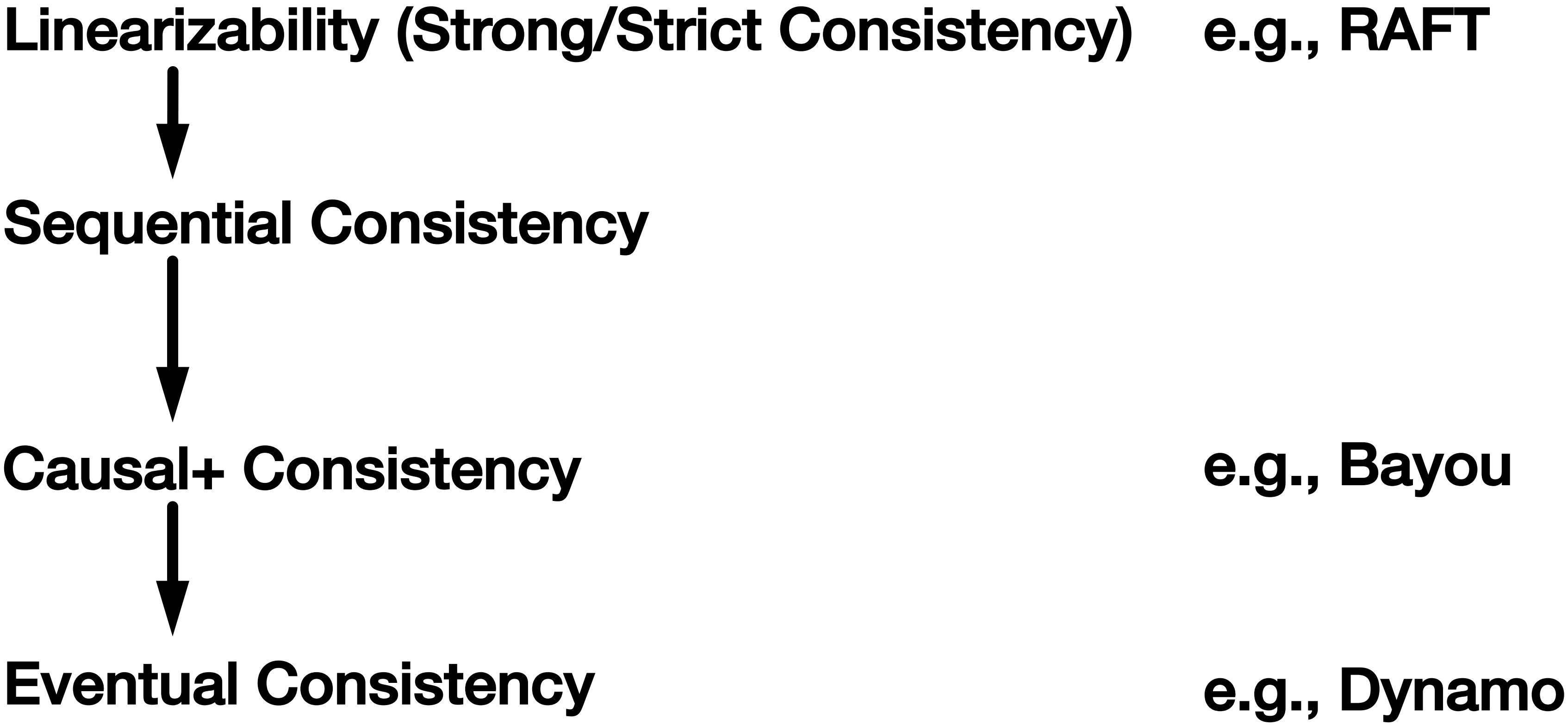
Scalable Causal Consistency

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

CS 240: Computing Systems and Concurrency
Lecture 16

Marco Canini

Consistency hierarchy



Linearizability (Strong/Strict Consistency) e.g., RAFT

Sequential Consistency

Causal+ Consistency e.g., Bayou

Eventual Consistency e.g., Dynamo

Causal+ Consistency

- Partially orders all operations, does not totally order them
  - Does not look like a single machine

- Guarantees
  - For each process, ∃ an order of all writes + that process's reads
  - Order respects the happens-before (→) ordering of operations
  - + replicas converge to the same state (conflict handling)
    - Skip details, makes it stronger than eventual consistency

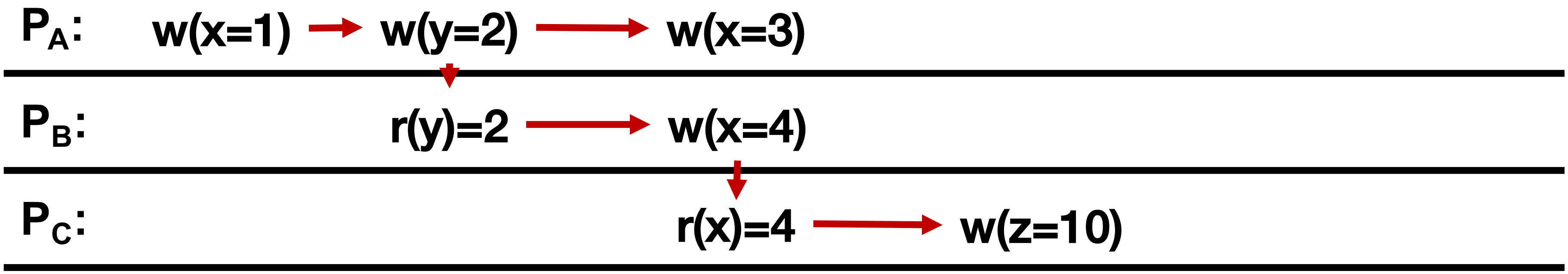
Causal Consistency

- Similar: respect partial order but there is no convergent conflict handling requirement

- Concurrent operations are unordered by causal consistency

- Thus, conflicts allow replicas to diverge forever

Causal Consistency: Relationships



P_A: w(x=1) w(y=2) w(x=3)

P_B: r(y)=2 w(x=4)

P_C: r(x)=4 w(z=10)

- Can P_C see x=4 and then x=1? **Why?**

Causal+ Examples

- Alice shares photo with Bob
    1. Upload the photo
    2. Add photo to album
    3. Bob checks album

- Under causal consistency, if the album has a reference to the photo, Bob must see the photo

- Under eventual consistency, album may have a reference to a photo that has not been written yet (the corresponding write has not propagated)

