

Scalable Causal Consistency



COS 418/518: Distributed Systems
Lecture 15

Jialin Ding, Mike Freedman

Consistency Hierarchy (review)

Linearizability

e.g., RAFT



Sequential Consistency



Causal+ Consistency

e.g., Bayou



Eventual Consistency

e.g., Dynamo

Causal+ Consistency (review)

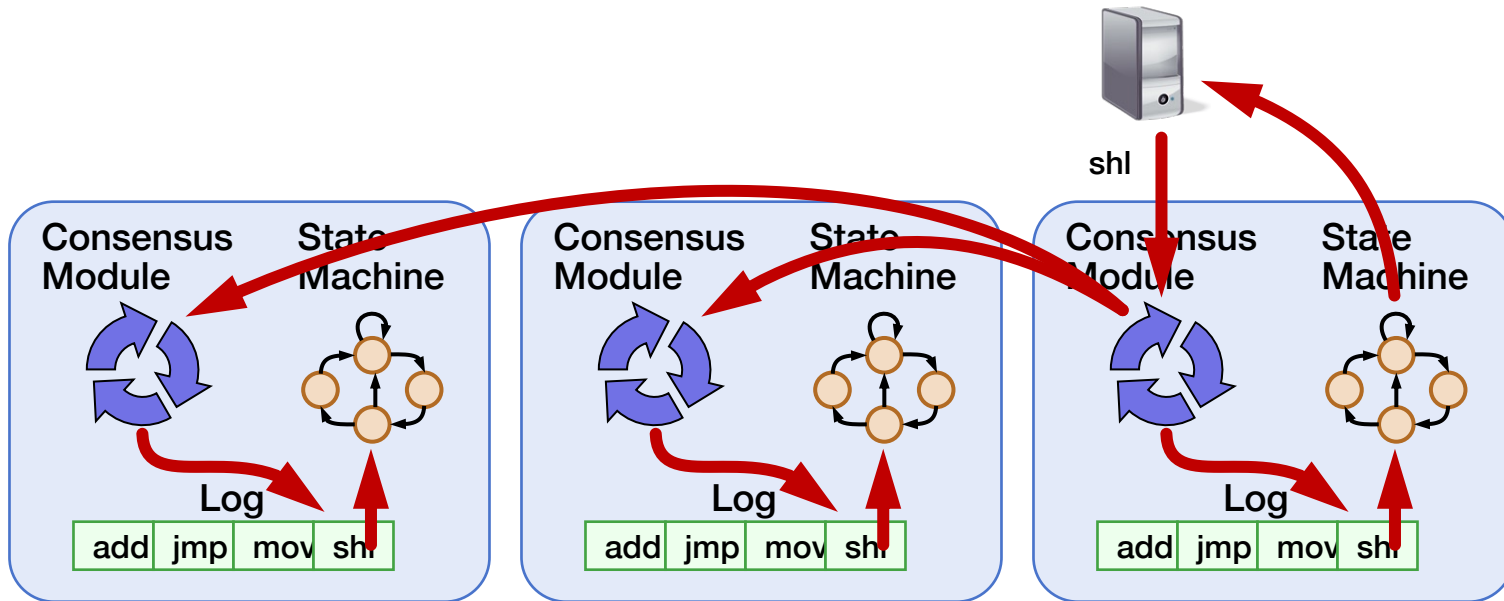
1. Writes that are **potentially** causally related must be seen by all processes in same order.
 2. Concurrent writes may be seen in a different order on different processes.
- Concurrent: Ops not causally related

Causal+ Consistency (review)

- Partially orders all operations, does not totally order them
 - Does not look like a single machine
- Guarantees
 - For each process, $\exists$ an order of all writes + that process's reads
 - Order respects the happens-before ($\rightarrow$) ordering of operations
 - + replicas converge to the same state
 - Skip details, makes it stronger than eventual consistency

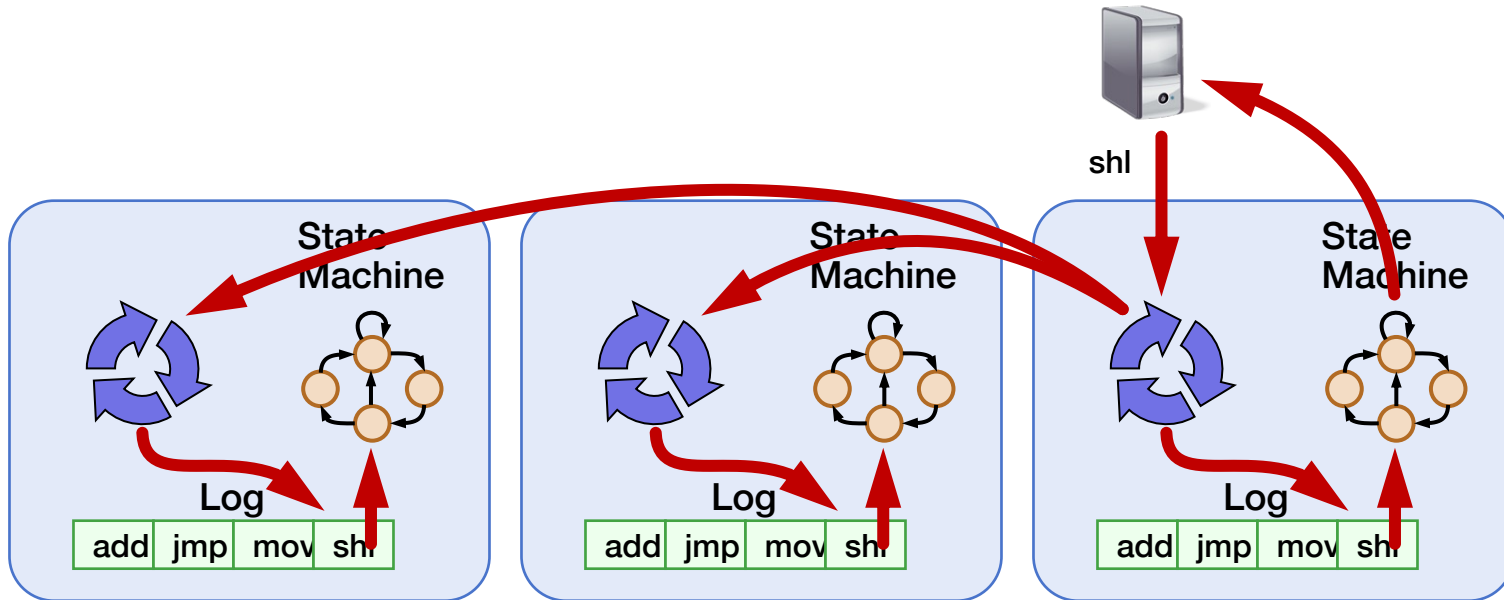
Causal consistency within replicated systems

