Data storage in distributed systems - part I

Tomasz Gintowt
RTB House

Community

What will this lecture be about?

In this part of the course we will consider the topic of data storage technologies for data intensive distributed systems.

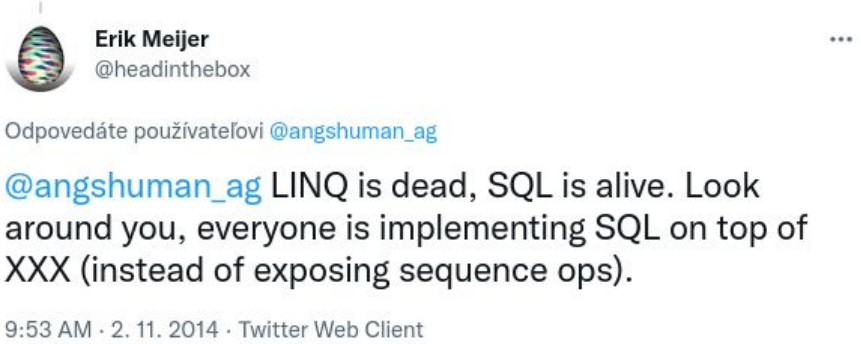
We will cover the topics of:

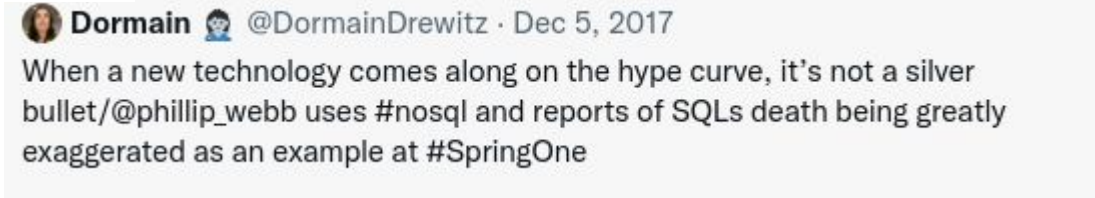
- The aspects of running a traditional RDBMS in a distributed system.
- The common problems with relations databases in distributed environment.

A little bit history

RDBMS **NoSQL** **NewSQL** **Distributed SQL**

A little bit of history





Distributed Storage

Relational Database Management Systems

RDBMS

Why still use RDBMS?

- Well established flexible data model.
- Stable and battle-proven.
- Powerful Structured Query Language.
- Full support for ACID transactions.
- Swiss army knife - supports most corner cases.
- Well documented.

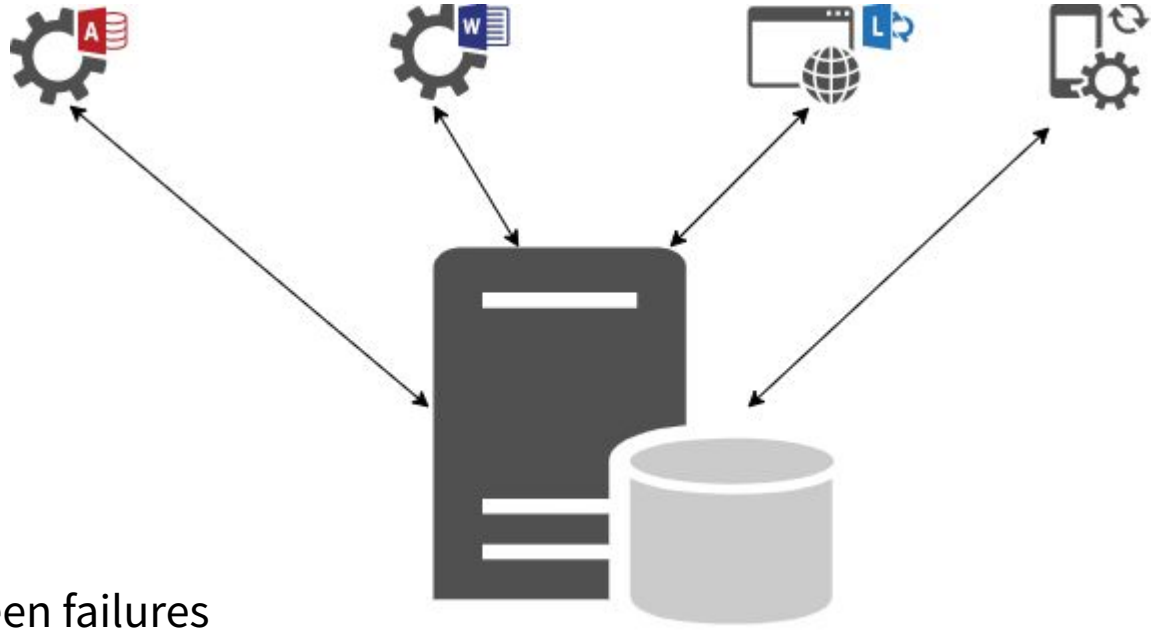
Transactions

The main advantage of using single node relational databases is the fact that they fully support the concept of a transaction.

- **A**tomicity - operations within transaction either completely succeed or abort.
- **C**onsistency - operations within transaction do not violate invariants or data integrity.
- **I**solation - different transactions do not interfere with each other.
- **D**urability - the data has been written to disk (including WAL).

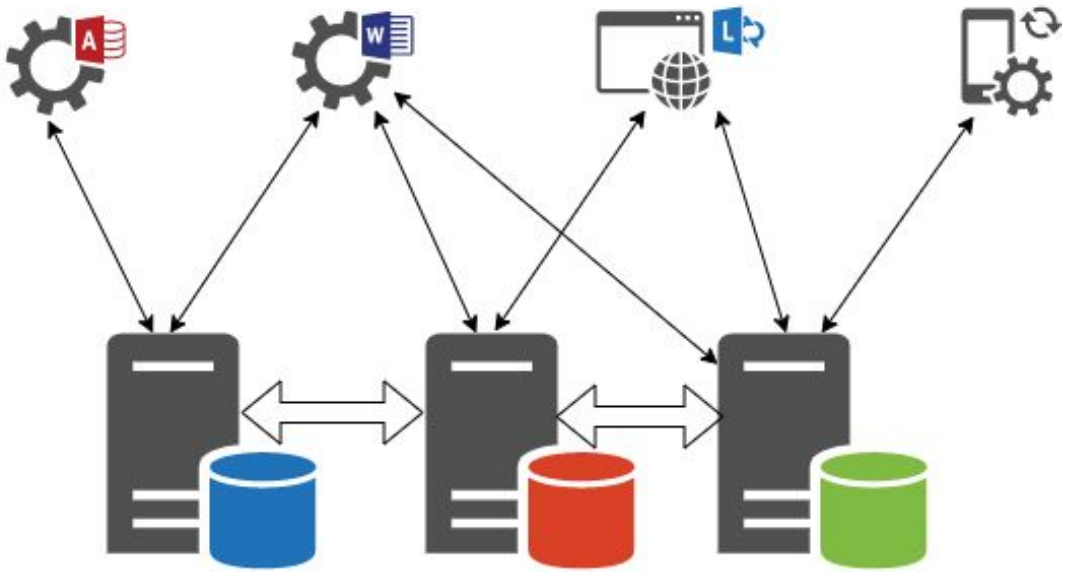
For a distributed system the isolation part is the most interesting one.

RDBMS as a silo



- Single server
- Very high cost
- 128 CPU, 512 G RAM
- Long mean time between failures
- Long mean time to repair
- SPOF - single point of failure

RDBMS as a cluster

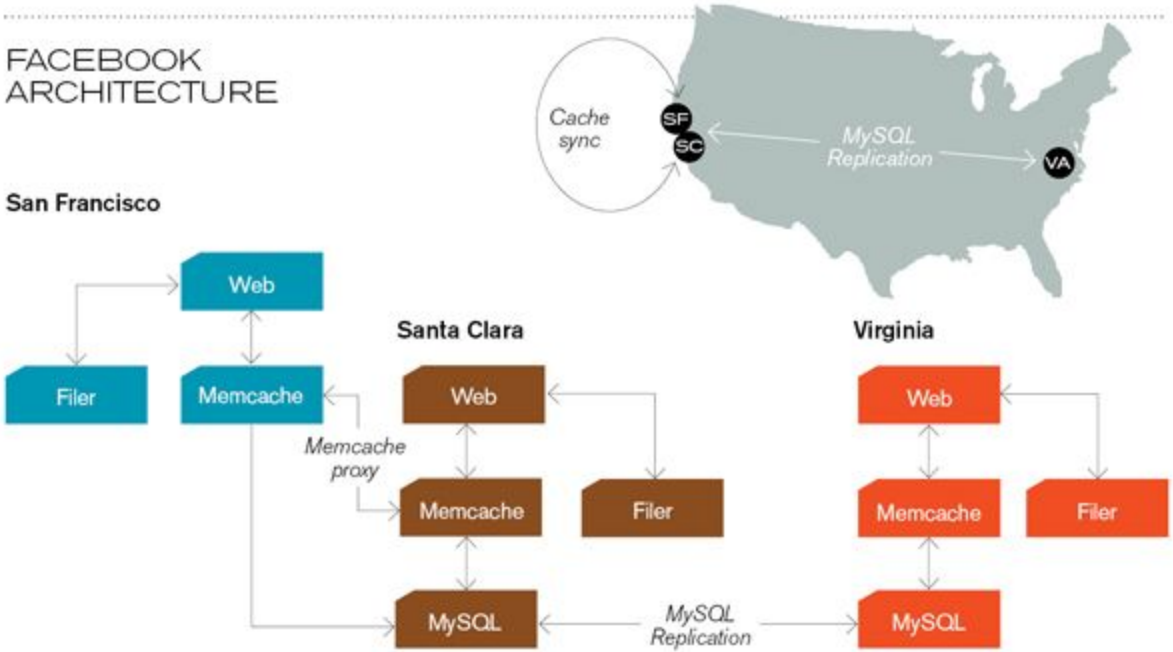


- Multiple servers
- Reasonable cost
- 3 x (64 CPU, 128 G RAM)
- Short mean time between failures
- Short mean time to repair
- High availability

