CSE 486/586 Distributed Systems
Google Spanner

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Recap

- Digital certificates
  - Binds a public key to its owner
  - Establishes a chain of trust
- TLS
  - Provides an application-transparent way of secure communication
  - Uses digital certificates to verify the origin identity
- Authentication
  - Needham-Schroeder & Kerberos

Google Spanner

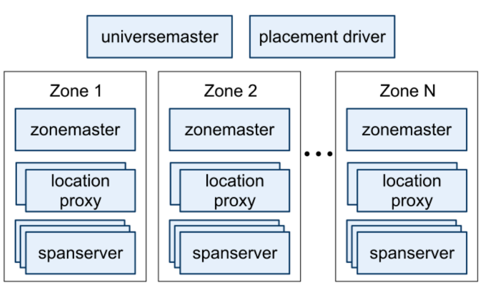
- Geo-distributed database
  - Multiple datacenters, not just a single cluster
- Like Dynamo, it's a combination of traditional techniques with some new twists.
- Traditional concepts used
  - Distributed transactions
  - Paxos
  - Two-phase locking
  - Two-phase commit
  - Linearizability (well, this is more of a property.)
- New twists
  - Relational data model + key-value store
  - *TrueTime* used for synchronization and consistency

Google Spanner

- Overcomes limitations of two other storage systems popular in Google---Bigtable and Megastore.
- Bigtable does not support strong consistency, only eventual.
- Bigtable's data model is not easy to maintain.
- Megastore provides strong consistency and easier-to-maintain data model (more like a relational database), but low performance.
  - Gmail, Picasa, Calendar, Play Store, AppEngine all used Megastore then.
- Transaction support brings lots of benefit.

System Overview

- Universes and Zones



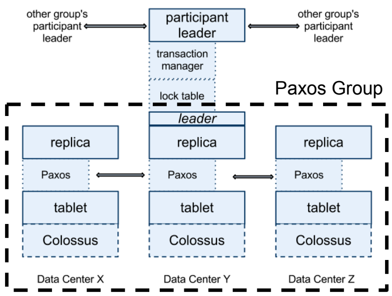
Data Model

- Spanservers manage *tablets* (100~1000).
  - A table contains multiple, mostly contiguous, of: (key, timestamp) -> value
  - This makes it more like a multi-version database than a key-value store.
- Relational data mode & support for SQL-like queries

```
CREATE TABLE Users {
  uid INT64 NOT NULL, email STRING
} PRIMARY KEY (uid), DIRECTORY;
CREATE TABLE Albums {
  uid INT64 NOT NULL, aid INT64 NOT NULL,
  name STRING
} PRIMARY KEY (uid, aid),
  INTERLEAVE IN PARENT Users ON DELETE CASCADE;
```

Spanserver

- Combination of many techniques



CSE 486/586 Administrivia

- PA4 due 5/9
- Final: 5/14, Wednesday, 3:30pm – 6:30pm
  - Norton 112

Paxos Group Leader Election

- The leader election uses *leases*.
- Protocol
  - A potential leader sends requests to others.
  - Others reply back with *lease votes*.
  - If the requester receives a quorum, it becomes the leader.

Transactions

- Read-write transactions
  - Combination of reads and writes
  - Standalone writes
- Read-only transactions
  - Only reads
  - Pre-declared
- Snapshot reads
  - Reads of a past version, not the most up-to-date version
  - A client-specified timestamp, upper-bounded, or Spanner-picked.

Transaction Ordering

- Necessary guarantee for linearizability
  - If T1 finishes before T2, then T2 should see the result of T1.
- Spanner uses physical time to achieve this.
- Each transaction gets a physical (not logical) timestamp.
- Transactions are ordered based on their timestamps.
  - Spanner's Paxos group decides in what order transactions should be committed according to the timestamps.
- Transaction ordering guarantee
  - If T1 commits at *time1* and T2 starts at *time2* where *time1* < *time2*, then T1's timestamp should be less than T2's.
- What is critical in this scenario?
  - Physical time synchronization!

