CS 244b - spanner

Learning goals:

- Putting it all together
  (2PC, Paxos, linearizability, witnesses)
- The power of real-time clocks... or how to change your assumptions when faced with a hard problem.
<table>
<thead>
<tr>
<th>operation</th>
<th>latency (ms)</th>
<th></th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean</td>
<td>std dev</td>
<td>count</td>
</tr>
<tr>
<td>all reads</td>
<td>8.7</td>
<td>376.4</td>
<td>21.5B</td>
</tr>
<tr>
<td>single-site commit</td>
<td>72.3</td>
<td>112.8</td>
<td>31.2M</td>
</tr>
<tr>
<td>multi-site commit</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.
Figure 1: Spanner server organization.

\[(key, timestamp) \rightarrow value\]
Fig. 2

participant leader

transaction manager

lock table

leader

Paxos

tablet

Colossus

Data Center X

Paxos

tablet

Colossus

Data Center Y

Paxos

tablet

Colossus

Data Center Z

other group's participant leader

other group's participant leader
Figure 3: Directories are the unit of data movement between Paxos groups.
- fine grained locking
- placement flexibility

<table>
<thead>
<tr>
<th># fragments</th>
<th># directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&gt;100M</td>
</tr>
<tr>
<td>2–4</td>
<td>341</td>
</tr>
<tr>
<td>5–9</td>
<td>5336</td>
</tr>
<tr>
<td>10–14</td>
<td>232</td>
</tr>
<tr>
<td>15–99</td>
<td>34</td>
</tr>
<tr>
<td>100–500</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 5: Distribution of directory-fragment counts in F1.
Double log

- Paxos
- Tablet
A directory is also the smallest unit whose geographic-replication properties (or *placement*, for short) can be specified by an application. The design of our placement-specification language separates responsibilities for managing replication configurations. Administrators control two dimensions: the number and types of replicas, and the geographic placement of those replicas. They create a menu of named options in these two dimensions (e.g., *North America, replicated 5 ways with 1 witness*). An application controls how data is replicated, by tagging each database and/or individual directories with a combination of those options. For example, an application might store each end-user’s data in its own directory, which would enable user A’s data to have three replicas in Europe, and user B’s data to have five replicas in North America.
Leaves - assumes bounded clock drift

Strawman #1

2-phase locking

Paxos leader keeps look table

What goes wrong?

A, B concurrently write to different groups

C, D concurrently reading both
Strawman #2

Acquire lock in multiple Paros groups
Use 2PL to commit across groups

Drawbacks

Lots of locking
Lots of reads at leader
Transactions might not make sense w. locking
<table>
<thead>
<tr>
<th>Method</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>TT.now()</code></td>
<td><code>TTinterval: [earliest, latest]</code></td>
</tr>
<tr>
<td><code>TT.after(t)</code></td>
<td>true if <code>t</code> has definitely passed</td>
</tr>
<tr>
<td><code>TT.before(t)</code></td>
<td>true if <code>t</code> has definitely not arrived</td>
</tr>
</tbody>
</table>

Table 1: TrueTime API. The argument `t` is of type `TTstamp`.

\[
drift \leq 200 \text{\mu sec/sec} \quad 0.0290\%
\]

A, B - each tx has timestamp
Read-write transaction - 1 Paxos

Client acquires read locks buffers writes
Send commit request to leader
Leader picks timestamp S
S > previous Paxos writes has to be in leader's lease term
S > T(\text{now}()). Latest Commit to Paxos
Wait until S < T(\text{now.earliest}())
RW transaction to multiple Paxos groups

Client picks coordinator - send req to leader

Coord leader broadcasts VOTE-REQ

Send VOTE-Commit w. prepare TS, which > TS of committed

in participant leader's lease term

Coord will pick S

S > floor( ), latest when commit req received

S ≥ max (prepare TS)

S has to be in all leader lease term S

Commit Wait
RO transaction

safest: s = Thrthrow_lates
better: 1 Paxos group
LastTS()

t_{safe} \geq t_{im}\tau
<table>
<thead>
<tr>
<th>replicas</th>
<th>latency (ms)</th>
<th>throughput (Kops/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>write</td>
<td>read-only transaction</td>
</tr>
<tr>
<td>1D</td>
<td>9.4±.6</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>14.4±1.0</td>
<td>1.4±.1</td>
</tr>
<tr>
<td>3</td>
<td>13.9±.6</td>
<td>1.3±.1</td>
</tr>
<tr>
<td>5</td>
<td>14.4±.4</td>
<td>1.4±.05</td>
</tr>
</tbody>
</table>

Table 3: Operation microbenchmarks. Mean and standard deviation over 10 runs. 1D means one replica with commit wait disabled.

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

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