SOSP 2026

TEXAS
The University of Texas at Austin

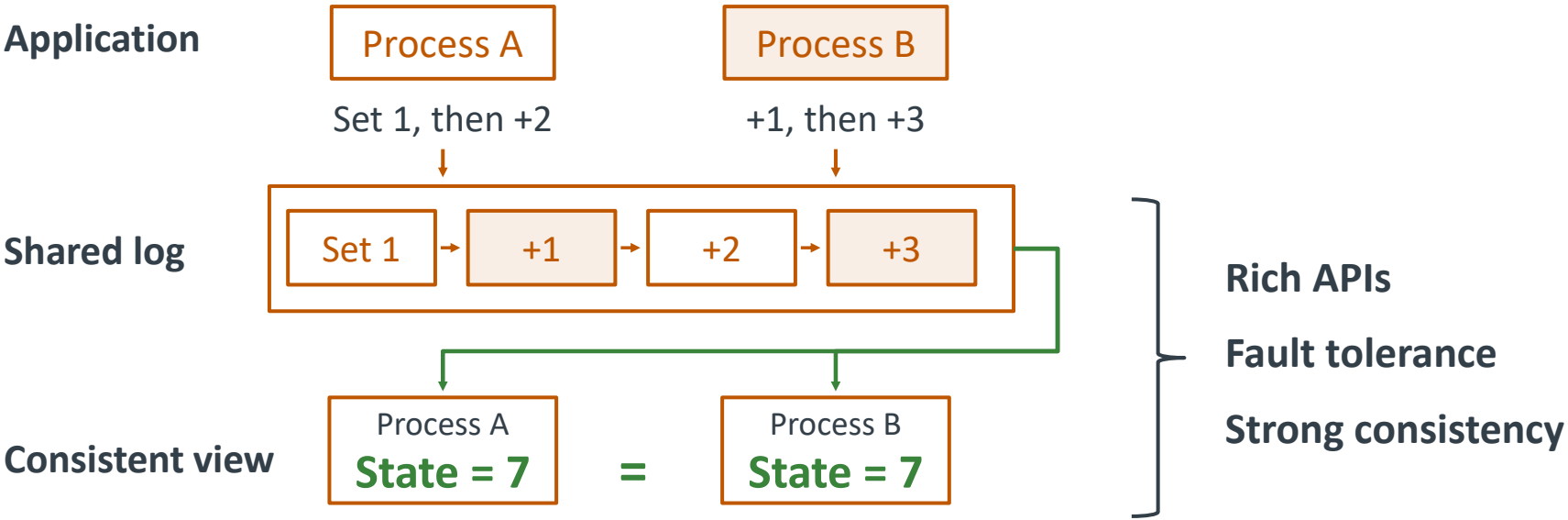
Borges: A Low-Latency Distributed Shared Log on a CXL Memory/SSD Hybrid

Haowei Chen, Yiming Xiang, Zhipeng Jia[‡], Yan Sun[†], Nam Sung Kim[†], Emmett Witchel
UT Austin, Anthropic PBC[‡], UIUC[†]

HAOWEI CHEN
The University of Texas at Austin

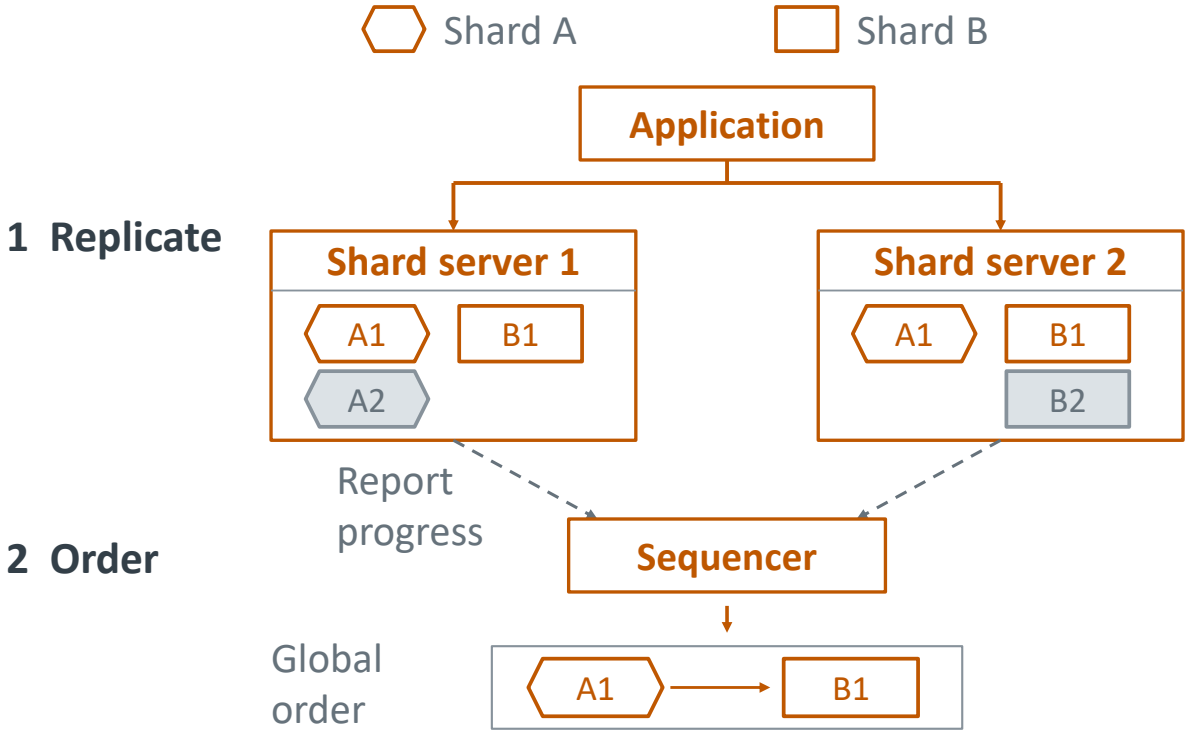
What is a shared log?

