PERCONA

www.percona.com

Percona XtraDB Cluster 5.5 Documentation

Release 5.5.41-25.11.1

Nov 14, 2022

CONTENTS

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

Percona XtraDB Cluster provides:

- Synchronous replication. Transaction either commited 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

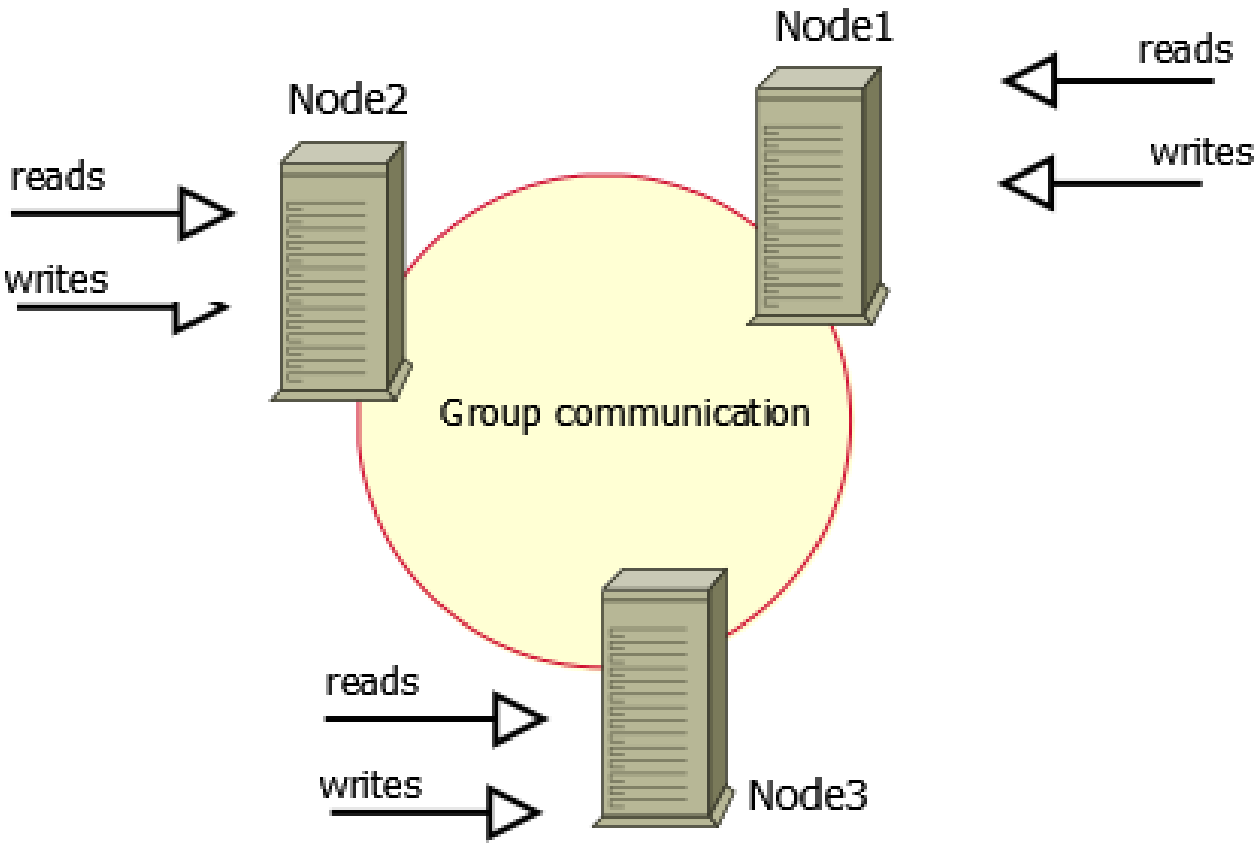
INTRODUCTION

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 direct read queries to any node.

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 2.x, a generic Synchronous Multi-Master replication plugin for transactional applications.

Galera library is developed by Codership Oy.

Galera 2.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.
- **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. 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 XtraDB Cluster nodes, and only one of the them can successfully commit. The failing one will be aborted. For cluster level aborts, 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, 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.

1.3 Percona XtraDB Cluster Errata (as of 5.5.34)

1.3.1 Known Issues

Following are issues which may impact you while running PXC:

- Create Table As Select (CTAS) can cause deadlocks and server hang when used with explicit TEMPORARY tables.
- bug #1192834: Joiner may crash after SST from donor with compaction enabled. Workaround is to disable the index compaction (compact under [xtrabackup]), if enabled. This crash requires specific configuration, hence you may not be affected. Also, this doesn't require any fix from PXC, but Xtrabackup with the fix included should do.
- For Debian/Ubuntu users: *Percona XtraDB Cluster* `5.5.33-23.7.6` onwards has a new dependency, the `socat` package. If the `socat` is not previously installed, `percona-xtradb-cluster-server-5.5` may be held back. In order to upgrade, you need to either install `socat` before running the `apt-get upgrade` or with the following command: `apt-get install percona-xtradb-cluster-server-5.5`. For *Ubuntu* users the `socat` package is in the universe repository, so the repository will have to be enabled in order to install the package.
- For fresh installs, if you skip the galera package on yum install command-line, yum may end up installing either of Percona-XtraDB-Cluster-galera-2 or 3. Hence, it is imperative to provide Percona-XtraDB-Cluster-galera-3 for PXC56 and Percona-XtraDB-Cluster-galera-2 for PXC55 on yum install command-line. Similarly for apt-get. Upgrades are not affected. This is not a bug but a situation due to existence of multiple providers. Please refer to installation guides of yum and apt for more details.
- Partition tables are not replicated unless debugging is turned on.

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

1.3.2 Major changes

Xtrabackup SST method compatibility notes:

- **5.5.33** introduced an enhanced Xtrabackup method that broke backward compatibility of SST with older versions of PXC. This is fixed in **5.5.34** by renaming the new method `xtrabackup-v2`, with default still being `xtrabackup` for backward compatibility. The following should be considered when upgrading:
  - If you are upgrading from **5.5.32** and lower: **5.5.34** Xtrabackup SST is backwards compatible. It is recommended you skip **5.5.33** when upgrading.
  - If you are upgrading from **5.5.33**: When upgrading, change your wsrep_sst_method=xtrabackup-v2 before restarting the **5.5.34** and up nodes

  Once you are on **5.5.34** and up, you can migrate to the new method by changing your wsrep_method on all your nodes. Note that the SST method of the Joiner is what is used when SST is needed, but the Donor must support the same method.
- Since, newer features, henceforth, are added only to xtrabackup-v2, it is recommended to use xtrabackup-v2 (to use new features) and use xtrabackup only when older (older than **5.5.33**) nodes are present. Also, note that, xtrabackup should support all the major options here, any v2 only options will be noted specifically (but older versions of xtrabackup still won't support newer features like encryption, compression, hence wsrep_sst_method=xtrabackup is recommended only for basic SST and during upgrades to maintain compatibility).

- So, if you are doing SST between a PXC **5.5.33** and PXC **5.5.34** node, due the above change, SST can fail, the workaround for this is to `ln -s /usr/bin/wsrep_sst_xtrabackup /usr/bin/wsrep_sst_xtrabackup-v2` on the donor node if it is 33 (and joiner is 34), `ln -sf /usr/bin/wsrep_sst_xtrabackup-v2 /usr/bin/wsrep_sst_xtrabackup` on the donor node if it is 34 (and joiner is 33). But this is not recommended, especially the latter (strong discouraged). Hence, all nodes with PXC **5.5.33** are strongly recommended to upgrade to PXC **5.5.34** if SST backward compatibility is a concern.

1.3.3 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 and *RHEL* 6
- *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.

Automatic Installation

Execute the following command as a `root` user:

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

You should see some output such as the following:

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

Resolving package conflicts

In *CentOS* 5 `mysql` package (a dependency of `perl-DBD-MySQL` for `libmysqlclient.so.15`) conflicts with `Percona-XtraDB-Cluster-client-55.x86_64` package. To avoid this, install `Percona-Server-shared-compat-51` first and then proceed to install *Percona XtraDB Cluster* with `yum install Percona-XtraDB-Cluster-55`. Also, in case, `mysql` package was installed, you will need to remove the it before installing *Percona XtraDB Cluster* as mentioned before

Install XtraDB Cluster

Following command will install Cluster packages:

```
$ yum install Percona-XtraDB-Cluster-55
```

Instead of `Percona-XtraDB-Cluster-55` you can install `Percona-XtraDB-Cluster-full-55` meta-package which will install `Percona-XtraDB-Cluster-devel-55`, `Percona-XtraDB-Cluster-test-55`, `Percona-XtraDB-Cluster-debuginfo-55`, `Percona-XtraDB-Cluster-galera-2-debuginfo`, and `Percona-XtraDB-Cluster-shared-55` 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` (both `$basearch` and `noarch`). **NOTE:** You'll need to install the Percona repository first if this hasn't been done already.

Percona `apt` Repository

Supported Architectures

- x86_64 (also known as amd64)
- x86

Supported Releases

Debian

- 7.0 wheezy

Ubuntu

- 12.04LTS precise
- 14.04LTS trusty

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

Installing from Repositories

1. Fetch the repository packages from Percona web:

```
wget https://repo.percona.com/apt/percona-release_0.1-4.$(lsb_release -sc)_all.deb
```

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

```
sudo dpkg -i percona-release_0.1-4.$(lsb_release -sc)_all.deb
```

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

1. Update the local cache:

```
sudo apt-get update
```

1. Install the server package:

```
sudo apt-get install percona-xtradb-cluster-55
```

Note: For Ubuntu 14.04 (Trusty) `percona-xtradb-cluster-galera-2.x` will have to be specified with the meta package so the installation command should look like:

```
$ sudo apt-get install percona-xtradb-cluster-55 percona-xtradb-cluster-galera-2.x
```

Note: Alternatively, you can install `percona-xtradb-cluster-full-55` meta-package which will install the following additional packages:

- `percona-xtradb-cluster-test-5.5`,
- `percona-xtradb-cluster-5.5-dbg`,
- `percona-xtradb-cluster-garbd-2.x`,
- `percona-xtradb-cluster-galera-2x-dbg`,
- `percona-xtradb-cluster-garbd-2.x-dbg` and
- `libmysqlclient18`

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

Note: Garbd is packaged separately as part of Debian split packaging. The garbd debian package is `percona-xtradb-cluster-garbd-2.x`. The package contains, garbd, daemon init script and related config files. This package will be installed if you install the `percona-xtradb-cluster-full-55` meta package.

Percona *apt* Testing repository

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

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

Apt-Pinning the packages

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

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

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

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

This is the recommend way of installing where possible.

YUM-Based Systems

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

```
$ yum install Percona-XtraDB-Cluster-55
```

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

Note: In *CentOS* 5 `mysql` package (a dependency of `perl-DBD-MySQL` for `libmysqlclient.so.15`) conflicts with `Percona-XtraDB-Cluster-client-55.x86_64` package. To avoid this, install `Percona-Server-shared-compat-51` first and then proceed to install *Percona XtraDB Cluster* with `yum install Percona-XtraDB-Cluster-55`. Also, in case, `mysql` package was installed, you will need to remove the it before installing *Percona XtraDB Cluster* as mentioned before.

