Lifetime-Leveling LSM-tree Compaction for ZNS SSD

Jeeyoon Jung, Dongkun Shin

Sungkunkwan University, Korea

HotStorage 2022

The 14th ACM Workshop on Hot Topics in Storage and File Systems

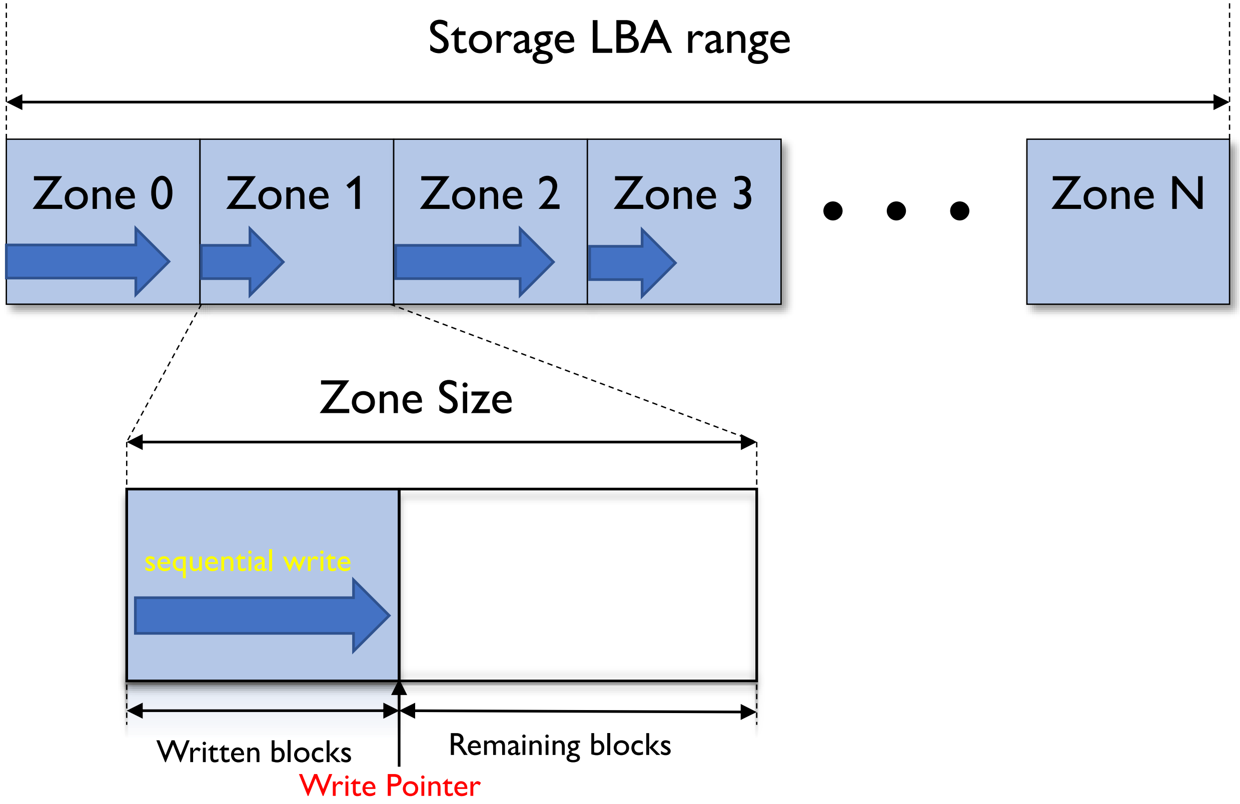
June 27-28

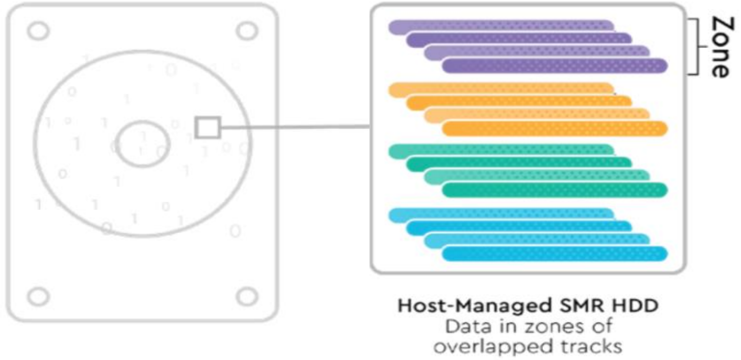
Virtual

SUNG KYUN KWAN UNIVERSITY(SKKU)

Zoned Namespace (ZNS)

- The logical address space is divided into fixed-size zones.
- Each zone must be written sequentially and reset explicitly for reuse
- No SSD-side garbage collection

