15 Tips to improve your Galera Cluster Experience
Who am I ?

- Frédéric Descamps “lefred”
- @lefred
- http://about.me/lefred
- Percona Consultant since 2011
- Managing MySQL since 3.23
- devops believer
- I installed my first galera cluster in feb 2010
Who am I?

- Frédéric Descamps "lefred"
- @lefred
- http://about.me/lefred
- Percona Consultant since 2011
- Managing MySQL since 3.23
- devops believer
- I installed my first galera cluster in Feb 2010
Ready for countdown?
15
How to perform point in time recovery?

- Binary log must be enabled
- `log-slave-updates` should be enabled
The environment

writes

writes

writes

writes
Suddenly!
Suddenly!

- Oups ! DimO truncated a production table... :-S
- We can have 2 scenarios :
  - The application can keep running even without that table
  - The application musts be stopped !
Suddenly!

- Oups! DimO truncated a production table... :-S

- We can have 2 scenarios:
  - The application can keep running even without that table
  - The application musts be stopped!
Suddenly!

- Oups! DimO truncated a production table... :-S

- We can have 2 scenarios:
  - The application can keep running even without that table
  - The application musts be stopped!
Suddenly!

- Oups! DimO truncated a production table... :-S
- We can have 2 scenarios:
  - The application can keep running even without that table
  - The application musts be stopped!
Scenario 1: application must be stopped!
Scenario 1: application must be stopped!

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Stop the each node of the cluster
Scenario 1: application must be stopped!

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Stop the each node of the cluster
Scenario 1: application must be stopped!

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Stop the each node of the cluster
Scenario 1: application must be stopped!

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Stop the each node of the cluster
Scenario 1: application must be stopped!

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Stop the each node of the cluster

```
/etc/init.d/mysql stop
or
service mysql stop
```
Scenario 1: application must be stopped!

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Stop the each node of the cluster
  - Find the binlog file and position before “the event” happened
Scenario 1: application must be stopped!

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Stop the each node of the cluster
  - Find the binlog file and position before “the event” happened

```
# mysqlbinlog binlog.000001 | grep truncate -B 2
#140123 23:37:03 server id 1 end_log_pos 1224
Query thread_id=4 exec_time=0 error_code=0
SET TIMESTAMP=1390516623/*!*/;
truncate table speakers
```
Scenario 1: application must be stopped!

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Stop the each node of the cluster
  - Find the binlog file and position before “the event” happened
  - Restore the backup on one node
Scenario 1: application must be stopped!

- Stop the each node of the cluster
- Find the binlog file and position before “the event” happened
- Restore the backup on one node

```
# cp binlog.00001 ~
# innobackupex --apply-log .
etc..
```
Scenario 1: application must be stopped!

- Stop the each node of the cluster
- Find the binlog file and position before “the event” happened
- Restore the backup on one node
- Restart that node (being sure the application doesn't connect to it)

# /etc/init.d/mysql bootstrap-pxc
Scenario 1: application must be stopped!
Scenario 1: application must be stopped !

- Replay all the binary logs since the backup **BUT**
  the position of the event
Scenario 1: application must be stopped! (2)

- Replay all the binary logs since the backup but the position of the event

```bash
# cat xtrabackup_binlog_info
Binlog.000001 565
# mysqlbinlog binlog.000001 | grep end_log_pos | \ grep 1224 -B 1
#140123 23:36:53 server id 1 end_log_pos 1136
#140123 23:37:03 server id 1 end_log_pos 1224
# mysqlbinlog binlog.000001 -j 565 \ --stop-position 1136 | mysql
```
Scenario 1: application must be stopped! (2)
Scenario 1: application must be stopped! (2)

- Replay all the binary logs since the backup **BUT** the position of the event
- Start other nodes 1 by 1 and let them perform SST
- Enable connections from the application
Scenario 1: application must be stopped! (2)

- Replay all the binary logs since the backup **BUT** the position of the event
- Start other nodes 1 by 1 and let them perform SST
- Enable connections from the application
Scenario 1: application must be stopped! (2)

- Replay all the binary logs since the backup **BUT**
  the position of the event
- Start other nodes 1 by 1 and let them perform SST
- Enable connections from the application
Scenario 2: application can keep running
Scenario 2: application can keep running

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Take care of quorum (add garbd, change pc.weight, pc.ignore_quorum)
  - Find the binlog file and position before “the event” happened (thank you dim0!)
  - Remove one node from the cluster (and be sure the app doesn't connect to it, load-balancer...)
Scenario 2: application can keep running

- **We have Xtrabackup** (and it creates daily backups!)
- **We have binary logs**
- **These are the steps:**
  - *Take care of quorum* (add garbd, change pc.weight, pc.ignore_quorum)
  - *Find the binlog file and position before “the event” happened* (thank you dim0!)
  - *Remove one node from the cluster* (and be sure the app doesn't connect to it, load-balancer...)

2/5/14
Scenario 2: application can keep running

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Take care of quorum (add garbd, change pc.weight, pc.ignore_quorum)
  - Find the binlog file and position before “the event” happened (thank you dim0!)
  - Remove one node from the cluster (and be sure the app doesn't connect to it, load-balancer...)
Scenario 2: application can keep running

- We have Xtrabackup (and it creates daily backups!)
- We have binary logs
- These are the steps:
  - Take care of quorum (add garbd, change pc.weight, pc.ignore_quorum)
  - Find the binlog file and position before “the event” happened (thank you dim0!)
  - Remove one node from the cluster (and be sure the app doesn't connect to it, load-balancer...)
Scenario 2: application can keep running (2)
Scenario 2: application can keep running (2)

- Restore the backup on the node we stopped
- Start mysql without joining the cluster (--\texttt{wsrep-cluster-address=dummy://})
- Replay the binary log until the position of “the event”
- Export the table we need (mysqldump)
- Import it on the cluster
- Restart mysql on the off-line node and let it perform SST
Scenario 2: application can keep running (2)

- Restore the backup on the node we stopped
- Start mysql without joining the cluster (--wsrep-cluster-address=dummy://)
- Replay the binary log until the position of “the event”
- Export the table we need (mysqldump)
- Import it on the cluster
- Restart mysql on the off-line node and let it perform SST
Scenario 2: application can keep running (2)

- Restore the backup on the node we stopped
- Start mysql without joining the cluster (--wsrep-cluster-address=dummy://)
- Replay the binary log until the position of “the event”
- Export the table we need (mysqldump)
- Import it on the cluster
- Restart mysql on the off-line node and let it perform SST
Scenario 2: application can keep running (2)

- Restore the backup on the node we stopped
- Start mysql without joining the cluster (--wsrep-cluster-address=dummy://)
- Replay the binary log until the position of “the event”
- Export the table we need (mysqldump)
- Import it on the cluster
- Restart mysql on the off-line node and let it perform SST
Scenario 2: application can keep running (2)

