Spanner: Google’s Globally-Distributed Database

Presented by: Liangyu Zhao, Xiangfeng Zhu
What is Spanner?

- Google’s scalable, multi-version, globally-distributed, and synchronously-replicated database
  - SQL Query Language
  - General-purpose transactions (ACID)
  - Schematized tables
  - Semi-relational data model
Why Spanner?

● Bigtable (OSDI 2006)
  ○ Difficult to use for complex, evolving schemas and by applications that requires strong consistency guarantees for geo-replicated sites.

● Megastore (CIDR 2011)
  ○ Layered on top of Bigtable
  ○ Provides semi-relational data model and similar schema language to Spanner’s
  ○ Bad write throughput (e.g., does not support long-lived leaders)
Linearizability and Serializability

● **Linearizability**: Guarantee for a single operation on a single object
  ○ Writes should be appear instantaneously; All later reads as defined by wall-clock time (i.e., real-time) reflect the written value or some latter written value
  ○ “C” in CAP theorem

● **Serializability**: Guarantee for transactions, or one or more operations on one or more objects
  ○ A set of transactions over some objects should execute as though each transaction ran in some serial order (doesn’t specify which order)
  ○ “Isolation” in ACID properties

● **Linearizability + Serializability = Strict Serializability**
  ○ Transactions have some serial behavior which corresponds to wall-clock time
Consistency Properties

- **Serializable** transaction isolation
- **Linearizable** reads and writes
- Atomic commit of transactions across shards
  - Each shard holds a subset of the data

- **Spanner’s techniques**
  - State machine replication (Paxos) within a shard
  - Two-phase locking (2PL) for serializability
  - Two-phase commit (2PC) for cross-shard atomicity
Lock-free read-only transactions

● Spanner guarantees a read-only transactions observes a consistent snapshot:
  ○ If $T_1 \rightarrow T_2$, snapshot reflects writes by $T_2$ also reflects writes by $T_1$ and snapshot that does not reflect writes by $T_1$ does not reflect writes by $T_2$
  ○ Snapshot is **consistent with causality**

● Spanner’s approach:
  ○ Each read-write transaction $T_w$ is assigned a commit timestamp $t_w$
  ○ Every value is tagged with the timestamp of corresponding transaction
  ○ Read-only transaction $T_r$ is assigned a timestamp $t_r$
  ○ $T_r$ contains values with $t_w \leq t_r$ but ignores values with $t_w > t_r$
Timestamps

- Can we use physical clocks?
  - No, inconsistent with causality
- Can we use Lamport clocks?
  - No, linearizability depends on real-time order, but Lamport clocks may not reflect this
TrueTime API & Implementation

- TrueTime is a global synchronized clock with bounded non-zero error.

<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>$TT.now()$</td>
<td>$TTinterval$: $[earliest, latest]$</td>
</tr>
<tr>
<td>$TT.after(t)$</td>
<td>true if $t$ has definitely passed</td>
</tr>
<tr>
<td>$TT.before(t)$</td>
<td>true if $t$ has definitely not arrived</td>
</tr>
</tbody>
</table>

- TrueTime uses GPS and atomic clocks for time references.
  - Each datacenter has a set of **time master** machines:
    - GPS **time master**: physically separated GPS receivers with dedicated antennas.
    - Armageddon **time master**: equipped with atomic clocks.
  - Every machine has a **timeslave daemon**, who polls a variety of time masters to reduce vulnerability from any one master.
TrueTime API & Implementation

- Daemon advertises a slowly increasing time uncertainty $\epsilon$.
  - Reset to 0ms at every poll. Poll happens every 30s.
  - Conservatively applied worst case clock drift (200μs/s).
    - Bad CPUs are 6 times more likely than bad clocks (exceeding 200μs/s).
  - $\epsilon$ also depends on time-master uncertainty and communication delay to time masters.
How does Spanner use TrueTime?

- **Paxos Leader Lease Disjointness Invariant**
  - For each Paxos group, each Paxos leader’s lease interval is disjoint from every other leader’s.

