Can ZNS SSDs be Better Storage Devices for Persistent Cache?

Chongzhuo Yang, Zhang Cao, Chang Guo,

Ming Zhao, **Zhichao Cao**

Intelligent Data Infrastructure Lab (***ASU-IDI***)

School of Computing and Augmented Intelligence

Arizona State University

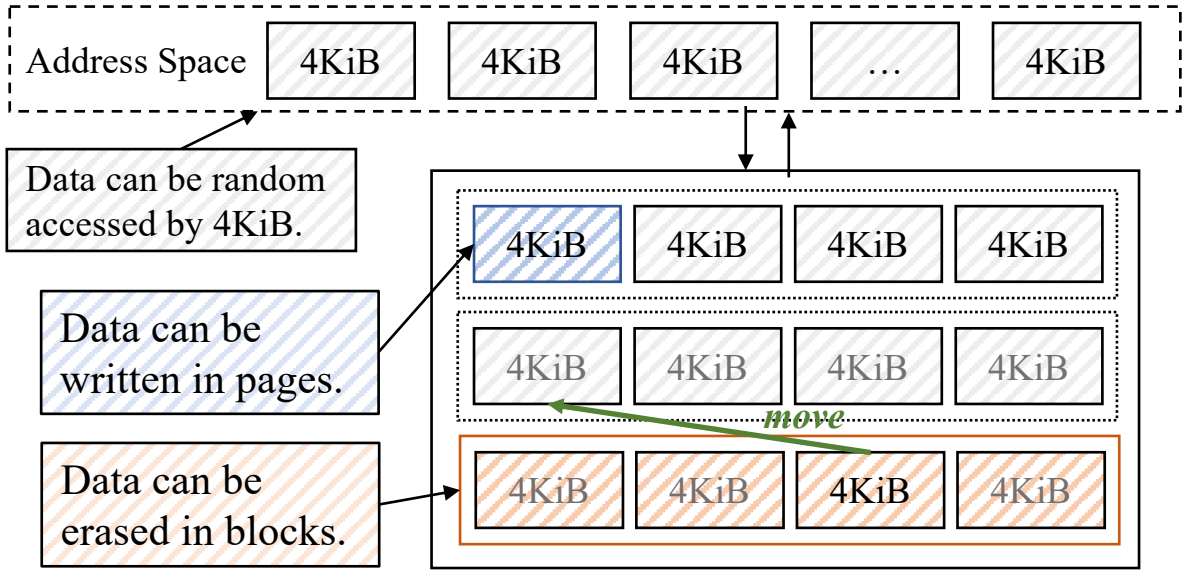
Overview

- Background
- Motivations
- Challenges
- Three Possible Solutions for ZNS Cache
  - Use a ZNS Compatible File System?
  - Match the Cache Management Unit?
  - Need a Simpler Middle Layer?
- Experiments
  - Micro Evaluations
  - Evaluations on RocksDB
- Further Research

Background

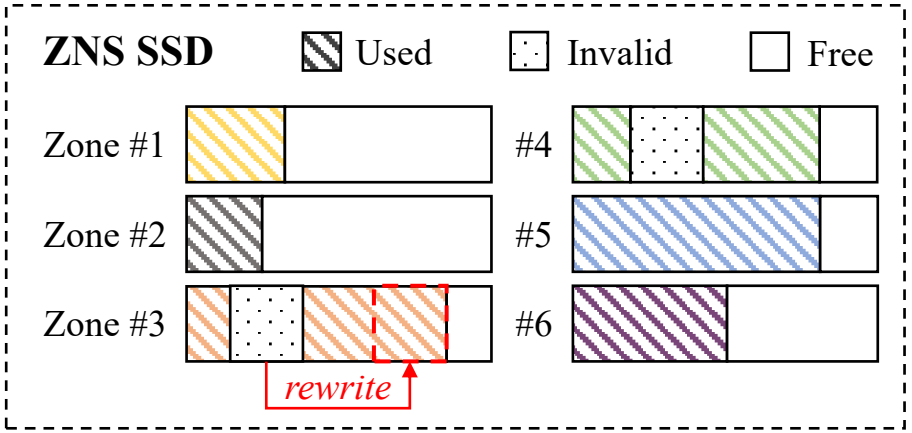
There is the **semantic gap** between Block Interface and Nand Flash.

