Consistency Hierarchy (review)

1. Linearizability
   - e.g., RAFT

2. Sequential Consistency

3. Causal+ Consistency
   - e.g., Bayou

4. 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
- 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!

<table>
<thead>
<tr>
<th>System</th>
<th>Consistency</th>
<th>Scalable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paxos/RAFT</td>
<td>Linearizable</td>
<td>No</td>
</tr>
<tr>
<td>Bayou</td>
<td>Causal</td>
<td>No</td>
</tr>
<tr>
<td>Dynamo</td>
<td>Eventual</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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!

<table>
<thead>
<tr>
<th>System</th>
<th>Consistency</th>
<th>Scalable?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamo</td>
<td>Eventual</td>
<td>Yes</td>
</tr>
<tr>
<td>Bayou</td>
<td>Causal</td>
<td>No</td>
</tr>
<tr>
<td>COPS</td>
<td>Causal</td>
<td>Yes</td>
</tr>
<tr>
<td>Paxos/RAFT</td>
<td>Linearizable</td>
<td>No</td>
</tr>
</tbody>
</table>

Next Time!
COPS: Scalable Causal Consistency for Geo-Replicated Storage
Geo-Replicated Storage
serves requests quickly
Inside the Datacenter

Web Tier

Storage Tier

A-F

G-L

M-R

S-Z

Replication

Remote DC

Web Tier

Storage Tier

A-F

G-L

M-R

S-Z
Scalability through Sharding
Causality By Example

Remove boss from friends group

Post to friends: “Time for a new job!”

Friend reads post

Causality (→)
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
- Enforce with distributed verifications
  - Delay exposing replicated writes until all dependencies are satisfied in the datacenter
COPS Architecture

All Ops Local = Available and Low Latency
COPS Architecture
Write

write after = ordering metadata

Client Library

write

write after

Replication

write after
Replicated Write

Exposing values after dep_checks return ensures causal

Locator Key

Unique Timestamp

deps

A

F

test

G-L

M-R

S-Z

dep_check(A_{195})
dep_check(L_{337})

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
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 = ??

Web Srv

from 1

Boss

New Job!

from 4

G-L

Boss

M-R

New Job!

New Job!

Turing’s Operations

Progress

Progress

Progress
Read-Only Transactions

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

More on transactions next time!
COPS Scaling Evaluation

Throughput (Kops)

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