IONIA: High-Performance Replication for Modern Disk-based KV Stores

Yi Xu*, Henry Zhu *, Prashant Pandey, Alex Conway, Rob Johnson,
Aishwarya Ganesan, Ramnatthan Alagappan
(* = equal contribution)

UC Berkeley, University of Utah, Cornell Tech,
VMware Research, **University of Illinois Urbana-Champaign**

Write-optimized KV Stores

Write-optimized KV (WO-KV) stores based on LSM or B^e-trees

Widely used in many applications

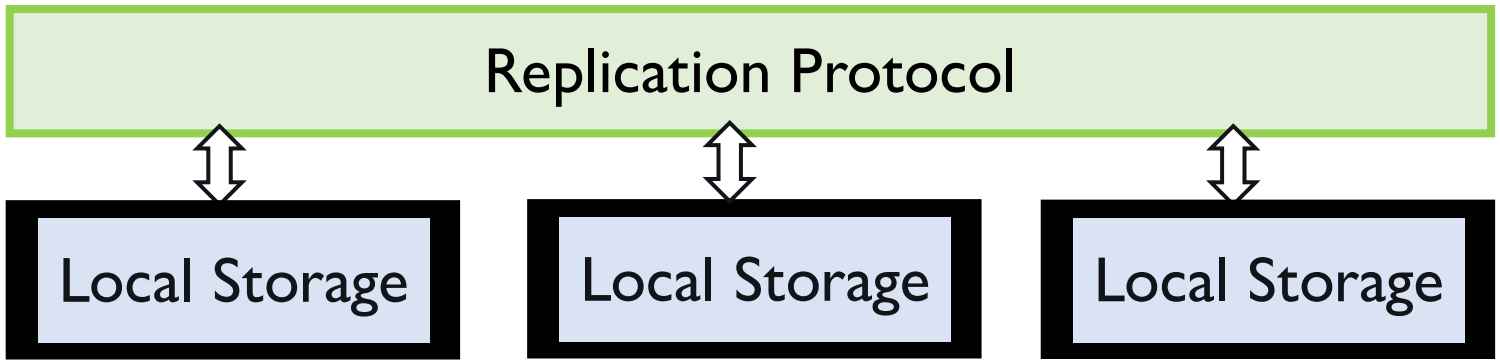
Optimized for SSDs

can offer high throughput and low latency

Fault Tolerance for WO-KV Stores

FT enabled by replicating the KV store on many machines

well-known replication protocols (Paxos, Raft)

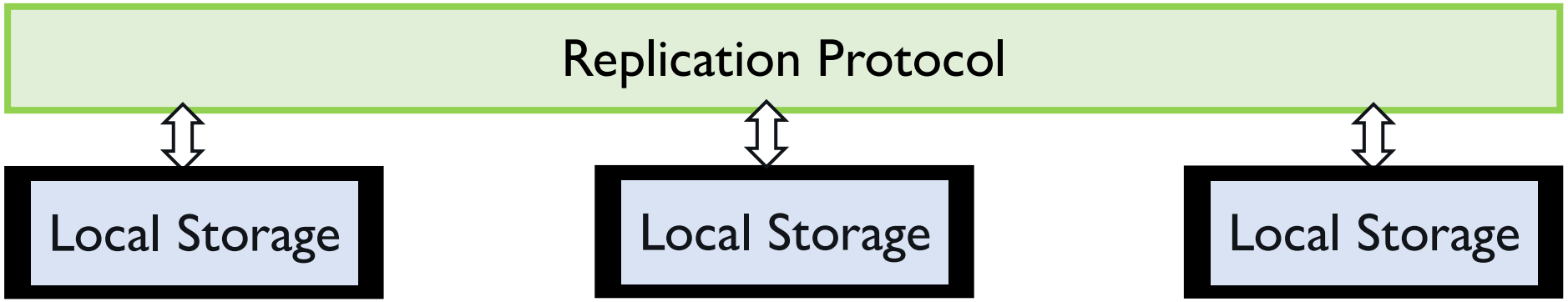


Black-box layering

Black-box layering → poor performance (high latency, low throughput)

This Work

We advocate for designing the replication layer by paying more attention to the local storage layer



IONIA – a new replication protocol that exploits unique characteristics of SSD-based WO-KV stores

