
Synchronous multi-master clusters with MySQL: an introduction to Galera

Henrik Ingo

OUGF Harmony conference
Aulanko, 2012-05-31

Please share and reuse this presentation
licensed under Creative Commons Attribution license
<http://creativecommons.org/licenses/by/3.0/>

codership

Agenda

- * MySQL replication issues
 - * Galera internal architecture
 - * Installing and configuring it
 - * Synchronous multi-master clustering, what does it mean?
 - * Load balancing and other options
 - * How network partitioning is handled
 - * WAN replication
- How does it perform?
 - * In memory workload
 - * Scale-out for writes - how is it possible?
 - * Disk bound workload
 - * NDB shootout

So what is Galera all about?

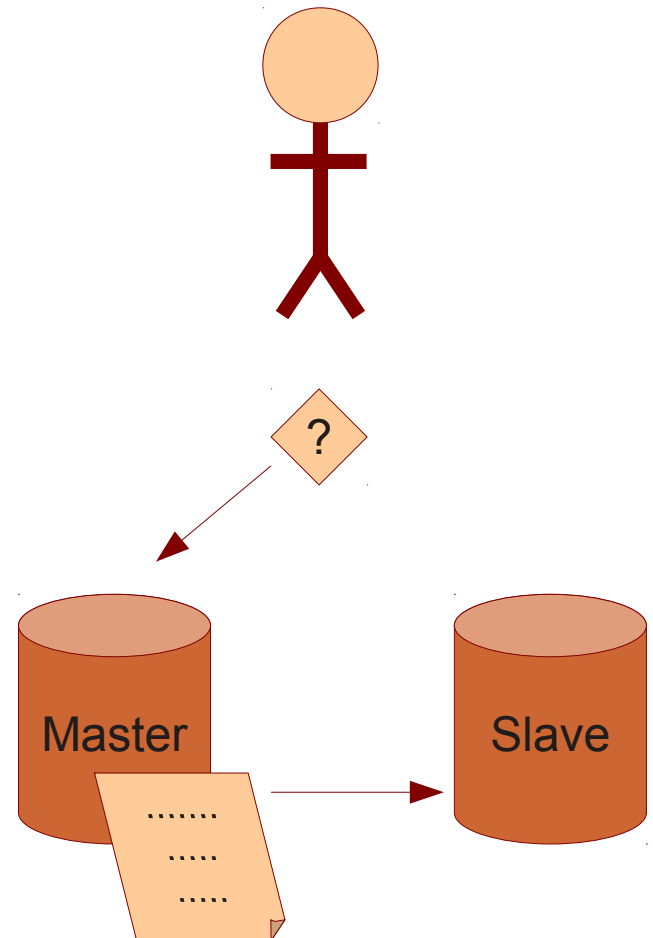
codership

- Participated in 3 MySQL cluster developments, since 2003
- Started Galera work 2007
- Galera is free, open source. Codership offers support and consulting
- Percona XtraDB Cluster based on Galera, launched 2012
- Is (and can be) integrated into other MySQL and non-MySQL products

codership

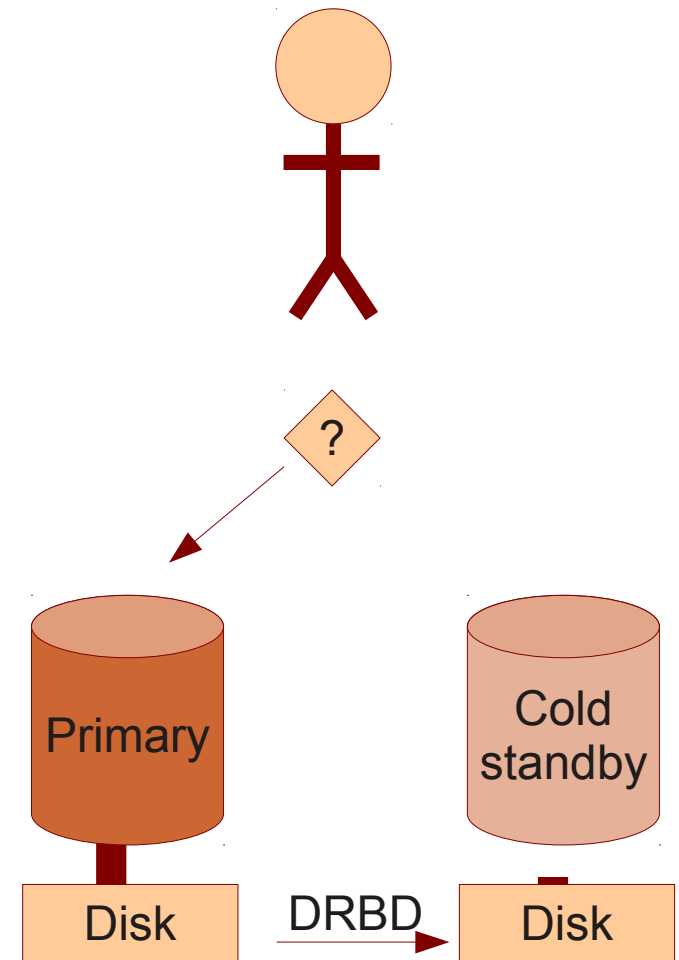
MySQL replication challenges

- Asynchronous = You will lose data
- MySQL 5.5 semi-sync: Better, but falls back to asynchronous when in trouble...
- Single-threaded => slave lag => 50-80% performance penalty
- Master-slave: read-write splitting, failovers, diverged slaves
- Low level: manually provision DB to slave, configure binlog positions, ...



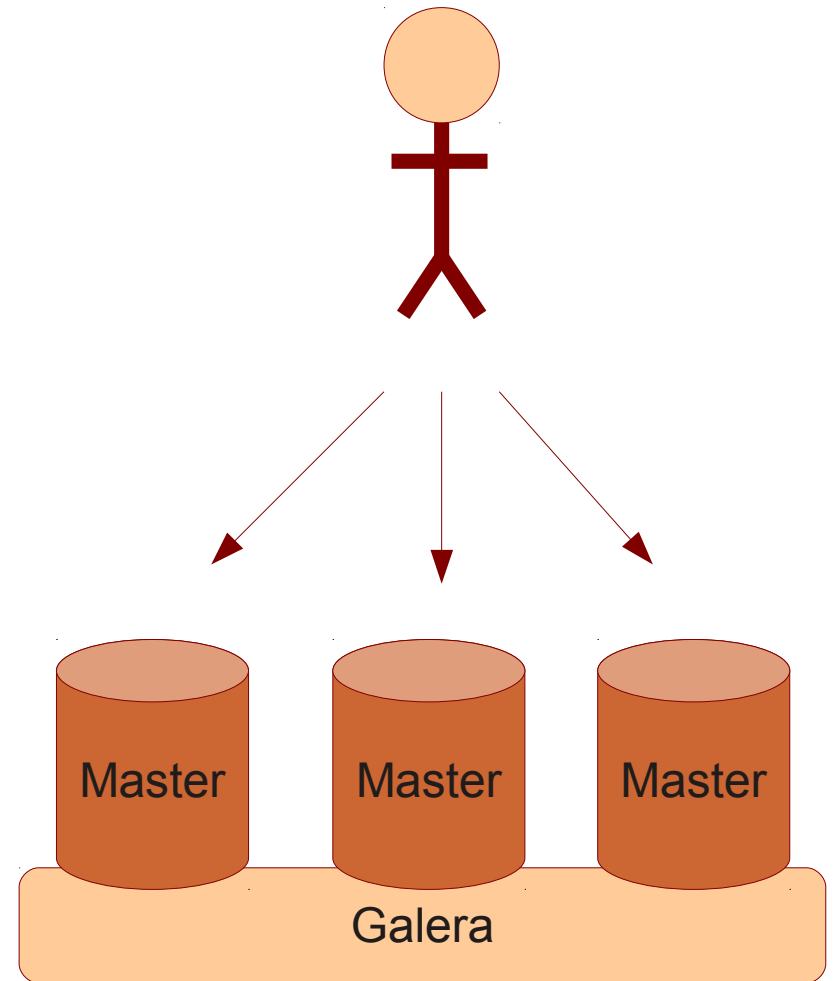
So what about DRBD?

- Synchronous, yay!
- Cold standby
- No scale-out
(But you can combine with MySQL replication...)
- 50% performance penalty
- (SAN based HA has these same issues btw)



Galera in a nutshell

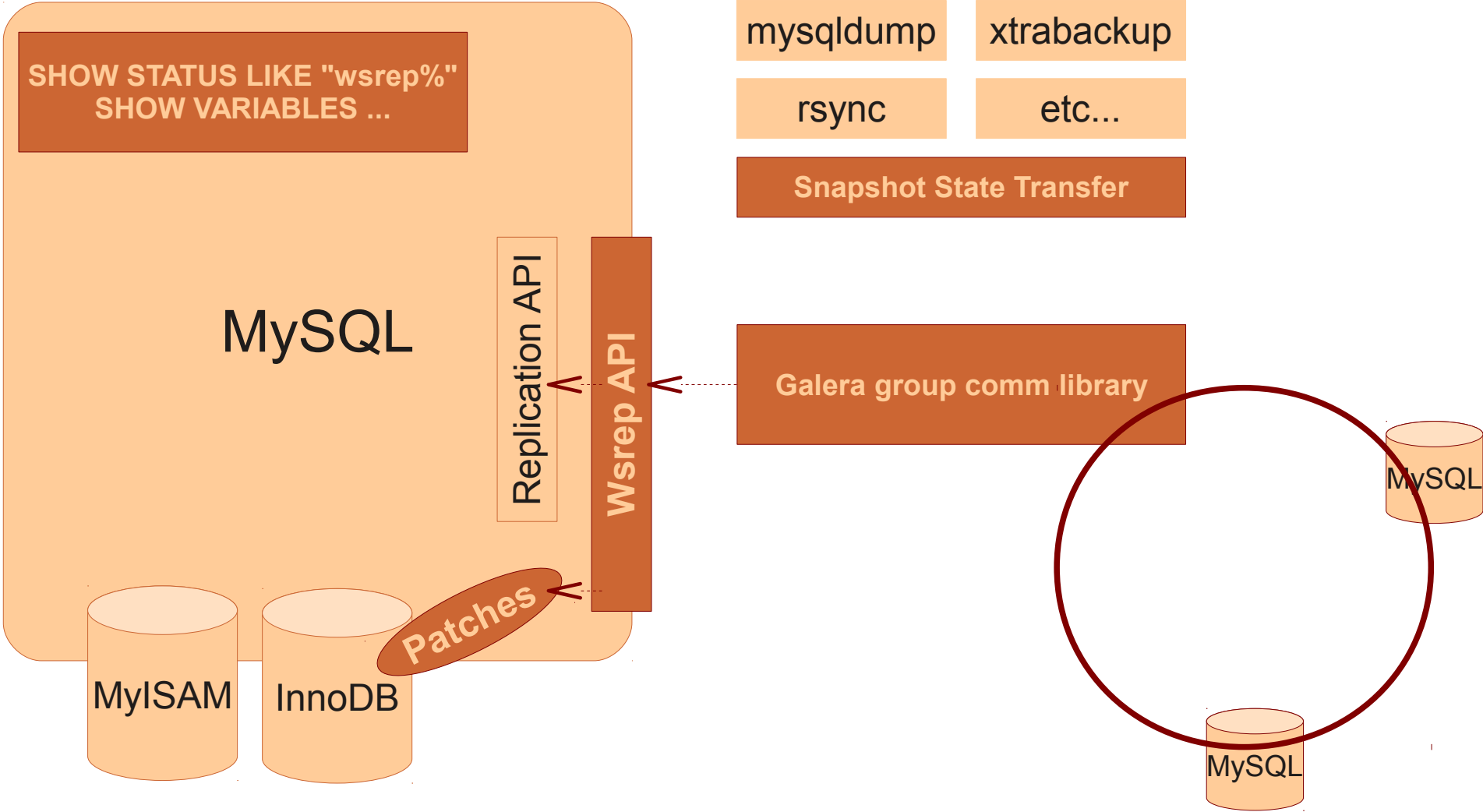
- True multi-master:
Read & write to any node
- Synchronous replication
- No slave lag
- No integrity issues
- No master-slave failovers or
VIP needed
- Multi-threaded slave, no
performance penalty
- Automatic node provisioning



Ok, let's take a closer look...

codership

A MySQL Galera cluster is...



