PRISM: Optimizing Key-Value Store for Modern Heterogeneous Storage Devices

Yongju Song, Wook-Hee Kim, Sumit Kumar Monga,

Changwoo Min, and Young Ik Eom

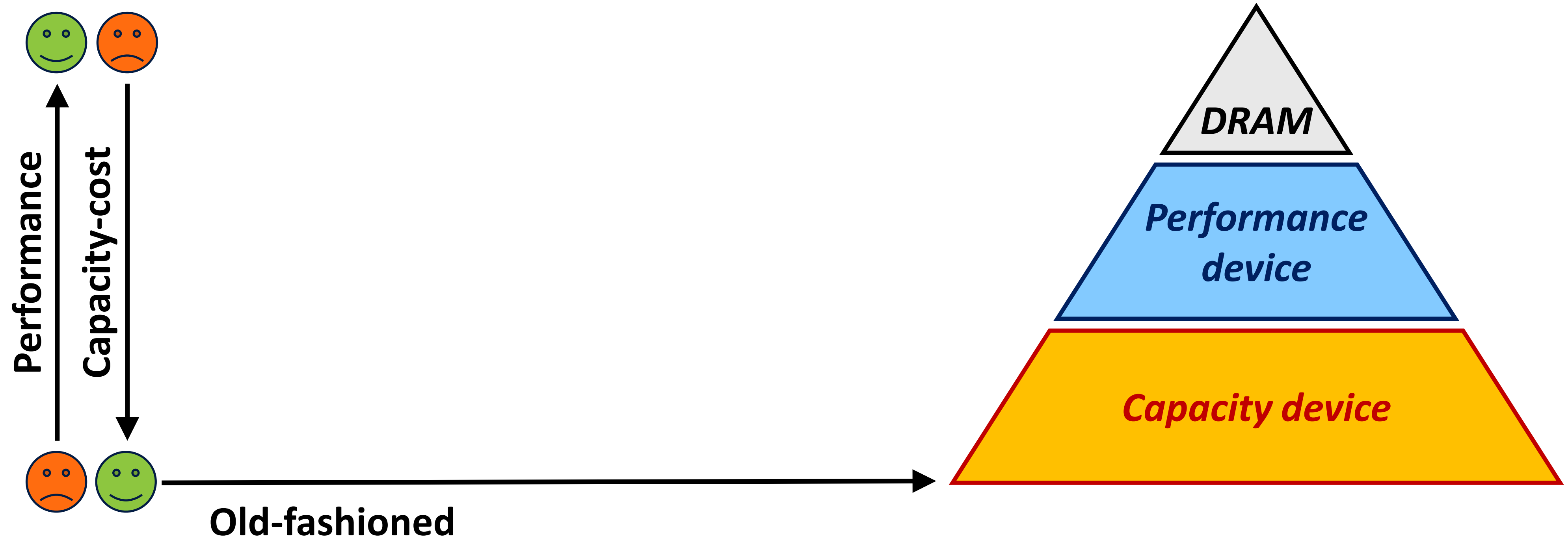
SUNGKYUNKWAN UNIVERSITY 1398 성균관대학교

KONKUK UNIVERSITY KOREA 1931 건국대학교

VIRGINIA TECH

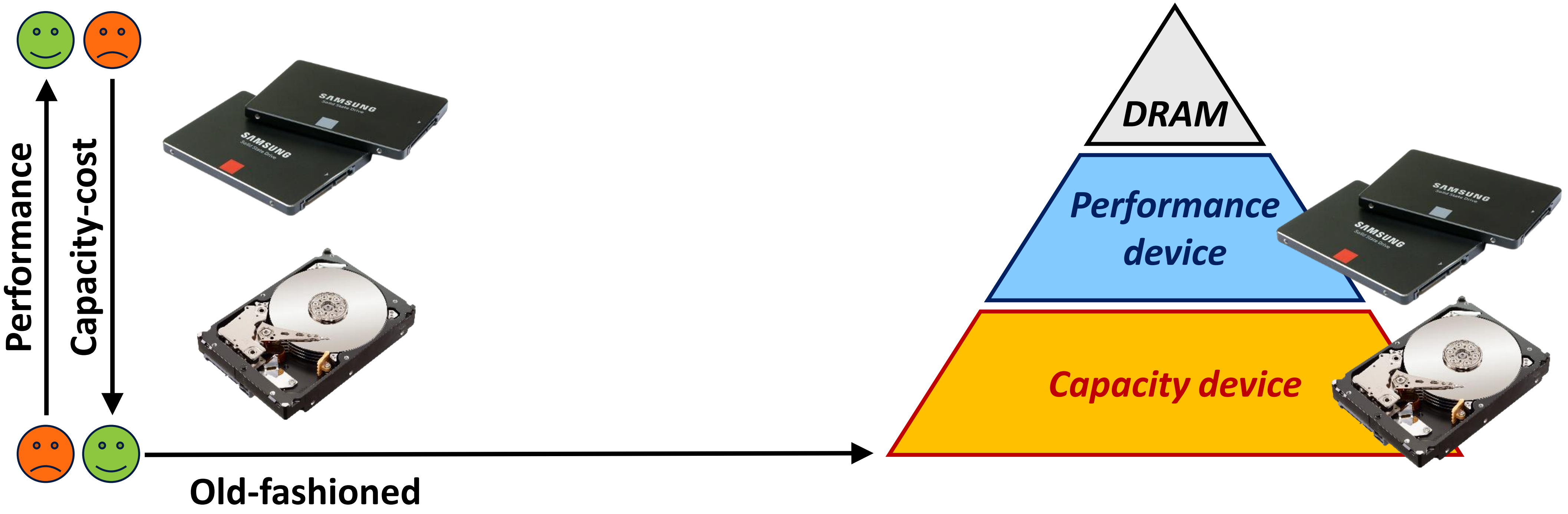
Heterogeneous Storage Systems

A method for assigning different categories of data to *various types of storage media* to *reduce overall storage costs*.



Heterogeneous Storage Systems

A method for assigning different categories of data to *various types of storage media* to *reduce overall storage costs*.



