Choosing a MySQL Replication & High Availability Solution

Henrik Ingo
Senior Performance Architect
Nokia

Percona Live UK
2011-10-25

open source technology and strategy specialist

active in MySQL-forks, Drupal communities

author of "Open Life: The Philosophy of Open Source"

www.openlife.cc
henrik.ingo@openlife.cc

Henrik Ingo

Senior Performance Architect, Nokia

- SOA: Each team does their own thing
- Nokia and web?
  App store, music store, Maps, SSO...
  200M clients, 100M accounts
- Architecture reviews, "internal consultant"
- MySQL improvements:
  Recommend backup, HA, version etc...
  best practices

NOKIA
Connecting People

High Availability

Performance

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

Durability

Speaking of databases
Committed data is not lost
D in ACID

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

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.

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									
Clustering framework									

Ok, so what do I really want?

	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.

MySQL level vs disk level replication

	MySQL 5.0	MySQL 5.1	MySQL 5.5	MySQL 5.6	Tungsten	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	Tungsten	Galera	DRBD	SAN	NDB
InnoDB	+	+	+	+	+	+	+	+	
Usability	+	+	+	+	++	++		-	+
Performance				(1)	(1)	+	-	-	+
Asynchronous	+	+	+	+	+	(2)			
Statement based	+	+	+	+	+				+
Row based		+	+	+	+	+	(3)	(3)	
Semi-sync			+	+					
Synchronous						(4)	+	+	+
Global trx id				+	+	+			+
Multi threaded				(1)	(1)	+			+

Row based = deterministic = good
Statement based = dangerous

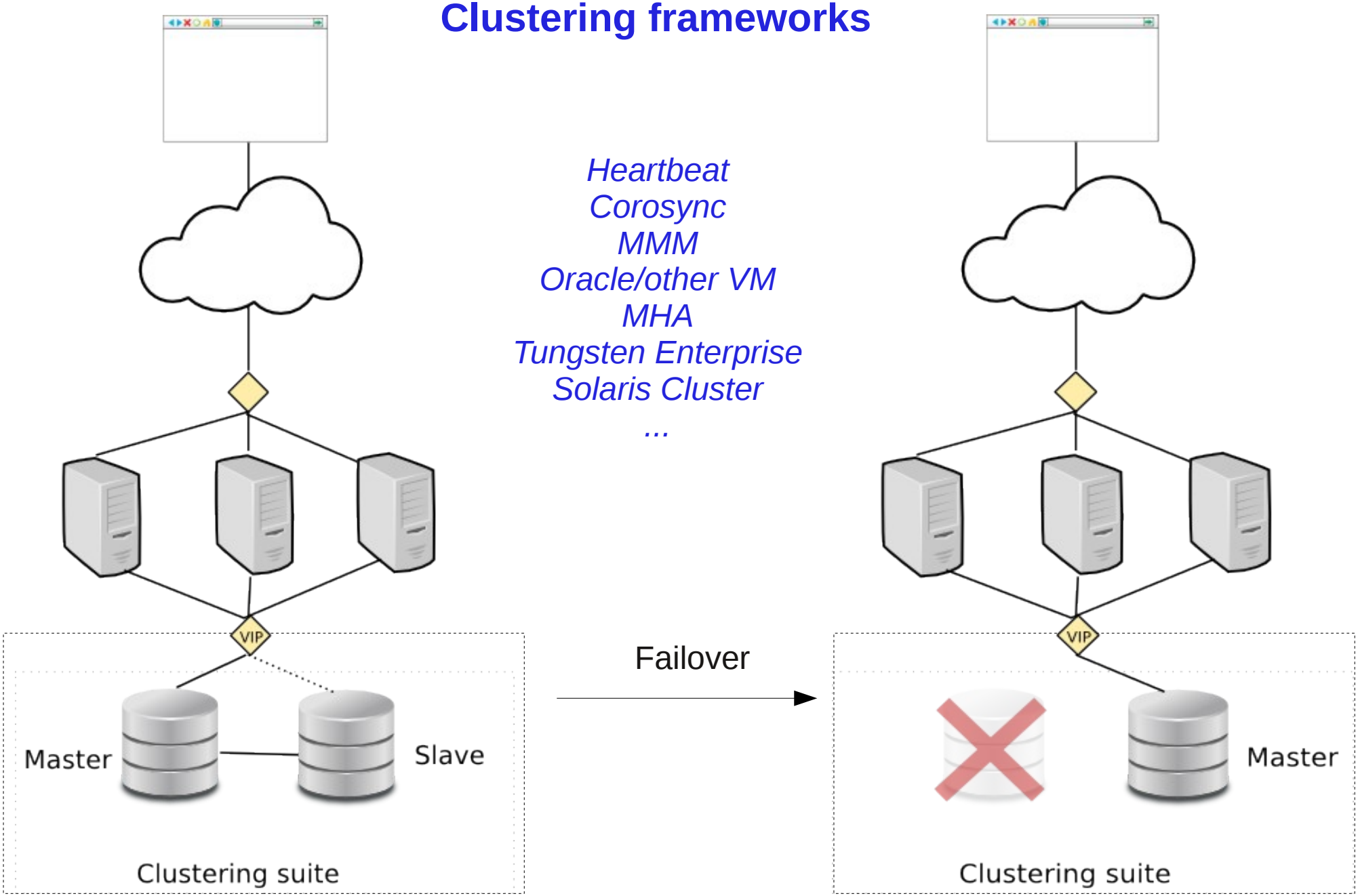
Asynchronous = data loss on failover
Synchronous = good

