Consistency and Replication (part b)

EECS 591 Lecture Notes

Farnam Jahanian

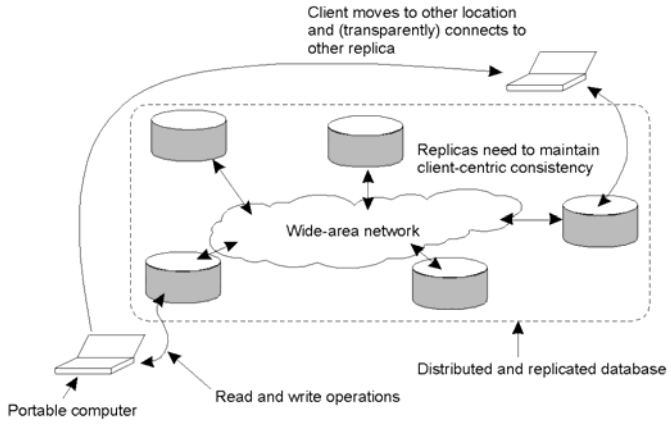
University of Michigan

Tanenbaum Chapter 6.1-6.5

Eventual Consistency

- A very weak consistency model characterized by the lack of simultaneous updates or easy-to-resolve simultaneous updates by a small set of processes
- Common property of data stores with eventual consistency: if no update takes place for a long time, *all replicas will gradually become consistent, i.e. propagate updates to all replicas in a lazy fashion.*
- Cheap to implement (later).
- Examples:
  - DNS: name space partitioned into domain, each domain is assigned to a single naming authority (responsible for making updates), updates propagated to all copies in a lazy fashion.
  - Web pages: typically updated by a single client, no write-write conflicts, cache copies may be out-of-date, acceptable to have inconsistency for a short interval.

Example: mobile user accessing different replicas



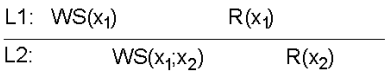
- Eventual consistency is easy to implement if a client always accesses the same replica
- If the user disconnects and reconnects to another replica, it may observe inconsistencies in the data store
- Client-centric consistency is one such model …

Client-Centric Consistency

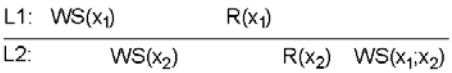
- Originated from the work on Bayou DB for mobile systems – wireless access and/or unreliable network connectivity
- Provides guarantees for a single client's access to a replicated data store
- No guarantees concerning accesses by different clients
- Four models in client-centric consistency:
  - Monotonic Read Consistency
  - Monotonic Write Consistency
  - Read-your-writes Consistency
  - Writes-follows-reads Consistency

Monotonic Reads

If a process reads the value of a data item x, any successive read operations on x by that process will always return that same value or a more recent value. (i.e. if P sees x, it will never see an older version of x)



(a)



(b)

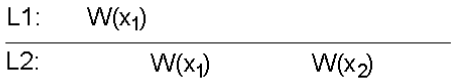
The read operations performed by a single process *P* at two different local copies of the same data store.

a) A monotonic-read consistent data store

b) A data store that does not provide monotonic reads.

Monotonic Writes

A write operation by a process on a data item x is completed before any successive write operation on x by the same process (i.e. a wrote operation on a copy of data item is performed only if that copy has been brought up to date by mean of any preceding write operation, even if taken place on another copy of x.)



(a)

(b)

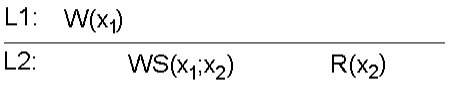
The write operations performed by a single process *P* at two different local copies of the same data store

a) A monotonic-write consistent data store.

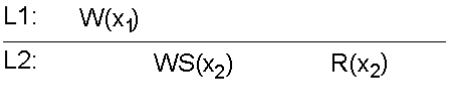
b) A data store that does not provide monotonic-write consistency.

Read Your Writes

The effect of a write operation by a process on data item x will always be seen a successive read operation on x by the same process. (i.e. a write operation is always completed before a successive read operation by the same process, no matter where the read takes place.)



