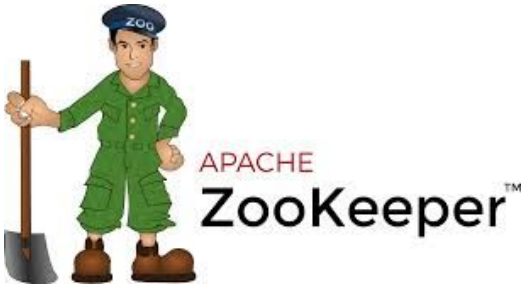
Performance and Reliability Isolation in ZooKeeper

Dax Chen, Yi-Shiun Chang
Chia-Wei Chen, Pei-Hsuan Wu

ZooKeeper: distributed coordination service

Common use cases:

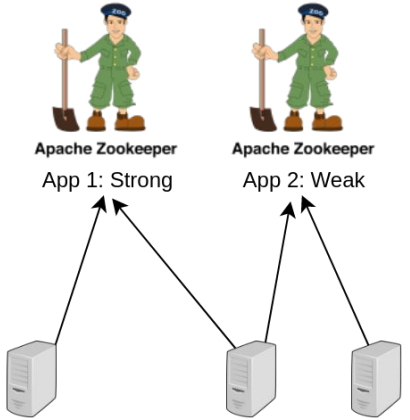
- distributed configuration service
- synchronization service (eg. distributed locks)



Currently, ZooKeeper can only be either **strong** or **weak**, but not both together

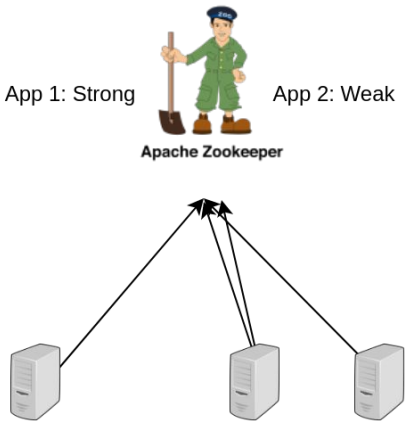
But what if clients need multiple different consistency?

-> Deploy **several** ZK services



Goal

- Provide multi-consistency on **single** ZooKeeper
- **Transparency** on client side
- **Isolate availability** of different consistency levels



Key idea: namespaces

Create namespaces for different consistency level

- Each namespace has its own isolated consistency
- Subtree under the namespace uses that level

-> With simplified prototype

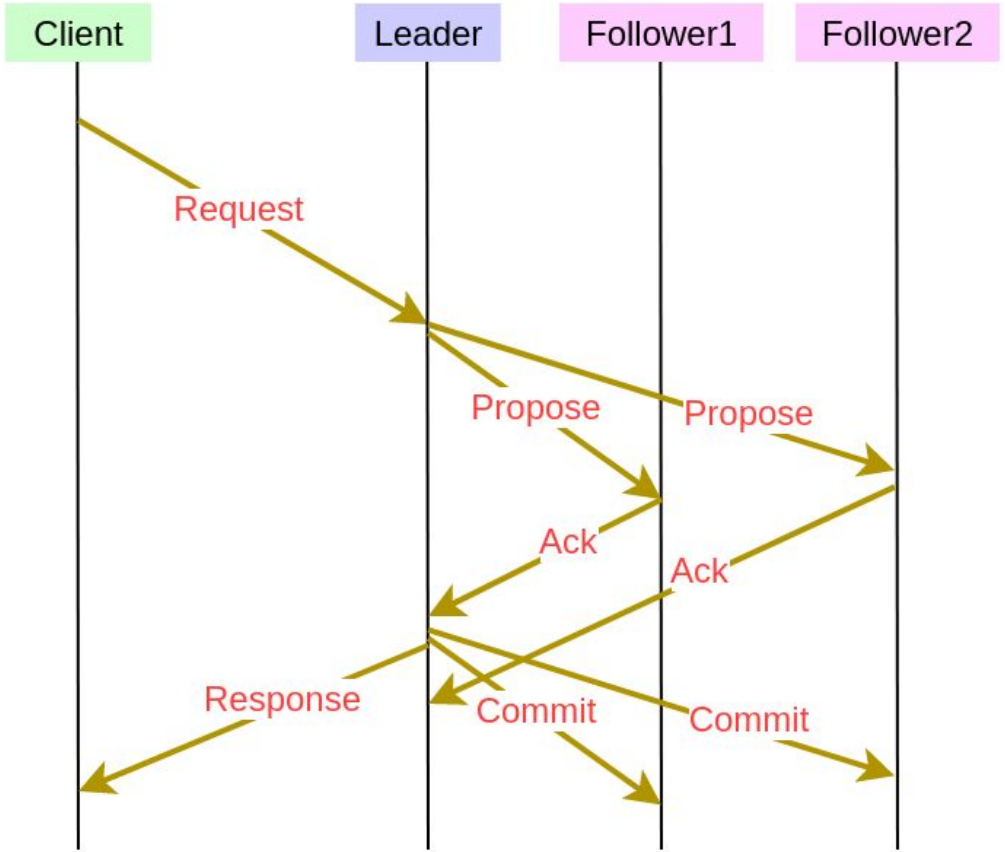
- **Strong: Under path /1**
- **Weak: Under path /2**

Result

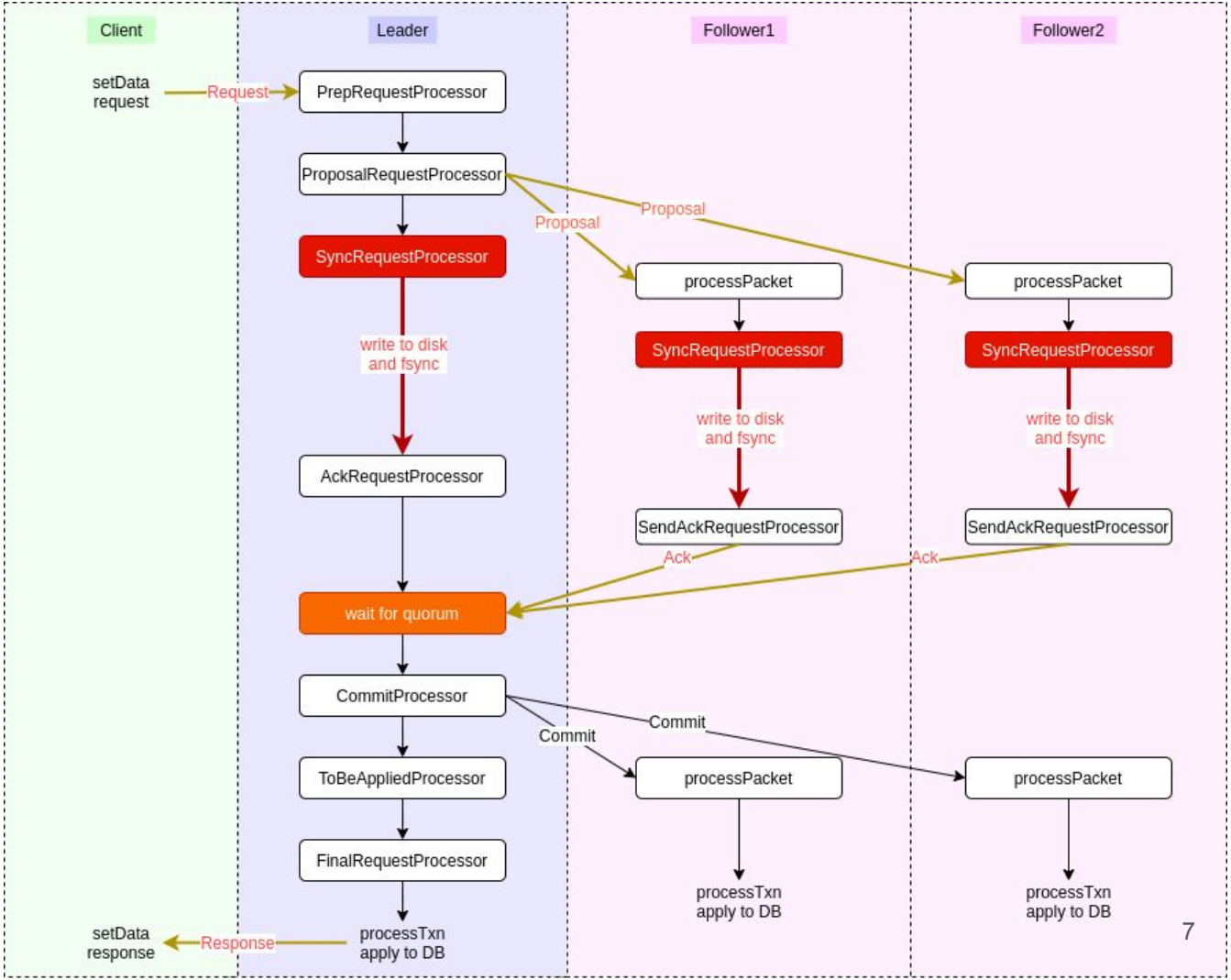
- Achieved **5.04x** in Weak compare to Strong request latency
- Successfully **separated latency entanglement** in different combination of Strong and Weak Clients
- Nearing **no effect** on Strong request **0.93x**

  and **1.98x** on Weak request latency comparing to 2 ZooKeeper
- Corrupted data isolation to **ensure availability** for **different consistency level**

