PolyStore: Exploiting Combined Capabilities of Heterogeneous Storage

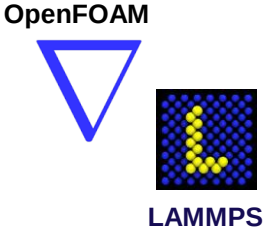
Yujie Ren David Domingo Jian Zhang Paul John Rekha Pitchumani

Sanidhya Kashyap Sudarsun Kannan

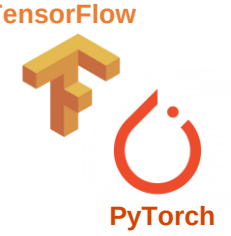
Applications demand storage capacity and bandwidth

Photo/video editing

Single-node databases



Scientific computing



Machine learning

Storage b/w need

~ 100 MB/s **~ 1 GB/s** **up to 10 GB/s ***

2005 2015 2025

https://dl.acm.org/doi/10.1109/SC.2018.00068

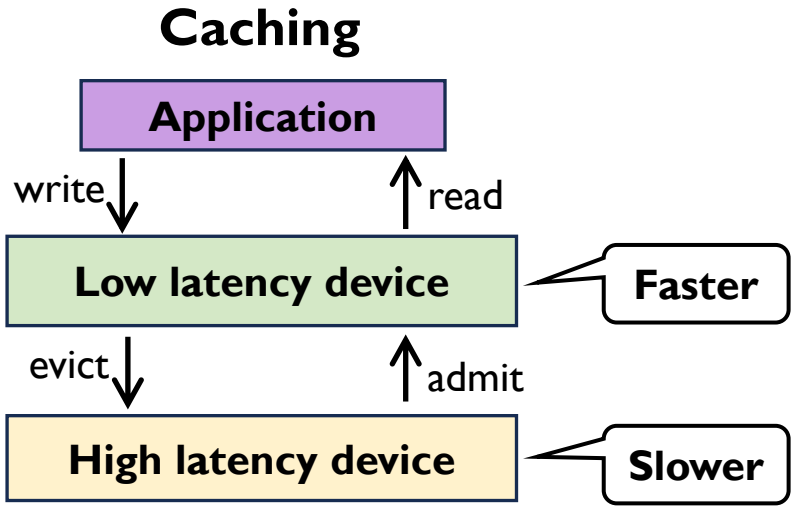
Evolving storage media with specialized file systems

	NVMe SSD	Persistent memory	Ultra-fast NVMe SSD
Storage hardware	 $	 $$ $$	 $$
2015 → 2025			
Read b/w	1.2 GB/s	13.6 GB/s	6.9 GB/s
Write b/w	1.3 GB/s	4.6 GB/s	7.4 GB/s
Latency	50 µs	300 ns	10 µs
Specialized file systems	SAMSUNG F2FS	NV SL NOVA	Ext4 XFS

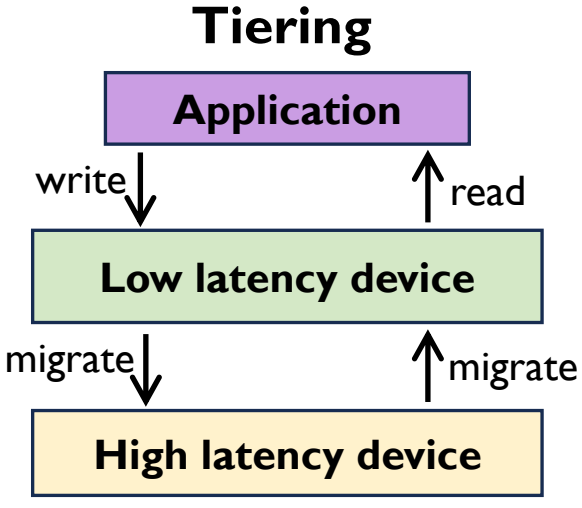
Use them collaboratively to reduce cost and retain performance!

Hiding latency with hierarchical design philosophy

Storage devices present diverse characteristics in latency, capacity, and price



Prompt admission upon cache miss

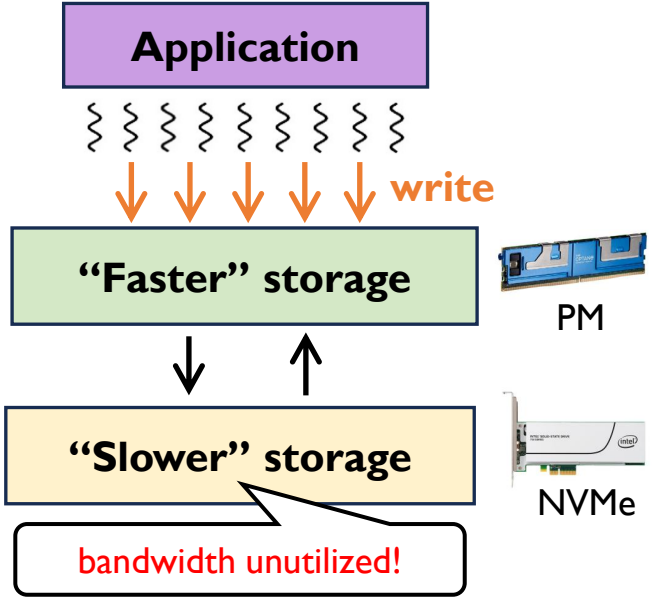


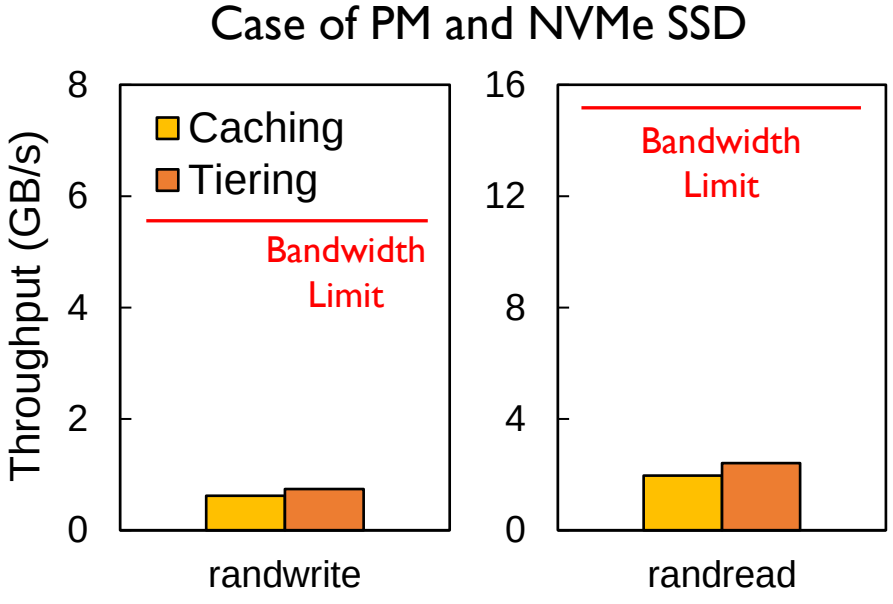
Coarse-grained data movement

"Faster" storage absorbs write requests of new data

Pitfalls in existing heterogeneous storage systems

Pitfall 1: Cannot utilize combined bandwidth of heterogeneous storage



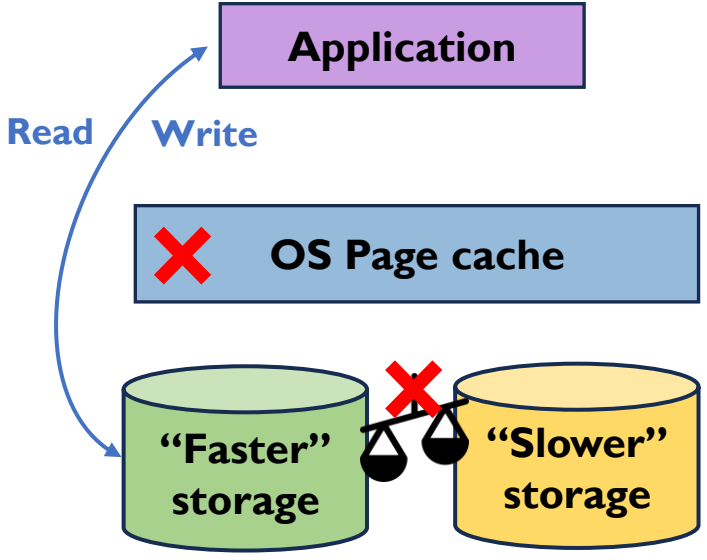


Hierarchical device layout limits cumulative bandwidth utilization!

Pitfalls in existing heterogeneous storage systems

Pitfall 2: Cannot provide device-specific DRAM buffering policies

- Hide latency on top of storage devices
- Bypassed for some devices by default
- No flexible admission and eviction control

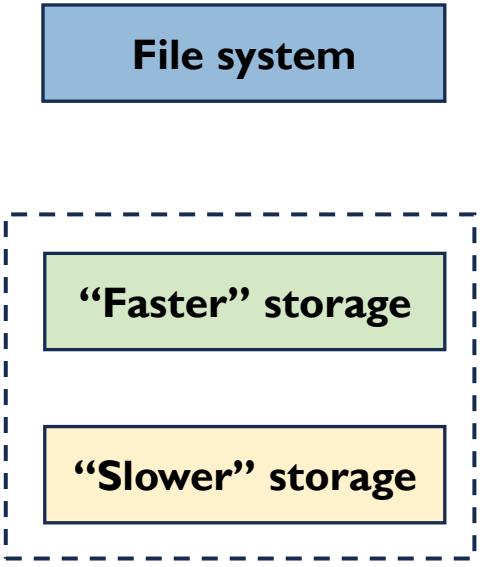


Lack mechanisms for applying device-specific policies for DRAM buffering!

Pitfalls in existing heterogeneous storage systems

Pitfall 3: Cannot capitalize on mature device-optimized file systems

- New tiered file systems for certain devices
  Strata [SOSP' 17], **Ziggurat** [FAST' 19]
- Managing in block layer with one file system
  Orthus [FAST' 21], **Bcache** [Linux]
- Leaving mature device-optimized FSes underutilized
  NOVA [FAST' 16], **F2FS** [FAST' 15], **ext4, XFS** [Linux]



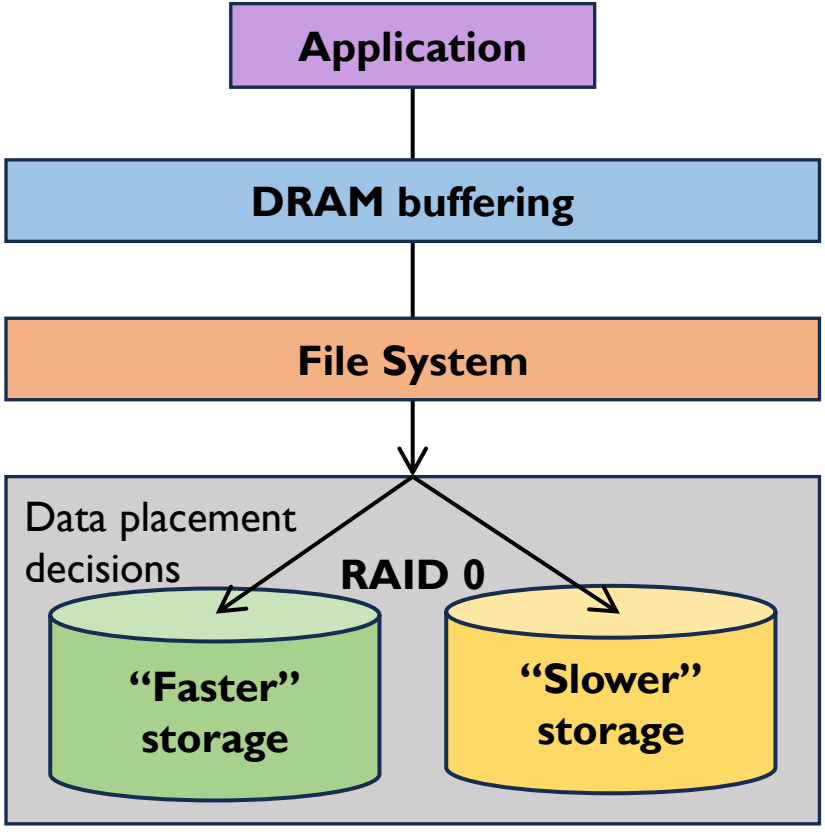
Coupling data placement decisions within the software stack!

Naïve solution with RAID 0 for heterogeneous devices?

✓ Combined bandwidth utilization

✗ Cannot capitalize on device-optimized file systems!

✗ Cannot provide device-specific DRAM buffer cache policies!



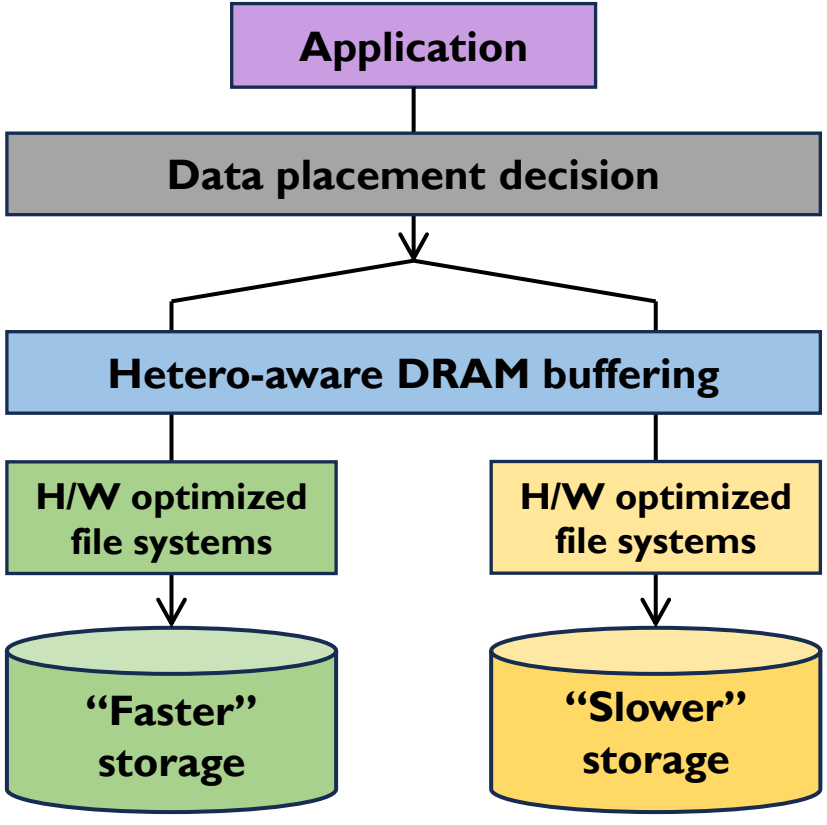
Ideal system for managing heterogeneous storage

✓ Combined bandwidth utilization

- Data placement decisions above the storage software stack

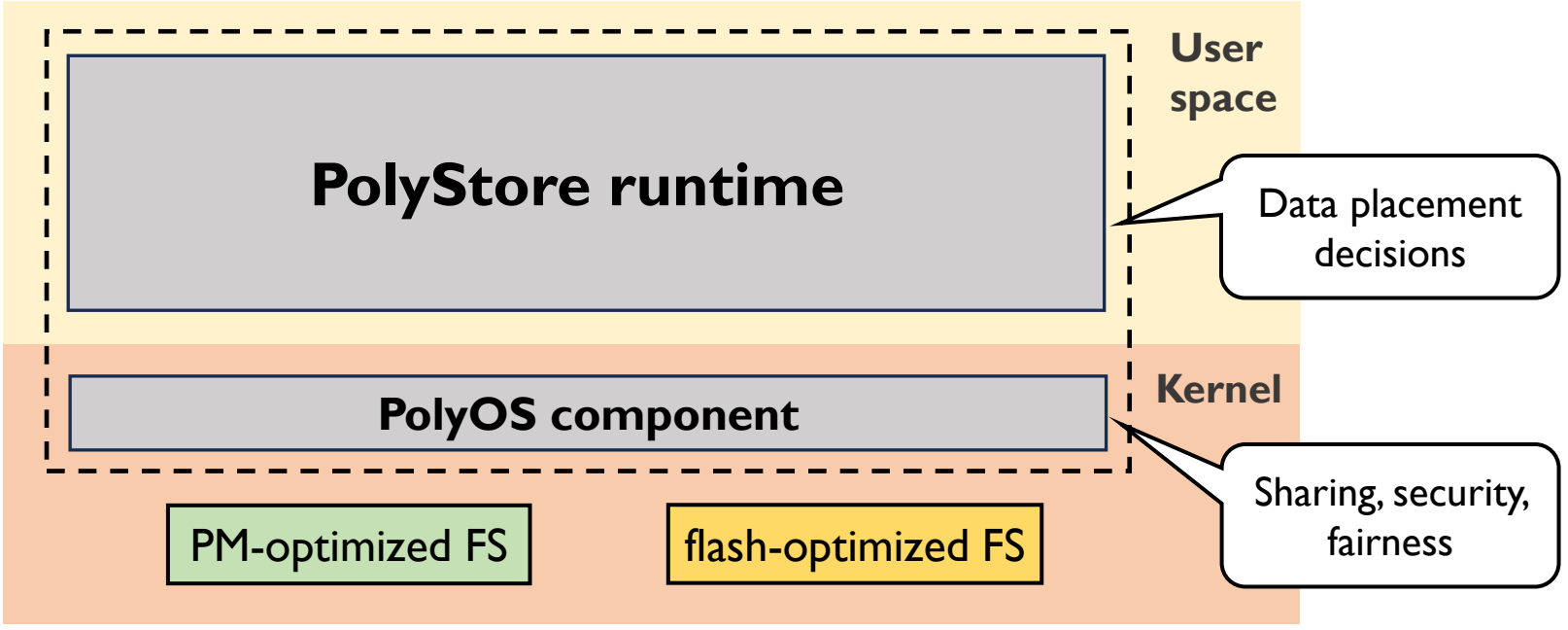
✓ Reuse mature file systems tailored for storage devices

✓ Flexible DRAM buffer cache admission/eviction



Our solution - PolyStore

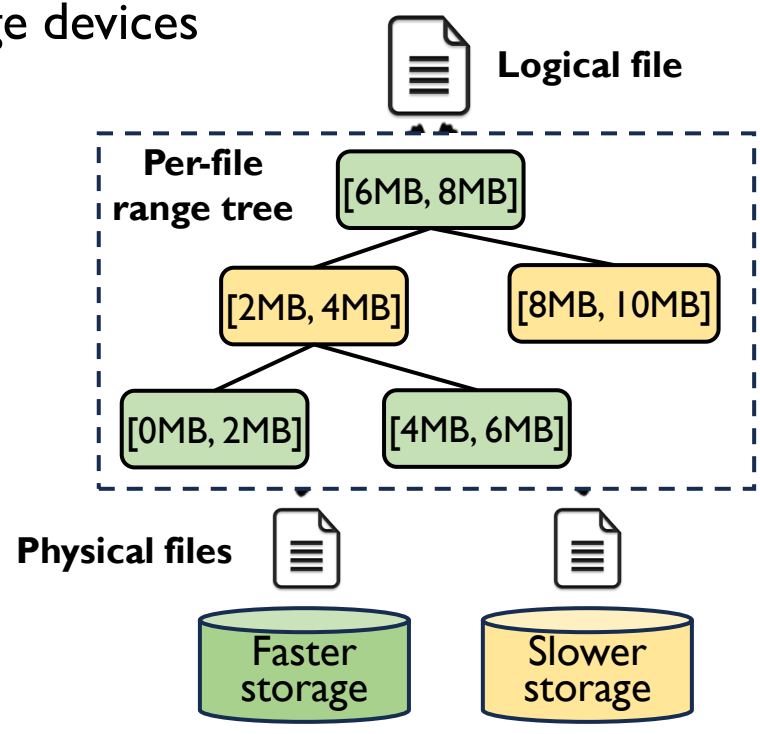
A meta-layer on top of device-optimized kernel-level file systems



Indexing data across device-optimized file systems

Distribute data across heterogeneous storage devices

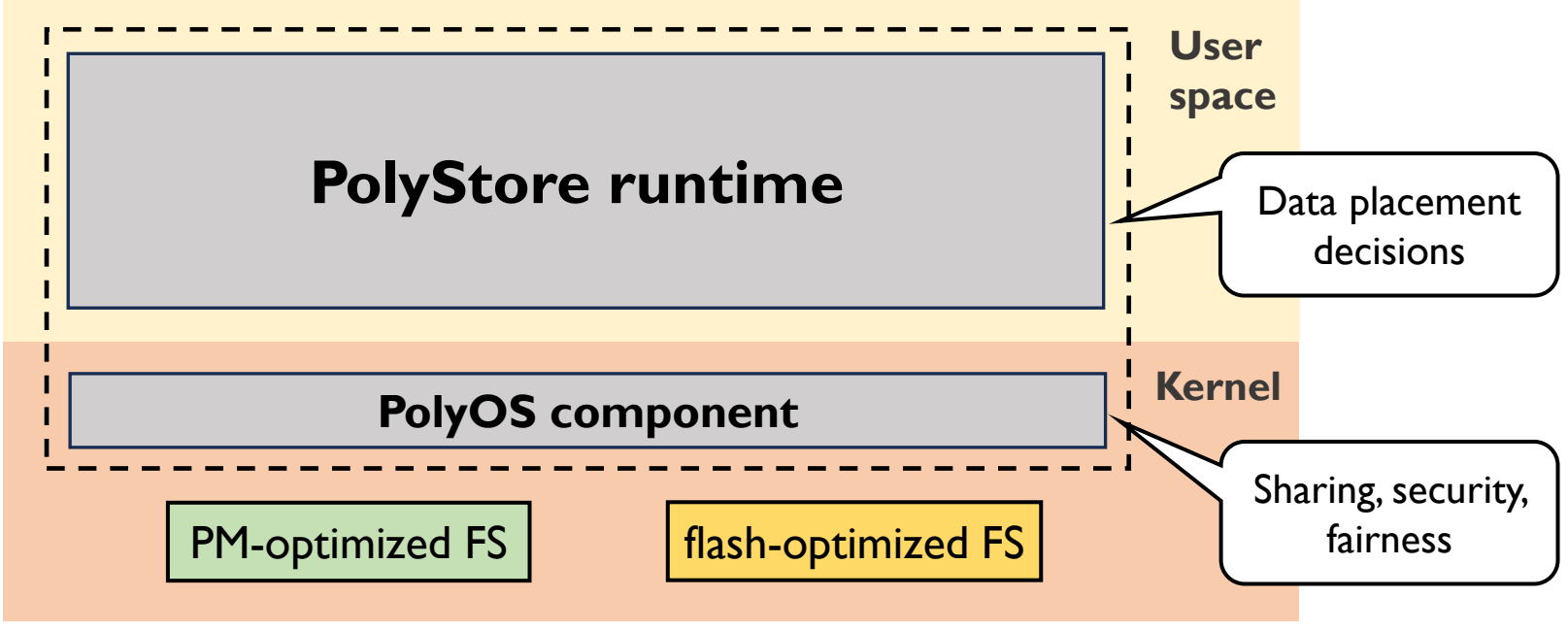
- Large files (e.g., streaming files) are bandwidth intensive
- Map a logical file to physical files with a scalable indexing structure
- Encode data placement and access patterns across devices



Key data structure for bandwidth utilization and flexible DRAM buffer cache!