A durable, globally ordered history of updates.



Shared logs

Order only fully replicated records.



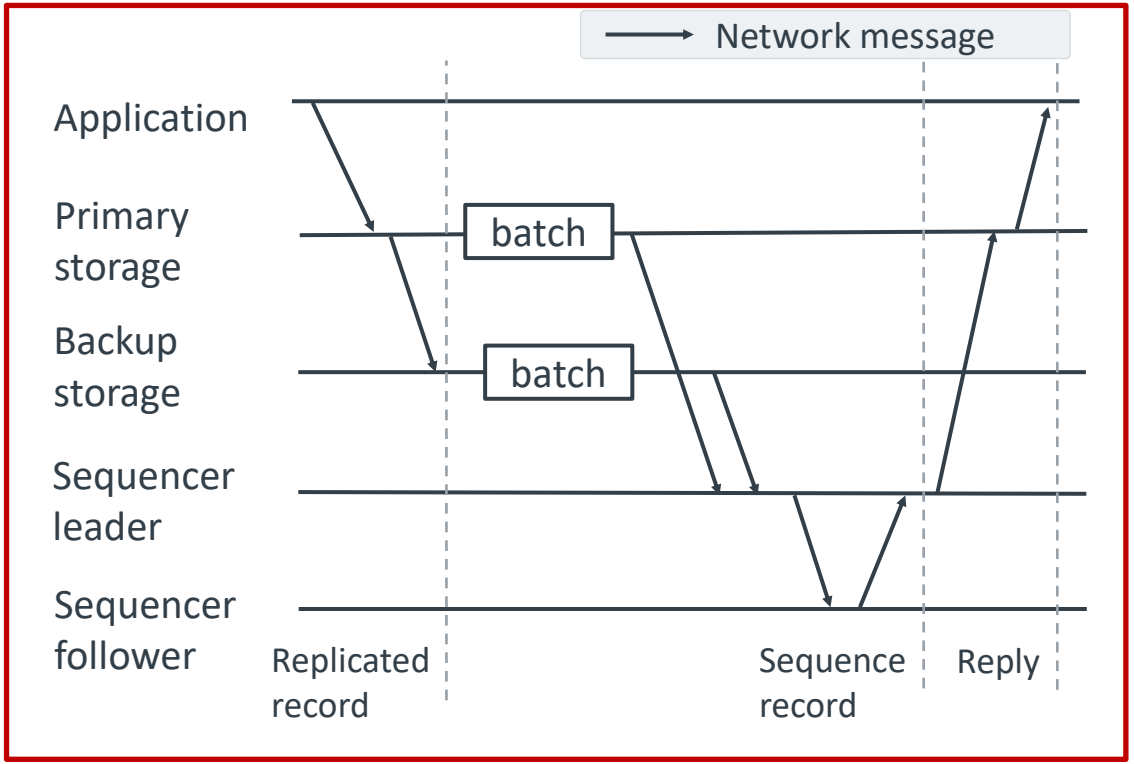
Borges (2026, this work)
SpecLog (OSDI 25)
LazyLog (SOSP 24)
Boki (SOSP 21)
Scalog (NSDI 20)

