Z-LFS: A Zoned Namespace-tailored Log-structured File System for Commodity Small-zone ZNS SSDs

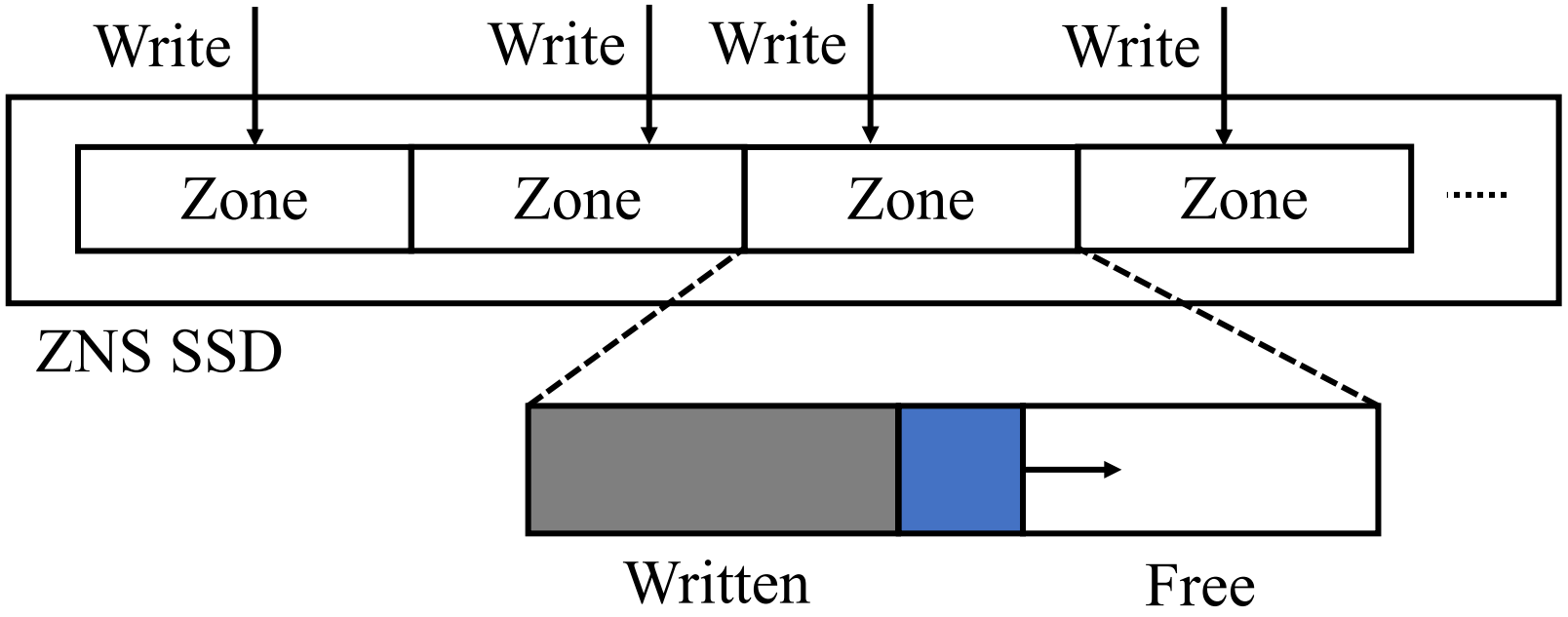
Inhwi Hwang[1], Sangjin Lee[3],
Sunggon Kim[2], Hyeonsang Eom[1], Yongseok Son[3]

[1] Seoul National University
[2] Seoul National University of Science and Technology
[3] Chung-Ang University

Zoned namespace SSD

❖SSD with zoned namespace (ZNS) interface

- Space is divided into fixed-size zones
- Data within a zone are written sequentially
- Data can be written in zones (i.e., active zones) in parallel



Constraints of ZNS SSD

❖Constraints #1. Sequential write constraint

- Data within a zone must be written sequentially
- A zone is explicitly reset for reuse

❖Constraints #2. Active zone constraint

- The maximum number of active zones is limited
- Data cannot be written in zones over the limitation

File system on ZNS SSD

❖Log-structured file systems

- Append-only data write
- Naturally be utilized for ZNS SSD
- F2FS supports ZNS SSD

