Exercise 01: Consistency and Replication (slide 07, p. 01 - 06)

Question: **What are the main reasons for using replication in distributed systems? How are replication and scalability related? Explain different ways of content replication and content placement**.

Replication is the process of maintaining several copies of an entity (object, data item, process, file, ...) at different nodes.

Reason for replication in distributed systems: (slide p. 05)

- Fault tolerance (redundancy)
  - switch-over in case of failures
  - protection against corrupted data (→ voting)
- Performance (and scalability)
  - <u>scale in numbers (cluster)</u>
    is used, when different processes access data managed by only one server. In this case, performance can be increased by replicating the whole server (cluster).
  - <u>scale in geographic/topological complexity (place copies of data and processes in proximity)</u>
    A copy of the data is placed near the process which uses it → reduces data access time
- BUT: price for replication: keeping replicas consistent in face of updates is costly → trade-off

Replication vs. Scalability (slide p. 05 - 06)

Replication implicates overhead, because data has to be kept consistent. So it has to be estimated if the use of replication is worth the price (traffic, resources).

Content Replication and Content Placement (no slides!)

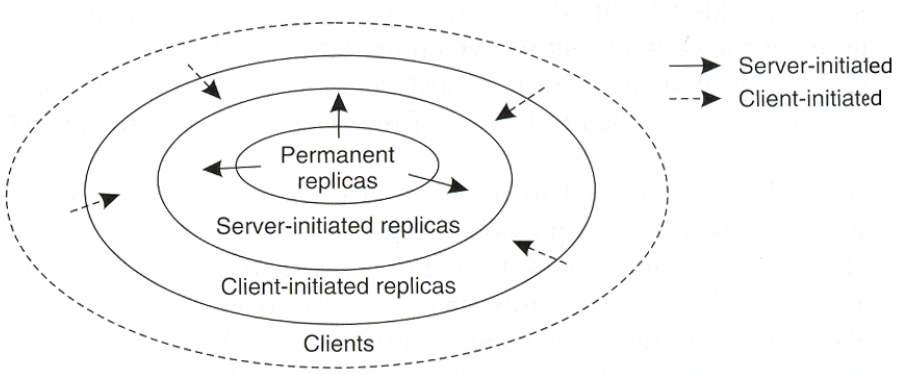


Figure 7-17. The logical organization of different kinds of copies of a database into three concentric rings.

<u>General:</u>

- Content Replication: Should replicas be reachable permanently or only at runtime?
- Content Placement: Where are replicas installed?

Techniques:

- Permanent Replication
    - Cluster based: Servers placed in the same location; e.g. Round Robin can be executed.
    - Mirroring: Mirrored Servers placed in the same location.

- Server initiated replicas:
    - Replica is initialized dynamically, if there are many requests. Therefore file-transactions are counted. It is also necessary to know a file-transaction comes from.
    - Create thresholds of replicas (when is a replica created?).
    - Replica is deleted when it is not need anymore (BUT there always has to at least one replica!).
    - Replica should preferably be created where the most requests come from!
    - Replicas are an ideal tool for intercepting flash crowds. Wikipedia: Flash crowd describes a network phenomenon where a network or host suddenly receives a lot of traffic. This is due to the appearance of a web site on a blog or news column.

- Client initiated replicas:
  Also known as cache; are placed in the local memory or local network. Used for holding access time low, when accessing data; are saved for a limited time for prevention of the use of extremely old data. If cache is shared, performance may be increased.

Question2: Compare different ways of update propagation (content distribution).

Concerning update propagation of replicas, several aspects need to be looked at.

State vs. Operation (slide 31)

Update propagation deals with what should be propagated to the replicas. There are 3 options:

- Notification:
  Other replicas get informed that there was an update (invalidation of their data), data only gets transferred if there is a read access
  - Requires low bandwidth
  - Good for small read-to-write ratio and large amount of data
- State transfer: changed data gets transferred
  - Requires high bandwith
  - Good for high read-to-write ration
- Operation transfer: the update operation gets propagated
  - Uses little bandwidth if parameter size is small
  - Requires more processing power, especially if operation is complex

Pull vs Push Protocols (slide 32, 33,34)

PUSH

- Updates get propagated without them asking for it
  - Server needs to keep track of clients → Stateful server, this can result in significant overhead for storing client caches etc.
  - Used by permanent and server-initiated replicas, but also by some client caches
  - Provides a high degree of consistency
  - Multicasting: server uses a multicast group in order to send updates to other servers

PULL

- **Replicas request another replica to send updates**
  - **used by client caches**
    - **E.g. web: browser checks if the cached data is still up-to-date**
  - Efficient, if the read/update ratio is relatively low
  - Response time increases if a cache entry is old and needs to be updated
  - Unicasting

Issue	**Push-based**	**Pull-based**
State of server	List of client replicas and caches	None
Message sent	Update (and possibly fetch update later)	Poll and update
Response time at client	Immediate (or fetch-update time)	Fetch-update time

It is common to use a combination of PUSH Aand PULL → Lease

LEASE (slide 35)
A Lease is promise by the server to send updates t othe client cache (push) for a fixed period of time
If a lease expires, the client has to poll or request a new lease

Types of Leases:

- **Age-based:** data that hasn't been modified for a long time is assumed to have little changes in the future. That data gets a high expiry time.
- **Renewal-frequency-based:** Client with a high update frequency get a higher expiry time
- **State-space overhead:** If a server is overloaded, it lowers the expiry time of its leases

Blocking vs non-blocking

- Synchronous (blocking, eager):

All replicas are updated immediately, then reply to client

- Asynchronous (non-blocking, lazy):

Update is applied on one copy, then reply to client, propagation to other replicas afterwards

Question 4 Slide 7: p. 38-58

Explain "replicated-write" protocols and compare different alternatives. What are the challenges of "active replication"? How do "quorum-based" replication protocols work and hox can they avoid read-write and write-write conflicts?

Replicated-Write Protocols

In replicated-write protocols, write operations can be carried out at multiple replicas instead of only one, as in the case of primary-based replicas. A distinction can be made between active replication, in which an operation is forwarded to all replicas, and consistency protocols based on majority voting.

Active Replication

In active replication, each replica has an associated process that carries out update operations. In contrast to other protocols, updates are generally propagated by means of the write operation that causes the update. In other words, the operations sent to each replica.

One problem with active replication is that operations need to be carried out in the same order everywhere. Consequently, what is needed is a totally-ordered multicast mechanism. Such a multicast can be implemented using Lamport's logical clocks.

Advantages:

☐Simplicity (same code everywhere)
☐No bottleneck
☐No single point of failure

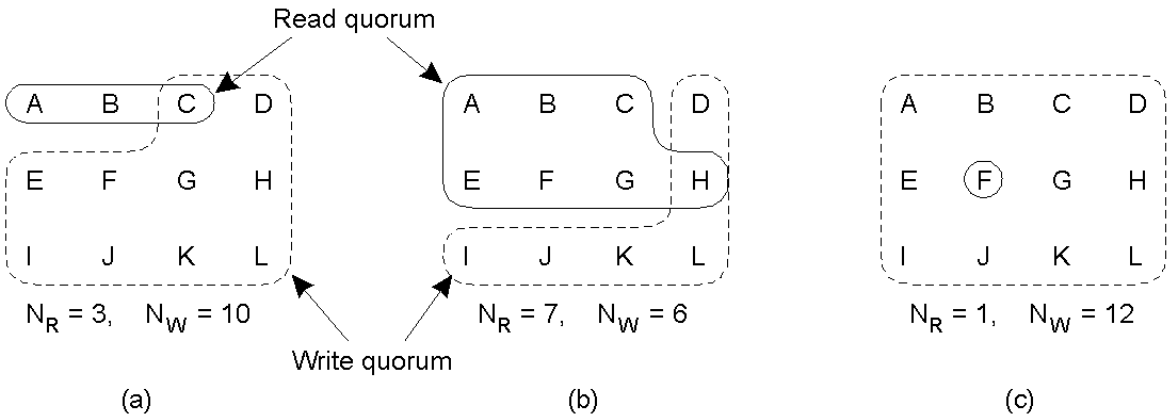
Disadvantage:

☐Determinism required
☐Ordering more difficult (group communication)

Quorum-Based Protocols

A **quorum** is the minimum number of votes that a distributed transaction has to obtain in order to be allowed to perform an operation. A **quorum**-based technique is implemented to enforce consistent operation in a distributed system.

