Data Replication in Distributed Systems

Kai Shen

Data Availability

- Data-driven applications
  - Web search, online message board, collaborative document editing, … …
- Availability of data is important
  - After a node fails but before it restarts
  - What if a node/storage permanently fails and has to be replaced?

Data Replication

- Replication
  - Maintain multiple copies of data at replicated machines
  - Many clients read/write; each write is done at all servers; each read is served by one
- Benefits
  - Data availability and durability over failures
  - Performance
- Introduce consistency issues (illusion of single data copy)
  - Eventual convergence of replicated state
  - Consistent client views during concurrent client accesses

Replication Consistency

- Eventual convergence of replicated state
  - All replicas agree on the same data at stabilization
    ⇨ also called eventual consistency
  - Requirements: all updates reach all replicas; updates are performed in a consistent order at all replicas
- Consistent client views during concurrent client accesses– Sequential consistency: there exists a hypothetical sequential order of all operations from all clients that
  - returned value in a read operation is that written by last write in the sequential order,
  - the sequential order matches the order of operations from each client.

Replication in Shared-Memory Multiprocessor

- Shared-memory multiprocessor
  - Data is potentially replicated in processor caches
- How to achieve consistency?
  - Bus snooping (invalidate/update a local copy if the data is modified at another replica)
  - Operations from each client proceed serially in program order

Replication in Distributed Systems

- A group of distributed machines connected by network
  - Can we use the bus snooping?
- Synchronous writes to all replicas
  - A write does not return until it is committed at all replica
  - Lock up relevant data during the write using two-phase commit
- Performance and scalability?

Scalable Distributed Data Structures [Gribble et al. 2000]

- "Synchronous writes to all replicas" does not necessarily sacrifice performance/scalability.
  - In databases with complex semantics, we often have to lock too much data for a write, then block too many reads.
  - If data unit of each write is simply defined, over-locking is not a problem.
- Consider hash table with write(key,value) and read(key)
  - Simple semantics, well defined read/write data units
  - Powerful enough to support many data access semantics
- Abstract data management (and its scalability, availability, consistency) into simple data structures.

Replication in Distributed Systems

- Synchronous writes to all replicas with two-phase commits
- Primary-secondary replication
  - Writes initiated only at the primary, ordered and distributed to secondaries asynchronously ⇨ ensuring eventual consistency
  - Where are reads served?
    - Primary-only
    - All nodes (primary and secondaries)
- Consistency?
- Performance and scalability?

Weak Consistency

- Sequential consistency (strong consistency): if there exists a hypothetical sequential order of all operations such that
  - returned value in a read operation is that written by last write in the sequential order
  - the sequential order matches the order of operations from each client
- Often poor performance/scalability
- Weak consistency (beyond eventual consistency):
  - Consistent from a single client's point of view (read own writes, monotonic reads)
    ⇨ also called session consistency

Bayou [Terry et al. 1995]

- A group of loosely connected mobile devices
- Writes are spread around, eventually reaching everyone
- Eventual consistency
  - Writes are executed at all nodes, in the same order
  - Write (X) is tentatively executed; when write (Y) with earlier order arrives, X is undone, Y is done, and then X is redone
  - How do we know X is settled forever?
- Session consistency (read own writes, monotonic reads)
  - Writes are locally performed right away
  - Reads are locally performed

Porcupine [Saito et al. 1999]

- Tentative writes and undo are nasty
- If all writes are commutative, then they can be executed at different replicas in any order.
  - Adding/appending to a set
  - Timestamped total object overwrites
    - a write is performed if it follows all committed writes on the object
    - a write is ignored if it precedes any already committed write on the object
- Only satisfies the convergence of replicated state (eventual consistency), but not sequential consistency
- May realize session consistency if all operations from one client session is done at one server
- Implement a highly scalable replicated email system on a cluster of machines

Chain Replication [van Renesse and Schneider 2004]

- All replicas organized in a chain:
  - writes go to the head, and then flow through the chain, and replies are sent at the tail
  - reads performed at the tail
- Satisfy strong (sequential) consistency
- Add node at the tail
- Failure management: simple due to the clear structure
- Performance and scalability?

Replication/Consistency in GoogleDocs

- Acknowledgement: learned from Amal Fahad
- Replication
  - Document copy at server and clients
- Eventual consistency:
  - All updates are submitted to server who decides order of writes, maintains only authoritative copy
- Client consistency:
  - Two copies: screen copy and core copy
  - Core copy incorporates authoritative updates broadcast from server
  - Screen copy contains speculative, local updates
  - Screen copy may be overwritten by core copy when new updates arrive from the server (but not the other way around)