CGAR: Strong Consistency without Synchronous Replication

Seo Jin Park

Advised by: John Ousterhout

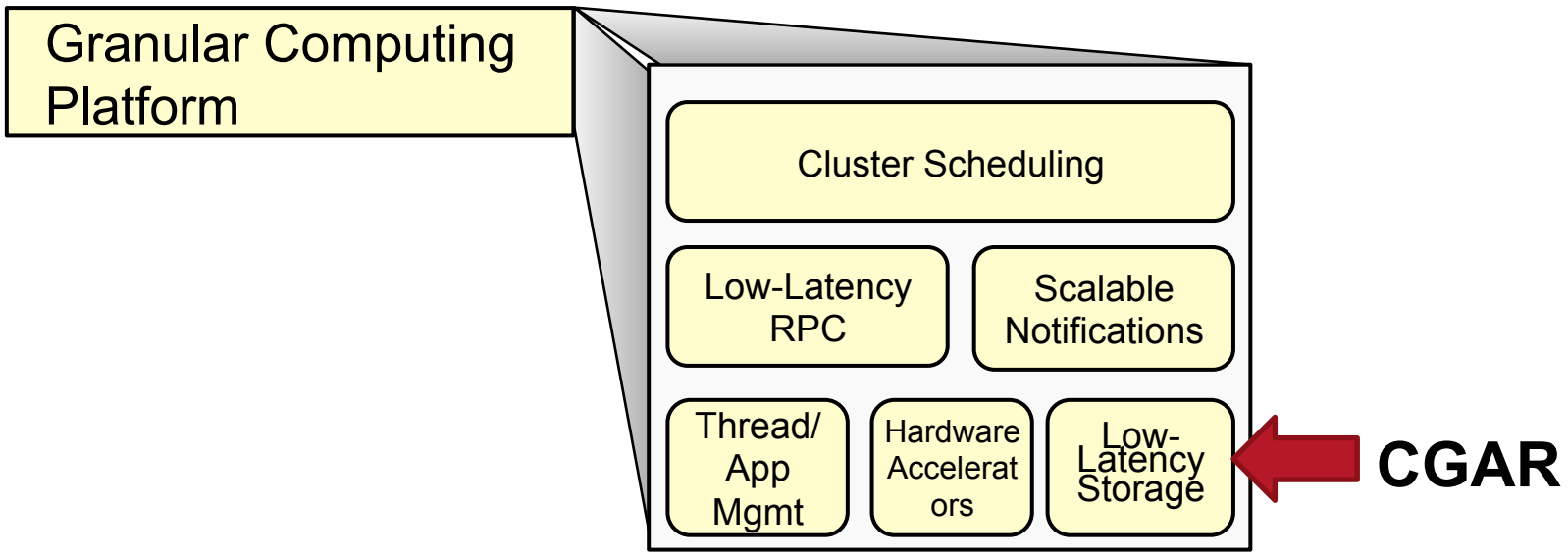
PLATFORMLAB

Stanford University

Overview

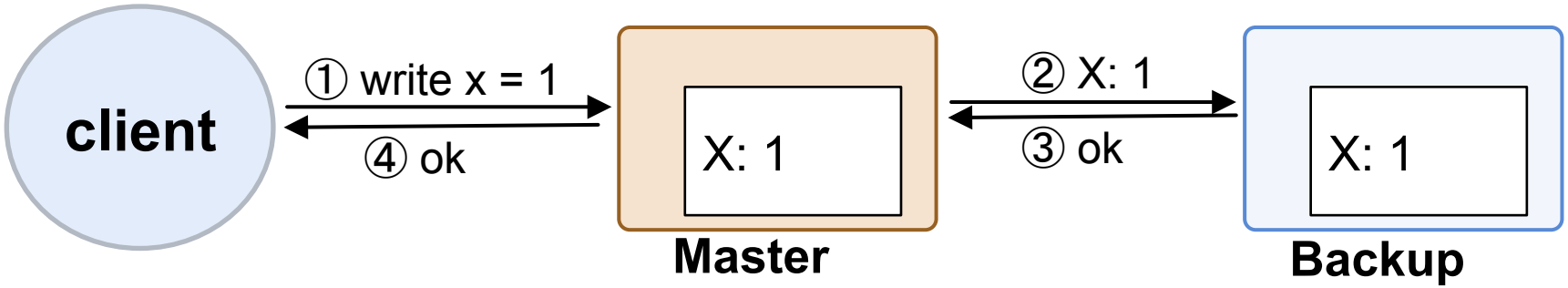
- **Improved update performance of storage systems with master-back replication**
  - **Fast:** updates complete before replication to backups
  - **Safe:** save RPC requests and retry if master crashes
- Two variants:
  - **CGAR-C**: save RPC requests in **client** library
  - **CGAR-W**: save RPC requests in a different server (**Witness**)
- Performance Result
  - **RAMCloud**: **0.5x** latency, **4x** throughput
  - **Redis**: strongly consistent (cost: 12% latency ↑)

CGAR's Role in Platform Lab



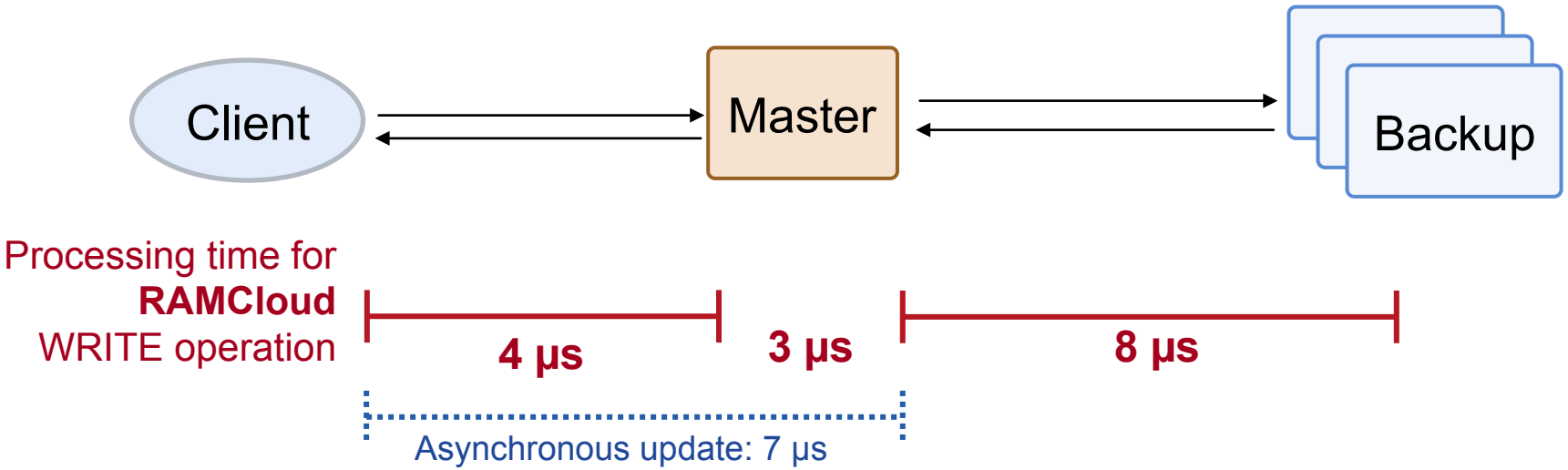
Consistency in Master-Backup

- **Master-backup replication:** client send updates to a master and master replicate state to backups.
- **Consistency after crash**
  - Responses for **update** operations must wait for **backup replications** (*synchronous replication*)
  - Must not reveal **non-replicated** value