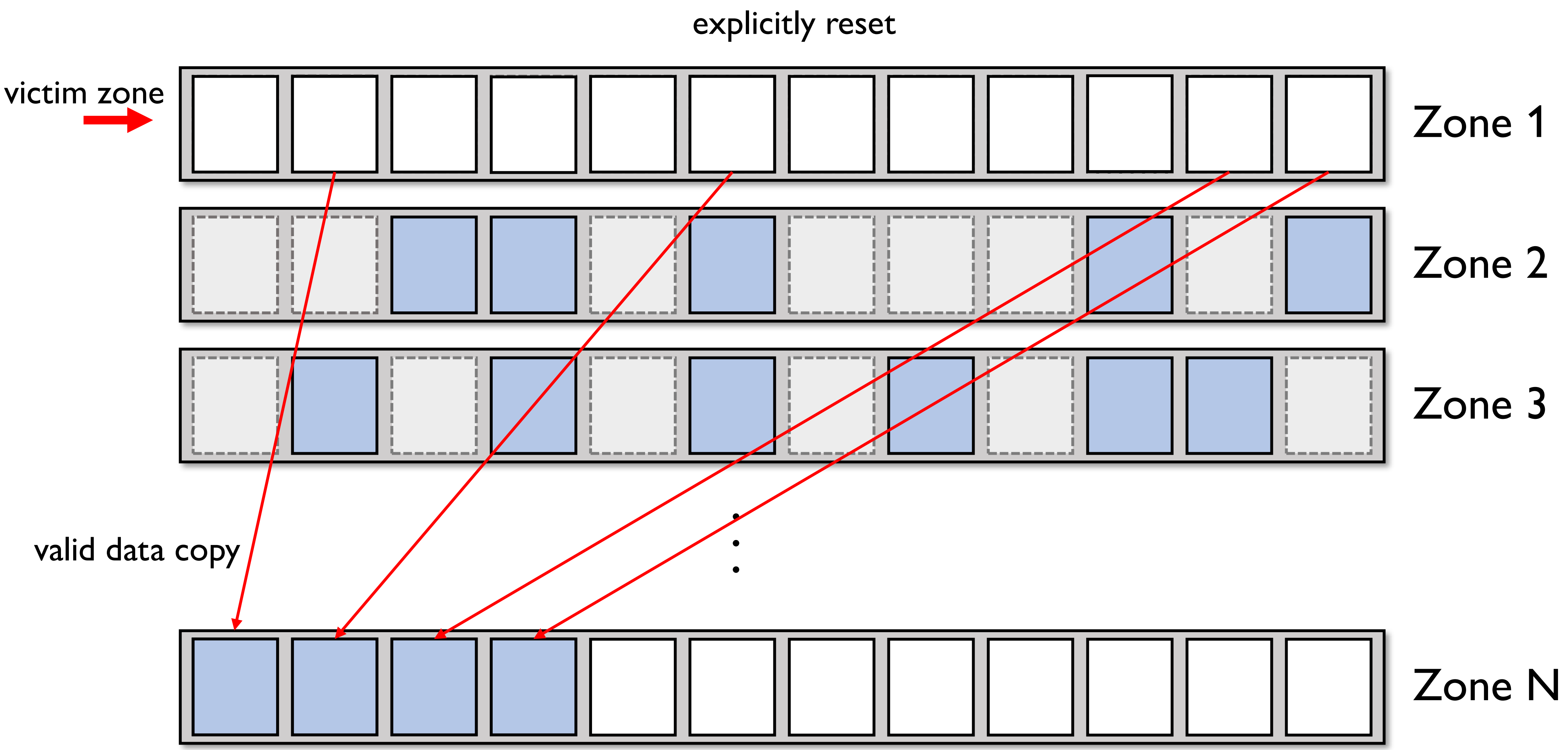




HM-SMR HDD

ZNS SSD

Host-managed GC on ZNS



HotStorage 2022
The 14th ACM Workshop on Hot Topics in Storage and File Systems
June 27-28 Virtual

LSM-tree SST at ZNS

Large SST

Zone0	Zone1	Zone2	Zone3

- Large Compaction Cost
- High memory pressure

Large Zone

SST0	SST1	SST2	SST3

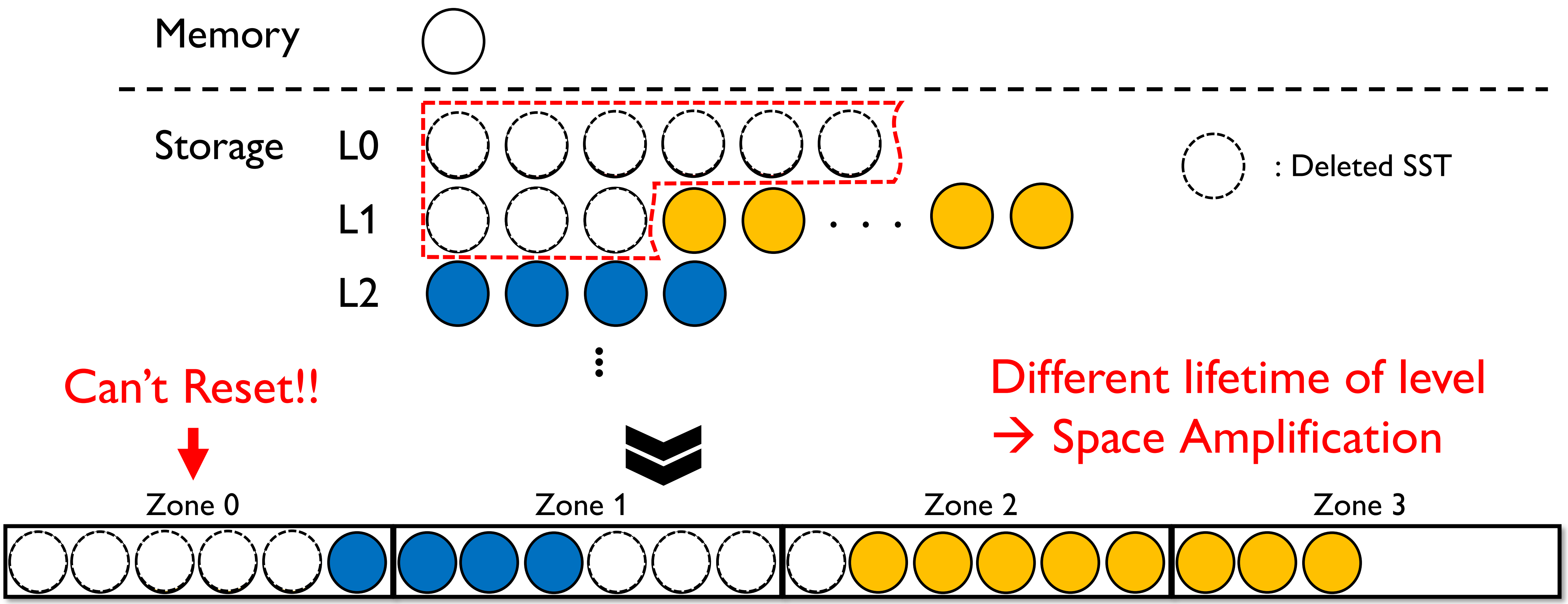
- Small Compaction Cost
- Low memory Size