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
- 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?
Paxos/RAFT	Linearizable	No
Bayou	Causal+	No

It's time to think about scalability!

Consistency vs Scalability

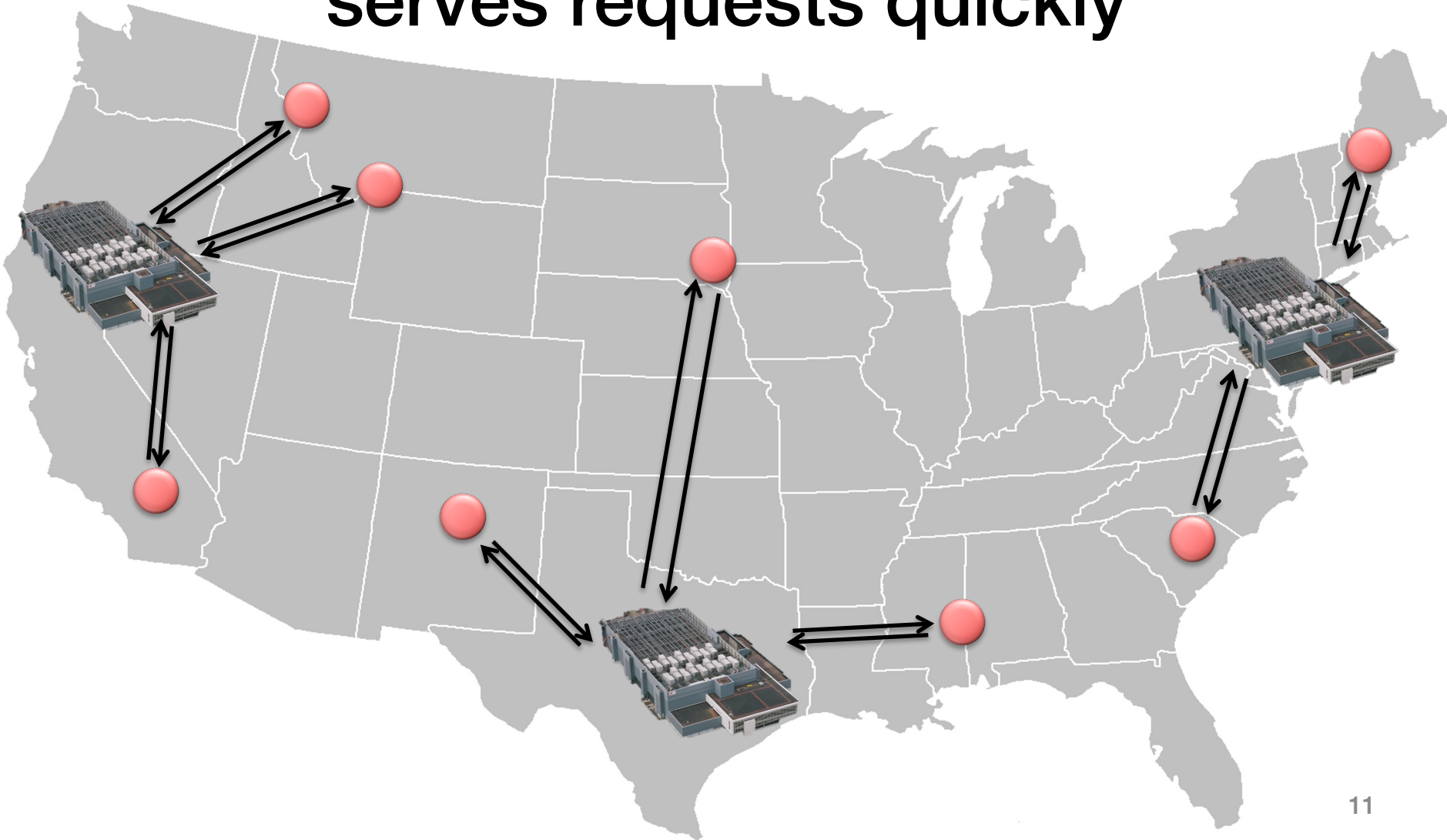
Scalability: Adding more machines allows more data to be stored and more operations to be handled!

