CLOUD COMPUTING APPLICATIONS

Cloud Databases – Multinode RDBMS

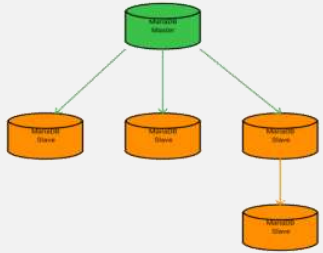
Prof. Reza Farivar

Beyond a Single Node

- Replication Options in MySQL
  - Classical MySQL Replication
    - One master, multiple slaves
    - Support for Many masters to many slaves
    - asynchronous
    - No Conflict Resolution or “Protection”
  - MySQL Group Replication
    - distributed state machine replication with strong coordination between servers
    - Built on Paxos
      - Majority vote for transaction commit
      - Network partition can stop the system
  - Multi Master - Galera
    - Multi-master
  - MySQL (NDB) Cluster
    - Synchronous

Replication in Databases

- Replication is a feature allowing the contents of one or more servers (called masters) to be mirrored on one or more servers (called slaves)
- **Scalability:** By having one or more slave servers, reads can be spread over multiple servers, reducing the load on the master.
  - The most common scenario for a high-read, low-write environment is to have one master, where all the writes occur, replicating to multiple slaves, which handle most of the reads.
- **Backup assistance:** Backups can more easily be run if a server is not actively changing the data.
  - A common scenario is to replicate the data to slave, which is then disconnected from the master with the data in a stable state. Backup is then performed from this server.



Replication and Binary Log

- The main mechanism used in replication is the binary log.
  - All updates to the database (data manipulation and data definition) are written into the binary log as binlog events.
  - The binary log contains a record of all changes to the databases, both data and structure, as well as how long each statement took to execute
    - It consists of a set of binary log files and an index
  - This means that statements such as CREATE, ALTER, INSERT, UPDATE and DELETE will be logged, but statements that have no effect on the data, such as SELECT and SHOW, will not be logged
- Slaves read the binary log from each master in order to access the data to replicate.

Database Replication

- A relay log is created on the slave server, using the same format as the binary log, and this is used to perform the replication
  - Old relay log files are removed when no longer needed
- A slave server keeps track of the position in the master's binlog of the last event applied on the slave
- This allows the slave server to re-connect and resume from where it left off after replication has been temporarily stopped
- It also allows a slave to disconnect, be cloned and then have the new slave resume replication from the same master
- There will be a measurable delay between the master and the replica. The data on the replica eventually becomes consistent with the data on the master
  - Use this feature for workloads that can accommodate this delay

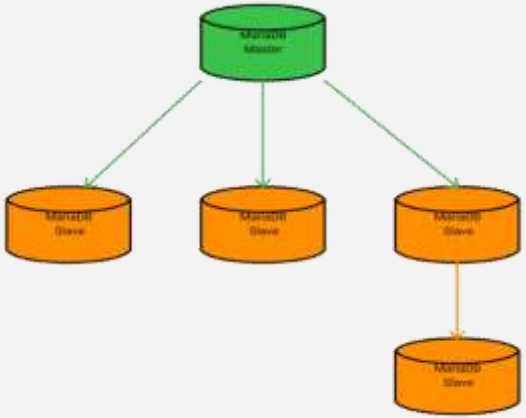
Replication Steps

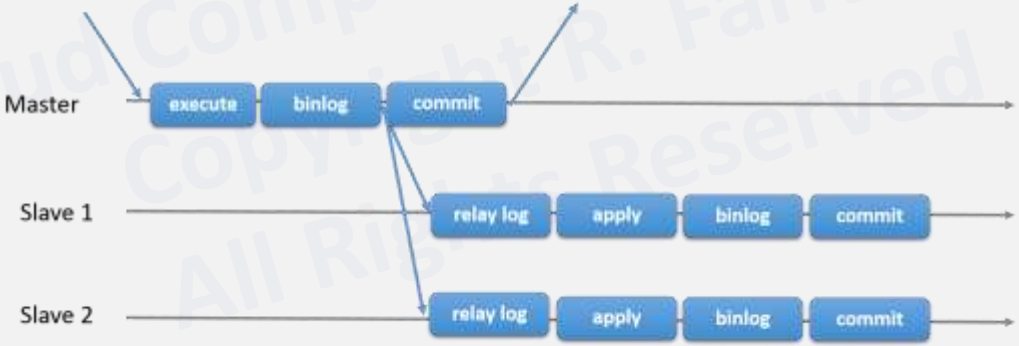
1. Replication events are read from the master by the IO thread and queued in the relay log
2. Replication events are fetched one at a time by the SQL thread from the relay log
3. Each event is applied on the slave to replicate all changes done on the master