RDBMS as a cluster

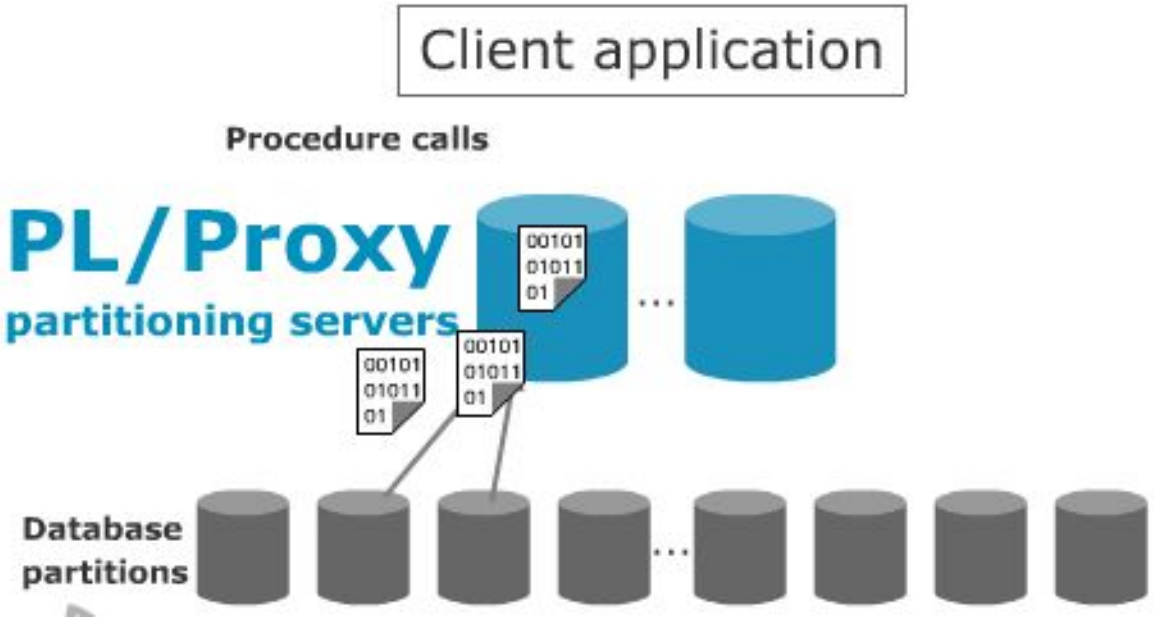
- Multiple regions
- Master-Slave replication
- Master-Master replication
- Low latency
- Millions of request per second
- High availability

Facebook database - MySQL



https://www.technologyreview.com/2008/06/23/219803/how-facebook-works/

Skype database - PostgreSQL

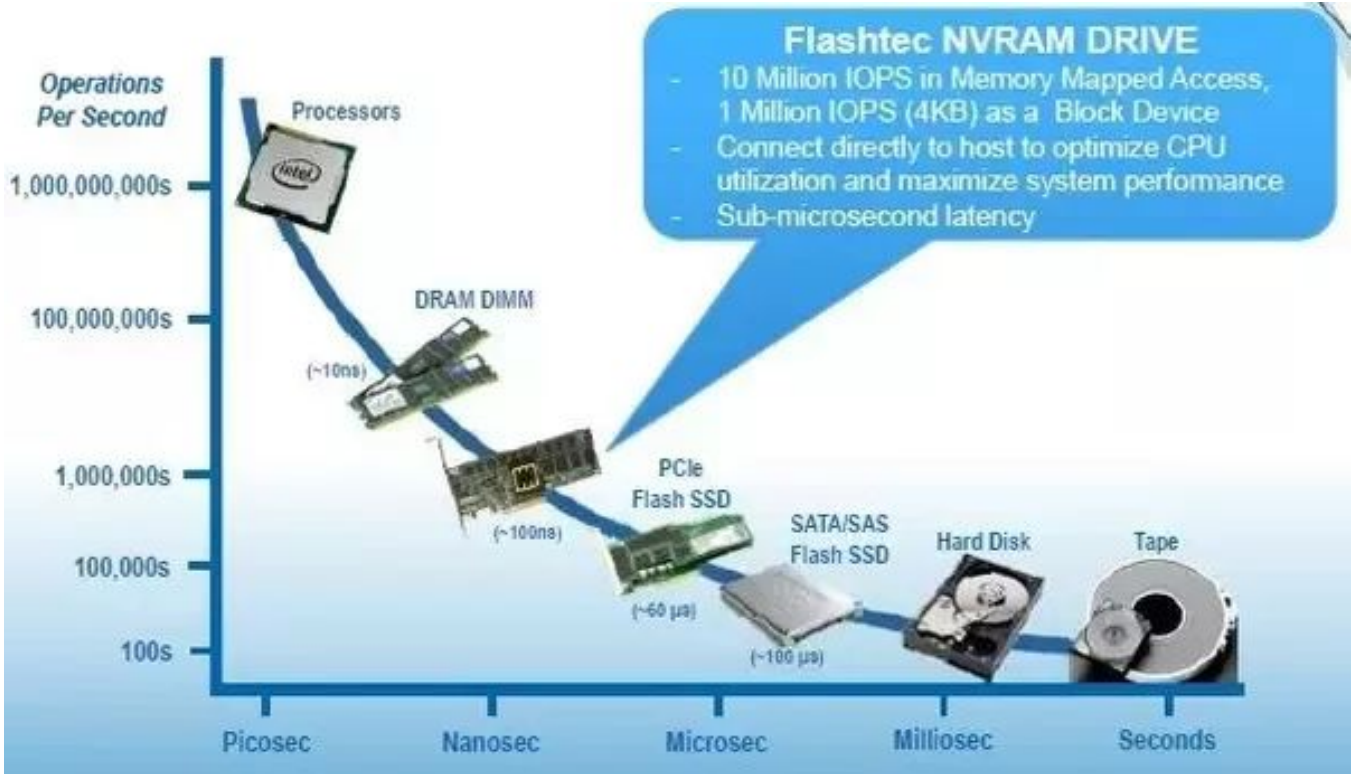


https://pankajconnect.medium.com/architecture-of-postgresql-databases-to-scale-to-1-billion-users-3f47853f48eb

Distributed Storage

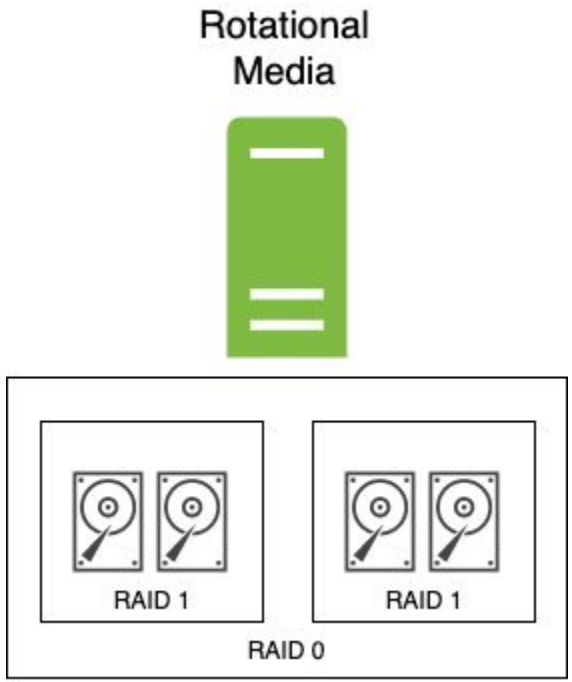
Hardware

How fast is your disk ?



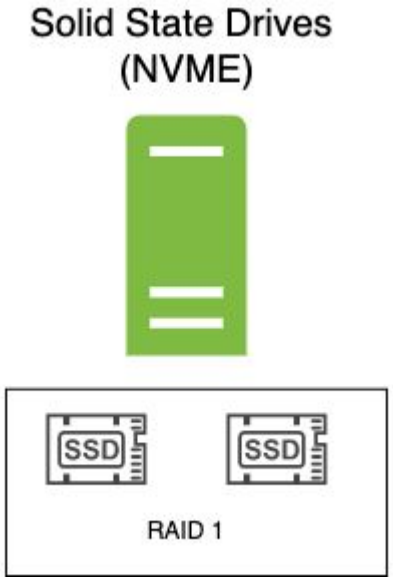
https://www.quora.com/Is-the-speed-of-SSD-and-RAM-the-same

Hardware Architecture for RDBMS



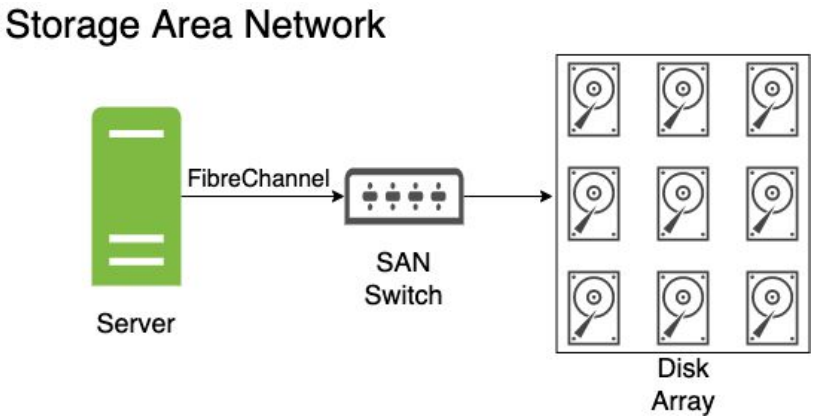
- With rotational media in DAS (Direct Access Storage) configuration the latency of the storage device is the performance bottleneck in RDBMS applications. To mitigate use the RAID 10 (1 + 0) configuration.
- In this configuration the striped data is additionally mirrored for better fault tolerance.
- RAID capabilities can be either provided by system software or dedicated hardware controllers.

