How to choose High Availability solutions for MySQL

MySQL UC 2010
Yves Trudeau
Read by Peter Zaitsev

Percona Inc
MySQLPerformanceBlog.com

About us

- http://www.percona.com
- http://www.mysqlperformanceblog.com/
- http://www.bigdbahead.com

Yves Trudeau, Ph. D.

Principal Consultant

Plan

1) Definitions of some High-Availability (HA) terms
2) Questions to ask
3) HA mindset
4) Common HA solutions with MySQL
    - Replication based
    - Shared storage based
    - NDB Cluster
5) Other solutions

Definitions

1) High-Availability (HA)
   - A computer architecture design and implementation that is targeted at improving the availability of a given service
2) Uptime and downtime
   - The proportion of time a high availability service is up or down over the total time. Normally, uptime + downtime = 100%.
3) Level of availability
   - Typically in term of the fraction of uptime and referred by the number of 9, 99% (2 9s), 99.9% (3 9s), etc.

Definitions

4) Single point of failure (SPOF)
   - An isolated device or piece of software for which a failure will cause a downtime of the HA service. The goal of an HA architecture is to remove the SPOFs.

5) Recovering or failover
   - The process by which a HA architecture recovers after a failure.

6) Fencing/Stonith
   - Often, an HA architecture is stuck by a non-responsive device that is not releasing a critical resource. Fencing or Stonith (*Shoot The Other Node In The Head*) is then required.

Definitions

7) Cluster
   - A group of computers acting together to offer a service.

8) Fault Tolerance
   - Ability to handle failures with graceful degradation. Not all components may need same level of HA

9) Disaster Recovery
   - The plan and technologies to recover in case of disaster. Often longer downtime allowed in this case.

Questions

1) Do you need HA?
   - can be rephrased to “What is your downtime cost?”
   - Include non-monetary aspects like corporate image and marketing
   - For the downtime cost, what is acceptable over a year?
   - Do you have maintenance windows that offers reduce downtime cost?
2) Can you afford to lose some data?
   - What is the cost of losing a transaction?
   - How critical is data consistency ?

Questions

3) Are relying on MyISAM only features?
   - Fulltext indexes?
   - GIS?
   - Sphinx or Lucene options?
4) What is the write load?
   - How many threads are writing simultaneously?
   - How many write ops/s?
5) What is the growth potential of your dataset?

Questions

6) How qualified is your IT department or support company?

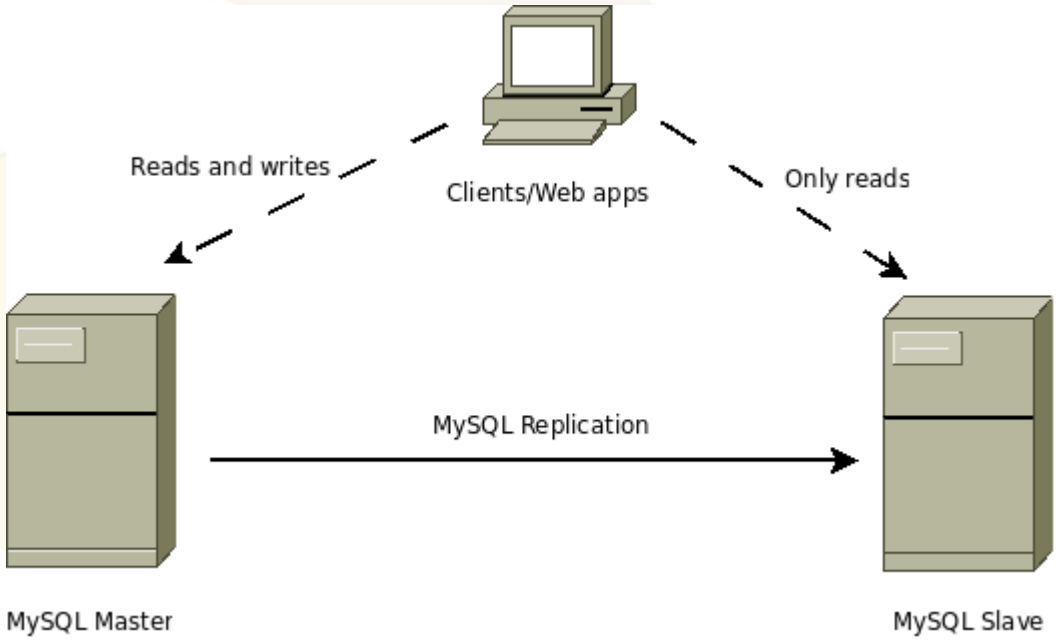
7) How much are you ready to invest?

HA Mindset

1) HA, not only about technologies
   - No technology is fool proof
   - Operating procedures are required
   - Testing and staging
   - Monitoring and alerting
2) A HA is not isolated, look at the broad picture
   - No need for HA of 99.999% if ISP SLA is 99.9%
   - Power
   - Cooling, more frequent problem than you might think
   - Very high HA requirements need multiple data centers.