❖Issues of F2FS from conventional namespace (CNS) SSD-based design

- Requirement of an additional CNS SSD
- Underutilization of small-zone ZNS SSD

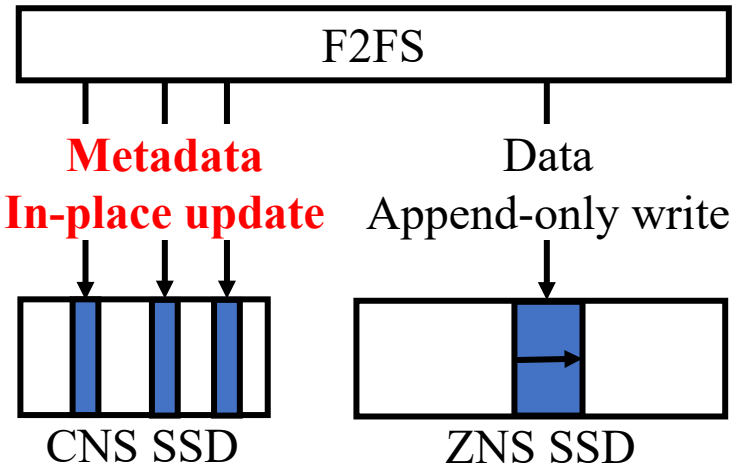
Requirement of an additional CNS SSD

❖Metadata management in F2FS

- Store LFS metadata on a fixed location
- Perform in-place metadata update
- Avoid update propagation of metadata

❖Metadata in F2FS on ZNS SSD

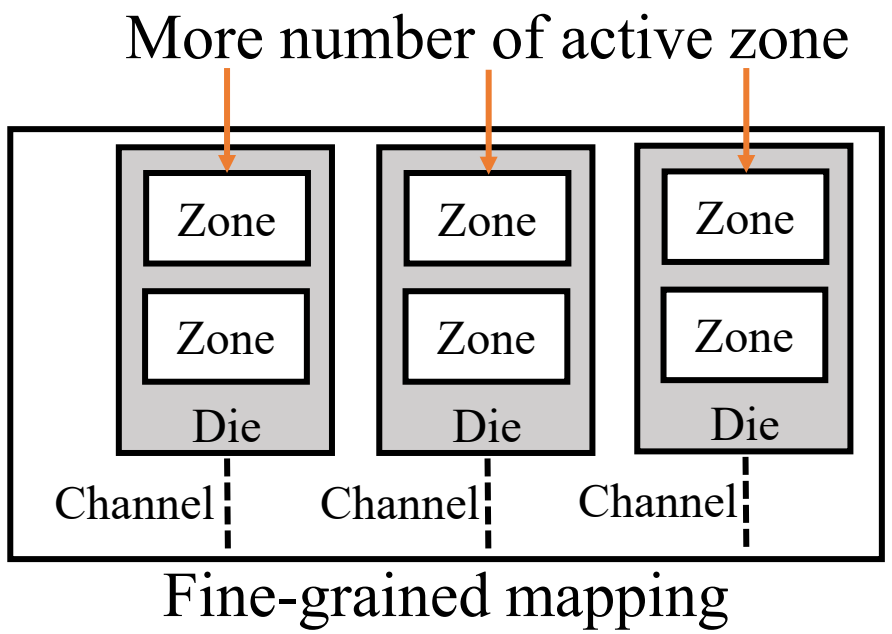
- Legacy design violates sequential write constraint
- Additional use of a CNS SSD for metadata
- Increase cost of a storage system

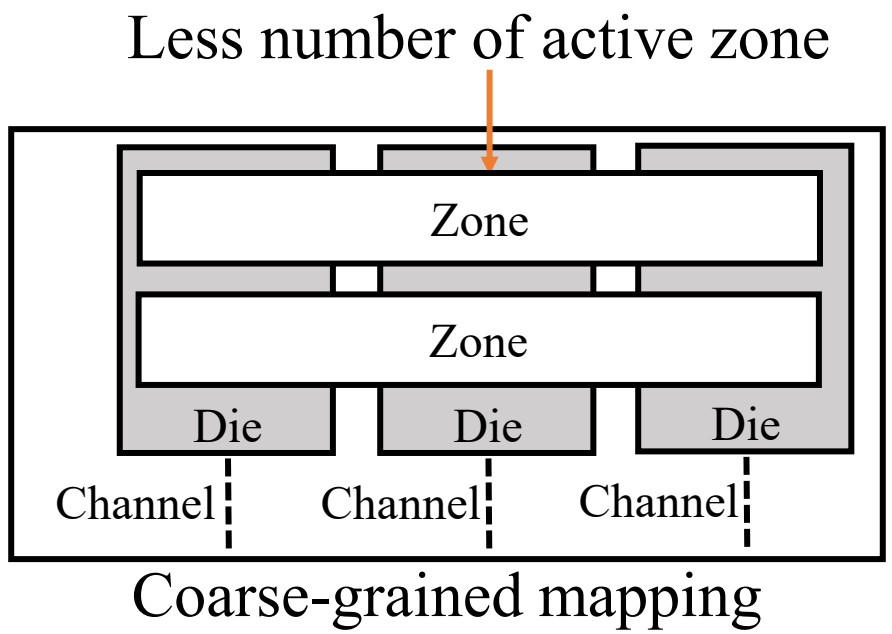


Underutilization of small-zone ZNS SSD

❖Small-zone ZNS SSD

- Fine-grained mapping between zone and internal resources
- The maximum number of active zones is large

