fsync()`.

`tokudb_tmp_dir`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	No
Data type	String

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.

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.

`tokudb_version`

Option	Description
Command-line	No
Config file	No
Scope	Global
Dynamic	No
Data type	String

This read-only variable documents the version of the TokuDB storage engine.

`tokudb_write_status_frequency`

Option	Description
Command-line	Yes
Config file	Yes
Scope	Global
Dynamic	Yes
Data type	Numeric
Default	1000
Range	0 - 4294967295

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 tokudb_read_status_frequency to 1, and then run `SHOW PROCESSLIST` several times to understand what progress is being made.

Download PDF

15.9 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:

```
mysqlSHOW STATUS LIKE 'tokudb%';
```

15.9.1 TokuDB Status Variables Summary

The following global status variables are available:

Name	Var Type
Tokudb_DB_OPENS	integer
Tokudb_DB_CLOSES	integer
Tokudb_DB_OPEN_CURRENT	integer
Tokudb_DB_OPEN_MAX	integer
Tokudb_LEAF_ENTRY_MAX_COMMITTED_XR	integer
Tokudb_LEAF_ENTRY_MAX_PROVISIONAL_XR	integer
Tokudb_LEAF_ENTRY_EXPANDED	integer
Tokudb_LEAF_ENTRY_MAX_MEMSIZE	integer
Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_IN	integer
Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_OUT	integer
Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_IN	integer
Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_OUT	integer
Tokudb_CHECKPOINT_PERIOD	integer
Tokudb_CHECKPOINT_FOOTPRINT	integer
Tokudb_CHECKPOINT_LAST_BEGAN	datetime
Tokudb_CHECKPOINT_LAST_COMPLETE_BEGAN	datetime
Tokudb_CHECKPOINT_LAST_COMPLETE_ENDED	datetime
[Tokudb_CHECKPOINT_DURATION](#tokudbcheckpointduration	integer
Tokudb_CHECKPOINT_DURATION_LAST	integer
Tokudb_CHECKPOINT_LAST_LSN	integer
Tokudb_CHECKPOINT_TAKEN	integer
Tokudb_CHECKPOINT_FAILED	integer
Tokudb_CHECKPOINT_WAITERS_NOW	integer
Tokudb_CHECKPOINT_WAITERS_MAX	integer
Tokudb_CHECKPOINT_CLIENT_WAIT_ON_MO	integer
Tokudb_CHECKPOINT_CLIENT_WAIT_ON_CS	integer
Tokudb_CHECKPOINT_BEGIN_TIME	integer
Tokudb_CHECKPOINT_LONG_BEGIN_TIME	integer
Tokudb_CHECKPOINT_LONG_BEGIN_COUNT	integer
Tokudb_CHECKPOINT_END_TIME	integer
Tokudb_CHECKPOINT_LONG_END_TIME	integer
Tokudb_CHECKPOINT_LONG_END_COUNT	integer
Tokudb_CACHETABLE_MISS	integer
Tokudb_CACHETABLE_MISS_TIME	integer
Tokudb_CACHETABLE_PREFETCHES	integer

Name	Var Type
Tokudb_CACHETABLE_SIZE_CURRENT	integer
Tokudb_CACHETABLE_SIZE_LIMIT	integer
Tokudb_CACHETABLE_SIZE_WRITING	integer
Tokudb_CACHETABLE_SIZE_NONLEAF	integer
Tokudb_CACHETABLE_SIZE_LEAF	integer
Tokudb_CACHETABLE_SIZE_ROLLBACK	integer
Tokudb_CACHETABLE_SIZE_CACHEPRESSURE	integer
Tokudb_CACHETABLE_SIZE_CLONED	integer
Tokudb_CACHETABLE_EVICTIONS	integer
Tokudb_CACHETABLE_CLEANER_EXECUTIONS	integer
Tokudb_CACHETABLE_CLEANER_PERIOD	integer
Tokudb_CACHETABLE_CLEANER_ITERATIONS	integer
Tokudb_CACHETABLE_WAIT_PRESSURE_COUNT	integer
Tokudb_CACHETABLE_WAIT_PRESSURE_TIME	integer
Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_COUNT	integer
Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_TIME	integer
Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS	integer
Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS_ACTIVE	integer
Tokudb_CACHETABLE_POOL_CLIENT_QUEUE_SIZE	integer
Tokudb_CACHETABLE_POOL_CLIENT_MAX_QUEUE_SIZE	integer
Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_ITEMS_PROCESSED	integer
Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_EXECUTION_TIME	integer
Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS	integer
Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS_ACTIVE	integer
Tokudb_CACHETABLE_POOL_CACHETABLE_QUEUE_SIZE	integer
Tokudb_CACHETABLE_POOL_CACHETABLE_MAX_QUEUE_SIZE	integer
Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_ITEMS_PROCESSED	integer
Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_EXECUTION_TIME	integer
Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS	integer
Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS_ACTIVE	integer
Tokudb_CACHETABLE_POOL_CHECKPOINT_QUEUE_SIZE	integer
Tokudb_CACHETABLE_POOL_CHECKPOINT_MAX_QUEUE_SIZE	integer
Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_ITEMS_PROCESSED	integer
Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_EXECUTION_TIME	integer
Tokudb_LOCKTREE_MEMORY_SIZE	integer

Name	Var Type
Tokudb_LOCKTREE_MEMORY_SIZE_LIMIT	integer
Tokudb_LOCKTREE_ESCALATION_NUM	integer
Tokudb_LOCKTREE_ESCALATION_SECONDS	numeric
Tokudb_LOCKTREE_LATEST_POST_ESCALATION_MEMORY_SIZE	integer
Tokudb_LOCKTREE_OPEN_CURRENT	integer
Tokudb_LOCKTREE_PENDING_LOCK_REQUESTS	integer
Tokudb_LOCKTREE_STO_ELIGIBLE_NUM	integer
Tokudb_LOCKTREE_STO_ENDED_NUM	integer
Tokudb_LOCKTREE_STO_ENDED_SECONDS	numeric
Tokudb_LOCKTREE_WAIT_COUNT	integer
Tokudb_LOCKTREE_WAIT_TIME	integer
Tokudb_LOCKTREE_LONG_WAIT_COUNT	integer
Tokudb_LOCKTREE_LONG_WAIT_TIME	integer
Tokudb_LOCKTREE_TIMEOUT_COUNT	integer
Tokudb_LOCKTREE_WAIT_ESCALATION_COUNT	integer
Tokudb_LOCKTREE_WAIT_ESCALATION_TIME	integer
Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_COUNT	integer
Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_TIME	integer
Tokudb_DICTIONARY_UPDATES	integer
Tokudb_DICTIONARY_BROADCAST_UPDATES	integer
Tokudb_DESCRIPTOR_SET	integer
Tokudb_MESSAGES_IGNORED_BY_LEAF_DUE_TO_MSN	integer
Tokudb_TOTAL_SEARCH_RETRIES	integer
Tokudb_SEARCH_TRIES_GT_HEIGHT	integer
Tokudb_SEARCH_TRIES_GT_HEIGHTPLUS3	integer
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT	integer
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_BYTES	integer
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_UNCOMPRESSED_BYTES	integer
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_SECONDS	numeric
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT	integer
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_BYTES	integer
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_UNCOMPRESSE	integer
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_SECONDS	numeric
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT	integer
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_BYTES	integer

Name	Var Type
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_UNCOMPRESSED_BYTES	integer
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_SECONDS	numeric
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT	integer
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_BYTES	integer
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_UNCOMPRESSED_BY	integer
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_SECONDS	numeric
Tokudb_LEAF_NODE_COMPRESSION_RATIO	numeric
Tokudb_NONLEAF_NODE_COMPRESSION_RATIO	numeric
Tokudb_OVERALL_NODE_COMPRESSION_RATIO	numeric
Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS	numeric
Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS_BYTES	integer
Tokudb_LEAF_NODE_PARTIAL_EVICTIONS	integer
Tokudb_LEAF_NODE_PARTIAL_EVICTIONS_BYTES	integer
Tokudb_LEAF_NODE_FULL_EVICTIONS	integer
Tokudb_LEAF_NODE_FULL_EVICTIONS_BYTES	integer
Tokudb_NONLEAF_NODE_FULL_EVICTIONS	integer
Tokudb_NONLEAF_NODE_FULL_EVICTIONS_BYTES	integer
Tokudb_LEAF_NODES_CREATED	integer
Tokudb_NONLEAF_NODES_CREATED	integer
Tokudb_LEAF_NODES_DESTROYED	integer
Tokudb_NONLEAF_NODES_DESTROYED	integer
Tokudb_MESSAGES_INJECTED_AT_ROOT_BYTES	integer
Tokudb_MESSAGES_FLUSHED_FROM_H1_TO_LEAVES_BYTES	integer
Tokudb_MESSAGES_IN_TREES_ESTIMATE_BYTES	integer
Tokudb_MESSAGES_INJECTED_AT_ROOT	integer
Tokudb_BROADCASE_MESSAGES_INJECTED_AT_ROOT	integer
Tokudb_BASEMENTS_DECOMPRESSED_TARGET_QUERY	integer
Tokudb_BASEMENTS_DECOMPRESSED_PRELOCKED_RANGE	integer
Tokudb_BASEMENTS_DECOMPRESSED_PREFETCH	integer
Tokudb_BASEMENTS_DECOMPRESSED_FOR_WRITE	integer
Tokudb_BUFFERS_DECOMPRESSED_TARGET_QUERY	integer
Tokudb_BUFFERS_DECOMPRESSED_PRELOCKED_RANGE	integer
Tokudb_BUFFERS_DECOMPRESSED_PREFETCH	integer
Tokudb_BUFFERS_DECOMPRESSED_FOR_WRITE	integer
Tokudb_PIVOTS_FETCHED_FOR_QUERY	integer

Name	Var Type
Tokudb_PIVOTS_FETCHED_FOR_QUERY_BYTES	integer
Tokudb_PIVOTS_FETCHED_FOR_QUERY_SECONDS	numeric
Tokudb_PIVOTS_FETCHED_FOR_PREFETCH	integer
Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_BYTES	integer
Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_SECONDS	numeric
Tokudb_PIVOTS_FETCHED_FOR_WRITE	integer
Tokudb_PIVOTS_FETCHED_FOR_WRITE_BYTES	integer
Tokudb_PIVOTS_FETCHED_FOR_WRITE_SECONDS	numeric
Tokudb_BASEMENTS_FETCHED_TARGET_QUERY	integer
Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_BYTES	integer
Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_SECONDS	numeric
Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE	integer
Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_BYTES	integer
Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_SECONDS	numeric
Tokudb_BASEMENTS_FETCHED_PREFETCH	integer
Tokudb_BASEMENTS_FETCHED_PREFETCH_BYTES	integer
Tokudb_BASEMENTS_FETCHED_PREFETCH_SECONDS	numeric
Tokudb_BASEMENTS_FETCHED_FOR_WRITE	integer
Tokudb_BASEMENTS_FETCHED_FOR_WRITE_BYTES	integer
Tokudb_BASEMENTS_FETCHED_FOR_WRITE_SECONDS	numeric
Tokudb_BUFFERS_FETCHED_TARGET_QUERY	integer
Tokudb_BUFFERS_FETCHED_TARGET_QUERY_BYTES	integer
Tokudb_BUFFERS_FETCHED_TARGET_QUERY_SECONDS	numeric
Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE	integer
Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_BYTES	integer
Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_SECONDS	numeric
Tokudb_BUFFERS_FETCHED_PREFETCH	integer
Tokudb_BUFFERS_FETCHED_PREFETCH_BYTES	integer
Tokudb_BUFFERS_FETCHED_PREFETCH_SECONDS	numeric
Tokudb_BUFFERS_FETCHED_FOR_WRITE	integer
Tokudb_BUFFERS_FETCHED_FOR_WRITE_BYTES	integer
Tokudb_BUFFERS_FETCHED_FOR_WRITE_SECONDS	integer
Tokudb_LEAF_COMPRESSION_TO_MEMORY_SECONDS	numeric
Tokudb_LEAF_SERIALIZATION_TO_MEMORY_SECONDS	numeric
Tokudb_LEAF_DECOMPRESSION_TO_MEMORY_SECONDS	numeric

Name	Var Type
Tokudb_LEAF_DESERIALIZATION_TO_MEMORY_SECONDS	numeric
Tokudb_NONLEAF_COMPRESSION_TO_MEMORY_SECONDS	numeric
Tokudb_NONLEAF_SERIALIZATION_TO_MEMORY_SECONDS	numeric
Tokudb_NONLEAF_DECOMPRESSION_TO_MEMORY_SECONDS	numeric
Tokudb_NONLEAF_DESERIALIZATION_TO_MEMORY_SECONDS	numeric
Tokudb_PROMOTION_ROOTS_SPLIT	integer
Tokudb_PROMOTION_LEAF_ROOTS_INJECTED_INTO	integer
Tokudb_PROMOTION_H1_ROOTS_INJECTED_INTO	integer
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_0	integer
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_1	integer
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_2	integer
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_3	integer
Tokudb_PROMOTION_INJECTIONS_LOWER_THAN_DEPTH_3	integer
Tokudb_PROMOTION_STOPPED_NONEMPTY_BUFFER	integer
Tokudb_PROMOTION_STOPPED_AT_HEIGHT_1	integer
Tokudb_PROMOTION_STOPPED_CHILD_LOCKED_OR_NOT_IN_MEMORY	integer
Tokudb_PROMOTION_STOPPED_CHILD_NOT_FULLY_IN_MEMORY	integer
Tokudb_PROMOTION_STOPPED_AFTER_LOCKING_CHILD	integer
Tokudb_BASEMENT_DESERIALIZATION_FIXED_KEY	integer
Tokudb_BASEMENT_DESERIALIZATION_VARIABLE_KEY	integer
Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_SUCCESS	integer
Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_FAIL_POS	integer
Tokudb_RIGHTMOST_LEAF_SHORTCUT_FAIL_REACTIVE	integer
Tokudb_CURSOR_SKIP_DELETED_LEAF_ENTRY	integer
Tokudb_FLUSHER_CLEANER_TOTAL_NODES	integer
Tokudb_FLUSHER_CLEANER_H1_NODES	integer
Tokudb_FLUSHER_CLEANER_HGT1_NODES	integer
Tokudb_FLUSHER_CLEANER_EMPTY_NODES	integer
Tokudb_FLUSHER_CLEANER_NODES_DIRTIED	integer
Tokudb_FLUSHER_CLEANER_MAX_BUFFER_SIZE	integer
Tokudb_FLUSHER_CLEANER_MIN_BUFFER_SIZE	integer
Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_SIZE	integer
Tokudb_FLUSHER_CLEANER_MAX_BUFFER_WORKDONE	integer
Tokudb_FLUSHER_CLEANER_MIN_BUFFER_WORKDONE	integer
Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_WORKDONE	integer

Name	Var Type
Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_STARTED	integer
Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_RUNNING	integer
Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_COMPLETED	integer
Tokudb_FLUSHER_CLEANER_NUM_DIRTIED_FOR_LEAF_MERGE	integer
Tokudb_FLUSHER_FLUSH_TOTAL	integer
Tokudb_FLUSHER_FLUSH_IN_MEMORY	integer
Tokudb_FLUSHER_FLUSH_NEEDED_IO	integer
Tokudb_FLUSHER_FLUSH_CASCADES	integer
Tokudb_FLUSHER_FLUSH_CASCADES_1	integer
Tokudb_FLUSHER_FLUSH_CASCADES_2	integer
Tokudb_FLUSHER_FLUSH_CASCADES_3	integer
Tokudb_FLUSHER_FLUSH_CASCADES_4	integer
Tokudb_FLUSHER_FLUSH_CASCADES_5	integer
Tokudb_FLUSHER_FLUSH_CASCADES_GT_5	integer
Tokudb_FLUSHER_SPLIT_LEAF	integer
Tokudb_FLUSHER_SPLIT_NONLEAF	integer
Tokudb_FLUSHER_MERGE_LEAF	integer
Tokudb_FLUSHER_MERGE_NONLEAF	integer
Tokudb_FLUSHER_BALANCE_LEAF	integer
Tokudb_HOT_NUM_STARTED	integer
Tokudb_HOT_NUM_COMPLETED	integer
Tokudb_HOT_NUM_ABORTED	integer
Tokudb_HOT_MAX_ROOT_FLUSH_COUNT	integer
Tokudb_TXN_BEGIN	integer
Tokudb_TXN_BEGIN_READ_ONLY	integer
Tokudb_TXN_COMMITS	integer
Tokudb_TXN_ABORTS	integer
Tokudb_LOGGER_NEXT_LSN	integer
Tokudb_LOGGER_WRITES	integer
Tokudb_LOGGER_WRITES_BYTES	integer
Tokudb_LOGGER_WRITES_UNCOMPRESSED_BYTES	integer
Tokudb_LOGGER_WRITES_SECONDS	numeric
Tokudb_LOGGER_WAIT_LONG	integer
Tokudb_LOADER_NUM_CREATED	integer
Tokudb_LOADER_NUM_CURRENT	integer

Name	Var Type
Tokudb_LOADER_NUM_MAX	integer
Tokudb_MEMORY_MALLOC_COUNT	integer
Tokudb_MEMORY_FREE_COUNT	integer
Tokudb_MEMORY_REALLOC_COUNT	integer
Tokudb_MEMORY_MALLOC_FAIL	integer
Tokudb_MEMORY_REALLOC_FAIL	integer
Tokudb_MEMORY_REQUESTED	integer
Tokudb_MEMORY_USED	integer
Tokudb_MEMORY_FREED	integer
Tokudb_MEMORY_MAX_REQUESTED_SIZE	integer
Tokudb_MEMORY_LAST_FAILED_SIZE	integer
Tokudb_MEM_ESTIMATED_MAXIMUM_MEMORY_FOOTPRINT	integer
Tokudb_MEMORY_MALLOCATOR_VERSION	string
Tokudb_MEMORY_MMAP_THRESHOLD	integer
Tokudb_FILESYSTEM_THREADS_BLOCKED_BY_FULL_DISK	integer
Tokudb_FILESYSTEM_FSYNC_TIME	integer
Tokudb_FILESYSTEM_FSYNC_NUM	integer
Tokudb_FILESYSTEM_LONG_FSYNC_TIME	integer
Tokudb_FILESYSTEM_LONG_FSYNC_NUM	integer

`Tokudb_DB_OPENS`

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.

`Tokudb_DB_CLOSES`

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.

`Tokudb_DB_OPEN_CURRENT`

This variable shows the number of currently opened databases.

`Tokudb_DB_OPEN_MAX`

This variable shows the maximum number of concurrently opened databases.

`Tokudb_LEAF_ENTRY_MAX_COMMITTED_XR`

This variable shows the maximum number of committed transaction records that were stored on disk in a new or modified row.

`Tokudb_LEAF_ENTRY_MAX_PROVISIONAL_XR`

This variable shows the maximum number of provisional transaction records that were stored on disk in a new or modified row.

`Tokudb_LEAF_ENTRY_EXPANDED`

This variable shows the number of times that an expanded memory mechanism was used to store a new or modified row on disk.

`Tokudb_LEAF_ENTRY_MAX_MEMSIZE`

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.

`Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_IN`

This variable shows the total number of bytes of leaf nodes data before performing garbage collection for non-flush events.

`Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_OUT`

This variable shows the total number of bytes of leaf nodes data after performing garbage collection for non-flush events.

`Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_IN`

This variable shows the total number of bytes of leaf nodes data before performing garbage collection for flush events.

`Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_OUT`

This variable shows the total number of bytes of leaf nodes data after performing garbage collection for flush events.

`Tokudb_CHECKPOINT_PERIOD`

This variable shows the interval in seconds between the end of an automatic checkpoint and the beginning of the next automatic checkpoint.

`Tokudb_CHECKPOINT_FOOTPRINT`

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.

`Tokudb_CHECKPOINT_LAST_BEGAN`

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.

`Tokudb_CHECKPOINT_LAST_COMPLETE_BEGAN`

This variable shows the time the last complete checkpoint started. Any data that changed after this time will not be captured in the checkpoint.

`Tokudb_CHECKPOINT_LAST_COMPLETE_ENDED`

This variable shows the time the last complete checkpoint ended.

`Tokudb_CHECKPOINT_DURATION`

This variable shows time (in seconds) required to complete all checkpoints.

`Tokudb_CHECKPOINT_DURATION_LAST`

This variable shows time (in seconds) required to complete the last checkpoint.

`Tokudb_CHECKPOINT_LAST_LSN`

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.

`Tokudb_CHECKPOINT_TAKEN`

This variable shows the number of complete checkpoints that have been taken.

`Tokudb_CHECKPOINT_FAILED`

This variable shows the number of checkpoints that have failed for any reason.

`Tokudb_CHECKPOINT_WAITERS_NOW`

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.

`Tokudb_CHECKPOINT_WAITERS_MAX`

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.

`Tokudb_CHECKPOINT_CLIENT_WAIT_ON_MO`

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.

`Tokudb_CHECKPOINT_CLIENT_WAIT_ON_CS`

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.

`Tokudb_CHECKPOINT_BEGIN_TIME`

This variable shows the cumulative time (in microseconds) required to mark all dirty nodes as pending a checkpoint.

`Tokudb_CHECKPOINT_LONG_BEGIN_TIME`

This variable shows the cumulative actual time (in microseconds) of checkpoint `begin` stages that took longer than 1 second.

`Tokudb_CHECKPOINT_LONG_BEGIN_COUNT`

This variable shows the number of checkpoints whose `begin` stage took longer than 1 second.

`Tokudb_CHECKPOINT_END_TIME`

This variable shows the time spent in checkpoint end operation in seconds.

`Tokudb_CHECKPOINT_LONG_END_TIME`

This variable shows the total time of long checkpoints in seconds.

`Tokudb_CHECKPOINT_LONG_END_COUNT`

This variable shows the number of checkpoints whose `end_checkpoint` operations exceeded 1 minute.

`Tokudb_CACHETABLE_MISS`

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.

`Tokudb_CACHETABLE_MISS_TIME`

This variable shows the total time, in microseconds, of how long the database has had to wait for a disk read to complete.

`Tokudb_CACHETABLE_PREFETCHES`

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.

`Tokudb_CACHETABLE_SIZE_CURRENT`

This variable shows how much of the uncompressed data, in bytes, is currently in the database's internal cache.

`Tokudb_CACHETABLE_SIZE_LIMIT`

This variable shows how much of the uncompressed data, in bytes, will fit in the database's internal cache.

`Tokudb_CACHETABLE_SIZE_WRITING`

This variable shows the number of bytes that are currently queued up to be written to disk.

`Tokudb_CACHETABLE_SIZE_NONLEAF`

This variable shows the amount of memory, in bytes, the current set of non-leaf nodes occupy in the cache.

`Tokudb_CACHETABLE_SIZE_LEAF`

This variable shows the amount of memory, in bytes, the current set of (decompressed) leaf nodes occupy in the cache.

`Tokudb_CACHETABLE_SIZE_ROLLBACK`

This variable shows the rollback nodes size, in bytes, in the cache.

`Tokudb_CACHETABLE_SIZE_CACHEPRESSURE`

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.

`Tokudb_CACHETABLE_SIZE_CLONED`

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.

`Tokudb_CACHETABLE_EVICTIONS`

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.

`Tokudb_CACHETABLE_CLEANER_EXECUTIONS`

This variable shows the total number of times the cleaner thread loop has executed.

`Tokudb_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.

`Tokudb_CACHETABLE_CLEANER_ITERATIONS`

This variable shows the number of cleaner operations that are performed every cleaner period.

`Tokudb_CACHETABLE_WAIT_PRESSURE_COUNT`

This variable shows the number of times a thread was stalled due to cache pressure.

`Tokudb_CACHETABLE_WAIT_PRESSURE_TIME`

This variable shows the total time, in microseconds, waiting on cache pressure to subside.

`Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_COUNT`

This variable shows the number of times a thread was stalled for more than one second due to cache pressure.

`Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_TIME`

This variable shows the total time, in microseconds, waiting on cache pressure to subside for more than one second.

`Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS`

This variable shows the number of threads in the client thread pool.

`Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS_ACTIVE`

This variable shows the number of currently active threads in the client thread pool.

`Tokudb_CACHETABLE_POOL_CLIENT_QUEUE_SIZE`

This variable shows the number of currently queued work items in the client thread pool.

`Tokudb_CACHETABLE_POOL_CLIENT_MAX_QUEUE_SIZE`

This variable shows the largest number of queued work items in the client thread pool.

`Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_ITEMS_PROCESSED`

This variable shows the total number of work items processed in the client thread pool.

`Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_EXECUTION_TIME`

This variable shows the total execution time of processing work items in the client thread pool.

`Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS`

This variable shows the number of threads in the cachetable threadpool.

`Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS_ACTIVE`

This variable shows the number of currently active threads in the cachetable thread pool.

`Tokudb_CACHETABLE_POOL_CACHETABLE_QUEUE_SIZE`

This variable shows the number of currently queued work items in the cachetable thread pool.

`Tokudb_CACHETABLE_POOL_CACHETABLE_MAX_QUEUE_SIZE`

This variable shows the largest number of queued work items in the cachetable thread pool.

`Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_ITEMS_PROCESSED`

This variable shows the total number of work items processed in the cachetable thread pool.

`Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_EXECUTION_TIME`

This variable shows the total execution time of processing work items in the cachetable thread pool.

`Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS`

This variable shows the number of threads in the checkpoint threadpool.

`Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS_ACTIVE`

This variable shows the number of currently active threads in the checkpoint thread pool.

`Tokudb_CACHETABLE_POOL_CHECKPOINT_QUEUE_SIZE`

This variable shows the number of currently queued work items in the checkpoint thread pool.

`Tokudb_CACHETABLE_POOL_CHECKPOINT_MAX_QUEUE_SIZE`

This variable shows the largest number of queued work items in the checkpoint thread pool.

`Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_ITEMS_PROCESSED`

This variable shows the total number of work items processed in the checkpoint thread pool.

`Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_EXECUTION_TIME`

This variable shows the total execution time of processing work items in the checkpoint thread pool.

`Tokudb_LOCKTREE_MEMORY_SIZE`

This variable shows the amount of memory, in bytes, that the locktree is currently using.

`Tokudb_LOCKTREE_MEMORY_SIZE_LIMIT`

This variable shows the maximum amount of memory, in bytes, that the locktree is allowed to use.

`Tokudb_LOCKTREE_ESCALATION_NUM`

This variable shows the number of times the locktree needed to run lock escalation to reduce its memory footprint.

