FlatStore: An Efficient Log-Structured Key-Value Storage Engine for Persistent Memory

Youmin Chen, Youyou Lu, Fan Yang, Qing Wang, Yang Wang*, Jiwu Shu

Tsinghua University
*The Ohio State University

http://storage.cs.tsinghua.edu.cn

PM-aware Systems in the past decade …

BPFS [SOSP'09]

Mnemosyne [ASPLOS'11]

SCMFS[SC'11]

CDDS-Tree [FAST'11]

NV-Heaps [ASPLOS'11]

PMFS [Eurosys'14]

Heapo [Eurosys'16]

Aerie [Eurosys'14]

wB+Tree [VLDB'15]

HiNFS [Eurosys'16]

NV-Tree [FAST'15]

LSNVMM [ATC'17]

NOVA [FAST'16]

FPTree [SIGMOD'16]

Hotpot [SoCC'17]

NOVA-Fortis [SOSP'17]

FAST&FAIR [FAST'18]

DudeTM [ASPLOS'17]

RECIPE [SOSP'19]

Strata [SOSP'17]

Level-Hashing [OSDI'18]

Wisper [ASPLOS'17]

ZoFS [SOSP'19]

Pisces [ATC'19]

SplitFS [SOSP'19]

Octopus [ATC'17]

CCEH [FAST'19]

Before 2019: The Emulation Era

Hardware Emulation Assumptions

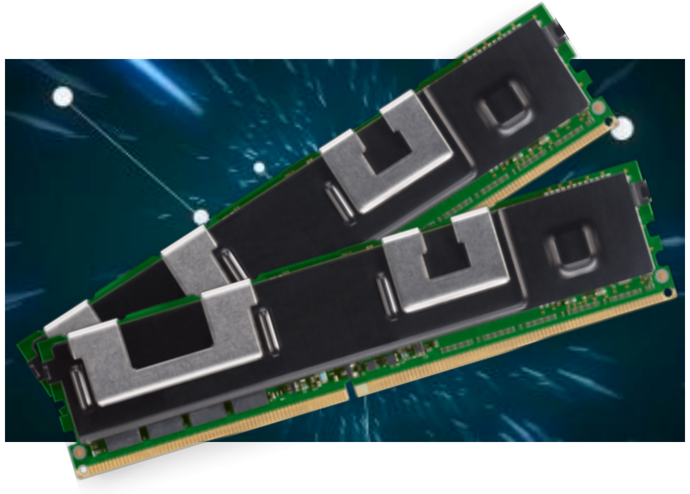
Assumptions:

Byte-addressability

Close-to-DRAM Bandwidth

High Write Latency

Low Read Latency



Cacheline & XPLine

64 bytes / 256 bytes

Slow Write Bandwidth

2.2 GB/s per DIMM
(1/3-1/6 of DRAM)

Comparable Write Latency

~100 ns

High Read Latency

Rnd: 300 ns (3.7x of DRAM)

Hardware Emulation Assumptions

Byte-addressability

Byte-addre

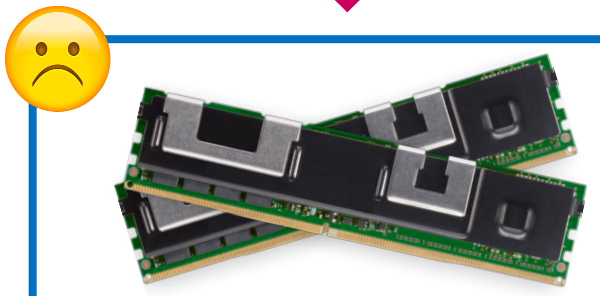
Fine-grained Journaling
PMFS [Eurosys'16], NOVA [FAST'16], etc.

Fine-grained Caching
HiNFS [Eurosys'16], Tinca [SC'17], etc.

Fine-grained Index Structures
Level-Hashing [OSDI'18], etc.

8-byte Atomic Operations
FPTree [SIGMOD'16], FAST&FAIR [FAST'18]

Generate a large number of **synchronized** & **small-sized** I/Os.

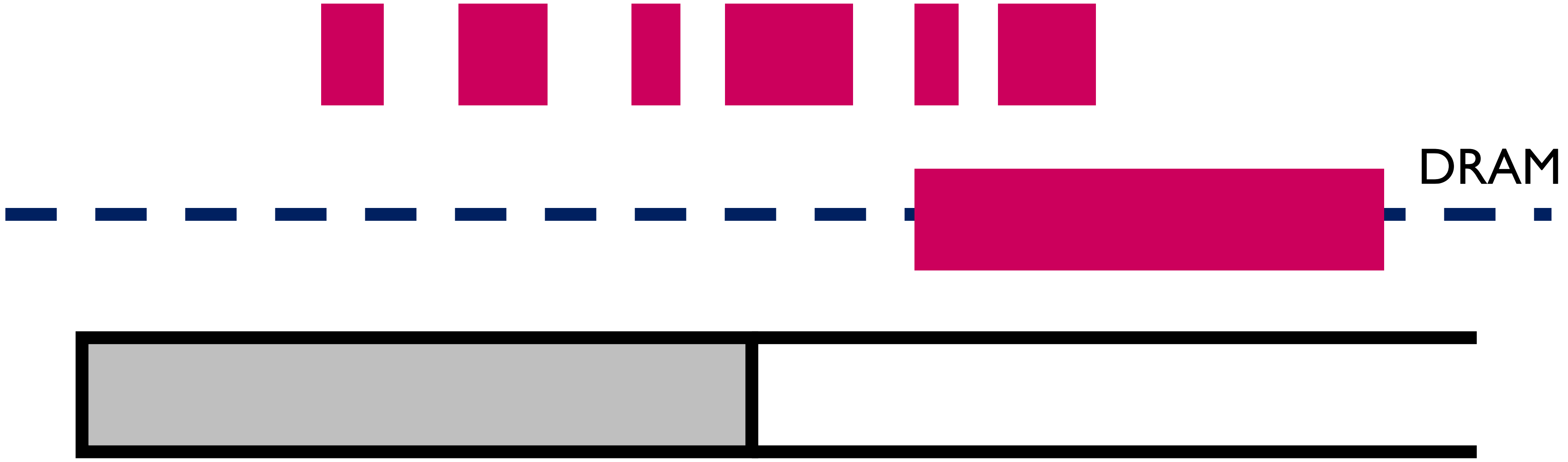


XPLine: 256 bytes

1/8 DRAM Bandwidth

Using a log structure: An intuitive approach

Buffer, then commit



Using a log structure: An intuitive approach

The idea of log structure is very successful for SSD/HDD

- SSD/HDDs prefer sequential access pattern
- The overhead of multiple storage accesses can be amortized via batching
  - Buffer up to tens of MBs of data before persist them

Q: Can a log structure still retain its benefits with Optane DCPMMs?

- Optane shows very close performance for random/sequential accesses
- 256-byte I/O units are enough to saturate the Optane bandwidth
  - It's not beneficial to batch data larger than this I/O size
- Log cleaning overhead

FlatStore: An Efficient Log-Structured Key-Value Storage Engine

Simple insight: Selective batch to maximize the potential performance.

