Paxos Quorum Leases

Sayed Hadi Hashemi
BACKGROUND
• Setting
  – Status: Key-Value Storage
  – Commands: Read / Write / Batch (Read, Write)
  – Goal: Minimized WAN Delay

• Original Paxos
  – Read: At least 2 RT (more in case of dueling leaders)
  – Write: At least 2 RT
Can we do any better?

**Paxos**

- Request (CMD)
- Prepare
- OK
- Accept (CMD)
- Accept (OK)
- Committed (CMD)
- Result / ACK
• **Multi Paxos**
  – Temporary Stable Leader Replica to ignore Prepare (election) phase
  – Read: 1 RT from the leader
  – Write is the same as the read
  – A replica becomes the stable leader by running the prepare phase for a large number of instances at the same time, taking ownership of all of them.

• **Google’s Megastore**
  – All Replica are leader!
  – Read: 0 RT from any Replica! (Reading Locally)
  – Write: At least 1 RT to All Replica
Steady state interaction in Multi-Paxos. The asynchronous messages are represented as dashed arrows.
Megastore

Client

Request (Read) → Result → Request (Write) → ACK

Replica 0

Accept (OK)

Replica 1

Accept (CMD)

Replica 2

Committed (CMD)

RT
Can we have benefits of the both?

• Quorum Leases
  – Middle ground
  – Read: Most of the time 0 RT (80% in the experiment), 1 RT otherwise
  – Write is almost the same as the Multi Paxos
QUORUM LEASES
Overview

• The idea is to have multiple leases for different sets of objects
• Each lease is granted to lease holders by a majority of grantors
• Read:
  – Lease holders can read locally while the lease is active
  – Any one else, use Multi-Paxos
• Write:
  – Notify Lease holders synchronously through Lease Grantors (Majority)
Figure 2. Leasing with and without revocation.
Figure 3. An example lease in which a majority of replicas (R1, R2, and R5) have granted leases to two lease holders (R4 and R5).
• **Lease Configuration**
  
  – Describes the set of granted objects to quorum leases
  
  • Replica is added to a lease if it reads an object frequently
  
  • Replica is removed from a lease if it fails, or it stop reading an object frequently

• **Granting and Refreshing leases**
  
  – \(|N+1|/2\) grantors will activate a lease for a set of holders

  – Grantor Promise Holder that:
    
    • Notify r synchronously before committing any update
    
    • Acknowledge “Accept” and “Prepare” for writing with the condition that the proposer must notify r synchronously
Lease Configuration

- Describes the set of granted objects to quorum leases
  - Replica is added to a lease if it reads an object frequently
  - Replica is removed from a lease if it fails, or it stop reading an object frequently

- Steps:
  - Replicas track the frequency of reads and sends this information to the leader
  - Leader periodically uses this tracking information to update the lease configuration
  - Lease Configuration Changes are distributed using another instance of Paxos
Granting and Refreshing leases

- Grantor Promise Holder that:
  - Notify r synchronously before committing any update
  - Acknowledge “Accept” and “Prepare” for writing with the condition that the proposer must notify r synchronously

- Establish:
  - Guard
  - send Promise to every other replica
  - Optional ACK

- Renew:
  - Promise, ACK

- Failed Holders
  - Grace
  - Lease Configuration
  - Wait
1. if Promise ACK has received
2. if Promise ACK has **not** received
1. if Promise ACK has received
2. if Promise ACK has not received
Establishing leases

Every replica $R$ becomes a grantor:

1. send $Guard(guard\_duration)$ to every other replica
2. for every $GuardACK$ from any replica $H$ do
3. set $grant\_timer_R[H] \leftarrow guard\_duration + lease\_duration$
4. send $Promise(lease\_duration)$ to $H$
5. for every $PromiseReply$ from any replica $H$ do
6. if reply received before $grant\_timer_R[H]$ expired then
7. set $grant\_timer_R[H] \leftarrow lease\_duration$

Any replica $H$, on receiving a $Guard(guard\_duration)$ from a replica $R$:

8. set $guard\_timer_H[R] \leftarrow guard\_duration$
9. reply with a $GuardACK$
10. wait for a $Promise(lease\_duration)$ from $R$
11. if Promise received before $guard\_timer_H[R]$ expires then
12. set $lease\_timer_H[R] \leftarrow lease\_duration$
13. reply with $PromiseReply$ to $R$

{A lease holder $H$ can consider the lease active if at least $\lceil N/2 \rceil$ promises from different replicas have yet to expire (where $N$ is the total number of replicas).}

Renewing leases

Every replica $R$ that is a grantor:

14. for every other replica $H$ do
15. set $grant\_timer_R[H] \leftarrow lease\_duration + guard\_duration$
16. set $t' \leftarrow$ the time since the most recent ACK from $H$
17. set $seq_{ACK} \leftarrow$ the sequence number of most recent ACK from $H$
18. send $Promise(lease\_duration, t', seq_{ACK})$ to $H$
19. for every $PromiseReply$ from any replica $H$ do
20. set $grant\_timer_R[H] \leftarrow \min(grant\_timer_R[H], lease\_duration)$

Any replica $H$, on receiving a $Promise(lease\_duration, t', seq_{ACK})$ from a replica $R$:

20. if Promise received before time $t' + guard\_duration$ since sending ACK with sequence $seq_{ACK}$ then
21. set $lease\_timer_H[R] \leftarrow lease\_time$
22. reply with $PromiseReply$ to $R$
EVALUATION
Evaluation

• Run implementations of quorum leases, classic leader leases and Megastore-type leases
• Geo-distributed Amazon EC2 cluster.
• 5 Multi-Paxos replicas in Virginia, Northern California, Oregon, Ireland and Japan.
• 10 Client co-located in each replica
• Workload
  – YCSB key-value workload (Zipf)
  – Uniform key-value workload
Selects as leader because of Low RTT

<table>
<thead>
<tr>
<th></th>
<th>JP</th>
<th>CA</th>
<th>OR</th>
<th>VA</th>
<th>IRL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan</td>
<td>0.4</td>
<td>120</td>
<td>120</td>
<td>180</td>
<td>270</td>
</tr>
<tr>
<td>California</td>
<td>0.4</td>
<td>20</td>
<td>85</td>
<td>150</td>
<td></td>
</tr>
<tr>
<td>Oregon</td>
<td></td>
<td>0.4</td>
<td>75</td>
<td>170</td>
<td></td>
</tr>
<tr>
<td>Virginia</td>
<td></td>
<td></td>
<td>0.4</td>
<td>92</td>
<td></td>
</tr>
<tr>
<td>Ireland</td>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td></td>
</tr>
</tbody>
</table>

**Table 1.** Approximate round-trip times between datacenters in milliseconds.
Test1: Latency Evaluation

- Multi-Paxos Leader: Northern California
- Each client sends 10000 request to its co-located replica
- Request:
  - 1:1 Read-Write
  - 9:1 Read-Write
- Parameters:
  - lease duration: 2s, renew duration: 500ms, lease configuration update: every 10s
Figure 6. CDFs of client-observed latency for each site, with all three lease techniques: quorum lease (QL), single leader lease (LL), and Megastore-type lease (ML). QL-uniform corresponds to quorum leases for a uniformly-distributed workload. The read-to-write ratio in these experiments was 1:1. The Multi-Paxos leader is always located in California. Note the log scale on the X axis.

LL is the best in writing, ML in reading
<table>
<thead>
<tr>
<th>Location</th>
<th>Fast local reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan</td>
<td>81%</td>
</tr>
<tr>
<td>California</td>
<td>95%</td>
</tr>
<tr>
<td>Oregon</td>
<td>89%</td>
</tr>
<tr>
<td>Virginia</td>
<td>89%</td>
</tr>
<tr>
<td>Ireland</td>
<td>81%</td>
</tr>
</tbody>
</table>

**Table 2.** Percentages of fast local reads (smaller than 10 ms) for wide-area quorum leases with 10% writes and 90% reads, Zipf-distributed.
Test2: Recovering from a Replica Failure

• Shutdown a (non leader) replica, 10s after starting the test (Lease Configuration Update)

• Parameters:
  – Guard duration: 2s, Grace delay: 5s, lease duration: 2s, renew duration: 500ms, lease configuration update: every 10s

• Recover time:
  – Update + Grace + Guard + Lease
Figure 7. Latency of write requests over time. Ten seconds into the experiment, a non-leader replica fails.
Test3: Throughput in a Cluster

- Run in one local cluster (no geo-distributed)
- Requests are generated open-loop by one client for each replica
- 2 Situations:
  - (1) different objects are popular at different replicas
  - (2) clients direct their reads uniformly at random across all replicas.
- Use batching to commit writes (the leader batches up to 5000 updates at a time)
**Figure 8.** Local-area read and write throughput for different leasing strategies. The “Uniformly-distributed reads” for quorum leases corresponds to the situation when clients do not know which replicas can read locally which objects. Error bars represent 95% confidence intervals.
REVIEW
Pro

• Strong Consistency
• Acceptable Availability
• Combine the best of two approaches
• Using objects, instead of Replica
• Separating “Lease Configuration Updates” than the other operations
• Compatibility with Multi-Paxos (or other implementations)
Cons

• What is the messaging overhead?
  – Lease Renewal
  – Lease Configuration

• Experiment
  – 1:1 Read-Write Ratio vs. 9:1

• Recovery Time in Practice:
  – Update + Grace + Guard + Lease
  – Worse case +20s
Thanks for your attention

QUESTIONS?