Revamping Block-Level I/O Caching for Emerging Tiered Storage

Lin Zhen[1,2], **Lianjie Cao**[1], Faraz Ahmed[1], Hui Lu[2], Puneet Sharma[1]

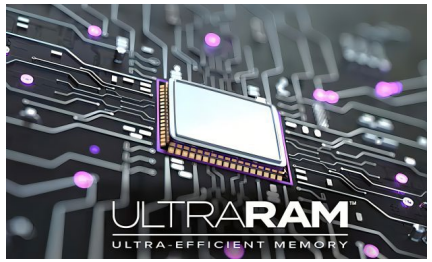
[1] HPE Labs, [2] University of Texas at Arlington

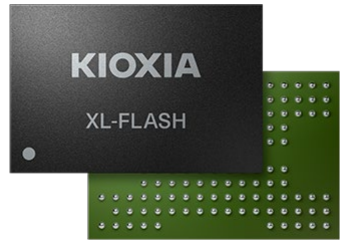
Agenda

- **Introduction**
- **Motivation**
- **EMSCache Design**
- **Evaluation**

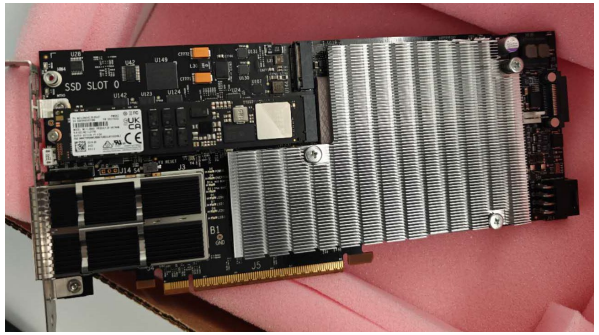
Emerging Storage Class Memory

- Storage Class Memory (SCM) / Non-Volatile Memory (NVM)
  - Byte-addressable memory interface ⇒ finer-granular access and lower I/O overhead
  - High bandwidth and low latency ⇒ 3~5X write bandwidth and latency 100s ns vs. 10s us
  - High endurance ⇒ much more P/E cycles (107) than NAND and suitable for write-intensive workloads





- CXL-SSD
  - Combines both DRAM and NAND flash in a single device ⇒ performance ≈ CXL memory + capacity ≈ SSD
  - Byte-addressable memory interface to NAND flash



Large-Capacity NAND Flash SSDs

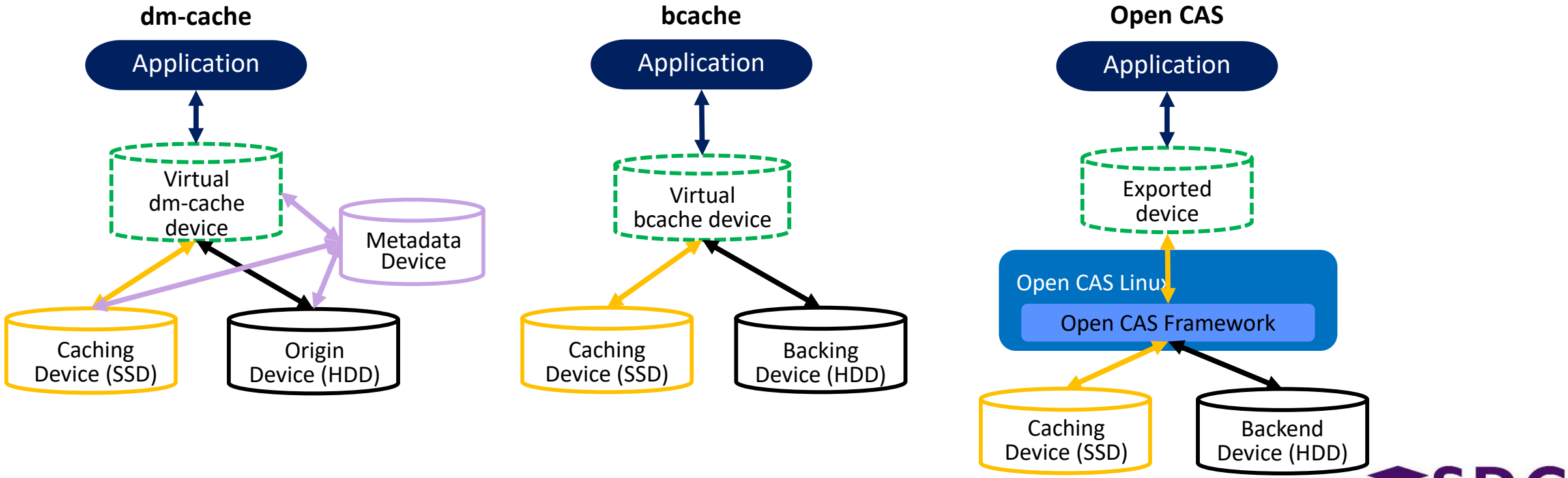
- Large-capacity SSDs (QLC/PLC)
  - SLC ⇒ MLC ⇒ TLC ⇒ QLC: capacity ⇑, performance ⇓, endurance ⇓
  - Lower price: declined ~20X from 2012 (~$200 for 256GB) to 2024 (~$200 for 4TB)
  - Larger capacity: increased ~100X from 2012 (1 TB) to 2024 (100 TB)
  - Shorter lifespan: 105 P/E cycles of SLC to 103 P/E cycles of QLC

