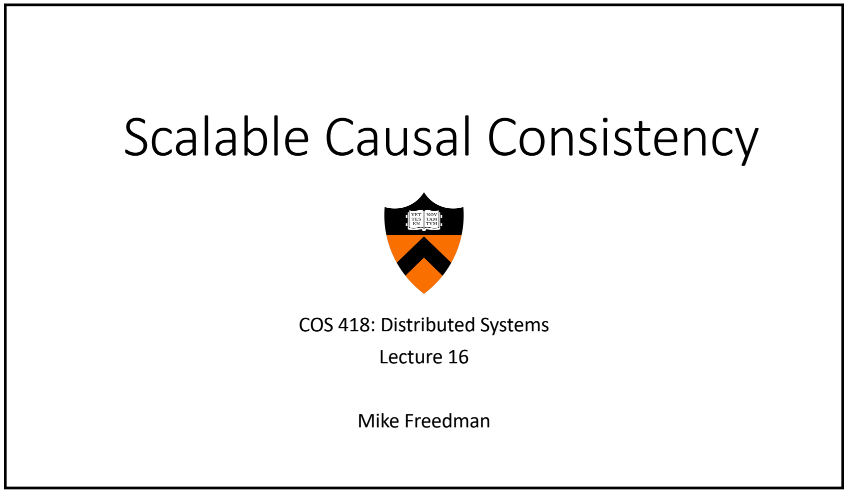
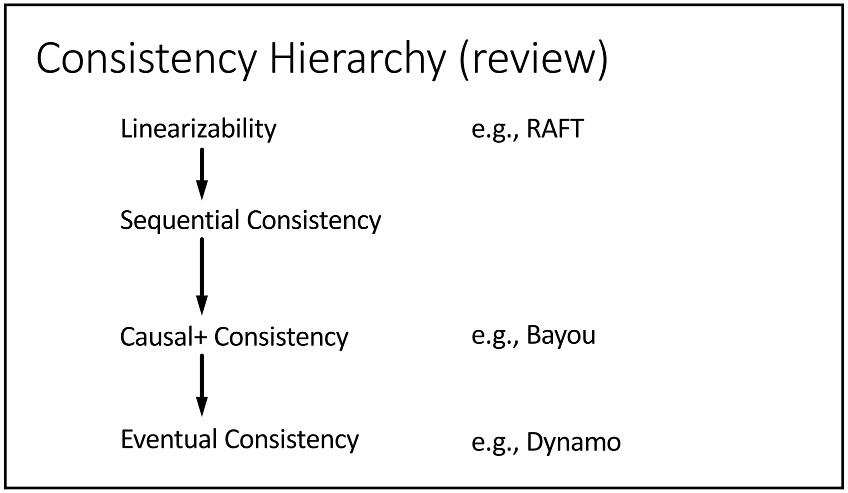


Scalable Causal Consistency

COS 418: Distributed Systems
Lecture 16

Mike Freedman

1



Consistency Hierarchy (review)

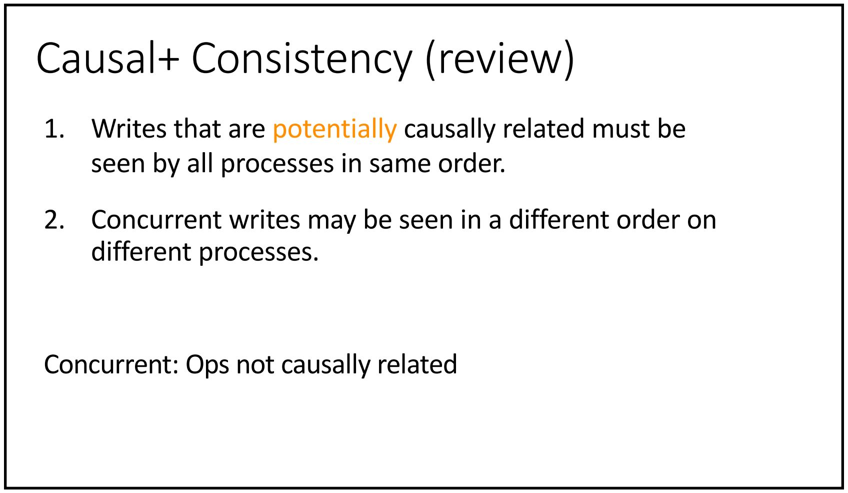
Linearizability — e.g., RAFT

Sequential Consistency

Causal+ Consistency — e.g., Bayou

Eventual Consistency — e.g., Dynamo

2

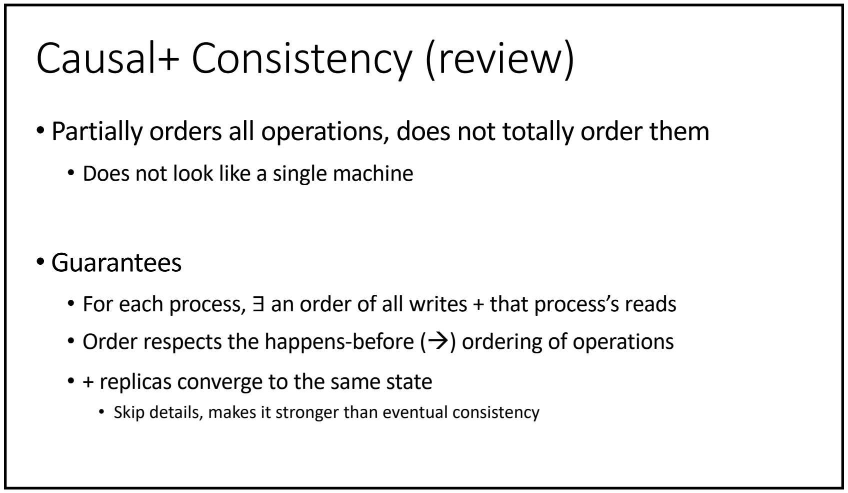


Causal+ Consistency (review)