`Tokudb_LOCKTREE_ESCALATION_SECONDS`

This variable shows the total number of seconds spent performing locktree escalation.

`Tokudb_LOCKTREE_LATEST_POST_ESCALATION_MEMORY_SIZE`

This variable shows the locktree size, in bytes, after most current locktree escalation.

`Tokudb_LOCKTREE_OPEN_CURRENT`

This variable shows the number of locktrees that are currently opened.

`Tokudb_LOCKTREE_PENDING_LOCK_REQUESTS`

This variable shows the number of requests waiting for a lock grant.

`Tokudb_LOCKTREE_STO_ELIGIBLE_NUM`

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.

`Tokudb_LOCKTREE_STO_ENDED_NUM`

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.

`Tokudb_LOCKTREE_STO_ENDED_SECONDS`

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.

`Tokudb_LOCKTREE_WAIT_COUNT`

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.

`Tokudb_LOCKTREE_WAIT_TIME`

This variable shows the total time, in microseconds, spent by client waiting for a lock conflict to be resolved.

`Tokudb_LOCKTREE_LONG_WAIT_COUNT`

This variable shows number of lock waits greater than one second in duration.

`Tokudb_LOCKTREE_LONG_WAIT_TIME`

This variable shows the total time, in microseconds, of the long waits.

`Tokudb_LOCKTREE_TIMEOUT_COUNT`

This variable shows the number of times that a lock request timed out.

`Tokudb_LOCKTREE_WAIT_ESCALATION_COUNT`

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.

`Tokudb_LOCKTREE_WAIT_ESCALATION_TIME`

This variable shows the total time, in microseconds, that a client thread spent waiting for lock escalation to free up memory.

`Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_COUNT`

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.

`Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_TIME`

This variable shows the total time, in microseconds, of the long waits for lock escalation to free up memory.

`Tokudb_DICTIONARY_UPDATES`

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.

`Tokudb_DICTIONARY_BROADCAST_UPDATES`

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.

`Tokudb_DESCRIPTOR_SET`

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).

`Tokudb_MESSAGES_IGNORED_BY_LEAF_DUE_TO_MSN`

This variable shows the number of messages that were ignored by a leaf because it had already been applied.

`Tokudb_TOTAL_SEARCH_RETRIES`

Internal value that is no use to anyone other than a developer debugging a specific query/search issue.

`Tokudb_SEARCH_TRIES_GT_HEIGHT`

Internal value that is no use to anyone other than a developer debugging a specific query/search issue.

`Tokudb_SEARCH_TRIES_GT_HEIGHTPLUS3`

Internal value that is no use to anyone other than a developer debugging a specific query/search issue.

`Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT`

This variable shows the number of leaf nodes flushed to disk, not for checkpoint.

`Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_BYTES`

This variable shows the size, in bytes, of leaf nodes flushed to disk, not for checkpoint.

`Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_UNCOMPRESSED_BYTES`

This variable shows the size, in bytes, of uncompressed leaf nodes flushed to disk not for checkpoint.

`Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_SECONDS`

This variable shows the number of seconds waiting for I/O when writing leaf nodes flushed to disk, not for checkpoint

`Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT`

This variable shows the number of non-leaf nodes flushed to disk, not for checkpoint.

`Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_BYTES`

This variable shows the size, in bytes, of non-leaf nodes flushed to disk, not for checkpoint.

`Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_UNCOMPRESSE`

This variable shows the size, in bytes, of uncompressed non-leaf nodes flushed to disk not for checkpoint.

`Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_SECONDS`

This variable shows the number of seconds waiting for I/O when writing non-leaf nodes flushed to disk, not for checkpoint

`Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT`

This variable shows the number of leaf nodes flushed to disk, for checkpoint.

`Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_BYTES`

This variable shows the size, in bytes, of leaf nodes flushed to disk, for checkpoint.

`Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_UNCOMPRESSED_BYTES`

This variable shows the size, in bytes, of uncompressed leaf nodes flushed to disk for checkpoint.

`Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_SECONDS`

This variable shows the number of seconds waiting for I/O when writing leaf nodes flushed to disk for checkpoint

`Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT`

This variable shows the number of non-leaf nodes flushed to disk, for checkpoint.

`Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_BYTES`

This variable shows the size, in bytes, of non-leaf nodes flushed to disk, for checkpoint.

`Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_UNCOMPRESSED_BY`

This variable shows the size, in bytes, of uncompressed non-leaf nodes flushed to disk for checkpoint.

`Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_SECONDS`

This variable shows the number of seconds waiting for I/O when writing non-leaf nodes flushed to disk for checkpoint

`Tokudb_LEAF_NODE_COMPRESSION_RATIO`

This variable shows the ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for leaf nodes.

`Tokudb_NONLEAF_NODE_COMPRESSION_RATIO`

This variable shows the ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for non-leaf nodes.

`Tokudb_OVERALL_NODE_COMPRESSION_RATIO`

This variable shows the ratio of uncompressed bytes (in-memory) to compressed bytes (on-disk) for all nodes.

`Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS`

This variable shows the number of times a partition of a non-leaf node was evicted from the cache.

`Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS_BYTES`

This variable shows the amount, in bytes, of memory freed by evicting partitions of non-leaf nodes from the cache.

`Tokudb_LEAF_NODE_PARTIAL_EVICTIONS`

This variable shows the number of times a partition of a leaf node was evicted from the cache.

`Tokudb_LEAF_NODE_PARTIAL_EVICTIONS_BYTES`

This variable shows the amount, in bytes, of memory freed by evicting partitions of leaf nodes from the cache.

`Tokudb_LEAF_NODE_FULL_EVICTIONS`

This variable shows the number of times a full leaf node was evicted from the cache.

`Tokudb_LEAF_NODE_FULL_EVICTIONS_BYTES`

This variable shows the amount, in bytes, of memory freed by evicting full leaf nodes from the cache.

`Tokudb_NONLEAF_NODE_FULL_EVICTIONS`

This variable shows the number of times a full non-leaf node was evicted from the cache.

`Tokudb_NONLEAF_NODE_FULL_EVICTIONS_BYTES`

This variable shows the amount, in bytes, of memory freed by evicting full non-leaf nodes from the cache.

`Tokudb_LEAF_NODES_CREATED`

This variable shows the number of created leaf nodes.

`Tokudb_NONLEAF_NODES_CREATED`

This variable shows the number of created non-leaf nodes.

`Tokudb_LEAF_NODES_DESTROYED`

This variable shows the number of destroyed leaf nodes.

`Tokudb_NONLEAF_NODES_DESTROYED`

This variable shows the number of destroyed non-leaf nodes.

`Tokudb_MESSAGES_INJECTED_AT_ROOT_BYTES`

This variable shows the size, in bytes, of messages injected at root (for all trees).

`Tokudb_MESSAGES_FLUSHED_FROM_H1_TO_LEAVES_BYTES`

This variable shows the size, in bytes, of messages flushed from `h1` nodes to leaves.

`Tokudb_MESSAGES_IN_TREES_ESTIMATE_BYTES`

This variable shows the estimated size, in bytes, of messages currently in trees.

`Tokudb_MESSAGES_INJECTED_AT_ROOT`

This variables shows the number of messages that were injected at root node of a tree.

`Tokudb_BROADCASE_MESSAGES_INJECTED_AT_ROOT`

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.

`Tokudb_BASEMENTS_DECOMPRESSED_TARGET_QUERY`

This variable shows the number of basement nodes decompressed for queries.

`Tokudb_BASEMENTS_DECOMPRESSED_PRELOCKED_RANGE`

This variable shows the number of basement nodes aggressively decompressed by queries.

`Tokudb_BASEMENTS_DECOMPRESSED_PREFETCH`

This variable shows the number of basement nodes decompressed by a prefetch thread.

`Tokudb_BASEMENTS_DECOMPRESSED_FOR_WRITE`

This variable shows the number of basement nodes decompressed for writes.

`Tokudb_BUFFERS_DECOMPRESSED_TARGET_QUERY`

This variable shows the number of buffers decompressed for queries.

`Tokudb_BUFFERS_DECOMPRESSED_PRELOCKED_RANGE`

This variable shows the number of buffers decompressed by queries aggressively.

`Tokudb_BUFFERS_DECOMPRESSED_PREFETCH`

This variable shows the number of buffers decompressed by a prefetch thread.

`Tokudb_BUFFERS_DECOMPRESSED_FOR_WRITE`

This variable shows the number of buffers decompressed for writes.

`Tokudb_PIVOTS_FETCHED_FOR_QUERY`

This variable shows the number of pivot nodes fetched for queries.

`Tokudb_PIVOTS_FETCHED_FOR_QUERY_BYTES`

This variable shows the number of bytes of pivot nodes fetched for queries.

`Tokudb_PIVOTS_FETCHED_FOR_QUERY_SECONDS`

This variable shows the number of seconds waiting for I/O when fetching pivot nodes for queries.

`Tokudb_PIVOTS_FETCHED_FOR_PREFETCH`

This variable shows the number of pivot nodes fetched by a prefetch thread.

`Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_BYTES`

This variable shows the number of bytes of pivot nodes fetched for queries.

`Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_SECONDS`

This variable shows the number seconds waiting for I/O when fetching pivot nodes by a prefetch thread.

`Tokudb_PIVOTS_FETCHED_FOR_WRITE`

This variable shows the number of pivot nodes fetched for writes.

`Tokudb_PIVOTS_FETCHED_FOR_WRITE_BYTES`

This variable shows the number of bytes of pivot nodes fetched for writes.

`Tokudb_PIVOTS_FETCHED_FOR_WRITE_SECONDS`

This variable shows the number of seconds waiting for I/O when fetching pivot nodes for writes.

`Tokudb_BASEMENTS_FETCHED_TARGET_QUERY`

This variable shows the number of basement nodes fetched from disk for queries.

`Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_BYTES`

This variable shows the number of basement node bytes fetched from disk for queries.

`Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_SECONDS`

This variable shows the number of seconds waiting for I/O when fetching basement nodes from disk for queries.

`Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE`

This variable shows the number of basement nodes fetched from disk aggressively.

`Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_BYTES`

This variable shows the number of basement node bytes fetched from disk aggressively.

`Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_SECONDS`

This variable shows the number of seconds waiting for I/O when fetching basement nodes from disk aggressively.

`Tokudb_BASEMENTS_FETCHED_PREFETCH`

This variable shows the number of basement nodes fetched from disk by a prefetch thread.

`Tokudb_BASEMENTS_FETCHED_PREFETCH_BYTES`

This variable shows the number of basement node bytes fetched from disk by a prefetch thread.

`Tokudb_BASEMENTS_FETCHED_PREFETCH_SECONDS`

This variable shows the number of seconds waiting for I/O when fetching basement nodes from disk by a prefetch thread.

`Tokudb_BASEMENTS_FETCHED_FOR_WRITE`

This variable shows the number of buffers fetched from disk for writes.

`Tokudb_BASEMENTS_FETCHED_FOR_WRITE_BYTES`

This variable shows the number of buffer bytes fetched from disk for writes.

`Tokudb_BASEMENTS_FETCHED_FOR_WRITE_SECONDS`

This variable shows the number of seconds waiting for I/O when fetching buffers from disk for writes.

`Tokudb_BUFFERS_FETCHED_TARGET_QUERY`

This variable shows the number of buffers fetched from disk for queries.

`Tokudb_BUFFERS_FETCHED_TARGET_QUERY_BYTES`

This variable shows the number of buffer bytes fetched from disk for queries.

`Tokudb_BUFFERS_FETCHED_TARGET_QUERY_SECONDS`

This variable shows the number of seconds waiting for I/O when fetching buffers from disk for queries.

`Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE`

This variable shows the number of buffers fetched from disk aggressively.

`Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_BYTES`

This variable shows the number of buffer bytes fetched from disk aggressively.

`Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_SECONDS`

This variable shows the number of seconds waiting for I/O when fetching buffers from disk aggressively.

`Tokudb_BUFFERS_FETCHED_PREFETCH`

This variable shows the number of buffers fetched from disk aggressively.

`Tokudb_BUFFERS_FETCHED_PREFETCH_BYTES`

This variable shows the number of buffer bytes fetched from disk by a prefetch thread.

`Tokudb_BUFFERS_FETCHED_PREFETCH_SECONDS`

This variable shows the number of seconds waiting for I/O when fetching buffers from disk by a prefetch thread.

`Tokudb_BUFFERS_FETCHED_FOR_WRITE`

This variable shows the number of buffers fetched from disk for writes.

`Tokudb_BUFFERS_FETCHED_FOR_WRITE_BYTES`

This variable shows the number of buffer bytes fetched from disk for writes.

`Tokudb_BUFFERS_FETCHED_FOR_WRITE_SECONDS`

This variable shows the number of seconds waiting for I/O when fetching buffers from disk for writes.

`Tokudb_LEAF_COMPRESSION_TO_MEMORY_SECONDS`

This variable shows the total time, in seconds, spent compressing leaf nodes.

`Tokudb_LEAF_SERIALIZATION_TO_MEMORY_SECONDS`

This variable shows the total time, in seconds, spent serializing leaf nodes.

`Tokudb_LEAF_DECOMPRESSION_TO_MEMORY_SECONDS`

This variable shows the total time, in seconds, spent decompressing leaf nodes.

`Tokudb_LEAF_DESERIALIZATION_TO_MEMORY_SECONDS`

This variable shows the total time, in seconds, spent deserializing leaf nodes.

`Tokudb_NONLEAF_COMPRESSION_TO_MEMORY_SECONDS`

This variable shows the total time, in seconds, spent compressing non leaf nodes.

`Tokudb_NONLEAF_SERIALIZATION_TO_MEMORY_SECONDS`

This variable shows the total time, in seconds, spent serializing non leaf nodes.

`Tokudb_NONLEAF_DECOMPRESSION_TO_MEMORY_SECONDS`

This variable shows the total time, in seconds, spent decompressing non leaf nodes.

`Tokudb_NONLEAF_DESERIALIZATION_TO_MEMORY_SECONDS`

This variable shows the total time, in seconds, spent deserializing non leaf nodes.

`Tokudb_PROMOTION_ROOTS_SPLIT`

This variable shows the number of times the root split during promotion.

`Tokudb_PROMOTION_LEAF_ROOTS_INJECTED_INTO`

This variable shows the number of times a message stopped at a root with height `0`.

`Tokudb_PROMOTION_H1_ROOTS_INJECTED_INTO`

This variable shows the number of times a message stopped at a root with height `1`.

`Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_0`

This variable shows the number of times a message stopped at depth `0`.

`Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_1`

This variable shows the number of times a message stopped at depth `1`.

`Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_2`

This variable shows the number of times a message stopped at depth `2`.

`Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_3`

This variable shows the number of times a message stopped at depth `3`.

`Tokudb_PROMOTION_INJECTIONS_LOWER_THAN_DEPTH_3`

This variable shows the number of times a message was promoted past depth `3`.

`Tokudb_PROMOTION_STOPPED_NONEMPTY_BUFFER`

This variable shows the number of times a message stopped because it reached a nonempty buffer.

`Tokudb_PROMOTION_STOPPED_AT_HEIGHT_1`

This variable shows the number of times a message stopped because it had reached height `1`.

`Tokudb_PROMOTION_STOPPED_CHILD_LOCKED_OR_NOT_IN_MEMORY`

This variable shows the number of times a message stopped because it could not cheaply get access to a child.

`Tokudb_PROMOTION_STOPPED_CHILD_NOT_FULLY_IN_MEMORY`

This variable shows the number of times a message stopped because it could not cheaply get access to a child.

`Tokudb_PROMOTION_STOPPED_AFTER_LOCKING_CHILD`

This variable shows the number of times a message stopped before a child which had been locked.

`Tokudb_BASEMENT_DESERIALIZATION_FIXED_KEY`

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.

`Tokudb_BASEMENT_DESERIALIZATION_VARIABLE_KEY`

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.

`Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_SUCCESS`

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.

`Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_FAIL_POS`

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.

`Tokudb_RIGHTMOST_LEAF_SHORTCUT_FAIL_REACTIVE`

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.

`Tokudb_CURSOR_SKIP_DELETED_LEAF_ENTRY`

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.

`Tokudb_FLUSHER_CLEANER_TOTAL_NODES`

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.

`Tokudb_FLUSHER_CLEANER_H1_NODES`

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.

`Tokudb_FLUSHER_CLEANER_HGT1_NODES`

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.

`Tokudb_FLUSHER_CLEANER_EMPTY_NODES`

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.

`Tokudb_FLUSHER_CLEANER_NODES_DIRTIED`

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.

`Tokudb_FLUSHER_CLEANER_MAX_BUFFER_SIZE`

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.

`Tokudb_FLUSHER_CLEANER_MIN_BUFFER_SIZE`

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.

`Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_SIZE`

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.

`Tokudb_FLUSHER_CLEANER_MAX_BUFFER_WORKDONE`

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.

`Tokudb_FLUSHER_CLEANER_MIN_BUFFER_WORKDONE`

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.

`Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_WORKDONE`

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.

`Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_STARTED`

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.

`Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_RUNNING`

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.

`Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_COMPLETED`

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.

`Tokudb_FLUSHER_CLEANER_NUM_DIRTIED_FOR_LEAF_MERGE`

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.

`Tokudb_FLUSHER_FLUSH_TOTAL`

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.

`Tokudb_FLUSHER_FLUSH_IN_MEMORY`

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.

`Tokudb_FLUSHER_FLUSH_NEEDED_IO`

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.

`Tokudb_FLUSHER_FLUSH_CASCADES`

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.

`Tokudb_FLUSHER_FLUSH_CASCADES_1`

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.

`Tokudb_FLUSHER_FLUSH_CASCADES_2`

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.

`Tokudb_FLUSHER_FLUSH_CASCADES_3`

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.

`Tokudb_FLUSHER_FLUSH_CASCADES_4`

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.

`Tokudb_FLUSHER_FLUSH_CASCADES_5`

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.

`Tokudb_FLUSHER_FLUSH_CASCADES_GT_5`

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.

`Tokudb_FLUSHER_SPLIT_LEAF`

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.

`Tokudb_FLUSHER_SPLIT_NONLEAF`

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.

`Tokudb_FLUSHER_MERGE_LEAF`

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.

`Tokudb_FLUSHER_MERGE_NONLEAF`

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.

`Tokudb_FLUSHER_BALANCE_LEAF`

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.

`Tokudb_HOT_NUM_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.

`Tokudb_HOT_NUM_COMPLETED`

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.

`Tokudb_HOT_NUM_ABORTED`

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.

`Tokudb_HOT_MAX_ROOT_FLUSH_COUNT`

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.

`Tokudb_TXN_BEGIN`

This variable shows the number of transactions that have been started.

`Tokudb_TXN_BEGIN_READ_ONLY`

This variable shows the number of read-only transactions started.

`Tokudb_TXN_COMMITS`

This variable shows the total number of transactions that have been committed.

`Tokudb_TXN_ABORTS`

This variable shows the total number of transactions that have been aborted.

`Tokudb_LOGGER_NEXT_LSN`

This variable shows the recovery logger next LSN. This is a not a useful value for a regular user to use for any purpose.

`Tokudb_LOGGER_WRITES`

This variable shows the number of times the logger has written to disk.

`Tokudb_LOGGER_WRITES_BYTES`

This variable shows the number of bytes the logger has written to disk.

`Tokudb_LOGGER_WRITES_UNCOMPRESSED_BYTES`

This variable shows the number of uncompressed bytes the logger has written to disk.

`Tokudb_LOGGER_WRITES_SECONDS`

This variable shows the number of seconds waiting for IO when writing logs to disk.

`Tokudb_LOGGER_WAIT_LONG`

This variable shows the number of times a logger write operation required 100ms or more.

`Tokudb_LOADER_NUM_CREATED`

This variable shows the number of times one of our internal objects, a loader, has been created.

`Tokudb_LOADER_NUM_CURRENT`

This variable shows the number of loaders that currently exist.

`Tokudb_LOADER_NUM_MAX`

This variable shows the maximum number of loaders that ever existed simultaneously.

`Tokudb_MEMORY_MALLOC_COUNT`

This variable shows the number of `malloc` operations by PerconaFT.

`Tokudb_MEMORY_FREE_COUNT`

This variable shows the number of `free` operations by PerconaFT.

`Tokudb_MEMORY_REALLOC_COUNT`

This variable shows the number of `realloc` operations by PerconaFT.

`Tokudb_MEMORY_MALLOC_FAIL`

This variable shows the number of `malloc` operations that failed by PerconaFT.

`Tokudb_MEMORY_REALLOC_FAIL`

This variable shows the number of `realloc` operations that failed by PerconaFT.

`Tokudb_MEMORY_REQUESTED`

This variable shows the number of bytes requested by PerconaFT.

`Tokudb_MEMORY_USED`

This variable shows the number of bytes used (requested + overhead) by PerconaFT.

`Tokudb_MEMORY_FREED`

This variable shows the number of bytes freed by PerconaFT.

`Tokudb_MEMORY_MAX_REQUESTED_SIZE`

This variable shows the largest attempted allocation size by PerconaFT.

`Tokudb_MEMORY_LAST_FAILED_SIZE`

This variable shows the size of the last failed allocation attempt by PerconaFT.

`Tokudb_MEM_ESTIMATED_MAXIMUM_MEMORY_FOOTPRINT`

This variable shows the maximum memory footprint of the storage engine, the max value of (used - freed).

`Tokudb_MEMORY_MALLOCATOR_VERSION`

This variable shows the version of the memory allocator library detected by PerconaFT.

`Tokudb_MEMORY_MMAP_THRESHOLD`

This variable shows the `mmap` threshold in PerconaFT, anything larger than this gets `mmap'ed`.

`Tokudb_FILESYSTEM_THREADS_BLOCKED_BY_FULL_DISK`

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.

`Tokudb_FILESYSTEM_FSYNC_TIME`

This variable shows the total time, in microseconds, used to `fsync` to disk.

`Tokudb_FILESYSTEM_FSYNC_NUM`

This variable shows the total number of times the database has flushed the operating system's file buffers to disk.

`Tokudb_FILESYSTEM_LONG_FSYNC_TIME`

This variable shows the total time, in microseconds, used to `fsync` to dis k when the operation required more than one second.

`Tokudb_FILESYSTEM_LONG_FSYNC_NUM`

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.

Last update: 2022-09-27

Download PDF

15.10 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.

15.10.1 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.

15.10.2 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.

Last update: 2022-09-27

Download PDF

15.11 TokuDB Troubleshooting

15.11.1 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 replica table may not match the auto-increment values on the source 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 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

Execute this command after every reboot because the default is `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 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 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.

Group Replication: TokuDB storage engine doesn't support Group Replication.

15.11.2 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 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 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 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 tokudb_lock_timeout_debug session variable

The 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 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 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 tokudb_last_lock_timeout session variable

The 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 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 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);
```

The output could be:

```
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

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

The output could be:

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

The output could be:

```
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;
```

The output could be:

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

The output could be:

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

The output could be:

```
+-----------------------------------------------------------------------------------------------------
+
|
@@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;
```

The output could be:

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

15.11.3 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 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.
```


Last update: 2022-09-27

Download PDF

15.12 TokuDB Performance Schema Integration

In *Percona Server for MySQL* Percona Server 5.7.20-18 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 PERFORMANCE_SCHEMA.SETUP_INSTRUMENTS table:

```
mysqlSELECT * FROM performance_schema.setup_instruments WHERE NAME LIKE "%/fti/%";
```

The output could be the following:

```
+------------------------------------------------------------+---------+-------+
| 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 PERFORMANCE_SCHEMA.events_waits_summary_global_by_event_name like:

```
mysqlSELECT * FROM performance_schema.events_waits_summary_global_by_event_name WHERE
EVENT_NAME LIKE '%fti%';
```

The output could be the following:

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


Last update: 2022-09-27

Download PDF

15.13 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.

Note

This feature is currently considered *Experimental*

15.13.1 Installing From Binaries

TokuBackup is included with *Percona Server for MySQL* Percona Server for MySQL 5.7.10-1 and later versions. Installation can be performed with the `ps-admin` script.

To install *Percona TokuBackup*:

1. Run `ps-admin --enable-tokubackup` to add the `preload-hotbackup` option into **[mysqld_safe]** section of `my.cnf`.

   ```
   shell
   $ sudo ps-admin --enable-tokubackup
   ```

   The output should be the following:

   ```text 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

   ```
   shell
   $ sudo service mysql restart
   ```

3. Run `ps-admin --enable-tokubackup` again to finish installation of TokuBackup plugin

   ```
   shell
   $ sudo ps-admin --enable-tokubackup
   ```

   The output should be the following:

   ```text

   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. ```

### 15.13.2 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:

```
mysqlset tokudb_backup_dir='/path_to_empty_directory';
```

Note

Setting the 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

If you get any error on this step (for example, caused by some misconfiguration), the Reporting Errors section explains how to find out the reason.

15.13.3 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 (tokudb_data_dir) or TokuDB log directory (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
```

15.13.4 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 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:

```
mysqlSET tokudb_backup_exclude='/lost\\+found($|/)';
```

Note

In *Percona Server for MySQL* Percona Server for MySQL 5.7.10-3 to address bug #125, server `pid` file is excluded by default. If you're providing your own additions to the exclusions and have the `pid` file in the default location, you will need to add the mysqld_safe.pid entry.

Throttling Backup Rate

You can throttle the backup rate using the 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.

```
mysqlSET tokudb_backup_throttle=1000000;
```

Restricting Backup Target

You can restrict the location of the destination directory where the backups can be located using the 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:

```
mysqlSHOW VARIABLES LIKE 'tokudb_backup_allowed_prefix';
```

The output should be the following:

```
+------------------------------+-----------+
| Variable_name                | Value     |
+------------------------------+-----------+
| tokudb_backup_allowed_prefix | /dumpdir  |
+------------------------------+-----------+
```

Reporting Errors

Percona TokuBackup uses two variables to capture errors. They are tokudb_backup_last_error and 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:

```
mysqlSET tokudb_backup_dir='/tmp/backupdir';
```

The output could be the following:

```
ERROR 1231 (42000): Variable 'tokudb_backup_dir' can't be set to the value of '/tmp/
backupdir'
```

```
mysqlSELECT @@tokudb_backup_last_error;
```

The output should be the following:

```
+----------------------------+
| @@tokudb_backup_last_error |
+----------------------------+
|                         17 |
+----------------------------+
```

```
mysqlSELECT @@tokudb_backup_last_error_string;
```

The output should be the following:

```
+--------------------------------------------------+
| @@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 replicas. You must know the last executed global transaction identifier (GTID) or binary log position both for the replica and source 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 replica and source. You can use this information to start a new replica from the source or replica.

