CSCI 5105

Instructor: Abhishek Chandra

Today

- Data Consistency
  - Consistency Protocols

Consistency Protocols

- Implementation of a consistency model
  - How do we order operations according to a consistency model?
  - How are multiple writes applied and propagated to different replicas?

Consistency Protocols

- Ordering-based Consistency Protocols
  - Maintain desired ordering of operations
- Continuous Consistency Protocols
  - Bound numerical deviation or staleness
- Client-Centric Consistency Protocols
  - Provide consistent view to individual clients

Ordering-based Consistency Protocols

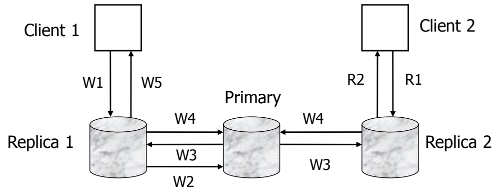
- Primary-based Protocols
  - Each data item has a primary replica
- Replication-based Protocols
  - Operations can be carried out at multiple replicas

Primary-based Protocols

- Each data item has a primary replica
- All writes are applied to and coordinated by the primary
- Two types:
  - Remote-Write: The primary is fixed and remote
  - Local-Write: The primary is copied locally before applying writes

Remote-Write

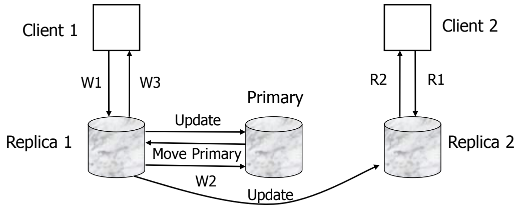
- Reads done locally, writes sent to primary
- A write is complete only when all backups have updated



- Problems?

Local-Write

- Primary is migrated before performing writes
  - Multiple copies of data item: reads done locally
  - Updates propagated to other replicas
  - Example: Mobile computing



Replicated-Write Protocols

- No single primary copy
- Writes can be performed at multiple replicas
- Two types:
  - Active Replication: All operations are forwarded to all replicas
  - Quorum-based: Operations are forwarded to a subset of all replicas

Active Replication

- All write operations are propagated to all replicas
  - Must be applied in the same order
- Need total ordering of writes
  - Use Lamport timestamps
  - Central sequencer

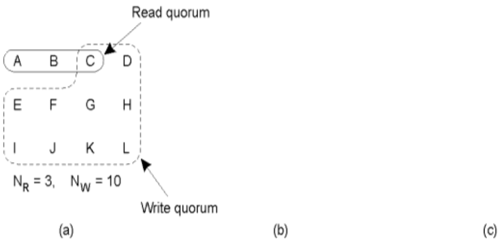
Quorum-Based Protocols

- Operations are sent to a subset of replicas
- Maintaining consistency
  - Use voting
  - If a quorum (e.g.: majority) agrees, then, consistency is maintained
  - Write: Apply write only if majority of replicas agree on the update
  - Read: Perform read from the latest version among a majority of replicas

Gifford's Quorum-Based Protocol

- N replicas
- Read quorum: Need N_R replicas to agree
- Write quorum: Need N_W replicas to agree
- Need to satisfy:
  - $N_R + N_W > N$ (Avoid read-write conflicts)
  - $N_W > N/2$ (Avoid write-write conflicts)

Gifford's Quorum-Based Protocol



Continuous Consistency: Bounding Numerical Deviation

- Each update originates at one replica
  - Each update has a numerical value (weight)
- Each replica i maintains
  - TW[i,i]: Total weight of its local updates
  - TW[i,j]: Total weight of other replicas' updates
  - $TW_i[k,j]$: View of other replicas' total weights
- Epidemic protocol:
  - Update total weight of replica k if exceeds bound
  - Update local view of k's total weights

Continuous Consistency: Bounding Staleness

- Each replica i maintains a real-time vector clock
  - $RVC_i[k]=t$
  - t is the time of last update on k seen by i
- Pull-based protocol:
  - If (curr-time - $RVC_i[k]$)> δ then pull update from replica k

Client-Centric Consistency

- Want to propagate updates in a client-centric manner
- Each write assigned a global identifier at the origin server
- For each client, two sets of writes:
  - Read set: Writes relevant to the client's reads
  - Write set: Writes performed by the client
- Different models implemented using these sets
  - Updates from either set either propagated locally or client requests are sent to an updated server

Implementing Different Consistency Models

- Monotonic reads:
  - When a client issues a read, the local replica will first update with the Read set of client
  - Client’s Read set is updated with any subsequent local writes that affect the Read operation
- Monotonic writes:
  - When a client issues a write, the local replica will first update with the Write set of client
  - The write is added to the client’s Write set

Optimizations

- Problem 1: Read and write sets can become very large
  - Session: Group of read/write operations when user is active
  - Discard reads/writes from earlier sessions
- Problem 2: The set representation is wasteful
  - Use vector timestamps for the write operations
  - Only pass around vector timestamps (not whole set)