PMA: A Persistent Memory Allocator with High Efficiency and Crash Consistency Guarantee

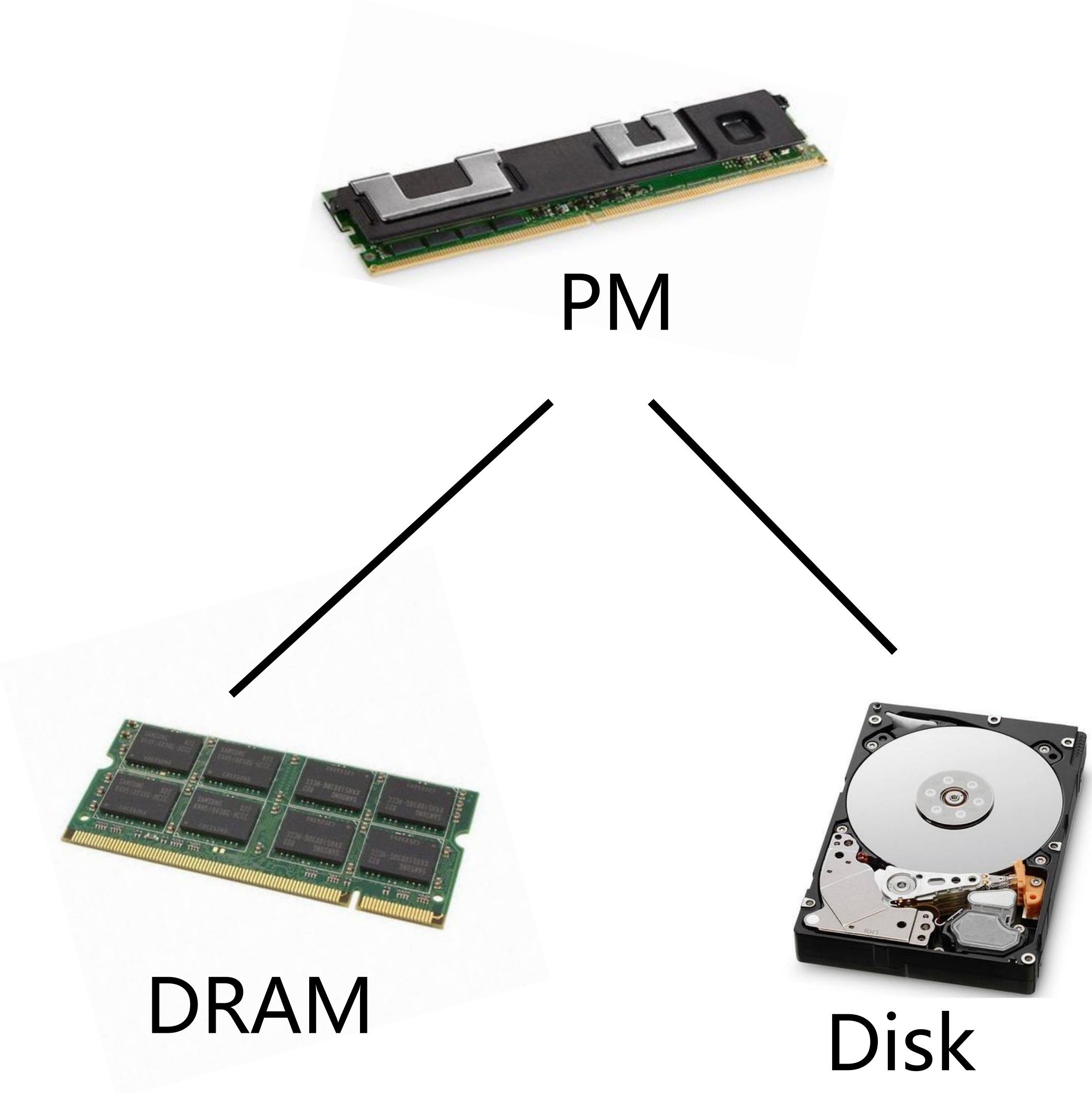
Xiangyu Xiang, Yu Hua, Hao Xu

Huazhong University of Science and Technology

ICCD 2023

Persistent Memory

- ➢Persistent memory (PM)
  - ✓ Persistency
  - ✓ Byte-addressability
  - ✓ Near-DRAM latency
  - ✓ Large capacity

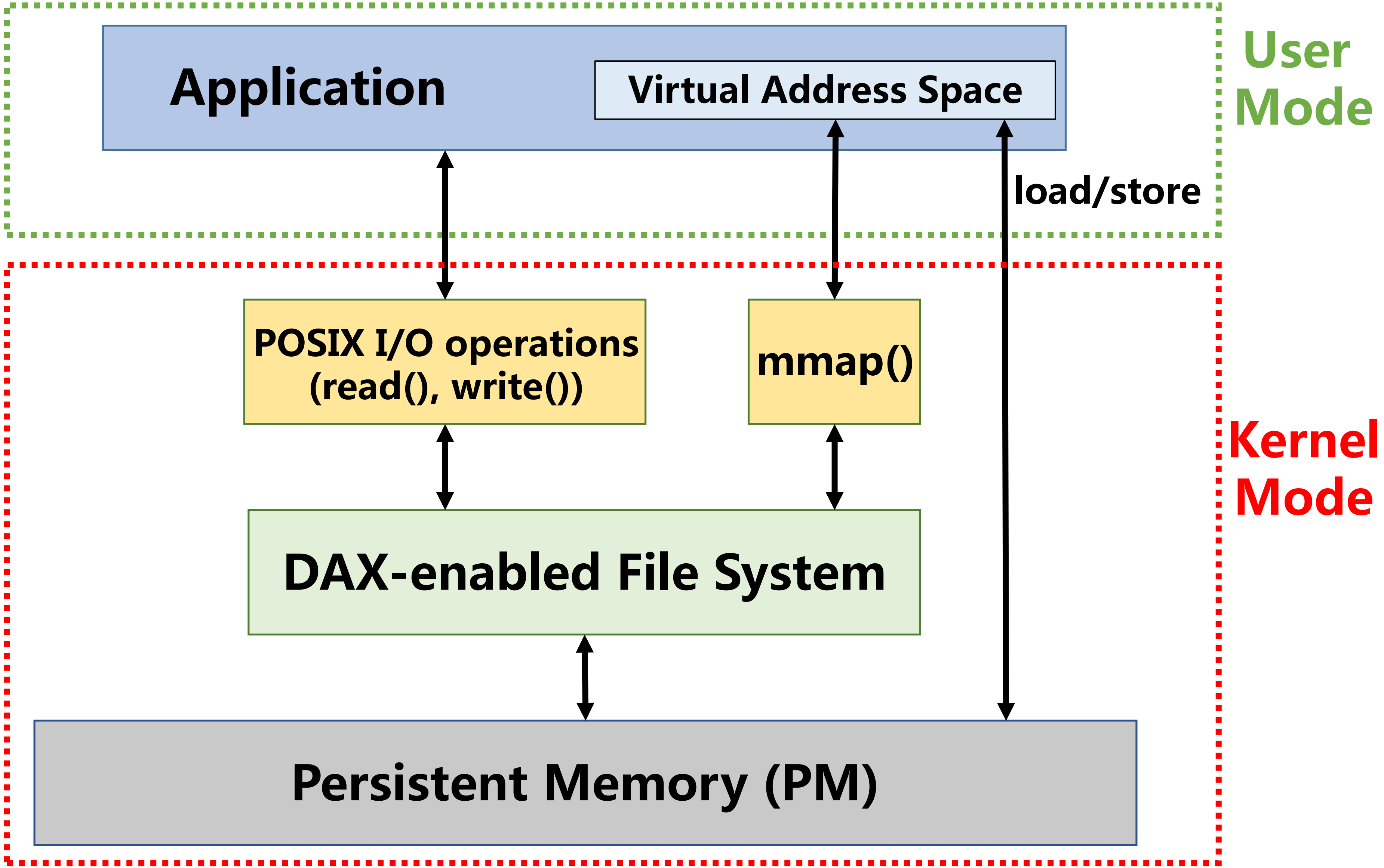


PM

DRAM

Disk

Direct Access (DAX)



Memory Allocator

➢Functionality

- Dynamically allocate/reclaim memory space for working threads
- Reduce memory fragmentations
- Avoid false sharing among multiple working threads

➢Existing DRAM allocator

- Hoard[1]
- Jemalloc[2]
- Tcmalloc[3]

[1] E. D. Berger, K. S. McKinley, R. D. Blumofe and P. R. Wilson, “Hoard: A Scalable Memory Allocator for Multithreaded Applications,” in ASPLOS, 2000

[2] J. Evans, “A Scalable Concurrent malloc(3) Implementation for FreeBSD,” in BSDCan, 2006.

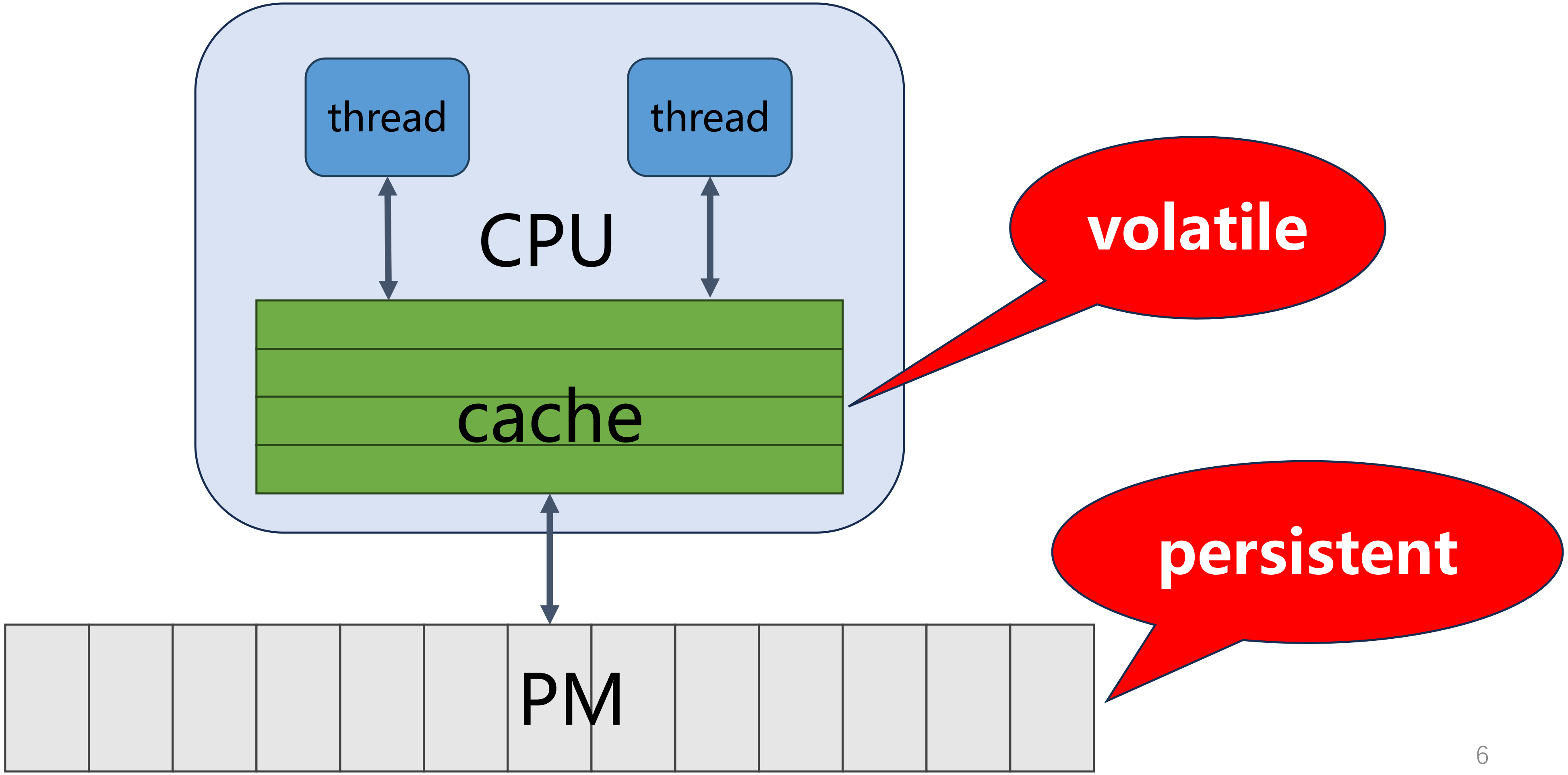
[3] Google. TCMalloc: Thread-Caching Malloc. https://google.github.io/tcmalloc/

Memory Allocator

Why can’t DRAM allocators be employed by PM?

