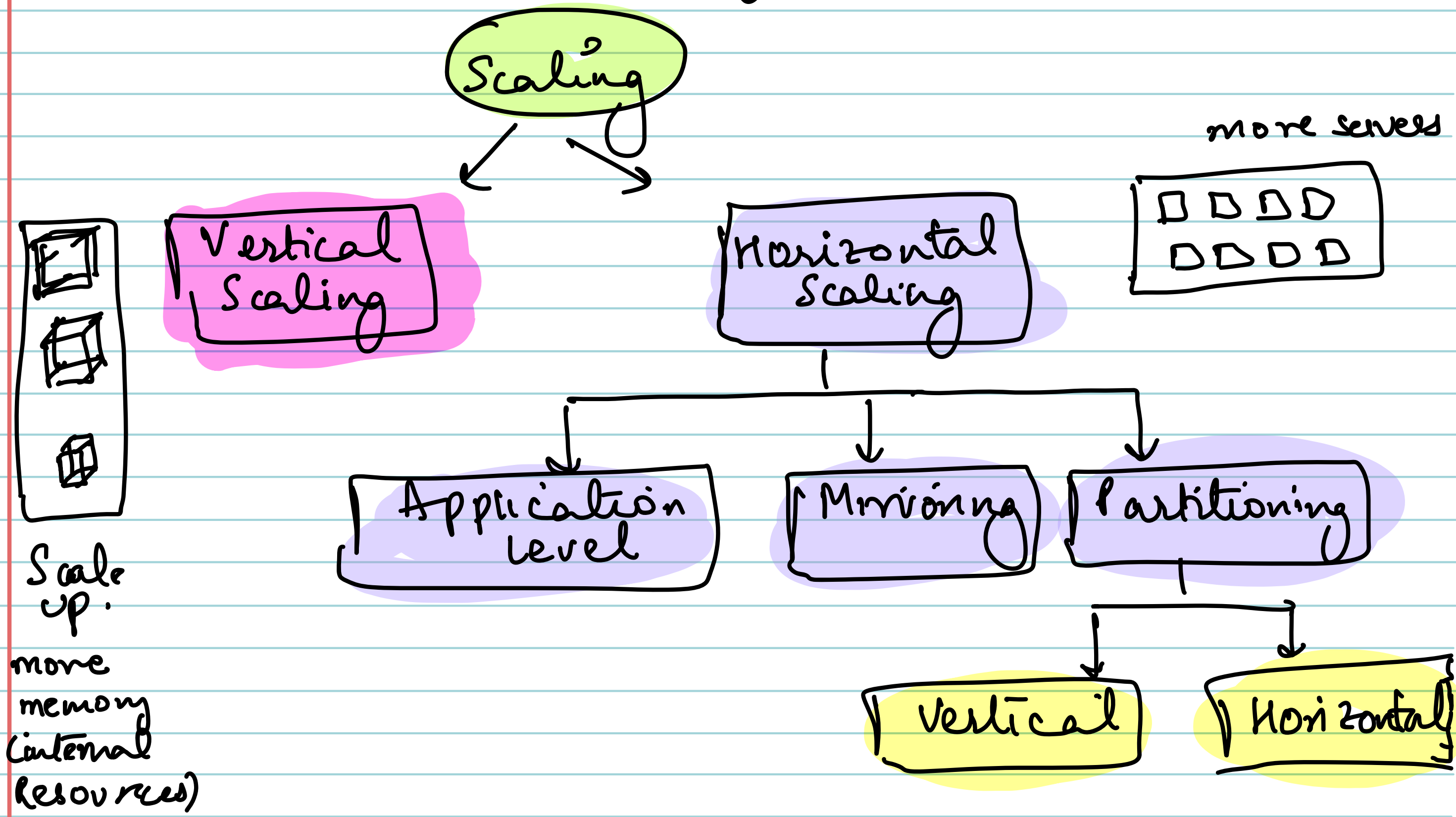
Database scaling and big data processing. Describe different approaches to database scaling (Vertical and Horizontal scaling, application level, mirroring, partitioning, sharding) and non-database approaches to big data storing and processing (DW, DataLakes, ETL/ELT).