Our solution - PolyStore

A meta-layer on top of device-optimized kernel-level file systems

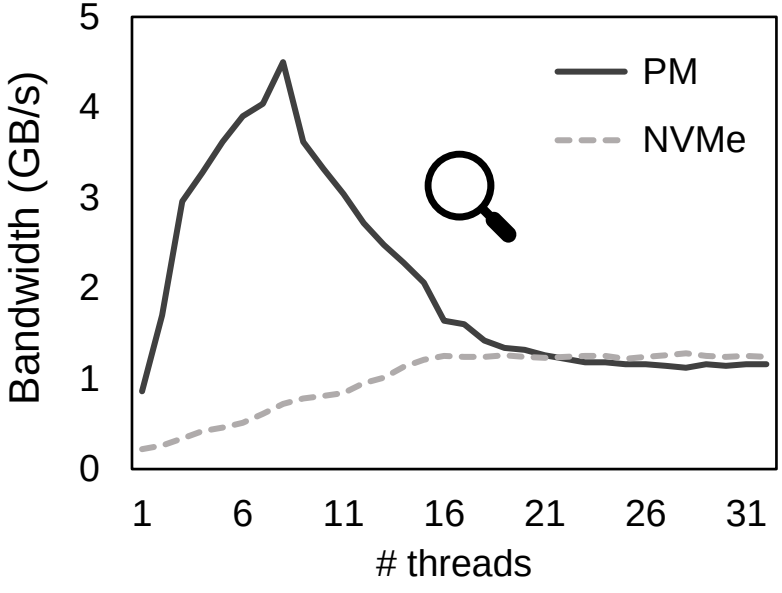


Dynamic data placement mechanism for combined bandwidth

Attaining combined bandwidth for dynamic workloads

Challenge - mapping I/O from application threads to storage devices

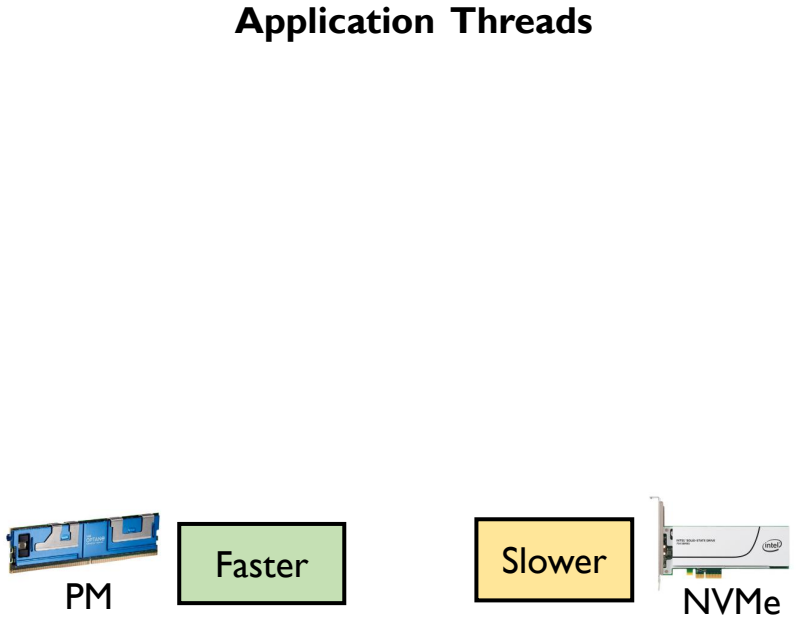
- Storage bandwidths show diverse characteristics
- Statically identifying the optimal mapping cannot adapt to workload changes in applications

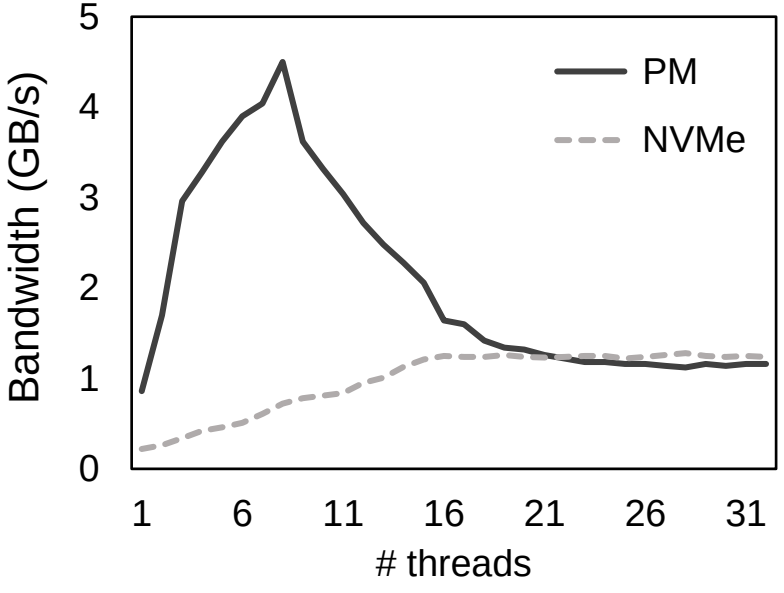


Need dynamic data placement to maximize the storage bandwidth utilization!

Attaining combined bandwidth for dynamic workloads

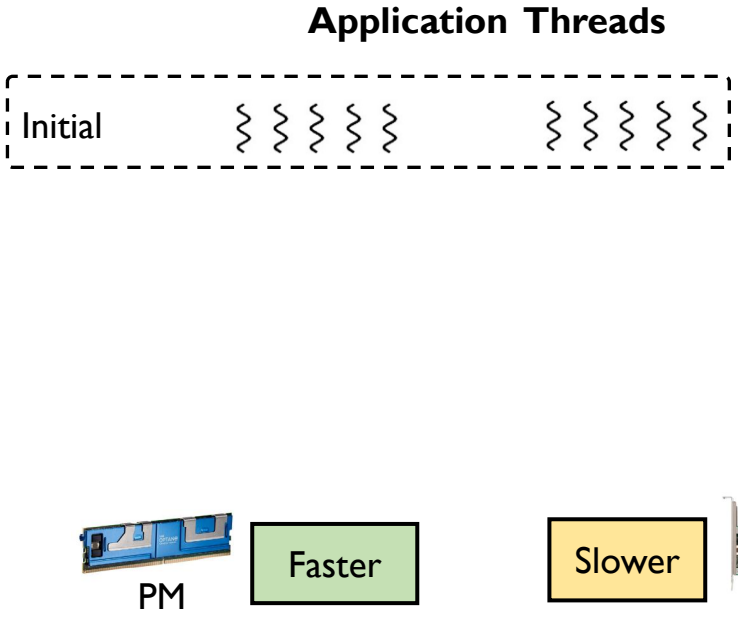
Epoch-based throughput monitoring and dynamic data placement

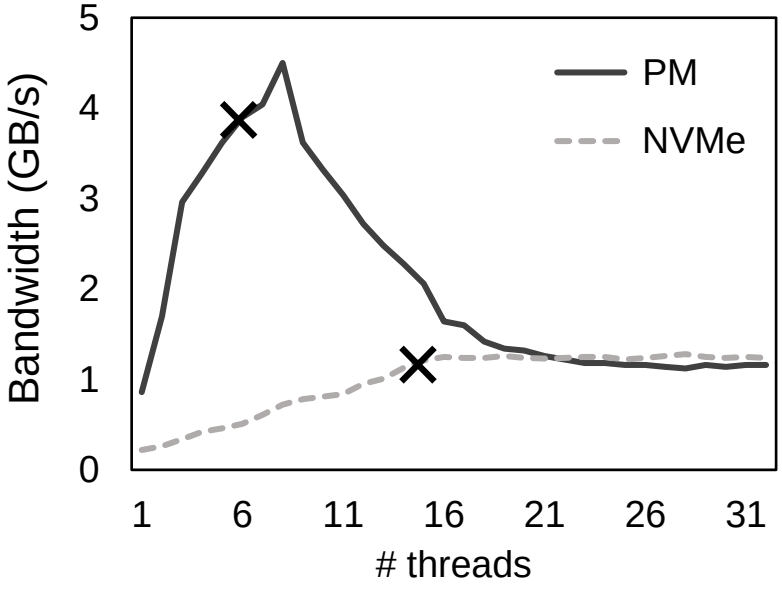




Attaining combined bandwidth for dynamic workloads

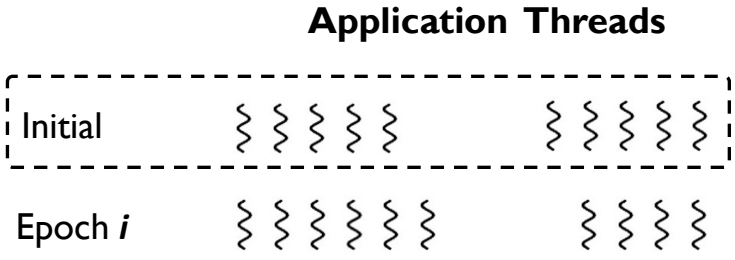
Epoch-based throughput monitoring and dynamic data placement





Attaining combined bandwidth for dynamic workloads

Epoch-based throughput monitoring and dynamic data placement

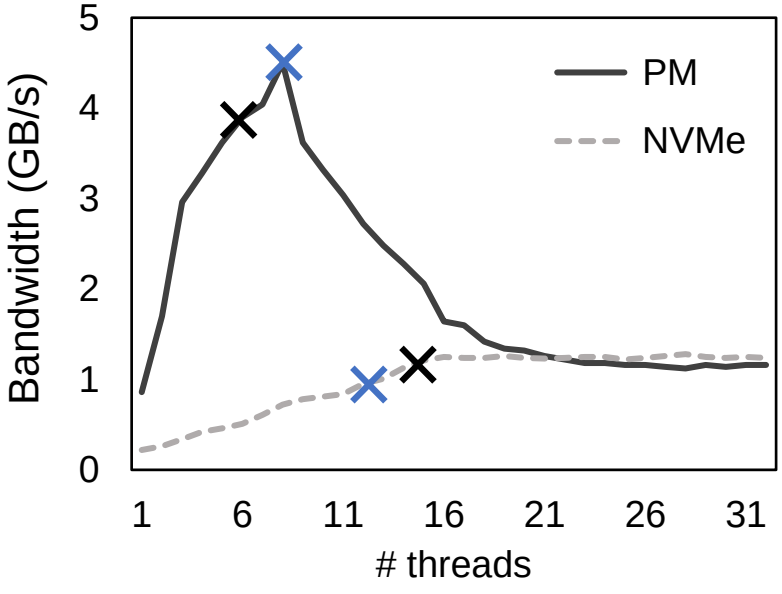


PM

Faster

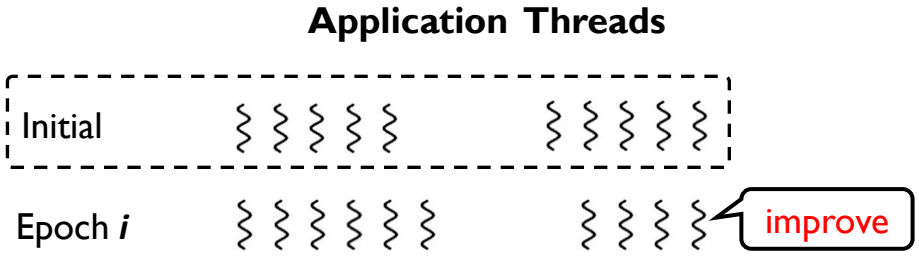
Slower

NVMe



Attaining combined bandwidth for dynamic workloads

Epoch-based throughput monitoring and dynamic data placement

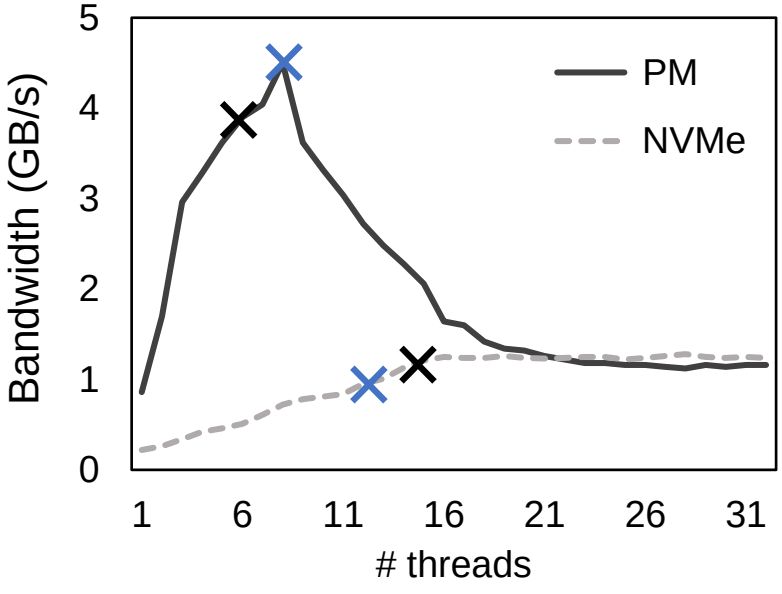


PM

Faster

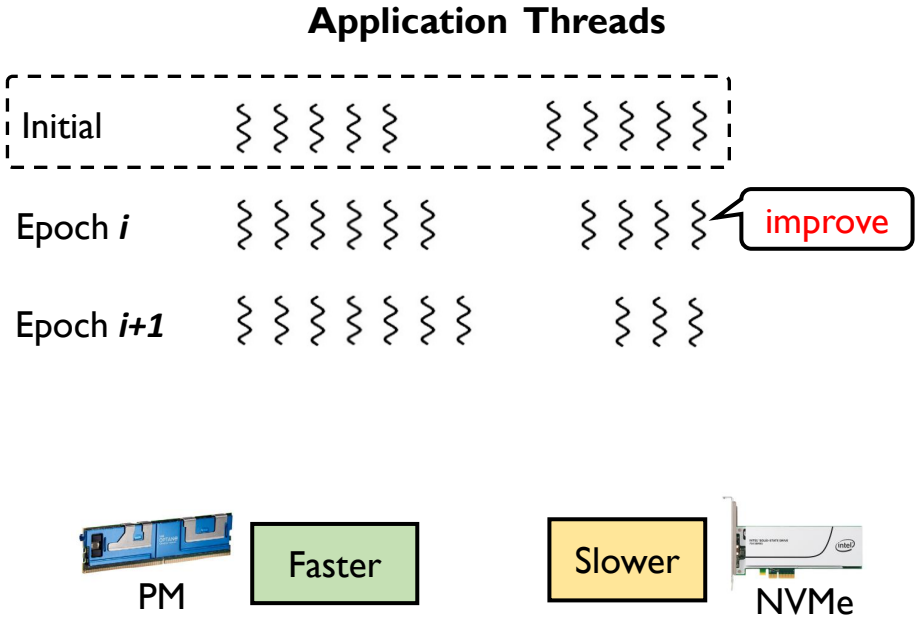
Slower

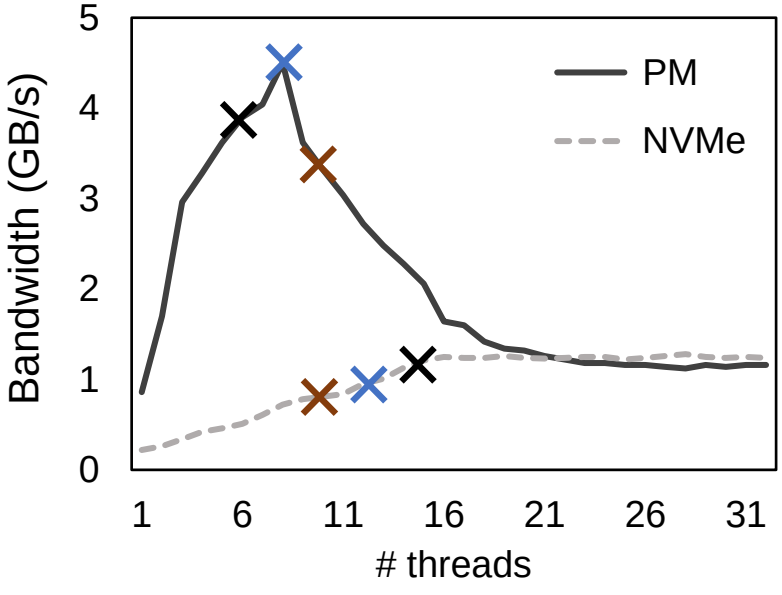
NVMe



Attaining combined bandwidth for dynamic workloads

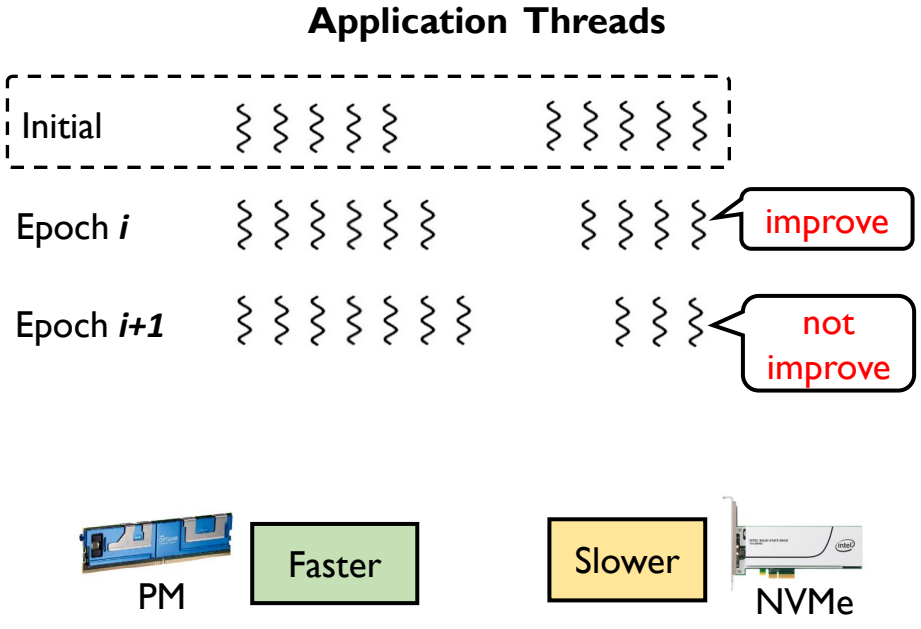
Epoch-based throughput monitoring and dynamic data placement

