Scalable Causal Consistency

COS 418: Distributed Systems
Lecture 14

Wyatt Lloyd

Consistency Hierarchy

Linearizability — e.g., RAFT

Sequential Consistency

CAP

PRAM 1988 (Princeton)

Causal+ Consistency — e.g., Bayou

Eventual Consistency — e.g., Dynamo

Consistency Hierarchy

Linearizability — e.g., RAFT

Sequential Consistency

CAP

Part 1: More on consistency

PRAM 1988 (Princeton)

Causal+ Consistency — e.g., Bayou

P2: Scalable Causal Consistency

Eventual Consistency

Last Time's Causal+ Consistency

- 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 (→) ordering of operations
  - + replicas converge to the same state
    - Skip details, makes it stronger than eventual consistency

Causal Consistency

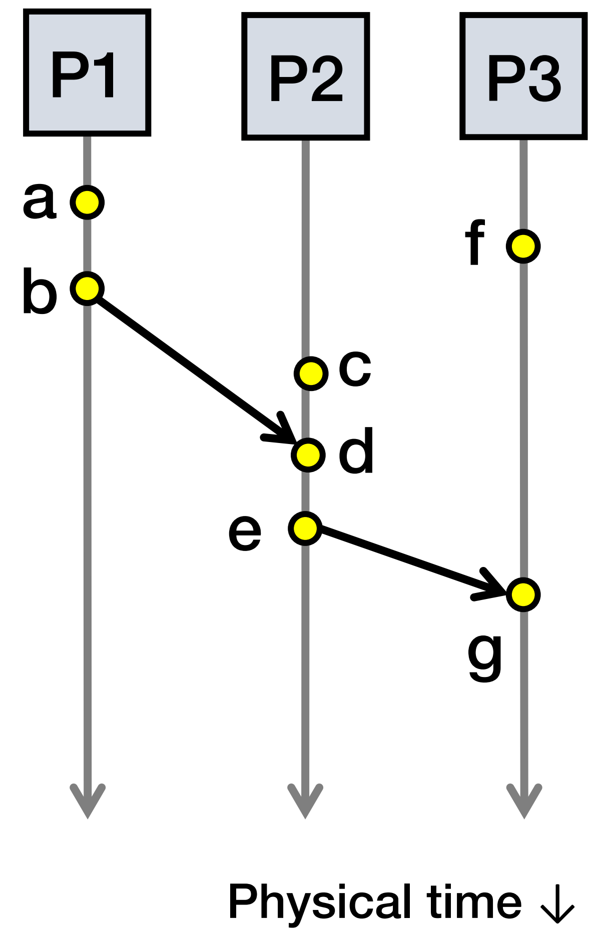
1. Writes that are potentially causally related must be seen by all processes in same order.
2. Concurrent writes may be seen in a different order on different processes.

- Concurrent: Ops not causally related

Causal Consistency

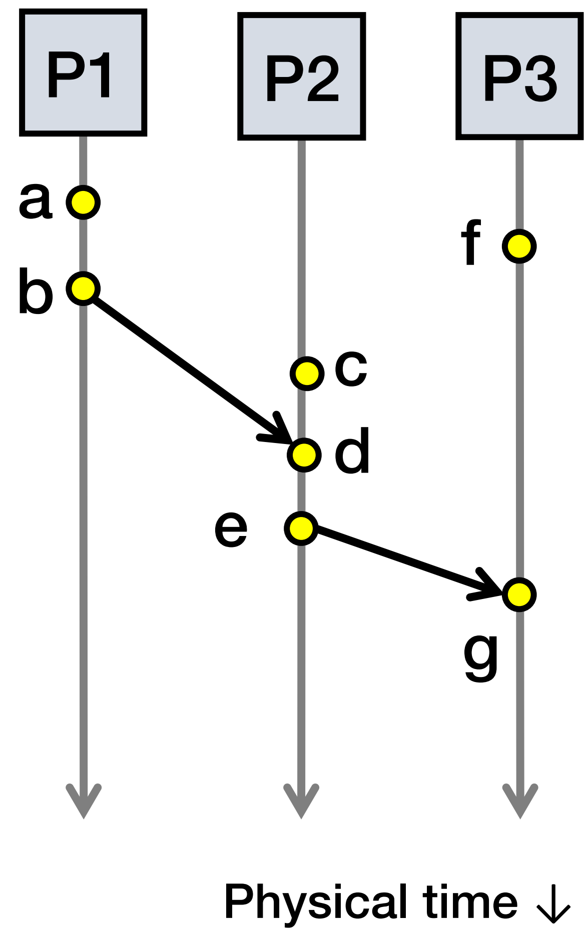
1. Writes that are potentially causally related must be seen by all processes in same order.
2. Concurrent writes may be seen in a different order on different processes.

- Concurrent: Ops not causally related



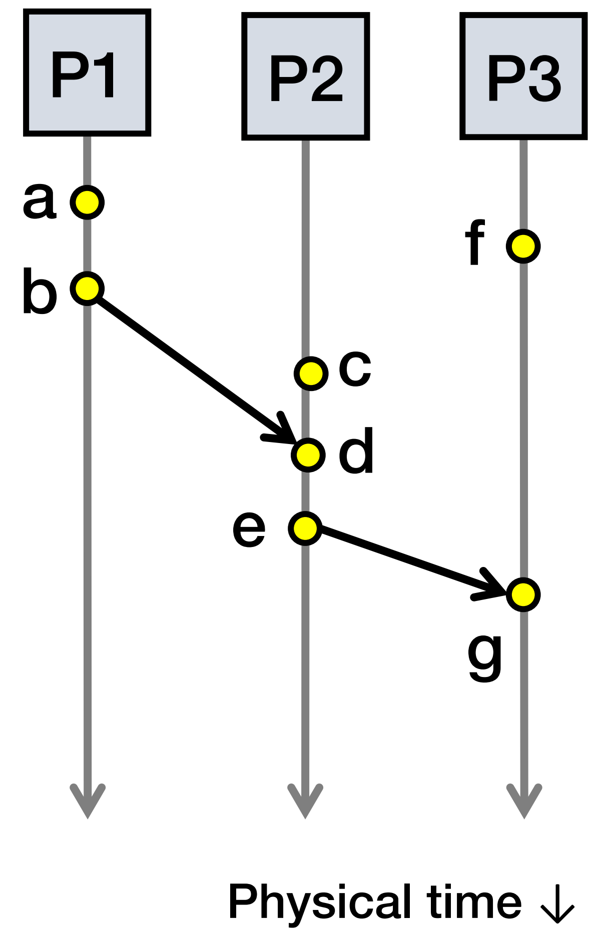
Causal Consistency

Operations	Concurrent?
a, b	
b, f	
c, f	
e, f	
e, g	
a, c	
a, e	



Causal Consistency

Operations	Concurrent?
a, b	N
b, f	Y
c, f	Y
e, f	Y
e, g	N
a, c	Y
a, e	N



Causal Consistency: Quiz

P1:	W(x)a			W(x)c		
P2:		R(x)a	W(x)b			
P3:		R(x)a			R(x)c	R(x)b
P4:		R(x)a			R(x)b	R(x)c

- Valid under causal consistency
- **Why?** W(x)b and W(x)c are concurrent
    - So all processes don’t (need to) see them in same order
- P3 and P4 read the values ‘a’ and ‘b’ in order as potentially causally related. No ‘causality’ for ‘b’ and ‘c’.

Sequential Consistency: Quiz

P1:	W(x)a			W(x)c		
P2:		R(x)a	W(x)b			
P3:		R(x)a			R(x)c	R(x)b
P4:		R(x)a			R(x)b	R(x)c

- Invalid under sequential consistency
- Why? P3 and P4 see b and c in different order
- But fine for causal consistency
  - B and C are not causually related

Causal Consistency

P1: W(x)a				
P2:	R(x)a	W(x)b		
P3:			R(x)b	R(x)a
P4:			R(x)a	R(x)b

(a)

P1: W(x)a			
P2:	W(x)b		
P3:		R(x)b	R(x)a
P4:		R(x)a	R(x)b

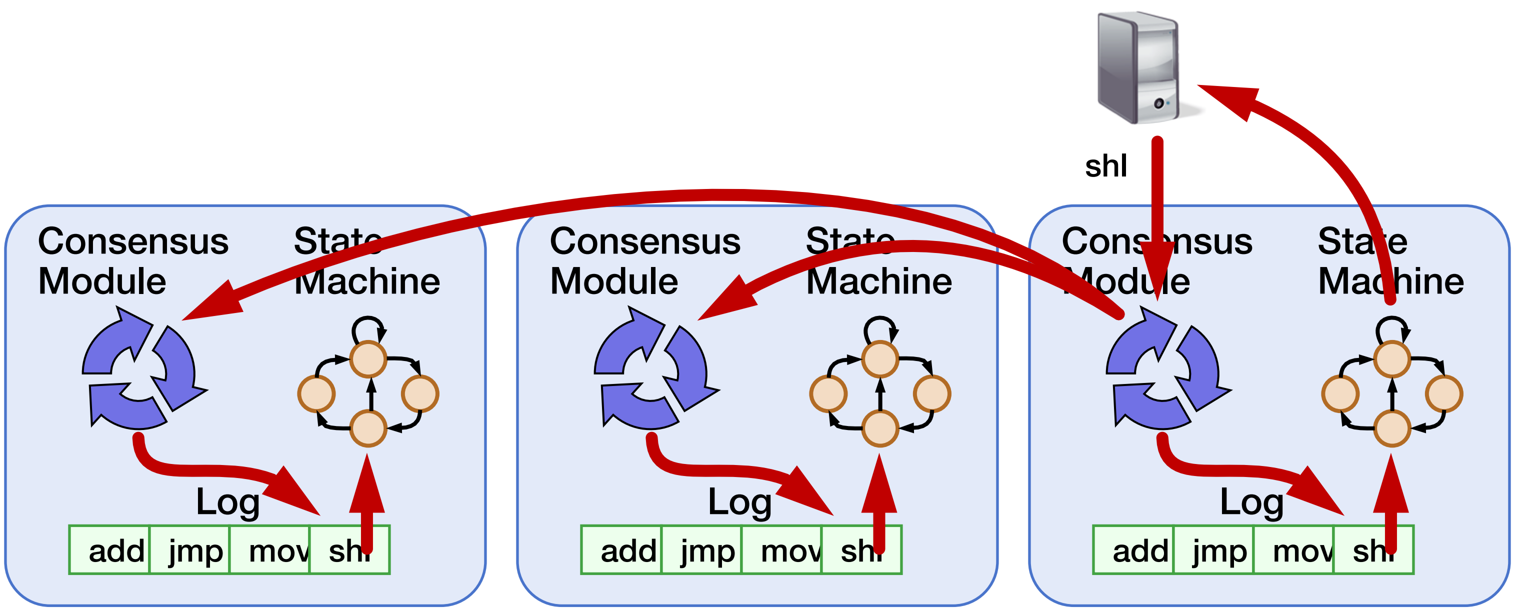
(b)

