Lecture 12

Wednesday, 18 October 2023 13:30

previously: fault tolerance
- process resilience
- process failure
- fault detection
- fault tolerance
- error recovery
- consensus
- link models

---

consistency & replication

reasons for partitioning system

partitioning vs. replication

↳ distribute multiple copies of same data over distinct nodes

↓ split data in smaller subsets + distribute over multiple nodes

reasons: ← quality drivers
1. data too large for single node
2. scalability
3. performance

often combined in practice

multiple replicas per node

key-value data can be partitioned by

key range — for totally ordered keys, efficient for range queries, 'hot ranges' due to skewed workload

hash of key — inefficient for range queries, 'hot keys' remain problem

↑ need not be cryptographic

↑ key with high read-/write volume

sometimes, partitions need to be rebalanced

scaling changes N, which redefines the key-partition mapping

quality drivers for replication:

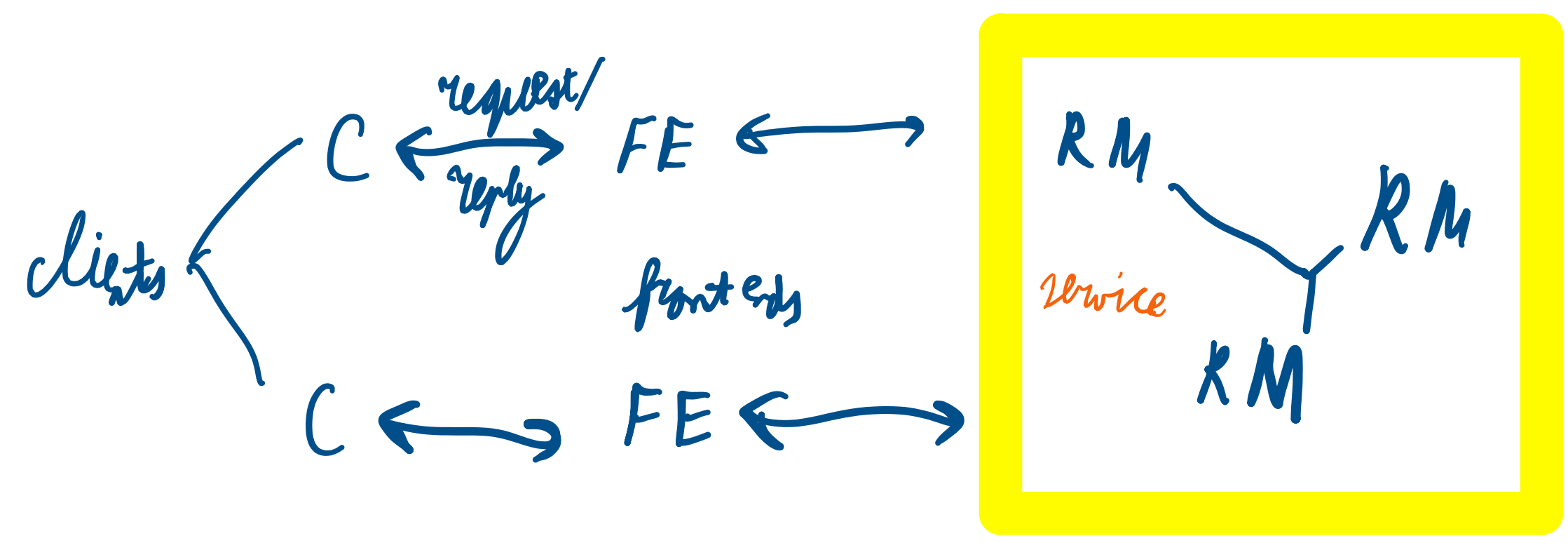
1. reliability — removes single point of failure; allow consensus protocols to deal with corrupted data
2. availability — reduced probability of all servers being unavailable
3. performance — concurrent access → improved throughput; latency reduction; network bandwidth reduction
4. scalability — load balancing

replication transparency: clients are unaware several replicas exist

↓

architectural concerns
- number + place of replicas
- maintaining consistency
- what architectural elements to use for storage & management
- protocols for read/write

basic model
- server is distributed data store with operations
  - update
  - query
- each server has special entity, replica manager (RM)
  - which manages local part of data store



1. request phase
2. coordination phase
3. execution phase
4. agreement phase
5. response phase

phases need not be executed in this order

Consistency means that replicas need to be kept the same

consistency models
- are contracts between data store & clients
- specify the unit of consistency
- determine outcome of sequence of read/write operations by one or more clients
- can be classified in two broad categories
  - data-centric models — perspective of data store
  - client-centric models — perspective of application

single-server paradigm

operations appear as if they were performed as indivisible actions by a single server with requires

same effect
- ↳ queries return same value
- ↳ updates leave data store in same state

same order → ...

two operations are conflicting if the outcome of executing them as a sequence of two atomic actions may potentially differ for the two possible execution orderings

read-write
write-write
conflicts

Consistency puts constraints on the interleaving of operations allowed to the single server

R1. writing requires lock — only one party can hold lock on an object

R2. operations issued by single client should be taken in order of issuance

R3. order of operations is consistent with global ordering