Global trx id = easier setup & failover for complex topologies

Multi threaded = scalability

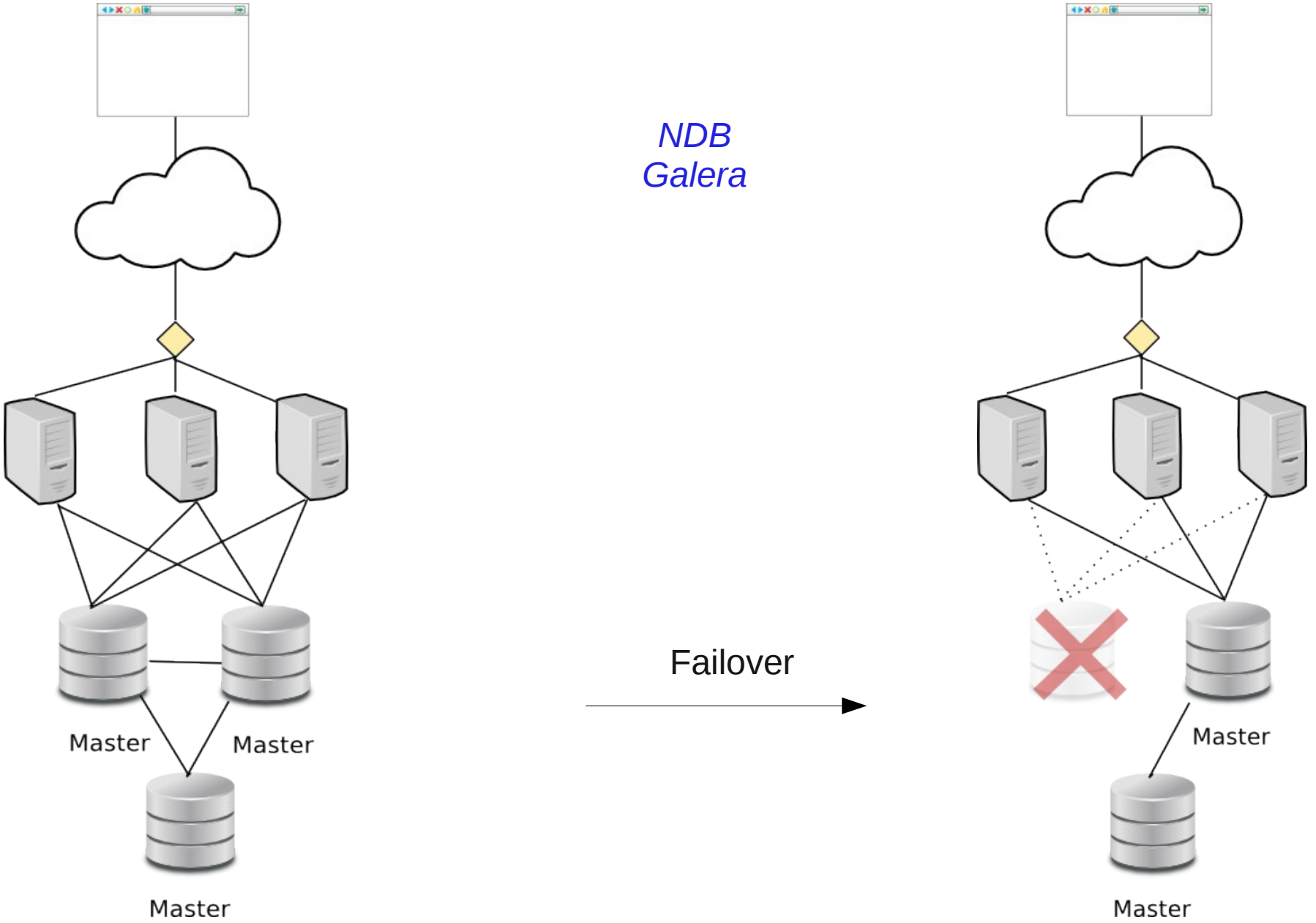
Clustering frameworks

Heartbeat
Corosync
MMM
Oracle/other VM
MHA
Tungsten Enterprise
Solaris Cluster
...



Synchronous multi-master

NDB
Galera



Is a clustering solution part of the solution or the part of the problem?

- "Causes of Downtime in Production MySQL Servers"
  by Baron Schwartz:

    #1: Human error
    #2: SAN

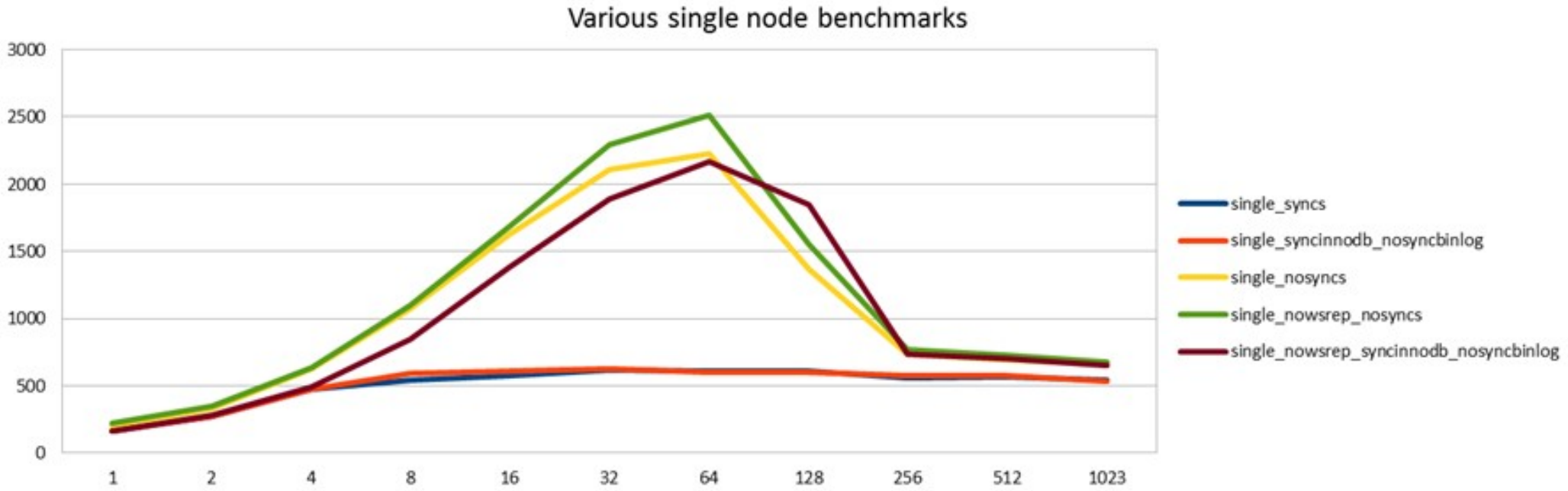
  - Complex clustering framework + SAN =
    - More problems, not less!
  - Galera (and NDB) =
    - Replication based, no SAN or DRBD
    - No "failover moment", no false positives
    - No clustering framework needed (JDBC loadbalance)
    - Simple and elegant!