Target of this paper

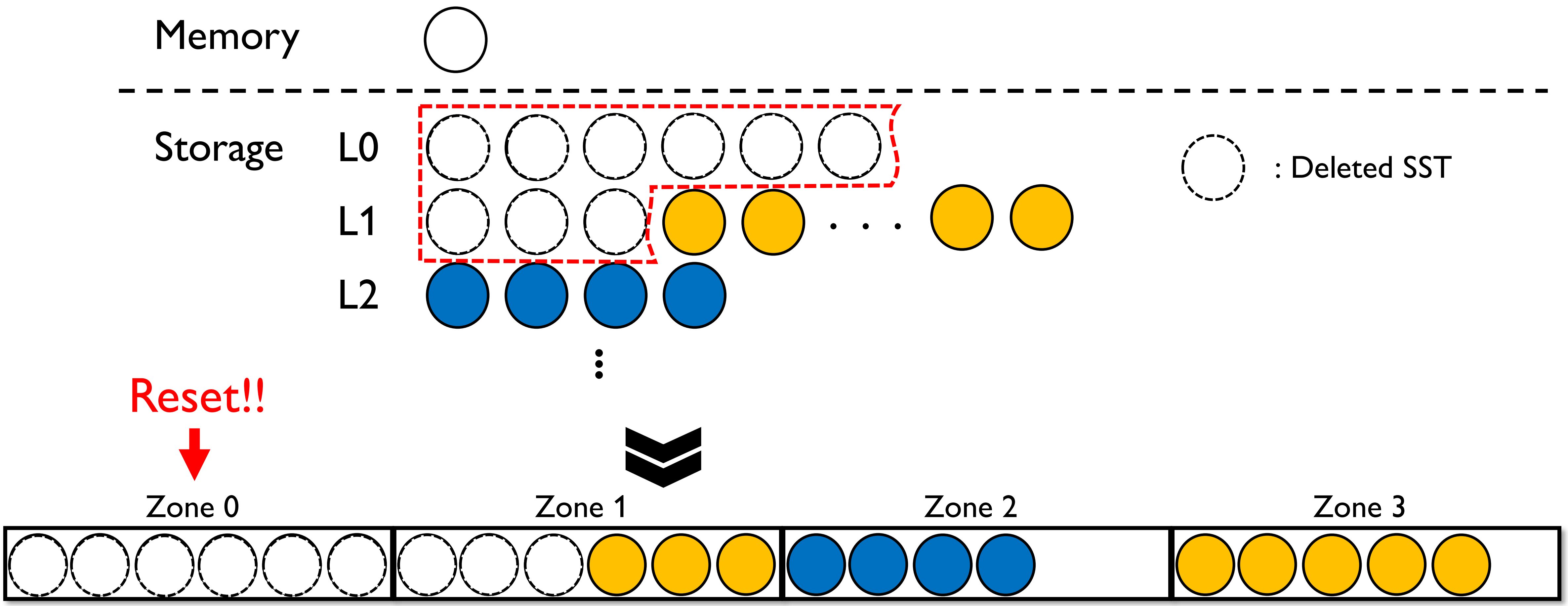
- **Space Amplification**

→ Host-managed GC

Level Separation

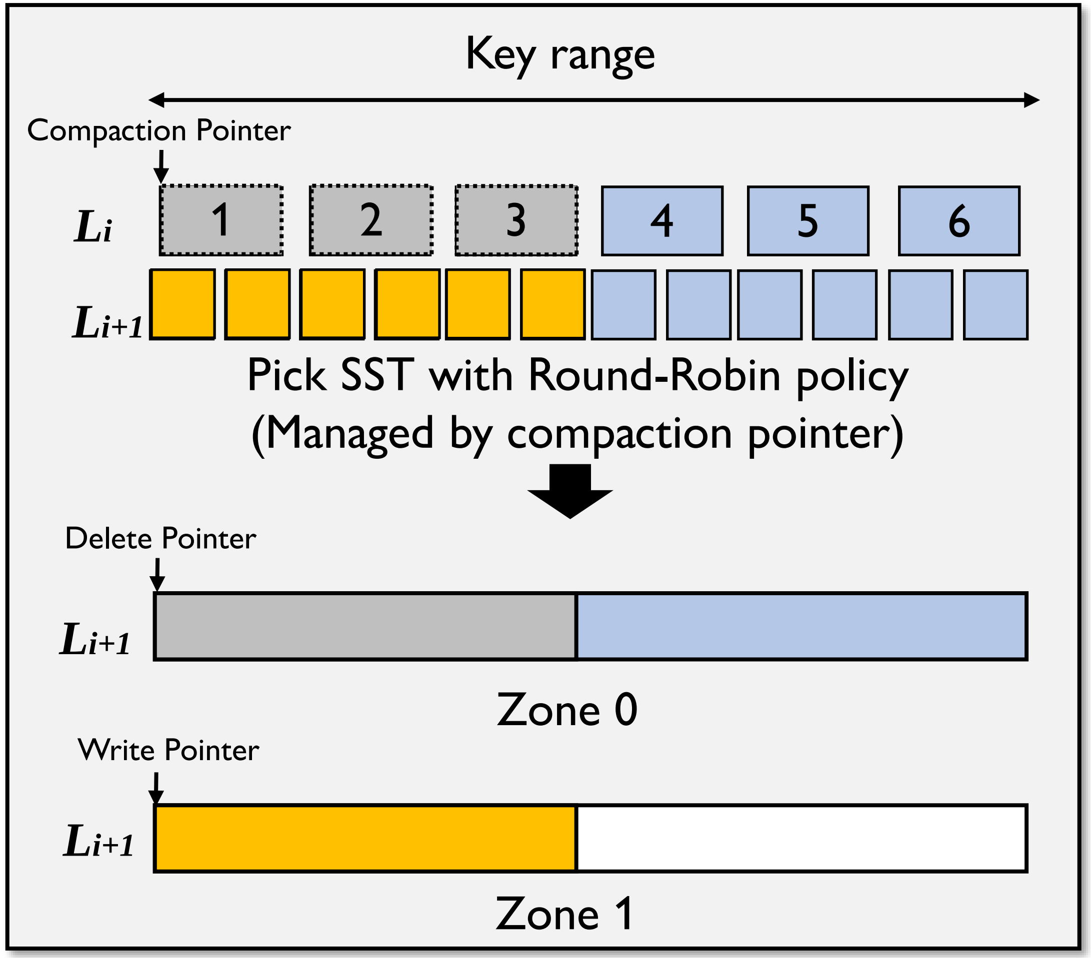


Level Separation (GearDB)