Performance

Capacity-cost

Old-fashioned

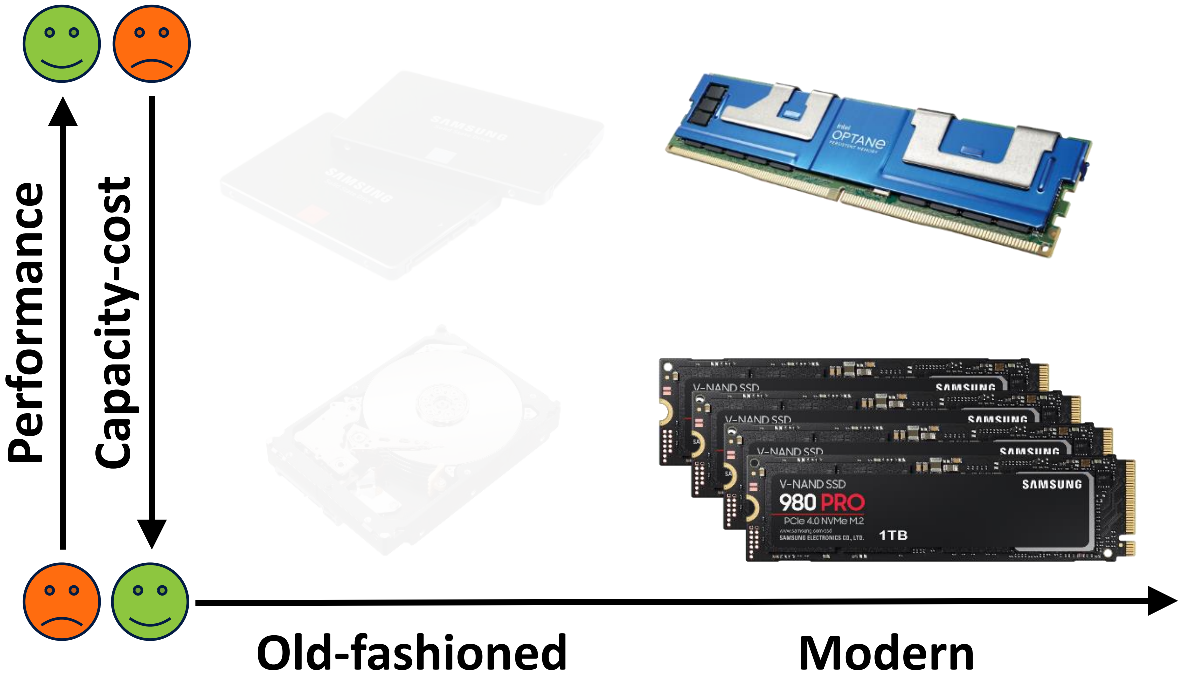
DRAM

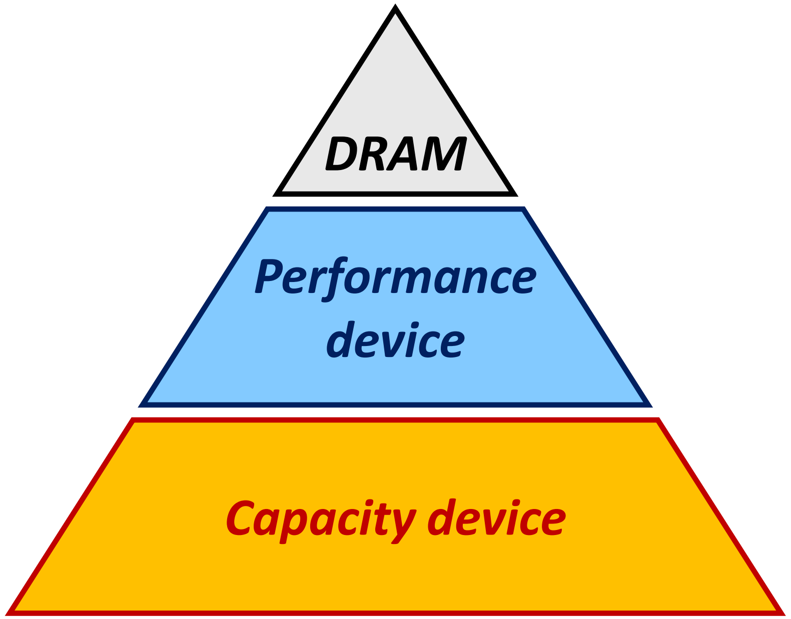
Performance device

Capacity device

Heterogeneous Storage Systems

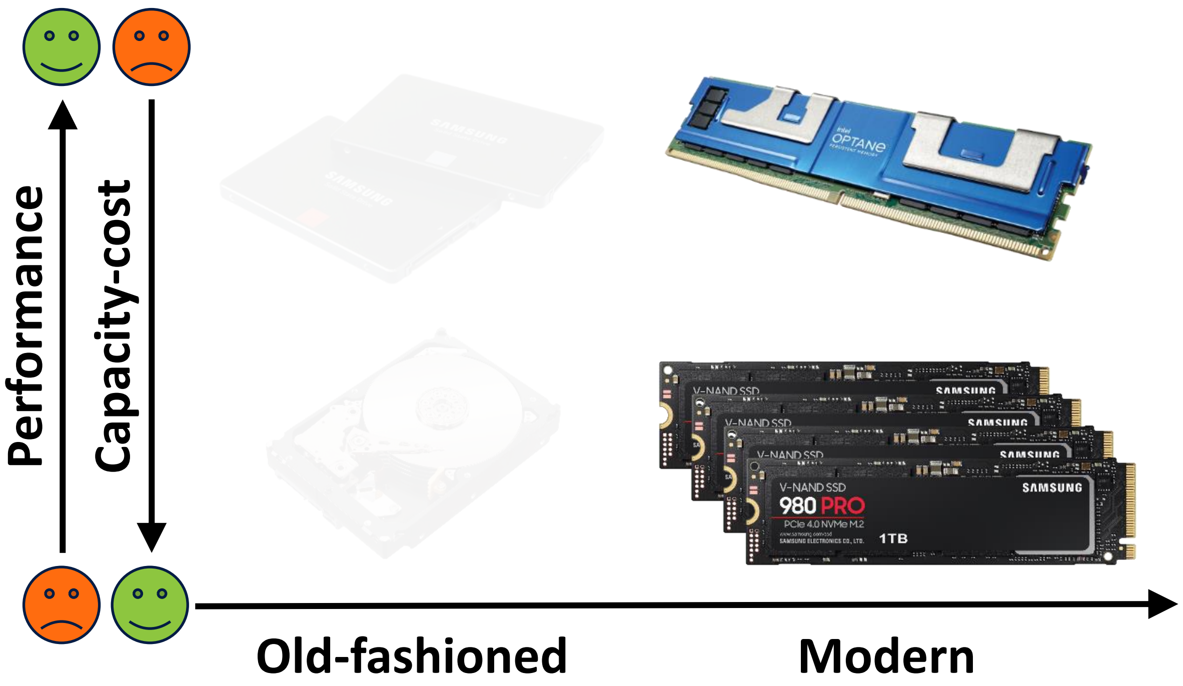
A method for assigning different categories of data to *various types of storage media* to *reduce overall storage costs*.





Heterogeneous Storage Systems

A method for assigning different categories of data to *various types of storage media* to *reduce overall storage costs*.

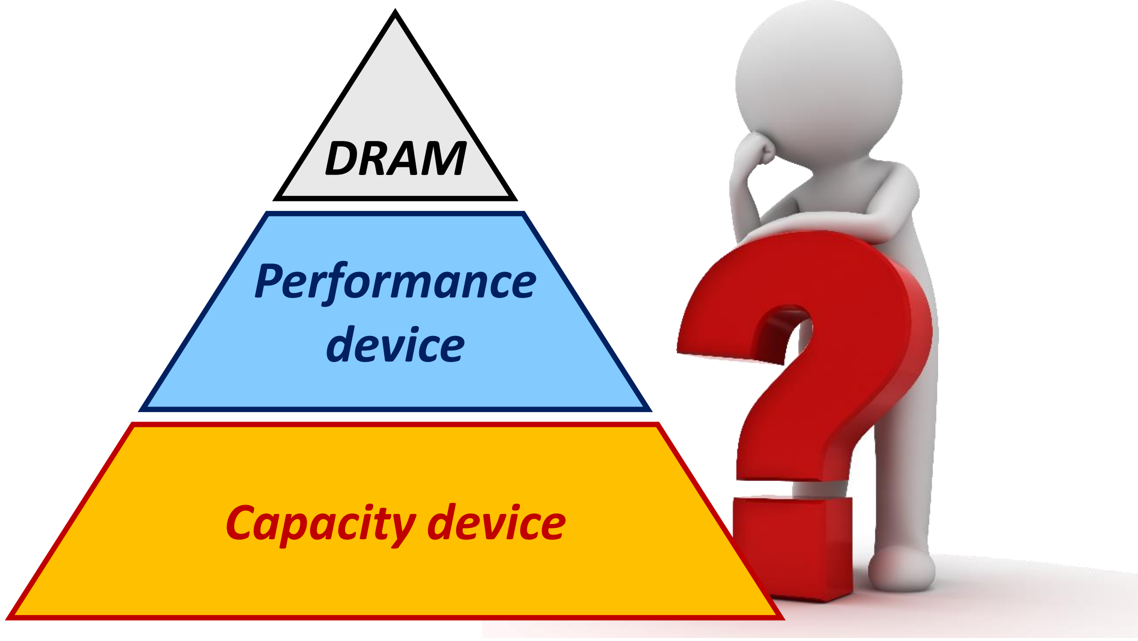


Performance

Capacity-cost

Old-fashioned

Modern



DRAM

Performance device

Capacity device

Evolution of Storage Heterogeneity

Specification		Capacity	Cost	Performance	
Type	Model	GB	$/TB	Read Latency (usec)	Write Latency (usec)
DRAM	SK Hynix DRAM w/DDR4	16	5,427	0.08	0.08
NVM	Intel Optane DCPMM w/DDR-T	128	4,096	0.30	0.09
NVM SSD	Intel Optane 905P w/PCIe 3	960	1,024	10	10
Flash SSD	Samsung 980 Pro w/PCIe 4	1024	150	50	20
Flash SSD	Samsung 980 w/PCIe 3	1024	100	60	20

Evolution of Storage Heterogeneity

	Specification		Capacity	Cost	Performance	
	Type	Model	GB	$/TB	Read Latency (usec)	Write Latency (usec)
Performance devices	DRAM	SK Hynix DRAM w/DDR4	16	5,427	0.08	0.08
	NVM	Intel Optane DCPMM w/DDR-T	128	4,096	**0.30**	**0.09**
Capacity devices	NVM SSD	Intel Optane 905P w/PCIe 3	960	1,024	10	10
	Flash SSD	Samsung 980 Pro w/PCIe 4	**1024**	**150**	50	20
	Flash SSD	Samsung 980 w/PCIe 3	**1024**	**100**	60	20

Evolution of Storage Heterogeneity

						✓	✓	✓
Specification		Capacity	Cost	Performance				Endurance
Type	Model	GB	$/TB	Read Latency (usec)	Write Latency (usec)	Read BW (GB/s)	Write BW (GB/s)	Warranty (PBW)
DRAM	SK Hynix DRAM w/DDR4	16	5,427	0.08	0.08	15	15	∞
NVM	Intel Optane DCPMM w/DDR-T	128	4,096	**0.30**	**0.09**	6.8	1.9	**292**
NVM SSD	Intel Optane 905P w/PCIe 3	960	1,024	10	10	2.6	2.2	17.52
Flash SSD	Samsung 980 Pro w/PCIe 4	**1024**	**150**	50	20	**7**	**5**	0.6
Flash SSD	Samsung 980 w/PCIe 3	**1024**	**100**	60	20	3.5	3	0.6

Evolution of Storage Heterogeneity

						✓	✓	✓
Specification		Capacity	Cost	Performance				Endurance
Type	Model	GB	$/TB	Read Latency (usec)	Write Latency (usec)	Read BW (GB/s)	Write BW (GB/s)	Warranty (PBW)
DRAM	SK Hynix DRAM w/DDR4	16	5,427	0.08	0.08	15	15	∞
NVM	Intel Optane DCPMM w/DDR-T	128	4,096	***0.30***	***0.09***	6.8	1.9	***292***
NVM SSD	Intel Optane 905P w/PCIe 3	960	1,024	10	10	2.6	2.2	17.52
Flash SSD	Samsung 980 Pro w/PCIe 4	***1024***	***150***	50	20	***7***	***5***	0.6
Flash SSD	Samsung 980 w/PCIe 3	***1024***	***100***	60	20	3.5	3	0.6

There is ***No Clear Separation*** between performance-/capacity- devices.

"The Storage Hierarchy is ***Becoming a Jungle***." [CIDR'21, Dong Xie]

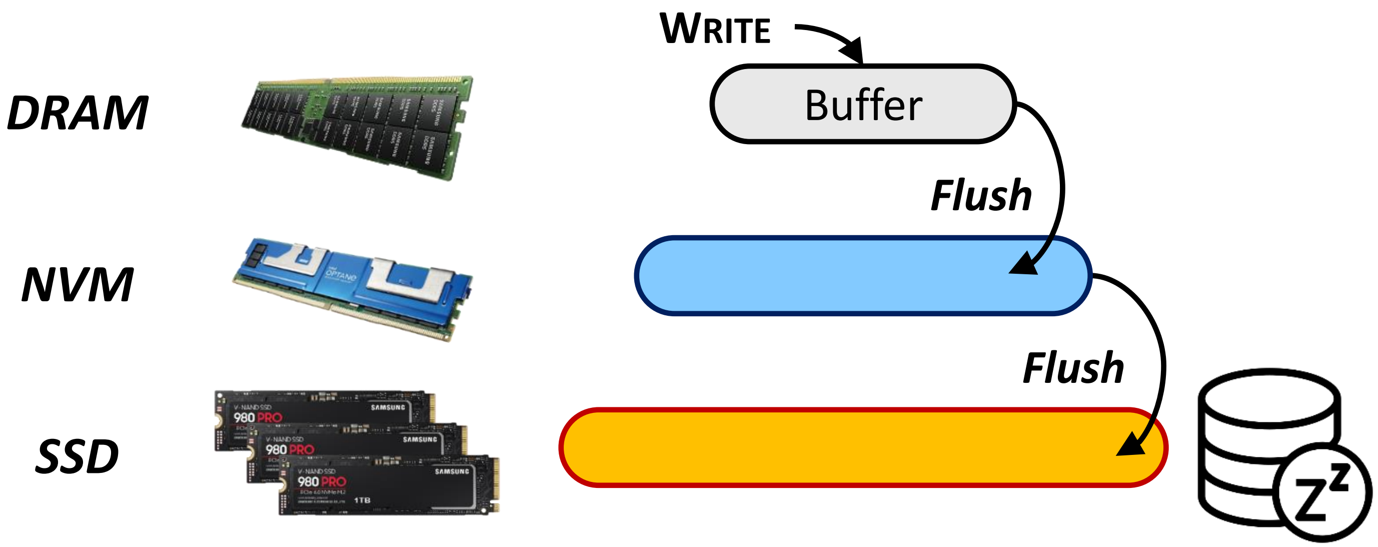
"The Storage Hierarchy is ***Not a Hierarchy***." [FAST'21, Remzi H. Arpaci-Dusseau]

Today’s Storage Hierarchy

Today’s Storage Hierarchy

Placing hot data on NVM

- System can leverage the low latency of NVM but *suffer from its limited bandwidth*.