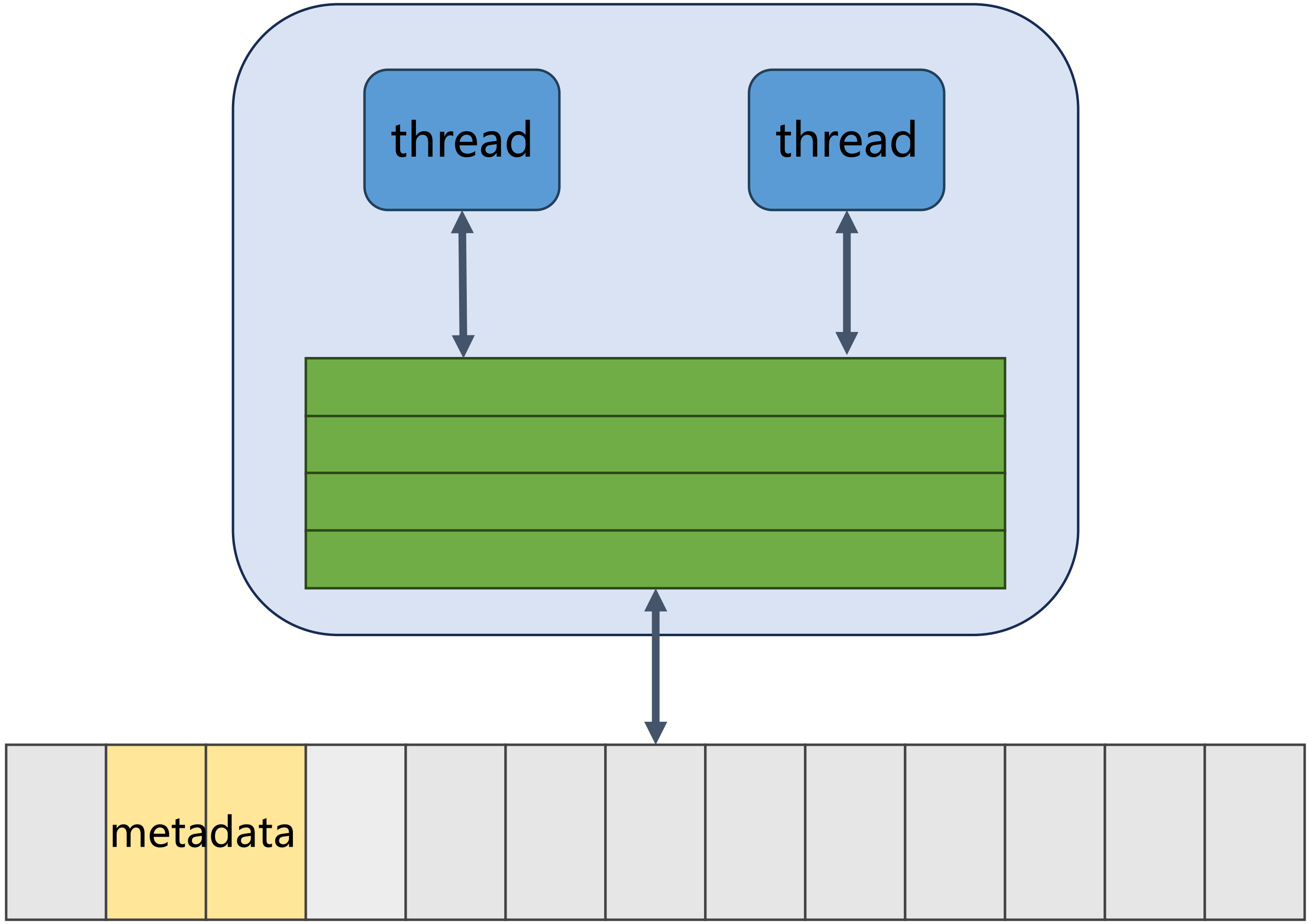
Memory Allocator

Why can’t DRAM allocators be employed by PM?



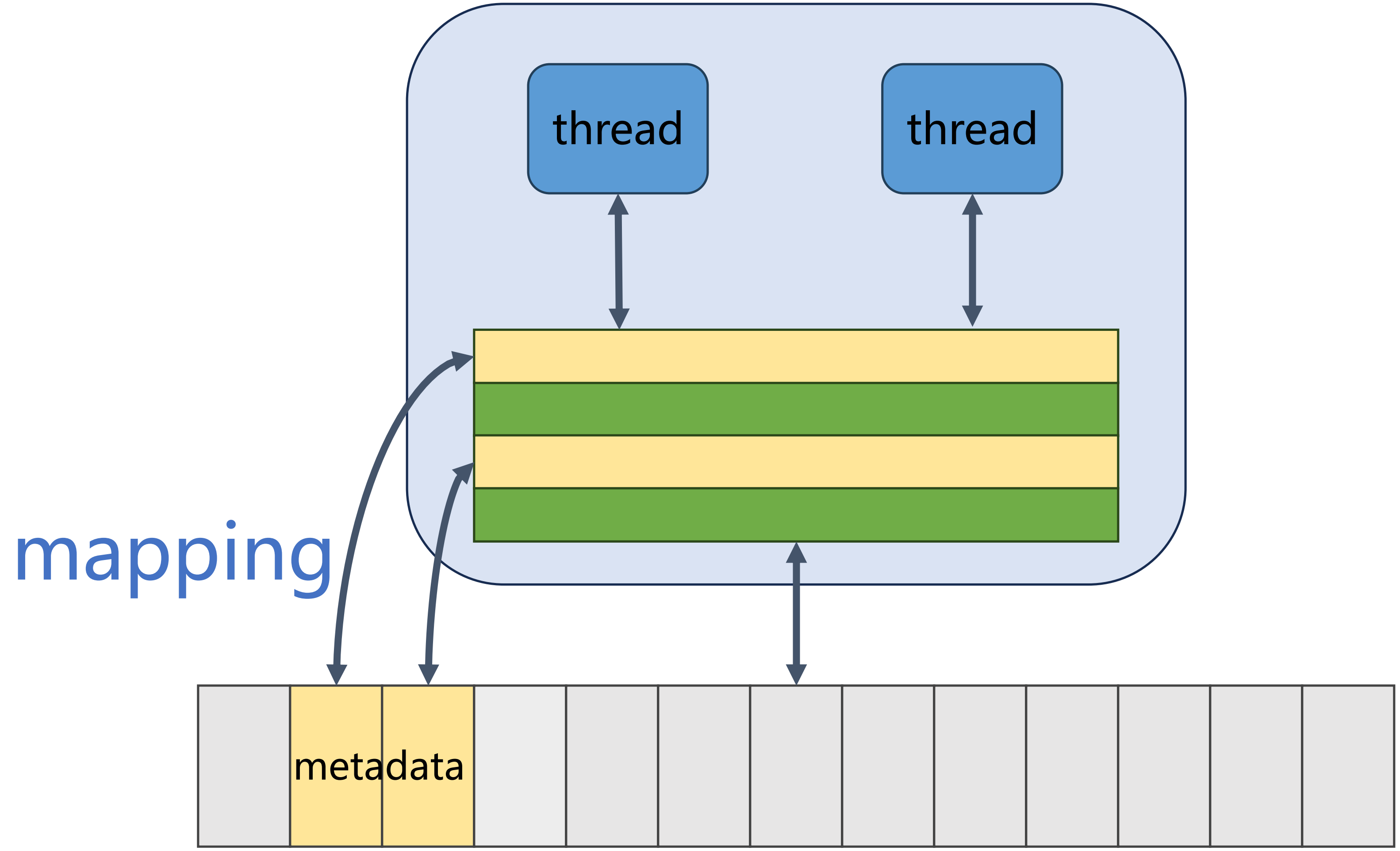
Memory Allocator

Why can’t DRAM allocators be employed by PM?



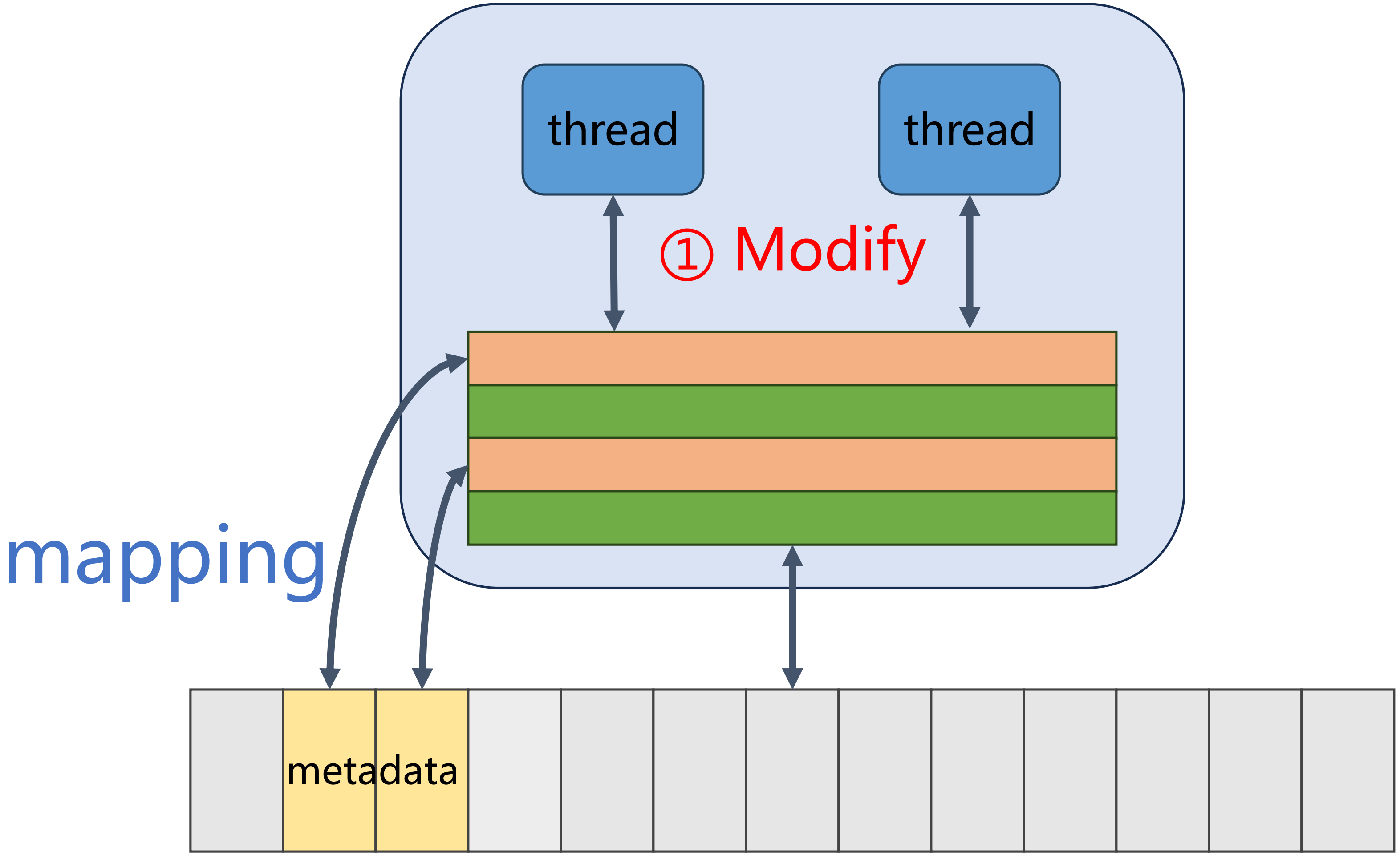
Memory Allocator

Why can’t DRAM allocators be employed by PM?



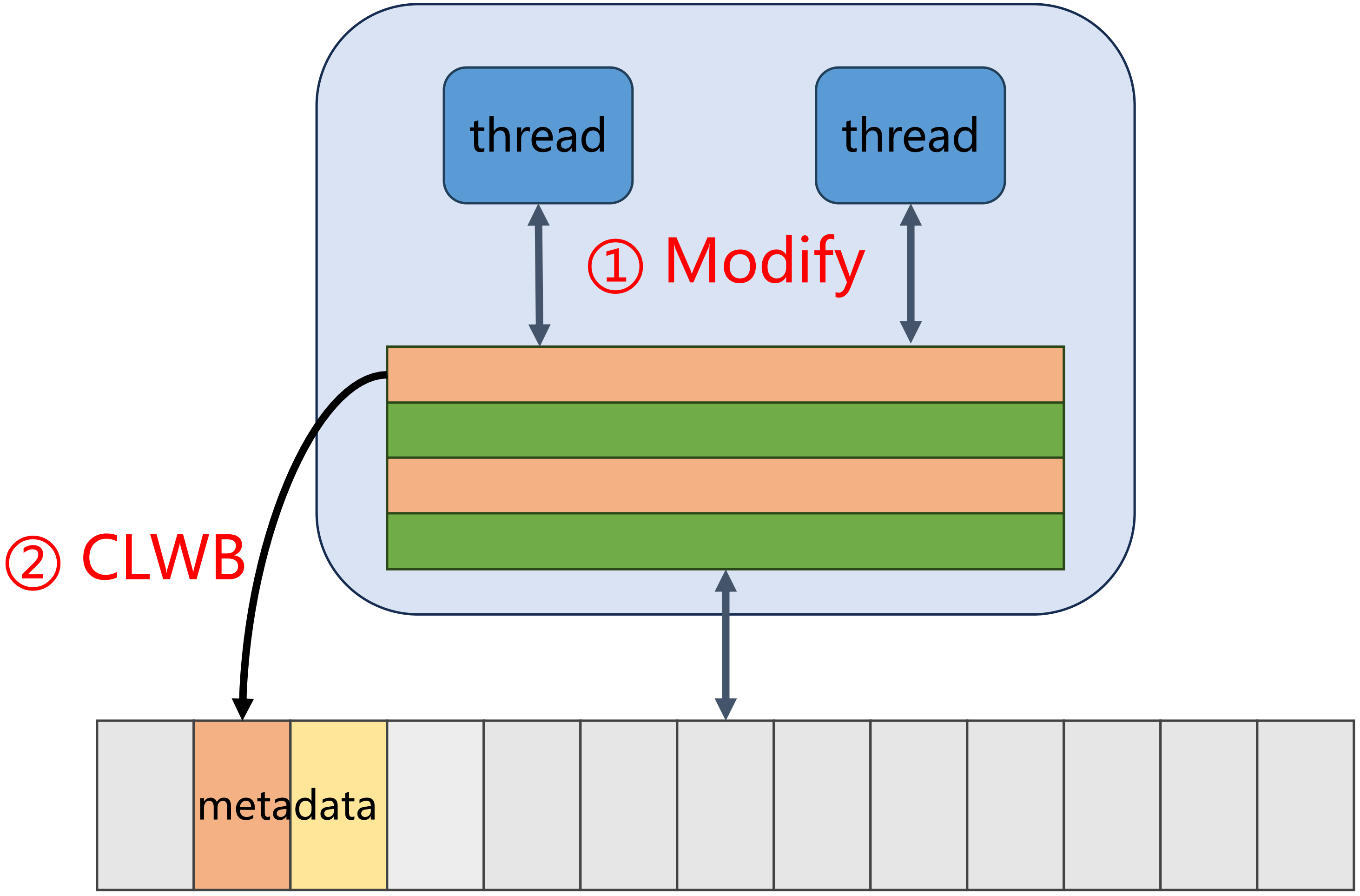
Memory Allocator

Why can’t DRAM allocators be employed by PM?



