CS 194: Lecture 8

Consistency and Replication

Scott Shenker and Ion Stoica
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720-1776

1

Today's Lecture

- Motivation
- Data-centric models of consistency
- Consistency mechanisms
- Eventual consistency
- Mechanisms for eventual consistency

2

Next Lecture

- Client-centric notions of consistency
- Bayou system
- Causally-consistent lazy replication

3

Why Replicate Data?

- High volume
- Low latency
- High availability

4

Examples

- DNS: caching enhances scalability
- Web: Akamai, etc.
- Distributed file systems: Coda, Bayou, etc.

5

Why Not Replicate?

- Must keep replicas transparent to clients
  - Clients operate on logical objects
  - Operations executed on physical objects
- Therefore, must keep replicas consistent

6

Inherent Tension

- If require all copies to be identical all the time, then can only have one copy
- If have multiple copies, must tolerate some degree of inconsistency
- The weaker the consistency requirement, the easier it is to build scalable solutions
- If consistency requirement is too strong, replication might hurt performance, not help it

7

Models of Consistency

- Described in terms of the data in various locations
- Next lecture we will describe this in terms of the clients reading the data
- These are two very different perspectives

8

Not Transactions!

- We are considering independent operations
- This means that reading a value and then writing based on that value appears as two independent operations
- Weaker requirement on consistency

9

Strict Consistency

- Any read on a data item x returns a value corresponding to the most recent write of x
- Problems:
  - "Most recent" only has meaning with perfectly synchronized clocks
  - Perfect synchronization physically impossible, unless only one replica
- When might you want this?
  - Auction?

10

Linearizable

- Operations executed in a sequential order dictated by a set of timestamps
- Timestamps within a process are time-ordered
- When might this be appropriate?
  - Formal analysis?

11

Sequential Consistency

- Operations appear in the same sequential order at all replicas
- Operations from the same client are processed in the order they were submitted by that process

12

Causal Consistency

- Writes that are causally related must be seen by processes in the same order. Concurrent writes may be seen in a different order on different machines.
- Similar to our notions of vector timestamps

13

FIFO Consistency

- Writes done by a single process are seen by all processes as occurring in the order in which they were issued

14

Focus on Sequential Consistency

- Good compromise between utility and practicality
  - We can do it
  - We can use it
- Stricter: too hard
- Less strict: replicas can disagree forever

15

Mechanisms for Sequential Consistency

- Local cache replicas
- Primary-based replication protocols
- Replicated-write protocols
- Cache-coherence protocols [won't cover]

16

Local Cache

- Primary copy of data (e.g., web server)
- Client reads data
- Client (or proxy cache on its behalf) saves copy of data for a short time (TTL)
- Reads issued during the TTL get cached copy
- What form of consistency is that?

17

Variety of Cache Updates

- Pull: client asks for update
- Push: server pushes updates to all sites that have cached copies
- Leases: Push for TTL, after that pull

18

Push vs Pull

- Push: server keeps state about all cached copies
  data sent even when unneeded
  response time low
- Pull: server keeps no state
  data only sent when needed
  response time can be higher

19

Why Not Multicast for Caches?

- Two multicast groups for each data item x
  - Invalidation group
  - Update group
- When x is updated, server sends messages to groups
  - Data to update group, only notice of update to invalidation group
- When x is cached somewhere, that replica joins one of the multicast groups
- Properties:
  - No state in server
  - Reliability of update delivery is hard

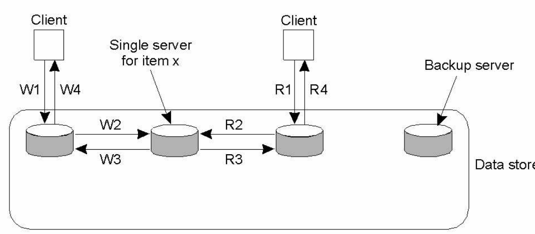
20

The Boring Methods

- Primary-based protocols
- Local write vs remote write
- Local read vs remote read
- Backup vs not

21

Primary with Remote Read/Write

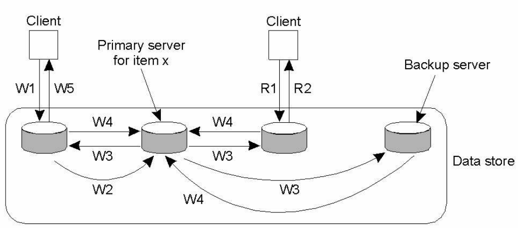


W1. Write request
W2. Forward request to server for x
W3. Acknowledge write completed
W4. Acknowledge write completed