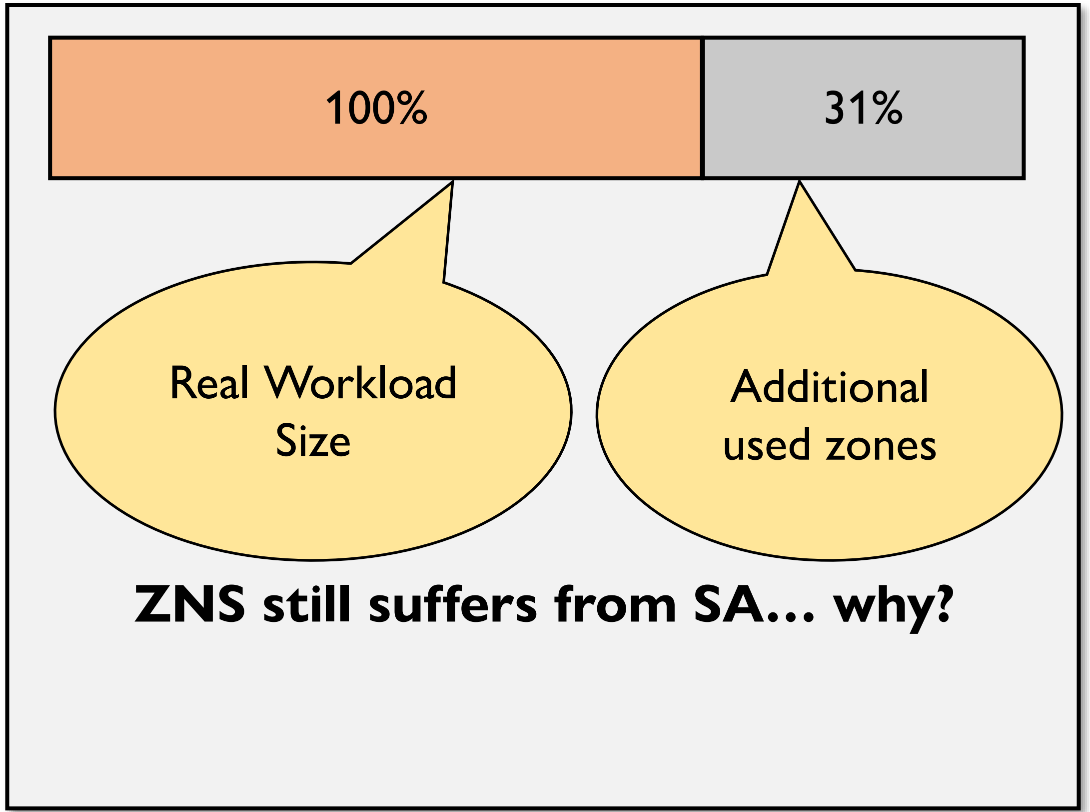


Space Amplification

Current LevelDB Compaction Algorithm

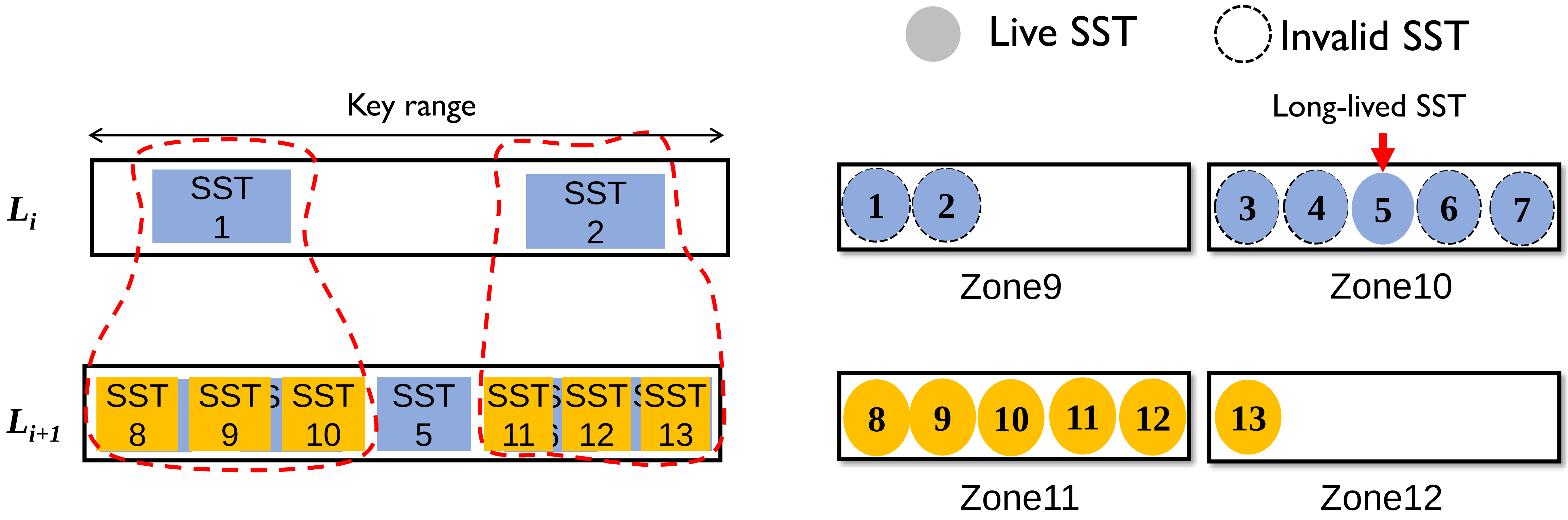


Even with level separation



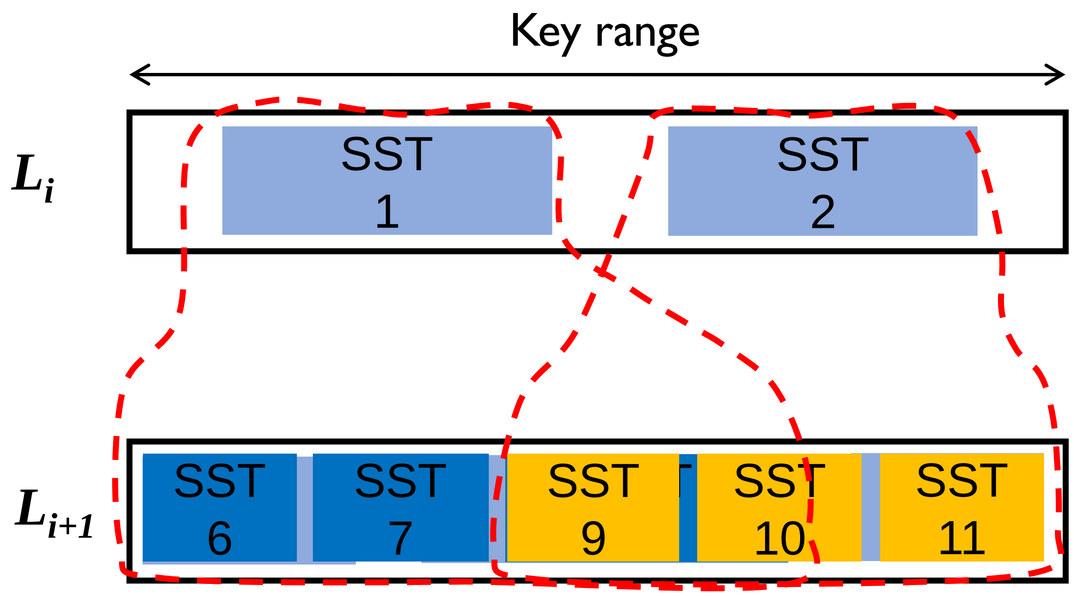
HotStorage 2022
The 14th ACM Workshop on Hot Topics in Storage and File Systems
June 27-28 Virtual

