A Free-Space Adaptive Runtime Zone-Reset Algorithm for Enhanced ZNS Efficiency

Sungjin Byeon[1)], Joseph Ro[1)], Safdar Jamil[1)], Jeong-Uk Kang[2)], Youngjae Kim[1)]

[1)]Sogang University

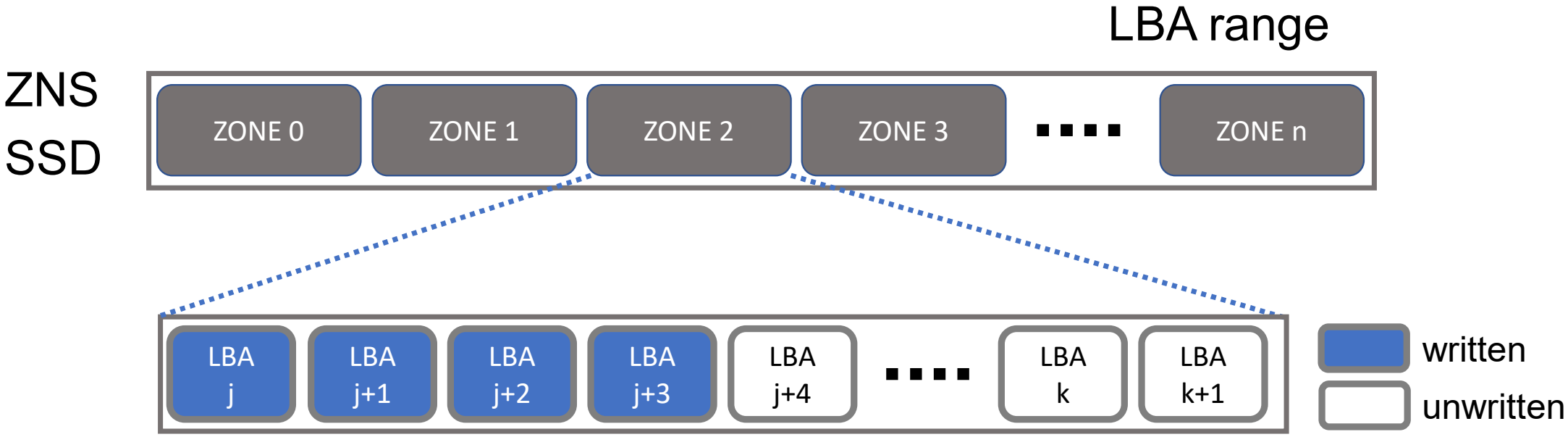
[2)]Samsung Electronics Co.

HotStorage 2023

HotStorage '23, July 9, Boston

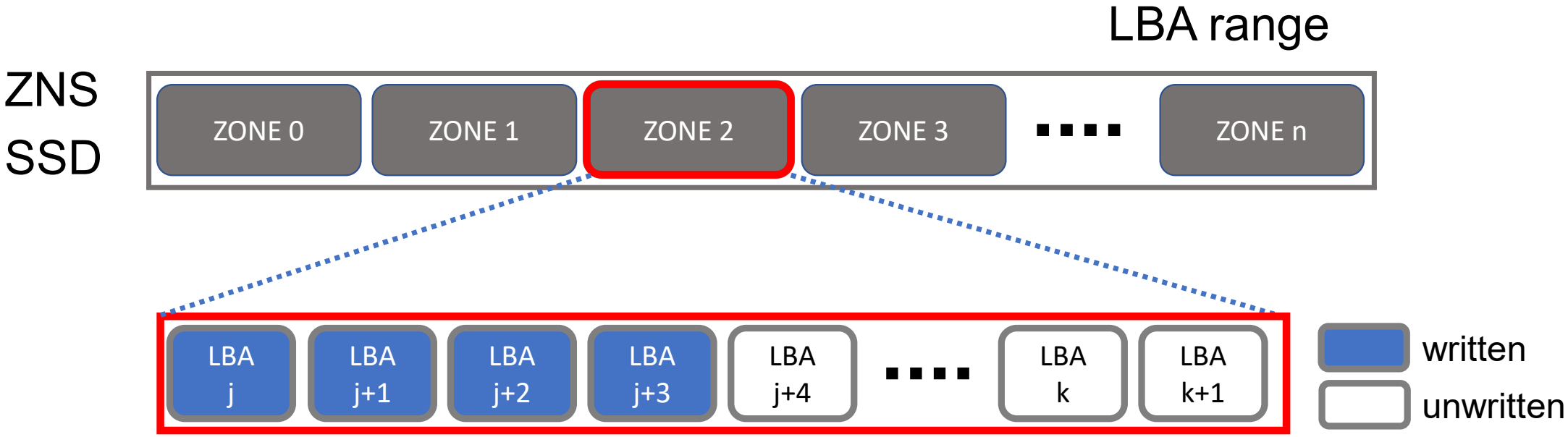
Contents

Zoned Namespace SSD(ZNS SSD)



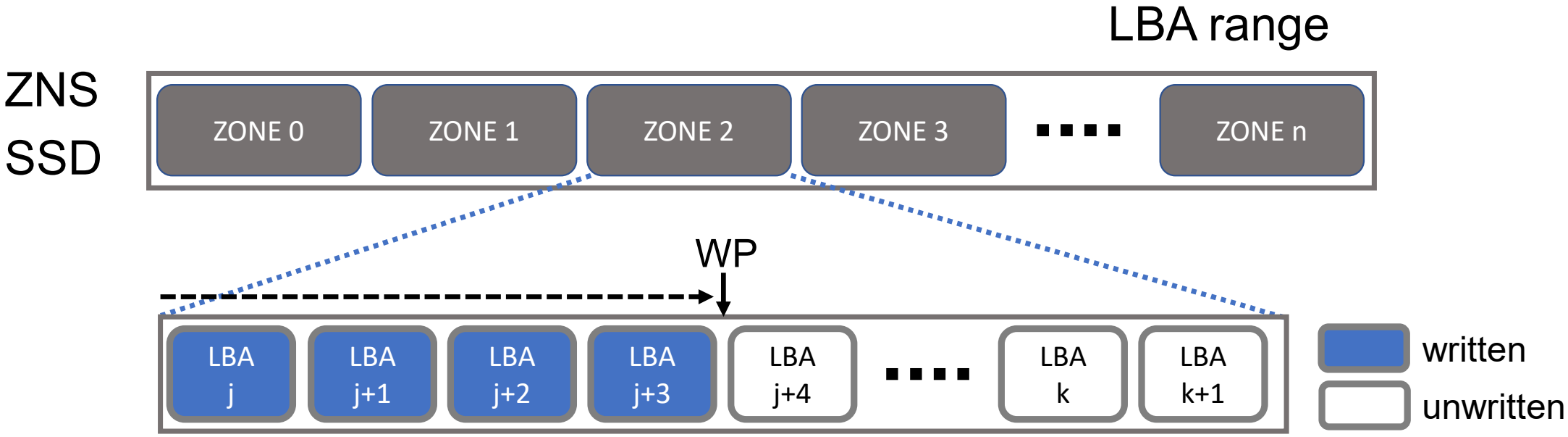
- ZNS groups the LBA space into fixed-size zones.

Zoned Namespace SSD(ZNS SSD)



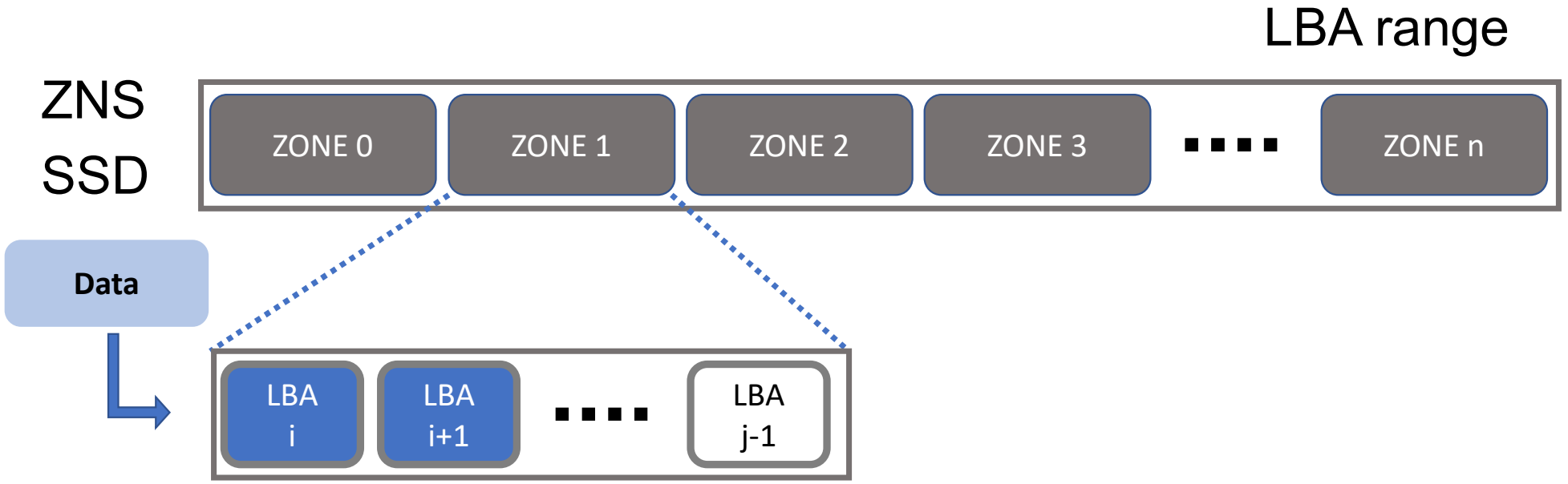
- ZNS groups the LBA space into fixed-size zones.
- Zone is an erase unit of ZNS SSD(zone-reset).

Zoned Namespace SSD(ZNS SSD)



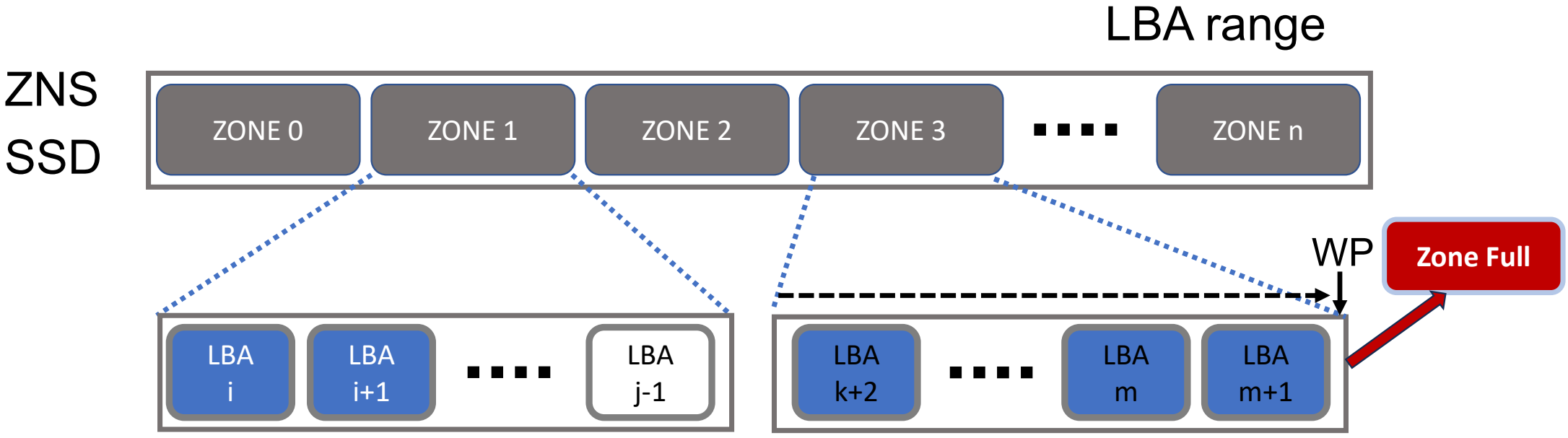
- ZNS groups the LBA space into fixed-size zones.
- Zone is an erase unit of ZNS SSD(zone-reset).
- Each zone must be written sequentially.

