PERCONA
www.percona.com

Percona XtraDB Cluster Documentation

Release 5.6.28-25.14

January 27, 2016

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* in the following meaning:

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

CHAPTER

ONE

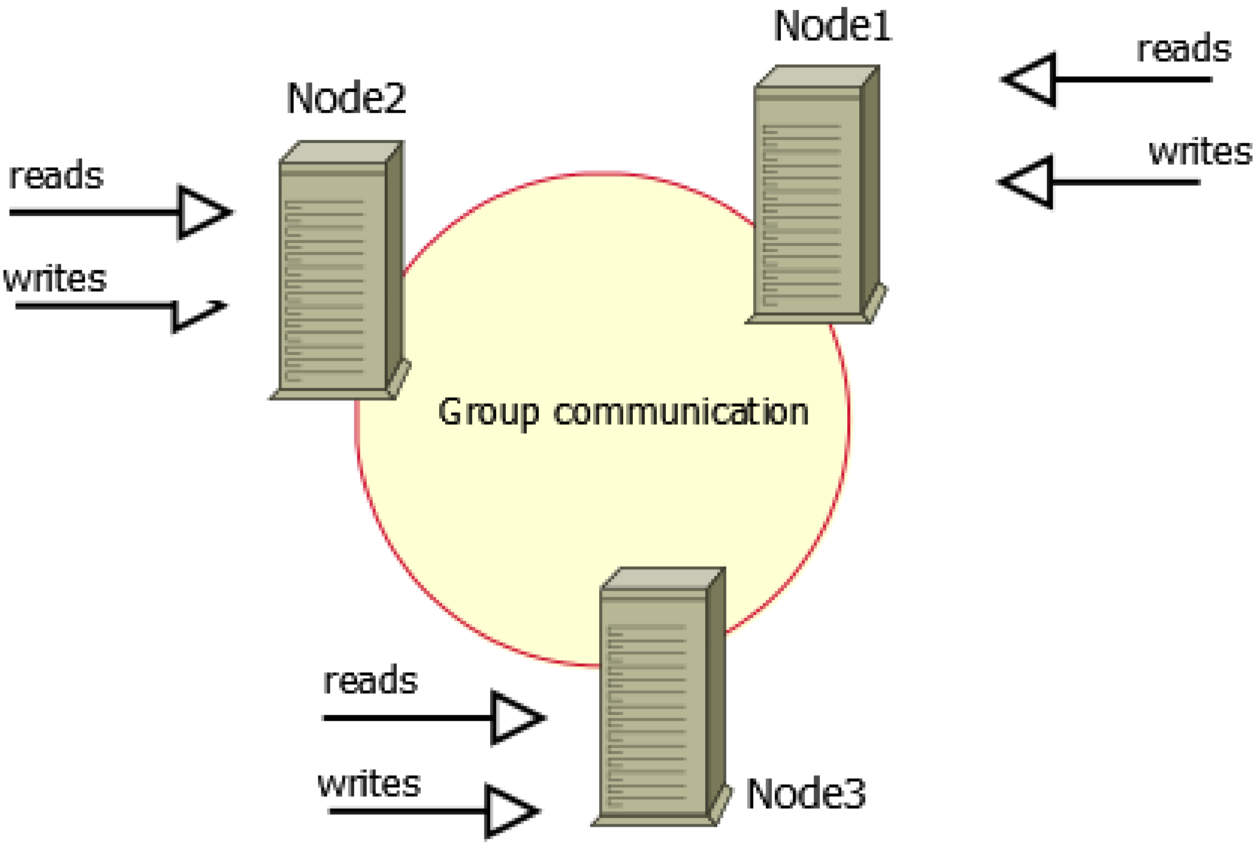
INTRODUCTION

1.1 About Percona XtraDB Cluster

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

1.1.1 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* 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.

1.1.2 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.

1.2 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.

1.3 Percona XtraDB Cluster Errata

1.3.1 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 from Binaries* 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).

1.3.2 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.

CHAPTER

TWO

INSTALLATION

2.1 Installing Percona XtraDB Cluster from Binaries

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

- `RPM` packages for *RHEL* 5, *RHEL* 6, and *RHEL* 7
- *Debian* packages
- Generic `.tar.gz` packages

2.1.1 Using Percona Software Repositories

Percona yum Repository

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

The easiest way to install the *Percona Yum* repository is to install an *RPM* that configures **yum** and installs the Percona GPG key. Installation can also be done manually.

Automatic Installation

Execute the following command as a `root` user:

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

You should see some output such as the following:

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

Install XtraDB Cluster

Make sure to remove existing PXC-5.5 and PS-5.5/5.6 packages before proceeding.

Following command will install Cluster packages:

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

Instead of Percona-XtraDB-Cluster-56 you can install Percona-XtraDB-Cluster-full-56 meta-package which will install Percona-XtraDB-Cluster-devel-56, Percona-XtraDB-Cluster-test-56, Percona-XtraDB-Cluster-debuginfo-56, Percona-XtraDB-Cluster-galera-3-debuginfo, and Percona-XtraDB-Cluster-shared-56 packages in addition.

> **Warning:** In order to sucessfully install *Percona XtraDB Cluster* socat package will need to be installed first. socat package can be installed from the EPEL repositories.

Percona *yum* Testing repository

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

Resolving package conflicts

In CentOS mysql-libs conflicts with Percona-XtraDB-Cluster-server-56.x86_64 package. To avoid this you need to remove the mysql-libs package before installing *Percona XtraDB Cluster*. Package Percona-Server-shared-51.x86_64 provides that dependency during installation if required.

Percona apt Repository

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

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

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

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

Remember to update the local cache:

```
$ apt-get update
```

Supported Architectures

- x86_64 (also known as amd64)
- x86

Supported Releases

Debian

- 6.0 squeeze
- 7.0 wheezy
- 8.0 jessie

Ubuntu

- 12.04LTS Precise Pangolin
- 14.04LTS Trusty Tahr
- 15.04 Vivid Vervet
- 15.10 Wily Werewolf

Install Percona XtraDB Cluster

Make sure to remove existing *Percona XtraDB Cluster* 5.5 and *Percona Server* 5.5/5.6 packages before proceeding.

Following command will install *Percona XtraDB Cluster* packages:

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

Instead of `percona-xtradb-cluster-56` you can install `percona-xtradb-cluster-full-56` meta-package which will install `percona-xtradb-cluster-test-5.6`, `percona-xtradb-cluster-5.6-dbg`, `percona-xtradb-cluster-garbd-3.x`, `percona-xtradb-cluster-galera-3.x-dbg`, `percona-xtradb-cluster-garbd-3.x-dbg` and `libmysqlclient18` packages in addition.

