Percona Server Documentation

Release 5.1.73-14.12

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.

Contents:

CHAPTER

ONE

INTRODUCTION

1.1 *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 5.1.65	MySQL 5.1.65
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 5.1.65	MySQL 5.1.65
NoSQL Socket-Level Interface	Yes	
Extra Hash/Digest Functions	Yes	

Extra Diagnostic Features	Percona Server 5.1.65	MySQL 5.1.65
INFORMATION_SCHEMA Tables	54	28
Global Performance and Status Counters	304	291
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 INNODB STATUS	Yes	
Enhanced Mutex Diagnostics	Yes	

Durability and Reliability Enhancements	Percona Server 5.1.65	MySQL 5.1.65
Transactional Replication State	Yes	
Handles Corrupted Tables Gracefully	Yes	

Performance & Scalability Enhancements	Percona Server 5.1.65	MySQL 5.1.65
Support for Multiple I/O Threads	Yes	
Dedicated Purge Threads	Yes	
Self-Tuning Checkpoint Algorithm	Yes	
Fine-Grained Mutex Locking	Yes	
Lock-Free Algorithms	Yes	
Partitioned Adaptive Hash Search	Yes	
Separate Doublewrite File	Yes	
Fast Checksum Algorithm	Yes	
Buffer Pool Pre-Load	Yes	
Fast Shut-Down	Yes	
Support for FlashCache	Yes	
Read-Ahead Improvements	Yes	

Extra Features for DBA/Operations Staff	Percona Server 5.1.65	MySQL 5.1.65
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	
Extended `mysqlbinlog` Utility	Yes	
Changed Page Tracking	Yes	

1.2 The *Percona XtraDB* Storage Engine

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

Percona XtraDB includes all of *InnoDB* ‘s robust, reliable `ACID`-compliant design and advanced `MVCC` architecture, and builds on that solid foundation with more features, more 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

INSTALLATION

2.1 Installing *Percona Server* 5.1 from Binaries

Before installing, you might want to read the *Percona Server 5.1 Release notes*.

Ready-to-use binaries are available from the *Percona Server* download page, including:

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

2.1.1 Using Percona Software Repositories

Percona `apt` Repository

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

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

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

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

Remember to update the local cache:

```
$ apt-get update
```

After that you can install the server and client packages

```
$ apt-get install percona-server-server-5.1 percona-server-client-5.1
```

Supported Platforms

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

Supported Releases

Debian

- 6.0 (squeeze)

Ubuntu

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

Percona *apt* Testing repository

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

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

Apt-Pinning the packages

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

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

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

Percona `yum` Repository

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

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

Automatic Installation

Execute the following command as a root user:

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

You should see some output such as the following:

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

Testing The Repository

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

```
yum list | grep percona
```

You should see output similar to the following:

```
percona-release.noarch                     0.1-3                        @/percona-
↪release-0.1-3.noarch
...
Percona-Server-client-51.x86_64            5.1.73-rel14.11.603.rhel6     percona
Percona-Server-devel-51.x86_64             5.1.73-rel14.11.603.rhel6     percona
Percona-Server-server-51.x86_64            5.1.73-rel14.11.603.rhel6     percona
Percona-Server-shared-51.x86_64            5.1.73-rel14.11.603.rhel6     percona
Percona-Server-test-51.x86_64              5.1.73-rel14.11.603.rhel6     percona
...
percona-xtrabackup.x86_64                  2.2.4-5004.el6                percona
```

Supported Platforms

- `x86_64`
- `i386`

Supported Releases

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

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

Percona *yum* Testing repository

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

[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.

Percona provides repositories for **yum** (RPM packages for *Red Hat*, *CentOS*, and *Amazon Linux AMI*) and **apt** (.deb packages for *Ubuntu* and *Debian*) for software such as *Percona Server*, *XtraDB*, *XtraBackup*, and *Percona Toolkit*. This makes it easy to install and update your software and its dependencies through your operating system's package manager.

This is the recommend way of installing where possible.

YUM-Based Systems

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

```
$ yum install Percona-Server-client-51 Percona-Server-server-51
```

DEB-Based Systems

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

```
$ sudo apt-get install percona-server-server-5.1
```

2.1.2 Using Standalone Packages

RPM-Based Systems

Download the packages of the desired series for your architecture from here.

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

```
$ wget -r -l 1 -nd -A rpm -R "*devel*,*debuginfo*"  \
http://www.percona.com/redir/downloads/Percona-Server-5.1/Percona-Server-5.1.58-12.9/
↪RPM/rhel6/x86_64/
```

Install them in one command:

```
$ rpm -ivh Percona-Server-server-51-5.1.58-rel12.9.271.rhel6.x86_64.rpm \
Percona-Server-client-51-5.1.58-rel12.9.271.rhel6.x86_64.rpm \
Percona-Server-shared-51-5.1.58-rel12.9.271.rhel6.x86_64.rpm
```

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

```
$ rpm -ivh Percona-Server-shared-51-5.1.58-rel12.9.271.rhel6.x86_64.rpm
$ rpm -ivh Percona-Server-client-51-5.1.58-rel12.9.271.rhel6.x86_64.rpm
$ rpm -ivh Percona-Server-server-51-5.1.58-rel12.9.271.rhel6.x86_64.rpm
```

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

What's in each RPM?

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

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

The `Percona-Server-51-debuginfo` package contains debug symbols for use debugging the database server.

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

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

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

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

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

`DEB`-Based Systems

Download the packages of the desired series for your architecture from here.

For example, at the moment of writing, for *Ubuntu* 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-5.1/Percona-Server-5.1.58-12.9/
↪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 installed and the Percona Server installation will be finished by **`apt`**.

What's in each DEB package?

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

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

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

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

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

The `libmysqlclient16` package contains the client shared library. The `16` is a reference to the version of the shared library. The version is incremented when there is a ABI change that requires software using the client library to be recompiled or their source code modified.

2.2 Installing *Percona Server* from a Source Tarball

Fetch and extract the source tarball. For example:

```
$ wget http://www.percona.com/redir/downloads/Percona-Server-5.1/Percona-Server-5.1.
↪58-12.9/source/Percona-Server-5.1.58.tar.gz
$ tar xfz Percona-Server-5.1.58.tar.gz
```

Next, run the configure script. 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:

```
$ ./configure --without-plugin-innobase --with-plugins=partition,archive,blackhole,
↪csv,example,federated,innodb_plugin --without-embedded-server --with-pic --with-
↪extra-charsets=complex --with-ssl --enable-assembler --enable-local-infile --enable-
↪thread-safe-client --enable-profiling --with-readline
```

Now, compile using make

```
$ make
```

Install:

```
$ make install
```

2.3 Installing *Percona Server* from the Bazaar Source Tree

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

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

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

You can now fetch the latest Percona Server 5.1 sources. In the future, we will provide instructions for fetching each specific Percona Server version and building it, but currently we will just talk about building the latest Percona Server 5.1 development tree.

```
$ cd ~/percona-server
$ bzr branch lp:percona-server/5.1
```

You can now change into the 5.1 directory and build Percona Server 5.1:

```
$ make
```

If you are building an older version of Percona Server 5.1, this will fetch the upstream MySQL source tarball and apply the Percona Server patches to it either using quilt or patch. If you are building a modern Percona Server 5.1, this will simply prepare the additional plugins that are distributed as part of Percona Server.

You will now have a directory named Percona-Server that is ready to run the configure script and build.

```
$ ./configure --without-plugin-innobase --with-plugins=partition,archive,blackhole,
↪csv,example,federated,innodb_plugin --without-embedded-server --with-pic --with-
↪extra-charsets=complex --with-ssl --enable-assembler --enable-local-infile --enable-
↪thread-safe-client --enable-profiling --with-readline
$ make
$ make install
```

CHAPTER

THREE

SCALABILITY IMPROVEMENTS

3.1 Improved Buffer Pool Scalability

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

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

Name	Protects
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.

3.1.1 Other Information

Detecting Mutex Contention

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

```
----------
SEMAPHORES
----------
OS WAIT ARRAY INFO: reservation count 36255, signal count 12675
--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.

3.2 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.

3.2.1 System variables:

variable `innodb_ibuf_accel_rate`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 100

Range 100 - 999999999

This variable allows a better control of the background thread processing the insert buffer. Each time the insert buffer merge is performed by the InnoDB master thread, its activity is affected 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_active_contract`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 0(1.0.5), 1(1.0.6)

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_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.

3.2.2 Other Reading

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

3.3 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

3.3.1 Version Specific Information

- *5.1.53-12.4*
  - Added variable *innodb_log_block_size*, method `keep_average` to *innodb_adaptive_checkpoint*.
- *5.1.54-12.5*
  - Added value `ALL_O_DIRECT` to *innodb_flush_method*.

3.3.2 System Variables

variable `innodb_adaptive_checkpoint`

Version Info

- ***5.1.54-12.5*** – Added

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type String

Default Value `none` (1.0.5), `estimate` (1.0.6)

Values `none`, `reflex`, `estimate`, `keep_average` or 0/1/2/3 (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:

- `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.
- `estimate`: 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.
- `keep_average`: 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.

In some cases `innodb_adaptive_checkpoint` needs larger transaction log files (`innodb_adaptive_checkpoint` makes the limit of modified age lower). So, doubling the length of the transaction log files may be safe.

variable **`innodb_adaptive_flushing`**

> **Command Line** No
>
> **Variable Type** BOOL
>
> **Default Value** TRUE
>
> **Range** TRUE/FALSE

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_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_enable_unsafe_group_commit`**

> **Version** This variable is not needed after *XtraDB* 1.0.5.
>
> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** Yes
>
> **Variable Type** Numeric
>
> **Default Value** 0
>
> **Range** 0 - 1

This variable allows you to change the default behavior of *InnoDB* concerning the synchronization between the transaction logs and the binary logs at commit time. The following values are available:

- 0 (default): *InnoDB* keeps transactions in the same order between the transaction logs and the binary logs. This is the safer value but also the slower.
- 1: Transactions can be group-committed but the order between transactions will not be guaranteed to be kept anymore. Thus there is a slight risk of desynchronization between transaction logs and binary logs. However for servers that perform write-intensive workloads (and have RAID without BBU), you may expect a significant improvement in performance.

variable **`innodb_flush_log_at_trx_commit_session`**

Version Info

- ***5.1.49-rel11.3*** – Added

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 3

Range 0-3

This variable implements a session-level version of the existing global variable `innodb_flush_log_at_trx_commit`. It allows a session to override the global setting when a different commit mode is required by the session.

The following values are available:

- 0 / 1 / 2: These values have the same meaning as for the global `innodb_flush_log_at_trx_commit`
- 3 (default): The session will ignore *`innodb_flush_log_at_trx_commit_session`* and stick to the global variable

variable `innodb_flush_method`

Version Info

- ***5.1.54-12.5*** – `ALL_O_DIRECT` option added

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Enumeration

Default Value fdatasync

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

This is an existing *MySQL* 5.1 system variable. It determines the method *InnoDB* uses to flush its data and log files. (See innodb_flush_method in the *MySQL* 5.1 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 not the log files. This option is recommended when *InnoDB* log files are big (more than 8GB), otherwise there might be even a performance degradation. **Note**: When using this option on *ext4* filesystem variable *`innodb_log_block_size`* should be set to 4096 (default log-block-size in *ext4*) in order to avoid the `unaligned AIO/DIO` warnings.

variable `innodb_flush_neighbor_pages`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 1

Range 0-1

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 are available:

- 0: Disables the feature
- 1 (default): Enables the feature

If you use a storage which has no "head seek delay" (e.g. SSD or enough memory for write buffering), 0 may show better performance.

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

- ***`1.0.6-10`*** – 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_read_ahead`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type String

Default Value `linear`

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.

3.3.3 Status Variables

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

- The max checkpoint age
- The current checkpoint age target
- The current age of the oldest page modification which has not been flushed to disk yet.
- The current age of the last checkpoint

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

3.4 More Concurrent Transactions Available

InnoDB provides a fixed number of 1024 undo slots in its rollback segment, leaving room for 1024 transactions to run in parallel. If all the slots are used any new transaction will fail until a slot is freed, which can cause strange behaviors. This change provides a variable to expand the number of undo slots, up to 4072.

This option is provided for servers that run out of undo slots. Use it if you find the following warning in the error log: `Warning:  cannot find a free slot for an undo log.`

We discourage its use unless you get this warning, because it breaks compatibility with other programs. Specifically, it makes the datafiles unusable for ibbackup or for a *MySQL* server that is not run with this option.

When you enable the option, the maximum number of undo slots is extended to 4072, instead of the default fixed value of 1024.

You can then check whether the expanded slots (1025-4072) are used by starting **mysqld** with `innodb_extra_undoslots=OFF`:

- If the expanded slots are used: **mysqld** refuses to start and prints an error in the error log:

```
InnoDB: Error: innodb_extra_undoslots option is disabled, but it was enabled␣
↪before.
InnoDB: The datafile is not normal for mysqld and disabled innodb_extra_undoslots.
InnoDB: Enable innodb_extra_undoslots if it was enabled before, and
InnoDB: ### don't use this datafile with other mysqld or ibbackup! ###
InnoDB: Cannot continue operation for the safety. Calling exit(1).
```

- If the expanded slots are not used: **mysqld** starts and prints only a warning in the error log:

```
InnoDB: Warning: innodb_extra_undoslots option is disabled, but it was enabled␣
↪before.
InnoDB: But extended undo slots seem not used, so continue operation.
```

3.4.1 System Variables

variable `innodb_extra_undoslots`

Command Line Yes

Conf Yes

Scope Global

Dynamic No

Variable Type BOOL

Def OFF

Range ON/OFF

If `ON`, expands the number of undo slots to 4072.

CHAPTER

FOUR

PERFORMANCE IMPROVEMENTS

4.1 Dedicated Purge Thread

With *InnoDB*, data modified by a transaction is written to an undo space in the main tablespace, so that the system can provide read consistency. When a transaction is finished, the corresponding area in the undo space is freed. But if there are so many transactions that the purge thread cannot free space quickly enough, the main tablespace will grow dramatically. This will make performance decrease severely and will possibly consume all the available space on disk. This feature lets you use a dedicated purge thread so that the purge activity will be much quicker. And even if the overall performance will decrease when the purge thread is enabled, performance will be more stable which is often highly desirable.

Purge of the undo space is periodically done by the *InnoDB* main thread, along with other maintenance tasks. In most cases for an OLTP application, the transactions are small and short-running so the undo space can fit in memory in the buffer pool. The purge is then quick and efficient.

But there are several reasons that can make the undo space grow very large and go to disk:

- long-running transactions
- transactions with lots of changes
- too many updates for the purge process to keep up
- In all cases performance will drop dramatically. In standard *InnoDB* it is difficult to find an efficient solution for this problem.

You can now have one or several threads dedicated to the purge. This feature provides several benefits:

- more control over the purge process
- more stable performance (no more performance drops)
- the *InnoDB* main thread does not need to take care of the purge anymore

But be aware that this feature comes at a cost: it reduces the overall performance because purging adds a non-negligible overhead. However we think it is better to have slightly worse but stable performance over time than to have better peak performance but unpredictable sharp drops.

Note: This feature hasn't been ported to the future *Percona Server* releases.

4.1.1 System Variables

The following system variable was introduced by this feature:

variable `innodb_use_purge_thread`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Type ULONG

Default Value 0(~1.0.5), 1(1.0.6~)

Range 0 - 32 (`UNIV_MAX_PARALLELISM`)

Using a value greater than 1 is experimental!

`UNIV_MAX_PARALLELISM` is the maximum number of parallel threads in a parallelized operation

4.1.2 Other Information

With `SHOW INNODB STATUS` you can monitor the number of unpurged transactions: look at History list length in the `TRANSACTIONS` section. This counter increases when a transaction commits and decreases when the purge process removes old versions of rows.

If this counter keeps increasing, it shows that the purge process cannot keep up, so you should use this option to create a dedicated purge thread.

4.1.3 Other Reading

- Tuning for heavy writing workloads
- Reasons for run-away main InnoDB tablespace
- Purge thread spiral of death

4.2 Drop table performance

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".

4.2.1 Version Specific Information

- *5.1.56-12.7* Feature added.

4.2.2 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.

4.2.3 Related Reading

- Drop table performance blog post.

4.3 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.

4.3.1 Detailed Information

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

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.

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.

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.

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.

Prior to release *`5.1.53-12.4`*, it was necessary to recreate your database and *InnoDB* system files when a dedicated file to contain the doublewrite buffer was specified. Beginning with release 5.1.53-12.4, you no longer need to do this.

4.3.2 Version Specific Information

- *`5.1.47-rel11.1`* Full functionality available.
- *`5.1.53-12.4`* Rebuild of database and system files no longer necessary.

4.3.3 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.

4.3.4 Related Reading

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

4.4 Query Cache Enhancements

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

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

4.4.1 Disabling the cache completely

This feature allows the user to completely disable use of the query cache. When the server is compiled with the query cache enabled, the query cache is locked during use by the query cache mutex. This lock can cause performance to decrease in some situations. By disabling use of the query cache altogether when the server is started, any possibility of locking it is eliminated, and performance may be improved.

The query cache can now be disabled at server startup or in an option file by:

```
--query_cache_type=0
```

The default is 1 (query cache enabled).

Note: This variable already exists in standard *MySQL*, but when setting query_cache_type=0, the query cache mutex will still be in used. Setting query_cache_type=0 in *Percona Server* ensures that both the cache is disabled and the mutex is not used.

If query caching is off and a user tries to turn it on from within a session, the following error will be reported:

```
SET GLOBAL query_cache_type=ON;
ERROR 1651(HY000): Query cache is disabled; restart the server with query_cache_
↪type=1 to enable it
```

Note: This variable is implemented in standard *MySQL* from version 5.5.0.

4.4.2 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
```

4.4.3 Ignoring comments

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

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

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

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

4.4.4 System Variables

variable query_cache_strip_comments

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value Off

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

Other Reading

- MySQL general thread states
- Query cache freezes

4.5 Fast *InnoDB* Checksum

Warning: This feature has been deprecated after *Percona Server* `5.1.66-14.2`.

