PERCONA
www.percona.com

Percona XtraDB Cluster Documentation

Release 5.6.51-28.46

Mar 22, 2021

CONTENTS

Percona XtraDB Cluster is High Availability and Scalability solution for MySQL Users.

Percona XtraDB Cluster provides:

- Synchronous replication. Transaction either committed on all nodes or none.
- Multi-master replication. You can write to any node.
- Parallel applying events on slave. Real "parallel replication".
- Automatic node provisioning.
- Data consistency. No more unsynchronized slaves.

Percona XtraDB Cluster is fully compatible with *MySQL* or *Percona Server for MySQL* in the following meaning:

- Data compatibility. *Percona XtraDB Cluster* works with databases created in *MySQL* / *Percona Server for MySQL*.
- Application compatibility. There is no or minimal application changes required to start work with *Percona XtraDB Cluster*.

Part I

Introduction

CHAPTER

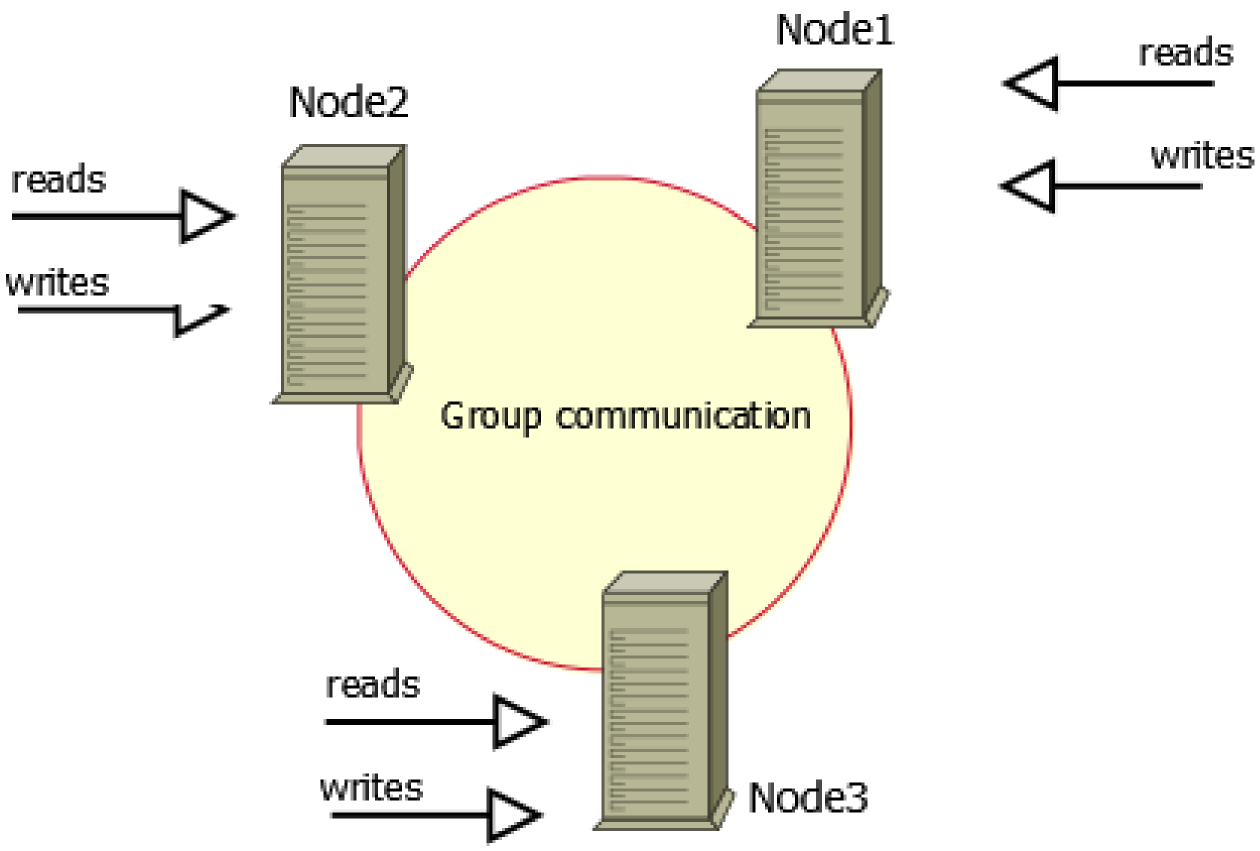
ONE

ABOUT PERCONA XTRADB CLUSTER

Percona XtraDB Cluster is open-source, free *MySQL* High Availability software

General introduction

1. The Cluster consists of Nodes. Recommended configuration is to have at least 3 nodes, but you can make it running with 2 nodes as well.
2. Each Node is regular *MySQL* / *Percona Server for MySQL* setup. The point is that you can convert your existing MySQL / Percona Server into Node and roll Cluster using it as a base. Or otherwise – you can detach Node from Cluster and use it as just a regular server.
3. Each Node contains the full copy of data.



Benefits and Drawbacks

Benefits:

- When you execute a query, it is executed locally on the node. All data is available locally, no need for remote access.
- No central management. You can loose any node at any point of time, and the cluster will continue to function.
- Good solution for scaling a read workload. You can put read queries to any of the nodes.

Drawbacks:

- Overhead of joining new node. The new node has to copy full dataset from one of existing nodes. If it is 100GB, it copies 100GB.
- This can't be used as an effective write scaling solution. There might be some improvements in write throughput when you run write traffic to 2 nodes vs all traffic to 1 node, but you can't expect a lot. All writes still have to go on all nodes.
- You have several duplicates of the data, for 3 nodes - 3 duplicates.

Components

Percona XtraDB Cluster is based on Percona Server with XtraDB and includes Write Set Replication patches. It uses the Galera library, version 3.x, a generic Synchronous Multi-Master replication plugin for transactional applications.

Galera library is developed by Codership Oy.

Galera 3.x supports such new features as:

- Incremental State Transfer (*IST*), especially useful for WAN deployments,
- RSU, Rolling Schema Update. Schema change does not block operations against table.

CHAPTER

TWO

PERCONA XTRADB CLUSTER LIMITATIONS

There are some limitations which you should be aware of. Some of them will be eliminated later as product is improved and some are design limitations.

- Currently replication works only with *InnoDB* storage engine. Any writes to tables of other types, including system (mysql.*) tables, are not replicated. However, `DDL` statements are replicated in statement level, and changes to mysql.* tables will get replicated that way. So, you can safely issue: `CREATE USER...`, but issuing: `INSERT INTO mysql.user...`, will not be replicated. You can enable experimental *MyISAM* replication support with `wsrep_replicate_myisam`.
- Unsupported queries:
  - `LOCK/UNLOCK TABLES` cannot be supported in multi-master setups.
  - lock functions (GET_LOCK(), RELEASE_LOCK()...)
- Query log cannot be directed to table. If you enable query logging, you must forward the log to a file: log_output = FILE. Use general_log and general_log_file to choose query logging and the log file name.
- Maximum allowed transaction size is defined by `wsrep_max_ws_rows` and `wsrep_max_ws_size` variables. `LOAD DATA INFILE` processing will commit every 10K rows. So large transactions due to `LOAD DATA` will be split to series of small transactions.
- Due to cluster level optimistic concurrency control, transaction issuing `COMMIT` may still be aborted at that stage. There can be two transactions writing to same rows and committing in separate *Percona XtraDB Cluster* nodes, and only one of the them can successfully commit. The failing one will be aborted. For cluster level aborts, *Percona XtraDB Cluster* gives back deadlock error code:

```
(Error: 1213 SQLSTATE: 40001  (ER_LOCK_DEADLOCK)).
```

- XA transactions can not be supported due to possible rollback on commit.
- The write throughput of the whole cluster is limited by weakest node. If one node becomes slow, whole cluster is slow. If you have requirements for stable high performance, then it should be supported by corresponding hardware.
- The minimal recommended size of cluster is 3 nodes. The 3rd node can be an arbitrator.
- Innodb fake changes feature is not supported.
- enforce_storage_engine=InnoDB is not compatible wsrep_replicate_myisam=OFF (default).
- `binlog_rows_query_log_events` variable not supported.
- Backup locks used during SST or with Xtrabackup can crash, on donor, either use inno-backup-opts='–no-backup-locks' under [sst] in my.cnf or set FORCE_FTWRL=1 in /etc/sysconfig/mysql (or /etc/sysconfig/mysql.%i for corresponding unit/service) for CentOS/RHEL or /etc/default/mysql in debian/ubuntu. You can also use rsync as the alternate SST method if those don't work.

- When running Percona XtraDB Cluster in cluster mode, avoid `ALTER TABLE ... IMPORT/EXPORT` workloads. It can lead to node inconsistency if not executed in sync on all nodes.

CHAPTER

THREE

PERCONA XTRADB CLUSTER ERRATA

Known Issues

Following are issues which may impact you while running PXC:

- Create Table As Select (CTAS) under highly concurent DDL workload (other than CTAS) may deadlock.
- For fresh installs, it is highly recommended to use the meta-packages to install packages. Refer to *Installing Percona XtraDB Cluster* guide for more details.

Also make sure to check limitations page *here*. You can also review this milestone for features/bugfixes to be included in future releases (i.e. releases after the upcoming/recent release).

Incompatibilities

Following are incompatibilities between PXC 5.5.33 (and higher) and older versions:

- wsrep_sst_donor now supports comma separated list of nodes as a consequence of bug #1129512. This only affects if you use this option as a list. This is not compatible with nodes running older PXC, hence **entire** cluster will be affected as far as SST is concerned, since donor nodes won't recognise the request from joiner nodes if former are not upgraded. Minimal workaround is to upgrade Galera package or to not use this new feature (wsrep_sst_donor with single node can still be used). However, upgrading the full PXC is strongly recommended, however, just upgrading PXC galera package will do for this.

Part II

Installation

CHAPTER

FOUR

INSTALLING PERCONA XTRADB CLUSTER

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

Prerequisites

Percona XtraDB Cluster requires the following ports for communication: 3306, 4444, 4567, 4568. Make sure that these ports are not blocked by firewall or used by other software.

Percona XtraDB Cluster does not work with `SELinux` and `AppArmor` security modules. Make sure these modules are disabled.

Installation Guides

It is recommended to install *Percona XtraDB Cluster* from official Percona repositories using the corresponding tool for your system:

- *Install using apt* if you are running Debian or Ubuntu
- *Install using yum* if you are running Red Hat Enterprise Linux or CentOS

Note: You can also download packages from the Percona website and install them manually using **`dpkg`** or **`rpm`**.

If you want to build and run *Percona XtraDB Cluster* from source, see *Compiling and Installing from Source Code*.

If you want to run *Percona XtraDB Cluster* using Docker, see *Running Percona XtraDB Cluster in a Docker Container*.

Installing *Percona XtraDB Cluster* on Debian and Ubuntu

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

The packages are available in the official Percona software repositories and on the download page. It is recommended to install *Percona XtraDB Cluster* from repositories using **`apt`**.

Installing from Repositories

Make sure to remove existing *Percona XtraDB Cluster* 5.5 packages and any *Percona Server for MySQL* packages before proceeding.

1. Configure Percona repositories as described in Percona Software Repositories Documentation.
2. Install the required *Percona XtraDB Cluster* package using **apt-get**. For example, to install the base package, run the following:

```
sudo apt-get install percona-xtradb-cluster-56
```

Note: Alternatively, you can install the `percona-xtradb-cluster-full-56` meta package, which contains the following additional packages:

- `percona-xtradb-cluster-5.6-dbg`
- `percona-xtradb-cluster-galera-3.x-dbg`
- `percona-xtradb-cluster-galera3-dbg`
- `percona-xtradb-cluster-garbd-3`
- `percona-xtradb-cluster-garbd-3.x`
- `percona-xtradb-cluster-garbd-3.x-dbg`
- `percona-xtradb-cluster-server-debug-5.6`
- `percona-xtradb-cluster-test-5.6`

For more information on how to bootstrap the cluster please check *Installing Percona XtraDB Cluster on Ubuntu*.

Note: Garbd is packaged separately as part of Debian split packaging. The `garbd` package in Debian is `percona-xtradb-cluster-garbd-3.x`. The package contains `garbd`, daemon init script and related config files. This package will be installed, if you install the `percona-xtradb-cluster-full-56` meta package.

Note: On *Debian Jessie (8.0)* and *Ubuntu Xenial (16.04)*, to stop or reload the node bootstrapped with `service mysql bootstrap-pxc`, you'll need to use `service mysql stop-bootstrap` or `service mysql reload-bootstrap` respectively. To check the status of the bootstrapped node, run `service mysql status-bootstrap`.

Installing *Percona XtraDB Cluster* on RHEL and CentOS

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

The packages are available in the official Percona software repositories and on the download page. It is recommended to install *Percona XtraDB Cluster* from repositories using **yum**.

Installing from Repositories

Make sure to remove existing *Percona XtraDB Cluster* 5.5 packages and any *Percona Server for MySQL* packages before proceeding.

1. Configure Percona repositories as described in Percona Software Repositories Documentation.
2. Install the required *Percona XtraDB Cluster* package using **yum**. For example, to install the base package, run the following:

   ```
   sudo yum install Percona-XtraDB-Cluster-56
   ```

   Note: Alternatively, you can install the `Percona-XtraDB-Cluster-full-56` meta package, which contains the following additional packages:

   - `Percona-XtraDB-Cluster-devel-56`
   - `Percona-XtraDB-Cluster-test-56`
   - `Percona-XtraDB-Cluster-debuginfo-56`
   - `Percona-XtraDB-Cluster-galera-3-debuginfo`
   - `Percona-XtraDB-Cluster-shared-56`

For more information on how to bootstrap the cluster please check *Installing Percona XtraDB Cluster on Ubuntu*.

Warning: *Percona XtraDB Cluster* requires the `socat` package, which can be installed from the EPEL repositories.

In CentOS, `mysql-libs` conflicts with the `Percona-XtraDB-Cluster-server-56` package. To avoid this, remove the `mysql-libs` package before installing *Percona XtraDB Cluster*. The `Percona-Server-shared-56` provides that dependency if required.

Compiling and Installing from Source Code

If you want to compile Percona XtraDB Cluster, you can find the source code on GitHub. Before you begin, make sure that the following packages are installed:

	apt	yum
Git	`git`	`git`
SCons	`scons`	`scons`
GCC	`gcc`	`gcc`
g++	`g++`	`gcc-c++`
OpenSSL	`openssl`	`openssl`
Check	`check`	`check`
CMake	`cmake`	`cmake`
Bison	`bison`	`bison`
Boost	`libboost-all-dev`	`boost-devel`
Asio	`libasio-dev`	`asio-devel`
Async I/O	`libaio-dev`	`libaio-devel`
ncurses	`libncurses5-dev`	`ncurses-devel`
Readline	`libreadline-dev`	`readline-devel`
PAM	`libpam-dev`	`pam-devel`

You will likely have all or most of the packages already installed. If you are not sure, run one of the following commands to install any missing dependencies:

```
$ sudo apt-get install -y git scons gcc g++ openssl check cmake bison \
libboost-all-dev libasio-dev libaio-dev libncurses5-dev libreadline-dev \
libpam-dev
```

```
$ sudo yum install -y git scons gcc gcc-c++ openssl check cmake bison \
boost-devel asio-devel libaio-devel ncurses-devel readline-devel pam-devel
```

To compile Percona XtraDB Cluster from source code:

1. Clone the Percona XtraDB Cluster repository:

   ```
   $ git clone https://github.com/percona/percona-xtradb-cluster.git
   ```

   Note: You have to clone the latest repository or update it to the latest state. Old codebase may not be compatible with the build script.

2. Clone Percona's fork of Galera into the same directory:

   ```
   $ cd percona-xtradb-cluster
   $ git clone https://github.com/percona/galera percona-xtradb-cluster-galera
   ```

   Note: The directory for Galera repository must be named `percona-xtradb-cluster-galera`.

3. Run the build script `./build-ps/build-binary.sh`. By default, it will build into the current directory, but you can specify another target output directory. For example, if you want to build into `./pxc-build`, run the following:

   ```
   $ mkdir ./pxc-build
   $ ./build-ps/build-binary.sh ./pxc-build
   ```

Running *Percona XtraDB Cluster* in a Docker Container

Docker images of *Percona XtraDB Cluster* are hosted publicly on Docker Hub at https://hub.docker.com/r/percona/percona-xtradb-cluster/.

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

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

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

The following procedure describes how to set up a simple 3-node cluster for evaluation and testing purposes, with all nodes running *Percona XtraDB Cluster* 5.6 in separate containers on one host:

1. Create a Docker network:

   ```
   docker network create pxc-network
   ```

2. Bootstrap the cluster (create the first node):

```
docker run -d \
  -e MYSQL_ROOT_PASSWORD=root \
  -e CLUSTER_NAME=cluster1 \
  --name=node1 \
  --net=pxc-network \
  percona/percona-xtradb-cluster:5.6
```

3. Join the second node:

```
docker run -d \
  -e MYSQL_ROOT_PASSWORD=root \
  -e CLUSTER_NAME=cluster1 \
  -e CLUSTER_JOIN=node1 \
  --name=node2 \
  --net=pxc-network \
  percona/percona-xtradb-cluster:5.6
```

4. Join the third node:

```
docker run -d \
  -e MYSQL_ROOT_PASSWORD=root \
  -e CLUSTER_NAME=cluster1 \
  -e CLUSTER_JOIN=node1 \
  --name=node3 \
  --net=pxc-network \
  percona/percona-xtradb-cluster:5.6
```

To ensure that the cluster is running:

1. Access the MySQL client. For example, on the first node:

```
$ sudo docker exec -it node1 /usr/bin/mysql -uroot -proot
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.6.37-82.2-56 Percona XtraDB Cluster (GPL), Release rel82.2,␣
↪Revision f3ba0dd, WSREP version 26.21, wsrep_26.21

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

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

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

mysql@node1>
```

2. View the wsrep status variables:

```
mysql@node1> show status like 'wsrep%';
+------------------------------+--------------------------------------------------+
| Variable_name                | Value                                            |
+------------------------------+--------------------------------------------------+
| wsrep_local_state_uuid       | 625318e2-9e1c-11e7-9d07-aee70d98d8ac             |
...
```

```
| wsrep_local_state_comment    | Synced                                       |
...
| wsrep_incoming_addresses     | 172.18.0.2:3306,172.18.0.3:3306,172.18.0.4:3306 |
...
| wsrep_cluster_conf_id        | 3                                            |
| wsrep_cluster_size           | 3                                            |
| wsrep_cluster_state_uuid     | 625318e2-9e1c-11e7-9d07-aee70d98d8ac         |
| wsrep_cluster_status         | Primary                                      |
| wsrep_connected              | ON                                           |
...
| wsrep_ready                  | ON                                           |
+------------------------------+----------------------------------------------+
59 rows in set (0.02 sec)
```

CHAPTER

FIVE

PERCONA XTRADB CLUSTER IN-PLACE UPGRADING GUIDE: FROM 5.5 TO 5.6

Upgrading cluster involves the following major stages:

1. Upgrade a single node.
2. Upgrade the whole cluster while not breaking replication.
3. [Optional] Restarting nodes with non-compat options.

Warning: Do not apply *SST* during upgrade. SST (State Snapshot Transfer) between nodes with **|percona-server|** or *MySQL* 5.5 and **|percona-server|** or *MySQL* 5.6 may not work as expected.

To avoid automatic SST for the duration of the upgrade, set the `wsrep-sst-method` variable to *skip* to ensure that you can handle SST manually when required. Consider setting Having a large enough the gcache size to a value large enough.

See also:

Percona Blog: How to calculate the correct size of Percona XtraDB Cluster's gcache https://www.percona.com/blog/2014/09/08/calculate-correct-size-percona-xtradb-clusters-gcache/

Following upgrade process is for a **rolling** upgrade, ie. an upgrade without downtime for the cluster. If you intend to allow for downtime - bring down all nodes, upgrade them, bootstrap and start nodes - then you can just follow Stage I sans the compatibility variables part. Make sure to bootstrap the first node in the cluster after upgrade.

Following is the upgrade process on CentOS 6.4

Step #1 Make sure all the nodes in cluster are upgraded to the latest 5.5 version and are in synced state.

Stage I

Assuming we are going to upgrade node A, (and other nodes B and C are on 5.5)

Step #2 On node A stop the mysql process and remove the old packages:

```
# service mysql stop
# yum remove 'Percona*'
```

Step #3 Install the new packages:

```
# yum install Percona-XtraDB-Cluster-56
```

Note: For more details on installation, refer to *Installing Percona XtraDB Cluster* guide. You may also want to install Percona-XtraDB-Cluster-full-56 which installs other ancillary packages like '-shared-56', '-test-56', debuginfos and so on.

Step #4 Fix the variables in the *MySQL* configuration file `my.cnf` which are not compatible with *Percona Server for MySQL* 5.6. Detailed list can be checked in Changed in Percona Server 5.6 documentation. In case you are not sure after this, you can also do following:

```
# mysqld --user=mysql --wsrep-provider='none'
```

If there are any invalid variables, it will print it there without affect galera grastate or any other things.

Note: It may also be worthwhile to backup the grastate.dat to use it if it gets zeroed (or sequence number to -1) accidentally (or due to network issues) since this can avoid SST.

Step #5 Add the following to `my.cnf` for compatibility with 5.5 replication for the duration of upgrade, and set the following options:

```
# Required for compatibility with galera-2
# Append socket.checksum=1 to other options if others are in wsrep_provider_options.␣
↪Eg.: "gmcast.listen_addr=tcp://127.0.0.1:15010; socket.checksum=1"
wsrep_provider_options="socket.checksum=1"
# Required for replication compatibility
log_bin_use_v1_row_events=1
avoid_temporal_upgrade=ON   # Available in 5.6.24 and up

gtid_mode=0
binlog_checksum=NONE
# Required under certain conditions
read_only=ON
```

Step #5.1 "read_only=ON" is required only when the tables you have contain timestamp/datetime/time data types as those data types are incompatible across replication from higher version to lower. This is currently a limitation of mysql itself. Also, refer to Replication compatibility guide. Any DDLs during migration are not recommended for the same reason.

Note: `read_only` does not apply to root connections (as per mysql specifications).

Step #5.2 To ensure 5.6 read-only nodes are not written to during migration, clustercheck (usually used with xinetd and HAProxy) distributed with PXC has been modified to return 503 when the node is read-only so that HAProxy doesn't send writes to it. Refer to clustercheck script for more details. Instead, you can also opt for read-write splitting at load-balancer/proxy level or at application level.

Note: On the last 5.5 node to upgrade to 5.6, the compatibility options of Step #5 are not required since all other nodes will already be upgrade and their compat. options are compatible with a 5.6 node without them.

Step #6 Next, start the node with the variable *`wsrep_provider`* set to `none`:

```
# mysqld --skip-grant-tables --user=mysql --wsrep-provider='none'
```

This is to ensure that other hosts are not affected by this upgrade (hence provider is none here).

Step #7 While Step #5 is running, in the background or in another session run:

```
# mysql_upgrade

Other options like socket, user, pass may need to provided here if not defined in my.
↪cnf.
```

Step #8 Step #7 must complete successfully, upon which, process started in Step #6 can be stopped/killed.

Step #9 If all the steps above have completed successfully node can be started with:

```
# service mysql start
```

Note: If this is the first node of cluster, then replace start with `bootstrap-pxc`. This shouldn't apply to rolling upgrade in general (since other nodes are up during this) but only for downtime-based upgrades (where you bring up nodes one by one).

Step #10 At this point, other nodes (B, C) should acknowledge that this node is up and synced!

Stage II

Step #11 After this has been set up all 5.5 nodes can be upgraded, one-by-one, as described in the Stage I.

1. If `read_only` was turned on in Step #5.1, then after all nodes in the cluster are upgraded to 5.6 or equivalently, after the last 5.5 has been take down for upgrade, option `read_only` can be set to `OFF` (since this is a dynamic variable, it can done without restart).
2. If read-write splitting was done in applications and/or in load-balancer then in previous step, instead of `read_only`, writes need to be directed to 5.6 nodes.

Stage III [Optional]

Step #12 This step is required to turn off the options added in #Step 5. Note, that this step is not required immediately after upgrade and can be done at a latter stage. The aim here is to turn off the compatibility options for performance reasons (only socket.checksum=1 fits this). This requires restart of each node. Hence, following can be removed/commented-out:

```
# Remove socket.checksum=1 from other options if others are in wsrep_provider_options.
↪ Eg.: "gmcast.listen_addr=tcp://127.0.0.1:15010"
# Removing this makes socket.checksum=2 which uses hardware accelerated CRC32␣
↪checksumming.
wsrep_provider_options="socket.checksum=1"

# Options added for replication compatibility, being removed here.
# You can keep some of these if you wish.

log_bin_use_v1_row_events=1
avoid_temporal_upgrade=ON    # Available in 5.6.24 and up

# You can keep if you are not adding async-slaves.
# Apropos, you may need to enable this if you are adding async-slaves, refer to MySQL␣
↪5.6 gtid_mode documentation for more details on this variable.
gtid_mode=0

# Galera already has full writeset checksumming, so
# you can keep this if async-slaves are not there and
# binlogging is not turned on.
binlog_checksum=NONE

# Remove it from cnf even though it was turned off at runtime in Step #11.
read_only=ON
```

Note: Making this stage without cluster downtime can be achieved by removing the config options and restarting node-by-node.

Following is the upgrade process on Ubuntu 12.04 (precise)

Step #1 Make sure all the nodes in cluster are upgraded to the latest 5.5 version and are in synced state.

Stage I

Assuming we are going to upgrade node A, (and other nodes B and C are on 5.5)

Step #2 On node A stop the mysql process and remove the old packages:

```
# /etc/init.d/mysql stop
# apt-get remove percona-xtradb-cluster-server-5.5 percona-xtradb-cluster-galera-2.x␣
↪percona-xtradb-cluster-common-5.5 percona-xtradb-cluster-client-5.5
```

Step #3 Fix the variables in the *MySQL* configuration file `my.cnf` which are not compatible with *Percona Server for MySQL* 5.6. Detailed list can be checked in Changed in Percona Server 5.6 documentation. Add the following to `my.cnf` for compatibility with 5.5 replication for the duration of upgrade, add 'socket.checksum=1' to the *`wsrep_provider_options`* variable and set *`wsrep_provider`* set to `none`

```
# Required for compatibility with galera-2
# Append socket.checksum=1 to other options if others are in wsrep_provider_options.␣
↪Eg.: "gmcast.listen_addr=tcp://127.0.0.1:15010; socket.checksum=1"
wsrep_provider_options="socket.checksum=1"
# Required for replication compatibility
log_bin_use_v1_row_events=1
avoid_temporal_upgrade=ON   # Available in 5.6.24 and up

gtid_mode=0
binlog_checksum=NONE
# Required under certain conditions
read_only=ON
wsrep_provider=none
```

Step #3.1 "read_only=ON" is required only when the tables you have contain timestamp/datetime/time data types as those data types are incompatible across replication from higher version to lower. This is currently a limitation of mysql itself. Also, refer to Replication compatibility guide. Any DDLs during migration are not recommended for the same reason.

Note: `read_only` does not apply to root connections (as per mysql specifications).

Step #3.2 To ensure 5.6 read-only nodes are not written to during migration, clustercheck (usually used with xinetd and HAProxy) distributed with PXC has been modified to return 503 when the node is read-only so that HAProxy doesn't send writes to it. Refer to clustercheck script for more details. Instead, you can also opt for read-write splitting at load-balancer/proxy level or at application level.

Note: It may also be worthwhile to backup the grastate.dat to use it if it gets zeroed (or sequence number to -1) accidentally (or due to network issues).

Note: On the last 5.5 node to upgrade to 5.6, the compatibility options of Step #3 are not required since all other nodes will already be upgrade and their configuration options are compatible with a 5.6 node without them.

Step #4 Install the new packages:

```
# apt-get install percona-xtradb-cluster-56
```

Note: For more details on installation, refer to *Installing Percona XtraDB Cluster* guide. You may also want to install percona-xtradb-cluster-full-56 which installs other ancillary packages like '-shared-56', '-test-56', debuginfos and so on.

Step #5 After node has been started you'll need to run `mysql_upgrade`:

```
# mysql_upgrade

Other options like socket, user, pass may need to provided here if not defined in my.
↪cnf.
```

Step #6 If all the steps above have completed successfully, shutdown the server, set the *`wsrep_provider`* to the location of the Galera library (from 'none' to something like /usr/lib/libgalera_smm.so) in my.cnf, and node can be started with:

```
# service mysql start
```

Note: If this is the first node of cluster, then replace start with `bootstrap-pxc`. This shouldn't apply to rolling upgrade in general (since other nodes are up during this) but only for downtime-based upgrades (where you bring up nodes one by one).

Step #7 At this point, other nodes (B, C) should acknowledge that this node is up and synced!

Stage II

Step #8 After this has been set up all 5.5 nodes can be upgraded, one-by-one, as described in the Stage I.

1. If `read_only` was turned on in Step #3.1, then after all nodes in the cluster are upgraded to 5.6 or equivalently, after the last 5.5 has been take down for upgrade, option `read_only` can be set to `OFF` (since this is a dynamic variable, it can done without restart).

2. If read-write splitting was done in applications and/or in load-balancer then in previous step, instead of `read_only`, writes need to be directed to 5.6 nodes.

Stage III [Optional]

Step #9 This step is required to turn off the options added in #Step 3. Note, that this step is not required immediately after upgrade and can be done at a latter stage. The aim here is to turn off the compatibility options for performance reasons (only socket.checksum=1 fits this). This requires restart of each node. Hence, following can be removed/commented-out:

```
# Remove socket.checksum=1 from other options if others are in wsrep_provider_options.
↪ Eg.: "gmcast.listen_addr=tcp://127.0.0.1:15010"
# Removing this makes socket.checksum=2 which uses hardware accelerated CRC32␣
↪checksumming.
wsrep_provider_options="socket.checksum=1"

# Options added for replication compatibility, being removed here.
# You can keep some of these if you wish.

log_bin_use_v1_row_events=1
avoid_temporal_upgrade=ON   # Available in 5.6.24 and up

# You can keep if you are not adding async-slaves.
# Apropos, you may need to enable this if you are adding async-slaves, refer to MySQL␣
↪5.6 gtid_mode documentation for more details on this variable.
gtid_mode=0

# Galera already has full writeset checksumming, so
# you can keep this if async-slaves are not there and
# binlogging is not turned on.
binlog_checksum=NONE

# Remove it from cnf even though it was turned off at runtime in Step #8.
read_only=ON
```

Note: Making this stage without cluster downtime can be achieved by removing the config options and restarting node-by-node.

Part III

Features

CHAPTER

SIX

HIGH AVAILABILITY

In a basic setup with 3 nodes, the *Percona XtraDB Cluster* will continue to function if you take any of the nodes down. At any point in time you can shutdown any Node to perform maintenance or make configuration changes. Even in unplanned situations like Node crash or if it becomes unavailable over the network, the Cluster will continue to work and you'll be able to run queries on working nodes.

In case there were changes to data while node was down, there are two options that Node may use when it joins the cluster: State Snapshot Transfer: (SST) and Incremental State Transfer (IST).

- SST is the full copy of data from one node to another. It's used when a new node joins the cluster, it has to transfer data from existing node. There are three methods of SST available in Percona XtraDB Cluster: **`mysqldump`**, **`rsync`** and **`xtrabackup`**. The downside of *mysqldump* and *rsync* is that your cluster becomes *READ-ONLY* while data is being copied from one node to another (SST applies **`FLUSH TABLES WITH READ LOCK`** command). Xtrabackup SST does not require **`READ LOCK`** for the entire syncing process, only for syncing *.frm* files (the same as with regular backup).
- Even with that, SST may be intrusive, that's why there is IST mechanism. If you put your node down for a short period of time and then start it, the node is able to fetch only those changes made during the period it was down. This is done using caching mechanism on nodes. Each node contains a cache, ring-buffer, (the size is configurable) of last N changes, and the node is able to transfer part of this cache. Obviously, IST can be done only if the amount of changes needed to transfer is less than N. If it exceeds N, then the joining node has to perform SST.

You can monitor current state of Node by using

```
SHOW STATUS LIKE 'wsrep_local_state_comment';
```

When it is *Synced (6)*, the node is ready to handle traffic.

CHAPTER

SEVEN

MULTI-MASTER REPLICATION

Multi-Master replication stands for the ability to write to any node in the cluster, and not to worry that eventually it will get out-of-sync situation, as it regularly happens with regular MySQL replication if you imprudently write to the wrong server. This is a long-waited feature and there has been growing demand for it for the last two years, or even more.

With *Percona XtraDB Cluster* you can write to any node, and the Cluster guarantees consistency of writes. That is, the write is either committed on all the nodes or not committed at all. For the simplicity, this diagram shows the use of the two-node example, but the same logic is applied with the N nodes:

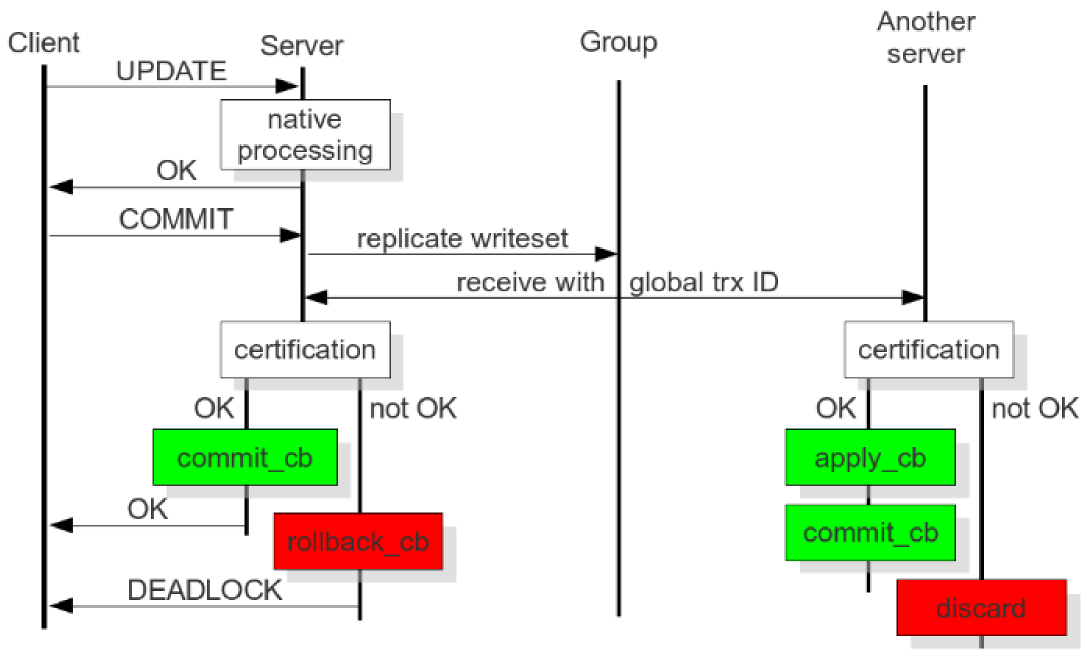


Fig. 7.1: *Image source:* Galera documentation - HOW CERTIFICATION-BASED REPLICATION WORKS

All queries are executed locally on the node, and there is a special handling only on *COMMIT*. When the *COMMIT* is issued, the transaction has to pass certification on all the nodes. If it does not pass, you will receive *ERROR* as a response on that query. After that, transaction is applied on the local node.

Response time of *COMMIT* consists of several parts:

- Network round-trip time,

- Certification time,
- Local applying

Please note that applying the transaction on remote nodes does not affect the response time of *COMMIT*, as it happens in the background after the response on certification.

The two important consequences of this architecture:

- First: we can have several appliers working in parallel. This gives us a true parallel replication. Slave can have many parallel threads, and this can be tuned by variable *`wsrep_slave_threads`*.
- Second: There might be a small period of time when the slave is out-of-sync from master. This happens because the master may apply event faster than a slave. And if you do read from the slave, you may read the data that has not changed yet. You can see that from the diagram. However, this behavior can be changed by using variable `wsrep_causal_reads=ON`. In this case, the read on the slave will wait until event is applied (this however will increase the response time of the read). This gap between the slave and the master is the reason why this replication is called "virtually synchronous replication", and not real "synchronous replication".

The described behavior of *COMMIT* also has the second serious implication. If you run write transactions to two different nodes, the cluster will use an optimistic locking model. That means a transaction will not check on possible locking conflicts during the individual queries, but rather on the *COMMIT* stage, and you may get *ERROR* response on *COMMIT*. This is mentioned because it is one of the incompatibilities with regular *InnoDB* that you might experience. In InnoDB usually *DEADLOCK* and *LOCK TIMEOUT* errors happen in response on particular query, but not on *COMMIT*. It's good practice to check the error codes after *COMMIT* query, but there are still many applications that do not do that.

If you plan to use Multi-Master capabilities of *XtraDB Cluster* and run write transactions on several nodes, you may need to make sure you handle response on *COMMIT* query.

See also:

***Percona* Blogpost: Multi-node writing and Unexpected Deadlocks** https://www.percona.com/blog/2012/08/17/percona-xtradb-cluster-multi-node-writing-and-unexpected-deadlocks/

***Percona Server for MySQL* Documentation: Count InnoDB Deadlocks** https://www.percona.com/doc/percona-server/5.6/diagnostics/innodb_deadlock_count.html

Part IV

User’s Manual

CHAPTER

EIGHT

BOOTSTRAPPING THE CLUSTER

Bootstrapping refers to getting the initial cluster up and running. By bootstrapping you are defining which node is has the correct information, that all the other nodes should synchronize to (via *SST*). In the event of a cluster-wide crash, bootstrapping functions the same way: by picking the initial node, you are essentially deciding which cluster node contains the database you want to go forward with.

The *MySQL* configuration file should contain necessary configuration options to start the *Percona XtraDB Cluster*:

```
[mysqld]
# Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so
# Cluster connection URL
wsrep_cluster_address=gcomm://
# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW
# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB
# This changes how |InnoDB| autoincrement locks are managed and is a requirement for␣
↪Galera
innodb_autoinc_lock_mode=2
```

Bootstrapping the cluster is a bit of a manual process. On the initial node, variable `wsrep_cluster_address` should be set to the value: `gcomm://`. The `gcomm://` tells the node it can bootstrap without any cluster to connect to. Setting that and starting up the first node should result in a cluster with a `wsrep_cluster_conf_id` of 1. After this single-node cluster is started, variable `wsrep_cluster_address` should be updated to the list of all nodes in the cluster. For example:

```
wsrep_cluster_address=gcomm://192.168.70.2,192.168.70.3,192.168.70.4
```

Although note that cluster membership is not defined by this setting, it is defined by the nodes that join the cluster with the proper cluster name configured Variable `wsrep_cluster_name` is used for that, if not explicitly set it will default to `my_wsrep_cluster`. Hence, variable `wsrep_cluster_address` does not need to be identical on all nodes, it's just a best practice because on restart the node will try all other nodes in that list and look for any that are currently up and running the cluster.

Once the first node is configured, then each other node should be started, one at a time. In a bootstrap situation, SST is most likely, so generally multiple nodes joining at once should be avoided.

In case cluster that's being bootstrapped has already been set up before, and to avoid editing the `my.cnf` twice to change the `wsrep_cluster_address` to `gcomm://` and then to change it back to other node addresses, first node can be started with:

```
/etc/init.d/mysql bootstrap-pxc
```

Note: On CentOS/RHEL 7 following bootstrap command should be used:

```
systemctl start mysql@bootstrap.service
```

This way values in `my.cnf` would remain unchanged. Next time node is restarted it won't require updating the configuration file. This can be useful in case cluster has been previously set up and for some reason all nodes went down and the cluster needs to be bootstrapped again.

Note: A service started with `mysql@bootstrap` must be stopped using the same name. For example, the `systemctl stop mysql` command does not stop an instance started with the `mysql@bootstrap` command.

Other Reading

- How to start a Percona XtraDB Cluster

CHAPTER

NINE

STATE SNAPSHOT TRANSFER

State Snapshot Transfer is a full data copy from one node (donor) to the joining node (joiner). It's used when a new node joins the cluster. In order to be synchronized with the cluster, new node has to transfer data from the node that is already part of the cluster. There are three methods of SST available in Percona XtraDB Cluster: **mysqldump**, **rsync** and **xtrabackup**. The downside of *mysqldump* and *rsync* is that the donor node becomes *READ-ONLY* while data is being copied from one node to another. Xtrabackup SST, on the other hand, uses backup locks, which means galera provider is not paused at all as with FTWRL (Flush Tables with Read Lock) earlier. State snapshot transfer method can be configured with the *wsrep_sst_method* variable.

For more information, see *wsrep_desync*.

Note: If the variable *gcs.sync_donor* is set to `Yes` (default `No`), whole cluster will get blocked if the donor is blocked by the State Snapshot Transfer and not just the donor node.

Choosing the SST Donor

If there are no nodes available that can safely perform an incremental state transfer, the cluster defaults to a state snapshot transfer. If there are nodes available that can safely perform an incremental state transfer, the cluster prefers a local node over remote nodes to serve as the donor. If there are no local nodes available that can safely perform an incremental state transfer, the cluster chooses a remote node to serve as the donor. Where there are several local or remote nodes available that can safely perform an incremental state transfer, the cluster chooses the node with the highest `seqno` to serve as the donor.

Using *Percona Xtrabackup*

This is the default SST method (version 2 of it: xtrabackup-v2). This is the least blocking method as it uses backup locks. *XtraBackup* is run locally on the donor node, so it's important that the correct user credentials are set up on the donor node. In order for PXC to perform the SST using the *XtraBackup*, credentials for connecting to the donor node need to be set up in the variable *wsrep_sst_auth*. Beside the credentials, one more important thing is that the *datadir* needs to be specified in the server configuration file `my.cnf`, otherwise the transfer process will fail.

More information about the required credentials can be found in the *XtraBackup* manual. Easy way to test if the credentials will work is to run the **innobackupex** on the donor node with the username and password specified in the variable *wsrep_sst_auth*. For example, if the value of the *wsrep_sst_auth* is `root:Passw0rd` **innobackupex** command should look like:

```
innobackupex --user=root --password=Passw0rd /tmp/
```

Detailed information on this method are provided in *Percona XtraBackup SST Configuration* documentation.

Using `mysqldump`

This method uses the standard **`mysqldump`** to dump all the databases from the donor node and import them to the joining node. For this method to work *`wsrep_sst_auth`* needs to be set up with the root credentials. This method is the slowest one and it also performs the global lock while doing the *SST* which will block writes to the donor node.

Script used for this method can be found in `/usr/bin/wsrep_sst_mysqldump` and it's provided with the *Percona XtraDB Cluster* binary packages.

Using `rsync`

This method uses **`rsync`** to copy files from donor to the joining node. In some cases this can be faster than using the *XtraBackup* but requires the global data lock which will block writes to the donor node. This method doesn't require username/password credentials to be set up in the variable *`wsrep_sst_auth`*.

Script used for this method can be found in `/usr/bin/wsrep_sst_rsync` and it's provided with the *Percona XtraDB Cluster* binary packages.

SST for tables with tablespaces that are not in the data directory

For example:

```
CREATE TABLE t1 (c1 INT PRIMARY KEY) DATA DIRECTORY = '/alternative/directory';
```

The result depends on the SST method:

- SST using `rsync`

  SST will report success, however the table's data will not be copied over, since `rsync` just copies the files. You will not be able to access the table on the joiner node:

  ```
  mysql> select * from t1;
  ERROR 1812 (HY000): Tablespace is missing for table `sbtest`.`t1`.
  ```

- SST using `mysqldump`

  Works as expected. If the file does not exist, it will be created. Otherwise it will attempt to use the file (if the file doesn't have the expected format, an error is returned).

- SST using Percona XtraBackup

  XtraBackup will restore the table to the same location on the joiner node. If the target directory does not exist, it will be created. If the target file already exists, an error will be returned, because XtraBackup cannot clear tablespaces not in the data directory.

Other Reading

- SST Methods for MySQL
- *Xtrabackup SST configuration*

CHAPTER

TEN

PERCONA XTRABACKUP SST CONFIGURATION

Percona XtraBackup SST works in two stages:

- First it identifies the type of data transfer based on the presence of `xtrabackup_ist` file on the joiner node.
- Then it starts data transfer:
  - In case of *SST*, it empties the data directory except for some files (`galera.cache`, `sst_in_progress`, `grastate.dat`) and then proceeds with the SST
  - In case of *IST*, it proceeds as before.

Note: To maintain compatibility with *Percona XtraDB Cluster* older than 5.5.33-23.7.6, use `xtrabackup` as SST method. For newer versions, `xtrabackup-v2` is recommended and also the default SST method.

Note: *Percona XtraBackup* 2.3.x and later is strongly recommended for XtraBackup SST.

SST Options

The following options specific to *SST* can be used in `my.cnf` under `[sst]`.

Note: Considerations:

- Non-integer options which have no default value are disabled if not set.
- `:Match: Yes` implies that option should match on donor and joiner nodes.
- SST script reads `my.cnf` when it runs on either donor or joiner node, not during `mysqld` startup.
- SST options must be specified in the main `my.cnf` file.

option `streamfmt`

Values xbstream, tar

Default xbstream

Match Yes

Used to specify the Percona XtraBackup streaming format. The recommended value is `streamfmt=xbstream`. Certain features are not available with `tar`: encryption, compression, parallel streaming, streaming incremental backups. For more information about the `xbstream` format, see The xbstream Binary.

option `transferfmt`

> **Values** `socat`, `nc`
>
> **Default** `socat`
>
> **Match** Yes

Used to specify the data transfer format. The recommended value is the default `transferfmt=socat` because it allows for socket options, such as transfer buffer sizes. For more information, see socat(1).

Note: Using `transferfmt=nc` does not support any of the SSL-based encryption modes (values `2`, `3`, and `4` for the *`encrypt`* option). Only `encrypt=1` is supported.

---

option `tca`

> **Example** tca=~/etc/ssl/certs/mycert.crt

Used to specify the full path to the certificate authority (CA) file for `socat` encryption based on OpenSSL.

option `tcert`

> **Example** tcert=~/etc/ssl/certs/mycert.pem

Used to specify the full path to the certificate file in PEM format for `socat` encryption based on OpenSSL.

Note: For more information about `tca` and `tcert`, refer to http://www.dest-unreach.org/socat/doc/socat-openssltunnel.html. The `tca` is essentially the self-signed certificate in that example, and `tcert` is the PEM file generated after concatenation of the key and the certificate generated earlier. The names of options were chosen to be compatible with `socat` parameter names as well as with MySQL's SSL authentication. For testing you can also download certificates from Github.

Note: Irrespective of what is shown in the example, you can use the same `.crt` and `.pem` files on all nodes and it will work, since there is no server-client paradigm here but a cluster with homogeneous nodes.

---

option `tkey`

Used to specify the full path to the key file for SST encryption based on OpenSSL.

option `encrypt`

> **Values** `0`, `1`, `2`, `3`, `4`
>
> **Default** `0`
>
> **Match** Yes

Note: Modes `1`, `2`, and `3` have been deprecated in favor of mode `4`.

Used to enable and specify SST encryption mode:

- Set `encrypt=0` to disable SST encryption. This is the default value.
- Set `encrypt=1` to perform symmetric SST encryption based on XtraBackup.

  Note: This mode has been deprecated.

- Set `encrypt=2` to perform SST encryption based on OpenSSL with `socat`. Ensure that `socat` is built with OpenSSL: `socat -V | grep OPENSSL`. This is recommended if your nodes are over WAN and security constraints are higher.

  Note: This mode has been deprecated.

- Set `encrypt=3` to perform SST encryption based on SSL for just the key and certificate files as implemented in Galera.

  Note: This mode has been deprecated.

  It does not provide certificate validation. In order to work correctly, paths to the key and certificate files need to be specified as well, for example:

  ```
  [sst]
  encrypt=3
  tkey=/etc/mysql/key.pem
  tcert=/etc/mysql/cert.pem
  ```

  Note: The `encrypt=3` option can only be used when *`wsrep_sst_method`* is set to `xtrabackup-v2` (which is the default now).

- Set `encrypt=4` for SST encryption with SSL files generated by MySQL. This is the recommended mode.

  Considering that you have all three necessary files:

  ```
  [sst]
  encrypt=4
  ssl-ca=ca.pem
  ssl-cert=server-cert.pem
  ssl-key=server-key.pem
  ```

For more information, see *Encrypting PXC Traffic*.

option `encrypt-algo`

Values `AES128`, `AES192`, `AES256`

Used to specify the SST encryption algorithm. It uses the same values as the `--encryption` option for *Percona XtraBackup* (see this document). The `encrypt-algo` option is considered only if *`encrypt`* is set to `1`.

option `sockopt`

Used to specify key/value pairs of socket options, separated by commas, for example:

```
[sst]
sockopt="retry=2,interval=3"
```

The previous example causes socat to try to connect three times (initial attempt and two retries with a 3-second interval between attempts).

Note: For versions of *Percona XtraDB Cluster* before 5.6.35-26.20-3, the value must begin with a comma, for example:

```
[sst]
sockopt=",cipher=AES128"
```

You can use the `tcpwrap` option to blacklist or whitelist clients. For more information about socket options, see socat (1).

Note: You can also enable SSL based compression with *sockopt*. This can be used instead of the Percona XtraBackup `compress` option.

option ncsockopt

Used to specify socket options for the `netcat` transfer format (`transferfmt=nc`).

option progress

Values `1` or path to file

Used to specify where to write SST progress. If set to `1`, it writes to MySQL `stderr`. Alternatively, you can specify the full path to a file. If this is a FIFO, it needs to exist and be open on reader end before itself, otherwise `wsrep_sst_xtrabackup` will block indefinitely.

Note: Value of `0` is not valid.

option rebuild

Values `0`, `1`

Default `0`

Used to enable rebuilding of index on joiner node. Set to `1` to enable. This is independent of compaction, though compaction enables it. Rebuild of indexes may be used as an optimization.

Note: Bug #1192834 affects this option.

option time

Values `0`, `1`

Default `0`

Enabling this option instruments key stages of backup and restore in SST.

option rlimit

Example `rlimit=128k`

Used to set a ratelimit in bytes. Add a suffix (k, m, g, t) to specify other units. For example, `128k` is 128 kilobytes. Refer to pv(1) for details.

Note: Rate is limited on donor node. The rationale behind this is to not allow SST to saturate the donor's regular cluster operations or to limit the rate for other purposes.

option incremental

Values `0`, `1`

Default 0

Used to supersede IST on joiner node. Requires manual setup and is not supported currently.

option use_extra

Values 0, 1

Default 0

Used to force SST to use the thread pool's extra_port. Make sure that thread pool is enabled and the `extra_port` option is set in `my.cnf` before you enable this option.

option cpat

Used to define the files that need to be retained in the *datadir* before running SST, so that the state of the other node can be restored cleanly. For example:

```
[sst]
cpat='.*galera\.cache$\|.*sst_in_progress$\|.*grastate\.dat$\|.*\.err$\|.*\.log$\|.
↪*RPM_UPGRADE_MARKER$\|.*RPM_UPGRADE_HISTORY$\|.*\.xyz$'
```

Note: This option can only be used when *`wsrep_sst_method`* is set to `xtrabackup-v2`.

option compressor

Default not set (disabled)

Example `compressor=gzip`

option decompressor

Default not set (disabled)

Example `decompressor='gzip -dc'`

Two previous options enable stream-based compression/decompression. When these options are set, compression/decompression is performed on stream, in contrast to earlier PXB-based one where decompression was done after streaming to disk, involving additional I/O. This saves a lot of I/O (up to twice less I/O on joiner node).

You can use any compression utility which works on stream: `gzip`, `pigz` (which is recommended because it is multi-threaded), etc. Compressor has to be set on donor node and decompressor on joiner node (although you can set them vice-versa for configuration homogeneity, it won't affect that particular SST). To use XtraBackup based compression as before, set `compress` under `[xtrabackup]`. Having both enabled won't cause any failure (although you will be wasting CPU cycles).

option inno-backup-opts

option inno-apply-opts

option inno-move-opts

Default Empty

Type Quoted String

This group of options can be used to pass innobackupex options for backup, apply, and move stages.

Note: Although these options are related to XtraBackup SST, they cannot be specified in `my.cnf`, because they are for passing innobackupex options.

option sst-initial-timeout

Default `100`

Unit seconds

This option is used to configure initial timeout (in seconds) to receive the first packet via SST. This has been implemented, so that if the donor node fails somewhere in the process, the joiner node will not hang up and wait forever.

By default, the joiner node will not wait for more than 100 seconds to get a donor node. The default should be sufficient, however, it is configurable, so you can set it appropriately for your cluster. To disable initial SST timeout, set `sst-initial-timeout=0`.

Note: If you are using *`wsrep_sst_donor`* and you want the joiner node to strictly wait for donors listed in the variable and not fall back (that is, without a terminating comma at the end), **and** there is a possibility of **all** nodes in that variable to be unavailable, disable initial SST timeout or set it to a higher value (maximum threshold that you want the joiner node to wait). You can also disable this option (or set it to a higher value) if you believe all other nodes in the cluster can potentially become unavailable at any point in time (mostly in small clusters) or there is a high network latency or network disturbance (which can cause donor selection to take longer than 100 seconds).

option `tmpdir`

Version Introduced in 5.6.35-26.20-3

Default Empty

Example /path/to/tmp/dir

This option specifies the location for storing the temporary file where the transaction log is stored before streaming or copying it to a remote host.

The `tmpdir` option can be set in the following `my.cnf` groups:

- `[sst]` is the primary location (others are ignored)
- `[xtrabackup]` is the secondary location (if not specified under `[sst]`)
- `[mysqld]` is used if it is not specified in either of the above

XtraBackup SST Dependencies

Each suppored version of *Percona XtraDB Cluster* is tested against a specific version of Percona XtraBackup:

- *Percona XtraDB Cluster* 5.6 requires Percona XtraBackup 2.3
- *Percona XtraDB Cluster* 5.7 requires Percona XtraBackup 2.4
- *Percona XtraDB Cluster* 8.0 requires Percona XtraBackup 8.0

Other combinations are not guaranteed to work.

The following are optional dependencies of *Percona XtraDB Cluster* introduced by `wsrep_sst_xtrabackup` (except for obvious and direct dependencies):

- `qpress` for decompression. It is an optional dependency of *Percona XtraBackup* and it is available in our software repositories.
- `my_print_defaults` to extract values from `my.cnf`. Provided by the server package.
- `openbsd-netcat` or `socat` for transfer. `socat` is a direct dependency of *Percona XtraDB Cluster* and it is the default.
- `xbstream` for streaming.

- `pv` is required for *`progress`* and *`rlimit`*.
- `mkfifo` is required for *`progress`*. Provided by `coreutils`.
- `mktemp` is required for *`incremental`*. Provided by `coreutils`.

XtraBackup-based Encryption

This is enabled when *`encrypt`* is set to `1` under `[sst]` in `my.cnf`. However, due to bug #1190335, it will also be enabled when you specify any of the following options under `[xtrabackup]` in `my.cnf`:

- `encrypt`
- `encrypt-key`
- `encrypt-key-file`

There is no way to disable encryption from innobackupex if any of the above are in `my.cnf` under `[xtrabackup]`. For that reason, consider the following scenarios:

1. If you want to use XtraBackup-based encryption for SST but not otherwise, use `encrypt=1` under `[sst]` and provide the above *Percona XtraBackup* encryption options under `[sst]`. Details of those options can be found here.
2. If you want to use XtraBackup-based encryption always, use `encrypt=1` under `[sst]` and have the above *Percona XtraBackup* encryption options either under `[sst]` or `[xtrabackup]`.
3. If you don't want to use XtraBackup-based encryption for SST, but want it otherwise, use `encrypt=0` or `encrypt=2` and do **NOT** provide any *Percona XtraBackup* encryption options under `[xtrabackup]`. You can still have them under `[sst]` though. You will need to provide those options on innobackupex command line then.
4. If you don't want to use XtraBackup-based encryption at all (or only the OpenSSL-based for SST with `encrypt=2`), then don't provide any *Percona XtraBackup* encryption options in `my.cnf`.

Note: The *`encrypt`* option under `[sst]` is different from the one under `[xtrabackup]`. The former is for disabling/changing encryption mode, while the latter is to provide an encryption algorithm. To disambiguate, if you need to provide the latter under `[sst]` (for example, in cases 1 and 2 above), it should be specified as *`encrypt-algo`*.

Warning: An implication of the above is that if you specify any of the *Percona XtraBackup* encryption options, and `encrypt=0` under `[sst]`, it will still be encrypted and SST will fail. Look at case 3 above for resolution.

Memory Allocation

The amount of memory for XtraBackup is defined by the `--use-memory` option. You can pass it using the *`inno-apply-opts`* option under `[sst]` as follows:

```
[sst]
inno-apply-opts="--use-memory=500M"
```

If it is not specified, the `use-memory` option under `[xtrabackup]` will be used:

```
[xtrabackup]
use-memory=32M
```

If neither of the above are specified, the size of the InnoDB memory buffer will be used:

```
[mysqld]
innodb_buffer_pool_size=24M
```

CHAPTER

ELEVEN

RESTARTING THE CLUSTER NODES

Restarting a cluster node is as simple as shutting down and restarting standard mysql. The node should gracefully leave the cluster (and the total vote count for *quorum* should decrement). When it rejoins, the node should receive an *IST* of changes since it left so it can get back in sync. If the set of changes needed for IST are not found in the `gcache` file on any other node in the entire cluster, then an *SST* will be performed instead. Therefore, restarting cluster nodes for rolling configuration changes or software upgrades should be fairly trivial to accomplish from the cluster's perspective.

Note: If a configuration change is done and mysql restarted and that change happens to contain a misspelling or some other mistake that prevents mysqld from loading, Galera will generally decide to drop its state and an SST will be forced for that node.

CHAPTER

TWELVE

CLUSTER FAILOVER

Cluster membership is determined simply by which nodes are connected to the rest of the cluster; there is no configuration setting explicitly defining the list of all possible cluster nodes. Therefore, every time a node joins the cluster, the total size of the cluster is increased and when a node leaves (gracefully) the size is decreased.

The size of the cluster is used to determine the required votes to achieve *quorum*. A quorum vote is done when a node or nodes are suspected to no longer be part of the cluster (they do not respond). This no response timeout is the `evs.suspect_timeout` setting in the `wsrep_provider_options` (default 5 sec), and when a node goes down ungracefully, write operations will be blocked on the cluster for slightly longer than that timeout.

Once the node (or nodes) is determined to be disconnected, then the remaining nodes cast a quorum vote and if a majority remain from the total nodes connected from before the disconnect, then that partition remains up. In the case of a network partition, some nodes will be alive and active on each side of the network disconnect. In this case, only the quorum will continue, the partition(s) without quorum will go to the non-Primary state.

Because of this, it's not possible to safely have automatic failover in a 2 node cluster, because the failure of one node will cause the remaining node to go non-Primary. Further, cluster with an even number of nodes (say two nodes in two different switches) have some possibility of a split brain condition when if network connectivity is lost between the two partitions, neither would retain quorum, and so both would go to Non-Primary. Therefore: for automatic failover, the "rule of 3s" is recommended. It applies at various levels of infrastructure, depending on how far cluster is spread out to avoid single points of failure. For example:

- A cluster on a single switch should have 3 nodes
- A cluster spanning switches should be spread evenly across at least 3 switches
- A cluster spanning networks should be span at least 3 networks
- A cluster spanning data centers should span at least 3 data centers

This is all to prevent split brain situations from preventing automatic failover from working.

Using an arbitrator

In the case where the expense of adding the third node/switch/datacenter/etc. above is prohibitively high, using an arbitrator node may be a viable alternative. An arbitrator is a voting member of the cluster which does receive and can relay replication, but it does not persist any data and does not run mysqld, it is a separate daemon. Placing even a single arbitrator in a 3rd location can add split brain protection to a cluster that is spread only across two nodes/locations.

Recovering a Non-Primary cluster

It is important to note that the rule of 3s only applies for automatic failover. In the event of a 2 node cluster (or in the event of some other outage that leaves a minority of nodes active), the failure of one will cause the other to shift to non-Primary and refuse operations. However, that is a recoverable state via a manual command:

```
SET GLOBAL wsrep_provider_options='pc.bootstrap=true';
```

This will tell the node (and all nodes still connected to its partition) that it can become a Primary cluster. However, this is only safe to do when you are sure there exists no other partition operating in Primary as well, or else *Percona XtraDB Cluster* will allow those two partitions to diverge (and now you have two databases that are impossible to remerge automatically). For example, if there are two data centers where one is primary and one is for Disaster Recovery, with even number of nodes in each. When an extra arbitrator node is run only in the Primary data center, the following High Availability features will be available:

- Auto-failover of any single node or nodes within the Primary or Secondary data center
- Failure of the secondary data center would not cause Primary to go down (because of the arbitrator)
- Failure of the primary data center would leave the secondary in a non-Primary state.
- If a disaster recovery failover has been executed. In this case you could simply tell the secondary data center to bootstrap itself with a single command, but disaster recovery failover remains in your control.

Other Reading

- PXC - Failure Scenarios with only 2 nodes

CHAPTER

THIRTEEN

MONITORING THE CLUSTER

The important bit about the cluster is that each node should be monitored independently. There is no centralized node, the cluster is the set of active, connected nodes, and each can have a different view of the cluster. Further, many of these variables are relative to the node you query them from: for example, replication sent (from this node) and received (from writes on the rest of the cluster). Having data from all nodes helps tracking down the source of issues (i.e., where are the flow control messages coming from? Where did that 100MB transaction came from?).

Manually

Manual cluster monitoring can be done with myq_gadgets.

Alerting

Standard *MySQL* alerting should apply here. *Percona XtraDB Cluster* specific alerting should include:

- Cluster state of each node (`wsrep_cluster_status` != Primary)
- Node state (`wsrep_connected`, `wsrep_ready` != ON)

Other optional alerting could be done on:

- Excessive replication conflicts (high rate of `wsrep_local_cert_failures` and `wsrep_local_bf_aborts`)
- Excessive Flow control messages (`wsrep_flow_control_sent`/ `wsrep_flow_control_recv`)
- Large replication queues (`wsrep_local_recv_queue`).

Metrics

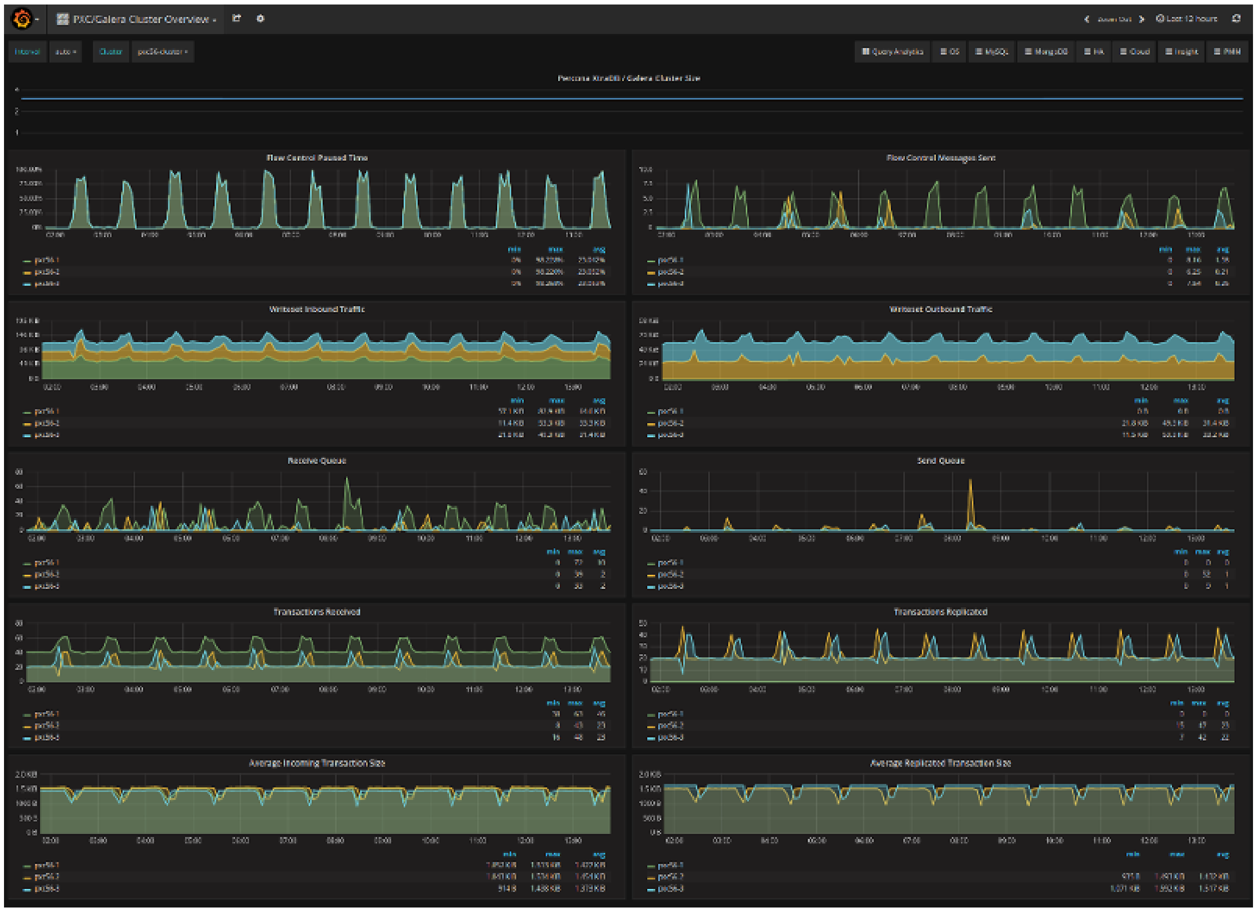
Metrics collection (i.e., long-term graphing) on the cluster should be done on:

- Queue sizes (`wsrep_local_recv_queue`, `wsrep_local_send_queue`)
- Flow control (`wsrep_flow_control_sent`, `wsrep_flow_control_recv`)
- Number of transactions in and out of this node (`wsrep_replicated`, `wsrep_received`)
- Number of transactions in and out in bytes (`wsrep_replicated_bytes`, `wsrep_received_bytes`)
- Replication conflicts (`wsrep_local_cert_failures` and `wsrep_local_bf_aborts`)

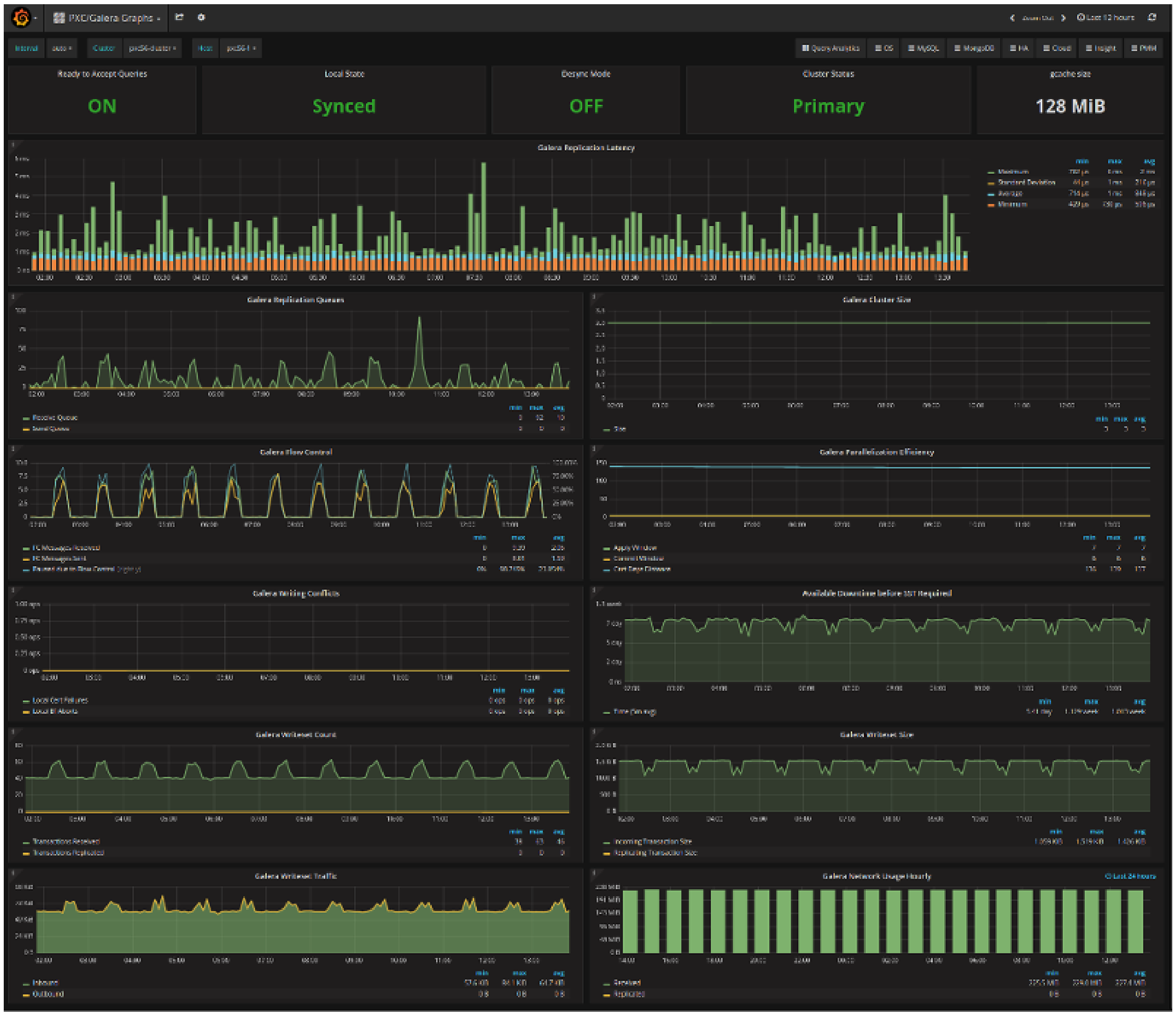
Using |PMM|

Percona Monitoring and Management includes two dashboards to monitor PXC:

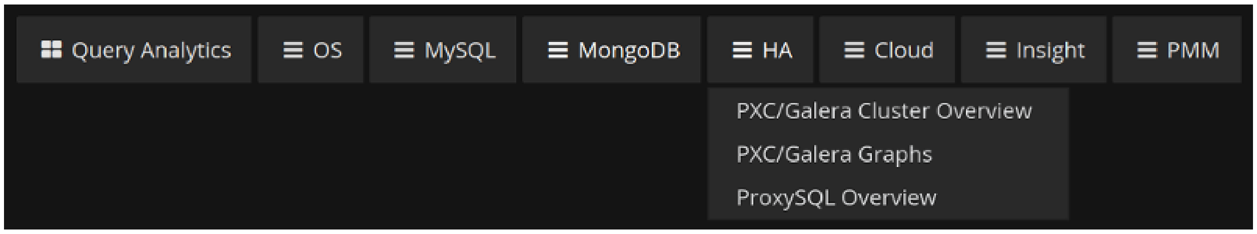
1. PXC/Galera Cluster Overview:



2. PXC/Galera Graphs:



These dashboards are available from the menu:



Please refer to the official documentation for details on |PMM| installation and setup.

Other Reading

- Realtime stats to pay attention to in PXC and Galera

CHAPTER

FOURTEEN

CERTIFICATION IN PERCONA XTRADB CLUSTER

Percona XtraDB Cluster replicates actions executed on one node to all other nodes in the cluster and make it fast enough to appear as it if is synchronous (aka virtually synchronous).

There are two main types of actions: DDL and DML. DDL actions are executed using Total Order Isolation (let's ignore Rolling Schema Upgrade for now) and DML using normal Galera replication protocol.

Note: This manual page assumes the reader is aware of Total Order Isolation and MySQL replication protocol.

DML (`INSERT`/`UPDATE`/`DELETE`) operations effectively change the state of the database, and all such operations are recorded in *XtraDB* by registering a unique object identifier (aka key) for each change (an update or a new addition).

