CS 194: Lecture 9

Bayou

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

Don't Worry, Reality is on its Way!

- Theory part of course is almost over
- After midterm, will talk more about real systems
- Are currently revising the lecture plan

Agenda

- Review of last lecture
- A really bad joke
- The Bayou system

Purpose of Review

- Bring all our timestamps up to current
- If you don't understand something, please ask
- If you want an example, ask (and I'll try)

Transactions, then Replication

- Transactions:
  - One copy of data
  - Transactions = set of operations
  - Multiple transactions, each over many data items
  - Locking policies
- Replication:
  - Many copies of data
  - Multiple operations
  - Not focusing on transactions, replication by itself is hard enough

Replication

- Why replication?
  - Volume, Proximity, Availability
- What not replication?
  - Replicas must be kept consistent (why?)
  - Overhead of keeping them consistent sometimes outweighs benefit of replication

Many Kinds of Consistency

- Strict
- Linearizable
- Sequential (~serializable)
- Causal
- FIFO

7

Examples

- What are some examples of replicated systems?
- What kinds of consistency do they offer?

8

Focus on Sequential Consistency

- Weakest model of consistency in which data items had to converge to the same value everywhere

9

Consistency Mechanisms

- Local caching: push/pull/lease
  - Role of multicast in making push easier
  - Often under client control, consistency can be tuned to user needs
- Primary copy: serialize at master
  - Local or remote reads (only remote reads support transactions)
- Quorums:
  - Assign votes to replicas
  - Can only read/write when have read/write quorum

10

Scaling

- None of these protocols scale
- To read or write, you have to either
  - Contact a primary copy
  - Contact over half the replicas
- Gray et al. model the scaling behavior of distributed trans.:
  - Deadlock ~n^3

11

Is Sequential Consistency Overkill?

- Sequential consistency requires that at each stage in time, the operations at a replica occur in the same order as at every other replica
- Ordering of writes causes the scaling problems!
- Why insist on such a strict order?

12

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.

13

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

14

Update Propagation

- Rumor or epidemic stage:
  - Attempt to spread an update quickly by contacting peers
  - Willing to tolerate incompletely coverage in return for reduced traffic overhead
  - Push/Pull distinction
- Correcting omissions:
  - Making sure that replicas that weren't updated during the rumor stage get the update
  - Anti-entropy exchanges: comparison of full databases
- Death certificates: needed for deleted items

15

Bayou

16

Why Should You Care about Bayou?

- Changed the paradigm
- Subset incorporated into next-generation WinFS
- Done by my friends
  - I always thought it was a silly project…..

17

System Assumptions

- Early days: nodes always on when not crashed
  - Bandwidth always plentiful (often LANs)
  - Never needed to work on a disconnected node
  - Nodes never moved
  - Protocols were "chatty"
- Now: nodes detach then reconnect elsewhere
  - Even when attached, bandwidth is variable
  - Reconnection elsewhere means often talking to different replica
  - Work done on detached nodes

18

Disconnected Operation

- Challenge to old paradigm
  - Standard techniques disallowed any operations while disconnected
  - Or disallowed operations by others
- But eventual consistency not enough
  - Reconnecting to another replica could result in strange results
    - E. g., not seeing your own recent writes
  - Merely letting latest write prevail may not be appropriate
  - No detection of read-dependencies
- What do we do?

19

Bayou

- System developed at PARC in the mid-90's
- First coherent attempt to fully address the problem of disconnected operation
- Several different components
- But first, why did they call it "Bayou"?

20

What's a Bayou?

- A body of water, such as a creek or small river, that is a tributary of a larger body of water.
- A sluggish stream that meanders through lowlands, marshes, or plantation grounds.

21

Possible Explanations*

- Bayous are ubiquitous, and Bayou supports ubiquitous computation (ubicomp)
- Bayou provides "fluid" replication
- Allows operation when you are "bayou self"
- Pronounced Bi-U, which makes it Ubi spelled backwards
- *All stolen from Alper Mizrak (UCSD)

22

Homework for Next Class

- Email me one bad joke (which I can use in my lectures)
- New intermission tradition:
  - Introduce yourself
  - Tell a joke
- Best joke (according to me) gets a pound of chocolate
- No joke, and you flunk.…

23

Motivating Scenario: Shared Calendar

- Calendar updates made by several people
  - e.g., meeting room scheduling, or exec+admin
- Want to allow updates offline
- But conflicts can't be prevented
- Two possibilities:
  - Disallow offline updates?
  - Conflict resolution?

