

Spanner: Google's Globally-Distributed Database

J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, and others, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, p. 8, 2013.



Helpful Definitions

Sharding: is a type of database partitioning that separates very large databases the into smaller, faster, more easily managed parts called data shards. The word shard means a small part of a whole.

Replication: Sharing of information to improve availability, durability

1. [Transactional replication](#). This is the model for replicating transactional data, for example a database or some other form of transactional storage structure with one-copy serializability.
2. [State machine replication](#). This model assumes that replicated process is a deterministic finite automaton and that atomic broadcast of every event is possible, usually implemented by a replicated log consisting of multiple subsequent rounds of the Paxos algorithm.
3. [Virtual synchrony](#). This computational model is used when a group of processes cooperate to replicate in-memory data or to coordinate actions.

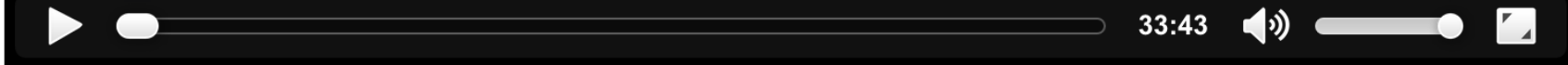
Helpful Definitions

Clock drift: several related phenomena where a clock does not run at the exact right speed compared to another clock. That is, after some time the clock "drifts apart" from the other clock.

- Everyday clocks such as wristwatches have finite precision. Eventually they require correction to remain accurate. The rate of drift depends on the clock's quality, sometimes the stability of the power source, the ambient temperature, and other subtle environmental variables. Thus the same clock can have different drift rates at different occasions.
- [Atomic clocks](#) are very precise and have nearly no clock drift. Even the [Earth's rotation rate](#) has more drift and variation in drift than an atomic clock.
- As Einstein predicted, relativistic effects can also cause clock drift due to time dilation or gravitational distortions.

Spanner: Google's Globally-Distributed Database

Wilson Hsieh
Google, Inc.