- Restore the backup on the node we stopped
- Start mysql without joining the cluster (--wsrep-cluster-address=dummy://)
- Replay the binary log until the position of “the event”
- Export the table we need (mysqldump)
- Import it on the cluster
- Restart mysql on the off-line node and let it perform SST
Scenario 2: application can keep running (2)

- Restore the backup on the node we stopped
- Start mysql without joining the cluster (--wsrep-cluster-address=dummy://)
- Replay the binary log until the position of “the event”
- Export the table we need (mysqldump)
- Import it on the cluster
- Restart mysql on the off-line node and let it perform SST
Reduce “donation” time during XtraBackup SST

- When performing SST with XtraBackup the donor can still be active
- by default this is disabled in clustercheck (AVAILABLE_WHEN_DONOR=0)
- Running XtraBackup can increase the load (CPU / IO) on the server
Reduce “donation” time during XtraBackup SST (2)

- Using Xtrabackup 2.1 features helps to reduce the time of backup on the donor

  [mysqld]
  wsrep_sst_method=xtrabackup-v2
  wsrep_sst_auth=root:dim0DidItAgain

  [sst]
  streamfmt=xbstream

  [xtrabackup]
  compress
  compact
  parallel=8
  compress-threads=8
  rebuild-threads=8

  compress & compact can reduce the size of payload transferred among nodes but in general it slows down the process.
Move asynchronous slave to a new master in 5.5

- Master Host: A
- Master Log File: binlog.000002
- Master Position: 102

- Slave B
- Slave C

- In sync
- Async
Move asynchronous slave to a new master in 5.5 (2)

- Master host: B
- Master log file: binlog.000002
- Master position: 102
Move asynchronous slave to a new master in 5.5

(2)

- **master_host**: B
- **master_log_file**: binlog.000002
- **master_pos**: 102

- **B** writes binlog.000039
- **C** writes binlog.00001
  - 402
Move asynchronous slave to a new master in 5.5 (3)
Move asynchronous slave to a new master in 5.5 (3)

- How can we know which file and position need to be used by the async slave?
- Find the last received Xid in the relay log on the async slave (using mysqlbinlog)
Move asynchronous slave to a new master in 5.5 (3)

- How can we know which file and position need to be used by the async slave?
- Find the last received Xid in the relay log on the async slave (using mysqlbinlog)
Move asynchronous slave to a new master in 5.5 (3)

- How can we know which file and position need to be used by the async slave?
- Find the last received Xid in the relay log on the async slave (using mysqlbinlog)

```sql
# mysqlbinlog percona4-relay-bin.000002 | tail

MjM5NDMxMDMxODEtNTI4NzYyMzUxMDctMTM3NTA5NTI2NjUtNTc1ODY3MTc0MTg= '
/*!*/;
# at 14611057
#140131 12:48:12 server id 1 end_log_pos 29105924 xid = 30097
COMMIT/*!*/;
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
```
Move asynchronous slave to a new master in 5.5 (3)
Move asynchronous slave to a new master in 5.5 (3)

- How can we know which file and position need to be used by the async slave?
- Find the last received Xid in the relay log on the async slave (using mysqlbinlog)
- Find in the new master which binary position matches that same Xid
- Use the binary log file and the position for your CHANGE MASTER statement
Move asynchronous slave to a new master in 5.5 (3)

- How can we know which file and position need to be used by the async slave?
- Find the last received Xid in the relay log on the async slave (using mysqlbinlog)
- Find in the new master which binary position matches that same Xid
- Use the binary log file and the position for your CHANGE MASTER statement
Move asynchronous slave to a new master in 5.5 (3)

- How can we know which file and position need to be used by the async slave?
  
  *Find the last received Xid in the relay log on the async slave* (using `mysqlbinlog`)

  ```
  # mysqlbinlog percona3-bin.000004 | grep 'Xid = 30097'
  #140131 12:48:12 server id 1  end_log_pos 28911093  Xid = 30097
  ```

- Find in the new master which binary position matches that same Xid

- Use the binary log file and the position for your `CHANGE MASTER` statement
Move asynchronous slave to a new master in 5.5 (3)

- How can we know which file and position need to be used by the async slave?
- Find the last received Xid in the relay log on the async slave (using mysqlbinlog)
- Find in the new master which binary position matches that same Xid
- Use the binary log file and the position for your CHANGE MASTER statement
Move asynchronous slave to a new master in 5.5 (3)

- How can we know which file and position need to be used by the async slave?

  Async mysql> slave stop;

  Async mysql> change master to master_host='percona3',
             -> master_log_file='percona3-bin.000004',
             -> master_log_pos=28911093;

  Async mysql> start slave;

- Find in the new master which binary position matches that same Xid

- Use the binary log file and the position for your \texttt{CHANGE MASTER} statement
Move asynchronous slave to a new master in 5.6
Move asynchronous slave to a new master in 5.6

- With 5.6 and GTID it's easier!
- ... but ...
- It requires rsync SST (binlogs are needed)
- Or since Jan 30th
  `wsrep_sst_xtrabackup-v2` supports Xtrabackup 2.1.7 that makes is possible !!!
- Just change master ;-}
Move asynchronous slave to a new master in 5.6

- With 5.6 and GTID it's easier!
- ... but ...
- It requires rsync SST (binlogs are needed)
- Or since Jan 30th
  
  `wsrep_sst_xtrabackup-v2` supports Xtrabackup 2.1.7 that makes it possible!!
- Just change master ;-(
Move asynchronous slave to a new master in 5.6

- With 5.6 and GTID it's easier!
- ... but ...

- It requires rsync SST (binlogs are needed)
- Or since Jan 30th
  wsrep_sst_xtrabackup-v2 supports Xtrabackup 2.1.7 that makes is possible!!!
- Just change master ;-)
Move asynchronous slave to a new master in 5.6

- With 5.6 and GTID it's easier!
- ... but ...

- It requires rsync SST (binlogs are needed)
- Or since Jan 30th
  wsrep_sst_xtrabackup-v2 supports
  Xtrabackup 2.1.7 that makes is possible !!!
- Just change master ;-)
Move asynchronous slave to a new master in 5.6

- With 5.6 and GTID it's easier!
- ... but ...
- It requires rsync SST (binlogs are needed)
- Or since Jan 30th
  wsrep_sst_xtrabackup-v2 supports
  Xtrabackup 2.1.7 that makes is possible !!!
- Just change master ;-)}
Allow longer downtime for a node
Allow longer downtime for a node

• When a node goes off-line, when it joins again the cluster, it sends its last replicated event to the donor
• If the donor can send all next events, IST will be performed (very fast)
• If not... SST is mandatory
Allow longer downtime for a node

- When a node goes off-line, when it joins again the cluster, it sends its last replicated event to the donor
- If the donor can send all next events, IST will be performed (very fast)
- If not... SST is mandatory
Allow longer downtime for a node

- When a node goes off-line, when it joins again the cluster, it sends its last replicated event to the donor.
- If the donor can send all next events, IST will be performed (very fast).
- If not... SST is mandatory.
Allow longer downtime for a node (2)
Allow longer downtime for a node (2)

- Those events are stored on a cache on disk: `galera.cache`
- The size of the cache is **128Mb** by default
- It can be increased using `gcache.size` provider option:
**Allow longer downtime for a node (2)**

- Those events are stored on a cache on disk: `galera.cache`
- The size of the cache is **128Mb** by default
- It can be increased using `gcached.size` provider option:
Allow longer downtime for a node (2)

- Those events are stored on a cache on disk: \texttt{galera.cache}
- The size of the cache is \texttt{128Mb} by default
- It can be increased using \texttt{gcachex.size} provider option:
Allow longer downtime for a node (2)

- Those events are stored on a cache on disk: `galera.cache`
- The size of the cache is **128Mb** by default
- It can be increased using `gcache.size` provider option:

In `/etc/my.cnf`:

```
wsrep_provider_options = "gcache.size=1G"
```
Then choose the right donor!