- The Characteristics of NAND Flash
  - Data can be written in pages (e.g., 4KiB).
  - Do not support in-place updates.
  - Data must be erased in blocks (e.g., 256KiB).
- The Limitation of Block Interface
  - Need the GC to support 4KiB in-place updates.
  - Can not be aware of the erase behavior.
  - Unstable latency due to uncontrollable GC.



Motivations

- Flash Cache
  - Flash-based cache are widely used, for example, CacheLib from Meta.
  - Flash cache usually organize cache items as large segments (e.g., Region), written to files or directly use raw devices to reduce I/O overheads and reduce write amplifications
- The Caching Workloads
  - The space is almost fully utilized.
  - Many random updates worsen the WA.
- Zone Interface Is More Friendly!
  - Only support sequential writes in Zone.
  - Require reset to update data in Zone.
  - Client-level FTL and GC.



- Potential benefit of using ZNS SSDs for persistent cache.
  - More space due to less device-level over-provisioning.
  - Client-level GC reduces tail latency.
  - More options to reduce WA.

Challenges

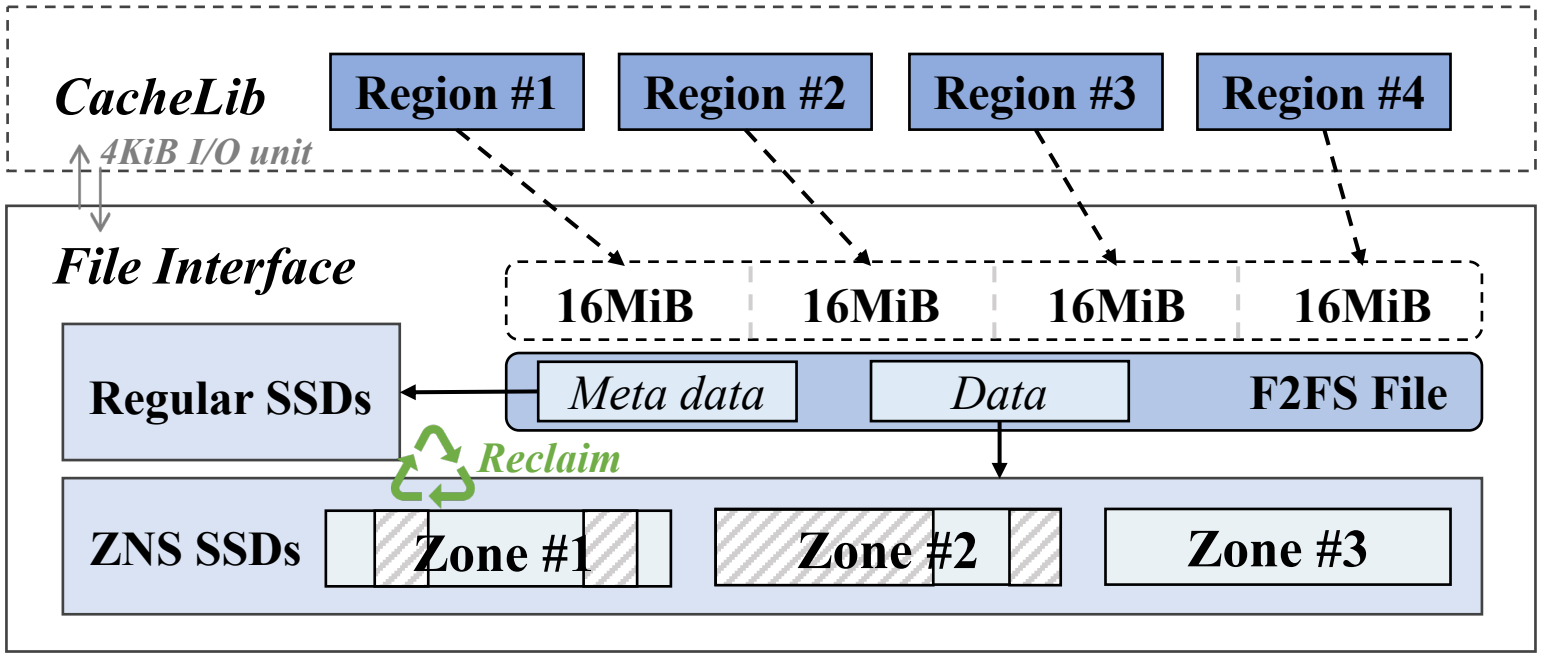
- Existing persistent cache engines are either based on file systems or raw SSDs, they are not ZNS compatible:
  - They have a lot of in-place updates
  - Interface is not compatible
  - No-zone management
- How to design the GC mechanism to achieve a better tradeoff ?
  - Write amplification
  - Cache performance
  - Cache hit ratio
- How to address the wearing issue of SSDs?

