

How to evaluate which MySQL High Availability solution best suits you

Henrik Ingo

Percona Live MySQL Conference and Expo, 2013

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



Henrik Ingo



open source technology and strategy specialist

active in MySQL-forks, Drupal communities

author of

"Open Life: The Philosophy of Open Source"

Senior Performance Architect, Nokia

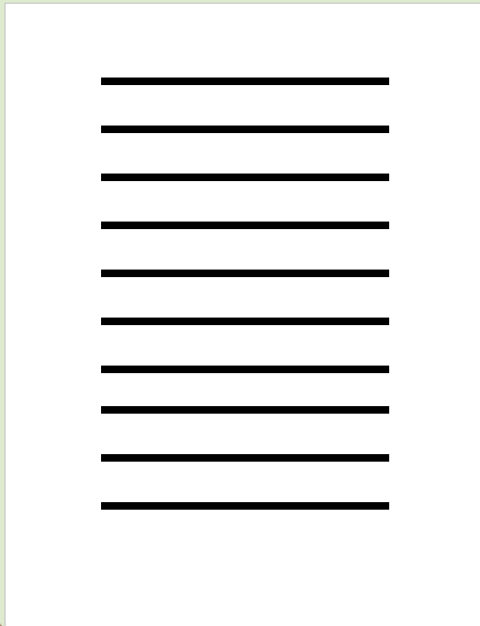
www.openlife.cc

henrik.ingo@openlife.cc



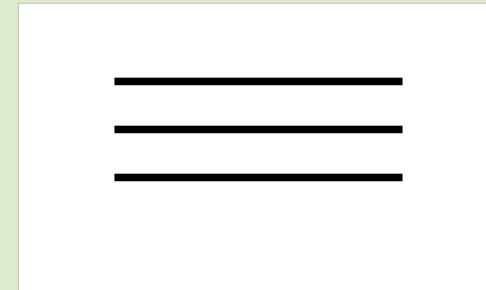
Choosing a technology (or vendor)

Long list



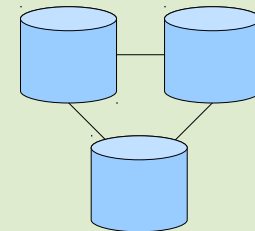
What is right
for my use case?

Short list



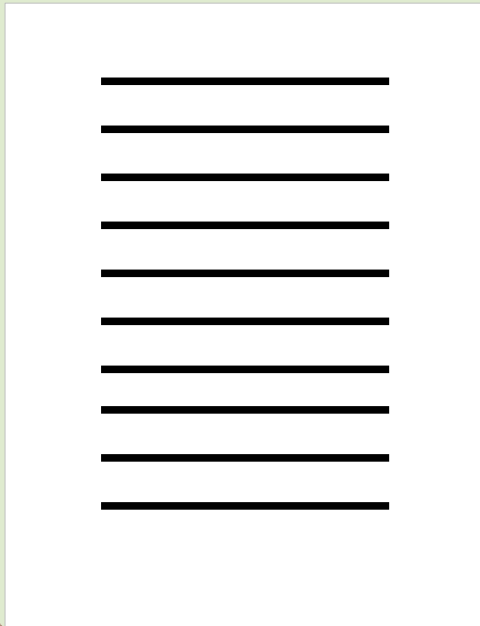
Test & bid

Solution



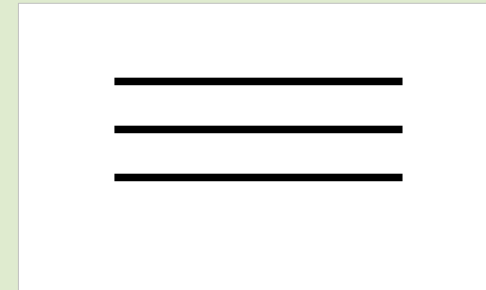
This tutorial

Long list



What is right
for my use case?

Short list



What is High Availability?



What is high availability?

Performance

Transactions / second (throughput)
Response time (latency)
Percentiles (95% - 99%)

Get any response at all (tps > 0)
Measured as percentile (99.999%)

Durability

Speaking of databases
Committed data is not lost
D in ACID

Replicas, snapshots
point in time, backups

High Availability

Clustering

Monitoring
Failover

Replication

Redundancy



Uptime

Percentile target	Max downtime per year
90%	36 days
99%	3.65 days
99.5%	1.83 days
99.9%	8.76 hours
99.99%	52.56 minutes
99.999%	5.26 minutes
99.9999%	31.5 seconds

Beyond system availability: Average downtime per user.



High Availability is Redundancy

- HA is achieved via redundancy:
 - RAID: If one disk crashes, other one still works
 - Clustering: If one server crashes, other one still works / can take over
 - Power: In case a fuse blows, have another power input
 - Network: If a switch/NIC crashes, have a second network route
 - Geographical: If a datacenter is destroyed (or just disconnected), move all computation to another data center.
 - Biological: If you lose a kidney, you have another one left.



Redundancy

Making data available



Durability

- Data is stored on physical disks
 - Is it really written to the disk?
 - Also: Written in transactional way, to guarantee
 - atomicity
 - integrity
 - crash safety

"Durability is an interesting concept. If I sync a commit to disk, the transaction is said to be durable. But if I now take a backup, then it is even more durable.

- Heikki Tuuri, MySQL Conference 2009



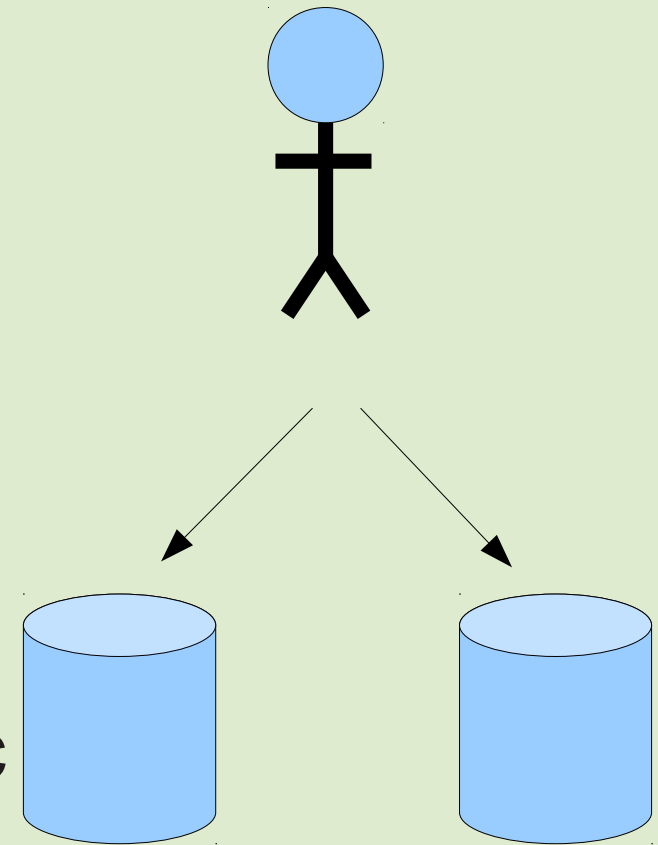
High Availability for databases

- HA is harder for databases
 - Must make both **HW resources and data redundant**
 - Not just data, but constantly changing data
 - HA means operation can continue "uninterrupted", i.e. not by restoring a backup to a new server



Redundancy through Client side XA transactions

- Client writes to 2 independent but identical databases
- Example: HA-JDBC
- No replication anywhere
- Sounds simple
- Got many databases out of sync
- Not covered in this talk



Redundancy through shared storage

- Requires specialist hardware

- e.g. SAN

- Complex to operate?

