Using and Benchmarking Galera in Different Architectures

Henrik Ingo, Alex Yurchenko

Percona Live MySQL Conference and Expo
Santa Clara, 2012-04-12

codership

Agenda

MySQL Galera

* Synchronous multi-master clustering, what does it mean?

* Load balancing and other options

* WAN replication

* How network partitioning is handled

* How network partitioning is handled in WAN replication

How does it perform?

* In memory workload

* Scale-out for writes - how is it possible?

* Disk bound workload

* WAN replication

* Parallel slave threads

* Allowing slave to replicate (commit) out-of-order

* NDB shootout

.

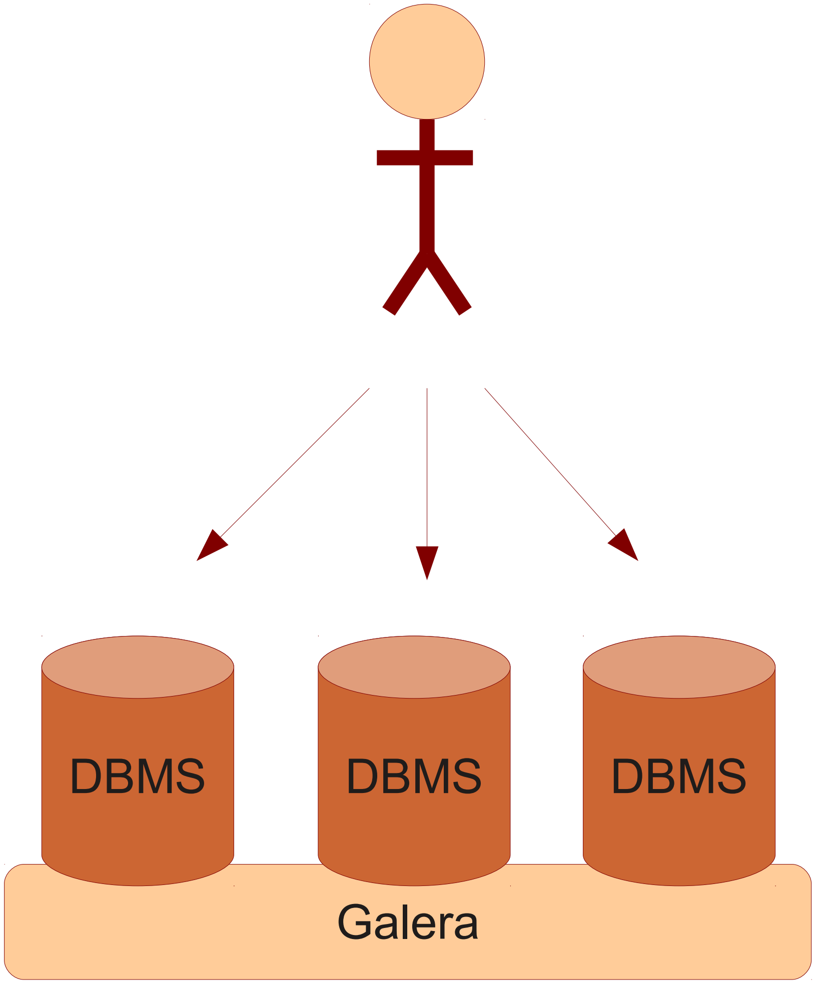
About Codership Oy

- Participated in 3 MySQL cluster developments earlier 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

Synchronous Multi-Master Clustering

Galera in a nutshell

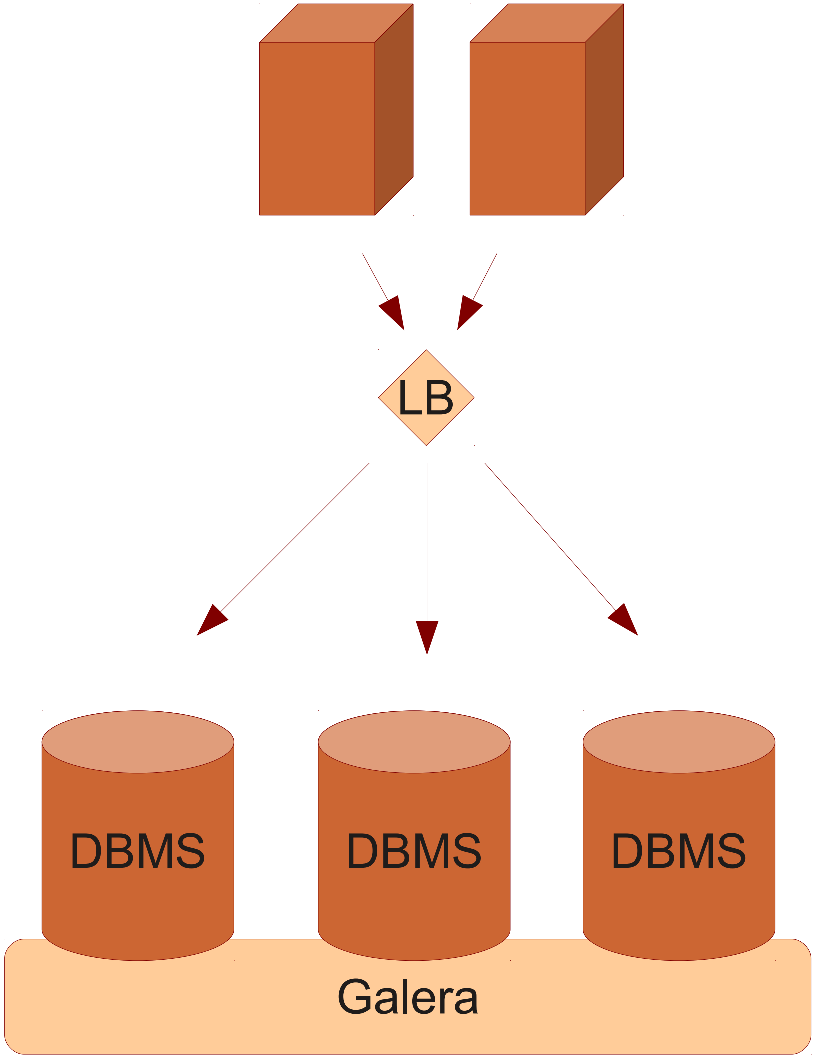
- True multi-master:
  Read & write to any node
- Synchronous replication
- No slave lag, integrity issues
- No master-slave failovers, no VIP needed
- Multi-threaded slave
- Automatic node provisioning



Load balancing

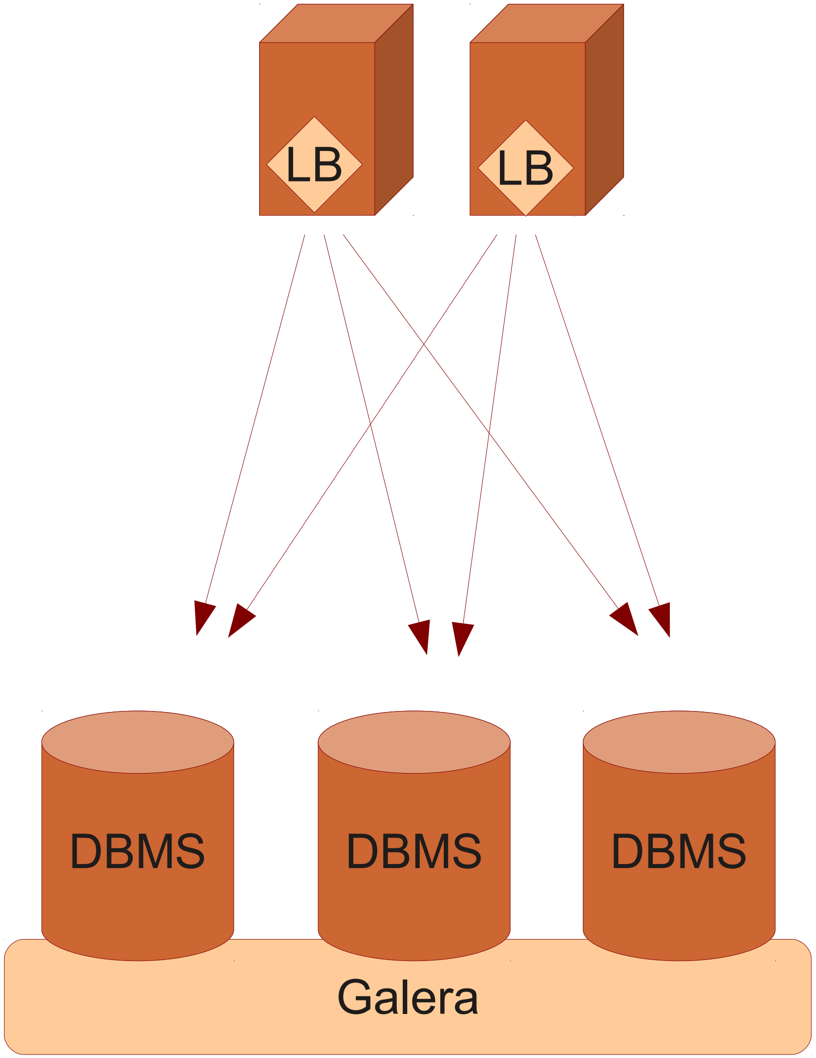
What do you mean no failover???

- 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



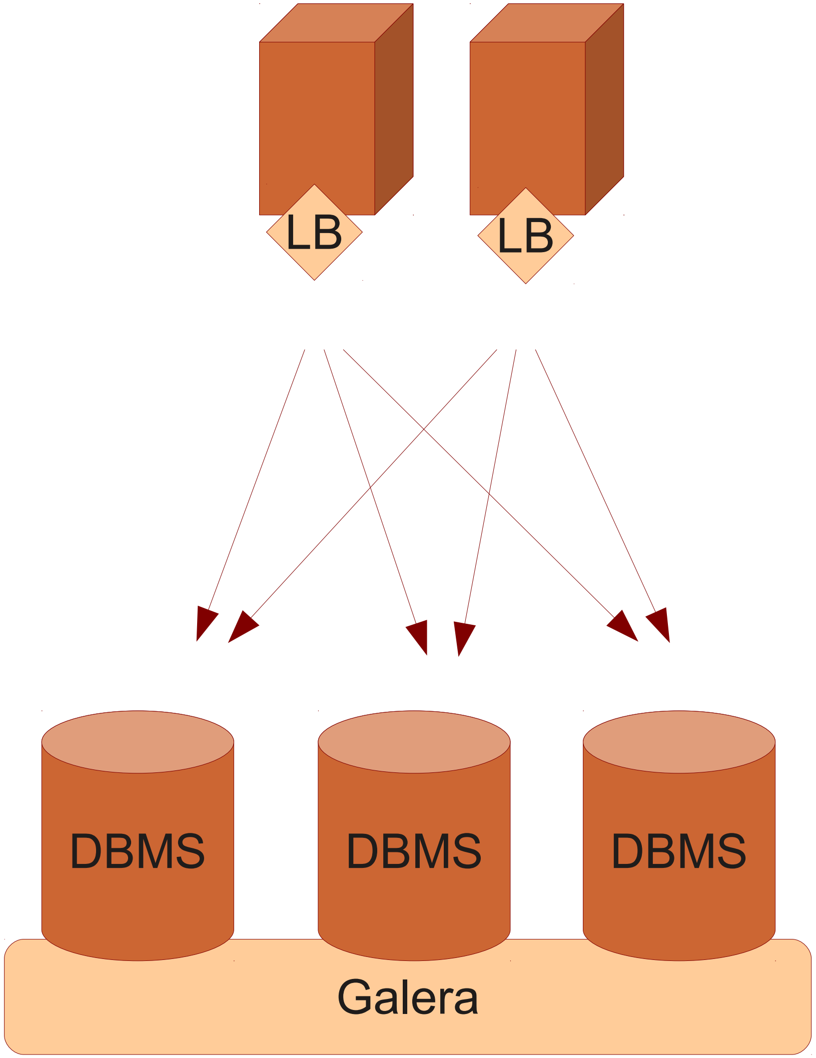
Protip: JDBC, PHP come with built-in load balancing!