Three Possible Solutions to Use ZNS SSD for Flash Cache

- File-Cache
  - Using a ZNS-compatible file system.
- Zone-Cache
  - Directly manages the ZNS SSDs, which matches cache management unit (region) with zone size.
- Region-Cache
  - Using a simple middle layer to provide a region interface.

S1: Use a ZNS Compatible File System?

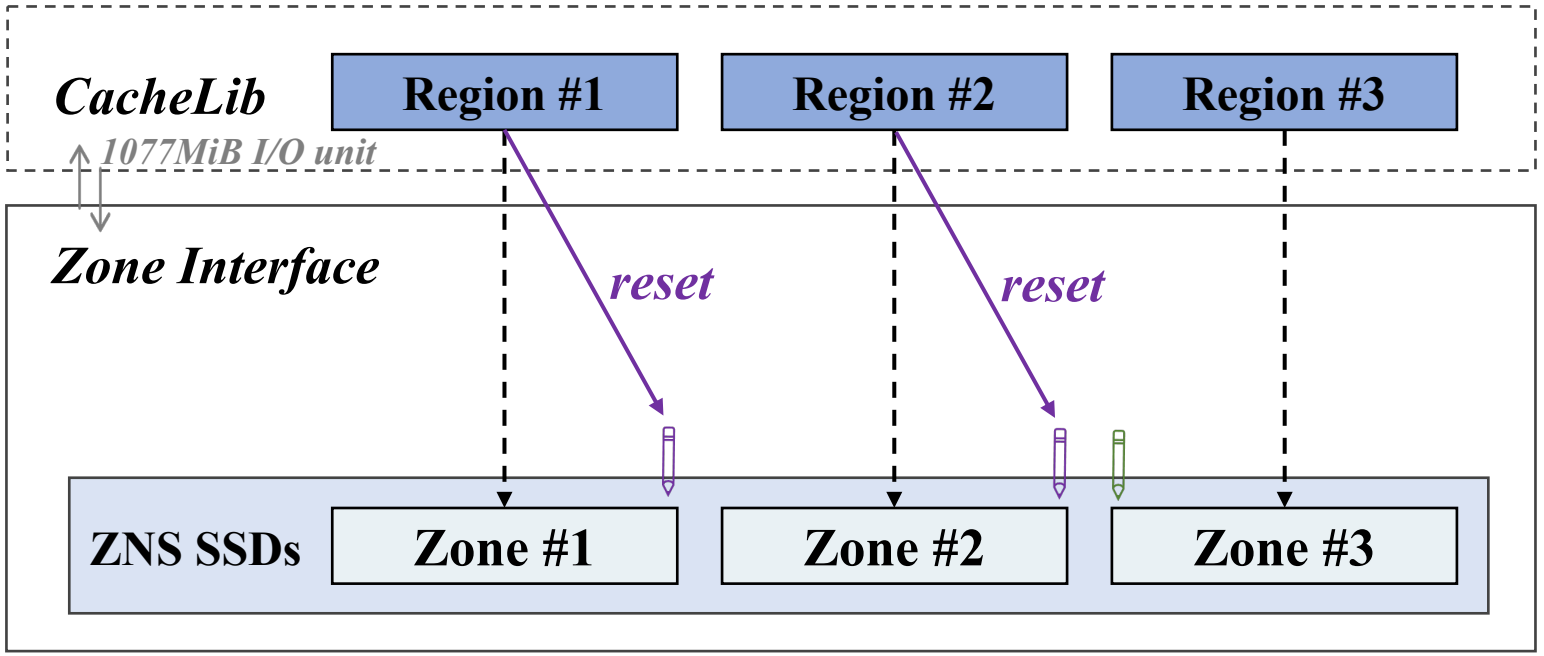
- Using file system to provide a general file interface.
  - Additional overheads for general use cases.
  - Additional regular SSDs to build the filesystem (e.g., F2FS).
  - Full transparency makes it hard to optimize GC.



