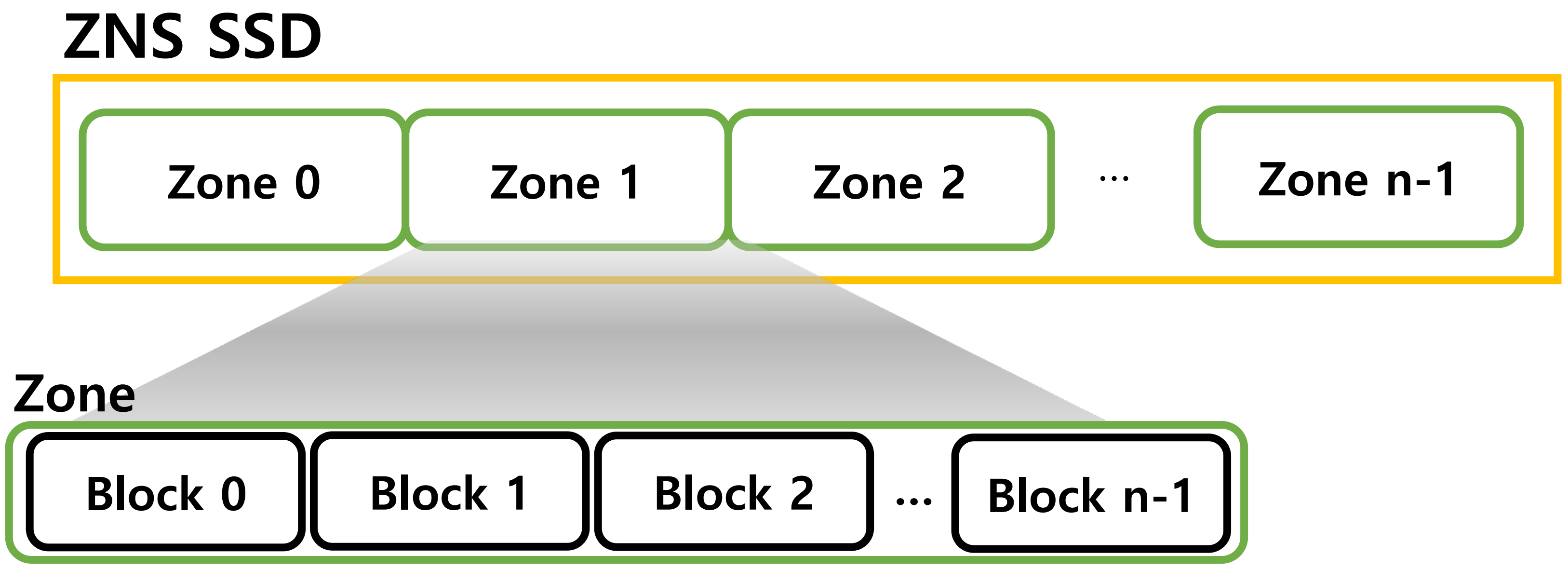
Compaction-Aware Zone Allocation for LSM based Key-Value Store on ZNS SSDs

Hee-Rock Lee, Chang-Gyu Lee, Seungjin Lee, and Youngjae Kim

Sogang University, Seoul, South Korea

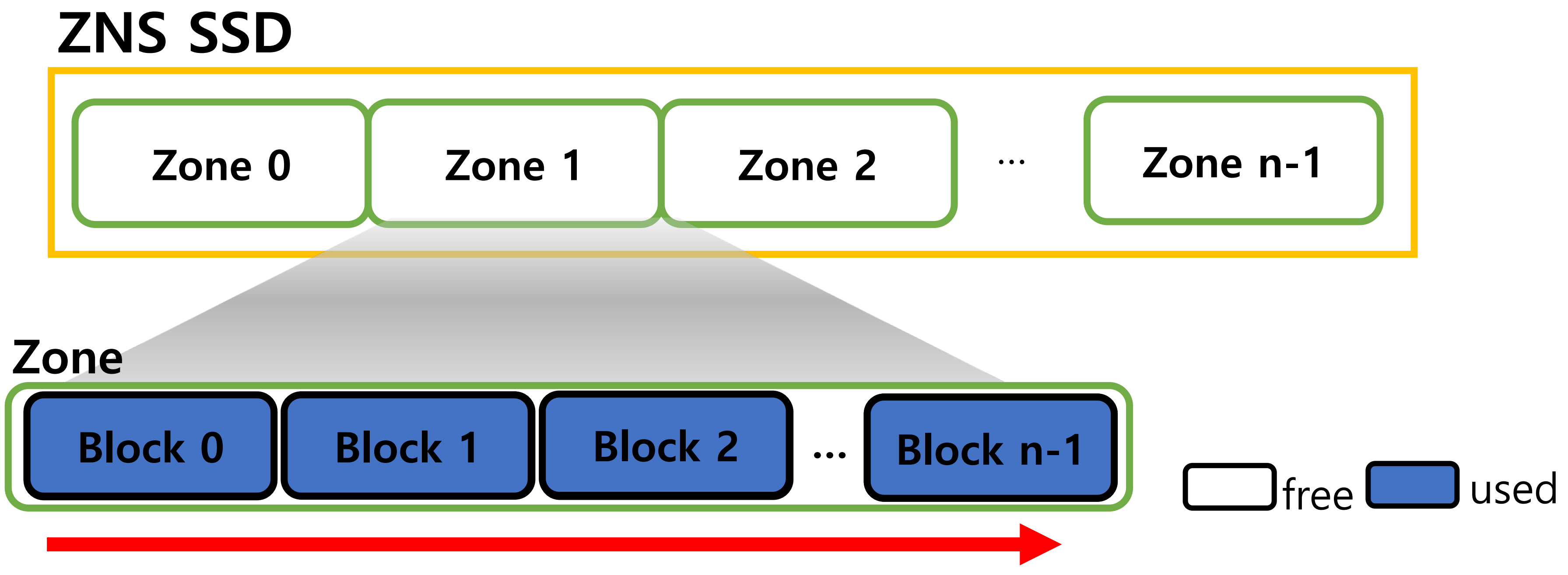
The 14th ACM Workshop on Hot Topics in Storage and File Systems
(HotStorage'22, June 27-28)

Zoned Namespace SSD(ZNS SSD)



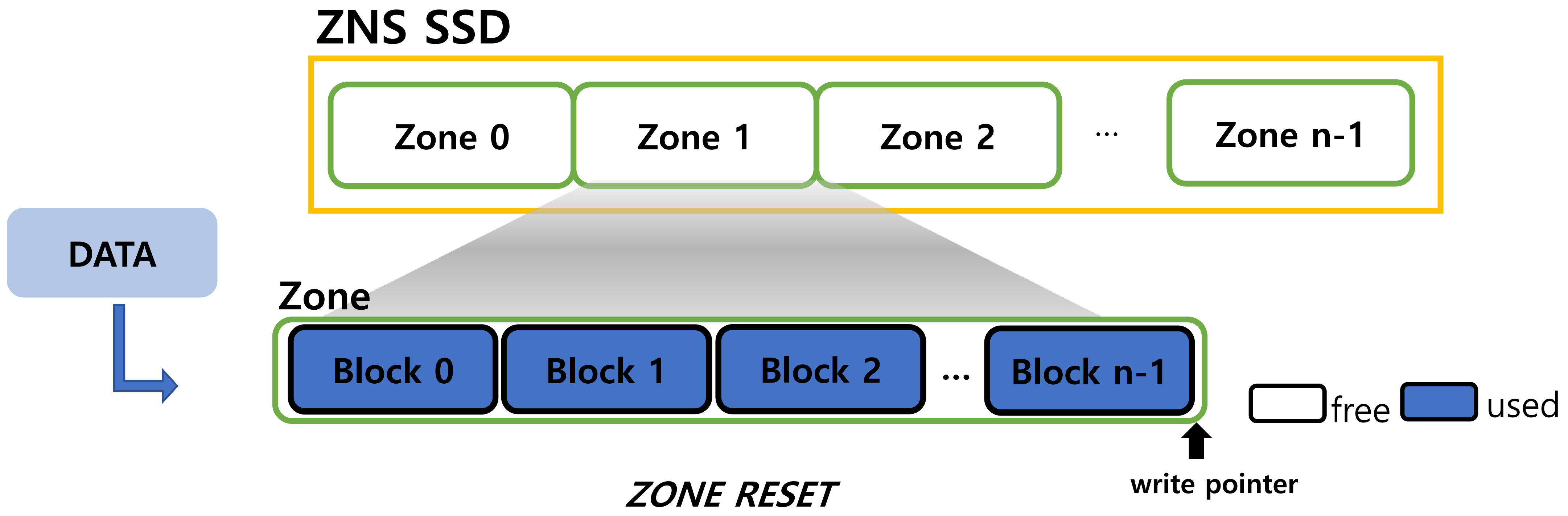
- ❑ Group multiple logical blocks into a **zone**
- ❑ Zone is an erase unit of ZNS SSD.

Zoned Namespace SSD(ZNS SSD)



❑ Zone allows only sequential writes.

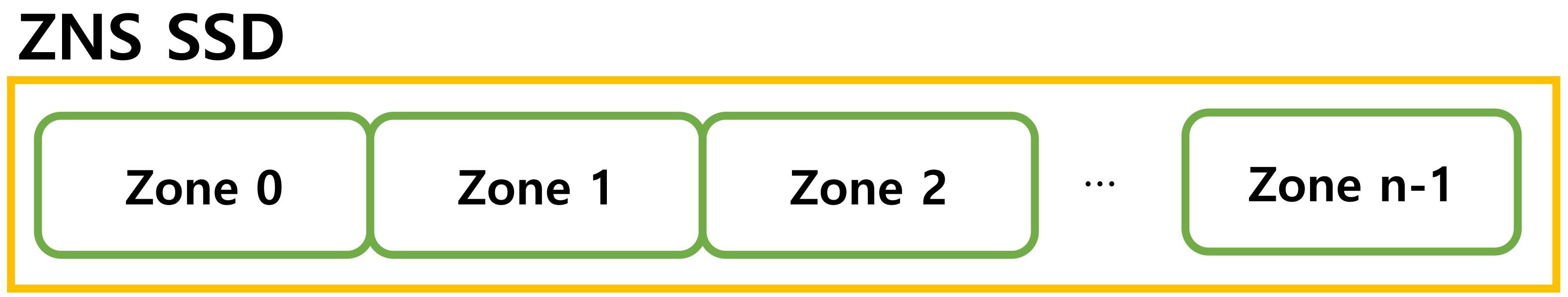
Zoned Namespace SSD(ZNS SSD)



❑ Zone allows only sequential writes.