DEB-Based Systems

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

```
$ sudo apt-get install percona-xtradb-cluster-55
```

Note: For Ubuntu 14.04 (Trusty) `percona-xtradb-cluster-galera-2.x` will have to be specified with the meta package so the installation command should look like:

```
$ sudo apt-get install percona-xtradb-cluster-55 percona-xtradb-cluster-galera-2.x
```

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

The source code is available from the *Launchpad* project here. The easiest way to get the code is with **`bzr branch`** of the desired release, such as the following:

```
bzr branch lp:percona-xtradb-cluster
```

You should then have a directory named after the release you branched, such as `percona-xtradb-cluster`.

Note: In place of a full bzr branch (which can take hours), if the intention is just to build, a source ball or a lightweight checkout, might be better options.

2.2.1 Compiling on Linux

Prerequisites

The following packages and tools must be installed to compile *Percona XtraDB Cluster* from source. These might vary from system to system.

In Debian-based distributions, you need to:

```
$ apt-get install build-essential flex bison automake autoconf bzr \
  libtool cmake libaio-dev mysql-client libncurses-dev zlib1g-dev libboost-dev
```

In RPM-based distributions, you need to:

```
$ yum install cmake gcc gcc-c++ libaio libaio-devel automake autoconf bzr \
  bison libtool ncurses5-devel boost
```

Compiling

The most esiest way to build binaries is to run script:

```
BUILD/compile-pentium64-wsrep
```

If you feel confident to use cmake, you make compile with cmake adding -DWITH_WSREP=1 to parameters.

Note: Galera tree needs to be checked out in the source directory of `percona-xtradb-cluster`. Otherwise the build - `build-binary.sh` would fail.

Examples how to build RPM and DEB packages can be found in packaging/percona directory in the source code.

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:

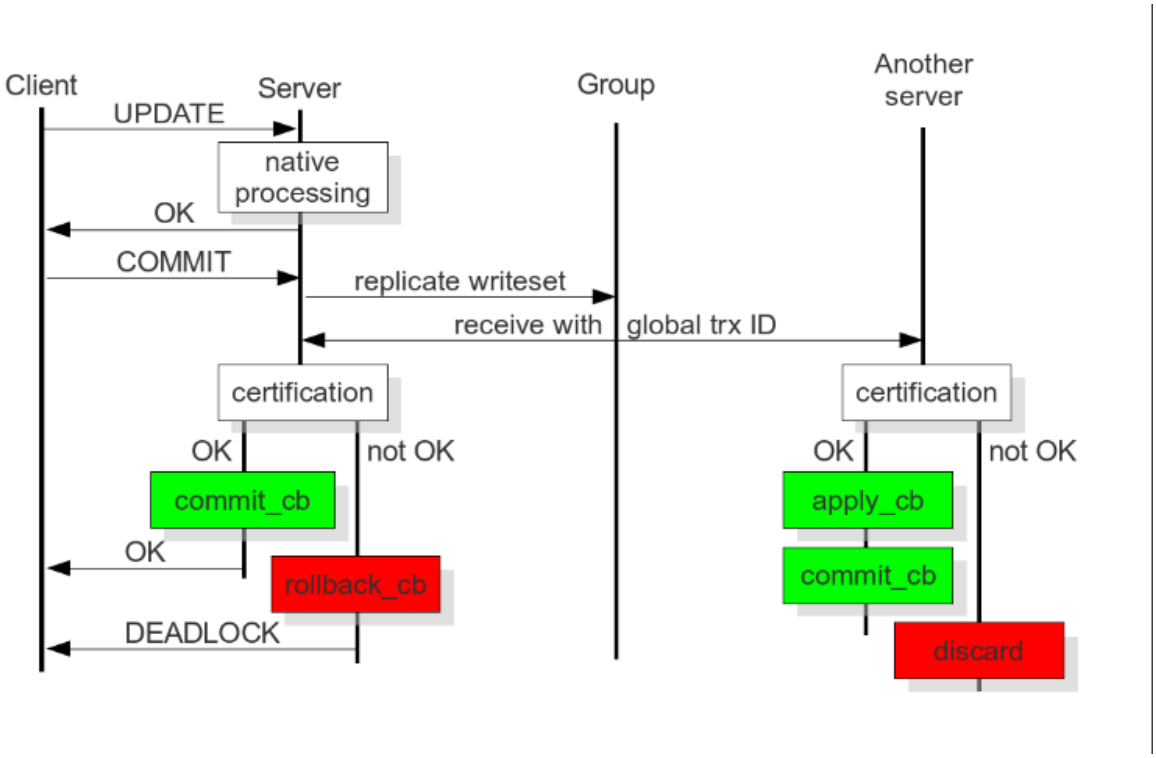


Fig. 3.1: *Image source:* Galera documentation - HOW CERTIFICATION-BASED REPLICATION WORKS

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
```

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 Version Specific Information

- *5.5.31-23.7.5* - New bootstrap init script option `bootstrap-pxc` introduced.

4.1.2 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 (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.

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 Using *Percona Xtrabackup*

This is the least blocking method as it locks the tables only to copy the *MyISAM* system tables. *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/
```

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

After *5.5.31-23.7.5* this method introduced the new `my.cnf` option which can be used to set *XtraBackup* streaming option:

```
[sst]
streamfmt=xbstream
```

Currently supported formats are tar (default, used in previous releases) and xbstream. Some of the benefits of choosing the xbstream over tar is that it supports more advanced features like encrypted, compact and compressed backups.

Note: Currently there is a limitations when using the `xbstream` option that the *datadir* needs to be empty before starting the SST or otherwise the process will fail and the node will not be able to join the cluster.

4.2.2 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.

4.2.3 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.

4.2.4 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.

Warning: *Xtrabackup SST Configuration* implementation added in *Percona XtraDB Cluster* *`5.5.33-23.7.6`* has been renamed to `xtrabackup-v2`, so *`wsrep_sst_method`* =xtrabackup will use xtrabackup implementation before *`5.5.33-23.7.6`* and will be compatible with older *Percona XtraDB Cluster* versions. In order to use the new version *`wsrep_sst_method`* should be set to `xtrabackup-v2`.

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.
- Following options are only applicable for `5.5.33-23.7.6` and above. For `5.5.31-23.7.5` refer to documentation in wsrep_sst_xtrabackup.

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-opensstunnel.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$\|.
↪*RPM_UPGRADE_MARKER$\|.*RPM_UPGRADE_HISTORY$\|.*\.xyz$'
```

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

option `sst_special_dirs`

Values 0,1

Default 0

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* *`5.5.34-25.9`*. 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.

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.

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

Percona repository should be set up as described in the *Percona yum Repository* guide. To enable the repository following command should be used:

```
$ rpm -Uhv http://www.percona.com/downloads/percona-release/percona-release-0.0-1.x86_
↪64.rpm
```

Following command will install *Percona XtraDB Cluster* packages:

```
$ yum install Percona-XtraDB-Cluster-server-55 Percona-XtraDB-Cluster-client-55␣
↪Percona-XtraDB-Cluster-galera-2
```

When these two commands have been executed successfully on all three nodes *Percona XtraDB Cluster* is installed.

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 PROCESS, RELOAD, LOCK TABLES, REPLICATION CLIENT ON *.* TO
↪'sstuser'@'localhost';
mysql@percona1> FLUSH PRIVILEGES;
```

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

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

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

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

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

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

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

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

# Node #2 address
wsrep_node_address=192.168.70.72
```

```
# Cluster name
wsrep_cluster_name=my_centos_cluster

# SST method
wsrep_sst_method=xtrabackup-v2

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

Second node can be started with the following command:

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

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

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

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

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

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

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

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

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

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

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

```
innodb_autoinc_lock_mode=2

# Node #3 address
wsrep_node_address=192.168.70.73

# Cluster name
wsrep_cluster_name=my_centos_cluster

# SST method
wsrep_sst_method=xtrabackup-v2

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

Third node can now be started with the following command:

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

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

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

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

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

# 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 *XtraBackup* new user needs to be set up with proper privileges:

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

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

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

```
[mysqld]

datadir=/var/lib/mysql
user=mysql

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

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

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

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

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

# Node #2 address
wsrep_node_address=192.168.70.62

# Cluster name
wsrep_cluster_name=my_ubuntu_cluster
```

```
# SST method
wsrep_sst_method=xtrabackup

#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

#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 Setting up Galera Arbitrator

Galera Arbitrator <http://galeracluster.com/documentation-webpages/arbitrator.html> is a member of the *Percona XtraDB Cluster* which is used for voting in case you have a small number of servers (usually two) and don't want to add any more resources. For example arbitrator can be set up on existing application server or on load balancer.

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

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

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

5.3.1 Installation

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

```
root@ubuntu:~# apt-get install percona-xtradb-cluster-garbd-2
```

on Debian/Ubuntu distributions, or:

```
[root@centos ~]# yum install Percona-XtraDB-Cluster-garbd-2
```

on CentOS/RHEL distributions.

5.3.2 Configuration

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

Configuration file should look like this after installation:

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

# REMOVE THIS AFTER CONFIGURATION
```

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

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

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

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

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

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

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

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

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

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

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

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

- On Debian or Ubuntu:

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

- On Red Hat Enterprise Linux or CentOS:

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

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

- On Debian or Ubuntu:

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

- On Red Hat Enterprise Linux or CentOS:

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

5.4 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-5.5.24-23.6.342.Linux.x86_64):

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

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

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

And you can check used ports:

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

After first node, we start second and third:

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

Successful start will produce the following output:

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

Cluster size can be checked with the:

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

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

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

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

```
/etc/my.4000.cnf

[mysqld]
port = 4000
socket=/tmp/mysql.4000.sock
datadir=/data/bench/d1
basedir=/usr/local/Percona-XtraDB-Cluster-5.5.24-23.6.342.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-5.5.24-23.6.342.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-5.5.24-23.6.342.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-5.5.24-23.6.342.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-5.5.24-23.6.342.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-5.5.24-23.6.342.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
```

5.5 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*.

Install 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-55 Percona-XtraDB-Cluster-client-55␣
↪Percona-XtraDB-Cluster-galera-2
```

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

wsrep_cluster_address=gcomm://10.93.46.58,10.93.46.59,10.93.46.60
binlog_format=ROW

wsrep_provider=/usr/lib64/libgalera_smm.so

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

innodb_autoinc_lock_mode=2
```

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

```
wsrep_node_name=node3
```

6. Start the *Percona XtraDB Cluster*

On the first node:

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

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

```
140113 12:23:43 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.5.34-55'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona XtraDB␣
↪Cluster (GPL), wsrep_25.9.r3928
```

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):

