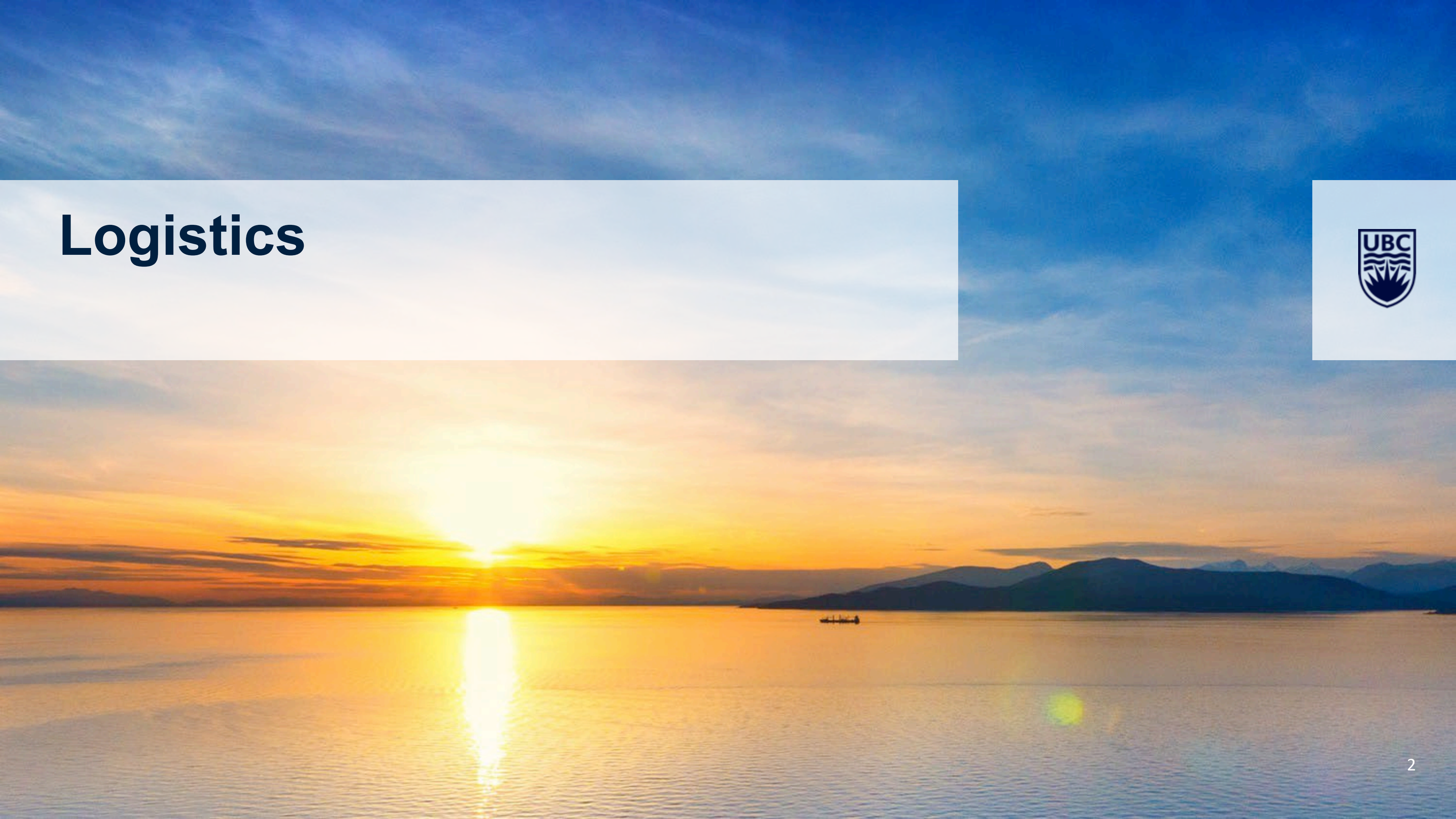
CPSC 416 Distributed Systems

Winter 2022 Term 2 (January 26, 2023)

Tony Mason (fsgeek@cs.ubc.ca), Lecturer



Logistics

Deadlines

Project 1 Code: Grades posted to Gradescope

Project 1 Report: Pending

Project 2 Code: Pending (Due January 31, 2023)

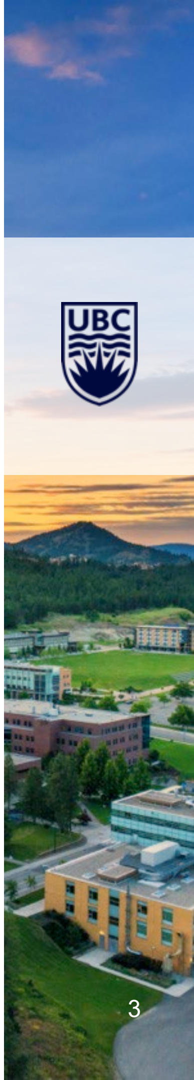
Project 2 Report: Pending

Project 3 Released. Initially Due: February 13, 2023

Project 4 Released. Initially Due: March 13, 2023

Project 5 Released Due: April 13, 2023

Note: all project work is due April 13, 2023. Late projects have a 75% score cap.