❑ Zone disallows overwrite on same logical blocks.

Zoned Namespace SSD(ZNS SSD)



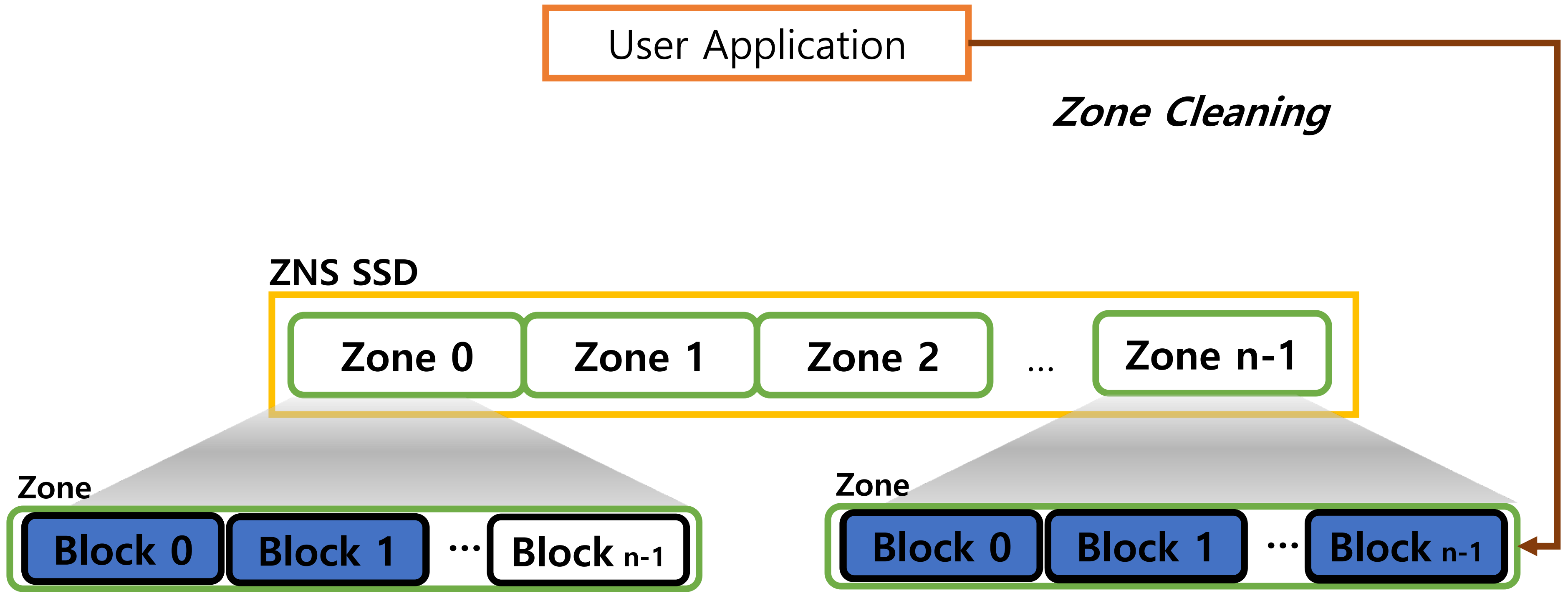
No More Garbage Collection in FTL

- ❑ Zone allows only sequential writes.
- ❑ Zone disallows overwrite on same logical blocks.

Zoned Namespace SSD(ZNS SSD)

User applications take over responsibility for

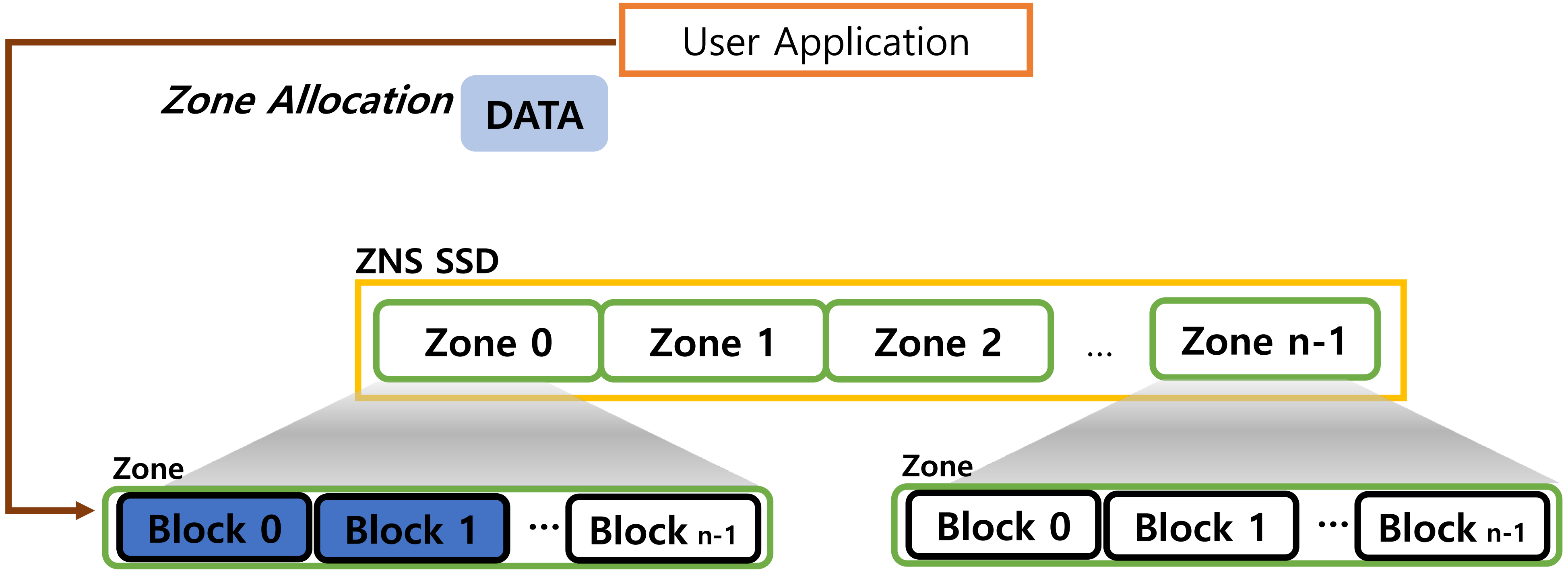
- **Free space reclamation (zone cleaning)**
- **Data placement (zone allocation)**



Zoned Namespace SSD(ZNS SSD)

User applications take over responsibility for

- **Free space reclamation (zone cleaning)**
- **Data placement (zone allocation)**



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

LSM-KV is suitable for ZNS SSD.

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

- ❑ Application

RocksDB LEVELDB

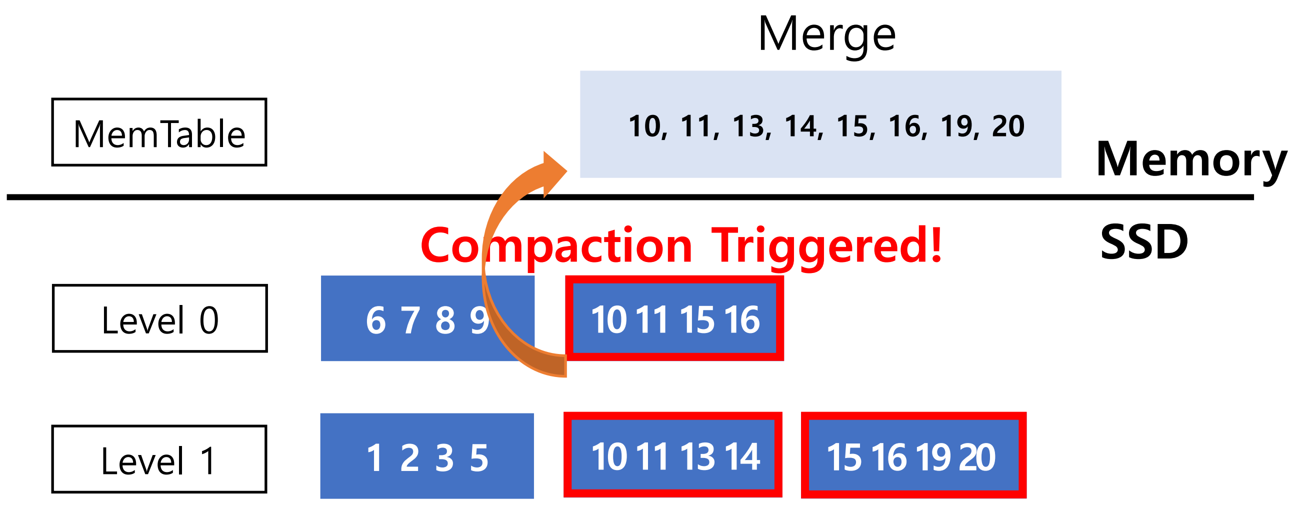
mongoDB® cassandra

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

Data Update in LSM-tree

Compaction

LSM-tree updates data file(SSTable) via compaction.