- Small updates are appended to the per-core log structure
- Large updates are stored separately via a persistent allocator

Techniques:

- Compacted Log Format: Improve the batching opportunity
- Pipelined Horizontal batching: Without increasing the latency

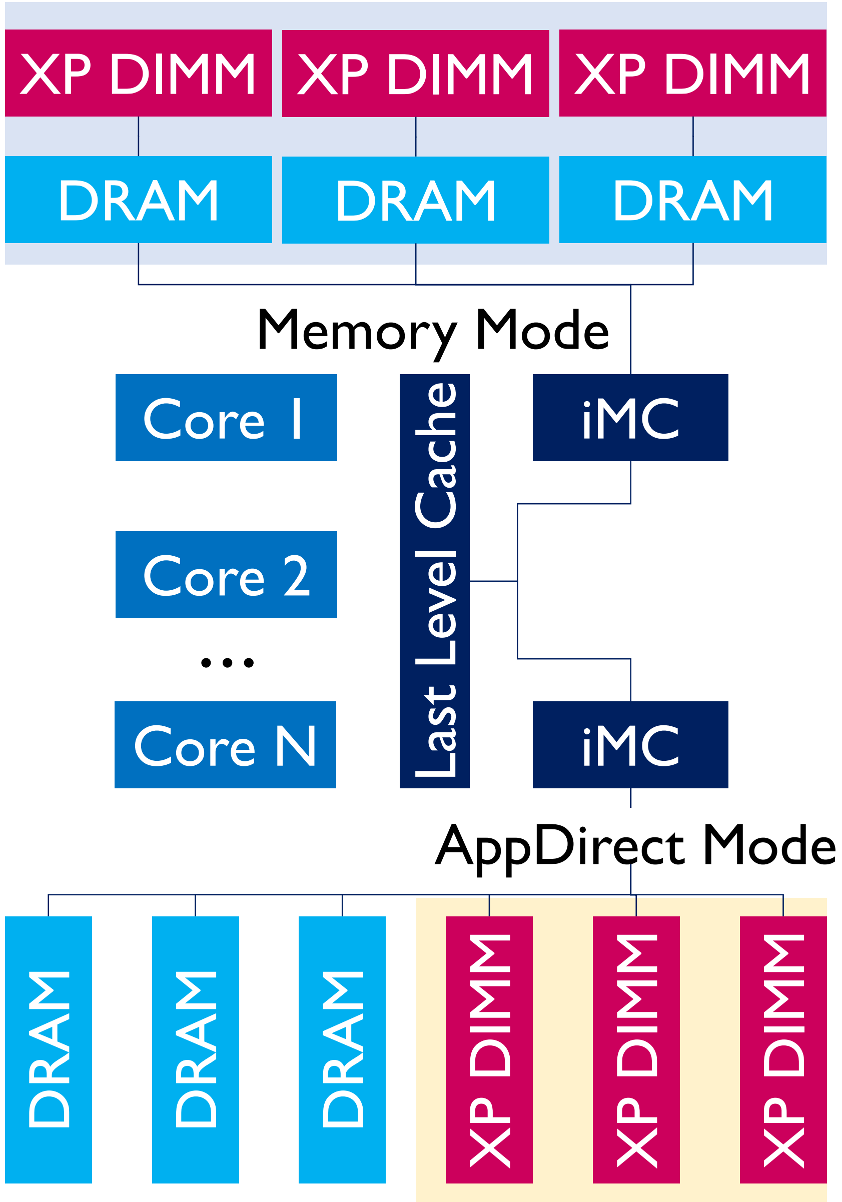
Results:

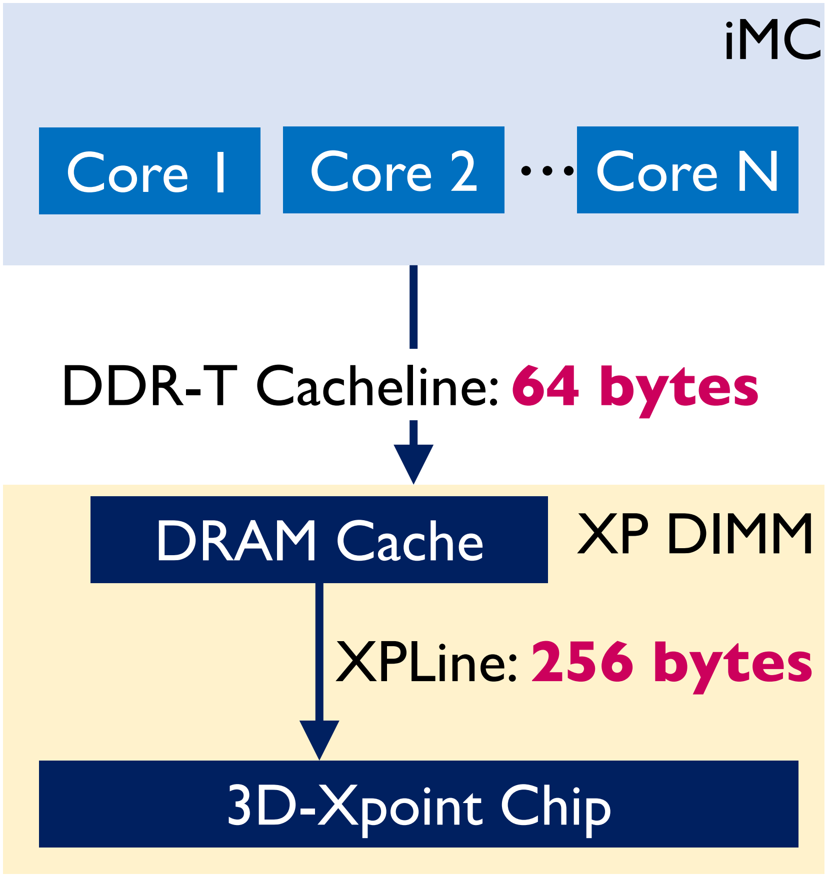
- Support both hash- and tree-based index structures
- Achieves up to **35 Mops/s** with a single server node
- **2.5 – 6.3 times** faster than existing systems

Outline

- Introduction
- Optane DC Persistent Memory Module
- FlatStore: An Efficient Log-structured Storage Engine
- Results
- Summary & Conclusion

Optane DC Persistent Memory Module



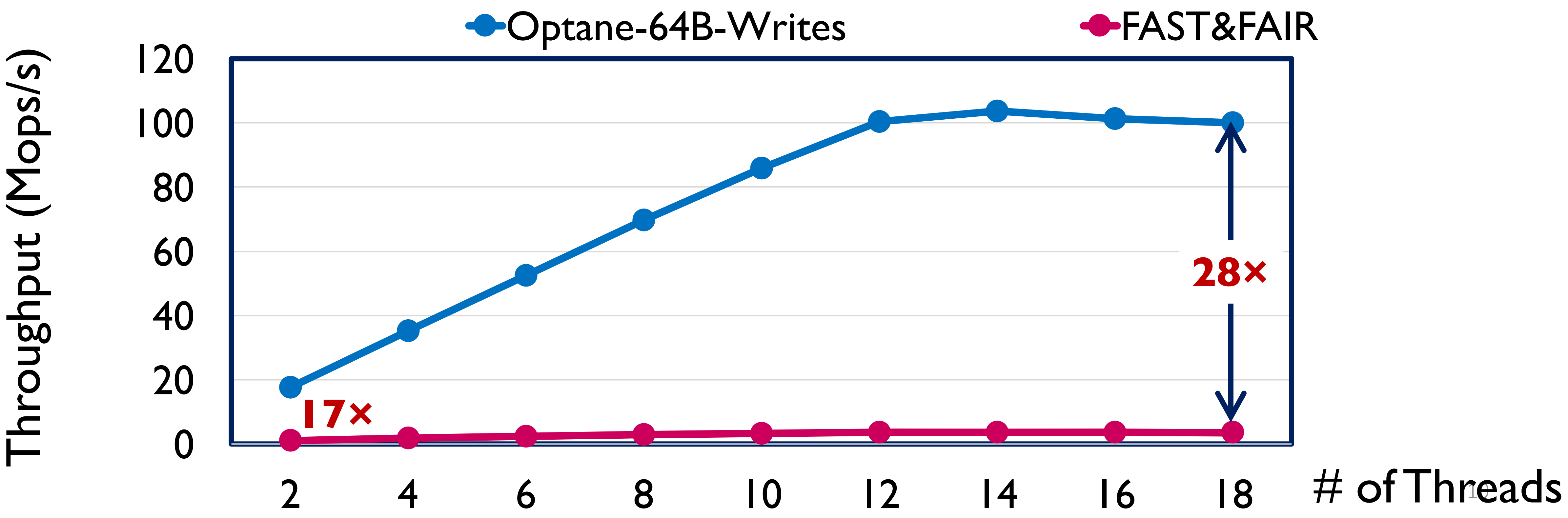


Images are reshaped from "An Empirical Guide to the Behavior and Use of Scalable Persistent Memory", FAST'20

