Percona Server Documentation

Release 5.5.62-38.14

Nov 14, 2022

CONTENTS

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

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

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

Part I

Introduction

CHAPTER

ONE

THE *PERCONA XTRADB* STORAGE ENGINE

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

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

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

CHAPTER

TWO

LIST OF FEATURES AVAILABLE IN *PERCONA SERVER* RELEASES

Percona Server 5.1	*Percona Server* 5.5	*Percona Server* 5.6
Improved Buffer Pool Scalability	Improved Buffer Pool Scalability	Improved Buffer Pool Scalability
Configurable Insert Buffer	Configurable Insert Buffer	Feature not implemented
Improved InnoDB I/O Scalability	Improved InnoDB I/O Scalability	Improved InnoDB I/O Scalability
More Concurrent Transactions Available	More Concurrent Transactions Available	Replaced by the upstream implementation[1]
Feature not implemented	*Multiple Adaptive Hash Search Partitions*	Multiple Adaptive Hash Search Partitions
Dedicated Purge Thread	Replaced by the upstream implementation[2]	Replaced by the upstream implementation[2]
Drop table performance	Drop table performance	Replaced by the upstream fix[3]
Feature not implemented	Atomic write support for Fusion-io devices	Atomic write support for Fusion-io devices
Configuration of the Doublewrite Buffer	Configuration of the Doublewrite Buffer	Feature not implemented
Query Cache Enhancements	Query Cache Enhancements	Query Cache Enhancements
Fast InnoDB Checksum[4]	Fast InnoDB Checksum[4]	Replaced by the upstream implementation[4]
Reduced Buffer Pool Mutex Contention	Reduced Buffer Pool Mutex Contention	Feature not implemented
InnoDB timer-based Concurrency Throttling	InnoDB timer-based Concurrency Throttling	Replaced by the upstream implementation[5]
HandlerSocket	HandlerSocket	Feature not implemented[6]
Feature not implemented	Improved NUMA support	Improved NUMA support
Feature not implemented	Thread Pool	Thread Pool
Feature not implemented	Binary Log Group Commit	Replaced by the upstream implementation[7]
Support of Multiple Page Sizes[8]	Support of Multiple Page Sizes[8]	Replaced by the upstream implementation[8]
Suppress Warning Messages	Suppress Warning Messages	Suppress Warning Messages
Handle BLOB End of Line	Handle BLOB End of Line	Replaced by the upstream implementation[9]
Ability to change database for mysqlbinlog	Ability to change database for mysqlbinlog	Ability to change database for mysqlbinlog
Replication Stop Recovery	Feature not implemented	Feature not implemented
Fixed Size for the Read Ahead Area	Fixed Size for the Read Ahead Area	Fixed Size for the Read Ahead Area
Fast Shutdown	Feature not implemented	Feature not implemented
Feature not implemented	Improved MEMORY Storage Engine	Improved MEMORY Storage Engine

Continued on next page

Table 2.1 – continued from previous page

Percona Server 5.1	*Percona Server* 5.5	*Percona Server* 5.6
Feature not implemented	Restricting the number of binlog files	Restricting the number of binlog files
Ignoring missing tables in mysqldump	Ignoring missing tables in mysqldump	Ignoring missing tables in mysqldump
Too Many Connections Warning	Too Many Connections Warning	Too Many Connections Warning
Handle Corrupted Tables	Handle Corrupted Tables	Handle Corrupted Tables
Crash-Resistant Replication	Crash-Resistant Replication	Replaced by the upstream implementation[10]
Lock-Free SHOW SLAVE STATUS	Lock-Free SHOW SLAVE STATUS	Lock-Free SHOW SLAVE STATUS
Fast InnoDB Recovery Process	Fast InnoDB Recovery Stats	Feature not implemented
InnoDB Data Dictionary Size Limit	InnoDB Data Dictionary Size Limit	Replaced by the upstream implementation[11]
Expand Table Import	Expand Table Import	Replaced by the upstream implementation[12]
Dump/Restore of the Buffer Pool	Dump/Restore of the Buffer Pool	Replaced by the upstream implementation[13]
Fast Index Creation	Fast Index Creation	Replaced by the upstream implementation[14]
Expanded Fast Index Creation	Expanded Fast Index Creation	Expanded Fast Index Creation
Prevent Caching to FlashCache	Prevent Caching to FlashCache	Feature not implemented
Percona Toolkit UDFs	Percona Toolkit UDFs	Percona Toolkit UDFs
Support for Fake Changes	Support for Fake Changes	Support for Fake Changes
Kill Idle Transactions	Kill Idle Transactions	Kill Idle Transactions
XtraDB changed page tracking	XtraDB changed page tracking	XtraDB changed page tracking
Feature not implemented	Enforcing Storage Engine	Enforcing Storage Engine
Feature not implemented	Utility user	Utility user
Feature not implemented	Extending the secure-file-priv server option	Extending the secure-file-priv server option
Feature not implemented	Expanded Program Option Modifiers	Expanded Program Option Modifiers
Feature not implemented	PAM Authentication Plugin	PAM Authentication Plugin
Feature not implemented	Feature not implemented	Log Archiving for XtraDB
InnoDB Statistics	InnoDB Statistics	Replaced by the upstream implementation[15]
User Statistics	User Statistics	User Statistics
Slow Query Log	Slow Query Log	Slow Query Log
Count InnoDB Deadlocks	Count InnoDB Deadlocks	Count InnoDB Deadlocks
Log All Client Commands (syslog)	Log All Client Commands (syslog)	Log All Client Commands (syslog)
Response Time Distribution	Response Time Distribution	Feature not implemented
Show Storage Engines	Show Storage Engines	Show Storage Engines
Show Lock Names	Show Lock Names	Show Lock Names
Process List	Process List	Process List
Misc. INFORMATION_SCHEMA Tables	Misc. INFORMATION_SCHEMA Tables	Misc. INFORMATION_SCHEMA Tables
Feature not implemented	Extended Show Engine InnoDB Status	Extended Show Engine InnoDB Status
Feature not implemented	Thread Based Profiling	Thread Based Profiling

Continued on next page

Table 2.1 – continued from previous page

Percona Server 5.1	*Percona Server* 5.5	*Percona Server* 5.6
Feature not implemented	Feature not implemented	XtraDB Performance Improvements for I/O-Bound Highly-Concurrent Workloads
Feature not implemented	Feature not implemented	Page cleaner thread tuning
Feature not implemented	Feature not implemented	Statement Timeout
Feature not implemented	Extended SELECT INTO OUTFILE/DUMPFILE	Extended SELECT INTO OUTFILE/DUMPFILE
Feature not implemented	Feature not implemented	Per-query variable statement
Feature not implemented	Extended mysqlbinlog	Extended mysqlbinlog
Feature not implemented	Slow Query Log Rotation and Expiration	Slow Query Log Rotation and Expiration
Feature not implemented	Metrics for scalability measurement	Metrics for scalability measurement
Feature not implemented	Audit Log	Audit Log
Feature not implemented	Feature not implemented	Backup Locks

2.1 Other Reading

- changed_in_56
- *Percona Server In-Place Upgrading Guide: From 5.1 to 5.5*
- Percona Sever In-Place Upgrading Guide: From 5.5 to 5.6
- Upgrading from MySQL 5.1 to 5.5
- What Is New in MySQL 5.5
- What Is New in MySQL 5.6

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

CHAPTER

THREE

PERCONA SERVER FEATURE COMPARISON

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

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

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

Features	Percona Server 5.5.27	MySQL 5.5.27
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

Extra Features for Developers	Percona Server 5.5.27	MySQL 5.5.27
NoSQL Socket-Level Interface	Yes	
Extra Hash/Digest Functions	Yes	

Extra Diagnostic Features	Percona Server 5.5.27	MySQL 5.5.27
INFORMATION_SCHEMA Tables	60	37
Global Performance and Status Counters	370	310
Per-Table Performance Counters	Yes	
Per-Index Performance Counters	Yes	
Per-User Performance Counters	Yes	
Per-Client Performance Counters	Yes	
Per-Thread Performance Counters	Yes	
High-Resolution Process List Timing	Yes	
Detailed Query Execution and Plan Log	Yes	
Global Query Response Time Statistics	Yes	
InnoDB Data Dictionary as I_S Tables	Yes	
Access to InnoDB Data Statistics	Yes	
Enhanced SHOW ENGINE INNODB STATUS	Yes	
Enhanced Mutex Diagnostics	Yes	
Undo Segment Information	Yes	

Durability and Reliability Enhancements	Percona Server 5.5.27	MySQL 5.5.27
Transactional Replication State	Yes	
Handles Corrupted Tables Gracefully	Yes	

Performance & Scalability Enhancements	Percona Server 5.5.27	MySQL 5.5.27
Support for Multiple I/O Threads	Yes	Yes
Dedicated Purge Thread	Yes	Yes
Self-Tuning Checkpoint Algorithm	Yes	
Fine-Grained Mutex Locking	Yes	
Lock-Free Algorithms	Yes	
Improved MEMORY Storage Engine	Yes	
Partitioned Adaptive Hash Search	Yes	
Separate Doublewrite File	Yes	
Fast Checksum Algorithm	Yes	
Buffer Pool Pre-Load	Yes	
Fast Shut-Down	Yes	
Background Table Drop	Yes	
Support for FlashCache	Yes	
Read-Ahead Improvements	Yes	

Extra Features for DBA/Operations Staff	Percona Server 5.5.27	MySQL 5.5.27
Configurable Page Sizes	Yes	
Import Tables From Different Servers	Yes	
Configurable Data Dictionary Size	Yes	
Configurable Insert Buffer Size	Yes	
Active Change Buffer Purging	Yes	
Error/Warning Logging Enhancements	Yes	
Configurable Fast Index Creation	Yes	
Support for Fake Changes	Yes	
Changed Page Tracking	Yes	

Running Database as a Service	Percona Server 5.5.27	MySQL 5.5.27
Special Utility User	Yes	
Expanded Program Option Modifiers	Yes	
Extended `secure-file-priv` option	Yes	
Enforcing the Specific Storage Engine	Yes	

Part II

Installation

CHAPTER

FOUR

INSTALLING *PERCONA SERVER* 5.5

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

- *Installing Percona Server from Repositories* (recommended)
- Installing *Percona Server* from Downloaded *:ref:'rpm <standalone_rpm>* or *apt* Packages
- *Installing Percona Server from a Binary Tarball*
- *Installing Percona Server from a Source Tarball*
- *Installing Percona Server from the Git Source Tree*
- *Compiling Percona Server from Source*

Before installing, you might want to read the *Percona Server 5.5 Release notes*.

4.1 Installing *Percona Server* 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*, *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 recommend way of installing where possible.

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

4.1.1 Installing *Percona Server* on *Debian* and *Ubuntu*

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

Supported Releases:

- Debian:
- 7.0 (wheezy)
- 8.0 (jessie)
- 9.0 (stretch)
- Ubuntu:
- 14.04LTS (trusty)
- 16.04LTS (xenial)
- 17.04 (zesty)

- 17.10 (artful)
- 18.04 (bionic)

Supported Platforms:

- x86
- x86_64 (also known as `amd64`)

What's in each DEB package?

The `percona-server-server-5.5` package contains the database server itself, the `mysqld` binary and associated files.

The `percona-server-common-5.5` package contains files common to the server and client.

The `percona-server-client-5.5` package contains the command line client.

The `percona-server-5.5-dbg` package contains debug symbols for the server.

The `percona-server-test-5.5` package contains the database test suite.

The `percona-server-source-5.5` package contains the server source.

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

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

Installing *Percona Server* from Percona `apt` repository

1. Fetch the repository packages from Percona web:

   ```
   wget https://repo.percona.com/apt/percona-release_0.1-6.$(lsb_release -sc)_all.deb
   ```

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

   ```
   dpkg -i percona-release_0.1-6.$(lsb_release -sc)_all.deb
   ```

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

3. Remember to update the local cache:

   ```
   apt-get update
   ```

4. After that you can install the server package:

   ```
   apt-get install percona-server-server-5.5
   ```

Percona `apt` Testing repository

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

```
# Testing & pre-release packages
#
deb http://repo.percona.com/apt VERSION testing
deb-src http://repo.percona.com/apt VERSION testing
```

Apt-Pinning the packages

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

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

For more information about the pinning you can check the official debian wiki.

Installing *Percona Server* 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* 5.5.44-37.3 release packages for *Debian* 8.0:

```
$ wget https://www.percona.com/downloads/Percona-Server-5.5/Percona-Server-5.
↪5.44-37.3/binary/debian/jessie/x86_64/Percona-Server-5.5.44-37.3-r729fbe2-
↪jessie-x86_64-bundle.tar
```

You should then unpack the bundle to get the packages:

```
$ tar xvf Percona-Server-5.5.44-37.3-r729fbe2-jessie-x86_64-bundle.tar
```

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

```
$ ls *.deb
libperconaserverclient18_5.5.44-rel37.3-1.jessie_amd64.deb
libperconaserverclient18-dev_5.5.44-rel37.3-1.jessie_amd64.deb
percona-server-5.5-dbg_5.5.44-rel37.3-1.jessie_amd64.deb
percona-server-client_5.5.44-rel37.3-1.jessie_amd64.deb
percona-server-client-5.5_5.5.44-rel37.3-1.jessie_amd64.deb
percona-server-common-5.5_5.5.44-rel37.3-1.jessie_amd64.deb
percona-server-server_5.5.44-rel37.3-1.jessie_amd64.deb
percona-server-server-5.5_5.5.44-rel37.3-1.jessie_amd64.deb
percona-server-source-5.5_5.5.44-rel37.3-1.jessie_amd64.deb
percona-server-test-5.5_5.5.44-rel37.3-1.jessie_amd64.deb
```

Now you can install *Percona Server* 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* installation (`libperconaserverclient18_5.5.44-rel37.3-1.jessie_amd64.deb`, `percona-server-client-5.5_5.5.44-rel37.3-1.jessie_amd64.deb`, `percona-server-common-5.5_5.5.44-rel37.3-1.jessie_amd64.deb`, and `percona-server-server-5.5_5.5.44-rel37.3-1.jessie_amd64.deb`).

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

Running *Percona Server*

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

1. Starting the service

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

   ```
   $ sudo service mysql start
   ```

2. Confirming that service is running

   You can check the service status by running:

   ```
   $ service mysql status
   ```

3. Stopping the service

   You can stop the service by running:

   ```
   $ sudo service mysql stop
   ```

4. Restarting the service

   You can restart the service by running:

   ```
   $ sudo service mysql restart
   ```

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

Uninstalling *Percona Server*

To uninstall *Percona Server* you'll need to remove all the installed packages. Removing packages with **`apt-get remove`** will leave the configuration and data files. Removing the packages with **`apt-get purge`** will remove all the packages with configuration files and data files (all the databases). Depending on your needs you can choose which command better suits you.

1. Stop the *Percona Server* service

   ```
   $ sudo service mysql stop
   ```

2. Remove the packages

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

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

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

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

4.1.2 Installing *Percona Server* on Red Hat Enterprise Linux and CentOS

Ready-to-use packages are available from the *Percona Server* 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.

Supported Releases:

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

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

Supported Platforms:

- x86
- x86_64 (also known as `amd64`)

What's in each RPM package?

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

The `Percona-Server-server-55` package contains the server itself (the `mysqld` binary).

The `Percona-Server-55-debuginfo` package contains debug symbols for the server.

The `Percona-Server-client-55` package contains the command line client.

The `Percona-Server-devel-55` package contains the header files needed to compile software using the client library.

The `Percona-Server-shared-55` package includes the client shared library.

The `Percona-Server-shared-compat` package includes shared libraries for software compiled against old versions of the client library. Following libraries are included in this package: `libmysqlclient.so.12`, `libmysqlclient.so.14`, `libmysqlclient.so.15`, and `libmysqlclient.so.16`.

The `Percona-Server-test-55` package includes the test suite for *Percona Server*.

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

Installing *Percona Server* from Percona `yum` repository

1. Install the Percona repository

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

```
yum install http://www.percona.com/downloads/percona-release/redhat/0.1-6/percona-
↪release-0.1-6.noarch.rpm
```

   You should see some output such as the following:

```
Retrieving http://www.percona.com/downloads/percona-release/redhat/0.1-6/percona-
↪release-0.1-6.noarch.rpm
Preparing...                ########################################### [100%]
   1:percona-release        ########################################### [100%]
```

Note: *RHEL/Centos* 5 doesn't support installing the packages directly from the remote location so you'll need to download the package first and install it manually with **`rpm`**:

```
wget http://www.percona.com/downloads/percona-release/redhat/0.1-6/percona-
↪release-0.1-6.noarch.rpm
rpm -ivH percona-release-0.1-6.noarch.rpm
```

2. Testing the repository

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

```
yum list | grep percona
```

   You should see output similar to the following:

```
...
Percona-Server-55-debuginfo.x86_64          5.5.44-rel37.3.el6           percona-
↪release-x86_64
Percona-Server-client-55.x86_64             5.5.44-rel37.3.el6           percona-
↪release-x86_64
Percona-Server-devel-55.x86_64              5.5.44-rel37.3.el6           percona-
↪release-x86_64
Percona-Server-server-55.x86_64             5.5.44-rel37.3.el6           percona-
↪release-x86_64
Percona-Server-shared-55.x86_64             5.5.44-rel37.3.el6           percona-
↪release-x86_64
Percona-Server-shared-compat.x86_64         5.1.68-rel14.6.551.rhel6     percona-
↪release-x86_64
Percona-Server-test-55.x86_64               5.5.44-rel37.3.el6           percona-
↪release-x86_64
...
```

3. Install the packages

   You can now install *Percona Server* by running:

```
yum install Percona-Server-server-55
```

Percona *yum* Testing repository

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

Installing *Percona Server* using downloaded rpm packages

1. Download the packages of the desired series for your architecture from the download page. The easiest way is to download bundle which contains all the packages. Following example will download *Percona Server* 5.5.44-37.3 release packages for *CentOS* 6:

```
wget https://www.percona.com/downloads/Percona-Server-5.5/Percona-Server-5.5.44-
↪37.3/binary/redhat/6/x86_64/Percona-Server-5.5.44-37.3-r729fbe2-el6-x86_64-
↪bundle.tar
```

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

```
tar xvf Percona-Server-5.5.44-37.3-r729fbe2-el6-x86_64-bundle.tar
```

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

```
$ ls *.rpm

Percona-Server-55-debuginfo-5.5.44-rel37.3.el6.x86_64.rpm
Percona-Server-client-55-5.5.44-rel37.3.el6.x86_64.rpm
Percona-Server-devel-55-5.5.44-rel37.3.el6.x86_64.rpm
Percona-Server-server-55-5.5.44-rel37.3.el6.x86_64.rpm
Percona-Server-shared-55-5.5.44-rel37.3.el6.x86_64.rpm
Percona-Server-test-55-5.5.44-rel37.3.el6.x86_64.rpm
```

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

```
rpm -ivh Percona-Server-server-55-5.5.44-rel37.3.el6.x86_64.rpm \
Percona-Server-client-55-5.5.44-rel37.3.el6.x86_64.rpm \
Percona-Server-shared-55-5.5.44-rel37.3.el6.x86_64.rpm
```

This will install only packages required to run the *Percona Server*. 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.

Running *Percona Server*

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

1. Starting the service

   Percona Server isn't started automatically on *RHEL* and *CentOS* after it gets installed. You should start it by running:

   ```
   service mysql start
   ```

2. Confirming that service is running

   You can check the service status by running:

   ```
   service mysql status
   ```

3. Stopping the service

   You can stop the service by running:

   ```
   service mysql stop
   ```

4. Restarting the service

   You can restart the service by running:

   ```
   service mysql restart
   ```

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

Uninstalling *Percona Server*

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

1. Stop the *Percona Server* service

   ```
   service mysql stop
   ```

2. Remove the packages

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

3. Remove the data and configuration files

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

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

4.2 Installing *Percona Server* from a Binary Tarball

In *Percona Server* `5.5.43-37.2` and newer, the single binary tarball was replaced with multiple tarballs depending on the *OpenSSL* library available in the distribution:

- ssl100 - for all *Debian/Ubuntu* versions except *Squeeze* (`libssl.so.1.0.0 => /usr/lib/x86_64-linux-gnu/libssl.so.1.0.0 (0x00007f2e389a5000)`);
- ssl098 - only for *Debian Squeeze* (`libssl.so.0.9.8 => /usr/lib/libssl.so.0.9.8 (0x00007f9b30db6000)`);
- ssl101 - for *CentOS* 6 and *CentOS* 7 (`libssl.so.10 => /usr/lib64/libssl.so.10 (0x00007facbe8c4000)`);
- ssl098e - to be used only for *CentOS* 5 (`libssl.so.6 => /lib64/libssl.so.6 (0x00002aed5b64d000)`).

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

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

```
$ wget http://www.percona.com/redir/downloads/Percona-Server-5.5/Percona-Server-5.5.
↪43-37.2/binary/tarball/Percona-Server-5.5.43-rel37.2-Linux.x86_64.ssl100.tar.gz
```

4.3 Installing *Percona Server* from a Source Tarball

Fetch and extract the source tarball. For example:

```
$ wget http://www.percona.com/downloads/Percona-Server-5.5/Percona-Server-5.5.43-37.2/
↪source/Percona-Server-5.5.42-rel37.1-Linux.x86_64.tar.gz
$ tar xfz Percona-Server-5.5.42-rel37.1-Linux.x86_64.tar.gz
```

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

4.4 Installing *Percona Server* from the Git Source Tree

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

You can now fetch the latest *Percona Server* 5.5 sources.

```
$ git clone https://github.com/percona/percona-server.git
$ cd percona-server
$ git checkout 5.5
```

If you are going to be making changes to *Percona Server* 5.5 and wanting to distribute the resulting work, you can generate a new source tarball (exactly the same way as we do for release):

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

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

4.5 Compiling *Percona Server* from Source

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

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

```
$ cmake . -DCMAKE_BUILD_TYPE=RelWithDebInfo -DBUILD_CONFIG=mysql_release -DFEATURE_
↪SET=community -DWITH_EMBEDDED_SERVER=OFF
```

Now, compile using make

```
$ make
```

Install:

```
$ make install
```

Percona Server 5.5 will now be installed on your system.

4.6 Building *Percona Server* Debian/Ubuntu packages

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

Extract the source tarball:

```
$ tar xfz percona-server-5.5.34-32.0.tar.gz
$ cd percona-server-5.5.34-32.0
```

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

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

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

```
$ dch -D unstable --force-distribution -v "5.5.34-32.0-1" "Update to 5.5.34-32.0"
```

Build the Debian source package:

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

Use sbuild to build the binary package in a chroot:

```
$ sbuild -d sid percona-server-5.5_5.5.34_32.0-1.dsc
```

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

Note: *PAM Authentication Plugin* has been merged into *Percona Server* in `5.5.24-26.0` but it is not built with the server by default. In order to build the Percona Server with PAM plugin, additional option `-DWITH_PAM=ON` should be used.

CHAPTER

FIVE

PERCONA SERVER IN-PLACE UPGRADING GUIDE: FROM 5.1 TO 5.5

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.

Having this in mind, the changes in the in the 5.5 series can be grouped into 3 areas:

- Server configuration
- Server behavior and functioning
- SQL changes

The following is a summary of the more relevant changes in the 5.5 series. For more details, see

- *Percona Server documentation*
- http://dev.mysql.com/doc/refman/5.5/en/upgrading-from-previous-series.html

Warning: Upgrade 5.1 to 5.5 on a crashed instance is not recommended. If the server instance has crashed, crash recovery should be run before proceeding with the upgrade.

5.1 Changes in Server Configuration

5.1.1 Features and Variables

The configuration options and table columns for the following features have been modified in *Percona Server* 5.5:

Feature	5.1 Series	5.5 Series
Improved InnoDB I/O Scalability	innodb_adaptive_checkpoint	innodb_adaptive_flushing_method
Suppress Warning Messages	suppress_log_warning_1592	log_warnings_suppress
Handle Corrupted Tables	innodb_pass_corrupt_table	innodb_corrupt_table_action
Expand Table Import	innodb_expand_import	innodb_import_table_from_xtrabackup
Dump/Restore of the Buffer Pool at Startup	innodb_auto_lru_dump	innodb_buffer_pool_restore_at_startup
Slow Query Log	log_slow_timestamp_every	slow_query_log_timestamp_always
	slow_query_log_microseconds_timestamp	slow_query_log_timestamp_precision
	use_global_log_slow_control	slow_query_log_use_global_control
Response Time Distribution	enable_query_response_time_stats	query_response_time_stats
Multiple Rollback Segments	innodb_extra_rsegments	**(removed)**
Dedicated Purge Thread	innodb_use_purge_thread	using upstream version

Shared Memory Buffer Pool

The *SHM buffer pool* patch has been replaced with the safer *LRU Dump/Restore* patch, which provides similar improvements in restart performance and has the advantage of persisting across machine restarts.

The configuration variables for my.cnf have been kept for compatibility and warnings will be printed for the deprecated options (`innodb_buffer_pool_shm_key` and `innodb_buffer_pool_shm_checksum`) if used.

Instructions for disabling the SHM buffer pool can be found *here* and for setting up LRU dump/restore *here*.

Multiple Rollback Segments

Percona Server 5.1 offered a feature that enabled InnoDB to use multiple rollback segments, relieving a major cause of resource contention in write-intensive workloads. In MySQL 5.5, Oracle implemented a similar feature, and so in Percona Server 5.5, the `innodb_extra_rsegments` option has been replaced by the MySQL 5.5 `innodb_rollback_segment` option.

InnoDB Statistics

Three fields in table INNODB_INDEX_STATS were renamed:

5.1 Series	5.5 Series
row_per_keys	rows_per_key
index_size	index_total_pages
leaf_pages	index_leaf_pages

For more information, see its documentation documentation.

Process List

The columns ROWS_EXAMINED, ROWS_SENT, and ROWS_READ have been added to the SHOW PROCESSLIST command and the table PROCESSLIST.

For more information, see its documentation documentation.

5.1.2 Grepping Old Variables

You can check if old variables are being used in your configuration file by issuing the following line in a shell:

```
egrep -ni 'innodb_adaptive_checkpoint|suppress_log_warning_1592|innodb_pass_corrupt_
↪table|innodb_expand_import|innodb_auto_lru_dump|log_slow_timestamp_every|slow_query_
↪log_microseconds_timestamp|use_global_log_slow_control|enable_query_response_time_
↪stats|innodb_buffer_pool_shm_key|innodb_buffer_pool_shm_checksum' /PATH/TO/my.cnf
```

5.1.3 New Features

You may also want to check the new features available in *Percona Server* 5.5:

- Multiple Adaptive Hash Search Partitions
- Crash-Resistant Replication
- Show Engine InnoDB Status
- Plugins

All plugins not included with Percona Server will have to be recompiled for *Percona Server* 5.5. There is a new plugin interface that complements the plugin API, plugins must be recompiled and linked to libmysqlservices. The plugins bundled with the server are already linked, you can list the installed plugins with the `SHOW PLUGINS` statement:

```
mysql> SHOW PLUGINS;
+----------------------+--------+--------------------+---------+---------+
| Name                 | Status | Type               | Library | License |
+----------------------+--------+--------------------+---------+---------+
| binlog               | ACTIVE | STORAGE ENGINE     | NULL    | GPL     |
 ...
+----------------------+--------+--------------------+---------+---------+
```

For more information, see:

- http://dev.mysql.com/doc/refman/5.5/en/plugin-services.html
- http://dev.mysql.com/doc/refman/5.5/en/plugin-installing-uninstalling.html

5.1.4 Upgrading from MySQL 5.1

If you are upgrading from *MySQL* 5.1 instead of *Percona Server* 5.1, you should take into account that the *InnoDB* Plugin has been included in the standard *MySQL* 5.5 distribution as default for the InnoDB storage engine.

This change does not affect *Percona Server* as it has the *XtraDB* storage engine - an enhanced version of *InnoDB* - built-in since the 5.1 series. If you are migrating from *MySQL* 5.1.X, and you were using the *InnoDB* plugin, make sure to remove it from the configuration file by deleting the following two lines from the `[mysqld]` section:

```
[mysqld]
ignore-builtin-innodb  # <- DELETE
plugin-load=innodb=ha_innodb_plugin.so # <- DELETE
```

otherwise, the server won't start. Strictly speaking, the ignore-builtin-innodb option will disable *XtraDB* in *Percona Server* 5.5 if set, and the server will not start if no other default storage engine is specified (i.e. `default-storage-engine=MyISAM`).

Also, the variable innodb_file_io_threads has been replaced by innodb_read_io_threads and innodb_write_io_threads (these variables were already introduced in Percona Server 5.1). All of them defaults to 4, you should replace the old variable with the two new ones with the proper value (or delete it if the default - 4 - is acceptable).

5.2 Changes in Server Behavior and Functioning

5.2.1 Privileges

The schema of the grants tables in *MySQL* 5.5 has changed and a new table has been introduced, `proxy_priv`.

The conversion to the new schema will be handled by **`mysql_upgrade`** (see below).

5.2.2 Logs

The server will not rename the current log file with the suffix `-old` when issuing a `FLUSH LOGS` statement.

The renaming must be done by the user before flushing. It is important to note this as if it is not renamed before, the past log will be lost.

5.2.3 Numeric calculations

On the numeric side, the server includes a new a library for conversions between strings and numbers, `dtoa`.

This library provides the basis for an improved conversion between string or `DECIMAL` values and approximate-value (`FLOAT` or `DOUBLE`) numbers. Also, all numeric operators and functions on integer, floating-point and `DECIMAL` values throw an `out of range` error (`ER_DATA_OUT_OF_RANGE`) rather than returning an incorrect value or `NULL`.

If an application rely on previous numeric results, it may have to be adjusted to the new precision or behavior.

5.2.4 Replication

When upgrading in a replication environment, a change in handling of `IF NOT EXISTS` results in an incompatibility for statement-based replication from a *MySQL* 5.1 master prior to 5.1.51 to a *MySQL* 5.5 slave.

If you use `CREATE TABLE IF NOT EXISTS ... SELECT` statements, upgrade the master first to 5.1.51 or higher.

Note that this differs from the usual replication upgrade advice of upgrading the slave first.

5.2.5 Indexes

The stopword file is loaded and searched using `latin1` if `character_set_server` is `ucs2`, `utf16`, or `utf32`. If any table was created with `FULLTEXT` indexes while the server character set was `ucs2`, `utf16`, or `utf32`, it should be repaired using this statement `REPAIR TABLE tbl_name QUICK;`.

5.2.6 Error Messages

The `--language` option has been deprecated and is an alias for `--lc-messages-dir` and `--lc-messages`.

Also, error messages are now constructed in `UTF-8` and returned with `character_set_results` encoding.

5.2.7 Unicode Support

The Unicode implementation has been extended to provide support for supplementary characters that lie outside the Basic Multilingual Plane (BMP), introducing the `utf16`, `utf32` and `utf8mb4` charsets.

If you are considering upgrading from utf8 to utf8mb4 to take advantage of the supplementary characters, you may have to adjust the size of the fields and indexes in the future. See http://dev.mysql.com/doc/refman/5.5/en/charset-unicode-upgrading.html.

Upgrading to `utf8mb4` will not take place unless you explicitly change the charset, i.e. with a ALTER TABLE... statement.

5.2.8 Changes in SQL

The following changes require modifications in the SQL statements in the client side:

- `INTO` clauses are no longer accepted in nested SELECT statements. Modify the SQL statements to not contain the clause.
- Alias declarations outside `table_reference` are not allowed for multiple-table `DELETE` statements. Modify those statements to use aliases only inside `table_reference` part.

- Alias resolution does not require qualification and alias reference should not be qualified with the database name.
- New reserved words:
  - GENERAL
  - IGNORE_SERVER_IDS
  - MASTER_HEARTBEAT_PERIOD
  - MAXVALUE
  - RESIGNAL
  - SIGNAL
  - SLOW
- TRUNCATE TABLE fails for a *XtraDB* table if there are any FOREIGN KEY constraints from other tables that reference the table. As *XtraDB* always use the fast truncation technique in 5.5 - equivalent to DROP TABLE and CREATE TABLE - you should modify the SQL statements to issue DELETE FROM table_name for such tables instead of TRUNCATE TABLE or an error will be returned in that cases.

5.3 BEFORE STARTING: FULL BACKUP

Before starting the upgrade, a full backup of the data must be done. Doing a full backup will guarantee us the safety of going back without consequences if something goes wrong. After all, it's only one line:

```
$ innobackupex --user=DBUSER --password=SECRET /path/where/to/store/backup/
```

This will backup all the data in your server to a time stamped subdirectory of the path provided.

innobackupex is a *Perl* script distributed with *XtraBackup*, a hot-backup utility for *MySQL* -based servers that doesn't block your database during the backup. If you don't have *XtraBackup* installed already, instructions can be found here.

You should backup your entire configuration file - my.cnf - also. The file is usually located in /etc/mysql/ or /etc/ or as .my.cnf in user's home directory,

```
$ cp /etc/mysql/my.cnf /path/where/to/store/backup/
```

While this is not an "in-place" upgrade technically, where possible, doing a full dump of the server's data for restoring it later is recommended. By this way, the indexes from all tables will be rebuilt explicitly, and any binary compatibility issue will be avoided:

```
$ mysqldump --user=root -p --all-databases --routines > mydata.sql
```

This is not possible in some cases because of available space or downtime requirements, but if it is feasible, it is highly recommended.

5.4 Upgrading using the Percona repositories

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

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

- *Percona APT Repository*

- *Percona YUM Repository*

5.4.1 DEB-based distributions

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

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

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

Note: For extra safety doing the slow InnoDB shutdown before the upgrade is recommended.

Then install the new server with:

```
$ sudo apt-get install percona-server-server-5.5
```

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

Note that this procedure is the same for upgrading from *MySQL* 5.1 or 5.5 to *Percona Server* 5.5.

5.4.2 RPM-based distributions

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

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

and check your installed packages with:

```
$ rpm -qa | grep Percona-Server
Percona-Server-client-51-5.1.57-rel12.8.232.rhel5.i686.rpm
Percona-Server-server-51-5.1.57-rel12.8.232.rhel5.i686.rpm
Percona-Server-shared-51-5.1.57-rel12.8.232.rhel5.i686.rpm
```

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

After checking, proceed to remove them without dependencies:

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

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

Note that this procedure is the same for upgrading from *MySQL* 5.1 or 5.5 to *Percona Server* 5.5: just grep `'^mysql-'` instead of `Percona-Server` and remove them.

You will have to install the following packages:

- `Percona-Server-server-55`
- `Percona-Server-client-55`

```
$ yum install Percona-Server-server-55 Percona-Server-client-55
```

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

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

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

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

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

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

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

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

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

5.5 Upgrading using Standalone Packages

5.5.1 DEB-based distributions

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

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

and remove the installed packages with their dependencies:

```
$ sudo apt-get autoremove percona-server-server-51 percona-server-client-51
```

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

Then, download the following packages for your architecture:

- `percona-server-server-5.5`
- `percona-server-client-5.5`
- `percona-server-common-5.5`
- `libperconaserverclient16`

At the moment of writing this guide, for *Ubuntu* Maverick on `i686`, a way of doing this is:

```
$ wget -r -l 1 -nd -A deb -R "*dev*" http://www.percona.com/redir/downloads/Percona-
↪Server-5.5/Percona-Server-5.5.12-20.3/deb/maverick/x86_64/
```

Install them in one command:

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

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

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

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

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

5.5.2 RPM-based distributions

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

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

and check your installed packages:

```
$ rpm -qa | grep Percona-Server
Percona-Server-client-51-5.1.57-rel12.8.232.rhel5.i686.rpm
Percona-Server-server-51-5.1.57-rel12.8.232.rhel5.i686.rpm
Percona-Server-shared-51-5.1.57-rel12.8.232.rhel5.i686.rpm
```

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

After checked that, proceed to remove them without dependencies:

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

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

Note that this procedure is the same for upgrading from *MySQL* 5.1 to *Percona Server* 5.5, just grep `'^mysql-'` instead of `Percona-Server` and remove them.

Download the following packages for your architecture:

- `Percona-Server-server-55`
- `Percona-Server-client-55`
- `Percona-Server-shared-55`

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

```
$ wget -r -l 1 -nd -A rpm -R "*devel*,*debuginfo*" http://www.percona.com/redir/
↪downloads/Percona-Server-5.5/Percona-Server-5.5.12-20.3/RPM/rhel5/i686/
```

Install them in one command:

```
$ rpm -ivh Percona-Server-server-55-5.5.12-rel20.3.118.rhel5.i686.rpm \
Percona-Server-client-55-5.5.12-rel20.3.118.rhel5.i686.rpm \
Percona-Server-shared-55-5.5.12-rel20.3.118.rhel5.i686.rpm
```

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

```
$ rpm -ivh Percona-Server-shared-55-5.5.12-rel20.3.118.rhel5.i686.rpm
$ rpm -ivh Percona-Server-client-55-5.5.12-rel20.3.118.rhel5.i686.rpm
$ rpm -ivh Percona-Server-server-55-5.5.12-rel20.3.118.rhel5.i686.rpm
```

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

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

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

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

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

```
$ mysql_upgrade
```

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

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

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

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

5.6 Other Reading

- Upgrading MySQL: Best Practices webinar,
- Upgrading MySQL webinar questiones

Part III

Scalability Improvements

CHAPTER

SIX

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
buf_pool_mutex	flags about IO
LRU_list_mutex	LRU list of blocks in buffer pool
flush_list_mutex	flush list of dirty blocks to flush
page_hash_latch	hash table to search blocks in buffer pool
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.

6.1 Other Information

6.1.1 Detecting Mutex Contention

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

Under normal circumstances this section should look like this:

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

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

```
1 ----------
2 SEMAPHORES
3 ----------
4 OS WAIT ARRAY INFO: reservation count 36255, signal count 12675
5 --Thread 10607472 has waited at buf/buf0rea.c line 420 for 0.00 seconds the␣
 ↪semaphore:
6 Mutex at 0x358068 created file buf/buf0buf.c line 597, lock var 0
7 waiters flag 0
```

```
8 --Thread 3488624 has waited at buf/buf0buf.c line 1177 for 0.00 seconds the␣
↪semaphore:
9 Mutex at 0x358068 created file buf/buf0buf.c line 597, lock var 0
10 waiters flag 0
11 --Thread 6896496 has waited at btr/btr0cur.c line 442 for 0.00 seconds the␣
↪semaphore:
12 S-lock on RW-latch at 0x8800244 created in file buf/buf0buf.c line 547
13 a writer (thread id 14879600) has reserved it in mode  exclusive
14 number of readers 0, waiters flag 1
15 Last time read locked in file btr/btr0cur.c line 442
16 Last time write locked in file buf/buf0buf.c line 1797
[...]
17 Mutex spin waits 0, rounds 452650, OS waits 22573
18 RW-shared spins 27550, OS waits 13682; RW-excl spins 0, OS waits 0
```

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

CHAPTER

SEVEN

CONFIGURABLE INSERT BUFFER

Percona has implemented several changes related to MySQL's InnoDB Insert Buffer. These features enable adjusting the insert buffer to the different workloads and hardware configurations.

7.1 System variables:

variable `innodb_ibuf_active_contract`

Version Info

- *`5.5.8-20.0`* – Introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 1

Range 0 - 1

This variable specifies whether the insert buffer can be processed before it reaches its maximum size. The following values are allowed:

- 0: the insert buffer is not processed until it is full. This is the standard *InnoDB* behavior.
- 1: the insert buffer can be processed even it is not full.

variable `innodb_ibuf_accel_rate`

Version Info

- *`5.5.8-20.0`* – Introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value 100

Range 100 - 999999999

This variable allows better control of the background thread processing the insert buffer. Each time the thread is called, its activity is altered by the value of both `innodb_io_capacity` and `innodb_ibuf_accel_rate` this way:

```
[real activity] = [default activity] * (innodb_io_capacity/100) * (innodb_ibuf_accel_
↪rate/100)
```

By increasing the value of *innodb_ibuf_accel_rate*, you will increase the insert buffer activity.

variable innodb_ibuf_max_size

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value Half the size of the *InnoDB* buffer pool

Range 0 - Half the size of the *InnoDB* buffer pool

Units Bytes

This variable specifies the maximum size of the insert buffer. By default the insert buffer is half the size of the buffer pool so if you have a very large buffer pool, the insert buffer will be very large too and you may want to restrict its size with this variable.

Setting this variable to 0 is equivalent to disabling the insert buffer. But then all changes to secondary indexes will be performed synchronously which will probably cause performance degradation. Likewise a too small value can hurt performance.

If you have very fast storage (ie storage with RAM-level speed, not just a RAID with fast disks), a value of a few MB may be the best choice for maximum performance.

7.2 Other Reading

- Some little known facts about InnoDB Insert Buffer
- 5.0.75-build12 Percona binaries

CHAPTER

EIGHT

IMPROVED *INNODB* I/O SCALABILITY

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*, however. The goal of this feature is to provide a more exhaustive set of options for *XtraDB*. Note that some of these parameters are already available in the *InnoDB* plugin.

These new variables are divided into several categories:

- Configuration of the capacity of the I/O subsystem (number of read and write threads, number of available I/O operations per second)
- Additional options to control the flushing and checkpointing activities
- Configuration of the insert buffer (maximum size, activity)
- Various other options

8.1 Version Specific Information

- 5.5.19-24.0
  - Added option value `cont` for variable *innodb_flush_neighbor_pages*.
- 5.5.8-20.0
  - Added variable *innodb_adaptive_flushing_method*.
  - Added variable `innodb_ibuf_active_merge`.
  - Added variable `innodb_ibuf_merge_rate`.
  - Added variable *innodb_use_global_flush_log_at_trx_commit*.
- 5.5.20-beta
  - The 'reflex' value was removed from *innodb_adaptive_flushing_method* in 5.5.20-beta as a fix for bug #689450.

8.2 System Variables

variable `innodb_adaptive_flushing`

Command Line NOT AVAILABLE

Variable Type BOOLEAN

Default Value TRUE

This is an existing *InnoDB* variable used to attempt flushing dirty pages in a way that avoids I/O bursts at checkpoints. In *XtraDB*, the default value of the variable is changed from that in *InnoDB*.

variable `innodb_adaptive_flushing_method`

Version Info

- ***`5.5.8-20.0`*** – Introduced

Command Line YES

Config File YES

Scope GLOBAL

Dynamic YES

Type STRING

Default Value `estimate`

Allowed Values `native`, `estimate`, `keep_average` (or 0/1/2, respectively, for compatibility)

This variable controls the way adaptive checkpointing is performed. *InnoDB* constantly flushes dirty blocks from the buffer pool. Normally, the checkpoint is done passively at the current oldest page modification (this is called "fuzzy checkpointing"). When the checkpoint age nears the maximum checkpoint age (determined by the total length of all transaction log files), *InnoDB* tries to keep the checkpoint age away from the maximum by flushing many dirty blocks. But, if there are many updates per second and many blocks have almost the same modification age, the huge number of flushes can cause stalls.

Adaptive checkpointing forces a constant flushing activity at a rate of approximately [modified age / maximum checkpoint age]. This can avoid or soften the impact of stalls casued by aggressive flushing.

