PERCONA

Simple recipes to optimize your Galera Cluster

Stéphane Combaudon
September 22nd, 2015

Durability

Durability settings - single server

- `innodb_flush_log_at_trx_commit`
  - 1: can't lose data, default recommended
  - 2: can lose data on OS crash, not MySQL crash
  - 0: can lose data on MySQL crash

- `sync_binlog`
  - 0: binlog is synced every second, default value
  - 1: binlog synced at each commit, very safe and very slow

Durability settings - Galera

- `innodb_flush_log_at_trx_commit`
  - 2 is recommended
  - Cluster wide durability ensures data won't be lost
  - One exception: all nodes crash at the same time

- `sync_binlog`
  - 0 is recommended
  - Binlog is not needed to recover from a crash

Point-in-time recovery

Binary logging and Galera

- Technically binary logging is not necessary
- But what happens if you run `DROP TABLE`?
  - It's immediately replicated on all nodes
  - No way to get data back
- So enable binary logging!
- Also enable `log_slave_updates` b/c writes can come from multiple nodes

Decode GRA* files

GRA files?

- Such files are created each time replication fails
  - Can be useful to understand the issue
- But it's not a text file, how to read it?
  - By using mysqlbinlog
  - But the trick is you need to add a header

Generating a GRA header

- Start MySQL with binary logging and `binlog_checksum=NONE`

- Copy the first 120 bytes of the binlog

Reading the GRA file

```
mv GRA-header GRA_X_Y-bin.log

cat GRA_X_Y.log >> GRA_X_Y-bin.log

mysqlbinlog -vvv GRA_X_Y-bin.log
```

Find a good gcache size

What is the gcache?

- Local cache of the last executed events
- Memory-mapped file
- Can allow IST when a node has been offline for some time
- Default size is 128MB

How large should the gcache be?

- For large datasets, SST is expensive
  - You want IST to happen as often as possible
- If the gcache contains events for the last 30mn
  - You can stop a node for 30mn
  - No SST is needed when the node is restarted
- How many MBs is 30mn of writes?

Measuring writes in the gcache

- We'll use two status variables
  - `wsrep_replicated_bytes`: size of sent writesets
  - `wsrep_received_bytes`: size of received writesets

- SQL query

```
mysql> show global status like 'wsrep_received_bytes';
show global status like 'wsrep_replicated_bytes';
select sleep(60);
show global status like 'wsrep_received_bytes';
show global status like 'wsrep_replicated_bytes';
```

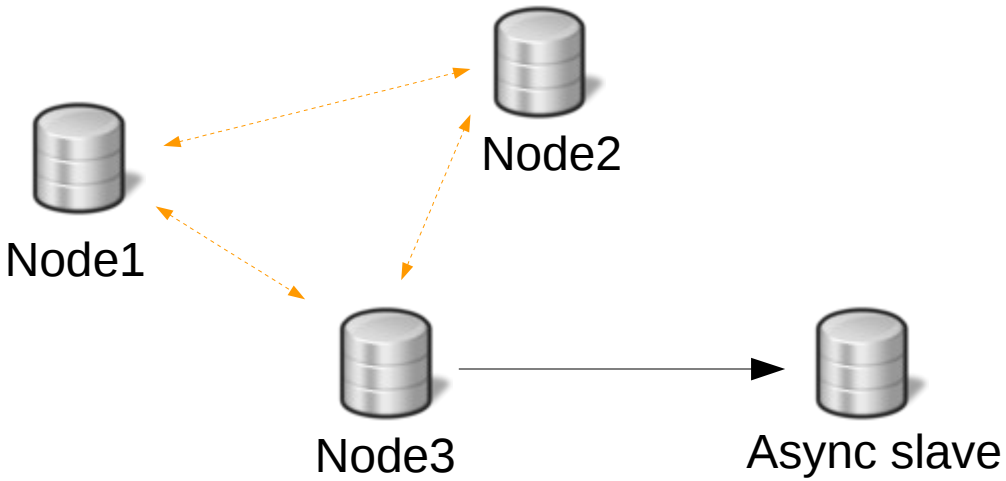
Now the magic formula

- (second `wsrep_received_bytes` – first `wsrep_received_bytes`) + (second `wsrep_replicated_bytes` – first `wsrep_replicated_bytes`) * 60 = Nb of bytes written in the gcache for 1 hour

