MySQL High Availability

Michael Messina
Senior Managing Consultant, Rolta-AdvizeX
mmessina@advizex.com / mike.messina@rolta.com
Introduction

• Michael Messina
• Senior Managing Consultant Rolta-AdvizeX,
• Working with Oracle Approximately 25 years, MySQL about 12 years.
• Background includes Performance Tuning, High Availability and Disaster Recovery
• MySQL Cloud Service 2018 Implementation Specialist
• Oracle Database OCP
• Oracle RAC Certified Expert
• Oracle Exadata Implementation Specialist
• Oracle ACE
• MMESSINA@ADVIZEX.COM / MIKE.MESSINA@ROLTA.COM
• www.advizex.com
Agenda

• MySQL History
• Why MySQL
• High Availability
• Virtualization
• High-level General MySQL Best Practices
• MySQL Replication Overview
• MySQL Replication Tools to Help Higher Availability
  ▪ MySQL Router
  ▪ MySQL High Availability (MHA)
• MySQL Replication
  ▪ Asynchronous Master/Slave Replication
  ▪ Semi-Synchronous Master/Slave Replication
Agenda

• Group Replication (Native MySQL High Availability/Enterprise HA)
  ▪ InnoDB Cluster

• MySQL Cluster (NDB)

• Questions / Discussion
MySQL History

- MySQL was created by a Swedish company, MySQL AB
- First release on 05/23/1995, making a 30 year History for MySQL
- Version 3.20 released January 1997
- Version 3.21 production release in 1998
- Version 4.0 Beta release August 2002
- Version 4.0 Production release March 2003
  - Unions
- Version 5.0 Beta release March 2005
- Version 5.0 Production release October 2005
  - cursors, stored procedures, triggers, views, XA transactions
MySQL History

- Sun Acquires MySQL AB in 2008
- Version 5.1 Production Release November 2008
  - event scheduler, partitioning, row-based replication, server log to tables
- Oracle Acquires Sun Microsystems January 2010
- MySQL Server 5.5 December 2010
  - Unicode character sets utf16, utf32, and utf8mb4
  - New Partitioning options
- Version 5.6 February 2013
  - full-text search for innodb
  - NoSQL memcache
- Version 5.7 October 2015
  - Group Replication
Why MySQL

- Long History of a Solid Scalable Database (Proven over 30 years)
- Great Proven Performance
- World's Most Popular Open Source Database
- Open Source and run Community Editions for Free
- Large Database Company Oracle more than 20,000 developers
- Enterprise Options for Support by Oracle at very low cost
- Rich tool set to manage and monitor available
- Enterprise Version has Database Firewall a security option from Oracle that only the Oracle Database can compete with no other open source database has this.
- Large Support Community
Why MySQL

- Run in Cloud / Virtualized / Dedicated Hosts
  - Cloud Infrastructure as a Service - IAAS
  - Cloud Database as a Service - DAAS
- Full and Incremental Database Backup Capability
- Point in Time Recovery
- Highly Scalable
  - Many scalability options
- Can Be Made Highly Available
  - Many Options to Provide Redundancy for availability based on needs
High Availability

- **Cost of Downtime (Direct and Indirect Costs)**
  - Damage to Organization Reputation?
  - Impact on client/customer?
  - Regulatory Issues?
  - Loss of Business?

- **RTO (Recovery Time Objectives)**
  - Maximum downtime per incident
  - Driven by the Uptime Requirements

- **RPO (Recovery Point Objective)**
  - Point in which data must be recovered to
  - Defines Maximum amount of loss due to an severe incident
High Availability

Oracle MySQL: Guide to High Availability, Oracle Corporation
High Availability

- **Service Level Agreements (SLAs)**
  - Business Requirements based on RTO, RPO, Cost of Downtime
  - Support Tiers (typically 4 tiers)
    - Tier 1 (MySQL Cluster)
      - 99.999% available for mission critical applications where loss of service has the highest impact and cost to the business.
    - Tier 2 (MySQL Group Replication)
      - Business Critical, but do not need 99.999% availability like Tier 1 therefore some HA level protection is needed just not to the level of tier 1
    - Tier 3 (MySQL Replication and Virtualization)
      - Needs protection from extended periods of loss of service and therefore requires a level of protection but short infrequent outages do not have severe impact on business or its customer.
    - Tier 4 (MySQL Virtualization)
      - Lowest availability and impact to business during loss of service incident typically development and smaller departmental applications fall into this tier.
Virtualization