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

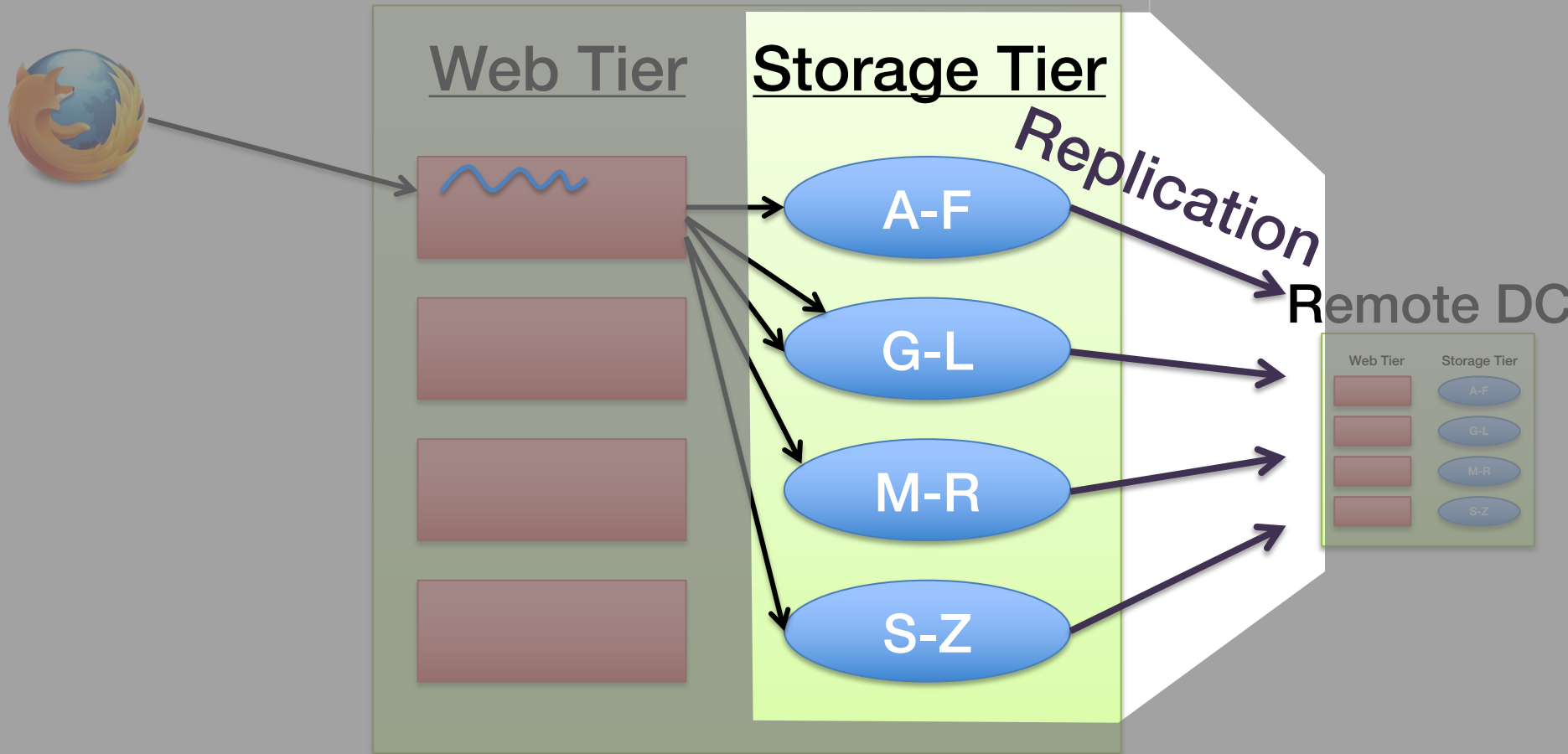
COPS: **Scalable Causal Consistency** **for Geo-Replicated Storage**