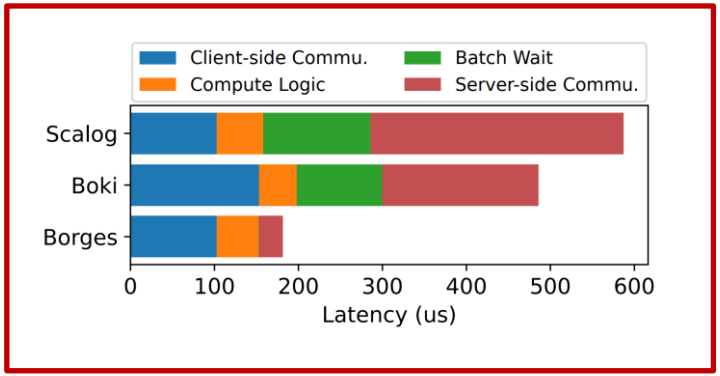
Fully replicated

Not fully replicated

Current shared logs have high latency

High network latency on append path



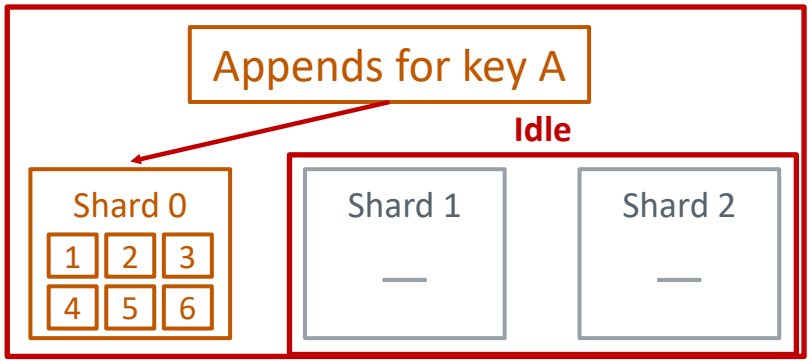


3.5 RTTs on each request's append path

The tradeoff: fast replay vs. load balance

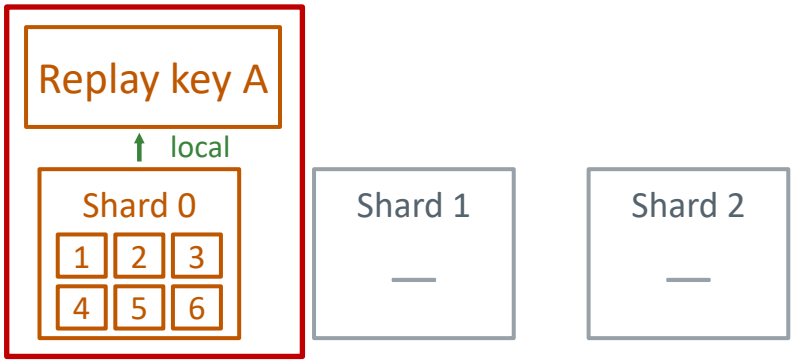
Single-shard placement: keep a key's records on one server.

1 Append key A



More shards do not fix the hotspot.

2 Replay key A on Shard 0 (read)



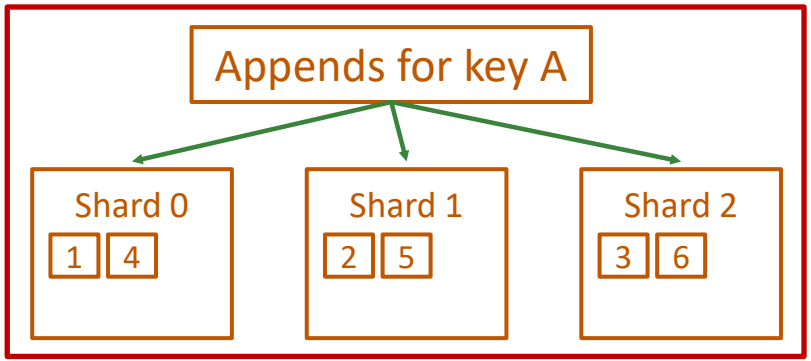
Fast replay: all records are local.

Takeaway: fast replay, but one hot shard.

The tradeoff: fast replay vs. load balance

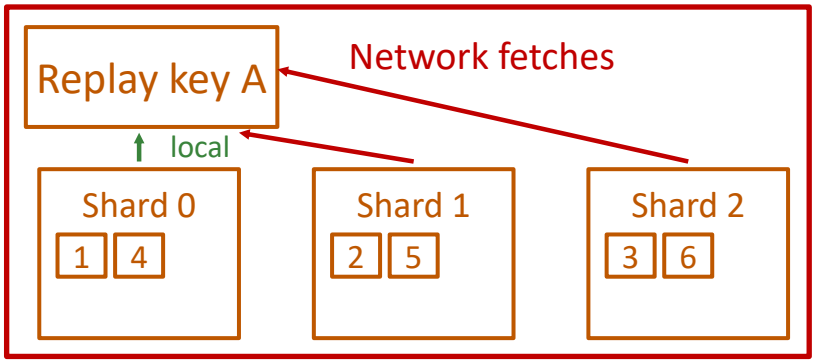
Cross-shard placement: spread a key's records across servers.

1 Append key A



Hot-key traffic is balanced.

2 Replay key A on Shard 0 (read)



Slow replay: remote fetches.

Takeaway: balanced load, but remote replay.

The opportunity of CXL-SSD (CMM-H)

CMM-H: shared DRAM buffer backed by persistent NAND.