- MySQL Databases in Virtual Machines
  - Protect against a single server failure
  - Virtual Machine Runs on another physical machine upon failure of physical machine running on

- Using Enterprise Storage with multi-disk failure protection
  - Protect against single and multi-disk failures impacting availability of database and virtual machine storage

- Automate building of new environments

- Can assist with large environments using shared storage for MySQL Binaries that can be shared across many virtual machines

- Place MySQL database volumes on direct mount and attached storage not on VM Files (vmdk) for best performance.
High-level General MySQL Best Practices

• Use Innodb Storage Engine (other then MySQL Cluster of Course)
• Take Advantage of Query Cache where you can
  ▪ High Update Environments and tend to see waits on Query Cache so very high DML activity could impact performance improvements from query cache
• Use innodb_file_per_table – default 5.6.6 and above
• Run with SQL Mode NO_ENGINE_SUBSTITUTION to prevent tables being created in other Storage Engines
• Always Use Redundant Storage for database files
• Always secure database behind firewall
• Do not Allow non-Unix or non-DB Admin access to database server(s)
• Always use MySQL username and host combinations do not use username with host % which would allow user connection from any host.
• Plan for Memory MySQL Runs Best with large innodb buffer pool
MySQL Replication Overview

- **Options**
  - Cluster and non-cluster databases
  - Asynchronous Replication
  - Semi-Synchronous Replication

- **Requirements**
  - Bin-logging must be turned on (transaction logging)

- **Architecture**
  - Master
    - Main database where database connections read and write to the database tables
    - Sometimes Called Primary
  - Slave
    - Database the receives transactions from a Master
    - Typically Database connections only read from the database
  - Master-Slave
  - Multi-Master
    - Each database is both a master and a slave database
MySQL Replication Overview

• Why
  ▪ Protect Database against hardware failure
    ▪ Eliminate Single Point of Failure for database
    ▪ Hardware fails other copies of the database ready to take over
  ▪ Provide read level scaling (scale-out)
    ▪ Write to 1 master
    ▪ Read from many slaves (spread the read load)
  ▪ Disaster Recovery Protection
    ▪ A copy of the database in another data center
  ▪ Rolling Upgrades
  ▪ Offload Backups of main database
  ▪ Create a slave that lags behind master to protect against erroneous data modification. (measured in seconds)
  ▪ Reporting environment
  ▪ Achieve Higher UpTime
MySQL Replication Overview

• Replication Format
  ▪ Statement
    ▪ SQL Statements are written to the binary logs from Master – same statements executed on slave
  ▪ Row
    ▪ Writes row events to binary logs for individual rows
  ▪ Mixed
    ▪ MySQL determines the best format to use for the transaction (statement/row)

• Replication Operation
  ▪ 3 threads
  ▪ Bin log dump thread (Master)
    ▪ Sends binary log to replication slave database
  ▪ Slave IO Thread (Slave)
    ▪ When replication is running connects to the Master and gets binary log data from bin log dump thread and writes to the relay logs on the slave
  ▪ Slave SQL Thread (Slave)
    ▪ Reads from the relay logs on the slave and executes the transactions.
MySQL Replication Tools to Help Higher Availability

- **MySQL Router**
  - Provides transparent routing between your applications and MySQL Servers
  - Lightweight
  - Allows additions and removal of MySQL Servers without application connection changes
  - Good use for Read/write Master and many read slaves.
  - Transparent access for MySQL Master-Slave and Group Replication
  - Load balancing without expensive load balancer

- **MySQL Master High Availability (MHA)**
  - Third party tool to facilitate failover from master to a slave in an automated fashion
  - Utilizes Virtual IP address for Master server that is failed over as part of the process so application connection strings to Read/write database never have to change.
  - Utilizes 3rd node to facilitate control (Manager)
  - Can handle automatic failover in event of master failure
MySQL Router Architecture

Client Application

MySQL Router

MySQL Master
MySQL Slave
MySQL Slave

Client Application
MySQL Router

[routing:balancing]
bind_address = localhost
bind_port = 7001
destinations = server1:3306,server2:3306,server3:3306
mode = read-only

[routing:read_write]
bind_address = localhost
bind_port = 7002
destinations = server1:3306
mode = read-write
Master High Availability (MHA) Architecture
MySQL Replication (Asynchronous Replication)

- Default Replication Type
- Does not require any acknowledgements from a slave before master can proceed.
- No Guarantee that transaction reaches slave
- Does not affect Master Transaction Processing if slave issues.
- Can support delayed replication (5.7)
  - Set slave to be so much time behind master (MASTER_DELAY setting)
  - Good for protecting against logical data corruption by rogue process or error.