<http://www.codership.com/downloads/download-mysqldgalera>

Starting your first cluster

Lets assume we have nodes 10.0.0.1, 10.0.0.2 and 10.0.0.3.

On node 1, set this in my.cnf:

```
wsrep_cluster_address="gcomm://"
```

Then start the first node:

```
/etc/init.d/mysql start
```

Important! Now change this in my.cnf to point to any of the other nodes. Don't leave it with empty gcomm string!

```
wsrep_cluster_address="gcomm://10.0.0.3"
```

On node 2, set this in my.cnf:

```
wsrep_cluster_address="gcomm://10.0.0.1"
```

And start it

```
/etc/init.d/mysql start
```

Node 2 now connects to Node 1. Then do the same on Node 3:

```
wsrep_cluster_address="gcomm://10.0.0.1" # or .2
```

```
/etc/init.d/mysql start
```

Node 3 connects to node 1, after which it establishes communication with all nodes currently in the cluster.

wsrep_cluster_address should be set to any one node that is currently part of the cluster. After startup, it doesn't have any meaning, all nodes connect to all others all the time.

Tip: Galera outputs a lot of info to MySQL error log, especially when nodes join or leave the cluster.

Use one of these to observe SST:

```
ps aux|grep mysqldump  
ps aux|grep xtrabackup  
ps aux|grep rsync
```

Checking that nodes are connected and ready

```
SHOW STATUS LIKE "wsrep%";
```

```
...
```

```
wsrep_local_state          4  
wsrep_local_state_comment Synced (6)
```

```
...
```

```
wsrep_cluster_conf_id     54  
wsrep_cluster_size        3  
wsrep_cluster_state_uuid  3108998a-67d4-11e1-...  
wsrep_cluster_status      Primary
```

Increments when a node joins or leaves the cluster.

Nr of nodes connected.

Primary or Non-Primary cluster component?

UUID generated when first node is started, all nodes share same UUID. Identity of the clustered database.

Q: Can you tell me **which** nodes are connected to this node/component?

A: See MySQL error log.

Your first query

```
# mysql -uroot -prootpass -h 10.0.0.1
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.1.53 wsrep_0.8.0

mysql> create table t (k int primary key auto_increment, v blob);

mysql> show create table t;
| Table | Create Table |
| t | CREATE TABLE `t` (
`k` int(11) NOT NULL auto_increment,
`v` blob,
PRIMARY KEY (`k`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
1 row in set (0.00 sec)

mysql> insert into t (k) values ( "first row");

mysql> insert into t (k) values ( "second row");
```


How it looks from another node

```
# mysql -uroot -prootpass -h 10.0.0.2
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.1.53 wsrep_0.8.0
```

```
mysql> show create table t;
| Table | Create Table |
| t | CREATE TABLE `t` (
`k` int(11) NOT NULL auto_increment,
`v` blob,
PRIMARY KEY (`k`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 |
```

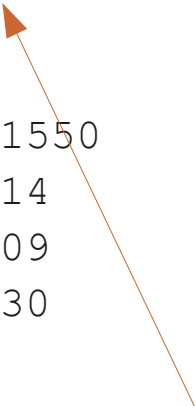
```
mysql> select * from t;
| k | v |
| 1 | first row |
| 4 | second row |
```

Galera automatically sets
auto_increment_increment
and auto_increment_offset



More interesting Galera (Wsrep) status vars

```
| wsrep_local_send_queue      | 34 |
| wsrep_local_send_queue_avg | 0.824589 |
| wsrep_local_recv_queue     | 30 |
| wsrep_local_recv_queue_avg | 1.415460 |
| wsrep_flow_control_paused | 0.790793 |
| wsrep_flow_control_sent    | 9 |
| wsrep_flow_control_recv    | 52 |
| wsrep_cert_deps_distance   | 105.201550 |
| wsrep_apply_oooe           | 0.349014 |
| wsrep_apply_ool           | 0.012709 |
| wsrep_apply_window         | 2.714530 |
```



Tip: These variables reset every time you do SHOW VARIABLES. So do it once, wait 10 sec, then do it again and read the second values.

Fraction of time that Galera replication was halted due to slave(s) overloaded. Bigger than 0.1 means you have a problem!

Did you say parallel slave threads?

```
mysql> show variables\G
....
***** 296. row *****
Variable_name: wsrep_slave_threads
Value: 32

mysql> SHOW PROCESSLIST;
...
| 31 | system user | | NULL | Sleep | 32 | committed 933 | NULL
| 32 | system user | | NULL | Sleep | 21 | committed 944 | NULL
```

- MySQL replication allows master to proceed and leaves slave lagging.
- Galera cluster's are tightly coupled: Master throughput will degrade if any slave can't keep up.
- For best throughput, use 4-64 slave threads. (Esp disk-bound workloads.)
- Generic solution:
 - works with any application, schema.
 - Commit order is preserved.
 - Also possible: Out-of-order-commits. Small additional benefit, unsafe for most apps.

MySQL options with Galera friendly values

```
# (This must be substituted by wsrep_format)
binlog_format=ROW

# Currently only InnoDB storage engine is supported
default-storage-engine=innodb

# No need to sync to disk when replication is synchronous
sync_binlog=0
innodb_flush_log_at_trx_commit=0
innodb_doublewrite=0

# to avoid issues with 'bulk mode inserts' using autoinc
innodb_autoinc_lock_mode=2

# This is a must for parallel applying
innodb_locks_unsafe_for_binlog=1

# Query Cache is not supported with wsrep
query_cache_size=0
query_cache_type=0
```


Setting WSREP and Galera options

```
# Most settings belong to wsrep api = part of MySQL
#
# State Snapshot Transfer method
wsrep_sst_method=mysqldump
#
# SST authentication string. This will be used to send SST to joining
nodes.
# Depends on SST method. For mysqldump method it is root:<root password>
wsrep_sst_auth=root:password

# Use of Galera library is opaque to MySQL. It is a "wsrep provider".
# Full path to wsrep provider library or 'none'
#wsrep_provider=none
wsrep_provider=/usr/local/mysql/lib/libgalera_smm.so

# Provider specific configuration options
# Here we increase window size for a WAN setup
wsrep_provider_options="evs.send_window=512; evs.user_send_window=512;"
```

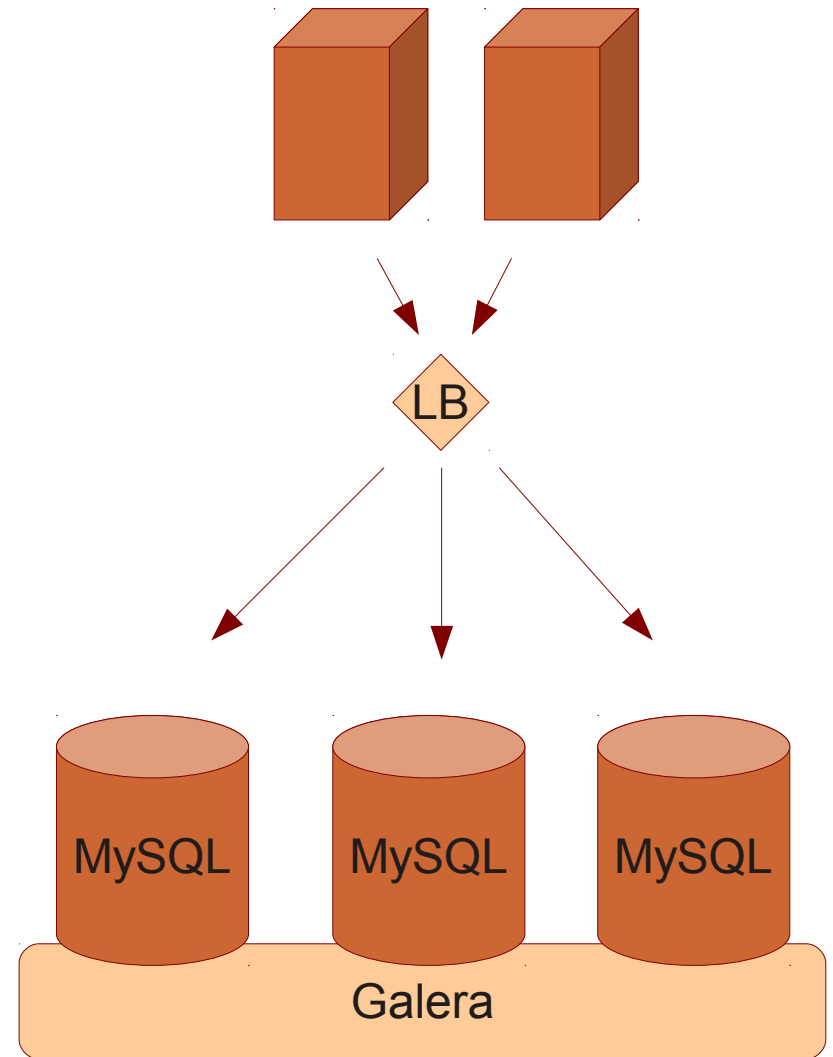
Synchronous Multi-Master Clustering

codership

Multi-master = you can write to any node & no failovers needed

What do you mean no failover???

