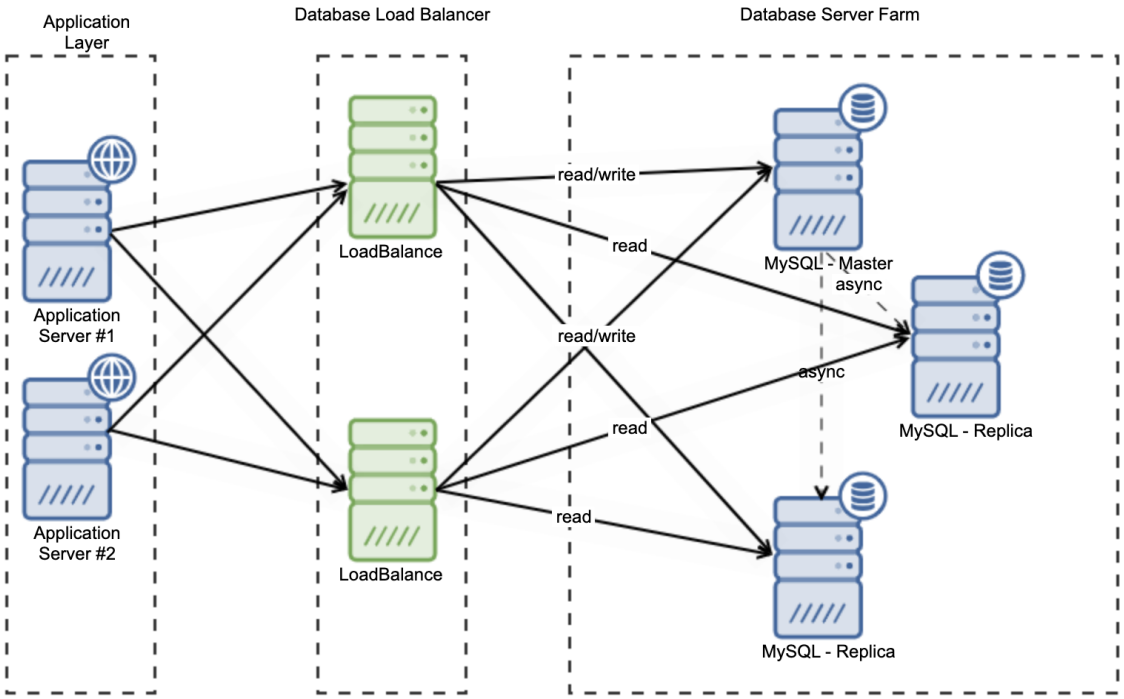
Database Load Balancing

Overview

Database load balancers sit between applications and database servers. They accept traffic from applications, and distribute the traffic to databases as shown below:



There are two categories of database load balancers: Layer 4 and Layer 7. They share the common (static and dynamic) load balancing algorithms with other load balancers.

With database load balancers, the **number** you require depends on how much **query** you handle and how much uptime **availability** you want. Prevent load balancers from being a Single Point of Failure (SPOF).

Database load balancers can be placed at any software layer. There are embedded load balancers or external load balancers.

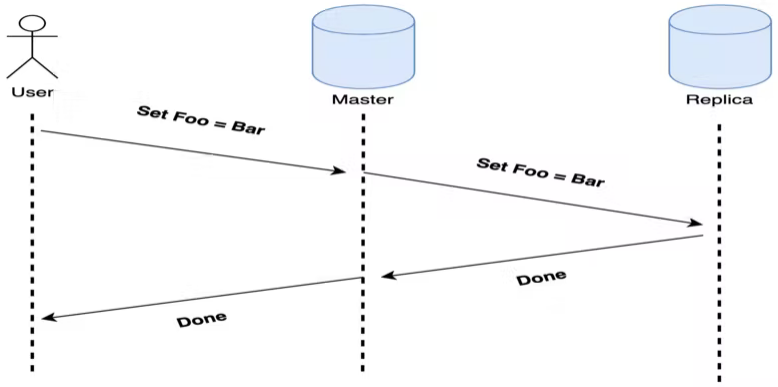
Database's Cooperation

To achieve database load balancing requires Database Server Farm's cooperation, such as replication and log shipping.

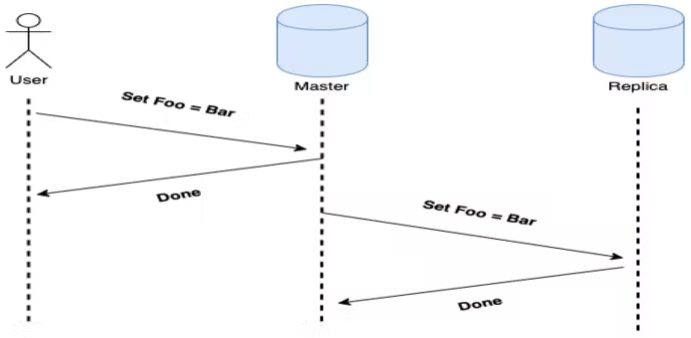
Replication

According to when the **primary** database give **confirmation** to the **application**'s update request, replication is categorized as

- Synchronous



- Asynchronous



- Semi-sync
  Some replicas replicate synchronously and others replicate asynchronously.

