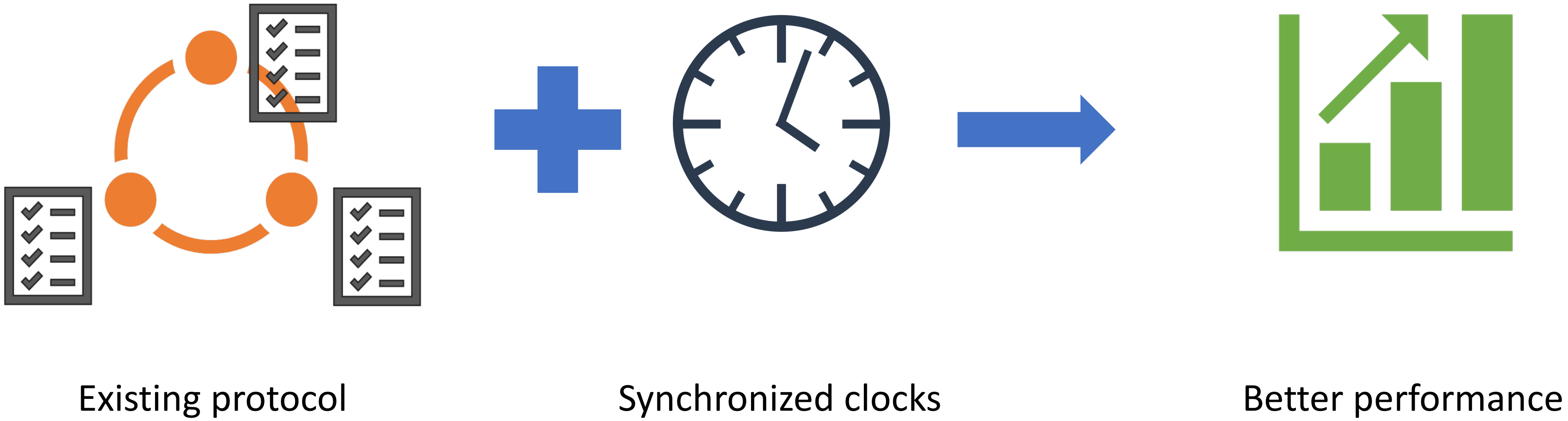
# CRaft: Building High-Performance Consensus Protocols with Accurate Clocks

Feiran Wang*, Balaji Prabhakar*, Mendel Rosenblum*, Gene Zhang†
*Stanford University, †eBay Inc.

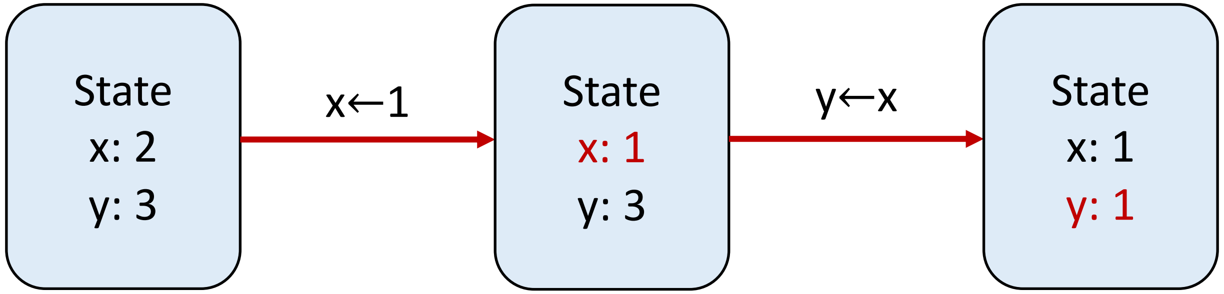
# Overview

- CRaft: a multi-leader extension to Raft enabled by accurate clocks



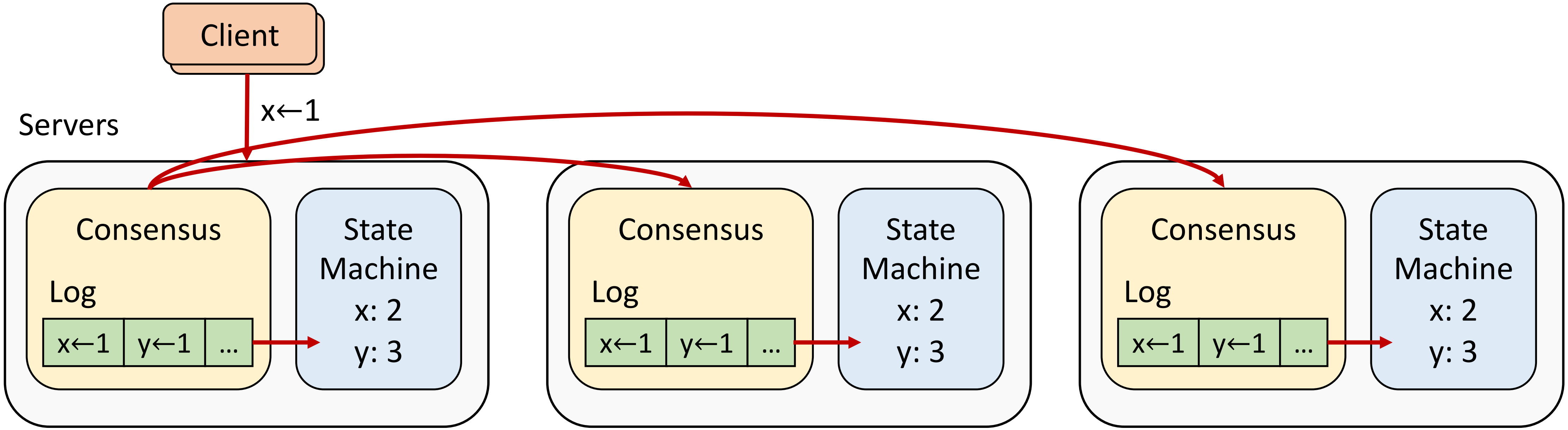
# State Machines

- Maintain internal states
- Respond to external requests
- Examples: databases, storage systems




- How do we make them reliable?

