PERCONA LIVE ONLINE MAY 12 - 13th 2021

Comparing Highly Available Solutions With Percona XtraDB Cluster and Percona Server with Group Replication

Marco Tusa, Percona

About Me

- Open-source enthusiast
- MySQL tech lead
- Principal architect
- Working in the DB/development world over 34 years (yes, I am that old)
- Open-source developer and community contributor

What This Presentation Covers

… What is it about?

- An overview of what HA and DR mean
- The most common issues
- How PXC and PS-GR cover the above needs
- Illustrate some differences

… What isn’t it about?

- We are not comparing performance
- We are not going into the deep tech mumbo jumbo (which is unusual for me!)

OVH Case

I think this case/picture will be in MANY slides for a long time!

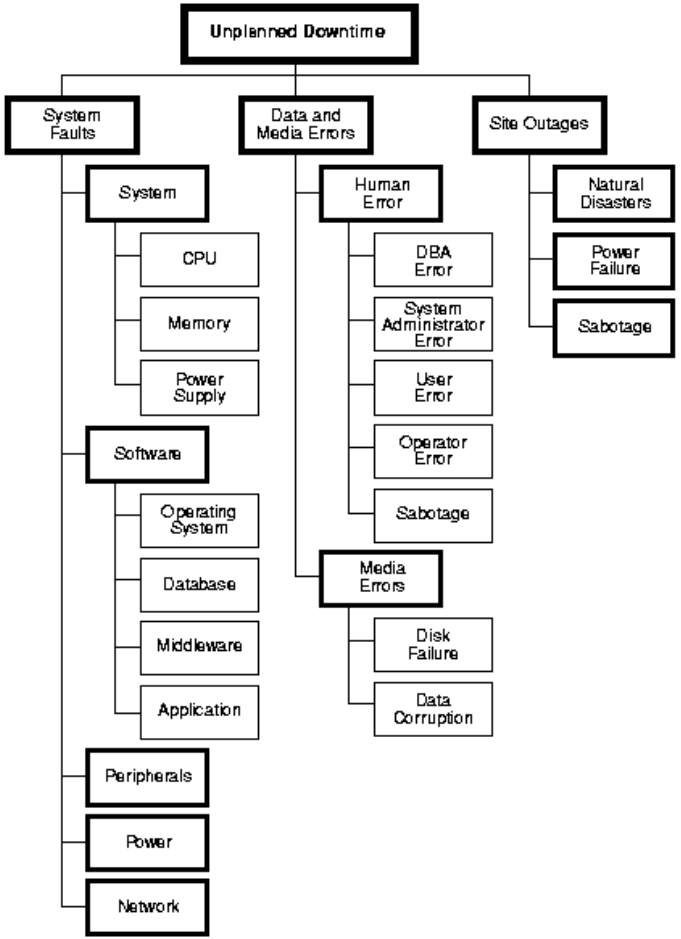
Lista Contatti Aperti

id_contatto	Gruppo	Creazione	Stato
0000426620	ASSISTENZA TECNICA	30/03/2021 14:06:17	CHIUSO
0000426397	ASSISTENZA TECNICA	22/03/2021 17:10:28	CHIUSO
0000426360	ASSISTENZA TECNICA	22/03/2021 09:51:24	CHIUSO
0000426359	ASSISTENZA TECNICA	22/03/2021 09:46:49	CHIUSO
0000424263	ASSISTENZA TECNICA	20/11/2018 13:09:07	CHIUSO

Unplanned Downtime

- Unplanned downtime can be classified into downtime caused by system faults, data and media errors, and site outages.
- System faults can occur in the system itself (CPU, memory, or power supply), the software (operating system, database, middleware, or application), peripherals, power, or the network.
- Data errors include human error by DBAs, system administrators, users, and operators, as well as sabotage.
- Media errors include disk failure and data corruption.
- Site outages include natural disasters, power failure, and sabotage.

The more you add to the stack, the more you increase the chance of failures.



https://docs.oracle.com/cd/A91202_01/901_doc/rac.901/a89867/pshavdtl.htm#6062

What is HA?

Availability %	Downtime per year	Downtime per month	Downtime per week	Downtime per day
99% ("two nines")	3.65 days	7.31 hours	1.68 hours	14.40 minutes
99.5% ("two nines five")	1.83 days	3.65 hours	50.40 minutes	7.20 minutes
99.9% ("three nines")	8.77 hours	43.83 minutes	10.08 minutes	1.44 minutes
99.95% ("three nines five")	4.38 hours	21.92 minutes	5.04 minutes	43.20 seconds
99.99% ("four nines")	52.60 minutes	4.38 minutes	1.01 minutes	8.64 seconds
99.995% ("four nines five")	26.30 minutes	2.19 minutes	30.24 seconds	4.32 seconds
99.999% ("five nines")	5.26 minutes	26.30 seconds	6.05 seconds	864.00 milliseconds

High Availability (HA) *is a characteristic of a system, which aims to ensure an agreed level of operational performance, usually uptime, for a higher than normal period.*

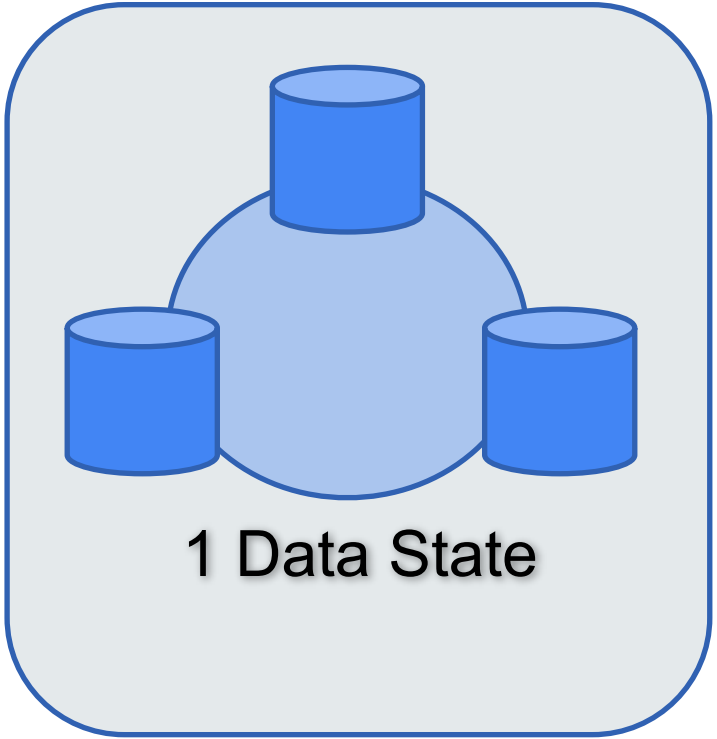
What is HA?

- **Geographic distribution in High Availability** is to cover service availability in a location
    - **10 Gb Ethernet best case scenario**
      400 metre distance max

What is DR?

- **Geographic distribution in Disaster Recovery** is to ensure you can restore service, in a geographically distributed location
  - Real speed may vary
  - Linear distance ~1000Km

The Datacentric Approach



<u>Tightly coupled cluster</u>

- **Percona PXC**
  - MySQL-based with custom Galera (Codership) implementation
- **Percona Server**
  - Uses Group Replication

Both solutions use additional elements presented in the distribution as:

