Spanner: Google’s Globally-Distributed Database

Phil Gibbons
15-712 F15
Lecture 14

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 Quinnan, 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.”

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

- 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

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

[Slides from OSDI’12 talk]
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...”
Version Management

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

<table>
<thead>
<tr>
<th>Time</th>
<th>My friends</th>
<th>My posts</th>
<th>X's friends</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>[X]</td>
<td>[P]</td>
<td>[me]</td>
</tr>
</tbody>
</table>

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**

- Pick $s = \text{TT.now().latest}$

- Wait until $\text{TT.now().earliest} > s$

**Commit Wait and Replication**

- Start consensus

- Achieve consensus

- Notify slaves
Commit Wait and 2-Phase Commit

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

Datacenter 1 ➔ Datacenter 2 ➔ – ➔ Datacenter n

- GPS timemaster
- GPS timemaster
- Atomic-clock timemaster
- GPS timemaster
- Client

Compute reference [earliest, latest] = now ± ε

TrueTime implementation

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

+6ms

ε

0sec 30sec 60sec 90sec

200 μs/sec

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

Network-induced uncertainty

Mar 29 Mar 30 Mar 31 Apr 1

28

99.9

99

90

Mar 29 Mar 30 Mar 31 Apr 16AM 8AM 10AM 12PM

99.9

99

90

Date

Date (April 13)
What’s in the Literature

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

Future Work

• Improving TrueTime
  – Lower $\varepsilon < 1$ 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 $\neq$ weaker semantics

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

[End of slides from OSDI’12 talk]
Read-Only Transactions Constraints

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

F1 Advertising Backend

<table>
<thead>
<tr>
<th>operation</th>
<th>latency (ms)</th>
<th>mean</th>
<th>std dev</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>all reads</td>
<td></td>
<td>8.7</td>
<td>376.4</td>
<td>21.5B</td>
</tr>
<tr>
<td>single-site commit</td>
<td></td>
<td>72.3</td>
<td>112.8</td>
<td>31.2M</td>
</tr>
<tr>
<td>multi-site commit</td>
<td></td>
<td>103.0</td>
<td>52.2</td>
<td>32.1M</td>
</tr>
</tbody>
</table>

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

Lock Conflicts
Only one DC had SSDs

Availability

Figure 5: Effect of killing servers on throughput.

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