	SLC	MLC	TLC	QLC	HDD
Space (TB)	0.6	1.2	6.4	23	22
Read (GB/s)	7.2	7.0	6.8	6.0	0.26
Write (GB/s)	6.1	5.2	4.2	2.5	0.26
IOPS-R (K)	1500	1300	1000	800	0.24
IOPS-W (K)	1350	800	150	5.7	0.46
Endurance (%)	2000	1500	333	100	-
Cost (%)	400	200	133	100	-

Zhou, Yanbo, et al. "CSAL: the Next-Gen Local Disks for the Cloud." Proceedings of the 19th European Conference on Computer Systems (EuroSys). 2024.

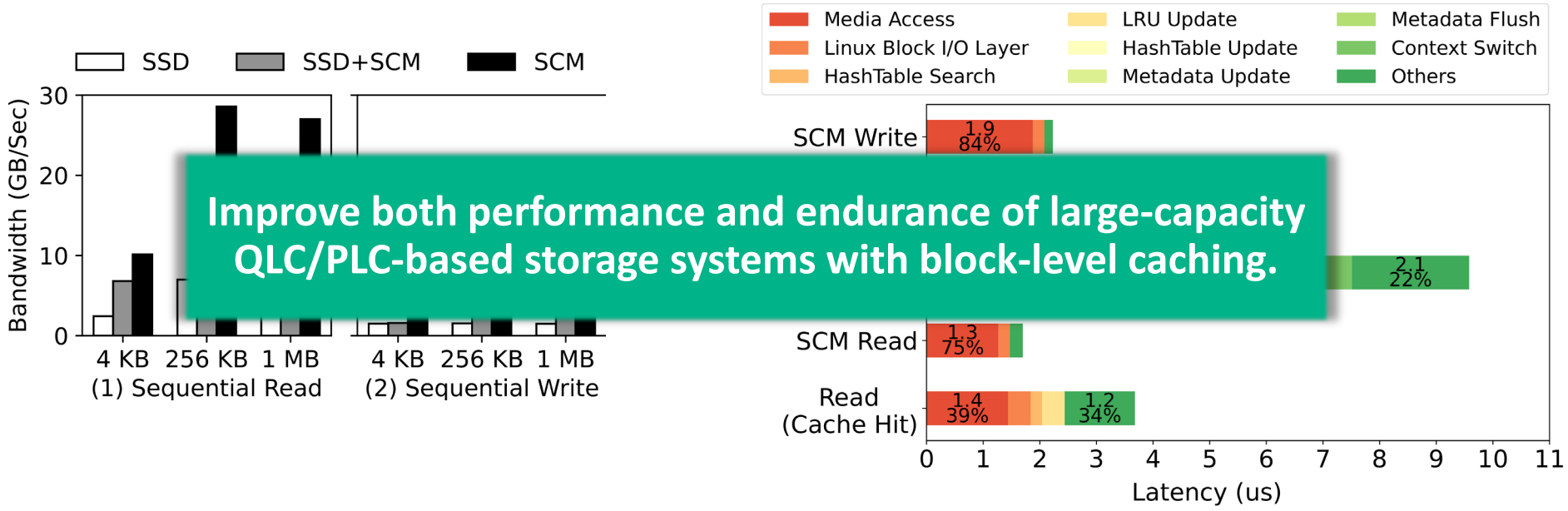
Bridging The Gap

- Block-level caching
  - Bridging the performance and capacity gap between SSD (performance tier) + HDD (capacity tier)
  - Well-established existing systems, e.g., bcache, dm-cache, Open CAS
  - Provide a generic low-level software-based solution, not tied to specific use cases and applications



Drop-in Replacement?