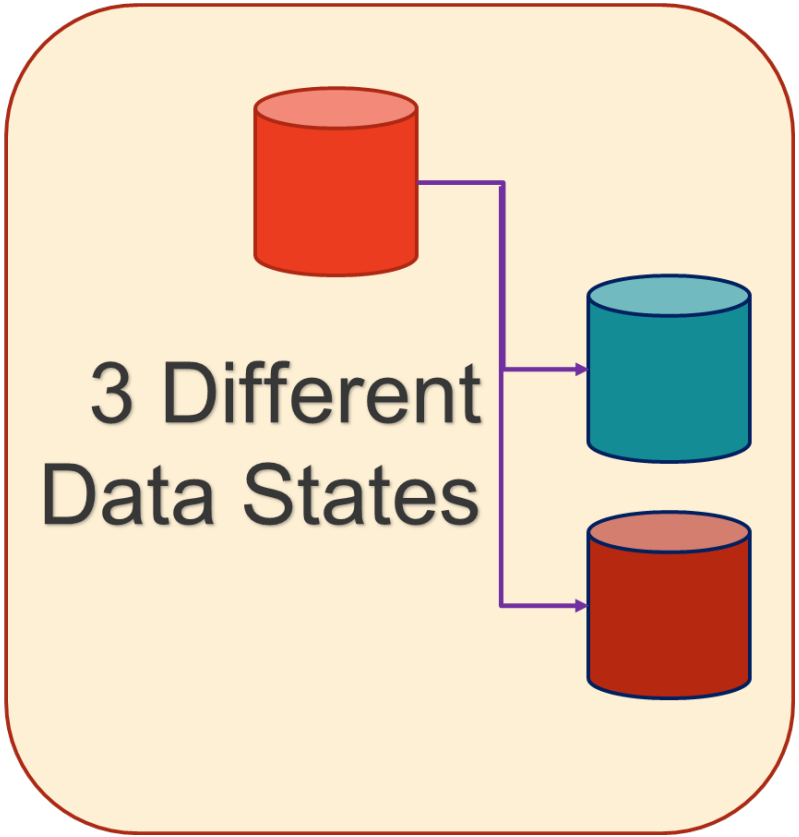
- HAProxy
- MySQL Router
- ProxySQL

The distribute approach

Loosely coupled cluster

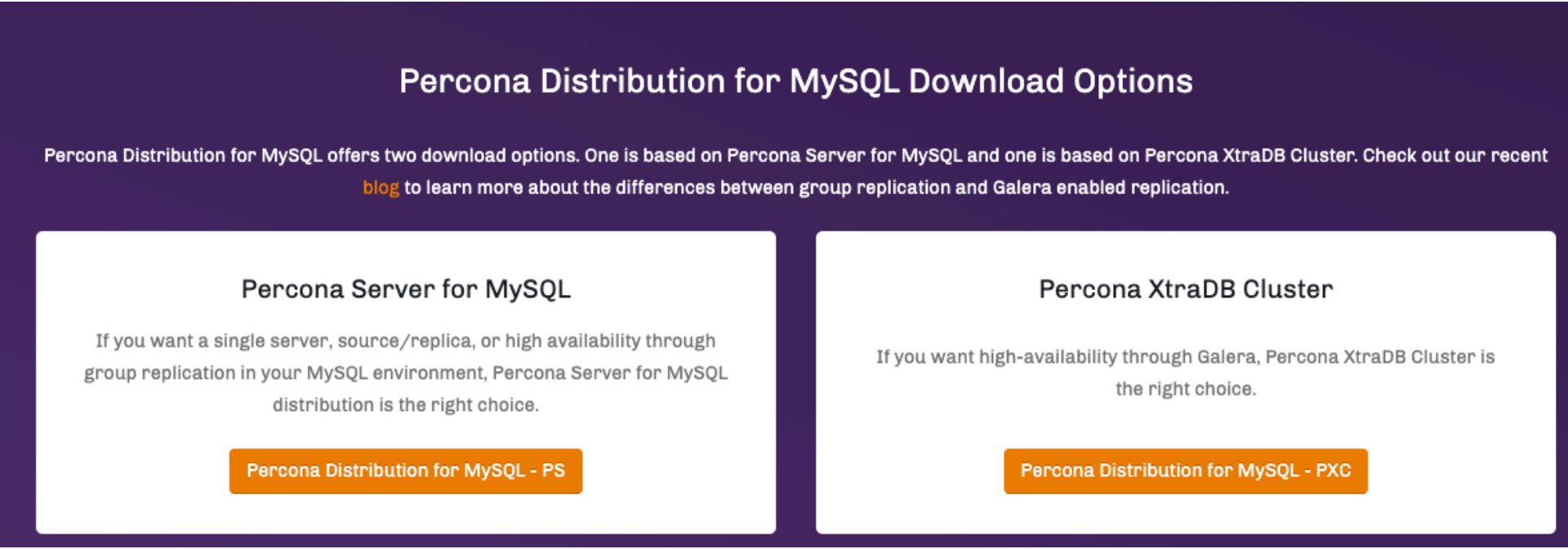
- **In MySQL world is based on Asynchronous replication**
- **Supported by PXC and PS with GR**

https://www.slideshare.net/marcotusa/best-practicehigh-availabilitysolutiongeodistributedfinal



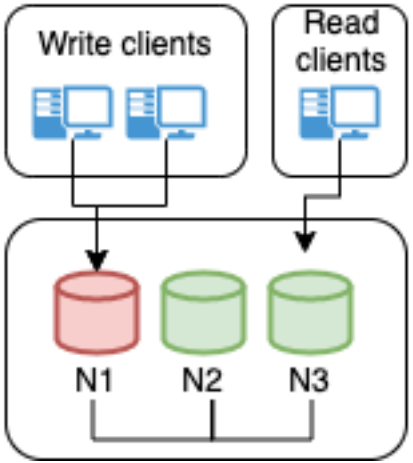
What Percona Implements (For MySQL)

We use three different solutions, bundled into two different Distributions



Single Primary vs Multi Primary

Single Primary



← Scaling reads

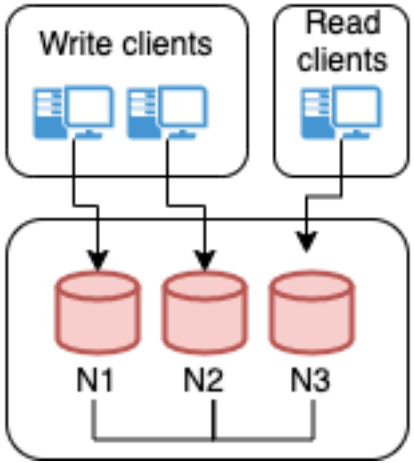
← Local conflict resolution

→ Complicate distributed conflict resolution

→ No write scaling

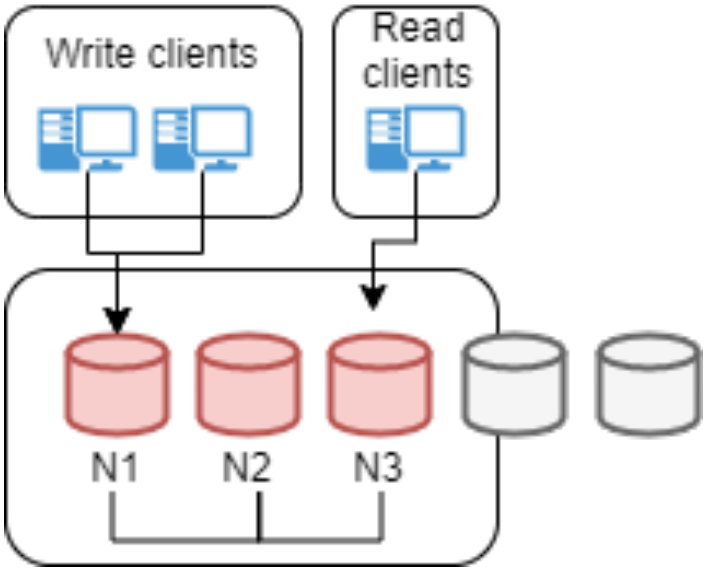
→ No sharding

Multi Primary



Cluster setup

PXC



- All W/R by default
- Number must be odd for quorum calculation
- Scaling READ only
- More nodes more noise Linear increase

GR

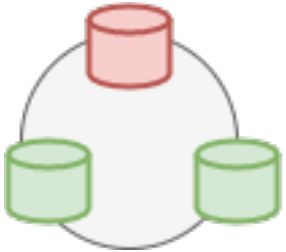
- Primary R/W Secondary R only
- Number cannot over 9 nodes
- Scaling READ only
- No need to be odd number, but I don’t buy it

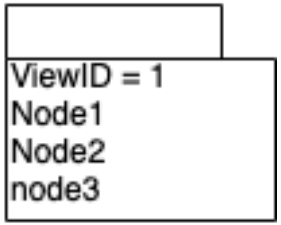
Birds-Eye View: Failure Detection

- **Node delay**
  - Saturation
  - Malfunction
  - Wrong dimensioning
- **Network issue**
  - Network latency
  - Network fluctuations
  - And more

Birds-Eye View: Failure Detection

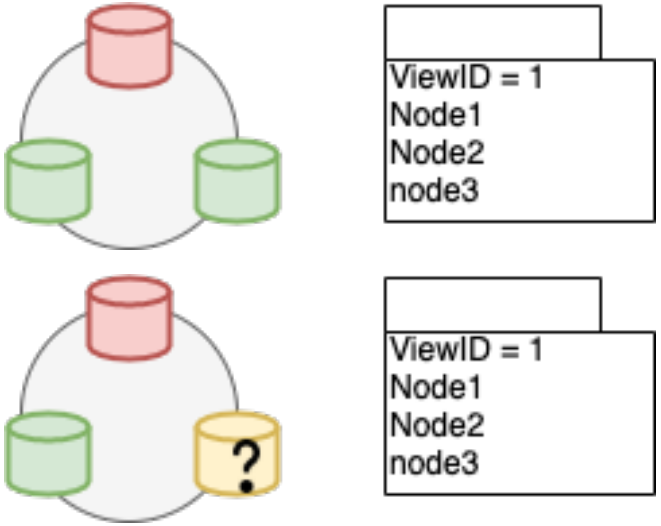
- **Node delay**
  - Saturation
  - Malfunction
  - Wrong dimensioning
- **Network issue**
  - Network latency
  - Network fluctuations
  - And more