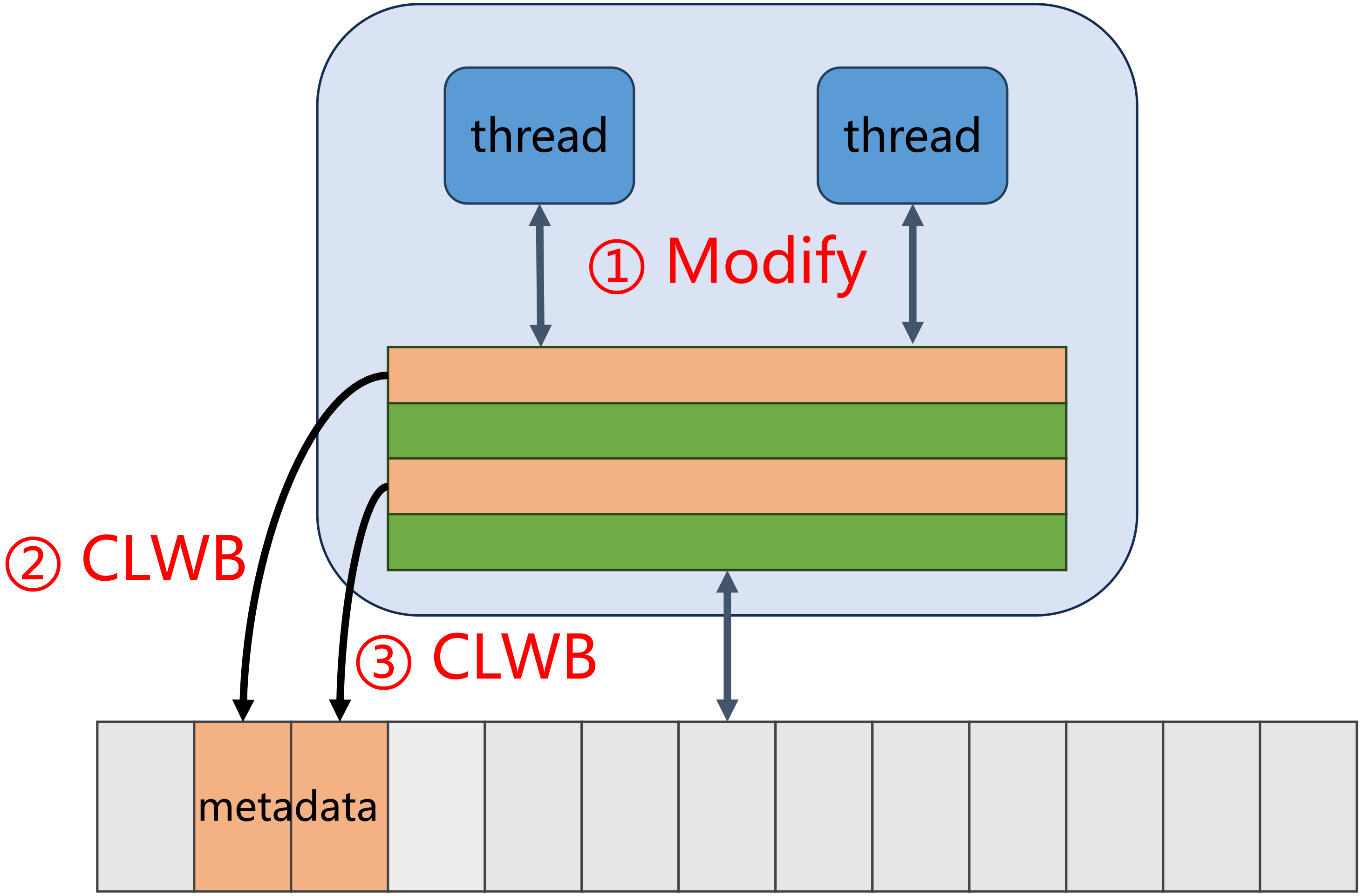
Memory Allocator

Why can’t DRAM allocators be employed by PM?



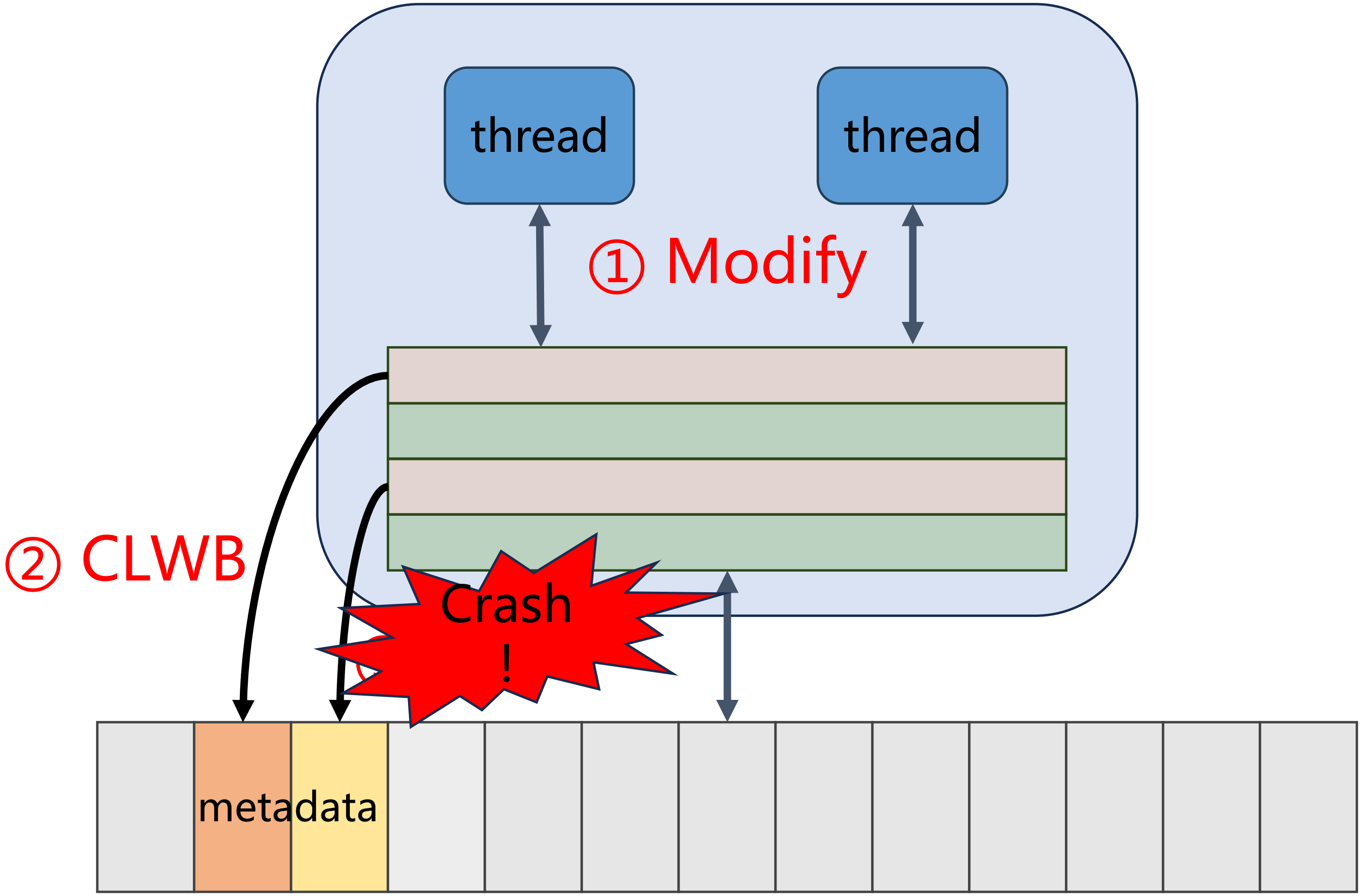
Memory Allocator

Why can’t DRAM allocators be employed by PM?



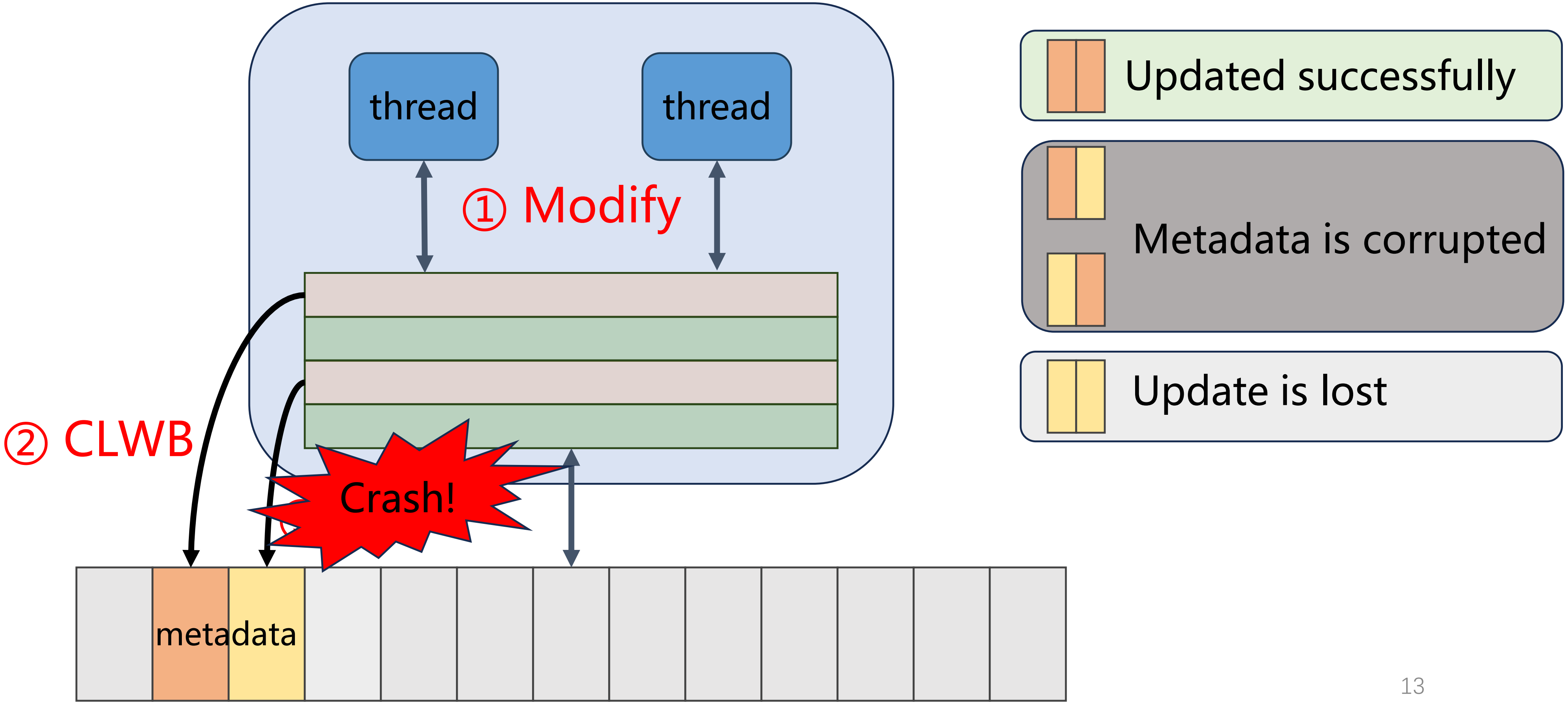
Memory Allocator

Why can’t DRAM allocators be employed by PM?