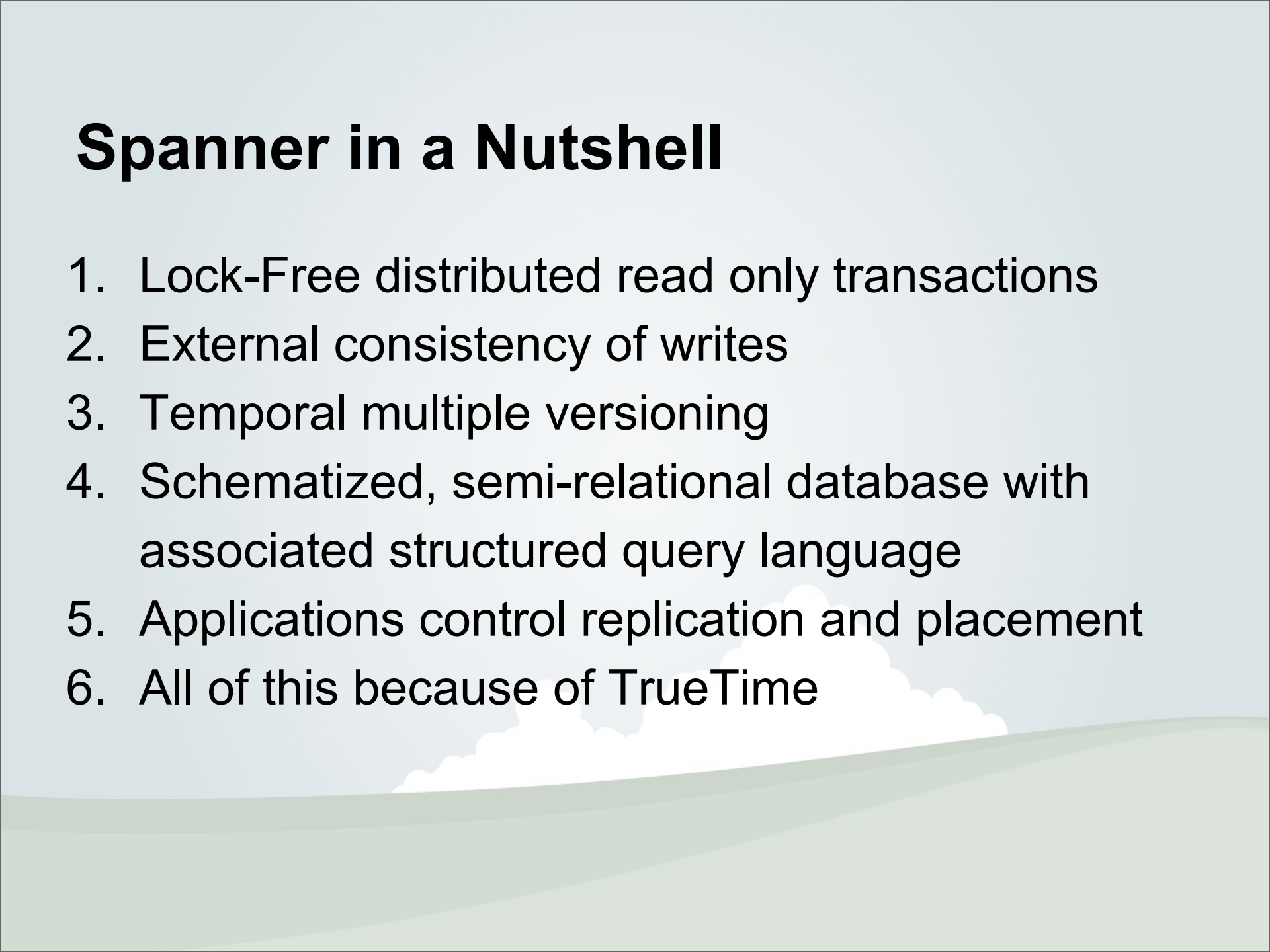


[Download Video](#)

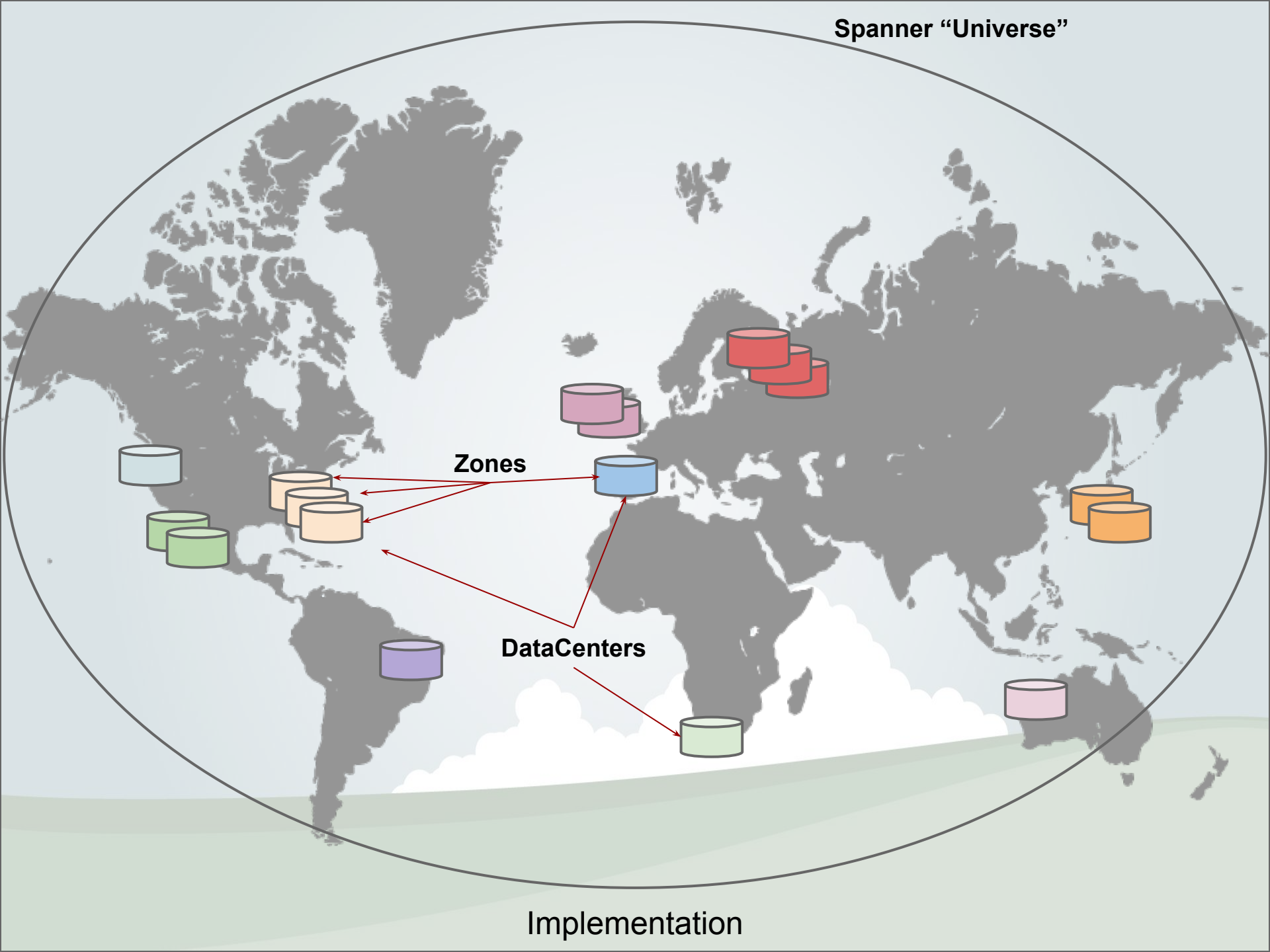
Wilson Hsieh, Google Inc presents Spanner at OSDI 2012