- **External Consistency Invariant**
  - If the start of a transaction $T2$ occurs after the commit of a transaction $T1$, then the commit timestamp of $T2$ must be greater than the commit timestamp of $T1$. 
Paxos Leader Leases

Candidate

Replica 1

Replica 2

Replica 3
Paxos Leader Leases

\[ v^{\text{leader}} = \text{TT.now().earliest} \]

\[ v^{\text{leader} + 10} \]
Paxos Leader Leases

Candidate

$e_{\text{send}}$

Replica 1

Replica 2

Replica 3

$\nu_{\text{leader}}$

$e_{\text{send}}$

$\nu_{\text{leader}} + 10$
Paxos Leader Leases
Paxos Leader Leases

Candidate

Replica 1

Replica 2

Replica 3

\( v_{\text{leader}} \)

\( e_{\text{send}} \)

\( e_{\text{receive}} \)

\( v_{\text{leader}} +10 \)

\( \text{TT.now().latest} \)

\( t_{\text{end}} = \text{TT.now().latest} +10 \)
Paxos Leader Leases

Candidate

Replica 1

Replica 2

Replica 3

$v_{\text{leader}}$ $e_{\text{send}}$ $e_{\text{receive}}$ $v_{\text{leader}} + 10$

$t_{\text{end}}$

Single-Vote Rule
Paxos Leader Leases

Candidate

Replica 1

Replica 2

Replica 3

\( v_{\text{leader}} \)
\( e_{\text{send}} \)
\( e_{\text{receive}} \)
\( e_{\text{grant}} \)
\( v_{\text{leader}} + 10 \)

Single-Vote Rule

\( t_{\text{end}} \)
Paxos Leader Leases

Candidate

Replica 1

Replica 2

Replica 3

$e_{\text{quorum}}^m$

Single-Vote Rule
Paxos Leader Leases

Candidate

Replica 1

Replica 2

Replica 3

Lease Interval

$[\text{TT.now().latest, v}^{\text{leader}}+10]$
Paxos Leader Leases

- The lease interval is contained in the single-vote rule interval.
- If another participant wants to be the leader, in order to hit the quorum, it needs the vote of at least one replica who voted for the previous candidate.
- The lease interval of another leader must be disjoint from the single-vote rule interval.
- Lease intervals of different leaders must be disjoint.

Lease Interval

\[ [\text{TT.now().latest}, v_{\text{leader}} + 10] \]
External Consistency Invariant
External Consistency Invariant

e_1^{server}

T1

T2
External Consistency Invariant

\[ e_{1}^{server} \quad s_{1} \quad = TT.now().latest \]
External Consistency Invariant
External Consistency Invariant

e_1^{server} \rightarrow s_1 \rightarrow \text{Commit Ready} \rightarrow e_1^{commit} \rightarrow \text{TT.after}(s_1)

T1

T2
External Consistency Invariant
External Consistency Invariant

\[ e_1^{\text{server}} \quad s_1 \quad \text{Commit Ready} \quad e_1^{\text{commit}} \quad \text{TT.after}(s_1) \]

\[ s_1 < e_1^{\text{commit}} \]
External Consistency Invariant

\[ s_1 < e_1^{\text{commit}} < e_2^{\text{start}} \]
External Consistency Invariant

$$s_1 < e_1^{\text{commit}} < e_2^{\text{start}} < e_2^{\text{server}}$$
External Consistency Invariant

\[ s_1 < e_1^{\text{commit}} < e_2^{\text{start}} < e_2^{\text{server}} < s_2 \]
Discussion

● Why is Paxos leader lease disjointness invariant necessary?
  ○ Invariant: for each Paxos group, each Paxos leader’s lease interval is disjoint from every other leader’s.

● How does external consistency invariant guarantee external consistency?
  ○ Invariant: if the start of a transaction $T_2$ occurs after the commit of a transaction $T_1$, then the commit timestamp of $T_2$ must be greater than the commit timestamp of $T_1$.

● What are the benefits and drawbacks of TrueTime?
Spanner and CAP Theorem

● CAP Theorem
  ○ Consistency, 100% availability, partition tolerance

● Is Spanner a CP system or a AP system?
  ○ Answer: CP - during partitions, Spanner chooses C and forfeits A.

