MySQL HA Solutions

Keeping it simple, kinda!

By: Chris Schneider
MySQL Architect
Ning.com

What we’ll cover today

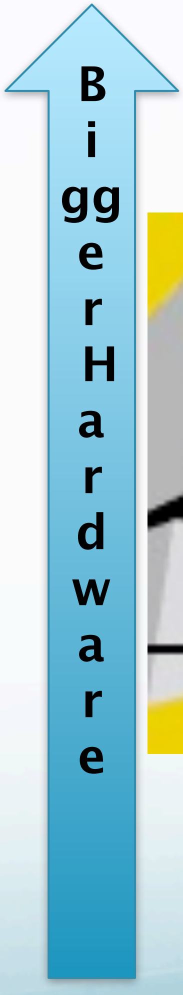
- High Availability Terms and Concepts
- Levels of High Availability
- What technologies are there for MySQL High Availability
- Example of how to implement High Availability
- How we leverage High Availability

HA terms

- High Availability is the availability of a system or service despite hardware failures
  - When you think of HA you think of five nines (99.999%)
  - A lot of companies are ok with three nines (99.9%)
- Two ways to gain High Availability
  - Redundant hardware and software
  - Commercial software
- Continuous Availability
  - No disruption of service during failover
- Single point of failure
  - Can one part of a system bring the whole system down
- Failover and or Failover Situations

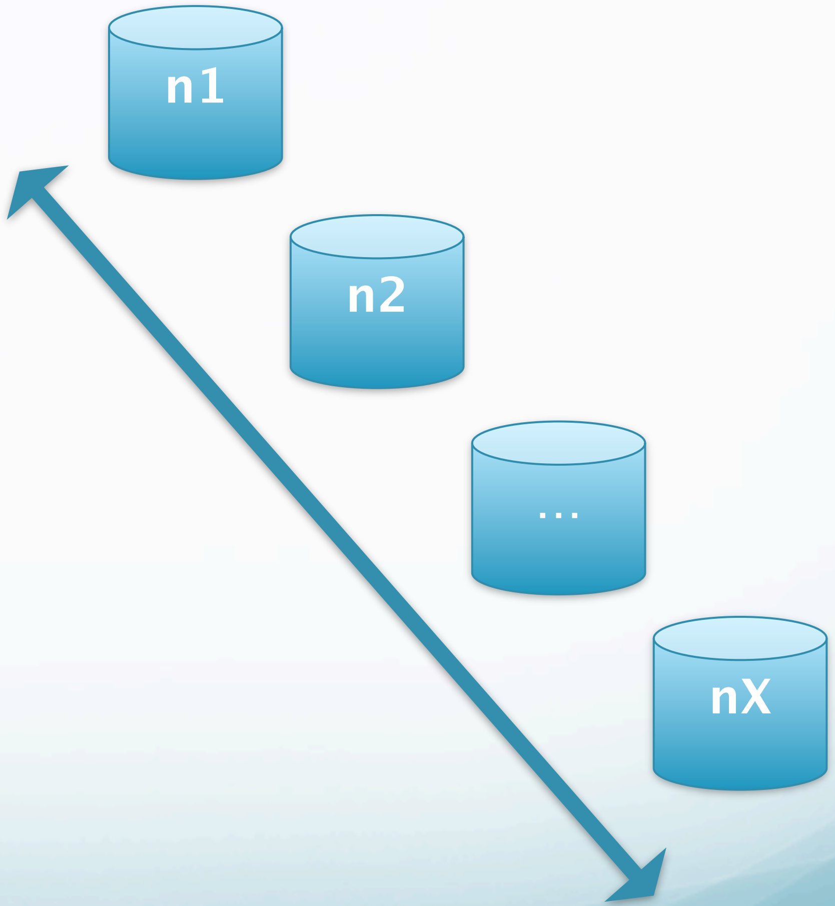
Scaling Up

- Vertically scaling your hardware
  - Bigger, better, faster, stronger!
- Cost of Scaling up
  - e25K
- Maintaining Vertical Scale
- Two Example Companies who specialize in Vertical Scale
  - Schooner – http://www.schoonerinfotech.com/
  - Fusion I/O – http://www.fusionio.com/