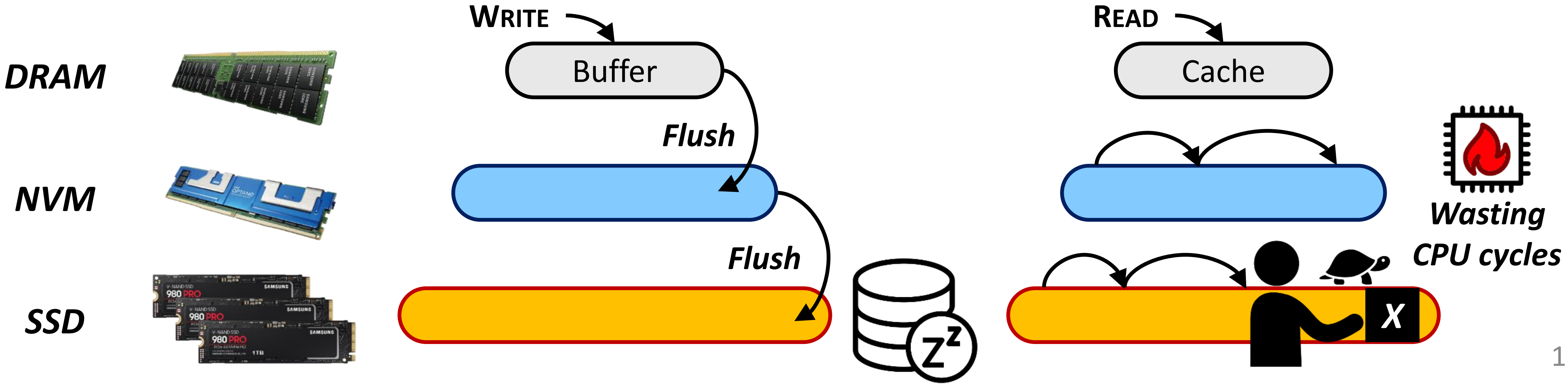
Today’s Storage Hierarchy

Placing hot data on NVM

- System can leverage the low latency of NVM but *suffer from its limited bandwidth*.

Traversing data layer by layer for handling read requests

- Inefficient traversal leads to *wasting CPU cycles*.
- Overall performance may be *bounded to the device with the lowest performance*.



Today's Storage Hierarchy

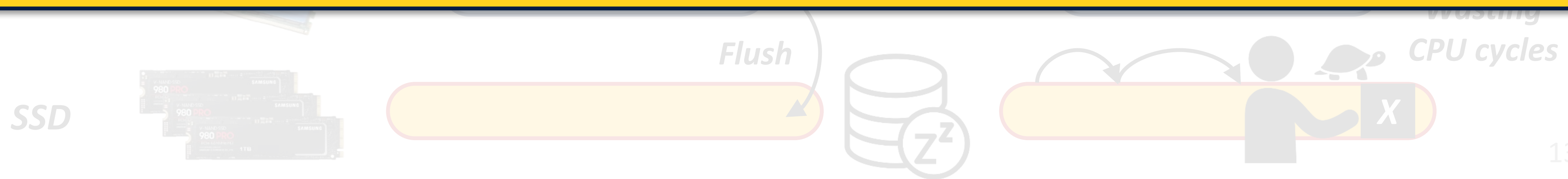
Placing hot data on NVM

- System can leverage the low latency of NVM but suffer from its limited bandwidth.

Traversing data layer by layer for handling read requests

- Inefficient traversal leads to wasting CPU cycles.
- Overall performance may be bounded to the device with the lowest performance.

How should we design a Heterogeneous Storage System in the Modern Storage Landscape?



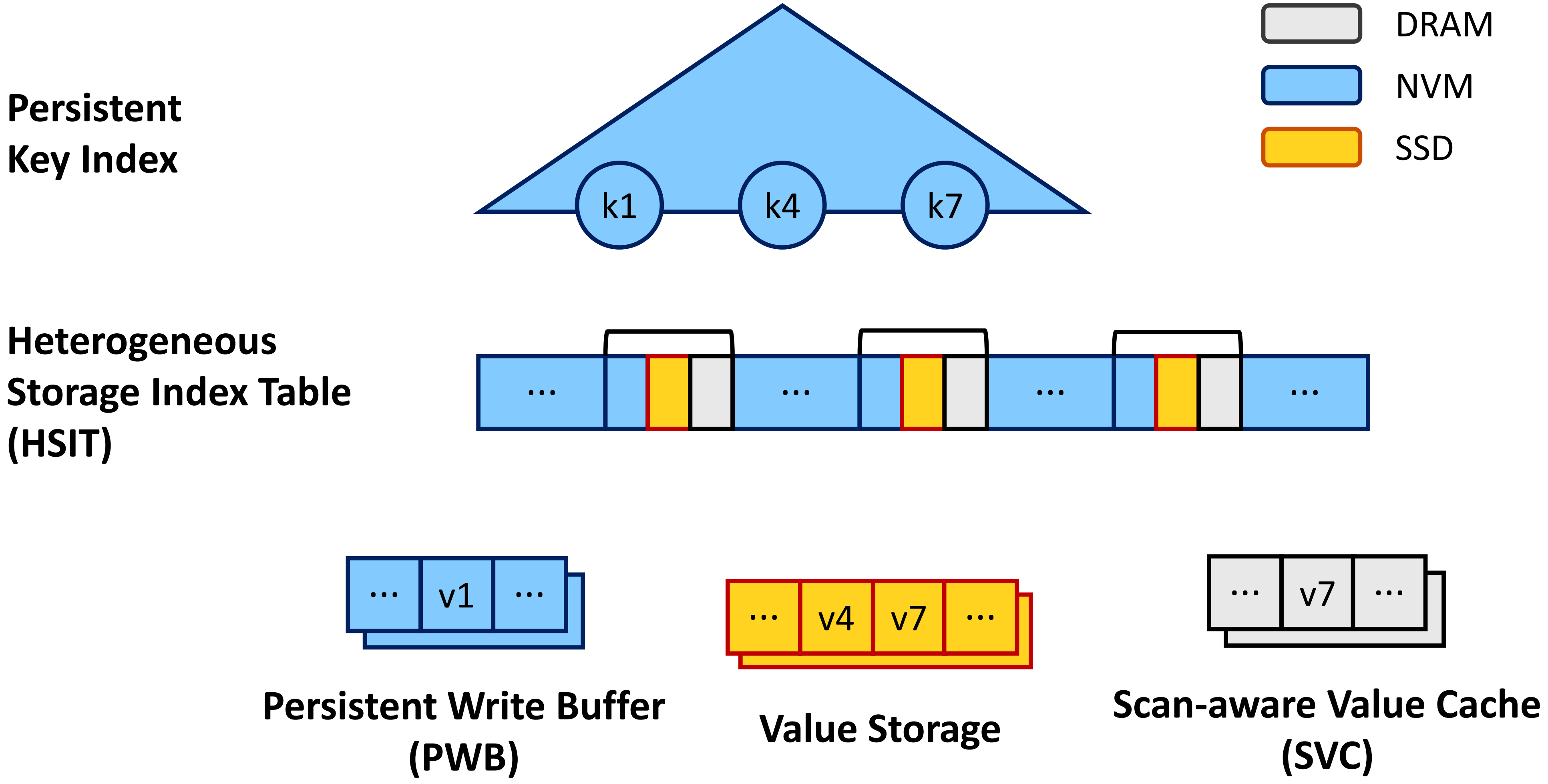
Design Goals of *PRISM*

Drawing the full potential of heterogeneous storage devices.

Minimizing the overhead of software stack for scalability

Providing a high level of crash consistency & concurrency

Overview of *PRISM*



Persistent Key Index

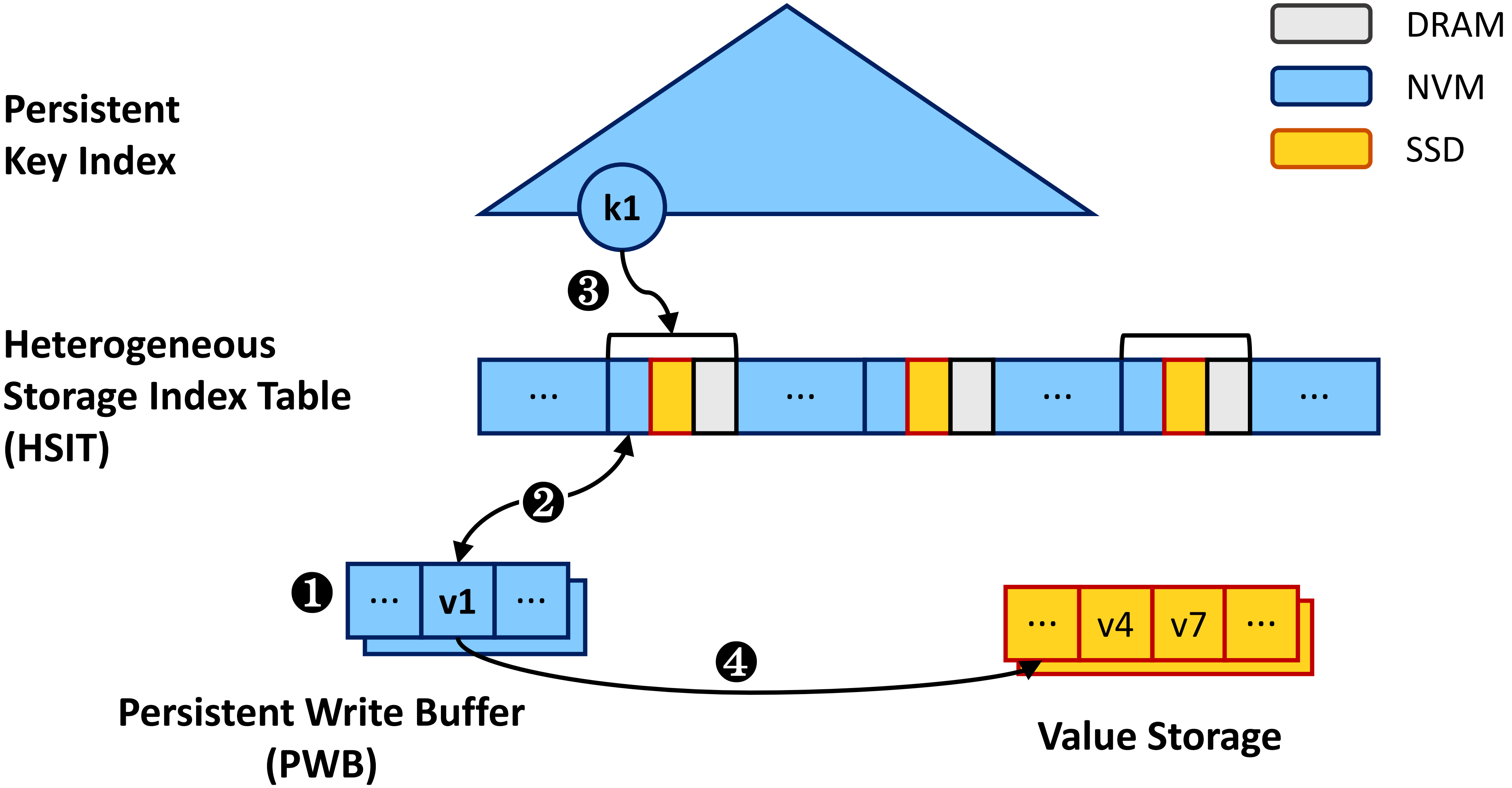
Heterogeneous Storage Index Table (HSIT)

Persistent Write Buffer (PWB)

Value Storage

Scan-aware Value Cache (SVC)

Overview of *PRISM*: Insert (k1, v1)


DRAM
NVM
SSD
Persistent Key Index
k1
❸
Heterogeneous Storage Index Table (HSIT)
…
…
…
…
❷
❶
…
v1
…
Persistent Write Buffer (PWB)
❹
…
v4
v7
…
Value Storage

