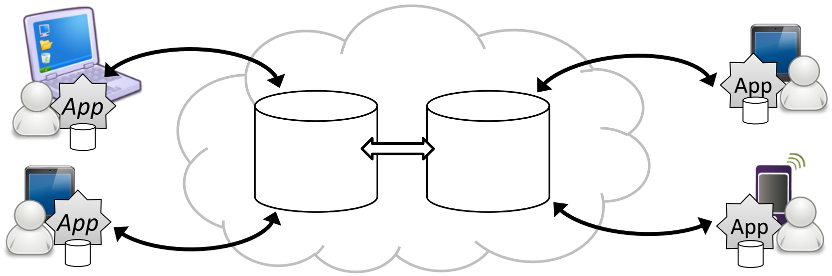
Write Fast, Read in the Past: Causal Consistency for Client-side Apps with SwiftCloud

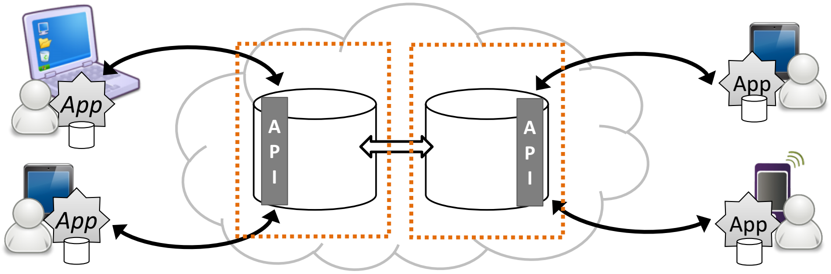
Presented by Marek Zawirski
Inria / UPMC-LIP6, Paris
(now at Google, Zürich)

Marek Zawirski, Nuno Preguiça, Sérgio Duarte, Annette Bieniusa, Valter Balegas, Marc Shapiro

Challenge: Database Access for Client-side Apps

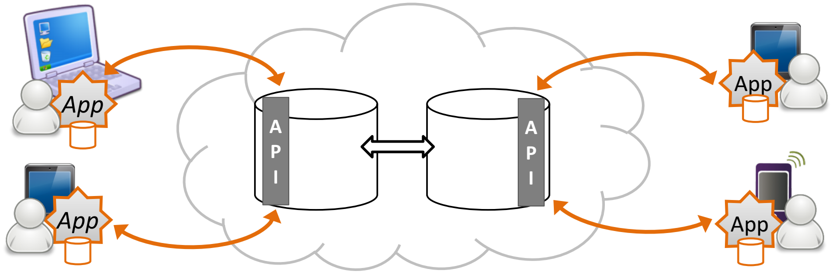


Challenge: Database Access for Client-side Apps



Limited boundaries of server-side database guarantees

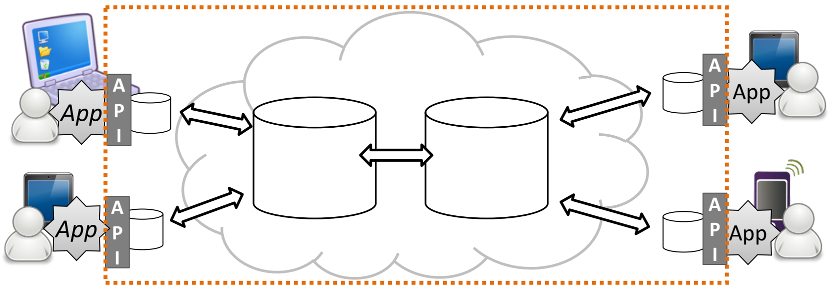
Challenge: Database Access for Client-side Apps



Limited boundaries of server-side database guarantees

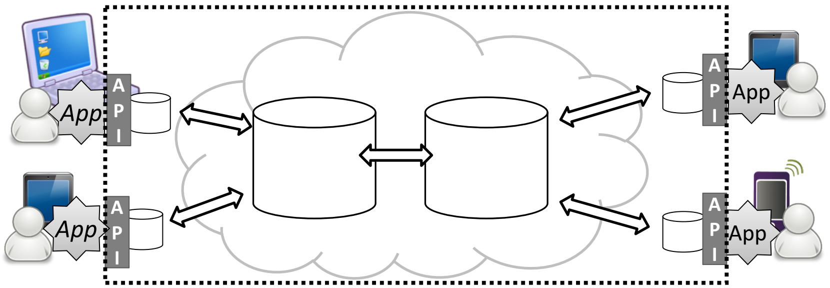
⇒ ad-hoc on the client-side

Challenge: Database Access for Client-side Apps



Extended boundaries with **SwiftCloud**

Challenge: Database Access for Client-side Apps

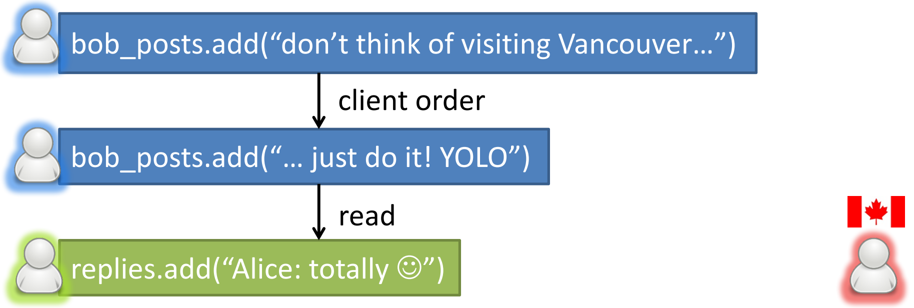


Extended boundaries with **SwiftCloud**

- **Consistent, available** and **convergent data access**
- **Scalability** with #objects and #clients
- **Fault-tolerance**

Stronger than Eventual: Causal Consistency

Default on client-side: eventual consistency ⇒ anomalies



bob_posts.add("don't think of visiting Vancouver…")

client order

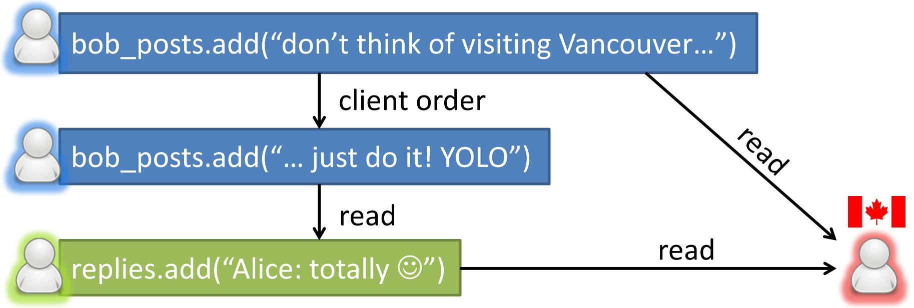
bob_posts.add("… just do it! YOLO")

read

replies.add("Alice: totally ☺")