Overhead of Accessing Granularity Mismatch

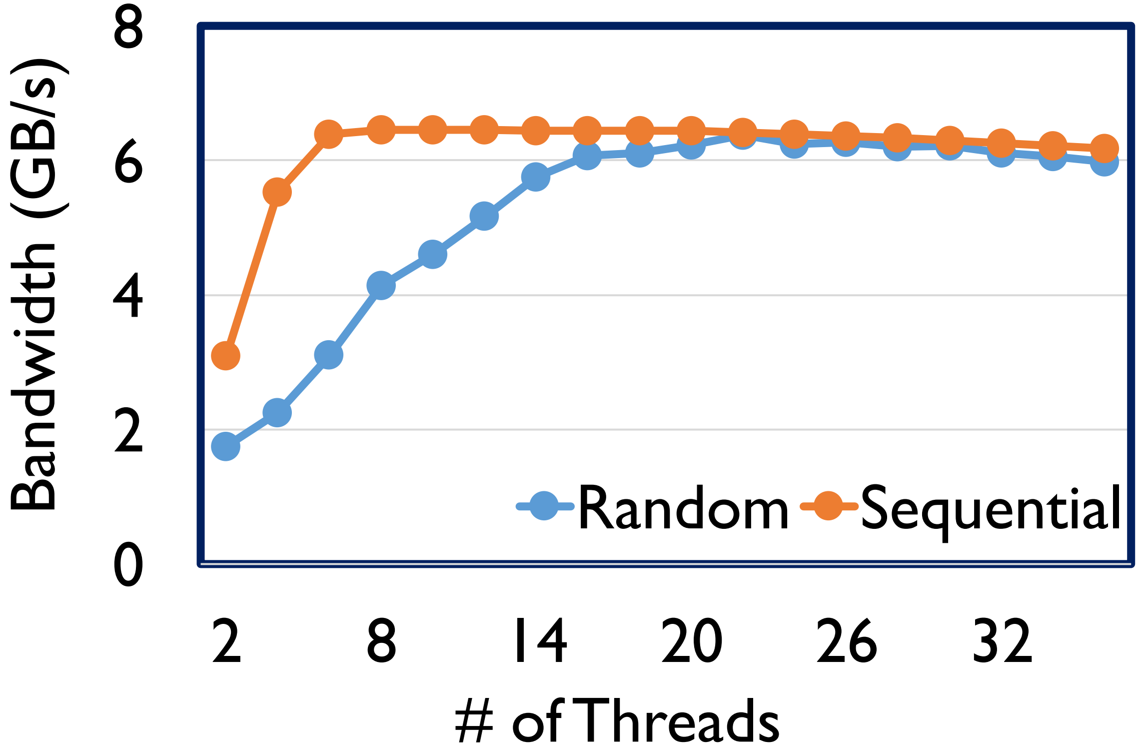
FAST&FAIR [FAST'18]: State-of-the-art Persistent B$^+$-Tree data structure

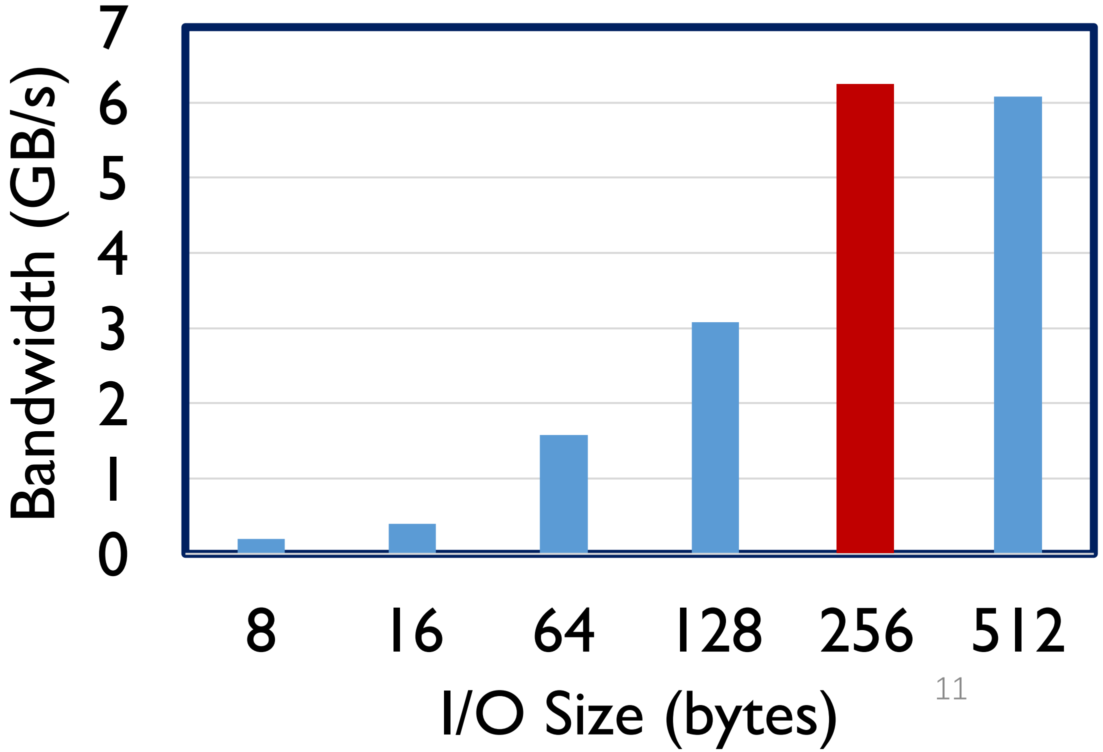
- Avoids logging and doesn't block reads by using synchronized 8-B atomic operations
- Sort & balance overhead



When Log Structure Meets Optane DCPMM

- Random and sequential accesses achieve the same peak performance
- Minimal IO units to saturate bandwidth: **256-byte blocks**
  - It is not beneficial to batch more data than a single I/O unit (i.e., 256 B)
- Batching increases latency inevitably

