ZNSwap: un-Block your Swap

Shai Bergman, Niklas Cassel, Matias Bjørling, Mark Silberstein

Motivation: Why is Swap Important?

Data center applications exhibit large memory footprint

VOLTDB

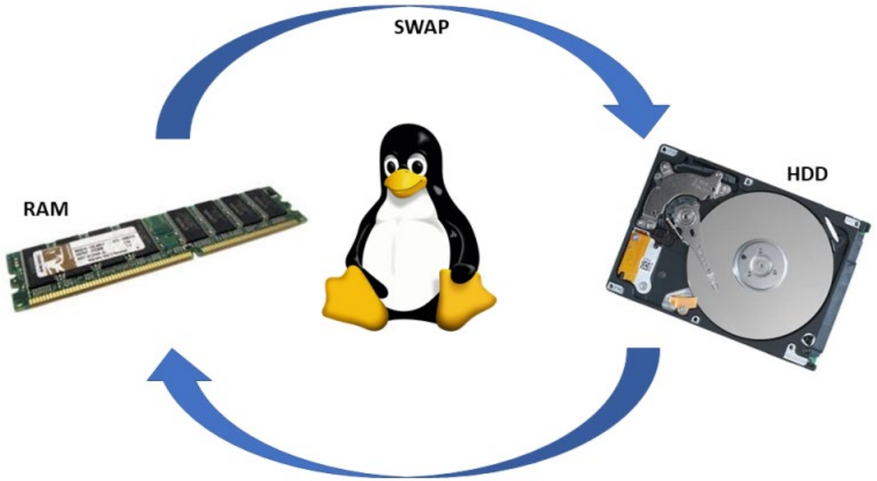
TensorFlow

Not all data is used frequently in the system

Motivation: Why is Swap Important?

Swap allows:

- Balance between file and anonymous pages in memory
- Even acting as memory extension



Swap use in industry:

- Facebook's fbtax2 swap controls
- Alibaba cloud: per-cgroup bg reclaim

Swap is regaining interest in academia, industry, and kernel communities

Motivation: Swap on Traditional SSDs

Flash technology is advancing:

- Raw access latencies decrease
- Available BW increases

} Great for memory swapping!

Motivation: Swap on Traditional SSDs?

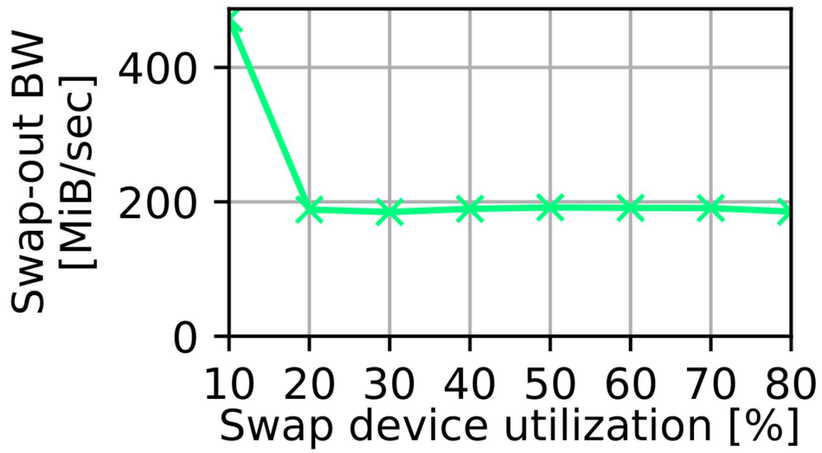
Flash technology is advancing:

- Raw access latencies decrease
- Available BW increases

} Great for memory swapping!

However, there are certain performance anomalies:

- Performance vs. space usage with swap:



Motivation: Swap on Traditional SSDs?

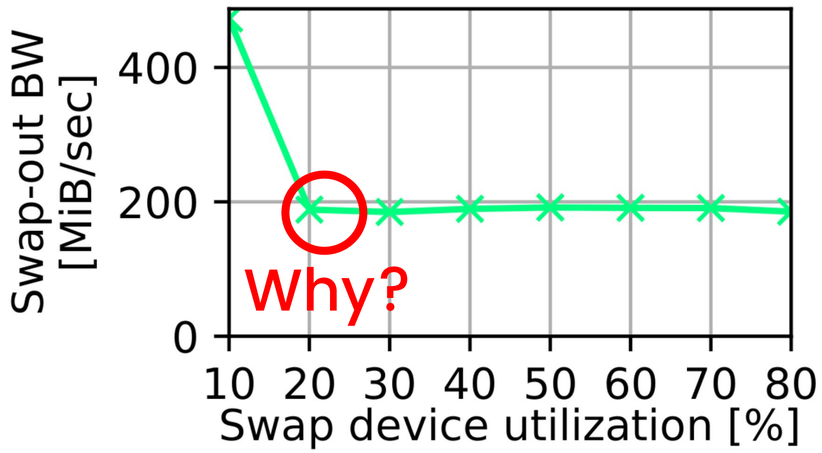
Flash technology is advancing:

- Raw access latencies decrease
- Available BW increases

} Great for memory swapping!

However, there are certain performance anomalies:

- Performance vs. space usage with swap:



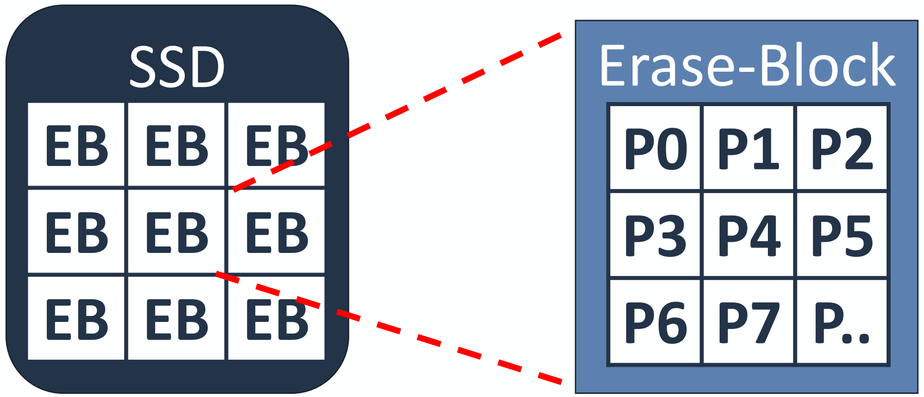
Agenda:

- Motivation
- Background on SSDs
- Analysis of swap on SSDs
- ZNS primer
- ZNSwap:
  - Design
  - Evaluation
  - Conclusion

Background: SSDs

Flash SSDs:

- Media is divided into “Erase-blocks”
- Erase-blocks divided into “pages”



Write ops: page granularity (sequential-only in erase-block)

Erase ops: erase-block granularity (blocks must be erased before re-written)

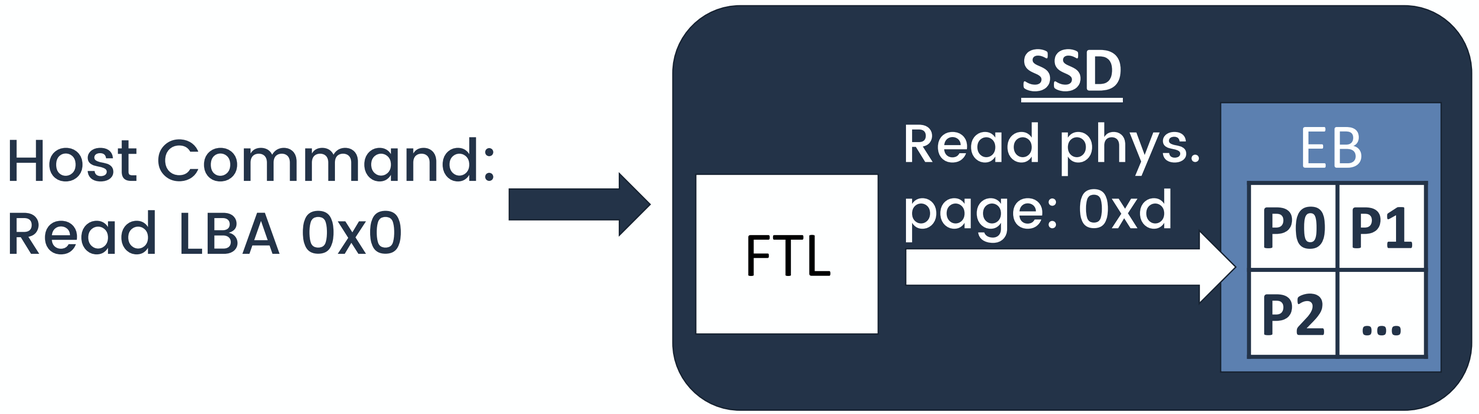
Background: SSDs

SSDs require a translation layer to:

- Allow in-place updates
- Allow random writes

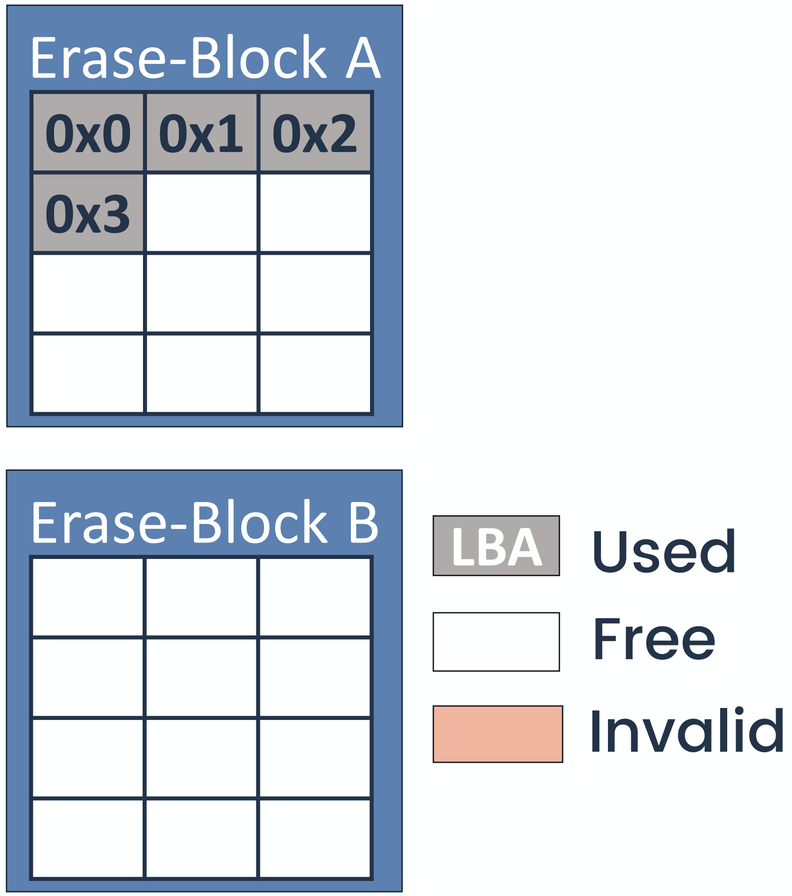
Flash Translation Layer (FTL):