- With this formula you can adjust the gcache size according to your needs

Asynchronous slaves (no GTID)

Cluster + Asynchronous slave



- How to move the async slave to Node2?
  - Remember binlog files on position don't match on Node2 and Node3

The Xid field

- The Xid is the Galera sequence number
  - Same for all nodes for a given transaction

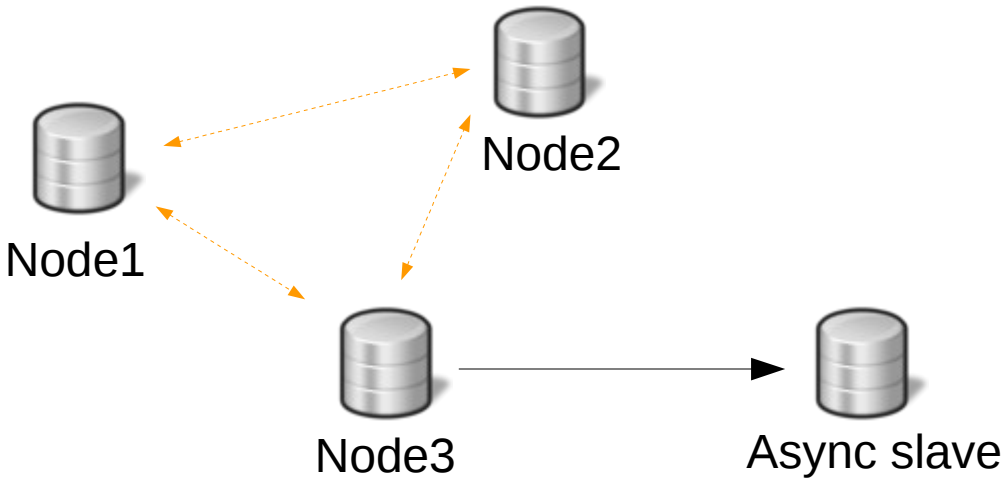
```
$ mysqlbinlog relay-log.000001
[…]
# at 26636
#140910 15:32:21 server id 127001  end_log_pos 26667 CRC32 0x5889ebca    Xid = 447
COMMIT/*!*/;
# at 26667
#700101  1:00:00 server id 127001  end_log_pos 26714 CRC32 0xb14028dc    Rotate to
mysql-bin.000036  pos: 4
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;
```

How to move an async slave

- #1 - Find the Xid of the last executed event in the relay log of the async slave
- #2 - Find the corresponding binlog file/position of the same Xid in the binlogs of the new master
- #3 - Run `CHANGE MASTER TO` with the coordinates found at step 2

Asynchronous slaves (GTID)

Cluster + Asynchronous slave



- How to move the async slave to Node2?
  - This time, async replication is GTID-based

How to move an async GTID slave

- Simply run `CHANGE MASTER TO` with the address of the new master
- No need to look at the relay logs or binlogs
- The replication protocol will take care of transmitting the potential missing events

But...

- GTIDs generated by the cluster have a special format
  - You can't know where the write came from
  - Can be confusing at first
- The usual fun can happen with errant transactions
  - Be careful, specifically with MyISAM tables

Understanding flow control

What is flow control?

- Replication feedback mechanism
- If one node is slowed down for some reason
  - It will tell the others (flow control messages)
  - The other nodes will stop processing writes
- Flow control will appear as a write stall across the whole cluster

Tuning flow control

- Flow control fires when the receive queue grows beyond a predefined threshold
  - This is tunable
  - Set `wsrep_provider_options="gcs.fc_limit=xxx"`
  - Default is 16 – very low!

Naive tuning

- “Let's set `gcs.fc_limit=99999999999”` and I'll never hear of flow control again”
- True, but a high flow control limit
  - Increases the likelihood of write conflicts
  - Increases memory and CPU usage
  - Will allow large apply lag ('I can't read my writes' at the app level)
- Values like `fc_limit=1024` or less are typical

Using wsrep_desync

Back to flow control

- Say we want to take a backup on Node1
  - The extra load is likely to trigger flow control
  - The whole cluster will see write stalls: not nice!