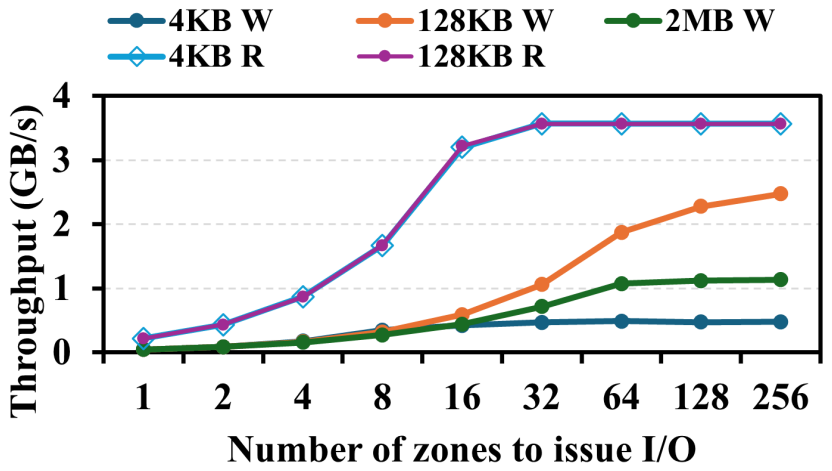




Underutilization of small-zone ZNS SSD

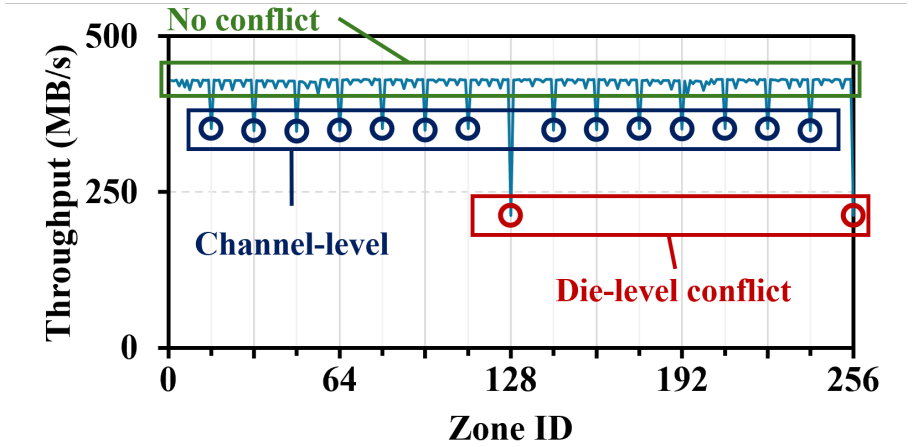
❖Active zone utilization

- More use of active zones means more use of internal resources
- Number of active zones significantly impact I/O performance



❖Resource conflict

- Zones mapped to overlapped resources decrease resource utilization
- Resource conflict degrades performance