Causal+ Examples

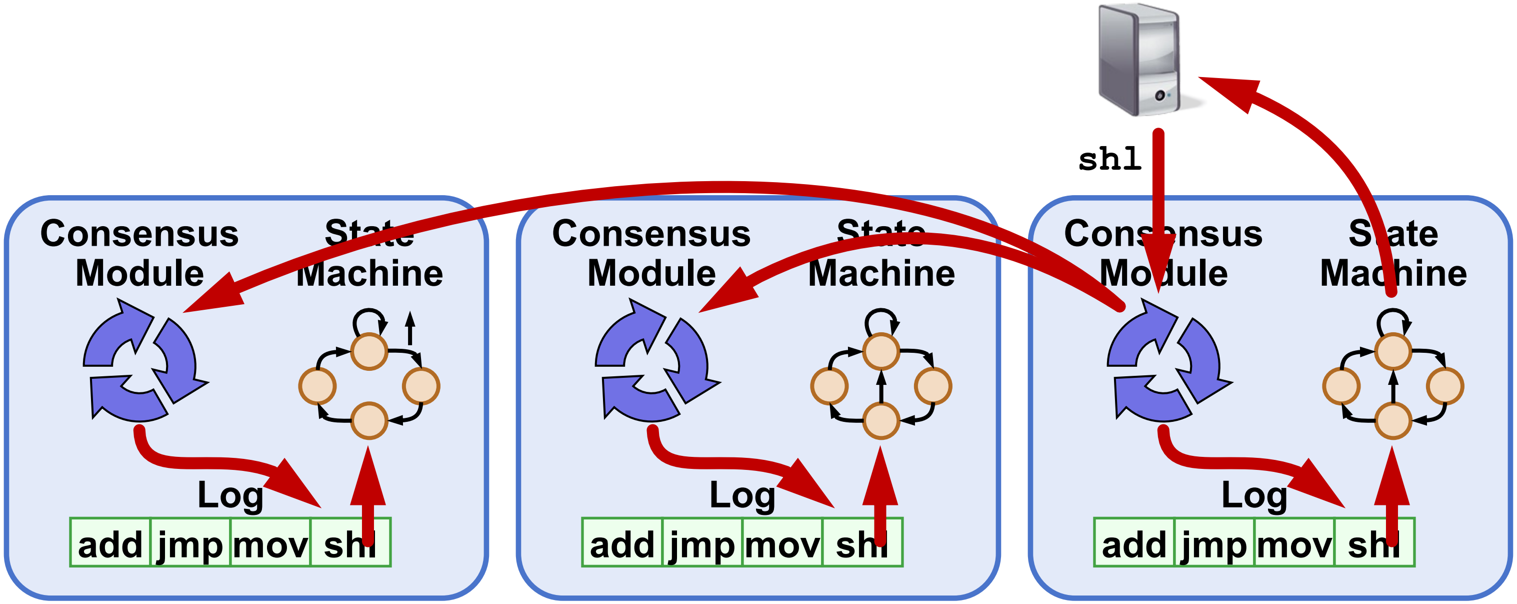
- Carol and Dan concurrently update event time (9pm)
  1. Carol sets 8pm
  2. Dan sets 10pm

- Under causal consistency, two replicas may forever return different times

- Under causal+ consistency, replicas must eventually handle the conflict in a convergent manner
  - If a last-writer-wins, either Carol's or Dan's write win

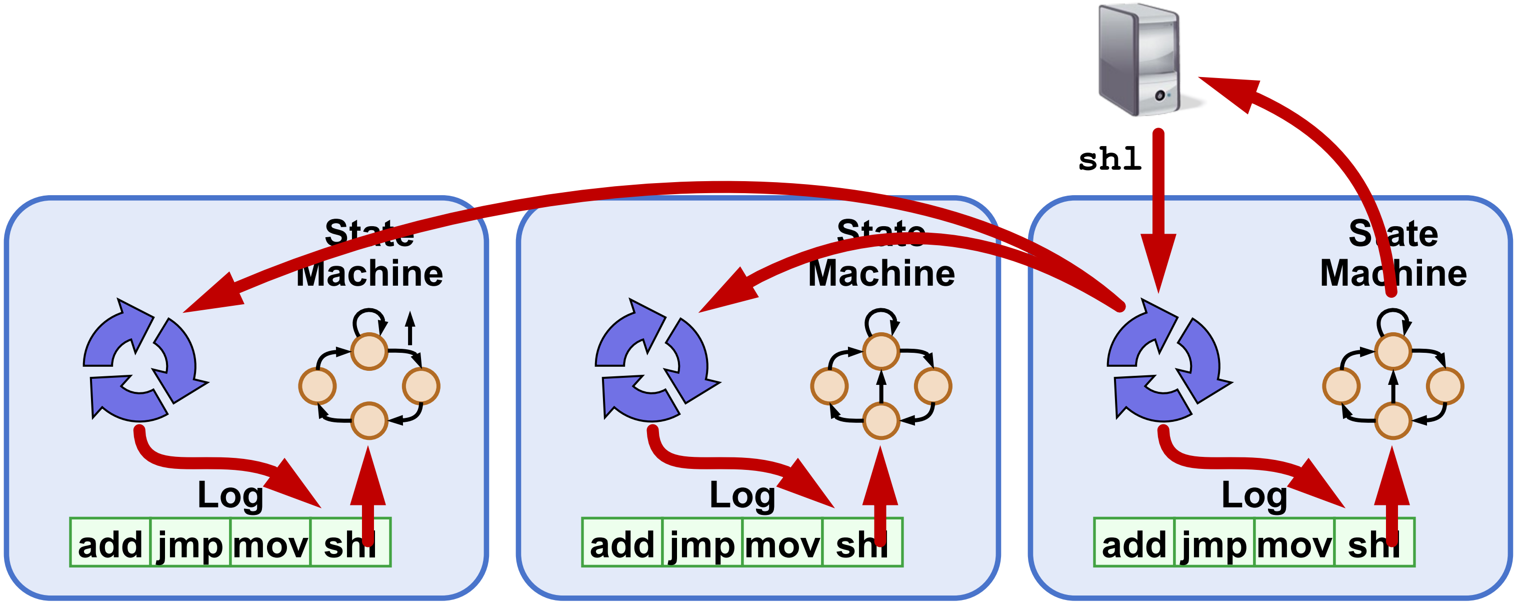
Causal consistency within replication systems

Implications of laziness on consistency



- Linearizability / sequential: Eager replication
- Trades off low-latency for consistency

Implications of laziness on consistency



- Causal consistency: Lazy replication
- Trades off consistency for low-latency
- Maintain local ordering when replicating
- Operations may be lost if failure before replication

Consistency vs Scalability

Scalability: Adding more machines allows more data to be stored and more operations to be handled!

System	Consistency	Scalable?
Dynamo	Eventual	Yes
Bayou	Causal	No
Paxos/RAFT	Linearizable	No

It's time to think about scability!