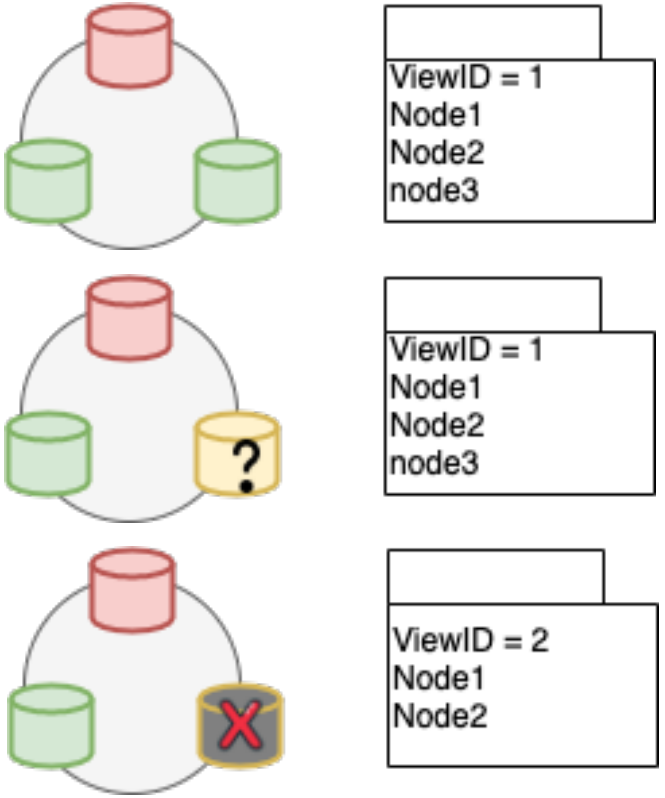
Birds-Eye View: Failure Detection

- **Node delay**
  - Saturation
  - Malfunction
  - Wrong dimensioning
- **Network issue**
  - Network latency
  - Network fluctuations
  - And more



Birds-Eye View: Failure Detection

- **Node delay**
  - Saturation
  - Malfunction
  - Wrong dimensioning
- **Network issue**
  - Network latency
  - Network fluctuations
  - And more



Birds-Eye View: Failure Detection

Split-brain

- **PXC**
  - Odd number of nodes
  - Use of quorum and views
    - Use of weight
  - Node either in Primary Component or not able to serve

Dangerous Options:

- **pc.ignore_sb** (ignore splt-brain)
- **pc.ignore_quorum** (ignore quorum calculation)

- **GR**
  - Group base on number of nodes and views
  - Weight is for Primary election not for quorum calculation
  - Group quorum is base on majority of nodes
  - In case of expulsion a node can be:
    - ABORT_SERVER
    - OFFLINE_MODE
    - READ_ONLY

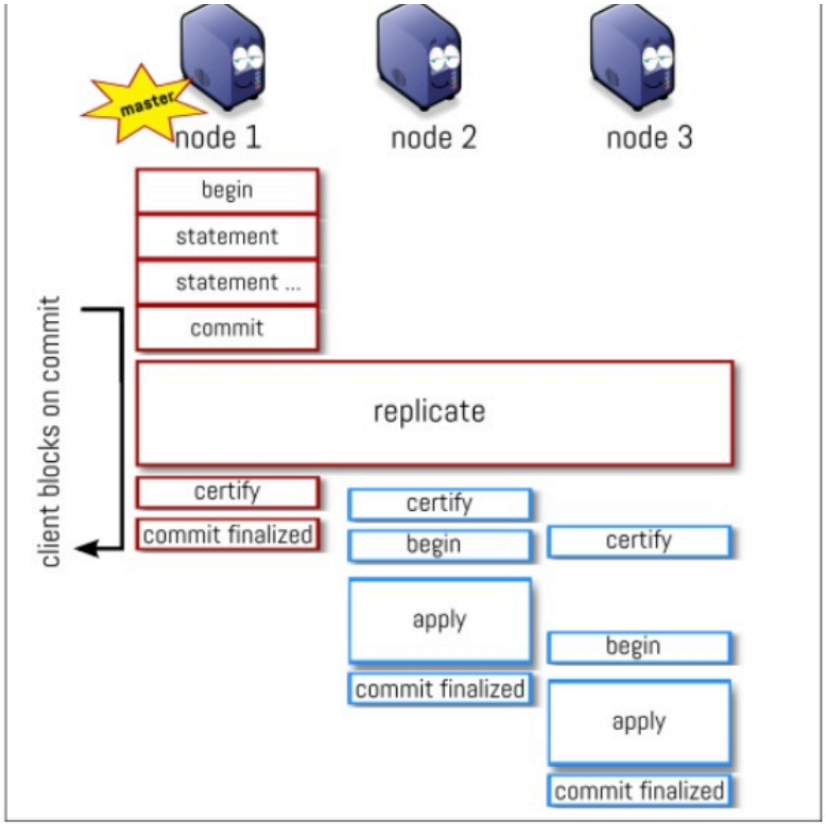
    (exit_state_action)

  - **group_replication_force_members**

For both the crucial component is the view and the existing members

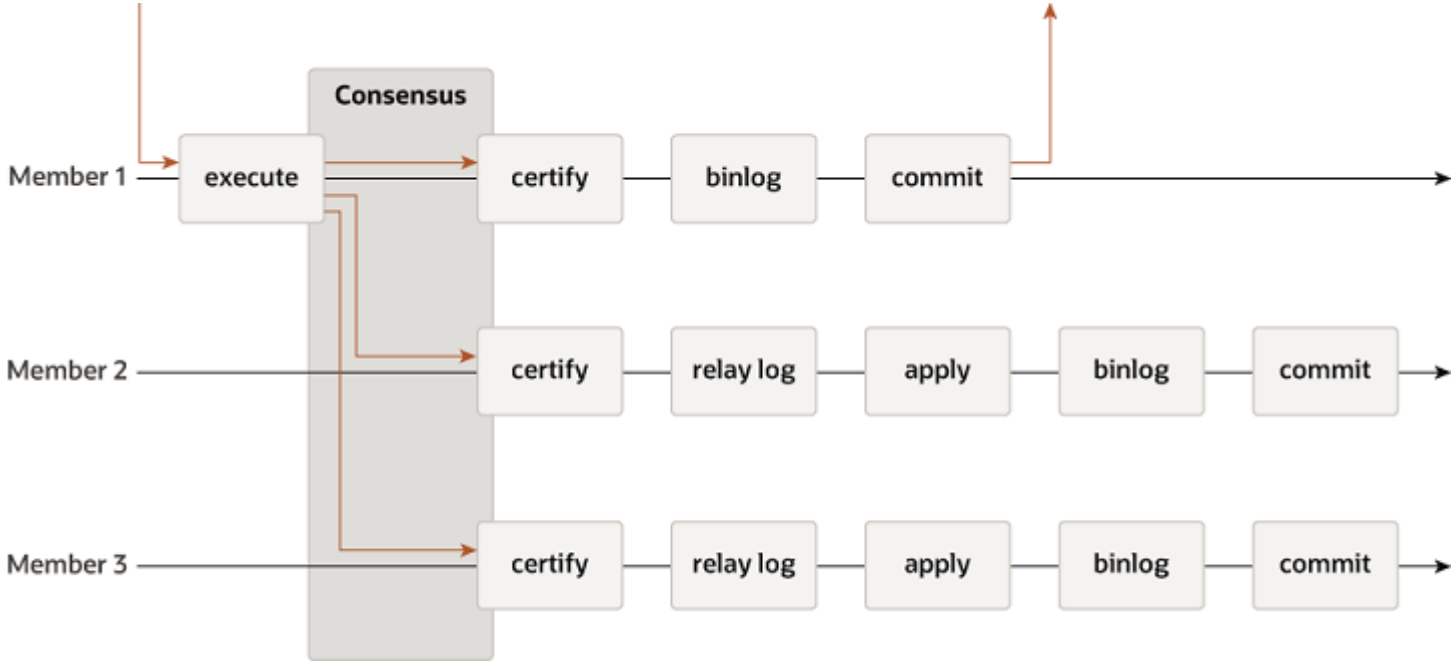
Transaction - the Apply flow

PXC



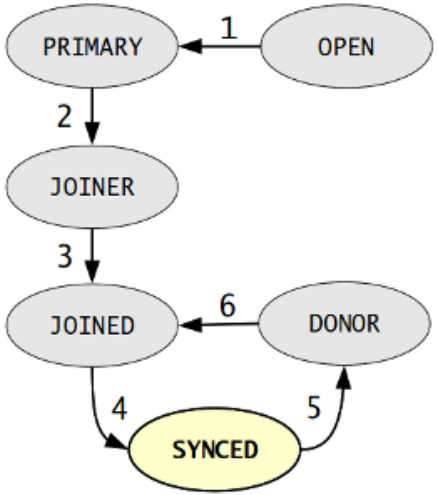
Transaction - the Apply flow

Group Replication



Birds-Eye View: Flow Control

- **Working against writeset pending in the receive queue**
  - No flow control
  - Writeset caching (joiner – Donor)
    - gcs.recv_q_hard_limit Maximum write-set cache size (in bytes).
    - gcs.max_throttle Smallest fraction to the normal replication rate the node can tolerate in the cluster.
    - gcs.recv_q_soft_limit Estimate of the average replication rate for the node
  - Catching up (Joined) Rate limited to what the joiner can apply
  - Cluster sync
    - gcs.fc_limit Used to determine the point where Flow Control engages.
    - gcs.fc_factor Used to determine the point where Flow Control disengages.



