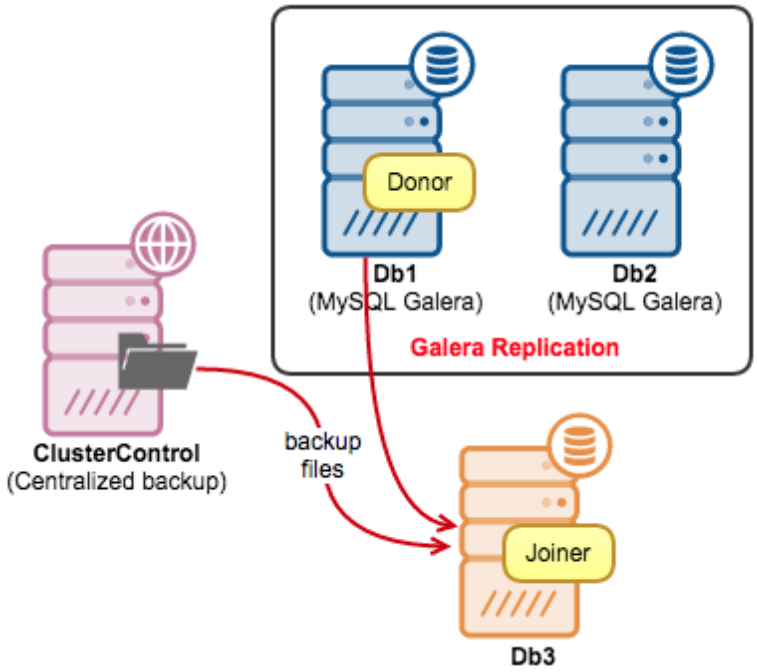
How to Avoid SST when adding a new node to Galera Cluster for MySQL or MariaDB

Severalnines July 15, 2015 Posted in: Backup Management

State Snapshot Transfer (SST) is a way for Galera to transfer a full data copy from an existing node (donor) to a new node (joiner). If you come from a MySQL replication background, it is similar to taking a backup of a master and restoring on a slave. In Galera Cluster, the process is automated and is triggered depending on the joiner state.

SST can be painful in some occasions, as it can block the donor node (with SST methods like mysqldump or rsync) and burden it when backing up the data and feeding it to the joiner. For a dataset of a few hundred gigabytes or more, the syncing process can take hours to complete - even if you have a fast network. It might be advisable to avoid e.g. when running in WAN environments with slower connects and limited bandwidth, or if you just want a very fast way of introducing a new node in your cluster.



In this blog post, we'll show you how to avoid SST.

SST Methods

Through the variable wsrep_sst_method, it is possible to set the following methods:

- mysqldump
- rsync
- xtrabackup/xtrabackup-v2

xtrabackup is non-blocking for the donor. Use xtrabackup-v2 (and not xtrabackup) if you are running on MySQL version 5.5.54 and later.

Incremental State Transfer (IST)

To avoid SST, we'll make use of IST and gcache. IST is a method to prepare a joiner by sending only the

missing writesets available in the donor's gcache. gcache is a file where a Galera node keeps a copy of writesets. IST is faster than SST, it is non-blocking and has no significant performance impact on the donor. It should be the preferred option whenever possible.

IST can only be achieved if all changes missed by the joiner are still in the gcache file of the donor. You will see the following in the donor's MySQL error log:

```
WSREP: async IST sender starting to serve tcp://10.0.0.124:4568 sending 689768-7612
```

And on the joiner side:

```
150707 17:15:53 [Note] WSREP: Signalling provider to continue.
150707 17:15:53 [Note] WSREP: SST received: d38587ce-246c-11e5-bcce-6bbd0831cc0f:689
150707 17:15:53 [Note] WSREP: Receiving IST: 71524 writesets, seqnos 689767-761291
```

Determining a good gcache size

Galera uses a pre-allocated gcache file of a specific size to store writesets in circular buffer style. By default, its size is 128MB. We have covered this in details here. It is important to determine the right size of the gcache size, as it can influence the data synchronization performance among Galera nodes.