Hardware Architecture for RDBMS



- With solid state media in DAS (Direct Access Storage) configuration, the devices connected directly to the PCI Express bus are no longer the performance bottleneck.
- In this scenario the usual configuration is to use RAID 1 for drive redundancy.

Hardware Architecture for RDBMS



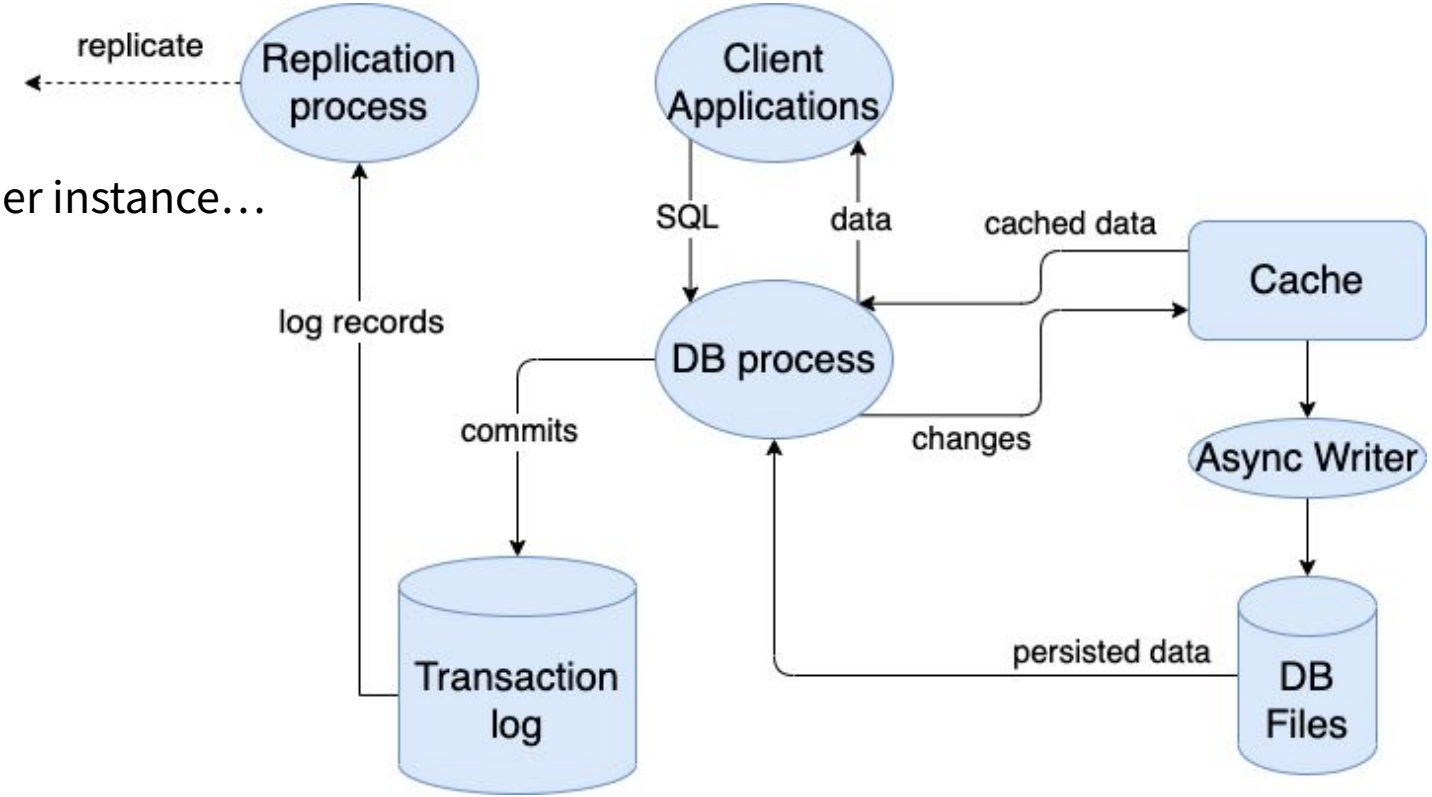
- The data storage for database can be also accessed via SAN.
- Not as efficient as directly accessed storage.
- Many vendors offer custom proprietary solutions.
- Offers more flexibility regarding storage management.

