Consistency & Replication

Amir H. Payberah
amir@sics.se

Amirkabir University of Technology
(Tehran Polytechnic)

Based on slides by Maarten Van Steen

What is the problem?

Motivation

- Data replication: to enhance reliability or improve performance.

Motivation

- Data replication: to enhance reliability or improve performance.
- Problem: keeping replicas consistent.

Motivation

- Data replication: to enhance reliability or improve performance.
- Problem: keeping replicas consistent.
- When one copy is updated we need to ensure that the other copies are updated as well, otherwise the replicas will no longer be the same.

Performance and Scalability

- To keep replicas consistent, we generally need to ensure that all conflicting operations are done in the same order everywhere.

Performance and Scalability

- To keep replicas consistent, we generally need to ensure that all conflicting operations are done in the same order everywhere.

- Conflicting operations:
  - Read-write conflict: a read operation and a write operation act concurrently.
  - Write-write conflict: two concurrent write operations.

Performance and Scalability

- To keep replicas consistent, we generally need to ensure that all conflicting operations are done in the same order everywhere.

- Conflicting operations:
  - Read-write conflict: a read operation and a write operation act concurrently.
  - Write-write conflict: two concurrent write operations.

- Guaranteeing global ordering on conflicting operations may be a costly operation, downgrading scalability.

Replica Management

Replica Management

- Where the replicas should be placed?

Replica Management

- Where the replicas should be placed?
- The placement problem is split into two subproblems:

Replica Management

- Where the replicas should be placed?
- The placement problem is split into two subproblems:
  - Placing replica servers: finding the best locations to place a server.

Replica Management

- Where the replicas should be placed?
- The placement problem is split into two subproblems:
  - Placing replica servers: finding the best locations to place a server.
  - Placing content: finding the best servers for placing content.

Distribution Protocols

- Replica server placement
- Content replication and placement
- Content distribution

Replica Placement

Replica Placement (1/2)

- Figure out what the best K places are out of N possible locations.
- The first chosen location minimizes the average distance to all clients.
- Select best location out of $N - k$ for which the average distance to clients is minimal. Then choose the next best server.

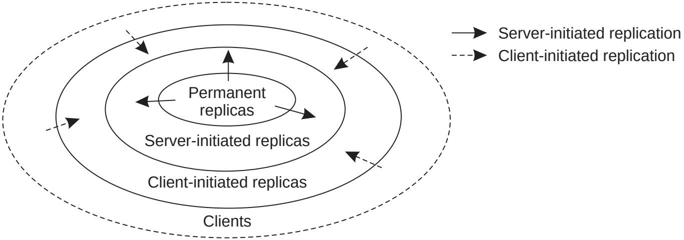
Replica Placement (2/2)

- Ignore the position of clients and only take the topology of the Internet as formed by the autonomous systems (AS).
- AS: the nodes all run the same routing protocol and which is managed by a single organization.
- Select the K-th largest autonomous system and place a server at the best-connected host.

Content Replication

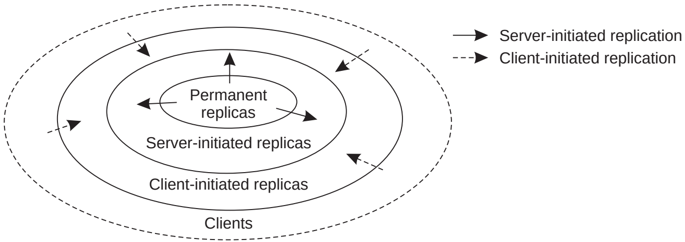
Content Replication

- Three different types of replicas:



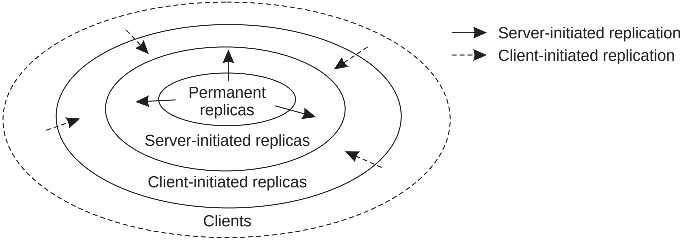
Content Replication

- Three different types of replicas:
  - Permanent replicas: process that always have a replica.