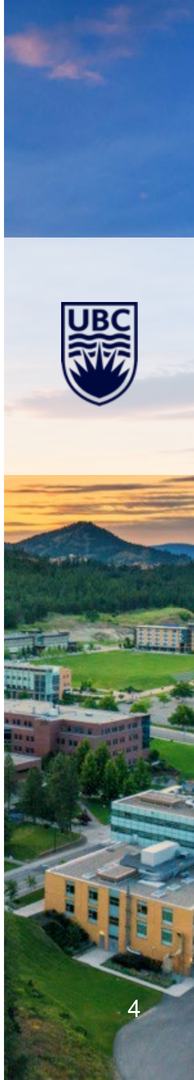
Deadlines

Alternate Path 1 & 2: Initial Proposal due January 30, 2023.

- **Submit on Canvas, no later than 11:59 pm PT**
- **All team members should submit the same proposal**
  - This is our consensus protocol: everyone agrees.

Instructor Office Hours:

- Zoom Office Hours (Tuesday) @ 13:00-14:00
- Discord (Casual) Office Hours (Thursday) @ 14:00-15:00



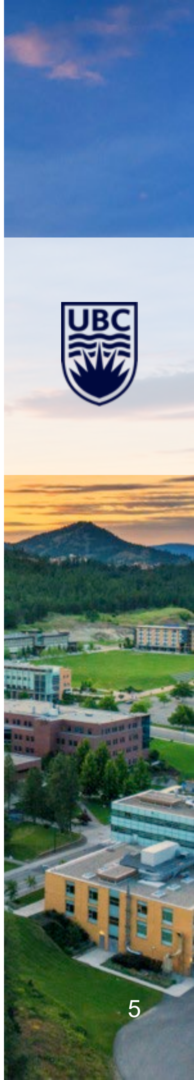
Readings

Required:

None

Recommended:

- Chain Replication for Supporting High Throughput and Availability
- Object Storage on CRAQ: High-throughput chain replication for read-only workloads

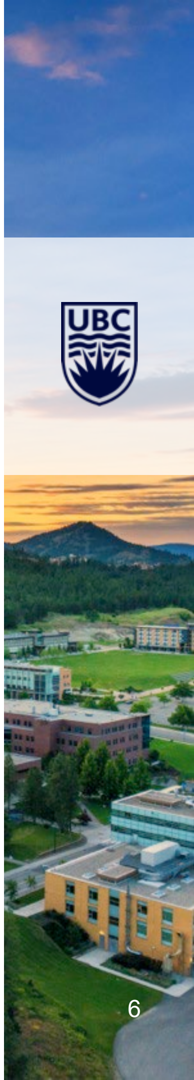


Questions?

Questions about the class?

Questions about the previous lecture?

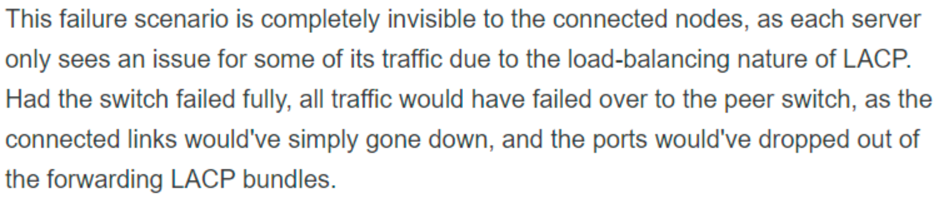
Funny stories to share?



Today’s Failure

Cloudfare

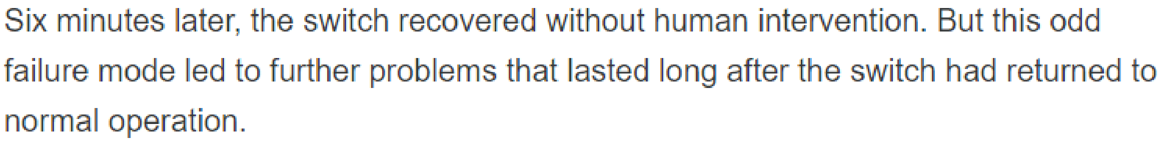
A Byzantine failure in the real world



This failure scenario is completely invisible to the connected nodes, as each server only sees an issue for some of its traffic due to the load-balancing nature of LACP. Had the switch failed fully, all traffic would have failed over to the peer switch, as the connected links would've simply gone down, and the ports would've dropped out of the forwarding LACP bundles.

November 2, 2020 14:43 UTC

- *Partial* switch failure
  - Link Aggegation Control Protocol working
  - Border Gateway Protocol working
  - Virtual Port control (vPC) *not* working
  - Data Plane dropping packets



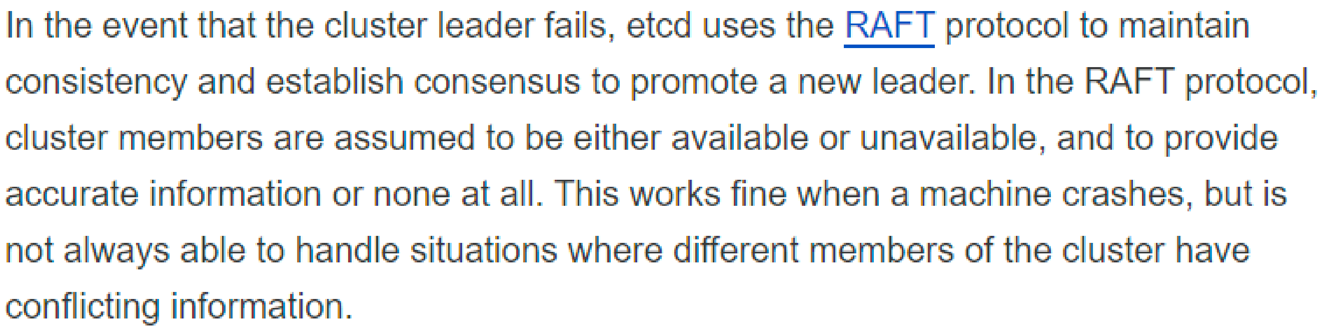
Six minutes later, the switch recovered without human intervention. But this odd failure mode led to further problems that lasted long after the switch had returned to normal operation.