<https://www.usenix.org/conference/osdi12/technical-sessions/presentation/corbett>

Spanner in a Nutshell

1. Lock-Free distributed read only transactions
 2. External consistency of writes
 3. Temporal multiple versioning
 4. Schematized, semi-relational database with associated structured query language
 5. Applications control replication and placement
 6. All of this because of TrueTime
- 

Spanner “Universe”



Zones

DataCenters

Implementation

Zone Server Organization

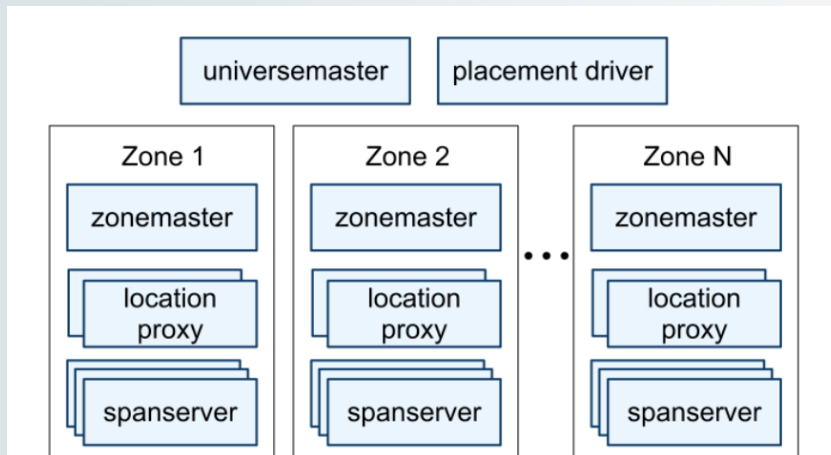


Figure 1: Spanner server organization.

- zonemaster (1 per zone) assigns data to the span servers (similar to NameNode in GFS)
- spanserver (100 - 3000 per zone) serves data to the clients
- Proxies are used by clients to locate span servers with the correct data (shard lookup) these are used to prevent bottleneck at zonemaster

1 per universe:

- universemaster has a console with status information
- placement driver handles data movement for replication or load balancing purposes