Stronger than Eventual: Causal Consistency

Default on client-side: eventual consistency ⇒ anomalies



bob_posts.add("don't think of visiting Vancouver…")

client order

read

bob_posts.add("… just do it! YOLO")

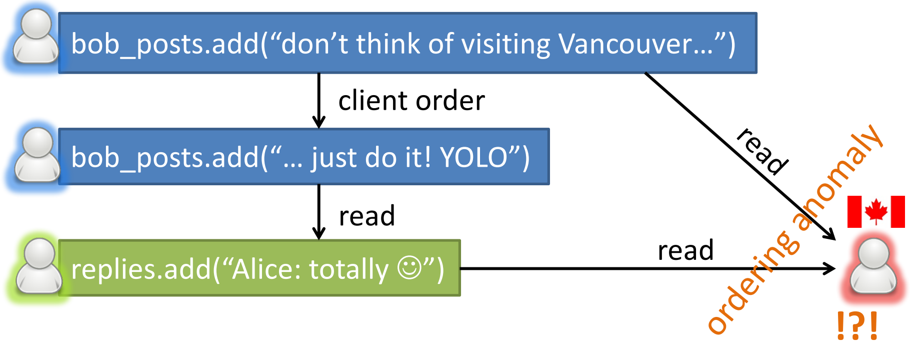
read

replies.add("Alice: totally ☺")

read

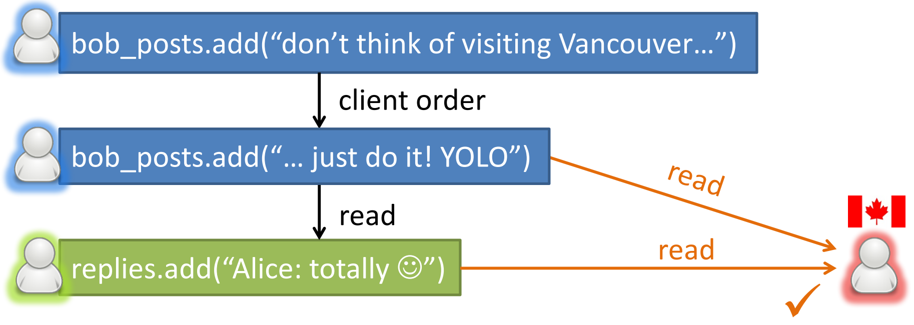
Stronger than Eventual: Causal Consistency

Default on client-side: eventual consistency ⇒ anomalies


bob_posts.add("don't think of visiting Vancouver…")
client order
bob_posts.add("… just do it! YOLO")
read
replies.add("Alice: totally ☺")
read
read
ordering anomaly
!?!

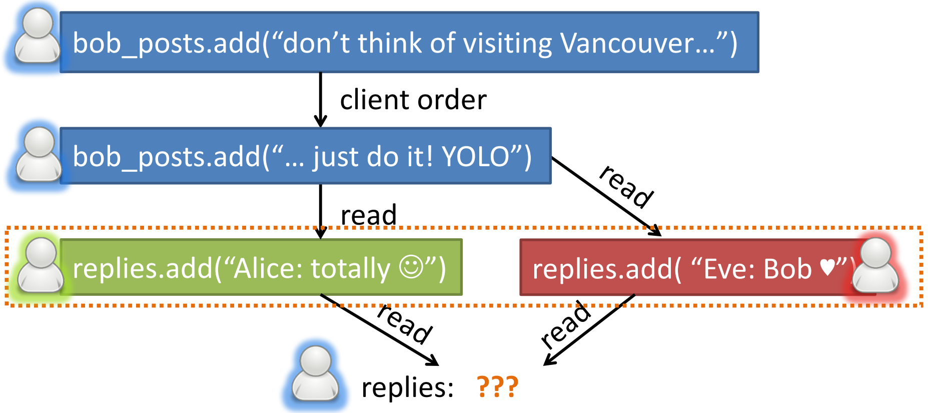
Stronger than Eventual: Causal Consistency

Default on client-side: eventual consistency ⇒ anomalies

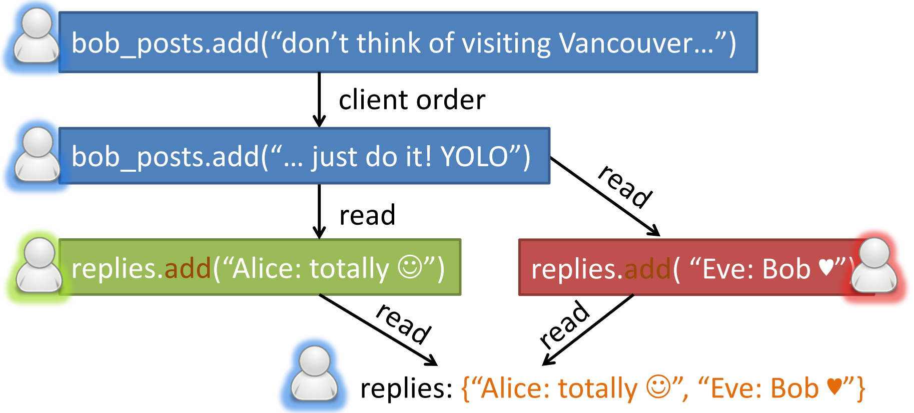

bob_posts.add("don't think of visiting Vancouver…")
client order
bob_posts.add("… just do it! YOLO")
read
replies.add("Alice: totally ☺")
read
read
✓

Causal consistency: reads from causally-closed snapshot

Convergent Causal Consistency: No Lost Updates


bob_posts.add("don't think of visiting Vancouver…")
client order
bob_posts.add("… just do it! YOLO")
read
read
replies.add("Alice: totally ☺")
replies.add("Eve: Bob ♥")
read
read
replies: ???