<http://www.percona.com/about-us/mysql-white-paper/causes-of-downtime-in-production-mysql-servers/>

- One set of data

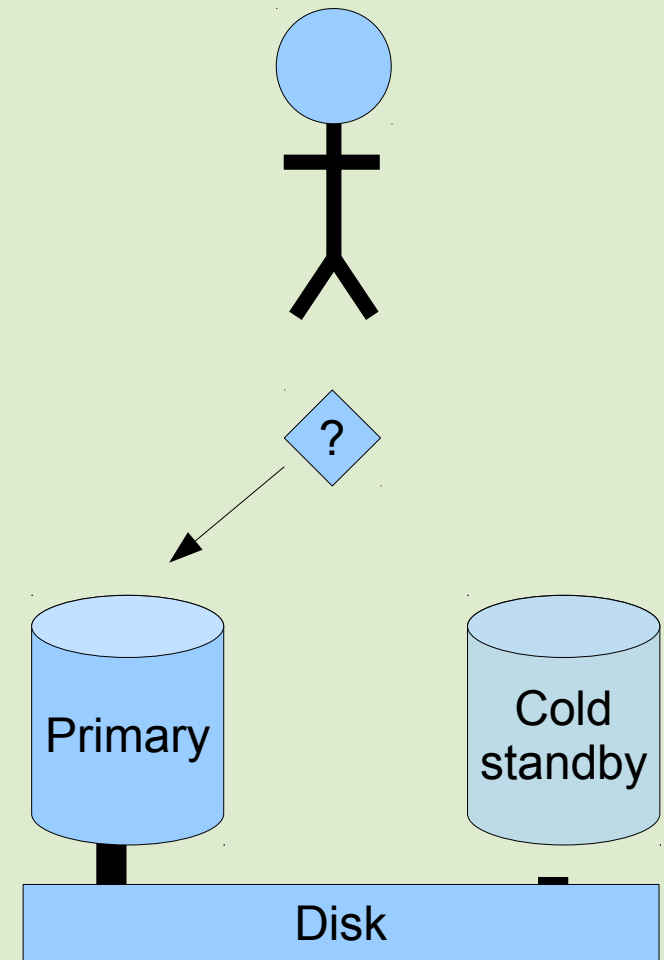
- Single Point of Failure

- Cold standby

- Failover 1-30 minutes

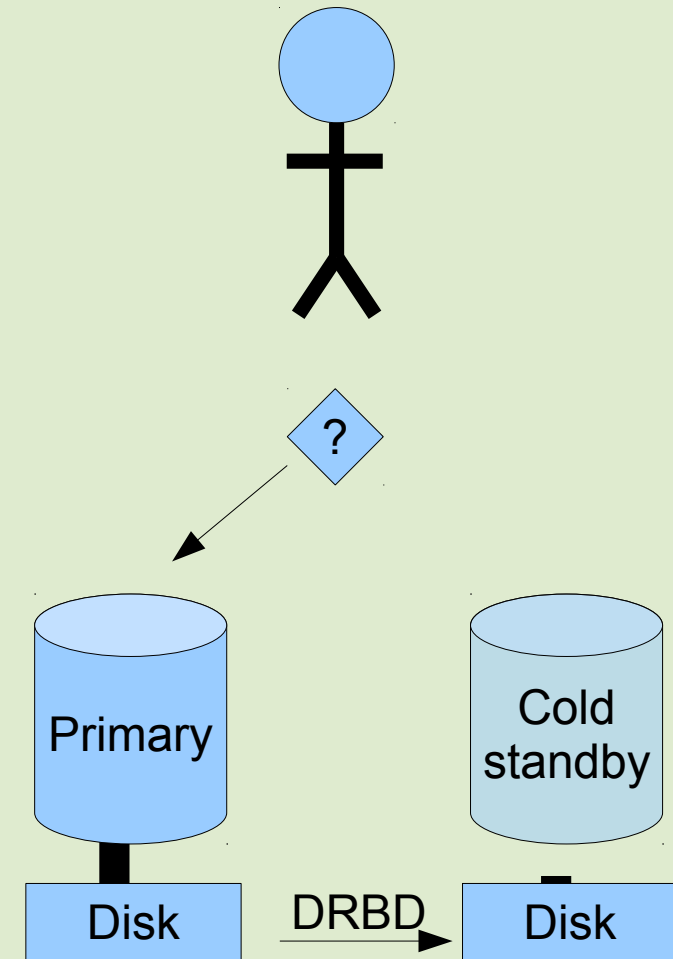
- No scale-out

- Active / Active: Oracle RAC, ScaleDB



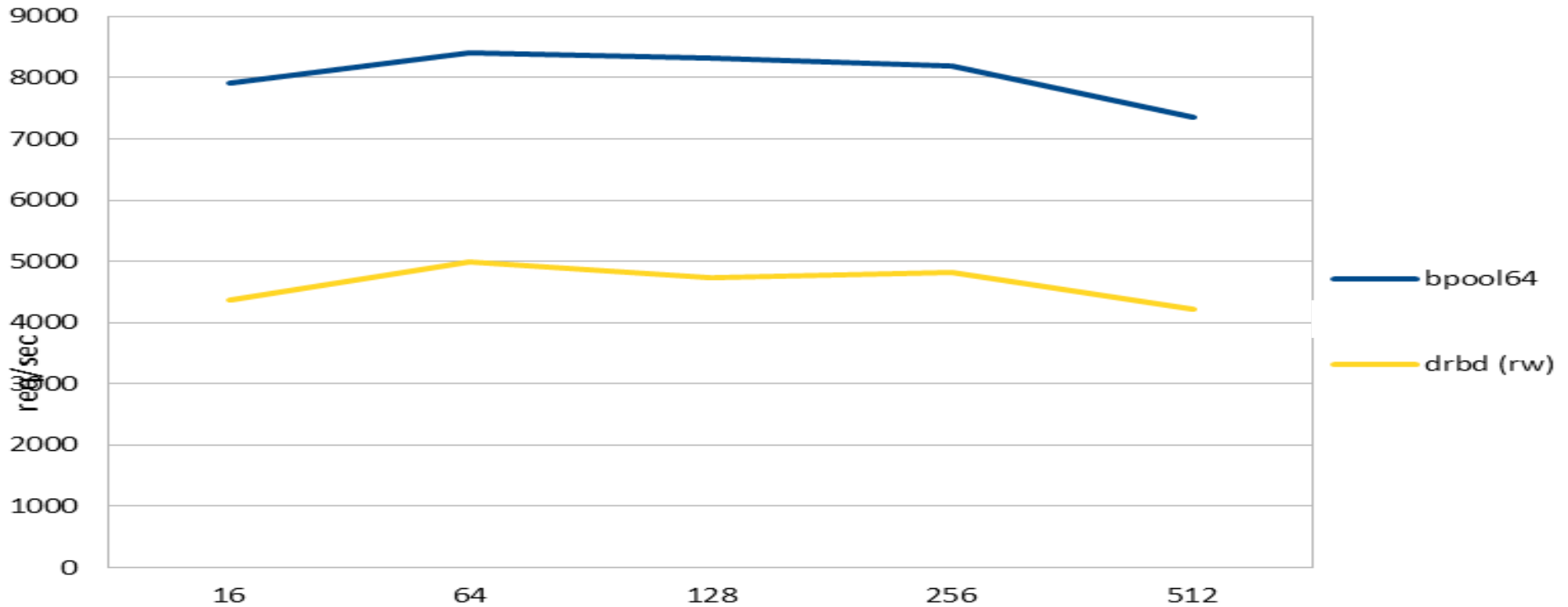
Redundancy through disk replication

- DRBD ("RAID over Ethernet")
 - Linux sysadmin vs DBA skills
 - or SAN-SAN replication
- Synchronous
- Second set of data inaccessible
- Cold standby
 - Failover 1-30 minutes



DRBD vs Single node

req/sec w smaller buffer pool



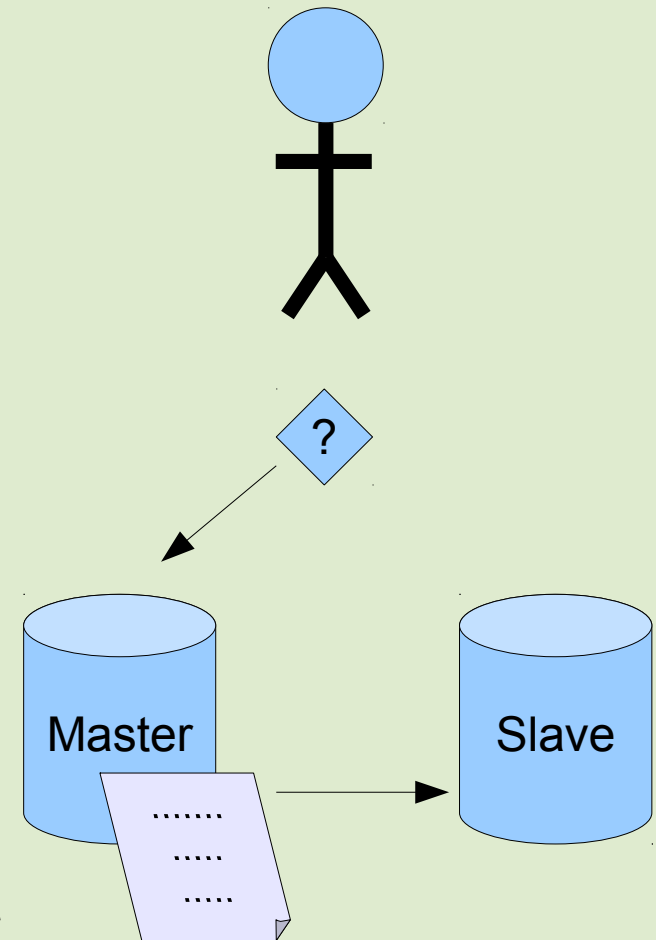
60% of single node performance

Minimum latency 10x higher but average is not so bad (not shown)



Redundancy through MySQL replication

- Replication at the RDBMS layer
 - MySQL
 - Tungsten Replicator
 - Galera
 - MySQL NDB Cluster
- Storage requirement multiplied
- Includes potential for scaling out



So what is MySQL Replication?