Waiting for Replication is Not Cheap

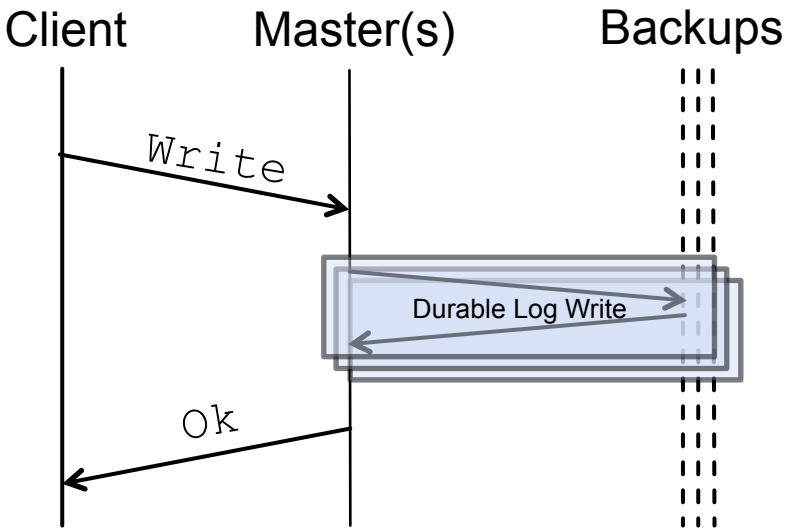
- **Synchronous replication increases latency of updates**
- **Alternative: asynchronous replication**
  - Non-replicated data can be lost
  - Sacrifice consistency if master crashes
  - Enables batched replication (more efficient)



Consistency over Performance: RAMCloud

RAMCloud uses synchronous replication

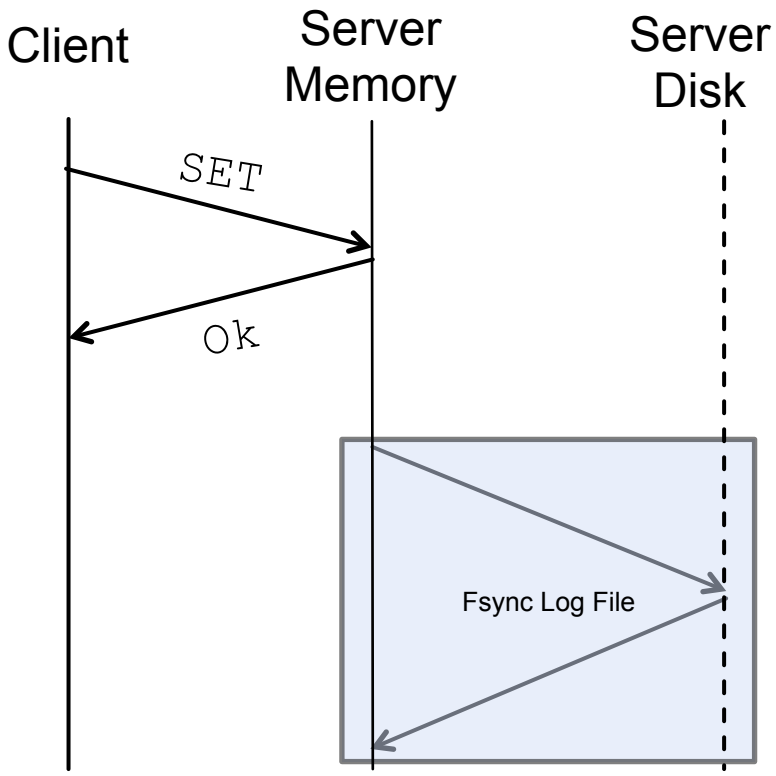
- Consistent even after crash
- Write: 14.3 µs vs. Read: 5 µs
- Focused on minimizing latency while consistent
- **Polling wait for replication**
  ➔ Write throughput is only 18% of read throughput



Performance over Consistency: Redis

Redis uses asynchronous replication

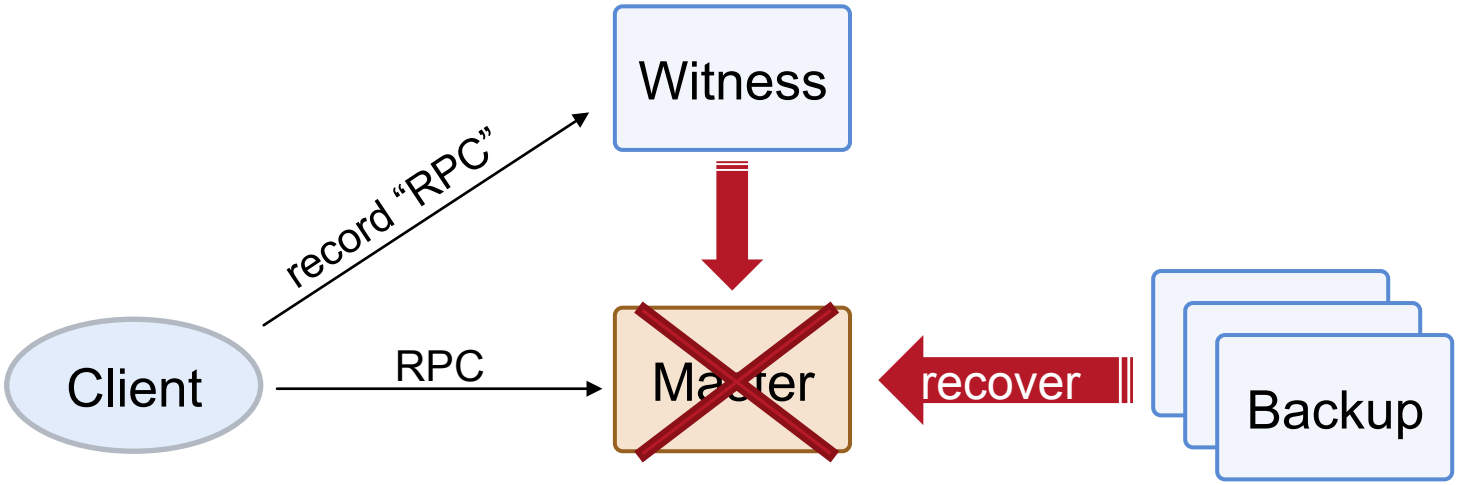
- Backup to a file in disk
- Default: fsync every second
  - Lose data if a master crashes
- Option for strong consistency: fsync-always
  - On SSDs, 1~2 ms delay
  - Without fsync, SET takes 25 μs.