S2: Match the Cache Management Unit with Zone?

- Directly uses the zone interface.
  - Less mapping overheads.
  - GC-free design and no write amplification.
  - No over-provisioning space.

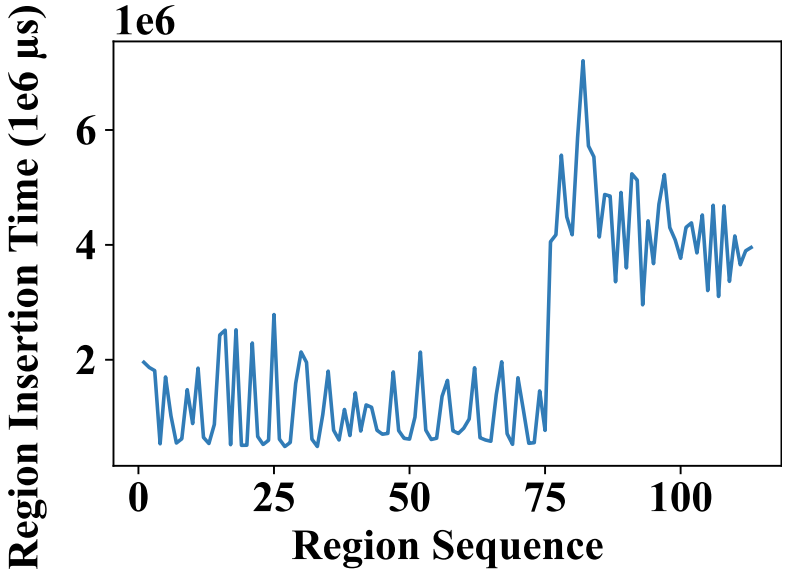
🤔 Perfect design?



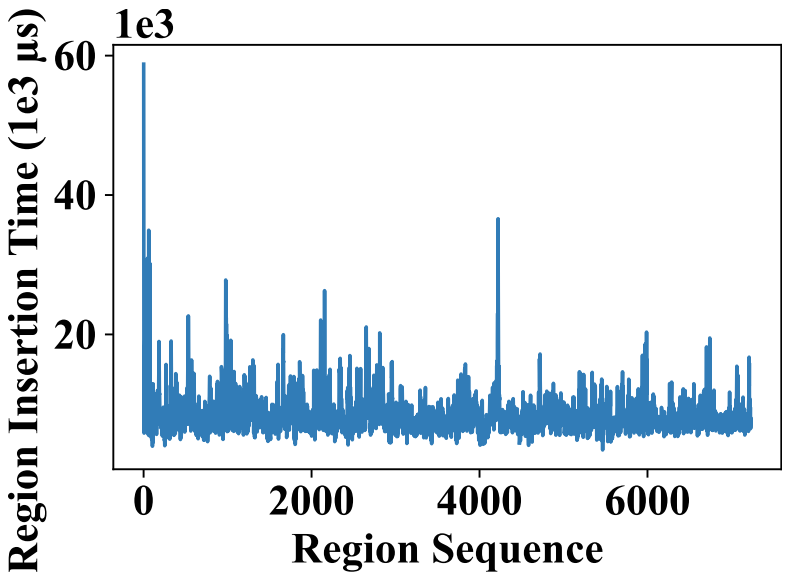
Match the Cache Management Unit?