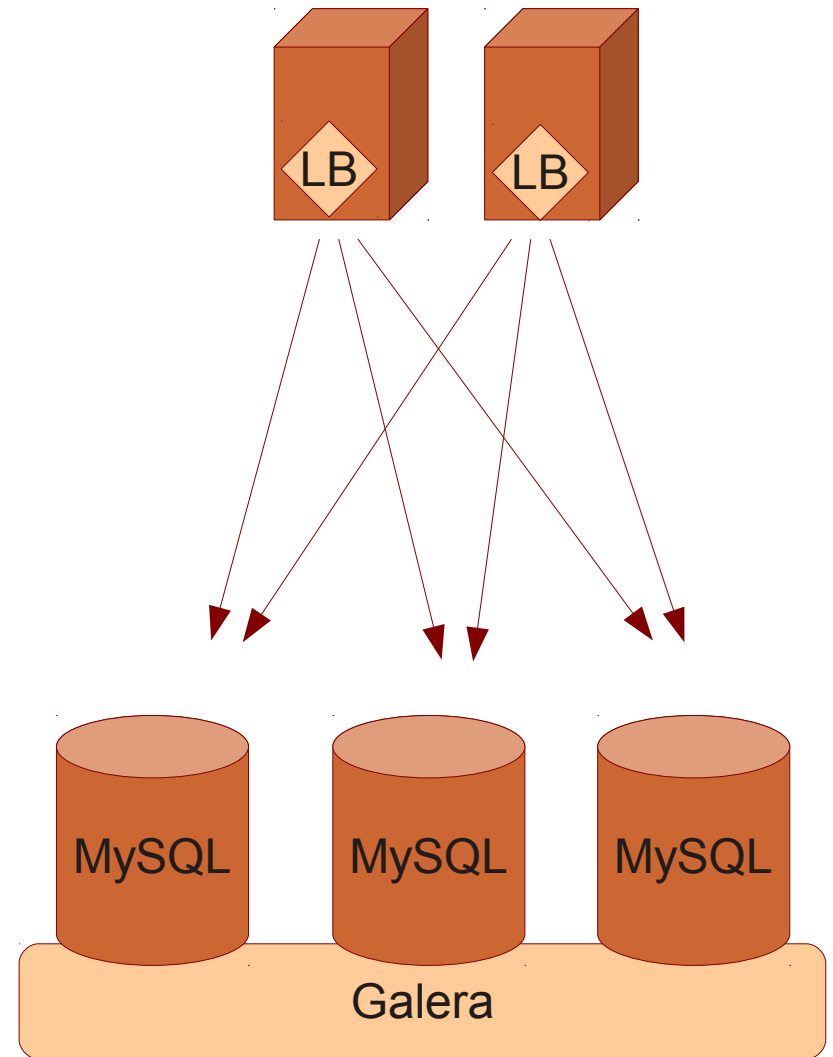
- Use a load balancer
- Application sees just one IP
- Write to any available node, round-robin
- If node fails, just write to another one
- What if load balancer fails?
-> Turtles all the way down



Protip: JDBC, PHP come with built-in load balancing!

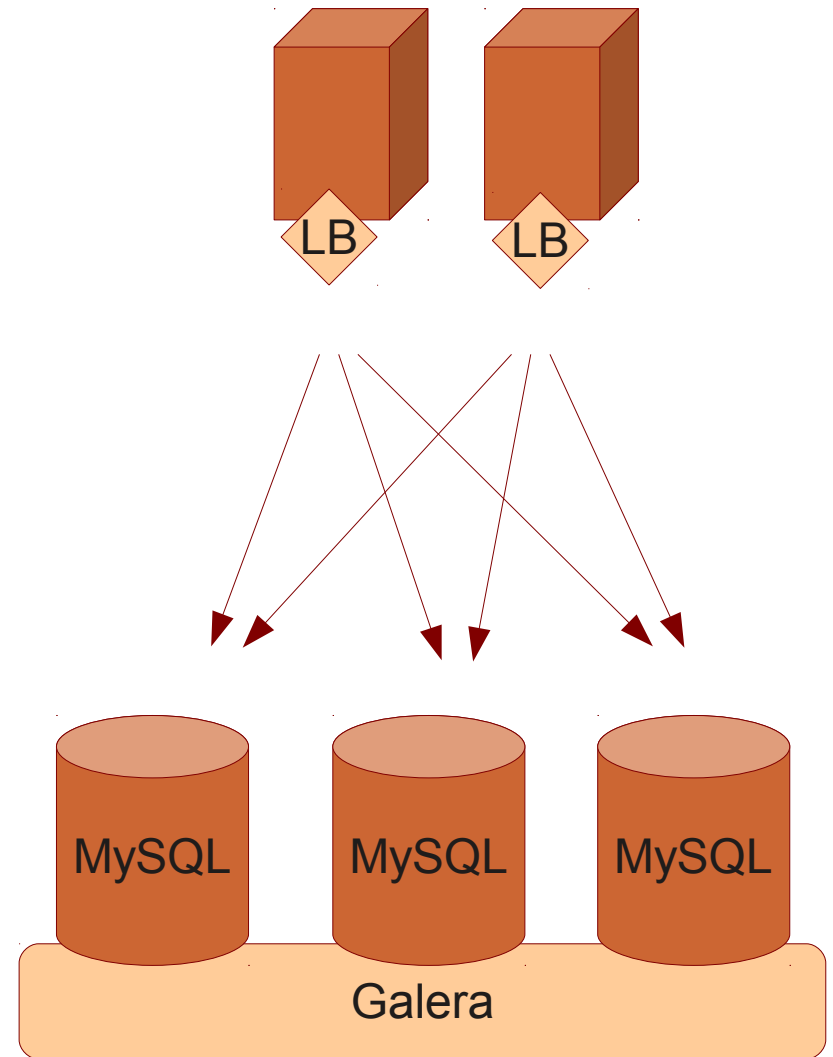
- No Single Point of Failure
- One less layer of network components
- Is aware of MySQL transaction states and errors

```
jdbc:mysql:loadbalance://  
10.0.0.1,10.0.0.2,10.0.0.3  
/<database>?  
loadBalanceBlacklistTimeout=5000
```

