MySQL Architecture Design Patterns for Performance, Scalability, and Availability

Brian Miezejewski
Principal Manager Consulting

Alexander Rubin
Principal Consultant
Agenda

- HA and Scale-Out
- Architectural Tools
  - MySQL Replication
  - DRDB Clustering
  - MySQL Cluster
- Design Patterns
  - Simple Master-Slave
  - Master and Many Slaves
  - Master-Master
  - DRBD Pair
  - DRBD Master with Many Slaves
  - MySQL Cluster
- Scale-out/Sharding
MySQL High Availability

• **What is High Availability?**
  – Availability of resources in a computer system

• **Continuous Availability**
  – Non-stop service
  – No disruption of service even during a fail over
The Five 9s of Availability

- **Well-Managed**
  - power failures
  - network failures
  - hardware failures
  - software failures
  - maintenance operations

- **Unmanaged**
  - software upgrades
  - hardware upgrades
  - disasters

- **Clustering & Geographical Redundancy**
  - Clustering Technologies
  - Replication Technologies

- **35 days**
  - Small Business

- **4 days**
  - ISPs & Mainstream Business

- **8 hours**
  - Data Centers

- **50 mins**
  - Banking

- **5 mins**
  - Medical
  - Telco
  - Military

- **%**
Scale-Up vs. Scale-Out

• **Scale-Up**
  - Vertical
  - Expensive SMP hardware
  - Proprietary software
  - Platform lock-in
  - “Fork Lift” to increase capacity & performance

• **Scale-Out**
  - Horizontal
  - Commodity Intel/AMD hardware
  - Open source software
  - Platform independence
  - Add servers to increase capacity & performance
MySQL Replication

MySQL Master

Web/App Server

Writes & Reads

data

index & binlogs

mysql

MySQL Slave

I/O Thread

SQL Thread

relays binlog

binlog

data

Replication

writes
MySQL Replication (Scale-Out)

Possible Roles
- Fail over server
- Used for performing backups
- Read/Write load balancing
- Additional slaves allow Scale-Out

Master Server
- Writes
- Index & Bin Log Rotation

Slave Server
- Writes & Reads

Backups

Web/App Server

Writes & Reads

Reads
MySQL Replication - Fail Over

Possible Roles
- Fail over server
- Used for performing backups
- Read/Write load balancing
- Additional slaves allow Scale-Out

Manual Fail Over

Web/App Server

Reads

Backups

MySQL Replication

Index & Bin Log Rotation

Master Server

Writes

Fail Over

Server

Writes
Linux Heartbeat, DRBD & MySQL

Active Server

= Private IP = 10.10.10.20

Primary DRBD

Passive Server

= Private IP = 10.10.10.21

Secondary DRBD

DRBD

Linux Heartbeat

= Virtual IP = 10.10.10.10

Web/App Server

= Private IP = 10.10.10.21

Web/App Server
Linux Heartbeat, DRBD & MySQL

Web/App Server

Linux Heartbeat
= Virtual IP =
10.10.10.10

Active Server
/Private IP =
10.10.10.21

Passive Server
/Private IP =
10.10.10.20

DRBD

Primary DRBD

Active Server
/Private IP =
10.10.10.21

Active Server
/Private IP =
10.10.10.20

Active

ORACLE
Scale-Out Cluster Architecture

MySQL Server

MySQL Server

MySQL Server

MySQL Server

MySQL Server

Data Node

Data Node

Data Node

MySQL Cluster

NDB API

NDB Storage Engine

Management Server

Management Server

ORACLE
### MySQL Cluster Data Node Architecture

<table>
<thead>
<tr>
<th>ID</th>
<th>Capital</th>
<th>Country</th>
<th>UTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Copenhagen</td>
<td>Denmark</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Berlin</td>
<td>Germany</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>New York City</td>
<td>USA</td>
<td>-5</td>
</tr>
<tr>
<td>4</td>
<td>Tokyo</td>
<td>Japan</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>Athens</td>
<td>Greece</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Moscow</td>
<td>Russia</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>Oslo</td>
<td>Norway</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Beijing</td>
<td>China</td>
<td>8</td>
</tr>
</tbody>
</table>

- **Partition 1**: Data Node 1 - Primary, Data Node 2 - Secondary
- **Partition 2**: Data Node 3 - Primary, Data Node 4 - Secondary
- **Partition 3**: Data Node 5 - Primary, Data Node 6 - Secondary
- **Partition 4**: Data Node 7 - Primary, Data Node 8 - Secondary