- No Single Point of Failure
- One less layer of network components
- Is aware of MySQL transaction states and errors
- Sysbench does this internally too (except it doesn't really failover)



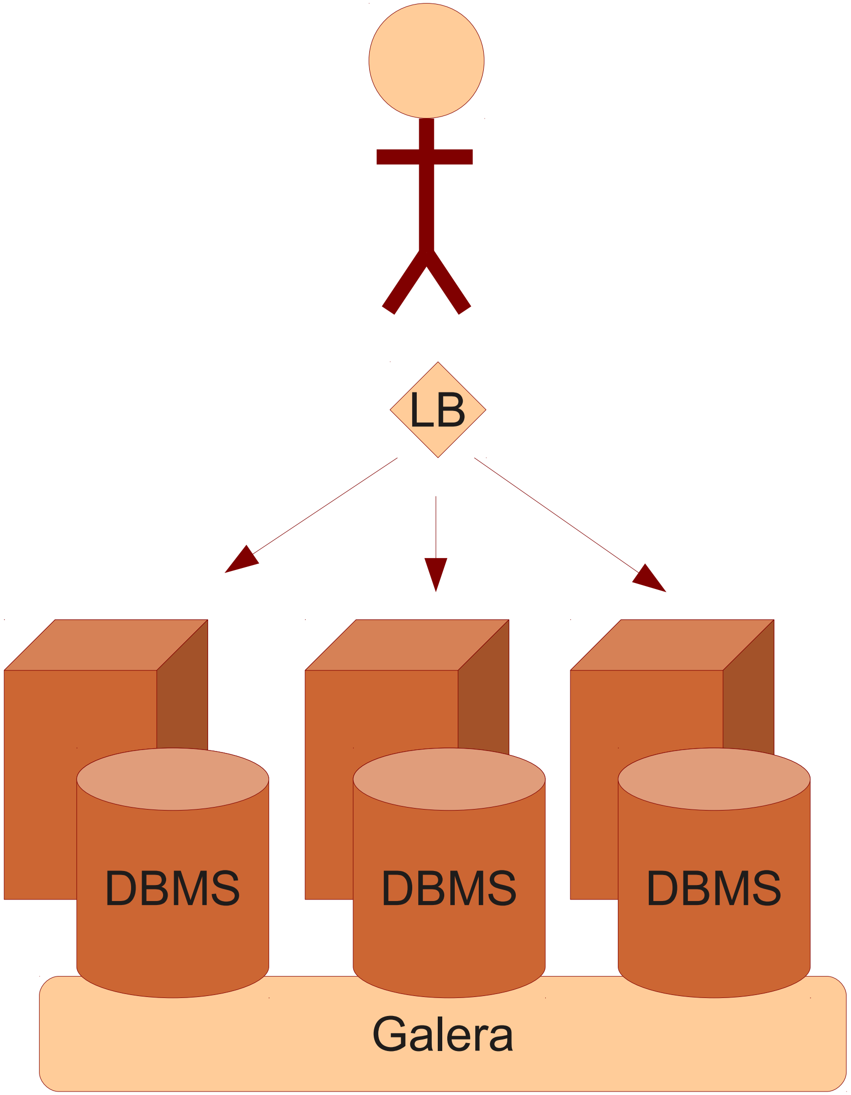
Load balancer per each app node

- 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

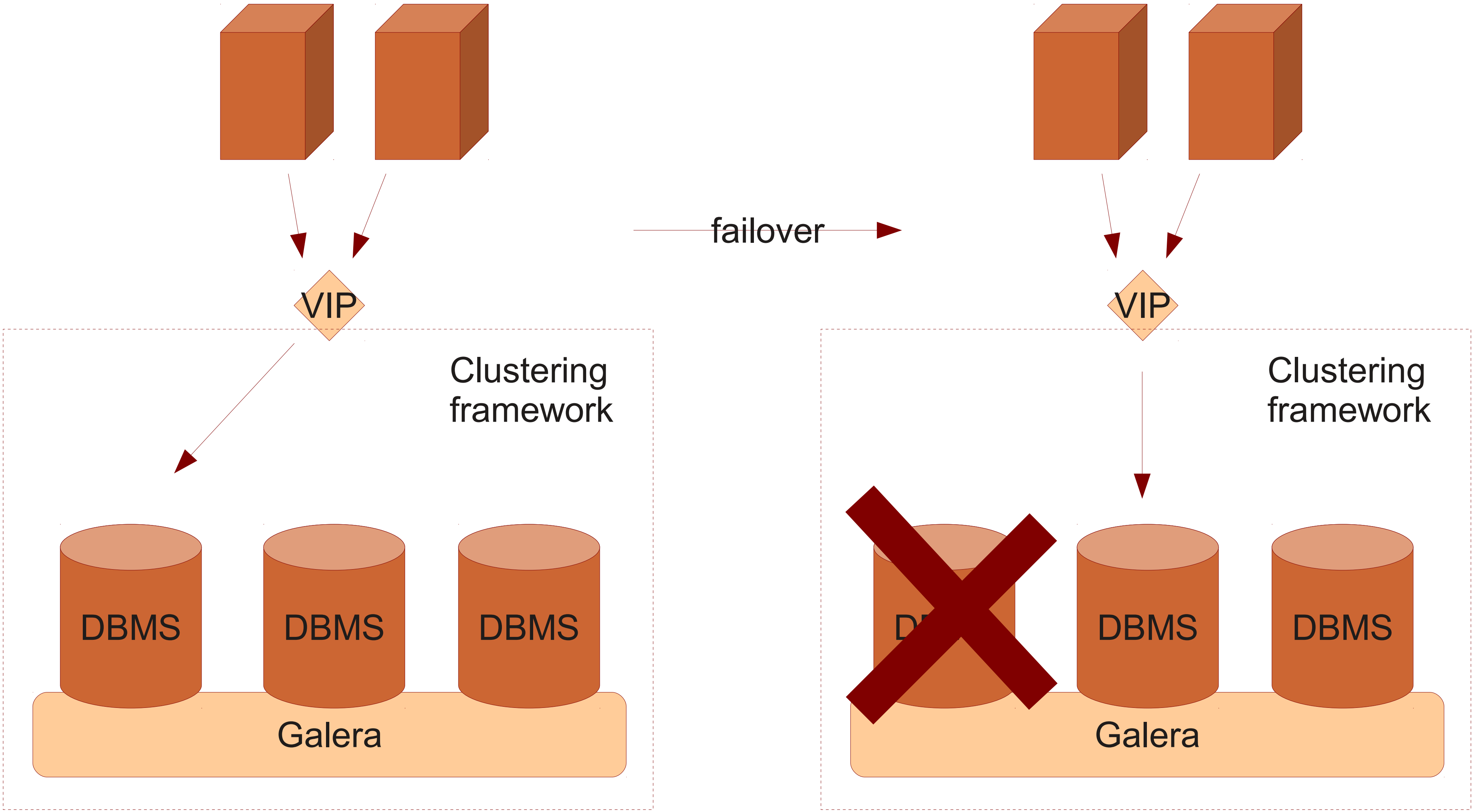


Whole stack cluster (no load balancing)

- One DB node per app server, usually same host
- LB on HTTP or DNS level
- Each app server connects to localhost
- Simple
- Usually app server cpu is bottleneck
  - Bad: This is a bit wasteful architecture, especially if DB is large
  - Good: Replication overhead w Galera is fairly small



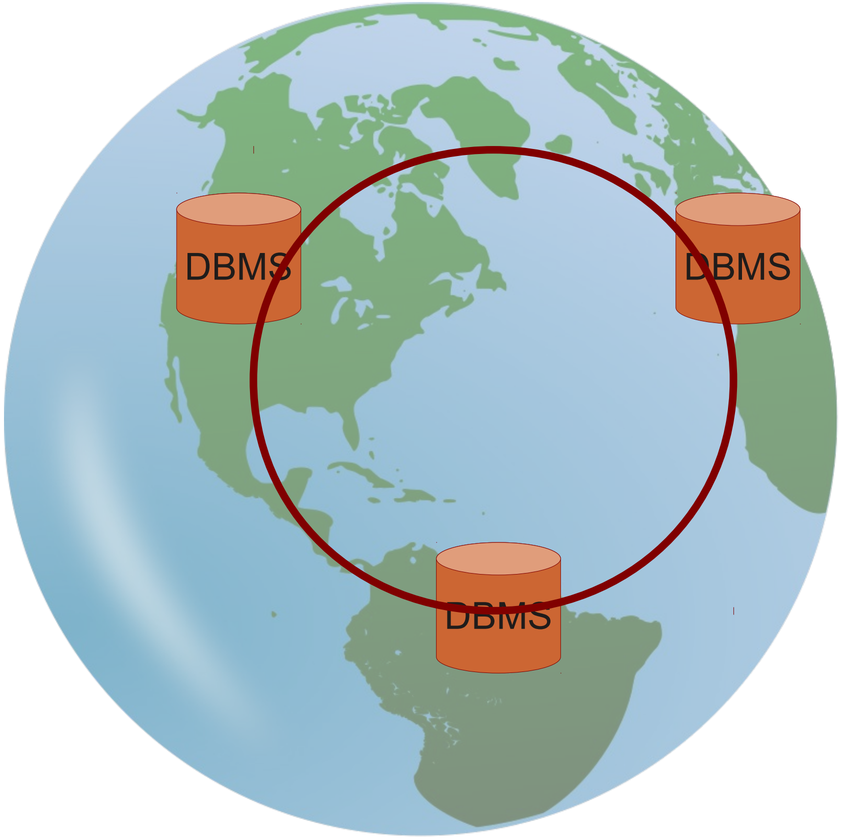
You can still do VIP based failovers. But why?