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

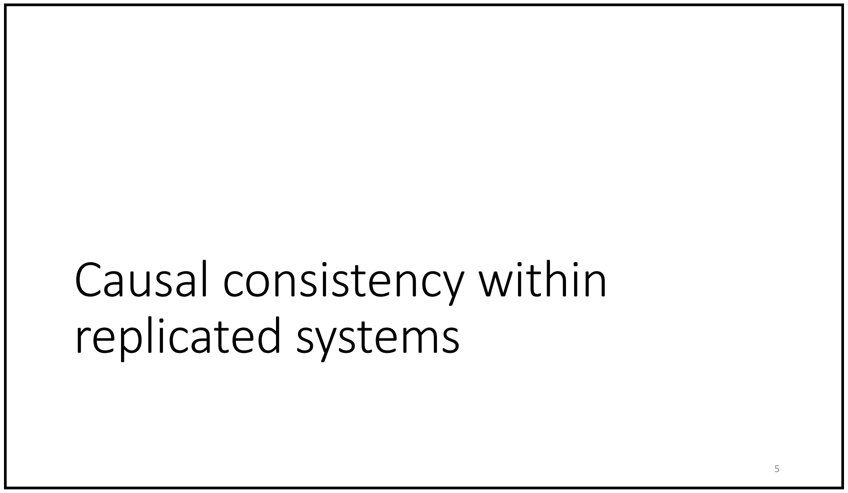
3



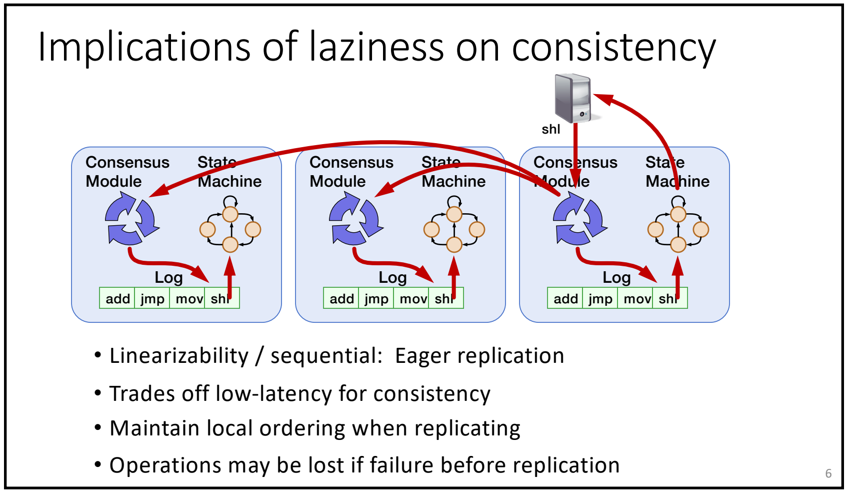
Causal+ Consistency (review)

- 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
    - Skip details, makes it stronger than eventual consistency

4

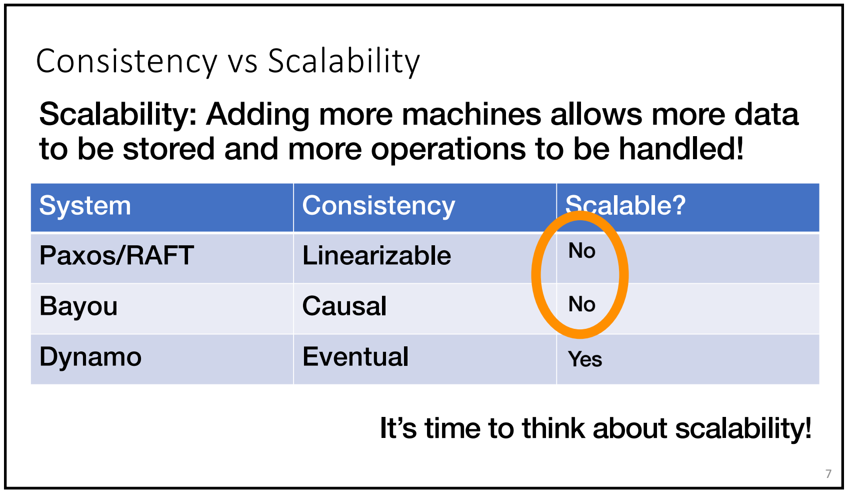


Causal consistency within replicated systems



Implications of laziness on consistency

- Linearizability / sequential: Eager replication
- Trades off low-latency for consistency
- Maintain local ordering when replicating
- Operations may be lost if failure before replication