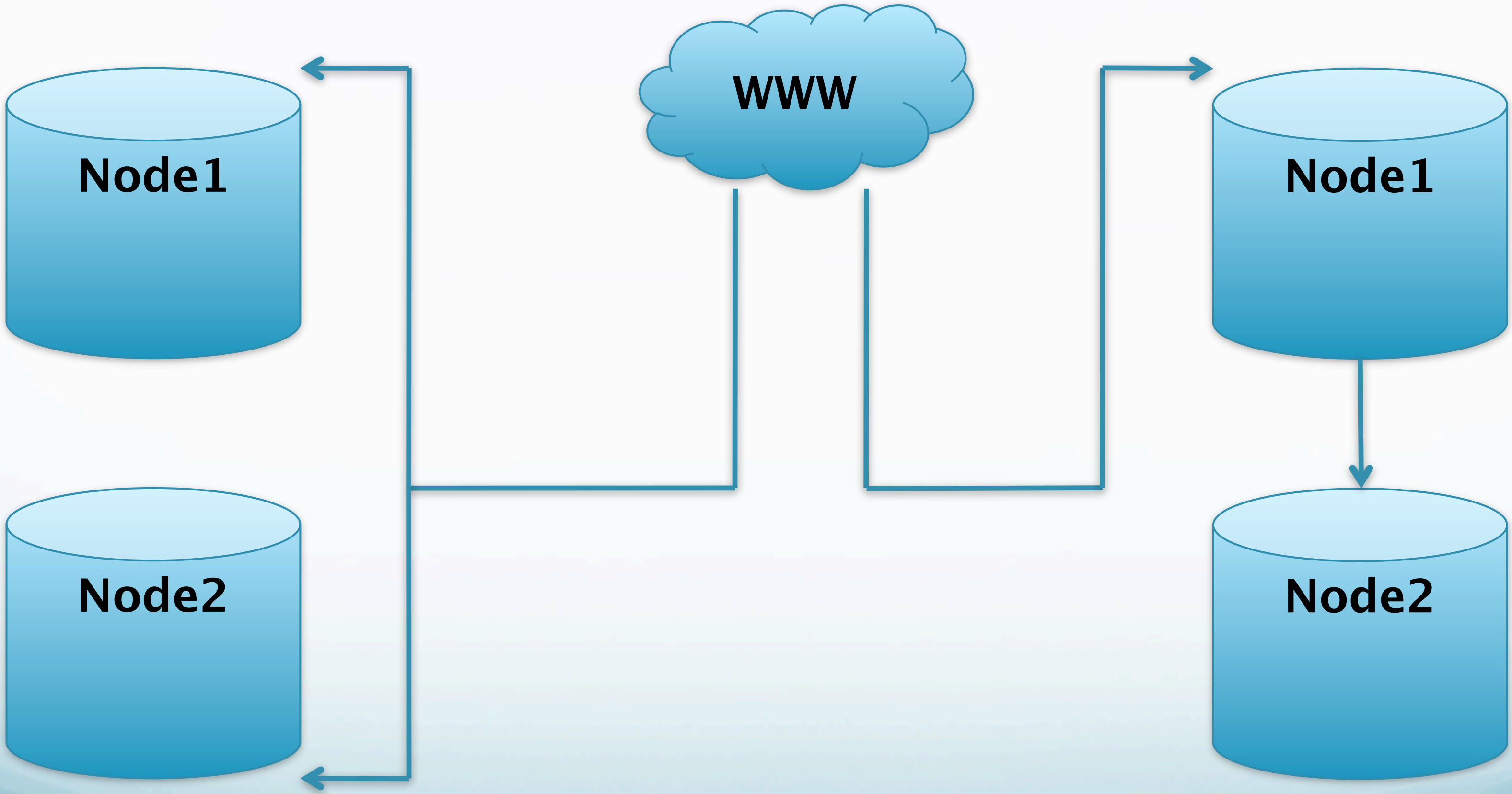
Scaling Out

- Scaling out Horizontally is what you see in most Web 2.0 companies today
- Commodity hardware
  - Dell, HP, SUN - 1U - 4U servers
- Open source software
  - CentOS
  - MySQL
  - DRBD + Heartbeat
  - MySQL Cluster
- Add servers or shard for increase capacity and or performance



Synchronous vs. Asynchronous Replication

- Synchronous
- Asynchronous



- Both Nodes at the same time
- Node1 first, then Node2

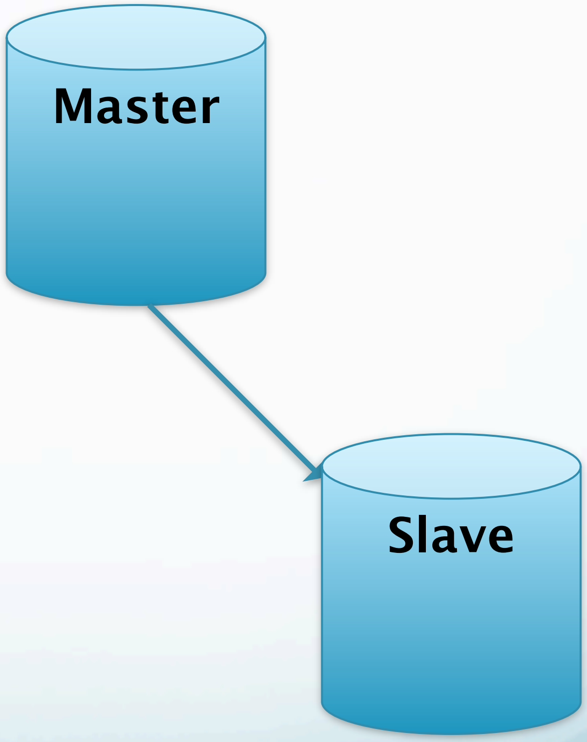
Levels of Availability

- Levels of MySQL Availability
  - **Availability** – Unmanaged Replication
  - **More Availability** – Managed Replication with possible third-party software
  - **Even More Availability** – Managed Multi-Master/Hot Standby Replication with custom and or third party software
  - **High Availability** – DRBD
  - **High Availability** – MySQL Cluster