- Directly uses the zone interface.
  - Less mapping overheads.
  - GC-free design and no write amplification.
  - No over-provisioning space.

The large region size incurs longer region insertion time and larger memory consumption!



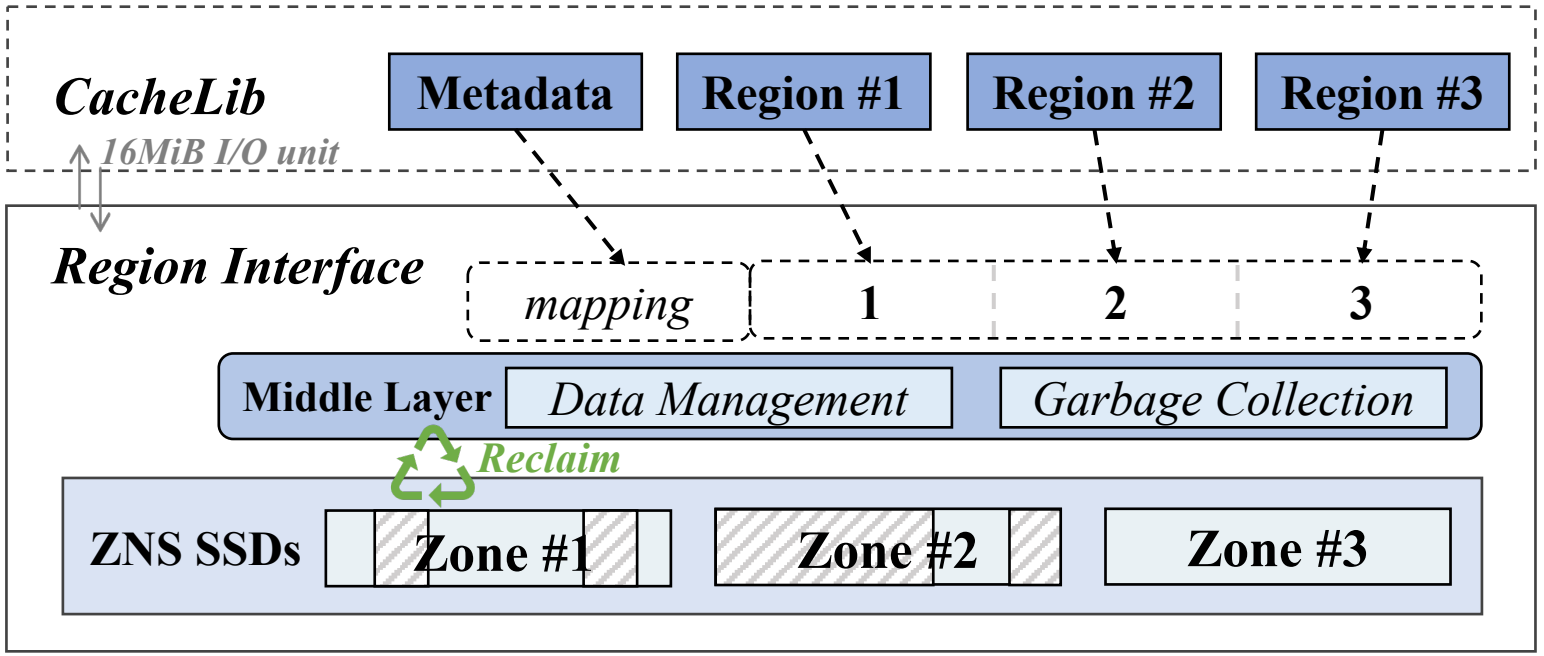
Large Region



Small Region

S3: Need A Simpler Middle Layer?

