Chain Replication with Apportioned Queries

Jeff Terrace and Michael J. Freedman, 2009

Problem

Design a **durable object storage system** with **strong consistency** guarantees**.** Bonus points awarded for **high read throughput.**

Definitions

- Object storage
  - write(key,val)
  - read(key)
- Strong consistency
  - All reads/writes happen at an instant in time between client request and acknowledgement
- Fail-stop servers
  - Dead servers do no wrong
  - Everyone can detect a dead server

Warmup: Primary/Backup Replication

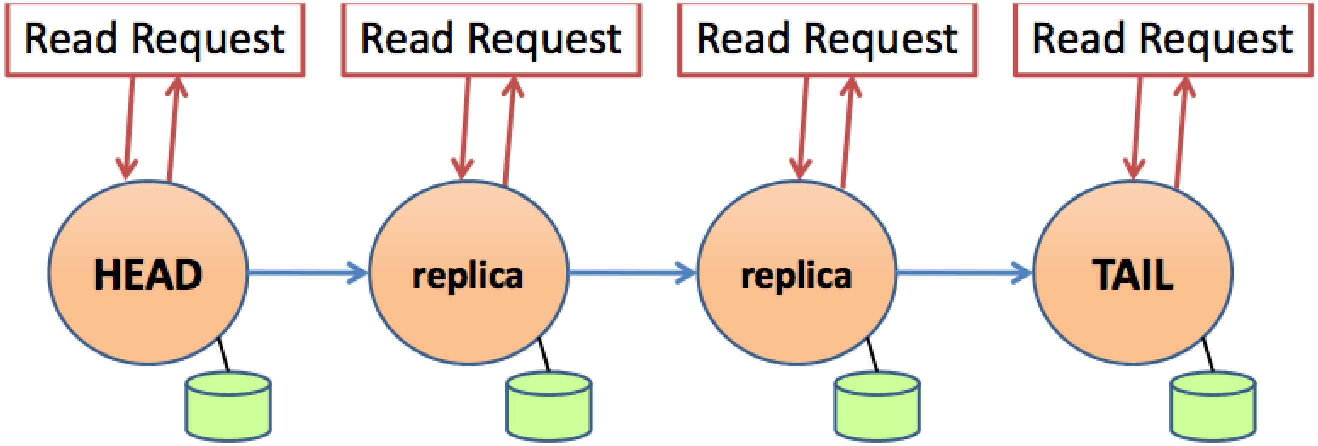
- Primary server handles all read/write requests
- On write request, primary first sends replicas to backup servers, waits for acknowledgement, then commits write.
- On primary fail
  - Messy! Need to synchronize state between all backups

Room for Improvement

- Backups can't be used to handle reads
  - Fix: Send ACKNOWLEDGE messages to backups signifying that a given write is complete
- Recovery from primary failure requires n-way synchronization
  - Fix: Impose a **linear order** on backups so that backup i is ahead of backup i+1

CRAQ: Clean Reads

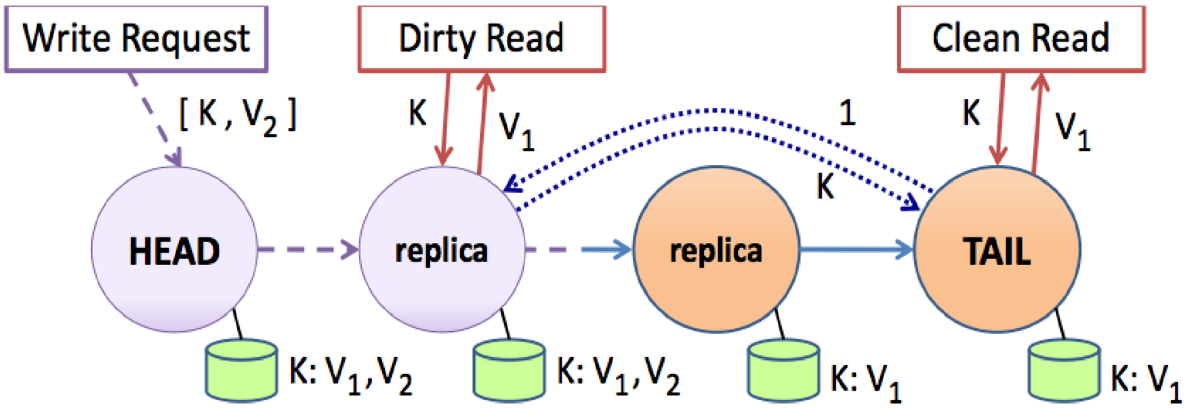
- When no writes are in progress, any node can handle reads