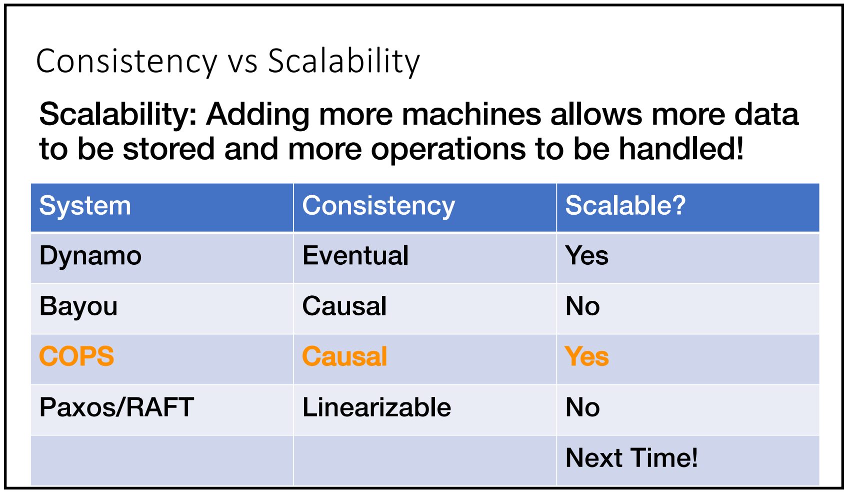


Consistency vs Scalability

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

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

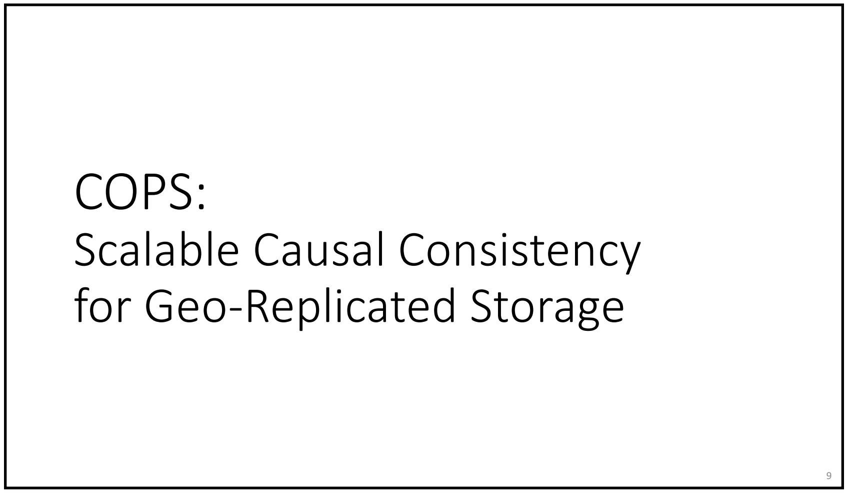
It's time to think about scalability!



Consistency vs Scalability

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

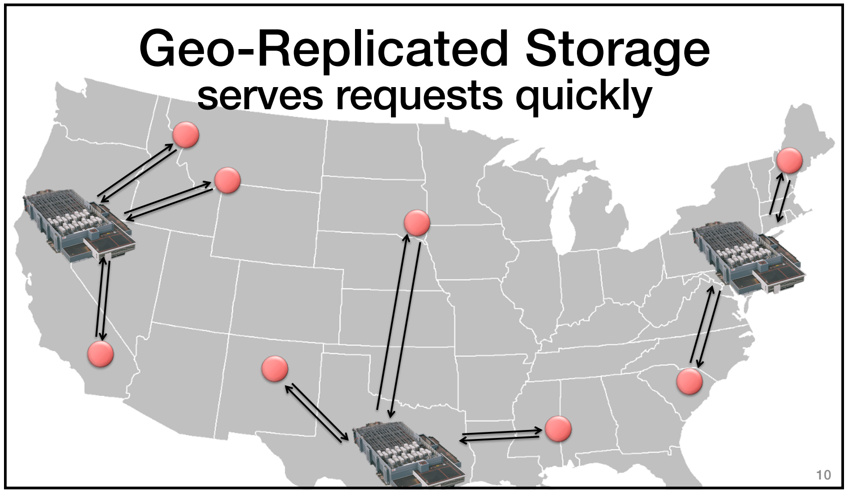
System	Consistency	Scalable?
Dynamo	Eventual	Yes
Bayou	Causal	No
COPS	Causal	Yes
Paxos/RAFT	Linearizable	No
		Next Time!



COPS:

Scalable Causal Consistency
for Geo-Replicated Storage

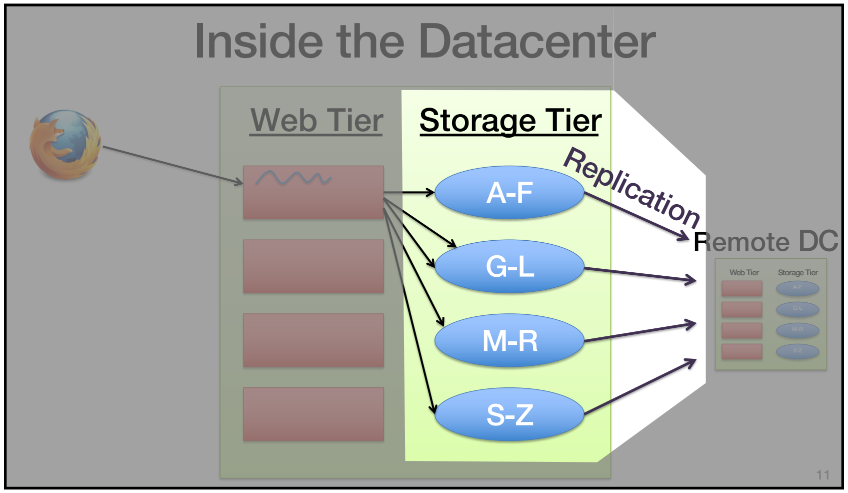
9



Geo-Replicated Storage

serves requests quickly

10



Inside the Datacenter

Web Tier

Storage Tier

A-F

G-L

M-R

S-Z

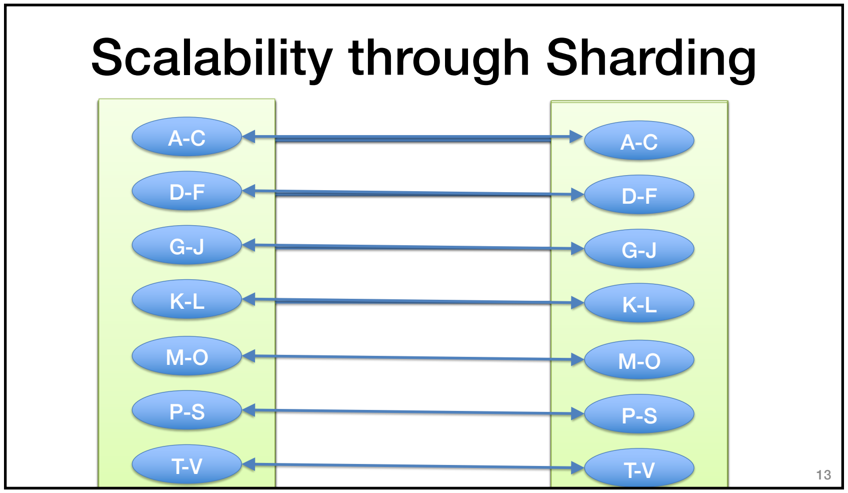
Replication

Remote DC

Web Tier

Storage Tier

11



Scalability through Sharding

A-C ↔ A-C

D-F ↔ D-F

G-J ↔ G-J

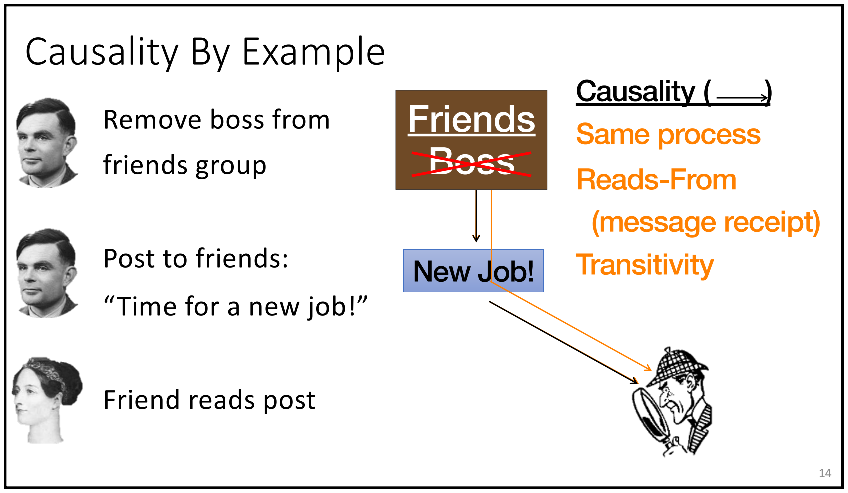
K-L ↔ K-L

M-O ↔ M-O

P-S ↔ P-S

T-V ↔ T-V

13



Causality By Example

Remove boss from friends group

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

Friend reads post

Friends
~~Boss~~

New Job!

Causality (⟶)

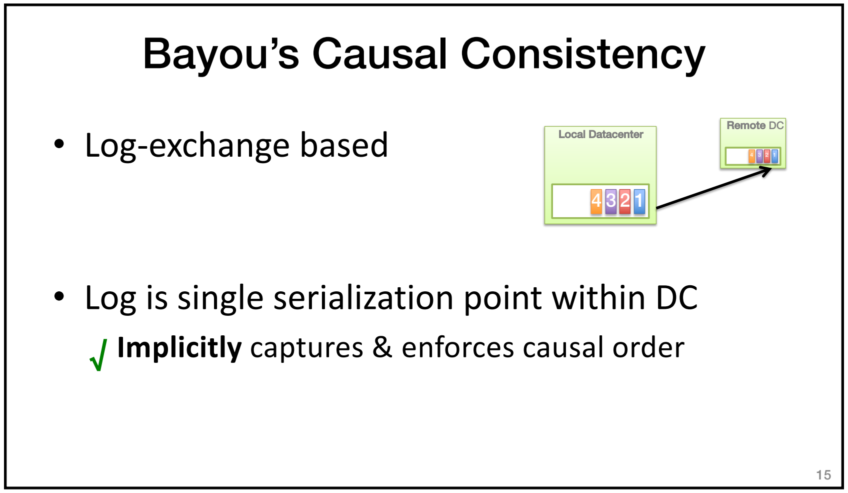
Same process

Reads-From
(message receipt)

