Object Storage on CRAQ

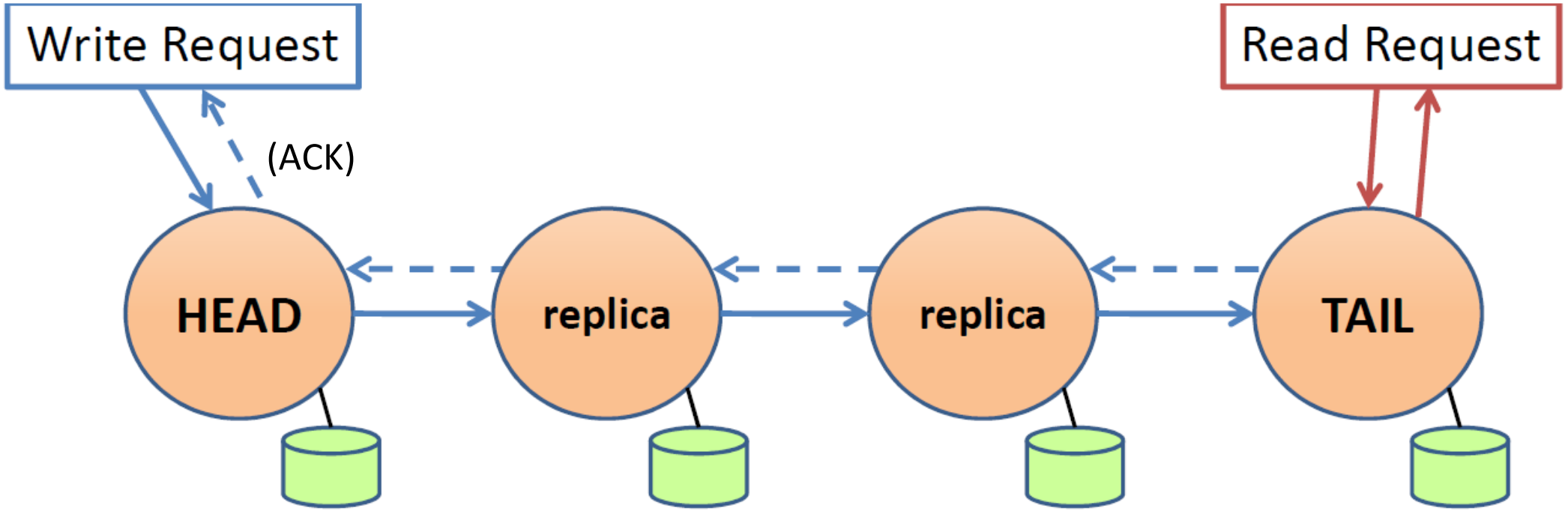
High-throughput chain replication for read-mostly workloads

Jeff Terrace and Michael J. Freedman
Princeton University

Chain Replication (CR)

- Provides **strongly-consistent** object-based storage
  - write(key, value)
  - read(key)
- Simple **chain** topology for nodes to replicate data
  - **Head** accepts and propagates all *writes*
  - **Tail** orders operations and serves all *reads*
  - Writes are **committed** when they reach the tail