Storage Data Model

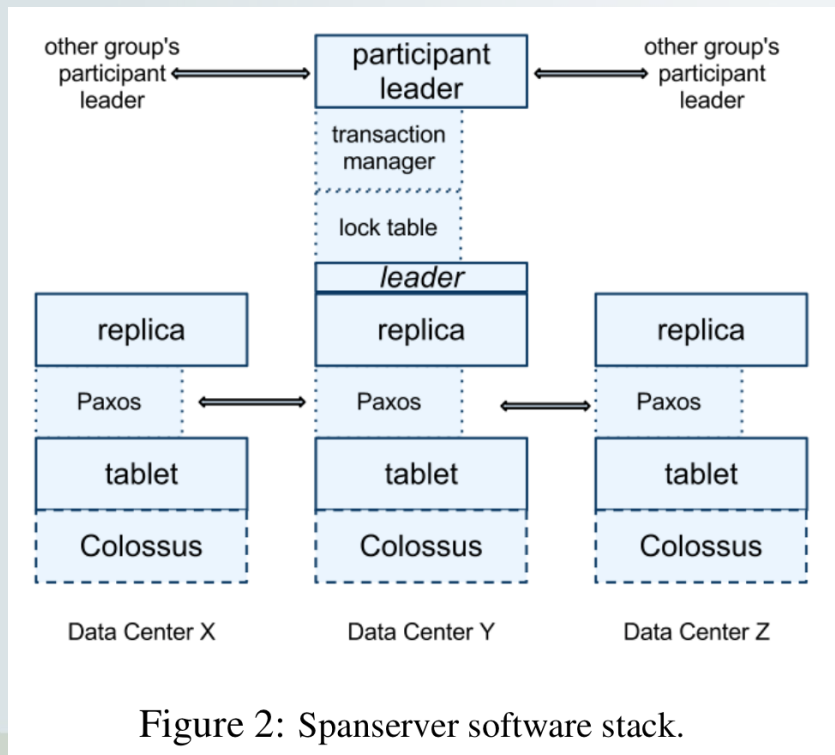
`(key:string, timestamp:int64) →
string`

- **Tablet:** a bag of timestamped key,val pairs as above
- Tablet's state is tracked and stored with B-tree-like files and a write-ahead log. Data storage is on a distributed filesystem (Colossus, the successor to GFS at Google).

Tablets are optimized data structures first implemented by BigTable to track data by row/column position (the keys) and their locations on GFS. In BigTable (and presumably Spanner as well) Tablets are compressed using Snappy to keep them approximately 200 MB in size.

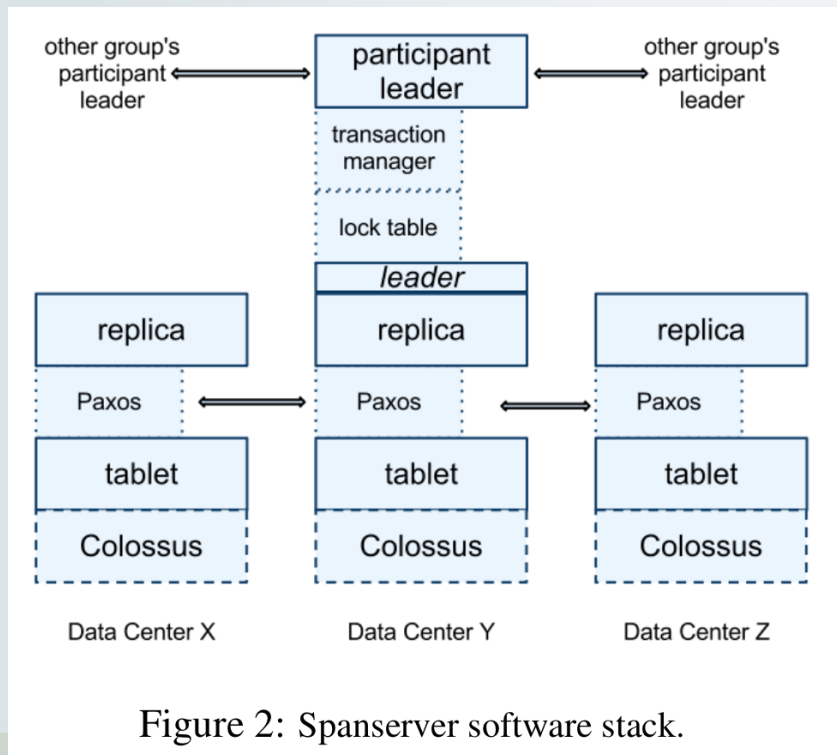
Tablets (for now) can be seen as the basic unit of replication/sharding.