Replication based

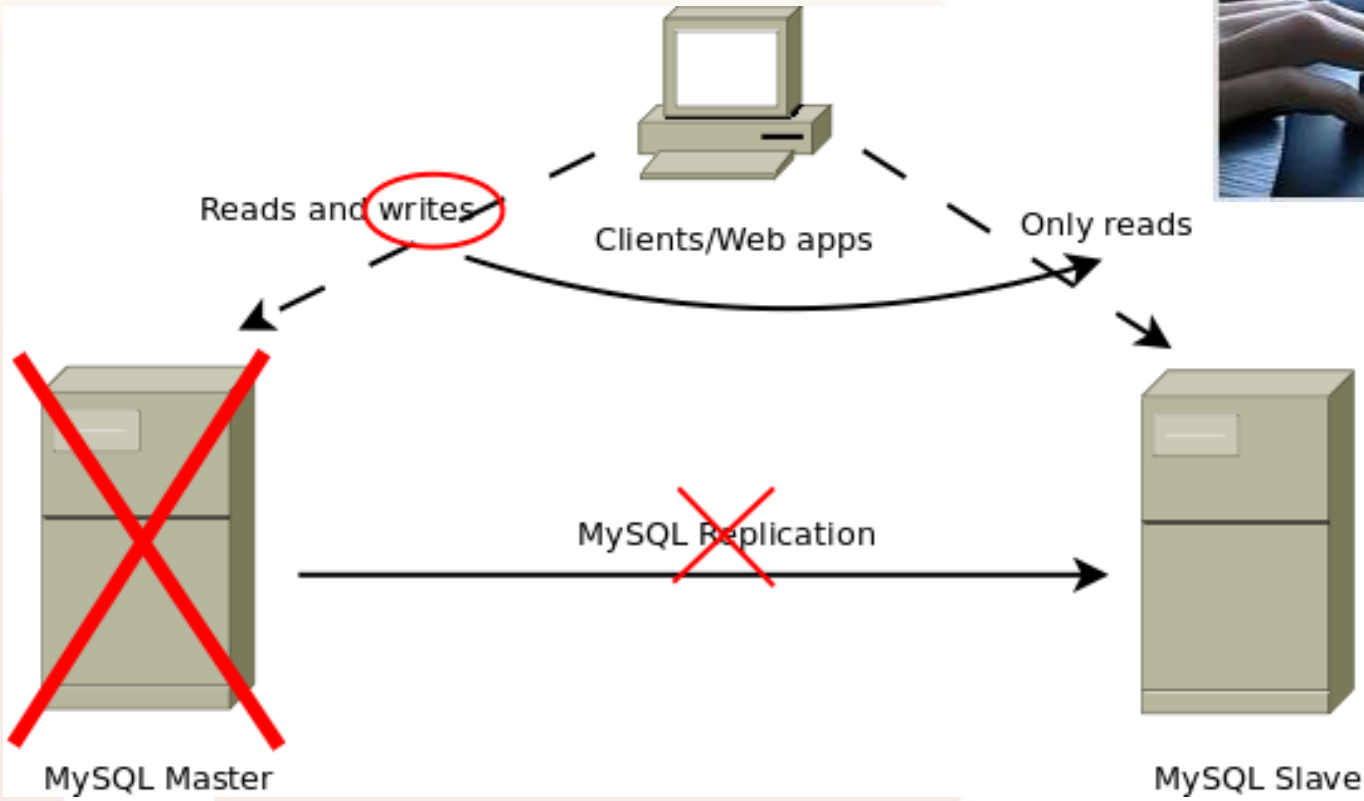
1) Simplest example, plain replication
   - Widely used
   - Manual failover