The below gives an idea of the amount of data replicated by Galera. Run the following statement on one of the Galera node during peak hours (this works on MariaDB 10 and PXC 5.6, galera 3.x):

```
mysql> set @start := (select sum(VARIABLE_VALUE/1024/1024) from information_schema.g

+--------+---------+-----------------+-----------------------+
| MB/min | MB/hour | gcache Size(MB) | Time to full(minutes) |
+--------+---------+-----------------+-----------------------+
|   7.95 |  477.00 |  128            |                 16.10 |
+--------+---------+-----------------+-----------------------+
```

We can tell that the Galera node can have approximately 16 minutes of downtime, without requiring SST to join (unless Galera cannot determine the joiner state). If this is too short time and you have enough disk space on your nodes, you can change the wsrep_provider_options="gcache.size=<value>" to an appropriate value. In this example, setting gcache.size=1G allows us to have 2 hours of node downtime with high probability of IST when the node rejoins.

Avoiding SST on New Node

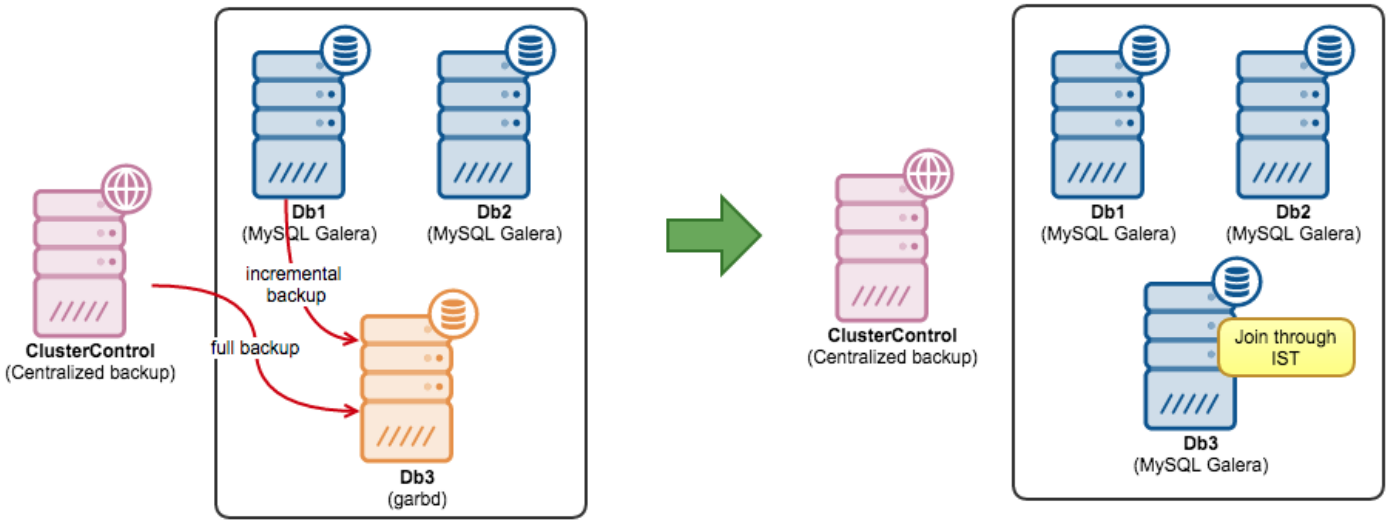
Sometimes, SST is unavoidable. This can happen when Galera fails to determine the joiner state when a node is joining. The state is stored inside grastate.dat. Should the following scenarios happen, SST will be triggered:

- grastate.dat does not exist under MySQL data directory - it could be a new node with a clean data directory, or e.g., the DBA manually deleted the file and intentionally forces Galera to perform SST
- This grastate.dat file has no seqno or group ID. This node crashed during DDL.
- ~~The seqno inside grastate.dat shows -1 while the MySQL server is still down, which means unclean shutdown or MySQL crashed/aborted due to database inconsistency.~~ (Thanks to Jay Janssen from Percona for pointing this out)
- grastate.dat is unreadable, due to lack of permissions or corrupted file system.

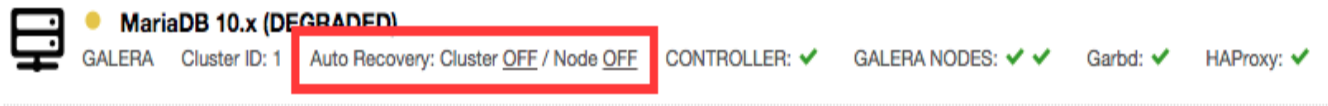
To avoid SST, we would get a full backup from one of the available nodes, restore the backup on the new

node and create a Galera state file so Galera can determine the node's state and skip SST.

In the following example, we have two Galera nodes with a garbd. We are going to convert the garbd node to a MySQL Galera node (Db3). We have a full daily backup created using xtrabackup. We'll create an incremental backup to get as close to the latest data before IST.