Zone Software Stack



- each tablet gets a Paxos state machine that maintains its log and state in the tablet (Paxos writes twice)
- Paxos has long lived leaders (10 seconds) and is pipelined to improve for WAN latencies
- Paxos is used to maintain consistency among replicas; every write must initiate at the Paxos leader.
- The set of replicas is collectively a Paxos group.

Zone Software Stack Continued



- Each leader maintains a lock table with state for two phase locking, mapping range of keys to lock states.
- Long lived leader and is critical, design for long-lived transactions (reports).
- Transaction manager is used to implement distributed transactions and to select a participant leader, which coordinates Paxos between participant groups.

Buckets* and Data Locality

- Directories: collection of contiguous keys that share a prefix.
- Prefixes allow applications to control the locality of their data.

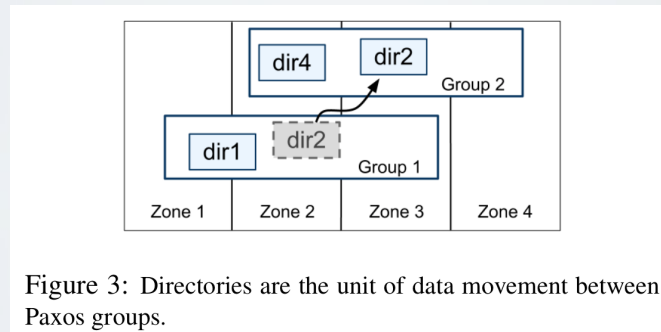


Figure 3: Directories are the unit of data movement between Paxos groups.

- All data in a directory has same replica configuration; movement of data between Paxos groups is via directory, in order to store data in a group closer to its accessors.
- Spanner Tablet != BigTable Tablet, Spanner containers encapsulate multiple partitions of row space.
- Smallest unit of geographic replication properties (placement)

Application Data Model

- schematized, semi-relational tables and a structured query language + general purpose transactions.
- Is 2PC too expensive? Leave it to the application to deal with performance.
- On top of Buckets an application creates a *database* in the universe, composed of *tables*.
- But not actually relational; all rows *must* have primary keys.

```
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;
```

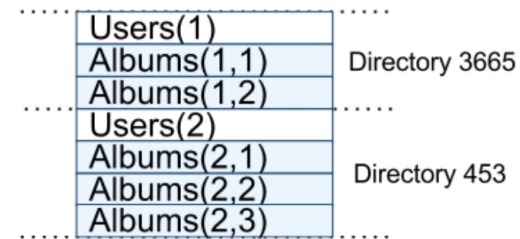


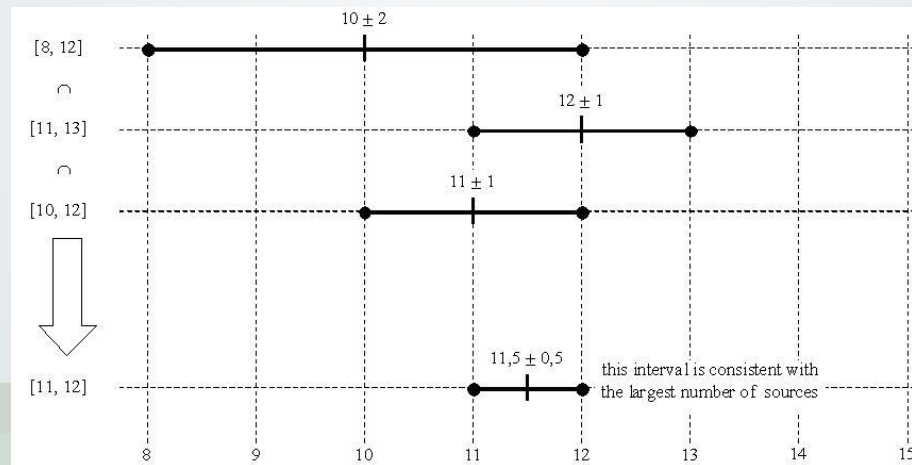
Figure 4: Example Spanner schema for photo metadata, and the interleaving implied by `INTERLEAVE IN`.