- Master-Slave
  - One master
  - One or More Slaves

- Master-Master
  - Limited to 2 master servers only can not have more then 2 masters
  - Round Robin Replication between the masters last in wins
  - No Collision Handling
MySQL Replication (Semi-Synchronous Replication)

- 5.7 and above
- Implemented via a plugin
- Requires a transaction on master be acknowledged by at least 1 slave prior to continuing transactions.
- Can cause performance issues on master if slaves are not able to handle transaction activity of master
- Assured at least 1 slave has transactions for better protection
- If no slave can get the transaction replication via timeout replication automatically reverts to asynchronous replication.
  - rpl_semi_sync_wait_for_slave_count
    - # of slaves that have to receive transaction, default 1
MySQL Group Replication

- Implemented via MySQL Server Plugin
- High Availability
- Scalable
- Fault tolerant can Achieve 99.99% uptime
- Modes
  - Single-primary mode (default)
    - Only 1 server accepts updates at a time (primary)
    - Automatic primary election
  - Multi-primary mode
    - More then one member of the groups is written to at a time
- Built-in group membership service
  - Sees the group of servers in the replication and their current status
  - Detects when servers leave or join the group
  - Allows servers to seamlessly leave and join the group without impact
MySQL Group Replication

• Failover
  ▪ Client side failover is not handled automatically
  ▪ Load Balancer
  ▪ MySQL Proxy
  ▪ MySQL Router

• Minimum of 3 Recommended
  ▪ So if one is lost protection still exists
  ▪ Scalable to 9 Servers in Group

• Monitoring
  ▪ Group Replication is not monitored by slave status it is monitored via the performance_schema
    ▪ replication_group_member_stats
    ▪ Server state ONLINE indicates group member is in sync
    ▪ UNREACHABLE would be a status of a lost member
    ▪ RECOVERING for a member coming into the group
    ▪ ERROR indicates issue that needs attention
  ▪ replication_group_members
MySQL Group Replication

- Required InnoDB Storage Engine
  - Works same and uses same engine as MySQL with no Group Replication or Regular Replication
- Database versions at same major release or higher are allowed to join group replication help availability for upgrades. Keeping mind higher major versions will not be allow to write only read.
- Can apply transactions to other databases in the group in parallel
  - Reduce apply lags
  - Improve Performance of Replication
  - Can Still Preserve Commit order
- Requires binlog Replication row based format
- Can be used in the Cloud
- Multi Data Center Capable
MySQL Group Replication

- Great Illustration of Single-Primary mode from Oracle Documentation
MySQL Group Replication

- Great Illustration of Multi-Primary mode from Oracle Documentation

1. All servers are primaries.
2. Server S1 fails.
3. S1’s client connects to S3.
MySQL Group Replication

• Network is Crucial to Group Replication esp. Multi-Primary

• Consensus
  ▪ Control changes that need to be replicated
  ▪ Control changes in the Group
  ▪ Majority in Group Agrees
  ▪ Unable to progress without majority
  ▪ Example: 5 servers in group 3 are lost no consensus can exist, Manual intervention must be made in this case
    ▪ Fix Servers
    ▪ Force Membership group to remaining servers
  ▪ When Update of same record from 2 different locations in the group first one to commit transaction wins.
  ▪ Example: 5 servers exist and 3 leave group one by one normally then the reconfigure down to 2 servers is able to happen and consensus can exist
MySQL Group Replication

- **Security**
  - White List by IP Address
    - `group_replication_ip_whitelist="10.1.10.112,10.1.19.113,10.1.10.114";`
  - SSL
    - Replication Connections
    - Recovery Connections
    - OpenSSL Supported
    - YaSSL Supported
    - Replication user requires SSL
      - `CREATE USER 'rep_ssl_user'@'%' REQUIRE SSL;`
      - `group_replication_recovery_use_ssl=1;`
      - `SET GLOBAL group_replication_recovery_ssl_ca= '.../cacert.pem';`
      - `group_replication_recovery_ssl_cert= '.../client-cert.pem';`
      - `group_replication_recovery_ssl_key= '.../client-key.pem';`
  - VPN
    - Operate Group Replication over Virtual Private Network
MySQL Group Replication

