The Introduction to ZNS SSD

and

its related literature

Presented by Chen Haodong

Supervised by Zhao Yumiao

HDU – Center of Cloud computing

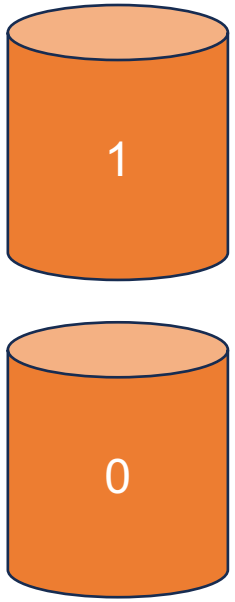
Outline

- 1. Basic characteristics of flash SSD
- 2. Conventional SSD
- 3. ZNS SSD
- 4. Related literatures
- 5. Future work

1. Basic characteristics of flash SSD

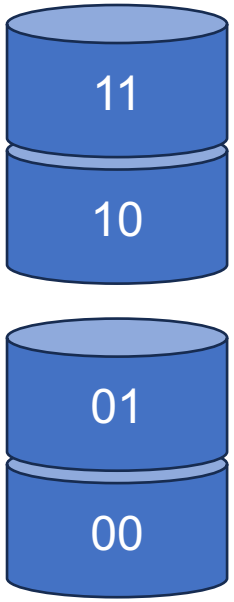
1.1. Flash Memory Cell

Single Level Cell



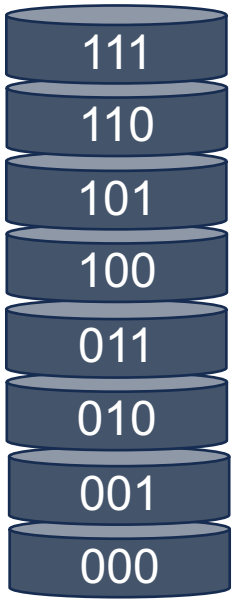
1 bit / cell
100,000 P/E cycles

Multiple Level Cell



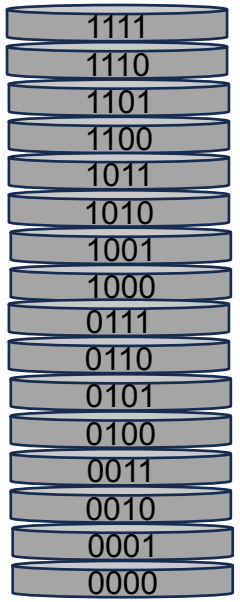
2 bits / cell
100,00 P/E cycles

Triple Level Cell



3 bits / cell
3,000 P/E cycles

Quad Level Cell



4 bits / cell
1,000 P/E cycles

1. Basic characteristics of flash SSD

1.2. Flash Array Structure

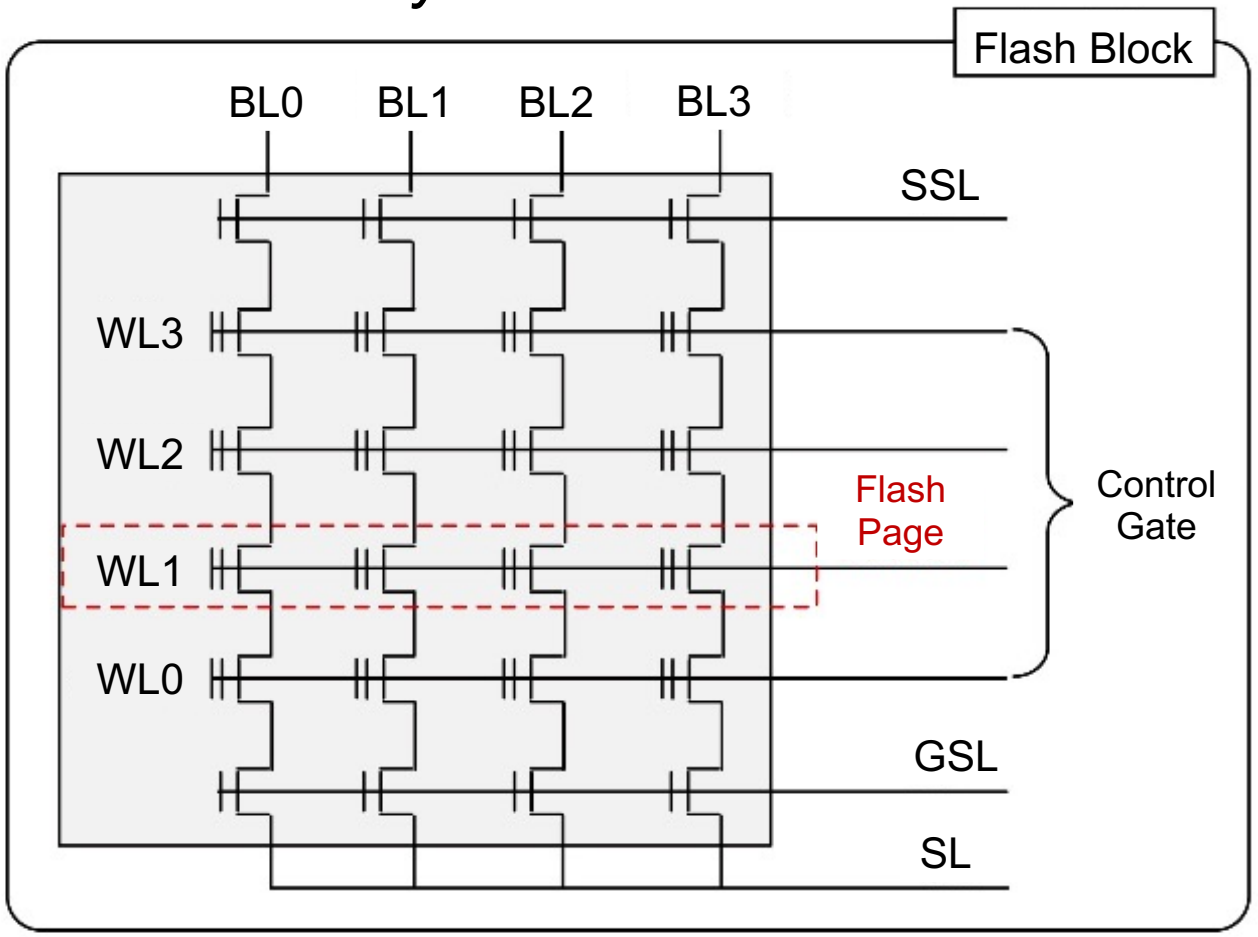


Fig. 1 Architecture of flash memory array

Flash Page:

- Smallest flash array structure
- Constituted by flash memory cells

Word Line:

- Each line constituting a flash Page

1. Basic characteristics of flash SSD

1.3. Basic operations of Flash SSD

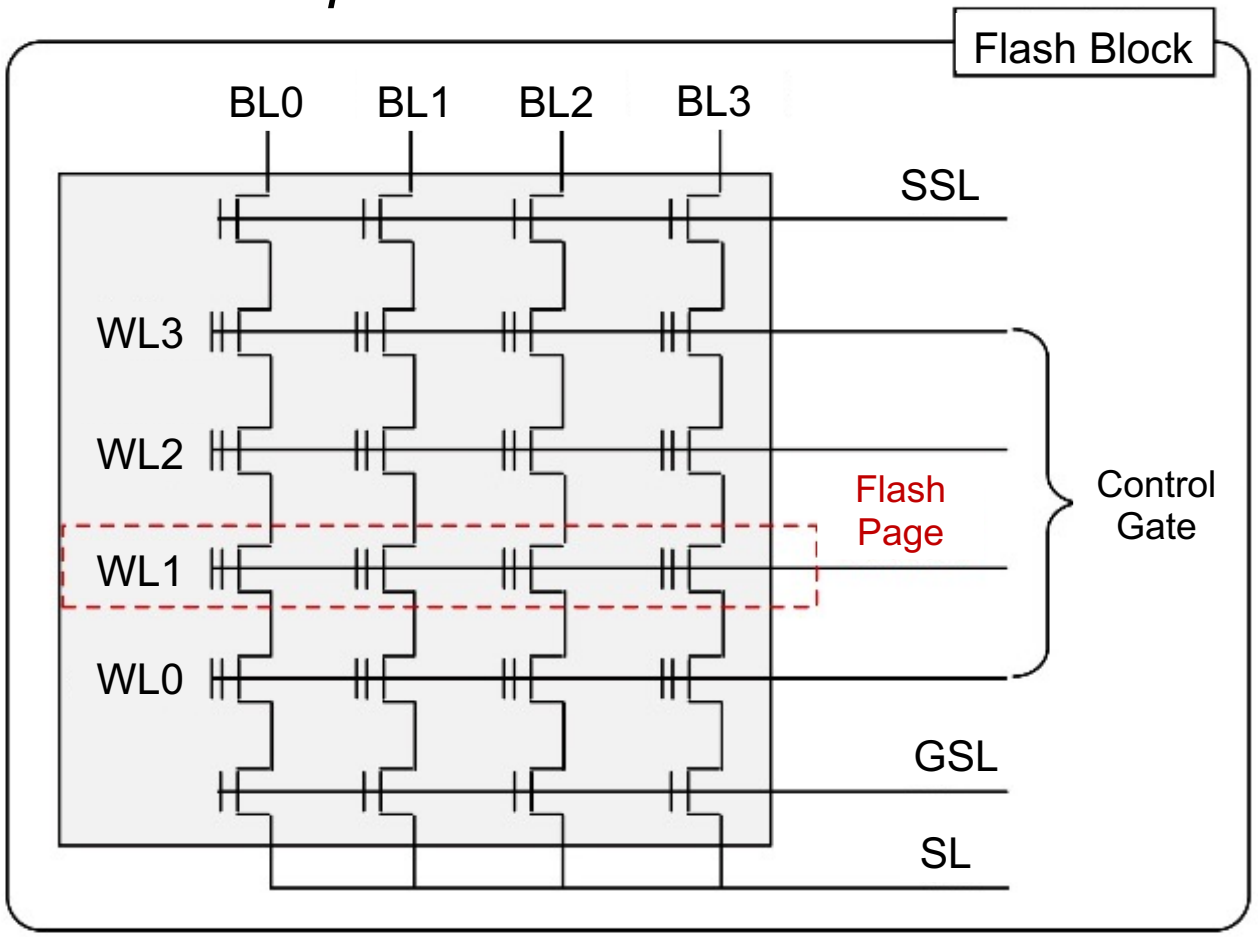


Fig. 1 Architecture of flash memory array

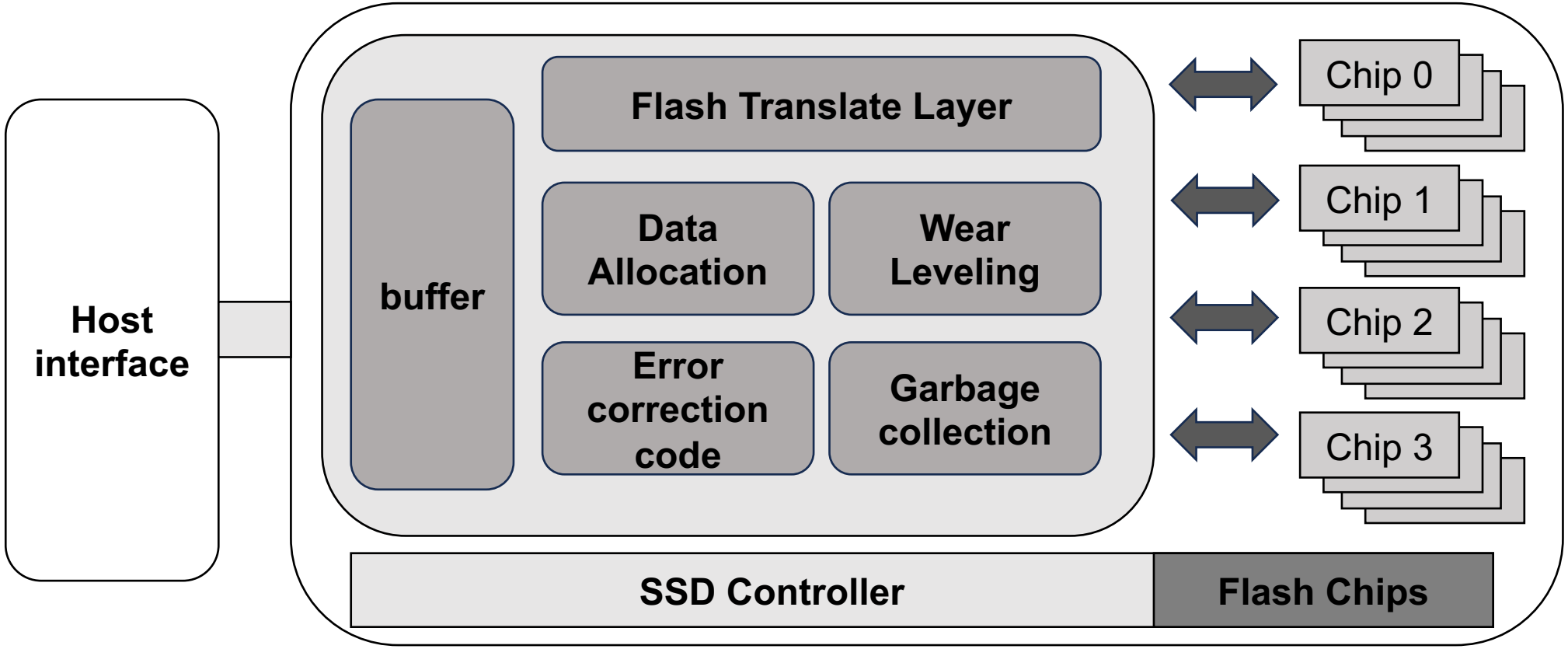
Write: the minimal unit of write is page

