Transactions across Heterogeneous NoSQL Key-Value Data Stores

Akon Dey

The University of Sydney

HPTS – Asilomar, 23 September 2013

Motivation

- NoSQL Data Stores
  - Gaining popularity, widely used, sometimes misunderstood
  - Scalable, Distributed, High-performance, Fault-tolerant
- Limited Support for ACID Transactions
  - Single Item or Entity Group transactions
- This works well for some applications
  - When primary-key access and limited scan is sufficient
- But a large number of applications need transactions
  - Megastore, G-Store, Percolator, Spanner, etc. address this
- ACID transactions across multiple items are desired

Also

- Let us also consider other application data
  - State stores, smart device configuration, others
  - This will become more relevant in the very near future
- Why is this even relevant?
  - IPv6 – many more devices are uniquely network addressable
  - Networks are faster – 4G, Infiniband, fiber to home, . . .
  - Connected devices – TVs, Home appliances, etc.
  - Devices are becoming more compute capable
    - ADSL routers, Network printers, others
    - Bonjour (DNS Service Discovery)
- You want to make consistent updates to multiple entities?

Problems

- Data is in Heterogeneous data stores
  - No apriori knowledge of these
- API is different across stores
  - Functionality may be slightly different
    - If-Modified-Since vs. If-None-Match
- There is no cross data store transaction infrastructure
- But the data store supports …
  - Consistent writes
  - Some form of test-and-set operations
  - Ability to add meta-data to the data record – for state
- Additionally access privileges to the data store may differ

Observation

- Data stores have single item transactions
  - Atomic operations on single items
  - Conditional update
- Enables moving one record from one consistent state to another
  - Update iff the state of the item is what was originally read
- Idea
  - Use these properties to coordinate transaction across multiple items

Approach

- Break the transaction coordination into two parts
  - Move the coordinator to the client (library) – Transaction class
  - Move state information to the data store (meta-data to record)
- Add meta-data to data records
- Global Transaction Status Record (TSR)
  - Globally readable – stored in a HA data store
  - Existence indicates transaction is committed

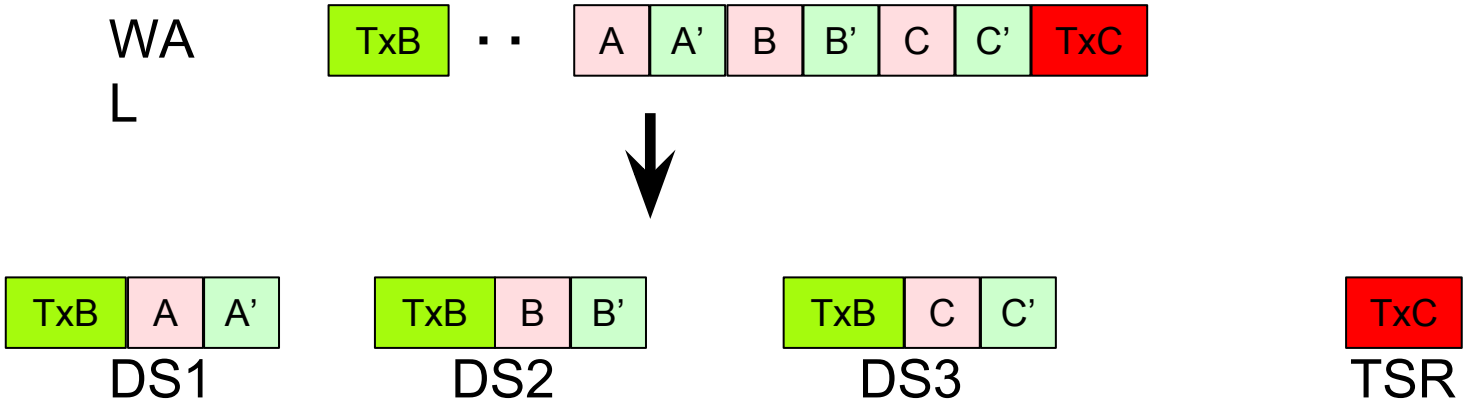
Concurrency Control

- Effectively MVCC
- Similar to Percolator (OSDI'10) … but
  - No central infrastructure
  - Support heterogeneous data stores
- Use the test-and-set features to implement
- Update records in order of hash(primary key)
  - Global order – conflicting transactions has one winner
  - Rest will rollback
- Deadlock avoidance
- Rollback
- Lazy recovery

Logging & Recovery

- Think of it as a deconstructed WAL
  - UNDO + REDO records → Data item
  - Commit record → globally visible Transaction Status Record (TSR)

- Once the TSR exists the transaction is committed
  - If client dies – lazy recovery by another client will recover the transaction



Atomic Commit

- We do it in two phases
- Prepare stage: ORDERED test-and-set (ETag or TS)
  - meta-data + data + previous version of data
  - meta-data = TxID, PREPARE, Commit TS, Lease TS
  - If Lease TS expires during prepare stage: rollback (timeout)
  - Write the TSR as COMMITTED
  - Rollforward after this point
- Commit stage: commit all records in parallel
  - Delete the TSR (asynchronously)

Challenges

- Time
  - Our approach is compatible with TrueTime (time consistency windows)
  - TrueTime depends on NTP with atomic and GPS clock at Google scale in a controlled environment
- Reduce network traffic
  - Both the size of messages and the number of messages
- Evaluation
  - Performance
  - Validity (ACID characteristics)

Summary

- Distributed Transactions over autonomous NoSQL data stores
- Prototype Java library works with
  - Google Cloud Storage
  - Window Azure Storage
- Evaluation
  - YSCB with support for Transactions
- Challenges
  - Time
  - Evaluation
- Question and Suggestions
  - akon.dey@sydney.edu.au