- Replication is essentially asynchronous
  - The third step can optionally be performed by a pool of separate replication worker threads
    - **In-order** executes transactions in parallel, but orders the commit step of the transactions to happen in the exact same order as on the master
      - Transactions are only executed in parallel to the extent that this can be automatically verified
    - **Out-of-order** can execute and commit transactions in parallel
      - The application must be tolerant to seeing updates occur in different
      - Only when explicitly enabled by the application

Asynchronous Replication

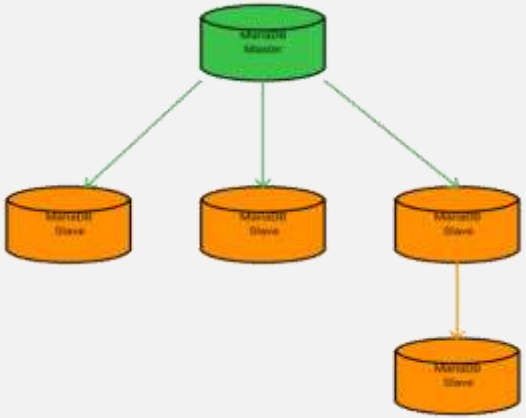
- A.k.a Standard replication
- Provides infinite read scale out
  - Most websites fit into this category, where users are browsing the website, reading articles, posts, or viewing products.
  - Updates only occur during session management, or when making a purchase or adding a comment/message to a forum.
- Provides high-availability by upgrading slave to master
- slaves read-only to ensure that no one accidentally updates them
- Eventual Consistency

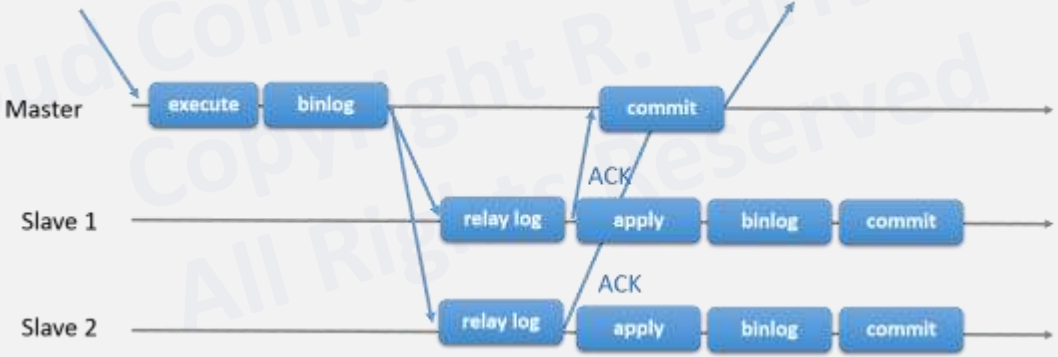




Semi-synchronous Replication

- Semi Synchronous Replication
- Better than eventual consistency
- The master waits for at least one ACK
  - Slower commits
- If no ACK received and timeout, master reverts to asynchronous
  - When at least one ACK is finally received, master goes back to semi-synchronous



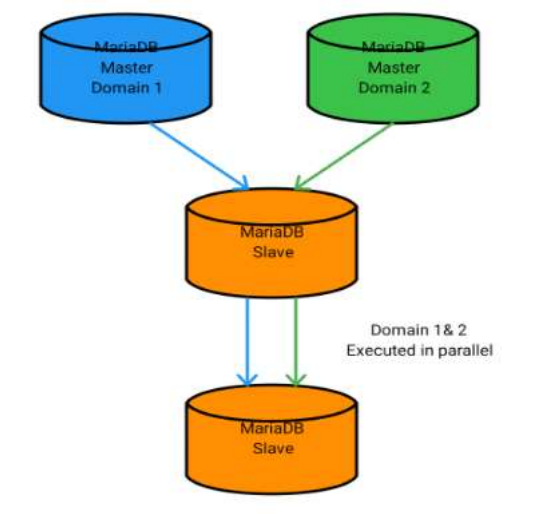


GTIDs-based Replication

- MySQL newer method based on global transaction identifiers (GTIDs)
- Transactional and therefore does not require working with log files or positions within these files
  - May simplify common replication tasks
- Replication using GTIDs guarantees consistency between master and slave as long as all transactions committed on the master have also been applied on the slave