Can we have both consistency and performance?

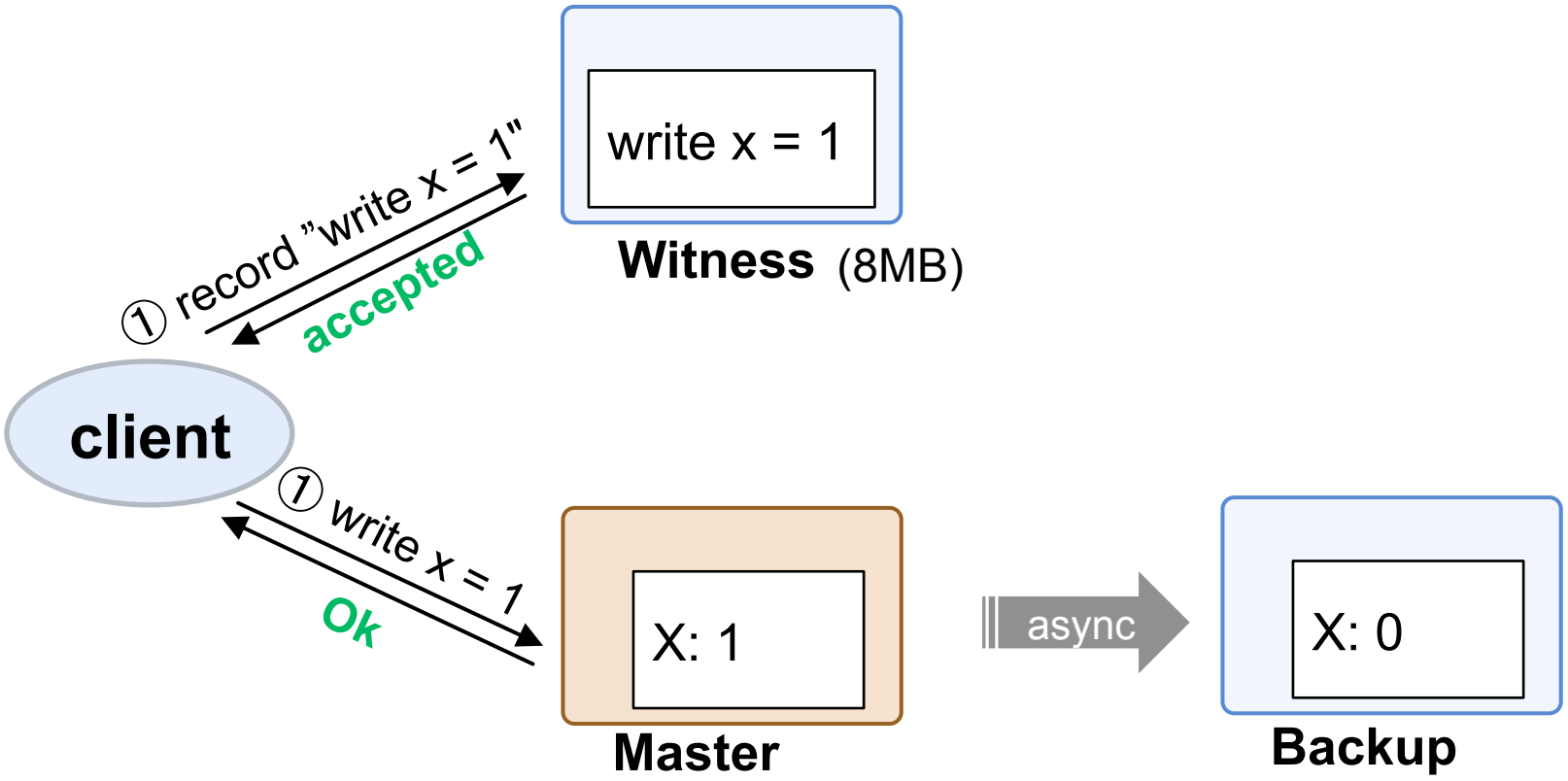
Consistency Guaranteed Asynchronous Replication

- **Asynchronous Replication → performance**
- **For consistency**
  - Save RPC requests in 3rd-party server (**Witness**)
  - Replay RPCs in Witness if master crashes