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 replica 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 –tokudb-backup-safe-slave, which enables or disables the safe-slave mode, and –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 replica SQL thread is stopped and checked to see if temporary tables produced by the replica exist or do not exist. If temporary tables exist, the replica 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.

15.13.5 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 datadir and optionally the tokudb_data_dir, 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.
- 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.
- TokuBackup does not backup MySQL configuration file(s).

- TokuBackup does not backup tablespaces if they are out of 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.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

15.14 Frequently Asked Questions

This section contains frequently asked questions regarding TokuDB and related software.

15.14.1 Transactional Operations

What transactional operations does TokuDB support?

TokuDB supports `BEGIN TRANSACTION`, `END TRANSACTION`, `COMMIT`, `ROLLBACK`, `SAVEPOINT`, and `RELEASE SAVEPOINT`.

15.14.2 TokuDB and the File System

How can I determine which files belong to the various tables and indexes in my schemas?

The 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).

```
mysqlSELECT * FROM information_schema.tokudb_file_map;
```

The output should be similar to the

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

15.14.3 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 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 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.

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 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.

15.14.4 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 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 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 tokudb_data_dir, 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.

15.14.5 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.

15.14.6 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.

15.14.7 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.

15.14.8 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.

15.14.9 Row Size

What is the maximum row size?

The maximum row size is 32 MiB.

15.14.10 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.

15.14.11 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.

15.14.12 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.

15.14.13 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.

15.14.14 Foreign Keys

Does TokuDB enforce foreign key constraints?

No, TokuDB ignores foreign key declarations.

15.14.15 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.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

15.15 Removing TokuDB storage engine

In case you want remove the TokuDB storage engine from *Percona Server for MySQL* without causing any errors, the following is the recommended procedure:

15.15.1 Change the tables from TokuDB to InnoDB

If you still need the data in the TokuDB tables you must alter the tables to another supported storage engine, i.e., InnoDB:

```
mysql> ALTER TABLE City ENGINE=InnoDB;
```

Note

Removing the TokuDB storage engine before you have changed your tables to another supported storage engine closes access to that data. You must re-install the TokuDB storage engine to regain access.

15.15.2 Removing the plugins

One option is to remove the TokuDB storage engine with all installed plugins by running the `ps-admin` script:

```
ps-admin --disable-tokudb -uroot -pPassw0rd
```

The 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
>> Successfully removed thp-setting=never option from /etc/mysql/my.cnf

Uninstalling TokuDB plugin...
>> Successfully uninstalled TokuDB plugin.
```

Note

The ps-admin removal may not restore the Transparent Huge Pages (THP) to the original operating system default state. You may have the following result:

```
$ cat /sys/kernel/mm/transparent_hugepage/enabled
```

The output could be the following:

```
always madvise [never]
```

On many operating systems, the default state is `[always]`. To enable transparent huge pages, run the following:

```
echo always > /sys/kernel/mm/transparent_hugepage/enabled
```

Another option is to manually remove the TokuDB storage engine with all installed plugins:

```
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;
UNINSTALL PLUGIN tokudb_background_job_status;
```

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-57.x86_64
```

or

```
root@wheezy:~# apt remove percona-server-tokudb-5.7
```

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 will not start.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

16. Release notes

Download PDF

16.1 Percona Server for MySQL 5.7 release notes index

- Percona Server for MySQL 5.7.17-13 (2017-04-05)
- Percona Server for MySQL 5.7.17-12 (2017-03-24)
- Percona Server for MySQL 5.7.17-11 (2017-02-03)
- Percona Server for MySQL 5.7.16-10 (2016-11-28)
- Percona Server for MySQL 5.7.15-9 (2016-10-21)
- Percona Server for MySQL 5.7.14-8 (2016-09-21)
- Percona Server for MySQL 5.7.14-7 (2016-08-23)
- Percona Server for MySQL 5.7.13-6 (2016-07-16)
- Percona Server for MySQL 5.7.12-5 (2016-06-06)
- Percona Server for MySQL 5.7.11-4 (2016-03-15)
- Percona Server for MySQL 5.7.10-3 (2016-02-23)
- Percona Server for MySQL 5.7.10-2 (2016-02-05)
- Percona Server for MySQL 5.7.10-1 (2015-12-14)

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-12-05

Download PDF

16.2 Percona Server for MySQL 5.7.44-48 (2023-12-05)

Important

Percona Server for MySQL 5.7.44-48 is the final release of the Percona Server for MySQL 5.7 series.

We recommend that you either upgrade to MySQL 8.0 or stay on 5.7; we'll support you.

Percona Server for MySQL 5.7.44-48 includes all the features and bug fixes available in MySQL 5.7.44 Community Edition in addition to enterprise-grade features developed by Percona.

16.2.1 Release highlights

Percona Server for MySQL implements telemetry that fills the gaps in our understanding of how you use Percona Server to improve our products. Participation in the anonymous program is optional. You can opt out if you prefer not to share this information. Find more details in the Telemetry on Percona Server for MySQL document.

Improvements and bug fixes provided by Oracle for MySQL 5.7.44 and included in Percona Server for MySQL are the following:

- Upgraded the linked Open SSL library to OpenSSL 3.0.10
- Removed the printed query string limit to display the characters for a detected deadlock section of the engine status log

Find the complete list of additions or bug fixes in the MySQL 5.7.44 Release Notes.

16.2.2 Bug fixes

- PS-7806: Working with tables with column compression broke asynchronous replication. Using column compression on the tables reduces the storage space and the memory usage of the data but also increases the CPU overhead and complexity. Asynchronous replication sends data from one node to another without waiting for acknowledgment, which improves performance and source availability but also risks data loss and inconsistency. With column compression, the replica must decompress the data in the columns, which can cause errors, delays, or conflicts.
- PS-8879: A large table received an out-of-memory error when running `ALTER TABLE ... COLUMN_FORMAT COMPRESSED;` because the internal DDL copy operation consumed the available memory (thank you to minghuan zhao for your contribution).

16.2.3 Useful links

Install Percona Server for MySQL

The Percona Server for MySQL GitHub repository

Contribute to the documentation

Download product binaries, packages, and tarballs at Percona Product Downloads

For training, contact Percona Training - Start learning now

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-12-05

Download PDF

16.3 Percona Server for MySQL 5.7.43-47 (2023-08-17)

Percona Server for MySQL 5.7.43-47 includes all the features and bug fixes available in MySQL 5.7.43 Community Edition in addition to enterprise-grade features developed by Percona.

Percona Server for MySQL is a freely available, fully compatible, enhanced, and open source drop-in replacement for any MySQL database. It provides superior and optimized performance, greater scalability and availability, enhanced backups, increased visibility, and instrumentation .

Percona Server for MySQL is trusted by thousands of enterprises to provide better performance and concurrency for their most demanding workloads.

16.3.1 Release highlights

Improvements and bug fixes provided by Oracle for MySQL 5.7.43 and included in Percona Server for MySQL are the following:

- OpenSSL 1.1.1 library has been upgraded to OpenSSL 3.0.9.

Find the full list of bug fixes and changes in the MySQL 5.7.43 Release Notes.

16.3.2 Bug fixes

This release merges the MySQL 5.7.43 code base. This release does not contain new improvements or new bug fixes from Percona.

16.3.3 Useful links

Install Percona Server for MySQL

The Percona Server for MySQL GitHub location

Contribute to the documentation

Download product binaries, packages, and tarballs at Percona Product Downloads

For training, contact Percona Training - Start learning now

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-08-18

Download PDF

16.4 Percona Server for MySQL 5.7.42-46 (2023-06-01)

Release date	**June 01, 2023**
Install instructions	Install Percona Server for MySQL

Percona Server for MySQL 5.7.42-46 includes all the features and bug fixes available in MySQL 5.7.42 Community Edition in addition to enterprise-grade features developed by Percona.

Percona Server for MySQL is a freely available, fully compatible, enhanced, and open source drop-in replacement for any MySQL database. It provides superior and optimized performance, greater scalability and availability, enhanced backups, increased visibility, and instrumentation .

Percona Server for MySQL is trusted by thousands of enterprises to provide better performance and concurrency for their most demanding workloads.

16.4.1 Release highlights

Improvements and bug fixes provided by Oracle for MySQL 5.7.42 and included in Percona Server for MySQL are the following:

- In InnoDB, online DDL operations are prevented from accessing out-of-bounds memory

Find the full list of bug fixes and changes in the MySQL 5.7.42 Release Notes.

16.4.2 Bug fixes

- PS-8776: The `mysqldump` client utility ignored `--set-gtid-purged=OFF`.

16.4.3 Useful links

The Percona Server for MySQL GitHub location

Contribute to the documentation

Download product binaries, packages, and tarballs at Percona Product Downloads

For training, contact Percona Training - Start learning now

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-05-30

Download PDF

16.5 Percona Server for MySQL 5.7.42-45 (2023-05-23)

Release date	**May 23, 2023**
Install instructions	Install Percona Server for MySQL

Percona Server for MySQL 5.7.42-45 includes all the features and bug fixes available in MySQL 5.7.42 Community Edition in addition to enterprise-grade features developed by Percona.

Percona Server for MySQL is a freely available, fully compatible, enhanced, and open source drop-in replacement for any MySQL database. It provides superior and optimized performance, greater scalability and availability, enhanced backups, increased visibility, and instrumentation .

Percona Server for MySQL is trusted by thousands of enterprises to provide better performance and concurrency for their most demanding workloads.

16.5.1 Release highlights

Improvements and bug fixes provided by Oracle for MySQL 5.7.42 and included in Percona Server for MySQL are the following:

- In InnoDB, online DDL operations are prevented from accessing out-of-bounds memory

Find the full list of bug fixes and changes in the MySQL 5.7.42 Release Notes.

16.5.2 Bug fixes

- PS-8719: Audit log plugin stalled on flush.
- PS-8749: Fixed bad compression stats in INFORMATION_SCHEMA.INNODB_CMP.

16.5.3 Useful links

The Percona Server for MySQL GitHub location

Contribute to the documentation

Download product binaries, packages, and tarballs at Percona Product Downloads

For training, contact Percona Training - Start learning now

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-05-17

Download PDF

16.6 Percona Server for MySQL 5.7.41-44 (2023-03-02)

Release date	**March 2, 2023**
Install instructions	Install Percona Server for MySQL

Percona Server for MySQL 5.7.41-44 includes all the features and bug fixes available in MySQL 5.7.41 Community Edition in addition to enterprise-grade features developed by Percona.

Percona Server for MySQL is a free, fully compatible, enhanced, and open source drop-in replacement for any MySQL database. It provides superior performance, scalability, and instrumentation.

Percona Server for MySQL is trusted by thousands of enterprises to provide better performance and concurrency for their most demanding workloads. It delivers more value to MySQL server users with optimized performance, greater performance scalability and availability, enhanced backups, and increased visibility.

16.6.1 Release highlights

Percona has removed an Oracle patch for `mysqldump` that performed, at the beginning of the dump, a `FLUSH_TABLES_WITH_READ_LOCK` to get consistent `GTID_EXECUTED` because the patch required additional user privileges, even when the user does not use GTID-based replication. The following bugs based on this patch were submitted to Oracle:

- MySQL 109701
- MySQL 109685

The Percona solution uses `START TRANSACTION WITH CONSISTENT SNAPSHOT`.

Improvements and bug fixes provided by Oracle for MySQL 5.7.41 and included in Percona Server for MySQL are the following:

- Updated the linked OpenSSL library for MySQL Server to 1.1.s
- Updated the bundled zlib library to zlib 1.2.13. This zlib library version is the minimum supported.
- While the SQL thread handled a transaction, issuing `STOP SLAVE_SQL_THREAD` caused replication to stop immediately instead of waiting for the event group to complete before the shutdown.

Find the full list of bug fixes and changes in the MySQL 5.7.41 Release Notes.

16.6.2 Bug fixes

- PS-7538: With `innodb_optimize_fulltext_only` enabled, running `OPTIMIZE TABLE` on a table with an FTS index caused a server exit.

16.6.3 Platform support

This release adds support for Ubuntu 22.04.

This release add support for Red Hat Enterprise Linux 9 and compatible derivatives.

16.6.4 Useful links

The Percona Server for MySQL GitHub location

Contribute to the documentation

For training, contact Percona Training - Start learning now

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-02-16

Download PDF

16.7 Percona Server for MySQL 5.7.40-43 (2022-11-28)

Release date	November 28, 2022
Install instructions	Install Percona Server for MySQL
Download this version	Percona Server for MySQL

Percona Server for MySQL 5.7.40-43 includes all the features and bug fixes available in MySQL 5.7.40 Community Edition in addition to enterprise-grade features developed by Percona.

Percona Server for MySQL is a free, fully compatible, enhanced, and open source drop-in replacement for any *MySQL* database. It provides superior performance, scalability, and instrumentation.

Percona Server for MySQL is trusted by thousands of enterprises to provide better performance and concurrency for their most demanding workloads. It delivers more value to *MySQL* server users with optimized performance, greater performance scalability and availability, enhanced backups, and increased visibility.

For paid support, managed services or consulting services, contact Percona Sales

For training, contact Percona Training - Start learning now

16.7.1 Release highlights

Improvements and bug fixes provided by Oracle for *MySQL* 5.7.40 and included in Percona Server for MySQL are the following:

- ISO 8601 timestamps in log messages did not consider daylight saving time when `--log-timestamps=SYSTEM` was used.
- The `GRANT OPTION` privilege was treated as related to database operations.
- In specific cases, a `TRUNCATE TABLE` operation failed to release an acquired mutex.
- A descending b-tree scan raised a debug assertion failure in debug builds.

Find the full list of bug fixes and changes in the MySQL 5.7.40 Release Notes.

16.7.2 Bug fixes

- PS-1098: Manually rotating the log files by calling `audit_log_flush` multiple times caused incorrect file rotation and the wrong log file to be used.
- PS-8327: `ALTER TABLE ... CHECK PARTITION` inside the stored procedure caused a server exit.

16.7.3 Platform support

Percona Server for MySQL 5.7.40-43 does not support Ubuntu 22.04.

16.7.4 Useful links

The *Percona Server for MySQL* GitHub location

Contribute to the documentation

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-11-28

Download PDF

16.8 Percona Server for MySQL 5.7.39-42 (2022-08-15)

Percona Server for MySQL 5.7.39-42 includes all the features and bug fixes available in MySQL 5.7.39 Community Edition in addition to enterprise-grade features developed by Percona.

Percona Server for MySQL is a free, fully compatible, enhanced, and open source drop-in replacement for any *MySQL* database. It provides superior performance, scalability, and instrumentation.

Percona Server for MySQL is trusted by thousands of enterprises to provide better performance and concurrency for their most demanding workloads. It delivers more value to *MySQL* server users with optimized performance, greater performance scalability and availability, enhanced backups, and increased visibility. Commercial support contracts are available.

16.8.1 Release Highlights

Improvements and bug fixes provided by Oracle for *MySQL* 5.7.39 and included in Percona Server for MySQL are the following:

- To provide process information, the `SHOW PROCESSLIST` statement collects thread data from all active threads. Since the implementation iterates across active threads from within the thread manager while holding a global mutex, it has a negative impact on performance, particularly on busy systems.

Now, an alternative `SHOW PROCESSLIST` implementation is available based on the new Performance Schema processlist table. This implementation queries active thread data from the Performance Schema rather than the thread manager and does not require a mutex:

- To enable the alternative implementation, enable the `performance_schema_show_processlist` system variable.

> **Note**
>
> For new installations of MySQL 5.7.39, or higher, the processlist table is automatically created in the Performance Schema. It is not created automatically by an upgrade. If you are upgrading from an earlier version of MySQL 5.7, and want to use the Performance Schema implementation of processlist, create the table manually.

Find more information in the Creating the processlist table.

- The alternative implementation of `SHOW PROCESSLIST` also applies to the *mysqladmin* processlist command.
- The alternative implementation does not apply to the `INFORMATION_SCHEMA PROCESSLIST` table or the `COM_PROCESS_INFO` command of the MySQL client/server protocol.
- To ensure that the default and alternative implementations give the same information, check the configuration requirements in The processlist Table.
- MySQL removes a 4GB tablespace file size limit on Windows 32-bit systems. The limit was set because of an incorrect calculation performed while extending the tablespace.
- When, during a session, an incorrect value for the `binlog_checksum` system variable is set, a `COM_BINLOG_DUMP` command ran in the same session to request a binary log stream from a source fails. Now, the server validates the specified checksum value before starting the checksum algorithm setup process.

Find the full list of bug fixes and changes in the MySQL 5.7.39 Release Notes.

16.8.2 Deprecation and removal

- The `myisam_repair_threads` system variable has been removed.
- **myisamchk** `--parallel-recover` option has been removed.

16.8.3 Improvements

The `SHOW PROCESSLIST` statement now displays an extra field `TIME_MS`. The `TIME_MS` field provides the information about the time in milliseconds that the thread has been in its current state.

16.8.4 Bugs Fixed

- PS-8205: `DICT_TF2_FLAG_SET` was used instead of `DICT_TF2_FLAG_IS_SET`.
- PS-8174: MySQL crashed at shutdown with `buf0flu.cc:3567:UT_LIST_GET_LEN(buf_pool->flush_list) == 0` assertion.

16.8.5 Useful links

- The Percona Server for MySQL installation instructions
- The Percona Server In-Place Upgrading Guide: From 5.6 to 5.7
- The Percona Software downloads
- The Percona Server for MySQL GitHub location
- To contribute to the documentation, review the Documentation Contribution Guide

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.9 Percona Server for MySQL 5.7.38-41 (2022-06-02)

Percona Server for MySQL 5.7.39-42 includes all the features and bug fixes available in MySQL 5.7.39 Community Edition in addition to enterprise-grade features developed by Percona.

Percona Server for MySQL is a free, fully compatible, enhanced, and open source drop-in replacement for any *MySQL* database. It provides superior performance, scalability, and instrumentation.

Percona Server for MySQL is trusted by thousands of enterprises to provide better performance and concurrency for their most demanding workloads. It delivers more value to *MySQL* server users with optimized performance, greater performance scalability and availability, enhanced backups, and increased visibility. Commercial support contracts are available.

16.9.1 Release Highlights

Improvements and bug fixes provided by Oracle for *MySQL* 5.7.38 and included in Percona Server for MySQL are the following:

- If a statement cannot be parsed, for example, if the statement contains syntax errors, that statement is not written to the slow query log.
- Loading an encrypted table failed if purge threads processed the undo records for that table.
- There was a memory leak when mysqldump was used on more than one table with the –order-by-primary option. The memory allocated to sort each row in a table is now released after every table.

Find the full list of bug fixes and changes in the MySQL 5.7.38 Release Notes.

16.9.2 Deprecation and removal

- The `myisam_repair_threads` system variable is deprecated. Values other than 1 (the default) for `myisam_repair_threads` throw a warning. Support for this variable may be removed in future versions.
- **myisamchk** `--parallel-recover` option is deprecated. Support for this option may be removed in future versions.

16.9.3 Bugs Fixed

- PS-6029: The data masking gen_rnd_us_phone() function had a different format compared to MySQL upstream version.
- PS-8129: A fix for when mutex hangs in thread_pool_unix.
- PS-8136: `LOCK TABLES FOR BACKUP` did not prevent InnoDB key rotation. Due to this behavior, Percona Xtrabackup couldn't fetch the key in case the key was rotated after starting the backup.
- PS-8143: Fixed the memory leak in `File_query_log::set_rotated_name()`.
- PS-8204: When the `audit_log_format` was set to XML, logged queries were truncated after a newline character.

16.9.4 Useful links

- The Percona Server for MySQL installation instructions
- The Percona Server In-Place Upgrading Guide: From 5.6 to 5.7

- The Percona Software downloads
- The Percona Server for MySQL GitHub location
- To contribute to the documentation, review the Documentation Contribution Guide

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.10 Percona Server for MySQL 5.7.37-40 (2022-03-31)

Percona Server for MySQL 5.7.37-40 includes all the features and bug fixes available in MySQL 5.7.37 Community Edition in addition to enterprise-grade features developed by Percona.

Percona Server for MySQL is a free, fully compatible, enhanced, and open source drop-in replacement for any MySQL database. It delivers more value to MySQL server users with optimized performance, greater performance scalability and availability, enhanced backups, and increased visibility. Commercial support contracts are available.

16.10.1 Release Highlights

The following lists a number of the notable updates and fixes for MySQL 5.7.37, provided by Oracle, and included in Percona Server for MySQL:

- The performance on debug builds has been improved by optimizing buf_validate() function in the *InnoDB* sources.
- Fix for when a query using an index that differs from the primary key of partitioned table results in excessive CPU load.

Find the complete list of bug fixes and changes in MySQL 5.7.37 Release Notes.

16.10.2 Improvements

- PS-7792: Allows setting an empty MASTER_USER user name if you always provide user credentials when using the START_SLAVE statement. This method requires user intervention to restart the replica.

16.10.3 Bugs Fixed

- PS-7929: Fix for when the row locks were duplicated when inserting an existing row into a table within the same transaction.
- PS-8007: *Percona Server for MySQL* can fail to start if the server starts before the network mounts the datadir or a local mount of the datadir.
- PS-7856: A partition table update caused a server exit.
- PS-7890: When the server was started with the –loose-rocksdb_persistent_cache_size_mb option, the RocksDB engine plugin installation failed.

16.10.4 Packaging Notes

- Red Hat Enterprise Linux 6 (and derivative Linux distributions) are no longer supported.
- Debian 9 is no longer supported.

16.10.5 Known issues

- The RPM packages for Red Hat Enterprise Linux 7 (and compatible derivatives) do not support TLSv1.3, as it requires OpenSSL 1.1.1, which is currently not available on this platform.

16.10.6 Useful links

- To install Percona Server for MySQL 5.7, follow the instructions in Installing Percona Server for MySQL .
- To upgrade Percona Server for MySQL from 5.6 to 5.7, follow the instructions in Percona Server In-Place Upgrading Guide: From 5.6 to 5.7.
- The GitHub location for Percona Server.
- To contribute to the Percona Server for MySQL documentation, review the Documentation Contribution Guide.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.11 Percona Server for MySQL 5.7.36-39 (2021-12-22)

- **Installation** Installing Percona Server for MySQL

Percona Server for MySQL 5.7.36-39 includes all the features and bug fixes available in the MySQL 5.7.36 Community Edition., in addition to enterprise-grade features developed by Percona.

Percona Server for MySQL® is a free, fully compatible, enhanced, and open source drop-in replacement for any MySQL database. It provides superior performance, scalability, and instrumentation.

Percona Server for MySQL is trusted by thousands of enterprises to provide better performance and concurrency for their most demanding workloads. It delivers more value to MySQL server users with optimized performance, greater performance scalability and availability, enhanced backups, and increased visibility. Commercial support contracts are available.

16.11.1 Release Highlights

The following lists some of the bug fixes for MySQL 5.7.36, provided by Oracle, and included in Percona Server for MySQL:

- Fix for the possibility for a deadlock or failure when an undo log truncate operation is initiated after an upgrade from *MySQL* 5.6 to *MySQL* 5.7.
- Fix for when a parent table initiates a cascading `SET NULL` operation on the child table, the virtual column can be set to NULL instead of the value derived from the parent table.
- On a view, the query digest for each SELECT statement is now based on the SELECT statement and not the view definition, which was the case for earlier versions.

Find the complete list of bug fixes and changes in MySQL 5.7.36 Release Notes.

16.11.2 Improvements

- PS-6730 The Last_errno field in the Slow Query Log only reports the errors.

16.11.3 Bugs Fixed

- PS-7868: Documentation - remove a reference to a 5.7 SELinux repository in the Yum installation document. (Thanks to user Simon Avery for reporting this issue)
- PS-7958: Fix for a MySQL exit when using a full-text search index with a special character.
- PS-1484: Fix for slow log rotation when the file name has an extension.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.12 Percona Server for MySQL 5.7.35-38 (2021-08-18)

- **Installation** Installing Percona Server for MySQL

Percona Server for MySQL 5.7.35-38 includes all the features and bug fixes available in MySQL 5.7.35 Community Edition in addition to enterprise-grade features developed by Percona.

16.12.1 Bugs Fixed

- PS-1346: LP #1163232: Anomaly with `opt_log_slow_slave_statements`.
- PS-1344: LP #1160436: The `log_slow_statement` is called unconditionally
- PS-7582: Segmentation fault with the data masking plugin
- PS-1108: LP #1704163: Changing a column from uncompressed to compressed for JSON crashes the server
- PS-2433: LP #1234346: Include a timestamp in the slow query log file when initializing a new file
- PS-1955: LP #1088529: The `log_slow_verbosity` help text missing the "minimal", "standard", and "full" options
- PS-1116: LP #1719506: Audit plugin reports "command_class=error" for server-side prepared statements.
- PS-7755: InnoDB: tried to purge `non-delete-marked` record (Upstream #86485).
- PS-6802: Configure fails with make-4.3 with CMake Error at storage/rocksdb/CMakeLists.txt:152 (STRING) (Thanks to user whissi for reporting this issue)
- PS-1659: LP #1508909: Connect without proxy information hangs if `proxy_protocol_networks` is enabled
- PS-7746: Possible double call to free_share() in ha_innobase::open()

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.13 Percona Server for MySQL 5.7.34-37 (2021-05-26)

- **Installation** Installing Percona Server for MySQL

Percona Server for MySQL 5.7.34-37 includes all the features and bug fixes available in MySQL 5.7.34 Community Edition in addition to enterprise-grade features developed by Percona.

16.13.1 Bugs Fixed

- PS-4497: Incorrect option error message for mysqlbinlog
- PS-7498: Prevent the replication coordinator thread stuck in Waiting until MASTER_DELAY seconds after master executed event while handling partial relay log transactions. (Upstream #102647)
- PS-7578: Replication failure with UPDATE when replica server has a PK and the source does not. (Upstream #102628)
- PS-7593: When changing the tx-isolation on a session, after a transaction has executed, the change is not honored and may cause a service failure. (Upstream #102831)
- PS-7657: An update query executed against partition tables with compressed columns can cause an unexpected server exit.

Last update: 2022-09-27

Download PDF

16.14 Percona Server for MySQL 5.7.33-36 (2021-03-02)

- **Installation** Installing Percona Server for MySQL

Percona Server for MySQL 5.7.33-36 includes all the features and bug fixes available in MySQL 5.7.33 Community Edition in addition to enterprise-grade features developed by Percona.