Zoned Namespace SSD(ZNS SSD)



- User applications take over responsibility for
  1) Data placement

Zoned Namespace SSD(ZNS SSD)



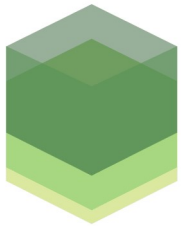
- User applications take over responsibility for
  1) Data placement
  2) **Free space reclamation**

No Garbage Collection in FTL

LSM-based Key-Value Store (LSM-KV)

- Log-structured merge-tree(LSM-tree)
  - Sequential I/O pattern
  - Out-of-place update

RocksDB



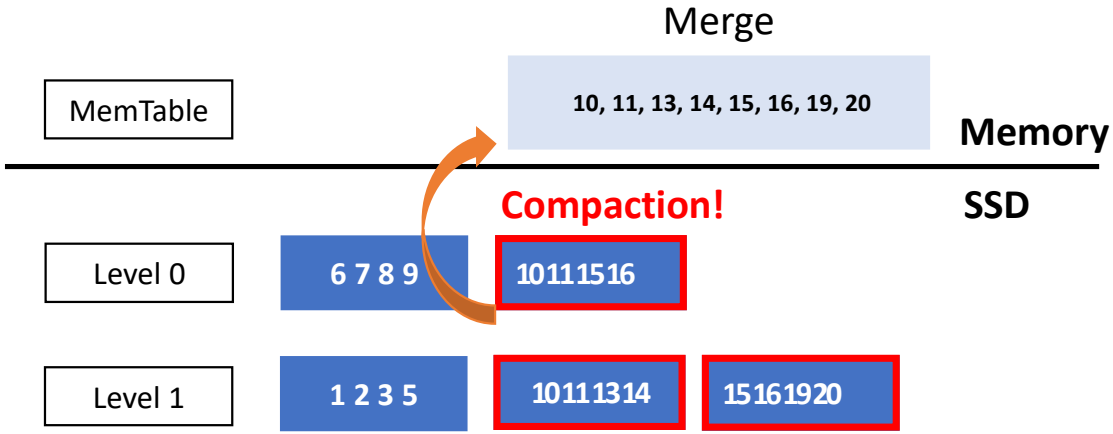
LEVELDB

mongoDB®

LSM-based Key-Value Store (LSM-KV)

- Data Update in LSM-tree
- Compaction
  - LSM-tree updates data files (SSTables) through compaction.

RocksDB

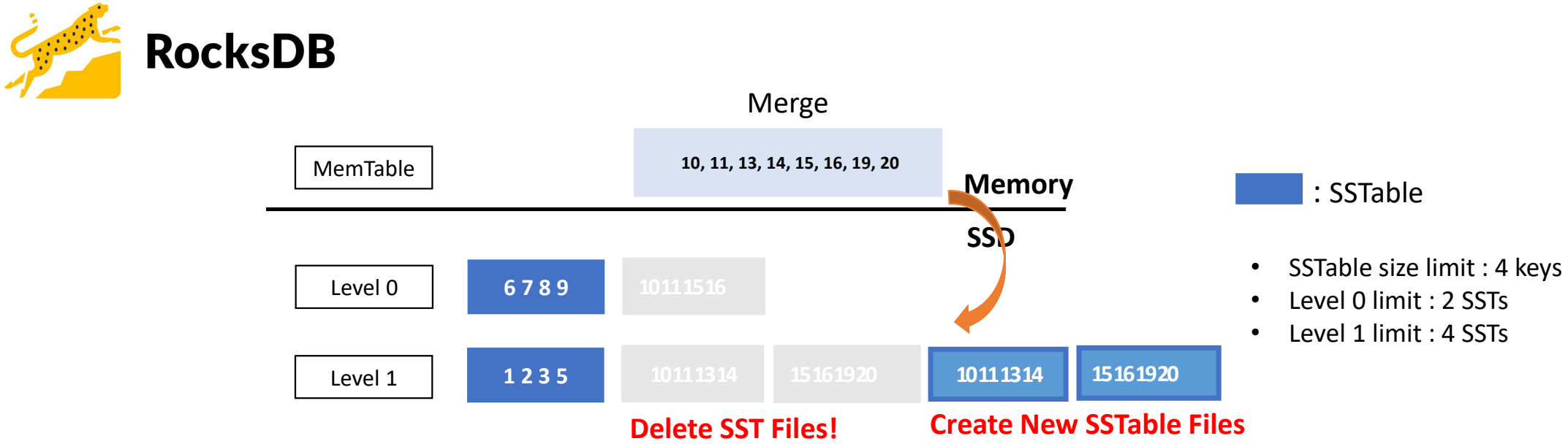


: SSTable

- SSTable size limit : 4 keys
- Level 0 limit : 2 SSTs
- Level 1 limit : 4 SSTs

LSM-based Key-Value Store (LSM-KV)

- Data Update in LSM-tree
- Compaction
  - LSM-tree updates data files (SSTables) through compaction.



ZenFS [ATC'21]

1) Bjørling, Matias, et al. *ZNS: Avoiding the Block Interface Tax for Flash-based SSDs, ATC'21*

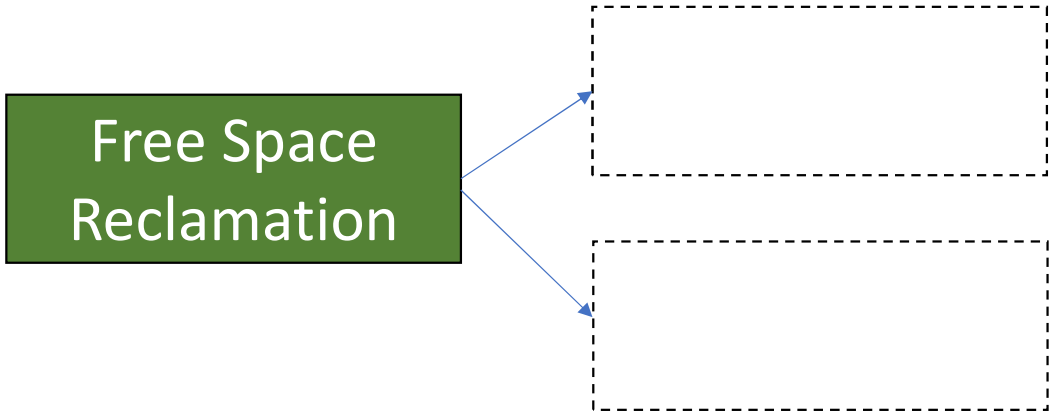
ZenFS [ATC'21]

- User-level file system for LSM-KV
  - Backend module for RocksDB
- Responsible for
  - Zone Allocation (Data placement)
  - Free space reclamation

1) Bjørling, Matias, et al. *ZNS: Avoiding the Block Interface Tax for Flash-based SSDs, ATC'21*

ZenFS [ATC'21]

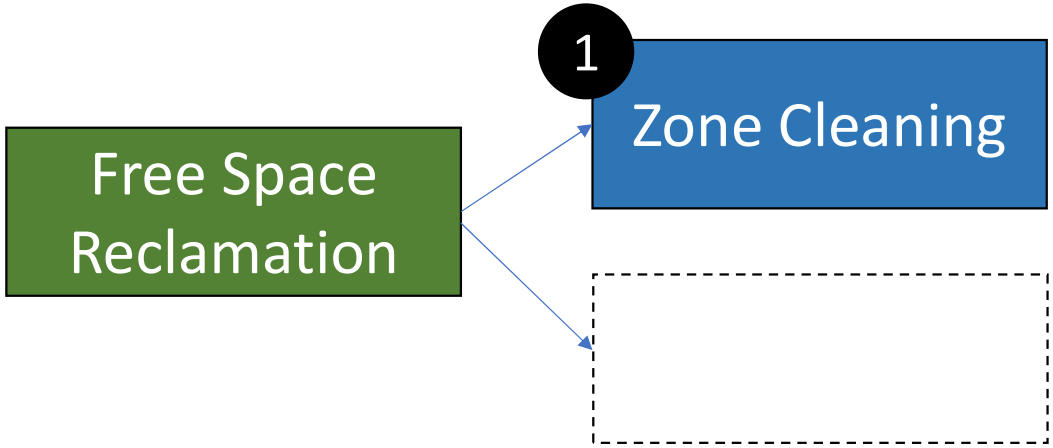
- User-level file system for LSM-KV
  - Backend module for RocksDB
- Responsible for
  - Zone Allocation (Data placement)
  - **Free space reclamation**



1) Bjørling, Matias, et al. *ZNS: Avoiding the Block Interface Tax for Flash-based SSDs, ATC'21*

ZenFS [ATC'21]

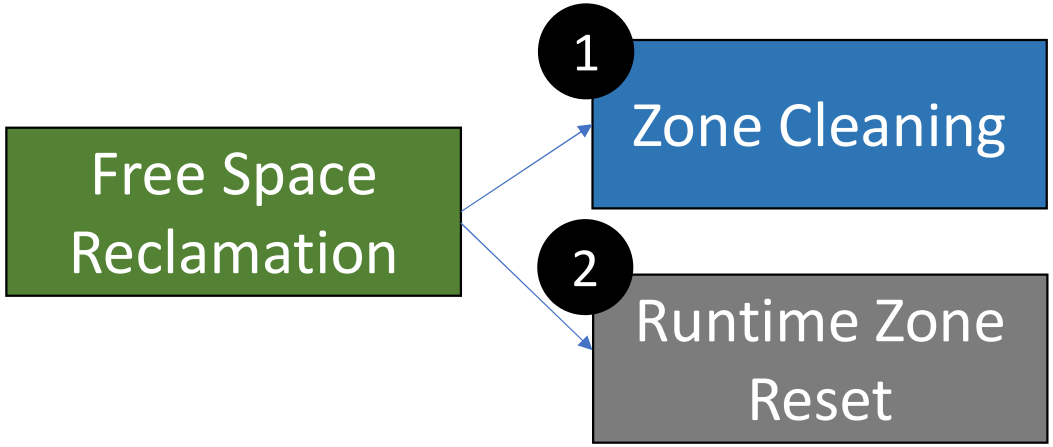
- User-level file system for LSM-KV
  - Backend module for RocksDB
- Responsible for
  - Zone Allocation (Data placement)
  - **Free space reclamation**



1) Bjørling, Matias, et al. *ZNS: Avoiding the Block Interface Tax for Flash-based SSDs, ATC'21*

ZenFS [ATC'21]

