LevelUpDB

A Cloud-Native version of LevelDB

Benita Kathleen Britto
Krishna Ganeriwal
Reetuparna Mukherjee
Shreyansh Sharma

Design

Specifications

LEVELDB: CLOUD KEY-VALUE DATA STORAGE

RAFT: TO REACH CONSENSUS

GRPC C++: FOR RPC

TUNABLE CONSISTENCY: STRONG AND EVENTUAL

LOAD BALANCER: TO DISTRIBUTE LOAD AMONG PARTICIPATING NODES AND MAINTAIN SYSTEM STATE

DYNAMIC RESOURCE ALLOCATOR: TO SUPPORT ON DEMAND ADDITION OR REMOVAL OF NODES

Assumptions

- No network partition
- Put and Get APIs support only string data type
- Batch Update/Creation not supported
- Put API for both KV Creation and Update
- Load Balancer will always be up
- Sufficient memory for commit logs on each node
- Cannot tolerate Byzantine Faults
- Does not account for security
- No disk failures and other forms of data corruption

APIs

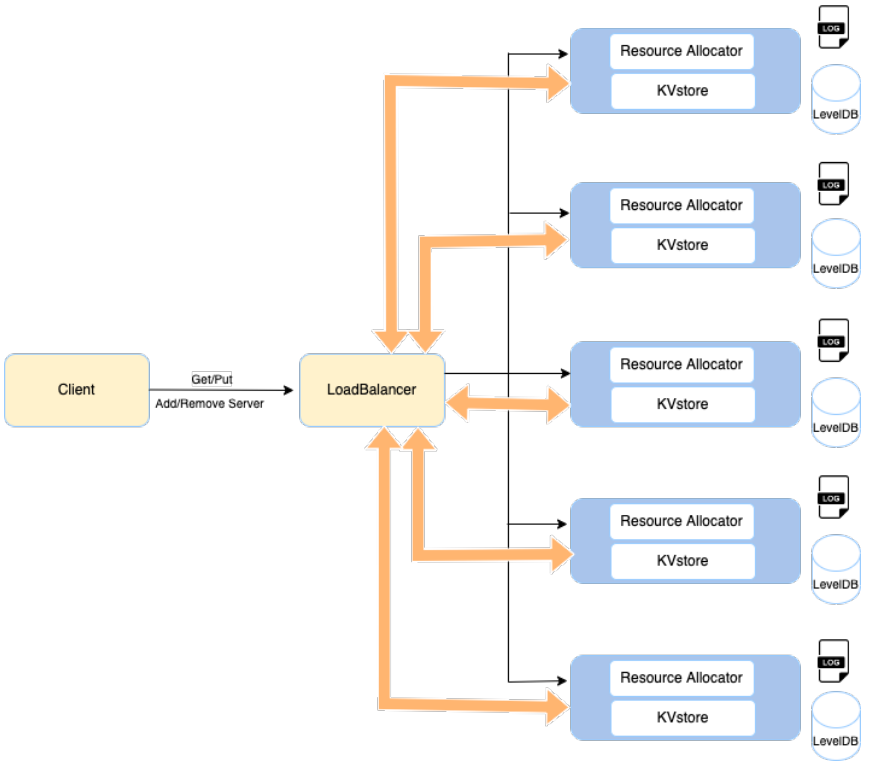
Get()

Put()

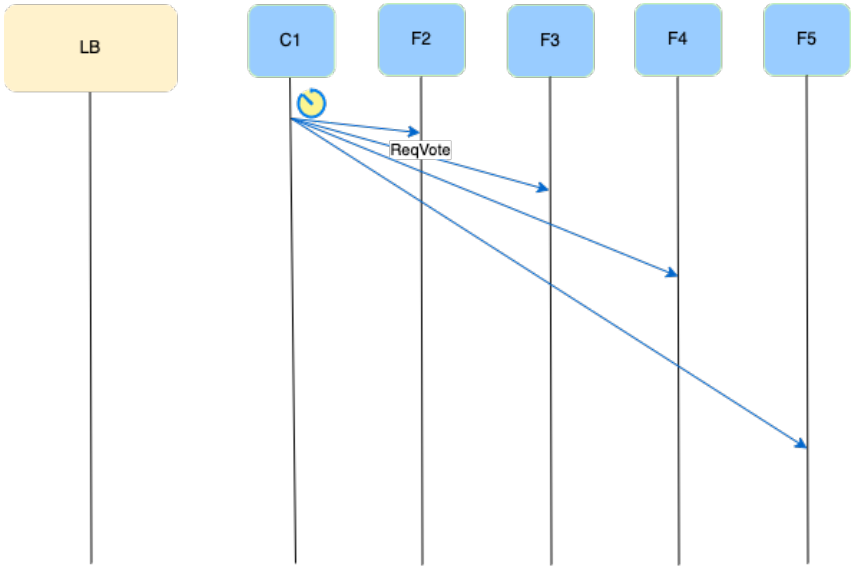
AddServer()

DeleteServer()

System Overview

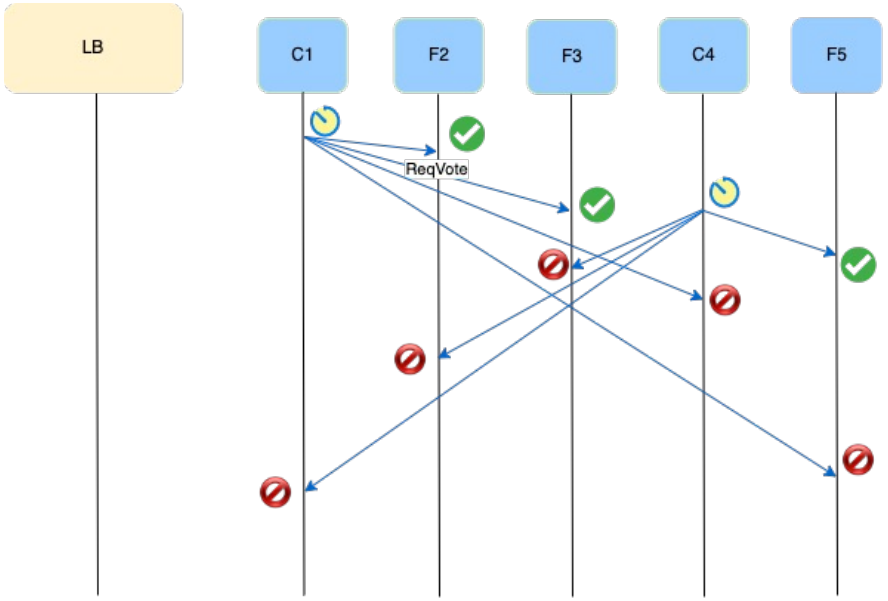


Leader Election



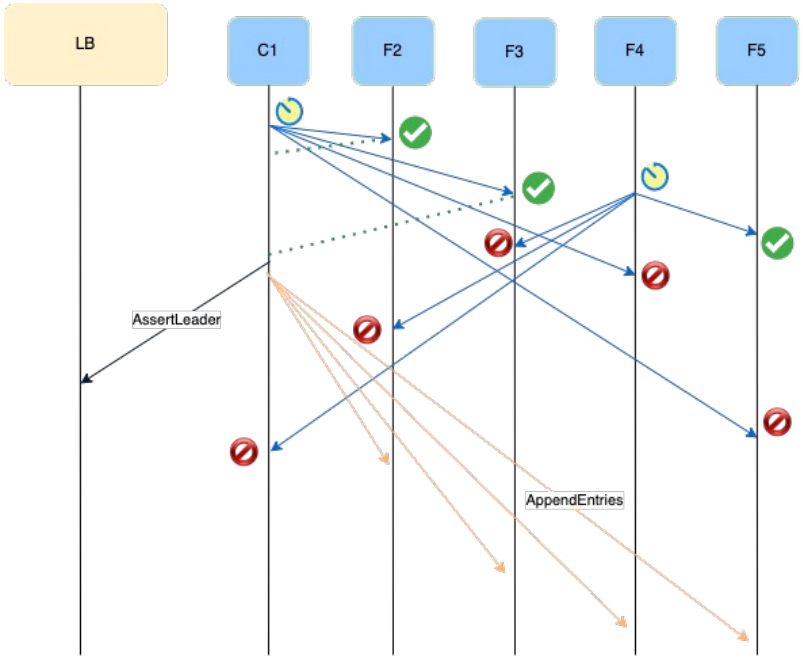
LB: Load Balancer
F: Follower
C: Candidate

Leader Election



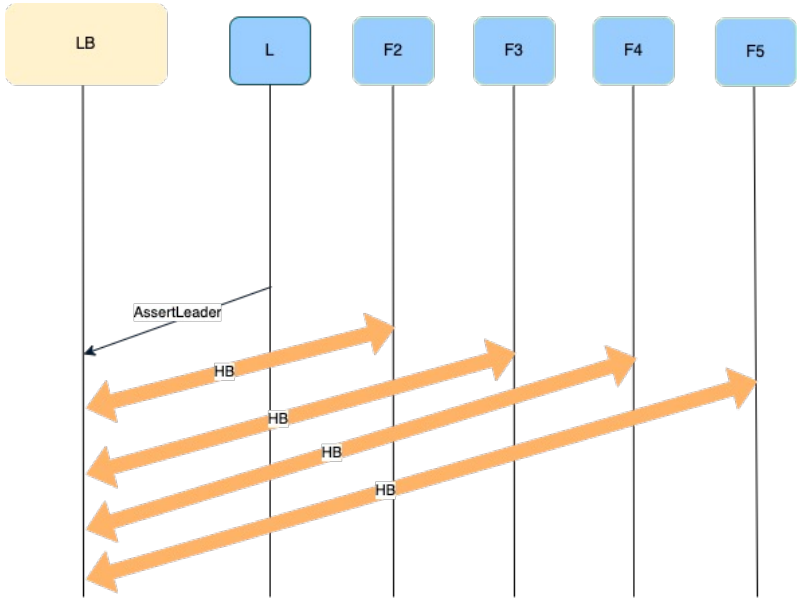
LB: Load Balancer
F: Follower
C: Candidate

Leader Election



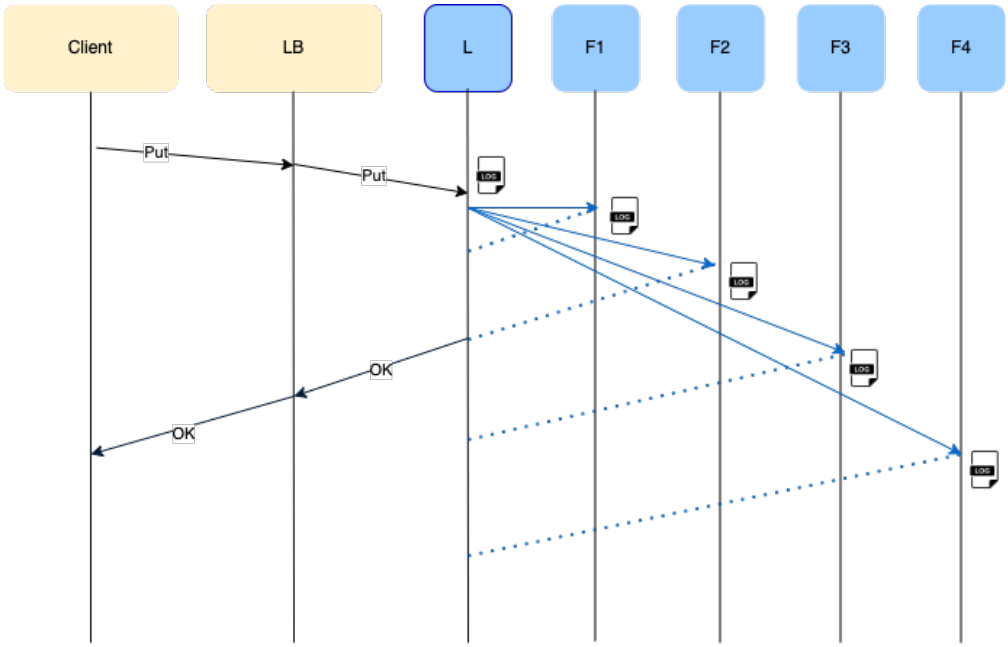
LB: Load Balancer
C: Candidate
F: Follower

Leader Election



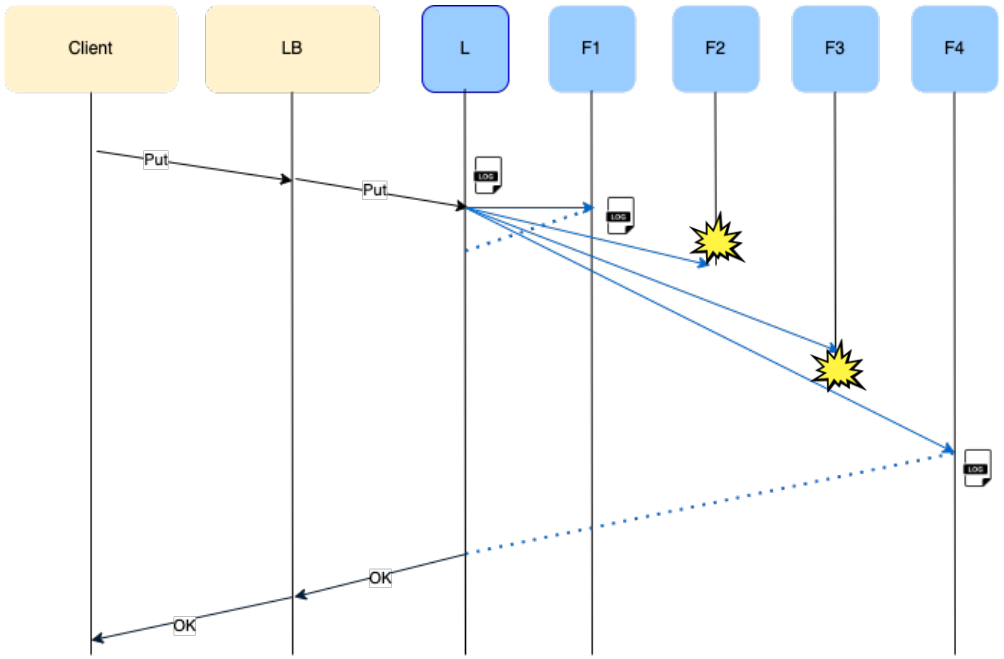
LB: Load Balancer
F: Follower
L: Leader

Put without Crash



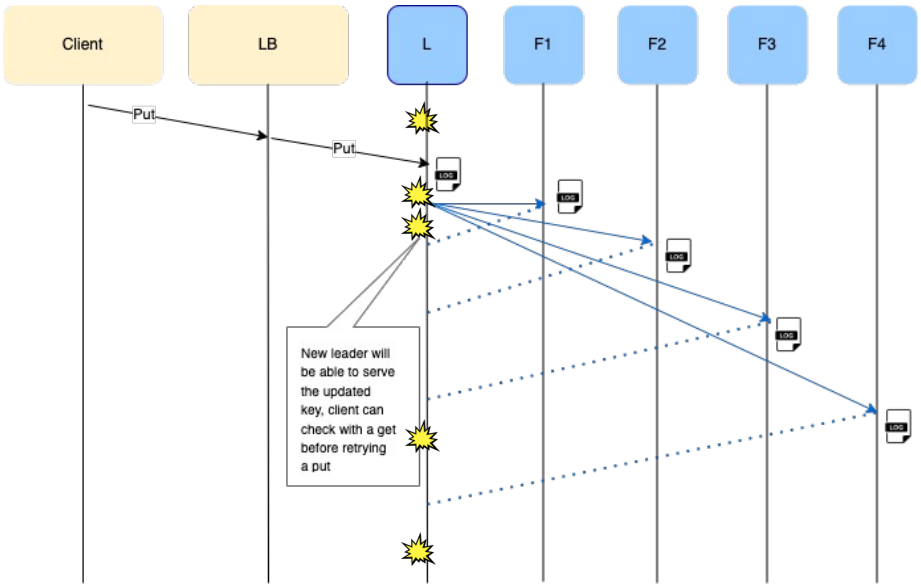
LB: Load Balancer
L: Leader
F: Follower

Put with Crash on Follower



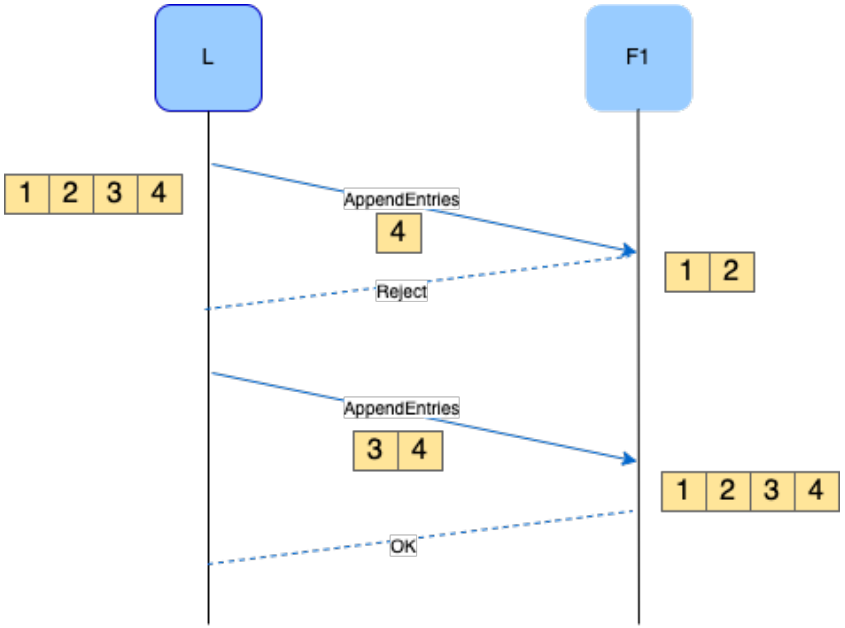
LB: Load Balancer
L: Leader
F: Follower

Put with Crash on Leader



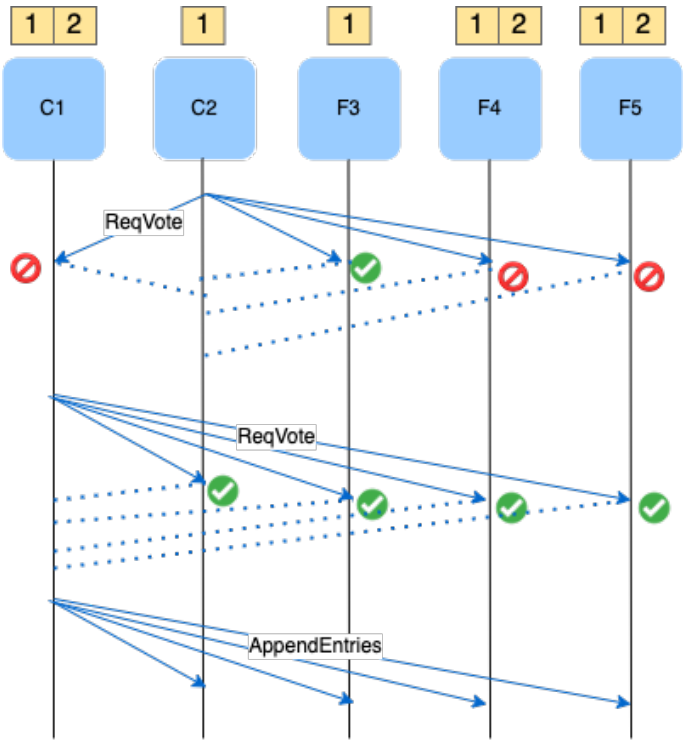
LB: Load Balancer
L: Leader
F: Follower

Correctness: Append Entries



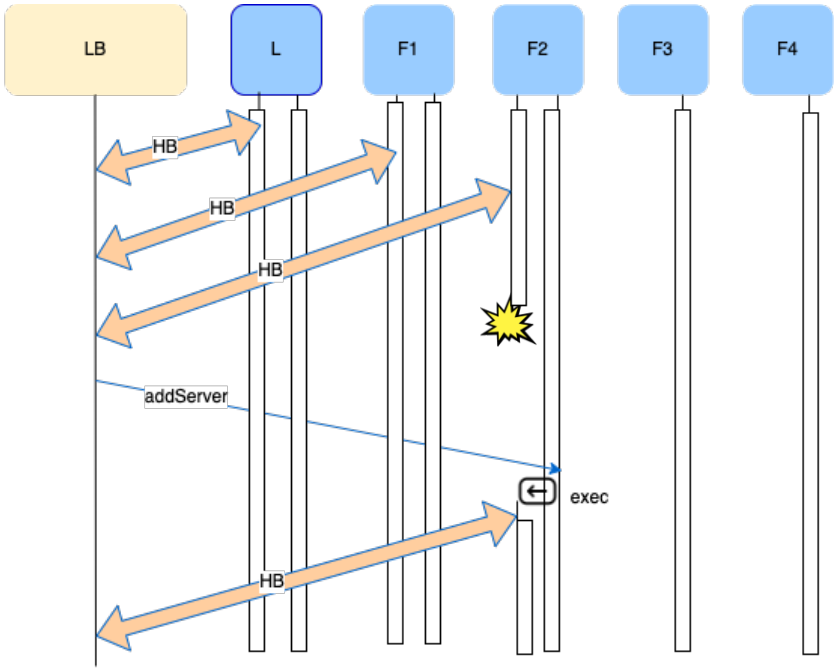
L: Leader
F: Follower

Correctness: Leader Election



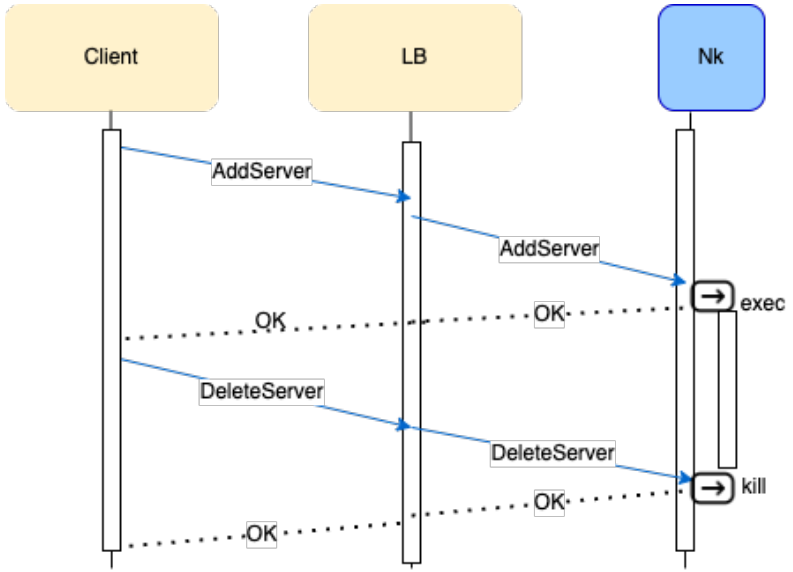
C: Candidate
F: Follower

Resource Allocator: Autoscaling



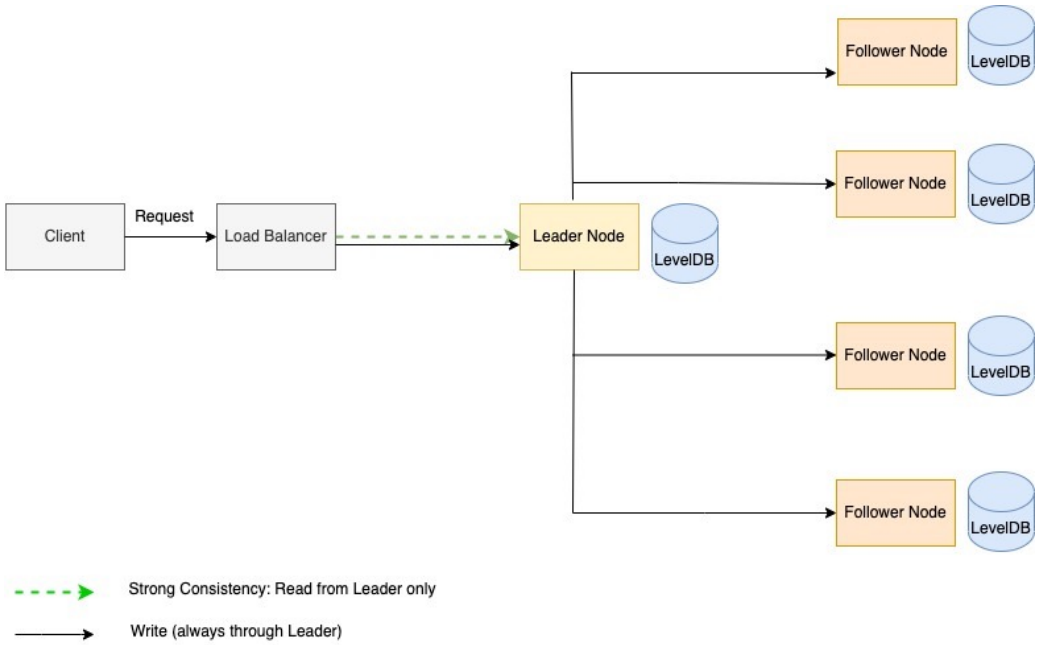
LB: Load Balancer
L: Leader
F: Follower

Resource Allocator: Scaling out and in

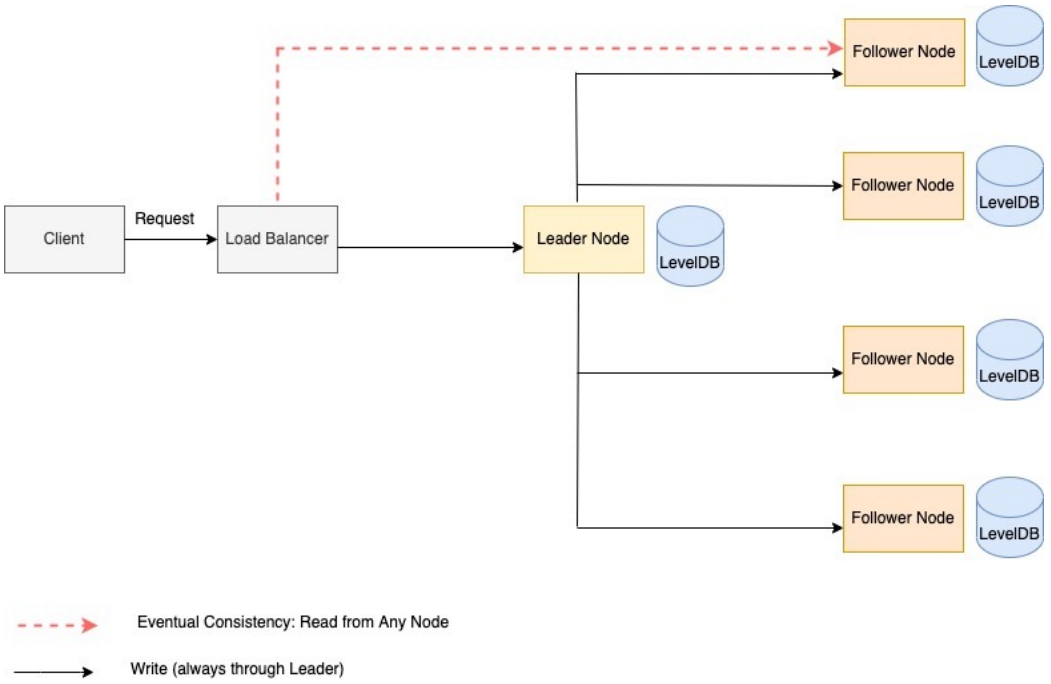


LB: Load Balancer
L: Leader
N: Node

Get:
Strong Consistency



Get:
Eventual Consistency (Quorum of 1)



Measurements

Testing Strategy and System Specs

Crash points: Macros that cause a seg fault

Automated scripts for Get() and Put()