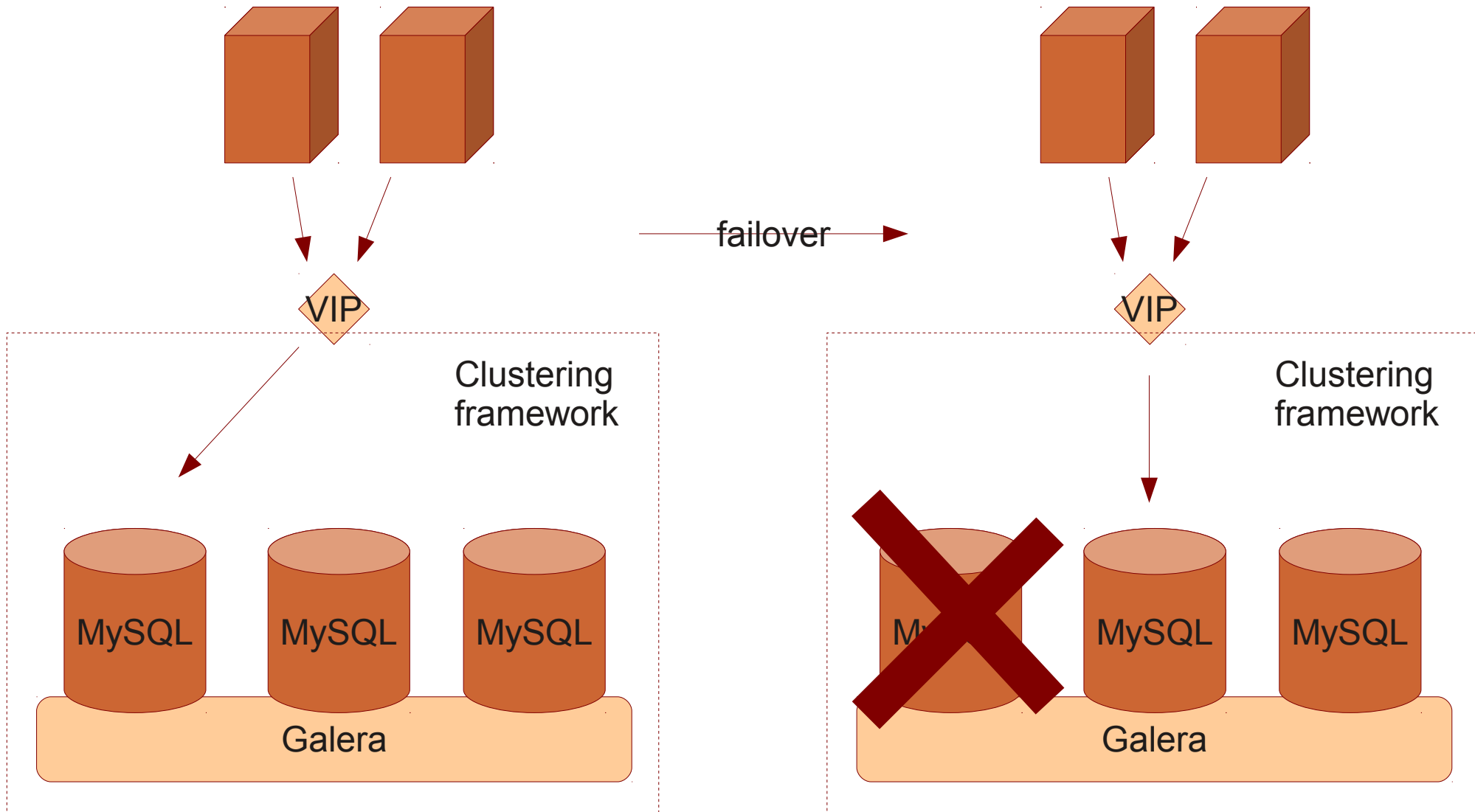


Load balancer per each app node

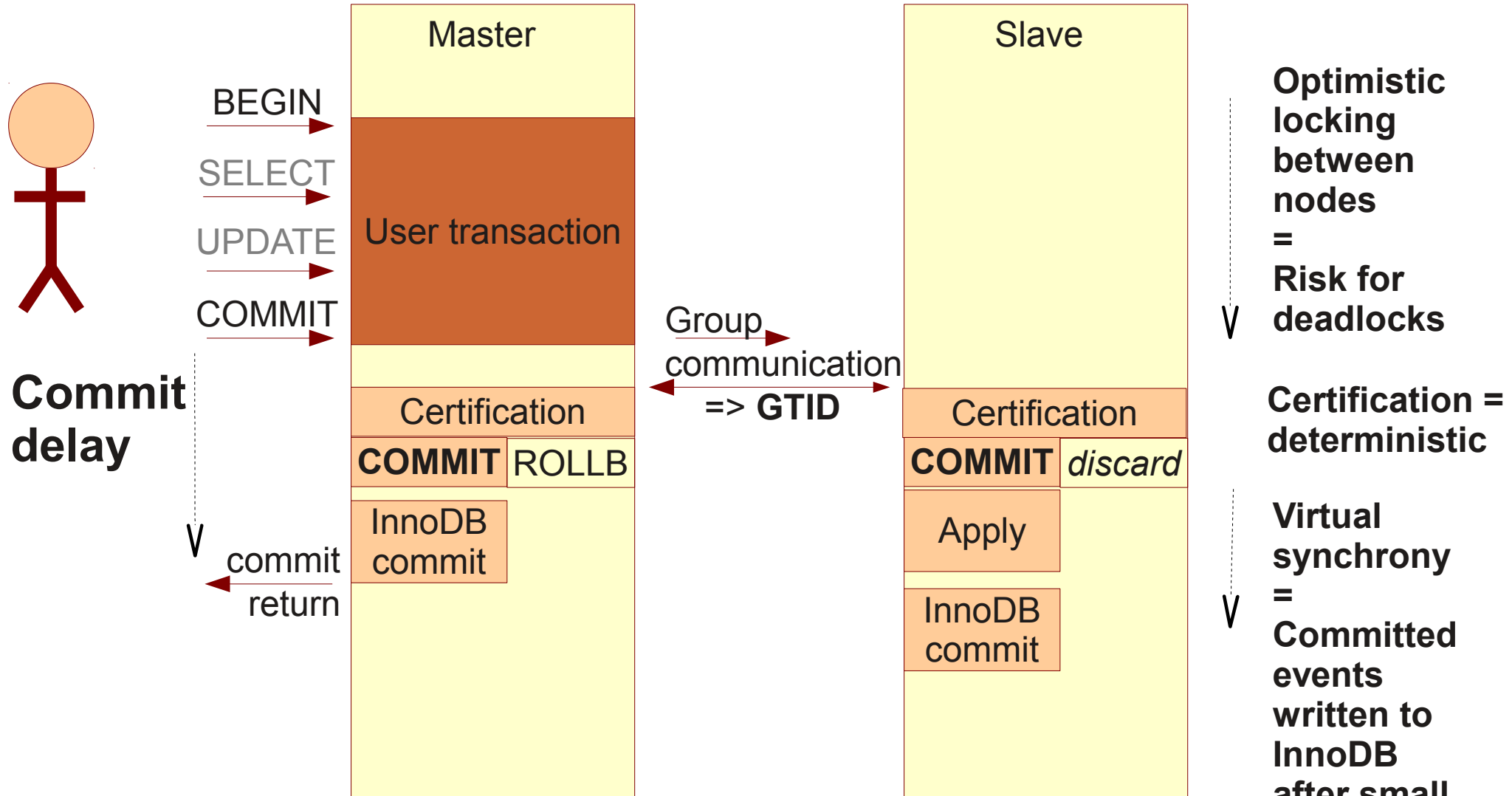
- Also no Single Point of Failure
- LB is an additional layer, but localhost = pretty fast
- Need to manage more load balancers
- Good for languages other than Java, PHP



You can still do VIP based failovers. But why?
=> You can run a 2 node Galera cluster like this.



Understanding the transaction sequence in Galera

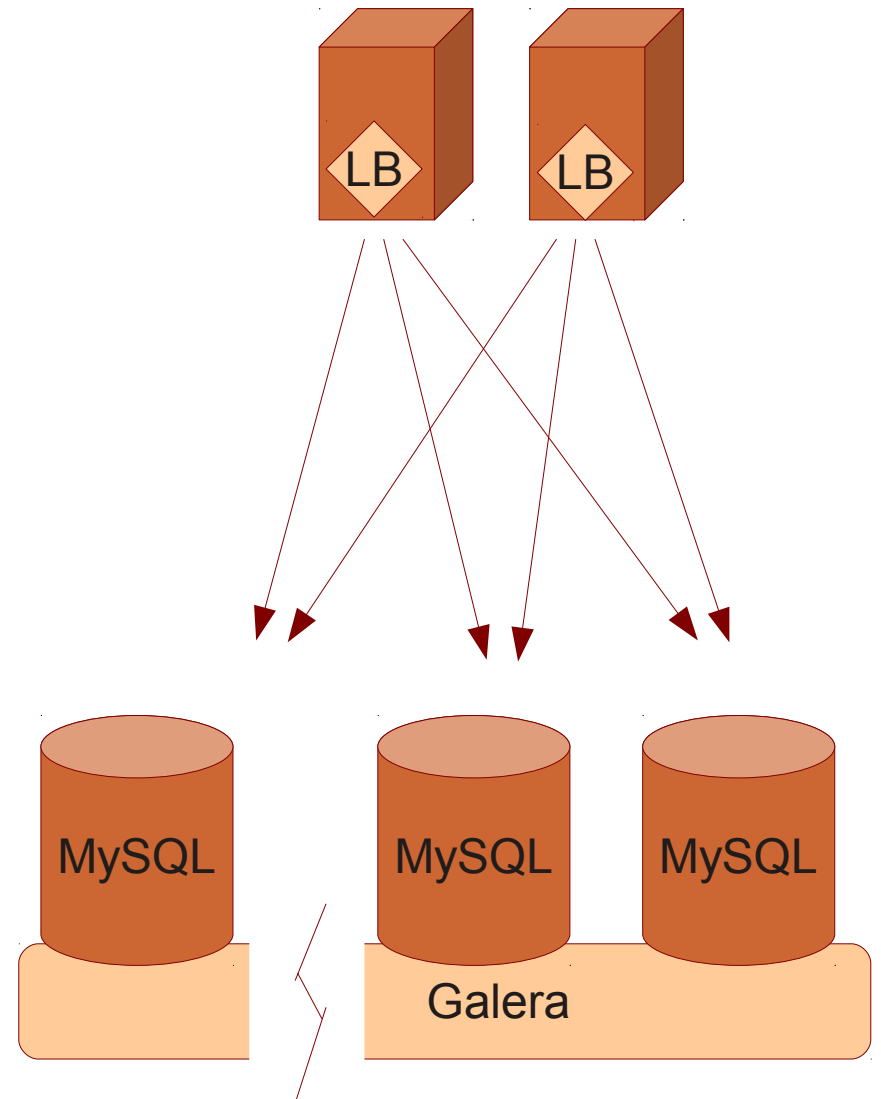


**How network partitioning is handled
aka
How split brain is prevented**

codership

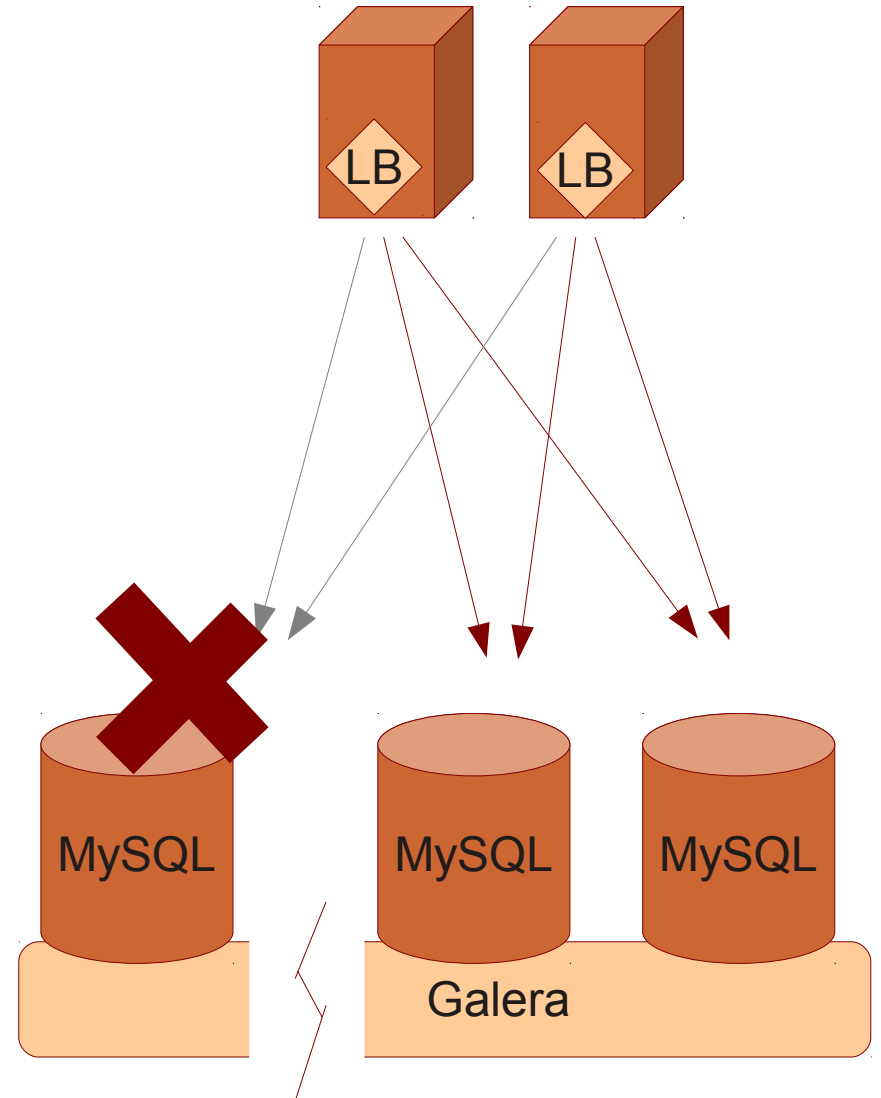
Preventing split brain

- If part of the cluster can't be reached, it means
 - The node(s) has crashed
 - Nodes are fine and it's a network connectivity issue
= **network partition**
 - A node cannot know which of the two has happened
 - Network partition may lead to **split brain** if both parts continue to commit transactions.
- Split brain leads to 2 diverging clusters, 2 diverged datasets
- Clustering SW must ensure there is only 1 cluster partition active at all times