Cloudflare failure

November 2, 2020 14:40 UTC

etcd begins exhibiting errors

- Uses the *Raft* consensus protocol (coming soon to a lecture near you!)
- Experiences a *Byzantine* fault

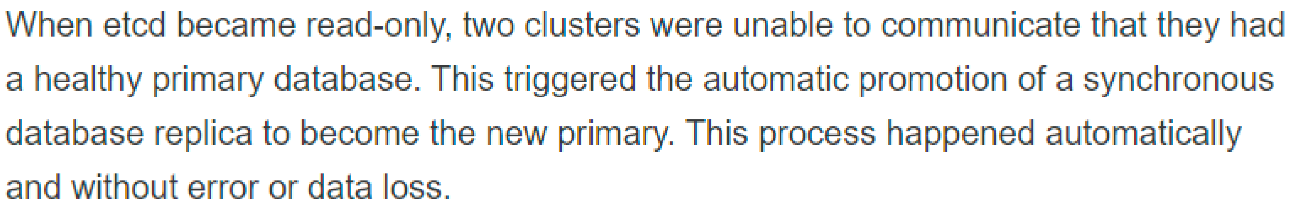


In the event that the cluster leader fails, etcd uses the RAFT protocol to maintain consistency and establish consensus to promote a new leader. In the RAFT protocol, cluster members are assumed to be either available or unavailable, and to provide accurate information or none at all. This works fine when a machine crashes, but is not always able to handle situations where different members of the cluster have conflicting information.

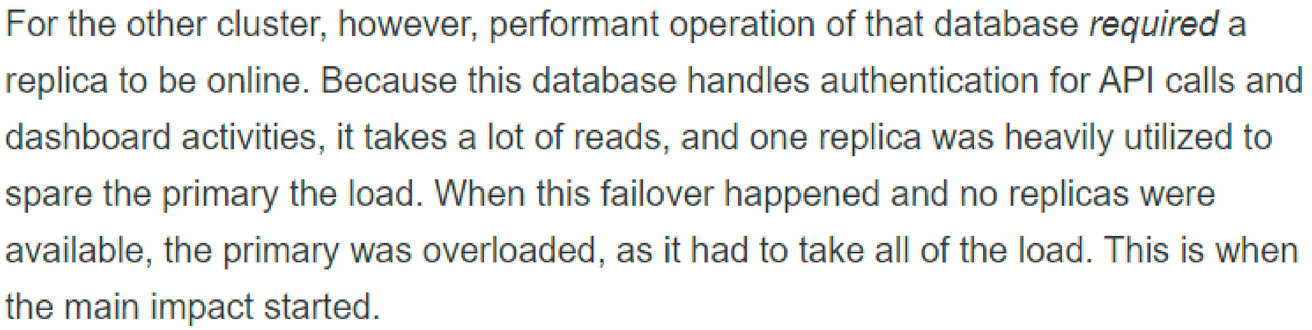
Cloudflare Failure

November 2, 2020 14:45 UTC

Database system promotes a new primary database



When etcd became read-only, two clusters were unable to communicate that they had a healthy primary database. This triggered the automatic promotion of a synchronous database replica to become the new primary. This process happened automatically and without error or data loss.



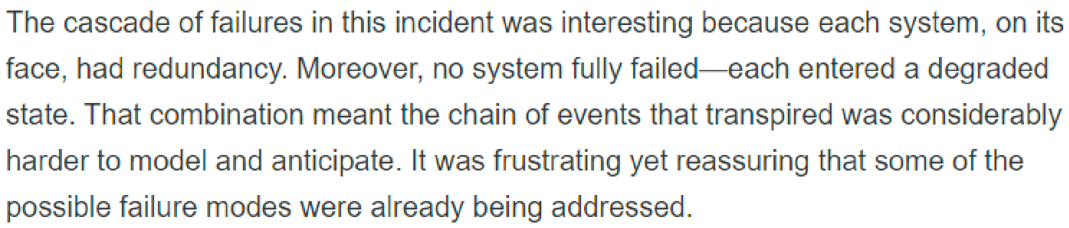
For the other cluster, however, performant operation of that database *required* a replica to be online. Because this database handles authentication for API calls and dashboard activities, it takes a lot of reads, and one replica was heavily utilized to spare the primary the load. When this failover happened and no replicas were available, the primary was overloaded, as it had to take all of the load. This is when the main impact started.