Statement vs Row based
Asynchronous vs Synchronous

	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						+(4)	+	+	+
Global trx id				+	+	+			+
Multi threaded				(1)	(1)	+			+
Clustering framework						+			+

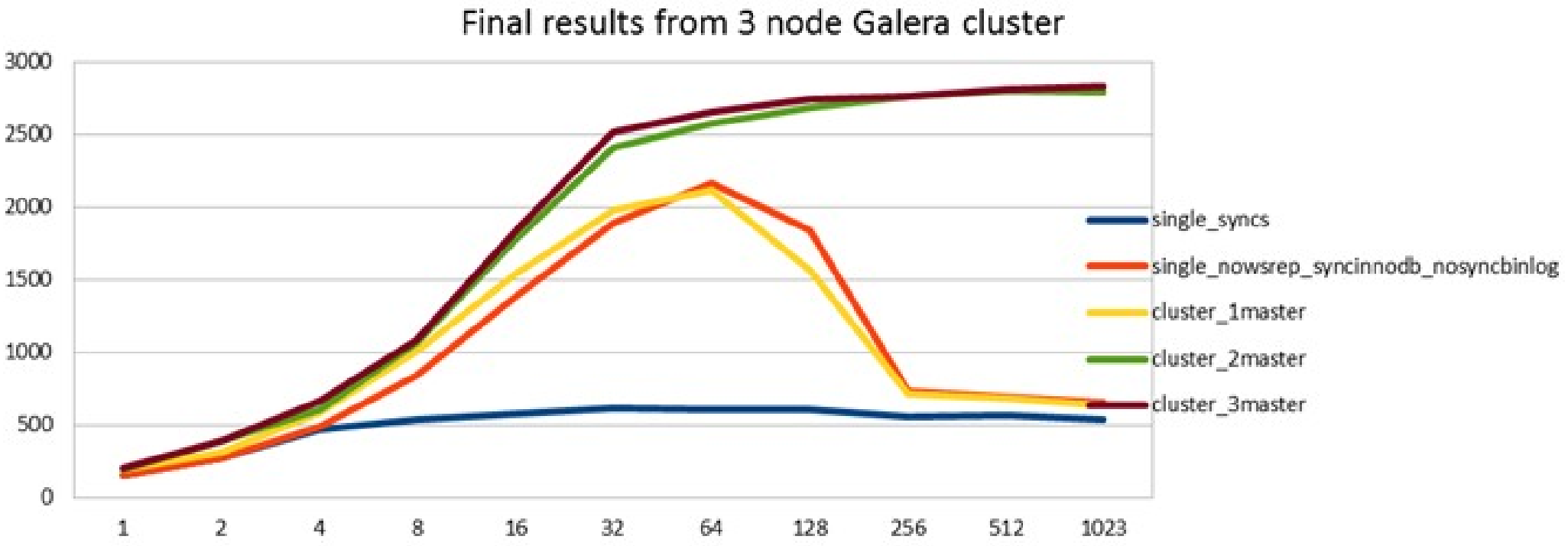
1) Multi-threaded slave, 1 per schema
2) No, but can be combined with MySQL replication
3) Reliability comparable to row based replication
4) Internally slave applier is asynchronous, but exposes synchronous caracteristics

Benchmarks!