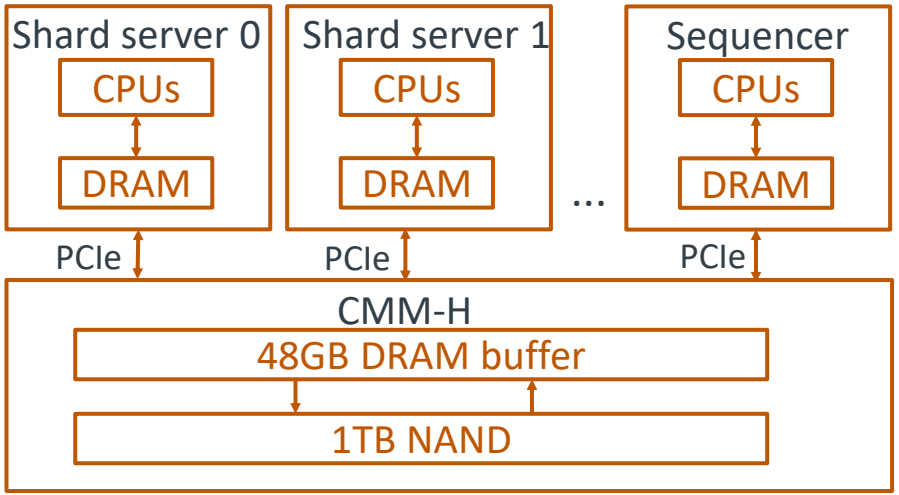
Byte-addressable loads/stores; 583ns measured access latency.

1 Low latency

- Read/write shared data instead of exchanging network messages.
- Efficient persistence

2 Balanced load + fast replay

Any host reads them directly from CMM-H, without fetching from remote shard servers.



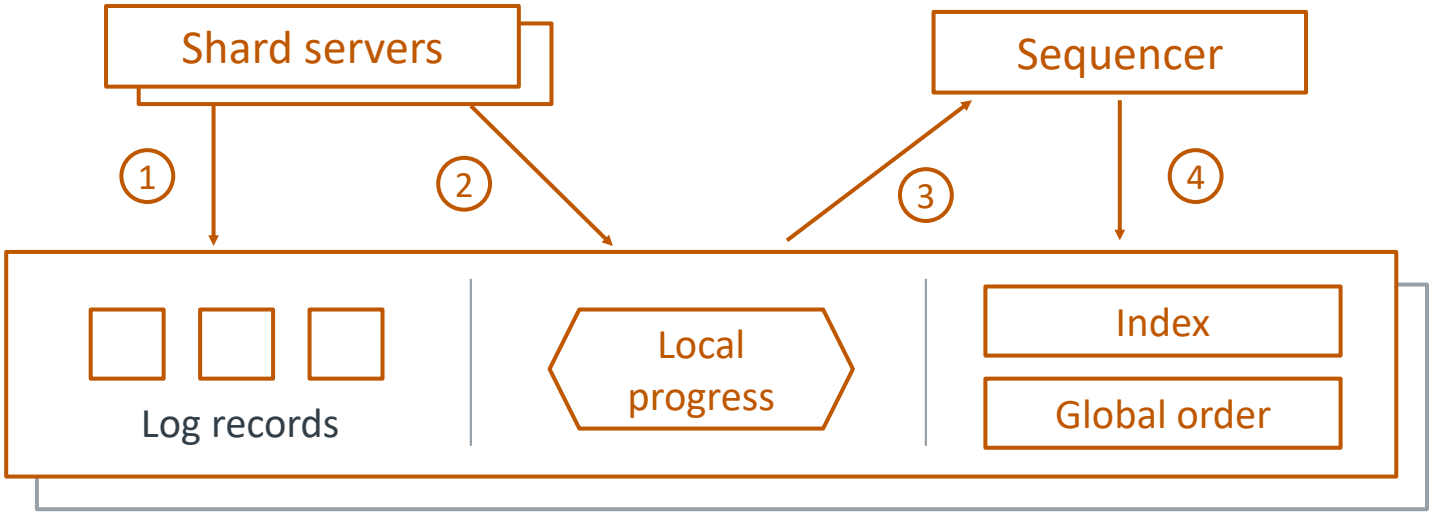
Overview of Borges

Data plane: shard server

1 Replicate records → 2 Report progress

Control plane: sequencer

3 Read → 4 Update order & index

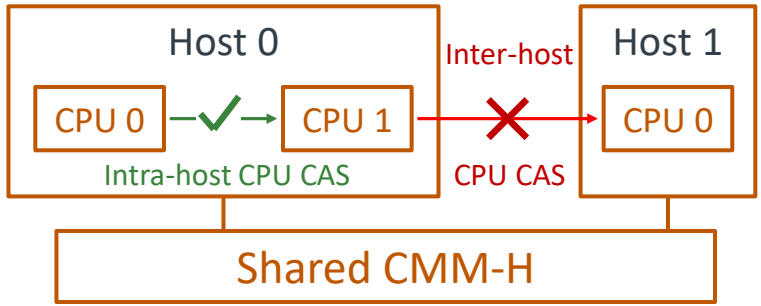


Challenge 1: synchronization

CMM-H provides no hardware cache coherence across hosts [Niagara 2].