Note: Garbd is packaged separately as part of debian split packaging. The garbd debian package 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-56-full` meta package.

Note: On *Debian Jessie (8.0)* and *Ubuntu Vivid (15.04)* in order to stop or reload the node bootstrapped with `service mysql bootstrap-pxc` command, you'll need to use `service mysql stop-bootstrap` or `service mysql reload-bootstrap` respectively. Also, status of the bootstrapped node should be checked with `service mysql status-bootstrap`.

Percona *apt* Testing repository

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

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

Apt-Pinning the packages

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

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

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

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

This is the recommend way of installing where possible. Make sure to remove existing PXC-5.5 and PS-5.5/5.6 packages before proceeding.

`YUM`-Based Systems

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

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

More detailed example of the *Percona XtraDB Cluster* installation and configuration can be seen in *Installing Percona XtraDB Cluster on CentOS* tutorial.

`DEB`-Based Systems

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

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

More detailed example of the *Percona XtraDB Cluster* installation and configuration can be seen in *Installing Percona XtraDB Cluster on Ubuntu* tutorial.

2.1.2 Prerequisites

In order for *Percona XtraDB Cluster* to work correctly firewall has to be set up to allow connections on the following ports: 3306, 4444, 4567 and 4568. *Percona XtraDB Cluster* currently doesn't work with `SELinux` or `apparmor` so they should be disabled, otherwise individual nodes won't be able to communicate and form the cluster.

2.1.3 Initial configuration

In order to start using the *Percona XtraDB Cluster*, following options are needed in the *MySQL* configuration file `my.cnf`:

```
[mysqld]

wsrep_provider -- a path to Galera library.
wsrep_cluster_address -- Cluster connection URL containing the IPs of other nodes in the cluster
wsrep_sst_method - method used for the state snapshot transfer

binlog_format=ROW - In order for Galera to work correctly binlog format should be ROW
default_storage_engine=InnoDB - MyISAM storage engine has only experimental support
innodb_autoinc_lock_mode=2 - This changes how InnoDB autoincrement locks are managed
```

Additional parameters to specify:

```
wsrep_sst_auth=user:password
```

If any other *State Snapshot Transfer* method beside the **rsync** is specified in the `wsrep_sst_method`, credentials for *SST* need to be specified.

Example:

```
wsrep_provider=/usr/lib64/libgalera_smm.so
wsrep_cluster_address=gcomm://10.11.12.206
wsrep_slave_threads=8
wsrep_sst_method=rsync
binlog_format=ROW
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
```

Detailed list of variables can be found in *Index of wsrep system variables* and *Index of wsrep status variables*.

2.2 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
```

2.3 Percona XtraDB Cluster In-Place Upgrading Guide: From 5.5 to 5.6

Warning:

- Some variables (possibly deprecated) in PS 5.5 may have been removed in PS 5.6 (hence in PXC 5.6), please check that the variable is still valid before upgrade.
- Also, make sure to avoid SST during upgrade since a SST between nodes with 5.5 and 5.6 may not work as expected (especially, if 5.5 is donor and 5.6 is joiner, mysql_upgrade will be required on joiner; vice-versa, package upgrade will be required on joiner). You can also avoid automatic SST for the duration of upgrade by setting `` `wsrep-sst-method` `` to 'skip' for the duration of upgrade. (Note that `` `wsrep-sst-method` `` is a dynamic variable.) Having a large enough gcache helps here. Also, setting it to `skip` ensures that you can handle SST manually if and when required.

Upgrading cluster involves 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.

Note: 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.

2.3.1 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 from Binaries* 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* 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.lis
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..
# 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
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
```

2.3.2 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-c
```

Step #3 Fix the variables in the *MySQL* configuration file `my.cnf` which are not compatible with *Percona Server* 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.li
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 from Binaries* 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..
# 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
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
```

CHAPTER

THREE

FEATURES

3.1 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.

3.2 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:

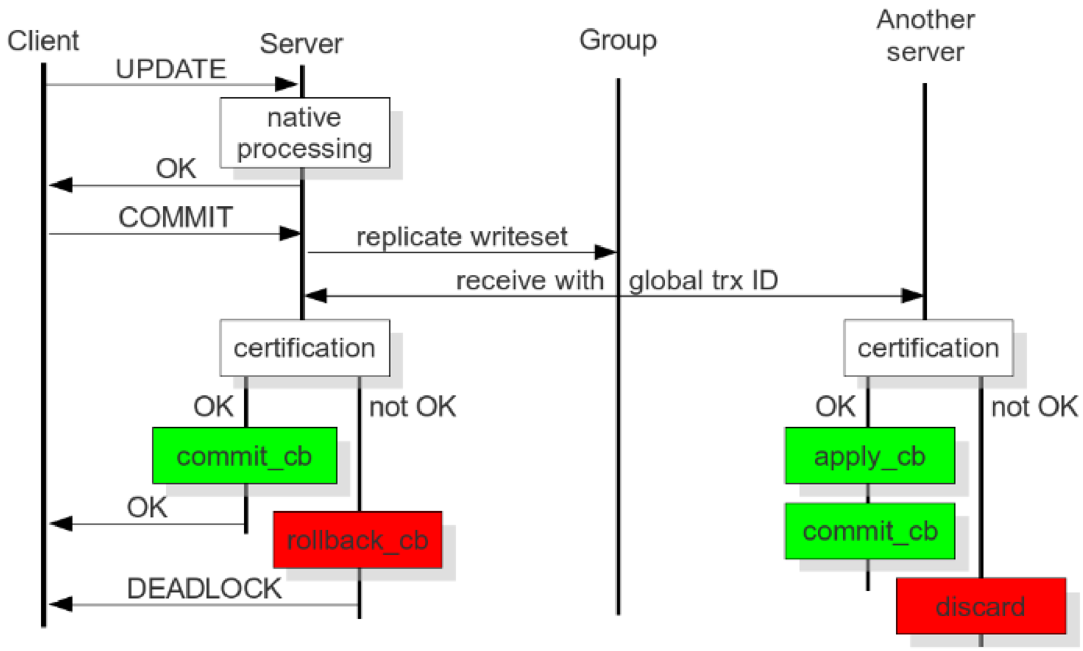


Figure 3.1: *Image source:* Galera documentation - HOW CERTIFICATION-BASED REPLICATION WORKS

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.