(a)

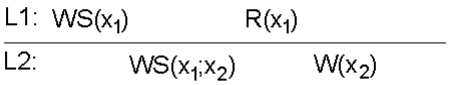


(b)

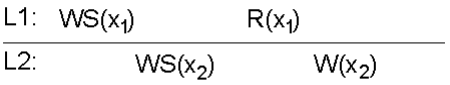
a) A data store that provides read-your-writes consistency.
b) A data store that does not.

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, it is guaranteed to take place on the same or a more recent value of x that was read. (i.e. any successive write operation by a process on x will be performed on a copy of x that's up to date with the value most recently read by that process.)



(a)



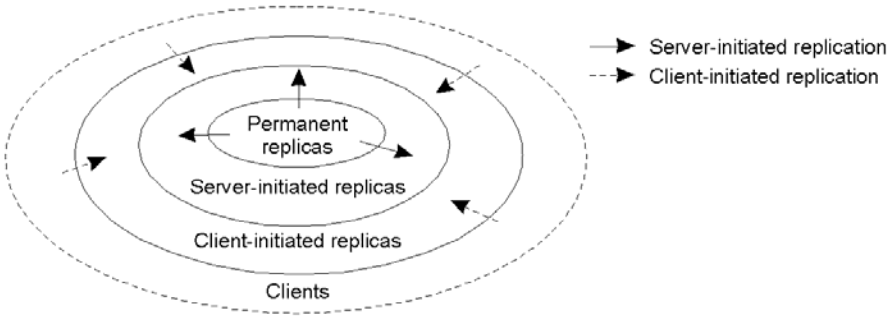
(b)

a) A writes-follow-reads consistent data store
b) A data store that does not provide writes-follow-reads consistency

Design Issue – Replica Placement

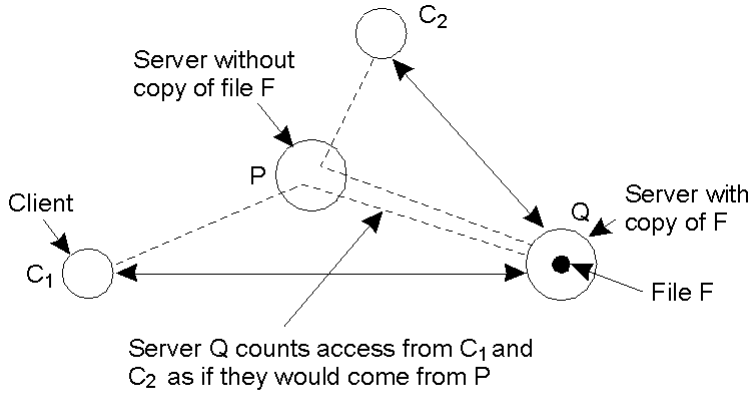
- So far, we discussed data-centric and client-centric consistency models.
- Replica placement
  1. Permanent Replicas
     - Typically small number of permanent replicas
     - Examples:
       - Replicated copies on a cluster of servers/workstations
       - Mirrored copies geographically distributed
  2. Server-initiated replicas
     - Cached copies of data pushed (created) by the server to enhance performance
     - Dynamic replication: When to create and delete replicas?
     - Dynamic replication is key for web hosting services (recall discussion on Akamai)
       - Note: static collection of servers; dynamic placement of replicated data on these servers close to demanding clients
  3. Client-initiated replicas
     - Also known as client caches
     - Local copy kept close to client to enhance read performance
       - Cache copy on client machine
       - copy on a storage device or a server on the same LAN as client
       - copy in the network (e.g. service provider network) serving the client

Replica Placement



The logical organization of different kinds of copies of a data store into three concentric rings.

Server-Initiated Replicas



Counting access requests from different clients.