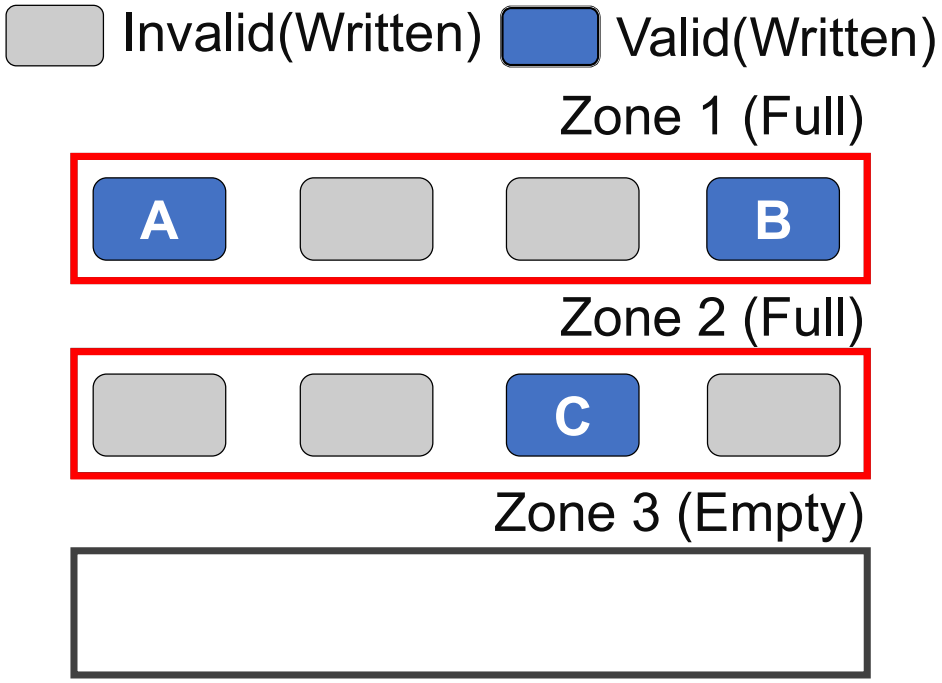
- User-level file system for LSM-KV
  - Backend module for RocksDB
- Responsible for
  - Zone Allocation (Data placement)
  - **Free space reclamation**



1) Bjørling, Matias, et al. *ZNS: Avoiding the Block Interface Tax for Flash-based SSDs, ATC'21*