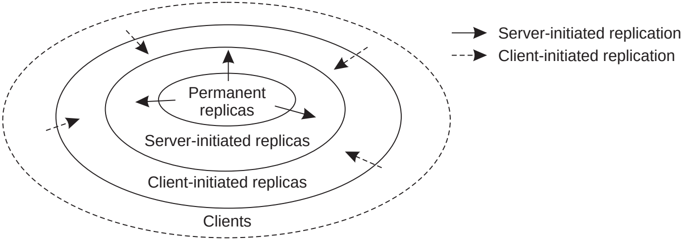
Content Replication

- Three different types of replicas:
  - Permanent replicas: process that always have a replica.
  - Server-initiated replica: process that can dynamically host a replica on request of another server in the data store.



Content Replication

- Three different types of replicas:
  - Permanent replicas: process that always have a replica.
  - Server-initiated replica: process that can dynamically host a replica on request of another server in the data store.
  - Client-initiated replica: process that can dynamically host a replica on request of a client (client cache).

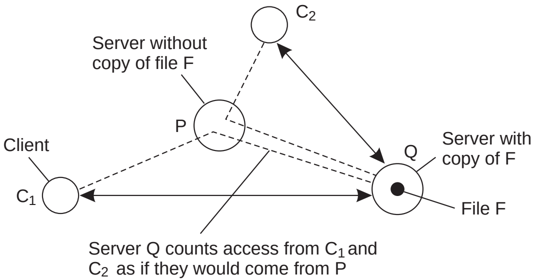


Permanent Replication

- Initial set of replicas that constitute a distributed data store.
- Example, web site:
  1. The web site files are replicated across a limited number of servers at a single location: the arrived requests are forwarded to one of the servers (round-robin strategy).
  2. Mirroring, where a web site is copied to a limited number of servers, which are geographically spread across the Internet.

Server-Initiated Replicas

- Keep track of access counts per file, aggregated by considering closest server to requesting clients.
- Number of accesses drops below threshold $D \Rightarrow$ drop file.
- Number of accesses exceeds threshold $R \Rightarrow$ replicate file.
- Number of accesses between D and $R \Rightarrow$ migrate file.



Client-Initiated Replicas

- Client-initiated replicas are more commonly known as (client) caches.

- A cache is a local storage facility that is used by a client to temporarily store a copy of the data it has just requested.

- Managing the cache is left entirely to the client.

Content Distribution

Content Distribution (1/2)

- What is actually to be propagated:

Content Distribution (1/2)

- What is actually to be propagated:
  - Propagate only notification/invalidation of update: (often used for caches)

Content Distribution (1/2)

- What is actually to be propagated:
  - Propagate only notification/invalidation of update: (often used for caches)
  - Transfer data from one copy to another (distributed databases): passive replication

Content Distribution (1/2)

- What is actually to be propagated:
  - Propagate only notification/invalidation of update: (often used for caches)
  - Transfer data from one copy to another (distributed databases): passive replication
  - Propagate the update operation to other copies: active replication

Content Distribution (1/2)

- What is actually to be propagated:
  - Propagate only notification/invalidation of update: (often used for caches)
  - Transfer data from one copy to another (distributed databases): passive replication
  - Propagate the update operation to other copies: active replication

- No single approach is the best, but depends highly on available bandwidth and read-to-write ratio at replicas.

Content Distribution (2/2)

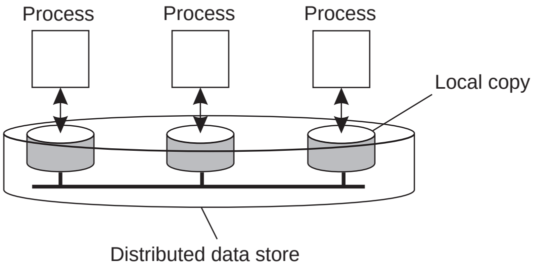
- Pushing updates: server-initiated approach, in which update is propagated regardless whether target asked for it.