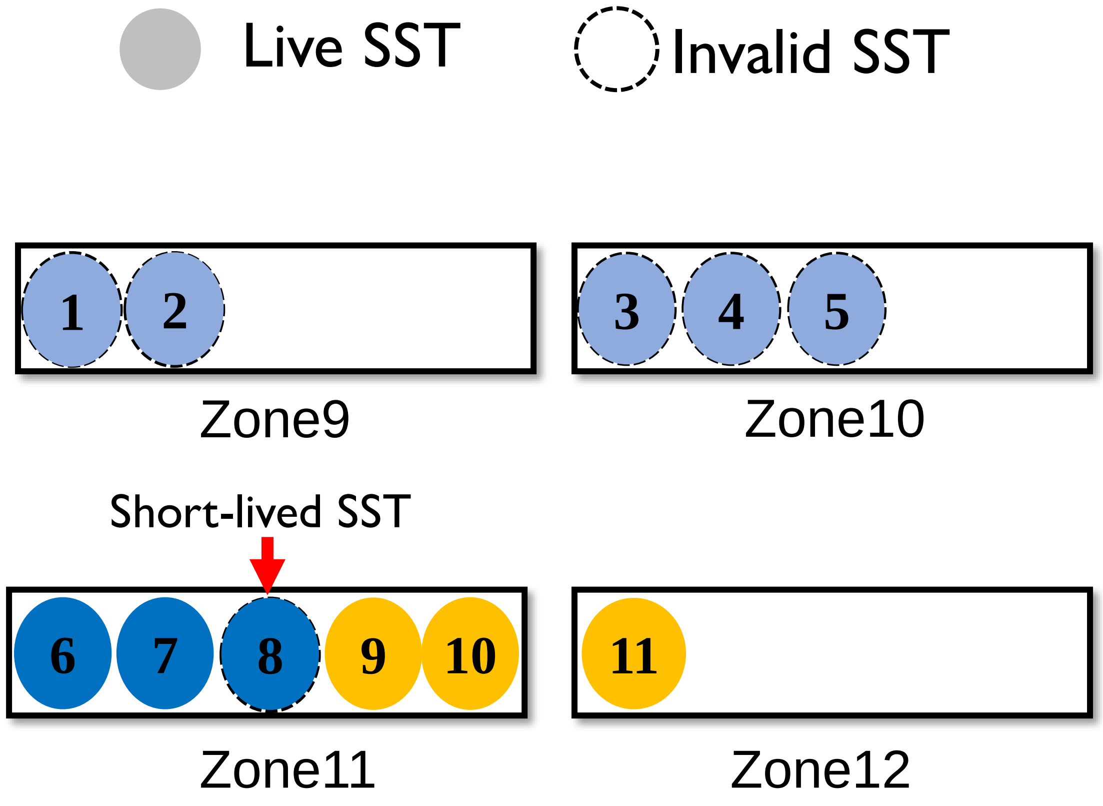
Long-lived SST



Additional 2% zones are allocated by Long-lived SST

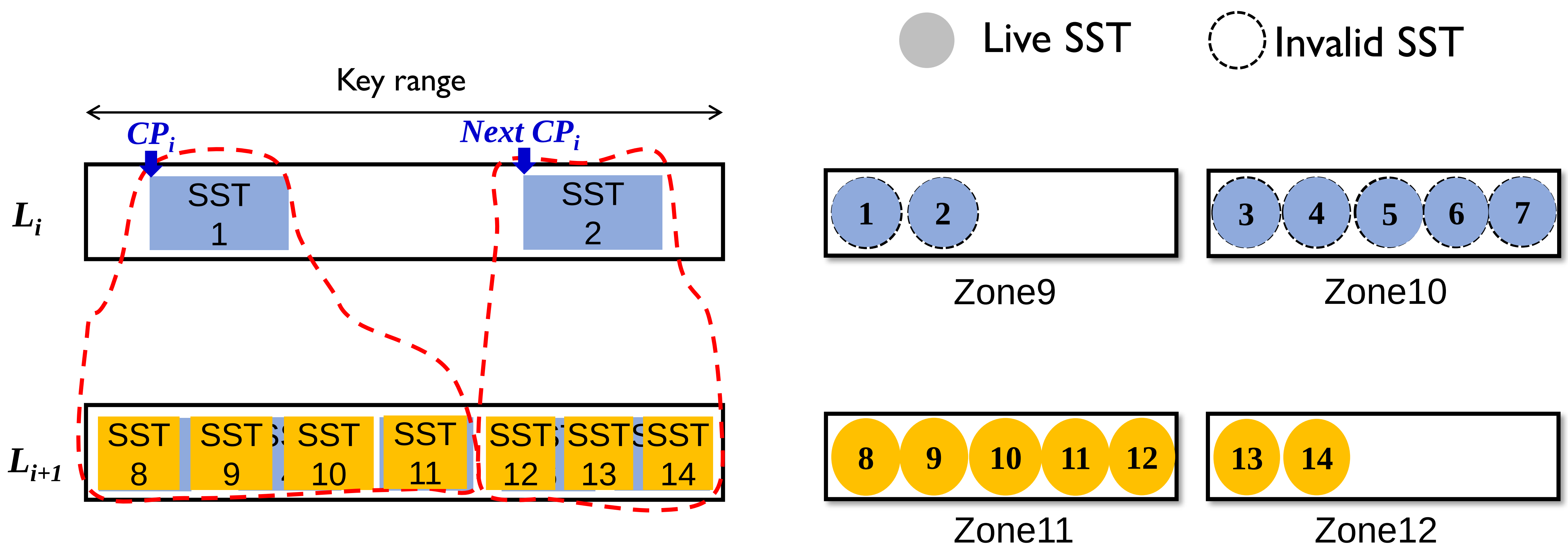
Short-lived SST





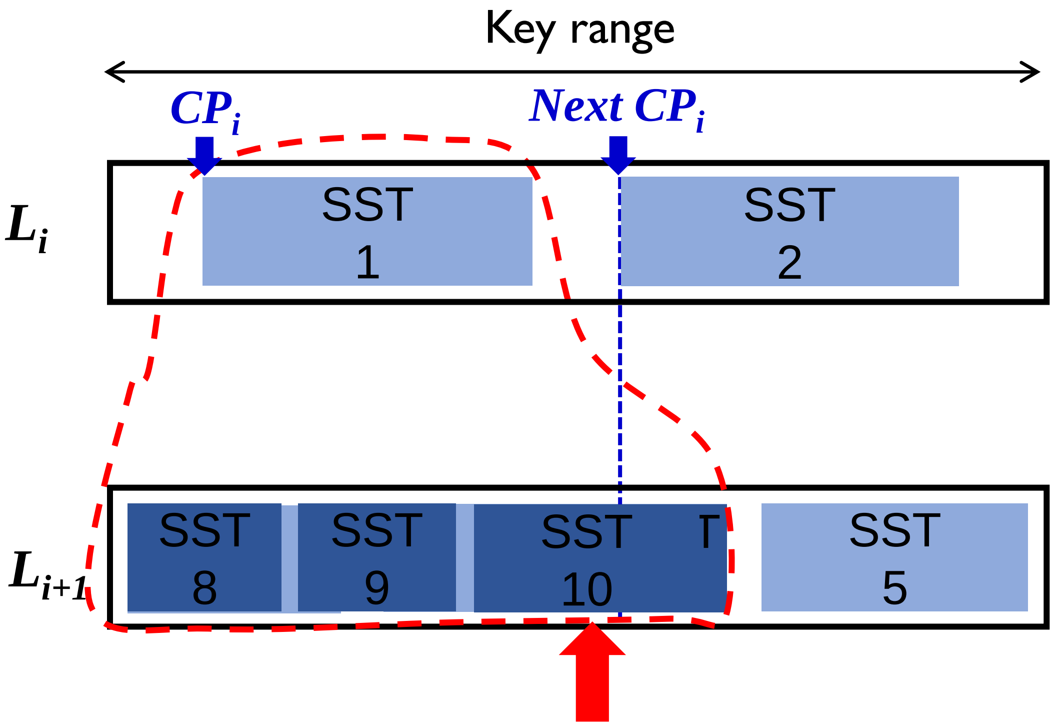
Additional 30% zones are allocated by Short-lived SST