- Let's imagine this:
Then choose the right donor! (2)

- Let's imagine this:
Then choose the right donor! (3)

- Let's imagine this:
Then choose the right donor! (4)

- Let's imagine this:
Then choose the right donor! (5)

Let's imagine this:

![Diagram showing the process of choosing the right donor with events and nodes labeled A, B, and C. The diagram includes arrows indicating the flow of information or actions.]
Then choose the right donor! (6)

- Let's imagine this:

Let's format the disk.
Then choose the right donor! (7)

- Let's imagine this:
Then choose the right donor! (8)

- Full SST needed
Then choose the right donor! (9)

- This is what we have now:

![Diagram showing a process flow with nodes A, B, and C, and events 1, 2, and 3.](image)
Then choose the right donor! (10)

- Let's remove node B for maintenance
Then choose the right donor! (11)

- Now let's remove **node C** to replace a disk :-(
Then choose the right donor! (12)

- Node C joins again and performs SST

writes

Event 1
Event 2
Event 3
Event 4
Event 5
Then choose the right donor! (12)

- Node C joins again and performs SST
Then choose the right donor! (13)

- **Node B** joins again but donor selection is not clever yet...

Join: last event = 3
Then choose the right donor! (13)

- Node B joins again but donor selection is not clever yet...

**SST will be needed!**
Then choose the right donor! (14)

- So how to tell node B that it needs to use node A?

```
# /etc/init.d/mysql start --wsrep-sst_donor=nodeA
```

```bash
Join:
last event = 3
```

Event 1
Event 2
Event 3
Event 4
Event 5
Event 6

writes
Then choose the right donor! (15)
Then choose the right donor! (15)

- With 5.6 you have now the possibility to know the lowest sequence number in `gcache` using `wsrep_local_cached_downto`

- To know the latest event's sequence number on the node that joins the cluster, you have two possibilities:
Then choose the right donor! (15)

- With 5.6 you have now the possibility to know the lowest sequence number in gcache using `wsrep_local_cached_downto`
- To know the latest event's sequence number on the node that joins the cluster, you have two possibilities:
Then choose the right donor! (15)

- With 5.6 you have now the possibility to know the lowest sequence number in `gcache` using `wsrep_local_cached_downto`

- To know the latest event's sequence number on the node that joins the cluster, you have two possibilities:

```bash
# cat grasdate.dat
# GALERA saved state
version: 2.1
uuid: 41920174-7ec6-11e3-a05a-6a2ab4033f05
seqno: 11
cert_index:
```
Then choose the right donor! (15)

- With 5.6 you have now the possibility to know the lowest sequence number in `gcach` using `wsrep_local_cached_downto`
- To know the latest event's sequence number on the node that joins the cluster, you have two possibilities:

```bash
# cat grasdate.dat
# GALERA saved state
version: 2.1
uuid: 41920174-7ec6-11e3-a05a-6a2ab4033f05
seqno: 11
cert_index:
```
Then choose the right donor! (15)

- With 5.6 you have now the possibility to know the lowest sequence number in gcachef using `wsrep_local_cached_downto`.

- To know the latest event's sequence number on the node that joins the cluster, you have two possibilities:

```bash
# mysqld_safe --wsrep-recover
140124 10:46:32 mysqld_safe Logging to '/var/lib/mysql/percona1_error.log'.
140124 10:46:32 mysqld_safe Starting mysqld daemon with databases from /var/lib/mysql
140124 10:46:32 mysqld_safe Skipping wsrep-recover for 41920174-7ec6-11e3-a05a-6a2ab4033f05:11 pair
140124 10:46:32 mysqld_safe Assigning 41920174-7ec6-11e3-a05a-6a2ab4033f05:11 to wsrep_start_position
140124 10:46:34 mysqld_safe mysqld from pid file /var/lib/mysql/percona1.pid ended
```
Then choose the right donor! (15)

- With 5.6 you have now the possibility to know the lowest sequence number in gcachex using `wsrep_local_cached_downto`.

- To know the latest event's sequence number on the node that joins the cluster, you have two possibilities:

```bash
# mysqld_safe --wsrep-recover
140124 10:46:32 mysqld_safe Logging to '/var/lib/mysql/percona1_error.log'.
140124 10:46:32 mysqld_safe Starting mysqld daemon with databases from '/var/lib/mysql'
140124 10:46:32 mysqld_safe Skipping wsrep-recover for 41920174-7ec6-11e3-a05a-6a2ab4033f05:11 pair
140124 10:46:32 mysqld_safe Assigning 41920174-7ec6-11e3-a05a-6a2ab4033f05:11 to wsrep_start_position
140124 10:46:34 mysqld_safe mysqld from pid file /var/lib/mysql/percona1.pid ended
```
Measuring Max Replication Throughput
Measuring Max Replication Throughput

- Since (5.5.33) `wsrep_desync` can be used to find out how fast a node can replicate
- The process is to collect the amount of transactions (events) during peak time for a define time range (let's take 1 min)
Measuring Max Replication Throughput

- Since (5.5.33) `wsrep_desync` can be used to find out how fast a node can replicate.
- The process is to collect the amount of transactions (events) during peak time for a define time range (let's take 1 min).
Measuring Max Replication Throughput

- Since (5.5.33) `wsrep_desync` can be used to find out how fast a node can replicate.

- The process is to collect the amount of transactions (events) during peak time for a defined time range (let's take 1 min).

```sql
mysql> pager grep wsrep
mysql> show global status like 'wsrep_last_committed';
   -> select sleep(60);
   -> show global status like 'wsrep_last_committed';

| wsrep_last_committed | 61472 |
| wsrep_last_committed | 69774 |
```
Measuring Max Replication Throughput

- Since (5.5.33) `wsrep_desync` can be used to find out how fast a node can replicate.

- The process is to collect the amount of transactions (events) during a peak time range (let's take 1 min):

```
mysql> pager grep wsrep
mysql> show global status like 'wsrep_last_committed';
   -> select sleep(60);
   -> show global status like 'wsrep_last_committed';

| wsrep_last_committed | 61472 |
| wsrep_last_committed | 69774 |
```

\[
69774 - 61472 = 8302
\]

\[
8302 / 60 = 138.36 \text{ tps}
\]
Measuring Max Replication Throughput
Since (5.5.33) `wsrep_desync` can be used to find out how fast a node can replicate.