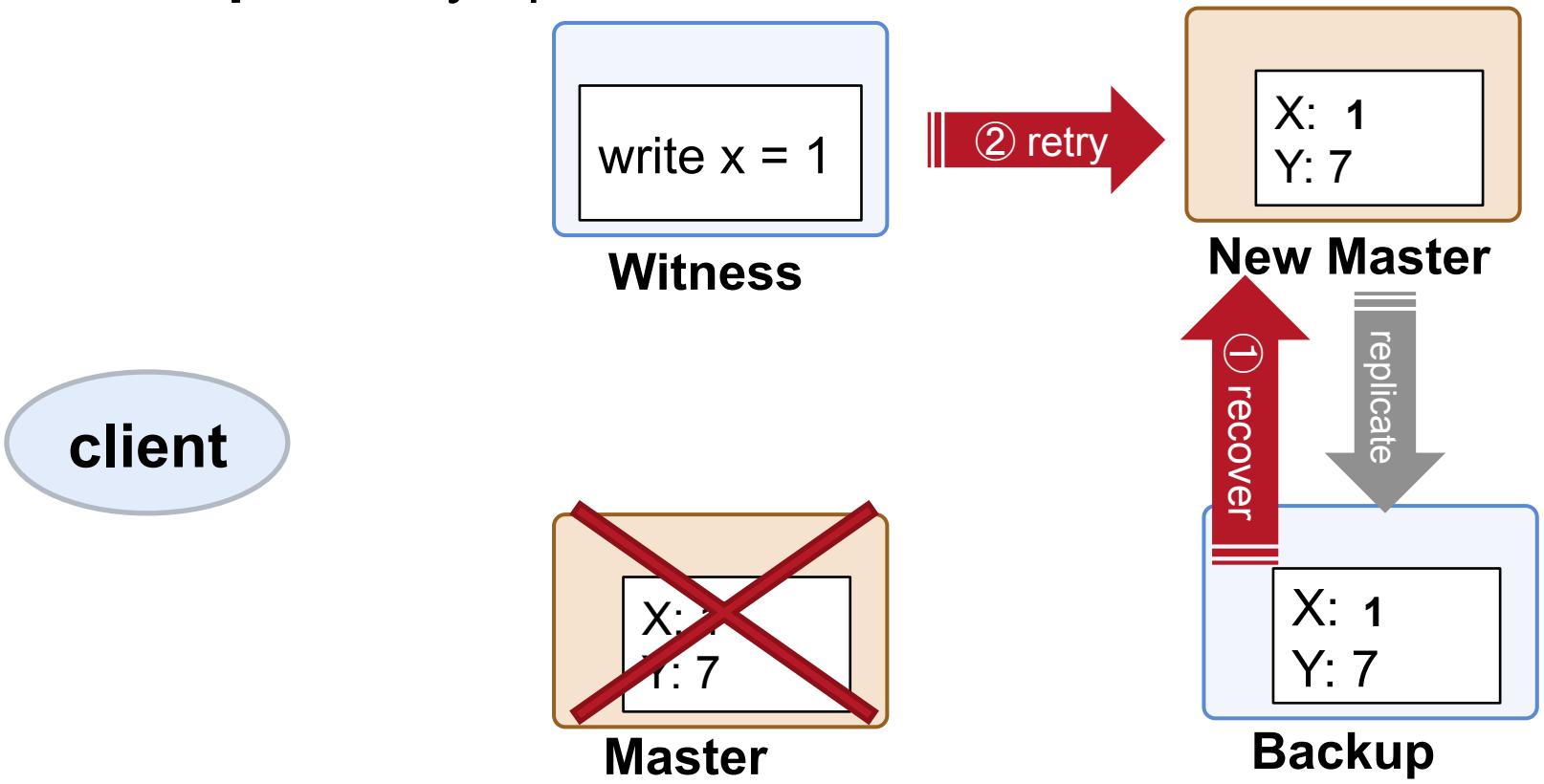
Witness Record Operation

- Client multicasts RPC request to master and witness
- Witness vouches the RPC will be retried if master crash