- Statement based, or Row based (5.1+)
- Asynchronous
- Semi Synchronous plugin in 5.5+
- MySQL 5.6
 - Global Transaction ID
 - Server UUID
 - Ignore (master) server-ids
 - Per-schema multi-threaded slave
 - Checksums
 - Crash safe binlog and relay-log
 - Delayed replication
 - <http://dev.mysql.com/doc/refman/5.6/en/mysql-nutshell.html>
- Due to the nature of replication, tools like pt-table-checksum and pt-table-sync are important part of the picture!



Inside the binary log (SBR)

```
> mysqlbinlog mysql-bin.*
[...]
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 240
#120331 0:54:56 server id 1 end_log_pos 339 Query thread_id=6 exec_time=0 error_code=0
use test/*!*/;
SET TIMESTAMP=1333144496/*!*/;
SET @@session.pseudo_thread_id=6/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=1, @@session.unique_checks=1,
@@session.autocommit=1/*!*/;
SET @@session.sql_mode=1574961152/*!*/;
SET @@session.auto_increment_increment=1, @@session.auto_increment_offset=1/*!*/;
/*!\C latin1 *//*!*/;
SET @@session.character_set_client=8,@@session.collation_connection=8,@@session.collation_server=8/*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
INSERT INTO testnumber VALUES (1334)
/*!*/;
DELIMITER ;
DELIMITER /*!*/;
ERROR: File is not a binary log file.
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@@OLD_COMPLETION_TYPE*/;
```



Row based replication event

```
> mysqlbinlog mysql-bin.*
DELIMITER /*!*/;
# at 4
#120331 0:52:23 server id 1 end_log_pos 240 Start: binlog v 4, server v 5.2.4-MariaDB-rpl-mariadb98~maverick-log
created 120331 0:52:23 at startup
# Warning: this binlog is either in use or was not closed properly.
ROLLBACK/*!*/;
BINLOG '
Fyt2Tw8BAAAA7AAAAPAAAAABAAQANS4yLjQtTWFyaWFEQi1ycGwtbWFyaWFkYjk4fm1hdmVyaWNr
LWxvZwAAAAAAAAAAAAAAAAAXK3ZPEzgNAAgAEgAEBAQEEgAA2QAEgGgAAAAICAgCAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAA='
/*!*/;
```

- Yes, you can execute that statement against MySQL!
- MariaDB 5.3 and MySQL 5.6 can also show the original SQL statement



SHOW SLAVE STATUS

```
mysql> show slave status\G
***** 1. row *****
Slave_IO_State: Waiting for master to send event
Master_Host: server1
Master_User: repluser
Master_Port: 3306
...
Master_Log_File: server1-binlog.000008      <- io_thread (read)
Read_Master_Log_Pos: 436614719             <- io_thread (read)
Relay_Log_File: server2-relaylog.000007    <- io_thread (write)
Relay_Log_Pos: 236                         <- io_thread (write)
Relay_Master_Log_File: server1-binlog.000008 <- sql_thread
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
...
Exec_Master_Log_Pos: 436614719            <- sql_thread
...
Seconds_Behind_Master: 0
```



MySQL 5.6 binary log

```
$ mysqlbinlog mysql-bin.000001
```

```
...
```

```
# at 207
```

```
#120331 22:38:30 server id 1 end_log_pos 282 Query thread_id=1 exec_time=0
```

```
error_code=0
```

```
SET TIMESTAMP=1333222710/*!*/;
```

```
BEGIN
```

```
/*!*/;
```

```
# at 282
```

```
#120331 22:38:30 server id 1 end_log_pos 377 Query thread_id=1 exec_time=0
```

```
error_code=0
```

```
SET TIMESTAMP=1333222710/*!*/;
```

```
insert into t1 values (1)
```

```
/*!*/;
```

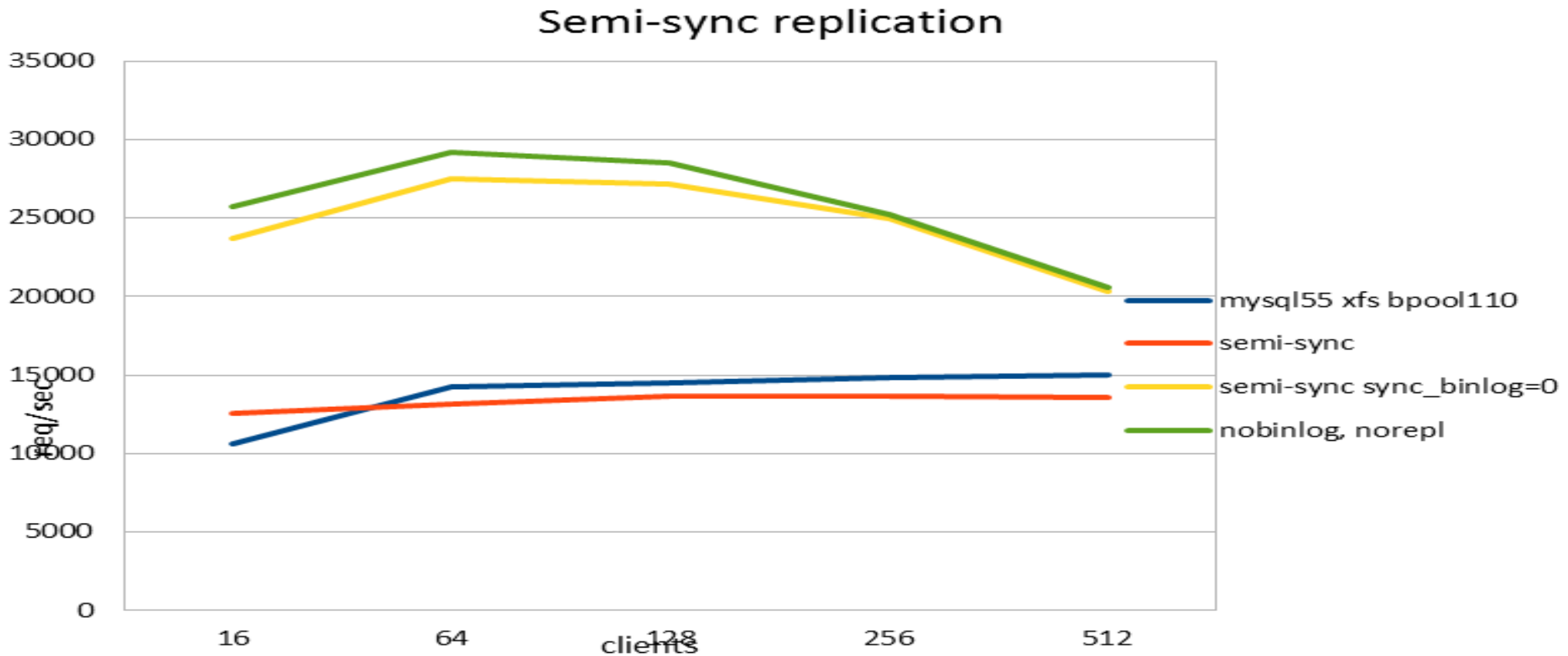
```
# at 377
```

```
#120331 22:38:30 server id 1 end_log_pos 404 Xid = 10
```

```
COMMIT/*!*/;
```



Semi sync vs Single node (memory bound)