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.

4.5.1 System Variables

variable **`innodb_fast_checksum`**

Command Line	Yes
Config File	Yes
Scope	Global
Dynamic	No
Variable Type	BOOL
Default Value	0

4.6 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.

4.7 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.

4.7.1 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.

4.8 *HandlerSocket*

4.8.1 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.

4.8.2 Version Specific Information

- *5.1.52-12.3* Full functionality available.

Other Information

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

4.8.3 Enabling the Plugin

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

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

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

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

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

4.8.4 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 tabs):

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

And check in SQL that the row has been written:

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

Configuration options

HandlerSocket has many configuration options that are detailed here.

4.8.5 Other Reading

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

4.9 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

4.9.1 Version Specific Information

- *`5.1.49-rel12.0`* : Full functionality available.

4.9.2 Other Information

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

4.9.3 Other Reading

- BUF_READ_AHEAD_AREA Bottleneck

CHAPTER FIVE

FLEXIBILITY IMPROVEMENTS

5.1 Support of Multiple Page Sizes

Warning: This feature has been deprecated in the *Percona Server* *5.1.68-14.6*. 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.

5.1.1 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!

5.2 Suppress Warning Messages

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

5.2.1 Version Specific Information

- *5.1.47-rel11.1* System variable *suppress_log_warning_1592* introduced.

5.2.2 System Variables

variable `suppress_log_warning_1592`

Version Info

- ***`5.1.47-rel11.1`*** – Introduced.

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value OFF

Range ON(='1592') / OFF

This has no effect on replication, but it can fill up your error log with unnecessary messages. This variable allows you to completely disable logging of this warning.

NOTE: Only *MySQL* 5.1 is subject to this bug. A partial solution has been published beginning with *MySQL* 5.1.37, but this bug still appears in some situations.

When `ON`, disables reporting of warning #1592 (unsafe statement for binary logging).

All warnings #1592 will be disabled, so you will not be able to know if your statements are really safe to replicate anymore. Use it at your own risk and only if you understand what you are doing.

In some circumstances, *MySQL* will warn you that a statement is unsafe to replicate even though it is perfectly safe. For example, in versions lower than 5.1.59, the warning will look like:

```
090213 16:58:54 [Warning] Statement is not safe to log in statement format.
```

or in versions 5.1.59 or higher:

```
010214 12:08:52 [Warning] Statement may not be safe to log in statement format.
```

5.2.3 Related Reading

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

5.3 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.

5.3.1 Version Specific Information

- *5.1.50-rel12.1* Full functionality.

5.3.2 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

5.4 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 variable 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.

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

5.4.1 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/*!*/;
```

5.4.2 Version Specific Information

- *5.1.62-13.3* Full functionality.

5.4.3 Client Command Line Parameter

variable `rewrite-db`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type String

Default Value Off

5.4.4 Related Reading

- WL #36

5.5 Replication Stop Recovery

After a replication stop, this feature allows skipping a user-specified number of events from the binary log, rather than all events remaining in the current event group.

The discussion of the system variable `sql_slave_skip_counter` in the MySQL 5.1 Reference Manual notes:

When using this statement, it is important to understand that the binary log is actually organized as a sequence of groups known as event groups. Each event group consists of a sequence of events.

- For transactional tables, an event group corresponds to a transaction.
- For nontransactional tables, an event group corresponds to a single SQL statement.

A single transaction can contain changes to both transactional and nontransactional tables.

When you use SET GLOBAL sql_slave_skip_counter to skip events and the result is in the middle of a group, the slave continues to skip events until it reaches the end of the group. Execution then starts with the next event group.

When used, this feature modifies the standard *MySQL* behavior described above. It provides the user additional flexibility in recovering from a replication stop, but it should be used with caution. In particular, *`sql_slave_statement_skip_counter`* and `sql_slave_skip_counter` should not be used at the same time (i.e., they should not both be non-zero), as this may cause unintended behavior.

5.5.1 Version Specific Information

- *`5.1.49-rel12.0`* Full functionality released.

5.5.2 System Variables

The following status variable was introduced by this feature.

variable `sql_slave_statement_skip_counter`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Numeric

Default Value 0

Range 1-MAXLONGINT

This variable specifies the number of events in the current event group to skip.

IMPORTANT: As mentioned above, never set both this variable and sql_slave_skip_counter to non-zero values at the same time. The behavior that will result from this is unknown.

Consider this case:

```
SET GLOBAL sql_slave_statement_skip_counter=n;
SET GLOBAL sql_slave_skip_counter=m;
```

The first statement will skip the next `n` events in the current event group. At that point, one would expect the second statement to operate as described in the *MySQL* documentation for `sql_slave_skip_counter`. However, depending on the values of `n` and `m` and the contents of the binary log, the effects of the interactions of the two statements can involve very complicated scenarios. **THESE SCENARIOS HAVE NOT BEEN TESTED**.

5.5.3 Example

In the example here, a master server with two slaves is assumed. A table is created on the master and then a replication error occurs.

Then the difference between using the new system variable *`sql_slave_statement_skip_counter`* and the *MySQL* system variable `sql_slave_skip_counter` to repair the replication stop is shown.

Setup

- Prepare the master server:

```
DROP TABLE IF EXISTS t;
CREATE TABLE t(id INT UNIQUE PRIMARY KEY) ENGINE=|InnoDB|;
INSERT INTO t VALUES (1),(2);
```

- Prepare Slave #1 and Slave #2:

```
DROP TABLE IF EXISTS t;
CREATE TABLE t(id INT UNIQUE PRIMARY KEY) ENGINE=|InnoDB|;
INSERT INTO t VALUES (1),(2),(3),(5);
```

- Start replication of the master to the two slaves.

Create the Replication Error

Run the following on the master:

```
START TRANSACTION;
INSERT INTO t VALUES (3);
INSERT INTO t VALUES (4);
COMMIT;
```

Both slaves will fail with the following error:

```
Error ``Duplicate entry ``3`` for key ``PRIMARY```` on query. Default database:␣
↪``test``. Query: ``INSERT INTO t VALUES (3)``
```

Repair the Replication Error

Now, let's compare the effects of using `sql_slave_skip_counter` to do the repair versus using *`sql_slave_statement_skip_counter`* to do it.

Slave Repair Using sql_slave_skip_counter

To repair either slave, do the following:

```
SET GLOBAL sql_slave_skip_counter=1;
START SLAVE;
```

This will cause the slave to skip the following statement:

```
INSERT INTO t VALUES (3);
```

In addition, since we are in the middle of an event group in the binary log, all other events in the event group will also be skipped, since that is the standard behavior of `sql_slave_skip_counter`. In this case, the following statements will also be skipped:

```
INSERT INTO t VALUES (4);
COMMIT;
```

Now run the following on the slave:

```
SELECT * FROM t;
```

This will give the result:

```
id
1
2
3
5
```

Since the table's original contents are unchanged, this shows that `sql_slave_skip_counter` caused the entire event group to be skipped.

Slave Repair Using sql_slave_statement_skip_counter

To repair either slave, do the following:

```
SET GLOBAL sql_slave_statement_skip_counter=1;
START SLAVE;
```

This will cause the slave to skip the following statement:

```
INSERT INTO t VALUES (3);
```

However, unlike with `sql_slave_skip_counter`, this is the only event that will be skipped. Every other event in the current event group in the binary log will be executed. In this case, that means these statements will not be skipped; they will also be executed:

```
INSERT INTO t VALUES (4);
COMMIT;
```

Now, we can see the difference in results when we run:

```
SELECT * FROM t;
```

Now, our results are:

```
id
1
2
```

In this case, `sql_slave_statement_skip_counter` caused the server to skip only single statement, not the entire remainder of the event group. The result is that the original table has been updated.

Other Reading

- MySQL 5.1 Reference Manual - "SET GLOBAL sql_slave_skip_counter Syntax"

5.6 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.

5.6.1 Version Specific Information

- *5.1.52-12.3* Full functionality available.

5.6.2 Other Information

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

5.6.3 Other reading

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

5.7 Ignoring missing tables in mysqldump

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

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

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

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

5.7.1 Version Specific Information

- *1.0.6-rel10.1* **mysqldump** option `--ignore-create-error` introduced

CHAPTER

SIX

RELIABILITY IMPROVEMENTS

6.1 Too Many Connections Warning

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

6.1.1 Version-Specific Information

- *5.1.49-rel12.0* Full functionality available.

6.2 Error Code Compatibility

Percona Server with *XtraDB* has error code incompatibilities with *MySQL* 5.5. It is important to maintain compatibility in the error codes used by the servers. For example, scripts that may be run on both servers could contain references to error codes.

The reasons for the current incompatibilities are:

- *Percona Server* with *XtraDB* contains features that have been backported from MyQL 5.5. Some of the *MySQL* 5.5 features added new error codes.
- Some *Percona Server* with *XtraDB* features have added new error codes.

The solution to the first problem is to preserve *MySQL* 5.5 error codes in the *Percona Server*. An example of where this has been done is *Percona Server* feature Query Cache Enhancements. This feature adds error `ER_QUERY_CACHE_DISABLED` to the *Percona Server*, which is defined as error code 1651 in *MySQL* 5.5.

After migrating *Percona Server* / *XtraDB* to *MySQL* 5.5, users might experience troubles because of this.

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

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

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

from the file `errmsg.txt`.

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

I propose patch to `comp_err`.

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

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

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

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

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

6.2.1 Version-Specific Information

- *5.1.49-rel12.0* Full functionality available.

6.3 Handle Corrupted Tables

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

This feature adds a new system variable.

6.3.1 System Variables

variable innodb_pass_corrupt_table

Command Line	Yes
Config File	Yes
Scope	Global
Dynamic	Yes
Variable Type	ULONG
Default Value	0
Range	0 - 1

Pass corruptions of user tables as `corrupt table` instead of crashing itself, when used with innodb_file_per_table. All file I/O for the datafile after detected as corrupt is disabled, except for the deletion.

6.4 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.

6.4.1 Restrictions

When *innodb_overwrite_relay_log_info* is 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*.

6.4.2 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_overwrite_relay_log_info 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.

6.4.3 System Variables

One new system variable was introduced by this feature.

variable **`innodb_overwrite_relay_log_info`**

Version Info

- ***`1.0.3-4`*** – 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*.

6.4.4 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

6.5 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.

6.5.1 Version Specific Information

- *5.1.52-12.3* - Introduced

CHAPTER

SEVEN

MANAGEMENT IMPROVEMENTS

7.1 Fast *InnoDB* Recovery Process

This feature implements several changes related to the recovery process (bug fixes and speed changes). The *InnoDB* plugin subsequently implemented similar functionality, so the feature is currently unused. The system variable it implemented now does nothing and has been kept to maintain compatibility.

- `recv_apply_hashed_log_recs()` may hang up when meets DB_TABLESPACE_DELETED pages
- Insert buffer operation may destroy the page during its recovery process adjustment of smaller sleep period in loop
- `buf_flush_insert_sorted_into_flush_list()` change to don't sort (new variable `innodb_fast_recovery` (default false(1.0.5), true(1.0.6~) - if set true, sorting is enabled.)

Output statistics of recovery process after it:

```
-------------------
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
```

7.1.1 Other Information

Is the `buf_flush_insert_sorted_into_flush_list()` change correct?

InnoDB recovers the each pages not in order of the oldest un-flushed change's *LSN*. But flush_list must has un-flushed blocks in order of the *LSN*. So, normal *InnoDB* does bubble sorting for each inserting block to `flush_lsn`. It is very expensive operation.

Then, why the order must be kept?

InnoDB should get the LSN of the oldest un-flushed change in all blocks for checkpointing. It is the oldest un-flushed change's *LSN* of the last block in the `flush_list`.

So, it may be correct that the last blocks' `oldest_modification` only have to keep the oldest `oldest_modification` in the buffer pool.

This change is simple. If the new page's `oldest_modification` is:

```
[newer than any oldest_modification in flushlist]
```

add to first of the `flush_list`:

```
[older than any oldest_modification in flushlist]
```

add to last of the `flush_list`:

```
[else]
```

These operation should not break consistency of `flush_list`. However, it may cause same-LSN-aged cluster of many pages and much flushing operation. But anyway, the most of the flushing should be done during the recovery process.

```
--- innodb_plugin-1.0.3_orig/buf/buf0flu.c      2009-07-07 18:03:24.000000000 +0900
+++ innodb_plugin-1.0.3_tmp/buf/buf0flu.c       2009-07-07 18:06:47.000000000 +0900
@@ -108,6 +108,17 @@
        prev_b = NULL;
        b = UT_LIST_GET_FIRST(buf_pool->flush_list);

+       if (b == NULL || b->oldest_modification < block->page.oldest_modification) {
+               UT_LIST_ADD_FIRST(flush_list, buf_pool->flush_list, &block->page);
+      } else {
+               b = UT_LIST_GET_LAST(buf_pool->flush_list);
+               if (b->oldest_modification < block->page.oldest_modification) {
+                       /* align oldest_modification not to sort */
+                       block->page.oldest_modification = b->oldest_modification;
+               }
+               UT_LIST_ADD_LAST(flush_list, buf_pool->flush_list, &block->page);
+       }
```

```
+/*
        while (b && b->oldest_modification > block->page.oldest_modification) {
                ut_ad(b->in_flush_list);
                prev_b = b;
@@ -120,6 +131,7 @@
                UT_LIST_INSERT_AFTER(flush_list, buf_pool->flush_list,
                                     prev_b, &block->page);
        }
+*/

 #if defined UNIV_DEBUG || defined UNIV_BUF_DEBUG
        ut_a(buf_flush_validate_low());
```

7.1.2 System Variables

One new system variable was introduced by this feature.

variable `innodb_fast_recovery`

Command Line	Yes
Config File	Yes
Scope	Global
Dynamic	No
Variable Type	BOOL
Default Value	FALSE
Range	TRUE/FALSE
Default Value	false(1.0.5), true(1.0.6)) - if set true, the change is enabled.

variable `innodb_recovery_stats`

Command Line	No
Variable Type	BOOL
Default Value	FALSE
Range	TRUE/FALSE

7.1.3 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

7.2 *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.

If you;re using *Percona Server*, you can determine the actual size of the data dictionary. (See Show Hashed Memory.) However, if you're not using *Percona Server*, you can still make an estimate of the data dictionary's size. (See "Estimating the Data Dictionary Size" below.)

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).

7.2.1 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*.

7.2.2 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.

7.2.3 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.

7.2.4 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.

Estimating the Data Dictionary Size

Percona Server provides instrumentation to show the data dictionary size directly, but if you‘‘re not using *Percona Server*, you can estimate the size of the data dictionary. By calculating how much memory *InnoDB* has allocated that is not attributable to the buffer pool, etc., you will have an idea of how much allocated memory is not accounted for. This will not be the exact size of the data dictionary, but it will be a reasonable estimate.

To make this estimate, first locate the following lines in the output of `SHOW INNODB STATUS`:

```
----------------------
BUFFER POOL AND MEMORY
----------------------
Total memory allocated 13563931762; in additional pool allocated 1048576
Buffer pool size   524288
```

The line beginning `Total memory allocated` shows the total memory *InnoDB* has allocated. The next line shows the buffer pool size in pages; you can either multiply that by the page size to get a value in bytes, or determine the size of the *InnoDB* buffer pool by executing the command `SHOW VARIABLES LIKE 'innodb_buffer_pool_size'` . The latter is easier, if you use *MySQL* 5.0 or later, so for the purpose of this example, assume we use that and it gives the value 8589934592. Finally, subtract the *InnoDB* buffer pool size from the total memory allocated:

```
13563931762 - 8589934592 = 4973997170
```

So, there is a little over 4.6 GB of memory that *InnoDB* has allocated and which is unaccounted for. This is a pretty large amount of extra memory usage; quite a bit more than the gigabyte or so suggested as a maximum. So, you may benefit from using this feature.

7.2.5 Other reading

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

7.3 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.

7.3.1 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.
...
```

7.3.2 Version Specific Information

- *5.1.50-rel12.1* Introduced variable *innodb_expand_import*.

7.3.3 System Variables

variable innodb_expand_import

Version Info

- ***5.1.50-rel12.1*** – 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).

7.3.4 Other reading

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

7.4 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.

7.4.1 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.