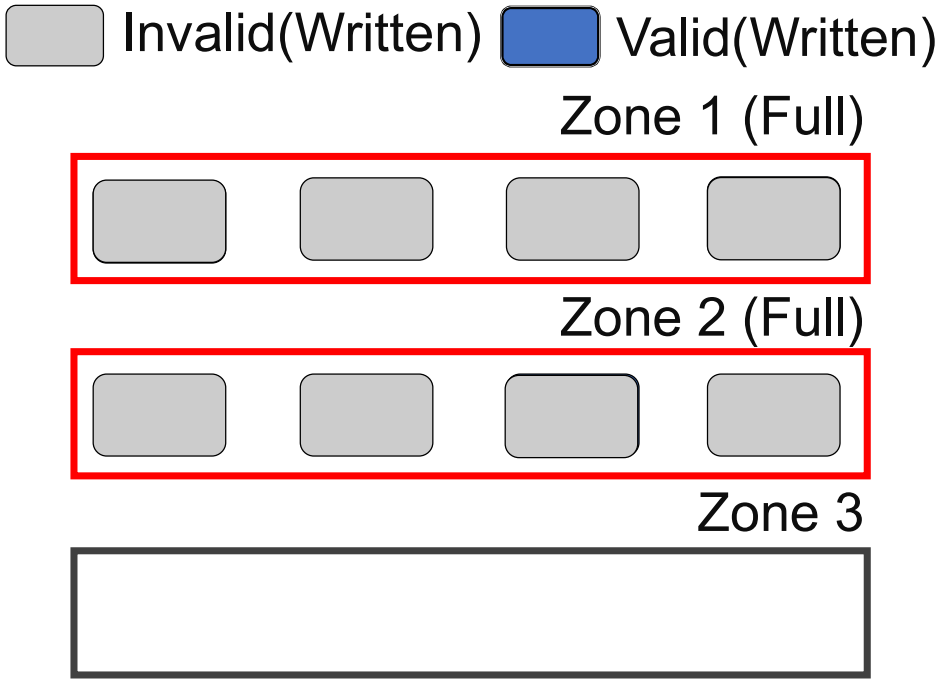
Contents

(1) Zone Cleaning



1) Victim Zone(s) Selection

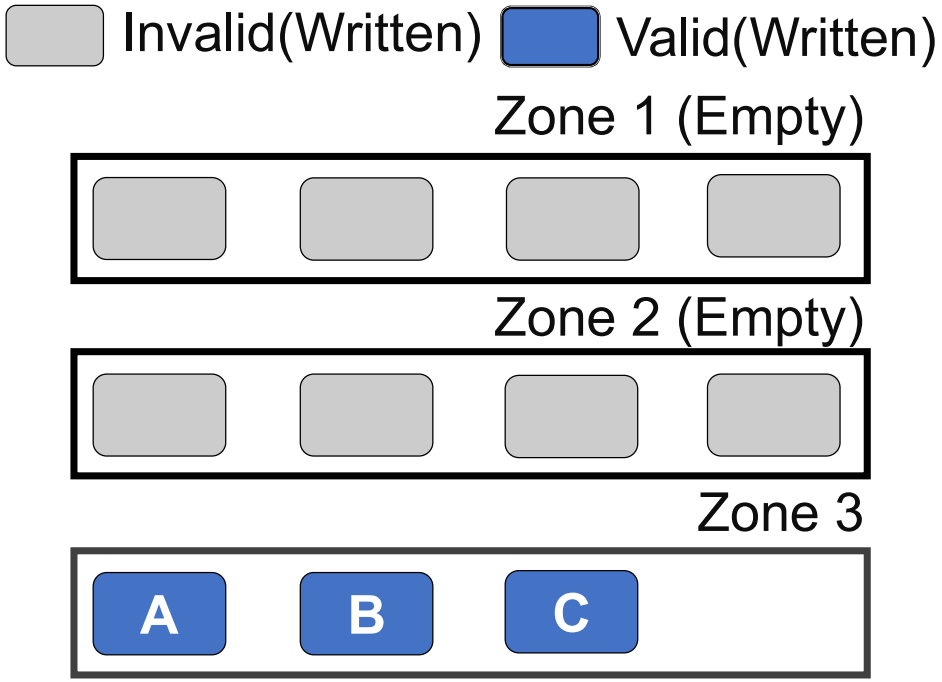
(1) Zone Cleaning



1) Victim Zone(s) Selection

2) Valid Data Copy

(1) Zone Cleaning

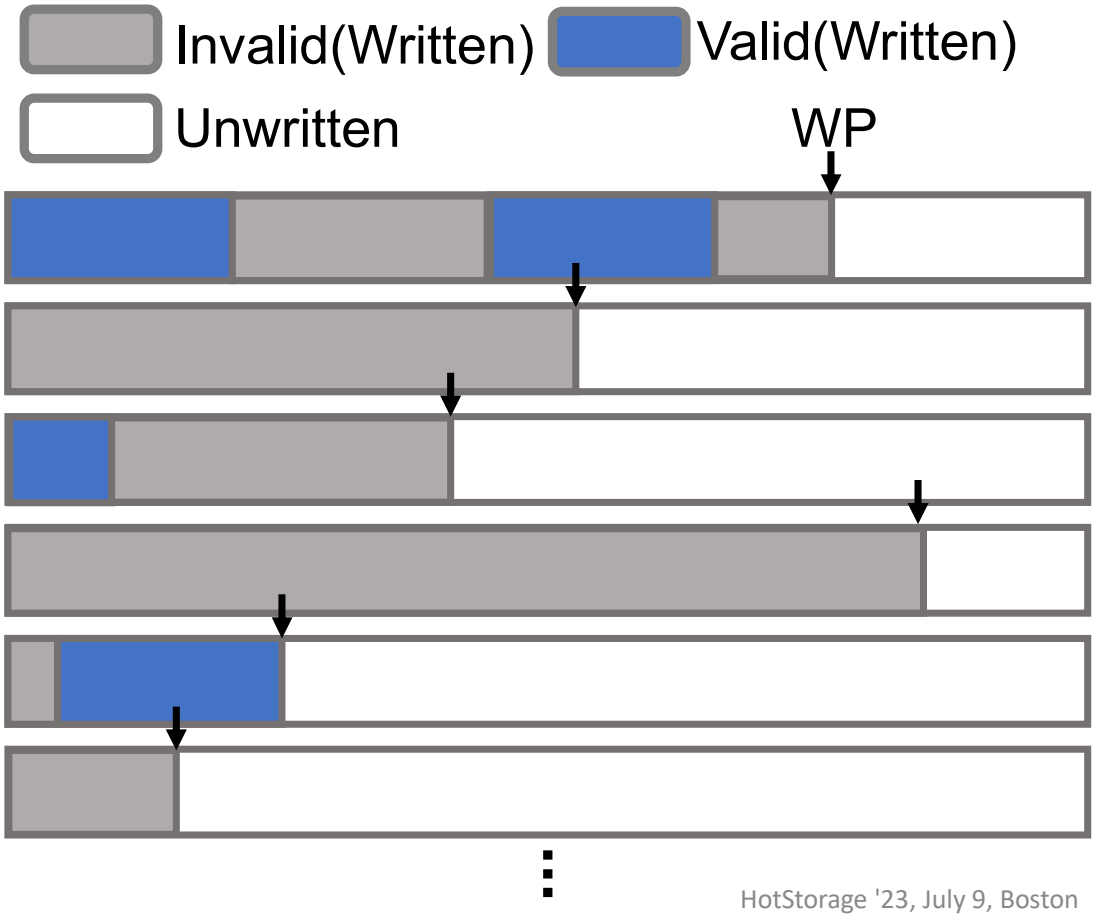


1) Victim Zone(s) Selection

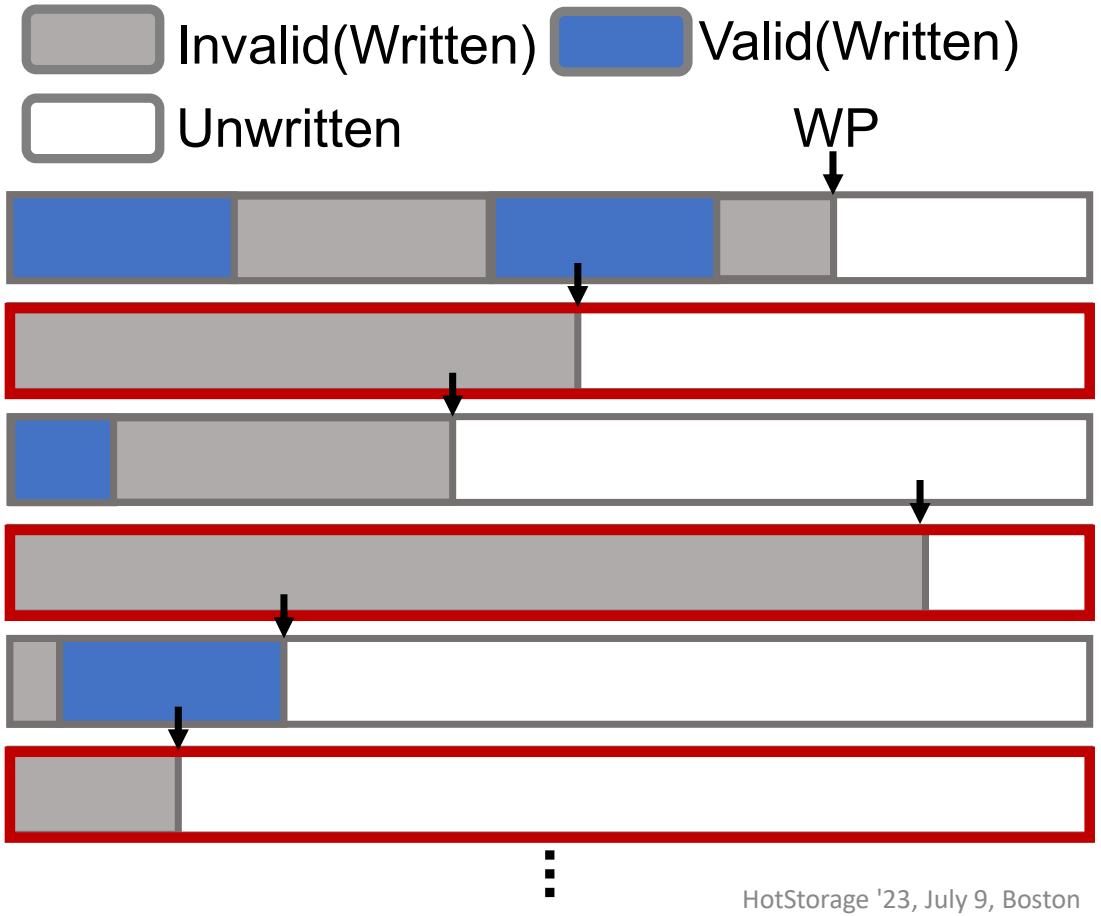
2) Valid Data Copy

3) Erase zones by zone-reset

(2) Runtime Zone-Reset

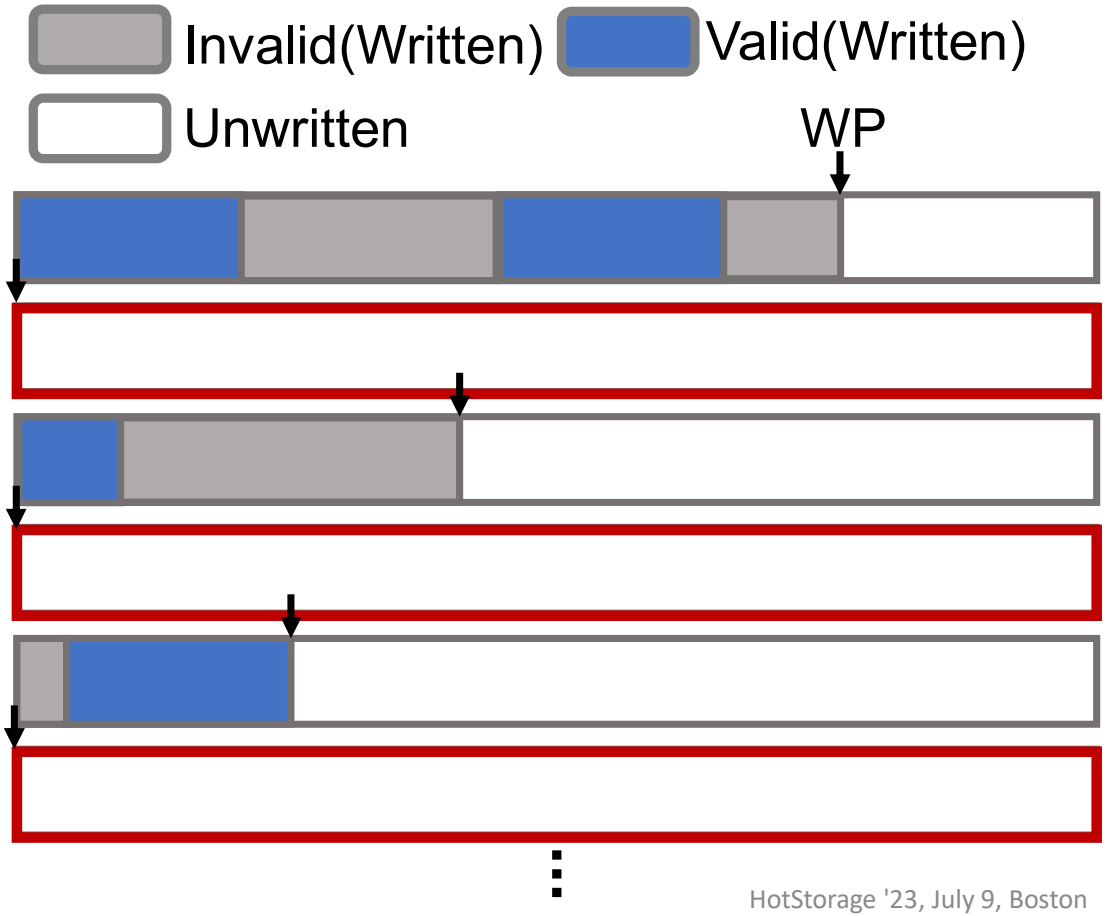


(2) Runtime Zone-Reset



1) Search for zones with all invalid data

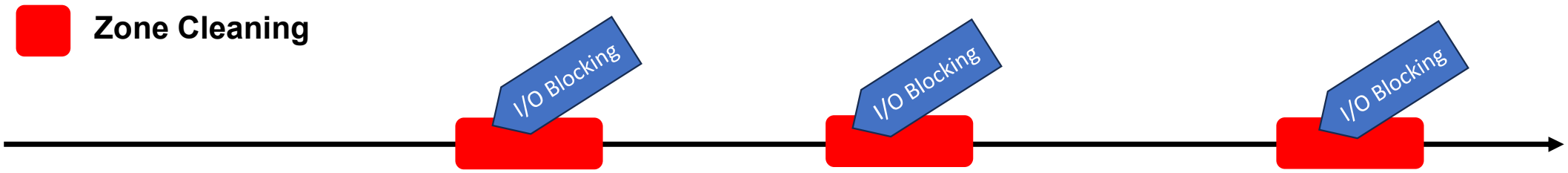
(2) Runtime Zone-Reset



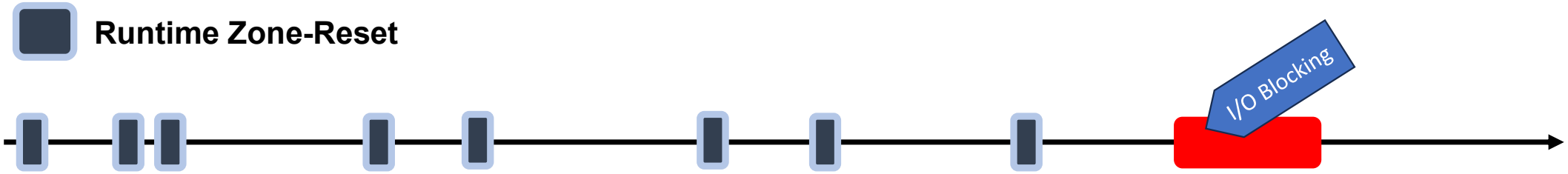
1) Search for zones with all invalid data

2) Erase zones by zone-reset

Zone Cleaning vs. Runtime Zone-Reset



- Zone Cleaning calls increase I/O blocking time, resulting poor performance.

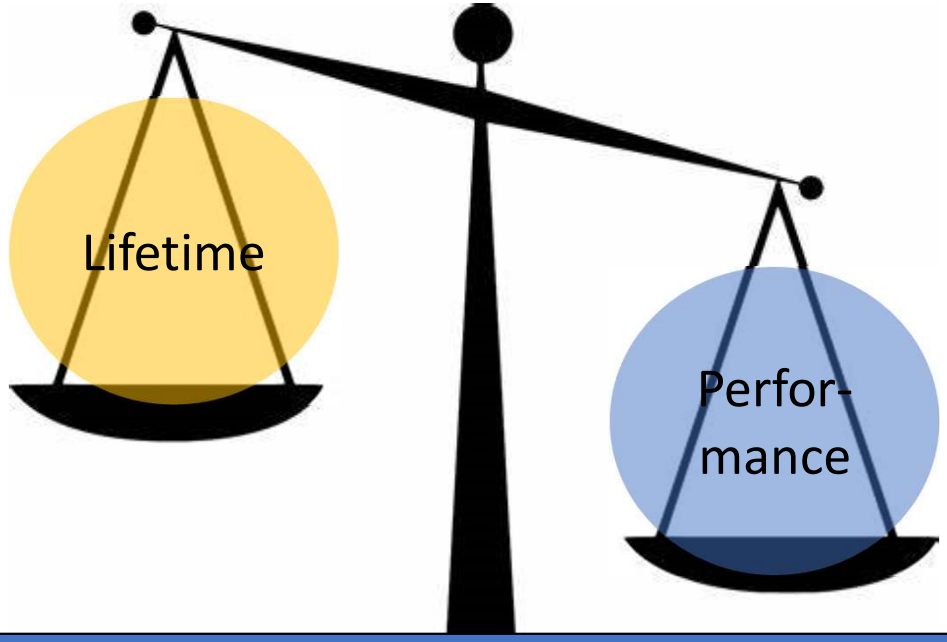


- Runtime Zone-Reset minimizes performance degradation by reducing Zone Cleaning calls.

Problem Definition

- The Runtime-Zone Reset algorithm unnecessarily performs zone-reset, even though there is enough free space in the ZNS SSD.
- This **shortens the lifespan** of ZNS SSD by executing NAND block erase frequently.
- However, excessively reducing Runtime-Zone Reset calls causes frequent ZC calls, resulting in **performance degradation**.

Tradeoffs



Is it possible to balance the lifespan and performance of the ZNS by carefully controlling the Runtime Zone-Reset call?

Contents

Key Ideas

- FAR **<u>determines whether to execute zone-reset</u>** according to a threshold (T_{wp}).

Key Ideas

- FAR **<u>determines whether to execute zone-reset</u>** according to a threshold (T_{wp}).
- More importantly, threshold (T_{wp}) changes depending on the **<u>free space</u>** remaining in the ZNS SSD.

Threshold (T_{wp}) $\propto$

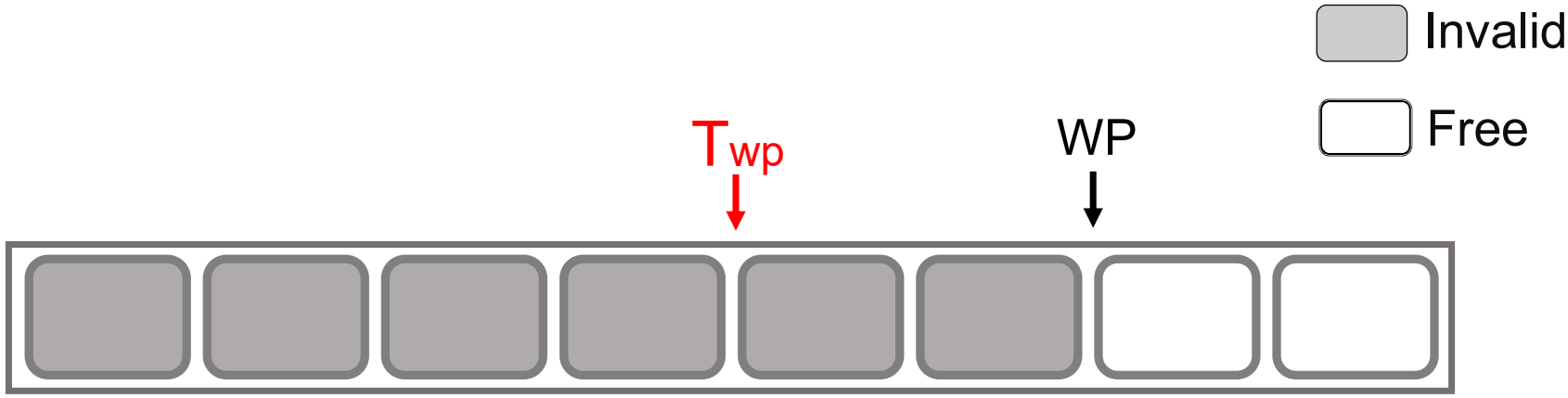
T_{wp} increases with free space increase!

Ready to execute "zone-reset" in runtime!