Design Issue – Update Propagation

- An update must be propagated to other copies while maintaining consistency
- Issue 1:
  - Transfer **state** (data) to copies vs. execute **operation** at replicas
- Issue 2:
  - replica (or cache) **invalidation** vs. transfer data (**write-thru**)
  - Best for Low read-to-write ratio vs. best for high read-to-write ratio
- Issue 3:
  - Push vs. pull protocols
  - Tighter consistency vs. looser consistency
  - Server-initiated vs. client-initiated
  - See chart on next page
- Issue 4: Exploit multicast in push-based protocols if available

Pull versus Push Protocols

Issue	Push-based	Pull-based
State of server	List of client replicas and caches	None
Messages sent	Update (and possibly fetch update later)	Poll and update
Response time at client	Immediate (or fetch-update time)	Fetch-update time
Read-to-update ratio	Best if high	Best if low

A comparison between push-based and pull-based protocols in the case of multiple client, single server systems.

Consistency Protocols

A consistency protocol describes a specific implementation of a consistency model

Most common consistency models in practice: sequential consistency, weak consistency with synchronization variables, eventual consistency, atomic transactions

Terminology: (other texts)

Primary-back approach = Passive replication

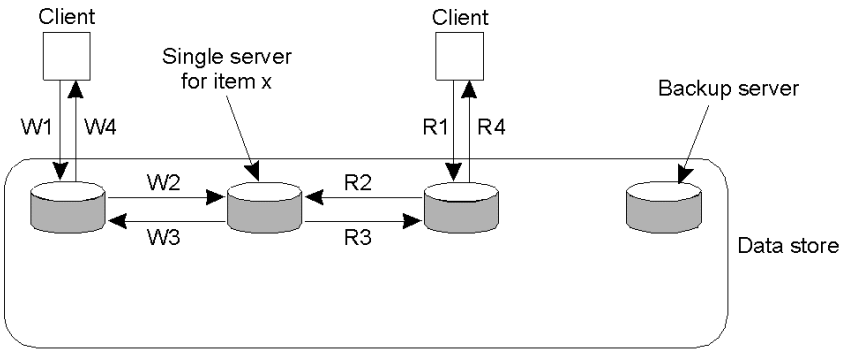
State-machine approach = Active Replication

Consistency Protocols Discussed

- Primary-based protocols
  - Remote-write protocol with no backup replica
  - Remote-write protocol: primary-backup approach (passive replication)
  - Local-write protocol with single migrating primary copy (no backup)
  - Local-write protocol with migrating primary copy and non-migrating backups (Primary-backup approach)
- Replicated-write protocols
  - Active replication
  - Quorum-based protocols
- Epidemic Protocols (eventual consistency)
- Cache-coherence protocols

Primary-based Approach

(no backup replication)



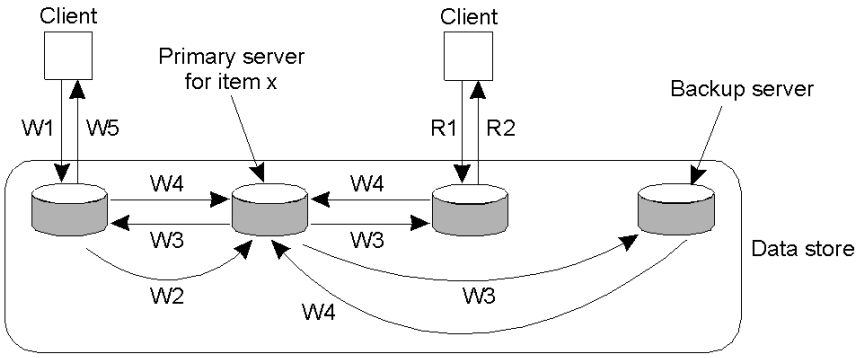
W1. Write request
W2. Forward request to server for x
W3. Acknowledge write completed
W4. Acknowledge write completed

R1. Read request
R2. Forward request to server for x
R3. Return response
R4. Return response

Primary-based remote-write protocol with a fixed server to which all read and write operations are forwarded.