- A transaction can change "n" different data objects. Each such object change is recorded in *XtraDB* using a so-call `append_key` operation. The `append_key` operation registers the key of the data object that has undergone a change by the transaction. The key for rows can be represented in three parts as `db_name`, `table_name`, and `pk_columns_for_table` (if `pk` is absent, a hash of the complete row is calculated). In short there is quick and short meta information that this transaction has touched/modified following rows. This information is passed on as part of the write-set for certification to all the nodes of a cluster while the transaction is in the commit phase.
- For a transaction to commit it has to pass XtraDB/Galera certification, ensuring that transactions don't conflict with any other changes posted on the cluster group/channel. Certification will add the keys modified by given the transaction to its own central certification vector (CCV), represented by `cert_index_ng`. If the said key is already part of the vector, then conflict resolution checks are triggered.
- Conflict resolution traces reference the transaction (that last modified this item in cluster group). If this reference transaction is from some other node, that suggests the same data was modified by the other node and changes of that node have been certified by the local node that is executing the check. In such cases, the transaction that arrived later fails to certify.

Changes made to DB objects are bin-logged. This is the same as how *MySQL* does it for replication with its Master-Slave ecosystem, except that a packet of changes from a given transaction is created and named as a write-set.

Once the client/user issues a `COMMIT`, *Percona XtraDB Cluster* will run a commit hook. Commit hooks ensure following:

- Flush the binary logs.
- Check if the transaction needs replication (not needed for read-only transactions like `SELECT`).
- If a transaction needs a replication, then it invokes a pre_commit hook in the Galera ecosystem. During this pre-commit hook, a write-set is written in the group channel by a "replicate" operation. All nodes (including the one that executed the transaction) subscribes to this group-channel and reads the write-set.
- `gcs_recv_thread` is first to receive the packet, which is then processed through different action handlers.

- Each packet read from the group-channel is assigned an `id`, which is a locally maintained counter by each node in sync with the group. When any new node joins the group/cluster, a seed-id for it is initialized to the current active id from group/cluster. (There is an inherent assumption/protocol enforcement that all nodes read the packet from a channel in same order, and that way even though each packet doesn't carry `id` information it is inherently established using the local maintained `id` value).

```
/*  Common situation -
* increment and assign act_id only for totally ordered actions
* and only in PRIM (skip messages while in state exchange) */
  rcvd->id = ++group->act_id_;

[This is an amazing way to solve the problem of the id co-ordination in
multiple master system, otherwise a node will have to first get an id from
central system or through a separate agreed protocol and then use it for the
packet there-by doubling the round-trip time].
```

What happens if two nodes get ready with their packet at same time?

- Both nodes will be allowed to put the packet on the channel. That means the channel will see packets from different nodes queued one-behind-another.
- It is interesting to understand what happens if two nodes modify same set of rows. For example:

  ```
  create -> insert (1,2,3,4)....nodes are in sync till this point.
  node-1: update i = i + 10;
  node-2: update i = i + 100;

  Let's associate transaction-id (trx-id) for an update transaction that
  is executed on node-1 and node-2 in parallel (The real algorithm is bit
  more involved (with uuid + seqno) but conceptually the same so for ease
  we're using trx_id here)

  node-1:
    update action: trx-id=n1x
  node-2:
    update action: trx-id=n2x
  ```

Both node packets are added to the channel but the transactions are conflicting. Let's see which one succeeds. The protocol says: FIRST WRITE WINS. So in this case, whoever is first to write to the channel will get certified. Let's say node-2 is first to write the packet and then node-1 makes immediately after it.

Note: each node subscribes to all packages including its own package. See below for details.

Node-2:

- Will see its own packet and will process it.
- Then it will see node-1 packet that it tries to certify but fails.

Node-1:

- Will see node-2 packet and will process it. (Note: InnoDB allows isolation and so node-1 can process node-2 packets independent of node-1 transaction changes)
- Then it will see the node-1 packet that it tries to certify but fails. (Note even though the packet originated from node-1 it will under-go certification to catch cases like thes. This is beauty of listening to own events that make consistent processing path even if events are locally generated)

The certification protocol will be described using the example from above. As discussed above, the central certification vector (CCV) is updated to reflect reference transaction.

Node-2:

- node-2 sees its own packet for certification, adds it to its local CCV and performs certification checks. Once these checks pass it updates the reference transaction by setting it to `n2x`
- node-2 then gets node-1 packet for certification. Said key is already present in CCV with a reference transaction set it to `n2x`, whereas write-set proposes setting it to `n1x`. This causes a conflict, which in turn causes the node-1 originated transaction to fail the certification test.

This helps point out a certification failure and the node-1 packet is rejected.

Node-1:

- node-1 sees node-2 packet for certification, which is then processed, the local CCV is updated and the reference transaction is set to `n2x`
- Using the same case explained above, node-1 certification also rejects the node-1 packet.

This suggests that the node doesn't need to wait for certification to complete, but just needs to ensure that the packet is written to the channel. The applier transaction will always win and the local conflicting transaction will be rolled back.

What happens if one of the nodes has local changes that are not synced with group?

```
create (id primary key) -> insert (1), (2), (3), (4);
node-1: wsrep_on=0; insert (5); wsrep_on=1
node-2: insert(5).
insert(5) will generate a write-set that will then be replicated to node-1.
node-1 will try to apply it but will fail with duplicate-key-error, as 5
already exist.

XtraDB will flag this as an error, which would eventually cause node-1 to
shutdown.
```

With all that in place, how is GTID incremented if all the packets are processed by all nodes (including ones that are rejected due to certification)? GTID is incremented only when the transaction passes certification and is ready for commit. That way errant-packets don't cause GTID to increment. Also, they don't confuse the group packet `id` quoted above with GTID. Without errant-packets, you may end up seeing these two counters going hand-in-hand, but they are no way related.

CHAPTER

FIFTEEN

PERCONA XTRADB CLUSTER THREADING MODEL

Percona XtraDB Cluster (PXC) creates a different set of threads to service its operations. These threads are in addition to existing *MySQL* threads. There are three main groups of threads:

Applier thread(s)

Applier threads are meant to apply write-sets that the node receives from other nodes (through cluster). (write-message is directed through `gcv_recv_thread` .)

The number of applier threads is controlled by using the *`wsrep_slave_threads`* configuration. (The default value is `1`, so at least one wsrep applier thread exists to process the request.)

Applier threads wait for an event, and once it gets the event, it applies it using the normal slave apply routine path (and relays the log info apply path with wsrep-customization). In short these threads are kind of similar to slave worker threads (but not exactly the same).

Coordination is achieved using Apply and Commit Monitor. A transaction passes through two important states: `APPLY -> COMMIT`. Every transaction registers itself with an apply monitor, where its apply order gets defined. So all transactions with apply-order sequence number (`seqno`) that is less than this transaction should be applied before applying this transaction. The same is done for commit as well (`last_left >= trx_.depends_seqno()`).

Rollback thread

Why do we need a rollback thread (there is only one rollback thread)?

- Transactions executed in parallel can conflict and may need to rollback. A rollback thread helps achieve this.
- Applier transactions always take priority over local transactions. This is natural, as applier transactions have been accepted by the cluster, and some of the nodes may have already applied them. Local conflicting transactions still has a window to rollback.

All the transactions that need to be rolled back are added to the rollback queue, and the rollback thread is notified. The rollback thread will then iterate over the queue and perform rollback operations.

If a transaction is active on a node, and a node receives a transaction-write-set from cluster-group that conflicts with the local active transaction, then such local transactions are always treated as a victim transaction to rollback. Transactions can be in a commit state or an execution stage when the conflict arises. Local transactions in the execution stage are forcibly killed so that the waiting applier transaction is allowed to proceed. Local transactions in the commit stage fail with a certification error.

Other threads

Service thread

This thread is created during boot-up and used to perform auxiliary services. It has two main functions:

- It releases the GCache buffer after the cached write-set is purged up to the said level.
- It notifies the cluster group that the respective node has committed a transaction up to this level. Each node maintains some basic status info about other nodes in the cluster. On receiving the message, the information is updated in this local metadata.

`gcs_recv_thread`

This thread is the first one to see all the messages received in a group.

It will try to assign actions against each message it receives. It adds these messages to a central FIFO queue, which are then processed by the Applier thread(s). Messages can include different operation like state change, configuration update, flow-control, etc. One important action is about processing write-set, which actually is applying transactions to database objects.

gcomm connection thread

There is also a gcomm connection thread `GCommConn::run_fn` which is used to co-ordinate the low-level group communication activity. Think of it as a black-box meant for communication.

Action-based threads

Besides the above, some threads are created on a needed basis. SST creates threads for donor and joiner (which eventually forks out a child process to host the needed SST script), IST creates receiver and async sender threads, PageStore creates a background thread for removing the files that were created. If the checksum is enabled and the replicated write-set is big enough, the checksum is done as part of a separate thread.

CHAPTER

SIXTEEN

UNDERSTANDING GCACHE AND RECORD-SET CACHE

In *Percona XtraDB Cluster* (PXC), there is a concept of GCache and Record-Set cache (which can also be called transaction write-set cache). The use of these two caches is often confusing if you are running long transactions, as both of them result in the creation of disk-level files. This manual describes what their main differences are.

Record-Set Cache

When you run a long-running transaction on any particular node, it will try to append a key for each row that it tries to modify (the key is a unique identifier for the row `{db,table,pk.columns}`). This information is cached in out-write-set, which is then sent to the group for certification.

To start with, keys are cached in HeapStore (which has `page-size=64K` and `total-size=4MB`). If the transaction data-size outgrows this limit, then the storage is switched from Heap to Page (which has a `page-size=64MB` and `total-limit=free-space-on-disk`). All these limits are non-configurable, but having a memory-page size greater than 4MB per transaction can cause things to stall due to memory pressure, so this limit is reasonable. (This is another limitation to address when Galera supports large transaction.)

The same long-running transaction will also generate binlog data that also appends to out-write-set on commit using the same technique explained above (`HeapStore->FileStore`). This data could be significant as it is a binlog image of rows inserted/updated/deleted by the transaction. Variable *`wsrep_max_ws_size`* controls the size of this part of the write set. (The threshold doesn't consider size allocated for caching-keys (above) and the header).

If `FileStore` is used, it creates a file on the disk (with names like `xxxx_keys` and `xxxx_data`) to store the cache data. These files are kept until a transaction is committed, so the lifetime of the transaction is linked.

When the node is done with the transaction and is about to commit, it will generate the final-write-set using the two files (if the data size grew enough to use `FileStore`) plus `HEADER`, and will publish it for certification to cluster.

The native node executing the transaction will also act as subscription node, and will receive its own write-set through the cluster publish mechanism. This time, the native node will try to cache write-set into its GCache. How much data GCache retains is controlled by the GCache configuration.

GCache

GCache holds the write-set published on the cluster for replication.

The lifetime of write-set in GCache is not transaction linked.

When a `JOINER` node needs an IST, it will be serviced through this GCache (if possible).

GCache will also create the files to disk. (You can read more about it here.)

Interestingly, at any given point in time, the native node has two copies of the write-set: one in GCache and other in Record-Set Cache.

For example:

If you `INSERT/UPDATE` 2M rows in a table with a schema like:

```
(int, char(100), char(100) with pk (int, char(100))
```

and it created write-set key/data files in the background that looked like this:

```
-rw------- 1 xxx xxx 67108864 Apr 11 12:26 0x00000707_data.000000
-rw------- 1 xxx xxx 67108864 Apr 11 12:26 0x00000707_data.000001
-rw------- 1 xxx xxx 67108864 Apr 11 12:26 0x00000707_data.000002
-rw------- 1 xxx xxx 67108864 Apr 11 12:26 0x00000707_keys.000000
```

Part V

How-tos

CHAPTER

SEVENTEEN

INSTALLING PERCONA XTRADB CLUSTER ON *CENTOS*

This tutorial will show how to install the *Percona XtraDB Cluster* on three CentOS 6.3 servers, using the packages from Percona repositories.

This cluster will be assembled of three servers/nodes:

```
node #1
hostname: percona1
IP: 192.168.70.71

node #2
hostname: percona2
IP: 192.168.70.72

node #3
hostname: percona3
IP: 192.168.70.73
```

Prerequisites

- All three nodes have a CentOS 6.3 installation.
- Firewall has been set up to allow connecting to ports 3306, 4444, 4567 and 4568
- SELinux is disabled

Installation

Installation information can be found in the *Installing Percona XtraDB Cluster* guide.

Configuring the nodes

Individual nodes should be configured to be able to bootstrap the cluster. More details about bootstrapping the cluster can be found in the *Bootstrapping the cluster* guide.

Configuration file `/etc/my.cnf` for the first node should look like:

```
[mysqld]

datadir=/var/lib/mysql
```

```
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so

# Cluster connection URL contains the IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.71,192.168.70.72,192.168.70.73

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This changes how InnoDB autoincrement locks are managed and is a requirement for␣
↪Galera
innodb_autoinc_lock_mode=2

# Node #1 address
wsrep_node_address=192.168.70.71

# SST method
wsrep_sst_method=xtrabackup-v2

# Cluster name
wsrep_cluster_name=my_centos_cluster

# Authentication for SST method
wsrep_sst_auth="sstuser:s3cret"
```

After this, first node can be started with the following command:

```
[root@percona1 ~]# /etc/init.d/mysql bootstrap-pxc
```

In case you're running this tutorial on *CentOS* 7 server, systemd bootstrap service should be used instead:

```
[root@percona1 ~]#  systemctl start mysql@bootstrap.service
```

This command will start the cluster with initial `wsrep_cluster_address` set to `gcomm://`. This way the cluster will be bootstrapped and in case the node or *MySQL* have to be restarted later, there would be no need to change the configuration file.

After the first node has been started, cluster status can be checked by:

```
mysql> show status like 'wsrep%';
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | c2883338-834d-11e2-0800-03c9c68e41ec |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 1                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+------------------------------+----------------------------------------+
40 rows in set (0.01 sec)
```

This output shows that the cluster has been successfully bootstrapped.

It's recommended not to leave the empty password for the root account. Password can be changed with:

```
mysql@percona1> UPDATE mysql.user SET password=PASSWORD("Passw0rd") where user='root';
mysql@percona1> FLUSH PRIVILEGES;
```

In order to perform successful *State Snapshot Transfer* using *XtraBackup* new user needs to be set up with proper privileges:

```
mysql@percona1> CREATE USER 'sstuser'@'localhost' IDENTIFIED BY 's3cret';
mysql@percona1> GRANT PROCESS, RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO
↪'sstuser'@'localhost';
mysql@percona1> FLUSH PRIVILEGES;
```

Note: MySQL root account can also be used for setting up the SST with Percona XtraBackup, but it's recommended to use a different (non-root) user for this.

Configuration file `/etc/my.cnf` on the second node (`percona2`) should look like this:

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so

# Cluster connection URL contains IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.71,192.168.70.72,192.168.70.73

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This changes how InnoDB autoincrement locks are managed and is a requirement for␣
↪Galera
innodb_autoinc_lock_mode=2

# Node #2 address
wsrep_node_address=192.168.70.72

# Cluster name
wsrep_cluster_name=my_centos_cluster

# SST method
wsrep_sst_method=xtrabackup-v2

#Authentication for SST method
wsrep_sst_auth="sstuser:s3cret"
```

Second node can be started with the following command:

```
[root@percona2 ~]# /etc/init.d/mysql start
```

After the server has been started it should receive the state snapshot transfer automatically. This means that the second node won't have the empty root password anymore. In order to connect to the cluster and check the status changed root password from the first node should be used. Cluster status can now be checked on both nodes. This is the example from the second node (`percona2`):

```
mysql> show status like 'wsrep%';
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | c2883338-834d-11e2-0800-03c9c68e41ec |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 2                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

This output shows that the new node has been successfully added to the cluster.

MySQL configuration file `/etc/my.cnf` on the third node (`percona3`) should look like this:

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so

# Cluster connection URL contains IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.71,192.168.70.72,192.168.70.73

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This changes how InnoDB autoincrement locks are managed and is a requirement for␣
↪Galera
innodb_autoinc_lock_mode=2

# Node #3 address
wsrep_node_address=192.168.70.73

# Cluster name
wsrep_cluster_name=my_centos_cluster

# SST method
wsrep_sst_method=xtrabackup-v2
```

```
#Authentication for SST method
wsrep_sst_auth="sstuser:s3cret"
```

Third node can now be started with the following command:

```
[root@percona3 ~]# /etc/init.d/mysql start
```

After the server has been started it should receive the SST same as the second node. Cluster status can now be checked on both nodes. This is the example from the third node (`percona3`):

```
mysql> show status like 'wsrep%';
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | c2883338-834d-11e2-0800-03c9c68e41ec |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 3                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

This output confirms that the third node has joined the cluster.

Testing the replication

Although the password change from the first node has replicated successfully, this example will show that writing on any node will replicate to the whole cluster. In order to check this, new database will be created on second node and table for that database will be created on the third node.

Creating the new database on the second node:

```
mysql@percona2> CREATE DATABASE percona;
Query OK, 1 row affected (0.01 sec)
```

Creating the `example` table on the third node:

```
mysql@percona3> USE percona;
Database changed

mysql@percona3> CREATE TABLE example (node_id INT PRIMARY KEY, node_name VARCHAR(30));
Query OK, 0 rows affected (0.05 sec)
```

Inserting records on the first node:

```
mysql@percona1> INSERT INTO percona.example VALUES (1, 'percona1');
Query OK, 1 row affected (0.02 sec)
```

Retrieving all the rows from that table on the second node:

```
mysql@percona2> SELECT * FROM percona.example;
+---------+-----------+
| node_id | node_name |
+---------+-----------+
|       1 | percona1  |
+---------+-----------+
1 row in set (0.00 sec)
```

This small example shows that all nodes in the cluster are synchronized and working as intended.

CHAPTER

EIGHTEEN

INSTALLING PERCONA XTRADB CLUSTER ON *UBUNTU*

This tutorial will show how to install the *Percona XtraDB Cluster* on three *Ubuntu* 12.04.2 LTS servers, using the packages from Percona repositories.

This cluster will be assembled of three servers/nodes:

```
node #1
hostname: pxc1
IP: 192.168.70.61

node #2
hostname: pxc2
IP: 192.168.70.62

node #3
hostname: pxc3
IP: 192.168.70.63
```

Prerequisites

- All three nodes have a *Ubuntu* 12.04.2 LTS installation.
- Firewall has been set up to allow connecting to ports 3306, 4444, 4567 and 4568
- AppArmor profile for *MySQL* is disabled

Installation

Installation information can be found in the *Installing Percona XtraDB Cluster* guide

Note: Debian/Ubuntu installation prompts for root password, this was set to: `Passw0rd`. After the packages have been installed, `mysqld` will be started automatically. In this example mysqld is stopped on all three nodes after successful installation with: `/etc/init.d/mysql stop`.

Configuring the nodes

Individual nodes should be configured to be able to bootstrap the cluster. More details about bootstrapping the cluster can be found in the *Bootstrapping the cluster* guide.

Configuration file /etc/mysql/my.cnf for the first node should look like:

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib/libgalera_smm.so

# Cluster connection URL contains the IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.61,192.168.70.62,192.168.70.63

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This changes how InnoDB autoincrement locks are managed and is a requirement for␣
↪Galera
innodb_autoinc_lock_mode=2

# Node #1 address
wsrep_node_address=192.168.70.61

# SST method
wsrep_sst_method=xtrabackup-v2

# Cluster name
wsrep_cluster_name=my_ubuntu_cluster

# Authentication for SST method
wsrep_sst_auth="sstuser:s3cretPass"
```

After this, first node can be started with the following command:

```
[root@pxc1 ~]# /etc/init.d/mysql bootstrap-pxc
```

This command will start the first node and bootstrap the cluster (more information about bootstrapping cluster can be found in *Bootstrapping the cluster* manual).

After the first node has been started, cluster status can be checked by:

```
mysql> show status like 'wsrep%';
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | b598af3e-ace3-11e2-0800-3e90eb9cd5d3 |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 1                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

This output shows that the cluster has been successfully bootstrapped.

In order to perform successful *State Snapshot Transfer* using *Percona XtraBackup* new user needs to be set up with proper privileges:

```
mysql@pxc1> CREATE USER 'sstuser'@'localhost' IDENTIFIED BY 's3cretPass';
mysql@pxc1> GRANT PROCESS, RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO 'sstuser
↪'@'localhost';
mysql@pxc1> FLUSH PRIVILEGES;
```

Note: MySQL root account can also be used for setting up the *State Snapshot Transfer* with *Percona XtraBackup*, but it's recommended to use a different (non-root) user for this.

Configuration file `/etc/mysql/my.cnf` on the second node (`pxc2`) should look like this:

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib/libgalera_smm.so

# Cluster connection URL contains IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.61,192.168.70.62,192.168.70.63

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This changes how InnoDB autoincrement locks are managed and is a requirement for␣
↪Galera
innodb_autoinc_lock_mode=2

# Node #2 address
wsrep_node_address=192.168.70.62

# Cluster name
wsrep_cluster_name=my_ubuntu_cluster

# SST method
wsrep_sst_method=xtrabackup-v2

#Authentication for SST method
wsrep_sst_auth="sstuser:s3cretPass"
```

Second node can be started with the following command:

```
[root@pxc2 ~]# /etc/init.d/mysql start
```

After the server has been started it should receive the state snapshot transfer automatically. Cluster status can now be checked on both nodes. This is the example from the second node (`pxc2`):

```
mysql> show status like 'wsrep%';
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | b598af3e-ace3-11e2-0800-3e90eb9cd5d3 |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 2                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

This output shows that the new node has been successfully added to the cluster.

MySQL configuration file `/etc/mysql/my.cnf` on the third node (`pxc3`) should look like this:

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

# Path to Galera library
wsrep_provider=/usr/lib/libgalera_smm.so

# Cluster connection URL contains IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.70.61,192.168.70.62,192.168.70.63

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This changes how InnoDB autoincrement locks are managed and is a requirement for␣
↪Galera
innodb_autoinc_lock_mode=2

# Node #3 address
wsrep_node_address=192.168.70.63

# Cluster name
wsrep_cluster_name=my_ubuntu_cluster

# SST method
wsrep_sst_method=xtrabackup-v2

#Authentication for SST method
wsrep_sst_auth="sstuser:s3cretPass"
```

Third node can now be started with the following command:

```
[root@pxc3 ~]# /etc/init.d/mysql start
```

After the server has been started it should receive the SST same as the second node. Cluster status can now be checked on both nodes. This is the example from the third node (`pxc3`):

```
mysql> show status like 'wsrep%';
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | b598af3e-ace3-11e2-0800-3e90eb9cd5d3 |
...
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
...
| wsrep_cluster_size         | 3                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
...
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
40 rows in set (0.01 sec)
```

This output confirms that the third node has joined the cluster.

Testing the replication

Although the password change from the first node has replicated successfully, this example will show that writing on any node will replicate to the whole cluster. In order to check this, new database will be created on second node and table for that database will be created on the third node.

Creating the new database on the second node:

```
mysql@pxc2> CREATE DATABASE percona;
Query OK, 1 row affected (0.01 sec)
```

Creating the `example` table on the third node:

```
mysql@pxc3> USE percona;
Database changed

mysql@pxc3> CREATE TABLE example (node_id INT PRIMARY KEY, node_name VARCHAR(30));
Query OK, 0 rows affected (0.05 sec)
```

Inserting records on the first node:

```
mysql@pxc1> INSERT INTO percona.example VALUES (1, 'percona1');
Query OK, 1 row affected (0.02 sec)
```

Retrieving all the rows from that table on the second node:

```
mysql@pxc2> SELECT * FROM percona.example;
+---------+-----------+
| node_id | node_name |
+---------+-----------+
|       1 | percona1  |
+---------+-----------+
1 row in set (0.00 sec)
```

This small example shows that all nodes in the cluster are synchronized and working as intended.

CHAPTER

NINETEEN

SETTING UP GALERA ARBITRATOR

Galera Arbitrator <http://galeracluster.com/documentation-webpages/arbitrator.html> is a member of *Percona XtraDB Cluster* which is used for voting in case you have a small number of servers (usually two) and don't want to add any more resources. Galera Arbitrator does not need a dedicated server. It can be installed on a machine running some other application. Just make sure it has good network connectivity.

Galera Arbitrator is a member of the cluster which participates in the voting, but not in the actual replication (although it receives the same data as other nodes).

This document will show how to add Galera Arbitrator node to a already set up cluster.

Note: For more information on how to set up a cluster you can read in the *Installing Percona XtraDB Cluster on Ubuntu* or *Installing Percona XtraDB Cluster on CentOS* manuals.

Installation

Galera Arbitrator can be installed from Percona's repository by running:

```
root@ubuntu:~# apt-get install percona-xtradb-cluster-garbd-3
```

on Debian/Ubuntu distributions, or:

```
[root@centos ~]# yum install Percona-XtraDB-Cluster-garbd-3
```

on CentOS/RHEL distributions.

Configuration

To configure *Galera Arbitrator* on *Ubuntu/Debian* you need to edit the `/etc/default/garbd` file. On *CentOS/RHEL* configuration can be found in `/etc/sysconfig/garb` file.

Configuration file should look like this after installation:

```
# Copyright (C) 2012 Codership Oy
# This config file is to be sourced by garb service script.

# REMOVE THIS AFTER CONFIGURATION

# A comma-separated list of node addresses (address[:port]) in the cluster
# GALERA_NODES=""

# Galera cluster name, should be the same as on the rest of the nodes.
# GALERA_GROUP=""

# Optional Galera internal options string (e.g. SSL settings)
# see http://galeracluster.com/documentation-webpages/galeraparameters.html
# GALERA_OPTIONS=""

# Log file for garbd. Optional, by default logs to syslog
# Deprecated for CentOS7, use journalctl to query the log for garbd
# LOG_FILE=""
```

To set it up you'll need to add the information about the cluster you've set up. This example is using cluster information from the *Installing Percona XtraDB Cluster on Ubuntu*.

```
# Copyright (C) 2012 Codership Oy
# This config file is to be sourced by garb service script.

# A comma-separated list of node addresses (address[:port]) in the cluster
GALERA_NODES="192.168.70.61:4567, 192.168.70.62:4567, 192.168.70.63:4567"

# Galera cluster name, should be the same as on the rest of the nodes.
GALERA_GROUP="my_ubuntu_cluster"

# Optional Galera internal options string (e.g. SSL settings)
# see http://galeracluster.com/documentation-webpages/galeraparameters.html
# GALERA_OPTIONS=""

# Log file for garbd. Optional, by default logs to syslog
# Deprecated for CentOS7, use journalctl to query the log for garbd
# LOG_FILE=""
```

Note: Please note that you need to remove the `# REMOVE THIS AFTER CONFIGURATION` line before you can start the service.

You can now start the *Galera Arbitrator* daemon (`garbd`) by running:

- On Debian or Ubuntu:

```
root@server:~# service garbd start
[ ok ] Starting /usr/bin/garbd: :.
```

- On Red Hat Enterprise Linux or CentOS:

```
root@server:~# service garb start
[ ok ] Starting /usr/bin/garbd: :.
```

You can additionally check the `arbitrator` status by running:

- On Debian or Ubuntu:

```
root@server:~# service garbd status
[ ok ] garb is running.
```

- On Red Hat Enterprise Linux or CentOS:

```
root@server:~# service garb status
[ ok ] garb is running.
```

CHAPTER

TWENTY

HOW TO SETUP 3 NODE CLUSTER ON SINGLE BOX

This example shows how to setup 3-node cluster on the single physical box. Assume you installed *Percona XtraDB Cluster* from binary `.tar.gz` into directory

```
/usr/local/Percona-XtraDB-Cluster-5.5.24-23.6.342.Linux.x86_64
```

To start the cluster with three nodes, three `my.cnf` mysql configuration files should be created with three separate data directories.

For this example we created (see the content of files at the end of document):

- /etc/my.4000.cnf
- /etc/my.5000.cnf
- /etc/my.6000.cnf

and data directories:

- /data/bench/d1
- /data/bench/d2
- /data/bench/d3

In this example local IP address is 192.168.2.21

Then we should be able to start initial node as (from directory `/usr/local/Percona-XtraDB-Cluster-5.6.15-25.3.706.Linux.x86_64`):

```
bin/mysqld_safe --defaults-file=/etc/my.4000.cnf --wsrep-new-cluster
```

Following output will let out know that node was started successfully:

```
111215 19:01:49 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 0)
111215 19:01:49 [Note] WSREP: New cluster view: global state: 4c286ccc-2792-11e1-0800-
↪94bd91e32efa:0, view# 1: Primary, number of nodes: 1, my index: 0, protocol version␣
↪1
```

And you can check used ports:

```
netstat -anp | grep mysqld
tcp        0      0 192.168.2.21:4030           0.0.0.0:*                   LISTEN     ␣
↪  21895/mysqld
tcp        0      0 0.0.0.0:4000                0.0.0.0:*                   LISTEN     ␣
↪  21895/mysqld
```

After first node, we start second and third:

```
bin/mysqld_safe --defaults-file=/etc/my.5000.cnf
bin/mysqld_safe --defaults-file=/etc/my.6000.cnf
```

Successful start will produce the following output:

```
111215 19:22:26 [Note] WSREP: Shifting JOINER -> JOINED (TO: 2)
111215 19:22:26 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 2)
111215 19:22:26 [Note] WSREP: Synchronized with group, ready for connections
```

Cluster size can be checked with the:

```
mysql -h127.0.0.1 -P6000 -e "show global status like 'wsrep_cluster_size';"
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
```

Now you can connect to any node and create database, which will be automatically propagated to other nodes:

```
mysql -h127.0.0.1 -P5000 -e "CREATE DATABASE hello_peter"
```

Configuration files (/etc/my.4000.cnf):

```
/etc/my.4000.cnf

[mysqld]
port = 4000
socket=/tmp/mysql.4000.sock
datadir=/data/bench/d1
basedir=/usr/local/Percona-XtraDB-Cluster-5.6.15-25.3.706.Linux.x86_64
user=mysql
log_error=error.log
binlog_format=ROW
wsrep_cluster_address='gcomm://192.168.2.21:5030,192.168.2.21:6030'
wsrep_provider=/usr/local/Percona-XtraDB-Cluster-5.6.15-25.3.706.Linux.x86_64/lib/
↪libgalera_smm.so
wsrep_sst_receive_address=192.168.2.21:4020
wsrep_node_incoming_address=192.168.2.21
wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_provider_options = "gmcast.listen_addr=tcp://192.168.2.21:4030;"
wsrep_sst_method=rsync
wsrep_node_name=node4000
innodb_autoinc_lock_mode=2
```

Configuration files (/etc/my.5000.cnf):

```
/etc/my.5000.cnf

[mysqld]
port = 5000
socket=/tmp/mysql.5000.sock
datadir=/data/bench/d2
basedir=/usr/local/Percona-XtraDB-Cluster-5.6.15-25.3.706.Linux.x86_64
user=mysql
log_error=error.log
binlog_format=ROW
```

```
wsrep_cluster_address='gcomm://192.168.2.21:4030,192.168.2.21:6030'
wsrep_provider=/usr/local/Percona-XtraDB-Cluster-5.6.15-25.3.706.Linux.x86_64/lib/
↪libgalera_smm.so
wsrep_sst_receive_address=192.168.2.21:5020
wsrep_node_incoming_address=192.168.2.21
wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_provider_options = "gmcast.listen_addr=tcp://192.168.2.21:5030;"
wsrep_sst_method=rsync
wsrep_node_name=node5000
innodb_autoinc_lock_mode=2
```

Configuration files (/etc/my.6000.cnf):

```
/etc/my.6000.cnf

[mysqld]
port = 6000
socket=/tmp/mysql.6000.sock
datadir=/data/bench/d3
basedir=/usr/local/Percona-XtraDB-Cluster-5.6.15-25.3.706.Linux.x86_64
user=mysql
log_error=error.log
binlog_format=ROW
wsrep_cluster_address='gcomm://192.168.2.21:4030,192.168.2.21:5030'
wsrep_provider=/usr/local/Percona-XtraDB-Cluster-5.6.15-25.3.706.Linux.x86_64/lib/
↪libgalera_smm.so
wsrep_sst_receive_address=192.168.2.21:6020
wsrep_node_incoming_address=192.168.2.21
wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_provider_options = "gmcast.listen_addr=tcp://192.168.2.21:6030;"
wsrep_sst_method=rsync
wsrep_node_name=node6000
innodb_autoinc_lock_mode=2
```

CHAPTER

TWENTYONE

HOW TO SETUP 3 NODE CLUSTER IN EC2 ENVIROMENT

This is how to setup 3-node cluster in EC2 enviroment.

Assume you are running *m1.xlarge* instances with OS *Red Hat Enterprise Linux 6.1 64-bit*. Make sure to remove existing PXC-5.5 and PS-5.5/5.6 packages before proceeding.

Install *Percona XtraDB Cluster* from RPM:

1. Install Percona's regular and testing repositories:

```
rpm -Uhv http://repo.percona.com/testing/centos/6/os/noarch/percona-testing-0.0-1.
↪noarch.rpm
rpm -Uhv http://www.percona.com/downloads/percona-release/percona-release-0.0-1.
↪x86_64.rpm
```

2. Install Percona XtraDB Cluster packages:

```
yum install Percona-XtraDB-Cluster-server-56 Percona-XtraDB-Cluster-client-56␣
↪Percona-XtraDB-Cluster-galera-3
```

3. Create data directories:

```
mkdir -p /mnt/data
mysql_install_db --datadir=/mnt/data --user=mysql
```

4. Stop firewall. Cluster requires couple TCP ports to operate. Easiest way:

```
service iptables stop
```

If you want to open only specific ports, you need to open 3306, 4444, 4567, 4568 ports. For example for 4567 port (substitute 192.168.0.1 by your IP):

```
iptables -A INPUT -i eth0 -p tcp -m tcp --source 192.168.0.1/24 --dport 4567 -j ACCEPT
```

5. Create `/etc/my.cnf` files.

   On the first node (assume IP 10.93.46.58):

```
[mysqld]
datadir=/mnt/data
user=mysql

binlog_format=ROW

wsrep_provider=/usr/lib64/libgalera_smm.so
wsrep_cluster_address=gcomm://10.93.46.58,10.93.46.59,10.93.46.60

```

```
wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_sst_method=rsync
wsrep_node_name=node1

innodb_autoinc_lock_mode=2
```

On the second node (assume IP 10.93.46.59):

```
[mysqld]
datadir=/mnt/data
user=mysql

binlog_format=ROW

wsrep_provider=/usr/lib64/libgalera_smm.so
wsrep_cluster_address=gcomm://10.93.46.58,10.93.46.59,10.93.46.60

wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_sst_method=rsync
wsrep_node_name=node2

innodb_autoinc_lock_mode=2
```

On the third (and following nodes) configuration is similar, with the following change:

```
wsrep_node_name=node3
```

6. Start the *Percona XtraDB Cluster*

On the first node:

```
[root@node1 ~]# /etc/init.d/mysql bootstrap-pxc
```

You should be able to see in console (or in error-log file):

```
2014-01-30 11:52:35 23280 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.6.15-56'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona XtraDB␣
↪Cluster (GPL), Release 25.3, Revision 706, wsrep_25.3.r4034
```

On the second (and following nodes):

```
[root@node2 ~]# /etc/init.d/mysql start
```

You should be able to see in console (or in error-log file):

```
2014-01-30 09:52:42 26104 [Note] WSREP: Flow-control interval: [28, 28]
2014-01-30 09:52:42 26104 [Note] WSREP: Restored state OPEN -> JOINED (2)
2014-01-30 09:52:42 26104 [Note] WSREP: Member 2 (percona1) synced with group.
2014-01-30 09:52:42 26104 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 2)
2014-01-30 09:52:42 26104 [Note] WSREP: New cluster view: global state: 4827a206-876b-
↪11e3-911c-3e6a77d54953:2, view# 7: Primary, number of nodes: 3, my index: 2,␣
↪protocol version 2
2014-01-30 09:52:42 26104 [Note] WSREP: SST complete, seqno: 2
2014-01-30 09:52:42 26104 [Note] Plugin 'FEDERATED' is disabled.
2014-01-30 09:52:42 26104 [Note] InnoDB: The InnoDB memory heap is disabled
2014-01-30 09:52:42 26104 [Note] InnoDB: Mutexes and rw_locks use GCC atomic builtins
```

```
2014-01-30 09:52:42 26104 [Note] InnoDB: Compressed tables use zlib 1.2.3
2014-01-30 09:52:42 26104 [Note] InnoDB: Using Linux native AIO
2014-01-30 09:52:42 26104 [Note] InnoDB: Not using CPU crc32 instructions
2014-01-30 09:52:42 26104 [Note] InnoDB: Initializing buffer pool, size = 128.0M
2014-01-30 09:52:42 26104 [Note] InnoDB: Completed initialization of buffer pool
2014-01-30 09:52:43 26104 [Note] InnoDB: Highest supported file format is Barracuda.
2014-01-30 09:52:43 26104 [Note] InnoDB: 128 rollback segment(s) are active.
2014-01-30 09:52:43 26104 [Note] InnoDB: Waiting for purge to start
2014-01-30 09:52:43 26104 [Note] InnoDB:  Percona XtraDB (http://www.percona.com) 5.6.
↪15-rel62.0 started; log sequence number 1626341
2014-01-30 09:52:43 26104 [Note] RSA private key file not found: /var/lib/mysql//
↪private_key.pem. Some authentication plugins will not work.
2014-01-30 09:52:43 26104 [Note] RSA public key file not found: /var/lib/mysql//
↪public_key.pem. Some authentication plugins will not work.
2014-01-30 09:52:43 26104 [Note] Server hostname (bind-address): '*'; port: 3306
2014-01-30 09:52:43 26104 [Note] IPv6 is available.
2014-01-30 09:52:43 26104 [Note]   - '::' resolves to '::';
2014-01-30 09:52:43 26104 [Note] Server socket created on IP: '::'.
2014-01-30 09:52:43 26104 [Note] Event Scheduler: Loaded 0 events
2014-01-30 09:52:43 26104 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.6.15-56'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona XtraDB␣
↪Cluster (GPL), Release 25.3, Revision 706, wsrep_25.3.r4034
2014-01-30 09:52:43 26104 [Note] WSREP: inited wsrep sidno 1
2014-01-30 09:52:43 26104 [Note] WSREP: wsrep_notify_cmd is not defined, skipping␣
↪notification.
2014-01-30 09:52:43 26104 [Note] WSREP: REPL Protocols: 5 (3, 1)
2014-01-30 09:52:43 26104 [Note] WSREP: Assign initial position for certification: 2,␣
↪protocol version: 3
2014-01-30 09:52:43 26104 [Note] WSREP: Service thread queue flushed.
2014-01-30 09:52:43 26104 [Note] WSREP: Synchronized with group, ready for connections
```

When all nodes are in SYNCED stage your cluster is ready!

7. Connect to database on any node and create database:

```
$ mysql -uroot
> CREATE DATABASE hello_tom;
```

The new database will be propagated to all nodes.

Enjoy!

CHAPTER TWENTYTWO

ENCRYPTING PXC TRAFFIC

Percona XtraDB Cluster supports encryption for all traffic involved in cluster operation.

Securing Client-Server Communication Traffic

This refers to communication between client applications and cluster nodes. To secure client connections, you need to generate and use SSL keys and certificates.

The following example shows `my.cnf` configuration for server nodes and client instances to use SSL:

```
[mysqld]
ssl-ca=ca.pem
ssl-cert=server-cert.pem
ssl-key=server-key.pem

[client]
ssl-ca=ca.pem
ssl-cert=client-cert.pem
ssl-key=client-key.pem
```

For more information, see the relevant sections about SSL certificates in Galera Cluster Documentation. and MySQL Server Documentation.

Securing SST Traffic

This refers to full data transfer that usually occurs when a new node (JOINER) joins the cluster and receives data from an existing node (DONOR).

For more information, see *State Snapshot Transfer*.

The following SST methods are available: `rsync`, `mysqldump`, and `xtrabackup`.

rsync

This SST method does not support encryption. Avoid using this method if you need to secure traffic between DONOR and JOINER nodes.

mysqldump

This SST method dumps data from DONOR and imports it to JOINER. Encryption in this case is performed using the SSL certificates configured for *secure MySQL client-server communication*.

Here is how to perform secure SST using `mysqldump`:

1. Ensure that the DONOR node is configured for SSL encryption (both `[mysqld]` and `[client]` sections). For more information, see *Securing Client-Server Communication Traffic*.
2. Create an SSL user on the DONOR node.