16.14.1 New Features

- PS-5364: Update keyring_vault plugin to support KV Secrets Engine Version 2 (kv-v2) (Thanks to user aprokofyev for reporting this issue)
- PS-7447: Backport variable innodb_buffer_pool_in_core_file and processing (Upstream #101825)
- PS-7459: Backport of InnoDB: Group purging of rows by table ID (WL#9387) to PS 5.7.33

16.14.2 Bugs Fixed

- PS-1956: Change data type for some microsecond times for the slow query log to 64-bit
- PS-7499: Improve error log when MyRocks fails with rocksdb_validate_tables=1
- PS-5112: Backport fix for PS-5027 from 8.0 to 5.7
- PS-7492: Update slow log formatting for tmp tables related stats
- PS-7498: Prevent the replication co-ordinator thread getting stuck due to MASTER_DELAY while handling partial relay log transactions. (Upstream #102647)

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.15 Percona Server for MySQL 5.7.32-35 (2020-11-24)

- **Installation** Installing Percona Server for MySQL

Percona Server for MySQL 5.7.32-35 includes all the features and bug fixes available in MySQL 5.7.32 Community Edition in addition to enterprise-grade features developed by Percona.

16.15.1 New Features

- PS-7238: Backport Data Masking plugin to 5.7 (Thanks to user Surenda Kumar Gupta for reporting this issue)

16.15.2 Bugs Fixed

- PS-7346: Correct the buffer calculation for the audit plugin used when large queries are executed(PS-5395).
- PS-7232: Modify Multithreaded Replica to correct the exhausted slave_transaction_retries when replica has slave_preserve_commit_order enabled (Upstream #99440)
- PS-7231: Modify Slave_transaction::retry_transaction() to call mysql_errno() only when thd->is_error() is true
- PS-7304: Correct package to include coredumper.a as a dependency of libperconaserverclient20-dev (Thanks to user Martin for reporting this issue)
- PS-7289: Restrict innodb encryption threads to 255 and add min/max values
- PS-7270: Fix admin_port to accept non-proxied connections when proxy_protocol_networks='*'

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.16 Percona Server for MySQL 5.7.31-34 (2020-08-24)

- **Installation** Installing Percona Server for MySQL

Percona Server for MySQL 5.7.31-34 includes all the features and bug fixes available in MySQL 5.7.31 Community Edition in addition to enterprise-grade features developed by Percona.

16.16.1 New Features

- PS-7128: Document RocksDB variables: `rocksdb_max_background_compactions`, `rocksdb_max_background_flushes`, and `rocksdb_max_bottom_pri_background_compactions`

16.16.2 Improvements

- PS-7132: Make default value of `rocksdb_wal_recovery_mode` compatible with InnoDB
- PS-7199: Add Coredumper functionality
- PS-7114: Enhance crash artifacts (core dumps and stack traces) to provide additional information to the operator

16.16.3 Bugs Fixed

- PS-7203: Fixed audit plugin memory leak on replicas when opening tables
- PS-7043: Correct constant equality expression is used in LEFT JOIN condition by setting the 'const_table' flag together with setting the row as a NULL-row. (Upstream #99499)
- PS-7212: Modify processing to binary compare in order to do native JSON comparison (Upstream #100307)
- PS-7076: Modify to not update Cardinality after setting `tokudb_cardinality_scale_percent`
- PS-7025: Fix reading ahead of insert buffer pages by dispatching of buffered AIO transfers (Upstream #100086)
- PS-7010: Modify to Lock buffer blocks before sanity check in btr_cur_latch_leaves
- PS-6995: Introduce a new optimizer switch to allow the user to reduce the cost of a range scan to determine best execution plan for Primary Key lookup
- PS-5978: Remove unneeded check of variable to allow mysqld_safe support –numa-interleave (Thanks to user springlin for reporting this issue)
- PS-7220: Fix activity counter update in purge coordinator and workers
- PS-7234: Modify PS minimal tarballs to remove COPYING.AGPLv3
- PS-7204: Add checks to linkingscript to correct failures in patchelf
- PS-7075: Provide binary tarball with shared libs and glibc suffix
- PS-7062: Modify ALTER INSTANCE ROTATE INNODB MASTER KEY to skip writing of redo for compressed encrypted temporary table.
- PS-5263: Update handle_binlog_flush_or_sync_error() to set my_ok(thd) after thd->clear_error() to correct assert in THD::send_statement_status (Upstream #93770)
- PS-4530: Add documentation that `ps-admin` removes jemalloc and THP settings on TokuDB uninstall

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.17 Percona Server for MySQL 5.7.30-33 (2020-05-20)

- **Installation** Installing Percona Server for MySQL

Percona Server for MySQL 5.7.30-33 includes all the features and bug fixes available in MySQL 5.7.30 Community Edition in addition to enterprise-grade features developed by Percona.

Merged MyRocks/RocksDB up to Facebook MySQL production tag fb-prod201907.

16.17.1 New Features

- PS-6951: Document new RocksDB variables: rocksdb_delete_cf, rocksdb_enable_iterate_bounds, and rocksdb_enable_remove_orphaned_dropped_cfs
- PS-4464: Expose the last global transaction identifier (GTID) executed for a CONSISTENT SNAPSHOT.
- PS-6926: Document RocksDB variables: rocksdb_table_stats_recalc_threshold_pct, rocksdb_table_stats_recalc_threshold_count, rocksdb_table_stats_background_thread_nice_value, rocksdb_table_stats_max_num_rows_scanned, rocksdb_table_stats_use_table_scan. rocksdb_table_stats_background_thread_nice_value, rocksdb_table_stats_max_num_rows_scanned, rocksdb_table_stats_use_table_scan, and rocksdb_trace_block_cache_access.
- PS-6901: Document RocksDB variable: rocksdb_read_free_rpl.
- PS-6890: Document RocksDB variable: rocksdb_blind_delete_primary_key.
- PS-6885: Document the new variable rocksdb_rollback_on_timeout which allows the rollback of an entire transaction on timeout.
- PS-6891: Document RocksDB variable: rocksdb_master_skip_tx_api.
- PS-6886: Document variable rocksdb_cache_dump which includes RocksDB block cache content in a core dump.
- PS-6910: Document RocksDB variable: rocksdb_stats_level.

16.17.2 Improvements

- PS-6984: Update the zstd submodule to v1.4.4.

16.17.3 Bugs Fixed

- PS-6979: Modify the processing to call clean up functions to remove CREATE USER statement from the processlist after the statement has completed (Upstream #99200)
- PS-6860: Merge innodb_buffer_pool_pages_LRU_flushed into buf_get_total_stat()
- PS-6811: Correct service failure of asserting ACL_PROXY_USER when skip-name-resolve=1 and there is a Proxy user (Upstream #98908)
- PS-6112: Correct Binlog_snapshot_gtid inconsistency when mysqldump was used with –single-transaction.
- PS-6945: Correct tokubackup plugin process exported API to allow large file backups.
- PS-6856: Correct binlogs corruptions in PS 5.7.28 and 5.7.29 (Upstream #97531)
- PS-6946: Correct tokubackup processing to free memory use from the address and thread sanitizers
- PS-5893: Add support for running multiple instances with systemD on Debian.

- PS-5620: Modify Docker image to support supplying custom TLS certificates
- PS-4573: Implement use of a single config file - mysqld.cnf file.
- PS-7041: Correct Compilation error when -DWITH_EDITLINE=bundled is used
- PS-7020: Modify MTR tests for Ubuntu 20.04 to include python2 (python 2.6 or higher) and python3
- PS-6974: Correct instability in the rocksdb.drop_cf_* tests
- PS-6969: Correct instability in the rocksdb.index_stats_large_table
- PS-6954: Correct tokudb-backup-plugin to avoid collision between -std=c++11 and -std=gnu++03.
- PS-6925: Correct mismatched default socket values for mysqld and mysqld_safe
- PS-6899: Correct main.events_bugs and main.events_1 to interpret date 01-01-2020 properly (Upstream #98860)
- PS-6796: Correct instability in percona_changed_page_bmp_shutdown_thread
- PS-6773: Initialize values in sha256_password_authenticate (Upstream #98223)
- PS-5844: Fix a memory leak after 'innodb.alter_crash' in 'prepare_inplace_alter_table_dict()' (Upstream #96472)
- PS-5735: Correct 5.7 package to install the charsets on CentOS 7
- PS-4757: Remove CHECK_IF_CURL_DEPENDS_ON_RTMP to build keyring_vault for unconditional test
- PS-4649: Document PerconaFT in TokuDB which is fractal tree indexing to enhance the B-tree data structure

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-04-21

Download PDF

16.18 Percona Server for MySQL 5.7.29-32 (2020-02-05)

- **Installation**: Installing Percona Server for MySQL

Percona Server for MySQL 5.7.29-32 includes all the features and bug fixes available in MySQL 5.7.29 Community Edition in addition to enterprise-grade features developed by Percona.

16.18.1 Bugs Fixed

- PS-1469: The Memory storage engine detected an incorrect "is full" condition when the space contained reusable memory chunks that could be reused.
- PS-6113: If `ANALYZE TABLE` with persistent statistics ran more than 600 seconds the execution of a diagnostic query may cause a server exit. (Upstream #97828)
- PS-5813: To set the `slow_query_log_use_global_control` to "none" could cause an error.
- PS-6150: The execution of SHOW ENGINE INNODB STATUS to show locked mutexes could cause a server exit.
- PS-6750: The installation of client packages could cause a file conflict in Red Hat Enterprise Linux 8.
- PS-5940: When a temporary table was dropped, the server exited. (Upstream #96766)
- PS-5675: Concurrent INSERT … ON DUPLICATE KEY UPDATE statements could cause a failure with a unique index violation. (Upstream #96578)
- PS-5421: MyRocks: Corrected documentation for `rocksdb_db_write_buffer_size`.
- PS-4794: Documented that using ps-admin to enable MyRocks does not disable Transparent Huge Pages.
- PS-6093: The execution of SHOW ENGINE INNODB STATUS to show locked mutexes with simultaneous access to a compressed table could cause a server exit.
- PS-6148: If ANALYZE TABLE with transient statistics ran more than 600 seconds the execution of a diagnostic query may cause a server exit. (Upstream #97828)
- PS-6125: MyRocks: To set `rocksdb_update_cf_options` with a non-existant column family created a partially-defined column family which could cause a server exit.
- PS-6123: A Debian/Ubuntu init script used an incorrect comparison which could cause the service command to return before the server start.
- PS-5956: Root session could kill Utility user session.
- PS-5952: Utility user was visible in performance_schema.threads.
- PS-5843: A memory leak could occur after "group_replication.gr_majority_loss_restart". (Upstream #96471)
- PS-5325: Conditional jump or move depended on uninitialized value on innodb_zip.wl5522_zip or innodb.alter_missing_tablespace.

Download PDF

16.19 Percona Server for MySQL 5.7.28-31 (2019-11-13)

Percona is glad to announce the release of *Percona Server for MySQL* 5.7.28-31 on November 13, 2019. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.28 and includes all the bug fixes in it. *Percona Server for MySQL* 5.7.28-31 is now the current GA (Generally Available) release in the 5.7 series.

All software developed by Percona is open-source and free.

Note

If you're currently using *Percona Server for MySQL* 5.7, Percona recommends upgrading to this version of 5.7 prior to upgrading to *Percona Server for MySQL* 8.0.

16.19.1 Bugs Fixed

- When using skip-innodb_doublewrite in my.cnf, a parallel doublewrite buffer is still created. Bugs fixed #3411.
- During a binlogging replication event, if the master crashes after the multi-threaded slave has begun copying to the slave's relay log and before the process has completed, a STOP SLAVE on the slave takes longer than expected. Bug fixed #5824.
- If pam_krb5 is configured to allow the user to change their password, and the password expired, the server crashed after receiving the new password. Bug fixed #6023.

Other bugs fixed: #5859, #5910, #5966, #4784, #5216, #5327, #5584, #5642, #5659, #5754, #5761, #5797, #5875, #5933, #5941, #5997, #6050, #6052, #3345, and #5585

16.19.2 Known Issues

- #5783: The length of time and resources required for a MySQL query execution increased with a large number of table partitions. Limiting the Estimation of Records in a Query describes the experimental options added to prevent index scans on the partitions and return a specified number of values.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.20 Percona Server for MySQL 5.7.27-30 (2019-08-22)

Percona is glad to announce the release of *Percona Server for MySQL* 5.7.27-30 on August 22, 2019. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.27 and includes all the bug fixes in it. *Percona Server for MySQL* 5.7.27-30 is now the current GA (Generally Available) release in the 5.7 series.

All software developed by Percona is open-source and free.

Note

If you're currently using *Percona Server for MySQL* 5.7, Percona recommends upgrading to this version of 5.7 prior to upgrading to *Percona Server for MySQL* 8.0.

16.20.1 Bugs Fixed

- Parallel doublewrite buffer writes must crash the server on an I/O error occurs. Bug fixed #5678.
- On a server with two million or more tables using foreign keys and AUTOINC columns, the shutdown may take a measurable length of time. Bug fixed #5639. (Upstream #95895)
- If large pages are enabled on the MySQL side, the maximum size for `innodb_buffer_pool_chunk_size` is effectively limited to 4GB. Bug fixed #5517. (Upstream #94747)
- 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.
- A multi-table `DELETE` with a foreign key breaks replication. Bug fixed #3845.
- A `TRUNCATE` with any table and interfacing with Adaptive Hash Index (AHI) can cause server stalls due to the interaction with AHI, whether the AHI is enabled or not. Bug fixed #5576. (Upstream #94610)
- In specific configurations and with `log_slow_verbosity` set to log InnoDB statistics, memory usage increases while running a stored procedure. Bug fixed #5581.
- Thread Pool functionality to track network I/O was disabled. Bug fixed #5723.
- When Adaptive Hash Index (AHI) is enabled or disabled, there is an AHI overhead during DDL operations. Bug fixed #5747.
- An instance started with the default values but setting the redo-log to encrypt without specifying the keyring plugin parameters does not fail or throw an error. Bug fixed #5476.
- Setting the encryption to `ON` for the system tablespace generates the encryption key and encrypts system temporary tablespace pages. Resetting encryption to `OFF` , all subsequent pages are written to the temporary tablespace without encryption. To allow any encrypted tables to be decrypted, the generated keys are not erased. Modifying the `innodb_temp_tablespace_encrypt` does not affect file-per-table temporary tables. This type of table is encrypted if `ENCRYPTION` ='Y' is set during the table creation. Bug fixed #5736.
- After resetting the `innodb_temp_tablespace_encrypt` to `OFF` during runtime, the subsequent file-per-table temporary tables continue to be encrypted. Bug fixed #5734.

Other bugs fixed: #5752, #5749, #5746, #5744, #5743, #5742, #5740, #5695, #5681, #5669, #5645, #5638, #5593, #5532, #5790, #5812, #3970, #5696, #5689, #5146, #5715, #5791, #5662, #5420, #5149, #5686, #5688, #5697, #5716, #5725, #5773, #5775, #5820, and #5839.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.21 Percona Server for MySQL 5.7.26-29 (2019-05-27)

Percona is glad to announce the release of *Percona Server for MySQL* 5.7.26-29 on May 27, 2019. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.26 and includes all the bug fixes in it. *Percona Server for MySQL* 5.7.26-29 is now the current GA (Generally Available) release in the 5.7 series.

16.21.1 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 `audit_log_buffer_size` variable.

16.21.2 Bugs Fixed

- TokuDB storage engine would assert on load when used with jemalloc 5.x. Bug fixed #5406.
- A read-write workload on compressed InnoDB tables could cause an assertion error. Bug fixed #3581.
- Using TokuDB or MyRocks native partitioning and `index_merge` access method could lead to a server crash. Bugs fixed #5206, #5562.
- A stack buffer overrun could happen if the redo log encryption with key rotation was enabled. Bug fixed #5305.
- TokuDB and MyRocks native partitioning handler objects were allocated from a wrong memory allocator. Memory was released only on shutdown and concurrent access to global memory allocator caused memory corruptions and therefore crashes. Bugs fixed #5508, #5525.
- Enabling redo log encryption resulted in redo log being written unencrypted. Bug fixed #5547.
- If there are multiple row versions in InnoDB, reading one row from PK may have O(N) complexity and reading from secondary keys may have O(N^2) complexity. Bugs fixed #4712, #5450 (upstream #84958).
- Setting the `log_slow_verbosity` to include `innodb` value and enabling the `slow_query_log` could lead to a server crash. Bug fixed #4933.
- 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.
- XtraDB changed page tracking was missing pages changed by the in-place DDL. Bug fixed #5447.
- The `innodb_encrypt_tables` variable accepted `FORCE` option only inside quotes as a string. Bug fixed #5538.
- Enabling redo log encryption and XtraDB changed page tracking together would result in the error log flooded with decryption errors. Bug fixed #5541.
- System keyring keys initialization wasn't thread safe. Bugs fixed #5554.
- when using the Docker image, if the root passwords set in the mounted `.cnf` file and the one specified with `MYSQL_ROOT_PASSWORD` are different, password from the `MYSQL_ROOT_PASSWORD` will be used. Bug fixed #5573.

- Long running `ALTER TABLE ADD INDEX` could cause a `semaphore wait > 600` assertion. Bug fixed #3410 (upstream #82940).

Other bugs fixed: #5007 (upstream #93164), #5018, #5561, #5570, #5578, #5610, #5441, and #5442.

This release also contains the fixes for the following security issues: CVE-2019-2632, CVE-2019-1559, CVE-2019-2628, CVE-2019-2581, CVE-2019-2683, CVE-2019-2592, CVE-2019-262, and CVE-2019-2614.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.22 Percona Server for MySQL 5.7.25-28 (2019-02-18)

Percona is glad to announce the release of *Percona Server for MySQL* 5.7.25-28 on February 18, 2019. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.25 and includes all the bug fixes in it. *Percona Server for MySQL* 5.7.25-28 is now the current GA (Generally Available) release in the 5.7 series.

All software developed by Percona is open-source and free.

In this release, *Percona Server for MySQL* introduces the variable `binlog_skip_flush_commands`. This variable controls whether or not `FLUSH` commands are written to the binary log. Setting this variable to **ON** can help avoid problems in replication. For more information, see Writing FLUSH Commands to the Binary Log.

Note

If you're currently using *Percona Server for MySQL* 5.7, Percona recommends upgrading to this version of 5.7 prior to upgrading to *Percona Server for MySQL* 8.0.

16.22.1 Bugs Fixed

- FLUSH commands written to the binary log could cause errors in case of replication. Bug fixed #1827: (upstream `88720`).
- Running LOCK TABLES FOR BACKUP followed by STOP SLAVE SQL_THREAD could block replication preventing it from being restarted normally. Bug fixed #4758.
- The `ACCESS_DENIED` field of the information_schema.user_statistics table was not updated correctly. Bug fixed #3956.
- MySQL could report that the maximum number of connections was exceeded with too many connections being in the CLOSE_WAIT state. Bug fixed #4716 (upstream #92108)
- Wrong query results could be received in semi-join sub queries with materialization-scan that allowed inner tables of different semi-join nests to interleave. Bug fixed #4907 (upstream bug #92809).
- In some cases, the server using the MyRocks storage engine could crash when TTL (Time to Live) was defined on a table. Bug fixed #4911.
- Running the SELECT statement with the ORDER BY and LIMIT clauses could result in a less than optimal performance. Bug fixed #4949 (upstream #92850)
- There was a typo in `mysqld_safe.sh`: **trottling** was replaced with **throttling**. Bug fixed #240. Thanks to Michael Coburn for the patch.
- MyRocks could crash while running `START TRANSACTION WITH CONSISTENT SNAPSHOT` if other transactions were in specific states. Bug fixed #4705.
- In some cases, `mysqld` could crash when inserting data into a database the name of which contained special characters (CVE-2018-20324). Bug fixed #5158.
- MyRocks incorrectly processed transactions in which multiple statements had to be rolled back. Bug fixed #5219.
- In some cases, the MyRocks storage engine could crash without triggering the crash recovery. Bug fixed #5366.
- When bootstrapped with undo or redo log encryption enabled on a very fast storage, the server could fail to start. Bug fixed #4958.

- Some fields in the output of `SHOW USER_STATISTICS` command did not contain correct information. Bug fixed #4996.

Other bugs fixed: #2455, #4791, #4855, #5268.

This release also contains fixes for the following CVE issues: CVE-2019-2534, CVE-2019-2529, CVE-2019-2482, CVE-2019-2434.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.23 Percona Server for MySQL 5.7.24-27 (2018-12-18)

Percona is glad to announce the release of *Percona Server for MySQL* 5.7.24-27 on December 18, 2018. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.24 and includes all the bug fixes in it. *Percona Server for MySQL* 5.7.24-27 is now the current GA (Generally Available) release in the 5.7 series.

All software developed by Percona is open-source and free.

Note

If you're currently using *Percona Server for MySQL* 5.7, Percona recommends upgrading to this version of 5.7 prior to upgrading to *Percona Server for MySQL* 8.0.

16.23.1 Bugs Fixed

- When uninstalling *Percona Server for MySQL* packages on *CentOS 7* default configuration file `my.cnf` would get removed as well. This fix makes the backup of the configuration file instead of removing it. Bug fixed #5092.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.24 Percona Server 5.7.24-26 (2018-12-04)

Percona announces the release of Percona Server for MySQL 5.7.24-26 on December 4, 2018 (downloads are available here and from the Percona Software Repositories). This release merges changes of MySQL 5.7.24, including all the bug fixes in it. Percona Server for MySQL 5.7.24-26 is now the current GA release in the 5.7 series. All of Percona's software is open-source and free.

This release includes fixes to the following upstream CVEs (Common Vulnerabilities and Exposures): *CVE-2016-9843*, *CVE-2018-3155*, *CVE-2018-3143*, *CVE-2018-3156*, *CVE-2018-3251*, *CVE-2018-3133*, *CVE-2018-3144*, *CVE-2018-3185*, *CVE-2018-3247*, *CVE-2018-3187*, *CVE-2018-3174*, *CVE-2018-3171*. For more information, see Oracle Critical Patch Update Advisory - October 2018 https://www.oracle.com/technetwork/security-advisory/cpuoct2018-4428296.html.

16.24.1 Improvements

- #4790: Improve user statistics accuracy

16.24.2 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).
- Setting the `tokudb_last_lock_timeout` variable via the command line could cause the 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.
- When using MyRocks storage engine, the server could crash after running `ALTER TABLE DROP INDEX` on a slave. Bug fixed #4744.
- The audit log could be corrupted when the `audit_log_rotations` variable was changed at runtime. Bug fixed #4950.

Other Bugs Fixed

- #4781: sql_yacc.yy uses SQLCOM_SELECT instead of SQLCOM_SHOW_XXXX_STATS
- #4881: Add LLVM/clang 7 to Travis-CI
- #4825: Backport MTR fixes from 8.0
- #4998: Valgrind: compilation fails with: writing to 'struct buf_buddy_free_t' with no trivial copy-assignment
- #4980: Valgrind: Syscall param write(buf) points to uninitialised byte(s): Event_encrypter::encrypt_and_write()
- #4982: Valgrind: Syscall param io_submit(PWRITE) points to uninitialised byte(s): buf_dblwr_write_block_to_datafile()
- #4983: Valgrind: Syscall param io_submit(PWRITE) points to uninitialised byte(s): buf_flush_write_block_low()
- #4951: Many libc-related Valgrind errors on CentOS7
- #5012: Valgrind: misused UNIV_MEM_ALLOC after ut_zalloc_nokey
- #4908: UBSan and valgrind errors with encrypted temporary files
- #4532: Replace obsolete HAVE_purify with HAVE_VALGRIND in ha_rocksdb.cc

- #4955: Backport mysqld fixes for valgrind warnings from 8.0
- #4529: MTR: index_merge_rocksdb2 inadvertently tests InnoDB instead of MyRocks
- #5056: handle_fatal_signal (sig=11) in ha_tokudb::write_row
- #5084: innodb_buffer_pool_size is an uninitialized variable
- #4836: Missing PFS signed variable aggregation
- #5033: rocksdb.show_engine: Result content mismatch
- #5034: rocksdb.rocksdb: Result content mismatch
- #5035: rocksdb.show_table_status: 1051: Unknown table 'db_new'

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.25 Percona Server 5.7.23-25 (2018-11-21)

Percona announces the release of Percona Server for MySQL 5.7.23-25 on November 21, 2018 (downloads are available here and from the Percona Software Repositories). This release merges changes of MySQL 5.7.23, including all the bug fixes in it. Percona Server for MySQL 5.7.23-25 is now the current GA release in the 5.7 series. All of Percona's software is open-source and free.

This release fixes a critical bug in a RocksDB submodule.

16.25.1 Bugs Fixed

- #5049: A severe memory leak regression in the RocksDB Block Cache

Find the release notes for Percona Server for MySQL 5.7.23-24 in our online documentation. Report bugs in the Jira bug tracker.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.26 Percona Server 5.7.23-24 (2018-11-09)

Percona announces the release of Percona Server for MySQL 5.7.23-24 on November 9, 2018 (downloads are available here and from the Percona Software Repositories). This release merges changes of MySQL 5.7.23, including all the bug fixes in it. Percona Server for MySQL 5.7.23-24 is now the current GA release in the 5.7 series. All of Percona's software is open-source and free.

This release introduces InnoDB encryption improvements and merges upstream MyRocks changes. Also, the usage of column families in MyRocks has been improved. The InnoDB encryption improvements are in **Alpha** quality and are not recommended to be used in production.

16.26.1 New Features

- #4905: Upstream MyRocks changes have been merged up to prod201810 tag
- #4976: InnoDB Undo Log Encryption has been implemented
- #4946: Add the `rocksdb_no_create_column_family` option to prevent the implicit creation of column families in MyRocks
- #4556: InnoDB Redo log has been implemented
- #3839: InnoDB Data Scrubbing has been implemented
- #3834: InnoDB Log Scrubbing has been implemented

16.26.2 Bugs Fixed

- #4723: `PURGE CHANGED_PAGE_BITMAPS` did not work when `innodb_data_home_dir` was used
- #4937: `rocksdb_update_cf_options` was ignored when specified in `my.cnf` or on command line
- #1107: The binlog could be corrupted when tmpdir got full
- #4834: The encrypted system tablespace could have an empty uuid
- #3906: The server instance could crash when running the `ALTER` statement

Other bugs fixed

- #4106: "Assertion `log.getting_synced` failed in `rocksdb::DBImpl::MarkLogsSynced(uint64_t, bool, const rocksdb::Status&)`"
- #4930: "main.percona_log_slow_innodb: Result content mismatch"
- #4811: "5.7 Merge and fixup for old DB-937 introduces possible regression"
- #4705: "crash on snapshot size check in RocksDB"

Find the release notes for Percona Server for MySQL 5.7.23-24 in our online documentation. Report bugs in the Jira bug tracker.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.27 Percona Server 5.7.23-23 (2018-09-12)

Percona is glad to announce the release of Percona Server 5.7.23-23 on September 12, 2018. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.23 and includes all the bug fixes in it. *Percona Server for MySQL* 5.7.23-23 is now the current GA (Generally Available) release in the 5.7 series.