WAN replication

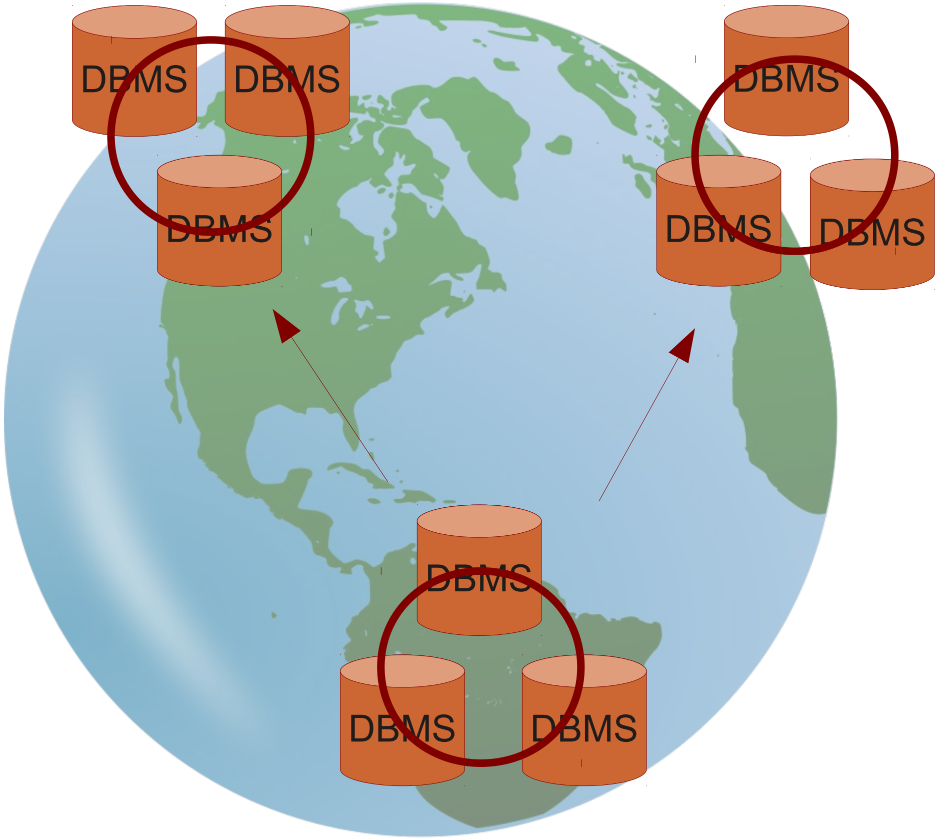
WAN replication

- Works fine
- Use higher timeouts
- No impact on reads
- No impact within a transaction
- adds 100-300 ms to commit latency (see benchmarks)



WAN with MySQL asynchronous replication

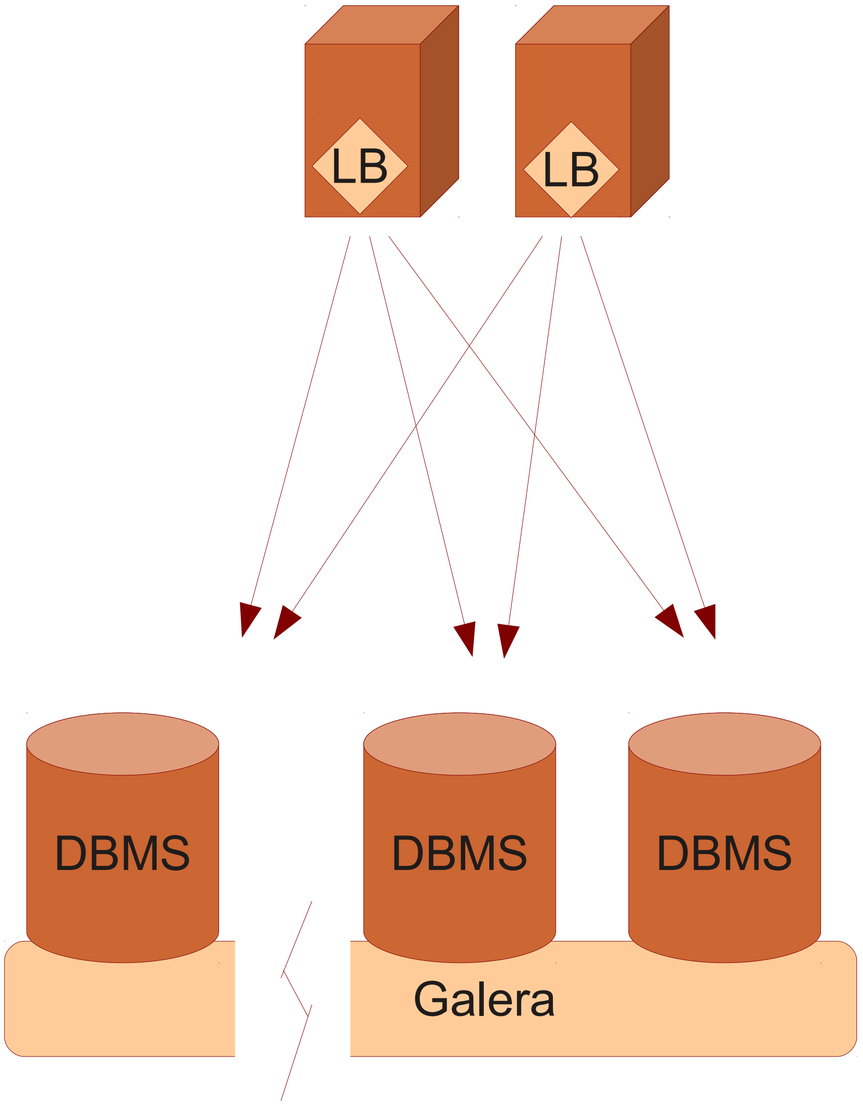
- You can mix Galera replication and MySQL replication
  - But it can give you a headache :-)
- Good option on slow WAN link (China, Australia)
- You'll possibly need more nodes than in pure Galera cluster
- Remember to watch out for slave lag, etc...
- If binlog position is lost (e.g. due to node crash)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



How network partitioning is handled
aka
How split brain is prevented

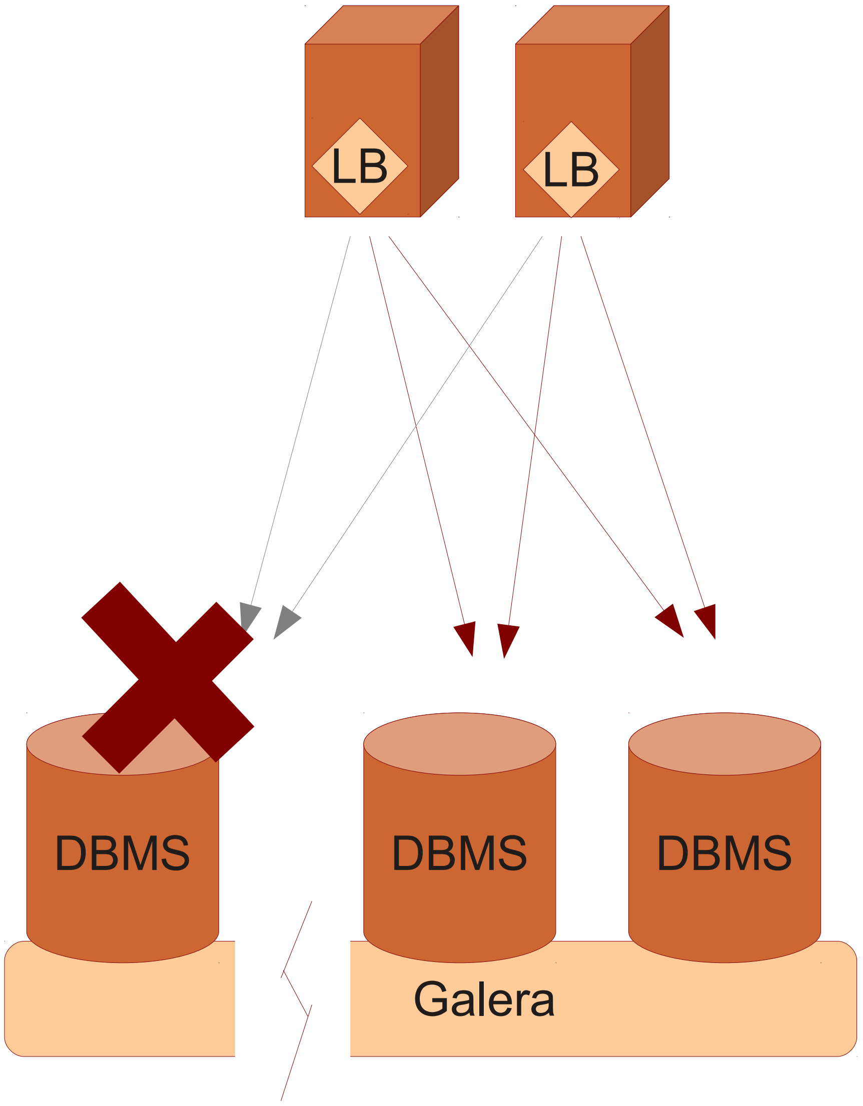
Preventing split brain

- 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**
  - Network partition may lead to **split brain** if both parts continue to commit transactions.
  - A node cannot know which of the two has happened
- Split brain leads to 2 diverging clusters, 2 diverged datasets
- Clustering SW must ensure there is only 1 cluster partition active at all times



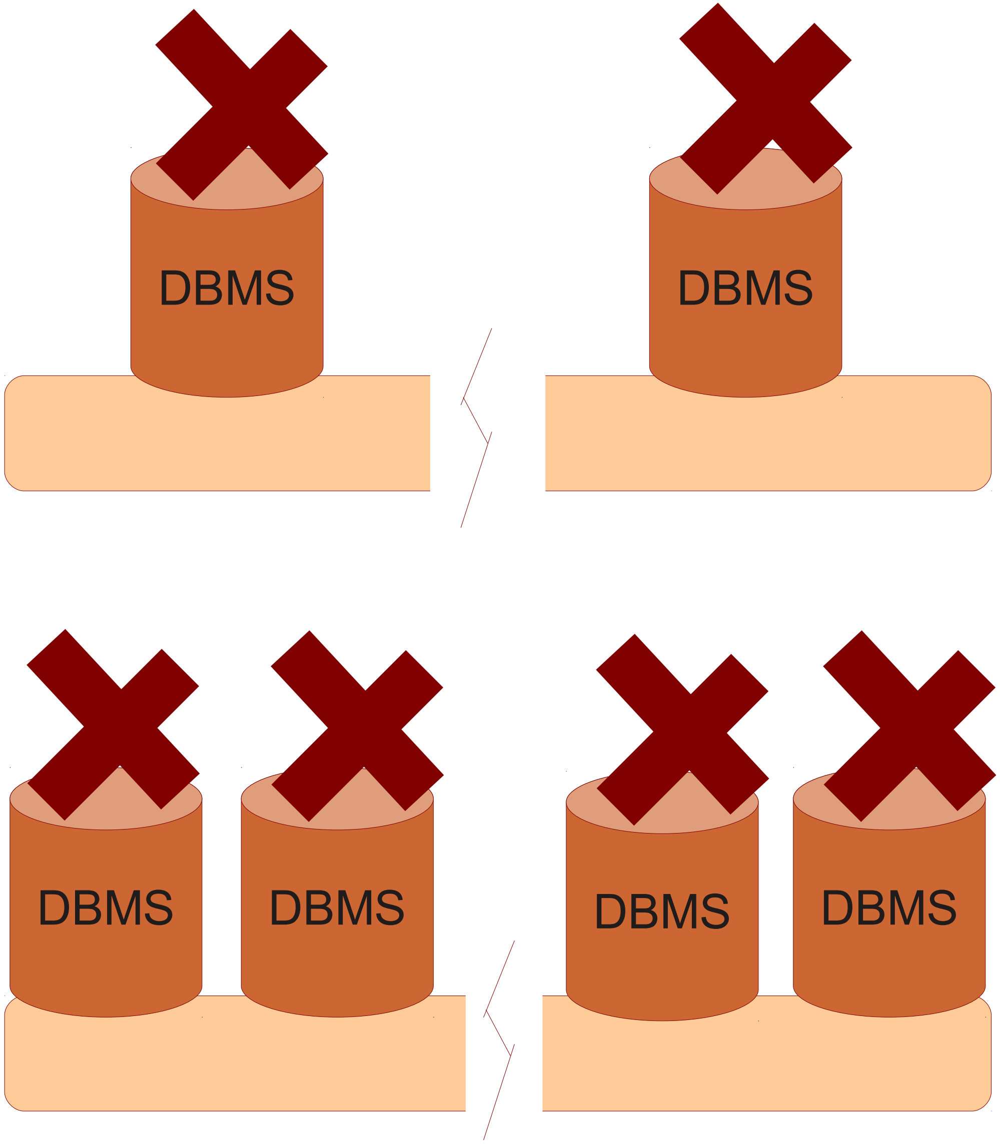
Quorum

- 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?

