I Can't Believe It's Not Causal!
Scalable Causal Consistency with No Slowdown Cascades

Syed Akbar Mehdi, Cody Littley and Natacha Crooks, *UT Austin*; Lorenzo Alvisi, *UT Austin and Cornell;* Nathan Bronson, *Facebook;* Wyatt Lloyd, *USC*

14th USENIX Symposium on Networked Systems Design and Implementation (NSDI '17)

PROBLEM

Causal Consistency

In Theory

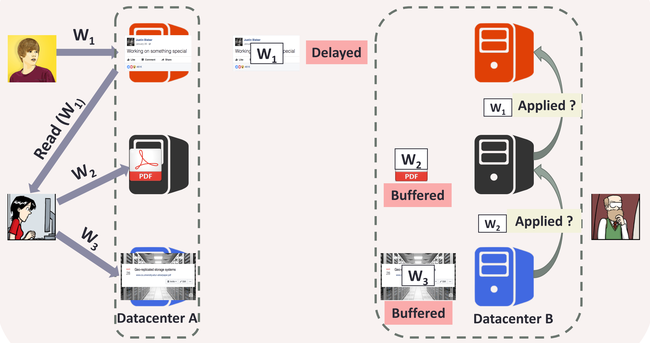
In Practice

- The largest web applications use eventually consistent datastores
- Examples:

Espresso TAO Manhattan

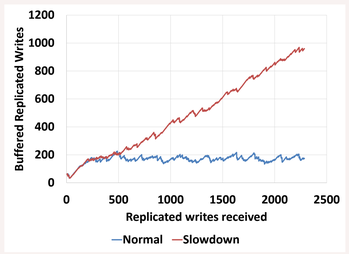
Key Adoption Hurdle: Slowdown Cascades

Slowdown Cascade Example



Writes causally ordered as $W_1 \rightarrow W_2 \rightarrow W_3$

Slowdown Cascades in Eiger (NSDI '13)



Replicated write buffers grow arbitrarily because Eiger enforces consistency inside the datastore

SOLUTION

Observable Causal Consistency

Causal Consistency guarantees that each client *observes* a monotonically non-decreasing set of updates (including its own) in an order that respects potential causality between operations

Key Idea:

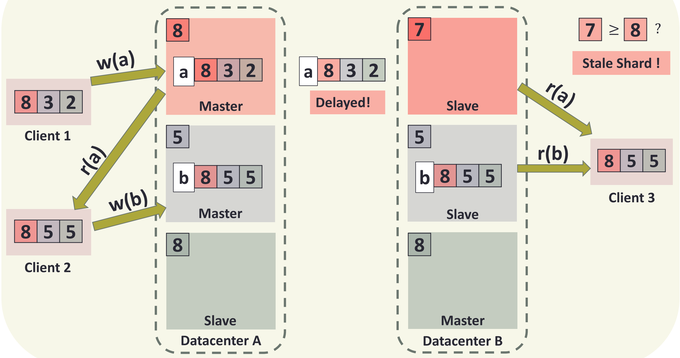
Don't implement a causally consistent data store

Let clients *observe* a causally consistent data store

Implementing Observable Causal Consistency

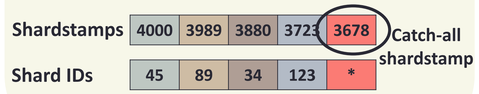
1. Our solution is a system called OCCULT (Observable Causal Consistency Using Lossy Timestamps)
2. Each client maintains metadata (called a Causal Timestamp) to encode the most recent snapshot of the datastore it has observed
3. Writes replicate asynchronously without any buffering
4. On reads clients use the causal timestamp to detect whether a shard is safe to read from

Example



Causal Timestamp Compression

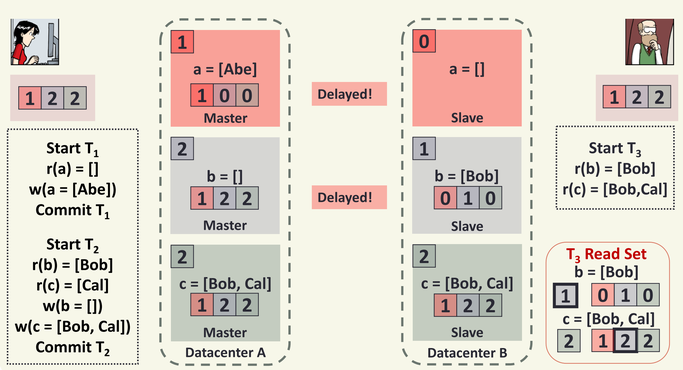
1. Use local clock of a partition to increment shardstamps.
   a) Keeps shardstamps loosely synchronized despite varying write rates on shards
2. Use high resolution for recent shardstamps and conflate the rest



Scalable Distributed Transactions

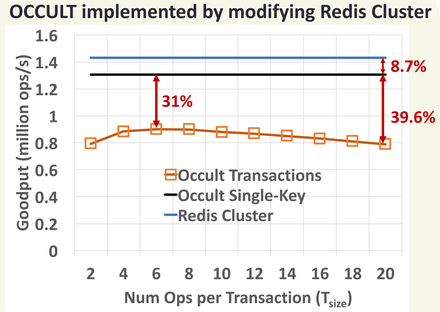
1. Occult is the first casual system to support general purpose read-write transactions !
   a) And still no slowdown cascades !
2. Transactions have the following properties:
   a) (Observable) Atomicity
   b) (Observable) Reads from a casually consistent snapshot
   c) No concurrent conflicting writes
3. Transactions are scalable
   a) No centralized timestamp authorities (or sequencers) ! Transactions are ordered using causal timestamps
   b) Transaction commit latency is independent of the number of replicas !

Transactions Example



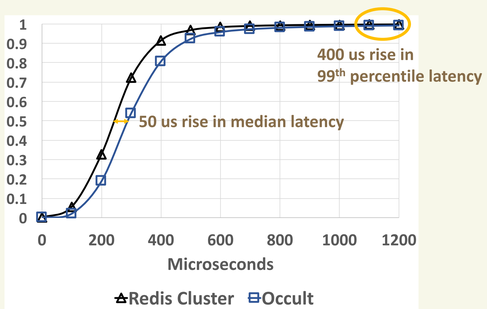
Evaluation

OCCULT implemented by modifying Redis Cluster



Goodput evaluated on a heavily contended, zipfian, read-heavy YCSB workload

Evaluation



Latency Overhead over Redis Cluster