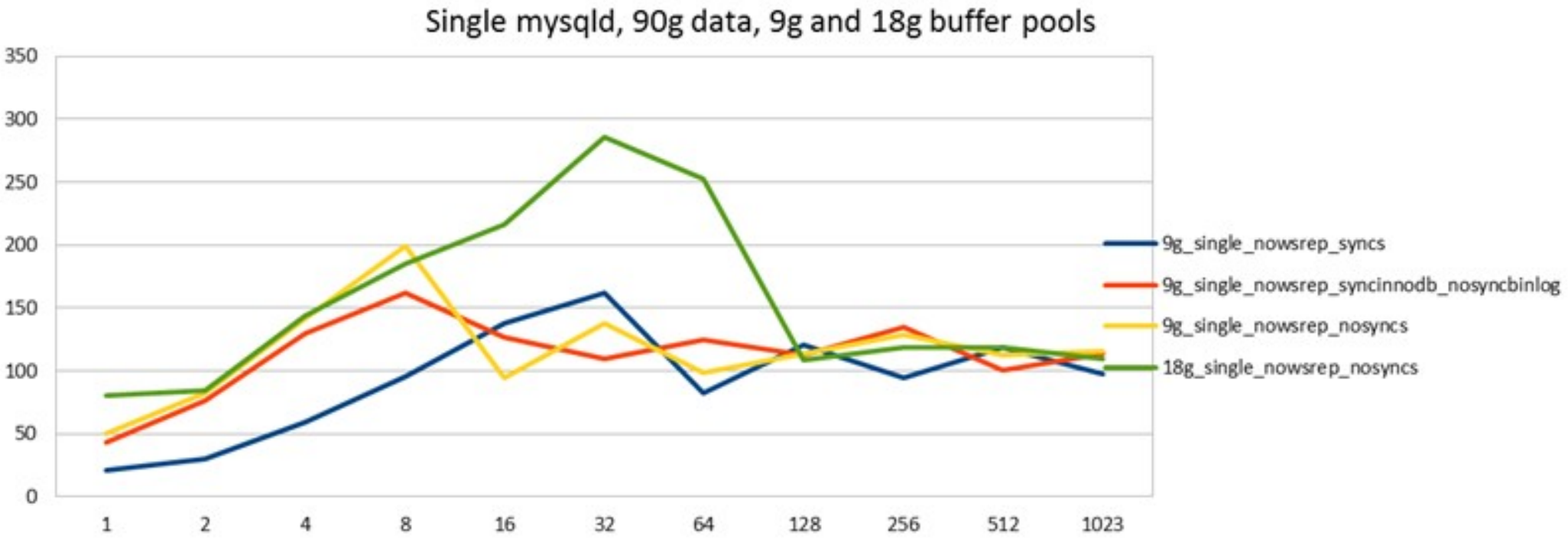
Baseline single node performance
"group commit bug" when sync_binlog=1 & innodb_flush_log_at_trx_commit=1
wsrep api (Galera module, no replication) adds minimal overhead