Cloudflare Failure

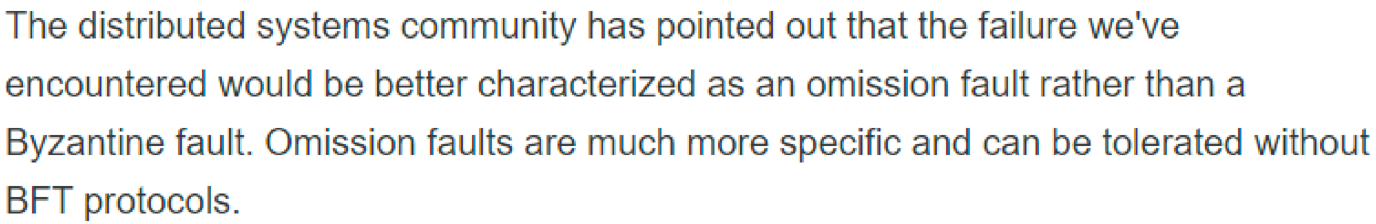
November 2, 2020 21:20 UTC

Database Replica Rebuilt

End of service disruption



The cascade of failures in this incident was interesting because each system, on its face, had redundancy. Moreover, no system fully failed—each entered a degraded state. That combination meant the chain of events that transpired was considerably harder to model and anticipate. It was frustrating yet reassuring that some of the possible failure modes were already being addressed.



The distributed systems community has pointed out that the failure we've encountered would be better characterized as an omission fault rather than a Byzantine fault. Omission faults are much more specific and can be tolerated without BFT protocols.

Lesson Goals

Replication

Primary-backup

Chain-replication

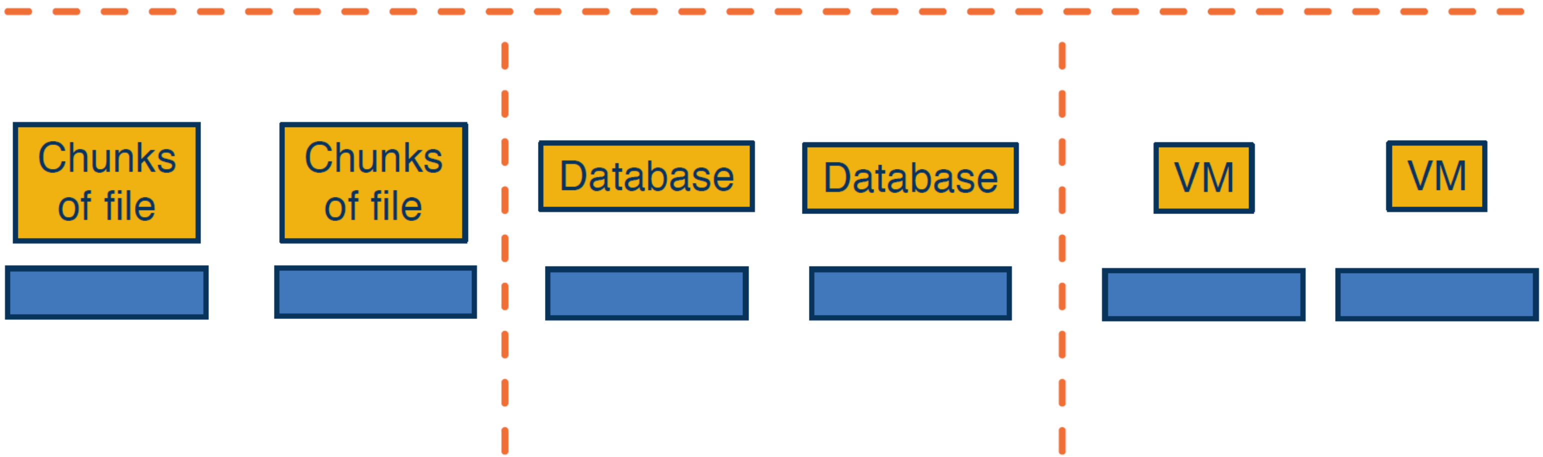
CRAQ

Replication Goal

State available from at least two nodes

Services available from at least two nodes

No single node failure should disrupt service

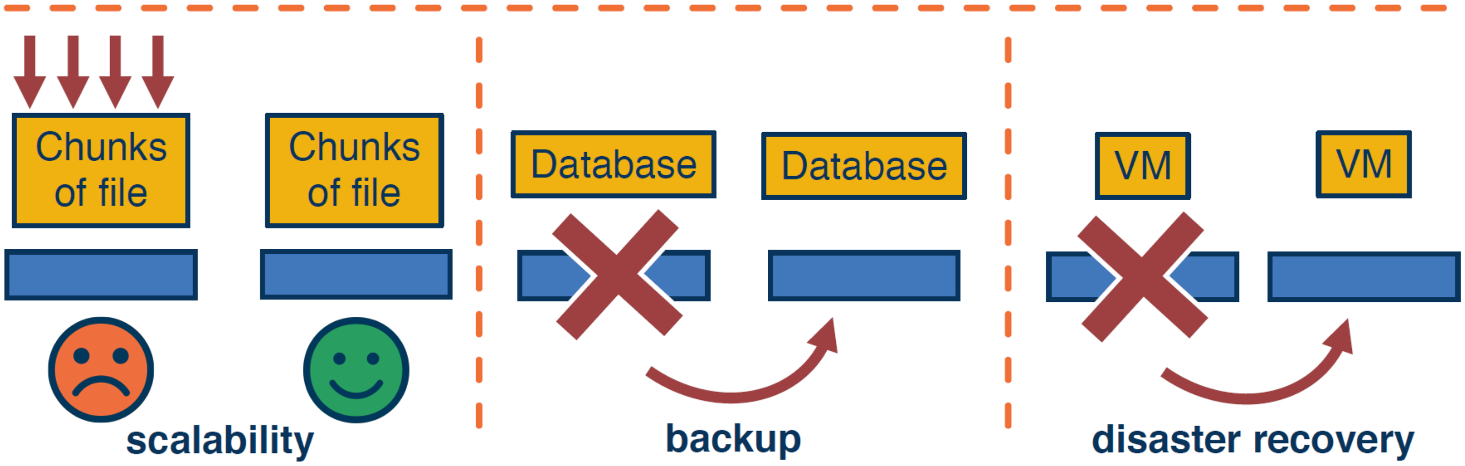


Replication Benefits

Fault-tolerance

Availability

Scalability (load balancing)

