Spanner: Google's Globally-Distributed Database

Phil Gibbons

15-712 F15

Lecture 14

Today's Reminders

- **Discuss Project Ideas with Phil & Kevin**
  - Phil's Office Hours: After class today
  - Sign up for a slot: 11-12:30 or 3-4:20 this Friday

Spanner: Google's Globally-Distributed Database

[OSDI'12 best paper]

James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, JJ Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, Dale Woodford
(Google x 26)

Database vs. Key-value Store

"We provide a database instead of a key-value store to make it easier for programmers to write their applications."

"We consistently received complaints from users that Bigtable can be difficult to use for some kinds of applications."

Spanner

- **Worked on Spanner for 4½ years at time of OSDI'12**
- **Scalable, multi-version, globally-distributed, synchronously-replicated database**
  - Hundreds of datacenters, millions of machines, trillions of rows
- **Transaction Properties**
  - Transactions are externally-consistent (a.k.a. Linearizable)
  - Read-only transactions are lock-free
  - (Read-write transactions use 2-phase-locking)
- **Flexible replication configuration**

[Slides from OSDI'12 talk]

Spanner: Google's Globally-Distributed Database

Wilson Hsieh
representing a host of authors

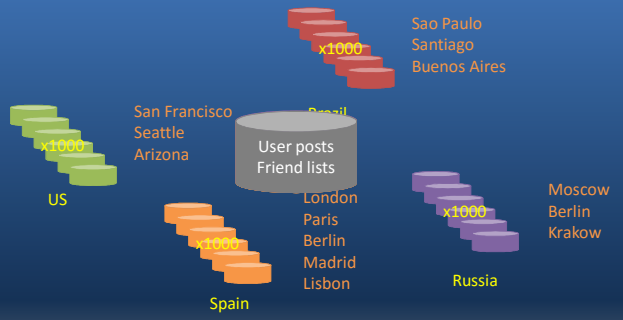
OSDI 2012

Google

What is Spanner?

- Distributed multiversion database
  - General-purpose transactions (ACID)
  - SQL query language
  - Schematized tables
  - Semi-relational data model
- Running in production
  - Storage for Google's ad data
  - Replaced a sharded MySQL database

Example: Social Network



Overview

- Feature: Lock-free distributed read transactions
- Property: External consistency of distributed transactions
  - First system at global scale
- Implementation: Integration of concurrency control, replication, and 2PC
  - Correctness and performance
- Enabling technology: TrueTime
  - Interval-based global time

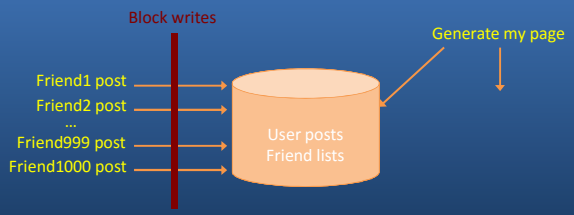
Read Transactions

- Generate a page of friends' recent posts
  - Consistent view of friend list and their posts

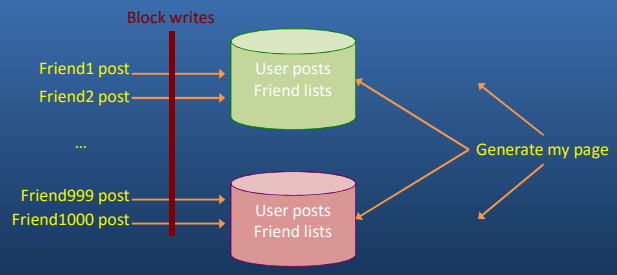
Why consistency matters

1. Remove untrustworthy person X as friend
2. Post P: "My government is repressive…"

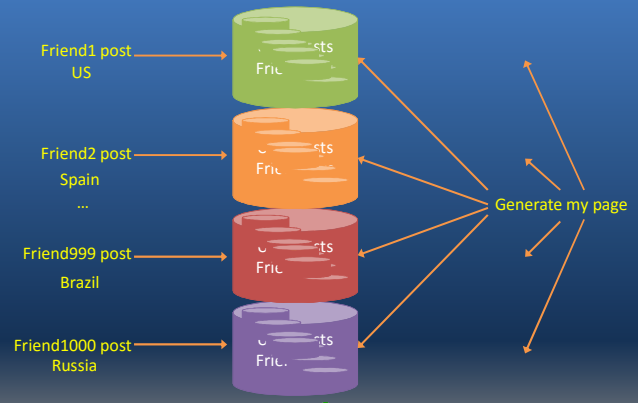
Single Machine



Multiple Machines



Multiple Datacenters



Version Management

- Transactions that write use strict 2PL
  - Each transaction *T* is assigned a timestamp *s*
  - Data written by *T* is timestamped with *s*