The process is to collect the amount of transactions (events) during peak time for a define time range (let's take 1 min).

Then collect the amount of transactions and the duration to process them after the node was in `desync` mode and not allowing writes.

In `desync` mode, the node doesn't sent flow control messages to the cluster.
Measuring Max Replication Throughput

- Since (5.5.33) `wsrep_desync` can be used to find out how fast a node can replicate.
- The process is to collect the amount of transactions (events) during peak time for a defined time range (let's take 1 min).
- Then collect the amount of transactions and the duration to process them after the node was in desync mode and not allowing writes.
- In desync mode, the node doesn't send flow control messages to the cluster.
Measuring Max Replication Throughput

- Since (5.5.33) `wsrep_desync` can be used to find out how fast a node can replicate.

- The process is to collect the amount of transactions (events) during peak time for a define time range (let's take 1 min).

- Then collect the amount of transactions and the duration to process them after the node was in `desync` mode and not allowing writes.

- In `desync` mode, the node doesn't sent flow control messages to the cluster.
Measuring Max Replication Throughput

The process is to collect the amount of transactions (events) during peak time for a define time range (let's take 1 min) then collect the amount of transactions and the duration to process them after the node was in desync mode and not allowing writes.

In desync mode, the node doesn't sent flow control messages to the cluster.

```sql
set global wsrep_desync=ON; flush tables with read lock;
show global status like 'wsrep_last_committed';
select sleep( 60 ); unlock tables;
```

<table>
<thead>
<tr>
<th>Variable_name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsrep_last_committed</td>
<td>145987</td>
</tr>
</tbody>
</table>

Then collect the amount of transactions and the duration to process them after the node was in desync mode and not allowing writes.
Measuring Max Replication Throughput

- In another terminal you run `myq_gadget` and when `wsrep_local_recv_queue (Queue Dn)` is back to 0 check again the value of `wsrep_last_committed`. 
Measuring Max Replication Throughput

- In another terminal you run `myq_gadget` and when `wsrep_local_recv_queue (Queue Dn)` is back to 0 check again the value of `wsrep_last_committed`.

<table>
<thead>
<tr>
<th>time</th>
<th>P</th>
<th>cnf</th>
<th>#</th>
<th>cmt</th>
<th>sta</th>
<th>Queue</th>
<th>Ops</th>
<th>Bytes</th>
<th>Flow</th>
<th>Conflict</th>
<th>PApply</th>
<th>Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:25:24</td>
<td>P</td>
<td>7</td>
<td>3</td>
<td>Dono</td>
<td>T/T</td>
<td>0</td>
<td>8k</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13:25:25</td>
<td>P</td>
<td>7</td>
<td>3</td>
<td>Dono</td>
<td>T/T</td>
<td>0</td>
<td>8k</td>
<td>0</td>
<td>197</td>
<td>0</td>
<td>300K</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:26:46</td>
<td>P</td>
<td>7</td>
<td>3</td>
<td>Dono</td>
<td>T/T</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>209</td>
<td>0</td>
<td>318K</td>
<td>0</td>
</tr>
<tr>
<td>13:26:47</td>
<td>P</td>
<td>7</td>
<td>3</td>
<td>Dono</td>
<td>T/T</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>148</td>
<td>0</td>
<td>222K</td>
<td>0</td>
</tr>
</tbody>
</table>
Measuring Max Replication Throughput

- In another terminal you run `myq_gadget` and when `wsrep_local_recv_queue (Queue Dn)` is back to 0 check again the value of `wsrep_last_committed`.

This is when FTWRL is released.

<table>
<thead>
<tr>
<th>Time</th>
<th>Node</th>
<th>Ops</th>
<th>Bytes</th>
<th>Flow</th>
<th>Conflict</th>
<th>PApply</th>
<th>Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:25:24 P</td>
<td>7 3 Dono T/T</td>
<td>8k</td>
<td>0</td>
<td>0</td>
<td>0.0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13:25:25 P</td>
<td>7 3 Dono T/T</td>
<td>8k</td>
<td>0</td>
<td>197</td>
<td>0</td>
<td>300K</td>
<td>0.0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13:26:46 P</td>
<td>7 3 Dono T/T</td>
<td>7</td>
<td>0</td>
<td>209</td>
<td>0</td>
<td>318K</td>
<td>0.0</td>
</tr>
<tr>
<td>13:26:47 P</td>
<td>7 3 Dono T/T</td>
<td>0</td>
<td>0</td>
<td>148</td>
<td>0</td>
<td>222K</td>
<td>0.0</td>
</tr>
</tbody>
</table>
In another terminal you run myq_gadget and when `wsrep_local_recv_queue (Queue Dn)` is back to 0 check again the value of `wsrep_last_committed`.

This is when Galera catch up.

wsrep_last_committed = 165871

This is when FTWRL is released.
In another terminal you run myq_gadget and when `wsrep_local_recv_queue (Queue Dn)` is back to 0 check again the value of `wsrep_last_committed`

```
165871 - 145987 = 19884
19884 / 82 = 242.48 tps
```

We're currently at 57% of our capacity
Multicast replication

- By default, galera uses unicast TCP
- 1 copy of the replication message sent to all other nodes in the cluster
Multicast replication (2)

- By default, galera uses unicast TCP
- 1 copy of the replication message sent to all other nodes in the cluster
- More nodes, more bandwidth
Multicast replication (3)
If your network supports it you can use Multicast UDP for replication

- `wsrep_provider_options = "gmacast.mcast_addr = 239.192.0.11"
- `wsrep_cluster_cluster_address = gcomm://239.192.0.11"`
If your network supports it you can use Multicast UDP for replication

- `wsrep_provider_options = "gmacast.mcast_addr = 239.192.0.11"
- `wsrep_cluster_cluster_address = gcomm://239.192.0.11"`
Multicast replication (3)

- If your network supports it you can use Multicast UDP for replication
- `wsrep_provider_options = "gmacast.mcast_addr = 239.192.0.11"
- `wsrep_cluster_cluster_address = gcomm://239.192.0.11`
SSL everywhere!
SSL everywhere!

- It's possible to have the Galera replication encrypted via SSL.
- But now it's also possible to have SST over SSL, with xtrabackup_v2 and with rsync.
- https://github.com/tobz/galera-secure-rsync
SSL everywhere!

• It's possible to have the Galera replication encrypted via SSL
• But now it's also possible to have SST over SSL, with xtrabackup_v2 and with rsync
• https://github.com/tobz/galera-secure-rsync
SSL everywhere!