The following values are allowed:

- `native` [0]: This setting causes checkpointing to operate exactly as it does in native *InnoDB*.
- `estimate` [1]: If the oldest modified age exceeds 1/4 of the maximum age capacity, *InnoDB* starts flushing blocks every second. The number of blocks flushed is determined by [number of modified blocks], [LSN progress speed] and [average age of all modified blocks]. So, this behavior is independent of the `innodb_io_capacity` variable for the 1-second loop, but the variable still has an effect for the 10-second loop.
- `keep_average` [2]: This method attempts to keep the I/O rate constant by using a much shorter loop cycle (0.1 second) than that of the other methods (1.0 second). It is designed for use with SSD cards.
- `reflex`: This behavior is similar to innodb_max_dirty_pages_pct flushing. The difference is that this method starts flushing blocks constantly and contiguously based on the oldest modified age. If the age exceeds 1/2 of the maximum age capacity, *InnoDB* starts weak contiguous flushing. If the age exceeds 3/4, *InnoDB* starts strong flushing. The strength can be adjusted by the *MySQL* variable `innodb_io_capacity`. In other words, we must tune `innodb_io_capacity` for the `reflex` method to work the best. This method was removed in `5.5.20-beta` as a fix for bug #689450.

variable `innodb_checkpoint_age_target`

Command Line Yes

Config File Yes

Scope GLOBAL

Dynamic Yes

Variable Type Numeric

Default Value 0

Range 0+

This variable controls the maximum value of the checkpoint age if its value is different from 0. If the value is equal to 0, it has no effect.

It is not needed to shrink `innodb_log_file_size` to tune recovery time.

variable `innodb_flush_method`

Command Line Yes

Config File Yes

Scope Global

Dyn No

Variable Type Enumeration

Default Value `fdatasync`

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

This is an existing *MySQL* 5.5 system variable. It determines the method *InnoDB* uses to flush its data and log files. (See `innodb_flush_method` in the *MySQL* 5.5 Reference Manual).

The following values are allowed:

- `fdatasync`: use fsync() to flush both the data and log files.
- `O_SYNC`: use O_SYNC to open and flush the log files; use fsync() to flush the data files.
- `O_DIRECT`: use O_DIRECT to open the data files and fsync() system call to flush both the data and log files.
- `ALL_O_DIRECT`: use O_DIRECT to open both data and log files, and use fsync() to flush the data files but it is skipped for all log files writes. This option is recommended when *InnoDB* log files are big (more than 8GB), otherwise there might be even a performance degradation. **Note**: When using this option on *ext4* filesystem variable *`innodb_log_block_size`* should be set to 4096 (default log-block-size in *ext4*) in order to avoid the `unaligned AIO/DIO` warnings.

variable `innodb_flush_neighbor_pages`

Version Info

- ***`5.5.19-24.0`*** – Introduced option value `cont`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Enumeration

Default Value `area`

Range `none`, `area`, `cont`

This variable specifies whether, when the dirty pages are flushed to the data file, the neighbor pages in the data file are also flushed at the same time or not. The following values (and their numeric counterparts `0`, `1` and `2` for older patch compatibility) are available:

- `none`: disables the feature.

- `area` (default): enables flushing of non-contiguous neighbor pages. For each page that is about to be flushed, look into its vicinity for other dirty pages and flush them too. This value implements the standard *InnoDB* behavior. If you use a storage which has no "head seek delay" (e.g. SSD or enough memory for write buffering), `none` or `cont` may show better performance.
- `cont`: enable flushing of contiguous neighbor pages. For each page that is about to be flushed, look if there is a contiguous block of dirty pages surrounding it. If such block is found it is flushed in a sequential I/O operation as opposed to several random I/Os if `area` is used.

variable `innodb_read_ahead`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type String

Default Value `linear`

Allowed Values `none`, `random` (*), `linear`, `both`

This variable controls the read-ahead algorithm of *InnoDB*. The following values are available:

- `none`: disables read-ahead
- `random`: if enough pages within the same extent are in the buffer pool, *InnoDB* will automatically fetch the remaining pages (an extent consists of 64 consecutive pages)
- `linear` (default): if enough pages within the same extent are accessed sequentially, *InnoDB* will automatically fetch the remaining pages
- `both`: enable both `random` and `linear` algorithms.

You can also control the threshold from which *InnoDB* will perform a read ahead request with the innodb_read_ahead_threshold variable

(*) `random` is removed from *InnoDB* Plugin 1.0.5, *XtraDB* ignores it after 1.0.5.

variable `innodb_use_global_flush_log_at_trx_commit`

Version Info

- ***`5.5.8-20.0`*** – Introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Type Boolean

Default Value True

Range True/False

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

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

```
SET innodb_use_global_flush_log_at_trx_commit=N
```

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

variable `innodb_log_block_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 512

Units Bytes

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

variable `innodb_log_file_size`

Version Info

- ***`5.5.8-20.0`*** – Introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Type Numeric

Default Value 5242880

Range 1048576 .. 4294967295

In upstream *MySQL* the limit for the combined size of log files must be less than 4GB. But in Percona Server it is:

- on 32-bit systems: individual log file limit is 4 GB and total log file size limit is 4 GB, i.e. the same as in the upstream server.
- on 64-bit systems: both individual log files and total log file size are practically unlimited (the limit is 2^63 - 1 bytes which is 8+ million TB).

variable `innodb_purge_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Type Numeric

Default Value 1

This variable is the same as the one in the upstream version. The only difference is the default value, in *Percona Server* it is `1` while in the upstream version is `0`.

8.2.1 Status Variables

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

```
The max checkpoint age
The current checkpoint age target
The current age of the oldest page modification which has not been flushed to disk␣
↪yet.
The current age of the last checkpoint
...
---
LOG
---
Log sequence number 0 1059494372
Log flushed up to   0 1059494372
Last checkpoint at  0 1055251010
Max checkpoint age  162361775
Checkpoint age target 104630090
Modified age        4092465
Checkpoint age      4243362
0 pending log writes, 0 pending chkp writes
...
```

8.3 Other Reading

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

CHAPTER

NINE

MULTIPLE ADAPTIVE HASH SEARCH PARTITIONS

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

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

Version Specific Information

Percona Server Version Comments 5.5.8-20.0 Initially released.

9.1 System Variables

variable `innodb_adaptive_hash_index_partitions`

Version Info

- *5.5.8-20.0* – Introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Def 1

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

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

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

9.2 Other reading

- Index lock and adaptive search

Part IV

Performance Improvements

CHAPTER

TEN

ATOMIC WRITE SUPPORT FOR FUSION-IO DEVICES

Note: This feature implementation is considered BETA quality.

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

10.1 Version Specific Information

- ***5.5.31-30.3*** `Atomic write support for Fusion-io` feature implemented. This feature was ported from *MariaDB*.

10.2 System Variables

variable `innodb_use_atomic_writes`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Default Value 0 (OFF)

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

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

10.3 Other Reading

- For general *InnoDB* tuning innodb_io_page documentation is available.
- FusionIO DirectFS atomic write support in *MariaDB*
- Atomic Writes Accelerate MySQL Performance

CHAPTER

ELEVEN

DROP TABLE PERFORMANCE

Warning: This feature has been removed and its controlling variable *innodb_lazy_drop_table* has been deprecated from *Percona Server 5.5.30-30.2*. Feature has been removed because the upstream DROP TABLE implementation has been improved.

When *innodb_file_per_table* is set to 1, doing a DROP TABLE can take a long time on servers with a large buffer pool, even on an empty *InnoDB* table. This is because InnoDB has to scan through the buffer pool to purge pages that belong to the corresponding tablespace. Furthermore, no other queries can start while that scan is in progress.

This feature allows you to do "background table drop".

11.1 Version Specific Information

- *5.5.10-20.1* Feature added.
- *5.5.30-30.2* Feature deprecated.

11.2 System Variables

variable innodb_lazy_drop_table

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type BOOL

Default Value FALSE

Range TRUE/FALSE

When this option is ON, XtraDB optimizes that process by only marking the pages corresponding to the tablespace being deleted. It defers the actual work of evicting those pages until it needs to find some free pages in the buffer pool.

When this option is OFF, the usual behavior for dropping tables is in effect.

11.3 Related Reading

- Drop table performance blog post.

CHAPTER

TWELVE

CONFIGURATION OF THE DOUBLEWRITE BUFFER

InnoDB and *XtraDB* use a special feature called the doublewrite buffer to provide a strong guarantee against data corruption. The idea is to write the data to a sequential log in the main tablespace before writing to the data files. If a partial page write happens (in other words, a corrupted write), *InnoDB* and *XtraDB* will use the buffer to recover the data. Even if the data is written twice the performance impact is usually small, but in some heavy workloads the doublewrite buffer becomes a bottleneck. Now we have an option to put the buffer on a dedicated disk in order to parallelize I/O activity on the buffer and on the tablespace.

This feature allows you to move the doublewrite buffer from the main tablespace to a separate location.

This option is for advanced users only. See the discussion below to fully understand whether you really need to use it.

12.1 Detailed Information

The following discussion will clarify the improvements made possible by this feature.

12.1.1 Goal of the Doublewrite Buffer

InnoDB and *XtraDB* use many structures, some on disk and others in memory, to manage data as efficiently as possible. To have an overview of the different components see this post. Let's now focus on the doublewrite buffer.

InnoDB / *XtraDB* uses a reserved area in its main tablespace, called the doublewrite buffer, to prevent data corruption that could occur with partial page writes. When the data in the buffer pool is flushed to disk, *InnoDB* / *XtraDB* will flush whole pages at a time (by default 16KB pages) and not just the records that have changed within a page. It means that, if anything unexpected happens during the write, the page can be partially written leading to corrupt data.

With the doublewrite buffer feature, *InnoDB* / *XtraDB* first writes the page in the doublewrite buffer and then to the data files.

If a partial page write occurs in the data files, *InnoDB* / *XtraDB* will check on recovery if the checksum of the page in the data file is different from the checksum of the page in the doublewrite buffer and thus will know if the page is corrupt or not. If it is corrupt, the recovery process will use the page stored in the doublewrite buffer to restore the correct data.

If a partial write occurs in the doublewrite buffer, the original page is untouched and can be used with the redo logs to recover the data.

12.1.2 Performance Impact of the Doublewrite Buffer

In usual workloads the performance impact is low-5% or so. As a consequence, you should always enable the doublewrite buffer because the strong guarantee against data corruption is worth the small performance drop.

But if you experience a heavy workload, especially if your data does not fit in the buffer pool, the writes in the doublewrite buffer will compete against the random reads to access the disk. In this case, you can see a sharp performance drop compared to the same workload without the doublewrite buffer-a 30% performance degradation is not uncommon.

Another case when you can see a big performance impact is when the doublewrite buffer is full. Then new writes must wait until entries in the doublewrite buffer are freed.

12.1.3 What's New with This Feature

In a standard *InnoDB* / *XtraDB* installation, the doublewrite buffer is located in the main tablespace (whether you activate the `innodb_file_per_table` or not) and you have no option to control anything about it.

The feature adds an option (`innodb_doublewrite_file`) to have a dedicated location for the doublewrite buffer.

12.1.4 How to Choose a Good Location for the Doublewrite Buffer

Basically if you want to improve the I/O activity, you will put the doublewrite buffer on a different disk. But is it better on an SSD or a more traditional HDD? First you should note that pages are written in a circular fashion in the doublewrite buffer and only read on recovery. So the doublewrite buffer performs mostly sequential writes and a few sequential reads. Second HDDs are very good at sequential write if a write cache is enabled, which is not the case of SSDs. Therefore you should choose a fast HDD if you want to see performance benefits from this option. For instance, you could place the redo logs (also written in a sequential way) and the doublewrite buffer on the same disk.

12.2 Version Specific Information

- *5.5.8-20.0* Full functionality available.

12.3 System Variables

The following system variable was introduced.

variable `innodb_doublewrite_file`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type STR

Def NULL

Use this option to create a dedicated tablespace for the doublewrite buffer.

This option expects a filename which can be specified either with an absolute or a relative path. A relative path is relative to the data directory.

12.4 Related Reading

- XtraDB / InnoDB internals in drawing
- InnoDB Double Write
- SSD and HDD for InnoDB

CHAPTER

THIRTEEN

QUERY CACHE ENHANCEMENTS

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

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

13.1 Diagnosing contention more easily

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

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

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

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

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

13.2 Ignoring comments

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

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

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

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

13.3 System Variables

variable `query_cache_strip_comments`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value Off

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

13.3.1 Other Reading

- MySQL general thread states
- Query cache freezes

CHAPTER

FOURTEEN

FAST *INNODB* CHECKSUM

Warning: This feature has been deprecated after *Percona Server* `5.5.28-29.2` and it will not be available in *Percona Server* 5.6, because the innodb_checksum_algorithm feature in *MySQL* 5.6 makes it redundant. If this feature was enabled, turning it off before the upgrade requires table(s) to be dumped and imported, since otherwise it will fail to start on data files created when `innodb_fast_checksum` was enabled. As an alternative you can use **innochecksum** from *MySQL* 5.7 as described in this blogpost.

InnoDB writes a checksum at the end of each data page in order to detect data files corruption. However computing this checksum requires CPU cycles and in some circumstances this extra overhead can become significant.

XtraDB can use a more CPU-efficient algorithm, based on 4-byte words, which can be beneficial for some workloads (for instance write-heavy workloads on servers that can perform lots of IO).

The original algorithm is checked after the new one, so you can have data pages with old checksums and data pages with new checksums. However in this case, you may experience slow reads from pages having old checksums. If you want to have the entire benefit of this change, you will need to recreate all your *InnoDB* tables, for instance by dumping and reloading all *InnoDB* tables.

Once enabled, turning it off will require table(s) to be dump/imported, since it will fail to start on data files created when `innodb_fast_checksums` was enabled. In this case ALTER TABLE won't work due to its implementation.

14.1 System Variables

variable `innodb_fast_checksum`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type BOOL

Default Value 0

CHAPTER

FIFTEEN

REMOVE EXCESSIVE FUNCTION CALLS (FCNTL)

This change removes a bottleneck at the client/server protocol level for high concurrency workloads.

When reading a packet from a socket, the read can be performed either in non-blocking mode or in blocking mode. The non-blocking mode was originally chosen because it avoids the cost of setting up an alarm in case of a timeout: thus the first attempt to read is done in non-blocking mode, and only if it fails, the next attempts are done in blocking mode.

However, this behavior can hurt performance as the switch from non-blocking mode to blocking mode is expensive, requiring calls to the fcntl function and calls into the kernel.

The solution is to use socket timeouts, with the `SO_RCVTIMEO` and `SO_SNDTIMEO` options. This way, the timeouts are automatically handled by the operating system, which means that all reads can be done in blocking mode.

15.1 Other Reading

- fcntl Bottleneck
- Use of non-blocking mode for sockets limits performance

CHAPTER

SIXTEEN

REDUCED BUFFER POOL MUTEX CONTENTION

We removed `buffer_pool` mutex operations on counting blocks on LRU list where it is safe to delete. As drawback we may have some inaccurate information of LRU list, but it does not affect storage engine operations. As result we have decreased contention on `buffer_pool` mutex.

CHAPTER

SEVENTEEN

INNODB TIMER-BASED CONCURRENCY THROTTLING

If the variable *innodb_thread_concurrency_timer_based* has been set to TRUE, lock-free timer-based *InnoDB* method of handling thread concurrency will be used instead of original mutex-based method.

17.1 System Variables

variable innodb_thread_concurrency_timer_based

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type BOOL

Default Value FALSE

Range TRUE/FALSE

Note: This feature depends on atomic op builtins being available.

CHAPTER

EIGHTEEN

IMPROVED NUMA SUPPORT

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

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

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

18.1 Version Specific Information

- ***5.5.28-29.1*** Improved `NUMA` support implemented. This feature was ported from Twitter's *MySQL* patches.

18.2 System Variables

variable innodb_buffer_pool_populate

Command Line Yes

Config File Yes

Location mysqld

Scope Global

Dynamic No

Variable Type Boolean

Default Value 0

Range 0/1

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

18.3 Command-line Options for mysqld_safe

variable `flush_caches`

Command Line Yes

Config File Yes

Location mysqld_safe

Dynamic No

Variable Type Boolean

Default Value 0

Range 0/1

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

variable `numa_interleave`

Command Line Yes

Config File Yes

Location mysqld_safe

Dynamic No

Variable Type Boolean

Default Value 0

Range 0/1

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

18.4 Other Reading

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

CHAPTER

NINETEEN

HANDLERSOCKET

19.1 Description

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

Complete documentation on the *HandlerSocket* plugin, including installation and configuration options, is located here.

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

19.2 Version Specific Information

- *5.5.11-20.2* Full functionality available.

Other Information

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

19.3 Enabling the Plugin

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

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

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

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

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

19.4 Testing the Plugin installation

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

```
mysql> SHOW PROCESSLIST;
+----+-------------+-----------------+---------------+---------+------+---------------
↪-----------------------------+------------------+
| Id | User        | Host            | db            | Command | Time | State         ␣
↪                            | Info             |
+----+-------------+-----------------+---------------+---------+------+---------------
↪-----------------------------+------------------+
|  1 | system user | connecting host | NULL          | Connect | NULL |␣
↪handlersocket: mode=rd, 0 conns, 0 active | NULL             |
|  2 | system user | connecting host | NULL          | Connect | NULL |␣
↪handlersocket: mode=rd, 0 conns, 0 active | NULL             |
...
| 16 | system user | connecting host | NULL          | Connect | NULL |␣
↪handlersocket: mode=rd, 0 conns, 0 active | NULL             |
| 17 | system user | connecting host | handlersocket | Connect | NULL |␣
↪handlersocket: mode=wr, 0 conns, 0 active | NULL             |
```

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

Create a new table:

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

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

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

And check in SQL that the row has been written:

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

19.4.1 Configuration options

HandlerSocket has many configuration options that are detailed here.

19.5 Other Reading

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

CHAPTER

TWENTY

FIXED SIZE FOR THE READ AHEAD AREA

InnoDB dynamically calculates the size of the read-ahead area in case it has to trigger its read-ahead algorithm. When the workload involves heavy I/O operations, this size is computed so frequently that it has a non-negligeable impact on the CPU usage.

This variable only depends on the size of the buffer pool set by the `innodb_buffer_pool_size` variable, and as soon as the buffer pool has a size properly greater than 1024 pages (or 16 MB), it is always 64. With this change, its value is fixed to 64, thus removing a bottleneck experienced by some users.

Please note that the minimum allowed value for the *InnoDB* buffer pool is de facto set to 32 MB.

This change is a port of the feature from Facebook:

- http://bazaar.launchpad.net/~mysqlatfacebook/mysqlatfacebook/5.1/revision/3538

20.1 Version Specific Information

- *`5.5.8-20.0`* : Full functionality available.

20.2 Other Information

- Author/Origin: Facebook
- Bugs fixed: #606811

20.3 Other Reading

- BUF_READ_AHEAD_AREA Bottleneck

CHAPTER

TWENTYONE

THREAD POOL

Note: The *thread pool* feature implementation is considered BETA quality.

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

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

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

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

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

Important: The current implementation of the *thread pool* feature is built into the server, unlike the upstream version which is implemented as a plugin. Another significant implementation difference is that this implementation doesn't try to minimize the number of concurrent transactions like the `MySQL Enterprise Threadpool`. Because of these differences, this implementation is not compatible with the upstream implementation.

21.1 Priority connection scheduling

The priority connection scheduling was implemented for *thread pool* in *Percona Server* `5.5.30-30.2`. 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. A higher number of open transactions has a number of implications on the currently running queries. To improve the performance a new variable *`thread_pool_high_prio_tickets`* was 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 both following conditions apply:

1. The connection has an open transaction in the server.
2. The number of high priority tickets of this connection is non-zero.

If these 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 then picks connections from the common queue only when the high priority queue is empty.

The goal is to minimize the number of open transactions on 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 are that each transaction enters the high priority queue and is committed before it is put into 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 a new variable *thread_pool_high_prio_mode* was been introduced in *Percona Server* *5.5.35-33.0*.

21.1.1 Low priority queue throttling

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

In this case, those threads in the pool that have marked themselves inactive are not accounted to the oversubscribe limit. As a result, the number of threads (both active and waiting) in the pool grows until it hits *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.

21.2 Version Specific Information

- ***5.5.29-30.0*** `Thread Pool` feature implemented. This feature was ported from *MariaDB*.
- ***5.5.30-30.2*** Implemented priority connection scheduling and introduced new variable *thread_pool_high_prio_tickets* to the original implementation introduced in *MariaDB*.
- ***5.5.35-33.0*** Default value for *thread_pool_max_threads* was changed from `500` to `100 000`. This change was introduced because limiting the total number of threads in the *Thread Pool* can result in deadlocks and uneven distribution of worker threads between thread groups in case of stalled connections.
- ***5.5.35-33.0*** Default value for *thread_pool_high_prio_tickets* was changed from `0` to `4294967295` to enable the priority connection scheduling by default.
- ***5.5.35-33.0*** Implemented new *thread_pool_high_prio_mode* variable.

- *5.5.35-33.0* Implemented *Low priority queue throttling*.

21.3 System Variables

variable thread_pool_idle_timeout

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 60 (seconds)

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

variable thread_pool_high_prio_mode

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Variable Type String

Default Value `transactions`

Allowed Values `transactions`, `statements`, `none`

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

The following values are allowed:

- `transactions` (the default). In this mode, only statements from already started transactions may go into the high priority queue depending on the number of high priority tickets currently available in a connection (see *thread_pool_high_prio_tickets*).
- `statements`. In this mode, all individual statements go into the high priority queue, regardless of connection's transactional state and the number of available high priority tickets. This value can be used to prioritize `AUTOCOMMIT` transactions or other kinds of statements such as administrative ones for specific connections. Note that setting this value globally essentially disables high priority scheduling, since in this case all statements from all connections will use a single queue (the high priority one)
- `none`. This mode disables high priority queue for a connection. Some connections (e.g. monitoring) may be insensitive to execution latency and/or never allocate any server resources that would otherwise impact performance in other connections and thus, do not really require high priority scheduling. Note that setting *thread_pool_high_prio_mode* to `none` globally has essentially the same effect as setting it to `statements` globally: all connections will always use a single queue (the low priority one in this case).

variable thread_pool_high_prio_tickets

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Variable Type Numeric

Default Value 4294967295

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

variable `thread_pool_max_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 100000

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

variable `thread_pool_oversubscribe`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 3

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

variable `thread_pool_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value Number of processors

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

variable `thread_pool_stall_limit`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 500 (ms)

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

variable `extra_port`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 0

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

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

variable `extra_max_connections`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 1

This variable can be used to specify the maximum allowed number of connections 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.

21.4 Status Variables

variable `Threadpool_idle_threads`

Command Line Yes

Variable Type Numeric

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

variable `Threadpool_threads`

Command Line Yes

Variable Type Numeric

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

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.

21.5 Other Reading

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

CHAPTER

TWENTYTWO

BINARY LOG GROUP COMMIT

In cases when strict durability and recoverability is required and the storage that provides fast syncs is unavailable, setting up the variables `innodb_flush_log_at_trx_commit` =1 and `sync_binlog` =1 can result in big performance drop.

Note: Variable `innodb_flush_log_at_trx_commit` makes sure that every transaction is written to the disk and that it can survive the server crash. In case the binary log is used for replication `sync_binlog` makes sure that every transaction written to the binary log matches the one executed in the storage engine. More information about these variables can be found in the *MySQL* documentation.

Performance drop happening when these variables are enabled is caused by additional *fsync()* system calls on both binary and *XtraDB* `REDO` log when committing a transaction, that are needed to store the additional information on the disk. `Binary Log Group Commit` feature can use a single *fsync()* call to force data to the storage for multiple concurrently committing transactions, which provides throughput improvements in a write-concurrent workload.

Because there are no negative effects of this feature, it has been enabled by default and can't be disabled. Effects of this feature can be measured by the *`binlog_commits`* and *`binlog_group_commits`* status variables. The bigger the difference between these two variables the bigger is the performance gained with this feature.

22.1 Version Specific Information

- ***`5.5.18-23.0`*** Ported *MariaDB* Group commit for the binary log patch

22.2 Status Variables

variable `binlog_commits`

Command Line Yes

Scope Session

Variable Type Numeric

This variable shows the total number of transactions committed to the binary log.

variable `binlog_group_commits`

Command Line Yes

Scope Session

Variable Type Numeric

This variable shows the total number of group commits done to the binary log.

22.3 Other Reading

- Testing the Group Commit Fix
- Fixing MySQL group commit

Part V

Flexibility Improvements

CHAPTER

TWENTYTHREE

SUPPORT OF MULTIPLE PAGE SIZES

Warning: This feature has been deprecated in the *Percona Server* `5.5.30-30.2`. It has been replaced by the upstream version released in *MySQL* 5.6.4.

Percona Server has implemented support for multiple *InnoDB* page sizes. This can be used to increase the IO performance by setting this value close to storage device block size. *InnoDB* page size can be set up with the `innodb_page_size` variable.

23.1 System Variables

variable `innodb_page_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type ULONG

Default Value 16384

Range 4096, 8192, 16384

EXPERIMENTAL: The universal page size of the database. Changing for an existing database is not supported. Use at your own risk!

CHAPTER

TWENTYFOUR

SUPPRESS WARNING MESSAGES

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

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

24.1 Version Specific Information

- *5.5.8-20.0*: System variable `log_warnings_silence` introduced.
- *5.5.10-20.1*: Renamed variable `log_warnings_silence` to *log_warnings_suppress*.

24.2 System Variables

variable `log_warnings_suppress`

Version Info

- ***5.5.8-20.0*** – Introduced.
- ***5.5.10-20.1*** – Renamed.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type SET

Default Value `(empty string)`

Range `(empty string),1592`

This variable was added in beta release `5.5.8-20.0` as `log_warnings_silence` and renamed in release 5.5.10-20.1.

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.

24.3 Related Reading

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

CHAPTER

TWENTYFIVE

HANDLE `BLOB` END OF LINE

At some point in the past, the *MySQL* command line client was modified to remove `\r` before `\n` in its input.

This caused problems in some workloads, specifically when loading `BLOB` fields containing `\r` characters. *Percona Server* solves this by implementing a new command line client option, *no-remove-eol-carret*.

When the *no-remove-eol-carret* option is specified, `\r` before `\n` is not removed.

25.1 Version Specific Information

- *5.5.8-20.0*: Full functionality.

25.2 Client Command Line Parameter

variable no-remove-eol-carret

Command Line Yes

Config File Yes

Scope Local

Dynamic No

Variable Type Boolean

Default Value Off

Range On/Off

CHAPTER

TWENTYSIX

FAST SHUTDOWN

Some *InnoDB* / *XtraDB* threads which perform various background activities are in the sleep state most of the time. They only wake up every few seconds to perform their tasks. They also check whether the server is in the shutdown phase, and if not, they go to the sleep state again. That means there could be a noticeable delay (up to 10 seconds) after a shutdown command and before all *InnoDB* / *XtraDB* threads actually notice this and terminate. This is not a big problem for most production servers, because a shutdown of a heavily loaded server normally takes much longer than 10 seconds.

The problem, however, had a significant impact on running the regression test suite, because it performs a lot of server restarts during its execution and also because there is not so much to do when shutting a test server. So it makes even less sense to wait up to 10 seconds.

This change modifies that behavior to make the sleep waiting interruptible, so that when the server is told to shutdown, threads no longer wait until the end of their sleep interval. This results in a measurably faster test suite execution (~40% in some cases).

The change was contributed by Kristian Nielsen.

26.1 Version Specific Information

- `5.5.8-20.0` Full functionality available.

26.2 Other Information

- Author / Origin: Kristian Nielsen
- Bugs fixed: #643463

26.3 Other reading

- How to decrease InnoDB shutdown times
- How long InnoDB shutdown may take

CHAPTER

TWENTYSEVEN

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.

27.1 Caveats

27.1.1 Ordering of Rows

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

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

27.1.2 Indexing

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

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

27.2 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.

27.2.1 Ordering of Columns

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

27.2.2 Minimum Block Size

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

27.3 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.

27.4 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.

27.4.1 Implicit Request

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

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

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

While this will not:

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

27.4.2 Explicit Request

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

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

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

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

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

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

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

27.5 Examples

On a 32-bit platform:

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

mysql> SHOW TABLE STATUS LIKE 't1';
Name  Engine  Version Row_format      Rows    Avg_row_length  Data_length     Max_
↪data_length Index_length    Data_free       Auto_increment  Create_time     Update_
↪time     Check_time      Collation       Checksum        Create_options  Comment
t1    MEMORY  10      Dynamic 0       X       0       X       0       0       NULL    ␣
↪NULL    NULL    NULL    latin1_swedish_ci       NULL    row_format=DYNAMIC KEY_
↪BLOCK_SIZE=124
```

On a 64-bit platform:

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

```
mysql> SHOW TABLE STATUS LIKE 't1';
Name  Engine  Version Row_format      Rows    Avg_row_length  Data_length     Max_
↪data_length Index_length    Data_free       Auto_increment  Create_time     Update_
↪time     Check_time      Collation       Checksum        Create_options  Comment
t1    MEMORY  10      Fixed   0       X       0       X       0       0       NULL
↪NULL    NULL    NULL    latin1_swedish_ci       NULL    row_format=DYNAMIC KEY_
↪BLOCK_SIZE=124
```

27.6 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*)`.

27.7 See Also

- Dynamic row format for MEMORY tables

CHAPTER

TWENTYEIGHT

RESTRICTING THE NUMBER OF BINLOG FILES

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

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

28.1 Version Specific Information

- 5.5.27-29.0: Variable *max_binlog_files* introduced.

28.2 System Variables

variable max_binlog_files

Version Info

- ***5.5.27-29.0*** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 0 (unlimited)

Range 0-102400

28.3 Example

Number of the binlog files before setting this variable

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

Variable *max_binlog_files* is set to 20:

```
max_binlog_files = 20
```

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

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

CHAPTER

TWENTYNINE

IGNORING MISSING TABLES IN MYSQLDUMP

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

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

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

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

29.1 Version Specific Information

- *5.5.8-20.0* **mysqldump** option `--ignore-create-error` introduced

CHAPTER

THIRTY

EXTENDED `SELECT INTO OUTFILE/DUMPFILE`

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

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

30.1 Version Specific Information

- *5.5.34-32.0* - Feature Implemented

30.2 Other Reading

- *MySQL* bug: #44835

CHAPTER

THIRTYONE

EXTENDED MYSQLBINLOG

Percona Server has implemented compression support for **mysqlbinlog** in *5.5.35-33.0*. This is similar to support that both `mysql` and `mysqldump` programs include (the `-C`, `--compress` options "Use compression in server/client protocol"). Using the compressed protocol helps reduce the bandwidth use and speed up transfers.

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

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

31.1 Version Specific Information

- *5.5.35-33.0* **mysqlbinlog** option `--compress` introduced
- *5.5.35-33.0* **mysqlbinlog** now has all SSL connection options as the rest of the *MySQL* client programs.

CHAPTER

THIRTYTWO

ABILITY TO CHANGE DATABASE FOR MYSQLBINLOG

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

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

Option *rewrite-db* of **mysqlbinlog** utility allows to setup rewriting rule "from->"to".

32.1 Example

mysqlbinlog output before rewrite-db

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

mysqlbinlog output when the new variable is used:

```
$ mysqlbinlog --rewrite-db='world->new_world' mysql-bin.000005
...
# at 106
use new_world/*!*/;
#120517 13:10:00 server id 2  end_log_pos 175   Query   thread_id=88    exec_time=0
↪  error_code=0
SET TIMESTAMP=1337253000/*!*/;
SET @@session.pseudo_thread_id=88/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=1, @@session.unique_
↪checks=1, @@session.autocommit=1/*!*/;
```

```
SET @@session.sql_mode=0/*!*/;
SET @@session.auto_increment_increment=1, @@session.auto_increment_offset=1/*!*/;
/*!\C latin1 *//*!*/;
SET @@session.character_set_client=8,@@session.collation_connection=8,@@session.
↪collation_server=8/*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
BEGIN
/*!*/;
# at 175
#120517 13:10:00 server id 2  end_log_pos 203   Intvar
SET INSERT_ID=4083/*!*/;
# at 203
#120517 13:10:00 server id 2  end_log_pos 367   Query   thread_id=88    exec_time=0
↪  error_code=0
SET TIMESTAMP=1337253000/*!*/;
insert into City (Name, CountryCode, District, Population) values ("New City", "ZMB",
↪"TEX", 111000)
/*!*/;
# at 367
#120517 13:10:00 server id 2  end_log_pos 394   Xid = 1414
COMMIT/*!*/;
```

32.2 Version Specific Information

- `5.6.36-34.0` Full functionality.

32.3 Client Command Line Parameter

option `rewrite-db`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type String

Default Value Off

32.4 Related Reading

- WL #36

CHAPTER

THIRTYTHREE

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.

33.1 Version Specific Information

- *5.5.36-34.0*: Variable *max_slowlog_size* introduced.
- *5.5.36-34.0*: Variable *max_slowlog_files* introduced.

33.2 System Variables

variable max_slowlog_size

Version Info

- ***5.5.36-34.0*** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type numeric

Default Value 0 (unlimited)

Range 4096 - 1073741824

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

variable `max_slowlog_files`

Version Info

- *`5.5.36-34.0`* – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type numeric

Default Value 0 (unlimited)

Range 0 - 102400

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

CHAPTER

THIRTYFOUR

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.

34.1 Example

Table:

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

Following example shows the difference in parsing for default and `IETF_QUOTES` `csv_quotes`.

```
mysql> 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:

```
mysql> SELECT * FROM albums;
+--------------+--------------------+
| 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';
Query OK, 0 rows affected (0.00 sec)
```

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

34.2 Version Specific Information

- *5.5.41-37.0*: Variable *csv_mode* implemented

34.3 System Variables

variable csv_mode

Version Info

- ***5.5.41-37.0*** – Introduced.

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Variable Type SET

Default Value (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.

34.4 Related Reading

- MySQL bug #71091

CHAPTER

THIRTYFIVE

SUPPORT FOR TLS V1.1 AND V1.2

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

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

35.1 Version Specific Information

- *`5.5.50-38.0`*: Implemented support for TLS v1.1 and TLS v1.2 protocols

35.2 System Variables

variable `tls_version`

Version Info

- ***`5.5.50-38.0`*** – Introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type String

Default Value `TLSv1.1,TLSv1.2`

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

variable `have_tlsv1_2`

Version Info

- ***`5.5.50-38.0`*** – Introduced

Command Line Yes

Config File No

Scope Global

Dynamic No

Variable Type Boolean

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

CHAPTER

THIRTYSIX

BINLOGGING AND REPLICATION IMPROVEMENTS

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

36.1 Temporary tables and mixed logging format

36.1.1 Summary of the fix:

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

36.1.2 Version Specific Information

- `5.5.41-37.0` Fix backported from *Percona Server* 5.6.21-70.1

36.1.3 Details:

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

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

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

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

36.2 Safety of statements with a LIMIT clause

36.2.1 Summary of the fix:

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

36.2.2 Version Specific Information

- *5.5.41-37.0* Fix implemented in *Percona Server* 5.5

Part VI

Reliability Improvements

CHAPTER

THIRTYSEVEN

CRASH-RESISTANT REPLICATION

This feature makes replication much more reliable after a crash by making the replica's position relative to the master transactional.

MySQL replication normally stores its position in a file that is neither durable nor consistent. Thus, if the replica crashes, it can re-execute committed transactions. This usually causes replication to fail, potentially forcing the replica''s data to be re-initialized from the master or from a recent backup.

The improvement in *Percona Server* makes *InnoDB* store the replication position transactionally, and overwrite the usual relay_log.info file upon recovery, so replication restarts from the correct position and does not try to re-execute committed transactions. This change greatly improves the durability of *MySQL* replication. It can be set to activate automatically, so replication "just works" and no intervention is necessary after a crash.

37.1 Use

To enable the feature, the following options need to be enabled together: *innodb_recovery_update_relay_log*, and relay_log_recovery.

37.2 Restrictions

When both *innodb_recovery_update_relay_log* and relay_log_recovery are enabled, you should only update *InnoDB* / *XtraDB* tables, not *MyISAM* tables or other storage engines. You should not use relay or binary log filenames longer than 480 characters (normal: up to 512). If longer, the replication position information is not recorded in *InnoDB*.

37.3 Example Server Error Log Output

Upon crash recovery, the error log on a replica will show information similar to the following:

```
InnoDB: Starting crash recovery.
....
InnoDB: Apply batch completed
InnoDB: In a MySQL replication slave the last master binlog file
InnoDB: position 0 468, file name gauntlet3-bin.000015
InnoDB: and relay log file
InnoDB: position 0 617, file name ./gauntlet3-relay-bin.000111
```

If this feature is enabled, the output will look like the following, with additional lines prefixed with a + symbol:

```
....
+ InnoDB: Warning: innodb_recovery_update_relay_log is enabled. Updates of other␣
↪storage engines may have problem of consistency.
+ InnoDB: relay-log.info is detected.
+ InnoDB: relay log: position 429, file name ./gauntlet3-relay-bin.000111
+ InnoDB: master log: position 280, file name gauntlet3-bin.000015
....
  InnoDB: Starting crash recovery.
....
  InnoDB: Apply batch completed
+ InnoDB: In a MySQL replication slave the last master binlog file
+ InnoDB: position 0 468, file name gauntlet3-bin.000015
+ InnoDB: and relay log file
+ InnoDB: position 0 617, file name ./gauntlet3-relay-bin.000111
  090205 17:41:31 InnoDB Plugin 1.0.2-3 started; log sequence number 57933
+ InnoDB: relay-log.info have been overwritten.
....
  090205 17:41:31 [Note] Slave SQL thread initialized, starting replication in log␣
↪``gauntlet3-bin.000015`` at position 468, relay log ``./gauntlet3-relay-bin.
↪000111`` position: 617
```

In this case, the master log position was overwritten to 468 from 280, so replication will start at position 468 and not repeat the transaction beginning at 280.

37.4 Version Specific Information

- *5.5.10-20.1*: Renamed variable *innodb_overwrite_relay_log_info* to *innodb_recovery_update_relay_log*.

37.5 System Variables

One new system variable was introduced by this feature.

variable `innodb_overwrite_relay_log_info`

Version Info

- ***5.5.10-20.1*** – Renamed.

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type BOOLEAN

Default Value FALSE

Range TRUE/FALSE

If set to true, *InnoDB* overwrites `relay-log.info` at crash recovery when the information is different from the record in *InnoDB*.

This variable was renamed to *innodb_recovery_update_relay_log*, beginning in release 5.5.10-20.1. It still exists as *innodb_overwrite_relay_log_info* in versions prior to that.

variable `innodb_recovery_update_relay_log`

Version Info

- ***`5.5.10-20.1`*** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type BOOLEAN

Default Value FALSE

Range TRUE/FALSE

If set to true, *InnoDB* overwrites `relay-log.info` at crash recovery when the information is different from the record in *InnoDB*.

This variable was added in release 5.5.10-20.1. Prior to that, it was named *`innodb_overwrite_relay_log_info`*, which still exists in earlier versions.

37.6 Other Reading

- Another solution for *MySQL* 5.0 is Google's transactional replication feature, but it had some problems and bugs.
- Related bug (fixed and re-implemented in this feature)
- A blog post explaining how this feature makes replication more reliable

CHAPTER

THIRTYEIGHT

TOO MANY CONNECTIONS WARNING

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

38.1 Version-Specific Information

- *`5.5.8-20.0`*: Full functionality available.

CHAPTER

THIRTYNINE

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.

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, it may still be running 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.

39.1 Version Specific Information

- 5.5.10-20.1: Renamed variable *innodb_pass_corrupt_table* to *innodb_corrupt_table_action*.

39.2 System Variables

variable innodb_pass_corrupt_table

Version Info

- ***5.5.10-20.1*** – Renamed.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 0

Range 0 - 1

Return error `1194 (ER_CRASHED_ON_USAGE)` instead of crashing with an assertion failure, when used with `innodb_file_per_table`. Once corruption is detected, access to the corrupted tablespace is disabled. The only allowed operation on a corrupted tablespace is `DROP TABLE`. The only exception to this rule is when the option value is `salvage` (see below). This variable was renamed to *`innodb_corrupt_table_action`*, beginning in release *`5.5.10-20.1`*. The option name was *`innodb_pass_corrupt_table`* in versions prior to that.

variable `innodb_corrupt_table_action`

Version Info

- ***`5.5.10-20.1`*** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ENUM

Default Value `assert`

Range `assert`, `warn`, `salvage`

- With the default value *XtraDB* will intentionally crash the server with an assertion failure as it would normally do when detecting corrupted data in a single-table tablespace.
- If the `warn` value is used it will pass corruption of the table as `corrupt table` instead of crashing itself. For this to work `innodb_file_per_table` should be enabled. All file I/O for the datafile after detected as corrupt is disabled, except for the deletion.
- When the option value is `salvage`, *XtraDB* allows read access to a corrupted tablespace, but ignores corrupted pages".

This variable was added in release *`5.5.10-20.1`*. Prior to that, it was named *`innodb_pass_corrupt_table`*, which still exists in earlier versions.

CHAPTER

FORTY

LOCK-FREE `SHOW SLAVE STATUS`

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

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

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

```
SHOW SLAVE STATUS NOLOCK
```

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

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

40.1 Status Variables

variable `Com_show_slave_status_nolock`

Variable Type Numeric

Scope Global/Session

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

40.2 Version Specific Information

- *`5.5.8-20.0`*: Introduced

Part VII

Management Improvements

CHAPTER

FORTYONE

INNODB RECOVERY STATS

When the variable *innodb_recovery_stats* is enabled and XtraDB has to do recovery on startup, server will write detailed recovery statistics information to the error log. This info will be written after the recovery process is finished.

Example of output statistics for recovery process:

```
-------------------
RECOVERY STATISTICS
-------------------
Recovery time: 18 sec. (1 turns)

Data page IO statistics
  Requested pages: 9126
  Read pages:      9126
  Written pages:   7957
  (Dirty blocks):  1156
  Grouping IO [times]:
        number of pages,
                read request neighbors (in 32 pages chunk),
                        combined read IO,
                                combined write IO
          1,    32,     335,    548
          2,    0,      121,    97
          3,    7,      49,     44
          4,    4,      43,     26
....
         64,    0,      2,      25

Recovery process statistics
  Checked pages by doublewrite buffer: 128
  Overwritten pages from doublewrite:  0
  Recovered pages by io_thread:        9145
  Recovered pages by main thread:      0
  Parsed log records to apply:         2572491
            Sum of the length:         71274689
  Applied log records:                 2376356
            Sum of the length:         68098300
  Pages which are already new enough:  93
  Oldest page's LSN:                   926917970
  Newest page's LSN:                   1526578232
```

41.1 System Variables

variable `innodb_recovery_stats`

Command Line No

Config File Yes

Dynamic No

Variable Type BOOL

Default Value FALSE

Range TRUE/FALSE

41.2 Other reading

- How to estimate time it takes InnoDB to recover?
- InnoDB recovery - is large buffer pool always better?
- What is the longest part of InnoDB recovery process?
- Improving InnoDB recovery time
- How long is recovery from 8G innodb_log_file

CHAPTER

FORTYTWO

INNODB DATA DICTIONARY SIZE LIMIT

This feature lets users limit the amount of memory used for *InnoDB* 's data dictionary. It was introduced in release 5.0.77-b13 of *Percona Server* with *XtraDB*.

The data dictionary is *InnoDB* 's internal catalog of tables. *InnoDB* stores the data dictionary on disk, and loads entries into memory while the server is running. This is somewhat analogous to *MySQL* 's table cache, but instead of operating at the server level, it is internal to the *InnoDB* storage engine. This feature permits you to control how *InnoDB* manages the data dictionary in memory, but does not modify on-disk storage.