All software developed by Percona is open-source and free.

16.27.1 New Features

- The `max_binlog_files` variable is deprecated and replaced with the `binlog_space_limit` variable. The behavior of `binlog_space_limit` is consistent with the variable `relay-log-space-limit` used for relay logs; both variables have the same semantics. For more information, see #275.
- Starting with 5.7.23-23, it is possible to encrypt all data in the InnoDB system tablespace and in the parallel double write buffer. A new variable `innodb_sys_tablespace_encrypt` is introduced to encrypt the system tablespace. This feature is considered **ALPHA** quality. The encryption of the parallel double write buffer file is controlled by the variable `innodb_parallel_dblwr_encrypt`. Both variables are `OFF` by default. For more information, see #3822.
- Changing the `rocksdb_update_cf_options` value returns any warnings and errors to the client instead of printing them to the server error log. For more information, see #4258.
- `rocksdb_number_stat_computers` and `rocksdb_rate_limit_delay_millis` variables have been removed. For more information, see #4780.
- A number of new variables were introduced for MyRocks: `rocksdb_rows_filtered` to show the number of rows filtered out for TTL in MyRocks tables, `rocksdb_bulk_load_allow_sk` to allow adding secondary keys using the bulk loading feature, `rocksdb_error_on_suboptimal_collation` toggling warning or error in case of an index creation on a char field where the table has a sub-optimal collation, `rocksdb_stats_recalc_rate` specifying the number of indexes to recalculate per second, `rocksdb_commit_time_batch_for_recovery` toggler of writing the commit time write batch into the database, and `rocksdb_write_policy` specifying when two-phase commit data are actually written into the database.

16.27.2 Bugs Fixed

- The statement `SELECT...ORDER BY` produced inconsistent results with the `euckr` charset or `euckr_bin` collation. Bug fixed #4513 (upstream #91091).
- InnoDB statistics could incorrectly report zeros in the slow query log. Bug fixed #3828.
- With the FIPS mode enabled and performance_schema=off, the instance crashed when running the `CREATE VIEW` command. Bug fixed #3840.
- The soft limit of the core file size was set incorrectly starting with Percona Server for MySQL `5.7.21-20`. Bug fixed #4479.
- The option `innodb-optimize-keys` could fail when a dumped table has two columns such that the name of one of them contains the other as as a prefix and is defined with the AUTO_INCREMENT attribute. Bug fixed #4524.
- When `innodb_temp_tablespace_encrypt` was set to `ON` the `CREATE TABLE` command could ignore the value of the `ENCRYPTION` option. Bug fixed #4565.
- If `FLUSH STATUS` was run from a different session, a statement could be counted twice in `GLOBAL STATUS`. Bug fixed #4570 (upstream #91541).

- In some cases, it was not possible to set the `flush_caches` variable on systems that use systemd. Bug fixed #3796.
- A message in the MyRocks log file did not clearly inform whether fast CRC32 was supported. Bug fixed #3988.
- `mysqld` could not be started on Ubuntu if the database recovery had taken longer than ten minutes. Bug fixed #4546 (upstream #91423).
- The ALTER TABLE command was slow when the number of dirty pages was high. Bug fixed #3702.
- Setting the global variable `version_suffix` to NULL could lead to a server crash. Bug fixed #4785.
- When more space was added to the data partition after the error that the disk partition was full, MyRocks could ignore data update commands. Bug fixed #4706.

16.27.3 Other Bugs Fixed

- #4620 "Enable encryption of temporary tablespace from foreground thread"
- #4727 "intrinsic temp table behaviour shouldn't depend on innodb_encrypt_tables"
- #4046 "Ship assert failure: 'res == 0' (bulk loader)"
- #3851 "Percona Ver 5.6.39-83.1 Failing assertion: sym_node->table != NULL"
- #4533 "audit_log MTR tests should refer to include files without parent directories"
- #4619 "main.flush_read_lock fails with timeout in wait_condition.inc."
- #4561 "Read after free at Binlog_crypt_data::load_latest_binlog_key()"
- #4587 "ROCKSDB_INCLUDE_RFR macro in wrong file"

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.28 Percona Server 5.7.22-22 (2018-05-31)

Percona is glad to announce the release of Percona Server 5.7.22-22 on May 31, 2018. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.22 and includes all the bug fixes in it. *Percona Server for MySQL* 5.7.22-22 is now the current GA (Generally Available) release in the 5.7 series.

All software developed by Percona is open-source and free.

16.28.1 New Features

- A new `--encrypt-tmp-files` option turns on encryption for the temporary files which *Percona Server for MySQL* may create on disk for filesort, binary log transactional caches and Group Replication caches.

16.28.2 Bugs Fixed

- 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).
- NUMA support was improved in *Percona Server for MySQL*, reverting upstream implementation back to the original one, due to the upstream variant being less effective in memory allocation. Now innodb_numa_interleave variable not only enables NUMA interleave memory policy for the InnoDB buffer pool allocation, but forces NUMA interleaved allocation at the buffer pool initialization time. Bug fixed #3967.
- `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.
- 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.
- Fixes were introduced to remove GCC 8 compilation warnings for the *Percona Server for MySQL* build. Bug fixed #3950.
- An InnoDB Memcached Plugin code clean-up was backported from MySQL 8.0. Bug fixed #4506.
- *Percona Server for MySQL* could not be built with `-DWITH_LZ4=system` option on Ubuntu 14.04 (Trusty) because of too old LZ4 packages. Bug fixed #3842.
- A regression brought during TokuDB code clean-up in `5.7.21-21` was causing assertion in cases when the FT layer returns an error during an alter table operation. Bug fixed #4294.

16.28.3 MyRocks Changes and Fixes

- `UPDATE` statements were returning incorrect results because of not making a full table scan on tables with unique secondary index. Bug fixed #4495 (upstream Facebook/mysql-5.6#830).

16.28.4 Other Bugs Fixed

- #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"
- #4488 "`-Werror` is always disabled for `innodb_memcached`"
- #1114 "Assertion `inited == INDEX' failed"
- #1130 "RBR Replication with concurrent XA in READ-COMMITTED takes supremum pseudo-records and breaks replication"

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.29 Percona Server 5.7.21-21 (2018-04-24)

Percona is glad to announce the release of Percona Server 5.7.21-21 on April 24, 2018. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.21 and includes all the bug fixes in it. Percona Server 5.7.21-21 is now the current GA (Generally Available) release in the 5.7 series. This version of Percona Server for MySQL marks the following encryption features, previously available as beta, as GA: Vault keyring plugin, encryption for InnoDB general tablespaces, and encryption for binary log files.

All software developed by Percona is open-source and free.

16.29.1 New Features

- A new variable, `innodb_temp_tablespace_encrypt`, is introduced to turn encryption of temporary tablespace and temporary InnoDB file-per-table tablespaces on/off. Bug fixed #3821.
- A new variable, `innodb_encrypt_online_alter_logs`, simultaneously turns on encryption of files used by InnoDB for merge sort, online DDL logs, and temporary tables created by InnoDB for online DDL. Bug fixed #3819.
- A new variable, `innodb_encrypt_tables`, can be set to `ON`, making InnoDB tables encrypted by default, to `FORCE`, disabling creation of unencrypted tables, or `OFF`, restoring the like-before behavior. Bug fixed #1525.
- Query response time plugin now can be disabled at session level with use of a new variable, `query_response_time_session_stats`.

16.29.2 Bugs Fixed

- Attempting to use a partially-installed query response time plugin could have caused server crash. Bug fixed #3959.
- There was a server crash caused by a materialized temporary table from semi-join optimization with key length larger than 1000 bytes. Bug fixed #296.
- A regression in the original 5.7 port was causing integer overflow with `thread_pool_stall_limit` variable values bigger than 2 seconds. Bug fixed #1095.
- A memory leak took place in *Percona Server for MySQL* when performance schema is used in conjunction with thread pooling. Bug fixed #1096.
- A code clean-up was done to fix compilation with clang, both general warnings (bug fixed #3814, upstream #89646) and clang 6 specific warnings and errors (bug fixed #3893, upstream #90111).
- Compilation warning was fixed for -DWITH_QUERY_RESPONSE_TIME=ON CMake compilation option, which makes QRT to be linked statically. Bug fixed #3841.
- *Percona Server for MySQL* returned empty result for `SELECT` query if number of connections exceeded 65535. Bug fixed #314 (upstream #89313).
- A clean-up in *Percona Server for MySQL* binlog-related code was made to avoid uninitialized memory comparison. Bug fixed #3925 (upstream #90238).
- `mysqldump` utility with `--innodb-optimize-keys` option was incorrectly working with foreign keys on the same table, producing invalid SQL statements. Bugs fixed #1125 and #3863.
- A fix of the mysqld startup script failed to detect jemalloc library location for preloading, thus not starting on systemd based machines, introduced in *Percona Server for MySQL* `5.7.21-20`, was improved to take into account previously created configuration file. Bug fixed #3850.

- The possibility of a truncated bitmap file name was fixed in InnoDB logging subsystem. Bug fixed #3926.
- Temporary file I/O was not instrumented for Performance Schema. Bug fixed #3937 (upstream #90264).
- A crash in the unsafe query warning checks with views took place for `UPDATE` statement in case of statement binlogging format. Bug fixed #290.

16.29.3 MyRocks Changes

- A re-implemented variable, `rpl_skip_tx_api`, allows a user to turn on simple RocksDB write batches functionality, increasing replication performance by the transaction API skip. Bug fixed MYR-47.
- Decoding value-less padded varchar fields could under some circumstances cause assertion and/or data corruption. Bug fixed MYR-232.

16.29.4 TokuDB Changes

- Two new variables introduced to facilitate the TokuDB fast updates feature, `tokudb_enable_fast_update`, and `tokudb_enable_fast_upsert`. Bugs fixed #63 and #148.
- A set of compilation fixes was introduced to make TokuDB successfully build in MySQL / *Percona Server for MySQL* 8.0. Bugs fixed #84, #85, #114, #115, #118, #128, #139, #141, and #172.
- Conditional compilation code dependent on version ID in the TokuDB tree was separated and arranged to specific version branches. Bugs fixed #133, #134, #135, and #136.
- `ALTER TABLE ... COMMENT = ...` statement caused TokuDB to rebuild the whole table, which is not needed, as only FRM metadata should be changed. Bugs fixed #130 and #137.
- Data race on the cache table pair attributes was fixed. Bug fixed #109.

Other bugs fixed: #3793, #3812, #3813, #3815, #3818, #3835, #3875 (upstream #89916), #3843 (upstream #89822), #3848, #3856, #3887, MYR-160, MYR-245, #109, #111, #180, #181, #182, and #188.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.30 Percona Server 5.7.21-20 (2018-02-19)

Percona is glad to announce the release of Percona Server 5.7.21-20 on February 19, 2018. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.21 and includes all the bug fixes in it. Percona Server 5.7.21-20 is now the current GA (Generally Available) release in the 5.7 series. All software developed by Percona is open-source and free.

16.30.1 New Features

- A new string variable `version_suffix` allows changing suffix for the *Percona Server for MySQL* version string returned by the read-only `version` variable. Also `version_comment` is converted from a global read-only to a global read-write variable.
- A new `keyring_vault_timeout` variable allows setting the number of seconds for the Vault server connection timeout. Bug fixed #298.

16.30.2 Bugs Fixed

- The mysqld startup script was unable to detect jemalloc library location for preloading, and that prevented starting the *Percona Server for MySQL* on systemd-based machines. Bugs fixed #3784 and #3791.
- There was a problem with fulltext search, which could find a word with punctuation marks in natural language mode only, but not in boolean mode. Bugs fixed #258, #2501 (upstream #86164).
- Build errors were present on FreeBSD (caused by fixing the bug #255 in *Percona Server for MySQL* `5.6.38-83.0` and on MacOS (caused by fixing the bug #264 in *Percona Server for MySQL* `5.7.20-19`. Bugs fixed #2284 and #2286.
- A bunch of fixes were introduced to remove GCC 7 compilation warnings for the *Percona Server for MySQL* build. Bugs fixed #3780 (upstream #89420, #89421, and #89422).
- CMake error took place at compilation with bundled zlib. Bug fixed #302.
- A GCC 7 warning fix introduced a regression in *Percona Server for MySQL* that led to a wrong SQL query built to access the remote server when the Federated storage engine was used. Bug fixed #1134.
- It was possible to enable `encrypt_binlog` with no binary or relay logging enabled. Bug fixed #287.
- Long buffer wait times were occurring on busy servers in case of the `IMPORT TABLESPACE` command. Bug fixed #276.
- 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.
- Percona Server now uses *Travis CI* for additional tests. Bug fixed #3777.

Other bugs fixed: #257, #264, #1090 (upstream #78048), #1109, #1127, #2204, #2414, #2415, #3767, #3794, and #3804 (upstream #89598).

This release also contains fixes for the following CVE issues: CVE-2018-2565, CVE-2018-2573, CVE-2018-2576, CVE-2018-2583, CVE-2018-2586, CVE-2018-2590, CVE-2018-2612, CVE-2018-2600, CVE-2018-2622, CVE-2018-2640, CVE-2018-2645, CVE-2018-2646, CVE-2018-2647, CVE-2018-2665, CVE-2018-2667, CVE-2018-2668, CVE-2018-2696, CVE-2018-2703, CVE-2017-3737.

16.30.3 MyRocks Changes

- A new behavior makes *Percona Server for MySQL* fail to restart on detected data corruption; `rocksdb_allow_to_start_after_corruption` variable can be passed to `mysqld` as a command line parameter to switch off this restart failure.
- A new cmake option `ALLOW_NO_SSE42` was introduced to allow MyRocks build on hosts not supporting SSE 4.2 instructions set, which makes MyRocks usable without FastCRC32-capable hardware. Bug fixed MYR-207.
- `rocksdb_bytes_per_sync` and `rocksdb_wal_bytes_per_sync` variables were turned into dynamic ones.
- `rocksdb_flush_memtable_on_analyze` variable has been removed.
- `rocksdb_concurrent_prepare` is now deprecated, as it has been renamed in upstream to `rocksdb_two_write_queues`.
- `rocksdb_row_lock_deadlocks` and `rocksdb_row_lock_wait_timeouts` global status counters were added to track the number of deadlocks and the number of row lock wait timeouts.
- Creating a table with string indexed column to non-binary collation now generates a warning about using inefficient collation instead of an error. Bug fixed MYR-223.

16.30.4 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.
- A data race was fixed in minicron utility of the PerconaFT. Bug fixed TDB-107.
- Row count and cardinality decrease to zero took place after long-running `REPLACE` load.

Other bugs fixed: TDB-48, TDB-78, TDB-93, and TDB-99.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.31 Percona Server 5.7.20-19 (2018-01-03)

Percona is glad to announce the release of Percona Server 5.7.20-19 on January 3, 2018. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.20 and includes all the bug fixes in it. Percona Server 5.7.20-19 is now the current GA (Generally Available) release in the 5.7 series. All software developed by Percona is open-source and free. Details of this release can be found in the 5.7.20-19 milestone on Launchpad.

16.31.1 New Features

- Now MyRocks Storage Engine has *General Availability* status.
- Binary log encryption has been implemented and can be now switched on using the `encrypt_binlog` variable.
- `innodb_print_lock_wait_timeout_info` variable, introduced in the previous release, but undocumented, allows to log information about all lock wait timeout errors.

16.31.2 Bugs Fixed

- Intermediary slave with Blackhole storage engine couldn't record updates from master to its own binary log in case the master has the `binlog_rows_query_log_events` option enabled. Bug fixed #1722789 (upstream #88057).
- Help command in the MySQL command line client provided a link to an older version of the *Percona Server for MySQL* manual. Bug fixed #1708073.
- A regression in the `mysqld_safe` script forced it to print an extra error when stopping the MySQL service. Bugs fixed #1738742.
- Blackhole storage engine was incompatible with a newer length limit of the InnoDB index key prefixes. Bug fixed #1733049 (upstream #53588).
- Heartbeats received by a slave were reacted with `mysql.slave_master_info` table sync on each of them even with `sync_master_info` set to zero, causing a huge increase in write load. Bug fixed #1708033 (upstream #85158).

16.31.3 MyRocks Changes

- The replication writebatch functionality has been removed from *Percona Server for MySQL* 5.7 due to the unsafety of the current implementation.
- The variables `rocksdb_block_cachecompressed_hit`, `rocksdb_block_cachecompressed_miss`, and `rocksdb_getupdatessince_calls` were renamed to `rocksdb_block_cache_compressed_hit`, `rocksdb_block_cache_compressed_miss`, and `rocksdb_get_updates_since_calls` respectively.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.32 Percona Server 5.7.20-18 (2017-12-14)

Percona is glad to announce the release of Percona Server 5.7.20-18 on December 14. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.20 and includes all the bug fixes in it. Percona Server 5.7.20-18 is now the current GA release in the 5.7 series. All software developed by Percona is open-source and free. Details of this release can be found in the 5.7.20-18 milestone on Launchpad.

16.32.1 New Features

- *Percona Server for MySQL* packages are now available for *Ubuntu 17.10 (Artful)*.
- As part of InnoDB Full-Text Search improvements a new `innodb_ft_ignore_stopwords` variable has been implemented which controls whether InnoDB Full-Text Search should ignore the stopword list when building/updating an FTS index. This feature is also fixing bug #1679135 (upstream #84420).
- *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`.
- As part of Data at Rest Encryption, *Percona Server for MySQL* has implemented support for innodb_general_tablespace_encryption and Loading the Keyring Plugin. This feature is considered **BETA** quality.

16.32.2 Bugs Fixed

- *Percona Server for MySQL* 5.7 docker images did not include TokuDB. Bugs fixed #1682419 and #1699241.
- 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*).
- 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.
- 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).
- The `CREATE TABLE ... LIKE ...` statement did not use source `row_format` on the target TokuDB table. Bug fixed #76.
- TokuDB would encode an already encoded database name for a directory name. Bug fixed #74.
- Optimizer would pick the wrong index for TokuDB tables having a hot created index, unless `ALTER TABLE` was run. Bug fixed #35.

Other bugs fixed: #1720810, #83, #80, and #75.

16.32.3 MyRocks Changes

- RocksDB has implemented a FlushWAL API which improves upon the performance of MySQL 2-phase-commit during binary log group commit flush stage. This feature adds support for using the FlushWAL API in MyRocks and also matches `rocksdb_flush_log_at_trx_commit` variable with `innodb_flush_log_at_trx_commit` behavior. Two implement this feature `rocksdb_manual_wal_flush` and `rocksdb_concurrent_prepare` variables have been implemented.
- New - `rocksdb_force_compute_memtable_stats_cachetime` variable has been implemented that can be used to specify how long the cached value of memtable statistics should be used instead of computing it every time during the query plan analysis.
- New - `rocksdb_large_prefix` variable has been implemented which, when enabled, allows index key prefixes longer than 767 bytes (up to 3072 bytes). This option mirrors the innodb_large_prefix The values for this variable should be the same between master and slave.
- New - `rocksdb_max_background_jobs` variable has been implemented to replace `rocksdb_base_background_compactions`, `rocksdb_max_background_compactions`, and `rocksdb_max_background_flushes` variables. This variable specifies the maximum number of background jobs. It automatically decides how many threads to allocate towards flush/compaction. It was implemented to reduce the number of (confusing) options for users and can tweak and push the responsibility down to the RocksDB level.
- New - `rocksdb_sim_cache_size` variable has been implemented to enable the simulated cache. This can be used to figure out the hit/miss rate with a specific cache size without changing the real block cache.
- Input can be now sorted by the Primary Key during the bulkload by enabling the `rocksdb_bulk_load_allow_unsorted` variable.
- New - `rocksdb_ignore_unknown_options` variable has been implemented, which when enabled (default) allows RocksDB to receive unknown options and not exit.

Download PDF

16.33 Percona Server 5.7.19-17 (2017-08-31)

Percona is glad to announce the release of Percona Server 5.7.19-17 on August 31, 2017. Downloads are available here and from the Percona Software Repositories.

This release is based on MySQL 5.7.19 and includes all the bug fixes in it. Percona Server 5.7.19-17 is now the current GA release in the 5.7 series. All software developed by Percona is open-source and free. Details of this release can be found in the 5.7.19-17 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 announcement is that these platforms have reached the end of life, will not receive updates, and are not recommended for use in production.

16.33.1 New Features

- Included the Percona MyRocks storage engine

Note

MyRocks for Percona Server is currently experimental and not recommended for production deployments until further notice. You are encouraged to try it in a testing environment and provide feedback or report bugs.

- #1708087: Added the `mysql-helpers` script to handle checking for missing `datadir` during startup. Also fixes #1635364.

16.33.2 Platform Support

- Stopped providing packages for Ubuntu 12.04 due to its end of life.

16.33.3 Bugs Fixed

- #1669414: Fixed handling of failure to set `O_DIRECT` on parallel doublewrite buffer file.
- #1705729: Fixed the `postinst` script to correctly locate the `datadir`. Also fixes #1698019.
- #1709811: Fixed `yum upgrade` to not enable the `mysqld` service if it was disabled before the upgrade.
- #1709834: Fixed the `mysqld_safe` script to correctly locate the `basedir`.
- Other fixes: #1698996, #1706055, #1706262, #1706981, #1686340 (upstream #86799), #1654256 (upstream #84640), and #1651941 (upstream #84442).

16.33.4 TokuDB Changes

- TDB-70: Removed redundant `fsync` of TokuDB redo log during binlog group commit flush stage. This fixes the issue that prevented TokuDB to run in reduced durability mode when the binlog was enabled.
- TDB-72: Fixed issue when renaming a table with non-alphanumeric characters in its name.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.34 Percona Server for MySQL 5.7.18-16 (2017-07-28)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.18-16 on July 28, 2017 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.18, including all the bug fixes in it, *Percona Server for MySQL* 5.7.18-16 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.18-16 milestone at Launchpad

Please note that RHEL 5, CentOS 5, and Ubuntu versions 12.04 and older are not supported in future releases of *Percona Server for MySQL* and no further packages are added for these distributions.

16.34.1 New Feature

Percona Server for MySQL is now available on Debian 9 (stretch). The support only covers the `amd64` architecture.

Percona Server for MySQL can now be built with the support of OpenSSL 1.1.

MyRocks storage engine has been merged into *Percona Server for MySQL*.

TokuDB is able to kill a query that is awaiting an FT locktree lock.

TokuDB enables using the `MySQL DEBUG_SYNC` facility within *Percona FT*.

16.34.2 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.

Percona FT could assert when opening a dictionary with no useful information to the error log. Bug fixed #23.

Percona FT could assert for various reasons deserializing nodes with no useful error output. Bug fixed #24.

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).

Packaging was using the `dpkg --verify` command which is not available on wheezy/precise. Bug fixed #1694907.

Enabling and disabling the slow query log rotation spuriously added the version suffix to the next slow query log file name. Bug fixed #1704056.

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.

Percona FT 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 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).

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.

When running `SHOW ENGINE INNODB STATUS`, the `Buffer pool size, bytes` entry contained **0**. Bug fixed #1586262.

The synchronization between the LRU manager and page cleaner threads was not done at shutdown. Bug fixed #1689552.

Removed spurious `lock_wait_timeout_thread` wakeups, potentially reducing `lock_sys_wait_mutex` contention. Patch by Inaam Rama merged from `WebScaleSQL`. Bug fixed #1704267 (upstream #72123).

Other bugs fixed: #1686603, #6, #44, #65, #1160986, #1686934, #1688319, #1689989, #1690012, #1691682, #1697700, #1699788, #1121072, and #1684601 (upstream #86016).

Note

Due to new package dependency, Ubuntu/Debian users should use `apt-get dist-upgrade` or `apt-get install percona-server-server-5.7` to upgrade.

16.34.3 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

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.35 Percona Server for MySQL 5.7.18-15 (2017-05-26)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.18-15 on May 26, 2017 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.18, including all the bug fixes in it, *Percona Server for MySQL* 5.7.18-15 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.18-15 milestone at Launchpad

16.35.1 Bugs Fixed

The server could crash when querying the partitioning table with a single partition. Bug fixed #1657941 (upstream #76418).

Running a query on the InnoDB table with ngram full-text parser and a `LIMIT` clause could lead to a server crash. Bug fixed #1679025 (upstream #85835).

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.36 Percona Server for MySQL 5.7.18-14 (2017-05-12)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.18-14 on May 12, 2017 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.18, including all the bug fixes in it, *Percona Server for MySQL* 5.7.18-14 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.18-14 milestone at Launchpad

16.36.1 New Features

Percona Server for MySQL 5.7 packages are now available for Ubuntu 17.04 (*Zesty Zapus*).

Percona Server for MySQL now supports Prefix Index Queries Optimization. This feature was ported from a Facebook MySQL patch.

Percona Server for MySQL has implemented support for Gap locks detection for transactional storage engines, like *MyRocks*, that do not support gap locks. This feature was ported from a Facebook MySQL patch.

`tokudb_dir_cmd` can now be used to edit the TokuDB directory map. This feature is currently considered Experimental.

16.36.2 Bugs Fixed

A deadlock could occur in I/O-bound workloads when the server was using several small buffer pool instances in combination with small redo log files and variable

`innodb_empty_free_list_algorithm` set to `backoff` algorithm. Bug fixed #1651657.

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 a failure to create a multi-threaded slave worker thread. Bug fixed #1675716.

In-Place upgrade from *Percona Server for MySQL* 5.6 to 5.7 by using standalone packages would fail if `/var/lib/mysql` wasn't defined as the `datadir`. Bug fixed #1687276.

A combination of using any audit API-using plugin, like Audit Log Plugin and Response Time Distribution, with multi-byte collation connection and a `PREPARE` statement with a parse error could lead to a server crash. Bug fixed #1688698 (upstream #86209).

Fix for a #1433432 bug that 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 an 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).