```
mysql> GRANT USAGE ON *.* TO 'sslsst' REQUIRE SSL;
```

3. Specify the DONOR superuser credentials in the `wsrep_sst_auth` variable.
4. Start the JOINER node without Galera library (`--wsrep_provider=none`) and create an SSL user with the same name and grants as on the DONOR node.
5. Configure SSL encryption on JOINER node with the same parameters as DONOR (both `[mysqld]` and `[client]` sections).
6. Restart JOINER node with Galera library.

If you do everything correctly, `mysqldump` will connect to DONOR using SSL user, generate a dump file, and import it to JOINER node.

For more information, see the relevant section in Galera Cluster documentation.

xtrabackup

This is the default SST method. It uses Percona XtraBackup to perform non-blocking transfer of files. For more information, see *Percona XtraBackup SST Configuration*.

Encryption mode for this method is selected using the `encrypt` option. Depending on the mode you select, other options will be required:

- `encrypt=0` is the default value, meaning that encryption is disabled.
- `encrypt=1` enables built-in XtraBackup encryption.

  Note: This mode has been deprecated.

```
[sst]
encrypt=1
encrypt-algo=AES256
encrypt-key=A1EDC73815467C083B0869508406637E
```

  In this example, you can set `encrypt-key-file` instead of `encrypt-key`.

  For more information, see Encrypted Backups.

- `encrypt=2` enables SST encryption based on OpenSSL with the certificate authority (`tca`) and certificate (`tcert`) files.

  Note: This mode has been deprecated.

  ```
  [sst]
  encrypt=2
  tcert=/path/to/server.pem
  tca=/path/to/server.crt
  ```

  For more information, see *Securing Traffic Between two Socat Instances Using SSL <http://www.dest-unreach.org/socat/doc/socat-openssltunnel.html>*.

- `encrypt=3` enables SST encryption based on OpenSSL with the key (`tkey`) and certificate (`tcert`) files.

  Note: This mode has been deprecated.

  ```
  [sst]
  encrypt=3
  tcert=/path/to/server.pem
  tkey=/path/to/server.key
  ```

  Certificate verification is not performed in this mode, meaning that the validity of the joining nodes is not checked.

- `encrypt=4` enables SST encryption based on SSL files generated by MySQL.

  Note: This is the recommended mode.

  ```
  [sst]
  encrypt=4
  ssl-ca=ca.pem
  ssl-cert=server-cert.pem
  ssl-key=server-key.pem
  ```

  For more information, see http://dev.mysql.com/doc/refman/5.6/en/creating-ssl-files-using-openssl.html

  The following procedure shows how to generate the `ca.pem`, `server-cert.pem`, and `server-key.pem` files.

  1. To generate the CA certificate:

     ```
     openssl genrsa 2048 > ca-key.pem
     openssl req -new -x509 -nodes -days 3600 -key ca-key.pem -out ca.pem
     ```

  2. To generate the server certificate, remove passphrase, and sign it:

     ```
     openssl req -newkey rsa:2048 -days 3600 -nodes -keyout server-key.pem -out␣
     ↪server-req.pem
     openssl rsa -in server-key.pem -out server-key.pem
     openssl x509 -req -in server-req.pem -days 3600 -CA ca.pem -CAkey ca-key.pem -
     ↪set_serial -1 -out server-cert.pem
     ```

  3. (Optional) To generate the client certificate, remove passphrase, and sign it:

```
openssl req -newkey rsa:2048 -days 3600 -nodes -keyout client-key.pem -out␣
↪client-req.pem
openssl rsa -in client-key.pem -out client-key.pem
openssl x509 -req -in client-req.pem -days 3600 -CA ca.pem -CAkey ca-key.pem -
↪set_serial 01 -out client-cert.pem
```

There are two ways to deploy SSL files across the cluster:

– (Preferred) Use the same files for all machines in the configuration. Generate the files on one machine (or use the MySQL-generated files), and copy the files to each node in the cluster. Follow steps (1) and (2) in the previous procedure to manually generate the files.

– (Alternative) Generate one CA file with separate server certificates signed by that one CA file. Do step (1) to generate the CA file and then do step (2) for each server using the same CA file. So each server will have different `server-cert.pem` files, but they will all share the same `ca.pem` file.

Note: Whatever method you use to generate the certificate and key files, the `Common Name` value used for the server and client certificates/keys must each differ from that value used for the CA certificate. Otherwise, the certificate and key files will not work for servers compiled using OpenSSL.

The easiest way to do this is to give the CA file a common name (CN) as follows:

```
openssl req -new -x509 -nodes -days 3600 -key ca-key.pem -out ca.pem -subj "/
↪CN=my_cluster_name"
```

Note: SSL clients require DH parameters to be at least 1024 bits, due to the *logjam vulnearability <https://en.wikipedia.org/wiki/Logjam_(computer_security)>*. However, versions of `socat` earlier than 1.7.3 use 512-bit parameters. If a `dhparams.pem` file of required length is not found during SST in the data directory, it is generated with 2048 bits, which can take several minutes. To avoid this delay, create the `dhparams.pem` file manually and place it in the data directory before joining the node to the cluster:

```
openssl dhparam -out dhparams.pem 2048
```

Securing IST Traffic, Write-Set Replication, and Service Messages

IST refers to transferring only missing transactions from DONOR to JOINER node. Write-set replication is the main workload in *Percona XtraDB Cluster* whenever a transaction is performed on one node, it is replicated to all other nodes. Service messages ensure that all nodes are synchronized.

All of this traffic is transferred via the same underlying communication channel used by Galera (`gcomm`). Securing this channel will ensure that IST traffic, write-set replication, and service messages are encypted.

To enable SSL for all internal node processes, define the paths to the key, certificate and certificate authority files using the following parameters:

- `socket.ssl_key`
- `socket.ssl_cert`
- `socket.ssl_ca`

To set these parameters, use the *`wsrep_provider_options`* variable.

```
wsrep_provider_options="socket.ssl=yes;socket.ssl_key=/path/to/server-key.pem;socket.
↪ssl_cert=/path/to/server-cert.pem;socket.ssl_ca=/path/to/cacert.pem"
```

For more information, see Index of wsrep provider options.

Note: You must use the same server key and certificate files on all nodes, preferably those used for *Securing Client-Server Communication Traffic*.

Upgrading Certificates

The following example shows how to upgrade certificates used for securing IST traffic, write-set replication, and service messages, assumig there are two nodes in the cluster:

1. Restart Node 1 with a `socket.ssl_ca` that includes both the new and the old certificates in a single file.

   For example, you can merge contents of `old-ca.pem` and `new-ca.pem` into `upgrade-ca.pem` as follows:

   ```
   cat old-ca.pem > upgrade-ca.pem && cat new-ca.pem >> upgrade-ca.pem
   ```

   Set the *`wsrep_provider_options`* variable similar to the following:

   ```
   wsrep_provider_options=socket.ssl=yes;socket.ssl_ca=/path/to/upgrade-ca.pem;
   ↪socket.ssl_cert=path/to/old-cert.pem;socket.ssl_key=/path/to/old-key.pem
   ```

2. Restart Node 2 with the new `socket.ssl_ca`, `socket.ssl_cert`, and `socket.ssl_key`.

   ```
   wsrep_provider_options=socket.ssl=yes;socket.ssl_ca=/path/to/upgrade-ca.pem;
   ↪socket.ssl_cert=/path/to/new-cert.pem;socket.ssl_key=/path/to/new-key.pem
   ```

3. Restart Node 1 with the new `socket.ssl_ca`, `socket.ssl_cert`, and `socket.ssl_key`.

   ```
   wsrep_provider_options=socket.ssl=yes;socket.ssl_ca=/path/to/upgrade-ca.pem;
   ↪socket.ssl_cert=/path/to/new-cert.pem;socket.ssl_key=/path/to/new-key.pem
   ```

CHAPTER

TWENTYTHREE

LOAD BALANCING WITH HAPROXY

This section describes how to configure *HAProxy* to work in front of the cluster.

Here is the simple configuration file example for *HAProxy*

```
# this config needs haproxy-1.4.20

global
        log 127.0.0.1   local0
        log 127.0.0.1   local1 notice
        maxconn 4096
        uid 99
        gid 99
        daemon
        #debug
        #quiet

defaults
        log     global
        mode    http
        option  tcplog
        option  dontlognull
        retries 3
        redispatch
        maxconn 2000
        contimeout      5000
        clitimeout      50000
        srvtimeout      50000

listen mysql-cluster 0.0.0.0:3306
    mode tcp
    balance roundrobin
    option mysql-check user root

    server db01 10.4.29.100:3306 check
    server db02 10.4.29.99:3306 check
    server db03 10.4.29.98:3306 check
```

With this configuration *HAProxy* will load balance between three nodes. In this case it only checks if mysqld listens on port 3306, but it doesn't take into an account state of the node. So it could be sending queries to the node that has mysqld running even if it's in "JOINING" or "DISCONNECTED" state.

To check the current status of a node we need a more complex checks. This idea was taken from codership-team google groups.

To implement this setup you will need two scripts:

- **clustercheck** (place to /usr/local/bin) and a config for xinetd and
- **mysqlchk** (place to /etc/xinetd.d) on each node.

Both scripts are available in binaries and source distributions of *Percona XtraDB Cluster*.

You'll need to change /etc/services file by adding the following line on each node:

```
mysqlchk        9200/tcp                # mysqlchk
```

The configuration file for *HAProxy* in this case may look like this:

```
# this config needs haproxy-1.4.20

global
        log 127.0.0.1   local0
        log 127.0.0.1   local1 notice
        maxconn 4096
        uid 99
        gid 99
        #daemon
        debug
        #quiet

defaults
        log     global
        mode    http
        option  tcplog
        option  dontlognull
        retries 3
        redispatch
        maxconn 2000
        contimeout      5000
        clitimeout      50000
        srvtimeout      50000

listen mysql-cluster 0.0.0.0:3306
    mode tcp
    balance roundrobin
    option  httpchk

    server db01 10.4.29.100:3306 check port 9200 inter 12000 rise 3 fall 3
    server db02 10.4.29.99:3306 check port 9200 inter 12000 rise 3 fall 3
    server db03 10.4.29.98:3306 check port 9200 inter 12000 rise 3 fall 3
```

CHAPTER

TWENTYFOUR

SETTING UP PXC REFERENCE ARCHITECTURE WITH HAPROXY

This tutorial is a step-by-step guide to set up *Percona XtraDB Cluster*, in a virtualized test sandbox. This example uses Amazon EC2 micro instances, but the content here is applicable for any kind of virtualization technology (for example VirtualBox). You will need 4 virtual machines. 3 for *Percona XtraDB Cluster* and 1 for the client, which will have *HAProxy*. In this how-to CentOS 6 is used as the operating system, the instructions are similar for any Linux distribution.

The client node will have HAProxy installed and it will redirect requests to *Percona XtraDB Cluster* nodes. This approach works well in real-world scenarios too. Running HAProxy on the application servers instead of having them as dedicated entities gives you benefits like no need for an extra network roundtrip, because loadbalancer and scalability of *Percona XtraDB Cluster*'s load balancing layer scales simply with application servers.

We'll use Percona and EPEL repositories for software installation.

After configuring the repositories you'll be able to install software that will be used. First, install *Percona XtraDB Cluster* on the database nodes.

```
# yum -y install Percona-XtraDB-Cluster-server Percona-XtraDB-Cluster-client percona-
↪xtrabackup
```

Install *HAProxy* and *sysbench* on the client node.

```
# yum -y install haproxy sysbench
```

After installing everything, we'll configure *Percona XtraDB Cluster* first. On the first node, `my.cnf` should look something like this on a relatively weak machine.

```
[mysqld]
server_id=1
binlog_format=ROW
log_bin=mysql-bin
wsrep_cluster_address=gcomm://
wsrep_provider=/usr/lib/libgalera_smm.so
datadir=/var/lib/mysql

wsrep_slave_threads=2
wsrep_cluster_name=pxctest
wsrep_sst_method=xtrabackup
wsrep_node_name=ip-10-112-39-98

log_slave_updates

innodb_autoinc_lock_mode=2
innodb_buffer_pool_size=400M
innodb_log_file_size=64M
```

You can start your first node now. Make sure that you only start second and third nodes when the first node is up and running (it will serve as a donor for *SST*).

This configuration is for the first node. For the second and third node, you need to change `wsrep_cluster_address` (alternatively, you can use `wsrep_urls` in [mysqld_safe] section), which should point to a node in the cluster which is already up, so it will join the cluster. The `server_id` and `wsrep_node_name` variables have to be different on each host, for `wsrep_node_name`, you can use the output of *hostname* command.

Based on that, for the second node, the differences in the configuration should be the following.

```
server_id=2
wsrep_cluster_address=gcomm://10.116.39.76 # replace this with the IP of your first␣
↪node
wsrep_node_name=ip-10-244-33-92
```

For the third node, the differences look like this.

```
server_id=3
wsrep_cluster_address=gcomm://10.116.39.76 # replace this with the IP of your first␣
↪node
wsrep_node_name=ip-10-194-10-179
```

For *SST* we use **xtrabackup**. This means at startup time, the new node will connect to an existing node in the cluster and it takes a backup of that node with xtrabackup and copies it to the new node with *netcat*. After a successful *SST*, you should see this in the error log.

```
120619 13:20:17 [Note] WSREP: State transfer required:
      Group state: 77c9da88-b965-11e1-0800-ea53b7b12451:97
      Local state: 00000000-0000-0000-0000-000000000000:-1
120619 13:20:17 [Note] WSREP: New cluster view: global state: 77c9da88-b965-11e1-0800-
↪ea53b7b12451:97, view# 18: Primary, number of nodes: 3, my index: 0, protocol␣
↪version 2
120619 13:20:17 [Warning] WSREP: Gap in state sequence. Need state transfer.
120619 13:20:19 [Note] WSREP: Running: 'wsrep_sst_xtrabackup 'joiner' '10.195.206.117
↪' '' '/var/lib/mysql/' '/etc/my.cnf' '20758' 2>sst.err'
120619 13:20:19 [Note] WSREP: Prepared |SST| request: xtrabackup|10.195.206.117:4444/
↪xtrabackup_sst
120619 13:20:19 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
120619 13:20:19 [Note] WSREP: Assign initial position for certification: 97, protocol␣
↪version: 2
120619 13:20:19 [Warning] WSREP: Failed to prepare for incremental state transfer:␣
↪Local state UUID (00000000-0000-0000-0000-000000000000) does not match group state␣
↪UUID (77c9da88-b965-11e1-0800-ea53b7b12451): 1 (Operation not permitted)
       at galera/src/replicator_str.cpp:prepare_for_IST():439. IST will be␣
↪unavailable.
120619 13:20:19 [Note] WSREP: Node 0 (ip-10-244-33-92) requested state transfer from
↪'*any*'. Selected 1 (ip-10-112-39-98)(SYNCED) as donor.
120619 13:20:19 [Note] WSREP: Shifting PRIMARY -> JOINER (TO: 102)
120619 13:20:19 [Note] WSREP: Requesting state transfer: success, donor: 1
120619 13:20:59 [Note] WSREP: 1 (ip-10-112-39-98): State transfer to 0 (ip-10-244-33-
↪92) complete.
120619 13:20:59 [Note] WSREP: Member 1 (ip-10-112-39-98) synced with group.
120619 13:21:17 [Note] WSREP: |SST| complete, seqno: 105
120619 13:21:17 [Note] Plugin 'FEDERATED' is disabled.
120619 13:21:17 InnoDB: The InnoDB memory heap is disabled
120619 13:21:17 InnoDB: Mutexes and rw_locks use GCC atomic builtins
120619 13:21:17 InnoDB: Compressed tables use zlib 1.2.3
120619 13:21:17 InnoDB: Using Linux native AIO
```

```
120619 13:21:17 InnoDB: Initializing buffer pool, size = 400.0M
120619 13:21:17 InnoDB: Completed initialization of buffer pool
120619 13:21:18 InnoDB: highest supported file format is Barracuda.
120619 13:21:18  InnoDB: Waiting for the background threads to start
120619 13:21:19 Percona XtraDB (http://www.percona.com) 1.1.8-rel25.3 started; log␣
↪sequence number 246661644
120619 13:21:19 [Note] Recovering after a crash using mysql-bin
120619 13:21:19 [Note] Starting crash recovery...
120619 13:21:19 [Note] Crash recovery finished.
120619 13:21:19 [Note] Server hostname (bind-address): '(null)'; port: 3306
120619 13:21:19 [Note]   - '(null)' resolves to '0.0.0.0';
120619 13:21:19 [Note]   - '(null)' resolves to '::';
120619 13:21:19 [Note] Server socket created on IP: '0.0.0.0'.
120619 13:21:19 [Note] Event Scheduler: Loaded 0 events
120619 13:21:19 [Note] WSREP: Signalling provider to continue.
120619 13:21:19 [Note] WSREP: Received |SST|: 77c9da88-b965-11e1-0800-ea53b7b12451:105
120619 13:21:19 [Note] WSREP: |SST| received: 77c9da88-b965-11e1-0800-ea53b7b12451:105
120619 13:21:19 [Note] WSREP: 0 (ip-10-244-33-92): State transfer from 1 (ip-10-112-
↪39-98) complete.
120619 13:21:19 [Note] WSREP: Shifting JOINER -> JOINED (TO: 105)
120619 13:21:19 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.5.24-log'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona␣
↪XtraDB Cluster (GPL), wsrep_23.6.r340
120619 13:21:19 [Note] WSREP: Member 0 (ip-10-244-33-92) synced with group.
120619 13:21:19 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 105)
120619 13:21:20 [Note] WSREP: Synchronized with group, ready for connections
```

For debugging information about the *SST*, you can check the sst.err file and the error log too.

After the SST's is done, you should check if you have a 3 node cluster.

```
mysql> show global status like 'wsrep_cluster_size';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
1 row in set (0.00 sec)
```

When all nodes are started, you can set up HAProxy on the client. The point of this is that the application will be able to connect to localhost as *MySQL* server, so although we are using *Percona XtraDB Cluster*, the application will see this as a single MySQL server running on localhost.

In order to achieve this, you'll need to configure HAProxy on the client node. There are 2 possible configurations here. First is configuring round robin, which means you will connect and write to all cluster nodes. This can be done, but because of optimistic locking at commit time, rollbacks can happen if you have conflicting writes. In the second configuration, you will configure HAProxy in a way that it writes only to one node, so the application doesn't have to be prepared about unexpected rollbacks. The first configuration is a good choice in most cases, not handling rollbacks is not healthy in a well behaving application anyway.

HAProxy can be configured in the /etc/haproxy/haproxy.cfg and it should look like this.

```
global
log 127.0.0.1 local0
log 127.0.0.1 local1 notice
maxconn 4096
chroot /usr/share/haproxy
user haproxy
```

```
group haproxy
daemon

defaults
log global
mode http
option tcplog
option dontlognull
retries 3
option redispatch
maxconn 2000
contimeout 5000
clitimeout 50000
srvtimeout 50000

frontend pxc-front
bind *:3307
mode tcp
default_backend pxc-back

frontend stats-front
bind *:80
mode http
default_backend stats-back

frontend pxc-onenode-front
bind *:3306
mode tcp
default_backend pxc-onenode-back

backend pxc-back
mode tcp
balance leastconn
option httpchk
server c1 10.116.39.76:3306 check port 9200 inter 12000 rise 3 fall 3
server c2 10.195.206.117:3306 check port 9200 inter 12000 rise 3 fall 3
server c3 10.202.23.92:3306 check port 9200 inter 12000 rise 3 fall 3

backend stats-back
mode http
balance roundrobin
stats uri /haproxy/stats
stats auth pxcstats:secret

backend pxc-onenode-back
mode tcp
balance leastconn
option httpchk
server c1 10.116.39.76:3306 check port 9200 inter 12000 rise 3 fall 3
server c2 10.195.206.117:3306 check port 9200 inter 12000 rise 3 fall 3 backup
server c3 10.202.23.92:3306 check port 9200 inter 12000 rise 3 fall 3 backup
```

In this configuration, three frontend-backend pairs are defined. The stats pair is for *HAProxy* statistics page, and the others are for *Percona XtraDB Cluster*. *MySQL* will be listening on ports 3306 and 3307. If you connect to port 3306, you'll connect to *pxc-onenode*, and you'll be only using one node at a time (to avoid rollbacks because of optimistic locking). If that node goes off-line, you'll start using an other one. However if you connect to port 3307, you'll be using all three nodes for reads and writes too. In this case the *leastconn* load balancing method is used instead of round robin, which means you always connect to the backend with the least connections established. The statistics page is accessible on the client node with a browser pointed to */haproxy/stats*, the stats auth parameter in the configuration has the credentials for that in plain text. You can also use this for monitoring purposes (the CSV version is good for trending and alerting).

Here *MySQL* is checked via HTTP checks. *MySQL* won't serve these requests. As part of *Percona XtraDB Cluster* packages, we distribute the clustercheck utility which has to be set up. After that, HAProxy will be able to use check *MySQL* via HTTP. The clustercheck script is a simple shell script, which accepts HTTP requests, and checks MySQL on incoming request. If the *Percona XtraDB Cluster* node is ok, it will emit a response with HTTP code 200 OK, otherwise, it emits 503. The script examines `wsrep_local_state` variable.

To set it up, create the clustercheck user.

```
mysql> grant process on *.* to 'clustercheckuser'@'localhost' identified by
↪'clustercheckpassword!';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

If you want to use a different username or password, you have to modify them in the script too. Let's test.

```
# clustercheck
HTTP/1.1 200 OK

Content-Type: Content-Type: text/plain
```

Node is running.

You can use *xinetd* to daemonize the script. If *xinetd* is not installed yet, you can install it with yum.

```
# yum -y install xinetd
```

The service itself should be configured in `/etc/xinetd.d/mysqlchk`.

```
# default: on
# description: mysqlchk
service mysqlchk
{
# this is a config for xinetd, place it in /etc/xinetd.d/
  disable = no
  flags = REUSE
  socket_type = stream
  port = 9200
  wait = no
  user = nobody
  server = /usr/bin/clustercheck
  log_on_failure += USERID
  only_from = 0.0.0.0/0
  # recommended to put the IPs that need
  # to connect exclusively (security purposes)
  per_source = UNLIMITED
}
```

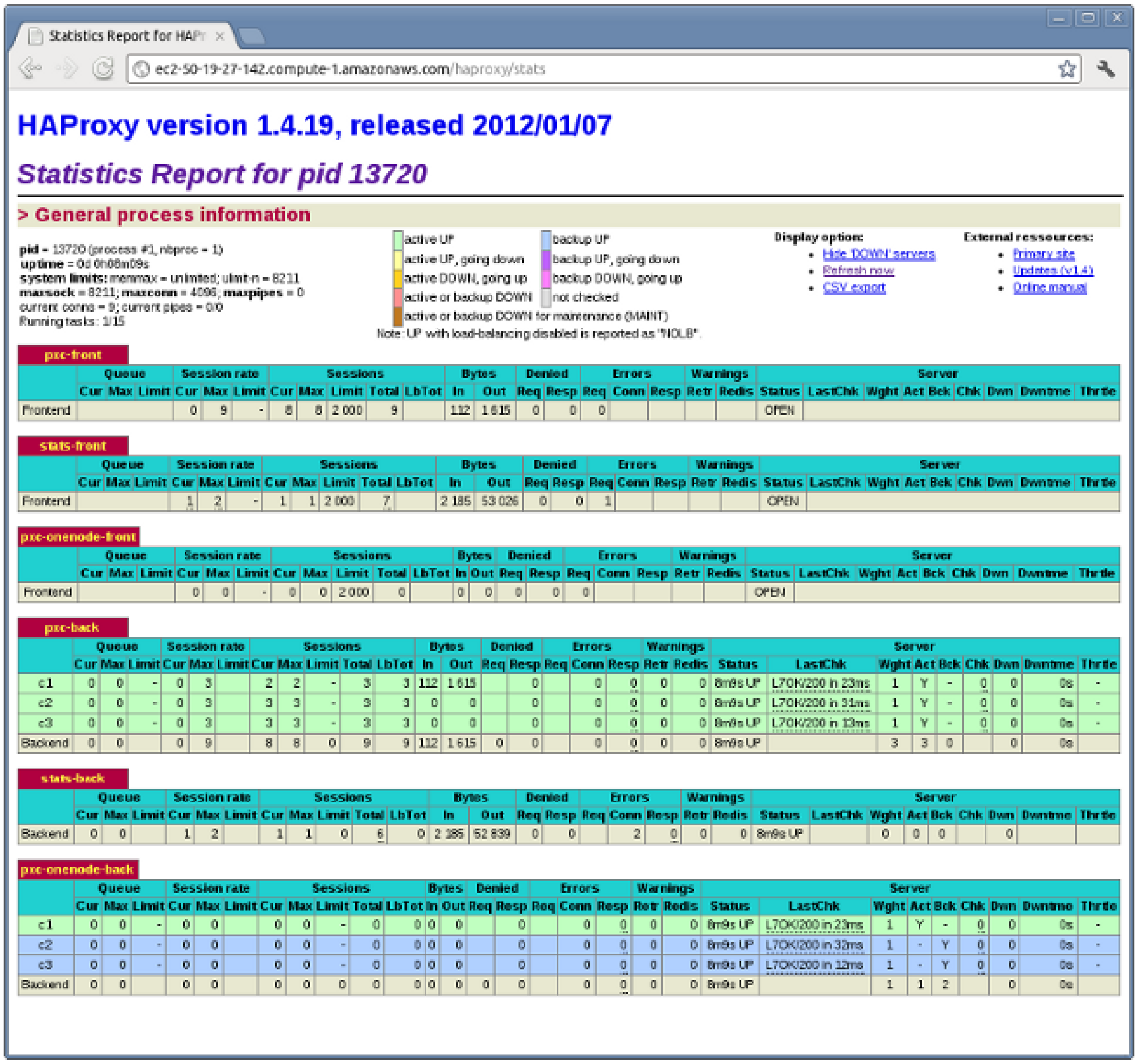
Also, you should add the new service to `/etc/services`.

```
mysqlchk 9200/tcp # mysqlchk
```

Clustercheck will now listen on port 9200 after xinetd restart, and *HAProxy* is ready to check *MySQL* via HTTP.

```
# service xinetd restart
```

If you did everything right so far, the statistics page of *HAProxy* should look like this.



Statistics Report for HAP

ec2-50-19-27-142.compute-1.amazonaws.com/haproxy/stats

HAProxy version 1.4.19, released 2012/01/07

Statistics Report for pid 13720

> General process information

pid = 13720 (process #1, nbproc = 1)
uptime = 0d 0h08m09s
system limits: memmax = unlimited; ulimit-n = 8211
maxsock = 8211; **maxconn** = 4096; **maxpipes** = 0
current conns = 9; current pipes = 0/0
Running tasks: 1/15

🟩	active UP	🟦	backup UP
🟨	active UP, going down	🟪	backup UP, going down
🟧	active DOWN, going up	🟪	backup DOWN, going up
🟥	active or backup DOWN	⬜	not checked
🟫	active or backup DOWN for maintenance (MAINT)		

Note: UP with load-balancing disabled is reported as "NOLB".

Display option:

- Hide 'DOWN' servers
- Refresh now
- CSV export

External ressources:

- Primary site
- Updates (v1.4)
- Online manual

pxc-front

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	9	-	8	8	2 000	9		112	1 615	0	0	0					OPEN								

stats-front

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				1	2	-	1	1	2 000	7		2 185	53 026	0	0	1					OPEN								

pxc-onenode-front

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	0	-	0	0	2 000	0		0	0	0	0	0					OPEN								

pxc-back

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
c1	0	0	-	0	3		2	2	-	3	3	112	1 615		0		0	0	0	0	8m9s UP	L7OK/200 in 23ms	1	Y	-	0	0	0s	-
c2	0	0	-	0	3		3	3	-	3	3	0	0		0		0	0	0	0	8m9s UP	L7OK/200 in 31ms	1	Y	-	0	0	0s	-
c3	0	0	-	0	3		3	3	-	3	3	0	0		0		0	0	0	0	8m9s UP	L7OK/200 in 13ms	1	Y	-	0	0	0s	-
Backend	0	0		0	9		8	8	0	9	9	112	1 615	0	0		0	0	0	0	8m9s UP		3	3	0		0	0s	

stats-back

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Backend	0	0		1	2		1	1	0	6	0	2 185	52 839	0	0		2	0	0	0	8m9s UP		0	0	0		0		

pxc-onenode-back

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
c1	0	0	-	0	0		0	0	-	0	0	0	0		0		0	0	0	0	8m9s UP	L7OK/200 in 23ms	1	Y	-	0	0	0s	-
c2	0	0	-	0	0		0	0	-	0	0	0	0		0		0	0	0	0	8m9s UP	L7OK/200 in 32ms	1	-	Y	0	0	0s	-
c3	0	0	-	0	0		0	0	-	0	0	0	0		0		0	0	0	0	8m9s UP	L7OK/200 in 12ms	1	-	Y	0	0	0s	-
Backend	0	0		0	0		0	0	0	0	0	0	0	0	0		0	0	0	0	8m9s UP		1	1	2		0	0s	

Testing the cluster with sysbench

You can test the cluster using the sysbench (this example uses one from the EPEL repository). First, you need to create a database and a user for it.

```
mysql> create database sbtest;
Query OK, 1 row affected (0.01 sec)

mysql> grant all on sbtest.* to 'sbtest'@'%' identified by 'sbpass';
Query OK, 0 rows affected (0.00 sec)
```

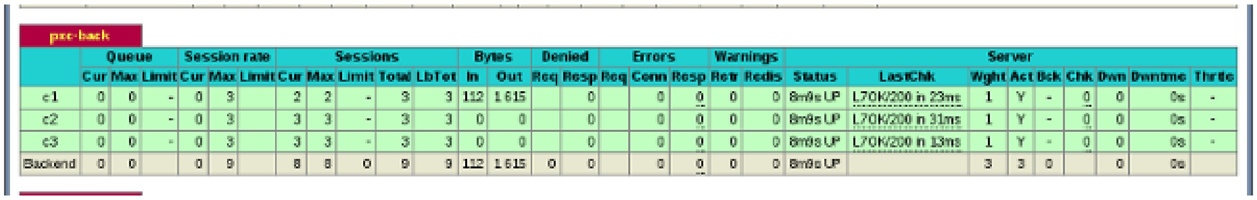
```
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

Populate the table with data for the benchmark.

```
# sysbench --test=oltp --db-driver=mysql --mysql-engine-trx=yes --mysql-table-
↪engine=innodb --mysql-host=127.0.0.1 --mysql-port=3307 --mysql-user=sbtest --mysql-
↪password=sbpass --oltp-table-size=10000 prepare
```

You can now run the benchmark against the 3307 port.

```
# sysbench --test=oltp --db-driver=mysql --mysql-engine-trx=yes --mysql-table-
↪engine=innodb --mysql-host=127.0.0.1 --mysql-port=3307 --mysql-user=sbtest --mysql-
↪password=sbpass --oltp-table-size=10000 --num-threads=8 run
```



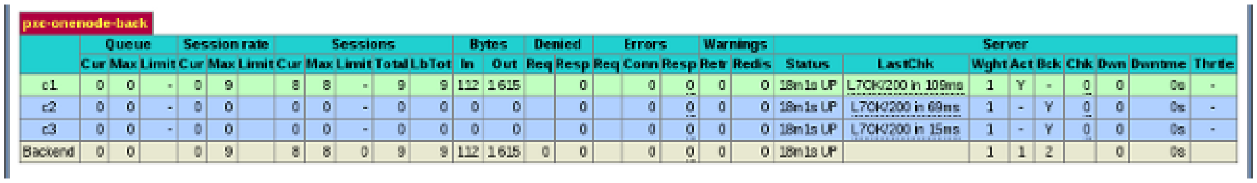
pxc-back

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
c1	0	0	-	0	3		2	2	-	3	3	112	1615		0		0	0	0	0	8m9s UP	L7OK/200 in 23ms	1	Y	-	0	0	0s	-
c2	0	0	-	0	3		3	3	-	3	3	0	0		0		0	0	0	0	8m9s UP	L7OK/200 in 31ms	1	Y	-	0	0	0s	-
c3	0	0	-	0	3		3	3	-	3	3	0	0		0		0	0	0	0	8m9s UP	L7OK/200 in 13ms	1	Y	-	0	0	0s	-
Backend	0	0		0	9		8	8	0	9	9	112	1615	0	0		0	0	0	0	8m9s UP		3	3	0		0	0s	

This is the status of *pxc-back backend* while the *sysbench* above is running. If you look at Cur column under Session, you can see, that c1 has 2 threads connected, c2 and c3 has 3.

If you run the same benchmark, but against the 3306 backend, *HAProxy* stats will show us that the all the threads are going to hit the c1 server.