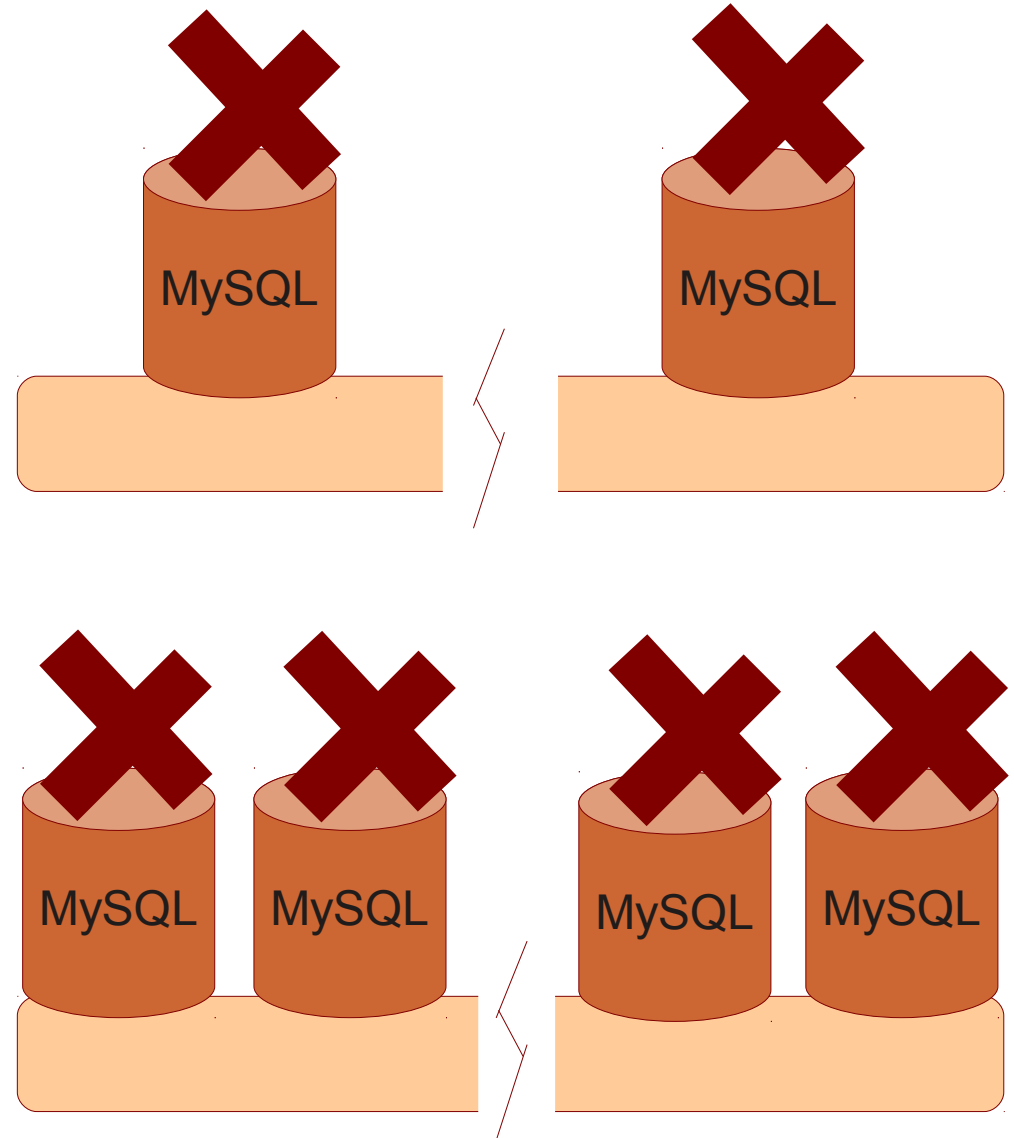
Quorum

- Galera uses quorum based failure handling:
 - When cluster partitioning is detected, the majority partition "has quorum" and can continue
 - A minority partition cannot commit transactions, but will attempt to re-connect to primary partition
- A load balancer will notice the errors and remove failed node from its pool



What is majority?

- 50% is not majority
- Any failure in 2 node cluster = both nodes must stop
- 4 node cluster split in half = both halves must stop
- `pc.ignore_sb` exists but don't use it
- You can manually/automatically enable one half by setting `wsrep_cluster_address`
- Use 3, 5, 7... nodes

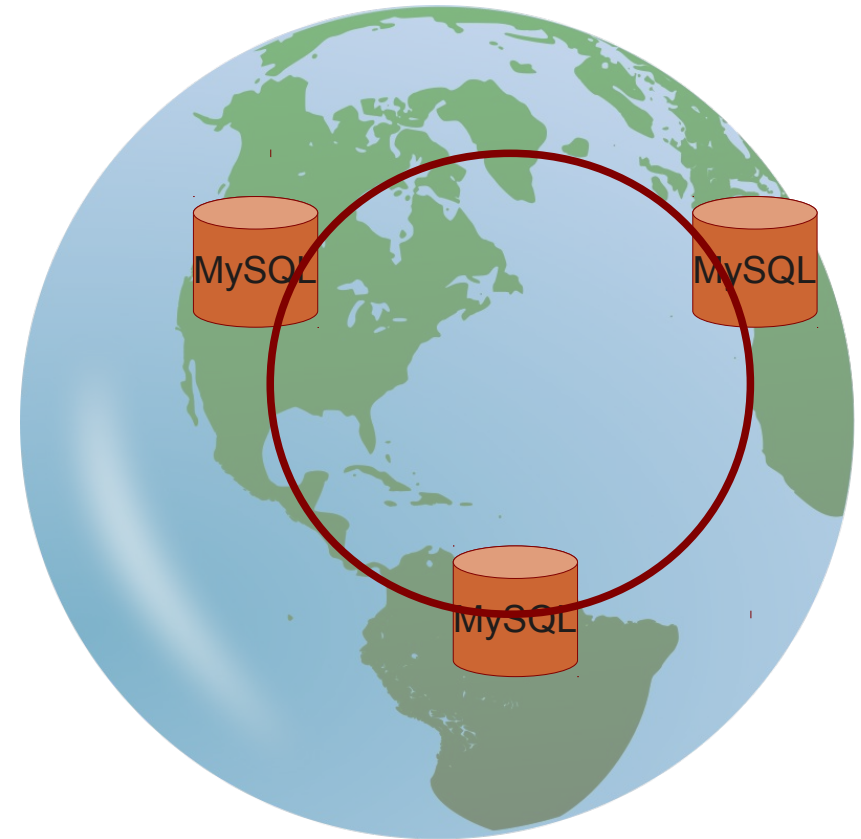


WAN replication

codership

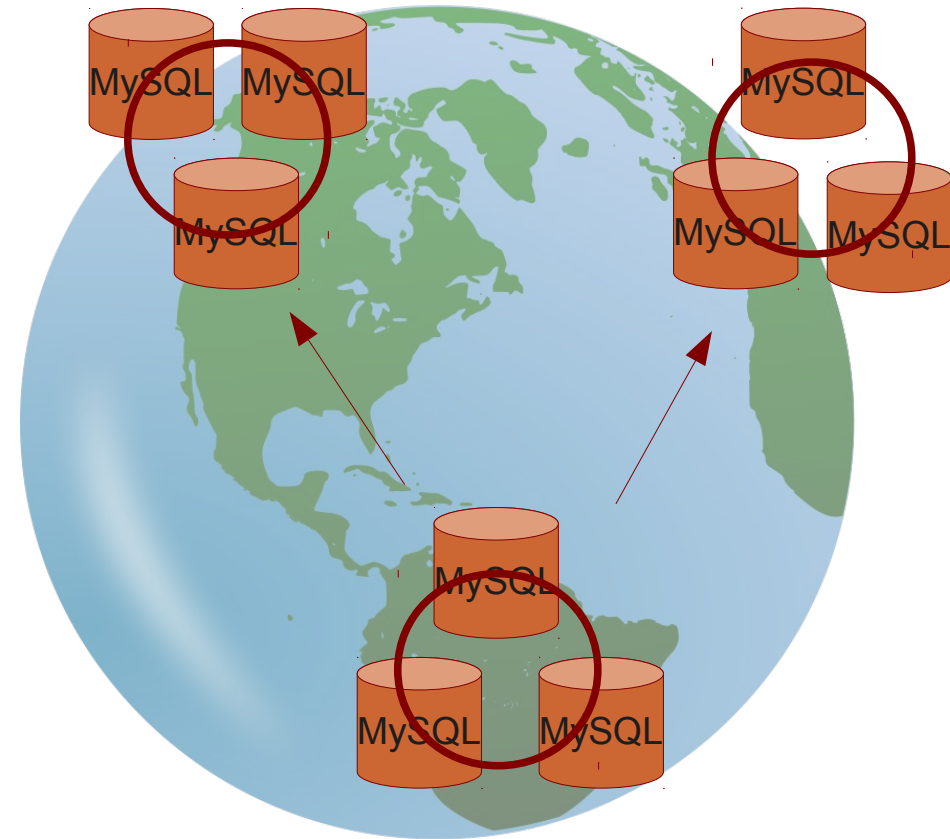
WAN replication

- Works fine
- Use higher timeouts and send windows
- No impact on reads
- No impact within a transaction
- adds 100-300 ms to commit latency



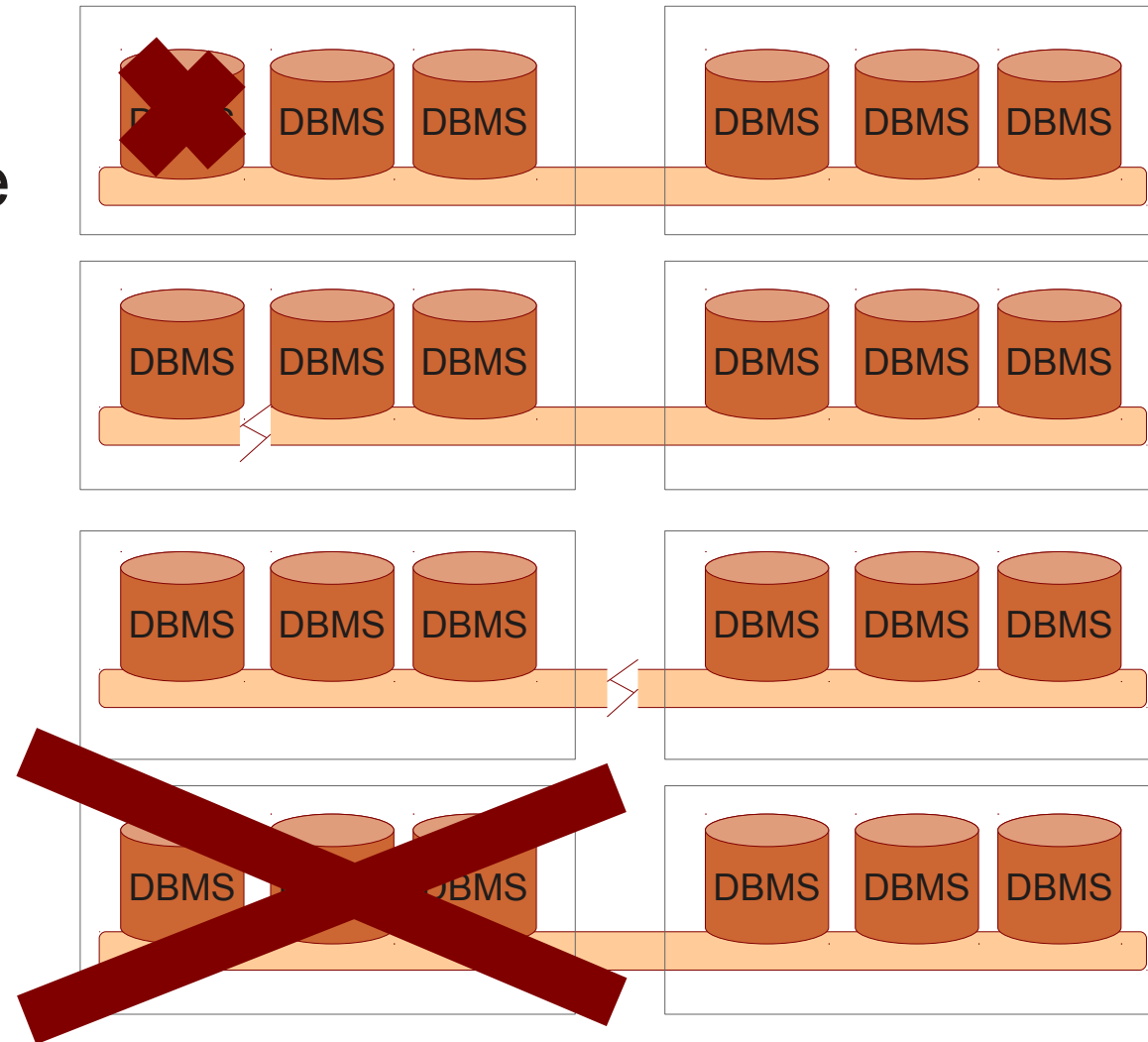
WAN with MySQL asynchronous replication

- You can mix Galera replication and MySQL replication
 - But it can give you a headache :-)
- Good option on slow WAN link (China, Australia)
- You'll possibly need more nodes than in pure Galera cluster
- Remember to watch out for slave lag, etc...
- If binlog position is lost (e.g. due to node crash) must reprovision whole cluster.
- Mixed replication also useful when you want an asynchronous slave (such as time-delayed, or filtered).