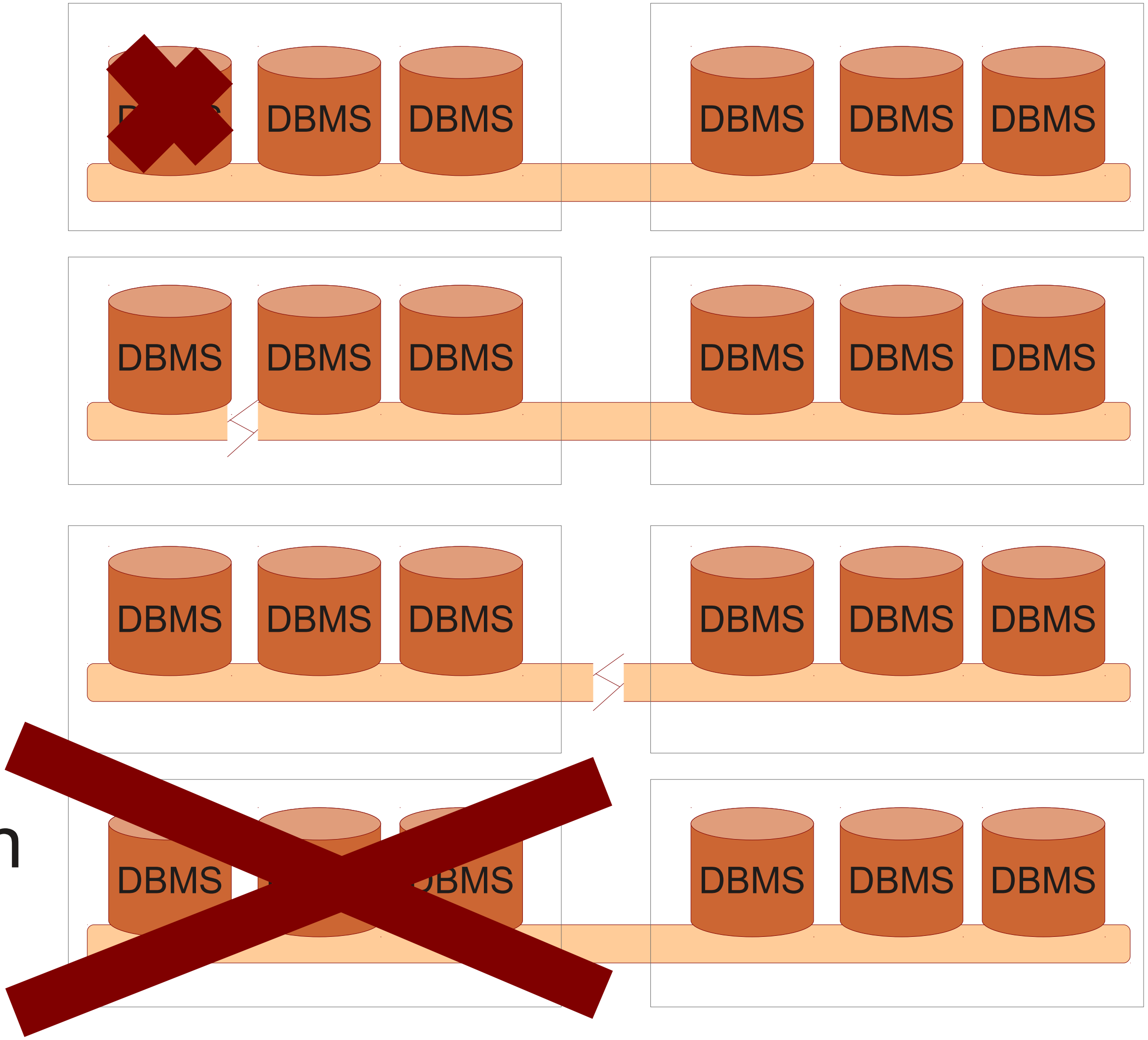
- 50% is not majority
- Any failure in 2 node cluster = both nodes must stop
- 4 node cluster split in half = both halves must stop
- pc.ignore_sb exists but don't use it
- You can manually/automatically enable one half by setting wsrep_cluster_address
- Use 3, 5, 7... nodes



Failures in WAN replication

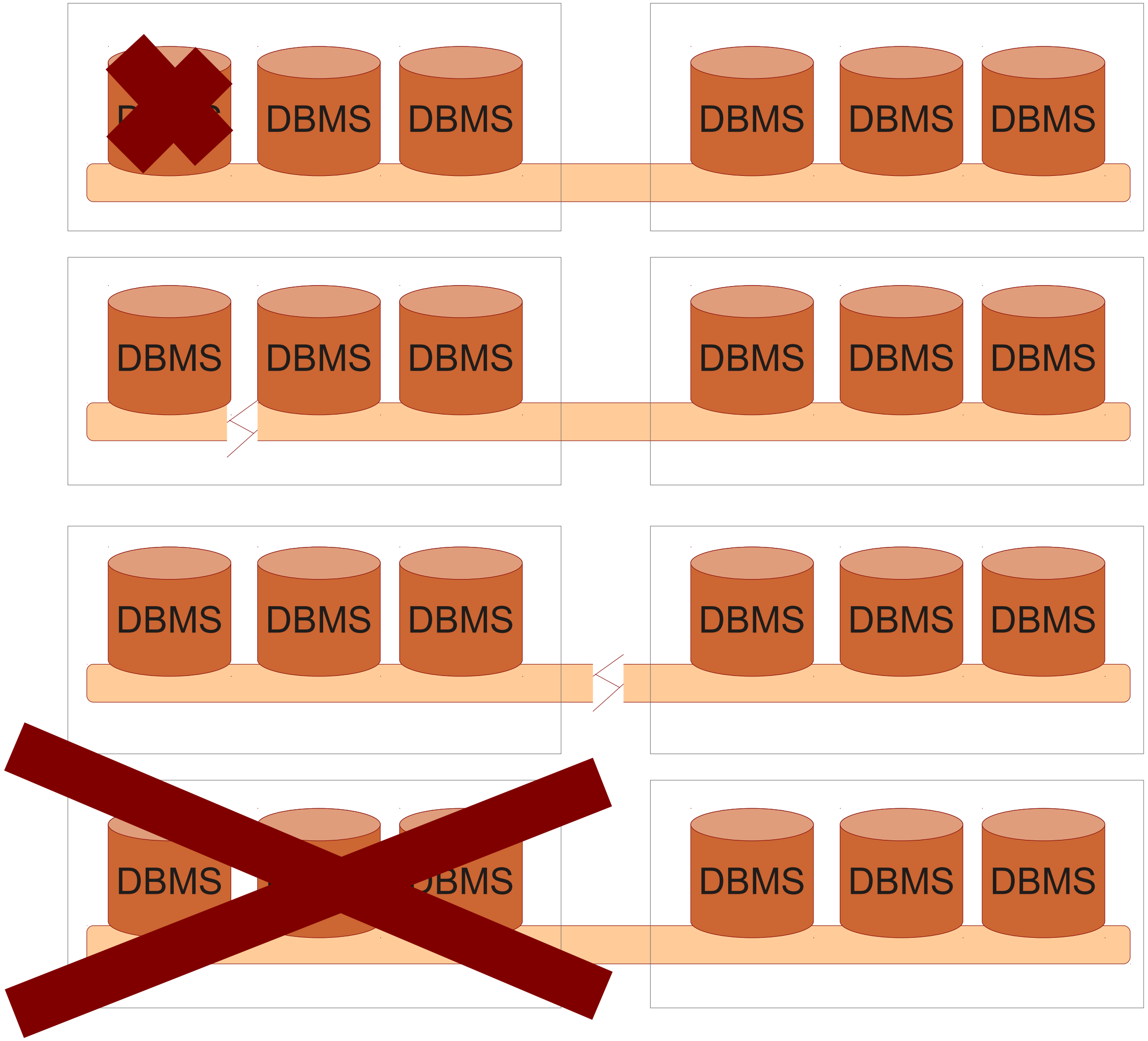
Multiple Data Centers

- A single node can fail
- A single node can have network connectivity issue
- The whole data center can have connectivity issue
- A whole data center can be destroyed



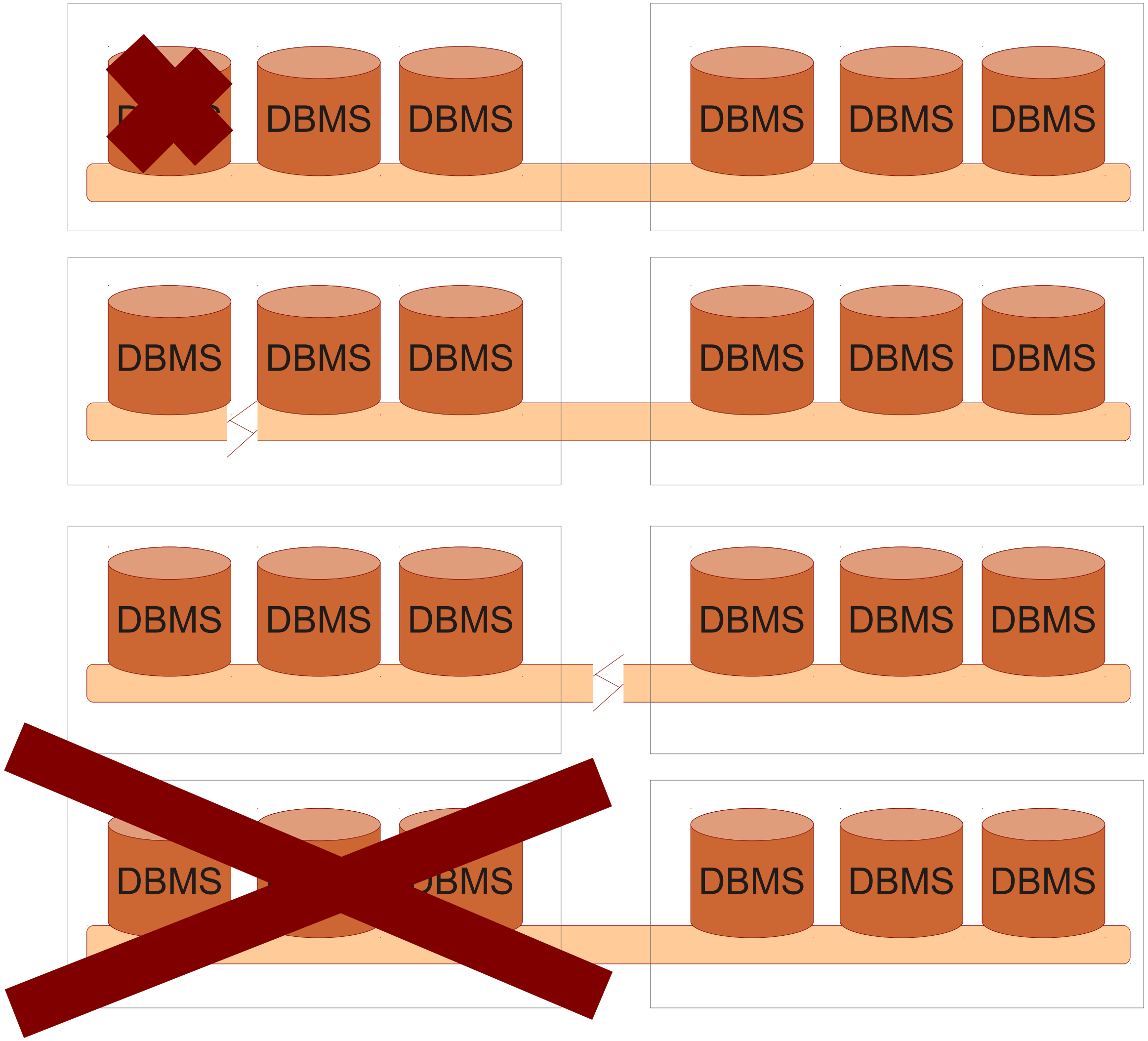
Pop Quiz

- Q: What does 50% rule mean in each of these cases?



