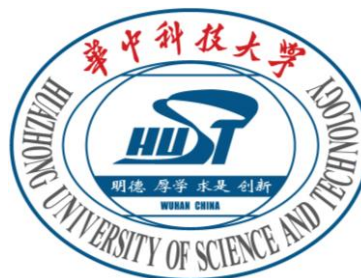




XTRA: Unifying Cache Coherence and Concurrency Control for Distributed Transactions in a CXL Pod

Zhijun Yang, Yu Hua, Ming Zhang, Menglei Chen, Xumin Chen, Aoyang Tong

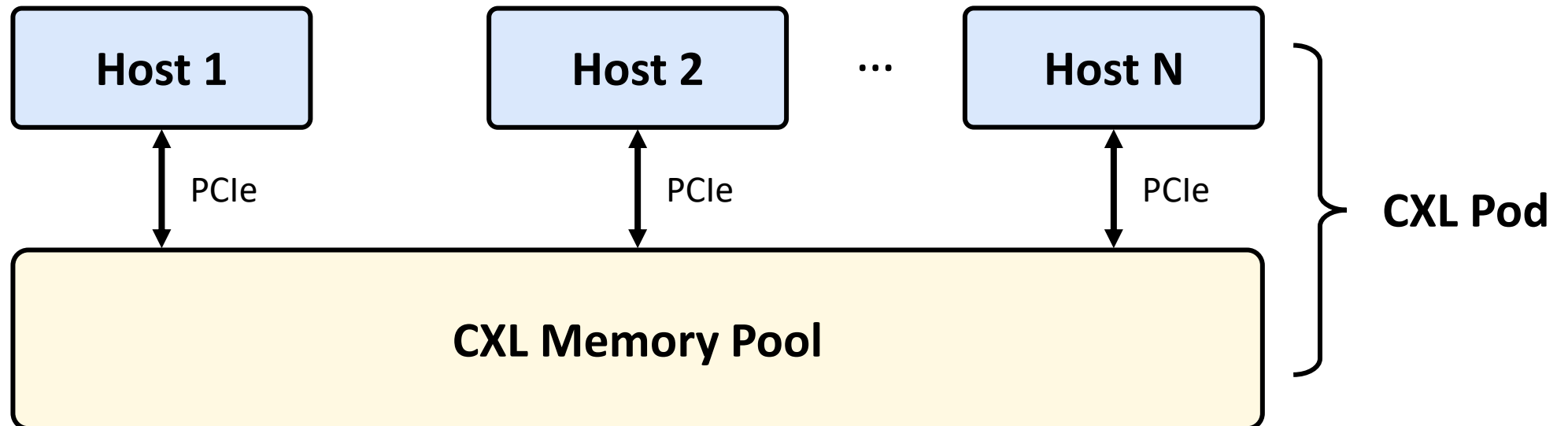
Huazhong University of Science and Technology



Compute Express Link (CXL) Memory



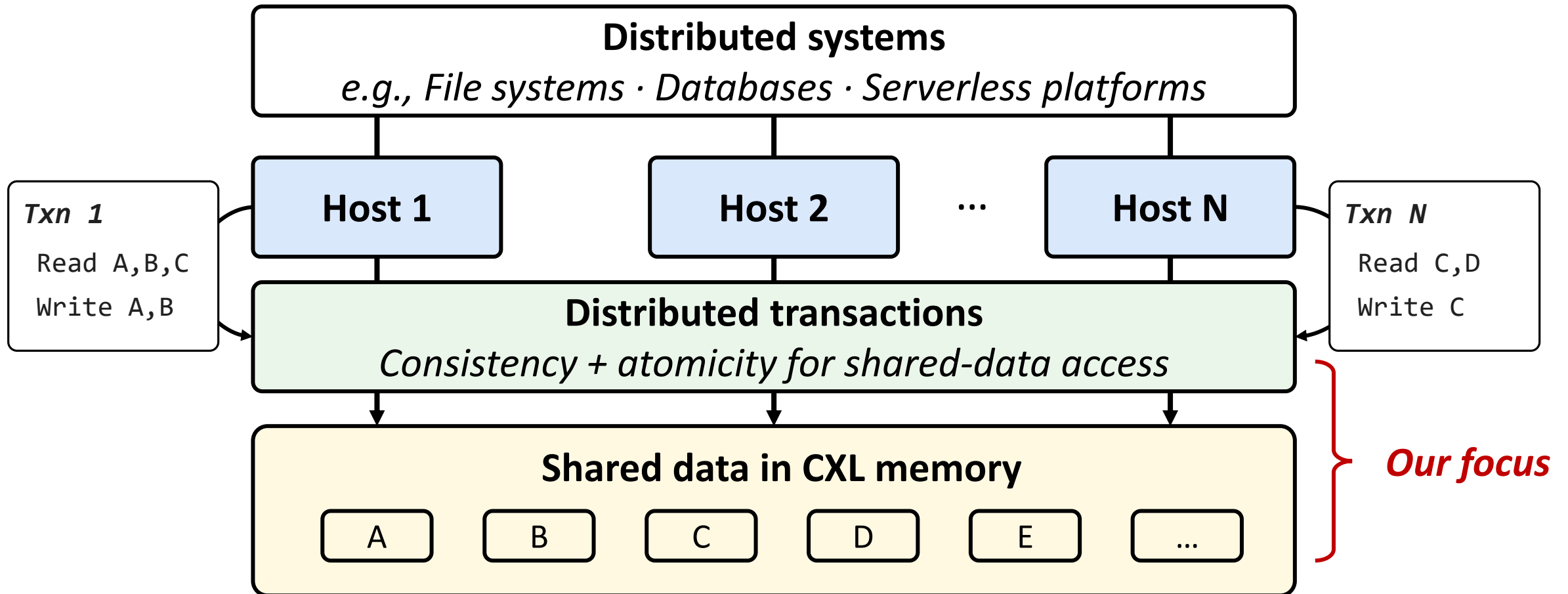
- CXL memory: attached to hosts over PCIe-based CXL links
 - Direct access via **CPU loads/stores**
 - Memory expansion (CXL 1.0), **pooling and sharing** (CXL 2.0 and 3.0)
- Our focus: **CXL pods**¹ (multi-host CXL memory pools)



[1] E. Witchel, “Challenges and Opportunities for Systems Using CXL Memory,” ASPLOS ’24 Keynote.