- The NINES in Downtime
  - 9% = 35 Days
  - 99% = 4 Days
  - 99.9% = 8 Hours
  - 99.99% = 50 Minutes
  - 99.999% = 5 Minutes

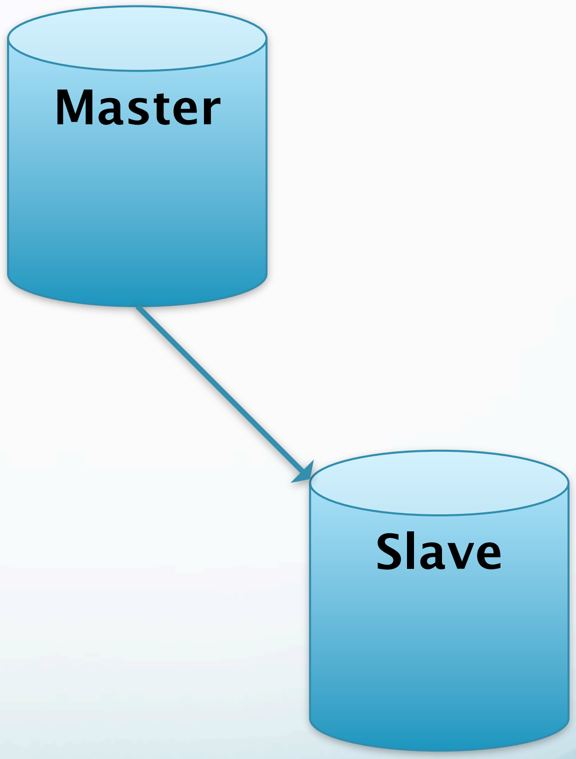
Unmanaged Replication

- Where most Web 2.0 companies start
  - A lot of risk
  - Low end, non-redundant hardware
- Asynchronous One Way Replication
- One Master server
  - Write and Reads
- One Slave server
  - Just sits there
- No automated Break-Fix
  - Manual intervention for everything



Managed Replication

- Production Ready but not the best
  - Less risk
  - Higher end, redundant hardware
- Asynchronous One Way Replication
- One Master server
  - Writes and Read
- One Slave server
  - Possible reads
  - Backups
  - ETL Processes
- Automated Break-Fix
- Fully Monitored

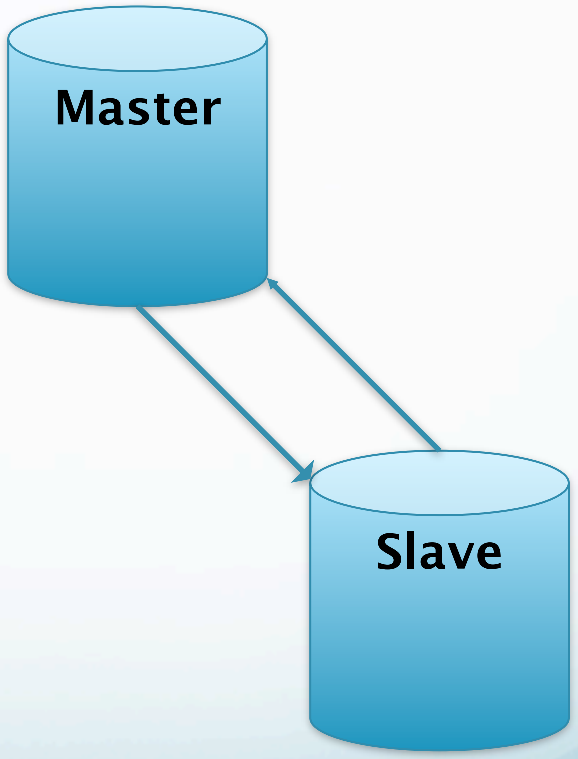


Failovers

- Unmanaged Replication
  - Excessive Customer Downtime
  - Manual process that involves multiple teams or individuals
  - Rebuilding the old master server after failover
  - Hard to keep everything straight at 3am
- Managed Replication
  - Excessive Customer Downtime
  - Rebuilding the old master server after failover
  - Hard to keep straight at 3am

Multi-Master/Hot Standby

- Production ready
  - Redundant hot swappable hardware
    - NICs
    - Disks
    - Power Supply
- Asynchronous Two Way Replication
- One Master server (Rack 1)
  - Reads and writes
  - Possible backups
  - Mk-query-digest
- One Slave server (Rack 2)
  - Read activity
  - Backups
  - ETL
- Automated Break-Fix
- Semi-Automated or Fully Automated Fail-Over
  - Keepalived with a floating IP (VIP)