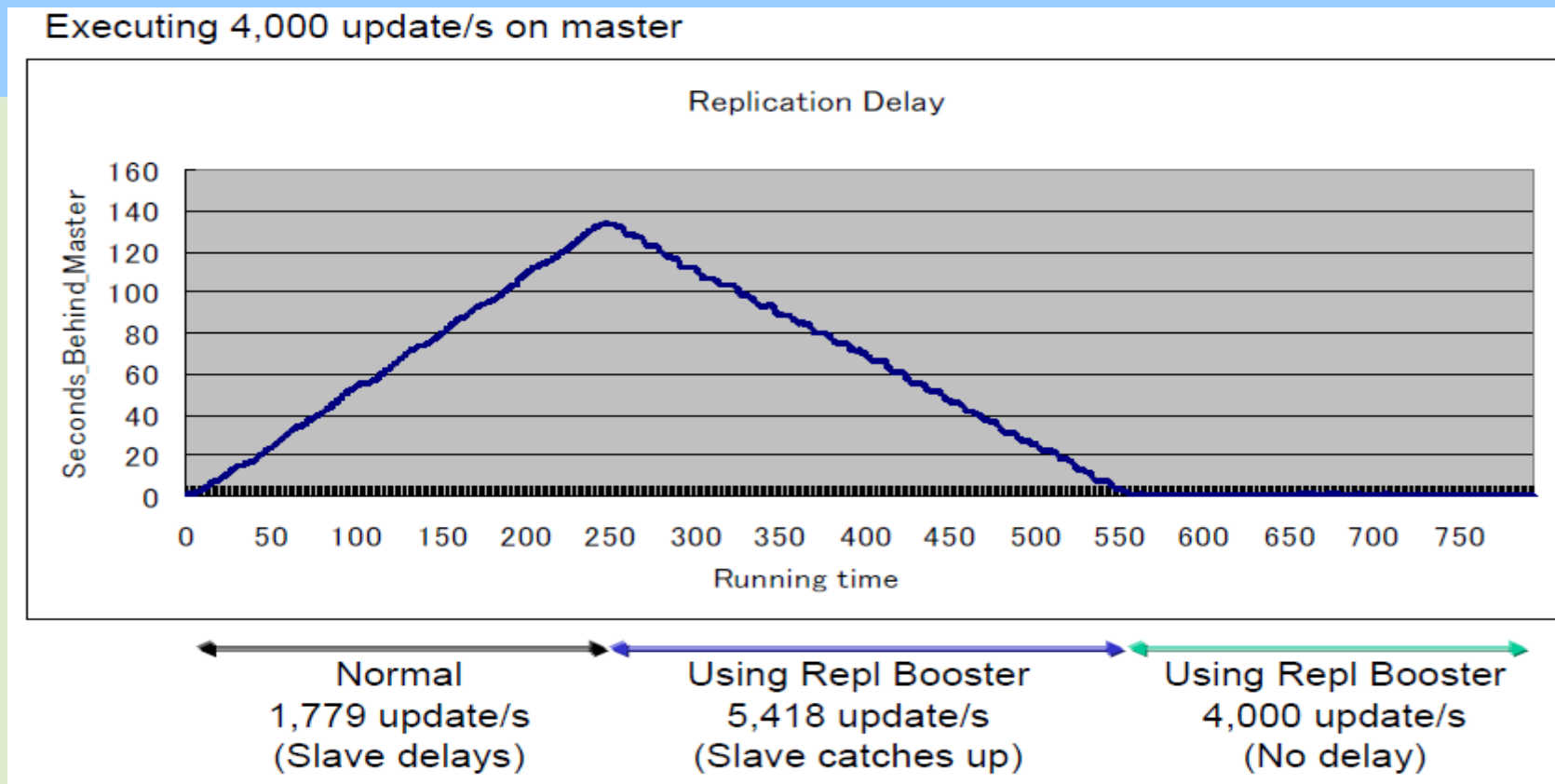
Practically no performance overhead on LAN

NOTE: Semi-sync on WAN: $\text{tps} = 1 / \text{RTT} = 10 \text{ tps!}$

Opportunity to relax sync_binlog setting (green - yellow)



Slave lag (disk bound)



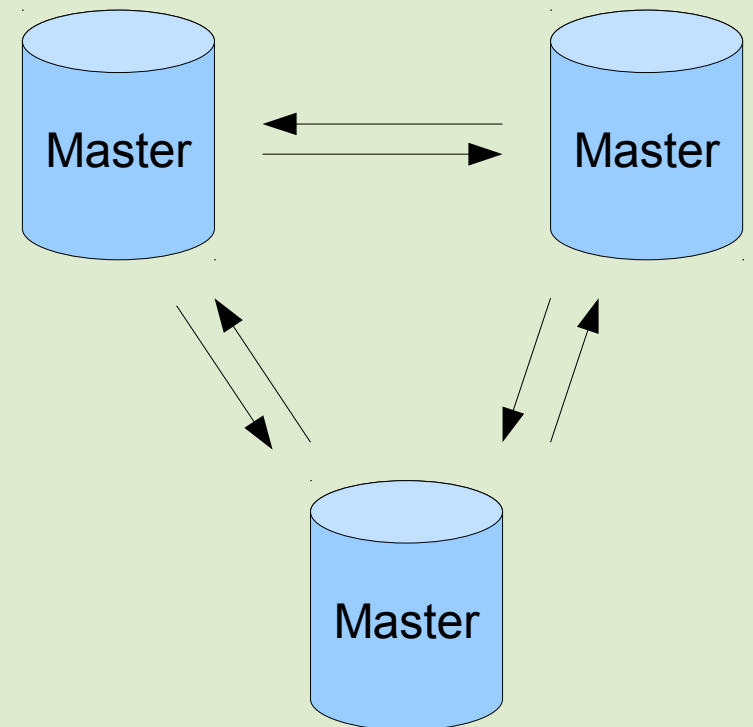
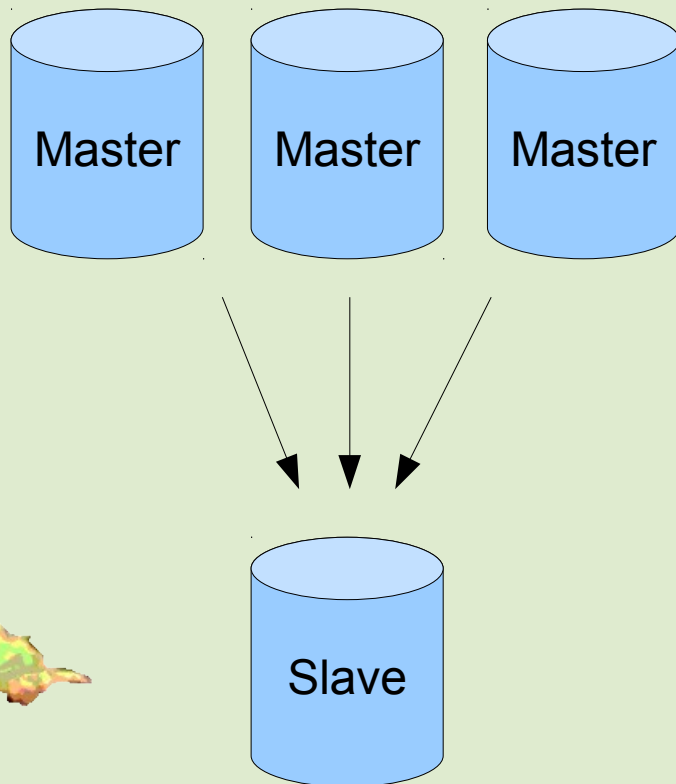
Graph and benchmark (C) Yoshinori Matsunobu, Percona Live UK 2011
<http://www.percona.com/files/presentations/percona-live/london-2011/PLUK2011-linux-and-hw-optimizations-for-mysql.pdf>

With disk bound workload (data set > RAM), slave lag is common
In practice limits master throughput 50-90%
Slave-prefetch tools combat this well. See:
Yoshinori Matsunobu, Anders Karlsson, Percona Toolkit



Multi-source replication

- MariaDB 10.0 alpha



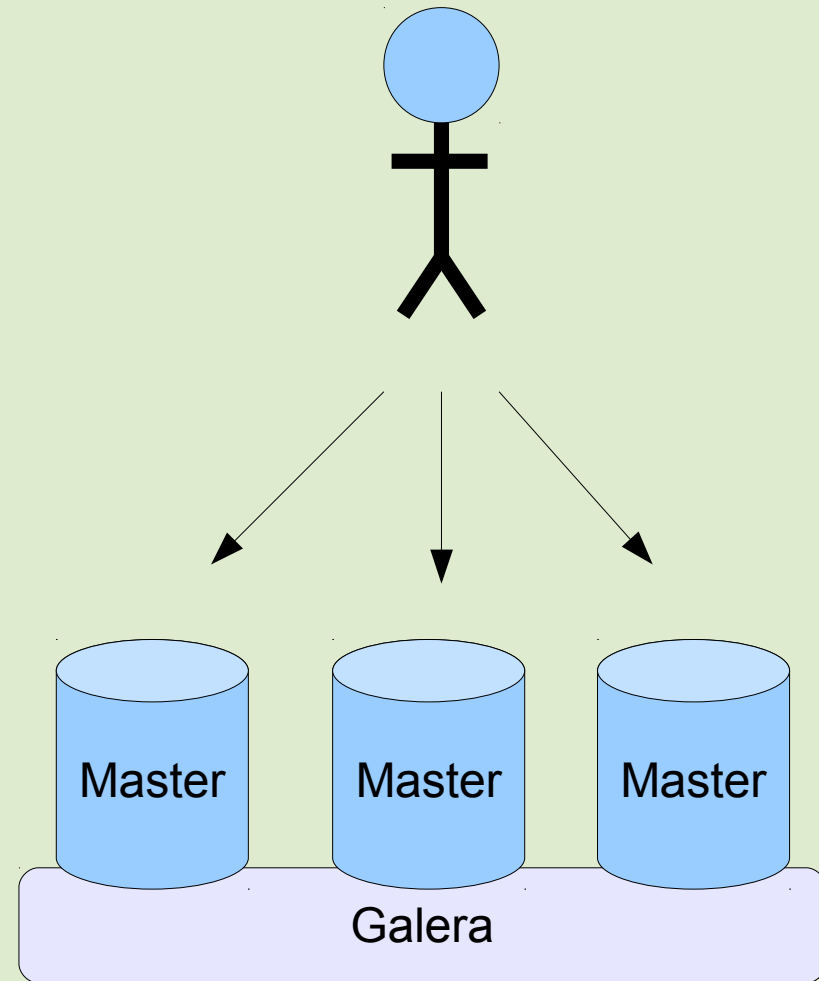
So what is Tungsten Replicator?