Orignal StrongZK Request Chain



Orignal StrongZK Request Chain



Orignal StrongZK Request Chain

Strong latency (SSD): 800μs

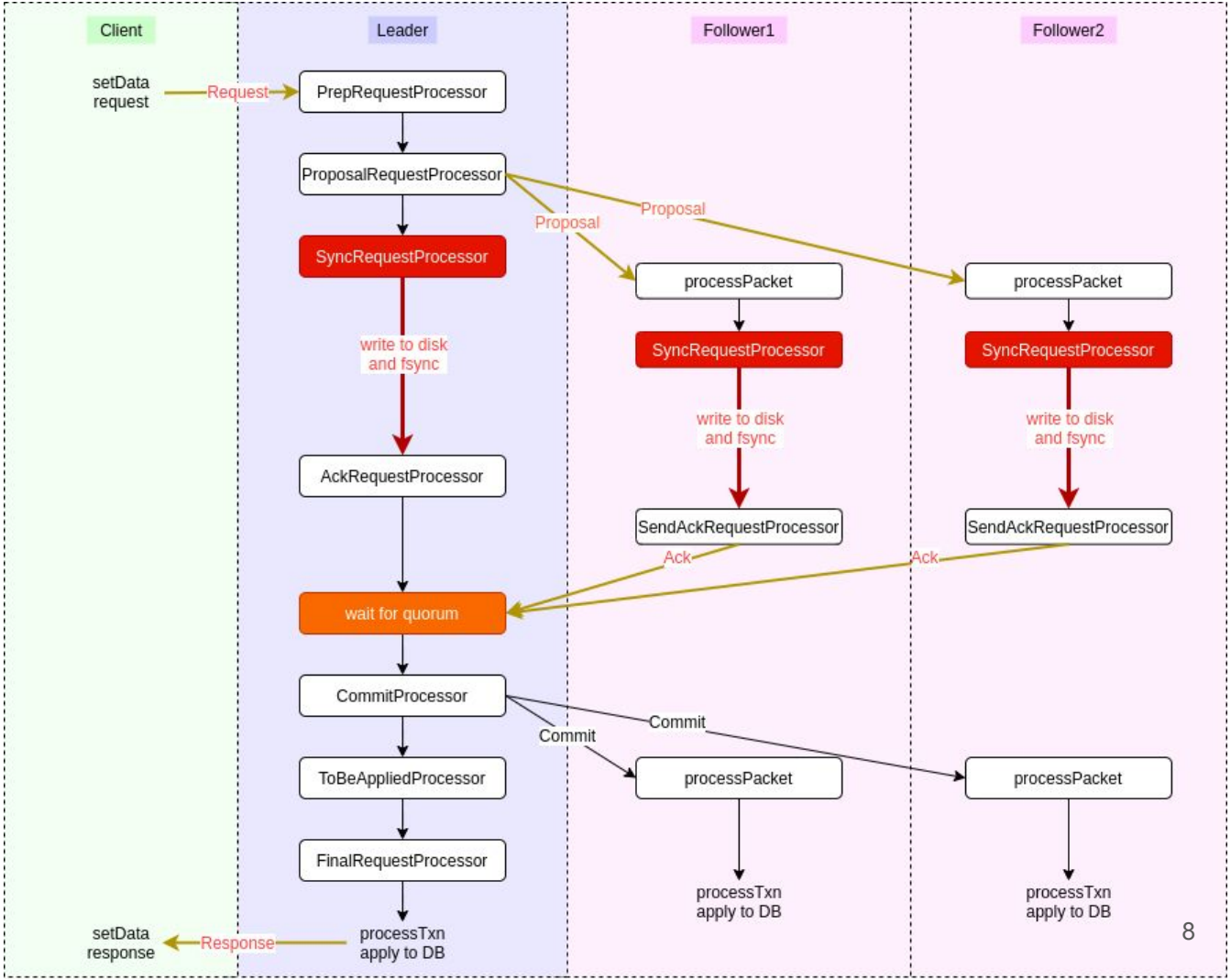
- 2 RTT
- 1 fsync

In our case:

- ping latency: ~120μs
- fsync latency: ~380μs
  (120 * 2 + 380 = 620μs)

Weak: 1RTT

- No fsync
- No wait for quorum



Design0 - Naïve approach

Recall goal

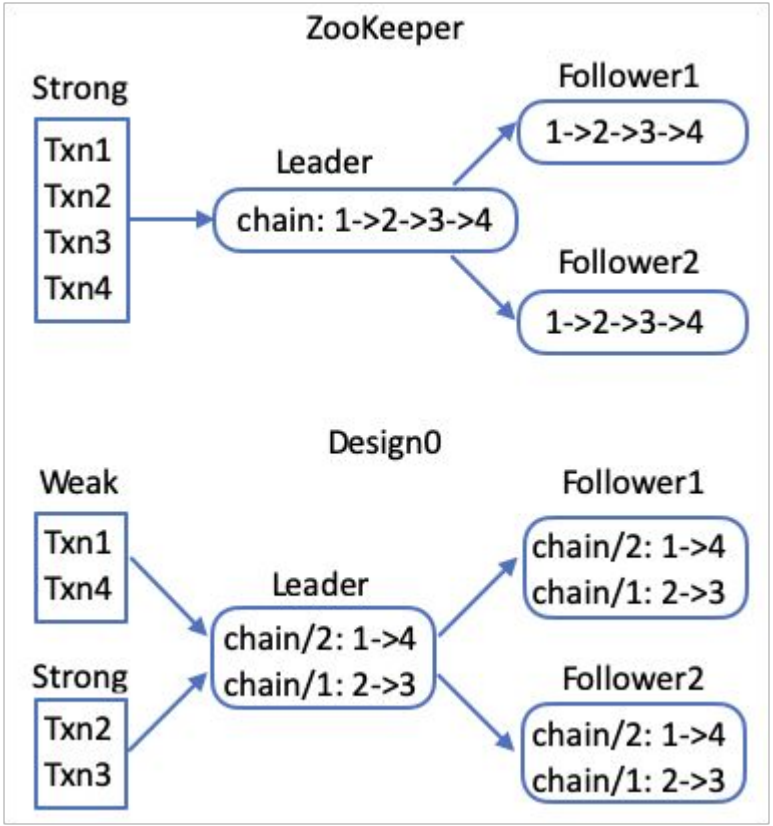
- Provide multi-consistency on single ZooKeeper

Raw idea

- Separate **Strong /1** and **weak /2** request chain
- Each consistency level has its pipeline in ZooKeeper

Problem

- Single pipeline
- **ZooKeeper does not allow out-of-order commits**



Design0 - Naïve approach

Recall goal

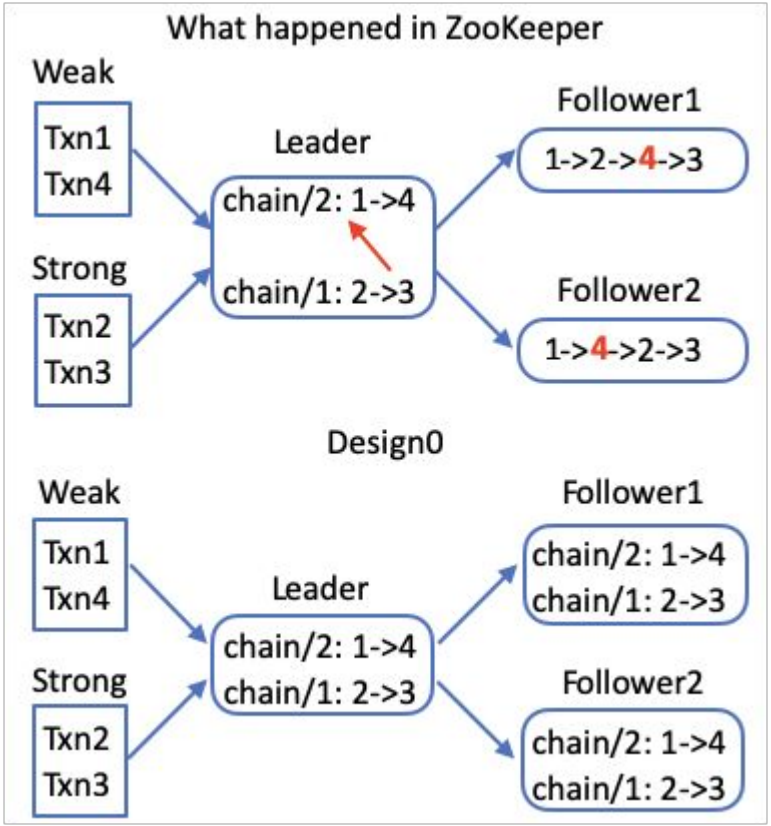
- Provide multi-consistency on single ZooKeeper