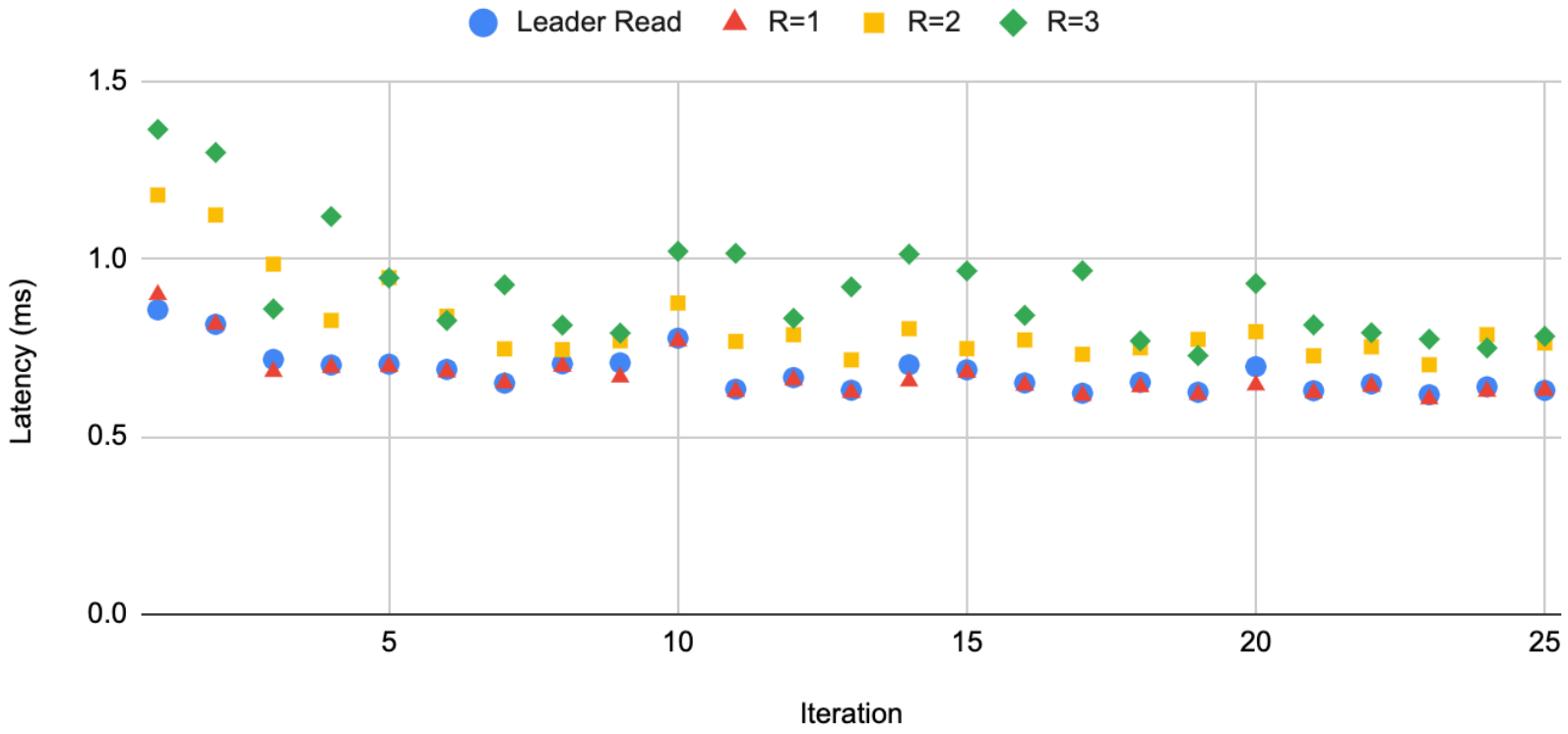
System Details :