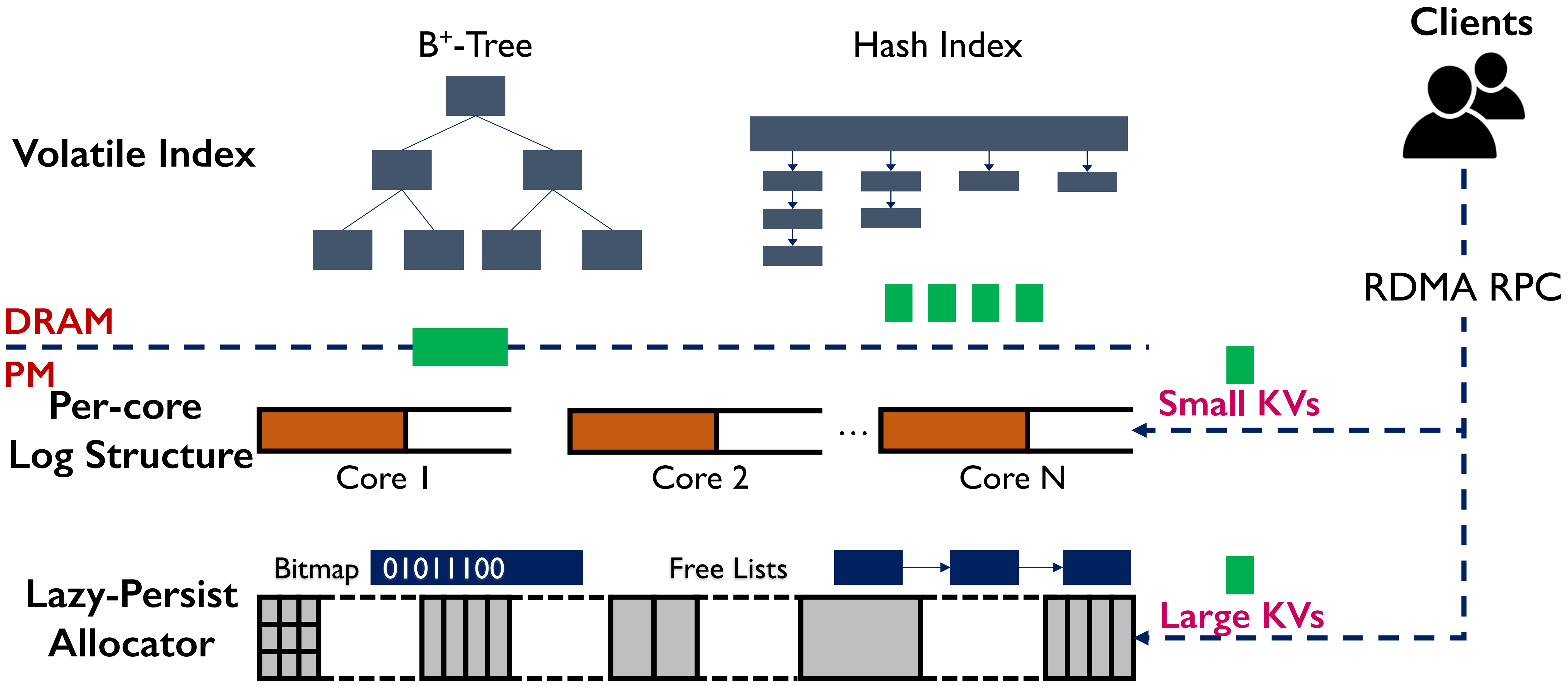




Outline

- Introduction
- Optane DC Persistent Memory Module
- FlatStore: An Efficient Log-structured Storage Engine
- Results
- Summary & Conclusion

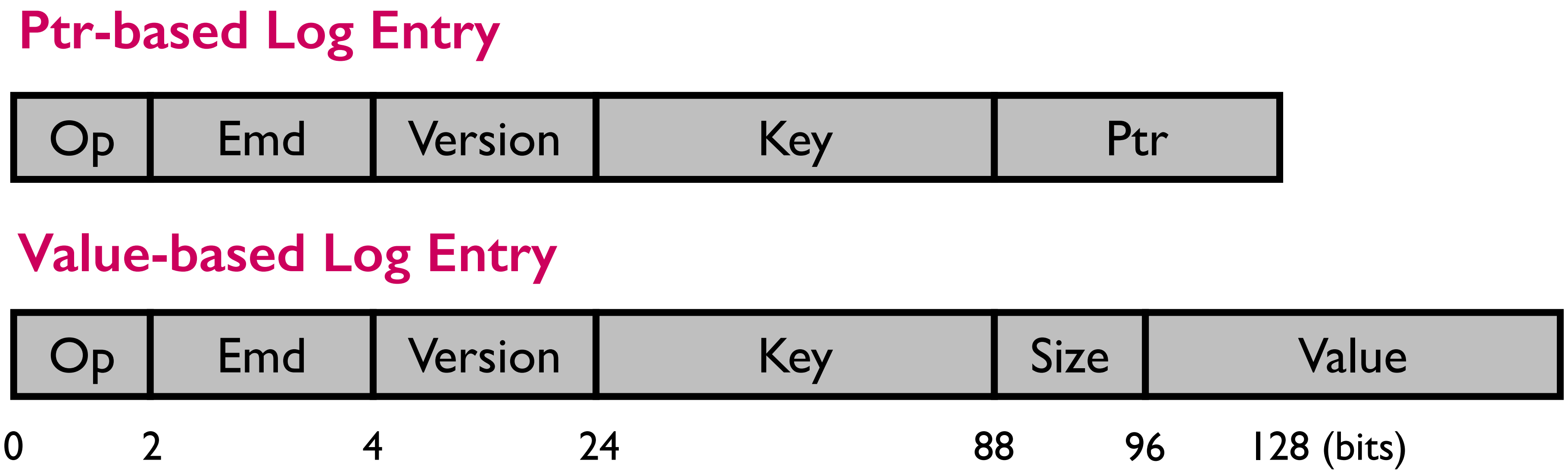
Overall Architecture of FlatStore



Compacted Log Format

Log entries are formatted via the **operation log technique**

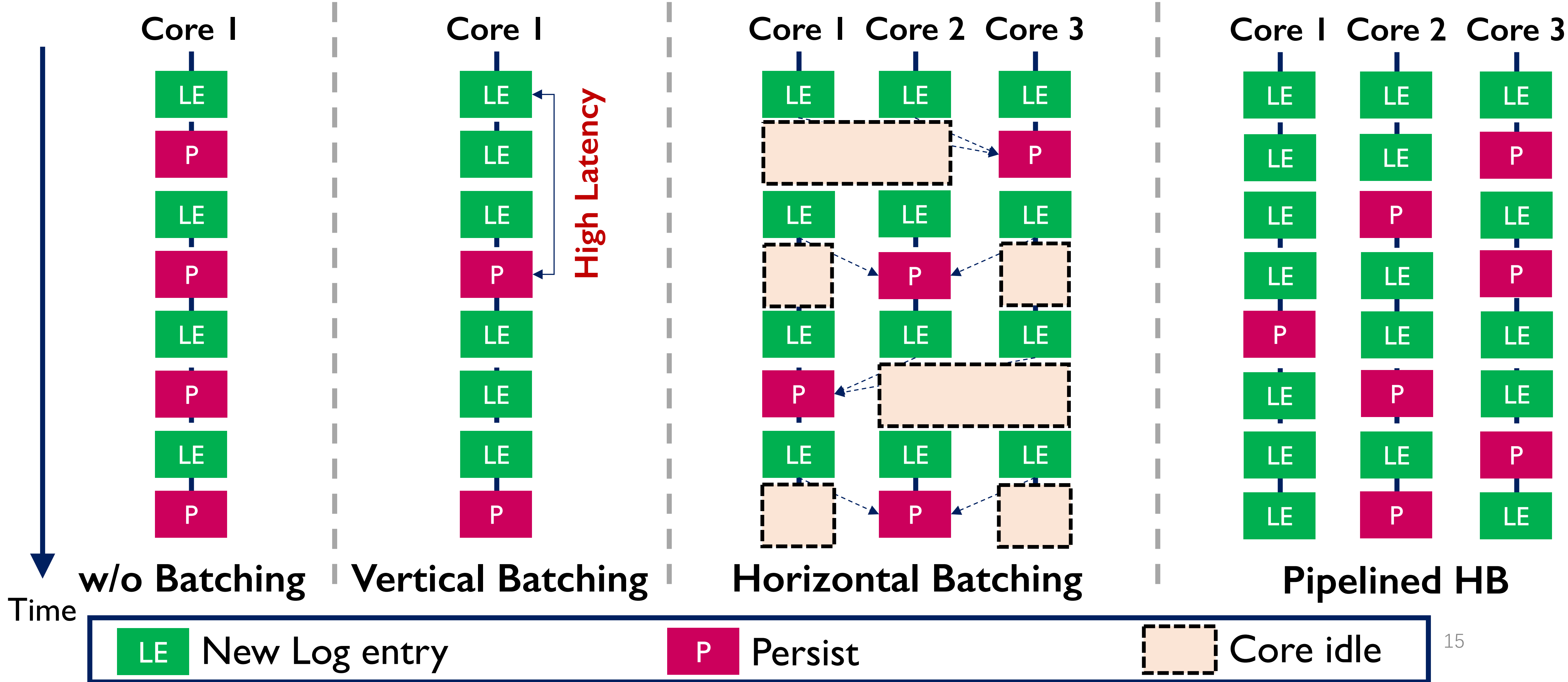
- Describe each operation, instead of recording the value