Pop Quiz

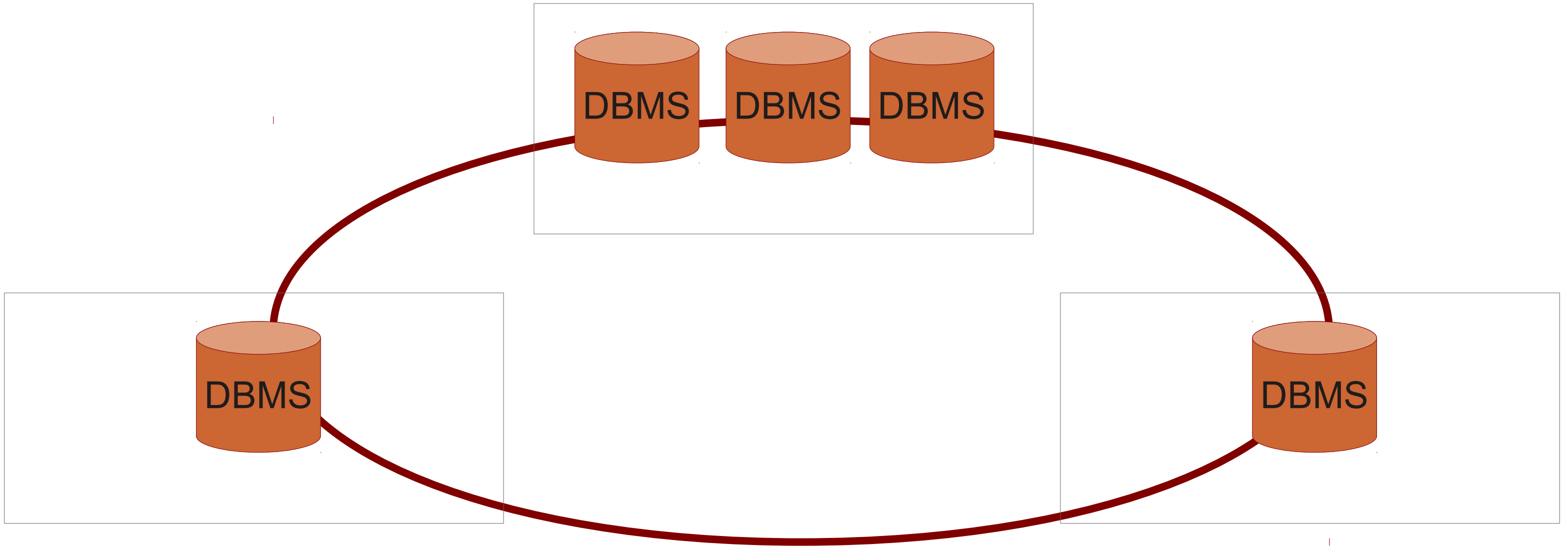
- Q: What does 50% rule mean in each of these cases?



- A: Better have 3 data centers too.

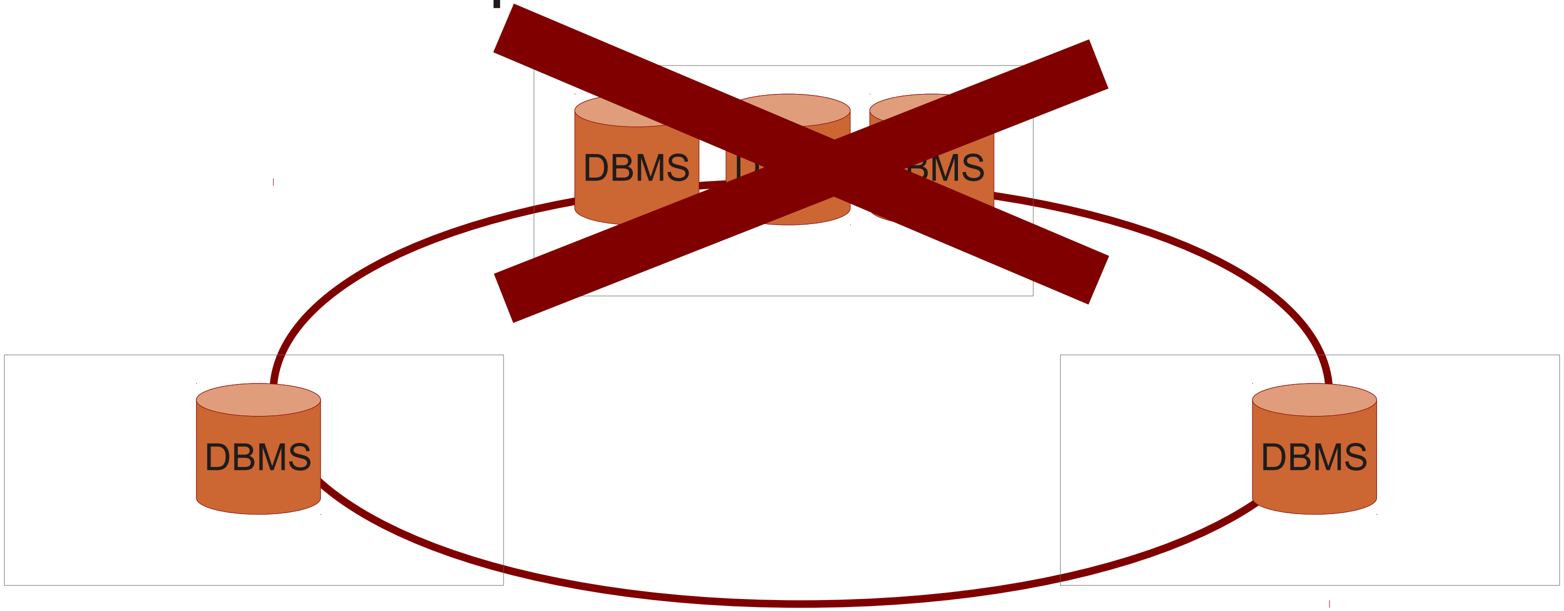
WAN replication with uneven node distribution

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



WAN replication with uneven node distribution

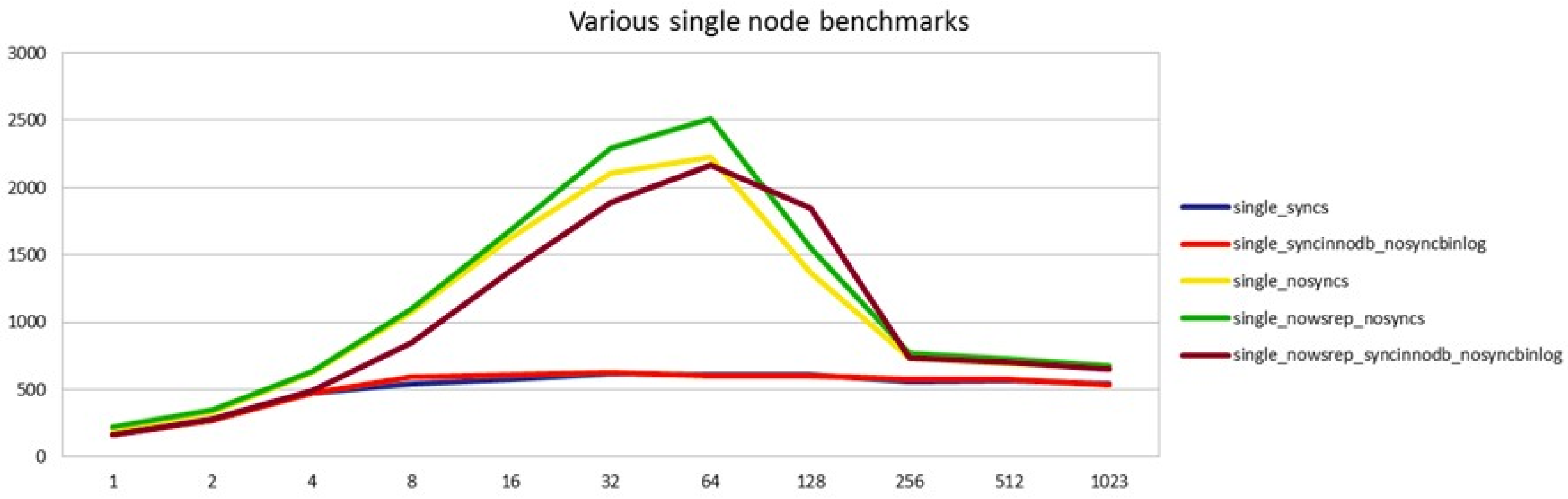
- Q: What does 50% rule mean when you have uneven amount of nodes per data center?



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

Benchmarks!

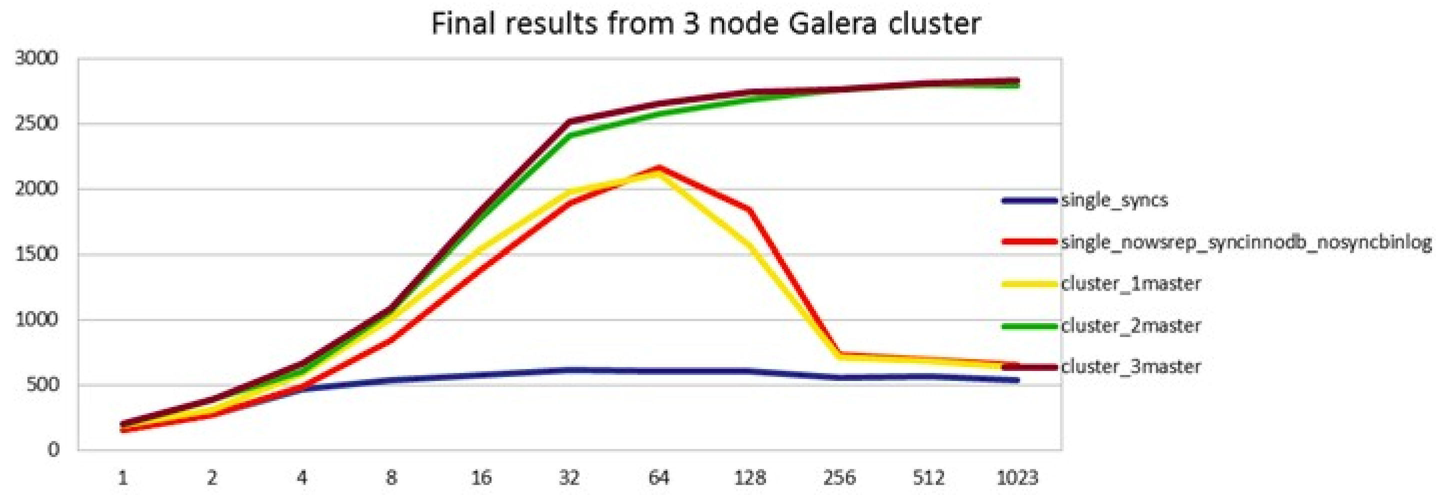
Baseline: Single node MySQL (sysbench oltp, in-memory)

