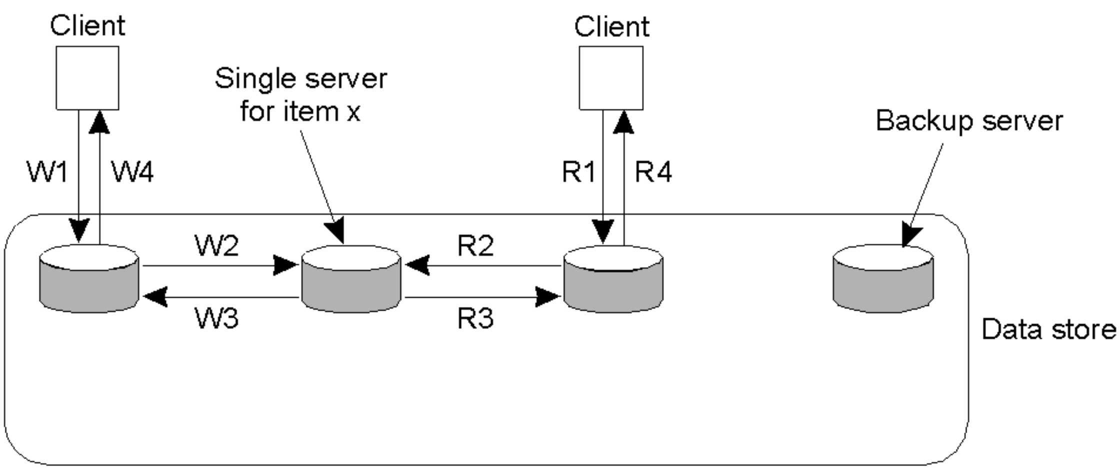
Implementation Issues

- Two techniques to implement consistency models
  - Primary-based protocols
    - Assume a primary replica for each data item
    - Primary responsible for coordinating all writes
  - Replicated write protocols
    - No primary is assumed for a data item
    - Writes can take place at any replica

Remote-Write Protocols

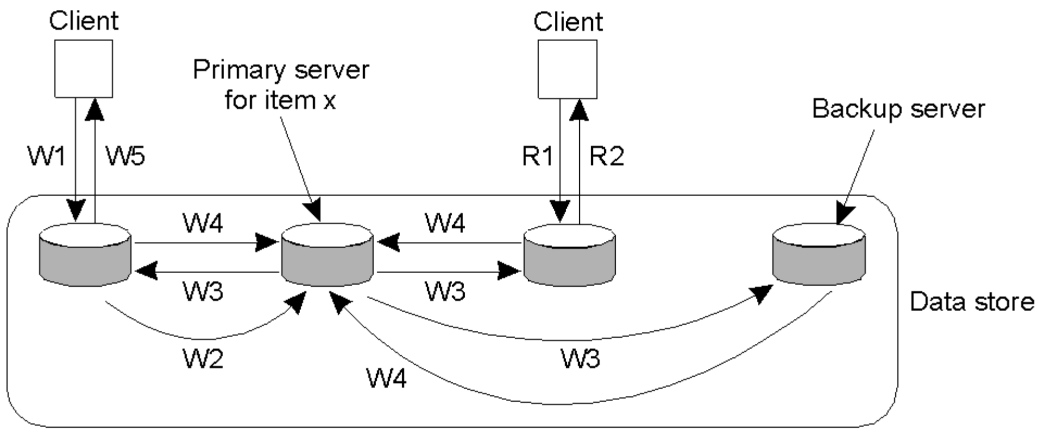


W1. Write request
W2. Forward request to server for x
W3. Acknowledge write completed
W4. Acknowledge write completed

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

- Traditionally used in client-server systems

Remote-Write Protocols (2)

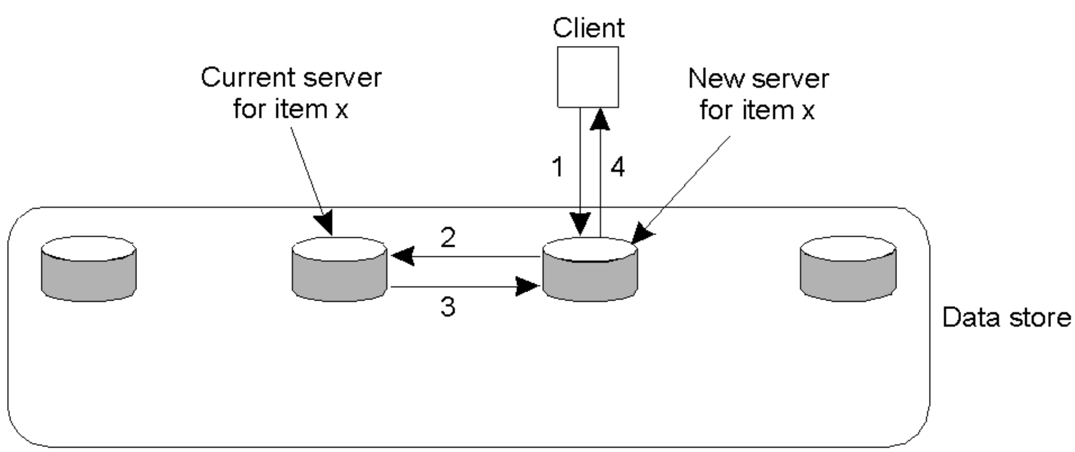


W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

- Primary-backup protocol
  - Allow local reads, sent writes to primary
  - Block on write until all replicas are notified
  - Implements sequential consistency

