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 Availability
- What technologies are there for MySQL High Availability
- Examples of how different levels of availability are implemented
- How we leverage High Availability
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
HA Terms

- 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
- Service Level Agreement (SLA)
Scaling Up

- Vertically scaling your hardware
  - Bigger, better, faster, stronger!

- Cost of Scaling up

- Maintaining Vertical Scale

- Two Example Companies who specialize in Vertical Scale
  - Schooner – Schoonerinfotech.com
  - Fusion I/O Fusionio.com
Scaling Out

- Scaling out Horizontally is what you see in most Web 2.0 companies today
- Commodity hardware
  - Dell, HP, SUN (Oracle) – 1U – 4U servers
- Add servers or shard for increase capacity and or performance
Scaling Out – Tier Approach

- It’s FREE, Scale Out
- Open source software
  - CentOS
  - MySQL
  - DRBD + Heartbeat
  - MySQL Cluster
Synchronous vs. Asynchronous Replication

- **Synchronous**
  - Both Nodes at the same time

- **Asynchronous**
  - Node1 first, then Node2
Lower Levels of Availability

- **Available** - Single Node
- **Availability** – Unmanaged Replication
  - Master/Slave
- **More Availability** – Managed Replication with possible third-party software
  - Master/Slave
  - Heartbeat or Keepalived
Higher Levels of Availability

- **Even More Availability** – Managed Multi-Master
  - Hot Standby or Passive Master
  - Replication with custom and or third party software

- **High Availability** – DRBD

- **High Availability** – MySQL Cluster
Levels of Availability
The Nines’ of Uptime in Downtime

• The NINES in Downtime per Year
  • 90.000% = 35 Days
  • 99.000% = 4 Days
  • 99.900% = 8 Hours
  • 99.990% = 50 Minutes
  • 99.999% = 5 Minutes

• Measurements
  • By Node, Cluster, System, Data Center?
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
Replication 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

- One Master server
  - Reads and writes
  - Possible backups - Xtrabackup
  - Mk-query-digest
- One Hot Standby server
  - Read activity
  - Backups - MySQLDump
  - ETL
- Asynchronous Two Way Replication
- Automated Break-Fix
- Semi-Automated or Fully Automated Failover
  - Keepalived with a floating IP (VIP)
Multi-Master/Hot Standby

- Redundant hot swappable hardware
  - NICs: two, 2 port Bonded
  - Disks (RAID 10)
  - Power Supply
- Redundant Switches
- Different Racks
- Different UPS
- Mk-Query-Digest
What’s Important in Multi-Master/Hot Standby

- Redundant hardware
  - Eliminating single point of failure is crucial
- Bidirectional Replication
  - Read-only can be set on Hot Standby Server
- Keepalived
  - VIP is attached to the Master server
  - The application talks to the VIP for WRITES
- Mk-Query-Digest
- Xtrabackup
Keeping Cache Hot / Ensuring Data Integrity

- **Mk-query-digest** *(Keeping Hot Standby Cache HOT)*
  - 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** *(Data Integrity)*
  - 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** *(Data Integrity)*
  - Synchronize MySQL tables efficiently
  - This tool changes data, so for maximum safety, you should back up your data before you use it
Keepalived – VIP Management

- Not the only solution but works great for this example
  - You should also look at:
    - Heatbeat
    - MySQL Proxy
    - HA Proxy
- Use Keepalived for:
  - Hardware failures
  - Kernel panics
  - Custom monitoring script to detect InnoDB corruption or other MySQL problems
- Plus, the probability of both servers experiencing a catastrophic failure are remote
Keepalived – VIP Management

- We want single, one way failover to prevent VIP flapping (happens in odd cases) and to maintain data integrity
- 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
Multi-Master Replication Manager for MySQL (MMM)

- 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
  - InnoDB Table Space growth and fragmentation
  - How much customer downtime do you have to play with
Schema Change – Large tables

- Example Process (*Very loose 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
Process Benefits

- The customer will only see seconds of downtime if any at all
  - The downtime will be during the VIP failover
  - Make sure to run this during off peak

- 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
  - Bottle Neck

- Asynchronous
  - Master, then Slave

- Can break in a lot of ways
  - Duplicate Key
  - Max Allowed Packet
  - Replication can fall behind

- Can be unreliable
DRBD – Overview

- Components Needed for Success
  - MySQL, Heartbeat and DRBD
- Distributed Storage
- Synchronous Replication
- No special networking components like HBA’s needed
- 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

VIP Management Over LAN

DOWN X

Active Server
Re-syncing Data After Failure

NOTE: No interruption of services during data re-sync
Scaling out with DRBD

- 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

Active Server

MySQL

VIP Management Over LAN

Synchronous Block Level Replication Cross-Over

MySQL

MySQL

MySQL

Passive Server

WWW
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
- Easy node addition
- 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

SQL Nodes
MySQL Servers/
NDB API Server

Management
Server -
Arbitrator

Data Nodes

Node Group1
NDB  NDB

Node Group2
NDB  NDB

WWW

API

API
When to 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 NOT to 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