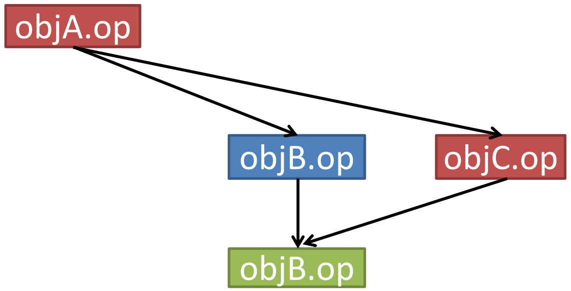
Convergent Causal Consistency: No Lost Updates


bob_posts.add("don't think of visiting Vancouver…")
client order
bob_posts.add("… just do it! YOLO")
read
read
replies.add("Alice: totally ☺")
replies.add("Eve: Bob ♥")
read
read
replies: {"Alice: totally ☺", "Eve: Bob ♥"}

High-level convergent objects[CRDTs] resolve concurrency

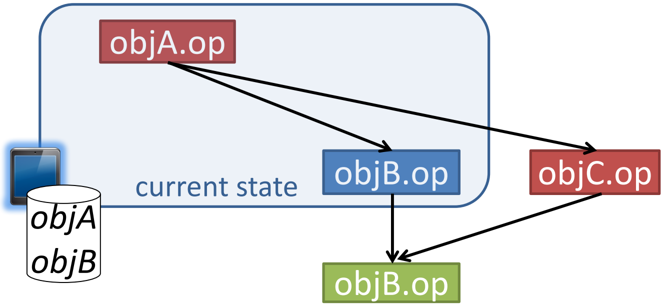
Challenge: Causal Consistency with Partial Replicas

[PRACTI, NSDI’06]



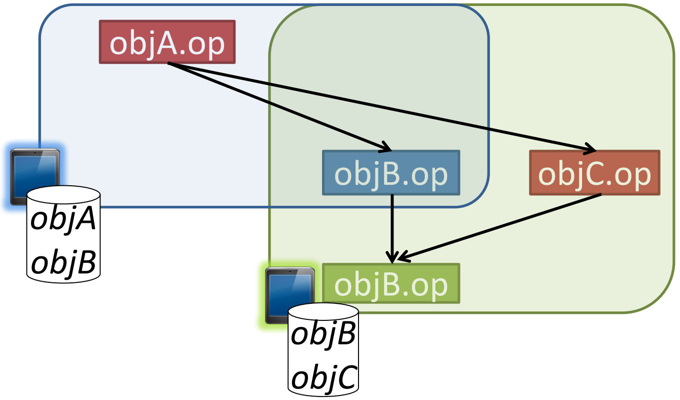
Challenge: Causal Consistency with Partial Replicas

[PRACTI, NSDI’06]



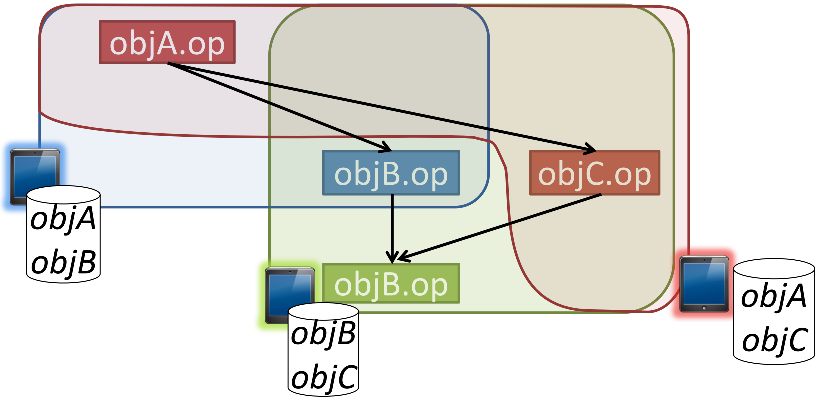
Challenge: Causal Consistency with Partial Replicas

[PRACTI, NSDI’06]



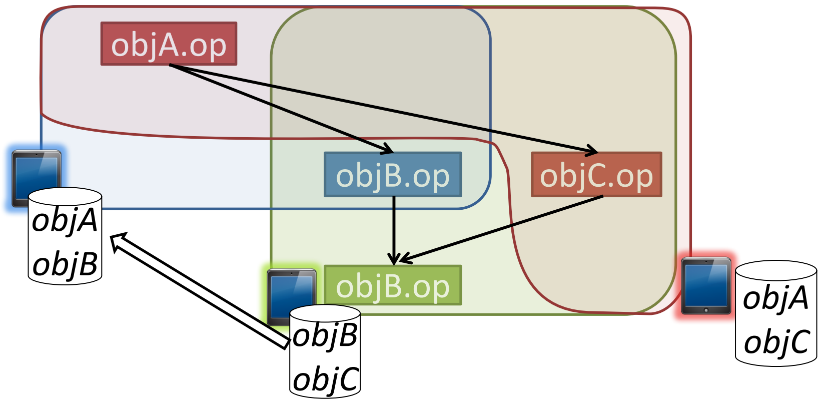
Challenge: Causal Consistency with Partial Replicas

[PRACTI, NSDI’06]



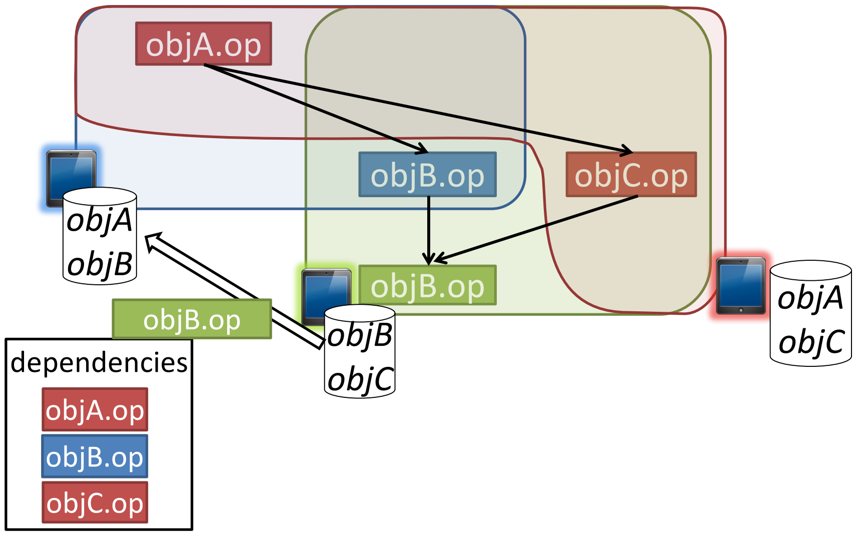
Challenge: Causal Consistency with Partial Replicas

[PRACTI, NSDI'06]