Birds-Eye View: Flow Control in Group Replication

Definitely more complicated and analytical, it analyzes things such as:

- The certifier queue size
- The replication applier queue size
- The total number of transactions certified
- The total number of remote transactions applied in the member
- The total number of local transactions

Variables

```
group_replication_flow_control_applier_threshold
group_replication_flow_control_certifier_threshold
group_replication_flow_control_member_quota_percent

group_replication_flow_control_hold_percent
group_replication_flow_control_release_percent

group_replication_flow_control_max_commit_quota
group_replication_flow_control_min_quota
group_replication_flow_control_min_recovery_quota

group_replication_flow_control_mode
group_replication_flow_control_period
```

Bird-Eye View: Message Fragmentation

- **PXC:**
  - For Load data 10K limit (configurable)
  - Streaming replication (by Session)
    - wsrep_trx_fragment_unit
    - wsrep_trx_fragment_size
- **GR:**
  - Supported from 8.0.16
  - All nodes must be able to support it
  - Managed at global level
    - group_replication_communication_max_message_size
  - Message delivery is complete when all nodes have the whole message
  - Node joining can recover message fragments from before it joins. Or it will be expelled

Data Consistency

What am I looking for?

The platform validate the consistency of each tuple that is replicated on each node, preventing any possible local deviation.

By inheritance data is the same on each node and cannot deviate.

- **Two problems**
  1. None of them provide a 100% data consistency guarantee
  2. PXC recently implement *Inconsistent voting*
  3. Both come with Defaults that also allow stale read (inconsistent data in read operations)

PXC:
Set wsrep_on=0;

GR:
Set sql_bin_log=0;

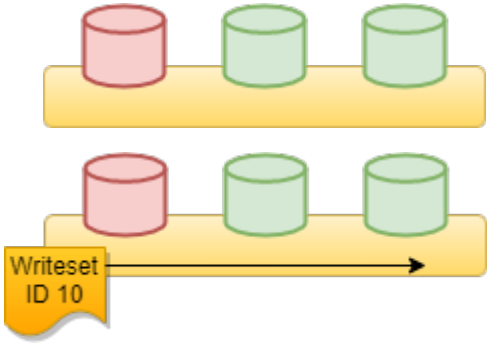
Data consistency: error during transactions