Processing GTIDs in the relay log that were already been executed were causing write/fsync amplification. Bug fixed #1669928 (upstream #85141).

Text/BLOB fields were not handling sorting of the empty string consistently between InnoDB and filesort. Bug fixed #1674867 (upstream #81810) by porting a Facebook patch for MySQL.

InnoDB adaptive hash index was using a partitioning algorithm that would produce uneven distribution when the server contained many tables with an identical schema. Bug fixed #1679155 (upstream #81814).

For plugin variables that are signed numbers, doing a `SHOW VARIABLES` would always show an unsigned number. Fixed by porting a Facebook patch for MySQL.

Other bugs fixed: #1629250 (upstream #83245), #1660828 (upstream #84786), #1664519 (upstream #84940), #1674299, #1670588 (upstream #84173), #1672389, #1674507, #1675623, #1650294, #1659224, #1662908, #1669002, #1671473, #1673800, #1674284, #1676441, #1676705, #1676847 (upstream #85671), #1677130 (upstream #85678), #1677162, #1677943, #1678692, #1680510 (upstream #85838), #1683993, #1684012, #1684078, #1684264, #1687386, #1687432, #1687600, and #1674281.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.37 Percona Server for MySQL 5.7.17-13 (2017-04-05)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.17-13 on April 5th, 2017 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.17, including all the bug fixes in it, *Percona Server for MySQL* 5.7.17-13 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.17-13 milestone at Launchpad

16.37.1 Bugs Fixed

MyRocks storage engine detection implemented in `mysqldump` in *Percona Server for MySQL* `5.6.17-12` was using a deprecated `INFORMATION_SCHEMA.SESSION_VARIABLES` table, causing `mysqldump` failures on servers running with the `show_compatibility_56` variable set to `OFF`. Bug fixed #1676401.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.38 Percona Server for MySQL 5.7.17-12 (2017-03-24)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.17-12 on March 24th, 2017 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.17, including all the bug fixes in it, *Percona Server for MySQL* 5.7.17-12 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.17-12 milestone at Launchpad

16.38.1 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 the storage engine is using a reverse order column family for a primary key.

mysqldump will now detect when MyRocks is installed and available by seeing if there is a session variable named `rocksdb_skip_fill_cache` and setting it to `1` if it exists.

Now **mysqldump** automatically enables the session variable `rocksdb_bulk_load` if it is supported by the target server.

16.38.2 Bugs Fixed

If the variable `thread_handling` was set to `pool-of-threads` in the MySQL configuration file, the server couldn't be gracefully shut down by a `SIGTERM` signal. Bug fixed #1537554.

When `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).

Executing the `FLUSH LOGS` on a read-only slave with a user that doesn't have the `SUPER` privilege would result in `Error 1290`. Bug fixed #1652852 (upstream #84350).

`FLUSH LOGS` was disabled with `read_only` and `super_read_only` variables. Bug fixed #1654682 (upstream #84437).

If `SHOW BINLOGS` or `PERFORMANCE_SCHEMA.GLOBAL_STATUS` query, and a transaction commit would run in parallel, they could deadlock. Bug fixed #1657128.

A long-running binary log commit would block `SHOW STATUS`, which in turn could block a number of other operations such as client connects and disconnects. Bug fixed #1646100.

Log tracking initialization did not find the last valid bitmap data correctly. Bug fixed #1658055.

A query using a range scan with a complex range condition could lead to a server crash. Bug fixed #1660591 (upstream #84736).

Race condition between buffer pool page optimistic access and eviction could lead to a server crash. Bug fixed #1664280.

If Audit Log Plugin was unable to create a file pointed by `audit_log_file`, the 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.

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.

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).

Other bugs fixed: #1665545, #1650321, #1654501, #1663251, #1659548, #1663452, #1670834, #1672871, #1626545, #1658006, #1658021, #1659218, #1659746, #1660239, #1660243, #1660348, #1662163 (upstream #81467), #1664219, #1664473, #1671076, and #1671123.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.39 Percona Server for MySQL 5.7.17-11 (2017-02-03)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.17-11 on February 3rd, 2017 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.17, including all the bug fixes in it, *Percona Server for MySQL* 5.7.17-11 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.17-11 milestone at Launchpad

16.39.1 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.

The 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.

16.39.2 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).

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 `USER_STATISTICS` table was showing incorrect values. Bug fixed #728082.

Audit Log Plugin when set to `JSON` format was not escaping characters properly. Bug fixed #1548745.

InnoDB index dives did not detect some of the concurrent tree changes, which could return bogus estimates. Bug fixed #1625151 (upstream #84366).

`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.

Lack of free pages in the buffer pool is not diagnosed with `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 `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, #1617715, #1633988, #1638198 (upstream #82823), #1642230, #1646384, #1640810, #1647530, #1651121, #1658843, #1156772, #1644583, #1648389, #1648737, #1650256, and #1647723.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.40 Percona Server for MySQL 5.7.16-10 (2016-11-28)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.16-10 on November 28th, 2016 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.16, including all the bug fixes in it, *Percona Server for MySQL* 5.7.16-10 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.16-10 milestone at Launchpad

16.40.1 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.

16.40.2 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 `log_slow_sp_statements` was overwritten by the value of `log_slow_admin_statements`. Bug fixed #719368.

The server wouldn't start after crash with with `innodb_force_recovery` set to `6` if a parallel doublewrite file existed. Bug fixed #1629879.

The Thread Pool `thread limit reached` and `failed to create thread` messages are now printed on the first occurrence as well. Bug fixed #1636500.

`INFORMATION_SCHEMA.TABLE_STATISTICS` and `INFORMATION_SCHEMA.INDEX_STATISTICS` tables were not correctly updated for TokuDB. Bug fixed #1629448.

Other bugs fixed: #1633061, #1633430, and #1635184.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.41 Percona Server for MySQL 5.7.15-9 (2016-10-21)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.15-9 on October 21st, 2016 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.15, including all the bug fixes in it, *Percona Server for MySQL* 5.7.15-9 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.15-9 milestone at Launchpad

16.41.1 New Features

A new TokuDB `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 enabled by default.

16.41.2 Bugs Fixed

The Audit Log plugin malformed record could be written after `audit_log_flush` was set to `ON` in `ASYNC` and `PERFORMANCE` modes. Bug fixed #1613650.

Running `SELECT DISTINCT x...ORDER BY y LIMIT N,N` could lead to a server crash. Bug fixed #1617586.

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 `thread_pool_high_prio_mode` set to `transactions`. Fixed by placing such statements into the high priority queue even with the above `thread_pool_high_prio_mode` setting. Bugs fixed #1619559 and #1374930.

Fixed memory leaks in Audit Log Plugin. Bug fixed #1620152 (upstream #71759).

A 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.

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 recovery of a torn page from the doublewrite buffer was shown as a warning in the error log. Bug fixed #1622985.

LRU manager threads 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).

The shutdown waiting for a purge to complete was undiagnosed for the first minute. Bug fixed #1616785.

Other bugs fixed: #1614439, #1614949, #1624993 (#736), #1613647, #1615468, #1617828, #1617833, #1626002 (upstream #83073), #904714, #1610102, #1610110, #1613728, #1614885, #1615959, #1616333, #1616404,

#1616768, #1617150, #1617216, #1617267, #1618478, #1618819, #1619547, #1619572, #1620583, #1622449, #1623011, #1624992 (#1014), #735, #1626500, #1628913, #952920, and #964.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.42 Percona Server for MySQL 5.7.14-8 (2016-09-21)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.14-8 on September 21st, 2016 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.14, including all the bug fixes in it, *Percona Server for MySQL* 5.7.14-8 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.14-8 milestone at Launchpad

16.42.1 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 a 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.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.43 Percona Server for MySQL 5.7.14-7 (2016-08-23)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.14-7 on August 23rd, 2016 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.13, including all the bug fixes in it, *Percona Server for MySQL* 5.7.14-7 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.14-7 milestone at Launchpad

16.43.1 New Features

Percona Server for MySQL Audit Log Plugin now supports filtering by user, database, and sql_command.

Percona Server for MySQL now supports tree map file block allocation strategy for TokuDB.

16.43.2 Bugs Fixed

Fixed the 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.

In the client library, any EINTR received during network I/O was not handled correctly. Bug fixed #1591202 (upstream #82019).

`SHOW GLOBAL STATUS` was locking more than the upstream implementation which made it less suitable to be called with high frequency. Bug fixed #1592290.

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.

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.

Changed page tracking was initialized too late by InnoDB. Bug fixed #1612574.

Fixed stack buffer overflow if `--ssl-cipher` had more than 4000 characters. Bug fixed #1596845 (upstream #82026).

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 a set of fixes to make PerconaFT build and run on the AArch64 (64-bit ARMv8) architecture. Bug fixed #726 (*Alexey Kopytov*).

Other bugs fixed: #1542874 (upstream #80296), #1610242, #1604462 (upstream #82283), #1604774 (upstream #82307), #1606782, #1607359, #1607606, #1607606, #1607671, #1609422, #1610858, #1612551, #1613663, #1613986, #1455430, #1455432, #1581195, #998, #1003, and #730.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.44 Percona Server for MySQL 5.7.13-6 (2016-07-16)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.13-6 on July 6th, 2016 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.13, including all the bug fixes in it, *Percona Server for MySQL* 5.7.13-6 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.13-6 milestone at Launchpad

16.44.1 New Features

TokuDB MTR suite is now part of the default MTR suite in *Percona Server for MySQL* 5.7.

16.44.2 Bugs Fixed

Querying the `GLOBAL_TEMPORARY_TABLES` table would cause a server crash if temporary table owning threads would execute new queries. Bug fixed #1581949.

`IMPORT TABLESPACE` and undo tablespace truncate could get stuck indefinitely with a writing workload in parallel. Bug fixed #1585095.

Requesting to flush the whole of the buffer pool with doublewrite parallel buffer wasn't working correctly. Bug fixed #1586265.

Audit Log Plugin would hang when trying to write a log record of `audit_log_buffer_size` length. Bug fixed #1588439.

Audit log in `ASYNC` mode could skip log records that don't fit into the log buffer. Bug fixed #1588447.

In order to support `innodb_flush_method` being set to `ALL_O_DIRECT`, the log I/O buffers were aligned to `innodb_log_write_ahead_size`. This missed that the variable is dynamic and could still cause the server to crash. Bug fixed #1597143.

InnoDB tablespace import would fail when trying to import a table with a different data directory. Bug fixed #1548597 (upstream #76142).

The Audit Log plugin was truncating SQL queries to 512 bytes. Bug fixed #1557293.

`mysqlbinlog` did not free the existing connection before opening a new remote one. Bug fixed #1587840 (upstream #81675).

Fixed a memory leak in `mysqldump`. Bug fixed #1588845 (upstream #81714).

Transparent Huge Pages check will now only happen if `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: #1541698 (upstream #80261), #1587426 (upstream, #81657), #1589431, #956, #964,

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.45 Percona Server for MySQL 5.7.12-5 (2016-06-06)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.12-5 on June 6th, 2016 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.12, including all the bug fixes in it, *Percona Server for MySQL* 5.7.12-5 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.12-5 milestone at Launchpad

16.45.1 Bugs Fixed

`MEMORY` storage engine did not support JSON columns. Bug fixed #1536469.

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 different memory allocators, this was causing `safemalloc` warnings in debug builds and crashes because memory accounting didn't add up. Bug fixed #1546538 (#962).

Adding an index to an InnoDB temporary table while `expand_fast_index_creation` was enabled could lead to server assertion. Bug fixed #1554622.

Percona Server for MySQL was missing the `innodb_numa_interleave` server variable. Bug fixed #1561091 (upstream #80288).

Running `SHOW STATUS` in parallel to online buffer pool resizing could lead to a server crash. Bug fixed #1577282.

InnoDB crash recovery might fail if `innodb_flush_method` was set to `ALL_O_DIRECT`. Bug fixed #1529885.

Fixed heap allocator/deallocator mismatch in Metrics for scalability measurement. Bug fixed #1581051.

Percona Server for MySQL is now built with the system `zlib` library instead of the older bundled one. Bug fixed #1108016.

`CMake` would fail if TokuDB tests passed. Bug fixed #1521566.

Reduced the memory overhead per page in the InnoDB buffer pool. The fix was based on the Facebook patch #91e979e. Bug fixed #1536693 (upstream #72466).

The `CREATE TABLE ... LIKE ...` statement 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.

Parallel doublewrite memory was not freed with `innodb_fast_shutdown` set to `2`. Bug fixed #1578139.

The server will now show a more descriptive error message when *Percona Server for MySQL* fails with `errno == 22 "Invalid argument"` if `innodb_flush_method` was set to `ALL_O_DIRECT`. Bug fixed #1578604.

The error log warning `Too many connections` was only printed for connection attempts when `max_connections` + 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.

`apt-cache show` command for `percona-server-client` was showing `innotop` included as part of the package. Bug fixed #1201074.

A replication slave would fail to connect to a master running 5.5. Bug fixed #1566642 (upstream #80962).

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* `5.7.11-4`. Bug fixed #717.

Fixed `jemalloc` version parsing error. Bug fixed #528.

If `ALTER TABLE` was run while `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_pk_insert_mode` session variable has been deprecated and the behavior will be that of the former `tokudb_pk_insert_mode` set to `1`. The optimization will be used were safe and not used were not safe. 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 a 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: #1510564 (upstream #78981), #1533482 (upstream #79999), #1553166, #1496282 (#964), #1496786 (#956), #1566790, #718, #914, #937, #954, #955, #970, #971, #972, #976, #977, #981, #982, #637, and #982.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.46 Percona Server for MySQL 5.7.11-4 (2016-03-15)

Percona is glad to announce the GA (Generally Available) release of *Percona Server for MySQL* 5.7.11-4 on March 15th, 2016 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.11, including all the bug fixes in it, *Percona Server for MySQL* 5.7.11-4 is the current GA release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.11-4 milestone at Launchpad

16.46.1 New Features

Percona Server for MySQL has implemented a Parallel doublewrite buffer.

The TokuDB Background ANALYZE TABLE feature is now enabled by default (`tokudb_analyze_in_background` is set to `ON` by default). Variable `tokudb_auto_analyze` default value has been changed from `0` to `30` . (#935)

Suppress Warning Messages feature has been removed from *Percona Server for MySQL* 5.7 because MySQL 5.7.11 has implemented a new system variable, log_statements_unsafe_for_binlog, which implements the same effect.

16.46.2 Bugs Fixed

If `pid-file` option wasn't specified with the full path, *Ubuntu*/*Debian* `sysvinit` script wouldn't notice the server is actually running and it will timeout or in some cases even hang. Bug fixed #1549333.

Buffer pool may fail to remove dirty pages for a particular tablespace from the flush list, as requested by, for example, `DROP TABLE` or `TRUNCATE TABLE` commands. This could lead to a crash. Bug fixed #1552673.

Audit Log Plugin worker thread may crash on write call writing fewer bytes than requested. Bug fixed #1552682 (upstream #80606).

Percona Server for MySQL 5.7 `systemd` script now takes the last option specified in `my.cnf` if the same option is specified multiple times. Previously it would try to take all values which would break the script and server would fail to start. Bug fixed #1554976.

`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.

Other bugs fixed: #1521120, #1549301 (upstream #80496), and #1554043 (upstream #80607).

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.47 Percona Server for MySQL 5.7.10-3 (2016-02-23)

Percona is glad to announce the first GA (Generally Available) release of *Percona Server for MySQL* 5.7.10-3 on February 23rd, 2016 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.10, including all the bug fixes in it, *Percona Server for MySQL* 5.7.10-3 is the current Generally Available release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.10-3 milestone at Launchpad

16.47.1 New Features

A complete list of changes between *Percona Server for MySQL* 5.6 and 5.7 can be seen in Changed in Percona Server 5.7.

Percona Server for MySQL has implemented a multi-threaded asynchronous LRU flusher. This work also allows to safely use of the `backoff` value for the `innodb_empty_free_list_algorithm` server system variable, and its default has been changed accordingly.

16.47.2 Known Issues

In *Percona Server for MySQL* 5.7 super_read_only feature has been replaced with upstream implementation. There are currently two known issues compared to *Percona Server for MySQL* 5.6 implementation:

- Bug #78963, `super_read_only` aborts `STOP SLAVE` if variable `relay_log_info_repository` is set to `TABLE` which could lead to a server crash in Debug builds.
- Bug #79328, `super_read_only` set as a server option has no effect.

InnoDB crash recovery might fail if `innodb_flush_method` is set to `ALL_O_DIRECT`. The workaround is to set this variable to a different value before starting up the crashed instance (bug #1529885).

16.47.3 Bugs Fixed

Percona Server for MySQL `5.7.10-1` didn't write the initial root password into the log file `/var/log/mysqld.log` during the installation on *CentOS 6*. Bug fixed #1541769.

The cardinality of partitioned TokuDB tables became inaccurate after the changes introduced by the TokuDB Background ANALYZE TABLE feature in *Percona Server for MySQL* `5.7.10-1`. 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.

Percona TokuBackup would fail with an unclear error if the backup process found `mysqld_safe.pid` file (owned by root) inside the `datadir`. Fixed by excluding the `pid` file by default. Bug fixed #125.

The PAM Authentication Plugin build warning has been fixed. Bug fixed #1541601.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.48 Percona Server for MySQL 5.7.10-2 (2016-02-05)

Percona is glad to announce the second Release Candidate release of *Percona Server for MySQL* 5.7.10-2 on February 5th, 2016 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.10, including all the bug fixes in it, *Percona Server for MySQL* 5.7.10-2 is the current Release Candidate release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.10-2 milestone at Launchpad

16.48.1 New Features

A complete list of changes between *Percona Server for MySQL* 5.6 and 5.7 can be seen in Changed in Percona Server 5.7.

The 5.7 binlog group commit algorithm is now supported in TokuDB as well.

New TokuDB index statistics reporting has been implemented to be compatible with the changes implemented in upstream 5.7. Following the InnoDB example, the default value for `tokudb_cardinality_scale_percent` has been changed from `50%` to `100%`. Implementing this feature also addresses a server crash deep in the optimizer code.

16.48.2 Known Issues

In *Percona Server for MySQL* 5.7 super_read_only feature has been replaced with upstream implementation. There are currently two known issues compared to *Percona Server for MySQL* 5.6 implementation:

- Bug #78963, `super_read_only` aborts `STOP SLAVE` if `relay_log_info_repository` is set to `TABLE` which could lead to a server crash in Debug builds.
- Bug #79328, `super_read_only` set as a server option has no effect. InnoDB crash recovery might fail if `innodb_flush_method` is set to `ALL_O_DIRECT`. The workaround is to set this variable to a different value before starting up the crashed instance (bug #1529885).

16.48.3 Bugs Fixed

Clustering secondary index could not be created on a partitioned TokuDB table. Bug fixed #1527730 (#720).

Percona TokuBackup was failing to compile with *Percona Server for MySQL* 5.7. Bug fixed #123.

Granting privileges to a user authenticating with PAM Authentication Plugin could lead to a server crash. Bug fixed #1521474. TokuDB status variables were missing from *Percona Server for MySQL* `5.7.10-1`. Bug fixed #1527364 (#923).

Attempting to rotate the audit log file would result in the audit log file name `foo.log.%u` (literally) instead of a numeric suffix. Bug fixed#1528603.

Adding an index to an InnoDB temporary table while `expand_fast_index_creation` was enabled could lead to server assertion. Bug fixed #1529555.

TokuDB would not be upgraded on *Debian*/*Ubuntu* distributions while performing an upgrade from *Percona Server for MySQL* 5.6 to *Percona Server for MySQL* 5.7 even if explicitly requested. Bug fixed #1533580.

The server would assert when both TokuDB and InnoDB tables were used within one transaction on a replication slave which has binary log enabled and slave updates logging disabled. Bug fixed #1534249 (upstream bug #80053).

MeCab Full-Text Parser Plugin has not been included in the previous release. Bug fixed #1534617.

Fixed server assertion caused by `Performance Schema` memory key mix-up in `SET STATEMENT ... FOR ...` statements. Bug fixed #1534874.

The service name on *CentOS* 6 has been renamed from `mysqld` back to `mysql`.This change requires a manual service restart after being upgraded from *Percona Server for MySQL* `5.7.10-1`

Bug fixed #1542332. Setting the `innodb_sched_priority_purge` (available only in debug builds) while purge threads were stopped would cause a server crash. Bug fixed #1368552.

Enabling TokuDB with the `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 been changed to a warning that a check is impossible. Bug fixed #1520890.

Write-heavy workload with a small buffer pool could lead to a deadlock when free buffers are exhausted. Bug fixed #1521905.

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).

Percona Server for MySQL 5.7 couldn't be restarted after TokuDB has been installed with the `ps_tokudb_admin` script. Bug fixed #1527535.

Fixed memory leak when `utility_user` is enabled. Bug fixed #1530918.

Page cleaner worker threads were not instrumented for `Performance Schema`. Bug fixed #1532747 (upstream bug #79894).

The busy server was preferring LRU flushing over flush list flushing too strongly which could lead to performance degradation. Bug fixed #1534114.

`libjemalloc.so.1` was missing from the binary tarball. Bug fixed #1537129.

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.

Added proper memory cleanup if for some reason a 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.

The variable `tokudb_support_xa` has been modified to prevent setting it to anything but `ON` / `ENABLED` and to print a SQL warning anytime an attempt is made to change it, just like `innodb_support_xa`. Bug fixed #928.

Other bugs fixed: #1179451, #1534246, #1524763, #1525109 (upstream #79569), #1530102, #897, #898, #899, #900, #901, #902, #903, #905, #906, #907, #908, #909, #910, #911, #912, #913, #915, #919, and #904.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

16.49 Percona Server for MySQL 5.7.10-1 (2015-12-14)

Percona is glad to announce the first Release Candidate release of *Percona Server for MySQL* 5.7.10-1 on December 14th, 2015 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.7.10, including all the bug fixes in it, *Percona Server for MySQL* 5.7.10-1 is the current Release Candidate release in the *Percona Server for MySQL* 5.7 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.7.10-1 milestone at Launchpad

This release contains all the bug fixes from the latest *Percona Server for MySQL* 5.6 release (currently *Percona Server for MySQL* 5.6.27-76.0).

16.49.1 New Features

- Percona Server for MySQL *5.7.10-1 is not available on either the* RHEL *5 family of Linux distributions or* Debian* 6 (squeeze).

A complete list of the changes between *Percona Server for MySQL* 5.6 and 5.7 can be found in Changed in Percona Server 5.7.

16.49.2 Known issues

MeCab Full-Text Parser Plugin has not been included in this release.

PAM Authentication Plugin currently isn't working correctly.

The following variables do not work correctly:

- `innodb_show_verbose_locks`
- `innodb_show_locks_held`

In *Percona Server for MySQL* 5.7 `super_read_only` feature has been replaced with upstream implementation. There are currently two known issues compared to *Percona Server for MySQL* 5.6 implementation:

- Bug #78963: If the `relay_log_info_repository` = TABLE, using `super_read_only` aborts the `STOP SLAVE` and could lead to a server crash in debug builds.
- Bug #79328, passing `super_read_only` as a server option has no effect.

Using a primary key with a `BLOB` in the TokuDB table could lead to a server crash (#916).

Using XA transactions with TokuDB could lead to a server crash(#900).

Percona TokuBackup has not been included in this release.

16.49.3 Bugs Fixed

Running `ALTER TABLE` without specifying the storage engine (without `ENGINE=` clause) or `OPTIMIZE TABLE` when `enforce_storage_engine` was enabled could lead to unrequested and unexpected storage engine changes. If done for a system table, it would circumvent regular system table storage engine compatibility checks, resulting in crashes or otherwise broken server operation. Bug fixed #1488055.

Some transaction deadlocks did not increase the `INFORMATION_SCHEMA.INNODB_METRICS` `lock_deadlocks` counter. Bug fixed #1466414 (upstream #77399).

Removed excessive locking during the buffer pool resize when checking whether AHI is enabled. Bug fixed #1525215 (upstream #78894).

Removed unnecessary code in the InnoDB error monitor thread. Bug fixed #1521564 (upstream #79477).

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.