```
140113 17:29:34 [Note] WSREP: New cluster view: global state: fb031d6e-7c4c-11e3-bf3a-
↪ae73fcffc079:0, view# 8: Primary, number of nodes: 2, my index: 1, protocol version␣
↪2
140113 17:29:34 [Warning] WSREP: Gap in state sequence. Need state transfer.
140113 17:29:36 [Note] WSREP: Running: 'wsrep_sst_rsync --role 'joiner' --address '10.
↪93.46.59' --auth '' --datadir '/mnt/data/' --defaults-file '/etc/my.cnf' --parent
↪'29627''
cat: /var/lib/mysql//rsync_sst.pid: No such file or directory
140113 17:29:36 [Note] WSREP: Prepared SST request: rsync|10.93.46.59:4444/rsync_sst
140113 17:29:36 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
140113 17:29:36 [Note] WSREP: Assign initial position for certification: 0, protocol␣
↪version: 2
140113 17:29:36 [Warning] WSREP: Failed to prepare for incremental state transfer:␣
↪Local state UUID (00000000-0000-0000-0000-000000000000) does not match group state␣
↪UUID (fb031d6e-7c4c-11e3-bf3a-ae73fcffc079): 1 (Operation not permitted)
at galera/src/replicator_str.cpp:prepare_for_IST():445. IST will be unavailable.
140113 17:29:36 [Note] WSREP: Node 1 (node1) requested state transfer from '*any*'.␣
↪Selected 0 (node1)(SYNCED) as donor.
140113 17:29:36 [Note] WSREP: Shifting PRIMARY -> JOINER (TO: 0)
140113 17:29:36 [Note] WSREP: Requesting state transfer: success, donor: 0
140113 17:29:39 [Note] WSREP: 0 (node1): State transfer to 1 (node1) complete.
140113 17:29:39 [Note] WSREP: Member 0 (node1) synced with group.
WSREP_SST: [INFO] Joiner cleanup. (20140113 17:29:39.792)
WSREP_SST: [INFO] Joiner cleanup done. (20140113 17:29:40.302)
140113 17:29:40 [Note] WSREP: SST complete, seqno: 0
140113 17:29:40 [Note] Plugin 'FEDERATED' is disabled.
140113 17:29:40 InnoDB: The InnoDB memory heap is disabled
140113 17:29:40 InnoDB: Mutexes and rw_locks use GCC atomic builtins
140113 17:29:40 InnoDB: Compressed tables use zlib 1.2.3
140113 17:29:40 InnoDB: Using Linux native AIO
140113 17:29:40 InnoDB: Initializing buffer pool, size = 128.0M
140113 17:29:40 InnoDB: Completed initialization of buffer pool
140113 17:29:40 InnoDB: highest supported file format is Barracuda.
InnoDB: Log scan progressed past the checkpoint lsn 1598437
140113 17:29:40  InnoDB: Database was not shut down normally!
InnoDB: Starting crash recovery.
InnoDB: Reading tablespace information from the .ibd files...
InnoDB: Restoring possible half-written data pages from the doublewrite
InnoDB: buffer...
InnoDB: Doing recovery: scanned up to log sequence number 1598607
140113 17:29:40  InnoDB: Waiting for the background threads to start
140113 17:29:41 Percona XtraDB (http://www.percona.com) 5.5.34-rel32.0 started; log␣
↪sequence number 1598607
140113 17:29:41 [Note] Event Scheduler: Loaded 0 events
140113 17:29:41 [Note] WSREP: Signalling provider to continue.
140113 17:29:41 [Note] WSREP: SST received: fb031d6e-7c4c-11e3-bf3a-ae73fcffc079:0
140113 17:29:41 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.5.34-55'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona XtraDB␣
↪Cluster (GPL), wsrep_25.9.r3928
```

```
140113 17:29:41 [Note] WSREP: 1 (node1): State transfer from 0 (node1) complete.
140113 17:29:41 [Note] WSREP: Shifting JOINER -> JOINED (TO: 0)
140113 17:29:41 [Note] WSREP: Member 1 (node1) synced with group.
140113 17:29:41 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 0)
140113 17:29:41 [Note] WSREP: Synchronized with group, ready for connections
140113 17:29:41 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
```

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!

5.6 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
```

5.7 Setting up PXC reference architecture with HAProxy

This tutorial is a step-by-step guide to set up *Percona XtraDB Cluster*, in a virtualized test sandbox. This example uses Amazon EC2 micro instances, but the content here is applicable for any kind of virtualization technology (for

example VirtualBox). You will need 4 virtual machines. 3 for *Percona XtraDB Cluster* and 1 for the client, which will have *HAProxy*. In this how-to CentOS 6 is used as the operating system, the instructions are similar for any Linux distribution.

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

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

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

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

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

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

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

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

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

log_slave_updates

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

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

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

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

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

For the third node, the differences look like this.

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

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

```
120619 13:20:17 [Note] WSREP: State transfer required:
      Group state: 77c9da88-b965-11e1-0800-ea53b7b12451:97
      Local state: 00000000-0000-0000-0000-000000000000:-1
120619 13:20:17 [Note] WSREP: New cluster view: global state: 77c9da88-b965-11e1-0800-
↪ea53b7b12451:97, view# 18: Primary, number of nodes: 3, my index: 0, protocol␣
↪version 2
120619 13:20:17 [Warning] WSREP: Gap in state sequence. Need state transfer.
120619 13:20:19 [Note] WSREP: Running: 'wsrep_sst_xtrabackup 'joiner' '10.195.206.117
↪' '' '/var/lib/mysql/' '/etc/my.cnf' '20758' 2>sst.err'
120619 13:20:19 [Note] WSREP: Prepared |SST| request: xtrabackup|10.195.206.117:4444/
↪xtrabackup_sst
120619 13:20:19 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
120619 13:20:19 [Note] WSREP: Assign initial position for certification: 97, protocol␣
↪version: 2
120619 13:20:19 [Warning] WSREP: Failed to prepare for incremental state transfer:␣
↪Local state UUID (00000000-0000-0000-0000-000000000000) does not match group state␣
↪UUID (77c9da88-b965-11e1-0800-ea53b7b12451): 1 (Operation not permitted)
       at galera/src/replicator_str.cpp:prepare_for_IST():439. IST will be␣
↪unavailable.
120619 13:20:19 [Note] WSREP: Node 0 (ip-10-244-33-92) requested state transfer from
↪'*any*'. Selected 1 (ip-10-112-39-98)(SYNCED) as donor.
120619 13:20:19 [Note] WSREP: Shifting PRIMARY -> JOINER (TO: 102)
120619 13:20:19 [Note] WSREP: Requesting state transfer: success, donor: 1
120619 13:20:59 [Note] WSREP: 1 (ip-10-112-39-98): State transfer to 0 (ip-10-244-33-
↪92) complete.
120619 13:20:59 [Note] WSREP: Member 1 (ip-10-112-39-98) synced with group.
120619 13:21:17 [Note] WSREP: |SST| complete, seqno: 105
120619 13:21:17 [Note] Plugin 'FEDERATED' is disabled.
120619 13:21:17 InnoDB: The InnoDB memory heap is disabled
120619 13:21:17 InnoDB: Mutexes and rw_locks use GCC atomic builtins
120619 13:21:17 InnoDB: Compressed tables use zlib 1.2.3
120619 13:21:17 InnoDB: Using Linux native AIO
120619 13:21:17 InnoDB: Initializing buffer pool, size = 400.0M
120619 13:21:17 InnoDB: Completed initialization of buffer pool
120619 13:21:18 InnoDB: highest supported file format is Barracuda.
120619 13:21:18  InnoDB: Waiting for the background threads to start
120619 13:21:19 Percona XtraDB (http://www.percona.com) 1.1.8-rel25.3 started; log␣
↪sequence number 246661644
120619 13:21:19 [Note] Recovering after a crash using mysql-bin
120619 13:21:19 [Note] Starting crash recovery...
120619 13:21:19 [Note] Crash recovery finished.
120619 13:21:19 [Note] Server hostname (bind-address): '(null)'; port: 3306
120619 13:21:19 [Note]   - '(null)' resolves to '0.0.0.0';
120619 13:21:19 [Note]   - '(null)' resolves to '::';
120619 13:21:19 [Note] Server socket created on IP: '0.0.0.0'.
120619 13:21:19 [Note] Event Scheduler: Loaded 0 events
120619 13:21:19 [Note] WSREP: Signalling provider to continue.
```

```
120619 13:21:19 [Note] WSREP: Received |SST|: 77c9da88-b965-11e1-0800-ea53b7b12451:105
120619 13:21:19 [Note] WSREP: |SST| received: 77c9da88-b965-11e1-0800-ea53b7b12451:105
120619 13:21:19 [Note] WSREP: 0 (ip-10-244-33-92): State transfer from 1 (ip-10-112-
↪39-98) complete.
120619 13:21:19 [Note] WSREP: Shifting JOINER -> JOINED (TO: 105)
120619 13:21:19 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.5.24-log'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona␣
↪XtraDB Cluster (GPL), wsrep_23.6.r340
120619 13:21:19 [Note] WSREP: Member 0 (ip-10-244-33-92) synced with group.
120619 13:21:19 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 105)
120619 13:21:20 [Note] WSREP: Synchronized with group, ready for connections
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

To set it up, create the clustercheck user.

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

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

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

```
# clustercheck
HTTP/1.1 200 OK

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

Node is running.

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

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

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

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

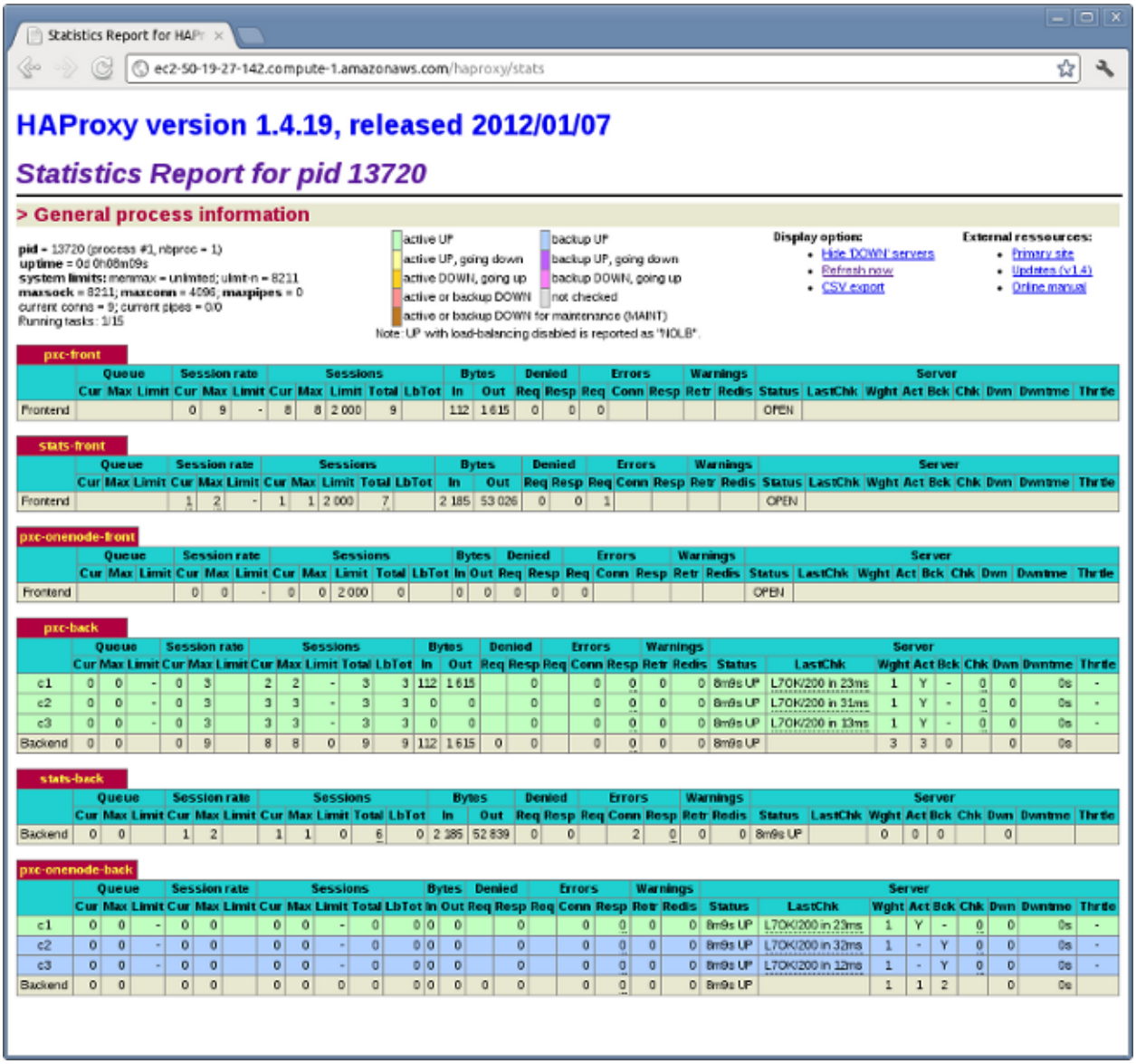
Also, you should add the new service to `/etc/services`.