Geo-Replicated Storage

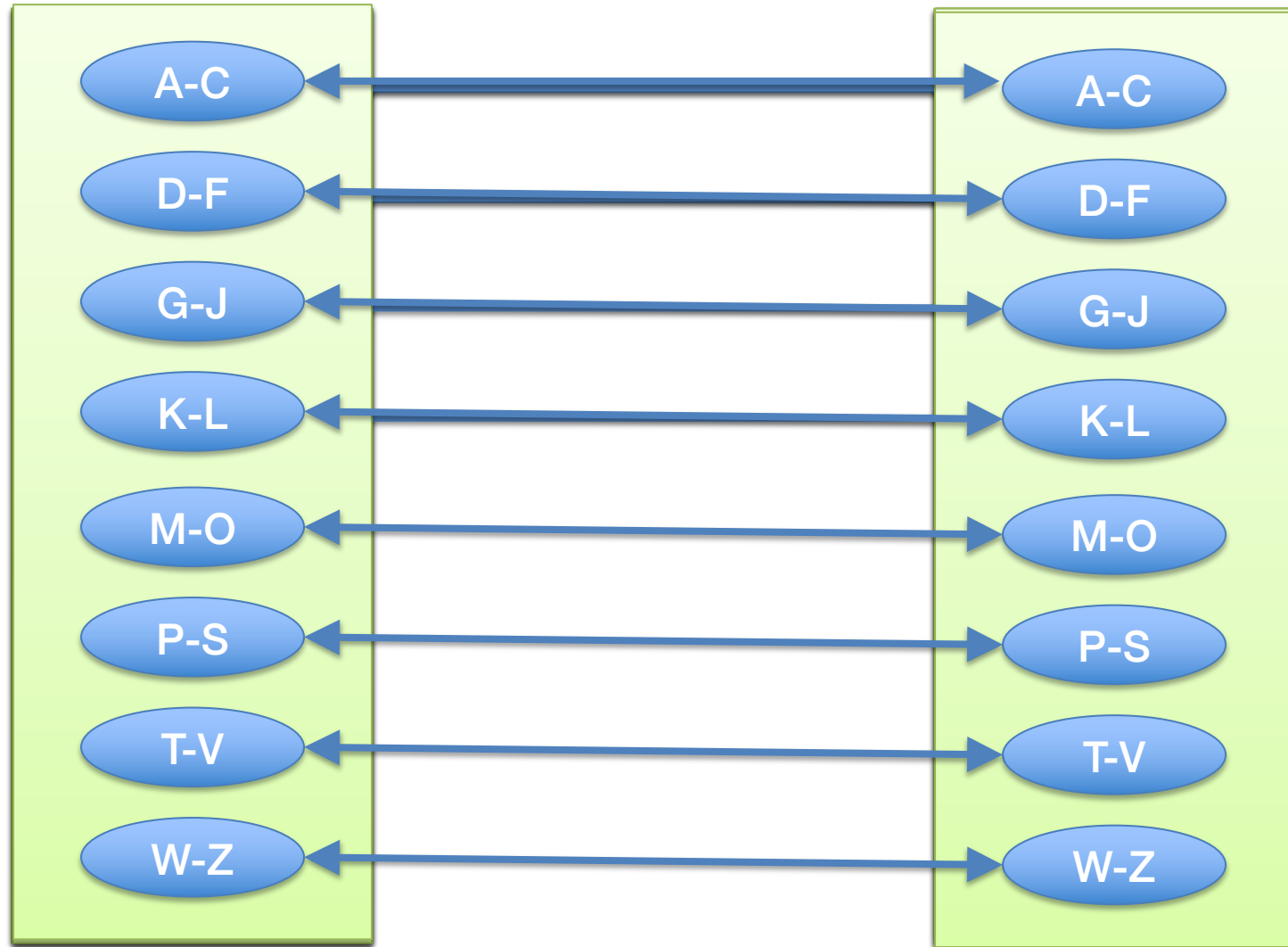
serves requests quickly



Inside the Datacenter

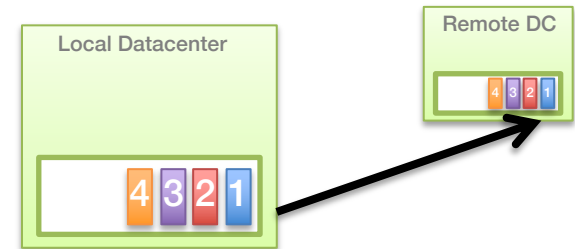


Scalability through Sharding



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?

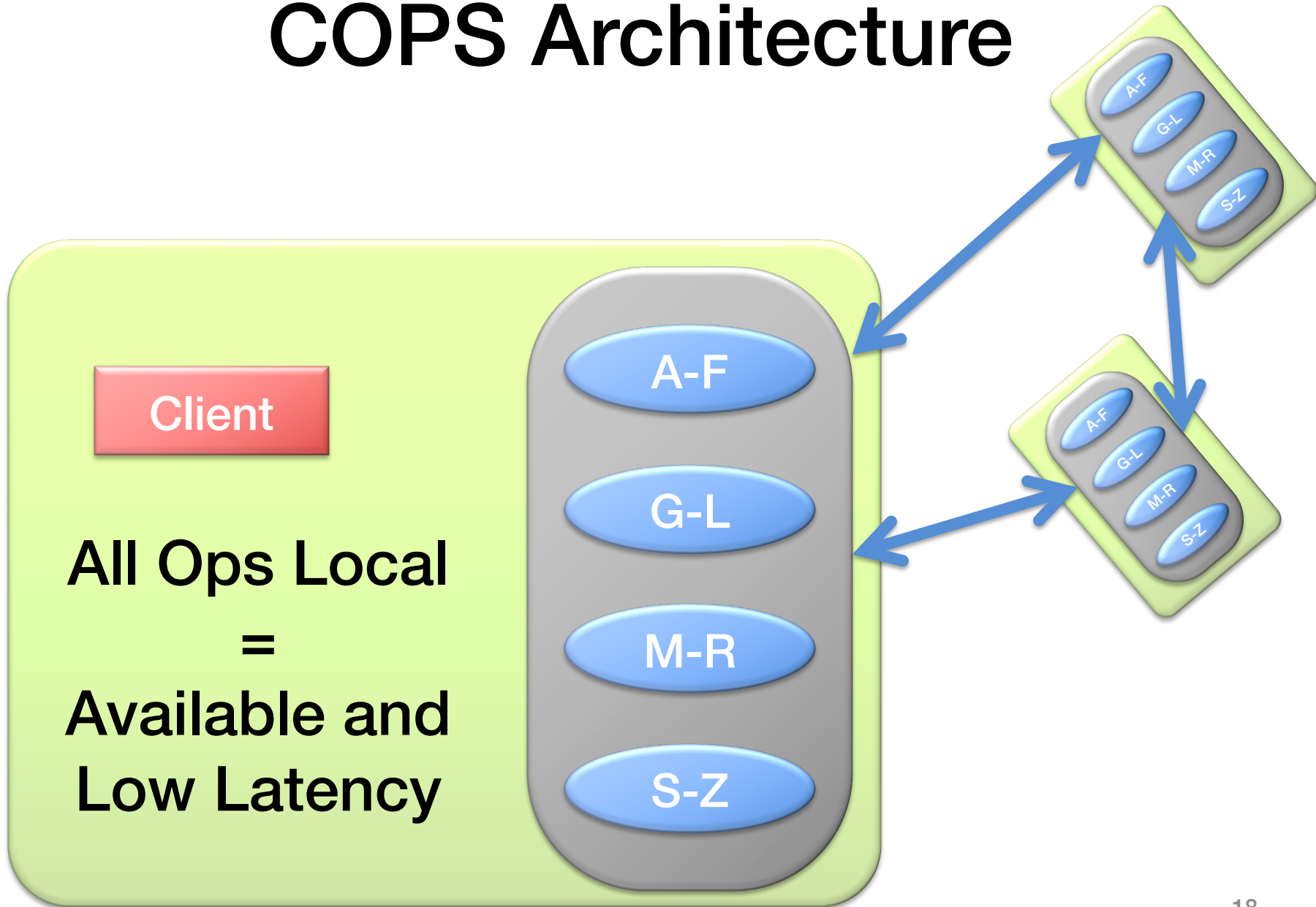
Scalability Idea

- Each key is associated with a monotonically increasing version number
- Capture causality with per-client dependency metadata