Log Shipping

Log shipping consists of three operations:

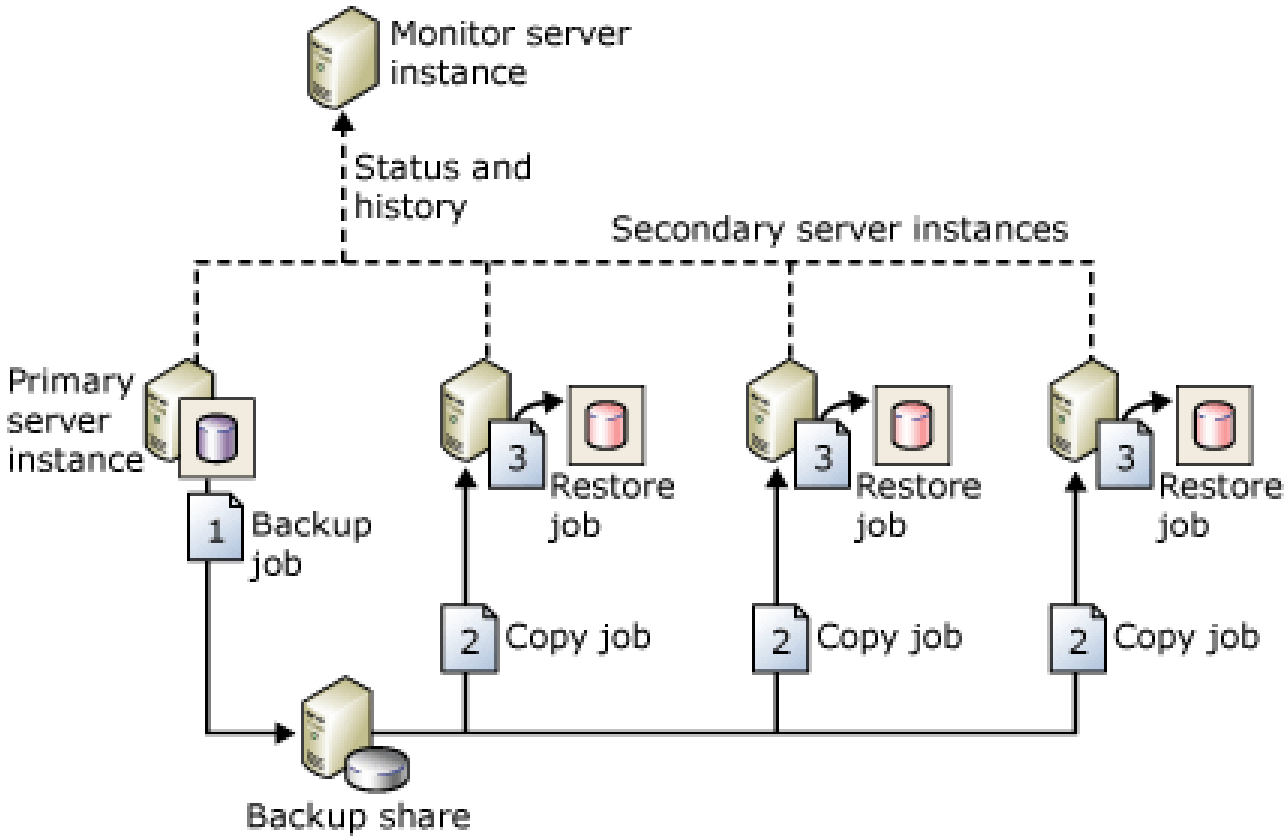
1. **Back up** the transaction log at the primary server instance.

2. **Copy** the transaction log file **to** the **secondary** server instance.

3. **Restore** the log backup on the secondary server instance.

The log can be shipped to **multiple secondary** server instances. In such cases, operations 2 and 3 are duplicated for each secondary server instance.

The following figure shows a log shipping configuration with the primary server instance, three secondary server instances, and a monitor server instance, as follows:



1. The primary server instance runs the **backup job** to back up the transaction log on the primary database. This server instance then places the log backup into a primary **log-backup file**, which it sends to the **backup folder**. In this figure, the backup folder is in a shared directory - the backup share.