- Maps Logical Block Address (LBA) to its physical location



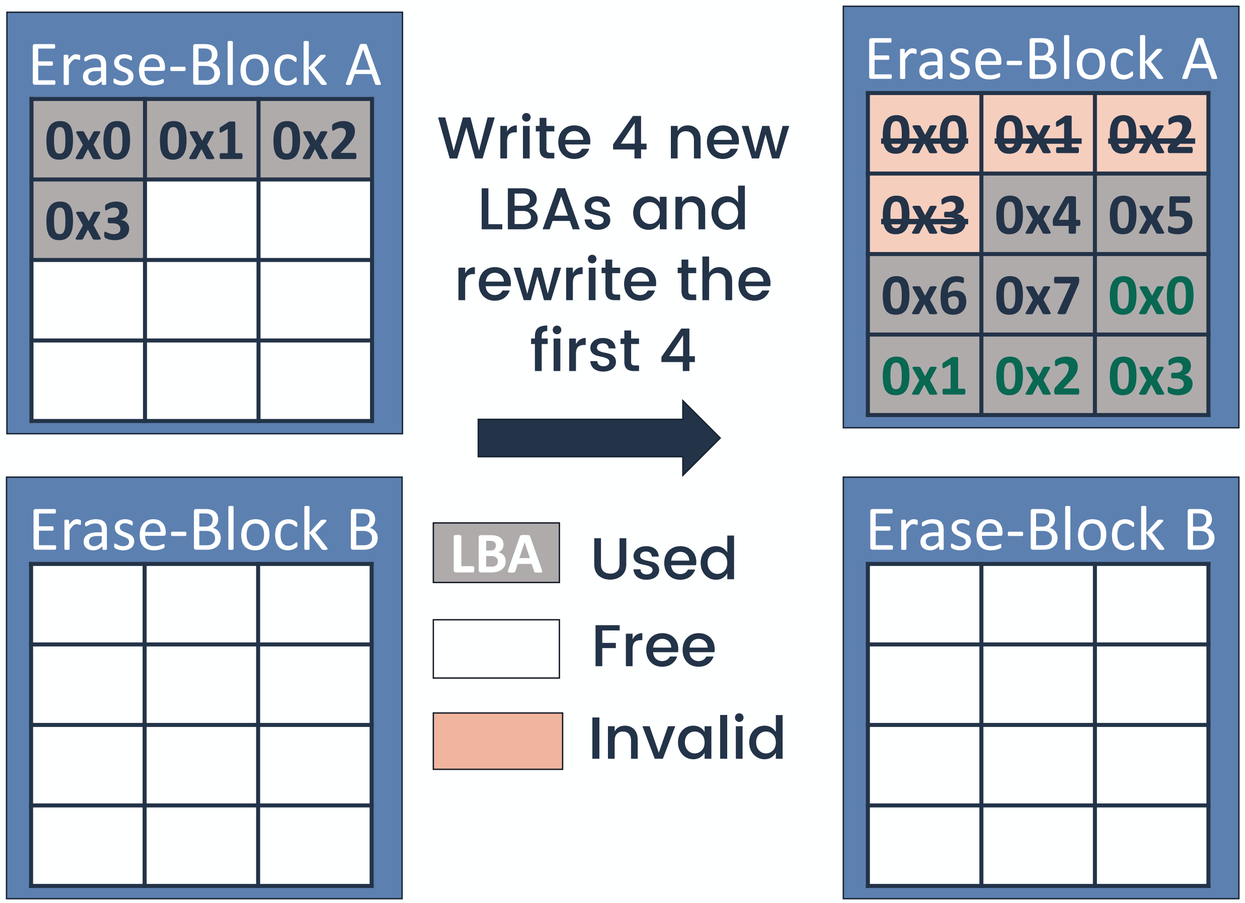
Background: SSDs

Garbage Collection (GC) & Write Amplification (WAF)



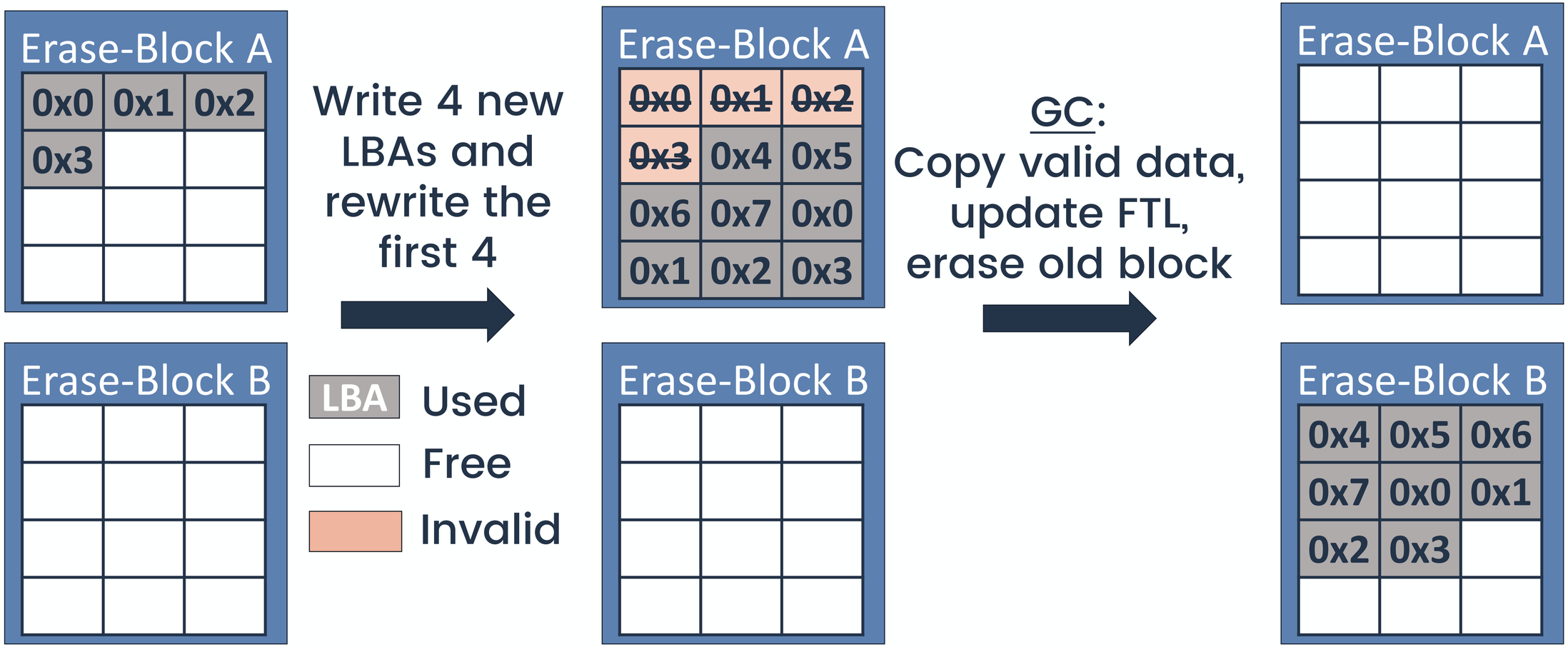
Background: SSDs

Garbage Collection (GC) & Write Amplification (WAF)



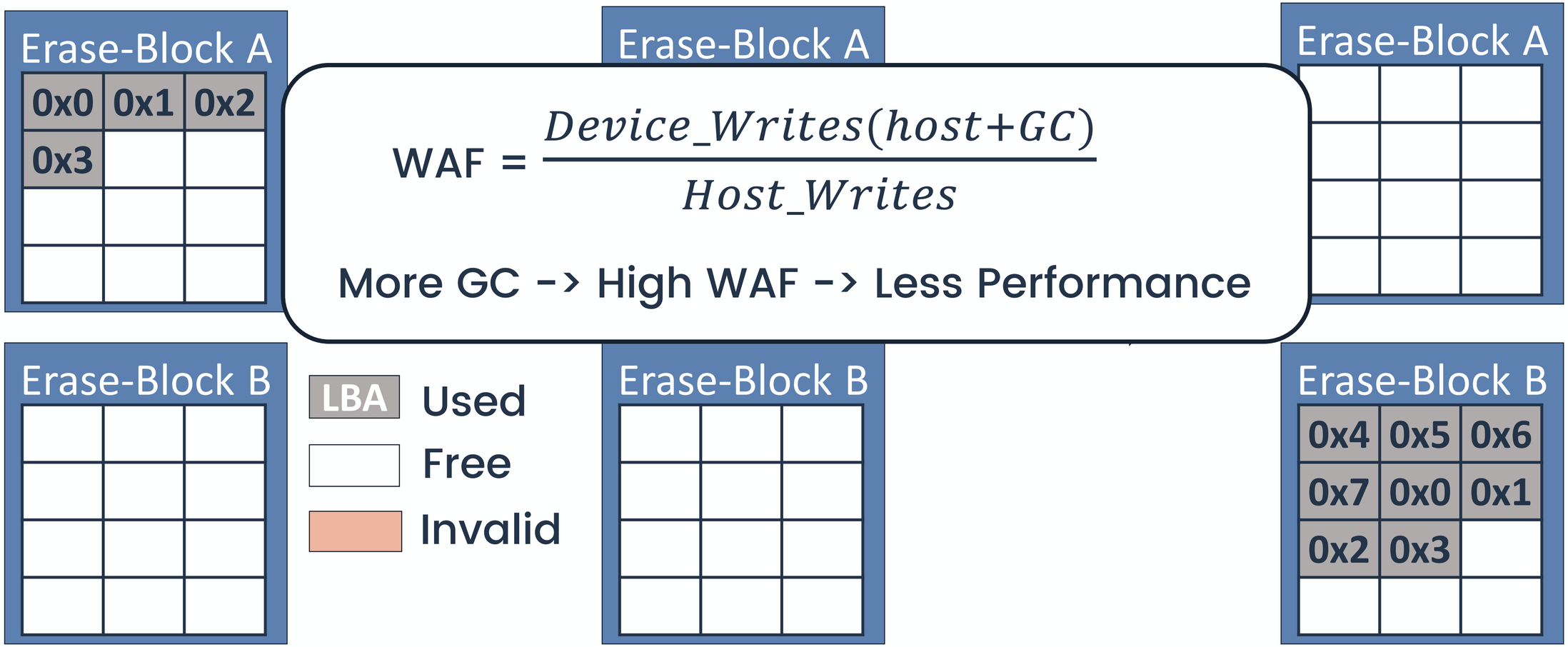
Background: SSDs

Garbage Collection (GC) & Write Amplification (WAF)



Background: SSDs

Garbage Collection (GC) & Write Amplification (WAF)

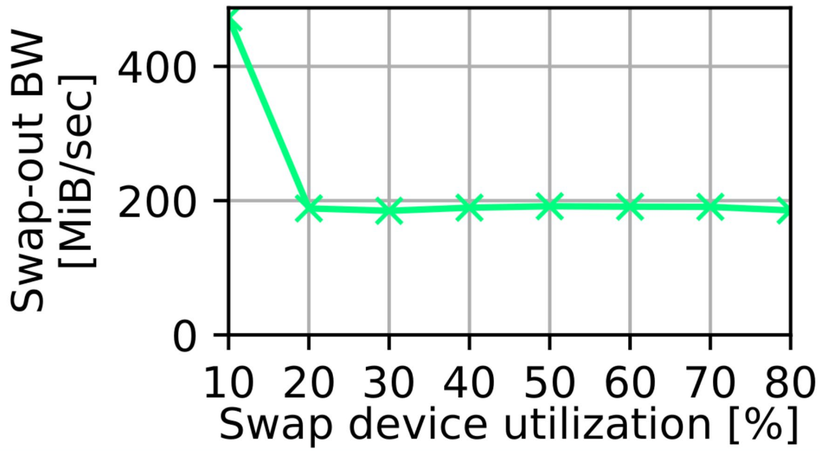


Analysis: Swap on SSDs

Swap performance drop caused by GC!