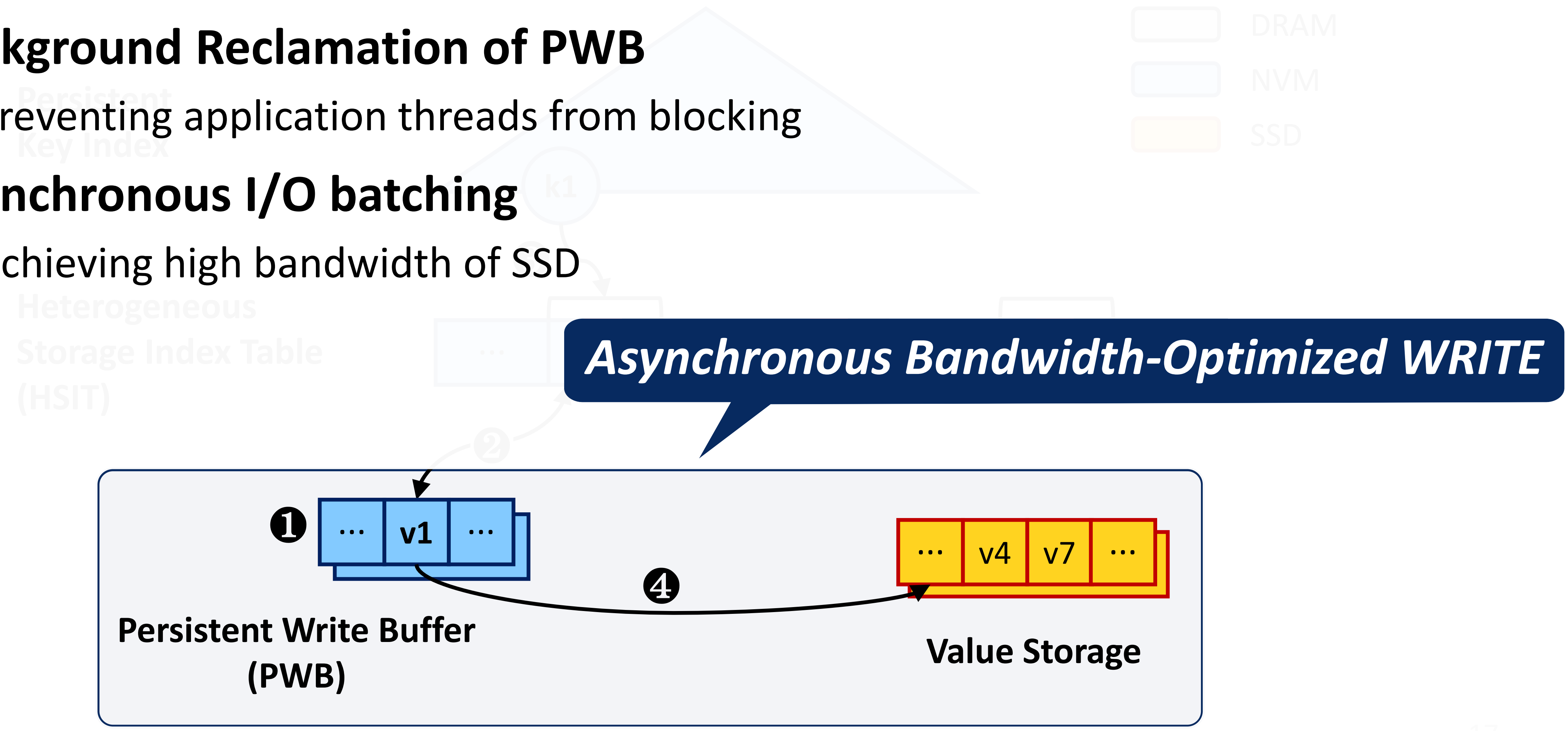
Overview of *PRISM*: Insert (k1, v1)

Background Reclamation of PWB

- Preventing application threads from blocking

Asynchronous I/O batching

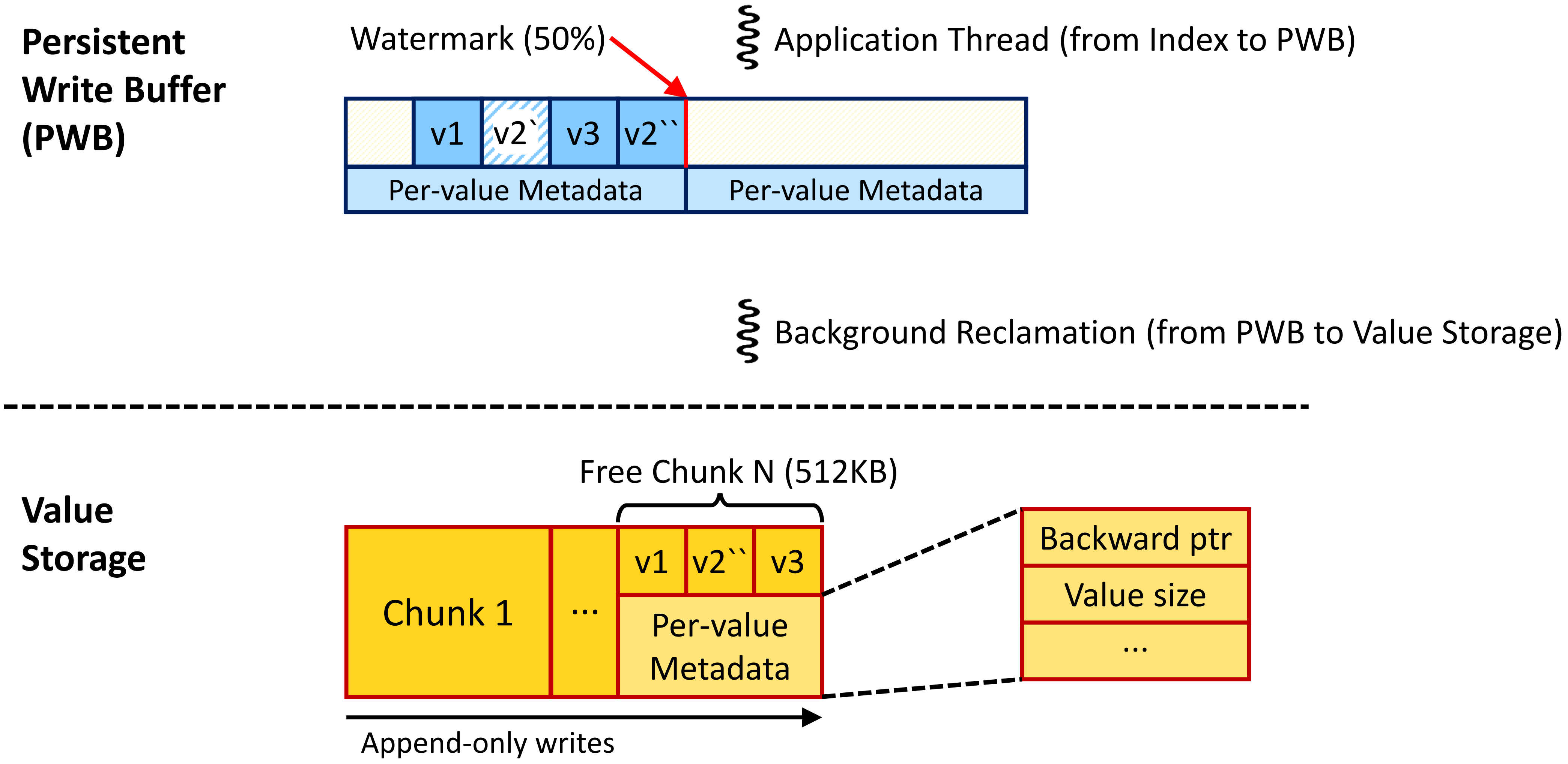
- Achieving high bandwidth of SSD



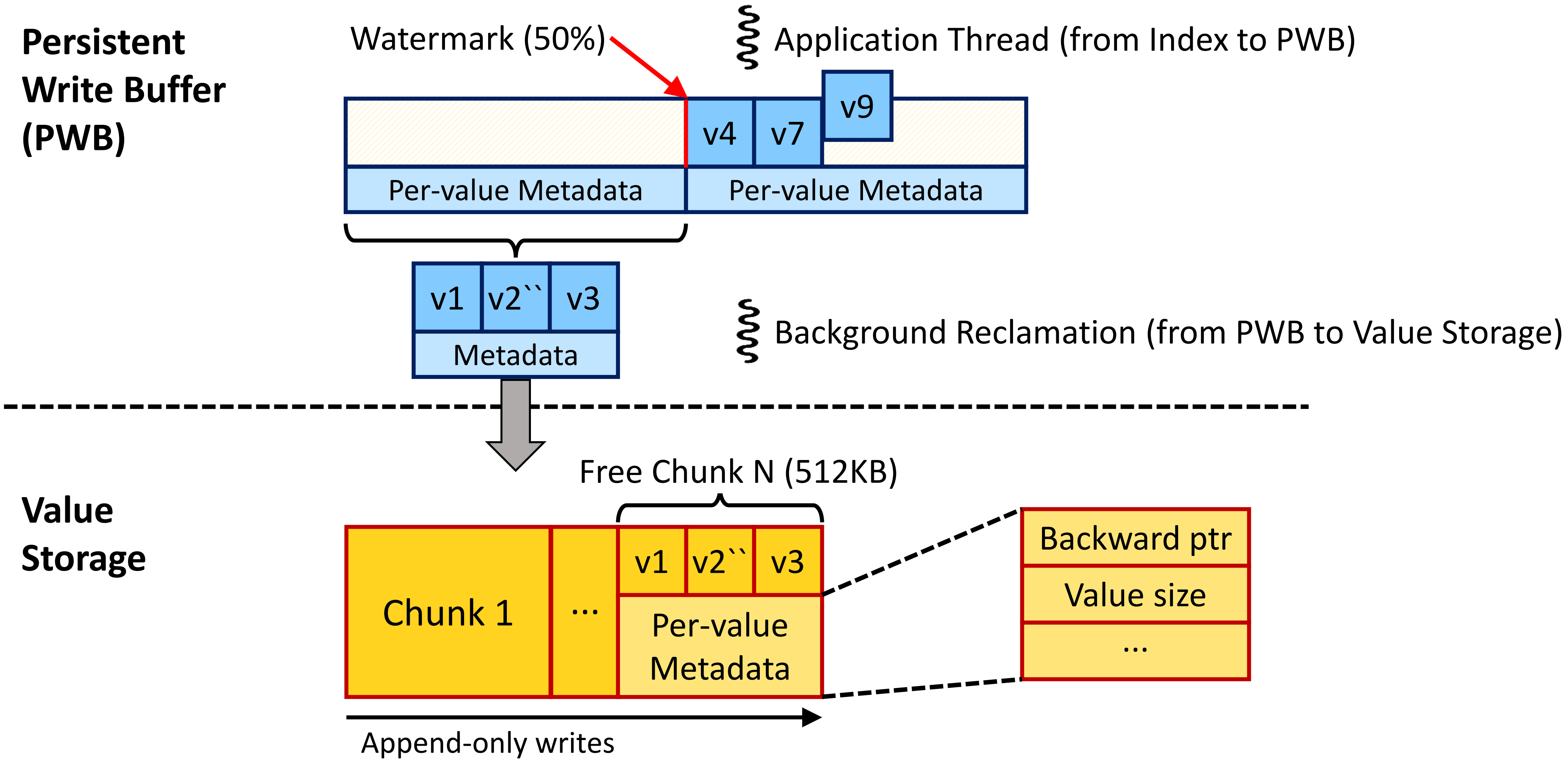
Asynchronous Bandwidth-Optimized WRITE

DRAM
NVM
SSD
Persistent Key Index
k1
Heterogeneous Storage Index Table (HSIT)
…
❷
❶
…
v1
…
❹
…
v4
v7
…
Persistent Write Buffer (PWB)
Value Storage