Primary-Backup Protocol

(remote-write protocol)



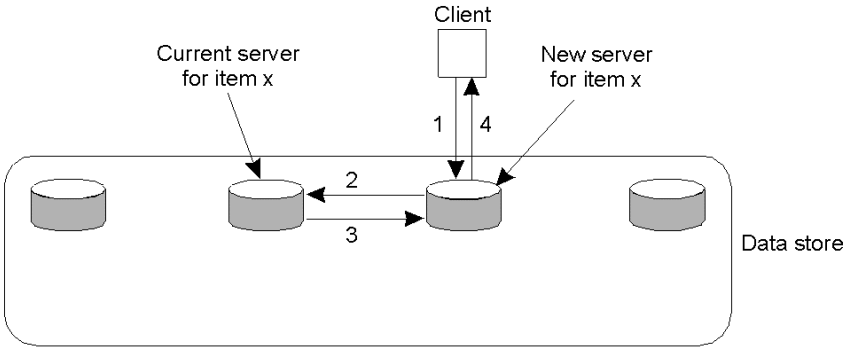
W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

Blocking vs. non-blocking update of backups

Local-Write Protocols

(no backup replica)

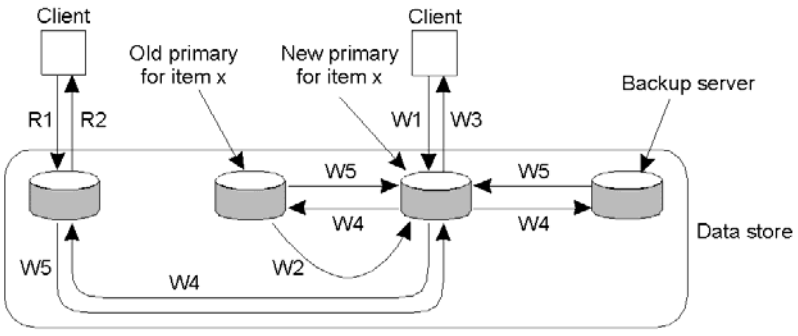


1. Read or write request
2. Forward request to current server for x
3. Move item x to client's server
4. Return result of operation on client's server

Primary-based local-write protocol in which a single copy is migrated between processes.

Local-Write Protocols

(P-B Approach with Migrating Primary)



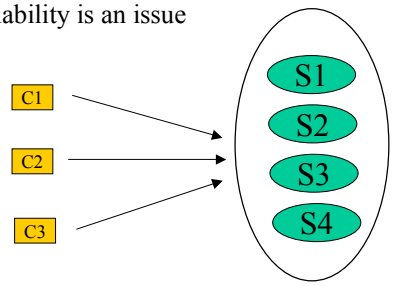
W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

Primary-backup protocol in which the primary migrates to the process wanting to perform an update.

Replicated Writes – Active Replication Approach

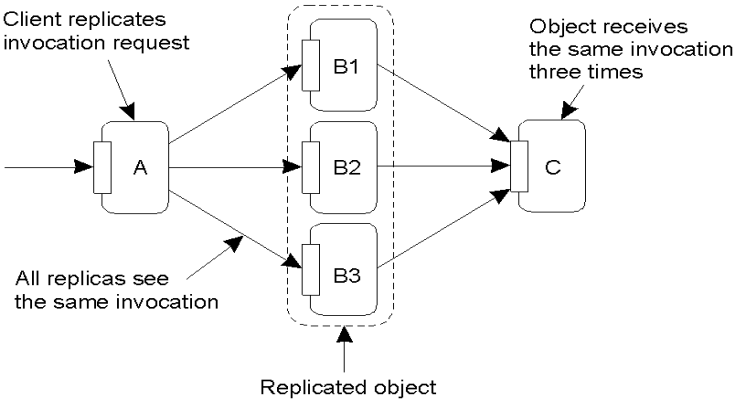
- Also known as state-machine replication
- Alternative to P-B approach
- Every replica is a primary, no backup
- Each replica has an associated process; each update is sent in the form of an operation to all replicas which execute the operation
- All updates (i.e. operations) need to be carried out in the same order at all replicas → the state of all replicas are updated in the same order
- totally-ordered multicast is the underlying mechanism for active replication
- Scalability is an issue