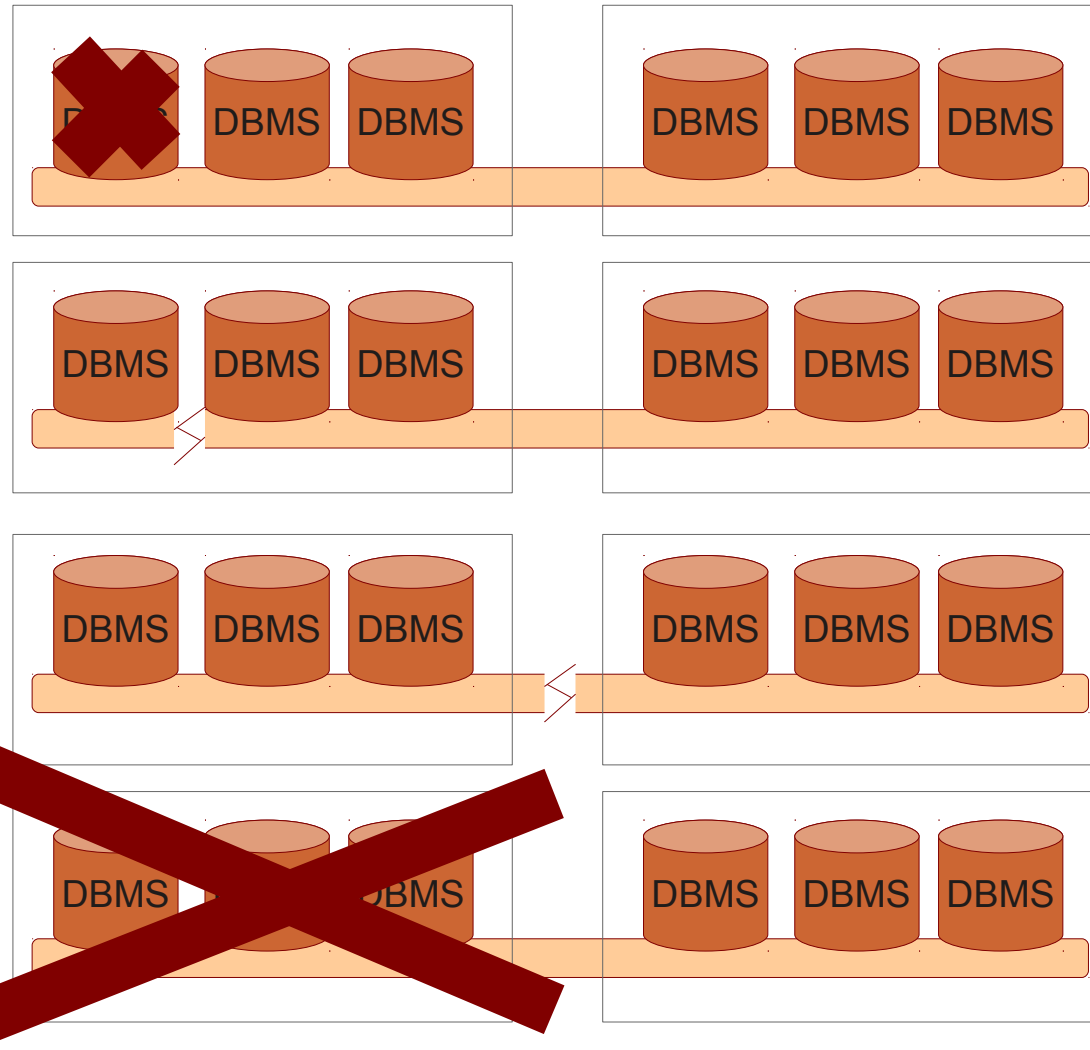
Failures & WAN replication

- Q: What does 50% rule mean when I have 2 datacenters?



Pop Quiz

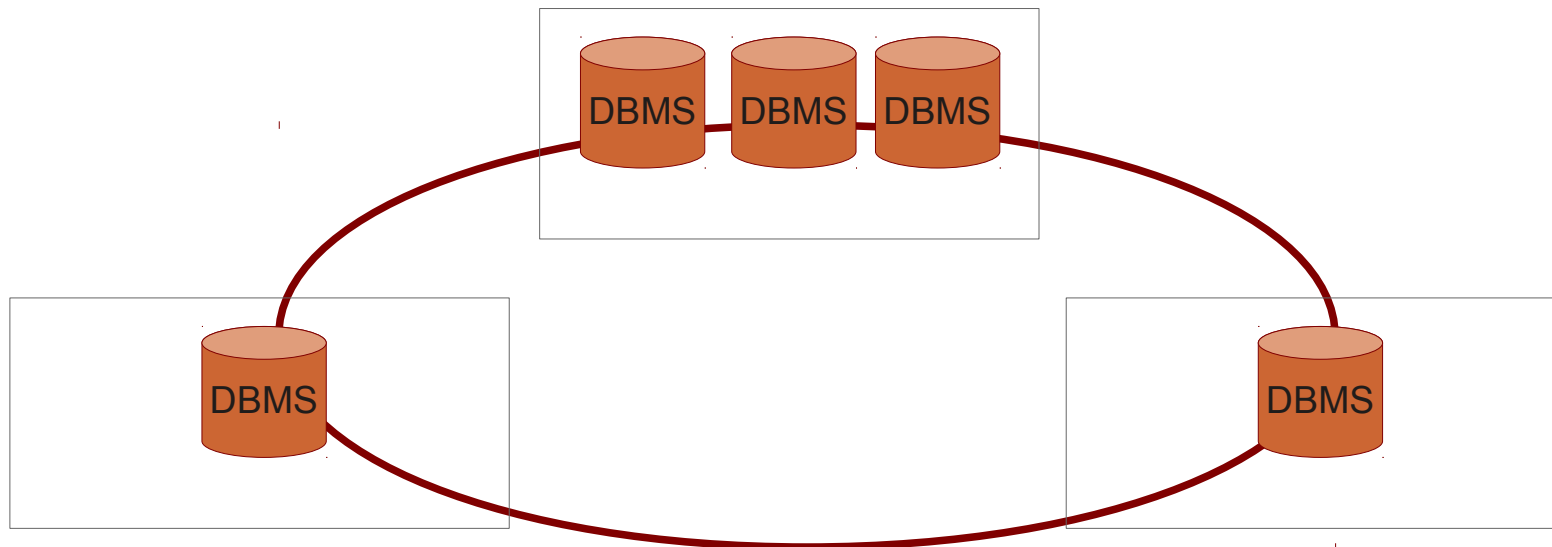
- Q: What does 50% rule mean when I have 2 datacenters?



- A: Better have 3 data centers too.

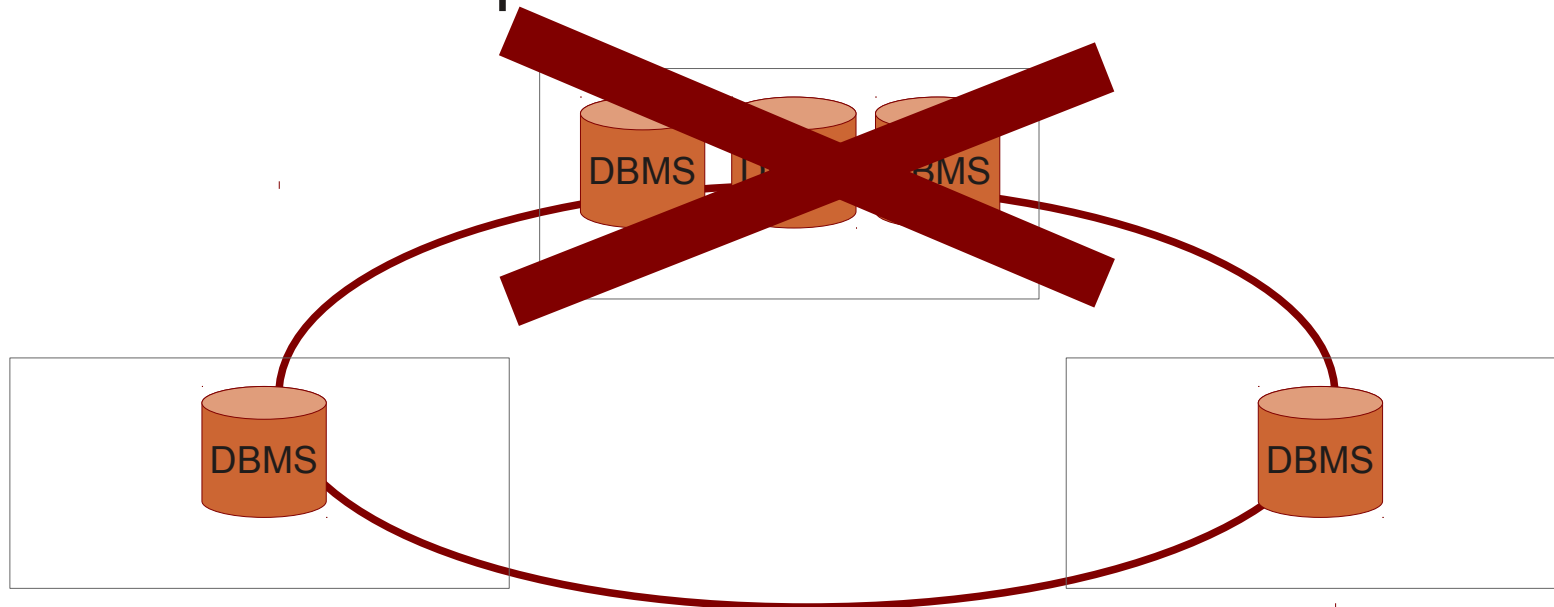
WAN replication with uneven node distribution

- Q: What does 50% rule mean when you have uneven amount of nodes per data center?