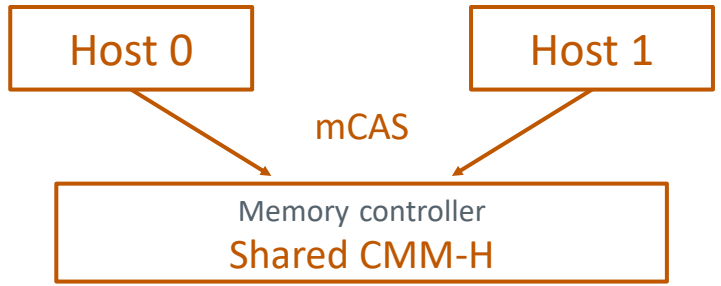
POSIX library, e.g., pthread_lock and futex, cannot work.

1 CPU CAS needs coherence



CPU CAS works only within a host.

2 mCAS serializes requests at the device



mCAS is slow and gets worse under contention

Takeaway: Need lock-free synchronization.

Challenge 2: consistent view

Direct access to CMM-H can expose an inconsistent log view.

Initial state: A = 0

1 Visible before commit

Reader sees: A = 1

Shared log crashes

After recovery: A = 0

An observed update disappears.

2 Visible out of order

Committed order: A+1 → A−2 → **A = −1**

Reader sees: missing → A−2 → **A = −2**

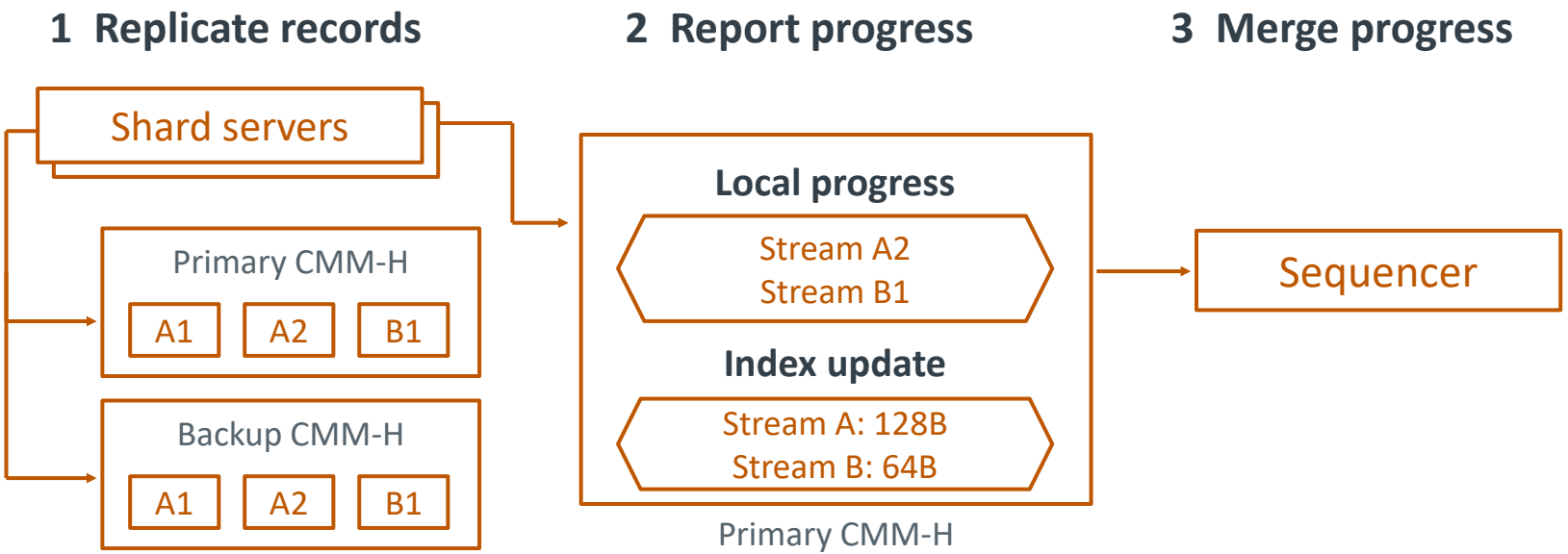
Not a prefix of the committed log.

Takeaway: Publish only committed records, in global order.

Append path: replication + sequencing

Stream: record history of a state, persistent in CMM-H

Index update: byte offset increase of a stream



Solution 1: Single-writer ownership for lock-free coordination.