1. Install MySQL server for Galera on Db3. We will not cover the installation steps here. Please use the same Galera vendor as for the existing Galera nodes (Codership, Percona or MariaDB). If you are running ClusterControl, you can just use 'Add Node' function and disable the cluster/node auto recovery beforehand (as we don't want ClusterControl to automatically join the new node, that will trigger SST):



2. The full backup is stored on the ClusterControl node (refer to the diagram above). Firstly, copy and extract the full backup to Db3:

```
[root@db3]$ mkdir -p /restore/full
[root@db3]$ cd /restore/full
[root@db3]$ scp root@clustercontrol:/root/backups/mysql_backup/BACKUP-1/backup-
[root@db3]$ gunzip backup-full-2015-07-08_113938.xbstream.gz
[root@db3]$ xbstream -x < backup-full-2015-07-08_113938.xbstream
```

3. Increase the gcache size on all nodes to increase the chance of IST. Append the gcache.size parameter in wsrep_provider_options line in the MySQL configuration file:

```
wsrep_provider_options="gcache.size=1G"
```

Perform a rolling restart, one Galera node a time (ClusterControl user can use *Manage* > *Upgrades* > *Rolling Restart*):

```
$ service mysql restart
```

4. Before creating an incremental backup on Db1 to get the latest data since the last full backup, we need to copy back the xtrabackup_checkpoints file that we got from the extracted full backup on Db3. The incremental backup, when applied, will bring us closer to the current database state. Since we got 2 hours of buffer after increasing the gcache size, we should have enough time to restore the backup and create the necessary files to skip SST.

Create a base directory, which in our case will be /root/temp. Copy xtrabackup_checkpoints from Db3 into it:

```
[root@db1]$ mkdir -p /root/temp
[root@db1]$ scp root@db3:/restore/backup/xtrabackup_checkpoints /root/temp/
```

5. Create a target directory for incremental backup in Db3:

```
[root@db3]$ mkdir -p /restore/incremental
```

6. Now it is safe to create an incremental backup on Db1 based on information inside /root/temp /xtrabackup_checkpoints and stream it over to Db3 using SSH:

```
[root@db1]$ innobackupex --user=root --password=password --incremental --galera
```

If you don't have a full backup, you can generate one by running the following command on Db1 and stream it directly to Db3:

```
[root@db1]$ innobackupex --user=root --password=password --galera-info --stream
```

7. Prepare the backup files:

```
[root@db3]$ innobackupex --apply-log --redo-only /restore/full
[root@db3]$ innobackupex --apply-log /restore/full --incremental-dir=/restore/i
```

Ensure you got the following line at the end of the output stream as an indicator that the above succeeded:

```
150710 14:08:20  innobackupex: completed OK!
```

8. Build a Galera state file under the /restore/full directory based on the latest information from the incremental backup. You can get the information inside /restore/incremental/xtrabackup_galera_info:

```
[root@db3]$ cat /restore/full/xtrabackup_galera_info
d38587ce-246c-11e5-bcce-6bbd0831cc0f:1352215
```

Create a new file called grastate.dat under the full backup directory:

```
[root@db3]$ vim /restore/full/grastate.dat
```

And add the following lines (based on the xtrabackup_galera_info):

```
# GALERA saved state
version: 2.1
uuid:    d38587ce-246c-11e5-bcce-6bbd0831cc0f
seqno:   1352215
cert_index:
```

9. The backup is prepared and ready to be copied over to the MySQL data directory. Clear the existing path, copy the prepared data and assign correct ownership:

```
[root@db3]$ rm -Rf /var/lib/mysql/*
```

```
[root@db3]$ innobackupex --copy-back /restore/full
[root@db3]$ chown -Rf mysql.mysql /var/lib/mysql
```

10. If you installed garbd through ClusterControl, remove it by going to *Manage* > *Load Balancer* > *Remove Garbd* > *Remove* and skip the following command. Otherwise, stop garbd service on this node:

```
[root@db3]$ killall -9 garbd
```

11. The new node is now ready to join the rest of the cluster. Fire it up:

```
[root@db3]$ service mysql start
```

Voila, the new node should bypass SST and sync via IST. Monitor the output of MySQL error log and ensure you see something like below:

```
150710 14:49:58 [Note] WSREP: Signalling provider to continue.
150710 14:49:58 [Note] WSREP: SST received: d38587ce-246c-11e5-bcce-6bbd0831cc0f:135
150710 14:49:58 [Note] WSREP: Receiving IST: 4921 writesets, seqnos 1352215-1357136
```