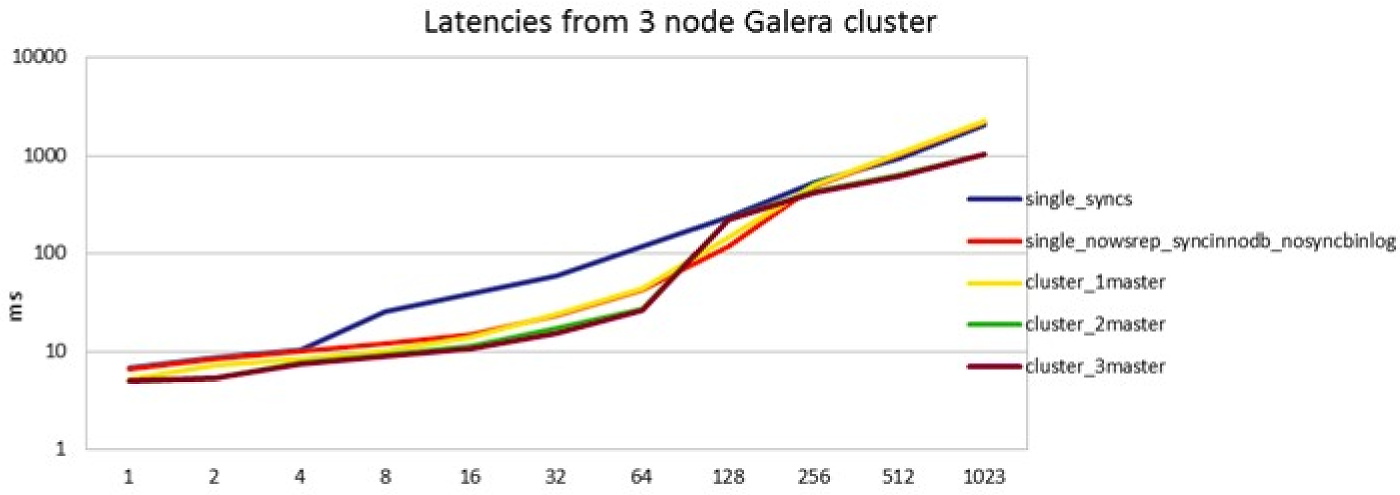


- Red, Blue: Constrained by InnoDB group commit bug
  - Fixed in Percona Server 5.5, MariaDB 5.3 and MySQL 5.6
- Brown: InnoDB syncs, binlog doesn't
- Green: No InnoDB syncing either
- Yellow: No InnoDB syncs, Galera wsrep module enabled

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

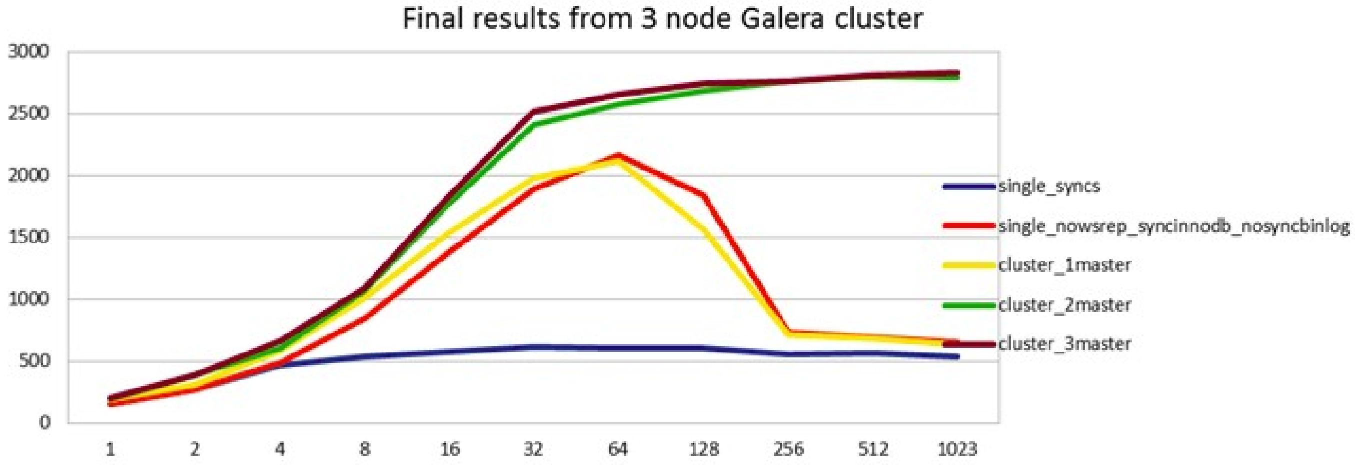
3 node Galera cluster (sysbench oltp, in memory)





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

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

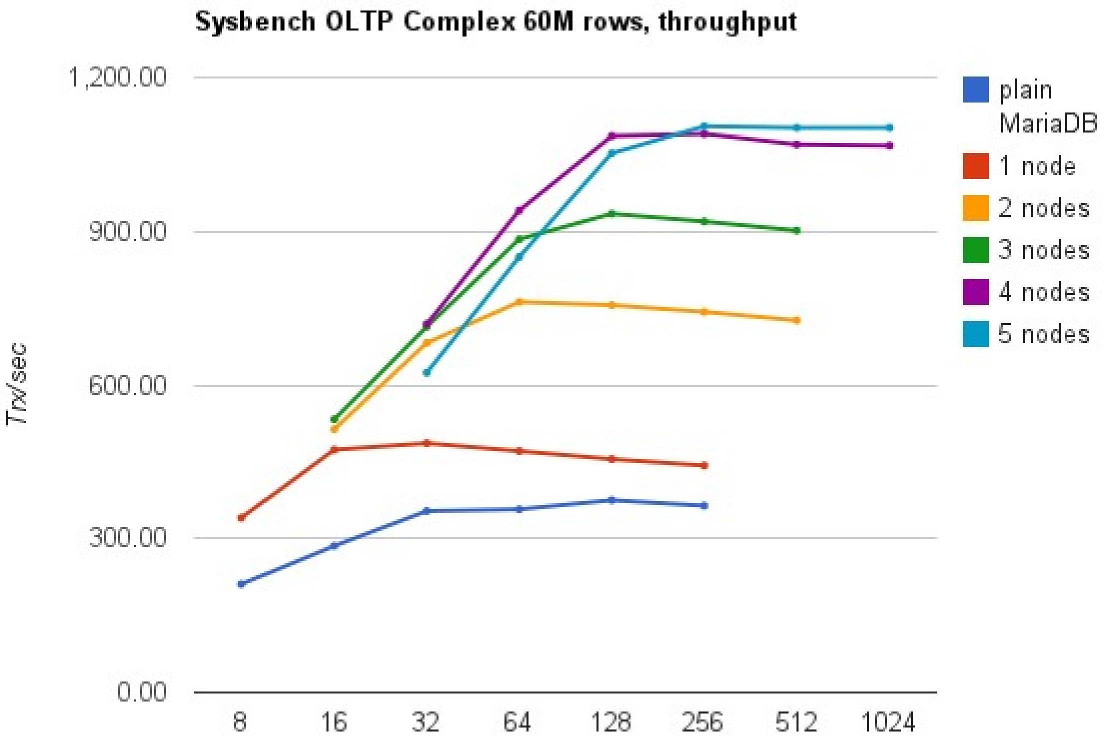


- Yellow, Red are equal
  -> No overhead or bottleneck from Galera replication!
- Green, Brown = writing to 2 and 3 masters
  -> scale-out for read-write workload!
  - Top shows 700% CPU util (8 cores)

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

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

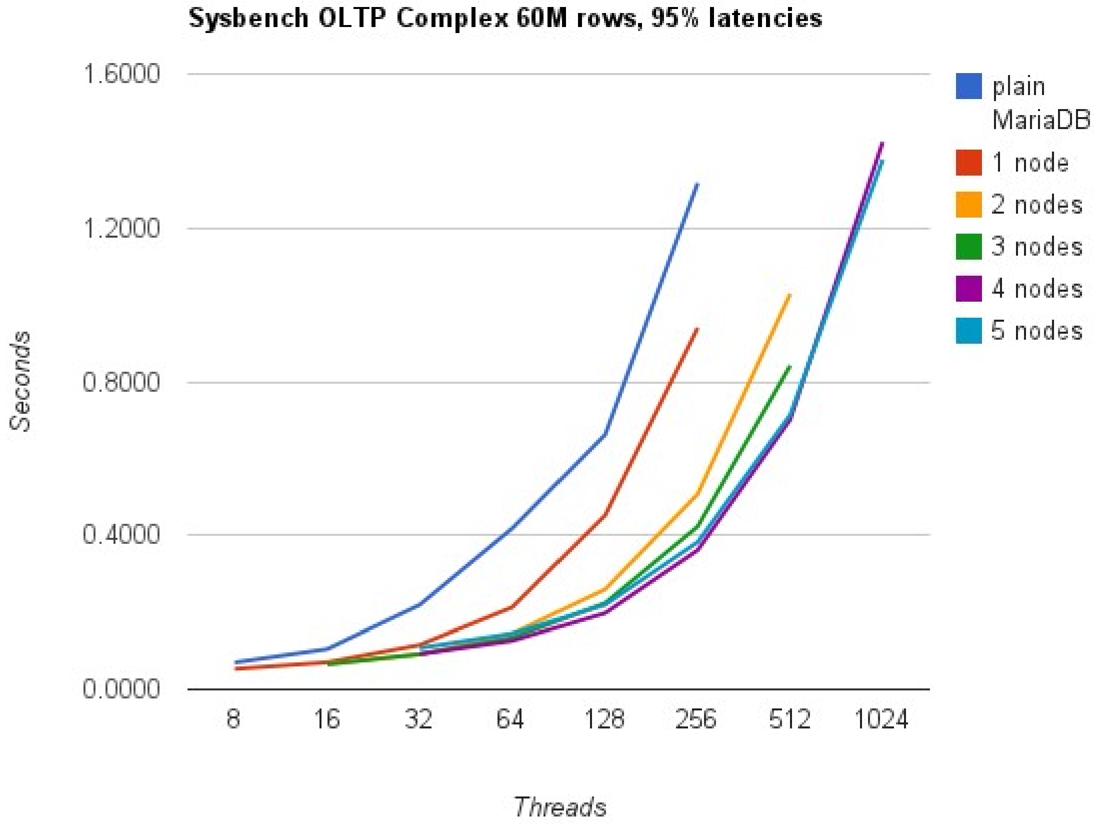
- EC2 w local disk
  - Note: pretty poor I/O here
- Blue vs red: turning off innodb_flush_log_at_trx_commit gives 66% improvement
- Scale-out factors:
  2N = 0.5 x 1N
  4N = 0.5 x 2N
- 5th node was EC2 weakness. Later test scaled a little more up to 8 nodes



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

Multi-threaded slave. Out-of-order slave commits.