Recovery Steps of CGAR-W

- **Step 1:** recover from backups
- **Step 2:** retry update RPCs in witness

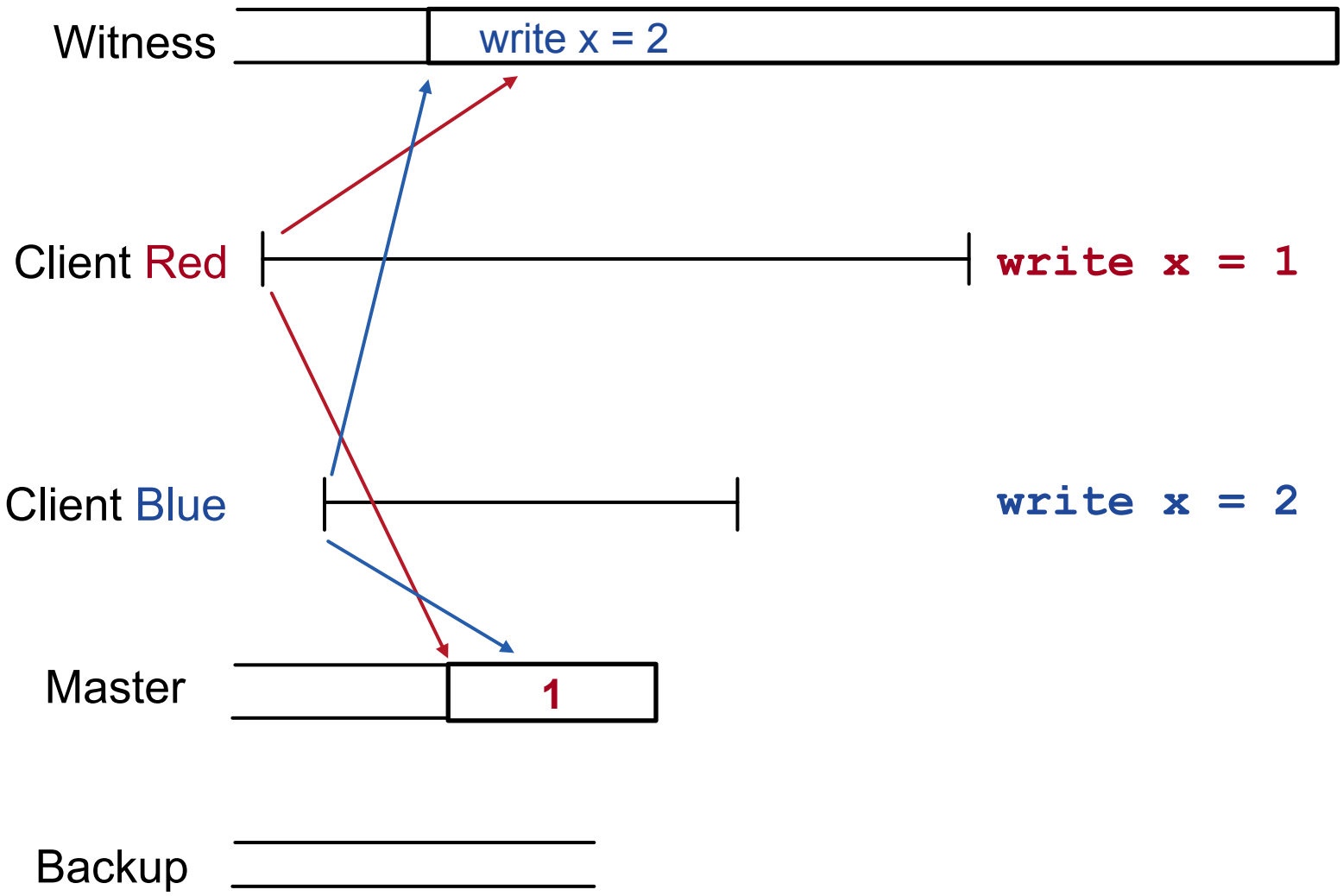


Challenges in Using Witness for Recovery

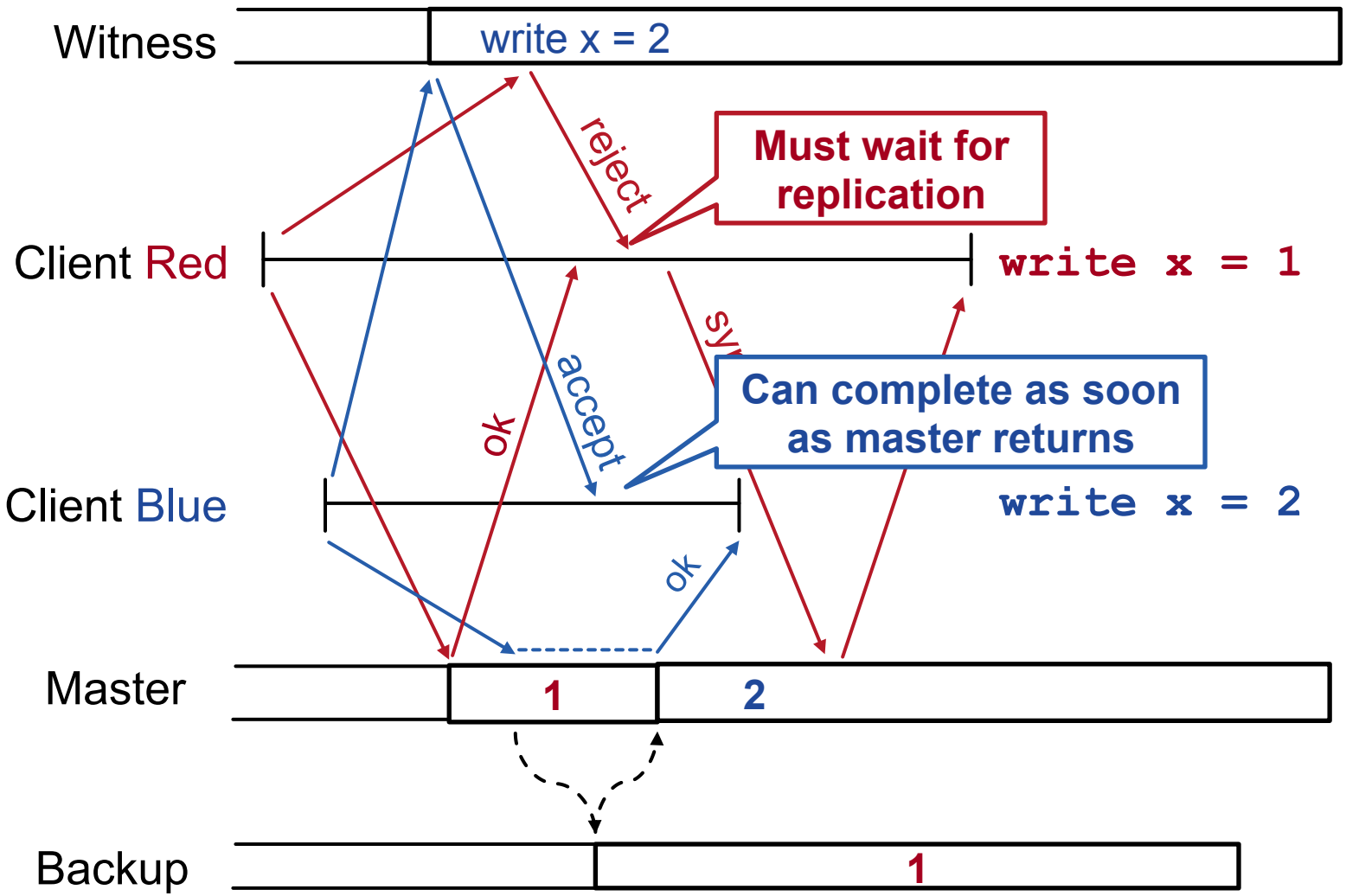
- **Witness may receive RPCs in a different order than master**
  - Solution: witness saves only 1 record per key
  - Concurrent operations on same key? Witness rejects all but first

- **Retry may re-execute an RPC**
  - Solution: use **RIFL** to ignore already completed RPC.
- **Update may depend on unreplicated value in master**
  - Master cannot assume witness saved the RPC request
  - Solution: delay update if current value is not yet replicated