Asynchronous Bandwidth-Optimized WRITE



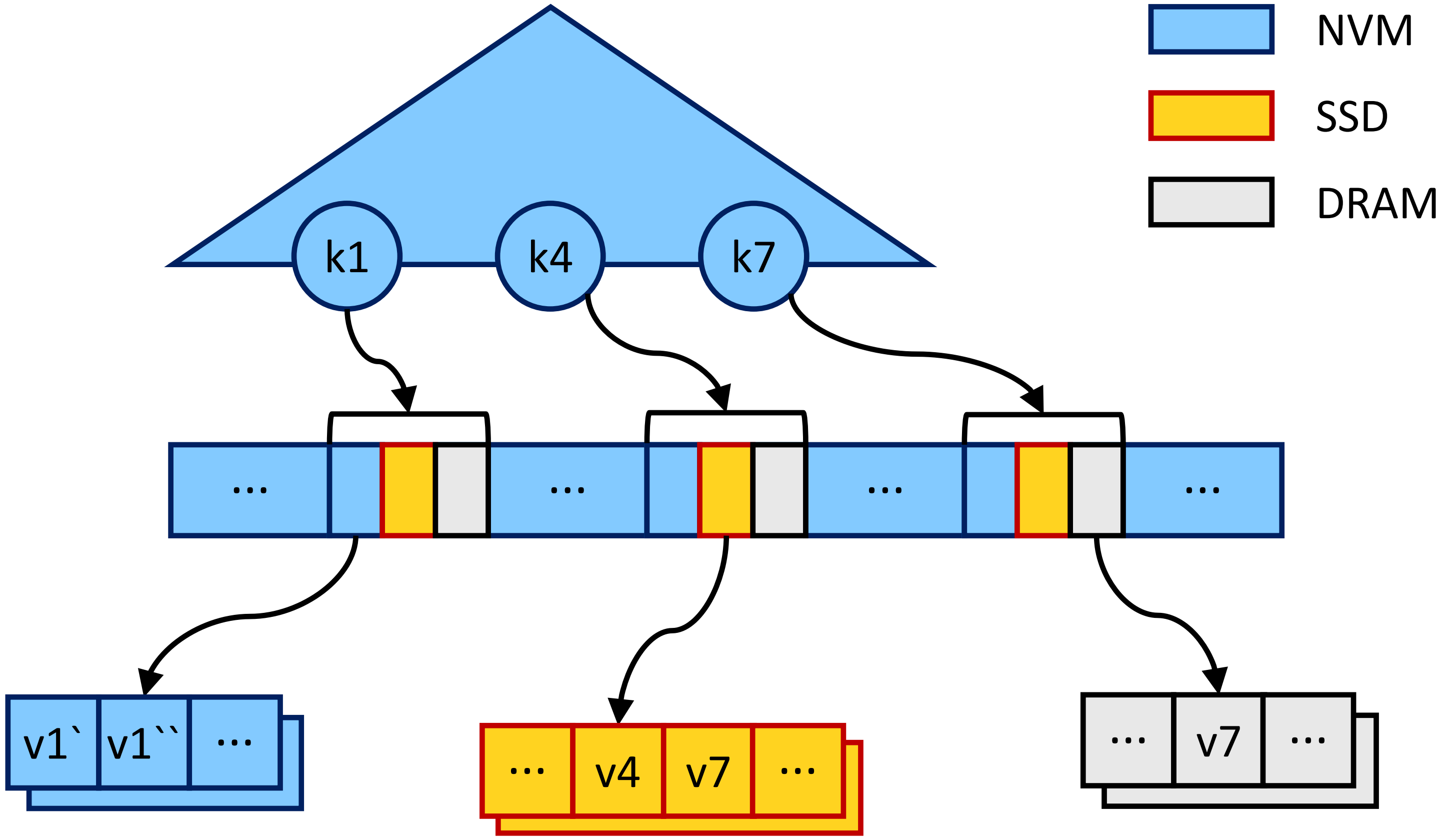
Asynchronous Bandwidth-Optimized WRITE



Design Overview of *PRISM*: Lookup(k4)

Persistent Key Index

Heterogeneous Storage Index Table (HSIT)



Persistent Write Buffer (PWB)

Value Storage

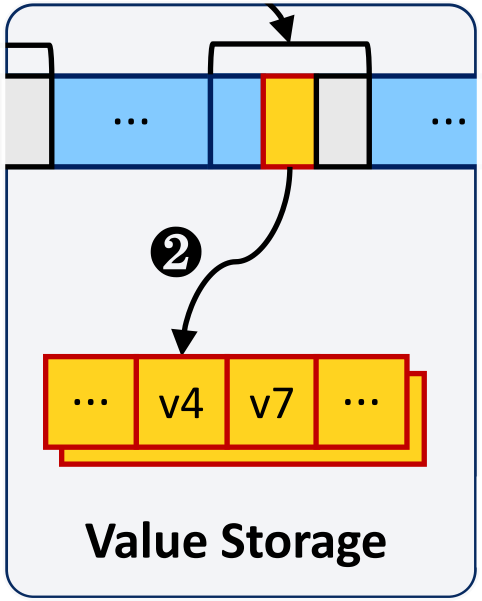
Scan-aware Value Cache (SVC)

Design Overview of *PRISM*: Lookup(k4)

Adjust the IO batch size for SSD reads according to thread concurrency

Combine reads from multiple threads to a single read operation

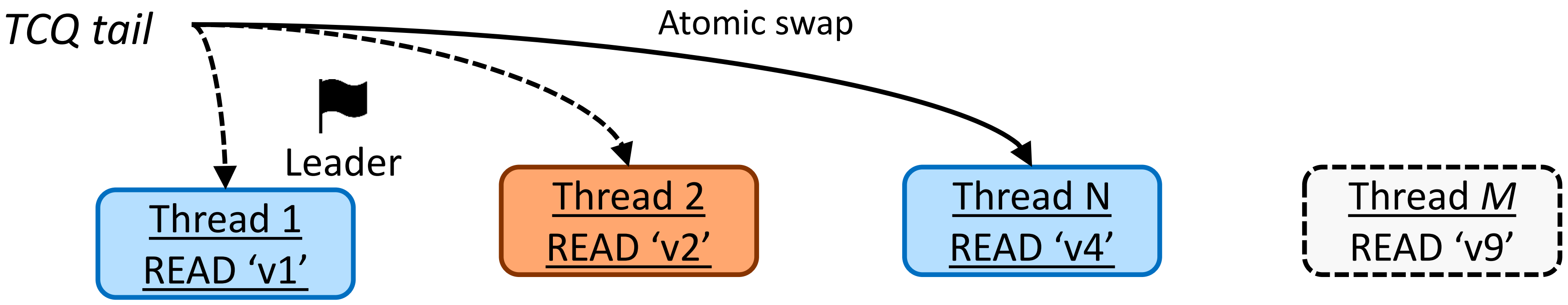
- Aggressively utilizing the bandwidth and hide latency of SSD



Opportunistic Thread Combining for READ

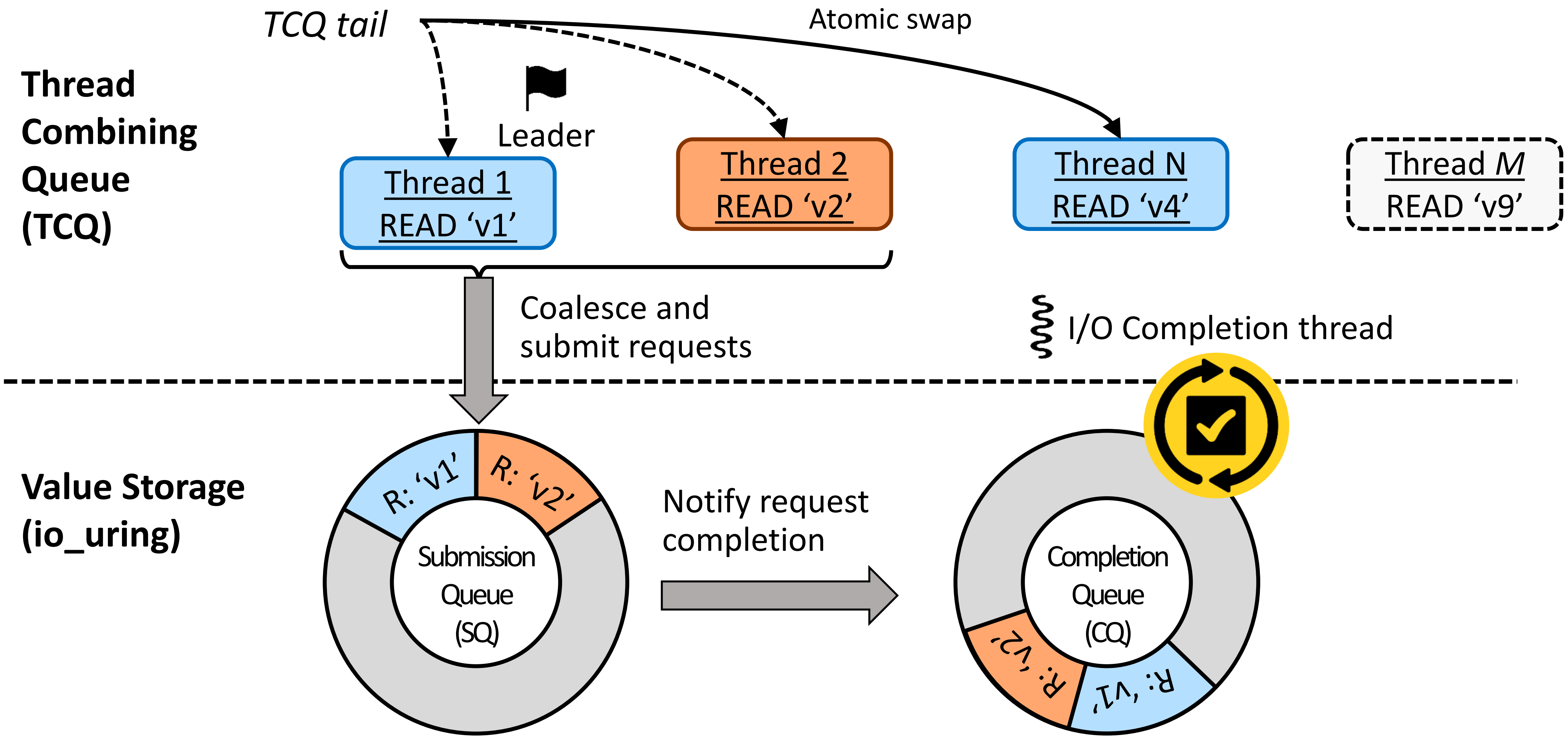
Opportunistic Thread Combining for READ

Thread Combining Queue (TCQ)