A: Violation: W(x)b happens after W(x)a

B: Correct. P2 doesn't read value of a before W

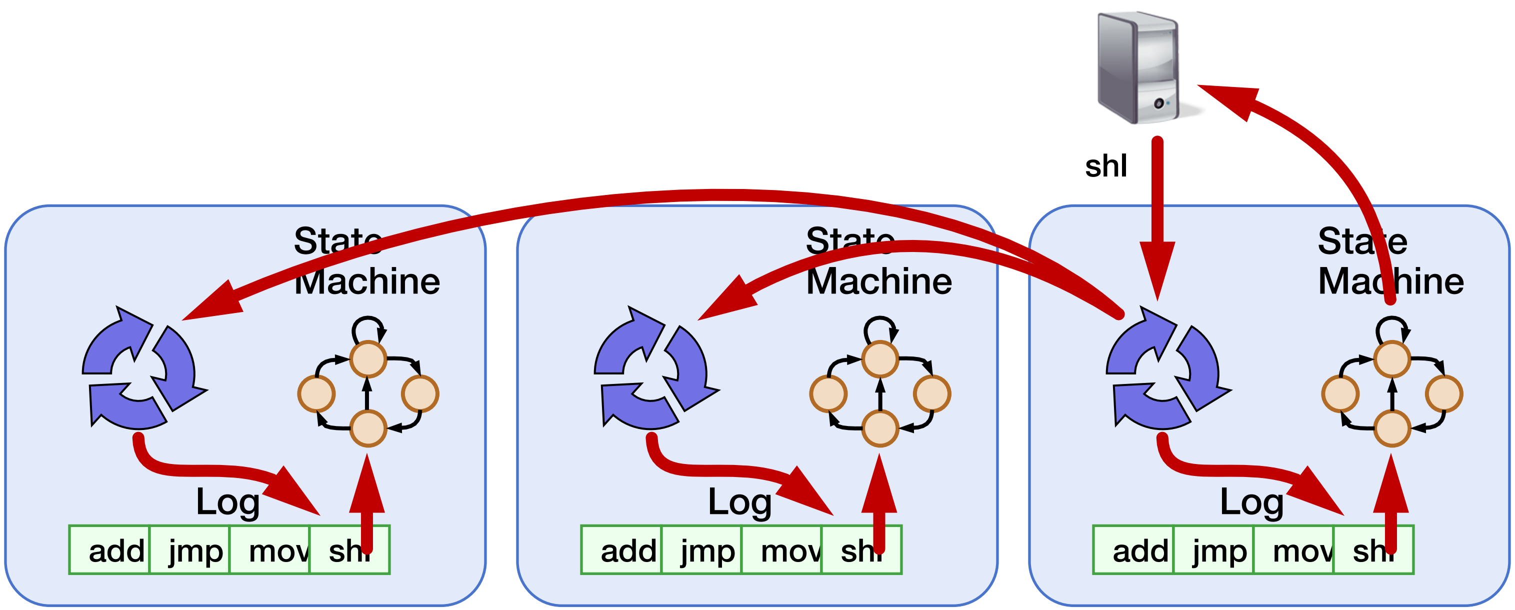
Causal consistency within replicated systems

Implications of laziness on consistency



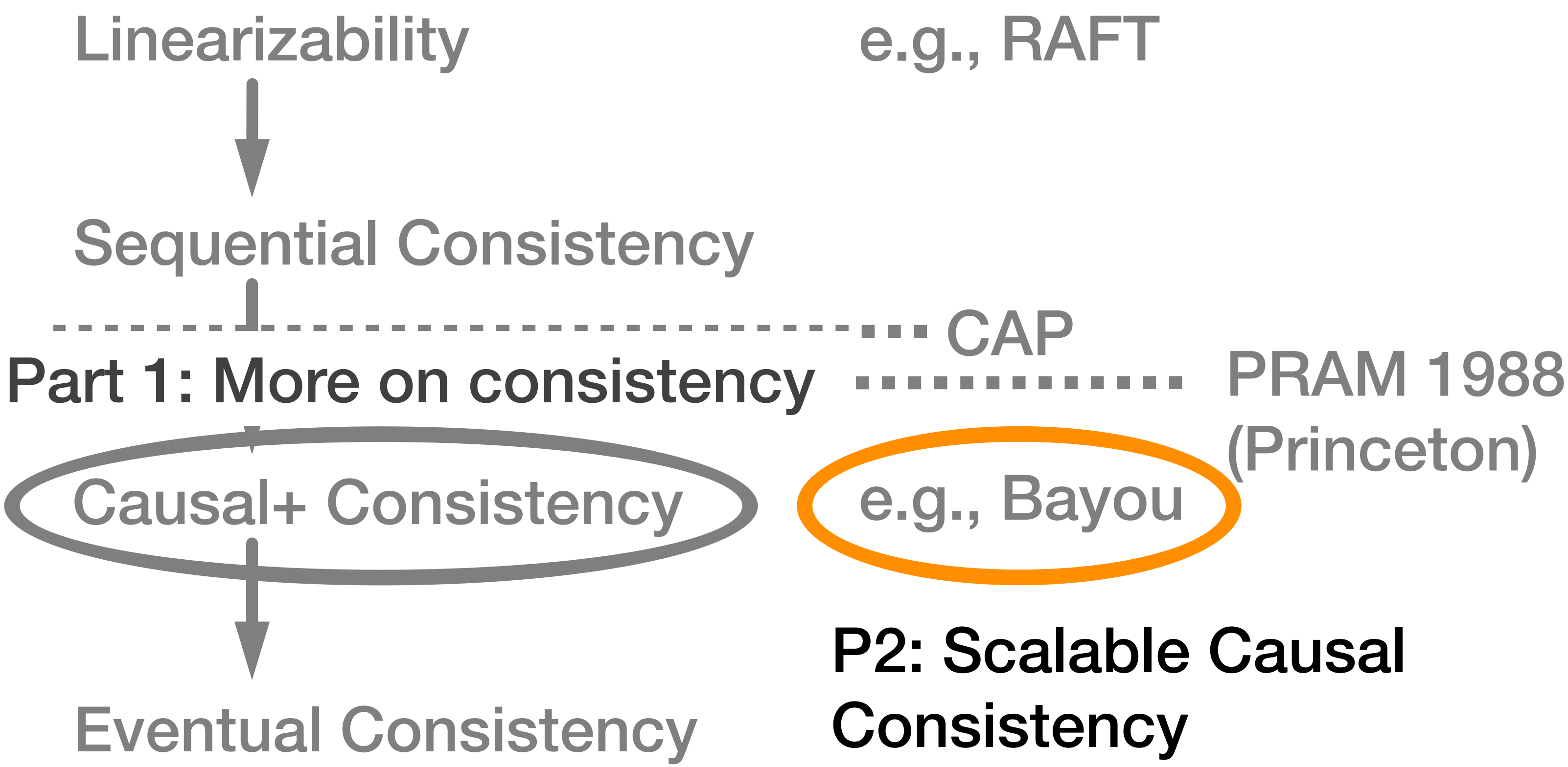
- Linearizability / sequential: Eager replication
- Trades off low-latency for consistency

Implications of laziness on consistency



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

Consistency Hierarchy



Consistency vs Scalability

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

System	Consistency	Scalable?
Dynamo	Eventual	
Bayou	Causal	
Paxos/RAFT	Linearizable	

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!

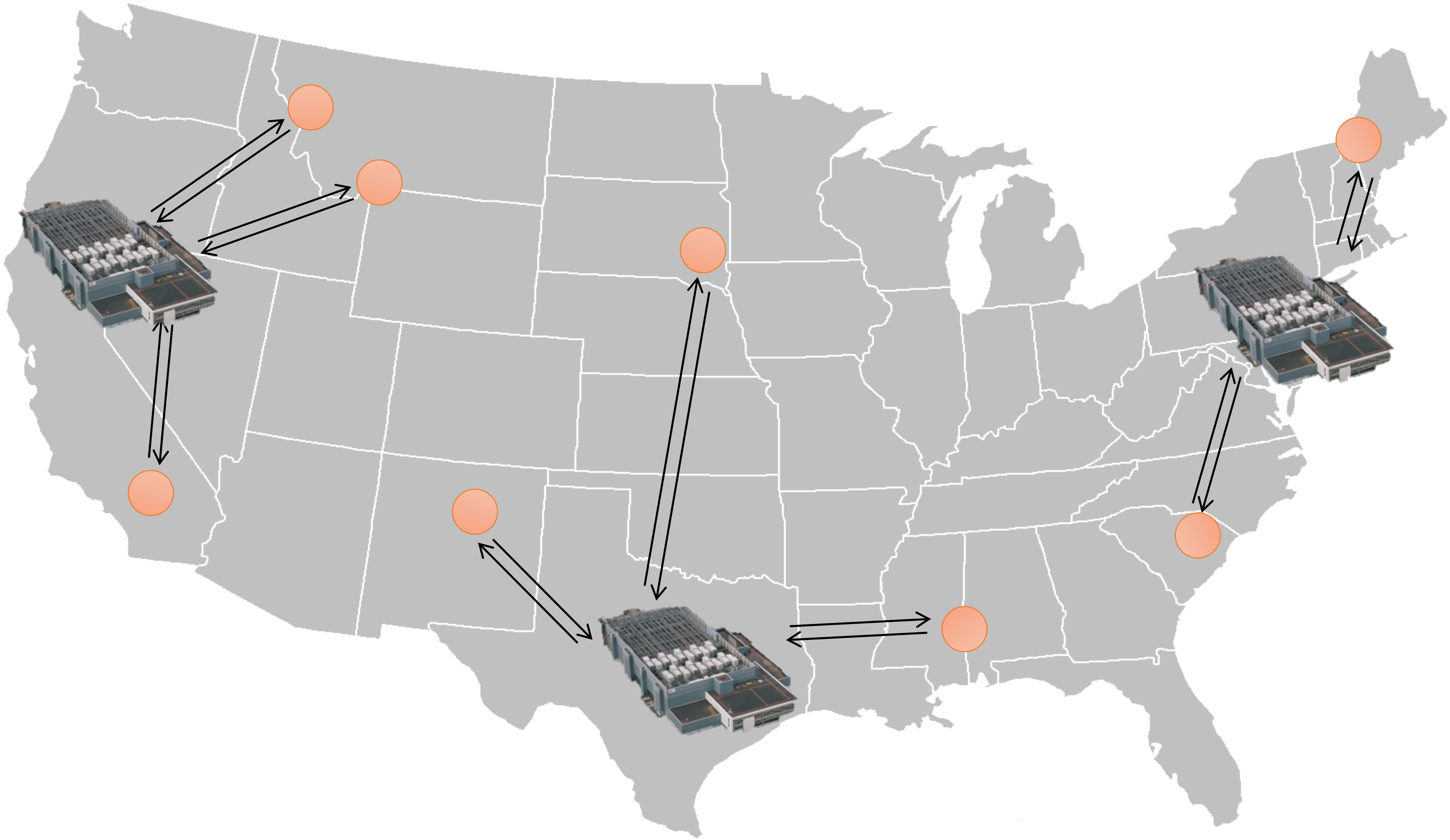
System	Consistency	Scalable?
Dynamo	Eventual	Yes
Bayou	Causal	No
COPS	Causal	Yes
Paxos/RAFT	Linearizable	No
		Next Time!