# Replicated State Machines

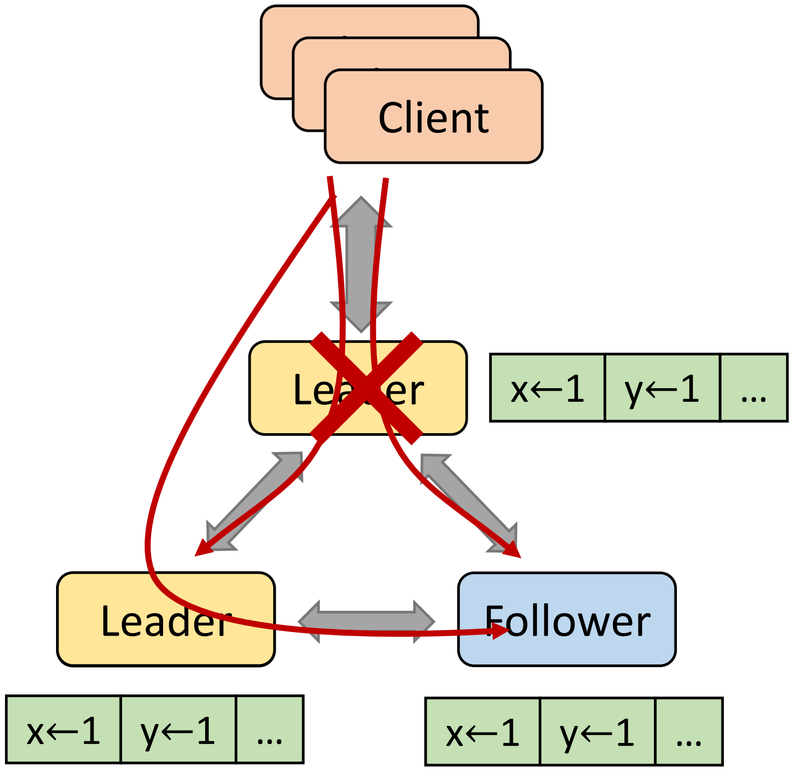



- Consensus: ensures all servers agree on the same log
- Continues to operate if at least a majority of servers are up

Diego Ongaro and John Ousterhout. The Raft consensus algorithm. https://raft.github.io

# The Raft Consensus Protocol

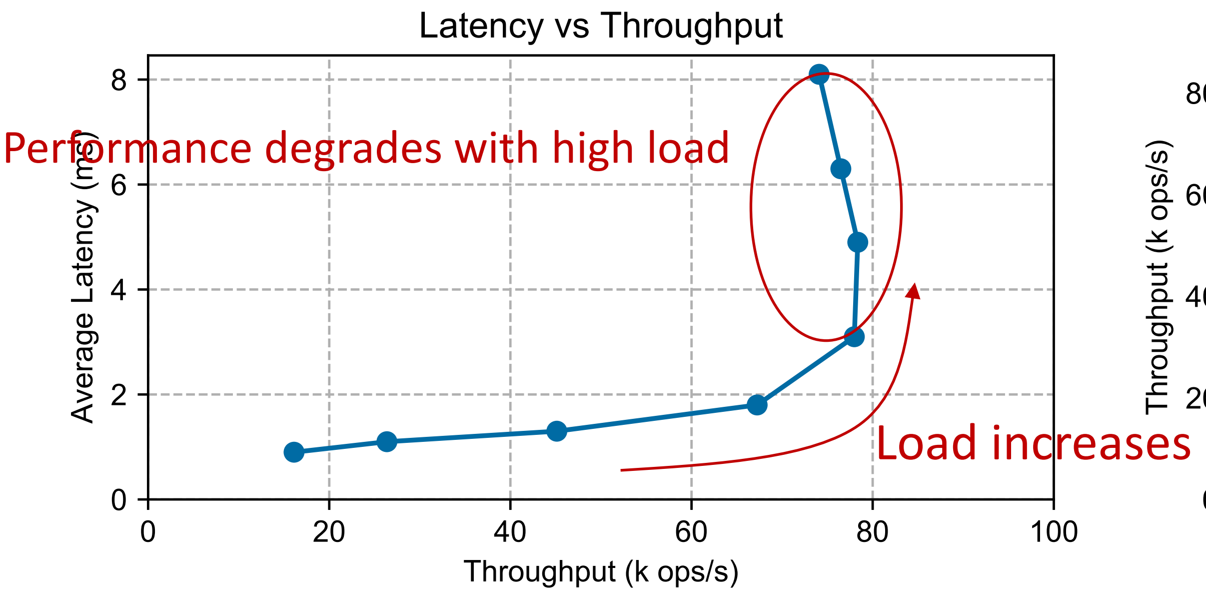
- A widely used consensus protocol
- Leader-based
- Benefits: simple and efficient
- Limitation: leader is the bottleneck for throughput and scalability

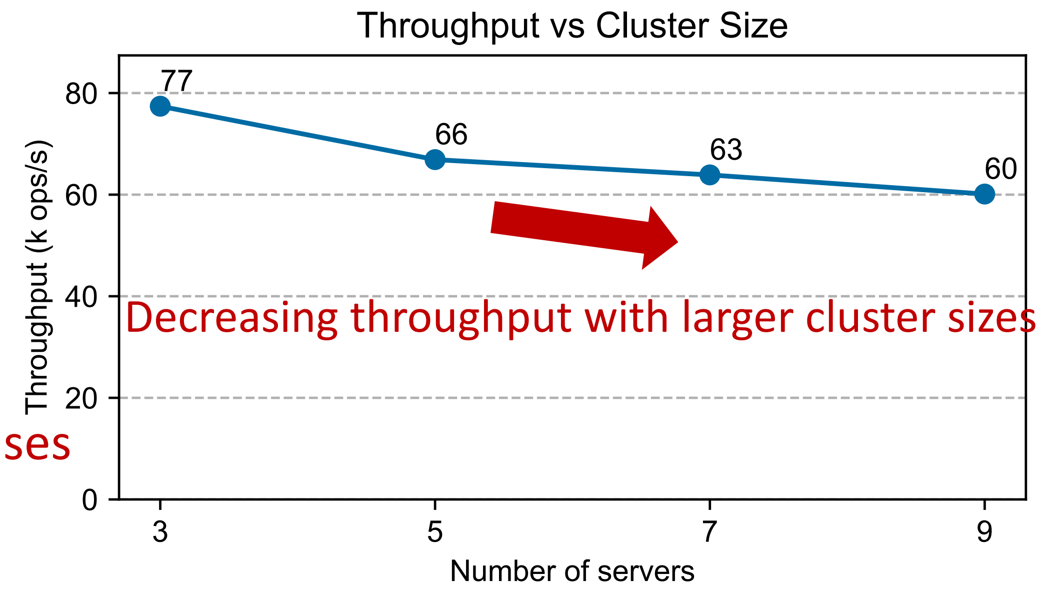



Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In USENIX Annual Technical Conference. 305–319.

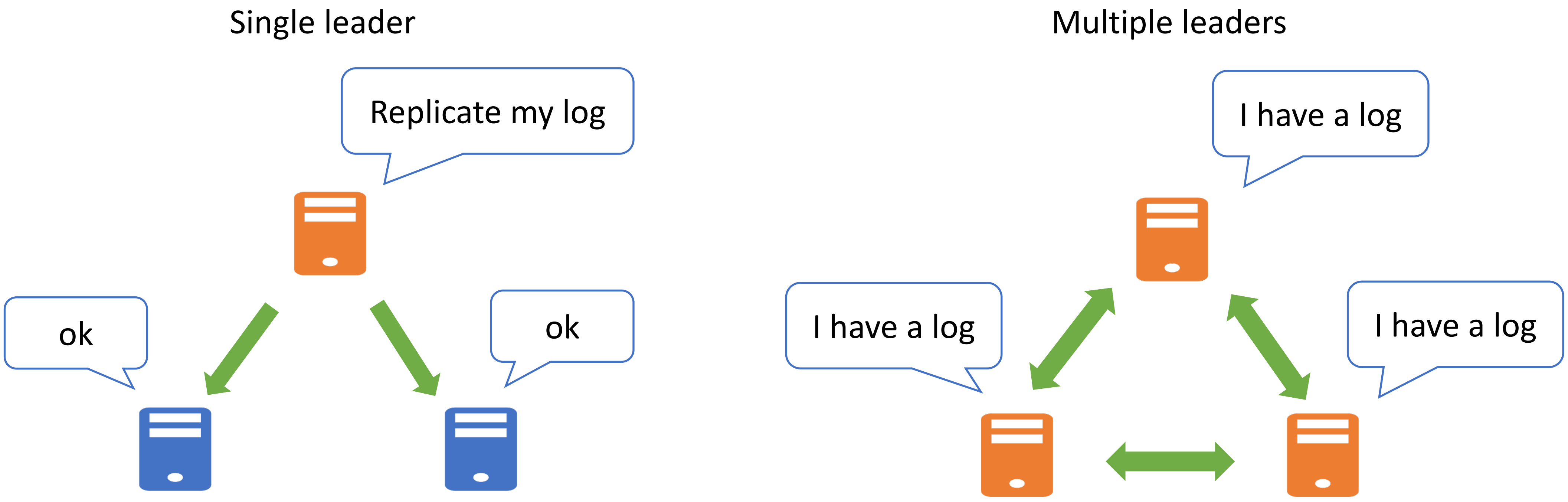
# Limitations with Single Leader

- Single leader limits throughput and scalability



# Challenge in a Multi-Leader Protocol



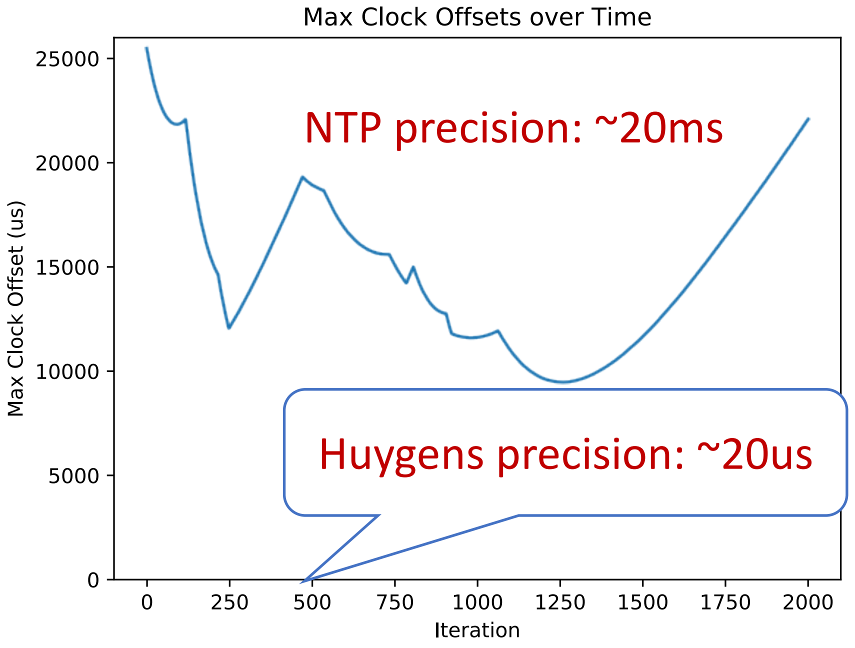

- Challenge: how to coordinate leaders?
- Solution: agreement on time => agreement on order

# Clock Synchronization

- Achieving agreement on time is not trivial in a distributed system
- Huygens: a software clock synchronization system

Distribution of clock offsets between servers
(20 machines on CloudLab)

| Percentile | 90th | 99th | 99.9th | max |
| --- | --- | --- | --- | --- |
| Clock offset | 7us | 11us | 15us | 26us |



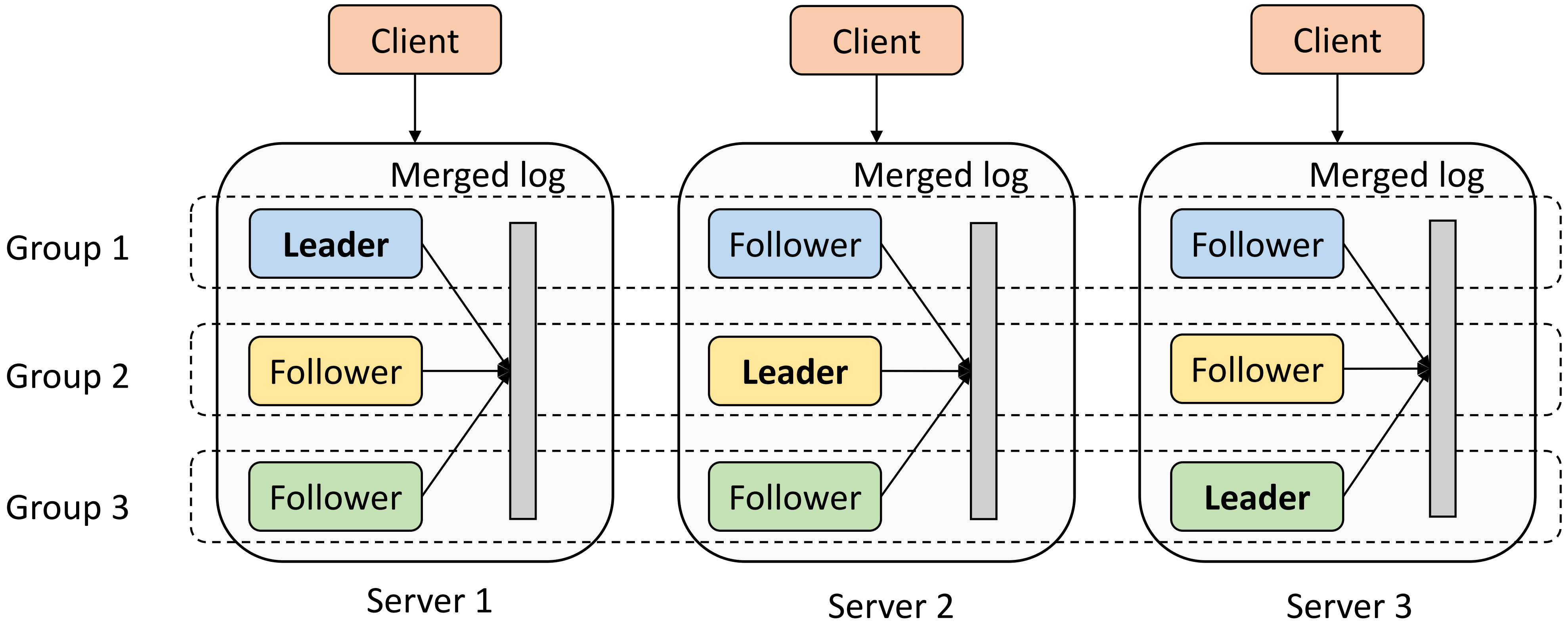

Yilong Geng, Shiyu Liu, Zi Yin, Ashish Naik, Balaji Prabhakar, Mendel Rosenblum, and Amin Vahdat. Exploiting a natural network effect for scalable, fine-grained clock synchronization. In NSDI 2018. 81–94.

# Our Approach: CRaft

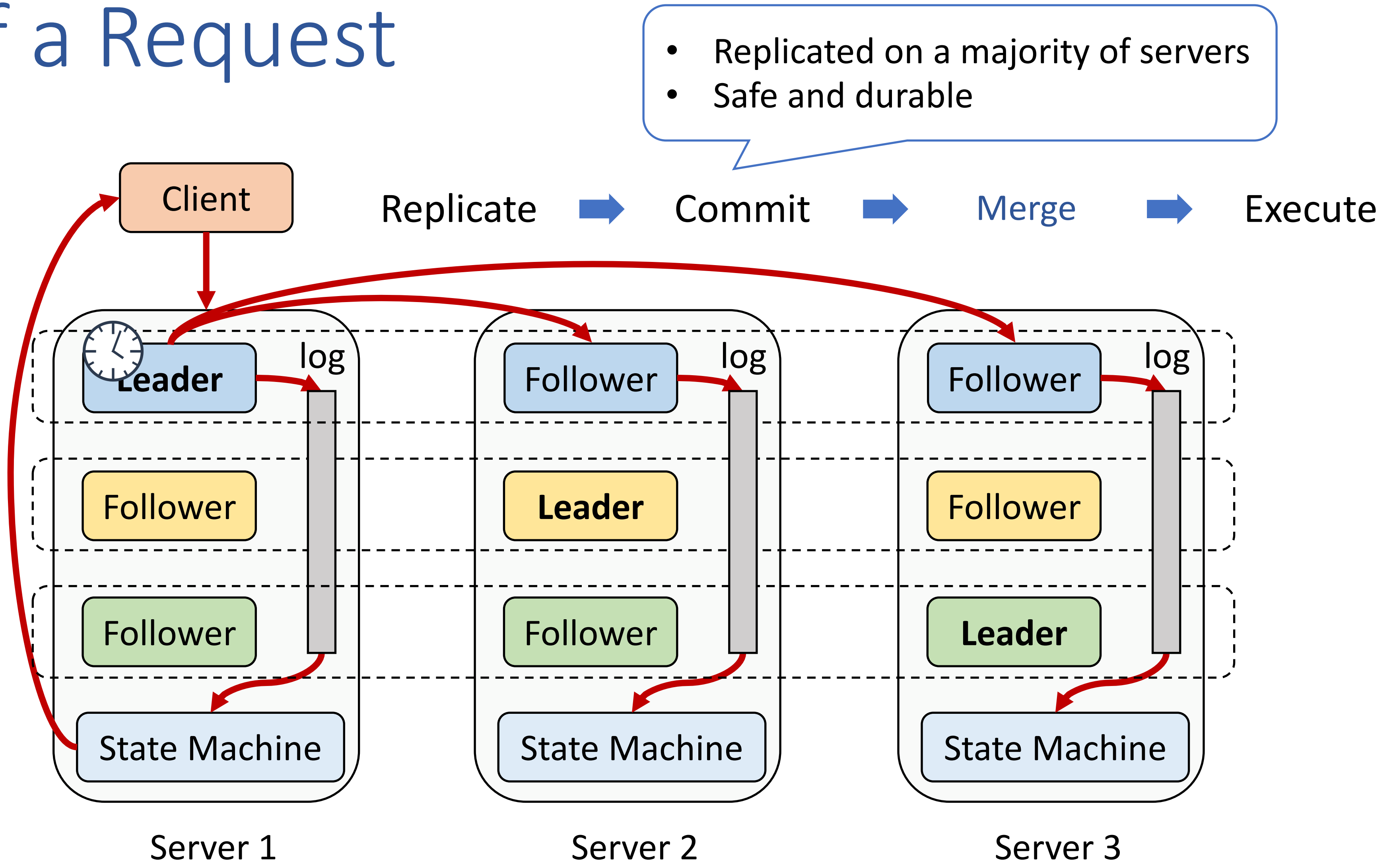
| | Raft | CRaft (Clocks + Raft) |
|---|---|---|
| Scalability |  |  |

# The CRaft Consensus Protocol

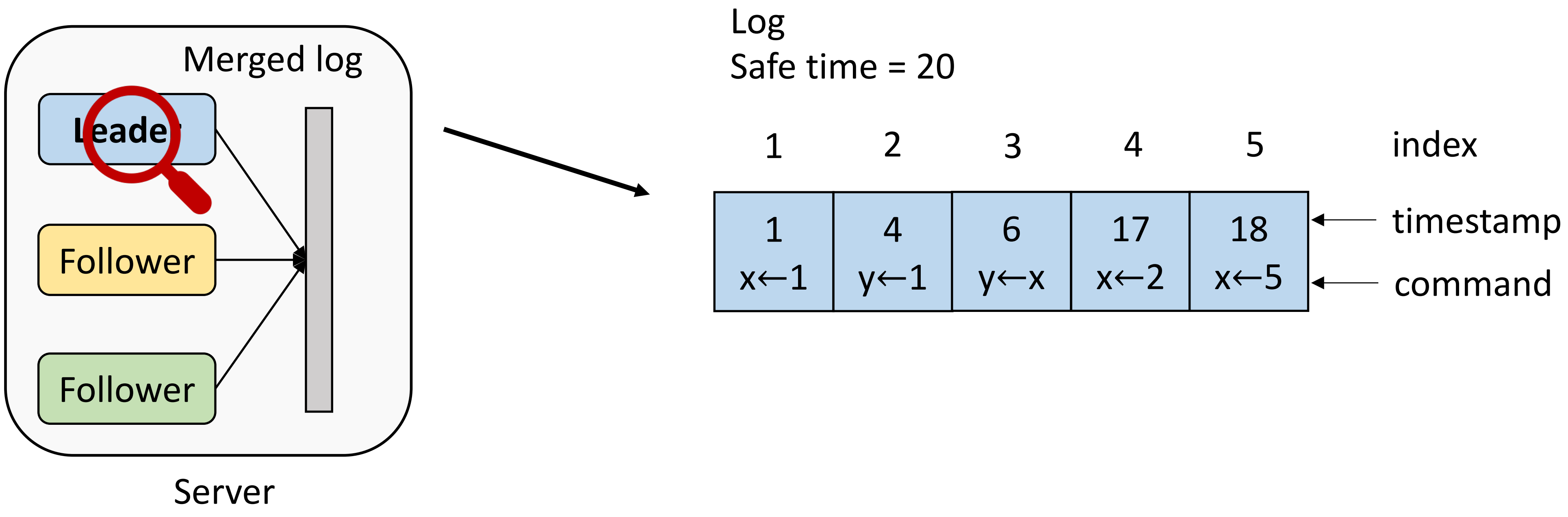
# CRaft Overview



# Life of a Request



# Timestamp Management



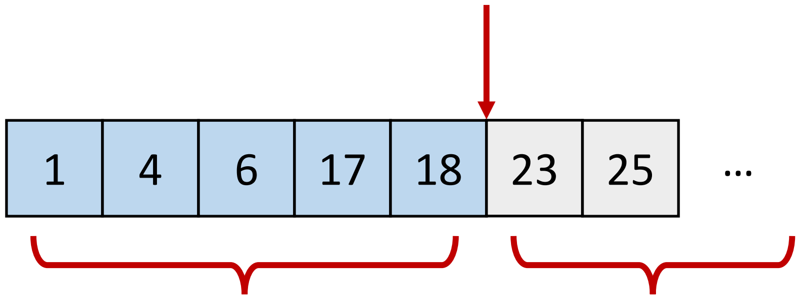

- CRaft guarantees monotonically increasing timestamps in each log
- Safe time: indicates how up-to-date a log is

# Safe Times

How up-to-date is this log?