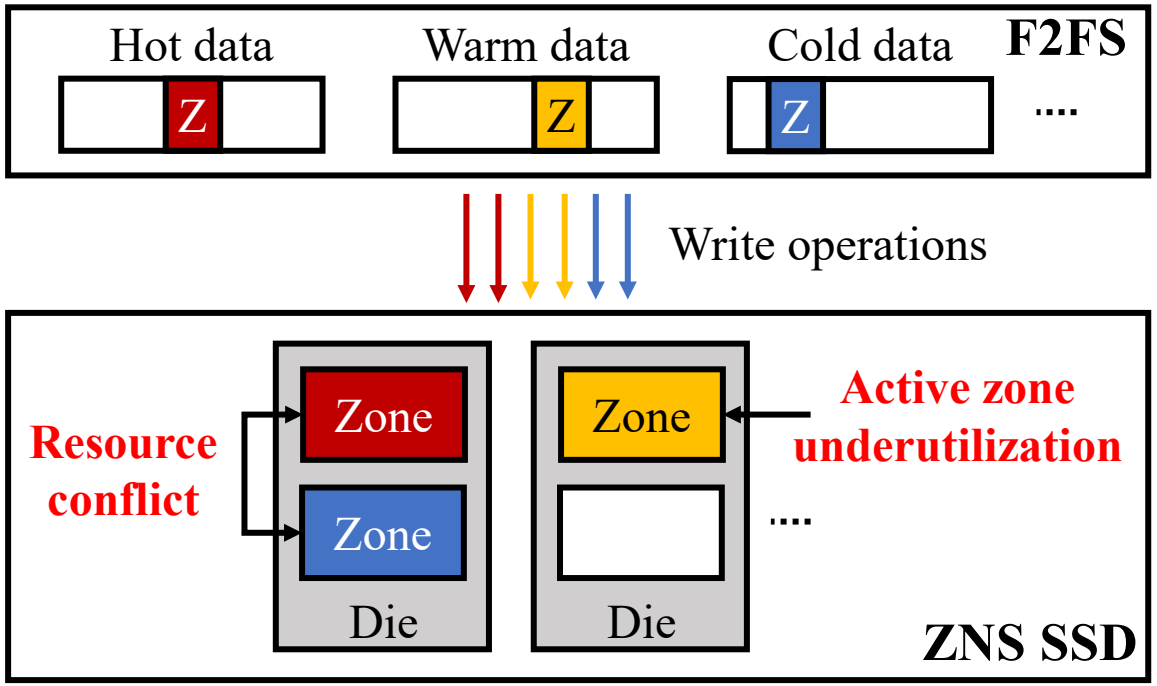
Underutilization of small-zone ZNS SSD

❖Data placement in F2FS

- Log data/node in six log streams
  - Temperature (hot/warm/cold) and type (data/node)
- Allocate an active zone for each log stream at a time

❖Resource underutilization of F2FS

- Limited utilization of active zones
- No consideration of resource conflict
- Lead performance degradation



Challenges

❖ **Enabling append-only metadata on ZNS SSD**

- Naïve append-only approach can increase write amplification

❖ **Maximizing utilization of active zones under ZNS constraints**

- Distributing equal number of active zones is not enough
- Write intensity of log stream depends on workloads

❖ **Mitigating SSD internal resource conflict among zones**

- Consideration of mapping between zones and internal resources is needed

Z-LFS

❖Goals

- Deploy file system on a standalone ZNS SSD while minimizing WAF
- Maximize the resource utilization of small-zone ZNS SSD
- Software-based approach with commodity ZNS SSD

❖Key design

- Append-only metadata management
- Speculative log stream management
- Conflict-aware zone allocation

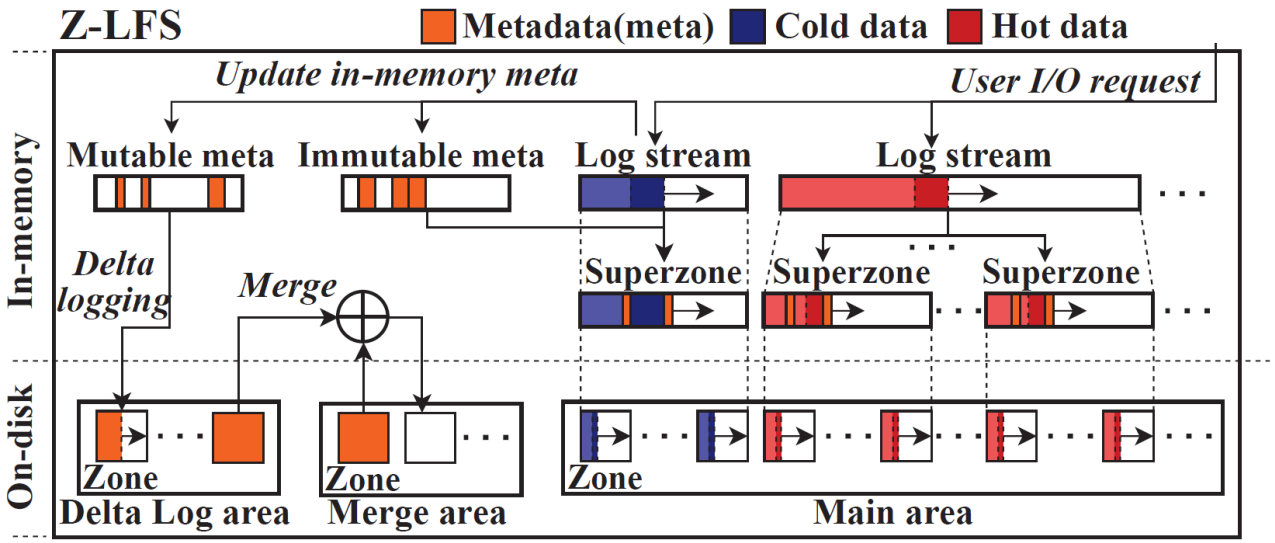
Overview of Z-LFS

❖In-memory structure

- Six log streams based on temperature (hot/warm/cold) and type (data/node)
- Group multiple zones into a superzone
- Two LFS metadata based on life cycle