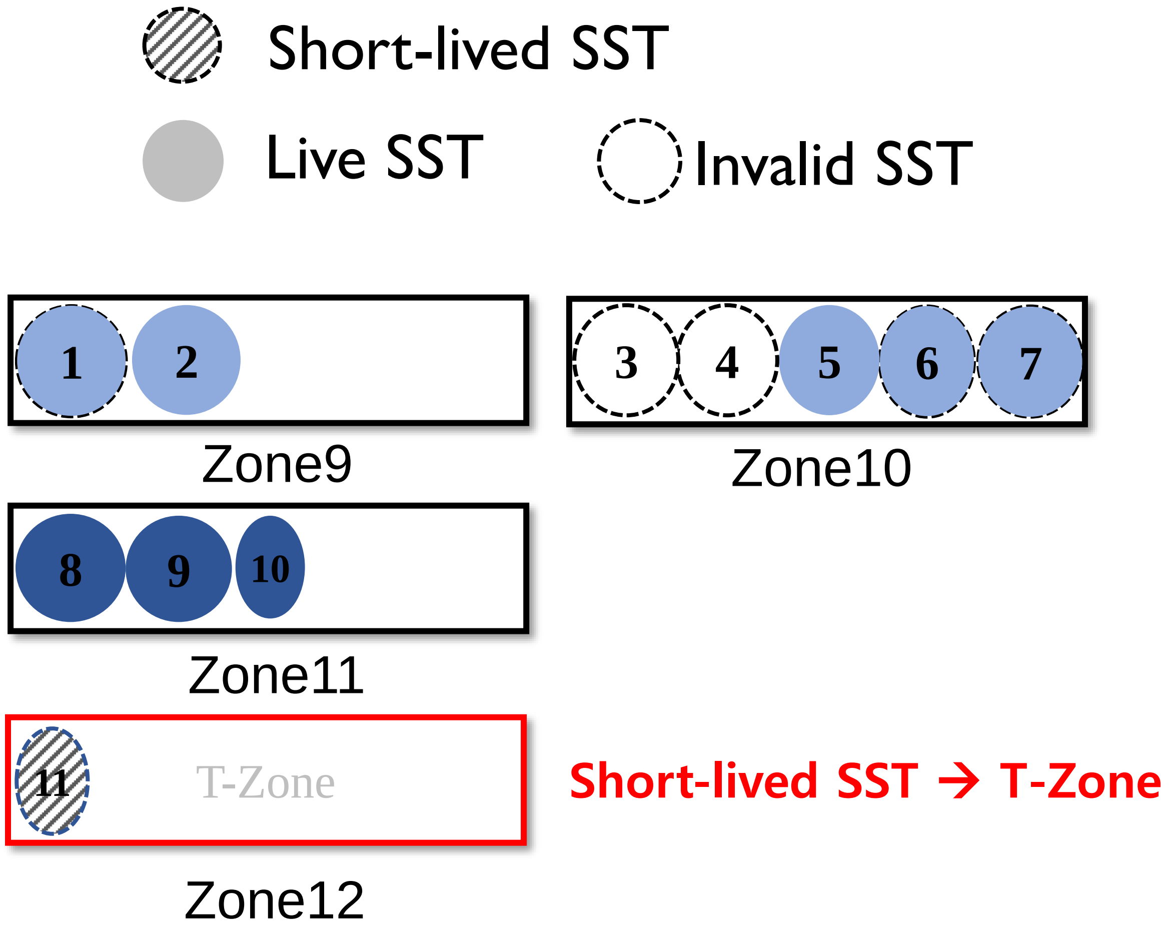
Solution for Long-Lived SST



Compaction involves all L_{i+1} SSTs between the current CP_i and the *Next* CP_i