Raw idea

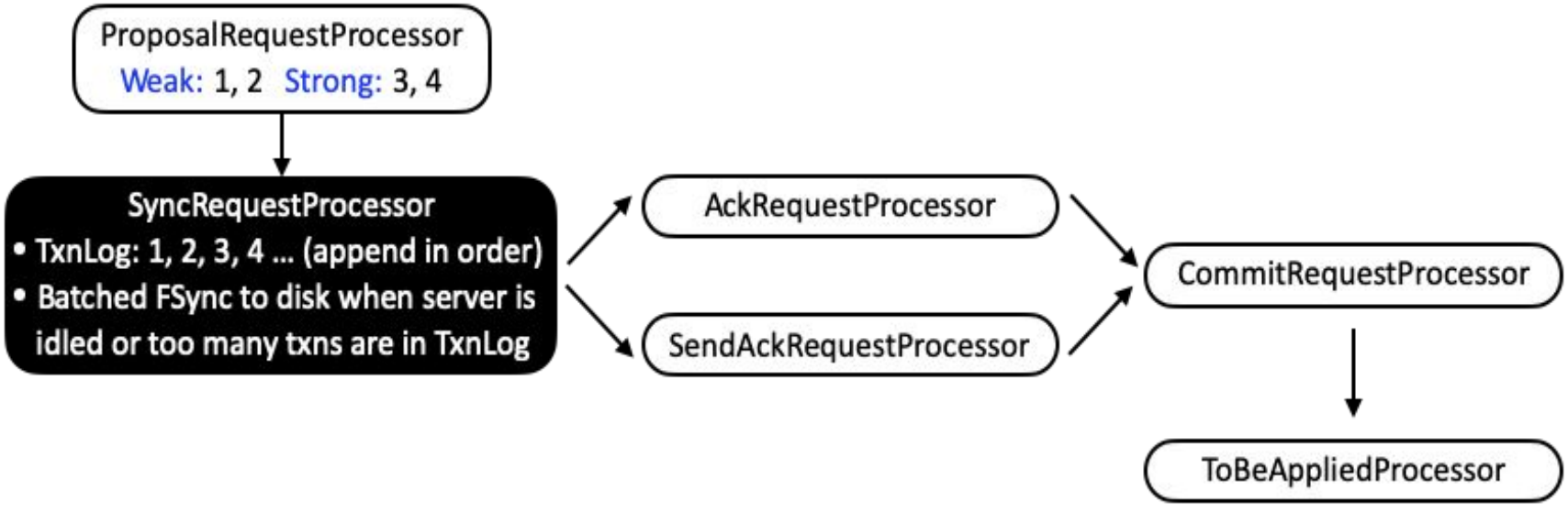
- Separate **Strong /1** and **weak /2** request chain
- Each consistency level has its pipeline in ZooKeeper

Problem

- Single pipeline
- **ZooKeeper does not allow out-of-order commits**



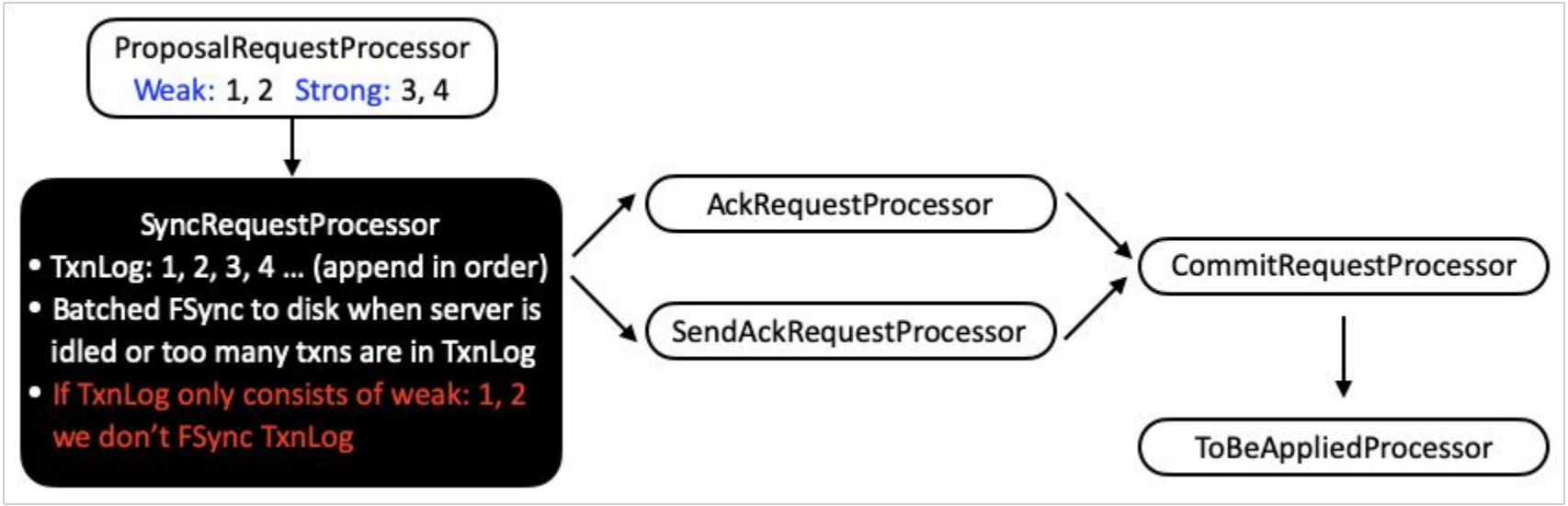
Design1 - Dynamic forceSync (1)



Opportunity in SyncRequestProcessor (respect the txn order)

- Strong consistency: forceSync txnLog into disk before servers ack back to leader
- Weak consistency: **don't forceSync** txnLog to disk, if txnLog doesn't have strong txns

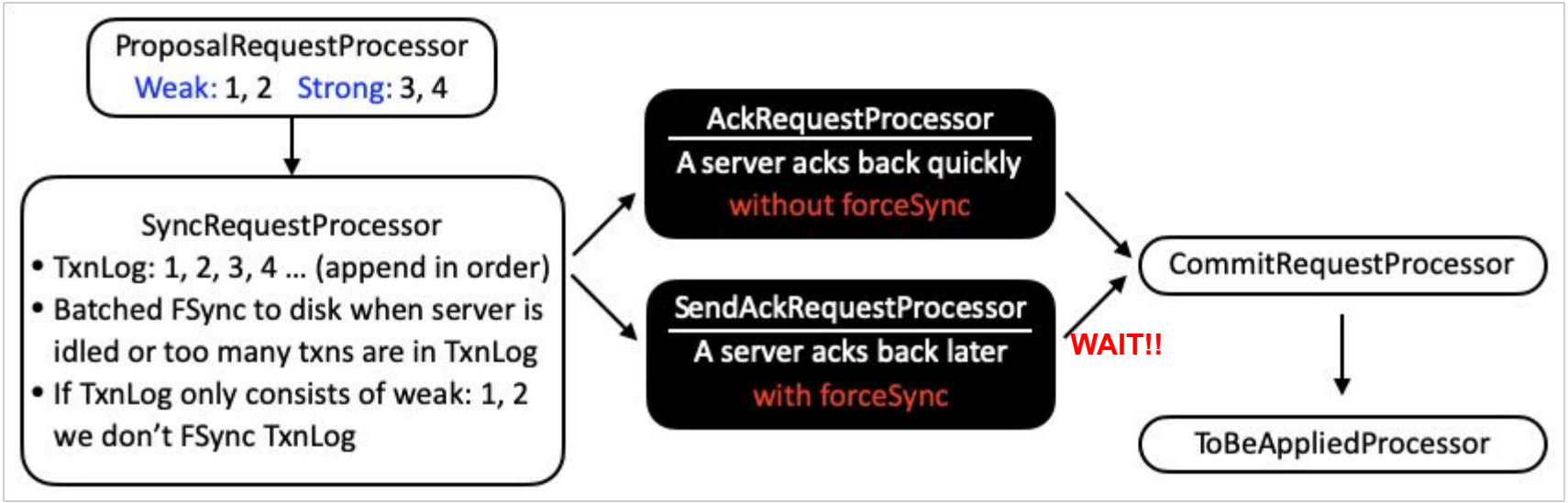
Design1 - Dynamic forceSync (1)



Opportunity in SyncRequestProcessor (respect the txn order)

- Strong consistency: forceSync txnLog into disk before servers ack back to leader
- Weak consistency: **don't forceSync** txnLog to disk, if txnLog doesn't have strong txns

Design1 - Dynamic forceSync (2)