Example: RPCs in a Different Order

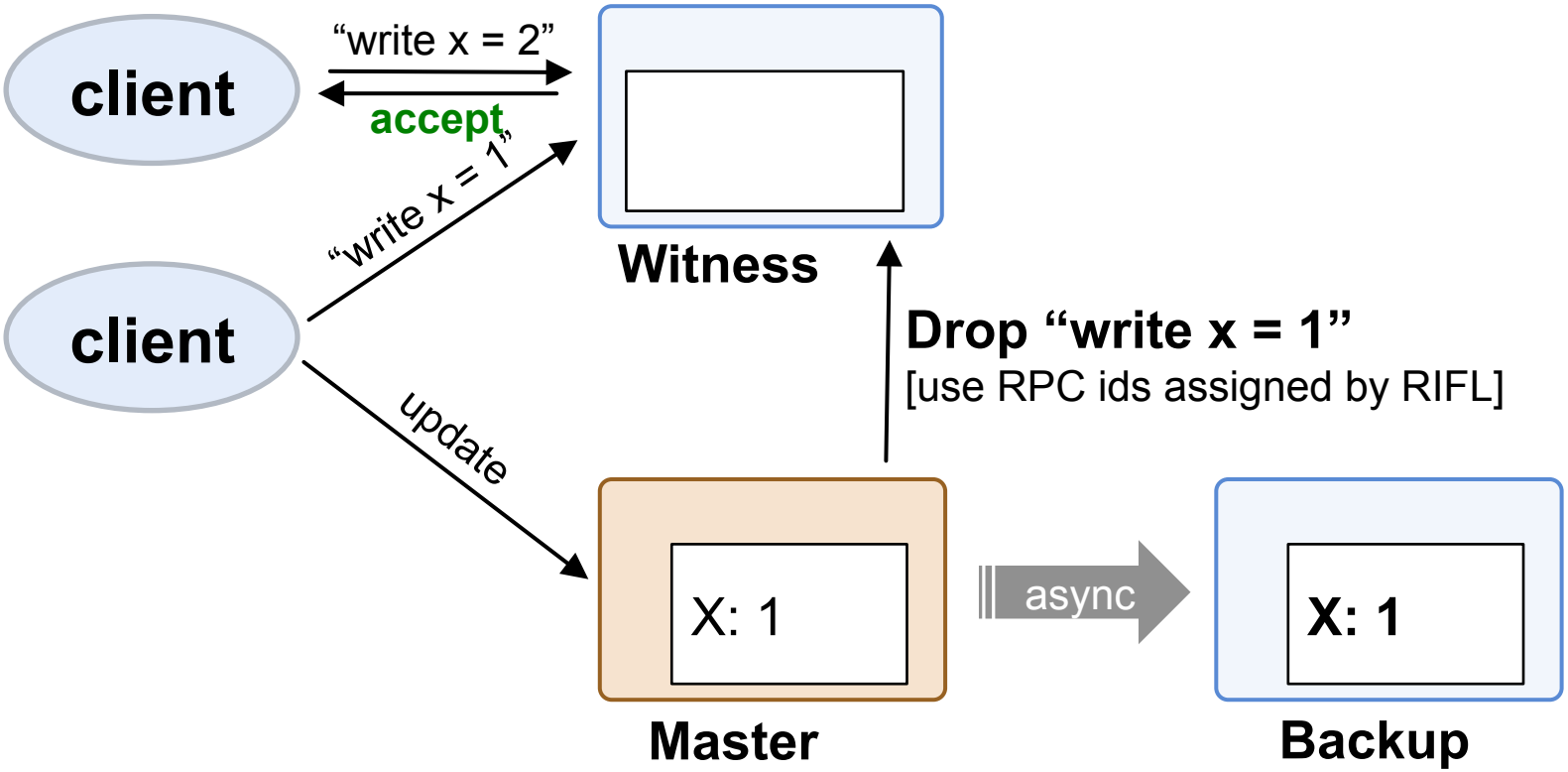


Example: RPCs in a Different Order



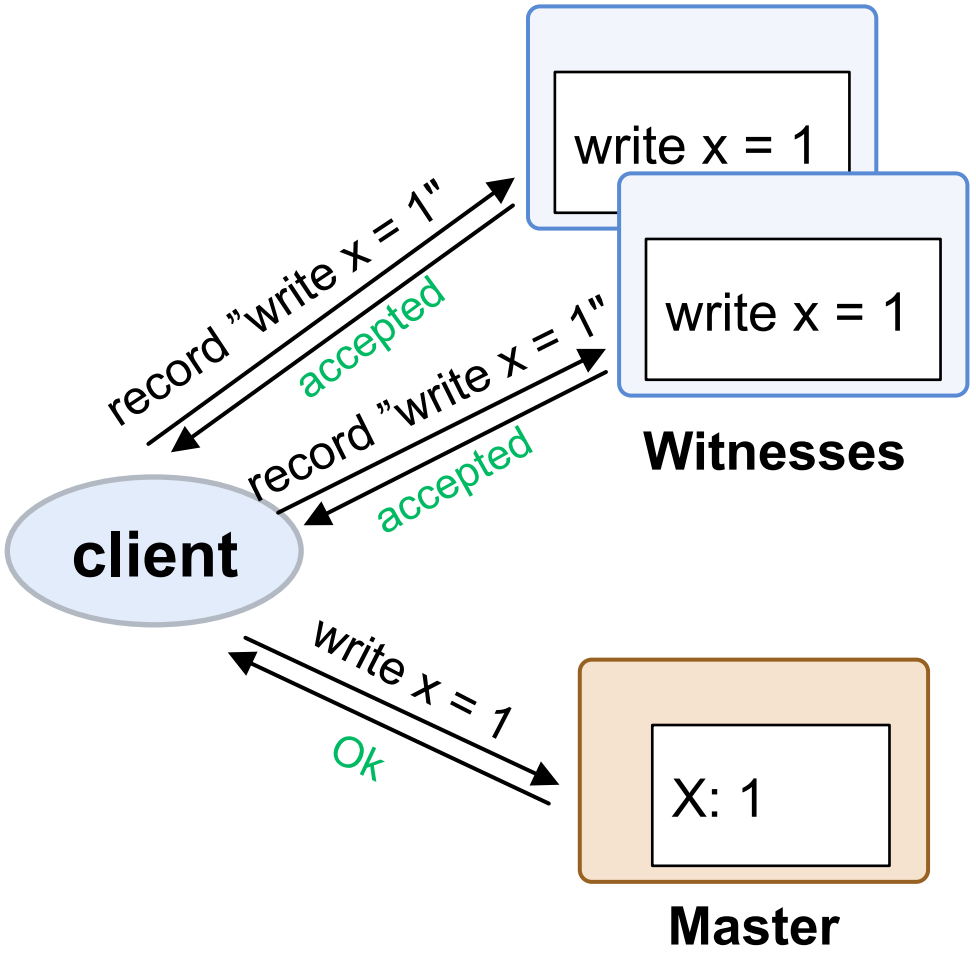
Garbage Collection

- **Witness must drop a record before accepting new one with same key**



Using Multiple Witnesses

- **A system can use multiple witnesses per each master**
  - Higher availability (recovery can use any witnesses)
  - To use async update, all witnesses must accept