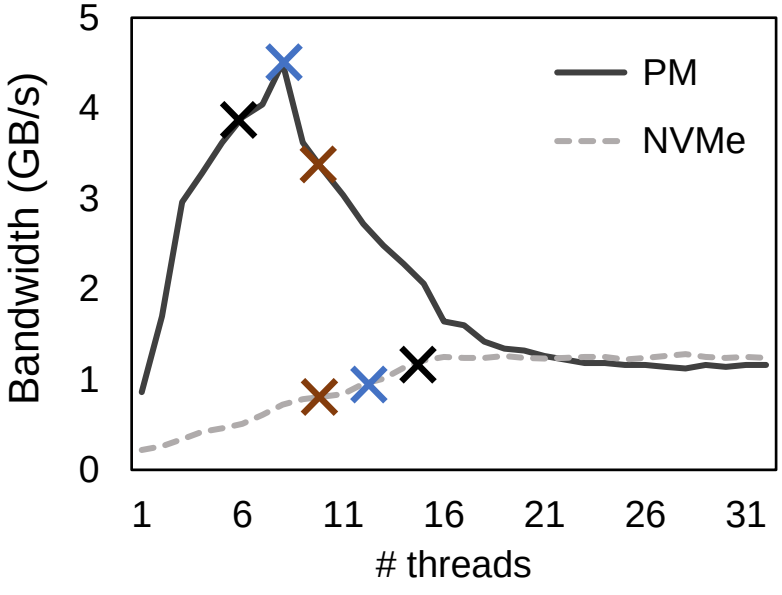




Attaining combined bandwidth for dynamic workloads

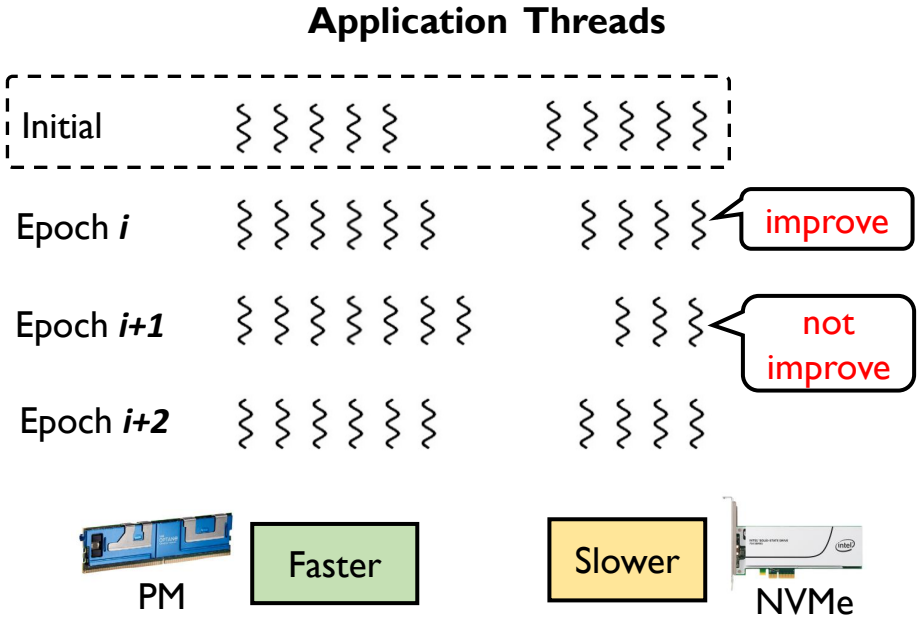
Epoch-based throughput monitoring and dynamic data placement

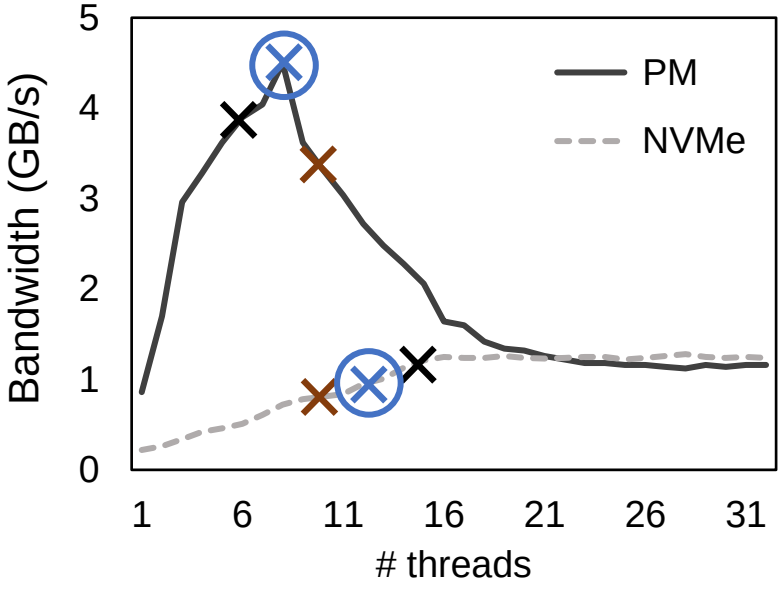




Attaining combined bandwidth for dynamic workloads

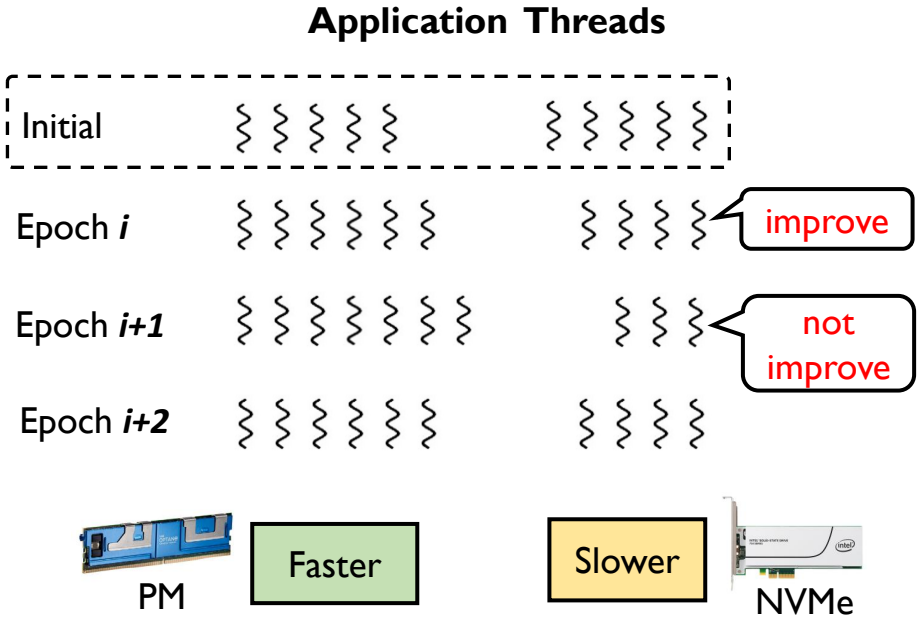
Epoch-based throughput monitoring and dynamic data placement

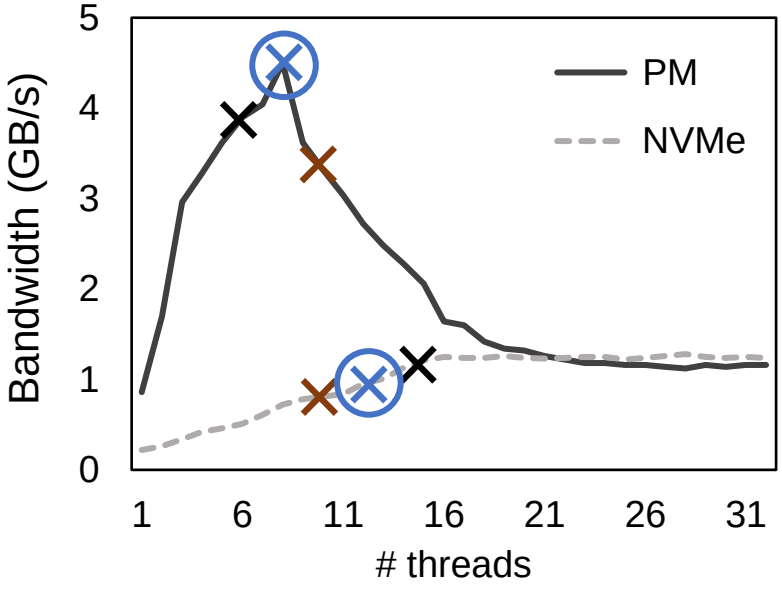




Attaining combined bandwidth for dynamic workloads

Epoch-based throughput monitoring and dynamic data placement

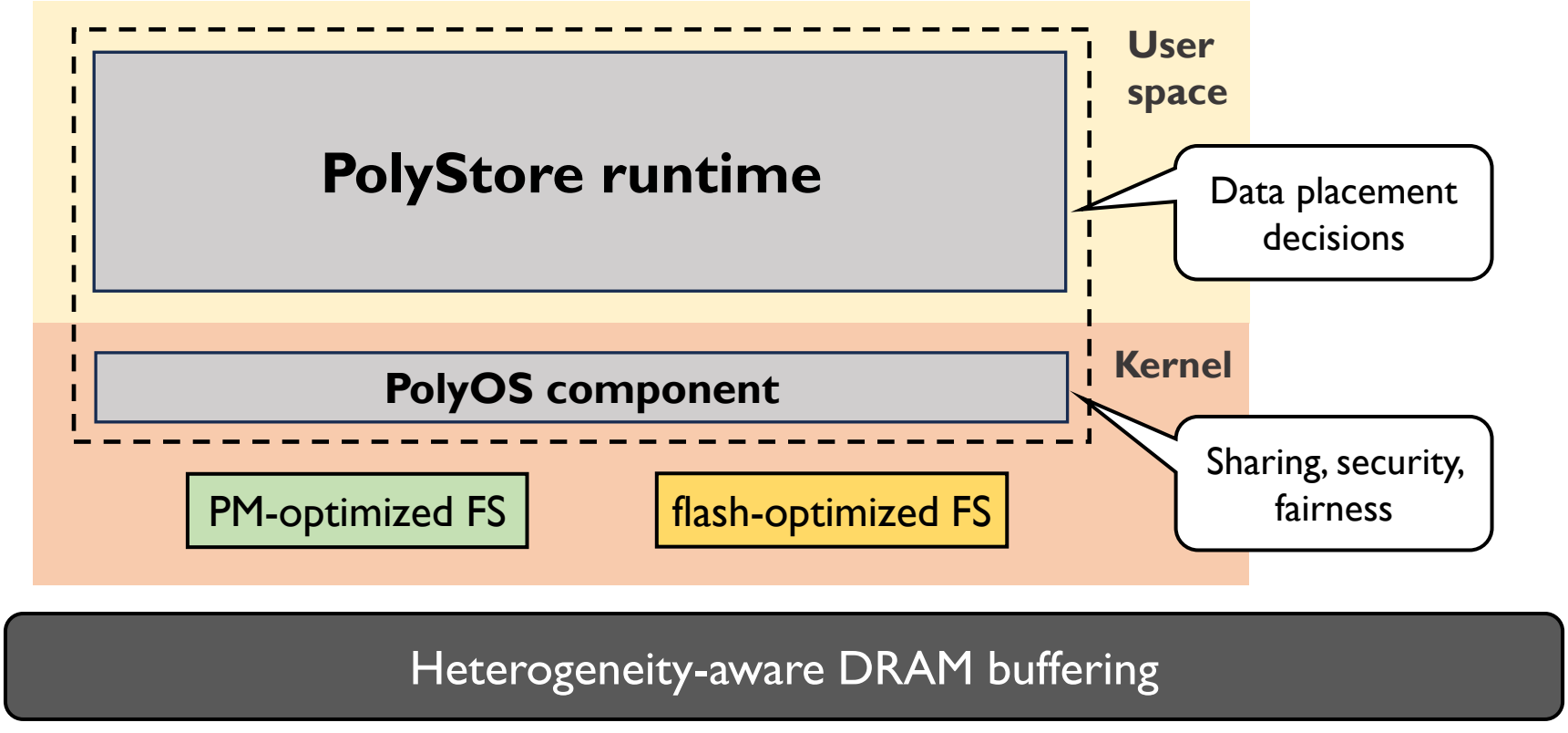




Converge to a local maximal configuration utilizing combined bandwidth!

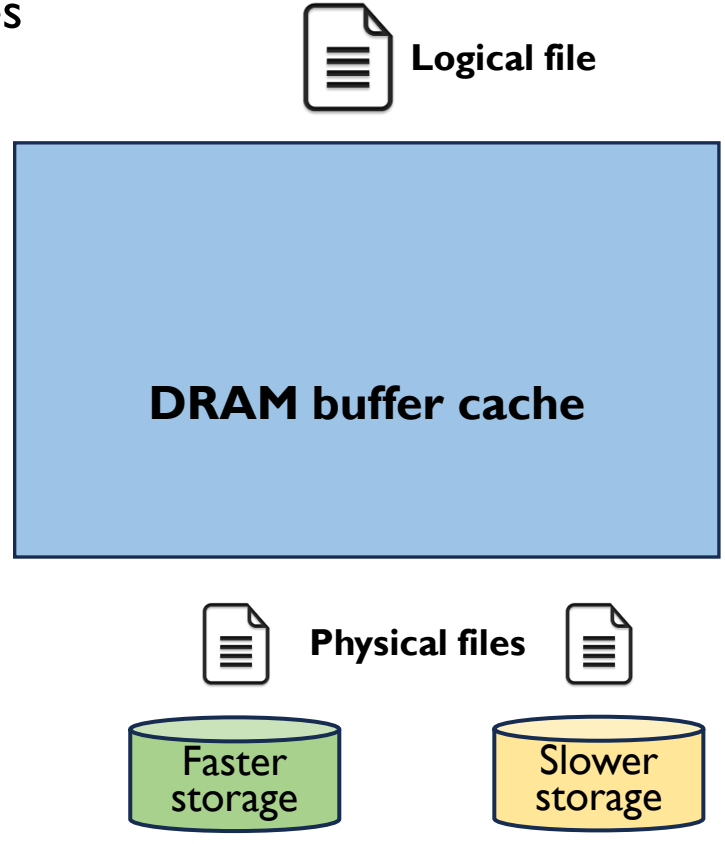
Our solution - PolyStore

A meta-layer on top of device-optimized kernel-level file systems



Heterogeneity-aware DRAM buffer cache

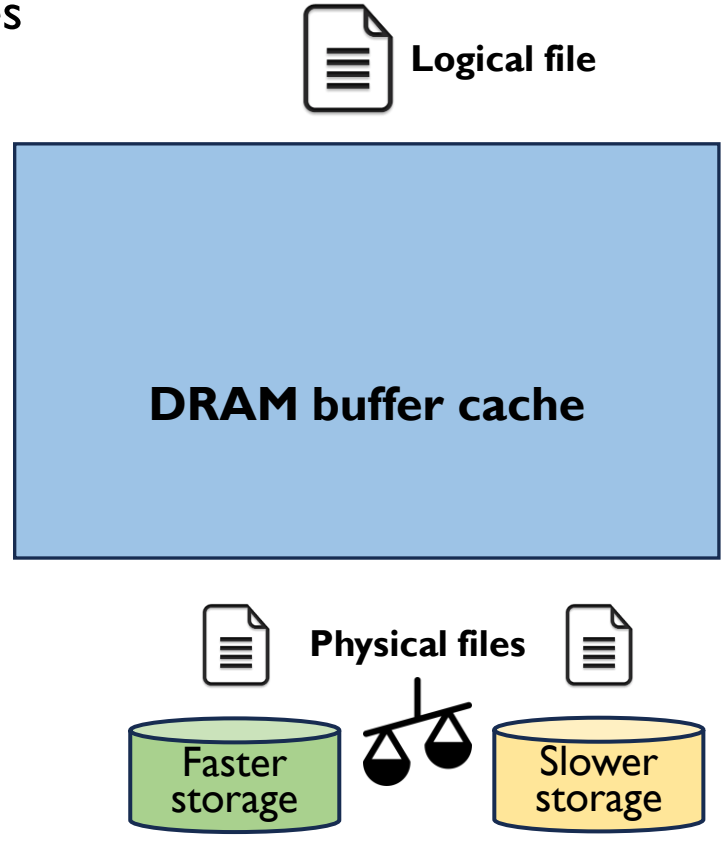
Hide access latency for heterogeneous devices



Heterogeneity-aware DRAM buffer cache

Hide access latency for heterogeneous devices

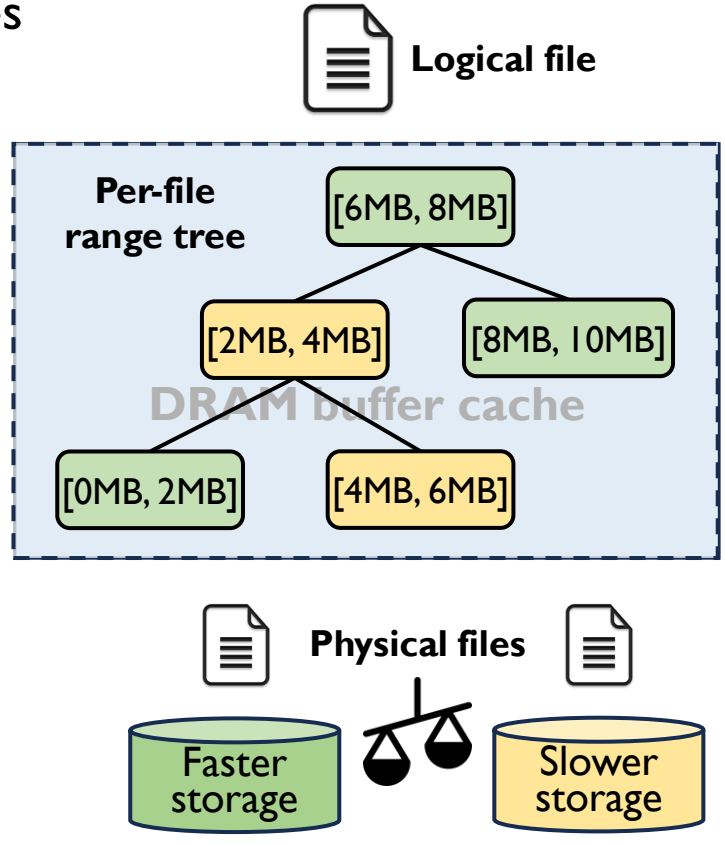
- Asymmetrical performance in heterogeneous storage devices



Heterogeneity-aware DRAM buffer cache

Hide access latency for heterogeneous devices

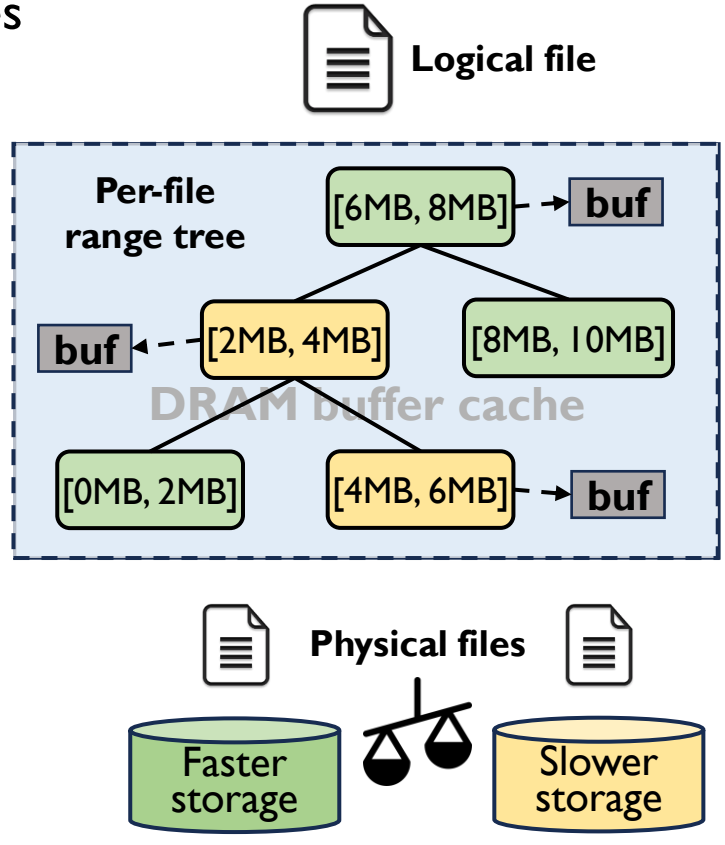
- Asymmetrical performance in heterogeneous storage devices
- Indexing structure encodes data placement information