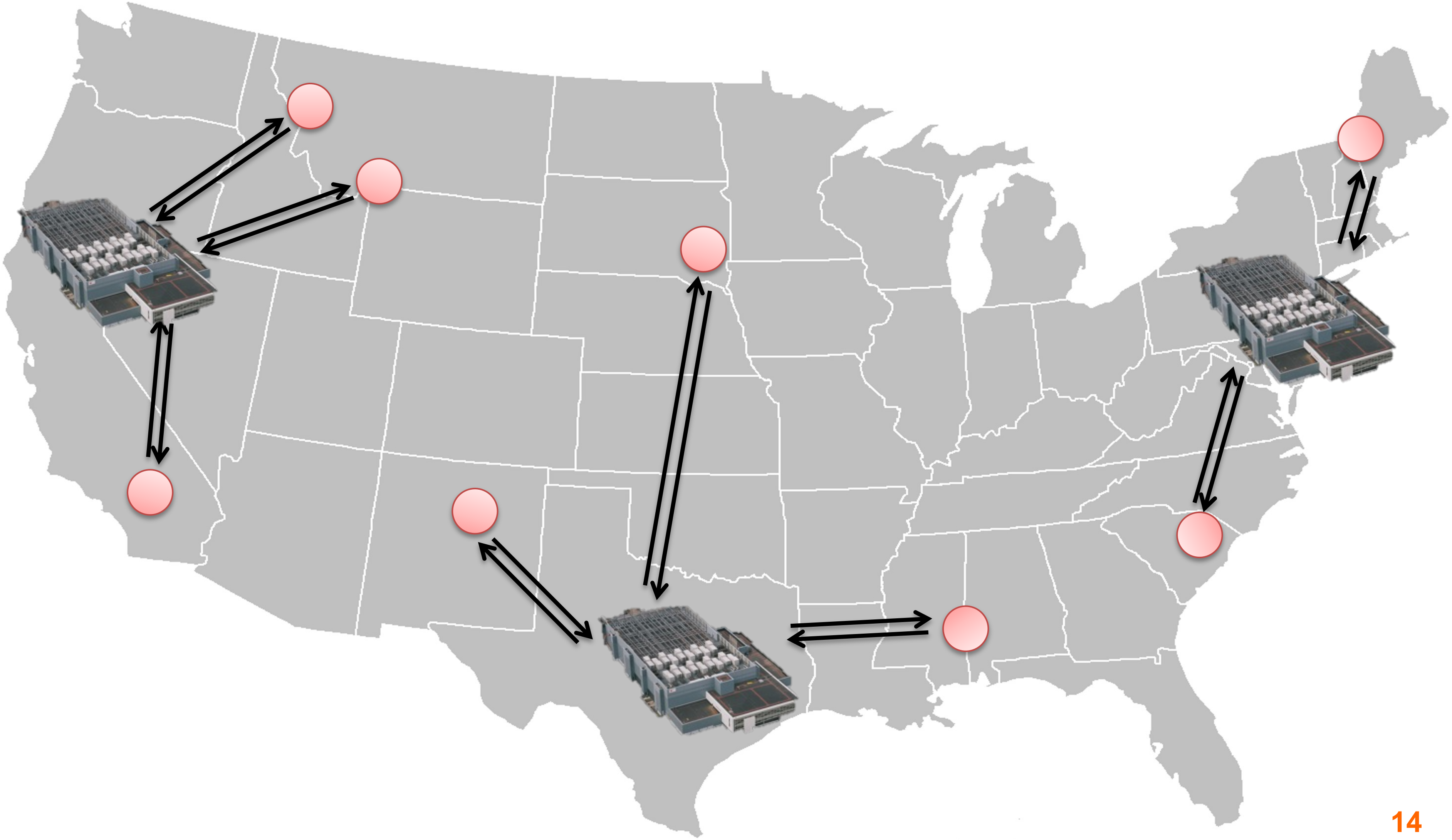
Consistency vs Scalability

Scalability: Adding more machines allows more data to be stored and more operations to be handled!

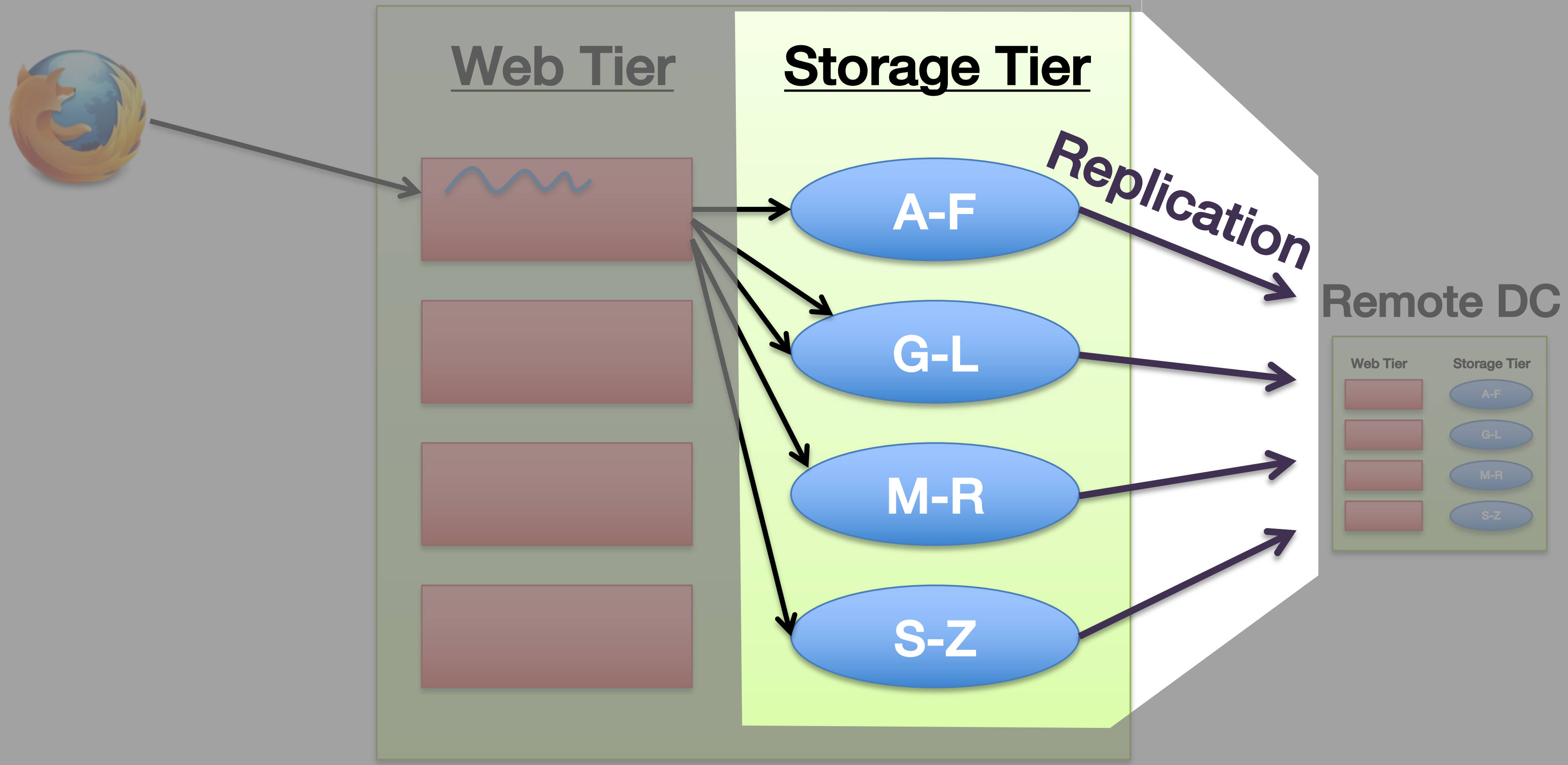
System	Consistency	Scalable?
Dynamo	Eventual	Yes
Bayou	Causal	No
COPS	Causal	Yes
Paxos/RAFT	Linearizable	No

COPS:
Scalable Causal Consistency for Geo-Replicated Storage

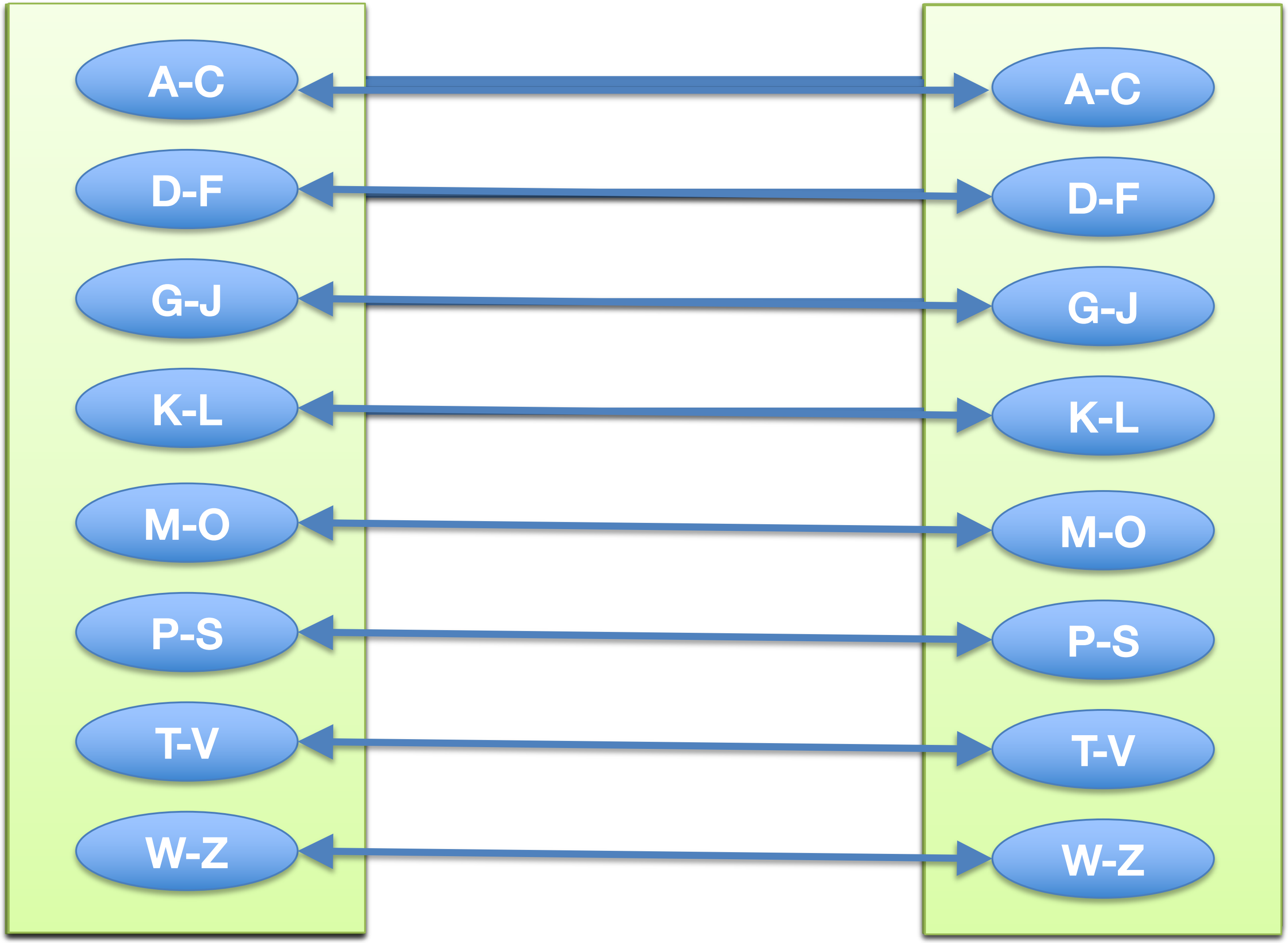
Geo-Replicated Storage:
Serve User Requests Quickly