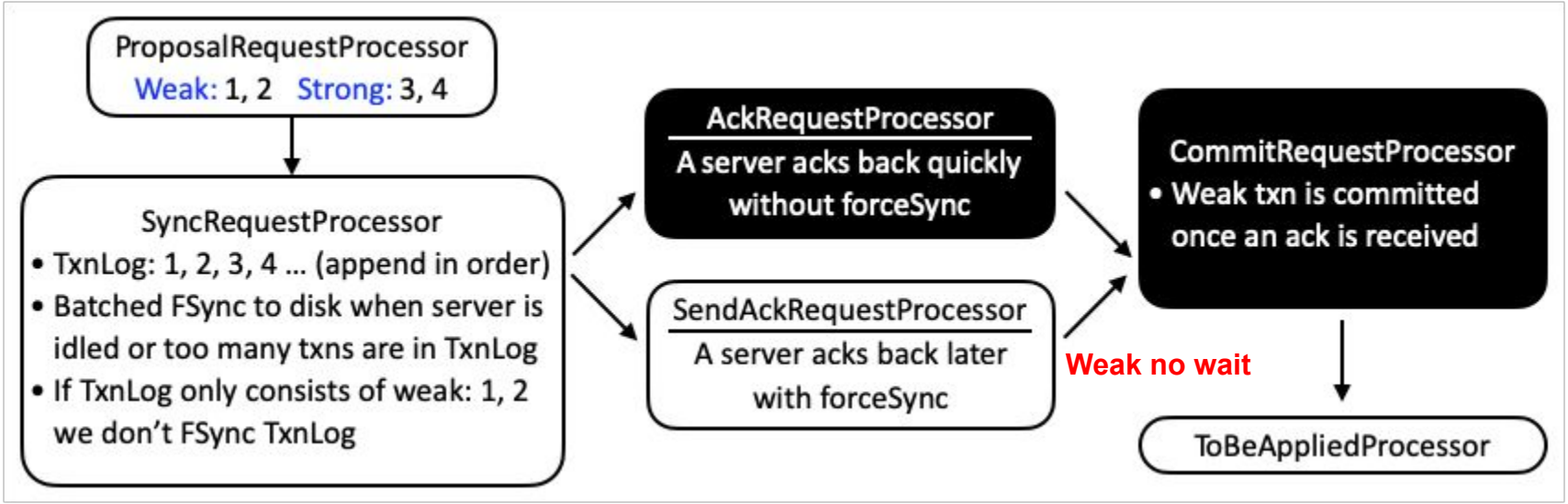


Opportunity in CommitProcessor

Strong consistency: leader broadcasts commit after receiving acks from majority

Weak consistency: leader broadcasts commit **without quorum** once it receives an ack

Design1 - Dynamic forceSync (2)

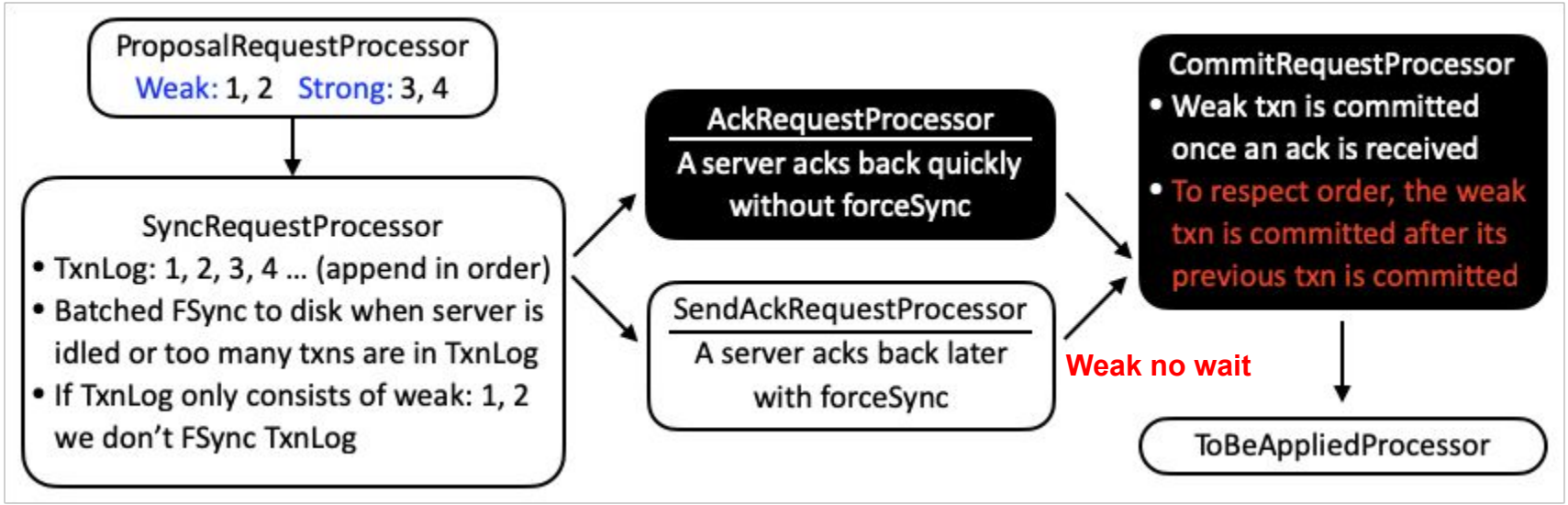


Opportunity in CommitProcessor

Strong consistency: leader broadcasts commit after receiving acks from majority

Weak consistency: leader broadcasts commit **without quorum** once it receives an ack

Design1 - Dynamic forceSync (2)

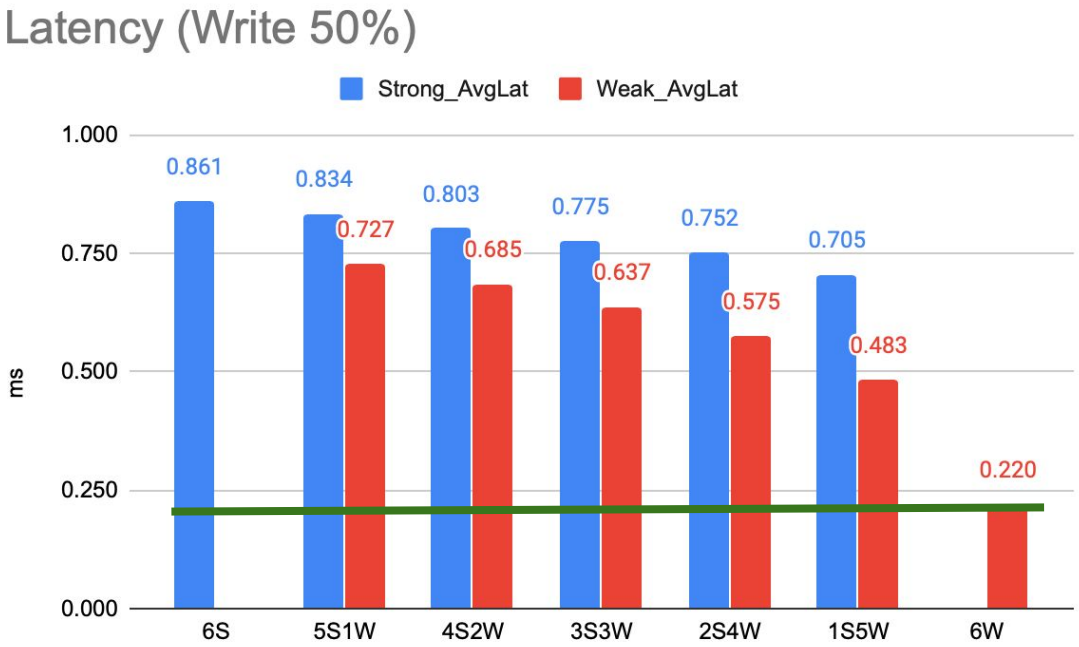


Opportunity in CommitProcessor

Strong consistency: leader broadcasts commit after receiving acks from majority

Weak consistency: leader broadcasts commit **without quorum** once it receives an ack

Design1 - Dynamic forceSync (3)



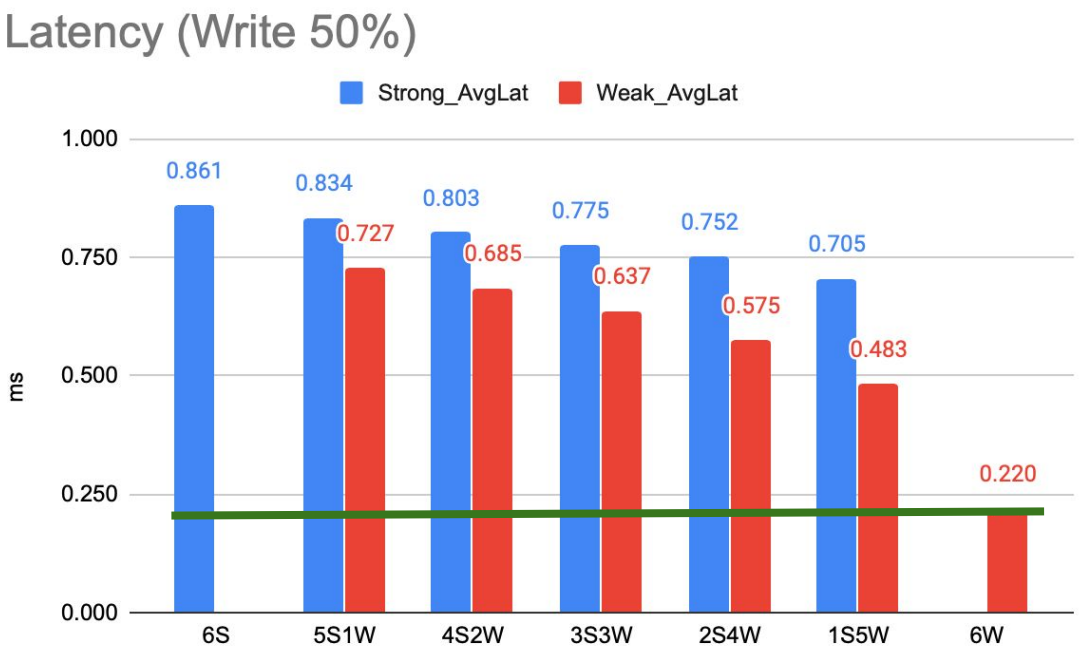
Weak consistency speed up:

- Servers ack **without FSync** to disk
- Leader commits **without waiting for majority**

Entanglement exists:

- **Batched Fsync**
- **Transaction order**

Design1 - Dynamic forceSync (3)



SyncRequestProcessor

- If TxnLog only consists of weak: 1, 2 we don't FSync TxnLog

CommitRequestProcessor

- To respect order, the weak txn is committed after its previous txn is committed

Entanglement exists:

- **Batched Fsync**
- **Transaction order**

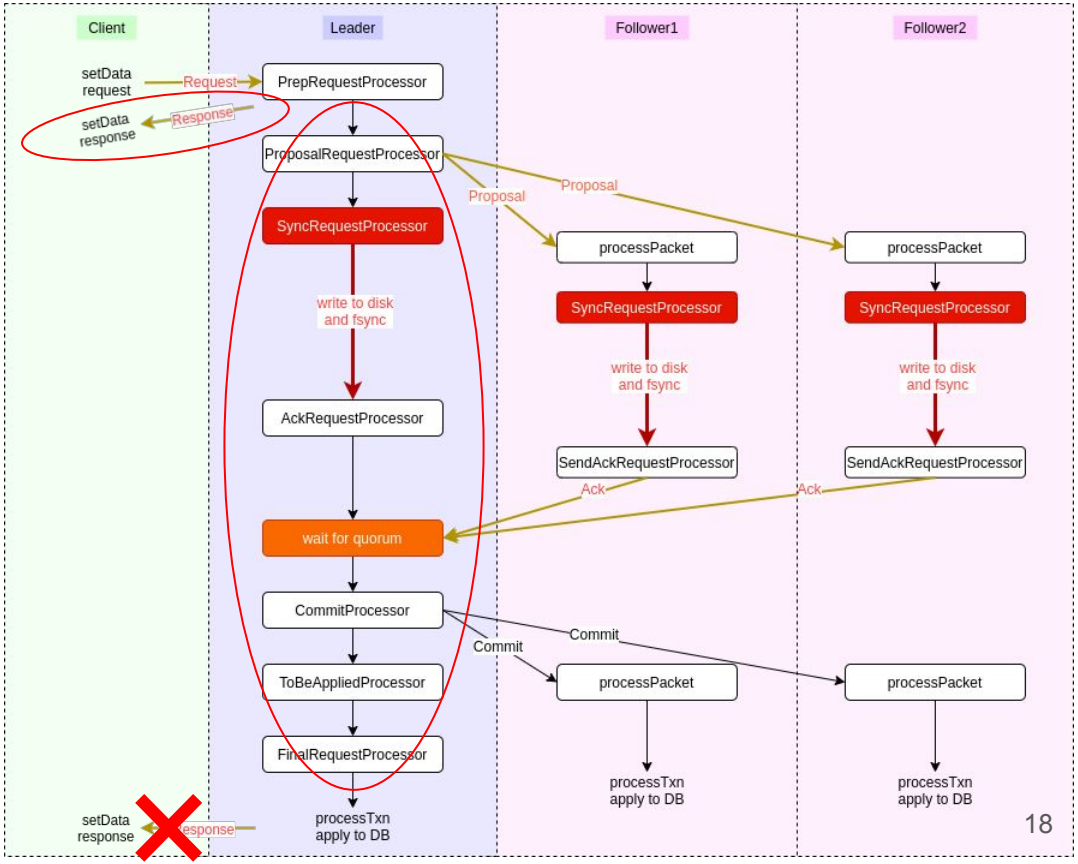
Design2 - Short-Circuit Response

Weak chain (latency 1 RTT)

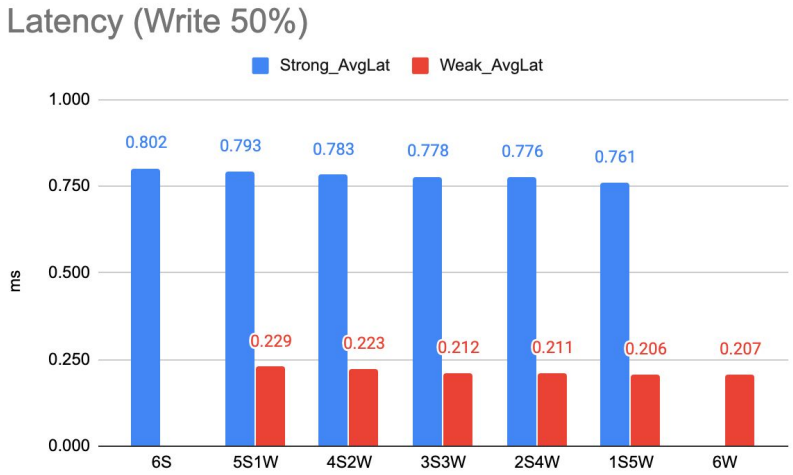
1. Respond early
2. **mark** request as responded
3. Request flows normally, commits **in-order**
4. At the end, don't respond

Guarantees Eventual Consistency

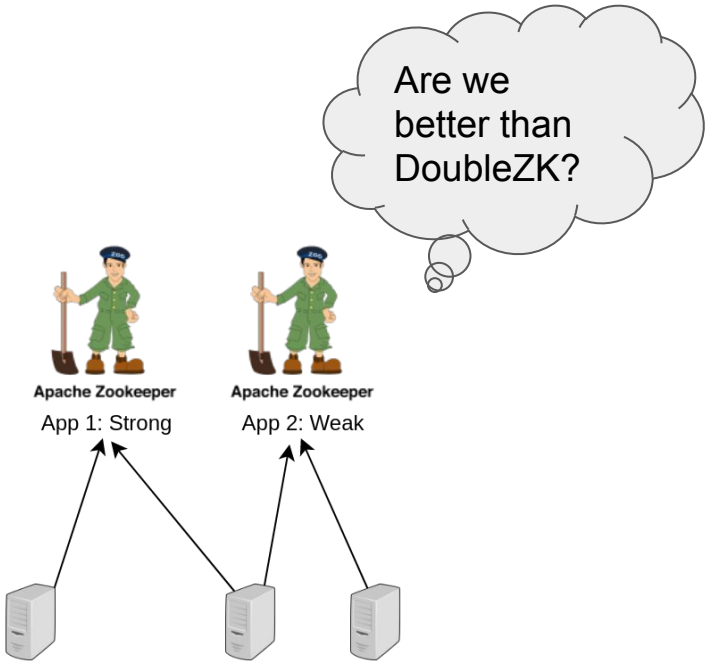
Weak not blocked by Strong



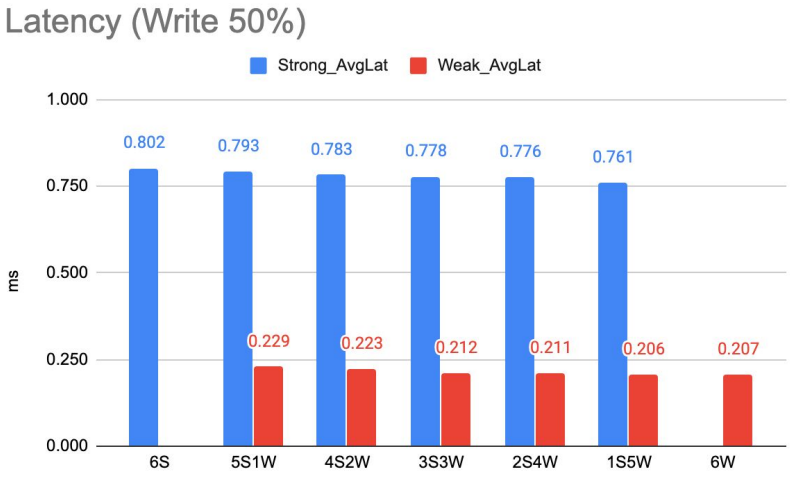
Design2 - Short-Circuit Result: No Entanglement