In standard *InnoDB*, the size of the data dictionary depends on the number and size of tables opened in the server. Once a table is opened, it is never removed from the data dictionary unless you drop the table or you restart the server. In some cases, the data dictionary grows extremely large. If this consumes enough memory, the server will begin to use virtual memory. Use of virtual memory can cause swapping, and swapping can cause severe performance degradation. By providing a way to set an upper limit to the amount of memory the data dictionary can occupy, this feature provides users a way to create a more predictable and controllable situation.

If your data dictionary is taking up more than a gigabyte or so of memory, you may benefit from this feature. A data dictionary of this size normally occurs when you have many tens of thousands of tables. For servers on which tables are accessed little by little over a significant portion of time, memory usage will grow steadily over time, as if there is a memory leak. For servers that access every table fairly soon after being started, memory usage will increase quickly and then stabilize.

Please note that this variable only sets a soft limit on the memory consumed by the data dictionary. In some cases, memory usage will exceed the limit (see "Implementation Details" below for more).

You can see the actual size of the data dictionary by running the `SHOW ENGINE INNODB STATUS` command. Data dictionary size will be shown under `BUFFER POOL AND MEMORY` section:

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total memory allocated 137756672; in additional pool allocated 0
Total memory allocated by read views 88
Internal hash tables (constant factor + variable factor)
   Adaptive hash index 2250352         (2213368 + 36984)
   Page hash           139112 (buffer pool 0 only)
   Dictionary cache    612843  (554768 + 58075)
   File system         83536   (82672 + 864)
   Lock system         333248  (332872 + 376)
   Recovery system     0       (0 + 0)
Dictionary memory allocated 58075
Buffer pool size        8191
```

42.1 System Variables

The following system variable was introduced by this feature.

variable `innodb_dict_size_limit`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 0

Range 0-LONG_MAX

Units Bytes

This variable places a soft upper bound on the amount of memory used by tables in the data dictionary. When the allocated memory exceeds this amount, *InnoDB* tries to remove some unused entries, if possible. The default value of 0 indicates an unlimited amount of memory and results in the same behavior as standard *InnoDB*.

42.2 Status Variables

The following status variable was introduced by this feature:

variable `Innodb_dict_tables`

Variable Type LONG

Scope Global

This status variable shows the number of entries in the *InnoDB* data dictionary cache.

42.3 Choosing a Good Value

As a rough guide, a server that is likely to run into problems with an oversized data dictionary is probably a powerful machine with a lot of memory, perhaps 48GB or more. A gigabyte seems like a comfortable upper limit on the data dictionary for such a server, but this is a matter of opinion and you should choose a value that makes sense to you.

You might find it helpful to understand how much memory each table requires in the dictionary. Quick tests on a 32-bit server show that the data dictionary requires a minimum of about 1712 bytes per table, plus 288 bytes per column, and about 570 bytes for each index. The number might be higher on a 64-bit server due to the increased size of pointers.

Please do not rely on these rules of thumb as absolute truth. We do not know an exact formula for the memory consumption, and would appreciate your input if you investigate it more deeply.

42.4 Implementation Details

This feature tries to remove the least recently used *InnoDB* tables from the data dictionary. To achieve this, we need to sort entries in the dictionary in a LRU fashion and to know whether the table is used by the server. The first part is provided by an existing LRU algorithm in *InnoDB*. To determine whether the server is using a table, we check the

server‘‘s table cache for the second part. If a table is in the table cache, it is considered to be in use by the server, and is kept in the dictionary. If it is not in the table cache, it can be removed from the dictionary.

Unfortunately, the table cache is not always an accurate way to know whether the table is used by *MySQL* or not. Tables that are in the table cache might not really be in use, so if you have a big table cache, the algorithm will only be able to remove some of the items in the dictionary, which means that the memory consumed by the dictionary may exceed the value of `innodb_dict_size_limit`. This is why we said this variable sets a soft limit on the size of the dictionary, not an absolute limit.

42.5 Other reading

- Limiting InnoDB data dictionary
- How much memory InnoDB dictionary can take

CHAPTER

FORTYTHREE

EXPAND TABLE IMPORT

Unlike MyISAM, *InnoDB* does not allow users to copy datafiles for a single table between servers. If exported with XtraBackup, a table can now be imported on another server running *XtraDB*.

This feature implements the abililty to import arbitrary .ibd files exported using the XtraBackup `--export` option. The *innodb_expand_import* variable makes to convert `.ibd` file during import process.

The normal version can import only the backed-up .ibd file at the same place.

Note: This feature is unsupported with InnoDB data files created with MySQL 5.0 and MySQL 5.1 prior to version 5.1.7 due to InnoDB file format limitation. It may work in some cases, but may result in crashes on import as well, see bug #1000221 and bug #727704 for examples and details.

Percona Server 5.5.28-29.2 extended the `innochecksum` with an option `-f` to read the file format information from a given *InnoDB* data file. As only the first page needs to be read to detect the format/version information, it can also be used on a running server. Example of the output should look like this:

```
$ innochecksum -f ibdata1
Detected file format: Antelope (5.1.7 or newer).
```

43.1 Example

Assuming that:

- *innodb_expand_import* is set to `1`.
- the files (`.ibd` and `.exp`) are prepared by the `xtrabackup --prepare --export` command.

First create "exactly same" structured tables to the target database.

Then discard the tables as preparation of import, for example,

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

mysql> alter table customer discard tablespace;
Query OK, 0 rows affected (0.01 sec)

mysql> alter table district discard tablespace;
Query OK, 0 rows affected (0.01 sec)

mysql> alter table history discard tablespace;
Query OK, 0 rows affected (0.00 sec)
```

```
...
put the .ibd and .exp files at the same place to .frm file.
import the tables
(command example)
mysql> set FOREIGN_KEY_CHECKS=0;
Query OK, 0 rows affected (0.00 sec)

mysql> set global innodb_expand_import=1;
Query OK, 0 rows affected (0.00 sec)

mysql> alter table customer import tablespace;
Query OK, 0 rows affected (0.17 sec)

mysql> alter table district import tablespace;
Query OK, 0 rows affected (0.00 sec)

mysql> alter table history import tablespace;
Query OK, 0 rows affected (0.04 sec)

...
(.err file example)
InnoDB: import: extended import of tpcc2/customer is started.
InnoDB: import: 2 indexes are detected.
InnoDB: Progress in %: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24␣
↪25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52␣
↪53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80␣
↪81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 done.
InnoDB: import: extended import of tpcc2/district is started.
InnoDB: import: 1 indexes are detected.
InnoDB: Progress in %: 16 33 50 66 83 100 done.
InnoDB: import: extended import of tpcc2/history is started.
InnoDB: import: 3 indexes are detected.
InnoDB: Progress in %: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24␣
↪25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52␣
↪53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80␣
↪81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 done.
...
```

43.2 Version Specific Information

- 5.5.10-20.1: Renamed variable *innodb_expand_import* to *innodb_import_table_from_xtrabackup*.

43.3 System Variables

variable innodb_expand_import

Version Info

- ***5.5.10-20.1*** – Renamed.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 0

Range 0-1

If set to 1, `.ibd` file is converted (`space id`, `index id`, etc.) with index information in `.exp` file during the import process (`ALTER TABLE ... IMPORT TABLESPACE` command).

This variable was renamed to *innodb_import_table_from_xtrabackup*, beginning in release 5.5.10-20.1. It still exists as *innodb_expand_import* in versions prior to that.

variable innodb_import_table_from_xtrabackup

Version Info

- ***5.5.10-20.1*** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 0

Range 0-1

If set to 1, `.ibd` file is converted (`space id`, `index id`, etc.) with index information in .exp file during the import process (`ALTER TABLE ... IMPORT TABLESPACE` command).

This variable was added in release 5.5.10-20.1. Prior to that, it was named *innodb_expand_import*, which still exists in earlier versions.

43.4 Other reading

- Moving InnoDB tables between servers
- Copying InnoDB tables between servers

CHAPTER

FORTYFOUR

DUMP/RESTORE OF THE BUFFER POOL

Percona Server can speed up restarts by saving and restoring the contents of the buffer pool, the largest memory buffer the *MySQL* server typically uses. Servers with large amounts of memory typically need a long time to warm up the buffer pool after a restart, so a server cannot be placed under production load for hours or even days. This special feature of *Percona Server* enables the buffer pool to be restored to its pre-shutdown state in a matter of minutes.

The feature works as follows. The buffer pool is a list of pages, usually 16kb in size, which are identified by an 8-byte number. The list is kept in least-recently-used order, which is why the buffer pool is sometimes referred to as an LRU list. The mechanism is to save the list of 8-byte page numbers just before shutdown, and after restart, to read the pages from disk and insert them back into the LRU at the correct position. The pages are sorted by ID to avoid random I/O, which is slower than sequential I/O on most disks. The LRU list is saved to the file ib_lru_dump in the directory specified by the datadir configuration setting, so you can back it up and restore it with the rest of your data easily.

Note that this feature does not store the contents of the buffer pool (i.e. it does not write 1GB of data to disk if you have a 1GB buffer pool). It stores only the identifiers of the pages in the buffer pool, which is a very small amount of data even for large buffer pools.

This feature can be used both manually and automatically. It is safe to enable automatically, and we have not found any performance regressions in it.

44.1 Automatic Operation

To perform dump/restore of the buffer pool automatically, set the *innodb_auto_lru_dump* configuration variable. A non-zero value for this variable causes the server to create a new thread at startup. This thread's first task is to read and sort the saved file, and then restore the LRU accordingly.

After finishing the restore operation, the thread switches into dump mode, to periodically dump the LRU. The period is specified by the configuration variable's value in seconds. For example, if you set the variable to 60, then the thread saves the LRU list once per minute.

44.2 Manual Operation

Manual dump/restore is done through the `INFORMATION_SCHEMA` using the following two administrative commands:

- `XTRA_LRU_DUMP`: Dumps the contents of the buffer pool (a list of space_id and page_no) to the file ib_lru_dump in the directory specified by the datadir configuration setting.
- `XTRA_LRU_RESTORE`: Restores pages based on the file ib_lru_dump.

Here is an example of how to manually save and restore the buffer pool. On a running server, examine the number of pages in the buffer pool, as in the following example:

```
mysql> show status like ``innodb_buffer_pool_pages_data``;
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| innodb_buffer_pool_pages_data | 6231  |
+-------------------------------+-------+
```

Save the contents of the LRU list to a file:

```
mysql> select * from information_schema.XTRADB_ADMIN_COMMAND /*!XTRA_LRU_DUMP*/;
+-----------------------------+
| result_message              |
+-----------------------------+
| XTRA_LRU_DUMP was succeeded. |
+-----------------------------+
1 row in set (0.02 sec)
```

This is a fast operation, and the resulting file is very small compared to the buffer pool. The file is in binary format, not text format. Now restart *MySQL*, and examine the number of pages in the buffer pool, for example,

```
mysql> show status like ``innodb_buffer_pool_pages_data``;
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| innodb_buffer_pool_pages_data | 22    |
+-------------------------------+-------+
```

The following command instructs *XtraDB* to restore the LRU from the file:

```
mysql> select * from information_schema.XTRADB_ADMIN_COMMAND /*!XTRA_LRU_RESTORE*/;
+---------------------------------+
| result_message                  |
+---------------------------------+
| XTRA_LRU_RESTORE was succeeded. |
+---------------------------------+
1 row in set (0.62 sec)
```

This command executes quickly, because it doesn't `use direct_io`. Afterwards, inspect the status of the buffer pool again:

```
mysql> show status like ``innodb_buffer_pool_pages_data``;
+-------------------------------+-------+
| Variable_name                 | Value |
+-------------------------------+-------+
| innodb_buffer_pool_pages_data | 6231  |
+-------------------------------+-------+
```

44.3 Status Information

Status information about the dump and restore is written to the server's error file:

```
....
091217 11:49:16 InnoDB: administration command ``XTRA_LRU_DUMP`` was detected.
....
```

```
091217 11:51:44 InnoDB: administration command ``XTRA_LRU_RESTORE`` was detected.
091217 11:51:45 InnoDB: reading pages based on the dumped LRU list was done.␣
↪(requested: 6231, read: 6209)
```

The requested number of pages is the number of pages that were in the LRU dump file. A page might not be read if it is already in the buffer pool, or for some other miscellaneous reasons, so the number of pages read can be less than the number requested.

44.4 Implementation Details

The mechanism used to read pages into the LRU is the normal *InnoDB* calls for reading a page into the buffer pool. This means that it still performs all of the usual checks for data integrity. It also means that if you decrease the size of the buffer pool, *InnoDB* uses the usual page replacement and flushing algorithm to free pages when it becomes full.

The pages are sorted by tablespace, and then by ID within the tablespace.

The dump file is not deleted after loading, so you should delete it if you wish to disable the feature. For example, suppose you dump the LRU, and then some time later you decide to enable automatic dumping and reloading. You set the configuration variable and restart *MySQL*. Upon restart, the server will load the LRU to its state in the previously saved file, which might be very stale and not what you want to happen.

44.5 Block Startup until LRU dump is loaded

Percona Server provides a boolean option to block the start of XtraDB until LRU is preloaded from dump. When the variable *innodb_blocking_buffer_pool_restore* is set to ON, XtraDB waits until the restore of the dump is completed before reporting successful startup to the server. This variable is OFF by default.

44.6 Version Specific Information

- *5.5.8-20.0*: Automatic dump/restore implemented.
- *5.5.10-20.1*: Renamed variable *innodb_auto_lru_dump* to *innodb_buffer_pool_restore_at_startup*.

44.7 System Variables

variable innodb_auto_lru_dump

Version Info

- ***5.5.10-20.1*** – Renamed.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 0

Range 0-UINT_MAX32

Units Seconds

This variable specifies the time in seconds between automatic buffer pool dumps. When set to zero, automatic dumps are disabled and must be done manually. When set to a non-zero value, an automatic restore of the buffer pool is also performed at startup, as described above.

This variable was renamed to *innodb_buffer_pool_restore_at_startup*, beginning in release 5.5.10-20.1. It still exists as *innodb_auto_lru_dump* in versions prior to that.

variable innodb_blocking_buffer_pool_restore

Version Info

- ***5.5.16-22.0*** – Added

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Default Value OFF

Range ON/OFF

When this variable is set to ON *XtraDB* waits until the restore of the dump is completed before reporting successful startup to the server.

variable innodb_buffer_pool_restore_at_startup

Version Info

- ***5.5.10-20.1*** – Added.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 0

Range 0-UINT_MAX32

Units Seconds

This variable specifies the time in seconds between automatic buffer pool dumps. When set to zero, automatic dumps are disabled and must be done manually. The variable *innodb_buffer_pool_restore_at_startup* controls both automatic buffer pool dumps and automatic restore on startup. When set to a non-zero value, an automatic restore of the buffer pool is also performed at startup, as described above.

This variable was added in release 5.5.10-20.1. Prior to that, it was named *innodb_auto_lru_dump*, which still exists in earlier versions.

44.8 INFORMATION_SCHEMA Tables

This feature provides the following table:

table INFORMATION_SCHEMA.**XTRADB_ADMIN_COMMAND**

Columns

- **result_message** – result message of the XTRADB_ADMIN_COMMAND

44.9 Other reading

- Save / restore buffer pool

CHAPTER

FORTYFIVE

FAST INDEX CREATION

Percona has implemented several changes related to *MySQL*'s fast index creation feature. Extended features, besides disabling *fast_index_creation*, can be enabled with *expand_fast_index_creation*.

45.1 Disabling Fast Index Creation

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

This feature implements a session variable that disables fast index creation. This causes indexes to be created in the way they were created before fast index creation was implemented. While this is slower, it avoids the problem of data dictionary inconsistency between *MySQL* and *InnoDB*.

45.2 Tunable buffer size for fast index creation

Percona Server supports tunable buffer size for fast index creation in *InnoDB*. This value was calculated based on the merge block size (which was hardcoded to 1 MB) and the minimum index record size. By adding the session variable *innodb_merge_sort_block_size* block size that is used in the merge sort can now be adjusted for better performance.

45.3 Version Specific Information

- *5.5.8-20.0*: Variable *fast_index_creation* implemented.
- *5.5.11-20.2*: Expanded the applicability of fast index creation to **mysqldump**, `ALTER TABLE`, and `OPTIMIZE TABLE`.
- *5.5.27-28.0* Variable *innodb_merge_sort_block_size* implemented.

45.4 System Variables

variable fast_index_creation

Command Line Yes

Config File No

Scope Local

Dynamic Yes

Variable Type Boolean

Default Value ON

Range ON/OFF

variable `innodb_merge_sort_block_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 1048576 (1M)

Range 1048576 - 1073741824 (1G)

45.5 Other Reading

- Thinking about running OPTIMIZE on your InnoDB Table? Stop!
- Building Indexes by Sorting In Innodb

CHAPTER

FORTYSIX

EXPANDED FAST INDEX CREATION

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.

46.1 mysqldump

A new option, `--innodb-optimize-keys`, was implemented in **mysqldump**. It changes the way *InnoDB* tables are dumped, so that secondary 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.

46.2 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.
- This optimization won't work in case the index is dropped and added in the same `ALTER TABLE` statement because in that case *MySQL* copies the table.

46.3 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`.

46.4 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.

46.4.1 Version Specific Information

- *5.5.16-22.0* Variable *expand_fast_index_creation* implemented. This variable controls whether fast index creation optimizations made by Percona are used.

46.4.2 System Variables

variable expand_fast_index_creation

Command Line Yes

Config File No

Scope Local/Global

Dynamic Yes

Variable Type Boolean

Default Value OFF

Range ON/OFF

46.4.3 Other Reading

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

CHAPTER

FORTYSEVEN

PREVENT CACHING TO FLASHCACHE

FlashCache increases performance by caching data on SSDs. It works even better when only hot data is cached. This feature prevents the caching of the unwanted blocks of data.

Better utilization of *FlashCache* partitions is achieved when caching of rarely used data is avoided. Use of this feature prevents blocks of data from being cached to *FlashCache* during a query.

Usage of the feature is as follows:

```
SELECT /* sql_no_fcache */ ...
```

The **mysqldump** binary was changed to use this option.

47.1 Version-Specific Information

- *5.5.8-20.0*: Full functionality available.
- *5.5.27-29.0*: Variable *have_flashcache* introduced.

47.2 System Variables

variable have_flashcache

Version Info

- ***5.5.27-29.0*** – Variable introduced

Command Line No

Config File No

Scope Global

Dynamic No

Variable Type Boolean

Range Yes/No

This variable shows if the server was compiled with Flashcache support.

47.3 Status Variables

variable `Flashcache_enabled`

Scope Global

Variable Type Boolean

Range OFF/ON

This status variable shows if the Flashcache support has been enabled.

47.4 Other Information

The feature is a port of the original *Facebook* change.

47.5 Other reading

- Releasing Flashcache
- Level 2 Flash cache is there

CHAPTER

FORTYEIGHT

PERCONA TOOLKIT UDFS

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

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

48.1 Version Specific Information

- *5.5.8-20.0*: Began distributing `libfnv1a_udf`, `libfnv_udf`, and `libmurmur_udf`.

48.2 Other Information

- Author / Origin: Baron Schwartz

48.3 Installation

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

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

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

48.4 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`.

48.5 Other Reading

- *Percona Toolkit* documentation

CHAPTER

FORTYNINE

SUPPORT FOR FAKE CHANGES

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

49.1 Concept of Replication Prefetching

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

49.2 Prefetching with InnoDB Fake Changes

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

Keeping in view the need of prefetching and the current caveats the *innodb_fake_changes* variable was implemented. The *innodb_fake_changes* variable enables an option for the server-side which allows for prefetching to work in a more performant manner. What enabling this option really does is that *InnoDB* reads the data needed by the `DML` queries but does not actually update the records, and hence no undo log record is generated, as nothing has changed, which means that rollback is instantaneous, and *InnoDB* doesn't have to do any additional work on rollback. However, the problem of locking contention is not completely solved, when the records are read, SHARED locks are taken on the records, so this can still cause contention with data changes that SQL thread needs to make. *Percona Server* does have a variable *innodb_locking_fake_changes* to make fake changes implementation

completely lock-less. Because the fake changes implementation is not ready for lock-less operation for all workloads this variable is not safe to use and that is why it is disabled by default.

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

49.3 Caveats

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

49.3.1 DML operations are supported

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

49.3.2 DDL operations are not supported

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

49.3.3 Explicit COMMIT will lead to an error

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

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

49.4 How to use InnoDB Fake Changes

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

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

Both the tools rely on the *Percona Server* `innodb_fake_changes` option.

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

49.5 System Variables

variable `innodb_fake_changes`

Version Info

- ***5.5.16-22.0*** – Introduced

Scope Global, Session

Type Boolean

Dynamic Yes

Default Value OFF

This variable enables the *Support for Fake Changes* feature.

variable `innodb_locking_fake_changes`

Version Info

- ***5.5.28-29.2*** – Introduced

Scope Global, Session

Type Boolean

Dynamic Yes

Default Value ON

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

49.6 Implementation Details

- The fake session is used as a prefetch of the replication, it should not affect to later replication SQL execution.
- The effective unit is each transaction. The behavior is decided at the start of the each one and never changed during the transaction
- `INSERT` operations doesn't use the `INSERT BUFFER`, it always causes the reading of the page actually for the option. `DELETE` also doesn't use the `INSERT BUFFER`.
- It never acquires `X_LOCK` from tables or records, only `S_LOCK`.

- The auto increment values behaves as usual.
- It reserves free pages as usual.
- Existed only `root ~ leaf` pages, which are accessed in the `DML` operation.
- It will not prefetch allocate/free, split/merge, `INODE`, `XDES` or other management pages. The same is for extern pages, i.e. large `BLOB` s).
- Foreign key constraints are checked (for causing IO), but passed always.

49.7 Related Reading

- on MySQL replication prefetching
- replication prefetching revisited

CHAPTER

FIFTY

KILL IDLE TRANSACTIONS

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

50.1 System Variables

variable `innodb_kill_idle_transaction`

Version Info

- ***`5.5.16-22.0`*** – Introduced

Scope `GLOBAL`

Config `YES`

Dynamic `YES`

Variable Type `INTEGER`

Default Value 0 (disabled)

Units Seconds

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

CHAPTER

FIFTYONE

ENFORCING STORAGE ENGINE

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

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

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

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

51.1 Version Specific Information

- *5.5.24-26.0* Variable *enforce_storage_engine* implemented.

51.2 System Variables

variable `enforce_storage_engine`

Command Line No

Config File Yes

Scope Global

Dynamic No

Variable Type String

Default Value NULL

Note: This variable is not case sensitive.

51.3 Example

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

```
enforce_storage_engine=InnoDB
```

CHAPTER

FIFTYTWO

UTILITY USER

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

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

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

52.1 System Variables

variable `utility_user`

Version Info

- ***5.5.27-28.0*** – Implemented

Command Line Yes

Config File utility_user=<user@host>

Scope Global

Dynamic No

Variable Type String

Default Value NULL

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

variable `utility_user_password`

Version Info

- ***5.5.27-28.0*** – Implemented

Command Line Yes

Config File utility_user_password=<password>

Scope Global

Dynamic No

Variable Type String

Default Value NULL

Specifies the password required for the utility user.

variable `utility_user_schema_access`

Version Info

- ***5.5.27-28.0*** – Implemented

Command Line Yes

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

Scope Global

Dynamic No

Variable Type String

Default Value NULL

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

variable `utility_user_privileges`

Version Info

- ***5.5.34-32.0*** – Implemented

Command Line Yes

Config File utility_user_privileges=<privilege1>,<privilege2>,<privilege3>

Scope Global

Dynamic No

Variable Type String

Default Value NULL

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

CHAPTER

FIFTYTHREE

EXTENDING THE `secure-file-priv` SERVER OPTION

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

In *Percona Server* *5.5.53-38.4* empty `secure-file-priv` became an alias for `NULL` value: both disable `LOAD_FILE()`, `LOAD DATA INFILE`, and `SELECT INTO OUTFILE`. With this change it is no longer possible to disable security checks by omitting the option as that would take the default value (`/var/lib/mysql-files/` for `.deb` and `.rpm` and `NULL` for `.tar.gz` packages. Instead, `--secure-file-priv=''` (or `=/`) should be used.

53.1 Version Specific Information

- *5.5.25a-27.1* Variable *secure-file-priv* extended behavior implemented.
- *5.5.53-38.4* Default value for *secure-file-priv* has been changed from `NULL` to `/var/lib/mysql-files/` when installed from `.deb` and `.rpm` packages.

53.2 System Variables

variable `secure-file-priv`

Command Line No

Config File Yes

Scope Global

Dynamic No

Variable Type String

Default Value `/var/lib/mysql-files/` - for `.deb` and `.rpm` packages

Default Value `NULL` - for `.tar.gz` packages

CHAPTER

FIFTYFOUR

EXPANDED PROGRAM OPTION MODIFIERS

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

Currently MySQL supports five existing option modifiers:

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

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

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

54.1 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>
```

54.2 Version Specific Information

- *5.5.27-28.0* Expanded program option modifiers implemented

54.3 Examples

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

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

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

Initial `query_cache_size` size:

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

Setting up bigger value:

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

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

Setting up smaller value:

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

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

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

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

```
hidden-query_cache_size=1
```

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

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

```
readonly-query_cache_size=1
```

Trying to change the variable value will result in error:

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

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

CHAPTER

FIFTYFIVE

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*.

55.1 User statements for handling the XtraDB changed page bitmaps

In *Percona Server 5.5.29-30.0* 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.

55.2 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.

55.3 INFORMATION_SCHEMA Tables

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

table `INFORMATION_SCHEMA.`**`INNODB_CHANGED_PAGES`**

Columns

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

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

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

55.4 System Variables

variable `innodb_max_changed_pages`

Version Info

- ***`5.5.27-29.0`*** – Variable `innodb_changed_pages_limit` introduced
- ***`5.5.29-30.0`*** – Variable renamed to *`innodb_max_changed_pages`*

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 1000000

Range 1 - 0 (unlimited)

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

variable `innodb_track_changed_pages`

Version Info

- *5.5.27-29.0* – Variable introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Default Value 0 - False

Range 0-1

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

variable `innodb_max_bitmap_file_size`

Version Info

- *5.5.28-29.2* – Variable introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 104857600 (100 MB)

Range 4096 (4KB) - 18446744073709551615 (16EB)

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

CHAPTER

FIFTYSIX

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*.

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.

56.1 Installation

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

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

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

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

56.2 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
```

56.3 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.

56.4 Supplementary groups support

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

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

56.5 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.

56.6 Version Specific Information

- *5.5.24-26.0* PAM authentication plugin has been integrated with *Percona Server*.
- *5.5.32-31.0* Implemented PAM support for supplementary groups.

CHAPTER

FIFTYSEVEN

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, it's own unique `RECORD_ID` field and a `TIMESTAMP` field. This implementation is alternative to the MySQL Enterprise Audit Log Plugin

Audit Log plugin produces the log of following events:

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

Example of the Audit event:

```
<AUDIT_RECORD
 "NAME"="Audit"
 "RECORD"="1_2014-04-29T09:29:40"
 "TIMESTAMP"="2014-04-29T09:29:40 UTC"
 "MYSQL_VERSION"="5.5.37-35.0-655.trusty"
 "STARTUP_OPTIONS"="--basedir=/usr --datadir=/var/lib/mysql --plugin-dir=/usr/lib/
↪mysql/plugin --user=mysql --log-error=/var/log/mysql/error.log --pid-file=/var/run/
↪mysqld/mysqld.pid --socket=/var/run/mysqld/mysqld.sock --port=3306"
 "OS_VERSION"="x86_64-debian-linux-gnu",
 />
```

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

Example of the Disconnect event:

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

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

Example of the Query event:

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

57.1 Installation

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

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

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

```
SHOW PLUGINS;
```

Audit log should be listed in the output:

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

57.2 Log Format

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

Example of the `OLD` format:

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

Example of the `NEW` format:

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

Example of the `JSON` format:

```
{"audit_record":{"name":"Query","record":"4707_2014-08-27T10:43:52","timestamp":"2014-
↪08-27T10:44:19 UTC","command_class":"show_databases","connection_id":"37","status
↪":0,"sqltext":"show databases","user":"root[root] @ localhost []","host":"localhost
↪","os_user":"","ip":""}}
```

Example of the `CSV` format:

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

57.3 Streaming the audit log to syslog

Ability to stream the audit log to syslog was implemented in *Percona Server 5.5.39-36.0*. To stream the audit log to syslog you'll need to set *audit_log_handler* variable to `SYSLOG`. To control the syslog file handler, the following variables can be used: *audit_log_syslog_ident*, *audit_log_syslog_facility*, and *audit_log_syslog_priority* These variables have the same meaning as appropriate parameters described in the syslog(3) manual.

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

57.4 System Variables

variable `audit_log_strategy`

Version Info

- ***5.5.37-35.0*** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value ASYNCHRONOUS

Allowed values `ASYNCHRONOUS`, `PERFORMANCE`, `SEMISYNCHRONOUS`, `SYNCHRONOUS`

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

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

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

variable `audit_log_file`

Version Info

- ***5.5.37-35.0*** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value audit.log

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

variable `audit_log_flush`

Version Info

- ***5.5.37-35.0*** – Implemented

Command Line Yes

Scope Global

Dynamic Yes

Variable Type String

Default Value OFF

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

variable `audit_log_buffer_size`

Version Info

- ***5.5.37-35.0*** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 4096

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

variable audit_log_format

Version Info

- ***5.5.37-35.0*** – Original implementation
- ***5.5.39-36.0*** – Implemented support for CSV and JSON log formats

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value OLD

Allowed values OLD, NEW, CSV, JSON

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

variable audit_log_policy

Version Info

- ***5.5.37-35.0*** – Implemented

Command Line Yes

Scope Global

Dynamic Yes

Variable Type String

Default Value ALL

Allowed values ALL, LOGINS, QUERIES, NONE

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

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

variable `audit_log_rotate_on_size`

Version Info

- ***5.5.37-35.0*** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 0 (don't rotate the log file)

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

variable `audit_log_rotations`

Version Info

- ***5.5.37-35.0*** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type Numeric

Default Value 0

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

variable `audit_log_handler`

Version Info

- ***5.5.39-36.0*** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value FILE

Allowed values `FILE`, `SYSLOG`

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

variable `audit_log_syslog_ident`

Version Info

- ***5.5.39-36.0*** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value percona-audit

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

variable `audit_log_syslog_facility`

Version Info

- ***`5.5.39-36.0`*** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value LOG_USER

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

variable `audit_log_syslog_priority`

Version Info

- ***`5.5.39-36.0`*** – Implemented

Command Line Yes

Scope Global

Dynamic No

Variable Type String

Default Value LOG_INFO

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

57.5 Version Specific Information

- *`5.5.37-35.0`* Audit Log plugin has been implemented in *Percona Server*.
- *`5.5.39-36.0`* *Percona Server* *Audit Log Plugin* now supports `JSON` and `CSV` log formats.
- *`5.5.39-36.0`* *Percona Server* *Audit Log Plugin* now supports *streaming the audit log to syslog*.

CHAPTER

FIFTYEIGHT

EXTENDED SHOW GRANTS

Prior to *Percona Server 5.5.42-37.1*, `SHOW GRANTS` displays only the privileges granted explicitly to the named account. Other privileges might be available to the account, but they are not displayed. For example, if an anonymous account exists, the named account might be able to use its privileges, but `SHOW GRANTS` will not display them. In *5.5.42-37.1* `SHOW GRANTS` command was extended to display all the effectively available privileges to the account.

58.1 Example

If we create the following users:

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

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

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

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

- `SHOW GRANTS` output before the change:

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

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

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

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

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

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

58.1.1 Version-Specific Information

- *5.5.42-37.1*: Feature implemented

58.1.2 Other reading

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

Part VIII

Diagnostics Improvements

CHAPTER

FIFTYNINE

INNODB STATISTICS

This feature provides new startup options (control method and collection of index statistics estimation) and information schema views to confirm the statistics.

59.1 Version Specific Information

- *5.5.8-20.0*: Renamed three fields in *INNODB_INDEX_STATS* table.

59.2 System Variables

Four new system variables were introduced by this feature.

variable `innodb_stats_method`

Command Line YES

Config File YES

Scope GLOBAL

Dynamic YES

Type STRING

Default Value `nulls_equal`

Allowed Values `nulls_equal`, `nulls_unequal`, `nulls_ignored`

The values and meanings are almost same to `myisam_stats_method` option of native *MySQL* (`nulls_equal`, `nulls_unequal`, `nulls_ignored`). But *InnoDB* doesn't have several patterns of statistics currently. Even though this option can be changed dynamically, statistics needs to be re-calculated to change the method for the table.

(reference: MyISAM Index Statistics Collection)

variable `innodb_stats_auto_update`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Type BOOLEAN

Default Value 1

InnoDB updates the each index statistics automatically (many updates were done, some information_schema is accessed, table monitor, etc.). Setting this option 0 can stop these automatic recalculation of the statistics except for "first open" and "ANALYZE TABLE command".

variable `innodb_stats_update_need_lock`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Type BOOLEAN

Default Value 1

If you meet contention of `&dict_operation_lock`, setting 0 reduces the contention. But 0 disables to update `Data_free:` of `SHOW TABLE STATUS`.

variable `innodb_use_sys_stats_table`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Type BOOLEAN

Default Value 0

If this option is enabled, *XtraDB* uses the `SYS_STATS` system table to store statistics of table indexes. Also, when *InnoDB* opens a table for the first time, it loads the statistics from `SYS_STATS` instead of sampling index pages. If you use a high `stats_sample_pages` value, the first open of a table is expensive. In such a case, this option will help. Intended behavior is to never update statistics unless an explicit `ANALYZE TABLE` is issued.

59.3 INFORMATION_SCHEMA Tables

table `INFORMATION_SCHEMA.`**`INNODB_SYS_STATS`**
Shows statistics of table indexes.

Columns

- **`INDEX_ID`** – Index ID
- **`KEY_COLS`** – Number of key columns
- **`DIFF_VALS`** – Number of Different Values
- **`NON_NULL_VALS`** – Number of Non `NULL` Values

table `INFORMATION_SCHEMA.`**`INNODB_SYS_TABLES`**
Shows the information about *InnoDB* tables

Columns

- **`TABLE_ID`** – Table ID
- **`SCHEMA`** – Database (schema) name
- **`NAME`** – Table name

- **FLAG** – Contains *0* if it is a InnoDB system table or *1* it is a user table
- **N_COLS** – Number of columns in the table
- **SPACE** – Tablespace ID

table INFORMATION_SCHEMA.**INNODB_SYS_TABLESTATS**
Shows the information about the performance statistics of *InnoDB* tables.

Columns

- **TABLE_ID** – Table ID
- **SCHEMA** – Database (schema) Name
- **NAME** – Table Name
- **STATS_INITIALIZED** – Contains `Initialized` value if the statistics are collected or `Uninitialized` if they are not collected.
- **NUM_ROWS** – Estimated number of rows in the table.
- **CLUST_INDEX_SIZE** – Number of pages on disk that store the clustered index.
- **OTHER_INDEX_SIZE** – Number of pages on disk that store all secondary indexes.
- **MODIFIED_COUNTER** – Number of rows modified by DML operations.
- **AUTOINC** –
- **MYSQL_HANDLES_OPENED** –

table INFORMATION_SCHEMA.**INNODB_SYS_INDEXES**
Shows the information about *InnoDB* indexes

Columns

- **INDEX_ID** – Index ID
- **NAME** – Index Name
- **TABLE_ID** – Table ID
- **TYPE** – Numeric identifier signifying the index type
- **N_FIELDS** – Number of columns in the index
- **PAGE_NO** – Page offset within its tablespace
- **SPACE** – Tablespace ID

table INFORMATION_SCHEMA.**INNODB_SYS_COLUMNS**
Shows the information about the *InnoDB* table columns

Columns

- **TABLE_ID** – Table ID
- **NAME** – Column Name
- **POS** – Position of the column inside the table.
- **MTYPE** – Numeric identifier for the column type.
- **PRTYPE** – Binary value with bits representing data type, character set code and nullability.
- **LEN** – Column length.

table INFORMATION_SCHEMA.**INNODB_SYS_FIELDS**
Shows the information about the *InnoDB* index key fields.

Columns

- **INDEX_ID** – Index ID
- **NAME** – Index Name
- **POS** – Position of the field inside the index.

table INFORMATION_SCHEMA.**INNODB_SYS_FOREIGN**
Shows the information about the *InnoDB* foreign keys.

Columns

- **ID** – Foreign Key ID
- **FOR_NAME** – Database/Table which contains the Foreign Key
- **FOR_REF** – Database/Table being referenced by the Foreign Key
- **N_COLS** – Number of columns in the foreign key.
- **TYPE** – Type of foreign key, represented by the bit flags.

table INFORMATION_SCHEMA.**INNODB_SYS_FOREIGN_COLS**
Shows the information about the columns of the *InnoDB* foreign keys.

Columns

- **ID** – Foreign Key ID
- **FOR_COL_NAME** – Foreign Key Column Name
- **FOR_REF** – Referenced Column Name
- **POS** – Position of the field inside the index.

table INFORMATION_SCHEMA.**INNODB_TABLE_STATS**
Shows table statistics information of dictionary cached.

Columns

- **table_schema** – Database name of the table.
- **table_name** – Table name.
- **rows** – estimated number of all rows.
- **clust_size** – cluster index (table/primary key) size in number of pages.
- **other_size** – Other index (non primary key) size in number of pages.
- **modified** – Internal counter to judge whether statistics recalculation should be done.

If the value of modified column exceeds "rows / 16" or 2000000000, the statistics recalculation is done when `innodb_stats_auto_update == 1`. We can estimate the oldness of the statistics by this value.

table INFORMATION_SCHEMA.**INNODB_INDEX_STATS**
Shows index statistics information of dictionary cached.

Columns

- **table_schema** – Database name of the table.
- **table_name** – Table name.
- **index_name** – Index name.
- **fields** – How many fields the index key has. (it is internal structure of *InnoDB*, it may be larger than the `CREATE TABLE`).

- **rows_per_key** – Estimate rows per 1 key value. ([1 column value], [2 columns value], [3 columns value], ...).
- **index_total_pages** – Number of index pages.
- **index_leaf_pages** – Number of leaf pages.

In releases before 5.5.8-20.0, these fields had different names:

- `rows_per_key` was `row_per_keys`
- `index_total_pages` was `index_size`
- `index_leaf_pages` was `leaf_pages`

59.4 Example

`[innodb_stats_method = nulls_equal (default behavior of InnoDB)]`

```
mysql> explain SELECT COUNT(*), 0 FROM orgs2 orgs LEFT JOIN sa_opportunities2 sa_
↪opportunities ON orgs.org_id=sa_opportunities.org_id LEFT JOIN contacts2 contacts␣
↪ON orgs.org_id=contacts.org_id;
+----+-------------+------------------+-------+-----------------+-----------------+---
↪------+------------------+-------+-------------+
| id | select_type | table            | type  | possible_keys   | key             |␣
↪key_len | ref              | rows  | Extra       |
+----+-------------+------------------+-------+-----------------+-----------------+---
↪------+------------------+-------+-------------+
|  1 | SIMPLE      | orgs             | index | NULL            | orgs$org_id     | 4␣
↪       | NULL             |   128 | Using index |
|  1 | SIMPLE      | sa_opportunities | ref   | sa_opp$org_id   | sa_opp$org_id   | 5␣
↪       | test2.orgs.org_id |  5751 | Using index |
|  1 | SIMPLE      | contacts         | ref   | contacts$org_id | contacts$org_id | 5␣
↪       | test2.orgs.org_id | 23756 | Using index |
+----+-------------+------------------+-------+-----------------+-----------------+---
↪------+------------------+-------+-------------+
3 rows in set (0.00 sec)
```

`[innodb_stats_method = nulls_unequal or nulls_ignored]`

```
mysql> explain SELECT COUNT(*), 0 FROM orgs2 orgs LEFT JOIN sa_opportunities2 sa_
↪opportunities ON orgs.org_id=sa_opportunities.org_id LEFT JOIN contacts2 contacts␣
↪ON orgs.org_id=contacts.org_id;
+----+-------------+------------------+-------+-----------------+-----------------+---
↪------+------------------+------+-------------+
| id | select_type | table            | type  | possible_keys   | key             |␣
↪key_len | ref              | rows | Extra       |
+----+-------------+------------------+-------+-----------------+-----------------+---
↪------+------------------+------+-------------+
|  1 | SIMPLE      | orgs             | index | NULL            | orgs$org_id     | 4␣
↪       | NULL             |  128 | Using index |
|  1 | SIMPLE      | sa_opportunities | ref   | sa_opp$org_id   | sa_opp$org_id   | 5␣
↪       | test2.orgs.org_id |    1 | Using index |
|  1 | SIMPLE      | contacts         | ref   | contacts$org_id | contacts$org_id | 5␣
↪       | test2.orgs.org_id |    1 | Using index |
+----+-------------+------------------+-------+-----------------+-----------------+---
↪------+------------------+------+-------------+
3 rows in set (0.00 sec)
<example of information_schema>
```

```
mysql> select * from information_schema.innodb_table_stats;
+------------------------+-------+------------+------------+----------+
| table_name             | rows  | clust_size | other_size | modified |
+------------------------+-------+------------+------------+----------+
| test/sa_opportunities2 | 11175 |         21 |         11 |        0 |
| test/orgs2             |   128 |          1 |          0 |        0 |
| test/contacts2         | 47021 |         97 |         97 |        0 |
+------------------------+-------+------------+------------+----------+
3 rows in set (0.00 sec)

mysql> select * from information_schema.innodb_index_stats;
+------------------------+-----------------+--------+--------------+------------+-----
↪-------+
| table_name             | index_name      | fields | row_per_keys | index_size |␣
↪leaf_pages |
+------------------------+-----------------+--------+--------------+------------+-----
↪-------+
| test/sa_opportunities2 | GEN_CLUST_INDEX |      1 | 1            |         21 |  ␣
↪    20 |
| test/sa_opportunities2 | sa_opp$org_id   |      2 | 338, 1       |          11|  ␣
↪    10 |
| test/orgs2             | orgs$org_id     |      1 | 1            |          1 |  ␣
↪     1 |
| test/contacts2         | GEN_CLUST_INDEX |      1 | 1            |       97   |  ␣
↪    80 |
| test/contacts2         | contacts$org_id |      2 | 516, 0       |       97   |  ␣
↪    37 |
+------------------------+-----------------+--------+--------------+------------+-----
↪-------+
5 rows in set (0.00 sec)
```

59.5 Other reading

- InnoDB Table/Index stats

CHAPTER

SIXTY

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.