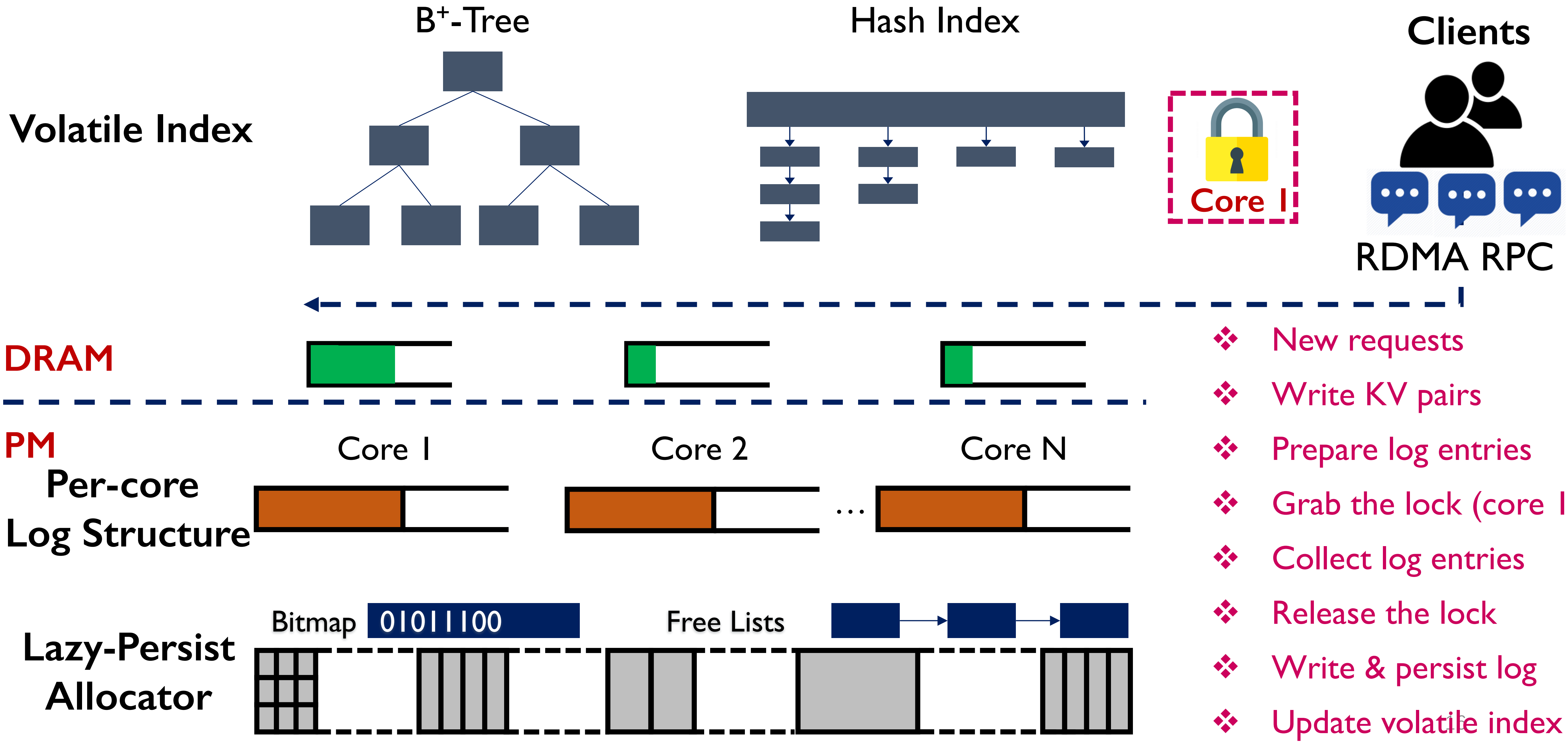
- **16 log entries** (256-byte) can be flushed to Optane DC altogether

Pipelined Horizontal Batching

Common wisdom: Batching increases both throughput and latency



Putting it all together



- New requests
- Write KV pairs
- Prepare log entries
- Grab the lock (core 1)
- Collect log entries
- Release the lock
- Write & persist log
- Update volatile index

More design details: Check our paper

Lazy-persist allocator are used to store large KV pairs

- Bitmaps describing the allocation states don't need to be persisted synchronously, since the log entries has already record such information

Grouping the cores to conduct pipelined horizontal batching

- The size of each group balances the contention level and batching opportunity

Non-blocking parallel log cleaning

- Obsolete log entries are reclaimed concurrently without blocking the front-end operations

Recovery of the volatile index

- Volatile index are kept in DRAM and is vulnerable to system/power failures

Outline

- Introduction
- Optane DC Persistent Memory Module
- FlatStore: An Efficient Log-structured Storage Engine
- Results
- Summary & Conclusion

Experimental Setup

Hardware Platform

Server Node	4 Optane DCPMMs (**1TB**), 2 Xeon Gold 6240m CPUs (**36 cores**), 128 GB DRAM
Client Nodes x11	2 Xeon E5-2650 v4 CPUs (**24 cores**), 128 GB DRAM
Switch	Mellanox MSB7790-ES2F Switch (**100 Gbps**)