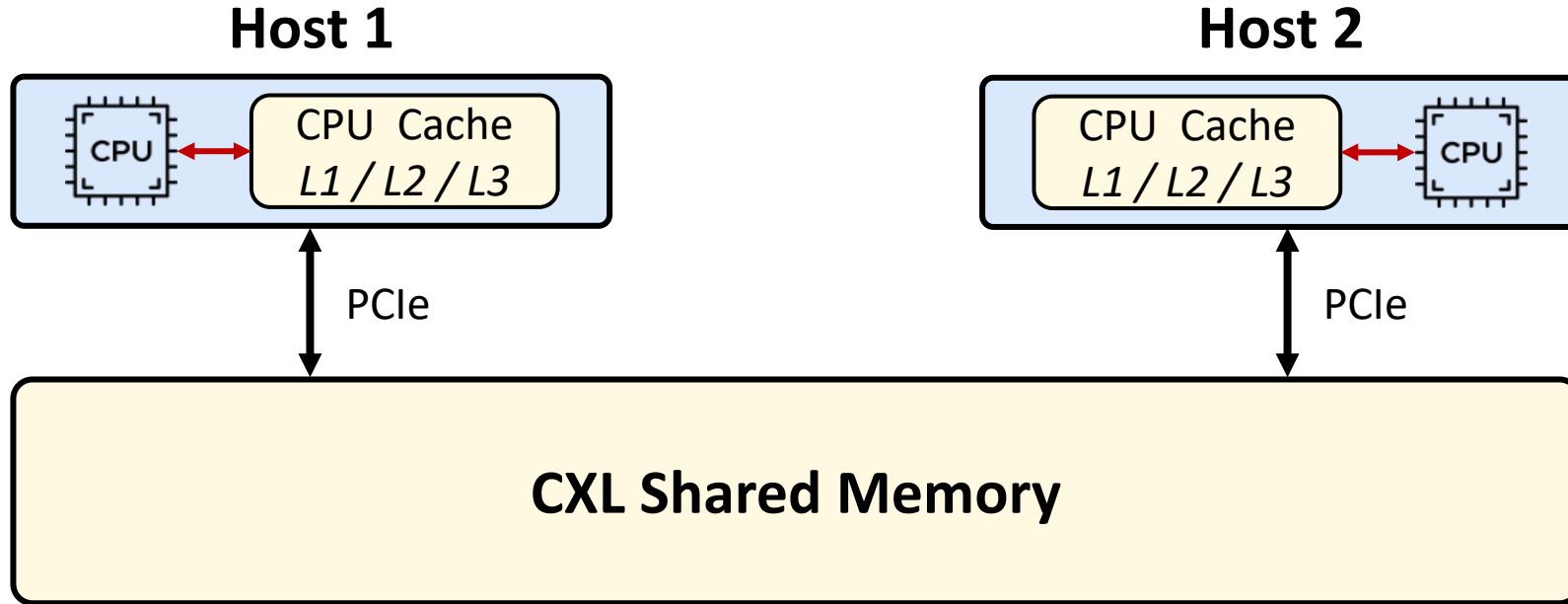
Distributed Transactions in a CXL Pod

CXL pod opportunity: **redesign** distributed systems around **shared memory**



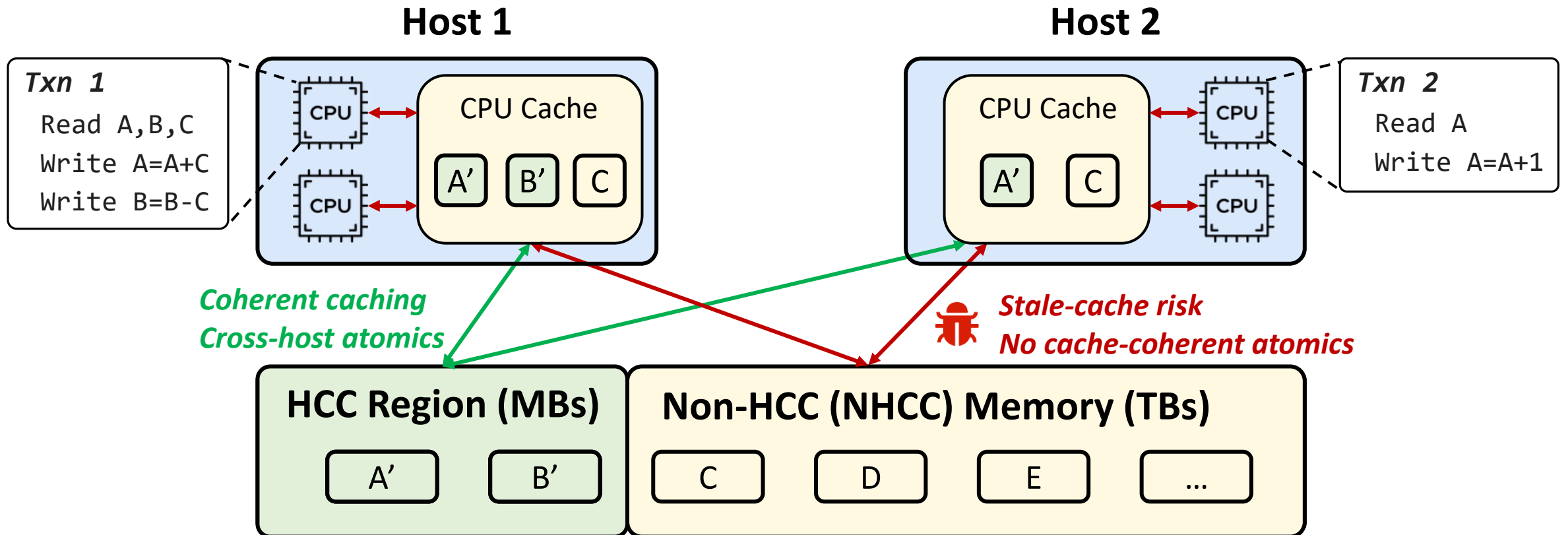
Limited Performance of CXL Shared Memory

- **Access** to shared CXL memory is **expensive**
 - Latency: **>2× higher** than local DRAM (230–600 ns¹)
 - Bandwidth: constrained by **PCIe links** (24.7 GiB/s per PCIe 5.0 ×8 link)
- Solution: hosts cache hot shared data in **CPU caches**



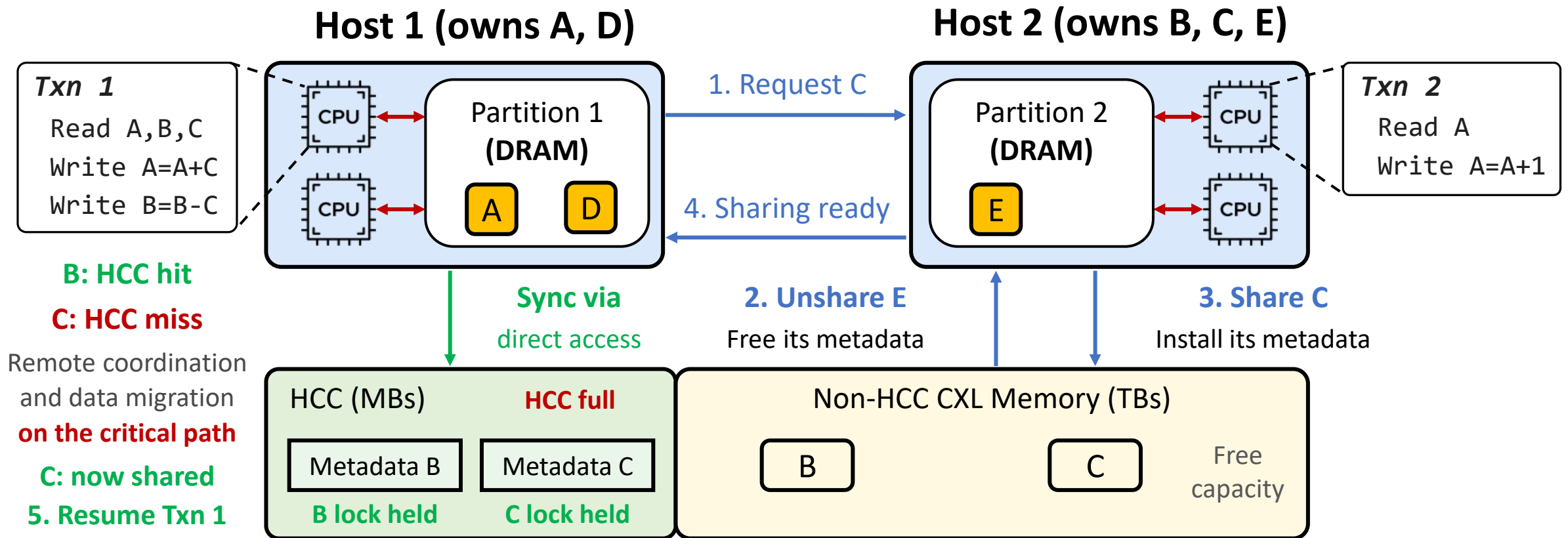
Limited Cross-Host Hardware Cache Coherence (HCC)

- CXL 2.0: no cross-host HCC ; CXL 3.0: limited HCC capacity
 - **Limited HCC** constrains pool-wide transactional synchronization



Prior Work: Data-Partitioned On-Demand Sharing

- **Per-object HCC metadata**¹ for synchronization and software coherence
 - **Limited HCC capacity** bounds how many objects can remain shared