60.1 Version Specific Information

- ***5.5.10-20.1***: Renamed variable `userstat_running` to *userstat*.
- ***5.5.24-26.0***: TOTAL_SSL_CONNECTIONS column has been added to CLIENT_STATISTICS, THREAD_STATISTICS and USER_STATISTICS tables

60.2 Other Information

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

60.3 System Variables

variable `userstat`

Version Info

- ***5.5.10-20.1*** – Renamed from `userstat_running`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type BOOLEAN

Default Value OFF

Range ON/OFF

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

variable `thread_statistics`

Version Info

- ***`5.5.8-20.0`*** – Feature ported from *Percona Server* 5.1

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type BOOLEAN

Default Value OFF

Range ON/OFF

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

Note: In Percona Server 5.5 *`thread_statistics`* is a reserved word. Which means you have to quote it when using the system variable with the same name: `mysql> set global `thread_statistics`=1;`

60.4 INFORMATION_SCHEMA Tables

table `INFORMATION_SCHEMA.`**`CLIENT_STATISTICS`**

Columns

- **`CLIENT`** – The IP address or hostname from which the connection originated.
- **`TOTAL_CONNECTIONS`** – The number of connections created for this client.
- **`CONCURRENT_CONNECTIONS`** – The number of concurrent connections for this client.
- **`CONNECTED_TIME`** – The cumulative number of seconds elapsed while there were connections from this client.
- **`BUSY_TIME`** – The cumulative number of seconds there was activity on connections from this client.
- **`CPU_TIME`** – The cumulative CPU time elapsed, in seconds, while servicing this client''s connections.
- **`BYTES_RECEIVED`** – The number of bytes received from this client's connections.
- **`BYTES_SENT`** – The number of bytes sent to this client's connections.
- **`BINLOG_BYTES_WRITTEN`** – The number of bytes written to the binary log from this client's connections.
- **`ROWS_FETCHED`** – The number of rows fetched by this client's connections.
- **`ROWS_UPDATED`** – The number of rows updated by this client's connections.

- **TABLE_ROWS_READ** – The number of rows read from tables by this client's connections. (It may be different from ROWS_FETCHED.)
- **SELECT_COMMANDS** – The number of SELECT commands executed from this client's connections.
- **UPDATE_COMMANDS** – The number of UPDATE commands executed from this client's connections.
- **OTHER_COMMANDS** – The number of other commands executed from this client's connections.
- **COMMIT_TRANSACTIONS** – The number of COMMIT commands issued by this client's connections.
- **ROLLBACK_TRANSACTIONS** – The number of ROLLBACK commands issued by this client's connections.
- **DENIED_CONNECTIONS** – The number of connections denied to this client.
- **LOST_CONNECTIONS** – The number of this client's connections that were terminated uncleanly.
- **ACCESS_DENIED** – The number of times this client's connections issued commands that were denied.
- **EMPTY_QUERIES** – The number of times this client's connections sent empty queries to the server.
- **TOTAL_SSL_CONNECTIONS** – The number of times this client's connections connected using SSL to the server.

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

Example:

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

table INFORMATION_SCHEMA.**INDEX_STATISTICS**

Columns

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

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

This table makes it possible to do many things that were difficult or impossible previously. For example, you can use it to find unused indexes and generate DROP commands to remove them. If the index has not been used it won't be in this table.

Example:

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

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

table INFORMATION_SCHEMA.**TABLE_STATISTICS**

Columns

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

This table is similar in function to the INDEX_STATISTICS table.

Example:

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

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

table INFORMATION_SCHEMA.**THREAD_STATISTICS**

Columns

- **THREAD_ID** – 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.

table INFORMATION_SCHEMA.**USER_STATISTICS**

Columns

- **USER** – The username. The value #mysql_system_user# appears when there is no username (such as for the slave SQL thread).
- **TOTAL_CONNECTIONS** – The number of connections created for this user.

- **CONCURRENT_CONNECTIONS** – The number of concurrent connections for this user.
- **CONNECTED_TIME** – The cumulative number of seconds elapsed while there were connections from this user.
- **BUSY_TIME** – The cumulative number of seconds there was activity on connections from this user.
- **CPU_TIME** – The cumulative CPU time elapsed, in seconds, while servicing this user's connections.
- **BYTES_RECEIVED** – The number of bytes received from this user's connections.
- **BYTES_SENT** – The number of bytes sent to this user's connections.
- **BINLOG_BYTES_WRITTEN** – The number of bytes written to the binary log from this user's connections.
- **ROWS_FETCHED** – The number of rows fetched by this user's connections.
- **ROWS_UPDATED** – The number of rows updated by this user's connections.
- **TABLE_ROWS_READ** – The number of rows read from tables by this user's connections. (It may be different from `ROWS_FETCHED`.)
- **SELECT_COMMANDS** – The number of `SELECT` commands executed from this user's connections.
- **UPDATE_COMMANDS** – The number of `UPDATE` commands executed from this user's connections.
- **OTHER_COMMANDS** – The number of other commands executed from this user's connections.
- **COMMIT_TRANSACTIONS** – The number of `COMMIT` commands issued by this user's connections.
- **ROLLBACK_TRANSACTIONS** – The number of `ROLLBACK` commands issued by this user's connections.
- **DENIED_CONNECTIONS** – The number of connections denied to this user.
- **LOST_CONNECTIONS** – The number of this user's connections that were terminated uncleanly.
- **ACCESS_DENIED** – The number of times this user's connections issued commands that were denied.
- **EMPTY_QUERIES** – The number of times this user's connections sent empty queries to the server.
- **TOTAL_SSL_CONNECTIONS** – The number of times this user's connections connected using SSL to the server.

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

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

Example:

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

60.5 Commands Provided

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

These commands discard the specified type of stored statistical information.

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

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

60.6 Status Variables

variable Com_show_client_statistics

Variable Type numeric

Scope Global/Session

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

variable Com_show_index_statistics

Variable Type numeric

Scope Global/Session

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

variable Com_show_table_statistics

Variable Type numeric

Scope Global/Session

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

variable Com_show_thread_statistics

Variable Type numeric

Scope Global/Session

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

variable Com_show_user_statistics

Variable Type numeric

Scope Global/Session

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

CHAPTER

SIXTYONE

SLOW QUERY LOG

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

The ability to log queries with microsecond precision is essential for measuring the work the *MySQL* server performs. The standard slow query log in *MySQL* 5.0 has only 1-second granularity, which is too coarse for all but the slowest queries. *MySQL* 5.1 has microsecond resolution, but does not have the extra information about query execution that is included in the *Percona Server*.

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.

61.1 Version Specific Information

- *5.5.8-20.0*: Added values `profiling` and `profiling_use_getrusage` to variable log_slow_verbosity.
- ***5.5.10-20.1***:
  - Renamed variable *slow_query_log_timestamp_always* to *slow_query_log_timestamp_always*.
  - Renamed variable `slow_query_log_microseconds_timestamp` to *slow_query_log_timestamp_precision*.
  - Renamed variable `use_global_log_slow_control` to *slow_query_log_use_global_control*.
- ***5.5.34-32.0***:
  - New *slow_query_log_always_write_time* variable introduced.
- ***5.5.41-37.0***:
  - Implemented improved slow log reporting for queries in stored procedures.
- ***5.5.44-37.3***:
  - *log_slow_sp_statements* now supports skipping the logging of stored procedures into the slow log entirely with new `OFF_NO_CALLS` option.

61.2 Other Information

- Author / Origin: Maciej Dobrzanski, Percona

61.3 System Variables

variable `log_slow_admin_statements`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

When this variable is enabled, administrative statements will be logged to the slow query log. Upstream version of the *MySQL* server has implemented command line option with same name. Significant difference is that this feature is implemented as variable in *Percona Server*, that means it can be enabled/disabled dynamically without restarting the server.

variable `log_slow_filter`

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

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

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

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

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

variable `log_slow_rate_type`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Enumerated

Default Value `session`

Range `session`, `query`

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

variable `log_slow_rate_limit`

Command Line Yes

Config File Yes

Scope Global, session

Dynamic Yes

Default Value 1

Range 1-1000

Behavior of this variable depends from *log_slow_rate_type*.

Specifies that only a fraction of `session/query` should be logged. Logging is enabled for every 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 if you change *log_slow_rate_limit* to `session` mode, you should reconnect for the setting to take 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 5.5.34-32.0* information about the *log_slow_rate_limit* has been added to the slow query log. This means that if the *log_slow_rate_limit* is effective it will be reflected in the slow query log for each written query. Example of the output looks like this:

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

Prior to *5.5.37-35.0* implementation of the *log_slow_rate_type* set to `query` with *log_slow_rate_limit* feature would log every nth query deterministically. With the current implementation each query has a non-deterministic probability of 1/n to get logged."

variable `log_slow_slave_statements`

Command Line Yes

Config File Yes

Scope Global, session

Dynamic Yes

Specifies that slow queries replayed by the slave SQL thread on a *MySQL* slave will be logged. Upstream version of the *MySQL* server has implemented command line option with same name. Significant difference is that this feature is

implemented as variable in *Percona Server*, that means it can be enabled/disabled dynamically without restarting the server.

To start the logging from the slave thread, you should change the global value: set global *log_slow_slave_statements* =ON; and then execute: STOP SLAVE; START SLAVE;. This will destroy and recreate the slave SQL thread, so it will see the newly set global value.

To stop the logging from the slave thread, you should just change the global value: set global *log_slow_slave_statements* =OFF; the logging stops immediately.

variable log_slow_sp_statements

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value ON

Range ON/OFF/OFF_NO_CALLS

This variable controls the logging of stored procedures in the slow query log. The options are to log the CALL statements, to log the individual statements in the stored procedure, or to log neither. These options correspond to values OFF, ON, and OFF_NO_CALLS respectively.

Prior to *5.5.41-37.0* implementation of logging stored procedures was logging the stored procedure CALLs themselves along

- 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.

In *5.5.44-37.3* new OFF_NO_CALLS option has been implemented that can be used for skipping the logging of stored procedures into the slow log entirely.

Example of the improved stored procedure slow query log entry:

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

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

```
# Time: 150109 11:38:55
# User@Host: root[root] @ localhost []
# Thread_id: 40  Schema: world  Last_errno: 0  Killed: 0
# Query_time: 0.012989  Lock_time: 0.000033  Rows_sent: 4079  Rows_examined: 4079
↪Rows_affected: 0  Rows_read: 4079
# Bytes_sent: 161085
```

```
# Stored routine: world.improved_sp_log
SET timestamp=1420803535;
SELECT * FROM City;
# User@Host: root[root] @ localhost []
# Thread_id: 40  Schema: world  Last_errno: 0  Killed: 0
# Query_time: 0.001413  Lock_time: 0.000017  Rows_sent: 4318  Rows_examined: 4318
↪Rows_affected: 0  Rows_read: 4318
# Bytes_sent: 194601
# Stored routine: world.improved_sp_log
SET timestamp=1420803535;
SELECT * FROM Country;
```

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.

---

variable `log_slow_verbosity`

Version Info

- ***5.5.8-20.0*** – Added `profiling` and `profiling_use_getrusage`

Command Line Yes

Config File Yes

Scope Global, session

Dynamic Yes

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

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

- `full`: Equivalent to all other values OR'ed together without the `profiling` and `profiling_use_getrusage` options.
- `profiling`: Enables profiling of all queries in all connections.
- `profiling_use_getrusage`: Enables usage of the getrusage function.

Values are OR'ed together.

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

variable `slow_query_log_timestamp_always`

Version Info

- ***`5.5.10-20.1`*** – Introduced (renamed from `use_global_log_slow_control`)

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value FALSE

Range TRUE/FALSE

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

variable `slow_query_log_timestamp_precision`

Version Info

- ***`5.5.10-20.1`*** – Introduced (renamed from `slow_query_log_microseconds_timestamp`)

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Enumerated

Default Value `second`

Range `second`, `microsecond`

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

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

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

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

variable `slow_query_log_use_global_control`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value None

Version Info

- ***5.5.10-20.1*** – Introduced (renamed from `log_slow_timestamp_every`)

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)

NOTE: This variable has been renamed from `log_slow_timestamp_every` since 5.5.10-20.1.

variable `slow_query_log_always_write_time`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value 10 (seconds)

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.

61.4 Other Information

61.4.1 Changes to the Log Format

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

```
# User@Host: mailboxer[mailboxer] @  [192.168.10.165]
# Thread_id: 11167745  Schema: board
# Query_time: 1.009400  Lock_time: 0.000190  Rows_sent: 4  Rows_examined: 1543719
↪Rows_affected: 0  Rows_read: 4
# Bytes_sent: 278  Tmp_tables: 0  Tmp_disk_tables: 0  Tmp_table_sizes: 0
# InnoDB_trx_id: 1500
# 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
#   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
```

```
SET timestamp=1346844943;
SELECT id,title,production_year FROM title WHERE title = 'Bambi';
```

Another example (*log_slow_verbosity* =profiling):

```
# Query_time: 0.962742  Lock_time: 0.000202  Rows_sent: 4  Rows_examined: 1543719
↪Rows_affected: 0  Rows_read: 4
# Bytes_sent: 278  Tmp_tables: 0  Tmp_disk_tables: 0  Tmp_table_sizes: 0
# Profile_starting: 0.000030 Profile_starting_cpu: 0.000028 Profile_Waiting_for_query_
↪cache_lock: 0.000003
  Profile_Waiting_for_query_cache_lock_cpu: 0.000003 Profile_Waiting_on_query_cache_
↪mutex: 0.000003
  Profile_Waiting_on_query_cache_mutex_cpu: 0.000003 Profile_checking_query_cache_for_
↪query: 0.000076
  Profile_checking_query_cache_for_query_cpu: 0.000076 Profile_checking_permissions:
↪0.000011
  Profile_checking_permissions_cpu: 0.000011 Profile_Opening_tables: 0.000078 Profile_
↪Opening_tables_cpu: 0.000078
  Profile_System_lock: 0.000022 Profile_System_lock_cpu: 0.000022 Profile_Waiting_for_
↪query_cache_lock: 0.000003
  Profile_Waiting_for_query_cache_lock_cpu: 0.000002 Profile_Waiting_on_query_cache_
↪mutex: 0.000054
  Profile_Waiting_on_query_cache_mutex_cpu: 0.000054 Profile_init: 0.000039 Profile_
↪init_cpu: 0.000040
  Profile_optimizing: 0.000015 Profile_optimizing_cpu: 0.000014 Profile_statistics: 0.
↪000021 Profile_statistics_cpu: 0.000021
  Profile_preparing: 0.000020 Profile_preparing_cpu: 0.000020 Profile_executing: 0.
↪000003 Profile_executing_cpu: 0.000003
  Profile_Sending_data: 0.962324 Profile_Sending_data_cpu: 0.961526 Profile_end: 0.
↪000006 Profile_end_cpu: 0.000005
  Profile_query_end: 0.000004 Profile_query_end_cpu: 0.000004 Profile_closing_tables:
↪0.000008 Profile_closing_tables_cpu: 0.000008
  Profile_freeing_items: 0.000007 Profile_freeing_items_cpu: 0.000007 Profile_Waiting_
↪for_query_cache_lock: 0.000000
  Profile_Waiting_for_query_cache_lock_cpu: 0.000001 Profile_Waiting_on_query_cache_
↪mutex: 0.000001
  Profile_Waiting_on_query_cache_mutex_cpu: 0.000001 Profile_freeing_items: 0.000017
↪Profile_freeing_items_cpu: 0.000016
  Profile_Waiting_for_query_cache_lock: 0.000001 Profile_Waiting_for_query_cache_lock_
↪cpu: 0.000001
  Profile_Waiting_on_query_cache_mutex: 0.000000 Profile_Waiting_on_query_cache_mutex_
↪cpu: 0.000001
  Profile_freeing_items: 0.000001 Profile_freeing_items_cpu: 0.000001 Profile_storing_
↪result_in_query_cache: 0.000002
  Profile_storing_result_in_query_cache_cpu: 0.000002 Profile_logging_slow_query: 0.
↪000001 Profile_logging_slow_query_cpu: 0.000001
# Profile_total: 0.962751 Profile_total_cpu: 0.961950
# InnoDB_trx_id: 1700
```

61.4.2 Connection and Schema Identifier

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

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

```
# Thread_id: 11167745  Schema: board
```

61.4.3 Microsecond Time Resolution and Extra Row Information

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

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

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

Values and context:

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

61.4.4 Memory Footprint

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

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

Values and context:

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

61.4.5 Query Plan Information

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

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

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

61.4.6 *InnoDB* Usage Information

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

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

Values:

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

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

61.5 Related Reading

- Impact of logging on MySQL's performance
- log_slow_filter Usage
- Blueprint in Launchpad

CHAPTER

SIXTYTWO

EXTENDED SHOW ENGINE *INNODB* STATUS

This feature reorganizes the output of `SHOW ENGINE INNODB STATUS` for a better readability and prints the amount of memory used by the internal hash tables. In addition, new variables are available to control the output.

This feature modified the `SHOW ENGINE INNODB STATUS` command as follows:

- `TRANSACTION` section was moved to the end of the output, so that important information is not overlooked when the there is a large amount of it.
- Added two variables to control `SHOW ENGINE INNODB STATUS` information presented (bugfix for upstream bug #29126):
  - *`innodb_show_verbose_locks`* - Whether to show records locked
  - *`innodb_show_locks_held`* - Number of locks held to print for each *InnoDB* transaction
- Added extended information about *InnoDB* internal hash table sizes (in bytes) in the `BUFFER POOL AND MEMORY` section; also added buffer pool size in bytes.
- Added additional LOG section information (beginning in release 5.5.8-20.0).

62.1 Version Specific Information

- *`5.5.8-20.0`* Added status variables showing information from `SHOW ENGINE INNODB STATUS`.
- *`5.5.8-20.0`* Added additional information in the LOG section.
- *`5.5.10-20.1`*: Renamed status variable `innodb_row_lock_numbers` to `innodb_current_row_locks`.
- *`5.5.31-30.3`*: Added `innodb_read_views_memory` and `innodb_descriptors_memory` to improve *InnoDB* memory diagnostics.

62.2 Other Information

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

62.3 System Variables

variable `innodb_show_verbose_locks`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 0

Range 0 - 1

Specifies to show records locked in `SHOW ENGINE INNODB STATUS`. The default is 0, which means only the higher-level information about the lock (which table and index is locked, etc.) is printed. If set to 1, then traditional *InnoDB* behavior is enabled: the records that are locked are dumped to the output.

variable `innodb_show_locks_held`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type ULONG

Default Value 10

Range 0 - 1000

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

62.4 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.

62.4.1 BACKGROUND THREAD

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

Insert an example of BACKGROUND THREAD section output here.

variable `Innodb_master_thread_1_second_loops`

Version Info

- ***`5.5.8-20.0`*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_master_thread_10_second_loops`

Version Info

- ***`5.5.8-20.0`*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_master_thread_background_loops`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_master_thread_main_flush_loops`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_master_thread_sleeps`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_background_log_sync`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

62.4.2 SEMAPHORES

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

```
----------
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 9664, signal count 11182
Mutex spin waits 20599, rounds 223821, OS waits 4479
RW-shared spins 5155, OS waits 1678; RW-excl spins 5632, OS waits 2592
Spin rounds per wait: 10.87 mutex, 15.01 RW-shared, 27.19 RW-excl
```

variable `Innodb_mutex_os_waits`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_mutex_spin_rounds`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_mutex_spin_waits`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_s_lock_os_waits`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_s_lock_spin_rounds`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_s_lock_spin_waits`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_x_lock_os_waits`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_x_lock_spin_rounds`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_x_lock_spin_waits`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

62.4.3 INSERT BUFFER AND ADAPTIVE HASH INDEX

The following variables contain information in the INSERT BUFFER AND ADAPTIVE HASH INDEX section of the output from `SHOW ENGINE INNODB STATUS`. An example of that output is:

```
-------------------------------------
INSERT BUFFER AND ADAPTIVE HASH INDEX
-------------------------------------
Ibuf: size 1, free list len 6089, seg size 6091,
44497 inserts, 44497 merged recs, 8734 merges
Hash table size 276707, node heap has 1 buffer(s)
0.00 hash searches/s, 0.00 non-hash searches/s
```

variable `Innodb_ibuf_discarded_delete_marks`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_discarded_deletes`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_discarded_inserts`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_free_list`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_merged_delete_marks`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_merged_deletes`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_merged_inserts`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_merges`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_segment_size`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_ibuf_size`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_adaptive_hash_cells`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_adaptive_hash_heap_buffers`

Version Info

- *5.5.8-20.0* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_adaptive_hash_hash_searches`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_adaptive_hash_non_hash_searches`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

62.4.4 LOG

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

```
---
LOG
---
Log sequence number 28219393219
Log flushed up to 28219393219
Last checkpoint at 28212583337
Max checkpoint age 7782360
Checkpoint age target 7539162
Modified age 6809882
Checkpoint age 6809882
0 pending log writes, 0 pending chkp writes
8570 log i/o's done, 2000.00 log i/o's/second
```

variable `Innodb_lsn_current`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_lsn_flushed`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_lsn_last_checkpoint`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_checkpoint_age`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_checkpoint_max_age`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_checkpoint_target_age`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

62.4.5 BUFFER POOL AND MEMORY

The following variables contain information in the BUFFER POOL AND MEMORY section of the output from `SHOW ENGINE INNODB STATUS`. An example of that output is:

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total memory allocated 137625600; in additional pool allocated 0
Total memory allocated by read views 88
Internal hash tables (constant factor + variable factor)
    Adaptive hash index 3774352 (2213656 + 1560696)
    Page hash 139144
    Dictionary cache 629811 (554864 + 74947)
    File system 83536 (82672 + 864)
    Lock system 380792 (332872 + 47920)
    Recovery system 0 (0 + 0)
    Threads 84040 (82696 + 1344)
Dictionary memory allocated 74947
Buffer pool size 8192
Buffer pool size, bytes 134217728
Free buffers 0
Database pages 8095
Old database pages 2968
Modified db pages 5914
Pending reads 0
Pending writes: LRU 0, flush list 129, single page 0
Pages made young 372084, not young 0
2546000.00 youngs/s, 0.00 non-youngs/s
Pages read 103356, created 154787, written 979572
469000.00 reads/s, 78000.00 creates/s, 138000.00 writes/s
```

```
Buffer pool hit rate 994 / 1000, young-making rate 34 / 1000 not 0 / 1000
Pages read ahead 0.00/s, evicted without access 15000.00/s
```

variable `Innodb_mem_adaptive_hash`

Version Info

- *`5.5.8-20.0`* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_mem_dictionary`

Version Info

- *`5.5.8-20.0`* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_mem_total`

Version Info

- *`5.5.8-20.0`* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_buffer_pool_pages_LRU_flushed`

Version Info

- *`5.5.8-20.0`* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_buffer_pool_pages_made_not_young`

Version Info

- *`5.5.8-20.0`* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_buffer_pool_pages_made_young`

Version Info

- *`5.5.8-20.0`* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_buffer_pool_pages_old`

Version Info

- *`5.5.8-20.0`* – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_descriptors_memory`

Version Info

- ***5.5.31-30.3*** – Introduced.

Variable Type Numeric

Scope Global

This status variable shows the current size of the descriptors array (in bytes). The descriptor array is an *XtraDB* data structure that contains the information on currently running transactions.

variable `Innodb_read_views_memory`

Version Info

- ***5.5.31-30.3*** – Introduced.

Variable Type Numeric

Scope Global

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

62.4.6 OLDEST VIEW

This part contains the information about the oldest active transaction in the system. An example of that output is:

```
---OLDEST VIEW---
Normal read view
Read view low limit trx n:o 3300
Read view up limit trx id 3300
Read view low limit trx id 3300
Read view individually stored trx ids:
```

- `Read view low limit trx n:o` and `Read view up limit trx id` are the highest transactions IDs 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` 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.
- `Read view individually stored trx ids` contains the list of active transactions at the time the view was created.

62.4.7 TRANSACTIONS

The following variables contain information in the TRANSACTIONS section of the output from `SHOW ENGINE 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 engine innodb status
---TRANSACTION F561FC, ACTIVE 29 sec, process no 993, OS thread id 140213152769808␣
↪updating or deleting
mysql tables in use 1, locked 1
```

variable `Innodb_history_list_length`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_max_trx_id`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_oldest_view_low_limit_trx_id`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_purge_trx_id`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_purge_undo_no`

Version Info

- ***5.5.8-20.0*** – Introduced.

Variable Type Numeric

Scope Global

variable `Innodb_current_row_locks`

version 5.5.8-20.0 Introduced.

version 5.5.10-20.1 Renamed.

vartype Numeric

scope Global

This variable was named `innodb_row_lock_numbers` in release 5.5.8-20.0.

62.5 Other reading

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

CHAPTER

SIXTYTHREE

COUNT *INNODB* DEADLOCKS

When running a transactional application you have to live with deadlocks. They are not problematic as long as they do not occur too frequently. The standard `SHOW ENGINE INNODB STATUS` gives information on the latest deadlocks but it is not very useful when you want to know the total number of deadlocks or the number of deadlocks per unit of time.

This change adds a status variable that keeps track of the number of deadlocks since the server startup, opening the way to a better knowledge of your deadlocks.

This feature was provided by Eric Bergen under BSD license (see InnoDB Deadlock Count Patch).

It adds a new global status variable (`innodb_deadlocks`) showing the number of deadlocks.*

You can use it with `SHOW GLOBAL STATUS`, e.g.:

```
mysql> SHOW GLOBAL STATUS LIKE 'innodb_deadlocks';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| innodb_deadlocks | 323   |
+------------------+-------+
```

or with `INFORMATION_SCHEMA`, e.g.:

```
mysql> SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS WHERE VARIABLE_
↪NAME = 'innodb_deadlocks';
+----------------+
| VARIABLE_VALUE |
+----------------+
| 323            |
+----------------+
```

A deadlock will occur when at least two transactions are mutually waiting for the other to finish, thus creating a circular dependency that lasts until something breaks it. *InnoDB* is quite good at detecting deadlocks and generally returns an error instantly. Most transactional systems have no way to prevent deadlocks from occurring and must be designed to handle them, for instance by retrying the transaction that failed.

63.1 Version Specific Information

- *5.5.8-20.0*: Full functionality available.

63.2 Status Variables

One new status variable was introduced by this feature.

variable `Innodb_deadlocks`

Variable Type LONG

Scope Global

63.3 Related Reading

- Original post by Eric Bergen

CHAPTER

SIXTYFOUR

LOG ALL CLIENT COMMANDS (SYSLOG)

When enabled, this feature causes all commands run by the command line client to be logged to syslog. If you want to enable this option permanently, add it to the [mysql] group in my.cnf.

64.1 Version Specific Information

- *5.5.8-20.0*: Full functionality available.

64.2 Other Information

- Author / Origin: Percona

64.3 Client Variables

variable syslog

Command Line Yes

Config File Yes

Server No

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value OFF

Range ON/OFF

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

64.4 Other Reading

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

CHAPTER

SIXTYFIVE

RESPONSE TIME DISTRIBUTION

The slow query log provides exact information about queries that take a long time to execute. However, sometimes there are a large number of queries that each take a very short amount of time to execute. This feature provides a tool for analyzing that information by counting and displaying the number of queries according to the the length of time they took to execute. The user can define time intervals that divide the range 0 to positive infinity into smaller intervals and then collect the number of commands whose execution times fall into each of those intervals.

Note that in a replication environment, the server will not take into account *any* queries executed by the slave's SQL thread (whether they are slow or not) for the time distribution unless the *log_slow_slave_statements* variable is set.

The feature isn't implemented in all versions of the server. The variable *have_response_time_distribution* indicates whether or not it is implemented in the server you are running.

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
```

65.1 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
```

65.1.1 SELECT

You can get the distribution using the query:

```
> SELECT * from INFORMATION_SCHEMA.QUERY_RESPONSE_TIME
time                  count   total
0.000001              0       0.000000
0.000010              0       0.000000
0.000100              1       0.000072
0.001000              0       0.000000
0.010000              0       0.000000
0.100000              0       0.000000
1.000000              0       0.000000
10.000000             8       47.268416
100.000000            0       0.000000
1000.000000           0       0.000000
10000.000000          0       0.000000
100000.000000         0       0.000000
1000000.000000        0       0.000000
TOO LONG QUERY        0       0.000000
```

You can write a complex query like:

```
SELECT c.count, c.time,
(SELECT SUM(a.count) FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME as a WHERE a.count !
↪= 0) as query_count,
(SELECT COUNT(*)     FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME as b WHERE b.count !
↪= 0) as not_zero_region_count,
(SELECT COUNT(*)     FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME) as region_count
FROM INFORMATION_SCHEMA.QUERY_RESPONSE_TIME as c WHERE c.count > 0;
```

Note: If *query_response_time_stats* is ON, the execution times for these two SELECT queries will also be collected.

65.1.2 SHOW

Also, you can use this syntax:

```
> SHOW QUERY_RESPONSE_TIME;
time                  count   total
0.000001              0       0.000000
0.000010              0       0.000000
0.000100              1       0.000072
0.001000              0       0.000000
0.010000              0       0.000000
0.100000              0       0.000000
1.000000              0       0.000000
10.000000             8       47.268416
100.000000            0       0.000000
1000.000000           0       0.000000
10000.000000          0       0.000000
100000.000000         0       0.000000
1000000.000000        0       0.000000
TOO LONG QUERY        0       0.000000
```

Note: The execution time for the SHOW query will also be collected.

65.1.3 FLUSH

Flushing can be done with:

```
FLUSH QUERY_RESPONSE_TIME;
```

`FLUSH` does two things:

- Clears the collected times from the *`QUERY_RESPONSE_TIME`* table
- Reads the value of *`query_response_time_range_base`* and uses it to set the range base for the table

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

65.1.4 Stored procedures

Stored procedure calls count as single query.

65.1.5 Collect time point

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

65.2 Limitations

- `String width for seconds`
  - Value: 7
  - Compile-time variable: `QUERY_RESPONSE_TIME_STRING_POSITIVE_POWER_LENGTH`
- `String width for microseconds`
  - Value: 6
  - Compile-time variable: `QUERY_RESPONSE_TIME_STRING_NEGATIVE_POWER_LENGTH`
- Minimum range base
  - Value: 2
  - Compile-time variable: `QUERY_RESPONSE_TIME_MINIMUM_BASE`
- Minimum range base
  - Value: 1000
  - Compile-time variable: `QUERY_RESPONSE_TIME_MAXIMUM_BASE`
- Minimum time interval
  - Value: 1 microsecond
- Maximum time interval
  - Value: 9999999 seconds

65.3 Version Specific Information

- *5.5.8-20.0*: Introduced variable *have_response_time_distribution*.
- *5.5.8-20.0*: Introduced variable *query_response_time_stats*.

65.4 System Variables

variable `have_response_time_distribution`

Command Line No

Config File No

Version Info

- ***5.5.8-20.0*** – Introduced.

Scope Global

Dynamic No

Variable Type Boolean

Default Value YES

Range YES/NO

Contains the value YES if the server you're running supports this feature; contains NO if the feature is not supported. It is enabled by default.

variable `query_response_time_range_base`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 10

Range 2-1000

Sets up the logarithm base for the scale.

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

```
FLUSH QUERY_RESPONSE_TIME;
```

variable `query_response_time_stats`

Version Info

- ***5.5.8-20.0*** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value OFF

Range ON/OFF

This variable enables and disables collection of query times if the feature is available in the server that's running. If the value of variable *have_response_time_distribution* is YES, then you can enable collection of query times by setting this variable to ON using `SET GLOBAL`.

Prior to release 5.5.8-20.0, this variable was named `enable_query_response_time_stats`.

65.5 INFORMATION_SCHEMA Tables

table `INFORMATION_SCHEMA.`**`QUERY_RESPONSE_TIME`**

Columns

- **`TIME`** (*VARCHAR*) – Interval range in which the query occurred
- **`COUNT`** (*INT(11)*) – Number of queries with execution times that fell into that interval
- **`TOTAL`** (*VARCHAR*) – Total execution time of the queries

CHAPTER

SIXTYSIX

SHOW STORAGE ENGINES

This feature changes the comment field displayed when the `SHOW STORAGE ENGINES` command is executed and *XtraDB* is the storage engine.

Before the Change:

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

After the Change:

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

66.1 Version-Specific Information

- *5.5.8-20.0*: Full functionality available.

66.2 Other Information

- Author / Origin: Percona

CHAPTER

SIXTYSEVEN

SHOW LOCK NAMES

This feature is currently undocumented except for the following example.

Example:

```
mysql> SHOW ENGINE INNODB MUTEX;
+--------+----------------------------+---------------+
| Type   | Name                       | Status        |
+--------+----------------------------+---------------+
| InnoDB | &rseg->mutex               | os_waits=210  |
| InnoDB | &dict_sys->mutex           | os_waits=3    |
| InnoDB | &trx_doublewrite->mutex    | os_waits=1    |
| InnoDB | &log_sys->mutex            | os_waits=1197 |
| InnoDB | &LRU_list_mutex            | os_waits=2    |
| InnoDB | &fil_system->mutex         | os_waits=5    |
| InnoDB | &kernel_mutex              | os_waits=242  |
| InnoDB | &new_index->lock           | os_waits=2    |
| InnoDB | &new_index->lock           | os_waits=415  |
.....
```

CHAPTER

SIXTYEIGHT

PROCESS LIST

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

The changes that have been made as of version 5.5 of the server are:

- `SHOW PROCESSLIST` command:
  - added columns `ROWS_EXAMINED`, `ROWS_SENT`, and `ROWS_READ`
- `PROCESSLIST` table:
  - added columns `TIME_MS`, `ROWS_EXAMINED`, `ROWS_SENT`, and `ROWS_READ`

68.1 Version Specific Information

- *5.5.10-20.1*:
  - Added columns `ROWS_EXAMINED`, `ROWS_SENT`, and `ROWS_READ` to `SHOW PROCESSLIST` command.
  - Added columns `ROWS_EXAMINED`, `ROWS_SENT`, and `ROWS_READ` to *PROCESSLIST* table.

68.2 INFORMATION_SCHEMA Tables

table `INFORMATION_SCHEMA.`**`PROCESSLIST`**

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

Columns

- **`ID`** – The connection identifier.
- **`USER`** – The *MySQL* user who issued the statement.
- **`HOST`** – The host name of the client issuing the statement.
- **`DB`** – The default database, if one is selected, otherwise NULL.
- **`COMMAND`** – The type of command the thread is executing.
- **`TIME`** – The time in seconds that the thread has been in its current state.
- **`STATE`** – An action, event, or state that indicates what the thread is doing.
- **`INFO`** – The statement that the thread is executing, or NULL if it is not executing any statement.

- **TIME_MS** – The time in milliseconds that the thread has been in its current state.
- **ROWS_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.
- **ROWS_READ** – The number of rows read by the statement being executed.

68.3 Example Output

`SHOW PROCESSLIST` Command:

```
mysql> show processlist;
+------+----------+-----------+--------+---------+------+-----------+---------------
↪-------------------------------+-----------+--------------+-----------+
| Id   | User     | Host      | db     | Command | Time | State     | Info          ␣
↪                                | ROWS_SENT | ROWS_EXAMINED | ROWS_READ |
+------+----------+-----------+--------+---------+------+-----------+---------------
↪-------------------------------+-----------+--------------+-----------+
|    2 | root     | localhost | test   | Query   |    0 | NULL      | SHOW␣
↪PROCESSLIST                     |         0 |            0 |         1 |
|   14 | root     | localhost | test   | Query   |    0 | User lock | SELECT GET_
↪LOCK(``t``,1000)                |         0 |            0 |         1 |
+------+----------+-----------+--------+---------+------+-----------+---------------
↪-------------------------------+-----------+--------------+-----------+
```

Table *PROCESSLIST*:

```
mysql> select * from information_schema.PROCESSLIST;
+------+----------+-----------+--------+---------+------+-----------+---------------
↪-------------------------------+----------+--------------+-----------+-----------+
| ID   | USER     | HOST      | DB     | COMMAND | TIME | STATE     | INFO          ␣
↪                                | TIME_MS  | ROWS_EXAMINED | ROWS_SENT | ROWS_READ |
+------+----------+-----------+--------+---------+------+-----------+---------------
↪-------------------------------+----------+--------------+-----------+-----------+
|   14 | root     | localhost | test   | Query   |    0 | User lock | SELECT GET_
↪LOCK(``t``,1000)                |        1 |            0 |         0 |         ␣
↪   1 |
|    2 | root     | localhost | test   | Query   |    0 | executing | SELECT * from␣
↪INFORMATION_SCHEMA.PROCESSLIST |        0 |            0 |         0 |         1 |
+------+----------+-----------+--------+---------+------+-----------+---------------
↪-------------------------------+----------+--------------+-----------+-----------+
```

CHAPTER

SIXTYNINE

MISC. INFORMATION_SCHEMA TABLES

69.1 Temporary tables

Only the temporary tables that were explicitly created with *CREATE TEMPORARY TABLE* or *ALTER TABLE* are shown, and not the ones created to process complex queries.

table INFORMATION_SCHEMA.**GLOBAL_TEMPORARY_TABLES**

Columns

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

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

This information is also available by running the following command: .. code-block:: mysql

SHOW GLOBAL TEMPORARY TABLES

table INFORMATION_SCHEMA.**TEMPORARY_TABLES**

Columns

- **SESSION_ID** – *MySQL* connection id
- **TABLE_SCHEMA** – Schema in which the temporary table is created
- **TABLE_NAME** – Name of the temporary table
- **ENGINE** – Engine of the temporary table
- **NAME** – Internal name of the temporary table

- **TABLE_ROWS** – Number of rows of the temporary table
- **AVG_ROW_LENGTH** – Average row length of the temporary table
- **DATA_LENGTH** – Size of the data (Bytes)
- **INDEX_LENGTH** – Size of the indexes (Bytes)
- **CREATE_TIME** – Date and time of creation of the temporary table
- **UPDATE_TIME** – Date and time of the latest update of the temporary table

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

This information is also available by running the following command: .. code-block:: mysql

SHOW TEMPORARY TABLES

69.1.1 Status Variables

variable `Com_show_temporary_tables`

Variable Type Numeric

Scope Global/Session

The *Com_show_temporary_tables* statement counter variable indicates the number of times the statements `SHOW GLOBAL TEMPORARY TABLES` and `SHOW TEMPORARY TABLES` have been executed.

69.2 Buffer Pool Data Structure Tables

The following tables provide various information about the contents of the *InnoDB* buffer pool.

table `INFORMATION_SCHEMA.INNODB_BUFFER_POOL_PAGES`

Columns

- **PAGE_TYPE** – Type of the page. Possible values: index, undo_log, inode, ibuf_free_list, allocated, bitmap, sys, trx_sys, fsp_hdr, xdes, blob, zblob, zblob2, unknown
- **SPACE_ID** – Tablespace ID
- **PAGE_NO** – Page offset within its tablespace
- **LRU_POSITION** – this field is always `0` and will be removed in a future Percona Server release
- **FIX_COUNT** – reference count of a page. It is incremented every time the page is accessed by *InnoDB*, and is 0 if and only if the page is not currently being accessed
- **FLUSH_TYPE** – type of the last flush of the page (0:LRU 2:flush_list)

Example:

```
mysql> select * from information_schema.INNODB_BUFFER_POOL_PAGES LIMIT 20;
+-----------+----------+---------+--------------+-----------+------------+
| page_type | space_id | page_no | lru_position | fix_count | flush_type |
+-----------+----------+---------+--------------+-----------+------------+
| allocated |        0 |       7 |            0 |         0 |          2 |
| allocated |        0 |       1 |            0 |         0 |          0 |
| allocated |        0 |       3 |            0 |         0 |          0 |
| inode     |        0 |       2 |            0 |         0 |          2 |
```

```
| index      |         0 |         4 |              0 |           0 |            2 |
| index      |         0 |        11 |              0 |           0 |            0 |
| index      |         0 |     12956 |              0 |           0 |            0 |
| allocated  |         0 |         5 |              0 |           0 |            2 |
| allocated  |         0 |         6 |              0 |           0 |            2 |
| undo_log   |         0 |        51 |              0 |           0 |            2 |
| undo_log   |         0 |        52 |              0 |           0 |            2 |
| index      |         0 |         8 |              0 |           0 |            0 |
| index      |         0 |       288 |              0 |           0 |            0 |
| index      |         0 |       290 |              0 |           0 |            2 |
| index      |         0 |       304 |              0 |           0 |            0 |
| allocated  |         0 |         0 |              0 |           0 |            2 |
| index      |         0 |        10 |              0 |           0 |            0 |
| index      |         0 |     12973 |              0 |           0 |            0 |
| index      |         0 |         9 |              0 |           0 |            2 |
| index      |         0 |        12 |              0 |           0 |            0 |
+------------+-----------+-----------+----------------+-------------+--------------+
20 rows in set (0.81 sec)
```

This table shows the characteristics of the allocated pages in buffer pool and current state of them.

table INFORMATION_SCHEMA.**INNODB_BUFFER_POOL_PAGES_INDEX**

Columns

- **index_id** – index name
- **space_id** – Tablespace ID
- **page_no** – Page offset within its tablespace
- **n_recs** – number of user records on page
- **data_size** – sum of the sizes of the records in page
- **hashed** – the block is in adaptive hash index (1) or not (0)
- **access_time** – time of the last access to that page
- **modified** – modified since loaded (1) or not (0)
- **dirty** – modified since last flushed (1) or not (0)
- **old** – is old blocks in the LRU list (1) or not (0)
- **lru_position** – page position in the LRU list
- **fix_count** – reference count of a page. It is incremented every time the page is accessed by *InnoDB*, and is 0 if and only if the page is not currently being accessed
- **flush_type** – type of the last flush of the page (0:LRU 2:flush_list)

Example:

```
+----------+----------+---------+--------+-----------+--------+-------------+---------
↪-+-------+-----+--------------+-----------+------------+
| index_id | space_id | page_no | n_recs | data_size | hashed | access_time |␣
↪modified | dirty | old | lru_position | fix_count | flush_type |
+----------+----------+---------+--------+-----------+--------+-------------+---------
↪-+-------+-----+--------------+-----------+------------+
|       39 |        0 |    5787 |    468 |     14976 |      1 |  2636182517 |       ␣
↪1 |     0 |   1 |            0 |         0 |          2 |
|       40 |        0 |    5647 |   1300 |     15600 |      1 |  2636182517 |       ␣
↪1 |     0 |   1 |            0 |         0 |          2 |
```

```
|       39 |         0 |     5786 |     468 |      14976 |       1 |  2636182516 |
↪1 |      0 |    1 |             0 |          0 |           2 |
|       40 |         0 |     6938 |    1300 |      15600 |       1 |  2636193968 |
↪1 |      0 |    1 |             0 |          0 |           2 |
|       39 |         0 |     5785 |     468 |      14976 |       1 |  2636182514 |
↪1 |      0 |    1 |             0 |          0 |           2 |
|       39 |         0 |     5784 |     468 |      14976 |       1 |  2636182512 |
↪1 |      0 |    1 |             0 |          0 |           2 |
|       40 |         0 |     5646 |    1300 |      15600 |       1 |  2636182511 |
↪1 |      0 |    1 |             0 |          0 |           2 |
|       39 |         0 |     7203 |     468 |      14976 |       1 |  2636193967 |
↪1 |      0 |    1 |             0 |          0 |           2 |
|       39 |         0 |     5783 |     468 |      14976 |       1 |  2636182507 |
↪1 |      0 |    1 |             0 |          0 |           2 |
|       39 |         0 |     5782 |     468 |      14976 |       1 |  2636182506 |
↪1 |      0 |    1 |             0 |          0 |           2 |
+----------+-----------+----------+---------+------------+---------+-------------+---------
↪-+--------+------+--------------+------------+-------------+
```

This table shows information about the index pages located in the buffer pool.

table INFORMATION_SCHEMA.**INNODB_BUFFER_POOL_PAGES_BLOB**

Columns

- **space_id** – tablespace ID
- **page_no** – page offset within its tablespace
- **compressed** – contains compressed data (1) or not (0)
- **part_len** – data length in the page
- **next_page_no** – page number of the next data
- **lru_position** – page position in the LRU list
- **fix_count** – reference count of a page. It is incremented every time the page is accessed by InnoDB, and is 0 if and only if the page is not currently being accessed
- **flush_type** – type of the last flush of the page (0:LRU 2:flush_list)

Example:

```
mysql> select * from information_schema.INNODB_BUFFER_POOL_PAGES_BLOB LIMIT 20;
+----------+---------+------------+----------+-------------+-------------+----------
↪-+------------+
| space_id | page_no | compressed | part_len | next_page_no | lru_position | fix_
↪count | flush_type |
+----------+---------+------------+----------+-------------+-------------+----------
↪-+------------+
|     1748 |     111 |          0 |    10137 |           0 |         263 |
↪0 |          2 |
|     1748 |     307 |          0 |     5210 |           0 |        1084 |
↪0 |          2 |
|     1748 |    1329 |          0 |     6146 |           0 |        4244 |
↪0 |          2 |
|     1748 |    1330 |          0 |    11475 |           0 |        4245 |
↪0 |          2 |
|     1748 |    1345 |          0 |     5550 |           0 |        4247 |
↪0 |          2 |
|     1748 |    1346 |          0 |     7597 |           0 |        4248 |
↪0 |          2 |
```

```
|      1748 |     3105 |           0 |      6716 |              0 |           8919 |
↪0 |           2 |
|      1748 |     3213 |           0 |      8170 |              0 |           9390 |
↪0 |           2 |
|      1748 |     6142 |           0 |      5648 |              0 |          19638 |
↪0 |           2 |
|      1748 |     7387 |           0 |     10634 |              0 |          24191 |
↪0 |           2 |
|      1748 |     7426 |           0 |      5355 |              0 |          24194 |
↪0 |           2 |
|      1748 |     7489 |           0 |     16330 |           7489 |          24196 |
↪0 |           2 |
|      1748 |     7490 |           0 |      7126 |              0 |          24197 |
↪0 |           2 |
|      1748 |     7657 |           0 |     13571 |              0 |          24681 |
↪0 |           2 |
|      1748 |     7840 |           0 |     11208 |              0 |          25737 |
↪0 |           2 |
|      1748 |     9599 |           0 |     11882 |              0 |          31989 |
↪0 |           2 |
|      1748 |    11719 |           0 |      7367 |              0 |          40466 |
↪0 |           2 |
|      1748 |    12051 |           0 |     11049 |              0 |          41441 |
↪0 |           2 |
|      1748 |    12052 |           0 |     16330 |          12052 |          41442 |
↪0 |           2 |
|      1748 |    12053 |           0 |      2674 |              0 |          41443 |
↪0 |           2 |
+----------+---------+-------------+-----------+---------------+---------------+-----------
↪-+-------------+
20 rows in set (0.05 sec)
```

This table shows information from blob pages located in buffer pool.