- Pulling updates: client-initiated approach, in which client requests to be updated.

Data Centric Consistency Model

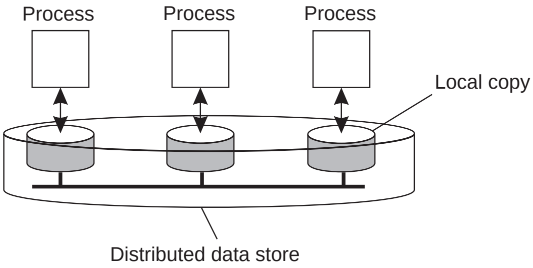
Data-Centric Consistency Models

- A data store is a distributed collection of storages:



Data-Centric Consistency Models

- A data store is a distributed collection of storages:



- Consistency model: a contract between a (distributed) data store and processes, in which the data store specifies precisely what the results of read and write operations are in the presence of concurrency.

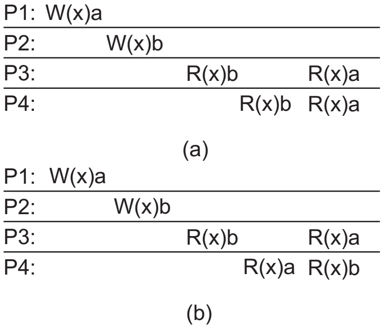
Consistency Models

- Sequential consistency
- Causal consistency

Sequential Consistency

Sequential Consistency

- The result of any execution is the same as if the operations of all processes (read and write) were executed in some sequential order, and the operations of each individual process appear in this sequence in the order specified by its program.



Sequential Consistency

- The result of any execution is the same as if the operations of all processes (read and write) were executed in some sequential order, and the operations of each individual process appear in this sequence in the order specified by its program.

- Nothing is said about time: no reference to the most recent write operation on a data item.

P1:	W(x)a				
P2:		W(x)b			
P3:			R(x)b		R(x)a
P4:				R(x)b	R(x)a

(a)

P1:	W(x)a				
P2:		W(x)b			
P3:			R(x)b		R(x)a
P4:				R(x)a	R(x)b

(b)

Causal Consistency

Causal Consistency (1/3)

- The causal consistency model represents a weakening of sequential consistency.

Causal Consistency (1/3)

- The causal consistency model represents a weakening of sequential consistency.
- It makes a distinction between events that are potentially causally related and those that are not.

Causal Consistency (1/3)

- The causal consistency model represents a weakening of sequential consistency.
- It makes a distinction between events that are potentially causally related and those that are not.
- If event *b* is caused or influenced by an earlier event *a*, causality requires that everyone else first see *a*, then see *b*.

Causal Consistency (1/3)

- The causal consistency model represents a weakening of sequential consistency.
- It makes a distinction between events that are potentially causally related and those that are not.
- If event *b* is caused or influenced by an earlier event *a*, causality requires that everyone else first see *a*, then see *b*.
- On the other hand, if two processes spontaneously and simultaneously write two different data items, these are not causally related: concurrent processes

Causal Consistency (2/3)

- Writes that are potentially causally related must be seen by all processes in the same order.
- Concurrent writes may be seen in a different order by different processes.

P1: W(x)a				
P2:	R(x)a	W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

(a)

P1: W(x)a				
P2:		W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

(b)

Causal Consistency (3/3)

- This sequence is allowed with a causally-consistent store, but not with a sequentially consistent store.

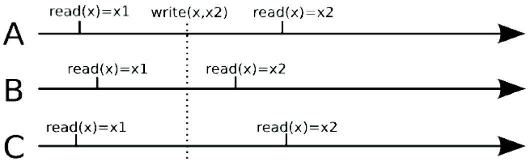
P1:	W(x)a			W(x)c		
P2:		R(x)a	W(x)b			
P3:		R(x)a			R(x)c	R(x)b
P4:		R(x)a			R(x)b	R(x)c

Client Centric Consistency Model

Client-Centric Consistency Models

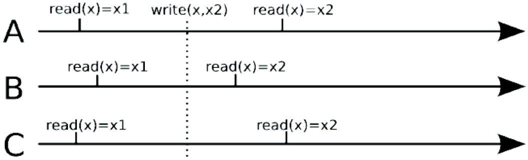
- Show how we can perhaps avoid system-wide consistency, by concentrating on what specific clients want, instead of what should be maintained by servers.