TrueTime

- Time is an interval with timestamp endpoints
 - `TT.now()` → returns a `TTinterval`: [earliest, latest]
 - `TT.after(t)` → returns `True` iff $t > \text{latest}$
 - `TT.before(t)` → returns `True` iff $t < \text{earliest}$
- \square is the instantaneous error bound (half the intervals width)
- Clients poll TimeMasters (GPS and Atomic Clocks) and use Marzullo's algorithm to detect and reject liars. Also detect bad local times or clocks.
- Between polls, \square gets increasingly larger until the next poll and this increase is derived from worst-case local clock drift (significantly larger than expected).

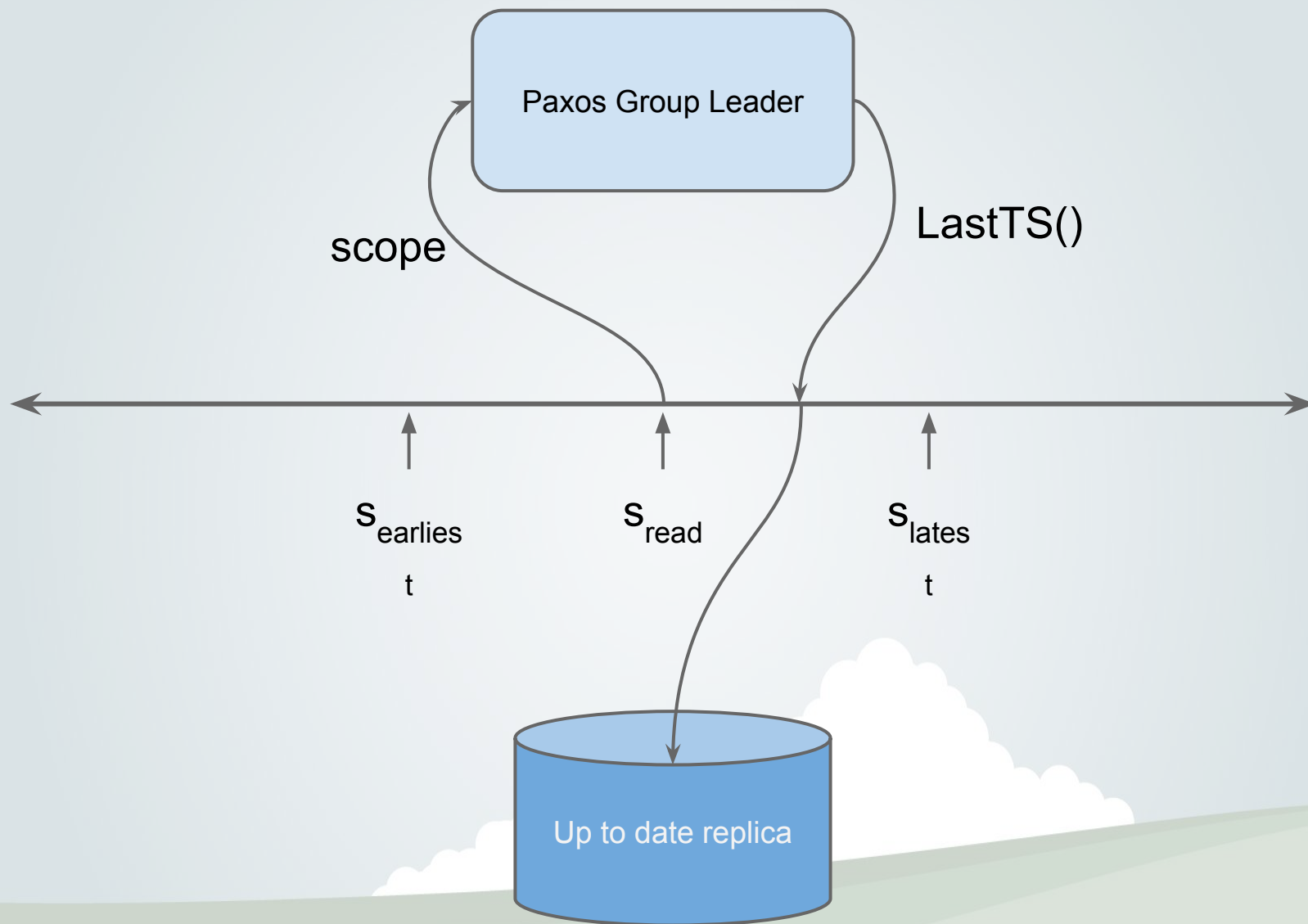
Marzullo's Algorithm

- An agreement algorithm used to select sources for estimating accurate time from a number of noisy time sources.
- A refined version of it, renamed the "intersection algorithm", forms part of the modern NTP.
- Usually the best estimate is taken to be the smallest interval consistent with the largest number of sources.



Concurrency Control

- read-only transactions
 - Not simply read-write transactions w/o writes