Replication based failover

2) Simple replication, failover process

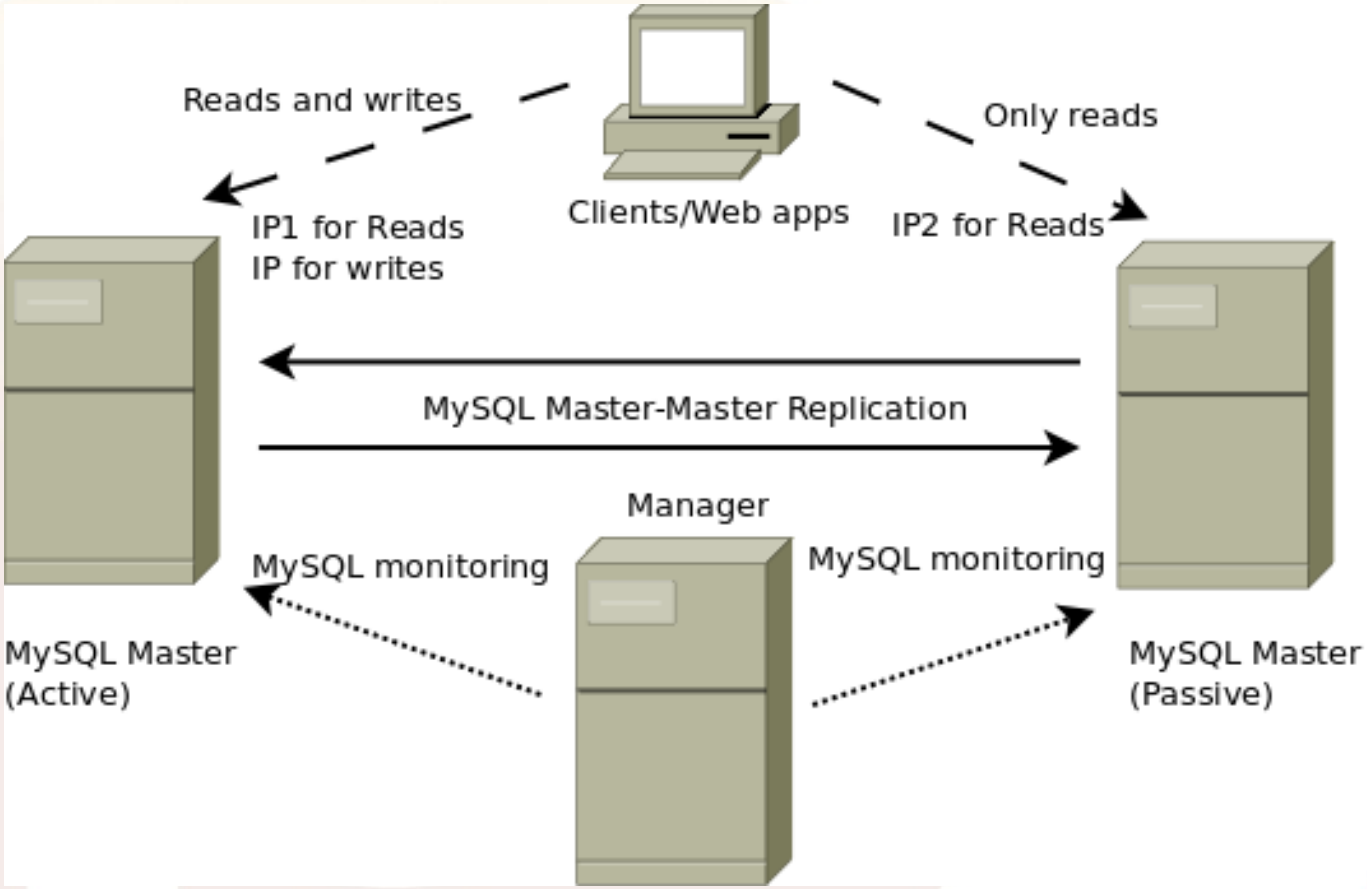
- Manual operation required



PERCONA
Performance Consulting Experts

Replication based MMM

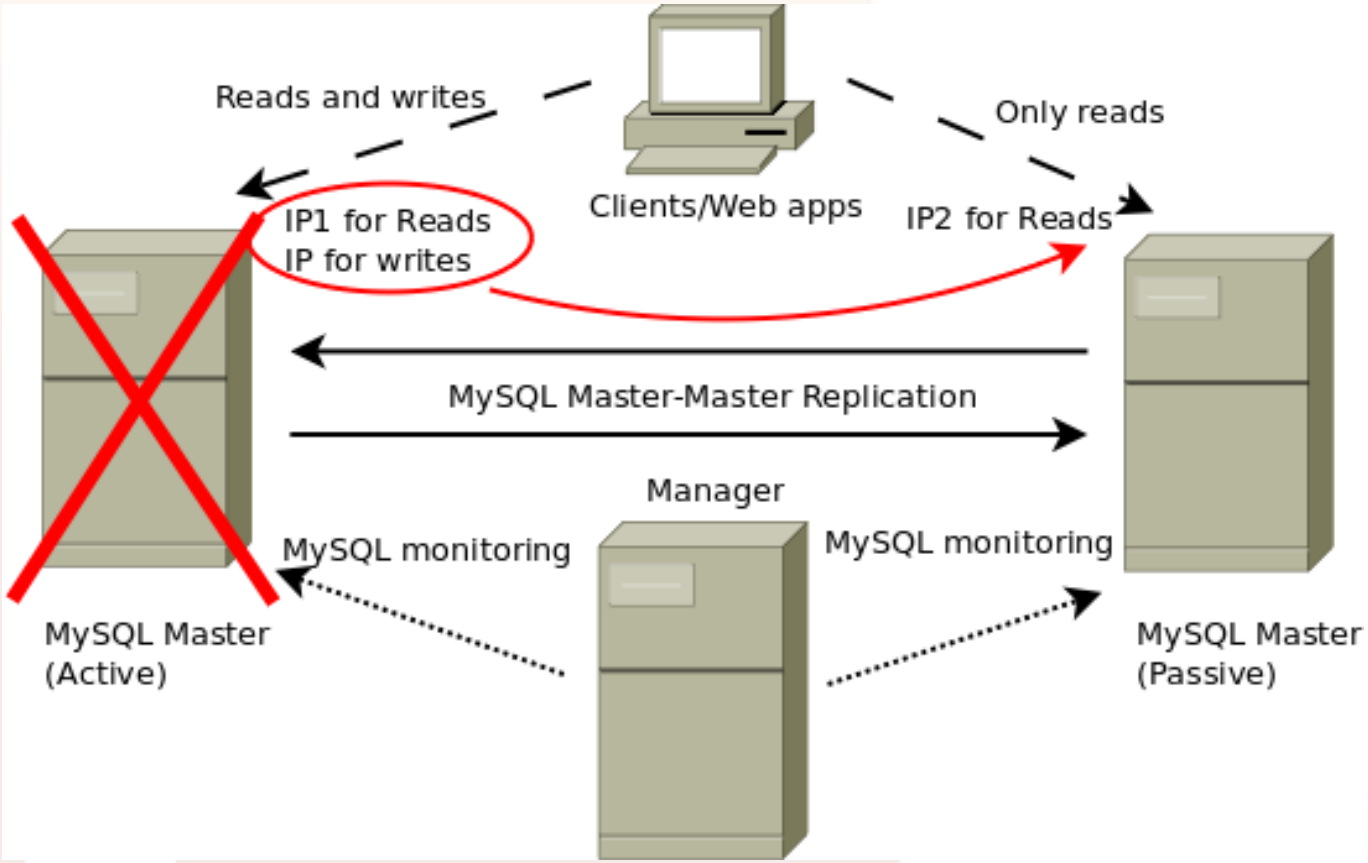
3)Example 2, using MMM



Replication based MMM failover

4) Failover with MMM

- Manager transfer IP1 and IP to the surviving server



Replication based other

4) Other solutions built on replication