Don't Settle for Eventual: Scalable Causal Consistency for [Geo-Replicated] Storage with COPS

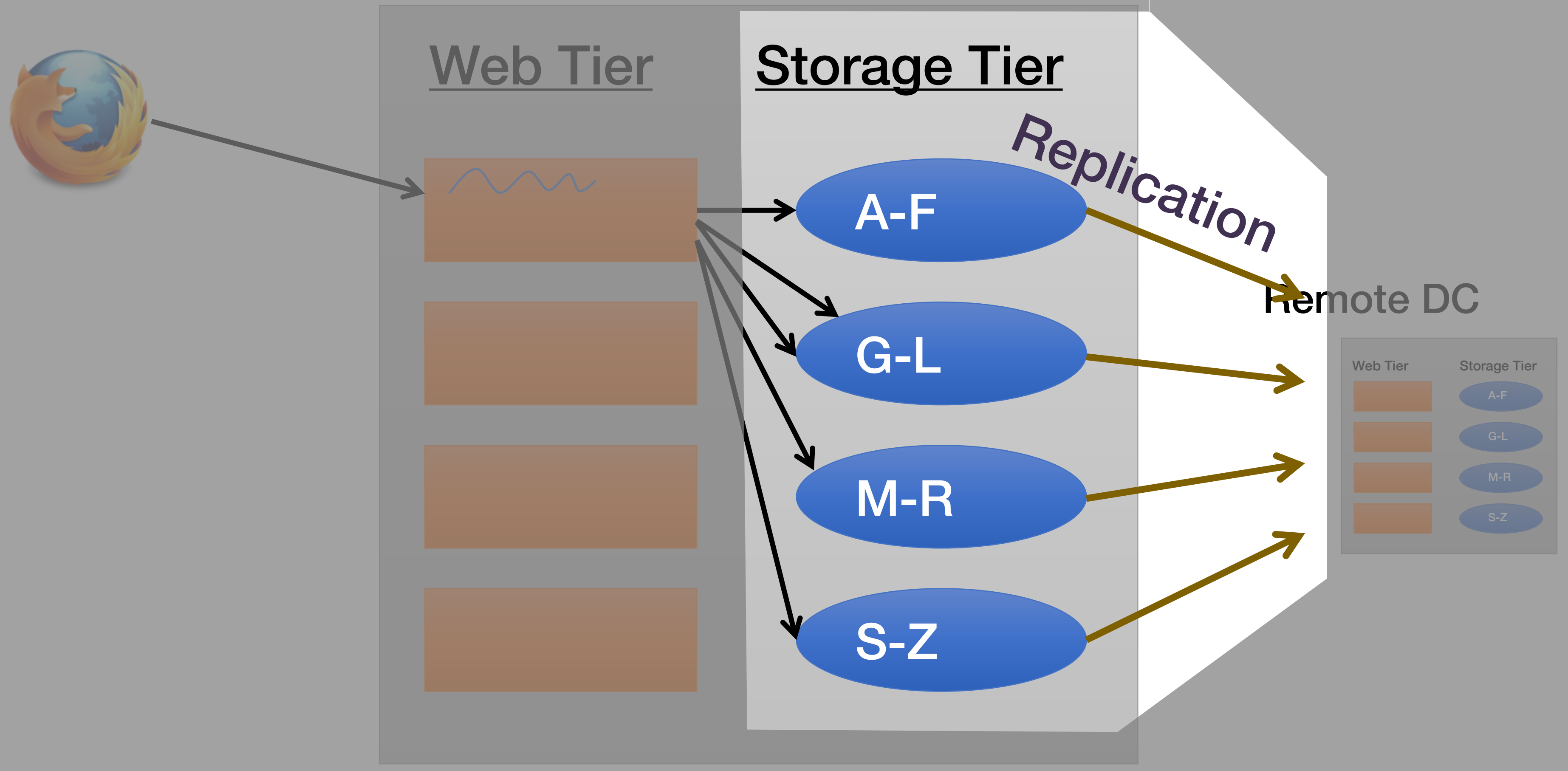
W. Lloyd, M. Freedman, M. Kaminsky, D. Andersen

SOSP 2011

Geo-Replicated Storage: Serve User Requests Quickly



Inside the Datacenter



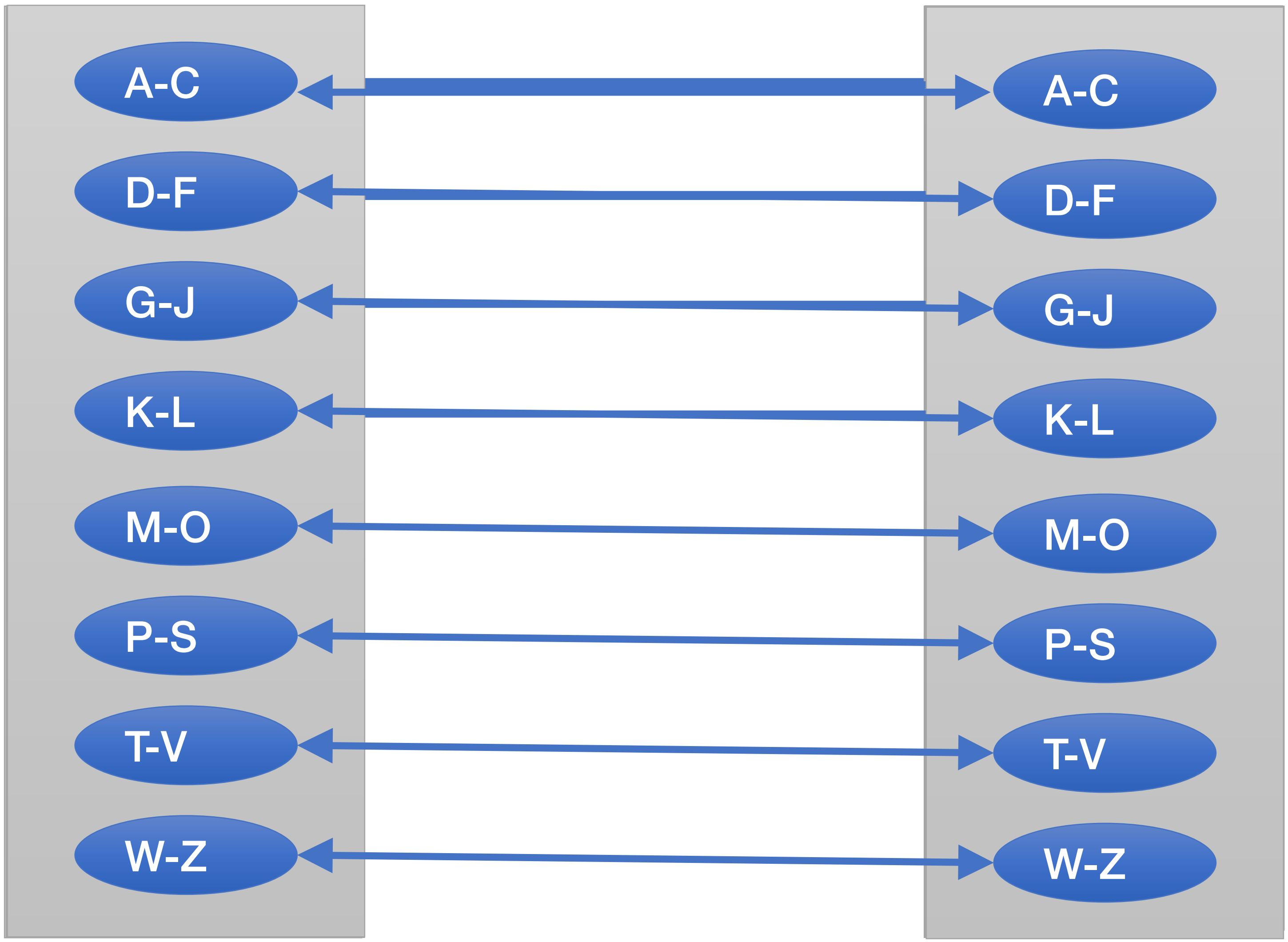
Trade-offs

- Consistency (Stronger)
- Partition Tolerance

vs.