CloudLab

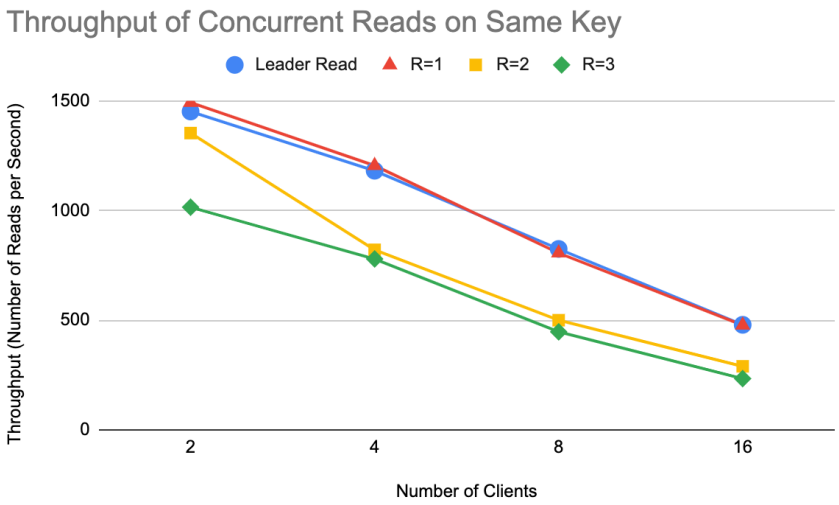
Ubuntu 20.04 LTS

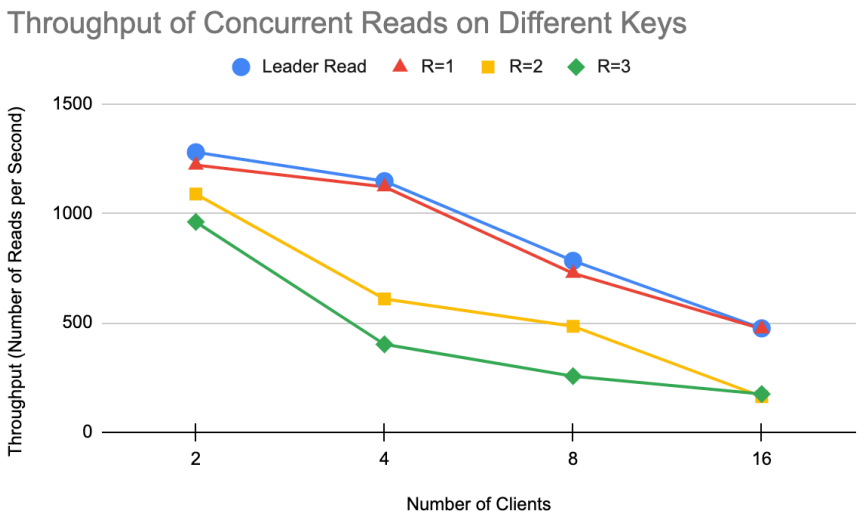
32 cores

Read Latency

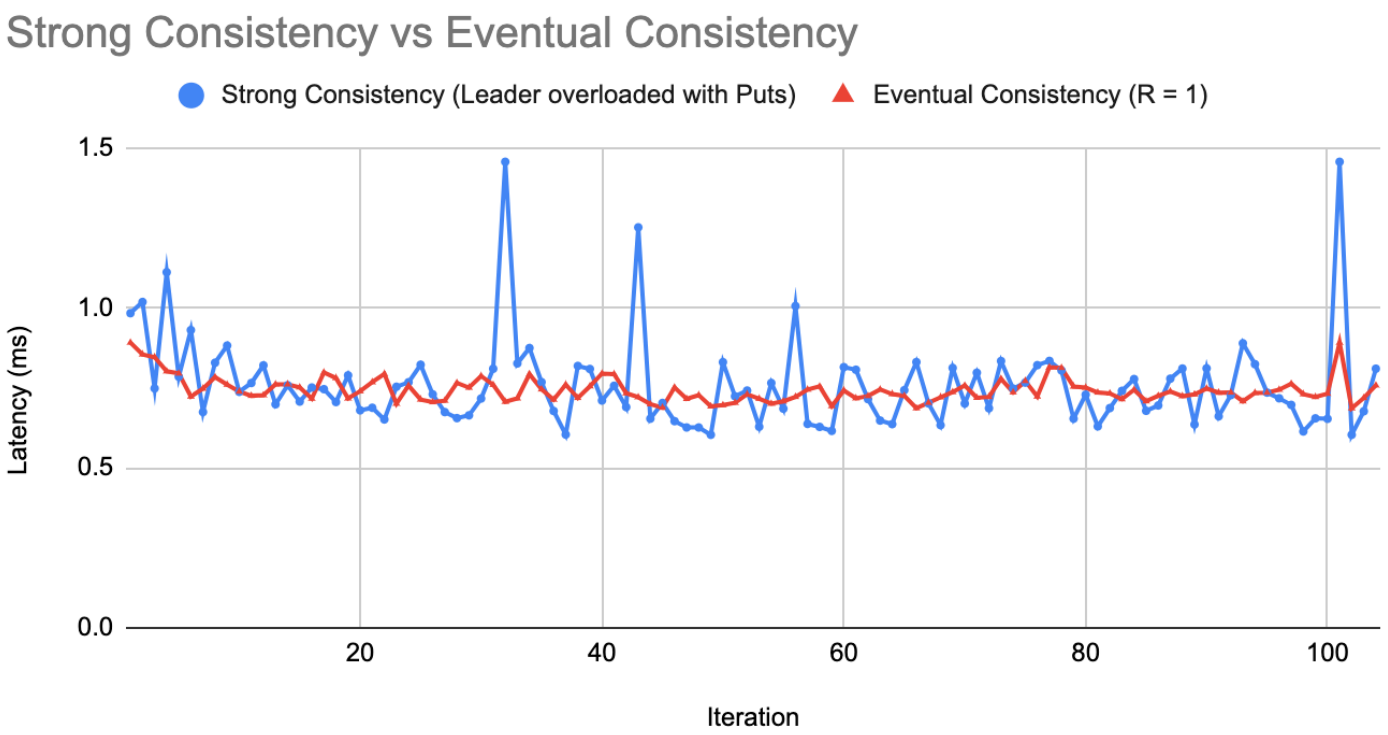


Total Servers = 3
sizeof(key) = 16 B
sizeof(value) = 100 B

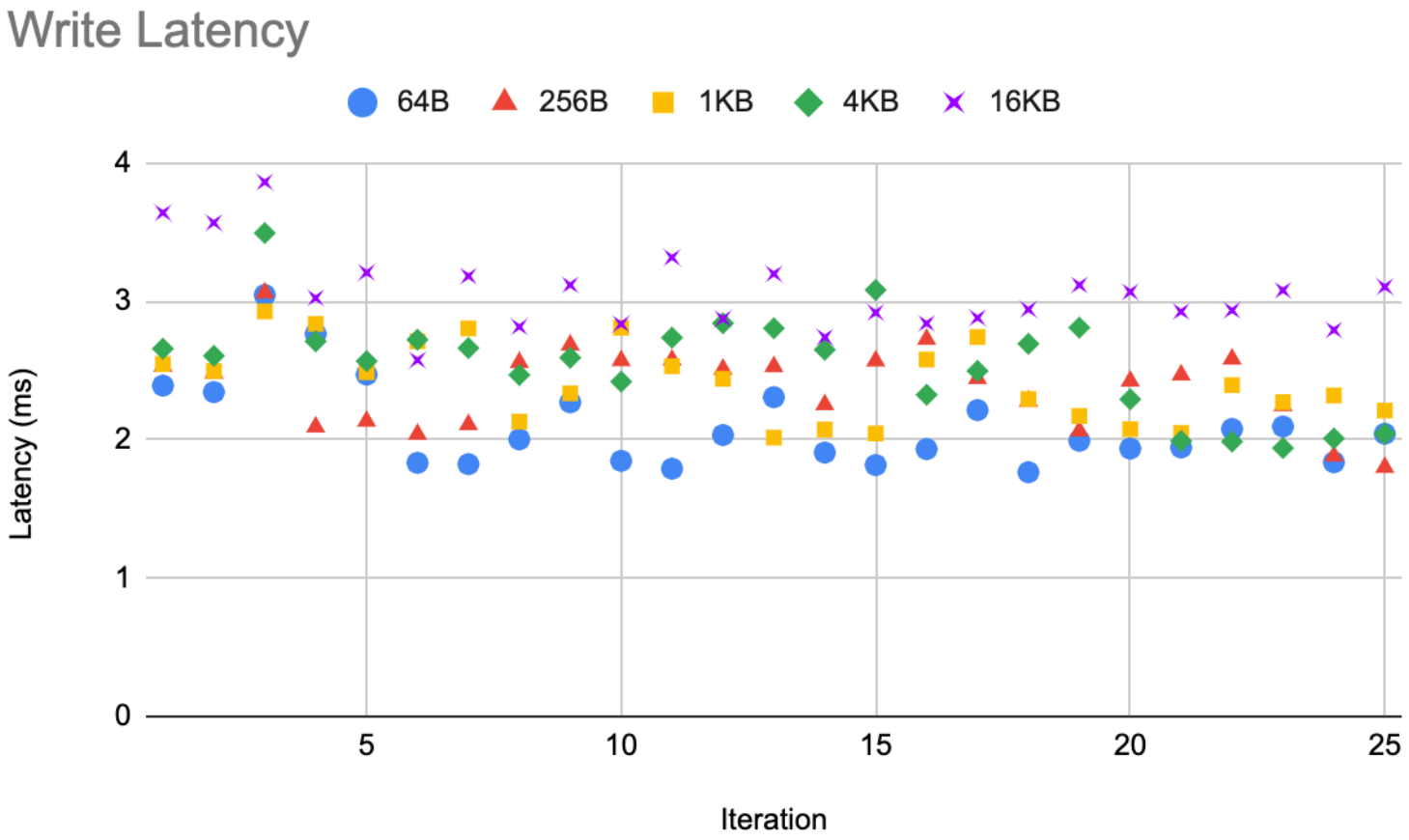




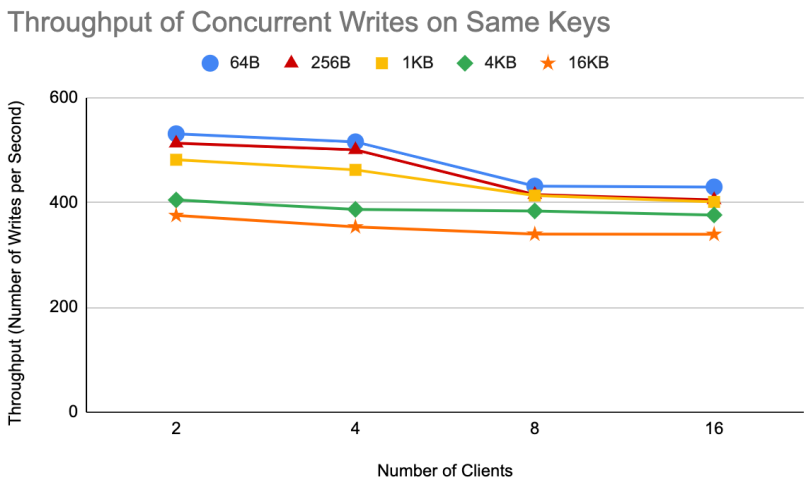
Total Servers = 3
sizeof(key) = 16 B
sizeof(value) = 100 B