- Replaces MySQL Replication
 - MySQL writes binary log, Tungsten reads it and uses its own replication protocol
- Global Transaction ID
- Per-schema multi-threaded slave
- Heterogeneous replication: MySQL <-> MongoDB <-> Pg
- Multi-master
 - Including multiple masters to single slave
 - Complex topologies
- Tungsten Enterprise

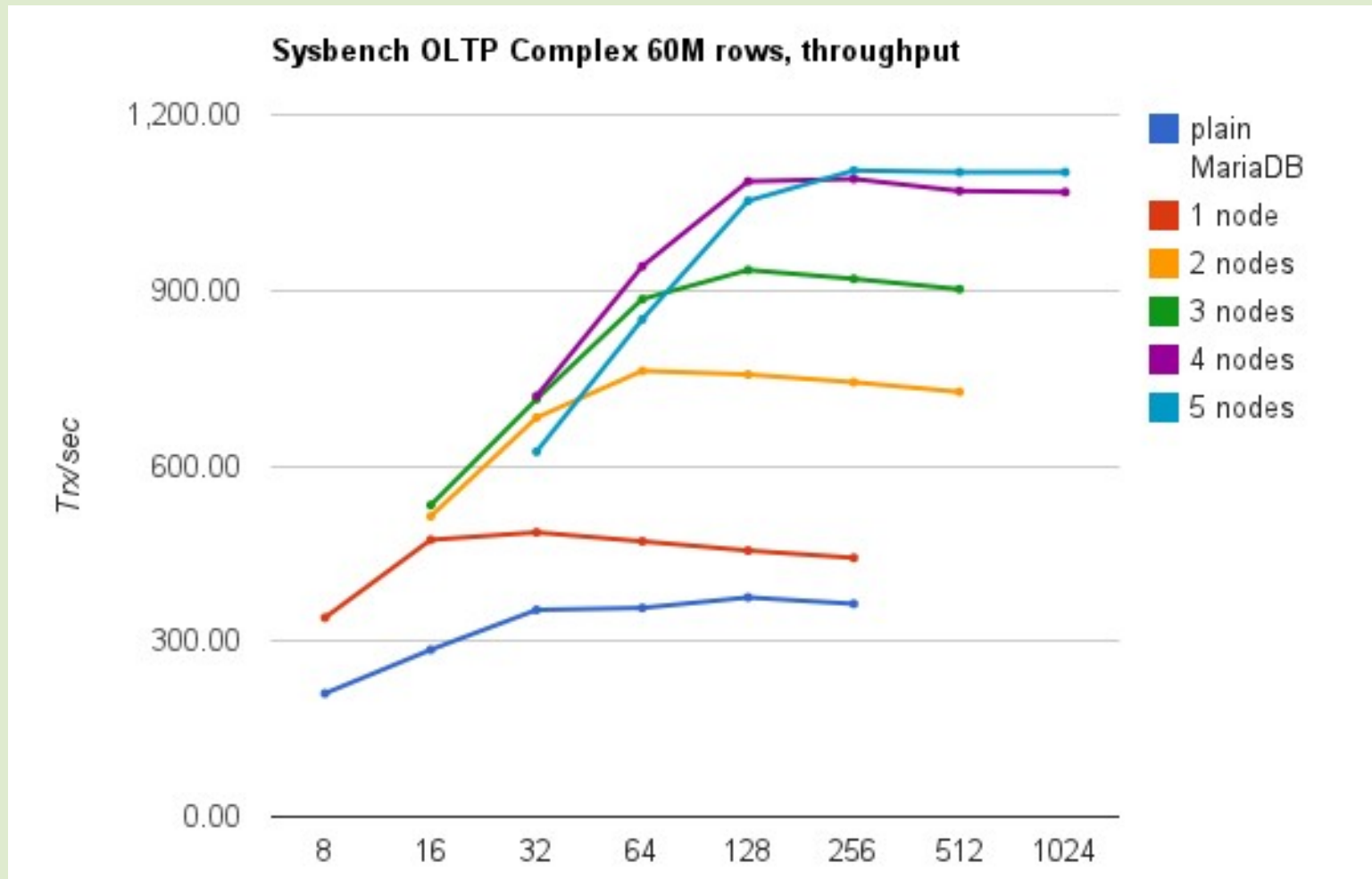


So what is Galera?

- Inside MySQL: a replication plugin (kind of)
 - Supports InnoDB only, MyISAM experimental
- Replaces MySQL replication (or you could use both)
- True multi-master, active-active
- Synchronous
- Even used over WAN: 100 - 300 ms / commit, but works in parallel
- Multi-threaded slaves, no limitation on use case
- No slave lag or integrity issues
- Automatic node provisioning
- <http://www.codership.com/downloads/download-mysqgalera>
- Percona XtraDB Cluster, MariaDB Galera Cluster



Galera w disk bound workload (EC2)



20 GB data / 6 GB buffer pool
Significant read-write scale-out up to 4 nodes!



So what is MySQL NDB Cluster?

- 3 node types: sql, data, and management.
 - MySQL node provides an interface to the data, alternate API is available: LDAP, Memcache, native NDB API
 - Data nodes aka NDB storage engine.
 - Note: Different features and performance compared to InnoDB! (Consider training.)
 - Transactions are synchronously written to 2 nodes (or more) aka replicas.
 - Transparent sharding:
Partitions = data nodes / replicas
 - Automatic node provisioning, online re-partitioning
- High-performance for some workloads: 1 billion updates / min



Summary of Replication Performance

- SAN has "some" latency overhead compared to local disk. Can be great for throughput.
- DRBD = 50% performance penalty
- Replication, when implemented correctly, has no performance penalty
 - But MySQL replication w disk bound data set has single-threadedness issues!
 - Semi-sync is poor on WAN
- Galera & NDB = r/w scale-out
= **more** performance



**Dealing with failures
aka
Clustering Frameworks**

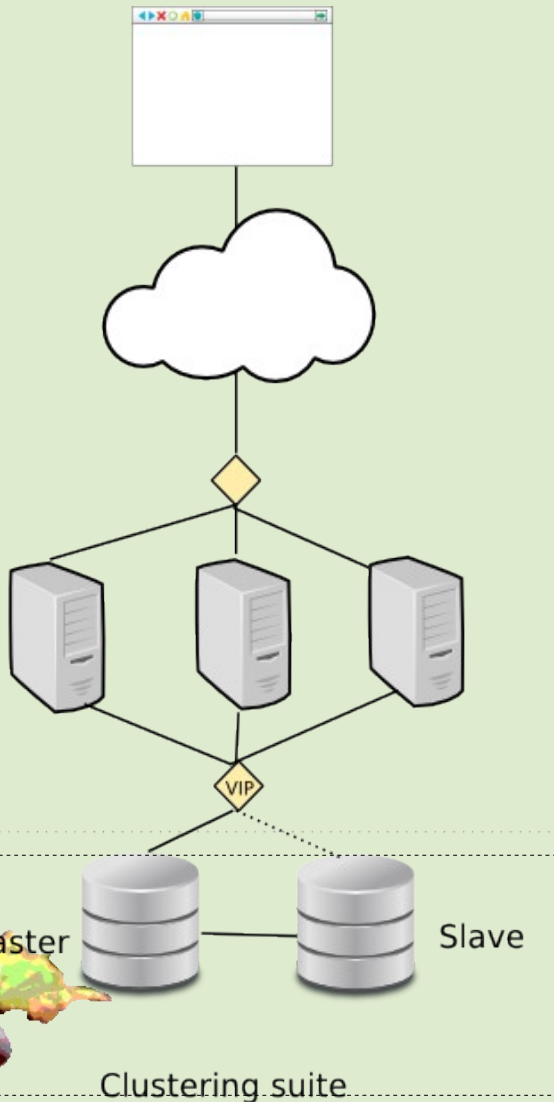


Dealing with failure

- Problem #1: How do we find out about failure?
 - Polling, monitoring, alerts...
 - Error returned to and handled in client side
- Problem #2: What should we do about it?
 - Direct requests to the spare nodes (or datacenters)
- Problem #3: Not as easy as you'd think, remember to protect data integrity:
 - Master-slave is unidirectional: Must ensure there is only one master at all times.
 - DRBD and SAN have cold-standby: Must mount disks and start mysqld.
 - In all cases must ensure that 2 disconnected replicas cannot both commit independently. (split brain)

