Distributed Systems

CS425/ECE428

Instructor: Radhika Mittal

Acknowledgements for the materials: Spanner authors
Logistics

- MP2 due soon (Monday).
Distributed Transactions and Replication

• Transaction processing can be distributed across multiple servers.
  • Different objects can be stored on different servers.
  • An object may be replicated across multiple servers.

• Case study: Google’s Spanner System
  • Note for exams:
    • no detailed questions from Spanner paper.
    • only some high-level objective questions from materials in slides.
Spanner: Google’s Globally-Distributed Database

• First three lines from the paper:
  • Spanner is a scalable, globally-distributed database designed, built, and deployed at Google.
  • At the highest level of abstraction, it is a database that shards data across many sets of Paxos state machines in datacenters spread all over the world.
  • Replication is used for global availability and geographic locality; clients automatically failover between replicas.
Spanner: Google’s Globally-Distributed Database

Wilson Hsieh
representing a host of authors
OSDI 2012
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

- User posts
- Friend lists

Locations:
- US: San Francisco, Seattle, Arizona
- Brazil: Sao Paulo, Santiago, Buenos Aires
- Spain: Madrid, Lisbon
- Russia: Moscow, Berlin, Krakow

Numbers: 7x10^6
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

Block writes

Friend1 post
Friend2 post
...
Friend999 post
Friend1000 post

User posts
Friend lists

Generate my page
Multiple Machines

Block writes

Friend1 post
Friend2 post
...
Friend999 post
Friend1000 post

User posts
Friend lists

Generate my page

OSDI 2012
Multiple Datacenters

Friend1 post
  US

Friend2 post
  Spain
  ...

Friend999 post
  Brazil

Friend1000 post
  Russia

Generate my page
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>&lt;8</th>
<th>8</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>My friends</td>
<td>[X]</td>
<td>[]</td>
<td></td>
</tr>
<tr>
<td>My posts</td>
<td></td>
<td></td>
<td>[P]</td>
</tr>
<tr>
<td>X's friends</td>
<td>[me]</td>
<td>[]</td>
<td></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

\[
Pick \ s = \text{now}()
\]
Timestamp Invariants

- Timestamp order == commit order
- Timestamp order respects global wall-time order
TrueTime

• “Global wall-clock time” with bounded uncertainty

\[
\text{TT.now()} \leq 2\epsilon \leq \text{TT.now()}
\]
Timestamps and TrueTime

\[ \text{Pick } s = \text{TT.now().latest} \]

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

Commit wait

Average \( \epsilon \)

Acquired locks

Release locks
Commit Wait and Replication

Start consensus  Achieve consensus  Notify other replicas

Acquired locks

T

Pick s

Release locks

Commit wait done
Commit Wait and 2-Phase Commit

- Acquired locks
- Release locks
- Start logging
- Done logging
- Acquired locks
- Acquired locks
- Released locks
- Committed
- Notify participants of s
- Release locks
- Computed
- Send s
- Prepare s for each
- Commit wait done
- Compute overall s

TC
TP1
TP2
Example

Remove X from my friend list

Remove myself from X's friend list

My friends
My posts
X's friends

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

\[ \text{now} = \text{reference now} + \text{local-clock offset} \]

\[ \varepsilon = \text{reference } \varepsilon + \text{worst-case local-clock drift} \]
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

Date

Epsilon (ms)

Mar 29  Mar 30  Mar 31  Apr 1

Date (April 13)

Epsilon (ms)

99.9
99
90

9AM 8AM 10AM 12PM
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
Thanks

• To the Spanner team and customers
• To our shepherd and reviewers
• To lots of Googlers for feedback
• To you for listening!

• Questions?