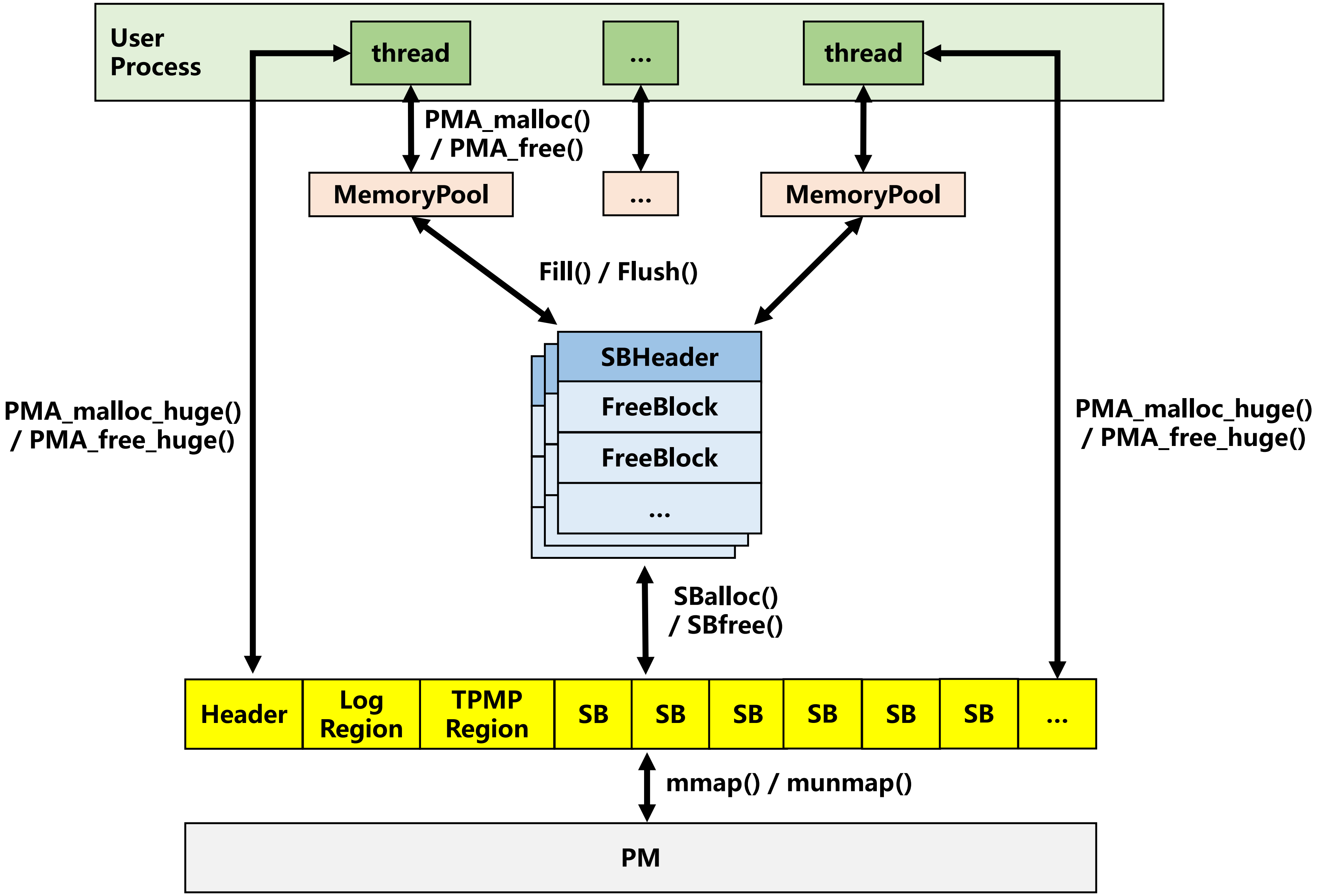


Memory Allocator

Why can’t DRAM allocators be employed by PM?



PMA: Overview



PMA: Size Class

- Predefine some standard size classes
    - E.g. {8, 10, 12, 14, 16, 20, 24, 28, 32, …, 8192}

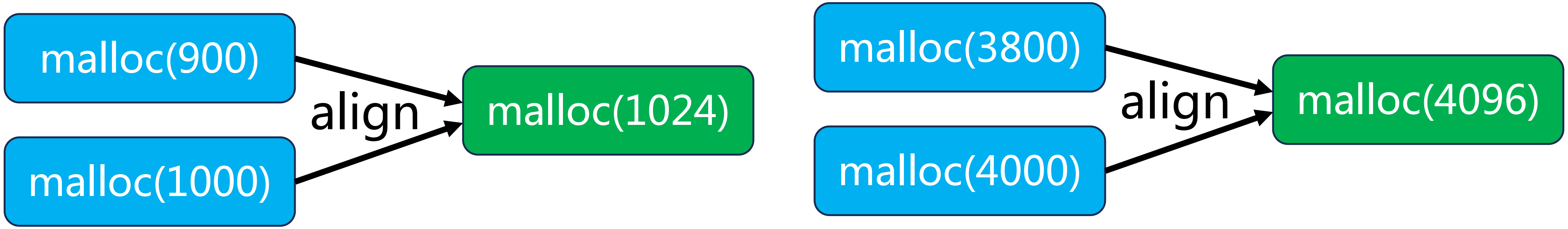
PMA: Size Class

- Predefine some standard size classes
    - E.g. {8, 10, 12, 14, 16, 20, 24, 28, 32, …, 8192}
- Align the allocation size up to the nearest size class

PMA: Size Class

- Predefine some standard size classes
  - E.g. {8, 10, 12, 14, 16, 20, 24, 28, 32, …, 8192}

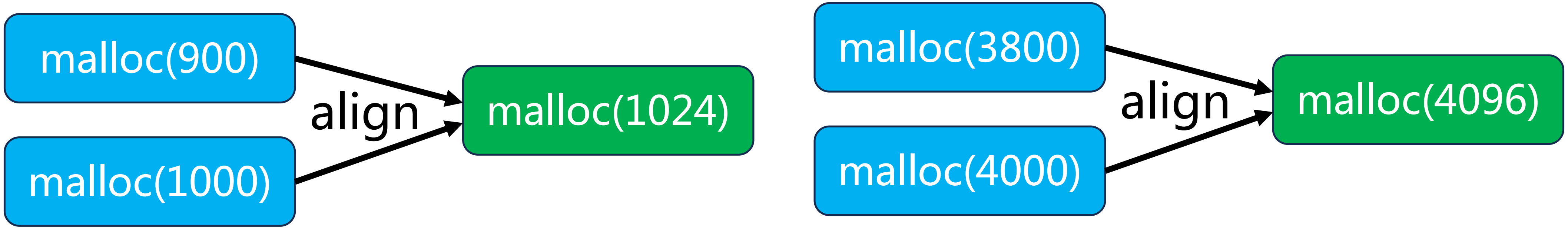
- Align the allocation size up to the nearest size class



PMA: Size Class

- Predefine some standard size classes
  - E.g. {8, 10, 12, 14, 16, 20, 24, 28, 32, …, 8192}

- Align the allocation size up to the nearest size class



malloc(900)

malloc(1000)

align

malloc(1024)

malloc(3800)

malloc(4000)

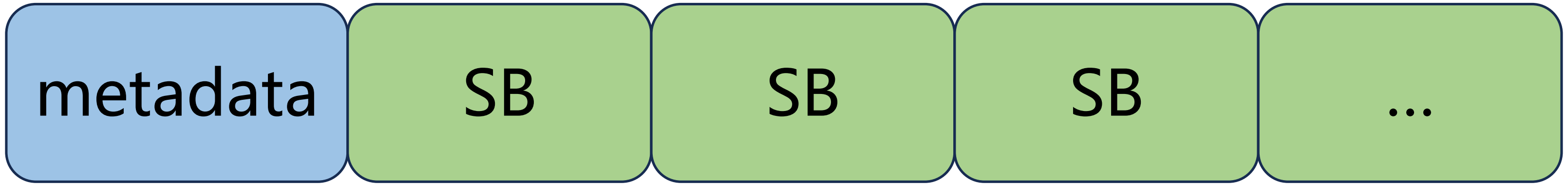
align

malloc(4096)

- Fragmentation rate ≤ 20% [1]

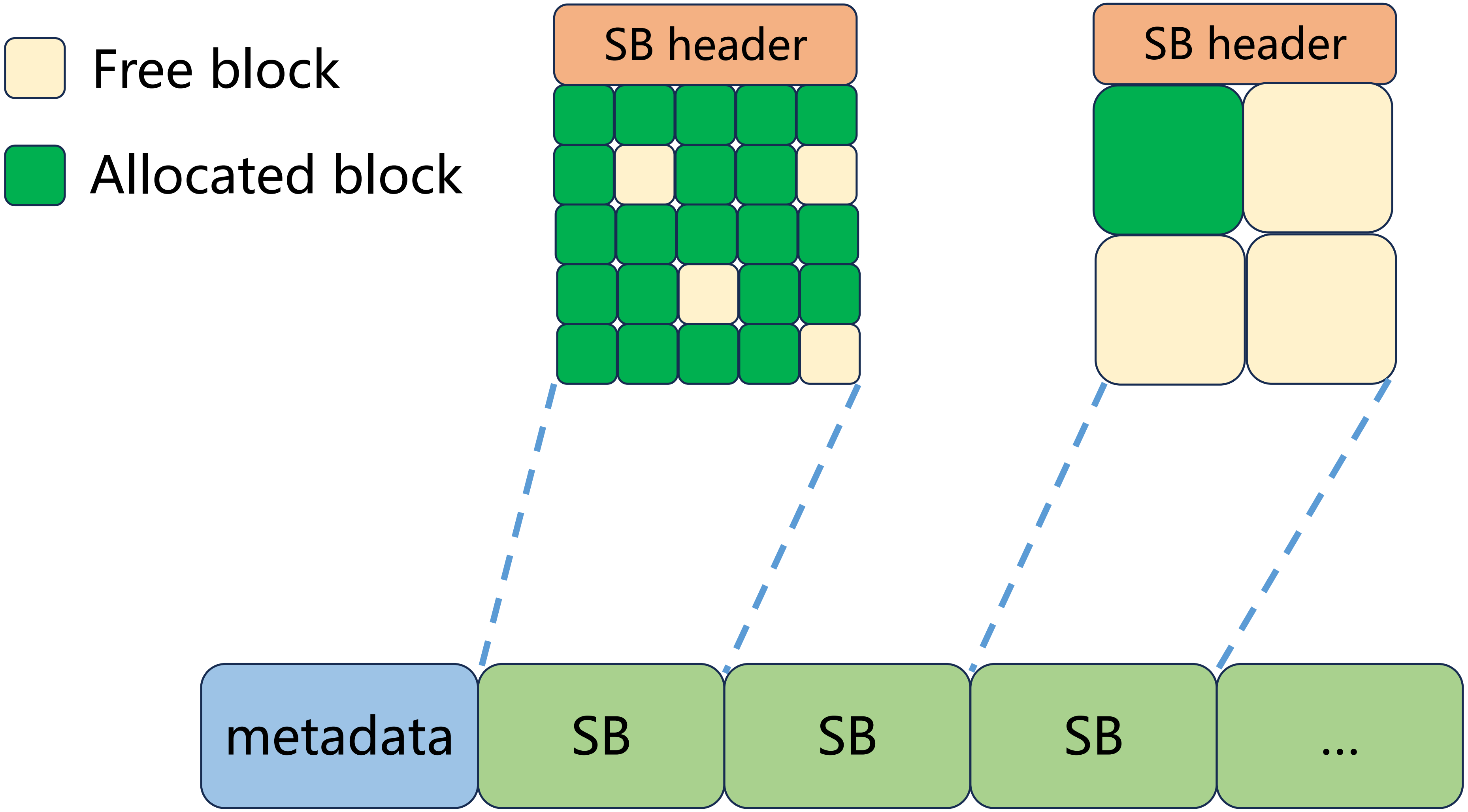
[1] R. Leite and R. Rocha, “A Modern and Competitive Lock-Free Dynamic Memory Allocator,” in VECPAR, 2018