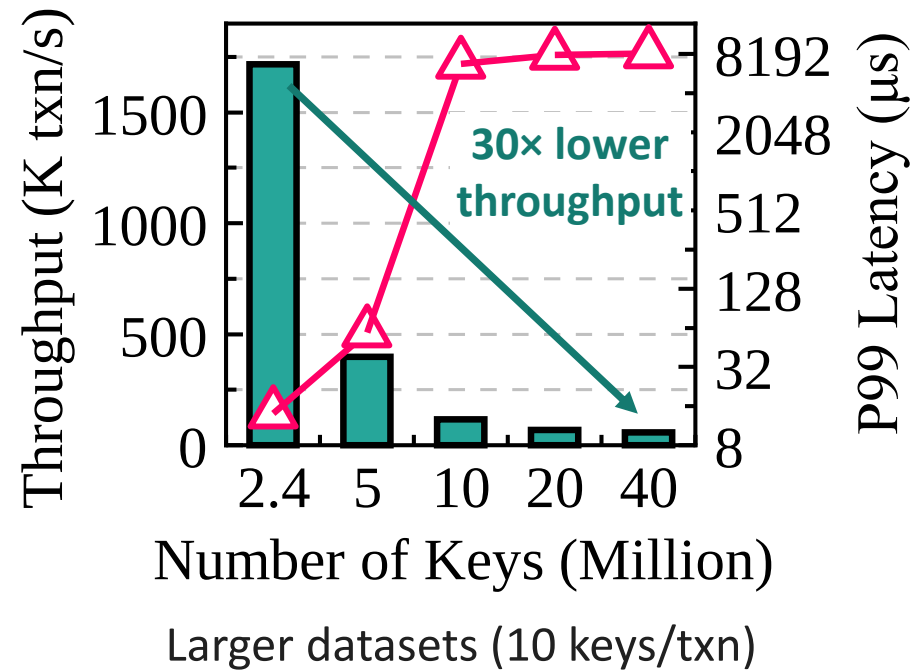
[1] Y. Huang et al., "Tigon: A Distributed Database for a CXL Pod," OSDI '25.

On-Demand Sharing Hits the HCC Capacity Wall

Throughput P99 Latency

Per-object HCC metadata scales with the shared working set

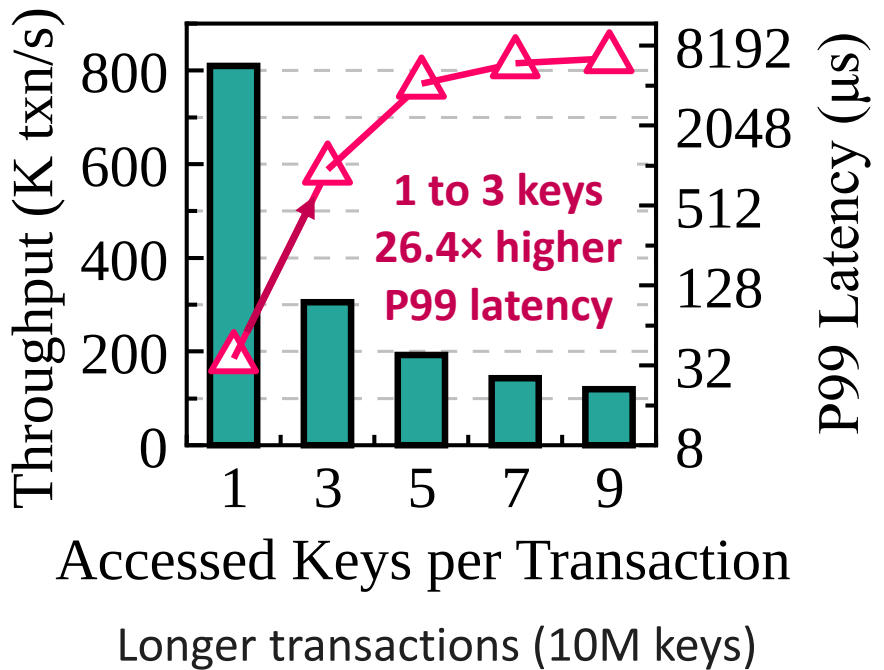
MB-scale HCC limits sharing across TB-scale CXL memory pools



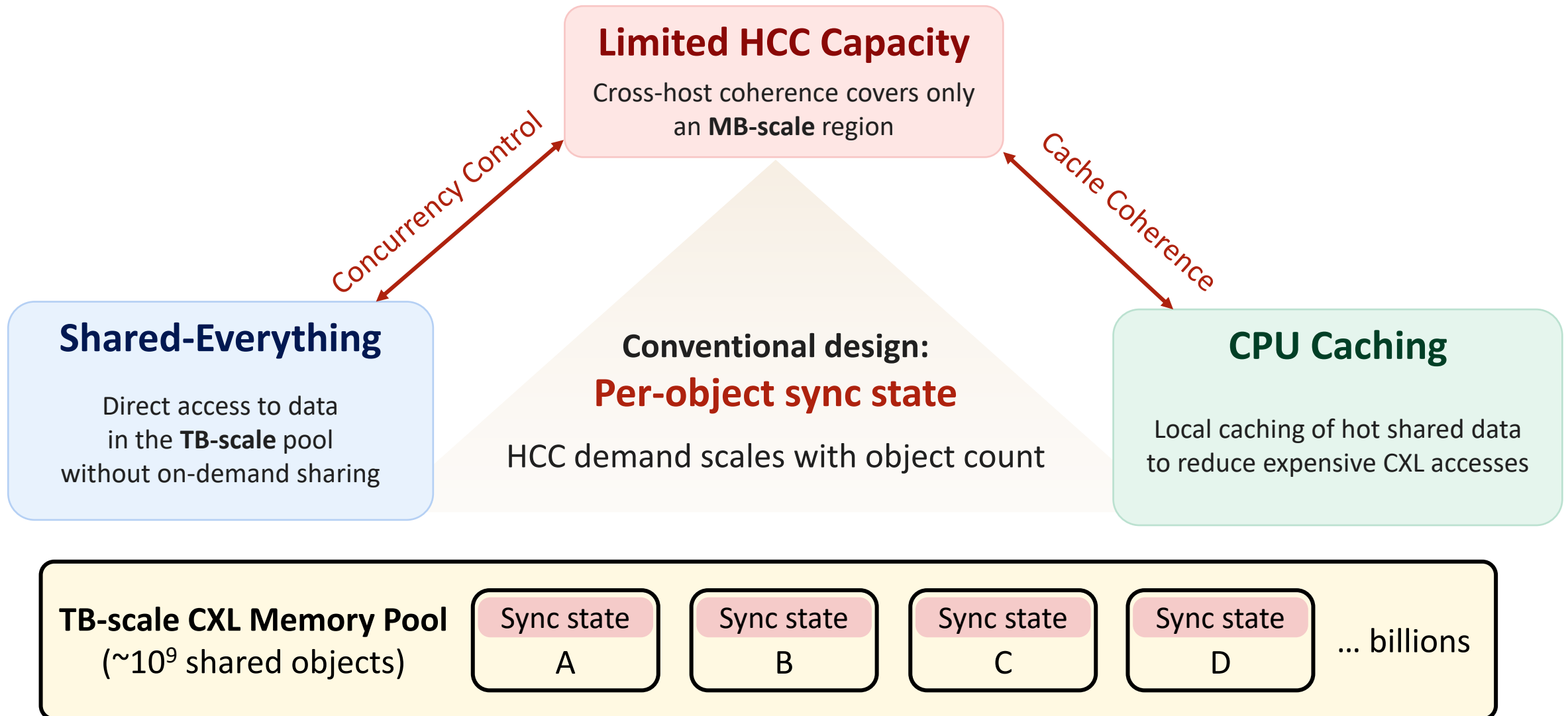
Tigon, KVS
8 hosts
200 MB HCC

Transactional atomicity amplifies on-demand sharing overheads

One HCC miss prolongs lock holding during coordination and migration



The Dilemma of CXL Distributed Transactions



XTRA: Resolving the CXL Distributed Transaction Dilemma

How can we scale **transactional synchronization**
for **concurrency control** and **cache coherence**
beyond **limited HCC capacity**?

