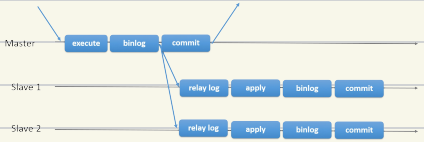
Distributed Databases

Replication



keep copy of data on several different nodes

- Data scalability - read throughput
- Geo-scalability - data close to client
- Fault tolerance

Replication Architectures

Leaders/Masters - accept write queries from clients

Followers/Slaves/Replicas - read-only access to data

Single leader/master-slave - a single leader accepts writes

Multi-leader/master-master - multiple leaders accept writes, keep in sync

Leaderless replication - all nodes peers in replication network

Write-ahead log (WAL)

write changes before modifying in append-only WAL (leader accepts after)

Logical-based replication

new record - values inserted

deleted record - unique id

updated record - id and updated values

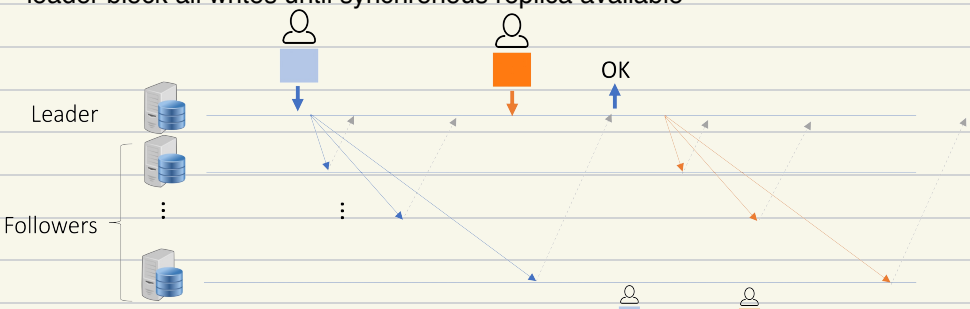
Creating replica

- consistent snapshot from leader
- ship to replica
- get id to state of leader's replication log
- replication function to latest leader id
- replica retrieve and apply replication log to catch up with leader

Synchronous replication

write confirmed by a configurable number of followers for success

- follower guaranteed to have up-to-date copy
- if leader fail, data still available on follower
- if synchronous follower doesn't respond, write cannot proceed
- leader block all writes until synchronous replica available



Asynchronous replication

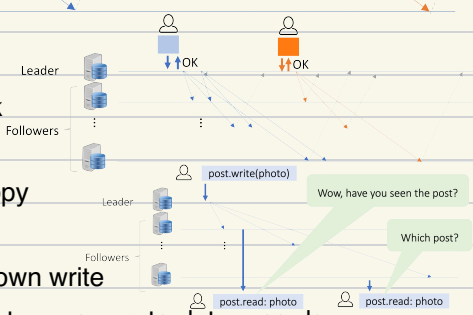
success when eager writes to own disk

- higher availability - no blocked writes
- follower not guaranteed up-to-date copy
- writes not durable if leader failure

Read-after-write - client might not see own write

(non-)Monotonic reads - stale replica return non up-to-date records

Causality violations - updates visible in different orders at replicas



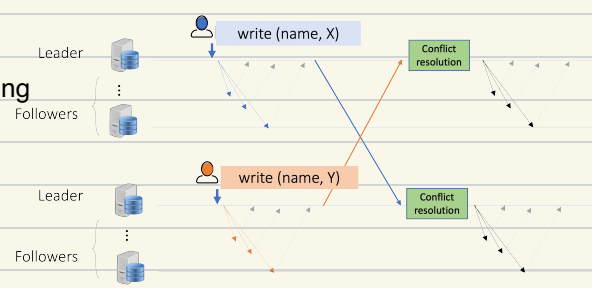
Multi-leader replication

write conflicts - two leaders writing

one leader per session

converge to consistent state:

- last-write-wins
- report conflict to app
- conflict-free data types

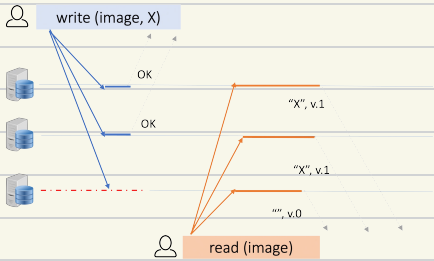


Leaderless replication

writes - successful id written to W replicas

reads - successful if written to R replicas

- W + R > N - up-to-date value
- W < N - process writes if node unavailable
- R < N - process reads if node unavailable

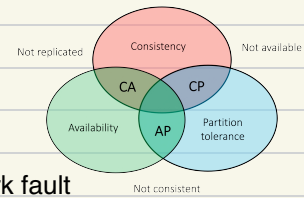


CAP theorem

Consistency - all nodes have same value

Availability - every request gets response

Partition tolerance - operation after component/network fault



Consistency models

contract between programmer and system

Strong - Linearisable, Sequential

Weak - Client-centric, Causal, Eventual

Eventual Consistency

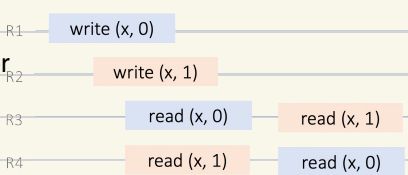
updates eventually delivered to replicas

all replicas reach consistent state if no more updates

Causal Consistency

operations delivered in correct order

partial order of events



Client-centric Consistency

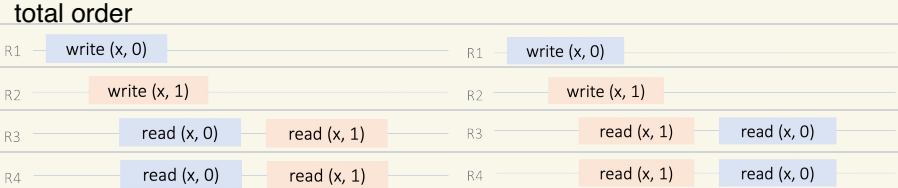
Monotonic read - read returns last value or a more recent value

Monotonic write - reflects effect of previous write operation

Read your writes - effects of write visible by read by same process

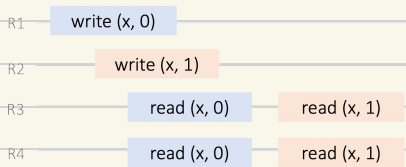
Sequential Consistency

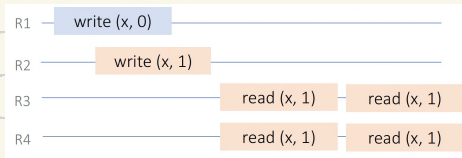
total order



Linearisability

Seuqnetial consistency + total order of operations conform real time order





Partitioning

break dataset into fractions, distributed to different hosts (sharding)

Scalability:

- queries in parallel
- reads/writes on multiple

always with replication

Range partitioning

Hash partitioning

Custom partitioning