Transitivity

14

14



Bayou's Causal Consistency

- Log-exchange based

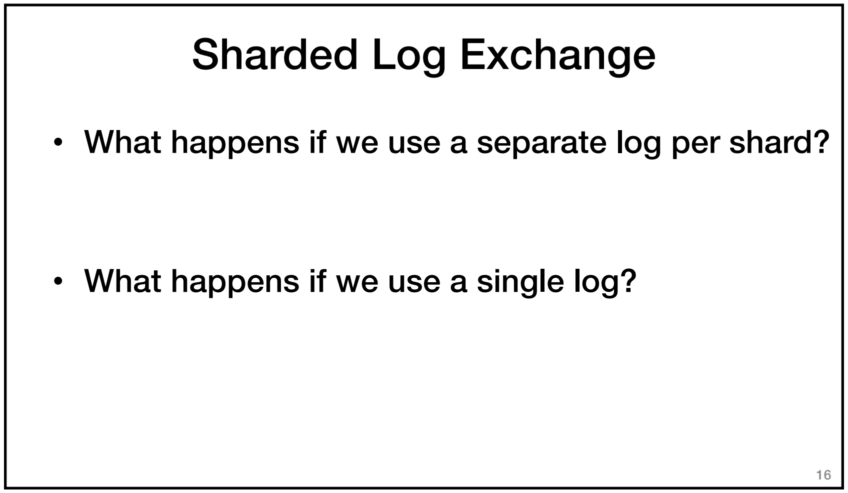
Local Datacenter
4 3 2 1
Remote DC

- Log is single serialization point within DC

√ **Implicitly** captures & enforces causal order

15

15



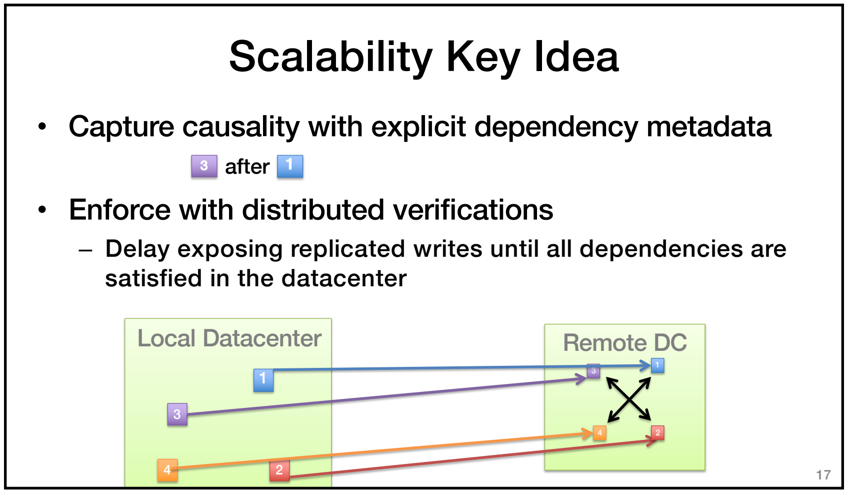
Sharded Log Exchange

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

- What happens if we use a single log?