Multi-thread slave

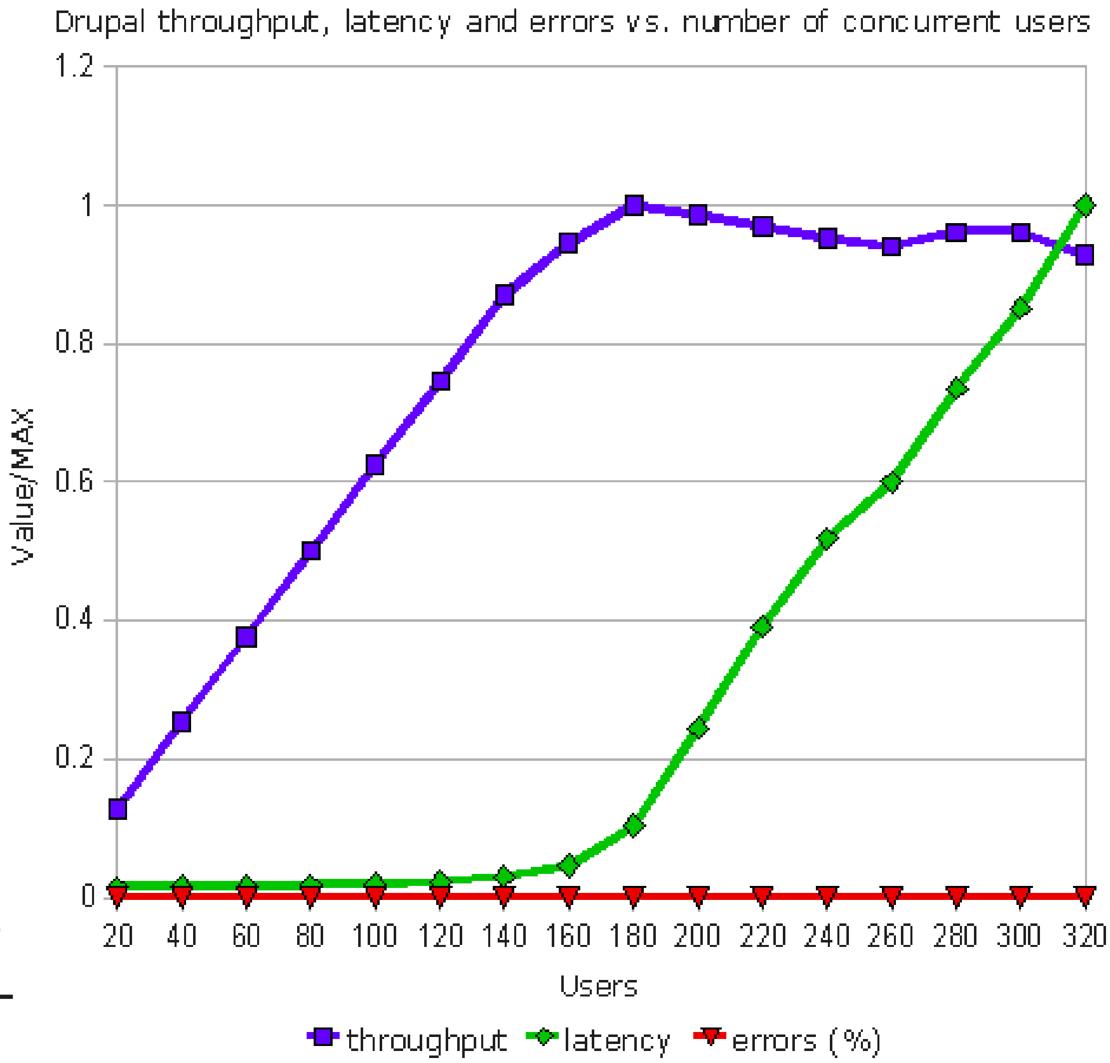
- For memory-bound workload, multi-threaded slave provides no benefit (there's nothing to fix)
- For disk-bound, multi-threaded slave helps. 2x better or more.

Out-of-order commits

- By default slave applies transactions in parallel, but preserves commit order
- OOOC is possible: wsrep_provider_options="replicator.commit_order=1"
- Not safe for most workloads, ask your developers
- Seems to help a little, in some case, but if you're really I/O bound then not
- Default multi-threaded setting is so good, we can forget this option

Drupal on Galera: baseline w single server

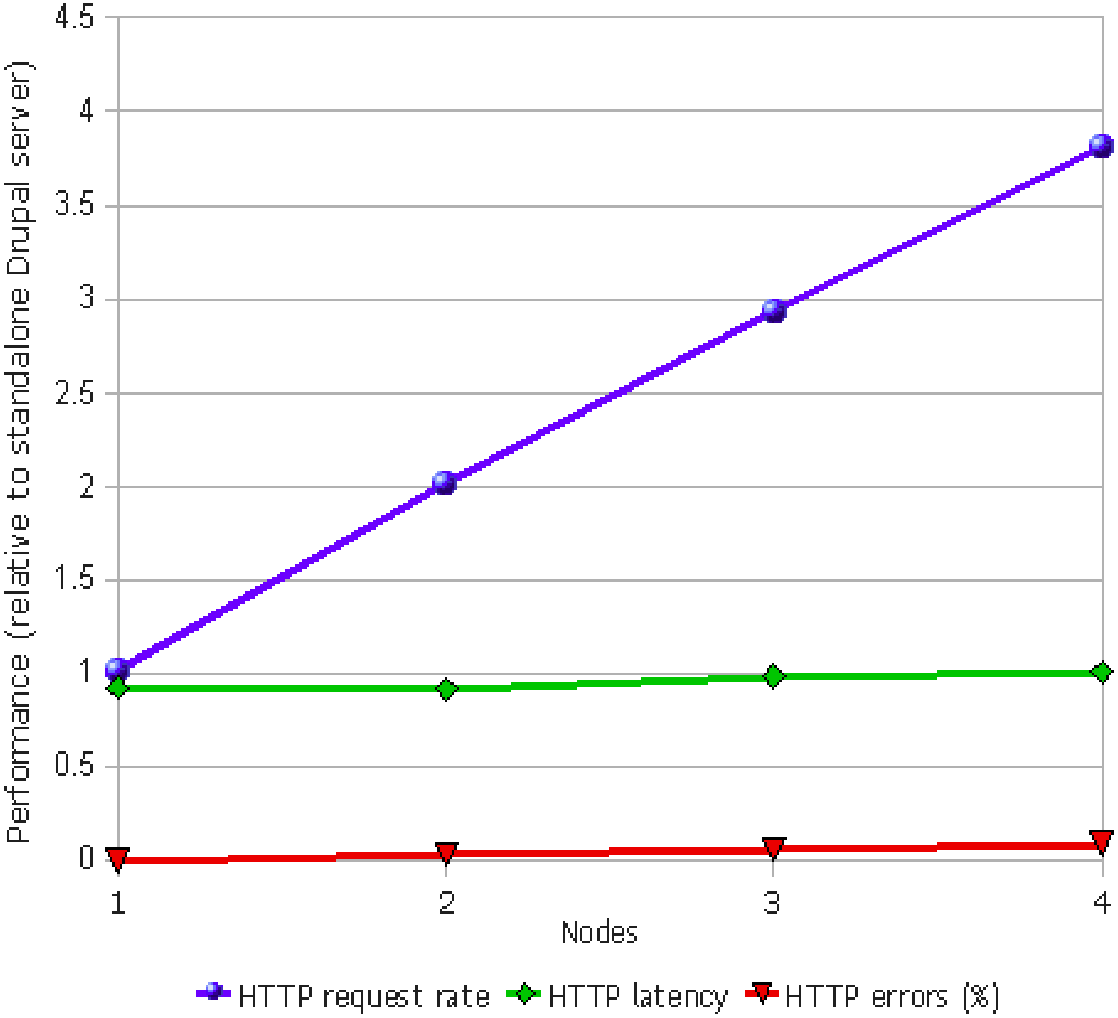
- Drupal, Apache, PHP, MySQL 5.1
- JMeter
  - 3 types of users: poster, commenter, reader
  - Gaussian (15, 7) think time
- Large EC2 instance
- Ideal scalability: linear until tipping point at 140-180 users
  - Constrained by Apache/PHP CPU utilization
  - Could scale out by adding more Apache in front of single MySQL



http://codership.com/content/scaling-drupal-stack-galera-part-2-mystery-failed-login

Drupal on Galera: Scale-out with 1-4 Galera nodes (tps)

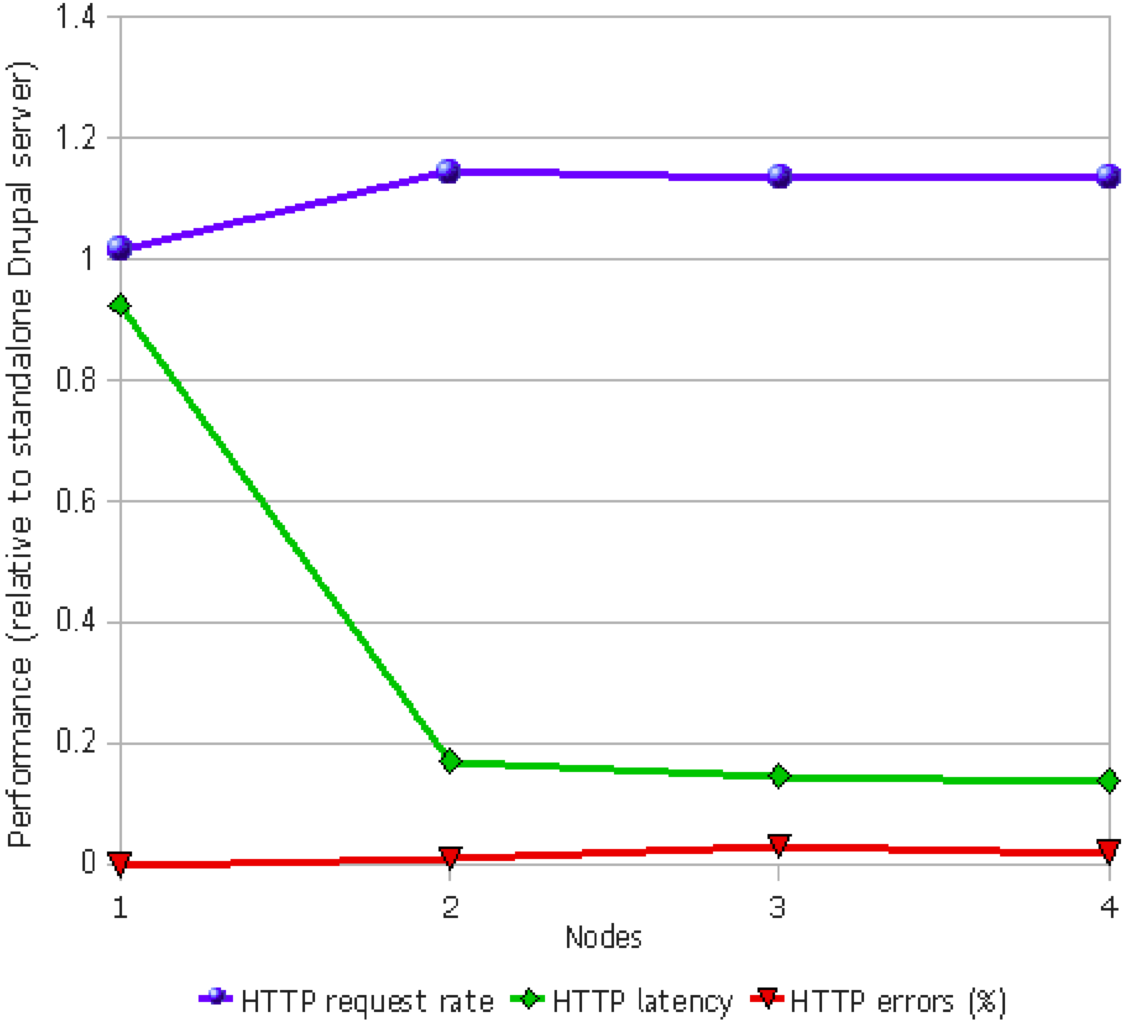
- Drupal, Apache, PHP, MySQL 5.1 w Galera
- 1-4 identical nodes
  - Whole stack cluster
  - MySQL connection to localhost
- Multiply nr of users
  - 180, 360, 540, 720
- 3 nodes = linear scalability, 4 nodes still near-linear
- Minimal latency overhead



http://codership.com/content/scaling-drupal-stack-galera-part-2-mystery-failed-login