Distributed Storage

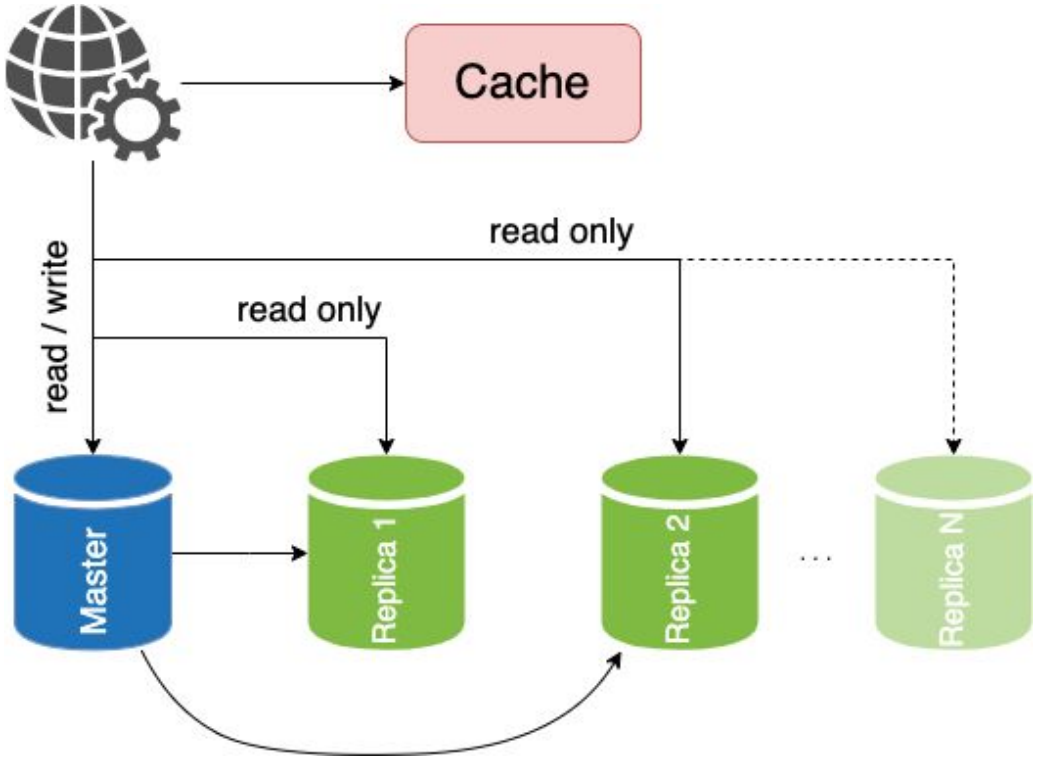
Replication

Typical RDBMS Process

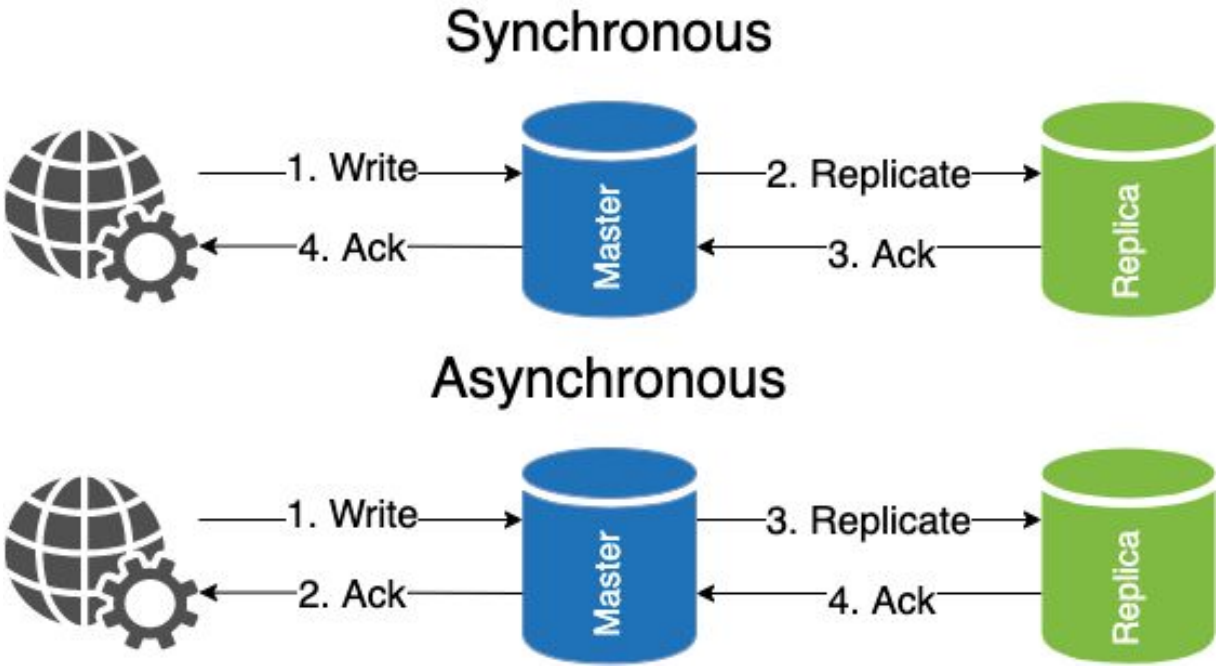
To another instance…



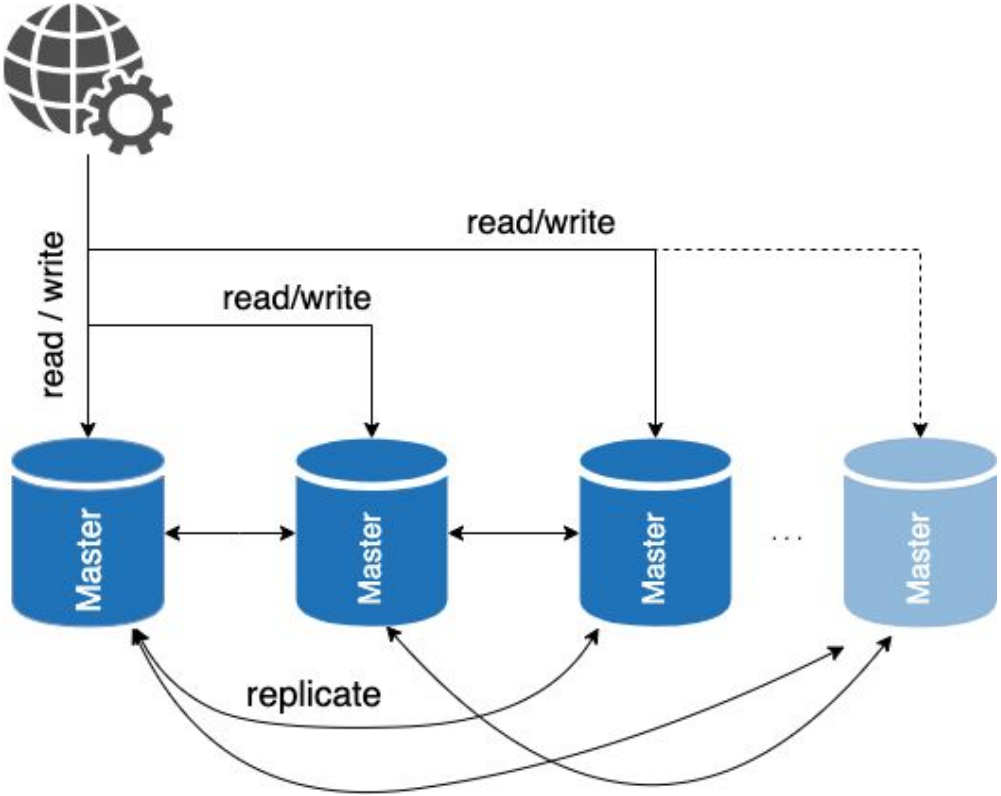
Typical Architecture Involving RDBMS



RDBMS Replication

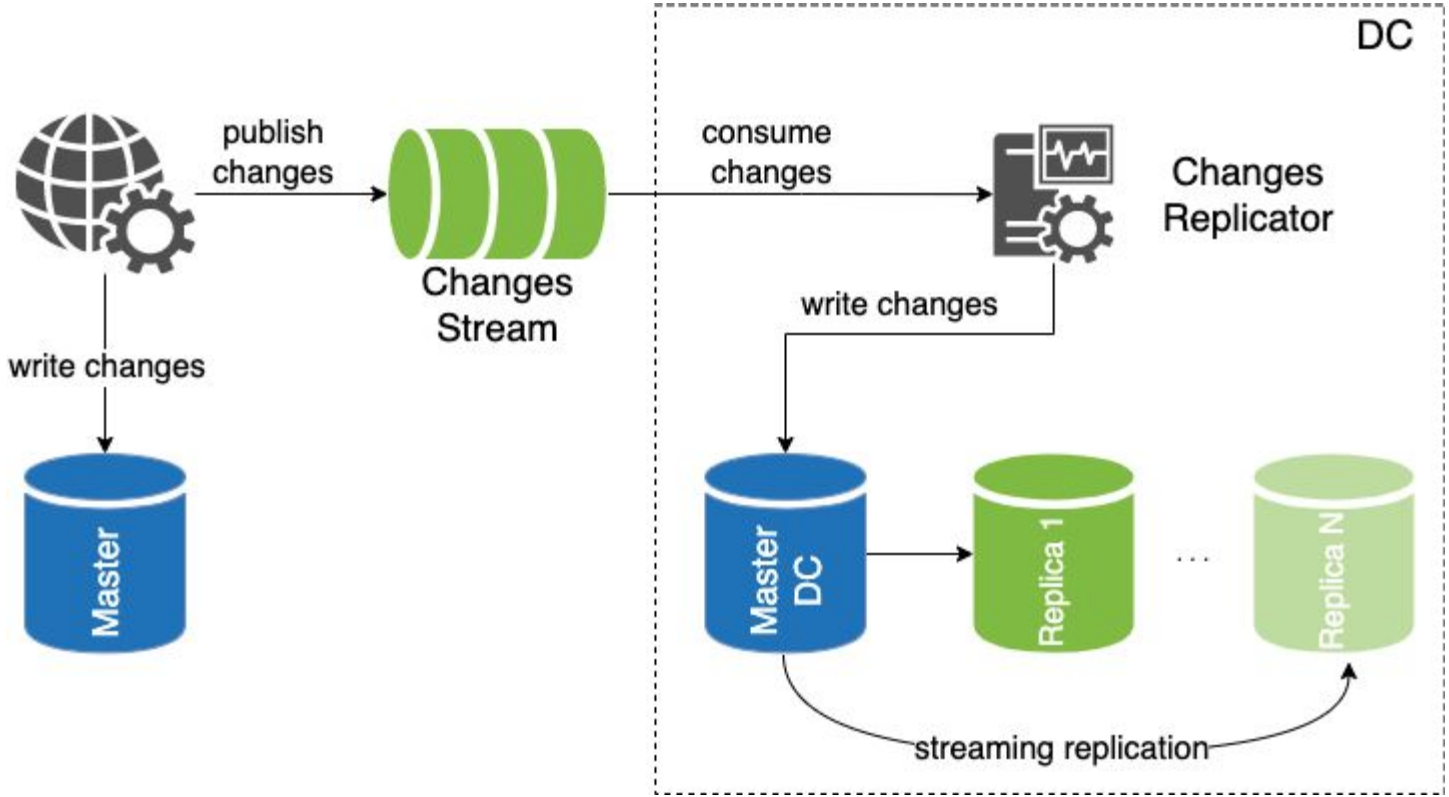


Multi-Master Replication



- Adds complexity.
- Unsupported natively by RDBMSes.
- Inevitable conflicts with non-trivial methods of resolution.

Custom Cross-DC Replication - Architecture



RDBMS Replication

An (incomplete) selection of possible methods:

- **Log Shipping** - copying completed WAL segments, better resource utilisation, higher risk of reduced durability (lost changes).
- **Log Streaming** - sending individual WAL entries over an open connection, higher resource utilisation, lesser risk of reduced durability.
- **SQL replication** - possible inconsistent results between replica nodes (ie NOW() function yields different results).
- **Logical replication -** changes are published in a defined message format and are applied by subscriber nodes.
- **File System (Block Device) Replication** - all changes to a file system are mirrored to a file system residing on another computer.

Replication Burden

Replication adds additional burden to primary node with growing number of follower nodes.

Cascading replication - follower nodes cascade changes to other followers.

Custom solution - when replicating between DC's:

- Allow only aggregates in database - denormalization with JSON columns.
- Force applications that mutate data to post aggregated changes on queue system.
- Feed "followers" on target DC's from queues (Kafka).
- Run checkers to fix consistency issues .
- The performance of replication lag consumption is scalable (more change consumers).

Distributed Storage

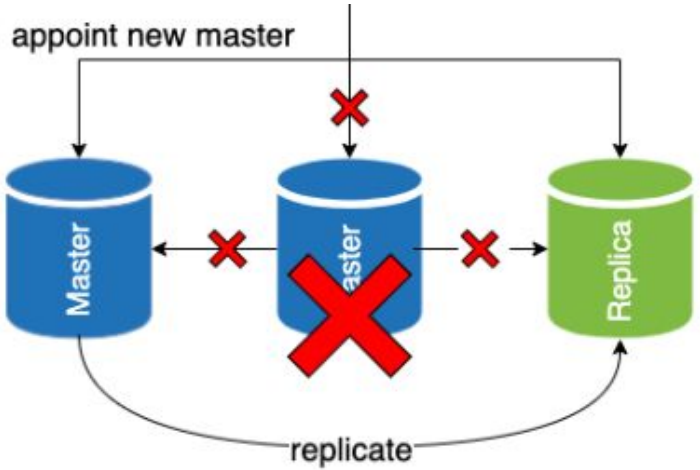
Challenges

Where is

MASTER

?

Where is Master ?



- Additional layer PROXY or load balancer with Virtual IP.
- Service Discovery with DNS.
- Dedicated tool like ProxySQL or PgPool-II.
- All DML queries should go to Master.

Connection

pooling

?

Load Balancing and Connection Pooling

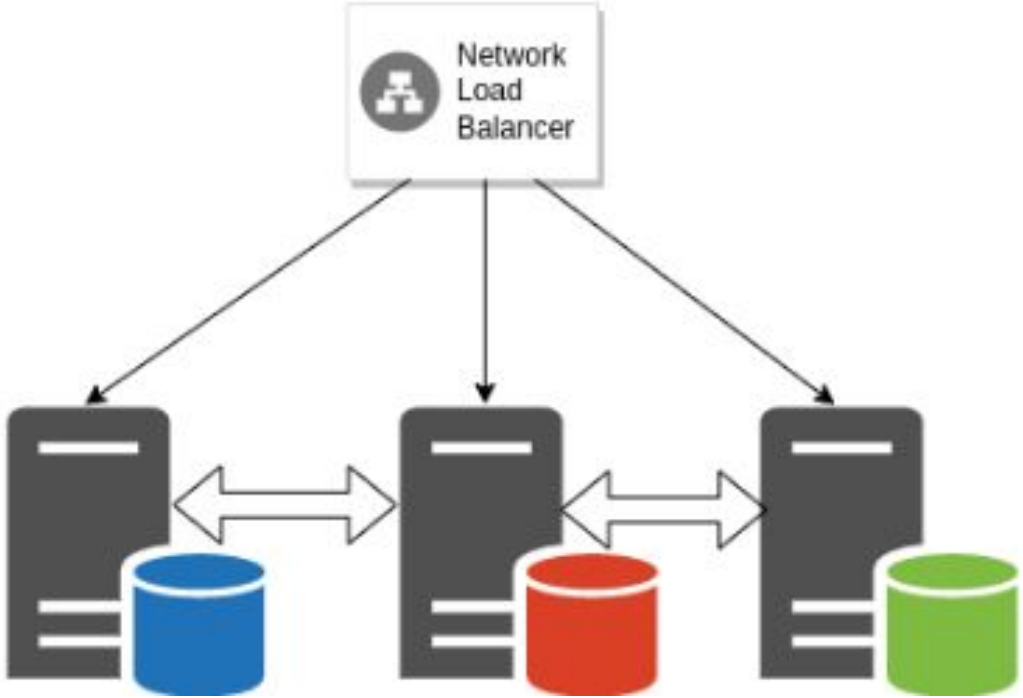
SELECT queries on RDBMS may be distributed among several replicas for load balancing reasons, whereas write queries are usually distributed to the master node.