CR: Read/Write Operations



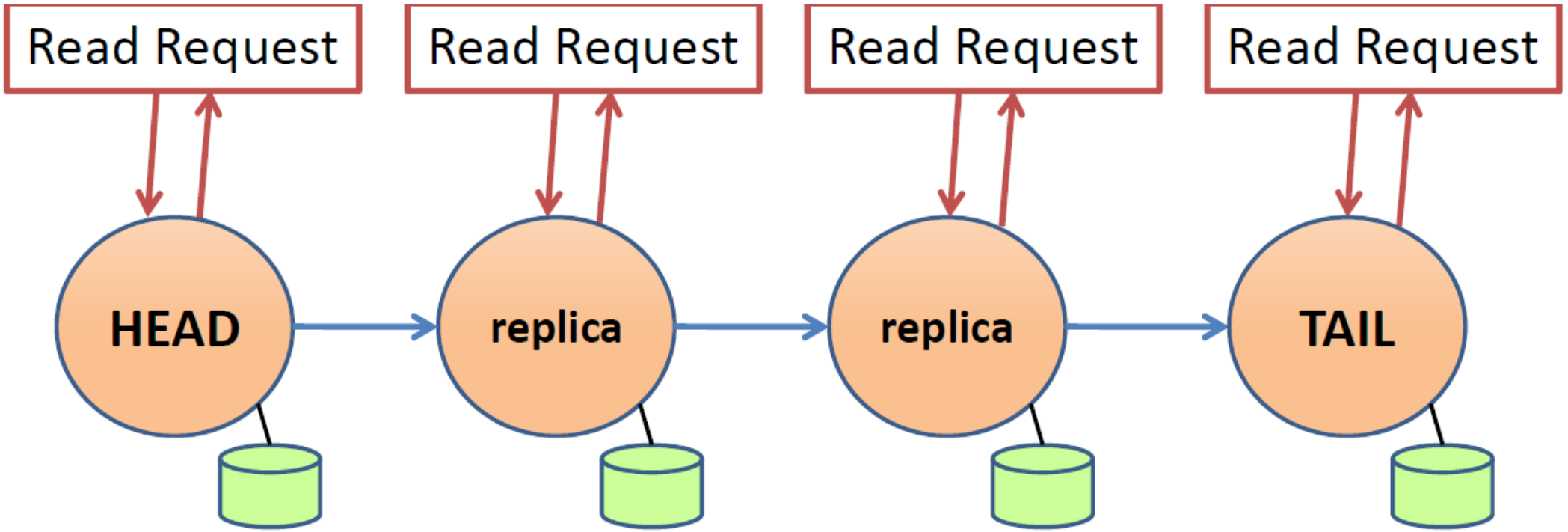
CR: Performance

- Good *write* throughput
  - Pipeline concurrent writes down the chain
- Poor *read* throughput
  - Only tail can handle reads
  - Chains across multiple (distant) datacenters?

Chain Replication with Apportioned Queries (CRAQ)

- Any node can serve *reads*
  - **Load balancing** across all nodes in a chain
  - Better **locality** in cross-datacenter chains
- Offers a choice of consistency models:
  - **Strong** consistency (same as CR)
  - **Eventual** consistency (lower latency, but read potentially uncommitted objects)
  - **Eventual** consistency with **maximum-bounded inconsistency**

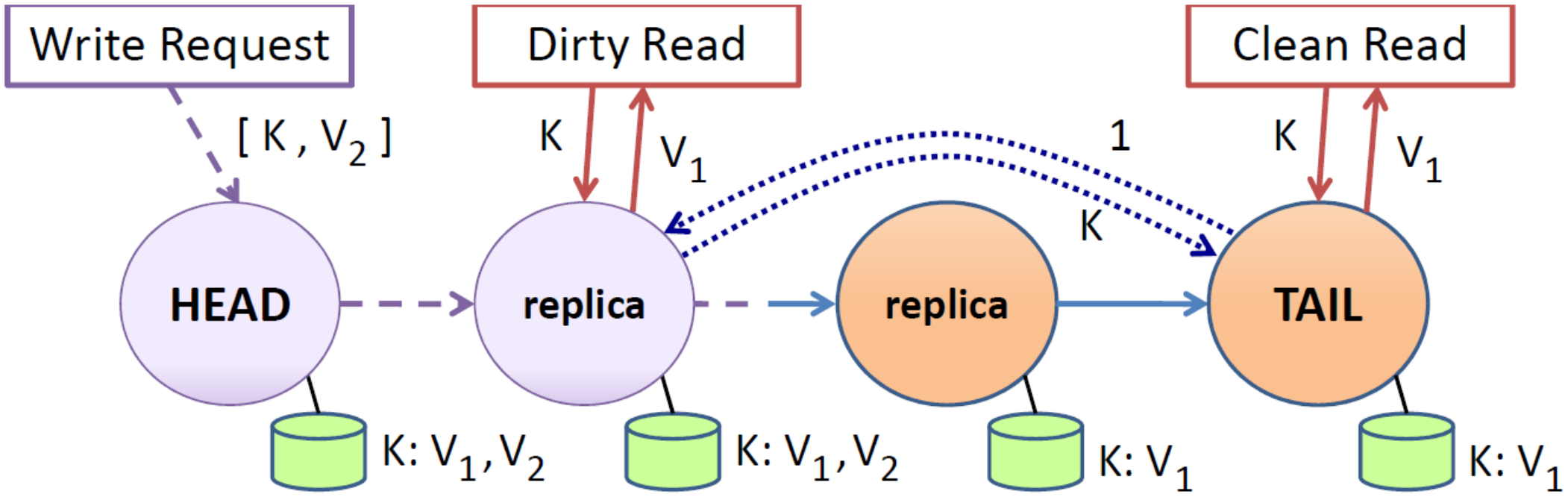
CRAQ: Clean Objects



CRAQ: Dirty Objects

- Nodes can store multiple versions of object, marked as **clean** or **dirty**
- Writes:
  - When propagating an object, store this newest version as *dirty*
  - Tail marks its version as *clean*, and propagates **acknowledgements** back up the chain
  - When receiving an acknowledgement, mark the version as *clean* and **delete** all prior versions
- Reads:
  - If latest known version is *clean*, **return** that value
  - If latest known version is *dirty*, send a **version query** to the tail to get the latest version