Time	<8	8	15
My friends	[X]	[]	
My posts			[P]
X's friends	[me]	[]	

Synchronizing Snapshots

Global wall-clock time

==

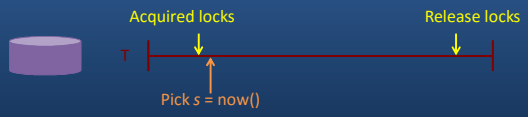
External Consistency:
Commit order respects global wall-time order

==

Timestamp order respects global wall-time order given
timestamp order == commit order

Timestamps, Global Clock

- Strict two-phase locking for write transactions
- Assign timestamp while locks are held



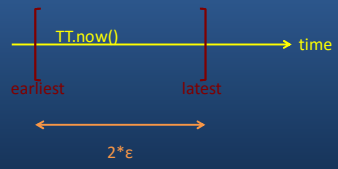
Timestamp Invariants

- Timestamp order == commit order

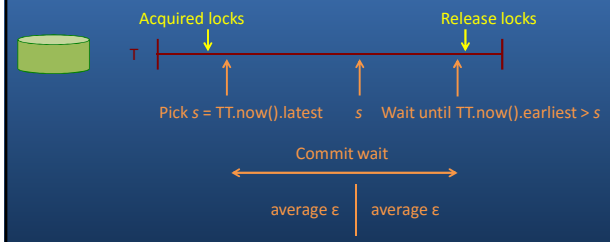
- Timestamp order respects global wall-time order

TrueTime

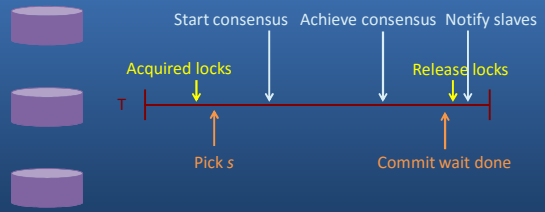
- "Global wall-clock time" with bounded uncertainty



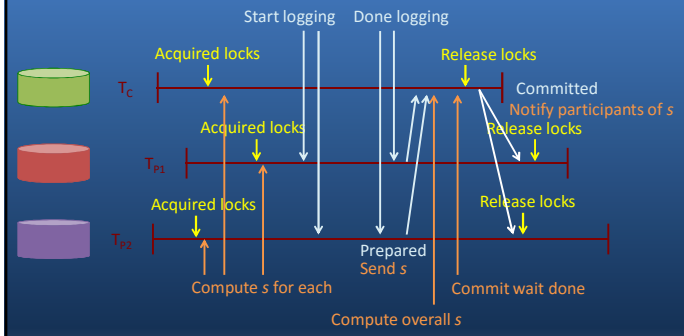
Timestamps and TrueTime



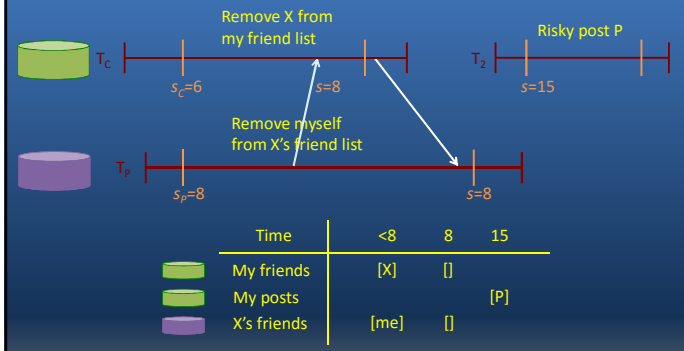
Commit Wait and Replication



Commit Wait and 2-Phase Commit



Example



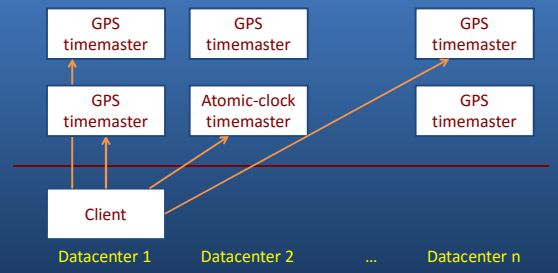
What Have We Covered?