Now

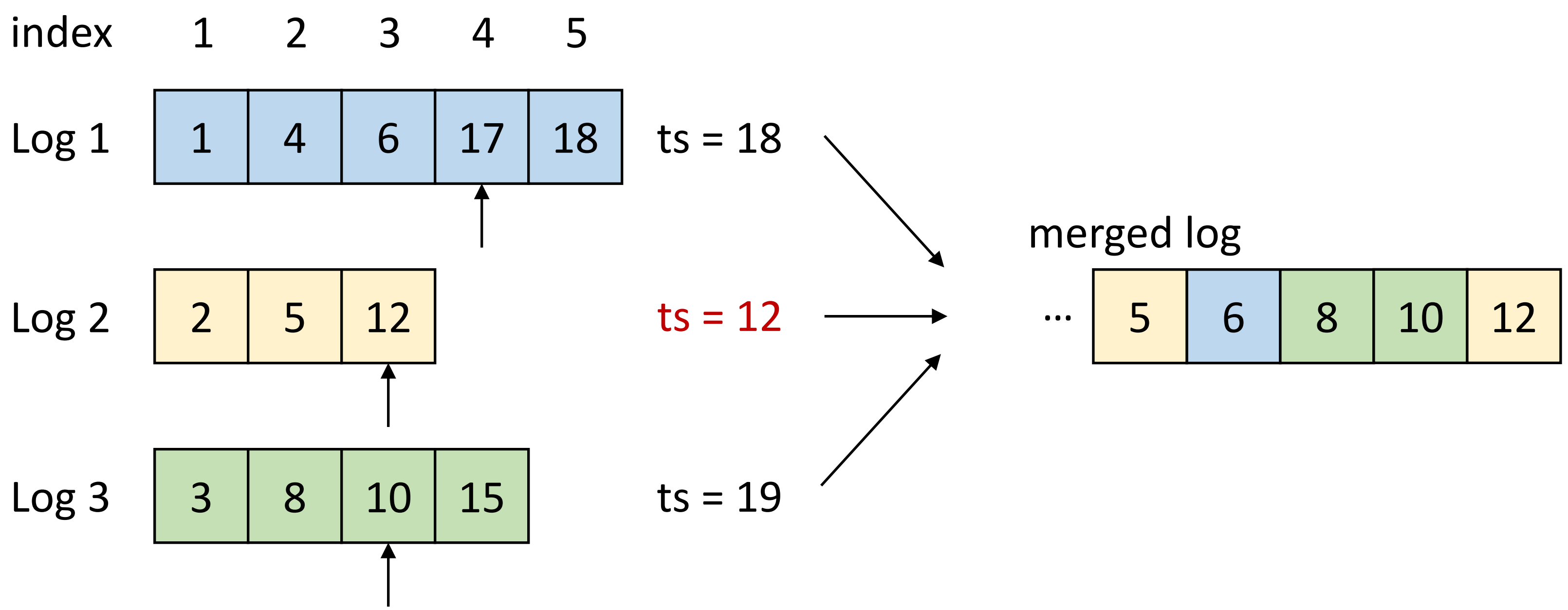
Log
Safe time = 20




Current entries: timestamps <= safe time

No entries come in with a timestamp smaller than safe time

# Merging




- Merge up to the smallest safe time
- CRaft ensures merged log in monotonically increasing timestamp order

# Optimization: Fast Path

Replicate → Commit → Merge → Execute

Fast path: respond before execution (from Commit)

Normal path: respond after execution (from Execute)

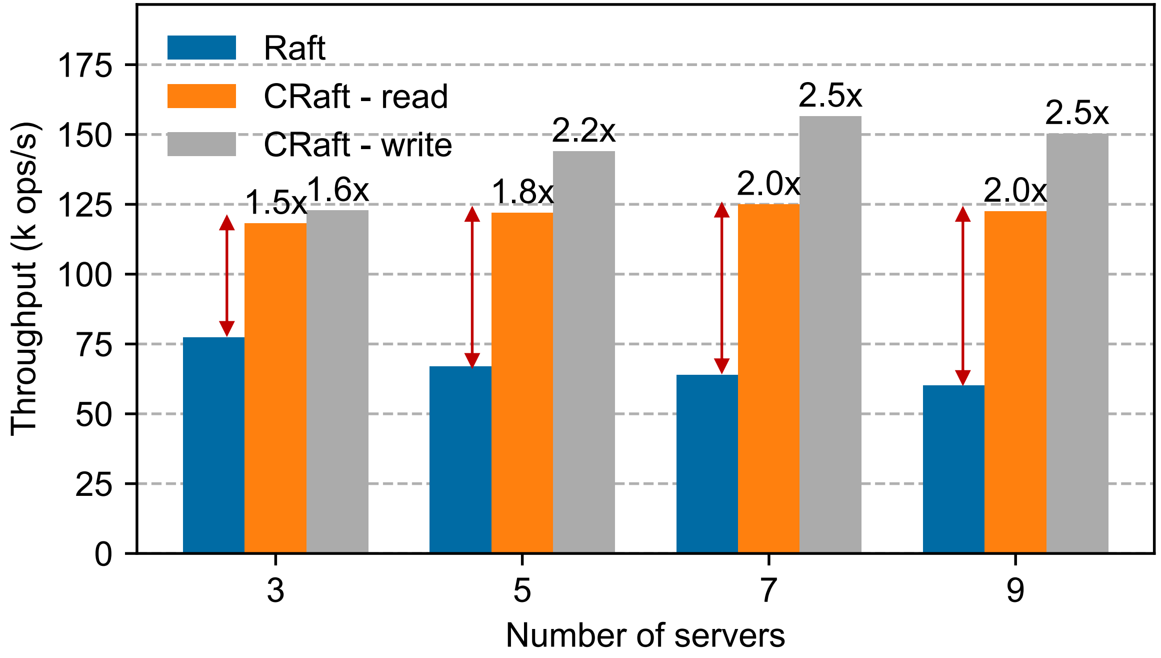
- Fast path: respond to clients early for certain write operations

# Evaluation

# Experiment Setup

- Implementation
  - Based on HashiCorp Raft – a popular and well-optimized implementation
- Environment
  - CloudLab, single data center
- Workload
  - In-memory key-value store
  - Multiple clients send get or set requests concurrently