Clients Replicated Servers

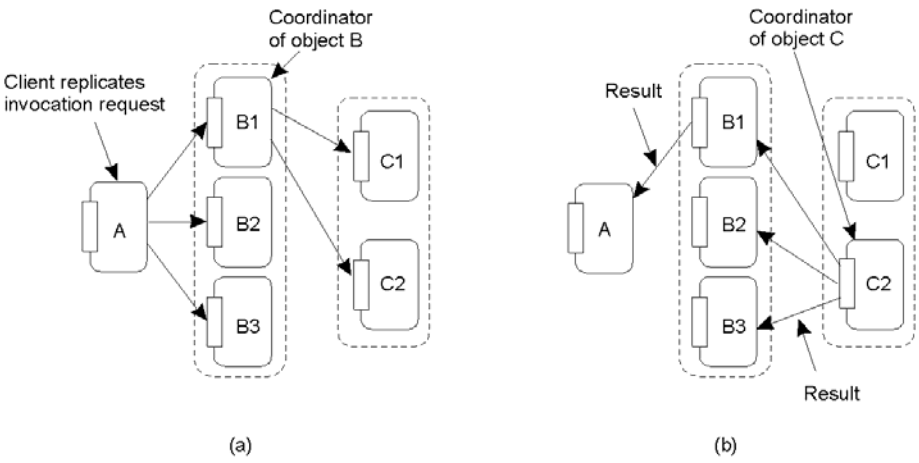
Active Replication

(problem of replicated invocations)



- The problem of replicated invocations:
  - A invokes B which in turn invokes C
  - What if B is replicated? Multiple invocations of C for the same operation

Active Replication



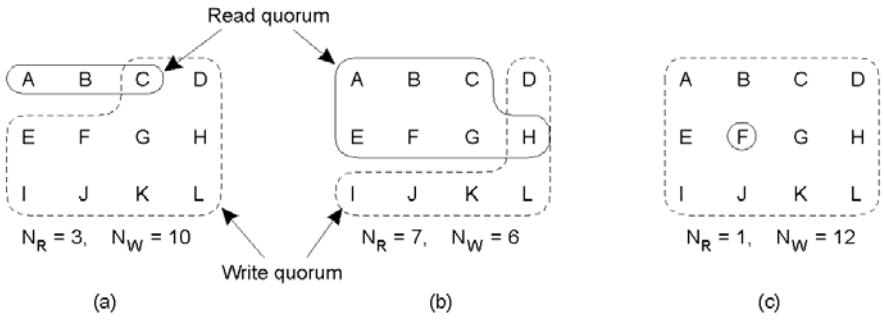
a) Forwarding an invocation request from a replicated object.
b) Returning a reply to a replicated object.

Replicated Writes- Quorum-Based Protocols

- Use voting: a client requests and acquires replies from multiple clients before reading or writing
- e.g., N replicated server, a client must contact at least one half plus one servers to perform a read or update operation.

- Quorum is more general than simple majority:
  - N replicas; Nr read quorum servers, Nw write quorum servers
  - Nr and Nw are subject to:
    - Nr+Nw > N (prevent r/w conflicts)
    - Nw > N/2 (prevent w/w conflicts)
- Many variations

Quorum-Based Protocols



Three examples of the voting algorithm:

a) A correct choice of read and write set

b) A choice that may lead to write-write conflicts because Nw <= N/2

c) A correct choice, known as ROWA (read one, write all)

Epidemic Protocols

- Ensures eventual consistency
- Epidemic spreading of updates, propagates updates to all replicas efficiently
- Highly scalable approach
- Does not solve any update conflicts directly

- Terminology:
  - Infective – a server that holds an update and it is willing to spread
  - Susceptible – a server that has not been updated
  - Removed – a server that is not willing or able to update

Epidemic Protocols

- Anti-entropy Propagation Approach:
  - A server P picks another server Q at random, three approaches in exchanging updates:
    1. P only pushes its updates to Q
    2. P only pulls in new updates from Q
    3. P and Q exchange updates (push-pull)

- Gossip Protocol:
  - A specific variant to make a number of servers infective after an update
  - If server P has just been updated for data item x,
    - Contact an arbitrary server Q and push update to Q
    - If Q was already updated, P may lose interest with some probability
  - Good way of rapidly spreading updates
  - Note: cannot guarantee that all servers will actually be updated