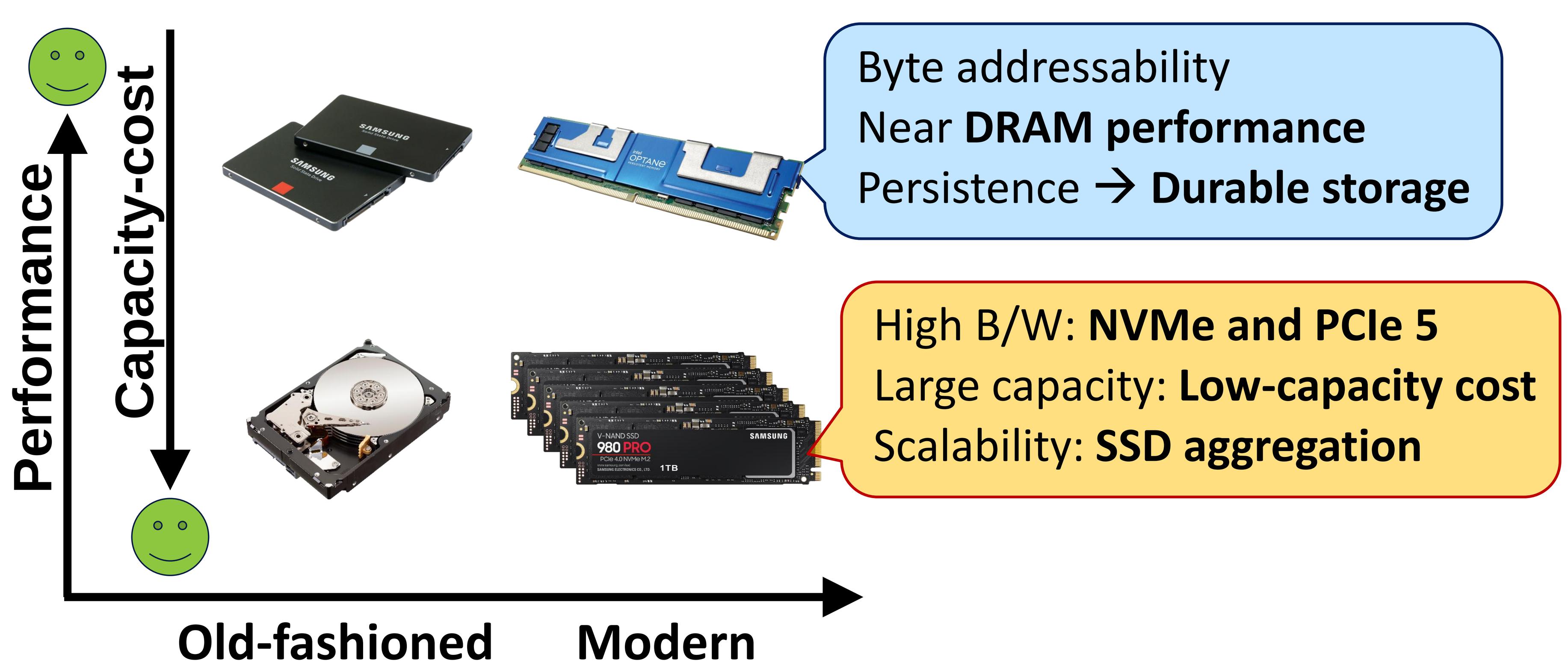
PRISM: Optimizing Key-Value Store for Modern Heterogeneous Storage Devices

Yongju Song, Wook-Hee Kim[1], Sumit Kumar Monga[2], Changwoo Min[2], and Young Ik Eom
Sungkyunkwan University, [1]Konkuk University, [2]Virginia Tech