CRAQ: Writes

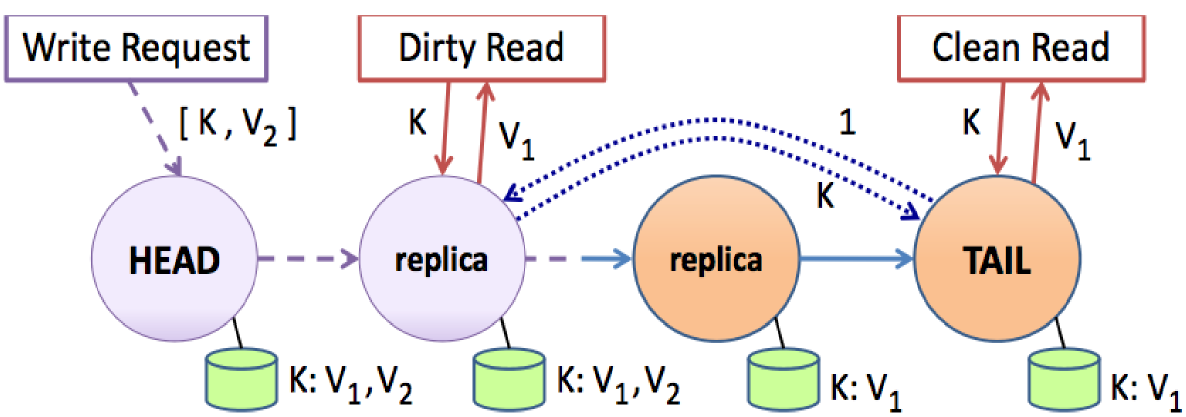
On write(key,val),

1. Head propagates PREPARE(key,val,version#) message down the chain.
2. Tail receives PREPARE and commits write.
3. Tail propagates ACKNOWLEDGE(key,version#) back up the chain
4. Upon receipt of ACKNOWLEDGE(key,version#), a node may commit the corresponding new object version.



CRAQ: Dirty Reads

- While waiting for an ACKNOWLEDGE, redirect read requests to the tail
- Only a version request is necessary!



CRAQ: Failure Recovery

- Only nodes at the splice site need to synchronize state (cf. primary/backup)
- Reads may continue away from the splice site.