Offers higher performance than traditional and existing optimized protocols

Outline

Introduction

Background and Motivation

IONIA Ideas

IONIA Design

Evaluation

WO-KV Stores

WO-KVs built atop LSMs or B^e-trees exhibit read-write asymmetry

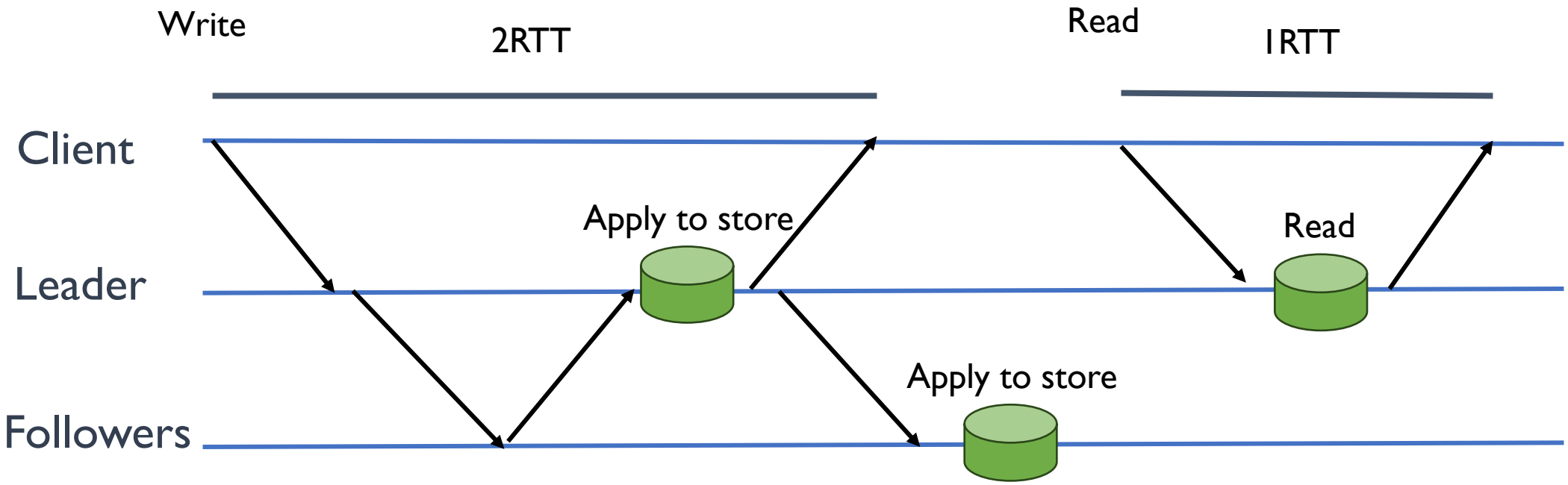
- writes limited by sequential bandwidth
- reads bottlenecked by random IOPs
- writes are faster than reads

Implication: avoid query-before-update for performance

Converts updates into blind writes: writes do not return execution results

- `Put` returns only an ack and does not check if key exists
- RMWs supported by recording functions that are later applied

Consistent Replicated KV Stores



Writes: high latency – 2 RTTs
throughput is limited (single-threaded execution for determinism)

Reads: 1 RTT
but does not scale (no reads at followers for consistency)
particularly bad for WO-KV because of read-write asymmetry

Many Prior Improvements, but they Fall Short

		Writes		Reads		Availability	
	Protocol/System	Latency	Tput	Scale?	Latency	HA Writes	Slowness tolerant?
Writes	Unreplicated	1 RTT	High	No	1 RTT	No	No
	Paxos, Raft	2 RTT	Low	No	1 RTT	Yes	Yes
	Parallel Execution	2 RTT + Δ	High	No	1 RTT	Yes	Yes
	Low-latency writes	1 RTT	Low	No	1 RTT	Yes	Yes
Reads	Gaios	2 RTT	Low	Yes	2.5 RTT + Δ	Yes	Yes
	CRAQ, Hermes	n/2 RTT (2RTT)	High	Yes	Low	No	No
	IONIA	1 RTT	High	Yes	Low	Yes	Yes

Ideal: match unreplicated write performance, scale reads with low latency, and tolerate failures and slow replicas