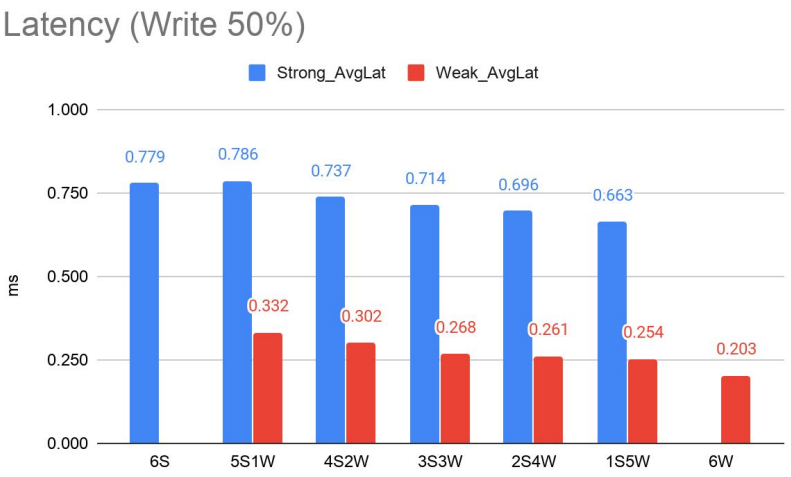
Short Circuit



Design2 - Short-Circuit Result: No Entanglement

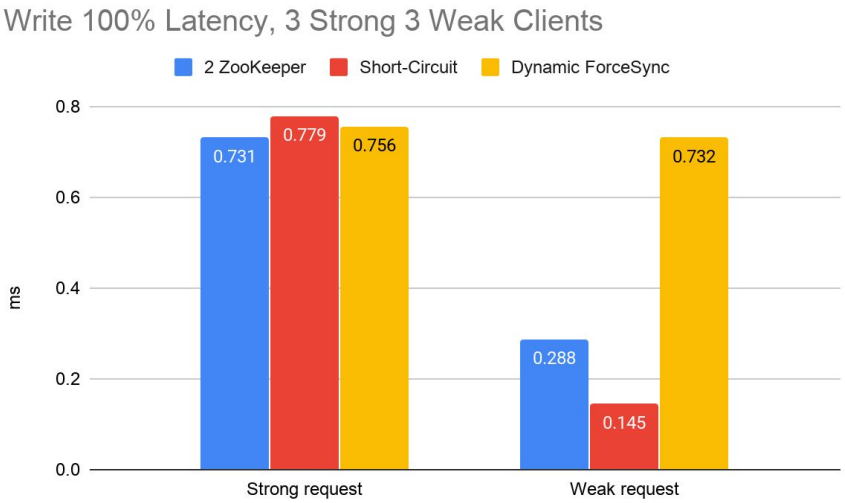


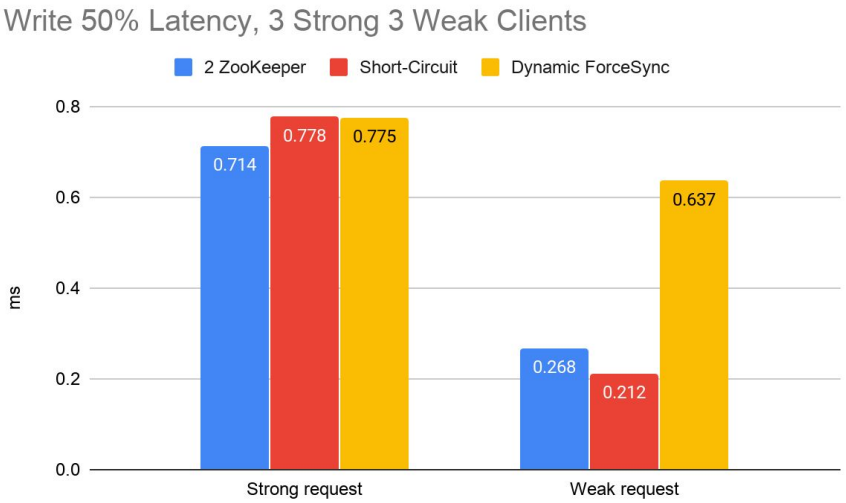
Short Circuit



DoubleZooKeeper

Evaluation - Write Latency Comparison

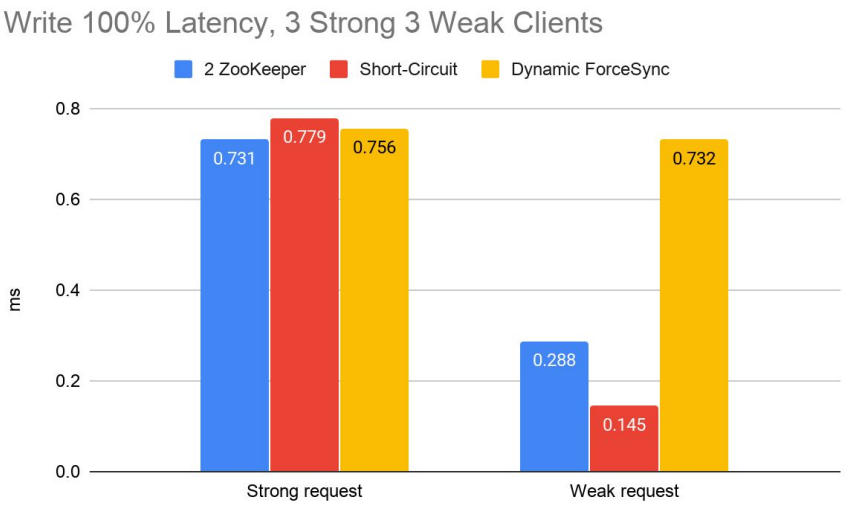




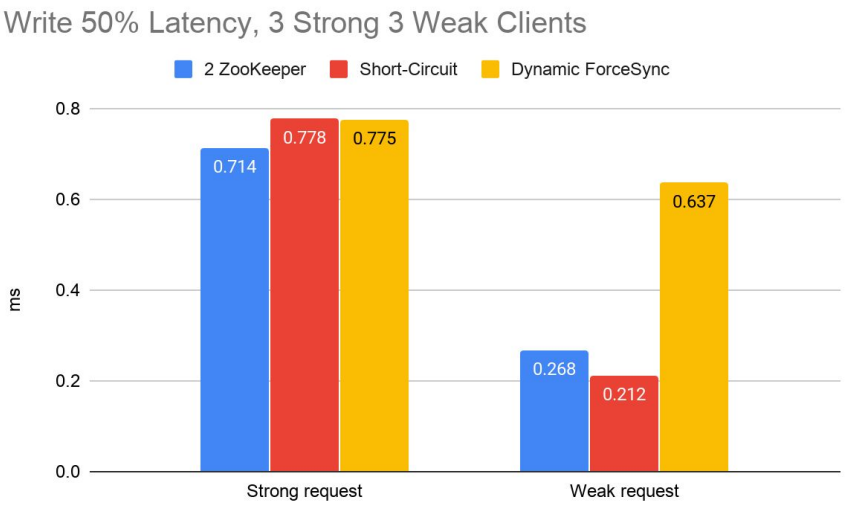
Evaluation - Write Latency Comparison

Short-Circuit vs 2 ZooKeeper

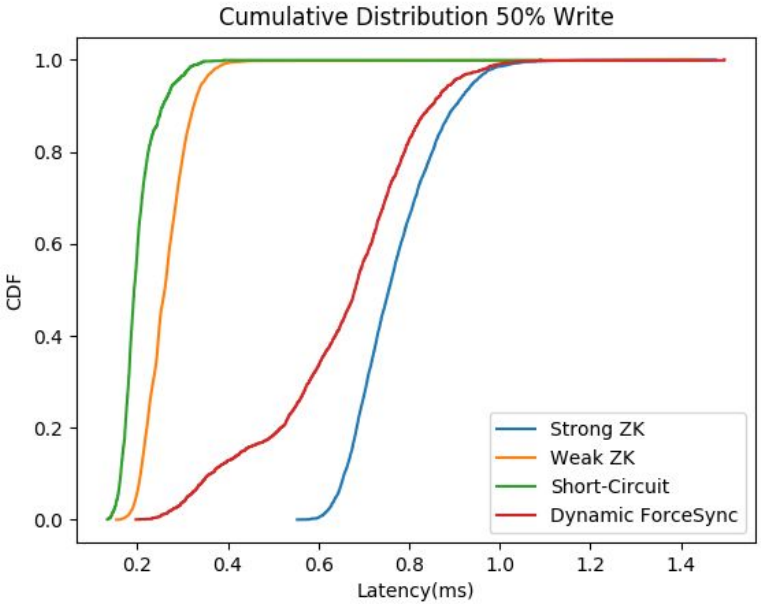
100% Write strong 0.93x weak 1.98x