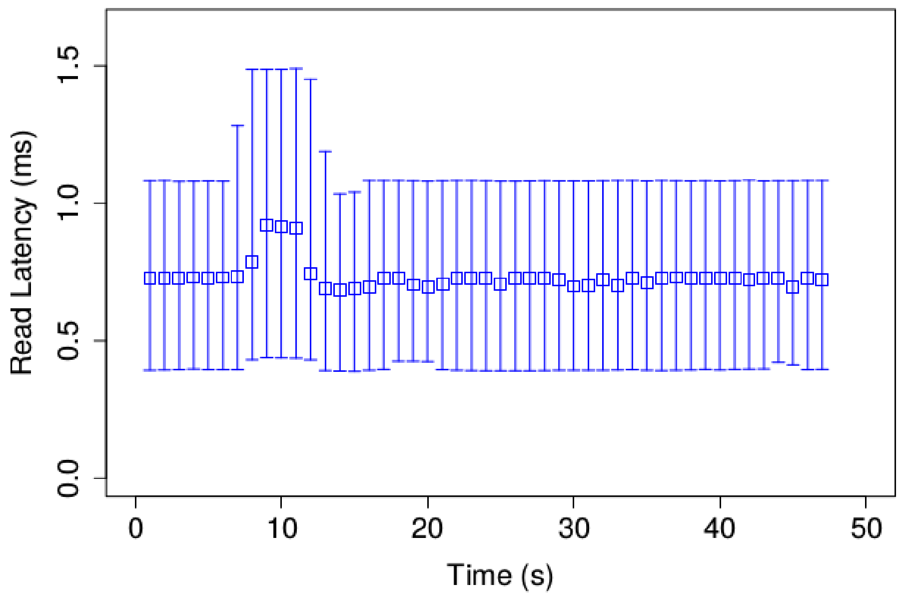


Figure 11: **CRAQ's read latency (shown here under moderate load) goes up slightly during failure, as requests to the failed node need to be retried at a non-faulty node.**

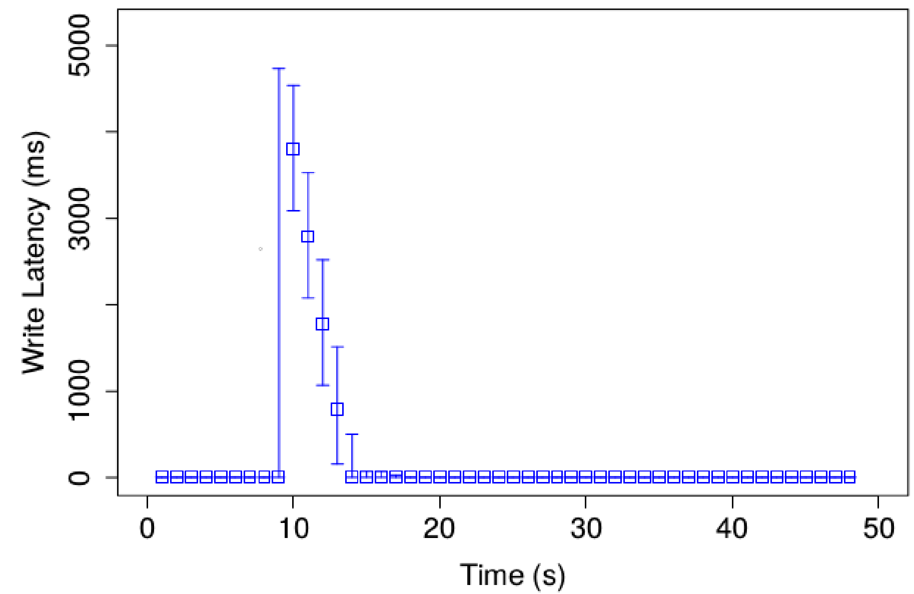


Figure 12: **CRAQ's write latency increases during failure, since the chain cannot commit write operations.**

Chain Layout & Management

- ~10 nodes per chain, ~1000 nodes, ~10000 chains
- 2 levels for chain management
  - Within a datacentre, use consistent hashing to map nodes to chains
  - Across datacentres, use manual layout or consistent hashing
- Make use of a black box coordination service (e.g. ZooKeeper) to maintain chain membership lists and notify nodes of changes

Optimizations and Extensions

- Relaxed consistency
  - Allow dirty reads (eventual consistency)
  - Allow dirty reads with a time limit (bounded inconsistency)
- Broadcast data, propagate metadata
- Broadcast acknowledgements
- Multi-object transactions for objects on the same chain

Experimental Design & Results

- Compare single chain performance of CR and CRAQ on Emulab over a 100MBit network