16

16



Scalability Key Idea

- Capture causality with explicit dependency metadata

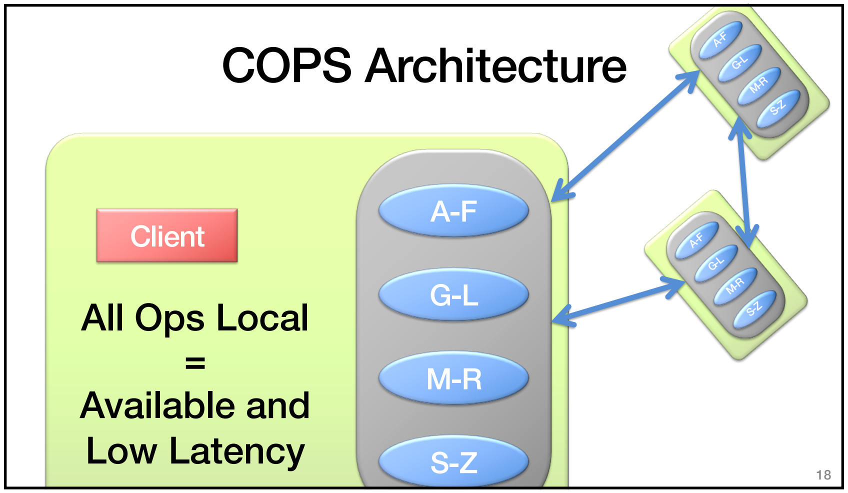
  3 after 1

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

Local Datacenter
1
3
4
2
Remote DC
3
1
4
2

17

17

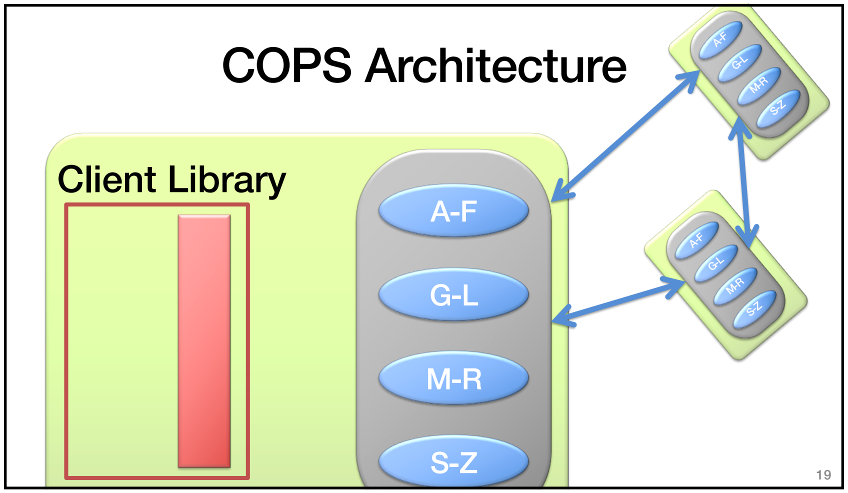


COPS Architecture

Client

All Ops Local
=
Available and
Low Latency

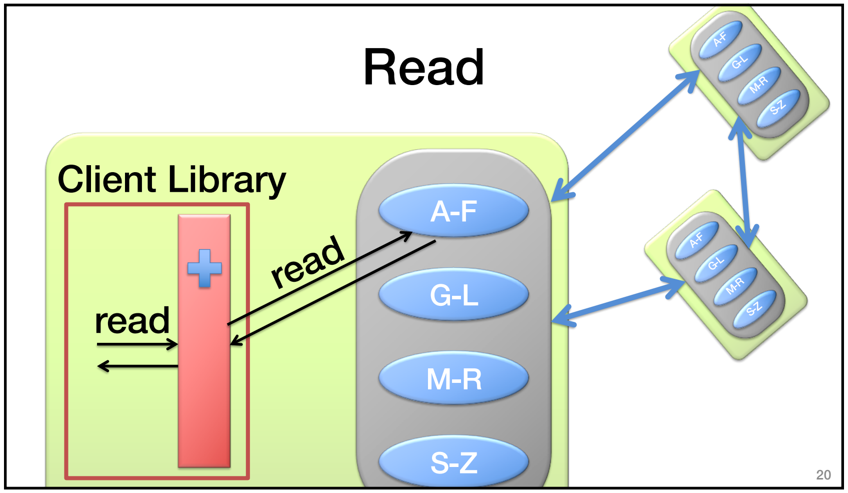
18



COPS Architecture

Client Library

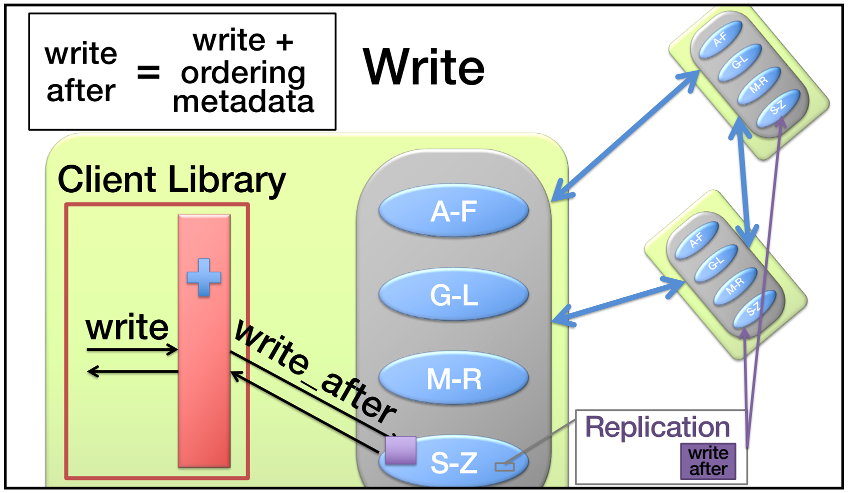
19



Read

Client Library

20

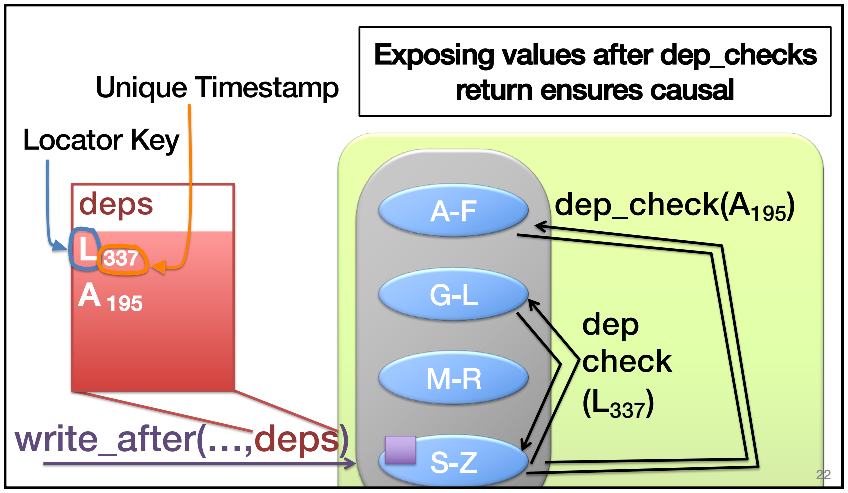


write after = write + ordering metadata

Write

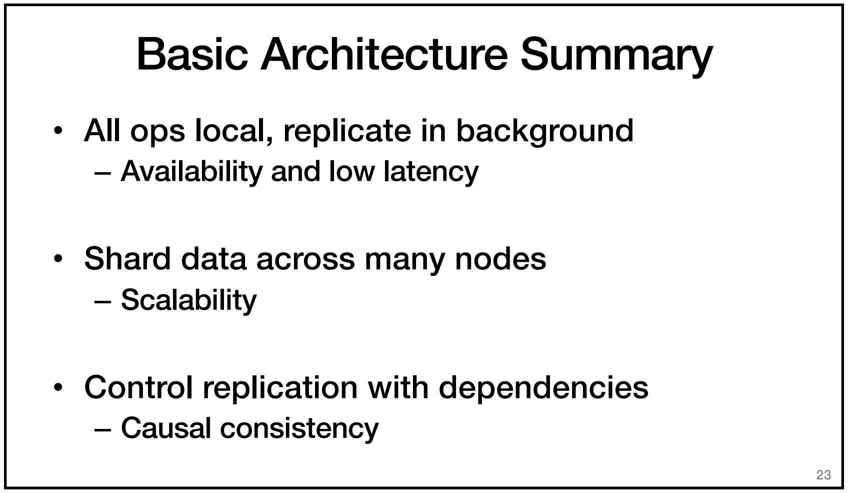
Client Library

21



Exposing values after dep_checks return ensures causal