- Existing block-level caching systems
  - Don't leverage characteristics of SCM
  - Complex and heavy data path
  - Significant write amplification for metadata update
  - Inefficient data eviction on critical path that exacerbates write amplification



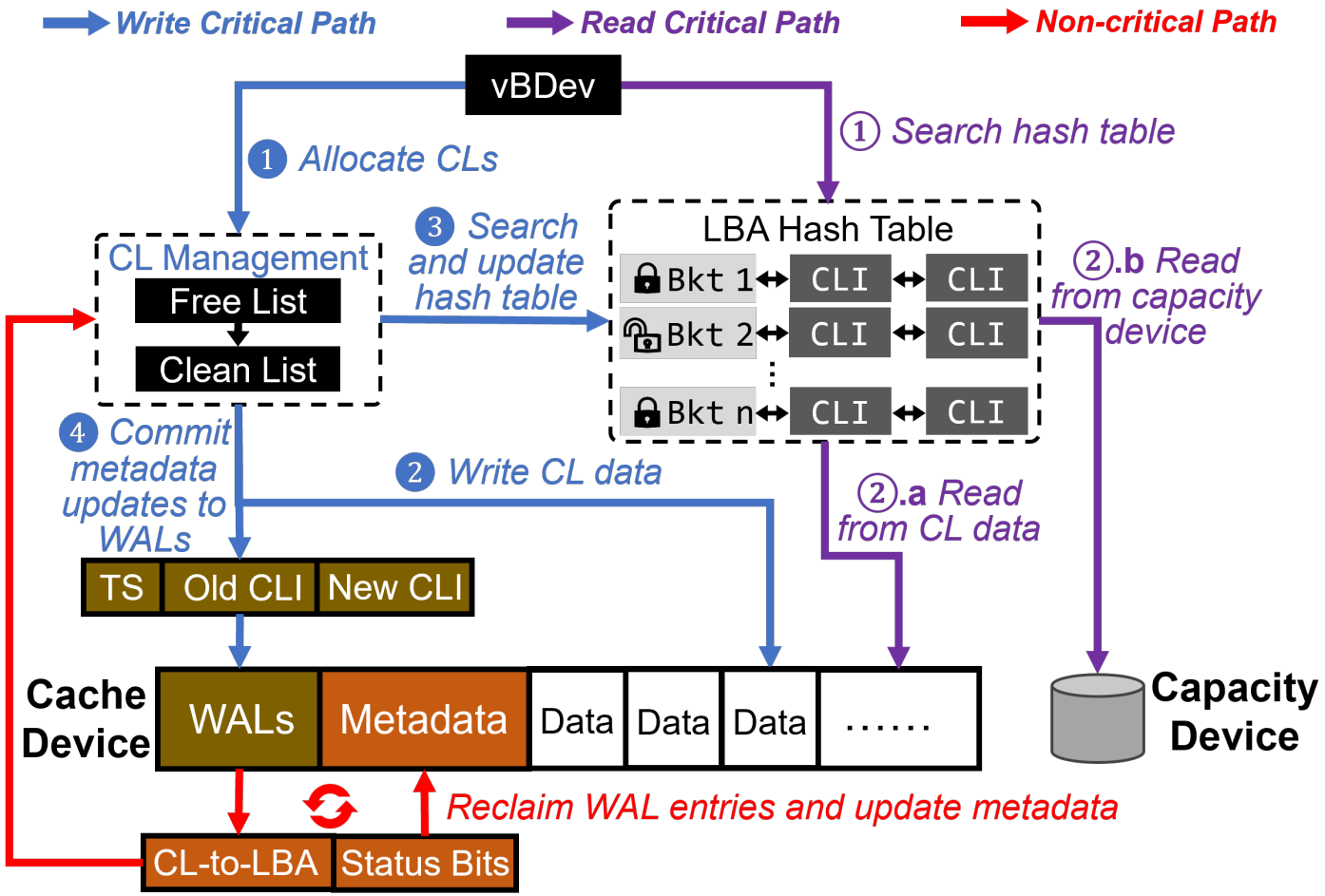
Improve both performance and endurance of large-capacity QLC/PLC-based storage systems with block-level caching.