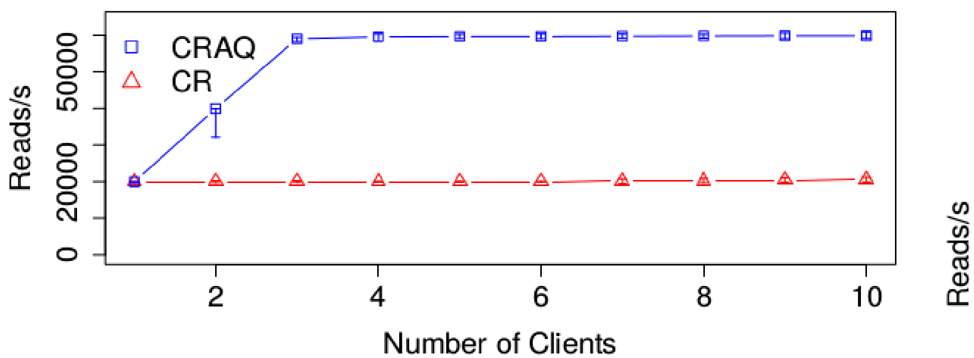


Figure 4: **Read throughput as the number of readers increase: A small number of clients can saturate both CRAQ and CR, although CRAQ's asymptotic behavior scales with chain size, while CR is constant.**

Experimental Design & Results

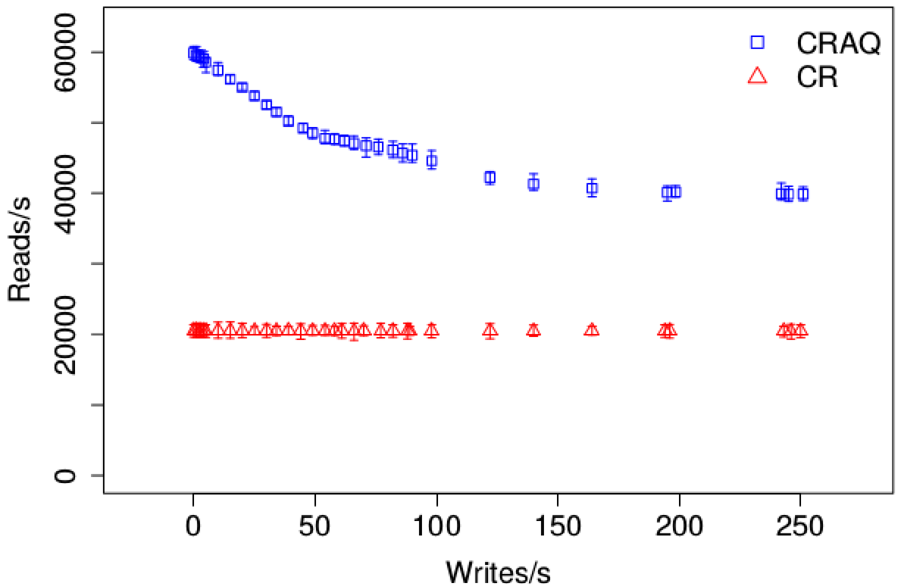


Figure 6: **Read throughput on a length-3 chain as the write rate increases (500B object).**