69.3 InnoDB Undo Logs

The purpose of this table is to report on the existence and usage of the internal undo log records. These undo records are stored in standard *InnoDB* pages and are used in a few ways but their main purpose is that currently executing but uncommitted user transactions can be rolled back after either a crash, fast shutdown or other recovery purpose. Each record within the table identifies an *InnoDB* undo segment and will refer to other INFORMATION_SCHEMA tables such as INNODB_TRX and INODB_RSEG. This table can be used to help troubleshoot large system tablespaces and identify run-away or long running transactions.

table INFORMATION_SCHEMA.**INNODB_UNDO_LOGS**

Columns

- **trx_id** – Transaction ID - this is the id of the transaction that has currently allocated the undo segment and will potentially place undo records within it. More information on this transaction can be found by matching the trx_id with that in the INFORMATION_SCHEMA.INNODB_TRX table.
- **rseg_id** – Rollback segment ID associated with this particular undo segment. More info on this rollback segment can be found by matching the rseg_id with that in the INFORMATION_SCHEMA.INNODB_RSEG.
- **useg_id** – Undo segment ID

- **type** – Segment type - identifies what type of operation the segments is allocated for.
- **state** – Segment state
- **size** – Segment size in pages

States of an undo log segment:

- ACTIVE - contains an undo log of an active transaction
- CACHED - cached for quick reuse
- TO_FREE - insert undo segment can be freed
- TO_PURGE - update undo segment will not be reused; it can be freed in purge when all undo data in it is removed
- PREPARED - contains an undo log of a prepared transaction

CHAPTER

SEVENTY

THREAD BASED PROFILING

Percona Server now uses thread based profiling by default, instead of process based profiling. This was implemented because with process based profiling, threads on the server, other than the one being profiled, can affect the profiling information.

Thread based profiling is using the information provided by the kernel getrusage function. Since the 2.6.26 kernel version, thread based resource usage is available with the **RUSAGE_THREAD**. This means that the thread based profiling will be used if you're running the 2.6.26 kernel or newer, or if the **RUSAGE_THREAD** has been ported back.

This feature is enabled by default if your system supports it, in other cases it uses process based profiling.

70.1 Version Specific Information

- *5.5.25a-27.1*: Thread based profiling introduced

CHAPTER

SEVENTYONE

METRICS FOR SCALABILITY MEASUREMENT

Note: This feature has been deprecated in *Percona Server* `5.5.53-38.5`. 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 has implemented extra scalability metrics in `5.5.37-35.0`. These metrics allow using Little's Law, queueing theory, and Universal Scalability Law to gain insights into server performance. This feature is implemented as a plugin.

71.1 Installation

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

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

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

```
SHOW PLUGINS;
```

The plugin should be listed in the output:

```
+-------------------------------+----------+--------------------+--------------------
↪----+---------+
| Name                          | Status   | Type               | Library
↪     | License |
+-------------------------------+----------+--------------------+--------------------
↪----+---------+
...
| scalability_metrics           | ACTIVE   | AUDIT              | scalability_
↪metrics.so | GPL     |
+-------------------------------+----------+--------------------+--------------------
↪----+---------+
```

71.2 System Variables

variable `scalability_metrics_control`

Version Info

- *5.5.37-35.0* – Implemented

Command Line Yes

Scope Global

Dynamic Yes

Variable Type String

Default Value `OFF`

Values `OFF`, `ON`, `RESET`

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

71.3 Status Variables

variable `scalability_metrics_elapsedtime`

Version Info

- *5.5.37-35.0* – Implemented

Variable Type Numeric

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

variable `scalability_metrics_queries`

Version Info

- *5.5.37-35.0* – Implemented

Variable Type Numeric

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

variable `scalability_metrics_concurrency`

Version Info

- *5.5.37-35.0* – Implemented

Variable Type Numeric

This status variable shows number of queries currently executed.

variable `scalability_metrics_totaltime`

Version Info

- *5.5.37-35.0* – Implemented

Variable Type Numeric

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

variable `scalability_metrics_busytime`

Version Info

- *5.5.37-35.0* – Implemented

Variable Type Numeric

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

71.4 Version Specific Information

- *5.5.37-35.0* Scalability Metrics plugin has been implemented in *Percona Server*.
- *5.5.53-38.4* Feature has been removed.
- *5.5.53-38.5* Feature has been added back but deprecated.

71.5 Other Reading

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

Part IX

Obsolete and Removed Features

CHAPTER

SEVENTYTWO

SHARED MEMORY BUFFER POOL

The *SHM buffer pool* patch, which provided the ability to use a shared memory segment for the buffer pool to enable faster server restarts, has been removed. Instead, we recommend using the *LRU Dump/Restore* patch which provides similar improvements in restart performance.

Replacement is due to `SHM` buffer pool both being very invasive and not widely used. Improved restart times are better provided by the much safer `LRU D/R` patch which has the advantage of also persisting across machine restarts.

The configuration variables for `my.cnf` have been kept for compatibility and warnings will be printed for the deprecated options (*innodb_buffer_pool_shm_key* and *innodb_buffer_pool_shm_checksum*) if used.

Instructions for disabling the `SHM` buffer pool can be found *here*.

Instructions on setting up `LRU` dump/restore can be found *here*.

72.1 Version Specific Information

- *5.5.8-20.0*: First Percona Server 5.5 release, also included Shared Memory Buffer Pool.
- *5.5.13-20.4*: Feature removed, as LRU Dump/Restore is less invasive, more reliable and a better solution.

72.2 System Variables

variable innodb_buffer_pool_shm_key

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Default Value OFF

Range ON/OFF

variable innodb_buffer_pool_shm_checksum

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Default Value ON

Range ON/OFF

CHAPTER

SEVENTYTHREE

MULTIPLE ROLLBACK SEGMENTS

Warning: This feature has been removed in *Percona Server* `5.5.11-20.2` because an equivalent variable, innodb_rollback_segments, has been implemented in the upstream version.

In *Percona Server* 5.1, an improvement was provided for write-intensive workloads that allowed multiple rollback segments to be used.

Percona Server, in addition to the upstream multiple rollback segment implementation, provides the additional Information Schema table: *INNODB_RSEG*.

73.1 INFORMATION_SCHEMA Tables

This feature provides the following table:

table INFORMATION_SCHEMA.**INNODB_RSEG**

Columns

- **rseg_id** – rollback segment id
- **space_id** – space where the segment placed
- **zip_size** – compressed page size in bytes if compressed otherwise 0
- **page_no** – page number of the segment header
- **max_size** – max size in pages
- **curr_size** – current size in pages

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

Here is an example of output with `innodb_extra_rsegments = 8`

```
mysql> select * from information_schema.innodb_rseg;
+---------+----------+----------+---------+------------+-----------+
| rseg_id | space_id | zip_size | page_no | max_size   | curr_size |
+---------+----------+----------+---------+------------+-----------+
|       0 |        0 |        0 |       6 | 4294967294 |         1 |
|       1 |        0 |        0 |      13 | 4294967294 |         2 |
|       2 |        0 |        0 |      14 | 4294967294 |         1 |
|       3 |        0 |        0 |      15 | 4294967294 |         1 |
|       4 |        0 |        0 |      16 | 4294967294 |         1 |
|       5 |        0 |        0 |      17 | 4294967294 |         1 |
|       6 |        0 |        0 |      18 | 4294967294 |         1 |
```

```
|       7 |         0 |         0 |      19 | 4294967294 |          1 |
|       8 |         0 |         0 |      20 | 4294967294 |          1 |
+---------+-----------+-----------+---------+------------+------------+
9 rows in set (0.00 sec)
```

Part X

Reference

CHAPTER

SEVENTYFOUR

LIST OF UPSTREAM *MYSQL* BUGS FIXED IN *PERCONA SERVER* 5.5

Upstream bug #90111 - Incorrect enum comparisons
JIRA bug #3893
Upstream state Verified (checked on 2018-05-17)
Fix Released `5.5.60-38.12`
Upstream fix N/A

Upstream bug #90264 - Some file operations in mf_iocache2.c are not instrumented
JIRA bug #3937
Upstream state Verified (checked on 2018-05-17)
Fix Released `5.5.60-38.12`
Upstream fix N/A

Upstream bug #90238 - Comparison of uninitailized memory in log_in_use
JIRA bug #3925
Upstream state No Feedback (checked on 2018-05-17)
Fix Released `5.5.60-38.12`
Upstream fix N/A

Upstream bug #89776 - a typo in *cmake/plugin.cmake* prevents *MYSQL_SERVER* to be defined...
JIRA bug #3871
Upstream state Verified (checked on 2018-05-17)
Fix Released `5.5.60-38.12`
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 2018-05-24)
Fix Released `5.5.59-38.11`
Upstream fix N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #83264 - uint3korr should stop reading four instead of three bytes on x86 **JIRA bug** #3567 **Upstream state** Closed **Fix Released** *5.5.52-38.3* **Upstream fix** 5.5.57
Upstream bug #68052 - SSL Certificate Subject ALT Names with IPs not respected with –ssl-ver... **JIRA bug** #1076 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.57-38.9* **Upstream fix** N/A
Upstream bug #83814 - Add support for OpenSSL 1.1 **JIRA bug** #1105 **Upstream state** Closed **Fix Released** *5.5.57-38.9* **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 2018-05-24) **Fix Released** *5.5.57-38.9* **Upstream fix** N/A
Upstream bug #83073 - GCC 5 and 6 miscompile mach_parse_compressed **JIRA bug** #1745 **Upstream state** Closed **Fix Released** 5.5.52-38.2 **Upstream fix** 5.5.54
Upstream bug #56155 - 'You cannot 'ALTER' a log table if logging is enabled' even if I log to... **JIRA bug** #595 **Upstream state** Closed **Fix Released** 5.5.52-38.2 **Upstream fix** N/A
Upstream bug #79249 - main.group_min_max fails under Valgrind **JIRA bug** #1668 **Upstream state** Closed **Fix Released** *5.5.51-38.1* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #82019 - Is client library supposed to retry EINTR indefinitely or not **JIRA bug** #1720 **Upstream state** Closed **Fix Released** *5.5.51-38.1* **Upstream fix** 5.5.52
Upstream bug #79610 - Failed DROP DATABASE due FK constraint on master breaks slave **JIRA bug** #1683 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.51-38.1* **Upstream fix** N/A
Upstream bug #81657 - DBUG_PRINT in THD::decide_logging_format prints incorrectly, access ... **JIRA bug** #2150 **Upstream state** Closed **Fix Released** *5.5.50-38.0* **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** *5.5.50-38.0* **Upstream fix** N/A
Upstream bug #78223 - memory leak in mysqlbinlog **JIRA bug** #3440 **Upstream state** Closed **Fix Released** *5.5.50-38.0* **Upstream fix** N/A
Upstream bug #81675 - mysqlbinlog does not free the existing connection before opening new... **JIRA bug** #1718 **Upstream state** Closed **Fix Released** *5.5.50-38.0* **Upstream fix** 5.5.52
Upstream bug #61619 - ssl.cmake file is broken when using custom OpenSSL build **JIRA bug** #3437 **Upstream state** Closed **Fix Released** *5.5.50-38.0* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #80014 - mysql build fails, memory leak in gen_lex_hash, clang address sanitizer **JIRA bug** #3433 **Upstream state** Closed **Fix Released** *5.5.50-38.0* **Upstream fix** N/A
Upstream bug #75239 - Support for TLSv1.1 and TLSv1.2 **JIRA bug** #926 **Upstream state** Closed **Fix Released** *5.5.50-38.0* **Upstream fix** N/A
Upstream bug #81324 - "rpl.rpl_start_stop_slave" fail sporadically on 5.5 **JIRA bug** #2145 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.49-37.9* **Upstream fix** N/A
Upstream bug #81295 - main.bigint/rpl.rpl_stm_user_variables fail on Ubuntu 15.10 Wily in ... **JIRA bug** #3427 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.49-37.9* **Upstream fix** N/A
Upstream bug #79185 - Innodb freeze running REPLACE statements **JIRA bug** #945 **Upstream state** Closed **Fix Released** *5.5.46-37.6* **Upstream fix** 5.5.49
Upstream bug #77275 - Newest RHEL/CentOS openssl update breaks mysql DHE ciphers **JIRA bug** #906 **Upstream state** Closed **Fix Released** *5.5.44-37.3* **Upstream fix** 5.5.45
Upstream bug #66779 - innochecksum does not work with compressed tables **JIRA bug** #1302 **Upstream state** Closed **Fix Released** *5.5.44-37.3* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #75480 - Selecting wrong pos with SHOW BINLOG EVENTS causes a potentially ... **JIRA bug** #1600 **Upstream state** N/A **Fix Released** *5.5.44-37.3* **Upstream fix** N/A
Upstream bug #69991 - MySQL client is broken without readline **JIRA bug** #1467 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.43-37.2* **Upstream fix** N/A
Upstream bug #63130 - CMake-based check for the presence of a system readline library is ... **JIRA bug** #1467 **Upstream state** Can't repeat (checked on 2018-05-24) **Fix Released** *5.5.43-37.2* **Upstream fix** N/A
Upstream bug #75868 - main.error_simulation fails on Mac OS X since 5.5.42 **JIRA bug** #3266 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.42-37.1* **Upstream fix** N/A
Upstream bug #75642 - Extend valid range of dummy certificates ni mysql-test/std_data **JIRA bug** #1605 **Upstream state** Closed **Fix Released** *5.5.42-37.1* **Upstream fix** 5.5.42
Upstream bug #53645 - SHOW GRANTS not displaying all the applicable grants **JIRA bug** #191 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.42-37.1* **Upstream fix** N/A
Upstream bug #74987 - mtr failure on Ubuntu Utopic, mysqlhotcopy fails with wrong error(255) **JIRA bug** #2102 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.41-37.0* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #73281 - openssl_1 tries to test a removed cipher on CentOS 7 **JIRA bug** #3242 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.41-37.0* **Upstream fix** N/A
Upstream bug #74440 - mysql_install_db not handling mysqld startup failure **JIRA bug** #1553 **Upstream state** Won't fix **Fix Released** *5.5.41-37.0* **Upstream fix** N/A
Upstream bug #72475 - Binlog events with binlog_format=MIXED are unconditionally logged in ROW.. **JIRA bug** #151 **Upstream state** Closed **Fix Released** *5.5.41-37.0* **Upstream fix** N/A
Upstream bug #75000 - 5.5 fails to compile with debug on Ubuntu Utopic **JIRA bug** #3236 **Upstream state** Closed **Fix Released** *5.5.41-37.0* **Upstream fix** 5.5.42
Upstream bug #73979 - wrong stack size calculation leads to stack overflow in pinbox allocator **JIRA bug** #807 **Upstream state** Closed **Fix Released** *5.5.41-37.0* **Upstream fix** N/A
Upstream bug #73126 - Numerous Valgrind errors in OpenSSL **JIRA bug** #3160 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.39-36.0* **Upstream fix** N/A
Upstream bug #73111 - Suppression typo causing spurious MTR Valgrind failures **JIRA bug** #3159 **Upstream state** Open (checked on 2018-05-24) **Fix Released** *5.5.39-36.0* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #73418 - Add –manual-lldb option to mysql-test-run.pl **JIRA bug** #2448 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.39-36.0* **Upstream fix** N/A
Upstream bug #60782 - Audit plugin API: no MYSQL_AUDIT_GENERAL_LOG notifications with general... **JIRA bug** #1369 **Upstream state** Closed **Fix Released** *5.5.37-35.0* **Upstream fix** N/A
Upstream bug #67352 - table_id is defined differently in sql/table.h vs sql/log_event.h **JIRA bug** #142 **Upstream state** Closed **Fix Released** *5.5.37-35.0* **Upstream fix** N/A
Upstream bug #71250 - Bison 3 breaks mysql build **JIRA bug** #376 **Upstream state** Closed **Fix Released** *5.5.37-35.0* **Upstream fix** N/A
Upstream bug #71375 - Slave IO thread won't attempt auto reconnect to the master/error-code 1593 **JIRA bug** #3086 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.36-34.1* **Upstream fix** N/A
Upstream bug #71374 - Slave IO thread won't attempt auto reconnect to the master/error-code 1159 **JIRA bug** #1470 **Upstream state** N/A **Fix Released** *5.5.36-34.1* **Upstream fix** N/A
Upstream bug #71089 - CMake warning when generating Makefile **JIRA bug** #2059 **Upstream state** Closed **Fix Released** *5.5.36-34.0* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #54430 - innodb should retry partial reads/writes where errno was 0 **JIRA bug** #1460 **Upstream state** Closed **Fix Released** *5.5.36-34.0* **Upstream fix** N/A
Upstream bug #62311 - segfault in mysqld during early SIGHUP handling **JIRA bug** #3068 **Upstream state** Closed **Fix Released** *5.5.36-34.0* **Upstream fix** N/A
Upstream bug #41975 - Support for SSL options not included in mysqlbinlog **JIRA bug** #1393 **Upstream state** Closed **Fix Released** *5.5.35-33.0* **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** *5.5.33-31.1* **Upstream fix** N/A
Upstream bug #68354 - Server crashes on update/join FEDERATED + local table when only 1 local... **JIRA bug** #96 **Upstream state** N/A **Fix Released** *5.5.32-31.0* **Upstream fix** N/A
Upstream bug #42415 - UPDATE/DELETE with LIMIT clause unsafe for SBL even with ORDER BY PK ... **JIRA bug** #44 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.32-31.0* **Upstream fix** N/A
Upstream bug #69179 - accessing information_schema.partitions causes plans to change **JIRA bug** #680 **Upstream state** Closed **Fix Released** *5.5.32-31.0* **Upstream fix** 5.5.34

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #68970 - fsp_reserve_free_extents switches from small to big tblspace handling ... **JIRA bug** #656 **Upstream state** Closed **Fix Released** *5.5.32-31.0* **Upstream fix** N/A
Upstream bug #79117 - "change_user" command should be aware of preceding "error" command **JIRA bug** #659 **Upstream state** Closed **Fix Released** *5.5.31-30.3* **Upstream fix** N/A
Upstream bug #65077 - internal temporary tables are contended on THR_LOCK_myisam **JIRA bug** #1362 **Upstream state** Closed **Fix Released** *5.5.31-30.3* **Upstream fix** N/A
Upstream bug #68999 - SSL_OP_NO_COMPRESSION not defined **JIRA bug** #362 **Upstream state** Closed **Fix Released** *5.5.31-30.3* **Upstream fix** 5.5.44
Upstream bug #68197 - InnoDB reports that it's going to wait for I/O but the I/O is async **JIRA bug** #362 **Upstream state** Closed **Fix Released** *5.5.30-30.2* **Upstream fix** 5.5.31
Upstream bug #68845 - Unnecessary log_sys->mutex reacquisition in mtr_log_reserve_and_write() **JIRA bug** #1347 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.30-30.2* **Upstream fix** N/A
Upstream bug #62578 - mysql client aborts connection on terminal resize **JIRA bug** #84 **Upstream state** Won't Fix **Fix Released** *5.5.30-30.2* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #49169 - read_view_open_now is inefficient with many concurrent sessions **JIRA bug** #636 and #637 **Upstream state** Closed **Fix Released** *5.5.30-30.2* **Upstream fix** N/A
Upstream bug #63144 - CREATE TABLE IF NOT EXISTS metadata lock is too restrictive **JIRA bug** #40 **Upstream state** Closed **Fix Released** *5.5.30-30.2* **Upstream fix** N/A
Upstream bug #68477 - Suboptimal code in skip_trailing_space() **JIRA bug** #1321 **Upstream state** Closed **Fix Released** *5.5.30-30.1* **Upstream fix** N/A
Upstream bug #68476 - Suboptimal code in my_strnxfrm_simple() **JIRA bug** #1320 **Upstream state** Closed **Fix Released** *5.5.30-30.1* **Upstream fix** N/A
Upstream bug #68116 - InnoDB monitor may hit an assertion error in buf_page_get_gen in debug ... **JIRA bug** #616 **Upstream state** Closed **Fix Released** *5.5.29-30.0* **Upstream fix** N/A
Upstream bug #67504 - Duplicate error in replication with slave triggers and auto increment **JIRA bug** #34 **Upstream state** Closed **Fix Released** *5.5.29-30.0* **Upstream fix** N/A
Upstream bug #67983 - Memory leak on filtered slave **JIRA bug** #581 **Upstream state** Closed **Fix Released** *5.5.29-30.0* **Upstream fix** 5.5.31

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #67974 - Server crashes in add_identifier on concurrent ALTER TABLE and SHOW ENGINE **JIRA bug** #344 **Upstream state** N/A **Fix Released** *5.5.29-30.0* **Upstream fix** N/A
Upstream bug #68045 - security vulnerability CVE-2012-4414 **JIRA bug** #348 **Upstream state** N/A **Fix Released** *5.5.29-29.4* **Upstream fix** N/A
Upstream bug #70277 - last argument of LOAD DATA ... SET ... statement repeated twice in binlog **JIRA bug** #3020 **Upstream state** Closed **Fix Released** *5.5.28-29.3* **Upstream fix** 5.5.35
Upstream bug #69380 - Incomplete fix for security vulnerability CVE-2012-5611 **JIRA bug** #666 **Upstream state** N/A **Fix Released** *5.5.28-29.3* **Upstream fix** N/A
Upstream bug #66550 - security vulnerability CVE-2012-4414 **JIRA bug** #348 **Upstream state** N/A **Fix Released** *5.5.28-29.3* **Upstream fix** N/A
Upstream bug #67685 - security vulnerability CVE-2012-5611 **JIRA bug** #350 **Upstream state** N/A **Fix Released** *5.5.28-29.3* **Upstream fix** N/A
Upstream bug #66237 - Temporary files created by binary log cache are not purged after transa... **JIRA bug** #599 **Upstream state** Closed **Fix Released** *5.5.28-29.3* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #69124 - Incorrect truncation of long SET expression in LOAD DATA can cause SQL ... **JIRA bug** #663 **Upstream state** N/A **Fix Released** *5.5.28-29.3* **Upstream fix** N/A
Upstream bug #71603 - file name is not escaped in binlog for LOAD DATA INFILE statement **JIRA bug** #3092 **Upstream state** N/A **Fix Released** *5.5.28-29.3* **Upstream fix** N/A
Upstream bug #67606 - MySQL crashes with segmentation fault when disk quota is reached **JIRA bug** #1948 **Upstream state** Duplicate **Fix Released** *5.5.28-29.3* **Upstream fix** N/A
Upstream bug #67737 - mysqldump test sometimes fails due to mixing stdout and stderr **JIRA bug** #547 **Upstream state** Closed **Fix Released** *5.5.28-29.2* **Upstream fix** 5.5.29
Upstream bug #66890 - Slave server crash after a START SLAVE **JIRA bug** #587 **Upstream state** Duplicate **Fix Released** *5.5.28-29.1* **Upstream fix** 5.5.29
Upstream bug #62856 - Check for “stack overrun” doesn’t work with gcc-4.6, server crashes **JIRA bug** #2795 **Upstream state** Closed **Fix Released** *5.5.28-29.1* **Upstream fix** N/A
Upstream bug #61180 - korr/store macros in my_global.h assume the argument to be a char pointer **JIRA bug** #2795 **Upstream state** Closed **Fix Released** *5.5.27-29.0* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #61178 - Incorrect implementation of intersect(ulonglong) in non-optimized Bitmap.. **JIRA bug** #2795 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.27-29.0* **Upstream fix** N/A
Upstream bug #54127 - mysqld segfaults when built using –with-max-indexes=128 **JIRA bug** #2795 **Upstream state** Closed **Fix Released** *5.5.27-29.0* **Upstream fix** N/A
Upstream bug #64800 - mysqldump with –include-master-host-port putting quotes around port no. **JIRA bug** #1923 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.27-28.0* **Upstream fix** N/A
Upstream bug #66301 - INSERT ... ON DUPLICATE KEY UPDATE + innodb_autoinc_lock_mode=1 is broken **JIRA bug** #576 **Upstream state** Closed **Fix Released** *5.5.27-28.0* **Upstream fix** N/A
Upstream bug #60743 - typo in cmake/dtrace.cmake **JIRA bug** #1924 **Upstream state** Closed **Fix Released** *5.5.25a-27.1* **Upstream fix** 5.5.33
Upstream bug #64663 - Segfault when adding indexes to InnoDB temporary tables **JIRA bug** #557 **Upstream state** N/A **Fix Released** *5.5.24-26.0* **Upstream fix** N/A
Upstream bug #64624 - Mysql is crashing during replication **JIRA bug** #535 **Upstream state** Closed **Fix Released** *5.5.24-26.0* **Upstream fix** 5.5.26

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #64160 - page size 1024 but the only supported page size in this release is=16384 **JIRA bug** #549 **Upstream state** Closed **Fix Released** *5.5.21-25.1* **Upstream fix** 5.5.22
Upstream bug #64432 - Bug #54330 (Broken fast index creation) was never fixed in 5.5 **JIRA bug** #544 **Upstream state** Closed **Fix Released** *5.5.21-25.0* **Upstream fix** 5.5.30
Upstream bug #49336 - mysqlbinlog does not accept input from stdin when stdin is a pipe **JIRA bug** #541 **Upstream state** Closed **Fix Released** *5.5.21-25.0* **Upstream fix** 5.5.28
Upstream bug #71183 - os_file_fsync() should handle fsync() returning EINTR **JIRA bug** #1461 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.20-24.1* **Upstream fix** N/A
Upstream bug #63451 - atomic/x86-gcc.h:make_atomic_cas_body64 potential miscompilation bug **JIRA bug** #508 **Upstream state** Closed **Fix Released** *5.5.18-23.0* **Upstream fix** N/A
Upstream bug #62557 - SHOW SLAVE STATUS gives wrong output with master-master and using SET... **JIRA bug** #2692 **Upstream state** Closed **Fix Released** *5.5.17-22.1* **Upstream fix** 5.5.28
Upstream bug #45702 - Impossible to specify myisam_sort_buffer > 4GB on 64 bit machines **JIRA bug** #2700 **Upstream state** Closed **Fix Released** *5.5.17-22.1* **Upstream fix** 5.5.22

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #62516 - Fast index creation does not update index statistics **JIRA bug** #2686 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.16-22.0* **Upstream fix** N/A
Upstream bug #25007 - memory tables with dynamic rows format **JIRA bug** #2407 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.15-21.0* **Upstream fix** N/A
Upstream bug #51196 - Slave SQL: Got an error writing communication packets, Error_code: 1160 **JIRA bug** #2666 **Upstream state** Closed **Fix Released** *5.5.14-20.5* **Upstream fix** 5.5.21
Upstream bug #43593 - dump/backup/restore/upgrade tools fails because of utf8_general_ci **JIRA bug** N/A **Upstream state** Closed **Fix Released** *5.5.14-20.5* **Upstream fix** 5.5.21
Upstream bug #61595 - mysql-test/include/wait_for_slave_param.inc timeout logic is incorrect **JIRA bug** #485 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.13-20.4* **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 2018-05-24) **Fix Released** *5.5.11-20.2* **Upstream fix** N/A
Upstream bug #57583 - fast index create not used during “alter table foo engine=innodb” **JIRA bug** #2619 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.11-20.2* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #49120 - mysqldump should have flag to delay creating indexes for innodb plugin... **JIRA bug** #2619 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.11-20.2* **Upstream fix** N/A
Upstream bug #54160 - InnoDB should retry on failed read or write, not immediately panic **JIRA bug** #2628 **Upstream state** Closed **Fix Released** *5.5.11-20.2* **Upstream fix** N/A
Upstream bug #51325 - Dropping an empty innodb table takes a long time with large buffer pool **JIRA bug** none **Upstream state** Closed **Fix Released** *5.5.10-20.1* **Upstream fix** 5.5.20
Upstream bug #56433 - Auto-extension of InnoDB files **JIRA bug** none **Upstream state** Closed **Fix Released** *5.5.10-20.1* **Upstream fix** N/A
Upstream bug #20001 - Support for temp-tables in INFORMATION_SCHEMA **JIRA bug** none **Upstream state** Closed **Fix Released** *5.5.8-20.0* **Upstream fix** N/A
Upstream bug #69146 - Optimization in buf_pool_get_oldest_modification if srv_buf_pool_instances **JIRA bug** #2418 **Upstream state** Verified (checked on 2018-05-24) **Fix Released** *5.5.8-20.0* **Upstream fix** N/A
Upstream bug #54814 - make BUF_READ_AHEAD_AREA a constant **JIRA bug** #1148 **Upstream state** Closed **Fix Released** *5.5.8-20.0* **Upstream fix** N/A

Continued on next page

Table 74.1 – continued from previous page

Upstream bug #75534 - Solve buffer pool mutex contention by splitting it **JIRA bug** *Improved Buffer Pool Scalability* **Upstream state** Closed **Fix Released** `5.5.8-20.0` **Upstream fix** N/A
Upstream bug #54790 - Use of non-blocking mode for sockets limits performance **JIRA bug** #1147 **Upstream state** Closed **Fix Released** `5.5.8-20.0` **Upstream fix** N/A
Upstream bug #67879 - Slave deadlock caused by stop slave, show slave status and global read... **Launchpad BP** *Lock-Free SHOW SLAVE STATUS* **Upstream state** Closed **Fix Released** `5.5.8-20.0` **Upstream fix** N/A
Upstream bug #56676 - 'show slave status' ,'show global status' hang when 'stop slave' takes... **Launchpad BP** *Lock-Free SHOW SLAVE STATUS* **Upstream state** Closed **Fix Released** `5.5.8-20.0` **Upstream fix** N/A

CHAPTER

SEVENTYFIVE

LIST OF VARIABLES INTRODUCED IN PERCONA SERVER 5.5

75.1 System Variables

Name	Cmd-Line	Option File	Var Scope	Dynamic
csv_mode	Yes	Yes	Both	Yes
enforce_storage_engine	No	Yes	Global	No
expand_fast_index_creation	Yes	No	Both	Yes
extra_max_connections	Yes	Yes	Global	Yes
extra_port	Yes	Yes	Global	No
fast_index_creation	Yes	No	Session	Yes
have_flashcache	No	No	Global	No
have_response_time_distribution	No	No	Global	No
innodb_adaptive_flushing_method	Yes	Yes	Global	Yes
innodb_adaptive_hash_index_partitions	Yes	Yes	Global	No
innodb_blocking_buffer_pool_restore	Yes	Yes	Global	No
innodb_buffer_pool_populate	Yes	Yes	Global	No
innodb_buffer_pool_restore_at_startup	Yes	Yes	Global	Yes
innodb_buffer_pool_shm_checksum	Yes	Yes	Global	No
innodb_buffer_pool_shm_key	Yes	Yes	Global	No
innodb_checkpoint_age_target	Yes	Yes	Global	Yes
innodb_corrupt_table_action	Yes	Yes	Global	Yes
innodb_dict_size_limit	Yes	Yes	Global	Yes
innodb_doublewrite_file	Yes	Yes	Global	No
innodb_fake_changes	Yes	Yes	Both	Yes
innodb_fast_checksum	Yes	Yes	Global	No
innodb_flush_neighbor_pages	Yes	Yes	Global	Yes
innodb_ibuf_accel_rate	Yes	Yes	Global	Yes
innodb_ibuf_active_contract	Yes	Yes	Global	Yes
innodb_ibuf_max_size	Yes	Yes	Global	No
innodb_import_table_from_xtrabackup	Yes	Yes	Global	Yes
innodb_kill_idle_transaction	Yes	Yes	Global	Yes
innodb_lazy_drop_table	Yes	Yes	Global	Yes
innodb_locking_fake_changes	Yes	Yes	Both	Yes
innodb_log_block_size	Yes	Yes	Global	No
innodb_max_bitmap_file_size	Yes	Yes	Global	Yes
innodb_max_changed_pages	Yes	Yes	Global	Yes
innodb_merge_sort_block_size	Yes	Yes	Global	Yes

Continued on next page

Table 75.1 – continued from previous page

Name	Cmd-Line	Option File	Var Scope	Dynamic
innodb_page_size	Yes	Yes	Global	No
innodb_read_ahead	Yes	Yes	Global	Yes
innodb_recovery_stats	No	Yes	Global	No
innodb_recovery_update_relay_log	Yes	Yes	Global	No
innodb_show_locks_held	Yes	Yes	Global	Yes
innodb_show_verbose_locks	Yes	Yes	Global	Yes
innodb_stats_auto_update	Yes	Yes	Global	Yes
innodb_stats_update_need_lock	Yes	Yes	Global	Yes
innodb_thread_concurrency_timer_based	Yes	Yes	Global	No
innodb_track_changed_pages	Yes	Yes	Global	No
innodb_use_atomic_writes	Yes	Yes	Global	No
innodb_use_global_flush_log_at_trx_commi	Yes	Yes	Global	Yes
innodb_use_sys_stats_table	Yes	Yes	Global	No
log_slow_admin_statements	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_slave_statements	Yes	Yes	Both	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
optimizer_fix	N/A	N/A	N/A	N/A
query_cache_strip_comments	Yes	Yes	Global	Yes
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_timestamp_always	Yes	Yes	Global	Yes
slow_query_log_timestamp_precision	Yes	Yes	Global	Yes
slow_query_log_use_global_control	Yes	Yes	Global	Yes
thread_pool_high_prio_mode	Yes	Yes	Both	Yes
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
userstat	Yes	Yes	Global	Yes

75.2 Status Variables

Name	Var Type	Var Scope
Com_show_client_statistics	Numeric	Both

Continued on next page

Table 75.2 – continued from previous page

Name	Var Type	Var Scope
Com_show_index_statistics	Numeric	Both
Com_show_slave_status_nolock	Numeric	Both
Com_show_table_statistics	Numeric	Both
Com_show_temporary_tables	Numeric	Both
Com_show_thread_statistics	Numeric	Both
Com_show_user_statistics	Numeric	Both
Flashcache_enabled	Boolean	Global
Innodb_adaptive_hash_cells	Numeric	Global
Innodb_adaptive_hash_heap_buffers	Numeric	Global
Innodb_adaptive_hash_hash_searches	Numeric	Global
Innodb_adaptive_hash_non_hash_searches	Numeric	Global
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_checkpoint_target_age	Numeric	Global
Innodb_deadlocks	Numeric	Global
Innodb_dict_tables	Numeric	Global
Innodb_history_list_length	Numeric	Global
Innodb_ibuf_discarded_delete_marks	Numeric	Global
Innodb_ibuf_discarded_deletes	Numeric	Global
Innodb_ibuf_discarded_inserts	Numeric	Global
Innodb_ibuf_free_list	Numeric	Global
Innodb_ibuf_merged_delete_marks	Numeric	Global
Innodb_ibuf_merged_deletes	Numeric	Global
Innodb_ibuf_merged_inserts	Numeric	Global
Innodb_ibuf_merges	Numeric	Global
Innodb_ibuf_segment_size	Numeric	Global
Innodb_ibuf_size	Numeric	Global
Innodb_lsn_current	Numeric	Global
Innodb_lsn_flushed	Numeric	Global
Innodb_lsn_last_checkpoint	Numeric	Global
Innodb_master_thread_1_second_loops	Numeric	Global
Innodb_master_thread_10_second_loops	Numeric	Global
Innodb_master_thread_background_loops	Numeric	Global
Innodb_master_thread_main_flush_loops	Numeric	Global
Innodb_master_thread_sleeps	Numeric	Global
Innodb_max_trx_id	Numeric	Global
Innodb_mem_adaptive_hash	Numeric	Global
Innodb_mem_dictionary	Numeric	Global
Innodb_mem_total	Numeric	Global
Innodb_mutex_os_waits	Numeric	Global
Innodb_mutex_spin_rounds	Numeric	Global
Innodb_mutex_spin_waits	Numeric	Global
Innodb_oldest_view_low_limit_trx_id	Numeric	Global
Innodb_purge_trx_id	Numeric	Global

Continued on next page

Table 75.2 – continued from previous page

Name	Var Type	Var Scope
Innodb_purge_undo_no	Numeric	Global
Innodb_current_row_locks	Numeric	Global
Innodb_read_views_memory	Numeric	Global
Innodb_descriptors_memory	Numeric	Global
Innodb_s_lock_os_waits	Numeric	Global
Innodb_s_lock_spin_rounds	Numeric	Global
Innodb_s_lock_spin_waits	Numeric	Global
Innodb_x_lock_os_waits	Numeric	Global
Innodb_x_lock_spin_rounds	Numeric	Global
Innodb_x_lock_spin_waits	Numeric	Global
Threadpool_idle_threads	Numeric	Global
Threadpool_threads	Numeric	Global
binlog_commits	Numeric	Session
binlog_group_commits	Numeric	Session

CHAPTER

SEVENTYSIX

DEVELOPMENT OF PERCONA SERVER

Percona Server is an open source project to produce a distribution of the *MySQL* server with improved performance, scalability and diagnostics.

76.1 Submitting Changes

We keep trunk in a constant state of stability to allow for a release at any time and to minimize wasted time by developers due to broken code from somebody else interfering with their day.

You should also be familiar with our *Jenkins* setup.

76.1.1 Overview

At Percona we use Bazaar for source control and launchpad for both code hosting and release management.

Changes to our software projects could be because of a new feature (blueprint) or fixing a bug (bug). Projects such as refactoring could be classed as a blueprint or a bug depending on the scope of the work.

Blueprints and bugs are targeted to specific milestones (releases). A milestone is part of a series - e.g. 1.6 is a series in Percona XtraBackup and 1.6.1, 1.6.2 and 1.6.3 are milestones in the 1.6 series.

Code is proposed for merging in the form of merge requests on launchpad.

Some software (such as Percona Xtrabackup) we maintain both a development branch and a stable branch. For example: Xtrabackup 1.6 is the current stable series, and changes that should make it into bugfix releases of 1.6 should be proposed for the 1.6 tree. However, most new features or more invasive (or smaller) bug fixes should be targeted to the next release, currently 1.7. If submitting something to 1.6, you should also propose a branch that has these changes merged to the development release (1.7). This way somebody else doesn't have to attempt to merge your code and we get to run any extra tests that may be in the tree (and check compatibility with all platforms).

For Percona Server, we have two current bzr branches on which development occurs: 5.1 and 5.5. As Percona Server is not a traditional project, instead being a set of patches against an existing product, these two branches are not related. That is, we do not merge from one to the other. To have your changes in both, you must propose two branches: one for 5.1 version of patch and one for 5.5.

76.1.2 Making a change to a project

In this case we're going to use percona-xtrabackup as an example. workflow is similar for Percona Server, but patch will need to be modified both in 5.1 and 5.5 branches.

- `bzr branch lp:percona-xtrabackup featureX` (where 'featureX' is a sensible name for the task at hand)

- (developer makes changes in featureX, testing locally)
- Developer pushes to `lp:~username/percona-xtrabackup/featureX`
- When the developer thinks the branch may be ready to be merged, they will run the branch through param build.
- If there are any build or test failures, developer fixes them (in the case of failing tests in trunk... no more tests should fail. Eventually all tests will pass in trunk)
- Developer can then submit a merge proposal to lp:percona-xtrabackup, referencing URL for the param build showing that build and test passes
- Code undergoes review
- Once code is accepted, it can be merged (see other section)

If the change also applies to a stable release (e.g. 1.6) then changes should be made on a branch of 1.6 and merged to a branch of trunk. In this case there should be two branches run through param build and two merge proposals (one for 1.6 and one with the changes merged to trunk). This prevents somebody else having to guess how to merge your changes.

76.1.3 Merging approved branches

Before code hits trunk, it goes through a "staging" branch, where some extra tests may be run (e.g. valgrind) along with testing that all branches behave well together (build and test) before pushing to trunk.

To ensure quality, **DO NOT push directly to trunk!** everything must go through adequate testing first. This ensures that at any point trunk is in a releasable state.

Please note that **ALL changes must go through staging first** This is to ensure that several approved merge requests do not interact badly with each other.

- Merge captain (for lack of a better term for the person merging approved code into trunk) may collate several approved branches that have individually passed param-build as run by the original developers.
  - Workflow would look something like this:
    * `bzr branch lp:percona-xtrabackup staging`
    * `bzr merge lp:~user/percona-xtrabackup/featureX`
    * `bzr commit -m "merge feature X"`
    * `bzr merge lp:~user/percona-xtrabackup/featureY`
    * `bzr commit -m "merge feature Y"`
    * `bzr push --overwrite lp:percona-xtrabackup/staging'`
    * Run `lp:percona-xtrabackup/staging` through param build (in future, we'll likely have a Jenkins job specifically for this)
    * If build succeeds, `bzr push lp:percona-server` (and branches will be automatically marked as 'merged'.. although bug reports will need to be manually changed to 'Fix Released')
    * If build or test fails, attempt to find which branch may be the cause, and repeat process but without that branch.
- Any failing branch will be set to 'Work in Progress' with a 'Needs fixing' review with the URL of the build in jenkins where the failure occured. This will allow developers to fix their code.

76.1.4 Resubmitting a merge request

In the event of a merge request being marked as 'Work In Progress' due to build/test failures when merging, the developer should fix up the branch, run through param build and then 'Resubmit' the merge proposal.

There is a link on launchpad to resubmit the merge proposal, this means it appears in the list of merge requests to review again rather than off in the "work in progress" section.