PXC use Cluster Error Voting

GR is based on the local conflict error

Data consistency: error during transactions

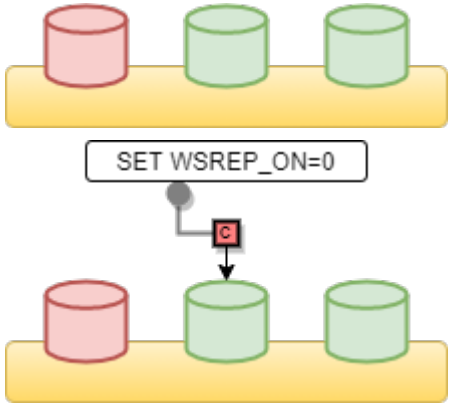
PXC use Cluster Error Voting



GR is based on the local conflict error

Data consistency: error during transactions

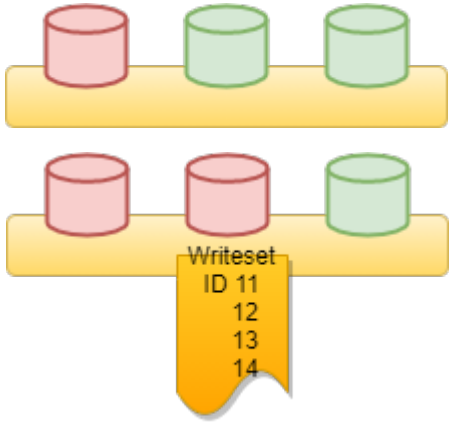
PXC use Cluster Error Voting



GR is based on the local conflict error

Data consistency: error during transactions

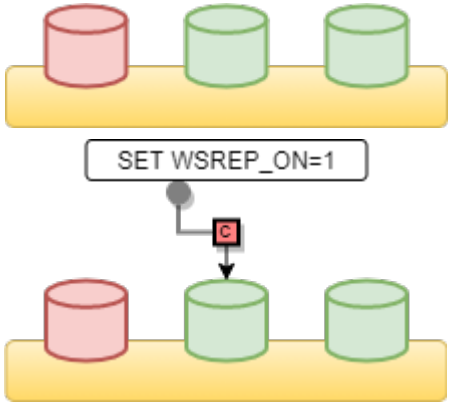
PXC use Cluster Error Voting



GR is based on the local conflict error

Data consistency: error during transactions

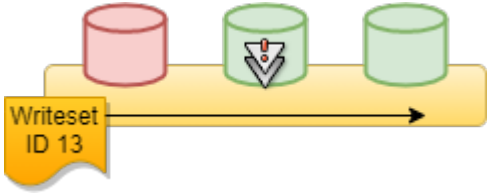
PXC use Cluster Error Voting



GR is based on the local conflict error

Data consistency: error during transactions

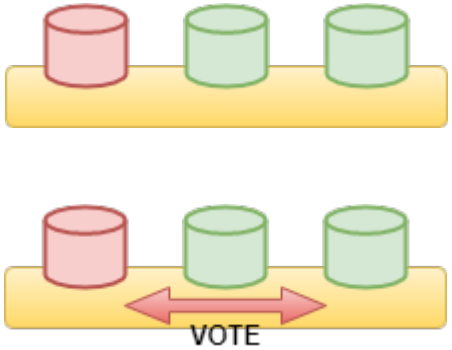
PXC use Cluster Error Voting



GR is based on the local conflict error

Data consistency: error during transactions

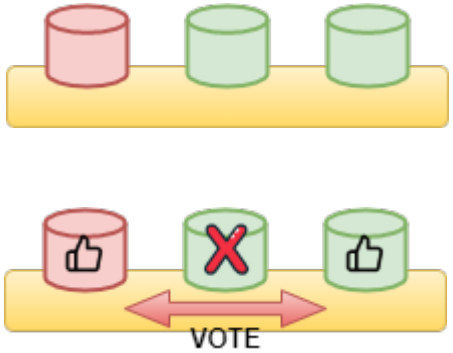
PXC use Cluster Error Voting



GR is based on the local conflict error

Data consistency: error during transactions

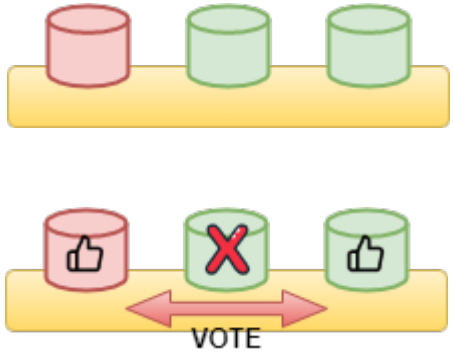
PXC use Cluster Error Voting



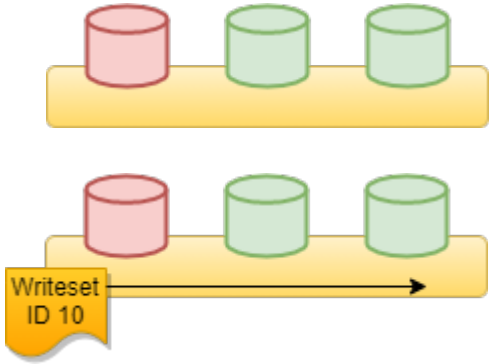
GR is based on the local conflict error

Data consistency: error during transactions

PXC use Cluster Error Voting

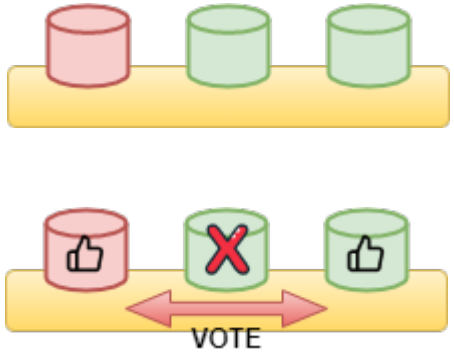


GR is based on the local conflict error

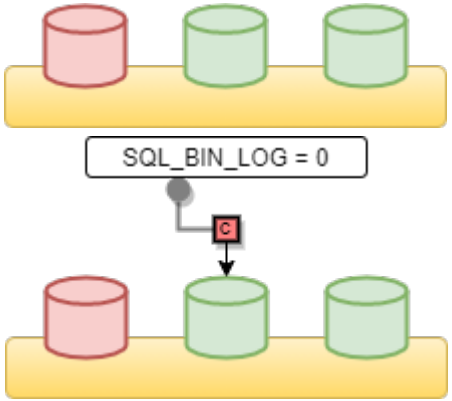


Data consistency: error during transactions

PXC use Cluster Error Voting

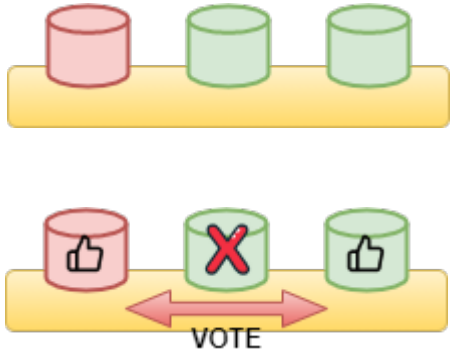


GR is based on the local conflict error

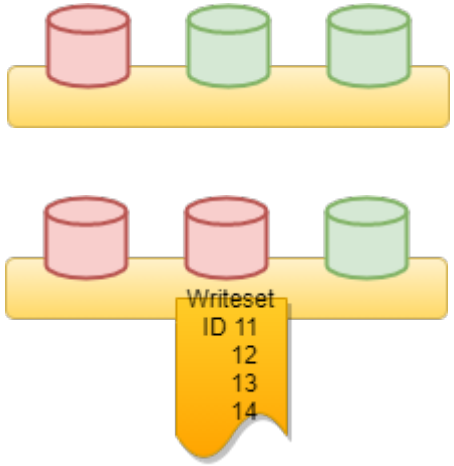


Data consistency: error during transactions

PXC use Cluster Error Voting

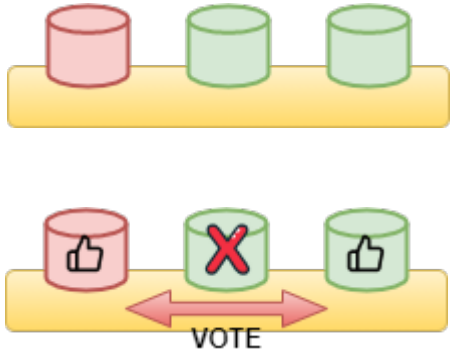


GR is based on the local conflict error

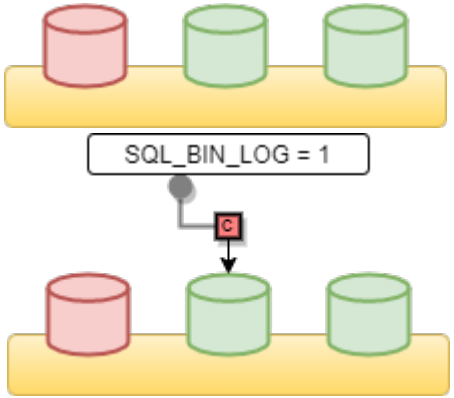


Data consistency: error during transactions

PXC use Cluster Error Voting

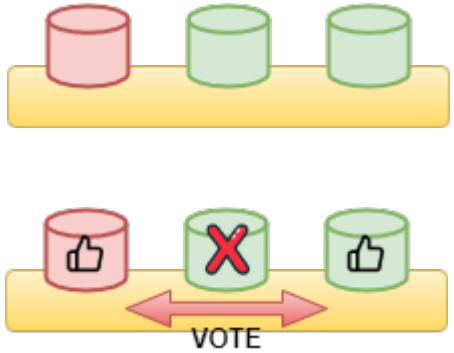


GR is based on the local conflict error

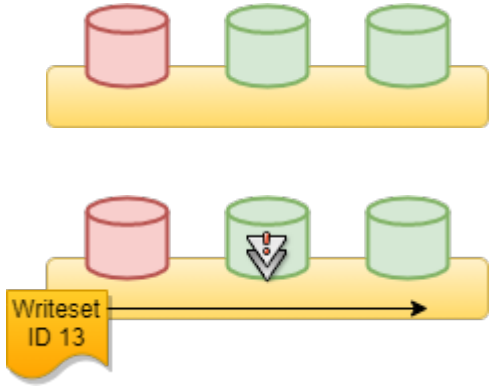


Data consistency: error during transactions

PXC use Cluster Error Voting

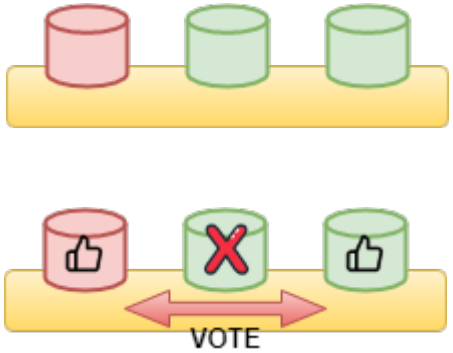


GR is based on the local conflict error

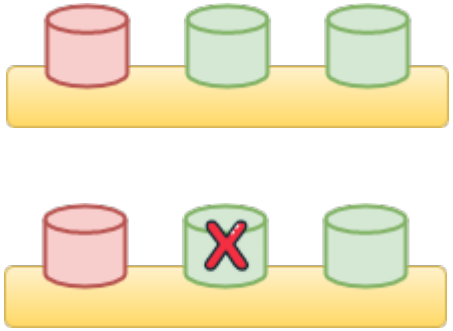


Data consistency: error during transactions

PXC use Cluster Error Voting



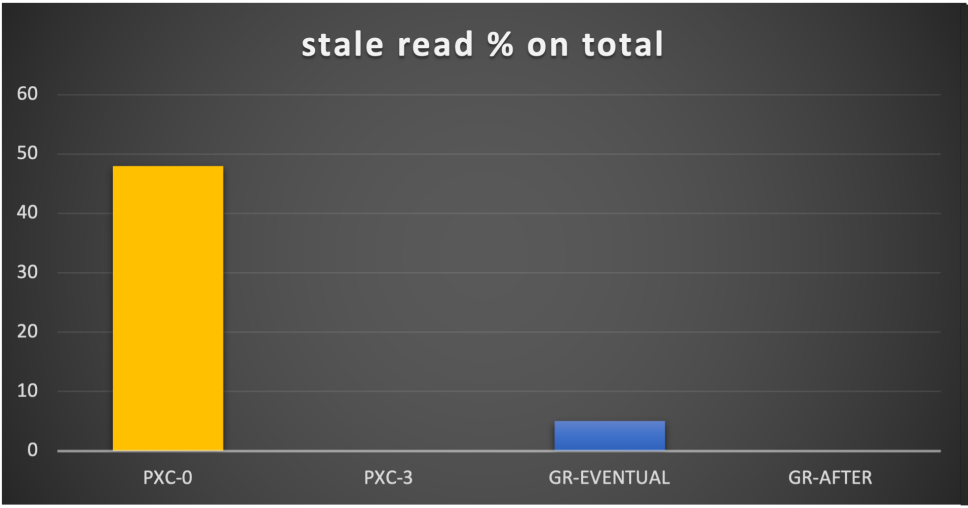
GR is based on the local conflict error

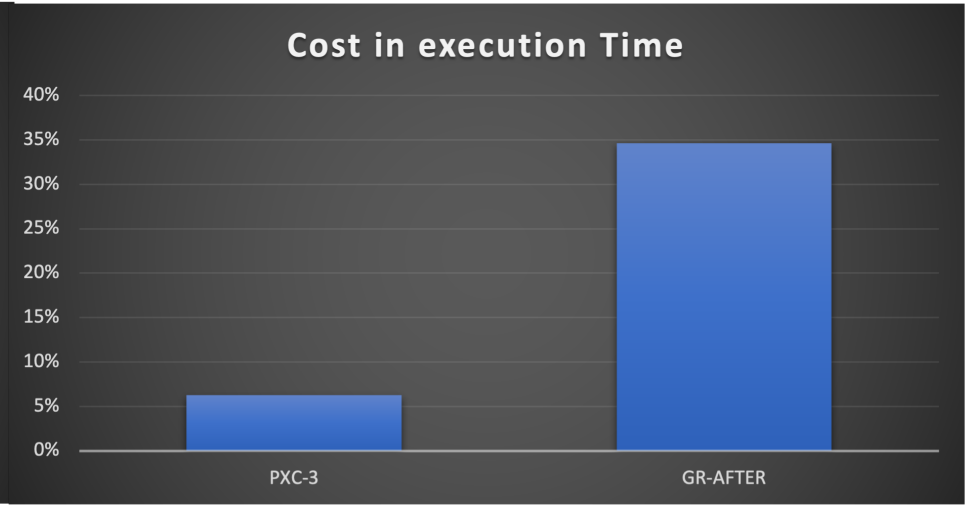


Data Consistency: Stale Reads

For full consistency change:

- **PXC →** `wsrep_sync_wait ≠ 0 (ie:3)`
- **GR →** group_replication_consistency `≠ EVENTUAL (ie: AFTER)`





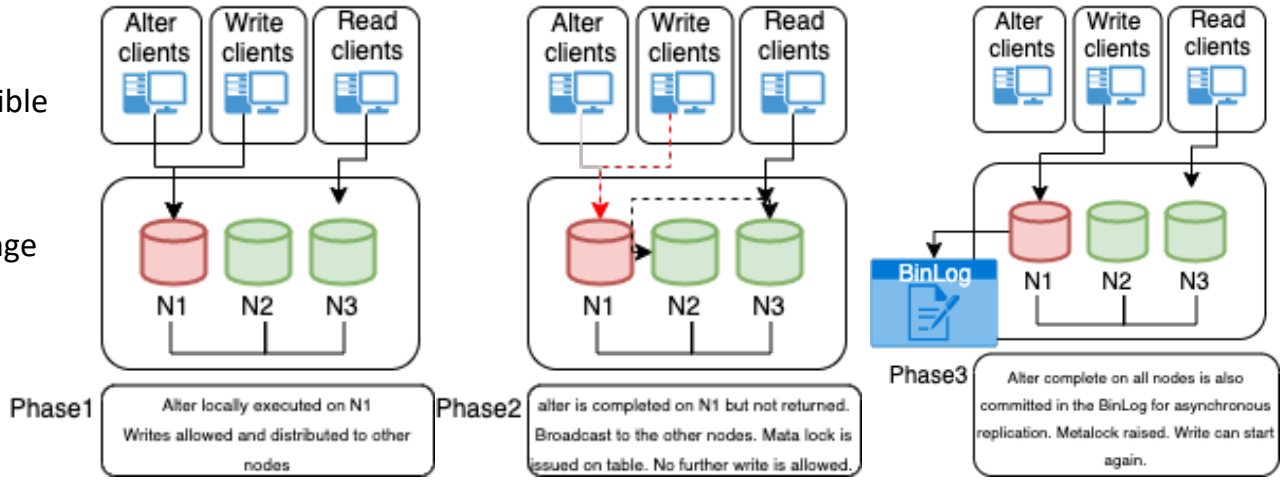
DDL

- **PXC**
  - TOI
    - Cluster unavailable for writes
  - RSU
    - Node temporary unavailable
    - May end up in different data by node
  - No online DDL
  - Not possible to kill DDL running
  - Significant history of issues due to DDL/locking/rollback
  - Manual RSU (https://www.percona.com/blog/2019/03/25/how-to-perform-compatible-schema-changes-in-percona-xtradb-cluster-advanced-alternative/)
  - Compatible VS incompatible changes
    - PT-OnlineSchemaChange

  https://jira.percona.com/browse/PXC-3645

DDL

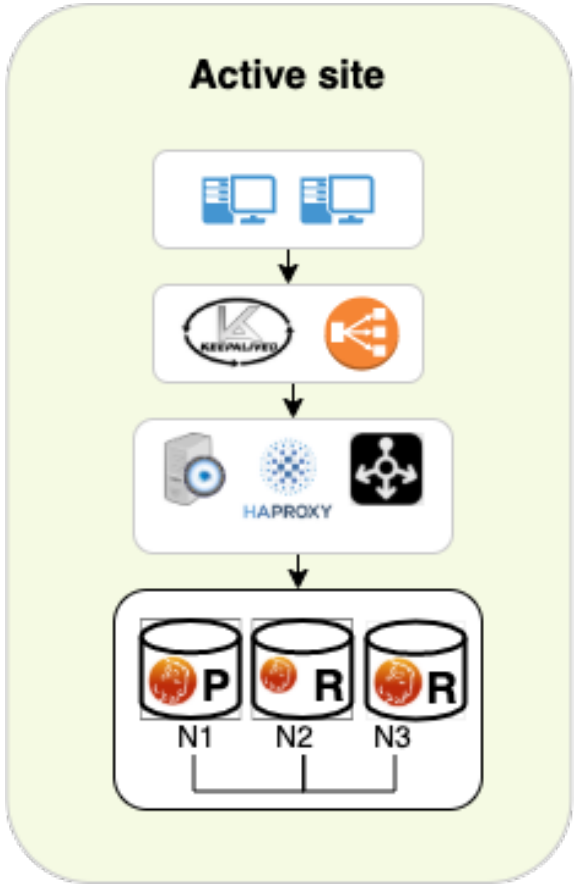
- **GR**
  - Standard alter
  - Online DDL
  - Compatible vs incompatible changes
    - PT-OnlineSchemaChange



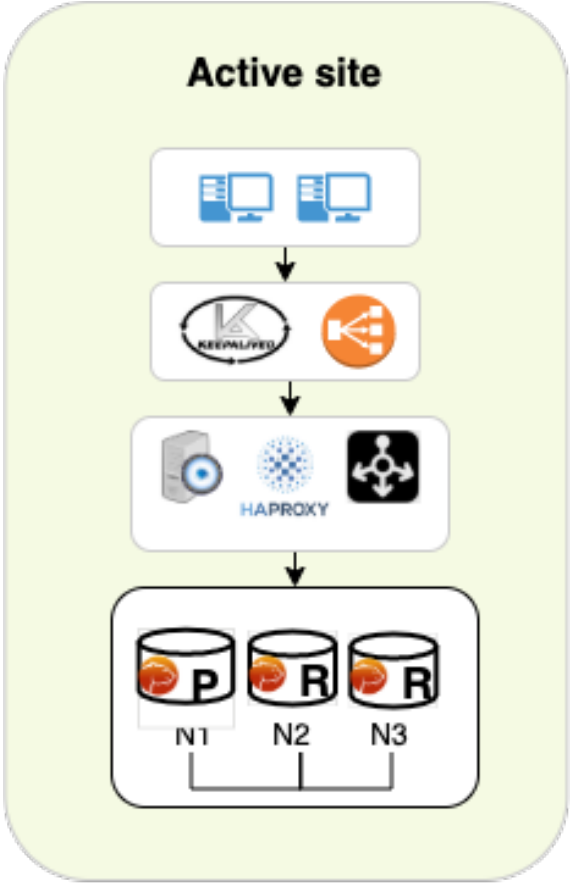
https://bugs.mysql.com/103421

HA Setup

PXC



GR

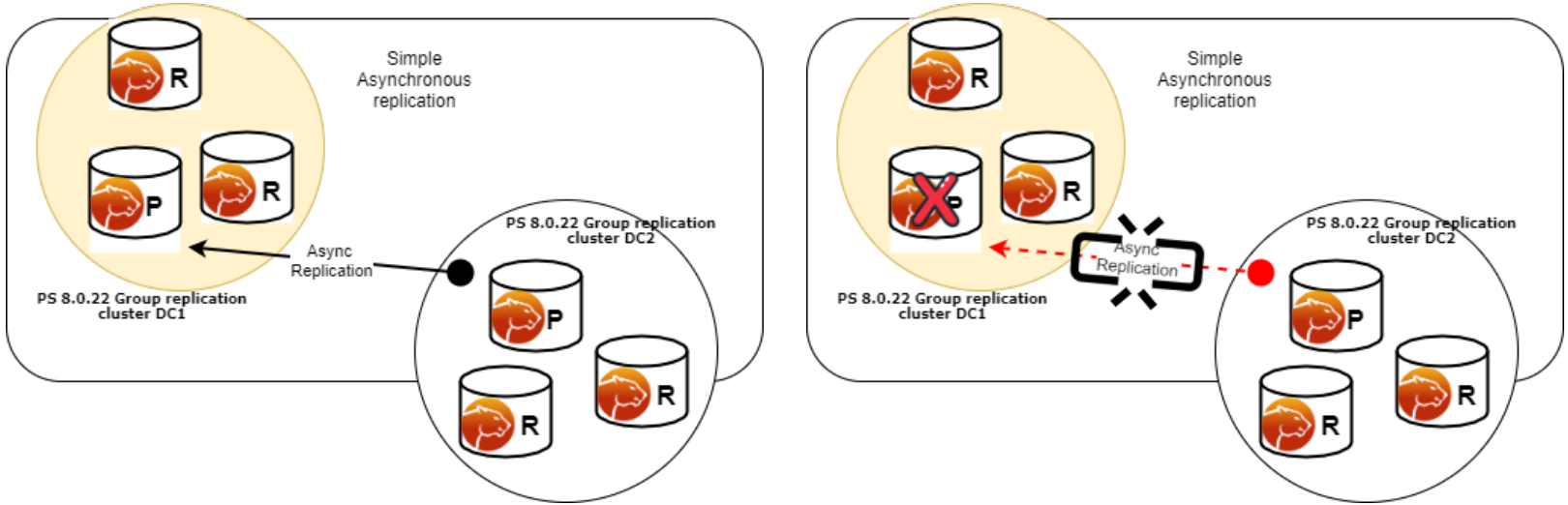


DR Claims

- **Both claim they can work on WAN natively**
- **Both then remark you need a very good network.**
  - A bit of contradiction here
- **I add you need to have the DCs too close to be a real DR**
  - Check out my previous presentation for Percona Live Europe about this
- **Long story short: DR cannot use Synchronous replication**

DR Case

- **The problem:**
  - Given Sync replication is not recommended, how we can keep the DR site up to date safely with Asynchronous replication?