Inside the Datacenter



Scalability through Sharding



Causality By Example

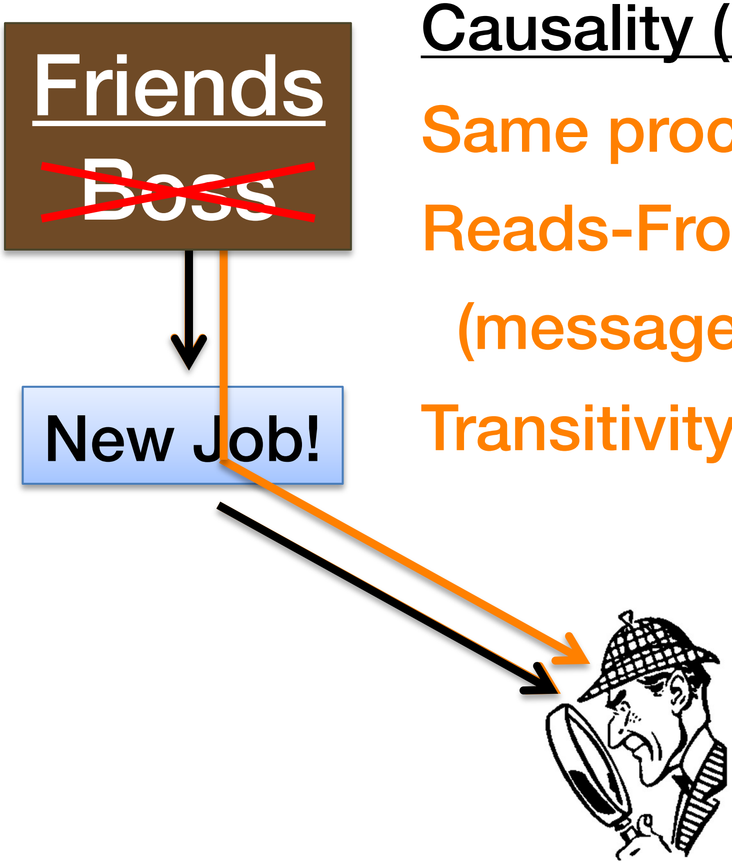
Remove boss from

friends group

Post to friends:

“Time for a new job!”

Friend reads post



Causality ($\longrightarrow$)

Same process

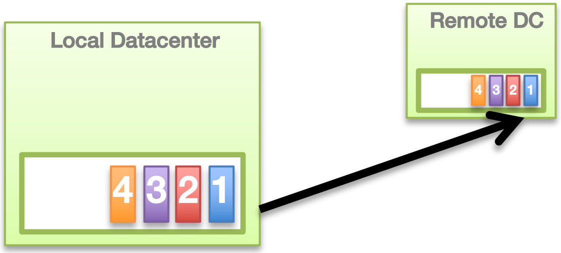
Reads-From

(message receipt)

Transitivity

Bayou’s Causal Consistency

- Log-exchange based



- Log is single serialization point within DC
  - √ **Implicitly** captures & enforces causal order

Sharded Log Exchange

- What happens if we use a separate log per shard?

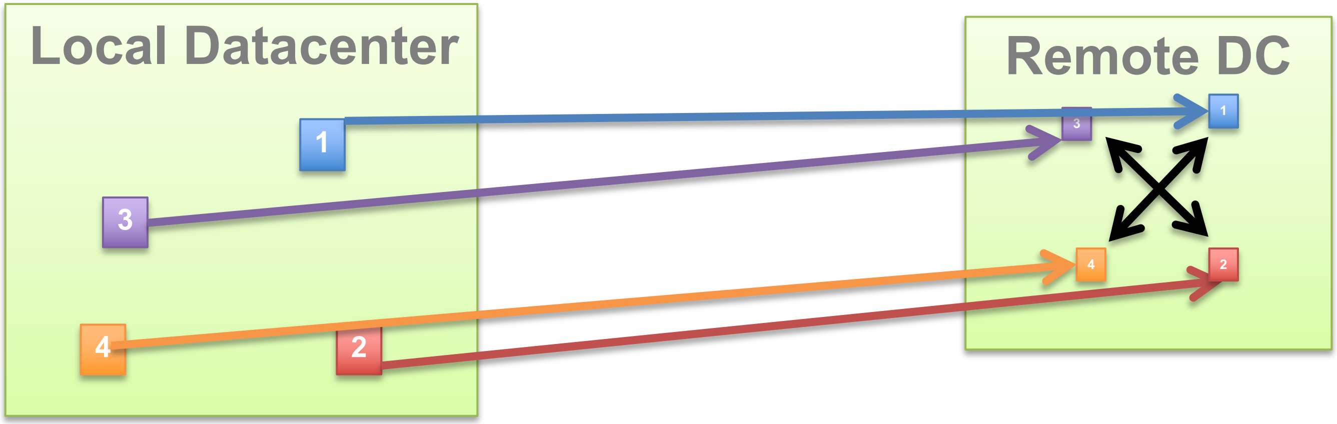
- What happens if we use a single log?