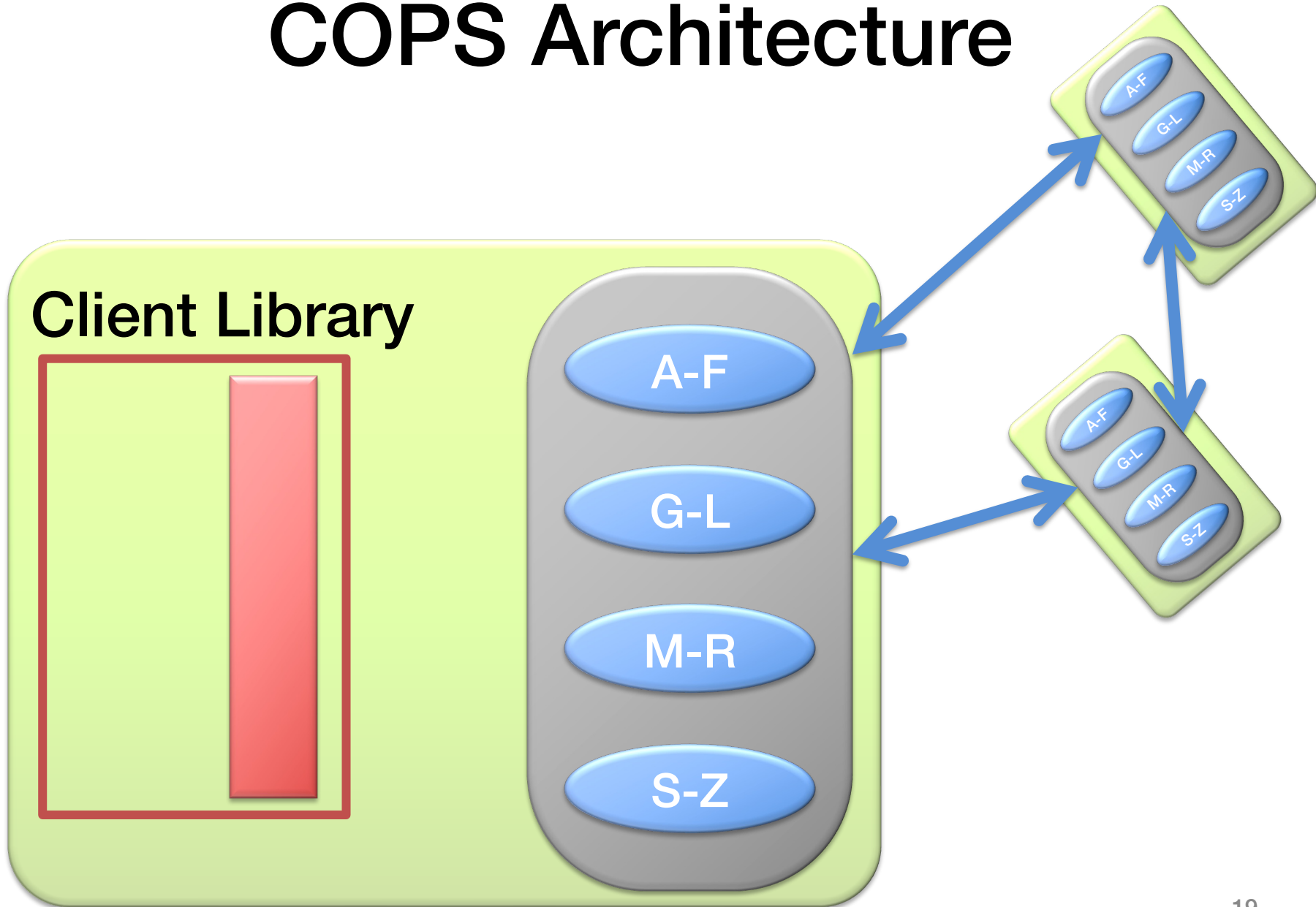
Client operations	Read key 1	Read key 2	Write key 3
Dependency metadata	Key 1 (version 100)	Key 2 (version 25)	Key 3 (version 50)

- Delay exposing replicated writes until all dependencies are satisfied in the datacenter