Our answer is **XTRA** built on two principles:

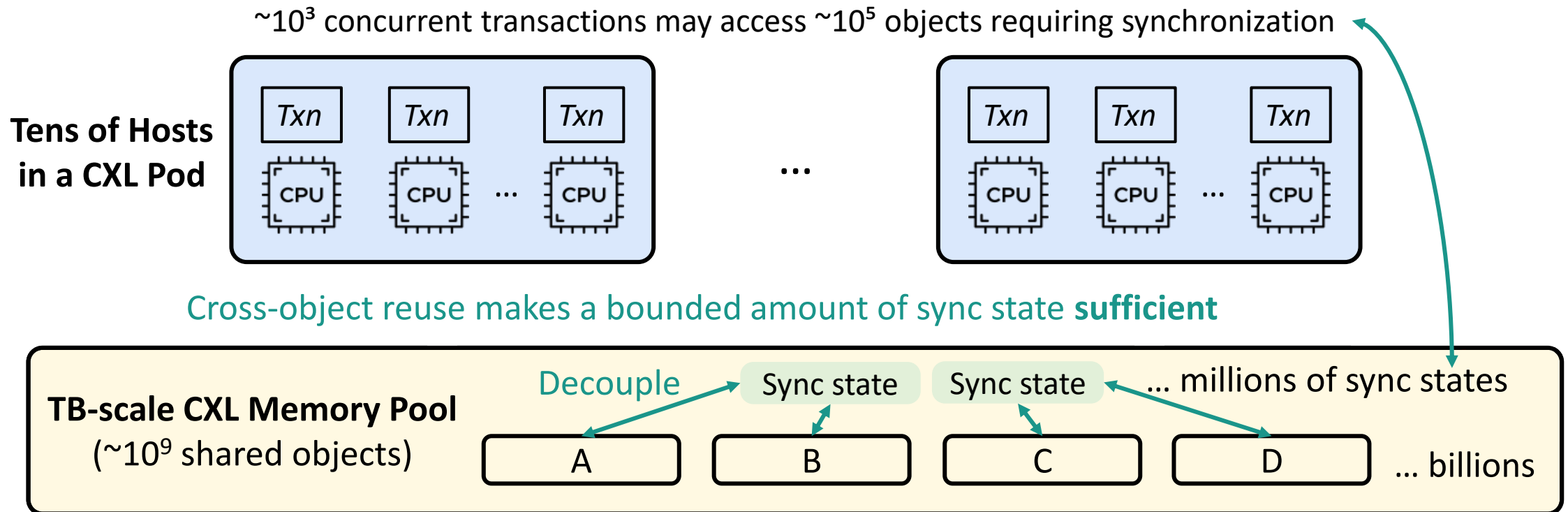
Decouple synchronization state
from individual objects

+

Unify cache coherence with
concurrency control

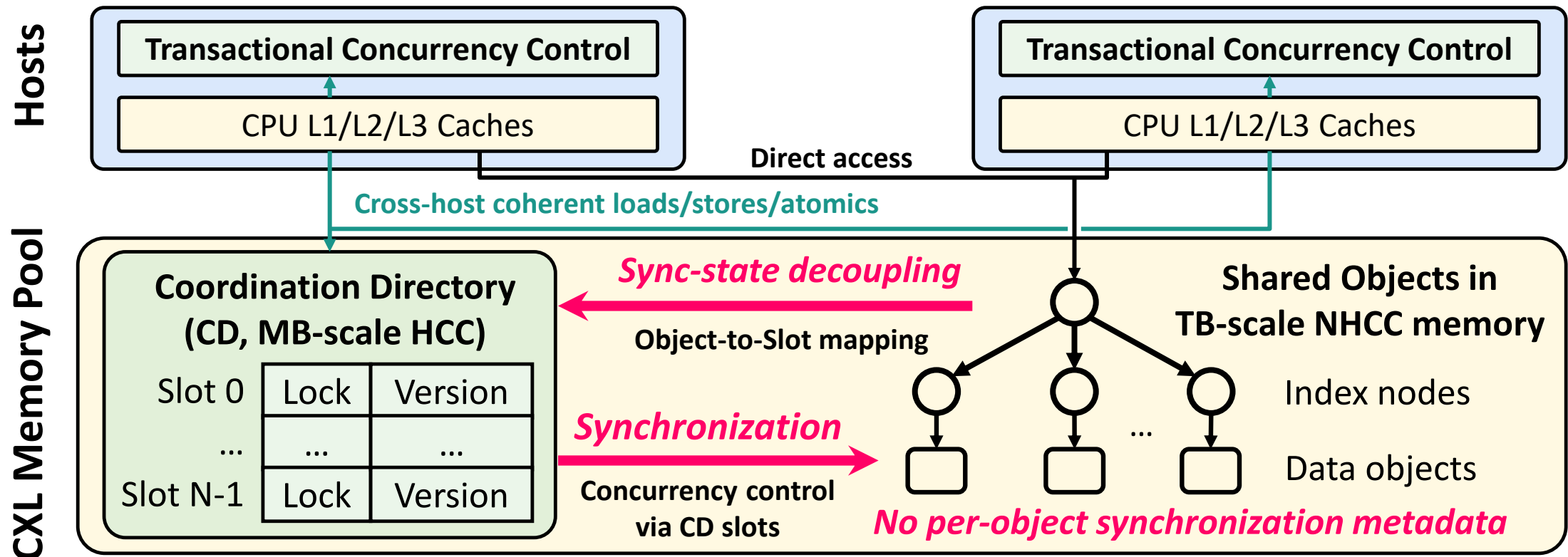
Conventional Per-Object Sync State Is Unnecessary

- Concurrent synchronization involves only a **limited number of objects** at any moment
- **Bounded concurrency-control state** can be reused across objects
- Sync-state capacity scales with **concurrent synchronization**, not shared object count



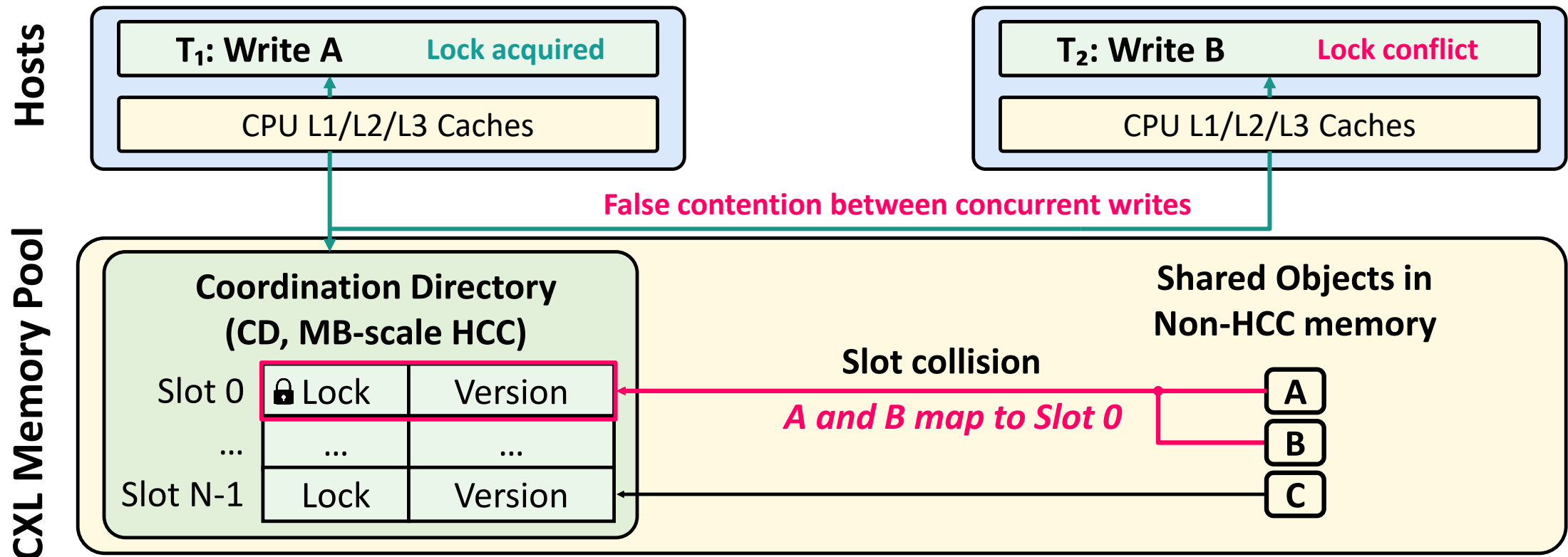
Decoupling Synchronization State from Individual Objects

