CSE 486/586 Distributed Systems
Transactions on Replicated Data

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap

- Gossiping?
- Dynamo
  - Gossiping for membership and failure detection
  - Consistent hashing for node & key distribution
  - Object versioning for eventually-consistent data objects
  - Quorums for partition/failure tolerance
  - Merkel tree for resynchronization after failures/partitions
- Causal consistency?
- Eventual consistency?

Optimistic Quorum Approaches

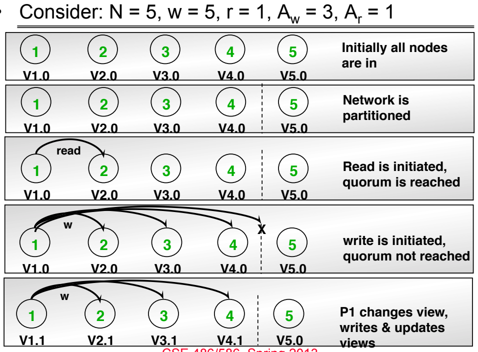
- An Optimistic Quorum selection allows writes to proceed in any partition.
- "Write, but don't commit"
  - Unless the partition gets healed in time.
- Resolve write-write conflicts after the partition heals.
- Optimistic Quorum is practical when:
  - Conflicting updates are rare
  - Conflicts are always detectable
  - Damage from conflicts can be easily confined
  - Repair of damaged data is possible or an update can be discarded without consequences
  - Partitions are relatively short-lived

View-based Quorum

- An optimistic approach
- Quorum is based on views at any time
  - Uses group communication as a building block (see previous lecture)
- We define thresholds for each of read and write :
  - W: regular writer quorum
  - R: regular reader quorum
  - A_w: minimum nodes in a view for write, e.g., $A_w > N/4$
  - A_r: minimum nodes in a view for read
  - E.g., $A_w + A_r > N/2$
- Protocol
  - Try regular quorum first; if it doesn't work, change the view. If the minimum is satisfied, then proceed.
  - A_w & A_r effectively determine which partition can proceed.

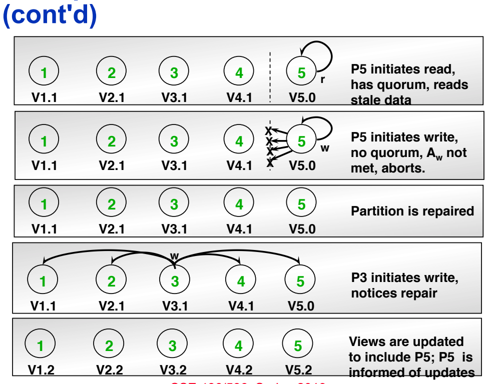
Example: View-based Quorum

- Consider: N = 5, w = 5, r = 1, A_w = 3, A_r = 1

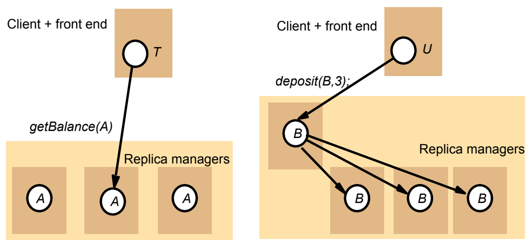


Example: View-based Quorum (cont'd)

•



Transactions on Replicated Data

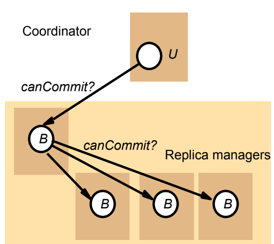


Correctness with Replication

- In a non-replicated system, transactions appear to be performed one at a time in some order. This is achieved by ensuring a *serially equivalent* interleaving of transaction operations.
  - Remember serial equivalence?
- How can we achieve something similar with replication? What do we want?
- One-copy serializability: The effect of transactions performed by clients on replicated objects should be the same as if they had been performed one at a time on a single set of objects (i.e., 1 replica per object).
  - Equivalent to combining serial equivalence + replication transparency/consistency

Revisiting Atomic Commit

- Participants need to agree on commit or abort.
- One way: use two level nested 2PC



Revisiting Atomic Commit

- In the first phase, the coordinator sends the canCommit? command to the participants, each of which then passes it onto the other RMs involved (e.g., by using view synchronous communication) and collects their replies before replying to the coordinator.
- In the second phase, the coordinator sends the doCommit or doAbort request, which is passed onto the members of the groups of RMs.