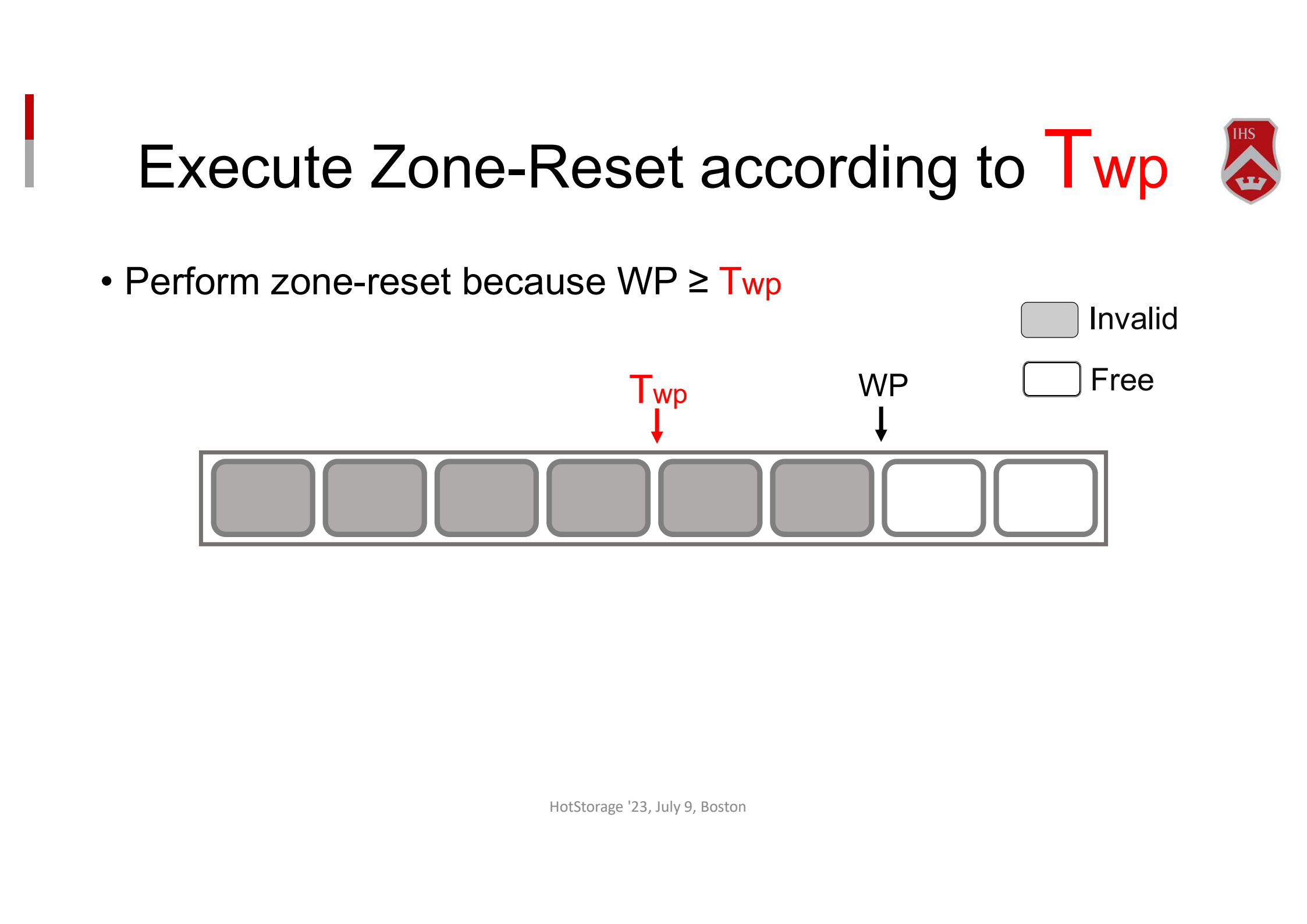
Key Ideas

- FAR determines whether to execute zone-reset according to a threshold (T_{wp}).
- More importantly, threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.

Execute Zone-Reset according to T_{wp}



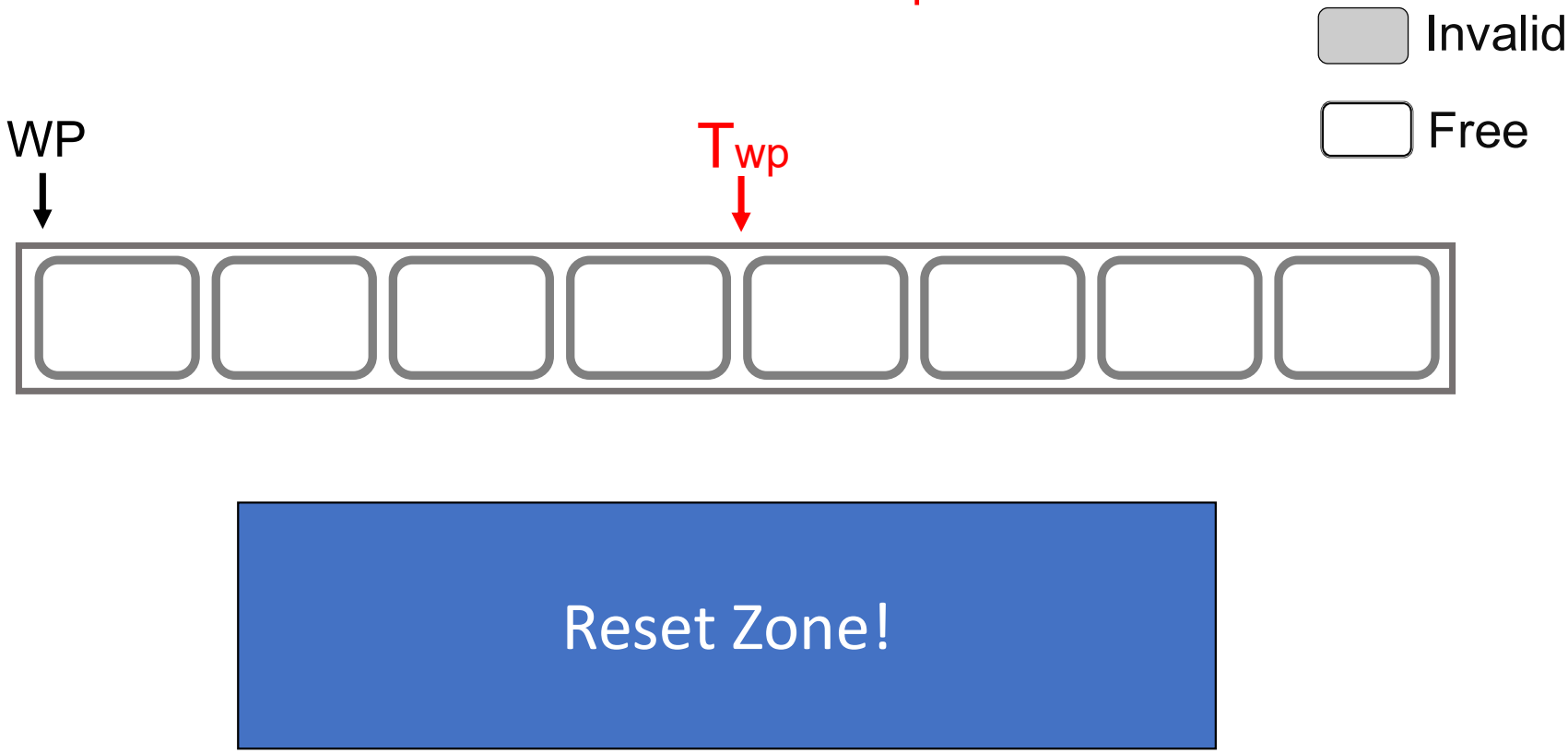
Execute Zone-Reset according to T_{wp}



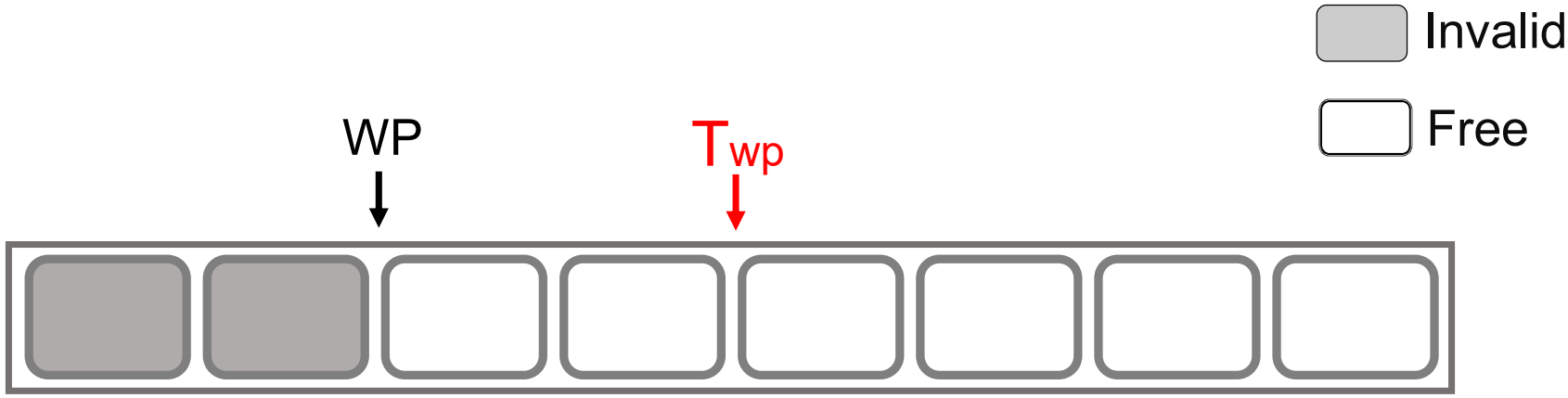
- Perform zone-reset because WP ≥ T_{wp}

Execute Zone-Reset according to T_{wp}

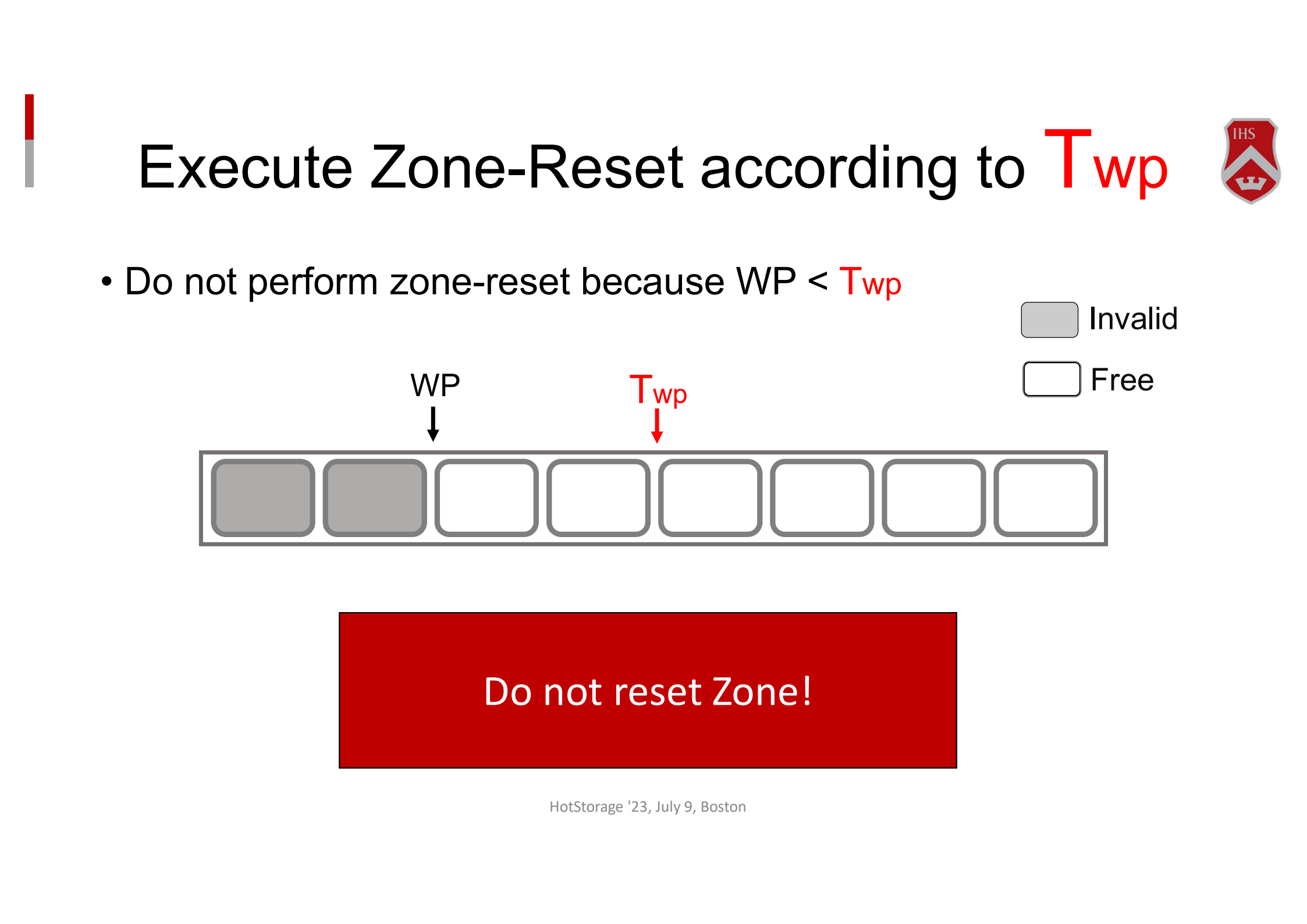
- Perform zone-reset because WP ≥ T_{wp}



