PERCONA

www.percona.com

Percona XtraDB Cluster Documentation

Release 5.5.20-23.4

Percona Inc

April 03, 2012

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

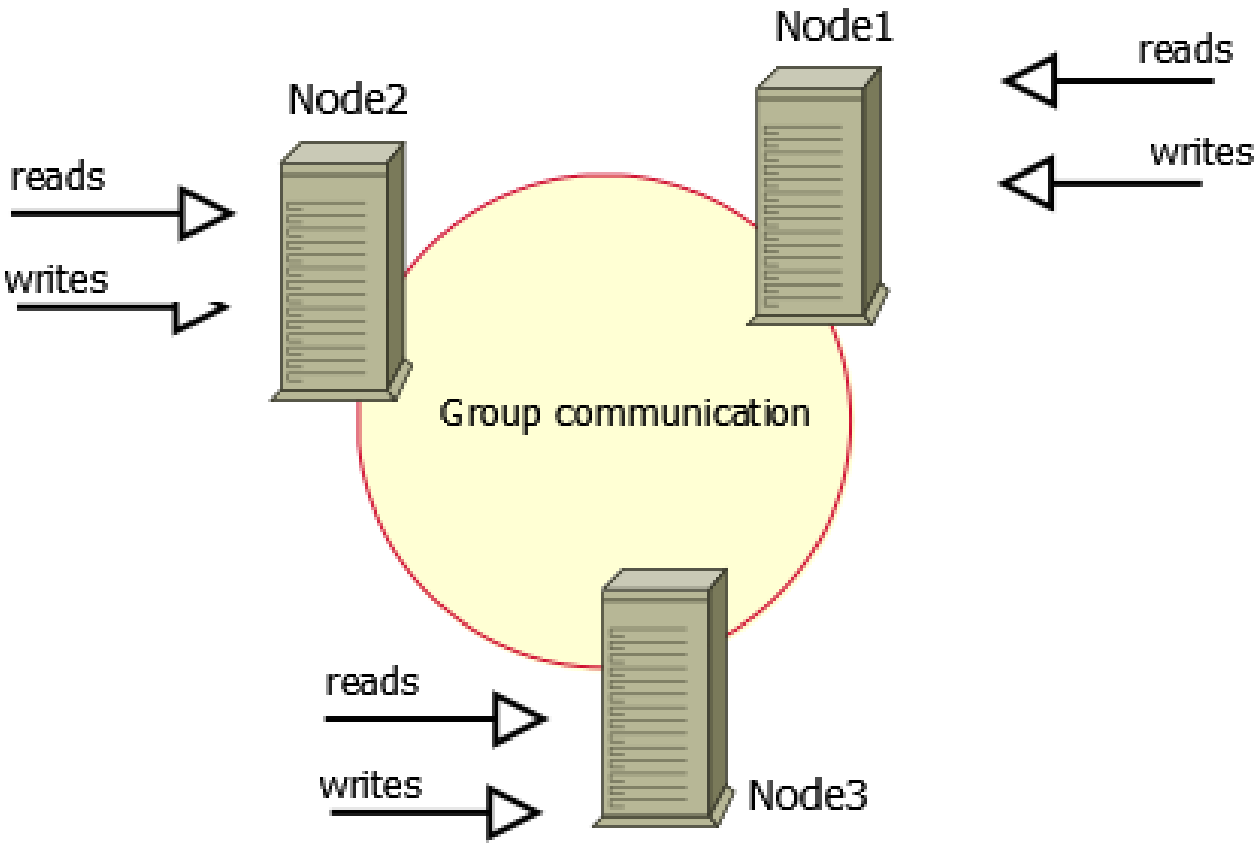
INTRODUCTION

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. That defines XtraDB Cluster behavior in many ways. And obviously there are benefits and drawbacks.



Benefits of such approach:

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

Drawbacks:

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

1.1.2 What is core difference Percona XtraDB Cluster from MySQL Replication ?

Let's take look into the well known CAP theorem for Distributed systems. Characteristics of Distributed systems:

C - Consistency (all your data is consistent on all nodes),

A - Availability (your system is AVAILABLE to handle requests in case of failure of one or several nodes),

P - Partitioning tolerance (in case of inter-node connection failure, each node is still available to handle requests).