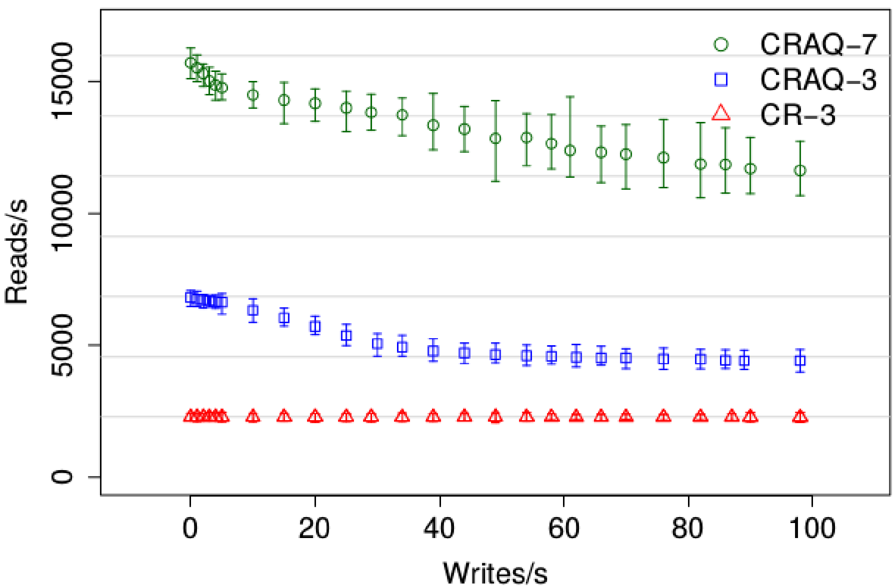


Figure 7: **Read throughput as writes increase (5KB object).**

Experimental Design & Results

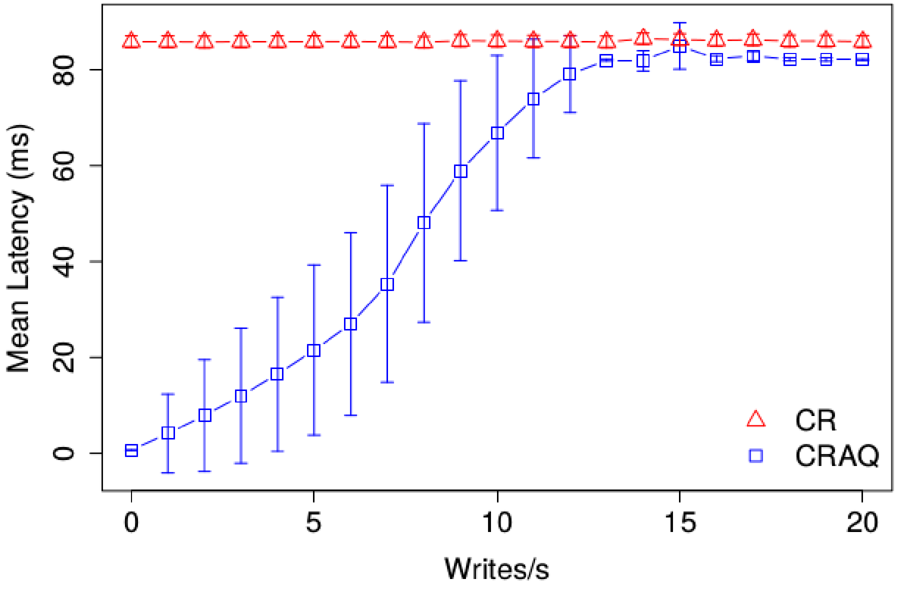


Figure 13: **CR and CRAQ's read latency to a local client when the tail is in a distant datacenter separated by an RTT of 80ms and the write rate of a 500-byte object is varied.**

Strengths

- Maximum read throughput scales with chain length
- Better read locality than CR
- Allows clients to trade strong consistency for higher read throughput
- Recovery protocols are simple and require small amounts of coordination
- Clean reads can continue during failure recovery

Weaknesses

- Write latency scales with chain length
- Limited comparison with other systems
  - Comparison with CR might go differently with multiple chains
  - Should compare with other systems offering similar guarantees
- Fail-stop is a very strong assumption
  - What happens during a network partition? Not even majority can continue writing!

References

- Van Renesse, Robbert, and Fred B. Schneider. "Chain Replication for Supporting High Throughput and Availability." In OSDI, vol. 4, pp. 91-104. 2004.
- Terrace, Jeff, and Michael J. Freedman. "Object Storage on CRAQ: High-Throughput Chain Replication for Read-Mostly Workloads." In USENIX Annual Technical Conference. 2009.