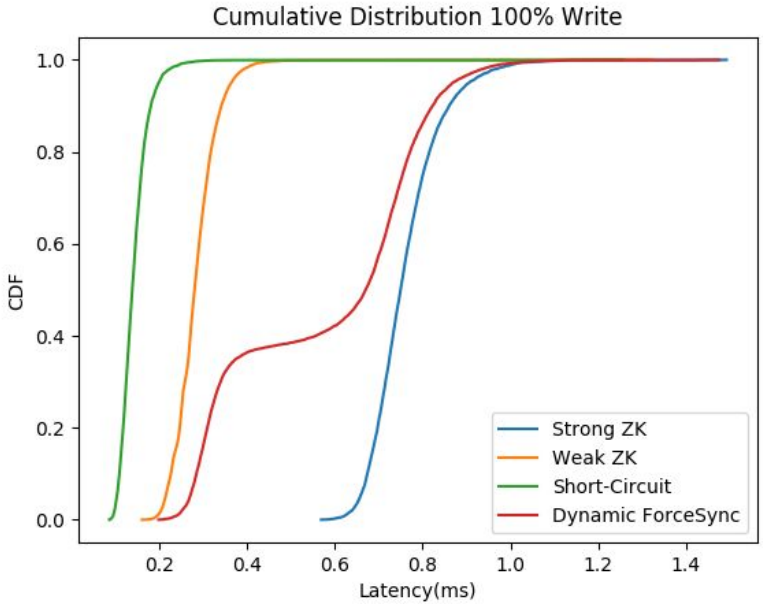


50% Write strong 0.91x weak 1.26x

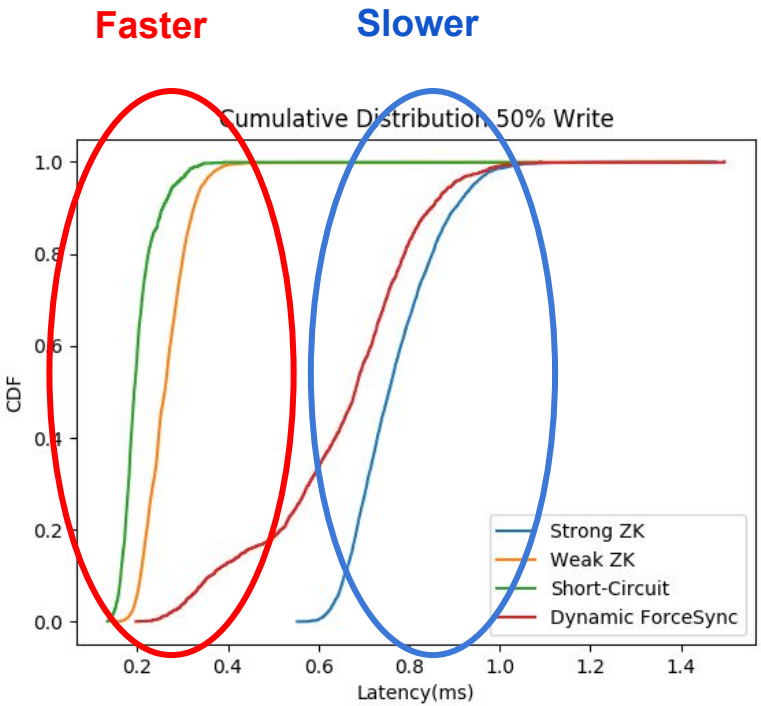


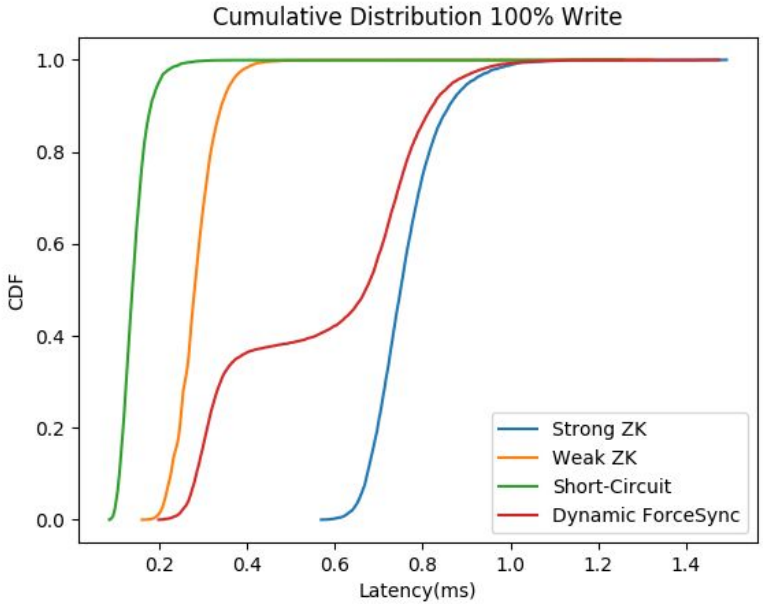
Evaluation - Write Latency CDF





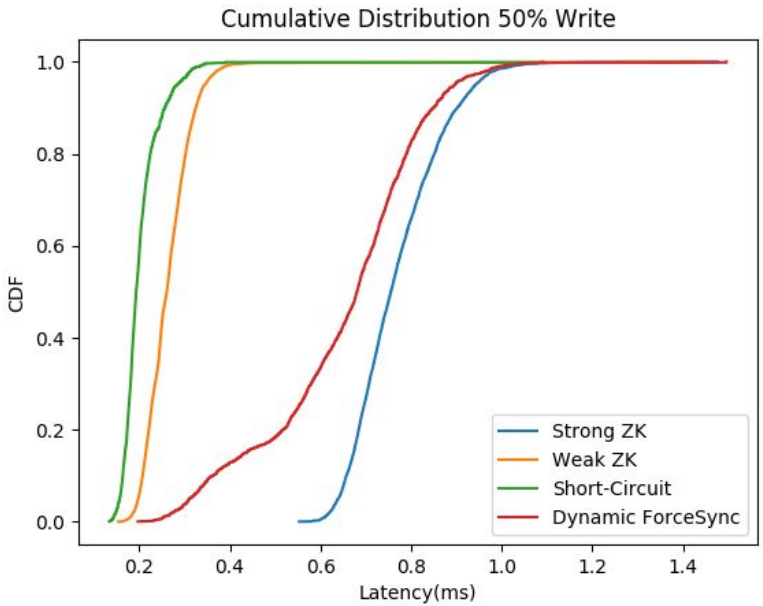
Evaluation - Write Latency CDF

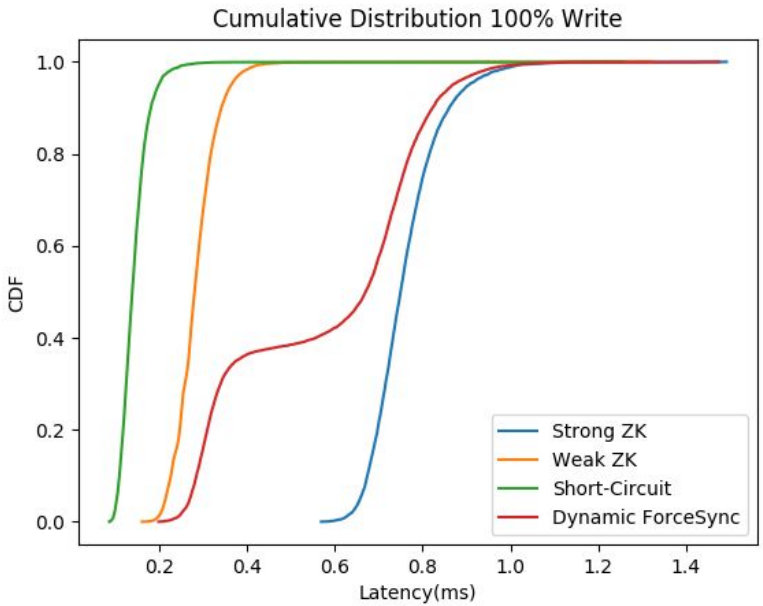




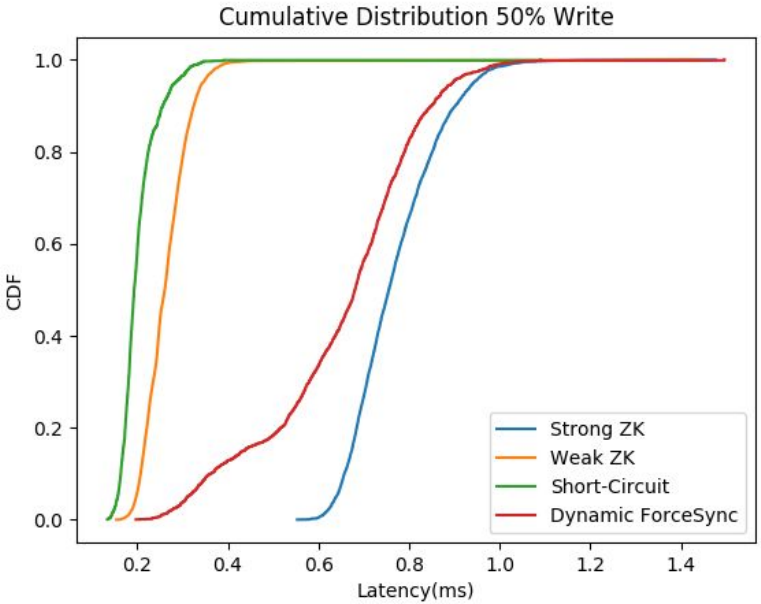
Evaluation - Write Latency CDF

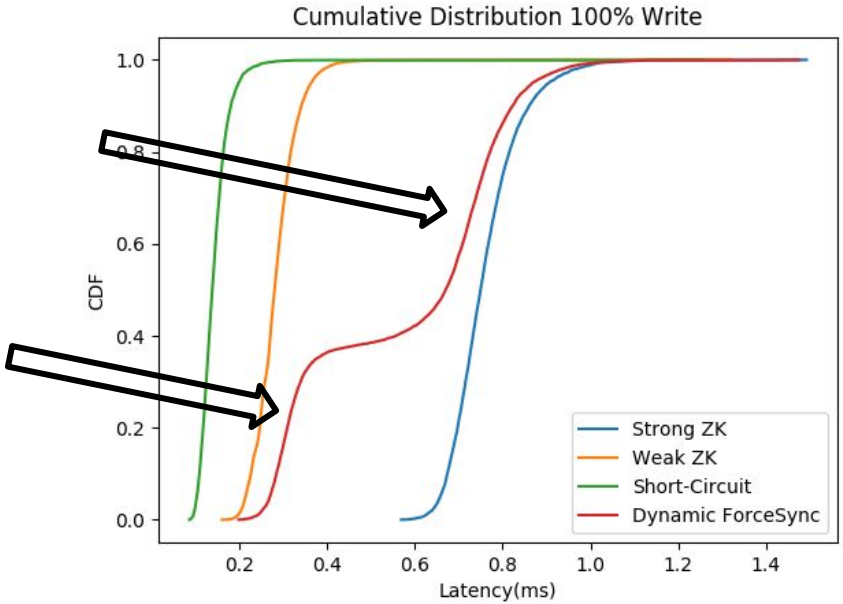
Short-Circuit < **weakZK** < **Dynamic ForceSync** < **strongZK**



