Replica Management

Slides are based on the book chapter from Distributed Computing: Principles and Paradigms (Chapter 7) by Tanenbaum and Van Steen
Replication Basics

• Data replication ➔ maintain multiple copies of data on separate servers/computers
  + Improves availability
  + Improves performance
  + Increased throughput

• Traditional Replication ➔ single-copy replication
  • Access to replicas are blocked till updates are consistent

• Optimistic Replication ➔ different approach to concurrency control
  • Allow data access without synchronous replication
    • Internet is slow and unreliable
    • Traditional replication algorithms scale poorly in WAN
    • Cooperative applications benefit from optimistic replication
Replication: Challenges

• Replica Server Placement
  – Selection of sites for replica servers
  – Given replica servers, where to place content

• What are the choices to propagate the changes to the replicas?

• How to implement changes on replicas to provide consistent view?
Replica Server Placement

• Optimization problem where $K$ out of $N$ servers are selected as replica site locations ($K \leq N$)

• Optimization criteria:
  – **Distance** between client and server
    • Bandwidth of the client-server links
    • Latency between clients and servers

• Dynamic identification of placement region
  – Region: collection of nodes accessing same content and internode latency is low
  – Select a node from a region as replica server
Replica placement

• Permanent Replicas
  – Initial set of replicas that comprise the data store
    • web sites have fixed pages
    • Web site mirroring
    • Databases can be distributed (shared-nothing architecture, where memory and disc are not shared)

• Server Initiated replicas
  – Copies of data store created dynamically to improve performance
    • Web hosting services replicate content near the edges of the network
    • Content distribution networks, like Akamai, maintains edge replicas
  – Server initiated replication can suffice if one can guarantee at least one replica, and not have any permanent replica

• Client initiated replica
  – Client caches
Content Distribution

• How to propagate updates to the content to replica servers?
  – Synchronous replication vs. asynchronous replication
    • Block other operations while replicating
      – Typical in transaction based systems, like Relational DBs
  – Content state vs. operations on content
    • State: transfer data from one copy to another
    • Operations: propagate update operation to other copies, and let
      the replica execute the operation to generate state
    • Can also propagate an invalidate message to mark a portion of
      the data that is not up to date
  – Push vs Pull
    • Server pushes the updates
    • clients or other replicas can periodically poll for changes
Implementation Issues

How to implement a consistent view of the data across the replicas?

- Bounded numerical deviation
- Bounded staleness deviation
- Bounded ordering deviations
  - Primary based protocols
  - Quorum based protocols
Primary based Protocols

• Remote Write (Single Master):
  – All write operations need to be forwarded to a single master server
  – Read operations can be read locally
  – Easy to order incoming writes in globally consistent order
  – Performance problem ➔ blocking operation since updates must propagate to all replicas before progress
  – What if we use non-blocking protocol? Fault tolerance issue
Primary based Protocols

- Local Write (Single Master but dynamic):
  - Primary copy migrates among replicas where writes are requested
    - Non-blocking protocol followed by update propagation after primary has completed local updates
  - Can be applied to mobile computing setting
    - Note: if primary is disconnected, then other nodes can only read, but cannot write
Replicated Write Protocols

• Multiple Masters: write operations can be carried out at multiple replicas \(\Rightarrow\) improved concurrency

• Active Replication
  – Operations are propagated
  – Problem is that all replicas must see the same order of operations \(\Rightarrow\) need a totally ordered multicast
  – Can use a sequencer \(\Rightarrow\) forward all operations to a central process that assigns unique id to an operation

• Quorum-based Protocols
  – Majority Voting based technique
    • clients request and acquire permission from multiple servers before reading or writing a replicated data item
Implementation: Client-centric consistency

• Note: Conflict detection and conflict resolution is an additional challenge

• Naïve approach:
  – Each write op $W$ is assigned a globally unique id
  – For each client, keep track of two sets of writes
    • Read set: writes relevant to the read operation
    • Write set: writes performed by a client
Implementation: Client-centric consistency

• Monotonic Read consistency
  – On receiving read request, server gets the read set to check all writes have been updated

• Monotonic Write consistency
  – Get the write set; execute all write operations before the write request

• Read your write consistency
  – Ensure that server where read request reaches has seen all preceding write operations (write set)

• Write follows read consistency
  – Bring selected server up to date with write ops in clients read set

**Problem:** Read and write sets can become large, and transferring across replicas will be costly

**Solution:**
Maintain sessions in applications; read and write sets are maintained only within sessions
Use efficient ways of representing the set information ➔ vector timestamps