CAP theorem says that each Distributed system can have only two out of these three.

MySQL replication has: Availability and Partitioning tolerance.

Percona XtraDB Cluster has: Consistency and Availability.

That is MySQL replication does not guarantee Consistency of your data, while Percona XtraDB Cluster provides data Consistency. (And yes, Percona XtraDB Cluster looses Partitioning tolerance property).

1.1.3 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 Resources

In general there are 4 resources that need to be different when you want to run several MySQL/Galera nodes on one host:

1. data directory
2. mysql client port and/or address
3. galera replication listen port and/or address
4. receive address for state snapshot transfer

and later incremental state transfer receive address will be added to the bunch. (I know, it is kinda a lot, but we don't see how it can be meaningfully reduced yet).

The first two are the usual mysql stuff.

You figured out the third. It is also possible to pass it via:

```
wsrep_provider_options="gmcast.listen_addr=tcp://127.0.0.1:5678"
```

as most other Galera options. This may save you some extra typing.

The fourth one is wsrep_sst_receive_address. This is the address at which the node will be listening for and receiving the state. Note that in galera cluster _joining_ nodes are waiting for connections from donors. It goes contrary to tradition and seems to confuse people time and again, but there are good reasons it was made like that.

If you use mysqldump SST it should be the same as this mysql client connection address plus you need to set wsrep_sst_auth variable to hold user:password pair. The user should be privileged enough to read system tables from donor and create system tables on this node. For simplicity that could be just the root user. Note that it also means that you need to properly set up the privileges on the new node before attempting to join the cluster.

If you use rsync or **xtrabackup** SST, wsrep_sst_auth is not necessary unless your SST script makes use of it. wsrep_sst_address can be anything local (it may even be the same on all nodes provided you'll be starting them one at a time).

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. You can also do the installation manually.

Automatic Installation

Execute the following command as a `root` user, replacing `x86_64` with `i386` if you are not running a 64-bit operating system:

```
$ rpm -Uhv http://www.percona.com/downloads/percona-release/percona-release-0.0-1.x86_64.rpm
```

The RPMs for the automatic installation are available at http://www.percona.com/downloads/percona-release/ and include source code.

You should see some output such as the following:

```
Retrieving http://www.percona.com/downloads/percona-release/percona-release-0.0-1.x86_64.rpm
Preparing...                ########################################### [100%]
   1:percona-release        ########################################### [100%]
```

The RPMs for the automatic installation are available at http://www.percona.com/downloads/percona-release/ and include source code.

Install XtraDB Cluster

Following command will install Cluster packages:

```
$ yum install Percona-XtraDB-Cluster-server Percona-XtraDB-Cluster-client xtrabackup
```

Percona apt Repository

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

```
$ gpg --keyserver  hkp://keys.gnupg.net --recv-keys 1C4CBDCDCD2EFD2A
... [some output removed] ...
gpg:               imported: 1

$ gpg -a --export CD2EFD2A | sudo apt-key add -
```

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

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

Remember to update the local cache:

```
$ apt-get update
```

Supported Architectures

- x86_64 (also known as amd64)
- x86

Supported Releases

Debian

- 6.0 squeeze

Ubuntu

- 10.04LTS lucid
- 11.04 natty
- 11.10 oneiric

Install XtraDB Cluster

Following command will install Cluster packages:

```
$ sudo apt-get install percona-xtradb-cluster-client-5.5 \
percona-xtradb-cluster-server-5.5 percona-xtrabackup
```

Percona *apt* Experimental repository

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

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

Percona provides repositories for **yum** (`RPM` packages for *Red Hat*, *CentOS*, *Amazon Linux AMI*, and *Fedora*) 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.