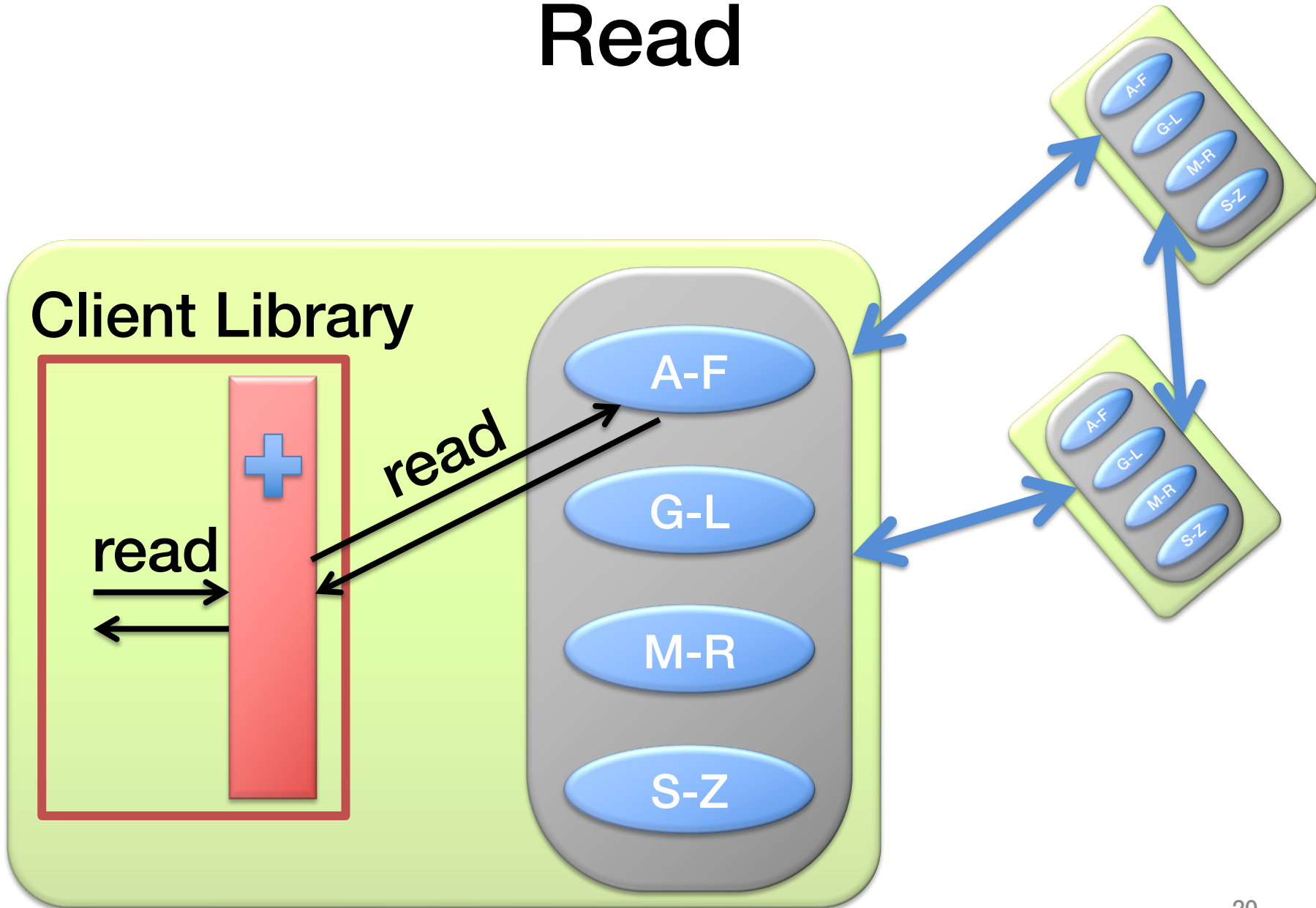
COPS Architecture



COPS Architecture

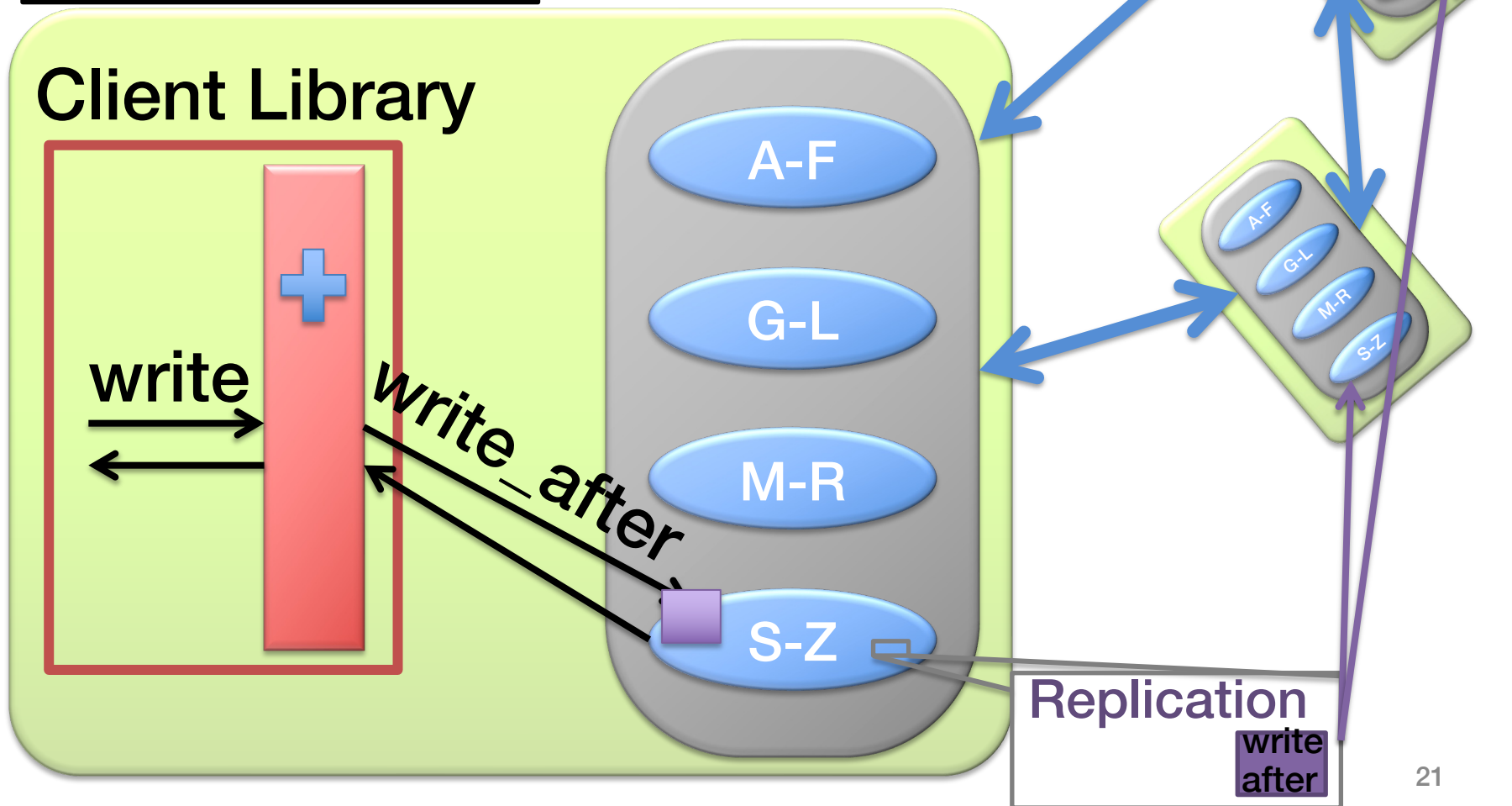


Read



write
after = write
+ ordering
metadata

Write