- **Availability**
- **Low Latency**
- **Partition Tolerance**
- **Scalability**

Scalability through Sharding

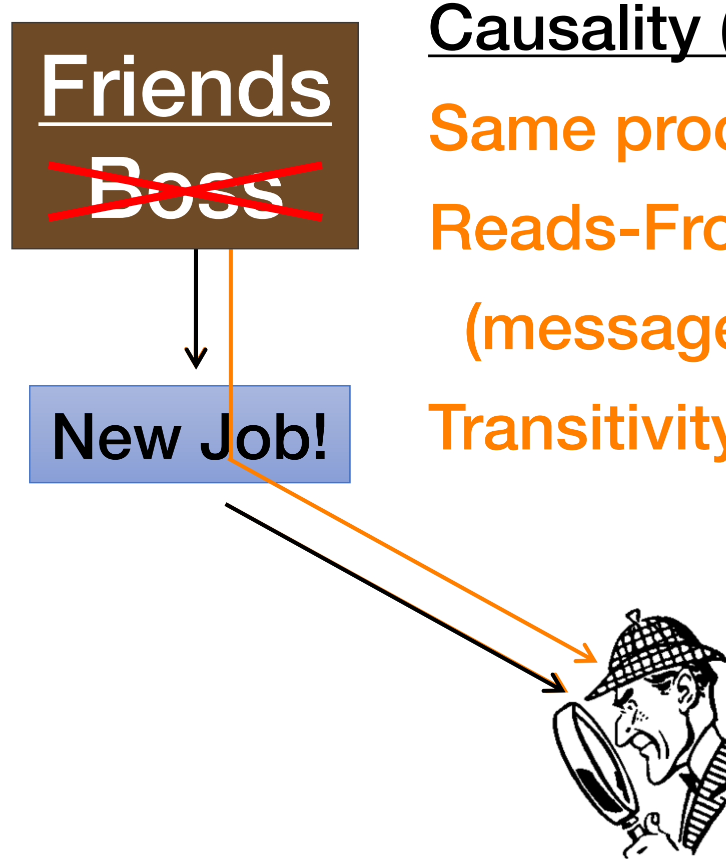


Causality By Example

Remove boss from friends group

Post to friends:
"Time for a new job!"

Friend reads post



<u>Causality ($\longrightarrow$)</u>

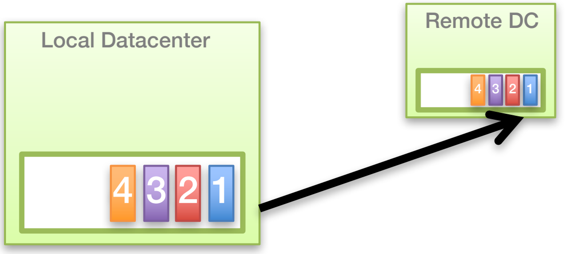
Same process

Reads-From
(message receipt)

Transitivity

Previous Causal Systems

- Bayou ‘94, TACT ‘00, PRACTI ‘06
  – Log-exchange based

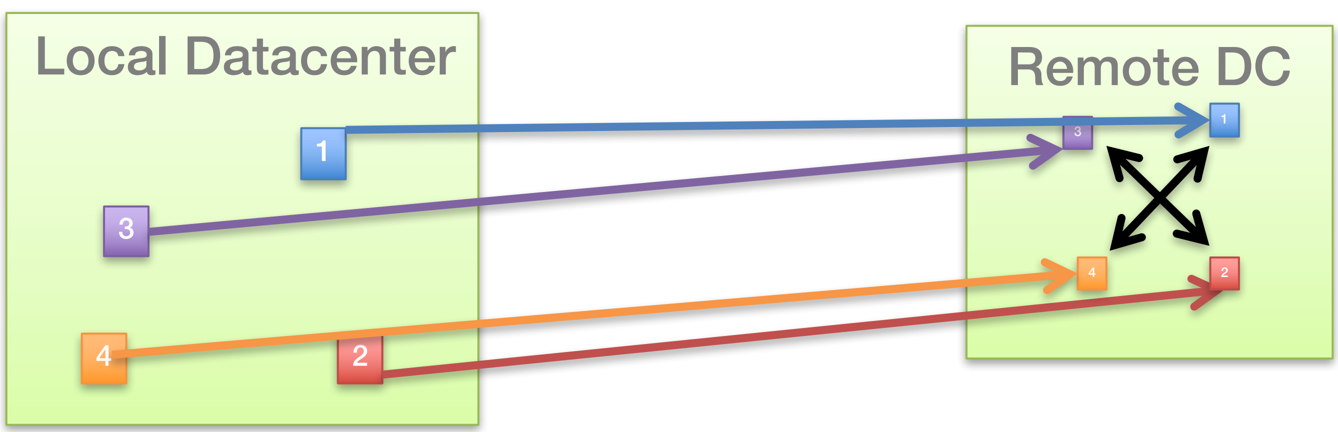


- Log is single serialization point
  √ **Implicitly** captures & enforces causal order
  X Limits scalability