7.4.2 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  |
+-------------------------------+-------+
```

7.4.3 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.

7.4.4 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.

7.4.5 System Variables

variable `innodb_auto_lru_dump`

Version Info

- *`1.0.6-9`* – 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. When set to a non-zero value, an automatic restore of the buffer pool is also performed at startup, as described above.

variable `innodb-blocking-lru-restore`

Version Info

- *`5.1.59-13.0`* – 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.

7.4.6 `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`

7.4.7 Other reading

- Save / restore buffer pool

7.5 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`*.

7.5.1 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*.

`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`.

7.5.2 Version Specific Information

- *`5.1.49-rel12.0`*: Variable *`fast_index_creation`* implemented.
- *`5.1.56-12.7`*: Expanded the applicability of fast index creation to **`mysqldump`**, `ALTER TABLE`, and `OPTIMIZE TABLE`.

7.5.3 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

7.5.4 Other Reading

- Thinking about running OPTIMIZE on your InnoDB Table? Stop!

7.6 Expanded Fast Index Creation

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

7.6.1 Enabling Expanded Fast Index Creation

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

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

mysqldump

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

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

ALTER TABLE

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

- Only non-unique keys can be involved in this optimization.

- If the table contains foreign keys, or a foreign key is being added as a part of the current `ALTER TABLE` statement, the optimization is disabled for all keys.
- 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.

`OPTIMIZE TABLE`

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

Caveats

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

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

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

7.6.2 Version Specific Information

- *`5.1.59-13.0`*: Variable *`expand_fast_index_creation`* implemented. This variable controls whether fast index creation optimizations made by Percona are used.

7.6.3 System Variables

variable `expand_fast_index_creation`

Command Line	Yes
Config File	No
Scope	Local/Global
Dynamic	Yes
Variable Type	Boolean
Default Value	OFF

Range ON/OFF

7.6.4 Other Reading

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

7.7 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.

7.7.1 Version-Specific Information

- *`5.1.49-rel12.0`*: Full functionality available.
- *`5.1.66-14.1`*: Variable *`have_flashcache`* introduced.

7.7.2 System Variables

variable `have_flashcache`

Version Info

- ***`5.1.66-14.1`*** – Variable introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Variable Type Boolean

Range Yes/No

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

7.7.3 Status Variables

variable `Flashcache_enabled`

Scope Global

Variable Type Boolean

Range OFF/ON

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

7.7.4 Other Information

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

7.7.5 Other reading

- Releasing Flashcache
- Level 2 Flash cache is there

7.8 *Percona Toolkit* UDFs

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

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

7.8.1 Version Specific Information

- *5.1.53-12.4*: Began distributing `libfnv1a_udf`, `libfnv_udf`, and `libmurmur_udf`.

7.8.2 Other Information

- Author / Origin: Baron Schwartz

7.8.3 Installation

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

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

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

7.8.4 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`.

7.8.5 Other Reading

- *Percona Toolkit* documentation

7.9 Support for Fake Changes

Replication in *MySQL* is single-threaded and because it needs to read the data before it can execute the queries, this can decrease performance when the full data set is not already in the buffer pool. The fake changes feature allows a process to replay a set of operations against a replication slave to warm the buffer pool for the replication apply thread.

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

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

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

7.9.1 Caveats

`DML` operations are supported

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

`DDL` operations are not supported

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

Explicit `COMMIT` will lead to an error

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

7.9.2 System Variables

variable `innodb_fake_changes`

Version Info

- *`5.1.59-13.0`* – Introduced

Scope `GLOBAL`

Type `BOOLEAN`

Dynamic `YES`

Default Value `FALSE`

This variable enables the *Fake Changes* feature.

variable **`innodb_locking_fake_changes`**

Version Info

- ***`5.1.66-14.2`*** – Introduced

Scope `GLOBAL`

Type `BOOLEAN`

Dynamic `YES`

Default Value `TRUE`

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

7.9.3 Implementation Details

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

7.9.4 Related Reading

- on MySQL replication prefetching

7.10 Kill Idle Transactions

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

7.10.1 System Variables

variable `innodb_kill_idle_transaction`

Version Info

- *`5.1.59-13.0`* – 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.

7.11 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 with the *`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*.

7.11.1 User statements for handling the XtraDB changed page bitmaps

In *Percona Server* *`5.1.67-14.4`* new statements have been introduced for handling the changed page bitmap tracking. All of these statements require `SUPER` privilege.

- `FLUSH CHANGED_PAGE_BITMAPS` - this statement can be used for synchronous bitmap write for immediate catch-up with the log checkpoint. This is used by innobackupex to make sure that XtraBackup indeed has all the required data it needs.

- `RESET CHANGED_PAGE_BITMAPS` - this statement will delete all the bitmap log files and restart the bitmap log file sequence.
- `PURGE CHANGED_PAGE_BITMAPS BEFORE <lsn>` - this statement will delete all the change page bitmap files up to the specified log sequence number.

7.11.2 Additional information in SHOW ENGINE INNODB STATUS

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

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

7.11.3 INFORMATION_SCHEMA Tables

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

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

Columns

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

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

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

7.11.4 System Variables

variable **`innodb_max_changed_pages`**

Version Info

- ***`5.1.65-14.0`*** – Variable `innodb_changed_pages_limit` introduced
- ***`5.1.67-14.4`*** – 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

- ***`5.1.65-14.0`*** – 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

- ***`5.1.66-14.2`*** – 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

EIGHT

DIAGNOSTICS IMPROVEMENTS

8.1 *InnoDB* Statistics

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

8.1.1 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)

Note: Beginning in release 5.1.56-12.7, a variable with the same and functionality was implemented in the upstream *InnoDB*.

variable `innodb_stats_auto_update`

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`

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`

Version 5.1.49-11.3 Variable introduced

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.

8.1.2 INFORMATION_SCHEMA Tables

table `INFORMATION_SCHEMA.`**`INNODB_SYS_STATS`**

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

- **`SCHEMA`** – Schema (database) name
- **`NAME`** – Table name
- **`ID`** – Table ID
- **`N_COLS`** – Number of Columns
- **`TYPE`** –
- **`MIX_ID`** – This value is obsolete, value is always 0
- **`MIX_LEN`** – Contains 0 for regular tables and 1 for temporary tables
- **`CLUSTER_NAME`** – This value isn't supported anymore, value is always NULL
- **`SPACE`** – Tablespace ID

table `INFORMATION_SCHEMA.`**`INNODB_SYS_INDEXES`**

Shows the information about InnoDB indexes

Columns

- **`TABLE_ID`** – Table ID
- **`ID`** – Index ID
- **`NAME`** – Index Name
- **`N_FIELDS`** – Number of fields

- **TYPE** –
- **SPACE** – Tablespace ID
- **PAGE_NO** – The page offset within its tablespace

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_keys** – Estimate rows per 1 key value. ([1 column value], [2 columns value], [3 columns value], ...).
- **index_size** – Number of index pages.
- **index_pages** – Number of leaf pages.

8.1.3 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)
```

8.1.4 Other reading

- InnoDB Table/Index stats

8.2 User Statistics

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

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

8.2.1 Other Information

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

8.2.2 System Variables

variable `userstat_running`

Version Info

- ***`5.1.49-rel11.3`*** – variable introduced

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type BOOLEAN

Default Value OFF

Range ON/OFF

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

8.2.3 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 while servicing this client‘‘s connections.
- **BYTES_RECEIVED** – The number of bytes received from this client’s connections.
- **BYTES_SENT** – The number of bytes sent to this client’s connections.
- **BINLOG_BYTES_WRITTEN** – The number of bytes written to the binary log from this client’s connections.
- **ROWS_FETCHED** – The number of rows fetched by this client’s connections.
- **ROWS_UPDATED** – The number of rows updated by this client’s connections.
- **TABLE_ROWS_READ** – The number of rows read from tables by this client’s connections. (It may be different from `ROWS_FETCHED`.)
- **SELECT_COMMANDS** – The number of `SELECT` commands executed from this client’s connections.
- **UPDATE_COMMANDS** – The number of `UPDATE` commands executed from this client’s connections.
- **OTHER_COMMANDS** – The number of other commands executed from this client’s connections.
- **COMMIT_TRANSACTIONS** – The number of `COMMIT` commands issued by this client’s connections.
- **ROLLBACK_TRANSACTIONS** – The number of `ROLLBACK` commands issued by this client’s connections.
- **DENIED_CONNECTIONS** – The number of connections denied to this client.
- **LOST_CONNECTIONS** – The number of this client’s connections that were terminated uncleanly.
- **ACCESS_DENIED** – The number of times this client’s connections issued commands that were denied.
- **EMPTY_QUERIES** – The number of times this client’s connections sent empty queries to the server.

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

Example:

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

table `INFORMATION_SCHEMA.`**`INDEX_STATISTICS`**

Columns

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

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

This table makes it possible to do many things that were difficult or impossible previously. For example, you can use it to find unused indexes and generate DROP commands to remove them. 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** – ID of the thread.
- **TOTAL_CONNECTIONS** – The number of connections created from this thread.
- **CONCURRENT_CONNECTIONS** – The number of concurrent connections from this thread.
- **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.

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 while servicing this user's connections.
- **`BYTES_RECEIVED`** – The number of bytes received from this user's connections.
- **`BYTES_SENT`** – The number of bytes sent to this user's connections.
- **`BINLOG_BYTES_WRITTEN`** – The number of bytes written to the binary log from this user's connections.
- **`ROWS_FETCHED`** – The number of rows fetched by this user's connections.
- **`ROWS_UPDATED`** – The number of rows updated by this user's connections.
- **`TABLE_ROWS_READ`** – The number of rows read from tables by this user's connections. (It may be different from `ROWS_FETCHED`.)
- **`SELECT_COMMANDS`** – The number of `SELECT` commands executed from this user's connections.
- **`UPDATE_COMMANDS`** – The number of `UPDATE` commands executed from this user's connections.
- **`OTHER_COMMANDS`** – The number of other commands executed from this user's connections.
- **`COMMIT_TRANSACTIONS`** – The number of `COMMIT` commands issued by this user's connections.
- **`ROLLBACK_TRANSACTIONS`** – The number of `ROLLBACK` commands issued by this user's connections.
- **`DENIED_CONNECTIONS`** – The number of connections denied to this user.
- **`LOST_CONNECTIONS`** – The number of this user's connections that were terminated uncleanly.
- **`ACCESS_DENIED`** – The number of times this user's connections issued commands that were denied.
- **`EMPTY_QUERIES`** – The number of times this user's connections sent empty queries to the server.

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

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

Example:

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

8.2.4 Commands Provided

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

These commands discard the specified type of stored statistical information.

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

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

8.3 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.

8.3.1 Version Specific Information

- `5.1.x`:
  - Microsecond resolution included in official *MySQL* server; this feature adds statistics.
- `5.1.47-11.0`:
  - Full functionality of *`use_global_log_slow_control`* available, except all option and link between *`use_global_long_query_time`* and *`use_global_log_slow_control`* `=long_query_time` available.
- `5.1.47-12.0`:
  - Full functionality of *`use_global_log_slow_control`* available.
  - Fixed #600684 - *`log_slow_verbosity`* `=Innodb` doesn't work on slave (statement-based replication)
- *`5.1.53-11.7`*:
  - Fixed #643149 - Slow query log entries were not being done in the usual parsing format; made parsing difficult. Made compatible.

8.3.2 Other Information

- Author / Origin: Maciej Dobrzanski, Percona

8.3.3 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. This feature has been implemented in the upstream *MySQL* as a command line option and requires server restart in order to be enabled/disabled. *Percona Server* has implemented this as global variable, which can be enabled/disabled 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_limit`

Command Line Yes

Config File Yes

Scope Global, session

Dynamic Yes

Range 1-ULONG_MAX (either 4294967295 or 18446744073709551615, depending on the platform)

Specifies that only a fraction of sessions should be logged. Logging is enabled for every nth session. By default, n is 1, so logging is enabled for every 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. 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. For example, if you set the value to 100, then one percent of sessions will be logged in their entirety. Note that this feature will not work well if your application uses any type of connection pooling or persistent connections.

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 TRUE

Range TRUE/FALSE

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

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

variable log_slow_timestamp_every

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 log_slow_verbosity

Command Line Yes

Config File Yes

Scope Global, session

Dynamic Yes

Default Value microtime

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 (mandatory).
- query_plan: Log information about the query's execution plan (optional).
- innodb: Log *InnoDB* statistics (optional).

- `minimal`: Equivalent to enabling just `microtime`.
- `standard`: Equivalent to enabling `microtime,innodb`.
- `full`: Equivalent to all other values OR'ed together.

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 `profiling_server`

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

When `ON`, this variable enables profiling of all queries (in all connections).

variable `profiling_use_getrusage`

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

When `ON`, this variable enables usage of the getrusage function in profiling. A possible problem is that this function is very expensive, and with profiling_server enabled it can cause performance degradation.

variable `slow_query_log_microseconds_timestamp`

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

When `TRUE`, entries to the slow log are done in microsecond precision.

Normally, the slow query log contains output in this format:

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

If TRUE, this variable causes the format to be like this:

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

variable use_global_log_slow_control

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value None

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

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

variable use_global_long_query_time

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

If 1 is set, global long_query_time is always used instead of the local long_query_time, and the local long_query_time is updated by global when used. 0 is same as normal behavior. (default 0)

8.3.4 Other Information

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
```

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
```

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: 0.000659  Lock_time: 0.000070  Rows_sent: 0  Rows_examined: 30  Rows_
↪affected: 0  Rows_read: 30
```

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`

Memory Footprint

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

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

Values and context:

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

Query Plan Information

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

```
# QC_Hit: No  Full_scan: No  Full_join: No  Tmp_table: Yes  Disk_tmp_table: No
# Filesort: Yes  Disk_filesort: No  Merge_passes: 0
```

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

InnoDB Usage Information

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

```
# innodb_IO_r_ops: 1  innodb_IO_r_bytes: 16384  innodb_IO_r_wait: 0.028487
# innodb_rec_lock_wait: 0.000000  innodb_queue_wait: 0.000000
# innodb_pages_distinct: 5
```

Values:

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

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

8.3.5 Related Reading

- Impact of logging on MySQL's performance
- log_slow_filter Usage
- Blueprint in Launchpad

8.4 Count *InnoDB* Deadlocks

When running a transactional application you have to live with deadlocks. They are not problematic as long as they do not occur too frequently. The standard `SHOW INNODB STATUS` gives information on the latest deadlocks but it is not very useful when you want to know the total number of deadlocks or the number of deadlocks per unit of time.

This change adds a status variable that keeps track of the number of deadlocks since the server startup, opening the way to a better knowledge of your deadlocks.

This feature was provided by Eric Bergen under BSD license (see InnoDB Deadlock Count Patch).

It adds a new global status variable (*`innodb_deadlocks`*) showing the number of deadlocks.*

You can use it with `SHOW GLOBAL STATUS`, e.g.:

```
mysql> SHOW GLOBAL_STATUS LIKE ``innodb_deadlocks``;
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| innodb_deadlocks | 323   |
+------------------+-------+
```

or with `INFORMATION_SCHEMA`, e.g.:

```
mysql> SELECT VARIABLE_VALUE FROM INFORMATION_SCHEMA.GLOBAL_STATUS WHERE VARIABLE_
↪NAME = ``innodb_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.

8.4.1 Version Specific Information

- *5.1.47-rel11.1*: Full functionality available.

8.4.2 Status Variables

One new status variable was introduced by this feature.

variable innodb_deadlocks

Variable Type LONG

Scope Global

8.4.3 Related Reading

- Original post by Eric Bergen

8.5 Log All Client Commands (syslog)

When enabled, this feature causes all commands run by the command line client to be logged to `syslog`. If you want to enable this option permanently, add it to the [mysql] group in my.cnf.

8.5.1 Version Specific Information

- *5.1.49-rel12.0*: Full functionality available.

8.5.2 Other Information

- Author / Origin: Percona

8.5.3 Client Variables

variable syslog

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Variable Type Boolean

Default Value OFF

Range ON/OFF

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

8.5.4 Other Reading

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

8.6 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
```

8.6.1 Usage

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.

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.

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.

Stored procedures

Stored procedure calls count as single query.

Collect time point

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

8.6.2 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

8.6.3 Version Specific Information

- *5.1.49-rel12.0*: Full functionality available.
- *5.1.53-12.4*: Introduced *have_response_time_distribution*.

8.6.4 System Variables

variable `have_response_time_distribution`

Version Info

- ***5.1.53-12.4*** – 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 `enable_query_response_time_stats`

Version Info

- ***5.1.49-rel12.0*** – 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`.

8.6.5 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

8.7 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        |
...
+------------+---------+----------------------------------------------------------------
↪--------------+--------------+------+------------+
```

8.7.1 Version-Specific Information

- *5.1.49-rel12.0*: Full functionality available.

8.7.2 Other Information

- Author / Origin: Percona

8.8 Show Lock Names

This feature is currently undocumented except for the following example.

Example:

```
mysql> SHOW ENGINE INNODB MUTEX;
+--------+---------------------------+---------------+
| Type   | Name                      | Status        |
+--------+---------------------------+---------------+
| InnoDB | &rseg->mutex              | os_waits=210  |
| InnoDB | &dict_sys->mutex          | os_waits=3    |
| InnoDB | &trx_doublewrite->mutex   | os_waits=1    |
| InnoDB | &log_sys->mutex           | os_waits=1197 |
| InnoDB | &LRU_list_mutex           | os_waits=2    |
| InnoDB | &fil_system->mutex        | os_waits=5    |
| InnoDB | &kernel_mutex             | os_waits=242  |
| InnoDB | &new_index->lock          | os_waits=2    |
| InnoDB | &new_index->lock          | os_waits=415  |
.....
```

8.9 Process List

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

8.9.1 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.