- **Node Group 1**: P1-Primary, P2-Secondary, P1-Secondary, P2-Primary
- **Node Group 2**: P3-Primary, P4-Secondary, P3-Secondary, P4-Primary

- **Four Data Nodes**
- **Two Replicas**
- **Two Node Groups**
Scale-Out Cluster Architecture – Fail Over

MySQL Server

Data Node

NDB Storage Engine

Management Server

MySQL Cluster

NDB API
Patterns

- Master/Slave Replication
- Master with Read Only Slaves
- Master to Master
- Ring Replication
- DRDB HA
- DRBD Master Pair with Read Only Slaves
- Master to Master Cross Datacenter + DRDB
- Simple Sharding
- Large Sharing/Scale-out solution + DRDB
- Sharding + DRDB Fail-Over + Geographical Redundancy
Pattern: Master/Slave Replication

- **Advantages**
  - Can read from either server
  - Up to 2x read scaling
  - Easy to setup, configure, and maintain
  - Easy fail-over, fail-back
  - Poor mans HA, ~99.9% (3 nines) HA

- **Disadvantages**
  - ~0.5 write scaling or slave will fall behind
  - All writes **must** go to the master
  - Replication is asynchronous; potential delay in writes to slave
  - Data can be left “un-replicated” on the Master in a fail-over.
    - Saving and re-syncing data left in the master on fail-over is a manual task that is not always straightforward.
Pattern: Master with Read Only Slaves

- A master can have many slaves,
  - ~1% load on master per slave
  - 5+ slaves are not unusual, Most I’ve heard of is 80 slaves
- Reading data is HA, can be designed to be 4-5 nines
- Add more slaves to scale read load
- Can load balance reads across slaves
- Can do long running reports on slave
- Use slave for backup
- JDBC drivers support this architecture
Pattern: Master to Master Replication

Master 1

• Advantages
  – Can read from or write to either server
  – Works across WAN - Supports geographical redundancy
  – Can be setup as active/passive
  – Can get some performance advantage by splitting load and keeping it sticky, i.e. east coast always on master 1 and west always on master 2

Master 2

• Disadvantages
  – Total insert/update/delete load is ~.5 one server or slaves begin falling behind
  – Applications and schema must be designed to eliminate duplicates that would break replication
  – Fail-back can be more complicated
Pattern: Ring Replication

- Advantages
  - Can read from or write any server
  - Works across WAN - Supports multiple geographical redundancy
  - BP - Use DRDB HA for each

- Disadvantages
  - Total insert/update/delete load for all servers is ~.5 of typical load of one server or slaves begin falling behind
  - Applications and schema must be designed to eliminate duplicates that would break replication
  - Fail-over and Fail-back is very complicated

- Don’t try this at home!
  - Unless you really know what you are doing!
Pattern: DRBD HA

- **Advantages**
  - No data loss
  - Can replicate from the pair via the fail-over VIP
  - Allows for heavier load on primary than replication on master
  - Fully automated fail-over when using Heartbeat
  - Very easy to rebuild old primary as new secondary after fail-over
  - Can be 3-4 9s Available

- **Disadvantages**
  - Secondary is passive
  - 30-300 Second delay in recovery after fail-over
  - Only works on Linux
**Pattern: DRBD Master Pair with Read Only Slaves**

- **HA** - Both for reads and write
- **Scale-out** for reads
- Often the optimal solution for most needs
- Replication is via fail-over VIP so slaves transparently fail-over to secondary on primary failure
- Can load balance across slaves
Pattern: DRBD Master Pair with Read Only Slaves
Pattern: Master to Master Cross Datacenter + DRDB
Pattern: Simple Sharding

Every app connects to every Shard

- App 1 (Shard F)
- App 2 (Shard F)
- App 3 (Shard F)
- App 4 (Shard F)
- App 5 (Shard F)
- App 6 (Shard F)
- App 7 (Shard F)

- Shard A: Customers 1 to 1,000,000
- Shard B: Customers 1,000,001 to 2,000,000
- Shard C: Customers 2,000,001 to 3,000,000
- Shard D: Customers 3,000,001 to 4,000,000
Pattern: Large Sharing/Scale-out solution + DRDB
Pattern: Sharding + DRDB Fail-Over + Geographical Redundancy
The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.