 - execute at a system chosen timestamp
 - non-blocking (incoming writes aren't locked)
 - proceeds on any replica sufficiently up to date
- snapshot-reads
 - client specified timestamp
 - proceeds on any replica sufficiently up to date
- commit is inevitable once timestamp is chosen so clients can avoid buffering.



Read Only Transactions

2PC and RW Transactions

- timestamps assigned anytime when locks have been acquired but before lock is released.
- within each Paxos group, Spanner assigns timestamps in monotonically increasing order, even across leaders.
- A leader must only assign timestamps within its leader lease

External consistency for T_1 and T_2 : $t_{\text{abs}}(e_1^{\text{commit}}) < t_{\text{abs}}(e_2^{\text{start}})$

- The coordinator leader for a write T_i assigns a commit timestamp s_i no less than the value of TT.now().latest
- Coordinator ensures clients see no data committed by T_i until $\text{TT.after}(s_i)$ is true using a commit wait.

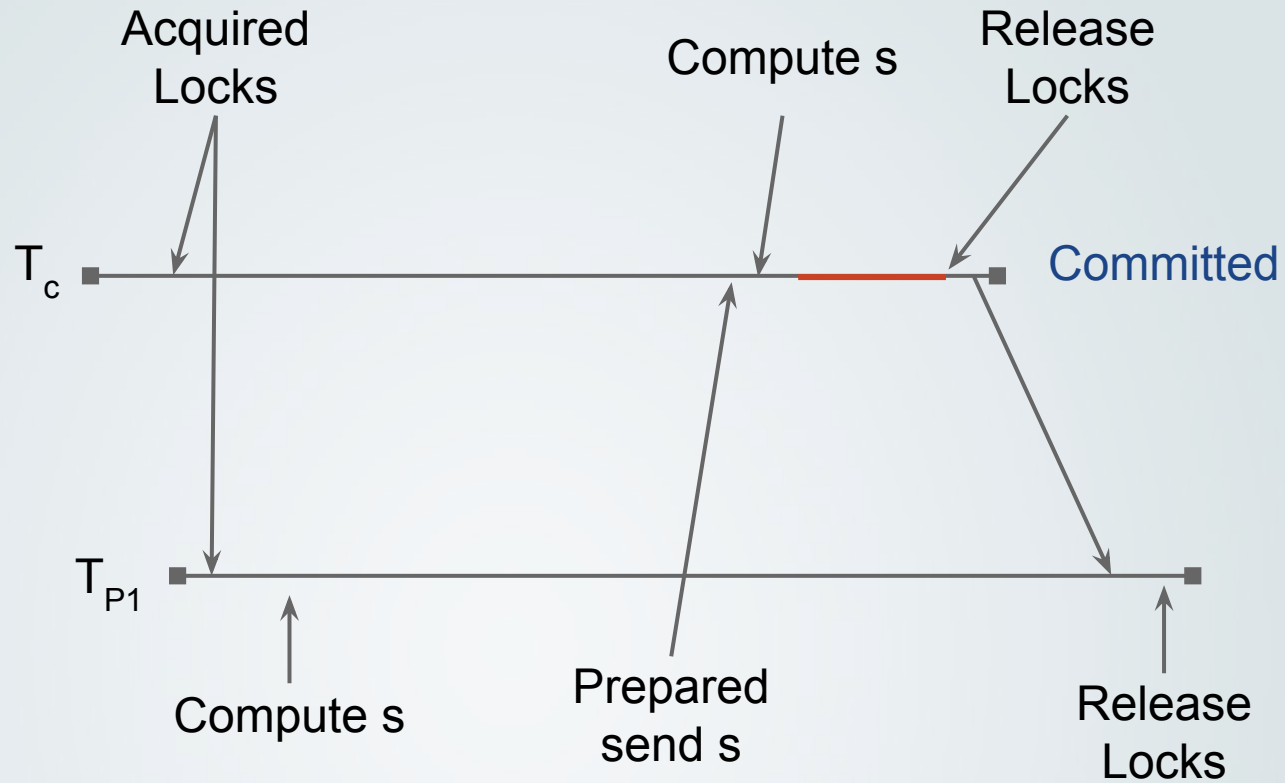
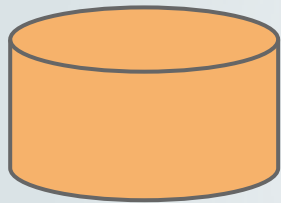
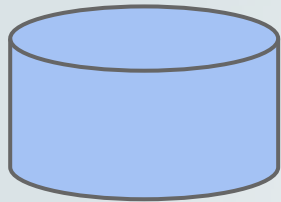
RW Transactions - Other Details