- Tungsten, Java proxy layer doing man in the middle work for queries and replication stream
- Pacemaker/Heartbeat, not released yet, developed by Linbit, will add fencing capabilities

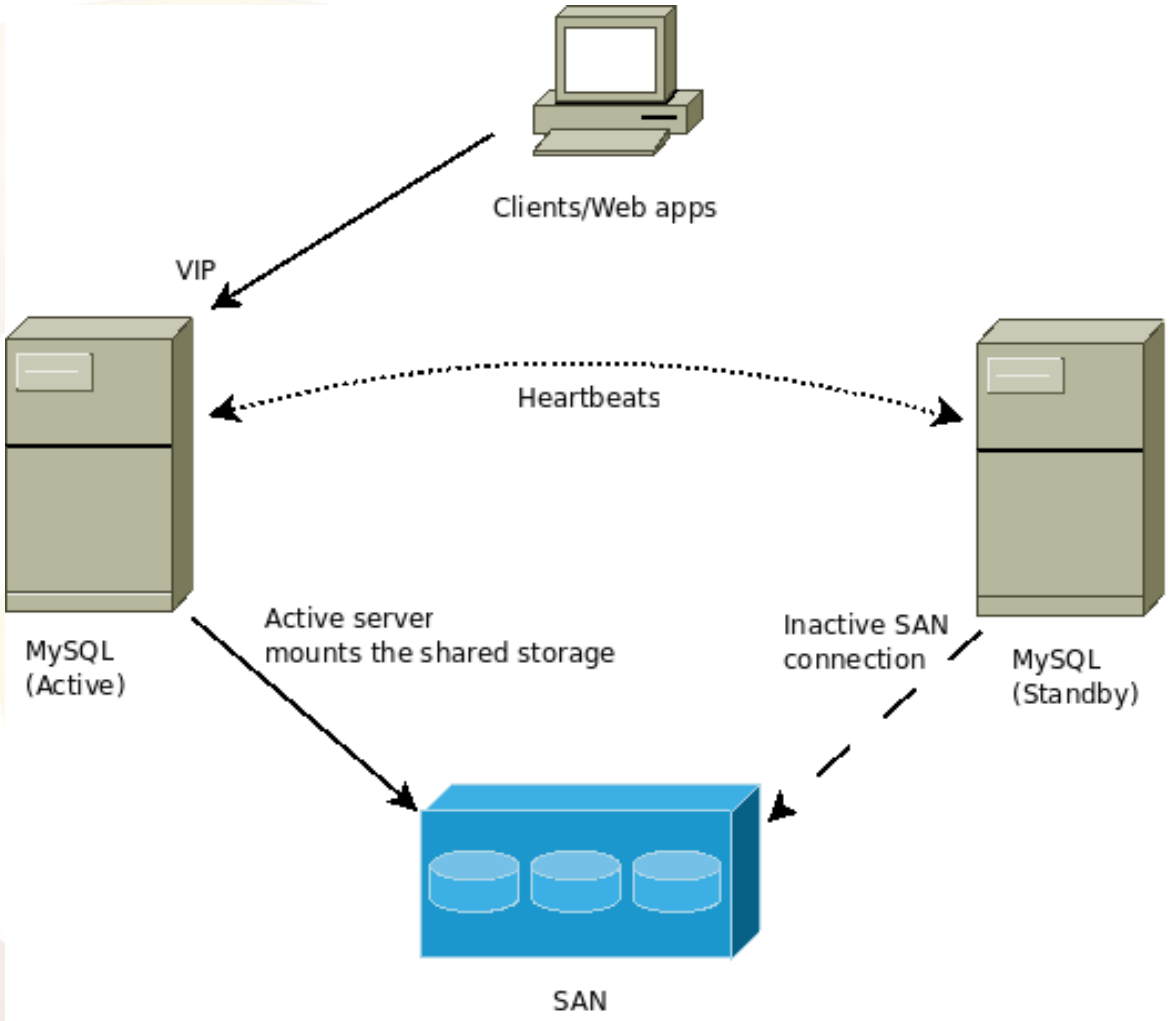
Replication based Pros

- Simple/Inexpensive
- Supports MyISAM
- All servers can be used, no standby
- Good to scale read ops
- Caches are kept warm
- Can be used for online schema changes, upgrades
- Loosely coupled