Append path: publication

1 Update the back index

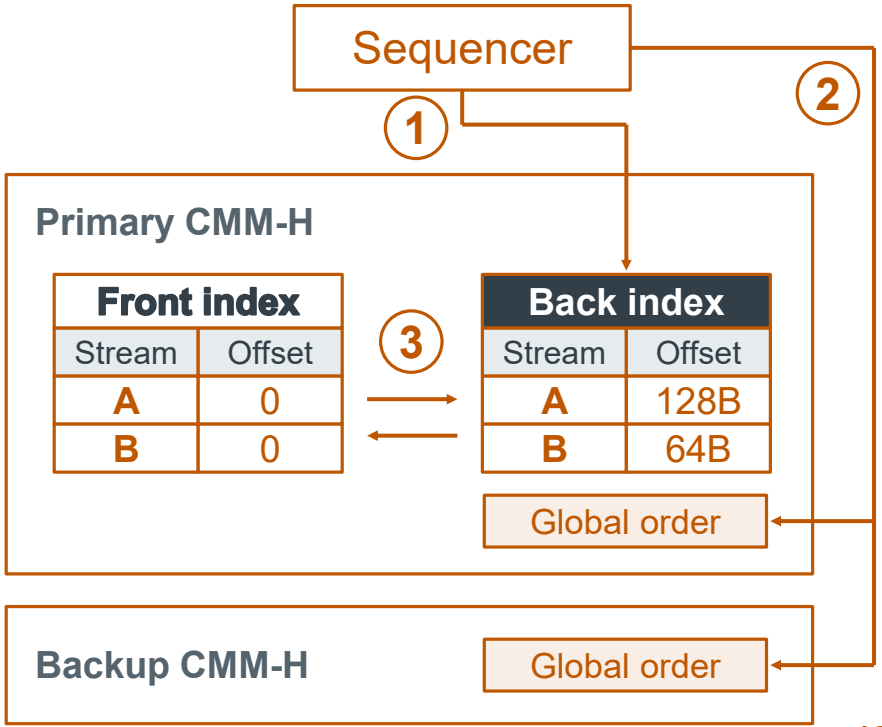
Readers keep using the front index.

2 Replicate the global order

Store the matching global order on both devices.

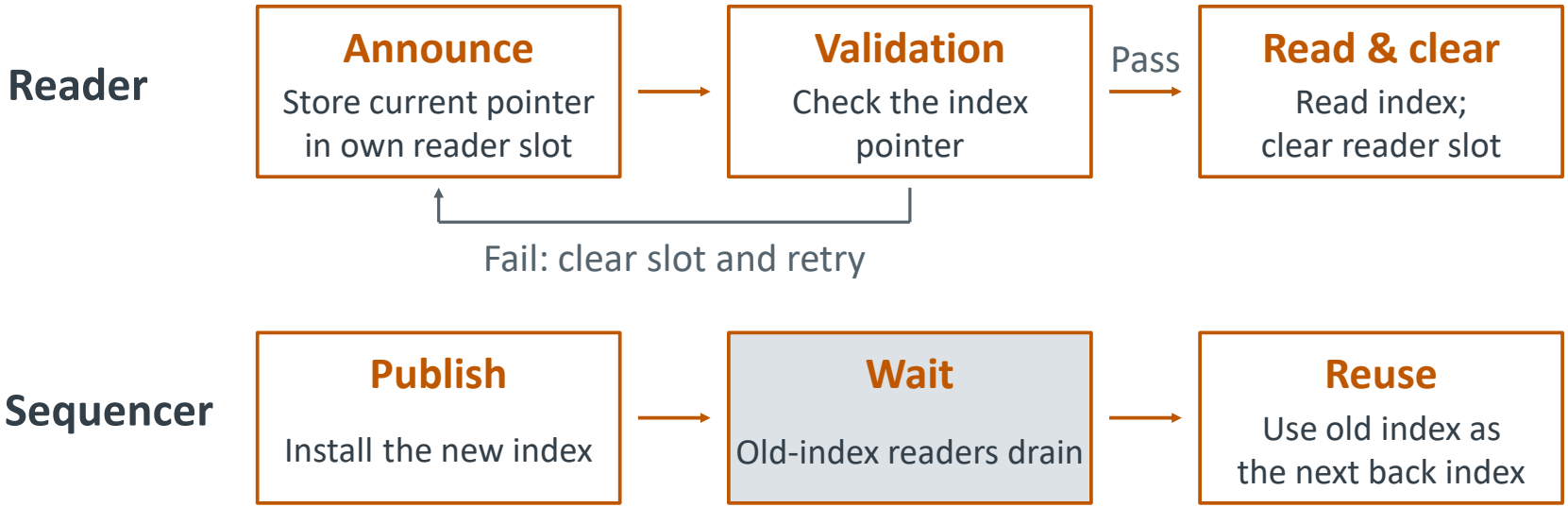
3 Swap the index pointer

The global order becomes visible only after the index swap.



Single-writer, multi-reader, lock-free publication

Publish immediately; only wait for old-index readers; leases prevent starvation.



Solution 2: readers only see consistent index and global order.