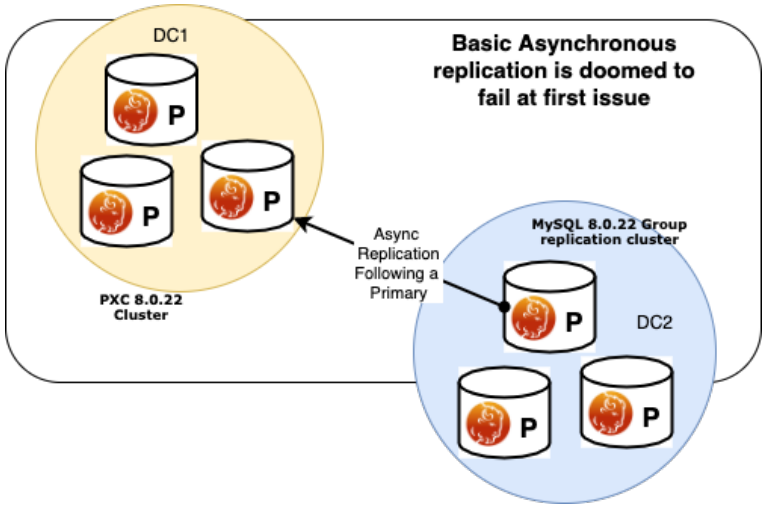


DR Solutions

- **PXC**
  - Basic Asynchronous replication
  - Async using Automatic failover is possible, but has few problems
    - No primary identification
    - No automatic fancing (like Read Only)
  - Only decent manager is PXC Replication Manager from Yves Trudeau
    - Deal with failover on Source and Replica
    - Allow to set order with weight
    - Allow multiple DCs

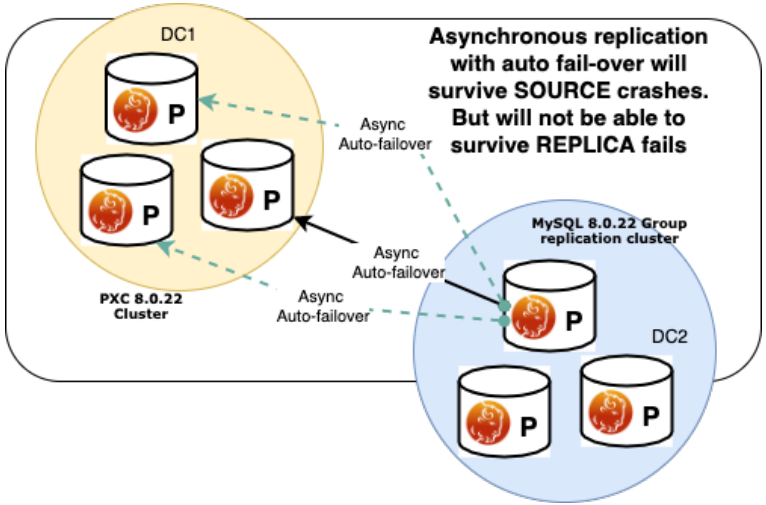
DR Solutions

- **PXC**
  - Basic Asynchronous replication
  - Async using Automatic failover is possible, but has few problems
    - No primary identification
    - No automatic fancing (like Read Only)
  - Only decent manager is PXC Replication Manager from Yves Trudeau
    - Deal with failover on Source and Replica
    - Allow to set order with weight
    - Allow multiple DCs



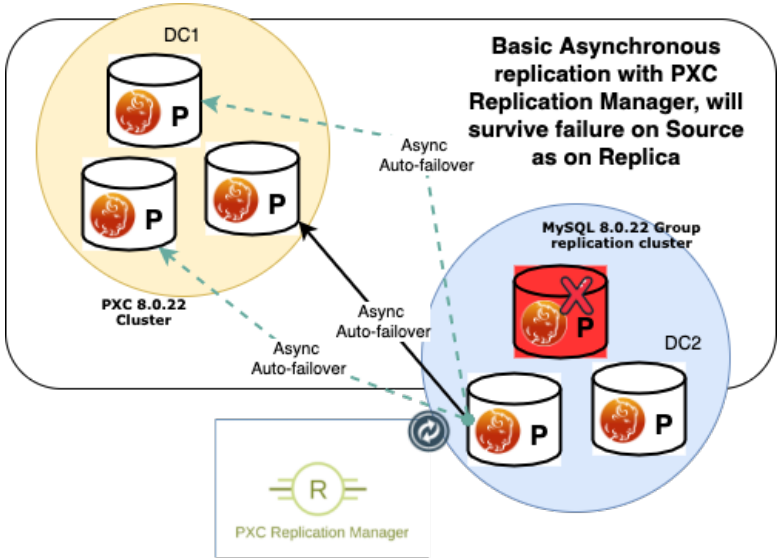
DR Solutions

- **PXC**
  - Basic Asynchronous replication
  - Async using Automatic failover is possible, but has few problems
    - No primary identification
    - No automatic fancing (like Read Only)
  - Only decent manager is PXC Replication Manager from Yves Trudeau
    - Deal with failover on Source and Replica
    - Allow to set order with weight
    - Allow multiple DCs



DR Solutions

- **PXC**
  - Basic Asynchronous replication
  - Async using Automatic failover is possible, but has few problems
    - No primary identification
    - No automatic fancing (like Read Only)
  - Only decent manager is PXC Replication Manager from Yves Trudeau
    - Deal with failover on Source and Replica
    - Allow to set order with weight
    - Allow multiple DCs



DR Solutions

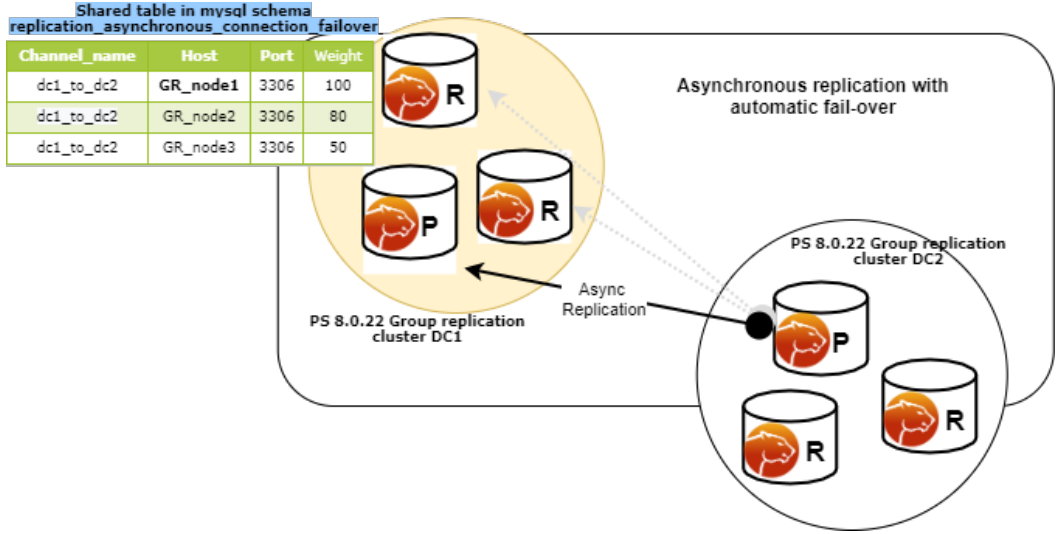
- **GR**
  - Async using Automatic failover
    - Primary identification by GR
    - All replicas are fenced (Read Only)
    - Add light weight tools as grFailOver for full coverage

DR Solutions

- **GR**
  - Async using Automatic failover
    - Primary identification by GR
    - All replicas are fenced (Read Only)
    - Add light weight tools as grFailOver for full coverage

Shared table in mysql schema replication_asynchronous_connection_failover

Channel_name	Host	Port	Weight
dc1_to_dc2	GR_node1	3306	100
dc1_to_dc2	GR_node2	3306	80
dc1_to_dc2	GR_node3	3306	50

