Spanner: Google’s Globally-Distributed Database


Google, Inc.
Abstract

Spanner is Google’s scalable, multi-version, globally-distributed, and synchronously-replicated database. It is the first system to distribute data at global scale and support externally-consistent distributed transactions. This paper describes how Spanner is structured, its feature set, the rationale underlying various design decisions, and a novel time API that exposes clock uncertainty. This API and its implementation are critical to supporting external consistency and a variety of powerful features: non-blocking reads in the past, lock-free read-only transactions, and atomic schema changes, across all of Spanner.
Context for Spanner

- Distributed database that supports transactions

<table>
<thead>
<tr>
<th>Employee ID</th>
<th>Date of Joining</th>
<th>Age</th>
<th>Title</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- BigTable: Database spread across servers within a datacenter
- Spanner: Database spread across data centers
Support for Transactions

- **Serializability**
  - Externally visible effects equivalent to *some* serial order of execution

- **How is this different from linearizability?**
  - Serializability does not guarantee real time order

- **Spanner offers strict serializability**
  - Serializability + Linearizability
  - Previously considered impractical at global scale
Two Phase Locking

Lock

Commit
Serializability in Spanner

- Database partitioned into shards
- Each shard has replica in 3 datacenters
  - Form a Paxos group
- Two phase commit to execute any transaction
  - First phase: Write lock acquisition to Paxos log
  - Second phase: Write updates + lock release to log
Executing Transactions

Round 1: **Lock**
Round 2: **Commit**

Transaction latency = 4 rounds of wide-area communication
2 rounds for each phase of 2PL
Executing Transactions

Round 1: Lock

Round 2: Commit

Leader-based Paxos → each phase of 2PL in 1 round-trip

Still too slow for read-only transactions
Read-Only Txns with 2PL

TC

P1

P2

P3

Lock

Commit
Lock-free Read-Only Txns

How to ensure T1’s read from P1 returns data from before T2?
Snapshot Reads

- Every transaction has a commit timestamp
- When a row is modified by transaction, create new version
  - Transaction’s timestamp is version number
- Read-only transaction reads latest version of data committed before its timestamp
- What property must be true for linearizability?
  - Commit timestamps should monotonically increase
Synchronizing Clocks

- Accurate sources of time:
  - GPS
  - Atomic clocks

- Why not put these into every server?

- Spanner deploys a few of these in every DC
- All servers periodically sync with timemasters
TrueTime Architecture

Current time = now ± ε
TrueTime Implementation

- Between syncs, assume worst case clock drift

Diagram:
- Reference uncertainty
- Time (0 sec, 30 sec, 60 sec, 90 sec)
- ε shift
- +6ms
- 200 μs/sec
TrueTime

- **TT.now() returns a range:** \([\text{earliest}, \text{latest}]\)
- **Current time is ...**
  - ... not less than earliest
  - ... not greater than latest

- **High-level principle:**
  - Known unknowns better than unknown unknowns
Timestamps and TrueTime

- How to use TrueTime to ensure linearizability?
  - How to pick commit timestamp?
  - Until when to wait before releasing locks?
- \( TT.\text{now}() \) returns \([\text{earliest}, \text{latest}]\)
Announcements

- Thursday:
  - Guest lecture from Google Ads back-end tech lead
  - Lecture won’t be recorded
  - Student meeting from 4:30 to 5:30 in 4941 BBB

- No lecture next Tuesday

- Project 4:
  - Due two weeks from Thursday
Leader Leases

- Must ensure at most one leader in any replica group at any point in time
- In Spanner, leader gets a lease on how long it gets to be the leader

- How can group members come to consensus on leader leases to ensure leader disjointness?
- How to leverage TrueTime?
Paxos-based Leader Leases

- When a replica responds to a proposer, it grants lease for interval $[t, t + 10s]$
- What should be value of $t$?
  - $t = \text{TT.now().earliest()}$

- Proposer assumes leader lease until minimum of replica leases
  - Lease valid until $\text{TT.now().latest} > \text{min lease}$
Schema-Change Transactions

- Not scalable to acquire locks on all shards
- How can we leverage TrueTime?
- Schedule transaction for time $t$ in future
  - All txns with timestamp $< t$, see old schema
  - All txns with timestamp $> t$, see new schema
Spanner’s Availability

- Spanner availability similar to Chubby
  - Uptime > 99.999%
- Spanner unavailability << Client unavailability
  - Don’t need app logic to deal with Spanner outage

- Spanner communicates over Google’s private network
  - Well engineered over last decade
  - Partitions are very uncommon
Causes of Spanner Outages

Week after Thanksgiving ...

- Memcached at Facebook
- Bitcoin
- Research at Michigan