WAN replication with uneven node distribution

- Q: What does 50% rule mean when you have uneven amount of nodes per data center?

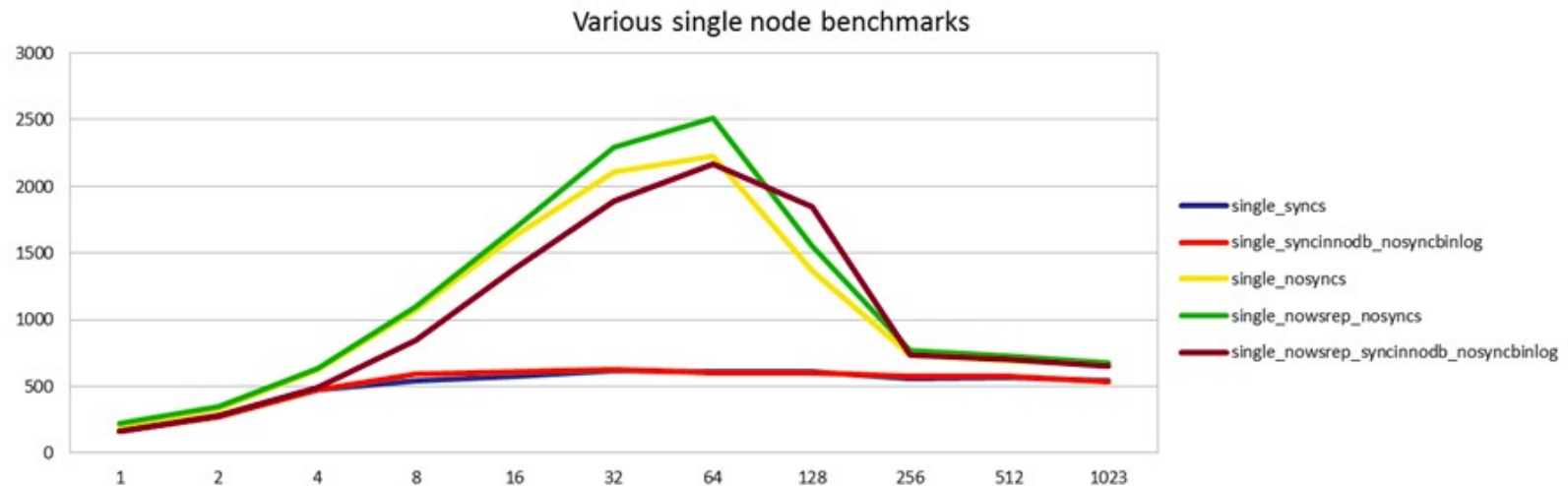


- A: Better distribute nodes evenly.
(We will address this in future release.)

Benchmarks!

codership

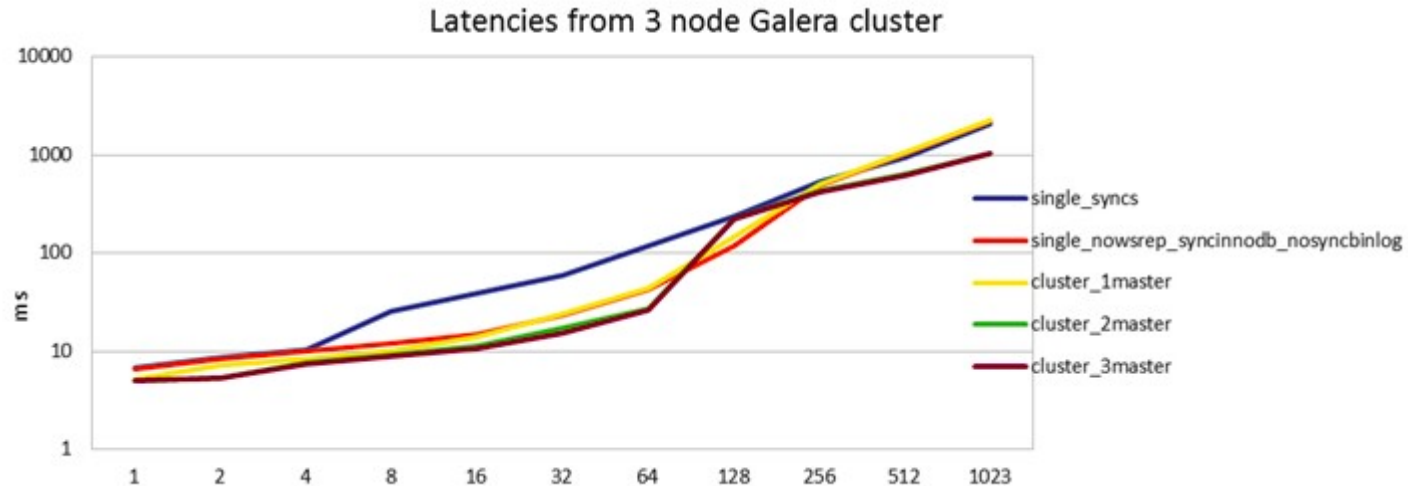
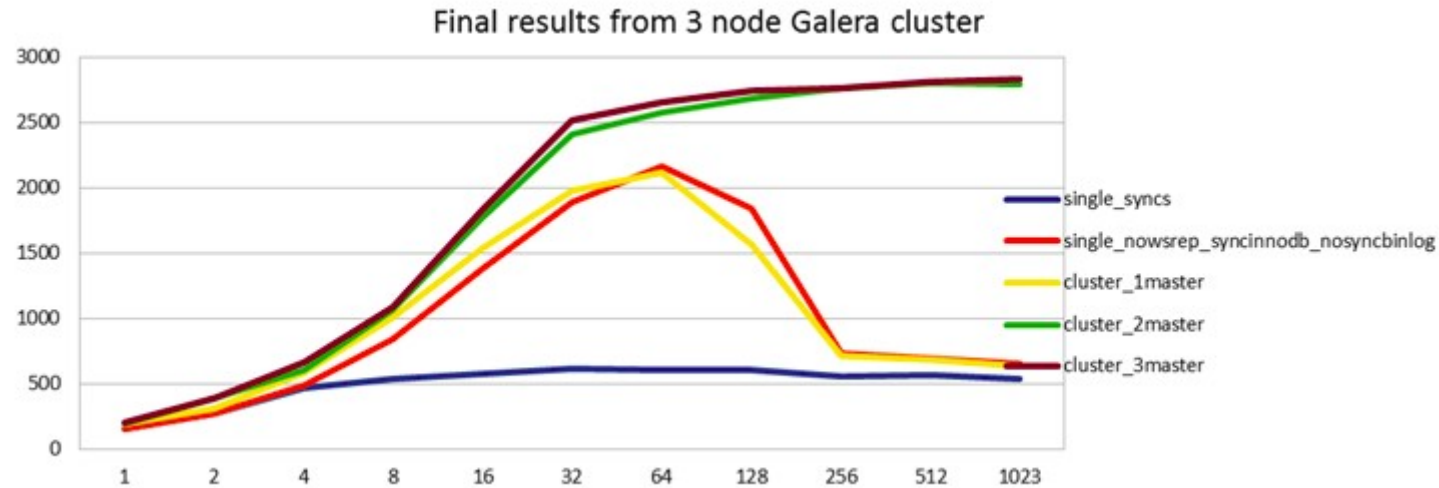
Baseline: Single node MySQL (sysbench oltp, in-memory)



- Red, Blue: Constrained by InnoDB group commit bug
 - Fixed in Percona Server 5.5, MariaDB 5.3 and MySQL 5.6
- Brown: InnoDB syncs, binlog doesn't
- Green: No InnoDB syncing either
- Yellow: No InnoDB syncs, Galera wsrep module enabled

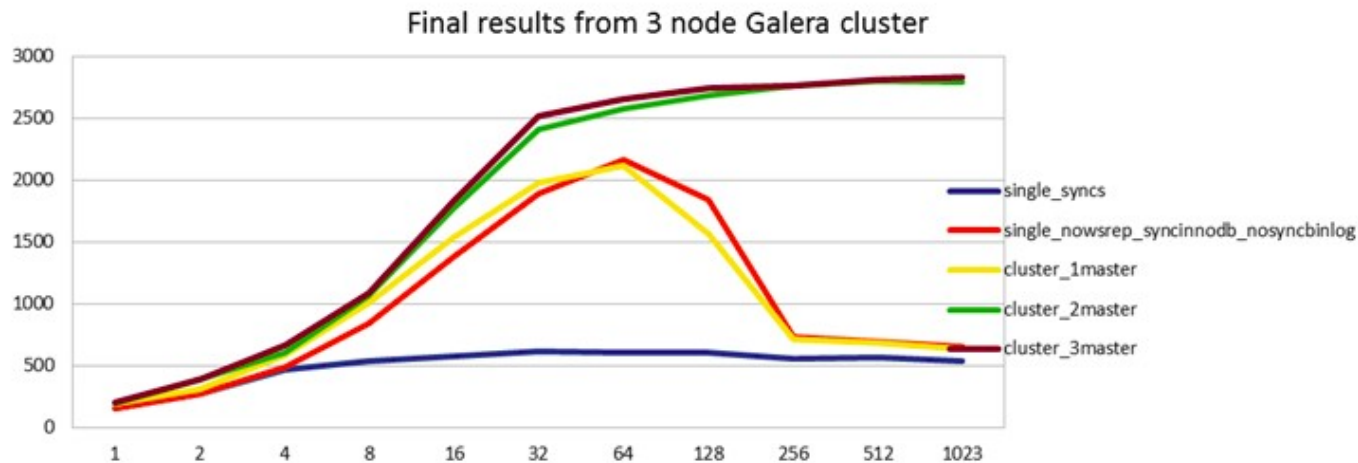
<http://openlife.cc/blogs/2011/august/running-sysbench-tests-against-galera-cluster>

3 node Galera cluster (sysbench oltp, in memory)



<http://openlife.cc/blogs/2011/august/running-sysbench-tests-against-galera-cluster>

Comments on 3 node cluster (sysbench oltp, in memory)