Primary Copy Replication

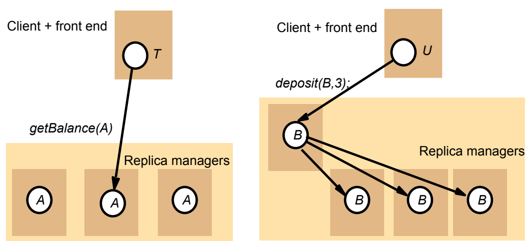
- All the client requests are directed to a single primary RM.
- Concurrency control is applied at the primary.
  - Let's assume we use strict two-phase locking.
- To commit a transaction, the primary communicates with the backup RMs and replies to the client.
- Communication is view synchronous totally-ordered group comm.
- One-copy serializability
  - View synchronous TO group comm.
  - Strict two-phase locking at the primary
- Disadvantage?
  - Performance is low since primary RM is bottleneck.

CSE 486/586 Administrivia

- PA2 grading done. Will post it today.
- Anonymous feedback form still available.
- Please come talk to me!

Read One/Write All Replication

- An FE (client front end) may communicate with any RM.
- Every write operation must be performed at all of the RMs.
- A read operation can be performed at any single RM.



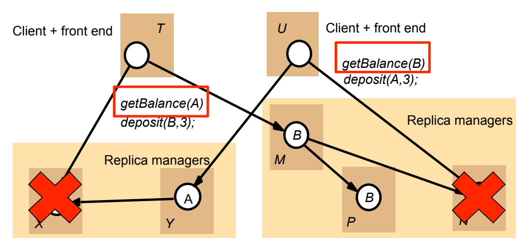
Read One/Write All Replication

- An FE (client front end) may communicate with any RM.
  - Use view synchronous TO group comm.
- Every write operation must be performed at all of the RMs
  - Each contacted RM sets a write lock on the object.
- A read operation can be performed at any single RM
  - A contacted RM sets a read lock on the object.
- Serial equivalence
  - Any pair of write operations will require locks at all of the RMs ➔ not allowed
  - A read operation and a write operation will require conflicting locks at some RM ➔ not allowed
- Consistency
  - Sequential consistency
- Disadvantage?
  - Failures block the system (esp. writes).

Available Copies Replication

- A client's read request on an object can be performed by any RM, but a client's update request must be performed across all available (i.e., non-faulty) RMs in the group.
- As long as the set of available RMs does not change, local concurrency control achieves one-copy serializability in the same way as in read-one/write-all replication.
- May not be true if RMs fail and recover during conflicting transactions.

Available Copies Approach



The Impact of RM Failure

- Assume that:
  - RM X fails just after T has performed getBalance; and
  - RM N fails just after U has performed getBalance.
  - Both failures occur before any of the deposit()'s.
- Subsequently:
  - T's deposit will be performed at RMs M and P
  - U's deposit will be performed at RM Y.
- The concurrency control on A at RM X does not prevent transaction U from updating A at RM Y.
- Solution: Must also serialize RM crashes and recoveries with respect to entire transactions.

Local Validation

- From T's perspective,
  - T has read from an object at X ➔ X must have failed after T's operation.
  - T observes the failure of N when it attempts to update the object B ➔ N's failure must be before T.
  - Thus: N fails → T reads object A at X; T writes objects B at M and P → T commits → X fails.
- From U's perspective,
  - Thus: X fails → U reads object B at N; U writes object A at Y → U commits → N fails.
- At the time T tries to commit,
  - it first checks if N is still not available and if X, M and P are still available. Only then can T commit.
  - If T commits, U's validation will fail because N has already failed.
- Can be combined with 2PC.
- Caveat: Local validation may not work if partitions occur in the network

Summary

- Optimistic quorum
- Distributed transactions with replication
  - One copy serialization
  - Primary copy replication
  - Read-one/write-all replication
  - Active copies replication

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC).