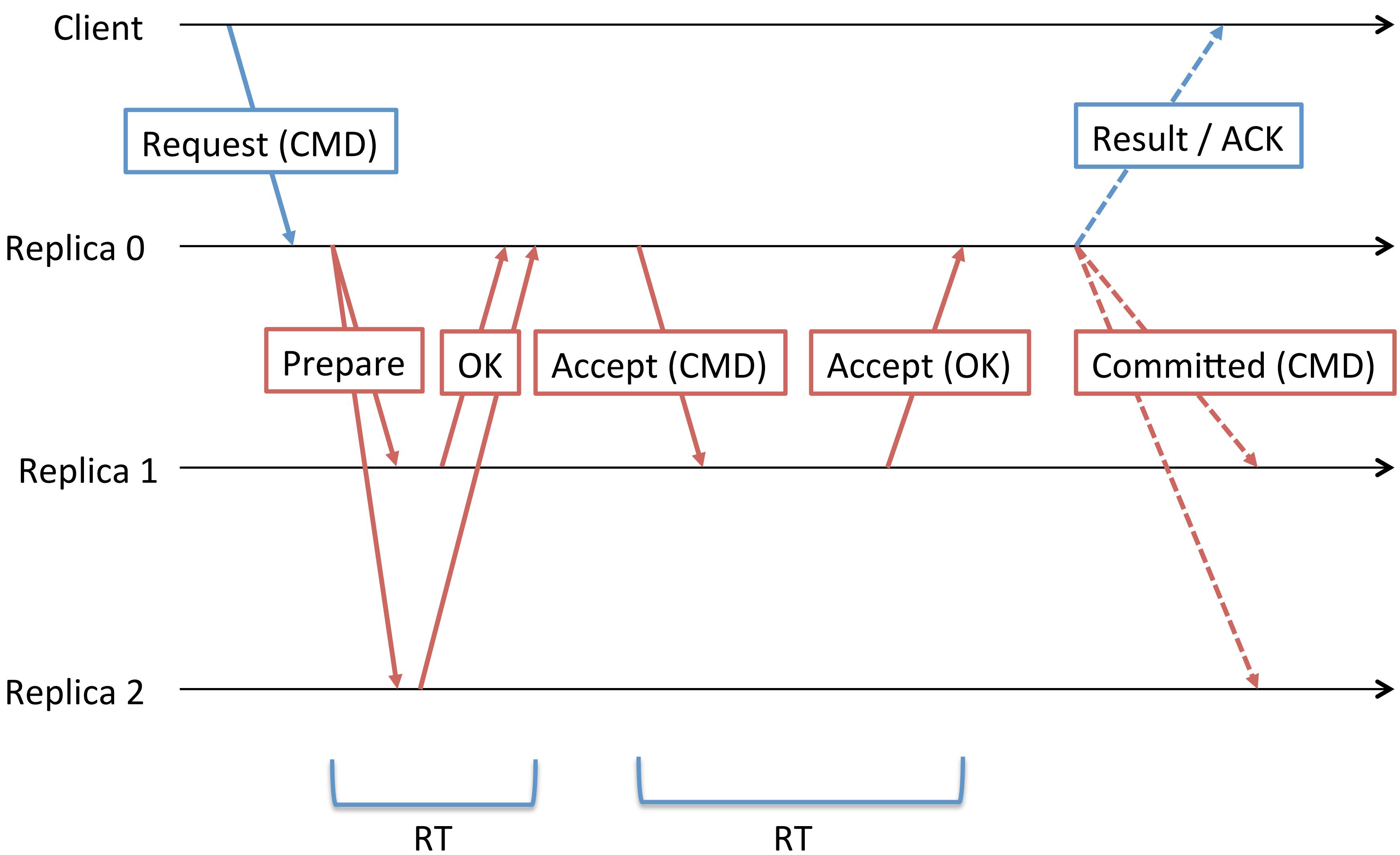
Paxos Quorum Leases

Sayed Hadi Hashemi

BACKGROUND

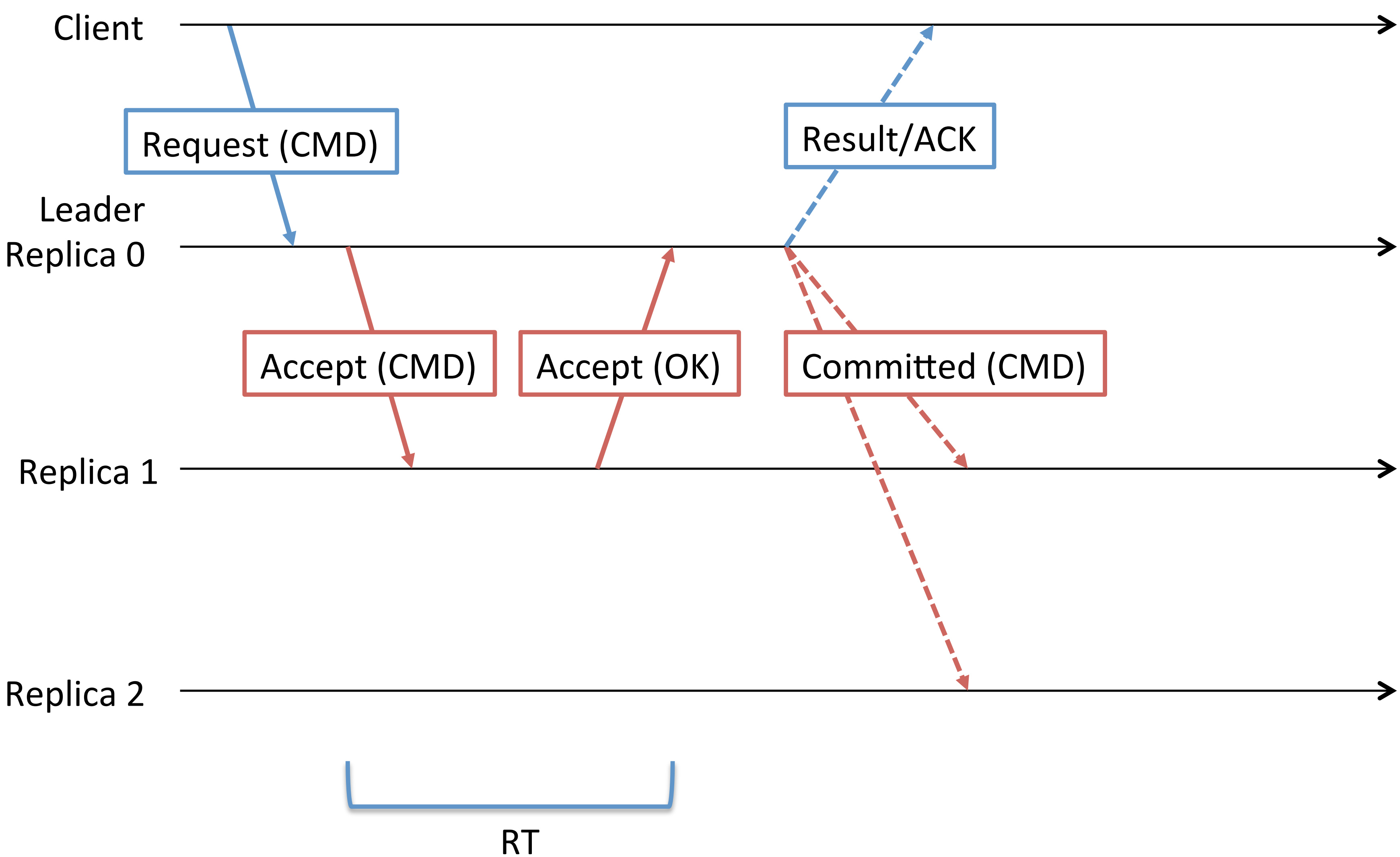
- Setting
  - Status: Key-Value Storage
  - Commands: Read / Write / Batch (Read, Write)
  - Goal: Minimized WAN Delay
- Original Paxos
  - Read: At least 2 RT (more in case of dueling leaders)
  - Write: At least 2 RT

Paxos
Can we do any better?

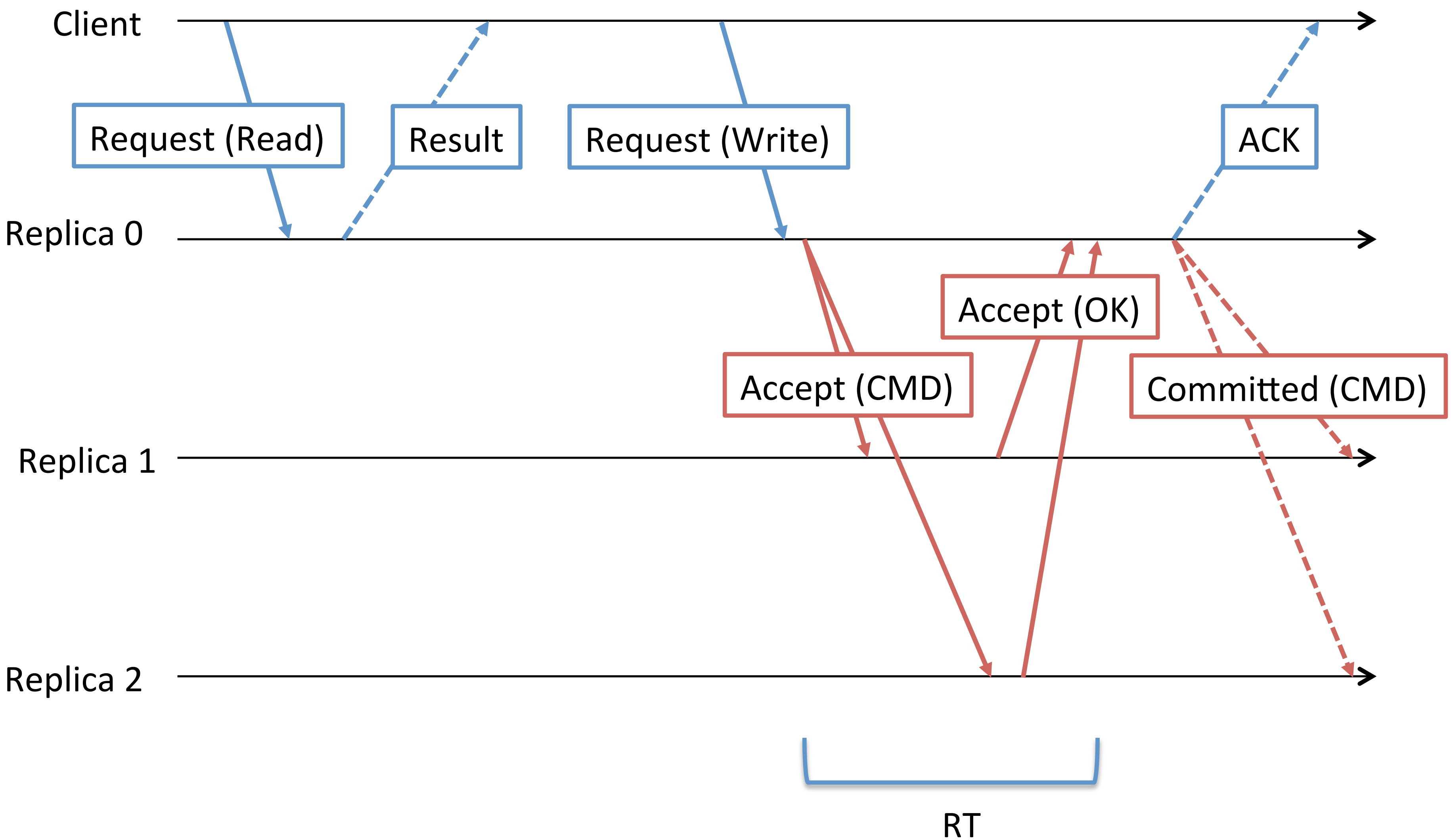


- Multi Paxos
  - Temporary Stable Leader Replica to ignore Prepare (election) phase
  - Read: 1 RT from the leader
  - Write is the same as the read
  - A replica becomes the stable leader by running the prepare phase for a large number of instances at the same time, taking ownership of all of them.
- Google's Megastore
  - All Replica are leader!
  - Read: 0 RT from any Replica! (Reading Locally)
  - Write: At least 1 RT to <u>All</u> Replica

Steady state interaction in Multi-Paxos.
The asynchronous messages are represented as dashed arrows.



Megastore


Client
Replica 0
Replica 1
Replica 2
Request (Read)
Result
Request (Write)
ACK
Accept (OK)
Accept (CMD)
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)

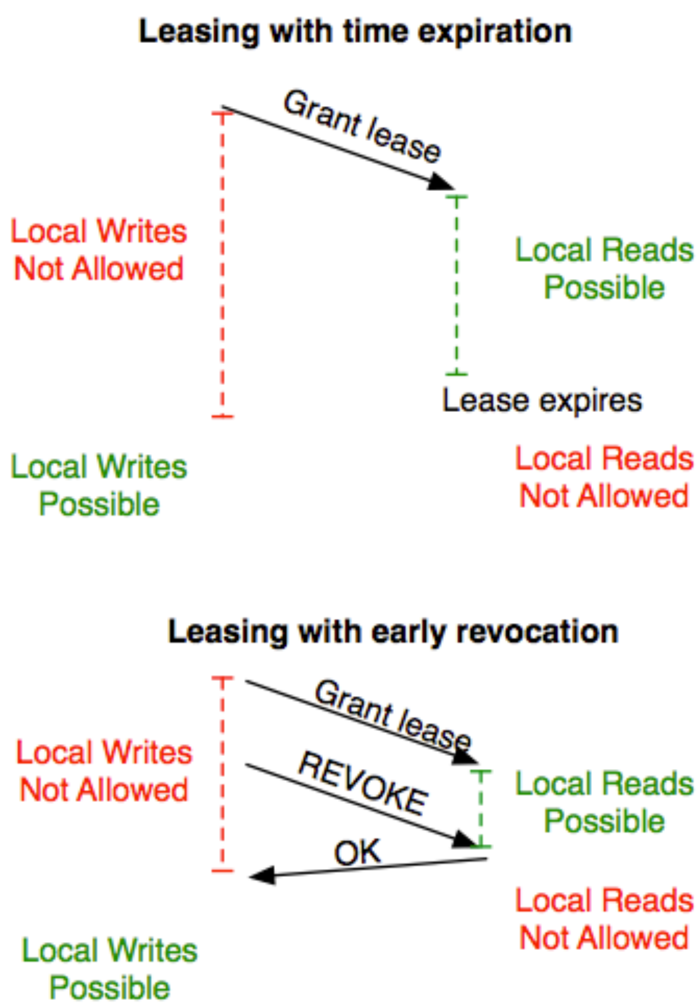


Figure 2. Leasing with and without revocation.

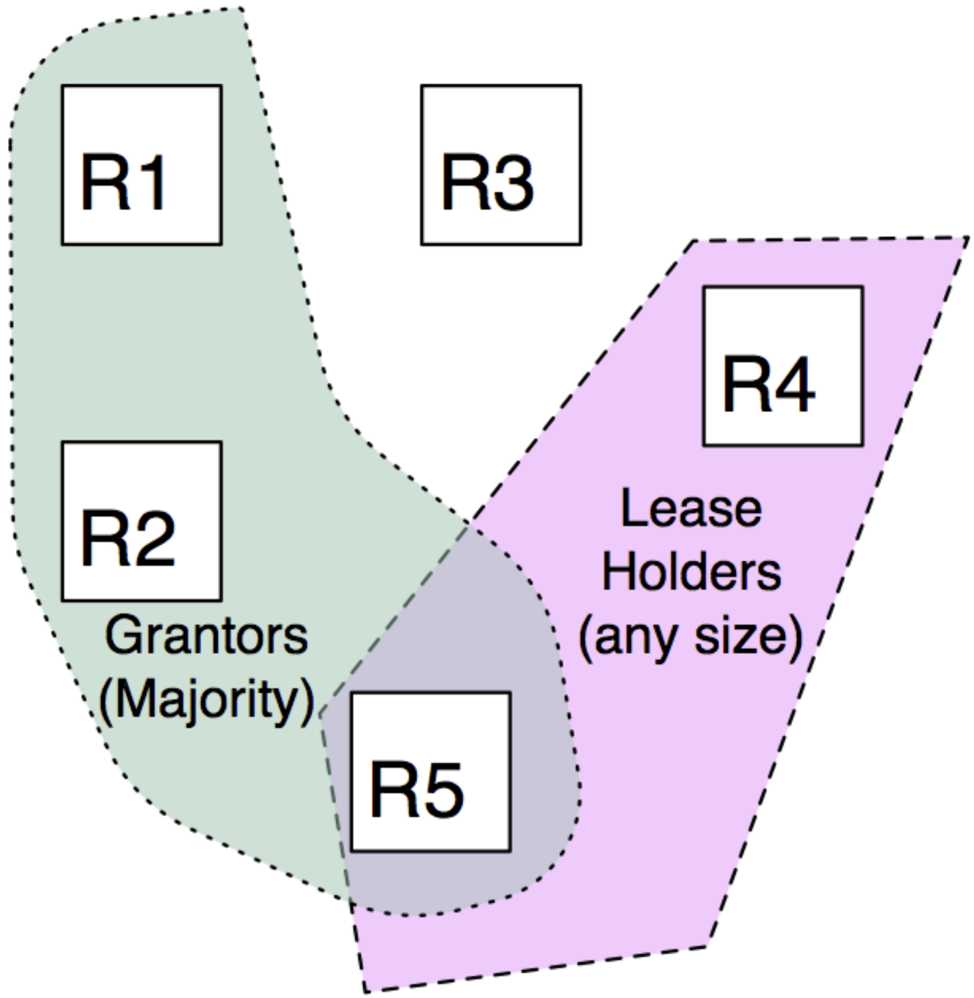


Figure 3. An example lease in which a majority of replicas (R1, R2, and R5) have granted leases to two lease holders (R4 and R5).

- 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

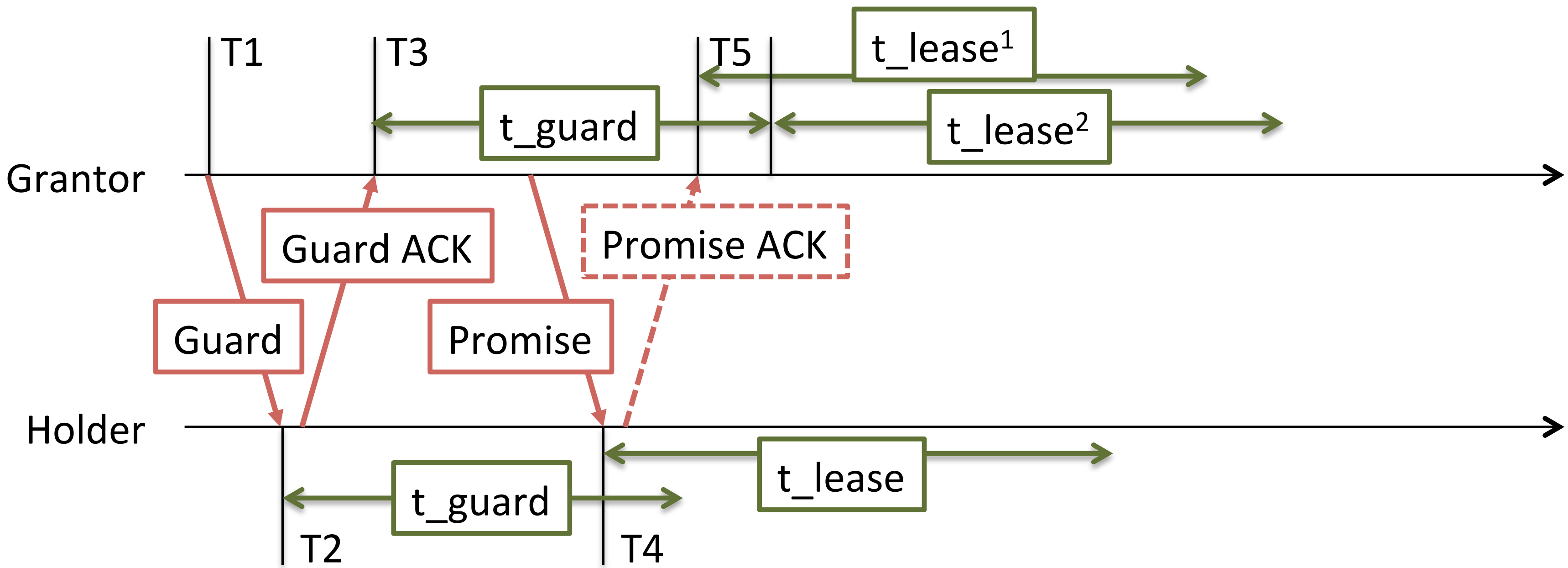
Lease Configuration

- 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

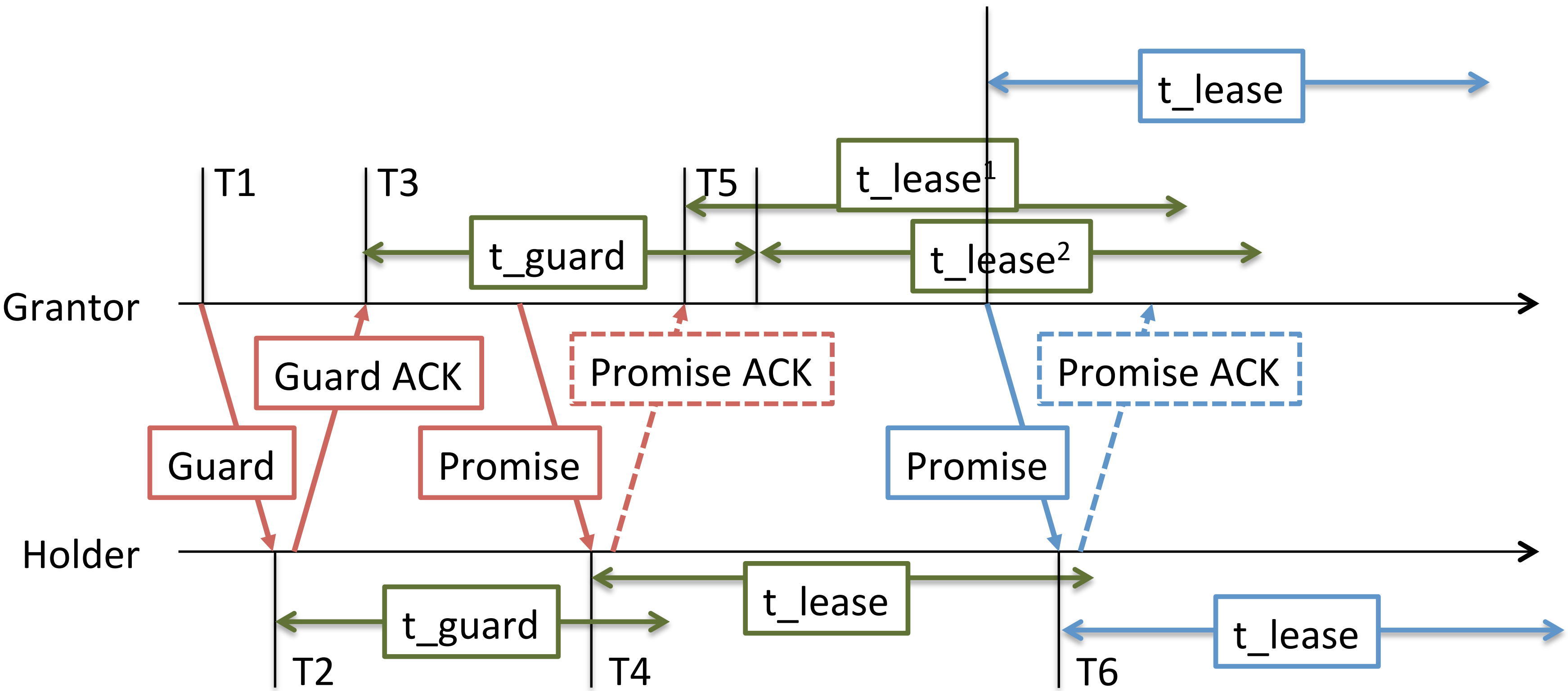
- Grantor Promise Holder that:
  - Notify r synchronously before committing any update
  - Acknowledge "Accept" and "Prepare" for writing with the condition that the proposer must notify r synchronously
- Establish:
  - Guard
  - send Promise to every other replica
  - Optional ACK
- Renew:
  - Promise, ACK
- Failed Holders
  - Grace
  - Lease Configuration
  - Wait

Grant Lease



1. if Promise ACK has received
2. if Promise ACK has **not** received

Grant Renew



1. if Promise ACK has received
2. if Promise ACK has **not** received

Establishing leases

Every replica *R* becomes a grantor:

send *Guard(guard_duration)* to every other replica
for every *GuardACK* from any replica *H* **do**
    set $grant\_timer_R[H] \leftarrow guard\_duration + lease\_duration$
    send *Promise(lease_duration)* to H
for every *PromiseReply* from any replica *H* **do**
    if reply received before $grant\_timer_R[H]$ expired **then**
        set $grant\_timer_R[H] \leftarrow lease\_duration$

Any replica *H*, on receiving a *Guard*(*guard_duration*) from a replica *R*:

set $guard\_timer_H[R] \leftarrow guard\_duration$
reply with a *GuardACK*
wait for a *Promise(lease_duration)* from *R*
if *Promise* received before $guard\_timer_H[R]$ expires **then**
    set $lease\_timer_H[R] \leftarrow lease\_duration$
    reply with *PromiseReply* to *R*

Renewing leases

Every replica *R* that is a grantor:

for every other replica *H* **do**
    set $grant\_timer_R[H] \leftarrow lease\_duration + guard\_duration$
    set $t' \leftarrow$ the time since the most recent ACK from *H*
    set $seq_{\mathrm{ACK}} \leftarrow$ the sequence number of most recent ACK from *H*
    send *Promise(lease_duration, t', seq_{ACK})* to *H*
for every *PromiseReply* from any replica *H* **do**
    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*:

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

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

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

	JP	CA	OR	VA	IRL
Japan	0.4	120	120	180	270
California		0.4	20	85	150
Oregon			0.4	75	170
Virginia				0.4	92
Ireland					0.4

Table 1. Approximate round-trip times between datacenters in milliseconds.

Selects as leader because of Low RTT

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

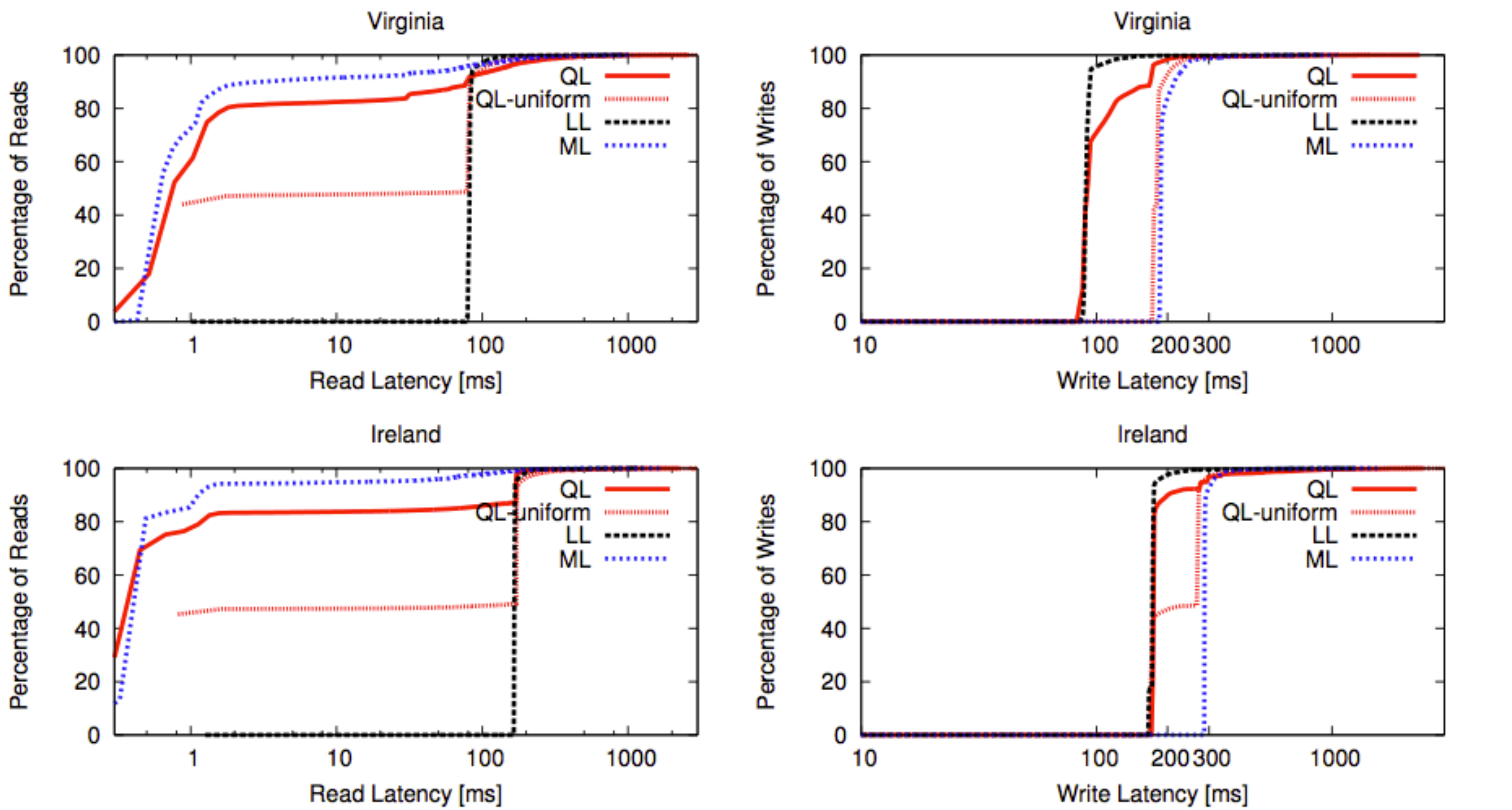
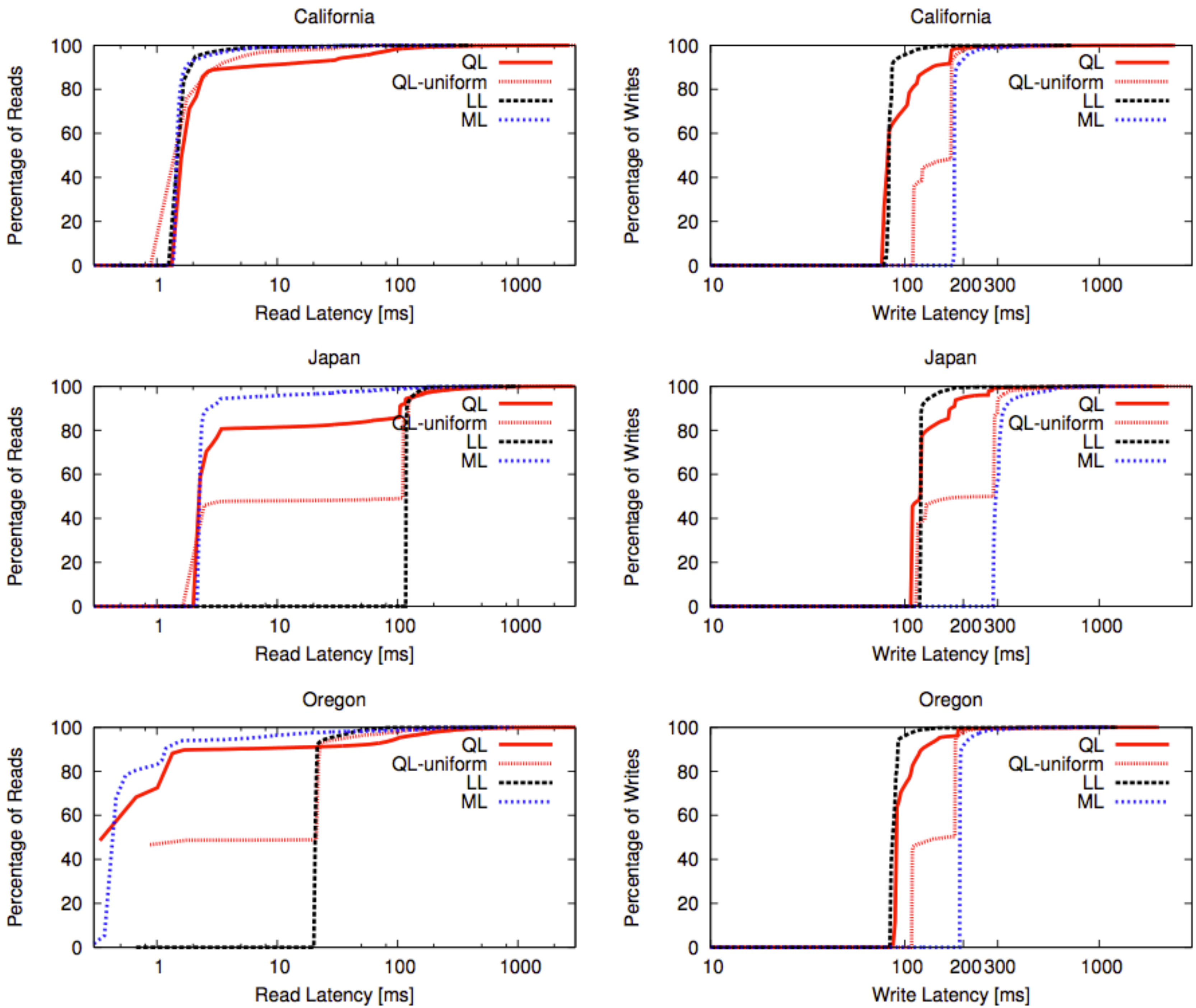


Figure 6. CDFs of client-observed latency for each site, with all three lease techniques: quorum lease (QL), single leader lease (LL), and Megastore-type lease (ML). QL-uniform corresponds to quorum leases for a uniformly-distributed workload. The read-to-write ratio in these experiments was 1:1. The 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


California
Percentage of Reads
Read Latency [ms]
Percentage of Writes
Write Latency [ms]
Japan
Oregon
QL
QL-uniform
LL
ML
0
20
40
60
80
100
1
10
100
1000
200
300

	Fast local reads
Japan	81%
California	95%
Oregon	89%
Virginia	89%
Ireland	81%

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

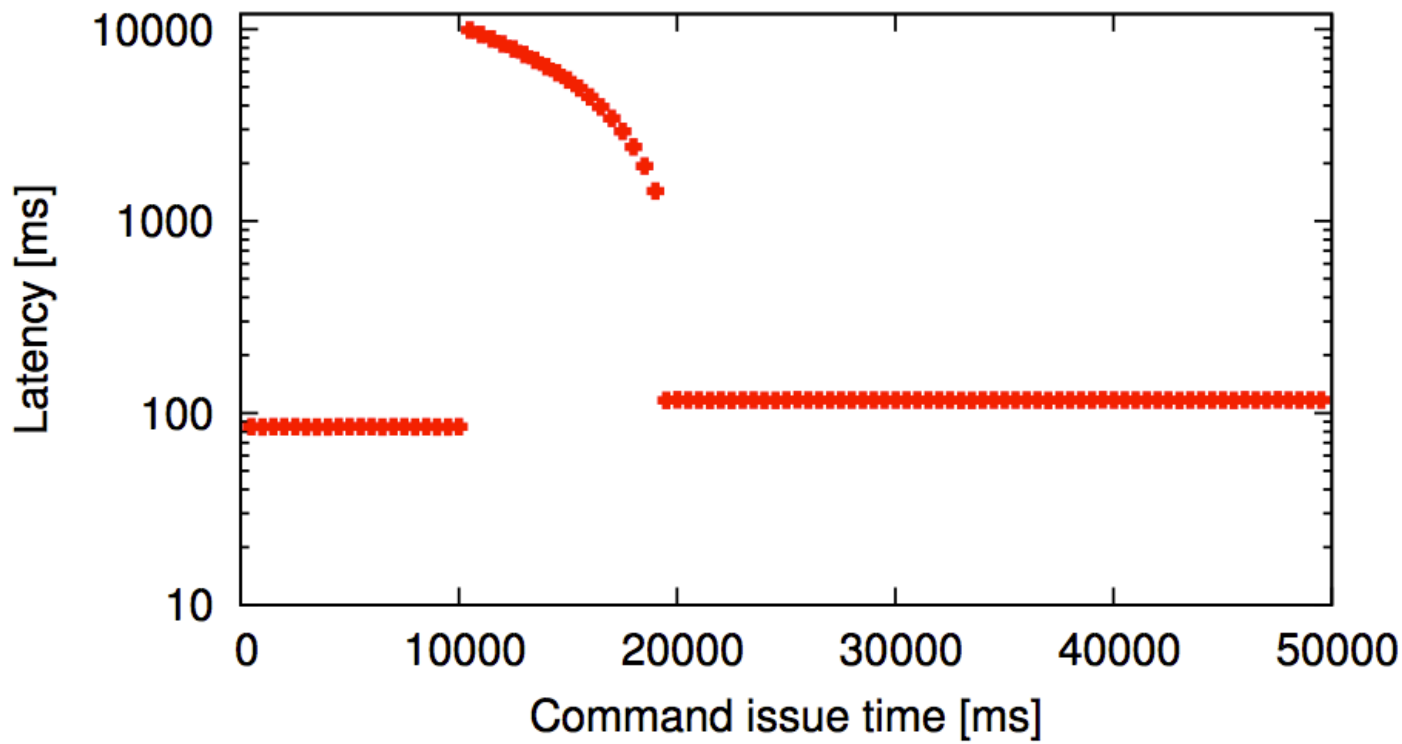


Figure 7. Latency of write requests over time. Ten seconds into the experiment, a non-leader replica fails.

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)

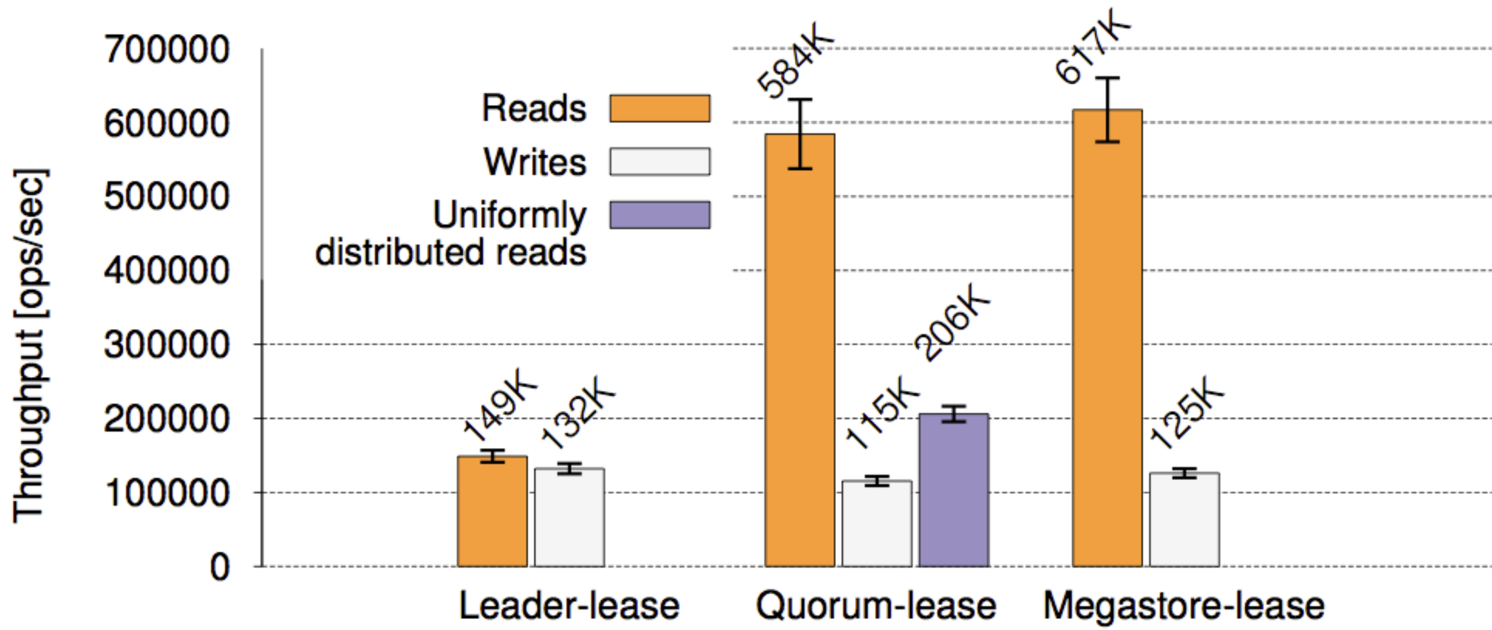


Figure 8. Local-area read and write throughput for different leasing strategies. The "Uniformly-distributed reads" for quorum leases corresponds to the situation when clients do not know which replicas can read locally which objects. Error bars represent 95% confidence intervals.

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?