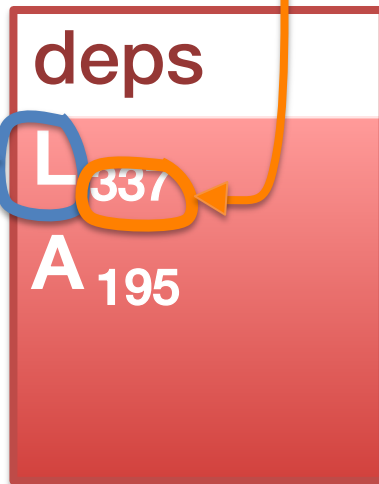


Replicated Write

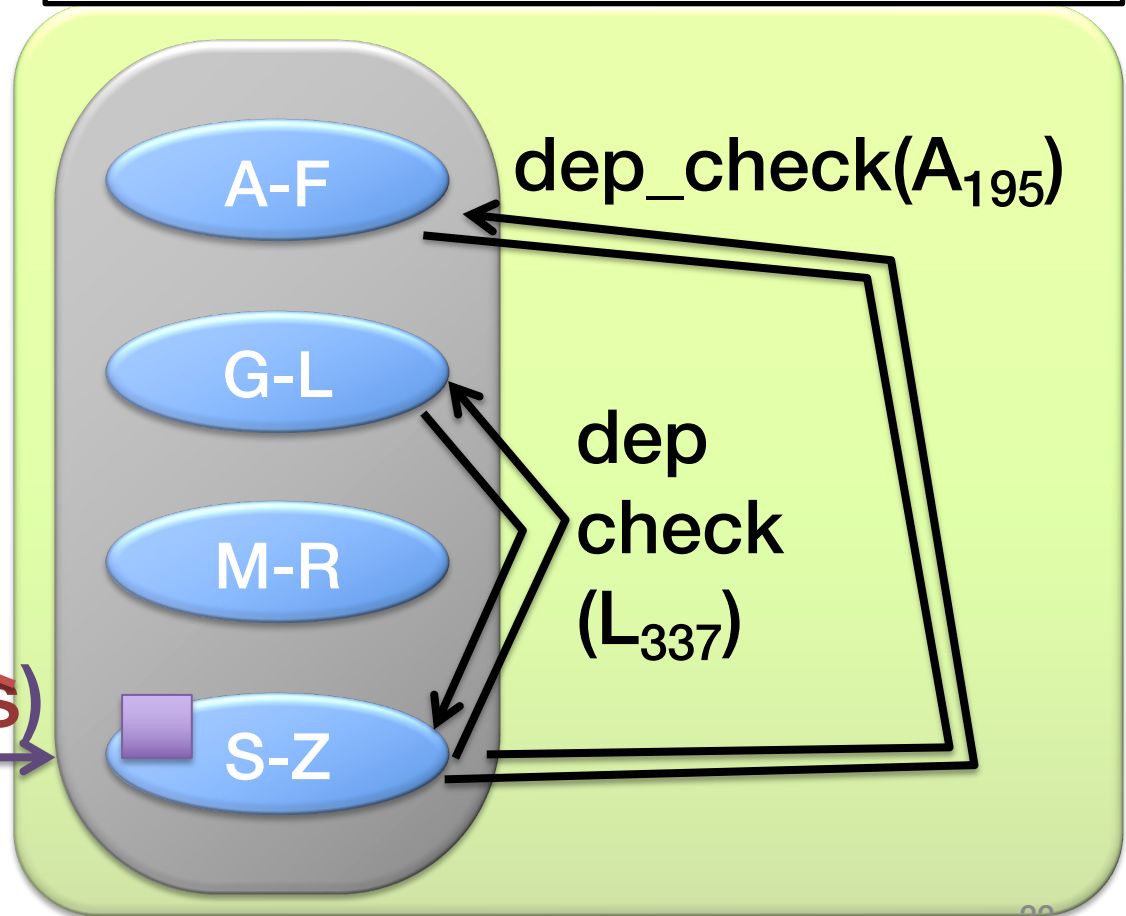
Exposing values after dep_checks return ensures causal

Version Number

Key



write_after(...,deps)