❖On-disk layout

- Main area for data, node, and immutable metadata
- Delta log area and merge area for mutable metadata



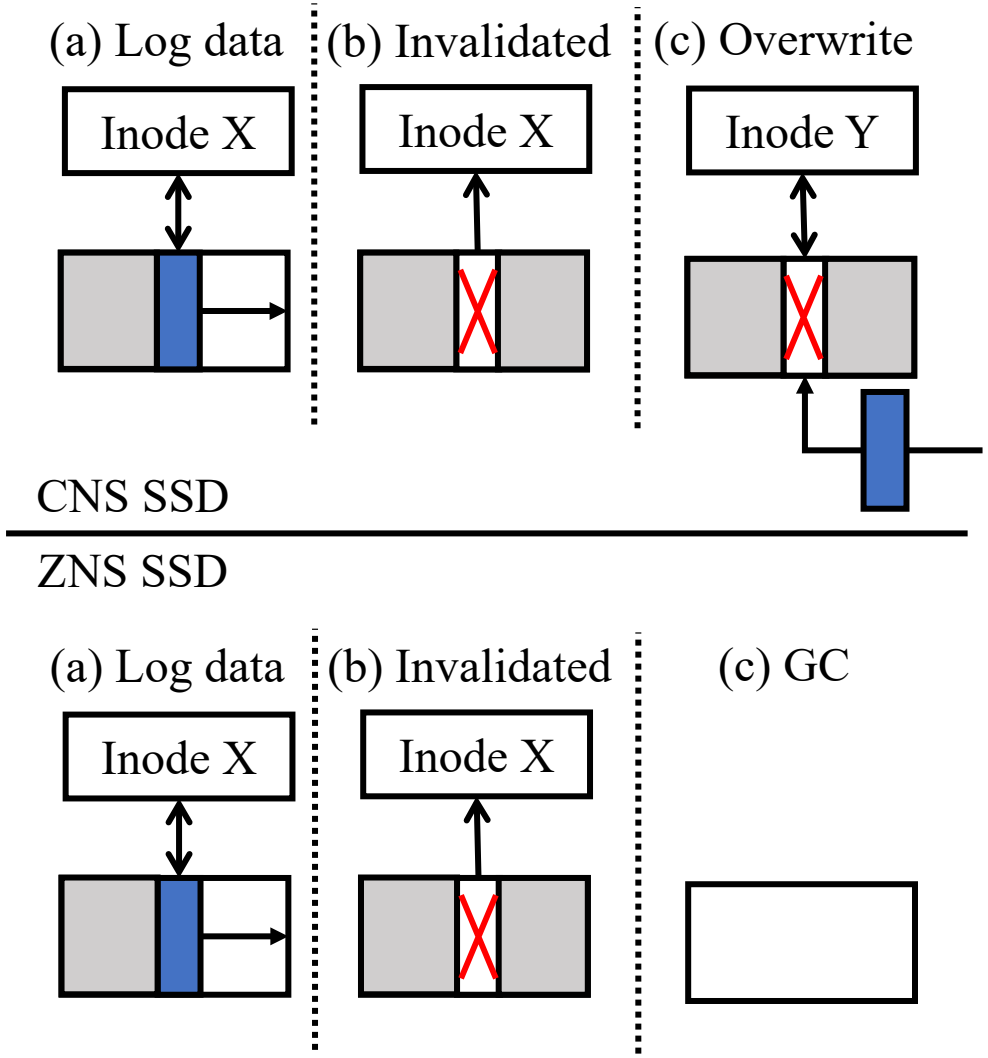
Insight for append-only metadata management

❖LFS metadata on CNS SSD

- Write operations on a segment are not strictly sequential due to optimizations
  - E.g., slack space recycling of F2FS
- All metadata have distinct life cycle from segments

❖LFS metadata on ZNS SSD

- Once a segment is written, it is never updated until GC due to ZNS constraint
- Some metadata share life cycle with their corresponding segment
  - E.g., reverse mapping of data blocks



Append-only metadata management - life cycle awareness

❖Immutable metadata

- Write-once, never updated
  - E.g., reverse mapping of data blocks
- Relatively large size (4KB for a segment)
- Append to the end of a segment

❖Mutable metadata

- Can be updated after written
  - E.g., valid block count/bitmap, logical block address of nodes
- Relatively small size (tens of bytes for an entry)
- Metadata delta logging