Read: the minimal unit of read is page

Erase: the minimal unit of erase is block

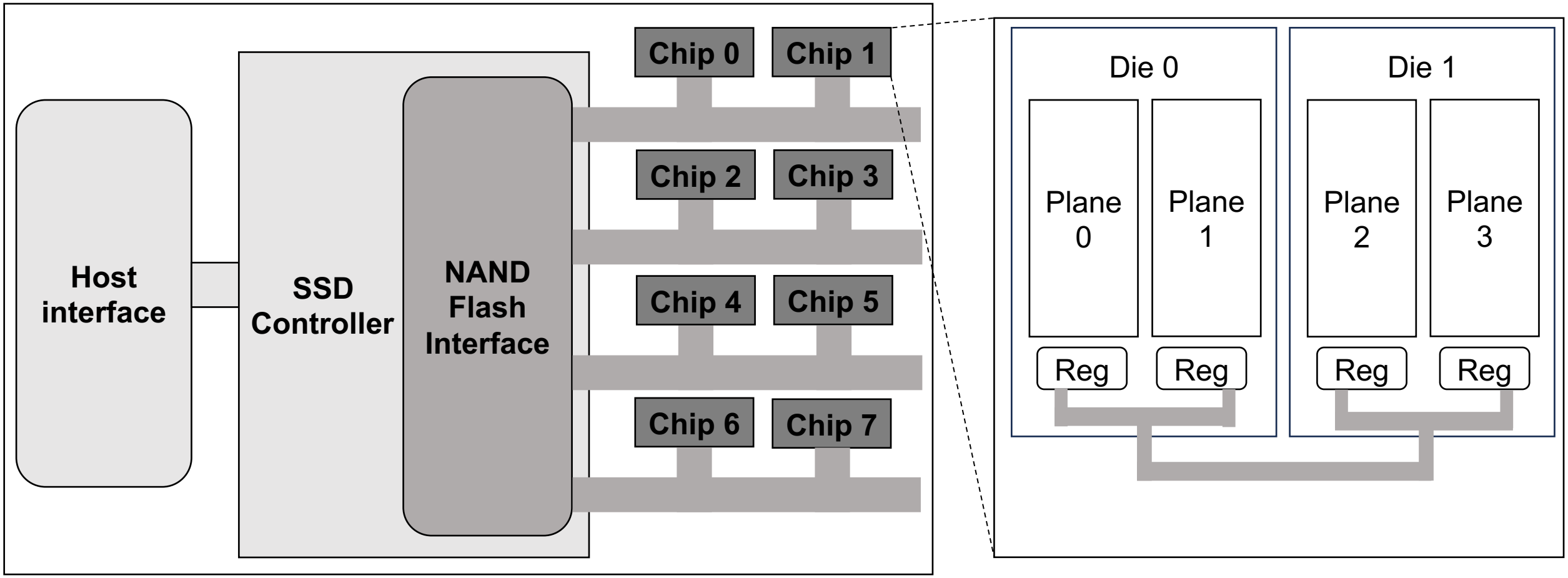
1. Basic characteristics of flash SSD

1.4. Architecture of SSD



1. Basic characteristics of flash SSD

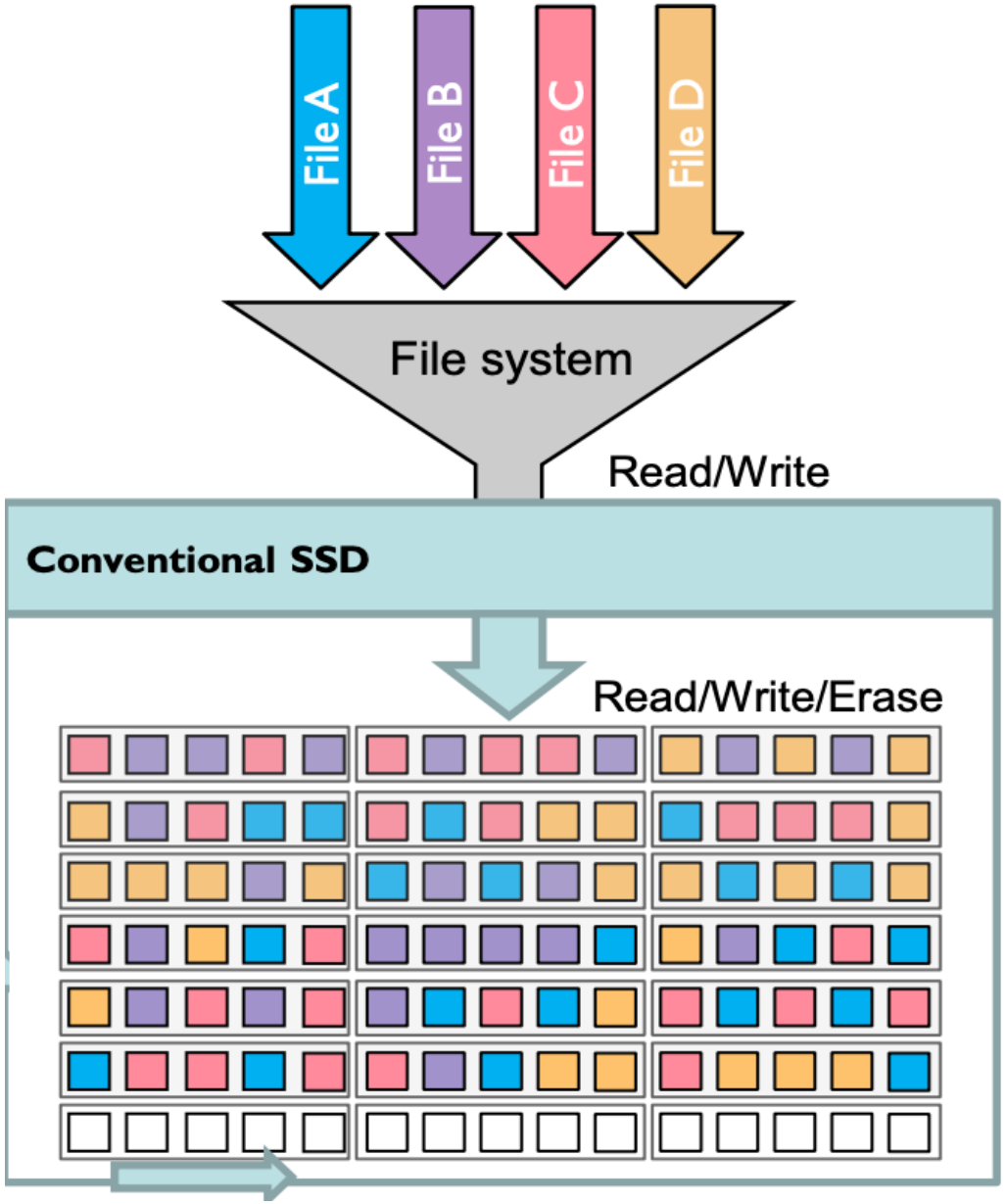
1.5. Parallel Architecture of SSD



2. Conventional SSD

2.1. Data placement

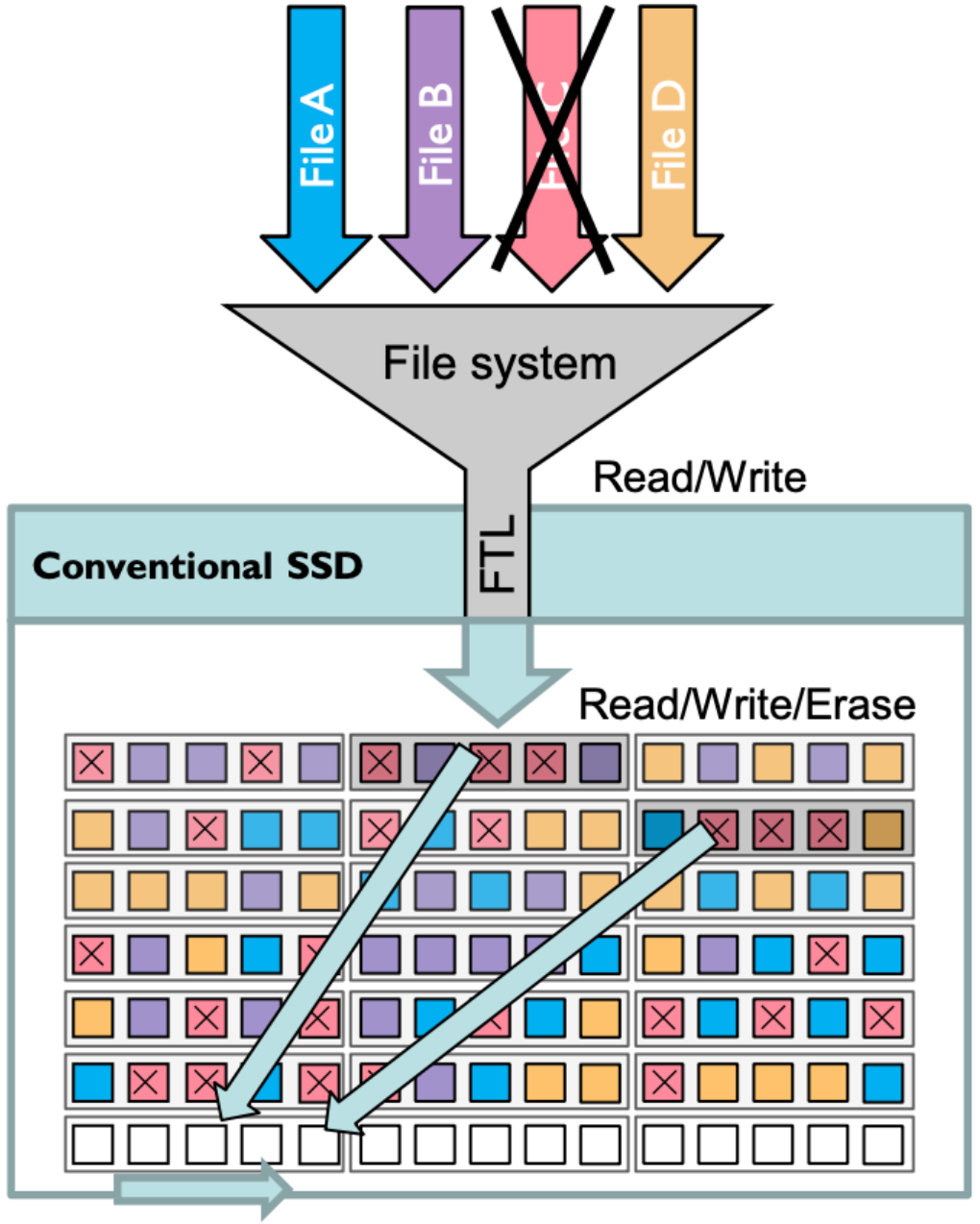
- SSD's FTL is responsible for mapping logical blocks to the physical addresses of media
- Files are written to media as they are served and can't be separated by the drive
- As a result, files with different lifetimes is stored within the same erase unit



2. Conventional SSD

2.2. Garbage collection

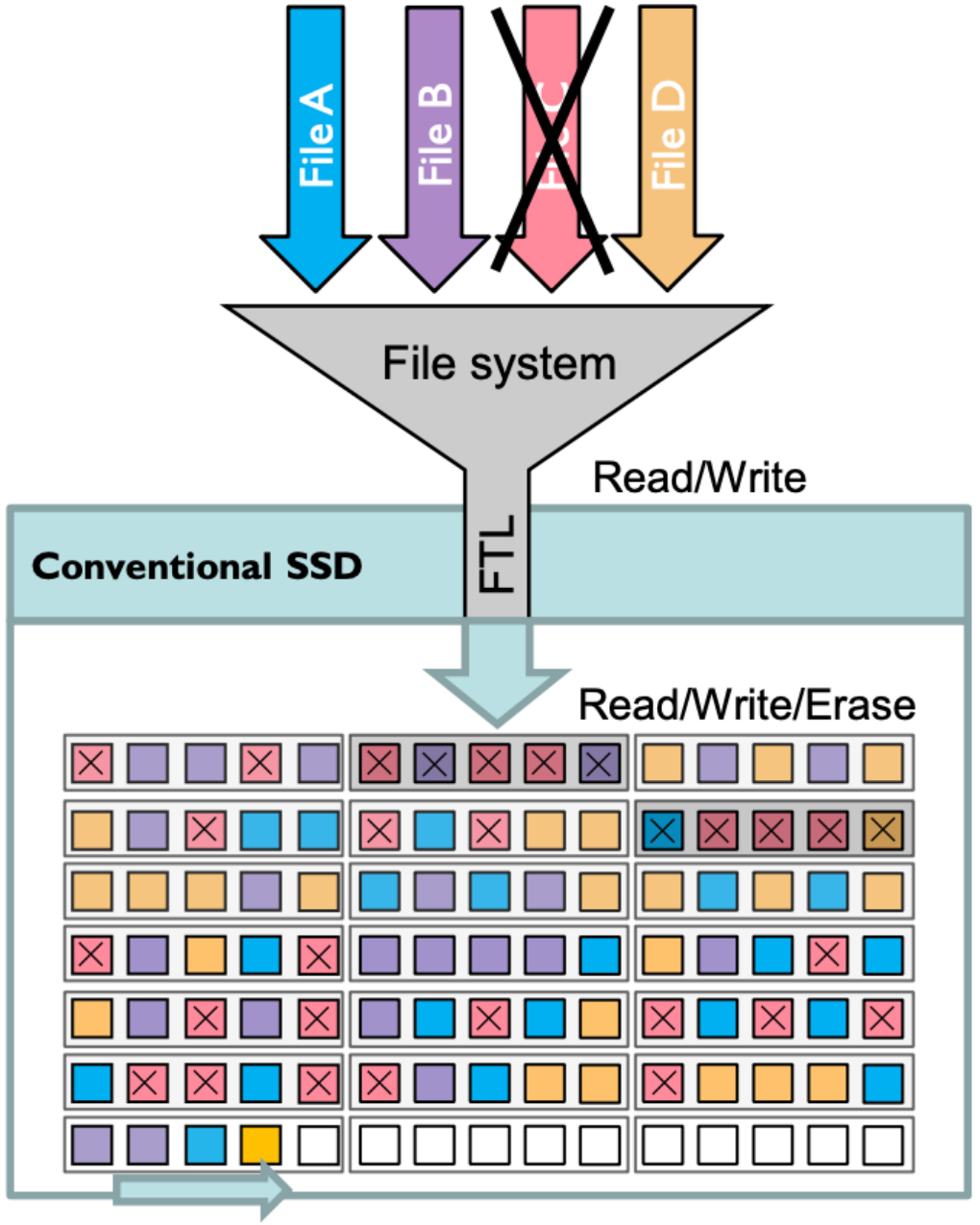
- When a file gets deleted, the space occupied can’t be reused without erasing the whole erase unit
- A garbage collection process is required to reclaim writeable media