- It's possible to have the Galera replication encrypted via SSL
- But now it's also possible to have SST over SSL, with xtrabackup_v2 and with rsync
- [https://github.com/tobz/galera-secure-rsync](https://github.com/tobz/galera-secure-rsync)
SSL everywhere : certs creation
SSL everywhere : certs creation

- openssl req -new -x500 -days 365000 -nodes -keyout key.pem -out cert.pem
- Same cert and key must be copied on all nodes
- Copy them in /etc/mysql for example and let only mysql read them
SSL everywhere : certs creation

- openssl req -new -x500 -days 365000 -nodes -keyout key.pem -out cert.pem

- Same cert and key must be copied on all nodes

- Copy them in /etc/mysql for example and let only mysql read them
SSL everywhere : certs creation

- `openssl req -new -x500 -days 365000 -nodes -keyout key.pem -out cert.pem`

- Same cert and key must be copied on all nodes

- Copy them in `/etc/mysql` for example and let only `mysql` read them
SSL everywhere: galera configuration
SSL everywhere: galera configuration

- `wsrep_provider_options = "socket.ssl.cert=/etc/mysql/cert.pem; socket.ssl_key=/etc/mysql/key.pem"

- It's possible to set a remote Certificate Authority for validation (use `socket.ssl_ca`)

- All nodes **must** have SSL enabled
SSL everywhere : galera configuration

- `wsrep_provider_options = “socket.ssl.cert=/etc/mysql/cert.pem; socket.ssl_key=/etc/mysql/key.pem”`
- It's possible to set a remote Certificate Authority for validation (use `socket.ssl_ca`)
- All nodes **must** have SSL enabled
SSL everywhere: galera configuration

- `wsrep_provider_options = "socket.ssl.cert=/etc/mysql/cert.pem; socket.ssl_key=/etc/mysql/key.pem"

- It's possible to set a remote Certificate Authority for validation (use `socket.ssl_ca`)

- All nodes must have SSL enabled
SSL everywhere : SST configuration
SSL everywhere : SST configuration

- As Xtrabackup 2.1 supports encryption, it's now also possible to use SSL for SST
- Use `wsrep_sst_method=xtrabackup-v2` 
  
  ```
  [sst]
tkey=/etc/mysql/key.pem
tcert=/etc/mysql/cert.pem
encrypt=3
  ```
SSL everywhere : SST configuration

- As Xtrabackup 2.1 supports encryption, it's now also possible to use SSL for SST
  - Use `wsrep_sst_method=xtrabackup-v2`
    
    ```
    [sst]
    tkey=/etc/mysql/key.pem
tcert=/etc/mysql/cert.pem
encrypt=3
    ```
SSL everywhere : SST configuration

- As Xtrabackup 2.1 supports encryption, it's now also possible to use SSL for SST
- Use `wsrep_sst_method=xtrabackup-v2`
  
  `[sst]
  tkey=/etc/mysql/key.pem
cert=/etc/mysql/cert.pem
encrypt=3`
SSL everywhere : SST configuration

- As Xtrabackup 2.1 supports encryption, it's now also possible to use SSL for SST
  - Use `wsrep_sst_method=xtrabackup-v2`
  ```
  [sst]
tkey=/etc/mysql/key.pem
tcert=/etc/mysql/cert.pem
encrypt=3
  ```
SSL everywhere : SST configuration

- As Xtrabackup 2.1 supports encryption, it's now also possible to use SSL for SST
- Use `wsrep_sst_method=xtrabackup-v2`

````
[sst]
tkey=/etc/mySql/key.pem
tcert=/etc/mySql/cert.pem
encrypt=3
```
SSL everywhere : SST configuration

- As Xtrabackup 2.1 supports encryption, it's now also possible to use SSL for SST

- Use `wsrep_sst_method=xtrabackup-v2`

[sst]
tkey=/etc/mysql/key.pem
tcert=/etc/mysql/cert.pem
encrypt=3
SSL everywhere : SST configuration
SSL everywhere : SST configuration

- And for those using rsync?
- `galera-secure-rsync` acts like `wsrep_sst_rsync` but secures the communication with SSL using `socat`.
- Uses also the same cert and key file
- `wsrep_sst_method=secure_rsync`
- https://github.com/tobz/galera-secure-rsync
SSL everywhere: SST configuration

- And for those using `rsync`?
- `galera-secure-rsync` acts like `wsrep_sst_rsync` but secures the communication with SSL using `socat`.
- Uses also the same cert and key file
- `wsrep_sst_method=secure_rsync`
- [https://github.com/tobz/galera-secure-rsync](https://github.com/tobz/galera-secure-rsync)
SSL everywhere : SST configuration

- And for those using `rsync`?

- `galera-secure-rsync` acts like `wsrep_sst_rsync` but secures the communication with SSL using `socat`.

- Uses also the same cert and key file

- `wsrep_sst_method=secure_rsync`

- `https://github.com/tobz/galera-secure-rsync`
SSL everywhere : SST configuration

- And for those using **rsync**?
- **galera-secure-rsync** acts like **wsrep_sst_rsync** but secures the communication with SSL using **socat**.
- Uses also the same cert and key file
- **wsrep_sst_method=secure_rsync**
- [https://github.com/tobz/galera-secure-rsync](https://github.com/tobz/galera-secure-rsync)
SSL everywhere : SST configuration

- And for those using `rsync`?
- `galera-secure-rsync` acts like `wsrep_sst_rsync` but secures the communication with SSL using `socat`.
- Uses also the same cert and key file
- `wsrep_sst_method=secure_rsync`
- `https://github.com/tobz/galera-secure-rsync`
SSL everywhere : SST configuration

- And for those using `rsync`?
- `galera-secure-rsync` acts like `wsrep_sst_rsync` but secures the communication with SSL using `socat`.
- Uses also the same cert and key file
- `wsrep_sst_method=secure_rsync`
- `https://github.com/tobz/galera-secure-rsync`
Decode GRA* files
Decode GRA* files

- When a replication failure occurs, a GRA_*.*log file is created into the datadir
- For each of those files, a corresponding message is present in the mysql error log file
- Can be a false positive (bad DDL statement)... or not!
- This is how you can decode the content of that file
Decode GRA* files

- When a replication failure occurs, a GRA_*.*.log file is created into the datadir.
- For each of those files, a corresponding message is present in the mysql error log file.
- Can be a false positive (bad DDL statement)... or not!
- This is how you can decode the content of that file.
Decode GRA* files

- When a replication failure occurs, a \texttt{GRA\_*.log} file is created into the datadir.
- For each of those files, a corresponding message is present in the mysql error log file.
- Can be a false positive (bad DDL statement)... or not!
- This is how you can decode the content of that file.
Decode GRA* files

• When a replication failure occurs, a **GRA*_*.log** file is created into the datadir

• For each of those files, a corresponding message is present in the mysql error log file

• Can be a false positive (bad DDL statement)... or not !