🟦 : SSTable

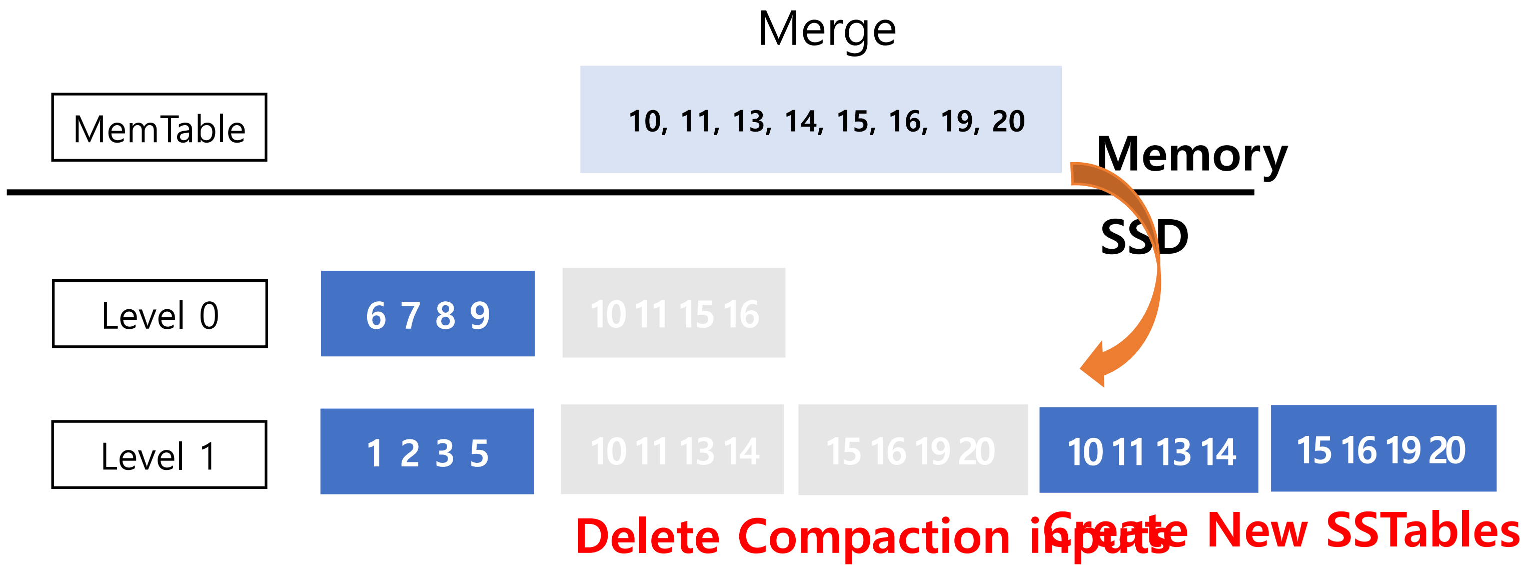
In this Example :
- SSTable size limit : 4 keys
- Level 0 limit : 2 SSTs
- Level 1 limit : 4 SSTs

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

Data Update in LSM-tree

Compaction

LSM-tree updates data file(SSTable) via compaction.



🟦 : SSTable

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

ZenFS[1] [ATC'21]

❑ User-level file system for LSM-KV

❑ Backend module for RocksDB

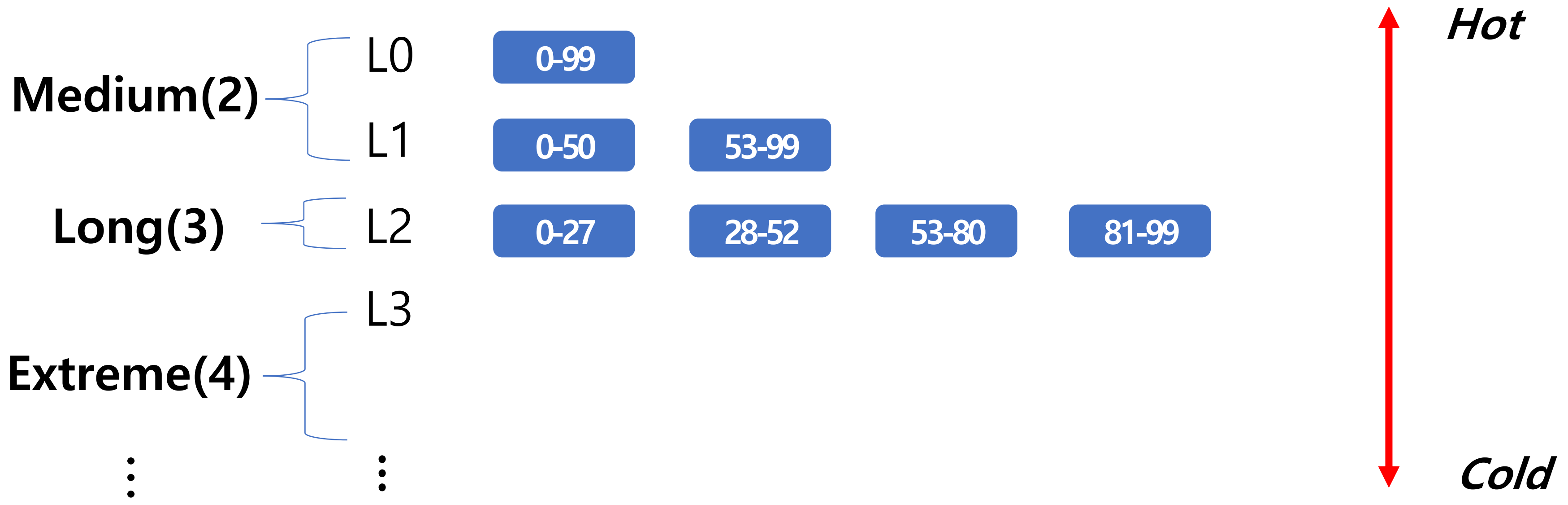
❑ Responsible for

- Zone Allocation of data in LSM-KV
- Zone Cleaning for free space reclamation

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

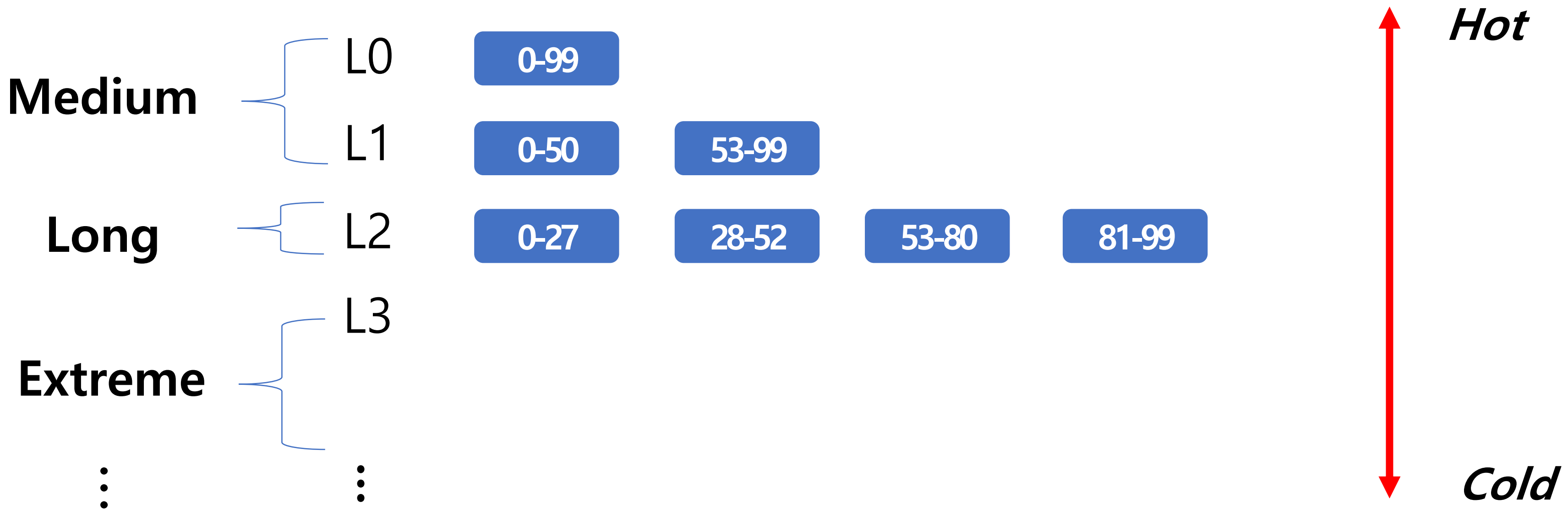
LifeTime based Zone Allocation in ZenFS

❑ **LifeTime :** Reflect data hotness according to level in LSM-tree



LifeTime based Zone Allocation in ZenFS

LifeTime - indicator of **when an SSTable is invalidated** in the device



서강대학교 SOGANG UNIVERSITY

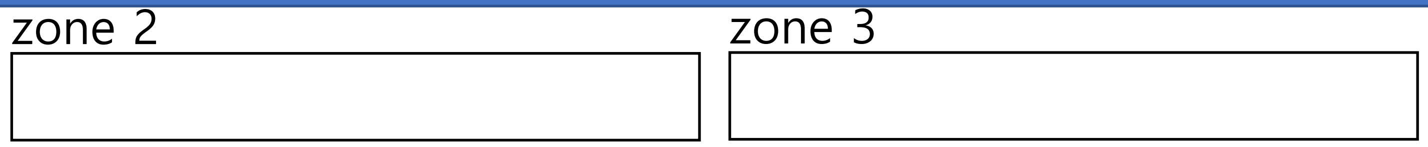
LifeTime based Zone Allocation in ZenFS

- **Zone Allocation using LifeTime**

  - ZenFS allocates SSTables with same LifeTime into the same zone.