Strong Consistency vs. Eventual Consistency

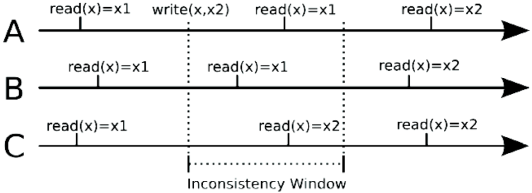


strong consistency

Strong Consistency vs. Eventual Consistency



strong consistency



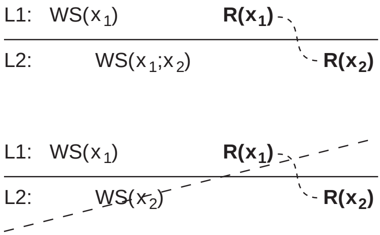
eventual consistency

Notation

- $WS(x_i[t])$ is the set of write operations (at local copy L_i) that lead to version x_i of x (at time t).
- $WS(x_i[t_1]; x_j[t_2])$ indicates that the operations in $WS(x_i[1])$ have also been performed at local copy L_j at a later time $t2$.
  - $WS(x_i[t_1])$ is part of $WS(x_j[t_2])$.
- Note: Parameter t is omitted from figures.

Monotonic Reads (1/2)

- If a process reads the value of a data item x, any successive read operation on x by that process will always return that same or a more recent value.



Monotonic Reads (2/2)

- Example 1:
  - Automatically reading your personal calendar updates from different servers.
  - Monotonic Reads guarantees that the user sees all updates, no matter from which server the automatic reading takes place.

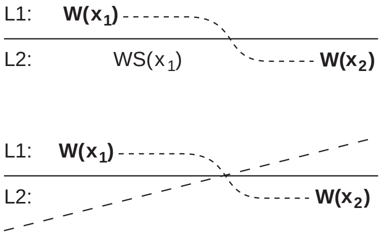
Monotonic Reads (2/2)

- Example 1:
  - Automatically reading your personal calendar updates from different servers.
  - Monotonic Reads guarantees that the user sees all updates, no matter from which server the automatic reading takes place.

- Example 2:
  - Reading (not modifying) incoming mail while you are on the move.
  - Each time you connect to a different e-mail server, that server fetches (at least) all the updates from the server you previously visited.

Monotonic Writes (1/2)

- A write operation by a process on a data item x is completed before any successive write operation on x by the same process.



Monotonic Writes (2/2)

- Example 1:
  - Updating a program at server S_2, and ensuring that all components on which compilation and linking depends, are also placed at S_2.

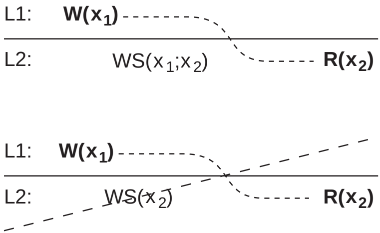
Monotonic Writes (2/2)

- Example 1:
  - Updating a program at server S_2, and ensuring that all components on which compilation and linking depends, are also placed at S_2.

- Example 2:
  - Maintaining versions of replicated files in the correct order everywhere (propagate the previous version to the server where the newest version is installed).

Read Your Writes

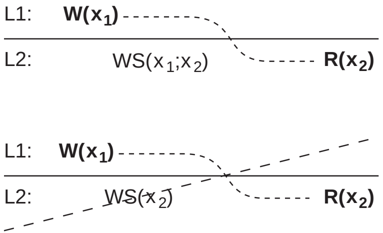
- The effect of a write operation by a process on data item x, will always be seen by a successive read operation on x by the same process.



