Object Storage on CRAQ

Presented by Jiaxiao Zhou (周佳孝 Mossaka)

Microsoft

So far, we have seen...

Three Replication Techniques!

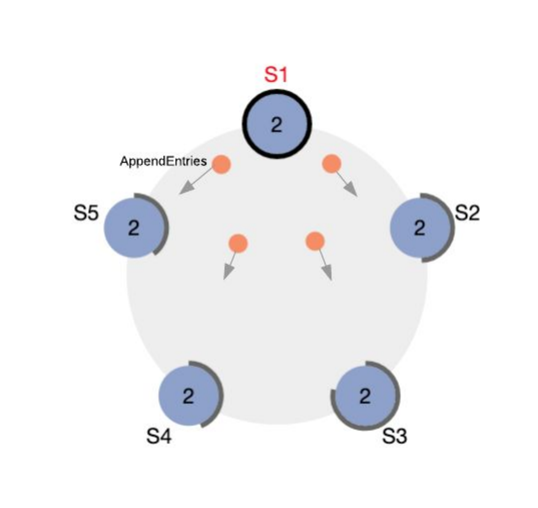
Can you name them?

1. Primary backup replication (GFS, VMware FT)
2. Consensus Protocol (Raft, Zookeeper)
3. Chain Replication (CR, CRAQ)

Motivation

how does RAFT WORK?
RAFT IS A CONSENSUS ALGORITHM
used in many projects in the real world – ex. etcd
a process for getting multiple machines to "agree"!
- RAFT doesn't stand for anything. It's a bunch of logs tied together.
WHY? Any part of a distributed system can fail! Consensus across several nodes gives us backups of data.
WHAT ABOUT NETWORK PARTITIONS?!
In a partition, if the leader falls on the smaller side, a new leader is elected within the majority side. When the partition ends, messages received by the minority side are discarded, and those nodes converge their state to match the majority's.
WHEN A NEW WRITE HAPPENS:
Any node can accept writes!
Message is forwarded to leader node
Message is queued...
Copy me!
ACK!
ACK
...while followers copy it to their logs
FISH
When a quorum have copied the message, leader "commits" the new data & will now return it on read
Followers also commit so they return new data!
@deniseyu21 with @mt165

Motivation



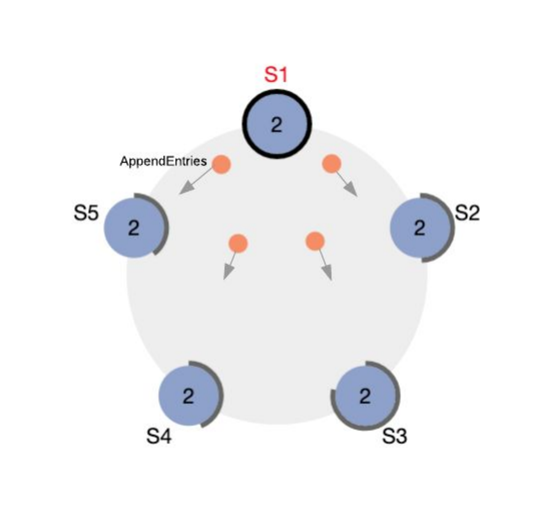
Raft / Paxos

Clients have to send reads to the leader

But no opportunity to divide the read load over the followers.

How can we improve?

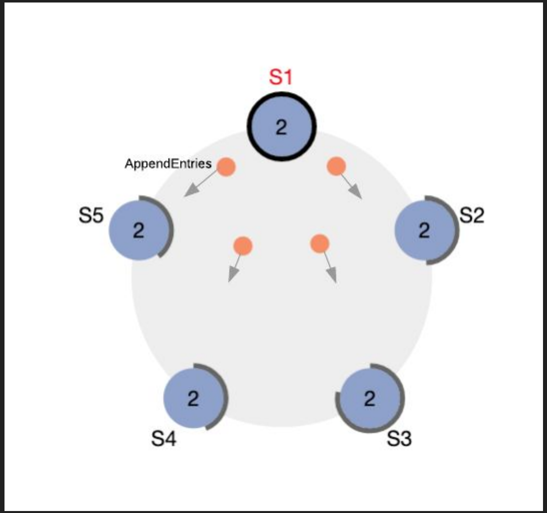
Case Study 1:



Raft / Paxos

"Let's change the definition of correctness!"