2. Conventional SSD

2.2. Garbage collection

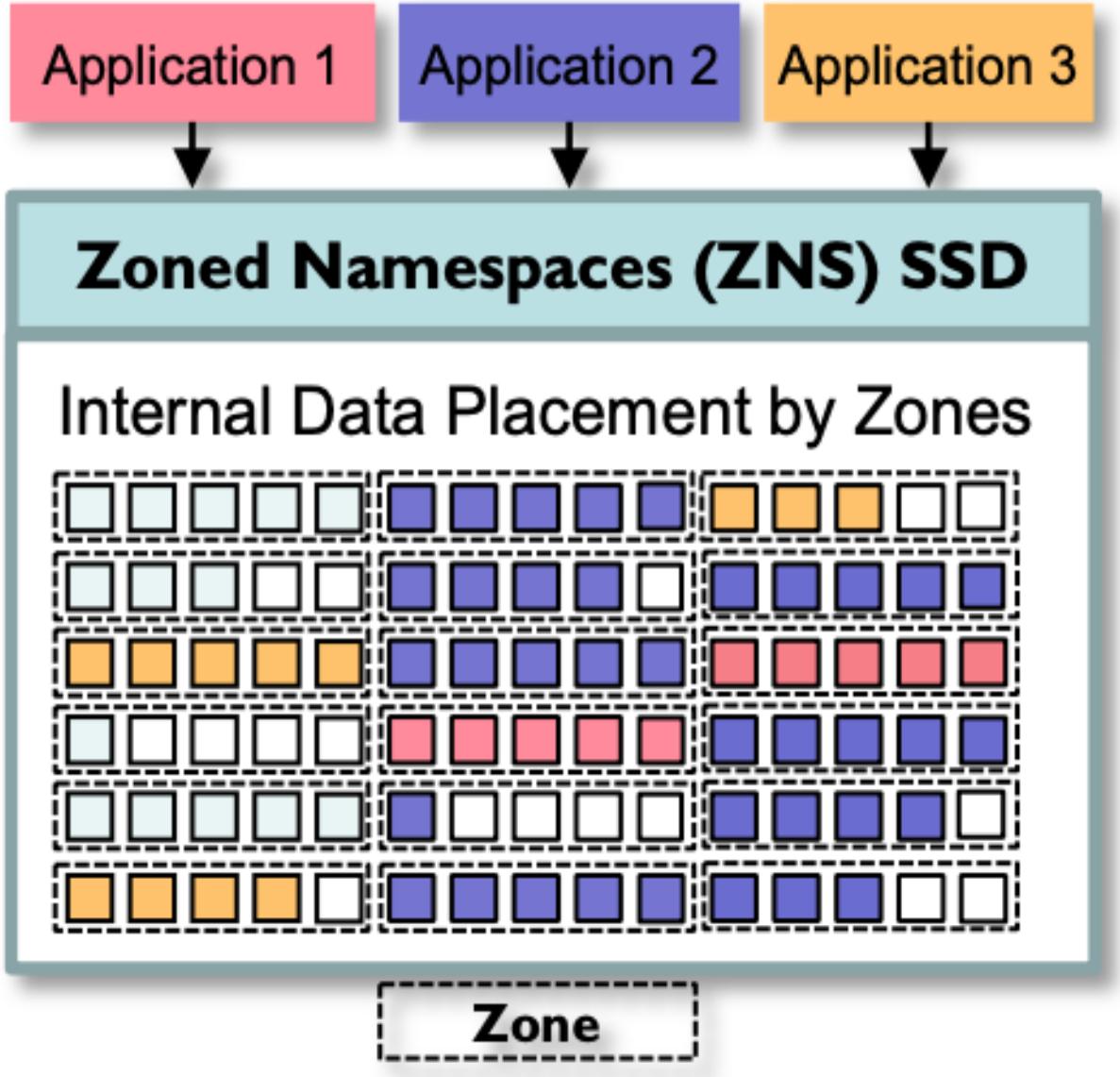
- This results a Write Amplification Factor, which is the number of times data is rewritten within a device after host has written the data once.
- Increased WAF leads to:
  - **Lower Write Throughput**
  - **Higher Read Latencies**
  - **Increase Device Cost**
  - **Decrease Device Lifetime**



3. ZNS SSD

3.1. Collaborate on Data Placement

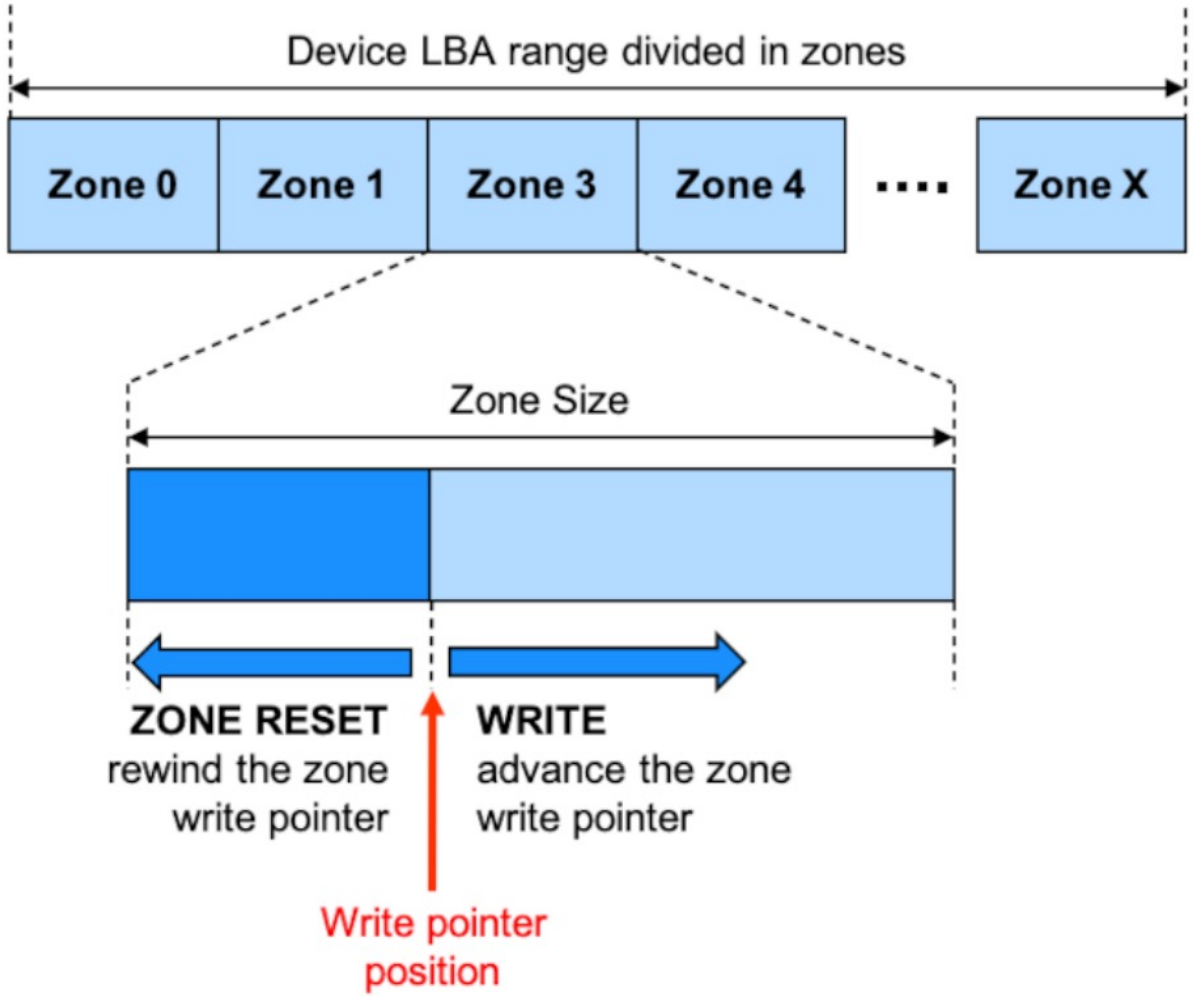
- Enable the host and storage device to collaborate on data placement, such that the host naturally write sequentially with a device erase unit.
- Eliminates the device-side garbage collection caused when writing
  - **Higher Write Throughput**
  - **Lower Read Latencies**
  - **Decrease Device Cost**
  - **Increase Device Lifetime**



3. ZNS SSD

3.2. Principle of Operation

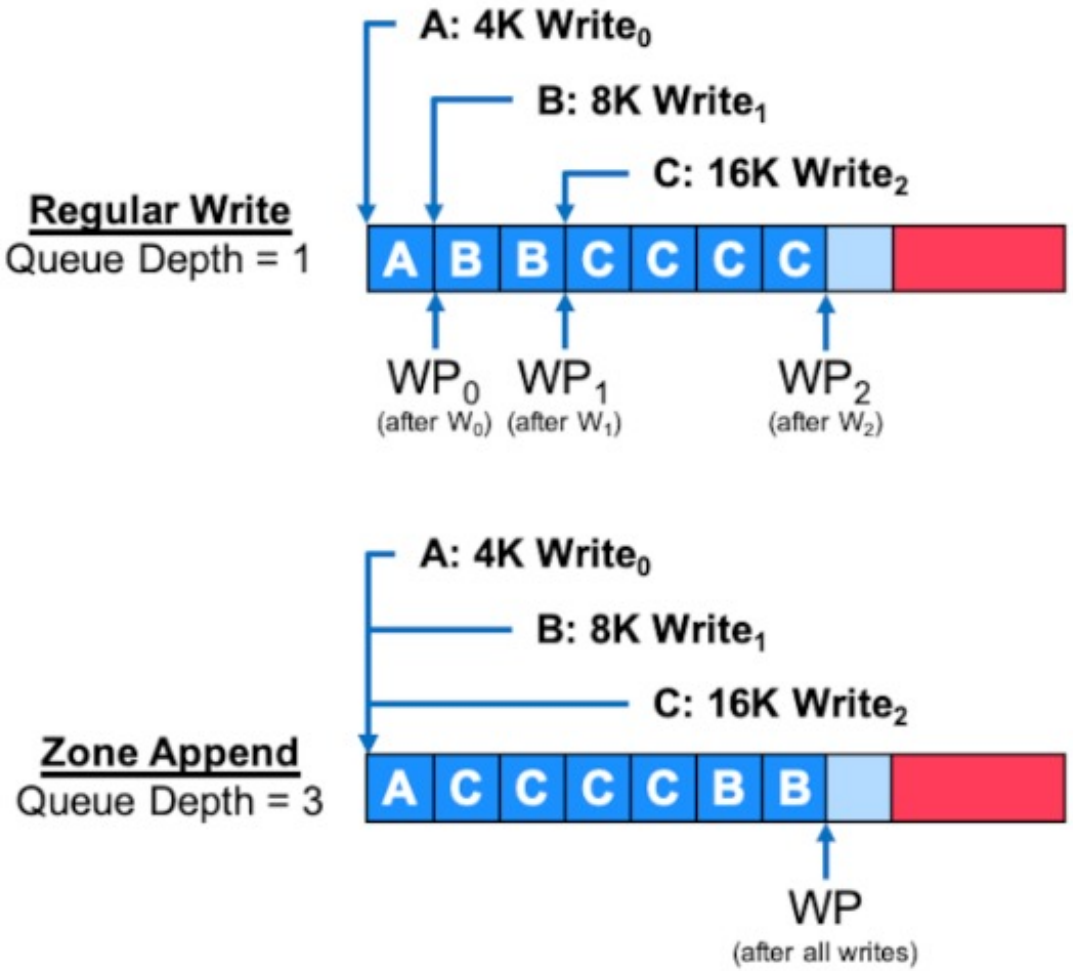
- **Sequential write constraint:** zones in ZNS SSD must be written sequentially
- **Write pointer:** keep track of the position of the next write
- **Zone reset:** zone must first be erased using this command before being overwritten



3. ZNS SSD

3.3. Zone Append

- All write commands directed at a sequential zone must not be reordered before they are received by the device and executed.
- Host software can avoid write errors by limiting the number of write commands outstanding per zone to one.