- **InnoDB Cluster**
  - Group Replication -> Can be run with No InnoDB Cluster defined
  - MySQL Router
    - Recommend to Install at Client/Application Server
    - Controls routing connections to primary database in group
    - Controls connections to read-write then to read slaves for readonly connections
  - MySQL Shell
    - We Install MySQL Shell on a MySQL Database Nodes
    - Interactive and Batch Code Execution
    - Create InnoDB Cluster
    - Manage InnoDB Cluster
    - Automation
Group Replication

- InnoDB Cluster
  - MySQL Shell
    - SQL
    - JS – Jave Script
    - PY – Python
  - Example:
    - mysqlsh –sql
    - mysqlsh –js  -- This is the default
    - mysqlsh –py
  - Create InnoDB Cluster
    - dba.createCluster('MyCLuster')
  - Add instance to Cluster
    - cluster.addInstance('user@hostname:3306')
  - Verify Cluster Status
    - cluster.status()
MySQL Cluster

- Write scalable
- Real time in memory
- ACID Compliant Transactional Database
- NDB Storage Engine
- 5 9s Availability Capable (99.999%)
- Distributed No Single Point of Failure Architecture
- Capable of millions of operations per second
- Full Active-Active Geographic Replication
- Horizontal scalability
  - Automatic Sharding
- SQL and No SQL Interfaces
  - MySQL Cluster’s native NoSQL NDB API
  - Java, PHP, .Net
MySQL Cluster

• 3 Different Node Types
  ▪ Data Node
    ▪ Storage and management of both in-memory and disk-based data
    ▪ Automatic and user defined partitioning (sharding) of tables across data nodes
    ▪ All Data Node accepts writes
    ▪ Synchronous replication of data between data nodes
    ▪ Transactions and data retrieval
    ▪ Automatic fail over
    ▪ Automatic resynchronization after failure for self-healing
  ▪ Application Nodes
    ▪ Connectivity from Application to the data nodes
  ▪ Management Nodes
    ▪ Cluster Configuration to all nodes
    ▪ Used for Node when joining cluster
    ▪ System Reconfiguration
    ▪ Provide Arbitration (Network Failure, Node Failure etc.) to prevent Split Brain
MySQL Cluster

- All hosts used in the cluster must have the same architecture.
  - Do not Mix Operating Systems and architectures within the cluster
- disk-based as well as in-memory data
  - Non-indexed data can reside on disk
  - Indexed columns must always be in memory
  - Larger the database the larger the amount of memory that will be needed and potentially the more nodes required
- Cluster Sizing
  - (in memory) Data Size * Replicas * 1.25 = Total Database Memory
  - Example:
    \[ 50 \text{ GB} \times 2 \times 1.25 = 125 \text{ GB} \]
    \[ \frac{(\text{Data Size} \times \text{Replicas} \times 1.25)}{\text{Nodes}} = \text{RAM Per Node} \]
    Example: \( \frac{2 \text{ GB} \times 2 \times 1.25}{4} = 31.25 \text{ GB} \)
MySQL Cluster

• Limit Full table scans
  ▪ Due to the way in which data is partitioned and distributed across many Data Nodes full tables scans can lower performance
  ▪ Primary key operations will typically benefit most from MySQL Cluster’s distribution of data
  ▪ Provides the best performance when used to process large volumes of relatively small transactions
MySQL Cluster Manager

- MySQL Cluster Manager (MCM) simplifies the creation and administration of the MySQL Cluster
  - Agent Runs on each nodes in the MySQL Cluster
  - Simplify the creation of the cluster
  - Simplify the Management of the cluster
  - Monitors for issues and will automatically recover MySQL application nodes, management nodes, and MySQL Cluster data nodes.
  - Cluster Manager for management operations allows rolling restarts in the cluster with a single command saves management time for management operations.
  - Control the entire Cluster as a single entity however can control down to individual processes.
  - Automate Upgrade, downgrade and reconfiguration of the cluster including node addition and node removal.
  - Does not replace the Need for a Management Node in the cluster
Credits

- MySQL Documentation, Oracle Corporation
- MySQL: A Guide to High Availability, Oracle Corporation
- Guide to Scaling Web Databases with MySQL Cluster, Oracle Corporation
Thank You!

Nobody does Oracle better

Any Further Questions?