Failure model & recovery

Fail-stop model

Shard server failure

Recovery from the **global order + index**

- Local sequence number
- Stream boundaries

Sequencer failure

Sequencer is stateless

No extra actions needed

CXL device failure

Recovery from the surviving **global order.**

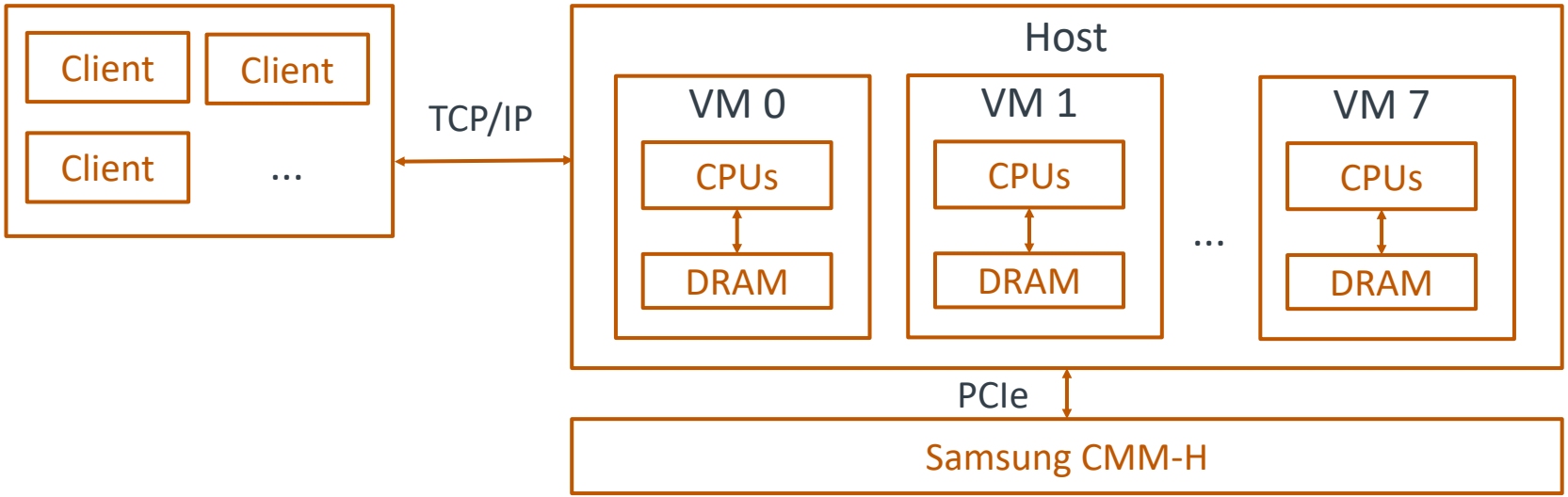
- Copy committed records
- Rebuild the index

Recovery anchor: latest complete global order

Evaluation: testbed

Emulated CXL-SSD pod

- A real Samsung CMM-H; partitioned to 2 CMM-H
- Run 8 VMs each with 5 cores



Evaluation: baselines

Improved baselines

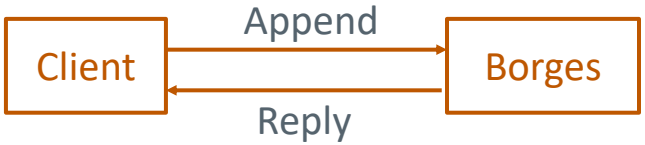
- Scalog-CXL
- Boki-CXL

Network transport replaced by message queues in CMM-H

Evaluation: microbenchmark breakdown

Mean latency (µs)

Parentheses: Borges vs. best competitor



	Replication (3.1× lower)	Sequencing (8.7× lower)	Publication (4.9× lower)
Borges	**26**	**13**	**27**
Scalog-CXL	81	116	132
Boki-CXL	125	113	171

Lock-free synchronization reduces all three phases.

Evaluation: end-to-end microbenchmark

Append microbenchmark

Parentheses: Borges vs. best competitor

Throughput (K req/s) ↑
(2.2× higher)



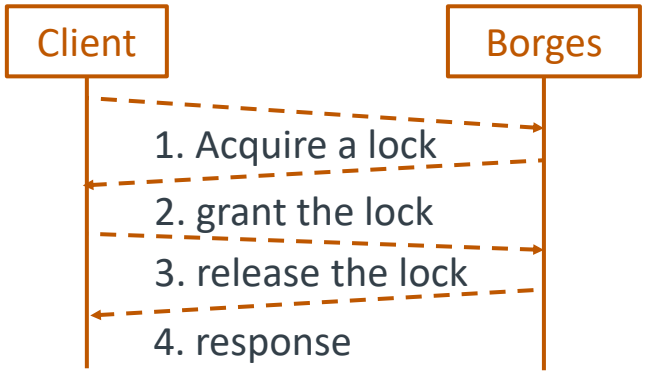
p99 Latency (µs) ↓
(2.8× lower)



Evaluation: lock workload

Exclusive lock benchmark

Parentheses: Borges vs. best competitor



Throughput (K req/s) ↑

(4.4× higher)