Knowledge gap between SSD and OS:

- SSD is unaware some swap data is “invalid”
- GC copies unnecessary data
- Performance decrease, WAF increase



Analysis: Swap on SSDs - TRIMs

TRIM ops:

- Hint by the host to invalidate a flash-page
- GC will not copy invalidated pages

TRIM processing cannot keep up with swap invalidation rate

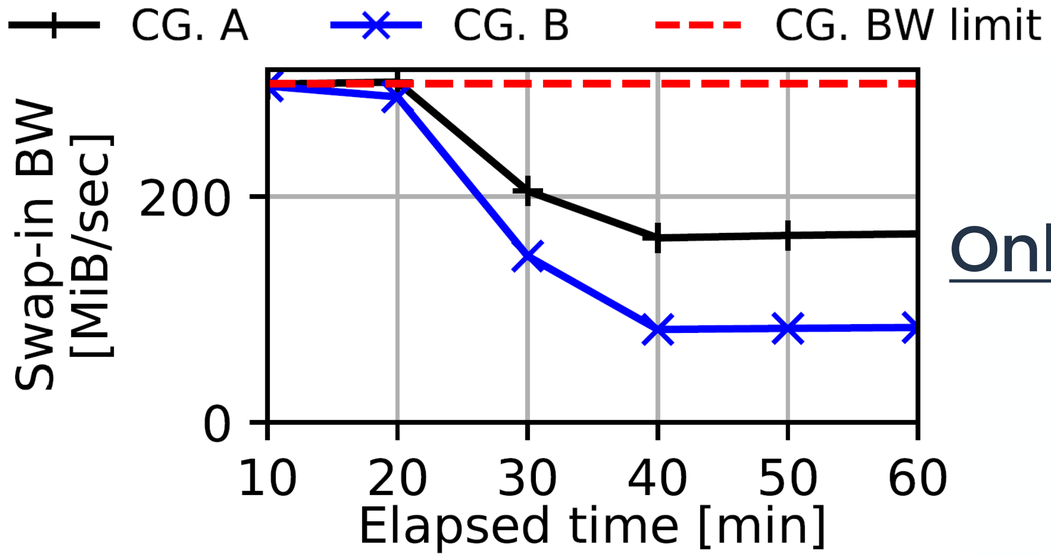
Detailed analysis is in the paper

Analysis: Swap on SSDs

GC affects the entire SSD:

Performance isolation cannot be guaranteed

- Cgroup A – 100% reads
- Cgroup B – 100% writes

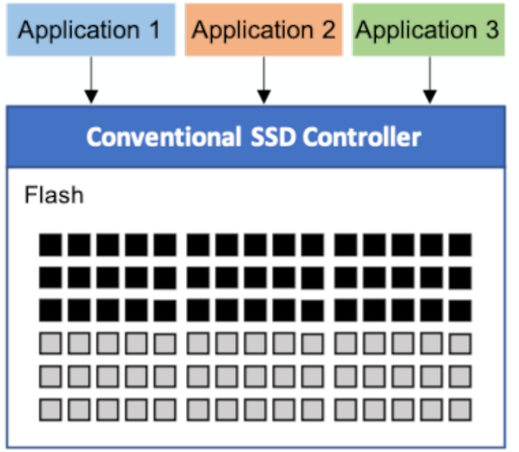


Only CG. B causes GC!

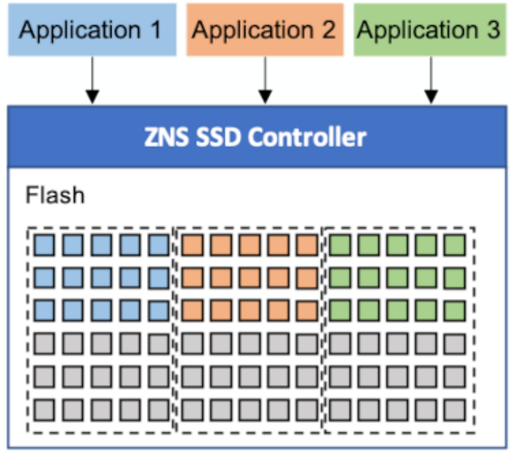
ZNS: Tighter SSD-APP coupling

Zoned Namespace (ZNS SSDs):

- SSD is divided into *zones* (erase-block size)
- Each zone is written sequentially
- Zones need to be *reset* before re-writing



Regular SSD: Device controls data placement



ZNS SSD: Applications control data placement in zones

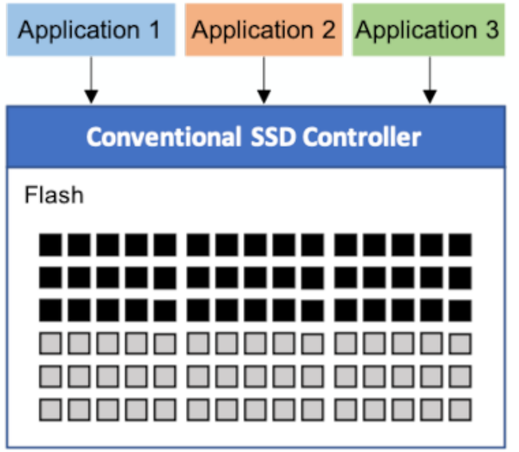
ZNS: Tighter SSD-APP coupling

Zoned Namespace (ZNS SSDs):

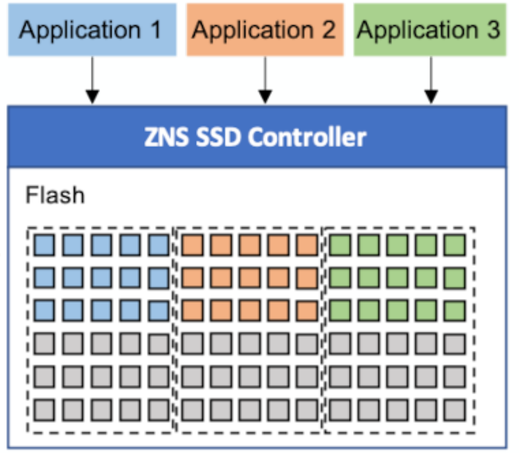
- SSD is divided into *zones* (erase-block size)
- Each zone is written sequentially
- Zones need to be *reset* before re-writing

✓ No complicated FTL, no device GC

✓ Higher degree of control over the device



Regular SSD: Device controls data placement



ZNS SSD: Applications control data placement in zones

ZNS: Tighter SSD-APP coupling

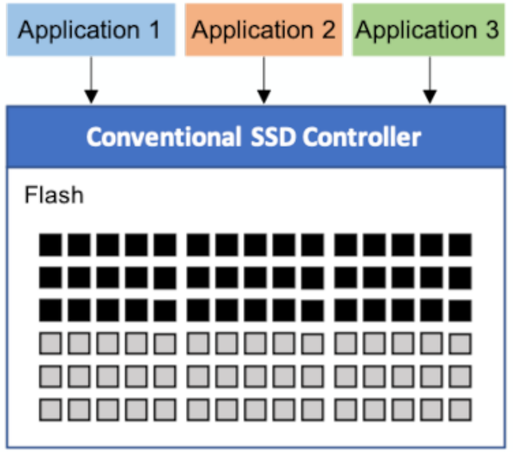
Zoned Namespace (ZNS SSDs):

- SSD is divided into *zones* (erase-block size)
- Each zone is written sequentially
- Zones need to be *reset* before re-writing

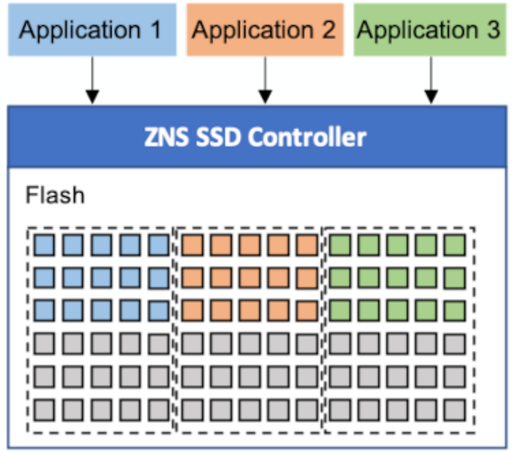
✓ No complicated FTL, no device GC

✓ Higher degree of control over the device

ZNS re-establishes host control over key SSD aspects



Regular SSD: Device controls data placement



ZNS SSD: Applications control data placement in zones

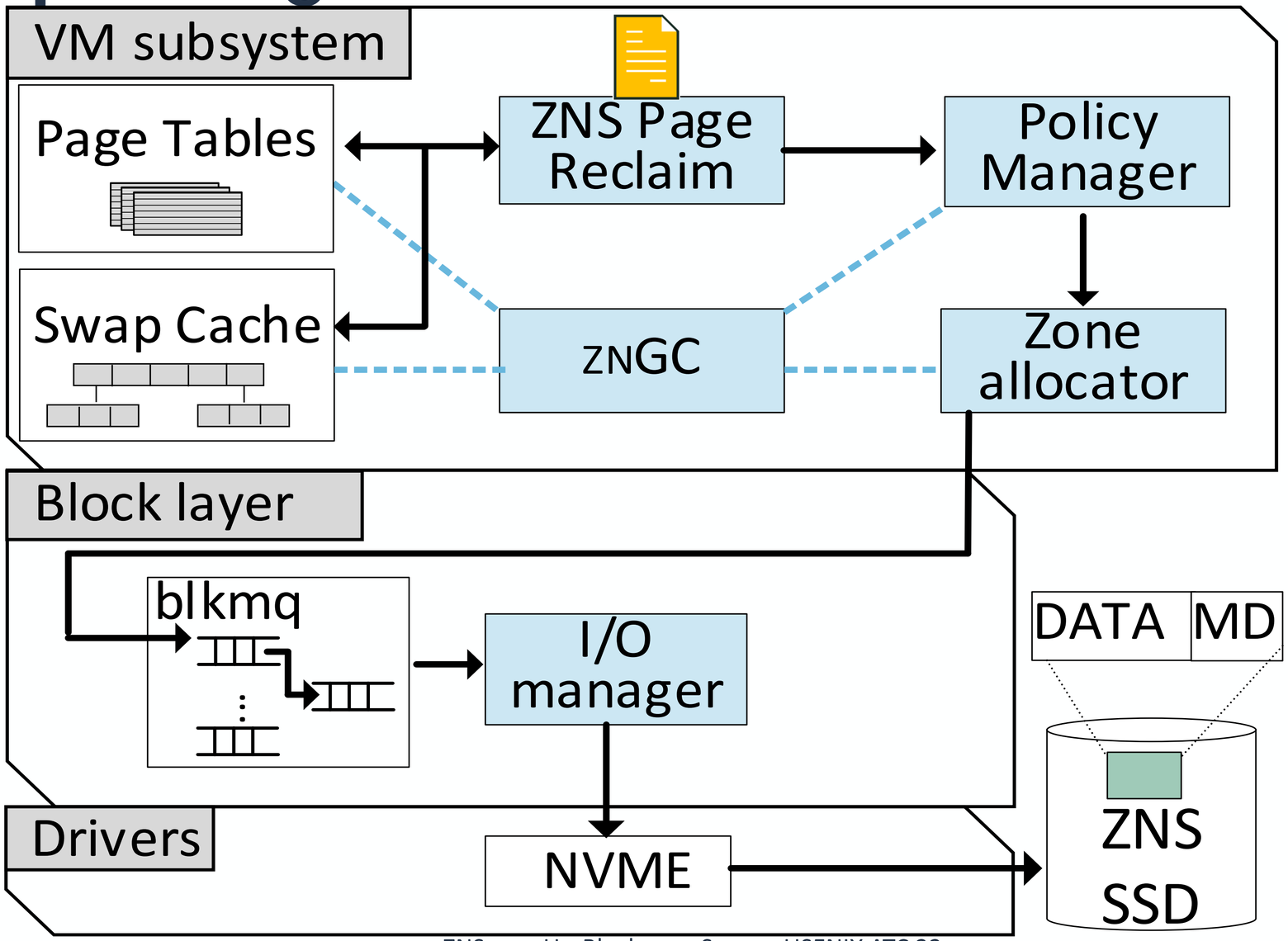
ZNSwap: new swap subsystem for ZNS

ZNSwap enables synergy between OS swap logic and SSD.