Outline

Introduction

Background and Motivation

IONIA Ideas

IONIA Design

Evaluation

IONIA Ideas

IONIA uses 3 ideas to improve over existing work

1. Deferred Parallel Execution With Immediate Durability
2. Decoupling Scalability from Locality
3. Client-side Consistency Checks

1 Deferred Parallel Execution With Immediate Durability

IONIA executes only non-conflicting writes concurrently for safe parallel execution like prior work

But avoids high latencies by exploiting interface semantics of WO-KV stores

writes do not return execution result to avoid query-before update

Based on our prior work on Skyros [1]

execution deferred to background with immediate durability

but with parallel execution

Result: 1RTT writes with high throughput due to parallel execution

[1] Exploiting Nil-Externality for Fast Replicated Storage SOSP 21

Conflation of Scalability and Locality in Existing Systems

Existing scalable, low-latency read protocols conflate scalability and locality

in-memory stores: network is the bottleneck

current protocols ensure read at a replica is local (no additional message)

Critical implication: to consistently read from anywhere, writes must be synchronously replicated to all replicas (not just a quorum)

High write latencies

Cannot tolerate failures → must reconfigure

Cannot also tolerate slow replicas

Fundamental tradeoff: write availability vs. read latency

2 Decoupling Scalability from Locality

Key insight: In SSD-based WO-KV stores, SSD random IOPS is the bottleneck, not the network messages

Reads can scale even if they are non-local
as long as the additional messages access only in-memory state (not the SSD) on other replicas.

That is, scalability can be decoupled from locality in SSD-based stores

Decoupling Scalability from Locality in IONIA

IONIA writes only to a quorum for availability and slowness tolerance

To handle stale replicas, IONIA sends a meta query to the leader

meta query helps identify stale reads from followers

Key requirement for scalability: meta query throughput on leader must exceed collective SSD random IOPS

True in practice:

collective SSD random IOPS = # replicas * SSD IOPS = 5 * 850K

our unoptimized meta query can run at 12 M queries/s

3 Client-side Consistency Checks

Meta queries help consistency and scalability but not low latency

Idea: clients send the meta query and read request in parallel

Problem: leader does not know the status of the follower when the read is performed

Solution: leader returns enough info about key being read, clients validate the follower result