Borges	**37.12**
Scalog-CXL	8.50
Boki-CXL	0.38

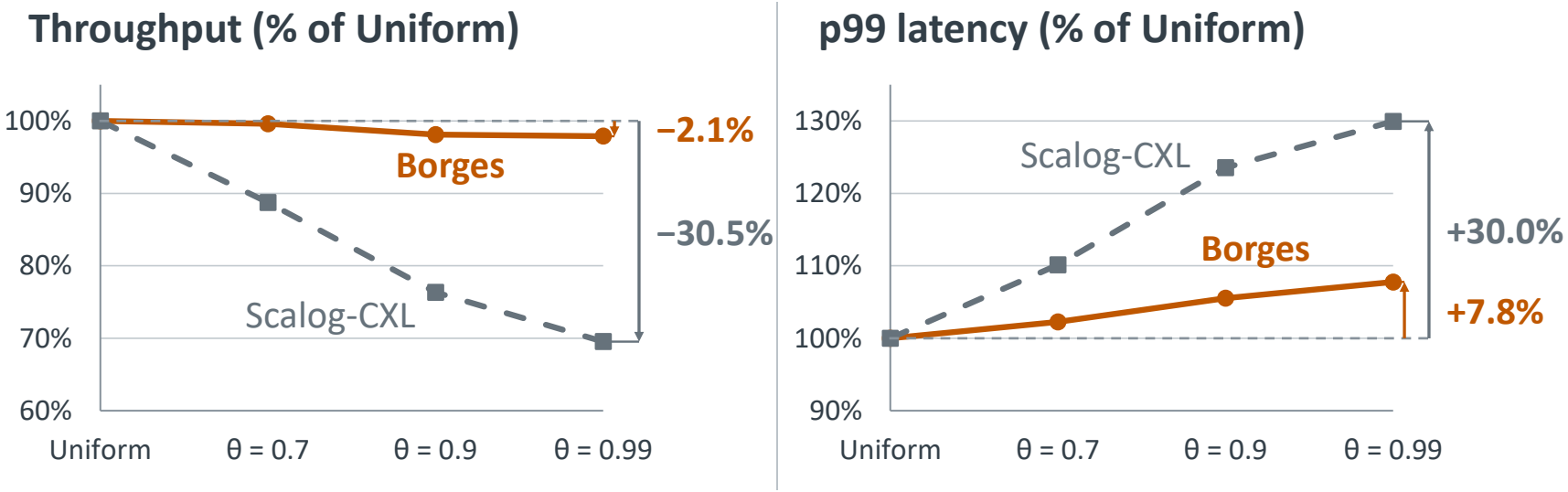
p99 Latency (µs) ↓

(3.3× lower)

Borges	**370.15**
Scalog-CXL	1239.19
Boki-CXL	10271.47

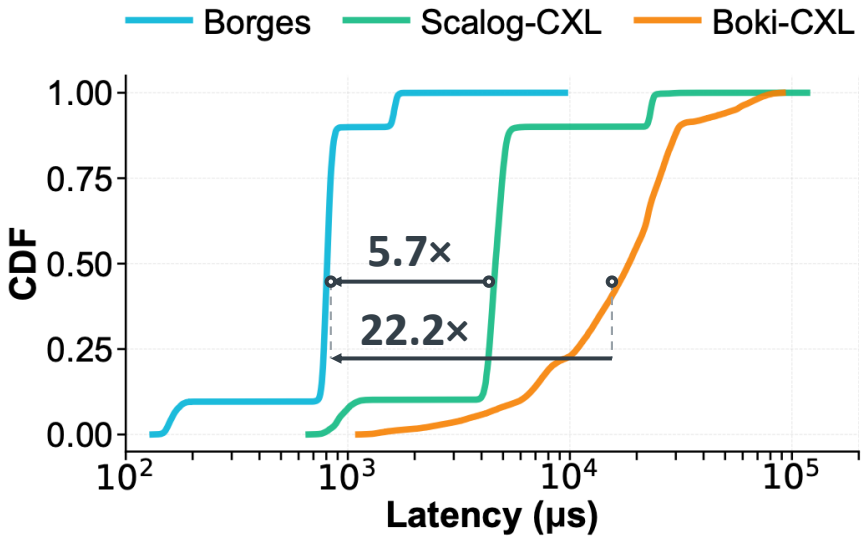
Evaluation: skewness

Increase Zipf skew for key selection



Evaluation: fault-tolerant workflows

Retwis



p50 latency speedup

Scalog-CXL, Boki-CXL

- External database for storing data
- The shared log for WAL

Borges

- Build data on top of the shared log
- Check duplication in the log

Conclusion

CXL introduces a new opportunity to shared-log systems

- Faster interconnect
- Shared storage

Borges is the first fault-tolerant shared log whose data and ordering state reside in CXL-SSD

- **Lock-free synchronization** to deal with no hardware coherence
- **Dual-index publication mechanism** to ensure consistent view

Questions?

Acknowledgement

Our work is supported in part by PRISM, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.