Scalability Key Idea

- Capture causality with explicit dependency metadata

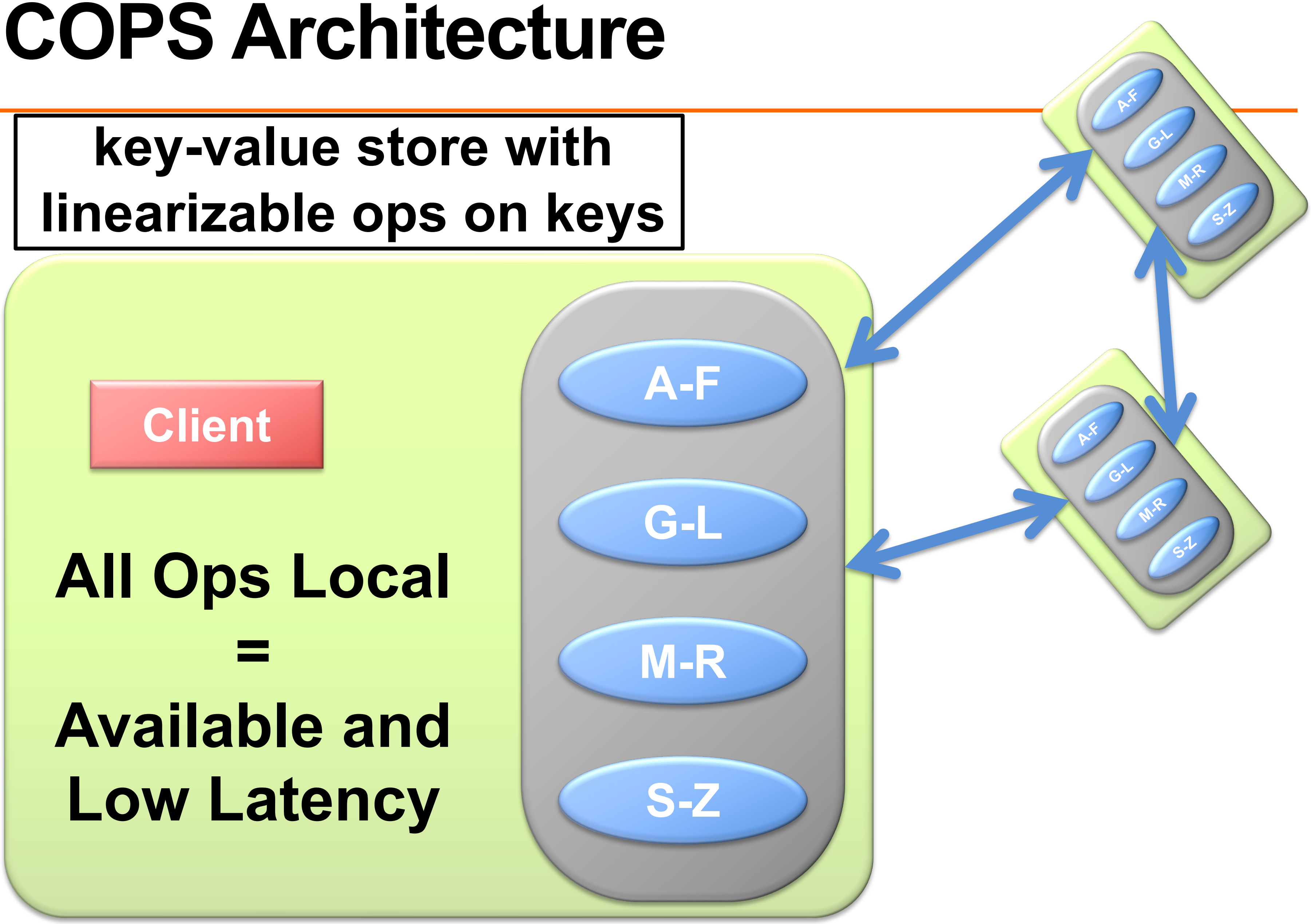
  3 **after** 1

- Enforce with distributed verifications
  - Delay exposing replicated writes until all dependencies are satisfied in the datacenter



COPS Architecture

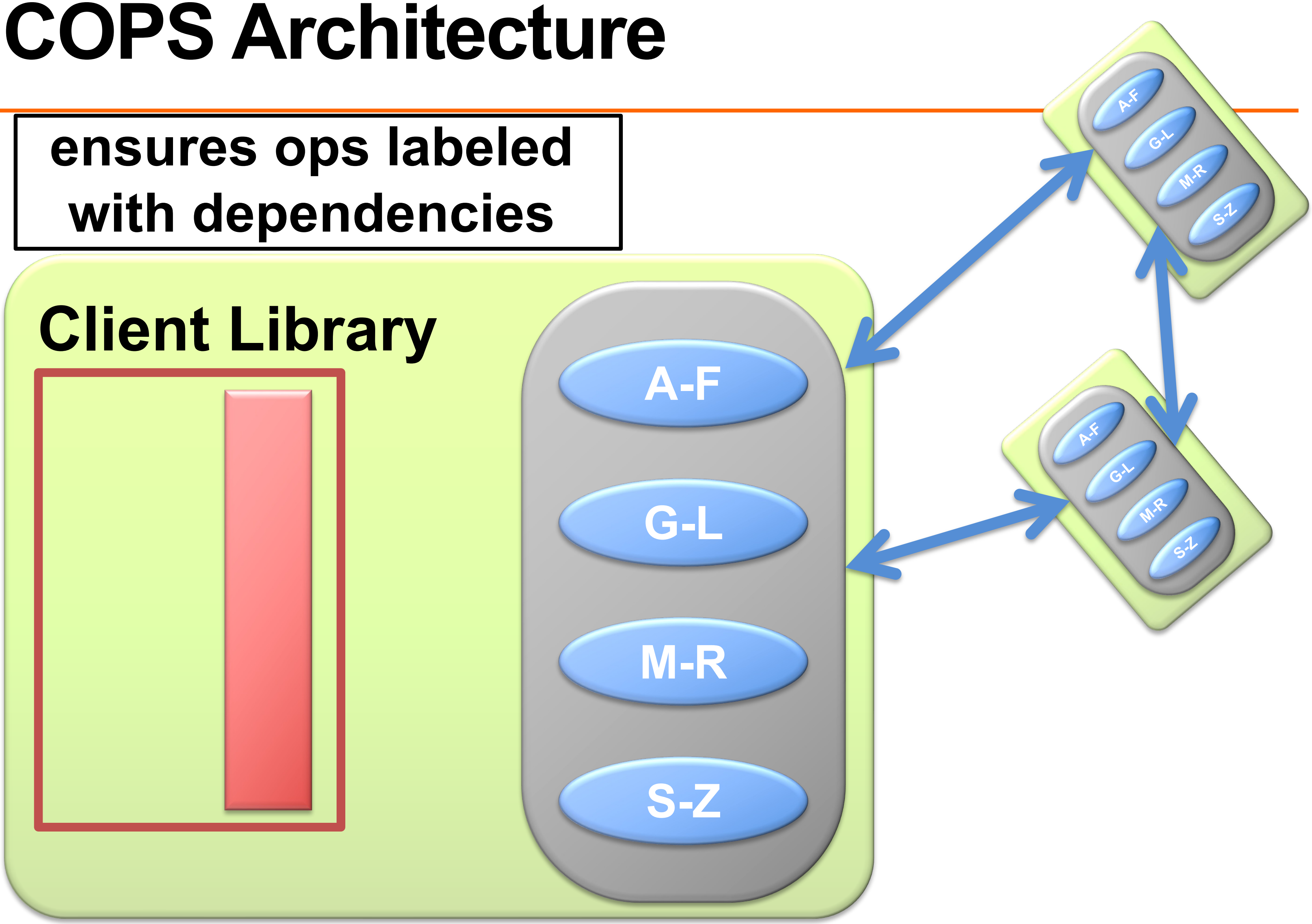
key-value store with linearizable ops on keys