CRAQ: Dirty Objects



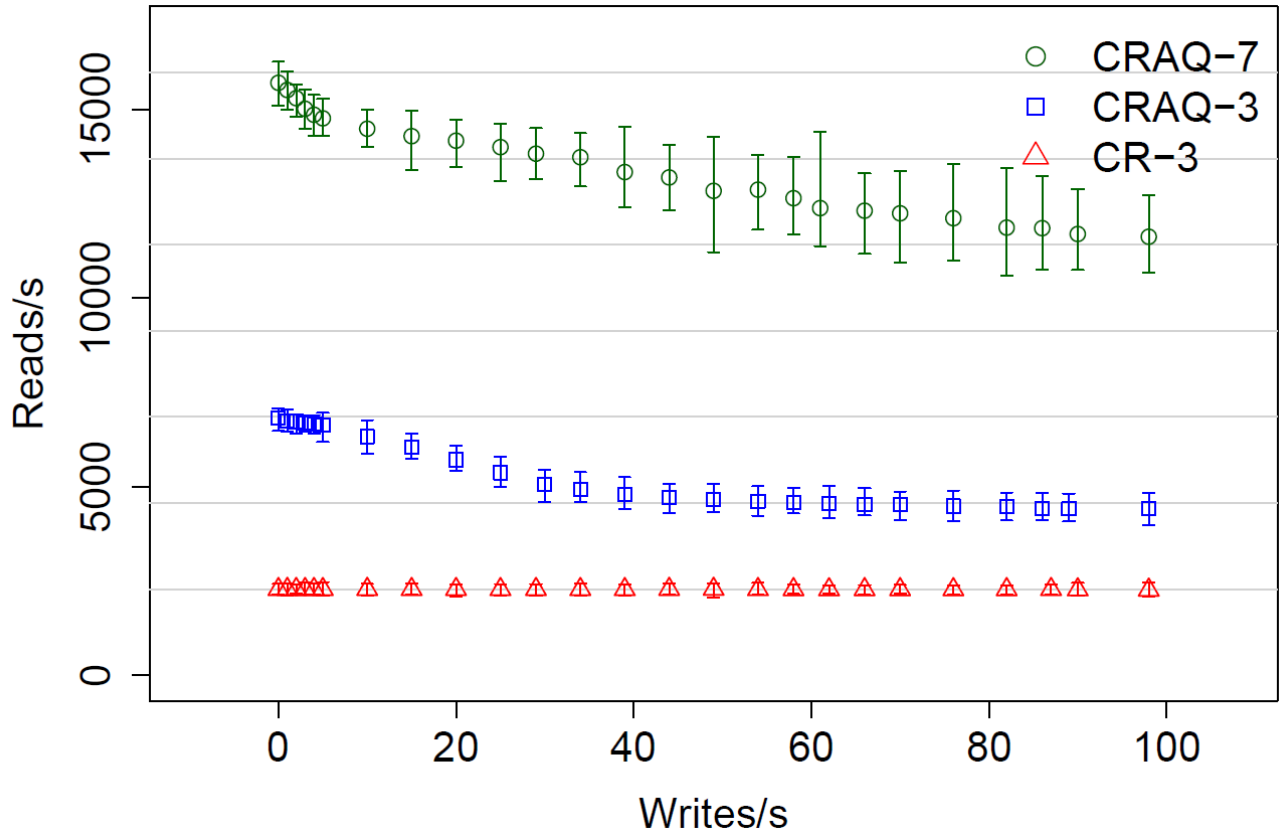
CRAQ: Optimizations

- **Multicast** writes and acknowledgements
  - *Head* multicasts new values to the entire chain, and propagates only *metadata* (smaller)
  - *Tail* multicasts acknowledgements to the entire chain
- Cross-datacenter **chain placement**
  - Manually assign order of datacenters to minimize write latency