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

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

```
# service xinetd restart
```

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



5.7.1 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-host=127.0.0.1 --mysql-port=3307 --mysql-user=sbtest --mysql-
↪password=sbpass --oltp-table-size=10000 prepare
```

You can now run the benchmark against the 3307 port.

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

pxc-back

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

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

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

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

pxc-onenode-back

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

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

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

5.8 How to Report Bugs

All bugs can be reported in Percona bug reporting system. Please note that error.log files from **all** the nodes need to be submitted.

CHAPTER

SIX

REFERENCE

6.1 *Percona XtraDB Cluster* Release notes

6.1.1 *Percona XtraDB Cluster* 5.5.41-25.12

Percona is glad to announce the release of *Percona XtraDB Cluster* on September 22nd, 2016. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.5.41-37.0 including all the bug fixes in it, Galera Replicator 2.11 and on wsrep API 25.11, *Percona XtraDB Cluster* 5.5.41-25.12 is now the current stable release. All of *Percona*'s software is open-source and free.

This is an General Availability release. 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.

Bugs fixed

Due to security reasons `ld_preload` libraries can now only be loaded from the system directories (`/usr/lib64`, `/usr/lib`) and the MySQL installation base directory. This fix also addresses issue with where limiting didn't work correctly for relative paths. Bug fixed #1624247.

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

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

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

Other bugs fixed: #1553938.

6.1.2 *Percona XtraDB Cluster* 5.5.41-25.11

Percona is glad to announce the release of *Percona XtraDB Cluster* on March 30th, 2015. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.5.41-37.0 including all the bug fixes in it, Galera Replicator 2.11 and on wsrep API 25.11, *Percona XtraDB Cluster* 5.5.41-25.11 is now the current stable release. All of *Percona*'s software is open-source and free.

This is an General Availability release. 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.

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.

`garbd` was returning incorrect return code, ie. when `garbd` was already started, return code was `0`. Bugs fixed #1308103 and #1422863.

`wsrep_sst_xtrabackup-v2` script was causing **`innobackupex`** to print a false positive stack trace into the log. Bug fixed #1407599.

MyISAM DDL (`CREATE TABLE` only) isn't replicated anymore 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). This also doesn't work if `default_storage_engine` is set to `MyISAM` (which is not recommended for *Percona XtraDB Cluster*) and a table is created without the `ENGINE` option. Bug fixed #1402338.

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.

`wsrep_causal_reads` variable was not honored when declared as global. Bug fixed #1361859.

`garbd` would not work when cluster address was specified without the port. Bug fixed #1365193.

`garbd` was running as root user on *Debian*. Bug fixed #1392388.

Errors in `garbd` init script stop/start functions have been fixed. Bug fixed #1367956.

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.

`gvwstate.dat` file was removed on joiner when *XtraBackup SST* method was used. Bug fixed #1388059.

`xtrabackup-v2` SST did not clean the undo log directory. Bug fixed #1394836.

`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.

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.

SST will now fail early if the `xtrabackup_checkpoints` file is missing on the joiner side. Bug fixed #1405985.

`socat` utility was not properly terminated after a timeout. Bug fixed #1409710.

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.

Conflict between enforce_storage_engine and *`wsrep_replicate_myisam`* for `CREATE TABLE` has been fixed. Bug fixed #1435482.

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.

Variable length arrays in WSREP code were causing debug builds to fail. Bug fixed #1409042.

Signal handling in `mysqld` has been fixed for SST processes. Bug fixed #1399175.

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: #1391634 and #1396757.

6.1.3 *Percona XtraDB Cluster* 5.5.39-25.11

Percona is glad to announce the release of *Percona XtraDB Cluster* on September 5th, 2014. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.5.39-36.0 including all the bug fixes in it, Galera Replicator 2.11 and on wsrep API 25.11), *Percona XtraDB Cluster* 5.5.39-25.11 is now the current stable release. All of *Percona*'s software is open-source and free.

This is an General Availability release. 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.

New Features

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).

systemd integration with *RHEL/CentOS* 7 is now available for *Percona XtraDB Cluster* (#1342223).

Bugs fixed

Percona XtraDB Cluster has implemented threadpool scheduling fixes. Bug fixed #1333348.

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.

Client connections were closed unconditionally before generating SST request. Fixed by avoiding closing connections when wsrep is initialized before storage engines. Bug fixed #1258658.

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.

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).

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.

Updating a unique key value could cause server hang if slave node has enabled parallel slaves. Bug fixed #1280896.

Fixed the events replication inconsistencies. Bug fixed #1312618.

Truncating the sorted version of multi-byte character conversion could lead to wsrep certification failures. Bug fixed #1314854.

`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` (`UNIV_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.

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.

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.

A memory leak in `wsrep_mysql_parse` function has been fixed. Bug fixed #1345023.

`SHOW STATUS` was generating debug output in the error log. Bug fixed #1347818.

The netcat in garbd init script has been replaced with nmap for compatibility in *CentOS* 7. Bug fixed #1349384.

Fixed `netcat` in SST script for *CentOS* 7 `nmap-ncat`. Bug fixed #1359767.

`percona-xtradb-cluster-garbd-3.x` package was installed incorrectly on *Debian*/*Ubuntu*. Bugs fixed #1360633 and #1334530.

Other bugs fixed: #1334331, #1338995, #1280270, #1272982, #1190774, and #1251765.

Percona XtraDB Cluster Errata can be found in our documentation.

6.1.4 *Percona XtraDB Cluster* 5.5.37-25.10

Percona is glad to announce the release of *Percona XtraDB Cluster* on May 8th, 2014. Binaries are available from downloads area or from our *software repositories*.

Based on Percona Server 5.5.37-35.0 including all the bug fixes in it, Galera Replicator 2.10 (including fixes in 2.9 and 2.10 milestones) and on wsrep API 25.10 (including fixes in 25.10), *Percona XtraDB Cluster* 5.5.37-25.10 is now the current stable release. All of *Percona*'s software is open-source and free.

This is an General Availability release. 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.

New Features

Percona XtraDB Cluster now supports stream compression/decompression with new xtrabackup-sst `compressor/decompressor` options.

`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-2.x` package.

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`.

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.

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.

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.

Support for Query Cache has been implemented.

Bugs fixed

To avoid disabling Parallel Apply in case of `SAVEPOINT` or `ROLLBACK TO SAVEPOINT` the `wsrep_cert_deps_distance` being `1.0` at all times. Bug fixed #1277703.

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.

`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.

Server would segfault on `INSERT DELAYED` with `wsrep_replicate_myisam` set to `1` due to unchecked dereference of `NULL` pointer. Bug fixed #1165958.

When `grastate.dat` file was not getting zeroed appropriately it would lead to RBR error during the IST. Bug fixed #1180791.

`gcomm` exception in PC `validate_state_msgs()` during cluster partitioning/re-merges" has been fixed. Bug fixed #1182367.

Due to the `Provides:` line in *Percona XtraDB Cluster* (which provides `Percona-Server-server`), the command `yum install Percona-Server-server` would install *Percona XtraDB Cluster* instead of the expected *Percona Server*. Bug fixed #1201499.

Replication of partition tables without binlogging enabled failed, partition truncation didn't work because of lack of TO isolation there. Bug fixed #1219605.

Exception during group merge after partitioning event has been fixed. Bug fixed #1232747.

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.

`CREATE TABLE AS SELECT` was not replicated, if the select result set was empty. Bug fixed #1246921.

`INSERT` would return deadlock instead of duplicate key on secondary unique key collision. Bug fixed #1255147.

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.

GCache file allocation could fail if file size was a multiple of page size. Bug fixed #1259952.

Percona XtraDB Cluster didn't validate the parameters of *`wsrep_provider_options`* when starting it up. Bug fixed #1260193.

Fixed rsync SST for compatibility with `rsync` version 3.1.0. Bug fixed #1261673.

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.

Node would get stuck and required restart if `DDL` was performed after `FLUSH TABLES WITH READ LOCK`. Bug fixed #1265656.

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.

`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.

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.

Galera2 is now installed in `/usr/lib/galera2/libgalera_smm.so` with a compatibility symlink to `/usr/lib/libgalera_smm.so`. Bug fixed #1279328.

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.

Updating an unique key value could cause server to hang if slave node has enabled parallel slaves. Bug fixed #1280896.

Fixed incorrect warnings and implemented better handling of repeated usage with same value for *wsrep_desync*. Bug fixed #1281696.

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 with `pc.bootstrap=1` at runtime in a `NON-PRIMARY` state would lead to crash. Bug fixed #1286450.

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. Bug fixed #1288479.

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.

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.

Compiling on *FreeBSD* 10.0 with `CLANG` would result in fatal error. Bug fixed #1309507.

`Use-after-free` memory corruption in `one_thread_per_connection_end` has been fixed. Bug fixed #1310875.

Xtrabackup wouldn't copy empty `test` database during the SST. Bug fixed #1231088.

Xtrabackup SST failed when `/tmp/test` directory existed, fixed by creating unique directories in SST script. Bug fixed #1294760.

Deadlock during server shutdown has been prevented. Bug fixed #1284670.

Race condition during update of *wsrep_slave_threads* has been avoided. Bug fixed #1290612.

Added new option to the init script to restart a node bootstrapped. Bug fixed #1291024.

Percona XtraDB Cluster server package no longer conflicts with `mysql-libs` package from CentOS repository. Bug fixed #1278516.

Server abort due to BF-BF lock conflict in `lock0lock.c` has been fixed. Bug fixed #1264809.

Crash due to segfault in `gcache::RingBuffer::get_new_buffer()` has been fixed. Bug fixed #1152565.

Applier would fail with `lock wait timeout exceeded` on rsync SST donor. Bug fixed #1255501.

rsync SST could hang due to missing `lsof`. Bug fixed #1313293.

Other bugs fixed: #1273101, #1297822, #1289776, #1279844, #1279343, #1278516, #1278417, #1277263, #1276994, #1272961, #1272723, #1269811, #1269351, #1269063, #1262887, #1262716, #1261996, #1261833, #1261138, #1269063, #1256887, #1256769, #1255501, and #1254633.

Percona XtraDB Cluster Errata can be found in our documentation.

6.1.5 *Percona XtraDB Cluster* 5.5.34-25.9

Percona is glad to announce the release of *Percona XtraDB Cluster* on December 3rd, 2013. Binaries are available from downloads area or from our *software repositories*.

This is an General Availability release. 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.

New Features

Percona XtraDB Cluster is now based on wsrep API 25 and Galera 25.2.x.

`RPM` packages are now made relocatable which means they now support installation to custom prefixes.

XtraBackup SST now supports `innodb_data_home_dir` and `innodb_log_home_dir` in the configuration file.

The binaries are now statically linked with regard to `Galera` library which depended on `OpenSSL` library.

Bugs fixed

Product suffix has been added to the *Percona XtraDB Cluster* `rpm` packages, which means that packages have been renamed from `Percona-XtraDB-Cluster-server` to `Percona-XtraDB-Cluster-server-55`. Bug fixed #1255616.

Fixed the dependency issue which caused Percona XtraDB Cluster 5.5 installation to fail on *Ubuntu* 12.04. Bug fixed #1247861.

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.

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 (in this case it was `TRUNCATE`), which would generates error messages on other nodes. Bug fixed #1194156.

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.

`CREATE TABLE AS SELECT` would fail with explicit temporary tables, when binlogging was enabled and `autocommit` was set to `0`. Bug fixed #1240098.

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 and cause deadlocks. 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 `Suggested` dependencies for *DEB* packages. Bug fixed #1250326.

`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.

Memory leak in `mem_root` has been fixed. Bug fixed #1249753.

Galera is now packaged with garbd init script. Bug fixed #1256769.

Other bugs fixed: bug fixed #1247980, bug fixed #891476, bugs fixed #1250805, bug fixed #1253923.

Note: Because some package names have been changed with the product suffix and and additional dependencies being added please check the *manual* before starting new installations. *Debian* users are requested to use `apt-get dist-upgrade` or `apt-get install` for upgrade, for more refer to installation *guide*.

Based on Percona Server 5.5.34-32.0 including all the bug fixes in it, Galera Replicator and on Codership wsrep API 5.5.34-25.9, *Percona XtraDB Cluster* 5.5.34-25.9 is now the current stable release. All of *Percona*'s software is open-source and free.

Percona XtraDB Cluster Errata can be found in our documentation.

6.1.6 *Percona XtraDB Cluster* 5.5.34-23.7.6

Percona is glad to announce the release of *Percona XtraDB Cluster* on November 4th, 2013. Binaries are available from downloads area or from our *software repositories*.

This is an General Availability release. 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.

Bugs fixed

Fixed the issues with upgrade on *Debian* caused by `debian-start` script. Bug fixed #1112363.

Adding the `AUTO_INCREMENT` column to a table would result in data inconsistency because different nodes use different `auto_increment_offset`. Bug fixed #587170.

`LOAD DATA INFILE` was not replicating properly in cluster. Bug fixed #1206129.

After a `COMMIT` processing was interrupted (e.g. certification failure) it didn't always release transactional `MDL` locks, which could lead to server deadlock while performing DDL. Bug fixed #1212955.

`pc.checksum` option was enabled by default, although its purpose was just to help with debugging. Bug fixed #1229107.

Server startup isn't interrupted if the `/var/lock/subsys/mysql` file was not cleaned up by the init script. Bug fixed #1231067.

State Snapshot Transfer cleanup of existing data directory is now configurable with a `cpat` parameter. Bug fixed #1231121.

Handler command and concurrent `DDL` could cause node hanging. The problem would happen when a handler session was aborted due to replication conflict. Bug fixed #1233353.

Command `service mysql status` was not producing valid exit codes, which could break the configuration management systems. Bug fixed #1237021.

Option `wsrep-new-cluster` could lead to server crash. Bug fixed #1238331.

No key information was provided for `DROP VIEW TOI` statement. Bug fixed #1240040.

`innodb_locks_unsafe_for_binlog` option is no longer checked because slave threads are `READ-COMMITTED` by default. Bug fixed #1240112.

If IO cache contained the `Query_log_event`, database pointer would be reset during transaction replay. This could lead to "1046 No database selected." error when next statement is issued. Bug fixed #1241760.

After test shutting down all the nodes indicate that all write set handlers were not released. Bug fixed #1244661.

`MDL` locks were not released properly on slaves at commit time. Bug fixed #1245333.

Implemented number of thread pool scheduler fixes. Bug fixed #1240500.