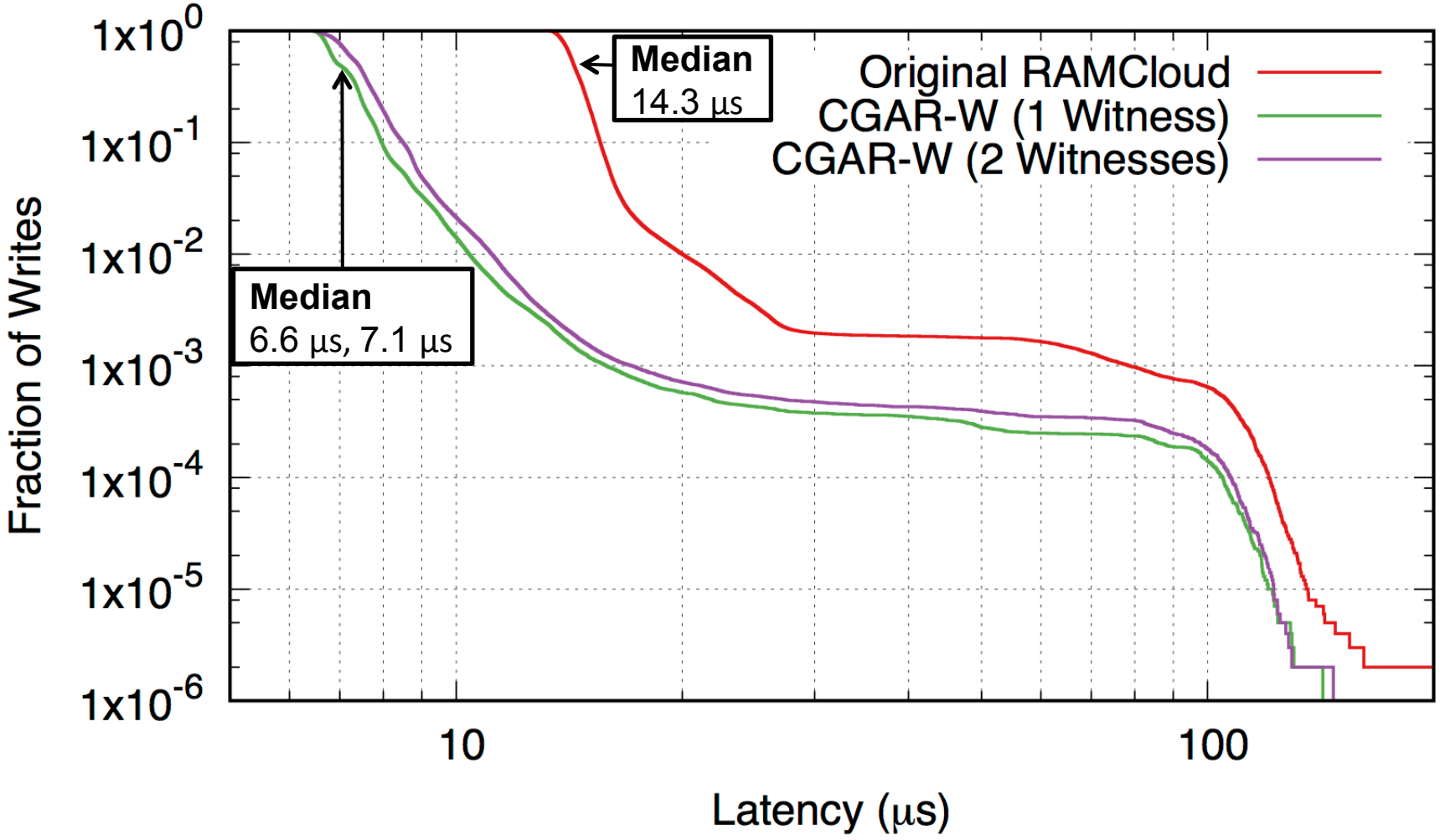


Evaluation of CGAR

- **RAMCloud implementation**
  - Performance improvement
  - Latency reduction
- **Redis implementation**
  - Supports wide range of operations