PMA: Super Block



metadata SB SB SB …

PMA: Super Block



□ Free block

🟩 Allocated block

SB header

SB header

metadata

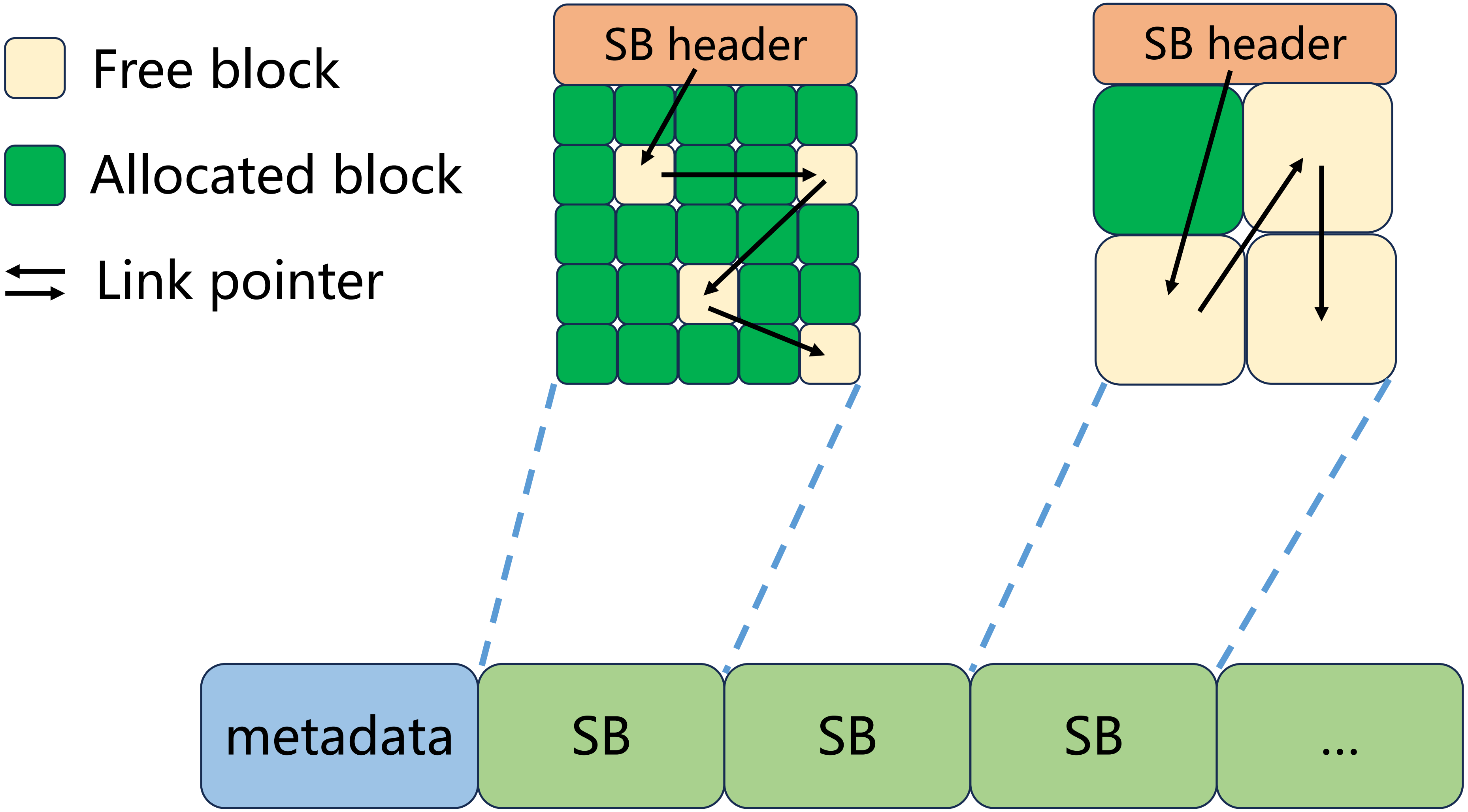
SB

SB

SB

...