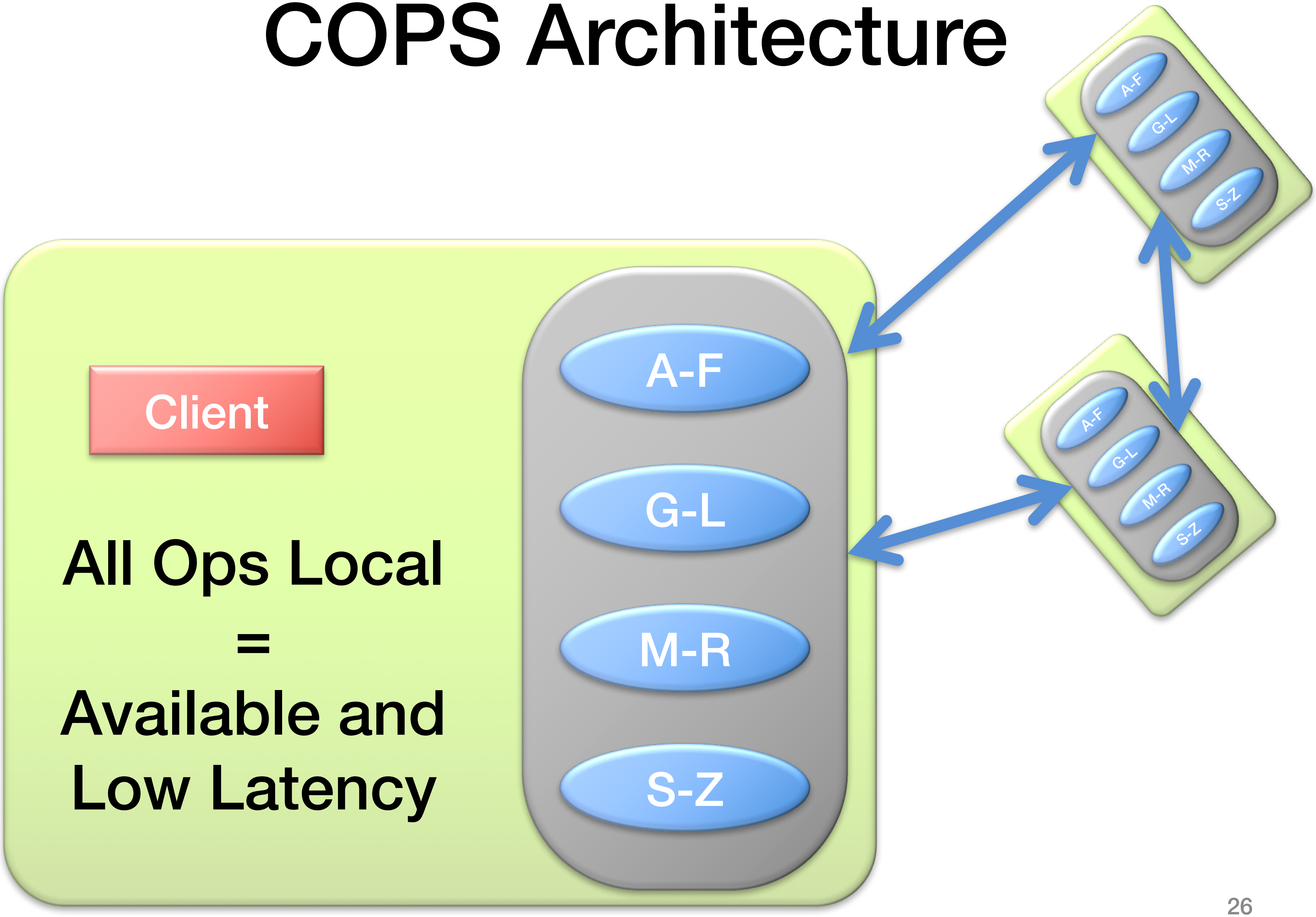
Scalability Key Idea

- Capture causality with explicit dependency metadata

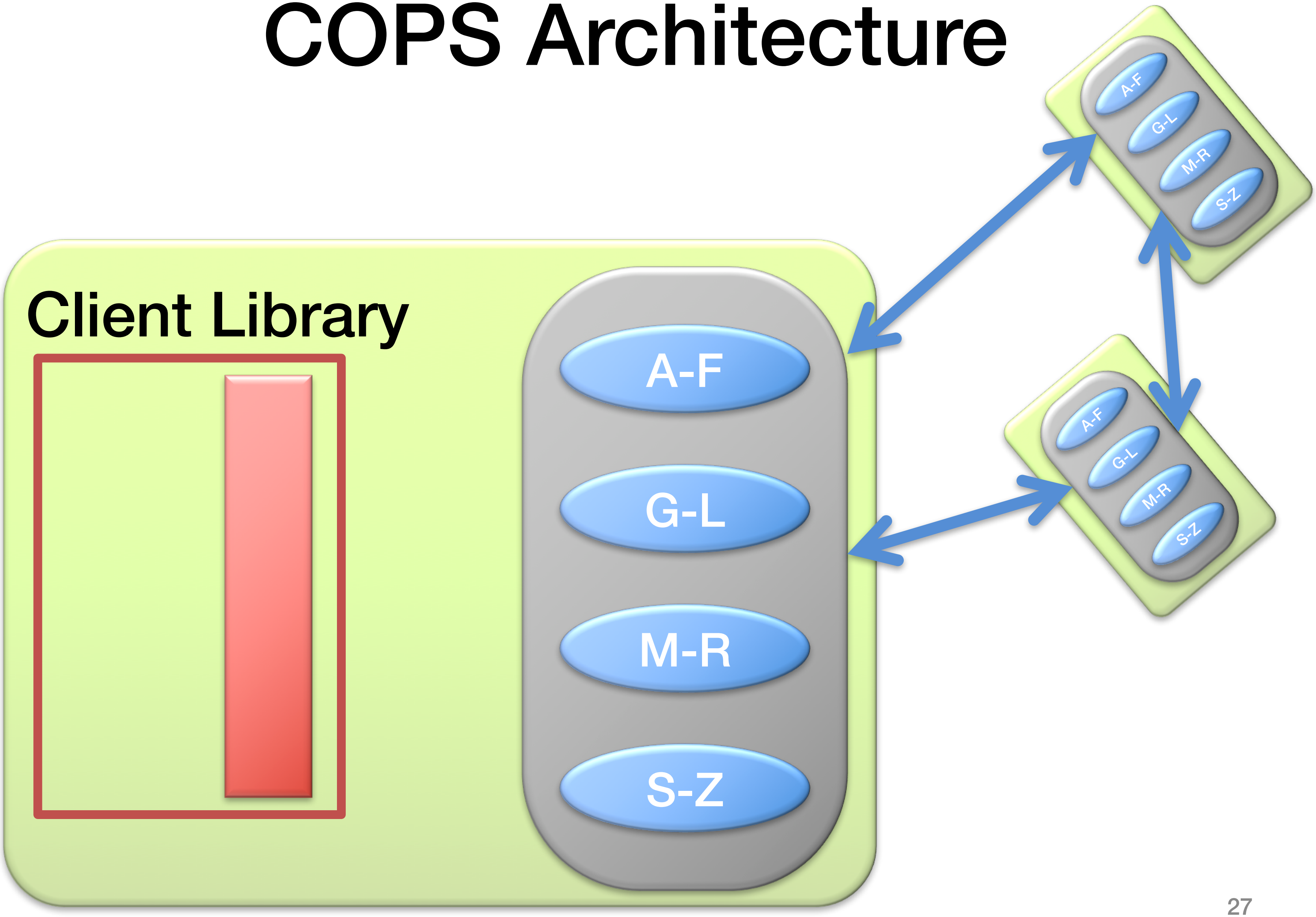
- Enforce with distributed verifications
  - Delay exposing replicated writes until all dependencies are satisfied in the datacenter