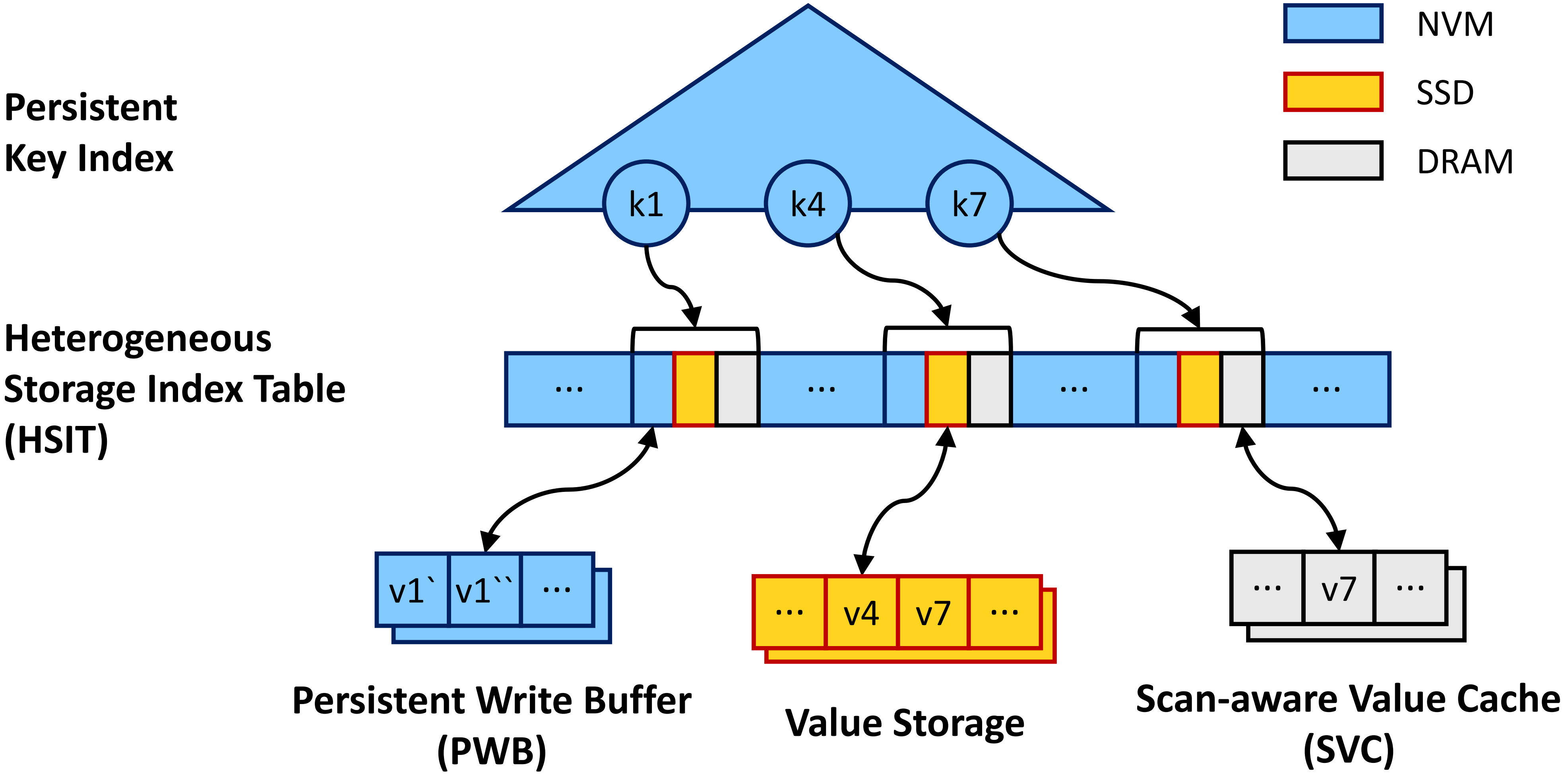


Value Storage (io_uring)

Opportunistic Thread Combining for READ



Cross-media Crash Consistency



🟦 NVM

🟨 SSD

⬜ DRAM

Persistent Key Index

Heterogeneous Storage Index Table (HSIT)

Persistent Write Buffer (PWB)

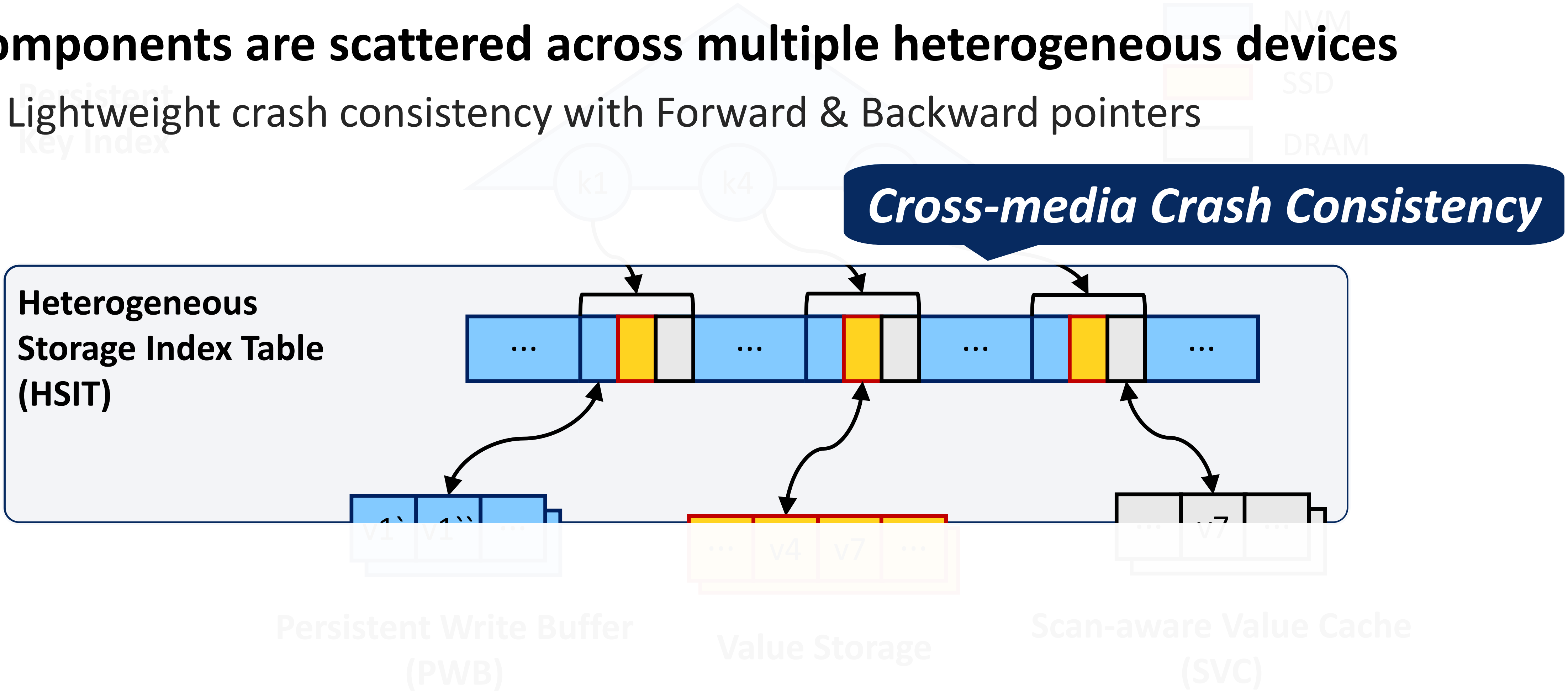
Value Storage

Scan-aware Value Cache (SVC)

Cross-media Crash Consistency

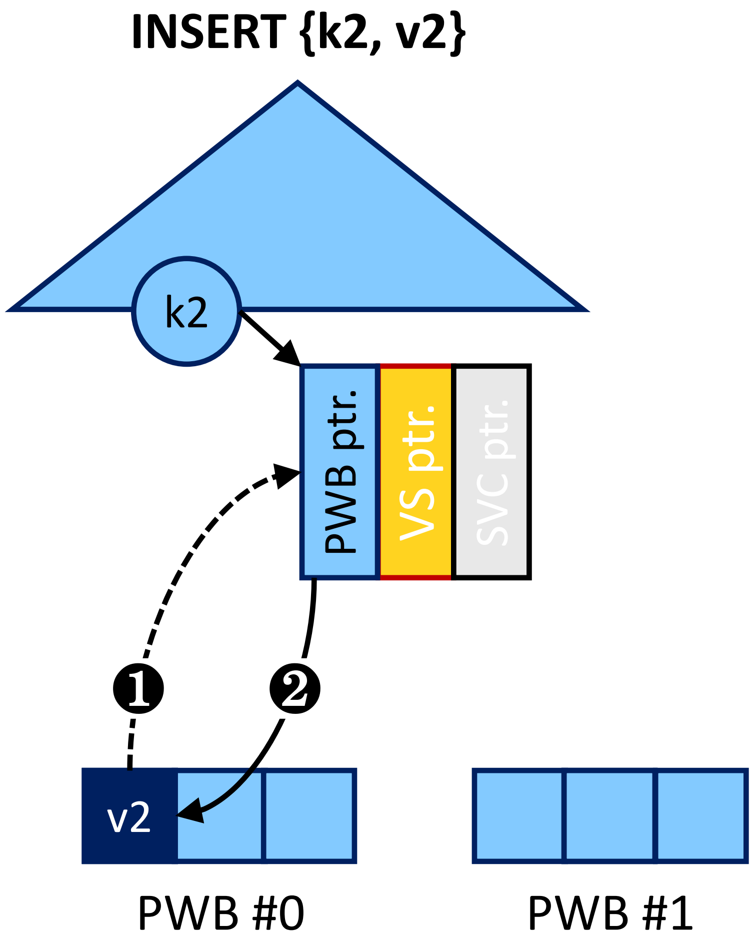
Components are scattered across multiple heterogeneous devices

- Lightweight crash consistency with Forward & Backward pointers



Cross-media Crash Consistency
Heterogeneous Storage Index Table (HSIT)
…
…
…
…
Persistent Key Index
k1
k4
NVM
SSD
DRAM
v1`
v1``
…
v4
v7
…
v7
Persistent Write Buffer (PWB)
Value Storage
Scan-aware Value Cache (SVC)