8.9.2 Example Output

`SHOW PROCESSLIST` Command:

```
mysql> show processlist;
+------+-----------+-----------+--------+---------+------+------------+---------------
↪------------------------------+
| Id   | User      | Host      | db     | Command | Time | State      | Info          ␣
↪                              |
+------+-----------+-----------+--------+---------+------+------------+---------------
↪------------------------------+
|    2 | root      | localhost | test   | Query   |    0 | NULL       | SHOW␣
↪PROCESSLIST                                 |
|   14 | root      | localhost | test   | Query   |    0 | User lock  | SELECT GET_
↪LOCK(``t``,1000)                          |
+------+-----------+-----------+--------+---------+------+------------+---------------
↪------------------------------+
```

Table *PROCESSLIST*:

```
mysql> select * from information_schema.PROCESSLIST;
+------+-----------+-----------+--------+---------+------+------------+---------------
↪------------------------------+
| ID   | USER      | HOST      | DB     | COMMAND | TIME | STATE      | INFO          ␣
↪                              |
+------+-----------+-----------+--------+---------+------+------------+---------------
↪------------------------------+
|   14 | root      | localhost | test   | Query   |    0 | User lock  | SELECT GET_
↪LOCK(``t``,1000)                          |
|    2 | root      | localhost | test   | Query   |    0 | executing  | SELECT * from␣
↪INFORMATION_SCHEMA.PROCESSLIST |
+------+-----------+-----------+--------+---------+------+------------+---------------
↪------------------------------+
```

8.10 Misc. INFORMATION_SCHEMA Tables

8.10.1 Temporary tables

Only temporary tables that were explicitly created with *CREATE TEMPORARY TABLE* are shown, and not the ones created during query execution. The temporary tables that are created for *ALTER TABLE* execution are not listed in *INFORMATION_SCHEMA.TEMPORARY_TABLES* or *GLOBAL_TEMPORARY_TABLES* tables.

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.

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.

8.10.2 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, it 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 this 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, it 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, it 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.

CHAPTER

NINE

OBSOLETE AND REMOVED FEATURES

9.1 Multiple Rollback Segments

To provide consistent reads, *InnoDB* writes data modified by active transactions in an area called the rollback segment. This single segment is protected by a single mutex, which is a major cause of contention for write-intensive workloads. With this change you can increase the number of rollback segments, which is likely to help you to greatly improve performance. You can see this post for a benchmark.

Some write-intensive workloads on boxes with many CPUs have scalability problems. The contention is caused by the rollback segment, which is single: all transactions are serialized when needing to access the segment. With this feature you can now create and use multiple segments (up to 256).

Note: This feature is incompatible with *InnoDB*. As long as a single rollback segment is used, there is no problem; the database can still be used by both *XtraDB* and *InnoDB*. However, creating multiple rollback segments will cause an internal format change to the system tablespace. Once multiple segments have been created, the database will no longer be compatible with *InnoDB*.

9.1.1 System Variables

The following system variable was introduced by this feature:

variable `innodb_extra_rsegments`

Command Line	Yes
Config File	Yes
Scope	Global
Dynamic	No
Type	ULONG
Def	0
Range	0-126

This option specifies the number of extra user rollback segments.

When you modify this variable, you must restart the *MySQL* server for the change to take effect. Please note that you must perform a slow shutdown (ie with `innodb_fast_shutdown = 0`). If you just perform a fast shutdown, the *MySQL* server will then restart without error but the additional segments will not be created.

If there is already any existing data stored in *XtraDB* or *InnoDB* it will need to be dumped and re-imported once this option has been enabled. This is needed so *XtraDB* can re-create the new data files with additional rollback segments.

To check that the extra segments have been created, you can run the following query:

```
SELECT COUNT(*) FROM information_schema.INNODB_RSEG;
```

The result should be the number of extra segments + 1 (as a default single segment always exists).