What's important in Multi-Master/Hot Standby

- Bidirectional Replication
  - Read-only is set on slave server
- Mk-query-digest (http://www.maatkit.org/doc/mk-query-digest.html)
  - Takes read activity on the Master server and runs it on the Slave server
  - This keeps the caches on the Slave server hot
- Mk-table-checksum
  - Perform an online replication consistency check, or checksum MySQL tables efficiently on one or many servers
  - Be sure to use the -replicate flag
- Mk-table-sync
  - Synchronize MySQL tables efficiently
  - This tool changes data, so for maximum safety, you should back up your data before you use it
- Keepalived
  - VIP is attached to the Master server
  - The application talks to the VIP for WRITES

Keepalived

- Not the only solution but works great
- You can use this to handle just hardware failures and kernel panics or run a custom monitoring script to detect InnoDB corruption or other MySQL problems
- Priority of servers
  - Both nodes are set up with the same priority in the keepalived.conf file for ONE way failover
  - Master
    - state BACKUP
    - priority 100
  - Slave
    - state BACKUP
    - priority 100
  - Priority could cause problems when in a Master / Backup configuration
- We want single, one way failover to prevent VIP flapping (happens in odd cases) and to maintain data integrity
- Plus, the probability of both servers experiencing a catastrophic failure are remote

Multi-Master Replication Manager for MySQL

- Great set of scripts if you don’t want to build your own from scratch
  - No keepalived needed
  - No custom monitoring script for MySQL
- Will perform monitoring, failover and management of MySQL master-master replication (with only one node writable at any time)
- Read balance standard master/slave configurations
  - Can handle 1 to many slave servers in a cluster
  - Moves VIP around read servers if they are behind
- Requires at least two MySQL hosts and one monitoring host
- Agent Based monitoring

Leveraging HA Failovers

- A real world example on how multi-master can help
  - Altering a table or multiple tables
  - Large tables (5GB to over 100GB)
  - Limited amount of allotted customer facing downtime, less than 5 minutes
- Problems with the above situation
  - If you've ever tried to run ALTER table on a 100GB table I bet it never finished
  - Do you have enough disk space
  - Table space(s)
  - How much customer downtime do you have to play with

Schema Change – Large tables

- Process
  - Stop replication
  - Run an alter or full dump out and reload on the slave server
    - With ALTER you may or may not want to specify set sql_log = 0;
    - With a dump out and reload you should use the following two features:
      - SELECT INTO OUT FILE
      - LOAD DATA INFILE
      - NOTE: Make sure you dump out by Primary key order
  - Fail the VIP over
  - Depending on how you ran the schema change you'll either:
    - Start replication
    - Run the same change on the FORMER Master server

Benefits

- The customer will only see seconds of downtime if any at all
  - The downtime will be during the VIP failover
- If you choose to use the dump out and reload method
  - Defragmentation
  - Innodb Table Space will not grow out of control
  - Disk space could be regained