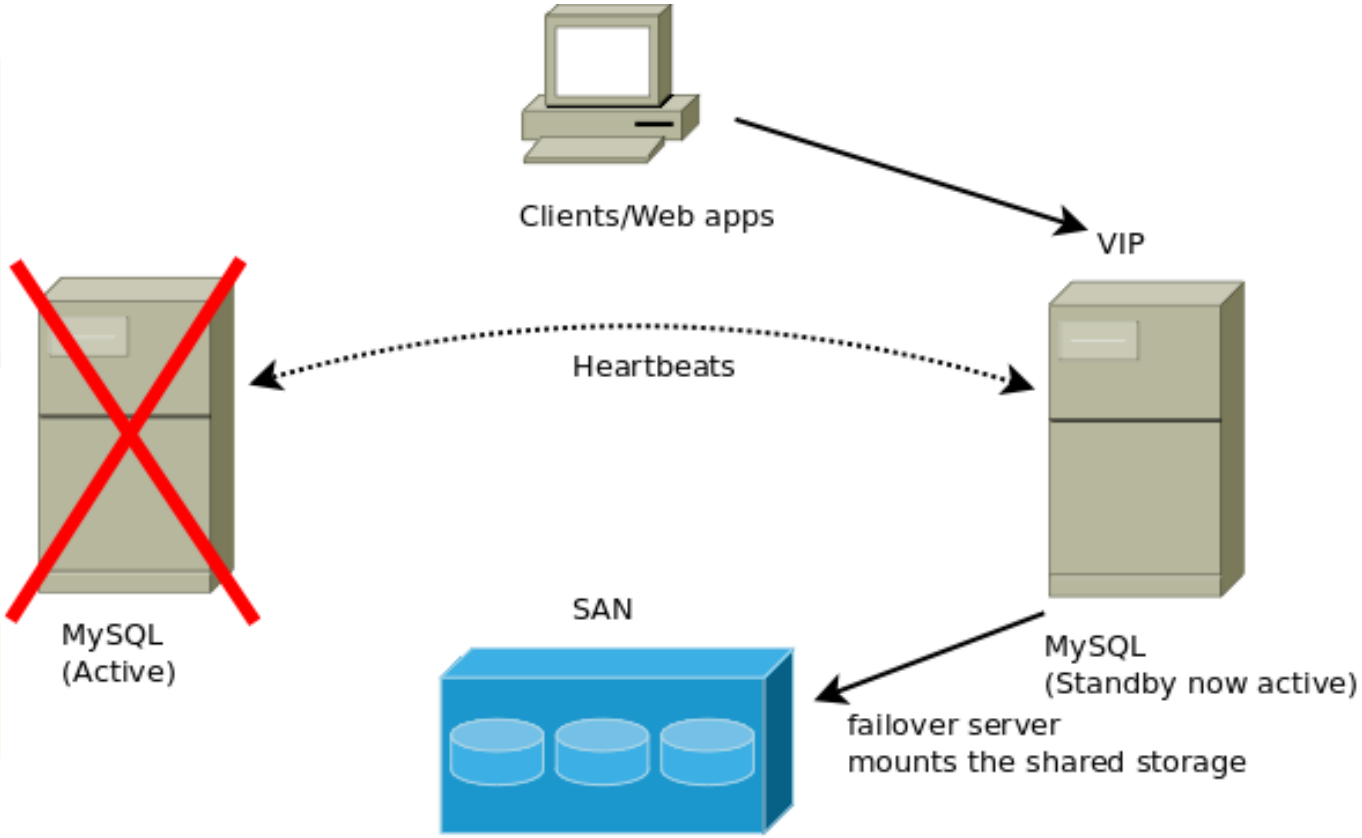
Replication based Cons

- Limited availability
  - Replication can break
  - Replication can lag behind
  - Replication can be out of sync
- Manual or at best semi-automatic failover, tricky to automate.
- Limited write capacity: single threaded
- Can lose data: async (with semi-sync repl?)
- Immature tools, edge cases not always handled