9.1.2 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)
```

9.1.3 Other Reading

- Fix of InnoDB/XtraDB scalability of rollback segment
- Tuning for heavy writing workloads

9.2 Remove Excessive Function Calls (`fcntl`)

Warning: This feature has been deprecated in *Percona Server* `5.1.66-14.1`.

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.

9.2.1 Version Specific Information

- `5.1.49-rel12.0`: Feature implemented
- `5.1.55-12.6`: Feature updated
- `5.1.66-14.1`: Feature deprecated
- `5.1.66-14.2`: Feature removed

9.2.2 Other Reading

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

9.3 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*.

9.3.1 Version Specific Information

- *5.1.49-rel12.0* Feature introduced.
- *5.1.50-rel12.1* System variable *innodb_buffer_pool_shm_checksum* added.
- *5.1.58-12.9* Feature removed, as LRU Dump/Restore is less invasive, more reliable and a better solution.

9.3.2 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

TEN

REFERENCE

10.1 Development of Percona Server

Percona Server is an open source project to produce a distribution of the *MySQL* server with improved performance, scalability and diagnostics.

10.1.1 Submitting Changes

This process is very much modeled on what is being used by Drizzle. The Drizzle project went through several iterations and refinements before settling on this process. It has been found to both keep trunk in a constant state of stability (allowing for a release at any time) and minimizing wasted time by developers due to broken code from somebody else interfering with their day.

You should also be familiar with our *Jenkins* setup.

Overview

At Percona we use Bazaar for source control and launchpad for both code hosting and release management.

Changes to our software projects could be because of a new feature (blueprint) or fixing a bug (bug). Projects such as refactoring could be classed as a blueprint or a bug depending on the scope of the work.

Blueprints and bugs are targeted to specific milestones (releases). A milestone is part of a series - e.g. 1.6 is a series in Percona XtraBackup and 1.6.1, 1.6.2 and 1.6.3 are milestones in the 1.6 series.

Code is proposed for merging in the form of merge requests on launchpad.

Some software (such as Percona Xtrabackup) we maintain both a development branch and a stable branch. For example: Xtrabackup 1.6 is the current stable series, and changes that should make it into bugfix releases of 1.6 should be proposed for the 1.6 tree. However, most new features or more invasive (or smaller) bug fixes should be targeted to the next release, currently 1.7. If submitting something to 1.6, you should also propose a branch that has these changes merged to the development release (1.7). This way somebody else doesn't have to attempt to merge your code and we get to run any extra tests that may be in the tree (and check compatibility with all platforms).

For Percona Server, we have two current 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.

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.

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.

Resubmitting a merge request

In the event of a merge request being marked as 'Work In Progress' due to build/test failures when merging, the developer should fix up the branch, run through param build and then 'Resubmit' the merge proposal.

There is a link on launchpad to resubmit the merge proposal, this means it appears in the list of merge requests to review again rather than off in the "work in progress" section.

Percona Server

The same process for Percona Server, but we have different branches (and merge requests) for 5.1 and 5.5 series.

Upgrading MySQL base version

- Same process as other modifications.
- create local branch
- make changes
- param build
- merge request

We will need some human processes to ensure that we do not merge extra things during the time when base MySQL version is being updated to avoid making life harder for the person doing the update.

10.1.2 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.

10.1.3 Jenkins

Our Jenkins instance uses a mixture of VMs on physical hosts that Percona runs and Virtual Machines in Amazon EC2 that are launched on demand.

Basic Concepts

We have some jobs that are activated based on source control changes (new commits in a 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.

Percona Xtrabackup

http://jenkins.percona.com/view/XtraBackup/

We currently build both xtrabackup 1.6 and xtrabackup trunk (will become 1.7).

There are param-builds for 1.6 and trunk too. These should be run for each merge request (and before any collection of merged branches is pushed to trunk)

Percona Server

We have separate jobs for Percona Server 5.1 and Percona Server 5.5 due to the different build systems that MySQL 5.1 and 5.5 use.

The `mysql-test-run.pl` test suite is integrated with Jenkins through subunit and `subunit2junitxml` allowing us to easily see which tests passed/failed on any particular test run.

Percona Server 5.1

http://jenkins.percona.com/view/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.

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).

10.2 Trademark Policy

This Trademark Policy is to ensure that users of Percona-branded products or services know that what they receive has really been developed, approved, tested and maintained by Percona. Trademarks help to prevent confusion in the marketplace, by distinguishing one company's or person's products and services from another's.

Percona owns a number of marks, including but not limited to Percona, XtraDB, Percona XtraDB, XtraBackup, Percona XtraBackup, Percona Server, and Percona Live, plus the distinctive visual icons and logos associated with these marks. Both the unregistered and registered marks of Percona are protected.

Use of any Percona trademark in the name, URL, or other identifying characteristic of any product, service, website, or other use is not permitted without Percona's written permission with the following three limited exceptions.

First, you may use the appropriate Percona mark when making a nominative fair use reference to a bona fide Percona product.

Second, when Percona has released a product under a version of the GNU General Public License ("GPL"), you may use the appropriate Percona mark when distributing a verbatim copy of that product in accordance with the terms and conditions of the GPL.

Third, you may use the appropriate Percona mark to refer to a distribution of GPL-released Percona software that has been modified with minor changes for the sole purpose of allowing the software to operate on an operating system or hardware platform for which Percona has not yet released the software, provided that those third party changes do not affect the behavior, functionality, features, design or performance of the software. Users who acquire this Percona-branded software receive substantially exact implementations of the Percona software.

Percona reserves the right to revoke this authorization at any time in its sole discretion. For example, if Percona believes that your modification is beyond the scope of the limited license granted in this Policy or that your use of the Percona mark is detrimental to Percona, Percona will revoke this authorization. Upon revocation, you must immediately cease using the applicable Percona mark. If you do not immediately cease using the Percona mark upon revocation, Percona may take action to protect its rights and interests in the Percona mark. Percona does not grant any license to use any Percona mark for any other modified versions of Percona software; such use will require our prior written permission.

Neither trademark law nor any of the exceptions set forth in this Trademark Policy permit you to truncate, modify or otherwise use any Percona mark as part of your own brand. For example, if XYZ creates a modified version of the Percona Server, XYZ may not brand that modification as “XYZ Percona Server” or “Percona XYZ Server”, even if that modification otherwise complies with the third exception noted above.

In all cases, you must comply with applicable law, the underlying license, and this Trademark Policy, as amended from time to time. For instance, any mention of Percona trademarks should include the full trademarked name, with proper spelling and capitalization, along with attribution of ownership to Percona Inc. For example, the full proper name for XtraBackup is Percona XtraBackup. However, it is acceptable to omit the word “Percona” for brevity on the second and subsequent uses, where such omission does not cause confusion.

In the event of doubt as to any of the conditions or exceptions outlined in this Trademark Policy, please contact trademarks@percona.com for assistance and we will do our very best to be helpful.

10.3 List of upstream *MySQL* bugs fixed in *Percona Server* 5.1

Upstream bug #54430 - innodb should retry partial reads/writes where errno was 0 **Launchpad bug** #1262500 **Upstream state** Closed **Fix Released** `5.1.73-14.12` **Upstream fix** N/A
Upstream bug #71315 - MTR test case crashes the server **Launchpad bug** #1266980 **Upstream state** N/A **Fix Released** `5.1.73-14.12` **Upstream fix** N/A
Upstream bug #71250 - Bison 3 breaks mysql build **Launchpad bug** #1262439 **Upstream state** Closed **Fix Released** `5.1.73-14.12` **Upstream fix** N/A
Upstream bug #69407 - Build warnings with mysql **Launchpad bug** #1244154 **Upstream state** Duplicate **Fix Released** `5.1.73-14.11` **Upstream fix** N/A

Continued on next page

Table 10.1 – continued from previous page

Upstream bug #68909 - In my_MD5Final in mysys/md5.c, ctx is not properly zeroed as intended **Launchpad bug** #1244154 **Upstream state** Closed **Fix Released** *5.1.73-14.11* **Upstream fix** N/A
Upstream bug #68354 - Server crashes on update/join FEDERATED + local table when only 1 local... **Launchpad bug** #1182572 **Upstream state** N/A **Fix Released** *5.1.70-14.8* **Upstream fix** N/A
Upstream bug #69379 - MySQL clients return bogus errno for host-not-found errors on Ubuntu 13.04 **Launchpad bug** #1186690 **Upstream state** Verified (checked on 2014-07-29) **Fix Released** *5.1.69-14.7* **Upstream fix** N/A
Upstream bug #68116 - InnoDB monitor may hit an assertion error in buf_page_get_gen in debug ... **Launchpad bug** #1100178 **Upstream state** Verified (checked on 2014-07-29) **Fix Released** *5.1.67-14.4* **Upstream fix** N/A
Upstream bug #67983 - Memory leak on filtered slave **Launchpad bug** #1042946 **Upstream state** Closed **Fix Released** *5.1.67-14.4* **Upstream fix** N/A
Upstream bug #68045 - security vulnerability CVE-2012-5611 **Launchpad bug** #1083377 **Upstream state** N/A **Fix Released** *5.1.67-14.3* **Upstream fix** N/A
Upstream bug #71603 - file name is not escaped in binlog for LOAD DATA INFILE statement **Launchpad bug** #1277351 **Upstream state** N/A **Fix Released** *5.1.66-14.2* **Upstream fix** N/A

Continued on next page

Table 10.1 – continued from previous page

Upstream bug #66237 - Temporary files created by binary log cache are not purged after transa... **Launchpad bug** #1070856 **Upstream state** Closed **Fix Released** *5.1.66-14.2* **Upstream fix** N/A
Upstream bug #69124 - Incorrect truncation of long SET expression in LOAD DATA can cause SQL ... **Launchpad bug** #1175519 **Upstream state** N/A **Fix Released** *5.1.66-14.2* **Upstream fix** N/A
Upstream bug #69380 - Incomplete fix for security vulnerability CVE-2012-5611 **Launchpad bug** #1186748 **Upstream state** N/A **Fix Released** *5.1.66-14.2* **Upstream fix** N/A
Upstream bug #67685 - security vulnerability CVE-2012-5611 **Launchpad bug** #1083377 **Upstream state** N/A **Fix Released** *5.1.66-14.2* **Upstream fix** N/A
Upstream bug #66550 - security vulnerability CVE-2012-4414 **Launchpad bug** #1042517 **Upstream state** N/A **Fix Released** *5.1.66-14.2* **Upstream fix** N/A
Upstream bug #61180 - korr/store macros in my_global.h assume the argument to be a char ... **Launchpad bug** #1042517 **Upstream state** Closed **Fix Released** *5.1.66-14.1* **Upstream fix** N/A
Upstream bug #61178 - Incorrect implementation of intersect(ulonglong) in non-optimized Bitmap.. **Launchpad bug** #1042517 **Upstream state** Verified (checked on 2014-07-29) **Fix Released** *5.1.66-14.1* **Upstream fix** N/A

Continued on next page

Table 10.1 – continued from previous page

Upstream bug #54127 - mysqld segfaults when built using –with-max-indexes=128 **Launchpad bug** #1042517 **Upstream state** Closed **Fix Released** *5.1.66-14.1* **Upstream fix** N/A
Upstream bug #67177 - MySQL 5.1 is incompatible with automake 1.12 **Launchpad bug** #1064953 **Upstream state** Closed **Fix Released** *5.1.66-14.1* **Upstream fix** 5.1.69
Upstream bug #62856 - Check for "stack overrun" doesn't work with gcc-4.6, server crashes **Launchpad bug** #902472 **Upstream state** Closed **Fix Released** *5.1.66-14.1* **Upstream fix** 5.1.70
Upstream bug #61509 - mysqld (5.1.57) segfaults with gcc 4.6 **Launchpad bug** #902471 **Upstream state** Closed **Fix Released** *5.1.66-14.1* **Upstream fix** N/A
Upstream bug #66301 - INSERT ... ON DUPLICATE KEY UPDATE + innodb_autoinc_lock_mode=1 is broken **Launchpad bug** #1035225 **Upstream state** Closed **Fix Released** *5.1.65-14.0* **Upstream fix** N/A
Upstream bug #64469 - Deadlock or crash on concurrent TRUNCATE TABLE and SELECT * FROM I_S **Launchpad bug** #903617 **Upstream state** Can't repeat **Fix Released** *5.1.62-13.3* **Upstream fix** N/A
Upstream bug #64128 - InnoDB error in server log of innodb_bug34300 **Launchpad bug** #937859 **Upstream state** Closed **Fix Released** *5.1.62-13.3* **Upstream fix** 5.1.63

Continued on next page

Table 10.1 – continued from previous page

Upstream bug #49336 - mysqlbinlog does not accept input from stdin when stdin is a pipe **Launchpad bug** #933969 **Upstream state** Closed **Fix Released** *5.1.62-13.3* **Upstream fix** 5.1.66
Upstream bug #64127 - MTR –warnings option misses some of InnoDB errors and warnings **Launchpad bug** #937859 **Upstream state** Verified (checked on 2014-07-29) **Fix Released** *5.1.62-13.3* **Upstream fix** N/A
Upstream bug #62557 - SHOW SLAVE STATUS gives wrong output with master-master and using SET... **Launchpad bug** #860910 **Upstream state** Closed **Fix Released** *5.1.60-13.1* **Upstream fix** 5.1.66
Upstream bug #45702 - Impossible to specify myisam_sort_buffer > 4GB on 64 bit machines **Launchpad bug** #878404 **Upstream state** Closed **Fix Released** *5.1.60-13.1* **Upstream fix** N/A
Upstream bug #53761 - RANGE estimation for matched rows may be 200 times different **Launchpad bug** #832528 **Upstream state** Closed **Fix Released** *5.1.59-13.0* **Upstream fix** N/A
Upstream bug #62516 - Fast index creation does not update index statistics **Launchpad bug** #857590 **Upstream state** Verified (checked on 2014-07-29) **Fix Released** *5.1.59-13.0* **Upstream fix** N/A
Upstream bug #63451 - atomic/x86-gcc.h:make_atomic_cas_body64 potential miscompilation bug **Launchpad bug** #803865 **Upstream state** Closed **Fix Released** *5.1.58-12.9* **Upstream fix** N/A

Continued on next page

Table 10.1 – continued from previous page

Upstream bug #43593 - dump/backup/restore/upgrade tools fails because of utf8_general_ci **Launchpad bug** N/A **Upstream state** Closed **Fix Released** `5.1.58-12.9` **Upstream fix** 5.1.62
Upstream bug #51196 - Slave SQL: Got an error writing communication packets, Error_code: 1160 **Launchpad bug** #813587 **Upstream state** Closed **Fix Released** `5.1.58-12.9` **Upstream fix** 5.1.62
Upstream bug #71183 - os_file_fsync() should handle fsync() returning EINTR **Launchpad bug** #1262651 **Upstream state** Verified (checked on 2014-07-29) **Fix Released** `5.1.56-12.7` **Upstream fix** N/A
Upstream bug #56433 - Auto-extension of InnoDB files **Launchpad bug** none **Upstream state** Closed **Fix Released** `5.1.56-12.7` **Upstream fix** N/A
Upstream bug #51325 - Dropping an empty innodb table takes a long time with large buffer pool **Launchpad bug** none **Upstream state** Closed **Fix Released** `5.1.56-12.7` **Upstream fix** N/A
Upstream bug #47337 - innochecksum not built for –with-plugin-innodb_plugin –without-plugin... **Launchpad bug** #671764 **Upstream state** Closed **Fix Released** `5.1.53-12.4` **Upstream fix** 5.1.60
Upstream bug #48883 - Test “innodb_information_schema” takes fewer locks than expected **Launchpad bug** #677407 **Upstream state** Closed **Fix Released** `5.1.53-11.7` **Upstream fix** 5.1.52sp1

Continued on next page

Table 10.1 – continued from previous page

Upstream bug #38551 - RBR/MBR + Query Cache + "invalidating query cache entries (table)" **Launchpad bug** #609027 **Upstream state** Closed **Fix Released** *5.1.49-rel12.0* **Upstream fix** N/A
Upstream bug #54814 - make BUF_READ_AHEAD_AREA a constant **Launchpad bug** #609027 **Upstream state** Closed **Fix Released** *5.1.49-rel12.0* **Upstream fix** N/A
Upstream bug #55032 - Query cache sometime insert queries to cache, but doesn't find ... **Launchpad bug** none **Upstream state** Verified (checked on 2014-07-29) **Fix Released** *5.1.47-rel11.2* **Upstream fix** N/A
Upstream bug #53371 - Parent directory entry ("..") can be abused to bypass table level grants. **Launchpad bug** none **Upstream state** Closed **Fix Released** *1.0.6-rel10.2* **Upstream fix** 5.1.51
Upstream bug #53237 - mysql_list_fields/COM_FIELD_LIST stack smashing - remote execution... **Launchpad bug** none **Upstream state** Closed **Fix Released** *1.0.6-rel10.2* **Upstream fix** 5.1.51
Upstream bug #50974 - Server keeps receiving big (> max_allowed_packet) packets indefinitely **Launchpad bug** none **Upstream state** Closed **Fix Released** *1.0.6-rel10.2* **Upstream fix** 5.1.51
Upstream bug #53237 - mysql_list_fields/COM_FIELD_LIST stack smashing - remote execution ... **Launchpad bug** #580324 **Upstream state** Closed **Fix Released** 5.1.47-rel11.0 **Upstream fix** 5.1.49

Continued on next page

Table 10.1 – continued from previous page

Upstream bug #47621 - MySQL and InnoDB data dictionaries will become out of sync when renaming..   **Launchpad bug** #488315   **Upstream state** Closed   **Fix Released** *1.0.6-9*   **Upstream fix** 5.1.47
Upstream bug #47622 - the new index is added before the existing ones in MySQL, but after one...   **Launchpad bug** #488315   **Upstream state** Closed   **Fix Released** *1.0.6-9*   **Upstream fix** 5.1.51
Upstream bug #39793 - Foreign keys not constructed when column has a '#' in a comment or ...   **Launchpad bug** none   **Upstream state** Closed   **Fix Released** *1.0.3-7*   **Upstream fix** 5.1.47
Upstream bug #44140 - Insert buffer operation may destroy the page during its recovery process   **Launchpad bug** none   **Upstream state** Won't fix   **Fix Released** *1.0.3-7*   **Upstream fix** N/A
Upstream bug #42101 - Race condition in innodb_commit_concurrency   **Launchpad bug** none   **Upstream state** Closed   **Fix Released** *1.0.3-7*   **Upstream fix** 5.1.47
Upstream bug #20001 - Support for temp-tables in INFORMATION_SCHEMA   **Launchpad bug** none   **Upstream state** Verified (checked on 2014-07-29)   **Fix Released** *1.0.3-7*   **Upstream fix** N/A

10.4 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*
- *TEMPORARY_TABLES*
- *TABLE_STATISTICS*
- *THREAD_STATISTICS*
- *USER_STATISTICS*
- *INNODB_BUFFER_POOL_PAGES*
- *INNODB_BUFFER_POOL_PAGES_BLOB*
- *INNODB_BUFFER_POOL_PAGES_INDEX*
- *INNODB_RSEG*
- *INNODB_CHANGED_PAGES*
- *INNODB_INDEX_STATS*
- *INNODB_TABLE_STATS*
- *INNODB_SYS_TABLES*
- *INNODB_SYS_STATS*
- *INNODB_SYS_INDEXES*
- *XTRADB_ADMIN_COMMAND*

10.5 Frequently Asked Questions

10.5.1 Q: Will *Percona Server* with *XtraDB* invalidate our *MySQL* support?

A: We don't know the details of your support contract. You should check with your *Oracle* representative. We have heard anecdotal stories from *MySQL* Support team members that they have customers who use *Percona Server* with *XtraDB*, but you should not base your decision on that.

10.5.2 Q: Will we have to *GPL* our whole application if we use *Percona Server* with *XtraDB*?

A: This is a common misconception about the *GPL*. We suggest reading the *Free Software Foundation* 's excellent reference material on the GPL Version 2, which is the license that applies to *MySQL* and therefore to *Percona Server* with *XtraDB*. That document contains links to many other documents which should answer your questions. *Percona* is unable to give legal advice about the *GPL*.

10.5.3 Q: Do I need to install *Percona* client libraries?

A: No, you don't need to change anything on the clients. *Percona Server* is 100% compatible with all existing client libraries and connectors.

10.5.4 Q: When using the *Percona XtraBackup* to setup a replication slave on Debian based systems I'm getting: "ERROR 1045 (28000): Access denied for user 'debian-sys-maint'@'localhost' (using password: YES)"

A: In case you're using init script on Debian based system to start `mysqld`, be sure that the password for `debian-sys-maint` user has been updated and it's the same as that user's password from the server that the backup has been taken from. Password can be seen and updated in `/etc/mysql/debian.cnf`. For more information on how to set up a replication slave using *Percona XtraBackup* see this how-to.

10.6 Copyright and Licensing Information

10.6.1 Documentation Licensing

10.6.2 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.