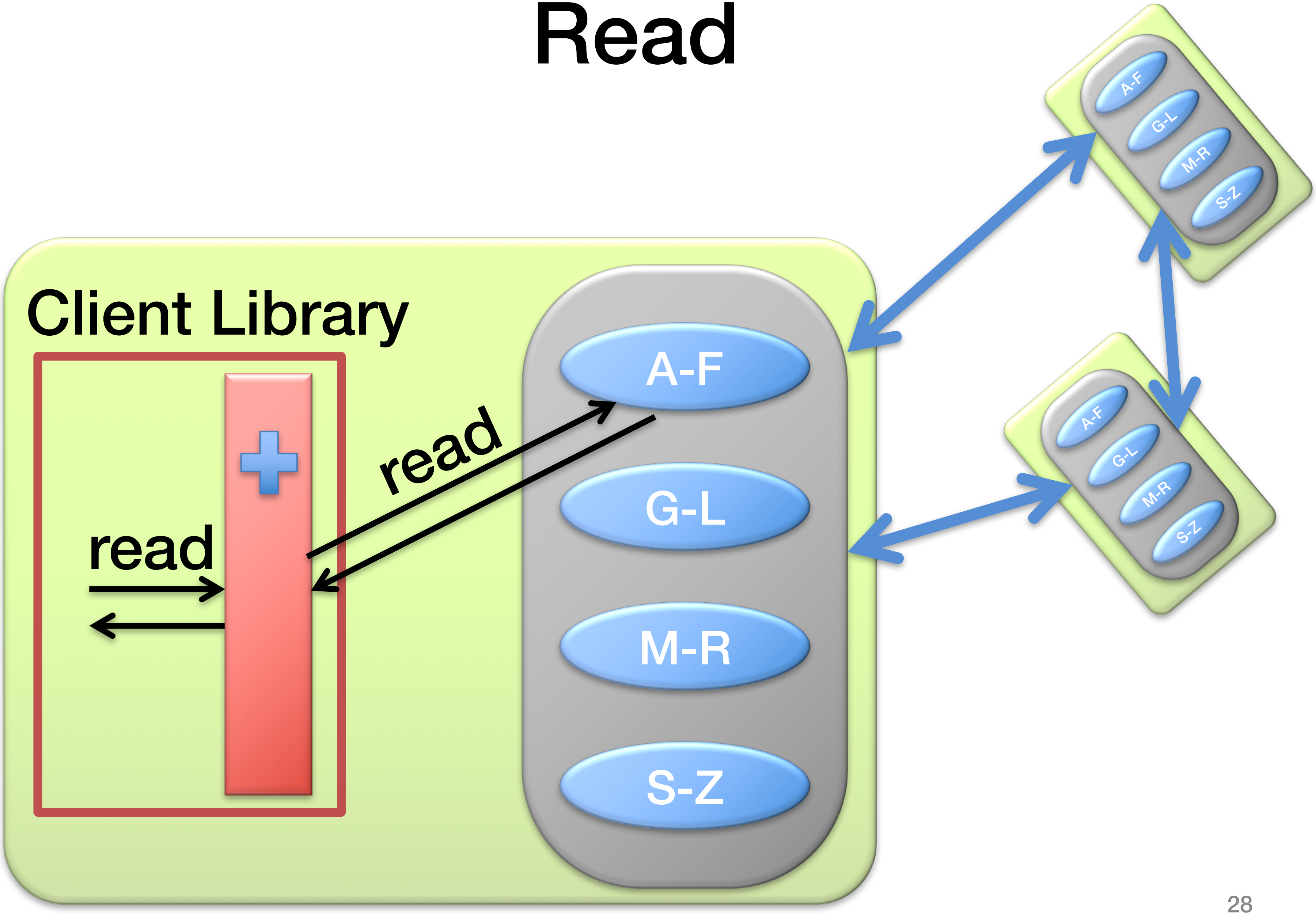
COPS Architecture



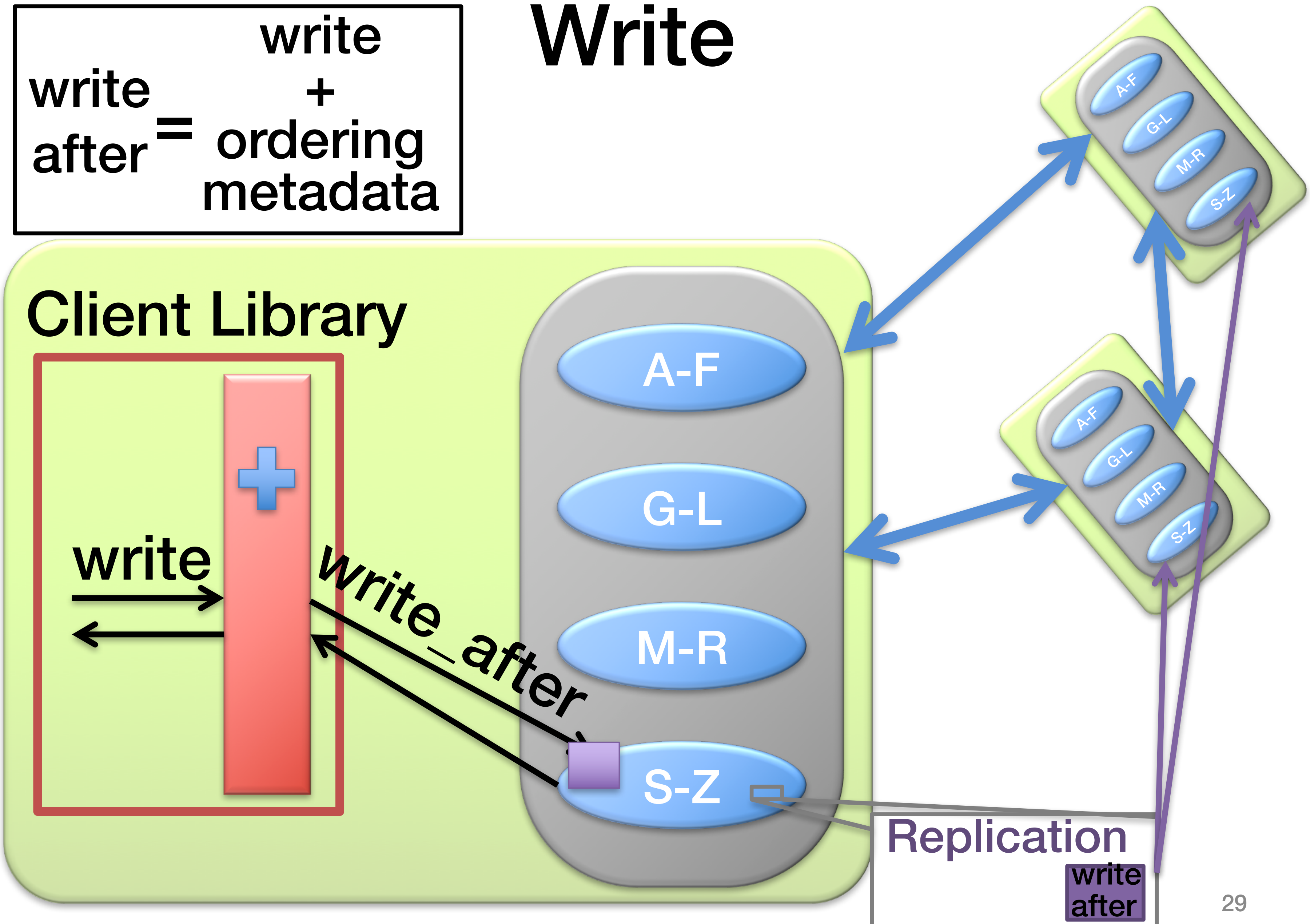
COPS Architecture



Read



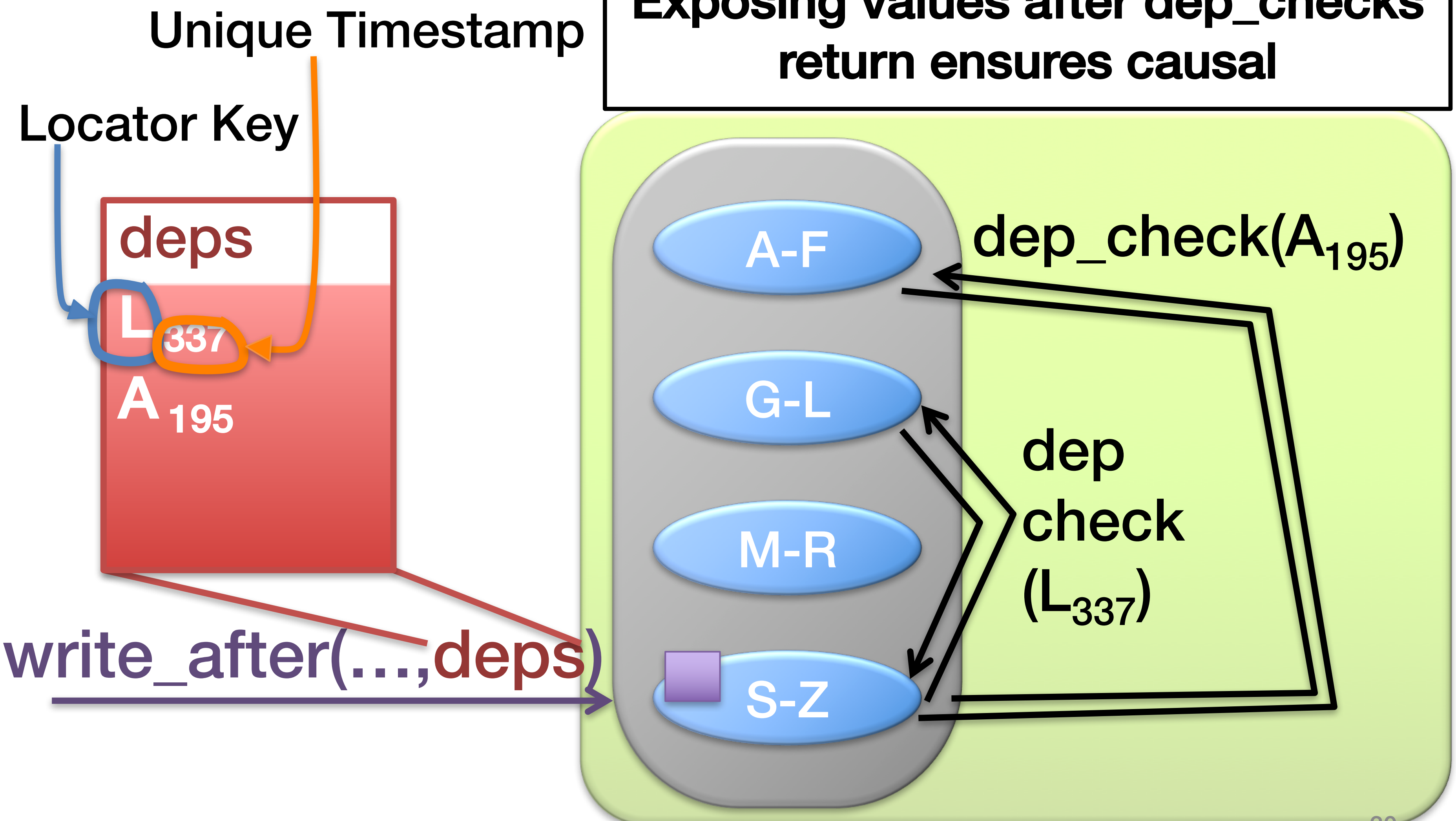
Write



write after = write + ordering metadata
Client Library
write
write_after
A-F
G-L
M-R
S-Z
Replication
write after