Heterogeneity-aware DRAM buffer cache

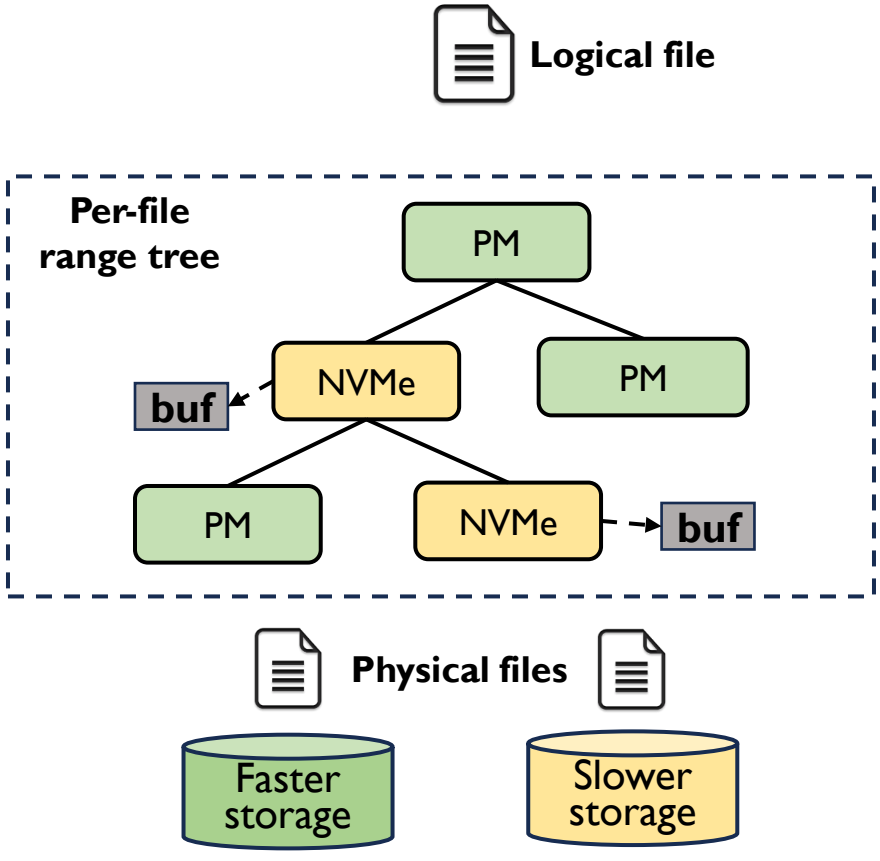
Hide access latency for heterogeneous devices

- Asymmetrical performance in heterogeneous storage devices
- Indexing structure encodes data placement information
- Allocate DRAM cache buffer with device-specific admission & eviction policies



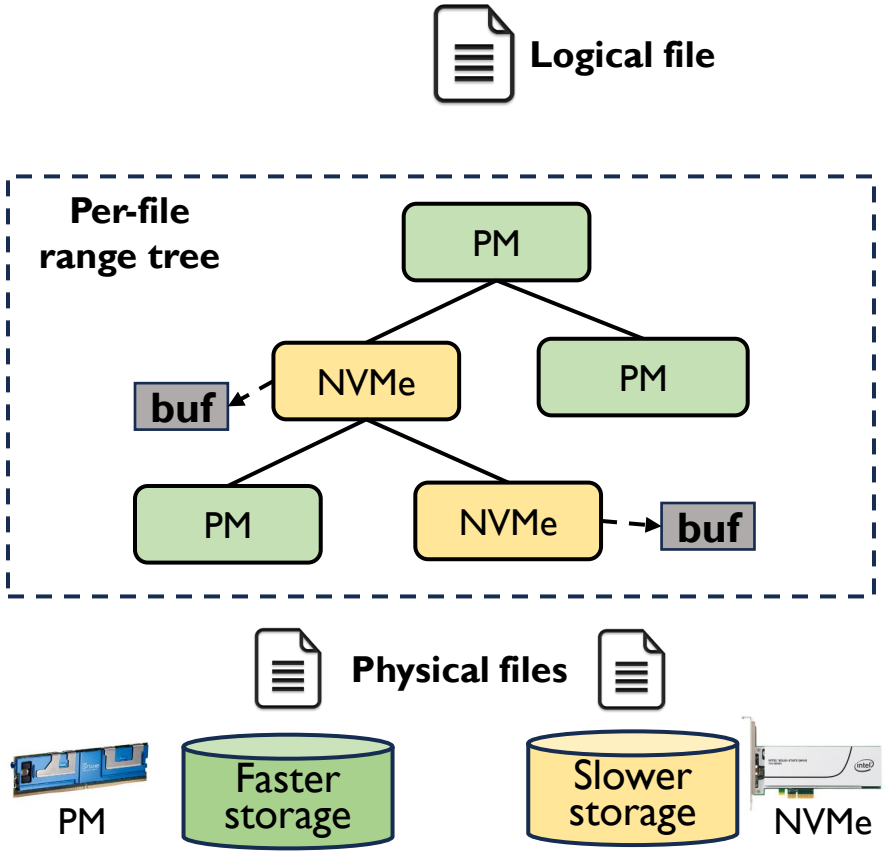
Heterogeneity-aware DRAM buffer cache

Case for using PM and NVMe



Heterogeneity-aware DRAM buffer cache

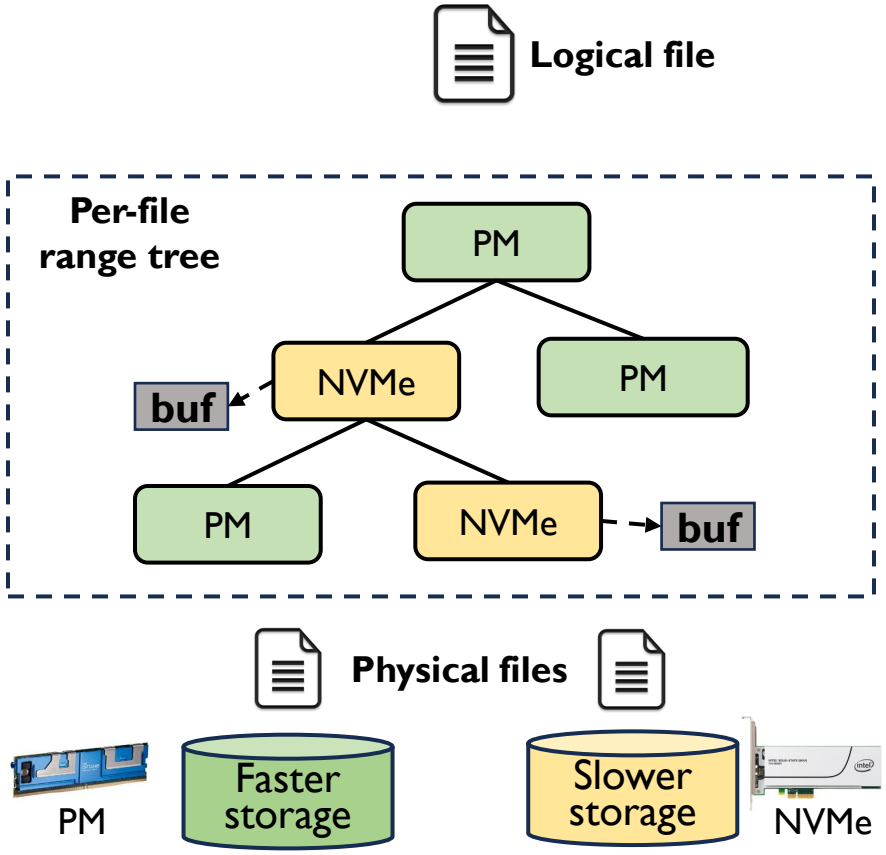
Case for using PM and NVMe



Heterogeneity-aware DRAM buffer cache

Case for using PM and NVMe

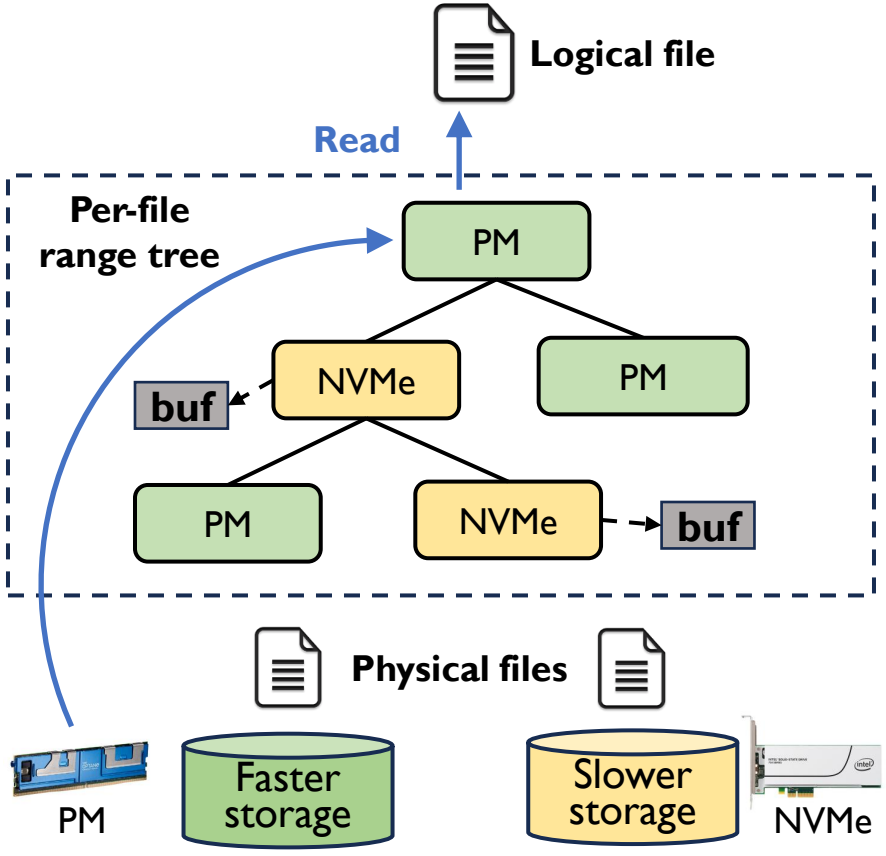
- PM read has same order of speed as DRAM



Heterogeneity-aware DRAM buffer cache

Case for using PM and NVMe

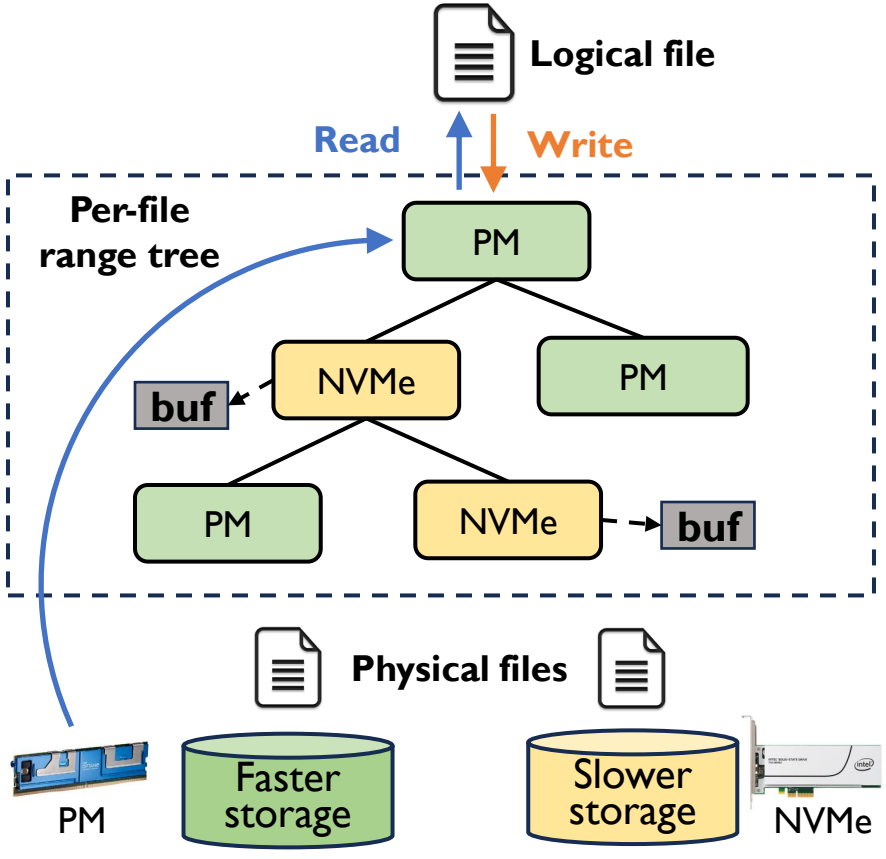
- PM read has same order of speed as DRAM
- Do not buffer data in DRAM for read-only data for PM



Heterogeneity-aware DRAM buffer cache

Case for using PM and NVMe

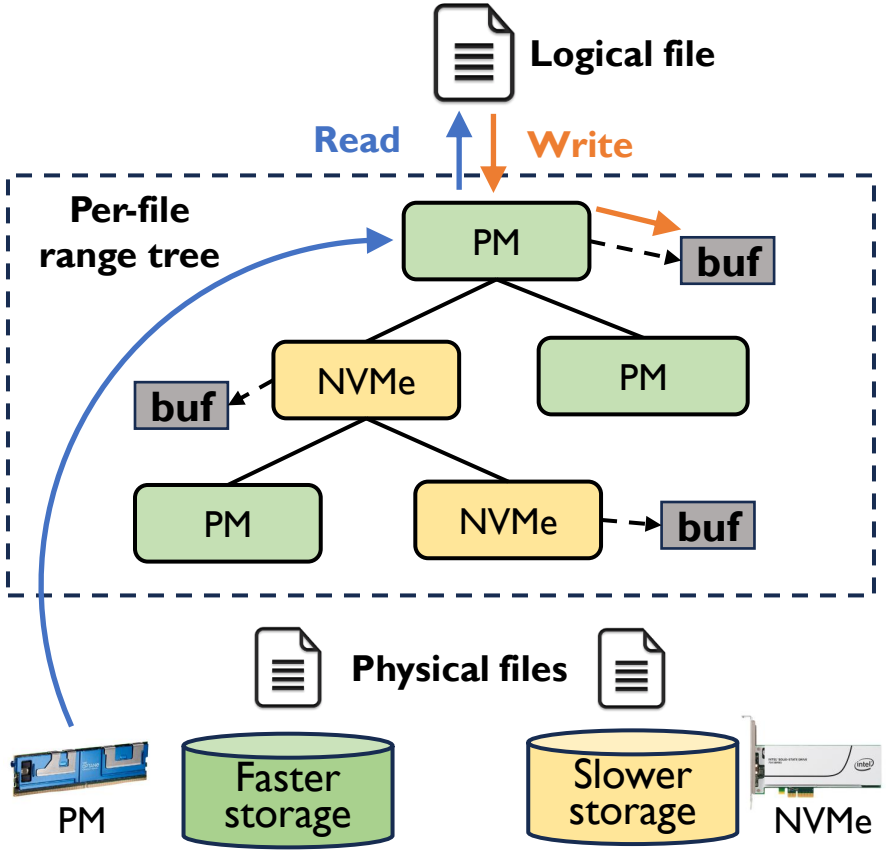
- PM read has same order of speed as DRAM
- Do not buffer data in DRAM for read-only data for PM
- PM write is slower than read



Heterogeneity-aware DRAM buffer cache

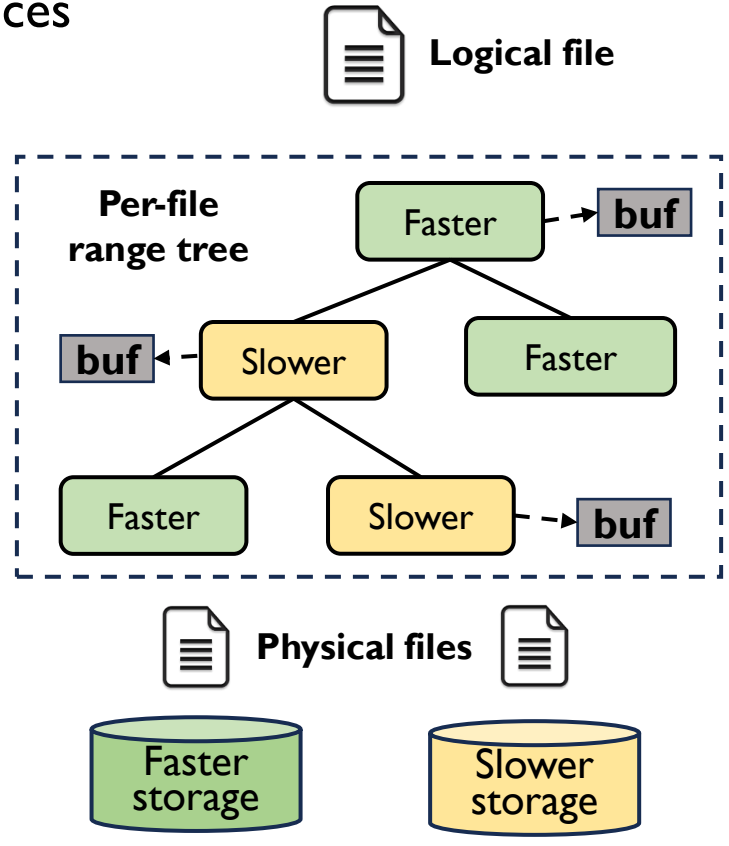
Case for using PM and NVMe

- PM read has same order of speed as DRAM
- Do not buffer data in DRAM for read-only data for PM
- PM write is slower than read
- Allocate DRAM buffer for PM only when data is modified



Heterogeneity-aware DRAM buffer cache

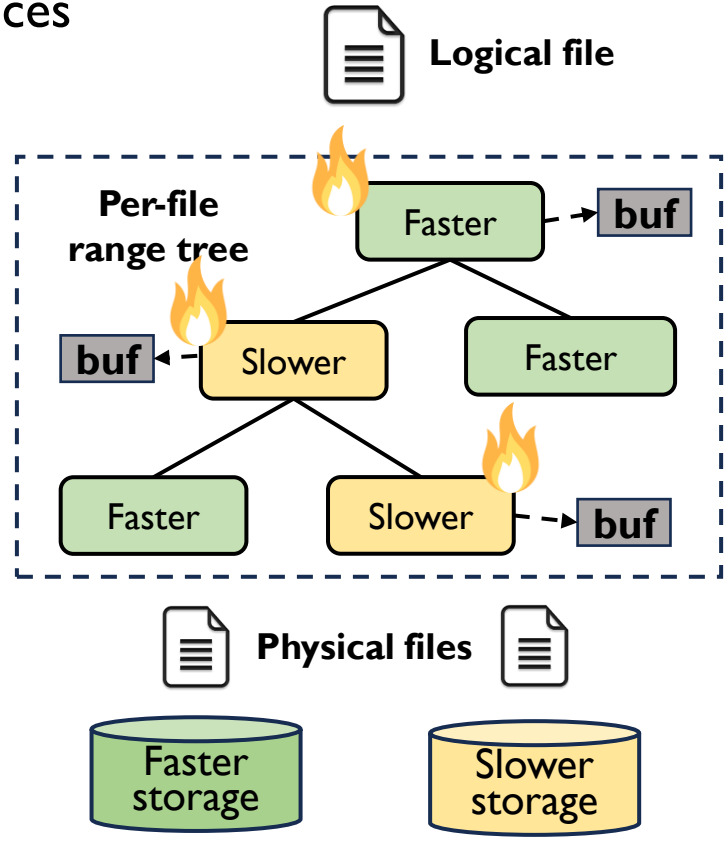
Facilitating data migration decisions across devices