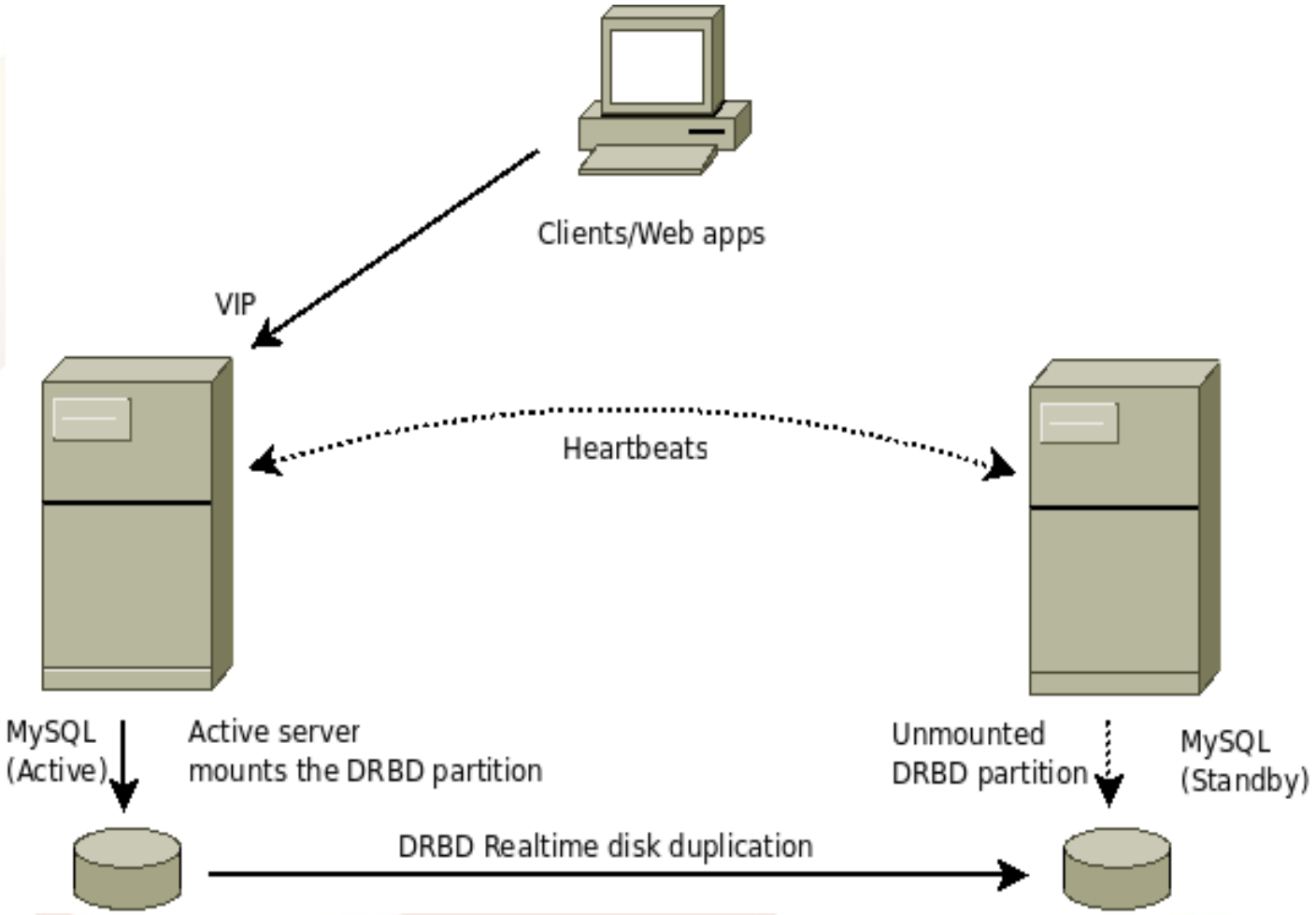
Shared storage/SAN



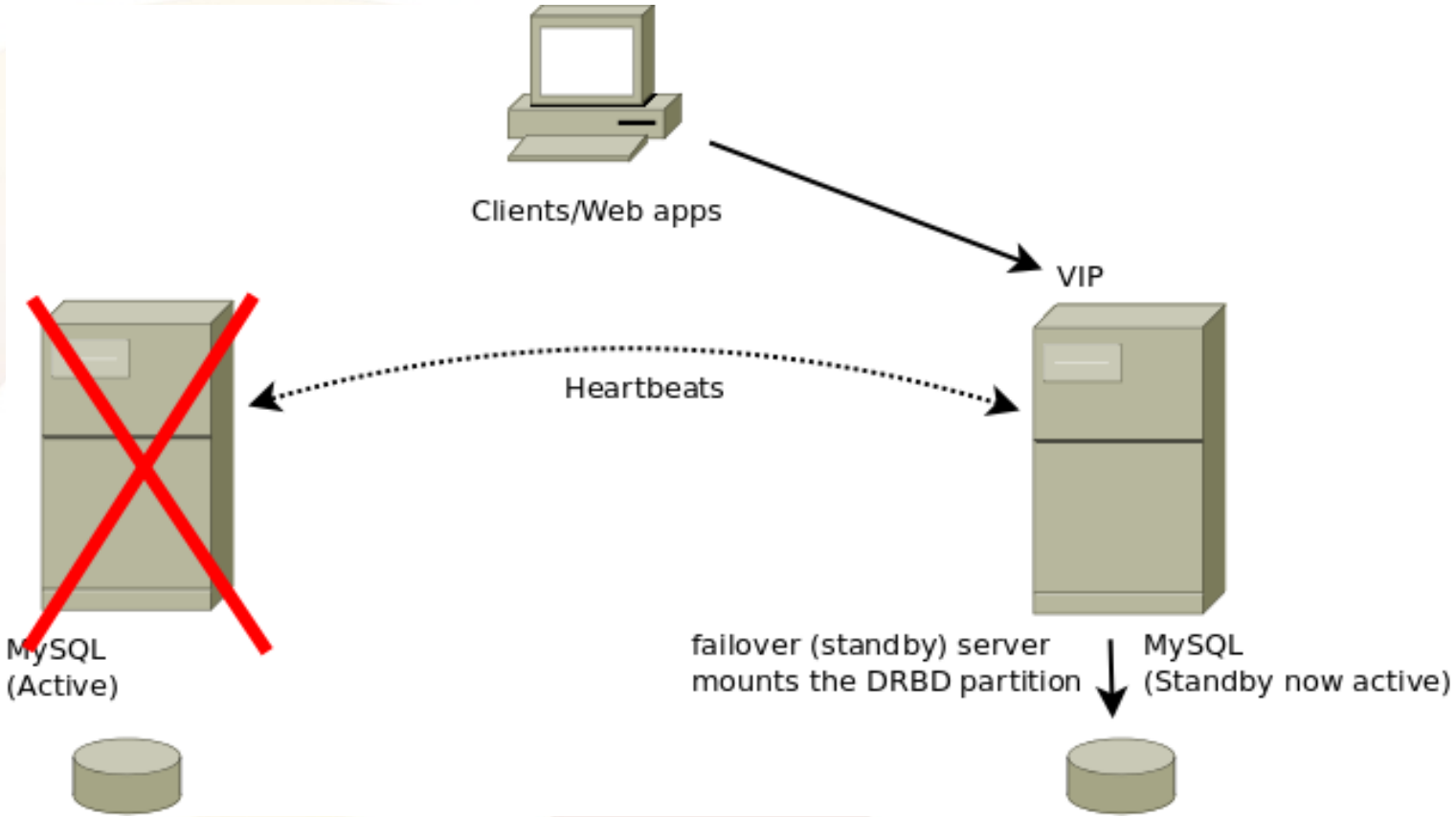
Shared storage/SAN failover



Shared storage/DRBD



Shared storage/DRBD failover



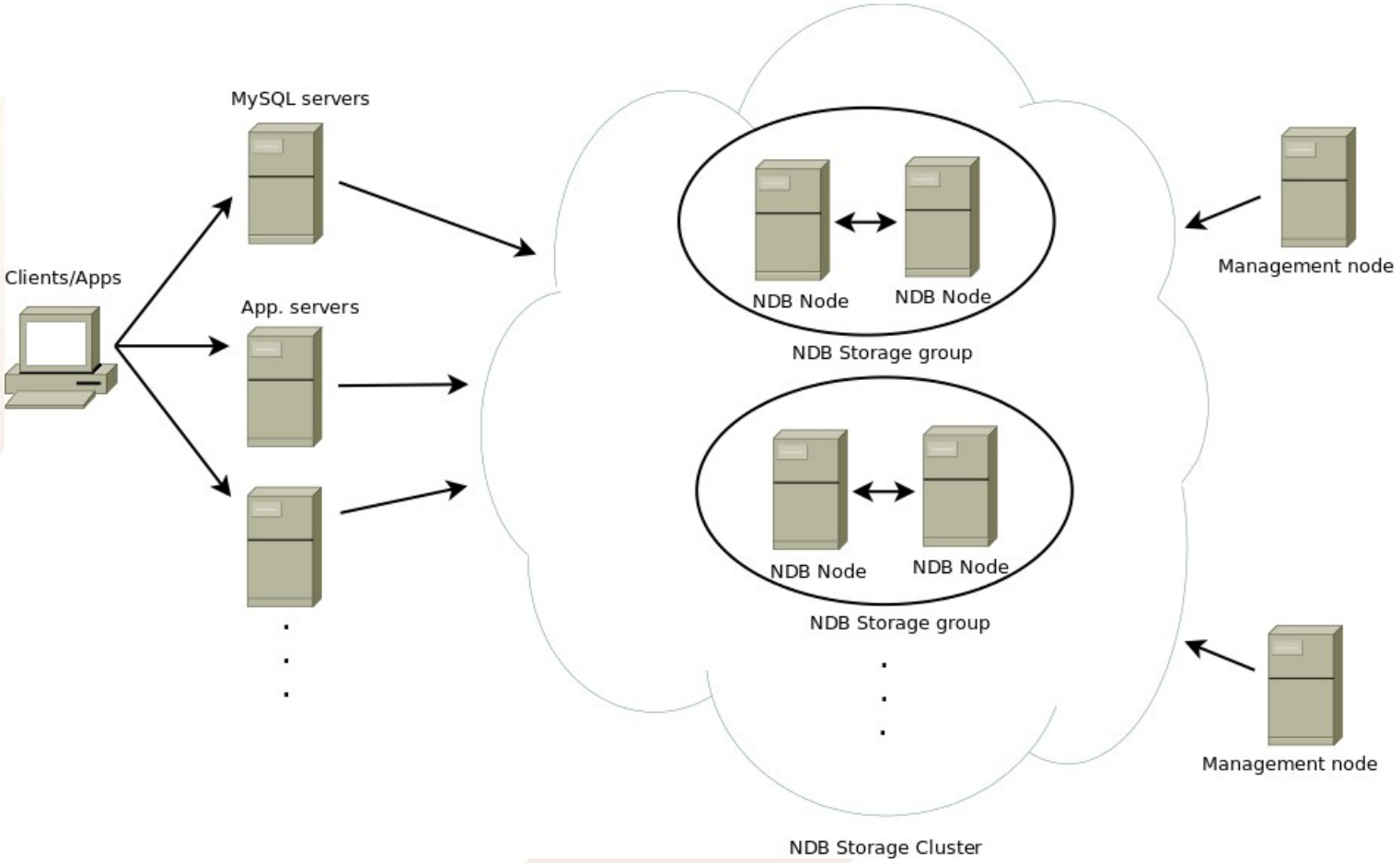
Shared storage Pros

- No data loss
- Much higher write capacity
- Automatic failover in about 1 minute with InnoDB log files of about 100 MB
  - Comes at performance cost
- No SPOF with DRBD