CHAPTER

FOUR

USER'S MANUAL

4.1 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.

4.1.1 Other Reading

- How to start a Percona XtraDB Cluster

4.2 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.

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.

4.2.1 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.

4.2.2 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 *Xtrabackup SST Configuration* documentation.

4.2.3 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.

4.2.4 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.

4.2.5 Other Reading

- SST Methods for MySQL
- *Xtrabackup SST configuration*

4.3 Xtrabackup SST Configuration

XtraBackup SST works in two stages:

- Stage I on joiner checks if it is *SST* or *IST* based on presence of `xtrabackup_ist` file.
- In Stage II it starts the data transfer, if it's *SST*, it empties the data directory sans few files (galera.cache, sst_in_progress, grastate.dat) and then proceed with the SST or if it's IST, proceeds as before.

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

Latest *Percona XtraBackup* 2.1.x is strongly recommended for Xtrabackup SST. Refer to Incompatibilities for possible caveats.

4.3.1 Following SST specific options are allowed in my.cnf under [sst]

Note: In following options:

- Non-integer options which have no default are disabled if not set.
- ":Match: Yes" implies that options should match on donor and joiner (in their cnf).
- ":Recommended: Yes" implies the value which is recommended.
- In following options, path always means full path.

- Also, these variables are cnf only (thus not mysqld runtime variables), and can be modified in config till the SST script runs on either donor or joiner (ie. config is read by SST script during SST time and not during mysqld startup).
- Only the main mysqld my.cnf is used. (so providing in ~/.my.cnf etc. won't work).
- All the following options are to be provided under [sst] in cnf unless indicated otherwise.

---

option **`streamfmt`**

Values xbstream, tar

Default xbstream

Match Yes

Xbstream is highly recommended. Refer to *Xbstream v/s Tar* for details and caveats of using tar v/s xbstream for SST.

option **`transferfmt`**

Values socat, nc

Default socat

Match Yes

socat is recommended because it allows for socket options like transfer buffer sizes. Refer to socat(1) for details.

option **`tca`**

Description CA file for openssl based encryption with socat.

Type Full path to CRT file (.crt).

option **`tcert`**

Description PEM for openssl based encryption with socat.

Type Full path to PEM (.pem).

Note: For tca and tcert, refer to http://www.dest-unreach.org/socat/doc/socat-openssltunnel.html for an example. 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 so as to be compatible with socat's parameter' names as well as with MySQL's SSL authentication. For testing you can also download certificates from launchpad. **Note** that irrespective of what is shown in the example, you can use 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 **`encrypt`**

Values 0,1,2,3

Default 0

Match Yes

Decides whether encryption is to be done or not, if this is zero, no encryption is done. `encrypt=2` is recommended if your nodes are over WAN and security constraints are higher, while `encrypt=1` (Xtrabackup-based symmetric encryption) is easier to setup.

- Xtrabackup based encryption with `encrypt=1`.
- OpenSSL based encryption with `encrypt=2`. Socat must be built with openSSL for encryption: `socat -V | grep OPENSSL`.

- Support for SSL encryption for just the key and crt files as implemented in Galera can be enabled with `encrypt=3` option. Information on this option can be found *here*.

Refer to this *document* when enabling with `encrypt=1`.

option `encrypt-algo`

This option is only considered when `encrypt` is equal to 1. Refer to *this* before setting this. This option takes the same value as encrypt option here.

option `sockopt`

Comma separated key/value pairs of socket options. Must begin with a comma. You can use tcpwrap option here to blacklist/whitelist the clients. Refer to socat manual for further details.

Note: You can also enable SSL based compression with `sockopt`. This can be used in place of compress option of Xtrabackup.

option `progress`

Values 1,path/to/file

If equal to:

- 1 it writes to mysql stderr
- path/to/file writes to that 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 only on joiner. 1 implies rebuild indexes. Note that this is independent of compaction, though compaction enables it. Rebuild of indexes may be used as an optimization. Note that #1192834 affects this, hence use of `compact` and `rebuild` are recommended after that is fixed in Percona Xtrabackup and released.

option `time`

Values 0,1

Default 0

Enabling it instruments key stages of backup/restore in SST.

option `rlimit`

Values x(k|m|g|t)

Ratelimit to `x` kilobytes, megabytes etc. Refer to pv(1) for details. Note this rate-limiting happens on donor. The rationale behind this is to not allow SST to saturate the donor's regular cluster operations and/or to ratelimit for other purposes.

option `incremental`

Values 0,1

Default 0

To be set on joiner only, supersedes IST if set. Currently requires manual setup. Hence, not supported currently.

option `use_extra`

Values 0,1

Default 0

If set to 1, SST will use the thread pool's extra_port. Make sure that thread pool is enabled and extra_port option is set in my.cnf before you turn on this option.

option **`cpat`**

During the SST, the *datadir* is cleaned up so that state of other node can be restored cleanly. This option provides the ability to define the files that need to be deleted before the SST. It can be set like:

```
[sst]
cpat='.*galera\.cache$\|.*sst_in_progress$\|.*grastate\.dat$\|.*\.err$\|.*\.log$\|.*RPM_UPGRADE_MARKE
```

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

option **`sst_special_dirs`**

Values 0,1

Default 1

In order for XtraBackup SST to support `innodb_data_home_dir` and `innodb_log_home_dir` variables in the configuration file this option was introduced in *Percona XtraDB Cluster* `5.6.15-25.2`. This requires sst-special-dirs to be set under [sst] in the configuration file to either 0 or 1. Also, `innodb-data-home-dir` and/or `innodb-log-group-home-dir` need to be defined in `my.cnf` under [mysqld]. *Percona XtraBackup* 2.1.6 or higher is required in order for this to work.

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

option **`compressor/decompressor`**

Values command-lines to compressor/decompressor

Default Not set, hence not enabled.

