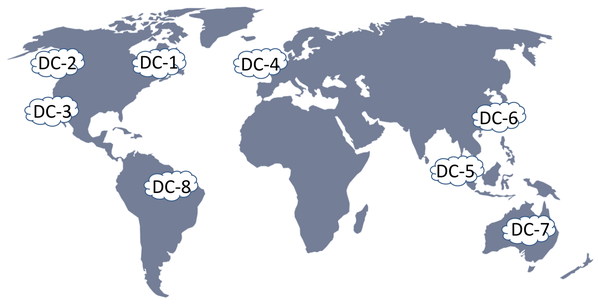
Karma: Cost-effective Geo-replicated Cloud Storage with Dynamic Enforcement of Causal Consistency

Tariq Mahmood, Shankaranarayanan P. N., Sanjay Rao, T. N. Vijaykumar, and Mithuna Thottethodi
School of Electrical and Computer Engineering, Purdue University

Motivation

Modern Cloud Storage Systems



- Span multiple geo-distributed DCs
  - Twitter, Facebook, Google, Amazon
  - Amazon has 8 worldwide DCs
- Replicate data for low latency
- Use asynchronous write propagation
- Have to handle failures, and partitions

The Consistency Spectrum

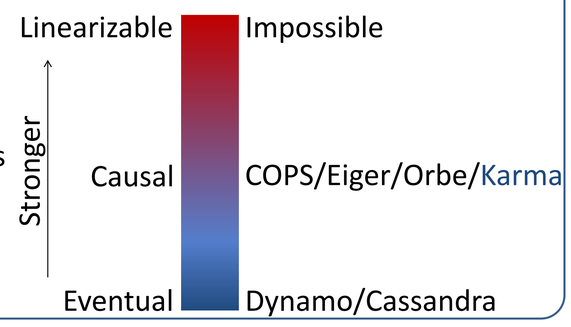
Replication, asynchronous write propagation create ordering issues

- Weak "eventually consistent" systems
  Widely deployed, but ordering can be confusing
- Strong ordering of all reads and writes across all clients
  CAP Theorem ⇒ unavailable on partition

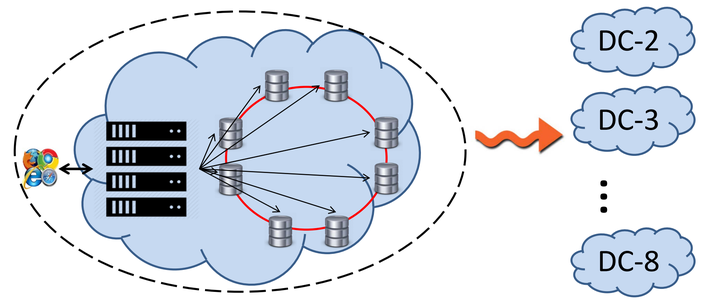
Causal consistency:
Partial order that preserves causality

Not confusing for users

Available under partition



Problems with Existing Causal Approaches

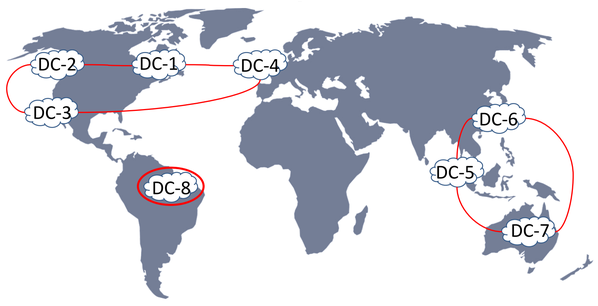


- Static binding: A user is allowed to access only one DC
- Full replication: Expensive, scalability issues
- Simple solutions do not work
  - Spreading data across DCs ⇒ Availability issues
  - Allowing users to switch DCs ⇒ Consistency violation

Karma: First causally consistent geo-replicated cloud storage system with partial replication while preserving consistency and availability

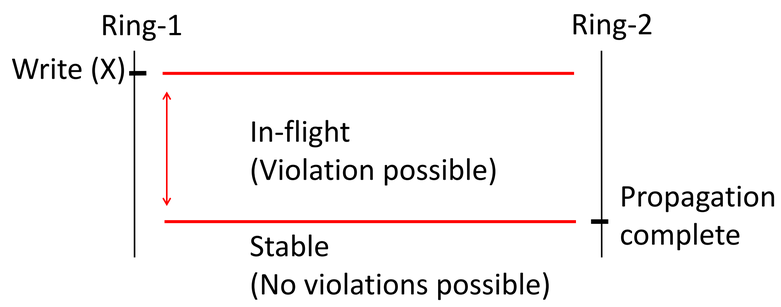
Karma's Key Ideas

Partial Replication



- Decouple rings and DCs
- Rings span multiple DCs
  Each ring contains full replica of dataset
- Availability in wide-area rings guaranteed by causality-preserving dynamic ring binding
- DC level caching used for fast reads of remote objects