Case Study 1:



Raft / Paxos

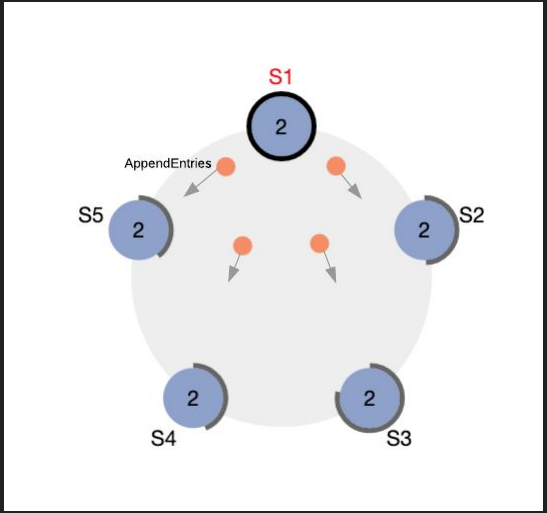
Zookeeper

Linearizable Writes

FIFO client order

But leader is still a bottleneck for write throughputs.

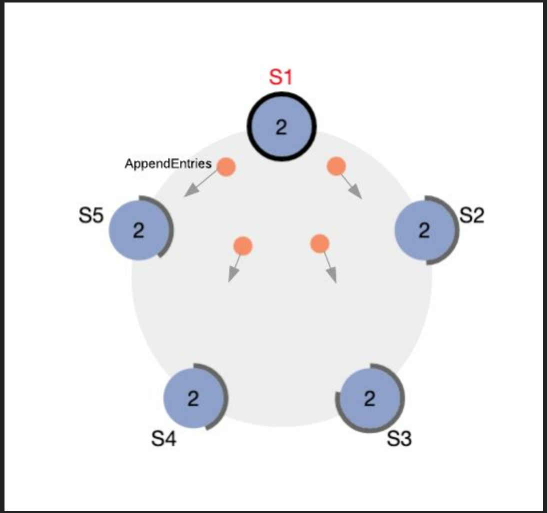
Case Study 2:



Raft / Paxos

We got a new idea!

Case Study 2:



Raft / Paxos

APACHE ZooKeeper™

Zookeeper

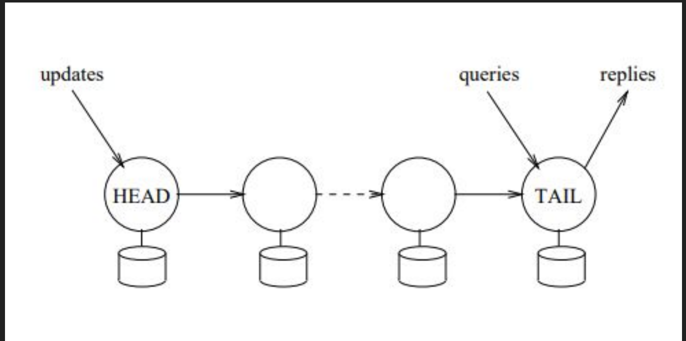
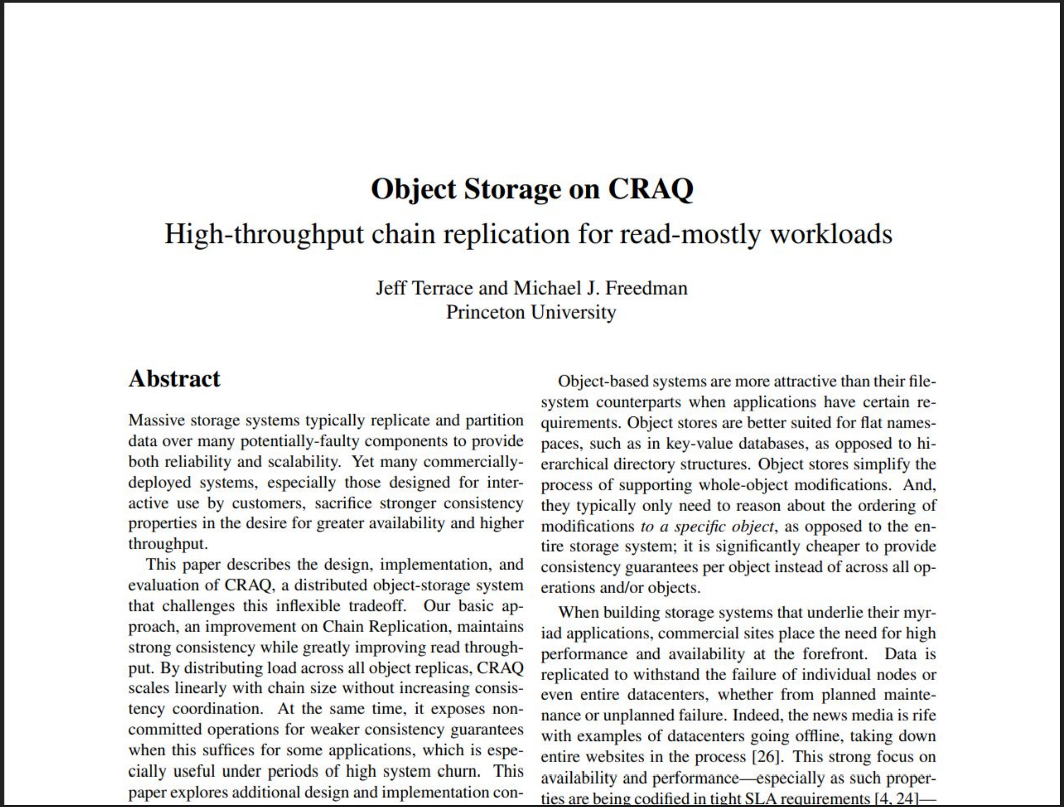