● What design choices make Spanner a CP system?
  ○ Use of Paxos group to achieve consensus on update - if the leader cannot maintain a quorum due to a partition, updates are stalled and the system is not available
  ○ Use of two-phase commit for cross-group transactions means partition can prevent commits
Availability of Spanner

- Spanner provides 99.999% availability (five “9”s)
  - Chubby paper reports 9 outages of 30 seconds in 700 days¹
- Spanner runs on Google’s own private global network
  - Every Spanner packet flows only over Google-controlled routers and links
  - Each data center has at least three independent fibers
  - Redundancy of equipment and paths within a datacenter

¹The Chubby lock service for loosely-coupled distributed systems - Burrows, OSDI ’06
Dynamo: Amazon’s Highly Available Key-value Store

- Query Model: simple read and write operations to a data item (uniquely identified by a key)
Dynamo: Amazon’s Highly Available Key-value Store

- Query Model: simple read and write operations to a data item (uniquely identified by a key)

- ACID Properties:
  - Provide weaker consistency guarantees - Eventual Consistency
  - No Isolation guarantees and permits only single key updates

- Choose “A” over “C”
## Dynamo’s Techniques

<table>
<thead>
<tr>
<th>Problem</th>
<th>Technique</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
<td>Consistent Hashing</td>
<td>Incremental Scalability</td>
</tr>
<tr>
<td>High Availability for writes</td>
<td>Vector clocks with reconciliation during reads</td>
<td>Version size is decoupled from update rates.</td>
</tr>
<tr>
<td>Handling temporary failures</td>
<td>Sloppy Quorum and hinted handoff</td>
<td>Provides high availability and durability guarantee when some of the replicas are not available.</td>
</tr>
<tr>
<td>Recovering from permanent failures</td>
<td>Anti-entropy using Merkle trees</td>
<td>Synchronizes divergent replicas in the background.</td>
</tr>
<tr>
<td>Membership and failure detection</td>
<td>Gossip-based membership protocol and failure detection</td>
<td>Preserves symmetry and avoids having a centralized registry for storing membership and node liveness information.</td>
</tr>
</tbody>
</table>

Table 1 presents a summary of the list of techniques Dynamo uses and their respective advantages.
Hybrid Logical Clocks (HLC)

- **Motivation:**
  - No perfectly synchronized physical clock (NTP can introduce 100-250ms clock drift)
  - Logical Clocks does not capture relationship between events in real-time
  - TrueTime requires special hardware and tight clock synchronization protocol
Hybrid Logical Clocks (HLC)

- HLC: combines the benefits of logical clock and physical time
  - One-way causality detection
  - Linear space representation
  - Bounded difference from physical time

Initially \( l.j := 0 \); \( c.j := 0 \)

**Send or local event**

\[
\begin{align*}
l'.j &:= l.j; \\
l.j &:= \max(l'.j, pt.j); \\
\text{If } (l.j = l'.j) \text{ then } c.j &:= c.j + 1 \\
\text{Else } c.j &:= 0; \\
\text{Timestamp with } l.j, c.j 
\end{align*}
\]

**Receive event of message** \( m \)

\[
\begin{align*}
l'.j &:= l.j; \\
l.j &:= \max(l'.j, l.m, pt.j); \\
\text{If } (l.j = l'.j = l.m) \text{ then } c.j &:= \max(c.j, c.m) + 1 \\
\text{Elseif } (l.j = l'.j) \text{ then } c.j &:= c.j + 1 \\
\text{Elseif } (l.j = l.m) \text{ then } c.j &:= c.m + 1 \\
\text{Else } c.j &:= 0; \\
\text{Timestamp with } l.j, c.j 
\end{align*}
\]

Figure 5: HLC algorithm for node \( j \)

pt.j denotes the physical time at node j
Hybrid Logical Clocks (HLC)

- HLC: combines the benefits of logical clock and physical time
  - One-way causality detection
  - Linear space representation
  - Bounded difference from physical time
References

- The Chubby lock service for loosely-coupled distributed systems (OSDI ‘06)
- Bigtable: A Distributed Storage System for Structured Data (OSDI ‘06)
- Dynamo: Amazon’s Highly Available Key-value Store (SOSP ‘07)
- Logical Physical Clocks and Consistent Snapshots in Globally Distributed Databases
- Spanner, TrueTime & The CAP Theorem by Eric Brewer
- Cloud Spanner: TrueTime and external consistency