The original SHOW PATCHES code is by Jeremy Cole.

Percona does not require copyright assignment.

See the COPYING files accompanying the software distribution.

10.7 Options that make XtraDB tablespaces not compatible with MySQL

10.7.1 Expanded undo slots

Enabling *`innodb_extra_undoslots`* breaks compatibility with other programs. Specifically, it makes the datafiles unusable for ibbackup or for a MySQL server that is not run with this option.

10.7.2 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 it will fail to start on data files created when `innodb_fast_checksums` was enabled.

10.7.3 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.

10.7.4 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 compatibility.

10.8 Percona Server 5.1 Release notes

10.8.1 *Percona Server* 5.1.73-14.12

Percona is glad to announce the release of *Percona Server* 5.1.73-14.12 on July 31st, 2014 (Downloads are available from Percona Server 5.1.73-14.12 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.73, this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.73-14.12 milestone at Launchpad.

Note: Packages for Debian Wheezy and Ubuntu 14.04 (trusty) are not available for this release due to conflict with newer `libmysqlclient18` packages available in those releases.

Bugs Fixed

Percona Server couldn't be built with *Bison* 3.0. Bug fixed #1262439 (upstream #71250).

Ignoring Query Cache Comments feature could cause server crash. Bug fixed #705688.

Database administrator password could be seen in plain text when `debconf-get-selections` was executed. Bug fixed #1018291.

If *XtraDB* variable `innodb_dict_size` was set, the server could attempt to remove a used index from the in-memory InnoDB data dictionary, resulting in a server crash. Bugs fixed #1250018 and #758788.

Ported a fix from *MySQL* 5.5 for upstream bug #71315 that could cause a server crash if a malformed `GROUP_CONCAT` function call was followed by another `GROUP_CONCAT` call. Bug fixed #1266980.

MTR tests from binary tarball didn't work out of the box. Bug fixed #1158036.

InnoDB did not handle the cases of asynchronous and synchronous I/O requests completing partially or being interrupted. Bugs fixed #1262500 (upstream #54430).

Percona Server version was reported incorrectly in *Debian/Ubuntu* packages. Bug fixed #1319670.

Percona Server source files were referencing *Maatkit* instead of *Percona Toolkit*. Bug fixed #1174779.

The *XtraDB* version number in `univ.i` was incorrect. Bug fixed #1277383.

Other bug fixes: #1272732, #1167486, and #1314568.

10.8.2 *Percona Server* 5.1.73-14.11

Percona is glad to announce the release of *Percona Server* 5.1.73-14.11 on December 20th, 2013 (Downloads are available from Percona Server 5.1.73-14.11 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.73, this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.73-14.11 milestone at Launchpad.

Bugs Fixed

`INSTALL PLUGIN` statement would crash server if *User Statistics* were enabled. Bug fixed #1011047.

Running top-level make would fail while running `autoreconf` in UDF subdirectory if the *Automake* version was 1.13. Bug fixed #1244110.

Server would fail to build with *GCC* 4.8 in build configurations that have the *MySQL* maintainer mode enabled. Bugs fixed #1244154 (upstream bug #68909) and #1186190 (upstream bug #69407).

`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*).

Other bug fixes: #1082333.

10.8.3 *Percona Server* 5.1.72-14.10

Percona is glad to announce the release of *Percona Server* 5.1.72-14.10 on October 28th, 2013 (Downloads are available from Percona Server 5.1.72-14.10 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.72, this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.72-14.10 milestone at Launchpad.

Bugs Fixed

Due to an incompatible upstream change that went in unnoticed, the page cleaner 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.

Percona Server could crash server could crash while accessing `BLOB` or `TEXT` columns in *InnoDB* tables if *Support for Fake Changes* was enabled. Bug fixed #1188168.

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.

Other bug fixes: bug fixed #1191589.

10.8.4 *Percona Server* 5.1.71-14.9

Percona is glad to announce the release of *Percona Server* 5.1.71-14.9 on August 27th, 2013 (Downloads are available from Percona Server 5.1.71-14.9 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.71, this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.71-14.9 milestone at Launchpad.

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.

The buffer pool mutex split patch implemented in *Percona Server* could cause server crash with an assertion error on read-write workloads with compressed tables in debug builds. Bug fixed #1103850.

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 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 bug fixes: bug fixed #1188162 and bug fixed #1203308.

10.8.5 *Percona Server* 5.1.70-14.8

Percona is glad to announce the release of *Percona Server* 5.1.70-14.8 on July 3rd, 2013 (Downloads are available from Percona Server 5.1.70-14.8 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.70, this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.70-14.8 milestone at Launchpad.

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.

When an upgrade was performed between major versions (e.g. by uninstalling a 5.1 RPM and then installing a 5.5 one), `mysql_install_db` was still called on the existing data directory which lead to re-creation of the `test` database. Bug fixed #1169522.

Fixed the upstream bug #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.

Other bug fixes: bug fixed #1191395.

10.8.6 *Percona Server* 5.1.69-14.7

Percona is glad to announce the release of *Percona Server* 5.1.69-14.7 on June 10th, 2013 (Downloads are available from Percona Server 5.1.69-14.7 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.69, this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.69-14.7 milestone at Launchpad.

Bug Fixes

In *Ubuntu* Precise `libmysqlclient18` package was chosen from the distribution's repository instead of Percona's which could lead to package conflicts. Bug fixed #1174271.

Fixed the RPM Percona-Server-shared-compat package naming issue that could lead to unresolved package dependencies when installing *Percona Server* 5.1. Bug fixed #893860.

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.

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.

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 RPM package dependencies for different major versions of *Percona Server*. Bug fixed #1167109.

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.

OpenSSL libraries were not found in 32-bit builds due to a typo. Bug fixed #1175447.

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.

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.

mysql_set_permission was failing on *Debian* due to missing libdbd-mysql-perl package. Fixed by adding the package dependency. Bug fixed #1003776.

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.

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.

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.

Fixed the upstream bug #69379 which caused *MySQL* clients to return bogus error number for host-not-found errors on *Ubuntu* 13.04. Bug fixed #1186690.

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.

The *INNODB_CHANGED_PAGES* table couldn't be queried if the log tracker wasn't running. Bug fixed #1185304.

Other bug fixes: bug fixed #1174346, bug fixed #1160951, bug fixed #1079688, bug fixed #1132412, bug fixed #1153651.

10.8.7 *Percona Server* 5.1.68-14.6

Percona is glad to announce the release of *Percona Server* 5.1.68-14.6 on April 19th, 2013 (Downloads are available from Percona Server 5.1.68-14.6 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.68, this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.68-14.6 milestone at Launchpad.

Bug Fixes

Fixed yum dependencies that were causing conflicts in `CentOS` 6.3 during installation. Bugs fixed #1031427 and #1051874.

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. Bugs fixed #1081016.

When **mysqldump** was used with `--innodb-optimize-keys` and `--no-data` options, all secondary key definitions would be lost. Bug fixed #989253.

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.

Fix for bug #1070856 introduced a regression in *Percona Server* *`5.1.66-14.2`* which could cause a server to hang when binary log is enabled. Bug fixed #1162085.

Percona Server would re-create the test database when using `rpm` on server upgrade, even if the database was previously removed. Bug fixed #710799.

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.

Percona Server was missing help texts in the *MySQL* client because the help tables were missing. Bug fixed #1041981.

Other bug fixes: bug fixed #1154962, bug fixed #1154959, bug fixed #1154957, bug fixed #1154954, bug fixed #1144059, bug fixed #1050536.

10.8.8 *Percona Server* 5.1.68-14.5

Percona is glad to announce the release of *Percona Server* 5.1.68-14.5 on March 15th, 2013 (Downloads are available from Percona Server 5.1.68-14.5 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.68, this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.68-14.5 milestone at Launchpad.

Bug Fixes

This release contains no futher bug fixes than what is included in the previous *Percona Server* release and MySQL 5.1.68.

10.8.9 *Percona Server* 5.1.67-14.4

Percona is glad to announce the release of *Percona Server* 5.1.67-14.4 on March 8th, 2013 (Downloads are available from Percona Server 5.1.67-14.4 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.67 this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.67-14.4 milestone at Launchpad.

New Features

New user statements have been introduced for handling the *XtraDB changed page tracking* log files.

Bug Fixes

Time in slow query log was displayed incorrectly when `slow_query_log_microseconds_timestamp` variable was set to microseconds. Bug fixed #887928 (*Laurynas Biveinis*).

Upstream bug #67983 was causing a memory leak on a filtered slave. Bug fixed #1042946 (*Sergei Glushchenko*).

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*).

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* `5.1.59-13.0`. Bug fixed #1100159 (*Laurynas Biveinis*).

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*).

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*).

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*).

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*).

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*).

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*).

Fixed the regression introduced with the fix for bug #1083058 which caused unnecessary mutex reacquisitions in adaptive flushing. Bug fixed #1117067 (*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 regular expressions used for filtering the *InnoDB* stats that were causing sporadic extraneous lines in `mysqldumpslow` output. Bug fixed #1097692 (*Laurynas Biveinis*).

Other bug fixes: bug fixed #1098436 (*Laurynas Biveinis*), bug fixed #1096904 (*Laurynas Biveinis*), bug fixed #1096899 (*Laurynas Biveinis*), bug fixed #1096895 (*Laurynas Biveinis*), bug fixed #1092142 (*Laurynas Biveinis*), bug fixed #1091712 (*Laurynas Biveinis*), bug fixed #1090874 (*Laurynas Biveinis*), bug fixed #1089961 (*Laurynas Biveinis*), bug fixed #1088954 (*Laurynas Biveinis*), bug fixed #1088867 (*Laurynas Biveinis*), bug fixed #1089265 (*Laurynas Biveinis*), bug fixed #1089031 (*Laurynas Biveinis*), bug fixed #1108874 (*Laurynas Biveinis*), bug fixed #1083669 (*Laurynas Biveinis*), bug fixed #1082437 (*Laurynas Biveinis*), bug fixed #909376 (*Laurynas Biveinis*).

10.8.10 *Percona Server* 5.1.67-14.3

Percona is glad to announce the release of *Percona Server* 5.1.67-14.3 on January 23rd, 2013 (Downloads are available from Percona Server 5.1.67-14.3 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.67, this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.67-14.3 milestone at Launchpad.

Bug Fixes

Fixed the upstream bug #68045 and ported a fix for the security vulnerability CVE-2012-4414 from the *Percona Server* `5.1.66-14.2`. 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*).

Other bug fixes: bug fixed #1087202 (*Vladislav Vaintroub*, *Laurynas Biveinis*) and bug fixed #1087218 (*Vladislav Vaintroub*, *Laurynas Biveinis*).

10.8.11 *Percona Server* 5.1.66-14.2

Percona is glad to announce the release of *Percona Server* 5.1.66-14.2 on January 15th, 2013 (Downloads are available from Percona Server 5.1.66-14.2 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.66, this release will include all the bug fixes in it. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.66-14.2 milestone at Launchpad.

New Features

Multiple bitmap file support for *XtraDB changed page tracking* has been implemented.

HandlerSocket has been updated to version 1.1.0 (rev. 83d8f3af176e1698acd9eb3ac5174700ace40fe0).

Fast InnoDB Checksum feature has now been deprecated.

Support for Fake Changes has been improved by fetching the sibling pages.

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 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*).

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*).

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*).

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*).

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*).

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*).

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*).

Slow Query Log 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 #901060 (*Laurynas Biveinis*), bug fixed #1071877 (*Laurynas Biveinis*), bug fixed #1090596 (*Stewart Smith*), bug fixed #1050466 (*Laurynas Biveinis*), bug fixed #890404 (*Laurynas Biveinis*), bug fixed #1061118 (*Hrvoje Matijakovic*).

10.8.12 *Percona Server* 5.1.66-14.1

Percona is glad to announce the release of *Percona Server* 5.1.66-14.1 on October 26th, 2012 (Downloads are available from Percona Server 5.1.66-14.1 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.66, including all the bug fixes in it, *Percona Server* 5.1.66-14.1 is now the current stable release in the 5.1 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.66-14.1 milestone at Launchpad.

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 (*Laurynas Biveinis*).

Fixed the upstream *MySQL* bug #61509. Crash in YaSSL was causing SSL tests failures on *Ubuntu Oneiric* hosts. Bug fixed #902471 (*Laurynas Biveinis*).

Fixed the upstream *MySQL* bug #62856 where check for "stack overrun" wouldn't work with gcc-4.6 and caused the server crash. Bug fixed #902472 (*Laurynas Biveinis*).

Resolved the *Ubuntu Percona Server* package conflicts with upstream packages. Bug fixed #907499 (*Ignacio Nin*).

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*).

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*).

Fix for bug #905334 regressed by adding debug-specific code with missing local variable that would break the debug build. Bug fixed #1046389 (*Laurynas Biveinis*).

Fix for bug #686534 caused a regression by mishandling LRU list mutex. Bug fixed #1053087 (*George Ormond Lorch III*).

Fixed the upstream *MySQL* bug #67177 *Percona Server* 5.1 was incompatible with *Automake* 1.12. Bug fixed #1064953 (*Alexey Kopytov*).

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*).

Cleaned up the test duplicates in regular atomic operation tests. Bug fixed #1039931 (*Laurynas Biveinis*).

10.8.13 *Percona Server* 5.1.65-14.0

Percona is glad to announce the release of *Percona Server* 5.1.65-14.0 on September 3rd, 2012 (Downloads are available from Percona Server 5.1.65-14.0 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.65, including all the bug fixes in it, *Percona Server* 5.1.65-14.0 is now the current stable release in the 5.1 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.65-14.0 milestone at Launchpad.

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.

New table *`INNODB_CHANGED_PAGES`* has been implemented. This table contains a list of modified pages from the modified page bitmap files produced by the log tracking thread.

HandlerSocket has been upgraded to version 1.1.0.

Bug Fixes

Loading LRU dump was preventing shutdown. Bug fixed #712055 (*George Ormond Lorch III*).

The kill idle transactions feature in *XtraDB* (if enabled) could sometimes cause the server to crash. Bug Fixed: #871722 (*Stewart Smith*)

Fixed server assertion error related to buffer pool, only visible in debug builds. Bug fixed #905334 (*Laurynas Biveinis*).

Querying I_S.GLOBAL_TEMPORARY_TABLES or TEMPORARY_TABLES would crash threads working with temporary tables. Bug fixed #951588 (*Laurynas Biveinis*).

A crash could leave behind an InnoDB temporary table with temporary indexes resulting in an unbootable server. Bug fixed #999147 (*Laurynas Biveinis*).

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*)

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 implemented. It now dumps to a temporary file first, and then renames it to the actual dump file. Bug fixed #686392 (*George Ormond Lorch III*).

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*).

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*).

A Server acting as a replication slave with the query cache enabled could crash with glibc detected memory corruption. Bug fixed #915814 (*George Ormond Lorch III*).

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*).

Error log messages are now more verbose for LRU dump. Bug fixed #713481 (*George Ormond Lorch III*).

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*).

Removed the unneeded lrusort.py script. The server now does this sorting automatically and has done for some time. Bug fixed #882653 (*Stewart Smith*).

Server started with `skip-innodb` crashes on *SELECT * FROM INNODB_TABLE_STATS* or *INNODB_INDEX_STATS*. Bug fixed #896439 (*Stewart Smith*).

Removed the INFORMATION_SCHEMA table INNODB_PATCHES (actually XTRADB_ENHANCEMENTS) as it was out of date and isn't in 5.5 or later either. Bug fixed #1009997 (*Stewart Smith*).

10.8.14 *Percona Server* 5.1.63-13.4

Percona is glad to announce the release of *Percona Server* 5.1.63-13.4 on May 24th, 2012 (Downloads are available from Percona Server 5.1.63-13.4 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.63, including all the bug fixes in it, *Percona Server* 5.1.63-13.4 is now the current stable release in the 5.1 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.63-13.4 milestone at Launchpad.

Bug Fixes

- Building Percona Server with the Clang compiler resulted in a compiler error. Bug fixed #997496 (*Alexey Kopytov*).

10.8.15 *Percona Server* 5.1.62-13.3

Percona is glad to announce the release of *Percona Server* 5.1.62-13.3 on April 24th, 2012 (Downloads are available from Percona Server 5.1.62-13.3 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.62, including all the bug fixes in it, *Percona Server* 5.1.62-13.3 is now the current stable release in the 5.1 series. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.1.62-13.3 milestone at Launchpad.

This release doesn't hold the bug fix for MySQL bug #48848 for 32-bit Percona Server release.

New Features

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.

Bug Fixes

- Dependency issues for libmysqlclient*-dev package(s) on Debian. Bug fixed #656933 (*Ignacio Nin*).
- Fixed MySQL bug #64469 that caused MySQL server hanging (sometimes crashing) when a TRUNCATE TABLE is done together with selecting data over same table from information_schema. Bug fixed #903617 (*Sergei Glushchenko*).
- Fixed MySQL bug #49336, mysqlbinlog couldn't handle stdin when "|" used. Bug fixed #933969 (*Sergei Glushchenko*).
- Fixed MySQL bug #64127, "mysql-test-run.pl" didn't expect InnoDB errors and warnings in the format "InnoDB: ERROR" and "InnoDB: WARNING" with all-uppercase "ERROR" and "WARNING". Bug fixed #937859 (*Laurynas Biveinis*).
- Dependency issue while installing libmysqlclient15-dev on Ubuntu systems. Bug fixed #803151 (*Ignacio Nin*).
- Removed auto-generated file in 5.1 branch. Bug fixed #917290 (*Ignacio Nin*).
- MySQL 5.0 does not flag BEGIN/SAVEPOINT/COMMIT/ROLLBACK statements in its binlogs with LOG_EVENT_SUPPRESS_USE_F like 5.1+ does. This causes unnecessary *use* statements around such statements when the binlog is dumped by mysqlbinlog. Fixed by always suppressing the output of *use* for these statements. Bug fixed #929521 (*Oleg Tsarev*).

10.8.16 *Percona Server* 5.1.61-13.2

Percona is glad to announce the release of *Percona Server* 5.1.61-13.2 on February 9th, 2012 (Downloads are available from Percona Server 5.1.61-13.2 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.61, including all the bug fixes in it, *Percona Server* 5.1.61-13.2 is now the current stable release in the 5.1 series. All of *Percona* 's software is open-source and free, all the details of the release can be found in the 5.1.61-13.2 milestone at Launchpad.

Bug Fixes

- *`innodb_fake_changes`* doesn't handle duplicate keys on REPLACE. In some cases it could cause infinite loop. Bug fixed: #898306 (*Mark Callaghan and Valentine Gostev*).
- Broken Makefile - space instead of tab was causing compile error. Bug fixed: #911223 (*Oleg Tsarev*).
- Unintentional change of innodb_version format in 5.1.60. Previously, it was reported as "1.0.x-x.x". In 5.1.60 it is reported as "1.0.17-rel13.1". Bug fixed: #917246 (*Alexey Kopytov*).