Figure 2: A chain.

Chain Replication

Can we improve read perf to CR as Zookeeper did to Raft?? 🤔🤔

Case Study 3:



Object Storage on CRAQ

High-throughput chain replication for read-mostly workloads

Jeff Terrace and Michael J. Freedman
Princeton University

Abstract

Massive storage systems typically replicate and partition data over many potentially-faulty components to provide both reliability and scalability. Yet many commercially-deployed systems, especially those designed for interactive use by customers, sacrifice stronger consistency properties in the desire for greater availability and higher throughput.

This paper describes the design, implementation, and evaluation of CRAQ, a distributed object-storage system that challenges this inflexible tradeoff. Our basic approach, an improvement on Chain Replication, maintains strong consistency while greatly improving read throughput. By distributing load across all object replicas, CRAQ scales linearly with chain size without increasing consistency coordination. At the same time, it exposes non-committed operations for weaker consistency guarantees when this suffices for some applications, which is especially useful under periods of high system churn. This paper explores additional design and implementation con-

Object-based systems are more attractive than their file-system counterparts when applications have certain requirements. Object stores are better suited for flat namespaces, such as in key-value databases, as opposed to hierarchical directory structures. Object stores simplify the process of supporting whole-object modifications. And, they typically only need to reason about the ordering of modifications *to a specific object*, as opposed to the entire storage system; it is significantly cheaper to provide consistency guarantees per object instead of across all operations and/or objects.

When building storage systems that underlie their myriad applications, commercial sites place the need for high performance and availability at the forefront. Data is replicated to withstand the failure of individual nodes or even entire datacenters, whether from planned maintenance or unplanned failure. Indeed, the news media is rife with examples of datacenters going offline, taking down entire websites in the process [26]. This strong focus on availability and performance—especially as such properties are being codified in tight SLA requirements [4, 24]—

Google Cloud Platform

What are the claims from CRAQ

1. Strong consistency! 🙌 (unlike Zookeeper)
2. Greatly improving read throughput
3. This means CRAQ is strictly more powerful than CR