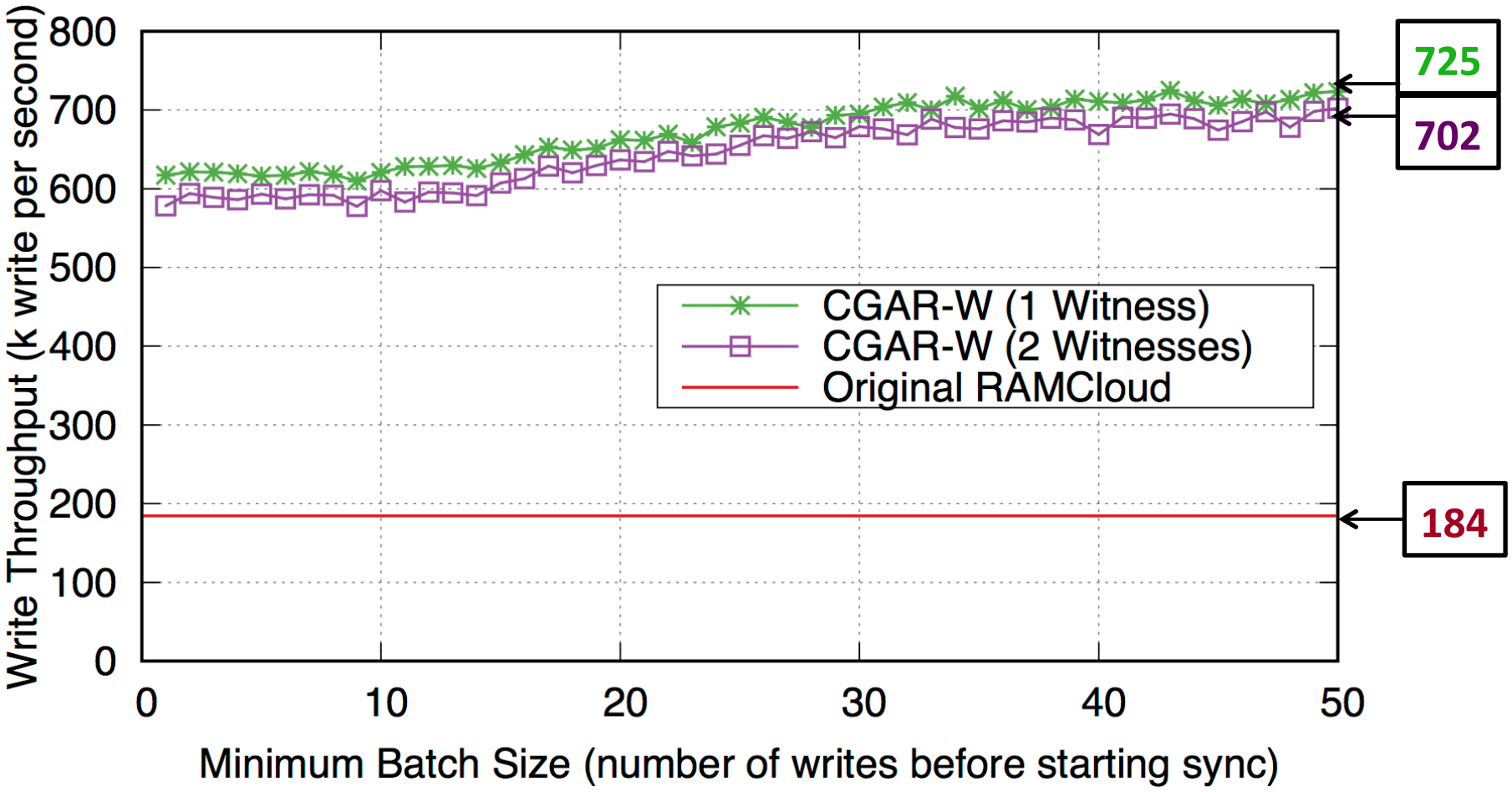
RAMCloud's Latency after CGAR

- **Writes are issued sequentially by a client to a master**



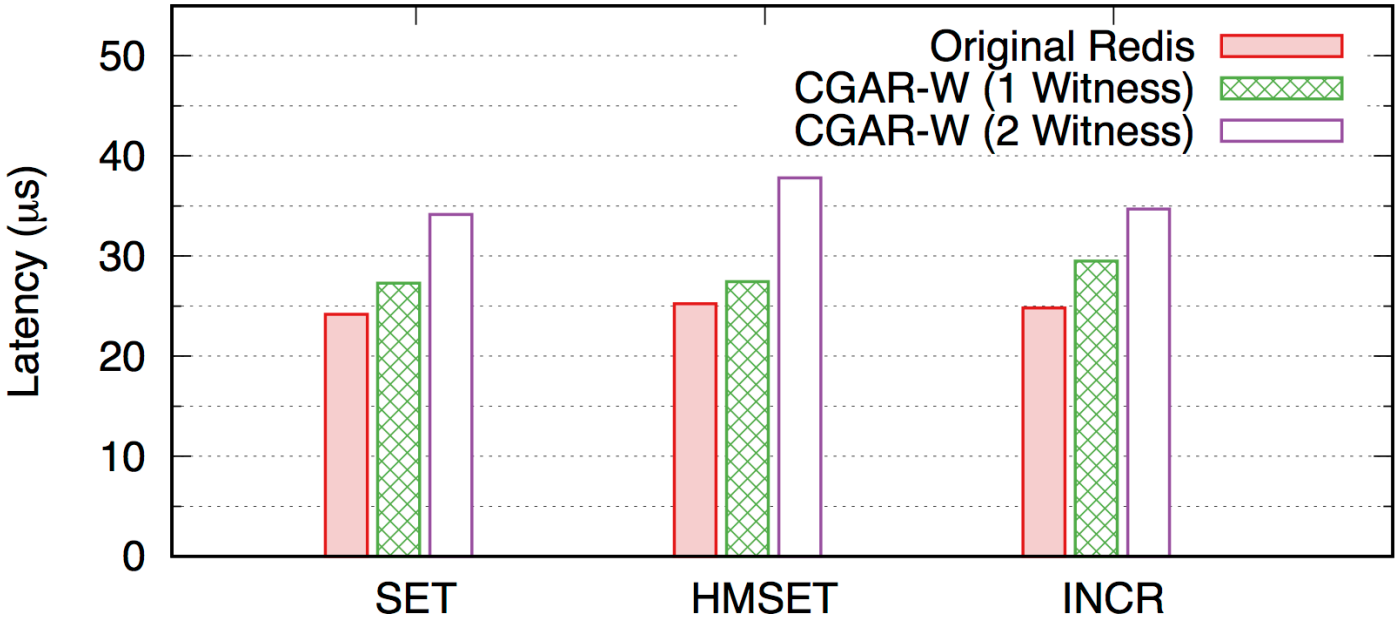
RAMCloud's Throughput after CGAR

- **Batching replication improved throughput**



Making Redis Consistent with Small Cost

- **SET:** write to **key-value** store
- **HMSET:** write to a member of **hashmap**
- **INCR:** increment an **integer counter**



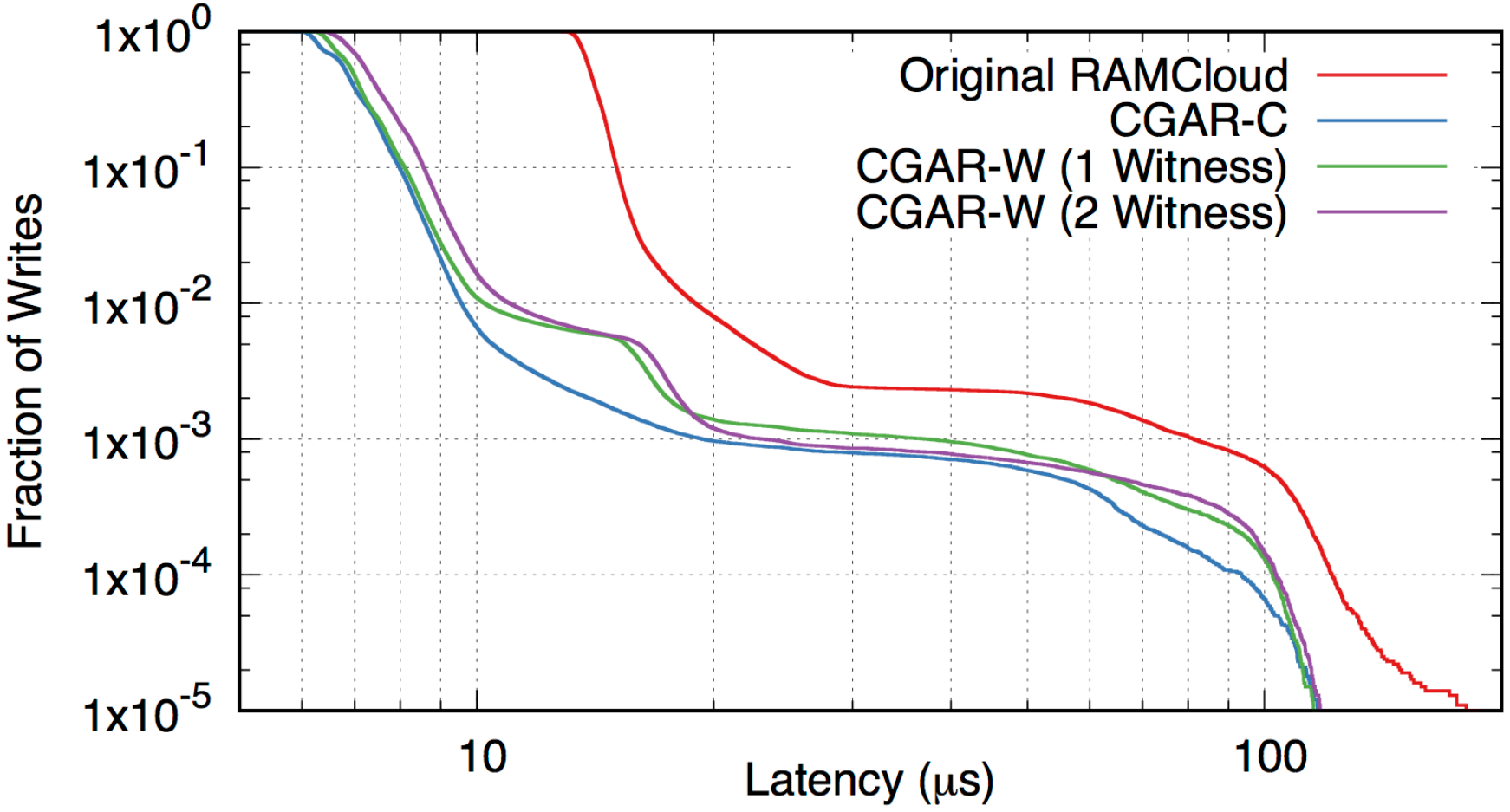
Conclusion

- **Fast:** updates don’t wait for replication
- **Consistent: CGAR** saves **RPC requests** in **witness**; If server crashes, **retry** the saved RPCs to recover
- **High throughput:** replication can be batched

Questions

Latency under Skewed Workloads

- **YCSB-A: Zipfian dist** (1M objects, p = 0.99)



CGAR Decoupled Replication from Update

- **Delay replication RPC's completion time**