Scan for Full Paper

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

1. Modern Heterogeneous Storage Devices



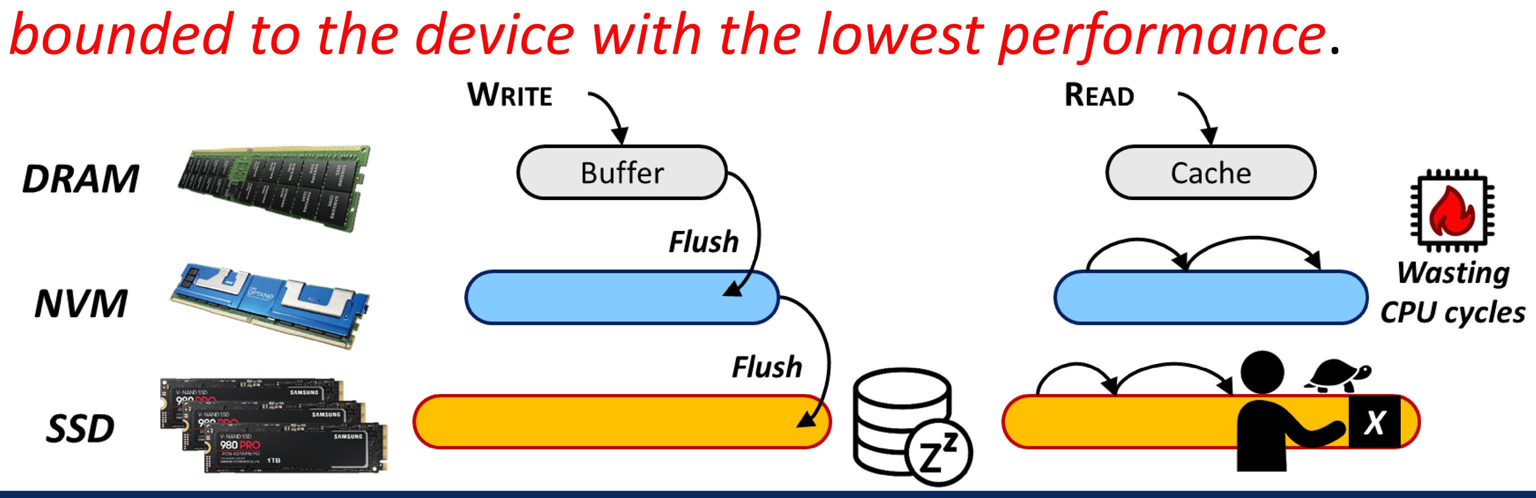
2. Evolution of Storage Heterogeneity

- **No clear separation between performance/capacity layers.**
  - ✓ "The Storage Hierarchy is *Becoming a Jungle*." [CIDR'21, Dong Xie]
  - ✓ "The Storage Hierarchy is *Not a Hierarchy*." [FAST'21, Remzi H. Arpaci-Dusseau]