Heterogeneity-aware DRAM buffer cache

Facilitating data migration decisions across devices

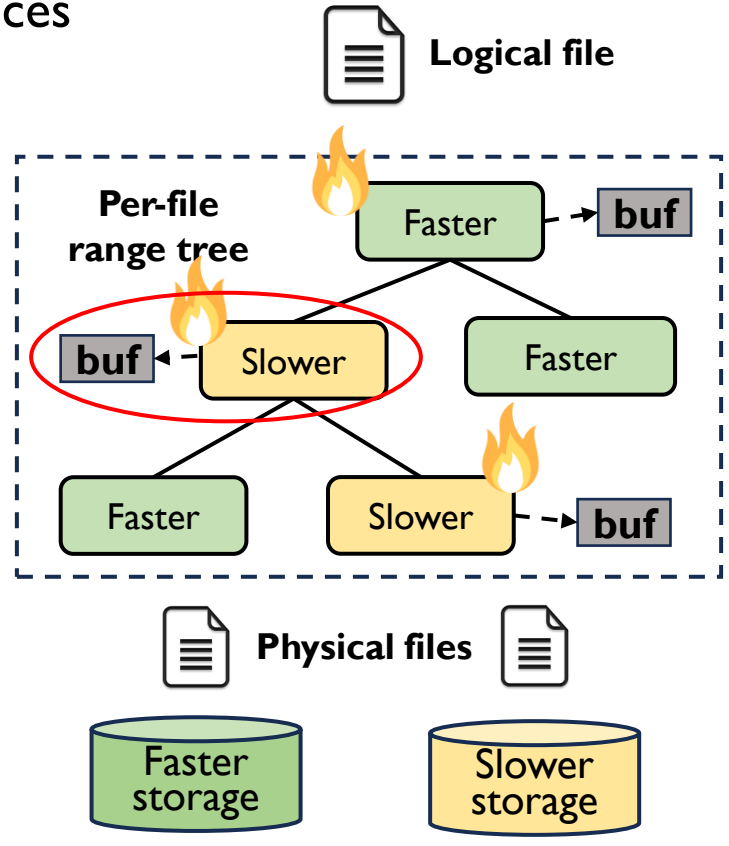
- Track data hotness/coldness



Heterogeneity-aware DRAM buffer cache

Facilitating data migration decisions across devices

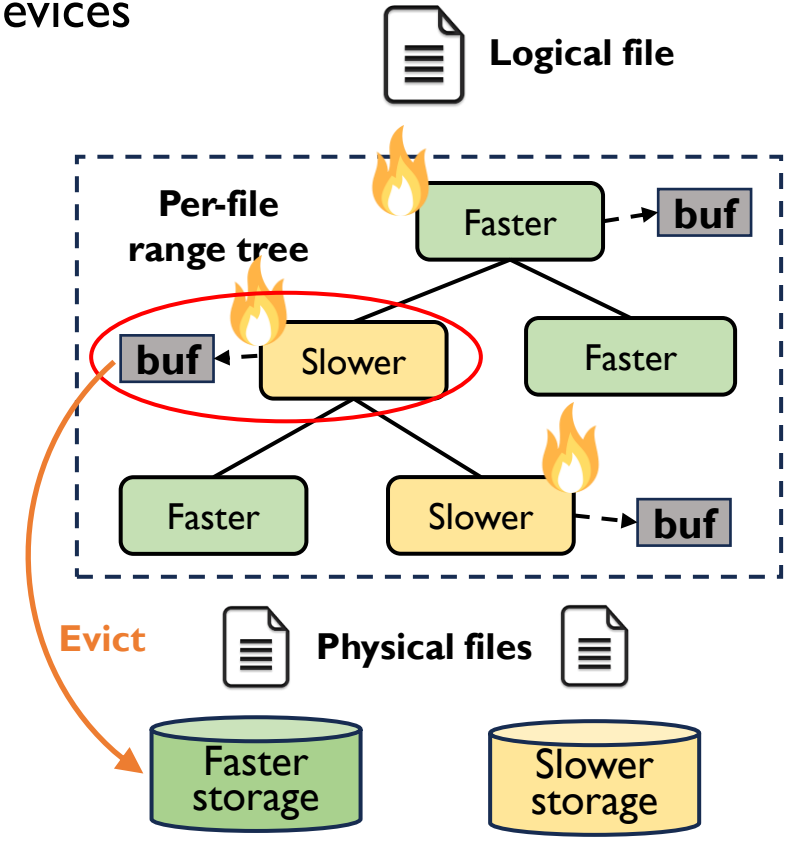
- Track data hotness/coldness
- Hot data on slow device selected as victim due to memory pressure



Heterogeneity-aware DRAM buffer cache

Facilitating data migration decisions across devices

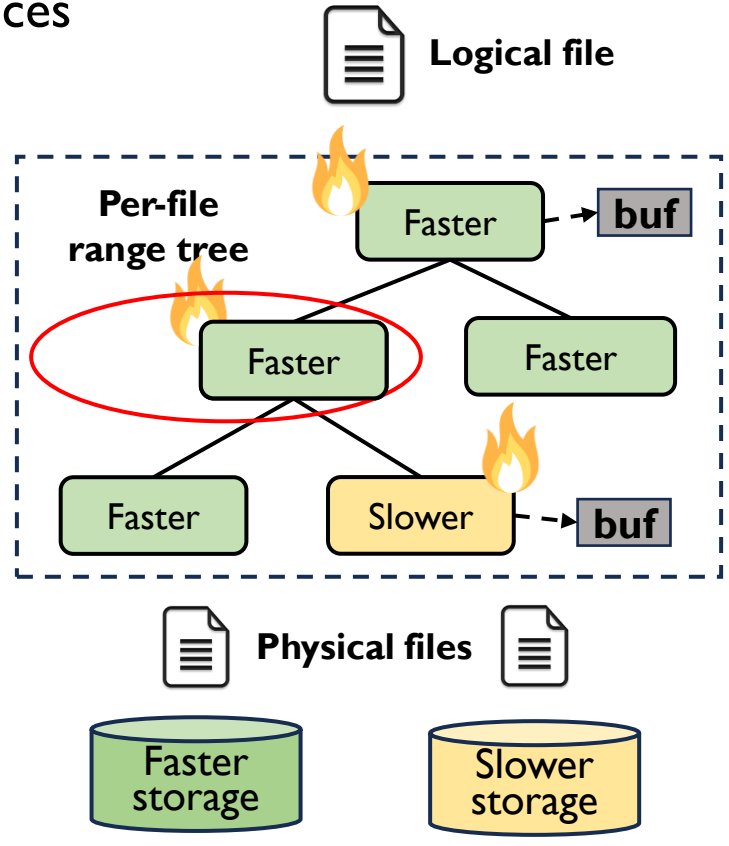
- Track data hotness/coldness
- Hot data on slow device selected as victim due to memory pressure
- Flush hot data on slower device to faster one if space permits



Heterogeneity-aware DRAM buffer cache

Facilitating data migration decisions across devices

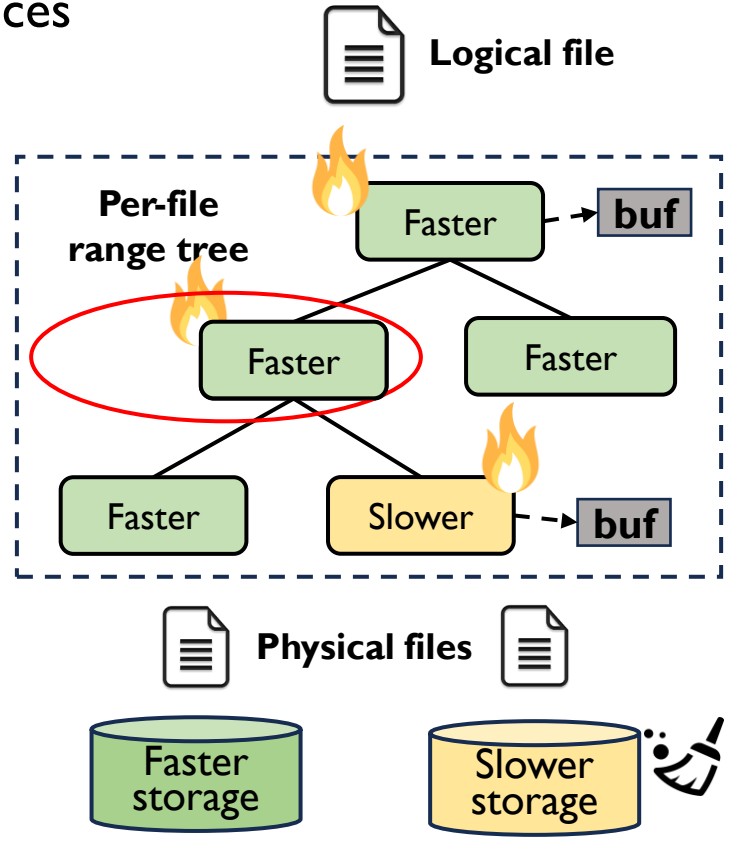
- Track data hotness/coldness
- Hot data on slow device selected as victim due to memory pressure
- Flush hot data on slower device to faster one if space permits



Heterogeneity-aware DRAM buffer cache

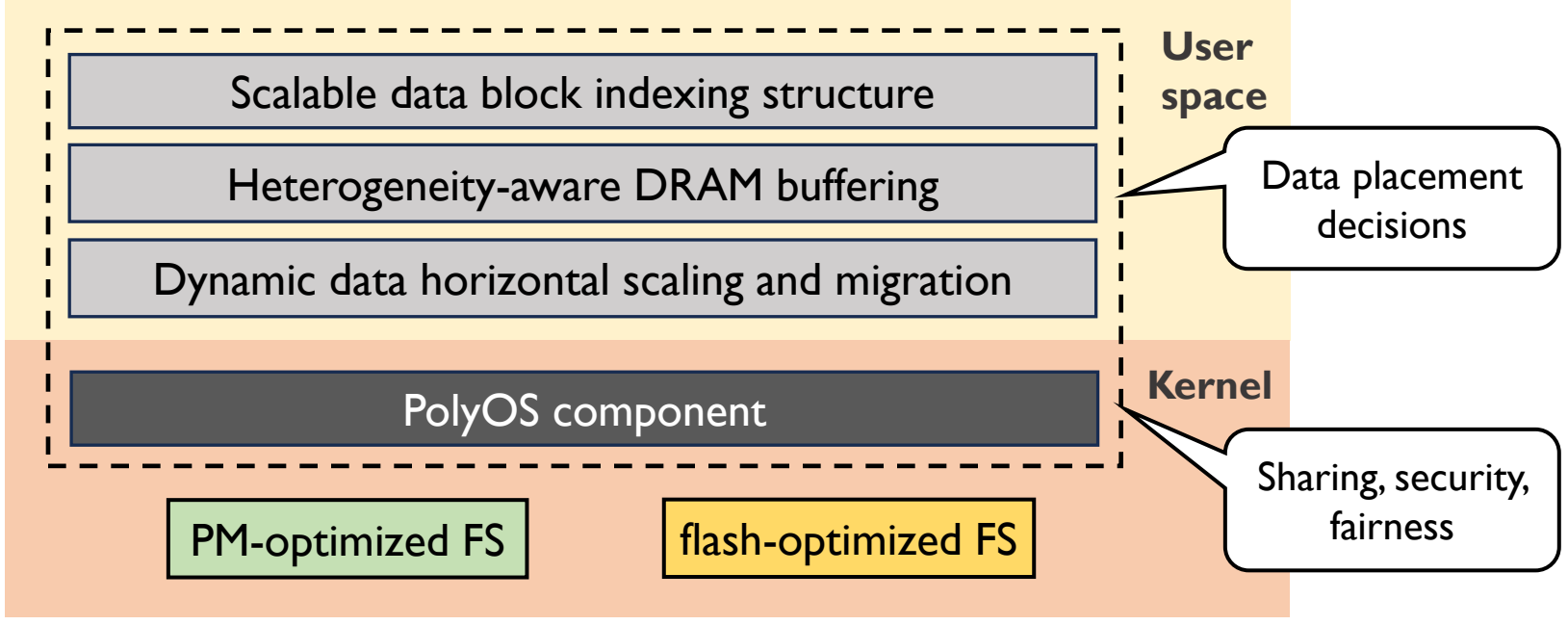
Facilitating data migration decisions across devices

- Track data hotness/coldness
- Hot data on slow device selected as victim due to memory pressure
- Flush hot data on slower device to faster one if space permits
- Do garbage collection for the original data block on slower device



Our solution - PolyStore

A meta-layer on top of device-optimized kernel-level file systems



More technical details in our paper!

Evaluation

- Can PolyStore utilize the combined bandwidth of storage devices?
- Can PolyStore improve performance for real-world applications?

Experiment Setup

PM

NVMe

NOVA

Ext4

State-of-the-art systems

Caching:

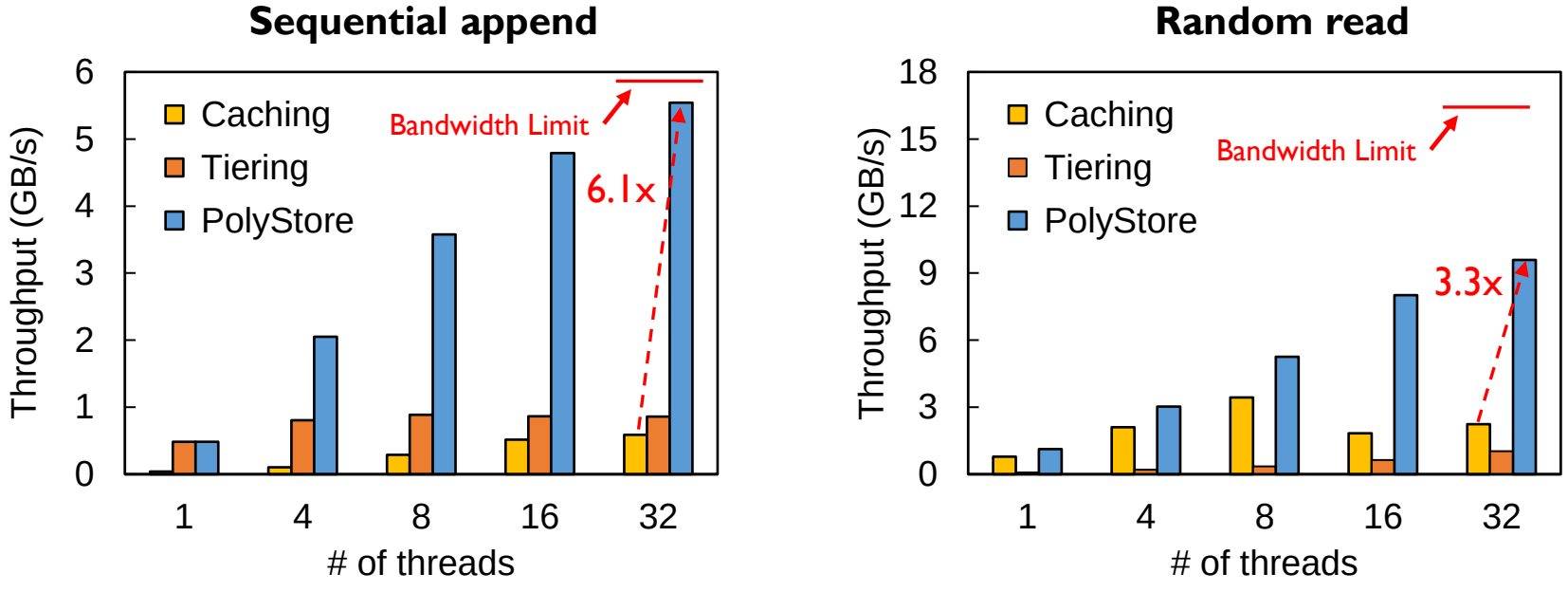
Orthus [FAST '21]

Tiering:

SPFS [FAST '23]

Evaluation: Utilize combined devices' bandwidth

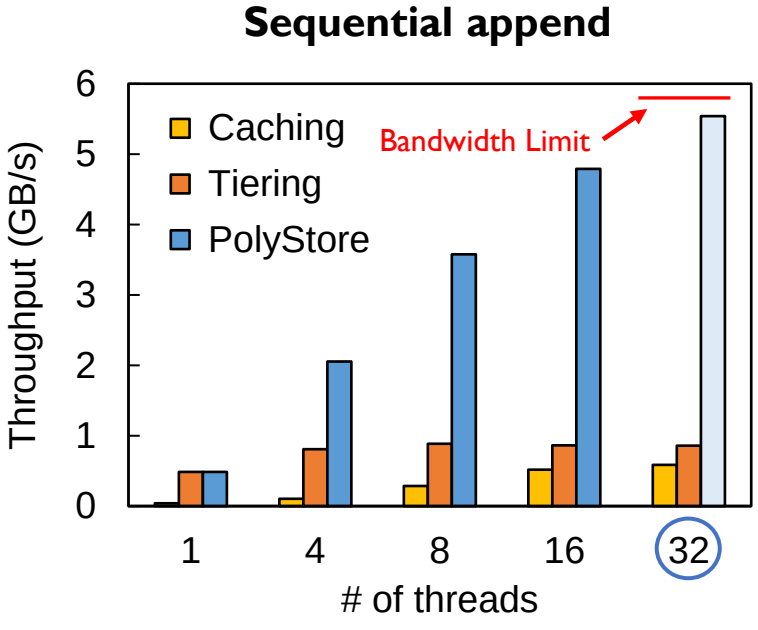
Multi-thread benchmark access private 2GB files (O_DIRECT flag)



- Scalable indexes distribute data across PM and NVMe
- Dynamic data placement effectively utilize the combined bandwidth

Evaluation: Utilize combined devices' bandwidth

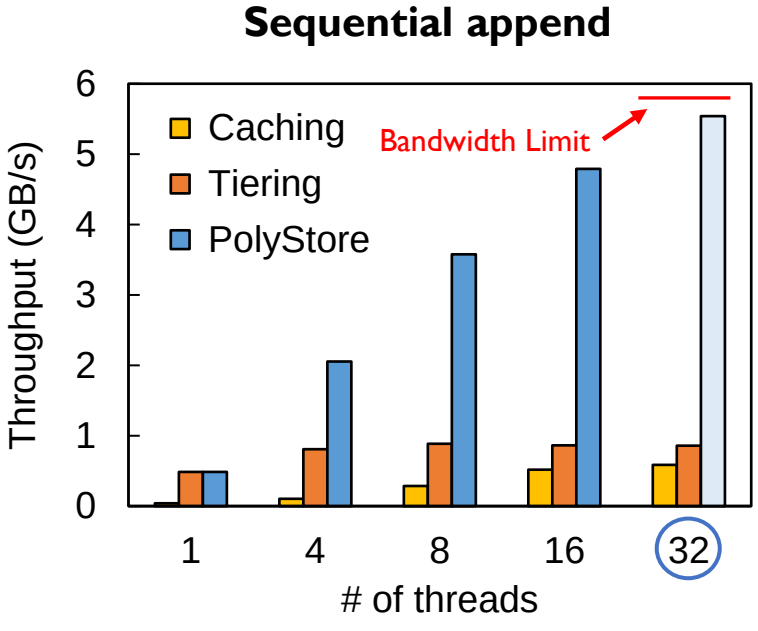
32 benchmark threads access private 2GB files (O_DIRECT flag)



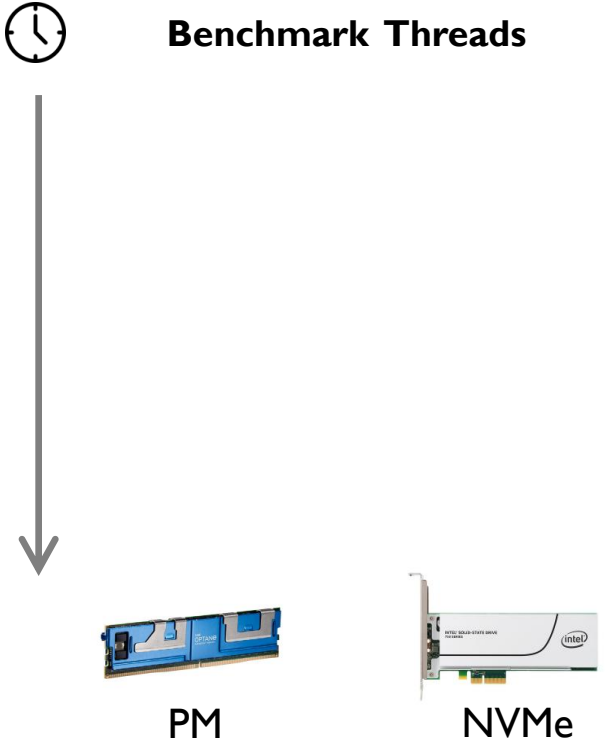
Static coarse-grained data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