76.1.5 Percona Server

The same process for Percona Server, but we have different branches (and merge requests) for 5.1 and 5.5 series.

76.1.6 Upgrading MySQL base version

- Same process as other modifications.
- create local branch
- make changes
- param build
- merge request

We will need some human processes to ensure that we do not merge extra things during the time when base MySQL version is being updated to avoid making life harder for the person doing the update.

76.2 Making a release

- `bzr branch lp:project release-project-VERSION`
- build packages
- perform any final tests (as we transition, this will already have been done by jenkins)
- `bzr tag project-version`
- merge request back to lp:project including the tag (TODO: write exact bzr commands for this)

This way anybody can easily check out an old release by just using bzr to branch the specific tag.

76.3 Jenkins

Our Jenkins instance uses a mixture of VMs on physical hosts that Percona runs and Virtual Machines in Amazon EC2 that are launched on demand.

76.3.1 Basic Concepts

We have some jobs that are activated based on source control changes (new commits in a bzr repository). We have some that are "param build" - that is, a user specifies parameters for the build (e.g. the bzr tree). A param-build allows developers to ensure their branch compiles and passes tests on all supported platforms *before* submitting a merge request. This helps us maintain the quality of the main bzr branches and not block other developers work.

Jenkins is a Master/Slave system and the jenkins master schedules the builds across available machines (and may launch new VMs in EC2 to meet demand).

Most of our jobs are what's known as "matrix builds". That is, a job that will be run with several different configurations of the project (e.g. release, debug) across several platforms (e.g. on a host matching the label of "centos5-32" and a host matching label of "ubuntu-natty-32bit"). Matrix builds show a table of lights to indicate their status. Clicking "build now" on one of these queues up builds for all of the combinations.

We have some integration of our regression test suites (currently xtrabackup) with Jenkins ability to parse JUnitXML, presenting a nice user interface to any test failures.

Because building some projects is non-trivial, in order to not duplicate the list of compile instructions for each job, we use template builds. You'll see builds such as percona-xtrabackup-template which is a disabled job, but all current xtrabackup jobs point to it for the commands to build and run the test suite.

76.3.2 Percona Xtrabackup

http://jenkins.percona.com/view/XtraBackup/

We currently build Xtrabackup 1.6, 2.0 and xtrabackup trunk (will become 2.1).

There are param-builds for 1.6 and trunk too. These should be run for each merge request (and before any collection of merged branches is pushed to trunk)

76.3.3 Percona Server

We have separate jobs for Percona Server 5.1 and Percona Server 5.5 due to the different build systems that MySQL 5.1 and 5.5 use.

The `mysql-test-run.pl` test suite is integrated with Jenkins through subunit and `subunit2junitxml` allowing us to easily see which tests passed/failed on any particular test run.

Percona Server 5.1

http://jenkins.percona.com/view/PS%205.1/

We have trunk and param jobs. We also have a valgrind job that will run after a successful trunk build.

Percona Server 5.5

http://jenkins.percona.com/view/PS%205.5/

Similar to 5.1, but for PS5.5 instead.

76.3.4 MySQL Builds

http://jenkins.percona.com/view/MySQL/

I've set up a few jobs in Jenkins that should help us predict the future for Percona Server. Namely, if upstream MySQL may cause us any problems.

I wanted to see if some test failures were possibly upstream, so I set up two jobs:

http://jenkins.percona.com/view/MySQL/job/mysql-5.1-url-param/ http://jenkins.percona.com/view/MySQL/job/mysql-5.5-url-param/

both of which ask for a URL to a MySQL source tarball and then do a full build and test across the platforms we have in jenkins.

But my next thought was that we could try and do this *before* the source tarballs come out - hopefully then being able to have MySQL release source tarballs that do in fact pass build and test everywhere where we're wanting to support Percona Server.

http://jenkins.percona.com/view/MySQL/job/mysql-5.1-trunk/ http://jenkins.percona.com/view/MySQL/job/mysql-5.5-trunk/

are scheduled to just try once per week (we can change the frequency if we want to) to build and test from the MySQL bzr trees.

I also have a valgrind build (same configuration as for Percona Server) to help us see if there's any new valgrind warnings (or missed suppressions).

I'm hoping that these jobs will help us catch any future problems before they become our problem. (e.g. we can easily see that the sporadic test failures we see in Percona Server are actually in upstream MySQL).

CHAPTER

SEVENTYSEVEN

TRADEMARK POLICY

This Trademark Policy is to ensure that users of Percona-branded products or services know that what they receive has really been developed, approved, tested and maintained by Percona. Trademarks help to prevent confusion in the marketplace, by distinguishing one company's or person's products and services from another's.

Percona owns a number of marks, including but not limited to Percona, XtraDB, Percona XtraDB, XtraBackup, Percona XtraBackup, Percona Server, and Percona Live, plus the distinctive visual icons and logos associated with these marks. Both the unregistered and registered marks of Percona are protected.

Use of any Percona trademark in the name, URL, or other identifying characteristic of any product, service, website, or other use is not permitted without Percona's written permission with the following three limited exceptions.

First, you may use the appropriate Percona mark when making a nominative fair use reference to a bona fide Percona product.

Second, when Percona has released a product under a version of the GNU General Public License ("GPL"), you may use the appropriate Percona mark when distributing a verbatim copy of that product in accordance with the terms and conditions of the GPL.

Third, you may use the appropriate Percona mark to refer to a distribution of GPL-released Percona software that has been modified with minor changes for the sole purpose of allowing the software to operate on an operating system or hardware platform for which Percona has not yet released the software, provided that those third party changes do not affect the behavior, functionality, features, design or performance of the software. Users who acquire this Percona-branded software receive substantially exact implementations of the Percona software.

Percona reserves the right to revoke this authorization at any time in its sole discretion. For example, if Percona believes that your modification is beyond the scope of the limited license granted in this Policy or that your use of the Percona mark is detrimental to Percona, Percona will revoke this authorization. Upon revocation, you must immediately cease using the applicable Percona mark. If you do not immediately cease using the Percona mark upon revocation, Percona may take action to protect its rights and interests in the Percona mark. Percona does not grant any license to use any Percona mark for any other modified versions of Percona software; such use will require our prior written permission.

Neither trademark law nor any of the exceptions set forth in this Trademark Policy permit you to truncate, modify or otherwise use any Percona mark as part of your own brand. For example, if XYZ creates a modified version of the Percona Server, XYZ may not brand that modification as "XYZ Percona Server" or "Percona XYZ Server", even if that modification otherwise complies with the third exception noted above.

In all cases, you must comply with applicable law, the underlying license, and this Trademark Policy, as amended from time to time. For instance, any mention of Percona trademarks should include the full trademarked name, with proper spelling and capitalization, along with attribution of ownership to Percona Inc. For example, the full proper name for XtraBackup is Percona XtraBackup. However, it is acceptable to omit the word "Percona" for brevity on the second and subsequent uses, where such omission does not cause confusion.

In the event of doubt as to any of the conditions or exceptions outlined in this Trademark Policy, please contact trademarks@percona.com for assistance and we will do our very best to be helpful.

CHAPTER

SEVENTYEIGHT

INDEX OF INFORMATION_SCHEMA TABLES

This is a list of the INFORMATION_SCHEMA TABLES that exist in *Percona Server* with *XtraDB*. The entry for each table points to the page in the documentation where it's described.

- *CLIENT_STATISTICS*
- *INDEX_STATISTICS*
- *GLOBAL_TEMPORARY_TABLES*
- *QUERY_RESPONSE_TIME*
- *TABLE_STATISTICS*
- *TEMPORARY_TABLES*
- *THREAD_STATISTICS*
- *USER_STATISTICS*
- *INNODB_RSEG*
- *INNODB_UNDO_LOGS*
- *INNODB_SYS_TABLESTATS*
- *INNODB_INDEX_STATS*
- *INNODB_CHANGED_PAGES*
- *INNODB_BUFFER_POOL_PAGES*
- *INNODB_BUFFER_POOL_PAGES_BLOB*
- *INNODB_BUFFER_POOL_PAGES_INDEX*
- *INNODB_SYS_TABLES*
- *INNODB_SYS_FIELDS*
- *INNODB_SYS_COLUMNS*
- *INNODB_SYS_STATS*
- *INNODB_SYS_FOREIGN*
- *INNODB_SYS_INDEXES*
- *XTRADB_ADMIN_COMMAND*
- *INNODB_TABLE_STATS*
- *INNODB_SYS_FOREIGN_COLS*

CHAPTER

SEVENTYNINE

FREQUENTLY ASKED QUESTIONS

79.1 Q: Will *Percona Server* with *XtraDB* invalidate our *MySQL* support?

A: We don't know the details of your support contract. You should check with your *Oracle* representative. We have heard anecdotal stories from *MySQL* Support team members that they have customers who use *Percona Server* with *XtraDB*, but you should not base your decision on that.

79.2 Q: Will we have to *GPL* our whole application if we use *Percona Server* with *XtraDB*?

A: This is a common misconception about the *GPL*. We suggest reading the *Free Software Foundation* 's excellent reference material on the GPL Version 2, which is the license that applies to *MySQL* and therefore to *Percona Server* with *XtraDB*. That document contains links to many other documents which should answer your questions. *Percona* is unable to give legal advice about the *GPL*.

79.3 Q: Do I need to install *Percona* client libraries?

A: No, you don't need to change anything on the clients. *Percona Server* is 100% compatible with all existing client libraries and connectors.

79.4 Q: When using the *Percona XtraBackup* to setup a replication slave on Debian based systems I'm getting: "ERROR 1045 (28000): Access denied for user 'debian-sys-maint'@'localhost' (using password: YES)"

A: In case you're using init script on Debian based system to start `mysqld`, be sure that the password for `debian-sys-maint` user has been updated and it's the same as that user's password from the server that the backup has been taken from. Password can be seen and updated in `/etc/mysql/debian.cnf`. For more information on how to set up a replication slave using *Percona XtraBackup* see this how-to.

CHAPTER

EIGHTY

COPYRIGHT AND LICENSING INFORMATION

80.1 Documentation Licensing

80.2 Software License

Percona Server is built upon MySQL from Oracle. Along with making our own modifications, we merge in changes from other sources such as community contributions and changes from MariaDB.

The original SHOW USER/TABLE/INDEX statistics code came from Google.

Percona does not require copyright assignment.

See the COPYING files accompanying the software distribution.

CHAPTER

EIGHTYONE

OPTIONS THAT MAKE XTRADB TABLESPACES NOT COMPATIBLE WITH MYSQL

81.1 Fast checksums

Enabling *innodb_fast_checksum* will use more CPU-efficient algorithm, based on 4-byte words which can be beneficial for some workloads. Once enabled, turning it off will require table to be dump/imported again, since *Percona Server* will fail to start on data files created when innodb_fast_checksums was enabled.

In case you've migrated from *Percona Server* to *MySQL* you could get the "corrupted checksum" error message. In order to recover that table you'll need to:

1. Reinstall Percona Server to read your tables that were created with fast checksums.
2. Dump the tables (or temporarily convert them to MyISAM).
3. Install stock MySQL (or at least disable fast checksums).
4. Restore the InnoDB tables (or convert back from MyISAM).

Note: This feature has been deprecated after *Percona Server* *5.5.28-29.2* and it is not available in *Percona Server* 5.6, because the innodb_checksum_algorithm feature in *MySQL* 5.6 makes it redundant.

81.2 Page sizes other than 16KiB

This is controlled by variable *innodb_page_size*. Changing the page size for an existing database is not supported. Table will need to be dumped/imported again if compatibility with *MySQL* is required.

Note: This feature has been deprecated in the *Percona Server* *5.5.30-30.2*. It has been replaced by the upstream version released in *MySQL* 5.6.4.

81.3 Relocation of the doublewrite buffer

Variable *innodb_doublewrite_file* provides an option to put the buffer on a dedicated disk in order to parallelize I/O activity on the buffer and on the tablespace. Only in case of crash recovery this variable cannot be changed, in all other cases it can be turned on/off without breaking the compatibility.

Note: This feature has not been ported to *Percona Server* 5.6.

CHAPTER

EIGHTYTWO

PERCONA SERVER 5.5 RELEASE NOTES

82.1 *Percona Server* 5.5.62-38.14

Percona announces the release of Percona Server for MySQL 5.5.62-38.14 on November 21, 2018 (downloads are available here and from the Percona Software Repositories). This release merges changes of MySQL 5.5.62, including all the bug fixes in it. Percona Server for MySQL 5.5.62-38.14 is now the current GA release in the 5.5 series. All of Percona's software is open-source and free.

Note that Percona Server for MySQL 5.5.62-38.14 is the last release of the 5.5 series. This series goes EOL on December 1st, 2018.

82.1.1 Improvements

- #4790: The accuracy of user statistics has been improved

82.1.2 Bugs Fixed

- The binary log could be corrupted when the disk partition used for temporary files (tmpdir system variable) had little free space. Bug fixed #1107
- `PURGE CHANGED_PAGE_BITMAPS` did not work when the `innodb_data_home_dir` system variable was used. Bug fixed #4723

Other Bugs Fixed

- #4773: Percona Server sources can't be compiled without server.
- #4781: `sql_yacc.yy` uses `SQLCOM_SELECT` instead of `SQLCOM_SHOW_XXXX_STATS`

Find the release notes for Percona Server for MySQL 5.5.62-38.14 in our online documentation. Report bugs in the Jira bug tracker.

82.2 *Percona Server* 5.5.61-38.13

Percona announces the release of Percona Server for MySQL 5.5.61-38.13 on August 17, 2018 (downloads are available here and from the Percona Software Repositories). This release merges changes of MySQL 5.5.61, including all the bug fixes in it. Percona Server for MySQL 5.5.61-38.13 is now the current GA release in the 5.5 series. All of Percona's software is open-source and free.

82.2.1 Bugs Fixed

- The `--innodb-optimize-keys` option of the `mysqldump` utility fails when a column name is used as a prefix of a column which has the `AUTO_INCREMENT` attribute. Bug fixed #4524.

Other Bugs Fixed:

- #4566: "`stack-use-after-scope` in `reinit_io_cache()`" (upstream #91603)
- #4581: "`stack-use-after-scope` in `_db_enter_()` / `mysql_select_db()`" (upstream #91604)
- #4600: "`stack-use-after-scope` in `_db_enter_()` / `get_upgrade_info_file_name()`" (upstream #91617)
- #3976: "Errors in MTR tests `main.variables-big`, `main.information_schema-big`, `innodb.innodb_bug14676111`"

Find the release notes for Percona Server for MySQL 5.5.61-38.13 in our online documentation. Report bugs in the Jira bug tracker.

82.3 *Percona Server* 5.5.60-38.12

Percona is glad to announce the release of *Percona Server* 5.5.60-38.12 on May 18th, 2018. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.60, including all the bug fixes in it, *Percona Server* 5.5.60-38.12 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free.

82.3.1 Bugs Fixed

- `mysqldump` utility with `--innodb-optimize-keys` option was incorrectly working with foreign keys pointing on the same table, producing invalid SQL statements. Bugs fixed #1125 and #3863.
- A typo in `plugin.cmake` file prevented to compile plugins statically into the server. Bug fixed #3871 (upstream #89766).
- Using `-DWITHOUT_<PLUGIN>=ON` CMake variable to exclude a plugin from the build didn't work for some plugins, including a number of storage engines. Bug fixed #3901.
- A fix was introduced to remove GCC 8 compilation warnings for the *Percona Server* build. Bug fixed #3950.
- A code clean-up was done to fix compilation warnings and errors specific for clang 6. Bug fixed #3893 (upstream #90111).
- *Percona Server* Debian packages description included reference to `/etc/mysql/my.cnf` file, which is not actually present in these packages. Bug fixed #2046.
- A clean-up in *Percona Server* binlog-related code was made to avoid uninitialized memory comparison. Bug fixed #3925 (upstream #90238).
- Some `IO_CACHE` file operations were not instrumented for Performance Schema causing inaccurate statistics generated. Bug fixed #3937 (upstream #90264).
- A `key_block_size` value was set automatically by the Improved MEMORY Storage Engine, which resulted in warnings when changing the engine type to *InnoDB*, and constantly growing `key_block_size` during alter operations. Bugs fixed #3936, #3940, and #3943.

Other bugs fixed: #3767 "Fix compilation warnings/errors with clang", #3778 "5.5 Tree received Percona-TokuBackup submodule where it should not", #3794 "MTR test main.percona_show_temp_tables_stress does not wait for events to start", #3798 "MTR test innodb.percona_extended_innodb_status fails if InnoDB status contains unquoted special characters", and #3926 "Potentially truncated bitmap file name in log_online_open_bitmap_file_read_only() (storage/innobase/log/log0online.cc)".

82.4 *Percona Server* 5.5.59-38.11

Percona is glad to announce the release of *Percona Server* 5.5.59-38.11 on January 30th, 2018. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.59, including all the bug fixes in it, *Percona Server* 5.5.59-38.11 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free.

82.4.1 Bugs Fixed

With `innodb_large_prefix` set to `1`, Blackhole storage engine was incompatible with InnoDB table definitions, thus adding new indexes would cause replication errors on the slave. Fixed #1126 (upstream #53588).

A GCC 7 warning fix introduced introduced regression in *Percona Server* `5.5.58-38.10` that lead to a wrong SQL query built to access the remote server when Federated storage engine was used. Bug fixed #1134.

Percona Server 5.5 embedded server builds were broken. Bug fixed #2893.

Percona Server now uses *TraviCI* for additional tests. Bug fixed #3777.

Other bugs fixed: #257 and #2415.

82.5 *Percona Server* 5.5.58-38.10

Percona is glad to announce the release of *Percona Server* 5.5.58-38.10 on December 7th, 2017. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.58, including all the bug fixes in it, *Percona Server* 5.5.58-38.10 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.58-38.10 milestone at Launchpad.

82.5.1 New Features

Percona Server packages are now available for *Ubuntu 17.10 (Artful)*.

82.5.2 Bugs Fixed

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*).

`MEMORY` storage engine incorrectly allowed `BLOB` columns before indexed columns. Bug fixed #1731483.

Other bugs fixed: #1729241.

82.6 Percona Server 5.5.57-38.9

Percona is glad to announce the release of Percona Server 5.5.57-38.9 on August 23, 2017. Downloads are available here and from the *Percona Software Repositories*.

This release is based on MySQL 5.5.57 and includes all the bug fixes in it. Percona Server 5.5.57-38.9 is now the current stable release in the 5.5 series. All software developed by Percona is open-source and free. Details of this release can be found in the 5.5.57-38.9 milestone on Launchpad.

Note: Red Hat Enterprise Linux 5 (including CentOS 5 and other derivatives), Ubuntu 12.04 and older versions are no longer supported by Percona software. The reason for this is that these platforms reached end of life, will not receive updates and are not recommended for use in production.

82.6.1 New Features

- #1702903: Added support of OpenSSL 1.1.

82.6.2 Platform Support

- Added support and packages for Debian 9 (stretch). Covers only the `amd64` architecture.
- Removed packages for RHEL 5 (CentOS 5) and Ubuntu 12.04.

82.6.3 Bugs Fixed

- #1622985: Downgraded diagnostic severity from warning to normal note for successful doublewrite buffer recovery.
- #1661488: Fixed crash of debug server build when two clients connected, one of them enabled `userstat` and ran either `FLUSH CLIENT_STATISTICS` or `FLUSH USER_STATISTICS`, and then both clients exited.
- #1673656: Added support of wildcards and Subject Alternative Names (SAN) in SSL certificates for `--ssl-verify-server-cert`. For more information, see *Compatibility Matrix*.
- #1705729: Fixed the `postinst` script to correctly locate the `datadir`.
- #1709834: Fixed the `mysqld_safe` script to correctly locate the `basedir`.
- Minor fixes: #1160986, #1684601, #1689998, #1690012.

82.6.4 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

82.7 *Percona Server* 5.5.55-38.8

Percona is glad to announce the release of *Percona Server* 5.5.55-38.8 on May 10th, 2017. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.55, including all the bug fixes in it, *Percona Server* 5.5.55-38.8 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.55-38.8 milestone at Launchpad.

82.7.1 New Features

Percona Server 5.5 packages are now available for Ubuntu 17.04 (*Zesty Zapus*).

82.7.2 Bugs Fixed

If a bitmap write I/O errors happened in the background log tracking thread while a `FLUSH CHANGED_PAGE_BITMAPS` is executing concurrently it could cause a server crash. Bug fixed #1651656.

Querying *`TABLE_STATISTICS`* in combination with a stored function could lead to a server crash. Bug fixed #1659992.

Queries from the *`INNODB_CHANGED_PAGES`* table would needlessly read potentially incomplete bitmap data past the needed LSN range. Bug fixed #1625466.

It was not possible to configure basedir as a symlink. Bug fixed #1639735.

Other bugs fixed: #1688161, #1683456, #1670588 (upstream #84173), #1672389, #1675623, #1660243, #1677156, #1680061, #1680510 (upstream #85838), #1683993, #1684012, #1684025, and #1674281.

82.8 *Percona Server* 5.5.54-38.7

Percona is glad to announce the release of *Percona Server* 5.5.54-38.7 on March 22nd, 2017. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.54, including all the bug fixes in it, *Percona Server* 5.5.54-38.7 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.54-38.7 milestone at Launchpad.

82.8.1 Bugs Fixed

Log tracking initialization did not find last valid bitmap data correctly, potentially resulting in needless redo log retracking or hole in the tracked LSN range. Bug fixed #1658055.

Other bugs fixed: #1652912, and #1655587.

82.9 *Percona Server* 5.5.54-38.6

Percona is glad to announce the release of *Percona Server* 5.5.54-38.6 on February 1st, 2017. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.54, including all the bug fixes in it, *Percona Server* 5.5.54-38.6 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.54-38.6 milestone at Launchpad.

82.9.1 Bugs Fixed

Fixed new compilation warnings with GCC 6. Bugs fixed #1641612 and #1644183.

`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.

`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.

Other bugs fixed: #1638897, #1644174, #1644547, and #1644558.

82.10 *Percona Server* 5.5.53-38.5

Percona is glad to announce the release of *Percona Server* 5.5.53-38.5 on November 23rd, 2016. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.53, including all the bug fixes in it, *Percona Server* 5.5.53-38.5 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.53-38.5 milestone at Launchpad.

82.10.1 New Features

Metrics for scalability measurement feature is being built by default but deprecated. Users who have installed this plugin but are not using its capability are advised to uninstall the plugin due to known crashing bugs.

82.11 *Percona Server* 5.5.53-38.4

Percona is glad to announce the release of *Percona Server* 5.5.53-38.4 on November 18th, 2016. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.53, including all the bug fixes in it, *Percona Server* 5.5.53-38.4 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.53-38.4 milestone at Launchpad.

82.11.1 Removed Features

Metrics for scalability measurement feature has been removed. **WARNING:** if you have *`scalability_metrics_control`* variable in your `my.cnf` configuration file you'll need to remove it, otherwise server won't be able to start.

82.11.2 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.

Thread Pool `thread limit reached` and `failed to create thread` messages are now printed on the first occurrence as well. Bug fixed #1636500.

Other bugs fixed: #1612076, #1633061, #1633430, and #1635184.

82.12 *Percona Server* 5.5.52-38.3

Percona is glad to announce the release of *Percona Server* 5.5.52-38.3 on October 4th, 2016. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.52, including all the bug fixes in it, *Percona Server* 5.5.52-38.3 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.52-38.3 milestone at Launchpad.

82.12.1 Bugs Fixed

`mysql_upgrade` now does not binlog its actions by default. To restore the previous behavior, use `--write-binlog` option. Bug fixed #1065841 (upstream #56155).

Audit Log Plugin would hang when trying to write a log record of *audit_log_buffer_size* length. Bug fixed #1588439.

After fixing bug #1540338, system table engine validation check is no longer skipped for tables that don't have an explicit `ENGINE` clause in a `CREATE TABLE` statement. If *MySQL* upgrade statements are replicated, and slave does not have the *MyISAM* set as a default storage engine, then the `CREATE TABLE mysql.server` statement would attempt to create an *InnoDB* table and fail because `mysql_system_tables.sql` script omitted explicit engine setting for this table. Bug fixed #1600056.

Audit Log Plugin malformed record could be written after *audit_log_flush* was set to `ON` in `ASYNC` and `PERFORMANCE` modes. Bug fixed #1613650.

`INFORMATION_SCHEMA.TABLES` (or other schema info table) table query running in parallel with `INFORMATION_SCHEMA.GLOBAL_TEMPORARY_TABLES` query may result in TABLES-query thread context having a mutex locked twice, or unlocked twice, or left locked, resulting in crashes or hangs. Bug fixed #1614849.

Other bugs fixed: #1626002 (upstream #83073), #904714, #1098718, #1610102, #1610110, #1613663, #1613728, #1613986, #1614885, #1615959, #1616091, #1616753, #1616768, #1616937, #1617150, #1617323, #1618478, #1618718, #1618811, #1618819, #1619547, #1619572, #1619665, #1620200, #1626458, #1626500, and #1628417.

82.13 *Percona Server* 5.5.51-38.2

Percona is glad to announce the release of *Percona Server* 5.5.51-38.2 on September 21st, 2016. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.51, including all the bug fixes in it, *Percona Server* 5.5.51-38.2 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.51-38.2 milestone at Launchpad.

82.13.1 Bugs Fixed

Limiting `ld_preload` libraries to be loaded from specific directories in `mysqld_safe` didn't work correctly for relative paths. Bug fixed #1624247.

Fixed possible privilege escalation that could be used when running `REPAIR TABLE` on a `MyISAM` table. Bug fixed #1624397.

The general query log and slow query log cannot be written to files ending in `.ini` and `.cnf` anymore. Bug fixed #1624400.

Implemented restrictions on symlinked files (`error_log`, `pid_file`) that can't be used with `mysqld_safe`. Bug fixed #1624449.

Other bugs fixed: #1553938.

82.14 *Percona Server* 5.5.51-38.1

Percona is glad to announce the release of *Percona Server* 5.5.51-38.1 on August 19th, 2016. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.51, including all the bug fixes in it, *Percona Server* 5.5.51-38.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.51-38.1 milestone at Launchpad.

82.14.1 Bugs Fixed

PAM Authentication Plugin would abort authentication while checking UNIX user group membership if there were more than a thousand members. Bug fixed #1608902.

PAM Authentication Plugin didn't support spaces in the UNIX user group names. Bug fixed #1544443.

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).

Percona Server 5.5 could not be built with the `-DMYSQL_MAINTAINER_MODE=ON` option. Bug fixed #1590454.

In the client library, any EINTR received during network I/O was not handled correctly. Bug fixed #1591202 (upstream #82019).

The included `.gitignore` in the percona-server source distribution had a line `*.spec`, which means someone trying to check in a copy of the percona-server source would be missing the spec file required to build the RPMs. Bug fixed #1600051.

The fix for bug #1341067 added a call to free some of the heap memory allocated by OpenSSL. This is not safe for repeated calls if OpenSSL is linked twice through different libraries and each is trying to free the same. Bug fixed #1604676.

If the changed page bitmap redo log tracking thread stops due to any reason, then shutdown will wait for a long time for the log tracker thread to quit, which it never does. Bug fixed #1606821.

Performing slow *InnoDB* shutdown (`innodb_fast_shutdown` set to `0`) could result in incomplete purge, if a separate purge thread is running (which is a default in *Percona Server*). Bug fixed #1609364.

Other bugs fixed: #1515591 (upstream #79249), #1612551, #1609523, #756387, #1097870, #1603073, #1606478, #1606572, #1606782, #1607224, #1607359, #1607606, #1607607, #1607671, #1608385, #1608424, #1608437, #1608515, #1608845, #1609422, #1610858, #1612084, #1612118, and #1613641.

82.15 *Percona Server* 5.5.50-38.0

Percona is glad to announce the release of *Percona Server* 5.5.50-38.0 on July 8th, 2016. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.50, including all the bug fixes in it, *Percona Server* 5.5.50-38.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.50-38.0 milestone at Launchpad.

82.15.1 New Features

Percona Server has implemented protocol support for *TLS 1.1 and TLS 1.2*. This implementation turns off TLS v1.0 support by default.

82.15.2 Bugs Fixed

Querying the *GLOBAL_TEMPORARY_TABLES* table would cause server crash if temporary table owning threads would execute new queries. Bug fixed #1581949.

The *innodb_log_block_size* feature attempted to diagnose the situation where the logs have been created with a log block value that differs from the current *innodb_log_block_size* setting. But this diagnostics came too late, and a misleading error `No valid checkpoints found` was produced first, aborting the startup. Bug fixed #1155156.

`AddressSanitizer` build with `LeakSanitizer` enabled was failing at `gen_lex_hash` invocation. Bug fixed #1580993 (upstream #80014).

`ssl.cmake` file was broken when custom OpenSSL build was used. Bug fixed #1582639 (upstream #61619).

`mysqlbinlog` did not free the existing connection before opening a new remote one. Bug fixed #1587840 (upstream #81675).

Fixed memory leaks in `mysqltest`. Bugs fixed #1582718 and #1588318.

Fixed memory leaks in `mysqlcheck`. Bug fixed #1582741.

Fixed memory leak in `mysqlbinlog`. Bug fixed #1582761 (upstream #78223).

Fixed memory leaks in `mysqldump`. Bug fixed #1587873 and #1588845 (upstream #81714).

Fixed memory leak in `innochecksum`. Bug fixed #1587873.

Fixed memory leak in non-existing defaults file handling. Bug fixed #1588344.

Fixed memory leak in `mysqlslap`. Bug fixed #1587873.

Other bugs fixed: #1588169, #1588386, #1529885, #1587757, #1587426 (upstream, #81657), #1587527, #1588650, and #1589819.

82.16 *Percona Server* 5.5.49-37.9

Percona is glad to announce the release of *Percona Server* 5.5.49-37.9 on May 19th, 2016. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.49, including all the bug fixes in it, *Percona Server* 5.5.49-37.9 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.49-37.9 milestone at Launchpad.

82.16.1 Bugs Fixed

Percona Server is now built with system `zlib` library instead of the older bundled one. Bug fixed #1108016.

`CREATE TABLE ... LIKE ...` could create a system table with an unsupported enforced engine. Bug fixed #1540338.

Server will now show more descriptive error message when *Percona Server* fails with `errno == 22 "Invalid argument"`, if *`innodb_flush_method`* was set to `ALL_O_DIRECT`. Bug fixed #1578604.

`apt-cache show` command for `percona-server-client` was showing `innotop` included as part of the package. Bug fixed #1201074.

`mysql-systemd` would fail with PAM authentication and proxies due to regression introduced when fixing #1534825 in *Percona Server* *`5.5.48-37.8`*. Bug fixed #1558312.

Other bugs fixed: #1578625 (upstream #81295), bug fixed #1553166, and bug fixed #1578303 (upstream #81324).

82.17 *Percona Server* 5.5.48-37.8

Percona is glad to announce the release of *Percona Server* 5.5.48-37.8 on March 4th, 2016. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.48, including all the bug fixes in it, *Percona Server* 5.5.48-37.8 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.48-37.8 milestone at Launchpad.

82.17.1 Bugs Fixed

With *Expanded Fast Index Creation* enabled, DDL queries involving *InnoDB* temporary tables would cause later queries on the same tables to produce warnings that their indexes were not found in the index translation table. Bug fixed #1233431.

Package upgrade on *Ubuntu* would run `mysql_install_db` even though data directory already existed. Bug fixed #1457614.

Starting *MySQL* with `systemctl` would fail with timeout if the socket was specified with a custom path. Bug fixed #1534825.

`mysqldumpslow` script has been removed because it was not compatible with *Percona Server* extended slow query log format. Please use pt-query-digest from *Percona Toolkit* instead. Bug fixed #856910.

When `cmake/make/make_binary_distribution` workflow was used to produce binary tarballs it would produce tarballs with `mysql-...` naming instead of `percona-server-...`. Bug fixed #1540385.

Other bugs fixed: #1521120 and #1534246.

82.18 *Percona Server* 5.5.47-37.7

Percona is glad to announce the release of *Percona Server* 5.5.47-37.7 on January 12th, 2015. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.47, including all the bug fixes in it, *Percona Server* 5.5.47-37.7 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.47-37.7 milestone at Launchpad.

82.18.1 Bugs Fixed

Running `OPTIMIZE TABLE` or `ALTER TABLE` without the `ENGINE` clause would silently change table engine if *`enforce_storage_engine`* variable was active. This could also result in system tables being changed to incompatible storage engines, breaking server operation. Bug fixed #1488055.

Other bugs fixed: #1179451, #1524763, and #1530102.

82.19 *Percona Server* 5.5.46-37.6

Percona is glad to announce the release of *Percona Server* 5.5.46-37.6 on December 4th, 2015. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.46, including all the bug fixes in it, *Percona Server* 5.5.46-37.6 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.46-37.6 milestone at Launchpad.

82.19.1 Bugs Fixed

An upstream fix for upstream bug #76135 might cause server to stall or hang. Bug fixed #1519094 (upstream #79185).

Fixed invalid memory accesses when **`mysqldump`** was running with `--innodb-optimize-keys` option. Bug fixed #1517444.

Other bugs fixed: #1517523.

82.20 *Percona Server* 5.5.46-37.5

Percona is glad to announce the release of *Percona Server* 5.5.46-37.5 on November 5th, 2015. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.46, including all the bug fixes in it, *Percona Server* 5.5.46-37.5 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.46-37.5 milestone at Launchpad.

82.20.1 New Features

Percona Server is now available for *Ubuntu* 15.10 (Wily).

82.20.2 Bugs Fixed

Manipulating the `innodb_track_redo_log_now` variable dynamically would crash the server if it was started without *`innodb_track_changed_pages`* enabled. This variable is available on debug builds only. Bug fixed #1368530.

A potential crash in handling corrupted tables with *`innodb_corrupt_table_action`* `warn` or `salvage` values has been fixed. Bug fixed #1426610.

If the user had duplicate pid-file options in config files when running `yum upgrade`, the upgrade would stop with error because it would think it found the duplicate pid while it was the same pid specified twice. Bug fixed #1454917.

On some filesystems server would not start if *XtraDB changed page tracking* feature was enabled and *`innodb_flush_method`* variable was set to `O_DIRECT`. Bugs fixed #1500720 and #1498891.

When *User Statistics* are enabled, executing any statement of the `SHOW` family with non-empty result, would bump *`USER_STATISTICS`* `ROWS_FETCHED` column values erroneously. Bug fixed #1510953.

Fixed the conflicting meta packages between 5.1, 5.5, and 5.6 release series in *Debian* and *Ubuntu* distributions. `percona-server-server` and `percona-server-client` meta packages now point to the latest 5.6 release. Bug fixed #1292517.

`INNODB_CHANGED_PAGES` table was unavailable with non-default `innodb_data_home_dir` setting if the variable had a trailing slash. Bug fixed #1364315.

`UPDATE` statement could crash the server with *Support for Fake Changes* enabled. Bug fixed #1395706.

Changing *`innodb_fake_changes`* variable value in the middle of a transaction would have an immediate effect, that is, making part of the transaction run with fake changes enabled and the rest with fake changes disabled, resulting in a broken transaction. Fixed by making any *`innodb_fake_changes`* value changes becoming effective at the start of the next transaction instead of the next statement. Bug fixed #1395579.

Startup would fail due to a small hard-coded timeout value in the init script for the pid file to appear. This has been fixed by creating default file for *Debian* init script timeout parameters in `etc/default/mysql`. Bug fixed #1434022.

Percona Server would fail to install on *CentOS* 7 if `mariadb-devel` package was already installed. Bug fixed #1499721.

The upstream bug #76627 was not fixed for the `ALL_O_DIRECT` case. Bug fixed #1500741.

Other bugs fixed: #1512301, #1160960, and #1497942.

82.21 *Percona Server* 5.5.45-37.4

Percona is glad to announce the release of *Percona Server* 5.5.45-37.4 on September 2nd, 2015. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.45, including all the bug fixes in it, *Percona Server* 5.5.45-37.4 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.45-37.4 milestone at Launchpad.

82.21.1 Bugs Fixed

Querying `INFORMATION_SCHEMA` *`GLOBAL_TEMPORARY_TABLES`* table would crash threads working with internal temporary tables used by `ALTER TABLE`. Bug fixed #1113388.

`FLUSH INDEX_STATISTICS` / `FLUSH CHANGED_PAGE_BITMAPS` and `FLUSH USER_STATISTICS` / `RESET CHANGE_PAGE_BITMAPS` pairs of commands were inadvertently joined, i.e. issuing either command had the effect of both. The first pair, besides flushing both index statistics and changed page bitmaps, had the effect of `FLUSH INDEX_STATISTICS` requiring `SUPER` instead of `RELOAD` privilege. The second pair resulted in `FLUSH USER_STATISTICS` destroying changed page bitmaps. Bug fixed #1472251.

If a new connection thread was created while a `SHOW PROCESSLIST` command or a `INFORMATION_SCHEMA.PROCESSLIST` query was in progress, it could have a negative `TIME_MS` value returned in the `PROCESSLIST` output. Bug fixed #1379582.

Other bugs fixed: #768038 and #1472256.

82.22 *Percona Server* 5.5.44-37.3

Percona is glad to announce the release of *Percona Server* 5.5.44-37.3 on June 29th, 2015. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.44, including all the bug fixes in it, *Percona Server* 5.5.44-37.3 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.44-37.3 milestone at Launchpad.

82.22.1 Bugs Fixed

Symlinks to `libmysqlclient` libraries were missing on *CentOS* 6. Bug fixed #1408500.

RHEL/CentOS 6.6 `openssl` package (1.0.1e-30.el6_6.9), containing a fix for CVE-2015-4000, changed the DH key sizes to a minimum of 768 bits. This caused an issue for *MySQL* as it uses 512 bit keys. Fixed by backporting an upstream 5.7 fix that increases the key size to 2048 bits. Bug fixed #1462856 (upstream #77275).

innochecksum would fail to check tablespaces in compressed format. The fix for this bug has been ported from *Facebook MySQL* 5.1 patch. Bug fixed #1100652 (upstream #66779).

Issuing `SHOW BINLOG EVENTS` with an invalid starting binlog position would cause a potentially misleading message in the server error log. Bug fixed #1409652 (upstream #75480).

While using *`max_slowlog_size`*, the slow query log was rotated every time `slow_query_log` was enabled, not really checking if the current slow log is indeed bigger than *`max_slowlog_size`* or not. Bug fixed #1416582.

If *`query_response_time_range_base`* was set as a command line option or in a configuration file, its value would not take effect until the first flush was made. Bug fixed #1453277 (*Preston Bennes*).

Prepared XA transactions with update undo logs were not properly recovered. Bug fixed #1468301.

Variable *log_slow_sp_statements* now supports skipping the logging of stored procedures into the slow log entirely with new `OFF_NO_CALLS` option. Bug fixed #1432846.

Other bugs fixed: #1380895 (upstream #72322).

82.23 *Percona Server* 5.5.43-37.2

Percona is glad to announce the release of *Percona Server* 5.5.43-37.2 on May 8th, 2015. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.43, including all the bug fixes in it, *Percona Server* 5.5.43-37.2 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.43-37.2 milestone at Launchpad.

82.23.1 Bugs Fixed

A server binary as distributed in binary tarballs could fail to load on different systems due to an unsatisfied `libssl.so.6` dynamic library dependency. This was fixed by replacing the single binary tarball with multiple tarballs depending on the *OpenSSL* library available in the distribution: 1) ssl100 - for all *Debian/Ubuntu* versions except *Squeeze/Lucid* (`libssl.so.1.0.0 => /usr/lib/x86_64-linux-gnu/libssl.so.1.0.0 (0x00007f2e389a5000)`); 2) ssl098 - only for *Debian Squeeze* and *Ubuntu Lucid* (`libssl.so.0.9.8 => /usr/lib/libssl.so.0.9.8 (0x00007f9b30db6000)`); 3) ssl101 - for *CentOS 6* and *CentOS 7* (`libssl.so.10 => /usr/lib64/libssl.so.10 (0x00007facbe8c4000)`); 4) ssl098e - to be used only for *CentOS 5* (`libssl.so.6 => /lib64/libssl.so.6 (0x00002aed5b64d000)`). Bug fixed #1172916.

`mysql_install_db` would make the server produce an "`Error in my_thread_global_end(): 1 threads didn't exit`" error message. While this error does not prevent `mysql_install_db` from completing successfully, its presence might cause any `mysql_install_db`-calling script to return an error as well. This is a regression introduced by backporting fix for bug #1319904. Bug fixed #1402074.

A string literal containing an invalid UTF-8 sequence could be treated as falsely equal to a UTF-8 column value with no invalid sequences. This could cause invalid query results. Bug fixed #1247218 by a fix ported from *MariaDB* (MDEV-7649).

Percona Server `.deb` binaries were built without fast mutexes. Bug fixed #1433980.

Installing or uninstalling the *Audit Log Plugin* would crash the server if the *audit_log_file* variable was pointing to an inaccessible path. Bug fixed #1435606.

The *audit_log_file* would point to random memory area if the *Audit Log Plugin* was not loaded into server, and then installed with `INSTALL PLUGIN`, and `my.cnf` contained *audit_log_file* setting. Bug fixed #1437505.

Percona Server client `.deb` packages were built with with `EditLine` instead of `Readline`. Further, a client built with `EditLine` could display incorrectly on *PuTTY* SSH client after its window resize. Bugs fixed #1266386 and #1332822 (upstream #63130 and #69991).

Other bugs fixed: #1436138 (upstream #76505).

82.24 *Percona Server* 5.5.42-37.1

Percona is glad to announce the release of *Percona Server* 5.5.42-37.1 on March 4th, 2015. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.42, including all the bug fixes in it, *Percona Server* 5.5.42-37.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.42-37.1 milestone at Launchpad.

82.24.1 Bugs Fixed

`RPM` pre-install script assumed that the `PID` file was always located in the `datadir`. If it was not, during installation, wrong assumption could be made if the server was running or not. Bug fixed #1201896.

`SHOW GRANTS` displayed only the privileges granted explicitly to the named account. Other effectively available privileges were not displayed. Fixed by implementing *Extended SHOW GRANTS* feature. Bug fixed #1354988 (upstream #53645).

InnoDB lock monitor output was printed even if it was not requested. Bug fixed #1418996.

The stored procedure key was made consistent with other keys in the *Slow Query Log* by replacing space with an underscore. Bug fixed #1419230.

Other bugs fixed: #1408232, #1415843 (upstream #75642), bug fixed #1407941, and bug fixed #1424568 (upstream #75868).

82.25 *Percona Server* 5.5.41-37.0

Percona is glad to announce the release of *Percona Server* 5.5.41-37.0 on January 9th, 2015. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.41, including all the bug fixes in it, *Percona Server* 5.5.41-37.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.41-37.0 milestone at Launchpad.

82.25.1 New Features

Percona Server has implemented *CSV engine mode for standard-compliant quote and comma parsing*. This feature also fixes the bug #1316042 (upstream #71091).

Percona Server has implemented improved slow log reporting for queries in *stored procedures*.

82.25.2 Bugs Fixed

Debian and *Ubuntu* init scripts no longer have a hardcoded server startup timeout. This has been done to accommodate situations where server startup takes a very long time, for example, due to a crash recovery or buffer pool dump restore. Bugs fixed #1072538 and #1328262.

If HandlerSocket was enabled, the server would hang during shutdown. Bug fixed #1319904.

Wrong stack calculation could lead to a server crash when Performance Schema tables were storing big amount of data or in case of server being under highly concurrent load. Bug fixed #1351148 (upstream #73979).