- Merging Kewpie into Percona Server tree. Bug fixed: #909431 (*Patrick Crews*).
- The statistics line is not being calculated properly because the regexp line does not work well with the Percona Server format. Bug fixed #856910 (*Oleg Tsarev*).
- ALTER TABLE ... IMPORT TABLESPACE is holding the data dictionary mutex during the entire import operation. This becomes a problem for innodb_expand_import, because that code scans the tablespace being imported, blocking all queries accessing any InnoDB tables during the scan. It does need to protect some data dictionary operations, but it is possible to release the mutex during the most expensive operation, i.e. the .ibd file scan. Bug fixed #901775 (*Alexey Kopytov*).
- Compiler warnings with GCC 4.6 on oneiric hosts. Bug fixed #902467 (*Laurynas Biveinis*).
- Percona Server 5.1.58 crashes on a specific query. Bug fixed #905711 (*Laurynas Biveinis*).
- Subunit.pm was working incorrectly with months. Bug fixed: #911237 (*Oleg Tsarev*).
- Removing some of the libraries from Percona-Server-shared-51 as they are already present in Percona-Server-server-51. Bug fixed: #924477 (*Ignacio Nin*).
- Resolved the dependency issue in Debian lenny dev package. Bug fixed: #656933 (*Ignacio Nin*).
- Resolved the dependency issues in Ubuntu 8.04.4 LTS (Hardy Heron) dev package. Bug fixed: #803151 (*Ignacio Nin*).

10.8.17 *Percona Server* 5.1.60-13.1

Percona is glad to announce the release of *Percona Server* 5.1.60-13.1 on December 16, 2011 (Downloads are available from Percona Server 5.1.60-13.1 downloads and from the Percona Software Repositories).

Based on MySQL 5.1.60, including all the bug fixes in it, *Percona Server* 5.1.60-13.1 is now the current stable release in the 5.1 series. All of *Percona* 's software is open-source and free, all the details of the release can be found in the 5.1.60-13.1 milestone at Launchpad.

Bug Fixes

- `SHOW SLAVE STATUS` could give incorrect output with master-master replication and using SET user variables. This could only occur with a sever having both master-master replication and `--log-slave-updates` enabled. This is also filed in MySQL bug tracker, but not fixed in upstream MySQL at the time of this Percona Server release. Bug Fixed: #860910 (*Alexey Kopytov*)

- MyISAM repair-by-sort buffer cannot be more than 4GB even on 64bit architectures. With this bug fix, both the server option `--myisam-sort-buffer-size` and the *myisamchk* `--sort-buffer-size` can be set to values over 4GB on 64bit systems. For users with large MyISAM tables, this could be a great improvement in *myisamchk*, CREATE INDEX and ALTER TABLE performance. Bug Fixed: #878404 (*Alexey Kopytov*)
- The atomic operations used in *Response Time Distribution* on 32bit systems could (in theory) be optimized incorrectly by the compiler. This has not been observed in the wild and may only be an issue with future compilers. With this bug fixed, we have corrected the inline assembly to always produce correct compiled code even if future compilers implement new optimizations. Bug Fixed: #878022 (*Laurynas Biveinis*)
- GCC 4.6 has expanded diagnostics and compiler warnings. We have audited and fixed these warnings for Percona Server 5.1, finding that the warnings were benign. Bug Fixed #878164 (*Laurynas Biveinis*)

10.8.18 *Percona Server* 5.1.59-13.0

Percona is glad to announce the release of *Percona Server* 5.1.59-13.0 on October 13, 2011 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.1.59, including all the bug fixes in it, *Percona Server* 5.1.59-13.0 is now the current stable release in the 5.1 series. All of *Percona* ‘s software is open-source and free, all the details of the release can be found in the 5.1.59-13.0 milestone at Launchpad.

New Features

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.

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.

Block Startup until LRU dump is loaded

Added a new boolean option, *`innodb-blocking-lru-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*).

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*).
- Backported fix for MySQL bug #53761 (Wrong estimate for `RANGE` query with compound indexes). Bug Fixed: #832528 (*Alexey Kopytov*).
- 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

Improvements and fixes on general distribution:

- #858467, #845019, (*Alexey Kopytov*).

Improvements and fixes on the *Percona Server* Test Suite:

- #862378, #862252, #860416, #838725, #760085, #870156, #794790 (*Oleg Tsarev*, *Alexey Kopytov*, *Valentine Gostev*).

10.8.19 *Percona Server* 5.1.58-12.9

Percona is glad to announce the release of *Percona Server* 5.1.58-12.9 on August 12, 2011 (Downloads are available here and from the Percona Software Repositories).

Based on MySQL 5.1.58, including all the bug fixes in it, *Percona Server* 5.1.58-12.9 is now the current stable release in the 5.1 series. All of Percona's software is open-source and free, all the details of the release can be found in the 5.1.58-12.9 milestone at Launchpad.

Highlights

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*).
- A remaining loop from an unimplemented feature degraded the performance when using compressed tables and has been removed (`buf_LRU_insert_zip_clean`). Bugs Fixed: #802825 / #61341 in *MySQL* (*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 introdu

- `utf8_general_ci` to `utf8_general50_ci`, and
- `ucs2_general_ci` to `ucs2_general50_ci`.

Blueprint: utf8-general50-ci-5.1 (*Alexey Kopytov*).

SHM Buffer Pool has been removed

The *Shared Memory Buffer Pool* has been removed due to being both invasive and not widely used.

Instead, we recommend using the much 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 *Shared Memory Buffer Pool* and for setting up LRU dump/restore *Dump/Restore of the Buffer Pool*.

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*).

- 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*).
- Uninitialized values in the *Slow Query Log* patch. Bug Fixed: #794774 (*Oleg Tsarev*).
- 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. Bugs Fixed: #745241 (*Alexey Kopytov*).
- As the option `ignore-builtin-innodb` is incompatible with *Percona Server* with *XtraDB*, the server will not start and print the corresponding error instead. Bug Fixed: #704216 (Laurynas Biveinis).
- Querying `INNODB_SYS_TABLES` after an `ALTER TABLE` statement leaded to a server crash. Bug Fixed: #627189 (*Yasufumi Kinoshita*).
- The 64-bit CAS implementation may lead to a server crash on IA32 systems. Bug Fixed: #803865 (Laurynas Biveinis).
- Using the `innodb_lazy_drop_table` option led to an assertion error when truncating a table in some scenarios. Bug Fixed: #798371 (*Yasufumi Kinoshita*).

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:
  - #806975, #790199, #782391, #802829, #700965, #794840, #766266, (*Alexey Kopytov*, *Oleg Tsarev*, Stewart Smith, Laurynas Biveinis)
- Improvements and fixes on the *Percona Server* Test Suite: #790199, #785566, #782391, #800559, #794790, #794780, #800035, #684250, #803140, #803137, #803124, #803110, #803100, #803093, #803088, #803076, #803071 (*Oleg Tsarev*, *Yasufumi Kinoshita*, Stewart Smith, *Alexey Kopytov*).

10.8.20 *Percona Server* 5.1.57-12.8

Percona is glad to announce the release of *Percona Server* 5.1.57-12.8 on June 8th, 2011 (Downloads are available here and from the Percona Software Repositories).

Percona Server 5.1.57-12.8 is now the current stable release in the 5.1 series. It is is based on *MySQL* 5.1.57.

Bug Fixes

- Fixed *InnoDB* I/O code so that the interrupted system calls are restarted if they are interrupted by a signal. *InnoDB* I/O code was not fully conforming to the standard on POSIX systems, causing a crash with an assertion failure when receiving a signal on `pwrite()`. Bug Fixed: LP #764395 / *MySQL* bug #60788 (*A. Kopytov*).
- The maximum value for `innodb_use_purge_threads` has been corrected to 32 (maximum number of parallel threads in a parallelized operation). The *Dedicated Purge Thread* patch accepted a value up to 64 for the `innodb_use_purge_thread` variable, leading to an assertion failure for greater than the actual maximum. Bug Fixed: #755017 (*L. Biveinis*).

- Increased the treshold of 600 seconds during the import, which was causing issues when importing big tables. Bug Fixed #684829 (*Y. Kinoshita*).
- The innodb_use_sys_stats_table feature could cause a crash under high concurrency Bug Fixed #791092 (*Y. Kinoshita*).

Other Changes

- `HandlerSocket`, a NoSQL plugin for *MySQL*, has been updated to the latest stable version as April 11th, 2011.
- The list of authors of the plugins used have been corrected. Bug Fixes: #723050 (*Y. Kinoshita*).

10.8.21 *Percona Server* 5.1.56-12.7

Percona is glad to announce the release of *Percona Server* 5.1.56-12.7 on April 18, 2011 (Downloads are available here and from the Percona Software Repositories).

Percona Server 5.1.56-12.7 is now the current stable release in the 5.1 series. It is is based on *MySQL* 5.1.56.

New Features

- Expanded the applicability of *InnoDB* *Fast Index Creation* to **`mysqldump`**, `ALTER TABLE` and `OPTIMIZE TABLE`. (*Alexey Kopytov*)

Variable Changes

- Variable *`innodb_stats_method`* has been implemented in the upstream *InnoDB*, with the same name and functionality that had previously existed only in *XtraDB*. (*Yasufumi Kinoshita*)

Other Changes

- Implemented support for variable *`innodb_stats_method`* being implemented in the upstream *InnoDB*, including adding a column to table `INNODB_SYS_STATS`. Bug fixed: #733317. (*Yasufumi Kinoshita*)
- Added `README-windows` with basic command to build under *Windows*, VC 2010 Express. (*Vadim Tkachenko*)
- Added helper script `README-Solaris` for building on *Solaris*. (*Vadim Tkachenko*)
- Changes were made to the post-install messages. (*Ignacio Nin*)

Bugs Fixed

- Bug #631667 - The *MySQL* binary is now built with readline support so that command line browsing is possible. (*Ignacio Nin*)
- Bug #716575 - When using the `innodb_import_table_from_xtrabackup` feature, importing `.ibd` files smaller than 1 MB could lead to a server crash. (*Yasufumi Kinoshita*)
- 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*)

- Bug #735423 - File ownerships and permissions are now changed after running `mysql_install_db` after installation or upgrade, in order to avoid "permission denied" error when starting **mysqld**. (*Aleksandr Kuzminsky*)
- *MySQL* bugs 56433 and 51325 - These *MySQL* bugs have been fixed in *Percona Server*. (*Yasufumi Kinoshita*)

10.8.22 *Percona Server* 5.1.55-12.6

Percona is glad to announce the release of *Percona Server* 5.1.55-12.6 on March 7th, 2011 (Downloads are available here and from the Percona Software Repositories).

Percona Server 5.1.55-12.6 is now the current stable release version. It is is based on *MySQL* 5.1.55.

Changes

- Fixed compiler warnings in both the core server and in *XtraDB*. (*Alexey Kopytov*, *Yasufumi Kinoshita*)

Bugs Fixed

- Bug #602047 - The `ROWS_READ` columns of `TABLE_STATISTICS` and `INDEX_STATISTICS` were not properly updated when a query involved index lookups on an *InnoDB* table. (*Yasufumi Kinoshita*)
- Bug #707742 - The server could crash when trying to import a table which had not been previously prepared using **xtrabackup --prepare --export**. Also, on servers with huge buffer pools, adding or removing an index even on an empty *InnoDB* table could take a long time due to excessive locking when *`innodb_dict_size_limit`* was non-zero. Locking was relaxed to alleviate this. (*Yasufumi Kinoshita*)
- 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*)

10.8.23 *Percona Server* 5.1.54-12.5

Percona is glad to announce the release of *Percona Server* 5.1.54-12.5 on January 11, 2011 (Downloads are available here and from the Percona Software Repositories).

Percona Server 5.1.54-12.5 is now the current stable release version.

Functionality Added or Changed

- *Percona Server* 5.1.54-12.5 is based on *MySQL* 5.1.54.
- New Features Added:
  - Added system variable *`innodb_log_block_size`* and new value "keep_average" to system variable, and *`innodb_adaptive_checkpoint`* in *Improved InnoDB I/O Scalability*. (*Yasufumi Kinoshita*)
  - Added new value "ALL_O_DIRECT" to system variable *`innodb_flush_method`* in *Improved InnoDB I/O Scalability*. (*Yasufumi Kinoshita*)
- Other Changes: None

Bugs Fixed

- Bug #689830 - The development environment tests of `show_slave_status_nolock` work only on statement-based replication. They were failing when row-based replication was attempted. A check is now made for the replication type to test. (*Oleg Tsarev*)
- Bugs #688643, #691234 - Boolean command line options and configuration variables in the `slow_extended` patch were not being processed properly. (*Oleg Tsarev*)
- Bug #692211 - Starting the server with a non-zero `innodb_auto_lru_dump` value could crash the server if the dump file did not exist. (*Alexey Kopytov*)

10.8.24 *Percona Server* 5.1.53-12.4

Percona is glad to announce the release of *Percona Server* 5.1.53-12.4 on December 29, 2010 (Downloads are available here and from the Percona Software Repositories).

Percona Server 5.1.53-12.4 is now the current stable release version.

Functionality Added or Changed

- *Percona Server* 5.1.53-12.4 is based on *MySQL* 5.1.53.
- New Features Added:
  - Precompiled UDFs for Maatkit (`FNV` and `MurmurHash` hash functions to provide faster checksums) are now included in distributions. Fixes feature request #689992. (*Aleksandr Kuzminsky*)
- Other Changes:
  - `percona_innodb_doublewrite_path` - It's no longer necessary to recreate your database and *InnoDB* system files when a dedicated file to contain the doublewrite buffer is specified. (*Yasufumi Kinoshita*)
  - Added system variable *`have_response_time_distribution`* and compile option `--without-response_time_distribution` in *Response Time Distribution*. (*Oleg Tsarev*)

Bugs Fixed

- Bug #643149 - Slow query log entries were not written in the usual parsing format. (*Alexey Kopytov*)
- Bug #671764 - `innochecksum` wasn't distributed with RPM and .DEB packages. (*Aleksandr Kuzminsky*)
- Bug #673426 - Use of these system variables as command-line options could cause a crash or undefined behavior: `log_slow_timestamp_every`, `log_slow_sp_statements`, `slow_query_log_microseconds_timestamp`, `use_global_long_query_time`. (*Oleg Tsarev*)
- Bug #673567 - Compiler could produce spurious warnings when building on non-Linux platforms. A check is now made to see if `clock_gettime()` is present in `librt` at the configure stage. If yes, `` -lrt`` is added to `LIBS`. (*Alexey Kopytov*)
- Bug #673929 - Query cache misses were being reported for some queries when hits were actually occurring.
- Bug #676146 - The development environment test of `log_slow_verbosity=innodb` on a slave for row-based replication was not working correctly. (*Oleg Tsarev*)
- Bug #676147, #676148 - The development environment tests of options `log_slow_slave_statements` and `use_global_long_query_time` work only on statement-based replication. They were failing when row-based replication was attempted. A check is now made for the replication type to test. (*Oleg Tsarev*)

- Bug #676158 - Setting the query cache size to 512M caused test failure on low memory systems (*Aleksandr Kuzminsky*)
- Bug #677407 - The `innodb_information_schema` test could fail sporadically due to flawed logic in the `INFORMATION_SCHEMA.INNODB_LOCKS` caching mechanism. (contributed by *Kristian Nielsen*) (*Alexey Kopytov*)
- Bug #681486 - A dependency between Debian packages `libmysqlclient16` and `percona-server-common` was removed. (*Aleksandr Kuzminsky*)
- Bug #693815 - The test `percona_innodb_buffer_pool_shm` was failing. It should be run with the `--big-test` option. As the buffer pool size used in the test is 128M, the shared memory segment should be increased appropriately in order to run the test successfully.
- Bug #693814, #693815, #693816, #693817, #693819 - Tests in the test environment were updated to reflect past `INFORMATION_SCHEMA` changes. (*Aleksandr Kuzminsky*)
- Bug #693818 - Warning and error messages for stored routines could incorrectly report row numbers due to a change in the `slow_extended` patch. (*Alexey Kopytov*)

10.8.25 *Percona Server* 5.1.53-11.7

Percona is glad to announce the release of *Percona Server* 5.1.53-11.7 on December 16, 2010 (Downloads are available here and from the Percona Software Repositories).

Functionality Added or Changed

- *Percona Server* 5.1.53-11.7 is based on *MySQL* 5.1.53.
- New Features Added: None
- Other Changes: None

Bugs Fixed

- Bug #643149 - Slow query log entries were not being done in the usual parsing format. (*Alexey Kopytov*)
- Bug #677407 - The `INNODB.innodb_information_schema` test could fail sporadically due to flawed logic in the `INFORMATION_SCHEMA.INNODB_LOCKS` caching mechanism. (*Alexey Kopytov*)

10.8.26 *Percona Server* 5.1.52-rel11.6

Percona is glad to announce the release of *Percona Server* 5.1.52-rel11.6 on November 23rd, 2010 (Downloads are available here and from the Percona Software Repositories).

Functionality Added or Changed

- *Percona Server* 5.1.52-rel11.6 is based on *MySQL* 5.1.52.
- New Features Added: None
- Other Changes: None

Bugs Fixed

- Bug #671764 - `innochecksum` wasn't distributed with RPM and .DEB packages (Aleksandr Kuzminsky)
- Bug #673426 - Use of some system variables as command-line options caused a crash or undefined behavior. (*Oleg Tsarev*)
- Bug #673929 - Query cache misses were being reported for some queries when hits were actually occurring.
- Bug #676146 - The development environment test of `log_slow_verbosity=innodb` on a slave for row-based replication was not working correctly. (*Oleg Tsarev*)
- Bug #676147 - The development environment test of option log_slow_slave_statements for row-based replication was not working correctly. (*Oleg Tsarev*)
- Bug #676148 - Similar to #676147. A check is now made for the replication type to test. (*Oleg Tsarev*)
- Bug #676158 - Setting the query cache size to 512M caused test failure on low memory systems (Aleksandr Kuzminsky)

10.8.27 *Percona Server* 5.1.52-12.3

Percona is glad to announce the release of *Percona Server* 5.1.52-12.3 on December 13th, 2010 (Downloads are available here and from the Percona Software Repositories).

Functionality Added or Changed

- *Percona Server* 5.1.52-12.3 is now based on *MySQL* 5.1.52.
- New Features Added:
  - *HandlerSocket plugin* support has been added to all `RPM` and Debian packages. This is an experimental feature. (*Inada Naoki*, DeNA Co., Ltd.)
  - *Lock-Free SHOW SLAVE STATUS* - Allows examination of slave status even when a lock prevents `SHOW SLAVE STATUS` from operating. (*Oleg Tsarev*)
  - *Fast Shutdown* - Sleeping threads could cause a delay of up to 10 seconds during *InnoDB* shutdown. The sleeping state of threads was made interruptible to avoid this. (contributed by *Kristian Nielsen*) (*Alexey Kopytov*)
- Other Changes: None

Bugs Fixed

- Bug #640576 - The false error "innodb_extra_undoslots option is disabled but it was enabled before" was sometimes reported after upgrading from *MySQL* 5.0. (*Yasufumi Kinoshita*)
- Bug #643463 - Shutting down *XtraDB* could take up to 10 seconds, even when there was no actual work to do. (Fix contributed by *Kristian Nielsen*) (*Alexey Kopytov*)
- Bug #663757 - On the FreeBSD platform, the "gcc atomic built-in" function isn't provided, so `response_time_distribution` now uses the native atomic API on FreeBSD. (*Oleg Tsarev*)
- Bug #673426 - Use of some system variables as command-line options caused a crash or undefined behavior. (*Oleg Tsarev*)
- Bug #673562 - Debug build was broken due a to failing compile-time assertion in `mysqld.cc`. (*Alexey Kopytov*)

- Bug #673567 - Compiler could produce spurious warnings when building on non-Linux platforms. A check is now made to see if `clock_gettime()` is present in `librt` at the configure stage. If yes, `` -lrt`` is added to `LIBS`. (*Alexey Kopytov*)
- Bug #673929 - Query cache misses were being reported for some queries when hits were actually occurring. (*Oleg Tsarev*)
- Bug #676146 - The development environment test of `log_slow_verbosity=innodb` on a slave for row-based replication was not working correctly. (*Oleg Tsarev*)
- Bug #676147 - The development environment test of option `log_slow_slave_statements` for row-based replication was not working correctly. (*Oleg Tsarev*)
- Bug #676148 - Similar to Bug #676147. A check is now made for the replication type to test. (*Oleg Tsarev*)

10.8.28 *Percona Server* 5.1.51-rel11.5

Percona is glad to announce the release of *Percona Server* 5.1.51-rel11.5 on October 28th, 2010 (Downloads are available here and from the Percona Software Repositories).

Functionality Added or Changed

- *Percona Server* 5.1.51-rel11.5 is now based on *MySQL* 5.1.51.
- New Features Added: None
- Other Changes: None

Bugs Fixed

- Bug #661354 - Fixed a problem compiling query_cache_with_comments for 5.1.51-rel11.5. (*Oleg Tsarev*)
- Bug #661844 - Fixed a problem with server variables failing test for 5.1.51-rel11.5. (*Oleg Tsarev*)

10.8.29 *Percona Server* 5.1.50-rel12.1

Percona is glad to announce the release of *Percona Server* 5.1.50-rel12.1 on September 23rd, 2010 (Downloads are available here and from the Percona Software Repositories).

Functionality Added or Changed