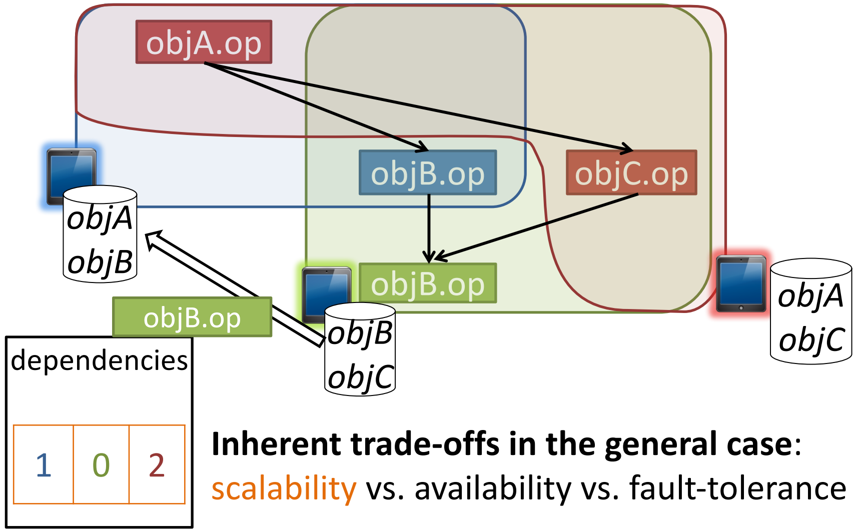
Challenge: Causal Consistency with Partial Replicas

[PRACTI, NSDI'06]



Challenge: Causal Consistency with Partial Replicas

[PRACTI, NSDI'06]

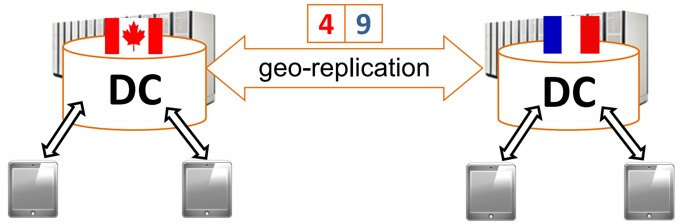


Inherent trade-offs in the general case:
scalability vs. availability vs. fault-tolerance

Approach: Cloud-backed Partial Replicas

Data Center full replicas:

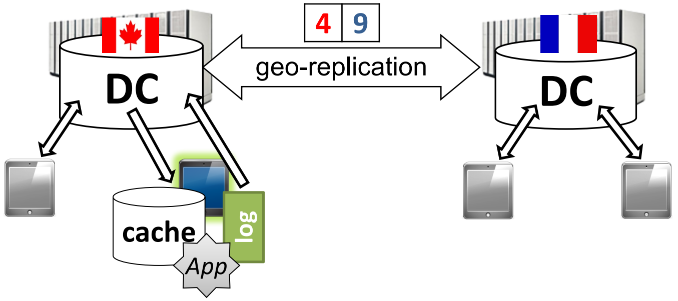
✓Provide consistent view ✓ Assign small metadata



Approach: Cloud-backed Partial Replicas

Data Center full replicas:

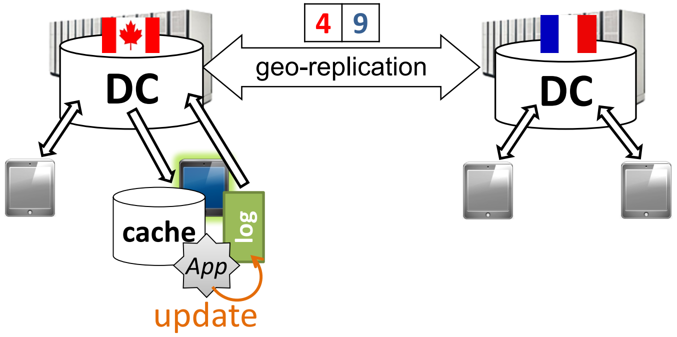
✓Provide consistent view ✓ Assign small metadata



Approach: Cloud-backed Partial Replicas

Data Center full replicas:

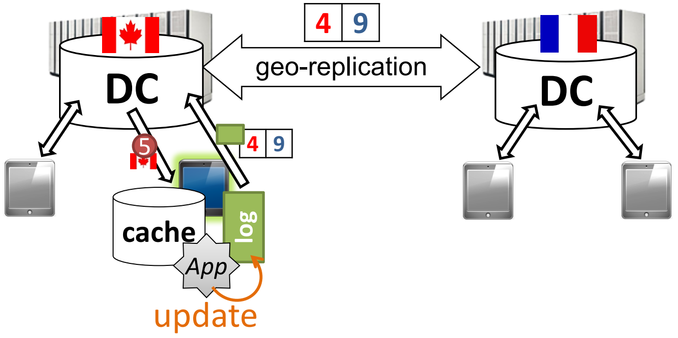
✓Provide consistent view ✓ Assign small metadata



Approach: Cloud-backed Partial Replicas

Data Center full replicas:

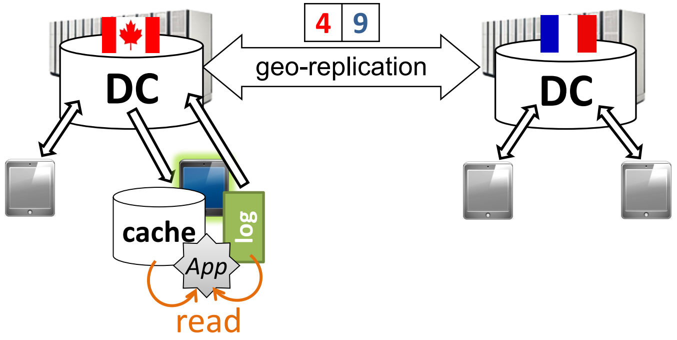
✓Provide consistent view ✓ Assign small metadata



Approach: Cloud-backed Partial Replicas

Data Center full replicas:

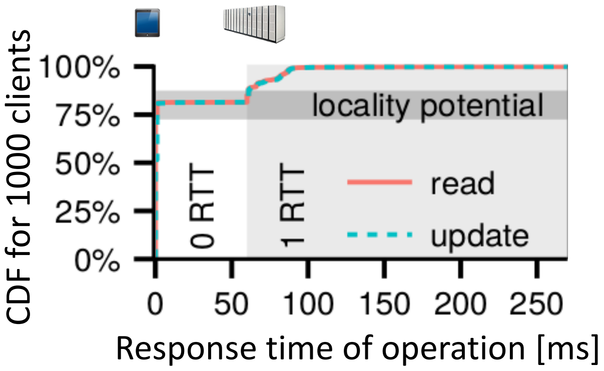
✓Provide consistent view ✓ Assign small metadata