• This is how you can decode the content of that file
Decode GRA* files (2)
Decode GRA* files (2)

- Download a binlog header file (http://goo.gl/kYTkY2)
- Join the header and one GRA_*/*.log file:
  - `cat GRA-header > GRA_3_3-bin.log`
  - `cat GRA_3_3.log >> GRA_3_3-bin.log`
- Now you can just use `mysqlbinlog -vvv` and find out what the problem was!
Choose GRA* files (2)

- Download a binlog header file (http://goo.gl/kYTkY2)
- Join the header and one GRA_* .log file:
  - cat GRA-header > GRA_3_3-bin.log
  - cat GRA_3_3.log >> GRA_3_3-bin.log
- Now you can just use mysqlbinlog -vvvv and find out what the problem was!
Decode GRA* files (2)

- Download a binlog header file (http://goo.gl/kYTkY2)
- Join the header and one GRA_*.log file:
  - `cat GRA-header > GRA_3_3-bin.log`
  - `cat GRA_3_3.log >> GRA_3_3-bin.log`
- Now you can just use `mysqlbinlog -vvv` and find out what the problem was!
Decode GRA* files (2)

- Download a binlog header file (http://goo.gl/kYTkY2)
- Join the header and one GRA_*.*.log file:
  
  - `cat GRA-header > GRA_3_3-bin.log`
  
  - `cat GRA_3_3.log >> GRA_3_3-bin.log`

- Now you can just use `mysqlbinlog -vvv` and find out what the problem was!
Decode GRA* files (2)

- Download a binlog header file (http://goo.gl/kYTkJ2)
- Join the header and one GRA_*.log file:
  - `cat GRA-header > GRA_3_3-bin.log`
  - `cat GRA_3_3.log >> GRA_3_3-bin.log`
- Now you can just use `mysqlbinlog -vvv` and find out what the problem was!
Decode GRA* files (2)

- Download a binlog header file (http://goo.gl/kYTkY2)
- Join the header and one GRA_*\.log file:
  - cat GRA-header > GRA_3_3-bin.log
  - cat GRA_3_3.log >> GRA_3_3-bin.log
- Now you can just use mysqlbinlog -vvv and find out what the problem was!

wsrep_log_conflicts = 1
wsrep_debug = 1
wsrep_provider_options = "cert.log_conflicts=1"
Avoiding SST when adding a new node
Avoiding SST when adding a new node

- It's possible to use a backup to prepare a new node.

- Those are the 3 prerequisites:
  - use XtraBackup >= 2.0.1
  - the backup needs to be performed with `--galera-info`
  - the `gcachex` must be large enough
Avoiding SST when adding a new node

• It's possible to use a backup to prepare a new node.

• Those are the 3 prerequisites:
  – use XtraBackup >= 2.0.1
  – the backup needs to be performed with --galera-info
  – the gcache must be large enough
Avoiding SST when adding a new node

• It's possible to use a backup to prepare a new node.

• Those are the 3 prerequisites:
  – use XtraBackup >= 2.0.1
  – the backup needs to be performed with --galera-info
  – the gcache must be large enough
Avoiding SST when adding a new node

- It's possible to use a backup to prepare a new node.
- Those are the 3 prerequisites:
  - use XtraBackup >= 2.0.1
  - the backup needs to be performed with `--galera-info`
  - the `gcach`e must be large enough
Avoiding SST when adding a new node

- It's possible to use a backup to prepare a new node.

- Those are the 3 prerequisites:
  - use XtraBackup >= 2.0.1
  - the backup needs to be performed with `--galera-info`
  - the `gcache` must be large enough
Avoiding SST when adding a new node (2)

- Restore the backup on the new node
- Display the content of xtrabackup_galera_info:
  5f22b204-dc6b-11e1-0800-7a9c9624dd66:23
- Create the file called grastate.dat like this:

  #GALERA saved state
  version: 2.1
  uuid: 5f22b204-dc6b-11e1-0800-7a9c9624dd66
  seqno: 23
  cert_index:
Avoiding SST when adding a new node (2)

- Restore the backup on the new node
- Display the content of `xtrabackup_galera_info`:
  
  5f22b204-dc6b-11e1-0800-7a9c9624dd66:23

- Create the file called `grastate.dat` like this:

  ```
  #GALERA saved state
  version: 2.1
  uuid: 5f22b204-dc6b-11e1-0800-7a9c9624dd66
  seqno: 23
  cert_index:
  ```
Avoiding SST when adding a new node (2)

- Restore the backup on the new node
- Display the content of `xtrabackup_galera_info`:
  5f22b204-dc6b-11e1-0800-7a9c9624dd66:23
- Create the file called `grastate.dat` like this:

  #GALERA saved state
  version: 2.1
  uuid: 5f22b204-dc6b-11e1-0800-7a9c9624dd66
  seqno: 23
  cert_index:
Avoiding SST when adding a new node (2)

- Restore the backup on the new node
- Display the content of `xtrabackup_galera_info`:
  5f22b204-dc6b-11e1-0800-7a9c9624dd66:23
- Create the file called `grastate.dat` like this:

  ```
  #GALERA saved state
  version: 2.1
  uuid: 5f22b204-dc6b-11e1-0800-7a9c9624dd66
  seqno: 23
  cert_index:
  ```
Avoiding SST when adding a new node (2)

- Restore the backup on the new node
- Display the content of xtrabackup_galera_info:
  5f22b204-dc6b-11e1-0800-7a9c9624dd66:23
- Create the file called grastate.dat like this:

  #GALERA saved state
  version: 2.1
  uuid:5f22b204-dc6b-11e1-0800-7a9c9624dd66
  seqno: 23
  cert_index:
Avoiding SST when adding a new node (2)

- Restore the backup on the new node
- Display the content of `xtrabackup galera info`:
  - `5f22b204-dc6b-11e1-0800-7a9c9624dd66:23`
- Create the file called `grastate.dat`
  
  ```
  #GALERA saved state
  version: 2.1
  uuid: 5f22b204-dc6b-11e1-0800-7a9c9624dd66
  seqno: 23
  cert_index:
  ```

```sql
mysql> show global status like 'wsrep_provider_version';
+-----------------+---------------------+
| Variable_name   | Value              |
+-----------------+---------------------+
| wsrep_provider_version | 2.1(r113) |
+-----------------+---------------------+
```
Play with quorum and weight
Play with quorum and weight

- Galera manages Quorum
- If a node does not see more than 50% of the total amount of nodes, reads/writes are not accepted
- Split brain is prevented
- This requires at least 3 nodes to work properly
- Can be disabled (but be warned!)
- You can cheat ;-


Play with quorum and weight

- Galera manages Quorum
- If a node does not see more than 50% of the total amount of nodes, reads/writes are not accepted
- Split brain is prevented
- This requires at least 3 nodes to work properly
- Can be disabled (but be warned!)
- You can cheat ;-)
Play with quorum and weight

- Galera manages Quorum
- If a node does not see more than 50% of the total amount of nodes, reads/writes are not accepted
- Split brain is prevented
- This requires at least 3 nodes to work properly
- Can be disabled (but be warned!)
- You can cheat ;-)
Play with quorum and weight