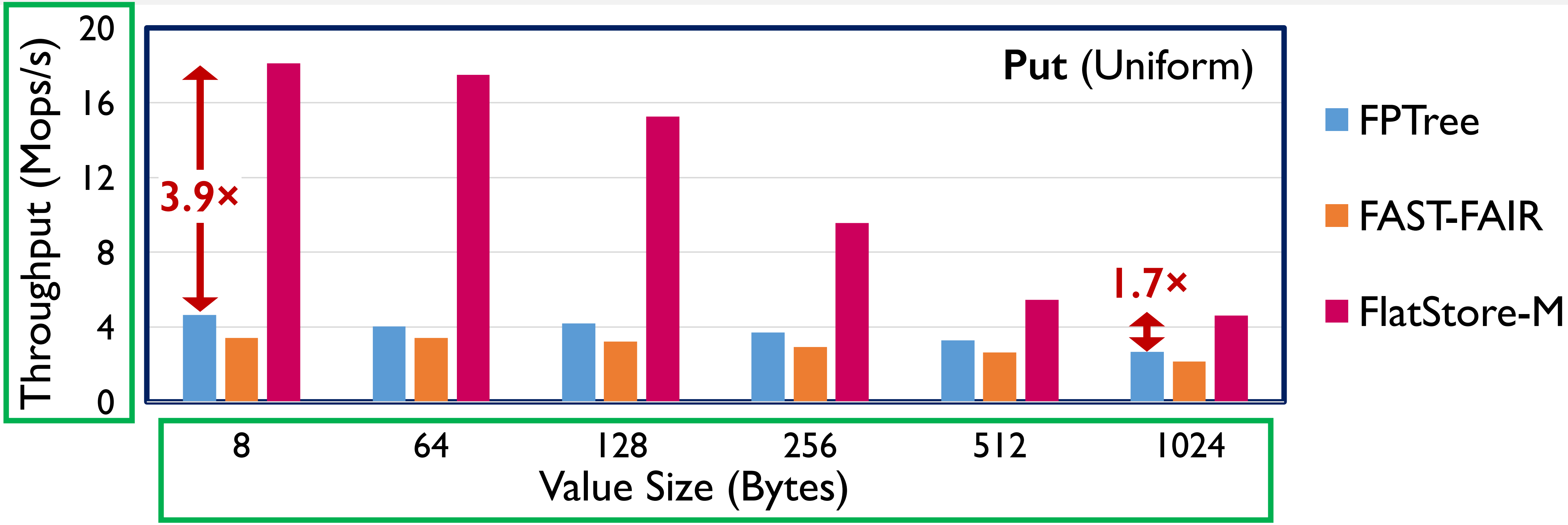
Compared Systems

Hash-based	CCEH	Three level (directory, segments, buckets), 4 slots in a bucket
	Level-Hashing	Two-level (top/bottom level), 4 slots in a bucket
Tree-based	FPTree	Inner nodes are placed in DRAM.
	FAST&FAIR	All nodes are placed in PM.

Workloads

- Facebook ETC Pool: Mixture of small & large KV pairs
- YCSB (varying r/w ratio, item size, skewness, etc)

Micro-benchmark: YCSB



FlatStore's performance is 3.9× higher than FPTree (2nd best) for 8-byte values

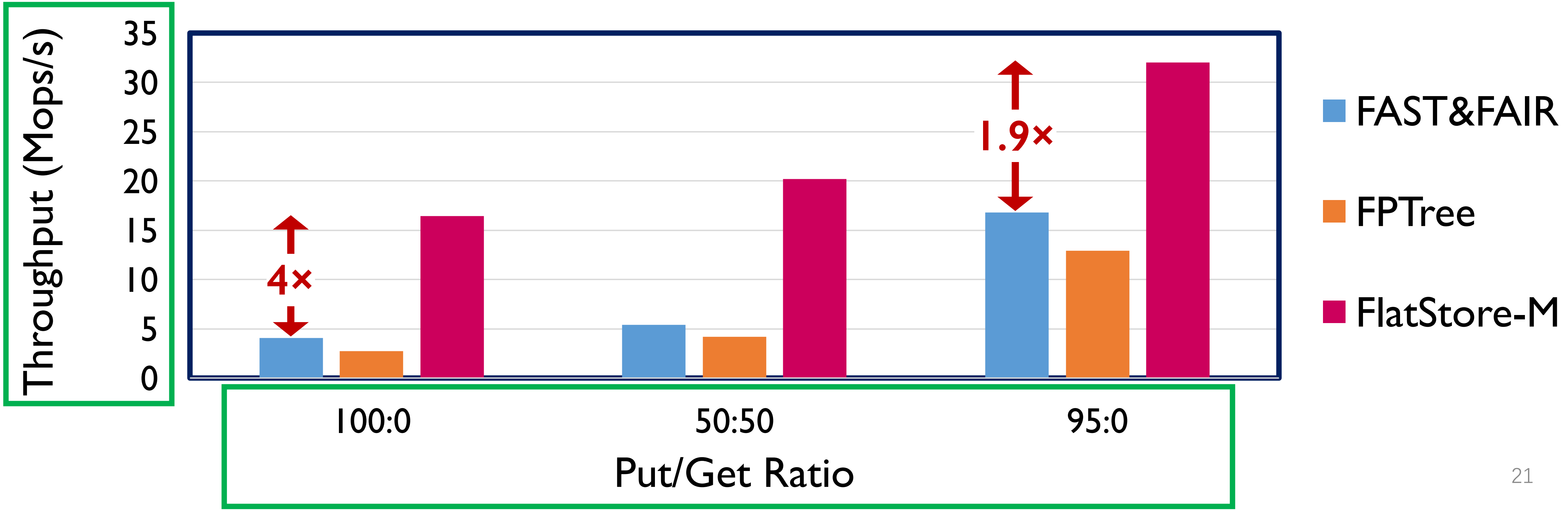
- Multiple small values can be persisted together
- FlatStore doesn't introduce structural modification overhead

For large values (e.g., 1024-byte), FlatStore still shows 1.7× higher throughput

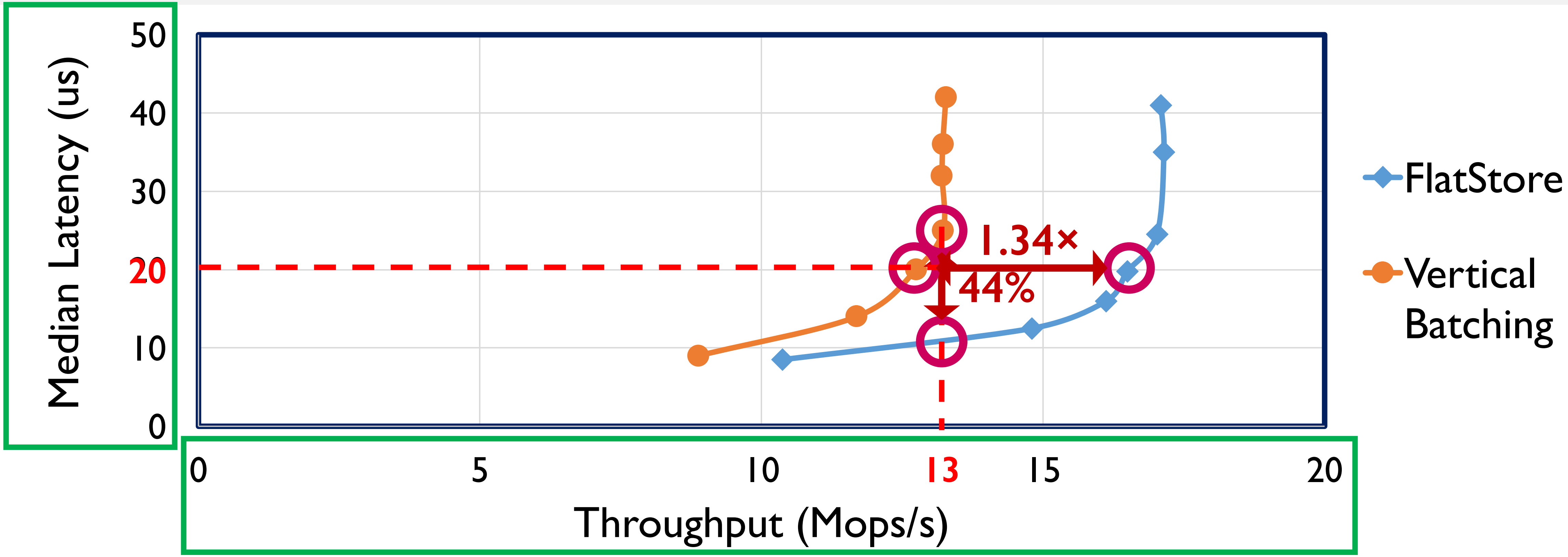
Macro-benchmark: Facebook ETC Pool

Facebook ETC Pool: mixture of small & large KV pairs.