Client reads: cached fragment of cloud version ∪ own log ✓

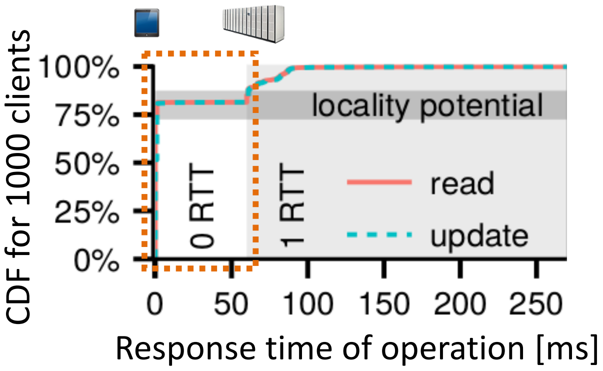
✓ High availability Consistency w/read-your-writes

Potential of Cloud-backed Client Replicas



Setup: DCs in 3 AWS EC2 regions, YCSB workload, cache=256 objects

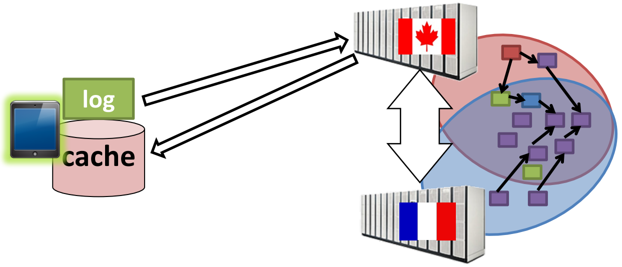
Potential of Cloud-backed Client Replicas



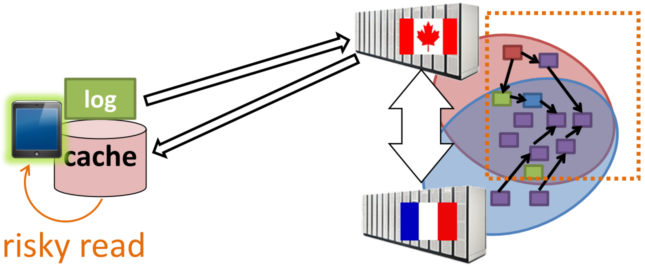
Objects in the cache ⇒ immediate, consistent response

Setup: DCs in 3 AWS EC2 regions, YCSB workload, cache=256 objects

Challenge for the Cloud Approach: Safe DC Failover

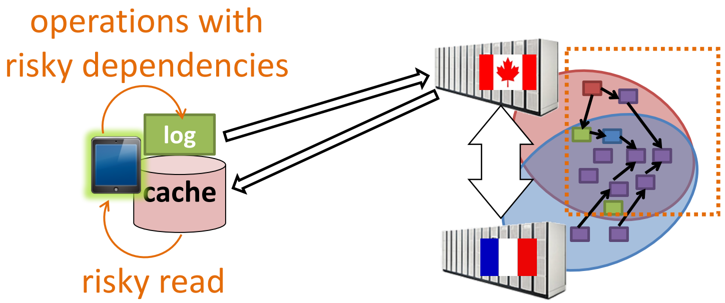


Challenge for the Cloud Approach: Safe DC Failover



risky read

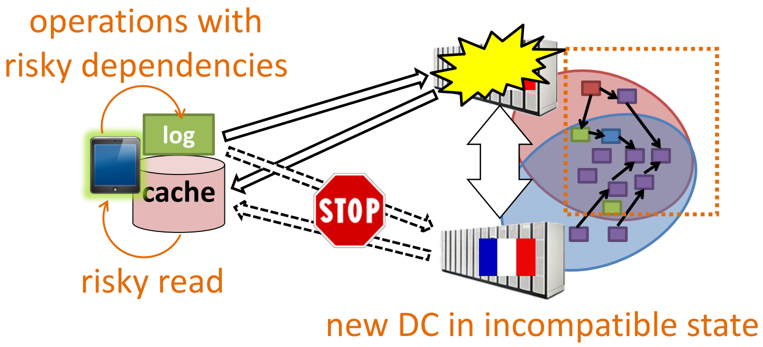
Challenge for the Cloud Approach: Safe DC Failover



operations with risky dependencies

risky read

Challenge for the Cloud Approach: Safe DC Failover

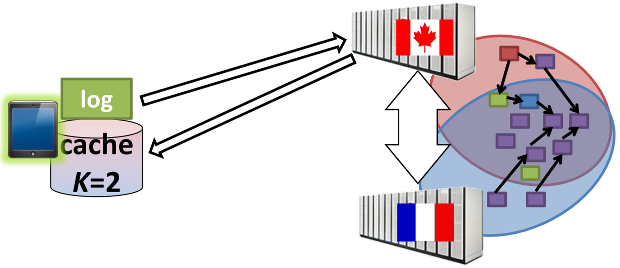


operations with risky dependencies

risky read

new DC in incompatible state

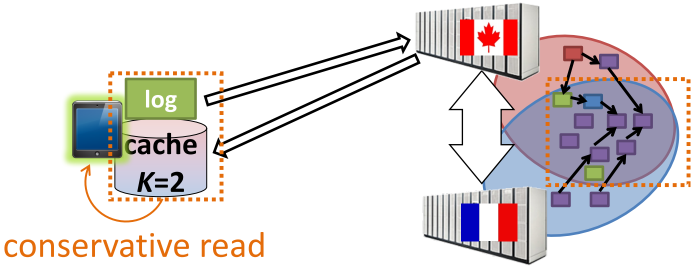
Supporting Failover by Conservative Reads