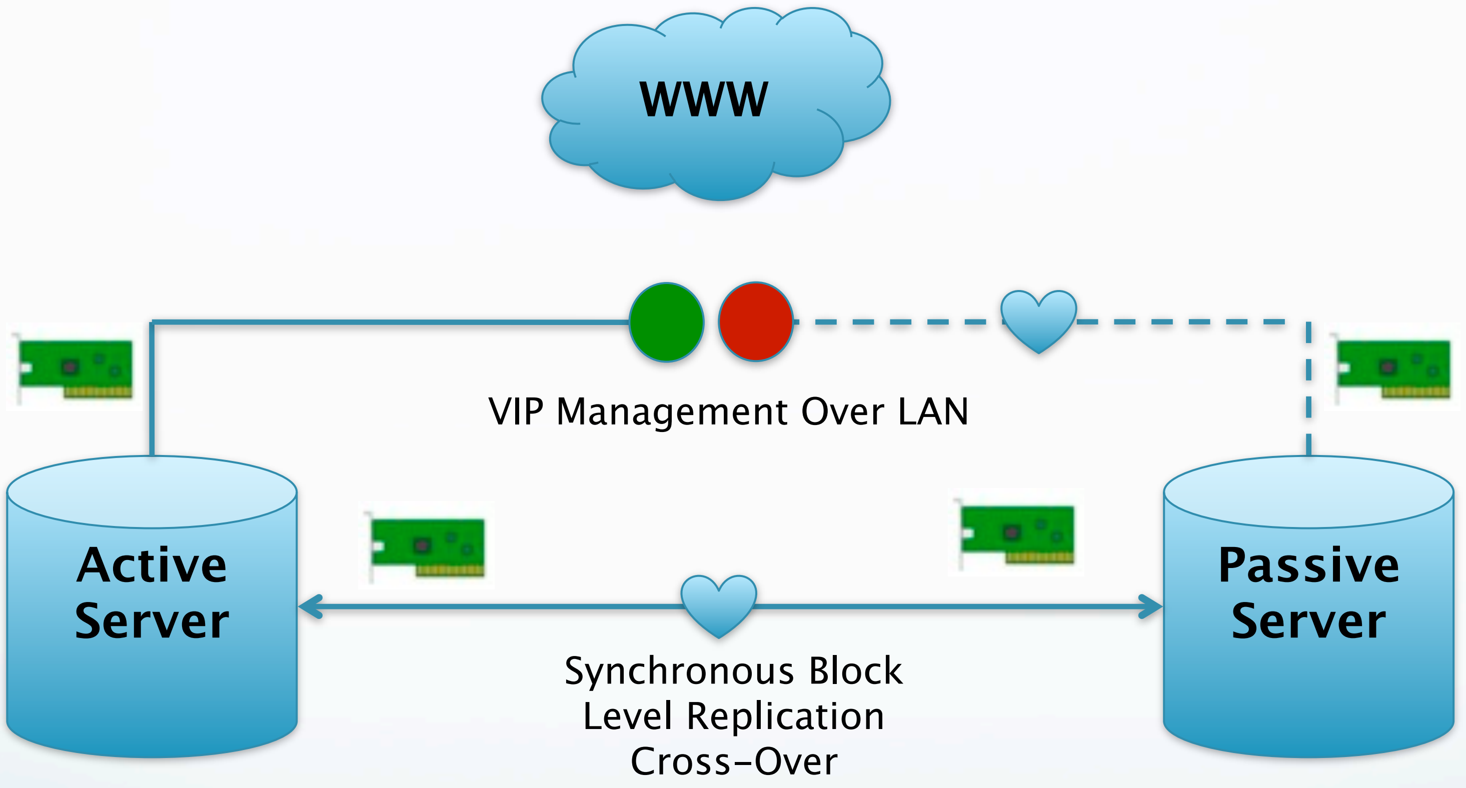
Replication Challenges

- Single Threaded
- Asynchronous
- Can break in a lot of ways
  - Duplicate Key
  - Max Allowed Packet
  - Replication can fall behind
- Can be unreliable

DRBD – Overview

- All components
  - MySQL, Heartbeat and DRBD
- Distributed Storage
- Synchronous Replication (block level)
- No special networking components like HBA’s
- Great Performance (Block vs Statement)
- Manages inconsistencies of data during a failure
- Streamlines many recovery actions
- Automated IP failover and management of VIPs through Heartbeat