- *Percona Server* 5.1.50-rel12.1 RC is now based on *MySQL* 5.1.50.
- New Features Added:
  - *Dump/Restore of the Buffer Pool* - Implemented automatic dumping of the buffer pool at specified intervals.
  - *Shared Memory Buffer Pool* - Implemented option *`innodb_buffer_pool_shm_checksum`*; when enabled, shared memory buffer pool is checksum validated. Bugs fixed: #643724, #649408, and #649393. (*Yasufumi Kinoshita*)
  - *Expand Table Import* - Implemented more exact checking to avoid corruption of the `.ibd` file using *`innodb_expand_import`*. (*Yasufumi Kinoshita*)
  - *Handle BLOB End of Line* - Implemented a *MySQL* client option to handle end-of-line in BLOB fields differently. Bug fixed: #625066. (*Sasha Pachev*)

  - *Response Time Distribution* - Counts queries with very short execution times and groups them by time interval. (*Oleg Tsarev*)

Bugs Fixed

- Bug #608992 - Added symbolic info to *Percona Server* binary to help in debugging. (*Aleksandr Kuzminsky*)
- Bug #624362 - Corrected wording in an *InnoDB* error message regarding too many locks printed. (*Vadim Tkachenko*)
- Bug #625066 - Fixed a problem handling end-of-line in `BLOB` fields in the *MySQL* client. (*Sasha Pachev*)
- Bug #640924 - Fixed a crash caused by `innodb_doublewrite_file`. (*Yasufumi Kinoshita*)
- Bug #643650 - Speeded up *InnoDB* shutdown when using shared memory buffer pool. (*Yasufumi Kinoshita*)
- Bug #643724 - Fixed an *InnoDB* crash when shared memory buffer pool was enabled. (*Yasufumi Kinoshita*)
- Bug #649408 - Fixed a problem causing a crash on startup when using shared memory buffer pool. (*Yasufumi Kinoshita*)
- Bug #649393 - *InnoDB* now recreates the shared memory segment for the buffer pool automatically after a crash. (*Yasufumi Kinoshita*)
- Bug #649623 - Fixed an error when compiling *Percona Server* 5.1.50-rel12.1 on FreeBSD (*Oleg Tsarev*)
- Bug #650977 - Fixed failed tests. (*Oleg Tsarev*)

10.8.30 *Percona Server* 5.1.50-rel11.4

Percona is glad to announce the release of *Percona Server* 5.1.50-rel11.4 on September 24th, 2010 (Downloads are available here and from the Percona Software Repositories).

New features

.The primary purpose of this release is to update to the latest version of *MySQL* 5.1.

- *Percona Server* 5.1.50-rel11.4 is now based on *MySQL* 5.1.50.

Fixed bugs

- *InnoDB Statistics* - The `SYS_STATS` table is now used only for non-temporary tables.

10.8.31 *Percona Server* 5.1.49-rel12.0

Percona is glad to announce the release of *Percona Server* 5.1.49-rel12.0 on September 15th, 2010 (Downloads are available here and from the Percona Software Repositories).

New features

- *Percona Server* 5.1.49-rel12.0 is based on *MySQL* 5.1.49.
- New features added:
  - *Error Code Compatibility* - Implements error code compatibilities with *MySQL* 5.5. (*Oleg Tsarev*)

   - `query_cache_totally_disable` - Allows the user to disable use of the query cache. (*Oleg Tsarev*, backport from *MySQL* 5.5)
   - `show_engines` - Changes `SHOW STORAGE ENGINES` to report *XtraDB* when appropriate. (*Oleg Tsarev*)
   - *Remove Excessive Function Calls (fcntl)* - Removes excessive fcntl calls. (*Oleg Tsarev*, port from *FaceBook* tree)
   - *Prevent Caching to FlashCache* - Prevents blocks of data from being cached to FlashCache during a query. (*Oleg Tsarev*, port from *FaceBook* tree)
   - `status_wait_query_cache_mutex` - Provides a new thread state - "Waiting on query cache mutex". (*Oleg Tsarev*)
   - *Too Many Connections Warning* - Issues the warning "Too many connection" if log_warnings is enabled. (*Oleg Tsarev*)
   - *Response Time Distribution* - Counts queries with very short execution times and groups them by time interval. (*Oleg Tsarev*)
   - *Shared Memory Buffer Pool* - Allows storing the buffer pool in a shared memory segment between restarts of the server. (*Yasufumi Kinoshita*)
   - Option *Log All Client Commands (syslog)* was added to the *MySQL* client. If enabled, all commands run on the client are logged to syslog. (*Oleg Tsarev*)

- New variables introduced:
   - *Improved InnoDB I/O Scalability* - Implements a session-level version of the *MySQL* global system variable `innodb_flush_log_at_trx_commit`. (*Yasufumi Kinoshita*)
   - *Fast Index Creation* - Allows disabling of fast index creation. (*Yasufumi Kinoshita*)
   - *InnoDB Statistics*- If ON, the table's statistics are stored statically in the internal table `SYS_STATS`. The table is populated with the `ANALYZE TABLE` command. (*Yasufumi Kinoshita*)

Fixed bugs

- Bug #576041 - Fixes long stalls while accessing the `innodb_buffer_pool_pages_index` table on systems with a large number of tables.
- Bug #592007 - More strictly enforces the maximum purge delay defined by `innodb_max_purge_lag` by removing the requirement that purge operations be delayed if an old consistent read view exists that could see the rows to be purged.
- Bug #607449 - Fixes a crash during shutdown when `userstat_running=1`.
- Bug #612954 - Fixes a problem with `SHOW PROCESSLIST` displaying an incorrect time.
- Bug #610525 - Reduces the number of compile time errors when the server is rebuilt.
- Bug #569275 - Fixes a crash when *XtraDB* shuts down in "crash resistent mode".
- Bug #589484 - Adds reporting of the query cache mutex status in `SHOW PROCESSLIST`.
- Bug #606965 - Allows preventing data caching to flash storage during a query.
- Bug #606810 - Ports a fix from to remove excessive `fcntl` calls.
- Bug #609027 - Allows query cache use to be completely disabled
- Bug #600352 - Fixes `SHOW STORAGE ENGINES` to correctly report "Percona-XtraDB" rather than "InnoDB"

10.8.32 *Percona Server* 5.1.49-rel11.3

Percona is glad to announce the release of *Percona Server* 5.1.49-rel11.3 on September 7th, 2010 (Downloads are available here and from the Percona Software Repositories).

New features

- *Percona Server* 5.1.49-rel11.3 is based on *MySQL* 5.1.49.
- A new variable was introduced: *innodb_use_sys_stats_table*. If ON, the table's statistics are stored statically in the internal table SYS_STATS. The table is populated with the ANALYZE TABLE command. (*Yasufumi Kinoshita*)
- A new session variable was introduced: *innodb_flush_log_at_trx_commit_session*. (*Yasufumi Kinoshita*)

Fixed bugs

- Bug #576041 - Fixes long stalls while accessing the innodb_buffer_pool_pages_index table on systems with a large number of tables
- Bug #592007 - More strictly enforces the maximum purge delay defined by innodb_max_purge_lag by removing the requirement that purge operations be delayed if an old consistent read view exists that could see the rows to be purged.
- Bug #607449 - Fixes a crash during shutdown when userstat_running=1
- Bug #612954 - Fixes a problem with SHOW PROCESSLIST displaying an incorrect time
- Bug #610525 - Reduces the number of compile time errors when the server is rebuilt
- Bug #569275 - Fixes a crash when *XtraDB* shuts down in "crash resistent mode"

10.8.33 *Percona Server* 5.1.47-rel11.2

Percona is glad to announce the release of *Percona Server* 5.1.47-rel11.2 on July 12th, 2010 (Downloads are available here and from the Percona Software Repositories).

Fixed bugs

- `/*!...*/` processing (#603618)
- Insert space instead of empty (#603618)
- Query cache insert to cache, but not found on it (MySQL bug #55032)

10.8.34 *Percona Server* 5.1.47-rel11.1

Percona is glad to announce the release of *Percona Server* 5.1.47-rel11.1 on June 25th, 2010 (Downloads are available here and from the Percona Software Repositories).

New features

- *Percona Server* is now based on *MySQL* 5.1.47, and *XtraDB* is now based on *InnoDB* plugin 1.0.8.
- *XtraDB* now uses the fast recovery code released in *InnoDB* Plugin version 1.0.8, instead of Percona's earlier fast-recovery code.
- Added the `percona_innodb_doublewrite_path` place the double-write-buffer into its own file (issue #584299).
- Added the *`suppress_log_warning_1592`* option to disable logging of error code 1592.
- Added the `microseconds_in_slow_query_log` option to use microsecond precision for the slow query log's timestamps (issue #358412).
- Added the *`use_global_log_slow_control`* option to control slow-query logging globally without restarting, similar to *`use_global_long_query_time`*.
- Added the *`query_cache_strip_comments`* option to strip comments from query before using it in query cache.
- Added a global *`innodb_deadlocks`* counter to `SHOW STATUS`, based on a patch by Eric Bergen (issue #569288, issue #590624).
- Added more tests to the *MySQL* test framework.

Fixed bugs

- #598576 - query_cache_with_comments feature crash the server
- #573100 - Can't compile 5.1.46
- #573104 - separate name in `INNODB_SYS_TABLES` table
- #580324 - Security bug in upstream
- #586579 - *Percona Server* 11 fails to compile with `CFLAGS=-DUNIV_DEBUG`
- #569156 - CentOS 5: `mysql-server` conflicts with `mysql-server`
- #589639 - Recovery process may hang when tablespaces are deleted during the recovery
- #570840 - `deb` package conflicts with `libdbd-mysql-perl`

10.8.35 *Percona Server* 1.0.6-rel10.2

Percona is glad to announce the release of *Percona Server* 1.0.6-rel10.2 on May 25th 2010.

New Features:

- Ubuntu Lucid(x86_64) binaries

Fixed bugs:

- Fixed Bug #573100

10.8.36 *Percona Server* 1.0.6-rel10.1

Percona is glad to announce the release of *Percona Server* 1.0.6-rel10.1 on May 25th, 2010.

New Features:

- Variable `innodb_trx_id` is added to slow log statistics

Fixed bugs

- Fixed MySQL bug #53371 Parent directory entry ("..") can be abused to bypass table level grants.
- Fixed MySQL bug #53237 `mysql_list_fields/COM_FIELD_LIST` stack smashing - remote execution possible.
- Fixed MySQL bug #50974 Server keeps receiving big (`> max_allowed_packet`) packets indefinitely.

10.8.37 *Percona Server* 1.0.6-9

Percona is glad to announce the release of *Percona Server* 1.0.6-9.1 on January 13th, 2010.

The release includes following new features:

- The release is base on 1.0.6 version of *InnoDB* plugin.
- *MySQL* 5.1.42 as a base release
- Separate purge thread and LRU dump is implemented (this feature was actually added in Release 8, but somehow it was forgotten)
- New patch `innodb_relax_table_creation`
- Added extended statistics to slow log
- Adjust defaults with performance intention
- Added parameter to control checkpoint age
- Added recovery statistics output when crash recovery (disabled by default)
- Adjusted intraction of patches
- Patch to dump and restore `innodb_buffer_pool`

Fixed bugs:

- Bug #488315: rename columns and add unique index cause serious inconsistent with **`mysqld`**.
- Bug #495342: MySQL 5.1.41 + InnoDB 1.0.6 + XtraDB patches extensions-1.0.6, rev. 127 fails to compile on OpenSolaris.
- *MySQL* bug #47621: change the fix of http://bugs.mysql.com/47621 more reasonable.

10.8.38 *Percona Server* 1.0.6-9.1

Percona is glad to announce the release of *Percona Server* 1.0.6-9.1 on March 15th, 2010.

New features in version 9.1

- *MySQL* 5.1.43 is taken as the basis
- packages name changed to `Percona-XtraDB`
- Enabled support of SSL
- Enabled profiling
- Added script to sort LRU dump
- New supported platforts are added. The full list includes:
  - CentOS 5 (x86_64 and i386)
  - CenOS 4 (x86_64 and i386)
  - Debian lenny (x86_64 and i386)
  - Debian etch (x86_64 and i386)
  - Ubuntu Jaunty (x86_64 and i386)
  - Ubuntu Intrepid (x86_64 and i386)
  - Ubuntu Hardy (x86_64 and i386)
  - FreeBSD 8 (x86_64 and i386)
  - OpenSolaris (x86_64)

Fixed bugs

- Bug #506894: `buf_flush_LRU_recommendation()` is too optimistic
- Fixed *MySQL*-tests:
  - mysqk
  - mysql_upgrade
  - ssl tests
  - enabled `rpl_killed_ddl` and `innodb-autoinc` tests

10.8.39 *Percona Server* 1.0.6-10

Percona is glad to announce the release of *Percona Server* on April 13th, 2010.

- *MySQL* 5.1.45 as a basis for the release
- `I_S SYS_TABLES`
- New patch `page_size_and_expand_log`. It allows to change *InnoDB* pages size to 4K or 8K. Also *InnoDB* log files now can be bigger 4G
- New patch innodb-fast-checksum. It adds new faster checksum algorithm used in *InnoDB* tables
- New supported platforts are added. The full list includes (karmic is new):
  - CentOS 5 (x86_64 and i386)
  - CenOS 4 (x86_64 and i386)
  - Debian lenny (x86_64 and i386)

  - Debian etch (x86_64 and i386)
  - Ubuntu Karmic (x86_64 and i386)
  - Ubuntu Jaunty (x86_64 and i386)
  - Ubuntu Intrepid (x86_64 and i386)
  - Ubuntu Hardy (x86_64 and i386)
  - FreeBSD 8 (x86_64 and i386)
  - OpenSolaris (x86_64)

Fixed bugs in the release

- Bug #551379:Change SE Name at *MySQL* start
- Bug #550867: 5.1.45+|XtraDB|-10 compilation fails on Ubuntu Karmic
- Bug #547230: Add new columns to `innodb_table_stats`
- Bug #548442: `INFORMATION_SCHEMA.TABLE_STATISTICS` is broken in *XtraDB* 9.1
- Bug #509017: `srv_table_reserve_slot()` should be protected by `kernel_mutex`
- Bug #520825: `ANALYZE TABLE` doesn't anything when `innodb_stats_on_metadata=OFF`

10.8.40 *Percona Server* 1.0.4-8

Percona is glad to announce the release of *Percona Server* 1.0.4-8 on October 12th, 2009.

The release includes following new features:

- The release is base on 1.0.4 version of *InnoDB* plugin. We thank Schooner Information Technology for sponsoring and testing the port of *XtraDB* to 1.0.4
- *MySQL* 5.1.39 as a base release

Fixed bugs:

- #413858: Crash from failed assertion in dict0dict.c]]
- #417751: *XtraDB* crashes on startup on windows]]
- fix-import-extern-pages
- Number of *MySQL*-tests are fixed

10.8.41 *Percona Server* 1.0.3-7

Percona is glad to announce the release of *Percona Server* 1.0.3-7 on August 19th, 2009.

The release includes following new features:

- *MySQL* 5.1.37 as a base release
- speed hack for `buf_flush_insert_sorted_into_flush_list` controlled by the new variable `innodb_fast_recovery`

Fixed bugs

- MySQL bug #39793, Bugs #45357: 5.1.35 crashes with `Failing assertion: index->type & DICT_CLUSTERED`
- Bug #405714 in Percona-XtraDB: `During page flush it may be enqueued for flush again when innodb_flush_neighbours=0`
- Bug #395783 in Percona-XtraDB: `main.innodb_bug42101 fails on 5.1.36`
- This fixes also #45749 and #42101

10.8.42 *Percona Server* 1.0.3-6

Percona is glad to announce the release of *Percona Server* 1.0.3-6 on July 20th, 2009.

The release includes following new features:

- *MySQL* 5.1.36 as a base release
- *Fast InnoDB Recovery Process*
- Experimental adaptive checkpoint method estimate
- *InnoDB Statistics* - the implementation of the fix for *MySQL* Bug #30423
- *Expand Table Import* - Support of import *InnoDB* / *XtraDB* tables from another server
- New patch to split buffer pool mutex
- `g-style-io-thread`: Google's fixes to *InnoDB* IO threads
- `innodb_dict_size_limit` - Limit of internal data dictionary

Fixed bugs

- *MySQL* Bug: #39793: Foreign keys not constructed when column has a `#` in a comment or default value
- Bug #395784 in Percona-XtraDB: `main.innodb_xtradb_bug317074 internal check fails on 5.1.36`
- Bug #395778 in Percona-XtraDB: `main.innodb-analyze internal check failed on 5.1.36`
- Bug #391189 in Percona-XtraDB: `main.innodb_bug36172 mysq test fails with innodb_stat.patch applied`
- Bug #388884 in Percona-XtraDB: `patching fails on 5.1.35`

10.8.43 *Percona Server* 1.0.3-5

Percona is glad to announce the release of *Percona Server* 1.0.3-5 on April 27th, 2009.

- move to MySQL 5.1.34
- Removed `rw_lock patch`, so we now use Google's `rw_locks`
- Fix bug #358980: crash after upgrade 5.1.31 -> 5.1.33
- Fix *MySQL*-test

10.8.44 *Percona Server* 1.0.3-4 Sakura

UPDATE April 27, 2009: Unfortunately we found significant issues in this release, please use or upgrade to 1.0.3-5

Percona is glad to announce the release of *Percona Server* 1.0.3-4 on April 8th, 2009.

- move to MySQL 5.1.33.
- move to *InnoDB* Plugin 1.0.3.
- More reliable replication, `innodb_overwrite_relay_log_info` patch now included in the release.
- Bug #354302: Fix assertions for `UNIV_DEBUG`
- Bug #333750 new fix for `rw_lock` patch

10.8.45 *Percona Server* 1.0.2-3 Sapporo

Percona is glad to announce the release of *Percona Server* 1.0.2-3 on March 2nd, 2009.

- Move to *MySQL* 5.1.31
- Fixed #329290: crash in insert/delete load.
- Fixed #324734: fails to build with *MySQL* using `--with-debug`
- More safe `rw-lock`: fix #326445 - `XtraDB hangs in 10 threads TPC-C bug`
- Scalability fix – replaced `page_hash` mutex to `page_hash`, `read-write` lock
- fixed #317074: *InnoDB* compression hangs
- Fix broken group commit]] in *InnoDB*, add new global variable to *Improved InnoDB I/O Scalability*: `enable_unsafe_group_commit`
- Scalability fix – ability to use *Multiple Rollback Segments*
- Fixed #316995: crash when attempt to select from `information_schema` table
- Fixed #316449 compilation warning
- Fixed #317584: failed assertion
- Fixed #316982: compilation failed with `UNIV_DEBUG` enabled
- Changed parameter for control read ahead activity in *Improved InnoDB I/O Scalability* - now it accepts string values
- Fixed compiler warnings on `buf_blockt`
- Changed parameter name from `innodb_ibuf_positive_contract` to *Improved InnoDB I/O Scalability*: `innodb_ibuf_active_contract`.
- Added parameter to control merging insert buffer to *Improved InnoDB I/O Scalability*: *`innodb_ibuf_accel_rate`*
- Added parameter to control neighbor flushing to *Improved InnoDB I/O Scalability*: *`innodb_flush_neighbor_pages`*
- Added parameters to restrict insert buffer size to *Improved InnoDB I/O Scalability*: *`innodb_ibuf_max_size`*, *`innodb_ibuf_active_contract`*
- Stop `adaptive_checkpoint` flushing when exceeds `LOG_POOL_PREFLUSH_RATIO_ASYNC`

10.8.46 *Percona Server* 1.0.2-2

Percona is glad to announce the release of *Percona Server* 1.0.2-2 on December 28th, 2008.

- *Improved Buffer Pool Scalability*
- *More Concurrent Transactions Available*

10.8.47 *Percona Server* 1.0.2

Percona is glad to announce the release of *Percona Server* 1.0.2 on December 16th, 2008.

- `innodb_show_enhancements_page`
- `innodb_show_status`
- *Improved InnoDB I/O Scalability*
- `innodb_rw_lock`
- *Reduced Buffer Pool Mutex Contention*
- `innodb_buffer_pool_pages`

10.9 Glossary

ACID Set of properties that guarantee database transactions are processed reliably. Stands for: *Atomicity*, *Consistency*, *Isolation*, *Durability*.

Atomicity Atomicity means that database operations are applied following a "all or nothing" rule. A transaction is either fully applied or not at all.

Consistency Consistency means that each transaction that modifies the database takes it from one consistent state to another.

Drizzle Drizzle: a database for the cloud.

Drizzle is a community-driven open source project that is forked from the popular MySQL database. The Drizzle team has removed non-essential code, re-factored the remaining code into a plugin-based architecture and modernized the code base moving to C++.

Drizzle Charter:

- A database optimized for Cloud infrastructure and Web applications.
- Design for massive concurrency on modern multi-cpu architecture
- Optimize memory for increased performance and parallelism
- Open source, open community, open design Scope
- Re-designed modular architecture providing plugins with defined APIs
- Simple design for ease of use and administration
- Reliable, ACID transactional

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*, *Drizzle*) originally written by Innobase Oy, since acquired by Oracle. It provides *ACID* compliant storage engine with *foreign key* support. As of *MySQL* version 5.5, InnoDB became the default storage engine on all platforms.

Jenkins Jenkins is a continuous integration system that we use to help ensure the continued quality of the software we produce. It helps us achieve the aims of:

- no failed tests in trunk on any platform,
- aid developers in ensuring merge requests build and test on all platforms,
- no known performance regressions (without a damn good explanation).

LSN Log Serial Number. A term used in relation to the *InnoDB* or *XtraDB* storage engines.

MariaDB A fork of *MySQL* that is maintained primarily by Monty Program AB. It aims to add features, fix bugs while maintaining 100% backwards compatibility with MySQL.

my.cnf The file name of the default MySQL configuration file.

MyISAM A *MySQL Storage Engine* that was the default until MySQL 5.5.

MySQL An open source database that has spawned several distributions and forks. MySQL AB was the primary maintainer and distributor until bought by Sun Microsystems, which was then acquired by Oracle. As Oracle owns the MySQL trademark, the term MySQL is often used for the Oracle distribution of MySQL as distinct from the drop-in replacements such as *MariaDB* and *Percona Server*.

Percona Server Percona's branch of *MySQL* with performance and management improvements.

Storage Engine A *Storage Engine* is a piece of software that implements the details of data storage and retrieval for a database system. This term is primarily used within the *MySQL* ecosystem due to it being the first widely used relational database to have an abstraction layer around storage. It is analogous to a Virtual File System layer in an Operating System. A VFS layer allows an operating system to read and write multiple file systems (e.g. FAT, NTFS, XFS, ext3) and a Storage Engine layer allows a database server to access tables stored in different engines (e.g. *MyISAM*, InnoDB).

XtraDB Percona's improved version of *InnoDB* providing performance, features and reliability above what is shipped by Oracle in InnoDB.

- genindex
- modindex
- search

INDEX