Result: scalable reads in 1 RTT without impacting availability

Outline

Introduction

Background and Motivation

IONIA Ideas

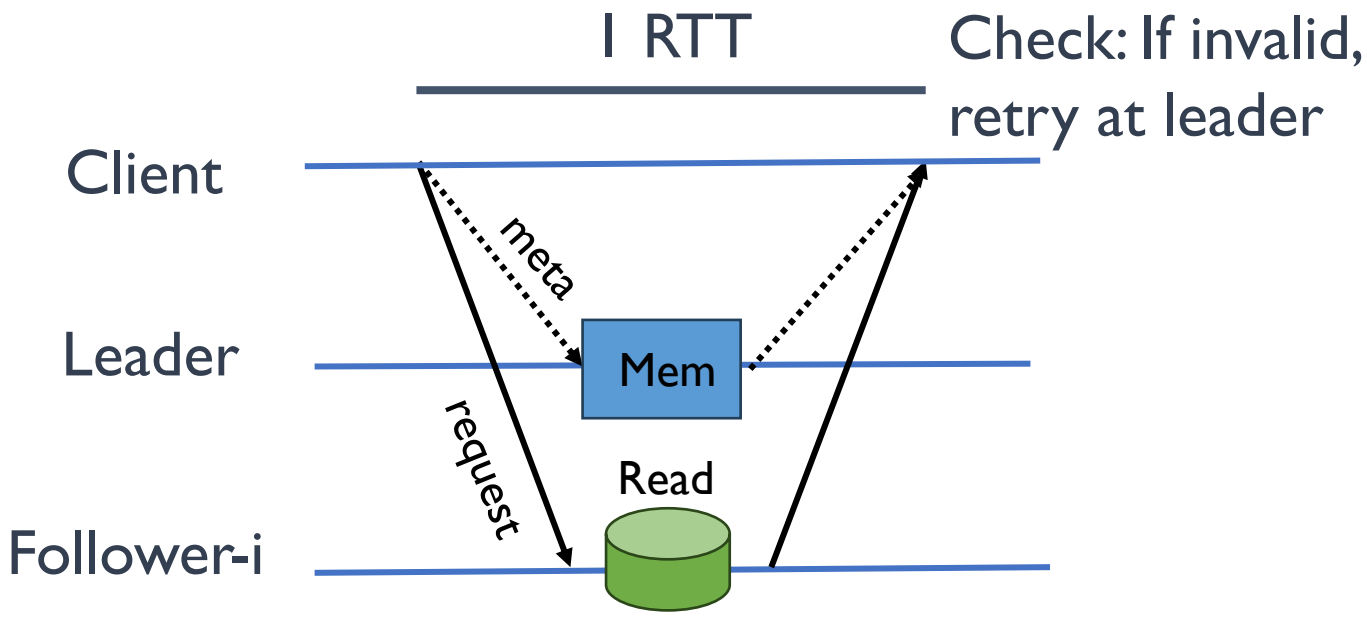
IONIA Design

Evaluation

Read Path

Focus on scalable, 1RTT reads

writes and leader reads in paper…



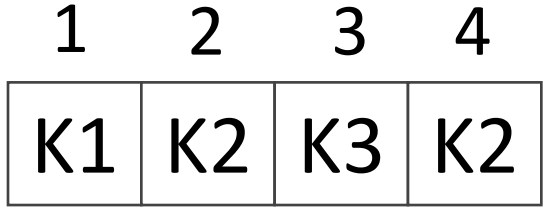
Follower returns data
And index it last applied to store

Leader returns the last index that modified the key

Client-side check:
if applied >= last_update_index(key)
then
follower data is latest
else
stale; needs to retry

Compact Histories

Leader must know latest index that modified every key

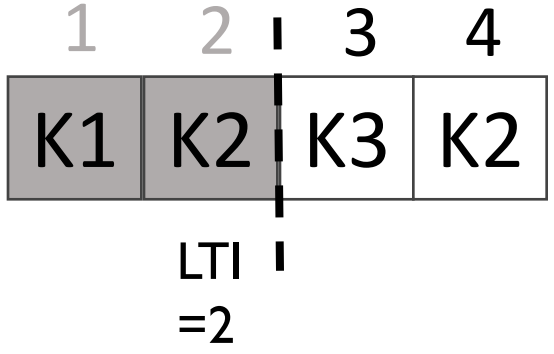


Example: sequence of writes: K1, K2, K3, K2

Can get the latest index that modified the key

Challenge: impossible to keep in memory for large datasets (condition for scalability)

Solution: maintain a compact history of recently written keys



Leader cannot get info for K1 – What to return?

IONIA returns Last Trimmed Index (LTI)