- Using a middle layer to provide region interface.
  - Dedicated for CacheLib.
  - Also needs OP and GC.
  - Regions can have a suitable size.



Implementation and Evaluation

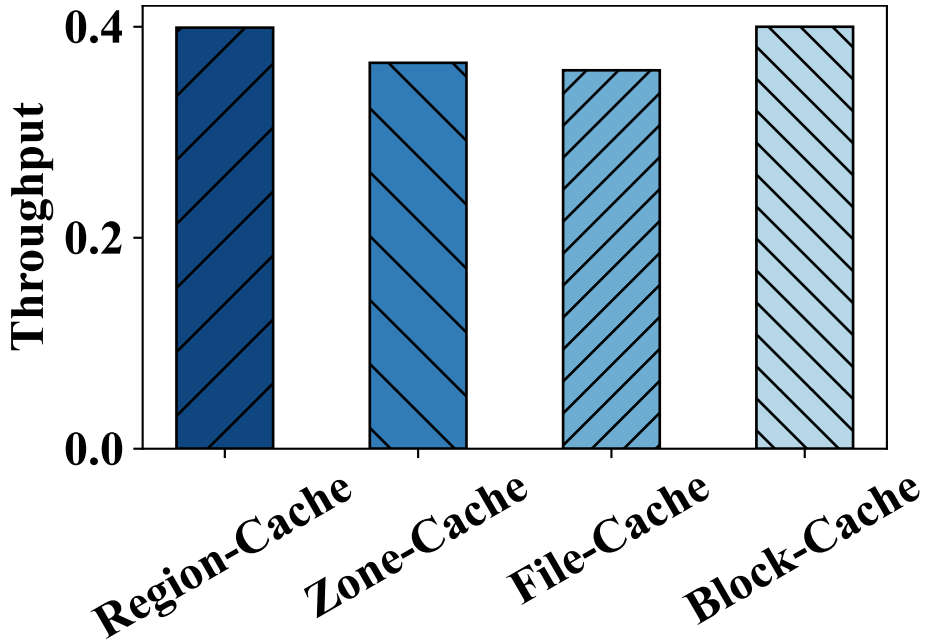
- Implementation details
  - Based on CacheLib, RocksDB, and F2FS
- Experimental setup
  - ASUSTeK ESC4000 server with Intel(R) Xeon(R) Silver 4210 CPUs and 187GiB DRAM memory.
  - The ZNS SSD is 1TB Western Digital Ultrastar DC ZN540 with 904 zones and the zone size is 1077MiB.
  - The regular SSD is a hardware-compatible 1TB SN540 SSD.
- Measurement Metrics
  - Throughput
  - Latency
  - Hit ratio