Append-only metadata management - delta logging

❖Delta logging

- Log only changed entries of mutable metadata
- Store on fixed delta log area not to track metadata location

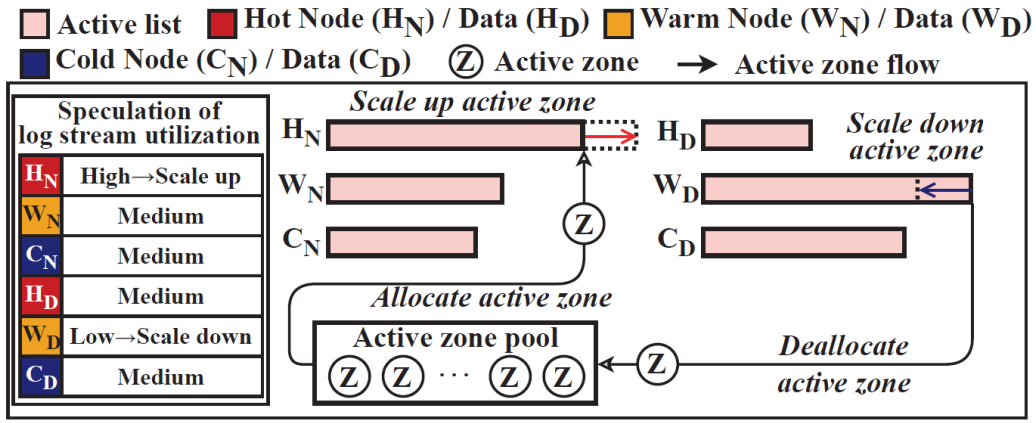
❖Asynchronous merge

- Clean delta log area
- Collect metadata on the log and make full table of metadata
- Performed in background

Speculative log stream management

❖Quota-based active zone management

- Determine the optimal number of active zones for each log stream
- Based on write intensity of log streams
- Scales the number of active zones



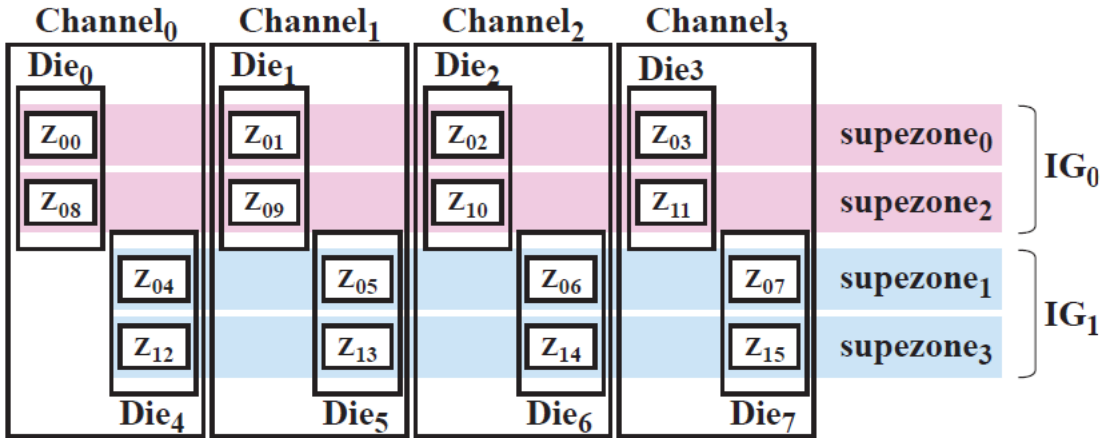
❖Active zone quota allocation

- A: available number of active zone, W_i: write request volume
- the number of active zones for log stream i: $A_i = A \times \frac{W_i}{\sum W_i}$

Conflict-aware zone allocation

❖Superzone

- Group of consecutive zones which is not overlapped in channel-level
- Avoid channel-level conflict



❖Interference group (IG)