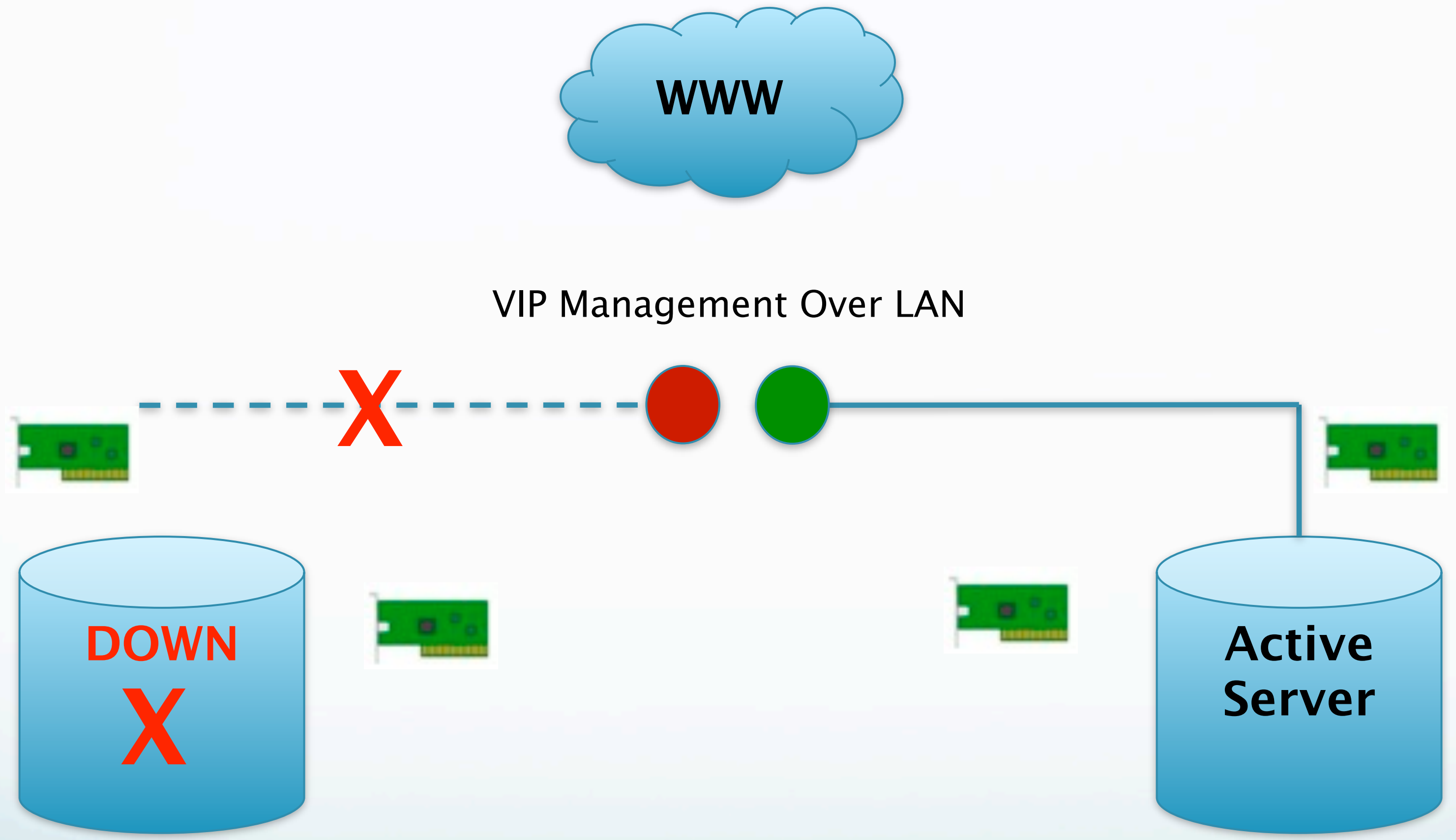
DRBD - Diagram



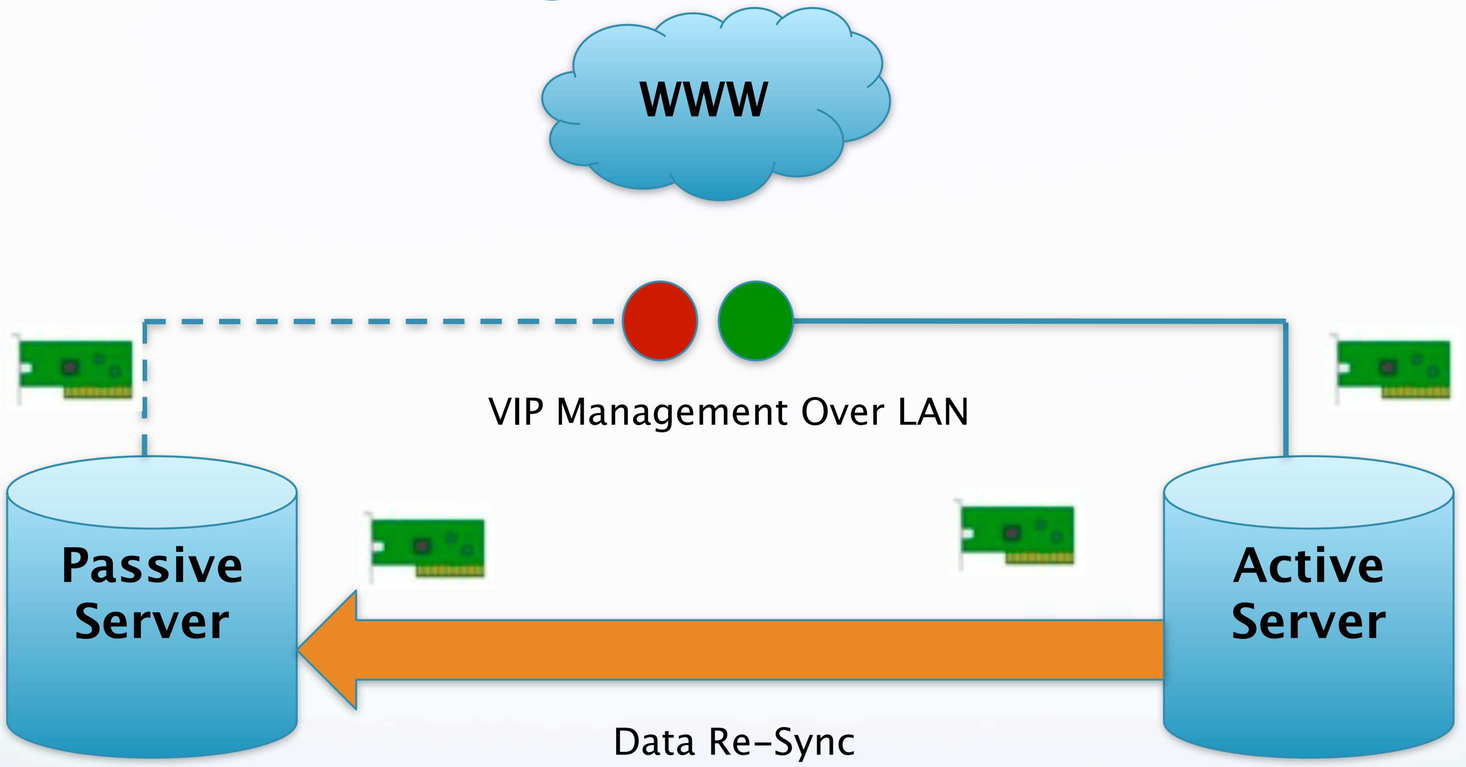
NOTE: Heartbeat needs multiple paths to avoid a split brain scenario