- Write operations are performed on a write quorum N_W of replicas
- Read operations are performed on a read quorum N_R of replicas
- Two constraints must be obeyed:
  $N_R + N_W > N$ (R-W conflicts)
  $N_W > N/2$ (W-W conflicts)
- Version numbers needed
- "Majority voting"



a) A correct choice of read and write set
b) A choice that may lead to write-write conflicts
c) A correct choice, known as ROWA (read one, write all)

Exercise 05: Object Replication (slide 07, p. 50 - 51)

Question: What are the particularities with object replication? Depict a possible realization of replication transparency in object systems ("replicated invocation").

Problem:

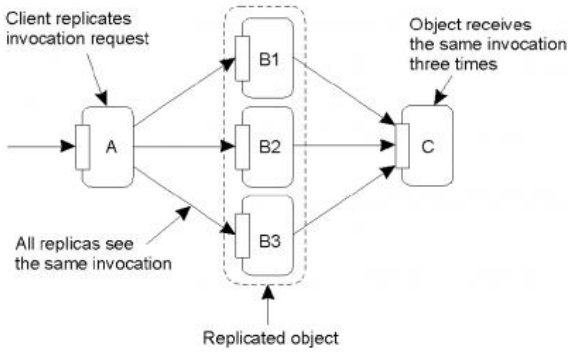
When using objects in distributed systems, which are distributed as well, every access on these objects has to be consistent. Not doing so would lead to inconsistency of the whole system (entry consistency)! So, concurrent executions of methods of an object have to be prevented (e.g. relatively easy through the use of locking techniques). Additionally modifications (method calls) have to be distributed to all replicas of an object, to prevent different concurrent transactions on the same distributed system at the same time.

This goal can be achieved by the following methods:

- Primary-based approach:
  a coordinator takes control of executions and delegates them.
  → Problem: Overhead, loose of scalability, coordinator has to "know" all other distributed objects.
- Totally-ordered-multicasts:
  Method calls are numbered e.g. through Lamport clocks. Afterwards these methods are executed on every object in the specified numbered order.
  → Problem: very complex, much overhead

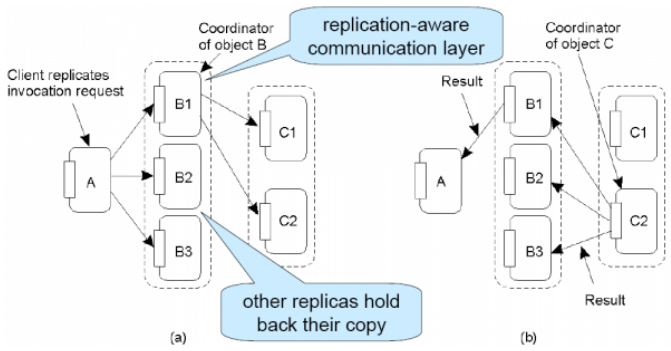
Replicated Invocation

Problem: Method calls on a distributed object, which maybe doesn't exist at all, are executed several times.



Solution: Realization of Replication Transpareny

If (outgoing from A) a call happens in B deriving in A, A calls all replicas in B (B1, B2, B3). If B is calling a Method of C now, this is only done by the coordinator of B (e.g. B1) at C1 and C2. Otherwise, if all 3 replicas of B executed the call, the method of C would be executed too often. The replies are only returned by the coordinator (e.g. from C2 to all 3 replicas of B, and from B1 to A).



Question6: What are epidemic protocols? What are the advatages and disadvantages? Explain "gossiping" ("rumor spreading") as one model for replica update propagation. What is the Anti-Entropy Modell in this context?

Epidemic Protocols: (slide 56)

Epidemic protocols are used if only eventual consistency is required, i.e. if there is an update, all replicas must be identical at some point. The aim of these protocols is to distribute information quickly among many nodes.

Advantages: Good Scalability, Updates get sent by using as little message as possible → low network usage

Disadvantages: Letting everyone know that data has been removed is difficult because of eventual consistency

Gossip Protpocols: (slide 57)

Gossiping (also "rumor spreading") is a propagation model for epidemic protocols. It a special form of an anti-entropy model (see section below). If a data element gets updated, the server pushed the updated to a random other server, who does the same thing and so on…

If a server receives an update that he already has (he is already "infected") the probability that he will continue gossiping decreases**.**

Advantages: Updates get propagated quickly

Disadvantage: There is no guarantee that each server will be receive an update

Anti-Entropy Model: (no slide because gossip protocols are special form of this)

An anti-entropy model in general is a propagation model chooses random servers for the exchange of updates. It can operate pull- or push-based. If many nodes are infected, the probability that a push-based approach will find more nodes is low, as opposed to a pull based approach.