Execute Zone-Reset according to T_{wp}



Execute Zone-Reset according to T_{wp}



- Do not perform zone-reset because WP < T_{wp}

Do not reset Zone!

Key Ideas

- FAR determines whether to execute zone-reset according to a threshold (T_{wp}).

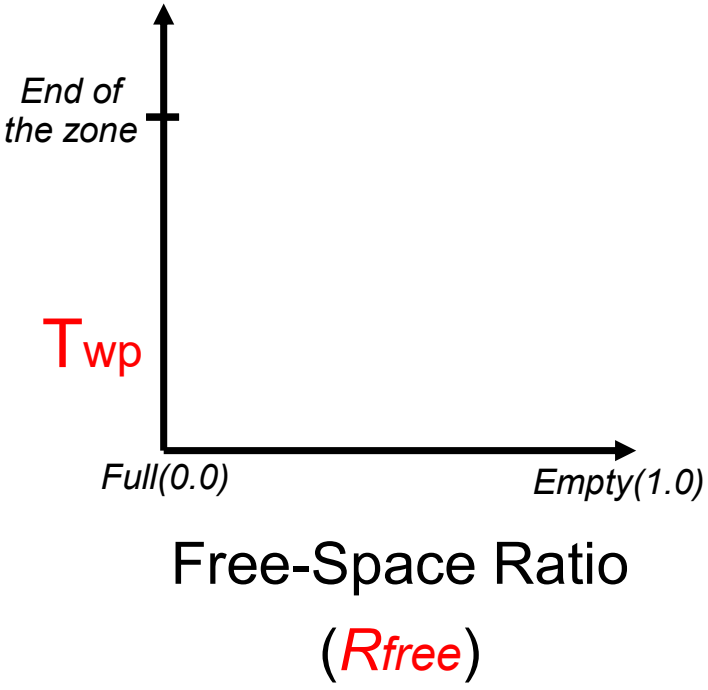
- More importantly, threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.

Change of T_{wp} according to Free Space

- Threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.

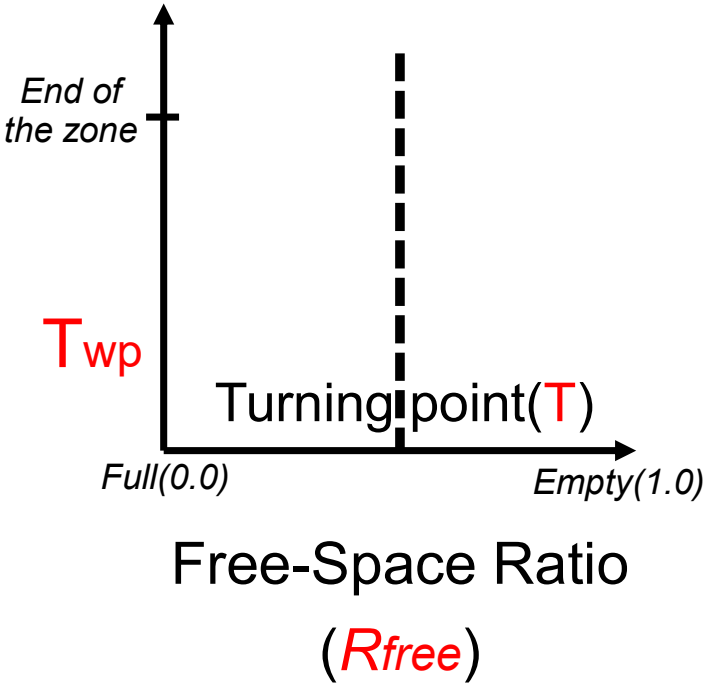
Change of T_{wp} according to Free Space

- Threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.

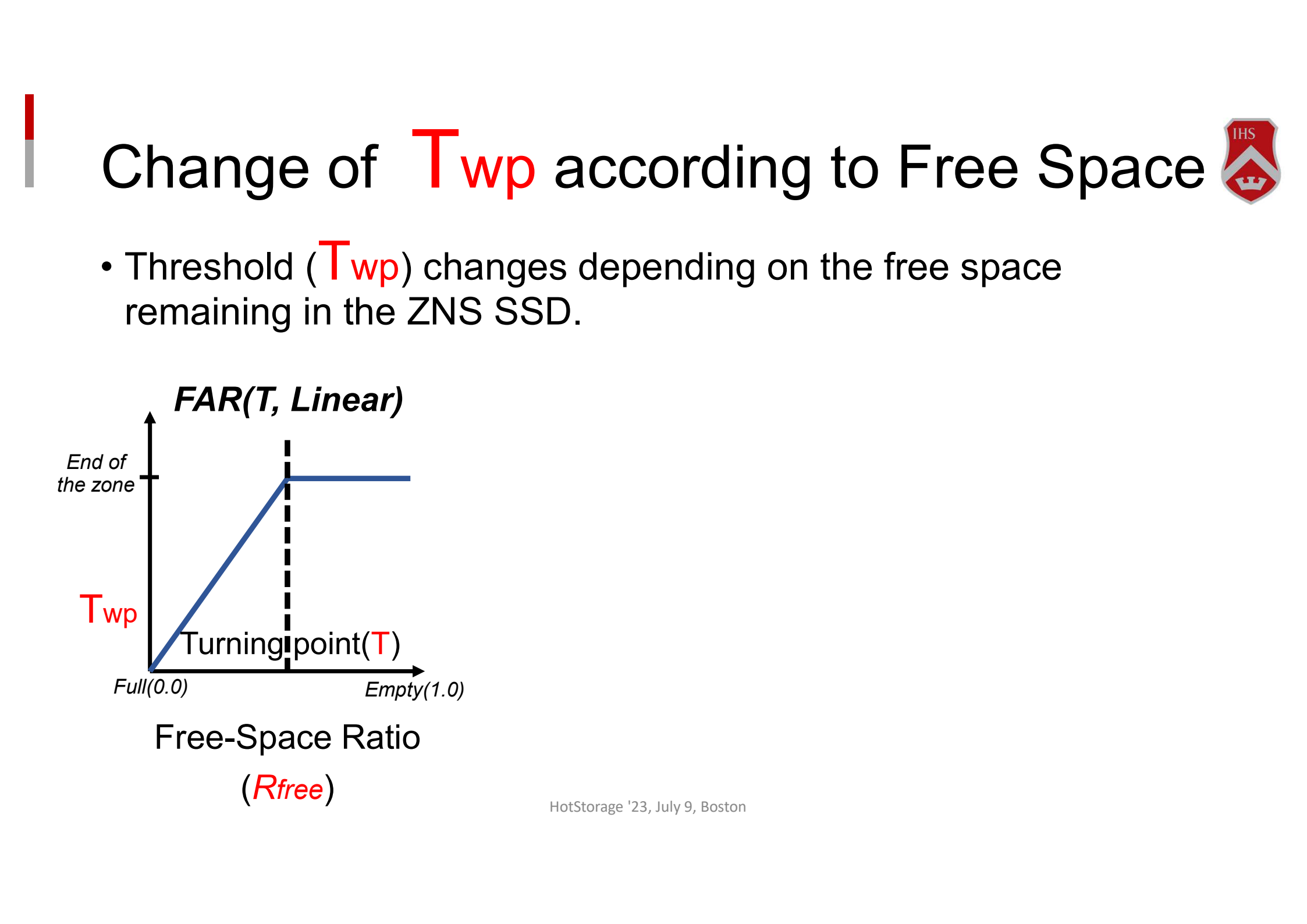


Change of T_{wp} according to Free Space

- Threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.



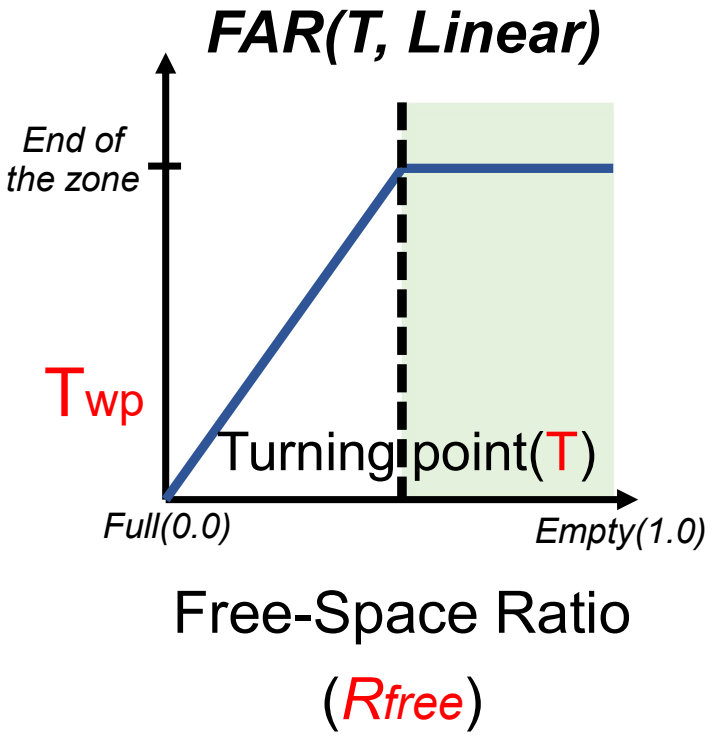
Change of T_{wp} according to Free Space



- Threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.