Crash Consistent Update of Values with HSIT



kN Leaf node of Persistent Key Index

PWB | VS | SVC HSIT entry

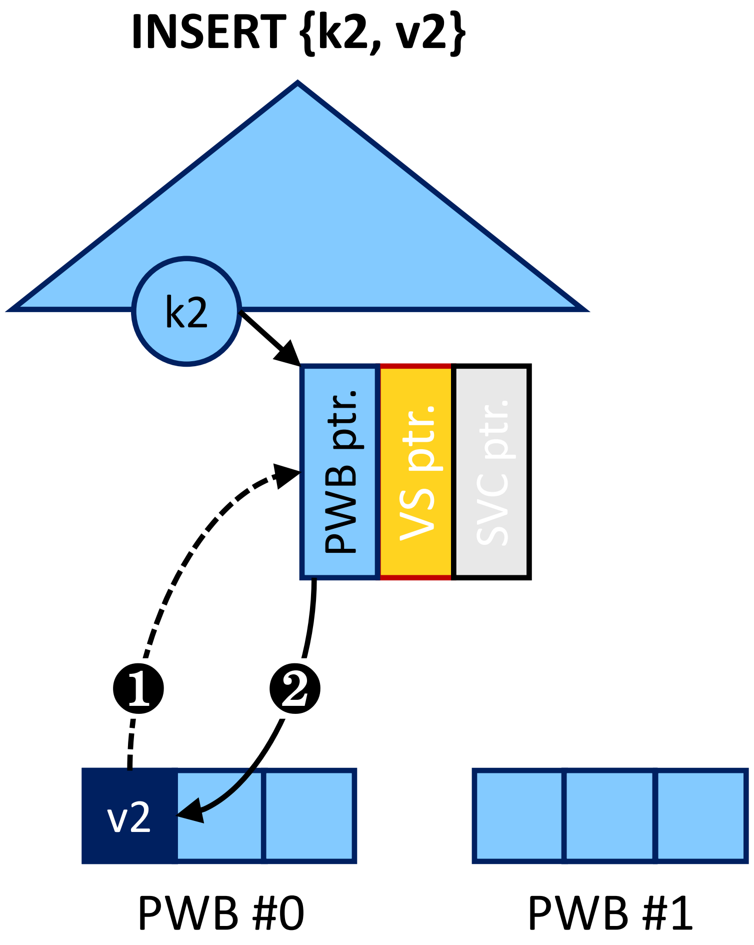
↓ Foward pointer

⇡ Backward pointer

▨ Obsolete value

■ Updated value

Crash Consistent Update of Values with HSIT



kN Leaf node of Persistent Key Index

PWB VS SVC HSIT entry

↓ Foward pointer

⇡ Backward pointer

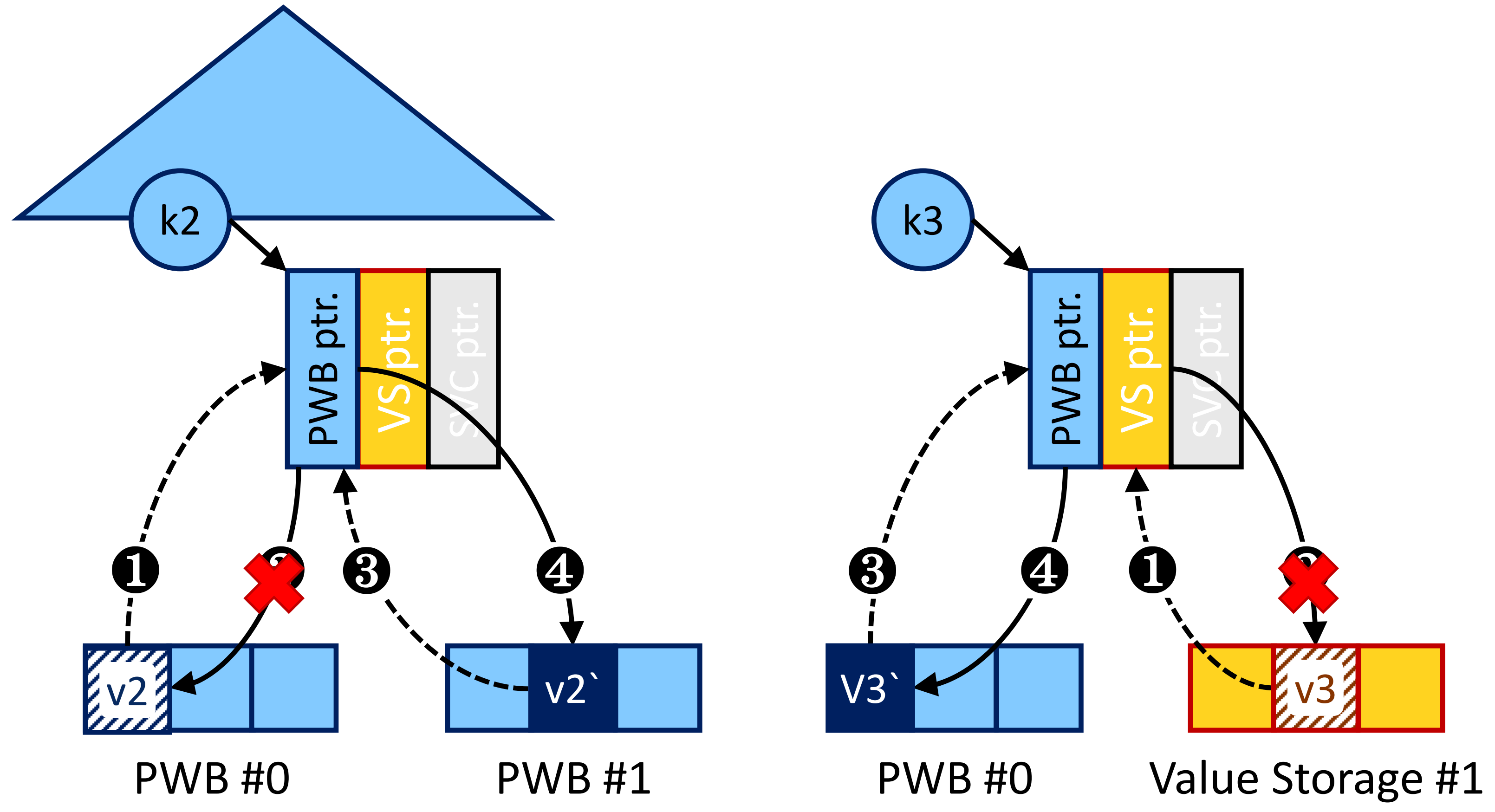
▨ Obsolete value

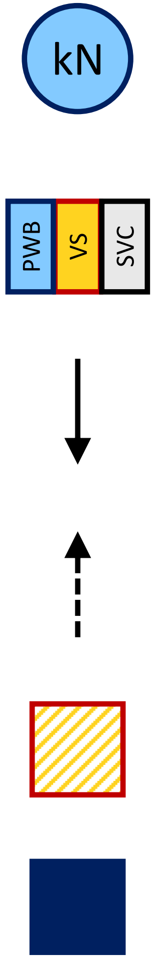
■ Updated value

Crash Consistent Update of Values with HSIT

UPDATE {k2, v2} to {k2, v2`}

UPDATE {k3, v3} to {k3, v3`}





Leaf node of Persistent Key Index

HSIT entry

Foward pointer

Backward pointer

Obsolete value

Updated value

Experimental Setup

Hardware environment

- Two-socket Intel Xeon machine
- Each socket: 20 CPU cores,
  Six 128GB Intel Optane DIMMSs, and 96GB DRAM
- **Eight Samsung 980 PRO 1TB SSDs**
  with two NVMe RAID Controllers HighPoint SSD7103

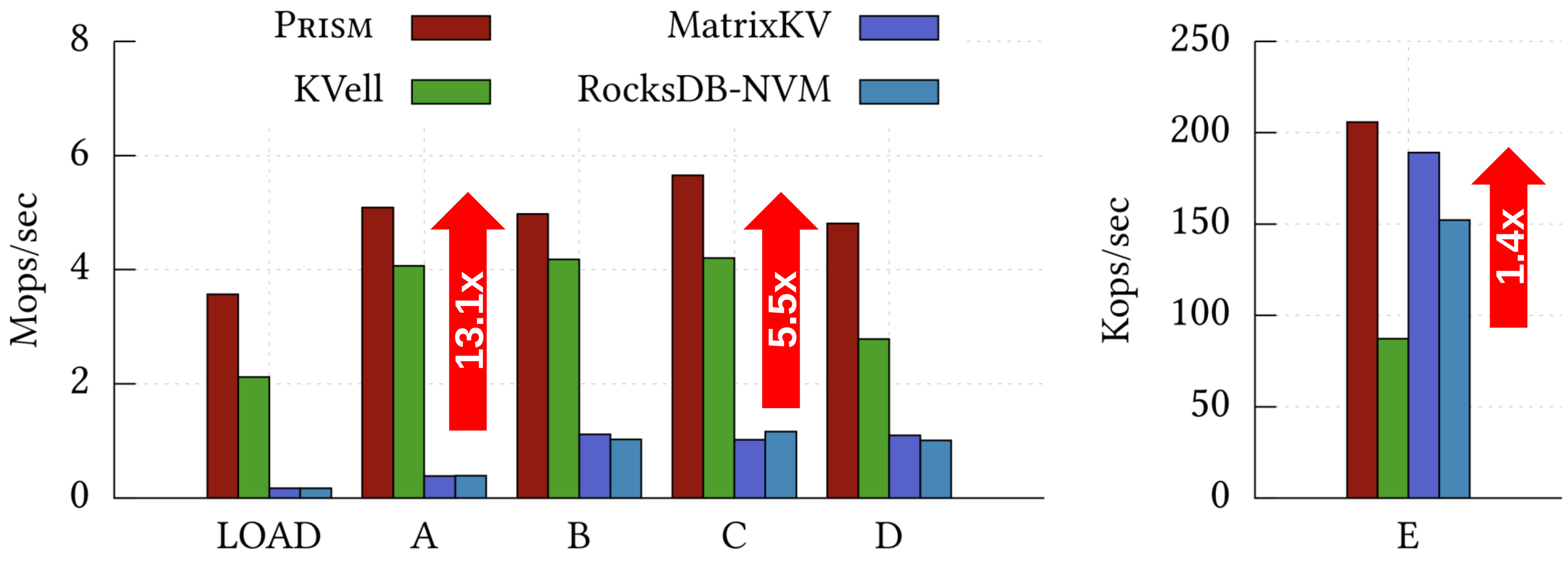
Competitors

- KVell: DRAM-SSD with up to 64 batched I/Os [SOSP'21]
- MatrixKV: DRAM-NVM-SSD [ATC'20]
- Allocated their hardware resources at the same cost levels