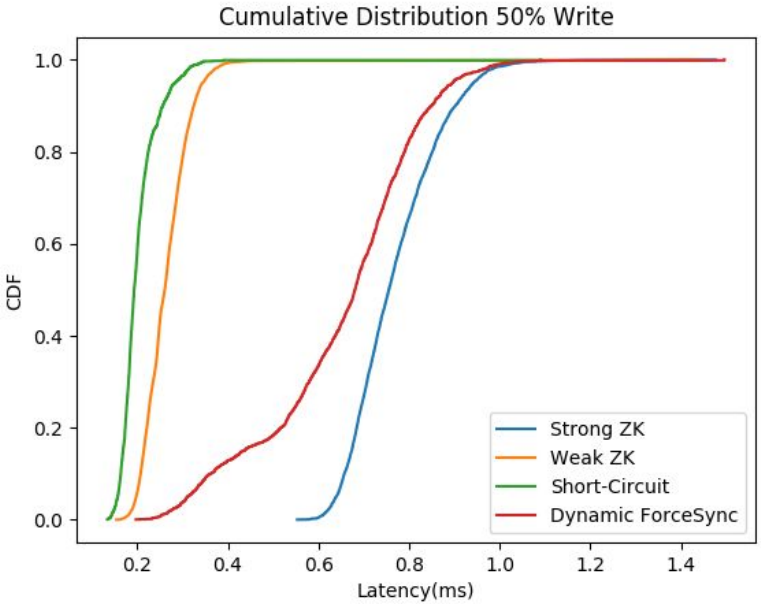


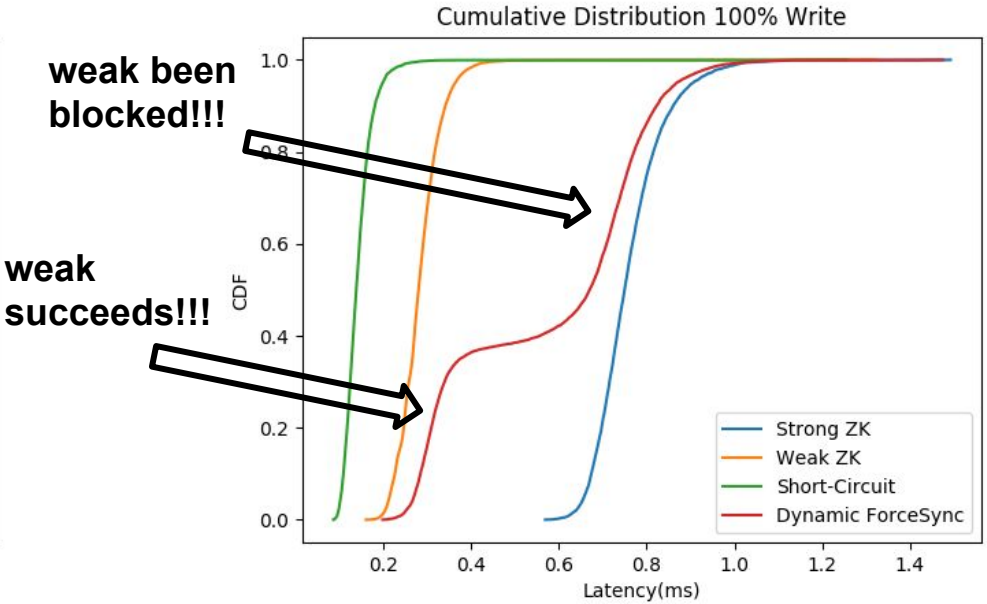
Evaluation - Write Latency CDF





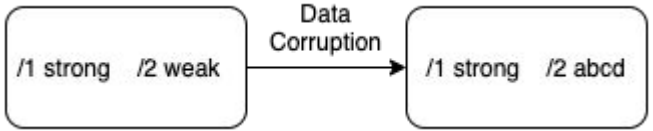
Evaluation - Write Latency CDF



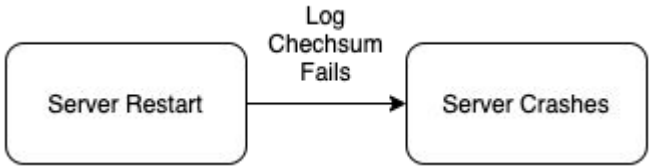


Reliability Separation

Data corruption



Original Zookeeper: Server shutdown when doing checksum

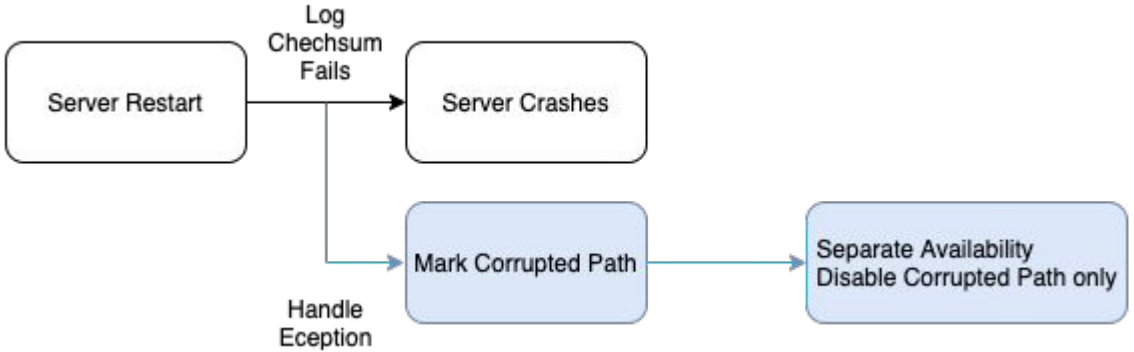


Reliability Separation

Data corruption

Original Zookeeper: Server shutdown when doing checksum

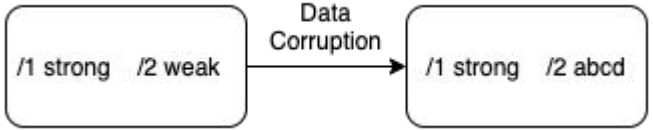
Ours: Separate different consistency level, serve **clean-only** level



-> Higher Availability

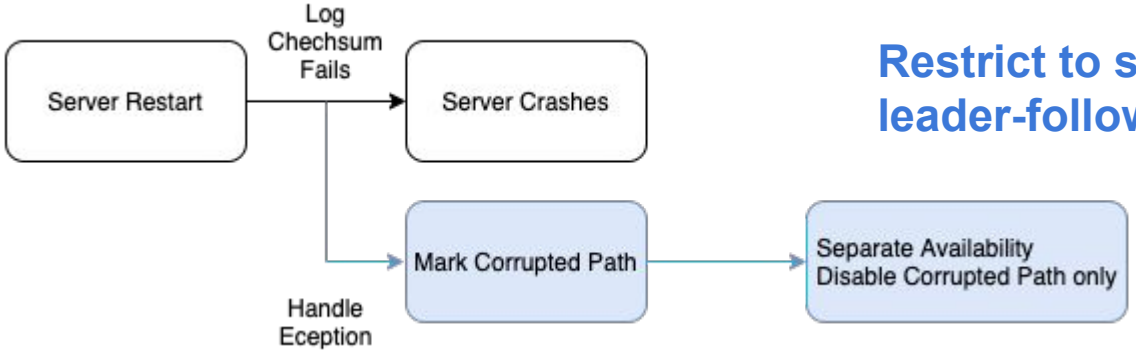
Reliability Separation

Data corruption



Original Zookeeper: Server shutdown when doing checksum

Ours: Separate different consistency level, serve **clean-only** level



Restrict to standalone mode now, leader-follower not yet

-> Higher Availability

Summary

- Provide multi-consistency on **single** ZooKeeper
  - Short-Circuit
    - Strong request: no harm
    - Weak request: **1.98x** faster comparing to 2 ZooKeeper
    - Dis-entanglement
- **Isolate availability** of different consistency levels
  - Only the corrupted namespace is down
    - restrict to standalone mode now

Thank you


A
Q