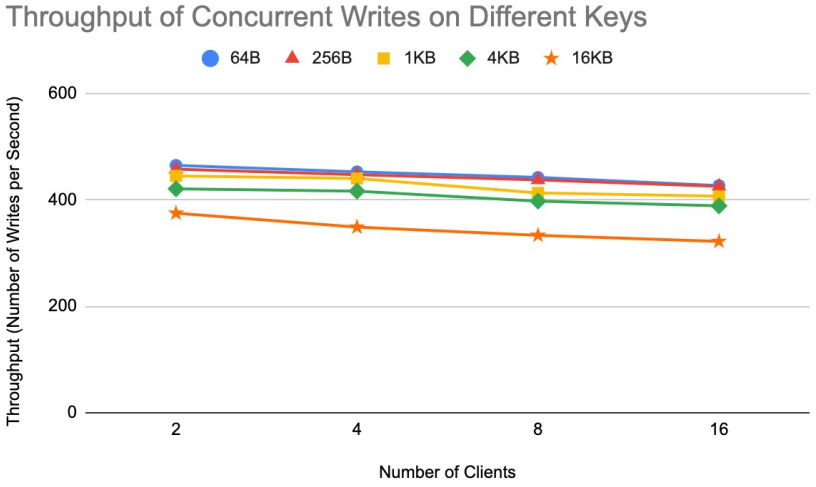


Total Servers = 3
sizeof(key) = 16 B
sizeof(value) = 100 B
Same Key

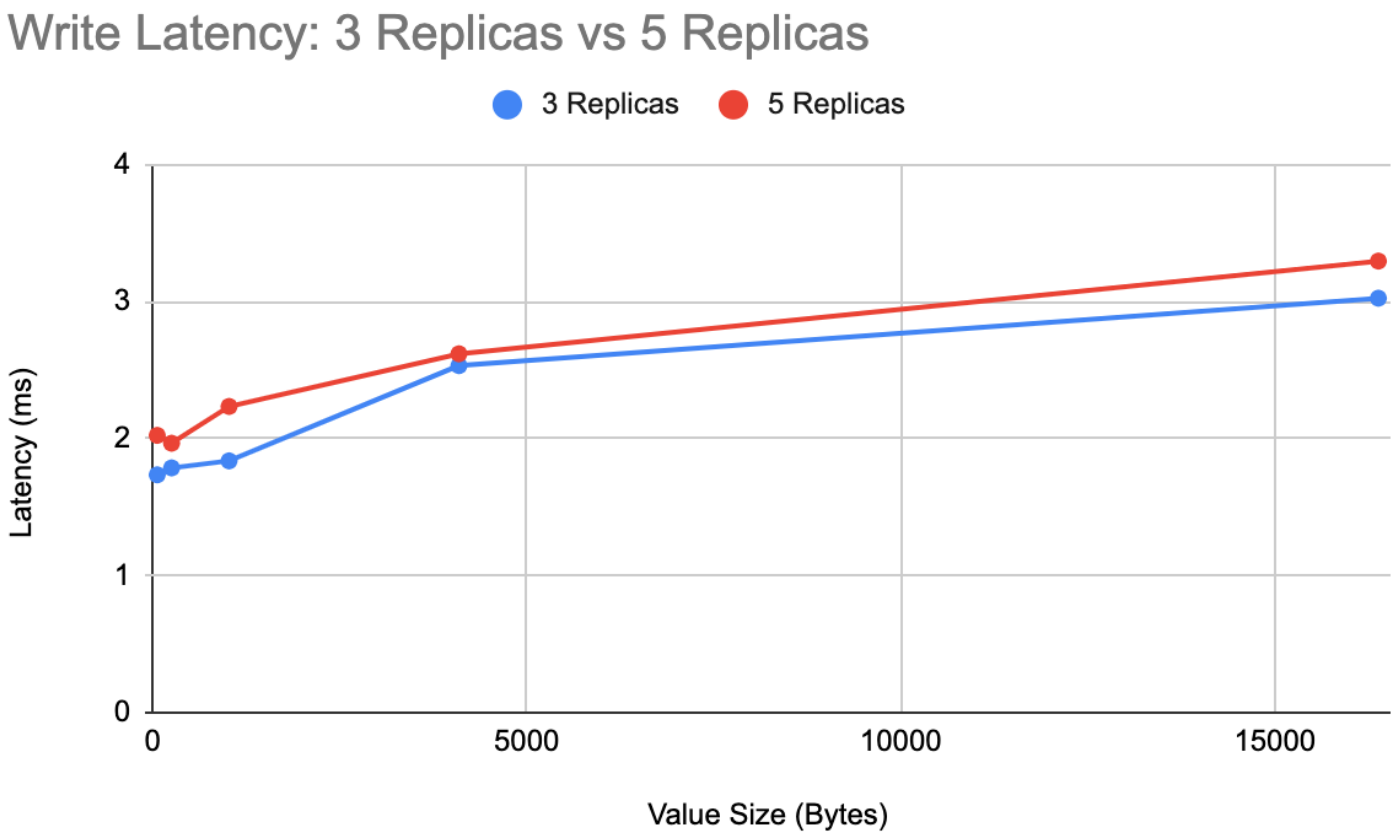


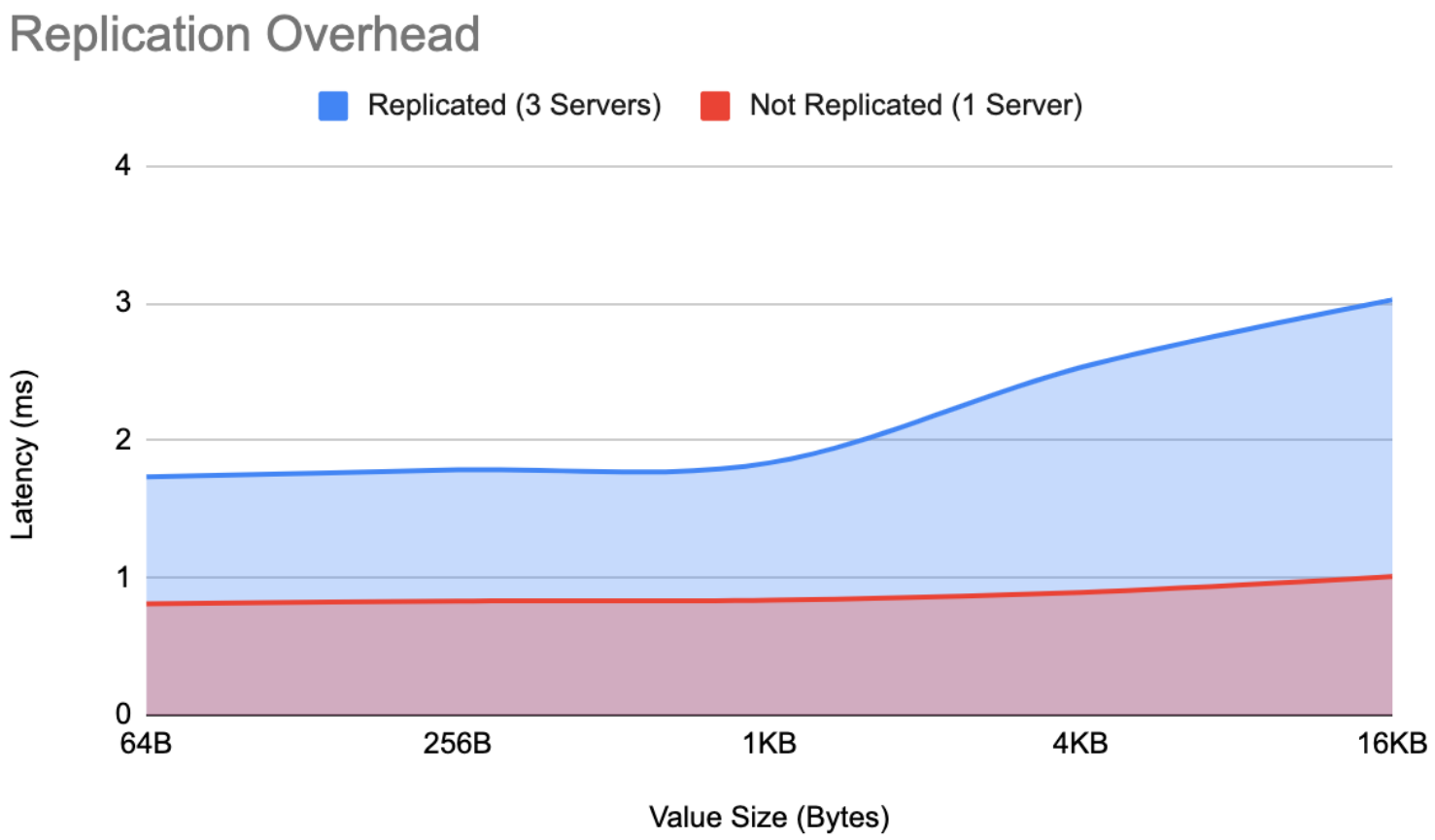
Total Servers = 3
sizeof(key) = 16 B





Total Servers = 3
sizeof(key) = 16 B


Write Latency: 3 Replicas vs 5 Replicas
3 Replicas
5 Replicas
Latency (ms)
4
3
2
1
0
0
5000
10000
15000
Value Size (Bytes)