2.1.2 Initial configuration

In order to start using XtraDB Cluster, you need to configure my.cnf file. Following options are needed:

```
wsrep_provider -- a path to Galera library.
wsrep_cluster_address -- cluster connection URL.
binlog_format=ROW
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
innodb_locks_unsafe_for_binlog=1
```

Additional parameters to tune:

```
wsrep_slave_threads   # specifies amount of threads to apply events
wsrep_sst_method
```

Example:

```
wsrep_provider=/usr/lib64/libgalera_smm.so
wsrep_cluster_address=gcomm://10.11.12.206
wsrep_slave_threads=8
wsrep_sst_method=rsync
#wsrep_sst_method=xtrabackup - alternative way to do SST
wsrep_cluster_name=percona_test_cluster
binlog_format=ROW
default_storage_engine=InnoDB
innodb_autoinc_lock_mode=2
innodb_locks_unsafe_for_binlog=1
```

2.1.3 Install XtraBackup SST method

To use Percona XtraBackup for State Transfer method (copy snapshot of data between nodes) you can use the regular xtrabackup package with the script what supports Galera information. You can take *innobackupex* script from source code innobackupex.

To inform node to use xtrabackup you need to specify in my.cnf:

```
wsrep_sst_method=xtrabackup
```

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`.

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
```

In `RPM`-based distributions, you need to:

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

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.

Examples how to build RPM and DEB packages you can find 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** (Percona *XtraBackup* with support of XtraDB Cluster will be released soon, currently you need to use our source code repository). 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

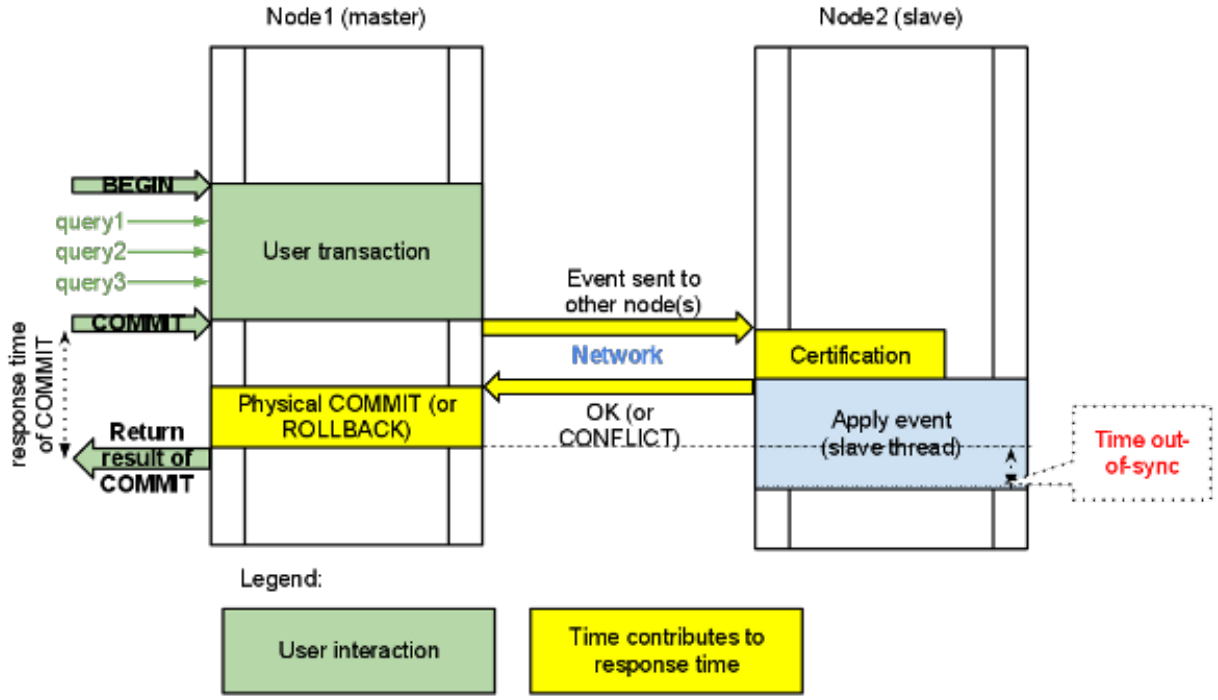
```
SHOW STATUS LIKE 'wsrep_local_state_comment';
```

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:



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

FAQ

4.1 Frequently Asked Questions

4.1.1 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.

4.1.2 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).

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

A: Your check should be simply:

```
SELECT * FROM someinnodbtable WHERE id=1;
```

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)

4.1.4 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.

4.1.5 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.

4.1.6 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.

4.1.7 Q: How would it handle split brain?

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

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

4.1.8 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. wsrep_start_position variable - start the nodes with the same *UUID:seqno* value and there you are.

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

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

4.1.10 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; "
   ```

4.1.11 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.

4.1.12 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.

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

A: Try to disable SELinux. Quick way is:

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

4.1.14 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 pushed in recent releases. Future releases of PXC will be made to be compatible with any netcat (bug #959970).

CHAPTER

FIVE

HOW-TO

5.1 How to setup 3 node cluster on single box

This is 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-5.5.17-22.1-3673.Linux.x86_64.rhel6

Now we need to create couple my.cnf files and couple of data directories.

Assume 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

and assume the local IP address is 10.11.12.205.

Then we should be able to start initial node as (from directory /usr/local/Percona-XtraDB-Cluster-5.5.17-22.1-3673.Linux.x86_64.rhel6):

```
bin/mysqld --defaults-file=/etc/my.4000.cnf
```

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

```
111215 19:01:49 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 0)
111215 19:01:49 [Note] WSREP: New cluster view: global state: 4c286ccc-2792-11e1-0800-94bd91e32efa:0,
```

And you can check used ports:

```
netstat -anp | grep mysqld
tcp        0      0 0.0.0.0:4000                0.0.0.0:*                   LISTEN      8218/mysqld
tcp        0      0 0.0.0.0:4010                0.0.0.0:*                   LISTEN      8218/mysqld
```

After first node, we start second and third:

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

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]
gdb

datadir=/data/bench/d1
basedir=/usr/local/Percona-XtraDB-Cluster-5.5.17-22.1-3673.Linux.x86_64.rhel6

port = 4000
socket=/tmp/mysql.4000.sock

user=root

binlog_format=ROW

wsrep_provider=/usr/local/Percona-XtraDB-Cluster-5.5.17-22.1-3673.Linux.x86_64.rhel6/lib/libgalera_sm

wsrep_cluster_address=gcomm://

wsrep_provider_options = "gmcast.listen_addr=tcp://0.0.0.0:4010; "
wsrep_sst_receive_address=10.11.12.205:4020

wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_sst_method=rsync
wsrep_node_name=node4000

innodb_locks_unsafe_for_binlog=1
innodb_autoinc_lock_mode=2
```

Configuration files (/etc/my.5000.cnf). PLEASE see the difference in *wsrep_cluster_address*:

```
/etc/my.5000.cnf
[mysqld]
gdb

datadir=/data/bench/d2
basedir=/usr/local/Percona-XtraDB-Cluster-5.5.17-22.1-3673.Linux.x86_64.rhel6

port = 5000
socket=/tmp/mysql.5000.sock

user=root

binlog_format=ROW

wsrep_provider=/usr/local/Percona-XtraDB-Cluster-5.5.17-22.1-3673.Linux.x86_64.rhel6/lib/libgalera_sm
```

```
wsrep_cluster_address=gcomm://10.11.12.205:4010

wsrep_provider_options = "gmcast.listen_addr=tcp://0.0.0.0:5010; "
wsrep_sst_receive_address=10.11.12.205:5020

wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_sst_method=rsync
wsrep_node_name=node5000

innodb_locks_unsafe_for_binlog=1
innodb_autoinc_lock_mode=2
```

Configuration files (/etc/my.6000.cnf). PLEASE see the difference in *wsrep_cluster_address*:

```
/etc/my.6000.cnf

[mysqld]
gdb

datadir=/data/bench/d3
basedir=/usr/local/Percona-XtraDB-Cluster-5.5.17-22.1-3673.Linux.x86_64.rhel6

port = 6000
socket=/tmp/mysql.6000.sock

user=root

binlog_format=ROW

wsrep_provider=/usr/local/Percona-XtraDB-Cluster-5.5.17-22.1-3673.Linux.x86_64.rhel6/lib/libgalera_sr

wsrep_cluster_address=gcomm://10.11.12.205:4010

wsrep_provider_options = "gmcast.listen_addr=tcp://0.0.0.0:6010; "
wsrep_sst_receive_address=10.11.12.205:6020

wsrep_slave_threads=2
wsrep_cluster_name=trimethylxanthine
wsrep_sst_method=rsync
wsrep_node_name=node6000

innodb_locks_unsafe_for_binlog=1
innodb_autoinc_lock_mode=2
```

5.2 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 Percona-XtraDB-Cluster-client
```

3. Create data directories:

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

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://

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

innodb_locks_unsafe_for_binlog=1
innodb_autoinc_lock_mode=2
```

On the second node:

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

binlog_format=ROW

wsrep_provider=/usr/lib64/libgalera_smm.so

wsrep_cluster_address=gcomm://10.93.46.58

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

innodb_locks_unsafe_for_binlog=1
innodb_autoinc_lock_mode=2
```

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

```
wsrep_node_name=node3
```

6. Start mysqld

On the first node:

```
/usr/sbin/mysqld
or
mysqld_safe
```

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

```
111216  0:16:42 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.5.17'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona XtraDB Cluster (GPL), Rel
111216  0:16:42 [Note] WSREP: Assign initial position for certification: 0, protocol version: 1
111216  0:16:42 [Note] WSREP: Synchronized with group, ready for connections
```

On the second (and following nodes):

```
/usr/sbin/mysqld
or
mysqld_safe
```

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

```
111216  0:21:39 [Note] WSREP: Flow-control interval: [12, 23]
111216  0:21:39 [Note] WSREP: Shifting OPEN -> PRIMARY (TO: 0)
111216  0:21:39 [Note] WSREP: New cluster view: global state: f912d2eb-27a2-11e1-0800-f34c520a3d4b:0,
111216  0:21:39 [Warning] WSREP: Gap in state sequence. Need state transfer.
111216  0:21:41 [Note] WSREP: Running: 'wsrep_sst_rsync 'joiner' '10.93.62.178' '' '/mnt/data/' '/etc
111216  0:21:41 [Note] WSREP: Prepared SST request: rsync|10.93.62.178:4444/rsync_sst
111216  0:21:41 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
111216  0:21:41 [Note] WSREP: Assign initial position for certification: 0, protocol version: 1
111216  0:21:41 [Note] WSREP: prepared IST receiver, listening in: tcp://10.93.62.178:4568
111216  0:21:41 [Note] WSREP: Node 1 (node2) requested state transfer from '*any*'. Selected 0 (node1
111216  0:21:41 [Note] WSREP: Shifting PRIMARY -> JOINER (TO: 0)
111216  0:21:41 [Note] WSREP: Requesting state transfer: success, donor: 0
111216  0:21:42 [Note] WSREP: 0 (node1): State transfer to 1 (node2) complete.
111216  0:21:42 [Note] WSREP: Member 0 (node1) synced with group.
111216  0:21:42 [Note] WSREP: SST complete, seqno: 0
111216  0:21:42 [Note] Plugin 'FEDERATED' is disabled.
111216  0:21:42 InnoDB: The InnoDB memory heap is disabled
111216  0:21:42 InnoDB: Mutexes and rw_locks use GCC atomic builtins
111216  0:21:42 InnoDB: Compressed tables use zlib 1.2.3
111216  0:21:42 InnoDB: Using Linux native AIO
111216  0:21:42 InnoDB: Initializing buffer pool, size = 128.0M
111216  0:21:42 InnoDB: Completed initialization of buffer pool
111216  0:21:42 InnoDB: highest supported file format is Barracuda.
111216  0:21:42  InnoDB: Waiting for the background threads to start
111216  0:21:43 Percona XtraDB (http://www.percona.com) 1.1.8-20.1 started; log sequence number 15979
111216  0:21:43 [Note] Event Scheduler: Loaded 0 events
111216  0:21:43 [Note] WSREP: Signalling provider to continue.
111216  0:21:43 [Note] WSREP: Received SST: f912d2eb-27a2-11e1-0800-f34c520a3d4b:0
111216  0:21:43 [Note] WSREP: SST finished: f912d2eb-27a2-11e1-0800-f34c520a3d4b:0
111216  0:21:43 [Note] /usr/sbin/mysqld: ready for connections.
Version: '5.5.17'  socket: '/var/lib/mysql/mysql.sock'  port: 3306  Percona XtraDB Cluster (GPL), Rel
111216  0:21:43 [Note] WSREP: 1 (node2): State transfer from 0 (node1) complete.
111216  0:21:43 [Note] WSREP: Shifting JOINER -> JOINED (TO: 0)
111216  0:21:43 [Note] WSREP: Member 1 (node2) synced with group.
```

```
111216  0:21:43 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 0)
111216  0:21:43 [Note] WSREP: Synchronized with group, ready for connections
```