ZenFS fails to precisely predict which

<u>SSTables will be deleted together</u>

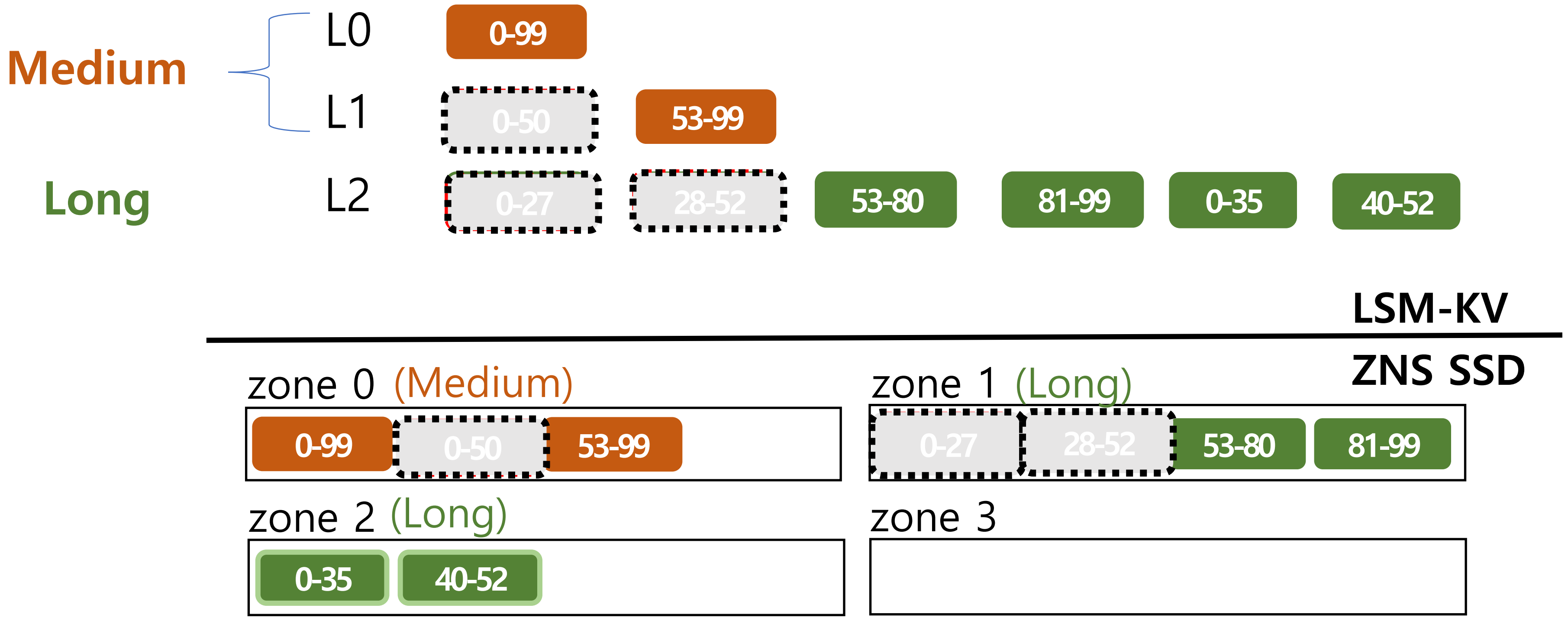


LifeTime based Zone Allocation in ZenFS

ZenFS fails to precisely predict which SSTables will be deleted together.

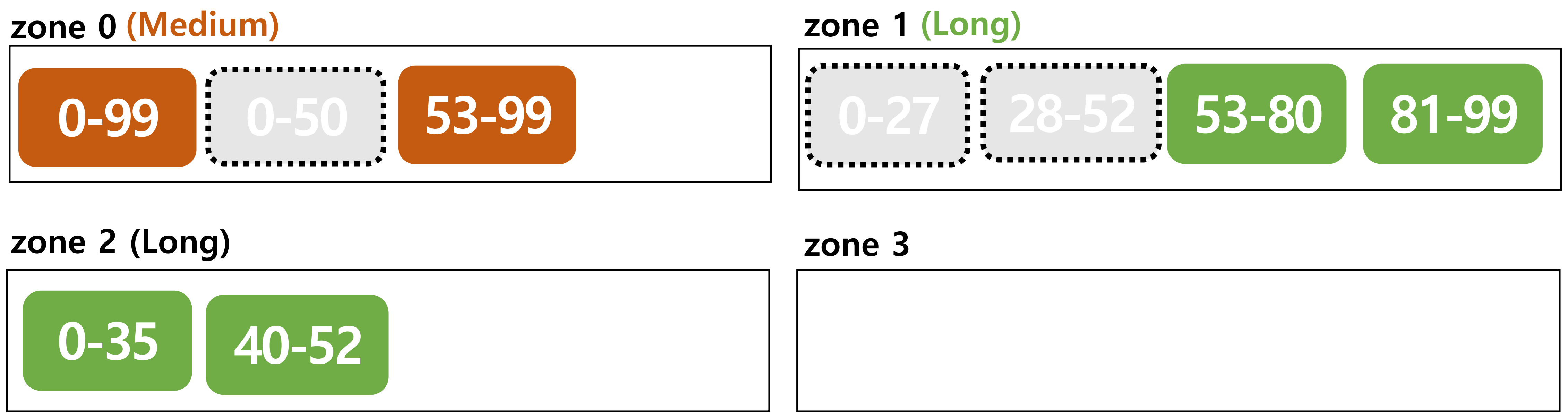
① Trigger Compaction in L1 ② Select Compaction inputs / Merge them

③ Create / Allocate new SSTables ④ Delete Compaction inputs



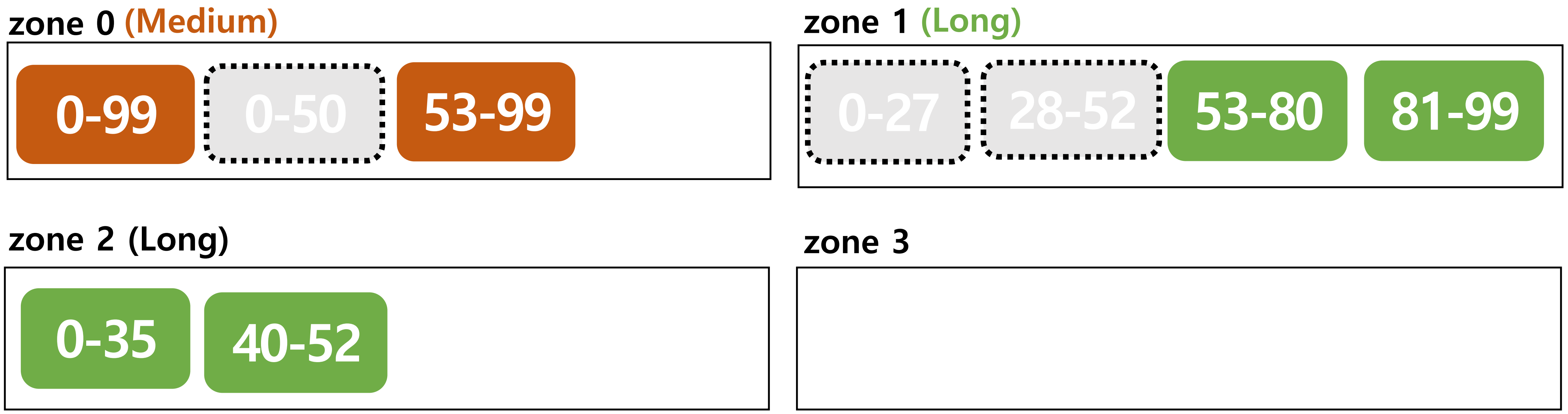
LifeTime based Zone Allocation in ZenFS

ZenFS fails to precisely predict which SSTables will be deleted together.



LifeTime based Zone Allocation in ZenFS

ZenFS fails to precisely predict which SSTables will be deleted together.



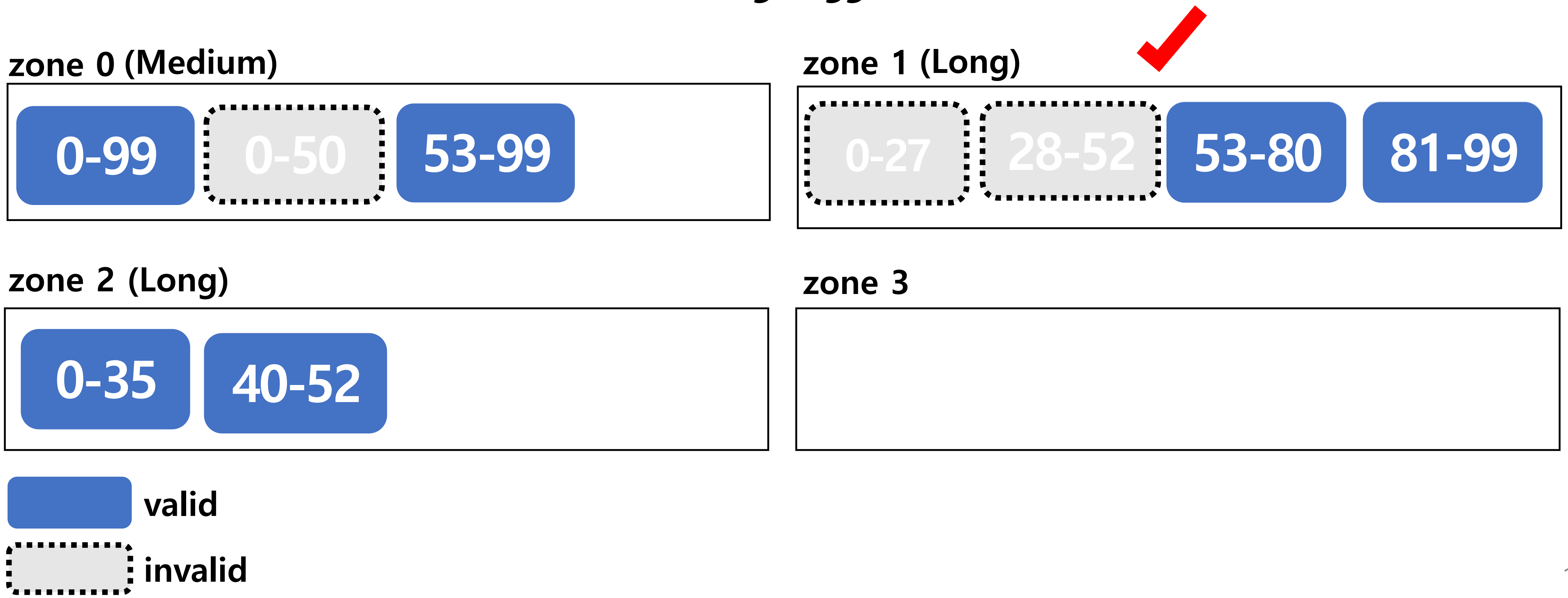
서강대학교 SOGANG UNIVERSITY

Overhead of ZenFS Allocation

Write Amplification

Prediction failure of deletion time leads to valid data copying during zone cleaning.