Solution for Short-Lived SST

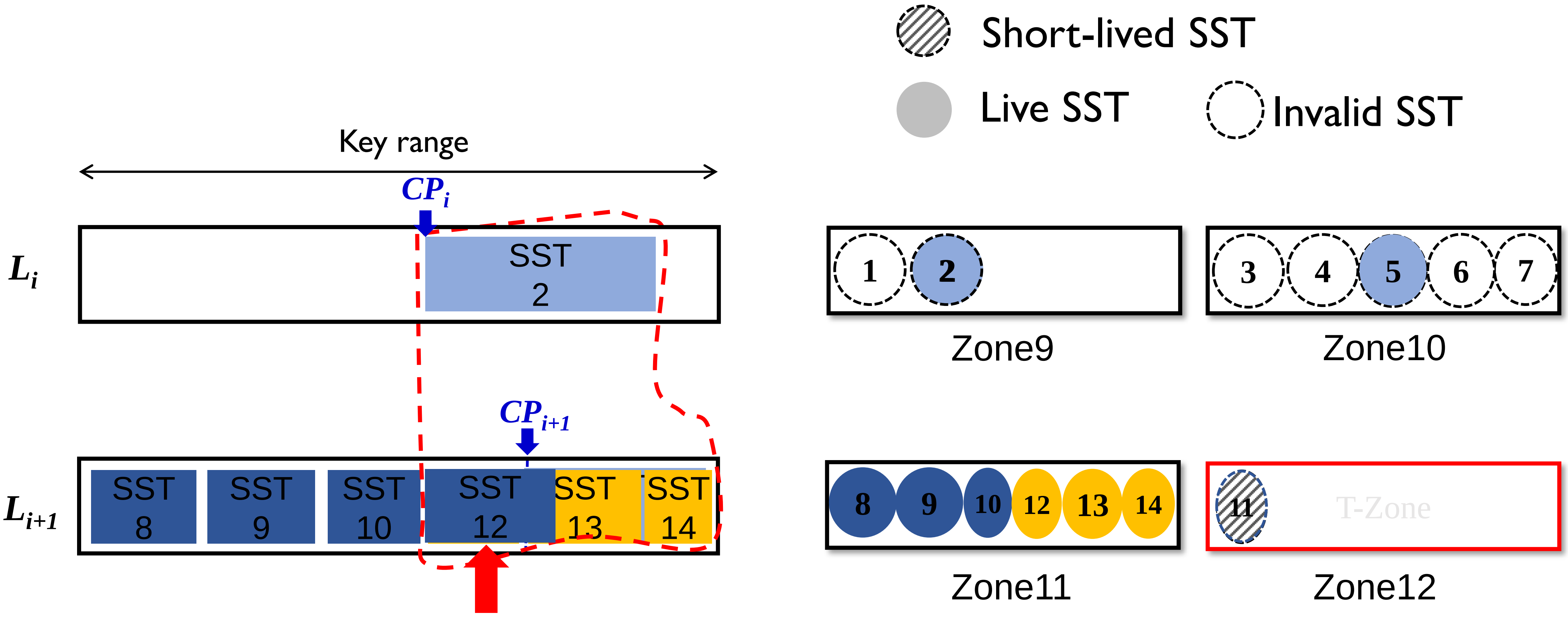




Split SST at *Next CP_i* to minimize the size of short-lived SST

Short-lived SST is written in T-Zone to prevent hole in normal Zone

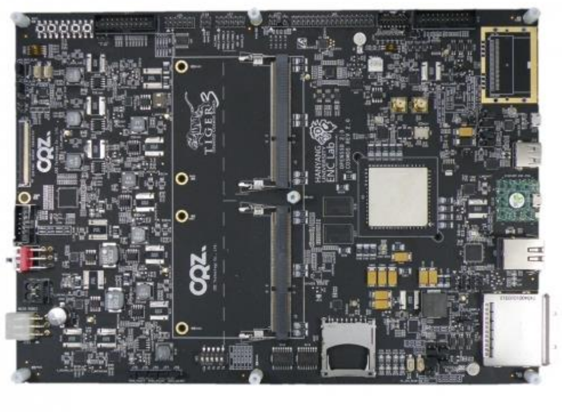
CP-Aware Split Algorithm



Split SST at CP_{i+1} to preserve compaction pointer

Evaluation Setup

- Comparisons
  - ➢BL : Baseline (use Infinite zone)
  - ➢GC : LevelDB + GC
  - ➢LS : LevelDB + Level Separation + GC
  - ➢Gear : GearDB
- Test environment

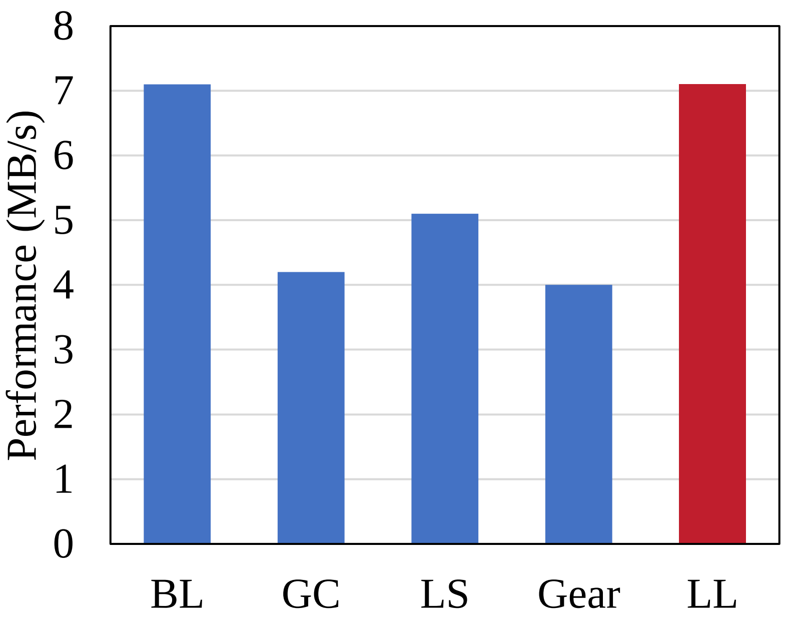


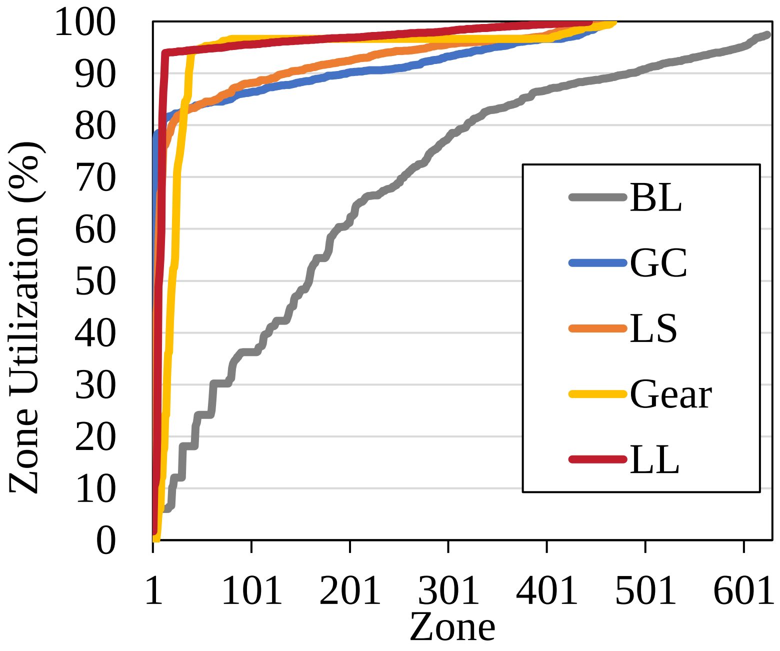
Cosmos+ OpenSSD

Linux	64-bit Linux 5.11.1
CPU	4GHz quad-core Intel i7-4790K CPU
Memory	16GB DDR4 , use 1.5GB user-level cache
ZNS	In-house ZNS SSD based on Cosmos+ OpenSSD
Defaults	LevelDB 1.19, key=16B, value=512B, SST size=4MB, GC=greedy, Zone=64MB

Performance & Zone utilization

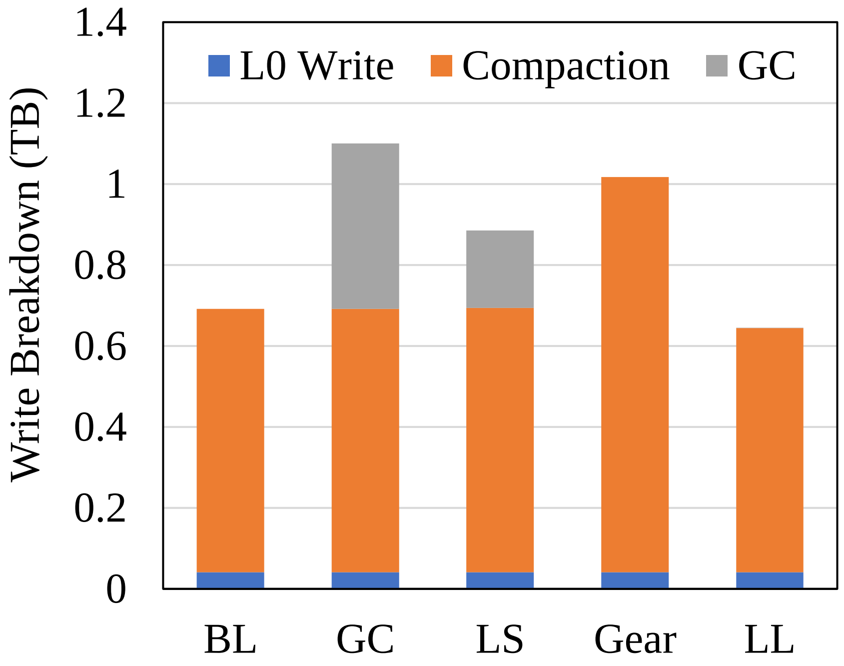
- BL and GC show trade off relationship between WA and SA
- LS has 1.3X better performance vs GC due to lower GC cost
- Gear has lower performance vs LS due to high-cost gear compaction
- LL-Compaction has 1.4X better performance vs LS, 1.8X better performance vs Gear