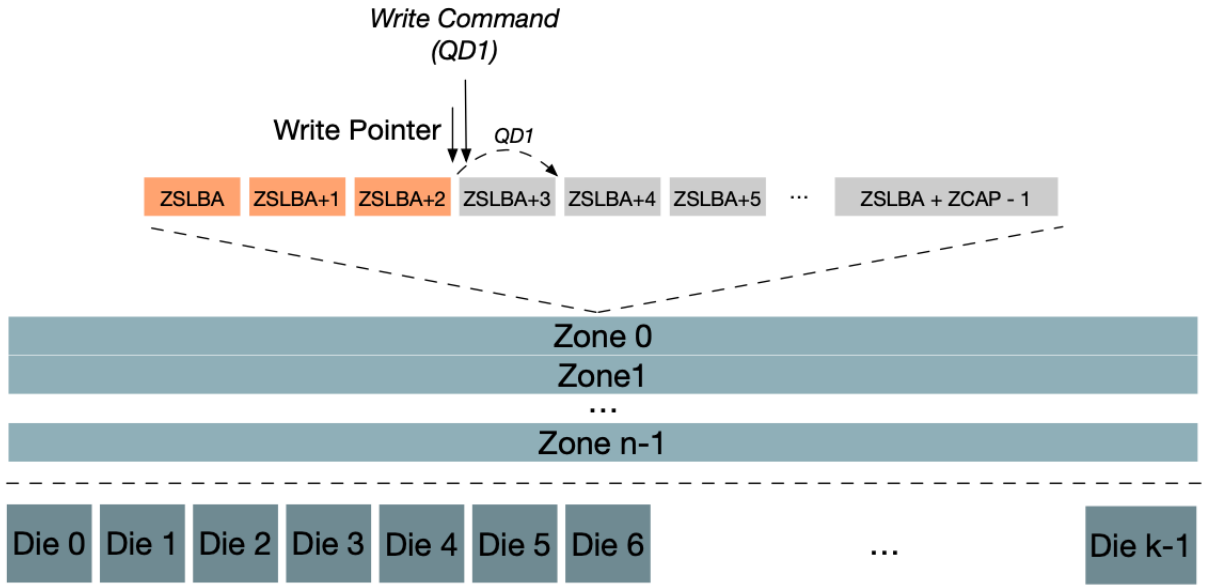


3. ZNS SSD

3.3. Zone Append

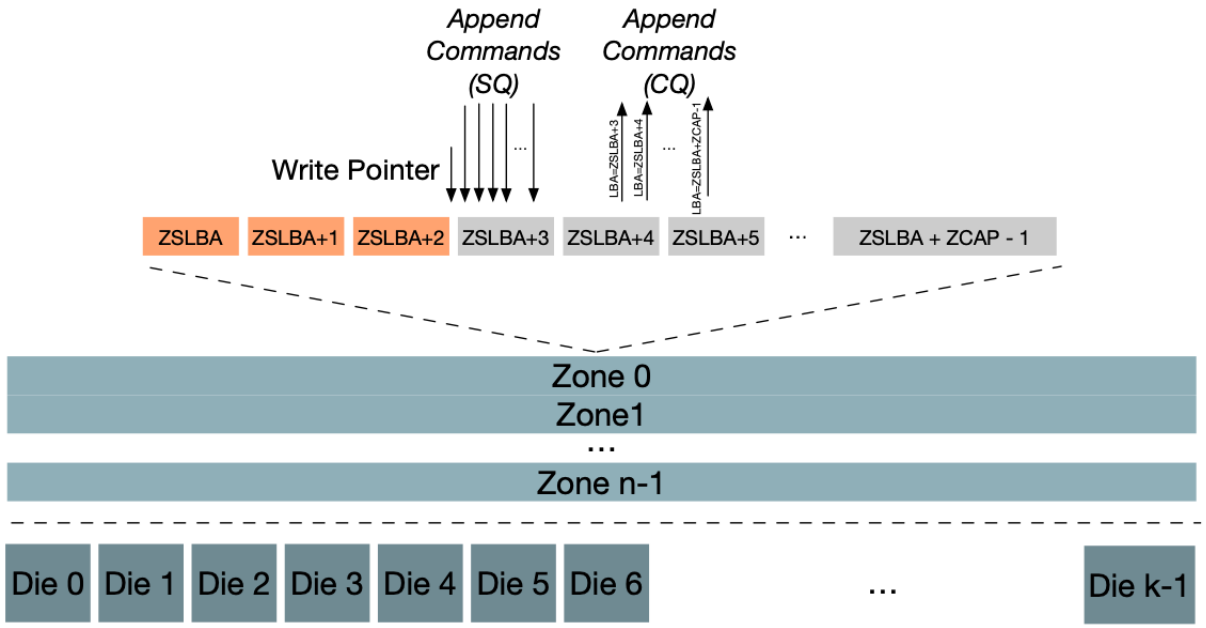
Write Path

- Required QD=1
  - Reminder: No ordering guarantees in NVMe



Append Path

- QD<=X / X = #LBAs in zone



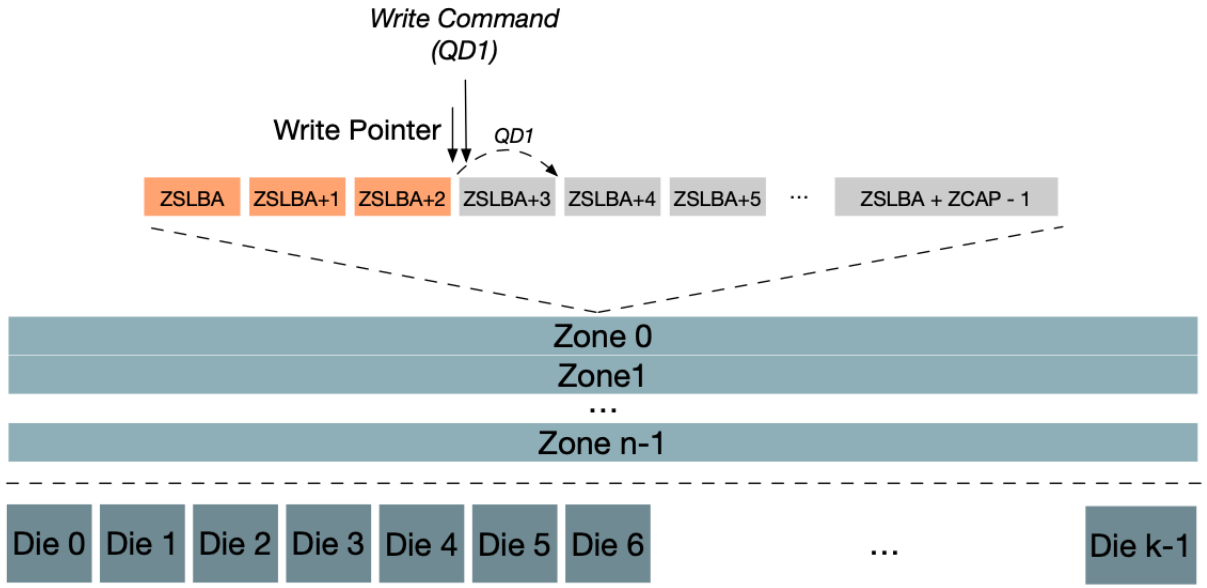
[1] Y. Zhang, L. P. Arulraj, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. De-indirection for flash-based SSDs with nameless writes. FAST, 2012.

3. ZNS SSD

3.3. Zone Append

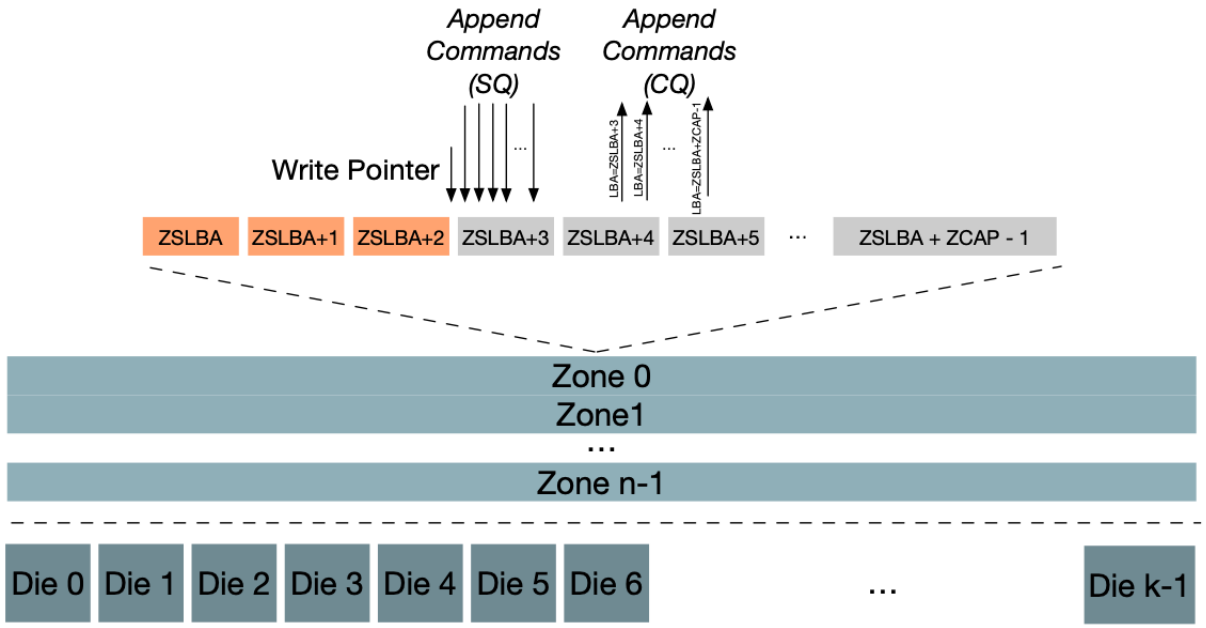
Write Path

- Required QD=1
  - Reminder: No ordering guarantees in NVMe



Append Path

- QD<=X / X = #LBAs in zone



[1] Y. Zhang, L. P. Arulraj, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. De-indirection for flash-based SSDs with nameless writes. FAST, 2012.

3. ZNS SSD

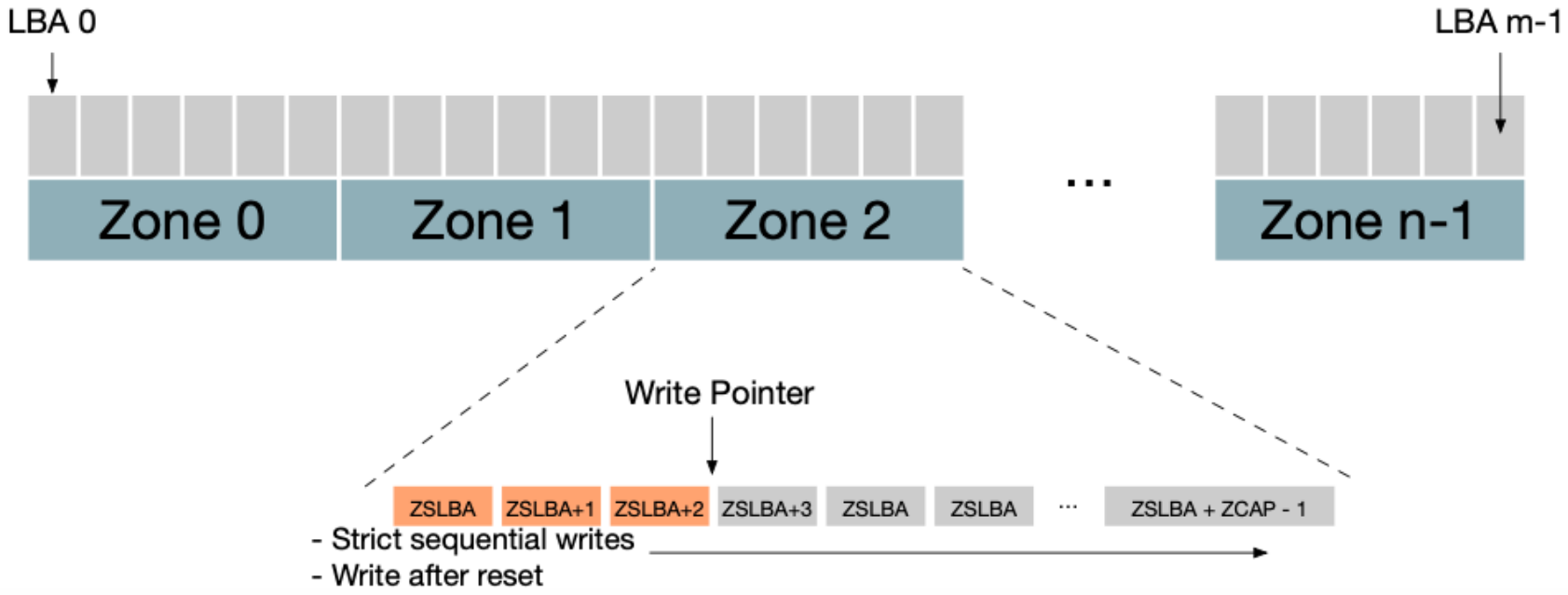
3.4. More to be explored

- **Host Responsibilities**
- **Zone Random Write Area**
- **Simple Copy**
- **… …**

4. Related literatures

4.1. Optimizing Data Migration for Garbage Collection in ZNS SSDs. DATE 2023

Problem to be solved: In ZNS SSD, programs are allowed to continuously write to different zones in an efficient data placement method. However, invalid data in a zone can only be released when the entire zone is erased, which results in low space utilization.