Running `FLUSH STATUS` would zero-up *`wsrep_cluster_size`* and *`wsrep_local_index`* variables. Bug fixed #1232789.

Fixed the dependency conflict between the `Percona-XtraDB-Cluster-shared` and `Percona-Server-shared-compat` packages. Bug fixed #1084309.

Fixed the memory leak in the `wsrep_write_cache()` function. Bug fixed #1230160.

Xtrabackup SST Configuration implementation added in *Percona XtraDB Cluster* *`5.5.33-23.7.6`* has been renamed xtrabackup-v2, so *`wsrep_sst_method`* =xtrabackup will use xtrabackup implementation before *`5.5.33-23.7.6`* and will be compatible with older *Percona XtraDB Cluster* versions. Bug fixed #1228618.

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. Bug fixed #1235244.

Other bugs fixed: bug fixed #1244741, bug fixed #1244667, bug fixed #1243150, bug fixed #1232890, bug fixed #999492, bug fixed #1245769, bug fixed #1244100.

Based on Percona Server 5.5.34-32.0 including all the bug fixes in it, Galera Replicator and on Codership wsrep API 5.5.34-25.9, *Percona XtraDB Cluster* 5.5.34-23.7.6 is now the current stable release. All of *Percona*'s software is open-source and free.

Percona XtraDB Cluster Errata can be found in our documentation.

6.1.7 *Percona XtraDB Cluster* 5.5.33-23.7.6

Percona is glad to announce the release of *Percona XtraDB Cluster* on September 18th, 2013. Binaries are available from downloads area or from our *software repositories*.

This is an General Availability release. 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.

New Features

Default *State Snapshot Transfer* method, defined in `wsrep_sst_method` has been changed from `mysqldump` to `rsync`.

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.

Variable `wsrep_sst_donor` has been extended to accept multiple hostnames that are preferred as *State Snapshot Transfer* donors. This can be helpful in case other nodes on the list go down or it can be used as a whitelist during automated provisioning.

Desync functionality has now been exposed to the client. This can be done either via `/*! WSREP_DESYNC */` comment on the query or by setting the global `wsrep_desync` variable to `1`.

Improvements to XtraBackup SST

Note:

There are several changes to Xtrabackup SST from 5.5.33 onwards. Make sure to check *XtraBackup SST* for details, and also incompatibilities for any issues.

Percona XtraDB Cluster has implemented progress indicator for *XtraBackup SST*.

Percona XtraDB Cluster has implemented new rate limiting, `rlimit`, option for *XtraBackup SST* that can be used to avoid saturating the donor node.

Percona XtraDB Cluster has added new *XtraBackup SST* `time` option that can be used to see how much time different stages of *State Snapshot Transfer* are taking.

Percona XtraDB Cluster has implemented additional *XtraBackup SST* encryption option. Beside standard *Percona XtraBackup* encryption, new `OpenSSL` based encryption can be specified in the `encrypt` option.

XtraBackup SST now works in two stages. This was implemented to avoid issues like bug bug:*1193240*.

Bugs fixed

When multiple slave threads were configured, if there was a query on different transaction that inserts a row, and a query on another transaction within the same thread id that depends on the first row (FK constraint), sometimes the second transaction would be executed first causing the Foreign Key violation. Bug fixed #1217653.

When variable `wsrep_provider` was set to none it would cause cluster to hang. Bug fixed #1208493.

Percona XtraDB Cluster would crash with message: `Error "no such a transition EXECUTING -> COMMITTED"` on the master node. This bug was fixed only for some cases. Bug fixed #1123233.

Running DDL commands while variable `wsrep_OSU_method` was set to `TOI - Total Order Isolation` could lead to server deadlock. Bug fixed #1212955.

Stopping mysql process with `inet` script didn't work if `PID` file was provided as `pid_file` option in the `my.cnf` configuration file. Bug fixed #1208865.

When `read-only` variable was set to `ON`, *Percona XtraDB Cluster* would block `SELECT` statements as well. Bug fixed #1091099.

In geo-DR setup using `garbd`, performance would degrade with node count when cross data center link was down. Bug fixed #1182960.

`wsrep_recover` was being run even if it wasn't used. Script now checks if `grastate.dat` file has non-zero `uuid` and -1 `seqno` before it decides to start with *`wsrep_recover`* option. Bug fixed #1193781.

`PID` detection in the `init` script wasn't working correctly if the `PID` file was specified with the relative path. Bug fixed #1194998.

State Snapshot Transfer authentication password was displayed in the `ps` output. Bug fixed #1200727.

Fixed the packaging issue caused by shared-compat linking. Bug fixed #1201393.

Fixed the platform dependent code in wsrep which was required to make the code portable to `MacOS X` and `FreeBSD`. Bug fixed #1201893.

Percona XtraDB Cluster donor node would get stuck during the *State Snapshot Transfer* when the thread-pool plugin was used. Bug fixed #1206565.

`pyclustercheck` script did not work correctly with HAProxy. Bug fixed #1210239.

`pyclustercheck` script didn't work as expected when available node was a donor. Bug fixed #1211249.

New bootstrap method `bootstrap-pxc` which was introduced in *`5.5.31-23.7.5`* didn't check if mysqld process was already running before starting the new process. Bug fixed #1211505.

When table was created with `LIKE` and the source table was temporary table, create statement would be replicated to the slave nodes where it couldn't be applied because the temporary table wasn't present on the slave nodes. This would cause other nodes to crash if there were later `DML` commands for this table. Bug fixed #1212247.

Non-unique indexes in a parent table (as referenced by some foreign key constraint), would be included in write set population. i.e. key values in *all* non-unique indexes will be appended in the write set's key set. This could cause excessive multi-master conflicts, especially if the parent table has non-unique indexes with low selectivity. Bug fixed #1216805.

Added information to *InnoDB* status if the transaction is waiting on `TOI`. Bug fixed #1219856.

Binary build was linked against `libssl.so.10` and `libcrypto.so.10` which was making it hard to run on `SUSE Linux Enterprise Server 11`. Bug fixed #1213855.

`GU_AVPHYS_SIZE` would report more available memory than could be addressed on 32-bit systems. Bug fixed #1204241.

Other bug fixes: bug fixed #1210638, bug fixed #1222777, bug fixed #1216904, bug fixed #1205467, bug fixed #1196898, bug fixed #1195355, bug fixed #1049599, bug fixed #1191395, bug fixed #1017526, bug fixed #1213073, bug fixed #1171759, bug fixed #1210618, bug fixed #1190756.

Known Issues

For Debian/Ubuntu users: *Percona XtraDB Cluster* 5.5.33-23.7.6 includes a new dependency, the `socat` package. If the `socat` is not previously installed, `percona-xtradb-cluster-server-5.5` may be held back. In order to upgrade, you need to either install `socat` before running the `apt-get upgrade` or by writing the following command: `apt-get install percona-xtradb-cluster-server-5.5`. For *Ubuntu* users the `socat` package is in the universe repository, so the repository will have to be enabled in order to install the package.

Based on Percona Server 5.5.33-31.1 including all the bug fixes in it, Galera Replicator and on Codership wsrep API 5.5.33-23.7.6, *Percona XtraDB Cluster* 5.5.33-23.7.6 is now the current stable release. All of *Percona*'s software is open-source and free.

Percona XtraDB Cluster Errata can be found in our documentation.

6.1.8 *Percona XtraDB Cluster* 5.5.31-23.7.5

Percona is glad to announce the release of *Percona XtraDB Cluster* on June 24th, 2013. Binaries are available from downloads area or from our *software repositories*.

This is an General Availability release. 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.

New Features

Information about the wsrep sequence number has been added to `INFORMATION_SCHEMA INNODB_TRX` table.

Percona XtraDB Cluster can now be *bootstrapped* with the new `bootstrap-pxc` option in the init script.

Percona XtraDB Cluster has implemented parallel copying for rsync *State Snapshot Transfer* method. Bug fixed #1167331 (*Mrten*).

Percona XtraDB Cluster has implemented new *Python* version of the clustercheck script.

Percona XtraDB Cluster now has better integration with *Percona XtraBackup* 2.1 by implementing new `xbstream` option for xtrabackup *State Snapshot Transfer* method.

Bugs fixed

Fixed the packaging issues that caused *Percona XtraDB Cluster* to conflict with *MySQL* libraries when performing fresh *CentOS* installation. Bug fixed #1183669.

The RPM installer script had the *datadir* hardcoded to `/var/lib/mysql` instead of using `my_print_defaults` to get the datadir info. bug fixed #1172621.

Fixed the package conflict for `percona-xtradb-cluster-client-5.5` package. Bug fixed #1172621.

Fixed the `Percona-Server-shared-55` and `Percona-XtraDB-Cluster-shared` `RPM` package dependences. Bug fixed #1050654.

`mysql_tzinfo_to_sql` script failed with *Percona XtraDB Cluster* since *MyISAM* DDL/DML were not consistently replicated. Fixed by enabling session variable *`wsrep_replicate_myisam`* in the generator file. Bug fixed #1161432.

Startup script on *Debian* and *Ubuntu* was failing prematurely during `SST`. Bug fixed #1099428.

Due to *Debian* way of stopping/reloading/status checking, there was failure after SST since the machine-specific *Debian* maintenance password didn't work. Fixes by using signals instead of `mysqladmin` as it is done in *CentOS*. Bug fixed #1098360.

`wsrep-recover` would create `ibdata1` and `ib_logfile*` files which the SST after that would remove. Bug fixed #1191767.

When running *`wsrep_recover`* to recover galera co-ordinates, LRU recover (and its blocking counterpart) were causing issues. Bug fixed #1108035.

Percona XtraDB Cluster improved status visibility in the `processlist` on other nodes when provider is locked with `FLUSH TABLES WITH READ LOCK` on one of them. Bug fixed #1109341.

Crash/stalling bugs

Percona Server Crash-Resistant Replication was causing conflicts with `wsrep_crash_recovery` on a node which is a slave to an async master in standard *MySQL* replication. Bugs fixed #1182441 and #1180802.

Percona XtraDB Cluster node would hang on shutdown when `thread_handling` was set to pool-of-threads. Bug Fixed #1185523.

`FLUSH TABLES WITH READ LOCK` didn't behave as expected, it had different behavior than the upstream *MySQL* version. Bug fixed #1170821.

When rsync *State Snapshot Transfer* failed, the rsync daemon would remain running which caused `pid` errors on subsequent `SST` retries. Bug fixed #1169676.

When doing cascading Foreign Key processing *Percona XtraDB Cluster* was doing unnecessary Foreign Key checks, which are needed only for the purpose of getting Foreign Key parent reference for child table modifications. Bug fixed #1134892.

High priority threads (slave threads, replaying threads, total order isolation threads) were not protected from kill signal and attempt to kill high priority thread could cause server to hang. Bug fixed #1088267.

Galera provider may deadlock if applier thread is still executing `apply_trx()` while processing commit, it could cause corresponding transaction to be purged from cert index.

Regressions

Previous *Percona XtraDB Cluster* release introduced the regression in the RPM packaging that prevented the server from restarting following an upgrade. Bug fixed #1168032.

Other bug fixes: bug fixed #1187526, bug fixed #1133266, bug fixed #1079688, bug fixed #1177211, bug fixed #1166423, bug fixed #1084573, bug fixed #1069829, bug fixed #1012138, bug fixed #1170066, bug fixed #1170706, bug fixed #1182187, bug fixed #1183530, bug fixed #1183997, bug fixed #1184034, bug fixed #1191763, bug fixed #1069829.

Based on Percona Server 5.5.31-30.3 including all the bug fixes in it and on Codership wsrep API 5.5.31-23.7.5, *Percona XtraDB Cluster* 5.5.31-23.7.5 is now the current stable release. All of *Percona*'s software is open-source and free.

6.1.9 *Percona XtraDB Cluster* 5.5.30-23.7.4

Percona is glad to announce the release of *Percona XtraDB Cluster* on April 17th, 2013. Binaries are available from downloads area or from our *software repositories*.

This is an General Availability release. 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.

New Features

Percona XtraDB Cluster has implemented initial implementation of weighted quorum. Weight for node can be assigned via *`pc.weight`* option in the *`wsrep_provider_options`* variable. Accepted values are in the range [0, 255] (inclusive). Quorum is computed using weighted sum over group members.

Percona XtraDB Cluster binary will now be shipped with the `libjemalloc` library. For `rpm/deb` packages, this library will be available for download from our repositories. Benchmark showing the impact of memory allocators on *MySQL* performance can be found in this blogpost.

This release of *Percona XtraDB Cluster* has fixed number of foreign key and packaging bugs.

Bugs fixed

Fixed yum dependencies that were causing conflicts in `CentOS` 6.3 during installation. Bug fixed #1031427 (*Ignacio Nin*).

In case the *Percona XtraDB Cluster* was built from the source rpm, wsrep revision information would be missing. Bug fixed #1128906 (*Alexey Bychko*).

The method of generating md5 digest over tuples in a table with no primary key was not deterministic and this could cause a node failure. Bug fixed #1019473 (*Seppo Jaakola*).

Percona XtraDB Cluster was built with YaSSL which could cause some of the programs that use it to crash. Fixed by building packages with OpenSSL support rather than the bundled YaSSL library. Bug fixed #1104977 (*Raghavendra D Prabhu*).

Clustercheck script would hang in case the *MySQL* server on a node is hung. As a consequence clustercheck script would never fail-over that server. Bug fixed #1035927 (*Raghavendra D Prabhu*).

High values in variables `evs.send_window` and `evs.user_send_window` could trigger cluster crash under high load. Bug fixed #1080539 (*Teemu Ollakka*).

Standard *MySQL* port would be used when port number isn't explicitly defined in the `wsrep_node_incoming_address`. Bug fixed #1082406 (*Alex Yurchenko*).

Dropping a non-existing temporary table would be replicated when TOI was used in `wsrep_OSU_method` variable. This bug was fixed for the case when `DROP TEMPORARY TABLE` statement was used, but it will still replicate in case `DROP TABLE` statement is used on a temporary table. Bug fixed #1084702 (*Seppo Jaakola*).

In case two nodes in a 3-node cluster had to abort due to inconsistency, one wouldn't correctly notify the surviving node which would lead to surviving node to loose the primary component and cause subsequent downtime. Bug fixed #1108165 (*Alex Yurchenko*).

In some cases non-uniform foreign key reference could cause a slave crash. Fixed by using primary key of the child table when appending exclusive key for cascading delete operation. Bug fixed #1089490 (*Seppo Jaakola*).

Parallel applying would fail in case mixed `CHAR` and `VARCHAR` columns would be used in foreign key definitions. Bug fixed #1100496 (*Seppo Jaakola*).

Debian packages included the old version of **innotop**. Fixed by removing **innotop** and its `InnoDBParser` Perl package from source and *Debian* installation. Bug fixed #1032139 (*Alexey Bychko*).

The mysqld_safe script would fail to retrieve the galera replication position on `Ubuntu` 10.04, because the different shell was used. Bug fixed #1108431 (*Alex Yurchenko*).

Cascading foreign key constraint could lead to unresolved replication conflict and leave a slave hanging. Bug fixed #1130888 (*Seppo Jaakola*).

If *MySQL* replication threads were started before running wsrep recovery, this would lead to memory corruption and server crash. Bug fixed #1132974 (*Seppo Jaakola*).

Conflicting prepared statements in multi-master use case could cause node to hang. This was happening due to prepared statement execution loop, which does not honor wsrep status codes correctly. Bug fixed #1144911 (*Seppo Jaakola*).

State Snapshot Transfer with *XtraBackup* would fail if the `tmpdir` was specified more than once in the *MySQL* configuration file (`my.cnf`). Bugs fixed #1160047 and #1086978 (*Raghavendra D Prabhu*).

Issues with compiling Galera on the `ARM` architecture has been fixed. Bug fixed #1133047 (*Alex Yurchenko*).

Upstream bugfix for bug #59354 triggered a regression that could cause transaction conflicts. Bug fixed #1158221 (*Seppo Jaakola*).

Galera builds would fail when they were built with the new `boost` library. Bug fixed #1131736 (*Alex Yurchenko*).

Folder `lost+found` wasn't included in the rsync SST filter, which caused the SST failure due to insufficient privileges. Fixed by excluding `lost+found` folder if found. Bug fixed #1154095 (*Alex Yurchenko*).

If variable `innodb_thread_concurrency` has been defined to throttle *InnoDB* access, and work load contained DDL statements, a cluster node could remain hanging for unresolved MDL conflict. Fixed by adding a new method to cancel a thread waiting for *InnoDB* concurrency. Bug fixed #1155183 (*Seppo Jaakola*).

Handling of the network issues in Galera has been improved. Bug fixed #1153727 (*Teemu Ollakka*).

Fixed the wrong path in the `/etc/xinetd.d/mysqlchk` script. Bugs fixed #1000761 and #1132934 (*Raghavendra D Prabhu*).