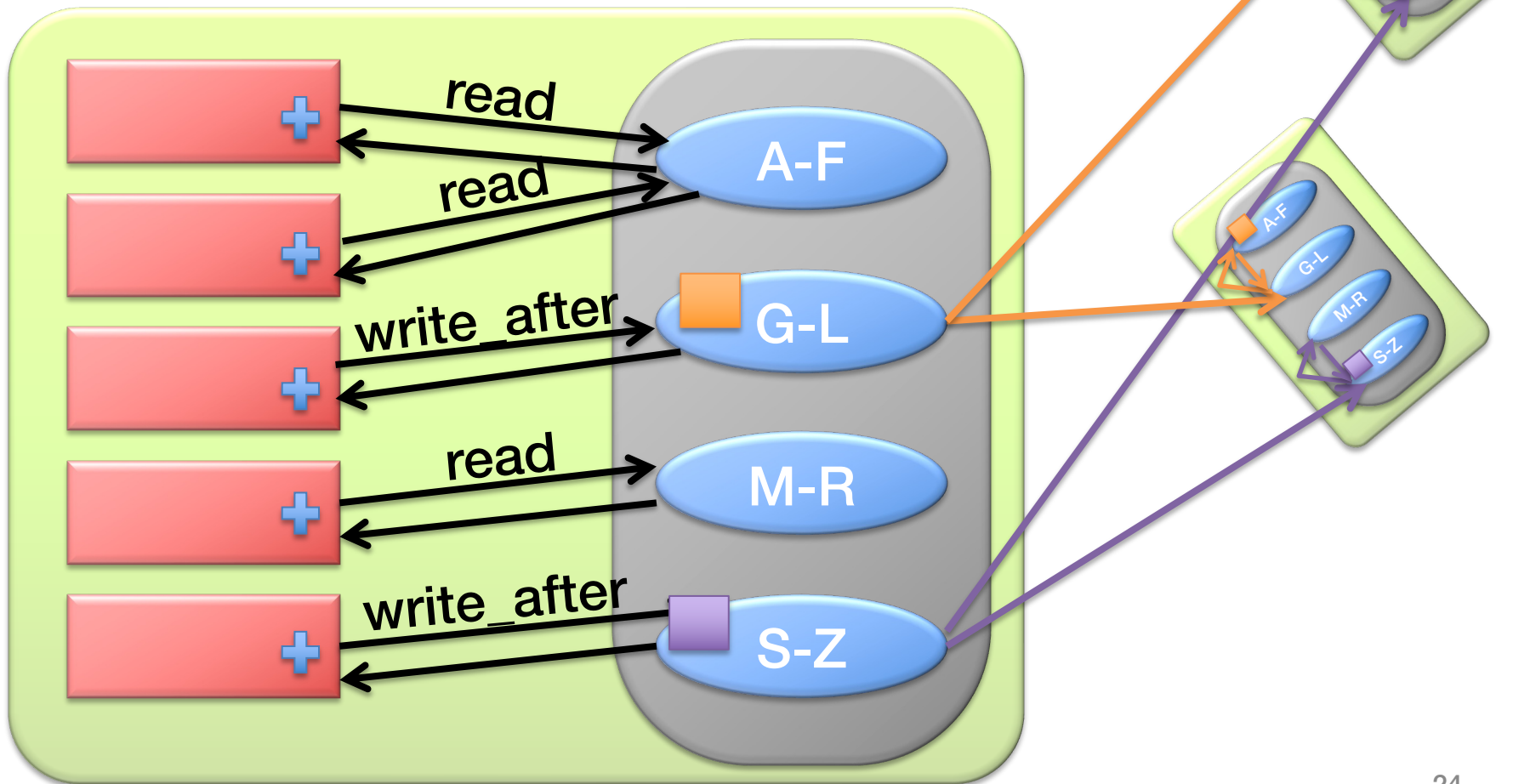


Basic Architecture Summary

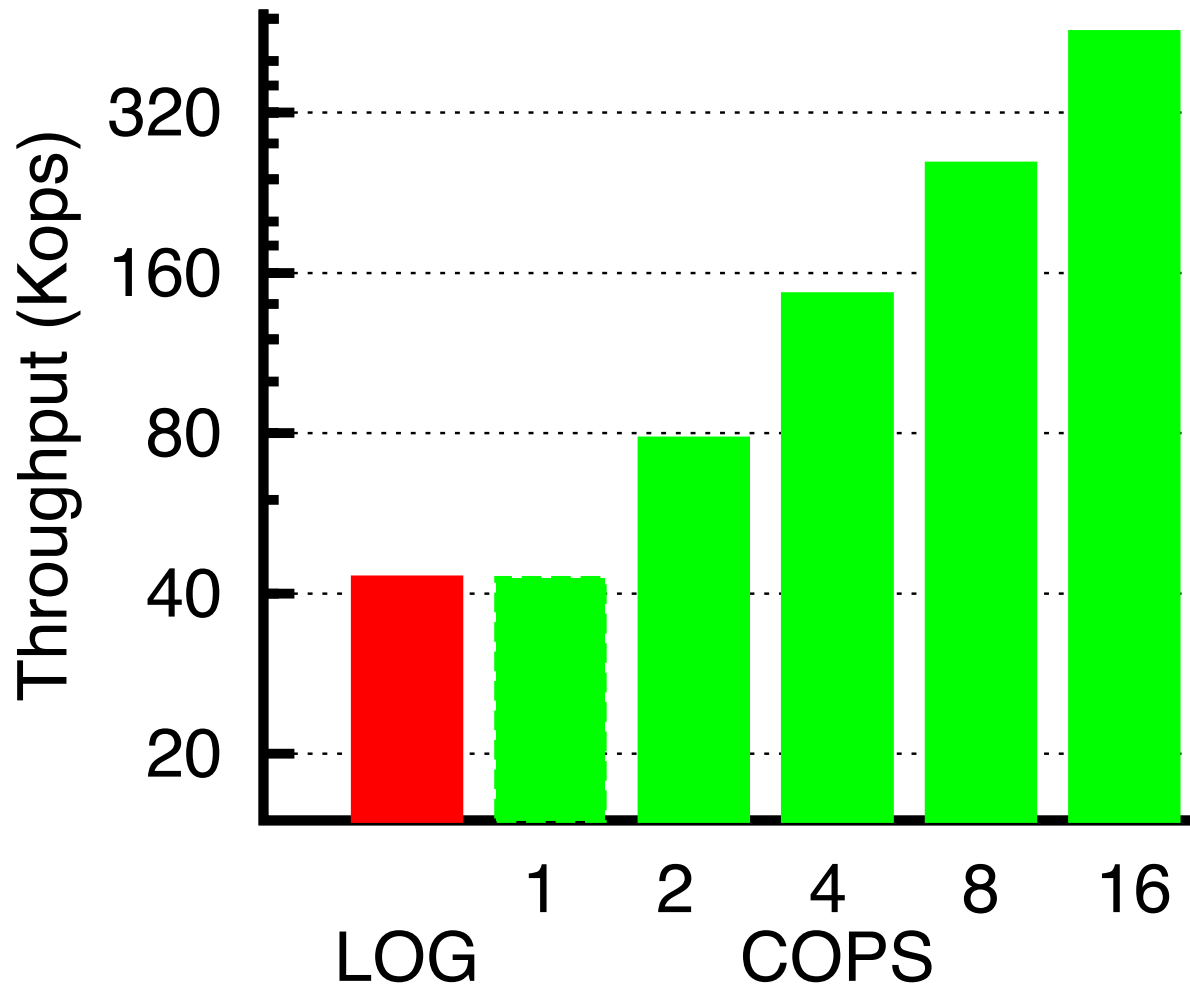
- All ops local, replicate in background
 - Availability and low latency
- Shard data across many nodes
 - Scalability
- Control replication with dependencies
 - Causal consistency

Scalable Causal+

From fully distributed operation



COPS Scaling Evaluation



More servers => More operations/sec

COPS

- **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