NOTE: Two, 2-port gigabit Ethernet cards on each node

DRBD Failover



Re-syncing Data After Failure

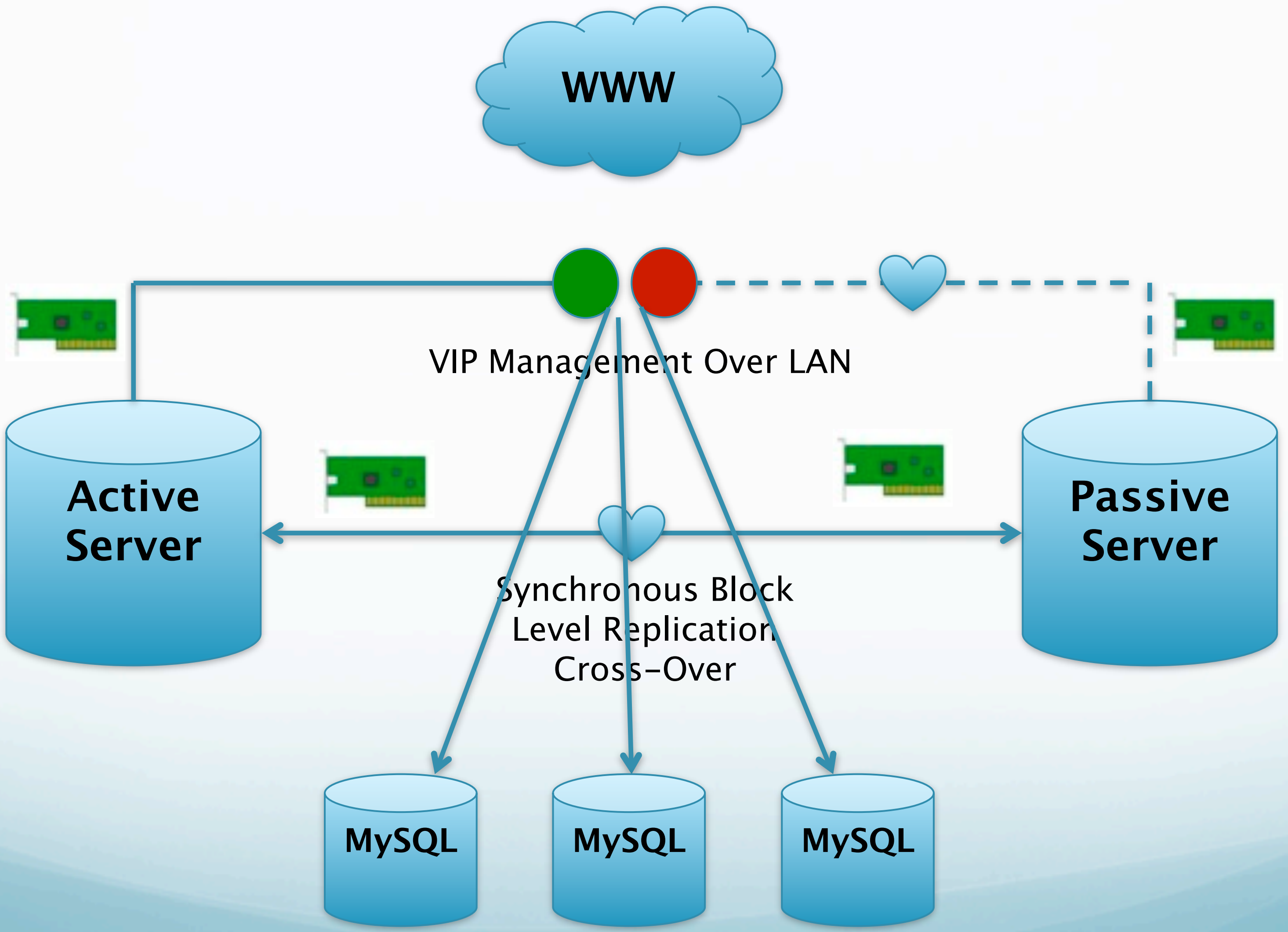


NOTE: No interruption of services during data re-sync

Scaling out with DRBD

- Important
  - DRBD replicates an entire block device
    - This includes Master information
      - binary logs
  - Slaves can attach to the **Virtual IP Address** managed by Heartbeat
  - MySQL Replication allows the slaves to continue with the new Active machine as their master with no intervention needed