Foreign updates: read version replicated in $K > 1$ DCs

Own writes: read from the log, recover to a new DC

Supporting Failover by Conservative Reads

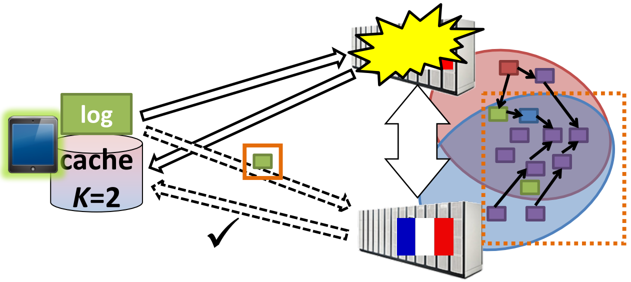


conservative read

Foreign updates: read version replicated in $K > 1$ DCs

Own writes: read from the log, recover to a new DC

Supporting Failover by Conservative Reads

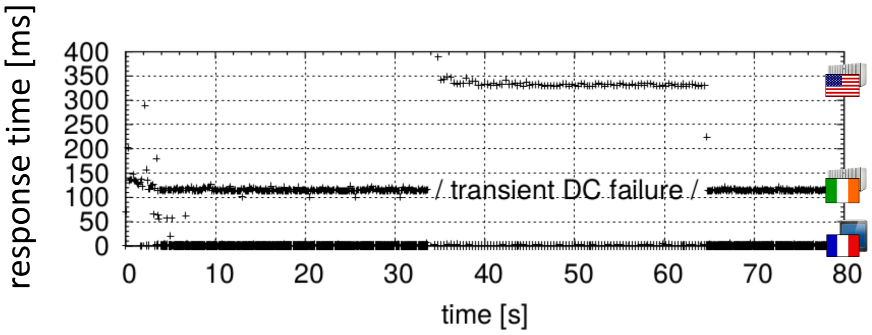


new DC in compatible state

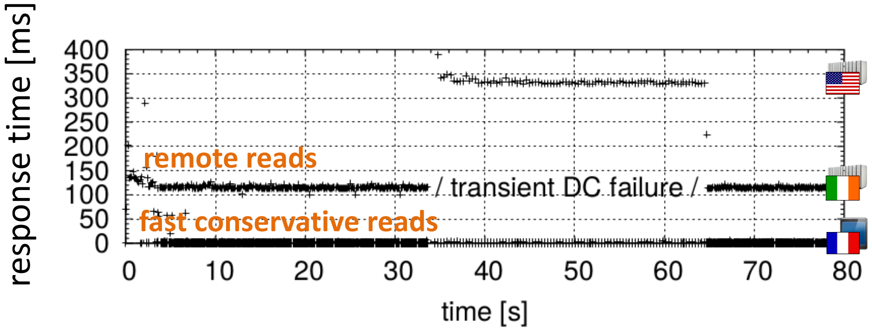
Foreign updates: read version replicated in $K > 1$ DCs

Own writes: read from the log, recover to a new DC

Experiment: Injection of Short DC Disconnection



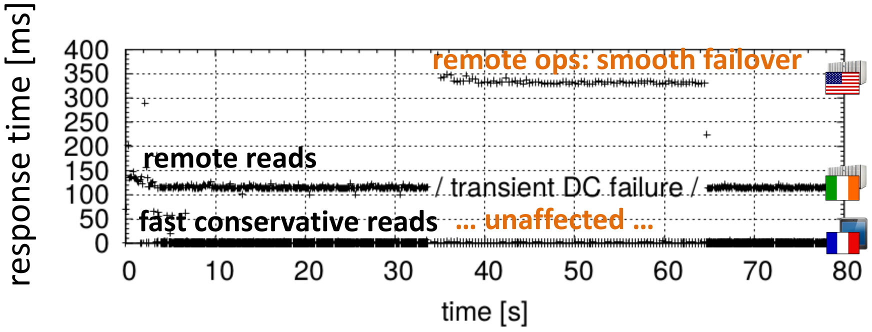
Experiment: Injection of Short DC Disconnection



remote reads

fast conservative reads

Experiment: Injection of Short DC Disconnection



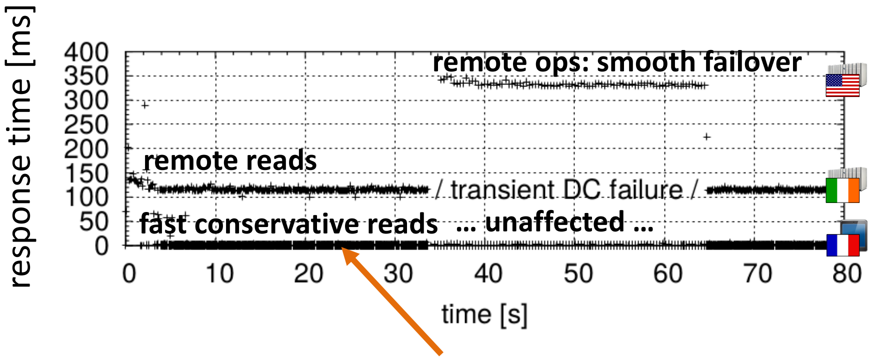
remote ops: smooth failover

remote reads

fast conservative reads

… unaffected …

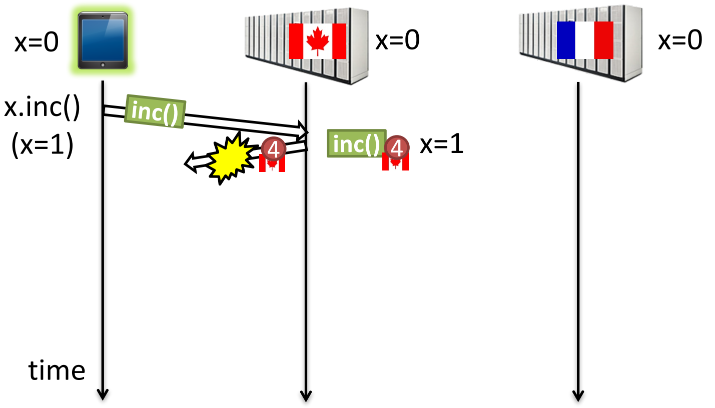
Experiment: Injection of Short DC Disconnection



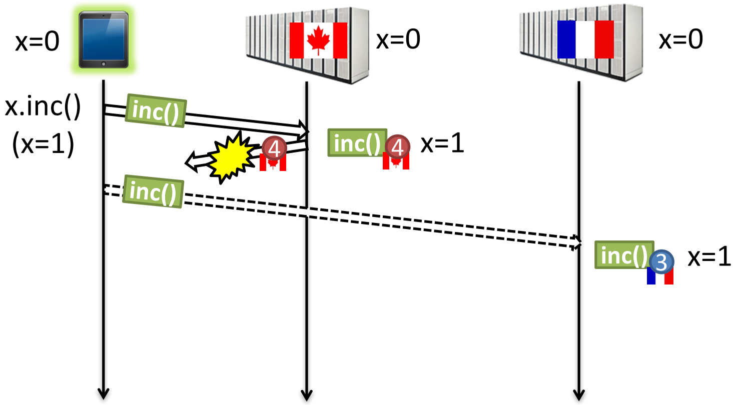
Trade-off controlled by *K*: **staleness vs. availability**