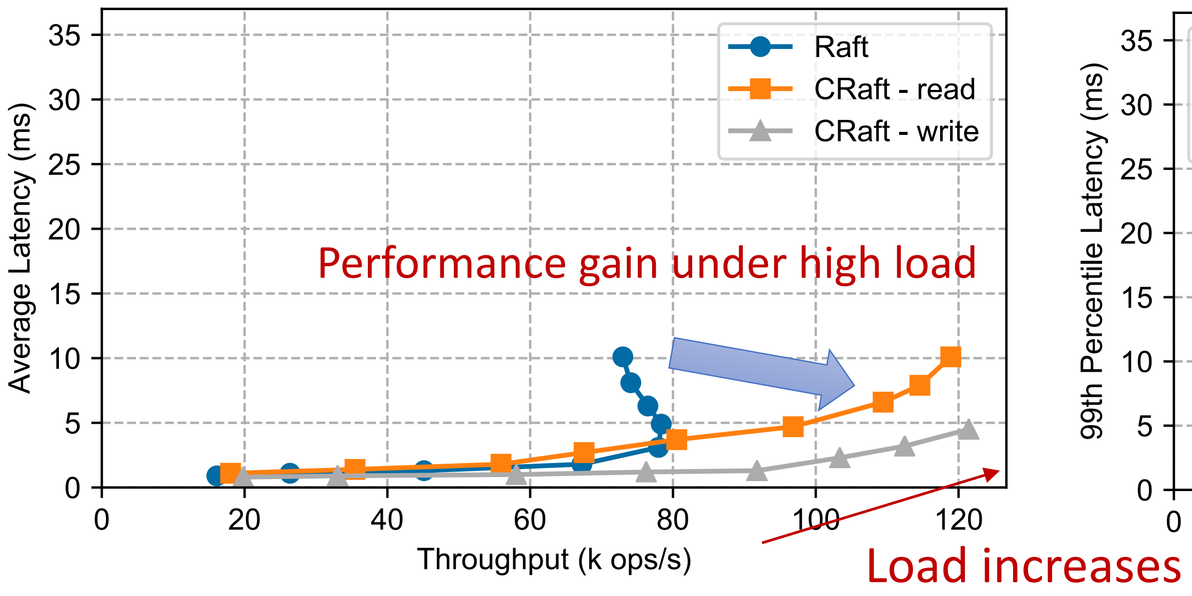
# Throughput vs Cluster Size



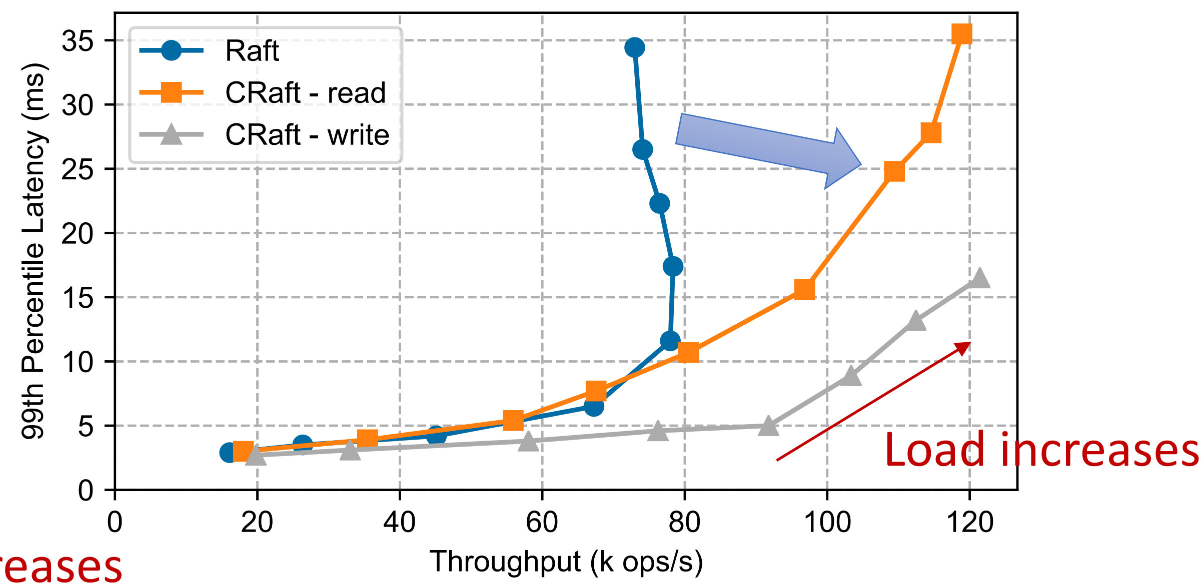

- Up to ~2x  read and ~2.5x write throughput compared to Raft

# Latency vs Throughput

Average latency vs throughput (3 servers)