- **Tiny** (1-13 bytes, 40%), zipfan distribution
- **Small** (14-300 bytes, 55%), zipfan distribution
- **Large** (> 300 bytes, 5%), uniform distribution



Pipelined Horizontal Batching: Latency Reduction



- By introducing pipelined horizontal batching, FlatStore uses less time to collect a batch, thus achieving lower latency
- Pipelined HB contributes to improving the performance, since it dynamically collect a batch, instead of using a predefined threshold (e.g., minimal batch size)

Summary & Conclusion

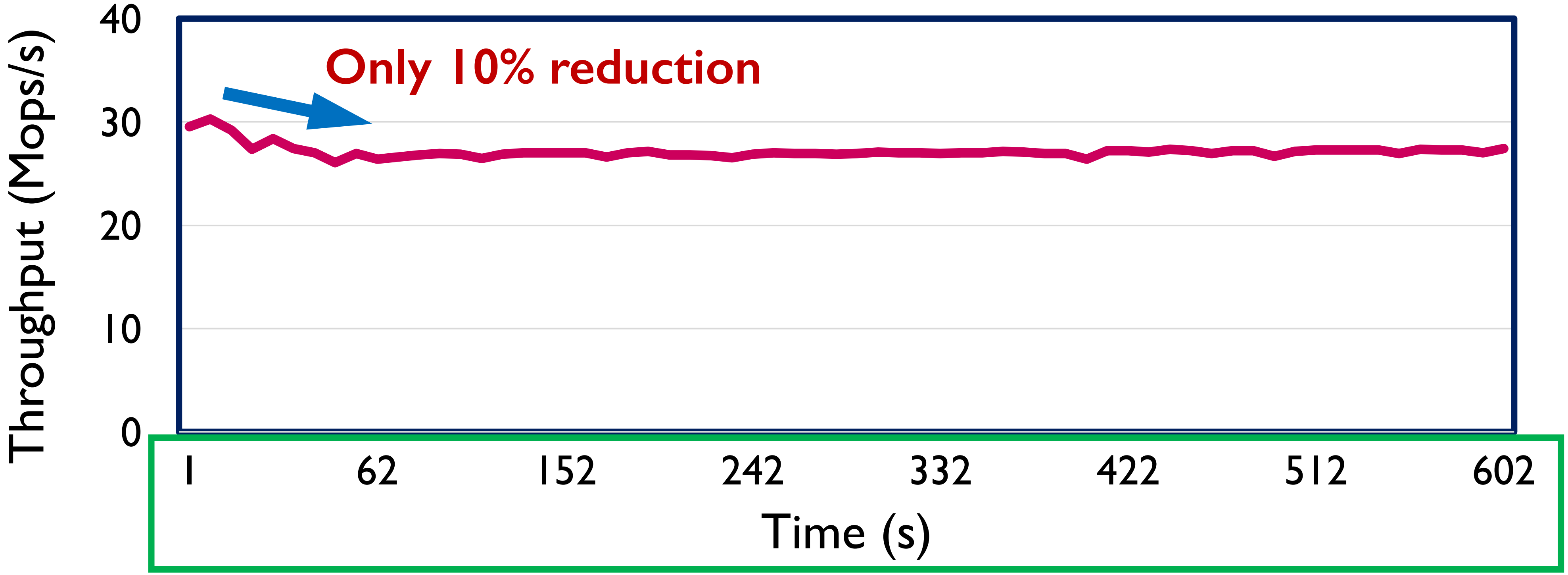
- Real PM device — Optane DCPMMs — exhibit much **different hardware properties** from what we assumed, which make many existing optimizations inapplicable
- We propose FlatStore to revitalize the **log-structured design on Optane Memory**. Key insight: **Selective batch** to maximize the potential performance
  - Compacted Log Format
  - Pipelined Horizontal Batching
- FlatStore supports hash- and tree-based index structure, which is **2.5 – 6.3 times** faster than existing systems.