- Writes are buffered until a commit; reads in a transaction do not see the effects of the transaction's writes.
- Uncommitted data is not assigned a timestamp
- Reads in RWX use wound-wait to avoid deadlocks (preference given to older transaction)

Client:

1. Acquire locks, read most recent data
2. Send keep alive messages to avoid timeouts
3. Begin two phase commit
4. Choose coordinator group, send commit to each leader

Having client drive 2PC avoids sending data twice



Read-Write Transactions

Schema Change Transactions

- The number of groups in the database could be in the millions, how do you make an atomic schema change? (BigTable only supports changes in one data center, but locks everything)
- Spanner has a non-blocking variant of standard transactions for schema changes:
 1. Assign a timestamp in the future (registered during prepare)
 2. Reads and Writes synchronize with any registered schema at the future timestamp, but not before
 3. If they have reached the timestamp, they must block if their schemas are after timestamp t

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

Table 3: Operation microbenchmarks. Mean and standard deviation over 10 runs. 1D means one replica with commit wait disabled.

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

Table 4: Two-phase commit scalability. Mean and standard deviations over 10 runs.

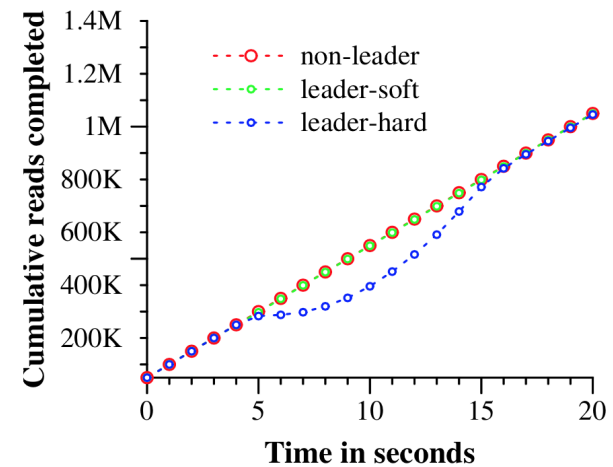


Figure 5: Effect of killing servers on throughput.

F1 Case Study

- Spanner started being experimentally evaluated under production workloads in early 2011, as part of a rewrite of Google's advertising backend called F1.
- Replaced a MySQL database that was manually sharded.
- The uncompressed dataset is tens of terabytes.
- Spanner removes need to manually reshard
- Spanner provides synchronous replication and auto failover
- F1 requires strong transactional semantics
- Fairly invisible transfer from MySQL to Spanner; data center loss and swapping required only an update to schema to move leaders to closer front ends.