3 node Galera cluster



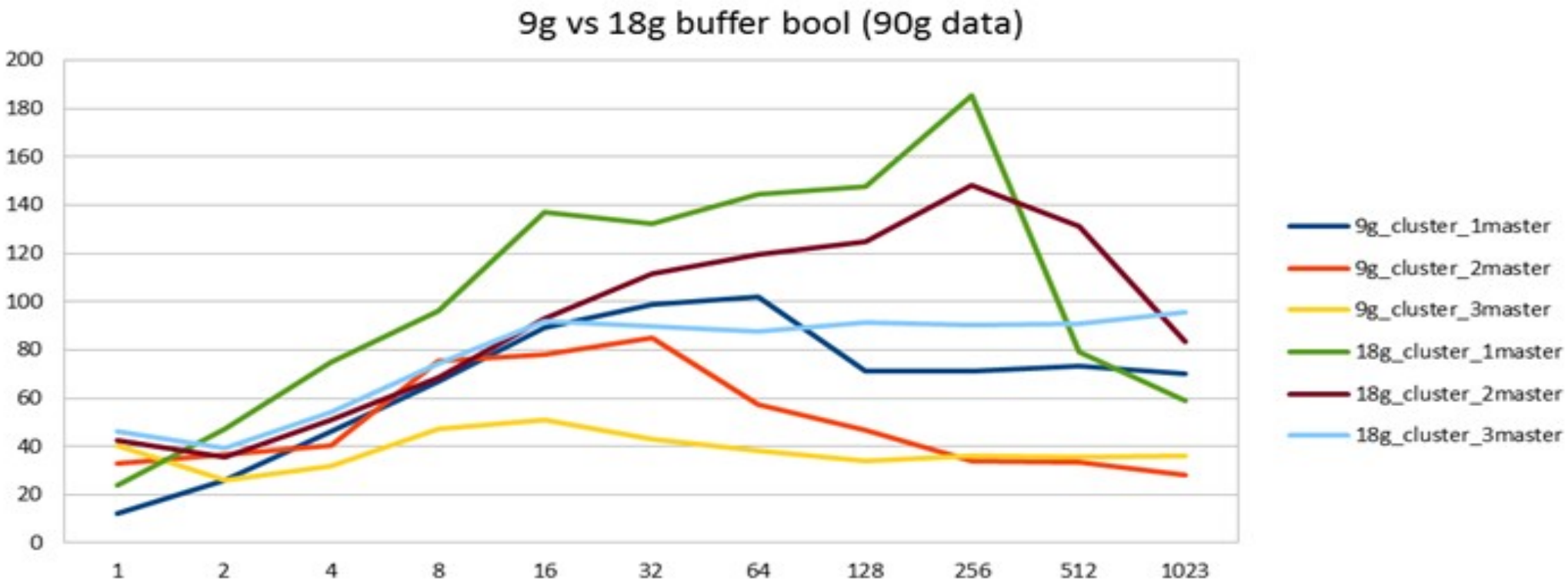
No overhead in master-slave mode (red vs yellow)
Small benefit in multi-master mode (depends on read/write ratio)

Single node, disk bound workload



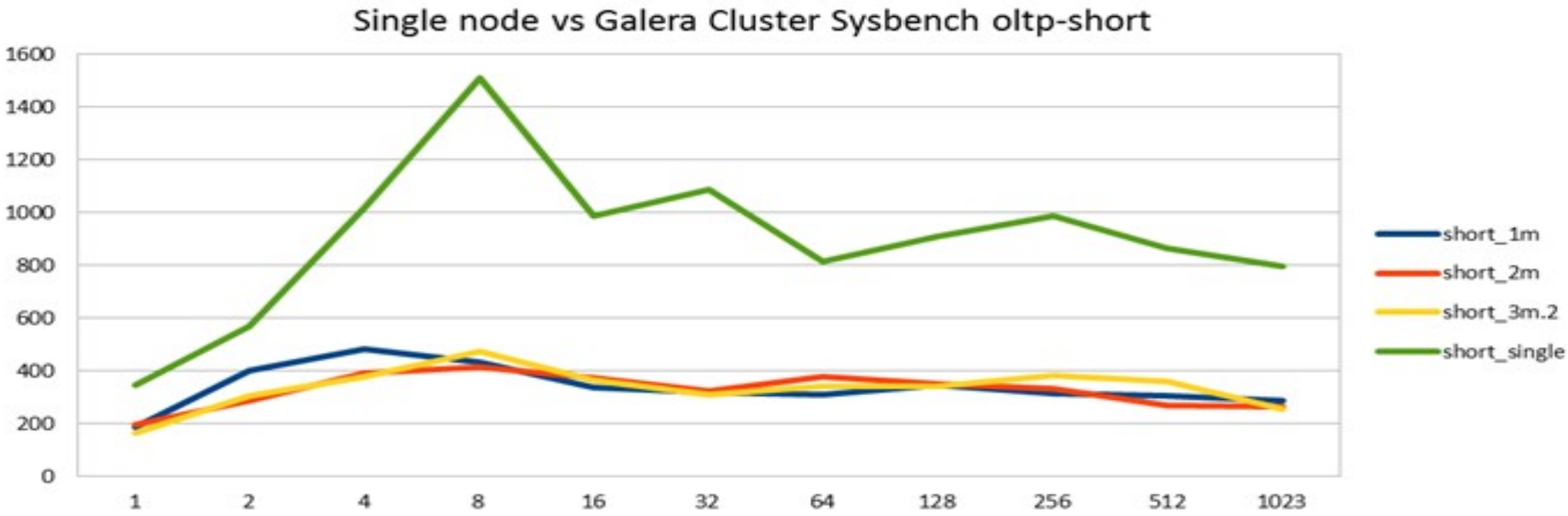
10% and 20% of data in cache
Optimization for dummies: Double amount of RAM, double performance