Multi-Source Replication

- Multi-source replication means that one server has many masters from which it replicates
- Allows you to combine data from different sources
- Different domains executed independently in parallel on all slaves

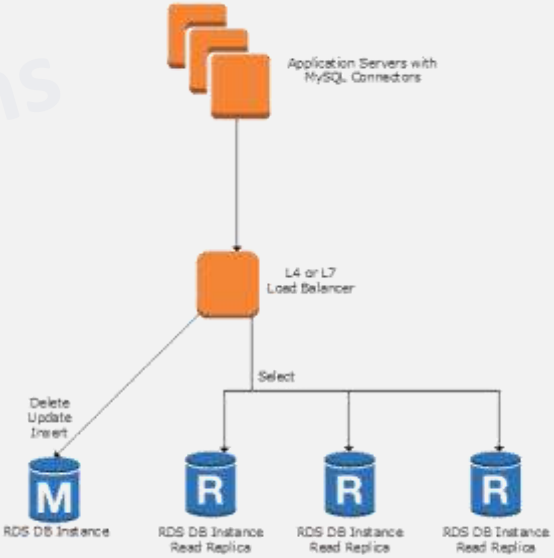


AWS RDS Horizontal Scaling

- Scaling beyond the compute or I/O capacity of a single DB instance for read-heavy database workloads
- Amazon RDS uses the MariaDB, MySQL, Oracle, PostgreSQL, and Microsoft SQL Server DB engines' built-in replication functionality to create a special type of DB instance called a read replica from a source DB instance
  - Up to five read replicas from one DB instance for MariaDB, MySQL
    - Similar limit of 5 replicas in Azure
    - Aurora allows 15 read replicas
  - specify an existing DB instance as the source
  - Amazon RDS takes a snapshot of the source instance and creates a read-only instance from the snapshot
  - Amazon RDS then uses the asynchronous replication method for the DB engine to update the read replica whenever there is a change to the source DB instance
- Updates to the source DB instance are **asynchronously** copied to the read replica
- If the read replica resides in a different AWS Region than its source DB instance, Amazon RDS sets up a secure communications channel between the source and the read replica
  - Amazon RDS establishes any AWS security configurations needed to enable the secure channel, such as adding security group entries
- Replicas can be “promoted” to full databases
  - They will reboot first

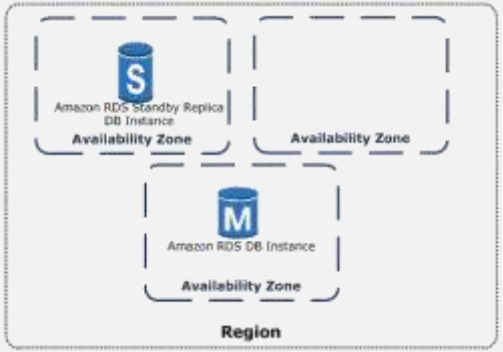
Load Balancing between RDS replicas

- Application-level load balancing
  - Different DNS record-sets using Route 53
- MySQL Connectors
  - If using the native MySQL driver, there are MySQL Connectors that allow read/write splitting and read-only endpoint load balancing without a major change in the application
- ELB does not support multiple RDS instances
- Level 4 proxy solutions
  - HAProxy: configure HAProxy to listen on one port for read queries and another port for write queries
- Level 7 proxy solutions
  - more sophisticated capability of understanding how to properly perform the read/write splits on multi-statements than a MySQL Connector does
  - This solution handles the scaling issues in a distributed database environment, so you don't have to handle scaling on the application layer, resulting in little or no change to the application itself
  - Several open-source solutions (such as MaxScale, ProxySQL, and MySQL Proxy) and also commercial solutions, some of which can be found in the AWS Marketplace



High Availability (Multi-AZ) for Amazon RDS

- Amazon RDS uses several different technologies to provide failover support
  - Multi-AZ deployments for MariaDB, MySQL, Oracle, and PostgreSQL DB instances use Amazon's failover technology
  - SQL Server DB instances use SQL Server Database Mirroring (DBM) or Always On Availability Groups (AGs)
- The high-availability feature is not a scaling solution for read-only scenarios