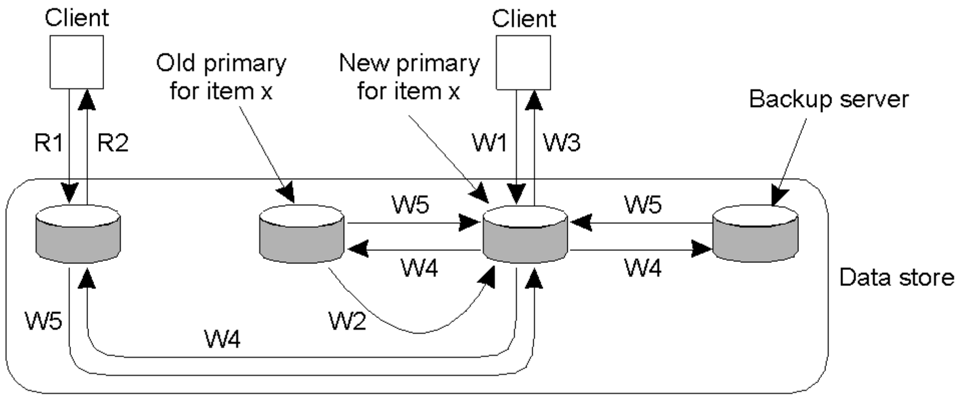
Local-Write Protocols (1)



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.
  - Limitation: need to track the primary for each data item

Local-Write Protocols (2)



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

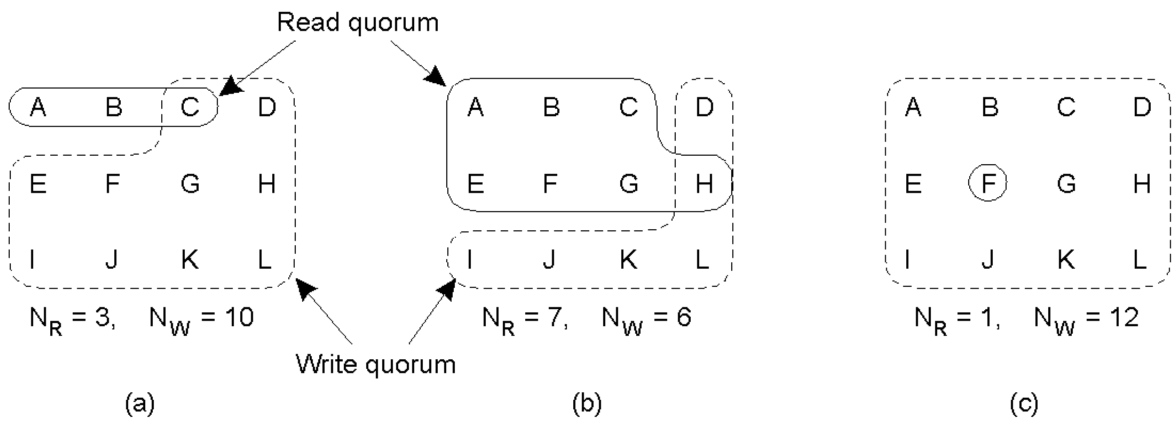
R1. Read request
R2. Response to read

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

Replicated-write Protocols

- Relax the assumption of one primary
  - No primary, any replica is allowed to update
  - Consistency is more complex to achieve
- Quorum-based protocols
  - Use voting to request/acquire permissions from replicas
  - Consider a file replicated on *N* servers
    - $N_R + N_W > N \qquad N_W > N/2$
  - Update: contact at least (N/2+1) servers and get them to agree to do update (associate version number with file)
  - Read: contact majority of servers and obtain version number
    - If majority of servers agree on a version number, read

Gifford's Quorum-Based Protocol



- 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

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

Replica Management

- Replica server placement
  - Web: geophically skewed request patterns
  - Where to place a proxy?
    - K-clusters algorithm
- Permanent replicas versus temporary
  - Mirroring: all replicas mirror the same content
  - Proxy server: on demand replication
- Server-initiated versus client-initiated

Content Distribution

- Will come back to this in Chap 12
- CDN: network of proxy servers
- Caching:
  - update versus invalidate
  - Push versus pull-based approaches
  - Stateful versus stateless
- Web caching: what semantics to provide?

Final Thoughts

- Replication and caching improve performance in distributed systems
- Consistency of replicated data is crucial
- Many consistency semantics (models) possible
  - Need to pick appropriate model depending on the application
  - Example: web caching: weak consistency is OK since humans are tolerant to stale information (can reload browser)
  - Implementation overheads and complexity grows if stronger guarantees are desired