- What can we do?
  - Increase `gcs.fc_limit`: okay but not ideal
  - Use `wsrep_desync=ON`!

wsrep_desync=ON

- Indicates that the node will never trigger flow control
  - Even if the receive queue gets very large
  - So our backup won't degrade performance on the whole cluster
- Here is our backup procedure
  - `SET GLOBAL wsrep_desync=ON;`
  - (Optionally remove node from load balancer)
  - Take backup
  - `SET GLOBAL wsrep_desync=OFF;`

Group segments for WAN replication

Group communication

- Requires point-to-point connections between nodes
- If the nodes are geographically distributed, group communication becomes expensive
- With Galera 3, a node can be a gateway for other local nodes
  - The gateway changes at each transaction

Defining segments

- Node1 (EU)
  - `wsrep_provider_options="gmcast.segment=1"`
- Node2 (EU)
  - `wsrep_provider_options="gmcast.segment=1"`
- Node3 (US)
  - `wsrep_provider_options="gmcast.segment=2"`
- Node4 (US)
  - `wsrep_provider_options="gmcast.segment=2"`

Schema changes

TOI

- “Total Order Isolation”
  - `ALTER TABLE` is executed at the same point in time on all nodes
  - Guarantees that all nodes are consistent, all the time
  - But no other concurrent write is allowed
  - Good for small tables, for non backward-compatible changes
  - Painful for large tables

RSU

- “Rolling Schema Upgrade”
  - `ALTER TABLE` is local only
  - Node desyncs with the cluster while applying the schema change
  - You need to run `ALTER TABLE` on all nodes!
  - Great for large tables
  - But only works for backward-compatible changes

pt-online-schema-change

- Only works with TOI
  - Does the usual magic to make the change appear as an online change
  - Best of both worlds
  - But make sure to understand the numerous options

Choosing the right SST method

wsrep_sst_method option

- `rsync`
  - Works out of the box
  - Locks the donor
  - Very good for test, not for production
- `xtrabackup-v2`
  - Needs some configuration (dedicated MySQL user)
  - Does not lock the donor
  - Recommended for production
- `wsrep_sst_method=xtrabackup` is deprecated

Speeding up XtraBackup SST

Compression & Parallelism

- Available when using the xbstream format

```
[mysqld]
wsrep_sst_method=xtrabackup-v2
wsrep_sst_auth=sst:mypwd

[sst]
streamfmt=xbstream

[xtrabackup]
parallel=8
compress
compress-threads=8
```

Compression & Parallelism

- Available when using the xbstream format

```
[mysqld]
wsrep_sst_method=xtrabackup-v2
wsrep_sst_auth=sst:mypwd

[sst]
streamfmt=xbstream

[xtrabackup]
parallel=8
compress
compress-threads=8
```

Set SST method

Compression & Parallelism

- Available when using the xbstream format

```
[mysqld]
wsrep_sst_method=xtrabackup-v2
wsrep_sst_auth=sst:mypwd
```

Set SST method

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

Use xbstream format

```
[xtrabackup]
parallel=8
compress
compress-threads=8
```

Compression & Parallelism

- Available when using the xbstream format

```
[mysqld]
wsrep_sst_method=xtrabackup-v2
wsrep_sst_auth=sst:mypwd
```

Set SST method

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

Use xbstream format

```
[xtrabackup]
parallel=8
compress
compress-threads=8
```

Parallel data copy

Compression & Parallelism

- Available when using the xbstream format

```
[mysqld]
wsrep_sst_method=xtrabackup-v2
wsrep_sst_auth=sst:mypwd

[sst]
streamfmt=xbstream

[xtrabackup]
parallel=8
compress
compress-threads=8
```

Set SST method

Use xbstream format

Parallel data copy

Compression with multiple threads

Donor selection

Joining the cluster back

- A donor is selected
- The joiner sends the GTID of its last replicated event

- If the missing events of the joiner are in the gcache of the donor: IST (fast!)
- Else: SST (slow!)