Values of `IP` and `DB` fields in the *Audit Log Plugin* were incorrect. Bug fixed #1379023.

Percona Server 5.5 would fail to build with GCC 4.9.1 (such as bundled with *Ubuntu Utopic*) in debug configuration. Bug fixed #1396358 (upstream #75000).

Default *MySQL* configuration file, `my.cnf`, was not installed during the new installation on *CentOS*. Bug fixed #1405667.

A session on a server in mixed mode binlogging would switch to row-based binlogging whenever a temporary table was created and then queried. This switch would last until the session end or until all temporary tables in the session were dropped. This was unnecessarily restrictive and has been fixed so that only the statements involving temporary tables were logged in the row-based format whereas the rest of the statements would continue to use the statement-based logging. Bug fixed #1313901 (upstream #72475).

Purging bitmaps exactly up to the last tracked LSN would abort *XtraDB changed page tracking*. Bug fixed #1382336.

`mysql_install_db` script would silently ignore any mysqld startup failures. Bug fixed #1382782 (upstream #74440).

Other bugs fixed: #1067103, #1394357, #1282599, #1335590, #1335590, #1401791 (upstream #73281), and #1396330 (upstream #74987).

82.26 *Percona Server* 5.5.40-36.1

Percona is glad to announce the release of *Percona Server* 5.5.40-36.1 on October 7th, 2014. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.40, including all the bug fixes in it, *Percona Server* 5.5.40-36.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.40-36.1 milestone at Launchpad.

82.26.1 Bugs Fixed

Audit Log Plugin would rotate the audit log in middle of an audit message. Bug fixed #1363370.

Fixed a memory leak in *Metrics for scalability measurement*. Bug fixed #1334570.

Fixed a memory leak if *secure-file-priv* option was used with no argument. Bug fixed #1334719.

82.27 *Percona Server* 5.5.39-36.0

Percona is glad to announce the release of *Percona Server* 5.5.39-36.0 on August 29th, 2014. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.39, including all the bug fixes in it, *Percona Server* 5.5.39-36.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.39-36.0 milestone at Launchpad.

82.27.1 New Features

Percona Server *Audit Log Plugin* now supports `JSON` and `CSV` formats. The format choice is controlled by `audit_log_format` variable.

Percona Server *Audit Log Plugin* now supports *streaming the audit log to syslog*.

82.27.2 Bugs Fixed

Querying *INNODB_CHANGED_PAGES* with a range condition `START_LSN > x AND END_LSN < y` would lead to a server crash if the range was empty with x greater than y. Bug fixed #1202252 (*Jan Lindström* and *Sergei Petrunia*).

With *XtraDB changed page tracking* feature enabled, queries from the *INNODB_CHANGED_PAGES* could read the bitmap data whose write was in still progress. This would cause the query to fail with an `ER_CANT_FIND_SYSTEM_REC` and a warning printed to the server error log. The workaround is to add an appropriate `END_LSN`-limiting condition to the query. Bug fixed #1346122.

`mysqld-debug` was missing from *Debian* packages. This regression was introduced in *Percona Server 5.5.36-34.0*. Bug fixed #1290087.

Fixed a memory leak in *Slow Query Log Rotation and Expiration*. Bug fixed #1314138.

The audit log plugin would write log with XML syntax errors when `OLD` and `NEW` formats were used. Bug fixed #1320879.

A server built with system OpenSSL support, such as the distributed Percona Server binaries, had SSL-related memory leaks. Bug fixed #1334743 (upstream #73126).

If the bitmap directory has a bitmap file sequence with a start LSN of one file less than a start LSN of the previous file, a debug build would assert when queries were run on *INNODB_CHANGED_PAGES* table. Bug fixed #1342494.

Server would crash on login attempt if `mysql.user` table was truncated. Bug fixed #1322218.

Other bugs fixed: #1337324, #1151723, #1182050, #1182072, #1280875, #1182046, #1328482 (upstream #73418), and #1334317 (upstream #73111).

82.28 *Percona Server* 5.5.38-35.2

Percona is glad to announce the release of *Percona Server* 5.5.38-35.2 on July 2nd, 2014. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.38, including all the bug fixes in it, *Percona Server* 5.5.38-35.2 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.38-35.2 milestone at Launchpad.

82.28.1 Bugs Fixed

Fixed the duplicate string in *InnoDB* version number. Bug fixed #1328421.

An uninitialized memory use in *User Statistics* has been fixed. Bug fixed #1167486.

82.29 *Percona Server* 5.5.37-35.1

Percona is glad to announce the release of *Percona Server* 5.5.37-35.1 on June 3rd, 2014. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.37, including all the bug fixes in it, *Percona Server* 5.5.37-35.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.37-35.1 milestone at Launchpad.

82.29.1 Bugs Fixed

InnoDB could crash if workload contained writes to compressed tables. Bug fixed #1305364.

GUI clients such as *MySQL Workbench* could not authenticate with a user defined with `auth_pam_compat` plugin. Bug fixed #1166938.

Help in *Percona Server* 5.5 command line client was linking to *Percona Server* 5.1 manual. Bug fixed #1198775.

Audit Log Plugin wasn't parsing escape characters correctly in the `OLD` format. Bug fixed #1313696.

Percona Server version was reported incorrectly in *Debian*/*Ubuntu* packages. Bug fixed #1319670.

Other bugs fixed: #1272732, #1219833, #1271178, and #1314568.

82.30 *Percona Server* 5.5.37-35.0

Percona is glad to announce the release of *Percona Server* 5.5.37-35.0 on May 6th, 2014. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.37, including all the bug fixes in it, *Percona Server* 5.5.37-35.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.37-35.0 milestone at Launchpad.

82.30.1 New Features

Percona Server now supports *Metrics for scalability measurement*.

Percona Server now supports *Audit Log Plugin*.

Percona Server packages are now available for *Ubuntu* 14.04.

82.30.2 Bugs Fixed

Percona Server couldn't be built with *Bison* 3.0. Bug fixed #1262439,upstream #71250 (*Ryan Gordon*).

Backported the upstream fix for overflow which would caused replication SQL thread to fail to execute events. Bug fixed #1070255 (upstream #67352).

Percona Server debug packages were not built for the previous releases. Bug fixed #1298352.

Queries that no longer exceed `long_query_time` were written to the slow query log if they matched the previous `long_query_time` value when *`slow_query_log_use_global_control`* variable was set to `all`. Bug fixed #1016991.

When writing audit plugins it was not possible to get notifications for general-log events without enabling the general-log. Bug fixed #1182535 (upstream #60782).

`mysqld_safe` did not correctly parse *`flush_caches`* and *`numa_interleave`* options. Bug fixed #1231110.

Thread Pool would handle a new client connection without notifying Audit Plugin. Bug fixed #1282008.

Fixed a performance issue in extending tablespaces if running under `fusionIO` with *atomic writes* enabled. Bug fixed #1286114 (*Jan Lindström*).

Previous implementation of the *`log_slow_rate_type`* set to `query` with *`log_slow_rate_limit`* feature would log every nth query deterministically instead of each

query having a 1/n probability to get logged. Fixed by randomly selecting the queries to be logged instead of logging every nth query. Bug fixed #1287650.

Percona Server source files were referencing *Maatkit* instead of *Percona Toolkit*. Bug fixed #1174779.

Maximum allowed value for *log_slow_rate_limit* was ULONG_MAX (ie. either 4294967295 or 18446744073709551615, depending on the platform). As it was unreasonable to configure the slow log for every four billionth session/query, new maximum allowed value is set to 1000. Bug fixed #1290714.

Other bugs fixed: #1272732.

82.31 *Percona Server* 5.5.36-34.2

Percona is glad to announce the release of *Percona Server* 5.5.36-34.2 on March 25th, 2014. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.36, including all the bug fixes in it, *Percona Server* 5.5.36-34.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.36-34.2 milestone at Launchpad.

82.31.1 Bugs Fixed

The upgrade to *Percona Server 5.5.36-34.1* would silently comment out any options in my.cnf that have paths specified that contain share/mysql. Bug fixed #1293867.

Percona Server could fail to start after upgrade if the lc-messages-dir option was set in the my.cnf configuration file. Bug fixed #1294067.

Dependency on mysql-common package, introduced in *Percona Server 5.5.36-34.0* could lead to wrongly chosen packages for upgrade, spurious removes and installs with some combination of packages installed which use the mysql libraries. Bug fixed #1294211.

These three bugs were fixed by removing the dependency on mysql-common package.

Percona Toolkit UDFs and *HandlerSocket* were missing from *Debian*/*Ubuntu* packages, this regression was introduced in *Percona Server 5.5.36-34.0*. Bugs fixed #1296416 and #1294216.

Percona Server installer will create the symlinks from libmysqlclient to libperconaserverclient during the installation on *CentOS*. This was implemented in order to provide the backwards compatibility after the libmysqlclient library has been renamed to libperconaserverclient .

82.32 *Percona Server* 5.5.36-34.1

Percona is glad to announce the release of *Percona Server* 5.5.36-34.1 on March 17th, 2013. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.36, including all the bug fixes in it, *Percona Server* 5.5.36-34.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.36-34.1 milestone at Launchpad.

82.32.1 Bugs Fixed

After installing the `auth_socket` plugin any local user might get root access to the server. If you're using this plugin upgrade is advised. This is a regression, introduced in *Percona Server* `5.5.31-30.3`. Bug fixed #1289599.

The new client and server packages included files with paths that were conflicting with the ones in `mysql-libs` package on *CentOS*. Bug fixed #1278516.

A clean installation of `Percona-Server-server-55` on *CentOS* would fail due to a typo in `mysql_install_db` call. Bug fixed #1291247.

`Percona-Server-shared-55` package was still declared as providing `mysql-libs`, but it is not actually providing it anymore. Bug fixed #1291249.

Slave I/O thread wouldn't attempt to automatically reconnect to the master after a network time-out (`error: 1159`). Bug fixed #1268729 (upstream #71374).

Slave I/O thread wouldn't attempt to automatically reconnect to the master if setting `master_heartbeat_period` failed with a transient network error. Bug fixed #1268735 (upstream #71375).

82.32.2 Renaming the `libmysqlclient` to `libperconaserverclient`

This release fixes some of the issues caused by the `libmysqlclient` rename to `libperconaserverclient` in *Percona Server* `5.5.36-34.0`. The old name was conflicting with the upstream `libmysqlclient`.

Except for packaging, `libmysqlclient` and `libperconaserverclient` of the same version do not have any differences. Users who previously compiled software against Percona-provided `libmysqlclient` will either need to install the corresponding package of their distribution, such as `mysql-lib` for *CentOS* and `libmysqlclient18` for *Ubuntu*/*Debian* or recompile against `libperconaserverclient`. Another workaround option is to create a symlink from `libperconaserverclient.so.18.0.0` to `libmysqlclient.so.18.0.0`.

82.33 *Percona Server* 5.5.36-34.0

Percona is glad to announce the release of *Percona Server* 5.5.36-34.0 on March 10th, 2014. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.36, including all the bug fixes in it, *Percona Server* 5.5.36-34.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.36-34.0 milestone at Launchpad.

82.33.1 New Features

Debian and *Ubuntu* packaging has been reworked to meet the packaging standards.

Ability to change database for mysqlbinlog has been ported from *Percona Server* 5.1.

Percona Server has implemented *Slow Query Log Rotation and Expiration* feature to provide users with better control of slow query log disk space usage.

In order to comply with Linux distribution packaging standards *Percona*'s version of `libmysqlclient` has been renamed to `libperconaserver`. The old name was conflicting with the upstream `libmysqlclient`. Except for packaging, `libmysqlclient` and `libperconaserverclient`

of the same version do not have any differences. Users wishing to continue using `libmysqlclient` will have to install the corresponding package of their distribution, such as `mysql-lib` for *CentOS* and `libmysqlclient18` for *Ubuntu*/*Debian*. Users wishing to build software against `libperconaserverclient` should install `libperconaserverclient-dev` package. An old version of Percona-built `libmysqlclient` will be available for download.

82.33.2 Bugs Fixed

The *XtraDB* version number in `univ.i` was incorrect. Bug fixed #1277383.

Percona Toolkit UDFs were only shipped with RPM packages. Bug fixed #1159625.

Server could crash if it was signaled with `SIGHUP` early in the server startup. Bug fixed #1249193 (upstream #62311).

Server could crash if *XtraDB* `innodb_dict_size` option was set due to incorrect attempts to remove indexes in use from the dictionary cache. Bugs fixed #1250018 and #758788.

Fix for bug #1227581, a buffer pool mutex split regression, was not complete, thus a combination of write workload and tablespace drop could crash the server if *InnoDB* compression was used. Bug fixed #1269352.

Binary RPM packages couldn't be built from source tarballs on *Fedora* 19. Bug fixed #1229598.

Percona Server that was compiled from source package had different server version string from that of binary packages. Bug fixed #1244178.

InnoDB did not handle the cases of asynchronous and synchronous I/O requests completing partially or being interrupted. Bugs fixed #1262500 (upstream #54430), and #1263087 (*Andrew Gaul*).

Fixed the `CMake` warnings that were happening when `Makefile` was generated. Bugs fixed #1274827 (upstream #71089).

Percona Server source tree has been reorganized to match the *MySQL* source tree layout closer. Bug fixed #1014477.

On *Ubuntu* Precise multiple architecture versions of `libmysqlclient18` couldn't be installed side by side. Bug fixed #1052636.

Database administrator password could be seen in plain text if when `debconf-get-selections` was executed. Bug fixed #1018291.

Other bugs fixed: #1005787.

82.34 *Percona Server* 5.5.35-33.0

Percona is glad to announce the release of *Percona Server* 5.5.35-33.0 on December 20th, 2013. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.35, including all the bug fixes in it, *Percona Server* 5.5.35-33.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.35-33.0 milestone at Launchpad.

82.34.1 New Features

Thread Pool high priority scheduling is now enabled by default by changing the default *`thread_pool_high_prio_tickets`* value from `0` to `4294967295`.

Percona Server now supports *Low priority queue throttling*. This feature should improve *Thread Pool* performance under high concurrency in a situation when thread groups are oversubscribed.

Introduced new `thread_pool_high_prio_mode` to provide more fine-grained control over high priority scheduling either globally or per connection in *Thread Pool*.

Percona Server has *extended* **`mysqlbinlog`** to provide `SSL` and compression support.

Percona Server has reduced the performance overhead of the *User Statistics* feature.

82.34.2 Bugs Fixed

`INSTALL PLUGIN` statement would crash server if *User Statistics* were enabled. Bug fixed #1011047.

Fixed the assertion error caused by a race condition between one thread performing a tablespace delete and another doing a compressed page flush list relocation. Bug fixed #1227581.

Server would crash on shutdown if *Atomic write support for Fusion-io devices* feature is enabled. Bug fixed #1255628 (*Jan Lindström*).

Fixed the compiler errors, caused by merge regression in *Percona Server* `5.5.33-31.1`. Bug fixed #1218417.

`mysqldump --innodb-optimize-keys` was generating incorrect `CREATE TABLE` statements for partitioned tables. Bug fixed #1233841.

Default value for `thread_pool_max_threads` has been changed from `500` to `100 000` (the maximum supported number of connections), because limiting the total number of threads in the threadpool can result in deadlocks and uneven distribution of worker threads between thread groups in case of stalled connections. Bug fixed #1258097.

`PURGE CHANGED_PAGE_BITMAPS BEFORE` statement would delete the changed page data after the specified LSN and up to the start of the next bitmap file. If this data were to be used for fast incremental backups, its absence would cause *Percona XtraBackup* to fall back to the full-scan incremental backup mode. Bug fixed #1260035 (*Andrew Gaul*).

Debug server build would crash during *InnoDB* crash recovery if the crash recovery had found transactions that needed cleaning up. Bug fixed #1247305.

Variable `thread_pool_high_prio_tickets` is now a session variable. Bug fixed #1166271.

Other bugs fixed: bug #1082333.

82.35 *Percona Server* 5.5.34-32.0

Percona is glad to announce the release of *Percona Server* 5.5.34-32.0 on October 28th, 2013. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.34, including all the bug fixes in it, *Percona Server* 5.5.34-32.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.34-32.0 milestone at Launchpad.

82.35.1 New Features

Percona Server has *extended* the `SELECT INTO ... OUTFILE` and `SELECT INTO DUMPFILE` to add the support for UNIX sockets and named pipes.

Percona Server now provides additional information in the slow query log when *log_slow_rate_limit* variable is enabled.

A new variable *slow_query_log_always_write_time* has been introduced. It can be used to specify an additional execution time threshold for the slow query log, that, when exceeded, will cause a query to be logged unconditionally, that is, *log_slow_rate_limit* will not apply to it.

Utility user feature has been extended by adding a new *utility_user_privileges* that allows a comma separated value list of extra access privileges that can be granted to the utility user.

82.35.2 Bugs Fixed

Due to an incompatible upstream change that went in unnoticed, the log tracker thread would attempt to replay any file operations it encountered. In most cases this were a no-op, but there were race conditions for certain `DDL` operations that would have resulted in server crash. Bug fixed #1217002.

`apt-get upgrade` of *Percona Server* would fail in post-installation step if server failed to start. Bug fixed #1002500.

Fixed the `libssl.so.6` dependency issues in binary tarballs releases. Bug fixed #1172916.

Error in `install_layout.cmake` could cause that some library files, during the build, end up in different directories on x86_64 environment. Bug fixed #1174300.

Percona Server could crash while accessing `BLOB` or `TEXT` columns in *InnoDB* tables if *Support for Fake Changes* was enabled. Bug fixed #1188168.

Memory leak was introduced by the fix for bug #1132194. Bug fixed #1204873.

The unnecessary overhead from persistent *InnoDB* adaptive hash index latching has been removed, potentially improving stability of the *Multiple Adaptive Hash Search Partitions* feature as well. Upstream bug fixed #70216, bug fixed #1218347.

Fixed the incorrect dependency with `libmysqlclient18-dev` from *Percona Server* *5.5.33-31.1*. Bug fixed #1237097.

A memory leak in *Utility user* feature has been fixed. Bug fixed #1166638.

Expanded Program Option Modifiers did not deallocate memory correctly. Bug fixed #1167487.

A server could crash due to a race condition between a *INNODB_CHANGED_PAGES* query and a bitmap file delete by `PURGE CHANGED_PAGE_BITMAP` or directly on the file system. Bug fixed #1191580.

Percona Server could not be built with *Thread Pool* feature and `-DWITH_PERFSCHEMA_ENGINE=OFF` option. Bug fixed #1196383.

Building *Percona Server* with `-DHAVE_PURIFY` option would result in an error. Fixed by porting the `close_socket` function from *MariaDB*. Bug fixed #1203567.

Adaptive hash index memory size was incorrectly calculated in `SHOW ENGINE INNODB STATUS` and *Innodb_mem_adaptive_hash* status variable. Bug fixed #1218330.

Some *Expanded Program Option Modifiers* didn't have an effect if they were specified in non-normalized way (`innodb_io_capacity` vs `innodb-io-capacity`). Bug fixed #1233294.

Enabling *Enforcing Storage Engine* feature could lead to error on *Percona Server* shutdown. Bug fixed #1233354.

Storage engine enforcement (*enforce_storage_engine*) is now ignored when the server is started in either bootstrap or skip-grant-tables mode. Bug fixed #1236938.

Fixed the build warnings caused by *User Statistics* code on non-Linux platforms. Bug fixed #711817.

Adaptive hash indexing partitioning code has been simplified, potentially improving performance. Bug fixed #1218321.

Other bugs fixed: bug fixed #1239630, bug fixed #1191589, bug fixed #1200162, bug fixed #1214449, and bug fixed #1190604.

82.36 *Percona Server* 5.5.33-31.1

Percona is glad to announce the release of *Percona Server* 5.5.33-31.1 on August 27th, 2013. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.33, including all the bug fixes in it, *Percona Server* 5.5.33-31.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.33-31.1 milestone at Launchpad.

82.36.1 Bugs Fixed

The buffer pool mutex split patch implemented in *Percona Server* could cause a race condition, involving a dirty compressed page block for which there is an uncompressed page image in the buffer pool, that could lead to a server crash. Bug fixed #1086680.

If binary log was enabled, *Fake Changes* transactions were binlogged. This could lead to data corruption issues with deeper replication topologies. Bug fixed #1190580.

Changes made to the `RPM` scripts for previous *Percona Server* version caused installer to fail if there were different datadir options in multiple configuration files. Bug fixed #1201036.

Percona Server `shared-compat` package was being built with the 5.1.66 version of the client, which didn't work with `OpenSSL`. Fixed by building the `shared-compat` package with a more recent version. Bug fixed #1201393.

Fixed the upstream bug #69639 which caused compile errors for *Percona Server* with `DTrace version Sun D 1.11` provided by recent `SmartOS` versions. Bug fixed #1196460.

Fixed a regression introduced in *Percona Server* `5.5.32-31.0`, where server wouldn't be able to start if *Atomic write support for Fusion-io devices* was enabled. Bug fixed #1214735.

Percona Server used to acquire the buffer pool `LRU` list mutex in the I/O completion routine for the compressed page flush list flushes where it was not necessary. Bug fixed #1181269.

Other bugs fixed: bug fixed #1189743, bug fixed #1188162 and bug fixed #1203308.

82.37 *Percona Server* 5.5.32-31.0

Percona is glad to announce the release of *Percona Server* 5.5.32-31.0 on July 2nd, 2013. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.32, including all the bug fixes in it, *Percona Server* 5.5.32-31.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.32-31.0 milestone at Launchpad.

82.37.1 New Features

Percona Server has implemented support for supplementary groups for *PAM Authentication Plugin*.

82.37.2 Bugs Fixed

Prevented a race condition that could lead to a server crash when querying the `INFORMATION_SCHEMA.INNODB_BUFFER_PAGE` table. Bug fixed #1072573.

Percona Server wouldn't start if the *XtraDB changed page tracking* was enabled and variable *innodb_flush_method* was set to `ALL_O_DIRECT`. Bug fixed #1131949.

Fixed the upstream bug #68970 that, in *Percona Server*, would cause small tablespaces to expand too fast around 500KB tablespace size. Bug fixed #1169494.

Query to the *INNODB_CHANGED_PAGES* table would cause server to stop with an I/O error if a bitmap file in the middle of requested LSN range was missing. Bug fixed #1179974.

Server would crash if an *INNODB_CHANGED_PAGES* query is issued that has an empty LSN range and thus does not need to read any bitmap files. Bug fixed #1184427.

Querying `INFORMATION_SCHEMA.PARTITIONS` could cause key distribution statistics for partitioned tables to be reset to those corresponding to the last partition. Fixed the upstream bug #69179. Bug fixed #1192354.

Incorrect schema definition for the *User Statistics* tables in `INFORMATION_SCHEMA` (*CLIENT_STATISTICS*, *INDEX_STATISTICS*, *TABLE_STATISTICS*, *THREAD_STATISTICS*, and *USER_STATISTICS*) led to the maximum counter values being limited to 32-bit signed integers. Fixed so that these values can be 64-bit unsigned integers now. Bug fixed #714925.

Fixed the upstream bug #42415 that would cause `UPDATE/DELETE` statements with the `LIMIT` clause to be unsafe for Statement Based Replication even when `ORDER BY` primary key was present. Fixed by implementing an algorithm to do more elaborate analysis on the nature of the query to determine whether the query will cause uncertainty for replication or not. Bug fixed #1132194.

When an upgrade was performed between major versions (e.g. by uninstalling a 5.1 RPM and then installing a 5.5 one), `mysql_install_db` was still called on the existing data directory which lead to re-creation of the `test` database. Bug fixed #1169522.

XtraDB changed page tracking used to hold the log system mutex for the log reads needlessly, potentially limiting performance on write-intensive workloads. Bug fixed #1171699.

The RPM installer script had the datadir hardcoded to `/var/lib/mysql` instead of using `my_print_defaults` function to get the correct datadir info. Bug fixed #1181753.

Missing path separator between the directory and file name components in a bitmap file name could stop the server starting if the `innodb_data_home_dir` variable didn't have the path separator at the end. Bug fixed #1181887.

Fixed the upstream bug #68354 that could cause server to crash when performing update or join on `Federated` and `MyISAM` tables with one row, due to a bug in the `Federated` storage engine. Bug fixed #1182572.

A warning is now returned if a bitmap file I/O error occurs after an *INNODB_CHANGED_PAGES* query started returning data to indicate an incomplete result set. Bug fixed #1185040.

Under very rare circumstances, deleting a zero-size bitmap file at the right moment would make server stop with an I/O error if changed page tracking is enabled. Bug fixed #1184517.

Fixed the compiler warnings caused by *Atomic write support for Fusion-io devices* when building *Percona Server* on non-Linux platforms. Bug fixed #1189429.

The *INNODB_CHANGED_PAGES* table couldn't be queried if the log tracker wasn't running. Bug fixed #1185304.

Transaction objects are now allocated calling `calloc()` directly instead of using *InnoDB* heap allocation. This may improve write performance for high levels of concurrency. Bug fixed #1185686.

Other bugs fixed: bug fixed #1099764, bug fixed #1132412, bug fixed #1191395, bug fixed #1079688, bug fixed #1132422, bug fixed #1153651, bug fixed #1160951, bug fixed #1183583, bug fixed #1133266.

82.38 *Percona Server* 5.5.31-30.3

Percona is glad to announce the release of *Percona Server* 5.5.31-30.3 on May 24th, 2013. Downloads are available here and from the *Percona Software Repositories*.

Based on MySQL 5.5.31, including all the bug fixes in it, *Percona Server* 5.5.31-30.3 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.31-30.3 milestone at Launchpad.

82.38.1 New Features

Percona Server has ported the *Atomic write support for Fusion-io devices* patch from *MariaDB*. This feature adds atomic write support for `directFS` filesystem on `Fusion-io` devices. This feature implementation is considered BETA quality.

Percona Server has introduced `innodb_read_views_memory` and `innodb_descriptors_memory` status variables in the *Extended Show Engine InnoDB Status* to improve *InnoDB* memory usage diagnostics.

82.38.2 Bug Fixes

Fix for bug #1131187 introduced a regression that could cause a memory leak if query cache was used together with *InnoDB*. Bug fixed #1170103.

Fixed `RPM` packaging regression that was introduced with the fix for bug #710799. This regression caused mysql schema to be missing after the clean `RPM` installation. Bug fixed #1174426.

Fixed the `Percona-Server-shared-55` and `Percona-XtraDB-Cluster-shared` `RPM` package dependences. Bug fixed #1050654.

Fixed the upstream bug #68999 which caused compiling *Percona Server* to fail on *CentOS* 5 and *Debian* squeeze due to older `OpenSSL` version. Bug fixed #1183610.

If a slave was running with its binary log enabled and then restarted with the binary log disabled, *Crash-Resistant Replication* could overwrite the relay log info log with an incorrect position. Bug fixed #1092593.

Fixed the CVE-2012-5615 vulnerability. This vulnerability would allow remote attacker to detect what user accounts exist on the server. This bug fix comes originally from *MariaDB* (see MDEV-3909). Bug fixed #1171941.

Fixed the CVE-2012-5627 vulnerability, where an unprivileged *MySQL* account owner could perform brute-force password guessing attack on other accounts efficiently. This bug fix comes originally from *MariaDB* (see MDEV-3915). Bug fixed #1172090.

`mysql_set_permission` was failing on *Debian* due to missing `libdbd-mysql-perl` package. Fixed by adding the package dependency. Bug fixed #1003776.

Rebuilding *Debian* source package would fail because `dpatch` and `automake` were missing from build-dep. Bug fixed #1023575 (*Stephan Adig*).

Backported the fix for the upstream bug #65077 from the *MySQL* 5.6 version, which removed MyISAM internal temporary table mutex contention. Bug fixed #1179978.

82.39 *Percona Server* 5.5.30-30.2

Percona is glad to announce the release of *Percona Server* 5.5.30-30.2 on April 10th, 2013 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.30, including all the bug fixes in it, *Percona Server* 5.5.30-30.2 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.30-30.2 milestone at Launchpad.

82.39.1 New Features

Percona Server has implemented priority connection scheduling for the *Thread Pool*. (*Alexey Kopytov*)

Percona Server `.tar.gz` binaries will now be shipped with the `libjemalloc` library. `jemalloc` package is now available in our *apt*/*yum* repositories as well. In case newer version is available in the distribution repositories that version will be used. Benchmark showing the impact of memory allocators on *MySQL* performance can be found in this blogpost. (*Ignacio Nin*)

This release of *Percona Server* has fixed a number of performance bugs. (*Alexey Kopytov*)

Drop table performance has been removed and its controlling variable *`innodb_lazy_drop_table`* has been deprecated. Feature has been removed because the upstream `DROP TABLE` implementation has been improved. (*Laurynas Biveinis*)

82.39.2 Known Issues

This release contains a regression introduced by the fix for bug #1131187. The workaround is to disable the query cache. Regression is reported as bug #1170103.

82.39.3 Bug Fixes

Due to parse error in the `percona-server.spec` *Percona Server* rpm packages couldn't be built on *RHEL* 5 and *CentOS* 5. Bug fixed #1144777 (*Ignacio Nin*).

When **mysqldump** was used with `--innodb-optimize-keys` option it produced invalid SQL for cases when there was an explicitly named foreign key constraint which implied an implicit secondary index with the same name. Fixed by detecting such cases and omitting the corresponding secondary keys from deferred key creation optimization. Bug fixed #1081016 (*Alexey Kopytov*).

Percona Server was built with YaSSL which could cause some of the programs that use it to crash. Fixed by building packages with OpenSSL support rather than the bundled YaSSL library. Bug fixed #1104977 (*Ignacio Nin*).

Running a DDL statement while *`innodb_lazy_drop_table`* was enabled could cause assertion failure. Bugs fixed #1086227 and #1128848 (*Laurynas Biveinis*).

Fixed yum dependencies that were causing conflicts in `CentOS` 6.3 during installation. Bugs fixed #1031427 and #1051874 (*Ignacio Nin*).

The log tracker thread was unaware of the situation when the oldest untracked log records are overwritten by the new log data. In some corner cases this could lead to assertion errors in the log parser or bad changed page data. Bug fixed #1108613 (*Laurynas Biveinis*).

Ported a fix from *MariaDB* for the upstream bug #63144. `CREATE TABLE` or `CREATE TABLE IF NOT EXISTS` statements on an existing table could wait on a metadata lock instead of failing or returning immediately if there is a transaction that executed a query which opened that table. Bug fixed #1127008 (*Sergei Glushchenko*).

Fix for bug #1070856 introduced a regression in *Percona Server* `5.5.28-29.3` which could cause a server to hang when binary log is enabled. Bug fixed #1162085 (*Alexey Kopytov*).

Fixed upstream bug #49169 by avoiding the `malloc` call in the `read_view_create_low()` in most cases. This significantly improves *InnoDB* scalability on read-only workloads, especially when the default glibc memory allocator is used. Bug fixed #1131187 (*Alexey Kopytov*).

Removed `trx_list` scan in `read_view_open_now()` which is another problem originally reported as upstream bug #49169. This also provides much better scalability in *InnoDB* high-concurrent workloads. Bugs fixed #1131189 (*Alexey Kopytov*).

In the event that a slave was disconnected from the master, under certain conditions, upon reconnect, it would report that it received a packet larger than the `slave_max_allowed_packet` variable. Bug fixed #1135097 (*George Ormond Lorch III*).

Fixed the upstream bug #62578 which caused *MySQL* client to abort the connections on terminal resize. Bug fixed #925343 (*Sergei Glushchenko*).

Percona Server would re-create the test database when using `rpm` on server upgrade, even if the database was previously removed. Bug fixed #710799 (*Alexey Bychko*).

Debian packages included the old version of **innotop**. Fixed by removing **innotop** and its `InnoDBParser` Perl package from source and Debian installation. Bug fixed #1032139 (*Alexey Bychko*).

UDF/configure.ac was incompatible with `automake` 1.12. Bug fixed #1099387 (*Alexey Bychko*).

Reduced the overhead from *innodb_pass_corrupt_table'* value checks by optimizing them for better CPU branch prediction. Bug fixed #1125248 (*Alexey Kopytov*).

`dialog.so` used by the *PAM Authentication Plugin* couldn't be loaded with Perl and Python clients when `plugin-dir` option was set in the `[client]` section of the `my.cnf`. Bug fixed #1155859 (*Sergei Glushchenko*).

Fixed the upstream bug #68845 which could unnecessarily increase contention on `log_sys->mutex` in write-intensive workloads. Bug fixed #1163439 (*Alexey Kopytov*).

Ported back from the upstream *MySQL* 5.6 the fix for unnecessary `log_flush_order_mutex` acquisition. Bug fixed #1163262 (*Alexey Kopytov*).

When **mysqldump** was used with `--innodb-optimize-keys` and `--no-data` options, all secondary key definitions would be lost. Bug fixed #989253 (*Alexey Kopytov*).

Warning about the *Percona Toolkit* UDFs was omitted when installing from Percona's *Debian* repositories. Bug fixed #1015506 (*Alexey Bychko*).

Percona Server was missing help texts in the *MySQL* client because the help tables were missing. Bug fixed #1041981 (*Alexey Bychko*).

Fixed the upstream bug #68197 that caused *InnoDB* to misclassify internal read operations as synchronous when they were actually asynchronous when *Thread Pool* feature was used. Bug fixed #1107539 (*Sergei Glushchenko*).

Suboptimal code for *User Statistics* feature has been optimized to make sure no additional work is done when `userstat` is disabled. Bug fixed #1128066 (*Alexey Kopytov*).

Other bug fixes: bug fixed #1103850 (*Laurynas Biveinis*), bug fixed #1146621 (*Laurynas Biveinis*), bug fixed #1050536 (*Alexey Bychko*), bug fixed #1144059 (*Roel Van de Paar*), bug fixed #1154962 (*Hrvoje Matijakovic*), bug fixed #1154959 (*Hrvoje Matijakovic*), bug fixed #1154957 (*Hrvoje Matijakovic*), bug fixed #1154954 (*Hrvoje Matijakovic*).

82.40 *Percona Server* 5.5.30-30.1

Percona is glad to announce the release of *Percona Server* 5.5.30-30.1 on March 7th, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.30, including all the bug fixes in it, *Percona Server* 5.5.30-30.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.30-30.1 milestone at Launchpad.

82.40.1 Bug Fixes

Minor optimization cleanup by removing unnecessary `current_thd` calls. Bug fixed #1121794 (*Alexey Kopytov*).

Fixed the regression introduced with the fix for bug #1083058 which caused unnecessary mutex re-acquisitions in adaptive flushing. Bug fixed #1117067 (*Laurynas Biveinis*).

Percona Server would do unnecessary slow log stats accounting even with the slow log disabled. Bug fixed #1123915 (*Alexey Kopytov*).

Optimization cleanup to avoid calls related to extended slow query log stats when this feature is disabled. Bug fixed #1123921 (*Alexey Kopytov*).

The static `srv_pass_corrupt_table` variable could share CPU cache lines with *InnoDB* row counters, which resulted in high false sharing effect in high-concurrent workloads. Bug fixed #1125259 (*Alexey Kopytov*).

Fixed the regression introduced with fix for bug #791030 in *Percona Server* `5.5.13-20.4` by implementing an optimized version of the same function. Bug fixed #1130655 (*Alexey Kopytov*).

Potentially improve server performance by implementing an optimized version of the `my_strnxfrm_simple` function. Bug fixed #1132350, upstream bug fixed #68476 (*Alexey Kopytov*).

Potentially improve server performance by implementing an optimized version of the `skip_trailing_space` function. Bug fixed #1132351, upstream bug fixed #68477 (*Alexey Kopytov*).

Other bug fixes: bug fixed #1089265 (*Laurynas Biveinis*).

82.41 *Percona Server* 5.5.29-30.0

Percona is glad to announce the release of *Percona Server* 5.5.29-30.0 on February 26th, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.29, including all the bug fixes in it, *Percona Server* 5.5.29-30.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.29-30.0 milestone at Launchpad.

82.41.1 New Features

Ported the *Thread Pool* patch from *MariaDB*. This feature enables the server to keep the top performance even with the increased number of client connections.

New user statements have been introduced for handling the *XtraDB changed page tracking* log files.

In addition to the `--debug` build option for `build-binary.sh` script, new `--valgrind` option has been introduced, which will build debug builds with the Valgrind instrumentation enabled.

82.41.2 Bug Fixes

Ported a fix from MariaDB for the upstream bug #67974, which caused server crash on concurrent `ALTER TABLE` and `SHOW ENGINE INNODB STATUS`. Bug fixed #1017192 (*Sergei Glushchenko*).

The server could crash when executing an `INSERT` or `UPDATE` statement containing `BLOB` values for a compressed table. This regression was introduced in *Percona Server* `5.5.28-29.2`. Bug fixed #1100159 (*Laurynas Biveinis*).

Upstream bug #67983 was causing a memory leak on a filtered slave. Bug fixed #1042946 (*Sergei Glushchenko*).

Percona Server would fail to install on a vanilla *Ubuntu* 12.04 server. Bug fixed #1103655 (*Ignacio Nin*).

The master thread was doing dirty buffer pool flush list reads to make its adaptive flushing decisions. Fixed by acquiring the flush list mutex around the list scans. Bug fixed #1083058 (*Laurynas Biveinis*).

Upstream changes made to improve *InnoDB* `DROP TABLE` performance were not adjusted for *XtraDB*. This could cause server assertion errors. Bugs fixed #934377, bug #1111211, bug #1116447 and #1110102 (*Laurynas Biveinis*).

The *XtraDB* used to print the open read view list without taking the kernel mutex. Thus any list element might become invalid during its iteration. Fixed by taking the kernel mutex. Bug fixed #1101030 (*Laurynas Biveinis*).

When option `innodb_flush_method=O_DIRECT` was set up, log bitmap files were created and treated as *InnoDB* data files for flushing purposes, which wasn't original intention. Bug fixed #1105709 (*Laurynas Biveinis*).

`INFORMATION_SCHEMA` plugin name `innodb_changed_pages` serves also as a command line option, but it is also a prefix of another command line option `innodb_changed_pages_limit`. *MySQL* option handling would then shadow the former with the latter, resulting in start up errors. Fixed by renaming the `innodb_changed_pages_limit` option to *`innodb_max_changed_pages`*. Bug fixed #1105726 (*Laurynas Biveinis*).

Time in slow query log was displayed incorrectly when *`slow_query_log_timestamp_precision`* variable was set to microseconds. Bug fixed #887928 (*Laurynas Biveinis*).

Writing bitmap larger than 4GB would cause write to fail. Also a write error for every bitmap page, except the first one, would result in a heap corruption. Bug fixed #1111226 (*Laurynas Biveinis*).

Fixed the upstream bug #67504 that caused spurious duplicate key errors. Errors would happen if a trigger is fired while a slave was processing replication events for a table that is present only on slave server while there are updates on the replicated table on the master which is used in that trigger. For this to happen

master needs to have more than one auto-increment table and the slave needs to have at least one of those tables specified in the `replicate-ignore-table`. Bug fixed #1068210 (*George Ormond Lorch III*).

Fixed failing `rpm` builds, that were caused by missing files. Bug fixed #1099809 (*Alexey Bychko*).

Fixed the upstream #68116 that caused the server crash with assertion error when *InnoDB* monitor with verbose lock info was used under heavy load. This bug is affecting only `-debug` builds. Bug fixed #1100178 (*Laurynas Biveinis*).

XtraDB changed page tracking wasn't compatible with `innodb_force_recovery=6`. When starting the server log tracking initialization would fail. The server would abort on startup. Bug fixed #1083596 (*Laurynas Biveinis*).

Newly created bitmap file would silently overwrite the old one if they had the same file name. Bug fixed #1111144 (*Laurynas Biveinis*).

A server would stop with an assertion error in I/O and AIO routines if large *`innodb_log_block_size`* value is used in the combination with changed page tracking. Bug fixed #1114612 (*Laurynas Biveinis*).

`Optimizer_fix` patch has been removed from *Percona Server*. Bug fixed #986247 (*Stewart Smith*).

InnoDB monitor was prefetching the data pages for printing lock information even if no lock information was going to be printed. Bug fixed #1100643 (*Laurynas Biveinis*).

InnoDB and the query plan information were being logged even if they weren't enabled for the slow query log. Bug fixed #730173 (*Laurynas Biveinis*).

Fixed the incorrect help text for *`slow_query_log_timestamp_precision`*. Bug fixed #1090965 (*Laurynas Biveinis*).

Other bug fixes: bug fixed #909376 (*Laurynas Biveinis*), bug fixed #1082437 (*Laurynas Biveinis*), bug fixed #1083669 (*Laurynas Biveinis*), bug fixed #1096904 (*Laurynas Biveinis*), bug fixed #1091712 (*Laurynas Biveinis*), bug fixed #1096899 (*Laurynas Biveinis*), bug fixed #1088954 (*Laurynas Biveinis*), bug fixed #1096895 (*Laurynas Biveinis*), bug fixed #1092142 (*Laurynas Biveinis*), bug fixed #1090874 (*Laurynas Biveinis*), bug fixed #1089961 (*Laurynas Biveinis*), bug fixed #1088867 (*Laurynas Biveinis*), bug fixed #1089031 (*Laurynas Biveinis*), bug fixed #1108874 (*Laurynas Biveinis*), bug fixed #1030783 (*Laurynas Biveinis*), and bug fixed #803072 (*Laurynas Biveinis*).

82.42 *Percona Server* 5.5.29-29.4

Percona is glad to announce the release of *Percona Server* 5.5.29-29.4 on January 23rd, 2013 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.29, including all the bug fixes in it, *Percona Server* 5.5.29-29.4 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.29-29.4 milestone at Launchpad.

82.42.1 Bug Fixes

Fixed the upstream bug #68045 and ported a fix for the security vulnerability CVE-2012-4414 from the *Percona Server* *`5.5.28-29.3`*. This bug fix replaces the upstream fix for the *MySQL* bug #66550. More details about this can be found in Stewart's blogpost. Bug fixed #1049871 (*Vlad Lesin*).

82.43 *Percona Server* 5.5.28-29.3

Percona is glad to announce the release of *Percona Server* 5.5.28-29.3 on January 8th, 2013 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.28, including all the bug fixes in it, *Percona Server* 5.5.28-29.3 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.28-29.3 milestone at Launchpad.

82.43.1 Bug Fixes

Fixed the upstream bug #66550 and the security vulnerability CVE-2012-4414. This was caused because user-supplied identifiers (table names, field names, etc.) are not always properly quoted, so authorized users that have privileges to modify a table (any non-temporary table) can inject arbitrary SQL into the binary log and that could cause multiple SQL injection like vulnerabilities. This bug fix comes originally from MariaDB (see MDEV-382). Bug fixed #1049871 (*Vlad Lesin*).

Fixed the upstream bug #67685 and the security vulnerability CVE-2012-5611. This vulnerability allowed remote authenticated users to execute arbitrary code via a long argument to the `GRANT FILE` command. This bug fix comes originally from MariaDB (see MDEV-3884). Bug fixed #1083377 (*Vlad Lesin*).

`Rows_read` was calculated in a way which lead to a negative value being printed in the slow query log. Fixed by making `Rows_read` to be a synonym for `Rows_examined` in the slow query log. Bug fixed #830286 (*Alexey Kopytov*).

Fixed the upstream bug #66237. Temporary files created by binary log cache were not purged after transaction commit. Fixed by truncating the temporary file, if used for a binary log transaction cache, when committing or rolling back a statement or a transaction. Bug fixed #1070856 (*Alexey Kopytov*).