Scaling Out with DRBD



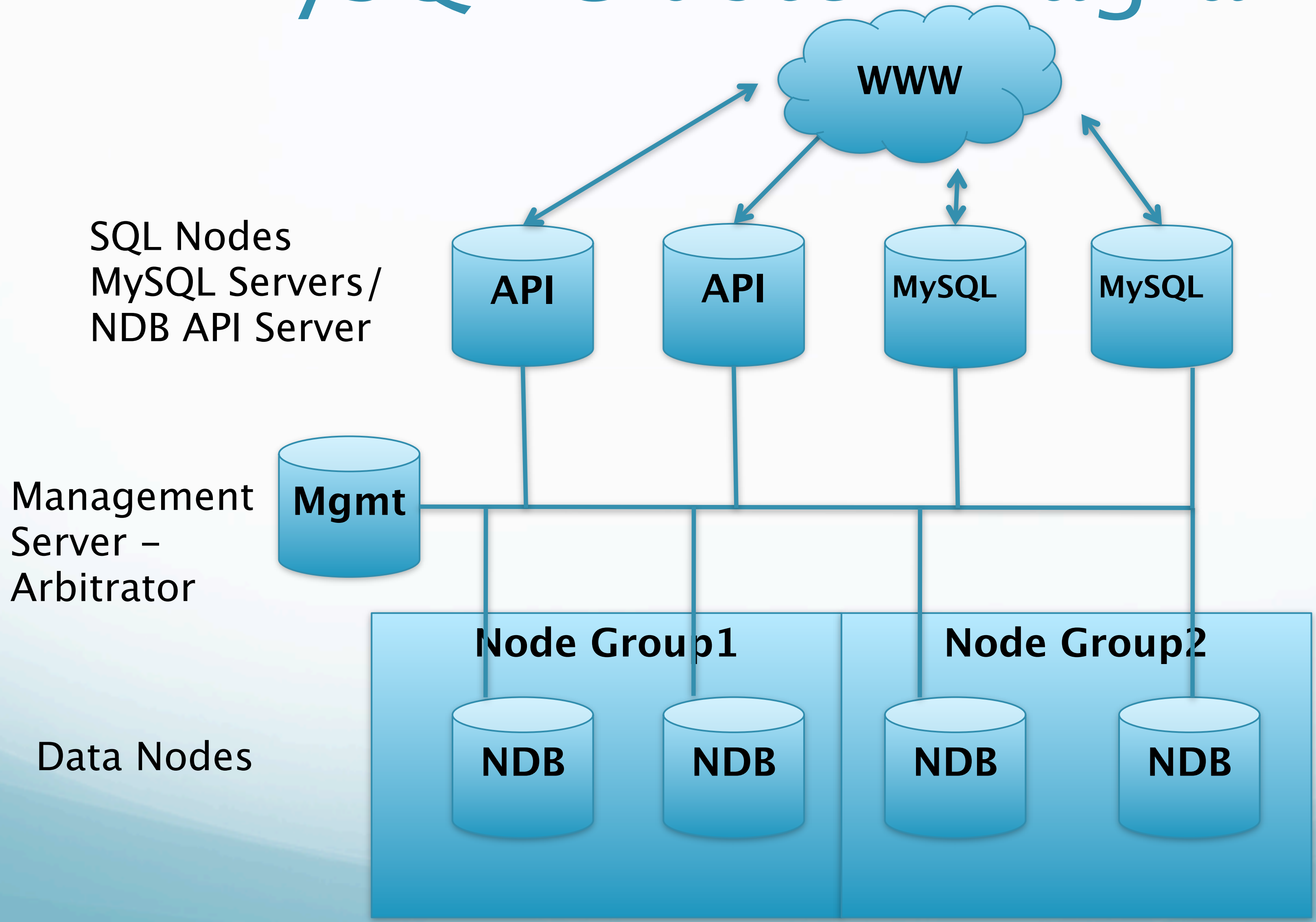
MySQL Cluster Pros

- Redundant nodes provide service when components fail
- Eliminate single points of failure with redundant hardware
  - Multiple network connections
  - SAN or RAID storage
- Share nothing storage
- Automatic data failover built in
- Synchronous Replication
- Transactional
- Online Backups
- No VIP failover required
- Automatic resynchronization of Data
- Scale up and Scale out made easy

MySQL Cluster Cons

- Geographical redundancy through MySQL replication (Asynchronous)
  - Can be a single point of failure
- In Memory data storage
  - Does not work well for large datasets
- Complex normalized schemas with JOIN don’t do well

MySQL Cluster Diagram



When to and when not to use MySQL Cluster

- When we can use MySQL Cluster
  - High Availability is a MUST have
  - Very large amount of update and selects
  - Mostly searching on a Clustered Index
  - Fast failover and crash recovery is needed
  - Geographical redundancy is needed or will be needed
- When we should NOT use MySQL Cluster
  - As a replacement for another storage engine, MyISAM or Innodb
  - When caching would work better (not a replacement for Memcached
  - When you have complex Schemas with a lot of JOINS
  - When you have large data sets

Questions??

- My Information:
  - Chris Schneider
  - chris@ning.com