Correct: LTI >= actual modified index

Efficient: Lazily trim after followers have applied entries

Outline

Introduction

Background and Motivation

IONIA Ideas

IONIA Design

Evaluation

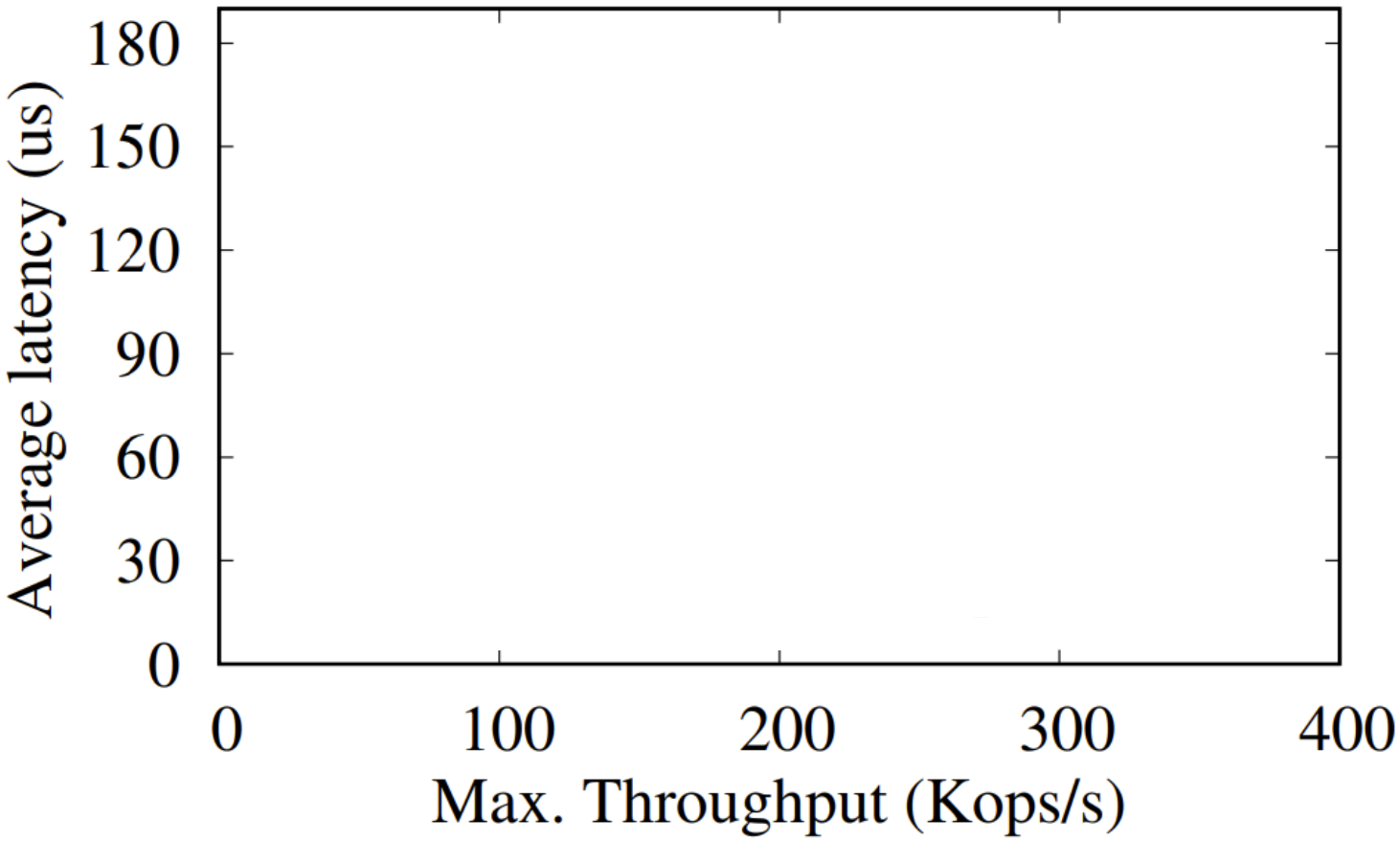
Evaluation

Replicate SplinterDB with IONIA

Compare IONIA against the following baselines:

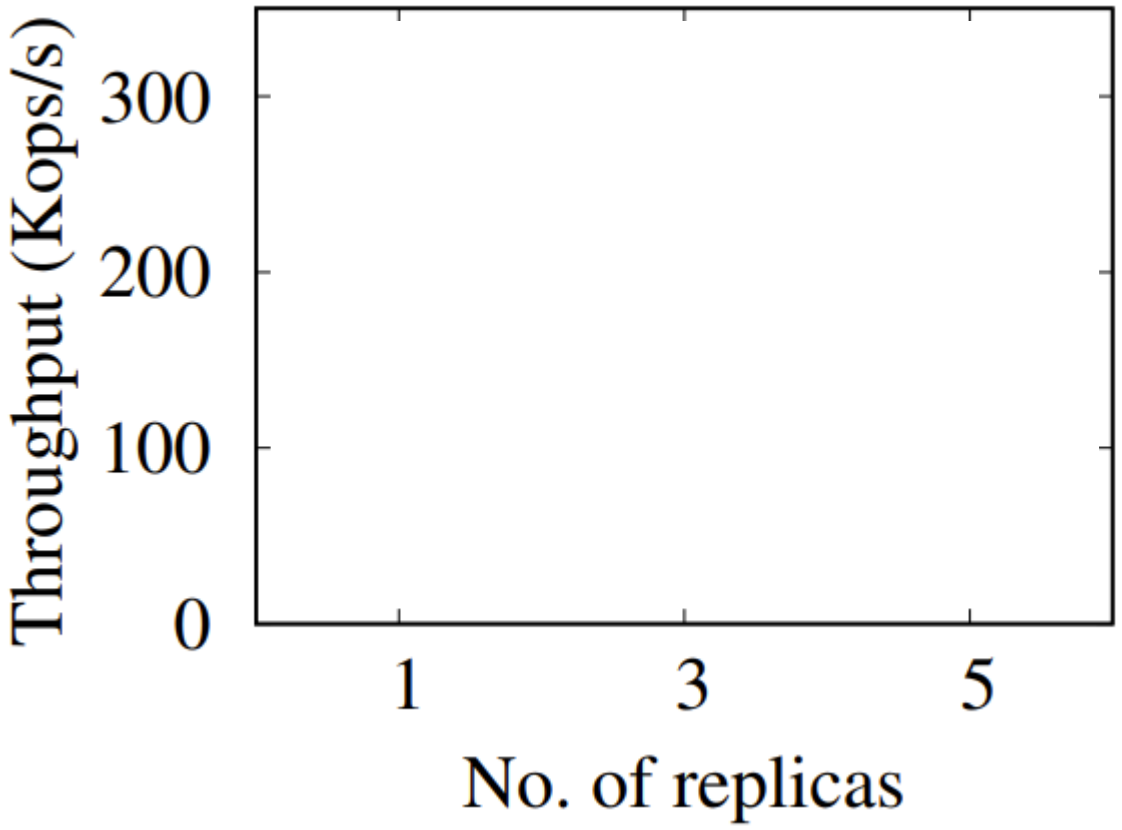
- Unreplicated SplinterDB
- MultiPaxos (with batching)
- Paxos-PE: parallel-execution baseline
- Skyros: low-latency protocol (our prior work)