Establishing connections to a database can be expensive so it is a good idea to set a pool of connections and reuse them.

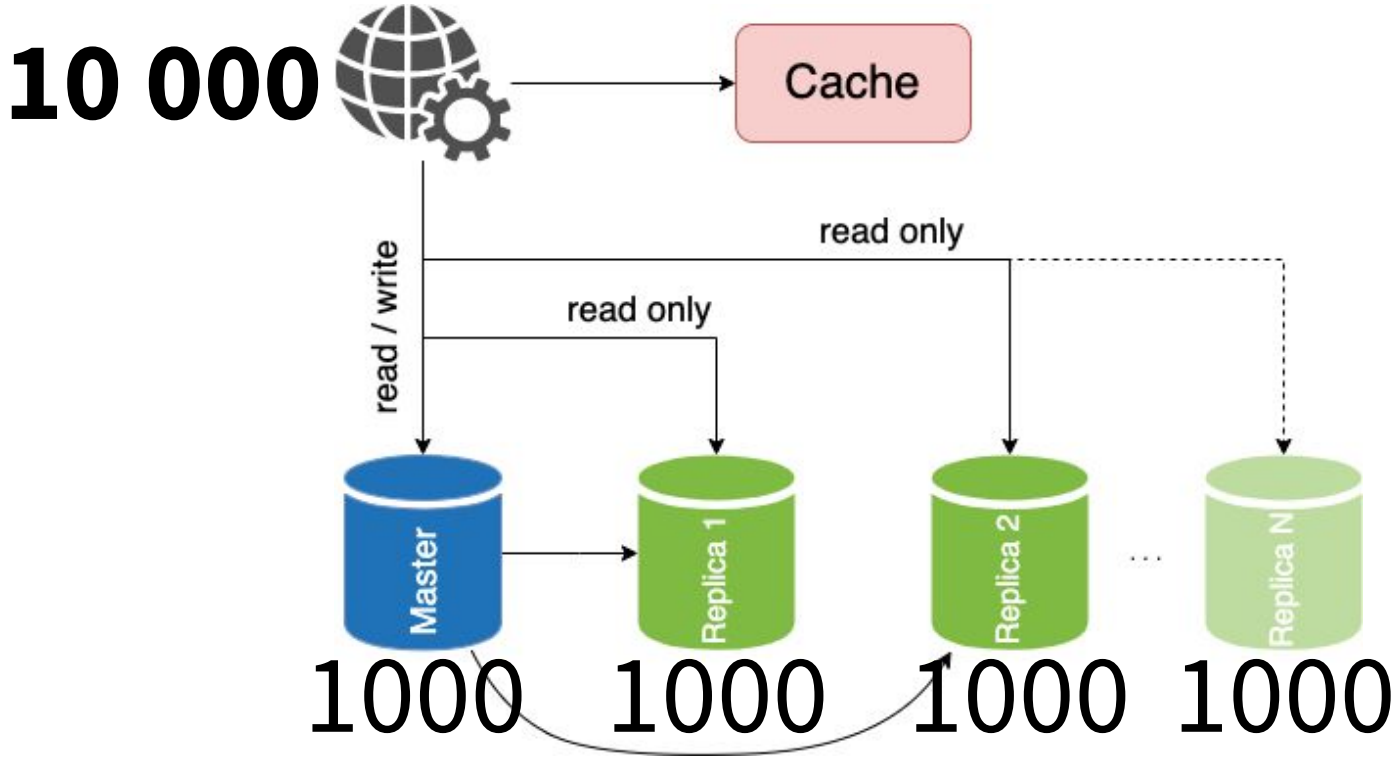
Both load balancing and connection pooling can be achieved in two ways:

- Internally by an instance of the application.
- By a centralized component.

Load Balancing



Connection Pooling



High

availability

?

RDBMS Failover

When the primary database fails and needs to be replaced by one of the replica nodes.

- **Manual** - the new master has to be promoted manually by system's operator. This will usually mean partial unavailability for a RDBMS.

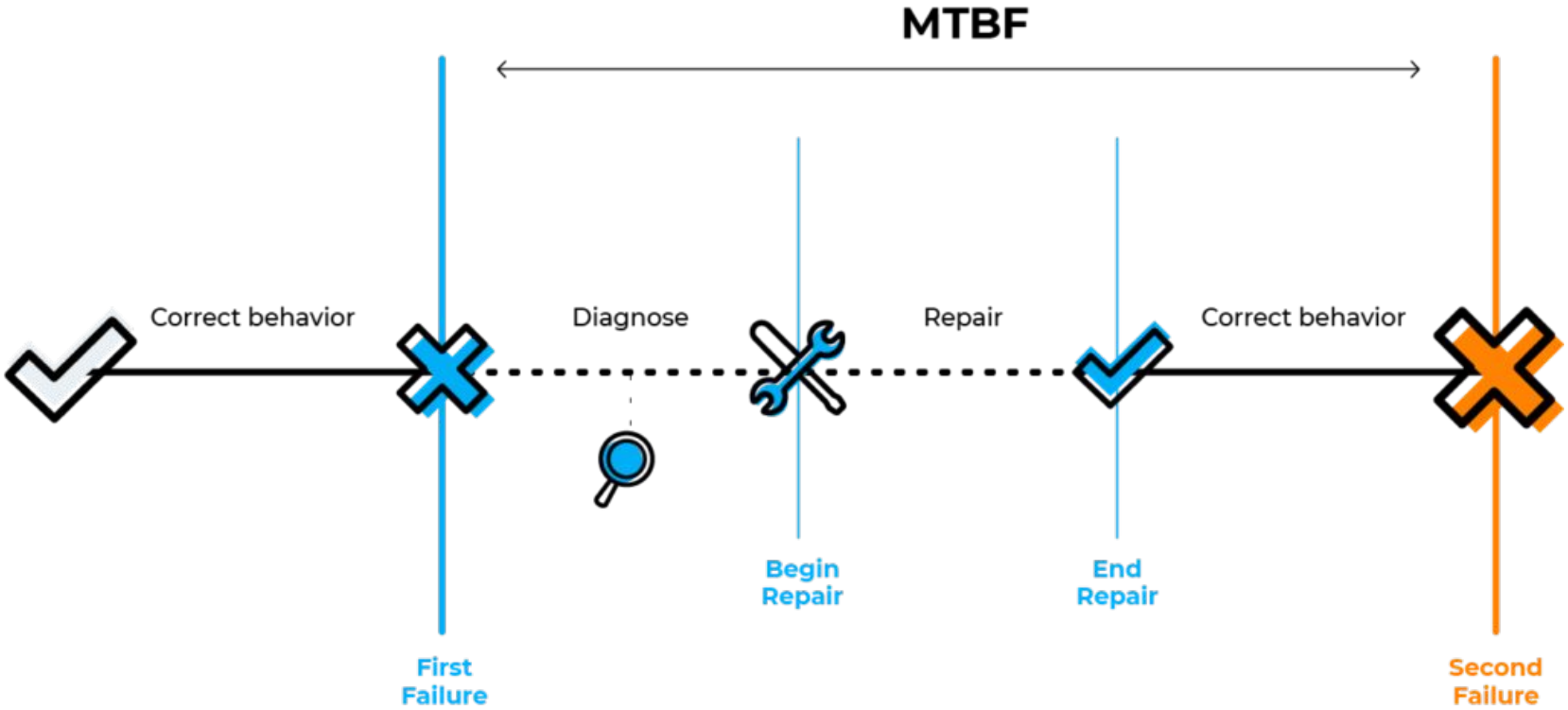
- **Automatic** - the database cluster manager will automatically promote new master from the set of available replicas.

RDBMS High Availability

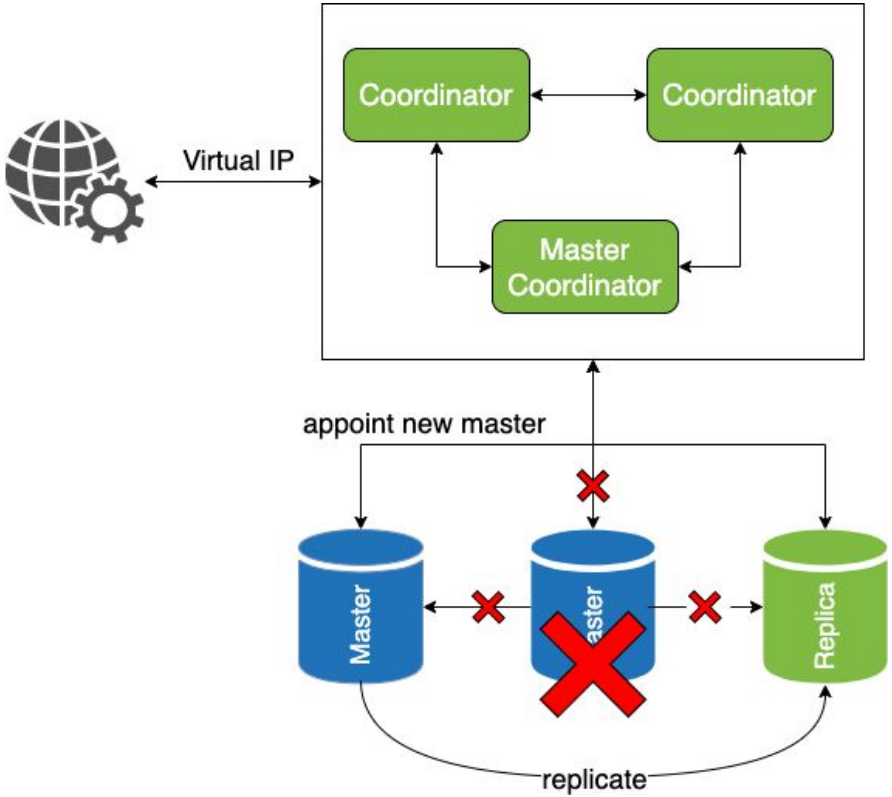
Failure Is Just Part of the Game

- **MTTR - Mean Time To Repair**
- **MTBF - Mean Time Between Failures**

RDBMS High Availability



RDBMS Failover



Scaling

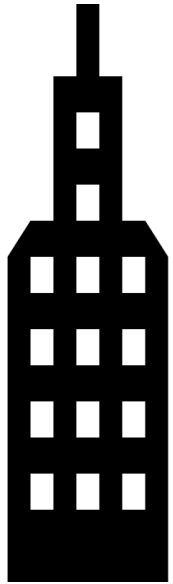
?