- Sync state is **decoupled from objects** and consolidated in a **compact CD**
- Each CD slot is shared by objects mapped to it (by **hashing their pool offsets**)
- All hosts coordinate concurrent accesses to the **same object** through the **same CD slot**



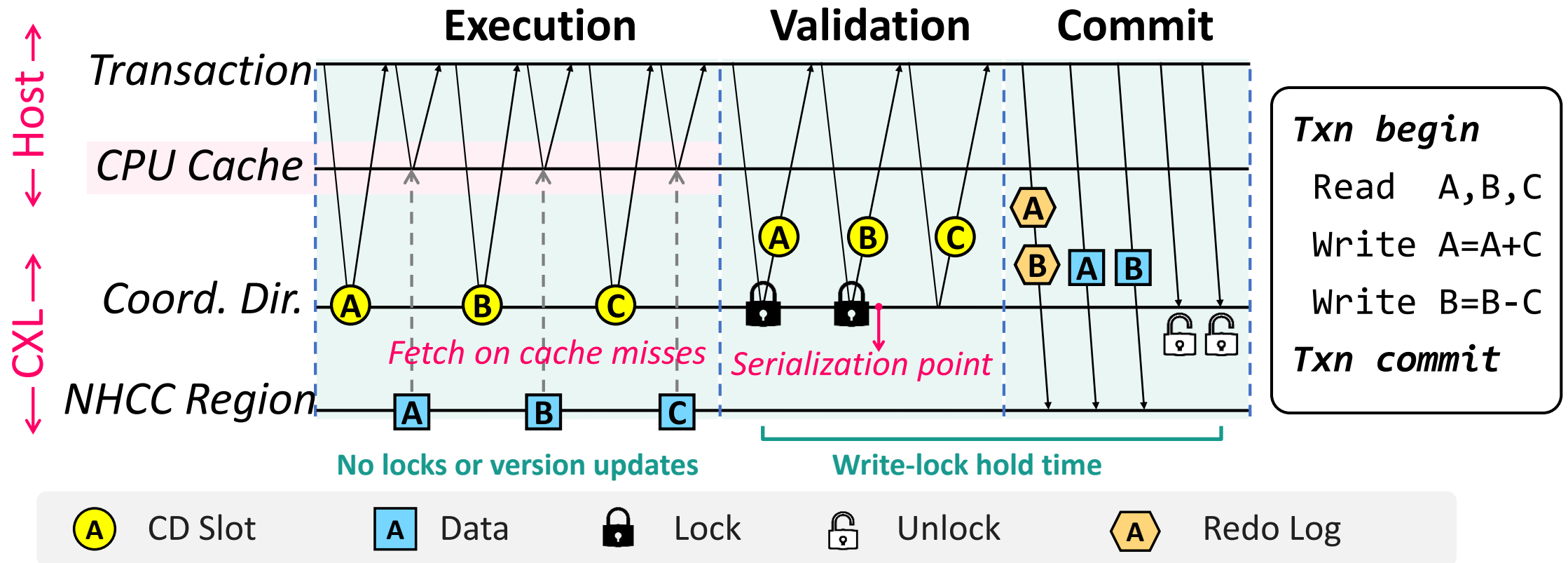
The Trade-off of Decoupled Synchronization

- **Slot collisions:** concurrent accesses to **different objects** may incur **false contention**
- **Correctness is preserved** because no true conflicts are missed
- **Scarce HCC capacity** is multiplexed across the **TB-scale Non-HCC pool**



Decoupled OCC over the Coordination Directory

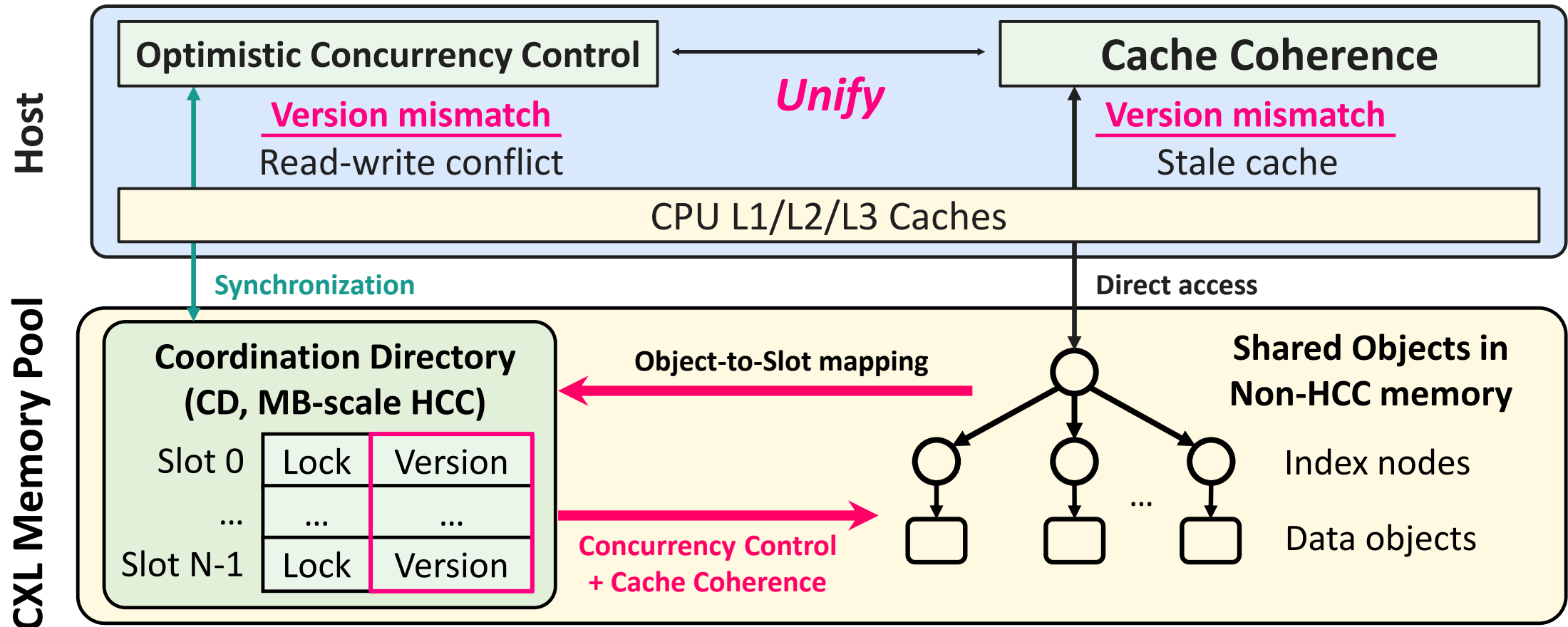
- Reads: **lock-free**, detect read–write conflicts via **CD version checks**
- Writes: **lock** after execution, **advance CD versions** on commit



How do we ensure **cache coherence** when **other hosts update Non-HCC data**?