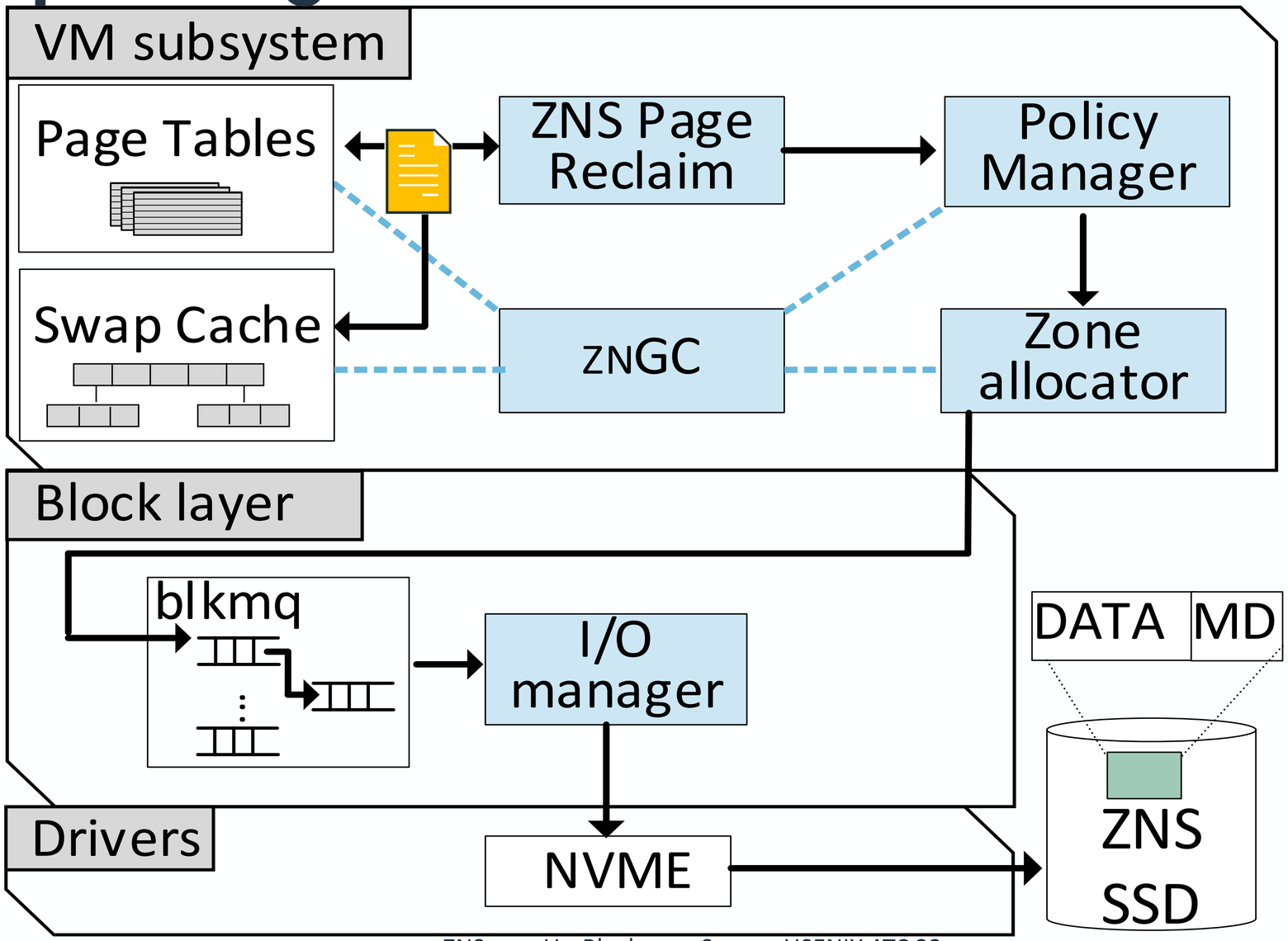
Design goals:

- Fine-grain space management with host-GC
- Swap-cache & host-GC co-design
- Custom data placement and GC reclaim policies:
  - E.g. hot/cold
- Multi-tenancy isolation