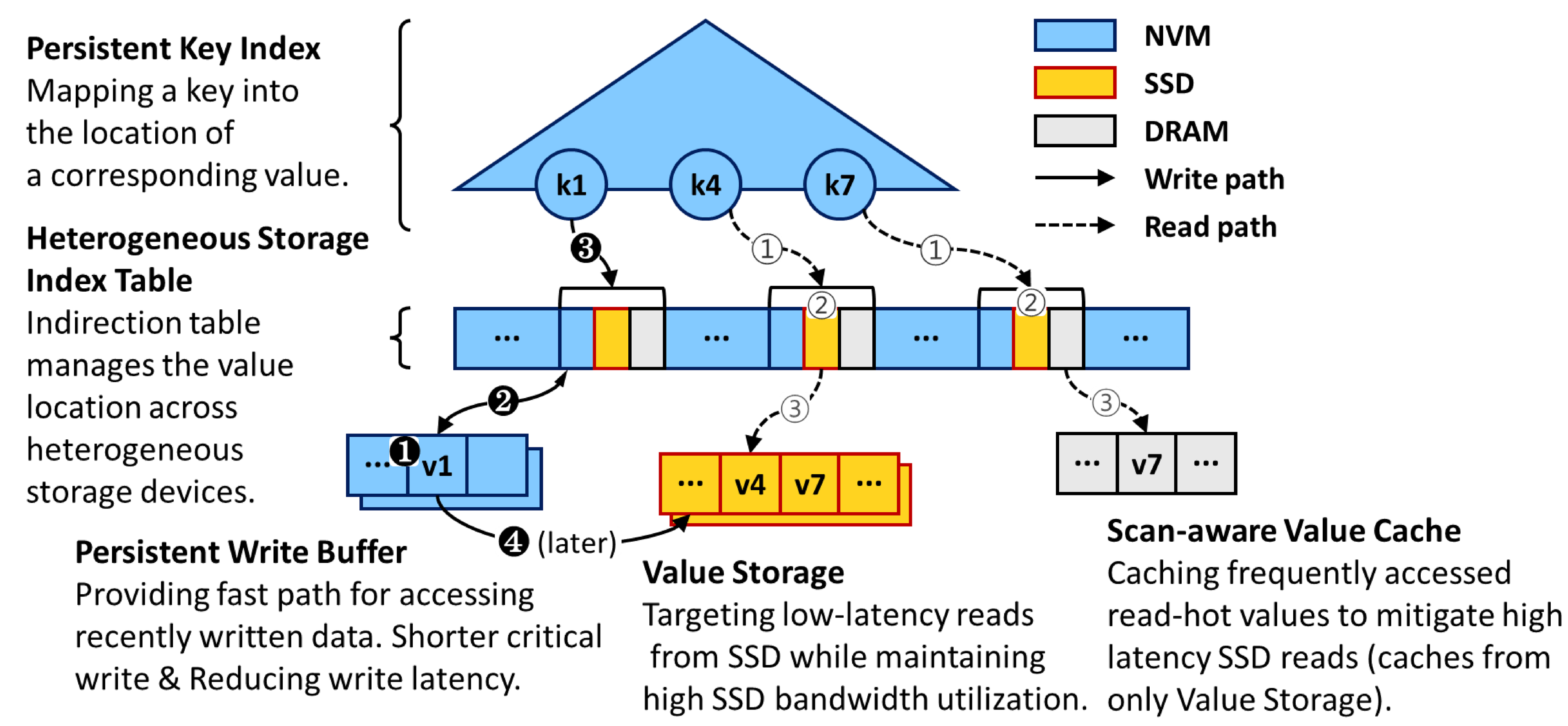
Model	GB	$/TB	Performance (µs, GB/s)				Endurance
			Read Latency	Write Latency	Read BW	Write BW	Warranty (PBW)
SK Hynix DRAM	16	5,427	0.08	0.08	15	15	∞
Intel Optane DCPMM	128	4,096	**0.30**	**0.09**	6.8	1.9	**292**
Intel Optane 905P PCIe 3	960	400	10	10	2.6	2.2	17.52
Samsung 980 Pro PCIe 4	**1,024**	**100**	50	20	**7**	**5**	0.6
Samsung 980 PCIe 3	**1,024**	**70**	60	20	3.5	3	0.6

3. Managing Storage Heterogeneity Today

- **Placing hot data on NVM**
  - ✓ System can leverage the low latency of NVM but *suffer from NVM's 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*.