Time Synchronization: TrueTime

- Each data center has
  - GPS and atomic clocks
  - These two provide very fine-grained clock synchronization down to a few milliseconds.
  - Every 30 seconds, there's maximum 7 ms difference.
- Multiple synchronization daemons per data center
  - GPS and atomic clocks can fail in various conditions.
  - Sync daemons talk to each other within a data center as well as across data centers.
- TrueTime API exposes uncertainty.
  - TT.now(): returns an interval [earliest, latest]
  - TT.after(t): true if t has definitely passed
  - TT.before(t): true if t has definitely not arrived

TrueTime for Transaction Ordering

- This is simplified.
- Principle: using TrueTime, always pick a clock value that is not uncertain.
- Commit timestamp is assigned after a commit request is received at the coordinator leader.
  - For transaction T(i), pick S(i) > TT.now().latest: this ensures that actual TT.now() has definitely passed.
- The coordinator leader starts two-phase commit.
  - This takes time and at some point of time all commits will be done.
  - The coordinator leader makes sure that no read can read the outcome of the commit until TT.after(S(i)) is true.
  - This makes sure that the commit time has definitely passed.

Combatting Replica Asynchrony

- Asynchrony still exists in replicas, i.e., different replicas proceed at different speeds.
  - Some replicas can be ready to serve a write, some others might not.
- Each replica maintains *safe time* (t_safe).
  - t_safe means up to what time the replica is up-to-date.
- Replica can serve read requests up to the safe time value.
  - A read request at time t can be served at a replica when t_safe >= t.
- Safe time is basically the timestamp from *the last fully-committed, fully-replicated* transaction (write).

Some Performance Numbers

- Read/write across data centers within 1 ms latency

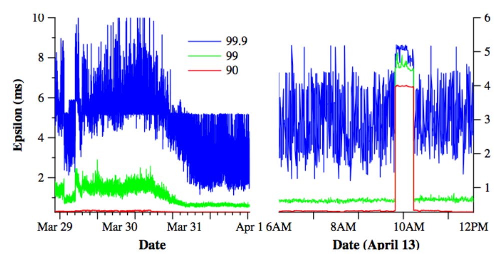
replicas	latency (ms)			throughput (Kops/sec)		
	write	read-only transaction	snapshot read	write	read-only transaction	snapshot read
1D	9.4±.6	—	—	4.0±.3	—	—
1	14.4±1.0	1.4±.1	1.3±.1	4.1±.05	10.9±.4	13.5±.1
3	13.9±.6	1.3±.1	1.2±.1	2.2±.5	13.8±3.2	38.5±.3
5	14.4±.4	1.4±.05	1.3±.04	2.8±.3	25.3±5.2	50.0±1.1

- Two-phase commit (transactions)

participants	latency (ms)	
	mean	99th percentile
1	17.0 ±1.4	75.0 ±34.9
2	24.5 ±2.5	87.6 ±35.9
5	31.5 ±6.2	104.5 ±52.2
10	30.0 ±3.7	95.6 ±25.4
25	35.5 ±5.6	100.4 ±42.7
50	42.7 ±4.1	93.7 ±22.9
100	71.4 ±7.6	131.2 ±17.6
200	150.5 ±11.0	320.3 ±35.1

TrueTime Performance

- Clock difference distribution



F1-Perceived Latency

- F1 is Google's ad backend

operation	latency (ms)		count
	mean	std dev	
all reads	8.7	376.4	21.5B
single-site commit	72.3	112.8	31.2M
multi-site commit	103.0	52.2	32.1M

Summary

- Spanner
  - Geo-distributed database
  - Supports a relational data model with a SQL-like language
  - Supports distributed transactions with linearizability
- Transaction ordering for linearizability
  - TrueTime-based timestamps
  - Principle: using a time value that is certain
- TrueTime
  - TT.now() returns an interval [earliest, latest].
  - TT.after(t) is true if t has definitely passed.
  - TT.before(t) is true if t has definitely not arrived.

Acknowledgements

- These slides contain material developed and copyrighted by Indranil Gupta (UIUC), Jennifer Rexford (Princeton) and Michael Freedman (Princeton).