3 node Galera cluster, disk bound



Again 2x RAM => 2x performance
Roughly 50% of single node performance
Decreases when writing to multiple masters (weird!)
=> Blame InnoDB redo log purge weirdness

Same disk bound cluster, modified Sysbench OLTP

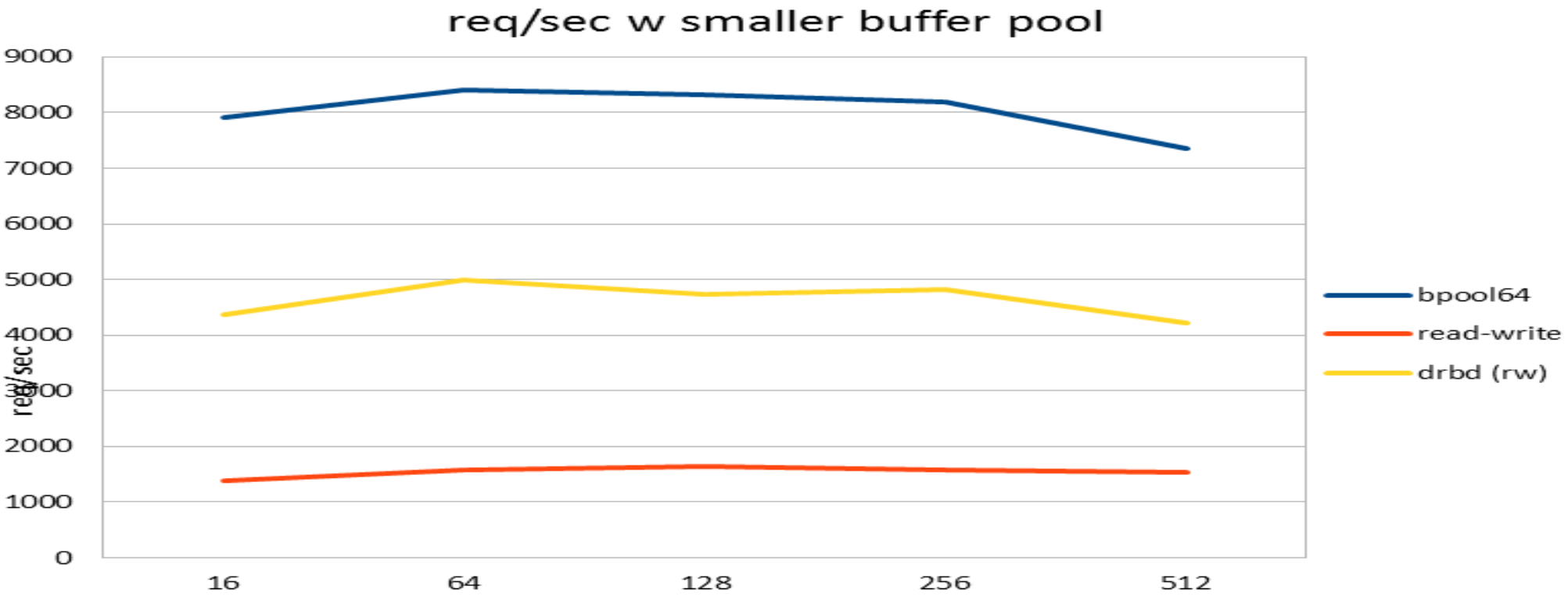


Modified sysbench oltp: Same queries as isolated short transactions.
- 75% read-only, 25% write-only