4. Related literatures

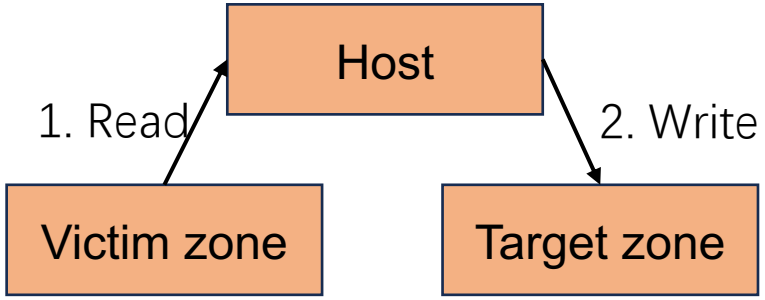
4.1. Optimizing Data Migration for Garbage Collection in ZNS SSDs. DATE 2023

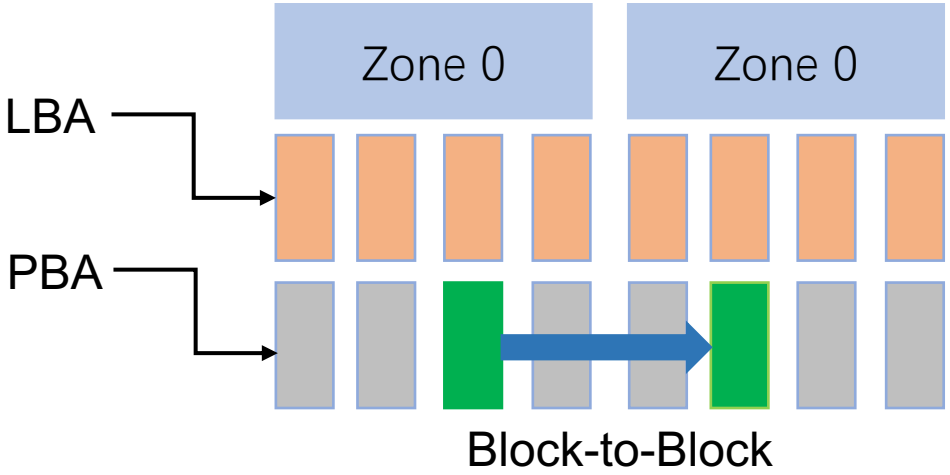
Previous solution: focus on decreasing data migration times

Low efficiency:

I. Data migration in ZNS SSD results in unnecessary **host-to-host overhead**

II. Overhead of **block-to-block** data migration between two zones is also very high, which will lead to huge **rewriting overhead**.





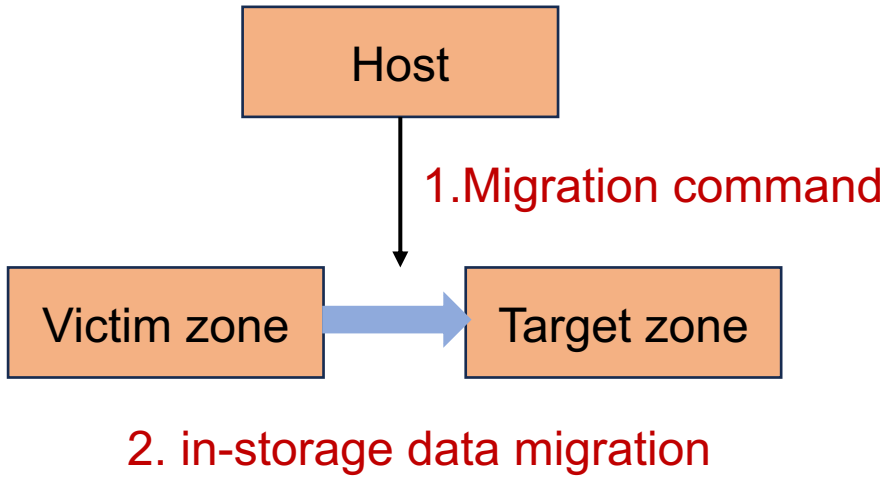
4. Related literatures

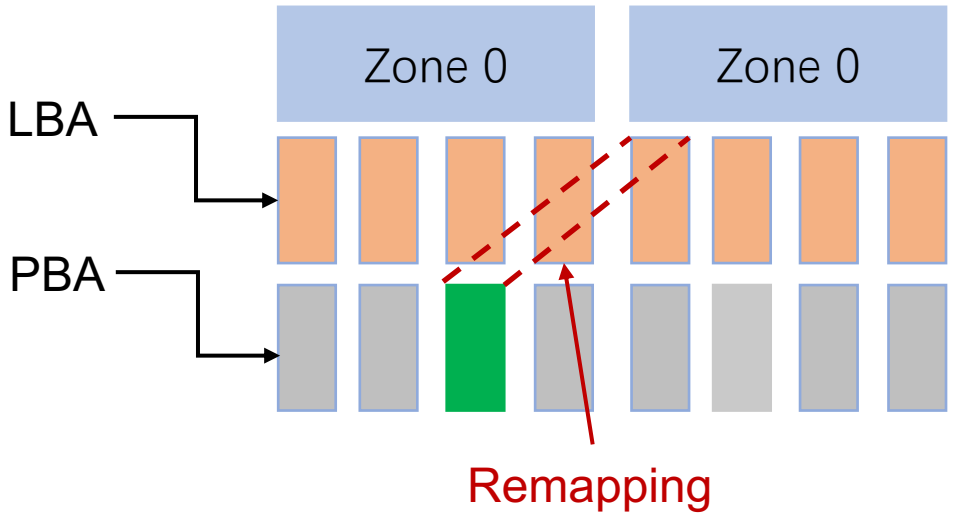
4.1. Optimizing Data Migration for Garbage Collection in ZNS SSDs. DATE 2023

An ideal solution:

I. In-storage data migration

II. Remapping blocks with valid data in victim zone to target zone





4. Related literatures

4.1. Optimizing Data Migration for Garbage Collection in ZNS SSDs. DATE 2023

Approaches:

I. In-storage data migration

Zone_MD(Src, Dst, Size)

- **Src:** starting logic address of valid data in victim zone
- **Dst:** write pointer address in target zone
- **Size:** size of migration data

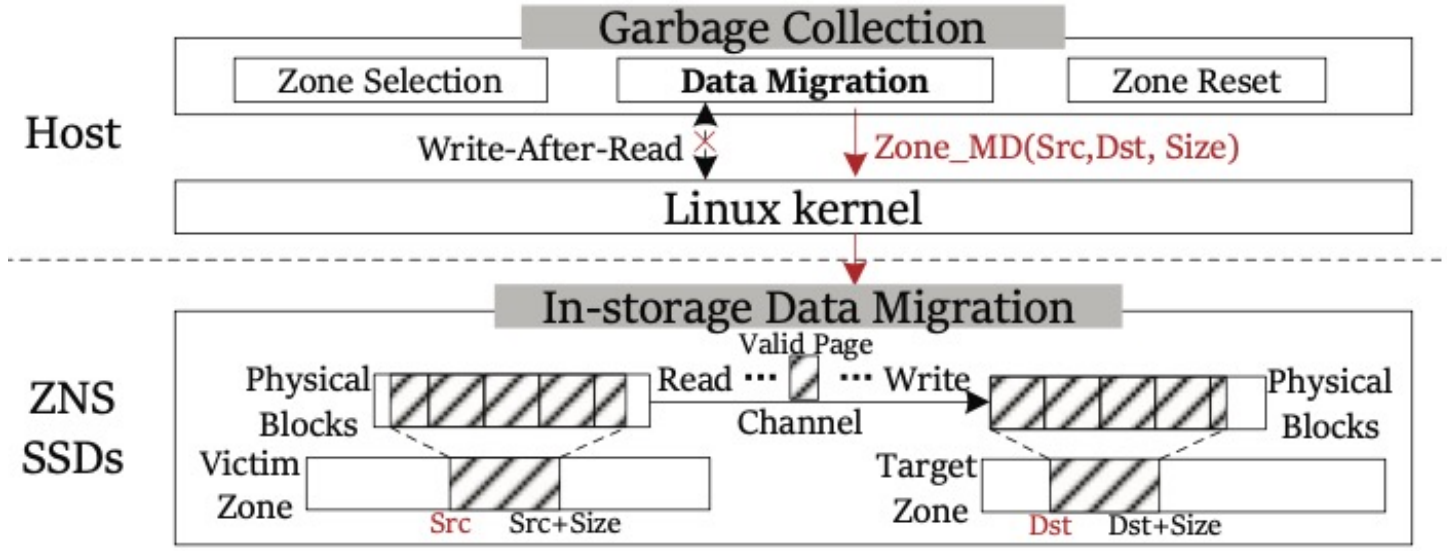


Fig. 1. The method of in-storage data migration.

4. Related literatures

4.1. Optimizing Data Migration for Garbage Collection in ZNS SSDs. DATE 2023

Approaches:

I. In-storage data migration

Pros: the entire in-storage data migration is done with the help of the ***internal channel*** of the storage, avoiding end-to-end transmission.

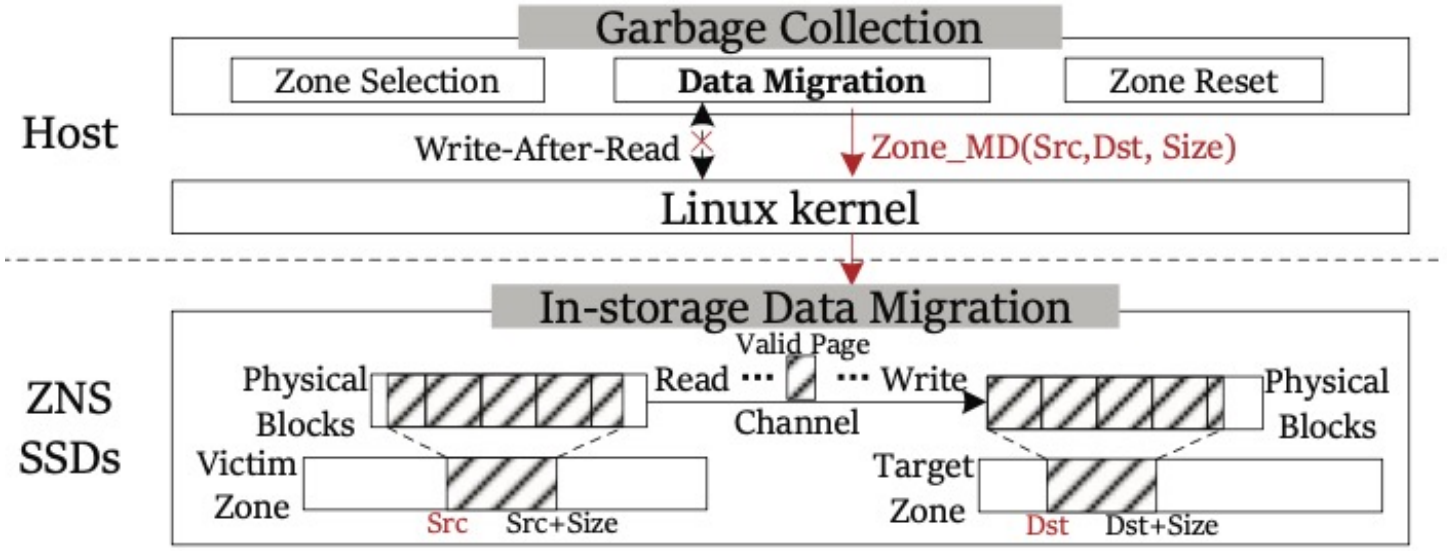


Fig. 1. The method of in-storage data migration.