EMSCache Design

- Streamlined, high-performance caching data plane
  - Decouples metadata updates from data writes via log-structured indirection
  - Coalesces small metadata operations to avoid block-level write amplification
  - Leverages byte-addressability in SCM for fast, failure-atomic persistence

- Endurance-aware eviction strategy
  - Align eviction writes with the SSD's internal write unit size
  - Reduce read-modify-write overheads.

Data Plane



Lightweight Logging

Persist metadata at coarse granularity (e.g., 4 KB) ⇒ Large write amplification and latency

- Log-based metadata updates
  - A compact, fixed-size metadata region
  - Per-core write-ahead log (WAL) region
- Multi-core parallelism and log reclamation
  - WAL entries are appended per core without coordination
  - Periodically scans all WALs and replays committed log entries in timestamp order

Algorithm 1 Atomic Persistence of Data Updates

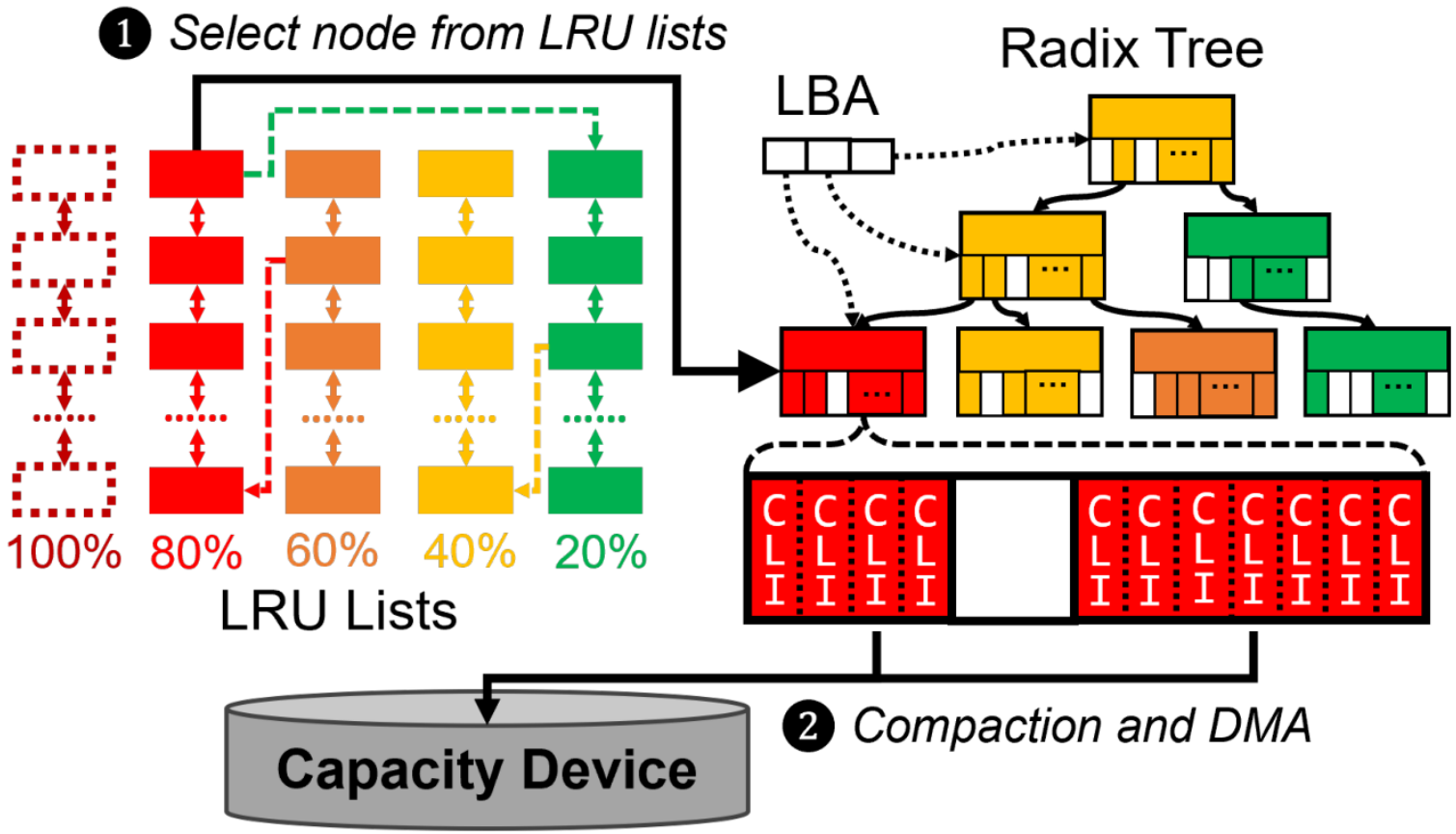
```
1: function PERSISTDATA(bio)
2:     newCacheLine ← ALLOCATECACHELINE( )   ▷ ❶ Obtain a free cache line from the free list
3:     logEntry ← CREATELOGENTRY(bio, newCacheLine)
4:     PERSISTTOSCM(logEntry)   ▷ ❷ Persist data to the SCM cache
5:     UPDATEDRAMHASHTABLE(logEntry)   ▷ ❸ Update volatile state
6:     SFENCE( )   ▷ Ensure write ordering
7:     APPENDLOG(logEntry)   ▷ ❹ Append update to persistent log
8:     UPDATELOGTAIL( )   ▷ Advance log tail pointer
9:     SFENCE( )   ▷ Ensure write ordering
10: end function
```