Database scaling



Vertical scaling: Single server → ↑ Resources
- RAM
- CPU
- Storage (HD)

to handle ↑ load.

- Typically used with Relational DB MySQL, PostgreSQL.

Horizontal Scaling: Multiple server → load distributed across several servers & each server works on part of the workload

(↑s capacity by adding ↑ nodes)

- Most common in NoSQL databases (Cassandra)
- SQL DB → possible with sharding

- In SQL, less common due to consistency requirements.

Application level scaling : Implemented at software /Application level → . to help manage ↑ load
- Performance
- demand

w/o scaling directly DB

Idea : to reduce Burden on DB, and do handling of some tasks at the application layer

- In both SQL and NoSQL

eg : Redis is used for cach
↓NR, NSQL

Mirroring : keeping copies of Database on multiple servers to ensure high availability.

SQL → MySQL → for Fault Tolerence

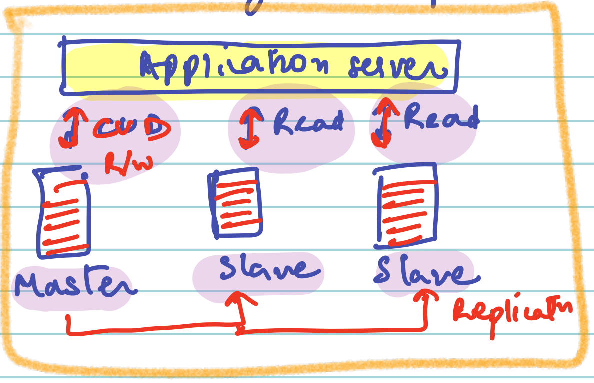
NoSQL → MongodB Replica sets.

Appln: Fast response for R Queries

Slow

slave inconsistent

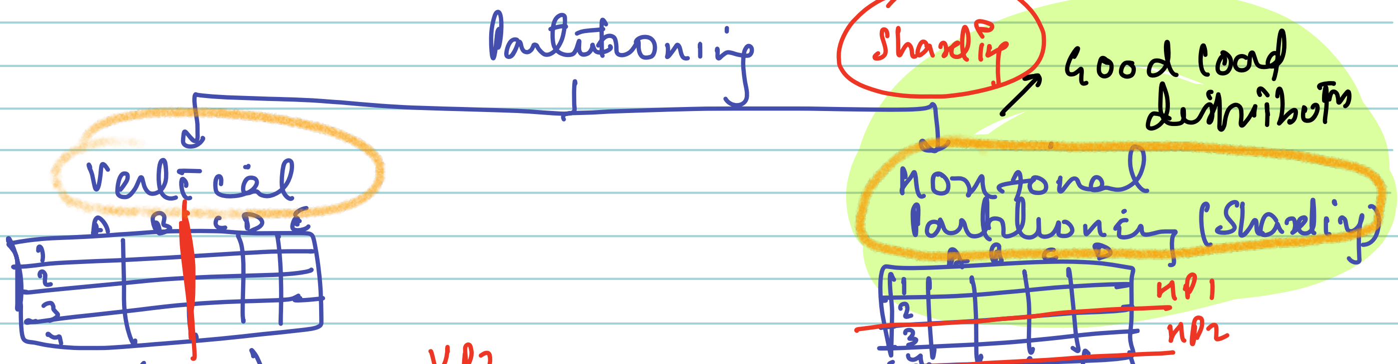
limited dB size

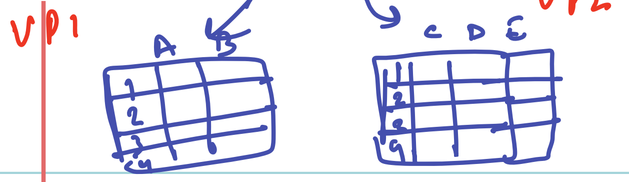


Partitioning : Divide large database into smaller manageable pieces. each stored on different server

Physically dividing data into separate data stores

→ To improve performance Scalability of large databases





Columns of dB ÷ into partitions

Partition : ÷ within same database

Sharding : ÷ among multiple databases or across serves

Types (Datadistribution)

Range based : 1-100
100-200

Random

data inner logic

Geo spatial location

Meth Function

Relationship b/w shards / servers don't exist

“ Management to route to shards ”