Sometimes it bites... - 1

- Assume:
  - 1TB dataset
  - Node1 has just restarted (gcache is almost empty)
  - The gcache of Node2 contains 1hr of events
  - Node3 has restarted after being down for 30mn
- Which node will be selected as the donor?

Sometimes it bites... - 2

- If you think Node2 will always be selected
  - You're wrong!
  - Donor selection is essentially random

- So if we pick Node2: IST (takes minutes)
- But if we pick Node1: SST (takes hours)

wsrep_sst_donor (Galera 3)

- You can use `wsrep_sst_donor=Node2`
  - Indicates the preferred donor node
- Other scenario
  - Some nodes in EU, some nodes in US
  - You want the EU node to resync with other EU nodes if possible, same for US nodes

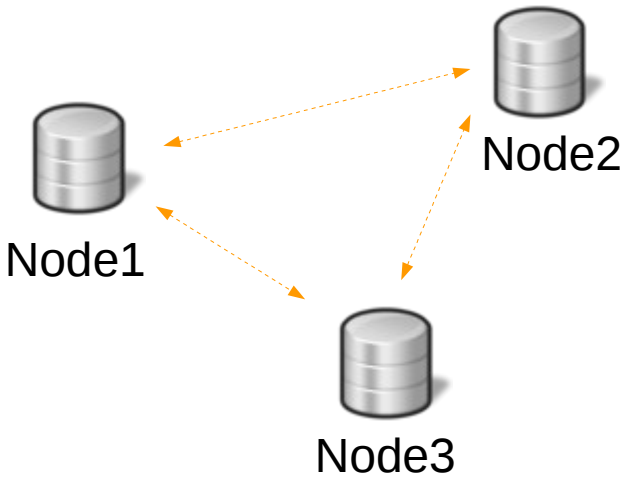
What's in the gcache?

- You sometimes need to know if events will be found in the gcache of a node
  - To choose the right donor

- In 5.6, use the `wsrep_local_cached_downto` status variable
  - Indicates the lowest seqno in the local gcache

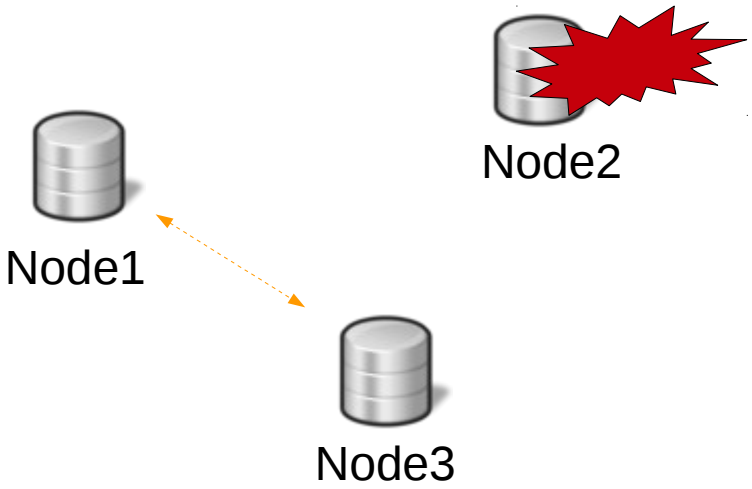
Understanding quorum

Typical 3-node cluster



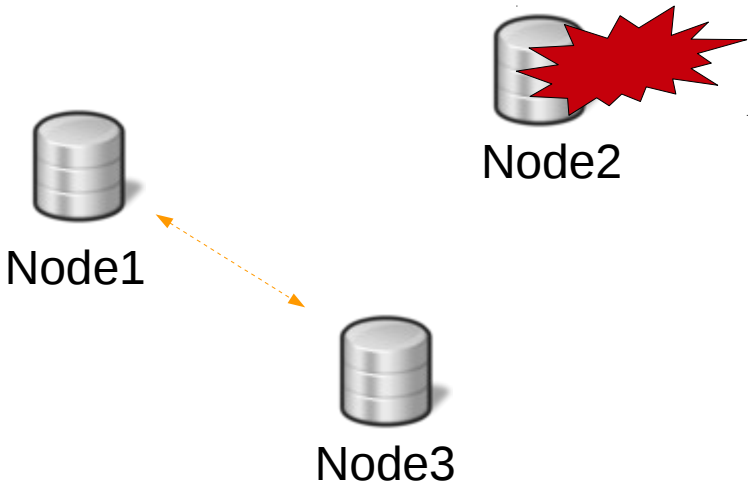
- What happens if Node2 fails?