Strong Crash Consistency

Asynchronous metadata persistence ⇒ better performance, but ambiguous recovery

- Atomic update protocol using byte-addressable SCM
  - Persists both data and metadata atomically and in order
  - Guarantees data is fully persisted before metadata is committed
- Fast and bounded crash recovery
  - Scan per-core WAL to identify and replay committed but not merged log entries
  - Recovery overhead is bounded and no need for full-cache reconciliation

Endurance-Aware Cache Eviction



Radix Tree-based Cache Line Management

Eviction generates small, random writes ⇒ Reduce lifespan of QLC SSDs

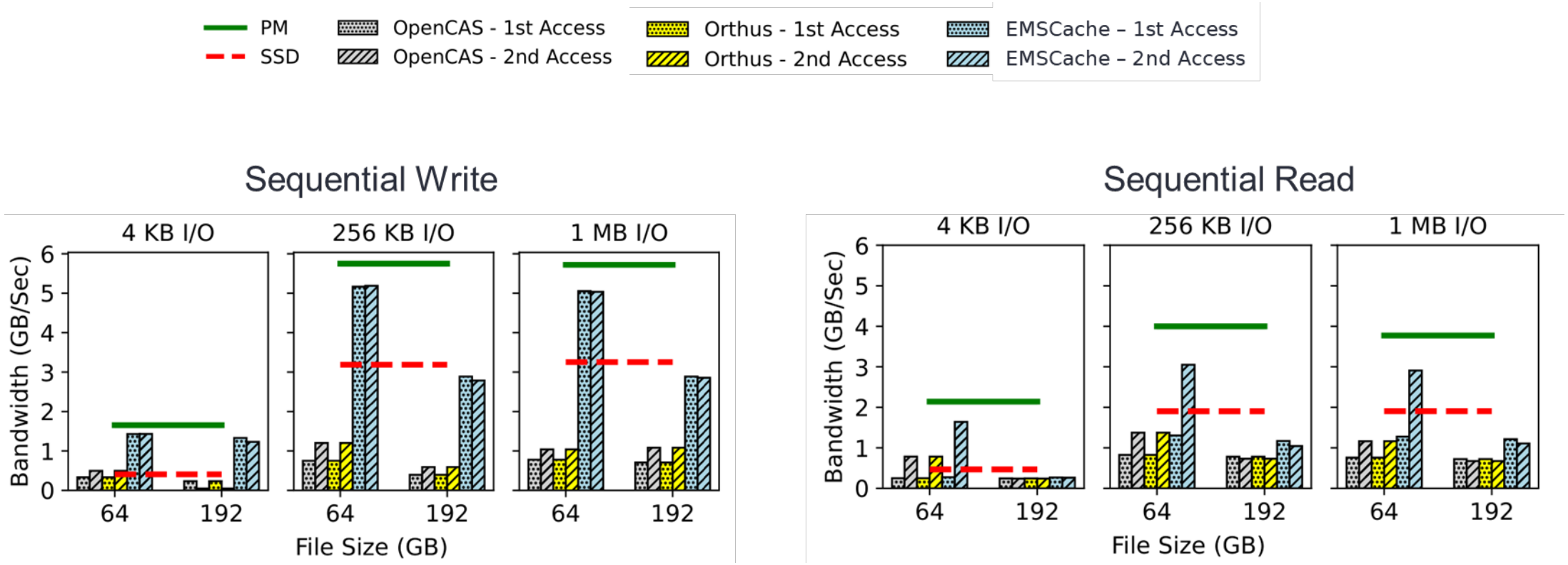
- Radix Tree-based Cache-line Management
  - Efficiently manages cache lines with locality information on capacity device
  - Select cache lines for eviction with IU-aligned leaf nodes
  - Maintain multiple LRU lists based on leaf node occupancy
  - Select the least recently used leaf node in the most occupied list
- Stochastic Cache Eviction
  - Trigger eviction based on cache device utilization state
  - Compact contiguous cache lines and evict to the capacity device