Zone Cleaning Trigger

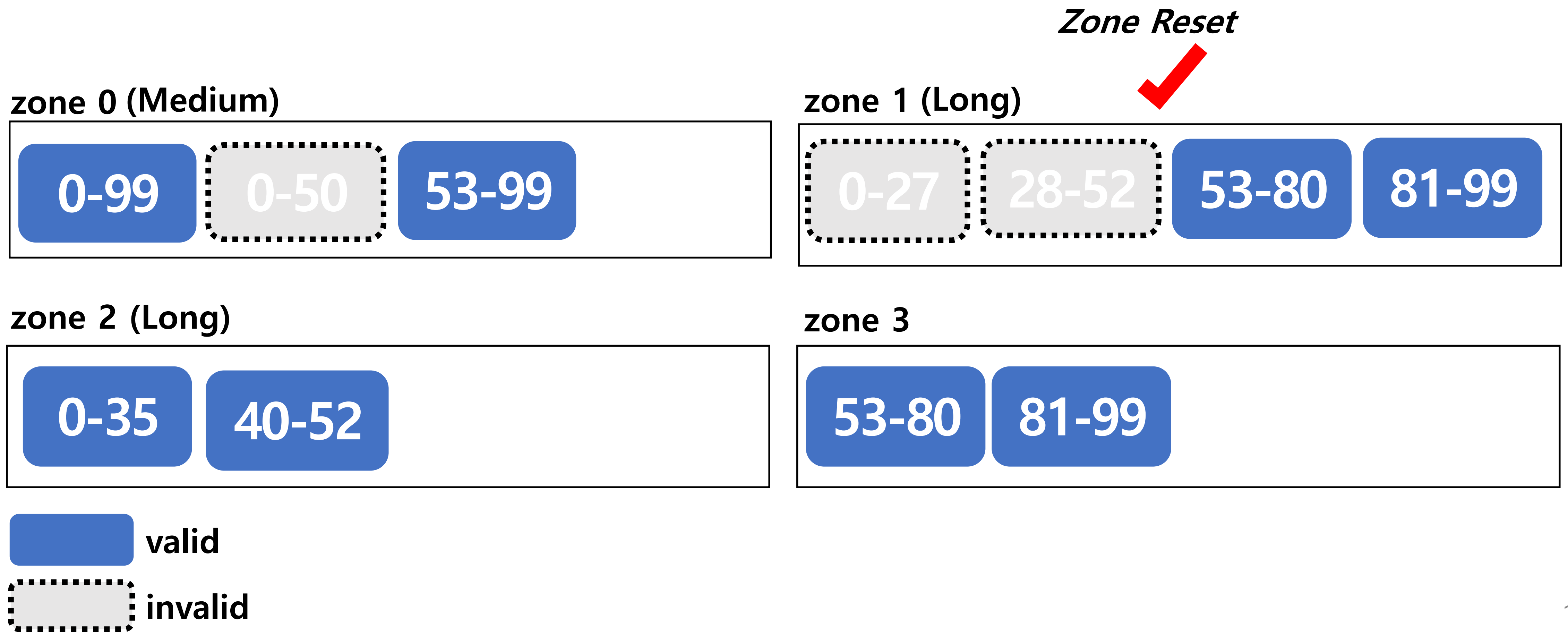


서강대학교 SOGANG UNIVERSITY

Overhead of ZenFS Allocation

Write Amplification

Prediction failure of deletion time leads to valid data copying during zone cleaning.



Compaction-Aware Zone Allocation (CAZA)

Main Idea

- ❑ We observed that SSTables to be compacted together are invalidated at the same time.
- ❑ We allocate SSTables to be compacted together into the same zone.

Challenge : How can we predict which SSTables will be compacted together ?

Compaction-Aware Zone Allocation (CAZA)

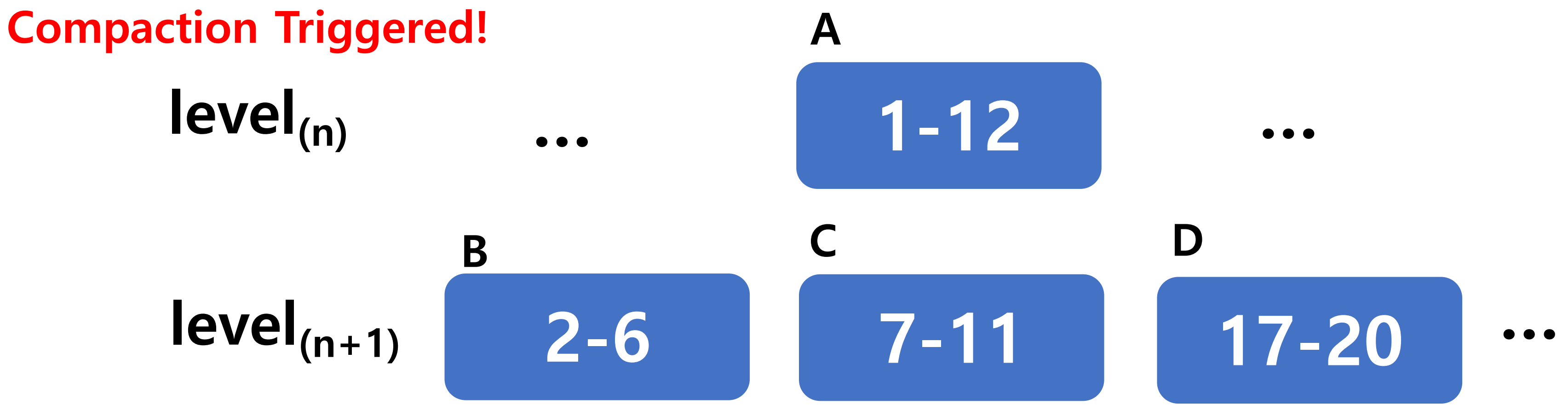
Conditions for SSTables to be compacted together

Condition 1: SSTables that are located in adjacent levels

Condition 2 : SSTables that have overlapping key ranges

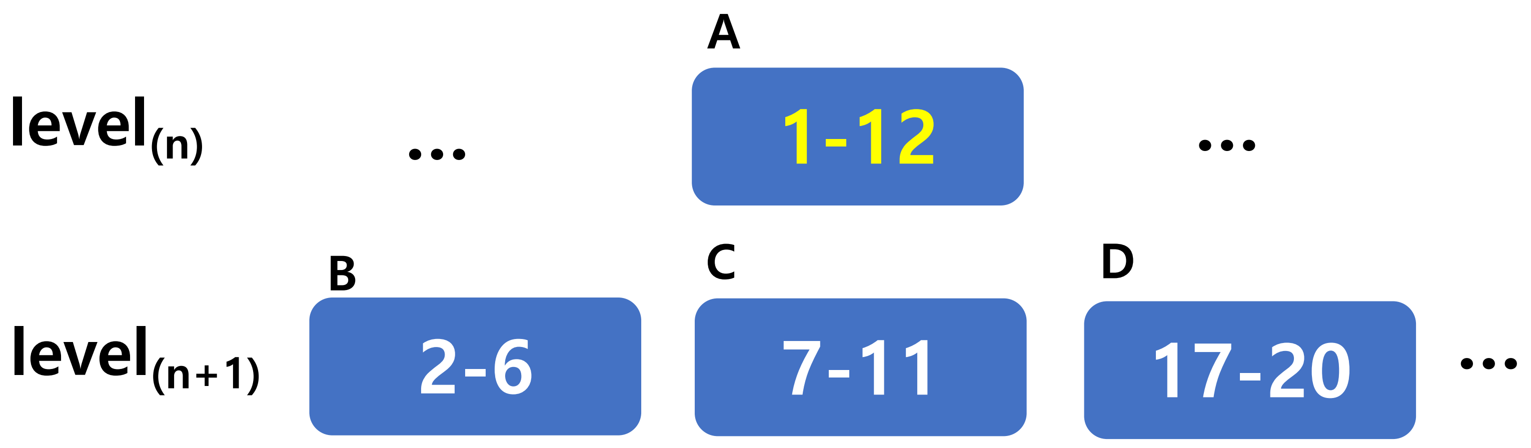
Compaction-Aware Zone Allocation (CAZA)

Compaction Input Picking in LSM-tree



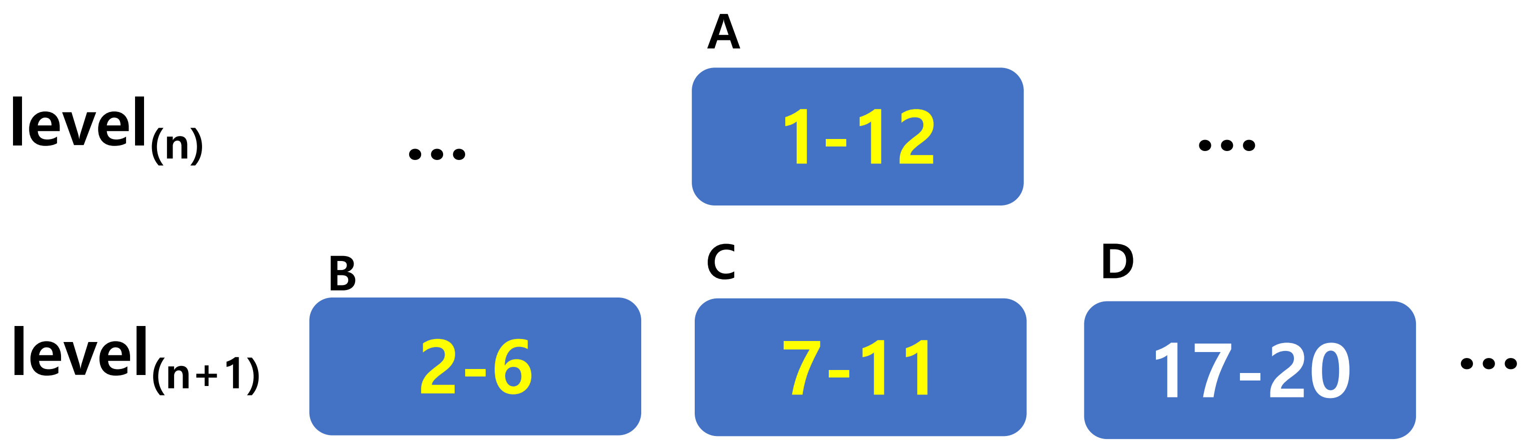
Compaction-Aware Zone Allocation (CAZA)