Other bugs fixed: #371752 (upstream #45379), #1441362 (upstream #56155), #1385062 (upstream #74810), #1519201 (upstream #79391), #1515602, #1506697 (upstream #57552), #1501089 (upstream #75239), #1447527 (upstream #75368), #1384658 (upstream #74619), #1384656 (upstream #74584), and #1192052.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

17. References

Download PDF

17.1 List of upstream MySQL bugs fixed in Percona Server for MySQL 5.7

| * **Upstream Bug**

```
[#93917](http://bugs.mysql.com/bug.php?id=93917) - Wrong binlog entry for BLOB on a
blackhole intermediary master
```

- **JIRA bug**

  #5353

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.26-29

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#93708](http://bugs.mysql.com/bug.php?id=93708) - Page Cleaner will sleep for long time if
clock changes
```

- **JIRA bug**

  #5221
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.26-29
- **Upstream Fix**

  N/A

```
|
```

| * **Upstream Bug**

#92850 - Bad select+order by+limit performance in 5.7

- **JIRA bug**

  #4949
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.25-28
- **Upstream Fix**

  N/A

```
                |
```

| * **Upstream Bug**

#92809 - Inconsistent ResultSet for different Execution Plans

- **JIRA bug**

  #4907
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.25-28
- **Upstream Fix**

  5.7.27

```
                                |
```

| * **Upstream Bug**

#92108 - Deadlock by concurrent show binlogs, pfs session_variables table ...

- **JIRA bug**

  #4716
- **Upstream State**

  Closed
- **Fix Released**

Percona Server for MySQL 5.7.25-28

- **Upstream Fix**

  5.7.22

```
                |
```

| * **Upstream Bug**

#91541 - Flush status statement adds twice to global values

- **JIRA bug**

  #4570

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server 5.7.23-23

- **Upstream Fix**

  5.7.26

```
                                    |
```

| * **Upstream Bug**

#91423 - Can't run mysql on Ubuntu systems with long recovery time

- **JIRA bug**

  #4546

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server 5.7.23-23

- **Upstream Fix**

  N/A

```
                |
```

| * **Upstream Bug**

#91091 - A simple SELECT on a table with CHARSET=euckr COLLATE=euckr_bin ...

- **JIRA bug**

  #4513

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server 5.7.23-23

- **Upstream Fix**

  5.7.24

| | * **Upstream Bug**

#90264 - Some file operations in mf_iocache2.c are not instrumented

- **JIRA bug**

  #3937
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server 5.7.21-21
- **Upstream Fix**

  N/A

```
                                                                                |
```

| * **Upstream Bug**

#90238 - Comparison of uninitailized memory in log_in_use

- **JIRA bug**

  #3925
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server 5.7.21-21
- **Upstream Fix**

  N/A

```
                                                                                |
```

| * **Upstream Bug**

#89916 - hp_test1/hp_test2 are not built unless WITH_EMBEDDED_SERVER is defined

- **JIRA bug**

  #3845
- **Upstream State**

  Won't fix
- **Fix Released**

  Percona Server 5.7.21-21
- **Upstream Fix**

  N/A

```
                                |
```

| * **Upstream Bug**

#89822 - InnoDB retries open on EINTR error only if innodb_use_native_aio is ...

- **JIRA bug**

  #3843
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

Percona Server 5.7.21-21

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

#89646 - Clang warnings in 5.7.21

- **JIRA bug**

  #3814

- **Upstream State**

  Won't fix

- **Fix Released**

  Percona Server 5.7.21-21

- **Upstream Fix**

  N/A

```
                                                                                |
```

| * **Upstream Bug**

#89598 - plugin_mecab.cc:54:19: warning: unused variable 'bundle_mecab'

- **JIRA bug**

  #3804

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server 5.7.21-20

- **Upstream Fix**

  N/A

```
                                    |
```

| * **Upstream Bug**

#89422 - Dangerous enum-ulong casts in sql_formatter_options

- **JIRA bug**

  #3780

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server 5.7.21-20

- **Upstream Fix**

  N/A

```
                          |
```

| * **Upstream Bug**

#89421 - Missing mutex_unlock in Slave_reporting_capability::va_report

- **JIRA bug**

  #3780
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server 5.7.21-20
- **Upstream Fix**

  N/A

```
                                                                                |
```

| * **Upstream Bug**

#89420 - Enforcing C++03 mode in non debug builds

- **JIRA bug**

  #3780
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server 5.7.21-20
- **Upstream Fix**

  N/A

```
                                                                                |
```

| * **Upstream Bug**

#89205 - gap locks on READ COMMITTED cause by page split

- **JIRA bug**

  #1130
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server 5.7.22-22
- **Upstream Fix**

  5.7.20

```
                                                                                |
```

| * **Upstream Bug**

#88720 - Inconsistent and unsafe FLUSH behavior in terms of replication

- **JIRA bug**

  #1827
- **Upstream State**

  Verified (checked on 2019-02-11)
- **Fix Released**

Percona Server for MySQL 5.7.25-28

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#88057](http://bugs.mysql.com/bug.php?id=88057) - Intermediary slave does not log master
changes with…
```

- **JIRA bug**

  #1119
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server 5.7.20-19
- **Upstream Fix**

  N/A

```
                    |
```

| * **Upstream Bug**

#87065 - Release lock on table statistics after query plan created

- **JIRA bug**

  #2503
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server 5.7.20-18
- **Upstream Fix**

  N/A

```
                |
```

| * **Upstream Bug**

#86260 - Assert on KILL'ing a stored routine invocation

- **JIRA bug**

  #1091
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.18-16
- **Upstream Fix**

  5.7.22

```
                                  |
```

| * **Upstream Bug**

#86209 - audit plugin + MB collation connection + PREPARE stmt parse error crash...

- **JIRA bug**

  #1089
- **Upstream State**

  N/A
- **Fix Released**

Percona Server for MySQL 5.7.18-14

- **Upstream Fix**

  N/A

```
                    |
```

| * **Upstream Bug**

  #86164 - Fulltext search can not find word which contains punctuation marks

- **JIRA bug**

  #2501

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server 5.7.21-20

- **Upstream Fix**

  N/A

```
  |
```

| * **Upstream Bug**

  #86016 - Make MTR show core dump stacktraces from unit tests too

- **JIRA bug**

  #2499

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.18-16

- **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**

  Percona Server for MySQL 5.7.18-14

- **Upstream Fix**

  N/A

```
                      |
```

| * **Upstream Bug**

#85835 - server crash n-gram full text searching

- **JIRA bug**

  #237

- **Upstream State**

  N/A

- **Fix Released**

  Percona Server for MySQL 5.7.18-15

- **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**

  Percona Server for MySQL 5.7.18-14

- **Upstream Fix**

  5.7.19

|

| * **Upstream Bug**

#85671 - segfault-t failing under recent AddressSanitizer

- **JIRA bug**

  #2252

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server for MySQL 5.7.18-14

- **Upstream Fix**

  N/A

|

| * **Upstream Bug**

#85258 - DROP TEMPORARY TABLE creates a transaction in binary log on read only...

- **JIRA bug**

  #1785

- **Upstream State**

Closed

- **Fix Released**

  Percona Server for MySQL 5.7.18-14

- **Upstream Fix**

  N/A

```
                |
```

| * **Upstream Bug**

#85158 - heartbeats/fakerotate cause a forced sync_master_info

- **JIRA bug**

  #1812

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server 5.7.20-19

- **Upstream Fix**

  5.7.26

```
                                |
```

| * **Upstream Bug**

#85141 - Write/fsync amplification w/ duplicate GTIDs

- **JIRA bug**

  #1786

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.18-14

- **Upstream Fix**

  N/A

```
                |
```

| * **Upstream Bug**

#84958 - InnoDB's MVCC has O(N^2) behaviors

- **JIRA bug**

  #4712

- **JIRA bug**

  #5450

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server for MySQL 5.7.26-29

- **Upstream Fix**

  N/A

  |

  | * **Upstream Bug**

  #84736 - 5.7 range optimizer crash
- **JIRA bug**

  #1055
- **Upstream State**

  N/A
- **Fix Released**

  Percona Server for MySQL 5.7.17-12
- **Upstream Fix**

  N/A

  |

  | * **Upstream Bug**

  #84437 - super-read-only does not allow FLUSH LOGS on 5.7
- **JIRA bug**

  #1772
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.17-12
- **Upstream Fix**

  5.7.18

  |

  | * **Upstream Bug**

  #84420 - stopwords and ngram indexes
- **JIRA bug**

  #1802
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server 5.7.20-18
- **Upstream Fix**

  N/A

  |

  | * **Upstream Bug**

#84415 - slave don't report Seconds_Behind_Master when running ...

- **JIRA bug**

  #1770

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server for MySQL 5.7.18-14

- **Upstream Fix**

  5.7.22

```
                                |
```

| * **Upstream Bug**

#84366 - InnoDB index dives do not detect concurrent tree changes, return bogus...

- **JIRA bug**

  #1089

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.17-11

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#84350](http://bugs.mysql.com/bug.php?id=84350) - Error 1290 executing flush logs in read-
only slave
```

- **JIRA bug**

  #1044
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.17-12
- **Upstream Fix**

  5.7.18

|

| * **Upstream Bug**

#83814 - Add support for OpenSSL 1.1
- **JIRA bug**

  #1105
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.18-16
- **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**

  Percona Server for MySQL 5.7.17-12
- **Upstream Fix**

  N/A

|

| * **Upstream Bug**

#83232 - replication breaks after bug #74145 happens in master
- **JIRA bug**

  #1017
- **Upstream State**

  Closed
- **Fix Released**

Percona Server 5.7.24-26

- **Upstream Fix**

  N/A

|

| * **Upstream Bug**

  #83124 - Bug 81657 fix merge to 5.6 broken

- **JIRA bug**

  #1750

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server for MySQL 5.7.15-9

- **Upstream Fix**

  5.7.17

|

| * **Upstream Bug**

  #83073 - GCC 5 and 6 miscompile mach_parse_compressed

- **JIRA bug**

  #1745

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server for MySQL 5.7.15-9

- **Upstream Fix**

  5.7.17

|

| * **Upstream Bug**

  #83003 - Using temporary tables on slaves increases GTID sequence number

- **JIRA bug**

  #964

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server for MySQL 5.7.17-11

- **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**

  Percona Server for MySQL 5.7.15-9

- **Upstream Fix**

  5.7.20

```
                                |
```

| * **Upstream Bug**

#82940 - mysqld crashes itself when creating index

- **JIRA bug**

  #3410

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.26-29

- **Upstream Fix**

  N/A

```
                                  |
```

| * **Upstream Bug**

#82935 - Cipher ECDHE-RSA-AES128-GCM-SHA256 listed in man/Ssl_cipher_list, not...

- **JIRA bug**

  #1737

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.15-9

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#82886](http://bugs.mysql.com/bug.php?id=82886) - Server may crash due to a glibc bug in
handling short-lived detached …
```

- **JIRA bug**

  #1006
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.15-9
- **Upstream Fix**

  5.7.16

```
                        |
```

| * **Upstream Bug**

#82307 - Memory leaks in unit tests

- **JIRA bug**

  #2157
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.14-7
- **Upstream Fix**

  5.7.18

```
                                                                |
```

| * **Upstream Bug**

#82283 - main.mysqlbinlog_debug fails with a LeakSanitizer error

- **JIRA bug**

  #2156
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.14-7
- **Upstream Fix**

  5.7.19

```
                                    |
```

| * **Upstream Bug**

#82026 - Stack buffer overflow with –ssl-cipher=

- **JIRA bug**

  #2155
- **Upstream State**

  Closed
- **Fix Released**

Percona Server for MySQL 5.7.14-7

- **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**

  Percona Server for MySQL 5.7.14-7

- **Upstream Fix**

  5.7.15

```
                                |
```

| * **Upstream Bug**

#81814 - InnoDB adaptive hash index uses a bad partitioning algorithm for the ...

- **JIRA bug**

  #2498

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.18-14

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#81810](http://bugs.mysql.com/bug.php?id=81810) - Inconsistent sort order for blob/text
between InnoDB and filesort
```

- **JIRA bug**

  #1799
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.18-14
- **Upstream Fix**

  N/A

```
                                |
```

| * **Upstream Bug**

#81714 - mysqldump get_view_structure does not free MYSQL_RES in one error path

- **JIRA bug**

  #2152
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.13-6
- **Upstream Fix**

  5.7.20

```
                                |
```

| * **Upstream Bug**

#81675 - mysqlbinlog does not free the existing connection before opening new ...

- **JIRA bug**

  #1718
- **Upstream State**

  Closed
- **Fix Released**

  5.7.12-6
- **Upstream Fix**

  5.7.15

```
                                                |
```

| * **Upstream Bug**

#81657 - DBUG_PRINT in THD::decide_logging_format prints incorrectly, access ...

- **JIRA bug**

  #2150
- **Upstream State**

  Closed
- **Fix Released**

5.7.12-6

- **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 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.17-12

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#80962](http://bugs.mysql.com/bug.php?id=80962) - Replication does not work when
@@GLOBAL.SERVER_UUID is missing on the…
```

- **JIRA bug**

  #1684
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.12-5
- **Upstream Fix**

  5.7.13

```
                                |
```

| * **Upstream Bug**

#80607 - main.log_tables-big unstable on loaded hosts

- **JIRA bug**

  #2141
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.11-4
- **Upstream Fix**

  5.7.18

```
                                                |
```

| * **Upstream Bug**

#80606 - my_write, my_pwrite no longer safe to call from THD-less server utility...

- **JIRA bug**

  #970
- **Upstream State**

  N/A
- **Fix Released**

  Percona Server for MySQL 5.7.11-4
- **Upstream Fix**

  N/A

```
                                |
```

| * **Upstream Bug**

#80496 - buf_dblwr_init_or_load_pages now returns an error code, but caller not...

- **JIRA bug**

  #3384
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

Percona Server for MySQL 5.7.11-4

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#80288](http://bugs.mysql.com/bug.php?id=80288) - missing innodb_numa_interleave
```

- **JIRA bug**

  #974

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server for MySQL 5.7.12-5

- **Upstream Fix**

  5.7.16

```
                                                                 |
```

| * **Upstream Bug**

#80053 - Assertion in binlog coordinator on slave with 2 2pc handler log_slave ...

- **JIRA bug**

  #3361

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-2

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#79894](http://bugs.mysql.com/bug.php?id=79894) - Page cleaner worker threads are not
instrumented for performance schema
```

- **JIRA bug**

  #3356

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-2

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#79703](http://bugs.mysql.com/bug.php?id=79703) - Spin rounds per wait will be negative in
InnoDB status if spin waits >…
```

- **JIRA bug**

  #1684
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.10-2
- **Upstream Fix**

  N/A

```
                                |
```

| * **Upstream Bug**

#79610 - Failed DROP DATABASE due FK constraint on master breaks slave

- **JIRA bug**

  #1683
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.14-7
- **Upstream Fix**

  N/A

```
                                        |
```

| * **Upstream Bug**

#79569 - Some –big-test tests were forgotten to update in 5.7.10

- **JIRA bug**

  #3339
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.10-2
- **Upstream Fix**

  5.7.11

```
                                        |
```

| * **Upstream Bug**

#79117 - “change_user” command should be aware of preceding “error” command

- **JIRA bug**

  #659
- **Upstream State**

  Closed
- **Fix Released**

Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  5.7.12

```
                                |
```

| * **Upstream Bug**

#78894 - buf_pool_resize can lock less in checking whether AHI is on or off

- **JIRA bug**

  #3340

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

```
  |
```

| * **Upstream Bug**

#77684 - DROP TABLE IF EXISTS may brake replication if slave has replication ...

- **JIRA bug**

  #1639

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  5.7.12

```
                                |
```

| * **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 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#77399](http://bugs.mysql.com/bug.php?id=77399) - Deadlocks missed by
INFORMATION_SCHEMA.INNODB_METRICS lock_deadlocks …
```

- **JIRA bug**

  #1635

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

  | | * **Upstream Bug**

  #76418 - Server crashes when querying partitioning table MySQL_5.7.14

- **JIRA bug**

  #1050

- **Upstream State**

  N/A

- **Fix Released**

  Percona Server for MySQL 5.7.18-15

- **Upstream Fix**

  N/A

  ```
                                  |
  ```

  | * **Upstream Bug**

  #76142 - InnoDB tablespace import fails when importing table w/ different data …

- **JIRA bug**

  #1697

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.13-6

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#75534](http://bugs.mysql.com/bug.php?id=75534) - Solve buffer pool mutex contention by
splitting it
```

- **JIRA bug**

  Improved Buffer Pool Scalability
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

```
                    |
```

| * **Upstream Bug**

#75504 - btr_search_guess_on_hash makes found block young twice?

- **JIRA bug**

  #2454
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **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**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

```
                                |
```

| * **Upstream Bug**

#75311 - Error for SSL cipher is unhelpful

- **JIRA bug**

  #1779
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

Percona Server for MySQL 5.7.17-12

- **Upstream Fix**

  N/A

```
                                                                                |
```

| * **Upstream Bug**

#75189 - engines suite tests depending on InnoDB implementation details

- **JIRA bug**

  #2103

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

```
                |
```

| * **Upstream Bug**

#74637 - make dirty page flushing more adaptive

- **JIRA bug**

  Multi-threaded asynchronous LRU flusher

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-3

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#73418](http://bugs.mysql.com/bug.php?id=73418) - Add –manual-lldb option to mysql-test-
run.pl
```

- **JIRA bug**

  #2448
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

```
                                |
```

| * **Upstream Bug**

#72615 - MTR –mysqld=–default-storage-engine=foo incompatible w/ dynamically…

- **JIRA bug**

  #2071
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

| | * **Upstream Bug**

#72475 - Binlog events with binlog_format=MIXED are unconditionally logged in …

- **JIRA bug**

  #151
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **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 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.12-5
- **Upstream Fix**

N/A

```
                |
```

| * **Upstream Bug**

#72123 - Spurious lock_wait_timeout_thread wakeup in lock_wait_suspend_thread()

- **JIRA bug**

  #2504

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.18-16

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#72108](http://bugs.mysql.com/bug.php?id=72108) - Hard to read history file
```

- **JIRA bug**

  #2066
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **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 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.15-9
- **Upstream Fix**

  N/A

| | * **Upstream Bug**

#71759 - memory leak with string thread variable that set memalloc flag

- **JIRA bug**

  #1004
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.15-9
- **Upstream Fix**

  N/A

```
|
```

| * **Upstream Bug**

#71411 - buf_flush_LRU() does not return correct number in case of compressed ...

- **JIRA bug**

  #1461
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **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 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **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 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

#71183 - os_file_fsync() should handle fsync() returning EINTR

- **JIRA bug**

  #1461

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

```
                    |
```

| * **Upstream Bug**

#71091 - CSV engine does not properly process “”, in quotes

- **JIRA bug**

  #153

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **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 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

  | | * **Upstream Bug**

  #70490 - Suppression is too strict on some systems
- **JIRA bug**

  #2038
- **Upstream State**

  Closed
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  5.7.20

  |

  | * **Upstream Bug**

  #69991 - MySQL client is broken without readline
- **JIRA bug**

  #1467
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **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**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

```
                                                                |
```

| * **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**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

```
                                  |
```

| * **Upstream Bug**

#69232 - buf_dblwr->mutex can be splited into two

- **JIRA bug**

  Parallel doublewrite buffer

- **Upstream State**

  No Feedback (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.11-4

- **Upstream Fix**

  N/A

```
   |
```

| * **Upstream Bug**

#69170 - buf_flush_LRU is lazy

- **JIRA bug**

  #2430

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

  |

  | * **Upstream Bug**

  #69146 - Needless log flush order mutex acquisition in buf_pool_get_oldest_mod...
- **JIRA bug**

  #2418
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

  | | * **Upstream Bug**

  #68714 - Remove literal statement digest values from perfschema tests
- **JIRA bug**

  #1340
- **Upstream State**

  Not a bug
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

  |

  | * **Upstream Bug**

  #68481 - InnoDB LRU flushing for MySQL 5.6 needs work
- **JIRA bug**

  #2432
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

  |

  | * **Upstream Bug**

  #68052 - SSL Certificate Subject ALT Names with IPs not respected with –ssl-ver...
- **JIRA bug**

#1076

- **Upstream State**

  Closed

- **Fix Released**

  Percona Server for MySQL 5.7.18-16

- **Upstream Fix**

  N/A

```
                                |
```

| * **Upstream Bug**

#67808 - in innodb engine, double write and multi-buffer pool instance reduce ...

- **JIRA bug**

  Parallel doublewrite buffer

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.11-4

- **Upstream Fix**

  N/A

| | * **Upstream Bug**

```
[#63130](http://bugs.mysql.com/bug.php?id=63130) - CMake-based check for the presence of a
system readline library is not…
```

- **JIRA bug**

  #1467
- **Upstream State**

  Can't Repeat
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

```
                                                                                |
```

| * **Upstream Bug**

#57583 - fast index create not used during "alter table foo engine=innodb"

- **JIRA bug**

  #2113
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

```
                                |
```

| * **Upstream Bug**

#53645 - SHOW GRANTS not displaying all the applicable grants

- **JIRA bug**

  #191
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

  Percona Server for MySQL 5.7.10-1
- **Upstream Fix**

  N/A

```
                                                                                |
```

| * **Upstream Bug**

#53588 - Blackhole : Specified key was too long; max key length is 1000 bytes

- **JIRA bug**

  #1126
- **Upstream State**

  Verified (checked on 2019-05-21)
- **Fix Released**

Percona Server 5.7.20-19

- **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 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **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 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **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 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

|

| * **Upstream Bug**

#35125 - Allow the ability to set the server_id for a connection for logging to...

- **Launchpad BP**

  Blueprint

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

```
        |
```

| * **Upstream Bug**

#25007 - memory tables with dynamic rows format

- **JIRA bug**

  #2407

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

```
                                                |
```

| * **Upstream Bug**

#20001 - Support for temp-tables in INFORMATION_SCHEMA

- **JIRA bug**

  Temporary tables

- **Upstream State**

  Verified (checked on 2019-05-21)

- **Fix Released**

  Percona Server for MySQL 5.7.10-1

- **Upstream Fix**

  N/A