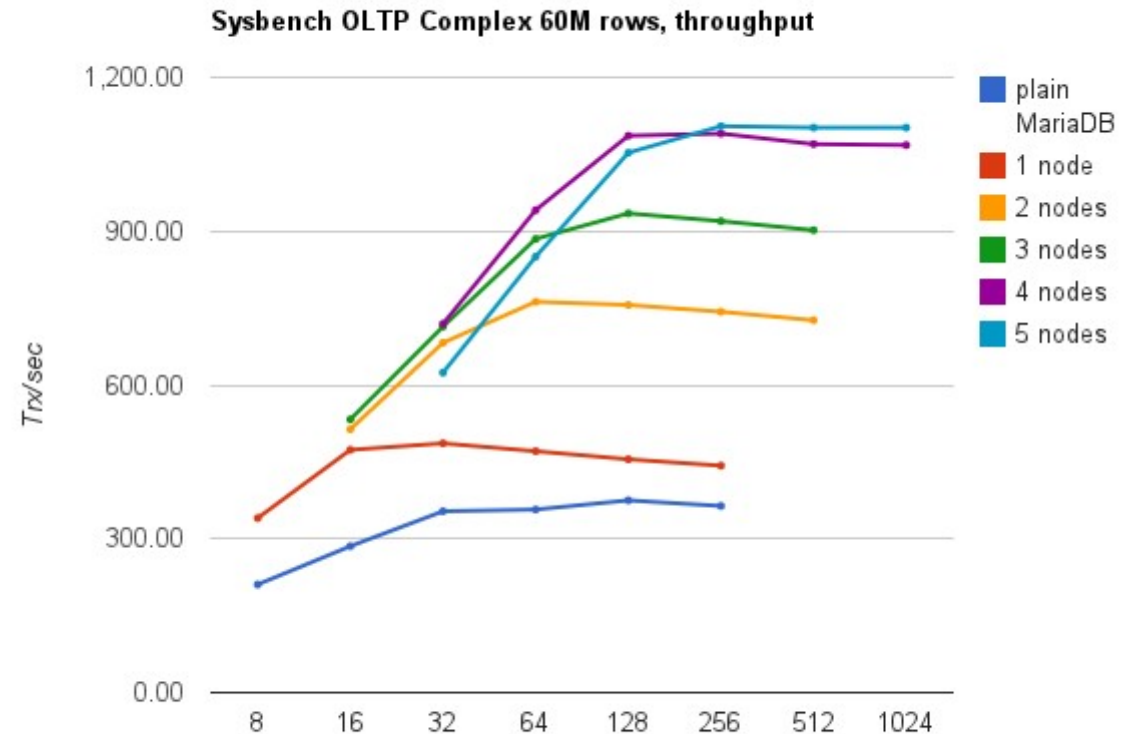


- Yellow, Red are equal
-> No overhead or bottleneck from Galera replication!
- Green, Brown = writing to 2 and 3 masters
-> scale-out for read-write workload!
 - top shows 700% CPU util (8 cores)

<http://openlife.cc/blogs/2011/august/running-sysbench-tests-against-galera-cluster>

Sysbench disk bound (20GB data / 6GB buffer), tps

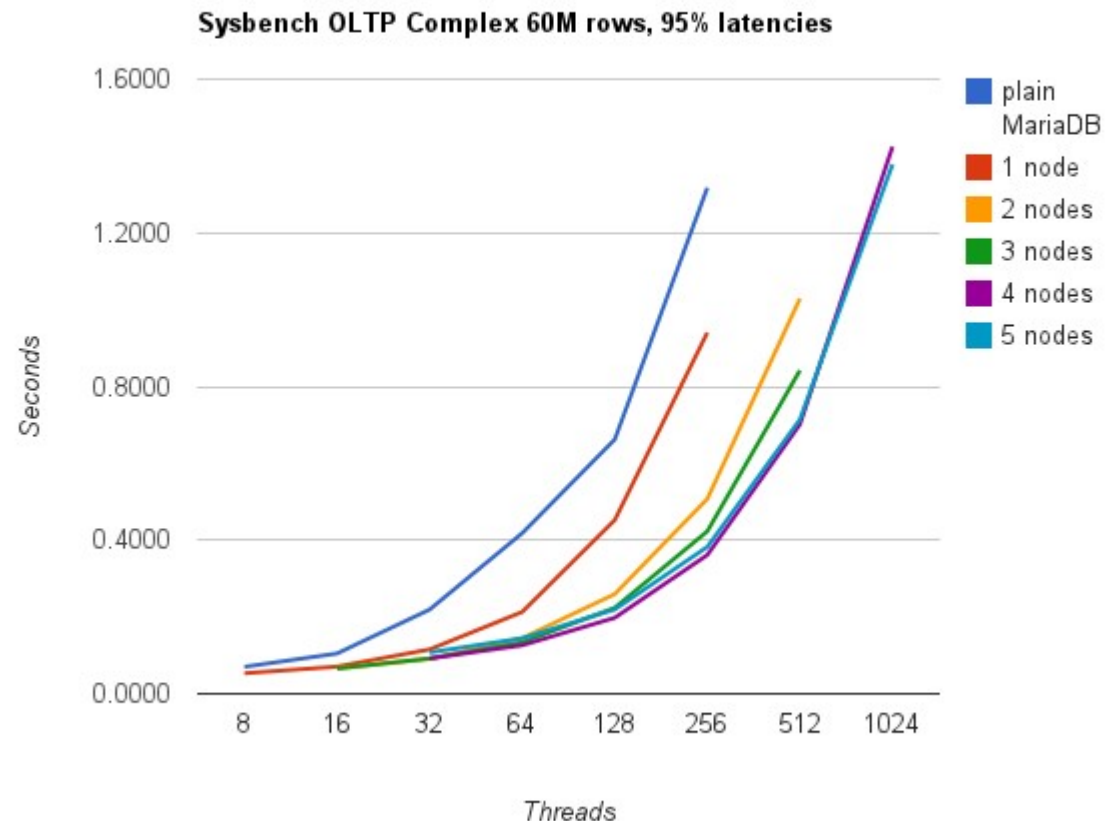
- EC2 w local disk
 - Note: pretty poor I/O here
- Blue vs red: turning off `innodb_flush_log_at_trx_commit` gives 66% improvement
- Scale-out factors:
 $2N = 0.5 \times 1N$
 $4N = 0.5 \times 2N$
- 5th node was EC2 weakness. Later test scaled a little more up to 8 nodes



<http://codership.com/content/scaling-out-oltp-load-amazon-ec2-revisited>

Sysbench disk bound (20GB data / 6GB buffer), latency

- As before
- Not syncing InnoDB decreases latency
- Scale-out decreases latency
- Galera does not add latency overhead



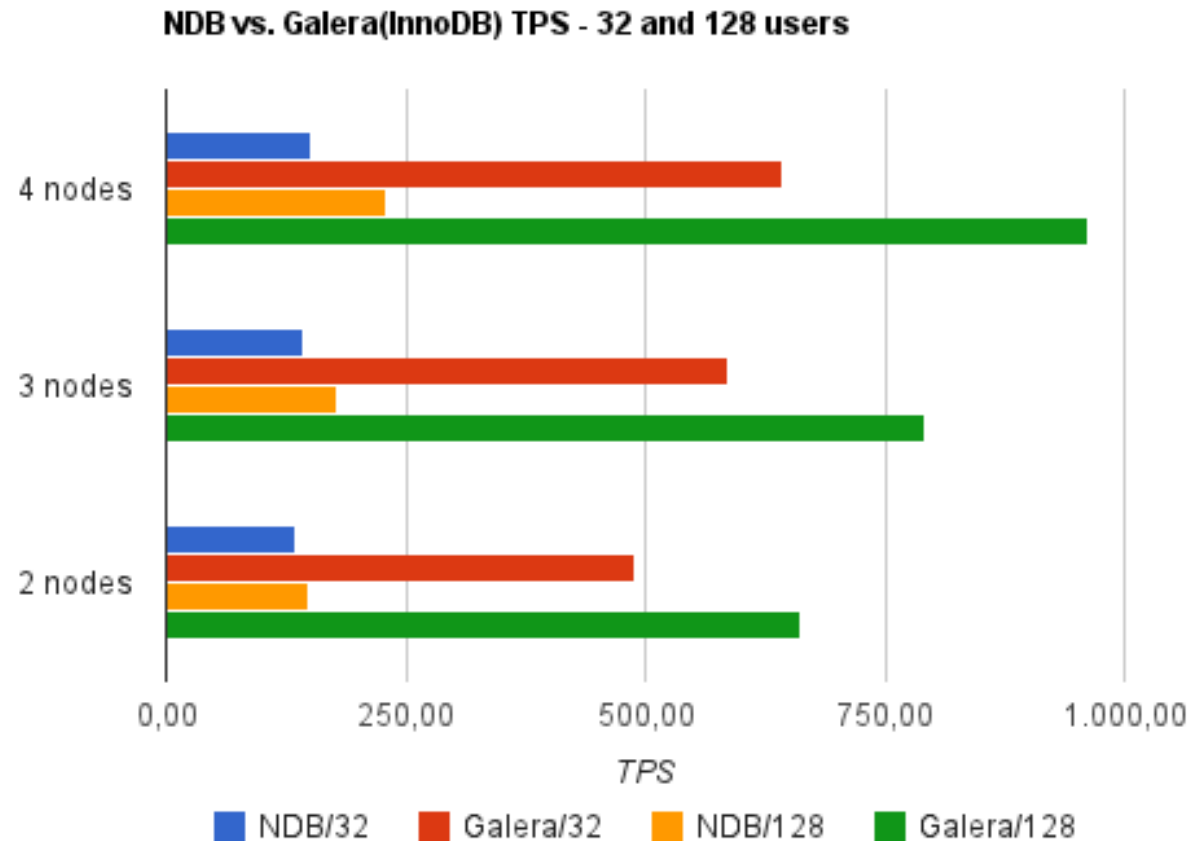
<http://codership.com/content/scaling-out-oltp-load-amazon-ec2-revisited>

Galera and NDB shootout: sysbench "out of the box"

- Galera is 4x better

Ok, so what does this really mean?

- That Galera is better...
 - For this workload
 - With default settings (Severalnines)
 - Pretty user friendly and general purpose
- NDB
 - Excels at key-value and heavy-write workloads (which sysbench is not)
 - Would benefit here from PARTITION BY RANGE



<http://codership.com/content/whats-difference-kenneth>

Summary

Many MySQL replication idioms go away: synchronous, multi-master, no slave lag, no binlog positions, automatic node provisioning.

Many LB and VIP architectures possible, JDBC/PHP load balancer is best.

Also for WAN replication. Adds 100-300 ms to commit.

Quorum based: Majority partition wins.

Minimum 3 nodes. Minimum 3 data centers.

Great benchmark results: more performance, not less, with replication!

Galera is good where InnoDB is good: general purpose, easy to use HA cluster

Compared to other alternatives:

Galera is better than MySQL replication:

- synchronous
- parallel slave
- multi-master
- easier

Galera is better than DRBD:

- no performance penalty
- active/active
- easier

Galera is better than NDB:

- based on good old InnoDB
- more general purpose
- easier

Questions?

Thank you for listening!
Happy Clustering :-)

codership