Evaluation

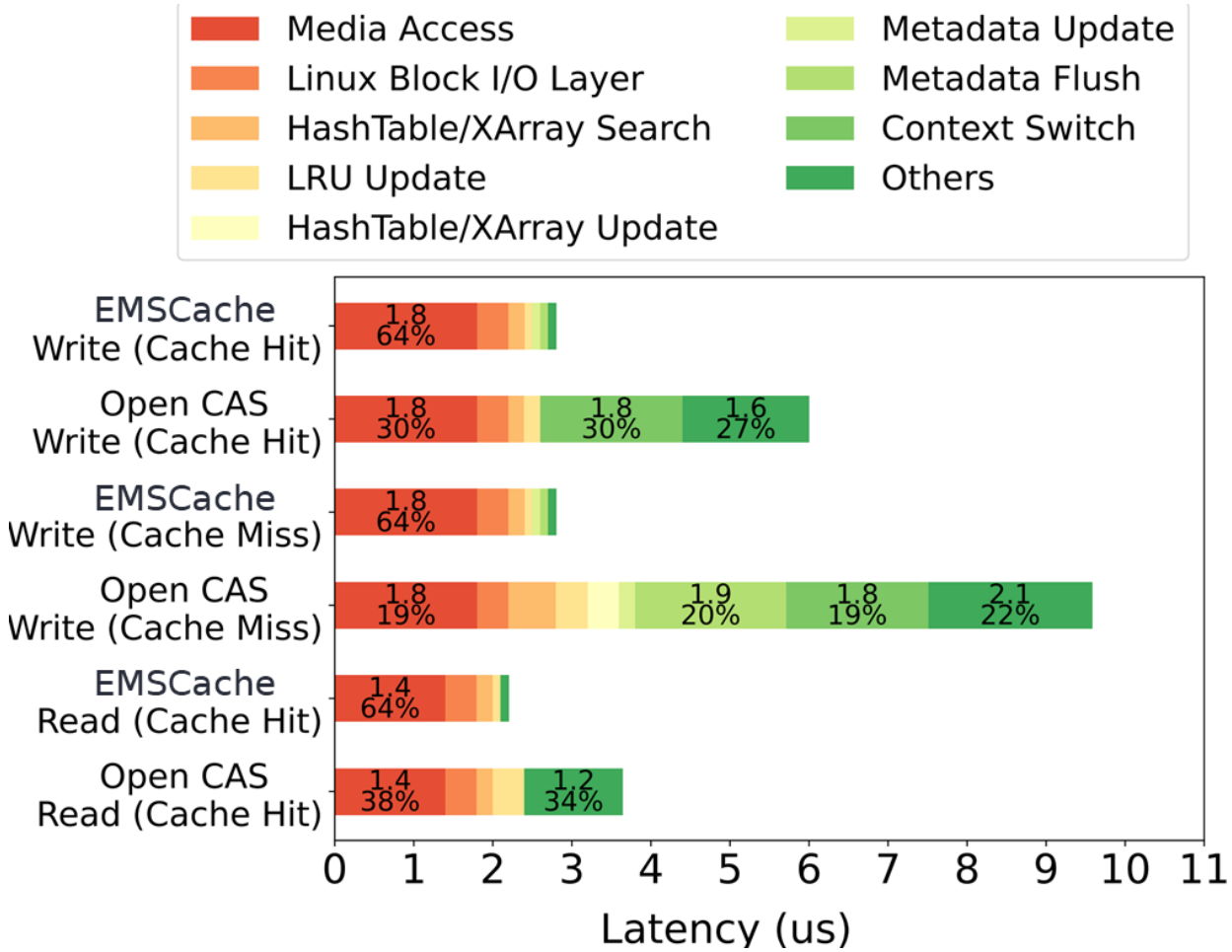
	Platform A	Platform B	Platform C
CPUs	Intel Xeon 4314 (2.4GHz)	Intel Xeon 8536Y (2.4GHz)	Intel Xeon 8536Y (2.4GHz)
DRAM	8 × 128 GB DDR4 (2.4GT/s)	8 × 32 GB DDR5 (4.8GT/s)	8 × 32 GB DDR5 (4.8GT/s)
Cache Device	8 × 128 GB Intel Optane Persistent Memory	1 × 256 GB Micron CZ120 CXL Memory	1 × 1 TB Samsung CMM-H
Capacity Device	1 × 15.68 TB Intel P5316	1 × 15.68 TB Intel P5316	1 × 15.68 TB Intel P5316