Example compressor='gzip', decompressor='gzip -dc'

This option introduces stream-based compression/decompression. When these options are set, compression/decompression are done on stream, in contrast to earlier PXB-based one where decompression was done after streaming to disk, involving additional I/O; hence I/O is saved here (almost halved on joiner). You can use any compression utility which works on stream - gzip, pigz (which is multi-threaded and hence, recommended) etc. Also, note that, compressor has to be set on donor and decompressor on joiner (though you can have decompressor set on donor and vice-versa for config homogeneity, it won't affect that particular SST). To use Xtrabackup-based compression as before use `compress` under `[xtrabackup]` as before, also having both enabled won't cause any failure (though you will be wasting CPU cycles with this).

option **`inno-backup-opts, inno-apply-opts, inno-move-opts`**

Default Empty

Type Quoted String

These group of options can be used to pass options to backup, apply, move stages of innobackupex. Note, this option is to be used to pass only those options which are innobackupex-only and thus cannot be provided in my.cnf. Otherwise, it is strongly recommended to pass xtrabackup options through my.cnf (under [xtrabackup]).

option **`sst-initial-timeout`**

Values 0 (Disabled)

Default 100

Unit seconds

This option is use to configure initial timeout (in seconds) to receive a first packet via SST. This has been implemented, so that if donor dies somewhere in between, joiner doesn't hang and wait forever. Thus joiner won't wait for more than 100 seconds to get a donor. The default should be sufficient, however, it is configurable, so you can set it appropriately for your cluster.

Note: If you are using `wsrep_sst_donor`, and you want joiner to strictly wait for donors listed in the variable and not fallback (ie. without a terminating comma at the end), **and** there is a possibility of **all** nodes in that variable to be unavailable, set this variable to 0 to disable it or set it to a higher value (maximum threshold that you want joiner to wait). You can also disable (or set higher) this if you believe all other nodes in the cluster can be potentially unavailable at any point in time (mostly in small clusters) or there is a high network latency / network disturbance (which can cause donor selection to take longer than 100 seconds).

4.3.2 Tar against xbstream

- Features - encryption, compression, parallel streaming, streaming incremental backups, compaction - won't work with tar. Refer to xbstream docs for more.

4.3.3 Xtrabackup SST Dependencies

Following are optional dependencies of PXC introduced by wsrep_sst_xtrabackup: (obvious and direct dependencies are not provided here)

- qpress for decompression. It is an optional dependency of *Percona XtraBackup* 2.1.4 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/tar for streaming. tar is default.
- pv. Required for `progress` and `rlimit`. Provided by pv.
- mkfifo. Required for `progress`. Provided by coreutils.
- mktemp. Required for `incremental`. Provided by coreutils.

4.3.4 Galera compatible encryption

Support for SSL encryption for just the key and crt files as implemented in Galera can be enabled with `encrypt=3` option. This has been implemented in `5.5.34-23.7.6` for compatibility with Galera. **NOTE**: This option does not provide certificate validation. In order to work correctly paths to the key and cert files need to be specified as well, like:

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

NOTE: This option can only be used when `wsrep_sst_method` is set to xtrabackup-v2 (which is default now).

4.3.5 Xtrabackup-based encryption

This is enabled when `encrypt` is set to 1 under [sst]. 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 disallow encryption from innobackupex if the above are in my.cnf under [xtrabackup]. For that reason, do the following:

1. If you want to use xtrabackup based encryption for SST but not otherwise, use `encrypt=1` under [sst] and provide xtrabackup_encrypt_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 those xtrabackup_encrypt_options either under [sst] or [xtrabackup].
3. If you don't want xtrabackup based encryption for SST but want it otherwise, use `encrypt=0` or `encrypt=2` and do **NOT** provide xtrabackup_encrypt_options under [xtrabackup]. You can still have them under [sst] though. You will need to provide those options on innobackupex commandline 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 you don't need worry about these options! (just don't provide them in my.cnf)

Note: The `encrypt` under [sst] is different from under [xtrabackup]. The former is for disabling/changing encryption mode, latter is to provide encryption algorithm. To disambiguate, if you need to provide latter under [sst] (which you need to, for points #1 and #2 above) then it should be specified as `encrypt-algo`.

Warning: An implication of the above is that if you specify xtrabackup_encrypt_options but `encrypt=0` under [sst], it will **STILL** be encrypted and SST will fail. Look at point#3 above for resolution.

4.4 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.

4.5 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.

4.5.1 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.

4.5.2 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.

4.5.3 Other Reading

- PXC - Failure Scenarios with only 2 nodes

4.6 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?).

4.6.1 Manually

Manual cluster monitoring can be done with myq_gadgets.

4.6.2 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`).

4.6.3 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`)

4.6.4 Other Reading

- Realtime stats to pay attention to in PXC and Galera

CHAPTER

FIVE

HOW-TOS

5.1 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
```

5.1.1 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

5.1.2 Installation

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

5.1.3 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 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.

5.1.4 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.

5.2 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
```

5.2.1 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

5.2.2 Installation

Installation information can be found in the *Installing Percona XtraDB Cluster from Binaries* 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`.

5.2.3 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 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.

5.2.4 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.

5.3 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.`

```
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,
```

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
```

5.4 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-Clu
   ```

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
```

In this example variable `wsrep_urls` is being used instead of `wsrep_cluster_address`. With this configuration, node will first try to reach a cluster on *10.93.46.58:4567* if there is no cluster node, then it will try on *10.93.46.59:4567* and then *10.93.46.60:4567*. If no nodes are up, it will start a new cluster. Variable `wsrep_urls` goes into the [mysql_safe] section so it's important that the mysql server instance is started with the */bin/mysql_safe* and not *bin/mysqld*.

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),
```

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-3e6a
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 sta
2014-01-30 09:52:43 26104 [Note] RSA private key file not found: /var/lib/mysql//private_key.pem. So
2014-01-30 09:52:43 26104 [Note] RSA public key file not found: /var/lib/mysql//public_key.pem. Some
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),
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 versio
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!

5.5 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
```

5.6 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
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/m
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
       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*'. Selecte
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 24
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),
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
}
```