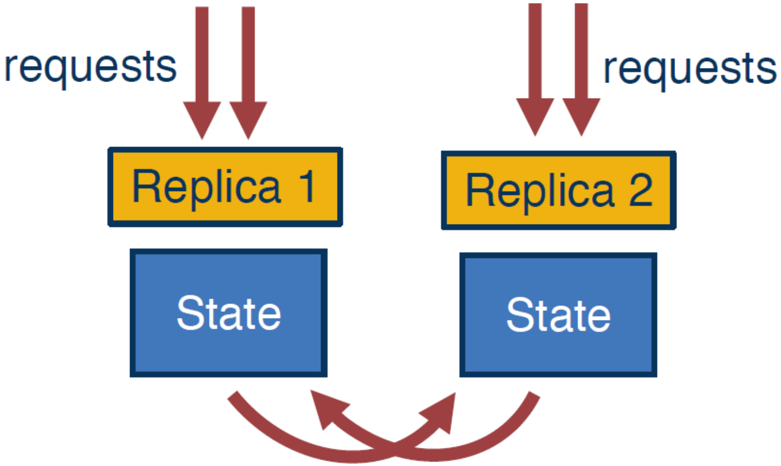


Replication Models

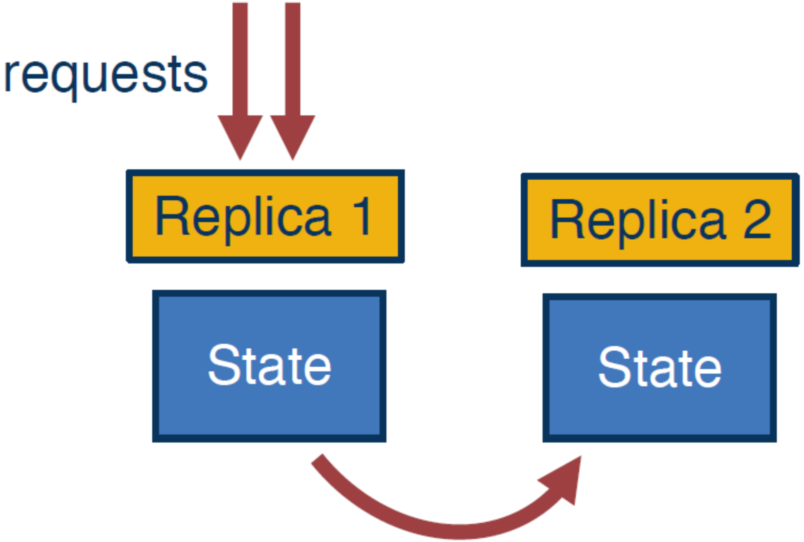
Active Replication

Each active replica

- Can serve read requests
- Can ensure update replication



Standby Replication



Primary-backup

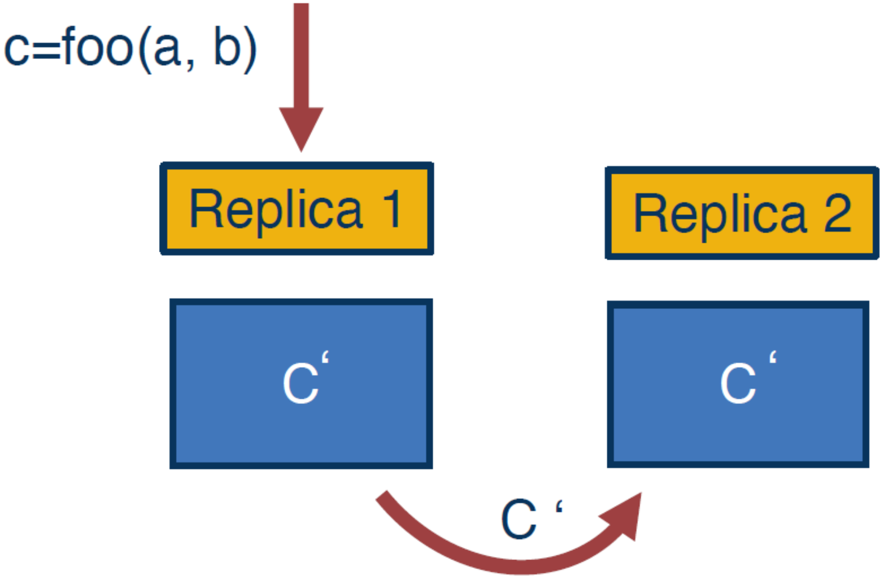
Single active replica (primary) at a time