Change of T_{wp} according to Free Space

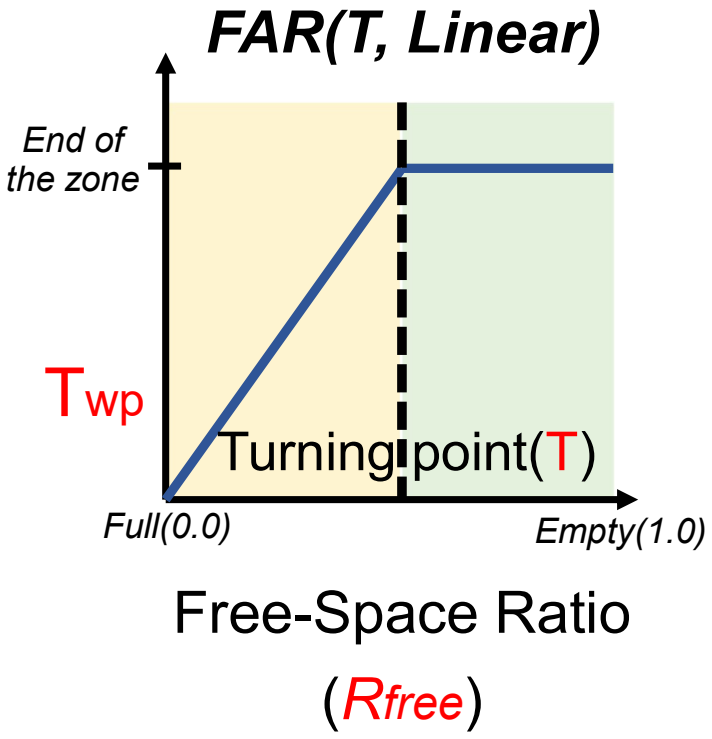
- Threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.



- If free space (R_{free}) > T, threshold (T_{wp}) is set to be the end of a zone.

Change of T_{wp} according to Free Space

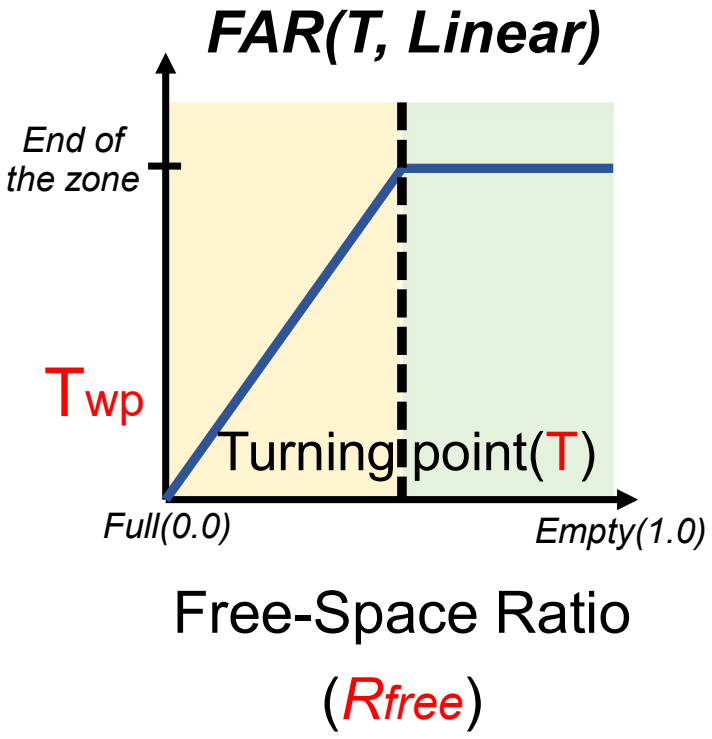
- Threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.



- If free space (R_{free}) > T, threshold (T_{wp}) is set to be the end of a zone.
- Else, threshold (T_{wp}) is a linear function of (R_{free}).

Change of T_{wp} according to Free Space

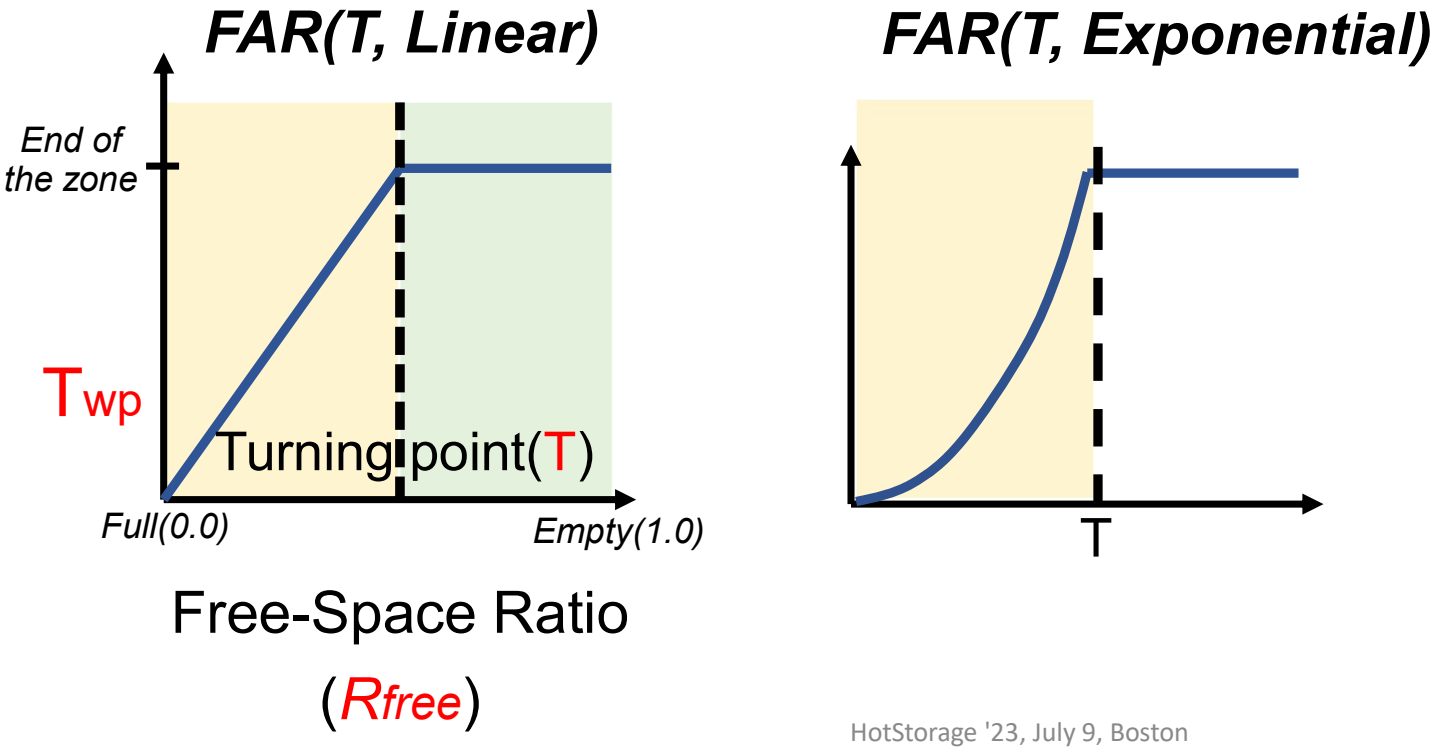
- Threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.



- If free space (R_{free}) > T, threshold (T_{wp}) is set to be the end of a zone.
- Else, threshold (T_{wp}) is a linear function of (R_{free}).
- In our experiment, we used 0.7 for T.

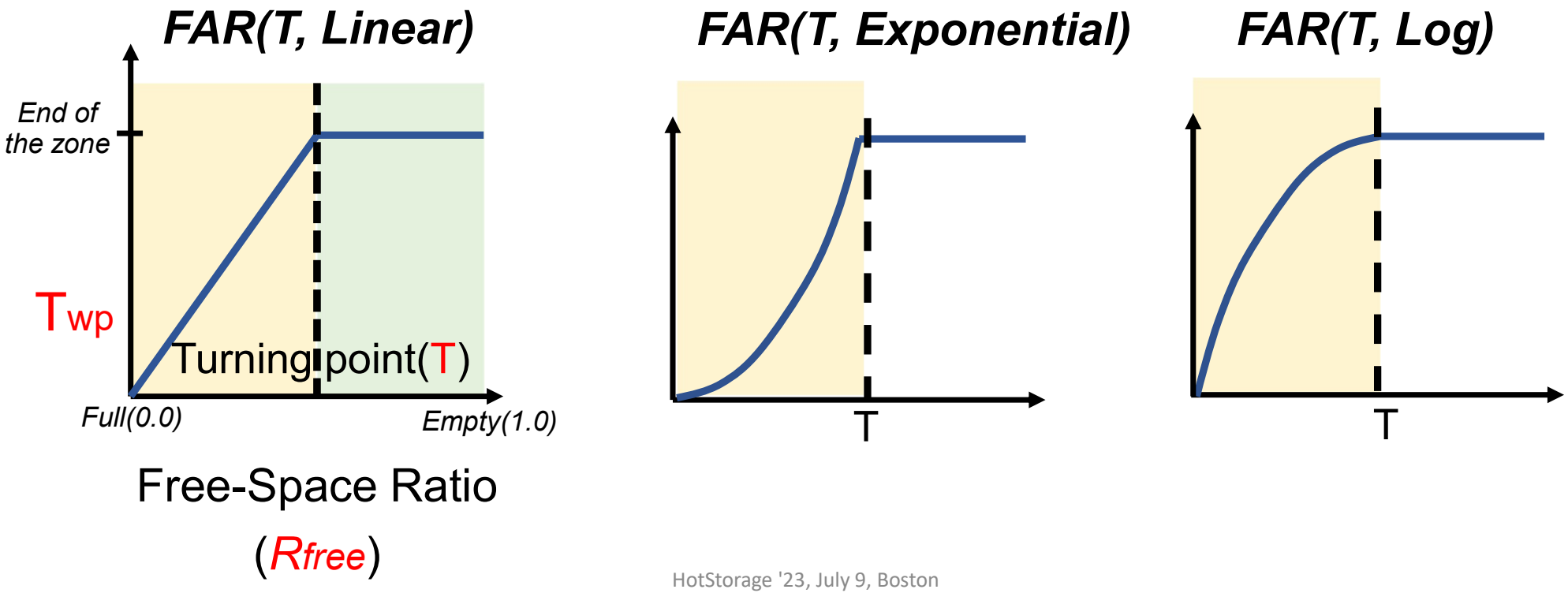
Change of T_{wp} according to Free Space

- Threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.



Change of T_{wp} according to Free Space

- Threshold (T_{wp}) changes depending on the free space remaining in the ZNS SSD.