Dynamic Ring Binding



- Karma's novel mechanism: Dynamic Ring Restrictions (DRR)
- If a client reads an in-flight object from Ring-1
  Temporarily restrict client to read all objects from Ring-1
  Client can access any ring once in-flight objects are stable

Caching/Write Buffers

- Partial replication ⇒ Remote objects, slow
- DC-level storage caches enable fast reads:
  - Problem: Normal cache operation violates causality
  - Solution: Stable value caching
- Persistent thread-private write buffers enable fast writes
  - All writes are local
  - Reads check write-buffer to avoid violations

Performance Evaluation (R/W : 95/5)

Experimental Setup
- 64-node testbed on PRObE cluster
- 8 data centers, 8 nodes each
- Amazon AWS emulation using DummyNet

Four Schemes:

	COPS-Ideal	Karma	COPS-PR	Karma-NC
Replication	Full	Partial	Partial	Partial
Ring binding	Static	Dynamic	Static	Dynamic
Write buffers	-	✓	✓	✓
Caching	-	✓	✗	✗

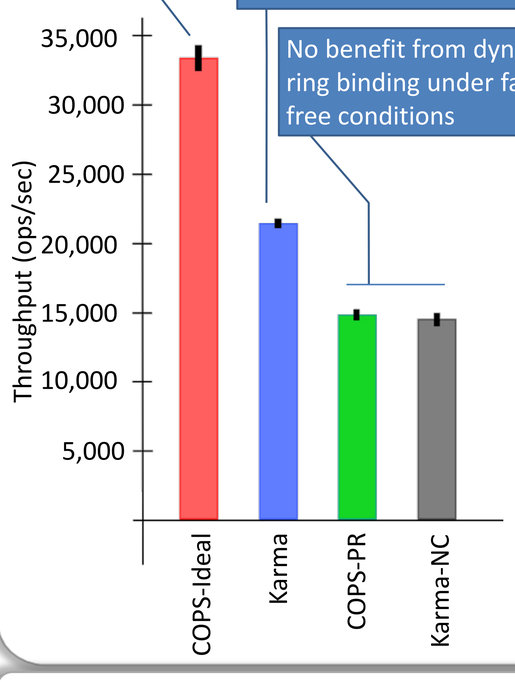
COPS-Ideal looks better, incurs high (impractical) cost of full replication

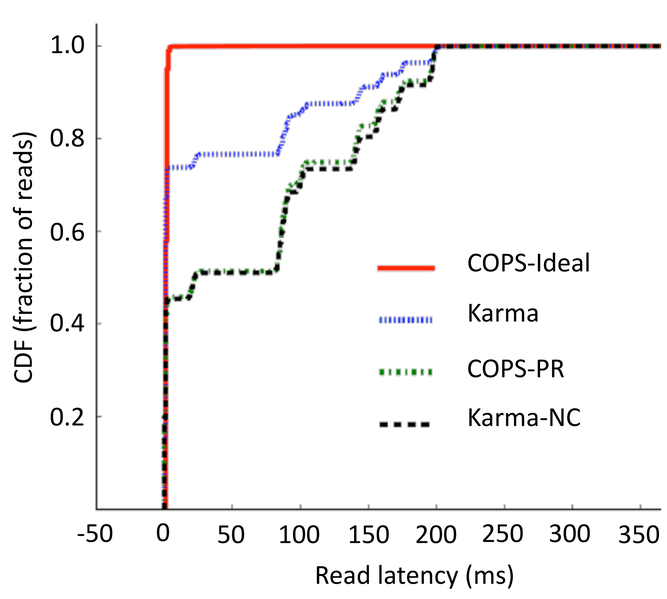
Karma has ≈30% cache misses

Karma's advantage diminished here since writes are only 5%

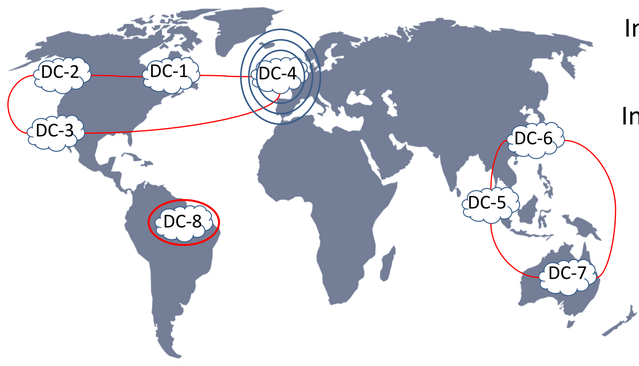
Karma achieves 43% higher throughput than COPS-PR