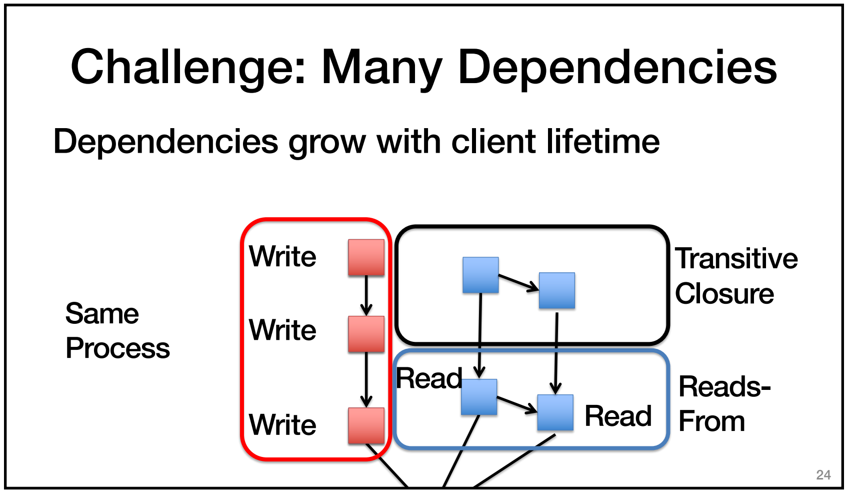
22



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

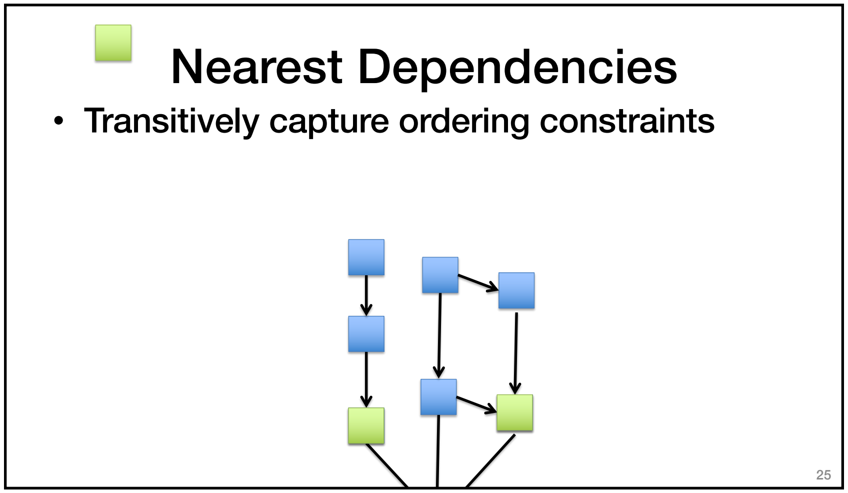
23



Challenge: Many Dependencies

Dependencies grow with client lifetime

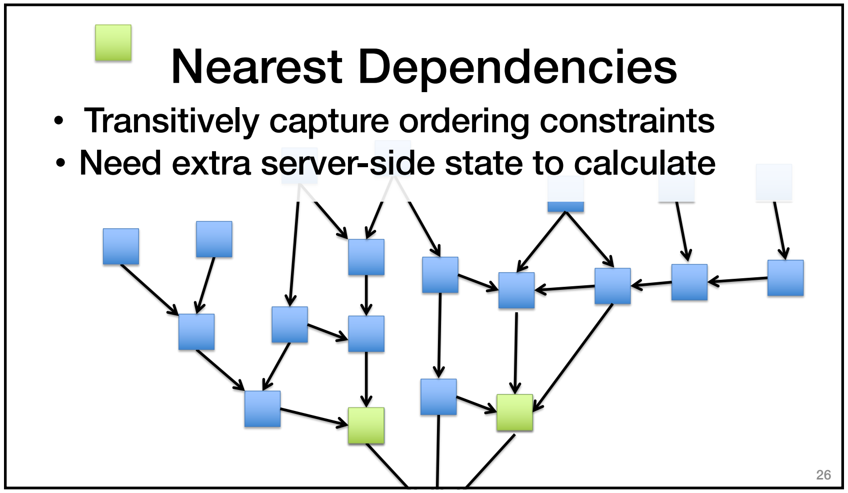
24



Nearest Dependencies

- Transitively capture ordering constraints

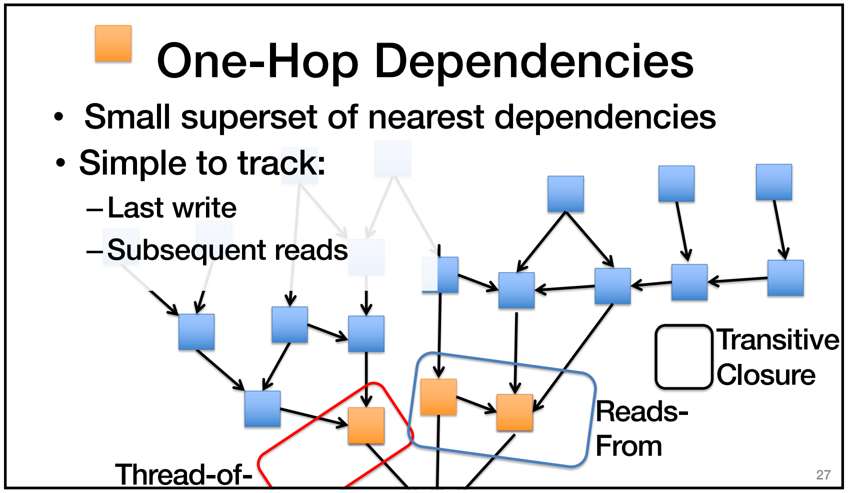
25



🟩 Nearest Dependencies

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

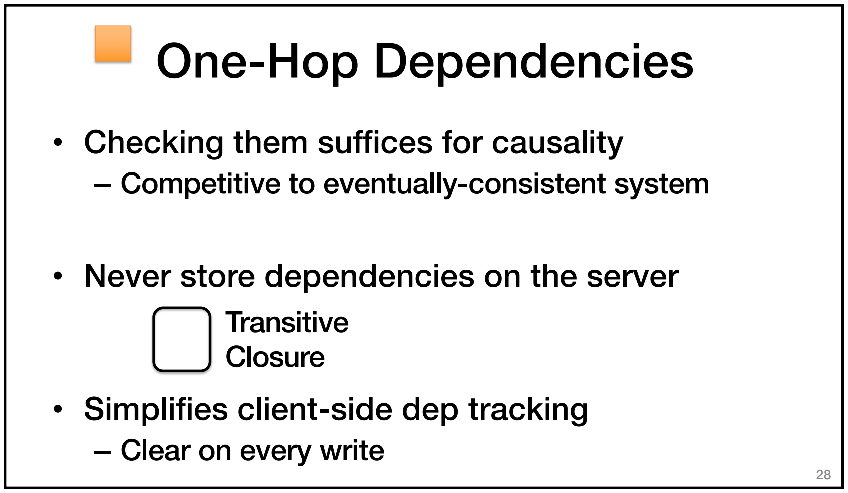
26



🟧 One-Hop Dependencies

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

27

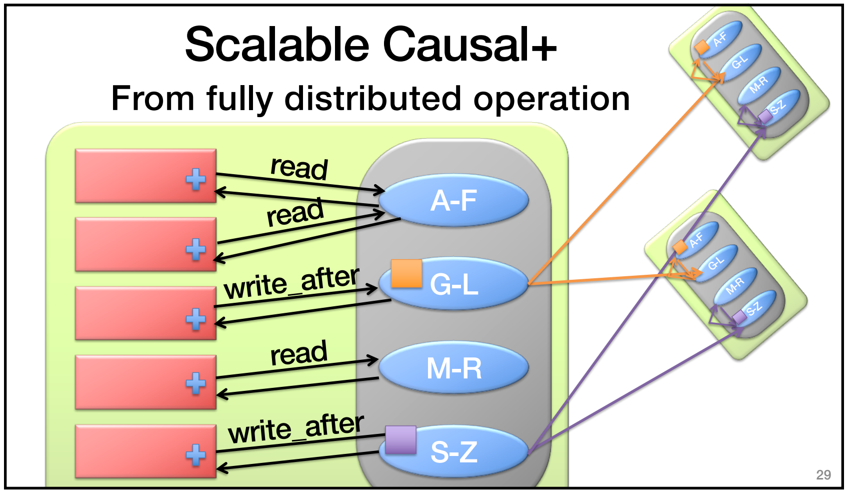


🟧 One-Hop Dependencies

- 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

28



Scalable Causal+