PMA: Super Block



□ Free block

🟩 Allocated block

⇆ Link pointer

SB header

SB header

metadata

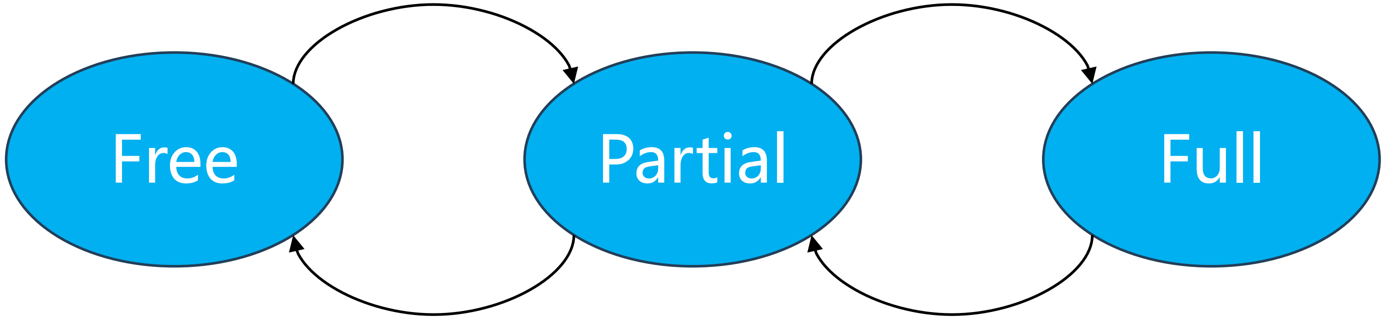
SB

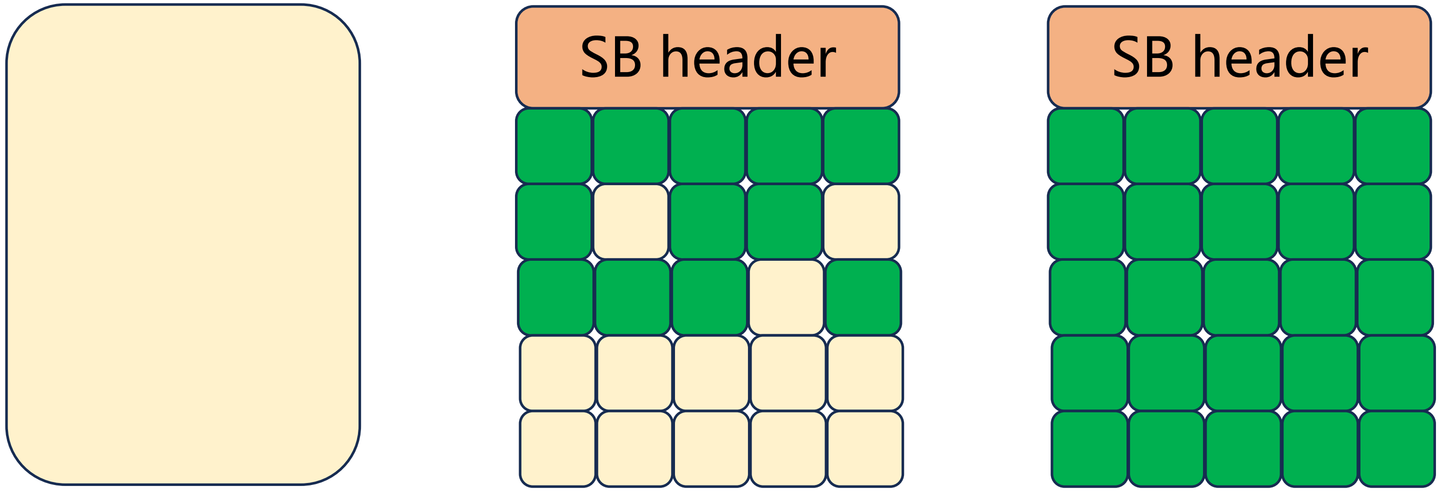
SB

SB

…

PMA: Super Block

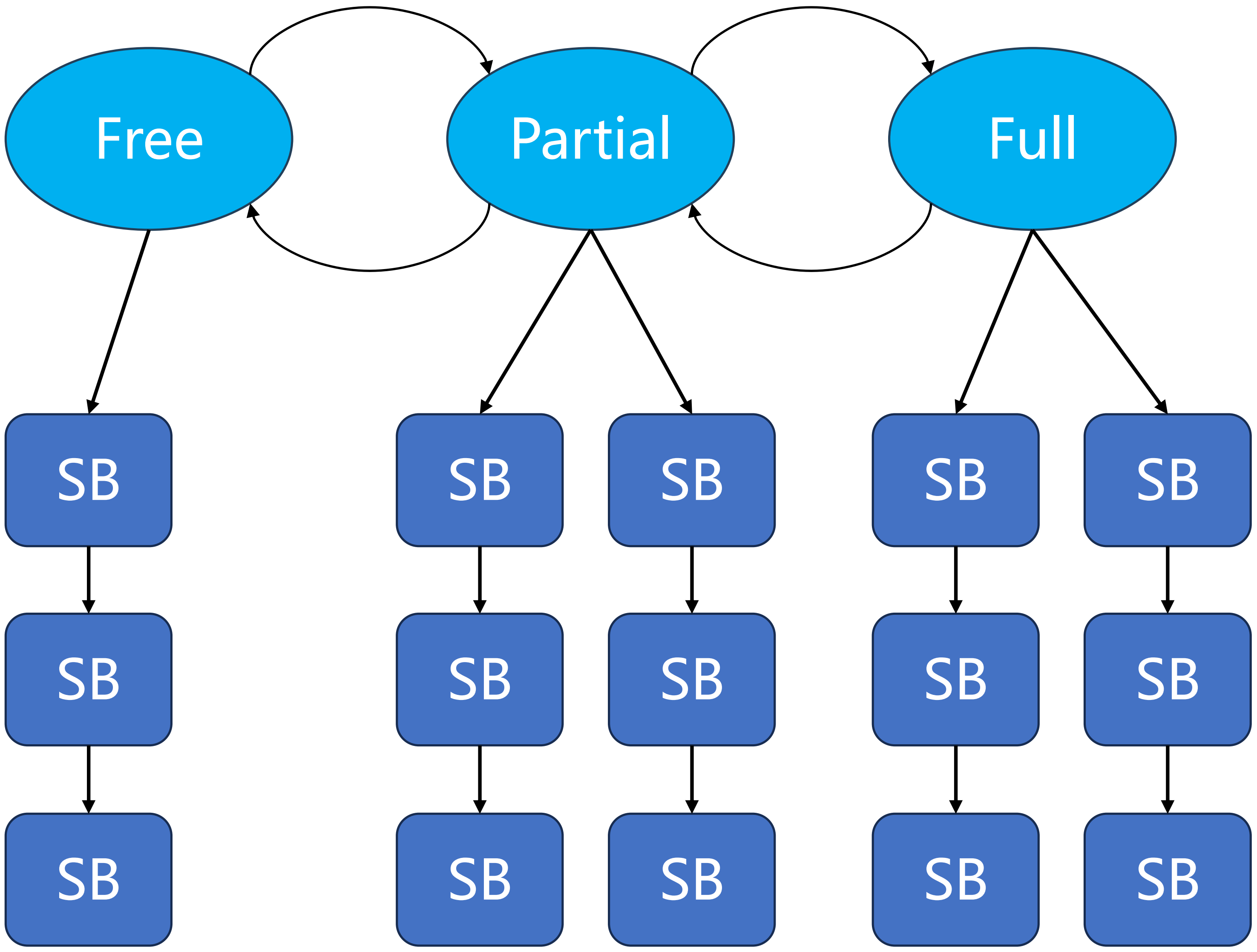




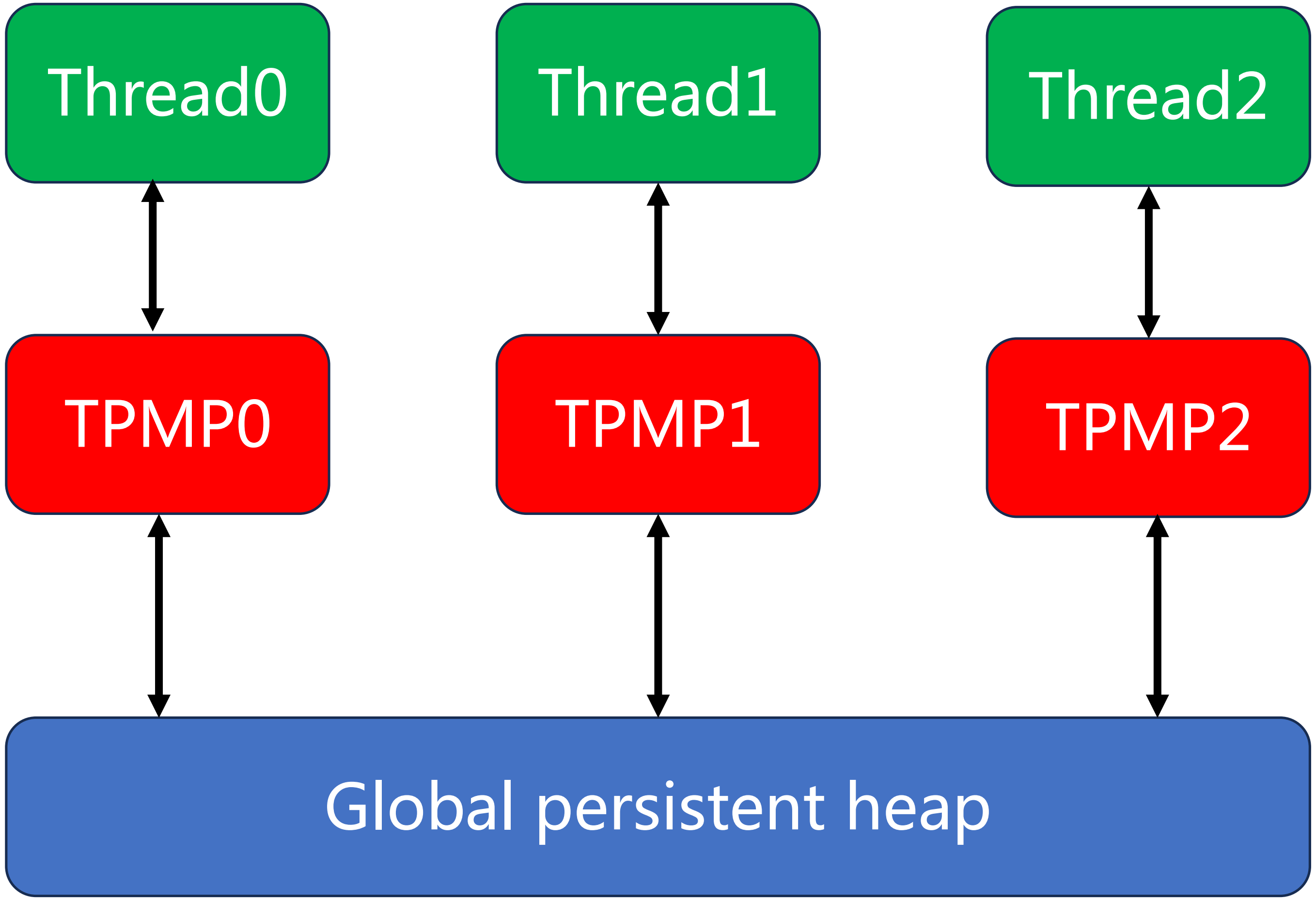
□ Free block

🟩 Allocated block

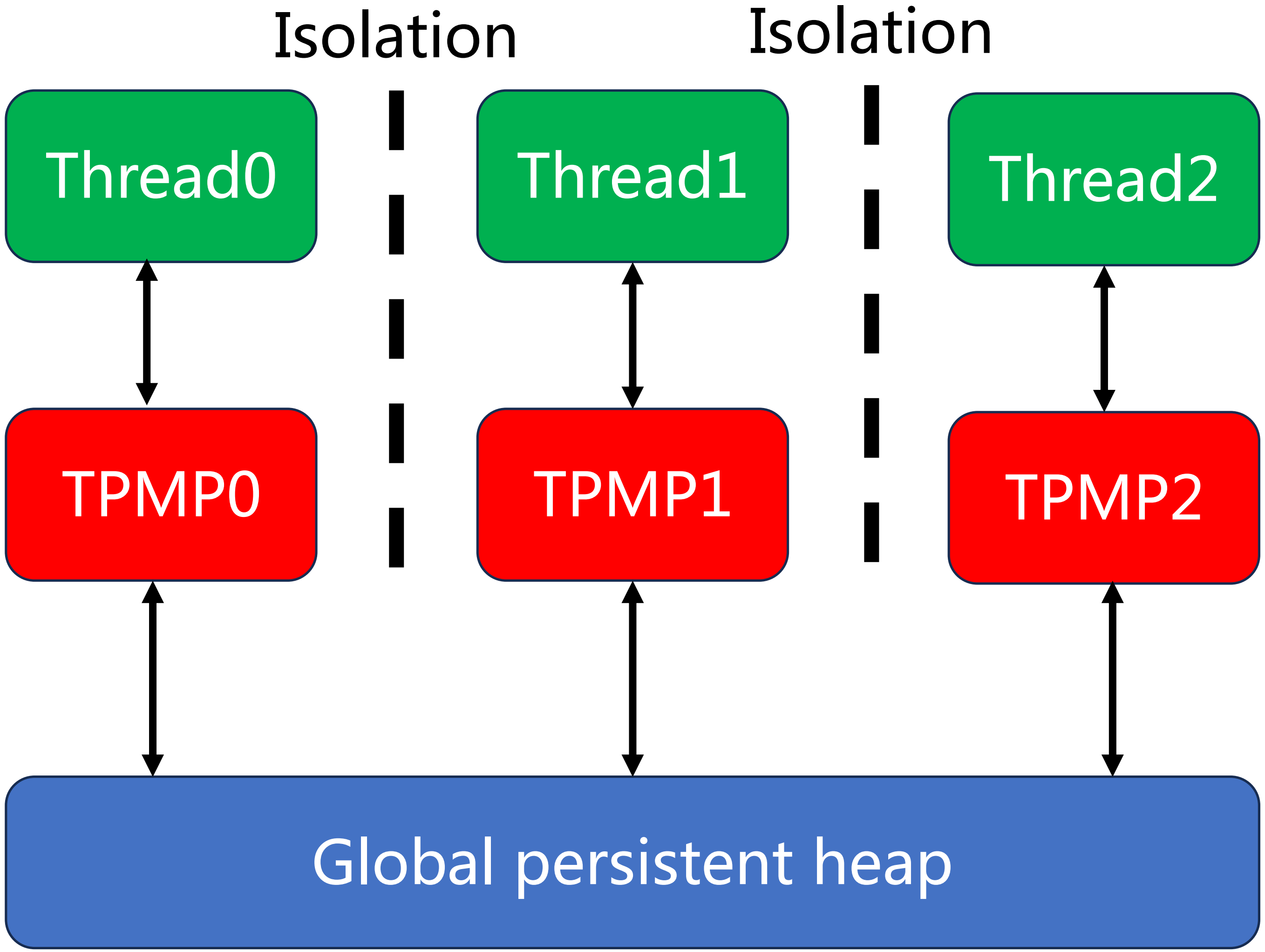
PMA: Super Block



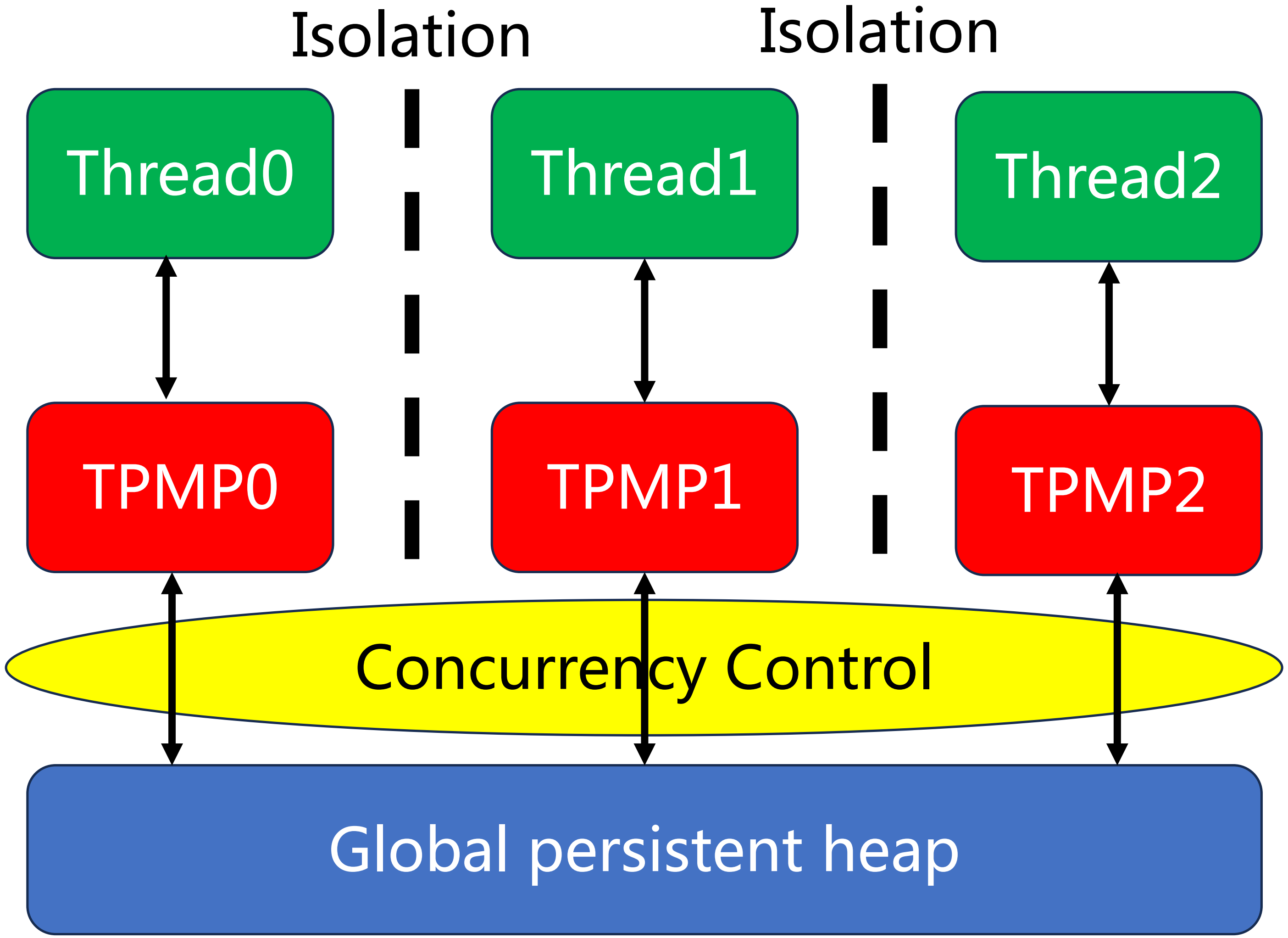
PMA: Thread-Private Memory Pool



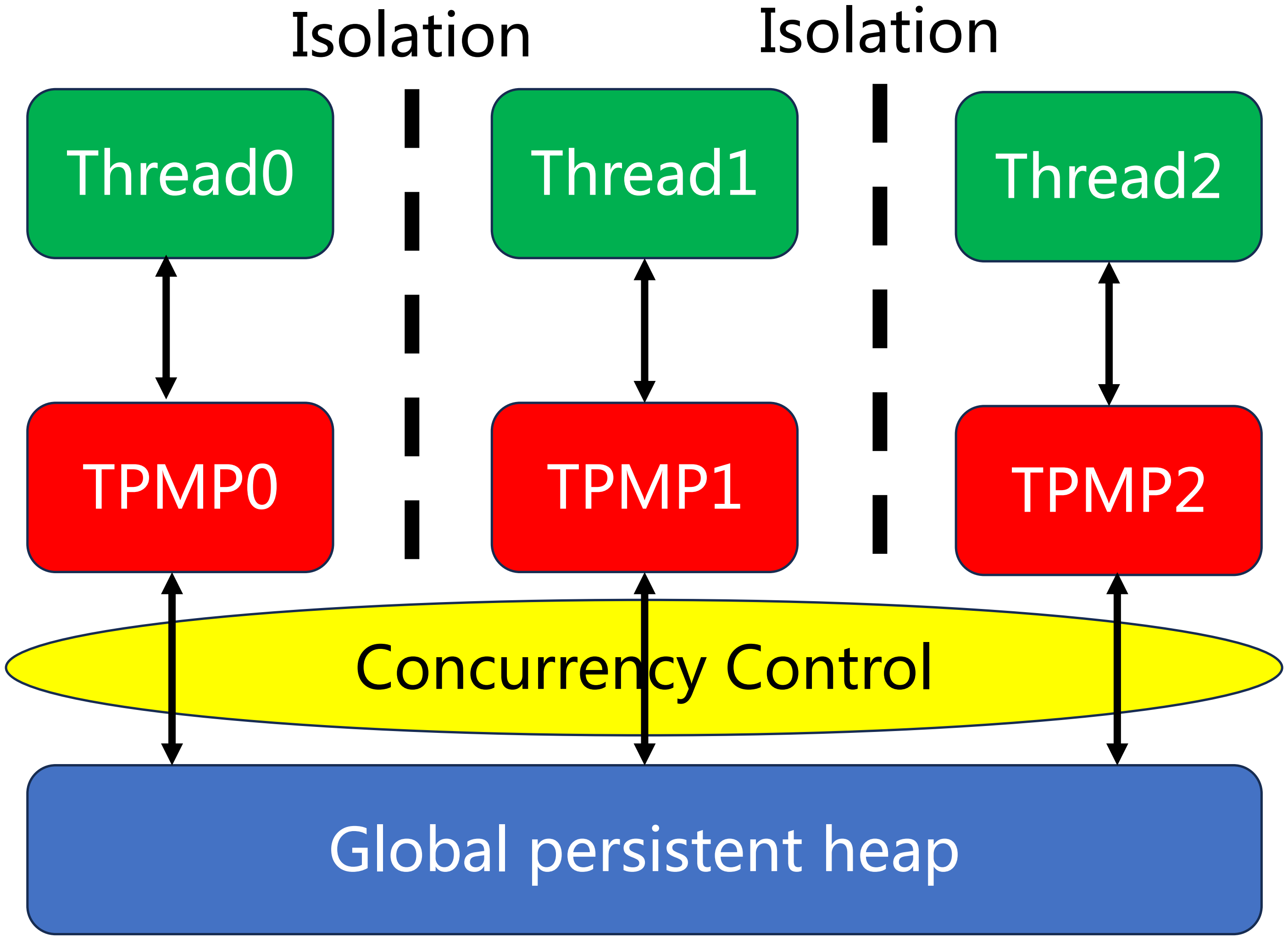
PMA: Thread-Private Memory Pool



PMA: Thread-Private Memory Pool



PMA: Thread-Private Memory Pool

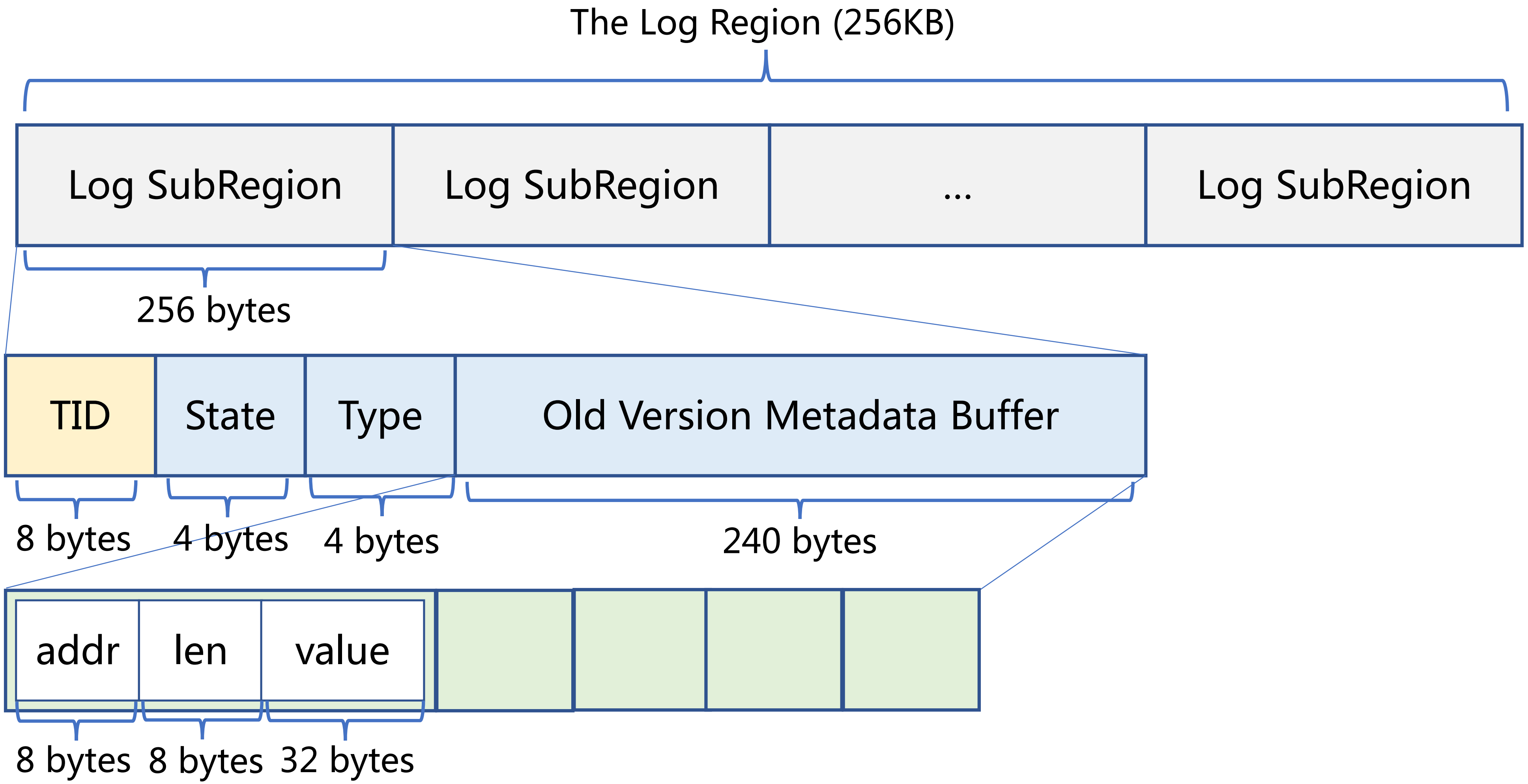