Replicated Write

Exposing values after dep_checks return ensures causal

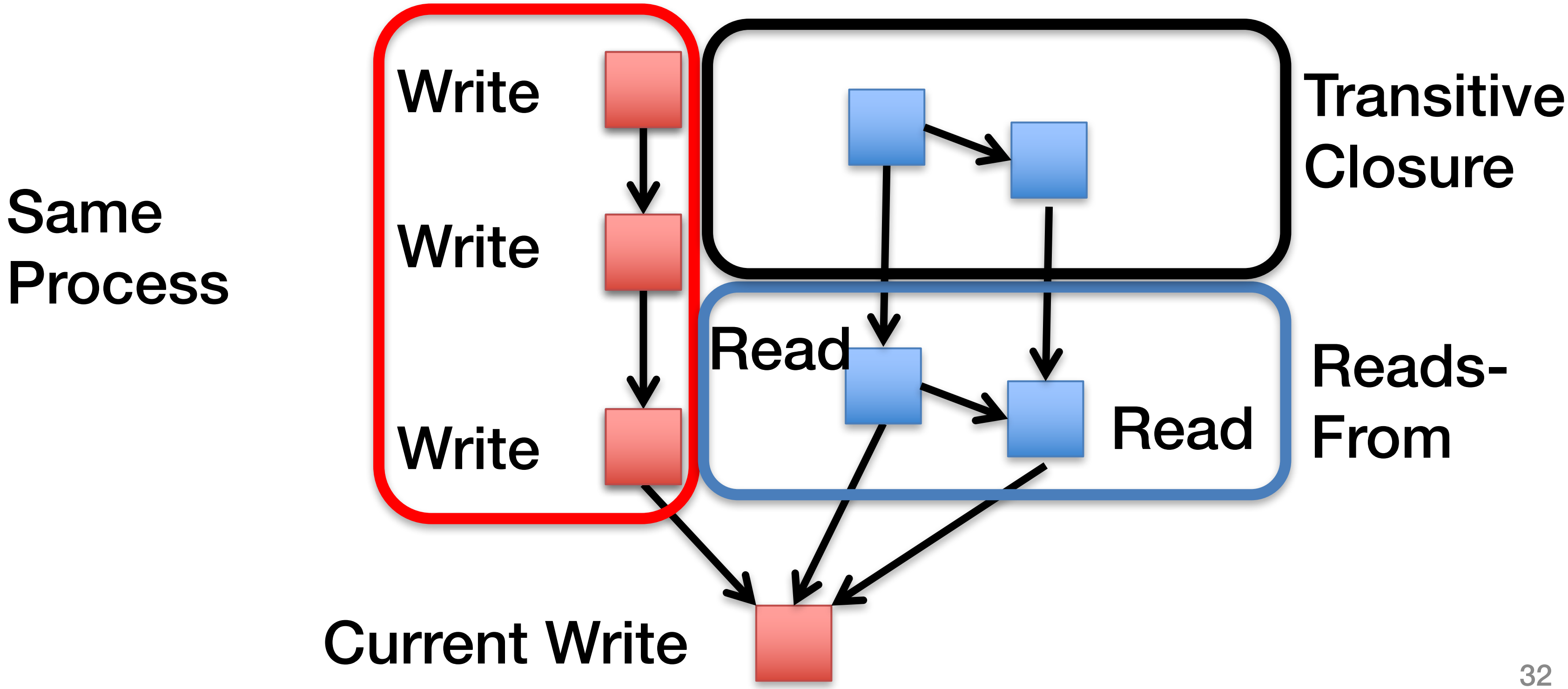


Basic Architecture Summary

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

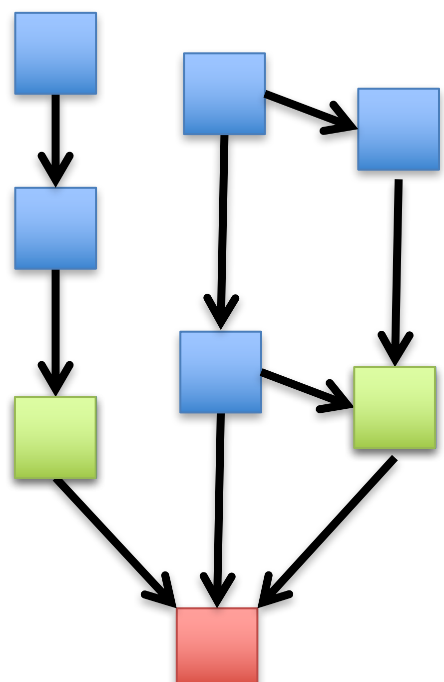
Challenge: Many Dependencies

- Dependencies grow with client lifetime



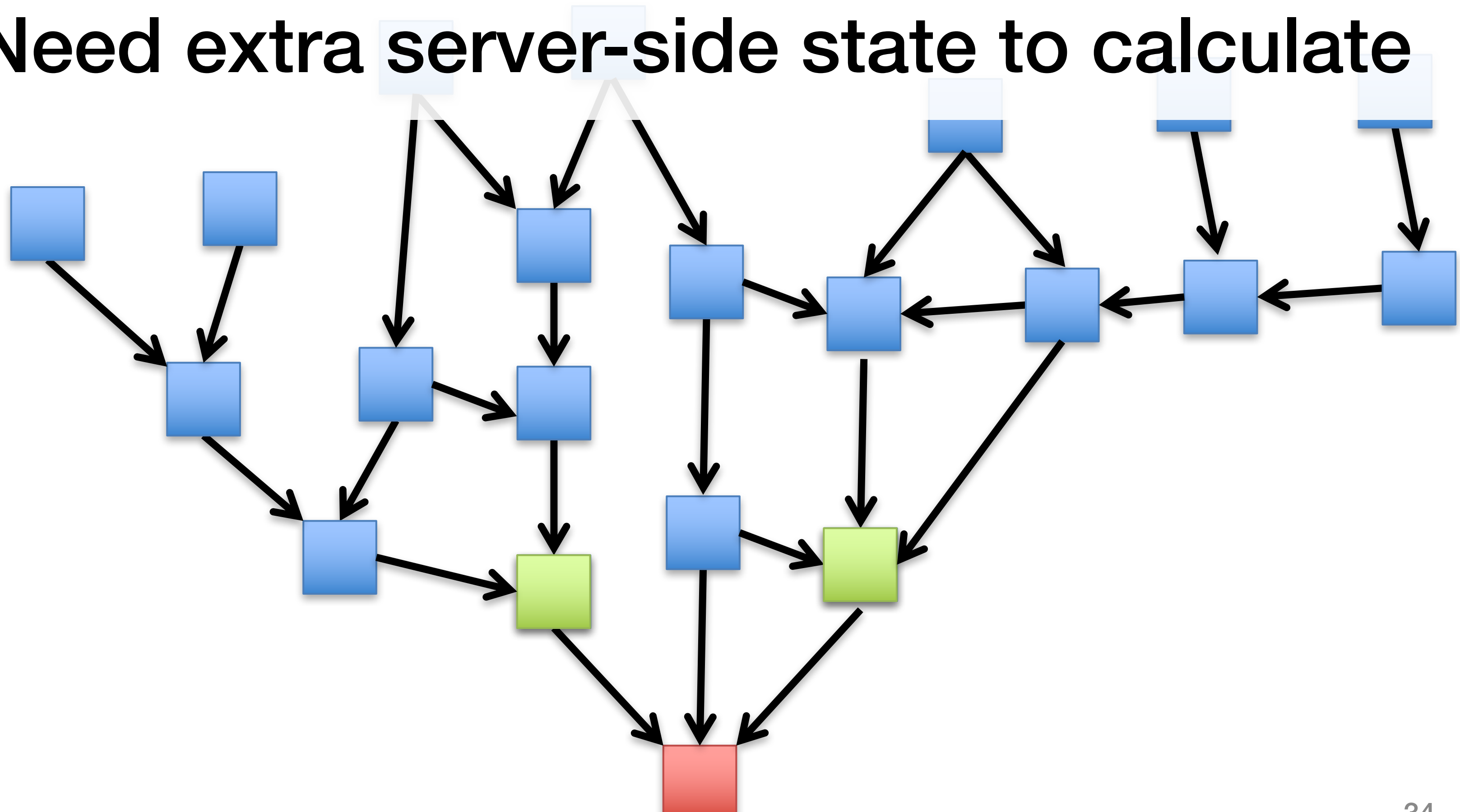
Nearest Dependencies

- Transitively capture ordering constraints



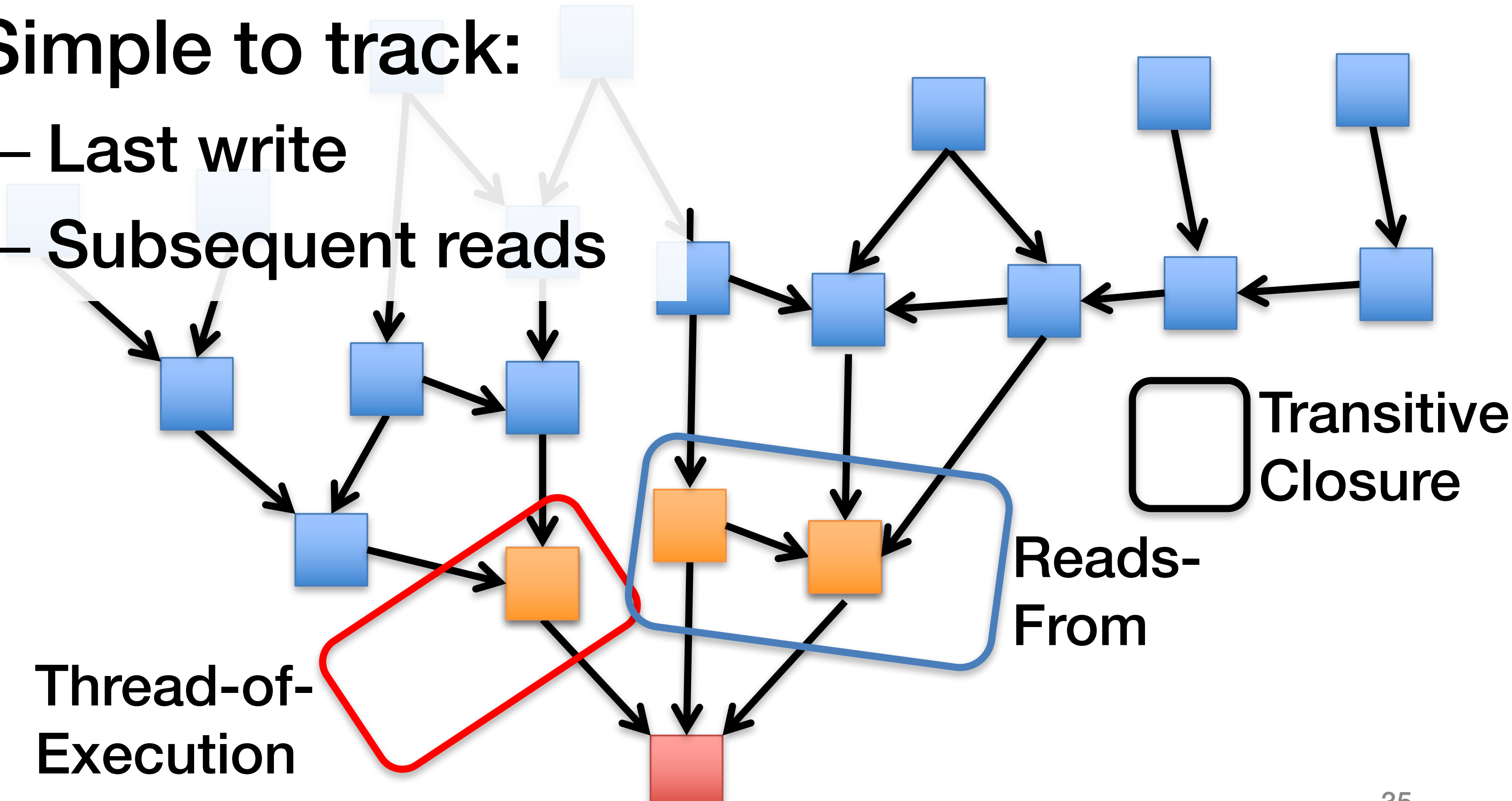
Nearest Dependencies

- Transitively capture ordering constraints
- Need extra server-side state to calculate



One-Hop Dependencies

- Small superset of nearest dependencies
- Simple to track:
  - Last write
  - Subsequent reads



One-Hop Dependencies

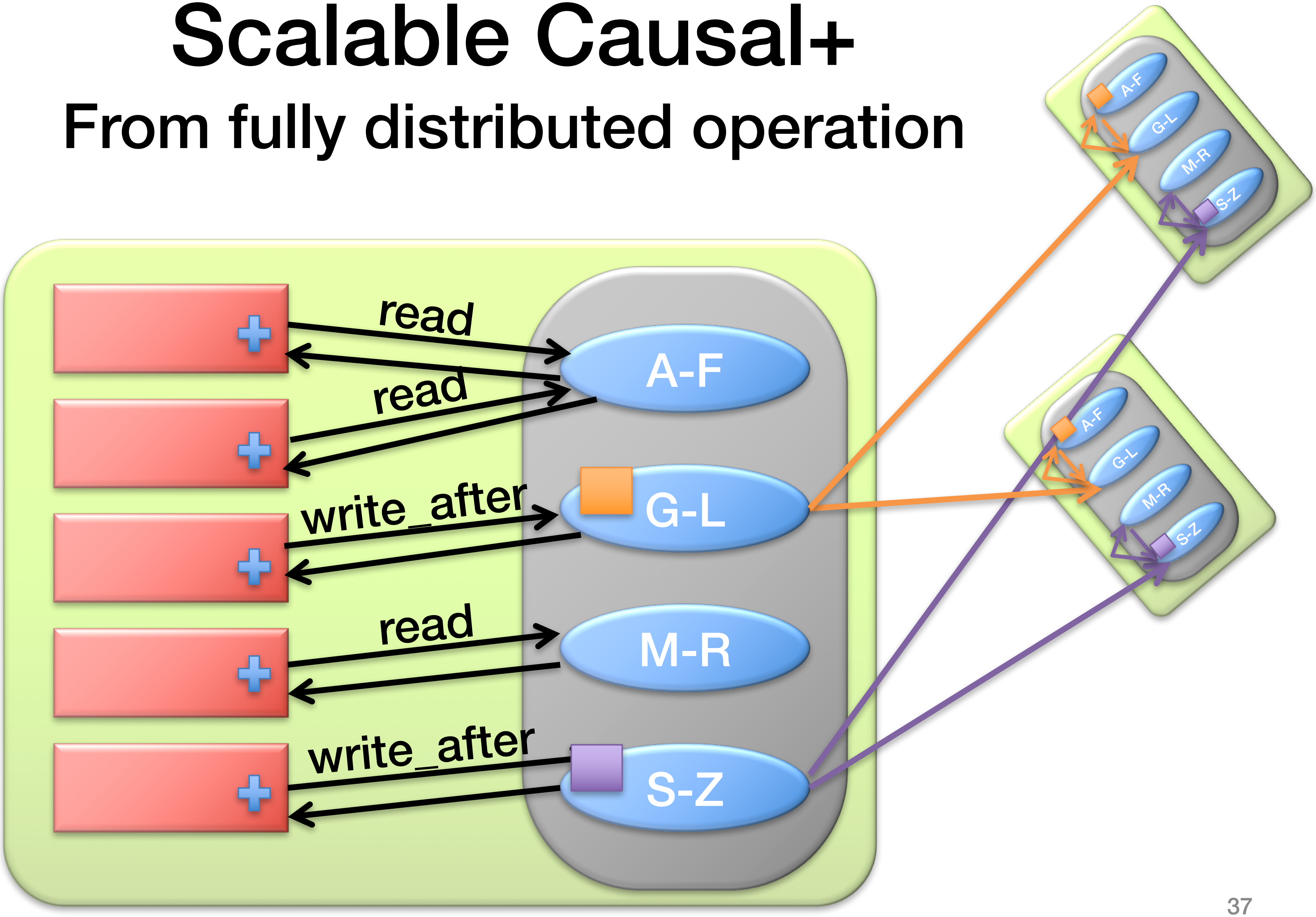
- Checking them suffices for causality
  - Competitive to eventually-consistent system