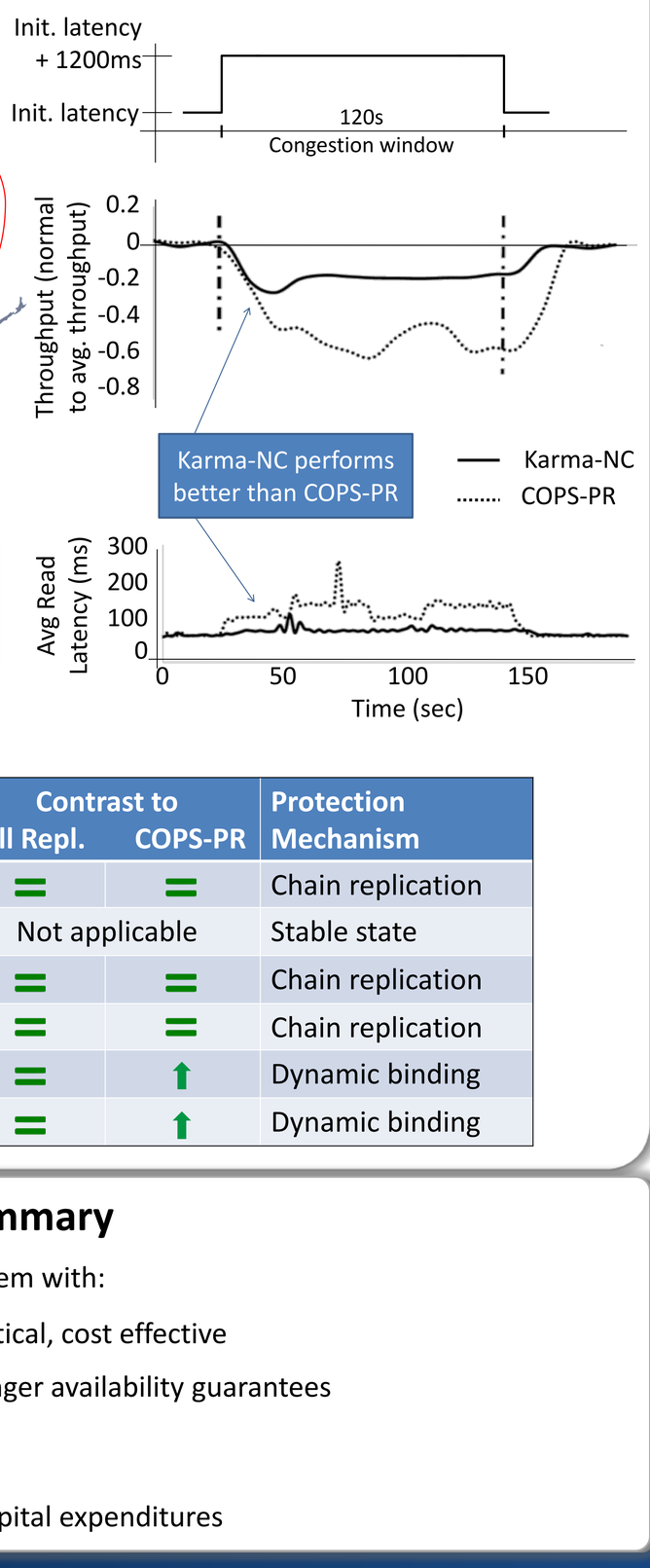
No benefit from dynamic ring binding under fault free conditions





Performance with Faults





Importance of Dynamic Ring Binding
- Induce congestion in Europe DC
- All traffic (in and out) is affected
- Table below shows avg. performance hit

	Karma-NC	COPS-PR
Throughput degradation	≈ 20%	≈ 50%
Latency increase	≈ 20%	≈ 85%

Fault Tolerance Analysis

Failure	Availability	Contrast to Full Repl.	Contrast to COPS-PR	Protection Mechanism
Backend Server	✓	=	=	Chain replication
Cache Server	✓	Not applicable		Stable state
Frontend Server	✓	=	=	Chain replication
Rack	✓	=	=	Chain replication
Single AZ	✓	=	↑	Dynamic binding
Partition	✓	=	↑	Dynamic binding

Cost Savings

Summary

- First causally-consistent cloud storage system with:
  - ✓ Partial replication ⇒ Practical, cost effective
  - ✓ Dynamic ring switching ⇒ Stronger availability guarantees
- 43% throughput improvement iso-cost
- Significant reduction in operational and capital expenditures

This material is based upon work supported by the National Science Foundation under Grant No. CSR-1618921