Scalable Causal Consistency

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

- 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>

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

### Geo-Replicated Storage
serves requests quickly

Inside the Datacenter

---

Remote DC
Scalability through Sharding

Causality By Example

Bayou’s Causal Consistency

Sharded Log Exchange
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

Client Library

Read

Client Library
Write

Replication

write after = ordering metadata

Client Library

A-F
G-L
M-R
S-Z

Write

Replicated Write

Exposing values after dep_checks return ensures causal

Unique Timestamp
Locator Key
deps
L_{337}
A_{195}
dep_check(A_{195})

write_after(...,deps)

Scalable Causal+

From fully distributed operation

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

COPS Scaling Evaluation

More on transactions next time!

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