- Staleness negligible in most *K*=2 setups, < 1% reads
- In cherry-picked unfavorable setup, 1.0–2.5% reads

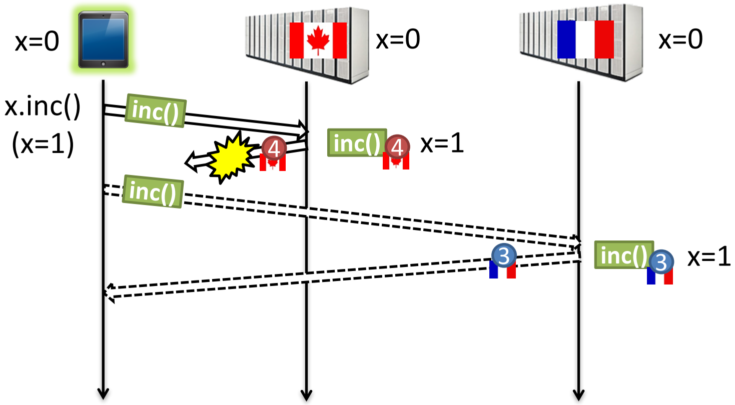
Challenge for the Cloud Approach: Protocol Retries



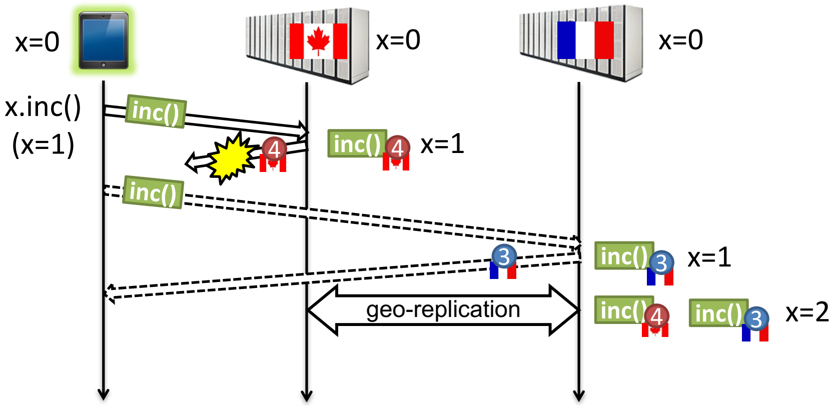
Challenge for the Cloud Approach: Protocol Retries



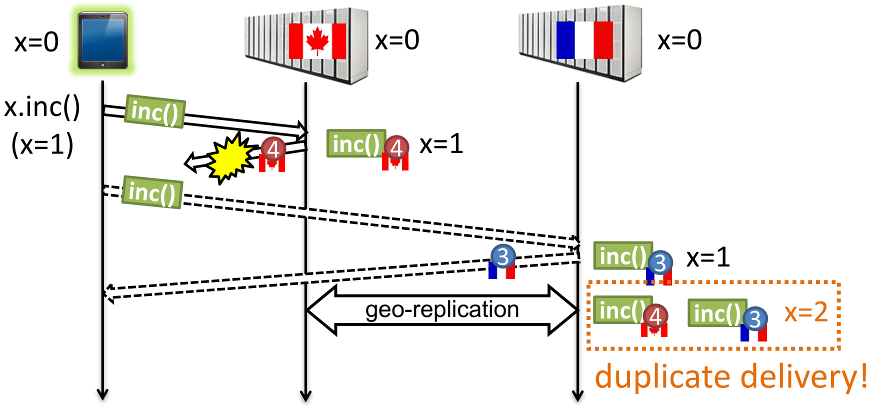
Challenge for the Cloud Approach: Protocol Retries



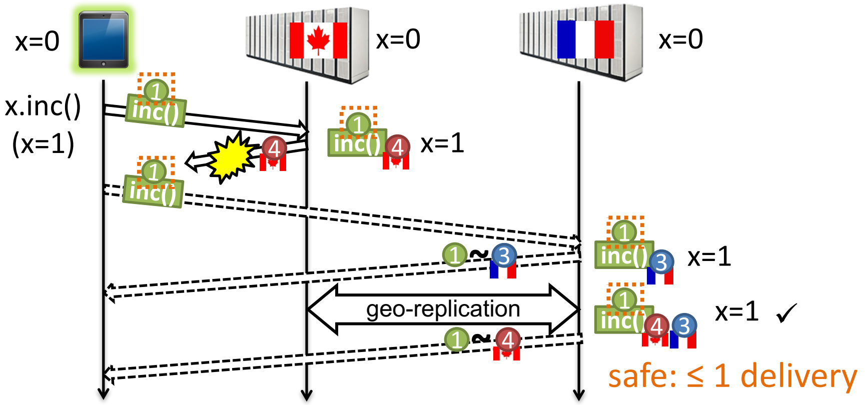
Challenge for the Cloud Approach: Protocol Retries