- Backup remains consistent
- Fast fail-over

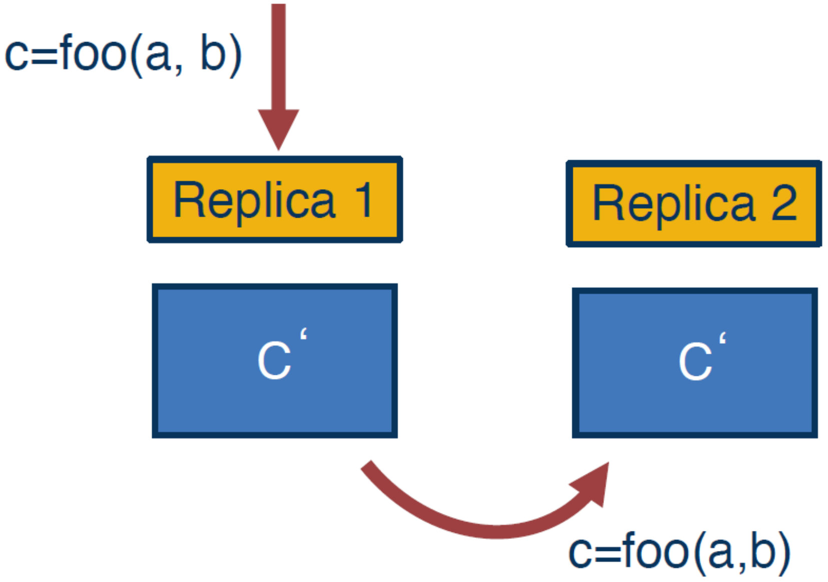
State Replication

Updates execute on one replica

Changes copied from to other replica



Replicated State Machine



Operation logs

- Copied to each replica
- Executed in each location
- Deterministic

State Replica versus Replicated State Machine

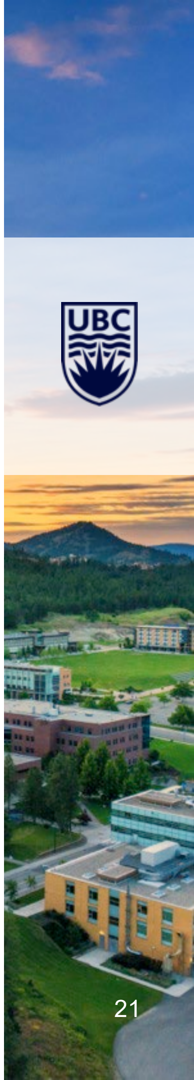
State Replication	Replicated State Machine
Single execution	Copy operations to each replica Execute: must produce same results
State might be large and/or scattered	Log operations are typically smaller
	Determinism required
	Must re-execute

State Replication versus Replicated State Machine

Either can be used for active or primary-backup replication

Choice depends upon:

- Performance
- Efficiency
- Other software engineering tradeoffs



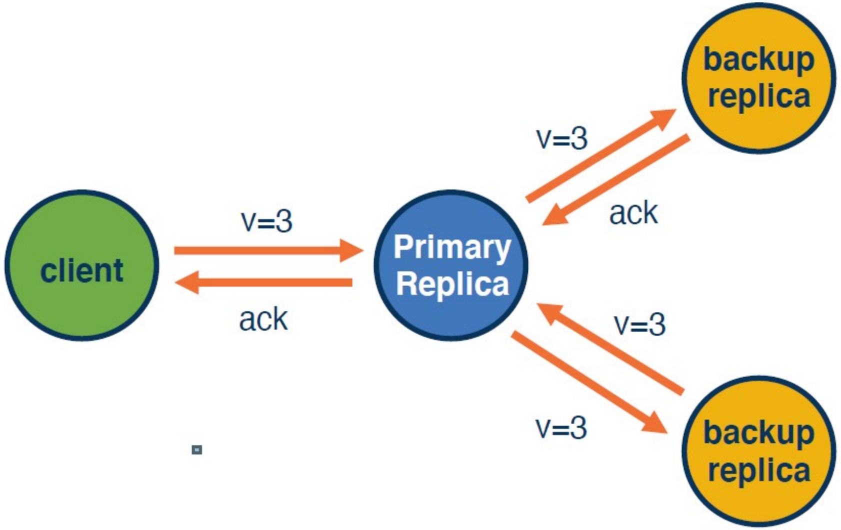
Consensus in a Replicated System

Must ensure *consensus* for updated to replicas

- State Replication *or* Replicated State Machine
- Primary-backup: primary is *coordinator* (or leader)
- Active replication: each replica *can* be leader

Ordering/Visiblity of Updates

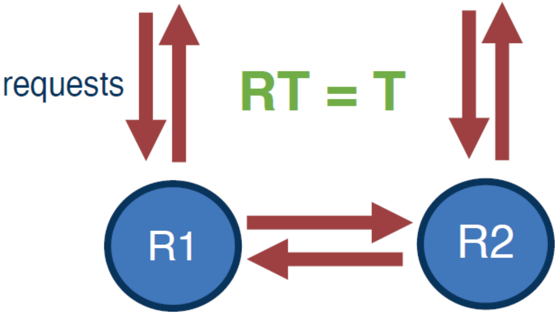
- Depends on consistency model (future lesson)
- Update granularity (objects or transactions)