OCC and Cache Coherence Ask the Same Question

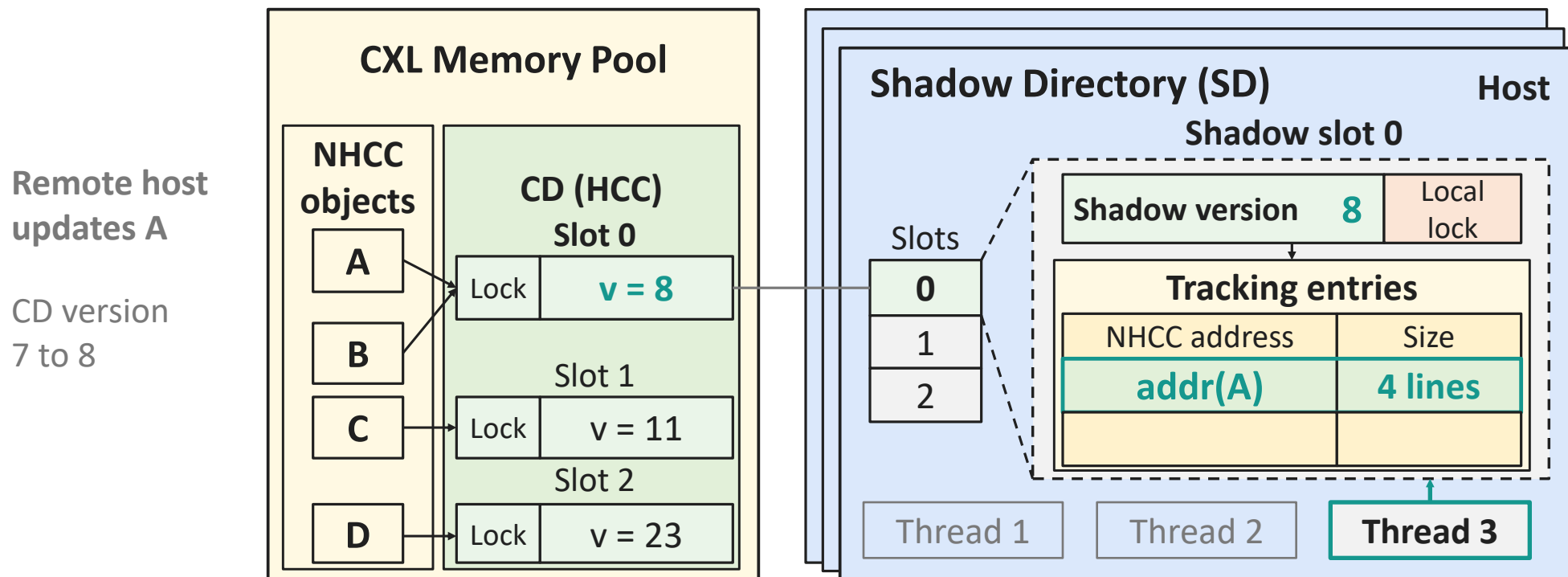
- **OCC:** Has the object been **updated by others** since **this** transaction read it?
- **Cache coherence:** Has the object been **updated by others** since **it was cached locally**?



Unified Cache Coherence via a Shadow Directory

Temporal and Spatial Mismatches between OCC and Cache Coherence

Shadow Directory (SD): coherence state in **host DRAM**, shared by local threads

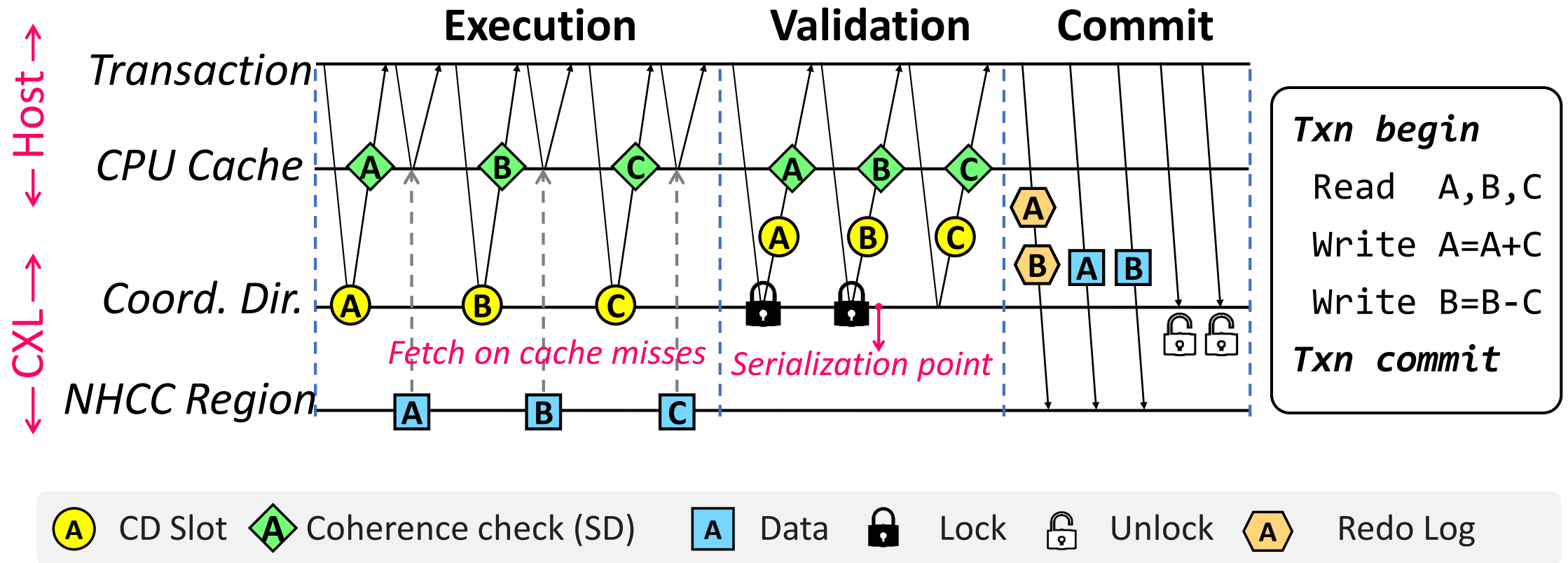


Re-establish coherence lazily on read

Invalidate and track A before reading it.
Defer B's invalidation until B is read.