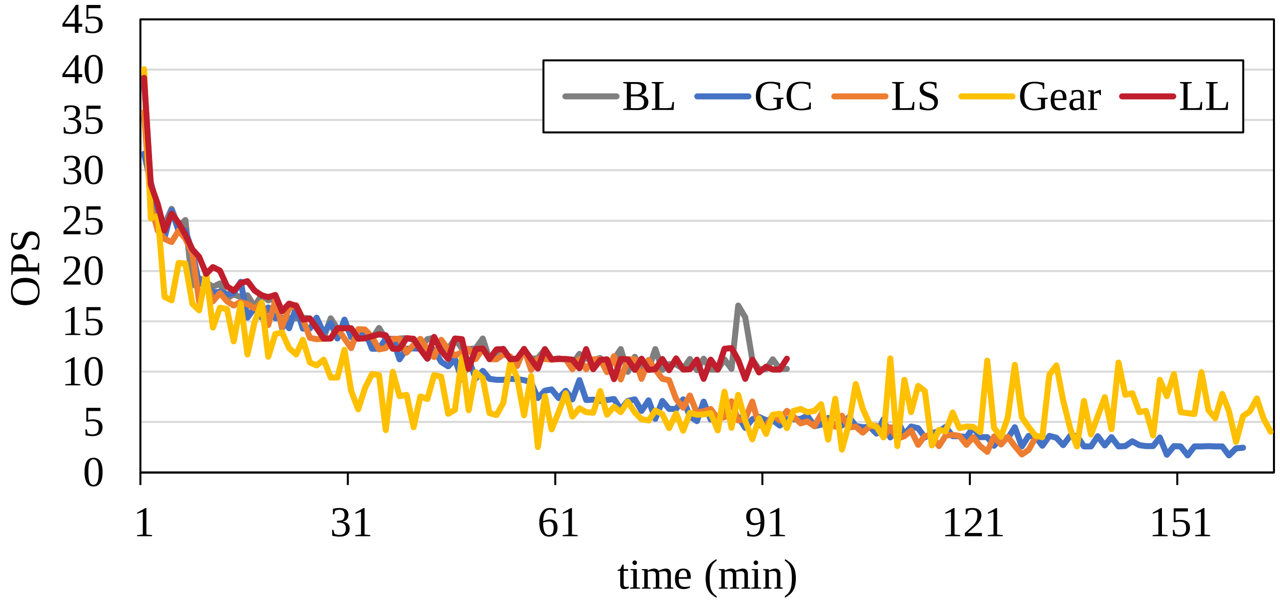
Write breakdown

- GC cost (LS, GC) makes write amplification
- Gear compaction needs more write than normal compaction
- LL-Compaction achieves less write cost vs BL → benefit by split policy



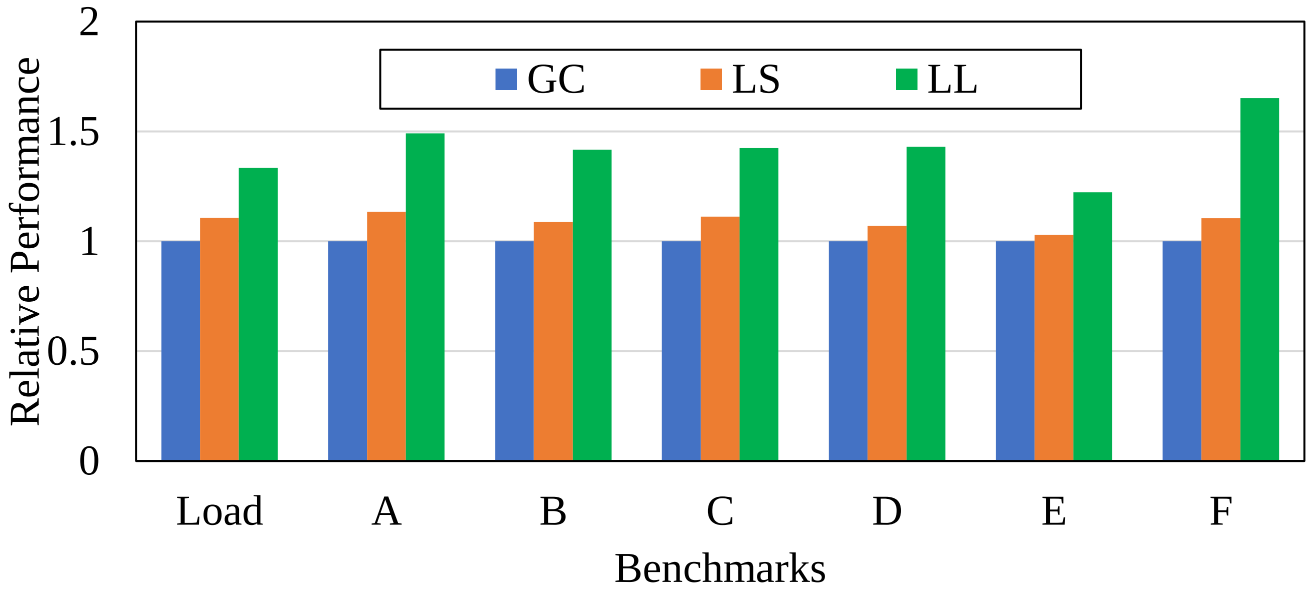
Incremental throughput

- BL & LL compaction achieve stable and fast performance
- Performance of GC & LS drops when GC occurred (55min GC, 75min LS)
- Performance of Gear compaction is not stable; compaction cost of Gear compaction is higher than normal compaction



Performance (Real workload)

- GearDB can't performs because GearDB needs more storage capacity
- LL-Compaction
  - 1.2~1.7X vs GC
  - 1.18~1.5X vs LS



Conclusion & Discussion

- **Current ZNS-unaware compaction can suffer from space amplification**
- **LL-Compaction**
  - ➢ Reduce space amplification without invoking host-managed garbage collection
  - ➢ 1.2~1.7X speed up for real workloads vs GC
- **Limitation of LL-Compaction**
  - ➢ Cannot use priority-driven compaction algorithms (e.g., RocksDB)
    - ➢ ex) Overlapping-key range (reduce compaction cost), age (reduce read cost)
  - ➢ We will analyze the impact of priority-driven compaction algorithms on GC

Thank You

Further Questions? wjdwldbs1@skku.edu