- Group of superzones mapped to the same dies
- Avoid die-level conflict

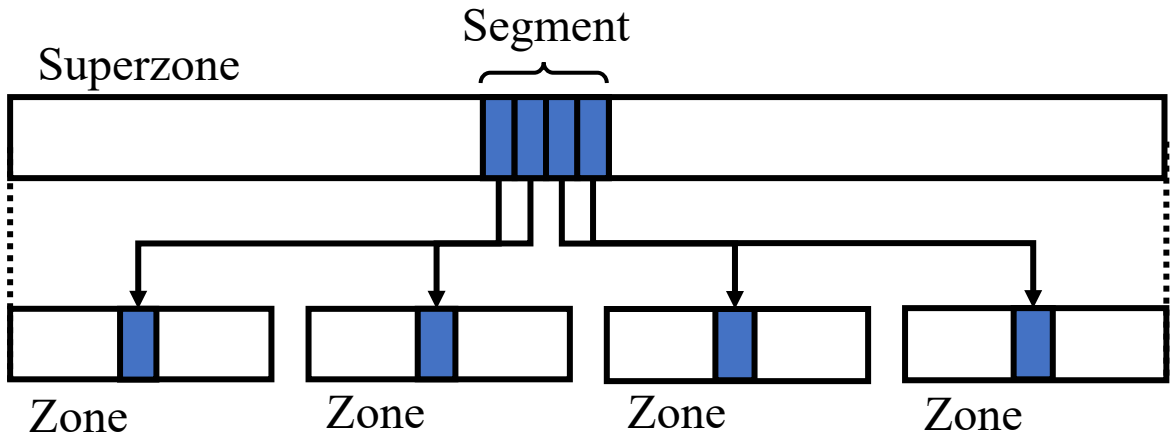
Conflict-aware zone allocation

❖Superzone allocation for log stream

- Allocate active zones in units of superzones
- Prioritize superzones from non-overlapping IGs to avoid conflict

❖Mapping between segment and superzone

- Writes on segment are distributed on zones of a superzone
- Sequential write/read on a segment can utilize the zone-level parallelism



Evaluation setup

❖Testbed

- i7-13700K 3.4GHz CPU, 32GB main memory
- Commodity small-zone ZNS SSD
  - Total capacity 3.92TB
  - Zone size: 96MB
  - Maximum number of active zone: 384

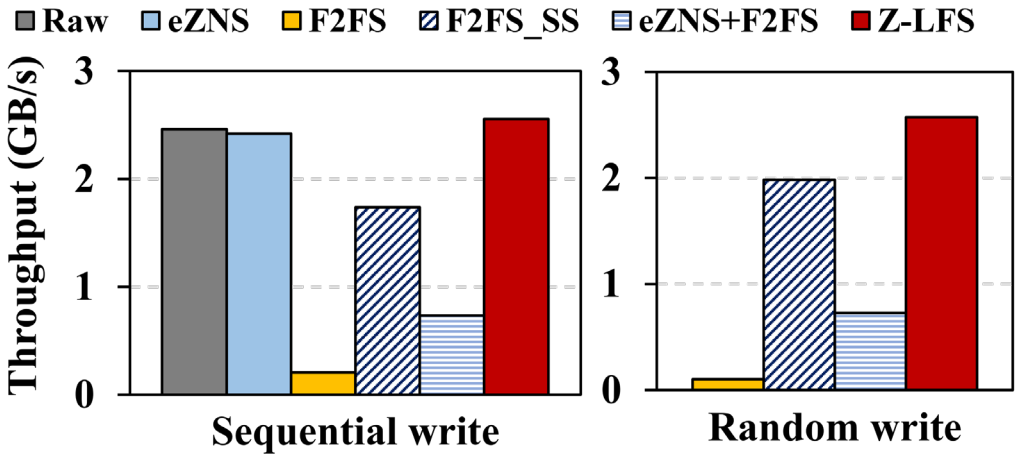
❖Comparison

- F2FS: Vanilla (F2FS), with static striping (F2FS_SS)
- eZNS: without file system (eZNS), with F2FS (eZNS + F2FS)
- All existing techniques use additional SSD for metadata

Micro-benchmark: write throughput

❖Compared with F2FS_SS

- Up to 47% higher throughput
- Due to better exploitation of zone-level parallelism



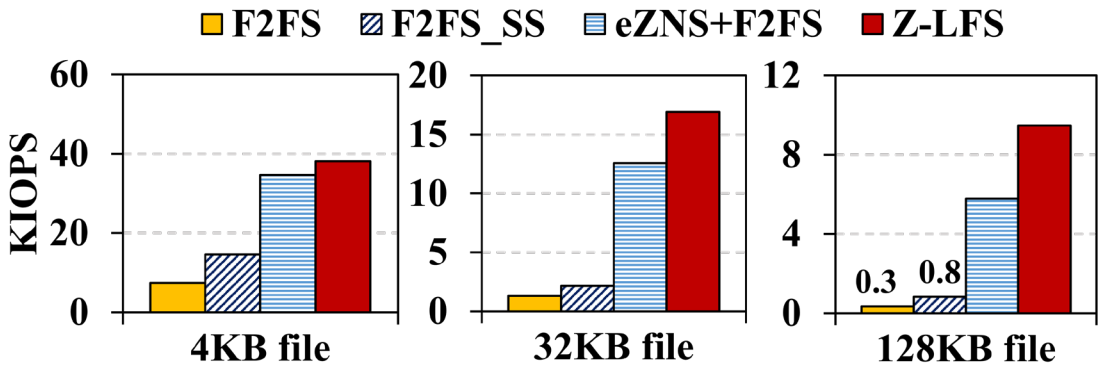
❖Compared with eZNS+F2FS

- Up to 3.5x higher throughput
- Due to lack of cross-layer awareness, eZNS achieve sub-optimal utilization of active zone