Contents

Experimental Setup

HOST

- i7-4790 3.60Ghz (8 cores)
- Linux Kernel 5.18.0
- UBUNTU 20.04
- RocksDB v.7.4.4
- ZenFS v2.0

ZNS

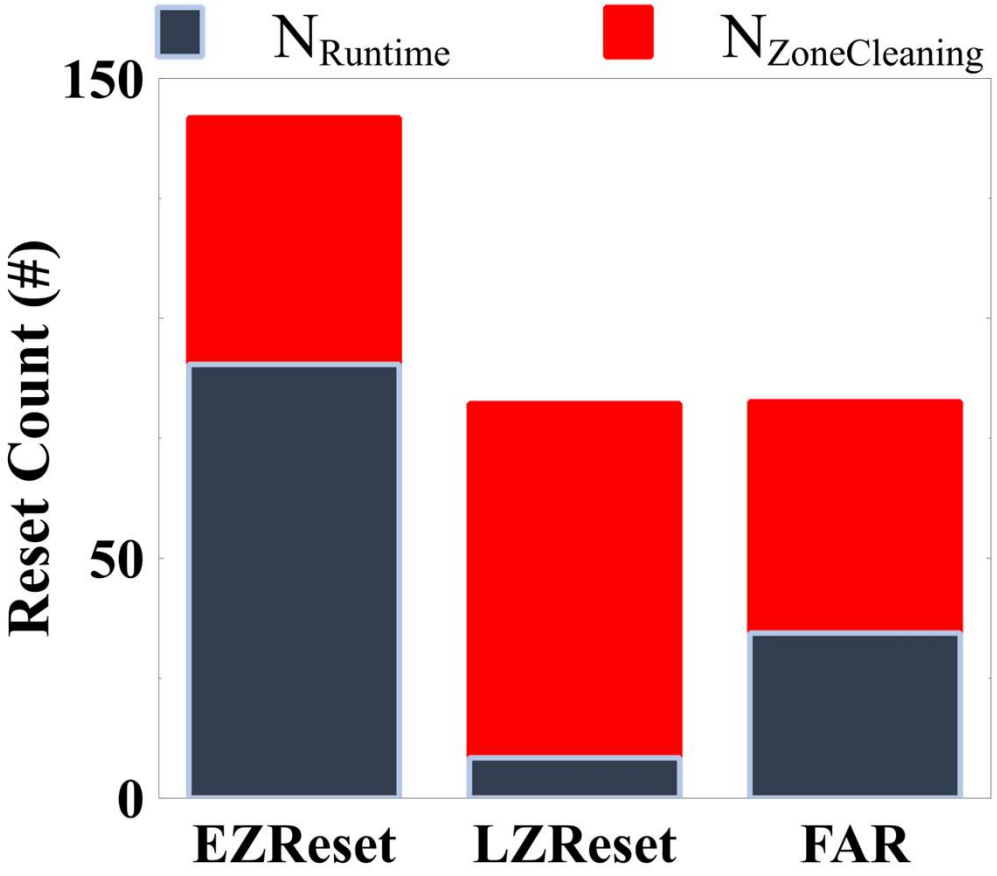
- ZNS Prototype
- Cosmos+ OpenSSD
- 512MB Fixed-size zone
- 40 I/O zones
- 20GB Total Capacity

Experimental Setup

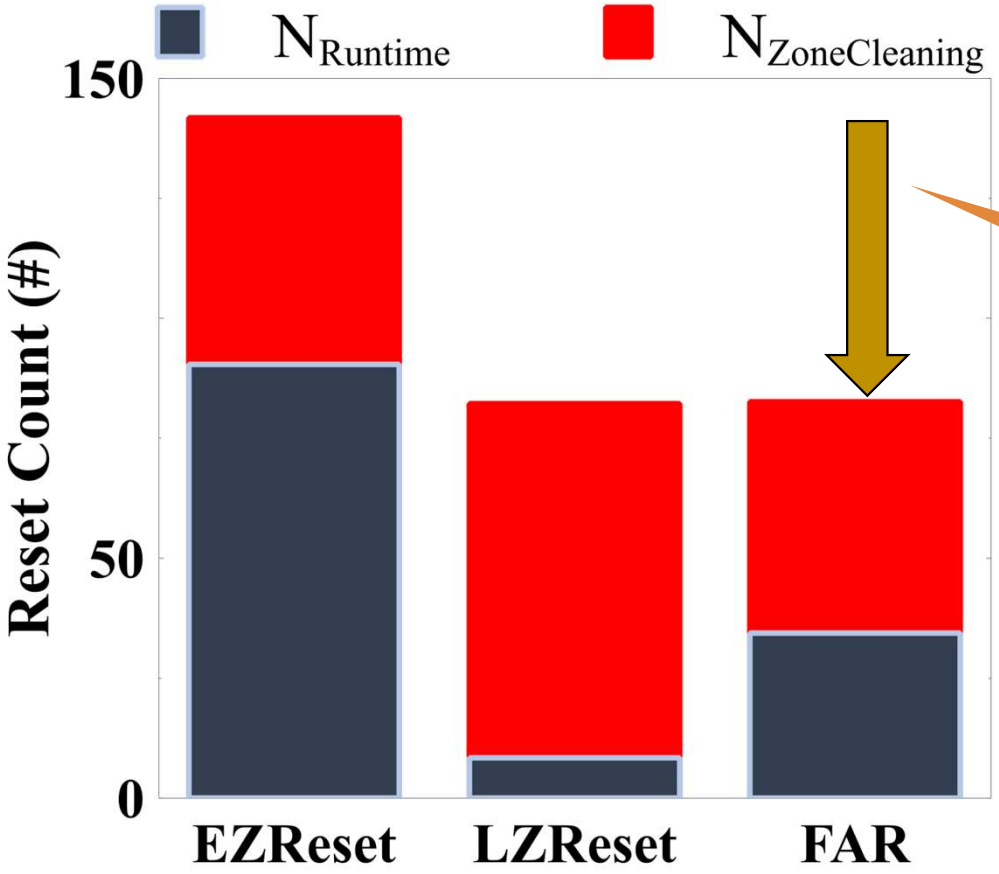
- Workloads
  - Used Rocksdb's db_bench "fillrandom" workloads
  - Used different data size: Small(9GB), Medium(12GB), Large(15GB)

- Comparisons
  - EZReset: Default runtime zone-reset in ZenFS
  - LZReset: Lazy runtime zone-reset with T_{wp} = *end of a zone*
  - FAR (T, $func$): free-space adaptive zone-reset algorithm, where T=0.7 and func=Linear

Lifetime Results

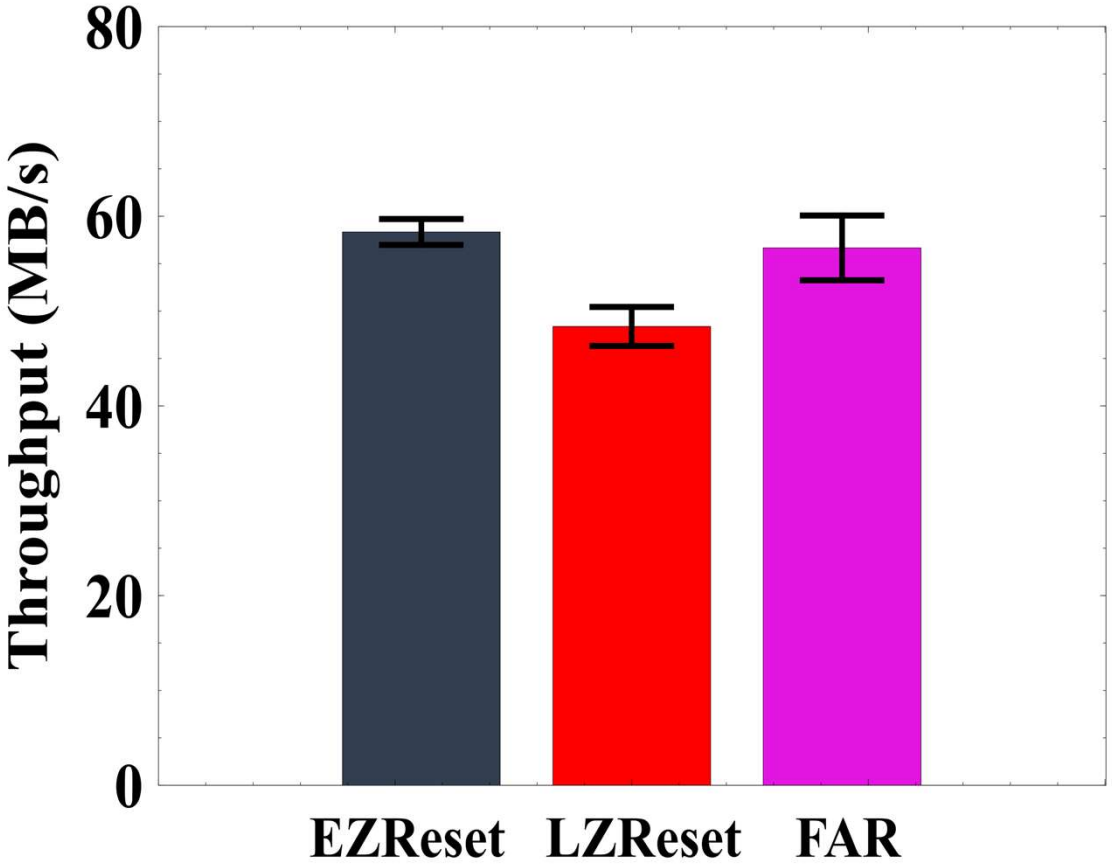


Lifetime Results

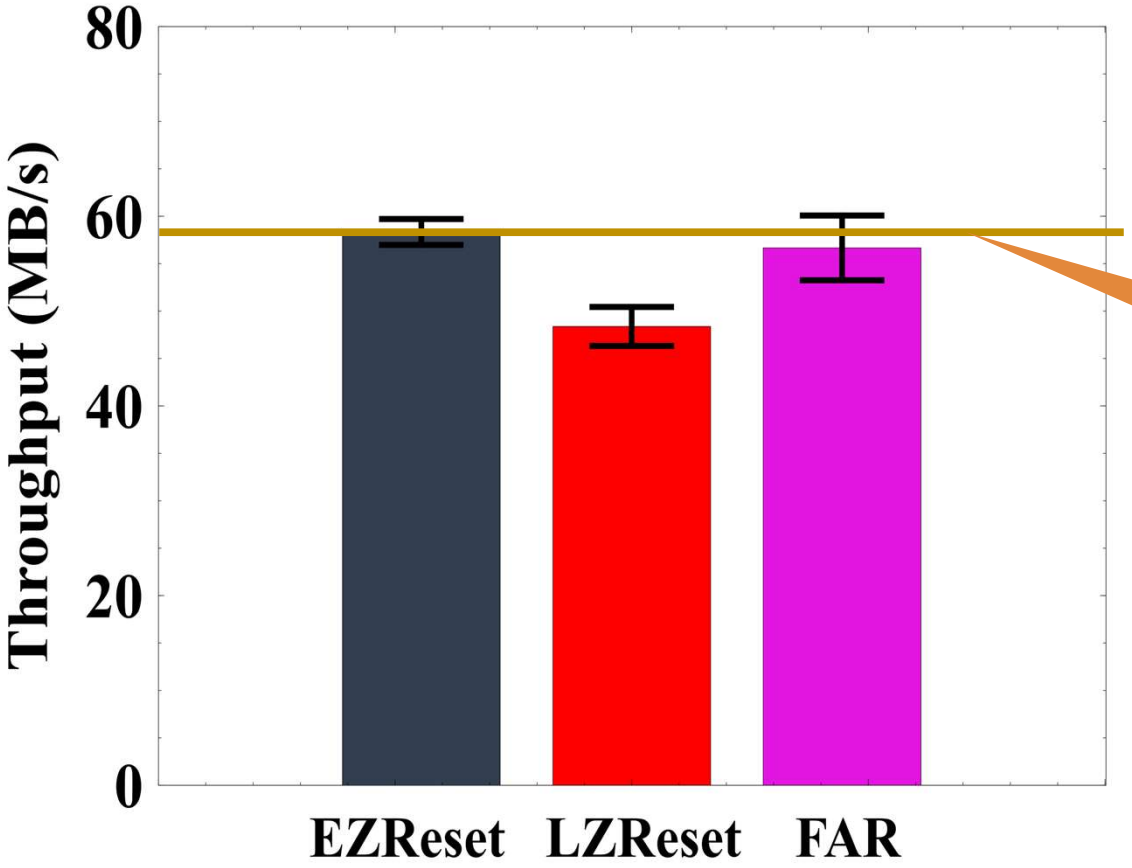


2x Lifetime Improvement

Performance Results



Performance Results



Minimal Performance Degradation

Contents

Conclusion

- We identified that the current implementation of ZenFS's runtime zone-reset algorithm can have a negative impact on device's lifetime.
- We proposed a free-space adaptive runtime zone-reset (FAR) algorithm that balances application's performance and device's lifetime.
- FAR maintains a high performance while improving device's lifetime as factor of 2.

Thank you for listening!
Q&A

Sungjin Byeon

sg20180546@gmail.com

sjbyeon@sogang.ac.kr

SOGANG UNIVERSITY