From fully distributed operation

29

Scalability

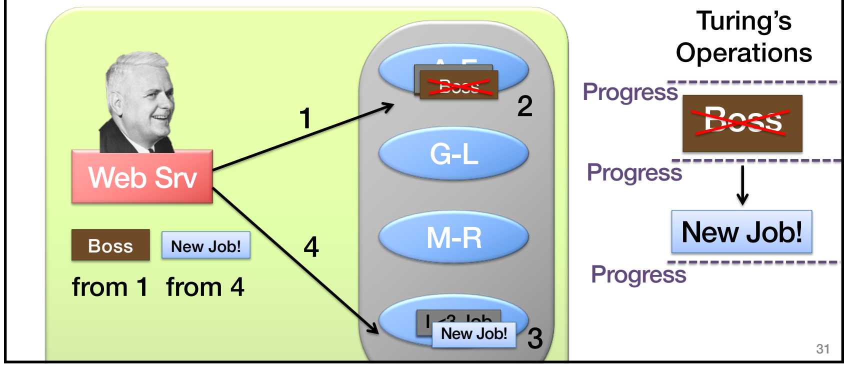
- Shard data for scalable storage
- New distributed protocol for scalably applying writes across shards
- Also need a new distributed protocol for consistently reading data across shards…

30

30

Reads Aren't Enough

Asynchronous requests + distributed data = ??

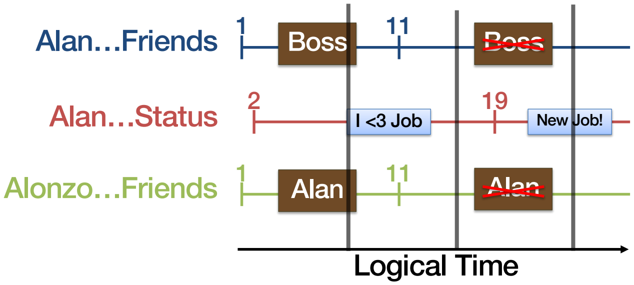


31

31

Read-Only Transactions

Consistent up-to-date view of data, across many servers

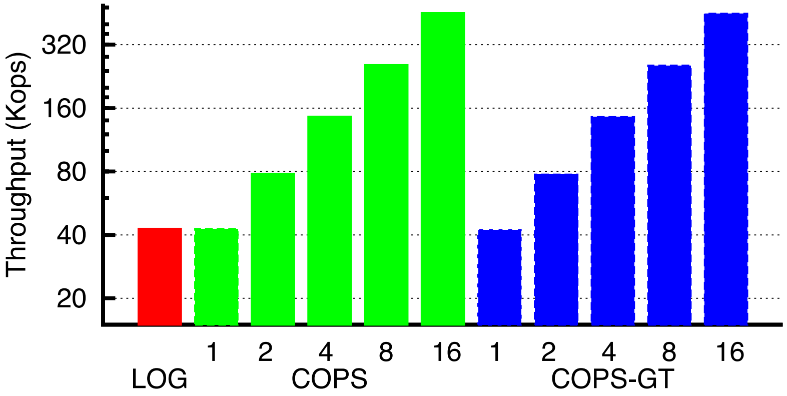


More on transactions next time!

32

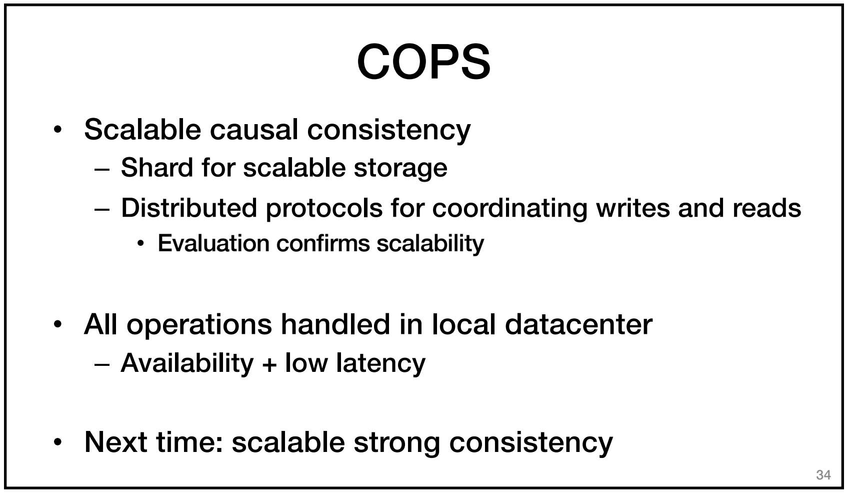
32

COPS Scaling Evaluation



33

33



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
- Next time: scalable strong consistency

34