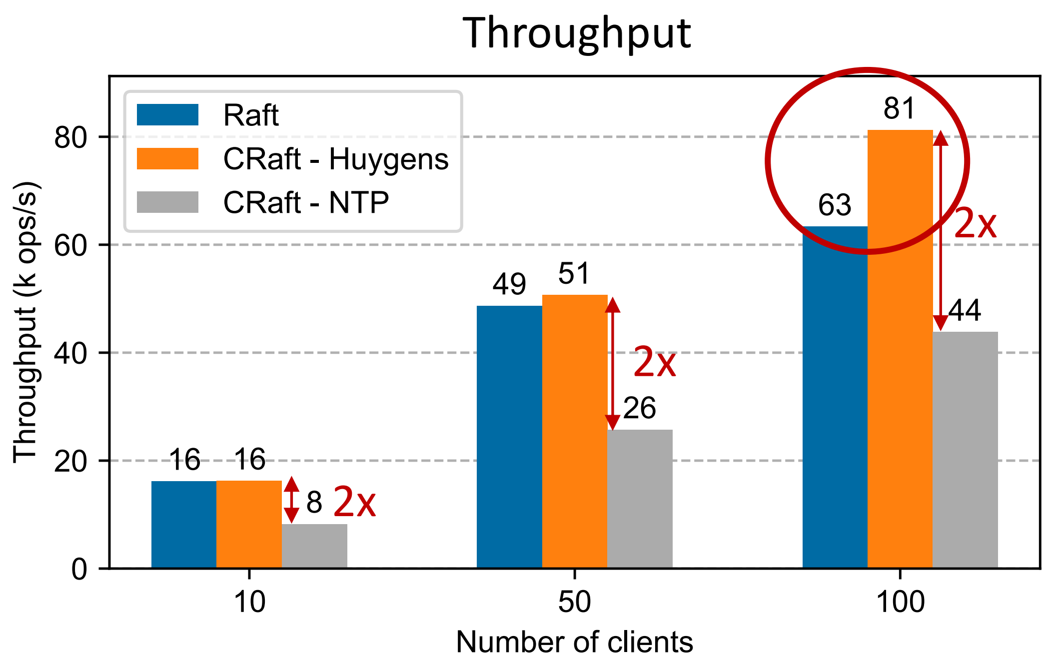


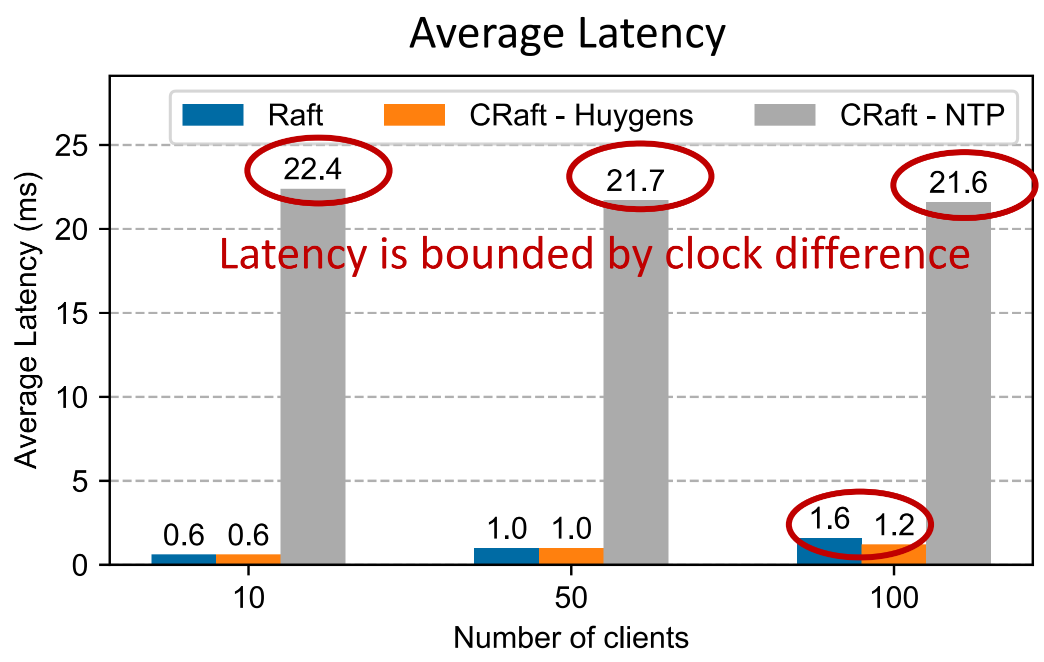

99th percentile latency vs throughput (3 servers)




- CRaft improves throughput and latency under high load

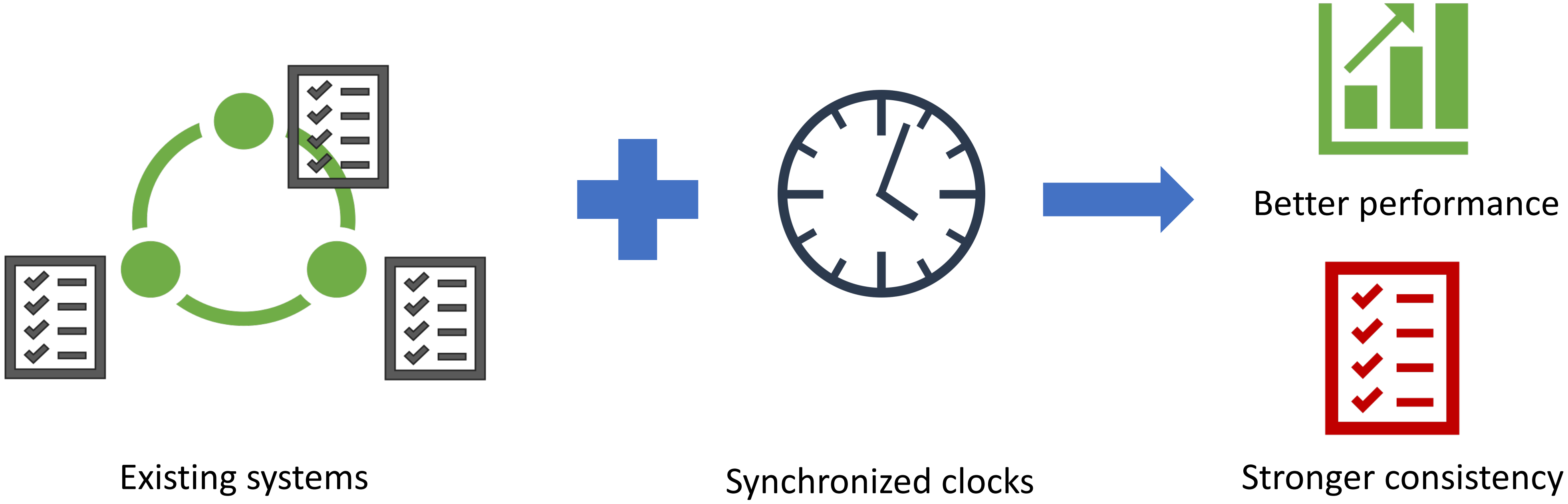
# Performance vs Number of Clients




- NTP precision: ~20ms, Huygens: ~20us

# Conclusion




- Accurate clocks enable better performance and/or consistency

Thank you!