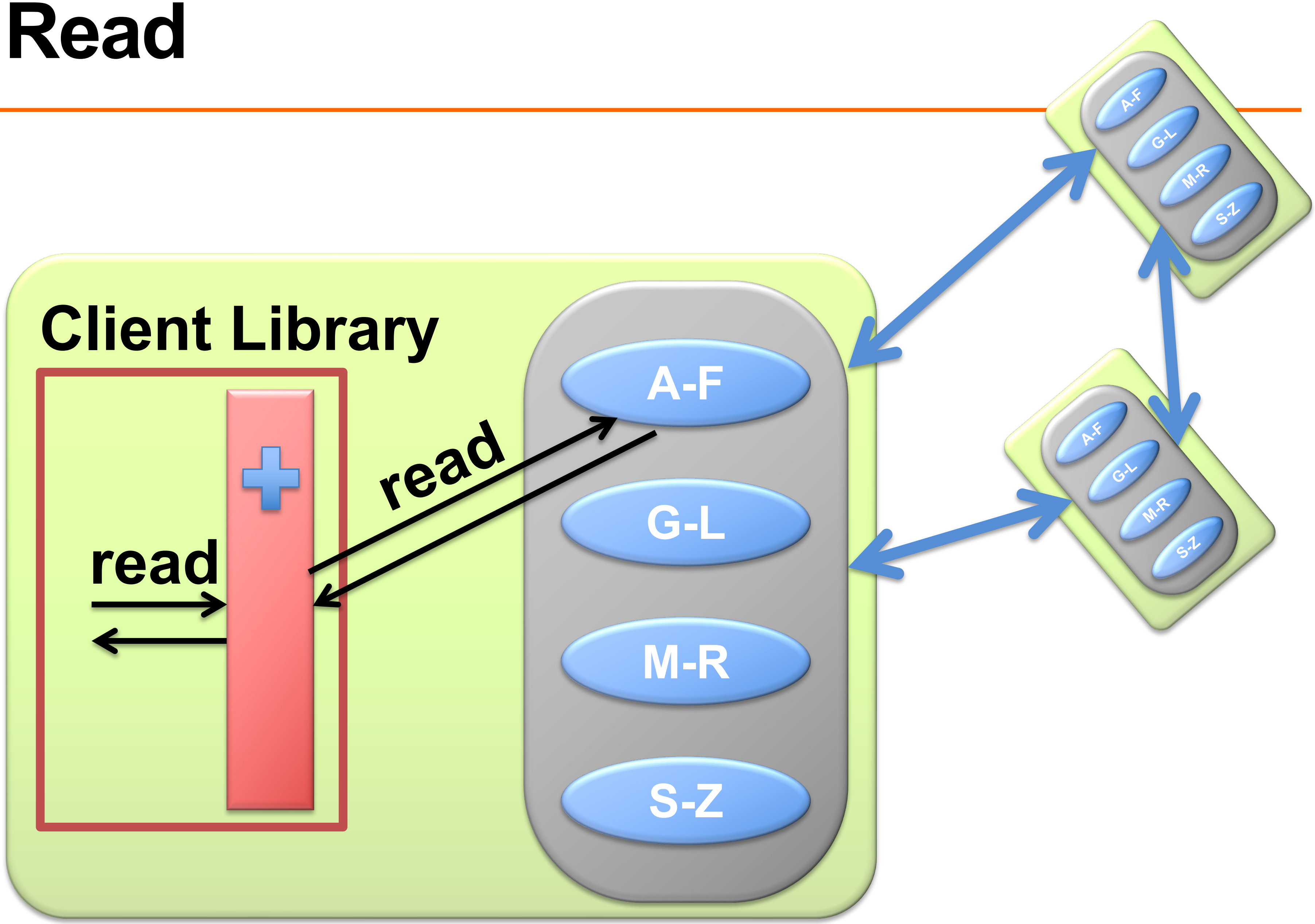
All Ops Local = Available and Low Latency