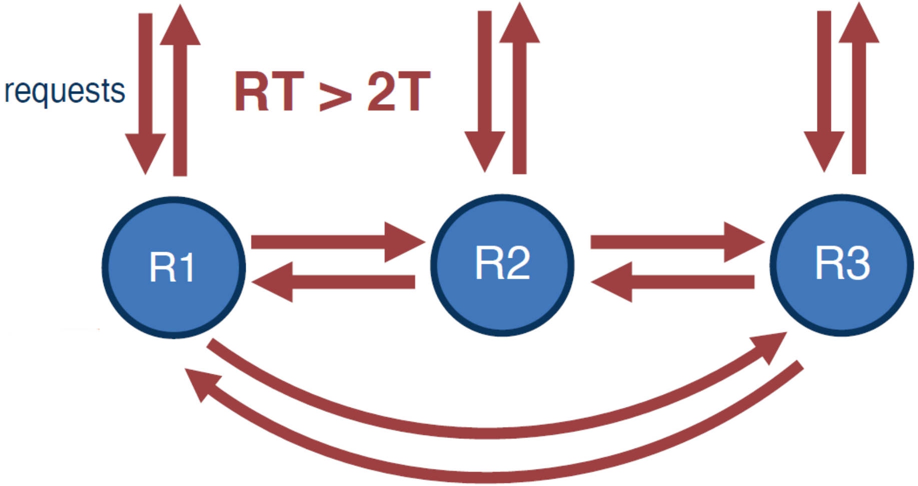


Chain Replication

Messages are *chain replicated* between nodes

- Minimum two message rounds
- $O(n)$ increase in cost for additional replicas

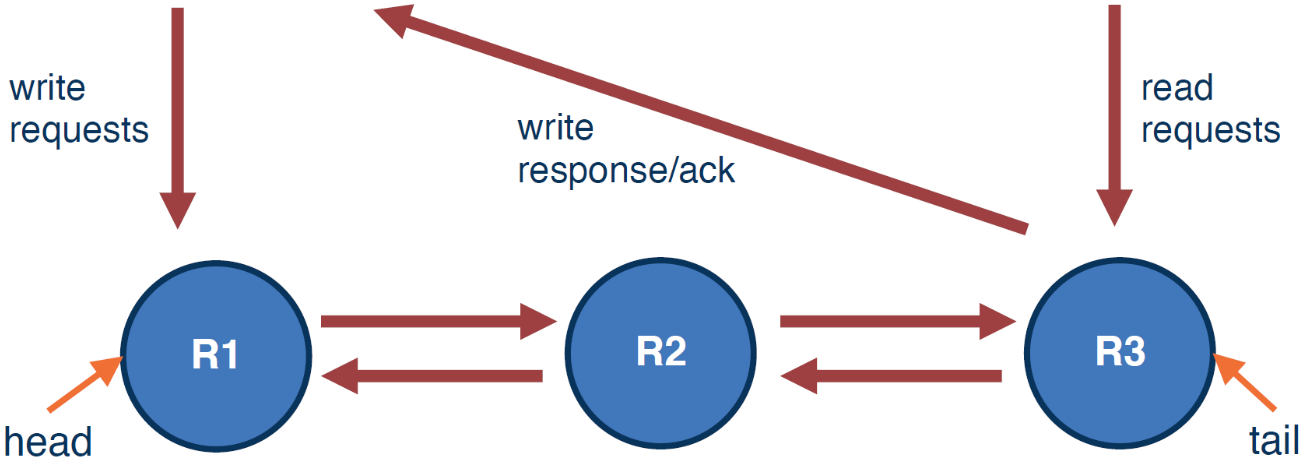




Can we do better?

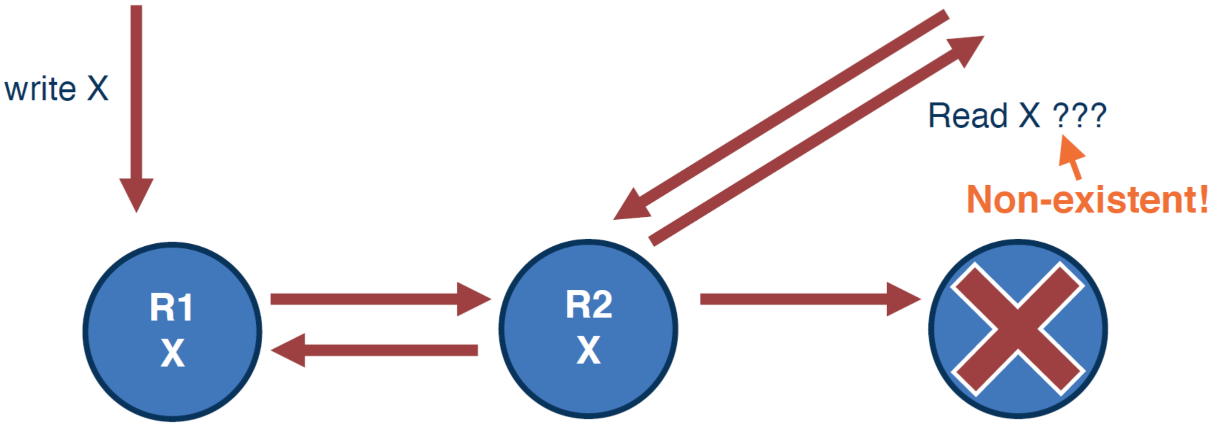
Chain Replication

Published in 2014 by van Renesse and Schneider (OSDI 2014)