```
                                    |
```

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

17.2 List of variables introduced in Percona Server 5.7

17.2.1 System Variables

Name	Cmd-Line	Option File	Var Scope	Dynamic	Status
audit_log_buffer_size	Yes	Yes	Global	No	
audit_log_file	Yes	Yes	Global	No	
audit_log_flush	Yes	Yes	Global	Yes	
audit_log_format	Yes	Yes	Global	No	
audit_log_handler	Yes	Yes	Global	No	
audit_log_policy	Yes	Yes	Global	Yes	
audit_log_rotate_on_size	Yes	Yes	Global	No	
audit_log_rotations	Yes	Yes	Global	No	
audit_log_strategy	Yes	Yes	Global	No	
audit_log_syslog_facility	Yes	Yes	Global	No	
audit_log_syslog_ident	Yes	Yes	Global	No	
audit_log_syslog_priority	Yes	Yes	Global	No	
binlog_space_limit	Yes	Yes	Global	Yes	
csv_mode	Yes	Yes	Both	Yes	
enforce_storage_engine	Yes	Yes	Global	No	
expand_fast_index_creation	Yes	No	Both	Yes	
extra_max_connections	Yes	Yes	Global	Yes	
extra_port	Yes	Yes	Global	No	
ft_query_extra_word_chars	Yes	Yes	Both	Yes	
have_backup_locks	Yes	No	Global	No	
have_backup_safe_binlog_info	Yes	No	Global	No	
have_snapshot_cloning	Yes	No	Global	No	
innodb_background_scrub_data_compressed	Yes	Yes	Global	Yes	
innodb_background_scrub_data_uncompressed	Yes	Yes	Global	Yes	
innodb_cleaner_lsn_age_factor	Yes	Yes	Global	Yes	
innodb_corrupt_table_action	Yes	Yes	Global	Yes	
innodb_default_encryption_key_id	Yes	Yes	Session	Yes	
innodb_empty_free_list_algorithm	Yes	Yes	Global	Yes	
innodb_encrypt_online_alter_logs	Yes	Yes	Global	Yes	
innodb_encrypt_tables	Yes	Yes	Global	Yes	
innodb_kill_idle_transaction	Yes	Yes	Global	Yes	
innodb_max_bitmap_file_size	Yes	Yes	Global	Yes	
innodb_max_changed_pages	Yes	Yes	Global	Yes	
innodb_online_encryption_rotate_key_age	Yes	Yes	Global	Yes	Deprecated

Name	Cmd-Line	Option File	Var Scope	Dynamic	Status
innodb_online_encryption_threads	Yes	Yes	Global	Yes	Deprecated
innodb_parallel_dblwr_encrypt	Yes	Yes	Global	Yes	Deprecated 5.7.31-34
innodb_print_lock_wait_timeout_info	Yes	Yes	Global	Yes	
innodb_redo_log_encrypt	Yes	Yes	Global	Yes	Deprecated 5.7.31-34
innodb_scrub_log	Yes	Yes	Global	Yes	
innodb_scrub_log_speed	Yes	Yes	Global	Yes	
innodb_show_locks_held	Yes	Yes	Global	Yes	
innodb_show_verbose_locks	Yes	Yes	Global	Yes	
innodb_sys_tablespace_encrypt	Yes	Yes	Global	No	Deprecated
innodb_temp_tablespace_encrypt	Yes	Yes	Global	No	Deprecated
innodb_track_changed_pages	Yes	Yes	Global	No	
innodb_undo_log_encrypt	Yes	Yes	Global	Yes	Deprecated
innodb_use_global_flush_log_at_trx_commit	Yes	Yes	Global	Yes	
keyring_vault_config	Yes	Yes	Global	Yes	
keyring_vault_timeout	Yes	Yes	Global	Yes	
log_slow_filter	Yes	Yes	Both	Yes	
log_slow_rate_limit	Yes	Yes	Both	Yes	
log_slow_rate_type	Yes	Yes	Global	Yes	
log_slow_sp_statements	Yes	Yes	Global	Yes	
log_slow_verbosity	Yes	Yes	Both	Yes	
log_warnings_suppress	Yes	Yes	Global	Yes	
max_binlog_files	Yes	Yes	Global	Yes	
max_slowlog_files	Yes	Yes	Global	Yes	
max_slowlog_size	Yes	Yes	Global	Yes	
proxy_protocol_networks	Yes	Yes	Global	No	
pseudo_server_id	Yes	No	Session	Yes	
query_cache_strip_comments	Yes	Yes	Global	Yes	
query_response_time_flush	Yes	No	Global	No	
query_response_time_range_base	Yes	Yes	Global	Yes	
query_response_time_stats	Yes	Yes	Global	Yes	
slow_query_log_always_write_time	Yes	Yes	Global	Yes	
slow_query_log_use_global_control	Yes	Yes	Global	Yes	
thread_pool_high_prio_mode	Yes	Yes	Both	Yes	

Name	Cmd-Line	Option File	Var Scope	Dynamic	Status
thread_pool_high_prio_tickets	Yes	Yes	Both	Yes	
thread_pool_idle_timeout	Yes	Yes	Global	Yes	
thread_pool_max_threads	Yes	Yes	Global	Yes	
thread_pool_oversubscribe	Yes	Yes	Global	Yes	
thread_pool_size	Yes	Yes	Global	Yes	
thread_pool_stall_limit	Yes	Yes	Global	No	
thread_statistics	Yes	Yes	Global	Yes	
tokudb_alter_print_error					
tokudb_analyze_delete_fraction					
tokudb_analyze_in_background	Yes	Yes	Both	Yes	
tokudb_analyze_mode	Yes	Yes	Both	Yes	
tokudb_analyze_throttle	Yes	Yes	Both	Yes	
tokudb_analyze_time	Yes	Yes	Both	Yes	
tokudb_auto_analyze	Yes	Yes	Both	Yes	
tokudb_block_size					
tokudb_bulk_fetch					
tokudb_cache_size					
tokudb_cachetable_pool_threads	Yes	Yes	Global	No	
tokudb_cardinality_scale_percent					
tokudb_check_jemalloc					
tokudb_checkpoint_lock					
tokudb_checkpoint_on_flush_logs					
tokudb_checkpoint_pool_threads	Yes	Yes	Global	No	
tokudb_checkpointing_period					
tokudb_cleaner_iterations					
tokudb_cleaner_period					
tokudb_client_pool_threads	Yes	Yes	Global	No	
tokudb_commit_sync					
tokudb_compress_buffers_before_eviction	Yes	Yes	Global	No	
tokudb_create_index_online					
tokudb_data_dir					
tokudb_debug					
tokudb_directio					
tokudb_disable_hot_alter					

Name	Cmd-Line	Option File	Var Scope	Dynamic	Status
tokudb_disable_prefetching					
tokudb_disable_slow_alter					
tokudb_empty_scan					
tokudb_enable_partial_eviction	Yes	Yes	Global	No	
tokudb_fanout	Yes	Yes	Both	Yes	
tokudb_fs_reserve_percent					
tokudb_fsync_log_period					
tokudb_hide_default_row_format					
tokudb_killed_time					
tokudb_last_lock_timeout					
tokudb_load_save_space					
tokudb_loader_memory_size					
tokudb_lock_timeout					
tokudb_lock_timeout_debug					
tokudb_log_dir					
tokudb_max_lock_memory					
tokudb_optimize_index_fraction					
tokudb_optimize_index_name					
tokudb_optimize_throttle					
tokudb_pk_insert_mode					
tokudb_prelock_empty					
tokudb_read_block_size					
tokudb_read_buf_size					
tokudb_read_status_frequency					
:ref:`tokudb_row_formatref					
tokudb_rpl_check_readonly					
tokudb_rpl_lookup_rows					
tokudb_rpl_lookup_rows_delay					
tokudb_rpl_unique_checks					
tokudb_rpl_unique_checks_delay					
tokudb_strip_frm_data	Yes	Yes	Global	No	
tokudb_support_xa					
tokudb_tmp_dir					
tokudb_version					

Name	Cmd-Line	Option File	Var Scope	Dynamic	Status
tokudb_write_status_frequency					
userstat	Yes	Yes	Global	Yes	
version_comment	Yes	Yes	Global	Yes	
version_suffix	Yes	Yes	Global	Yes	

17.2.2 Status variables

Name	Var Type	Var Scope
Binlog_snapshot_file	String	Global
Binlog_snapshot_position	Numeric	Global
Com_lock_binlog_for_backup	Numeric	Both
Com_lock_tables_for_backup	Numeric	Both
Com_show_client_statistics	Numeric	Both
Com_show_index_statistics	Numeric	Both
Com_show_table_statistics	Numeric	Both
Com_show_thread_statistics	Numeric	Both
Com_show_user_statistics	Numeric	Both
Com_unlock_binlog	Numeric	Both
Innodb_background_log_sync	Numeric	Global
Innodb_buffer_pool_pages_LRU_flushed	Numeric	Global
Innodb_buffer_pool_pages_made_not_young	Numeric	Global
Innodb_buffer_pool_pages_made_young	Numeric	Global
Innodb_buffer_pool_pages_old	Numeric	Global
Innodb_checkpoint_age	Numeric	Global
Innodb_checkpoint_max_age	Numeric	Global
Innodb_ibuf_free_list	Numeric	Global
Innodb_ibuf_segment_size	Numeric	Global
Innodb_lsn_current	Numeric	Global
Innodb_lsn_flushed	Numeric	Global
Innodb_lsn_last_checkpoint	Numeric	Global
Innodb_master_thread_active_loops	Numeric	Global
Innodb_master_thread_idle_loops	Numeric	Global
Innodb_max_trx_id	Numeric	Global
Innodb_mem_adaptive_hash	Numeric	Global
Innodb_mem_dictionary	Numeric	Global
Innodb_oldest_view_low_limit_trx_id	Numeric	Global
Innodb_purge_trx_id	Numeric	Global
Innodb_purge_undo_no	Numeric	Global
Threadpool_idle_threads	Numeric	Global
Threadpool_threads	Numeric	Global
Tokudb_DB_OPENS		
Tokudb_DB_CLOSES		

Name	Var Type	Var Scope
Tokudb_DB_OPEN_CURRENT		
Tokudb_DB_OPEN_MAX		
Tokudb_LEAF_ENTRY_MAX_COMMITTED_XR		
Tokudb_LEAF_ENTRY_MAX_PROVISIONAL_XR		
Tokudb_LEAF_ENTRY_EXPANDED		
Tokudb_LEAF_ENTRY_MAX_MEMSIZE		
Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_IN		
Tokudb_LEAF_ENTRY_APPLY_GC_BYTES_OUT		
Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_IN		
Tokudb_LEAF_ENTRY_NORMAL_GC_BYTES_OUT		
Tokudb_CHECKPOINT_PERIOD		
Tokudb_CHECKPOINT_FOOTPRINT		
Tokudb_CHECKPOINT_LAST_BEGAN		
Tokudb_CHECKPOINT_LAST_COMPLETE_BEGAN		
Tokudb_CHECKPOINT_LAST_COMPLETE_ENDED		
Tokudb_CHECKPOINT_DURATION		
Tokudb_CHECKPOINT_DURATION_LAST		
Tokudb_CHECKPOINT_LAST_LSN		
Tokudb_CHECKPOINT_TAKEN		
Tokudb_CHECKPOINT_FAILED		
Tokudb_CHECKPOINT_WAITERS_NOW		
Tokudb_CHECKPOINT_WAITERS_MAX		
Tokudb_CHECKPOINT_CLIENT_WAIT_ON_MO		
Tokudb_CHECKPOINT_CLIENT_WAIT_ON_CS		
Tokudb_CHECKPOINT_BEGIN_TIME		
Tokudb_CHECKPOINT_LONG_BEGIN_TIME		
Tokudb_CHECKPOINT_LONG_BEGIN_COUNT		
Tokudb_CHECKPOINT_END_TIME		
Tokudb_CHECKPOINT_LONG_END_TIME		
Tokudb_CHECKPOINT_LONG_END_COUNT		
Tokudb_CACHETABLE_MISS		
Tokudb_CACHETABLE_MISS_TIME		
Tokudb_CACHETABLE_PREFETCHES		
Tokudb_CACHETABLE_SIZE_CURRENT		

Name	Var Type	Var Scope
Tokudb_CACHETABLE_SIZE_LIMIT		
Tokudb_CACHETABLE_SIZE_WRITING		
Tokudb_CACHETABLE_SIZE_NONLEAF		
Tokudb_CACHETABLE_SIZE_LEAF		
Tokudb_CACHETABLE_SIZE_ROLLBACK		
Tokudb_CACHETABLE_SIZE_CACHEPRESSURE		
Tokudb_CACHETABLE_SIZE_CLONED		
Tokudb_CACHETABLE_EVICTIONS		
Tokudb_CACHETABLE_CLEANER_EXECUTIONS		
Tokudb_CACHETABLE_CLEANER_PERIOD		
Tokudb_CACHETABLE_CLEANER_ITERATIONS		
Tokudb_CACHETABLE_WAIT_PRESSURE_COUNT		
Tokudb_CACHETABLE_WAIT_PRESSURE_TIME		
Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_COUNT		
Tokudb_CACHETABLE_LONG_WAIT_PRESSURE_TIME		
Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS		
Tokudb_CACHETABLE_POOL_CLIENT_NUM_THREADS_ACTIVE		
Tokudb_CACHETABLE_POOL_CLIENT_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CLIENT_MAX_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_ITEMS_PROCESSED		
Tokudb_CACHETABLE_POOL_CLIENT_TOTAL_EXECUTION_TIME		
Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS		
Tokudb_CACHETABLE_POOL_CACHETABLE_NUM_THREADS_ACTIVE		
Tokudb_CACHETABLE_POOL_CACHETABLE_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CACHETABLE_MAX_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_ITEMS_PROCESSED		
Tokudb_CACHETABLE_POOL_CACHETABLE_TOTAL_EXECUTION_TIME		
Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS		
Tokudb_CACHETABLE_POOL_CHECKPOINT_NUM_THREADS_ACTIVE		
Tokudb_CACHETABLE_POOL_CHECKPOINT_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CHECKPOINT_MAX_QUEUE_SIZE		
Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_ITEMS_PROCESSED		
Tokudb_CACHETABLE_POOL_CHECKPOINT_TOTAL_EXECUTION_TIME		
Tokudb_LOCKTREE_MEMORY_SIZE		

Name	Var Type	Var Scope
Tokudb_LOCKTREE_MEMORY_SIZE_LIMIT		
Tokudb_LOCKTREE_ESCALATION_NUM		
Tokudb_LOCKTREE_ESCALATION_SECONDS		
Tokudb_LOCKTREE_LATEST_POST_ESCALATION_MEMORY_SIZE		
Tokudb_LOCKTREE_OPEN_CURRENT		
Tokudb_LOCKTREE_PENDING_LOCK_REQUESTS		
Tokudb_LOCKTREE_STO_ELIGIBLE_NUM		
Tokudb_LOCKTREE_STO_ENDED_NUM		
Tokudb_LOCKTREE_STO_ENDED_SECONDS		
Tokudb_LOCKTREE_WAIT_COUNT		
Tokudb_LOCKTREE_WAIT_TIME		
Tokudb_LOCKTREE_LONG_WAIT_COUNT		
Tokudb_LOCKTREE_LONG_WAIT_TIME		
Tokudb_LOCKTREE_TIMEOUT_COUNT		
Tokudb_LOCKTREE_WAIT_ESCALATION_COUNT		
Tokudb_LOCKTREE_WAIT_ESCALATION_TIME		
Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_COUNT		
Tokudb_LOCKTREE_LONG_WAIT_ESCALATION_TIME		
Tokudb_DICTIONARY_UPDATES		
Tokudb_DICTIONARY_BROADCAST_UPDATES		
Tokudb_DESCRIPTOR_SET		
Tokudb_MESSAGES_IGNORED_BY_LEAF_DUE_TO_MSN		
Tokudb_TOTAL_SEARCH_RETRIES		
Tokudb_SEARCH_TRIES_GT_HEIGHT		
Tokudb_SEARCH_TRIES_GT_HEIGHTPLUS3		
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT		
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_BYTES		
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_UNCOMPRESSED_BYTES		
Tokudb_LEAF_NODES_FLUSHED_NOT_CHECKPOINT_SECONDS		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_BYTES		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_UNCOMPRESSE		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_NOT_CHECKPOINT_SECONDS		
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT		

Name	Var Type	Var Scope
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_BYTES		
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_UNCOMPRESSED_BYTES		
Tokudb_LEAF_NODES_FLUSHED_CHECKPOINT_SECONDS		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_BYTES		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_UNCOMPRESSED_BY		
Tokudb_NONLEAF_NODES_FLUSHED_TO_DISK_CHECKPOINT_SECONDS		
Tokudb_LEAF_NODE_COMPRESSION_RATIO		
Tokudb_NONLEAF_NODE_COMPRESSION_RATIO		
Tokudb_OVERALL_NODE_COMPRESSION_RATIO		
Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS		
Tokudb_NONLEAF_NODE_PARTIAL_EVICTIONS_BYTES		
Tokudb_LEAF_NODE_PARTIAL_EVICTIONS		
Tokudb_LEAF_NODE_PARTIAL_EVICTIONS_BYTES		
Tokudb_LEAF_NODE_FULL_EVICTIONS		
Tokudb_LEAF_NODE_FULL_EVICTIONS_BYTES		
Tokudb_NONLEAF_NODE_FULL_EVICTIONS		
Tokudb_NONLEAF_NODE_FULL_EVICTIONS_BYTES		
Tokudb_LEAF_NODES_CREATED		
Tokudb_NONLEAF_NODES_CREATED		
Tokudb_LEAF_NODES_DESTROYED		
Tokudb_NONLEAF_NODES_DESTROYED		
Tokudb_MESSAGES_INJECTED_AT_ROOT_BYTES		
Tokudb_MESSAGES_FLUSHED_FROM_H1_TO_LEAVES_BYTES		
Tokudb_MESSAGES_IN_TREES_ESTIMATE_BYTES		
Tokudb_MESSAGES_INJECTED_AT_ROOT		
Tokudb_BROADCASE_MESSAGES_INJECTED_AT_ROOT		
Tokudb_BASEMENTS_DECOMPRESSED_TARGET_QUERY		
Tokudb_BASEMENTS_DECOMPRESSED_PRELOCKED_RANGE		
Tokudb_BASEMENTS_DECOMPRESSED_PREFETCH		
Tokudb_BASEMENTS_DECOMPRESSED_FOR_WRITE		
Tokudb_BUFFERS_DECOMPRESSED_TARGET_QUERY		
Tokudb_BUFFERS_DECOMPRESSED_PRELOCKED_RANGE		
Tokudb_BUFFERS_DECOMPRESSED_PREFETCH		

Name	Var Type	Var Scope
Tokudb_BUFFERS_DECOMPRESSED_FOR_WRITE		
Tokudb_PIVOTS_FETCHED_FOR_QUERY		
Tokudb_PIVOTS_FETCHED_FOR_QUERY_BYTES		
Tokudb_PIVOTS_FETCHED_FOR_QUERY_SECONDS		
Tokudb_PIVOTS_FETCHED_FOR_PREFETCH		
Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_BYTES		
Tokudb_PIVOTS_FETCHED_FOR_PREFETCH_SECONDS		
Tokudb_PIVOTS_FETCHED_FOR_WRITE		
Tokudb_PIVOTS_FETCHED_FOR_WRITE_BYTES		
Tokudb_PIVOTS_FETCHED_FOR_WRITE_SECONDS		
Tokudb_BASEMENTS_FETCHED_TARGET_QUERY		
Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_BYTES		
Tokudb_BASEMENTS_FETCHED_TARGET_QUERY_SECONDS		
Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE		
Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_BYTES		
Tokudb_BASEMENTS_FETCHED_PRELOCKED_RANGE_SECONDS		
Tokudb_BASEMENTS_FETCHED_PREFETCH		
Tokudb_BASEMENTS_FETCHED_PREFETCH_BYTES		
Tokudb_BASEMENTS_FETCHED_PREFETCH_SECONDS		
Tokudb_BASEMENTS_FETCHED_FOR_WRITE		
Tokudb_BASEMENTS_FETCHED_FOR_WRITE_BYTES		
Tokudb_BASEMENTS_FETCHED_FOR_WRITE_SECONDS		
Tokudb_BUFFERS_FETCHED_TARGET_QUERY		
Tokudb_BUFFERS_FETCHED_TARGET_QUERY_BYTES		
Tokudb_BUFFERS_FETCHED_TARGET_QUERY_SECONDS		
Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE		
Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_BYTES		
Tokudb_BUFFERS_FETCHED_PRELOCKED_RANGE_SECONDS		
Tokudb_BUFFERS_FETCHED_PREFETCH		
Tokudb_BUFFERS_FETCHED_PREFETCH_BYTES		
Tokudb_BUFFERS_FETCHED_PREFETCH_SECONDS		
Tokudb_BUFFERS_FETCHED_FOR_WRITE		
Tokudb_BUFFERS_FETCHED_FOR_WRITE_BYTES		
Tokudb_BUFFERS_FETCHED_FOR_WRITE_SECONDS		

Name	Var Type	Var Scope
Tokudb_LEAF_COMPRESSION_TO_MEMORY_SECONDS		
Tokudb_LEAF_SERIALIZATION_TO_MEMORY_SECONDS		
Tokudb_LEAF_DECOMPRESSION_TO_MEMORY_SECONDS		
Tokudb_LEAF_DESERIALIZATION_TO_MEMORY_SECONDS		
Tokudb_NONLEAF_COMPRESSION_TO_MEMORY_SECONDS		
Tokudb_NONLEAF_SERIALIZATION_TO_MEMORY_SECONDS		
Tokudb_NONLEAF_DECOMPRESSION_TO_MEMORY_SECONDS		
Tokudb_NONLEAF_DESERIALIZATION_TO_MEMORY_SECONDS		
Tokudb_PROMOTION_ROOTS_SPLIT		
Tokudb_PROMOTION_LEAF_ROOTS_INJECTED_INTO		
Tokudb_PROMOTION_H1_ROOTS_INJECTED_INTO		
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_0		
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_1		
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_2		
Tokudb_PROMOTION_INJECTIONS_AT_DEPTH_3		
Tokudb_PROMOTION_INJECTIONS_LOWER_THAN_DEPTH_3		
Tokudb_PROMOTION_STOPPED_NONEMPTY_BUFFER		
Tokudb_PROMOTION_STOPPED_AT_HEIGHT_1		
Tokudb_PROMOTION_STOPPED_CHILD_LOCKED_OR_NOT_IN_MEMORY		
Tokudb_PROMOTION_STOPPED_CHILD_NOT_FULLY_IN_MEMORY		
Tokudb_PROMOTION_STOPPED_AFTER_LOCKING_CHILD		
Tokudb_BASEMENT_DESERIALIZATION_FIXED_KEY		
Tokudb_BASEMENT_DESERIALIZATION_VARIABLE_KEY		
Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_SUCCESS		
Tokudb_PRO_RIGHTMOST_LEAF_SHORTCUT_FAIL_POS		
Tokudb_RIGHTMOST_LEAF_SHORTCUT_FAIL_REACTIVE		
Tokudb_CURSOR_SKIP_DELETED_LEAF_ENTRY		
Tokudb_FLUSHER_CLEANER_TOTAL_NODES		
Tokudb_FLUSHER_CLEANER_H1_NODES		
Tokudb_FLUSHER_CLEANER_HGT1_NODES		
Tokudb_FLUSHER_CLEANER_EMPTY_NODES		
Tokudb_FLUSHER_CLEANER_NODES_DIRTIED		
Tokudb_FLUSHER_CLEANER_MAX_BUFFER_SIZE		
Tokudb_FLUSHER_CLEANER_MIN_BUFFER_SIZE		

Name	Var Type	Var Scope
Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_SIZE		
Tokudb_FLUSHER_CLEANER_MAX_BUFFER_WORKDONE		
Tokudb_FLUSHER_CLEANER_MIN_BUFFER_WORKDONE		
Tokudb_FLUSHER_CLEANER_TOTAL_BUFFER_WORKDONE		
Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_STARTED		
Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_RUNNING		
Tokudb_FLUSHER_CLEANER_NUM_LEAF_MERGES_COMPLETED		
Tokudb_FLUSHER_CLEANER_NUM_DIRTIED_FOR_LEAF_MERGE		
Tokudb_FLUSHER_FLUSH_TOTAL		
Tokudb_FLUSHER_FLUSH_IN_MEMORY		
Tokudb_FLUSHER_FLUSH_NEEDED_IO		
Tokudb_FLUSHER_FLUSH_CASCADES		
Tokudb_FLUSHER_FLUSH_CASCADES_1		
Tokudb_FLUSHER_FLUSH_CASCADES_2		
Tokudb_FLUSHER_FLUSH_CASCADES_3		
Tokudb_FLUSHER_FLUSH_CASCADES_4		
Tokudb_FLUSHER_FLUSH_CASCADES_5		
Tokudb_FLUSHER_FLUSH_CASCADES_GT_5		
Tokudb_FLUSHER_SPLIT_LEAF		
Tokudb_FLUSHER_SPLIT_NONLEAF		
Tokudb_FLUSHER_MERGE_LEAF		
Tokudb_FLUSHER_MERGE_NONLEAF		
Tokudb_FLUSHER_BALANCE_LEAF		
Tokudb_HOT_NUM_STARTED		
Tokudb_HOT_NUM_COMPLETED		
Tokudb_HOT_NUM_ABORTED		
Tokudb_HOT_MAX_ROOT_FLUSH_COUNT		
Tokudb_TXN_BEGIN		
Tokudb_TXN_BEGIN_READ_ONLY		
Tokudb_TXN_COMMITS		
Tokudb_TXN_ABORTS		
Tokudb_LOGGER_NEXT_LSN		
Tokudb_LOGGER_WRITES		
Tokudb_LOGGER_WRITES_BYTES		

Name	Var Type	Var Scope
Tokudb_LOGGER_WRITES_UNCOMPRESSED_BYTES		
Tokudb_LOGGER_WRITES_SECONDS		
Tokudb_LOGGER_WAIT_LONG		
Tokudb_LOADER_NUM_CREATED		
Tokudb_LOADER_NUM_CURRENT		
Tokudb_LOADER_NUM_MAX		
Tokudb_MEMORY_MALLOC_COUNT		
Tokudb_MEMORY_FREE_COUNT		
Tokudb_MEMORY_REALLOC_COUNT		
Tokudb_MEMORY_MALLOC_FAIL		
Tokudb_MEMORY_REALLOC_FAIL		
Tokudb_MEMORY_REQUESTED		
Tokudb_MEMORY_USED		
Tokudb_MEMORY_FREED		
Tokudb_MEMORY_MAX_REQUESTED_SIZE		
Tokudb_MEMORY_LAST_FAILED_SIZE		
Tokudb_MEM_ESTIMATED_MAXIMUM_MEMORY_FOOTPRINT		
Tokudb_MEMORY_MALLOCATOR_VERSION		
Tokudb_MEMORY_MMAP_THRESHOLD		
Tokudb_FILESYSTEM_THREADS_BLOCKED_BY_FULL_DISK		
Tokudb_FILESYSTEM_FSYNC_TIME		
Tokudb_FILESYSTEM_FSYNC_NUM		
Tokudb_FILESYSTEM_LONG_FSYNC_TIME		
Tokudb_FILESYSTEM_LONG_FSYNC_NUM		

Last update: 2022-09-27

Download PDF

17.3 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.

17.3.1 Submitting Changes

We keep trunk in a constant state of stability to allow for a release at any time and to minimize wasted time by developers due to broken code.

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, 5.7 is a series in Percona XtraBackup and 5.7.15, 5.7.16 and 5.7.17 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 Git branches on which development occurs: 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.

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.

Percona Server for MySQL

Percona Server for MySQL has the same process, but we have different branches and merge requests.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

17.4 Telemetry on Percona Server for MySQL

Percona telemetry fills in the gaps in our understanding of how you use Percona Server for MySQL to improve our products. Participation in the anonymous program is optional. You can opt-out if you prefer to not share this information.

17.4.1 What information is collected

At this time, telemetry is added only to the Percona packages and Docker images. Percona Server for MySQL collects only information about the installation environment. Future releases may add additional metrics.

Be assured that access to this raw data is rigorously controlled. Percona does not collect personal data. All data is anonymous and cannot be traced to a specific user. To learn more about our privacy practices, read our Percona Privacy statement.

An example of the data collected is the following:

```
[{"id" : "c416c3ee-48cd-471c-9733-37c2886f8231",
"product_family" : "PRODUCT_FAMILY_PS",
"instanceId" : "6aef422e-56a7-4530-af9d-94cc02198343",
"createTime" : "2023-10-16T10:46:23Z",
"metrics":
[{"key" : "deployment","value" : "PACKAGE"},
{"key" : "pillar_version","value" : "5.7.44-48"},
{"key" : "OS","value" : "Oracle Linux Server 8.8"},
{"key" : "hardware_arch","value" : "x86_64 x86_64"}]}]
```

17.4.2 Disable telemetry

Telemetry is enabled by default. If you decide not to send usage data to Percona, you can set the `PERCONA_TELEMETRY_DISABLE=1` environment variable for either the root user or in the operating system prior to the installation process.

Debian-derived distribution

Add the environment variable before the install process.

```
{.bash data-prompt="$"}
$ sudo PERCONA_TELEMETRY_DISABLE=1 apt install percona-server-server-57
```

Red Hat-derived distribution

Add the environment variable before the install process.

```
{.bash data-prompt="$"}
$ sudo PERCONA_TELEMETRY_DISABLE=1 yum install percona-server-server-57
```

DOCKER

Add the environment variable when running a command in a new container.

```
{.bash data-prompt="$"}
$ docker run -d -e MYSQL_ROOT_PASSWORD=test1234# -e PERCONA_TELEMETRY_DISABLE=1 -e --name=percona-server
percona/percona-server:5.7.44-48
```

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-11-27

Download PDF

17.5 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.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-06-26

Download PDF

17.6 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.

- INFORMATION_SCHEMA.CLIENT_STATISTICS
- INFORMATION_SCHEMA.GLOBAL_TEMPORARY_TABLES
- INFORMATION_SCHEMA.INDEX_STATISTICS
- INFORMATION_SCHEMA.INNODB_CHANGED_PAGES
- INFORMATION_SCHEMA.QUERY_RESPONSE_TIME
- INFORMATION_SCHEMA.TABLE_STATISTICS
- INFORMATION_SCHEMA.TEMPORARY_TABLES
- THREAD_STATISTICS
- INFORMATION_SCHEMA.USER_STATISTICS
- XTRADB_INTERNAL_HASH_TABLES
- XTRADB_READ_VIEW
- INFORMATION_SCHEMA.XTRADB_RSEG
- INFORMATION_SCHEMA.XTRADB_ZIP_DICT
- INFORMATION_SCHEMA.XTRADB_ZIP_DICT_COLS

Last update: 2022-09-27

Download PDF

17.7 Frequently Asked Questions

17.7.1 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.

17.7.2 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*.

17.7.3 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.

17.7.4 Q: When using the *Percona XtraBackup* to setup a replication replica 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 replica using *Percona XtraBackup* see How to set up a replica in 6 simple steps.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2022-09-27

Download PDF

17.8 Copyright and licensing information

17.8.1 Documentation licensing

17.8.2 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.

CONTACT US

For free technical help, visit the Percona Community Forum.

To report bugs or submit feature requests, open a JIRA ticket.

For paid support and managed or consulting services , contact Percona Sales.

Last update: 2023-02-14

Download PDF

17.9 Glossary

17.9.1 ACID

Set of properties that guarantee database transactions are processed reliably. Stands for Atomicity, Consistency, Isolation, Durability.

17.9.2 Atomicity

Atomicity means that database operations are applied following a “all or nothing” rule. A transaction is either fully applied or not at all.

17.9.3 Consistency

Consistency means that each transaction that modifies the database takes it from one consistent state to another.

17.9.4 Durability

Once a transaction is committed, it will remain so.

17.9.5 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.

17.9.6 Isolation

The Isolation requirement means that no transaction can interfere with another.

17.9.7 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.

17.9.8 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).

17.9.9 LSN

Log Serial Number. A term used in relation to the InnoDB or XtraDB storage engines.

17.9.10 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.

17.9.11 my.cnf

The file name of the default MySQL configuration file.

17.9.12 MyISAM

A MySQL storage engine that was the default until MySQL 5.5.

17.9.13 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.

17.9.14 NUMA

Non-Uniform Memory Access (NUMA) is a computer memory design used in multiprocessing, where the memory access time depends on the memory location relative to a processor. Under NUMA, a processor can access its own local memory faster than non-local memory, that is, memory local to another processor or memory shared between processors. The whole system may still operate as one unit, and all memory is basically accessible from everywhere, but at a potentially higher latency and lower performance.

17.9.15 Percona Server for MySQL

Percona's branch of MySQL with performance and management improvements.

17.9.16 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).

17.9.17 XtraDB

Percona's improved version of InnoDB providing performance, features and reliability above what is shipped by Oracle in InnoDB.

Last update: 2022-09-27