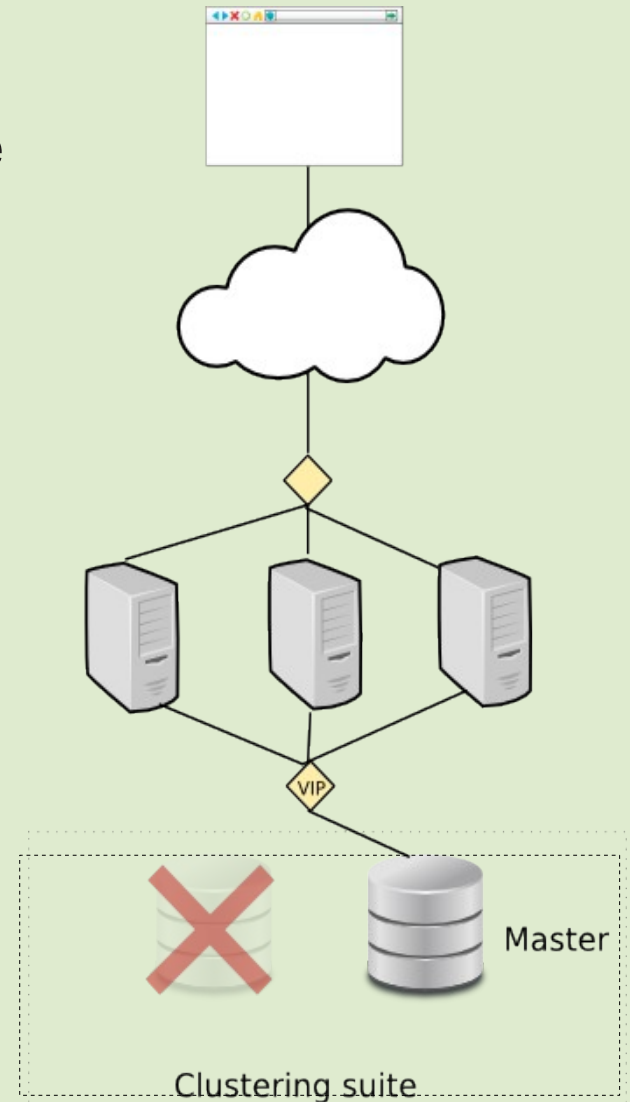


Clustering frameworks



- VIP points to Master
- External clustering suite polls all nodes for health
- In case of Master error, move VIP to Slave
- + other management tasks
- Solutions:
 - Automated Replication Failover
 - Cluster Suites
 - VM based

Failover



MySQL specialist solutions

- When using MySQL replication
 - *NEW: mysqlfailover, mysqlrpladmin*
 - MySQL-MMM, MySQL-MHA, Severalnines
 - Tungsten Enterprise to manage Tungsten Replicator
- Specialized solutions - understand MySQL and MySQL replication



So what is MySQL-MMM?

- You have to setup all nodes and replication manually
- MMM gives Monitoring + Automated and manual failover on top
- Architecture consists of Monitor and Agents
- Typical topology:
 - 2 master nodes
 - Read slaves replicate from each master
 - If a master dies, all slaves connected to it are stale
- Support from Open Query and Percona
- Is there still a place for MMM?
- <http://mysql-mmm.org/>



MMM example

```
# mmm_control show
```

```
db1(192.168.0.31) master/ONLINE. Roles: writer(192.168.0.50), reader(192.168.0.51)  
db2(192.168.0.32) master/ONLINE. Roles: reader(192.168.0.52)  
db3(192.168.0.33) slave/ONLINE. Roles: reader(192.168.0.53)
```

```
# mmm_control set_offline db1
```

```
OK: State of 'db1' changed to ADMIN_OFFLINE. Now you can wait some time and check  
all roles!
```

```
mon:~# mmm_control show
```

```
db1(192.168.0.31) master/ADMIN_OFFLINE. Roles:  
db2(192.168.0.32) master/ONLINE. Roles: writer(192.168.0.50), reader(192.168.0.52)  
db3(192.168.0.33) slave/ONLINE. Roles: reader(192.168.0.51), reader(192.168.0.53)
```

Courtesy and copyright of <http://mysql-mmm.org/mysql-mmm.html>



So what is Severalnines ClusterControl?

- Origin as automated deployment of MySQL NDB Cluster
 - 4 node cluster up and running in 5 min!
- Now also supports
 - MySQL replication and Galera (and MongoDB!)
 - Semi-sync replication
 - Automated failover
 - Manual failovers, status check, start & stop of node, replication, full cluster... from single command line.
 - Monitoring
- Topology: Pair of semi-sync masters, additional read-only slaves
 - Can move slaves to new master
- Commercial closed source features: backup, online add node, rolling restart
- <http://severalnines.com/>



So what is MySQL-MHA?

- Like MMM, specialized solution for MySQL replication
 - Developed by Yoshinori Matsunobu at DeNA
 - Support from SkySQL
- Automated failover and manual failover
- Topology: 1 master, many slaves
 - Choose new master by comparing slave binlog positions
- Can be used in conjunction with other solutions
- <http://code.google.com/p/mysql-master-ha/>



So what is Tungsten Enterprise?

- Use with Tungsten Replicator
- Like "all of the above"
- Includes proxy / load balancer that can further protect slaves from accidental writes, etc...
- Closed source, commercial
- <http://continuent.com/>



Cluster suites

- Heartbeat, Pacemaker, Red Hat Cluster Suite
- Generic, can be used to cluster any server daemon
- Usually used in conjunction with Shared Disk or Replicated Disk solutions (preferred)
- Can be used with replication.
- Robust, Node Fencing / STONITH



So what is Pacemaker?

- Heartbeat v1, Heartbeat v2, Pacemaker
- Heartbeat and Corosync
- Resource Agents, Percona-PRM
- <http://www.clusterlabs.org/>
- Percona Replication Manager
 - Pacemaker agent specialized on MySQL replication
 - "Done right" (but not yet there?)
 - <https://launchpad.net/percona-prm>