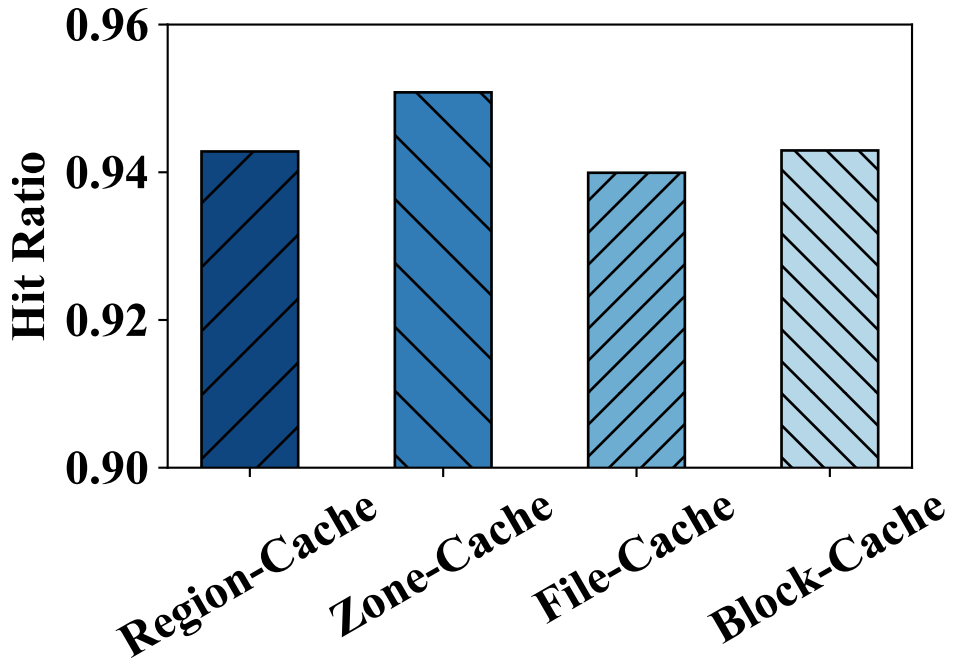
Micro Evaluations

- Three Solutions for ZNS Cache
  - The Block-Cache is the cache using regular SSDs.
  - Zone-Cache can get highest hit ratio due to its GC-free design.
  - Region-Cache can get high throughput (similar with Block-Cache).

Table 1: WA Factor under different OP ratios.

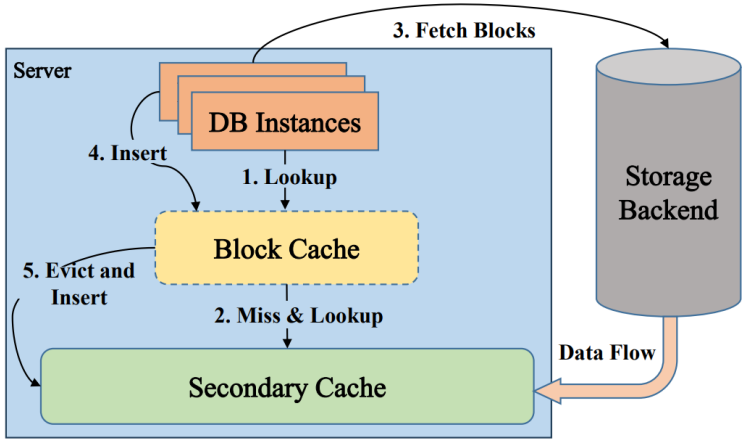
Scheme	10%	15%	20%
Region-Cache	1.39	1.30	1.15
File-Cache	1.25	1.19	1.11