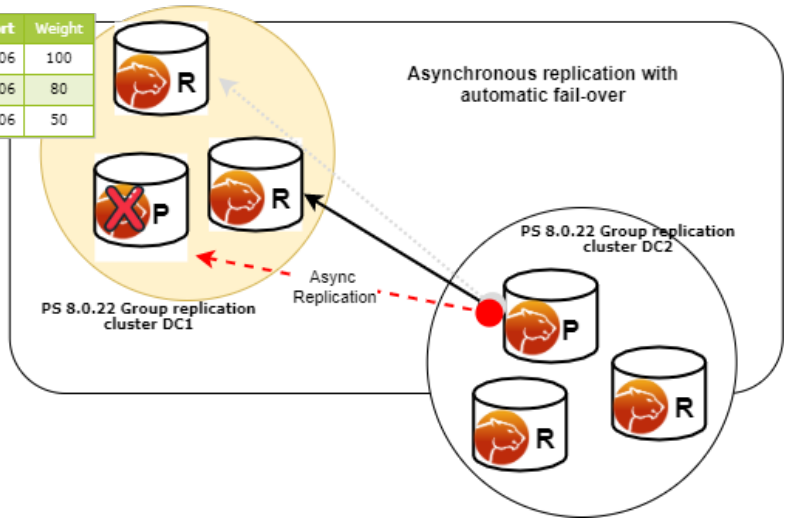


DR Solutions

- **GR**
  - Async using Automatic failover
    - Primary identification by GR
    - All replicas are fenced (Read Only)
    - Add light weight tools as grFailOver for full coverage

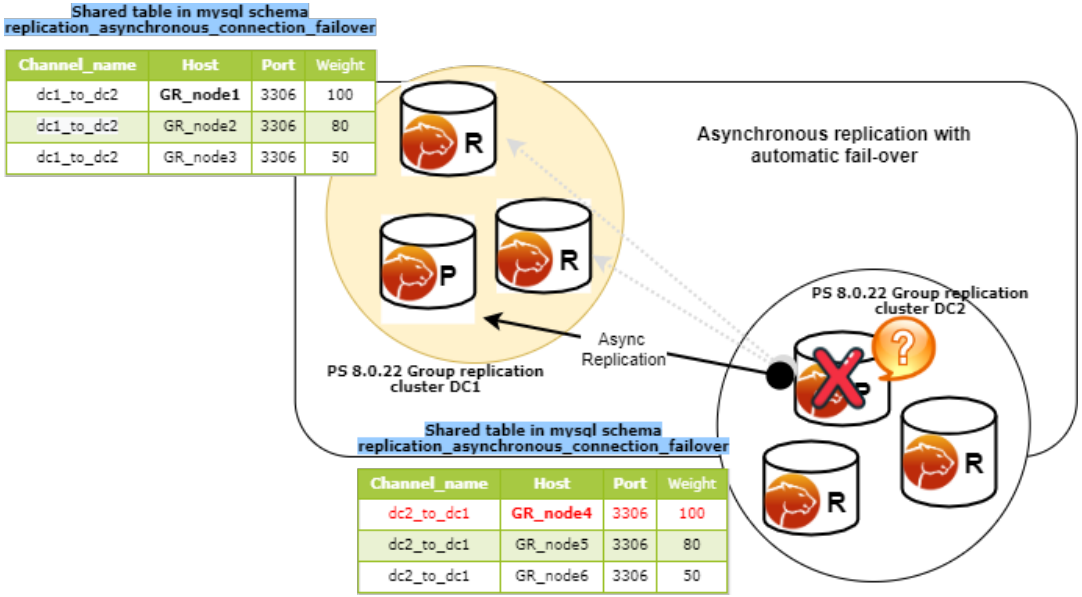
Shared table in mysql schema replication_asynchronous_connection_failover

Channel_name	Host	Port	Weight
dc1_to_dc2	GR_node1	3306	100
dc1_to_dc2	GR_node2	3306	80
dc1_to_dc2	GR_node3	3306	50



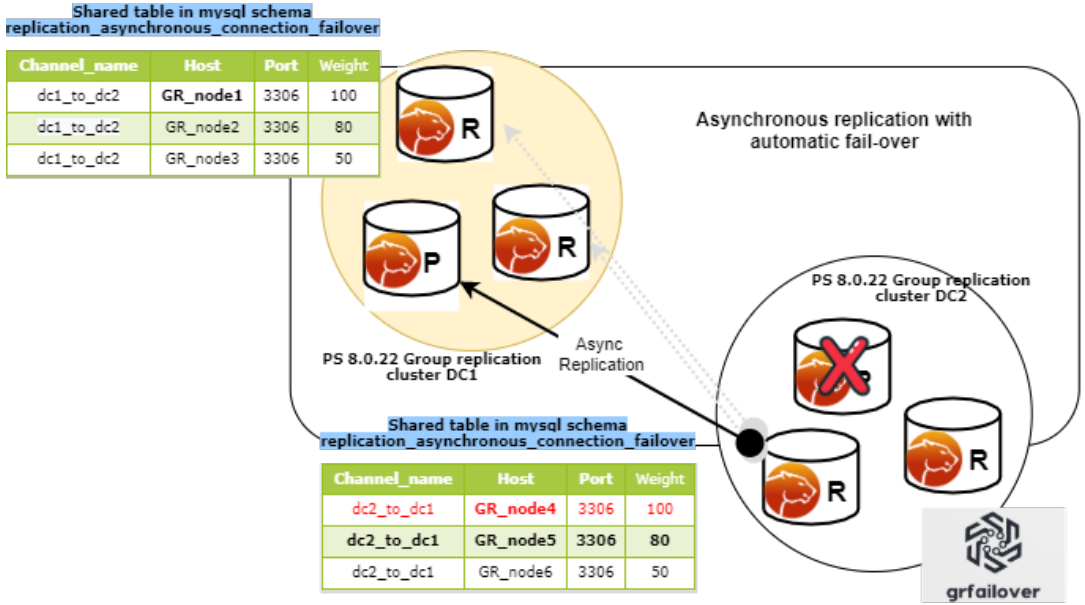
DR Solutions

- **GR**
  - Async using Automatic failover
    - Primary identification by GR
    - All replicas are fenced (Read Only)
    - Add light weight tools as grFailOver for full coverage



DR Solutions

- **GR**
  - Async using Automatic failover
    - Primary identification by GR
    - All replicas are fenced (Read Only)
    - Add light weight tools as grFailOver for full coverage



Adoption

PXC

- **is around ~13 years now**
  - Pros
    - Widely used as HA solution
    - Significant history of debugging issues
    - Very strong and consolidated knowledge
  - Cons
    - Galera (Codership) development moving to be more bound to MariaDB
      - IE: MariaDB GTID ; NBO/Black Box Enterprise only feature
    - Code is shared as tar
    - No visibility on the git repo
      - More difficult for others to interact and optimize the code

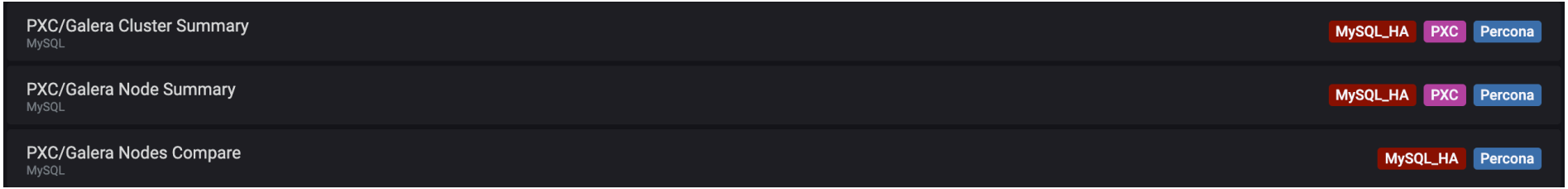
Adoption

GR

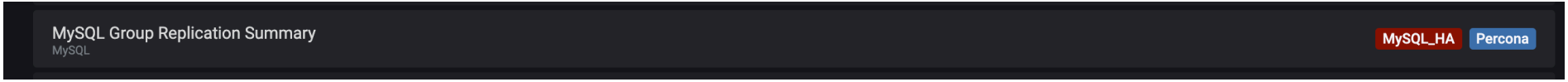
- **Pros**
  - Core function for MySQL/Oracle
  - Use well known elements like GTID and binlog
  - Knowledge is ramping up quickly (Percona training)
  - Code is available in github
- **Cons**
  - Relatively new
    - “Hello word” in 2014
    - First version really production ready was 8.0.20)
  - Limited adoption (well also PXC was not used until we start to…)
  - We all require more large scale deployments to be able to catch/fix deviations

Monitoring - PMM

- **PXC**
  - Extensive dashboards covering many different aspects of the cluster



- **GR**
  - New dashboard with a lot of initial information
  - Working on extending it with more advanced insights



Conclusions

- In Percona we support both PXC and PS with Group Replication
- As architecture components the two solutions are very similar
- GR is more articulated in DDL execution and Flow Control
- Consistency like avoid stale read has less impact in PXC
- The set of tooling that comes with MySQL Shell is not available with PXC
- In my opinion, in the future we will see the use of GR growing a lot, but we need to work on packaging it better, especially on automating it and more in-depth monitoring
- The more adoption we will see for GR, the better the product will become due the feedback
- PXC is still a strong HA solution, and it is still playing a key role in design strong High Available architectures

Any questions?

You can find me at:

- Marco.tusa@percona.com
- @marcotusa