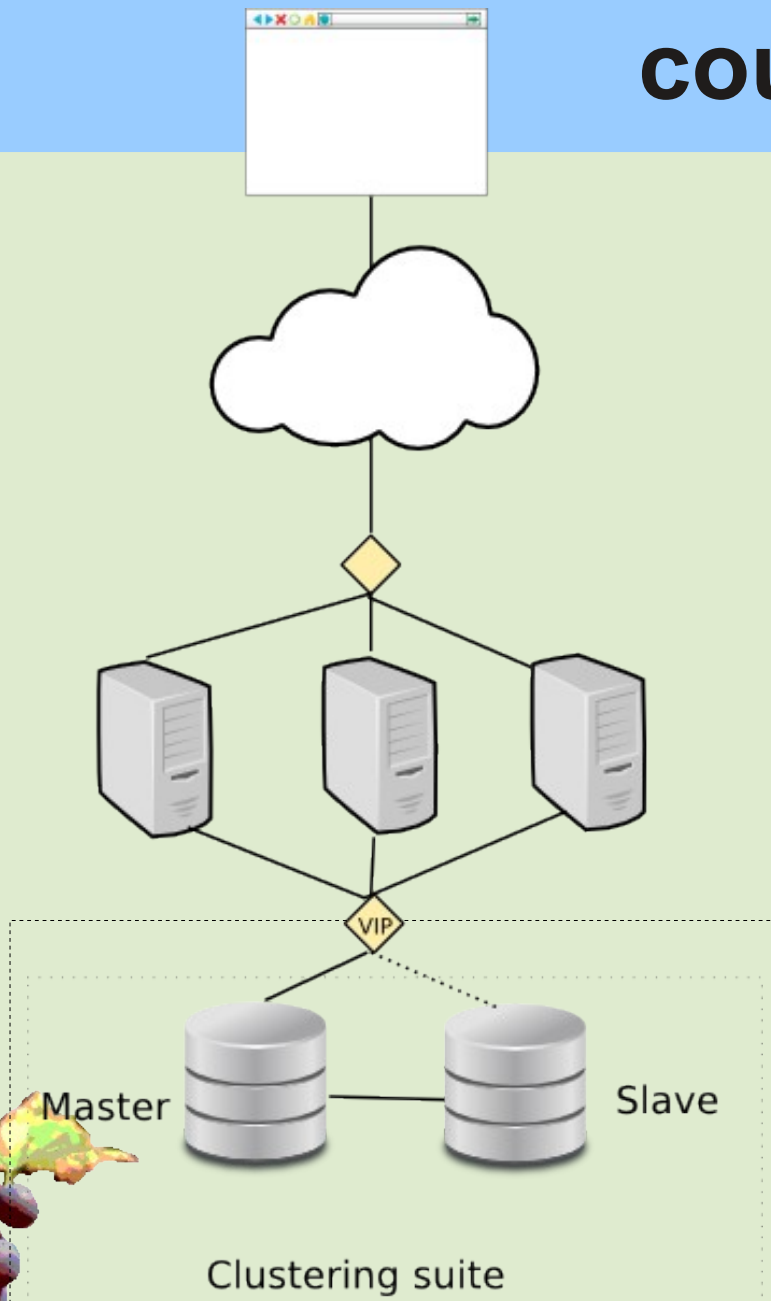


VM based

- VMWare, Oracle VM, etc can migrate / failover the entire VM guest
- Has been recommended by Oracle MySQL sales
- Rocket science!
- But... I asked for failover solution, not virtualization
- Otoh, maybe everything is virtualized anyway?

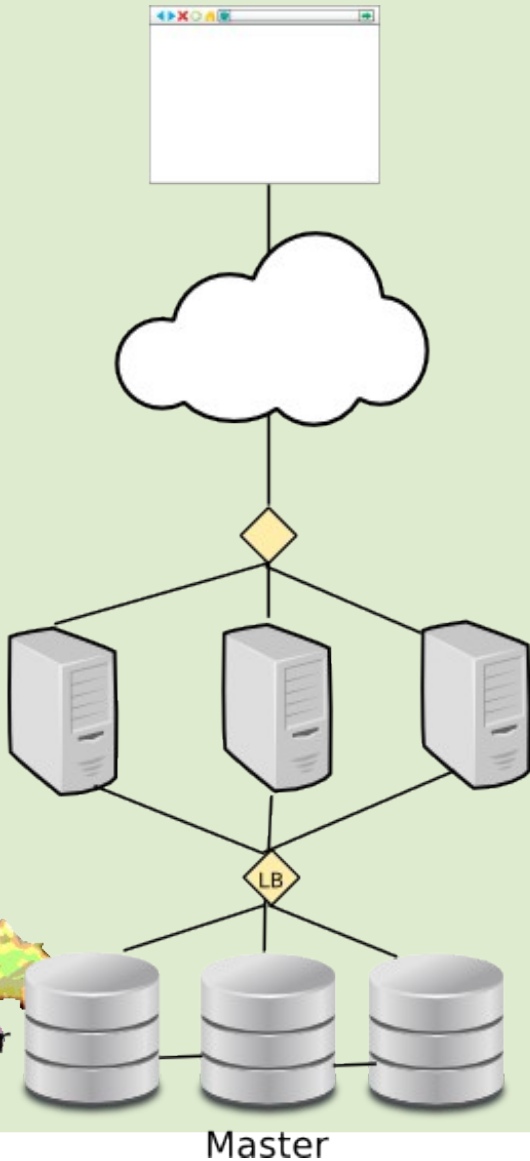


Sounds simple. What could possibly go wrong?



- Old Master must stop service (VIP, os, DB). But it is not responding, so how do you make it stop?
- Polling from the outside. Interval = 1 sec, 10 sec, 60 sec!
- What if replication fails first and client transactions don't?
- Polling connectivity of DB nodes but not client p.o.v.
- Failover can be expensive (SAN, DRBD) -> false positives costly
- <https://github.com/blog/1261-github-availability-this-week>

Load Balancers for Multi-Master clusters



Synchronous Multi-Master Clusters:

Galera
NDB

Load balancers:

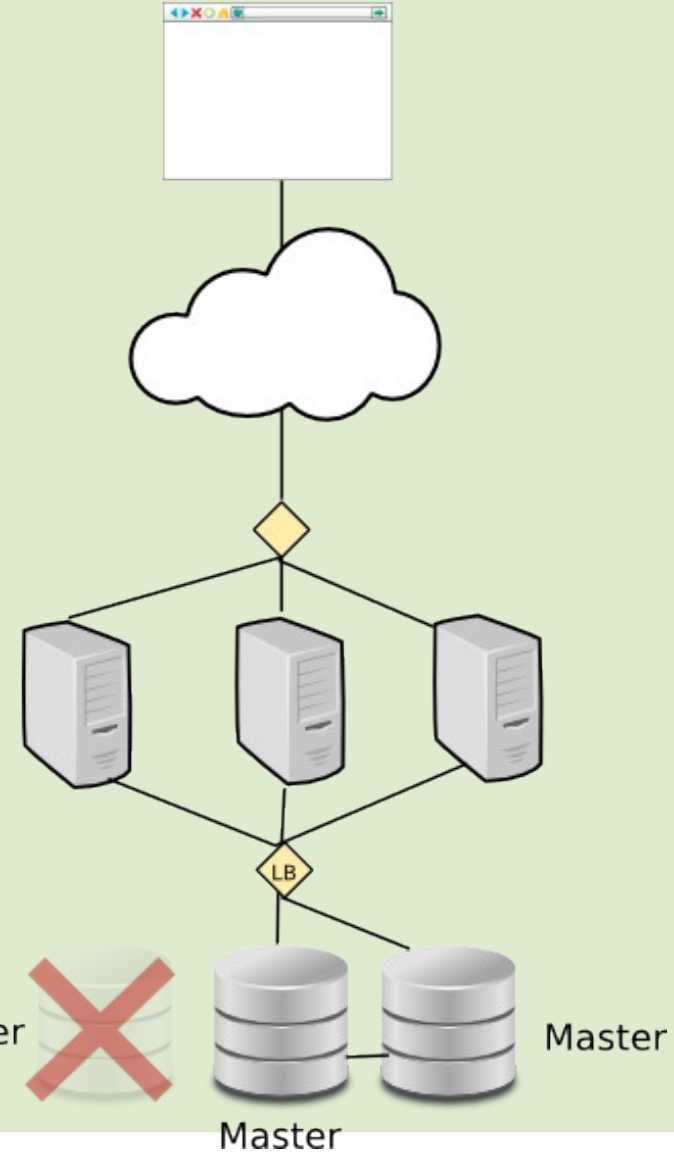
HAProxy, GLB
JDBC/PHP Driver
Hardware (e.g. F5, Cisco)

Clustering Suites:

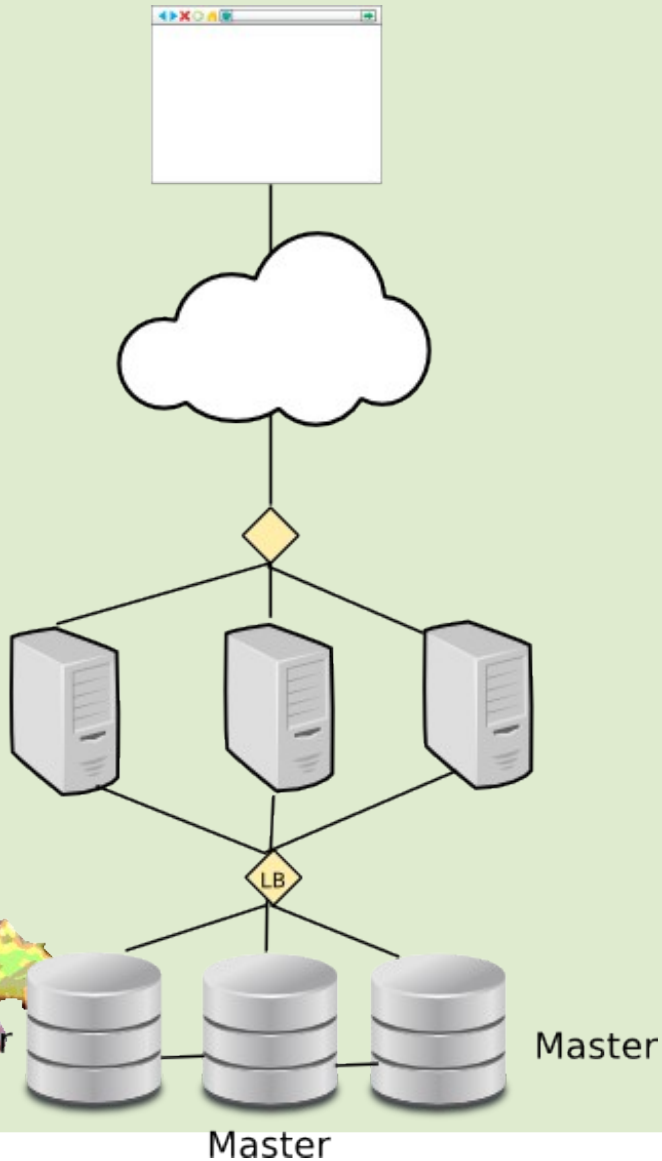
You could use VIP based failover too, but why?