- Galera manages Quorum
- If a node does not see **more** than 50% of the total amount of nodes, reads/writes are not accepted
- Split brain is prevented
- This requires at least 3 nodes to work properly
- Can be disabled (but be warned!)
- You can cheat ;-)
Play with quorum and weight

- Galera manages Quorum
- If a node does not see more than 50% of the total amount of nodes, reads/writes are not accepted
- Split brain is prevented
- This requires at least 3 nodes to work properly
- Can be disabled (but be warned!)
- You can cheat ;-)

2/5/14
Play with quorum and weight

- Galera manages Quorum
- If a node does not see more than 50% of the total amount of nodes, reads/writes are not accepted
- Split brain is prevented
- This requires at least 3 nodes to work properly
- Can be disabled (but be warned!)
- You can cheat ;-(
Quorum: lost of connectivity
Quorum: lost of connectivity
Quorum: lost of connectivity

Network Problem

Does not accept Reads & Writes
Quorum: lost of connectivity
Quorum: lost of connectivity
Quorum: lost of connectivity
Quorum: lost of connectivity
Quorum: arbitrator
Quorum: arbitrator

- It's possible to use an arbitrator (garbd) to play an extra node. All traffic will pass through it but it won't have any MySQL running.
- Useful in case of storage available only for 2 nodes or if you have an even amount of nodes.
- Odd number of nodes is always advised
Quorum: arbitrator

- It's possible to use an arbitrator (garbd) to play an extra node. All traffic will pass through it but it won't have any MySQL running.
- Useful in case of storage available only for 2 nodes or if you have an even amount of nodes.
- Odd number of nodes is always advised
Quorum: arbitrator

• It's possible to use an arbitrator (garbd) to play an extra node. All traffic will pass through it but it won't have any MySQL running.

• Useful in case of storage available only for 2 nodes or if you have an even amount of nodes.

• Odd number of nodes is always advised
Quorum: cheat!
You can disable quorum but watch out! (you have been warned):

```plaintext
wsrep_provider_options = "pc.ignore_quorum=true"
```

You can define the weight of a node to affect the quorum calculation (default is 1):

```plaintext
wsrep_provider_options = "pc.weight=1"
```
Quorum: cheat!

- You can disable quorum but watch out! (you have been warned):
  
  \texttt{wsrep\_provider\_options = \textasciitilde pc.ignore\_quorum=true}

- You can define the weight of a node to affect the quorum calculation (default is 1):
  
  \texttt{wsrep\_provider\_options = \textasciitilde pc.weight=1}
Quorum: cheat!

- You can disable quorum but watch out! (you have been warned):
  
  ```
  wsrep_provider_options = "pc.ignore_quorum=true"
  ```

- You can define the weight of a node to affect the quorum calculation (default is 1):
  
  ```
  wsrep_provider_options = "pc.weight=1"
  ```
Quorum: cheat!

- You can disable quorum but watch out! (you have been warned):
  
  `wsrep_provider_options = "pc.ignore_quorum=true"`

- You can define the weight of a node to affect the quorum calculation (default is 1):
  
  `wsrep_provider_options = "pc.weight=1"`
Quorum: cheat!

- You can disable quorum but watch out! (you have been warned):
  
  `wsrep_provider_options = "pc.ignore_quorum=true"`

- You can define the weight of a node to affect the quorum calculation (default is 1):
  
  `wsrep_provider_options = "pc.weight=1"`
How to optimize WAN replication?

- Galera 2 requires all point-to-point connections for replication
How to optimize WAN replication?

- Galera 2 requires all point-to-point connections for replication.
How to optimize WAN replication?

- Galera 2 requires all point-to-point connections for replication
How to optimize WAN replication?

- Galera 2 requires all point-to-point connections for replication
How to optimize WAN replication?

- Galera 2 requires all point-to-point connections for replication
How to optimize WAN replication?

- Galera 2 requires all point-to-point connections for replication
How to optimize WAN replication?

- Galera 2 requires all point-to-point connections for replication.
How to optimize WAN replication?

- Galera 2 requires all point-to-point connections for replication
How to optimize WAN replication? (2)

- Galera 3 brings the notion of “cluster segments”
How to optimize WAN replication? (3)

- Segments gateways can change per transaction
How to optimize WAN replication? (3)

- Replication traffic between segments is minimized. Writesets are relayed to the other segment through one node.
How to optimize WAN replication? (3)

- Replication traffic between segments is minimized. Writesets are relayed to the other segment through one node.
How to optimize WAN replication? (4)

- From those local relays replication is propagated to every nodes in the segment
How to optimize WAN replication? (4)

- From those local relays replication is propagated to every nodes in the segment

```
gmcasts.segment = 1...255
```
Load balancers
Load balancers

- Galera is generally used in combination with a load balancer
- The most used is HA Proxy
- Codership provides one with Galera: glbd
Load balancers

- Galera is generally used in combination with a load balancer
- The most used is HA Proxy
- Codership provides one with Galera: glbd
Load balancers

- Galera is generally used in combination with a load balancer
- The most used is HA Proxy
- Codership provides one with Galera: glbd
Load balancers

- Galera is generally used in combination with a load balancer
- The most used is HA Proxy
- Codership provides one with Galera: glbd
Load balancers: myths, legends and reality
Load balancers: myths, legends and reality

- **TIME_WAIT**
  - On heavy load, you may have an issue with a large amount of TCP connections in TIME_WAIT state
  - This can lead to a TCP port exhaustion!

- **How to fix?**
  - Use `nolinger` option in HA Proxy (for glbd check [http://www.lefred.be/node/168](http://www.lefred.be/node/168)), but this lead to an increase of `Aborted_clients` if the client is connecting and disconnecting to MySQL too fast
  - Modify the value of `tcp_max_tw_buckets`
TIME_WAIT

- On heavy load, you may have an issue with a large amount of TCP connections in TIME_WAIT state
- This can lead to a TCP port exhaustion!

How to fix?

- Use nolinger option in HA Proxy (for glbd check http://www.lefred.be/node/168), but this lead to an increase of Aborted_clients if the client is connecting and disconnecting to MySQL too fast
- Modify the value of tcp_max_tw_buckets
Load balancers: myths, legends and reality

- **TIME_WAIT**
  - On heavy load, you may have an issue with a large amount of TCP connections in TIME_WAIT state
  - This can lead to a TCP port exhaustion!

- **How to fix?**
  - Use `nolinger` option in HA Proxy (for glbd check http://www.lefred.be/node/168), but this lead to an increase of `Aborted_clients` if the client is connecting and disconnecting to MySQL too fast
  - Modify the value of `tcp_max_tw_buckets`
Load balancers: myths, legends and reality

**TIME_WAIT**
- On heavy load, you may have an issue with a large amount of TCP connections in TIME_WAIT state
- This can lead to a TCP port exhaustion!