Typical 3-node cluster



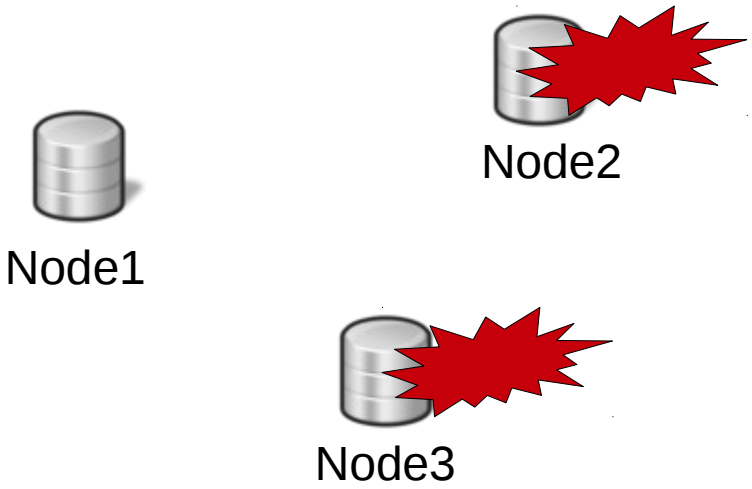
- Node1 and Node3 keep working normally
  - They have > 50% of the votes

Typical 3-node cluster



- What happens if Node3 also fails?

Typical 3-node cluster



- Node1 stops processing queries
  - It has <= 50% of the votes
  - Reads and writes are forbidden
  - New `wsrep_dirty_reads` variable in PXC 5.6.24+

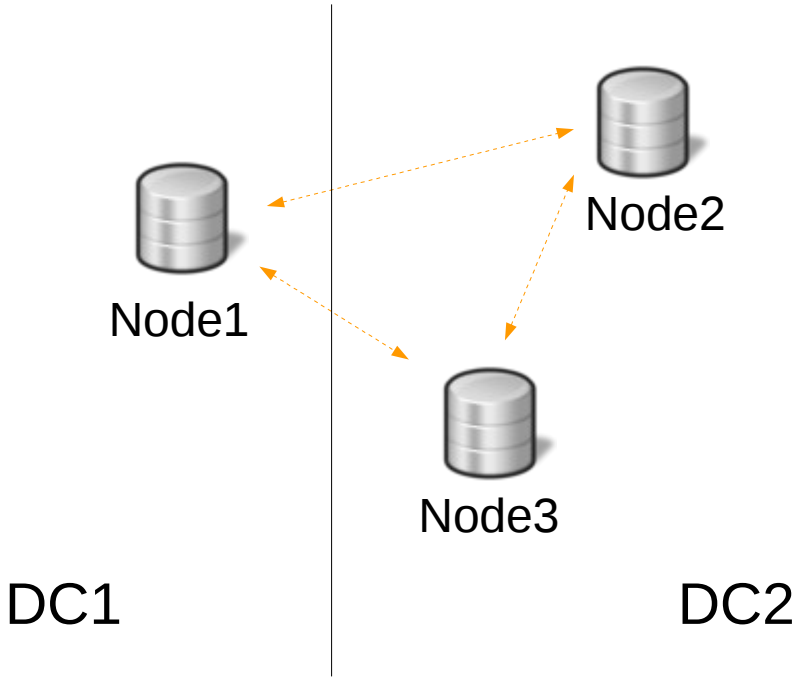
Multiple node failures

- 3 nodes don't protect you from a failure of 2 nodes
  - Because the remaining node doesn't have quorum
- With 5 nodes, the remaining 3 nodes have quorum, so:
  - With 3 nodes, you can lose up to 1 node
  - With 5 nodes, you can lose up to 2 nodes
  - ...