Compaction Input Picking in LSM-tree



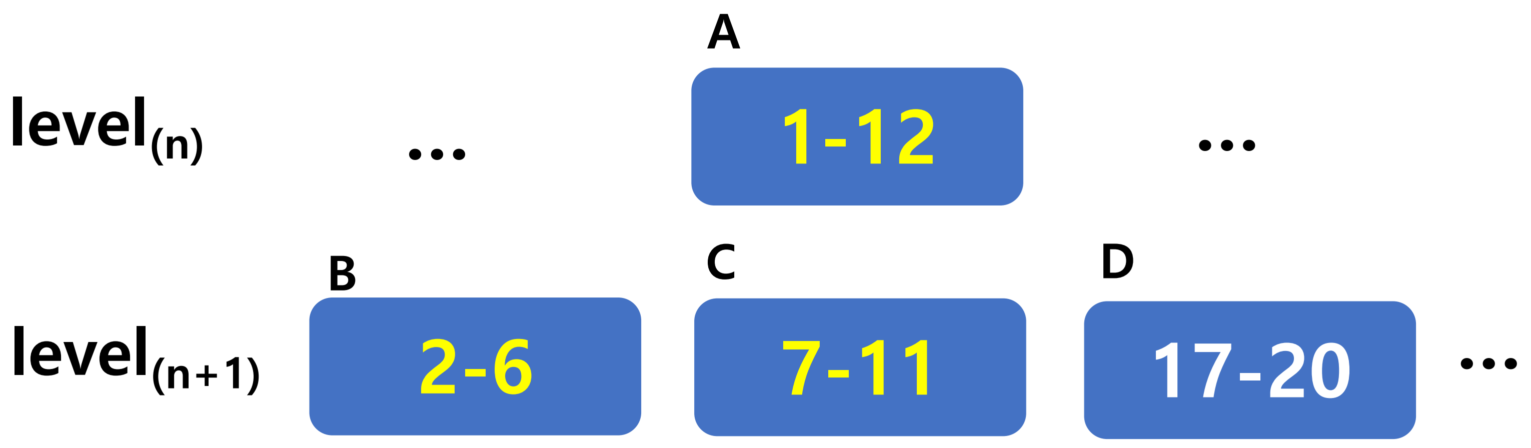
Compaction-Aware Zone Allocation (CAZA)

Compaction Input Picking in LSM-tree

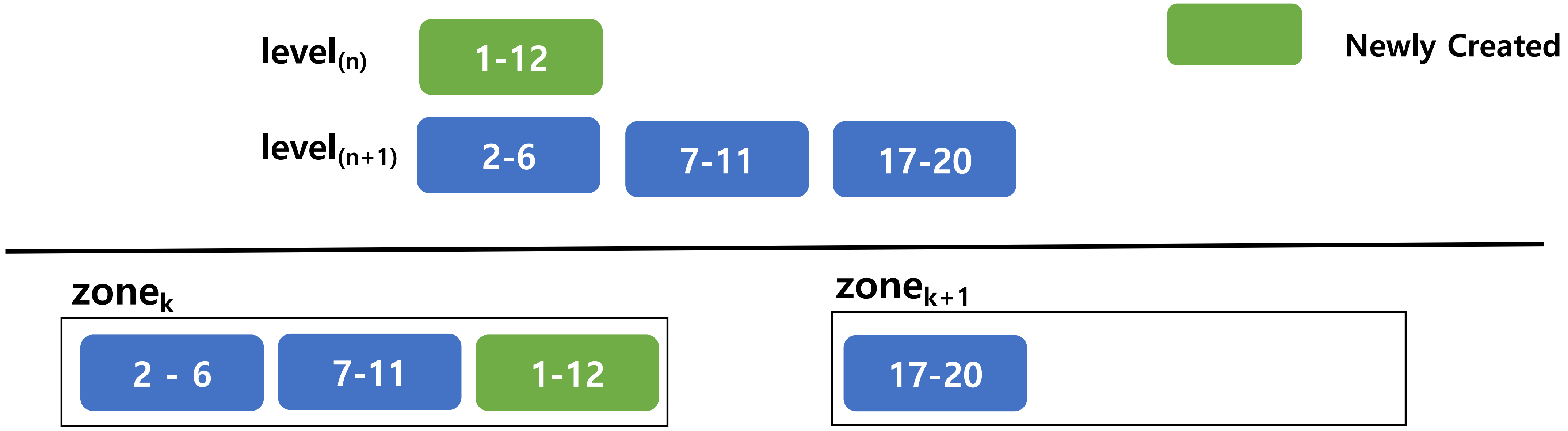


Compaction-Aware Zone Allocation (CAZA)

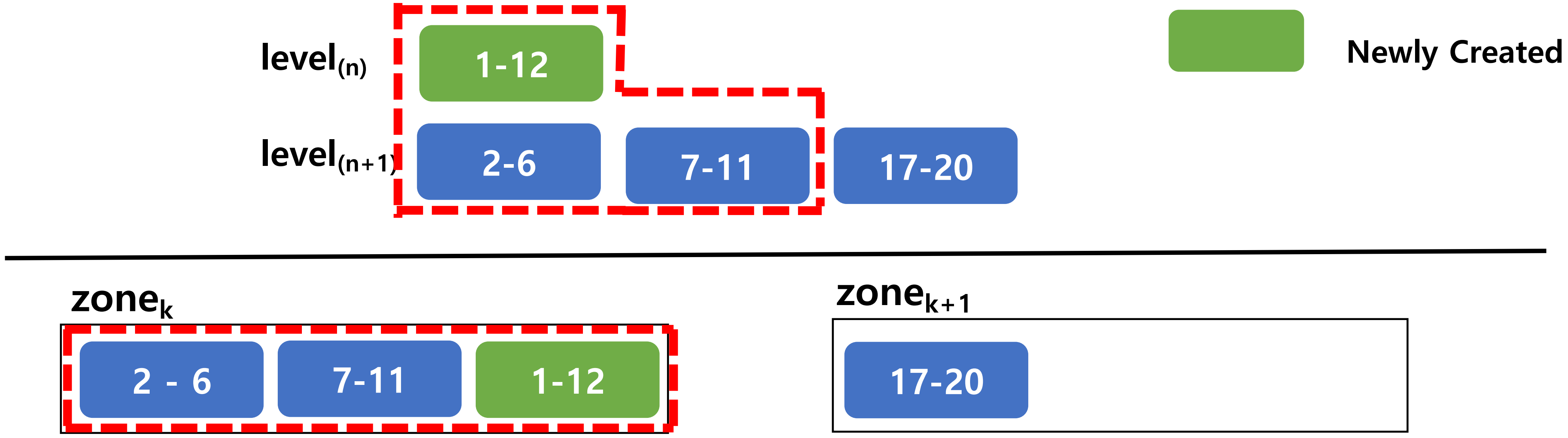
Compaction Input Picking in LSM-tree



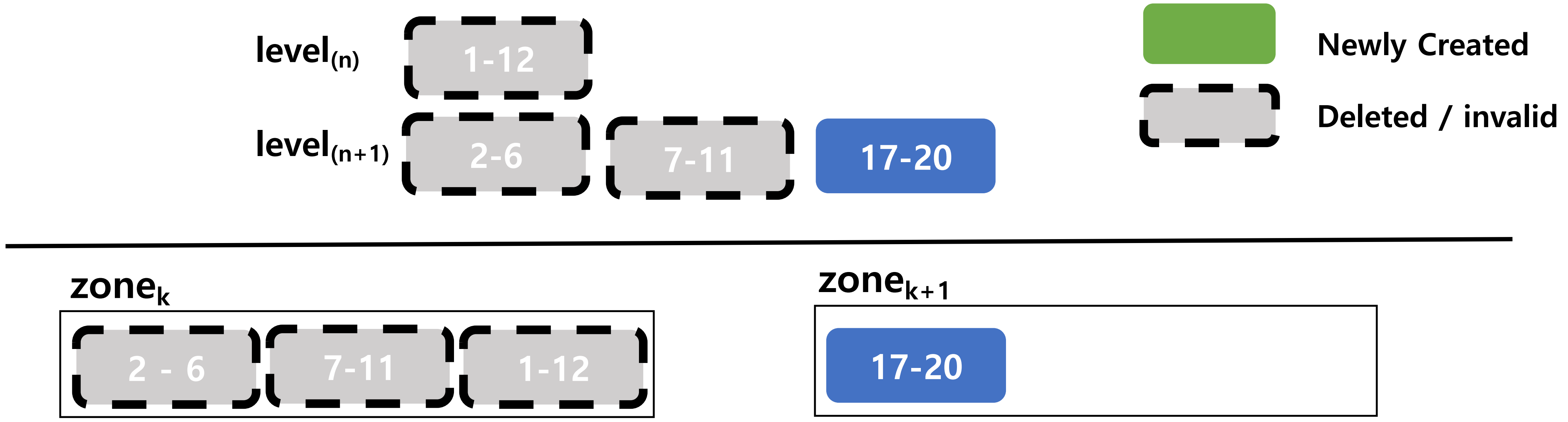
Compaction-Aware Zone Allocation (CAZA)