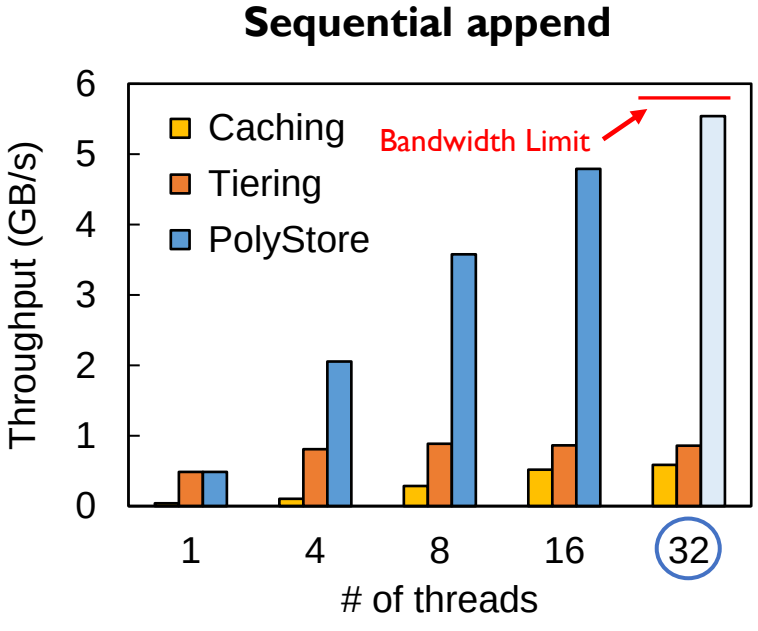


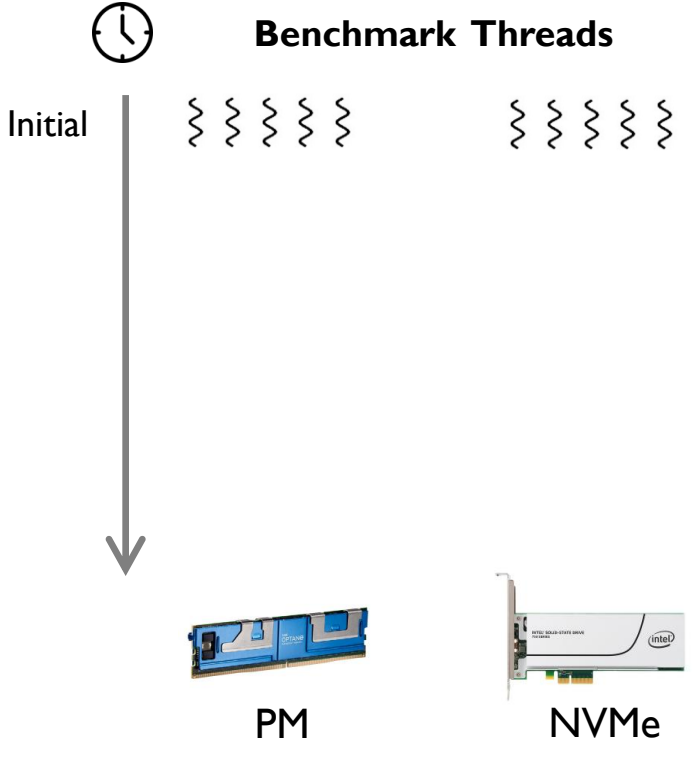
Static coarse-grained data placement



Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

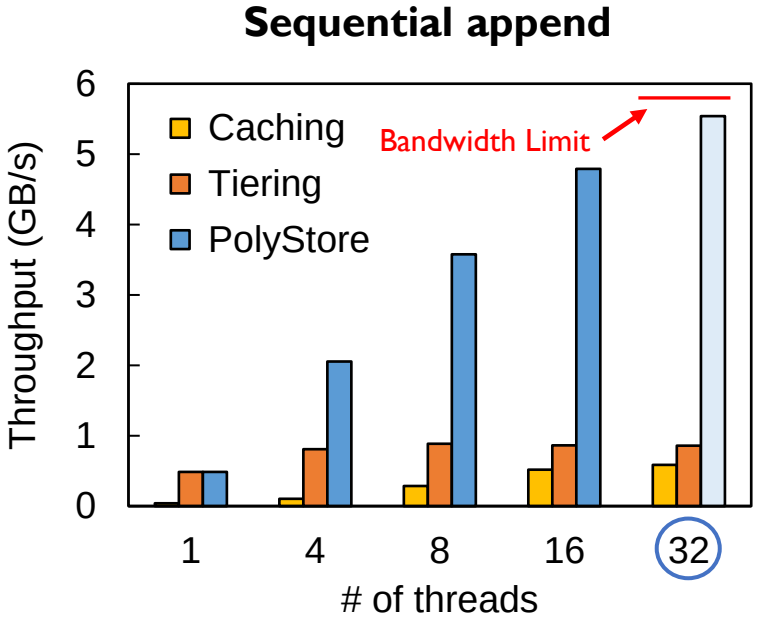


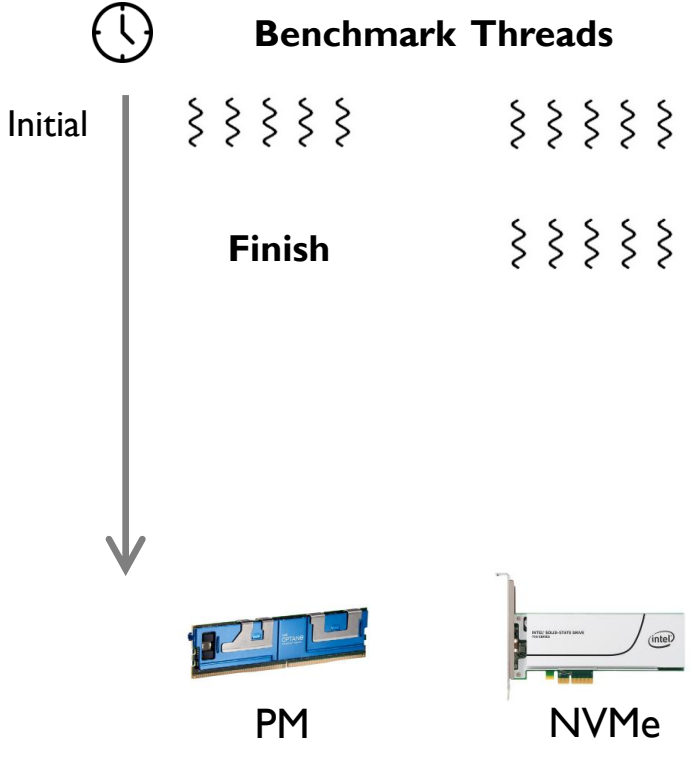


Static coarse-grained data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

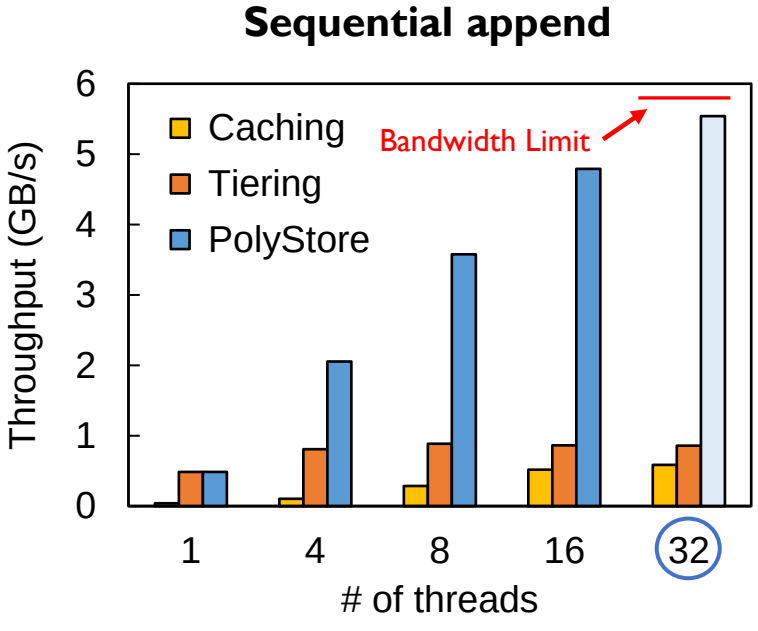


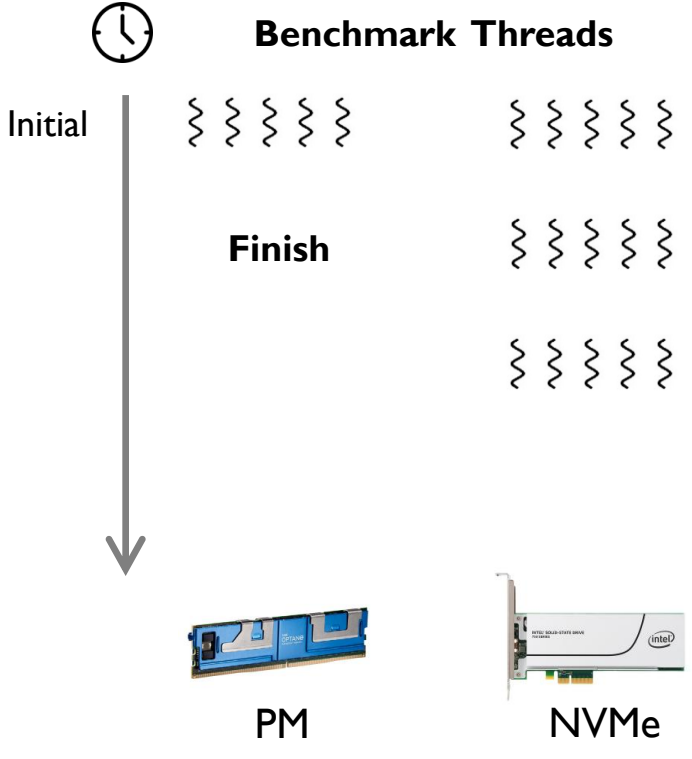


Static coarse-grained data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

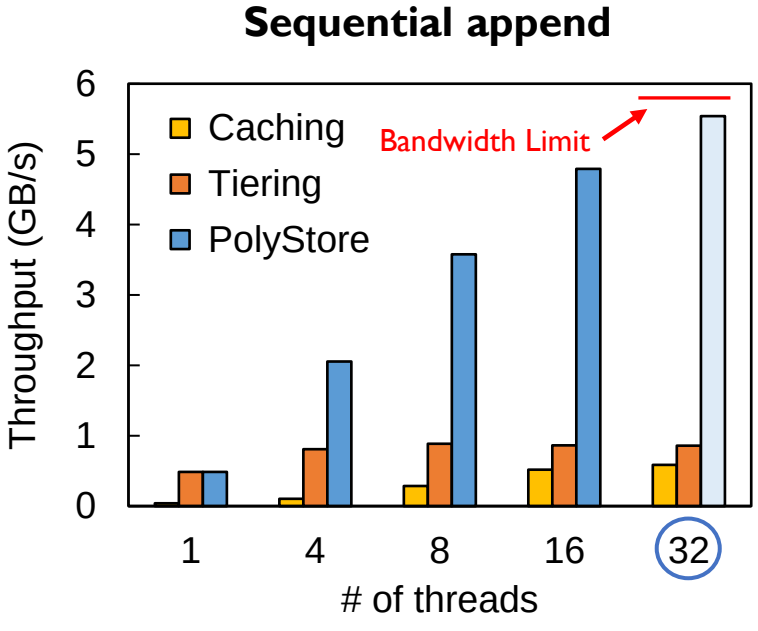


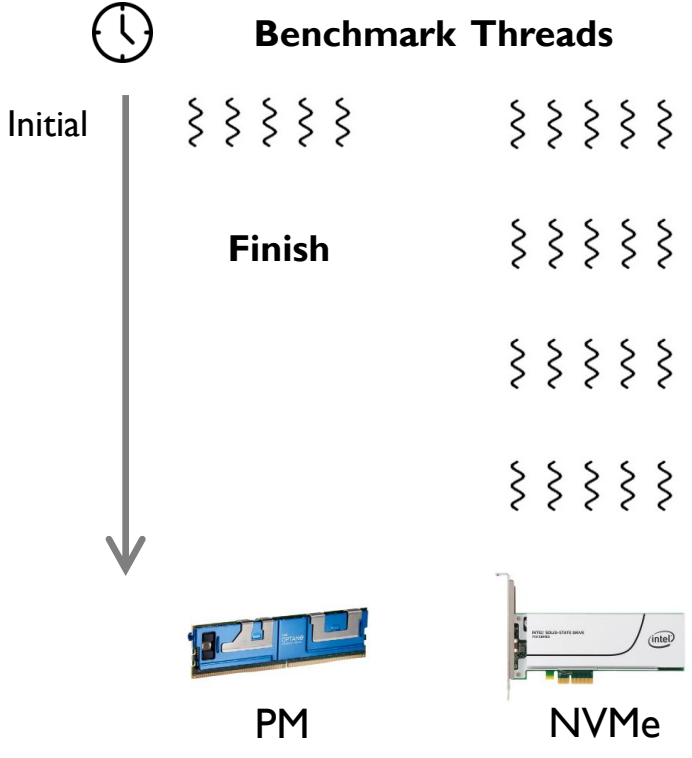


Static coarse-grained data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

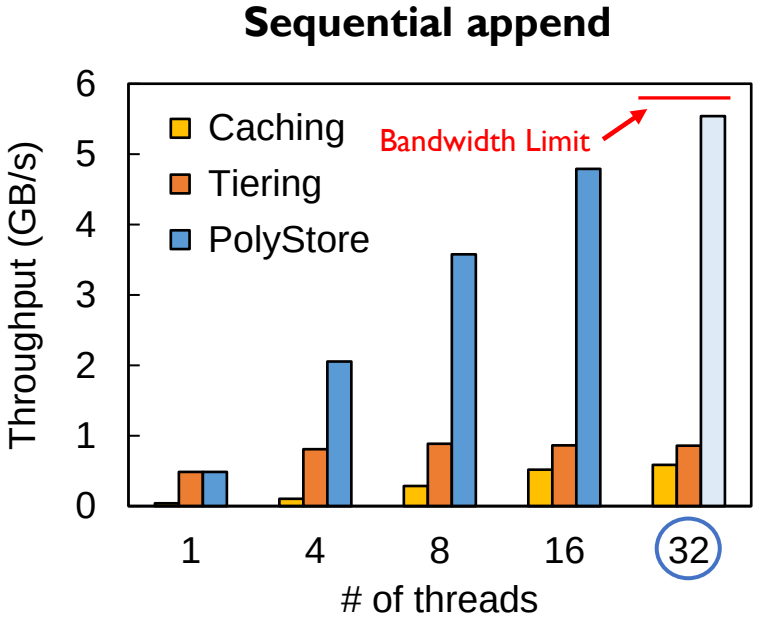


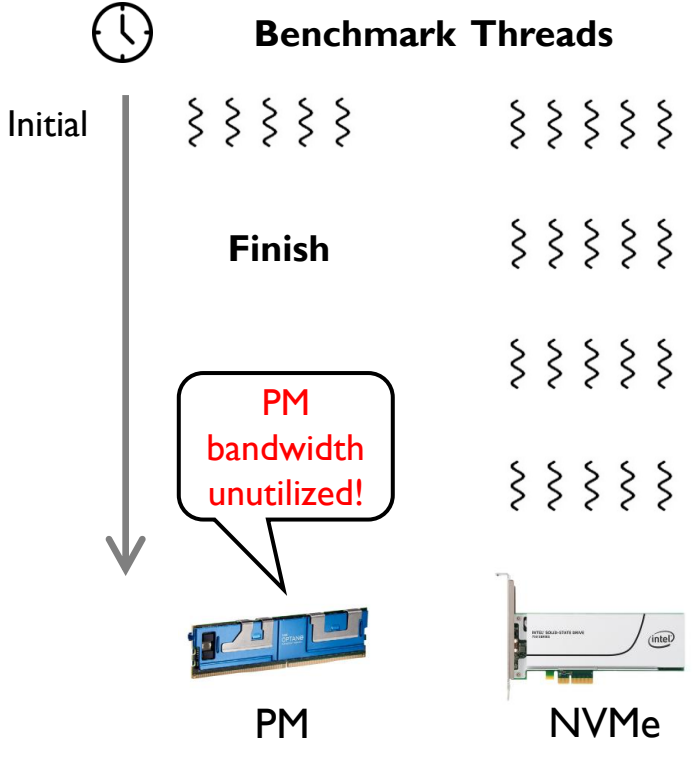


Static coarse-grained data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

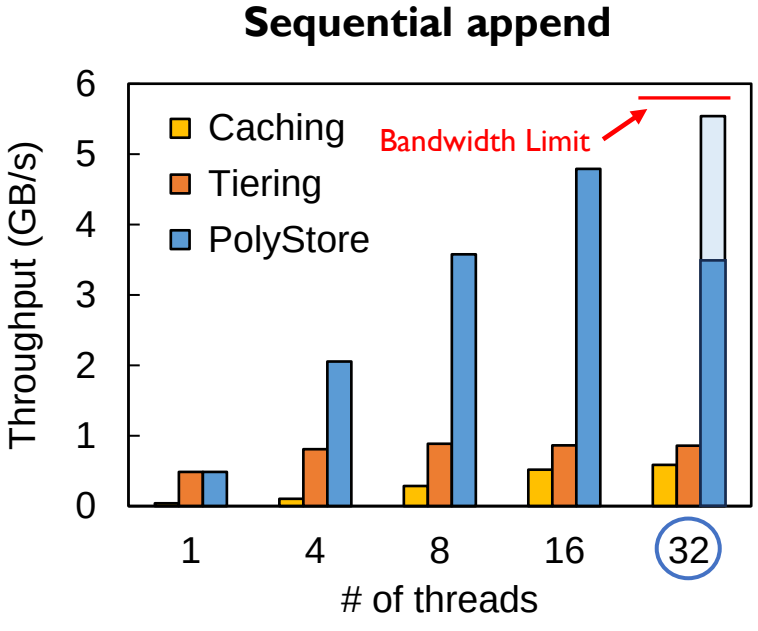


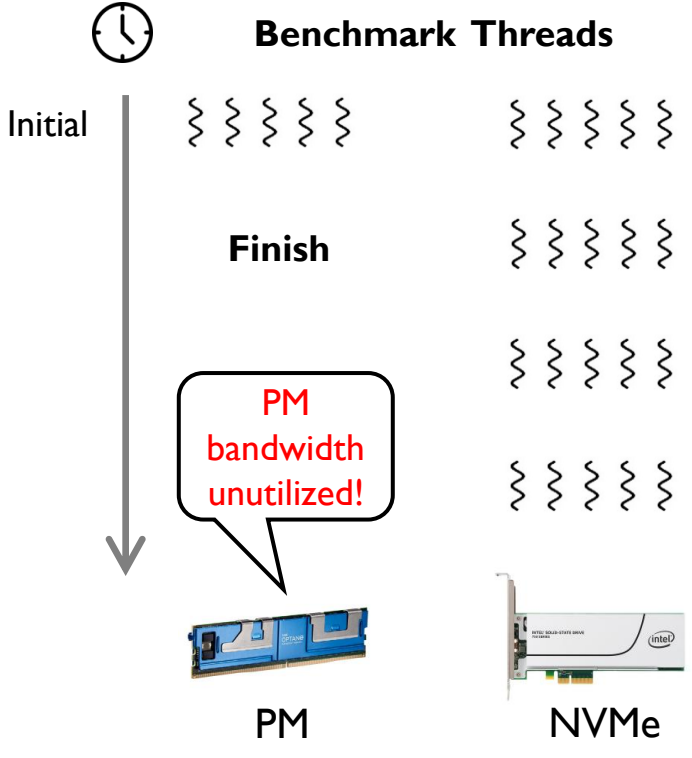


Static coarse-grained data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

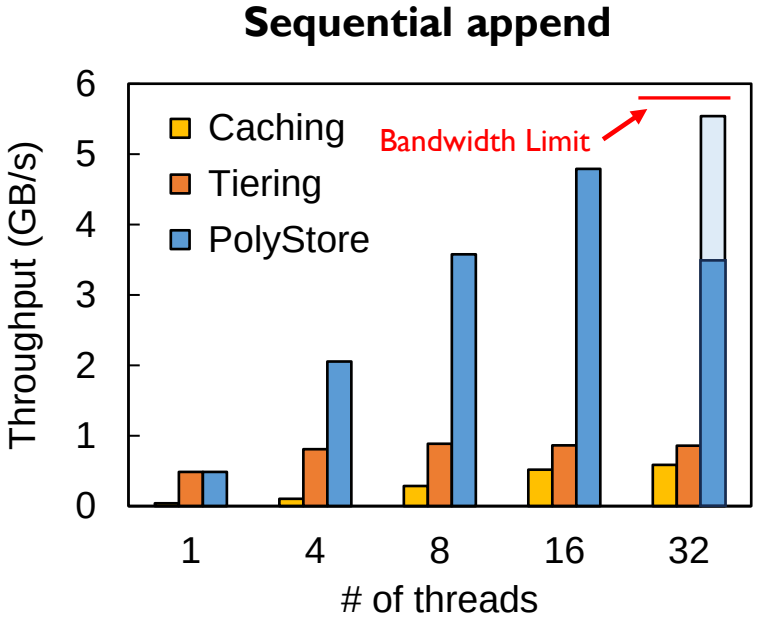




Static coarse-grained data placement

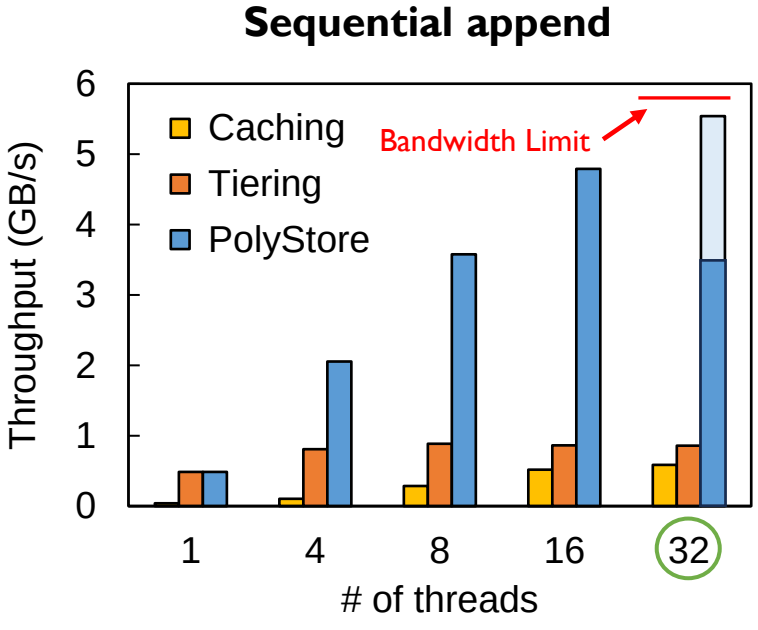
Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)