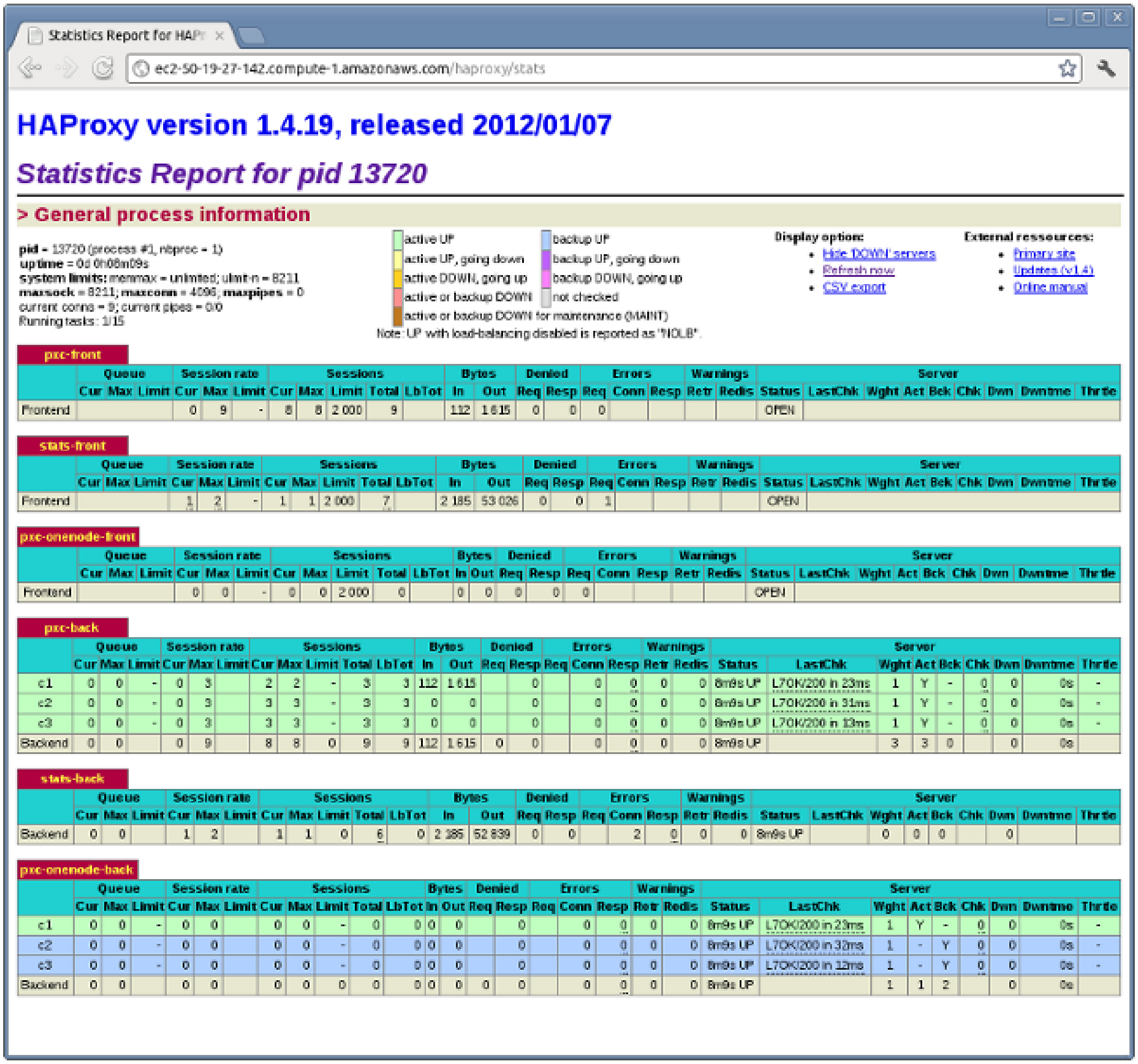
Also, you should add the new service to `/etc/services`.

```
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 HAPr
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	

5.6.1 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.

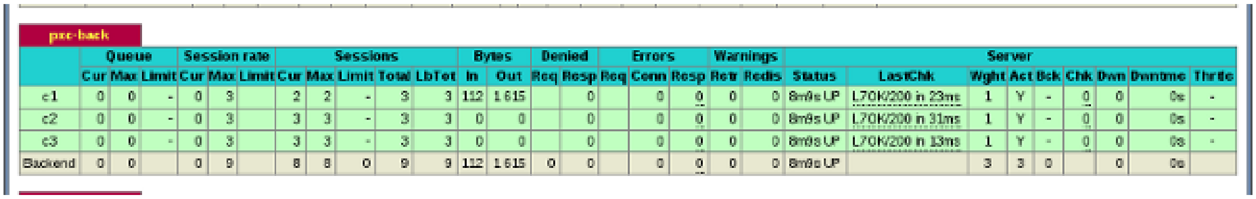
```
mysql> create database sbtest;
Query OK, 1 row affected (0.01 sec)

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-
```

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-
```



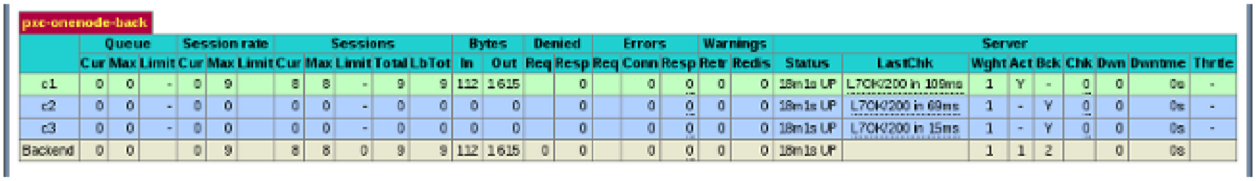
pxc-back

	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-
```



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.

5.7 How to Report Bugs

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

CHAPTER

SIX

REFERENCE

6.1 *Percona XtraDB Cluster* 5.6 Release notes

6.1.1 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.

6.1.2 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.

6.1.3 *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.

6.1.4 *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* 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.

6.1.5 *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 writesets in gcache.

wsrep_OSU_method is now both Global and Session variable.

Other bugs fixed: #1290526.

6.1.6 *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 *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 *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 writesets 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!

6.1.7 *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 *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!

6.1.8 *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!

6.1.9 *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!

6.1.10 *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* 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.

6.1.11 *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.

6.1.12 *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.

6.1.13 *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.

6.1.14 *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.

6.2 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`
- `Non-Primary`
- `Disconnected`

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_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.

6.3 Index of wsrep system variables

variable wsrep_auto_increment_control

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value ON

This variable manages the `auto_increment_increment` and `auto_increment_offset` variables automatically depending on the size of the cluster. This helps prevent `auto_increment` replication conflicts across the cluster by giving each node it's own range of `auto_increment` values. This may not be desirable depending on application's use and assumptions of auto-increments. It can be turned off in Master/Slave clusters.

variable `wsrep_causal_reads`

Version Info

- **5.6.20-25.7** – Variable deprecated by `wsrep_sync_wait`

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

Default Value OFF

In some cases, master may apply event 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`, slave will wait until that event is applied before doing any other queries. Enabling this variable will also result in larger latencies.

variable `wsrep_certify_nonPK`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value ON

When this variable is enabled, primary keys will be generated automatically for rows that don't have them. Using tables without primary keys is not recommended.

variable `wsrep_cluster_address`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

This variable needs to specify at least one other node's address that 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:

```
gcomm://<node1_ip>,<node2_ip>,<node3_ip>
```

Besides the IP address of the node, you can also specify port and options, for example:

```
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). This is not recommended for production after the cluster has been bootstrapped initially. If you want to bootstrap a new cluster, you should pass the `--wsrep-new-cluster` option when starting.

variable `wsrep_cluster_name`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value my_wsrep_cluster

This is the name of the cluster and should be identical on all nodes belonging to the same cluster.