Node failure

No "failover"

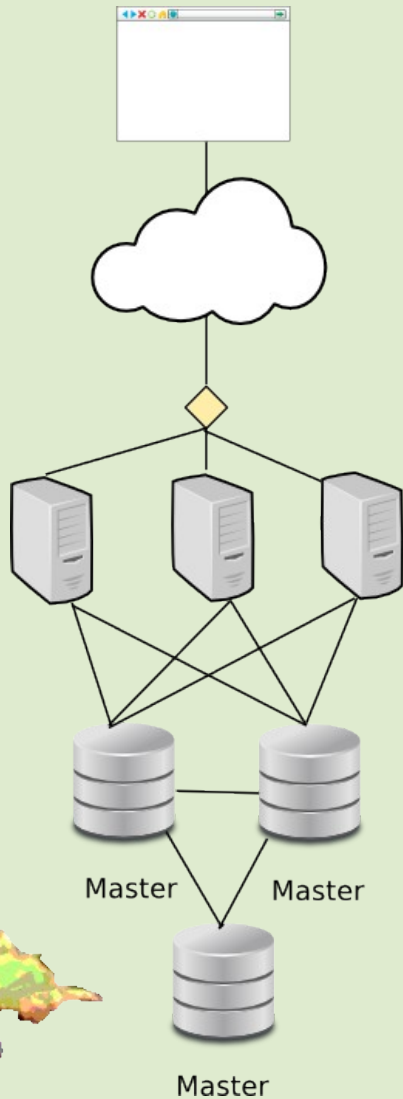


No failover needed

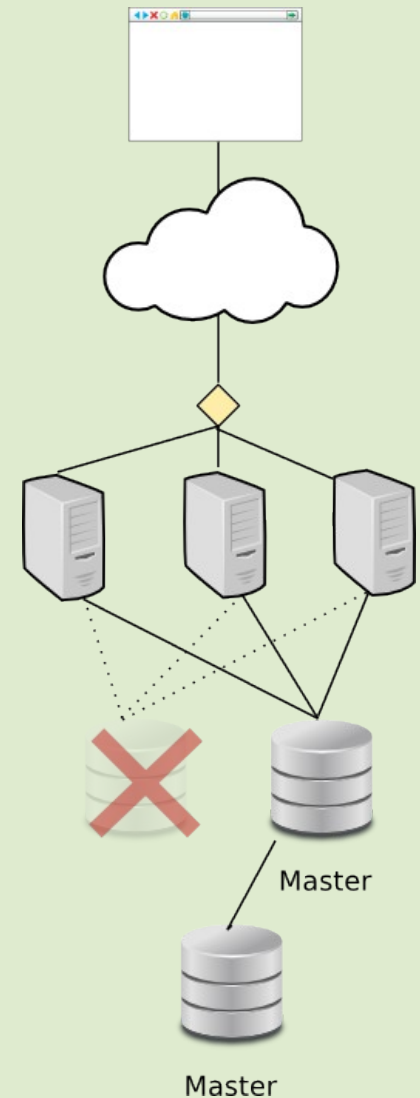


- What do you mean no failover???
- Use a load balancer
- Application sees just one IP
- Write to any available node, round-robin
- If node fails, just write to another one
- What if load balancer fails?
-> Turtles all the way down

Load Balancer in JDBC/PHP client



- No Single Point of Failure
- One less layer of network components
- Is aware of MySQL transaction states and errors
- Variant: Load balancer (like HA proxy) installed on each app node
 - > For other languages than Java & PHP



Key takeaway: Is a clustering solution part of the solution or part of the problem?

- "Causes of Downtime in Production MySQL Servers" by Baron Schwartz:
 - #1: Human error
 - #2: SAN
- Complex clustering framework + SAN =
 - More problems, not less!
- Galera and NDB =
 - Replication based, no SAN or DRBD
 - No "failover moment", no false positives
 - No clustering framework needed (JDBC loadbalance)
 - Simple and elegant!



Choosing a solution that best suits you



So we pick a HA solution and are done!

	MySQL 5.0	MySQL 5.1	MySQL 5.5	MySQL 5.6	Tungsten	Galera	DRBD	SAN	NDB
InnoDB									
Usability									
Performance									
Asynchronous									
Statement based									
Row based									
Semi-sync									
Synchronous									
Global trx id									
Multi threaded									
HA Options									



InnoDB based?

	MySQL 5.0	MySQL 5.1	MySQL 5.5	MySQL 5.6	Tungsten	Galera	DRBD	SAN	NDB
InnoDB	+	+	+	+	+	+	+	+	

InnoDB

We use InnoDB. We want to continue using InnoDB.
Which solutions support InnoDB?

NDB is it's own storage engine.

It's great. It can blow away all others in a benchmark.

But it's not InnoDB and is not considered here.



Replication type?

	MySQL 5.0	MySQL 5.1	MySQL 5.5	MySQL 5.6	Tung sten	Galera	DRBD	SAN	NDB
InnoDB	+	+	+	+	+	+	+	+	
Usability	+	+	+	+	++	++		-	+
Performance				(1)	(1)	+	-	-	+

<----- MySQL server level replication -----> <- disk level-> <engine>

Higher level replication is better

Competence:

Replication = MySQL DBA can manage
 DRBD = Linux sysadmin can manage
 SAN = Nobody can manage

Performance:

SAN has higher latency than local disk
 DRBD has higher latency than local disk
 Replication has surprisingly little overhead

Operations:

Disk level = cold standby = long failover time
 Replication = hot standby = short failover time
 ++ for global trx id, easy provisioning

Redundancy:

Shared disk = Single Point of Failure
 Shared nothing = redundant = good



Statement vs Row based?

Asynchronous vs Synchronous?

	MySQL 5.0	MySQL 5.1	MySQL 5.5	MySQL 5.6	Tung sten	Galera	DRBD	SAN	NDB
InnoDB	+	+	+	+	+	+	+	+	
Usability	+	+	+	+	++	++		-	+
Performance				(1)	(1)	+	-	-	+
Asynchronous	+	+	+	+	+	(2)			
Statement based	+	+	+	+	+				
Row based		+	+	+	+	+	(3)	(3)	+
Semi-sync			+	+					
Synchronous						+	+	+	+
Global trx id				+	+	+			+
Multi threaded				(1)	(1)	+			+



Row based = deterministic = good
Statement based = dangerous

Asynchronous = data loss on failover
Synchronous = good

Global trx id = easier setup & failover for complex topologies

Multi-threaded = scalability

Clustering framework vs load balancing?

	MySQL 5.0	MySQL 5.1	MySQL 5.5	MySQL 5.6	Tungsten	Galera	DRBD	SAN	NDB
InnoDB	+	+	+	+	+	+	+	+	
Usability	+	+	+	+	+	++		-	+
Performance				(1)	(1)	+	-	-	+
Asynchronous	+	+	+	+	+	(2)			
Statement based	+	+	+	+	+				
Row based		+	+	+	+	+	(3)	(3)	+
Semi-sync			+	+					
Synchronous						+	+	+	+
Global trx id				+	+	+			+
Multi threaded				(1)	(1)	+			+
Cluster suite / LB						+			+

1) Multi-threaded slave, 1 per schema

2) No, but can be combined with MySQL replication

3) Reliability comparable or better than row based replication



Conclusions

- Simpler is better
- Higher level replication is better: MySQL level replication is better than DRBD which is better than SAN
- Synchronous replication = no data loss
- Asynchronous replication = no latency (WAN replication)
- Synchronous Multi-Master = no failover = no failover / clustering frameworks
- Multi-threaded slave increases performance in disk bound workload
- Global trx id, autoprovisioning increases operations usability
- Galera and NDB provide all these with good performance and stability



References

- <http://openlife.cc/blogs/2011/july/ultimate-mysql-high-availability-solution>
- <http://openlife.cc/category/topic/galera>
- <http://openlife.cc/blogs/2011/may/drbd-and-semi-sync-shootout-large-server>
- <http://www.percona.com/about-us/white-papers/>
- <http://www.mysqlperformanceblog.com/2011/09/18/disaster-mysql-5-5-flushing/>
- <https://github.com/blog/1261-github-availability-this-week>