Shared storage Cons

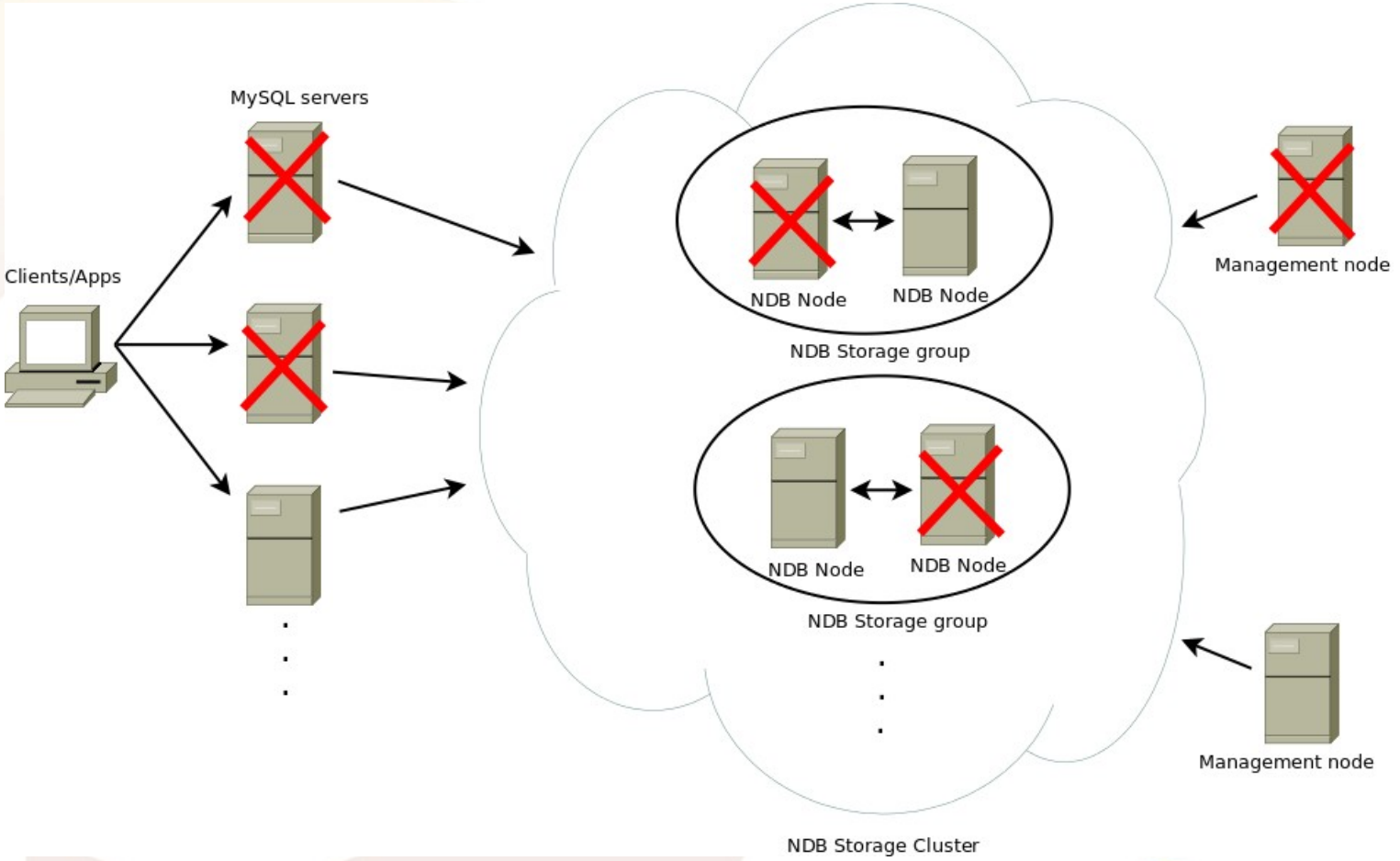
- Only works with engine supporting recovery (InnoDB), should work with PBXT and Maria (Have not tested)
- More complex: nic bounding, fencing, etc.
- Requires fencing
- A server is standby, idle hardware
- Cold cache after failover although XtraDB LRU dump can be a big winner here
- No online schema change
- Corruption Propagation
-

NDB Cluster



NDB Cluster failover

Still up!



NDB Cluster Pros

- Sharding framework by hashes
- No SPOF, high level of HA
- Scalable, if the schema is well designed, adding data nodes adds processing capacity
- Huge write capacity

NDB Cluster Cons

- Complex, much than other solutions
- Needs work on schema and queries for good performance
- Higher skill set required
- Poor for large joins
- Size of dataset more limited, large memory footprint
- Minimum of physical servers

Emerging Solutions

1) Hot market a lot of work going on !
2) ScaleDB
3) Galera Replication
4) Number of Solutions which are still in stealth

Mixed solutions

- Geo-redundancy
  - Shared storage + replication
  - NDB + replication
- Scaling reads ops
  - Shared storage HA Master with reads slaves
- Sharding solutions
  - Global database using NDB
  - shards using shared storage

Questions

Questions?

Yves Trudeau: yves@percona.com

http://www.percona.com/

http://www.mysqlperformanceblog.com/