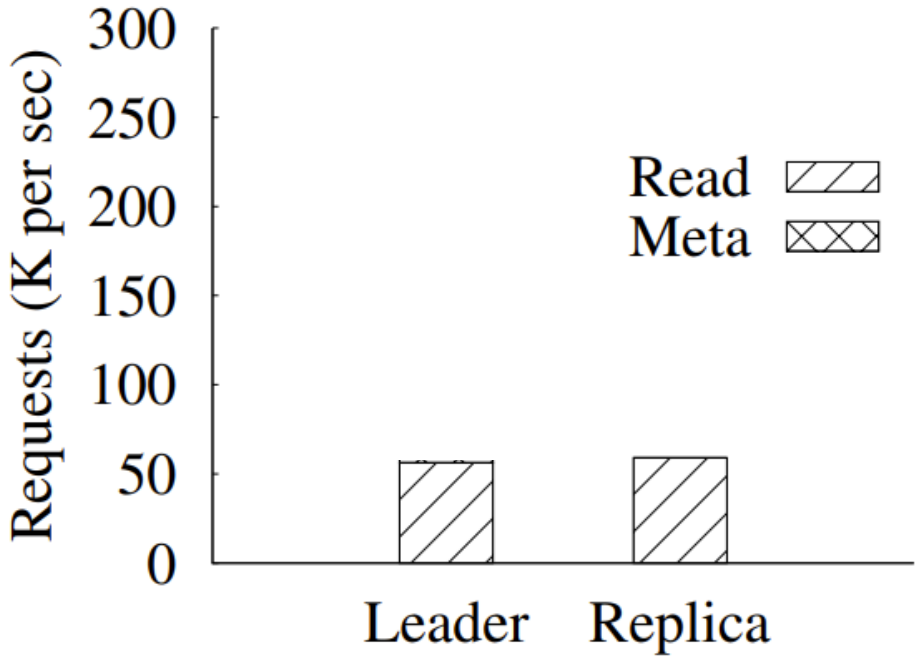
How does IONIA perform for write-only workloads?



IONIA offers low latency and high throughput like unreplicated due to deferred parallel execution with immediate durability

How does IONIA perform for read workloads?





IONIA scales reads despite decoupling locality and scalability due to cheap meta queries

Additionally, client-side consistency checks enable mostly 1RTT reads

More in the paper…

Writes and leader reads

Failures

Correctness

Model checking

More evaluation

- Performance under failures
- Different data distributions
- YCSB and mixed workloads
- Comparison to unreplicated

Concluding Thoughts

Often, layers are combined in a black-box manner in distributed storage
This leads to poor performance

Our work shows how to design a more performant replication protocol by being more aware of the underlying storage

More opportunities to be explored!

Thank you!