Note: It should not exceed 32 characters.

variable **wsrep_convert_lock_to_trx**

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

This variable is used to convert `LOCK/UNLOCK TABLES` statements to `BEGIN/COMMIT`. Although this can help some older applications to work with multi-master setup it can also result in having huge writesets.

variable **wsrep_data_home_dir**

Command Line No

Config File Yes

Scope Global

Dynamic No

Default Value mysql *datadir*

This variable can be used to set up the directory where wsrep provider will store its files (like `grastate.dat`).

variable **wsrep_dbug_option**

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

This variable is used to send the `DBUG` option to the wsrep provider.

variable **wsrep_debug**

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

When this variable is set to `ON`, debug messages will also be logged to the `error_log`. This can be used when trying to diagnose problems or when submitting a bug.

variable **wsrep_desync**

Command Line No

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

This variable controls whether the node participates in Flow Control. Setting the `wsrep_desync` to `ON` does not automatically mean that a node will be out of sync with the cluster. It will continue to replicate in and out the writesets as usual. The only difference is that flow control will no longer take care of the `desynced` node. The result is that if `wsrep_local_recv_queue` gets higher than maximum allowed, all the other nodes will ignore the replication lag on the node being in `desync` mode. Toggling this back will require an IST or an SST depending on how long it was desynchronized. This is similar to cluster de-synchronization, which occurs during RSU TOI. Because of this, it's not a good idea to keep desync set for a long period of time, nor should you desync several nodes at once. Also, you'll need to desync a node before it starts causing flow control for it to have any effect. Node can also be desynchronized with `/*!  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 Yes

Config File Yes

Scope Session, Global

Dynamic Yes

Default Value OFF

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

variable `wsrep_drupal_282555_workaround`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

This variable was introduced as a workaround for Drupal/MySQL bug #282555. In some cases, duplicate key errors would occur when inserting the `default` value into the `auto_increment` field.

variable `wsrep_forced_binlog_format`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value NONE

This variable defines a binlog format that will always be effective regardless of session binlog format setting. Possible values for

- `ROW`
- `STATEMENT`

- MIXED
- NONE: Resets the forced state of the binlog format (default)

variable **wsrep_load_data_splitting**

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

This variable controls whether LOAD DATA transaction splitting is wanted or not. It doesn't work as expected with autocommit=0 when enabled.

variable **wsrep_log_conflicts**

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

This variable is used to control whether sole cluster conflicts should be logged. When enabled, details of conflicting *InnoDB* lock will be logged.

variable **wsrep_max_ws_rows**

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value 131072 (128K)

This variable is used to control maximum number of rows each writeset can contain. Anything bigger than the specified value will be rejected.

variable **wsrep_max_ws_size**

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value 1073741824 (1G)

This variable is used to control maximum writeset size (in bytes). Anything bigger than the specified value will be rejected.

variable **wsrep_mysql_replication_bundle**

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 0 (no grouping)

Range 0-1000

This variable controls how many replication events will be grouped together. Replication events are grouped in SQL slave thread by skipping events which may cause commit. This way the wsrep node acting in *MySQL* slave role and all other wsrep nodes in provider replication group, will see same (huge) transactions. The implementation is still experimental. This may help with the bottleneck of having only one *MySQL* slave facing commit time delay of synchronous provider.

variable `wsrep_node_address`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Format <ip address>[:port]

Default Value Usually set up as primary network interface (`eth0`)

This variable is used to specify the network address of the node. In some cases, when there are multiple NICs available, state transfer might not work if the default NIC is on different network. Setting this variable explicitly to the correct value will make SST and IST work correctly out of the box. Even in multi-network setups, IST/SST can be configured to use other interfaces/addresses.

variable `wsrep_node_incoming_address`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value <`wsrep_node_address`>:3306

This is the address at which the node accepts client connections. This information is used for status variable `wsrep_incoming_addresses` which shows all the active cluster nodes.

variable `wsrep_node_name`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value Host name

Unique name of the node. Defaults to the host name.

variable `wsrep_notify_cmd`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

This variable is used to set the notification command that the server should execute every time cluster membership or local node status changes.

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

This variable is used to enable/disable wsrep replication. When set to `OFF`, server will stop replication and behave like standalone *MySQL* server.

variable **wsrep_OSU_method**

Version Info

- **5.6.24-25.11** – Variable available both in global and session scope

Command Line Yes

Config File Yes

Scope Global and Session

Dynamic Yes

Default Value TOI

This variable can be used to select schema upgrade method. Available values are:

- `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 cluster node. This guarantees data consistency. In case of DDL statements, cluster will have parts of 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 high 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, 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, 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: Prior to *Percona XtraDB Cluster* `5.6.24-25.11`, `wsrep_OSU_method` was only a global variable. Current behavior is now 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 Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

Use this variable to enable new, transparent handling of preordered replication events (like replication from traditional master). When this variable is enabled, such events will be applied locally first before being replicated to the other nodes of 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 Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value None

This variable should contain the path to the Galera library (like `/usr/lib64/libgalera_smm.so` on *CentOS*/*RHEL* and `/usr/lib/libgalera_smm.so` on *Debian*/*Ubuntu*).

variable `wsrep_provider_options`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

This variable contains settings currently used by Galera library.

variable `wsrep_recover`

Command Line No

Config File Yes

Scope Global

Dynamic No

Default Value OFF

Location mysqld_safe

When server is started with this variable, it will parse Global Transaction ID (GTID) from log, and if the GTID is found, assign it as initial position for actual server start. This option is used to recover GTID.

variable `wsrep_reject_queries`

Command Line No

Config File Yes

Scope Global

Dynamic Yes

Default Value NONE

This variable can be used to reject queries for the node. This can be useful during upgrades for keeping node up (with provider enabled) without accepting queries. Using read-only is recommended here unless you want to kill existing queries. This variable accepts the following values:

- `NONE`: Nothing is rejected (default)
- `ALL`: All queries are rejected with `Error 1047:  Unknown command`
- `ALL_KILL`: All queries are rejected and existing client connections are also killed without waiting

Note: This variable doesn't affect Galera replication in any way, only the applications which connect to database are affected. If you want to desync a node, then use `wsrep_desync`.

variable **`wsrep_replicate_myisam`**

Version Info

- **5.6.24-25.11** – Variable available both in global and session scope

Command Line Yes

Config File Yes

Scope Session, Global

Dynamic No

Default Value OFF

This variable defines whether MyISAM should be replicated or not. It is disabled by default, because MyISAM replication is still experimental.