Evaluation: Utilize combined devices' bandwidth

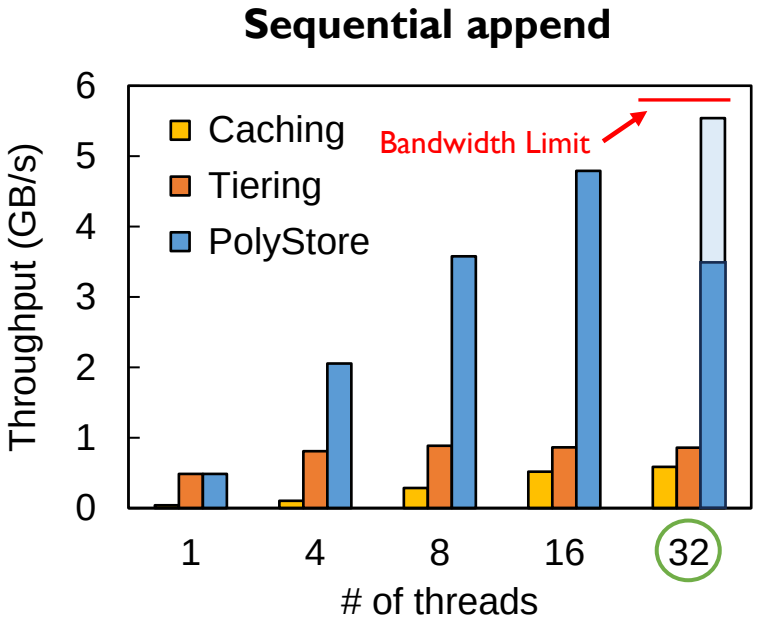
32 benchmark threads access private 2GB files (O_DIRECT flag)

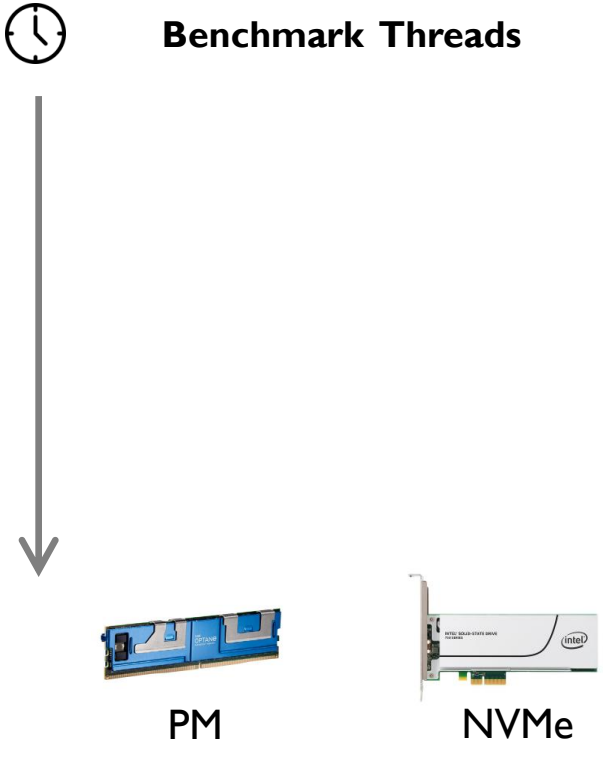


PolyStore data indexes + dynamic data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

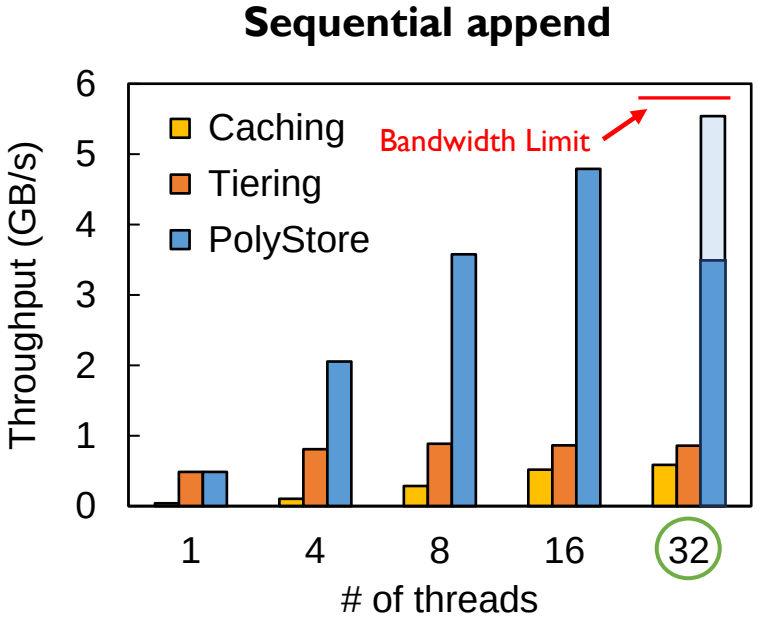


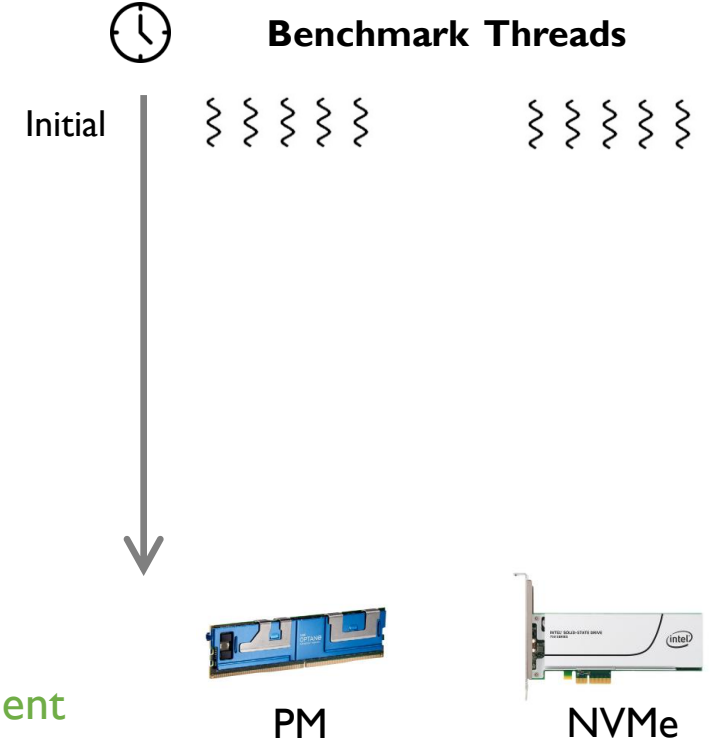


PolyStore data indexes + dynamic data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

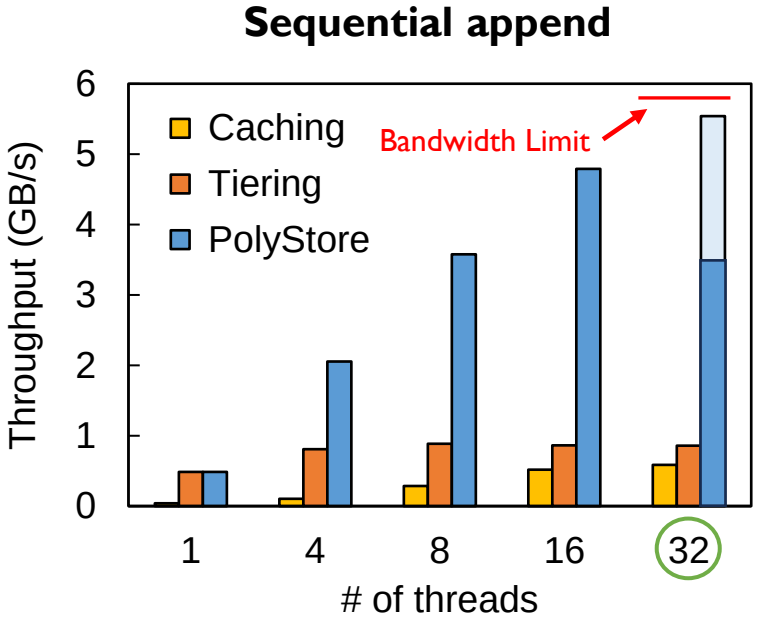


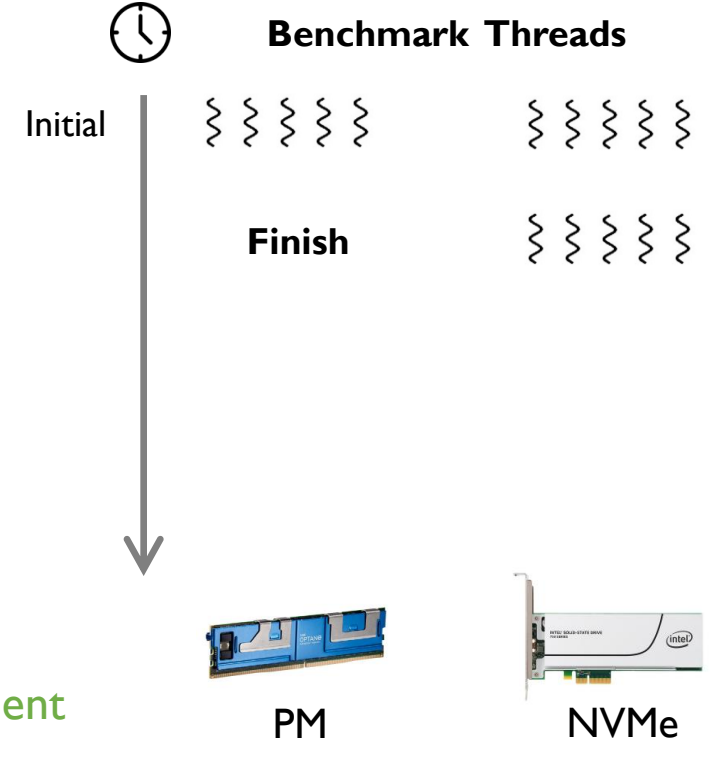


PolyStore data indexes + dynamic data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

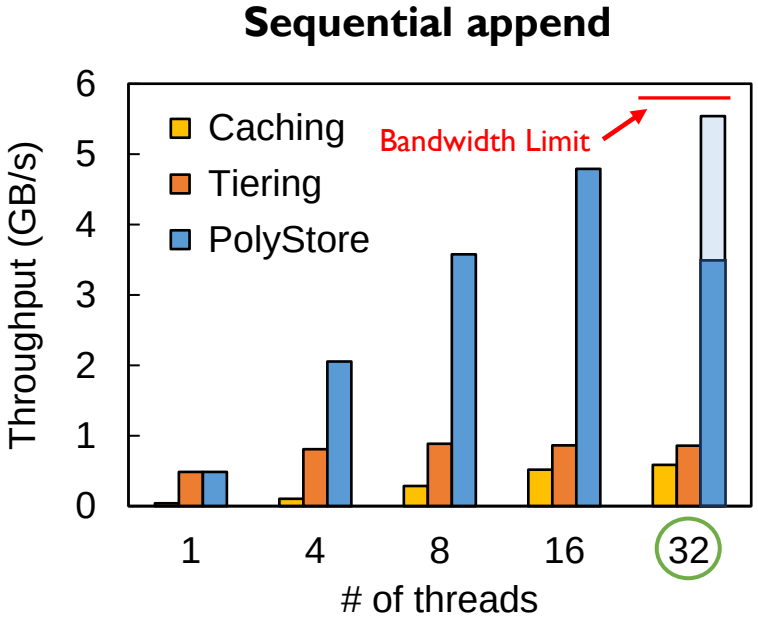


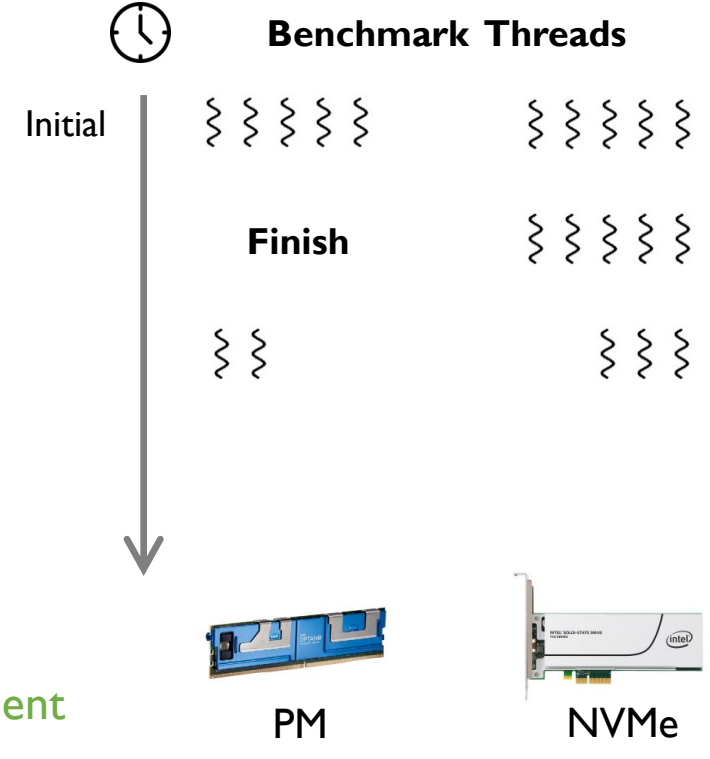


PolyStore data indexes + dynamic data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

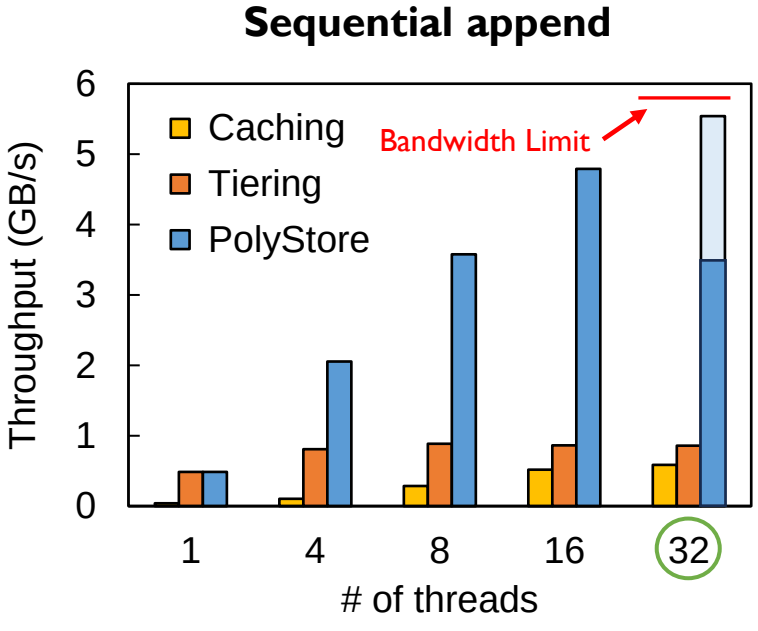


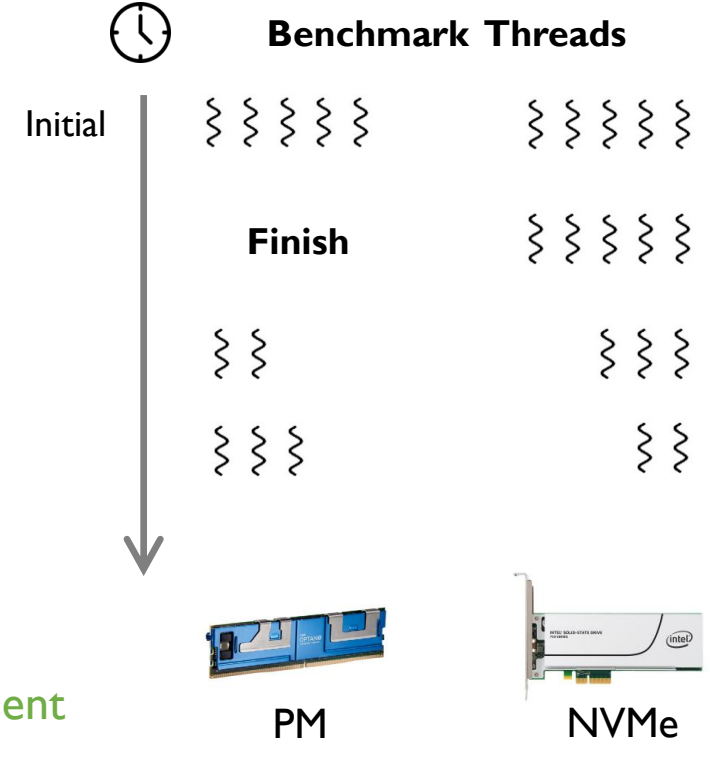


PolyStore data indexes + dynamic data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

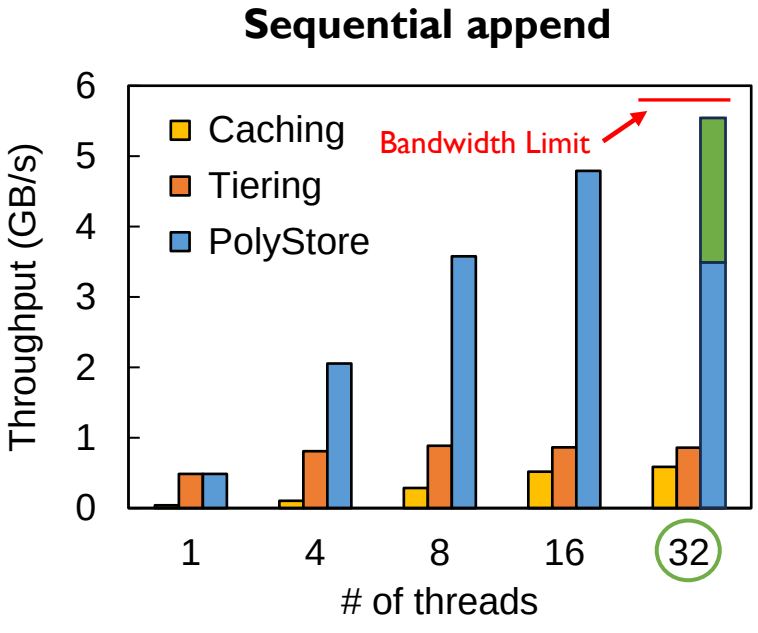


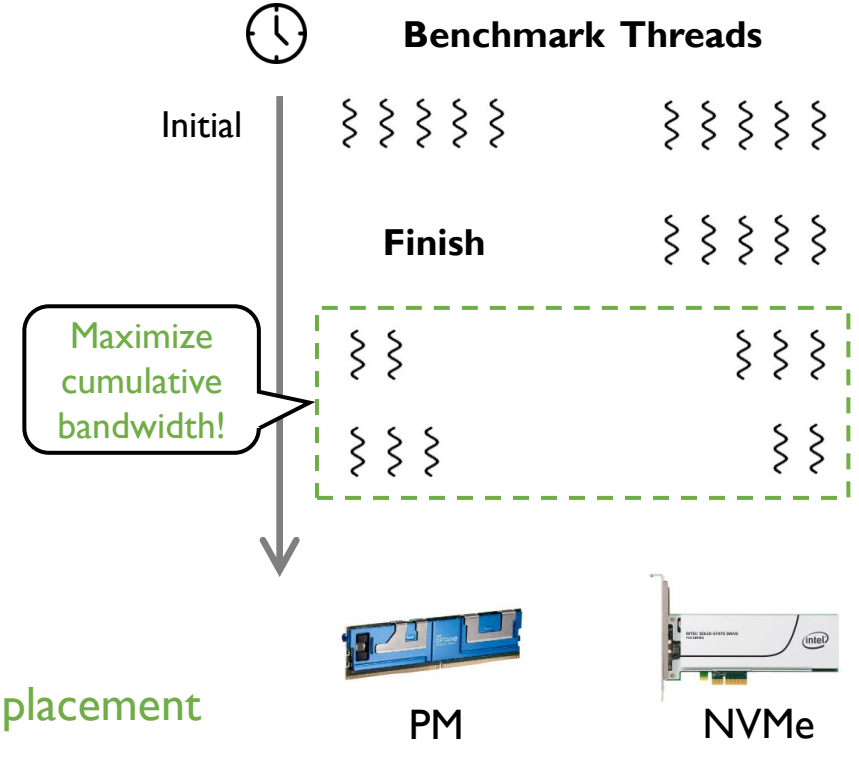


PolyStore data indexes + dynamic data placement

Evaluation: Utilize combined devices' bandwidth

32 benchmark threads access private 2GB files (O_DIRECT flag)