Compaction-Aware Zone Allocation (CAZA)



Compaction-Aware Zone Allocation (CAZA)

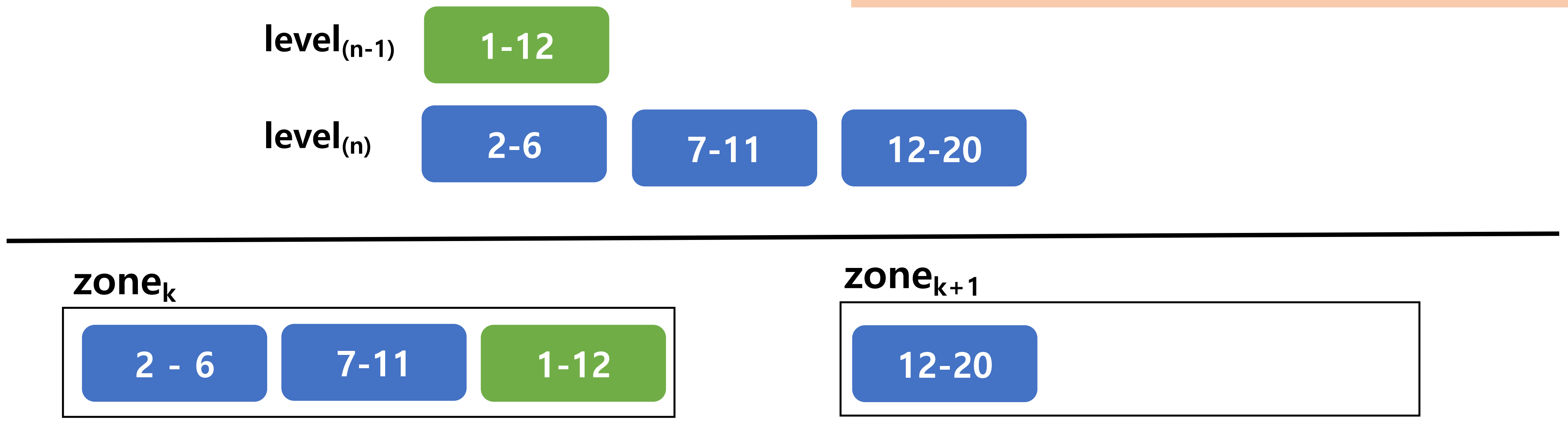


Compaction-Aware Zone Allocation (CAZA)

❑ Compaction-Aware Zone Allocation Rules

Case 1) SSTables with overlapping key ranges spread in several zones.

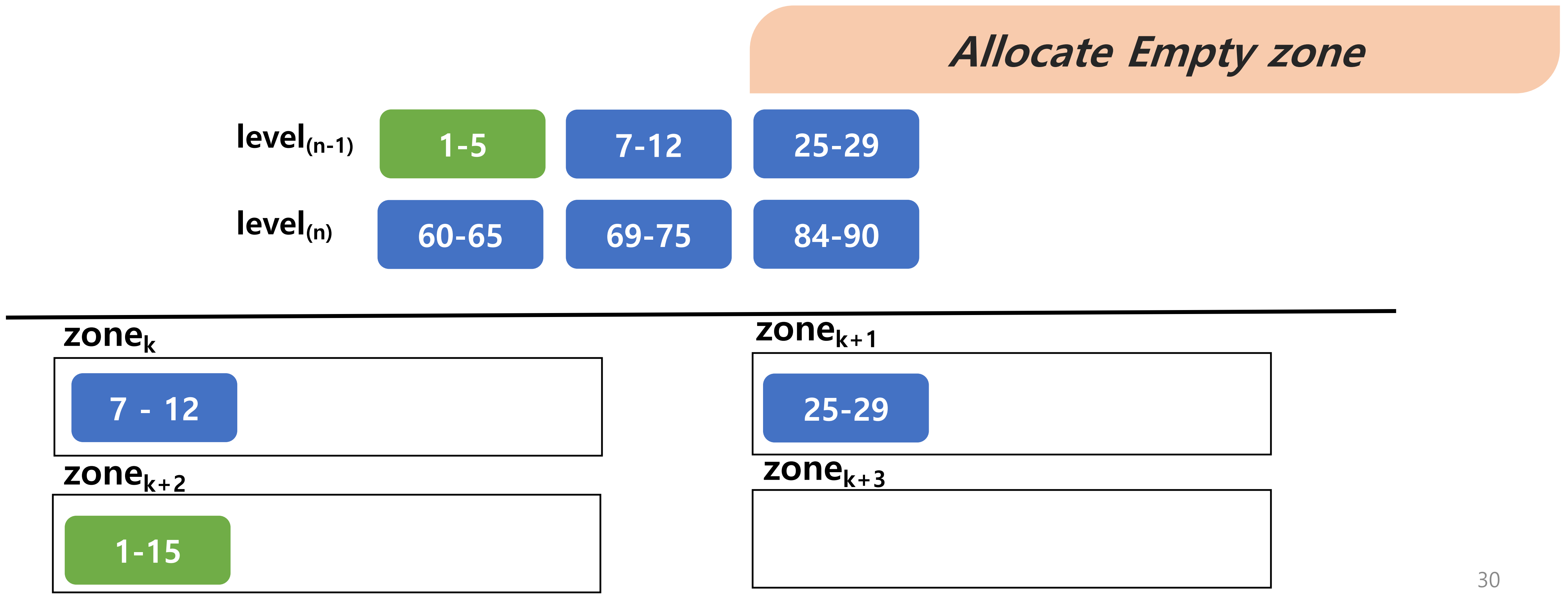
Allocate the zone with more SSTables



Compaction-Aware Zone Allocation (CAZA)

❑ Compaction-Aware Zone Allocation Rules

Case 2) No matching SSTables that meet the compaction condition

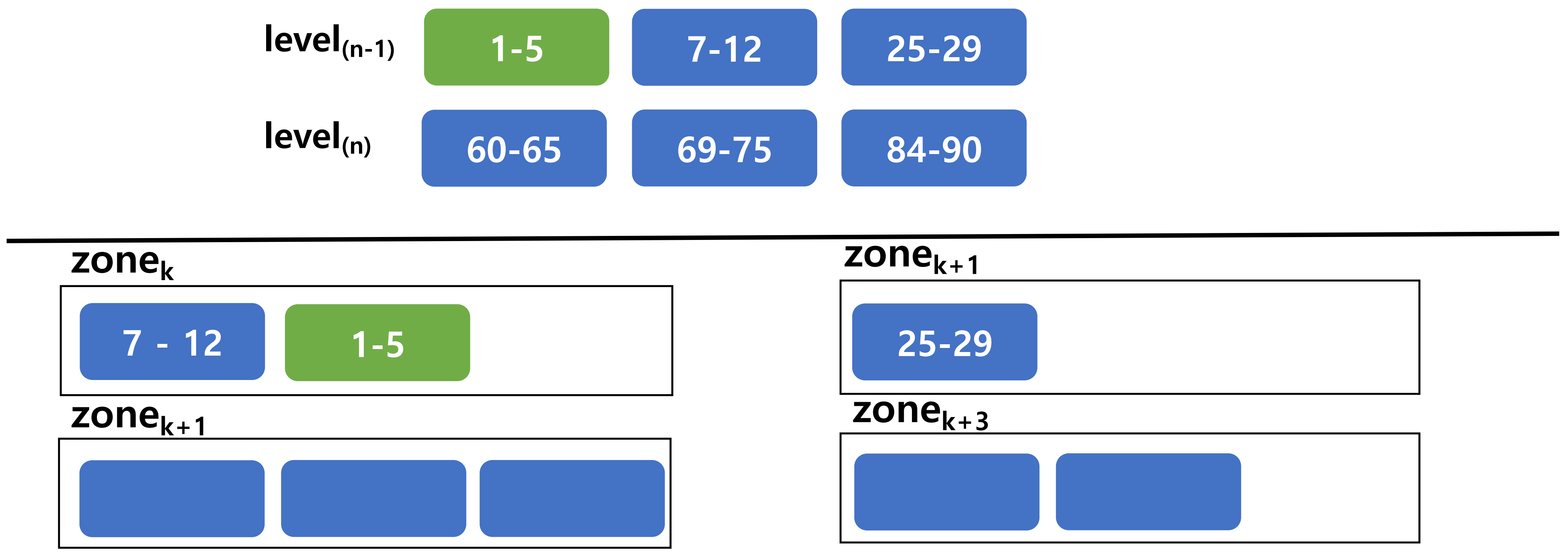


Compaction-Aware Zone Allocation (CAZA)