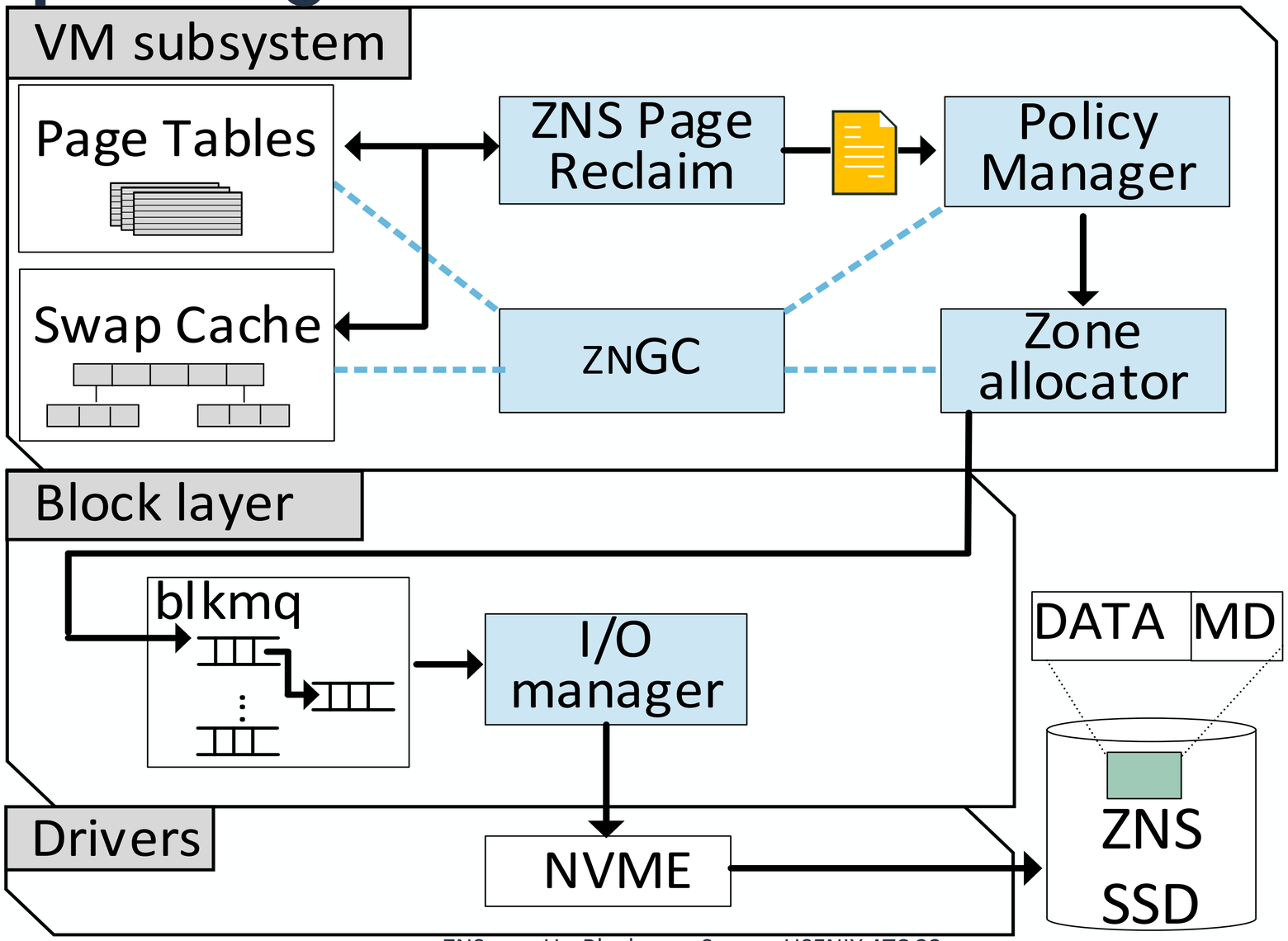
ZNSwap: Design Overview



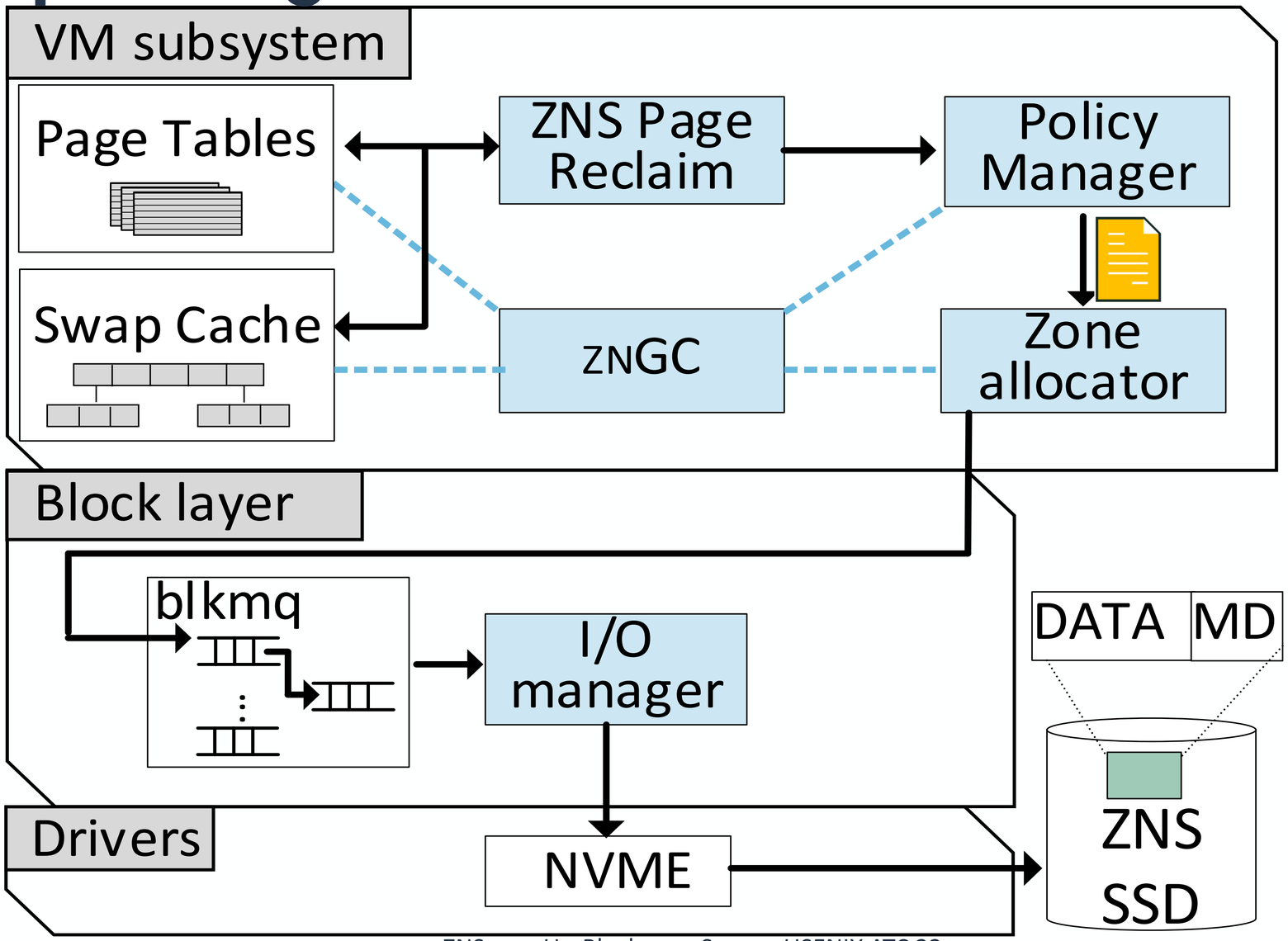
ZNSwap: Design Overview



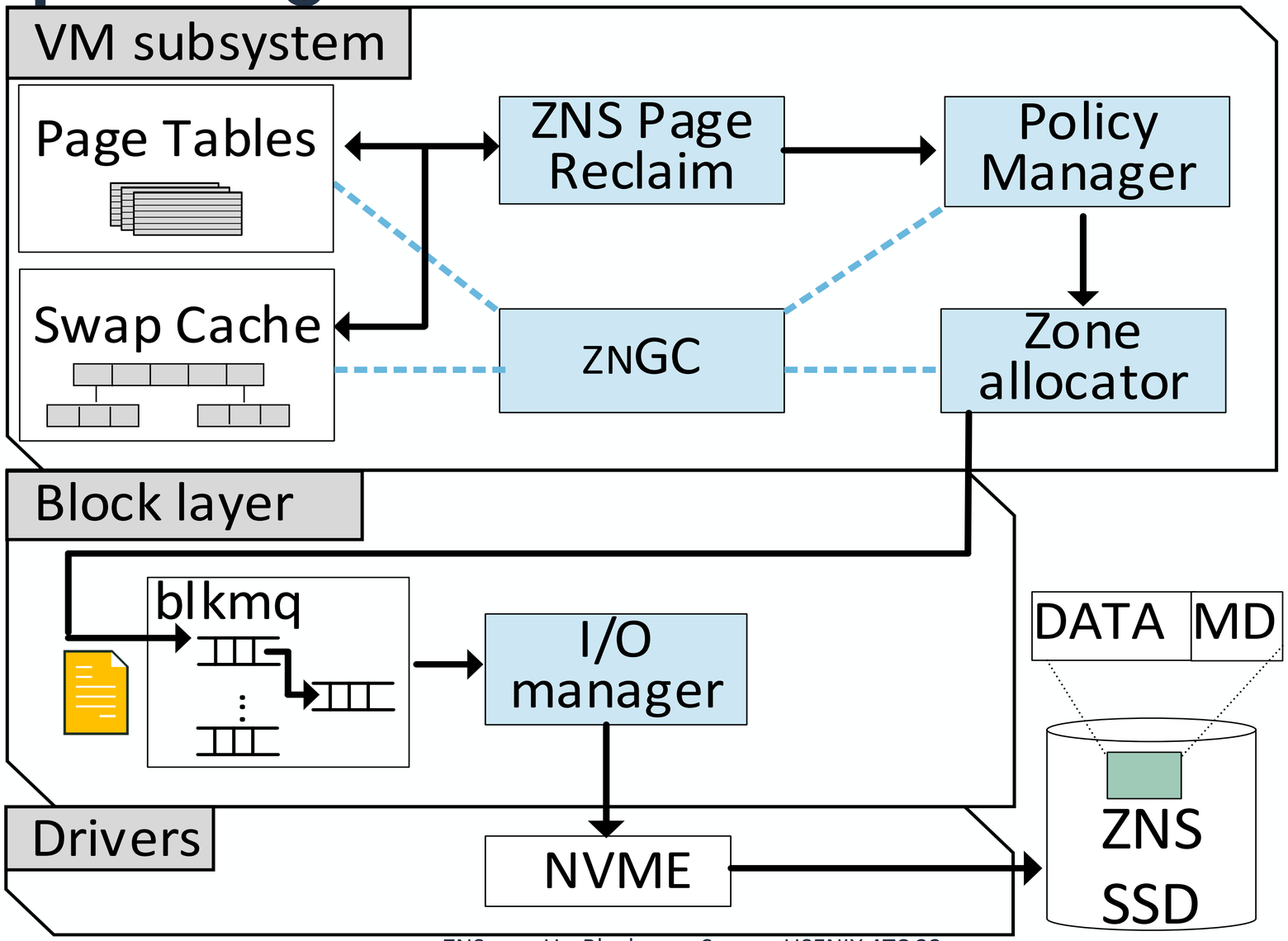
ZNSwap: Design Overview



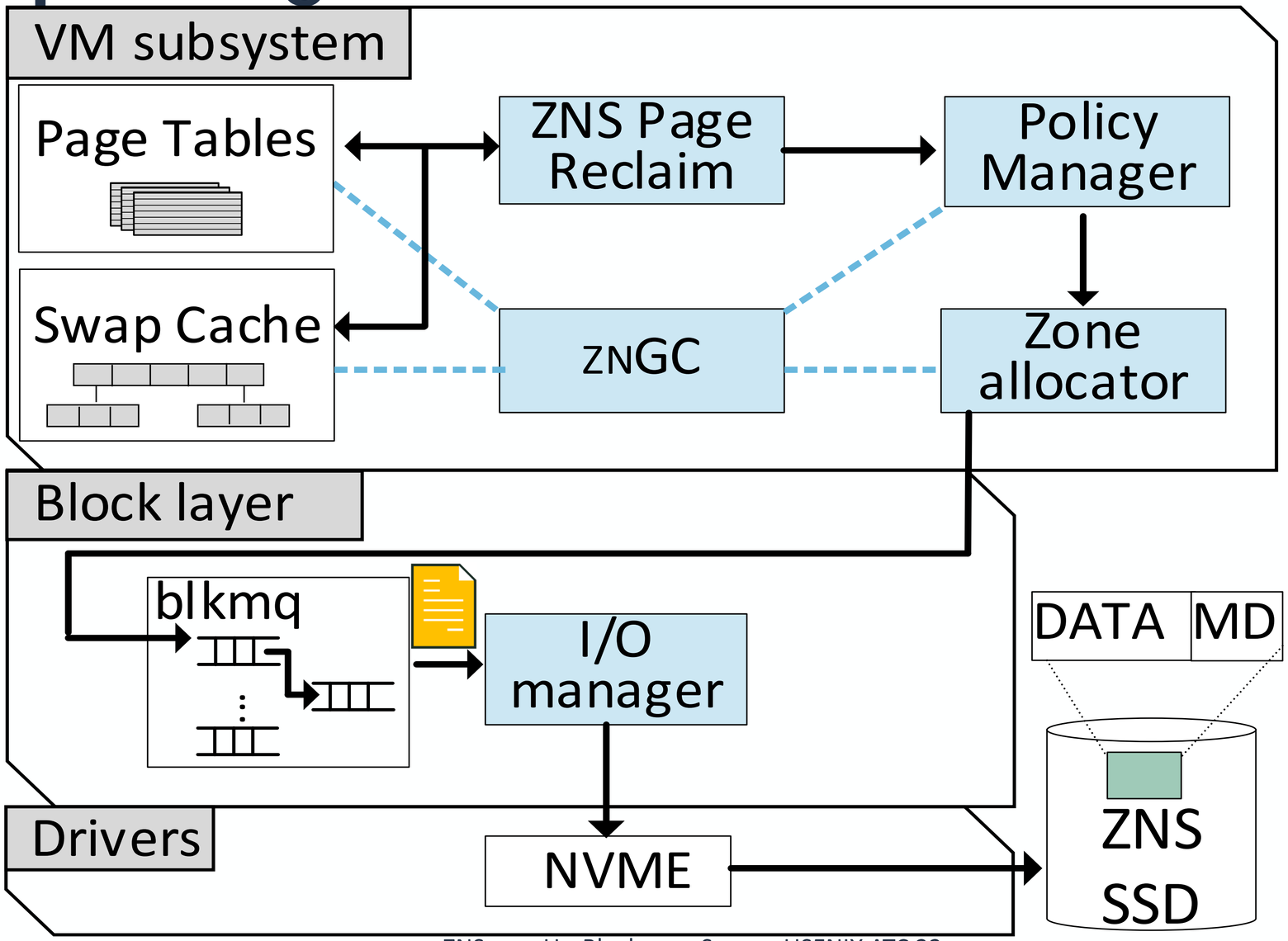
ZNSwap: Design Overview



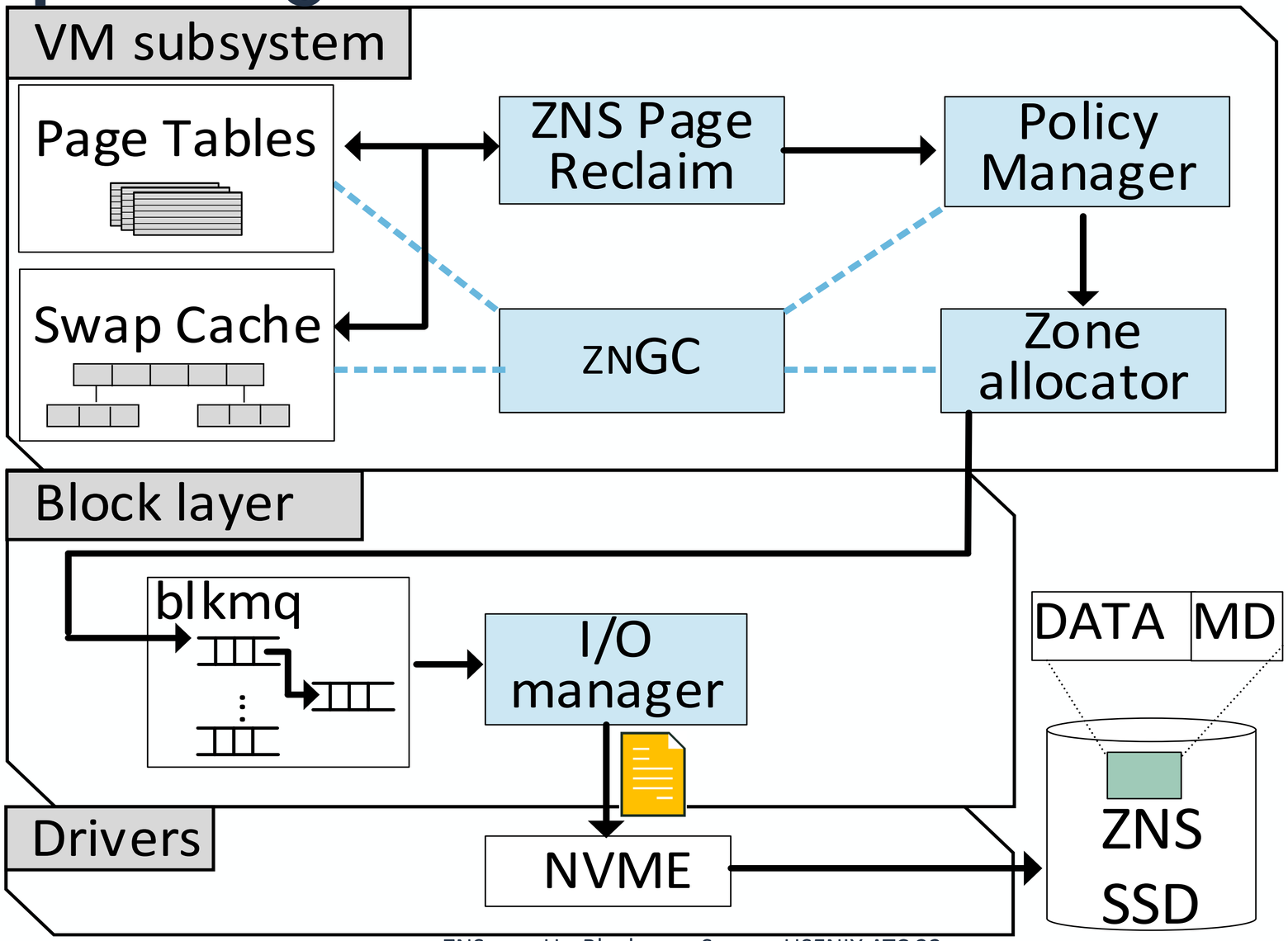
ZNSwap: Design Overview



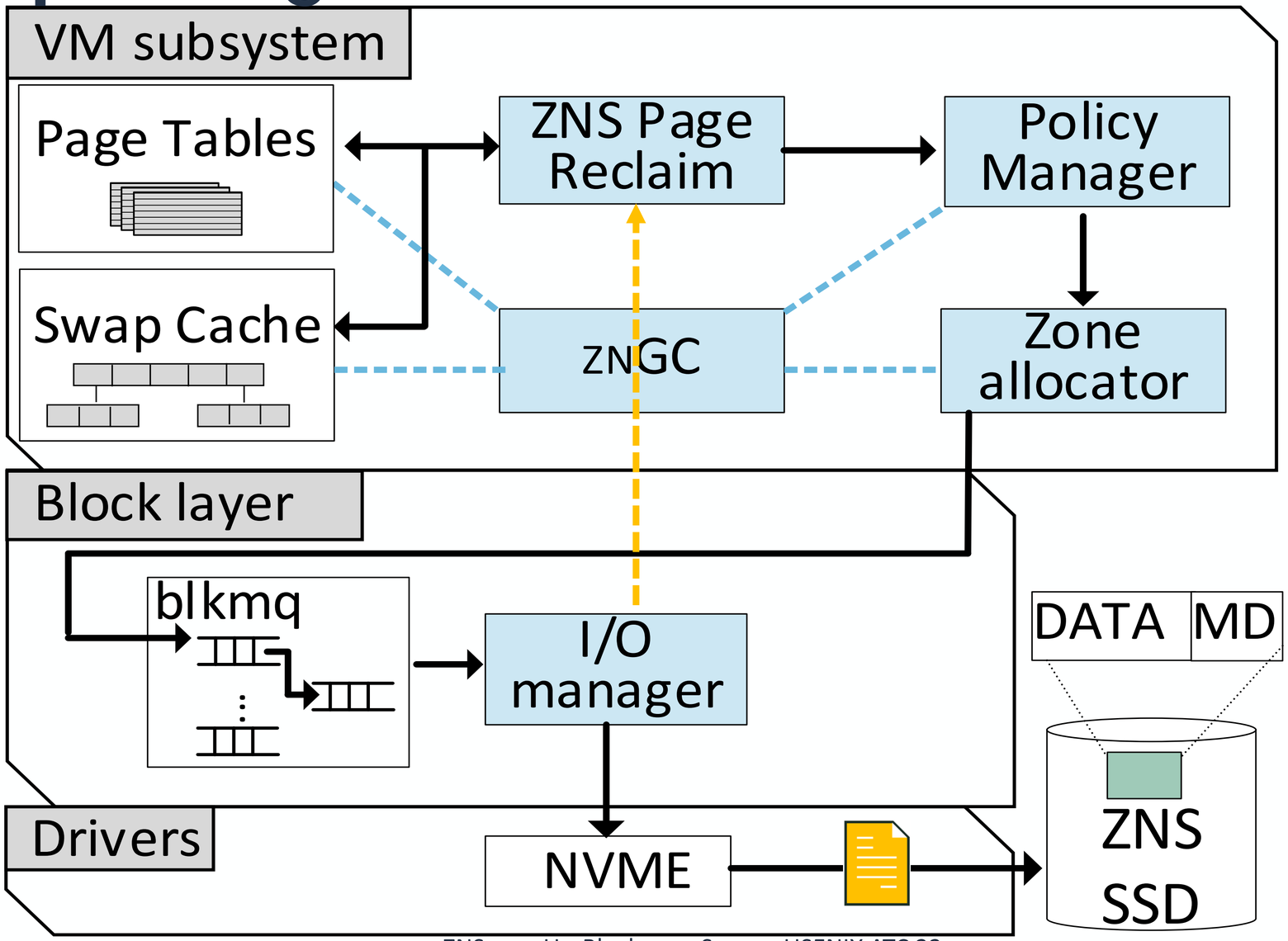
ZNSwap: Design Overview



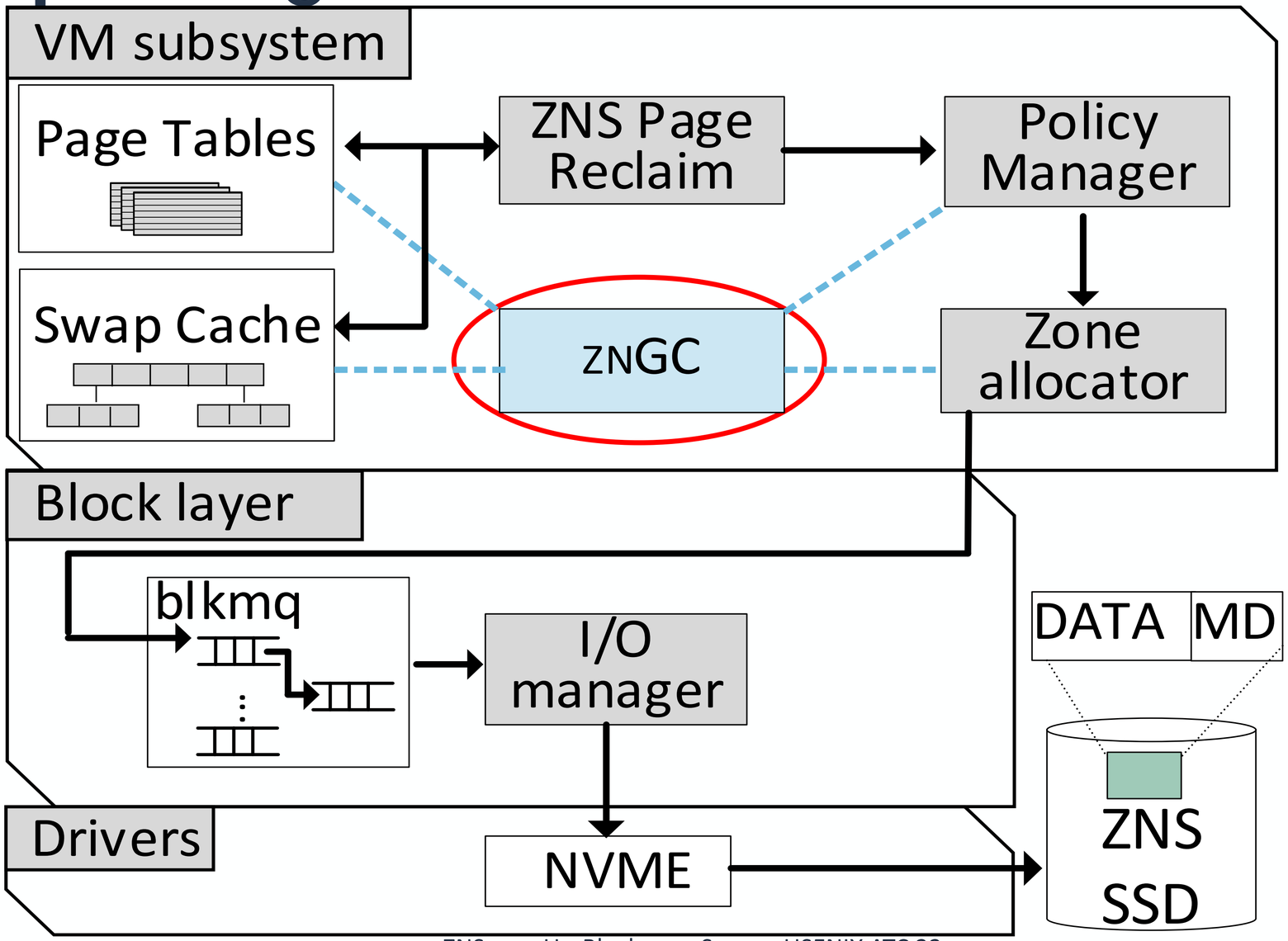
ZNSwap: Design Overview



ZNSwap: Design Overview



ZNSwap: Design Overview

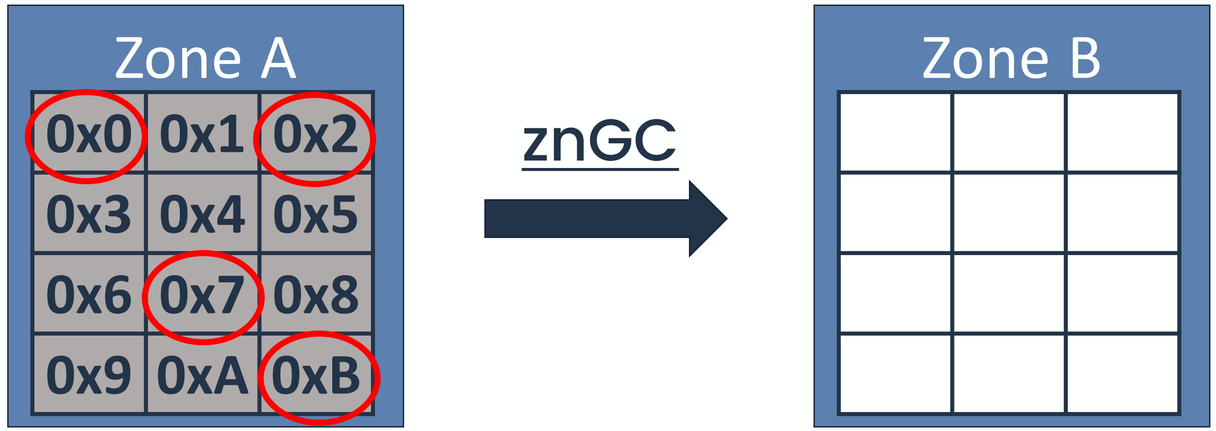


ZNSwap Design: ZNGC

- Host-side GC for ZNS devices eliminates:
  - TRIMs
  - Uncertainty of autonomous GC
  - Copy of invalid data

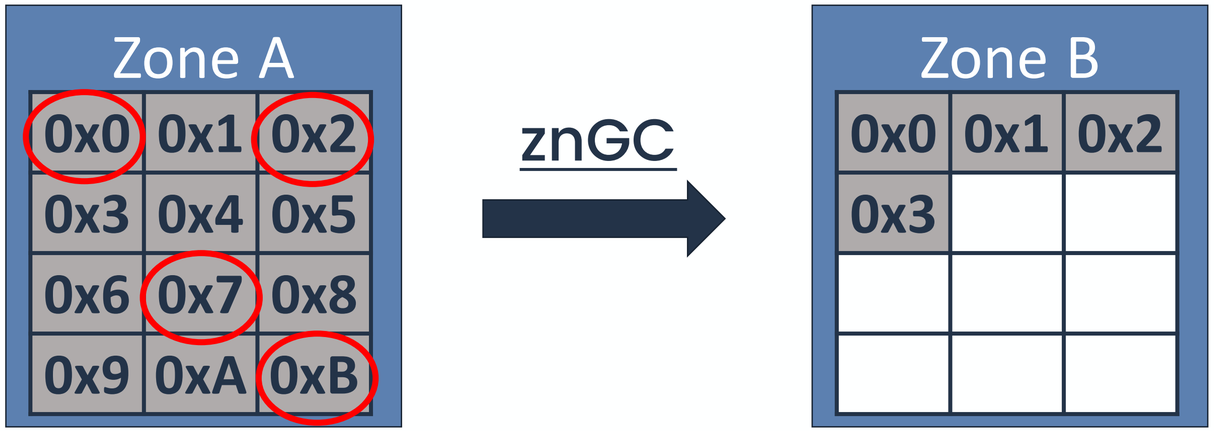
ZNSwap Design: ZNGC

- Host-side GC for ZNS devices eliminates:
  - TRIMs
  - Uncertainty of autonomous GC
  - Copy of invalid data



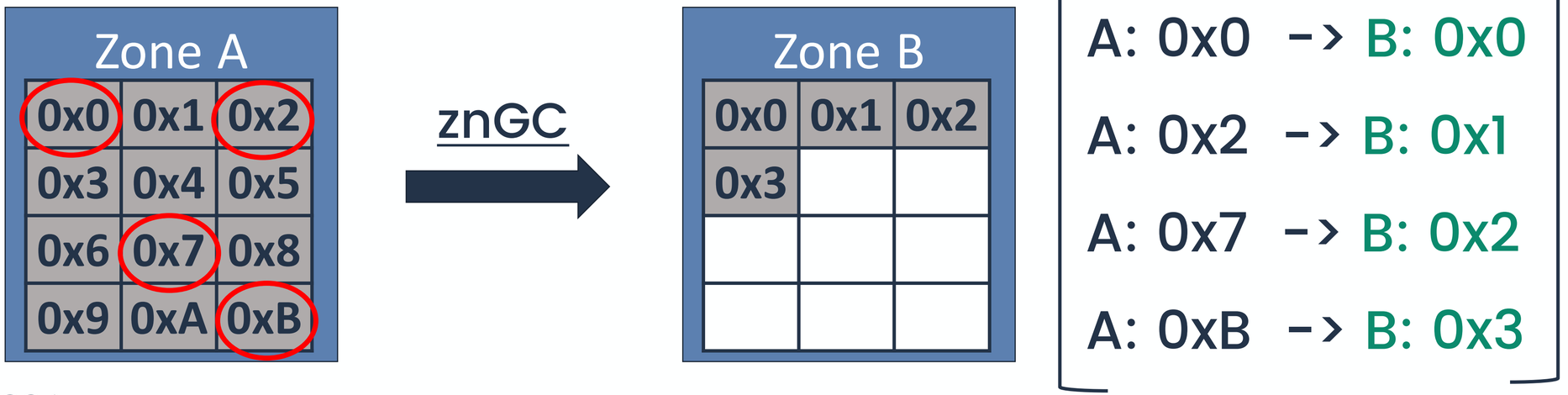
ZNSwap Design: ZNGC

- Host-side GC for ZNS devices eliminates:
  - TRIMs
  - Uncertainty of autonomous GC
  - Copy of invalid data



ZNSwap Design: ZNGC

Host-side GC moves valid swap data to new locations:



Problem:

- No FTL for indirection in ZNS
- Page tables point to old locations in SSD

How to locate all page table entries?