COPS Architecture

ensures ops labeled with dependencies

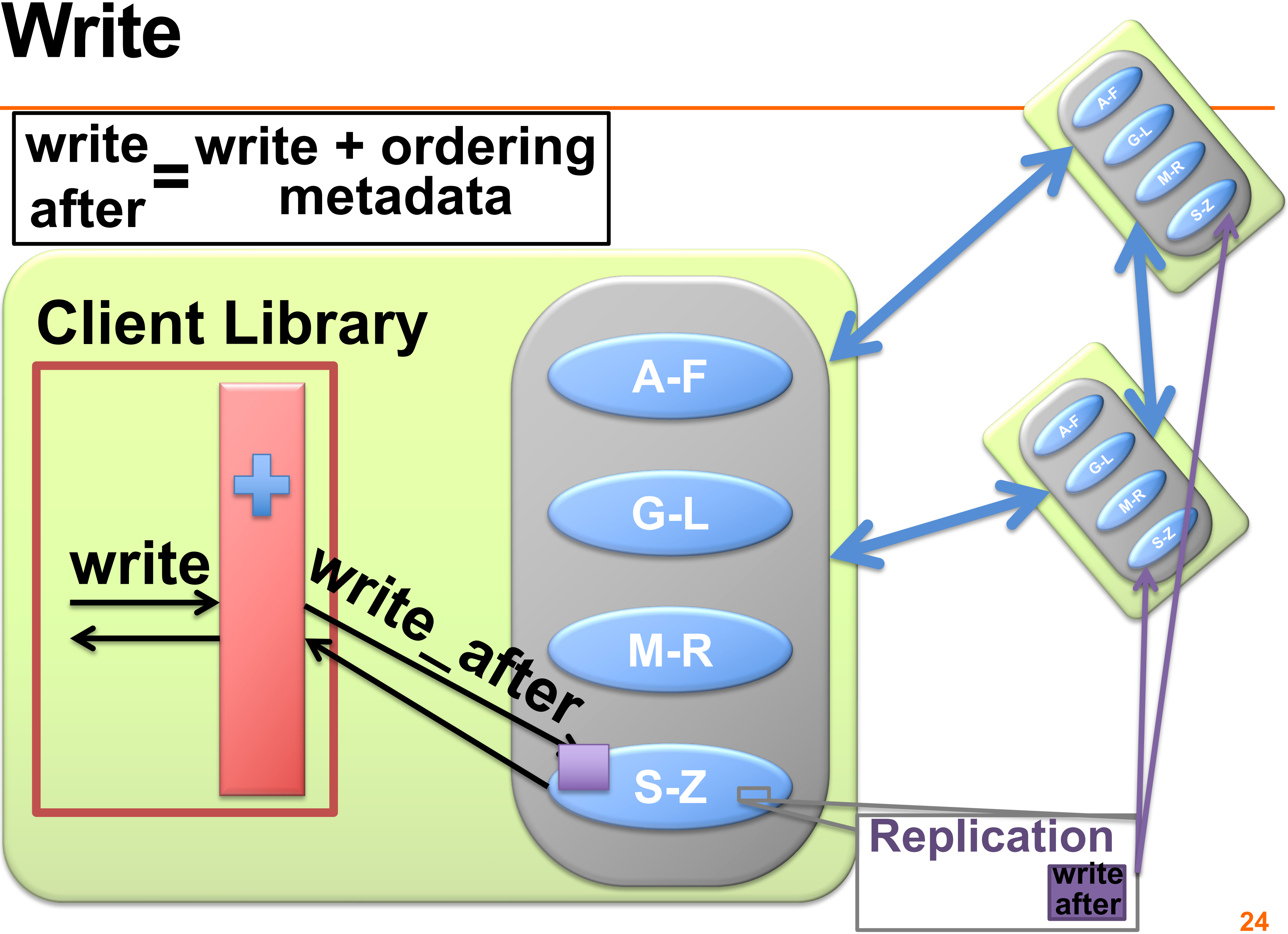


Read



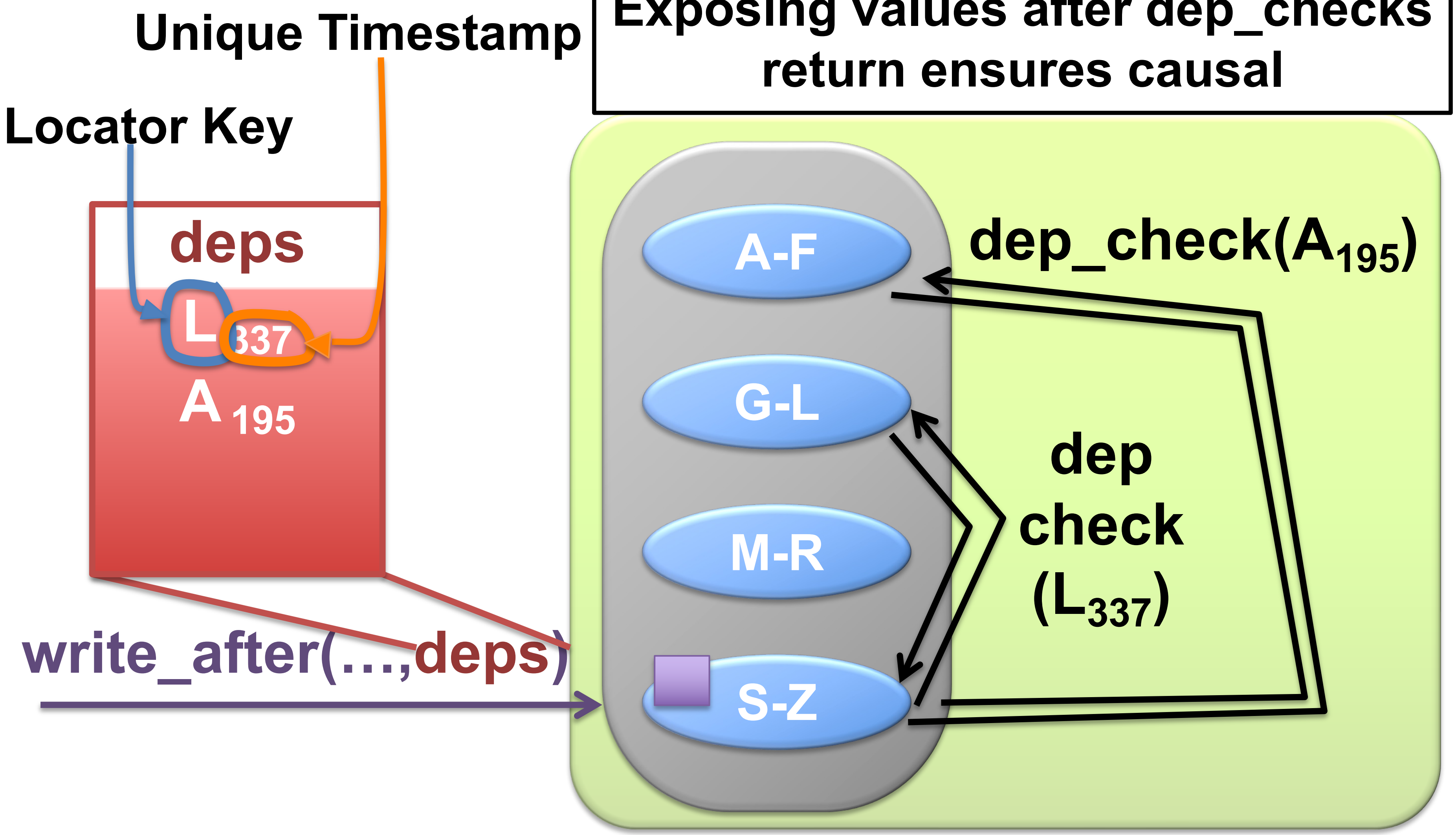
Write

write_after = write + ordering metadata



Replicated Write

Exposing values after dep_checks return ensures causal



Basic Architecture Summary

- All ops local, replicate in background
  - Availability and low latency
- Shard data across many nodes
  - Scalability
- Control replication with dependencies
  - Causal consistency

Scalability

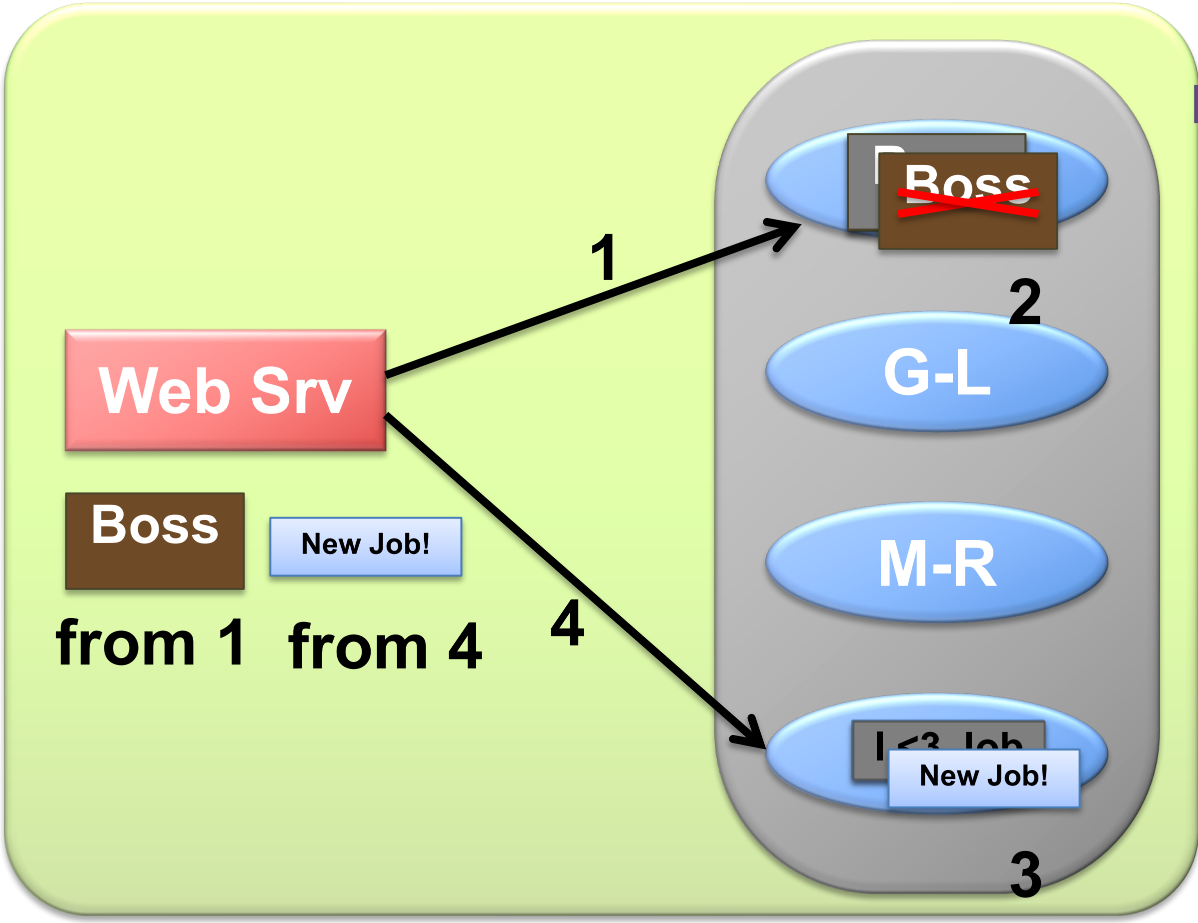
- Shard data for scalable storage

- New distributed protocol for scalably applying writes across shards

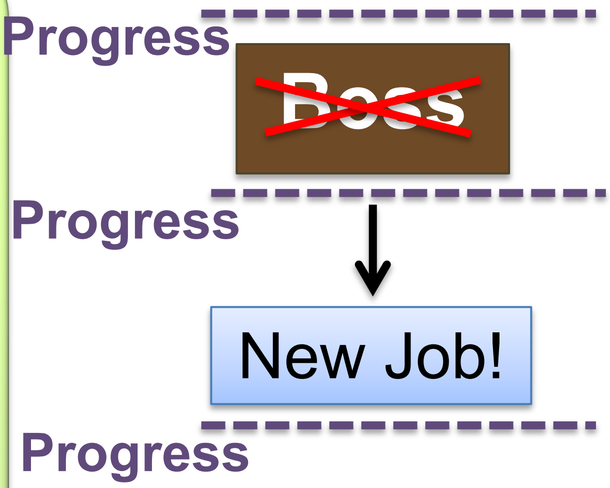
- Also need a new distributed protocol for consistently reading data across shards…

Reads Aren't Enough

Asynchronous requests + distributed data = ??