Unified Cache Coherence on the OCC Read Path

Coherence checks piggyback on **OCC's existing CD-version reads**

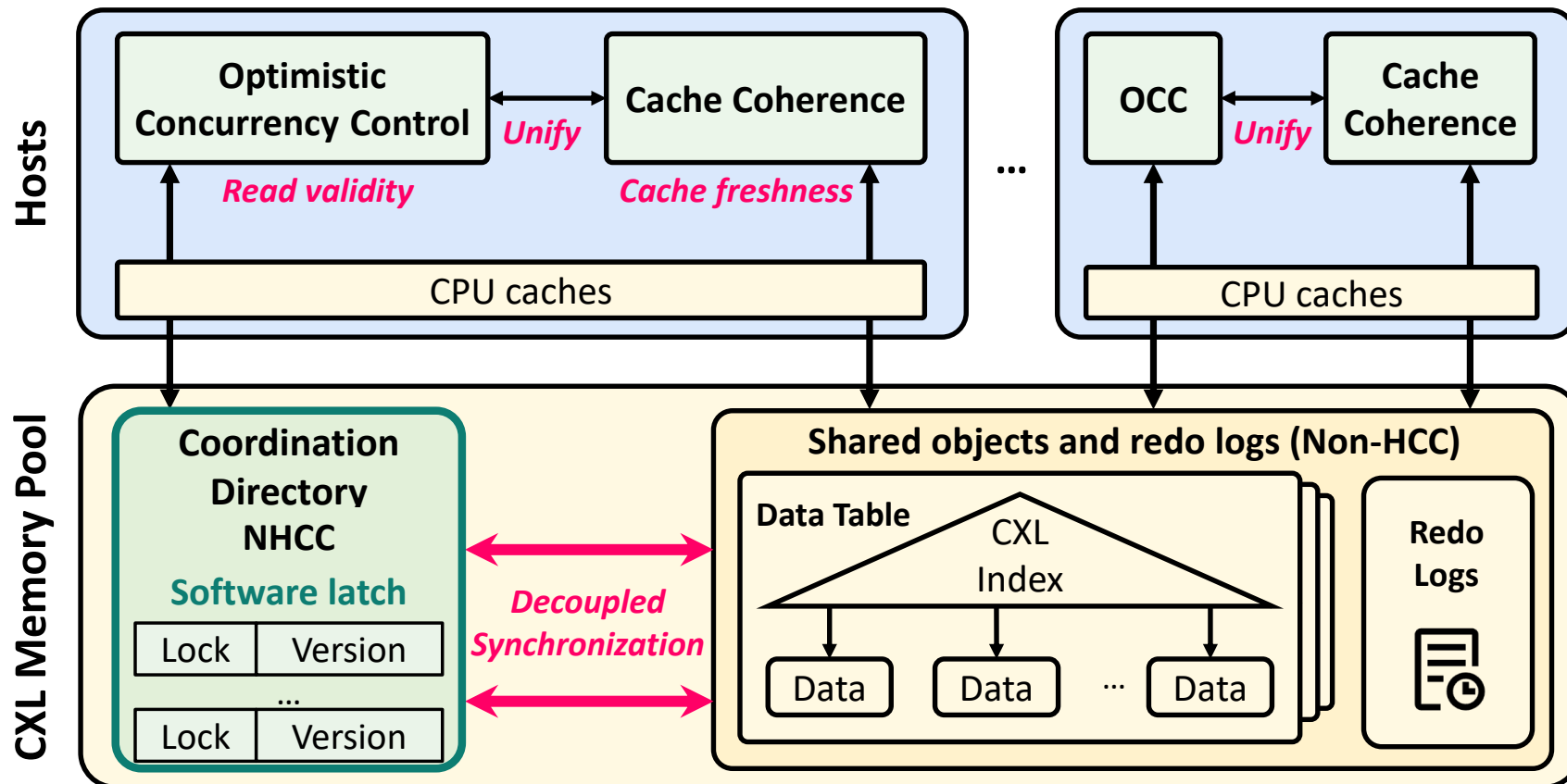


Lazy invalidation on read avoids eager cross-host invalidations on write

Generalizing XTRA to CXL 2.0

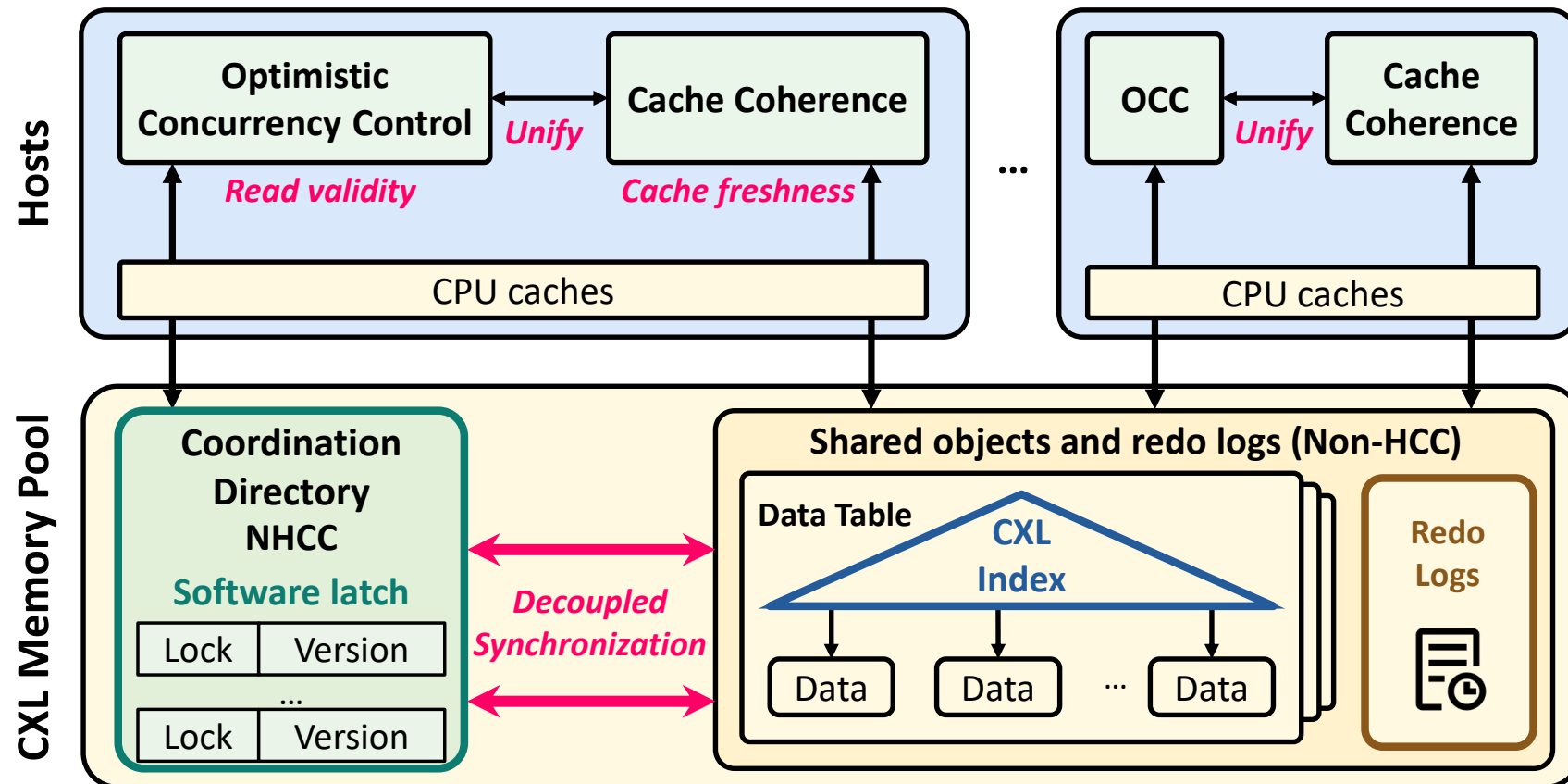
- **CD in Non-HCC:** invalidate cached slots before reads
- **Software latches** (Lamport's fast mutual exclusion) for **atomic CD-lock acquisition**

Decoupled OCC and unified coherence remain unchanged



More Design Details in the Paper

- **Software latches:** compact layout and hierarchical synchronization
- **Index operations:** CD-based optimistic crabbing
- **Fault tolerance:** **redo logging** for recovery from host failures



Evaluation Setup

- **Emulated 8-host CXL pod:** 4 cores + 32 GB DRAM per VM
 - 8 VMs share **one physical 128 GB CXL 2.0 device**. Emulated HCC: 200 MB (default)

CXL baselines¹: on-demand sharing **Our systems: shared-everything**

- **Sundial-CXL:** timestamp leasing
- **Tigon:** 2PL
- **XTRA:** CD in HCC, hardware atomics
- **XTRA-NHCC:** CD in Non-HCC, software latches

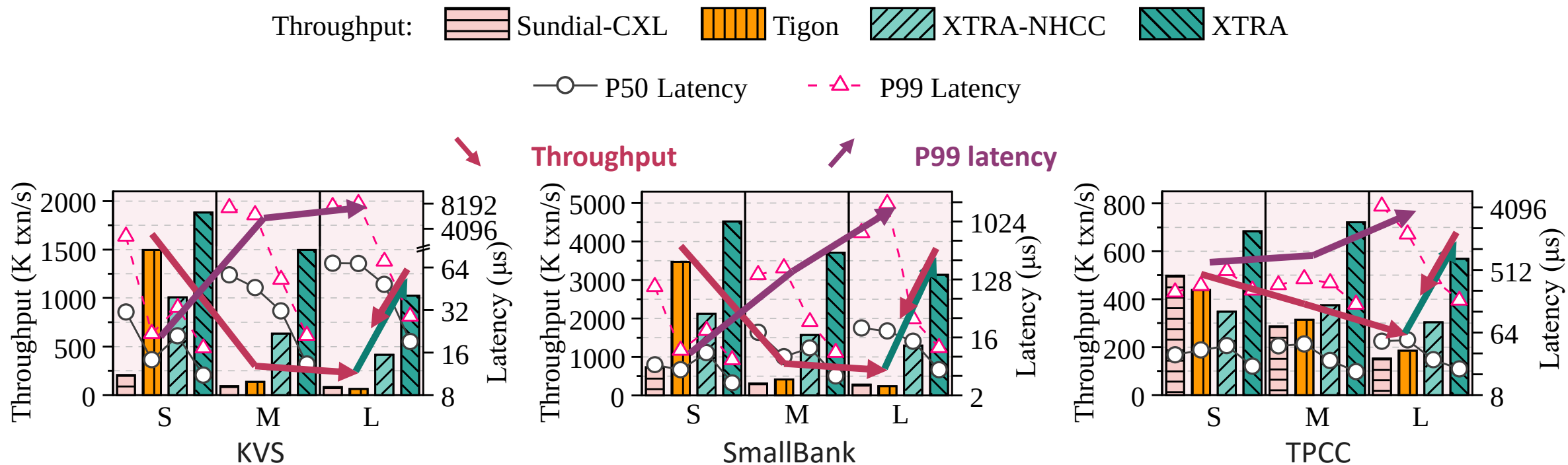
Workloads and **dataset sizes** (total across 8 hosts)

Workload	S (small)	M (medium)	L (large)	Key characteristics
KVS (records)	2.4M	10M	40M	10 keys/txn, 5% write accesses, Zipf $\theta = 0.7$
SmallBank (accounts)	1.2M	5M	20M	≤ 3 records/txn, 85% read-write txns, Zipf $\theta = 0.7$
TPCC (warehouses)	24	240	960	Standard mix, strong warehouse locality

[1] Tigon (OSDI '25). Sundial-CXL uses Tigon's CXL-optimized implementation of Sundial (VLDB '18).