On the global level, `wsrep_replicate_myisam` can be set only before boot-up. 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 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 (involvement of MyISAM table will cause CTAS statement to skip replication)

variable `wsrep_restart_slave`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

This variable controls if *MySQL* slave should be restarted automatically when node joins back to cluster, because asynchronous replication slave thread is stopped when the node tries to apply next replication event while the node is in non-primary state.

variable `wsrep_retry_autocommit`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 1

This variable sets the number of times autocommitted transactions will be tried 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 the client's knowledge 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` transaction won't be retried and if it is set to `1`, it will be retried once.

variable `wsrep_slave_FK_checks`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value ON

This variable is used to control if Foreign Key checking is done for applier threads.

variable `wsrep_slave_threads`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value 1

This variable controls the number of threads that can apply replication transactions in parallel. Galera supports true parallel replication, replication that applies transactions in parallel only when it is safe to do so. The variable is dynamic, you can increase/decrease it at any time.

Note: When you decrease it, it won't kill the threads immediately but stop them after they are done applying current transaction (the effect with 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, in `JOINED` state for instance to speed up the catchup process to `SYNCED`.

You can also estimate the optimal value for this from `wsrep_cert_deps_distance` as suggested on this page.

You can also refer to this for more configuration tips.

variable `wsrep_slave_UK_checks`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

This variable is used to control if Unique Key checking is done for applier threads.

variable `wsrep_sst_auth`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Format <username>:<password>

This variable should contain the authentication information needed for State Snapshot Transfer (SST). Required information depends on the method selected in the `wsrep_sst_method`. More information about required authentication can be found in the *State Snapshot Transfer* documentation. This variable will appear masked in the logs and in the `SHOW VARIABLES` query.

variable `wsrep_sst_donor`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

This variable contains the name (`wsrep_node_name`) of the preferred donor for SST. If no node is selected as a preferred donor, it will be chosen from one of the available nodes automatically **if and only if** there is a terminating comma at the end (like 'node1,node2,'). Otherwise, if there is no terminating comma, the list of nodes in `wsrep_sst_donor` is considered absolute, and thus it won't fall back even if other nodes are available. Please check the note for `sst-initial-timeout` if you are using it without terminating comma or want joiner to wait more than default 100 seconds.

variable `wsrep_sst_donor_rejects_queries`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

When this variable is enabled, SST donor node will not accept incoming queries, instead it will reject queries with `UNKNOWN COMMAND` error code. This can be used to signal load-balancer that the node isn't available.

variable `wsrep_sst_method`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value xtrabackup-v2

Recommended xtrabackup-v2

This variable sets up the method for taking the State Snapshot Transfer (SST). Available values are:

- `xtrabackup`: Uses Percona XtraBackup to perform the SST, this method requires `wsrep_sst_auth` to be set up with <user>:<password> which *XtraBackup* will use on donor. Privileges and permissions needed for running *XtraBackup* can be found here.
- `xtrabackup-v2`: This is the same as `xtrabackup`, except that it uses newer protocol, hence is not compatible. This is the **recommended** option for PXC 5.5.34 and above. For more details, please check *Xtrabackup SST Configuration* and *Percona XtraDB Cluster Errata*. This is also the default SST method. For SST with older nodes (< 5.5.34), use `xtrabackup` as the SST method.

  Note: This method is currently recommended if you have `innodb-log-group_home-dir/innodb-data-home-dir` in your config. Refer to `sst-special-dirs` for more information.

- `rsync`: Uses `rsync` to perform the SST, this method doesn't use the `wsrep_sst_auth`
- `mysqldump`: Uses `mysqldump` to perform the SST, this method requires `wsrep_sst_auth` to be set up with <user>:<password>, where user has root privileges on the server.

  Note: This mehotd 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 if set so.

- `<custom_script_name>`: Galera supports Scriptable State Snapshot Transfer. This enables users to create their own custom script for performing an 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, it can be used when initially starting the cluster and manually restoring the same data to all nodes. It shouldn't be used as permanent setting because it could lead to data inconsistency across the nodes.

Note: Only `xtrabackup-v2` and `rsync` provide `gtid_mode async-slave` support during SST.

variable `wsrep_sst_receive_address`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value AUTO

This variable is used to configure address on which the node expects SST.

variable wsrep_start_position

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

This variable contains the `UUID:seqno` value. By setting all the nodes to have the same value for this option, cluster can be set up without the state transfer.

variable wsrep_sync_wait

Version Info

- **5.6.20-25.7** – Variable introduced

Command Line Yes

Config File Yes

Scope Global, Session

Dynamic Yes

This variable is used to control causality checks on some SQL statements, such as `SELECT`, `BEGIN`/`END`, `SHOW STATUS`, but not on some autocommit SQL statements `UPDATE` and `INSERT`. Causality check is determined by bitmask:

- `1` Indicates check on `READ` statements, including `SELECT`, `SHOW`, `BEGIN`/`START TRANSACTION`.
- `2` Indicates check on `UPDATE` and `DELETE` statements.
- `4` Indicates check on `INSERT` and `REPLACE` statements

This variable replaced the `wsrep_causal_reads` variable. Setting `wsrep_sync_wait` to `1` is the equivalent of setting `wsrep_causal_reads` to `ON`.

6.4 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";
```

6.4.1 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 **`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 either this or `gcache.keep_pages_size` variables are 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 specify total size of the pages in storage to keep for caching purposes. If only page storage is enabled, one page is always present.

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 No

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 No

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 `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.

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.

6.5 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.