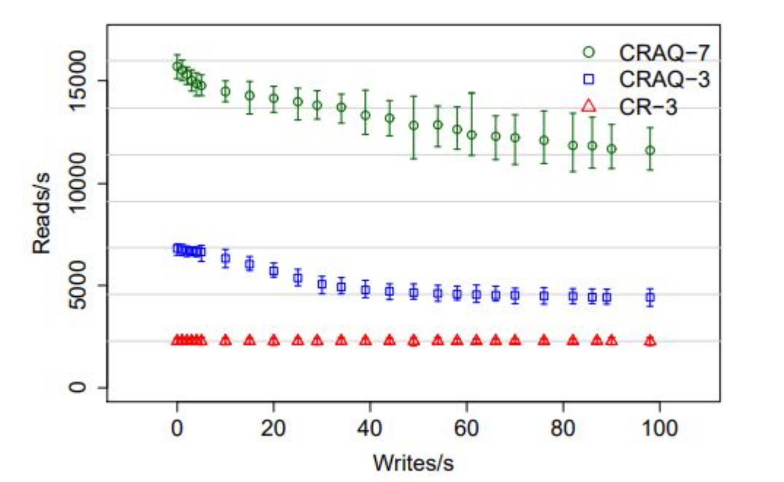


Figure 7: Read throughput as writes increase (5KB object).

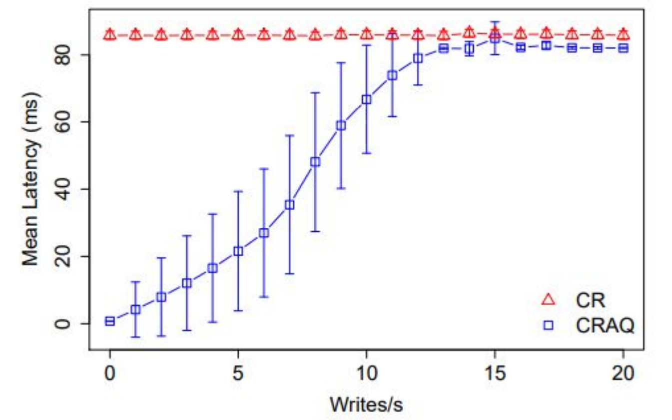


Figure 13: CR and CRAQ's read latency to a local client when the tail is in a distant datacenter separated by an RTT of 80ms and the write rate of a 500-byte object is varied.

What is object storage?

1. Best suited for unstructured data (files, music, photos, movies)
2. Support 2 primitive operations
   a. *Read* (return data block stored under object name)
   b. *Write* (change the state of a single object)
3. Flat namespaces (key-value)
4. Supporting whole object modifications
5. Significantly cheaper to provide consistency

Strong vs Eventual consistency

1. Strong consistency = linearizability = a read to an object always sees the latest written value
2. Eventual consistency:
   a. Writes are linearizable
   b. Reads can return stale data
   c. When all replicas receive the write, read can never return older version
   d. Monotonic read consistency for one session.

Some of the most well-known key-value databases

Dynamo: Amazon's Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels

Amazon.com

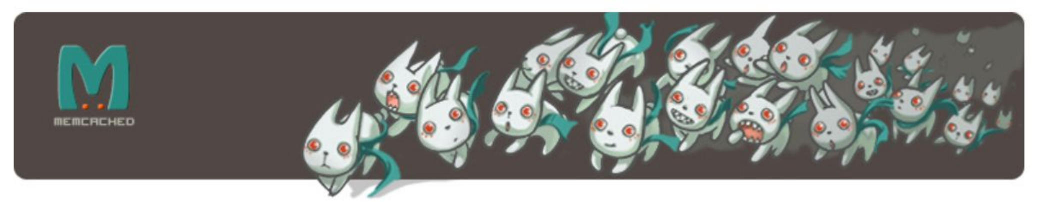
ABSTRACT

Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters around the world. At this scale, small and large compon...

Cassandra - A Decentralized S... rage System

cassandra

Sacrifice strong consistency due to tight SLA requirements



Chain Replication (CR)

- Strong consistency ✅
- Good throughput ✅
- Easy recovery ✅
- Tail hotspots ❌

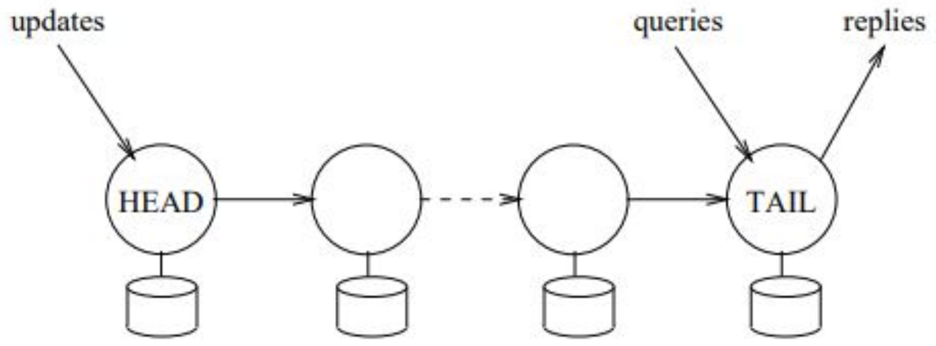


Figure 2: A chain.