This talk focuses on CXL systems. Our paper also evaluates RDMA-based Motor (OSDI '24).

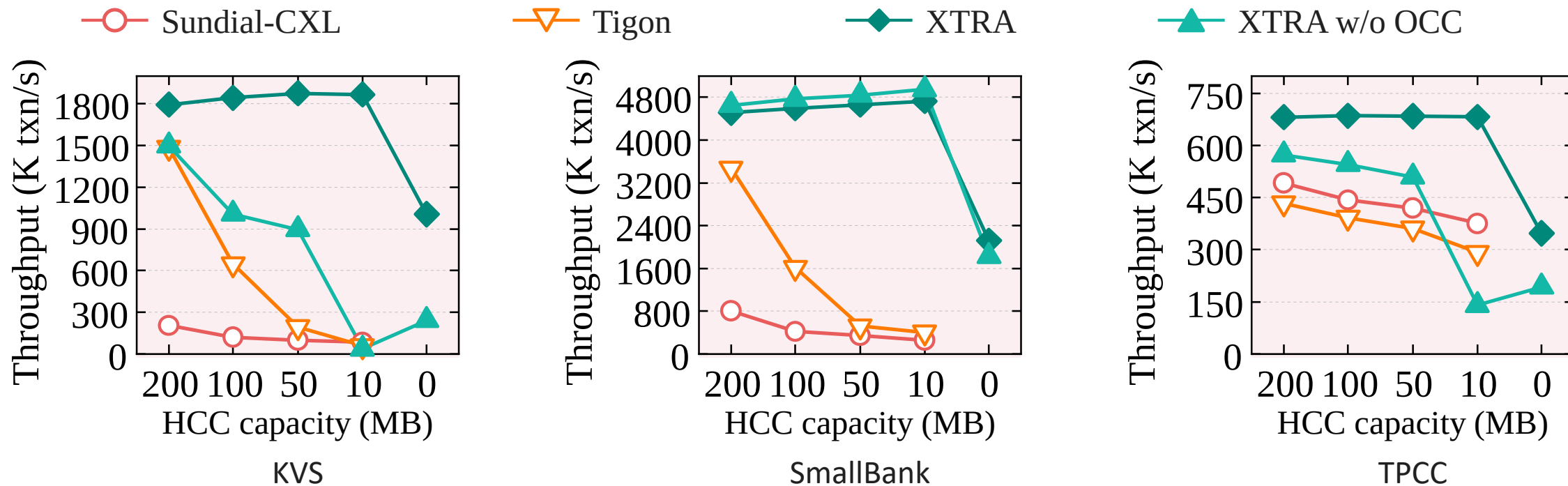
End-to-End Performance across Dataset Scales



At the **large (L) scale**, XTRA-NHCC has **46.6–59.3% lower throughput than XTRA**

Software latches and CD-slot invalidation require more CXL accesses.
At M/L scales, **XTRA-NHCC (no HCC)** outperforms **Tigon (200 MB HCC)**.

Impact of HCC Capacity and Decoupled OCC

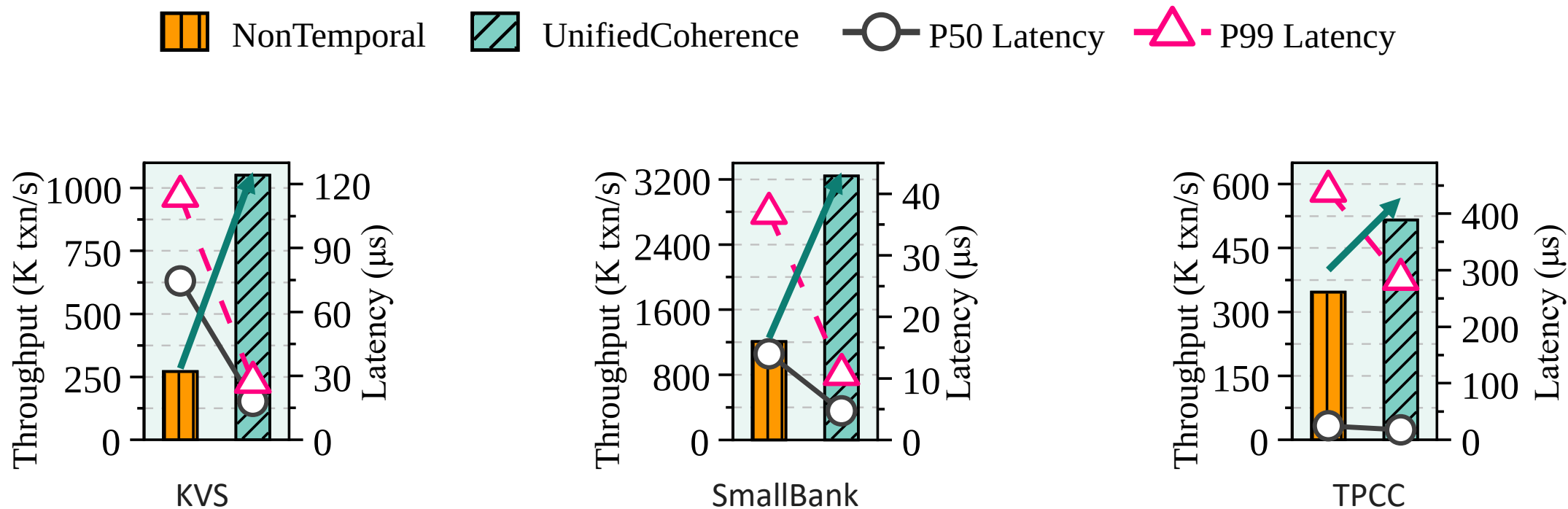


At 0 MB HCC, XTRA and XTRA w/o OCC use a 200 MB CD in Non-HCC memory.

XTRA vs. XTRA w/o OCC (two-phase locking) under limited HCC:
Higher throughput in KVS and TPCC, slightly lower in SmallBank

OCC's **lock-free reads** and **short write-lock hold times** limit false contention from slot collisions.
Write-heavy **SmallBank** favors **2PL** by avoiding OCC validation aborts.

Benefits of Unified Cache Coherence



At the **large (L) scale**, UnifiedCoherence vs. NonTemporal:
1.5–3.9× higher throughput and **1.5–4.2× lower P99 latency**

CD-version checks and the **SD** enable **safe CPU-cache reuse**.
CPU-cache hits avoid **expensive CXL accesses**.

Conclusion

- **CXL pod opportunity:** Redesign distributed systems around shared memory
- **Challenge:** Limited cross-host HCC bottlenecks concurrent and consistent transactions

XTRA: distributed transactions in a CXL pod beyond HCC constraints

Principle 1: Decouple synchronization state from data

- A bounded CD scales concurrency control across the shared pool

Principle 2: Unify cache coherence with concurrency control

- Lazy cache invalidation piggybacks on OCC version checks

XTRA's design principles enable transactions with limited or no HCC and provide a scalable blueprint for other distributed systems in CXL pods.



Source code

github.com/cszjyang/xtra