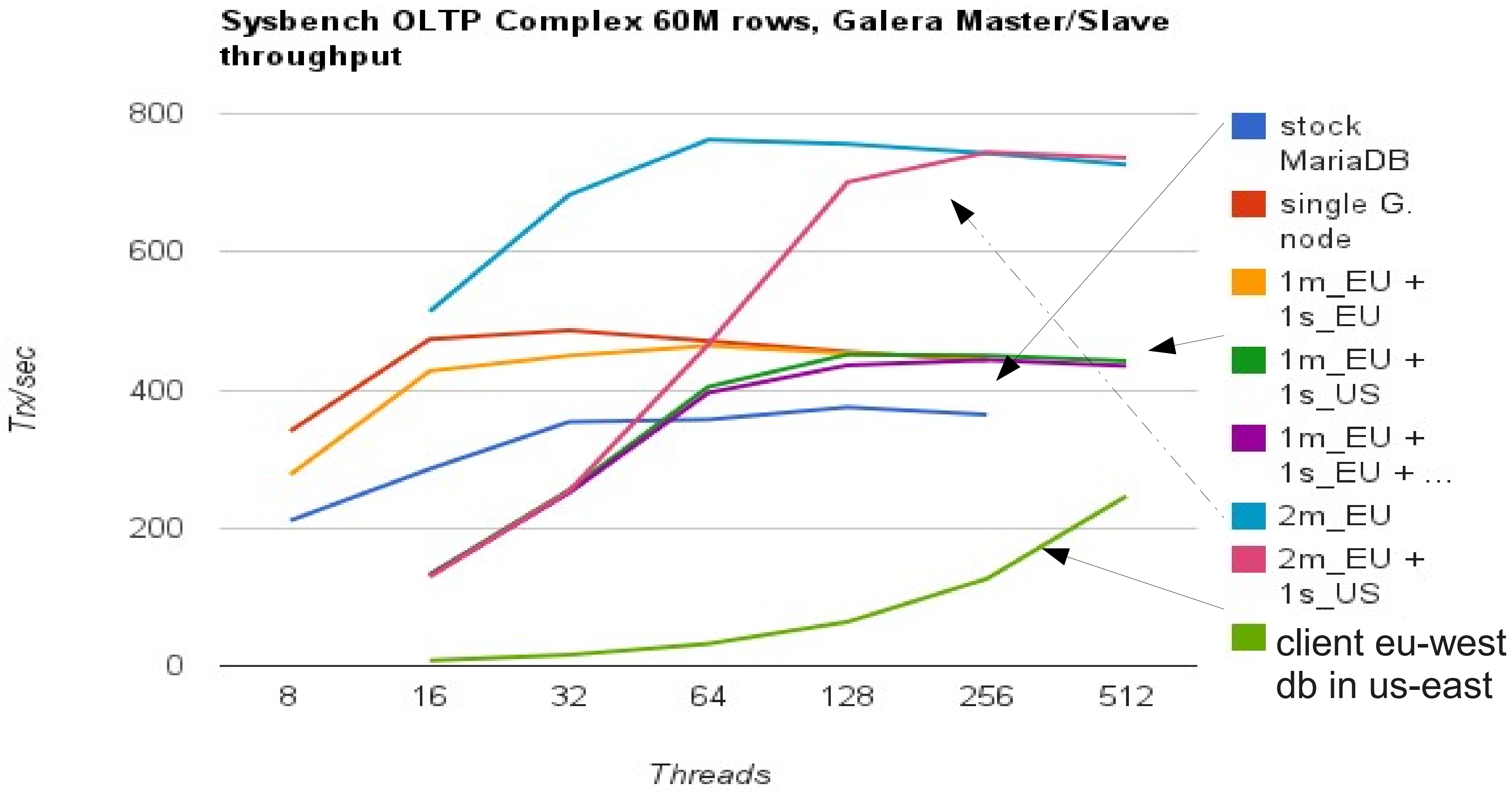
Drupal on Galera: Scale-out with 1-4 Galera nodes (latency)

- Like before
- Constant nr of users
  - 180, 180, 180, 180
- Scaling from 1 to 2
  - drastically reduces latency
  - tps back to linear scalability
- Scaling to 3 and 4
  - No more tps as there was no bottleneck.
  - Slightly better latency
  - Note: No overhead from additional nodes!



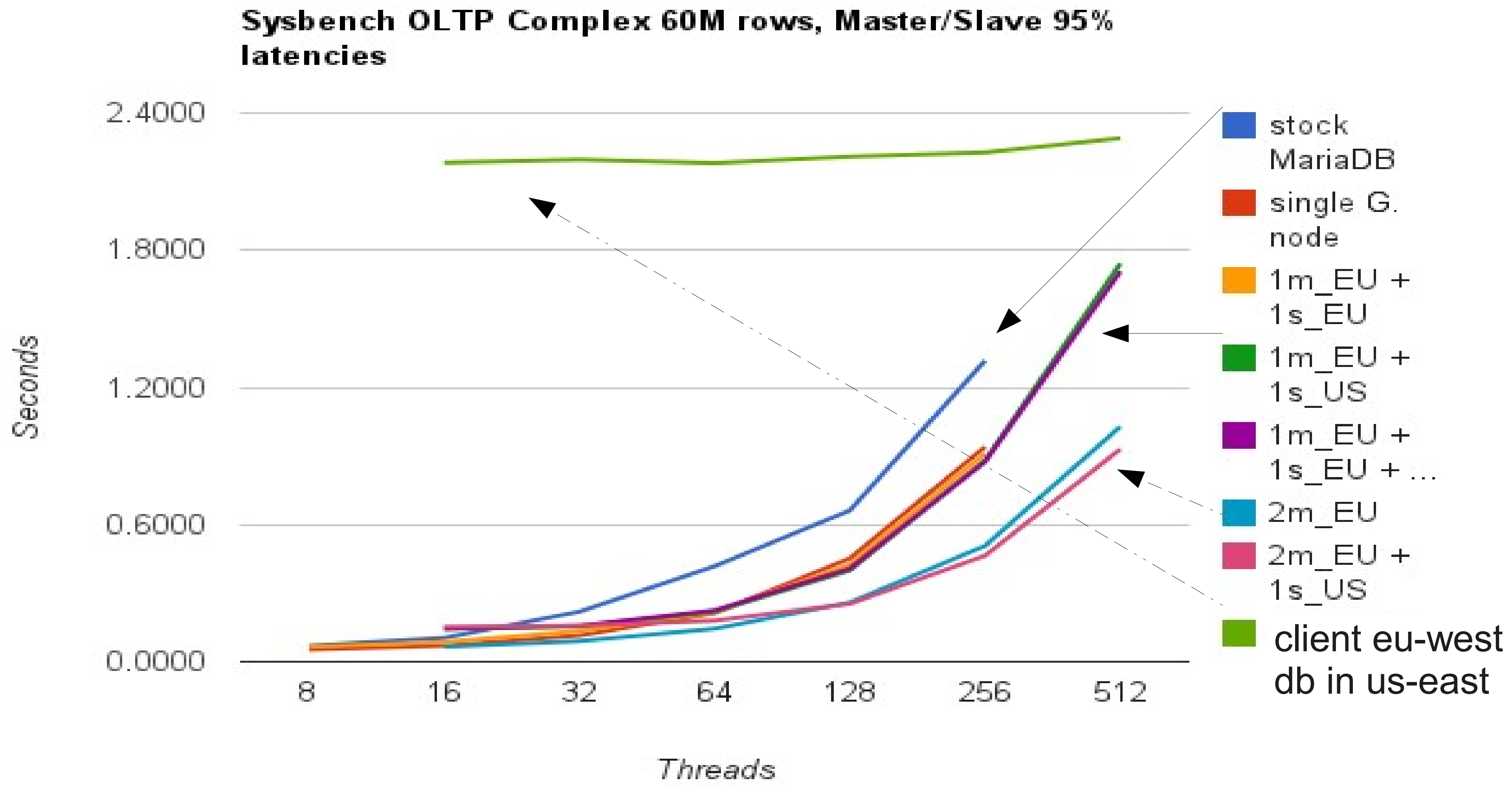
http://codership.com/content/scaling-drupal-stack-galera-part-2-mystery-failed-login

WAN replication, EC2 eu-west + us-east, tps



http://codership.com/content/synchronous-replication-loves-you-again

WAN replication, EC2 eu-west + us-east, latency



http://codership.com/content/synchronous-replication-loves-you-again

WAN adds some commit latency, that's all

Alex:

- EU-west <-> US-east:
  - 90 ms
  - "best case"

Vadim:

- EU <-> JPN:
  - 275 ms
- EU <-> JPN <-> USA
  - 295 ms

You can choose latency between:

- user and web server (ok)
- web server and db (bad)
- db and db (great!)
- Master of Record (best, but app specific)

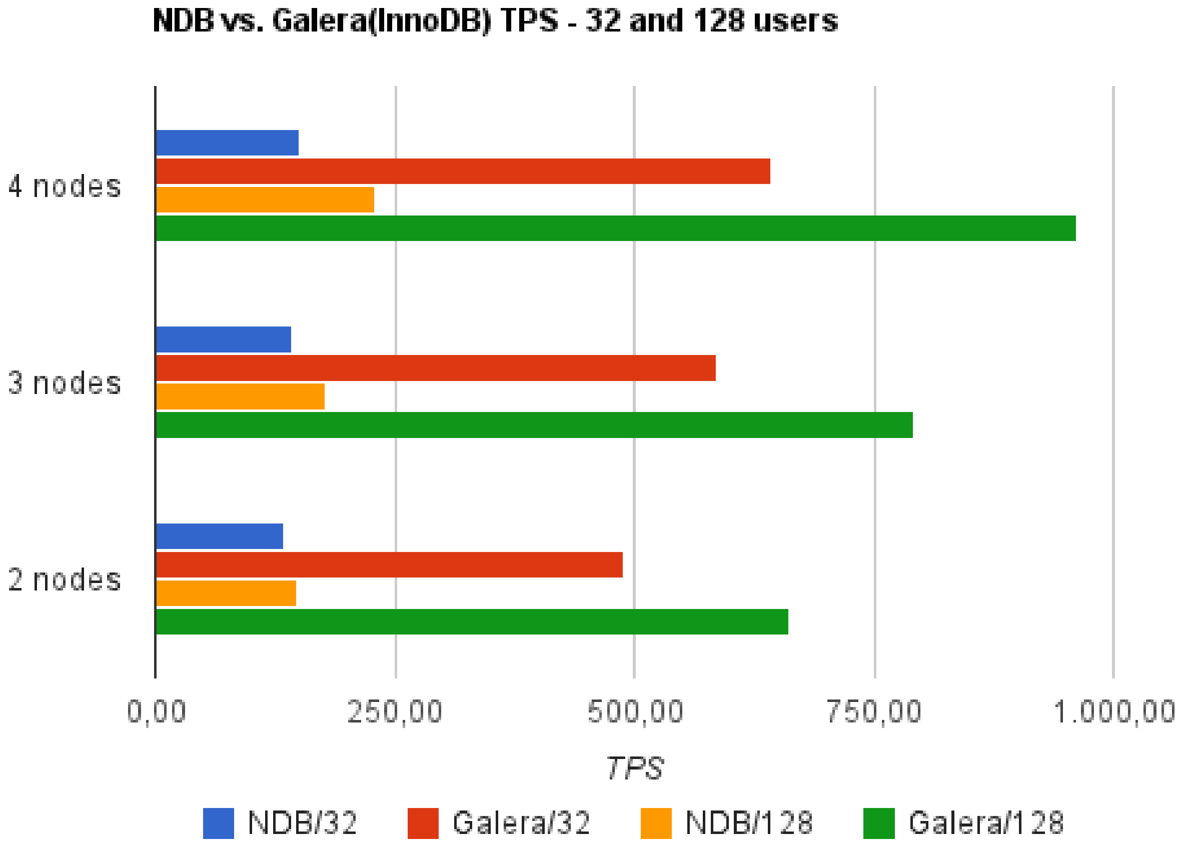
http://codership.com/content/synchronous-replication-loves-you-again
http://www.mysqlperformanceblog.com/2012/01/11/making-the-impossible-3-nodes-intercontinental-replication/

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
  - Would benefit here from PARTITION BY RANGE



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

Conclusions

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 recommended.

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

Quorum based: Majority partition wins.

Minimum 3 nodes. Minimum 3 data centers.

Negligible overhead compared to single node case (when properly configured)

Better than single node:
- No InnoDB syncs needed
- Can read & write to all nodes

Similar results both memory bound and disk bound workloads.

Whole stack load balancing: no performance penalty from gratuitiously adding Galera nodes

For a global service, network latency will always show up somewhere. Synchronous Galera replication is often an excellent choice!

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

Questions?

Thank you for listening!
Happy Clustering :-)

codership