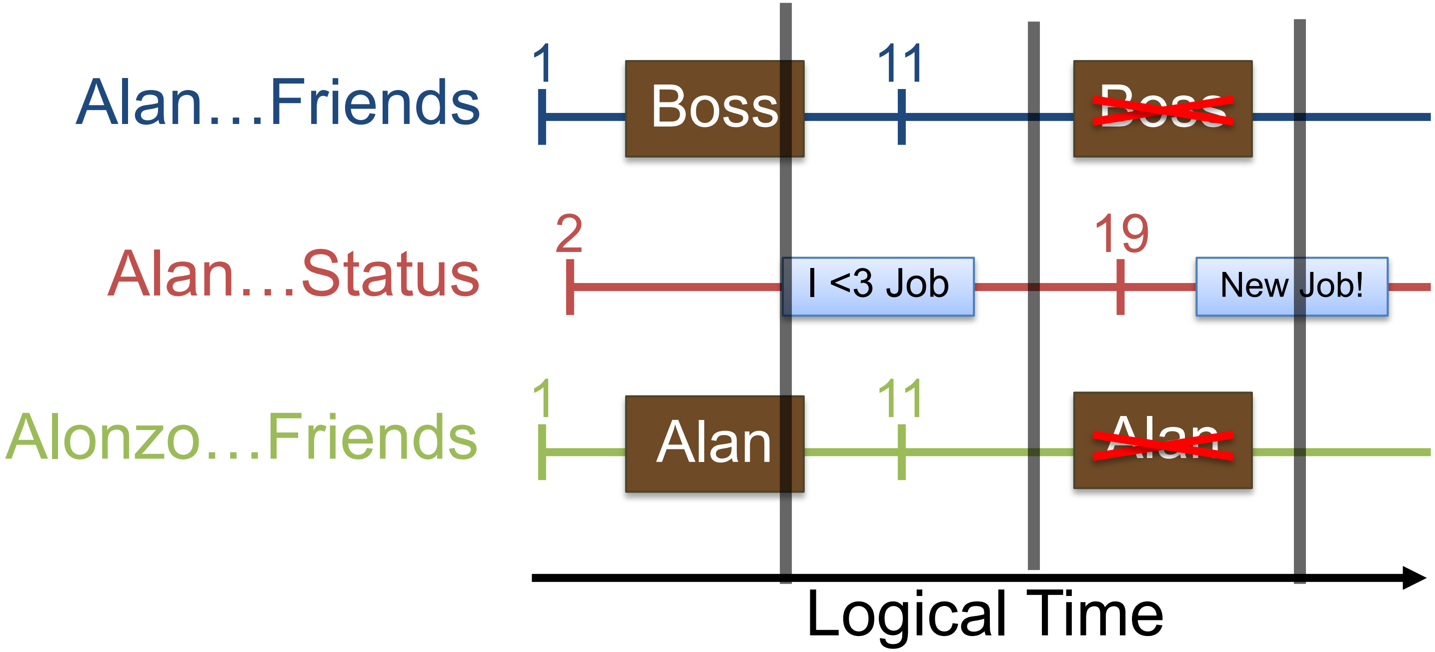


Turing's Operations



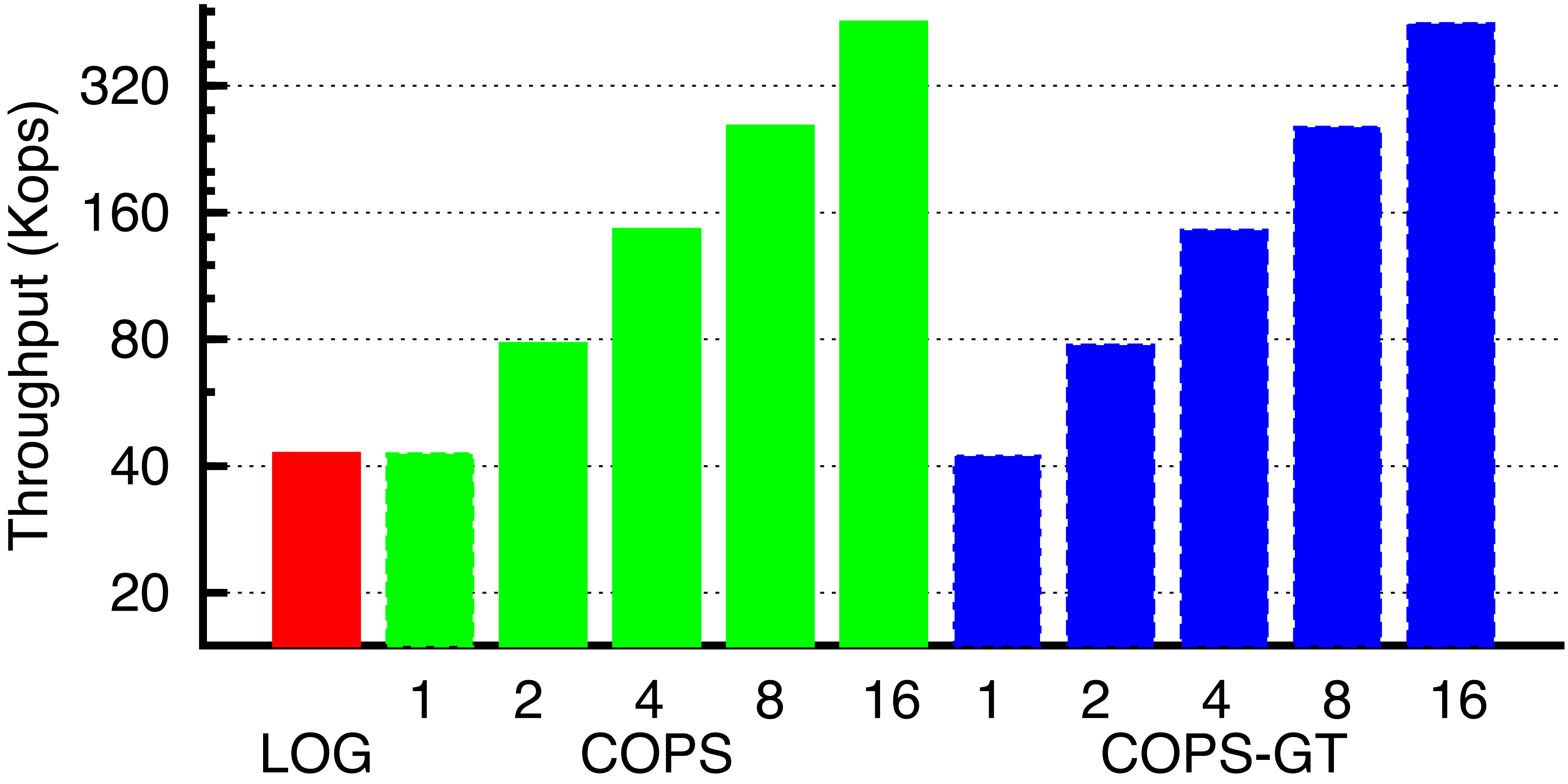
Read-Only Transactions

- Consistent up-to-date view of data
  - Across many servers



More on transactions next time!

COPS Scaling Evaluation



More servers => More operations/sec

COPS Summary

- Scalable causal consistency
  - Shard for scalable storage
  - Distributed protocols for coordinating writes and reads
    - Evaluation confirms scalability
- All operations handled in local datacenter
  - Availability
  - Low latency
- We're thinking scalably now!
  - Next time: scalable strong consistency