When upgrading the Percona-XtraDB-Cluster-server package, `/usr/bin/clustercheck` would get overwritten, and any changes (such as username and password) would be lost. Bug fixed #1158443 (*Raghavendra D Prabhu*).

In case `CREATE TABLE AS SELECT` statement was running in parallel with the DDL statement on the selected table, in some cases first statement could be left hanging. Bug fixed #1164893 (*Seppo Jaakola*).

Galera builds would fail when `gcc` 4.8 was used. Bug fixed #1164992 (*Alex Yurchenko*).

`Percona-XtraDB-Cluster-galera` package version number didn't match the *`wsrep_provider_version`* one. Bug fixed #1111672 (*Alexey Bychko*).

Only `rpm` debug build was available for *Percona XtraDB Cluster*, fixed by providing the `deb` debug build as well. Bug fixed #1096123 (*Ignacio Nin*).

Other bug fixes: bug fixed #1162421 (*Seppo Jaakola*), bug fixed #1093054 (*Alex Yurchenko*), bug fixed #1166060 (*Teemu Ollakka*), bug fixed #1166065 (*Teemu Ollakka*).

Based on Percona Server 5.5.30-30.2 including all the bug fixes in it and on Codership wsrep API 5.5.30-23.7.4, *Percona XtraDB Cluster* 5.5.30-23.7.4 is now the current stable release. All of *Percona*'s software is open-source and free.

6.1.10 *Percona XtraDB Cluster* 5.5.29-23.7.2

Percona is glad to announce the release of *Percona XtraDB Cluster* on February 12th, 2013. Binaries are available from downloads area or from our software repositories.

This is an General Availability release. 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.

Bugs fixed

`DML` operations on temporary tables would try to append the key for provider library, which could cause a memory leak. Bug fixed for *MyISAM* temporary tables, *InnoDB* temporary tables still have this issue, probably caused by upstream bug #67259. Bug fixed #1112514 (*Seppo Jaakola*).

Bug fix for bug #1078346 introduced a regression. Foreign key checks were skipped in the parent table, which would cause the foreign key constraint errors. Bug fixed #1117175 (*Seppo Jaakola*).

Based on Percona Server 5.5.29-29.4 including all the bug fixes in it, *Percona XtraDB Cluster* 5.5.29-23.7.2 is now the current stable release. All of *Percona*'s software is open-source and free.

6.1.11 *Percona XtraDB Cluster* 5.5.29-23.7.1

Percona is glad to announce the release of *Percona XtraDB Cluster* on January 30th, 2013. Binaries are available from downloads area or from our software repositories.

This is an General Availability release. 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.

Bugs fixed

In some cases when node is recovered variable `threads_running` would become huge. Bug fixed #1040108 (*Teemu Ollakka*).

Variable `wsrep_defaults_file` would be set up to the value in the last configuration file read. Bug fixed by keeping the value found in the top configuration file. Bug fixed #1079892 (*Alex Yurchenko*).

Variable *`wsrep_node_name`* was initialized before the `glob_hostname`, which lead to empty value for *`wsrep_node_name`* if it wasn't set explicitly. Bug fixed #1081389 (*Alex Yurchenko*).

Running `FLUSH TABLES WITH READ LOCK` when slave applier needed to abort transaction that is on the way caused the deadlock situation. Resolved by grabbing Global Read Lock before pausing wsrep provider. Bug fixed #1083162 (*Teemu Ollakka*).

Percona XtraDB Cluster would crash when processing a delete for a table with foreign key constraint. Bug fixed #1078346 (*Seppo Jaakola*).

When variable `innodb_support_xa` was set to `0`, wsrep position wasn't stored into the InnoDB tablespace. Bug fixed #1084199 (*Teemu Ollakka*).

Using *XtraBackup* for State Snapshot Transfer would fail due to `mktemp` error. Bug fixed #1080829 (*Alex Yurchenko*).

XtraBackup donor would run *XtraBackup* indefinitely if the xtrabackup `--tmpdir` was on `tmpfs`. Bug fixed #1086978 (*Alex Yurchenko*).

In some cases non-uniform foreign key reference could cause a slave crash. Fixed by using primary key of the child table when appending exclusive key for cascading delete operation. Bug fixed #1089490 (*Seppo Jaakola*).

Percona XtraDB Cluster would crash when `binlog_format` was set to `STATEMENT`. This was fixed by introducing the warning message. Bug fixed #1088400 (*Seppo Jaakola*).

An explicitly set *`wsrep_node_incoming_address`* may make "`SHOW STATUS LIKE 'wsrep_incoming_addresses';`" return the address without the port number. Bug fixed #1082406 (*Alex Yurchenko*).

Percona XtraDB Cluster would crash if the node's own address would be specified in the *`wsrep_cluster_address`* variable. Bug fixed #1099413. (*Alexey Yurchenko*)

When installing from yum repository, `Percona-XtraDB-Cluster-server` and `Percona-XtraDB-Cluster-client` would conflict with `mysql` and `mysql-server` packages. Bug fixed #1087506 (*Ignacio Nin*).

Other bug fixes: bug fixed #1037165, bug fixed #812059.

Based on Percona Server 5.5.29-29.4 including all the bug fixes in it, *Percona XtraDB Cluster* 5.5.29-23.7.1 is now the current stable release. All of *Percona*'s software is open-source and free.

6.1.12 *Percona XtraDB Cluster* 5.5.28-23.7

Percona is glad to announce the release of *Percona XtraDB Cluster* on November 15th, 2012. Binaries are available from downloads area or from our software repositories.

This is an General Availability release. 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.

Features

- *Percona XtraDB Cluster* has ported Twitter's *MySQL* `NUMA` patch. This patch implements improved NUMA support as it prevents imbalanced memory allocation across NUMA nodes.
- Number of binlog files can be restricted when using *Percona XtraDB Cluster* with the new max_binlog_files option.
- New status variable *`wsrep_incoming_addresses`* has been introduced. It contains a comma-separated list of incoming (client) addresses in the cluster component.
- Multiple addresses can now be specified in the `gcomm://  URL` when defining the address to connect to cluster. Option `wsrep_urls` is now deprecated, *`wsrep_cluster_address`* should be used to specify multiple addresses instead.
- GTID can now be recovered by starting mysqld with `--wsrep-recover` option. This will parse GTID from log, and if the GTID is found, assign it as initial position for actual server start.
- `SET PASSWORD` command replication has been implemented. The SET statement can contain other variables and several password changes, and each password change will be replicated separately as one TOI (total order isolation) call. The user does not have to be explicitly defined, current user will be detected and correct `SET PASSWORD` statement will be generated for TOI replication in every use case.
- SQL statements aborted due to multi-master conflict can be automatically retried. This can be controlled with *`wsrep_retry_autocommit`* variable. The retrying will happen only if there is real abort for the SQL statement. If statement itself enters into commit phase and detects certification failure, there will be no retry.
- This version of *Percona XtraDB Cluster* implements significant speedups and memory footprint reduction on ranged queries.

Bugs fixed

- Some wsrep variables, like *`wsrep_provider`*, *`wsrep_provider_options`* and *`wsrep_cluster_address`*, could be "doubly" allocated which could cause memory leak. Bug fixed #1072839.
- In case variables *`wsrep_node_address`* and *`wsrep_node_incoming_address`* were not set explicitly, port number would be set to `0`. Bug fixed #1071882.
- Server would hang before storage engine initialization with `rsync` SST and with option *`wsrep_start_position`* enabled. Bug fixed #1075495.
- SST script interface has been upgraded to use named instead of positional parameters. Bug fixed #1045026.

- When server was started with no wsrep_provider and regular *MySQL* replication slave was configured, could cause intermittent failure to start MySQL slave during startup. Bug fixed #1075617.
- If DDL is processing with RSU method and there is MDL conflict with local transaction, this conflict would remain unresolved. Bug fixed #1039514.
- Static SST method list has been removed. Bug fixed #1045040.
- DDL in multi-statement would cause a server crash. Bug fixed #1049024.
- Mysqld_safe doesn't restart the node automatically anymore. Bug fixed #1077632.
- Server would fail to start if the variable `wsrep_provider` was unset. Bug fixed #1077652.

Other bug fixes: bug #1049103, bug #1045811, bug #1057910, bug #1052668, bug #1073220, bug #1076382.

Based on Percona Server 5.5.28-29.1 including all the bug fixes in it, *Percona XtraDB Cluster* 5.5.28-23.7 is now the current stable release. All of *Percona*'s software is open-source and free.

6.1.13 *Percona XtraDB Cluster* 5.5.27-23.6

Percona is glad to announce the release of *Percona XtraDB Cluster* on September 7th, 2012. Binaries are available from downloads area or from our software repositories.

This is an General Availability release. 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.

Features

- *Percona XtraDB Cluster* supports tunable buffer size for fast index creation in *InnoDB*. This value was calculated based on the merge block size (which was hardcoded to 1 MB) and the minimum index record size. By adding the session variable innodb_merge_sort_block_size block size that is used in the merge sort can now be adjusted for better performance.
- *Percona XtraDB Cluster* has implemented ability to have a MySQL Utility user who has system access to do administrative tasks but limited access to user schema. This feature is especially useful to those operating MySQL As A Service.
- New Expanded Program Option Modifiers have been added to allow access control to system variables.
- New table INNODB_UNDO_LOGS has been added to allow access to undo segment information. Each row represents an individual undo segment and contains information about which rollback segment the undo segment is currently owned by, which transaction is currently using the undo segment, and other size and type information for the undo segment. This information is 'live' and calculated for each query of the table.
- *Percona XtraDB Cluster* now has improved `clustercheck.sh` script which makes a proxy (ie HAProxy) capable of monitoring Percona XtraDB Cluster nodes properly.

Bugs fixed

- Increasing the number of slave threads doesn't require provider reloading anymore. Bug fixed #852094.
- No other SQL statement except INSERT...SELECT would replicate if */!99997/* comment was present. Bug fixed #1023911.
- Fixed the bug that caused DDL command hanging when RSU was enabled and server was under high rate SQL write load. Bug fixed #1026181.
- Cascading foreign key constraint could cause a slave crash. Bug fixed #1013978.

- Fixed bug when donor accepts query, which then blocks. Bug fixed #1002714.

Based on Percona Server 5.5.27-28.1 including all the bug fixes in it, and the latest Galera 2.x replicator, *Percona XtraDB Cluster* 5.5.27-23.6 is now the current stable release. All of *Percona*'s software is open-source and free.

6.1.14 *Percona XtraDB Cluster* 5.5.24-23.6

Percona is glad to announce the release of *Percona XtraDB Cluster* on June 7, 2012. Binaries are available from downloads area or from our software repositories.

This is an General Availability release. We did our best to eliminate bugs and problems during alpha and beta testing release, but this is a software, so bugs are expected. If you encounter them, please report them to our bug tracking system.

Bugs fixed:

- Table with no PK but still having UK, can cause crash in slave using PA. Bug fixed #1003929.

Based on Percona Server 5.5.24-26.0 including all the bug fixes in it, and the latest Galera 2.x replicator, *Percona XtraDB Cluster* 5.5.24-23.6 is now the current stable release. All of *Percona*'s software is open-source and free.

6.1.15 *Percona XtraDB Cluster* 5.5.23-23.5

Percona is glad to announce the release of *Percona XtraDB Cluster* on May 14, 2012. Binaries are available from downloads area or from our software repositories.

This is an General Availability release. We did our best to eliminate bugs and problems during alpha and beta testing release, but this is a software, so bugs are expected. If you encounter them, please report them to our bug tracking system.

List of changes:

- Fixes merged from upstream (Codership-mysql).
- Support for MyISAM, now changes to MyISAM tables are replicated to other nodes.
- Improvements to XtraBackup SST methods, better error handling.
- New SST wsrep_sst_method=skip, useful when you start all nodes from the same sources (i.e. backup).
- Ability to pass list of IP addresses for a new node, it will connect to the first available.

Based on Percona Server 5.5.23-25.3 including all the bug fixes in it, and the latest Galera 2.x replicator, *Percona XtraDB Cluster* 5.5.23-25.3 is now the current stable release. All of *Percona*'s software is open-source and free.

6.2 Index of wsrep status variables

variable `wsrep_local_state_uuid`

This variable contains *UUID* state stored on the node.

variable `wsrep_protocol_version`

Version of the wsrep protocol used.

variable `wsrep_last_committed`

Sequence number of the last committed transaction.

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.

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_local_commits`

Number of writesets commited on the node.

variable `wsrep_local_cert_failures`

Number of writesets that failed the certification test.

variable `wsrep_local_bf_aborts`

Number of local transactions that were aborted by slave transactions while being executed.

variable `wsrep_local_replays`

Number of transaction replays due to "asymmetric lock granularity".

variable `wsrep_local_send_queue`

Current length of the send queue. Show 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_recv_queue`

Current length of the receive queue. Show 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're received. This could be sign that node is overloaded and it will cause the replication throttling.

variable `wsrep_flow_control_paused`

Time since the last status query that replication was paused due to flow control.

variable `wsrep_flow_control_sent`

Number of `FC_PAUSE` events sent since the last status query.

variable `wsrep_flow_control_recv`

Number of `FC_PAUSE` events sent and received since the last status query.

variable `wsrep_cert_deps_distance`

Average distance between highest and lowest sequence numbet that can be possibly applied in parallel.

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 was writeset with higher sequence number applied before the one with lower sequence number.

variable wsrep_apply_window

Average distance between highest and lowest concurrently applied sequence number.

variable wsrep_commit_oooe

This variable shows how often a transaction has been applied out of order.

variable wsrep_commit_oool

This variable currently isn't being used.

variable wsrep_commit_window

Average distance between highest and lowest concurrently committed sequence number.

variable wsrep_local_state

This variable shows internal Galera state number. Possible values are:

- 1 - Joining (requesting/receiving State Transfer) - 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_comment

Description of the *wsrep_local_state* variable.

variable wsrep_cert_index_size

variable wsrep_causal_reads_

Shows the number of writesets processed while the variable *wsrep_causal_reads* was set to `ON`.

variable wsrep_incoming_addresses

Shows the comma-separated list of incoming node addresses in the cluster.

variable wsrep_cluster_conf_id

Number of cluster membership changes happened.

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_connected

variable wsrep_local_index

Node index in the cluster

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 the queries will fail with `ERROR 1047 (08S01) Unknown Command` error (unless *`wsrep_on`* variable is set to `0`)

6.3 Index of wsrep system variables

variable `wsrep_OSU_method`

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value TOI

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

- TOI - Total Order Isolation - When this method is selected `DDL` is 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` is processed under total order isolation (TOI) the `DDL` statement will be replicated up front to the cluster. i.e. cluster will assign global transaction ID for the `DDL` statement before the `DDL` processing begins. Then every node in the cluster has the responsibility to execute the `DDL` in the given slot in the sequence of incoming transactions, and this `DDL` execution has to happen with high priority.
- RSU - Rolling Schema Upgrade - When this 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 the `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 the replication breaks the SST will be triggered when the node tries to join again but the change will be undone.

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

- ***5.5.39-25.11*** – Variable deprecated by *wsrep_sync_wait*

Command Line Yes

Config File Yes

Scope Global, Local

Dynamic Yes

Default Value OFF

In some cases master may apply event faster than a slave, which can cause master and slave being out-of-sync for a brief moment. When this variable is set to `ON` slave will wait till 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 the rows that the rows 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 minimally needs to be any single other cluster 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. This takes the form of:

```
gcomm://<node:ip>,<node:ip>,<node:ip>
```

If an empty `gcomm://` is provided, this tells the node to bootstrap it self (i.e., form a new cluster). This is not recommended for production after the cluster has been bootstrapped initially.

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.

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 the problem 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 continue working ignoring the replication lag on the node being in desync mode. Toggling this back will require a IST or a 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_drupal_282555_workaround

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value OFF

This variable was introduced as workaround for Drupal/MySQL bug #282555. In some cases duplicate key error would occur when inserting the default value in 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 be always be effective regardless of session binlog format setting. Supported optio

- ROW
- STATEMENT
- MIXED
- NONE - This option resets the forced state of the binlog format

variable wsrep_load_data_splitting

Command Line Yes

Config File Yes

Scope Global

Dynamic Yes

Default Value ON

This variable controls whether LOAD DATA transaction splitting is wanted or not.

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 this 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 this 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. This 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 makes SST and IST work correctly out of the box. Even in the 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 is 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

This variable is used to set up the unique node 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 server will execute every time cluster membership or local node status changes.

variable `wsrep_on`

Command Line No

Config File No

Scope Local, Global

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_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 Yes

Config File No

Scope Global

Dynamic No

Default Value OFF

When server is started with this variable (as `--wsrep-recover`) it will parse Global Transaction ID from log, and if the GTID is found, output to stderr (which usually goes into the log). This option is used to recover GTID, mysqld is called with this automatically in mysqld_safe, hence running this manually is not required, also no need to set it in my.cnf.

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 that 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. Following values are supported:

- `NONE` - default - nothing is rejected.
- `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, that this doesn't affect galera replication in any way, only the applications which connect to database are affected. If you are looking for desyncing a node then *`wsrep_desync`* is the right option for that.

variable `wsrep_replicate_myisam`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value Off

This variable controls if *MyISAM* will be replicated or not. *MyISAM* replication is still experimental and that is one of the reasons why this variable is set to `OFF` by default. From version *5.5.41-25.11* *MyISAM* DDL (`CREATE TABLE` only) isn't replicated 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). This also doesn't work if `default_storage_engine` is set to `MyISAM` (which is not recommended for *Percona XtraDB Cluster*) and a table is created without the `ENGINE` option.

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 with the hopes that it will pass the next time. This can be useful to help an application using autocommit to avoid the 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

Version Info

- ***5.5.39-25.11*** – Variable introduced

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_UK_checks

Version Info

- ***5.5.39-25.11*** – Variable introduced

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_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 anytime, note that, 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 many a time 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_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. 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 the SST. If no node is selected as a preferred donor it will be chosen from one of the available nodes automatically.

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 rsync

Recommended xtrabackup-v2

This variable sets up the method for taking the State Snapshot Transfer (SST). Available options 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 same as xtrabackup SST 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 (as of 5.5.34)*.
- 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.
- custom_script_name - Galera supports Scriptable State Snapshot Transfer. This enables users to create their own custom script for performing an SST.
- skip - this option can be used to skip the SST, it can be used when initially starting the cluster and manually restore 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:

Note the following:

- mysqldump SST is not recommended unless it is required for specific reasons. Also, it is not compatible with `bind_address = 127.0.0.1 or localhost` and will cause startup to fail if set so.
- Xtrabackup-v2 SST is currently recommended if you have innodb-log-group_home-dir/innodb-data-home-dir in your cnf. Refer to `sst-special-dirs` for more.

---

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 the 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

- ***`5.5.39-25.11`*** – 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 deprecates 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

Following variables can be set and checked in the *`wsrep_provider_options`* variable. 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 of the in the `my.cnf` following syntax should be used:

```
wsrep_provider_options="variable1=value1;[variable2=value2]"
```

For example to increase the size of the Galera buffer storage from the default value to 512MB, `my.cnf` option should look like:

```
wsrep_provider_options="gcache.size=512M"
```

Dynamic variables can be changed from the *MySQL* client by using the `SET GLOBAL` syntax. To change the value of the *`pc.ignore_sb`* following command should be used:

```
mysql> SET GLOBAL wsrep_provider_options="pc.ignore_sb=true";
```

6.4.1 Index

variable `base_host`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value value of the *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 the 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.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.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 pronounced 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 PT15S

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 will keepalive beacons will be emmited (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 may be set considerably higher, e.g. 512. This variable must be no 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 may be set considerably higher, e.g. 512.

variable evs.version

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 0

This status variable is used to check which EVS (Extended Virtual Synchrony) protocol version is used.

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_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 page storage pages to keep for caching purposes. If only page storage is enabled, one page is always present.

variable `gcache.mem_size`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value 0

This variable is used to define how much RAM system has available. **NOTE:** This parameter is for use on systems with spare memory. You should not use it otherwise, as it may lead to unexpected results.

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*. If this value is bigger there are better 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 1.0

This variable is used for replication flow control. Replication will be paused till the value of this variable goes below the value of *`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. When slave queue exceeds this limit replication will be paused.

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 half of (RAM + swap). 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` 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 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.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 status variable is used to check 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 nodes 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 the other nodes are discovered.

variable `pc.checksum`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value false

This variable controls will the replicated messages will 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` node will completely ignore the 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 us set ti `TRUE` node will process updates even in the case of 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 which 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 component overrides older ones in case of conflicting prims.

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 being used.

variable `pc.wait_prim`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value true

When this variable is set to `true` node will wait for *`pc.wait_prim_timeout`* value for the primary component. This option can be used to bring up a non-primary component and make it primary with `pc.bootstrap` option.

variable `pc.wait_prim_timeout`

Command Line Yes

Config File Yes

Scope Global

Dynamic No

Default Value P30S

This variable is used to define how much time node should wait for the 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 show 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). Allowed values are:

- `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)

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-log created 120715  9:45:56 at startup
# 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     error_code=0
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, @@session.autocommit=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_server=8/*!*/;
SET @@session.lc_time_names=0/*!*/;
```

```
SET @@session.collation_database=DEFAULT/*!*/;
drop table test
/*!*/;
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
```

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: 'test'. Query: 'drop table test', Error_code: 1051
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: dd40ad88-7ff9-11e2-0800-e93cbffe93d7 version: 2 local: 0 state:␣
↪APPLYING flags: 65 conn_id: 3 trx_id: -1 seqnos (l: 5, g: 3, s: 2, d: 2,␣
↪ts: 1361875730070283555)
```

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:
```

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
↪'*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19';
```

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. If there is a loss of quorum and no primary component, you can make a node PRIMARY by:

```
SET GLOBAL wsrep_provider_options="pc.bootstrap=1";
```

However it is also possible to make a node immune to quorum issues with:

```
SET GLOBAL 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).

6.6.15 Q: How do I use Galera-3 with PXC 5.5?

A: For ubuntu/debian, you can just do:

```
apt-get install percona-xtradb-cluster-galera-3.x
```

For centos, you will need to install yum replace plugin. After this, doing following will work:

```
yum replace Percona-XtraDB-Cluster-galera-3 --replace-with Percona-XtraDB-Cluster-
↪galera-2
```

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

INDEX

Symbols

B

C

D

E

G

X