Chapter 1

Spanner: Globally distributed database

Spanner is a scalable, multi-version, globally-distributed, synchronously-replciated database.

External Consistency/Linearizability If transaction T1 commits before another transaction T2 starts, then T1's commit timestamp is smaller than T2's

1.1 Issues addressed

- Wanted strong consistency (which via CAP theorem meant less availability)
- Wanted ACID transactions
- Wanted schema
- Spanner is an example of NewSQL (SQL like model, but scalability and performance)
- Built for F1 (important app for Google). Was MySQL cluster, had big problems of resharding (done manually) and schema migration.
- Wanted easy geodistribution (coping with whole datacentre failure)
- Wanted to automate the process of replication

1.2 Main Ideas

- TrueTime API
- General-purpose transactions (ACID). Support different transaction types which are optimised e.g. non- blocking reads
  - Write transactions guarantees 'external consistency' (strong form of consistency)
    - Writes buffered.
    - Write locks acquired at commit time (when Paxos prepare is done.
    - But reducing availabilty for this (CAP) - block to wait out the uncertainty.
    - Writes must initiate the Paxos protocol at the leader.
  - Reads access state directly from the underlying tablet at any replcia that is sufficiently up-to-date.
  - Read-Write - requires locks
  - Read-Only - lock free
    - Requires declaration before start of transaction
    - Reads information that is up-to-date

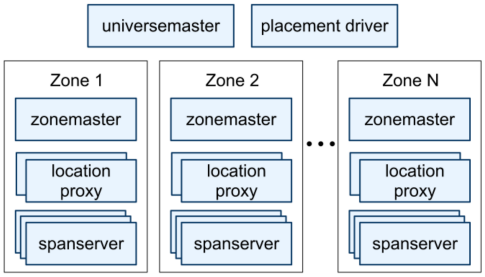
  - Snapshot-Read - read information from past by specifying timestamp or bound
    * Lock free (non-blocking)
    * Globally consistent
    * Use specific timestamp from past or timestamp bound so that data until that point will be read

- Atomic schema changes
- Temporal database. Transaction serialization via global timestamps
- Schematised, semi-relational (tabular) data model
  - SQL-like query interface
- Data is versioned. Each version is automatically timestamped at commit time - while locks are held
- Data chunks - directory/bucket
  - Unit of data movement and for defining replication properties
  - Split into fragments
  - Set of contiguous keys that share a common prefix.
  - Locality encoded in to it - pick keys to get better locality
- Shards data across many sets of Paxos groups
  - Want multiple groups to be able to partition data so you have smaller set of nodes to run Paxos on so its quicker.
  - Also, with multiple paxos groups they can have different ways of replication (nr replicas, location, etc).
- Layering of how transactions are executed.
  - If transaction can be done in one Paxos group then just run on that group.
  - When the transaction involves multiple paxos groups, then use the top layer (which uses strict two phase commit). This means users can do cross-row transactions
- Automatic resharding, rebalancing, failure response
- Globally distributed for high availability and geographic locality.

1.2.1 TrueTime API

- Global timestamps
- Exposes uncertainty in time and guarantees a bound on it
- Timestamp as a range
- Truetime timestamp getting is local, but these timestamps can be globally ordered (by reasoning about clock uncertainty/clock skew and waiting out the uncertainty to avoid overlapping timestamps)
- TrueTime has masters that globally synchronize and decide on the global bound.
- If network partition you lose the synchronization between the truetime masters. So must assume the clock skew increases over time, so over time the wait time will increase to where the system is very slow. (smaller clock uncertainty, larger throughput rate)
- Uses GPS and atomic clocks to get accurate time.

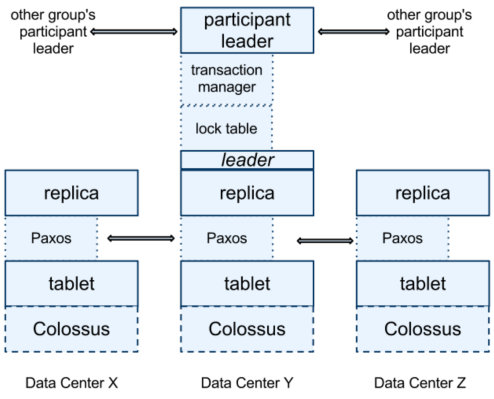
1.3 Additional points



- Universe = Spanner deployment
- Zone = unit of administrative deployment; unit of physical isolation
- Zonemaster = assigns data to spanservers
- Spanservers = serve data to clients
- Location proxies = used by clients to locate the spanservers assigned to serve their data.
- Universe master = console that displays status info about all zones for interactive debugging
- Placement driver = handles automated movement of data across zones
- Tablet: data structure. Bag of key-value mappings (key, timestamp) -¿ string. A container that may encapsulate multiple partitions of the row space, hence its possible to colocate mutiple directories that are frequently accessed together. Tablets are stored on top of the Colossus distributed file system.
- Paxos state machine on each tablet.
- Long-lived leaders with time-based leader leases. If network partition happens, Paxos waits for 10 secs to get new leader.
- Lock table at each replica that is a leader. Contains state for 2-phase locking.
- Transaction manager = used to support distributed transactions
- Participant leader = built on top of transaction manager; used when transactions involve multiple Paxos groups.

As the number of replicas increases, the latency to achieve a quorum becomes less sensitive to slowness at one slave replica.

Shorter lease times would reduce the effect of server deaths on availability, but would require gretaer amounts of lease-renewal network traffic.



Chapter 2

Zookeeper: Wait-free coordination for large scale systems

Coordination examples: Group membership • Leader election • Dynamic Configuration • Status monitoring • Queuing • Critical sections

What is Zookeeper: Highly available, scalable, distributed, configuration, consensus, group membership, leader election, naming, and coordination service.

2.1 Main contributions

- Coordination kernel – Wait-free coordination
- Coordination recipes – Build higher primitives
- Experience with coordination – Some application use ZooKeeper

2.2 Zookeeper properties

- File API without partial reads/writes
- Ordered updates and strong persistence guarantees
- Conditional updates (version)
- Watches for data changes
- Ephemeral nodes
- Generated file names
- No renames

2.3 Zookeeper guarantees

- Linearisable writes – Writes serialisable + respect precedence
- FIFO client order
  - Clients never detect old data
  - Clients get notified of change to watched data within bounded time
  - All requests from client processed in order
  - All results received by client consistent with results received by other clients

2.4 Zookeeper model

- Znode
  - In-memory data node
  - Hierarchical namespace
  - Types: Regular/Ephemeral (can be automatically removed by system)
  - Flags: Sequential
  - Designed to store only meta-data or configuration, not general data storage
  - Can store information such as timestamp version
- Watch mechanism
  - Get notification upon update to data
  - One time triggers
- Client sessions
  - Session = connection to server from client
  - Timeout mechanism

Bibliography