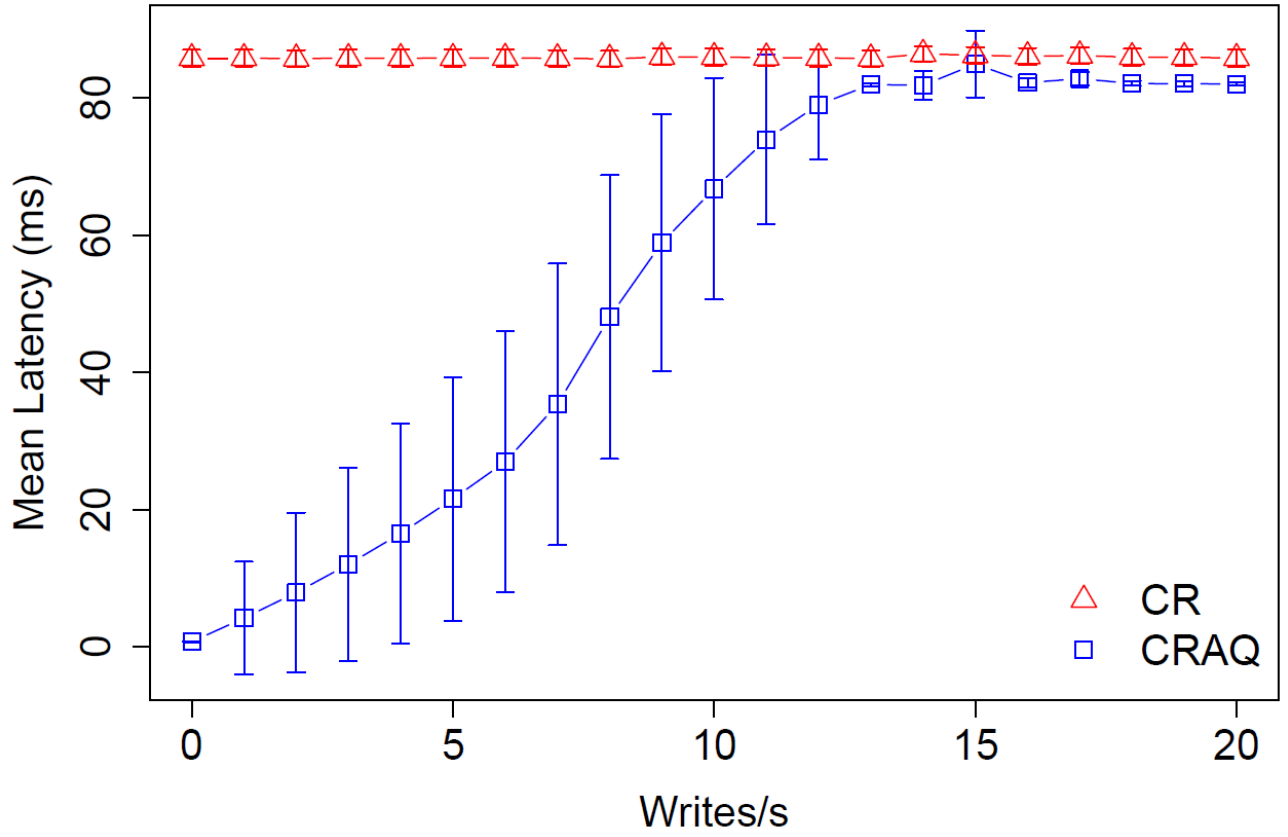
CRAQ: Membership Changes

- Uses Zookeeper to maintain node list membership
- When a node **fails** (times out), add a new node to the chain
  - Predecessor and successor send *full object state* (all versions) to the new node
  - To maintain consistency, new node *rejects reads* until it agrees with successor