**How to fix?**
- Use `nolinger` option in HA Proxy (for glbd check http://www.lefred.be/node/168), but this lead to an increase of `Aborted_clients` if the client is connecting and disconnecting to MySQL too fast
- Modify the value of `tcp_max_tw_buckets`
**TIME_WAIT**

- On heavy load, you may have an issue with a large amount of TCP connections in **TIME_WAIT** state
- This can lead to a TCP port exhaustion!

**How to fix?**

- Use `nolinger` option in HA Proxy (for `glbd` check [http://www.lefred.be/node/168](http://www.lefred.be/node/168)), but this leads to an increase of `Aborted_clients` if the client is connecting and disconnecting to MySQL too fast
- Modify the value of `tcp_max_tw_buckets`
Load balancers: myths, legends and reality

- **TIME_WAIT**
  - On heavy load, you may have an issue with a large amount of TCP connections in TIME_WAIT state
  - This can lead to a TCP port exhaustion!

- **How to fix?**
  - Use `nolinger` option in HA Proxy (for glbd check [http://www.lefred.be/node/168](http://www.lefred.be/node/168)), but this leads to an increase of `Aborted_clients` if the client is connecting and disconnecting to MySQL too fast
  - Modify the value of `tcp_max_tw_buckets`
Load balancers: common issues

- Persistent Connections
  - Many people expect the following scenario:
Load balancers: common issues

- **Persisten Connections**
  - When the node that was specified to receive the persistent write fails for example
Load balancers: common issues

- Persistent Connections
  - When the node is back on-line...
Load balancers: common issues

- **Persistent Connections**
  - Only the new connections will use again the preferred node
Load balancers: common issues

- **Persistent Connections**
  - Only the new connections will use again the preferred node

![Diagram of load balancers and applications with persistent connections](image-url)
Load balancers: common issues

- Persistent Connections
  - Only the new connections will use again the preferred node
Load balancers: common issues
Load balancers: common issues

• Persitent Connections
  – HA Proxy decides where the connection will go at TCP handshake
  – Once the TCP session is established, the sessions will stay where they are!

• Solution?
  – With HA Proxy 1.5 you can now specify the following option:
    on-marked-down shutdown-sessions
Load balancers: common issues

• **Persisten Connections**
  - HA Proxy decides where the connection will go at TCP handshake
  - Once the TCP session is established, the sessions will stay where they are!

• **Solution ?**
  - With HA Proxy 1.5 you can now specify the following option:
    
    on-marked-down shutdown-sessions
Load balancers: common issues

- **Persistent Connections**
  - HA Proxy decides where the connection will go at TCP handshake
  - Once the TCP session is established, the sessions will stay where they are!

- **Solution?**
  - With HA Proxy 1.5 you can now specify the following option:
    `on-marked-down shutdown-sessions`
Load balancers: common issues

• **Persistent Connections**
  - HA Proxy decides where the connection will go at TCP handshake
  - Once the TCP session is established, the sessions will stay where they are!

• **Solution ?**
  - With HA Proxy 1.5 you can now specify the following option:

  `on-marked-down shutdown-sessions`
Load balancers: common issues

• **Persistent Connections**
  - HA Proxy decides where the connection will go at TCP handshake
  - Once the TCP session is established, the sessions will stay where they are!

• **Solution?**
  - With HA Proxy 1.5 you can now specify the following option: `on-marked-down shutdown-sessions`
Load balancers: common issues

• **Persistenent Connections**
  
  – HA Proxy decides where the connection will go at TCP handshake
  
  – Once the TCP session is established, the sessions will stay where they are!

• **Solution ?**

  – With HA Proxy 1.5 you can now specify the following option: on-marked-down shutdown-sessions
Taking backups without stalls
Taking backups without stalls

- When you want to perform a consistent backup, you need to take a FLUSH TABLES WITH READ LOCK (FTWRL)
- By default even with Xtrabackup
- This causes a Flow Control in galera
- So how can we deal with that?
Taking backups without stalls

- When you want to perform a consistent backup, you need to take a FLUSH TABLES WITH READ LOCK (FTWRL)
- By default even with Xtrabackup
- This causes a Flow Control in galera
- So how can we deal with that?
Taking backups without stalls

- When you want to perform a consistent backup, you need to take a FLUSH TABLES WITH READ LOCK (FTWRL)
- By default even with Xtrabackup
- This causes a Flow Control in galera
- So how can we deal with that?
Taking backups without stalls

- When you want to perform a consistent backup, you need to take a FLUSH TABLES WITH READ LOCK (FTWRL)
- By default even with Xtrabackup
- This causes a Flow Control in galera
- So how can we deal with that?
Taking backups without stalls
Taking backups without stalls

- Choose the node from which you want to take the backup
- Change the state to 'Donor/Desynched' (see tip 9)
  
  ```
  set global wsrep_desync=ON
  ```
- Take the backup
- Wait that `wsrep_local_recv_queue` is back down to 0
- Change back the state to 'Joined'
  
  ```
  set global wsrep_desync=OFF
  ```
Taking backups without stalls

- Choose the node from which you want to take the backup
- Change the state to 'Donor/Desynched' (see tip 9)
  
  ```
  set global wsrep_desync=ON
  ```
- Take the backup
- Wait that `wsrep_local_recv_queue` is back down to 0
- Change back the state to 'Joined'
  
  ```
  set global wsrep_desync=OFF
  ```
Taking backups without stalls

- Choose the node from which you want to take the backup
- Change the state to 'Donor/Desynced' (see tip 9)
  
  set global wsrep_desync=ON

- Take the backup
- Wait that wsrep_local_recv_queue is back down to 0
- Change back the state to 'Joined'
  
  set global wsrep_desync=OFF
Taking backups without stalls

- Choose the node from which you want to take the backup
- Change the state to 'Donor/Desynched' (see tip 9)
  ```
  set global wsrep_desync=ON
  ```
- Take the backup
- Wait that `wsrep_local_recv_queue` is back down to 0
- Change back the state to 'Joined'
  ```
  set global wsrep_desync=OFF
  ```
Taking backups without stalls

- Choose the node from which you want to take the backup
- Change the state to 'Donor/Desynched' (see tip 9)
  ```
  set global wsrep_desync=ON
  ```
- Take the backup
- Wait that `wsrep_local_recv_queue` is back down to 0
- Change back the state to 'Joined'
  ```
  set global wsrep_desync=OFF
  ```
Taking backups without stalls

- Choose the node from which you want to take the backup
- Change the state to 'Donor/Desynched' (see tip 9)
  
  ```
  set global wsrep_desync=ON
  ```
- Take the backup
- Wait that `wsrep_local_recv_queue` is back down to 0
- Change back the state to 'Joined'
  
  ```
  set global wsrep_desync=OFF
  ```
Taking backups without stalls

- Choose the node from which you want to take the backup
- Change the state to 'Donor/Desynched' (see tip 9)
  
  ```
  set global wsrep_desync=ON
  ```
- Take the backup
- Wait that `wsrep_local_recv_queue` is back down to 0
- Change back the state to 'Joined'
  
  ```
  set global wsrep_desync=OFF
  ```