Chain Replication

Requires "read from tail" to ensure correctness



Chain Replication: Pros and Cons

Benefits

Leader Scalability

- Decreased messages per replication

High write throughput

- Parallel writes

Strong consistency

- Reads only return committed writes

Limitations

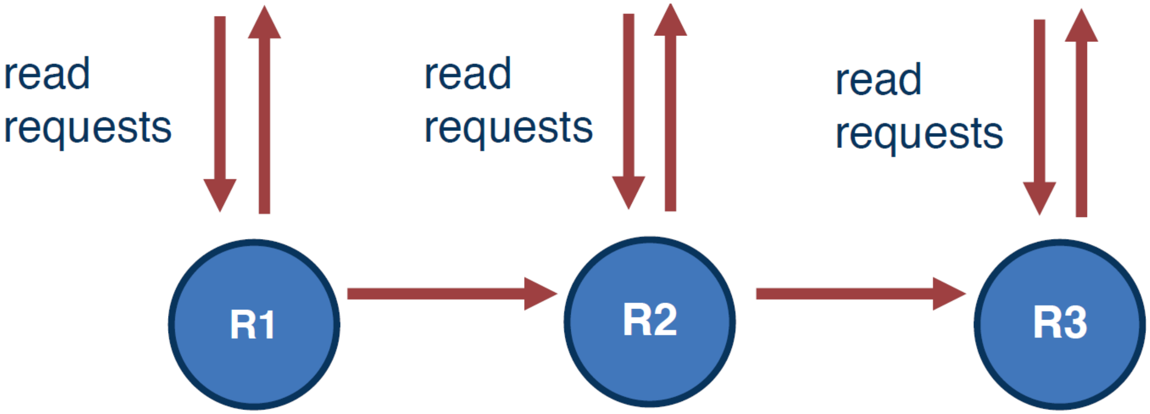
Read-only workloads are *common*

Intermediate nodes typically underutilized

Chain Replication with Apportioned Queries

How to improve on Chain Replication?

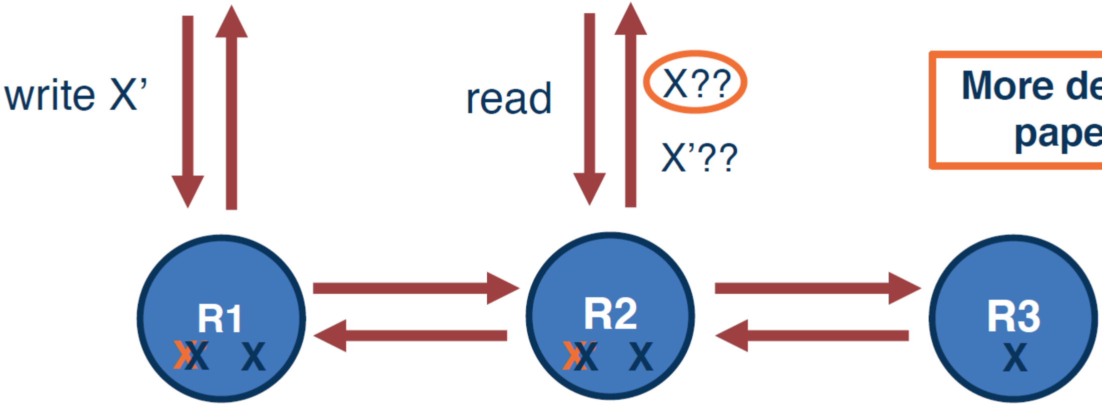
Object storage on CRAQ: high-throughput chain replication for read-mostly workloads, Terrace & Freedman, USENIX ATC 2009.



Chain Replication with Read Scalability

Correctness: maintains both old *and* new data versions

If both values present, tail confirms "most recent"



More detail in paper…

CRAQ versus Chain Replication: Scalability

CR versus CRAQ read throughput

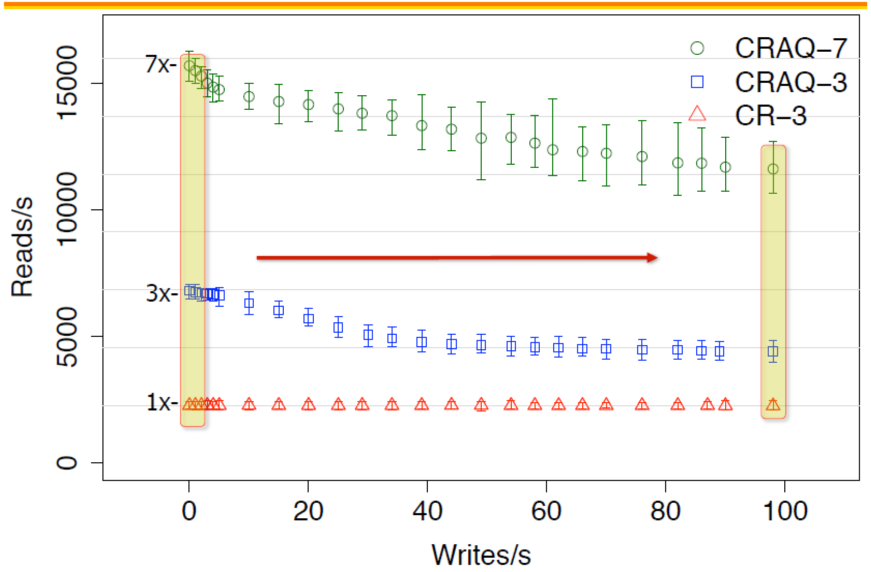
Experiment:

- 3 or 7 replicas in chain
- Up to 100 writes/sec @ head
- CR reads tail
- CRAQ distributes reads

Observations:

- CRAQ provides *better* read throughput
- CRAQ scales with increased number of replicas

Read Throughput as Writes Increase



Lesson Summary

Replication

Active and standby (“fail over” or “primary-backup”) replication

State replication

State *machine* replication

Chain replication

CRAQ

Replication

Best choice depends on

- Workload
    - Reads
    - Writes
- System Configuration
    - Node count
    - Distribution
    - Network properties
- Consistency Requirements
    - Failures
    - Fault tolerance methods

Questions?

THE UNIVERSITY OF BRITISH COLUMBIA