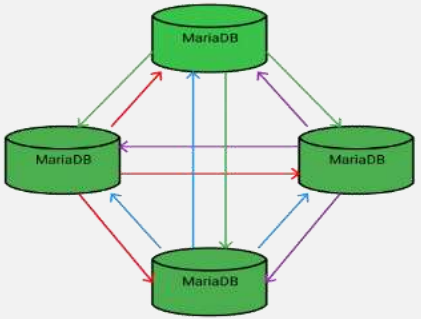


Read replicas, Multi-AZ deployments, and multi-region deployments (Amazon AWS)

Multi-AZ deploymentss	Multi-Region deployments	Read replicas
Main purpose is high availability	Main purpose is disaster recovery and local performance	Main purpose is scalability
Non-Aurora: synchronous replication; Aurora: asynchronous replication	Asynchronous replication	Asynchronous replication
Non-Aurora: only the primary instance is active; Aurora: all instances are active	All regions are accessible and can be used for reads	All read replicas are accessible and can be used for readscaling
Non-Aurora: automated backups are taken from standby; Aurora: automated backups are taken from shared storage layer	Automated backups can be taken in each region	No backups configured by default
Always span at least two Availability Zones within a single region	Each region can have a Multi-AZ deployment	Can be within an Availability Zone, Cross-AZ, or Cross-Region
Non-Aurora: database engine version upgrades happen on primary; Aurora: all instances are updated together	Non-Aurora: database engine version upgrade is independent in each region; Aurora: all instances are updated together	Non-Aurora: database engine version upgrade is independent from source instance; Aurora: all instances are updated together
Automatic failover to standby (non-Aurora) or read replica (Aurora) when a problem is detected	Aurora allows promotion of a secondary region to be the master	Can be manually promoted to a standalone database instance (non-Aurora) or to be the primary instance (Aurora)

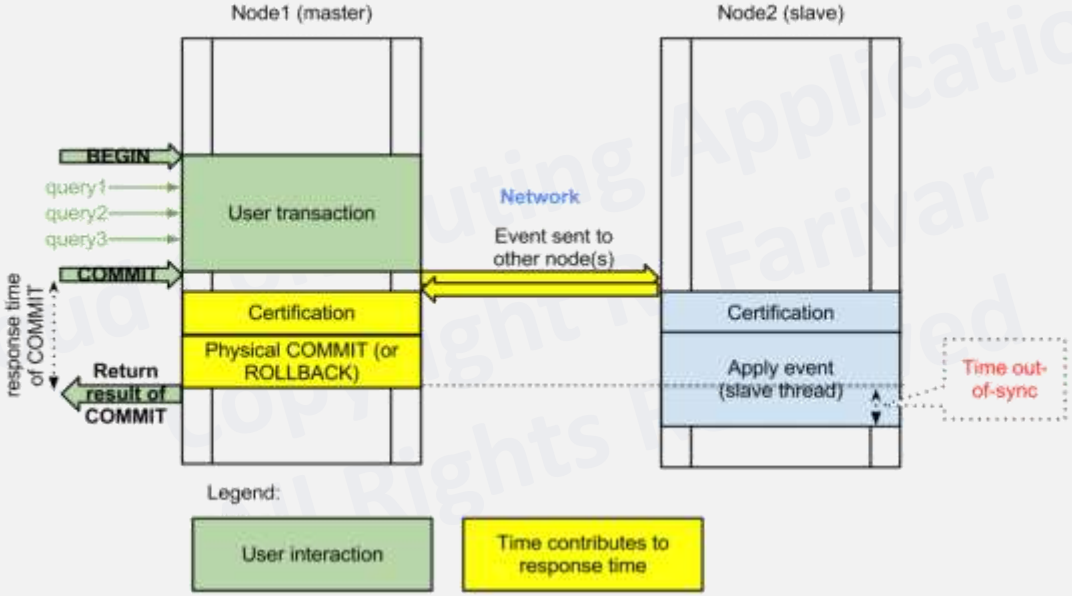
Multi-Master Cluster

- Galera (MySQL, MariaDB)
- Synchronous replication
- Active-active multi-master topology
- Read and write to any cluster node
- Automatic membership control, failed nodes drop from the cluster
- Automatic node joining
- True parallel replication, on row level
- Direct client connections, native MariaDB look & feel



Galera Transaction Commit Flow

- Certification Based Replication
- Virtually Synchronous



MySQL NDB Cluster

- Network DataBase Engine
  - Replaces InnoDB
  - Separation of Compute and Data
    - SQL Nodes
    - Data Nodes
  - Shared Nothing Archiecture