- Lock-free read transactions across datacenters
- External consistency
- Timestamp assignment
- TrueTime
  - Uncertainty in time can be waited out

What Haven't We Covered?

- How to read at the present time
- Atomic schema changes
  - Mostly non-blocking
  - Commit in the future
- Non-blocking reads in the past
  - At any sufficiently up-to-date replica

TrueTime Architecture

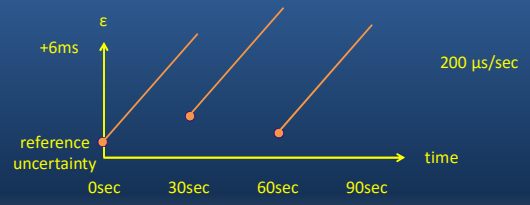


Compute reference [earliest, latest] = now ± ε

TrueTime implementation

now = reference now + local-clock offset

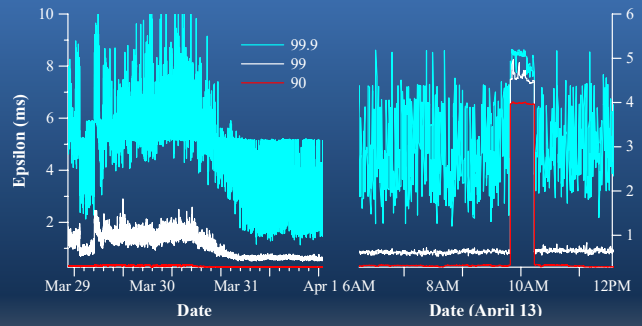
ε = reference ε + worst-case local-clock drift



What If a Clock Goes Rogue?

- Timestamp assignment would violate external consistency
- Empirically unlikely based on 1 year of data
  - Bad CPUs 6 times more likely than bad clocks

Network-Induced Uncertainty



What's in the Literature

- External consistency/linearizability
- Distributed databases
- Concurrency control
- Replication
- Time (NTP, Marzullo)

Future Work

- Improving TrueTime
  - Lower ε < 1 ms
- Building out database features
  - Finish implementing basic features
  - Efficiently support rich query patterns

Conclusions

- Reify clock uncertainty in time APIs
  - Known unknowns are better than unknown unknowns
  - Rethink algorithms to make use of uncertainty
- Stronger semantics are achievable
  - Greater scale != weaker semantics

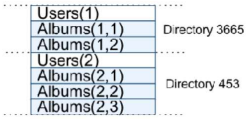
[End of slides from OSDI'12 talk]

Semi-relational Data Model

- **Rows must have names**
  - Every table must have one or more primary-key columns
- **Applications control data locality thru their choice of 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;
```



Read-Only Transactions Constraints

- **Must declare "read-only" upfront**
- **Must have "scope" expression**
  - Summarize keys that will be read by the entire transaction

Availability

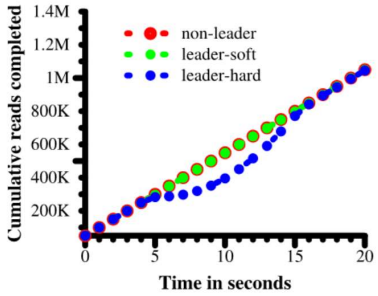


Figure 5: Effect of killing servers on throughput.

F1 Advertising 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

Table 6: F1-perceived operation latencies measured over the course of 24 hours.

Lock Conflicts
Only one DC had SSDs

Friday

Discuss Projects with Phil & Kevin

1) the problem you want to solve

2) the approach you are going to take

3) how it pertains to the course