High Availability across multiple DCs

Node failure vs DC failure

- We saw earlier how to select the size of the cluster wrt simultaneous node failures
- Now what about a whole DC failure?

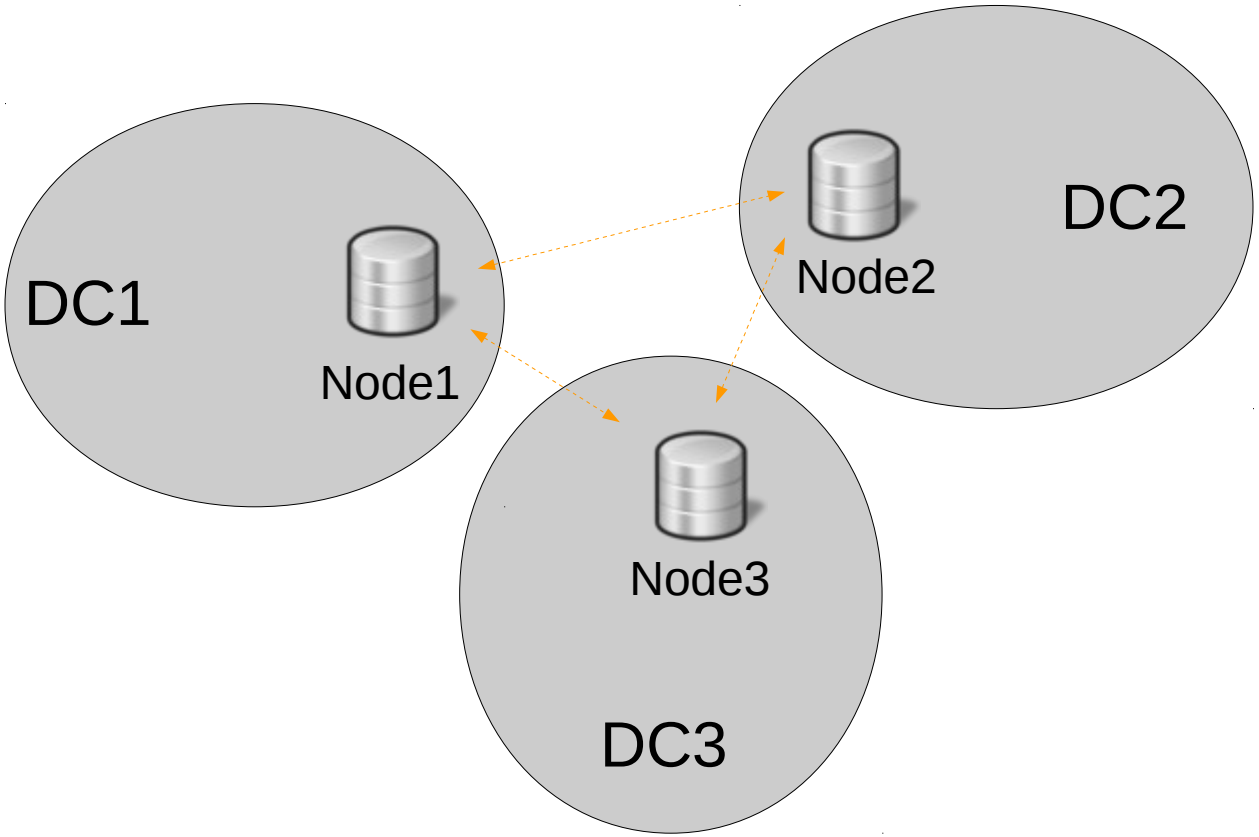


DC failure

- If DC1 fails, 2 nodes are remaining
  - Cluster is up
- But if DC2 fails, only 1 node is remaining
  - Cluster is down
- What about more nodes?
  - Same situation
- With 2 DC, a DC failure can affect all nodes
  - And manual intervention can be needed

HA with multiple DCs

- You need 3 DCs



Thanks for attending!

Feel free to drop me a line at:

stephane.combaudon@percona.com