Performance is same in master-slave and multi-master modes
- Limited by write throughput

40% of single node performance

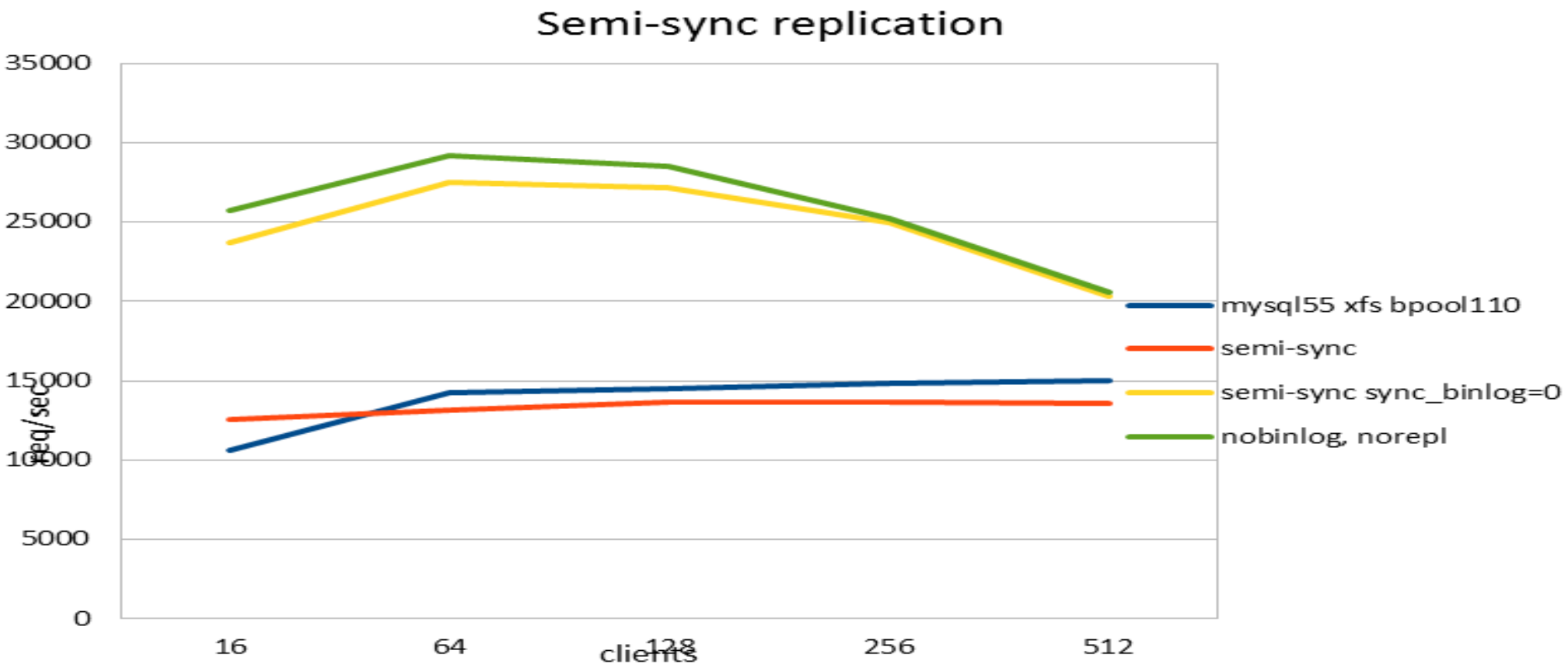
DRBD vs Single node



60% of single node performance
Minimum latency 10x higher but average is not so bad (not shown)

Note: This is different HW than the Galera test, and different metric

Semi sync vs Single node



Practically no performance overhead
Opportunity to relax sync_binlog setting (green - yellow)

Conclusions

- Simpler 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)
- Multi-master = no clustering frameworks
- Multi-threaded slave increases performance in disk bound workload
- Global trx id, autoprovisioning increases operations usability
- Galera (and NDB) provides 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/