Chain Replication with Apportioned Queries (CRAQ)✨✨

Strong consistency ✅

Good throughput ✅

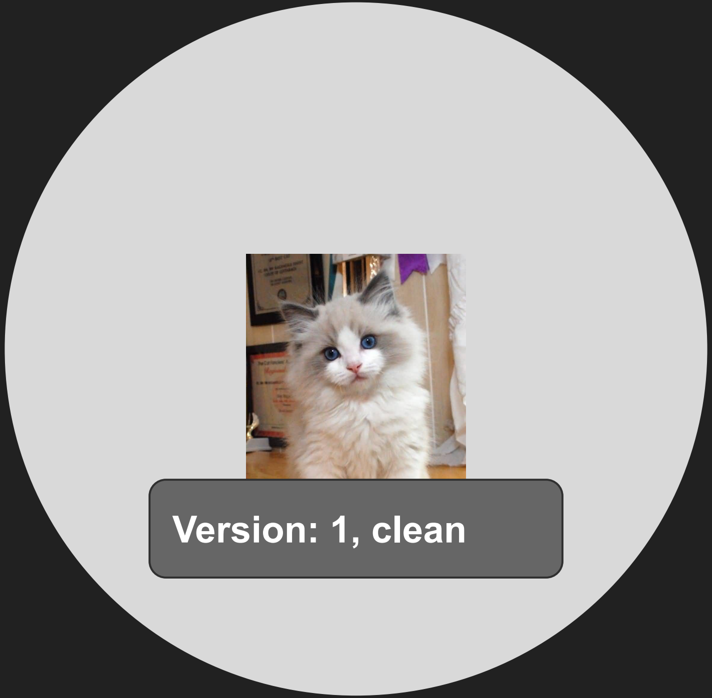
Easy recovery ✅

Lower latency ✅

High throughput for read ✅

Basic System Model

Let’s start with a node storing a Cat pic in a chain.



Basic System Model

Receives a new version of a cat pic.

Non-tail: marks dirty
Tail:
1. marks clean,
2. commit,
3. send **ACK** to all other nodes

Basic System Model

ACK arrived, the node marks the version as clean

Basic System Model

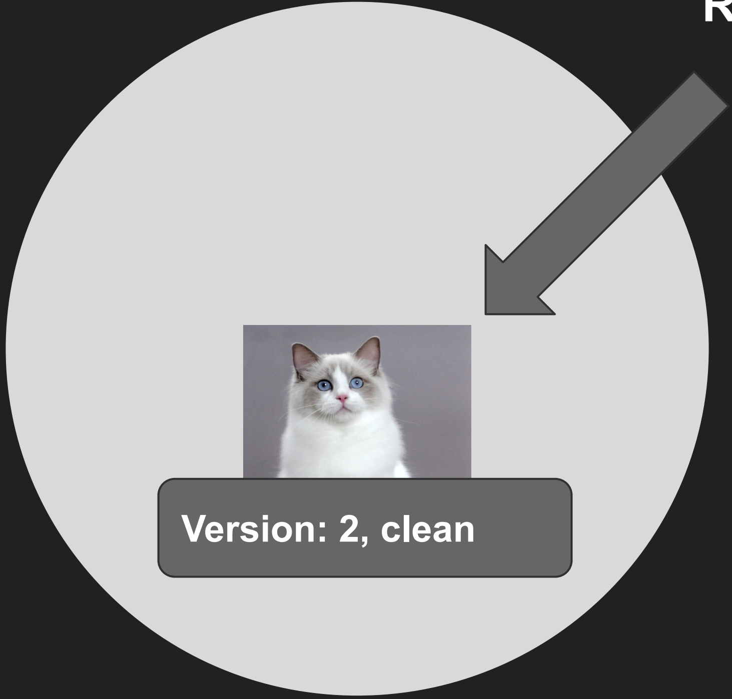
ACK arrived, the node marks the version as clean

Delete all prior versions

Basic System Model

Basic System Model

Receives read



If the latest version is clean: return the cat picture

If dirty: The node makes a version query, return that version of the cat picture.