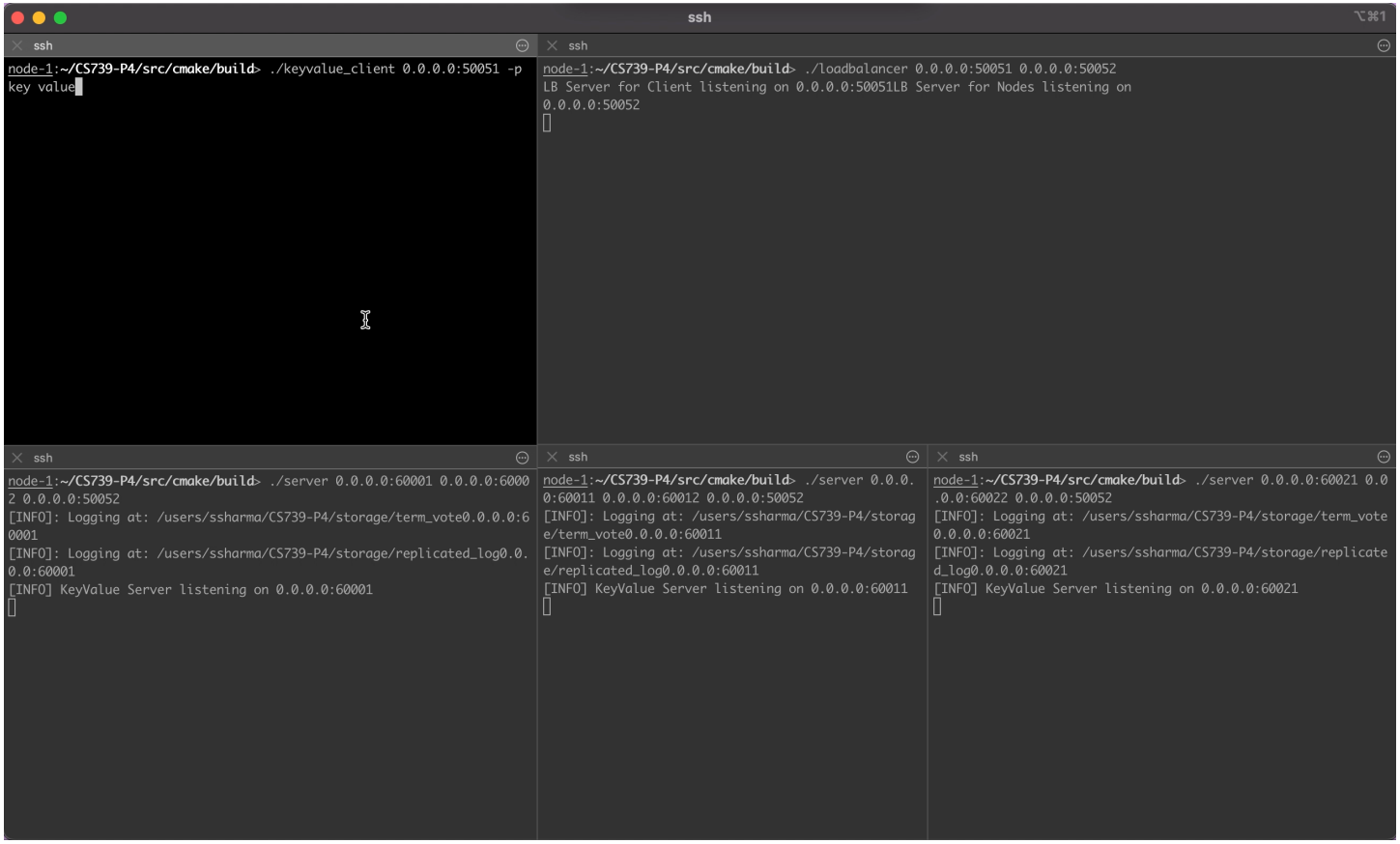
sizeof(key) = 16 B

Demos

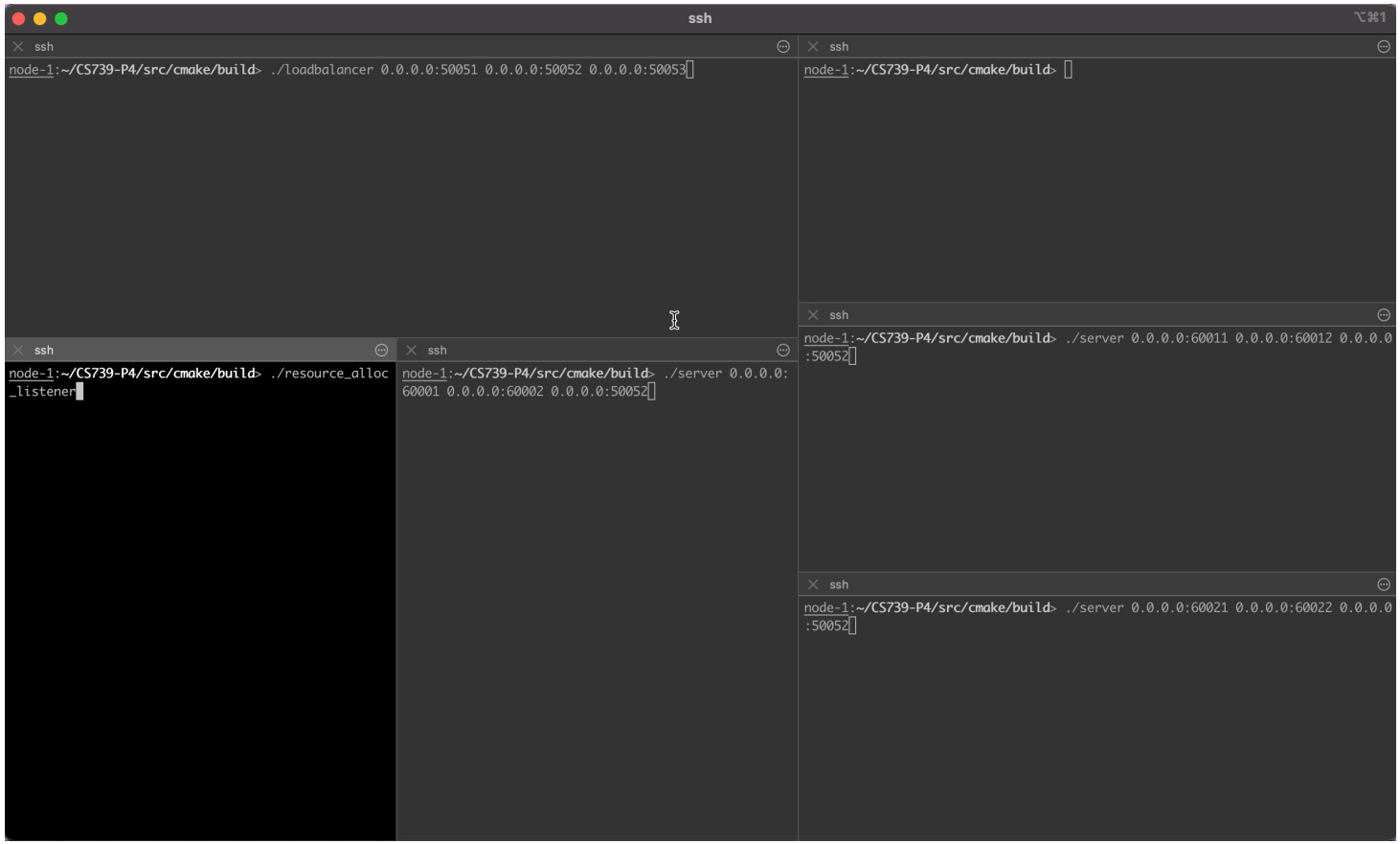
Demo List

- Raft In Action
- Autoscaling
- Correctness: Leader Election
- Leader Crash
- Tunable Consistency