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

22

Primary Remote-Write w/Backup

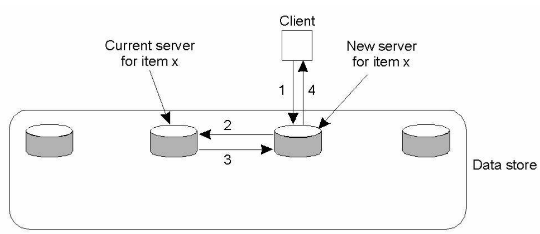


W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

23

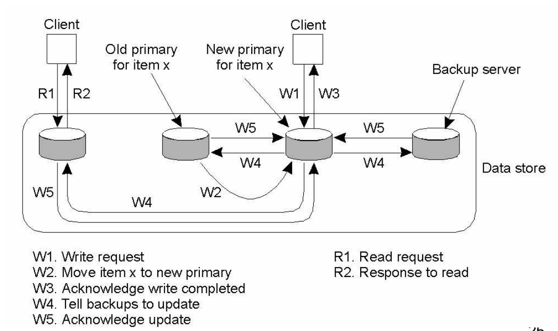
Primary-Based Local-Write



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

24

Primary-Backup with Local Writes



25

Slightly More Interesting

- Distributed Writing
- No primary copy!

26

Quorum-based Protocols

- Assign a number of votes V(I) to each replica I
- Let V be the total number of votes
- Define VR=read quorum, VW=write quorum
- VR+VW > V (why?)
- VW > V/2 (why?)

27

Results

- Only one writer at a time can achieve write quorum
- Every reader sees at least one copy of the most recent read (takes one with most recent version number)

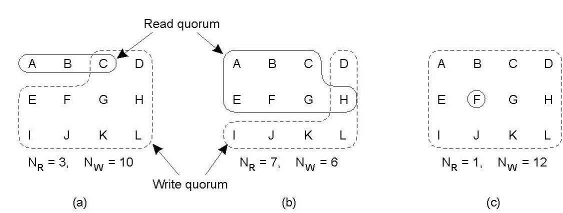
28

Possible Policies

- ROWA: VR=1, VW=N
  - Fast reads, slow writes (and easily blocked)
- RAWO: VR=N, VW=1
  - Fast writes, slow reads (and easily blocked)
- Majority: VR=VW=N/2+1
  - Both moderately slow, but extremely high availability
- See Gifford's paper

29

Quorum



30

Scaling

- None of these protocols scale
- To read or write, you have to either
  - (a) contact a primary copy
  - (b) contact over half of the replicas
- All this complication is to ensure sequential consistency
- Can we weaken sequential consistency without losing some important features?

31

What Consistency Do We Want?

- Sequential consistency requires that at every point, every replica has a value that could be the result of the globally-agreed sequential application of writes
- This does not require that all replicas agree at all times, just that they always take on the same sequence of values
- Why is this so important?
- Why not allow temporary out-of-sequence writes?

32

What Consistency Do We Want? (2)

- Note: all forms of consistency weaker than sequential allow replicas to disagree forever
- We want to allow out-of-order operations, but only if the effects are temporary

33

Eventual Consistency

- If all updating stops then eventually all replicas will converge to the identical values
- Furthermore, the value towards which these values converge has sequential consistency of writes.

34

Implementing Eventual Consistency

- All writes eventually propagate to all replicas
- Writes, when they arrive, are applied in the same order at all replicas
  - Easily done with timestamps

35

Update Propagation

- Rumor or epidemic stage:
  - Attempt to spread an update quickly
  - Willing to tolerate incompletely coverage in return for reduced traffic overhead
- Correcting omissions:
  - Making sure that replicas that weren't updated during the rumor stage get the update

36

Rumor Spreading: Push

- When a server P has just been updated for data item x, it contacts some other server Q at random and tells Q about the update
- If Q doesn't have the update, then it (after some time interval) contacts another server and repeats the process
- If Q already has the update, then P decides, with some probability, to stop spreading the update

37

Performance of Push Scheme

- Not everyone will hear!
  - Let S be fraction of servers not hearing rumors
  - Let M be number of updates propagated per server
- S= exp{-M}
- Note that M depends on the probability of continuing to push rumor.

38

Pull Schemes

- Periodically, each server Q contacts a random server P and asks for any recent updates
- P uses the same algorithm as before in deciding when to stop telling rumor
- Performance: better (next slide), but requires contact even when no updates

39

Variety of Schemes

- When to stop telling rumor: (conjectures)
  - Counter: S ~ exp{-M^3}
  - Min-counter: S ~ exp{-2^M}
- Controlling who you talk to next
  - Can do better
- Knowing N:
  - Can choose parameters so that S << 1/N
- Spatial dependence

40

Finishing Up

- There will be some sites that don't know after the initial rumor spreading stage
- How do we make sure everyone knows?

41

Anti-Entropy

- Every so often, two servers compare compete datasets
- Use various techniques to make this cheap
- If any data item is discovered to not have been fully replicated, it is considered a new rumor and spread again

42

We Don’t Want Lazarus!

- Consider server P that does offline
- While offline, data item x is deleted
- When server P comes back online, what happens?

43

Death Certificates

- Deleted data is replaced by a death certificate
- That certificate is kept by all servers for some time T that is assumed to be much longer than required for all updates to propagate completely
- But every death certificate is kept by at least one server forever

44

Next Lecture

- Client-centric notions of consistency
- Bayou system
- Causally-consistent lazy replication

45