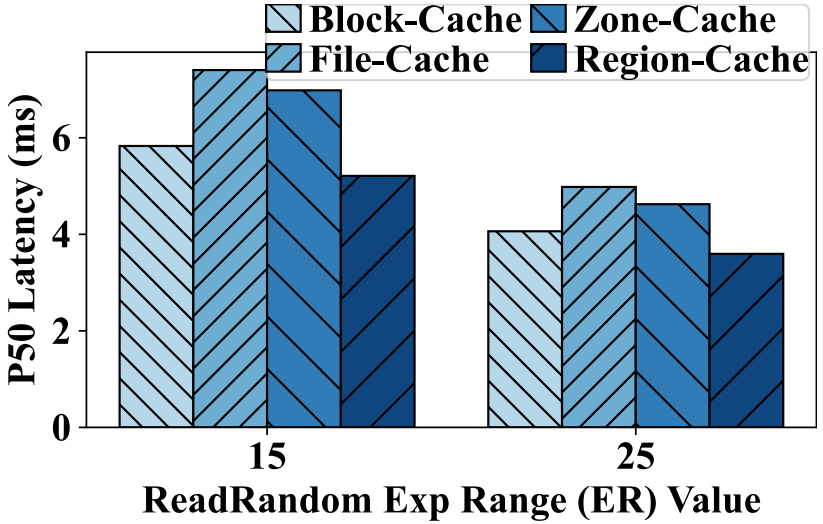


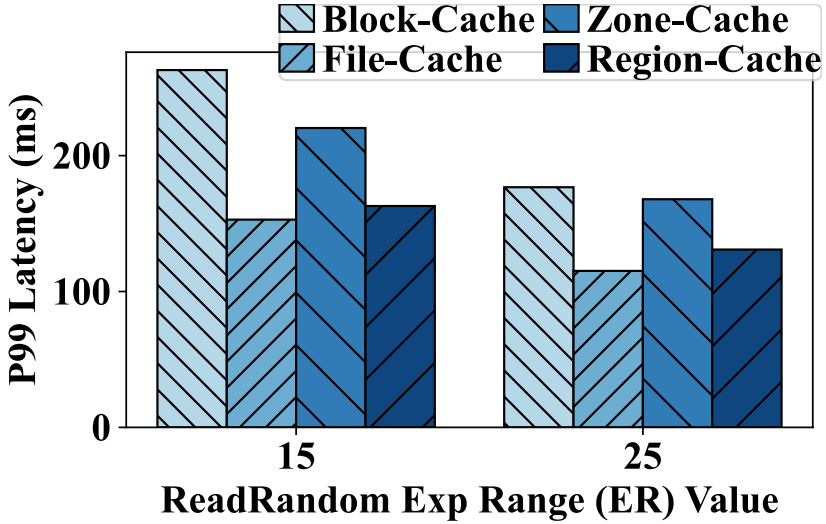


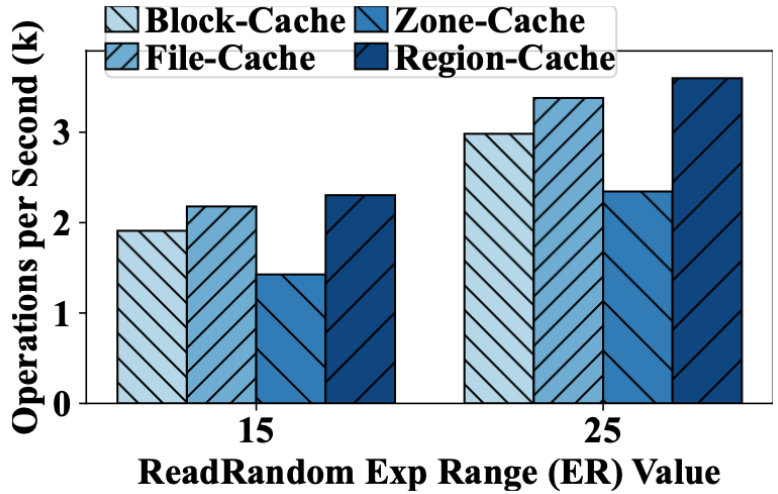
Evaluations on RocksDB

- The evaluations as the secondary cache of RocksDB
  - Region-Cache have lowest latency and highest throughput than other designs.
  - Block-Cache have highest tail latency (p99) than ZNS-based cache.









Conclusion and Further Research

- In this paper, we propose, analyze, and evaluate three possible schemes to use ZNS SSDs for persistent cache. Our findings suggest that ZNS SSDs can be better storage devices for persistent cache compared with regular SSDs, which is also verified by using ZNS SSD as a secondary cache for RocksDB.

- Future Research Directions
  - Consider to add hint mechanism to better management the zones
  - Discover more eviction policies.
  - Design new eviction policy which can achieve device-cache-application co-designs

Thank You!

Q & A

ZNS Cache

Contact

Zhichao Cao (zhichao.cao@asu.edu)

Source Code: https://github.com/asu-idi/ZNS-Cache

Seed Questions

- Why choose ZNS SSDs to optimize the cache? FDP?
- Will the large zone size bring more WA than a smaller zone size?
- Why Block-Cache have the highest tail latency?
- What difference between Region-Cache and Fairy WREN will appear in OSDI 24?