```
$ 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
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#120715  9:45:56 server id 1  end_log_pos 107     Start: binlog v 4, server v 5.5.25-debug-l
# Warning: this binlog is either in use or was not closed properly.
ROLLBACK/*!*/;
BINLOG '
NHUCUA8BAAAAZwAAAGsAAAABAAQANS41LjI1LWRlYnVnLWxvZwAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAA0dQJQEzgNAAgAEgAEBAQEEgAAVAAEGggAAAAICAgCAA==
'/*!*/;
# at 107
#130226 11:48:50 server id 1  end_log_pos 83      Query   thread_id=3     exec_time=0     er
use `test`/*!*/;
SET TIMESTAMP=1361875730/*!*/;
SET @@session.pseudo_thread_id=3/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=0, @@session.unique_checks=1,
SET @@session.sql_mode=1437073440/*!*/;
SET @@session.auto_increment_increment=3, @@session.auto_increment_offset=2/*!*/;
/*!\C utf8 *//*!*/;
SET @@session.character_set_client=33,@@session.collation_connection=33,@@session.collation_
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*/;
```

This information can be used for checking the *MySQL* error log for the corresponding error message.

```
130226 11:48:50 [ERROR] Slave SQL: Error 'Unknown table 'test'' on query. Default database:
130226 11:48:50 [Warning] WSREP: RBR event 1 Query apply warning: 1, 3
130226 11:48:50 [Warning] WSREP: Ignoring error for TO isolated action: source: dd40ad88-7ff
```

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
  - `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
```

6.6 Frequently Asked Questions

6.6.1 Q: How do you solve locking issues like auto increment?

A: For auto-increment particular, Cluster changes auto_increment_offset for each new node. In the single node workload, locking handled by usual way how *InnoDB* handles locks. In case of write load on several nodes, Cluster uses optimistic locking and application may receive lock error in the response on COMMIT query.

6.6.2 Q: What if one of the nodes crashes and innodb recovery roll back some transactions?

A: When the node crashes, after the restart it will copy whole dataset from another node (if there were changes to data since crash).

6.6.3 Q: How can I check the Galera node health?

A: Your check should be simply:

```
SELECT 1 FROM dual;
```

3 different results 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 the node health with the `clustercheck` script. You need to set up `clustercheck` user:

```
GRANT USAGE ON *.* TO 'clustercheck'@'localhost' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005
```

You can then check the node 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 off-line 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: clustercheck 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"

6.6.4 Q: How does XtraDB Cluster handle big transaction?

A: XtraDB Cluster populates write set in memory before replication and this sets one limit for how large transactions make sense. There are wsrep variables for max row count and max size of of write set to make sure that server is not running out of memory.

6.6.5 Q: Is there a chance to have different table structure on the nodes?

What I mean is like having 4 nodes, 4 tables like sessions_a, sessions_b, sessions_c and sessions_d and have each only on one of the nodes?

A: Not at the moment for InnoDB tables. But it will work for MEMORY tables.

6.6.6 Q: What if a node fail and/or what if there is a network issue between them?

A: Then Quorum mechanism in XtraDB Cluster will decide what nodes can accept traffic and will shutdown nodes that not belong to quorum. Later when the failure is fixed, the nodes will need to copy data from working cluster.

6.6.7 Q: How would it handle split brain?

A: It would not handle it. The *split brain* is hard stop, XtraDB Cluster can't resolve it. That's why the minimal recommendation is to have 3 nodes. However there is possibility to allow a node to handle the traffic, option is:

```
wsrep_provider_options="pc.ignore_sb = yes"
```

6.6.8 Q: Is it possible to set up cluster without state transfer

A: It is possible in two ways:

1. By default Galera reads starting position from a text file <datadir>/grastate.dat. Just make this file identical on all nodes, and there will be no state transfer upon start.
2. With `wsrep_start_position` variable - start the nodes with the same *UUID:seqno* value and there you are.

6.6.9 Q: I have a two nodes setup. When node1 fails, node2 does not accept commands, why?

A: This is expected behavior, to prevent *split brain*. See previous question.

6.6.10 Q: What tcp ports are used by Percona XtraDB Cluster?

A: You may need to open up to 4 ports if you are using firewall.

1. Regular MySQL port, default 3306.
2. Port for group communication, default 4567. It can be changed by the option:

   ```
   wsrep_provider_options ="gmcast.listen_addr=tcp://0.0.0.0:4010; "
   ```

3. Port for State Transfer, default 4444. It can be changed by the option:

   ```
   wsrep_sst_receive_address=10.11.12.205:5555
   ```

4. Port for Incremental State Transfer, default port for group communication + 1 (4568). It can be changed by the option:

   ```
   wsrep_provider_options = "ist.recv_addr=10.11.12.206:7777; "
   ```

6.6.11 Q: Is there "async" mode for Cluster or only "sync" commits are supported?

A: There is no "async" mode, all commits are synchronous on all nodes. Or, to be fully correct, the commits are "virtually" synchronous. Which means that transaction should pass "certification" on nodes, not physical commit. "Certification" means a guarantee that transaction does not have conflicts with another transactions on corresponding node.

6.6.12 Q: Does it work with regular MySQL replication?

A: Yes. On the node you are going to use as master, you should enable log-bin and log-slave-update options.

6.6.13 Q: Init script (/etc/init.d/mysql) does not start

A: Try to disable SELinux. Quick way is:

```
echo 0 > /selinux/enforce
```

6.6.14 Q: I'm getting "nc: invalid option – 'd"' in the sst.err log file

A: 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 (bug #959970).

6.7 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.

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.

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`, 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.

innodb_file_per_table InnoDB option to use separate .ibd files for each table.

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 `.frm` extension containing the table definition (for all storage engines).

.ibd On a multiple tablespace setup (*innodb_file_per_table* enabled), *MySQL* will store each newly created table on a file with a `.ibd` extension.

.MYD Each *MyISAM* table has `.MYD` (MYData) file which contains the data on it.

.MYI Each *MyISAM* table has `.MYI` (MYIndex) file which contains the table's indexes.

.MRG Each table using the **MERGE** storage engine, besides of a *.frm* file, will have *.MRG* file containing the names of the *MyISAM* tables associated with it.

.TRG File containing the triggers associated to a table, e.g. `mytable.TRG`. With the *.TRN* file, they represent all the trigger definitions.

.TRN File containing the triggers' Names associated to a table, e.g. `mytable.TRN`. With the *.TRG* file, they represent all the trigger definitions.

.ARM Each table with the **Archive Storage Engine** has `.ARM` file which contains the metadata of it.

.ARZ Each table with the **Archive Storage Engine** has `.ARZ` file which contains the data of it.

.CSM Each table with the **CSV Storage Engine** has `.CSM` file which contains the metadata of it.

.CSV Each table with the **CSV Storage** engine has `.CSV` file which contains the data of it (which is a standard Comma Separated Value file).

.opt *MySQL* stores options of a database (like charset) in a file with a `.opt` extension in the database directory.

- *genindex*
- *search*