Performance Comparison on YCSB

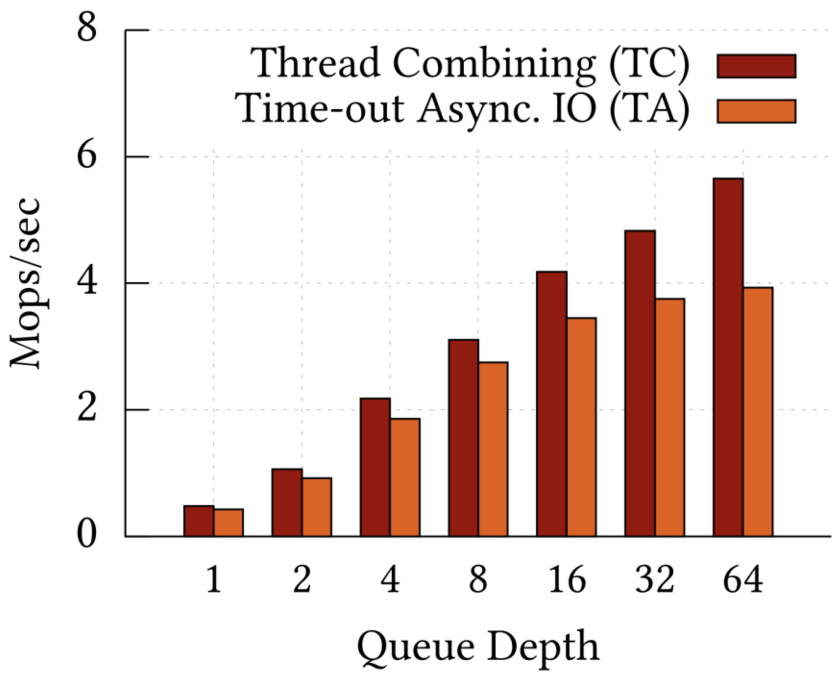
WRITE: Does not require level-compaction & Per-thread write buffer

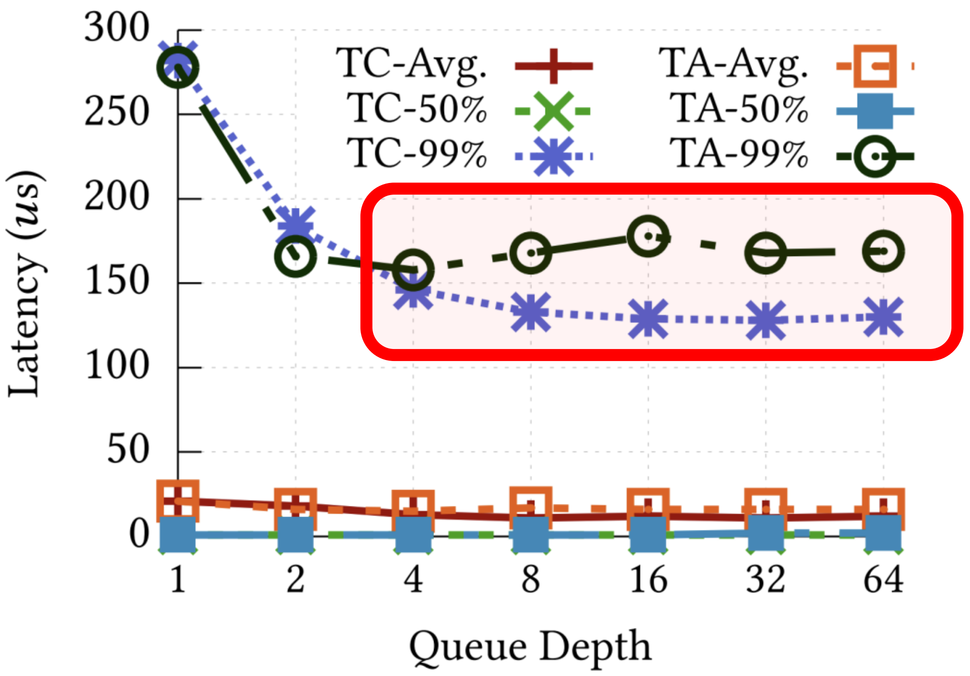
READ: No need for traversing multiple levels & Efficient KV item caching



Opportunistic Thread Combining

Prism opportunistically adjust the IO batch size
for read operations according to thread concurrency.





In the paper…

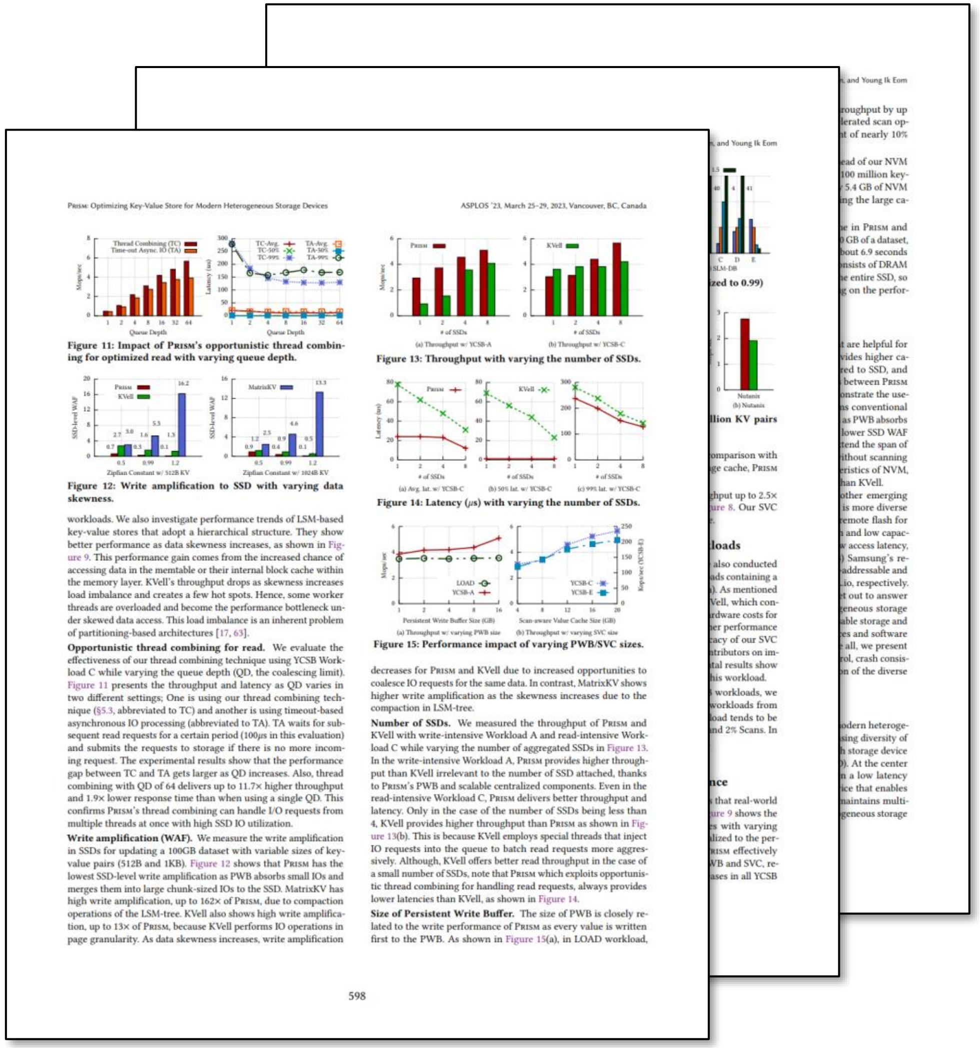
Performance under other workloads

Performance impact of ..

- Number of SSDs
- Size of PWB/SVC
- Write amplification
- Garbage collection in Value Storage
- Individual techniques

Size of NVM space

Recovery



Prism: Optimizing Key-Value Store for Modern Heterogeneous Storage Devices — ASPLOS '23, March 25–29, 2023, Vancouver, BC, Canada

Figure 11: Impact of Prism's opportunistic thread combining for optimized read with varying queue depth.

Figure 12: Write amplification to SSD with varying data skewness.

workloads. We also investigate performance trends of LSM-based key-value stores that adopt a hierarchical structure. They show better performance as data skewness increases, as shown in Figure 9. This performance gain comes from the increased chance of accessing data in the memtable or their internal block cache within the memory layer. KVell's throughput drops as skewness increases load imbalance and creates a few hot spots. Hence, some worker threads are overloaded and become the performance bottleneck under skewed data access. This load imbalance is an inherent problem of partitioning-based architectures [17, 63].

Opportunistic thread combining for read. We evaluate the effectiveness of our thread combining technique using YCSB Workload C while varying the queue depth (QD, the coalescing limit). Figure 11 presents the throughput and latency as QD varies in two different settings; One is using our thread combining technique (§5.3, abbreviated to TC) and another is using timeout-based asynchronous IO processing (abbreviated to TA). TA waits for subsequent read requests for a certain period (100μs in this evaluation) and submits the requests to storage if there is no more incoming request. The experimental results show that the performance gap between TC and TA gets larger as QD increases. Also, thread combining with QD of 64 delivers up to 11.7× higher throughput and 1.9× lower response time than when using a single QD. This confirms Prism's thread combining can handle I/O requests from multiple threads at once with high SSD IO utilization.

Write amplification (WAF). We measure the write amplification in SSDs for updating a 100GB dataset with variable sizes of key-value pairs (512B and 1KB). Figure 12 shows that Prism has the lowest SSD-level write amplification as PWB absorbs small IOs and merges them into large chunk-sized IOs to the SSD. MatrixKV has high write amplification, up to 162× of Prism, due to compaction operations of the LSM-tree. KVell also shows high write amplification, up to 13× of Prism, because KVell performs IO operations in page granularity. As data skewness increases, write amplification

(a) Throughput w/ YCSB-A

(b) Throughput w/ YCSB-C

Figure 13: Throughput with varying the number of SSDs.

(a) Avg. lat. w/ YCSB-C

(b) 50% lat. w/ YCSB-C

(c) 99% lat. w/ YCSB-C

Figure 14: Latency (μs) with varying the number of SSDs.

(a) Throughput w/ varying PWB size

(b) Throughput w/ varying SVC size

Figure 15: Performance impact of varying PWB/SVC sizes.

decreases for Prism and KVell due to increased opportunities to coalesce IO requests for the same data. In contrast, MatrixKV shows higher write amplification as the skewness increases due to the compaction in LSM-tree.

Number of SSDs. We measured the throughput of Prism and KVell with write-intensive Workload A and read-intensive Workload C while varying the number of aggregated SSDs in Figure 13. In the write-intensive Workload A, Prism provides higher throughput than KVell irrelevant to the number of SSD attached, thanks to Prism's PWB and scalable centralized components. Even in the read-intensive Workload C, Prism delivers better throughput and latency. Only in the case of the number of SSDs being less than 4, KVell provides higher throughput than Prism as shown in Figure 13(b). This is because KVell employs special threads that inject IO requests into the queue to batch read requests more aggressively. Although, KVell offers better read throughput in the case of a small number of SSDs, note that Prism which exploits opportunistic thread combining for handling read requests, always provides lower latencies than KVell, as shown in Figure 14.

Size of Persistent Write Buffer. The size of PWB is closely related to the write performance of Prism as every value is written first to the PWB. As shown in Figure 15(a), in LOAD workload,

598

Conclusion

We answered the question:

How should we design a Heterogeneous Storage System in the Modern Storage Landscape?

- Synergistic Five Components
- Asynchronous Bandwidth-Optimized WRITE
- Opportunistic Thread Combining
- Cross-media Crash Consistency & Concurrency Control using Forward & Backward Pointers

PRISM: Optimizing Key-Value Store for Modern Heterogeneous Storage Devices

Yongju Song, Wook-Hee Kim, Sumit Kumar Monga,

Changwoo Min, and Young Ik Eom

Paper