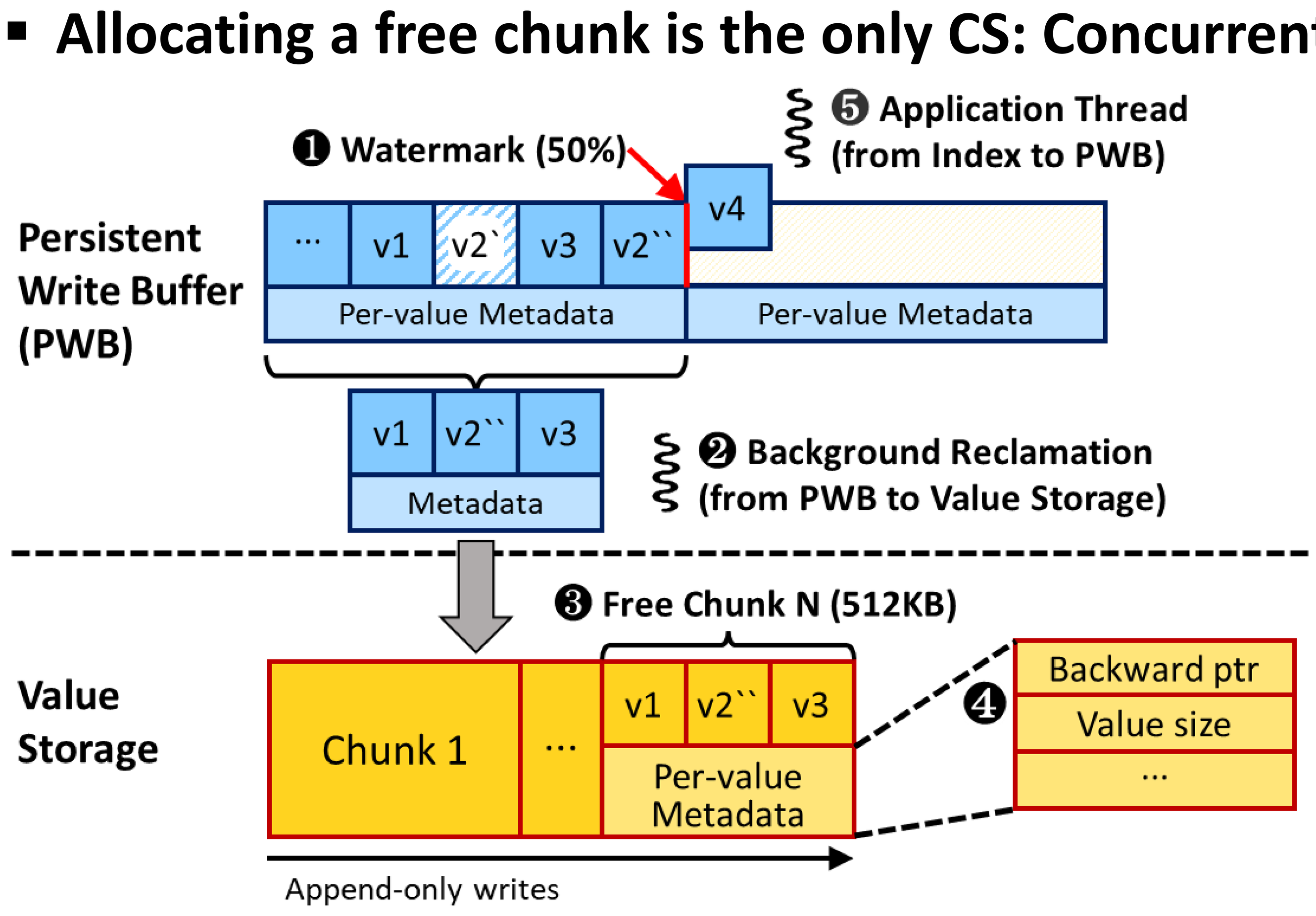


4. Design Overview of PRISM



5. Asynchronous Bandwidth-Optimized Write

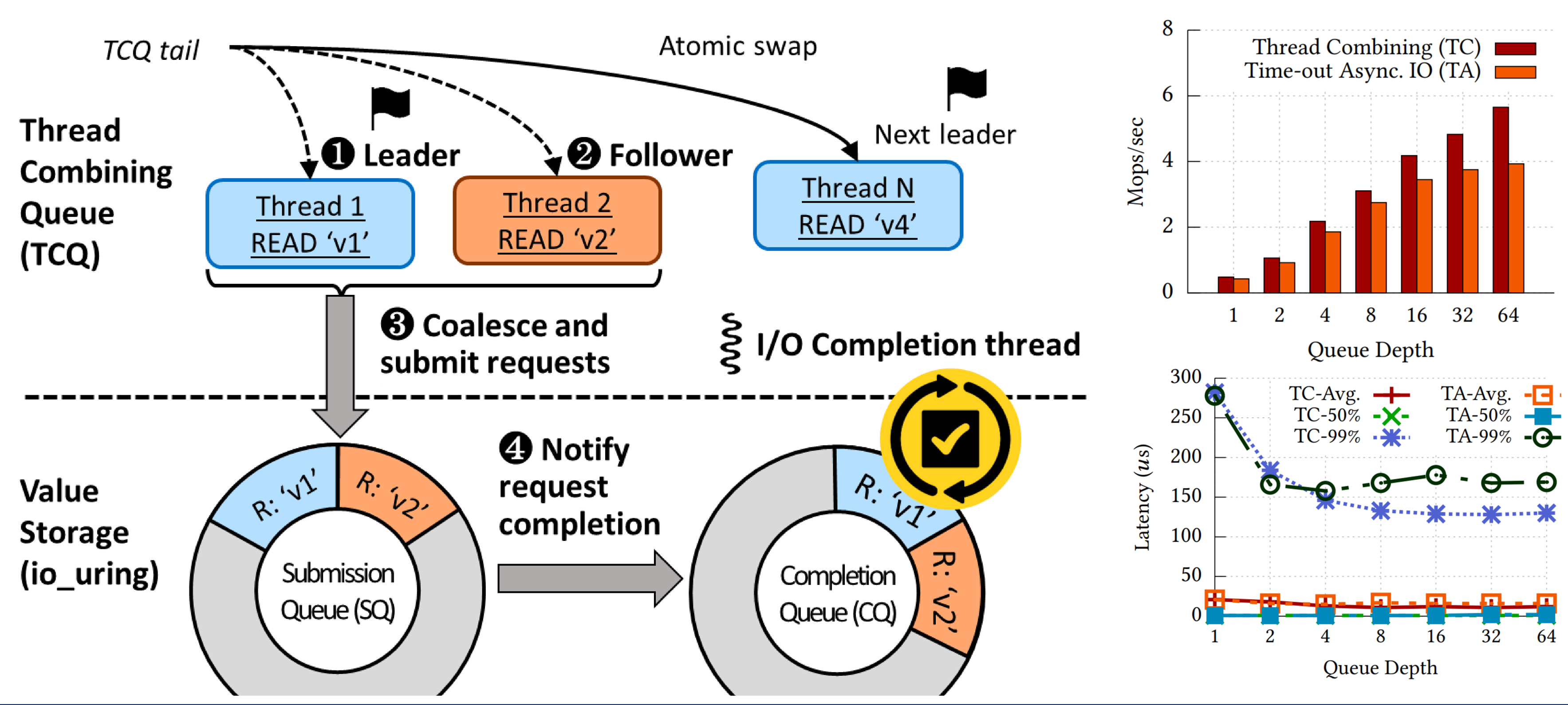
- **Background reclamation : Preventing App. Thread from blocking**
- **Asynchronous IO batching: Achieving high bandwidth of SSDs**
- **Allocating a free chunk is the only CS: Concurrent Writes**