4. Related literatures

4.1. Optimizing Data Migration for Garbage Collection in ZNS SSDs. DATE 2023

Approaches:

I. In-storage data migration

II. Remapping strategy

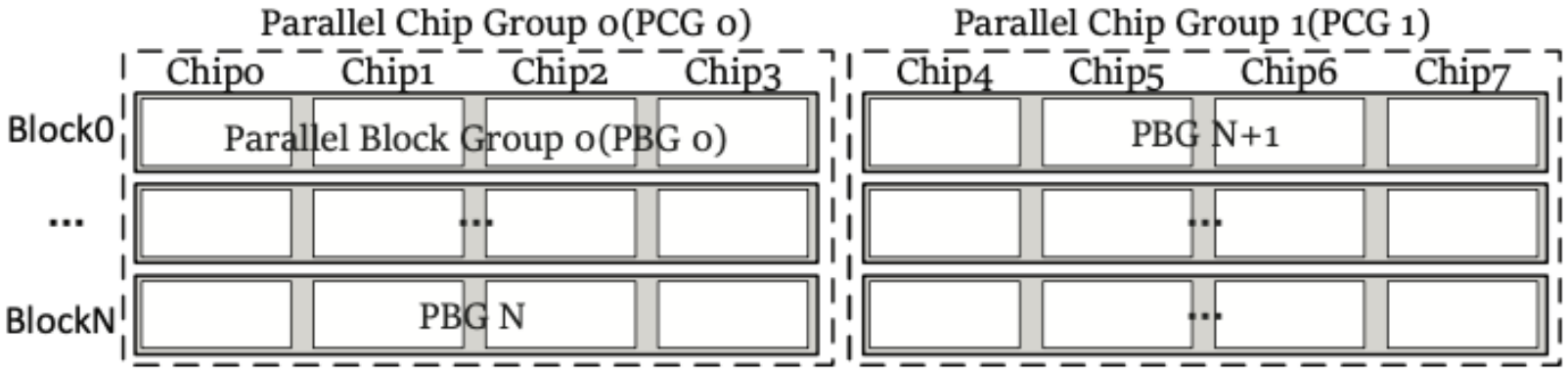


Fig. 2. A dynamic zone mapping method.

- **PCG:** facilitate parallel chips performance
- **PBG:** the blocks with the same offset in the PCG are further divided into multiple PBGs
- We take the parallel block group (PBG) as the smallest remapping unit and ensure the parallelism within the region.
- When a region is allocated, we dynamically map PBGs to the region.

4. Related literatures

4.1. Optimizing Data Migration for Garbage Collection in ZNS SSDs. DATE 2023

Approaches:

I. In-storage data migration

II. Remapping strategy

- **V1:** The entire PBG is legal data.
- **V2:** Part of the PBG is legal data.
- **T1:** All parts of the PBG are written.
- **T2:** Only part of the PBG will be written.

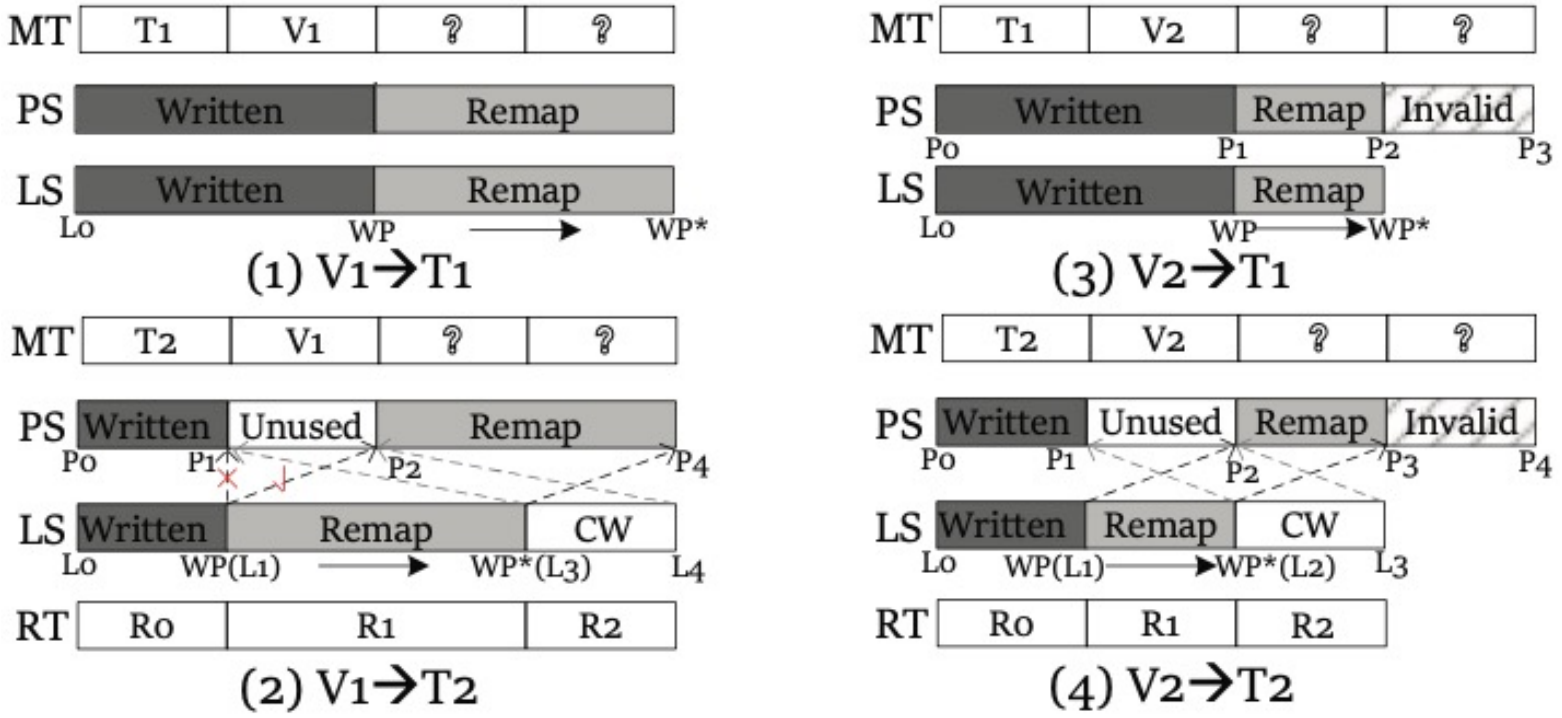


Fig. 3. The diagram of address remapping.

4. Related literatures

4.1. Optimizing Data Migration for Garbage Collection in ZNS SSDs. DATE 2023

Approaches:

I. In-storage data migration

II. Remapping strategy

① **V1->T1**

② **V1->T2**

③ **V2->T1**

④ **V2->T2**

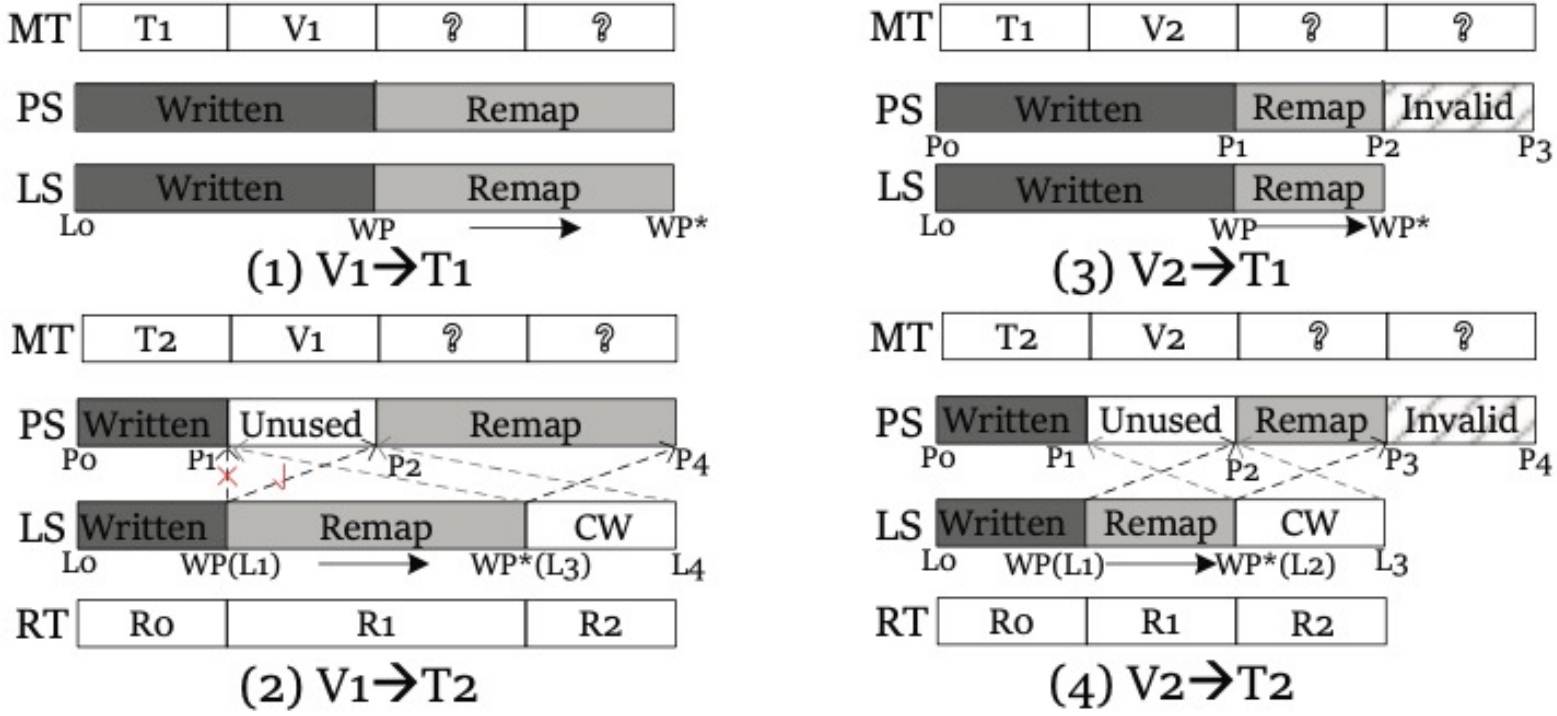


Fig. 3. The diagram of address remapping.

4. Related literatures

4.1. Optimizing Data Migration for Garbage Collection in ZNS SSDs. DATE 2023

Approaches:

I. In-storage data migration

II. Remapping strategy

Therefore, we use the current space utilization (**su**) of ZNS SSD as the benchmark.

If the percentage of valid data in V2 is greater than su, we perform remapping.

Otherwise we directly rewrite all valid data in V2

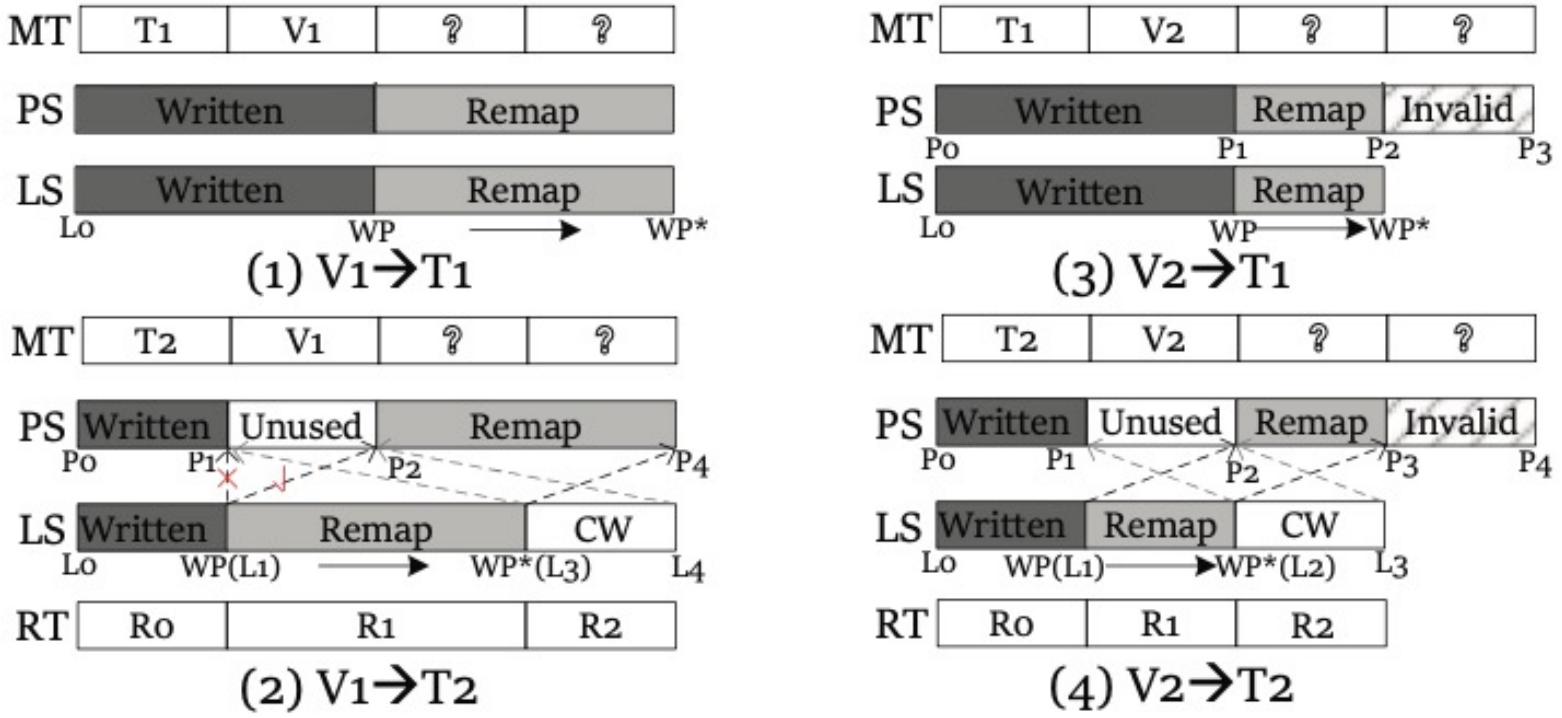


Fig. 3. The diagram of address remapping.

4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

Problem to be solved: The traditional garbage collection scheme, which is based on entire zones, incurs long latencies due to the large size of a zone.