RDBMS Scaling

RDBMS Vertical Scaling

How to scale the RDBMS vertically with losing as little availability as possible?

- CPU, RAM - these are hot swappable elements in the high-end server equipment.
- Increasing disk capacity - in some SAN solutions this can be done with zero downtime by adding/replacing disks.
- On commodity hardware it usually requires short downtime or partial unavailability (read only mode).

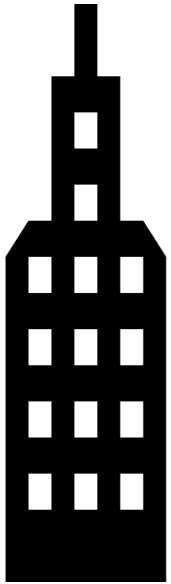


RDBMS Vertical Scaling

In case of commodity servers the exemplary strategies to scale capacity can be used:

By using Logical Volumes:

1. Add new disk to the system and expand the current logical volume and resize the filesystem online.
2. Replace the existing logical volume:
   - Add new disk and create new logical volume.
   - Synchronize the data between the old and new logical volumes.
   - Disable database for a short moment and replace the volumes.
   - Re-enable the database.

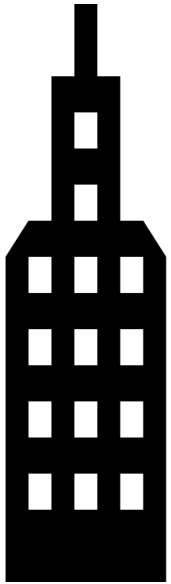


RDBMS Vertical Scaling

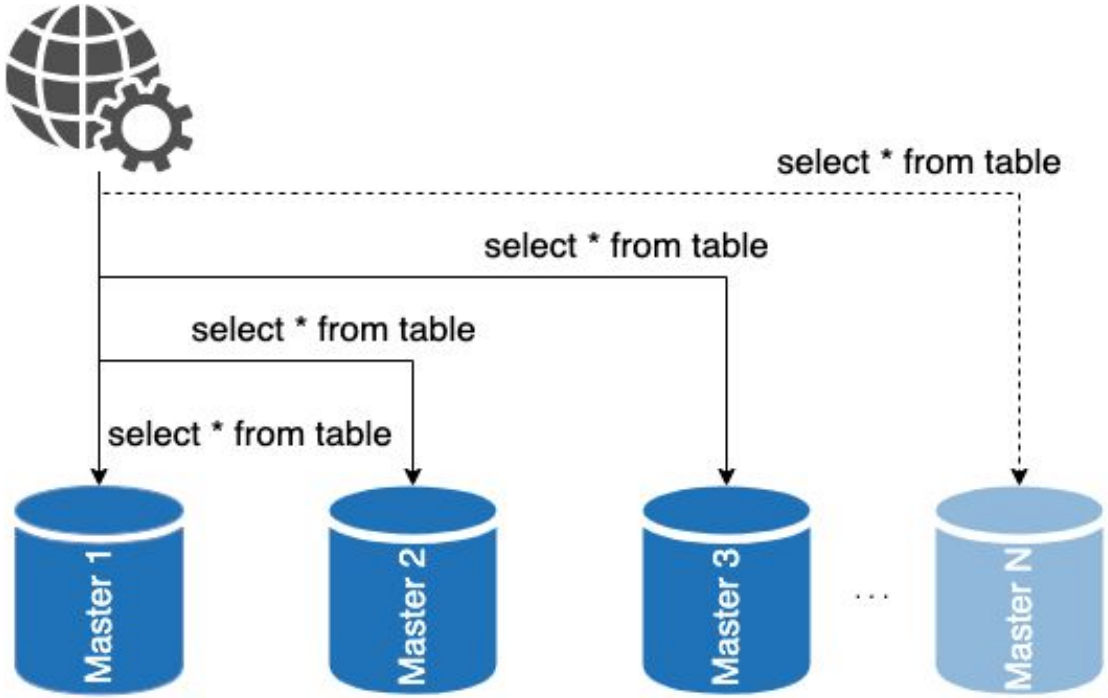
In case of commodity servers the exemplary strategies to scale capacity can be used:

By adding a new replica:

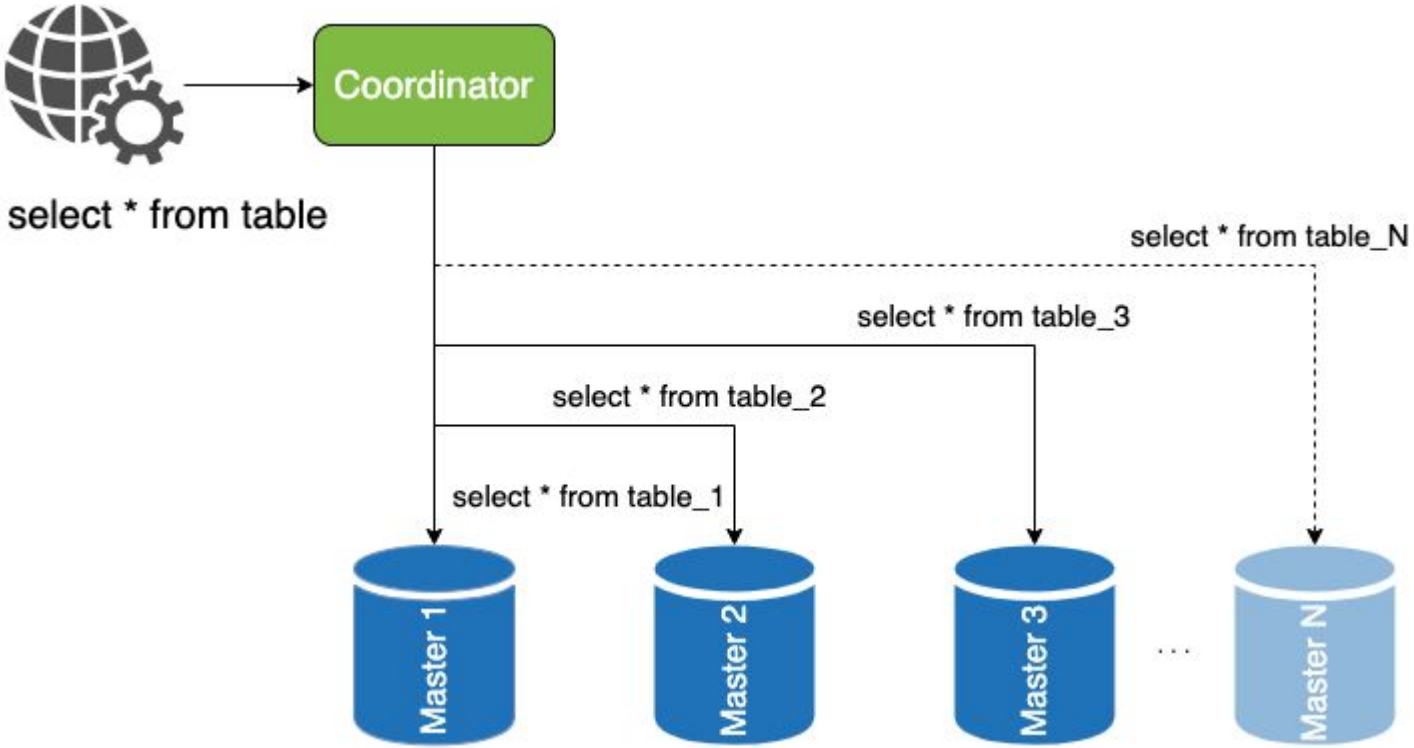
- Add new machine with desired new capacity (and other specification).
- Configure it as a replica and let it synchronize with the master.
- Wait for the replica to fully synchronize and promote it to leadership.