Read ops are serialized w.r.t the tail.

Basic System Model (from paper)

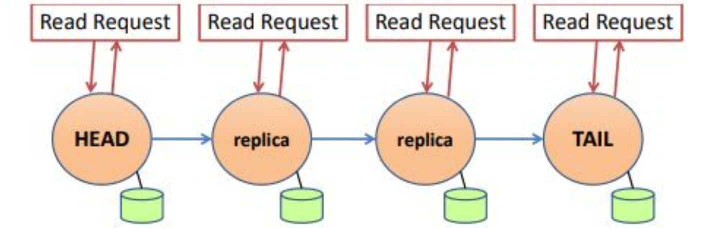


Figure 2: **Reads to clean objects in CRAQ can be completely handled by any node in the system.**

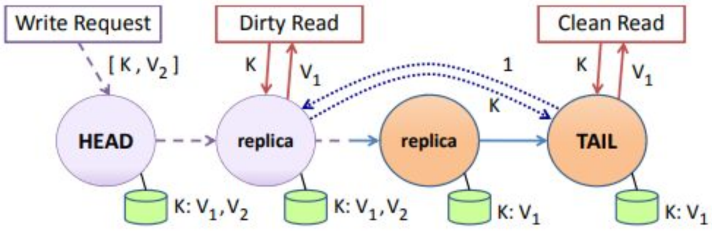


Figure 3: **Reads to dirty objects in CRAQ can be received by any node, but require small version requests (dotted blue line) to the chain tail to properly serialize operations.**

How does CRAQ improve throughput?

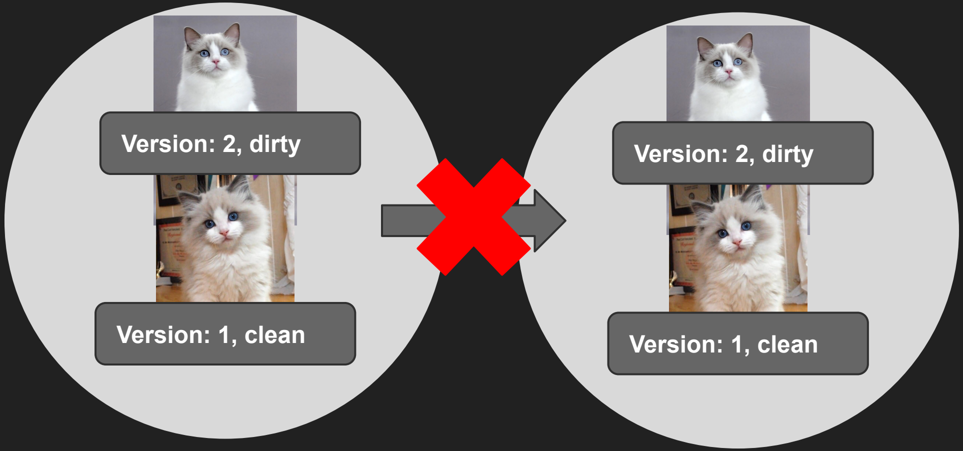
Read-Mostly Workloads: most reads handled by C-1 non-tail nodes. Throughput scales linearly with chain size C

Write-Heavy Workloads: non-tail nodes make lighter-weight version queries for reads. Still a big win over CR.

One could optimize read throughput by having the tail node only handle version queries. 🤯🤯

Split-brain problem 🧠🧠

If the head node and the node next to head cannot talk to each other:



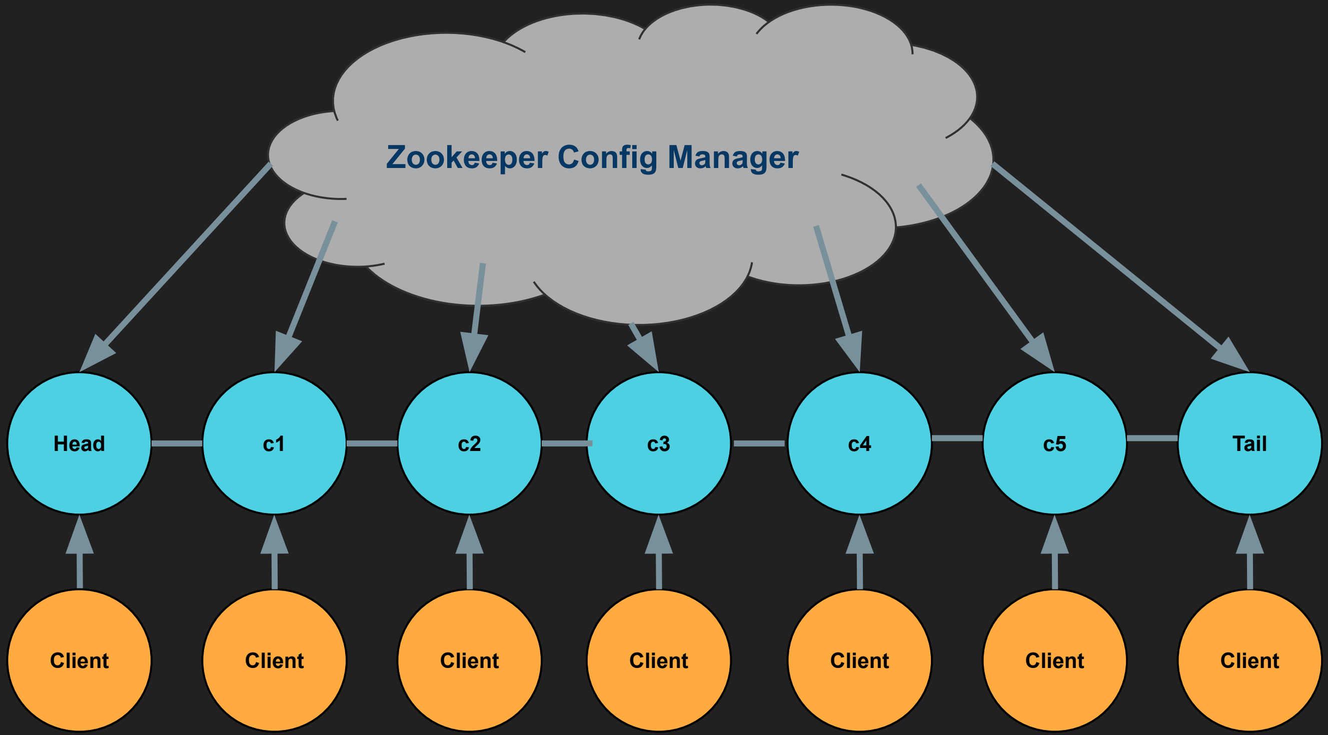
Can the node next to head claim itself as the new head?

Split-brain problem

If the head node and the node next to head cannot talk to each other:

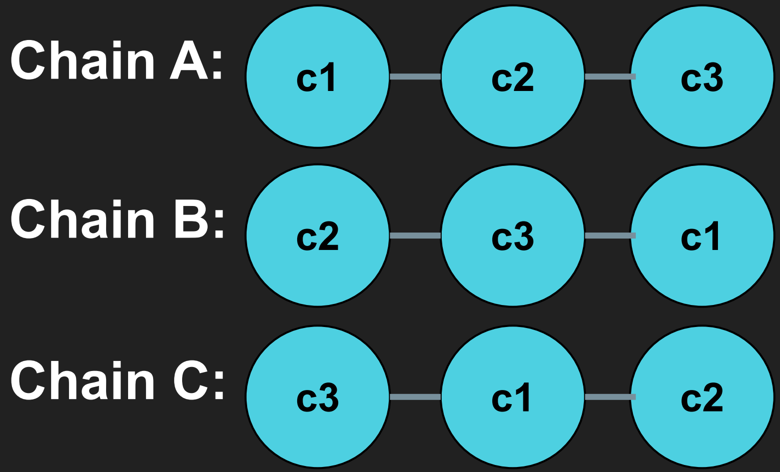
No!

Split-brain problem 🧠🧠



Other solution to reduce tail's load

Multi-chain:



Topics not covered

Scaling CRAQ:

1. Implicit data centers & global chain size
2. Explicit data centers & global chain size
3. Explicit data center chain sizes

Extensions:

1. Mini-transactions on CRAQ
2. Handling memberships changes (using Zookeeper)

Questions❓