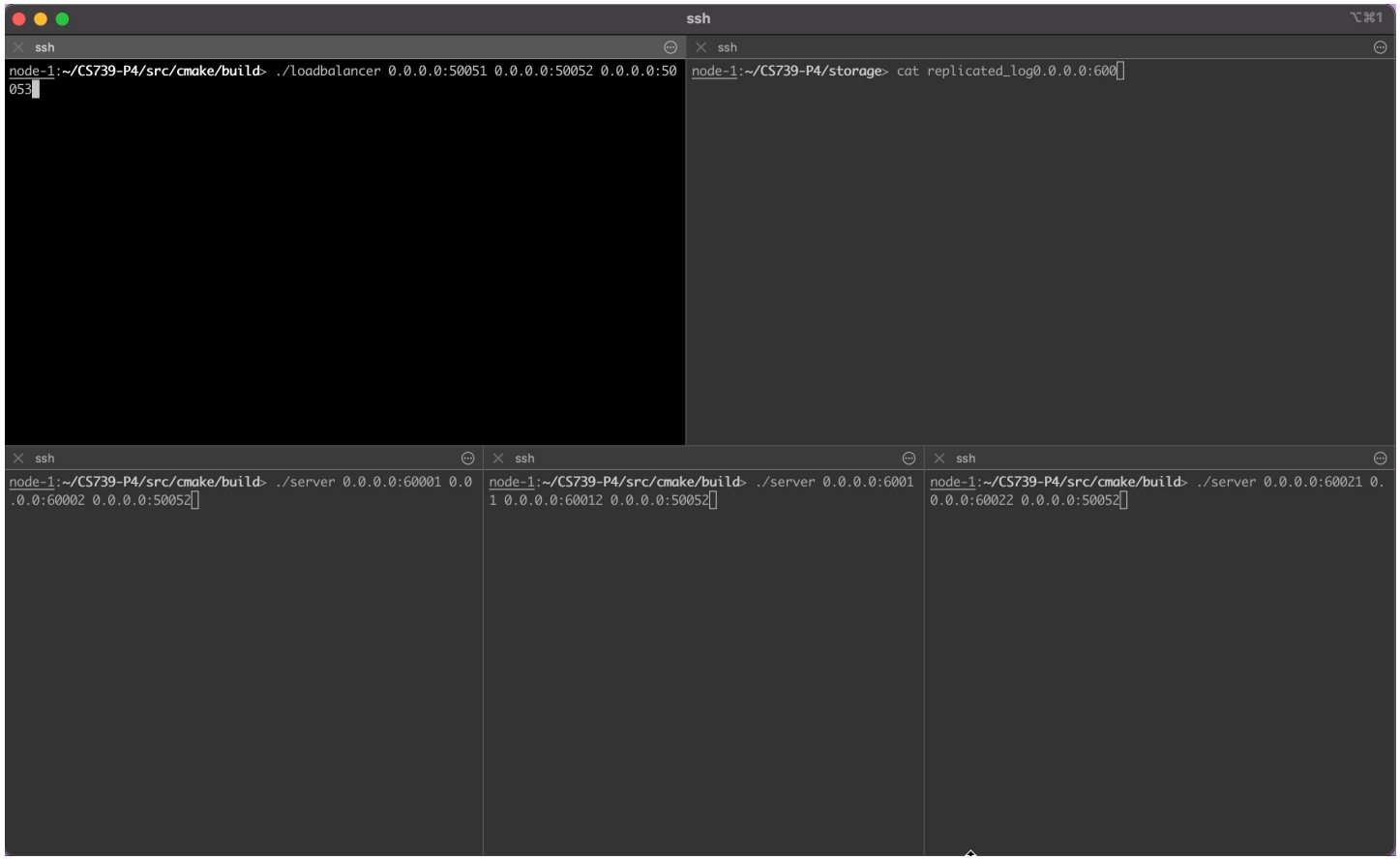
Raft in Action



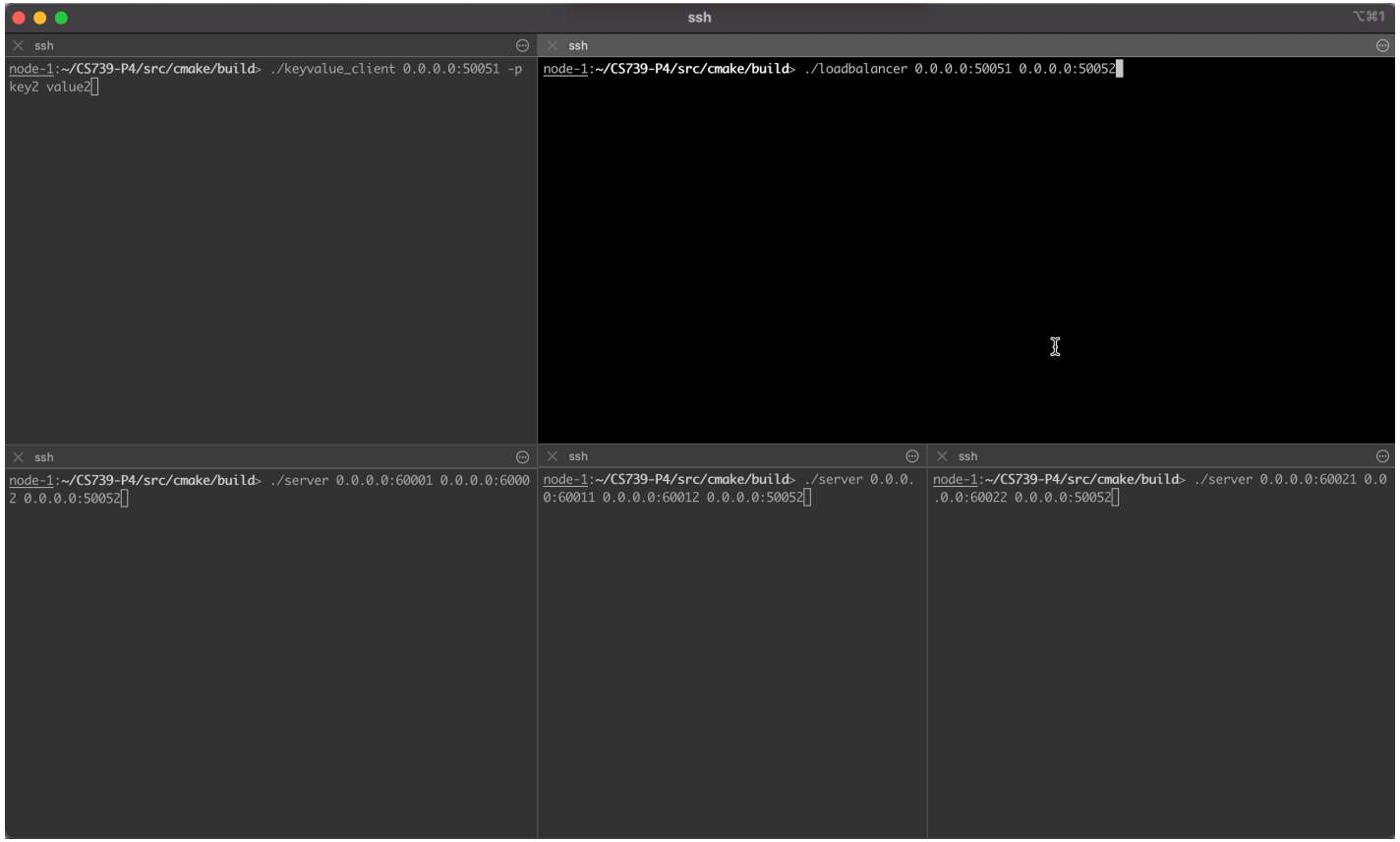
Autoscaling



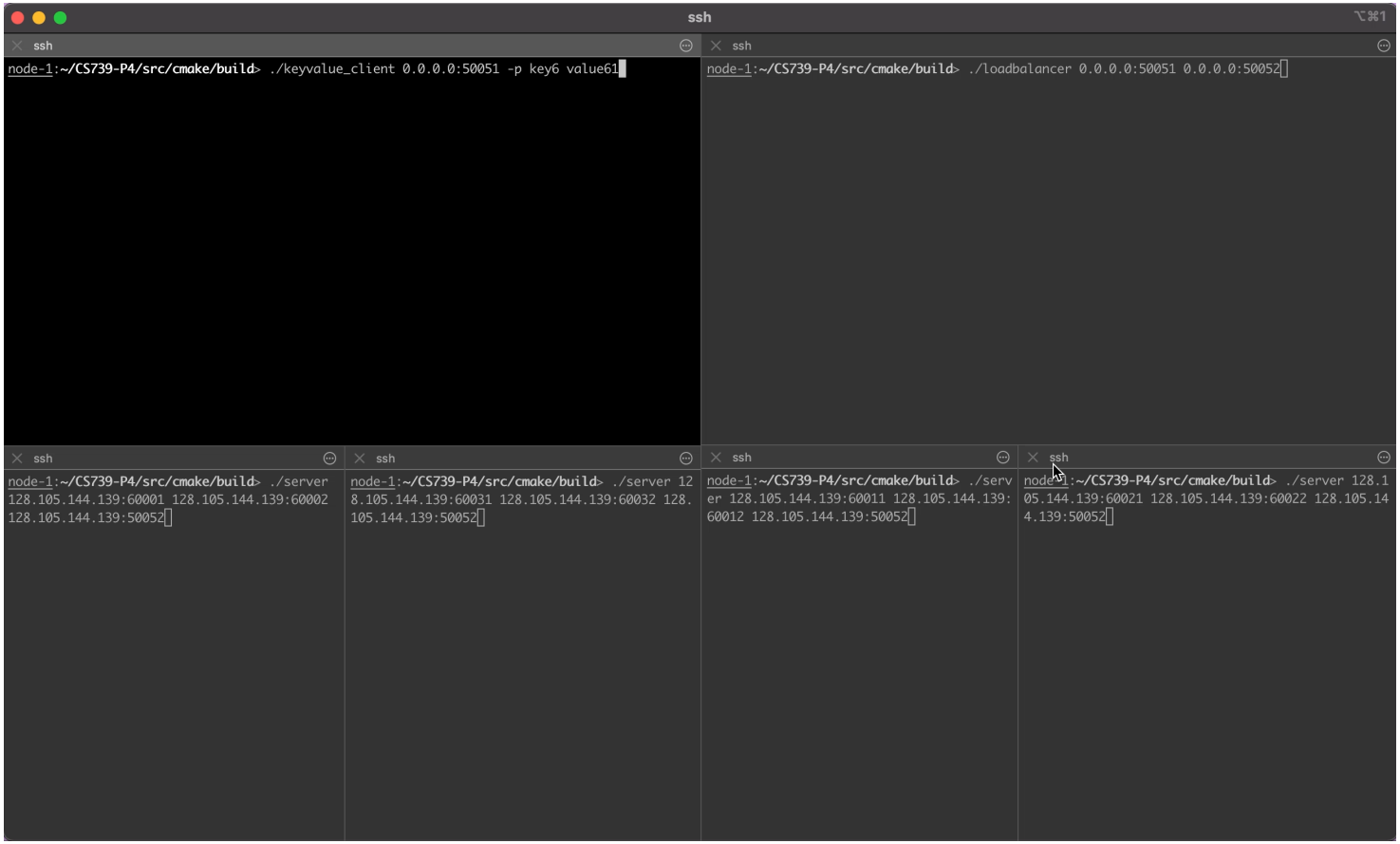
Correctness: Leader Election



Leader Crash



Tunable Consistency



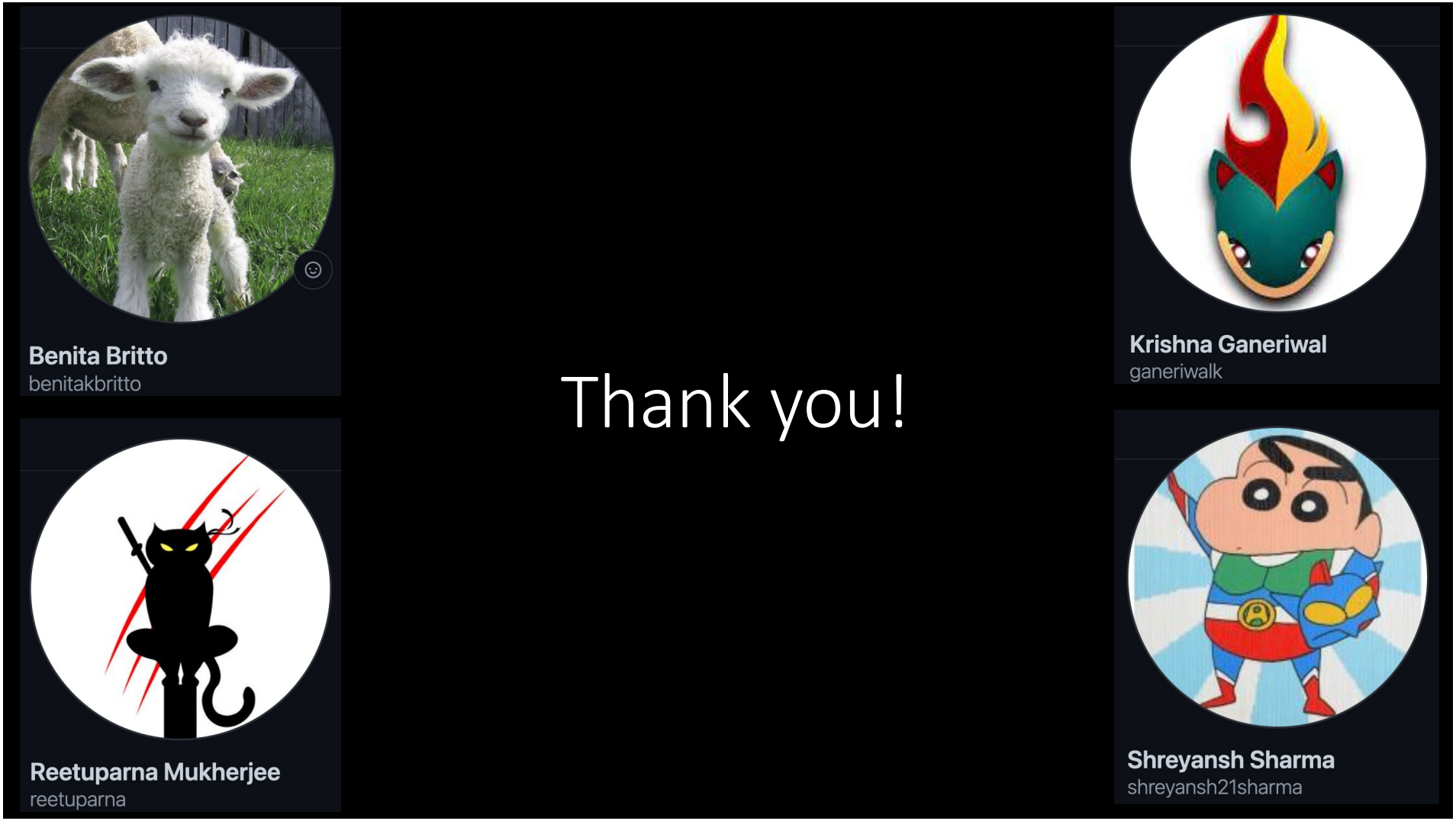
Work In Progress

SNAPSHOT:

- For Log Compaction, save KV State with the Last Index and Last Term of System when Snapshot is taken.
- Snapshot is taken periodically at every node.
- Leaders shares its updated snapshot with newly joining nodes in cluster for quick updates.

Conclusion

- LevelUpDB, our cloud native version of LevelDB, uses RAFT as the consensus protocol to maintain consistency between replicas.
- Replication comes with a tradeoff on performance
- To provide flexibility for different use-cases, LevelUpDB offers tunable consistency levels, wherein clients can specify how they want their read requests to be processed.
- It offers a Resource Allocator to manage the number of replicas running and offers capability to dynamically add and remove servers while also managing crashes.


Benita Britto
benitakbritto
Reetuparna Mukherjee
reetuparna
Thank you!
Krishna Ganeriwal
ganeriwalk
Shreyansh Sharma
shreyansh21sharma