❶ PWB utilization hits the watermark(50%).
❷ the up-to-date values (v1, v2``, v3) in the PWB
❸ are written to Value Storage asynchronously.
❹ Each chunk is fixed-size and stores values with their metadata (ptr. to an HSIT entry and value size)
❺ While reclaiming PWB, an application thread can still write values (v4) to the rest of the PWB

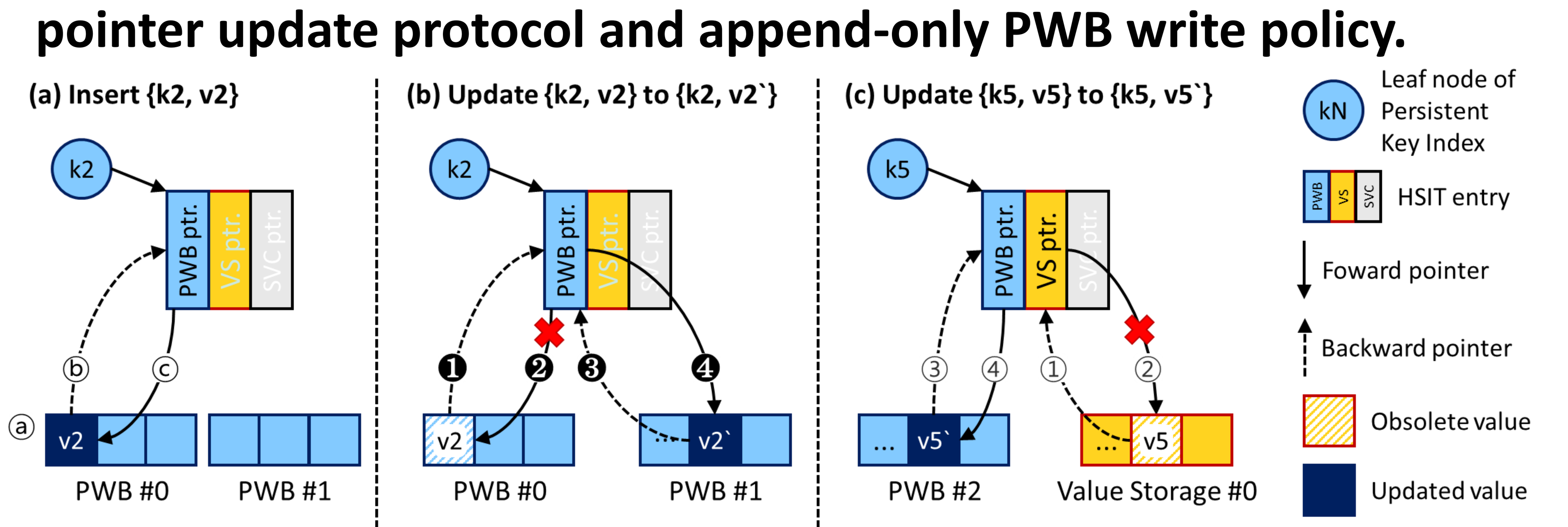
6. Opportunistic Thread Combining Read

- **Dynamically determining the right IO batch size** (High B/W & Low Lat.)
  - ✓ Leader dynamically coalesces read req. of followers and submit
  - ✓ Conditions for submitting requests: (prerequisite: Value Storage is idle).
    1. No more followers or 2. When leader thread reaches limit QD.

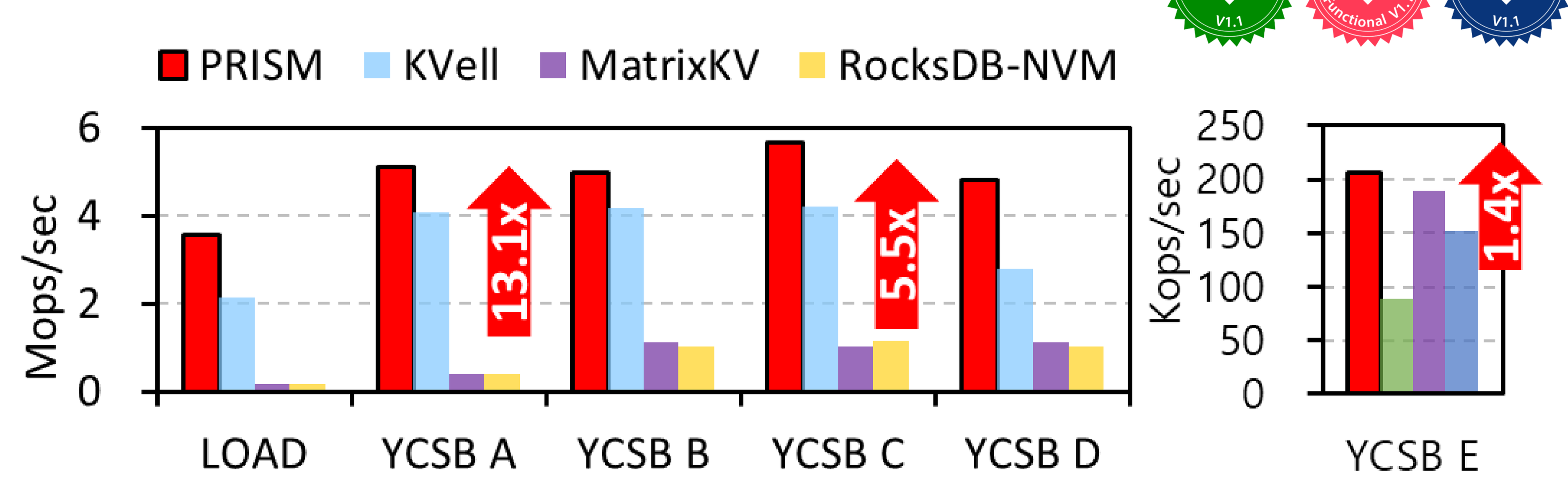


7. Cross-media Crash Consistency

- **Crash consistent update of values on PWB & Value Storage w/ HSIT.**
  - ✓ First writes a value with a backward pointer (❸, ③), and then it updates its forward pointer (❹, ④), invalidating the old forward pointer (❷, ②).
- **Efficiently guarantees cross-media crash consistency with our pointer update protocol and append-only PWB write policy.**



8. Evaluation



- **Other interesting evaluations and in-depth analysis**
  - ✓ Multicore Scalability, Data Skewness, and Write Amplification
  - ✓ Performance Impact from various system configurations
  - ✓ More details for understanding *PRISM* performance
- **Available in GitHub at https://github.com/cosmoss-jigu/prism**

For further information, please contact Yongju Song at yongju@skku.edu