Thanks & QA

Tsinghua University & The Ohio State University

http://storage.cs.tsinghua.edu.cn

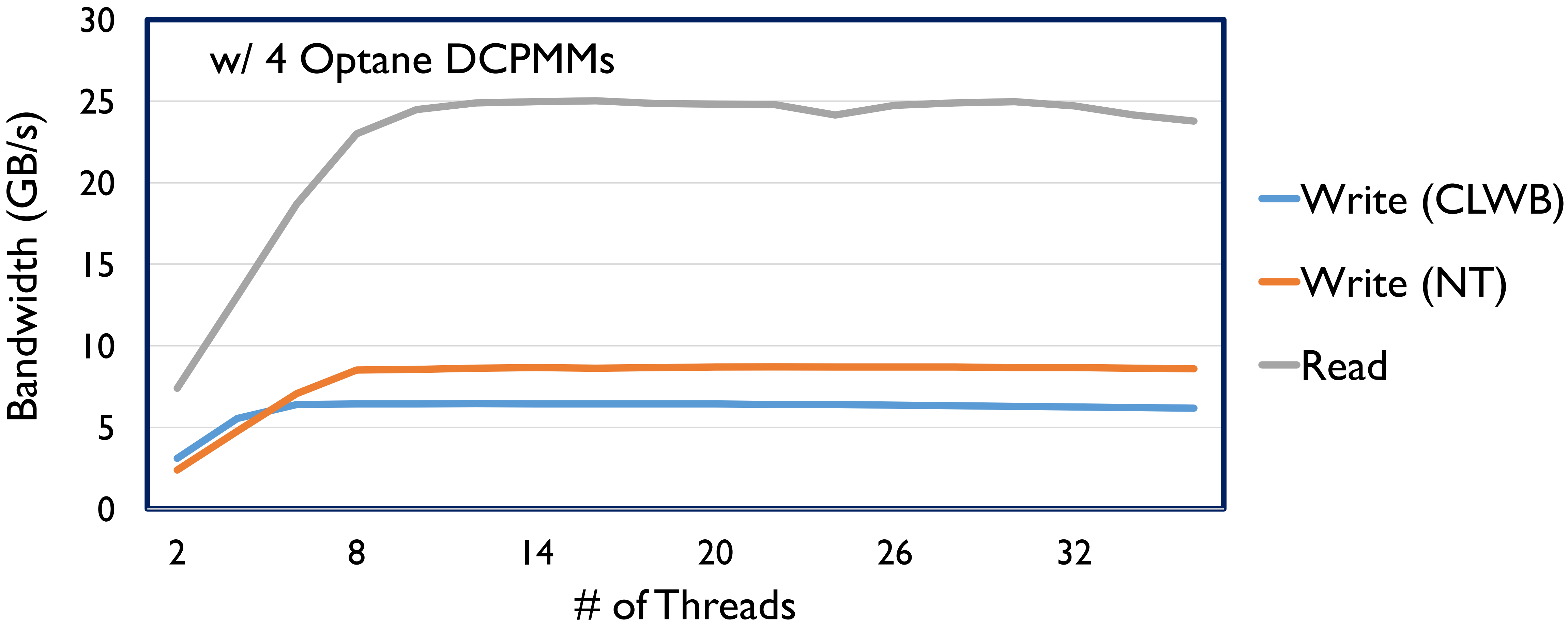
Log Cleaning Overhead



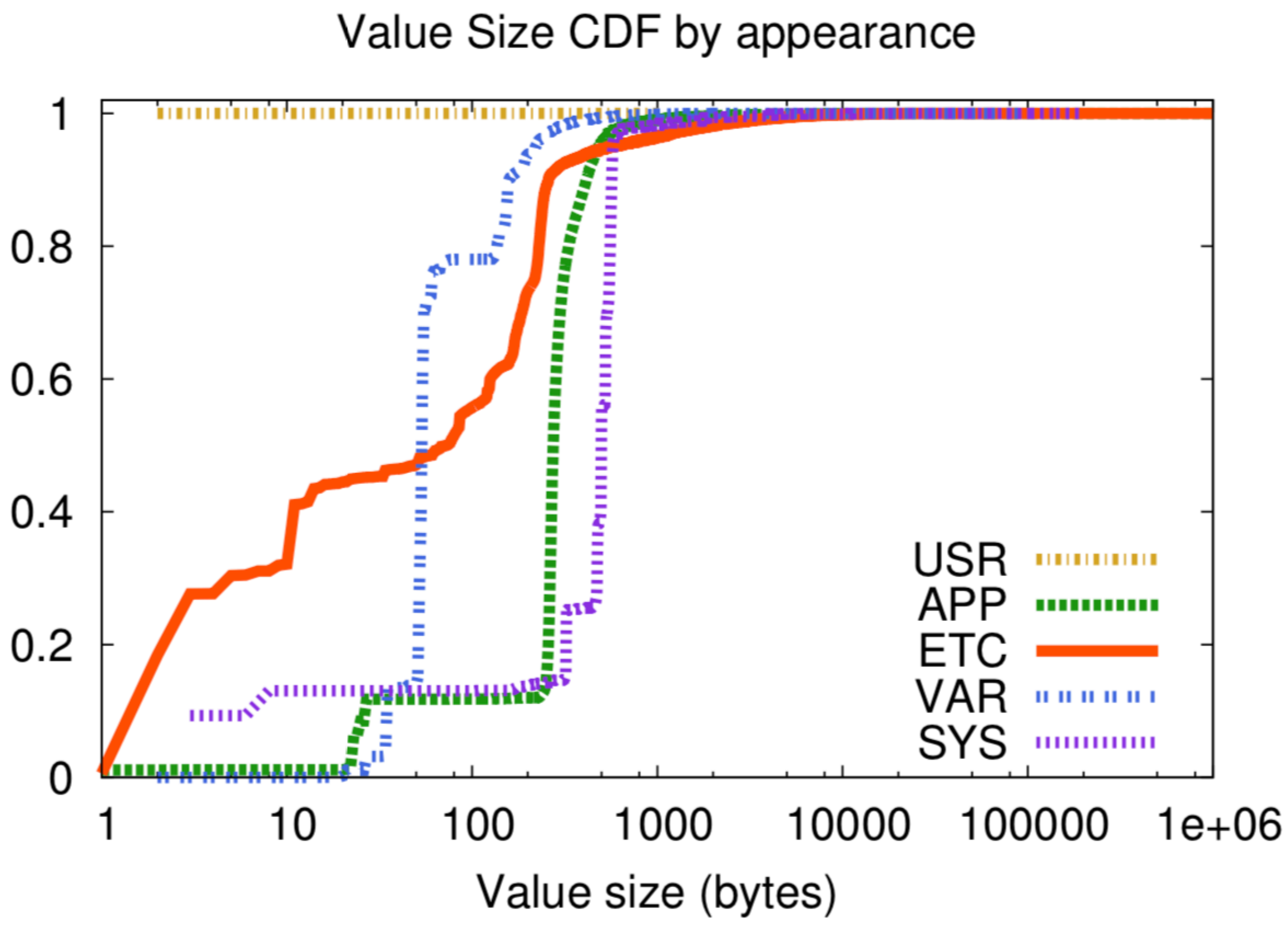
Workload: YCSB (64B values)

- background cleaner reclaims the blocks without blocking the normal requests
- Log-structure only contains small-sized metadata or KV items
- Multiple GC groups (check our paper for details)

Basic Performance of Optane DCPMMs



Value size distribution in real-world workloads

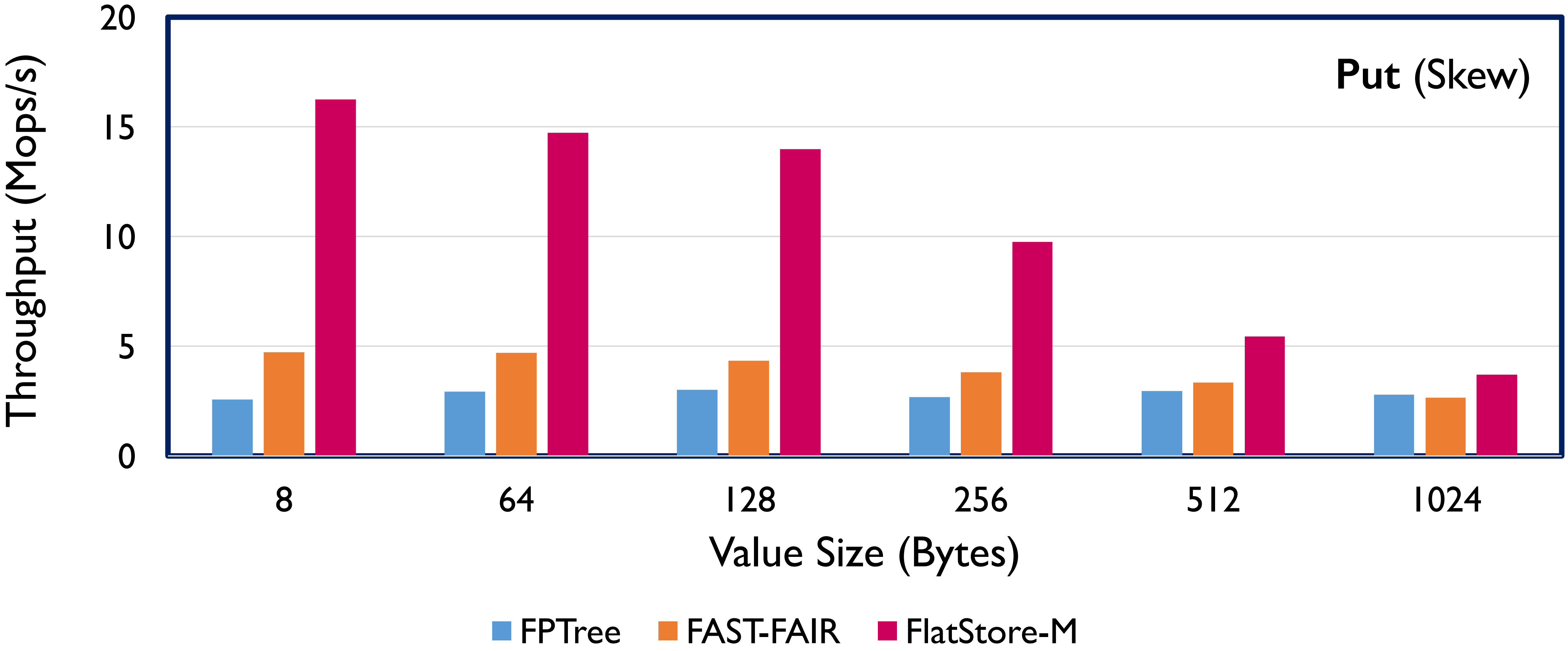


pool	p5	p25	p50	p75	p95	p99
wildcard	77	102	169	363	3.65 K	18.3 K
app	103	247	269	337	1.68K	10.4 K
replicated	62	68	68	68	68	68
regional	231	824	5.31 K	24.0 K	158 K	381 K

Table comes from "Scaling Memcache at Facebook", NSDI'13

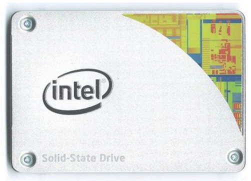
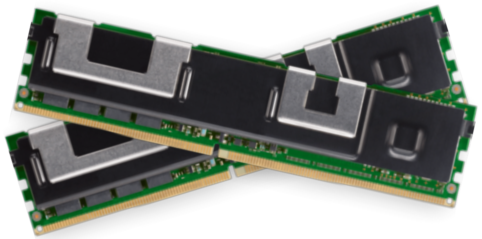
Figure comes from "Workload Analysis of a Large-Scale Key-Value Store ", SIGMETRICS'12

Micro-benchmark: YCSB



Using a log structure: An intuitive approach

Q: Can a log structure still retain its benefits with Optane DCPMMs?

	 	
w/o data journaling	🙂	GC: 🙂 or 😢 ?
Sequential access	🙂	😐
Batching	🙂	😐