- Comparison
  - Cache device only
  - Capacity device only
  - Orthus
  - Open CAS

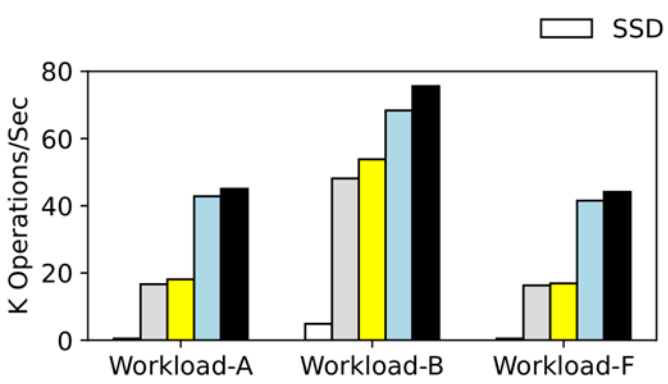
Microbenchmarks – Single Thread



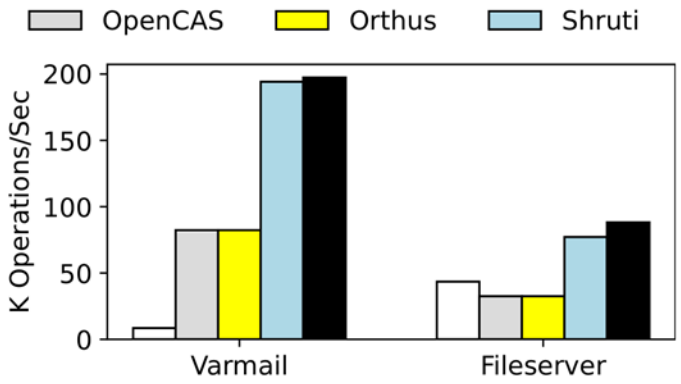
Latency Analysis



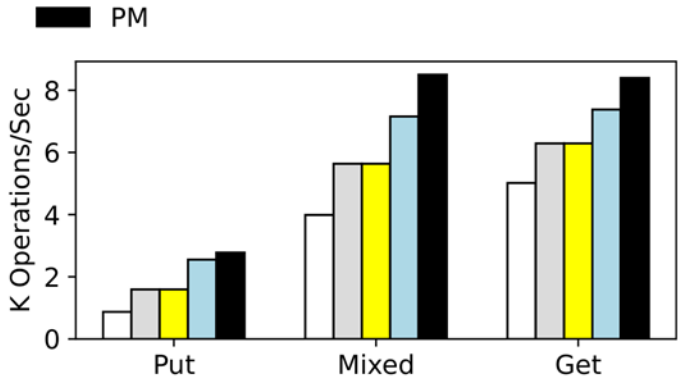
Application Performance



(a) YCSB - RocksDB.