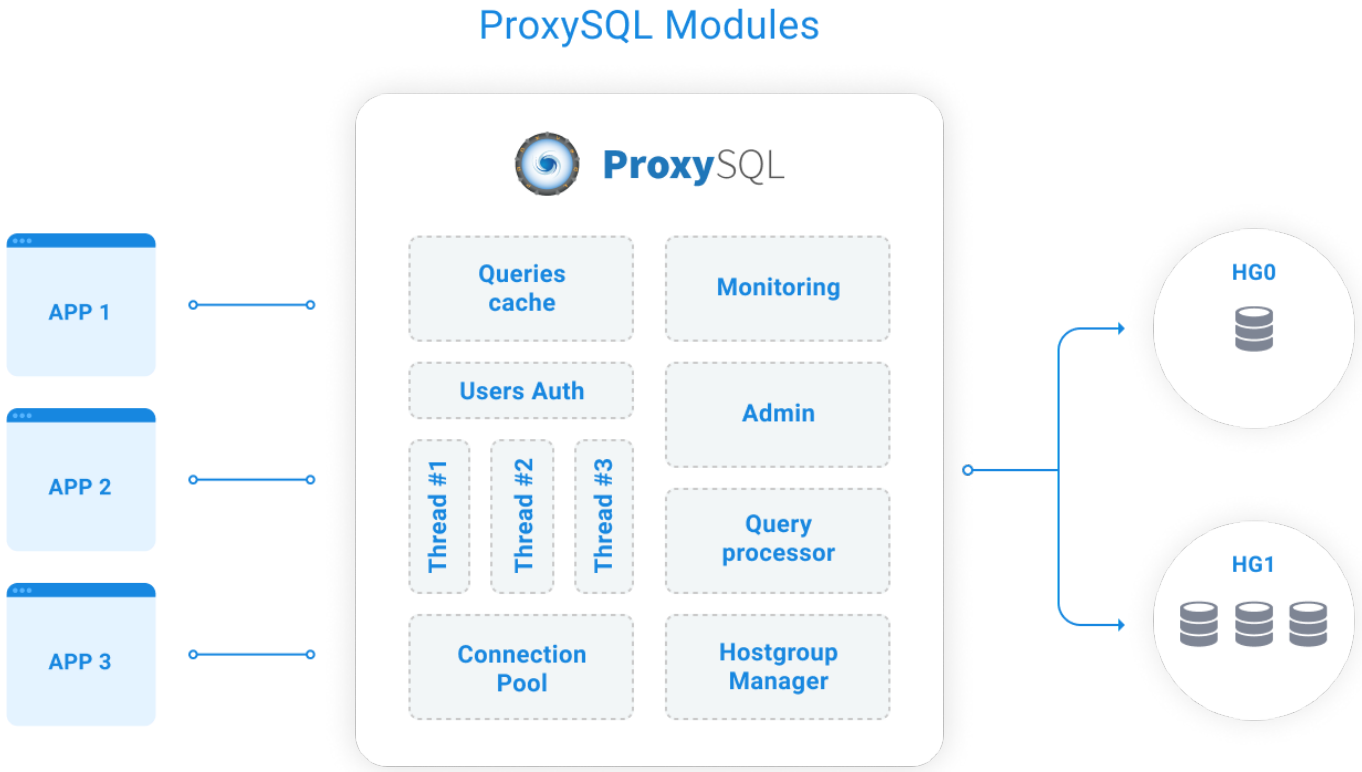
2. Each of the three secondary server instances runs its own **copy job** to copy the primary log-backup file to its own local destination folder.

3. Each secondary server instance runs its own **restore** job to restore the log backup from the local destination folder onto the local secondary database.

The primary and secondary server instances send their own history and status to the **monitor** server instance.

Case Study: ProxySQL

ProxySQL is a high performance (My)SQL Proxy. It resides on Layer 7 of the OSI model, which gives more features from a database perspective.



- Query processor/rules engine

The rules engine matches incoming traffic, and defines whether to cache a query, or block, re-route, re-write or mirror it onto a hostgroup target.

- User authentication
  User credentials for the underlying databases are hashed and stored in the proxy.

- Hostgroup manager
  This manages the groups of **servers** to send traffic to, and tracks their **state**.

- Connection pool
  This manages the connections to the backend databases. A pool of connections is established towards the backends, and that is shared/**reused** by all applications.

- Monitoring
  This monitors the backends and collects **metrics**. For example, it monitors for unresponsive hosts or replication lag and shuns them as necessary.

- Query caching
  All we need to cache the resultset is to define the query rule(**matching criteria**) and the **time-to-live**.

  The only way to **invalidate** entries from the ProxySQL query cache is through a **time-to-live** in milliseconds. This is in contrast to MySQL query cache, where the query cache gets **invalidated** each time a table gets **updated**. Note that every query that is cached may return **stale** data, but it scales well.

*References

https://severalnines.com/database-blog/how-does-database-load-balancer-work
https://docs.microsoft.com/en-us/sql/database-engine/log-shipping/about-log-shipping-sql-server?view=sql-server-ver15
https://proxysql.com/
https://severalnines.com/resources/tutorials/proxysql-tutorial-mysql-mariadb
https://www.percona.com/blog/2018/02/07/proxysql-query-cache/