❑ Compaction-Aware Zone Allocation Rules

Case 2) No matching SSTables that meet the compaction condition

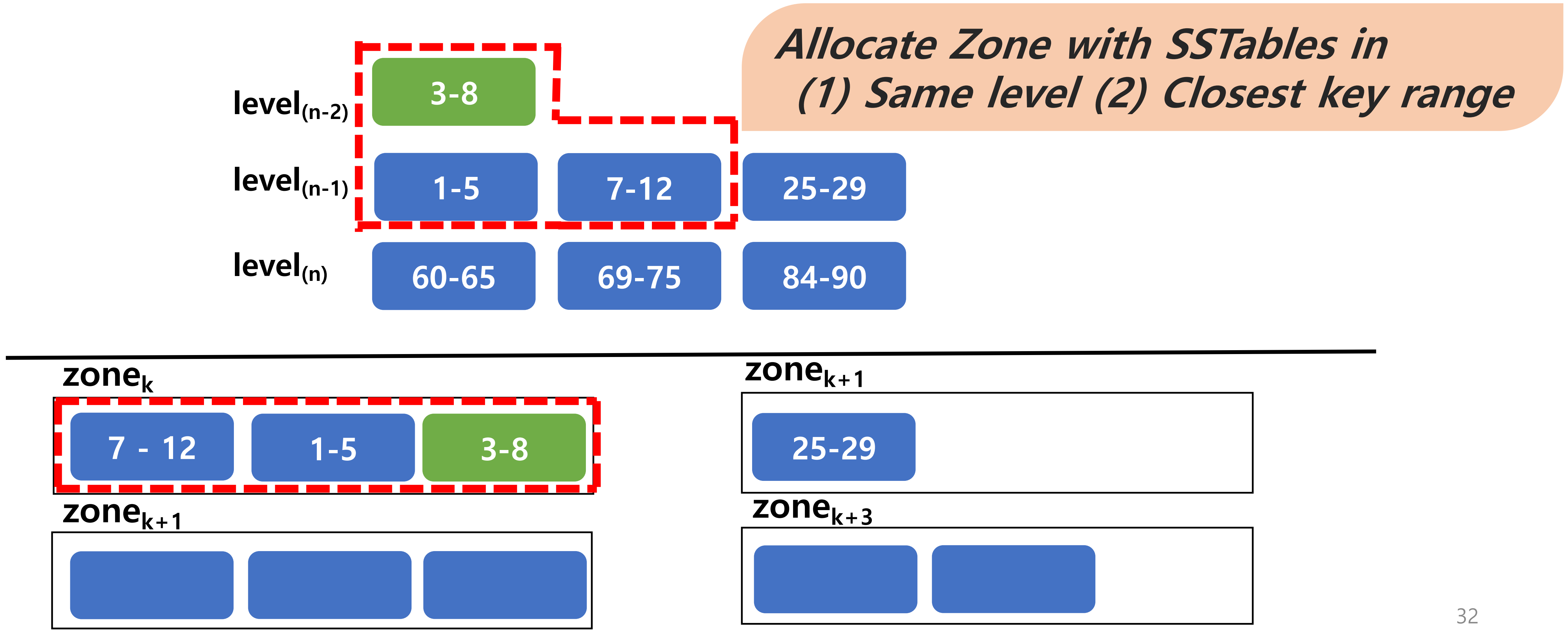
Allocate Zone with SSTables in (1) Same level (2) Closest key range



Compaction-Aware Zone Allocation (CAZA)

- Compaction-Aware Zone Allocation Rules

Case 2) No matching SSTables that meet the compaction condition



서강대학교 SOGANG UNIVERSITY

Compaction-Aware Zone Allocation (CAZA)

❑ Compaction-Aware Zone Allocation Rules

No matching case..

➔ Follow Zone Allocation in ZenFS

Evaluation

❑ Comparison

- **LIZA** : LifeTime-based Zone Allocation in ZenFS
- **CAZA** : Compaction-Aware Zone Allocation

Testbed Specifications	
ZNS SSD	100GB Capacity DRAM Emulation 1GB-sized zone (total 100 zones)
CPU	Intel Xeon E5-2640
LSM-KV	RocksDB v6.13
OS	Linux Kernel 5.10.13

❑ Workload

- RocksDB micro-benchmark
  (*overwrite after fillseq*)
- 40GB KVs, 16B key & 128B Value

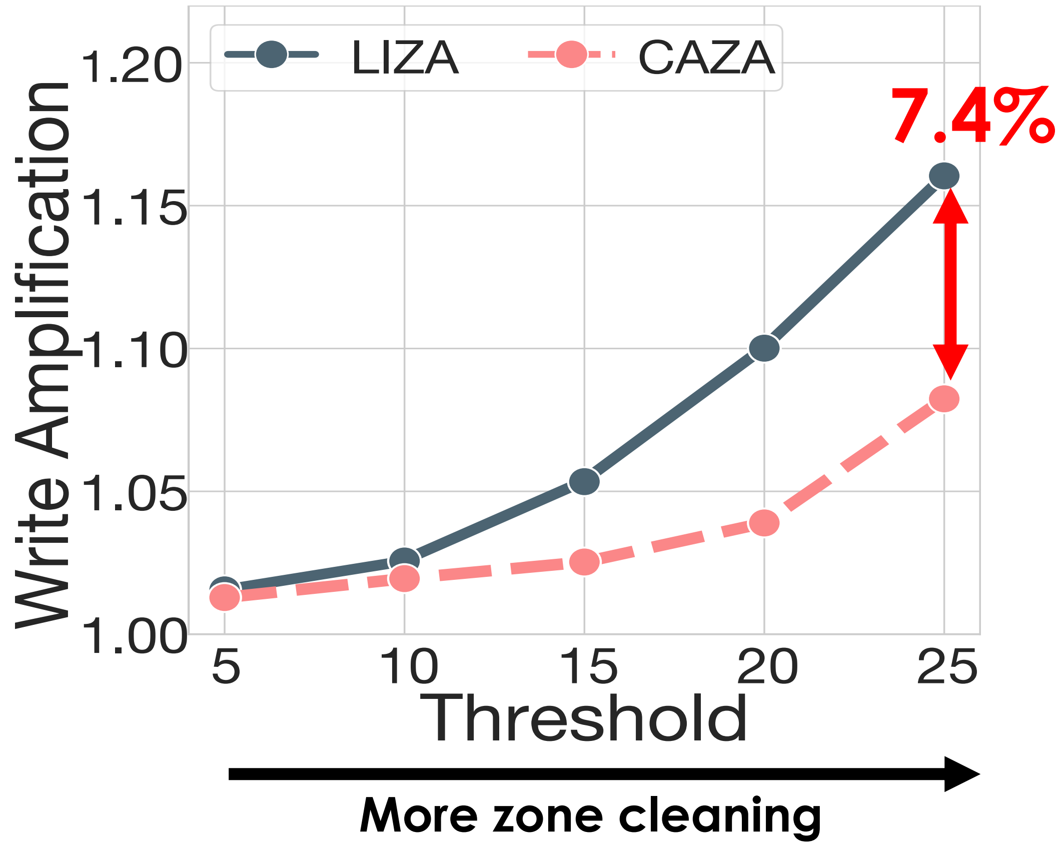
❑ Zone Cleaning

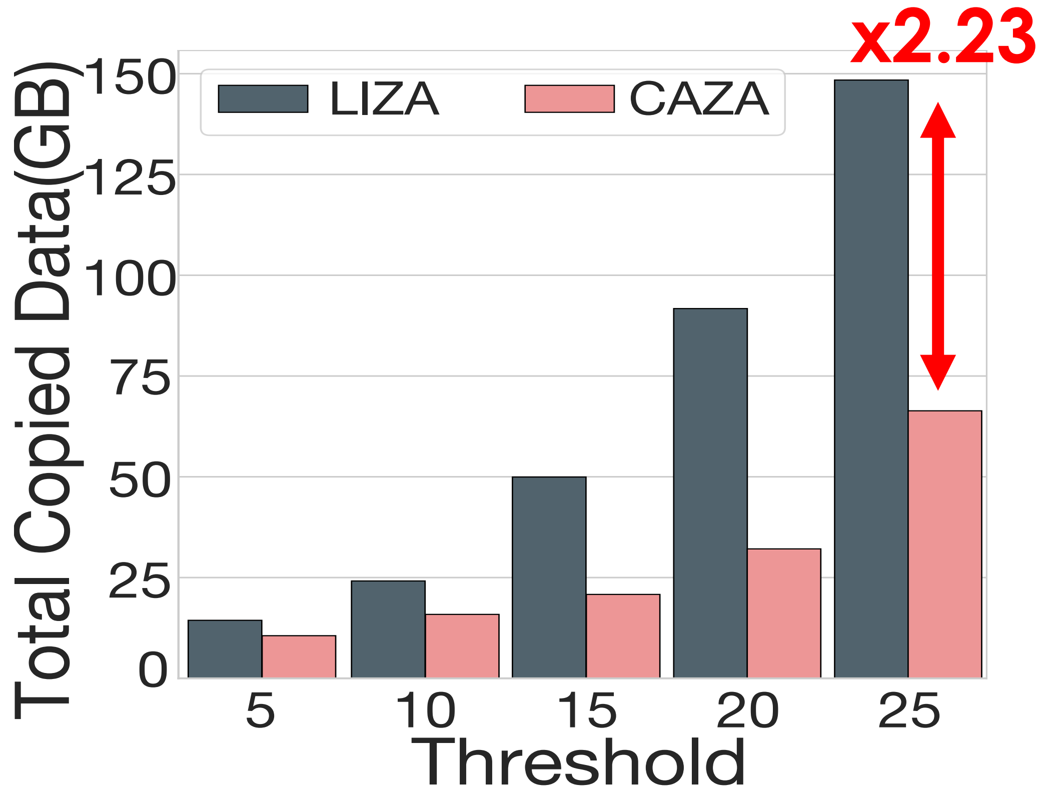
- **Greedy Zone Cleaning** : Select zones with most invalid data for reset
- Varying threshold(5%, 10%, 15%, 20%, 25%) where zone cleaning must reclaim for free space

Evaluation

Write Amplification(WA)

$$WA = \frac{Written\ by\ LSM{-}KV + \ Copied\ by\ Zone\ cleaing}{Written\ by\ LSM{-}KV}$$





서강대학교 SOGANG UNIVERSITY

Conclusion

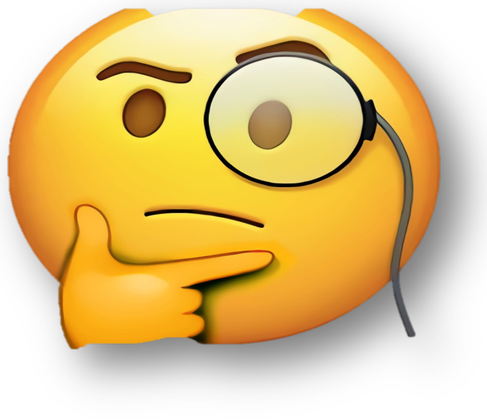
❑ Compaction-Aware Zone Allocation(CAZA)

- We proposed novel data placement algorithm for LSM-KV on ZNS SSDs.
- CAZA precisely estimates lifetime of SSTables.
- CAZA offers 7.4% lower write amplification and 2x lower data copying during zone cleaning than LIZA.

Q&A

Thank you!

Any Questions?



Hee-Rock : heeock@sogang.ac.kr / heerock1.lee@gmail.com