4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

Motivation:

I. Choose areas with low utilization rates as areas for garbage collection.

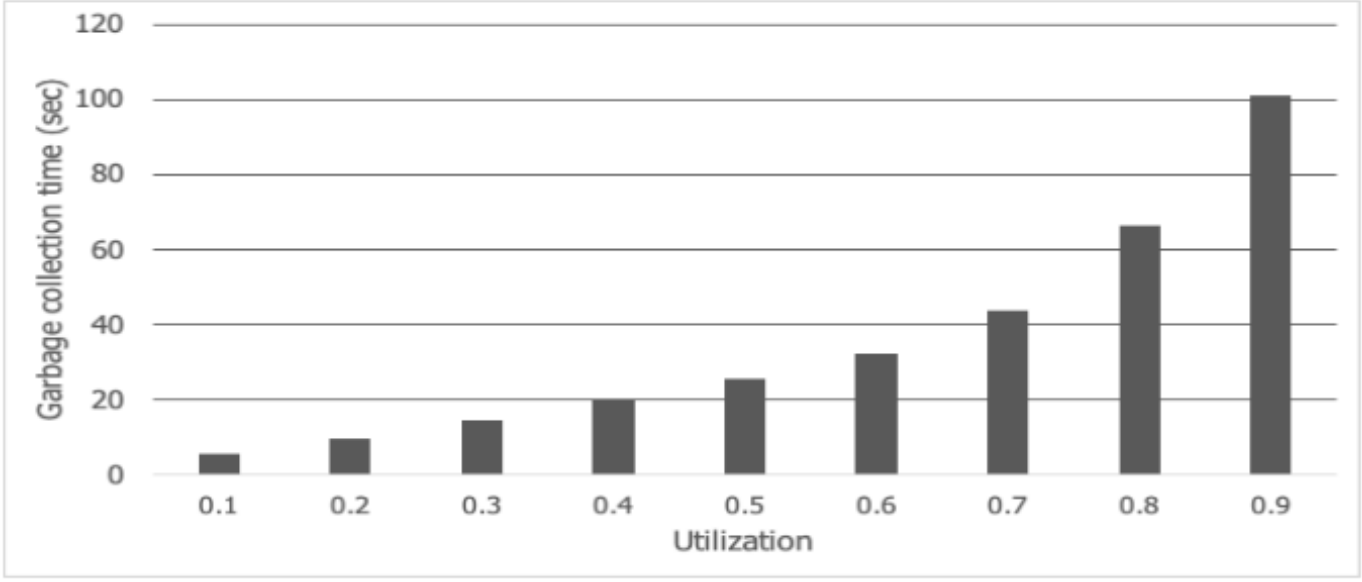


Figure 1: Zone garbage collection overhead under various utilizations of a zone

4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

Motivation:

I. Choose areas with low utilization rates as areas for garbage collection.

II. When reading a region, the time-consuming mode of reading blocks individually is higher than the mode of group reading of blocks.

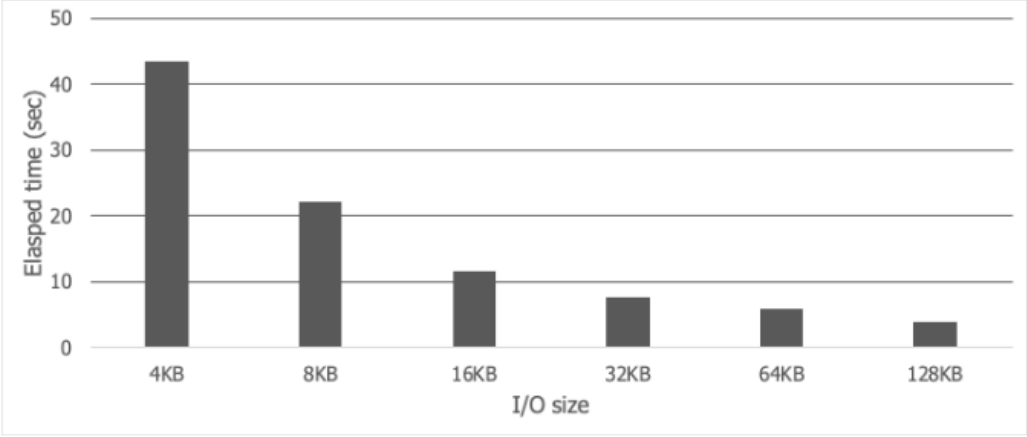


Figure 2: Access blocks individually vs. in a group manner

4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

Previous solution: Basic_ZGC

I. Choose an area with the least utilization.

II. Read the valid data blocks in this area and write them to the new area.

III. Finally, reset the selected block through the reset command.

4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

Approaches: LSM_ZGC:

- LSM_ZGC divides the area into multiple segments and manages the information of these segments, respectively.
- Garbage collection is performed in the style of LSM, reading all data in candidate regions, identifying cold data, merging them into one region, while merging the remaining data into another region.

4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

Approaches: LSM_ZGC:

- Using a segment-based (rather than zone-based) approach is conducive to hot and cold data segmentation, and the strategy can be implemented in a pipeline manner.
- Reading all valid and invalid data in the segment can reduce garbage collection overhead by taking advantage of the internal parallelism in the region.
- Putting hot and cold data into different regions increases your chances of finding less utilized regions.

4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

Approaches: LSM_ZGC:

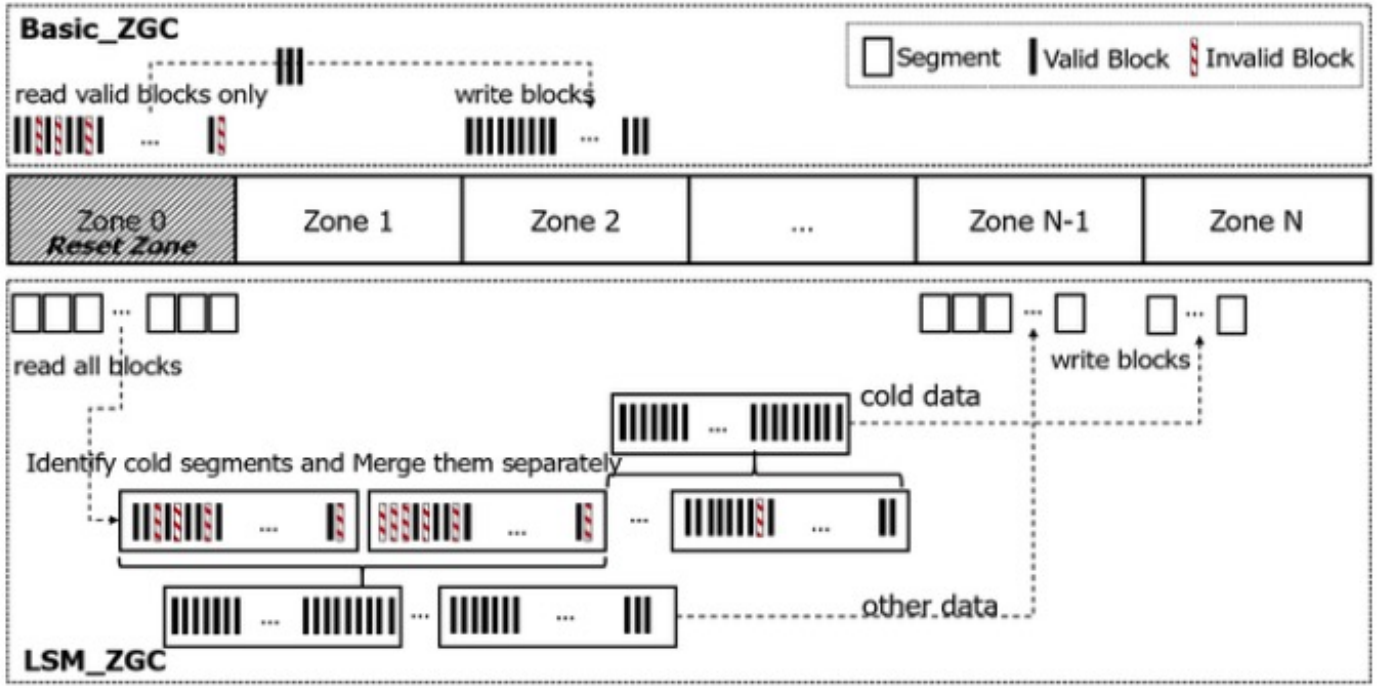


Figure 3: Design concept of LSM_ZGC in comparison with Basic_ZGC

4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

Approaches: LSM_ZGC:

Pros:

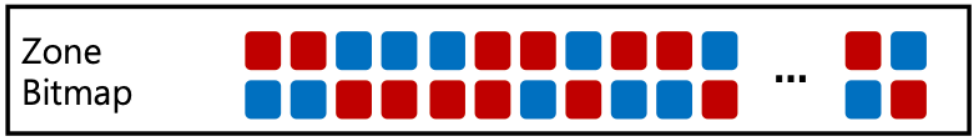
- Garbage collection for a zone will be implemented on the segment
- During the garbage collection phase, it uses 128KB IO, not only reading valid blocks, but also reading invalid blocks.
- LSM_ZGC identifies cold data and merges the cold data into a separate area.

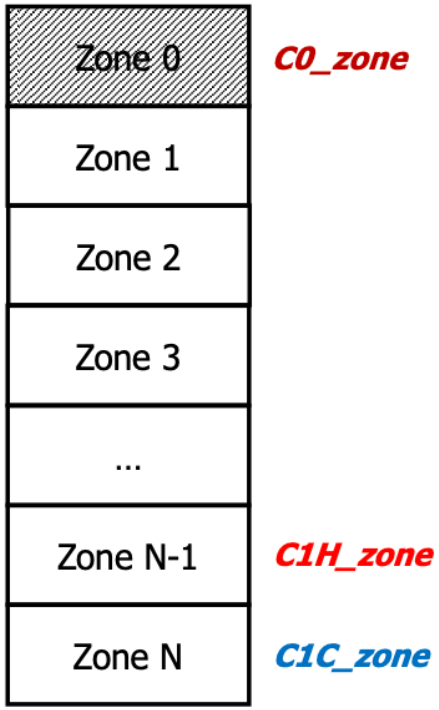
4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

Approaches: LSM_ZGC:

LSM(Log Structured Merge) Zone GC





C0_zone

Step 1.
Select a candidate zone (or zones, Greedy or CB)