- Never store dependencies on the server
  - Transitive Closure

- Simplifies client-side dep tracking
  - Clear on every write

Scalable Causal+

From fully distributed operation

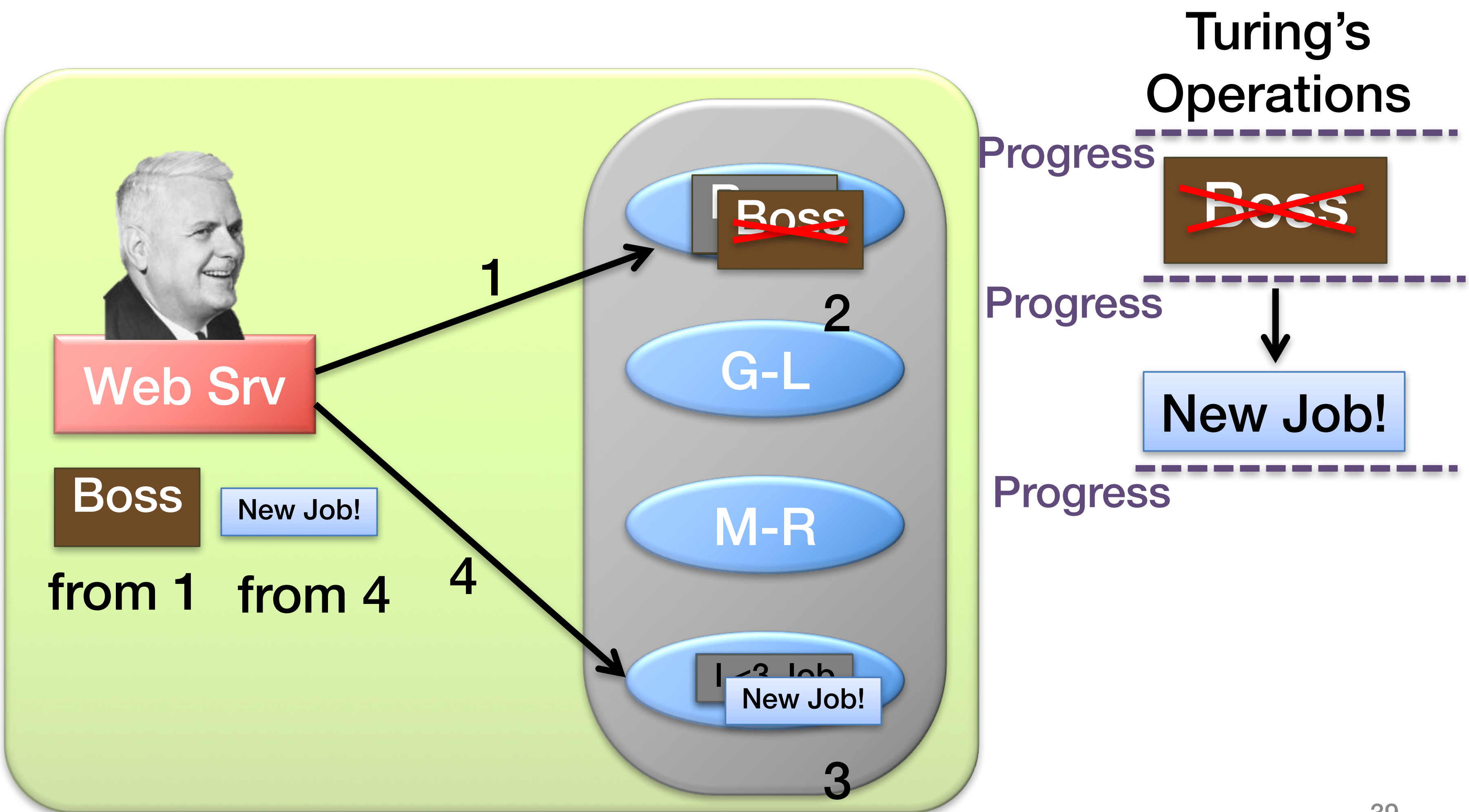


Scalability

- 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…

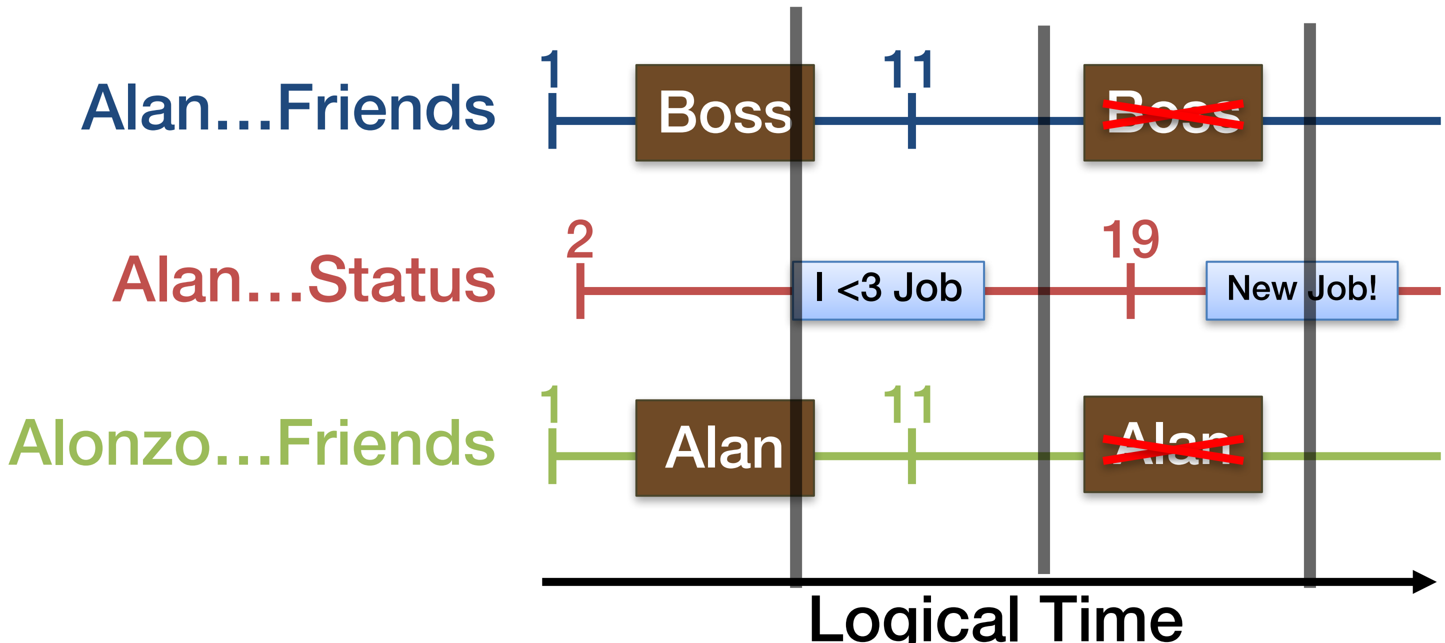
Reads Aren't Enough

Asynchronous requests + distributed data = ??



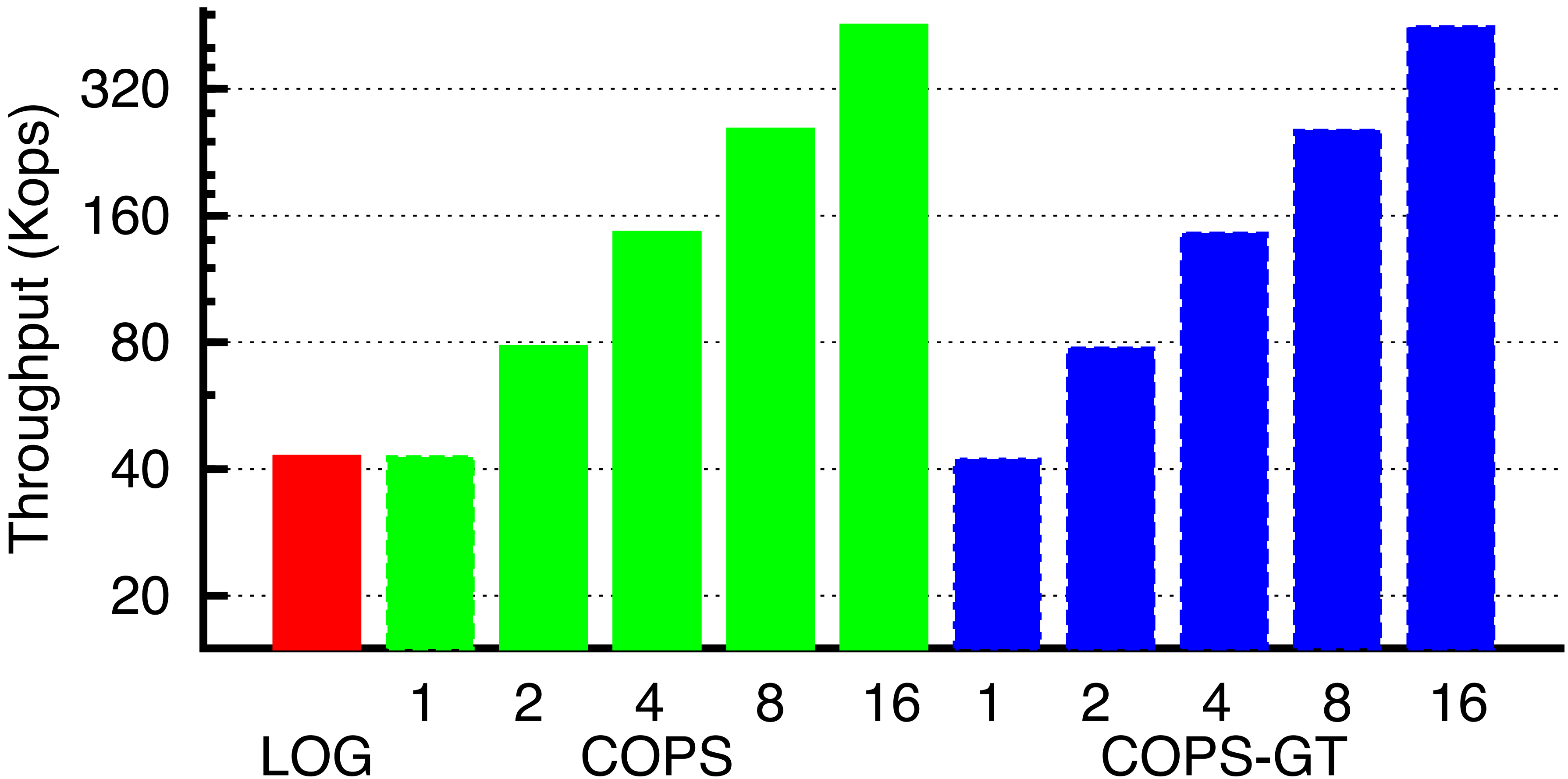
Read-Only Transactions

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



More on transactions next time!

COPS Scaling Evaluation



More servers => More operations/sec

COPS

- 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