Micro-benchmark: metadata intensive and write amplification

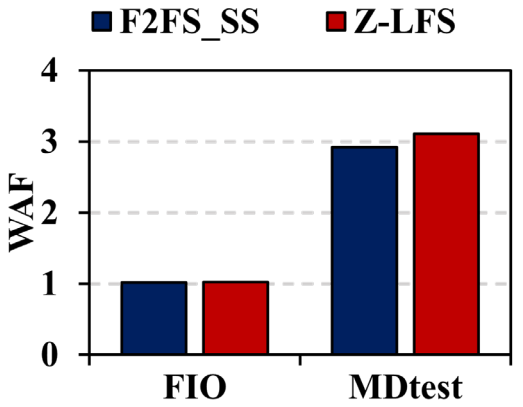
❖Metadata-intensive (MDtest)

- Node block writes on ZNS SSD is a bottleneck point
- Better zone-level parallelism benefits in metadata-intensive workload



❖Write amplification

- Similar write amplification on both data and metadata intensive workloads
- Append-only metadata management minimizes metadata write amplification



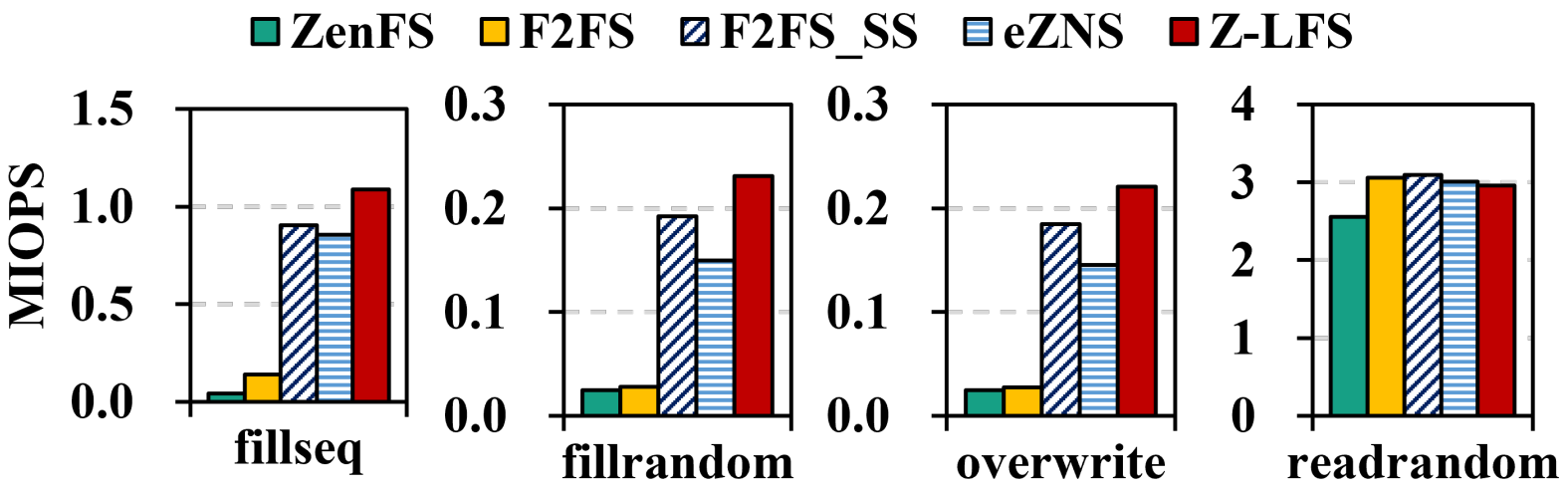
Real-world application: RocksDB dbbench

❖**Write-intensive**

- Z-LFS outperform existing techniques due to better zone-level parallelism

❖**Read-intensive**

- Without overhead in read operations



Conclusion

❖Issues on file system with small-zone ZNS SSD

- Need of additional CNS SSD
- Low resource utilization of ZNS SSD

❖Z-LFS

- ZNS-tailored LFS for commodity small-zone ZNS SSD
- Deploy file system on a standalone ZNS SSD
- Improve zone-level parallelism
- Avoid internal resource conflict

Thank you

Q & A