ZNSwap Design: ZNGC

Host-s

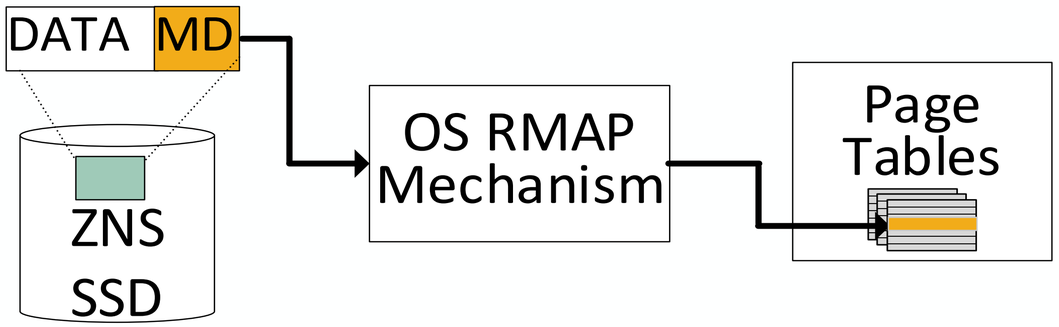
Proble

- No
- Pag

ZNGC Solution:

Store OS reverse-mapping info (anon. VMA ptr + index)

Utilize NVMe per-block Metedata region



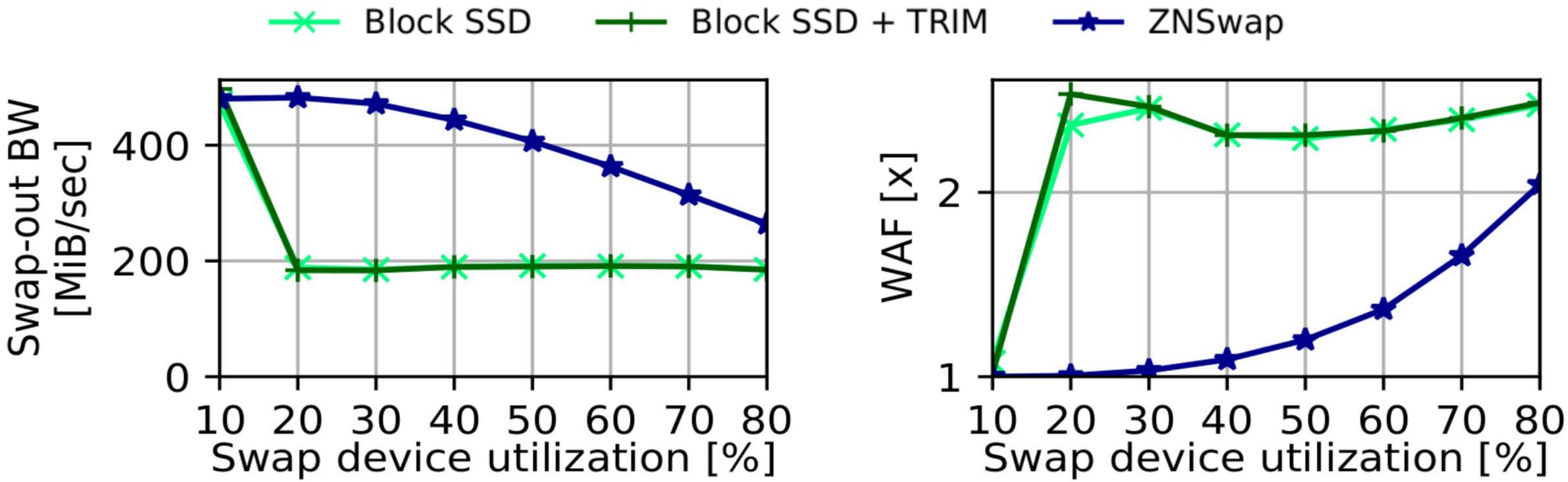
Efficiently update page tables with new location

ZNSwap Evaluation: Setup

- Linux Kernel 5.12
- 512 GiB RAM
- 1 TB Western Digital ZN540 ZNS SSD
- 1 TB Equivalent Conventional SSD

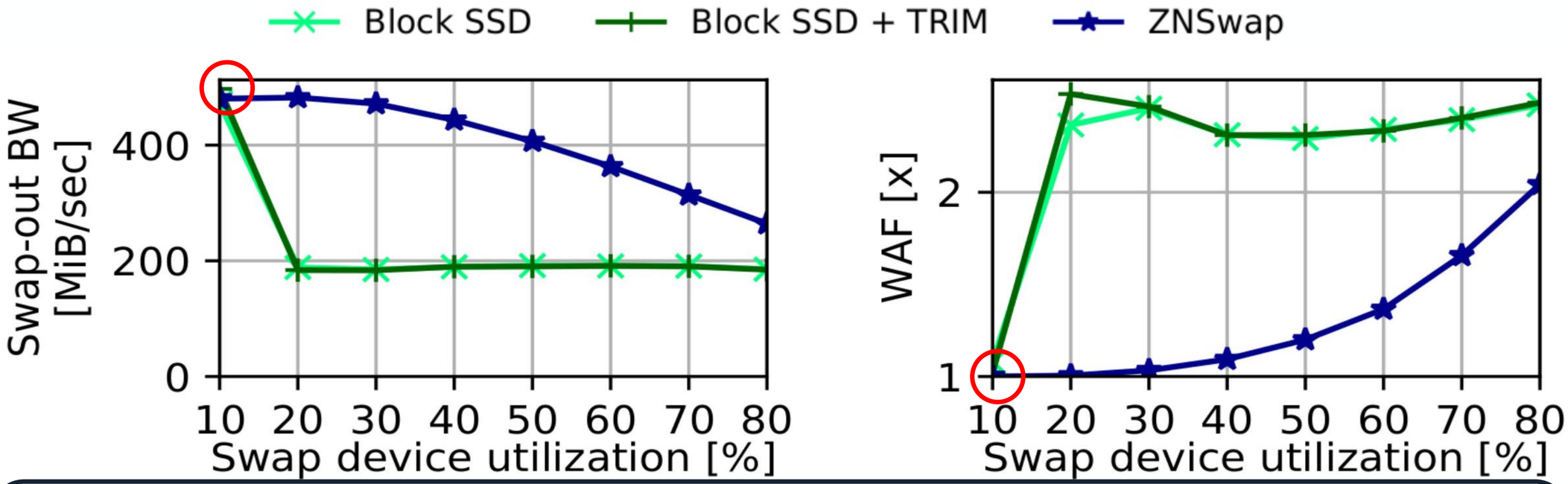
ZNSwap Evaluation: vm-scalability

Random memory writes



ZNSwap Evaluation: vm-scalability

Random memory writes



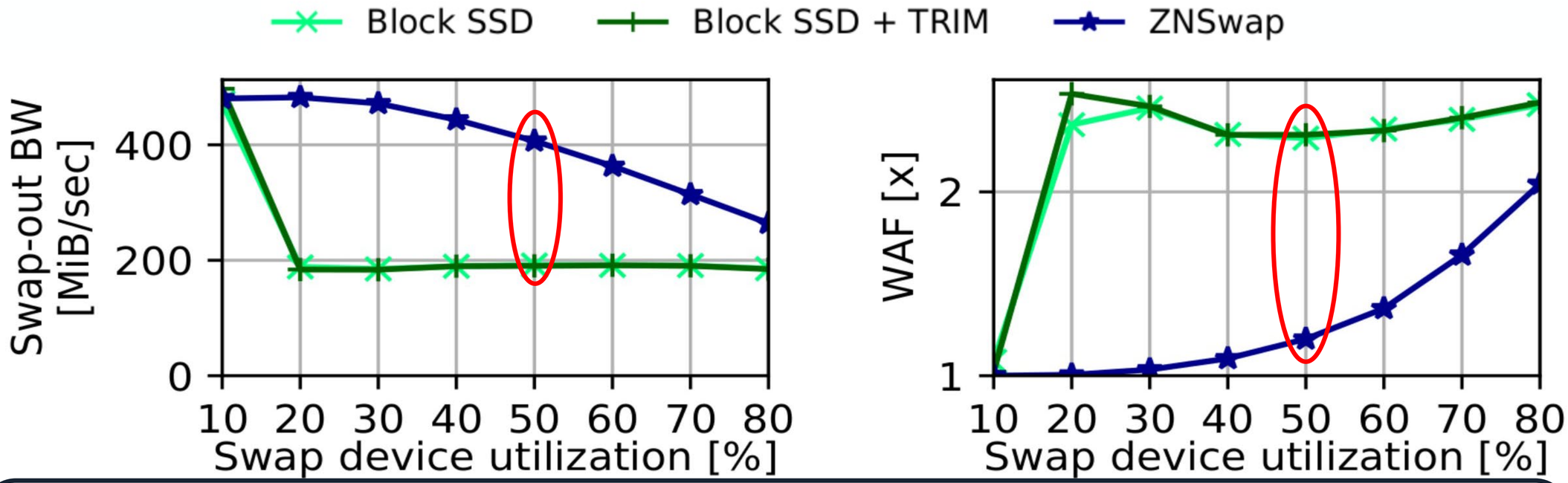
Low device utilization (10% utilization):

ZNSwap & Block SSD achieve highest device throughput and WAF = 1

ZNGC CPU overhead: 0.3% of a single core

ZNSwap Evaluation: vm-scalability

Random memory writes



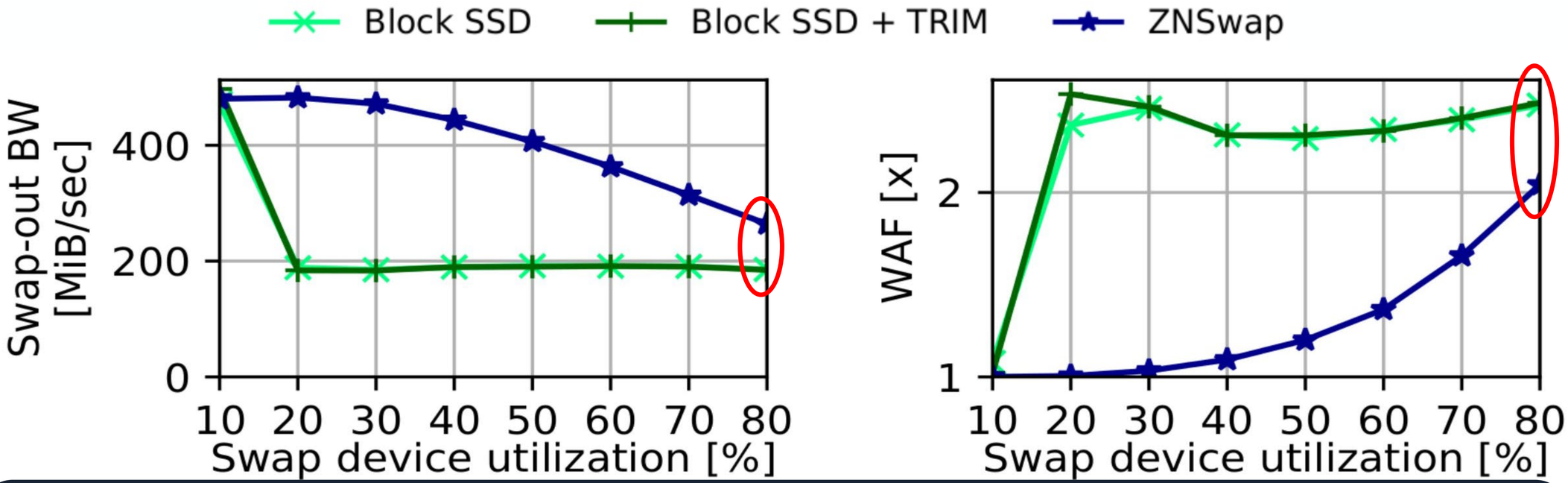
Higher device utilizations:

ZNSwap avoids unnecessary data copies. TRIMs are ineffective for Block SSD

50% util: 2x higher throughput, 2x lower WAF

ZNSwap Evaluation: vm-scalability

Random memory writes

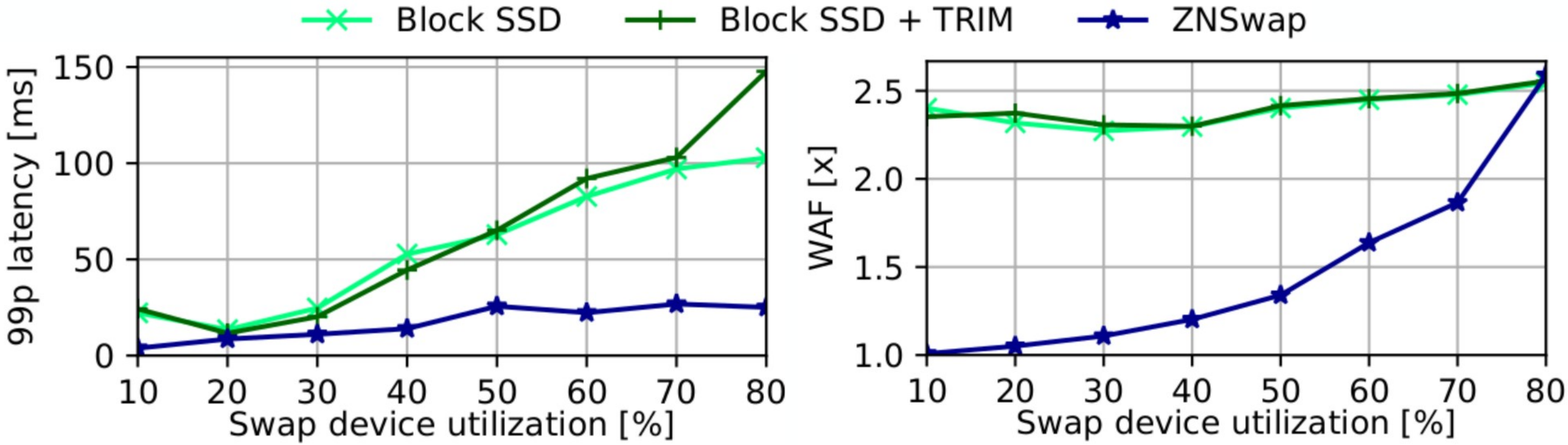


Max observable ZNGC CPU overhead: 15% of a single core

Fine-granularity TRIM dispatching CPU overheads: 32% of a single core

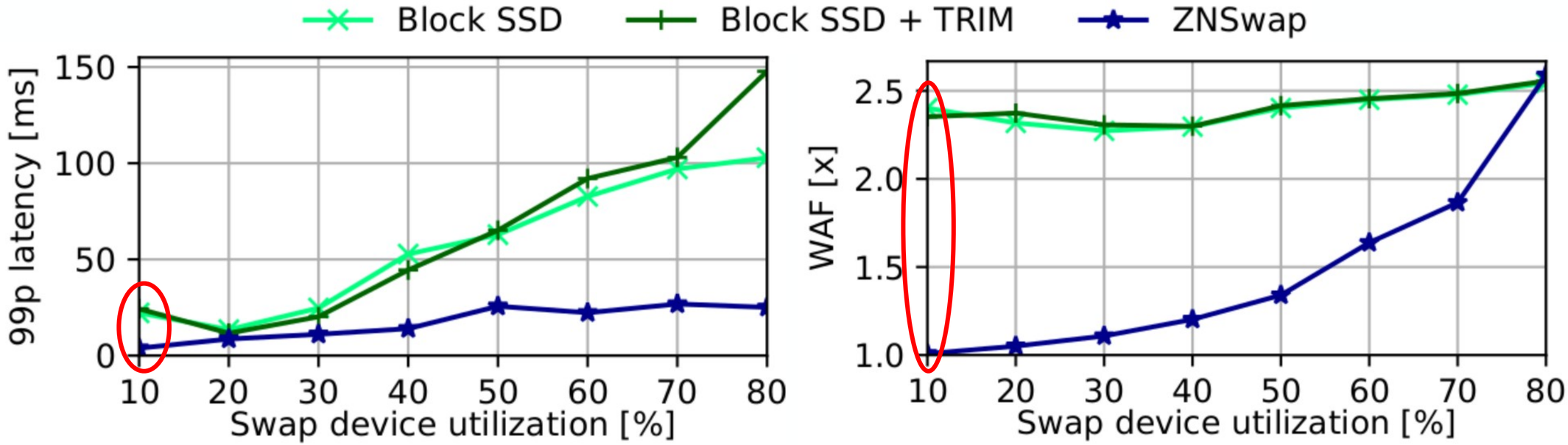
ZNSwap Evaluation: memcached

Serving Facebook ETC workload



ZNSwap Evaluation: memcached

Serving Facebook ETC workload



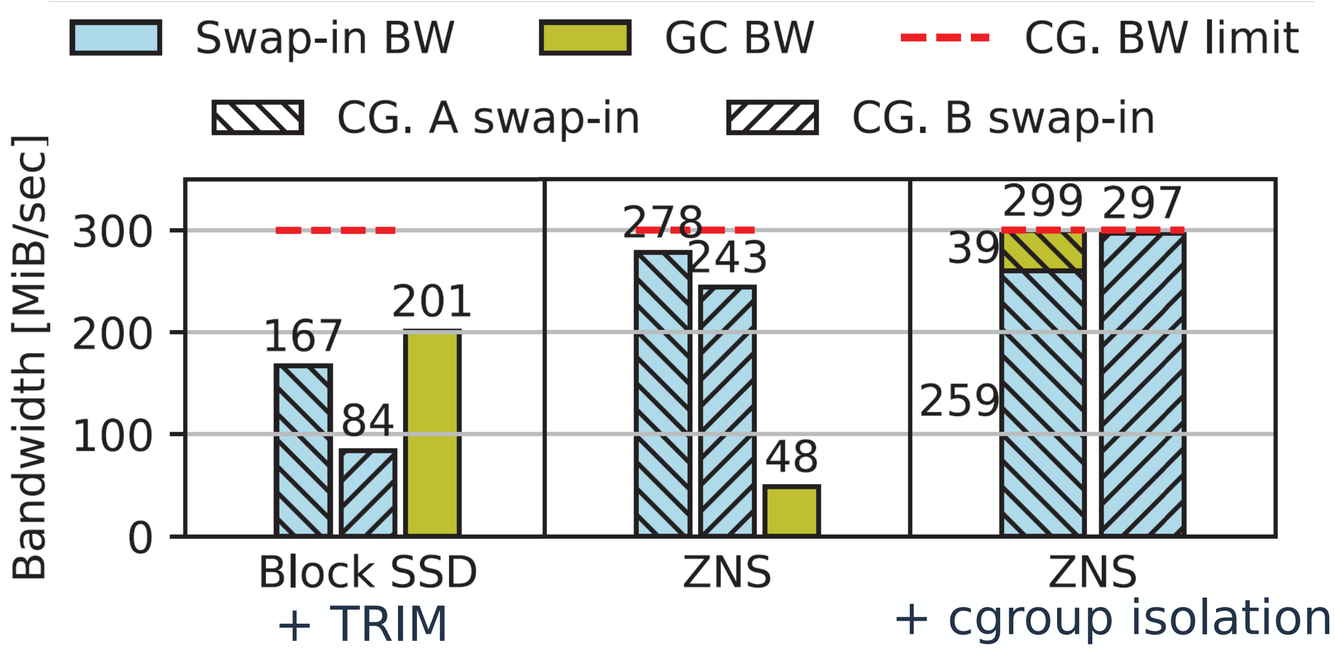
ZNSwap outperforms Block SSD under all device utilizations

10% util: 10x lower 99p latency, no WAF (vs. 2.4x)

ZNGC is efficient, predictable and issued by the host -> lower 99p

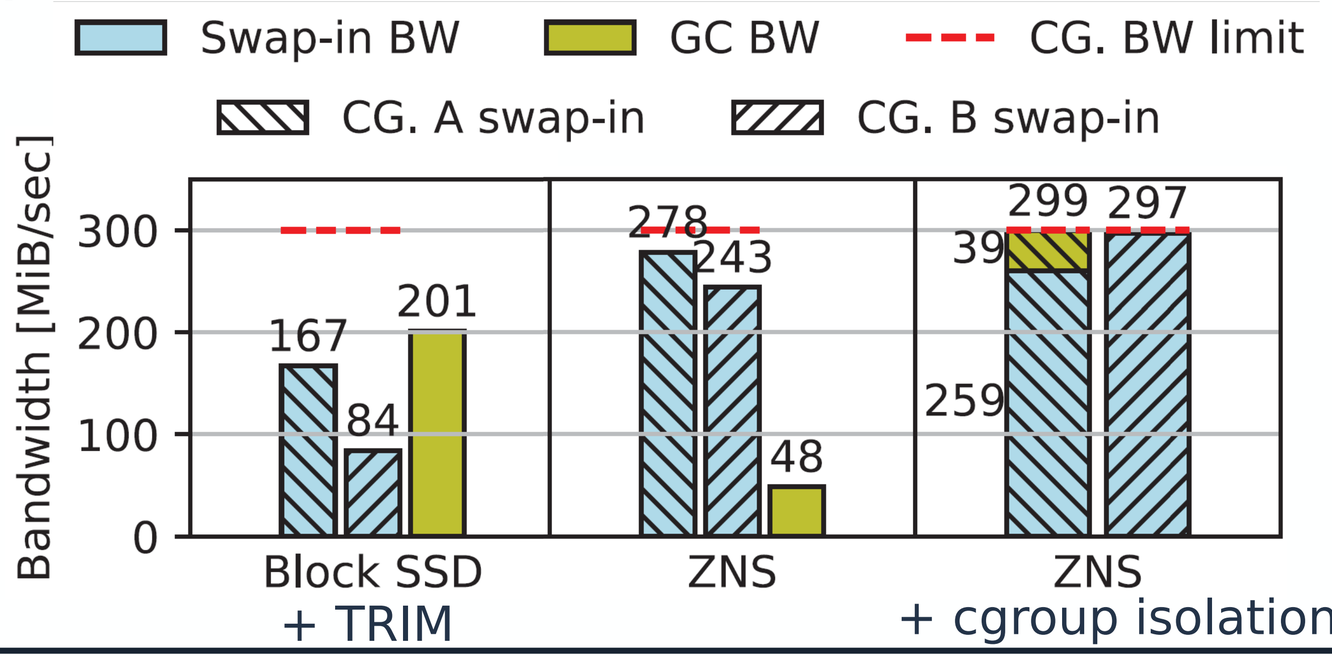
ZNSwap Evaluation: Cgroup Isolation

- Cgroup A – 100% reads
- Cgroup B – 100% writes



ZNSwap Evaluation: Cgroup Isolation

- Cgroup A – 100% reads
- Cgroup B – 100% writes



ZNSwap outperforms Block SSD

ZNSwap's Cgroup isolation policy, "punts" GC cost to its corresponding cgroup

ZNSwap controls key SSD mechanisms, enabling performance isolation

Conclusions

- Swap is regaining interest
- Swap on Traditional SSDs suffer from knowledge divide:
  - TRIMs unable to bridge the gap
  - Autonomous GC – no isolation
- ZNSwap enables tight SSD <-> OS swap integration:
  - Native ZNS support for swap with efficient host-side GC
  - Lowers WAF, higher performance, custom placement and reclaim policies

Thank you!

Questions?


shaiberg1@tx.technion.ac.il


github.com/acsl-technion/znswap