Over 99% of allocation requests can be served by the TPMP!

PMA: Crash Consistency

- ➢Transactions in PMA
  - ✓ Moving a SB from one list to another
  - ✓ Filling a TPMP with a SB or vice versa
  - ✓ Allocating/Reclaiming a memory block from a TPMP

PMA: Crash Consistency



Performance Evaluation

➢ Testbed

- ✓ 2 × Intel Xeon Gold 6230R @2.10GHz (26 cores)
- ✓ 6 × 128GB Intel Optane DC PMM 100 Series

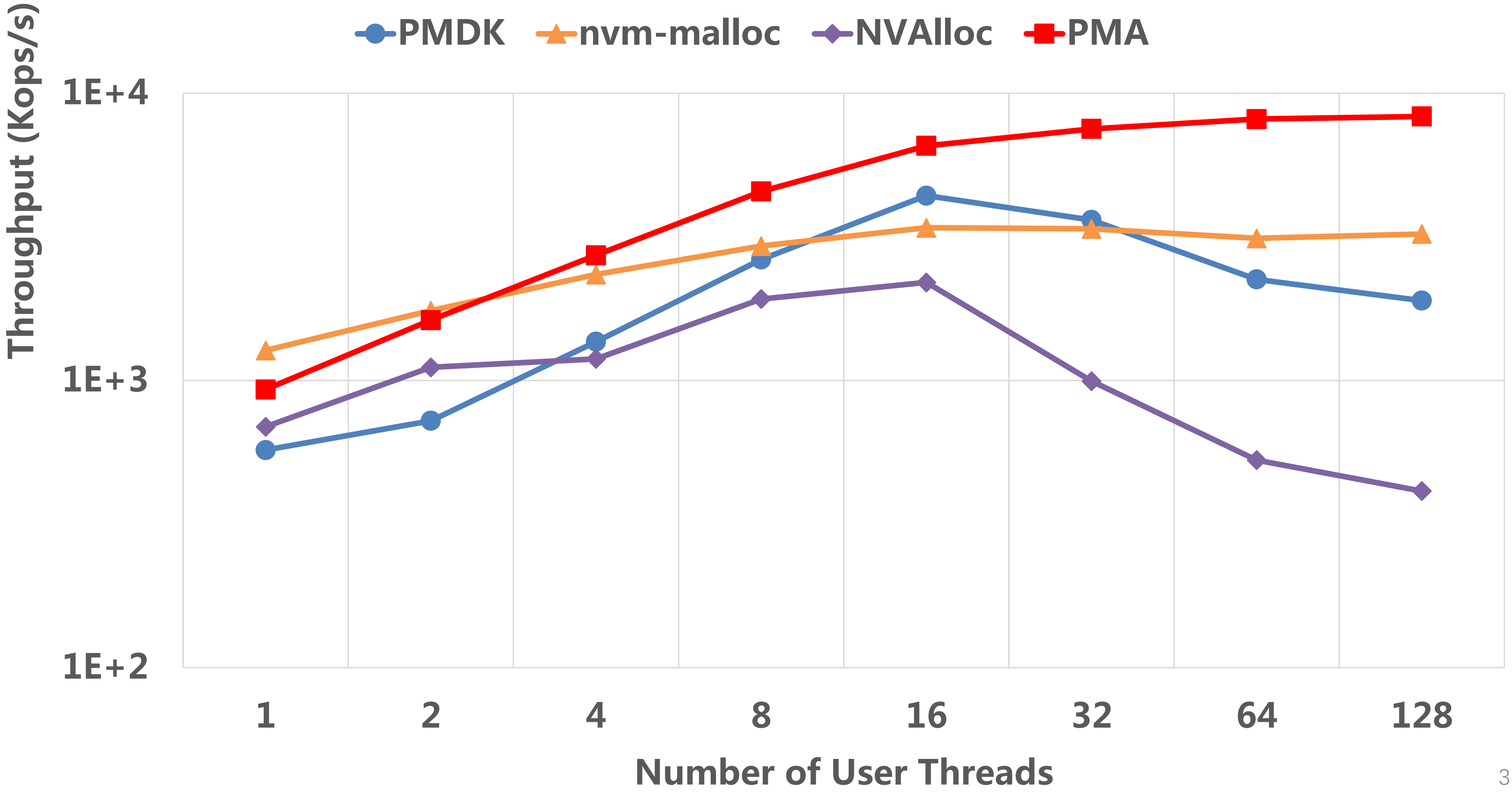
➢ Workload

- ✓ Allocating and reclaiming 10^6 memory blocks of 64 bytes
- ✓ Repeating for 10^3 iterations

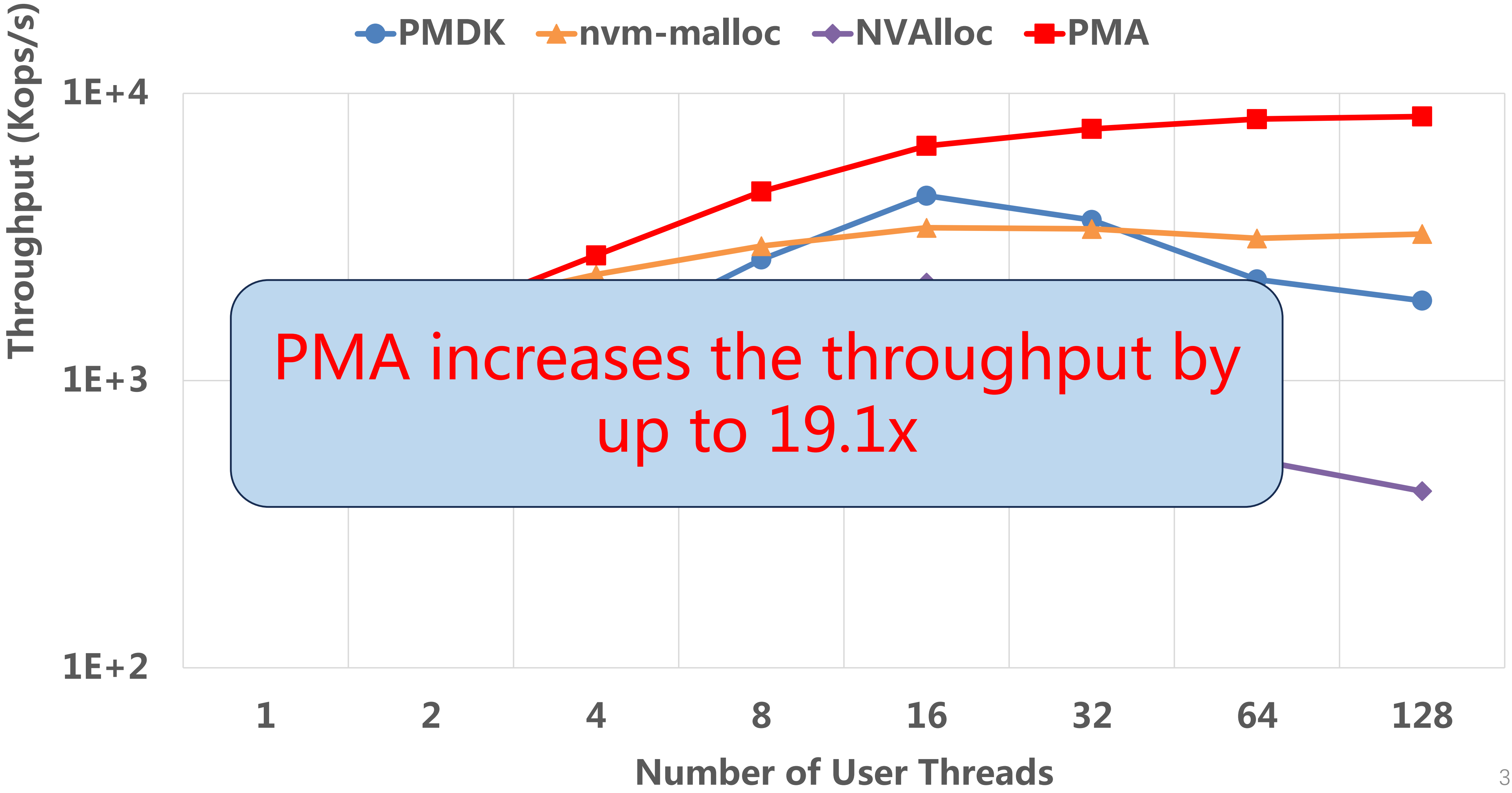
➢ Competitors

✓	PMDK	[Intel]
✓	nvm_malloc	[VLDB 2015]
✓	NVAlloc-log	[ASPLOS 2022]