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

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

```
mysql
> CREATE DATABASE hello_tom;
```

The new database will be propagated to all nodes.

Enjoy!

5.3 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 xined and
- **mysqlchk** (place to /etc/xined.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
```

5.4 How to Execute Kewpie Tests

To use kewpie for testing it's recommended to use this MP. As it removes dbqp and integrates kewpie (and cuts size down to 25MB from 400+). To execute tests:

```
cd kewpie ; ./kewpie.py [--force ] [--libeatmydata] [--wsrep-provider-path=...]
```

The defaults are to run the cluster_basic and cluster_randgen suites against a 3 node cluster. Cluster_basic is used for small atomic tests like ADD/DROP single/multiple columns on a table and ensuring the change is replicated.

cluster_randgen is used for high stress transactional loads. There are single and multi-threaded variants. The load is a mix of INSERT/UPDATE/DELETE/SELECT statements. This includes both regular transactions, single queries, ROLLBACK's and SAVEPOINTs, and a mix of good and bad SQL statements.

To view all options, one may look at "./kewpie.py –help". Basic documentation is also available as sphinx docs in kewpie/docs folder. Here are the some of the most used options:

-force
: Run all tests despite failures (default is to stop test execution on first failure)

-libeatmydata
: Use libeatmydata if installed. This can greatly speed up testing in many cases. Can be used in conjunction with:

-libeatmydata-path to specify where the library is located.

-wsrep-provider-path
: By default, we expect / look for it in /usr/lib/galera/libgalera_smm.so (where it ends up via 'make install'...at least on Ubuntu). If one has an alternate library/location, specify it with this option.

Any additional suites may be run this way:

```
./kewpie.py [options] --suite=any/suitedir/from/kewpie/percona_tests
./kewpie.py --suite=crashme
```

5.5 How to Report Bugs

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

CHAPTER

SIX

PERCONA XTRADB CLUSTER LIMITATIONS

6.1 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.
- DELETE operation is unsupported on tables without primary key. Also rows in tables without primary key may appear in different order on different nodes. As a result SELECT...LIMIT... may return slightly different sets.
- **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. Anything bigger (e.g. huge LOAD DATA) will be rejected.
- 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 (10Gb network, SSD).
- The minimal recommended size of cluster is 3 nodes.
- DDL statements are problematic and may stall cluster. Later, the support of DDL will be improved, but will always require special treatment.

CHAPTER

SEVEN

GALERA DOCUMENTATION

The full documentation and reference is available on Galera Wiki

CHAPTER

EIGHT

MISC

8.1 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*.

IST Incremental State Transfer. Functionallity 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.

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* 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.

datadir The directory in which the database server stores its databases. Most Linux distribution use `/var/lib/mysql` by default.

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.

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. *:file:'mytable.TRG*. With the *.TRN* file, they represent all the trigger definitions.

.TRN File containing the triggers' Names associated to a table, e.g. *:file:'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.

CHAPTER

NINE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

INDEX

Symbols

C

D

I

L

M

P

S

X