Challenge for the Cloud Approach: Protocol Retries

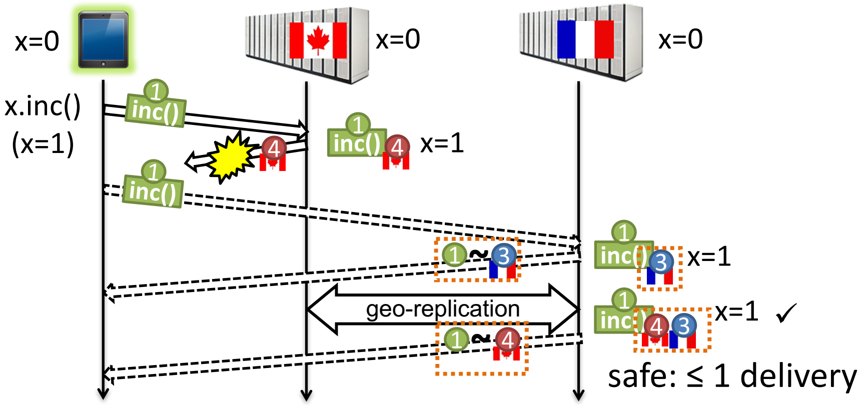


Safe Retries with Decoupled Metadata



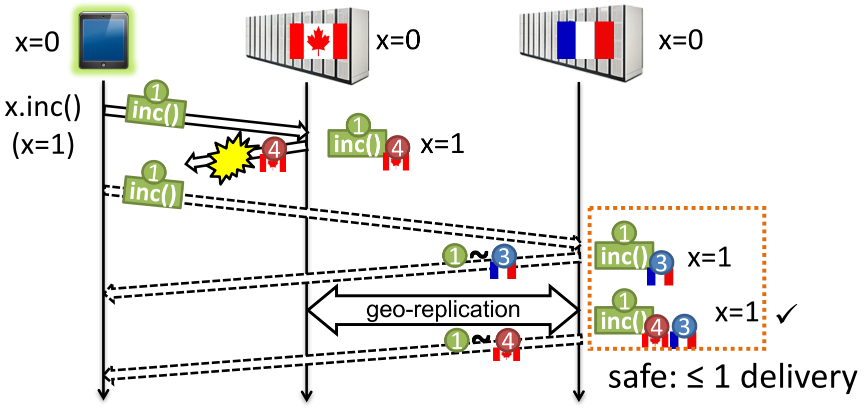
Solution: client-assigned timestamps for safety
+ 1..N DC timestamps for efficient summary

Safe Retries with Decoupled Metadata



Solution: client-assigned timestamps for safety
+ 1..N DC timestamps for efficient summary

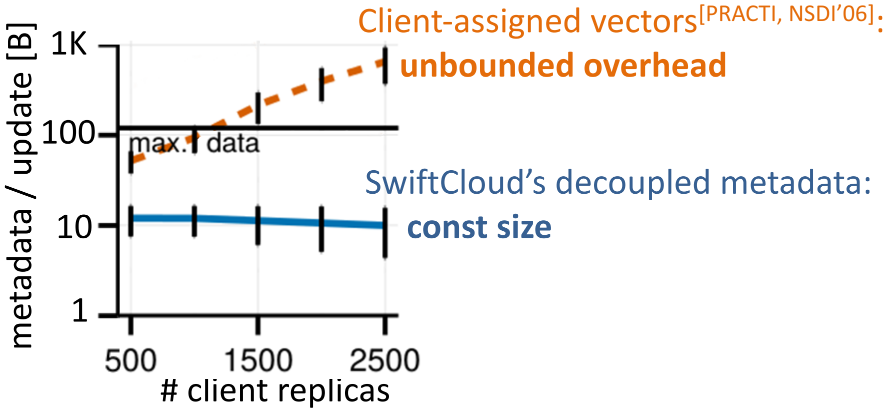
Safe Retries with Decoupled Metadata



Solution: client-assigned timestamps for safety
+ 1..N DC timestamps for efficient summary

Extension: log pruning independent of client availability

Experiment: Size of Metadata on Client-DC Link



Setup: 3DCs, YCSB B uniform workload

Summary

SwiftCloud provides **client-side** apps:

- **Consistent, available** and **convergent object database**
- **Scalability:** full replicas at DC back partial at client
  ⇒ small causality metadata (< 15B/update)
- **Fast failover** thanks to conservative reads (< 1% stale)
- **Safe retry** of interrupted transfer and **safe log pruning** thanks to decoupled metadata

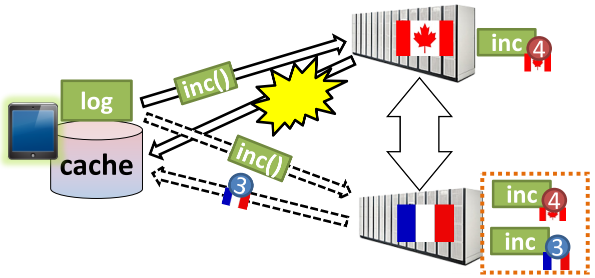
Research prototype at github.com/SyncFree/SwiftCloud

SwiftCloud compared to "Lazy Replication"

- Assume client-side application logic
- Describe causal consistency support
- Support communication with multiple servers
- Use decoupled metadata

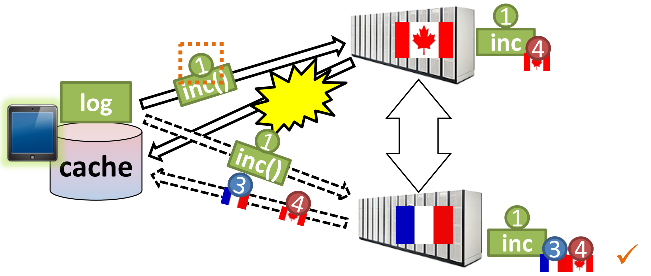
• DB = RDT objects + global transactions	• Monolithic DB
• Supports partial client replicas => fast reads and read-your-writes	• No client-side replicas
• K-stability-driven trade-off	• Stability discussion
• GC independent of clients	• Physical-clock-driven GC
	• More consistency choices

Challenge for the Cloud Approach: Protocol Retries



duplicate delivery!

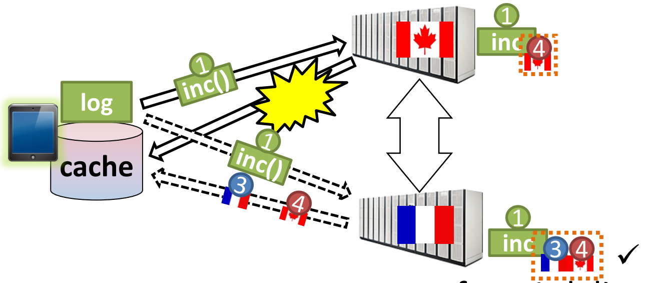
Safe Retries with Decoupled Metadata



safe: ≤ 1 delivery

Solution: client-assigned timestamps for safety
+ 1..N DC timestamps for efficient summary

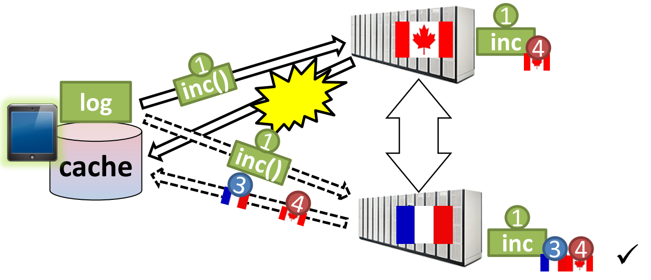
Safe Retries with Decoupled Metadata



safe: ≤ 1 delivery

Solution: client-assigned timestamps for safety
+ 1..N DC timestamps for efficient summary

Safe Retries with Decoupled Metadata



safe: ≤ 1 delivery

Solution: client-assigned timestamps for safety
+ 1..N DC timestamps for efficient summary
Extension: log pruning independent of client availability