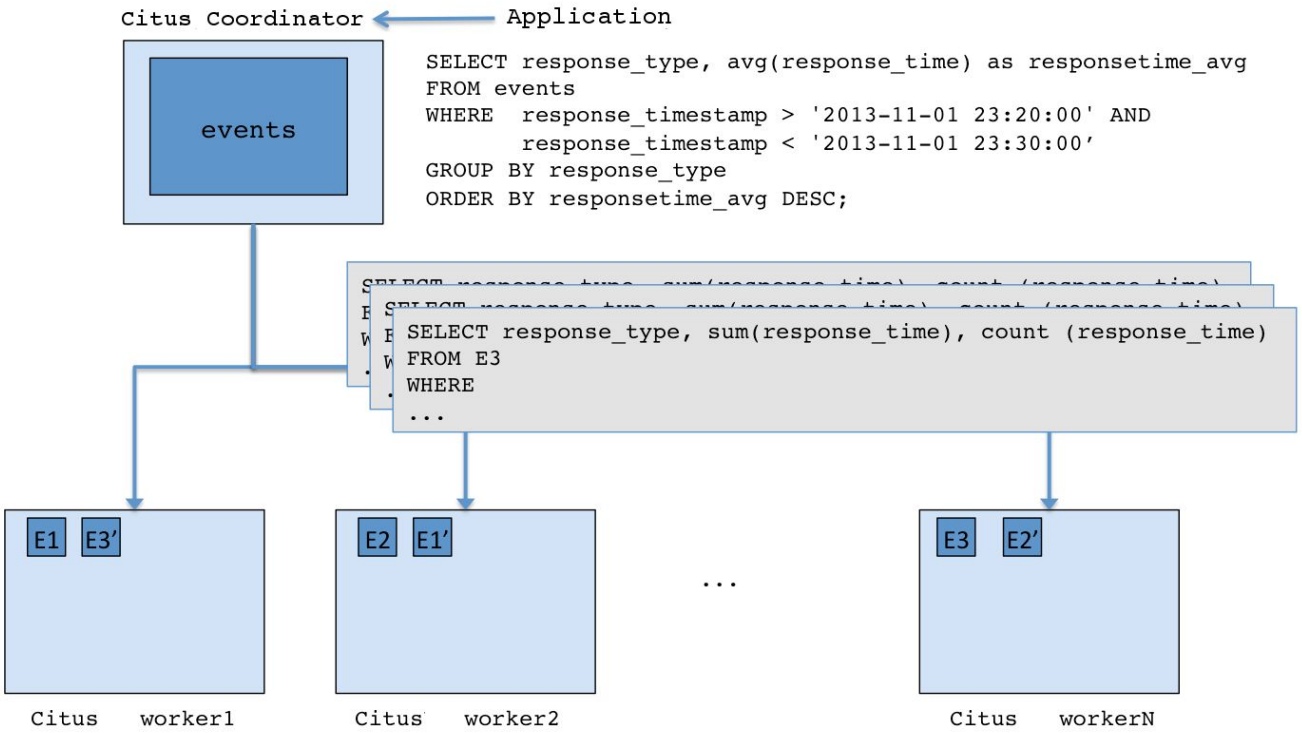
RDBMS Horizontal Scaling (Sharding)



RDBMS Horizontal Scaling (Sharding)

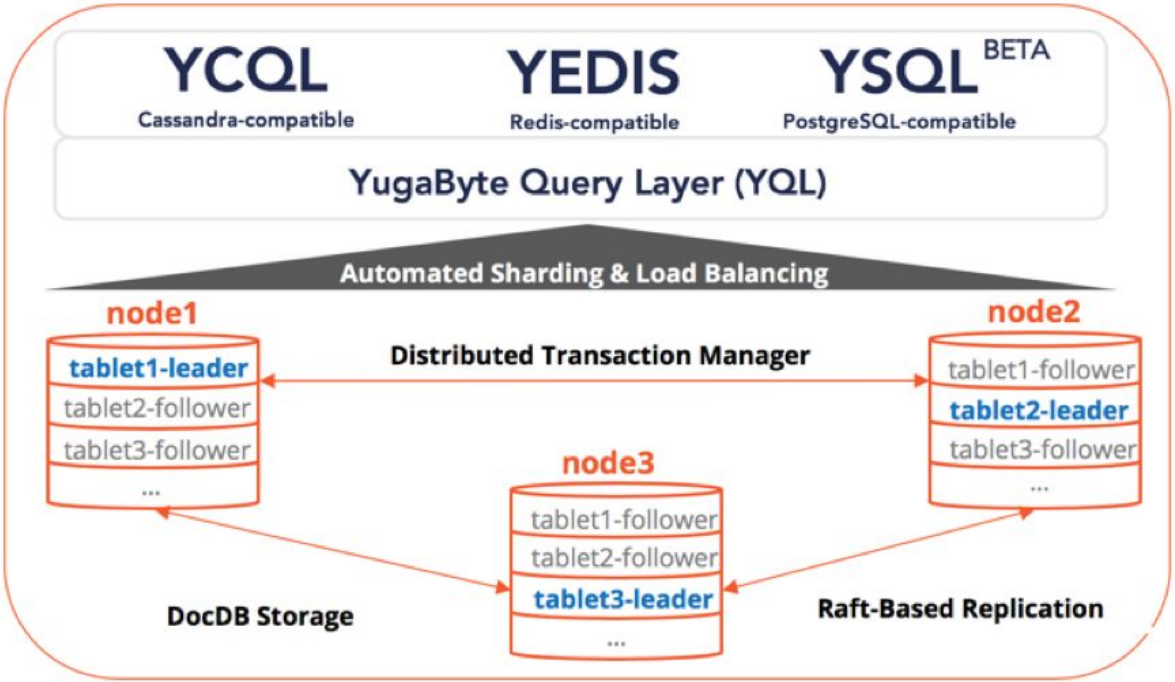


RDBMS Horizontal Scaling (Sharding)



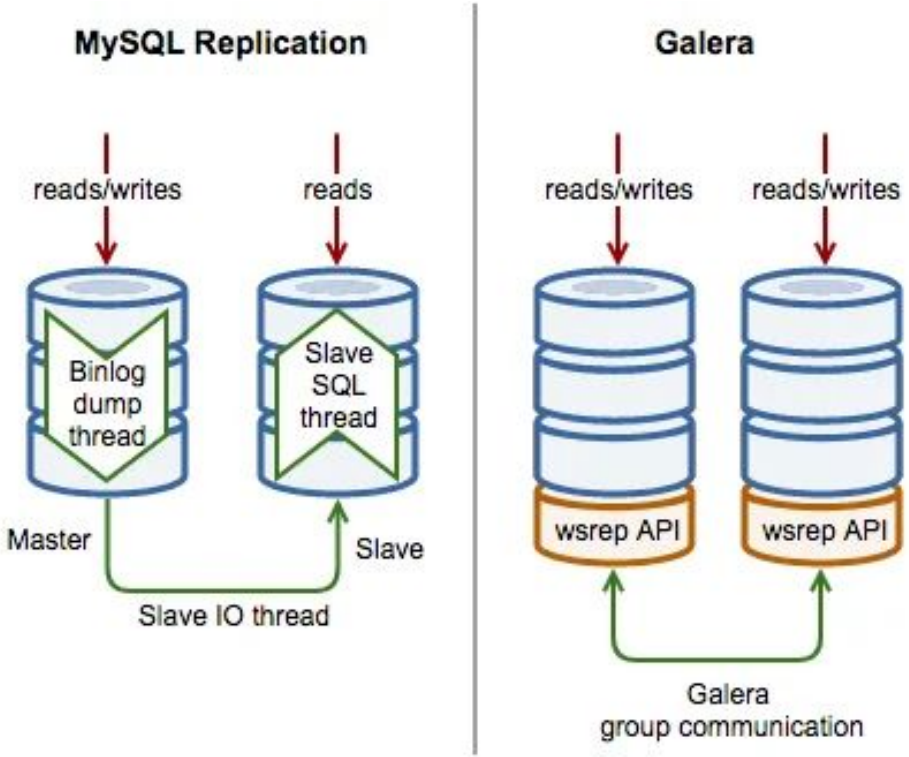
https://docs.citusdata.com/en/v7.0/aboutcitus/introduction_to_citus.html

RDBMS Horizontal Scaling

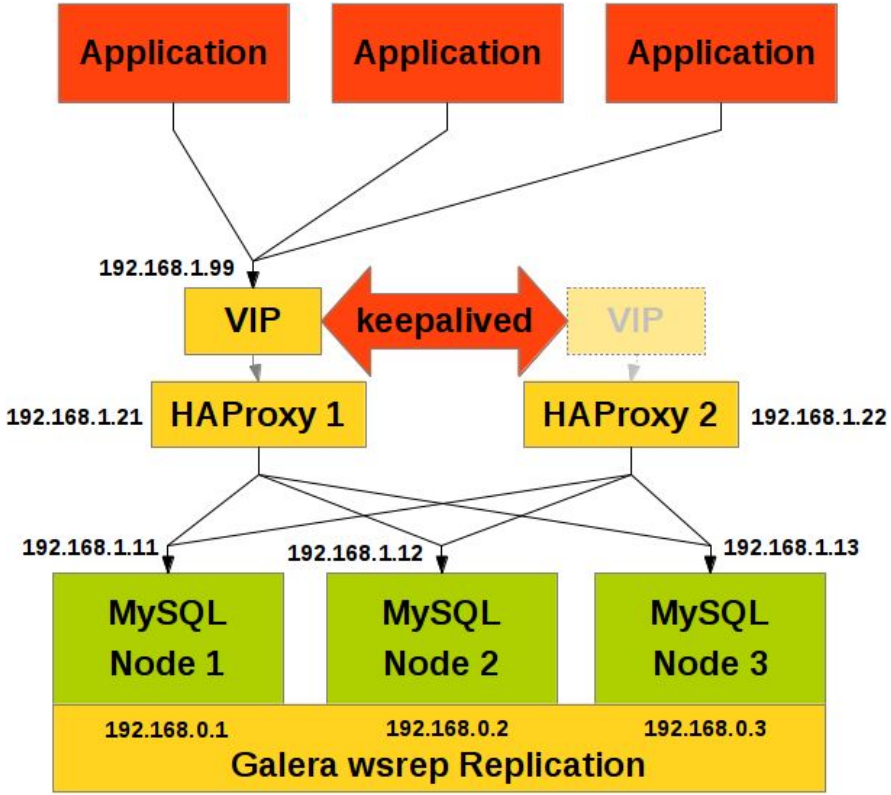


https://www.yugabyte.com/blog/ysql-architecture-implementing-distributed-postgresql-in-yugabyte-db/