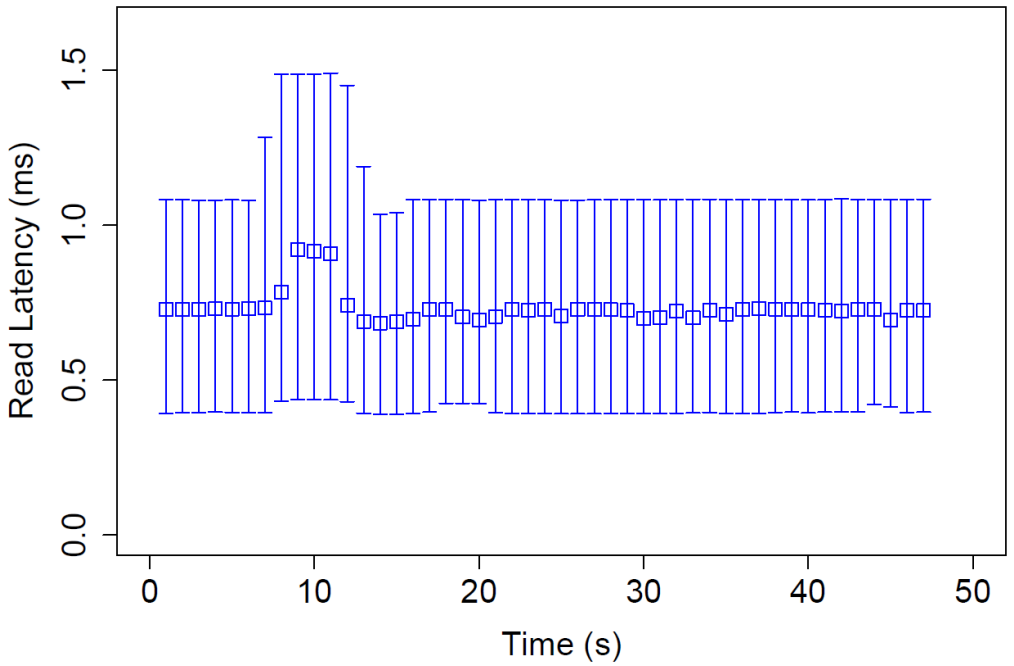
CRAQ: Read Throughput vs. Writes

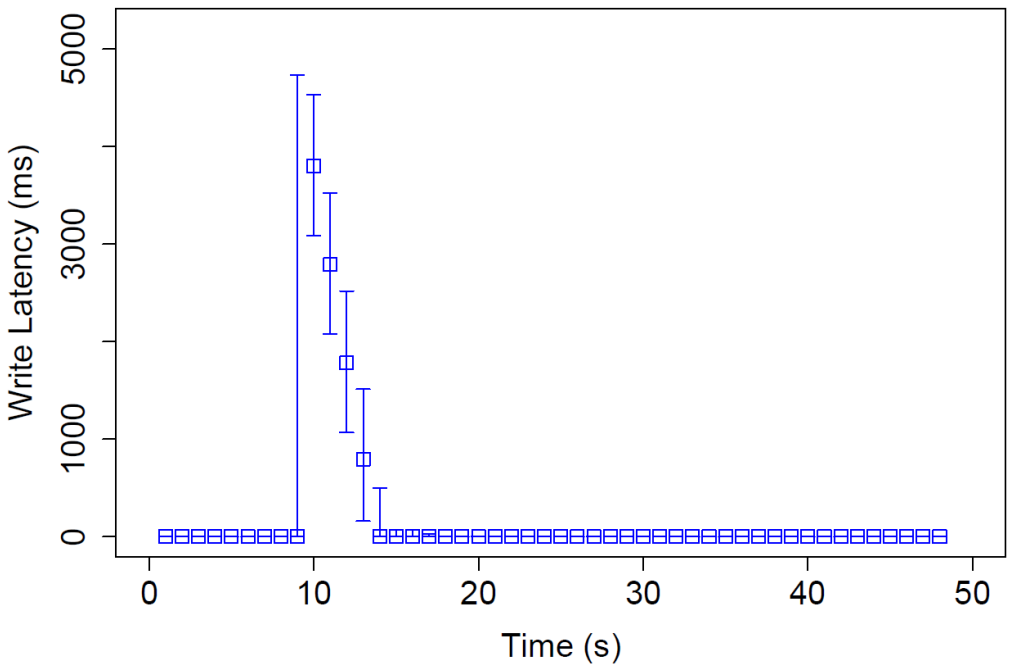


CRAQ: Geo-Replicated Read Latency vs. Writes



CRAQ: Latency During Failures





CRAQ: Evaluation

- Greatly improves *read* performance
  - In **read-mostly** workloads (throughput scales linearly with chain size)
  - In **write-heavy** workloads (version queries to the tail are lighter-weight than full reads)
- Limitations:
  - Write latency **increases linearly** with chain size
  - No writes can commit when **partitioned** or with any **failures** (before recovery)
  - Strong consistency in ordering operations **per object**, but not for different objects