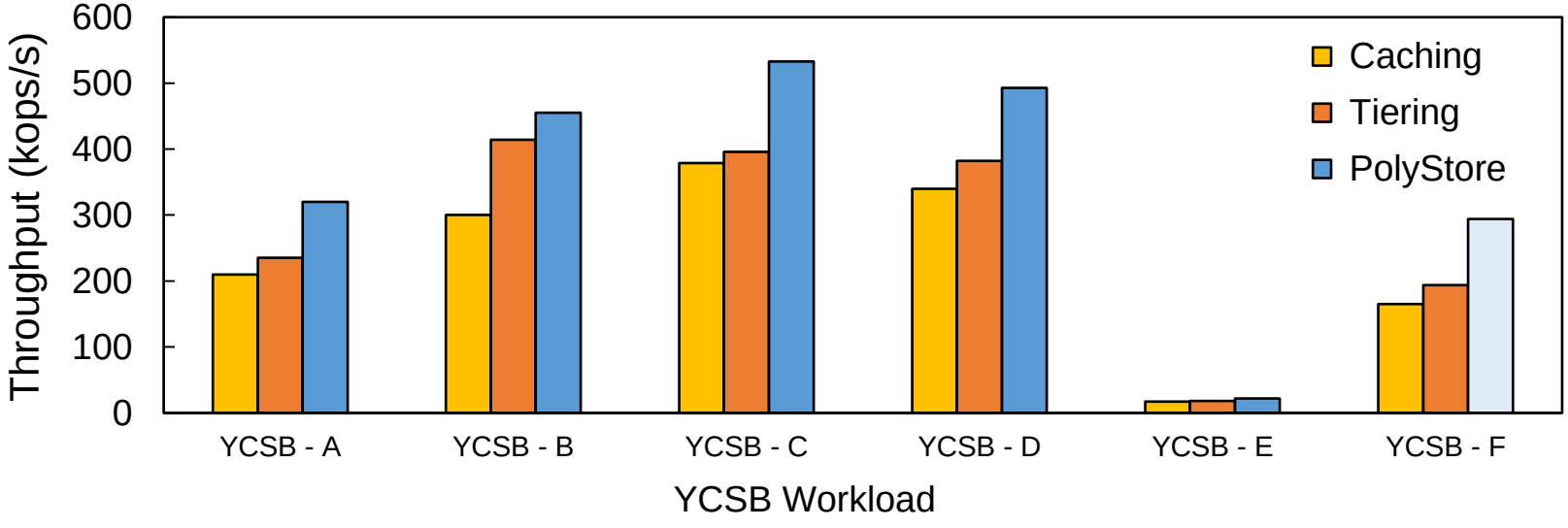




PolyStore data indexes + dynamic data placement

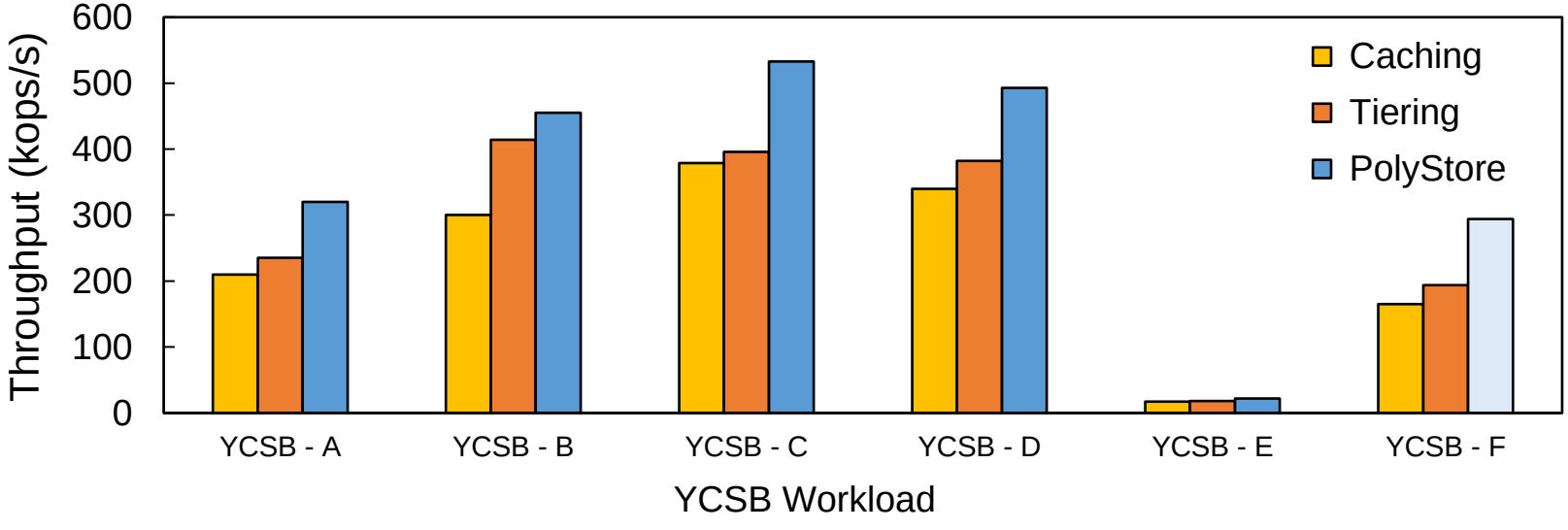
Real applications: RocksDB with YCSB workload

Enabling DRAM buffer cache for all approaches



Real applications: RocksDB with YCSB workload

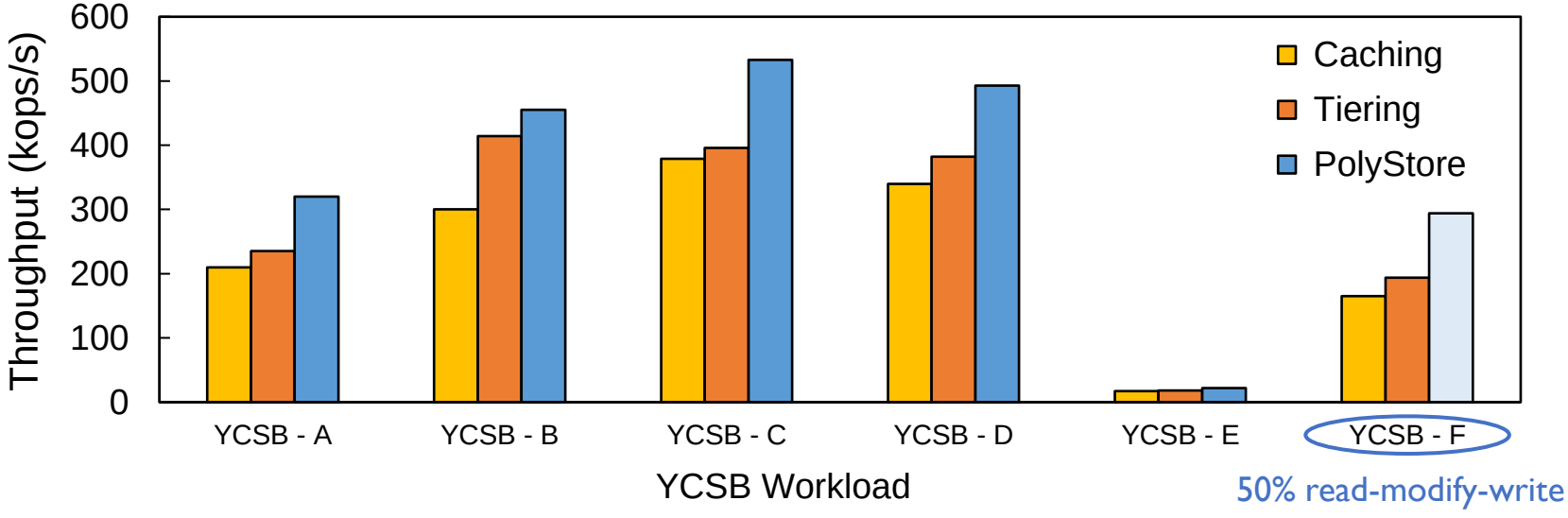
Enabling DRAM buffer cache for all approaches



- The OS page cache cannot handle device characteristics

Real applications: RocksDB with YCSB workload

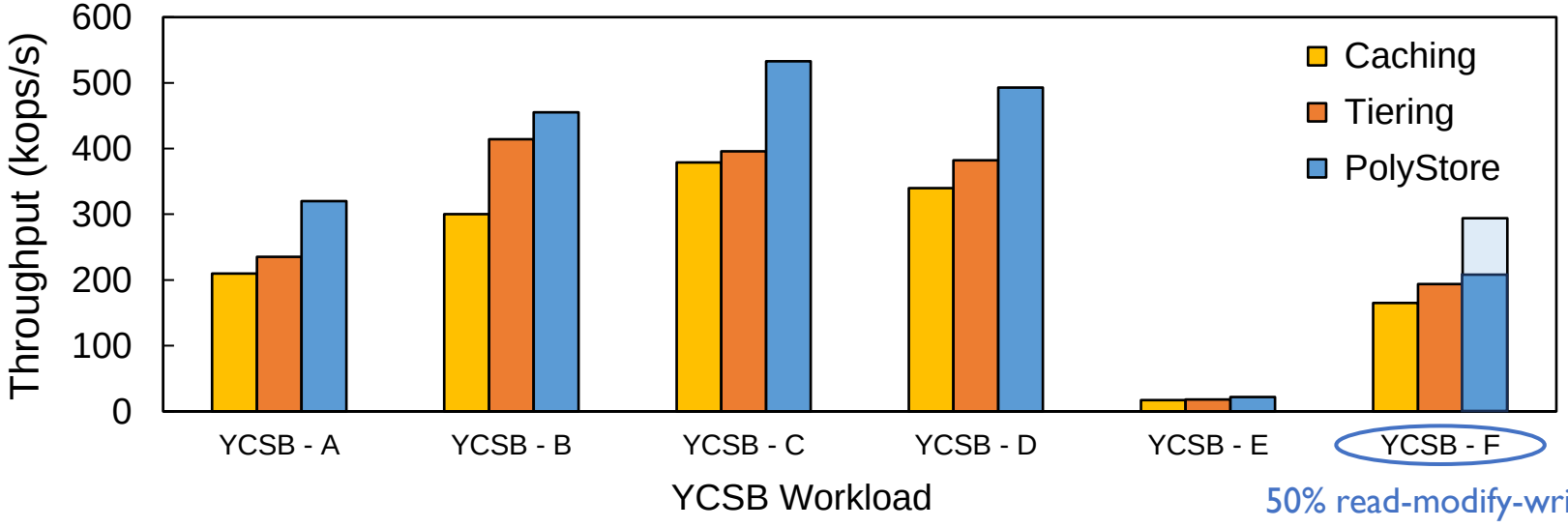
Enabling DRAM buffer cache for all approaches



- The OS page cache cannot handle device characteristics

Real applications: RocksDB with YCSB workload

Enabling DRAM buffer cache for all approaches



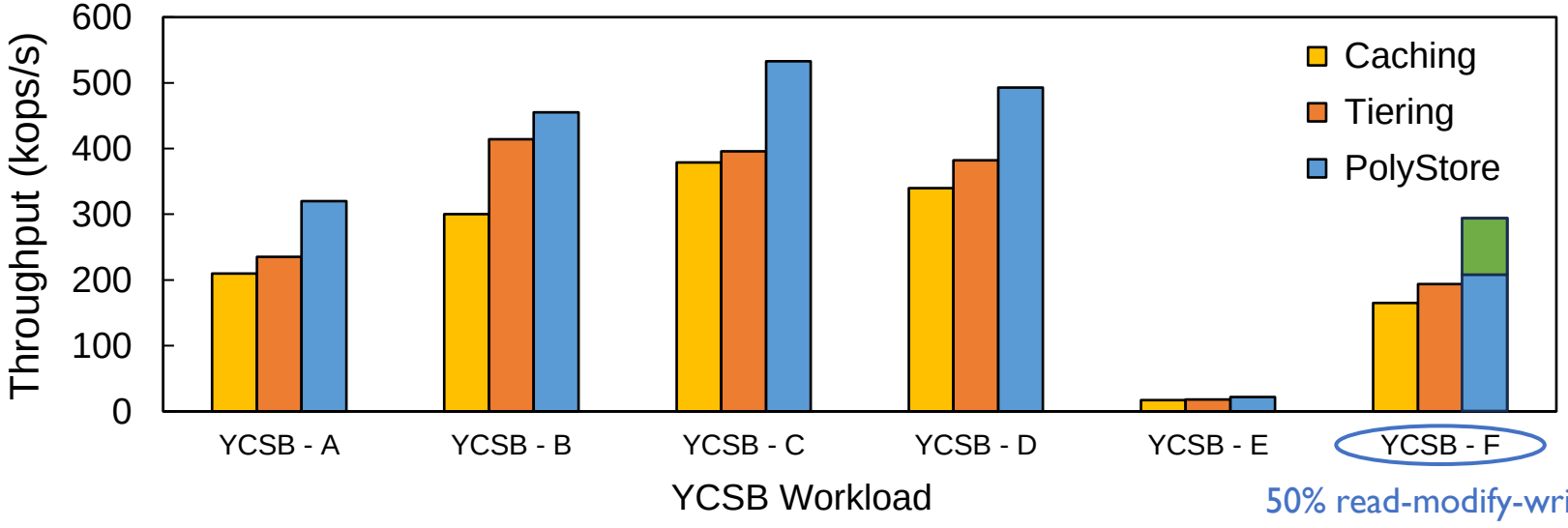
50% read-modify-write

- The OS page cache cannot handle device characteristics

Using OS page cache

Real applications: RocksDB with YCSB workload

Enabling DRAM buffer cache for all approaches

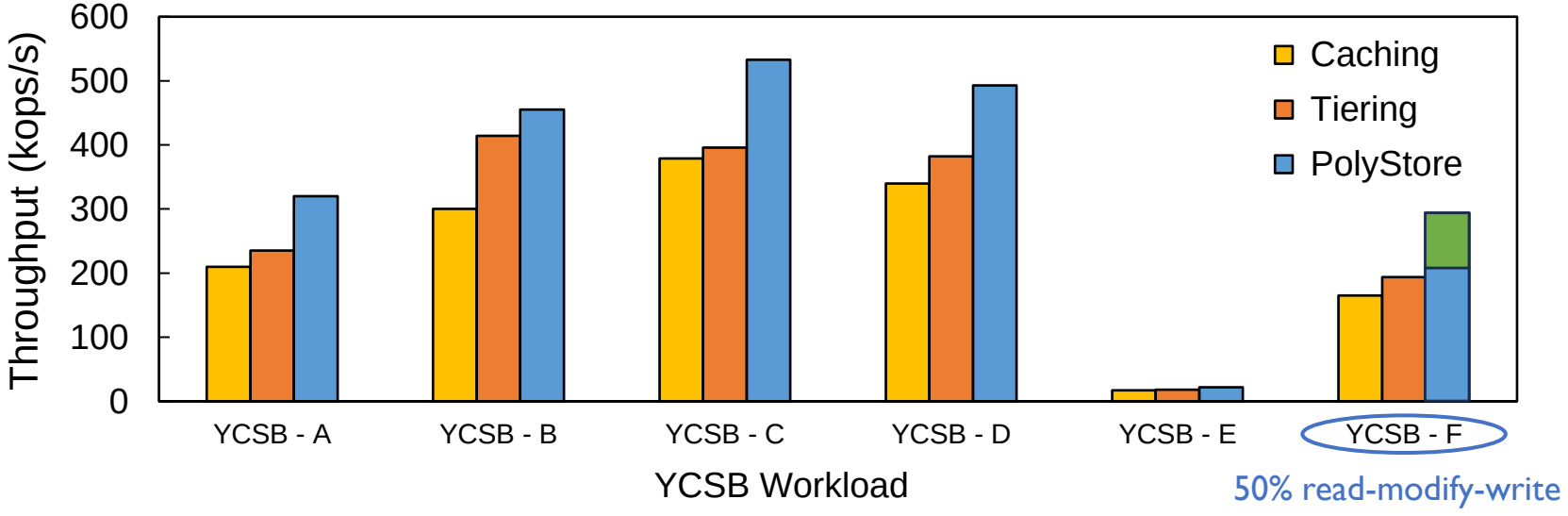


- The OS page cache cannot handle device characteristics

PolyStore DRAM buffer cache

Real applications: RocksDB with YCSB workload

Enabling DRAM buffer cache for all approaches

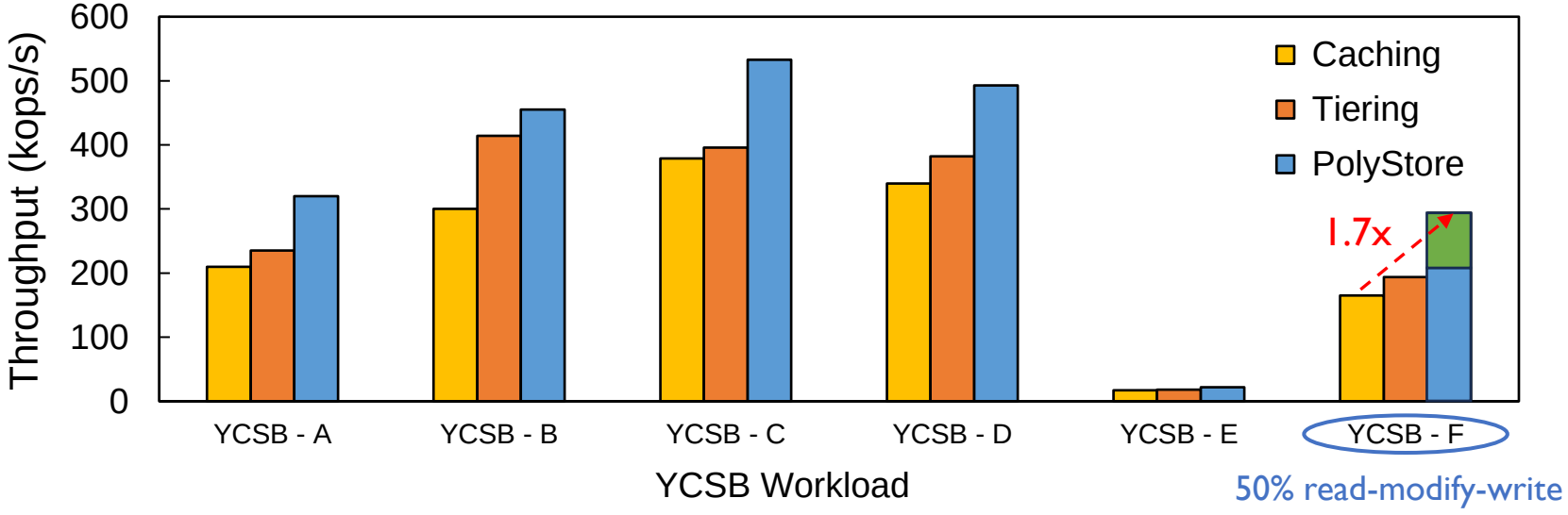


- The OS page cache cannot handle device characteristics
- PolyStore provide flexible data admission/eviction control

PolyStore DRAM buffer cache

Real applications: RocksDB with YCSB workload

Enabling DRAM buffer cache for all approaches



- The OS page cache cannot handle device characteristics
- PolyStore provide flexible data admission/eviction control

PolyStore DRAM buffer cache

Conclusion

Hierarchical design philosophy no longer suits bandwidth-intensive applications

PolyStore - a meta layer on top of mature device-optimized file systems

- Scaling device bandwidth horizontally with dynamic data placement
- DRAM buffering mechanism adapting to device characteristics

https://github.com/RutgersCSSystems/PolyStore

Thanks!

Project site