Values for `Rows_sent` and `Rows_read` would be identical in the *Slow Query Log*. This bug was introduced when slow_extended.patch was ported to *Percona Server* 5.5. Fixed by making `Rows_read` identical to `Rows_examined` instead. Bug fixed #721176 (*Alexey Kopytov*).

Fixed unsigned math error in `fsp_reserve_free_extents` that in some specific cases would cause the function to believe that billions more extents have been reserved than have actually been reserved. Bug fixed #1083700 (*George Ormond Lorch III*).

When **`mysqldump`** was used with `--innodb-optimize-keys`, it did not handle composite indexes correctly when verifying if the optimization is applicable with respect to `AUTO_INCREMENT` columns. Bug fixed #1039536 (*Alexey Kopytov*).

Upstream bug #67606 would cause *Percona Server* to crash with segmentation fault when disk quota was reached. Bug fixed #1079596 (*George Ormond Lorch III*).

In cases where indexes with `AUTO_INCREMENT` columns where correctly detected, **`mysqldump`** prevented all such keys from optimization, even though it is sufficient to skip just one (e.g. the first one). Bug fixed #1081003 (*Alexey Kopytov*).

Other bug fixes: bug fixed #1071986 (*Alexey Kopytov*), bug fixed #901060 (*Laurynas Biveinis*), bug fixed #1090596 (*Stewart Smith*), bug fixed #1087202 (*Vladislav Vaintroub, Laurynas Biveinis*) and bug fixed #1087218 (*Vladislav Vaintroub, Laurynas Biveinis*).

82.44 *Percona Server* 5.5.28-29.2

Percona is glad to announce the release of *Percona Server* 5.5.28-29.2 on December 7th, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.28, including all the bug fixes in it, *Percona Server* 5.5.28-29.2 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.28-29.2 milestone at Launchpad.

82.44.1 New Features

Multiple bitmap file support for *XtraDB changed page tracking* has been implemented.

Percona Server now has an option to build the binary tarball with enabled debugging. New flag `--dubug` has been added to the build script, that will create a build with the debug-enabled binaries. New binaries will have `-debug` appended in case it is a debug build, ie. `mysqld-debug`.

HandlerSocket has been updated to version 1.1.0 (rev. 83d8f3af176e1698acd9eb3ac5174700ace40fe0).

`innochecksum` has been extended with an option to read file format information from a given InnoDB data file. As only the first page needs to be read to detect the format/version information, it can also be used on a running server. This information can be useful when doing the *Expand Table Import*.

Support for Fake Changes has been improved by fetching the sibling pages.

Fast InnoDB Checksum feature has now been deprecated.

82.44.2 Bug Fixes

innodb_fake_changes didn't handle duplicate keys on `REPLACE`. In some cases this could cause infinite loop. Bug fixed #898306 (*Mark Callaghan*, *Laurynas Biveinis*).

Fixed the package dependencies for CentOS 6, that caused conflicts during the install. Bug fixed #908620 (*Ignacio Nin*).

innodb_fake_changes would allocate too many extents on `UPDATE`. In some cases this could cause infinite loop. Bug fixed #917942 (*Mark Callaghan*, *Laurynas Biveinis*).

Fixed the upstream bug #67737. `mysqldump` test would fail due to mixing STDOUT and STDERR. Bug fixed #959198 (*Stewart Smith*).

Although fake change transactions downgrade the requested exclusive (X) row locks to shared (S) locks, these S locks prevent X locks from being taken and block the real changes. This fix introduces a new option *innodb_locking_fake_changes* which, when set to `FALSE`, makes fake transactions not to take any row locks. Bug fixed #1064326 (*Mark Callaghan*, *Laurynas Biveinis*).

Fake changes were increasing the changed row and userstat counters. Bug fixed #1064333 (*Laurynas Biveinis*).

Log tracking was initialized too late during the *InnoDB* startup. Bug fixed #1076892 (*Laurynas Biveinis*).

Debuginfo *Debian* packages have been added for *Percona Server*. Bugs fixed #711062 and #1043873 (*Ignacio Nin*).

There is no need to scan buffer pool for AHI entries after the B-trees for the tablespace have been dropped, as that will already clean them. Bug fixed #1076215 (*Laurynas Biveinis*).

slow_extended code did not handle the case of individual statements in stored procedures correctly. This caused `Query_time` to increase for every query stored procedure logged to the slow query log. Bug fixed #719386 (*Alexey Kopytov*).

Other bug fixes: bug fixed #890404 (*Laurynas Biveinis*), bug fixed #1071877 (*Laurynas Biveinis*), bug fixed #1050466 (*Laurynas Biveinis*).

82.45 *Percona Server* 5.5.28-29.1

Percona is glad to announce the release of *Percona Server* 5.5.28-29.1 on October 26th, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.28, including all the bug fixes in it, *Percona Server* 5.5.28-29.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.28-29.1 milestone at Launchpad.

82.45.1 New Features

Percona Server has ported Twitter's *MySQL* `NUMA` patch. This patch implements *Improved NUMA support* as it prevents imbalanced memory allocation across NUMA nodes.

82.45.2 Bug Fixes

Percona Server would disconnect clients if `gdb` was attached and detached. This was caused by wrong signal handling. Bugs fixed: #805805 and #1060136 (upstream *MySQL* bug #67052) (*Laurynas Biveinis*).

Fixed the upstream *MySQL* bug #66890, where slave server would crash after update statement. Bug fixed #1053342 (*George Ormond Lorch III*).

Reads from tablespaces being deleted would result in buffer pool locking error. This regression was introduced by porting the recently introduced InnoDB code to XtraDB in *Percona Server* `5.5.27-28.0`. Bug fixed #1042640 (*Stewart Smith*).

Resolved the *Ubuntu Percona Server* package conflicts with upstream packages. Bug fixed #907499 (*Ignacio Nin*).

Crash-resistant replication would break with binlog XA transaction recovery. If a crash would happened between XA PREPARE and COMMIT stages, the prepared *InnoDB* transaction would not have the slave position recorded and thus would fail to update it once it is replayed during binlog crash recovery. Bug fixed #1012715 (*Laurynas Biveinis*).

82.46 *Percona Server* 5.5.27-29.0

Percona is glad to announce the release of *Percona Server* 5.5.27-29.0 on October 11th, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.27, including all the bug fixes in it, *Percona Server* 5.5.27-29.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.27-29.0 milestone at Launchpad.

82.46.1 New Features

- *Percona Server* now supports *XtraDB changed page tracking*. This feature will be used for implementing faster incremental backups that use this information to avoid full data scans.
- Number of binlog files can be restricted when using *Percona Server* with the new `max_binlog_files` option.

82.46.2 Bug Fixes

- Fixed server assertion error related to buffer pool, only visible in debug builds. Bug fixed #905334 (*Stewart Smith*).
- Fix for bug #978036 introduced the `innodb_sys_stats_root_page` debugging option (only present in debug builds), rendering the previously-existing innodb_sys_stats option its prefix. As such, it became unsettable from command line. Fixed by renaming `innodb_sys_stats_root_page` to `innodb_persistent_stats_root_page`. Bug fixed #1013644 (*Laurynas Biveinis*).
- Multiple adaptive hash index partitions would cause overly large hash index. Fixed by changing the way partition sizes are calculated initially. Bug fixed #1018264 (*George Ormond Lorch III*).
- Postfix would crash on CentOS/RHEL 6.x when using shared dependency (`libmysqlclient.so`). Fixed by building packages with OpenSSL support rather than the bundled YaSSL library. Bug fixed #1028240 (*Ignacio Nin*).
- Fixed the issue where LRU dump would hold LRU_list_mutex during the entire dump process. Now the mutex is periodically released in order not to block server while the dump is in progress. Bug fixed #686534 (*George Ormond Lorch III*).
- Option `expire_logs_days` was broken by group_commit patch introduced in *Percona Server* `5.5.18-23.0`. Bug fixed #1006214 (*Stewart Smith*).
- Fixed issue where `innodb_blocking_lru_restore` did not take an optional bool argument similar to other bool options. Bug fixed #881001 (*George Ormond Lorch III*).
- The binlog shouldn't be rotated while it contains XA transactions in the PREPARED state. Bug fixed #1036040 (*Stewart Smith*).
- Flashcache support resulted in confusing messages in the error log on *Percona Server* startup even when flashcache was not used. This was fixed by adding new boolean option `flashcache`. When set to 0 (default), flashcache checks are disabled and when set to 1 checks are enabled. Error message has been made more verbose including error number and system error message as well. Bug fixed #747032 (*Sergei Glushchenko*).
- Custom server builds would crash when compiled with a non-default maximum number of indexes per table. Upstream MySQL bugs: #54127, #61178, #61179 and #61180. Bug fixed #1042517 (*Sergei Glushchenko*).

82.47 *Percona Server* 5.5.27-28.1

Percona is glad to announce the release of *Percona Server* 5.5.27-28.1 on September 5th, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.27, including all the bug fixes in it, *Percona Server* 5.5.27-28.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.27-28.1 milestone at Launchpad.

82.47.1 Bug Fixes

Percona Server `5.5.27-28.0` would crash or deadlock in *XtraDB* buffer pool code. This was caused by incorrect mutex handling in porting of the recently introduced InnoDB code to *XtraDB*. Bug fixed #1038225 (*Laurynas Biveinis*).

Variables `innodb_adaptive_flushing_method` and `innodb_flush_neighbor_pages` would not correctly translate some values internally. Bug fixed #1039384 (*Laurynas Biveinis*).

82.48 *Percona Server* 5.5.27-28.0

Percona is glad to announce the release of *Percona Server* 5.5.27-28.0 on August 23rd, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.27, including all the bug fixes in it, *Percona Server* 5.5.27-28.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.27-28.0 milestone at Launchpad.

Bug fix for bug #1007268 has been targeted for the next *Percona Server* release. Workaround for this bug exists and it's setting up the `innodb_lazy_drop_table` to 1.

82.48.1 New Features

- *Percona Server* supports tunable buffer size for fast index creation in *InnoDB*. This value was calculated based on the merge block size (which was hardcoded to 1 MB) and the minimum index record size. By adding the session variable `innodb_merge_sort_block_size` block size that is used in the merge sort can now be adjusted for better performance.
- *Percona Server* has implemented ability to have a MySQL *Utility 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.
- New *Expanded Program Option Modifiers* have been added to allow access control to system variables.
- New table `INNODB_UNDO_LOGS` has been added to allow access to undo segment information. Each row represents an individual undo segment and contains information about which rollback segment the undo segment is currently owned by, which transaction is currently using the undo segment, and other size and type information for the undo segment. This information is 'live' and calculated for each query of the table.

82.48.2 Bug Fixes

- Fixed incorrect merge of MySQL bug #61188 fix which caused server to freeze with "has waited at buf0buf.c line 2529 for XXX seconds the semaphore" errors. This regression was introduced in *Percona Server* 5.5.23-25.3. Bug fixed #1026926 (*Stewart Smith*).
- Fixed regression introduced in *Percona Server* 5.5.23-25.3 when merging the upstream fix for MySQL bug #64284. Bug fixed #1015109 (*Stewart Smith*).
- Fixed the upstream MySQL bug #66301. Concurrent INSERT ... ON DUPLICATE KEY UPDATE statements on a table with an AUTO_INCREMENT column could result in spurious duplicate key errors (and, as a result, lost data due to some rows being updated rather than inserted) with the default value of innodb_autoinc_lock_mode=1. Bug fixed #1035225 (*Alexey Kopytov*).
- Removed error log warnings that occured after enabling `innodb_use_sys_stats_table` and before ANALYZE TABLE is run for each table. Bug fixed #890623 (*Alexey Kopytov*).
- Removed the unneeded lrusort.py script. The server now does this sorting automatically and has done for some time. Bug fixed #882653 (*Stewart Smith*).
- Fixed the malformed CHANGE MASTER query in the output of `mysqldump` with `--include-master-host-port` option. Bug fixed #1013432 (*Stewart Smith*).

82.49 *Percona Server* 5.5.25a-27.1

Percona is glad to announce the release of *Percona Server* 5.5.25a-27.1 on July 20th, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.25a, including all the bug fixes in it, *Percona Server* 5.5.25a-27.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.25a-27.1 milestone at Launchpad.

82.49.1 Features

- *Percona Server* has extended standard behavior of variable `secure-file-priv`. When used with no argument, the *LOAD_FILE()* function will always return NULL. The *LOAD DATA INFILE* and *SELECT INTO OUTFILE* statements will fail with the following error: "The MySQL server is running with the `--secure-file-priv` option so it cannot execute this statement". *LOAD DATA LOCAL INFILE* is not affected by the `--secure-file-priv` option and will still work when it's used without an argument.
- *Percona Server* now uses thread based profiling by default, instead of process based profiling. This was implemented because with process based profiling, threads on the server, other than the one being profiled, can affect the profiling information.

82.49.2 Bug Fixes

- *Percona Server* 5.5.24 would crash if userstats were enabled with any replication configured. This was a regression introduced with ssl connections count in statistics tables in Percona Server `5.5.24-26.0`. Bug fixed #1008278 (*Vladislav Lesin*).
- PAM authentication plugin was in different directories in 32bit and 64bit binary tarballs. Bug fixed #1007271 (*Ignacio Nin*).
- Querying I_S.GLOBAL_TEMPORARY_TABLES or TEMPORARY_TABLES would crash threads working with temporary tables. Bug fixed #951588 (*Laurynas Biveinis*).
- mysqld crash message wasn't pointing to *Percona Server* bugtracker. Bug fixed #1007254 (*Vadim Tkachenko*).
- If the tablespace has been created with MySQL 5.0 or older, importing that table could crash *Percona Server* in some cases. Bug fixed #1000221 (*Alexey Kopytov*).
- Server started with `skip-innodb` crashes on *SELECT * FROM INNODB_TABLE_STATS* or *INNODB_INDEX_STATS*. Bug fixed #896439 (*Stewart Smith*).
- Fixed typo for `log_slow_verbosity` in the code. Bug fixed #987737 (*Stewart Smith*).
- Removed some patch-based source code management leftovers from the bzr branch migration. Bug fixed #988383 (*Stewart Smith*).
- Fixed upstream mysql bug #60743, typo in cmake/dtrace.cmake that was making dtrace unusable. Bug fixed #1013455 (*Stewart Smith*).

Other bugfixes: bug #1022481 (*Ignacio Nin*) and bug #987348 (*Ignacio Nin*).

82.50 *Percona Server* 5.5.24-26.0

Percona is glad to announce the release of *Percona Server* 5.5.24-26.0 on June 1st, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.24, including all the bug fixes in it, *Percona Server* 5.5.24-26.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.24-26.0 milestone at Launchpad.

82.50.1 Features

- Percona *PAM Authentication Plugin* has now been integrated into the *Percona Server*.
- *Percona Server* has implemented variable `enforce_storage_engine` which can be used for enforcing the use of a specific storage engine.
- New column, TOTAL_CONNECTIONS_SSL, has been added in the CLIENT_STATISTICS, THREAD_STATISTICS and USER_STATISTICS tables in the INFORMATION_SCHEMA database.

82.50.2 Bug Fixes

- A Server acting as a replication slave with the query cache enabled could crash with glibc detected memory corruption. This bug was introduced in MySQL 5.5.18 and Percona Server inherited it from MySQL. Bug fixed #915814 (*George Ormond Lorch III*).
- Loading LRU dump was preventing shutdown. Bug fixed #712055 (*George Ormond Lorch III*).
- A crash could leave behind an InnoDB temporary table with temporary indexes resulting in an unbootable server. Bug fixed #999147 (*Laurynas Biveinis*).
- Since the output file is simply overwritten when dumping the LRU list, we could end up with a partially written dump file in case of a crash, or when making a backup copy of it. Safer approach has been implemente. It now dumps to a temporary file first, and then rename it to the actual dump file. Bug fixed #686392 (*George Ormond Lorch III*).
- LRU messages are now more verbose for LRU dump. Bug fixed #713481 (*George Ormond Lorch III*).
- Building Percona Server with the Clang compiler resulted in a compiler error. Bug fixed #997496 (*Alexey Kopytov*).
- Variable `thread_statistics` was a reserved word in Percona Server 5.5. As a result, the server variable with that name had to be quoted with backticks when used. Bug fixed #997036 (*Vladislav Lesin*).

82.51 *Percona Server* 5.5.23-25.3

Percona is glad to announce the release of *Percona Server* 5.5.23-25.3 on May 16, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.23, including all the bug fixes in it, *Percona Server* 5.5.23-25.3 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.23-25.3 milestone at Launchpad.

82.51.1 Bug Fixes

- Percona Server would crash on a DDL statement if an XtraDB internal SYS_STATS table was corrupted or overwritten. This is now fixed by detecting the corruption and creating a new SYS_STATS table. Bug fixed #978036 (*Laurynas Biveinis*).

82.52 *Percona Server* 5.5.22-25.2

Percona is glad to announce the release of *Percona Server* 5.5.22-25.2 on April 23, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.22, including all the bug fixes in it, *Percona Server* 5.5.22-25.2 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.22-25.2 milestone at Launchpad.

82.52.1 Bug Fixes

- Fixed non-determinism in one test case of MEMORY engine. Bug fixed #892951 (*Laurynas Biveinis*).

82.53 *Percona Server* 5.5.21-25.1

Percona is glad to announce the release of *Percona Server* 5.5.21-25.1 on March 30, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.21, including all the bug fixes in it, *Percona Server* 5.5.21-25.1 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.21-25.1 milestone at Launchpad.

82.53.1 Bug Fixes

- Fixed a memory corruption regression introduced in 5.5.18-23.0. Bug fixed #915814 (*Alexey Kopytov*).
- Fixed InnoDB compilation warnings on CentOS 5. Bug fixed #962940 (*Laurynas Biveinis*).
- Fixed MySQL upstream bug #64160 that was causing issues on upgrade to 5.5.20 and 5.5.21. Bug fixed #966844 (*Stewart Smith*).

82.54 *Percona Server* 5.5.21-25.0

Percona is glad to announce the release of *Percona Server* 5.5.21-25.0 on March 20, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.21, including all the bug fixes in it, *Percona Server* 5.5.21-25.0 is now the current stable release in the 5.5 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.5.21-25.0 milestone at Launchpad.

Improvements to the XtraDB's sync flush algorithm made in *Percona Server* 5.5.19-24.0 have been reverted because of the performance issues on SSDs (*Laurynas Biveinis*).

82.54.1 New Features

Slow query logging features have been expanded by adding new variable `log_slow_rate_type`. It now provides option to specify the query sampling being taken. If the variable is set up to "query", sampling is done on per query basics instead of session, which is the default (*Oleg Tsarev*).

82.54.2 Bug Fixes

- Fixed MySQL bug #49336, mysqlbinlog couldn't handle stdin when "|" used. Bug fixed: #933969 (*Sergei Glushchenko*).
- Fixed MySQL bugs: #64432 and #54330, broken fast index creation. Bug fixed: #939485 (*Laurynas Biveinis*).

82.55 *Percona Server* 5.5.20-24.1

Percona is glad to announce the release of *Percona Server* 5.5.20-24.1 on February 9th, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.20, including all the bug fixes in it, *Percona Server* 5.5.20-24.1 is now the current stable release in the 5.5 series. All of *Percona* 's software is open-source and free, all the details of the release can be found in the 5.5.20-24.1 milestone at Launchpad.

82.55.1 Bug Fixes

- GCC 4.6 has expanded diagnostics. New warnings reported: #878164 (*Laurynas Biveinis*).
- Dependency issue while installing libmysqlclient15-dev on Ubuntu systems: #803151 (*Ignacio Nin*).
- Dependency issues for libmysqlclient*-dev package(s) on Debian: #656933 (*Ignacio Nin*).
- HandlerSocket failed to compile if the package mysql-devel 5.0 is installed on RHEL5 #922768 (*Ignacio Nin*).

82.56 *Percona Server* 5.5.19-24.0

Percona is glad to announce the release of *Percona Server* 5.5.19-24.0 on January 13th, 2012 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.19, including all the bug fixes in it, *Percona Server* 5.5.19-24.0 is now the current stable release in the 5.5 series. All of *Percona* 's software is open-source and free, all the details of the release can be found in the 5.5.19-24.0 milestone at Launchpad.

82.56.1 New Features

- Variable `innodb_flush_neighbor_pages` can be now set to a new value `cont`. The previously-available option values 0 and 1 now have more descriptive names `none` and `area`. The value of `none` disables the neighbor page flush and `area` matches the default *InnoDB* behavior: any dirty pages in the vicinity of the page selected for flushing may be flushed too. The new option value `cont` improves the neighbor flushing by considering only contiguous blocks of neighbor pages, thus performing the flush by sequential instead of random I/O. (*Yasufumi Kinoshita*, *Laurynas Biveinis*)
- Improvements to the XtraDB's sync flush algorithm. If the XtraDB checkpoint age grows dangerously close to its limit and XtraDB is forced to perform a sync flush, these changes should slightly improve the user query performance instead of completely blocking them. (*Yasufumi Kinoshita*, *Laurynas Biveinis*)

82.56.2 Bug Fixes

- Minor MEMORY engine test suite fix: #849921 (*Laurynas Biveinis*)
- A fix for testsuite integration into Jenkins: #911237 (*Oleg Tsarev*)

82.57 *Percona Server* 5.5.18-23.0

Percona is glad to announce the release of *Percona Server* 5.5.18-23.0 on December 17th, 2011 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.18, including all the bug fixes in it, *Percona Server* 5.5.18-23.0 is now the current stable release in the 5.5 series. All of *Percona* 's software is open-source and free, all the details of the release can be found in the 5.5.18-23.0 milestone at Launchpad.

82.57.1 New Features

Percona Server now supports group commit between *XtraDB* and the replication binlog. Percona has imported the group commit patch from MariaDB and is making the performance improvements that group commit brings available to users of *Percona Server* 5.5. See the Testing the Group Commit Fix blog post for the kind of performance improments that can be expected.

82.57.2 Bug Fixes

- Several crashes were reported when using the `--query-cache-strip-comments` feature of *Percona Server*. We have fixed several causes for crashes, especially around the handling of escaped characters. Bugs fixed: #856404, #705688 (*Oleg Tsarev*)
- The *Expand Table Import* was improved not to hold the InnoDB data dictionary mutex for the full duration of the import operation. This allows queries accessing other InnoDB tables to proceed normally and not be blocked until the import completes. Bug fixed: #901775 (*Alexey Kopytov*)
- As a follow-up to the already-fixed #803865, further fixes were made to the implementation of atomic operations which is used on 32-bit systems when compiled without i686+ support. There were no observed issues with the previous implementation, the fixes were made proactively for benign issues. Additionally, the *Response Time Distribution*, which uses those operations, was made slightly more efficient. Bug fixed: #878022 (*Laurynas Biveinis*)
- An output buffer truncation check in *Response Time Distribution* was fixed. Bug fixed: #810272 (*Laurynas Biveinis*)
- The compilation warnings, produced by GCC versions up to and including 4.6, were audited and fixed. Bug fixed: #878164 (*Laurynas Biveinis*)
- Testsuite stability fix for the percona_status_wait_query_cache_mutex test. Bug fixed: #878709 (*Oleg Tsarev*)
- A missing link was added to the Percona Server upgrade documentation. Bug fixed: #885633 (*Alexey Kopytov*)

82.58 *Percona Server* 5.5.17-22.1

Percona is glad to announce the release of *Percona Server* 5.5.17-22.1 on November 19th, 2011 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.17, including all the bug fixes in it, *Percona Server* 5.5.17-22.1 is now the current stable release in the 5.5 series. All of *Percona* 's software is open-source and free, all the details of the release can be found in the 5.5.17-22.1 milestone at Launchpad.

82.58.1 Bug Fixes

- MyISAM repair-by-sort buffer could not be greater than 4GB even on 64bit architectures. Bug Fixed: #878404 (*Alexey Kopytov*).
- The kill idle transactions feature in *XtraDB* (if enabled) could sometimes cause the server to crash. Bug Fixed: #871722 (*Yasufumi Kinoshita*).
- In a master-master setup when using SET user variables it was possible to have *SHOW SLAVE STATUS* give incorrect output due to a corrupted relay log. Bug Fixed: #860910 (*Alexey Kopytov*).

82.59 *Percona Server* 5.5.16-22.0

Percona is glad to announce the release of *Percona Server* 5.5.16-22.0 on October 14, 2011 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.5.16, including all the bug fixes in it, *Percona Server* 5.5.16-22.0 is now the current stable release in the 5.5 series. All of *Percona* 's software is open-source and free, all the details of the release can be found in the 5.5.16-22.0 milestone at Launchpad.

82.59.1 InnoDB Fake Changes

When restarting a slave server in a replication environment, the process can be speed up by having prefetch threads to warm the server: replay statements and then rollback at commit.

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

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

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

82.59.2 Kill Idle Transactions

NOTE: Percona classes this feature as Beta and possibly not yet suited for production environments.

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

82.59.3 Block Startup until LRU dump is loaded

Added a new boolean option, `innodb_blocking_buffer_pool_restore`, which is `OFF` by default. When set to `ON`, restoring from the LRU dump file is synchronous, i.e. *XtraDB* waits until it is complete before reporting successful startup to the server. Bug Fixed: #785489 (*Alexey Kopytov*).

82.59.4 Behavior changes

The *Fast Index Creation Feature* has been disabled by default to align the behavior with upstream. The boolean variable `innodb_expand_fast_index_creation` has been introduced for enabling or disabling this feature. Bug Fixed: #858945 (*Alexey Kopytov*).

Bug Fixes

- *XtraDB* requires a full table rebuild for foreign key changes. This unnecessarily delays their creation in a mysqldump output, so `--innodb-optimize-keys` should ignore foreign key constrains. Bug Fixed: #859078 (*Alexey Kopytov*).
- After adding an index using the *Fast Index Creation Feature*, statistics for that index provided by *XtraDB* were left in a bogus state until an explicit `ANALYZE TABLE` is executed. Bug Fixed: #857590 (*Alexey Kopytov*).
- *QUERY_RESPONSE_TIME* did not respect `QUERY_RESPONSE_TIME_STATS`. Bug Fixed: #855312 (*Oleg Tsarev*).
- The **mysqldump** option `--innodb-optimize-keys` did not work correctly with tables where the first `UNIQUE` key on non-nullable columns was picked as the clustered index by *XtraDB* in the absence of a `PRIMARY KEY`. Bug Fixed: #851674 (*Alexey Kopytov*).
- The *Slow Query Log* did not log the error number correctly. #830199 (Oleg Tsarev).
- Variable `log-slow-admin-statements` was not listed with `SHOW VARIABLES`. Bug Fixed: #830199 (*Oleg Tsarev*).
- Fixed assertion failure in *XtraDB*. Bug Fixed: #814404 (*Yasufumi Kinoshita*).
- Since `AUTO_INCREMENT` columns must be defined as keys, omitting key specifications and then adding them back in `ALTER TABLE` doesn't work for them. **mysqldump --innodb-optimize-keys** has been fixed to take this into account. Bug Fixed: #812179 (*Alexey Kopytov*).

Other Changes

82.59.5 Improvements and fixes on general distribution:

- #845019, #702376, #795747 (*Alexey Kopytov*, *Ignacio Nin*, *Yasufumi Kinoshita*).

82.59.6 Improvements and fixes on the *Percona Server* Test Suite:

- #760085, #803140, #803137, #803120, #803110, #803100, #803093, #803088, #803076, #803071, #794780, #803072 (*Oleg Tsarev*, *Alexey Kopytov*, *Valentine Gostev*).

82.60 *Percona Server* 5.5.15-21.0

Percona is glad to announce the release of *Percona Server* 5.5.15-21.0 on August 31, 2011 (Downloads are available here and from the Percona Software Repositories).

Based on *MySQL* 5.5.15, including all the bug fixes in it, *Percona Server* 5.5.15-21.0 is now the current stable release in the 5.5 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.5.15-21.0 milestone at Launchpad.

82.60.1 New features

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, making impossible to have a TEXT or BLOB field with that engine implementation. This feature also fixed the upstream #25007.

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*.

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. 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.

82.61 *Percona Server* 5.5.14-20.5

Percona is glad to announce the release of *Percona Server* 5.5.14-20.5 on August 12, 2011 (Downloads are available here and from the Percona Software Repositories).

Based on *MySQL* 5.5.14, including all the bug fixes in it, *Percona Server* 5.5.14-20.5 is now the current stable release in the 5.5 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.5.14-20.5 milestone at Launchpad.

82.61.1 Improvements

Performance Improvements

`fsync()` has been replaced with `fdatasync()` to improve perfomance where possible. The former is intended to sync the metadata of the file also (size, name, access time, etc.), but for the transaction log and the doublewrite buffer, such sync of metadata isn't needed. Bug Fixed: #803270 (*Yasufumi Kinoshita*).

Compatibility Collations

Two new collations, `utf8_general50_ci` and `ucs2_general50_ci`, have been to improve compatibility for those upgrading from *MySQL* 5.0 or 5.1 prior to version 5.1.24.

A fix for a *MySQL* bug (#27877) introduced an incompatible change in collations in *MySQL* 5.1.24. If the following collations were used:

- `utf8_general_ci`
- `ucs2_general_ci`

and any of the indexes contained the German letter "U+00DF SHARP S" `ß` (which became equal to `s`), when upgrading from 5.0 / 5.1.23 or lower:

- any indexes on columns in that situation must be rebuilt after the upgrade, and
- unique constrains may get broken after upgrade due to possible duplicates.

This problem is avoided when upgrading to *Percona Server* by converting the affected tables or columns to the collations introduced:

- `utf8_general_ci` to `utf8_general50_ci`, and
- `ucs2_general_ci` to `ucs2_general50_ci`.

Blueprint: utf8-general50-ci-5.5 (*Alexey Kopytov*).

82.61.2 Bug Fixes

- When adding a table to the cache, the server may evict and close another if the table cache is full. If the closed table was on the `FEDERATED` engine and a replication environment, its client connection to the remote server was closed leading to an unappropriated network error and stopping the Slave SQL thread. Bugs Fixed #813587 / #51196 and #61790 in *MySQL* (*Alexey Kopytov*).
- Querying `global_temporary_tables` caused the server to crash in some scenarios due to insufficient locking. Fixed by introducing a new mutex to protect from race conditions. Bug Fixed: #745241 (*Alexey Kopytov*).
- Using the `innodb_lazy_drop_table` option led to an assertion error when truncating a table in some scenarios. Bug Fixed: #798371 (*Yasufumi Kinoshita*).

82.61.3 Other Changes

Improvements and fixes on platform-specific distribution:

- The compilation of the *Response Time Distribution* patch has been fixed on *Solaris* (supported platform) and *Windows* (experimental). Bug Fixed: #737947 (*Laurynas Biveinis*)

Improvements and fixes on general distribution:

- #766266, #794837, #806975 (*Laurynas Biveinis*, *Stewart Smith*, *Alexey Kopytov*)

Improvements and fixes on the *Percona Server* documentation:

- #803109, #803106, #803097 (*Rodrigo Gadea*)

82.62 *Percona Server* 5.5.13-20.4

Percona is glad to announce the release of *Percona Server* 5.5.13-20.4 on July 1, 2011 (Downloads are available here and from the Percona Software Repositories).

Based on *MySQL* 5.5.13, *Percona Server Percona Server* 5.5.13-20.4 is now the current stable release in the 5.5 series. All of *Percona* 's software is open-source and free, all the details of the release can be found in the 5.5.13-20.4 milestone at Launchpad.

82.62.1 Improvements

SHM Buffer Pool has been replaced with LRU Dump/Restore

The *SHM buffer pool* patch, which provided the ability to use a shared memory segment for the buffer pool to enable faster server restarts, has been removed. Instead, we recommend using the *LRU Dump/Restore* patch which provides similar improvements in restart performance.

Replacement is due to `SHM` buffer pool both being very invasive and not widely used. Improved restart times are better provided by the much safer `LRU D/R` patch which has the advantage of also persisting across machine restarts.

The configuration variables for `my.cnf` have been kept for compatibility and warnings will be printed for the deprecated options (*innodb_buffer_pool_shm_key* and *innodb_buffer_pool_shm_checksum*) if used.

Instructions for disabling the `SHM` buffer pool can be found *here*.

Instructions on setting up `LRU` dump/restore can be found *here*.

82.62.2 Bug Fixes

- On a high concurrency environment with compressed tables, users may experience crashes due to improper mutex handling in `buf_page_get_zip()`. Bug Fix: #802348 (*Yasufumi Kinoshita*).
- *XtraDB* crashed when importing big tables (e.g. 350G) using the *Expand Table Import* feature due to a timeout. Bug Fix: #684829 (*Yasufumi Kinoshita*).
- Partitioning adaptive hash index may leave to a hangup of the server in some scenarios. Bug Fix: #791030 (*Yasufumi Kinoshita*).
- Statistics gathering for each record's update. Bug #791092 (*Yasufumi Kinoshita*)

82.62.3 Other Changes

Improvements and fixes on the *Percona Server* Test Suite

- #693415, #794840, #800035, #800559, #782391, #785566, #790199 (*Oleg Tsarev*, *Yasufumi Kinoshita*, *Stewart Smith*).

Improvements and fixes on platform-specific distribution:

- #737947, #764038 (!), #656933 (*Ignacio Nin*, *Laurynas Biveinis*)

82.63 *Percona Server* 5.5.12-20.3

Percona is glad to announce the release of *Percona Server* 5.5.12-20.3 on June 9, 2011 (Downloads are available here and from the Percona Software Repositories).

Based on *MySQL* 5.5.12, *Percona Server* 5.5.12-20.3 is now the current stable release in the 5.5 series.

82.63.1 Other Changes

- The list of authors of the plugins used have been corrected. Bug Fixes: #723050 (*Yasufumi Kinoshita*)

82.64 *Percona Server* 5.5.11-20.2

Released on April 28, 2011 (Downloads are available here and from the Percona Software Repositories. An experimental build for MacOS is available.)

Percona Server 5.5.11-20.2 is a stable release.

82.64.1 New Features

- `HandlerSocket`, a `NoSQL` plugin for *MySQL*, has been updated to the latest stable version as April 11th, 2011.
- *InnoDB* *Fast Index Creation* now works with **`mysqldump`**, `ALTER TABLE` and `OPTIMIZE TABLE`. (*Alexey Kopytov*)

82.64.2 Variable Changes

Variable `innodb_extra_rsegments` was removed because the equivalent, `innodb_rollback_segments`, has been implemented in *MySQL* 5.5. (*Yasufumi Kinoshita*)

82.64.3 Bug Fixes

- Bug #757749 - Using `ALTER TABLE` to convert an *InnoDB* table to a `MEMORY` table could fail due to a bug in the *Fast Index Creation* patch. (Alexey Kopytov)
- Bug #764395 - *InnoDB* crashed with an assertion failure when receiving a signal on `pwrite()`. The problem was that *InnoDB* I/O code was not fully conforming to the standard on `POSIX` systems. Calls to `fsync()`, `pread()`, and `pwrite()` can be interrupted by a signal. In such cases, *InnoDB* would crash with an assertion failure, rather than just restarting the interrupted call. (*Alexey Kopytov*)
- Bug #766236 - A crash was occurring in some cases when `innodb_lazy_drop_table` was enabled with very large buffer pools. (*Yasufumi Kinoshita*)
- Bug #733317 - `SYS_STATS` internal table of *XtraDB* has been expanded for supporting `innodb_stats_method` from *InnoDB* -plugin. (*Yasufumi Kinoshita*)

82.64.4 Known Bugs

The version of Percona *XtraDB* shown in logs is not correct. The actual version is 1.1.6-20.2 (instead of 1.1.6-20.1).

82.65 *Percona Server* 5.5.10-20.1

Released on April 4, 2011 (Downloads are available here and from the Percona Software Repositories.)

Percona Server 5.5.10-20.1 is a release candidate.

82.65.1 New Features

- Added columns `ROWS_EXAMINED`, `ROWS_SENT`, and `ROWS_READ` to table `PROCESSLIST` and to the output of `SHOW PROCESSLIST`. (*Laurynas Biveinis*)

82.65.2 Variable Changes

- Old status variable innodb_row_lock_numbers was renamed to innodb_current_row_locks. (*Yasufumi Kinoshita*)
- Old system variable `innodb_enable_unsafe_group_commit` was deleted. The existing *MySQL* variable `innodb_support_xa` can be used instead. (*Yasufumi Kinoshita*)

- Old system variable `log_warnings_silence` was renamed to *`log_warnings_suppress`*. (*Oleg Tsarev*)
- Old system variable `log_slow_timestamp_every` was renamed to *`slow_query_log_timestamp_always`*. (*Oleg Tsarev*)
- Old system variable `slow_query_log_microseconds_timestamp` was renamed to *`slow_query_log_timestamp_precision`*. (*Oleg Tsarev*)
- Old system variable `use_global_log_slow_control` was renamed to *`slow_query_log_use_global_control`*. (*Oleg Tsarev*)
- Old system variable `userstat_running` was renamed to *`userstat`*. (*Oleg Tsarev*)
- Old system variable *`innodb_expand_import`* was renamed to *`innodb_import_table_from_xtrabackup`*. (*Yasufumi Kinoshita*)
- Old system variable *`innodb_auto_lru_dump`* was renamed to *`innodb_buffer_pool_restore_at_startup`*. (*Yasufumi Kinoshita*)
- Old system variable *`innodb_overwrite_relay_log_info`* was renamed to *`innodb_recovery_update_relay_log`*. (*Yasufumi Kinoshita*)
- Old system variable *`innodb_pass_corrupt_table`* was renamed to *`innodb_corrupt_table_action`*. (*Yasufumi Kinoshita*)

82.65.3 Bug Fixes

- Bug #724674 - Ported an updated version of the original implementation of the *Remove Excessive Function Calls (fcntl)* feature, which removes some `fcntl` calls to improve performance. (*Oleg Tsarev*)
- Bug #727704 - When using the *Expand Table Import* feature, importing .ibd files created on *MySQL* 5.0 or *Percona Server* versions prior to 5.1.7 could crash the server. (*Yasufumi Kinoshita*)
- *MySQL* bugs 56433 and 51325 - These *MySQL* bugs have been fixed in *Percona Server*. (*Yasufumi Kinoshita*)

82.66 *Percona Server* 5.5.8-20.0

Released on February 16, 2011 (Downloads are available here and from the Percona Software Repositories.)

Percona Server 5.5.8-20.0 is a beta release.

82.66.1 New Features

- *InnoDB* adaptive hash function searches can now be spread across multiple partitions (see *Multiple Adaptive Hash Search Partitions*). Bug fixed: #688866. (*Yasufumi Kinoshita*)
- Information from `SHOW ENGINE INNODB STATUS` was made available in new status variables in *InnoDB* Show Status. Bug fixed: #698797.

82.66.2 Variable Changes

- New variable *`innodb_adaptive_flushing_method`* was added.
- New variable *`innodb_use_global_flush_log_at_trx_commit`* was added. Bug fixed: #635399. (*Yasufumi Kinoshita*)

- New variable `log_warnings_silence` replaced old variable `suppress_log_warning_1592`. Bug fixed: #692413. (*Oleg Tsarev*)
- Old variable `innodb_adaptive_checkpoint` was deleted. Bug fixed: #689450. (*Yasufumi Kinoshita*)
- Old variable `innodb_flush_log_at_trx_commit_session` was deleted. Bug fixed: #635399. (*Yasufumi Kinoshita*)
- Old variable `use_global_long_query_time` was deleted. Bug fixed: #692415. (*Oleg Tsarev*)
- Old variable `innodb_ibuf_accel_rate` was renamed to `innodb_ibuf_merge_rate`. Bug fixed: #695906 (*Yasufumi Kinoshita*)
- Old variable enable_query_response_time_stats was renamed to *`query_response_time_stats`*. (Oleg Tsarev)
- Existing variable *`log_slow_verbosity`* had two new values added: `profiling` and `profiling_use_getrusage`. (Oleg Tsarev)
- Existing variables `profiling_server` and `profiling_use_getrusage` were merged into the Slow Query Log page. (*Oleg Tsarev*)

82.66.3 Other Changes

- Additional information was added to the `LOG` section of the `SHOW STATUS` command. Bug fixed: #693269. (*Yasufumi Kinoshita*)
- The `SHOW PATCHES` command was removed. (*Vadim Tkachenko*)
- The `INFORMATION_SCHEMA` table `XTRADB_ENHANCEMENTS` was removed. (*Yasufumi Kinoshita*)
- Several fields in the `INFORMATION_SCHEMA` table `INNODB_INDEX_STATS` were renamed. Bug fixed: #691777. (Yasufumi Kinoshita)
- The *XtraDB* version was set to 20.0. (*Aleksandr Kuzminsky*)
- Many *InnoDB* compilation warnings were fixed. Bug fixed: #695273. (*Yasufumi Kinoshita*)
- An *Amazon* OS repository was created. Bug fixed: #691996. (*Aleksandr Kuzminsky*)

CHAPTER

EIGHTYTHREE

GLOSSARY

ACID Set of properties that guarantee database transactions are processed reliably. Stands for *Atomicity*, *Consistency*, *Isolation*, *Durability*.

Atomicity Atomicity means that database operations are applied following a "all or nothing" rule. A transaction is either fully applied or not at all.

Consistency Consistency means that each transaction that modifies the database takes it from one consistent state to another.

Durability Once a transaction is committed, it will remain so.

Foreign Key A referential constraint between two tables. Example: A purchase order in the purchase_orders table must have been made by a customer that exists in the customers table.

Isolation The Isolation requirement means that no transaction can interfere with another.

InnoDB A *Storage Engine* for MySQL and derivatives (*Percona Server*, *MariaDB*) originally written by Innobase Oy, since acquired by Oracle. It provides *ACID* compliant storage engine with *foreign key* support. As of *MySQL* version 5.5, InnoDB became the default storage engine on all platforms.

Jenkins Jenkins is a continuous integration system that we use to help ensure the continued quality of the software we produce. It helps us achieve the aims of:

- no failed tests in trunk on any platform,
- aid developers in ensuring merge requests build and test on all platforms,
- no known performance regressions (without a damn good explanation).

LSN Log Serial Number. A term used in relation to the *InnoDB* or *XtraDB* storage engines.

MariaDB A fork of *MySQL* that is maintained primarily by Monty Program AB. It aims to add features, fix bugs while maintaining 100% backwards compatibility with MySQL.

my.cnf The file name of the default MySQL configuration file.

MyISAM A *MySQL Storage Engine* that was the default until MySQL 5.5.

MySQL An open source database that has spawned several distributions and forks. MySQL AB was the primary maintainer and distributor until bought by Sun Microsystems, which was then acquired by Oracle. As Oracle owns the MySQL trademark, the term MySQL is often used for the Oracle distribution of MySQL as distinct from the drop-in replacements such as *MariaDB* and *Percona Server*.

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.

Percona Server Percona's branch of *MySQL* with performance and management improvements.

Storage Engine A *Storage Engine* is a piece of software that implements the details of data storage and retrieval for a database system. This term is primarily used within the *MySQL* ecosystem due to it being the first widely used relational database to have an abstraction layer around storage. It is analogous to a Virtual File System layer in an Operating System. A VFS layer allows an operating system to read and write multiple file systems (e.g. FAT, NTFS, XFS, ext3) and a Storage Engine layer allows a database server to access tables stored in different engines (e.g. *MyISAM*, InnoDB).

XtraDB Percona's improved version of *InnoDB* providing performance, features and reliability above what is shipped by Oracle in InnoDB.

- genindex
- modindex

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

P

Q

R

S

T

U

X