```
# sysbench --test=oltp --db-driver=mysql --mysql-engine-trx=yes --mysql-table-
↪engine=innodb --mysql-host=127.0.0.1 --mysql-port=3306 --mysql-user=sbtest --mysql-
↪password=sbpass --oltp-table-size=10000 --num-threads=8 run
```



pxc-onenode-back

	Queue			Session rate			Sessions					Bytes		Denied		Errors			Warnings		Server							
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
c1	0	0	-	0	9		8	8	-	9	9	112	1615		0		0	0	0	0	18m1s UP	L7OK/200 in 109ms	1	Y	-	0	0	0s	-
c2	0	0	-	0	0		0	0	-	0	0	0	0		0		0	0	0	0	18m1s UP	L7OK/200 in 69ms	1	-	Y	0	0	0s	-
c3	0	0	-	0	0		0	0	-	0	0	0	0		0		0	0	0	0	18m1s UP	L7OK/200 in 15ms	1	-	Y	0	0	0s	-
Backend	0	0		0	9		8	8	0	9	9	112	1615	0	0		0	0	0	0	18m1s UP		1	1	2		0	0s	

This is the status of *pxc-onenode-back* while *sysbench* above is running. Here only c1 has 8 connected threads, c2 and c3 are acting as backup nodes.

If you are using *HAProxy* for *MySQL* you can break the privilege system's host part, because *MySQL* will think that the connections are always coming from the load balancer. You can work this around using T-Proxy patches and some *iptables* magic for the backwards connections. However in the setup described in this how-to this is not an issue, since each application server has it's own *HAProxy* instance, each application server connects to 127.0.0.1, so MySQL will see that connections are coming from the application servers. Just like in the normal case.

CHAPTER

TWENTYFIVE

HOW TO REPORT BUGS

All bugs can be reported on jira.percona.com. Please note that error.log files from **all** the nodes need to be submitted.

Part VI

Reference

CHAPTER

TWENTYSIX

PERCONA XTRADB CLUSTER 5.6 RELEASE NOTES

Percona XtraDB Cluster 5.6.51-28.46

Date March 22, 2021

Installation Installing Percona XtraDB Cluster

This release fixes security vulnerability CVE-2021-27928, a similar issue to CVE-2020-15180.

Bugs Fixed

- PXC-3565: Correct Performance of SELECT in PXC

Percona XtraDB Cluster 5.6.50-28.44

Date January 5, 2021

Installation Installing Percona XtraDB Cluster

Bugs Fixed

- PXC-3442: Fix crash when log_slave_updates=ON and consistency check statement is executed
- PXC-3424: Fix error handling when the donor is not able to serve SST

Percona XtraDB Cluster 5.6.49-28.42.3

Date October 22, 2020

Installation Installing Percona XtraDB Cluster

Bugs Fixed

- PXC-3456: Allow specific characters in SST method names and SST request data.

Percona XtraDB Cluster 5.6.49-28.42.2

Date October 9, 2020

Installation Installing Percona XtraDB Cluster

This release fixes the security vulnerability CVE-2020-15180

Percona XtraDB Cluster 5.6.49-28.42

Date September 15, 2020

Installation Installing Percona XtraDB Cluster

Improvements

- PXC-2187: Enhance SST documentation to include warning about use of command line parameters

Bugs Fixed

- PXC-3243: Modify the BF-abort process to propagate and abort and retry the Stored Procedure instead of the statement

Percona XtraDB Cluster 5.6.48-28.40

Date June 8, 2020

Installation Installing Percona XtraDB Cluster

This release is based on Percona Server 5.6.48-88.0

All Percona software is open-source and free.

Percona XtraDB Cluster 5.6.47-28.40

Date May 4, 2020

Installation Installing Percona XtraDB Cluster

Improvements

- PXC-2197: Modified the SST Documentation to Include Package Dependencies for Percona XtraBackup (PXB).
- PXC-2602: Added the ability to configure xbstream options with wsrep_sst_xtrabackup.

Bugs Fixed

- PXC-2954: DDL to add FK on one node fails but completes on other nodes causing inconsistency
- PXC-2904: Ensured the "Percona-XtraDB-Cluster-57" yum package installed the required xtrabackup version.
- PXC-2684: Modified error handling to prevent deadlock when stored procedure was aborted.
- PXC-2912: Modified the netcat Configuration to Include the -N option, which is required by more recent versions of netcat. The option allows the shutdown of the network socket after the input EOF.

Percona XtraDB Cluster 5.6.46-28.38

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.46-28.38 on December 4, 2019. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.46-28.38 is now the current release, based on the following:

- Percona Server 5.6.46
- Codership WSREP API release 5.6.46
- Codership Galera library 3.28

All Percona software is open-source and free.

Bugs Fixed

- PXC-2729: A cluster node could hang when trying to access a table which was being updated by another node.
- PXC-2704: After a row was updated with a variable-length unique key, the entire cluster could crash.

Percona XtraDB Cluster 5.6.45-28.36

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.45-28.36 on September 17, 2019. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.45-28.36 is now the current release, based on the following:

- Percona Server 5.6.45
- Codership WSREP API release 5.6.44
- Codership Galera library 3.28

All Percona software is open-source and free.

Bugs Fixed

- PXC-2432: PXC was not updating the information_schema user/client statistics properly.
- PXC-2555: SST initialization delay: fixed a bug where the SST process took too long to detect if a child process was running.

Percona XtraDB Cluster 5.6.44-28.34

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.44-28.34 on June 19, 2019. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.44-28.34 is now the current release, based on the following:

- Percona Server 5.6.44
- Codership WSREP API release 5.6.43
- Codership Galera library 3.26

All Percona software is open-source and free.

Bugs Fixed

- PXC-2480: In some cases, *Percona XtraDB Cluster* could not replicate `CURRENT_USER()` used in the `ALTER` statement. `USER()` and `CURRENT_USER()` are no longer allowed in any `ALTER` statement since they fail when replicated.
- PXC-2487: The case when a DDL or DML action was in progress from one client and the provider was updated from another client could result in a race condition.
- PXC-2490: *Percona XtraDB Cluster* could crash when `binlog_space_limit` was set to a value other than zero during `wsrep_recover` mode.
- PXC-2497: The user can set the preferred donor by setting the *`wsrep_sst_donor`* variable. An IP address is not valid as the value of this variable. If the user still used an IP address, an error message was produced that did not provide sufficient information. The error message has been improved to suggest that the user check the value of the *`wsrep_sst_donor`* for an IP address.

Percona XtraDB Cluster 5.6.43-28.32

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.43-28.32 on February 28, 2019. Binaries are available from the downloads section or from our *software repositories*.

This release of *Percona XtraDB Cluster* includes the support of Ubuntu 18.10 (Cosmic Cuttlefish). *Percona XtraDB Cluster* 5.6.43-28.32 is now the current release, based on the following:

- Percona Server 5.6.43-84.3
- Codership WSREP API release 5.6.42
- Codership Galera library 3.25

All Percona software is open-source and free.

Bugs Fixed

- PXC-2388: In some cases, `DROP FUNCTION` with an explicit name was not replicated.

Percona XtraDB Cluster 5.6.42-28.30

Percona is glad to announce the release of Percona XtraDB Cluster 5.6.42-28.30 on January 4, 2019. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.42-28.30 is now the current release, based on the following:

- Percona Server 5.6.42
- Codership WSREP API release 5.6.42
- Codership Galera library 3.25

All Percona software is open-source and free.

Fixed Bugs

- PXC-2281: Debug symbols were missing in Debian dbg packages.
- PXC-2220: Starting two instances of *Percona XtraDB Cluster* on the same node could cause writing transactions to a page store instead of a galera.cache ring buffer, resulting in huge memory consumption because of retaining already applied write-sets.
- PXC-2230: `gcs.fc_limit=0` not allowed as dynamic setting to avoid generating flow control on every message was still possible in `my.cnf` due to the inconsistent check.
- PXC-2238: setting `read_only=1` caused race condition.

Percona XtraDB Cluster 5.6.41-28.28

Percona is glad to announce the release of Percona XtraDB Cluster 5.6.41-28.28 on September 18, 2018. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.41-28.28 is now the current release, based on the following:

- Percona Server 5.6.41
- Codership WSREP API release 5.6.41
- Codership Galera library 3.24

All Percona software is open-source and free.

Fixed Bugs

- PXC-1017: Memcached API is now disabled if node is acting as a cluster node, because InnoDB Memcached access is not replicated by Galera.
- PXC-2164: SST script compatibility with SELinux was improved by forcing it to look for port associated with the said process only.
- PXC-2155: Temporary folders created during SST execution are now deleted on cleanup.
- PXC-2199: TOI replication protocol was fixed to prevent unexpected GTID generation caused by the `DROP TRIGGER IF EXISTS` statement logged by MySQL as a successful one due to its `IF EXISTS` clause.

Percona XtraDB Cluster 5.6.40-26.25

Percona is glad to announce the release of Percona XtraDB Cluster 5.6.40-26.25 on June 20, 2018. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.40-26.25 is now the current release, based on the following:

- Percona Server 5.6.40
- Galera/Codership WSREP API Release 5.6.39
- Galera Replication library 3.23

All Percona software is open-source and free.

New feature

- PXC-907: New variable `wsrep_RSU_commit_timeout` allows to configure RSU wait for active commit connection timeout (in microseconds).

Fixed Bugs

- PXC-2128: Duplicated auto-increment values were set for the concurrent sessions on cluster reconfiguration due to the erroneous readjustment.
- PXC-2059: Error message about the necessity of the `SUPER` privilege appearing in case of the `CREATE TRIGGER` statements fail due to enabled WSREP was made more clear.
- PXC-2091: Check for the maximum number of rows, that can be replicated as a part of a single transaction because of the Galera limit, was enforced even when replication was disabled with `wsrep_on=OFF`.
- PXC-2103: Interruption of the local running transaction in a `COMMIT` state by a replicated background transaction while waiting for the binlog backup protection caused the commit fail and, eventually, an assert in Galera.
- PXC-2130: *Percona XtraDB Cluster* failed to build with Python 3.

Percona XtraDB Cluster 5.6.39-26.25

Percona is glad to announce the release of Percona XtraDB Cluster 5.6.39-26.25 on February 26, 2018. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.39-26.25 is now the current release, based on the following:

- Percona Server 5.6.39
- Galera/Codership WSREP API Release 5.6.39
- Galera Replication library 3.23

Starting from this release, Percona XtraDB Cluster issue tracking system was moved from launchpad to JIRA. All Percona software is open-source and free.

Fixed Bugs

- PXC-904: Replication filters were not working with account management statements like `CREATE USER` in case of galera replication; as a result such commands were blocked by the replication filters on async slave nodes but not on galera ones.
- PXC-2043: SST script was trying to use `pv` (the pipe viewer) for *`progress`* and *`rlimit`* options even on nodes with no `pv` installed, resulting in SST fail instead of just ignoring these options for inappropriate nodes.
- PXC-911: When node's own IP address was defined in the *`wsrep_cluster_address`* variable, the node was receiving "no messages seen in" warnings from it's own IP address in the info log.

This release also contains fixes for the following CVE issues: CVE-2018-2562, CVE-2018-2573, CVE-2018-2583, CVE-2018-2590, CVE-2018-2591, CVE-2018-2612, CVE-2018-2622, CVE-2018-2640, CVE-2018-2645, CVE-2018-2647, CVE-2018-2665, CVE-2018-2668, CVE-2018-2696, CVE-2018-2703, CVE-2017-3737.

Percona XtraDB Cluster 5.6.38-26.23

Percona is glad to announce the release of Percona XtraDB Cluster 5.6.38-26.23 on January 24, 2018. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.38-26.23 is now the current release, based on the following:

- Percona Server 5.6.38
- Galera/Codership WSREP API Release 5.6.38
- Galera Replication library 3.21

All Percona software is open-source and free.

Fixed Bugs

- PXC-889: Fixed an issue where a node with an invalid value for *`wsrep_provider`* was allowed to start up and operate in standalone mode, which could lead to data inconsistency. The node will now abort in this case. Bug fixed #1728774
- PXC-850: ensured that a node, because of data inconsistency, isolates itself before leaving the cluster, thus allowing pending nodes to re-evaluate the quorum. Bug fixed #1704404
- PXC-875: Fixed an issue where toggling *`wsrep_provider`* off and on failed to reset some internal variables and resulted in PXC logging an "Unsupported protocol downgrade" warning. Bug fixed #1379204
- PXC-883: Fixed `ROLLBACK TO SAVEPOINT` incorrect operation on slaves by avoiding useless wsrep plugin register for a savepoint rollback. Bug fixed #1700593
- PXC-887: gcache .page files were unnecessarily created due to an error in projecting gcache free size when configured to recover on restart.
- PXC-895: fixed transaction loss after recovery by avoiding interruption of the binlog recovery based on wsrep saved position. Bug fixed #1734113
- PXC-897: fixed empty `gtid_executed` variable after recovering the position of a node with `--wsrep_recover`.
- PXC-906: fixed certification failure in the case of a node restarting at the same time when frequent `TRUNCATE TABLE` commands and DML writes occur simultaneously on other nodes. Bug fixed #1737731

- PXC-914: Suppressing DDL/TOI replication in case of `sql_log_bin` zero value didn't work when DDL statement was modifying an existing table, resulting in an error.

Percona XtraDB Cluster 5.6.37-26.21-3

Percona is glad to announce the release of Percona XtraDB Cluster 5.6.37-26.21-3 on October 27, 2017. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.37-26.21-3 is now the current release, based on the following:

- Percona Server 5.6.37-82.2
- Galera Replication library 3.21
- wsrep API version 26

Fixed Bugs

- Added access checks for DDL commands to make sure they do not get replicated if they failed without proper permissions. Previously, when a user tried to perform certain DDL actions that failed locally due to lack of privileges, the command could still be replicated to other nodes, because access checks were performed after replication.

  This vulnerability is identified as CVE-2017-15365.

Percona XtraDB Cluster 5.6.37-26.21

Percona is glad to announce the release of Percona XtraDB Cluster 5.6.37-26.21 on September 20, 2017. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.37-26.21 is now the current release, based on the following:

- Percona Server 5.6.37-82.2
- Galera Replication library 3.21
- wsrep API version 26

All Percona software is open-source and free. Details of this release can be found in the 5.6.37-26.21 milestone on Launchpad.

Improvements

- PXC-851: Added version compatibility check during SST with XtraBackup:
  - If donor is 5.6 and joiner is 5.7: A warning is printed to perform `mysql_upgrade`.
  - If donor is 5.7 and joiner is 5.6: An error is printed and SST is rejected.

Fixed Bugs

- PXC-825: Fixed script for SST with XtraBackup (`wsrep_sst_xtrabackup-v2`) to include the `--defaults-group-suffix` when logging to syslog. For more information, see #1559498.

- PXC-827: Fixed handling of different binlog names between donor and joiner nodes when GTID is enabled. For more information, see #1690398.
- PXC-830: Rejected the `RESET MASTER` operation when wsrep provider is enabled and `gtid_mode` is set to `ON`. For more information, see #1249284.
- PXC-833: Fixed connection failure handling during SST by making the donor retry connection to joiner every second for a maximum of 30 retries. For more information, see #1696273.
- PXC-841: Added check to avoid replication of DDL if `sql_log_bin` is disabled. For more information, see #1706820.
- PXC-853: Fixed cluster recovery by enabling `wsrep_ready` whenever nodes become PRIMARY.
- PXC-862: Fixed script for SST with XtraBackup (`wsrep_sst_xtrabackup-v2`) to use the `ssl-dhparams` value from the configuration file.

Note: As part of fix for PXC-827, version communication was added to the SST protocol. As a result, newer version of PXC (as of 5.6.37 and later) cannot act as donor when joining an older version PXC node (prior to 5.6.37). It will work fine vice versa: old node can act as donor when joining nodes with new version.

Percona XtraDB Cluster 5.6.36-26.20

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.36-26.20 on June 2, 2017. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.36-26.20 is now the current release, based on the following:

- Percona Server 5.6.36-82.0
- Galera Replication library 3.20
- wsrep API version 26

All Percona software is open-source and free. Details of this release can be found in the 5.6.36-26.20 milestone on Launchpad.

Fixed Bugs

- PXC-749: Fixed memory leak when running `INSERT` on a table without primary key defined and *`wsrep_certify_nonPK`* disabled (set to `0`).

  Note: It is recommended to have primary keys defined on all tables for correct write set replication.

- PXC-813: Fixed SST script to use UTC time format.
- PXC-823: Fixed SST flow to gracefully shut down JOINER node if SST fails because DONOR leaves the cluster due to network failure. This ensures that the DONOR is then able to recover to synced state when network connectivity is restored For more information, see #1684810.

Percona XtraDB Cluster 5.6.35-26.20-3

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.35-26.20-3 on April 19, 2017. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.35-26.20-3 is now the current release, based on the following:

- Percona Server 5.6.35-81.0
- Galera Replication library 3.20
- wsrep API version 26

All Percona software is open-source and free. Details of this release can be found in the 5.6.35-26.20-3 milestone on Launchpad.

Note: Due to end of life, Percona will stop producing packages for the following distributions after July 31, 2017:

- Red Hat Enterprise Linux 5 (Tikanga)
- Ubuntu 12.04 LTS (Precise Pangolin)

You are strongly advised to upgrade to latest stable versions if you want to continue using Percona software.

Fixed Bugs

- Updated semantics for gcache page cleanup to trigger when either *`gcache.keep_pages_size`* or *`gcache.keep_pages_count`* exceeds the limit, instead of both at the same time.
- Added support for passing the XtraBackup buffer pool size with the `use-memory` option under `[xtrabackup]` and the `innodb_buffer_pool_size` option under `[mysqld]` when the `--use-memory` option is not passed with the `inno-apply-opts` option under `[sst]`.
- Fixed gcache page cleanup not triggering when limits are exceeded.
- PXC-782: Updated `xtrabackup-v2` script to use the *`tmpdir`* option (if it is set under `[sst]`, `[xtrabackup]` or `[mysqld]`, in that order).
- PXC-784: Fixed the `pc.recovery` procedure to abort if the `gvwstate.dat` file is empty or invalid, and fall back to normal joining process. For more information, see #1669333.
- PXC-794: Updated the *`sockopt`* option to include a comma at the beginning if it is not set by the user.
- PXC-797: Blocked *`wsrep_desync`* toggling while node is paused to avoid halting the cluster when running `FLUSH TABLES WITH READ LOCK`. For more information, see #1370532.
- Fixed several packaging and dependency issues.

Percona XtraDB Cluster 5.6.35-26.20

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.35-26.20 on March 10, 2017. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.35-26.20 is now the current release, based on the following:

- Percona Server 5.6.35-80.0
- Galera Replication library 3.20

- wsrep API version 26

All Percona software is open-source and free. Details of this release can be found in the 5.6.35-26.20 milestone on Launchpad.

Fixed Bugs

- BLD-593: Limited the use of `rm` and `chown` by `mysqld_safe` to avoid exploits of the CVE-2016-5617 vulnerability. For more information, see #1660265.

  Credit to Dawid Golunski (https://legalhackers.com).

- BLD-610: Added version number to the dependency requirements of the full RPM package.
- BLD-645: Fixed `mysqld_safe` to support options with a forward slash (`/`). For more information, see #1652838.

Percona XtraDB Cluster 5.6.34-26.19

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.34-26.19 on December 14, 2016. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.34-26.19 is now the current release, based on the following:

- Percona Server 5.6.34-79.1
- Galera Replication library 3.19
- wsrep API version 26

All Percona software is open-source and free. Details of this release can be found in the 5.6.34-26.19 milestone on Launchpad.

Deprecated

- The following encryption modes are now deprecated:
  - `encrypt=1`
  - `encrypt=2`
  - `encrypt=3`

  The default is `encrypt=0` with encryption disabled. The recommended mode now is the new `encrypt=4`, which uses SSL files generated by MySQL.

  For more information, see *Encrypting PXC Traffic*.

New Features

- Added `encrypt=4` mode for SST encryption that uses SSL files generated by MySQL. Modes `1`, `2`, and `3` are now deprecated.

Fixed Bugs

- Optimized IST donor selection logic to avoid SST. Child processes are now cleaned-up and node state is resumed if SST fails.
- Added `init.ok` to the list of files that do not get removed during SST.
- Fixed error with ASIO library not acknowledging an `EPOLLIN` event when building Galera.
- Fixed stalling of DML workload on slave node caused by `FLUSH TABLE` executed on master.

  For more information, see #1629296.
- Fixed `super_read_only` to not apply to Galera replication applier.

  For more information, see #1634295.
- Redirected `netcat` output to `stdout` to avoid it in the log.

  For more information, see #1625968.
- Enabled replication of `ALTER USER` statements.

  For more information, see #1376269.
- Changed the *`wsrep_max_ws_rows`* variable to ignore non-replicated write-sets generated by DML action on temporary tables (explict or implicit).

  For more information, see #1638138.
- Fixed SST to fail with an error if SSL is not supported by `socat`, instead of switching to unencrypted mode.
- Fixed SST with SSL to auto-generate a 2048-bit `dhparams` file for versions of `socat` before 1.7.3. These older versions use 512-bit `dhparams` file by default that get rejected by newer clients with `dh key too small` error.
- PXC-731: Changed the *`wsrep_cluster_name`* variable to read-only, because changing it dynamically leads to high overhead.

  For more information, see #1620439.
- PXC-732: Improved error message when any of the SSL files required for SST are missing.
- PXC-735: Fixed SST to fail with an error when `netcat` is used (*`transferfmt`* set to `nc`) with SSL encryption (*`encrypt`* set to `2`, `3` or `4`), instead of silently switching to unencrypted mode.
- Fixed faulty switch case that caused cluster to stall when the *`repl.commit_order`* variable was set to `2` (`LOCAL_OOOC` mode that should allow out-of-order committing for local transactions).

Percona XtraDB Cluster 5.6.32-25.17

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.32-25.17 on October 4, 2016. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.32-25.17 is now the current release, based on the following:

- Percona Server 5.6.32-78.1
- Galera Replication library 3.17
- wsrep API version 25

All Percona software is open-source and free. Details of this release can be found in the 5.6.32-25.17 milestone on Launchpad.

For more information about relevant Codership releases, see this announcement.

Fixed Bugs

- Fixed `DONOR` node getting stuck in `Joined` state after successful SST. Bugs fixed #1608680 and #1611728.
- Removed protection against repeated calls of `wsrep->pause()` on the same node to allow parallel RSU operation.
- Fixed error when running `SHOW STATUS` during group state update.
- Setting the *`wsrep_auto_increment_control`* to `OFF` didn't restore the original user set value.
- Corrected the return code of `sst_flush_tables()` function to return a non-negative error code and thus pass assertion.
- Using *Percona XtraBackup* as the SST method now requires *Percona XtraBackup* 2.3.5 or later.
- Fixed memory leak and stale pointer due to stats not freeing when toggling the *`wsrep_provider`* variable.
- Fixed failure of `ROLLBACK` to register `wsrep_handler`.
- Improved rollback process to ensure that when a transaction is rolled back, any statements open by the transaction are also rolled back.
- Fixed failure of symmetric encryption during SST.
- Performing an SST would display the encryption key in the logs. It's recommended to use use `encrypt-key-file` instead of `encrypt-key` option.
- Node transitioning to non-primary state with active `ALTER TABLE` could result in a crash.
- Fixed setting of `seqno` in `grastate.dat` to `-1` on clean shutdown.
- Fixed failure of asynchronous TOI actions (like `DROP`) for non-primary nodes.
- Removed the unused `sst_special_dirs` variable.
- Added support of `defaults-group-suffix` for SST scripts.
- Other low-level fixes and improvements for better stability.

Percona XtraDB Cluster 5.6.30-25.16.3

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.30-25.16.3 on September 21, 2016. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.30-25.16.3 is now the current release, based on the following:

- Percona Server 5.6.30-76.3
- Galera Replication library 3.16
- wsrep API version 25

Bug Fixed

Limiting `ld_preload` libraries to be loaded from specific directories in `mysqld_safe` didn't work correctly for relative paths. Bug fixed #1624247.

Fixed possible privilege escalation that could be used when running `REPAIR TABLE` on a `MyISAM` table. Bug fixed #1624397.

The general query log and slow query log cannot be written to files ending in `.ini` and `.cnf` anymore. Bug fixed #1624400.

Implemented restrictions on symlinked files (`error_log`, `pid_file`) that can't be used with `mysqld_safe`. Bug fixed #1624449.

Other bugs fixed: #1553938.

Percona XtraDB Cluster 5.6.30-25.16.2

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.30-25.16.2 on September 15, 2016. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.30-25.16.2 is now the current release, based on the following:

- Percona Server 5.6.30-76.3
- Galera Replication library 3.16
- wsrep API version 25

This release is providing a fix for CVE-2016-6662.

Bug Fixed

Due to security reasons `ld_preload` libraries can now only be loaded from the system directories (`/usr/lib64`, `/usr/lib`) and the *MySQL* installation base directory.

Percona XtraDB Cluster 5.6.30-25.16

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.30-25.16 on June 10, 2016. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.30-25.16 is now the current release, based on the following:

- Percona Server 5.6.30-76.3
- Galera Replication library 3.16
- wsrep API version 25

All Percona software is open-source and free. Details of this release can be found in the 5.6.30-25.16 milestone on Launchpad.

For more information about relevant Codership releases, see this announcement.

New Features

- PXC now uses `wsrep_desync_count` introduced in Galera 3.16 by Codership, instead of the concept that was previously implemented by Percona. The following logic applies:
  - If a node is explicitly desynced, then implicitly desyncing a node using RSU/FTWRL is allowed.
  - If a node is implicitly desynced using RSU/FTWRL, then explicitly desyncing a node is blocked until implicit desync is complete.
  - If a node is explicitly desynced and then implicitly desycned using RSU/FTWRL, then any request for another implicit desync is blocked until the former implicit desync is complete.

Fixed Bugs

- Changing `wsrep_provider` while node is paused or desynced is not allowed.
- TOI now checks that a node is ready to process DDL and DML before starting execution, to prevent node from crashing if it becomes non-primary.
- The `wsrep_row_upd_check_foreign_constraints` function now checks that `fk-reference-table` is open before marking it open.

Percona XtraDB Cluster 5.6.29-25.15

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.29-25.15 on May 20, 2016. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.29-25.15 is now the current release, based on the following:

- Percona Server 5.6.29-76.2
- Galera Replicator 3.15
- Codership wsrep API 25.15

All Percona software is open-source and free. Details of this release can be found in the 5.6.29-25.15 milestone on Launchpad.

For more information about relevant Codership releases, see this announcement.

Bugs Fixed

- Node eviction in the middle of SST now causes the node to shut down properly.
- After an error during node startup, the state is now marked *unsafe* only if SST is required.
- Fixed data inconsistency during multi-insert auto-increment workload on async master with `binlog-format=STATEMENT` when a node begins async slave with `wsrep_auto_increment_control=ON`.
- Fixed crash when a prepare statement is aborted (due to conflict with applier) and then replayed.
- Removed a special case condition in `wsrep_recover()` that whould not happen under normal conditions.
- Percona XtraDB Cluster no longer fails during SST, if a node reserves a very large amount of memory for InnoDB buffer pool.

- If the value of `wsrep_cluster_address` is not valid, trying to create a slave thread will now generate a warning instead of an error, and the thread will not be created.
- Fixed error with loading data infile (LDI) into a multi-partitioned table.
- The `wsrep_node_name` variable now defaults to host name.
- Starting `mysqld` with unknown option now fails with a clear error message, instead of randomly crashing.
- Optimized the operation of SST and IST when a node fails during startup.
- The `wsrep_desync` variable can now be enabled only after a node is synced with cluster. That is, it cannot be set during node bootup configuration).
- Fixed crash when setting a high flow control limit (`fc_limit`) and the recv queue fills up.
- Only the default 16 KB page size (`innodb_page_size=16384`) is accepted until the relevant upstream bug is fixed by Codership (see https://github.com/codership/galera/issues/398). All other sizes will report `Invalid page size` and shut down (the server will not start up).
- If a node is executing RSU/FTWRL, explicit desync of the node will not happen until the implicit desync action is complete.
- Fixed multiple bugs in test suite to improve quality assurance.

Percona XtraDB Cluster 5.6.28-25.14

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.28-25.14 on January 29, 2016. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.28-25.14 is now the current release, based on the following:

- Percona Server 5.6.28-76.1
- Galera Replicator 3.14
- Codership wsrep API 25.13

All Percona software is open-source and free. Details of this release can be found in the 5.6.28-25.14 milestone on Launchpad.

For more information about relevant Codership releases, see this announcement.

Bugs Fixed

- #1494399: Fixed issue caused by replication of events on certain system tables (for example, `mysql.slave_master_info`, `mysql.slave_relay_log_info`). Replication in the Galera eco-system is now avoided when bin-logging is disabled for said tables.

  Note: As part of this fix, when bin-logging is enabled, replication in the Galera eco-system will happen only if `BINLOG_FORMAT` is set to either `ROW` or `STATEMENT`. The recommended format is `ROW`, while `STATEMENT` is required only for the `pt-table-checksum` tool to operate correctly. If `BINLOG_FORMAT` is set to `MIXED`, replication of events in the Galera eco-system tables will not happen even with bin-logging enabled for those tables.

- #1522385: Fixed GTID holes caused by skipped replication. A slave might ignore an event replicated from master, if the same event has already been executed on the slave. Such events are now propagated in the form of special GTID events to maintain consistency.

- #1532857: The installer now creates a /var/lib/galera/ directory (assigned to user nobody), which can be used by garbd in the event it is started from a directory that garbd cannot write to.

Known Issues

- #1531842: Two instances of garbd cannot be started from the same working directory. This happens because each instance creates a state file (gvwstate.dat) in the current working directory by default. Although garbd is configured to use the base_dir variable, it was not registered due to a bug. Until garbd is fixed, you should start each instance from a separate working directory.

Percona XtraDB Cluster 5.6.27-25.13

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.27-25.13 on January 11, 2016. Binaries are available from the downloads section or from our *software repositories*.

Percona XtraDB Cluster 5.6.27-25.13 is now the current release, based on the following:

- Percona Server 5.6.27-75.0
- Percona Server 5.6.27-76.0
- Galera Replicator 3.13
- Codership wsrep API 25.12

All Percona software is open-source and free. Details of this release can be found in the 5.6.27-25.13 milestone on Launchpad.

For more information about relevant Codership releases, see this announcement.

Note: Due to new dependency on libnuma1 package in Debian/Ubuntu, please run one of the following commands to upgrade the percona-xtradb-cluster-server-56 package:

- `aptitude safe-upgrade`
- `apt-get dist-upgrade`
- `apt-get install percona-xtradb-cluster-server-5.6`

New Features

- There is a new script for building Percona XtraDB Cluster from source. For more information, see *Compiling and Installing from Source Code*.
- *wsrep_on* is now a session only variable. That means toggling it will not affect other clients connected to said node. Only the session/client modifying it will be affected. Trying to toggle *wsrep_on* in the middle of a transaction will now result in an error. Trx will capture the state of *wsrep_on* during start and will continue to use it. Start here means when the first logical changing statement is executed within transaction context.

Bugs Fixed

- #1261688 and #1292842: Fixed race condition when two skipped replication transactions were rolled back, which caused `[ERROR] WSREP: FSM: no such a transition ROLLED_BACK -> ROLLED_BACK` with `LOAD DATA INFILE`

- #1362830: Corrected xtrabackup-v2 script to consider only the last specified log_bin directive in my.cnf. Multiple log_bin directives caused SST to fail.
- #1370532: Toggling *wsrep_desync* while node is paused is now blocked.
- #1404168: Removed support for innodb_fake_changes variable.
- #1455098: Fixed failure of LDI on partitioned table. This was caused by partitioned table handler disabling bin-logging and Native Handler (InnoDB) failing to generate needed bin-logs eventually causing skipping of statement replication.
- #1503349: garbd now uses default port number if it is not specified in sysconfig.
- #1505184: Corrected wsrep_sst_auth script to ensure that user name and password for SST is passed to XtraBackup through internal command-line invocation. ps -ef doesn't list these credentials so passing it internally is fine, too.
- #1520491: FLUSH TABLE statements are not replicated any more, because it lead to an existing upstream fix pending deadlock error. This fix also takes care of original fix to avoid increment of local GTID.
- #1528020: Fixed async slave thread failure caused by redundant updates of mysql.event table with the same value. Redundant updates are now avoided and will not be bin-logged.
- Fixed **garb** init script causing new UUIDs to be generated every time it runs. This error was due to missing base_dir configuration when gardb didn't have write-access to current working directory. garbd will now try to use cwd. Then it will try to use /var/lib/galera (like most Linux daemons). If it fails to use or create /var/lib/galera, it will throw a fatal error.
- Fixed replication of DROP TABLE statement with a mix of temporary and non-temporary tables (for example, DROP TABLE temp_t1, non_temp_t2), which caused errorneous DROP TEMPORARY TABLE stmt on replicated node. Corrected it by detecting such scenarios and creating temporary table on the replicated node, which is then dropped by follow-up DROP statement. All this workload should be part of same unit as temporary tables are session-specific.
- Fixed error when *wsrep_cluster_name* value over 32 characters long caused gmcast message to exceed maximum length. Imposed a limit of 32 character on *wsrep_cluster_name*.
- Added code to properly handle default values for wsrep_* variables, which caused an error/crash.
- Fixed error when a CREATE TABLE AS SELECT (CTAS) statement still tried to certify a transaction on a table without primary key even if certification of tables without primary key was disabled. This error was caused by CTAS setting trx_id (fake_trx_id) to execute SELECT and failing to reset it back to -1 during INSERT as certification is disabled.
- Fixed crashing of INSERT .... SELECT for MyISAM table with *wsrep_replicate_myisam* set to ON. This was caused by TOI being invoked twice when source and destination tables were MyISAM.
- Fixed crash when caching write-set data beyond configured limit. This was caused by TOI flow failing to consider/check error resulting from limit enforcement.
- Fixed error when loading MyISAM table from schema temporary table (with *wsrep_replicate_myisam* set to ON). This was caused by temporary table lookup being done using get_table_name(), which could be misleading as table_name for temporary tables is set to temporary generated name. Original name of the table is part of table_alias. The fix corrected condition to consider both table_name and alias_name.
- Fixed error when changing *wsrep_provider* in the middle of a transaction or as part of a procedure/trigger. This is now blocked to avoid inconsistency.
- Fixed TOI state inconsistency caused by DELAYED_INSERT on MyISAM table (TOI_END was not called). Now the DELAYED_ qualifier will be ignored and statement will be interpreted as normal INSERT.
- Corrected locking semantics for FLUSH TABLES WITH READ LOCK (FTWRL). It now avoids freeing inheritted lock if follow-up FLUSH TABLE statement fails. Only frees self-acquired lock.

- Fixed crash caused by `GET_LOCK` + *wsrep_drupal_282555_workaround*. `GET_LOCK` path failed to free all instances of user-level locks after it inherited `multiple-user-locks` from Percona Server. The cleanup code now removes all possible references of locks.
- Fixed cluster node getting stuck in `Donor/Desync` state after a hard recovery, because of an erroneous type cast in source code.
- Corrected DDL and DML semantics for MyISAM:
  - DDL (CREATE/DROP/TRUNCATE) on MyISAM will be replicated irrespective of `wsrep_replicate_miysam` value
  - DML (INSERT/UPDATE/DELETE) on MyISAM will be replicated only if *wsrep_replicate_myisam* is enabled
  - SST will get full transfer irrespective of *wsrep_replicate_myisam* value (it will get MyISAM tables from donor if any)
  - Difference in configuration of `pxc-cluster` node on enforce_storage_engine front may result in picking up different engine for same table on different nodes
  - `CREATE TABLE AS SELECT` (CTAS) statements use non-TOI replication and are replicated only if there is involvement of InnoDB table that needs trx (involvement of MyISAM table will cause CTAS statement to skip replication)
- SST fails with `innodb_data_home_dir/innodb_log_home_dir`. This was a bug in Percona XtraBackup, which is fixed in 2.3.3 and 2.2.12.

Known Issues

- 1330941: Conflict between *wsrep_OSU_method* set to `RSU` and *wsrep_desync* set to `ON` was not considered a bug.
- 1443755: Causal reads introduces surprising latency in single node clusters.
- 1522385: Holes are introduced in Master-Slave GTID eco-system on replicated nodes if any of the cluster nodes are acting as asynchronous slaves to an independent master.
- Enabling *wsrep_desync* (from previous `OFF` state) will wait until previous `wsrep_desync=OFF` operation is completed.

Percona XtraDB Cluster 5.6.26-25.12

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.26-25.12 on October 14th 2015. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.26-74.0 including all the bug fixes in it, *Galera Replicator* 3.12 and on *Codership wsrep API* 25.11 is now the current **General Availability** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.26-25.12 milestone at Launchpad.

New Features

Variable *wsrep_dirty_reads* now has Global scope as well. (#1505609).

Bugs Fixed

Asynchronous replication slave thread would continue to run and execute even if the asynchronous slave was non-primary. Bug fixed #188.

If a cluster conflict happened while asynchronous replication was running it would cause the following message to be printed repeatedly in the error log: `conflict state 7 after post commit`. Bug fixed #181.

If a transaction was Brute-Force aborted, subsequent prepared statement would either cause the client to assert or fail to return a result set. Bug fixed #126.

Server assertion would happen when the `Perfomance Schema` instrumentation from a previous prepared statement was not correctly reset before the next one or if a statement could not be certified because one of the nodes did not respond. Bug fixed #125.

Percona XtraDB Cluster 5.6.25-25.12

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6.25-25.12 on September 20th 2015. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.25-73.1 including all the bug fixes in it, *Galera Replicator* 3.12 and on *Codership wsrep API* 25.11 is now the current **General Availability** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.24-25.12 milestone at Launchpad.

New Features

Percona XtraDB Cluster has implemented Support for PROXY protocol. The implementation is based on a patch developed by Thierry Fournier.

MTR coverage has been added to all tests in galera suite. Many bugs associated with mtr tests have been fixed.

A new variable *`gcache.keep_pages_count`*, analogous to *`gcache.keep_pages_size`*, has been added. The variable limits the number of overflow pages rather than the total memory occupied by all overflow pages. Whenever either of the variables are updated at runtime to a non-zero value, cleanup is called on excess overflow pages to delete them. This feature also fixes the bugs with integer overflow in the `gcache` module.

Updates have been made to wsrep code to ensure greater concordance with binary log and GTID so that failover of async slaves, among nodes of the cluster is seamless and consistent. To ensure this in #1421360, all `FLUSH` commands (except `FLUSH BINARY LOG` and `FLUSH LOGS`, and read lock-based flush such as `FLUSH TABLES WITH READ LOCK` and `FLUSH TABLES FOR EXPORT`), `ANALYZE TABLE`, Percona Server-specific flush statements for user statistics and page tracking bitmaps are executed under Total Order Isolation (TOI) so that they are replicated to other nodes in the cluster when they are written to binary log.

Percona XtraDB Cluster has temporarily disabled savepoints in triggers and stored functions. The reason is that even having fixed bug #1438990 and bug #1464468 we have found more cases where savepoints in triggers break binary logging and replication, resulting in server crashes and broken slaves. This feature will be disabled until the above issues are properly resolved.

Bugs fixed

SHOW STATUS LIKE ... and SHOW STATUS were taking time proportional to size of *gcache.size*. Bug fixed #1462674.

When disk space would get filled with gcache.page files, Galera would crash when the next page file was created. Bug fixed #1488535.

XtraBackup SST didn't clobber backup-my.cnf which caused SST to fail. Bug fixed #1431101.

Error from process::wait was not checked in joiner thread leading to joiner starting erroneously even when SST had failed. Bug fixed #1402166.

Due to an regression introduced in *Percona XtraDB Cluster 5.6.24-25.11*, update of the *wsrep_cluster_address* variable, following the update of *wsrep_provider_options* variable would cause the server to deadlock. Bug fixed PXC-421.

mysqldump SST could stall due to a regression in desync mutex introduced in *Percona XtraDB Cluster 5.6.24-25.11* by fixing the bug #1288528. Bug fixed PXC-423.

mysql_tzinfo_to_sql sets *wsrep_replicate_myisam* variable at session scope so that statements are replicated correctly. Bug fixed PXC-332.

Percona-XtraDB-Cluster-devel-56 package was not included in the Percona-XtraDB-Cluster-full-56 metapackage on *CentOS* due to a conflict with upstream mysql package. Bug fixed PXC-381.

Running service mysql start and then service mysql@boostrap start afterwards would cause server shutdown. Bug fixed PXC-385.

NO_WRITE_TO_BINLOG / LOCAL for ANALYZE TABLE, OPTIMIZE TABLE, REPAIR TABLE, FLUSH commands will ensure it is not written to binary log (as in mysql async replication) and not replicated in wsrep. Bug fixed PXC-391.

FLUSH TABLES WITH READ LOCK failure (with non-existent tables) didn't resume the galera provider, causing deadlock. Bug fixed PXC-399.

Percona XtraDB Cluster will not blocking DDL statements on tables which are used with ... FOR EXPORT or ... WITH READ LOCK, it will give return an error message about read lock. Bug fixed PXC-403.

Fixed the update of variable wsrep_slave_threads variable regarding the default value assignment, invalid value truncation, and error issued while threads are still being closed. Bug fixed PXC-420.

Server would crash during startup if *gcs.fc_limit* variable was specified twice in *wsrep_provider_options*. Bug fixed PXC-428.

The mysql client in *Percona XtraDB Cluster* has been built with system readline instead of editline. Bug fixed PXC-430.

Bugs in 32-bit galera associated with statvfs in available_storage and integer overflow after multiplication in offset calculation have been fixed. Bug fixed PXC-433.

Galera 3 was failing to build on all non-intel platforms. Architecture specific CCFLAGS have been removed and provision for inheriting CCFLAGS, CFLAGS, CXXFLAGS and LDFLAGS have been added to SConstruct. Bug fixed PXC-326.

Non-global read locks such as FLUSH TABLES WITH READ LOCK and FLUSH TABLES FOR EXPORT paused galera provider but didn't block commit globally which wsrep_to_isolation_begin (for DDL) was made aware of. Bug fixed PXC-403.

Following bug fixes have been ported from *Percona Server for MySQL* 5.6.26-74.0: bug #1454441, bug #1470677, bug #1472256, and bug #1472251.

Other bugs fixed: PXC-370, PXC-429, PXC-415, PXC-400 and PXC-416.

Percona XtraDB Cluster 5.6.24-25.11

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6 on June 3rd 2015. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.24-72.2 including all the bug fixes in it, *Galera Replicator* 3.11 and on *Codership wsrep API* 25.11 is now the current **General Availability** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.24-25.11 milestone at Launchpad.

New Features

Percona XtraDB Cluster now allows reads in non-primary state by introducing a new session variable, `wsrep_dirty_reads`. This variable is boolean and is `OFF` by default. When set to `ON`, a *Percona XtraDB Cluster* node accepts queries that only read, but not modify data even if the node is in the non-PRIM state (#1407770).

Percona XtraDB Cluster now allows queries against `INFORMATION_SCHEMA` and `PERFORMANCE_SCHEMA` even with variables `wsrep_ready` and `wsrep_dirty_reads` set to `OFF`. This allows monitoring applications to monitor the node when it is even in non-PRIM state (#1409618).

`wsrep_replicate_myisam` variable is now both session and global variable (#1280280).

Percona XtraDB Cluster now uses `getifaddrs` for node address detection. It detects the address from the interface to which `mysqld` is bound if the address is not obtained from `bind_address` or `wsrep_node_address`, when both are unset (#1252700).

Percona XtraDB Cluster has implemented two new status variables: `wsrep_cert_bucket_count` and `wsrep_gcache_pool_size` for better instrumentation of galera memory usage. `wsrep_cert_bucket_count` shows the number of cells in the certification index hash-table and `wsrep_gcache_pool_size` shows the size of the page pool and/or dynamic memory allocated for gcache (in bytes).

Bugs fixed

Using concurrent `REPLACE`, `LOAD DATA REPLACE` or `INSERT ON DUPLICATE KEY UPDATE` statements in the `READ COMMITTED` isolation level or with the `innodb_locks_unsafe_for_binlog` option enabled could lead to a unique-key constraint violation. Bug fixed #1308016.

Using the Rolling Schema Upgrade as a schema upgrade method due to conflict with `wsrep_desync` would allows only one `ALTER TABLE` to run concurrently. Bugs fixed #1330944 and #1330941.

SST would resume even when the donor was already detected as being in `SYNCED` state. This was caused when `wsrep_desync` was manually set to `OFF` which caused the conflict and resumed the donor sooner. Bug fixed #1288528.

DDL would fail on a node when running a TOI DDL, if one of the nodes has the table locked. Bug fixed #1376747.

`xinet.d` mysqlchk file was missing `type = UNLISTED` to work out of the box. Bug fixed #1418614.

Conflict between enforce_storage_engine and `wsrep_replicate_myisam` for `CREATE TABLE` has been fixed. Bug fixed #1435482.

A specific trigger execution on the master server could cause a slave assertion error under row-based replication. The trigger would satisfy the following conditions: 1) it sets a savepoint; 2) it declares a condition handler which releases this savepoint; 3) the trigger execution passes through the condition handler. Bug fixed #1438990.

Percona XtraDB Cluster Debian init script was testing connection with wrong credentials. Bug fixed #1439673.

Race condition between IST and SST fixed in xtrabackup-v2 SST. Bugs fixed #1441762, #1443881, and #1451524.

SST will now fail when move-back fails instead of continuing and failing at the next step. Bug fixed #1451670.

Percona XtraDB Cluster `.deb` binaries were built without fast mutexes. Bug fixed #1457118.

The error message text returned to the client in the non-primary mode is now more descriptive (`"WSREP has not yet prepared node for application use"`), instead of `"Unknown command"` returned previously. Bug fixed #1426378.

Out-of-bount memory access issue in `seqno_reset()` function has been fixed.

`wsrep_local_cached_downto` would underflow when the node on which it is queried has no write-sets in gcache.

`wsrep_OSU_method` is now both Global and Session variable.

Other bugs fixed: #1290526.

Percona XtraDB Cluster 5.6.22-25.8

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6 on March 5th 2015. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.22-72.0 including all the bug fixes in it, Galera Replicator 3.9 and on Codership wsrep API 25.8 is now the current **General Availability** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.22-25.8 milestone at Launchpad.

Bugs fixed

XtraBackup SST wouldn't stop when *MySQL* was `SIGKILLed`. This would prevent *MySQL* to initiate a new transfer as port 4444 was already utilized. Bug fixed #1380697.

`wsrep_sst_xtrabackup-v2` script was causing **`innobackupex`** to print a false positive stack trace into the log. Bug fixed #1407599.

MyISAM DDL (`CREATE/DROP`) isn't replicated any more when *`wsrep_replicate_myisam`* is `OFF`. Note, for older nodes in the cluster, `wsrep-replicate-myisam` should work since the TOI decision (for MyISAM DDL) is done on origin node. Mixing of non-MyISAM and MyISAM tables in the same DDL statement is not recommended with *`wsrep_replicate_myisam`* `OFF` since if any table in list is *MyISAM*, the whole DDL statement is not put under TOI (total order isolation). Bug fixed #1402338.

`gcache.mem_size` has been deprecated. A warning will now be generated if the variable has value different than `0`. Bug fixed #1392408.

stderr of SST/Innobackupex is logged to syslog with appropriate tags if sst-syslog is in [sst] or [mysqld_safe] has syslog in my.cnf. This can be overridden by setting the sst-syslog to -1 in [sst]. Bug fixed #1399134.

clustercheck can now check if the node is PRIMARY or not, to allow for synced nodes which go out of PRIMARY not to take any writes/reads. Bug fixed #1403566.

SST will now fail early if the xtrabackup_checkpoints is missing on the joiner side. Bug fixed #1405985.

socat utility was not properly terminated after a timeout. Bug fixed #1409710.

When started (without bootstrap), the node would hang if it couldn't find a primary node. Bug fixed #1413258.

10 seconds timeout in *Percona XtraBackup SST Configuration* script was not enough for the joiner to delete existing files before it started the socat receiver on systems with big datadir. Bug fixed #1413879.

Non booststrap node could crash while attempting to perform table%cache operations with the BF applier failed to open_and_lock_tables warning. Bug fixed #1414635.

Percona XtraDB Cluster 5.6 would crash on ALTER TABLE / CREATE INDEX with Failing assertion: table->n_rec_locks == 0. Bug fixed #1282707.

Variable length arrays in WSREP code were causing debug builds to fail. Bug fixed #1409042.

Race condition between donor and joiner in *Percona XtraBackup SST Configuration* has been fixed. This caused *XtraBackup SST* to fail when joiner took longer to spawn the second listener for SST. Bug fixed #1405668.

Signal handling in mysqld has been fixed for SST processes. Bug fixed #1399175.

SST processes are now spawned with fork/exec instead of posix_spawn to allow for better cleanup of child processes in event of non-graceful termination (SIGKILL or a crash etc.). Bug fixed #1382797.

wsrep_local_cached_downto would underflow when the node on which it is queried had no write-sets in gcache. Bug fixed #1262179.

A typo in *wsrep_provider_options* could cause an unhandled exception. Bug fixed #215.

Interrupted IST would result in HA_ERR_KEY_NOT_FOUND error in subsequent IST. Bug fixed #210.

garb logger was showing incorrect error message. Bug fixed #168.

Other bugs fixed: #1275814.

Known Issues

For those affected by crashes on donor during SST due to backup locks (#1401133), please add the following to your my.cnf configuration file:

```
[sst]
inno-backup-opts='--no-backup-locks'
```

option as a workaround to force FLUSH TABLES WITH READ LOCK (**NOTE:** This workaround will is available only if you're using *Percona XtraBackup* 2.2.9 or newer.). Or as an alternative you can set your environment variable FORCE_FTWRL to 1.

Help us improve quality by reporting any bugs you encounter using our bug tracking system. As always, thanks for your continued support of Percona!

Percona XtraDB Cluster 5.6.21-25.8

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6 on November 25th 2014. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.21-70.1 including all the bug fixes in it, Galera Replicator 3.8 and on Codership wsrep API 25.7 is now the current **General Availability** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.21-25.8 milestone at Launchpad.

New Features

Galera 3.8 introduces auto-eviction for nodes in the cluster experiencing network issues like packet loss. It is off by default and is turned on with *`evs.auto_evict`* option. This feature requires EVS protocol version (*`evs.version`*) `1`. During the EVS protocol upgrade all membership changes are communicated over EVS protocol version `0` to preserve backwards compatibility, protocol is upgraded to the highest commonly supported version when forming a new group so if there exist a single node with older version in the group, the group protocol version remains as `0` and auto-eviction is not functional. (#1274192).

Percona XtraDB Cluster now supports backup locks in XtraBackup SST (in the default `xtrabackup-v2` *`wsrep_sst_method`*). Backup locks are used in lieu of `FLUSH TABLES WITH READ LOCK` on the donor during SST. This should allow for minimal disruption of existing and incoming queries, even under high load. Thus, this should allow for even faster SST and node being in 'donor/desynced' state. This also introduces following constraints: *Percona XtraDB Cluster* 5.6.21 requires *Percona XtraBackup* 2.2.5 or higher; An older (< 5.6.21) joiner cannot SST from a newer (>= 5.6.21) donor. This is enforced through SST versioning (sent from joiner to donor during SST) and logged to error log explicitly. (#1390552).

Percona XtraDB Cluster is now shipped with Galera MTR test suite.

Bugs fixed

Percona XtraDB Cluster now shows a warning in case additional utilities, like `pv` which may not affect critical path of SST, are not installed. Bug fixed #1248688.

Fixed the `UNIV_DEBUG` build failures. Bug fixed #1384413.

`mysqldump` SST can now use username/password from *`wsrep_sst_auth`* under group of `[sst]` in `my.cnf` in order not to display the credentials in the error log. Bug fixed #1293798.

Normal shutdown under load would cause server to remain hanging because replayer failed to finish. Bug fixed #1358701.

`wsrep_causal_reads` variable was not honored when declared as global. Bug fixed #1361859.

Assertion failure `lock != ctx->wait_lock` has been fixed. Bug fixed #1364840.

`garbd` would not work when cluster address was specified without the port. Bug fixed #1365193.

Fixed wsrep options compiler warnings in *Fedora* 20. Bug fixed #1369916.

If `mysqld` gets killed during the SST it will leave an unclean data directory behind. This would cause *Percona XtraDB Cluster* to fail when the server would be started next time because the data directory would be corrupted. This was fixed by resuming the startup in case `wsrep-recover` failed to recover due to corrupted data directory. The old behavior is still achievable through `--exit-on-recover-fail` command line parameter to `mysqld_safe` or `exit-on-recover-fail` under `[mysqld_safe]` in `my.cnf`. Bug fixed #1378578.

Percona XtraDB Cluster now reads environment variables for mysqld from following files (if present): /etc/default/mysql in Debian/Ubuntu; `/etc/sysconfig/mysql` in CentOS 6 or lower; `/etc/sysconfig/mysql` in CentOS 7 with `mysql.service`; `/etc/sysconfig/XYZ` in CentOS 7 with `mysql@XYZ.service` (`/etc/sysconfig/bootstrap` is supplied by default). Bug fixed #1381492.

`gvwstate.dat` file was removed on joiner when *Percona XtraBackup SST Configuration* method was used. Bug fixed #1388059.

Percona XtraDB Cluster now detects older joiners which don't have the backup lock support. Bug fixed #1390552.

Longer `wsrep-recover` is now handled gracefully in Debian init scripts rather than returning immediately with a false positive fail.

`wsrep-recover` log is now also written to mysql error log now.

Issue with stale PID files and Debian init script have been fixed now. It now emits a warning for stale PID files.

`sst_in_progress` file is not removed anymore in case of failed SST.

In case stored procedure containing a non-InnoDB statement (MyISAM) performed autocommit, that commit would be entered two times: at statement end and next time at stored procedure end. Bug fixed #2.

TOI now skips replication if all tables are temporary. Bugs fixed #11 and #13.

Two appliers conflicting with local transaction and resulting later in (acceptable) BF-BF lock conflict, would cause cluster to hang when the other BF thread would not grant the lock back after its local transaction got aborted. Bug fixed #7.

Bootstrapping a node tried to resolve gcomm address list specified in `wsrep-cluster-address`. Bug fixed #88.

`xtrabackup-v2` SST did not clean the undo log directory. Bug fixed #1394836.

Inserts to a table with autoincrement primary key could result in duplicate key error if another node joined or dropped from the cluster during the insert processing. Bug fixed #1366997.

Other bugs fixed #1378138, #1377226, #1376965, #1356859, #1026181, #1367173, #1390482, #1391634, and #1392369.

Help us improve quality by reporting any bugs you encounter using our bug tracking system. As always, thanks for your continued support of Percona!

Percona XtraDB Cluster 5.6.20-25.7

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6 on September 1st 2014. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.20-68.0 including all the bug fixes in it, Galera Replicator 3.7 and on Codership wsrep API 25.7 is now the current **General Availability** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.20-25.7 milestone at Launchpad.

New Features

systemd integration with *RHEL/CentOS* 7 is now available for *Percona XtraDB Cluster* (#1342223).

New session variable `wsrep_sync_wait` has been implemented to control causality check. The old session variable `wsrep_causal_reads` is deprecated but is kept for backward compatibility (#1277053).

Bugs fixed

Running `START TRANSACTION WITH CONSISTENT SNAPSHOT`, `mysqldump` with `--single-transaction` or `mydumper` with disabled binlog would lead to a server crash. Bug fixed #1353644.

`percona-xtradb-cluster-garbd-3.x` package was installed incorrectly on *Debian/Ubuntu*. Bug fixed #1360633.

Fixed `netcat` in SST script for Centos7 `nmap-ncat`. Bug fixed #1359767.

TO isolation was run even when wsrep plugin was not loaded. Bug fixed #1358681.

The error from net read was not handled in native *MySQL* mode. This would cause duplicate key error if there was unfinished transaction at the time of shutdown, because it would be committed during the startup recovery. Bug fixed #1358264.

The netcat in garbd init script has been replaced with nmap for compatibility in *CentOS* 7. Bug fixed #1349384.

`SHOW STATUS` was generating debug output in the error log. Bug fixed #1347818.

Incorrect source string length could lead to server crash. This fix allows maximum of 3500 bytes of key material to be populated, longer keys will be truncated. Bug fixed #1347768.

A memory leak in `wsrep_mysql_parse` function has been fixed. Bug fixed #1345023.

wsrep consistency check is now enabled for `REPLACE ... SELECT` as well. This was implemented because `pt-table-checksum` uses `REPLACE .. SELECT` during checksumming. Bug fixed #1343209.

Client connections were closed unconditionally before generating SST request. Fixed by avoiding closing connections when wsrep is initialized before storage engines. Bug fixed #1258658.

Session-level binlog_format change to `STATEMENT` is now allowed to support `pt-table-checksum`. A warning (to not use it otherwise) is also added to error log.

Other bug fixes: #1280270.

Help us improve quality by reporting any bugs you encounter using our bug tracking system. As always, thanks for your continued support of Percona!

Percona XtraDB Cluster 5.6.19-25.6

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6 on July 21st 2014. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.19-67.0 including all the bug fixes in it, Galera Replicator 3.6 and on Codership wsrep API 25.6 is now the current **General Availability** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.19-25.6 milestone at Launchpad.

New Features

During joiner joining the group, state message exchange provides us with `gcache seqno limits`. That info is now used to choose a donor through IST first, if not possible, only then SST is attempted. *`wsrep_sst_donor`* is also honored here. This is also segment aware. (#1252461)

Asynchronous replication slave thread is stopped when the node tries to apply next replication event while the node is in non-primary state. But it would then remain stopped after node successfully re-joined the cluster. A new variable *`wsrep_restart_slave`* has been implemented, which controls if *MySQL* slave should be restarted automatically when the node joins back to the cluster. (#1288479)

Handling install message and install state message processing has been improved to make group forming more stable in case many nodes are joining the cluster. (#14)

Percona XtraDB Cluster now supports storing the Primary Component state to disk by setting the *`pc.recovery`* variable to `true`. The Primary Component can then recover automatically when all nodes that were part of the last saved state reestablish communications with each other. This feature can be used for automatic recovery from full cluster crashes, such as in the case of a data center power outage and graceful full cluster restarts without the need for explicitly bootstrapping a new Primary Component. (#10)

New *`wsrep_evs_repl_latency`* status variable has been implemented which provides the group communication replication latency information. (#15)

Node consistency issues with foreign keys have been fixed. This fix introduces two new variables: *`wsrep_slave_FK_checks`* and *`wsrep_slave_UK_checks`*. These variables are set to `TRUE` and `FALSE` respectively by default. They control whether Foreign Key and Unique Key checking is done for applier threads. (#1260713).

Bugs fixed

Fixed the race condition in Foreign Key processing that could cause assertion. Bug fixed #1342959.

The restart sequence in `scripts/mysql.server` would fail to capture and return if the start call failed to start the server, so a restart could occur that failed upon start-up, and the script would still return `0` as if it worked without any issues. Bug fixed #1339894.

Updating a unique key value could cause server hang if slave node has enabled parallel slaves. Bug fixed #1280896.

Percona XtraDB Cluster has implemented threadpool scheduling fixes. Bug fixed #1333348.

`garbd` was returning incorrect return code, ie. when `garbd` was already started, return code was `0`. Bug fixed #1308103.

`wsrep_sst_rsync` would silently fail on joiner when `rsync` server port was already taken. Bug fixed #1099783.

Example `wsrep_notify` script failed on node shutdown. Bug fixed #1132955.

When *`gmcast.listen_addr`* was configured to a certain address, local connection point for outgoing connections was not bound to listen address. This would happen if OS has multiple interfaces with IP addresses in the same subnet, it may happen that OS would pick wrong IP for local connection point and other nodes would see connections originating from IP address which was not listened to. Bug fixed #1240964.

Issue with re-setting galera provider (in *`wsrep_provider_options`*) has been fixed. Bug fixed #1260283.

Variable *wsrep_provider_options* couldn't be set in runtime if no provider was loaded. Bug fixed #1260290.

Percona XtraDB Cluster couldn't be built with *Bison* 3.0. Bug fixed #1262439.

mysqld wasn't handling exceeding max writeset size wsrep error correctly. Bug fixed #1270920.

When FLUSH TABLES WITH READ LOCK was used on a node with *wsrep_causal_reads* set to 1 while there was a DML on other nodes then, subsequent SELECTs/SHOW STATUS didn't hang earlier providing non-causal output, that has been fixed here. Bug fixed #1271177.

Lowest group communication layer (evs) would fail to handle the situation properly when big number of nodes would suddenly start to see each other. Bugs fixed #1271918 and #1249805.

Percona XtraDB Cluster server package no longer conflicts with mysql-libs package from *CentOS* repository. Bug fixed #1278516.

The mysql-debug UNIV_DEBUG binary was missing from RPM/DEB server packages. Bug fixed #1290087.

XtraBackup SST would fail if progress option was used with large number of files. Bug fixed #1294431.

When Query Cache was used and a node would go into non-PRIM state, queries which returned results earlier (and cached into query cache) would still return results whereas newer queries (or the ones not cached) would return unknown command. Bug fixed #1296403.

Brute Force abort did not work with INSERTs to table with single unique key. Bug fixed #1299116.

InnoDB buffer pool dump and load was not working as expected due to *wsrep_recover* overwriting the buffer pool dump file. Bug fixed #1305955.

Close referenced table opened in the same function when foreign constraints were checked, otherwise it could lead to server stall when running DROP TABLE. Bug fixed #1309241.

Compiling on *FreeBSD* 10.0 with CLANG would result in fatal error. Bug fixed #1309507.

Truncating the sorted version of multi-byte character conversion could lead to wsrep certification failures. Bug fixed #1314854.

Cluster node acting as async slave would stop with the wrong position after hitting max_write_set_size. Bug fixed #1309669.

Fixed the events replication inconsistencies. Bug fixed #1312618.

wsrep_slave_threads was counted towards max_connections which could cause ERROR 1040 (HY000): Too many connections error. Bug fixed #1315588.

Leaving node was not set nonoperational if processed leave message originated from different view than the current one which could cause other nodes to crash. Bug fixed #1323412 (#41).

garb couldn't be started with init script on *RHEL* 6.5. Bug fixed #1323652.

SST would fail when binlogs were in dedicated directory that's located inside datadir. This bug was a regression introduced by bug fix for #1273368. Bug fixed #1326012.

GTID of TOI operations is now also synced to *InnoDB* tablespace in order to get consistent backups. Bug fixed #1329055.

mysql-debug is now distributed with binary tar.gz along with RPM and DEB packages. Bug fixed #1332073.

Startup failure with Undetected state gap has been fixed. Bug fixed #1334606.

Galera3 is now installed in /usr/lib/galera3/libgalera_smm.so with a compatibility symlink to /usr/lib/libgalera_smm.so. Bug fixed #1279328.

Galera could not be compiled on PowerPC. Bug fixed #59.

Cluster could stall if leaving node failed to acknowledge all messages it had received due to exception and remaining nodes failed to reach consensus because of that. Bug fixed #37.

When two node instances were set up on the same server with the two different IPs, they couldn't not work well because they were use wrong IP addresses. Bug fixed #31.

Automated donor selection with segments gave inconsistent results. Bug fixed #29.

Other bug fixes: #1297822, #1269811, #1262887, #1244835, #1338995, #11, #40, #38, #33, and #24.

Help us improve quality by reporting any bugs you encounter using our bug tracking system. As always, thanks for your continued support of Percona!

Percona XtraDB Cluster 5.6.15-25.5

Percona is glad to announce the release of *Percona XtraDB Cluster* 5.6 on March 20th 2014. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.15-63.0 including all the bug fixes in it, Galera Replicator 3.4 and on Codership wsrep API 25.5 is now the current **General Availability** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.15-25.5 milestone at Launchpad.

New Features

wsrep patch did not allow server to start with query cache enabled. This restriction and check have been removed now and query cache can be fully enabled from configuration file.

New SST options have been implemented: `inno-backup-opts`, `inno-apply-opts`, `inno-move-opts` which pass options to backup, apply and move stages of innobackupex.

The joiner would wait and not fall back to choosing other potential donor nodes (not listed in `wsrep_sst_donor`) by their state. This happened even when comma was added at the end. This fixes it for that particular case.

Initial configurable timeout, of 100 seconds, to receive a first packet via SST has been implemented, so that if donor dies somewhere in between, joiner doesn't hang. Timeout can be configured with the `sst-initial-timeout` variable.

Bugs fixed

Replication of partition tables without binlogging enabled failed, partition truncation didn't work because of lack of TO isolation there. Bug fixed #1219605.

Using `LOAD DATA INFILE` in with `autocommit` set to `0` and `wsrep_load_data_splitting` set to `ON` could lead to incomplete loading of records while chunking. Bug fixed #1281810.

`Garbd` could crash on *CentOS* if variable `gmcast.listen_addr` wasn't set. Bug fixed #1283100.

Node couldn't be started with `wsrep_provider_options` option `debug` set to `1`. Bug fixed #1285208.

Boostrapping a Node in a `NON-PRIMARY` state would lead to crash. Bug fixed #1286450.

New versions of xtrabackup SST scripts were ignoring `--socket` parameter passed by mysqld. Bug fixed #1289483.

Regression in Galera required explicitly setting *socket.ssl* to Yes even if you set up variables *socket.ssl_key* and *socket.ssl_cert*. Bug fixed #1290006.

Fixed the clang build issues that were happening during the Galera build. Bug fixed #1290462.

Better diagnostic error message has been implemented when *wsrep_max_ws_size* limit has been succeeded. Bug fixed #1280557.

Fixed incorrect warnings and implemented better handling of repeated usage with same value for *wsrep_desync*. Bug fixed #1281696.

Fixed the issue with *wsrep_slave_threads* wherein if the number of slave threads was changed before closing threads from an earlier change, it could increase the total number of threads beyond value specified in *wsrep_slave_threads*.

A regression in mutex handling caused dynamic update of *wsrep_log_conflicts* to hang the server. Bug fixed #1293624.

Presence of /tmp/test directory and an empty test database caused *Percona XtraBackup* to fail, causing SST to fail. This is an *Percona XtraBackup* issue. But, this has been fixed in PXC's xtrabackup SST separately by using unique temporary directories with *Percona XtraBackup*. Bug fixed #1294760.

After installing the auth_socket plugin any local user might get root access to the server. If you're using this plugin upgrade is advised. This is a regression, introduced in *Percona Server for MySQL* 5.6.11-60.3. Bug fixed #1289599

Other bug fixes: #1287098, #1289776, #1279343, #1259649, #1292533, #1272982, #1284670, and #1264809.

We did our best to eliminate bugs and problems during the testing release, but this is a software, so bugs are expected. If you encounter them, please report them to our bug tracking system.

Percona XtraDB Cluster 5.6.15-25.4

Percona is glad to announce the new release of *Percona XtraDB Cluster* 5.6 on February 20th 2014. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.15-63.0 including all the bug fixes in it, Galera Replicator 3.3 and on Codership wsrep API 5.6.15-25.2 is now the current **General Availability** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.15-25.4 milestone at Launchpad.

Bugs fixed

Parallel Applying was not functioning which was evident from the *wsrep_cert_deps_distance* being 1.0 at all times. Bug fixed #1277703.

Binlog events were created for the statements for non-InnoDB tables, but they were never cleaned from transaction cache, which could lead to node crash. Bug fixed #1277986.

Percona XtraDB Cluster didn't validate the parameters of *wsrep_provider_options* when starting it up. Bug fixed #1260193.

clustercheck script would mark node as down on *Debian* based systems if it was run with default values because it was looking for the defaults-extra-file in the wrong directory. Bug fixed #1276076.

Deadlock would happen when NULL unique key was inserted. Workaround has been implemented to support NULL keys, by using the md5 sum of full row as key value. Bug fixed #1276424.

Variables innodb-log-group-home-dir and innodb-data-home-dir are now handled by default (ie., there is no need to set them up in sst_special_dirs). Bug fixed #1276904.

Builds now use system Zlib instead of bundled one. Bug fixed #1277928.

If transaction size exceeds the *wsrep_max_ws_size* limit, there will appear a warning message in the error log and replication is skipped. However, the transaction was committed in the master node, and cluster would be in inconsistent state. Bug fixed #1280557.

wsrep_load_data_splitting defaults to OFF now, using it turned ON with autocommit set to 0 is not recommended.

Other bugs fixed: #1279844.

We did our best to eliminate bugs and problems during the testing release, but this is a software, so bugs are expected. If you encounter them, please report them to our bug tracking system.

Percona XtraDB Cluster Errata can be found in our documentation.

Percona XtraDB Cluster 5.6.15-25.3

Percona is glad to announce the first General Availability release of *Percona XtraDB Cluster* 5.6 on January 30th 2014. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.15-63.0 including all the bug fixes in it, Galera Replicator 3.3 and on Codership wsrep API 5.6.15-25.2 is now the first **General Availability** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.15-25.3 milestone at Launchpad.

New Features

New meta packages are now available in order to make the *Percona XtraDB Cluster installation* easier.

Debian/*Ubuntu* debug packages are now available for Galera and garbd.

xtrabackup-v2 SST now supports the GTID replication.

Bugs fixed

Node would get stuck and required restart if DDL was performed after FLUSH TABLES WITH READ LOCK. Bug fixed #1265656.

Galera provider pause has been fixed to avoid potential deadlock with replicating threads.

Default value for binlog_format is now ROW. This is done so that *Percona XtraDB Cluster* is not started with wrong defaults leading to non-deterministic outcomes like crash. Bug fixed #1243228.

During the installation of percona-xtradb-cluster-garbd-3.x package, *Debian* tries to start it, but as the configuration is not set, it would fail to start and leave the installation in iF state. Bug fixed #1262171.

Runtime checks have been added for dynamic variables which are Galera incompatible. Bug fixed #1262188.

During the upgrade process, parallel applying could hit an unresolvable conflict when events were replicated from *Percona XtraDB Cluster* 5.5 to *Percona XtraDB Cluster* 5.6. Bug fixed #1267494.

xtrabackup-v2 is now used as default *SST* method in *wsrep_sst_method*. Bug fixed #1268837.

FLUSH TABLES WITH READ LOCK behavior on the same connection was changed to conform to *MySQL* behavior. Bug fixed #1269085.

Read-only detection has been added in clustercheck, which can be helpful during major upgrades (this is used by xinetd for HAProxy etc.) Bug fixed #1269469.

Binary log directory is now being cleanup as part of the *XtraBackup SST*. Bug fixed #1273368.

First connection would hang after changing the *wsrep_cluster_address* variable. Bug fixed #1022250.

When *gmcast.listen_addr* was set manually it did not allow nodes own address in gcomm address list. Bug fixed #1099478.

GCache file allocation could fail if file size was a multiple of page size. Bug fixed #1259952.

Group remerge after partitioning event has been fixed. Bug fixed #1232747.

Fixed the OpenSSL linking exceptions. Bug fixed #1259063.

Fixed multiple build bugs: #1262716, #1269063, #1269351, #1272723, #1272732, and #1261996.

Other bugs fixed: #1273101, #1272961, #1271264, and #1253055.

We did our best to eliminate bugs and problems during the testing release, but this is a software, so bugs are expected. If you encounter them, please report them to our bug tracking system.

Percona XtraDB Cluster 5.6.15-25.2

Percona is glad to announce the first Release Candidate release of *Percona XtraDB Cluster* 5.6 on December 18th 2013. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.15-63.0 including all the bug fixes in it, Galera Replicator 3.2 and on Codership wsrep API 5.6.15-25.2 is now the first **RELEASE CANDIDATE** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.15-25.2 milestone at Launchpad.

This release contains all of the features and bug fixes in Percona XtraDB Cluster 5.5.34-25.9, plus the following:

New Features

Percona XtraDB Cluster now supports stream compression/decompression with new xtrabackup-sst compressor/decompressor options.

New *wsrep_reject_queries* has been implemented that can be used to reject queries for that node. This can be useful if someone wants to manually run maintenance on the node like mysqldump without need to change the settings on the load balancer.

XtraBackup SST now supports innodb_data_home_dir and innodb_log_group_home_dir in the configuration file with sst_special_dirs option.

New *wsrep_local_cached_downto* status variable has been introduced. This variable shows the lowest sequence number in gcache. This information can be helpful with determining IST and/or SST.

Garbd init script and configuration files have been packaged for *CentOS* and *Debian*, in addition, in *Debian* garbd is packaged separately in percona-xtradb-cluster-garbd-3.x package.

Bugs fixed

When `grastate.dat` file was not getting zeroed appropriately it would lead to RBR error during the IST. Bug fixed #1180791.

`init stop` script on *CentOS* didn't wait for the server to be fully stopped. This would cause unsuccessful server restart because the `start` action would fail because the daemon would still be running. Bug fixed #1254153.

`DELETE FROM` statement (without `WHERE` clause) would crash slaves if master did not have binlogging enabled. Bug fixed #1254179.

Missing protection for brute force threads against `innodb lock wait time out` would cause applier to fail with lock wait timeout exceeded on `rsync` SST donor. Bug fixed #1255501.

Recent optimizations in 3.x branch introduced a regression in base filename construction which could lead big transactions fail with: `WSREP: Failed to open file '...'`. Bug fixed #1255964.

Joiner node would not initialize storage engines if `rsync` was used for SST and the first view was non-primary. Bug fixed #1257341.

Table level lock conflict resolving was releasing the wrong lock. Bug fixed #1257678.

Resolved the `perl` dependencies needed for *Percona XtraDB Cluster* 5.6. Bug fixed #1258563.

Obsolete dependencies have been removed from *Percona XtraDB Cluster*. Bug fixed #1259256.

`CREATE TABLE AS SELECT` process would remain hanging in case it was run in parallel with the DDL statement on the selected table. Bug fixed #1164893.

Naming of the *Galera* packages have been fixed to avoid the confusion, ie. `Percona-XtraDB-Cluster-galera-56` is `Percona-XtraDB-Cluster-galera-3` now. Bug fixed #1253923.

Fixed rsync SST for compatibility with `rsync` version 3.1.0. Bug fixed #1261673.

Other bugs fixed: #1261138, #1254633.

Note: On *CentOS* 5/6, those who may have installed `percona-xtrabackup-20` with *Percona XtraDB Cluster* 5.6 due to bug #1226185, the upgrade may fail, the dependency issue has been fixed since then, hence replace `percona-xtrabackup-20` with `percona-xtrabackup` before upgrade.

We did our best to eliminate bugs and problems during the testing release, but this is a software, so bugs are expected. If you encounter them, please report them to our bug tracking system.

Percona XtraDB Cluster 5.6.14-25.1

Percona is glad to announce the first Beta release of *Percona XtraDB Cluster* 5.6 on November 21st, 2013. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.6.14-62.0 including all the bug fixes in it, Galera Replicator 3.1 and on Codership wsrep API 5.6.14-25.1 is now the first **BETA** release. All of *Percona*'s software is open-source and free, all the details of the release can be found in the 5.6.14-25.1 milestone at Launchpad.

This release contains all of the features and bug fixes in Percona XtraDB Cluster 5.5.34-23.7.6, plus the following:

New Features

Percona XtraDB Cluster is now using Galera Replicator 3.1 and wsrep API 5.6.14-25.1.

Percona XtraDB Cluster has implemented a number of XtraDB performance improvements for I/O-bound high-concurrency workloads.

Percona XtraDB Cluster has implemented a number of performance improvements for Page cleaner thread tuning.

`ALL_O_DIRECT` method for innodb_flush_method has been ported from 5.5 version.

Statement Timeout feature has been ported from the Twitter branch.

Percona XtraDB Cluster has extended the `SELECT INTO ... OUTFILE` and `SELECT INTO DUMPFILE` to add the support for `UNIX` sockets and named pipes.

Percona XtraDB Cluster has implemented more efficient log block checksums with new innodb_log_checksum_algorithm variable.

Percona XtraDB Cluster now supports Per-query variable statements.

Limited support for Query Cache has been implemented. Query cache cannot still be fully enabled during the startup. To enable query cache, `mysqld` should be started with `query_cache_type=1` and `query_cache_size=0` and then query_cache_size should be changed to desired value during runtime.

`RPM` packages are now made relocatable which means they now support installation to custom prefixes.

Features from Galera

Following are salient features of Galera 3:

- new writeset format optimized for performance and reduced memory usage.
- 128-bit writeset checksums, checked every time before writeset is applied, so that corruption cannot sneak in neither on disk, nor in transfer.
- hardware accelerated CRC32-C algorithm for network packets (and ability to turn it off completely). Parameter: socket.checksum.
- improved handling of preordered (read: master-slave replication) events, that preserves original event ID and offers better throughput.
- ability to divide cluster into segments with minimized communication between them to minimize network traffic over WAN. Parameter: gmcast.segment.

Bugs fixed

Some wsrep variables (*`wsrep_provider`*, *`wsrep_provider_options`*, *`wsrep_cluster_address`*...) could be "doubly" allocated which caused memory leaks. Bug fixed #1072839.

If `SELECT FOR UPDATE...` query was aborted due to multi-master conflict, the client wouldn't get back the deadlock error. From client perspective the transaction would be successful. Bug fixed #1187739.

Temporary tables are not replicated, but any DDL on those tables were, which would generates error messages on other nodes. Bugs fixed #1194156, #1084702, #1212247.

When setting the *`gcache.size`* to a larger value than the default 128M, the mysql service command did not allow enough time for the file to be preallocated. Bug fixed #1207500.

When installing first `Percona-XtraDB-Cluster-client` and then `Percona-XtraDB-Cluster-server` on two single statements or a single statement with both packages , yum would install `percona-xtrabackup-20` instead `percona-xtrabackup` package as dependency of `Percona-XtraDB-Cluster-server`. Bug fixed #1226185.

Different mutex implementation in the 5.6 could lead to server assertion error. Bug fixed #1233383.

Enabling *`wsrep_log_conflicts`* variable could cause issues with `lock_mutex`. Bug fixed #1234382.

Server could freeze with mixed DML/DDL load because TOI brute force aborts were not properly propagated. Bug fixed #1239571.

`CREATE TABLE AS SELECT` would fail with explicit temporary tables, when binlogging was enabled and `autocommit` was set to `0`. Bug fixed #1240098.

Transaction cleanup function did not get called for autocommit statements after rollback, it would stay in `LOCAL_COMMIT` even after rollback finished which caused problems when the next transaction started. Bug fixed #1240567.

`DDL` statements like `CREATE TEMPORARY TABLE LIKE` would be replicated and applied in all cluster nodes. This caused temporary table definitions to pile up in slave threads. Bug fixed #1246257.

`CREATE TABLE AS SELECT` was not replicated, if the select result set was empty. Bug fixed #1246921.

Write set flags defined in wsrep API are now exposed to application side appliers too. Bug fixed #1247402.

Local brute force aborts are counted accurately. Bug fixed #1247971.

Certain combinations of transaction rollbacks could leave stale transactional `MDL` locks. Bug fixed #1247978.

After turning `UNIV_SYNC_DEBUG` on, node that was started from clean state would crash immediately at startup. Bug fixed #1248908.

Server built with `UNIV_SYNC_DEBUG` would assert if SQL load has `DELETE` statements on tables with foreign key constraints with `ON DELETE CASCADE` option. Bug fixed #1248921.

Xtrabackup SST dependencies have been added as Optional/Recommended/Suggested dependencies. Bug fixed #1250326.

Other bugs fixed: bug fixed #1020457, bug fixed #1250865, bug fixed #1249753, bug fixed #1248396, bug fixed #1247980, bug fixed #1238278, bug fixed #1234421, bug fixed #1228341, bug fixed #1228333, bug fixed #1225236, bug fixed #1221354, bug fixed #1217138, bug fixed #1206648, bug fixed #1200647, bug fixed #1180792, bug fixed #1163283, bug fixed #1234229, bugs fixed #1250805, bug fixed #1233301, and bug fixed #1210509.

We did our best to eliminate bugs and problems during the testing release, but this is a software, so bugs are expected. If you encounter them, please report them to our bug tracking system.

CHAPTER

TWENTYSEVEN

INDEX OF WSREP STATUS VARIABLES

variable `wsrep_apply_oooe`

This variable shows parallelization efficiency, how often writests have been applied out of order.

variable `wsrep_apply_oool`

This variable shows how often a writeset with a higher sequence number was applied before one with a lower sequence number.

variable `wsrep_apply_window`

Average distance between highest and lowest concurrently applied sequence numbers.

variable `wsrep_causal_reads_`

Shows the number of writesets processed while the variable *`wsrep_causal_reads`* was set to `ON`.

variable `wsrep_cert_bucket_count`

This variable, implemented in *`5.6.24-25.11`*, shows the number of cells in the certification index hash-table.

variable `wsrep_cert_deps_distance`

Average distance between highest and lowest sequence number that can be possibly applied in parallel.

variable `wsrep_cert_index_size`

Number of entries in the certification index.

variable `wsrep_cluster_conf_id`

Number of cluster membership changes that have taken place.

variable `wsrep_cluster_size`

Current number of nodes in the cluster.

variable `wsrep_cluster_state_uuid`

This variable contains *UUID* state of the cluster. When this value is the same as the one in *`wsrep_local_state_uuid`*, node is synced with the cluster.

variable `wsrep_cluster_status`

Status of the cluster component. Possible values are:

- `Primary` - Node has a quorum,
- `Non-Primary` - Node has lost a quorum (for example 2 out 4 nodes get partitioned and see each other but don't have a quorum),
- `Disconnected` - Node is unable to connect to other nodes/cluster.

variable `wsrep_commit_oooe`

This variable shows how often a transaction was committed out of order.

variable `wsrep_commit_oool`

This variable currently has no meaning.

variable `wsrep_commit_window`

Average distance between highest and lowest concurrently committed sequence number.

variable `wsrep_connected`

This variable shows if the node is connected to the cluster. If the value is `OFF`, the node has not yet connected to any of the cluster components. This may be due to misconfiguration.

variable `wsrep_desync_count`

Number of desync operations currently in progress on the node. This status variable increments every time a node is desynced from the cluster (for example, when *`wsrep_desync`* is enabled). The node can only sync to the cluster when the value is `0`, meaning that there are no operations running on the node that require the node to desync.

variable `wsrep_evs_delayed`

Comma separated list of nodes that are considered delayed. The node format is `<uuid>:<address>:<count>`, where `<count>` is the number of entries on delayed list for that node.

variable `wsrep_evs_evict_list`

List of UUIDs of the evicted nodes.

variable `wsrep_evs_repl_latency`

This status variable provides information regarding group communication replication latency. This latency is measured in seconds from when a message is sent out to when a message is received.

The format of the output is `<min>/<avg>/<max>/<std_dev>/<sample_size>`.

variable `wsrep_evs_state`

Internal EVS protocol state.

variable `wsrep_flow_control_paused`

Time since the last status query that was paused due to flow control.

variable `wsrep_flow_control_paused_ns`

Total time spent in a paused state measured in nanoseconds.

variable `wsrep_flow_control_recv`

Number of `FC_PAUSE` events received since the last status query.

variable `wsrep_flow_control_sent`

Number of `FC_PAUSE` events sent since the last status query.

variable `wsrep_gcache_pool_size`

This variable, implemented in *`5.6.24-25.11`*, shows the size of the page pool and dynamic memory allocated for GCache (in bytes).

variable `wsrep_gcomm_uuid`

This status variable exposes UUIDs in `gvwstate.dat`, which are Galera view IDs (thus unrelated to cluster state UUIDs). This UUID is unique for each node. You will need to know this value when using manual eviction feature.

variable `wsrep_incoming_addresses`

Shows the comma-separated list of incoming node addresses in the cluster.

variable `wsrep_last_committed`

Sequence number of the last committed transaction.

variable `wsrep_local_bf_aborts`

Number of local transactions that were aborted by slave transactions while being executed.

variable `wsrep_local_cached_downto`

The lowest sequence number in GCache. This information can be helpful with determining IST and SST. If the value is `0`, then it means there are no writesets in GCache (usual for a single node).

variable `wsrep_local_cert_failures`

Number of writesets that failed the certification test.

variable `wsrep_local_commits`

Number of writesets commited on the node.

variable `wsrep_local_index`

Node's index in the cluster.

variable `wsrep_local_recv_queue`

Current length of the receive queue (that is, the number of writesets waiting to be applied).

variable `wsrep_local_recv_queue_avg`

Average length of the receive queue since the last status query. When this number is bigger than `0` this means node can't apply writesets as fast as they are received. This could be a sign that the node is overloaded and it may cause replication throttling.

variable `wsrep_local_replays`

Number of transaction replays due to *asymmetric lock granularity*.

variable `wsrep_local_send_queue`

Current length of the send queue (that is, the number of writesets waiting to be sent).

variable `wsrep_local_send_queue_avg`

Average length of the send queue since the last status query. When cluster experiences network throughput issues or replication throttling, this value will be significantly bigger than `0`.

variable `wsrep_local_state`

variable `wsrep_local_state_comment`

Internal number and the corresponding human-readable comment of the node's state. Possible values are:

Num	Comment	Description
1	Joining	Node is joining the cluster
2	Donor/Desynced	Node is the donor to the node joining the cluster
3	Joined	Node has joined the cluster
4	Synced	Node is synced with the cluster

variable `wsrep_local_state_uuid`

The *UUID* of the state stored on the node.

variable `wsrep_protocol_version`

Version of the wsrep protocol used.

variable **`wsrep_provider_name`**

Name of the wsrep provider (usually `Galera`).

variable **`wsrep_provider_vendor`**

Name of the wsrep provider vendor (usually `Codership Oy`)

variable **`wsrep_provider_version`**

Current version of the wsrep provider.

variable **`wsrep_ready`**

This variable shows if node is ready to accept queries. If status is `OFF`, almost all queries will fail with `ERROR 1047 (08S01) Unknown Command` error (unless the *`wsrep_on`* variable is set to `0`)

variable **`wsrep_received`**

Total number of writesets received from other nodes.

variable **`wsrep_received_bytes`**

Total size (in bytes) of writesets received from other nodes.

variable **`wsrep_repl_data_bytes`**

Total size (in bytes) of data replicated.

variable **`wsrep_repl_keys`**

Total number of keys replicated.

variable **`wsrep_repl_keys_bytes`**

Total size (in bytes) of keys replicated.

variable **`wsrep_repl_other_bytes`**

Total size of other bits replicated.

variable **`wsrep_replicated`**

Total number of writesets sent to other nodes.

variable **`wsrep_replicated_bytes`**

Total size (in bytes) of writesets sent to other nodes.

CHAPTER

TWENTYEIGHT

INDEX OF WSREP SYSTEM VARIABLES

Percona XtraDB Cluster introduces a number of MySQL system variables related to write-set replication.

variable `wsrep_auto_increment_control`

Command Line `--wsrep-auto-increment-control`

Config File Yes

Scope Global

Dynamic Yes

Default Value `ON`

Enables automatic adjustment of auto-increment system variables depending on the size of the cluster:

- `auto_increment_increment` controls the interval between successive `AUTO_INCREMENT` column values
- `auto_increment_offset` determines the starting point for the `AUTO_INCREMENT` column value

This helps prevent auto-increment replication conflicts across the cluster by giving each node its own range of auto-increment values. It is enabled by default.

Automatic adjustment may not be desirable depending on application's use and assumptions of auto-increments. It can be disabled in master-slave clusters.

variable `wsrep_causal_reads`

Version Info

- ***5.6.20-25.7*** – Variable deprecated

Command Line `--wsrep-causal-reads`

Config File Yes

Scope Global, Session

Dynamic Yes

Default Value `OFF`

In some cases, master may apply events faster than a slave, which can cause master and slave to become out of sync for a brief moment. When this variable is set to `ON`, the slave will wait until that event is applied before doing any other queries. Enabling this variable will also result in larger latencies.

Note: This variable was deprecated because enabling it is the equivalent of setting *`wsrep_sync_wait`* to `1`.

variable `wsrep_certify_nonPK`

Command Line `--wsrep-certify-nonpk`

Config File Yes

Scope Global

Dynamic Yes

Default Value `ON`

Enables automatic generation of primary keys for rows that don't have them. Write set replication requires primary keys on all tables to allow for parallel applying of transactions. This variable is enabled by default. As a rule, make sure that all tables have primary keys.

variable `wsrep_cluster_address`

Command Line `--wsrep-cluster-address`

Config File Yes

Scope Global

Dynamic Yes

Defines the back-end schema, IP addresses, ports, and options that the node uses when connecting to the cluster. This variable needs to specify at least one other node's address, which is alive and a member of the cluster. In practice, it is best (but not necessary) to provide a complete list of all possible cluster nodes. The value should be of the following format:

```
<schema>://<address>[?<option1>=<value1>[&<option2>=<value2>]],...
```

The only back-end schema currently supported is `gcomm`. The IP address can contain a port number after a colon. Options are specified after `?` and separated by `&`. You can specify multiple addresses separated by commas.

For example:

```
wsrep_cluster_address="gcomm://192.168.0.1:4567?gmcast.listen_addr=0.0.0.0:5678"
```

If an empty `gcomm://` is provided, the node will bootstrap itself (that is, form a new cluster). It is not recommended to have empty cluster address in production config after the cluster has been bootstrapped initially. If you want to bootstrap a new cluster with a node, you should pass the `--wsrep-new-cluster` option when starting.

variable `wsrep_cluster_name`

Command Line `--wsrep-cluster-name`

Config File Yes

Scope Global

Dynamic Yes

Default Value `my_wsrep_cluster`

Specifies the name of the cluster and should be identical on all nodes.

Note: It should not exceed 32 characters.

variable `wsrep_convert_lock_to_trx`

Command Line `--wsrep-convert-lock-to-trx`

Config File Yes

Scope Global

Dynamic Yes

Default Value `OFF`

Defines whether locking sessions should be converted into transactions. By default, this is disabled.

Enabling this variable can help older applications to work in a multi-master setup by converting `LOCK/UNLOCK TABLES` statements into `BEGIN/COMMIT` statements. It is not the same as support for locking sessions, but it does prevent the database from ending up in a logically inconsistent state. Enabling this variable can also result in having huge write-sets.

variable `wsrep_data_home_dir`

Command Line No

Config File Yes

Scope Global

Dynamic No

Default Value `/var/lib/mysql` (or whatever path is specified by *datadir*)

Specifies the path to the directory where the wsrep provider stores its files (such as `grastate.dat`).

variable `wsrep_dbug_option`

Command Line `--wsrep-dbug-option`

Config File Yes

Scope Global

Dynamic Yes

Defines `DBUG` options to pass to the wsrep provider.

variable `wsrep_debug`

Command Line `--wsrep-debug`

Config File Yes

Scope Global

Dynamic Yes

Default Value `OFF`

Enables additional debugging output for the database server error log. By default, it is disabled. This variable can be used when trying to diagnose problems or when submitting a bug.

Note: Do not enable debugging in production environments, because it logs authentication info (that is, passwords).

variable `wsrep_desync`

Command Line No

Config File Yes

Scope Global

Dynamic Yes

Default Value `OFF`

Defines whether the node should participate in Flow Control. By default, this variable is disabled, meaning that if the receive queue becomes too big, the node engages in Flow Control: it works through the receive queue until it reaches a more manageable size. For more information, see *wsrep_local_recv_queue* and `wsrep_flow_control_interval`.

Enabling this variable will disable Flow Control for the node. It will continue to receive write-sets that it is not able to apply, the receive queue will keep growing, and the node will keep falling behind the cluster indefinitely.

Toggling this back to `OFF` will require an IST or an SST, depending on how long it was desynchronized. This is similar to cluster desynchronization, which occurs during RSU TOI. Because of this, it's not a good idea to enable `wsrep_desync` for a long period of time or for several nodes at once.

Note: You can also desync a node using the `/*! WSREP_DESYNC */` query comment.

variable **`wsrep_dirty_reads`**

Version Info

- ***`5.6.24-25.11`*** – Variable introduced
- ***`5.6.26-25.12`*** – Variable available both on session and global scope

Command Line `--wsrep-dirty-reads`

Config File Yes

Scope Session, Global

Dynamic Yes

Default Value `OFF`

Defines whether the node accepts read queries when in a non-operational state, that is, when it loses connection to the Primary Component. By default, this variable is disabled and the node rejects all queries, because there is no way to tell if the data is correct.

If you enable this variable, the node will permit read queries (`USE`, `SELECT`, `LOCK TABLE`, and `UNLOCK TABLES`), but any command that modifies or updates the database on a non-operational node will still be rejected (including DDL and DML statements, such as `INSERT`, `DELETE`, and `UPDATE`).

To avoid deadlock errors, set the *wsrep_sync_wait* variable to `0` if you enable `wsrep_dirty_reads`.

variable **`wsrep_drupal_282555_workaround`**

Command Line `--wsrep-drupal-282555-workaround`

Config File Yes

Scope Global

Dynamic Yes

Default Value `OFF`

Enables a workaround for MySQL InnoDB bug that affects Drupal (Drupal bug #282555 and MySQL bug #41984). In some cases, duplicate key errors would occur when inserting the `DEFAULT` value into an `AUTO_INCREMENT` column.

variable **`wsrep_forced_binlog_format`**

Command Line `--wsrep-forced-binlog-format`

Config File Yes

Scope Global

Dynamic Yes

Default Value `NONE`

Defines a binary log format that will always be effective, regardless of the client session `binlog_format` variable value.

Possible values for this variable are:

- `ROW`: Force row-based logging format
- `STATEMENT`: Force statement-based logging format
- `MIXED`: Force mixed logging format
- `NONE`: Do not force the binary log format and use whatever is set by the `binlog_format` variable (default)

variable `wsrep_load_data_splitting`

Command Line `--wsrep-load-data-splitting`

Config File Yes

Scope Global

Dynamic Yes

Default Value `ON`

Defines whether the node should split large `LOAD DATA` transactions. This variable is enabled by default, meaning that `LOAD DATA` commands are split into transactions of 10 000 rows or less.

If you disable this variable, then huge data loads may prevent the node from completely rolling the operation back in the event of a conflict, and whatever gets committed stays committed.

Note: It doesn't work as expected with `autocommit=0` when enabled.

variable `wsrep_log_conflicts`

Command Line `--wsrep-log-conflicts`

Config File Yes

Scope Global

Dynamic No

Default Value `OFF`

Defines whether the node should log additional information about conflicts. By default, this variable is disabled and *Percona XtraDB Cluster* uses standard logging features in MySQL.

If you enable this variable, it will also log table and schema where the conflict occurred, as well as the actual values for keys that produced the conflict.

variable `wsrep_max_ws_rows`

Command Line `--wsrep-max-ws-rows`

Config File Yes

Scope Global

Dynamic Yes

Default Value `0` (no limit)

Defines the maximum number of rows each write-set can contain.

By default, there is no limit for the maximum number of rows in a write-set. The maximum allowed value is `1048576`.

variable **`wsrep_max_ws_size`**

> **Command Line** `--wsrep_max_ws_size`
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** Yes
>
> **Default Value** `2147483647` (2 GB)

Defines the maximum write-set size (in bytes). Anything bigger than the specified value will be rejected.

You can set it to any value between `1024` and the default `2147483647`.

variable **`wsrep_mysql_replication_bundle`**

> **Command Line** `--wsrep-mysql-replication-bundle`
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** `0` (no grouping)
>
> **Range** From `0` to `1000`

Specifies the number of replication events to group together. By default, it is set to `0`, which means there is no grouping.

Replication events are grouped in SQL slave thread by skipping events, which may cause commit. In this case, the wsrep node acting as a slave node and all other wsrep nodes in provider replication group, will see the same (huge) transactions. The implementation of grouping is experimental. It may help with the bottleneck of having only one slave facing commit time delay of synchronous provider. You can set it up to group up to 1000 events.

Note: This feature and variable is not available in latest versions of *Percona XtraDB Cluster* 5.7.

variable **`wsrep_node_address`**

> **Command Line** `--wsrep-node-address`
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** IP of the first network interface (`eth0`) and default port (`4567`)

Specifies the network address of the node. By default, this variable is set to the IP address of the first network interface (usually `eth0` or `enp2s0`) and the default port (`4567`).

While default value should be correct in most cases, there are situations when you need to specify it manually. For example:

- Servers with multiple network interfaces
- Servers that run multiple nodes

- Network Address Translation (NAT)
- Clusters with nodes in more than one region
- Container deployments, such as Docker
- Cloud deployments, such as Amazon EC2 (use the global DNS name instead of the local IP address)

The value should be specified in the following format:

```
<ip_address>[:port]
```

For example:

```
192.168.0.1:4567
```

Note: The value of this variable is also used as the default value for the *wsrep_sst_receive_address* variable and the *ist.recv_addr* option.

---

variable wsrep_node_incoming_address

Command Line `--wsrep-node-incoming-address`

Config File Yes

Scope Global

Dynamic No

Default Value `AUTO`

Specifies the network address from which the node expects client connections. By default, it uses the IP address from *wsrep_node_address* and port number 3306.

This information is used for the *wsrep_incoming_addresses* variable, which shows all active cluster nodes.

variable wsrep_node_name

Command Line `--wsrep-node-name`

Config File Yes

Scope Global

Dynamic Yes

Default Value The node's host name

Defines a unique name for the node. Defaults to the host name.

In many situations, you may use the value of this variable as a means to identify the given node in the cluster as the alternative to using the node address (the value of the *wsrep_node_address*).

Note: The variable *wsrep_sst_donor* is an example where you may only use the value of *wsrep_node_name* and the node address is not permitted.

---

variable wsrep_notify_cmd

Command Line `--wsrep-notify-cmd`

Config File Yes

Scope Global

Dynamic Yes

Specifies the notification command that the node should execute whenever cluster membership or local node status changes. This can be used for alerting or to reconfigure load balancers.

Note: The node will block and wait until the command or script completes and returns before it can proceed. If the script performs any potentially blocking or long-running operations, such as network communication, you should consider initiating such operations in the background and have the script return immediately.

variable `wsrep_on`

Version Info

- ***`5.6.27-25.13`*** – Variable available only in session scope

Command Line No

Config File No

Scope Session

Dynamic Yes

Default Value `ON`

Defines whether updates from the current session should be replicated. If disabled, it does not cause the node to leave the cluster and the node continues to communicate with other nodes.

variable `wsrep_OSU_method`

Version Info

- ***`5.6.24-25.11`*** – Variable available both in global and session scope

Command Line `--wsrep-OSU-method`

Config File Yes

Scope Global and Session

Dynamic Yes

Default Value `TOI`

Defines the method for Online Schema Upgrade that the node uses to replicate DDL statements. The following methods are available:

- `TOI`: When the *Total Order Isolation* method is selected, data definition language (DDL) statements are processed in the same order with regards to other transactions in each node. This guarantees data consistency.

  In the case of DDL statements, the cluster will have parts of the database locked and it will behave like a single server. In some cases (like big `ALTER TABLE`) this could have impact on cluster's performance and availability, but it could be fine for quick changes that happen almost instantly (like fast index changes).

  When DDL statements are processed under TOI, the DDL statement will be replicated up front to the cluster. That is, the cluster will assign global transaction ID for the DDL statement before DDL processing begins. Then every node in the cluster has the responsibility to execute the DDL statement in the given slot in the sequence of incoming transactions, and this DDL execution has to happen with high priority.

- `RSU`: When the *Rolling Schema Upgrade* method is selected, DDL statements won't be replicated across the cluster. Instead, it's up to the user to run them on each node separately.

  The node applying the changes will desynchronize from the cluster briefly, while normal work happens on all the other nodes. When a DDL statement is processed, the node will apply delayed replication events.

The schema changes must be backwards compatible for this method to work, otherwise, the node that receives the change will likely break Galera replication. If replication breaks, SST will be triggered when the node tries to join again but the change will be undone.

Note: This variable's behavior is consistent with MySQL behavior for variables that have both global and session scope. This means if you want to change the variable in current session, you need to do it with `SET wsrep_OSU_method` (without the `GLOBAL` keyword). Setting the variable with `SET GLOBAL wsrep_OSU_method` will change the variable globally but it won't have effect on the current session.

variable `wsrep_preordered`

Command Line `--wsrep-preordered`

Config File Yes

Scope Global

Dynamic Yes

Default Value `OFF`

Defines whether the node should use transparent handling of preordered replication events (like replication from traditional master). By default, this is disabled.

If you enable this variable, such events will be applied locally first before being replicated to other nodes in the cluster. This could increase the rate at which they can be processed, which would be otherwise limited by the latency between the nodes in the cluster.

Preordered events should not interfere with events that originate on the local node. Therefore, you should not run local update queries on a table that is also being updated through asynchronous replication.

variable `wsrep_provider`

Command Line `--wsrep-provider`

Config File Yes

Scope Global

Dynamic Yes

Specifies the path to the Galera library. This is usually `/usr/lib64/libgalera_smm.so` on *CentOS/RHEL* and `/usr/lib/libgalera_smm.so` on *Debian/Ubuntu*.

If you do not specify a path or the value is not valid, the node will behave as standalone instance of MySQL.

variable `wsrep_provider_options`

Command Line `--wsrep-provider-options`

Config File Yes

Scope Global

Dynamic No

Specifies optional settings for the replication provider documented in *Index of wsrep_provider options*. These options affect how various situations are handled during replication.

variable `wsrep_recover`

Command Line `--wsrep-recover`

Config File Yes

Scope Global

Dynamic No

Default Value `OFF`

Location `mysqld_safe`

Recovers database state after crash by parsing GTID from the log. If the GTID is found, it will be assigned as the initial position for server.

variable **`wsrep_reject_queries`**

Command Line No

Config File Yes

Scope Global

Dynamic Yes

Default Value `NONE`

Defines whether the node should reject queries from clients. Rejecting queries can be useful during upgrades, when you want to keep the node up and apply write-sets without accepting queries.

When a query is rejected, the following error is returned:

```
Error 1047: Unknown command
```

The following values are available:

- `NONE`: Accept all queries from clients (default)
- `ALL`: Reject all new queries from clients, but maintain existing client connections
- `ALL_KILL`: Reject all new queries from clients and kill existing client connections

Note: This variable doesn't affect Galera replication in any way, only the applications that connect to the database are affected. If you want to desync a node, use *`wsrep_desync`*.

variable **`wsrep_replicate_myisam`**

Version Info

- ***`5.6.24-25.11`*** – Variable available both in global and session scope

Command Line `--wsrep-replicate-myisam`

Config File Yes

Scope Session, Global

Dynamic No

Default Value `OFF`

Defines whether DML statements for MyISAM tables should be replicated. It is disabled by default, because MyISAM replication is still experimental.

On the global level, *`wsrep_replicate_myisam`* can be set only during startup. On session level, you can change it during runtime as well.

For older nodes in the cluster, *`wsrep_replicate_myisam`* should work since the TOI decision (for MyISAM DDL) is done on origin node. Mixing of non-MyISAM and MyISAM tables in the same DDL statement is not recommended when *wsrep_replicate_myisam* is disabled, since if any table in the list is MyISAM, the whole DDL statement is not put under TOI.

Note: You should keep in mind the following when using MyISAM replication:

- DDL (CREATE/DROP/TRUNCATE) statements on MyISAM will be replicated irrespective of `wsrep_replicate_miysam` value
- DML (INSERT/UPDATE/DELETE) statements on MyISAM will be replicated only if *wsrep_replicate_myisam* is enabled
- SST will get full transfer irrespective of *wsrep_replicate_myisam* value (it will get MyISAM tables from donor)
- Difference in configuration of `pxc-cluster` node on enforce_storage_engine front may result in picking up different engine for the same table on different nodes
- `CREATE TABLE AS SELECT` (CTAS) statements use non-TOI replication and are replicated only if there is involvement of InnoDB table that needs transactions (in case of MyISAM table, CTAS statements will not be replicated).

variable `wsrep_restart_slave`

Command Line `--wsrep-restart-slave`

Config File Yes

Scope Global

Dynamic Yes

Default Value `OFF`

Defines whether replication slave should be restarted when the node joins back to the cluster. Enabling this can be useful because asynchronous replication slave thread is stopped when the node tries to apply the next replication event while the node is in non-primary state.

variable `wsrep_retry_autocommit`

Command Line `--wsrep-retry-autocommit`

Config File Yes

Scope Global

Dynamic No

Default Value `1`

Specifies the number of times autocommit transactions will be retried in the cluster if it encounters certification errors. In case there is a conflict, it should be safe for the cluster node to simply retry the statement without returning an error to the client, hoping that it will pass next time.

This can be useful to help an application using autocommit to avoid deadlock errors that can be triggered by replication conflicts.

If this variable is set to `0`, autocommit transactions won't be retried.

variable `wsrep_RSU_commit_timeout`

Command Line `--wsrep-RSU-commit-timeout`

Config File Yes

Scope Global

Dynamic Yes

Default Value `5000`

Range From `5000` (5 millisecons) to `31536000000000` (365 days)

Specifies the timeout in microseconds to allow active connection to complete COMMIT action before starting RSU.

While running RSU it is expected that user has isolated the node and there is no active traffic executing on the node. RSU has a check to ensure this, and waits for any active connection in `COMMIT` state before starting RSU.

By default this check has timeout of 5 millisecons, but in some cases COMMIT is taking longer. This variable sets the timeout, and has allowed values from the range of (5 millisecons, 365 days). The value is to be set in microseconds. Unit of variable is in micro-secs so set accordingly.

Note: RSU operation will not auto-stop node from receiving active traffic. So there could be a continuous flow of active traffic while RSU continues to wait, and that can result in RSU starvation. User is expected to block active RSU traffic while performing operation.

---

variable `wsrep_slave_FK_checks`

Command Line `--wsrep-slave-FK-checks`

Config File Yes

Scope Global

Dynamic Yes

Default Value `ON`

Defines whether foreign key checking is done for applier threads. This is enabled by default.

variable `wsrep_slave_threads`

Command Line `--wsrep-slave-threads`

Config File Yes

Scope Global

Dynamic Yes

Default Value `1`

Specifies the number of threads that can apply replication transactions in parallel. Galera supports true parallel replication that applies transactions in parallel only when it is safe to do so. This variable is dynamic. You can increase/decrease it at any time.

Note: When you decrease the number of threads, it won't kill the threads immediately, but stop them after they are done applying current transaction (the effect with an increase is immediate though).

If any replication consistency problems are encountered, it's recommended to set this back to `1` to see if that resolves the issue. The default value can be increased for better throughput.

You may want to increase it as suggested in Codership documentation for flow control: when the node is in `JOINED` state, increasing the number of slave threads can speed up the catchup to `SYNCED`.

You can also estimate the optimal value for this from *`wsrep_cert_deps_distance`* as suggested on this page.

For more configuration tips, see this document.

variable `wsrep_slave_UK_checks`

Command Line `--wsrep-slave-UK-checks`

Config File Yes

Scope Global

Dynamic Yes

Default Value `OFF`

Defines whether unique key checking is done for applier threads. This is disabled by default.

variable **`wsrep_sst_auth`**

Command Line `--wsrep-sst-auth`

Config File Yes

Scope Global

Dynamic Yes

Format `<username>:<password>`

Specifies authentication information for State Snapshot Transfer (SST). Required information depends on the method specified in the *`wsrep_sst_method`* variable.

For more information about SST authentication, see *State Snapshot Transfer*.

Note: Value of this variable is masked in the log and in the `SHOW VARIABLES` query output.

variable **`wsrep_sst_donor`**

Command Line `--wsrep-sst-donor`

Config File Yes

Scope Global

Dynamic Yes

Specifies a list of nodes (using their *`wsrep_node_name`* values) that the current node should prefer as donors for *SST* and *IST*.

Warning: Using IP addresses of nodes instead of node names (the value of *`wsrep_node_name`*) as values of *`wsrep_sst_donor`* results in an error.

Error message

[ERROR] WSREP: State transfer request failed unrecoverably: 113 (No route to host). Most likely it is due to inability to communicate with the cluster primary component. Restart required.

If the value is empty, the first node in SYNCED state in the index becomes the donor and will not be able to serve requests during the state transfer.

To consider other nodes if the listed nodes are not available, add a comma at the end of the list, for example:

```
wsrep_sst_donor=node1,node2,
```

If you remove the trailing comma from the previous example, then the joining node will consider *only* `node1` and `node2`.

Note: By default, the joiner node does not wait for more than 100 seconds to receive the first packet from a donor. This is implemented via the *`sst-initial-timeout`* option. If you set the list of preferred donors without the trailing comma or believe that all nodes in the cluster can often be unavailable for SST (this is common for small clusters), then you may want to increase the initial timeout (or disable it completely if you don't mind the joiner node waiting for the state transfer indefinitely).

---

variable `wsrep_sst_donor_rejects_queries`

Command Line `--wsrep-sst-donor-rejects-queries`

Config File Yes

Scope Global

Dynamic Yes

Default Value `OFF`

Defines whether the node should reject blocking client sessions when it is serving as a donor during a blocking state transfer method (when *`wsrep_sst_method`* is set to `mysqldump` or `rsync`). This is disabled by default, meaning that the node accepts such queries.

If you enable this variable, queries will return the `Unknown command` error. This can be used to signal load-balancer that the node isn't available.

variable `wsrep_sst_method`

Command Line `--wsrep-sst-method`

Config File Yes

Scope Global

Dynamic Yes

Default Value `xtrabackup-v2`

Defines the method or script for *State Snapshot Transfer* (SST).

Available values are:

- `xtrabackup-v2`: Uses *Percona XtraBackup* to perform SST. This method requires *`wsrep_sst_auth`* to be set up with credentials (`<user>:<password>`) on the donor node. Privileges and perimssions for running *Percona XtraBackup* can be found in Percona XtraBackup documentation.

  Note: This is the **recommended** and default method for *Percona XtraDB Cluster*. For more information, see *Percona XtraBackup SST Configuration*.

- `rsync`: Uses `rsync` to perform SST. This method doesn't use the *`wsrep_sst_auth`* variable.
- `mysqldump`: Uses `mysqldump` to perform SST This method requires superuser credentials for the donor node to be specified in the *`wsrep_sst_auth`* variable.

  Note: This method is not recommended unless it is required for specific reasons. Also, it is not compatible with `bind_address` set to `127.0.0.1` or `localhost`, and will cause startup to fail in this case.

- `<custom_script_name>`: Galera supports Scriptable State Snapshot Transfer. This enables users to create their own custom scripts for performing SST. For example, you can create a script `/usr/bin/wsrep_MySST.sh` and specify `MySST` for this variable to run your custom SST script.
- `skip`: Use this to skip SST. This can be used when initially starting the cluster and manually restoring the same data to all nodes. It shouldn't be used permanently because it could lead to data inconsistency across the nodes.

Note: Only `xtrabackup-v2` and `rsync` provide support for clusters with GTIDs and async slaves.

variable `wsrep_sst_receive_address`

Command Line `--wsrep-sst-receive-address`

Config File Yes

Scope Global

Dynamic Yes

Default Value `AUTO`

Specifies the network address where donor node should send state transfers. By default, this variable is set to `AUTO`, meaning that the IP address from *`wsrep_node_address`* is used.

variable `wsrep_start_position`

Command Line `--wsrep-start-position`

Config File Yes

Scope Global

Dynamic Yes

Default Value `00000000-0000-0000-0000-00000000000000:-1`

Specifies the node's start position as `UUID:seqno`. By setting all the nodes to have the same value for this variable, the cluster can be set up without the state transfer.

variable `wsrep_sync_wait`

Version Info

- ***`5.6.20-25.7`*** – Variable introduced

Command Line `--wsrep-sync-wait`

Config File Yes

Scope Session

Dynamic Yes

Default Value `0`

Controls cluster-wide causality checks on certain statements. Checks ensure that the statement is executed on a node that is fully synced with the cluster.

Note: Causality checks of any type can result in increased latency.

The type of statements to undergo checks is determined by bitmask:

- `0`: Do not run causality checks for any statements. This is the default.

- `1`: Perform checks for `READ` statements (including `SELECT`, `SHOW`, and `BEGIN` or `START TRANSACTION`).
- `2`: Perform checks for `UPDATE` and `DELETE` statements.
- `3`: Perform checks for `READ`, `UPDATE`, and `DELETE` statements.
- `4`: Perform checks for `INSERT` and `REPLACE` statements.
- `5`: Perform checks for `READ`, `INSERT`, and `REPLACE` statements.
- `6`: Perform checks for `UPDATE`, `DELETE`, `INSERT`, and `REPLACE` statements.
- `7`: Perform checks for `READ`, `UPDATE`, `DELETE`, `INSERT`, and `REPLACE` statements.

Note: Setting *`wsrep_sync_wait`* to `1` is the equivalent of setting the deprecated *`wsrep_causal_reads`* to `ON`.

CHAPTER

TWENTYNINE

INDEX OF `WSREP_PROVIDER` OPTIONS

The following variables can be set and checked in the *wsrep_provider_options* variable. The value of the variable can be changed in the *MySQL* configuration file, `my.cnf`, or by setting the variable value in the *MySQL* client.

To change the value in `my.cnf`, the following syntax should be used:

```
wsrep_provider_options="variable1=value1;[variable2=value2]"
```

For example to set the size of the Galera buffer storage to 512 MB, specify the following in `my.cnf`:

```
wsrep_provider_options="gcache.size=512M"
```

Dynamic variables can be changed from the *MySQL* client using the `SET GLOBAL` command. For example, to change the value of the *pc.ignore_sb*, use the following command:

```
mysql> SET GLOBAL wsrep_provider_options="pc.ignore_sb=true";
```

Index

variable `base_dir`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value value of `datadir`

This variable specifies the data directory.

variable `base_host`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value value of *wsrep_node_address*

This variable sets the value of the node's base IP. This is an IP address on which Galera listens for connections from other nodes. Setting this value incorrectly would stop the node from communicating with other nodes.

variable **`base_port`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 4567

This variable sets the port on which Galera listens for connections from other nodes. Setting this value incorrectly would stop the node from communicating with other nodes.

variable **`cert.log_conflicts`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value no

This variable is used to specify if the details of the certification failures should be logged.

variable **`debug`**

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value no

When this variable is set to `yes`, it will enable debugging.

variable **`evs.auto_evict`**

Version Introduced in *5.6.21-25.8*

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value 0

Number of entries allowed on delayed list until auto eviction takes place. Setting value to 0 disables auto eviction protocol on the node, though node response times will still be monitored. EVS protocol version (*`evs.version`*) `1` is required to enable auto eviction.

variable **`evs.causal_keepalive_period`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value value of *evs.keepalive_period*

This variable is used for development purposes and shouldn't be used by regular users.

variable evs.debug_log_mask

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value 0x1

This variable is used for EVS (Extended Virtual Synchrony) debugging. It can be used only when *wsrep_debug* is set to ON.

variable evs.delay_margin

Version Introduced in *5.6.21-25.8*

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value PT1S

Time period that a node can delay its response from expected until it is added to delayed list. The value must be higher than the highest RTT between nodes.

variable evs.delayed_keep_period

Version Introduced in *5.6.21-25.8*

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value PT30S

Time period that node is required to remain responsive until one entry is removed from delayed list.

variable evs.evict

Version Introduced in *5.6.21-25.8*

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Manual eviction can be triggered by setting the *evs.evict* to a certain node value. Setting the *evs.evict* to an empty string will clear the evict list on the node where it was set.

variable evs.inactive_check_period

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value PT0.5S

This variable defines how often to check for peer inactivity.

variable `evs.inactive_timeout`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value PT15S

This variable defines the inactivity limit, once this limit is reached the node will be considered dead.

variable `evs.info_log_mask`

Command Line No

Config File Yes

Scope Global

Dynamic No

Default Value 0

This variable is used for controlling the extra EVS info logging.

variable `evs.install_timeout`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value PT7.5S

This variable defines the timeout on waiting for install message acknowledgments.

variable `evs.join_retrans_period`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value PT1S

This variable defines how often to retransmit EVS join messages when forming cluster membership.

variable `evs.keepalive_period`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value PT1S

This variable defines how often to emit keepalive beacons (in the absence of any other traffic).

variable **`evs.max_install_timeouts`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 1

This variable defines how many membership install rounds to try before giving up (total rounds will be *`evs.max_install_timeouts`* + 2).

variable **`evs.send_window`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 4

This variable defines the maximum number of data packets in replication at a time. For WAN setups, the variable can be set to a considerably higher value than default (for example,512). The value must not be less than *`evs.user_send_window`*.

variable **`evs.stats_report_period`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value PT1M

This variable defines the control period of EVS statistics reporting.

variable **`evs.suspect_timeout`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value PT5S

This variable defines the inactivity period after which the node is "suspected" to be dead. If all remaining nodes agree on that, the node will be dropped out of cluster even before *`evs.inactive_timeout`* is reached.

variable **`evs.use_aggregate`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value true

When this variable is enabled, smaller packets will be aggregated into one.

variable **`evs.user_send_window`**

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value 2

This variable defines the maximum number of data packets in replication at a time. For WAN setups, the variable can be set to a considerably higher value than default (for example, 512).

variable **`evs.version`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 0

This variable defines the EVS protocol version. Auto eviction is enabled when this variable is set to `1`. Default `0` is set for backwards compatibility.

variable **`evs.view_forget_timeout`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value P1D

This variable defines the timeout after which past views will be dropped from history.

variable **`gcomm.thread_prio`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Set the priority of the `gcomm` thread to a higher level if *Percona XtraDB Cluster* threads receive little CPU time because of other MySQL threads. This should prevent timeouts that might lead to nodes dropping from the cluster.

The format for this option is `<policy>:<priority>`, for example:

```
wsrep_provider_options="gcomm.thread_prio=rr:2"
```

Set the priority to an integer number. Set the policy to one of the following values:

- `other` means the default time-sharing scheduling in Linux. Threads can run until they are blocked by an I/O request or preempted by higher priorities or superior scheduling designations.
- `fifo` means *first-in-first-out* scheduling. Threads always immediately preempt any currently running other, batch or idle threads. They can run until they are either blocked by an I/O request or preempted by a FIFO thread of a higher priority.
- `rr` means *round-robin* scheduling. Threads always preempt any currently running other, batch or idle threads. The scheduler allows these threads to run for a fixed period of a time. If the thread is still running when this time period is exceeded, they are stopped and moved to the end of the list, allowing another round-robin thread of the same priority to run in their place. They can otherwise continue to run until they are blocked by an I/O request or are preempted by threads of a higher priority.

variable `gcache.dir`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value *datadir*

This variable can be used to define the location of the `galera.cache` file.

variable `gcache.keep_pages_count`

Version Implemented in *5.6.25-25.12*

Command Line Yes

Config File Yes

Scope Local, Global

Dynamic Yes

Default Value 0

This variable is used to limit the number of overflow pages rather than the total memory occupied by all overflow pages. Whenever `gcache.keep_pages_count` is set to a non-zero value, excess overflow pages will be deleted (starting from the oldest to the newest).

Whenever either the `gcache.keep_pages_count` or the *`gcache.keep_pages_size`* variable is updated at runtime to a non-zero value, cleanup is called on excess overflow pages to delete them.

variable `gcache.keep_pages_size`

Command Line Yes

Config File Yes

Scope Local, Global

Dynamic No

Default Value 0

This variable is used to limit the total size of overflow pages rather than the count of all overflow pages. Whenever `gcache.keep_pages_size` is set to a non-zero value, excess overflow pages will be deleted (starting from the oldest to the newest) until the total size is below the specified value.

Whenever either the *`gcache.keep_pages_count`* or the `gcache.keep_pages_size` variable is updated at runtime to a non-zero value, cleanup is called on excess overflow pages to delete them.

variable `gcache.mem_size`

Version Deprecated in *5.6.22-25.8*

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 0

This variable was used to define how much RAM is available for the system.

Warning: This variable has been deprecated and shouldn't be used as it could cause a node to crash.

variable `gcache.name`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value /var/lib/mysql/galera.cache

This variable can be used to specify the name of the Galera cache file.

variable `gcache.page_size`

Command Line No

Config File Yes

Scope Global

Dynamic No

Default Value 128M

This variable can be used to specify the size of the page files in the page storage.

variable `gcache.size`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 128M

Size of the transaction cache for Galera replication. This defines the size of the `galera.cache` file which is used as source for *IST*. The bigger the value of this variable, the better are chances that the re-joining node will get IST instead of *SST*.

variable `gcs.fc_debug`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** 0

This variable specifies after how many writesets the debug statistics about SST flow control will be posted.

variable `gcs.fc_factor`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** Yes
>
> **Default Value** 1

This variable is used for replication flow control. Replication is resumed when the slave queue drops below *`gcs.fc_factor`* * *`gcs.fc_limit`*.

variable `gcs.fc_limit`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** Yes
>
> **Default Value** 16

This variable is used for replication flow control. Replication is paused when the slave queue exceeds this limit.

variable `gcs.fc_master_slave`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** NO

This variable is used to specify if there is only one master node in the cluster.

variable `gcs.max_packet_size`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No

Default Value 64500

This variable is used to specify the writeset size after which they will be fragmented.

variable **`gcs.max_throttle`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 0.25

This variable specifies how much the replication can be throttled during the state transfer in order to avoid running out of memory. Value can be set to `0.0` if stopping replication is acceptable in order to finish state transfer.

variable **`gcs.recv_q_hard_limit`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 9223372036854775807

This variable specifies the maximum allowed size of the receive queue. This should normally be `(RAM + swap) / 2`. If this limit is exceeded, Galera will abort the server.

variable **`gcs.recv_q_soft_limit`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 0.25

This variable specifies the fraction of the *`gcs.recv_q_hard_limit`* after which replication rate will be throttled.

variable **`gcs.sync_donor`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value NO

This variable controls if the rest of the cluster should be in sync with the donor node. When this variable is set to `YES`, the whole cluster will be blocked if the donor node is blocked with SST.

variable **`gmcast.listen_addr`**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value tcp://0.0.0.0:4567

This variable defines the address on which the node listens to connections from other nodes in the cluster.

variable `gmcast.mcast_addr`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value None

This variable should be set up if UDP multicast should be used for replication.

variable `gmcast.mcast_ttl`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 1

This variable can be used to define TTL for multicast packets.

variable `gmcast.peer_timeout`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value PT3S

This variable specifies the connection timeout to initiate message relaying.

variable `gmcast.segment`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 0

This variable specifies the group segment this member should be a part of. Same segment members are treated as equally physically close.

variable `gmcast.time_wait`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value PT5S

This variable specifies the time to wait until allowing peer declared outside of stable view to reconnect.

variable `gmcast.version`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 0

This variable shows which gmcast protocol version is being used.

variable `ist.recv_addr`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value value of *`wsrep_node_address`*

This variable specifies the address on which the node listens for Incremental State Transfer (*IST*).

variable `ist.recv_bind`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value value of *`ist.recv_addr`*

Defines the address that the node binds on for receiving an Incremental State Transfer (IST), for example:

```
wsrep_provider_options="ist.recv_bind=192.168.1.1"
```

This option may be useful if a node runs behind a NAT or in similar cases where the public and private addresses differ.

variable `pc.announce_timeout`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value PT3S

Cluster joining announcements are sent every 1/2 second for this period of time or less if other nodes are discovered.

variable `pc.checksum`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value true

This variable controls whether replicated messages should be checksummed or not.

variable `pc.ignore_quorum`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value false

When this variable is set to `TRUE`, the node will completely ignore quorum calculations. This should be used with extreme caution even in master-slave setups, because slaves won't automatically reconnect to master in this case.

variable `pc.ignore_sb`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value false

When this variable is set to `TRUE`, the node will process updates even in the case of a split brain. This should be used with extreme caution in multi-master setup, but should simplify things in master-slave cluster (especially if only 2 nodes are used).

variable `pc.linger`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value PT20S

This variable specifies the period for which the PC protocol waits for EVS termination.

variable `pc.npvo`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value false

When this variable is set to `TRUE`, more recent primary components override older ones in case of conflicting primaries.

variable `pc.recovery`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** true

When this variable is set to `true`, the node stores the Primary Component state to disk. The Primary Component can then recover automatically when all nodes that were part of the last saved state re-establish communication with each other. This feature allows automatic recovery from full cluster crashes, such as in the case of a data center power outage. A subsequent graceful full cluster restart will require explicit bootstrapping for a new Primary Component.

Note: The cluster must have already been initially bootstrapped (see _bootstrap for details). A `gwvstate.dat` file described in _wsrep_file_index must exist as well.

---

variable `pc.version`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** 0

This status variable is used to check which PC protocol version is used.

variable `pc.wait_prim`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** true

When set to `TRUE`, the node waits for a primary component for the period of time specified in *`pc.wait_prim_timeout`*. This is useful to bring up a non-primary component and make it primary with `pc.bootstrap`.

variable `pc.wait_prim_timeout`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** PT30S

This variable is used to specify the period of time to wait for a primary component.

variable `pc.weight`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** Yes
>
> **Default Value** 1

This variable specifies the node weight that's going to be used for Weighted Quorum calculations.

variable `protonet.backend`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** asio

This variable is used to define which transport backend should be used. Currently only `ASIO` is supported.

variable `protonet.version`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** 0

This status variable is used to check which transport backend protocol version is used.

variable `repl.causal_read_timeout`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** Yes
>
> **Default Value** PT30S

This variable specifies the causal read timeout.

variable `repl.commit_order`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** 3

This variable is used to specify out-of-order committing (which is used to improve parallel applying performance). The following values are available:

- `0` - BYPASS: all commit order monitoring is turned off (useful for measuring performance penalty)
- `1` - OOOC: allow out-of-order committing for all transactions
- `2` - LOCAL_OOOC: allow out-of-order committing only for local transactions
- `3` - NO_OOOC: no out-of-order committing is allowed (strict total order committing)

variable `repl.key_format`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value FLAT8

This variable is used to specify the replication key format. The following values are available:

- `FLAT8` - short key with higher probability of key match false positives
- `FLAT16` - longer key with lower probability of false positives
- `FLAT8A` - same as `FLAT8` but with annotations for debug purposes
- `FLAT16A` - same as `FLAT16` but with annotations for debug purposes

variable `repl.max_ws_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 2147483647

This variable is used to specify the maximum size of a write-set in bytes. This is limited to 2 gygabytes.

variable `repl.proto_max`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 7

This variable is used to specify the highest communication protocol version to accept in the cluster. Used only for debugging.

variable `socket.checksum`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 2

This variable is used to choose the checksum algorithm for network packets. The following values are available:

- `0` - disable checksum
- `1` - plain `CRC32` (used in Galera 2.x)
- `2` - hardware accelerated `CRC32-C`

variable `socket.ssl`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** No

This variable is used to specify if SSL encryption should be used.

variable `socket.ssl_cert`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No

This variable is used to specify the path (absolute or relative to working directory) to an SSL certificate (in PEM format).

variable `socket.ssl_key`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No

This variable is used to specify the path (absolute or relative to working directory) to an SSL private key for the certificate (in PEM format).

variable `socket.ssl_compression`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No
>
> **Default Value** yes

This variable is used to specify if the SSL compression is to be used.

variable `socket.ssl_cipher`

> **Command Line** Yes
>
> **Config File** Yes
>
> **Scope** Global
>
> **Dynamic** No

Default Value AES128-SHA

This variable is used to specify what cypher will be used for encryption.

CHAPTER

THIRTY

INDEX OF FILES CREATED BY PXC

- **GRA_*.log** These files contain binlog events in ROW format representing the failed transaction. That means that the slave thread was not able to apply one of the transactions. For each of those file, a corresponding warning or error message is present in the mysql error log file. Those error can also be false positives like a bad `DDL` statement (dropping a table that doesn't exists for example) and therefore nothing to worry about. However it's always recommended to check these log to understand what's is happening.

  To be able to analyze these files binlog header needs to be added to the log file. To create the `GRA_HEADER` file you need an instance running with `binlog_checksum` set to `NONE` and extract first 120 bytes from the binlog file:

```
$ head -c 120 mysqld-bin.000001 > GRA_HEADER
$ cat GRA_HEADER > /var/lib/mysql/GRA_1_2-bin.log
$ cat /var/lib/mysql/GRA_1_2.log >> /var/lib/mysql/GRA_1_2-bin.log
$ mysqlbinlog -vvv /var/lib/mysql/GRA_1_2-bin.log

/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#160805  9:35:35 server id 1  end_log_pos 120     Start: binlog v 4, server v␣
↪5.6.30-76.3-56-log created 160805  9:35:35 at startup
# Warning: this binlog is either in use or was not closed properly.
ROLLBACK/*!*/;
BINLOG '
512kVw8BAAAAdAAAAHgAAAABAAQANS42LjMwLTc2LjMtNTYtbG9nAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAADnXaRXEzgNAAgAEgAEBAQEEgAAXAAEGggAAAAICAgCAAAACgoKGRkAAOfS
sRE=
'/*!*/;
# at 120
#160805  9:33:37 server id 0  end_log_pos 79  Query   thread_id=27    exec_
↪time=0 error_code=0
use `world`/*!*/;
SET TIMESTAMP=1470389617/*!*/;
SET @@session.pseudo_thread_id=27/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=0, @@session.
↪unique_checks=1, @@session.autocommit=1/*!*/;
SET @@session.sql_mode=1073741824/*!*/;
SET @@session.auto_increment_increment=1, @@session.auto_increment_offset=1/*!
↪*/;
/*!\C utf8 *//*!*/;
SET @@session.character_set_client=33,@@session.collation_connection=33,
↪@@session.collation_server=8/*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
drop table test
/*!*/;
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
```

This information can be used for checking the *MySQL* error log for the corresponding error message.

```
160805  9:33:37 8:52:21 [ERROR] Slave SQL: Error 'Unknown table 'test'' on␣
↪query. Default database: 'test'. Query: 'drop table test', Error_code: 1051
160805  9:33:37 8:52:21 [Warning] WSREP: RBR event 1 Query apply warning: 1, 3
```

In this example `DROP TABLE` statement was executed on a table that doesn't exist.

- **`galera.cache`** This file is used as a main writeset store. It's implemented as a permanent ring-buffer file that is preallocated on disk when the node is initialized. File size can be controlled with the variable *`gcache.size`*. If this value is bigger, more writesets are cached and chances are better that the re-joining node will get *IST* instead of *SST*. Filename can be changed with the *`gcache.name`* variable.
- **`grastate.dat`** This file contains the Galera state information.
  - `version` - grastate version
  - `uuid` - a unique identifier for the state and the sequence of changes it undergoes.For more information on how UUID is generated see *UUID*.
  - `seqno` - Ordinal Sequence Number, a 64-bit signed integer used to denote the position of the change in the sequence. `seqno` is `0` when no writesets have been generated or applied on that node, i.e., not applied/generated across the lifetime of a `grastate` file. `-1` is a special value for the `seqno` that is kept in the `grastate.dat` while the server is running to allow Galera to distinguish between a clean and an unclean shutdown. Upon a clean shutdown, the correct `seqno` value is written to the file. So, when the server is brought back up, if the value is still `-1` , this means that the server did not shut down cleanly. If the value is greater than `0`, this means that the shutdown was clean. `-1` is then written again to the file in order to allow the server to correctly detect if the next shutdown was clean in the same manner.
  - `cert_index` - cert index restore through grastate is not implemented yet

  Examples of this file look like this:

  In case server node has this state when not running it means that that node crashed during the transaction processing.

```
# GALERA saved state
version: 2.1
uuid:    1917033b-7081-11e2-0800-707f5d3b106b
seqno:   -1
cert_index:
```

  In case server node has this state when not running it means that the node was gracefully shut down.

```
# GALERA saved state
version: 2.1
uuid:    1917033b-7081-11e2-0800-707f5d3b106b
seqno:   5192193423942
cert_index:
```

In case server node has this state when not running it means that the node crashed during the DDL.

```
# GALERA saved state
version: 2.1
uuid:    00000000-0000-0000-0000-000000000000
seqno:   -1
cert_index:
```

- `gvwstate.dat` This file is used for Primary Component recovery feature. This file is created once primary component is formed or changed, so you can get the latest primary component this node was in. And this file is deleted when the node is shutdown gracefully.

  First part contains the node *UUID* information. Second part contains the view information. View information is written between `#vwbeg` and `#vwend`. View information consists of:

- view_id: [view_type] [view_uuid] [view_seq]. - `view_type` is always `3` which means primary view. `view_uuid` and `view_seq` identifies a unique view, which could be perceived as identifier of this primary component.
- bootstrap: [bootstarp_or_not]. - It could be `0` or `1`, but it does not affect primary component recovery process now.
- member: [node's uuid] [node's segment]. - it represents all nodes in this primary component.

  Example of this file looks like this:

```
my_uuid: c5d5d990-30ee-11e4-aab1-46d0ed84b408
#vwbeg
view_id: 3 bc85bd53-31ac-11e4-9895-1f2ce13f2542 2
bootstrap: 0
member: bc85bd53-31ac-11e4-9895-1f2ce13f2542 0
member: c5d5d990-30ee-11e4-aab1-46d0ed84b408 0
#vwend
```

CHAPTER THIRTYONE

FREQUENTLY ASKED QUESTIONS

How do you solve locking issues like auto-increment?

For auto-increment, Percona XtraDB Cluster changes `auto_increment_offset` for each new node. In a single-node workload, locking is handled in the same way as *InnoDB*. In case of write load on several nodes, Percona XtraDB Cluster uses optimistic locking and the application may receive lock error in the response to the `COMMIT` query.

What if one of the nodes crashes and InnoDB recovery rolls back some transactions?

When a node crashes, after restart it will copy the whole dataset from another node (if there were changes to data since the crash).

How can I check the Galera node health?

To check the health of a Galera node, use the following query:

```
SELECT 1 FROM dual;
```

The following results of the previous query are possible:

- You get the row with `id=1` (node is healthy)
- Unknown error (node is online but Galera is not connected/synced with the cluster)
- Connection error (node is not online)

You can also check a node's health with the `clustercheck` script. First set up the `clustercheck` user:

```
GRANT USAGE ON *.* TO 'clustercheck'@'localhost' IDENTIFIED BY PASSWORD
↪'*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19';
```

You can then check a node's health by running the `clustercheck` script:

```
/usr/bin/clustercheck clustercheck password 0
```

If the node is running correctly you should get the following status:

```
HTTP/1.1 200 OK
Content-Type: text/plain
Connection: close
Content-Length: 40

Percona XtraDB Cluster Node is synced.
```

In case node isn't sync or if it is offline, status will look like:

```
HTTP/1.1 503 Service Unavailable
Content-Type: text/plain
Connection: close
Content-Length: 44

Percona XtraDB Cluster Node is not synced.
```

Note: The `clustercheck` script has the following syntax:

`<user> <pass> <available_when_donor=0|1> <log_file> <available_when_readonly=0|1> <defaults_extra_file>`

Recommended: `server_args = user pass 1 /var/log/log-file 0 /etc/my.cnf.local`

Compatibility: `server_args = user pass 1 /var/log/log-file 1 /etc/my.cnf.local`

How does Percona XtraDB Cluster handle big transactions?

Percona XtraDB Cluster populates write set in memory before replication, and this sets the limit for the size of transactions that make sense. There are wsrep variables for maximum row count and maximum size of write set to make sure that the server does not run out of memory.

Is it possible to have different table structures on the nodes?

For example, if there are four nodes, with four tables: `sessions_a`, `sessions_b`, `sessions_c` and `sessions_d`. If you want each table in a separate node, this is not possible for InnoDB tables. However, it will work for MEMORY tables.

What if a node fails or there is a network issue between nodes?

The quorum mechanism in Percona XtraDB Cluster will decide what nodes can accept traffic and will shut down the nodes that do not belong to the quorum. Later when the failure is fixed, the nodes will need to copy data from the working cluster.

The algorithm for quorum is Dynamic Linear Voting (DLV). The quorum is preserved if (and only if) the sum weight of the nodes in a new component strictly exceeds half that of the preceding Primary Component, minus the nodes which left gracefully.

The mechanism is described in detail in Galera documentation.

How would the quorum mechanism handle split brain?

It would not. If there is no way to decide on the primary component, Percona XtraDB Cluster has no way to resolve a *split brain*. The minimal recommendation is to have 3 nodes. However, it is possibile to allow a node to handle the traffic with the following option:

```
wsrep_provider_options="pc.ignore_sb = yes"
```

I have a two nodes setup. When node1 fails, node2 does not accept commands, why?

This is expected behavior, to prevent *split brain*. For more information, see previous question or Galera documentation.

Is it possible to set up a cluster without state transfer?

It is possible in two ways:

1. By default, Galera reads starting position from a text file `<datadir>/grastate.dat`. Simply make this file identical on all nodes, and there will be no state transfer upon start.
2. Use the *`wsrep_start_position`* variable to start the nodes with the same `UUID:seqno` value.

What TCP ports are used by Percona XtraDB Cluster?

You may need to open up to four ports if you are using a firewall:

1. Regular MySQL port (default is 3306).
2. Port for group communication (default is 4567). It can be changed using the following option:

```
wsrep_provider_options ="gmcast.listen_addr=tcp://0.0.0.0:4010; "
```

3. Port for State Snaphot Transfer (default is 4444). It can be changed using the following option:

```
wsrep_sst_receive_address=10.11.12.205:5555
```

4. Port for Incremental State Transfer (default is port for group communication + 1 or 4568). It can be changed using the following option:

```
wsrep_provider_options = "ist.recv_addr=10.11.12.206:7777; "
```

Is there “async” mode or only “sync” commits are supported?

Percona XtraDB Cluster does not support “async” mode, all commits are synchronous on all nodes. To be precise, the commits are “virtually” synchronous, which means that the transaction should pass “certification” on nodes, not physical commit. “Certification” means a guarantee that the transaction does not have conflicts with other transactions on the corresponding node.

Does it work with regular MySQL replication?

Yes. On the node you are going to use as master, you should enable `log-bin` and `log-slave-update` options.

Why doesn’t the init script (/etc/init.d/mysql) start?

Try to disable SELinux with the following command:

```
$ echo 0 > /selinux/enforce
```

What does “nc: invalid option – ‘d’” in the sst.err log file mean?

This is Debian/Ubuntu specific error, Percona XtraDB Cluster uses `netcat-openbsd` package. This dependency has been fixed in recent releases. Future releases of PXC will be compatible with any `netcat` (see bug #959970).

CHAPTER

THIRTYTWO

GLOSSARY

LSN Each InnoDB page (usually 16kb in size) contains a log sequence number, or LSN. The LSN is the system version number for the entire database. Each page's LSN shows how recently it was changed.

InnoDB Storage engine which provides ACID-compliant transactions and foreign key support, among others improvements over *MyISAM*. It is the default engine for *MySQL* as of the 5.5 series.

MyISAM Previous default storage engine for *MySQL* for versions prior to 5.5. It doesn't fully support transactions but in some scenarios may be faster than *InnoDB*. Each table is stored on disk in 3 files: *.frm*, `.MYD`, `.MYI`.

GTID Global Transaction ID, in *Percona XtraDB Cluster* it consists of *UUID* and an ordinal sequence number which denotes the position of the change in the sequence.

> **Warning:** A node running Percona Server may lose the GTID (Global Transaction ID) information if the variable `max-binlog-files` is set to **1** either before the node is restarted or after *SST*.
>
> **See also:**
>
> **Percona Server Documentation: Restricting the number of binlog files** https://www.percona.com/doc/percona-server/5.6/flexibility/max_binlog_files.html

HAProxy HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for web sites crawling under very high loads while needing persistence or Layer7 processing. Supporting tens of thousands of connections is clearly realistic with todays hardware. Its mode of operation makes its integration into existing architectures very easy and riskless, while still offering the possibility not to expose fragile web servers to the net.

IST Incremental State Transfer. Functionality which instead of whole state snapshot can catch up with te group by receiving the missing writesets, but only if the writeset is still in the donor's writeset cache.

SST State Snapshot Transfer is the full copy of data from one node to another. It's used when a new node joins the cluster, it has to transfer data from existing node. There are three methods of SST available in Percona XtraDB Cluster: **`mysqldump`**, **`rsync`** and **`xtrabackup`**. The downside of *mysqldump* and *rsync* is that the node becomes *READ-ONLY* while data is being copied from one node to another (SST applies **`FLUSH TABLES WITH READ LOCK`** command). Xtrabackup SST does not require **`READ LOCK`** for the entire syncing process, only for syncing the *MySQL* system tables and writing the information about the binlog, galera and slave information (same as the regular XtraBackup backup). State snapshot transfer method can be configured with the *`wsrep_sst_method`* variable.

UUID Universally Unique IDentifier which uniquely identifies the state and the sequence of changes node undergoes. 128-bit UUID is a classic DCE UUID Version 1 (based on current time and MAC address). Although in theory this UUID could be generated based on the real MAC-address, in the Galera it is always (without exception) based on the generated pseudo-random addresses ("locally administered" bit in the node address (in the UUID structure) is always equal to unity).

Complete structure of the 128-bit UUID field and explanation for its generation are as follows:

From	To	Length	Content
0	31	32	Bits 0-31 of Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582, encoded as big-endian 32-bit number.
32	47	16	Bits 32-47 of UTC as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582, encoded as big-endian 16-bit number.
48	59	12	Bits 48-59 of UTC as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582, encoded as big-endian 16-bit number.
60	63	4	UUID version number: always equal to 1 (DCE UUID).
64	69	6	most-significants bits of random number, which generated from the server process PID and Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582.
70	71	2	UID variant: always equal to binary 10 (DCE variant).
72	79	8	8 least-significant bits of random number, which generated from the server process PID and Coordinated Universal Time (UTC) as a count of 100-nanosecond intervals since 00:00:00.00, 15 October 1582.
80	80	1	Random bit ("unique node identifier").
81	81	1	Always equal to the one ("locally administered MAC address").
82	127	46	Random bits ("unique node identifier"): readed from the `/dev/urandom` or (if `/dev/urandom` is unavailable) generated based on the server process PID, current time and bits of the default "zero node identifier" (entropy data).

XtraBackup *Percona XtraBackup* is an open-source hot backup utility for *MySQL* - based servers that doesn't lock your database during the backup.

XtraDB *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. More information here.

XtraDB Cluster *Percona XtraDB Cluster* is a high availability solution for MySQL.

Percona XtraDB Cluster *Percona XtraDB Cluster* (PXC) is a high availability solution for MySQL.

my.cnf This file refers to the database server's main configuration file. Most Linux distributions place it as `/etc/mysql/my.cnf` or `/etc/my.cnf`, but the location and name depends on the particular installation. Note that this is not the only way of configuring the server, some systems does not have one even and rely on the command options to start the server and its defaults values.

cluster replication Normal replication path for cluster members. Can be encrypted (not by default) and unicast or multicast (unicast by default). Runs on tcp port 4567 by default.

datadir The directory in which the database server stores its databases. Most Linux distribution use `/var/lib/mysql` by default.

donor node The node elected to provide a state transfer (SST or IST).

ibdata Default prefix for tablespace files, e.g. `ibdata1` is a 10MB autoextendable file that *MySQL* creates for the shared tablespace by default.

joiner node The node joining the cluster, usually a state transfer target.

node A cluster node – a single mysql instance that is in the cluster.

primary cluster A cluster with *quorum*. A non-primary cluster will not allow any operations and will give `Unknown command` errors on any clients attempting to read or write from the database.

quorum A majority (> 50%) of nodes. In the event of a network partition, only the cluster partition that retains a quorum (if any) will remain Primary by default.

split brain Split brain occurs when two parts of a computer cluster are disconnected, each part believing that the other is no longer running. This problem can lead to data inconsistency.

.frm For each table, the server will create a file with the :file‘.frm‘ extension containing the table definition (for all storage engines).

- genindex
- search

INDEX