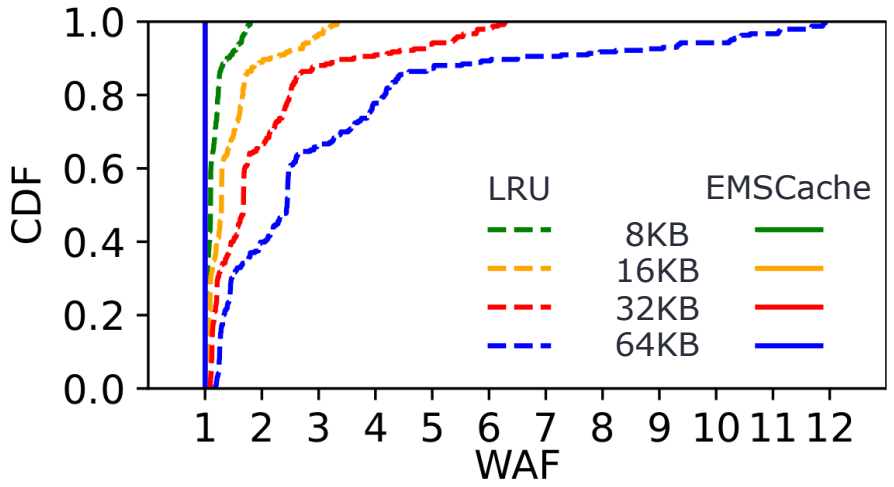
(b) Filebench.



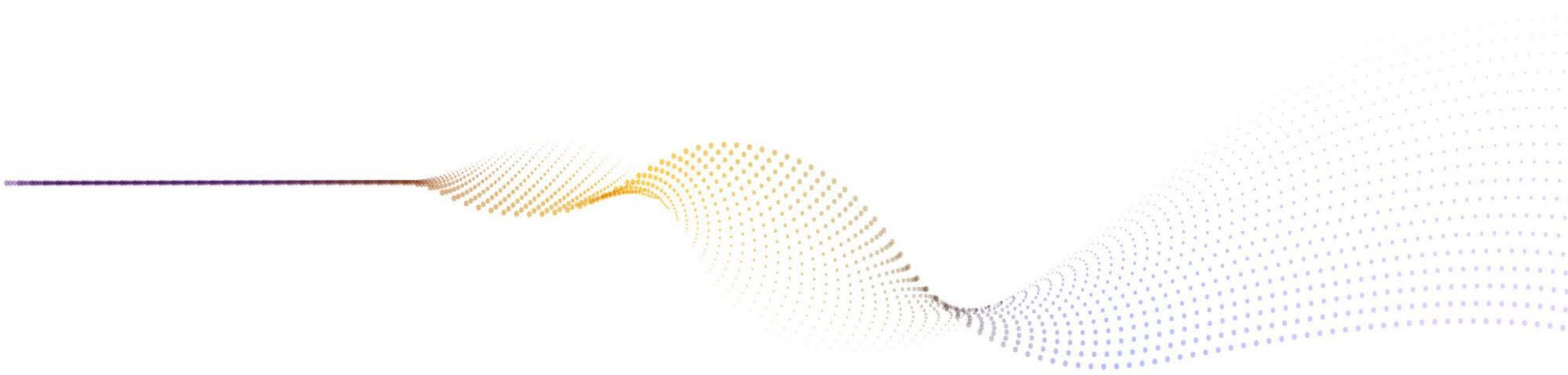
(c) MinIO.

Write Amplification Analysis

- Write amplification factor (WAF) = $\frac{Bytes\ To\ NAND}{Bytes\ From\ Host}$
- Tested 243/1000 real-world I/O traces in Alibaba Block Traces*
- Minimal impacts on cache hit ratio (<1%)



* Alibaba Block Traces, https://github.com/alibaba/block-traces



Thank you for attending!

Please remember to rate this session. You get access the presentations at
http://sniadeveloper.org/conference