24

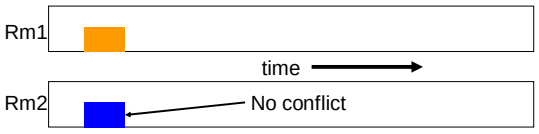
Conflict Resolution

- Replication **not** transparent to application
  - Only the application knows how to resolve conflicts
  - Application can do record-level conflict detection, not just file-level conflict detection
  - Calendar example: record-level, and easy resolution
- Split of responsibility:
  - Replication system: propagates updates
  - Application: resolves conflict
- Optimistic application of writes requires that writes be "undo-able"

25

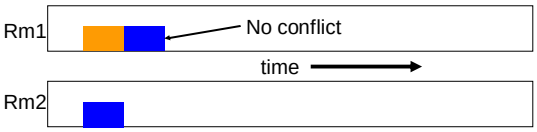
Meeting room scheduler

Reserve same room at same time: conflict
Reserve different rooms at same time: no conflict
Reserve same room at different times: no conflict
Only the application would know this!



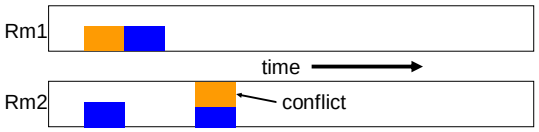
26

Meeting Room Scheduler



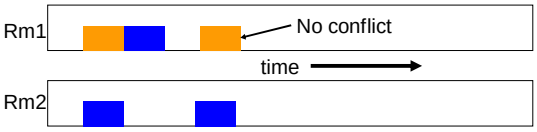
27

Meeting Room Scheduler



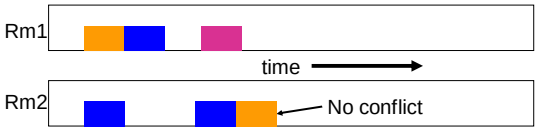
28

Meeting Room Scheduler



29

Meeting Room Scheduler



30

Other Resolution Strategies

- Classes take priority over meetings
- Faculty reservations are bumped by admin reservations
- Move meetings to bigger room, if available
- Point:
  - Conflicts are detected at very fine granularity
  - Resolution can be policy-driven

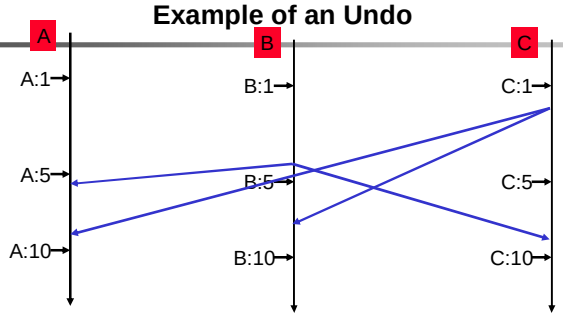
31

Rolling Back Updates

- Keep log of updates
- Order by some timestamp
- When a new update comes in, place it in the correct order and reapply log of updates
- Need to establish when you can truncate the log
- Requires old updates to be “committed”, new ones tentative

32

Example of an Undo



A will undo update from B, apply C and then B

33

Two Basic Issues

- Flexible update propagation
- Dealing with inconsistencies

34

Flexible Update Propagation

Requirements:

- Can deal with arbitrary communication topologies
- Can deal with low-bandwidth links
- Incremental progress (if get disconnected)
- Eventual consistency
- Flexibile storage management
- Can use portable media to deliver updates
- Lightweight management of replica sets
- Flexible policies (when to reconcile, with whom, etc.)

35

Update Mechanism

- Updates timestamped by the receiving server
- Writes from a particular server delivered in order
- Servers conduct anti-entropy exchanges
- State of database is expressed in terms of a timestamp vector
- By exchanging vectors, can easily identify which updates are missing

36

Replica Creation/Deletion

- Because updates are eventually “committed” you can be sure that certain updates have been spread everywhere
- By including replica creation/deletion as a normal “update” you can know which replicas are know to exist by everyone and which are known to be deleted by everyone
- Can discard “death certificates” when the deletion update is “committed”

37

Dealing with Inconsistencies

- Session guarantees
- Conflict detection (update dependencies)
- Conflict resolution (already discussed)

38

Session Guarantees

- When client move around and connects to different replicas, strange things can happen
  - Updates you just made are missing
  - Database goes back in time
  - Etc.
- Design choice:
  - Insist on stricter consistency
  - Enforce some “session” guarantees

39

Read Your Writes

- Every read in a session should see all previous writes in that session

40

Monotonic Reads and Writes

- A later read should never be missing an update present in an earlier read
- Same for writes

41

Writes Follow Reads

- If a write W followed a read R at a server X, then at all other servers
  - If W is in Y’s database then any writes relevant to R are also there

42

Supporting Session Guarantees

- Responsibility of “session manager”, not servers!
- Two sets:
  - Read-set: set of writes that are relevant to session reads
  - Write-set: set of writes performed in session
- Causal ordering of writes
  - Use Lamport clocks

43

Update Dependencies

- Needed for conflict detection
- Captured in write-set, read-sets
- But can be more general

44

Next Lecture

- Brewer’s conjecture about CAP
- Lynch’s proof of the CAP theorem
- Something else…..

45