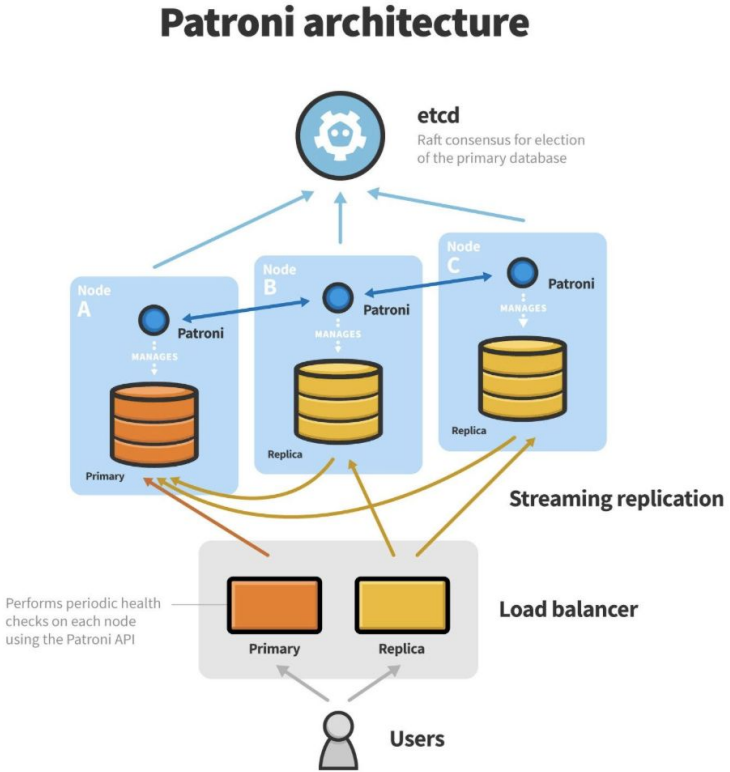
RDBMS Horizontal Scaling



RDBMS Horizontal Scaling



RDBMS Horizontal Scaling



RDBMS Shards/Partitions

Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
1	Alice	Anderson	Austin
2	Bob	Best	Boston
3	Carrie	Conway	Chicago
4	David	Doe	Denver

Vertical Shards

VS1

CUSTOMER ID	FIRST NAME	LAST NAME
1	Alice	Anderson
2	Bob	Best
3	Carrie	Conway
4	David	Doe

VS2

CUSTOMER ID	CITY
1	Austin
2	Boston
3	Chicago
4	Denver

Horizontal Shards

HS1

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
1	Alice	Anderson	Austin
2	Bob	Best	Boston

HS2

CUSTOMER ID	FIRST NAME	LAST NAME	CITY
3	Carrie	Conway	Chicago
4	David	Doe	Denver

RDBMS Shards/Partitions

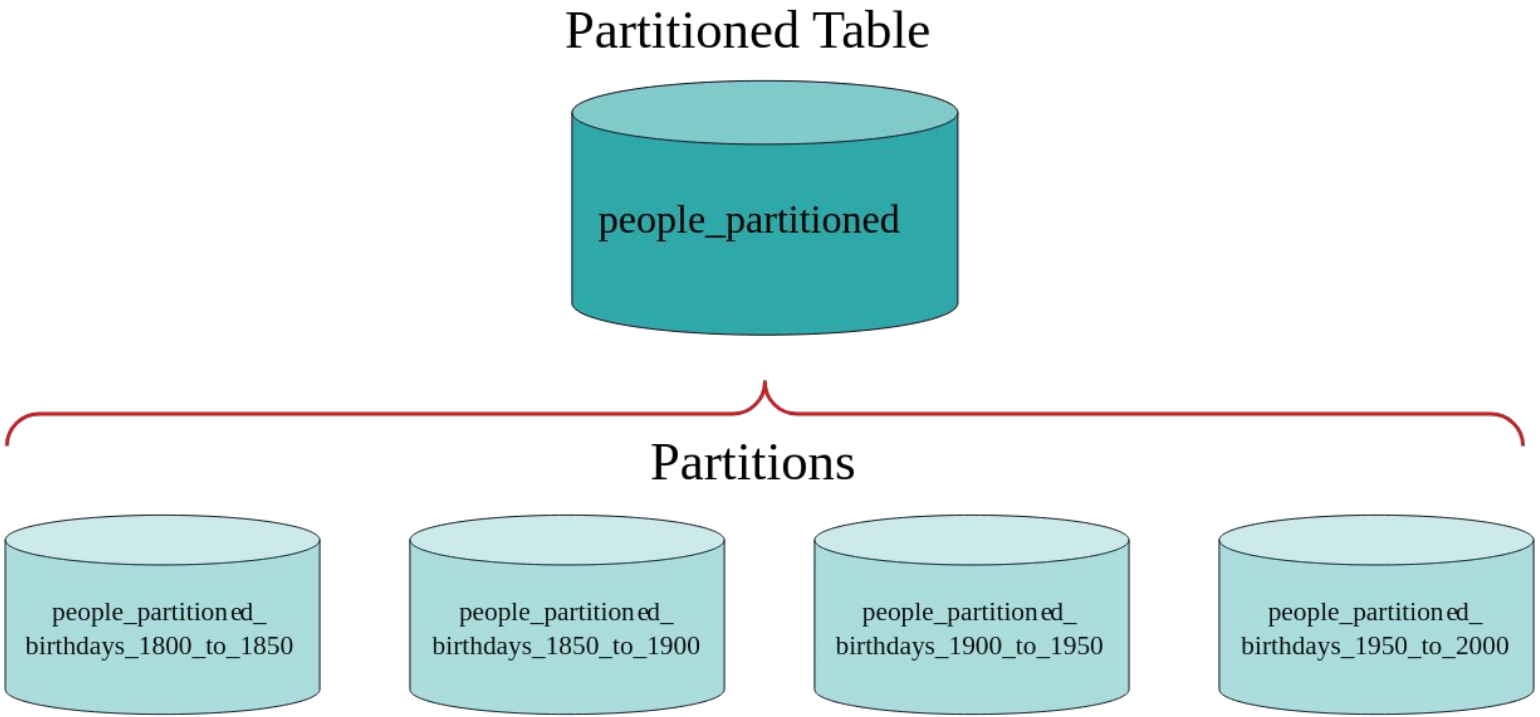
Range based - each partition/shard allocates rows with keys in a specified range of sharding key values:

- Range read queries can be satisfied from a single partition/shard.
- Writes may be bottlenecked due to operations on one partition.
- Tendention to introduce hot spots.

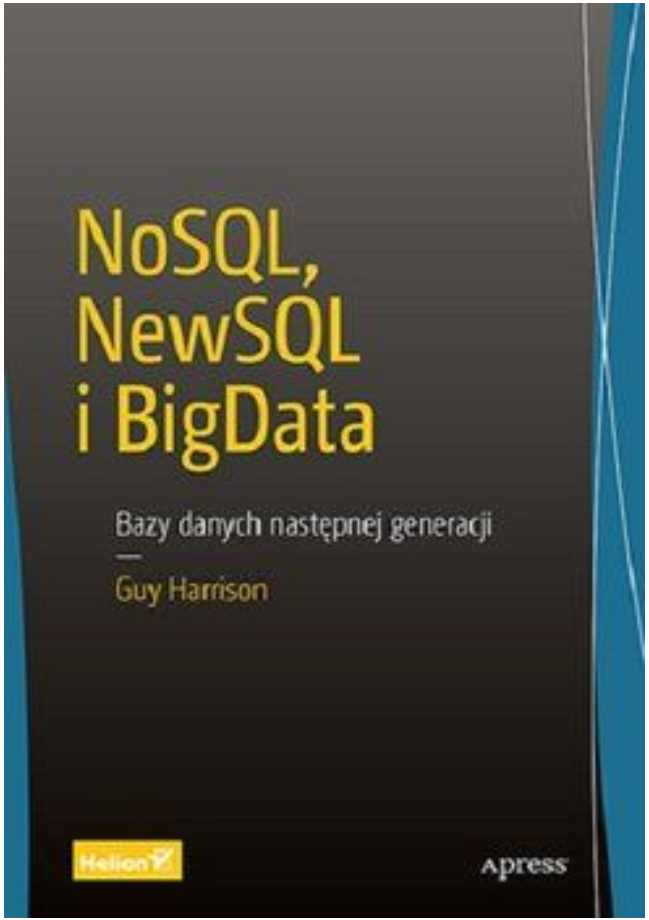
Hash based - rows are distributed according to some hashing function applied to the sharding key:

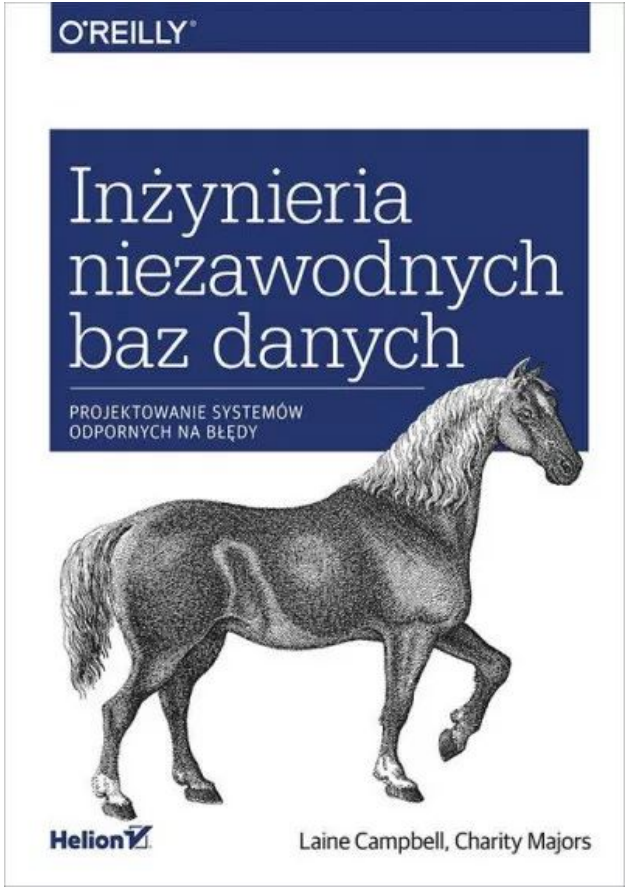
- Range read queries involve all partition/shards.
- Even distribution of records.

RDBMS Horizontal Scaling (Partitioning)



Inspiration





Summary

We have discussed:

- The aspects of replication, load balancing, high availability of RDBMS.
- Modern architecture of RDBMS.
- Common challenges.