Performance Evaluation

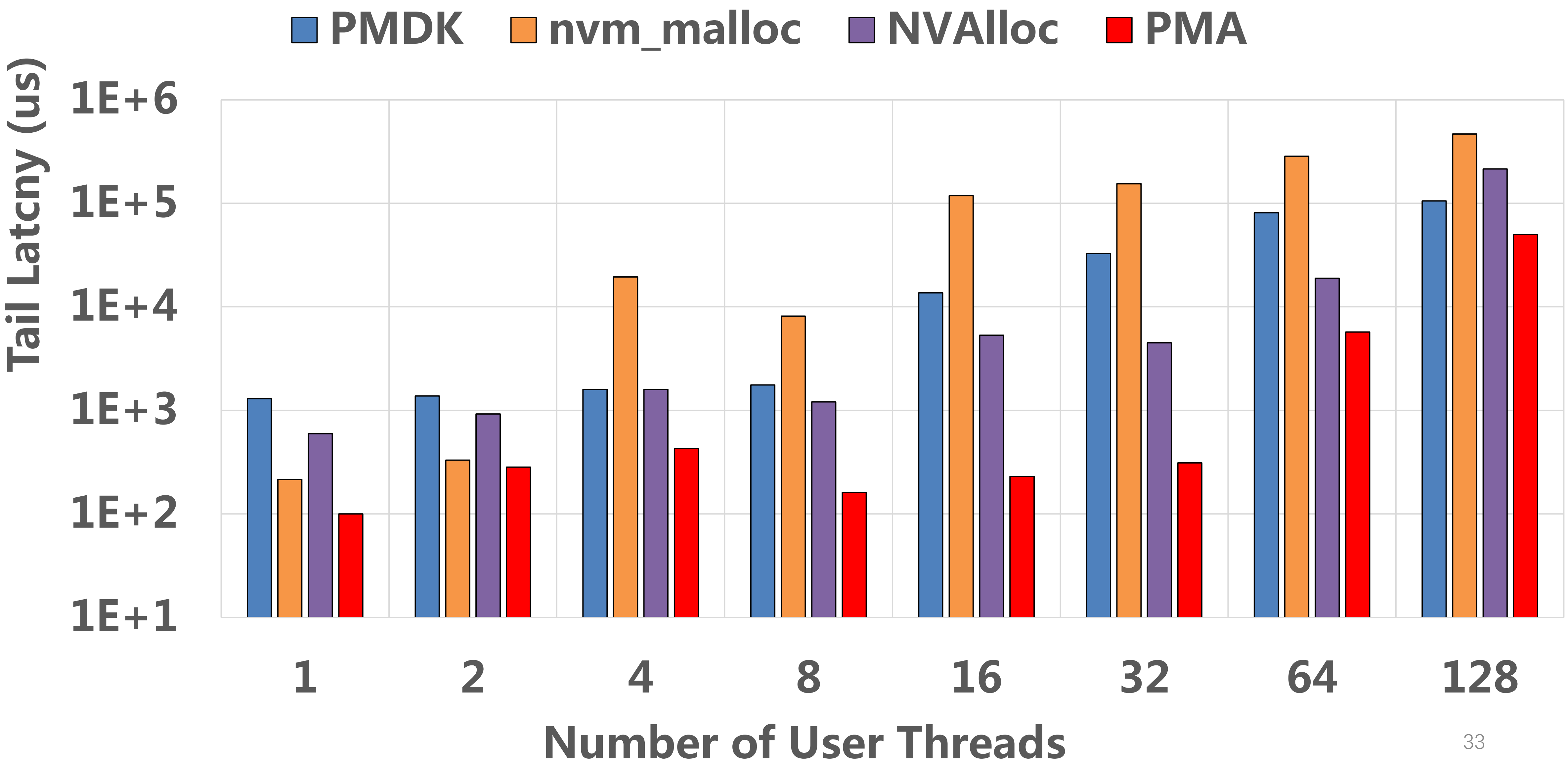


Performance Evaluation

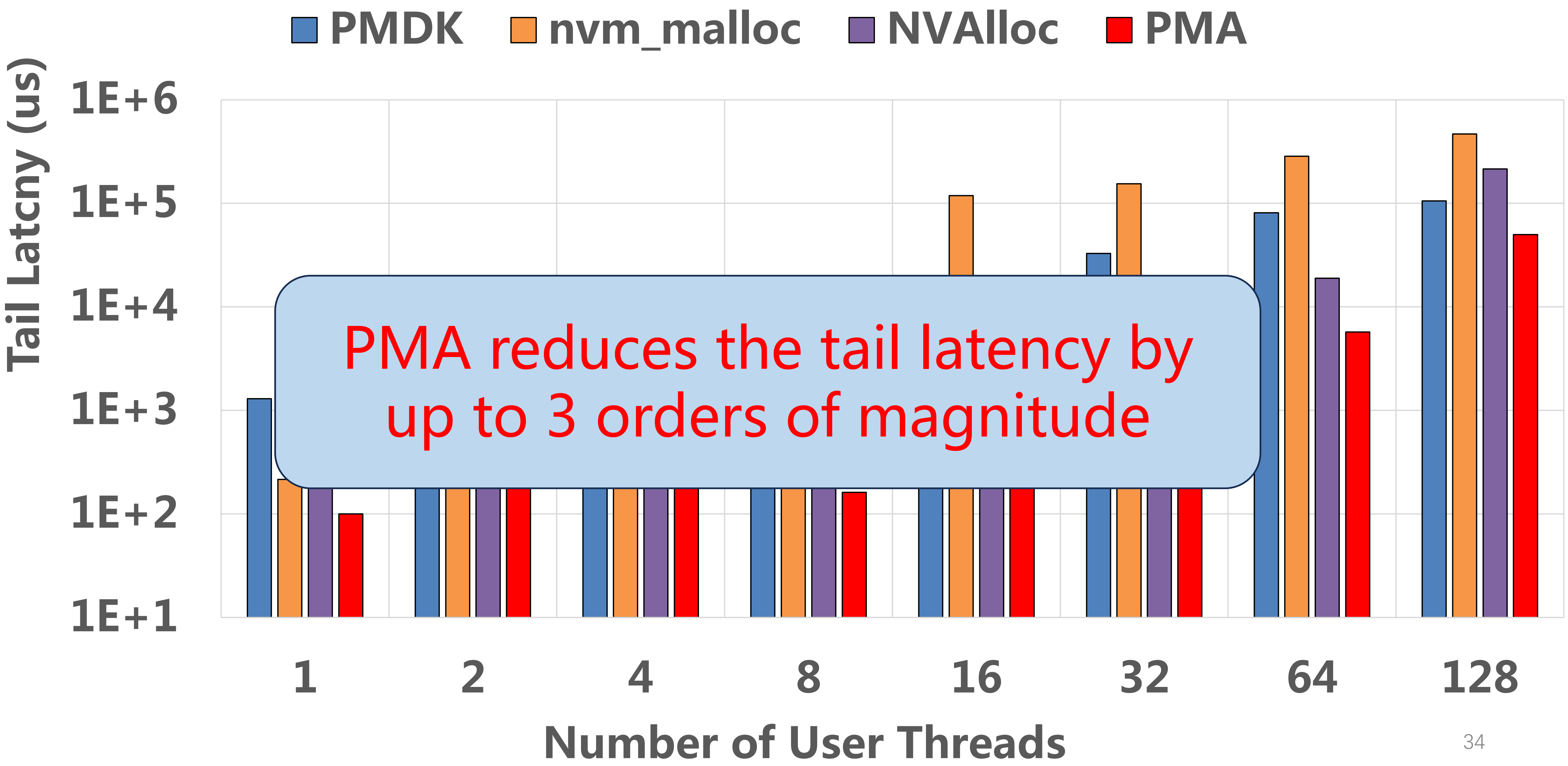


PMA increases the throughput by up to 19.1x

Performance Evaluation

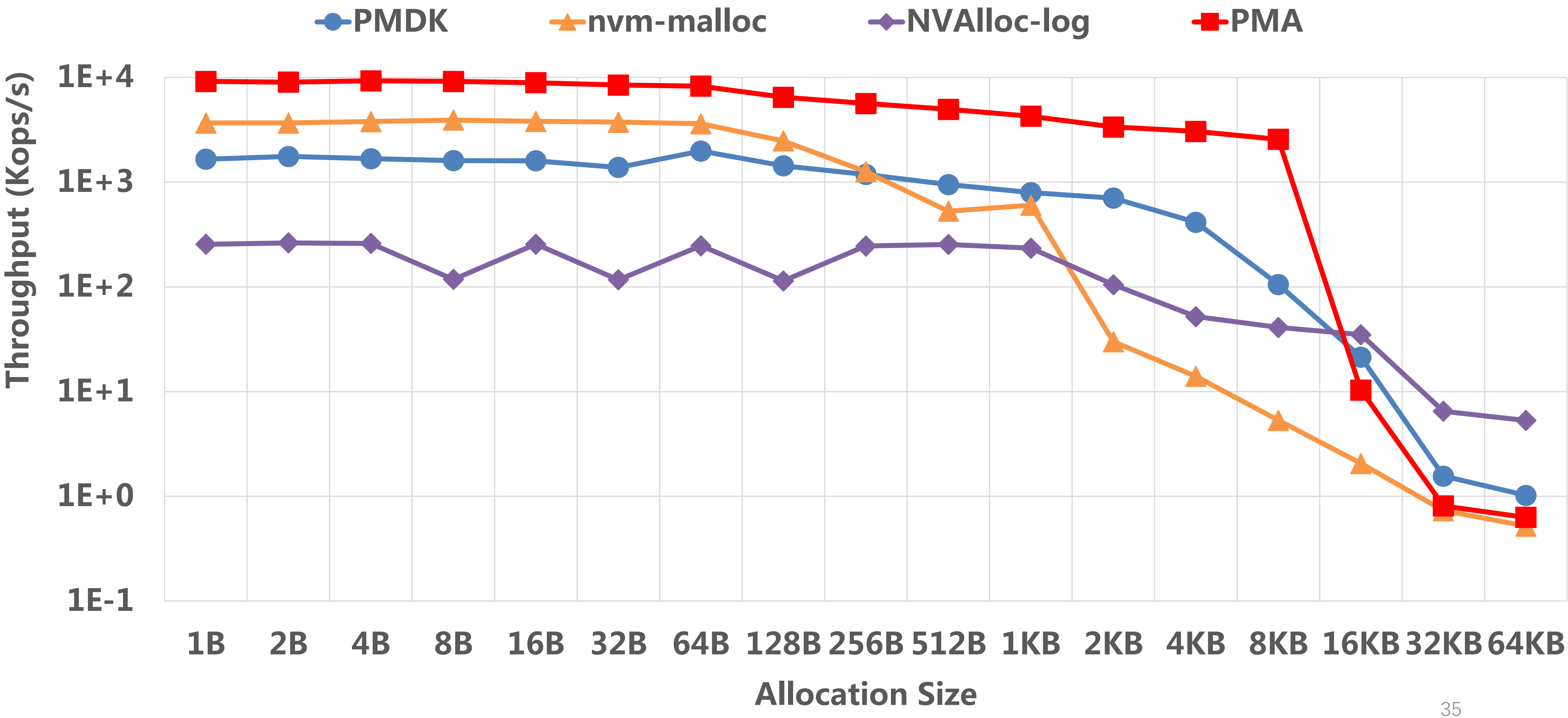


Performance Evaluation

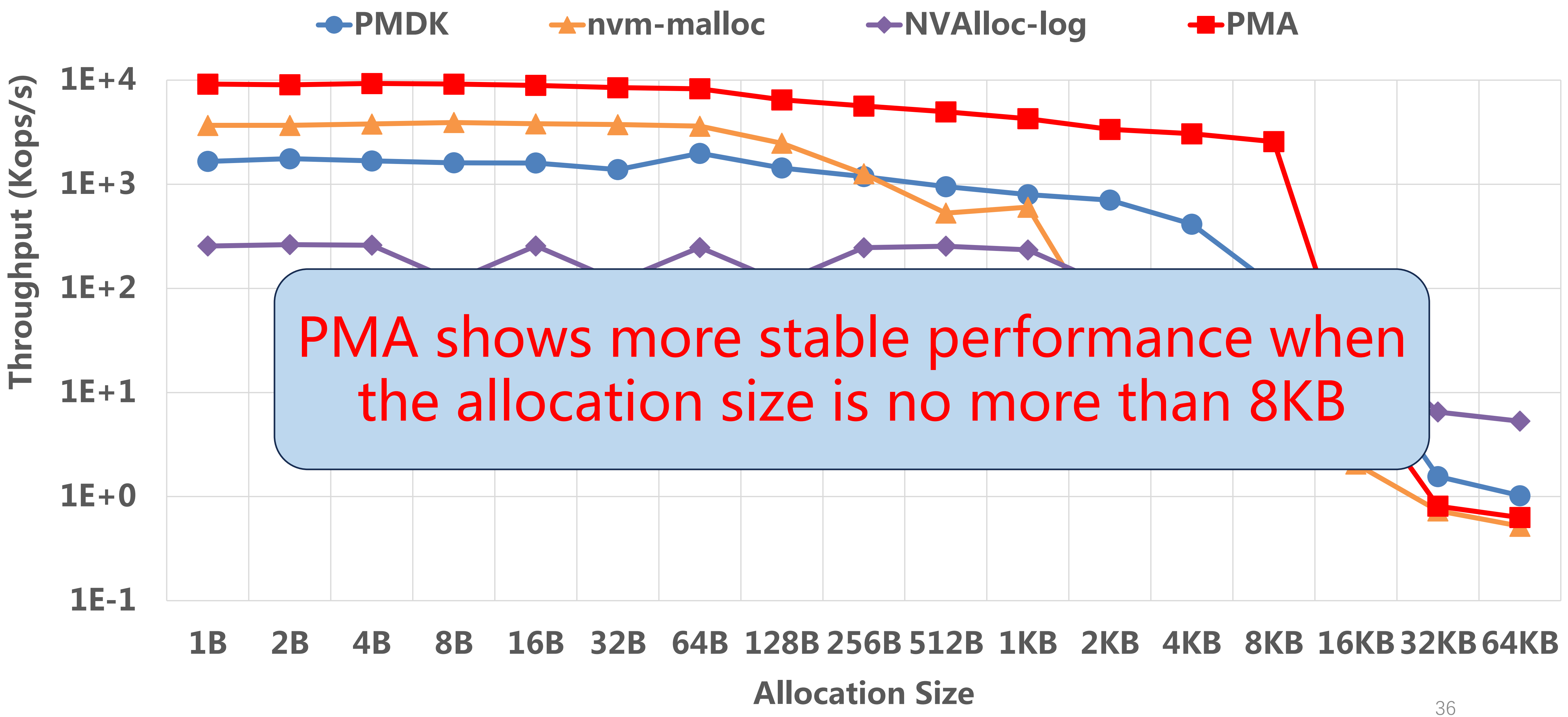


PMA reduces the tail latency by up to 3 orders of magnitude

Performance Evaluation



Performance Evaluation



PMA shows more stable performance when the allocation size is no more than 8KB

Conclusion

- DRAM allocators cannot be used for PM due to crash consistency issues

- PMA: a high-performance and crash-consistent allocator for PM
  - ✓ Two-level memory management
  - ✓ Thread-private memory pool
  - ✓ Lightweight write-ahead log

- 3.4x~19.1x higher throughput and orders of magnitude lower tail latency

Thanks!

Q&A

https://github.com/HUSTxyxiang/PMA/

xyxiang@hust.edu.cn