Read Your Writes

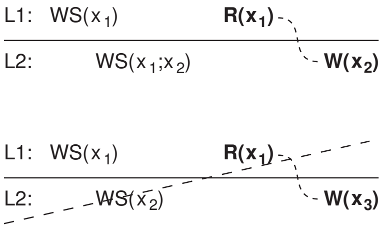
- The effect of a write operation by a process on data item x, will always be seen by a successive read operation on x by the same process.

- Example: updating your Web page and guaranteeing that your Web browser shows the newest version instead of its cached copy.



Writes Follow Reads

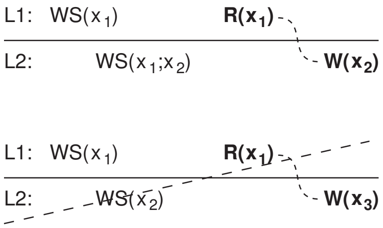
- A write operation by a process on a data item x following a previous read operation on x by the same process, is guaranteed to take place on the same or a more recent value of x that was read.



Writes Follow Reads

- A write operation by a process on a data item x following a previous read operation on x by the same process, is guaranteed to take place on the same or a more recent value of x that was read.

- Example: see reactions to posted articles only if you have seen the original posting.



Consistency Protocols

Consistency Protocols

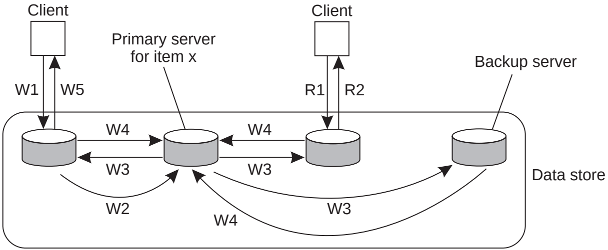
- Consistency protocol: describes the implementation of a specific consistency model.
  - Primary-based protocols
  - Replicated-write protocols

Primary-Based Protocols

Primary-Based Protocols

- In these protocols, each data item x in the data store has an associated primary, which is responsible for coordinating write operations on x.
- Primary-based protocols:
  - Remote-write protocols
  - Local-write protocols

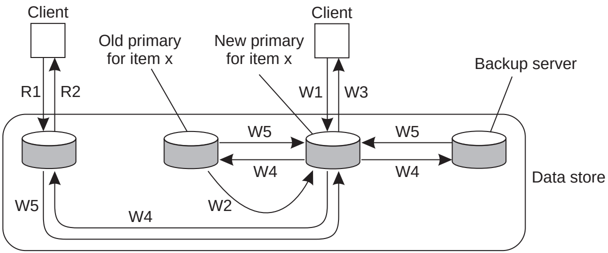
Remote-Write Protocols



W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

Local-Write Protocols



W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

Replicated-Write Protocols

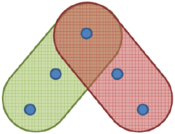
Replicated-Write Protocols

- Quorum Model
- N: the number of nodes to which a data item is replicated.
- R: the number of nodes a value has to be read from to be accepted.
- W: the number of nodes a new value has to be written to before the write operation is finished.

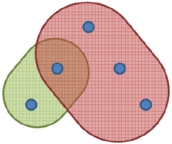
Replicated-Write Protocols

- Quorum Model
- N: the number of nodes to which a data item is replicated.
- R: the number of nodes a value has to be read from to be accepted.
- W: the number of nodes a new value has to be written to before the write operation is finished.

- To enforce consistency: $R + W > N$



R = 3, W = 3, N = 5



R = 4, W = 2, N = 5

Summary

Summary

- Replication *rightarrow* problem: consistency
- Replica management: replica server placement, content placement, content distribution
- Data-centric consistency models: sequential, casual
- Client-centric consistency models: eventual consistency
- Monotonic reads, monotonic writes, read your write, writes follow reads
- Consistency protocols: primary-based, quorum

Reading

- Chapter 7 of the Distributed Systems: Principles and Paradigms.

Questions?