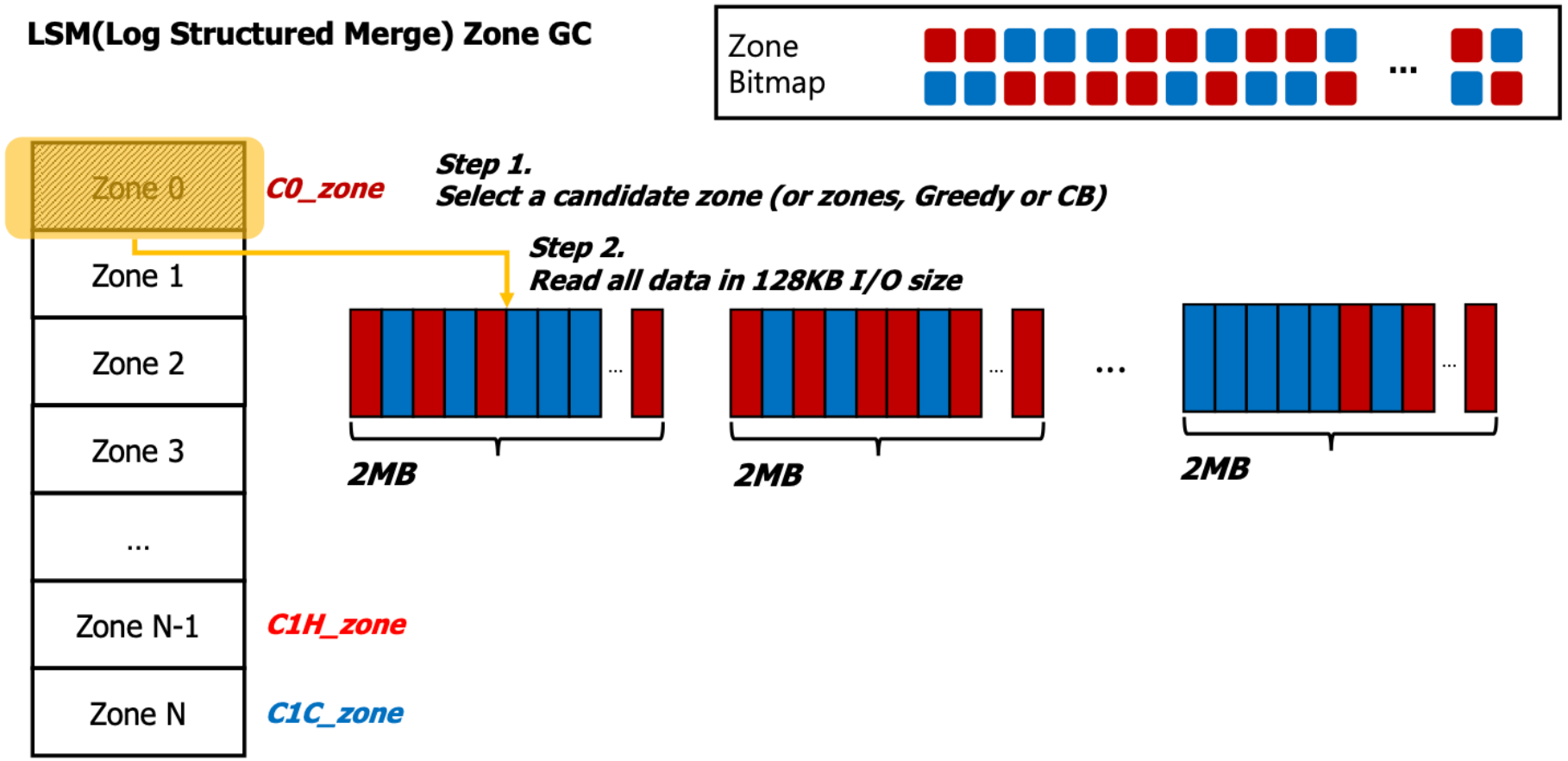
C1H_zone

C1C_zone

4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

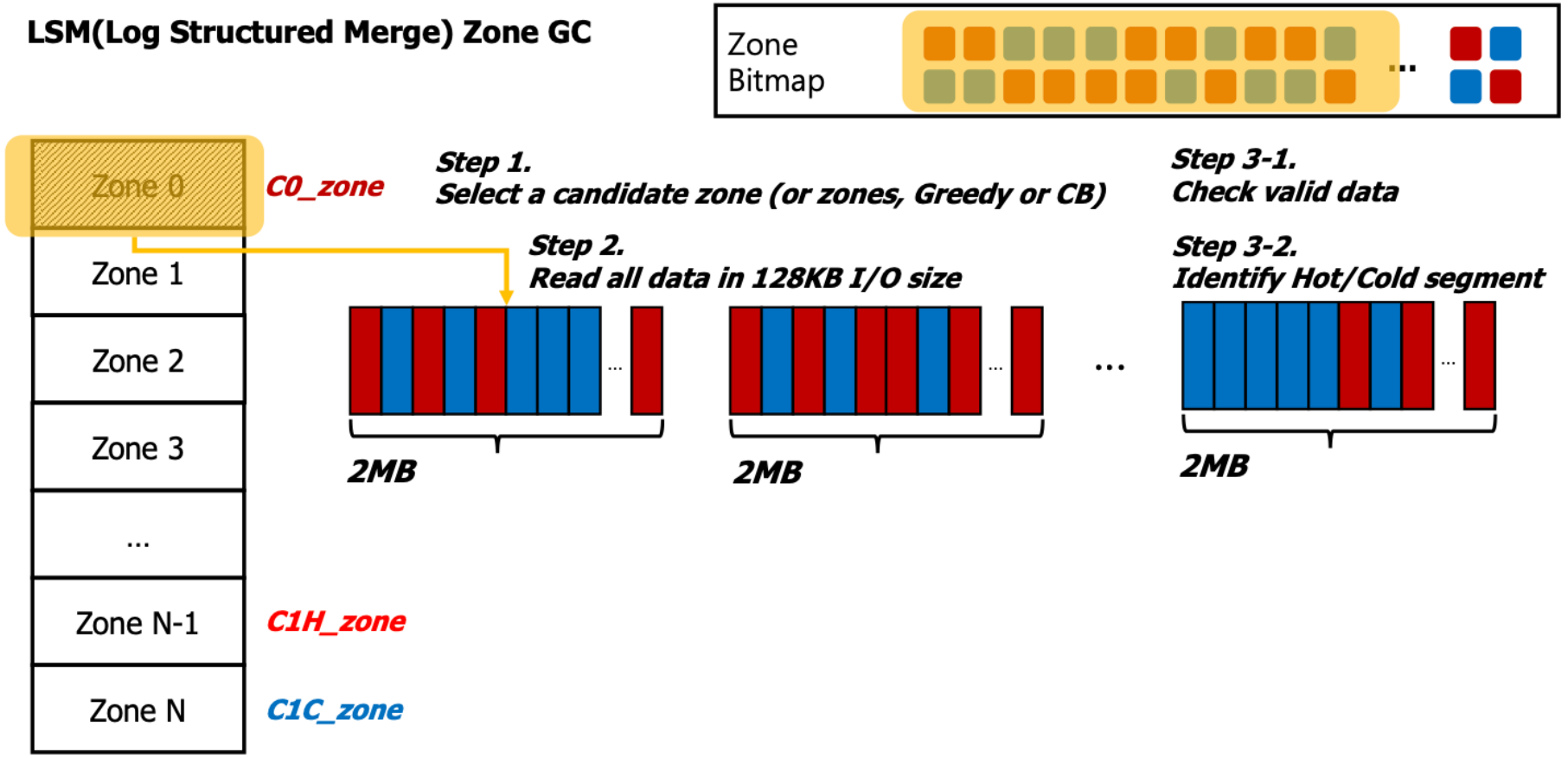
Approaches: LSM_ZGC:



4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

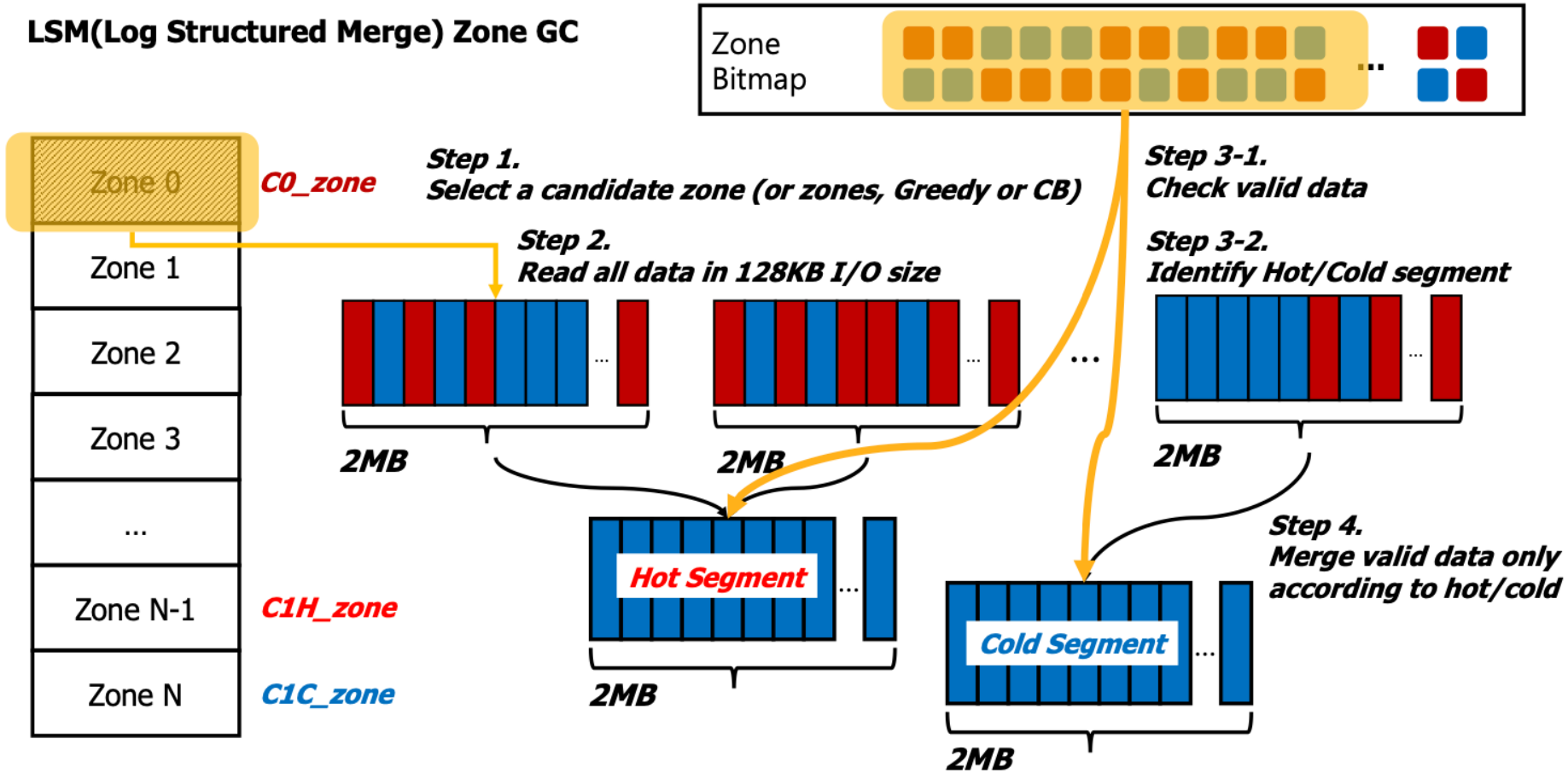
Approaches: LSM_ZGC:



4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

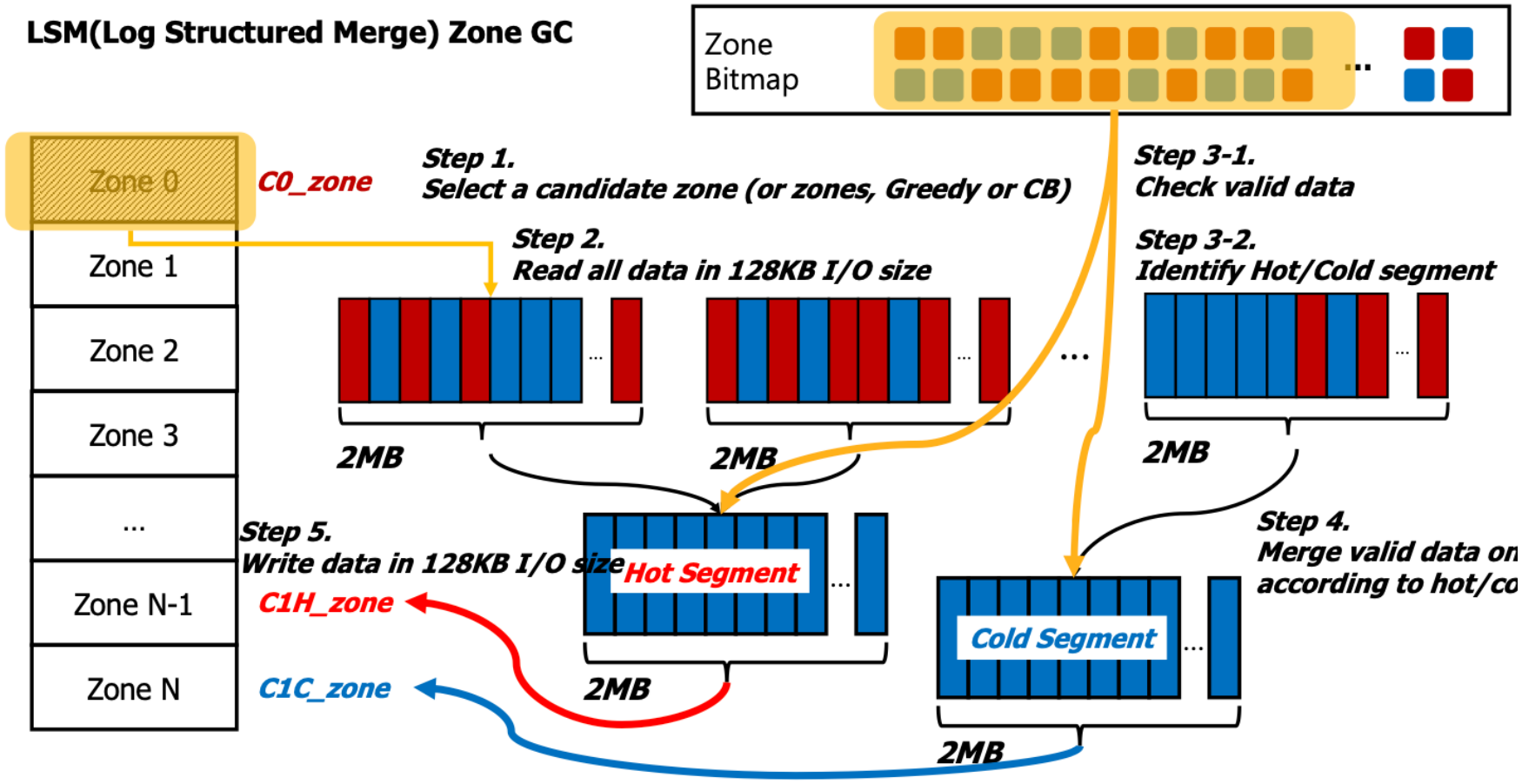
Approaches: LSM_ZGC:



4. Related literatures

4.2. A New LSM-style Garbage Collection Scheme for ZNS SSDs. HotStorage' 20

Approaches: LSM_ZGC:



4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

Problem to be solved: ZNS SSD is vulnerable to fragmentation due to its improper updating and multi-threaded writing behavior of the application.

Lead to: The I/O request will be divided (to read different areas, so one I/O is divided into multiple I/Os).

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

Previous solution cons: Choose an area with the least utilization.

I. Since most Defragmentation tools are designed for traditional SSDs, they cannot handle physical fragmentation (this is because traditional SSDs do not expose physical addresses to the host).

II. Traditional defragmentation tools, such as defrag.f2fs, are very expensive when defragmenting.

III. For users unfamiliar with system programming, using a tool like defrag.f2fs can be difficult.

IV. Flash memory devices have a limited number of erases, and excessive data migration will reduce the life of ZNS SSD.

V. Defragmentation consumes a lot of time and reduces the performance of front-end applications (due to additional I/O operations)

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

Definition of physical and logical fragment:

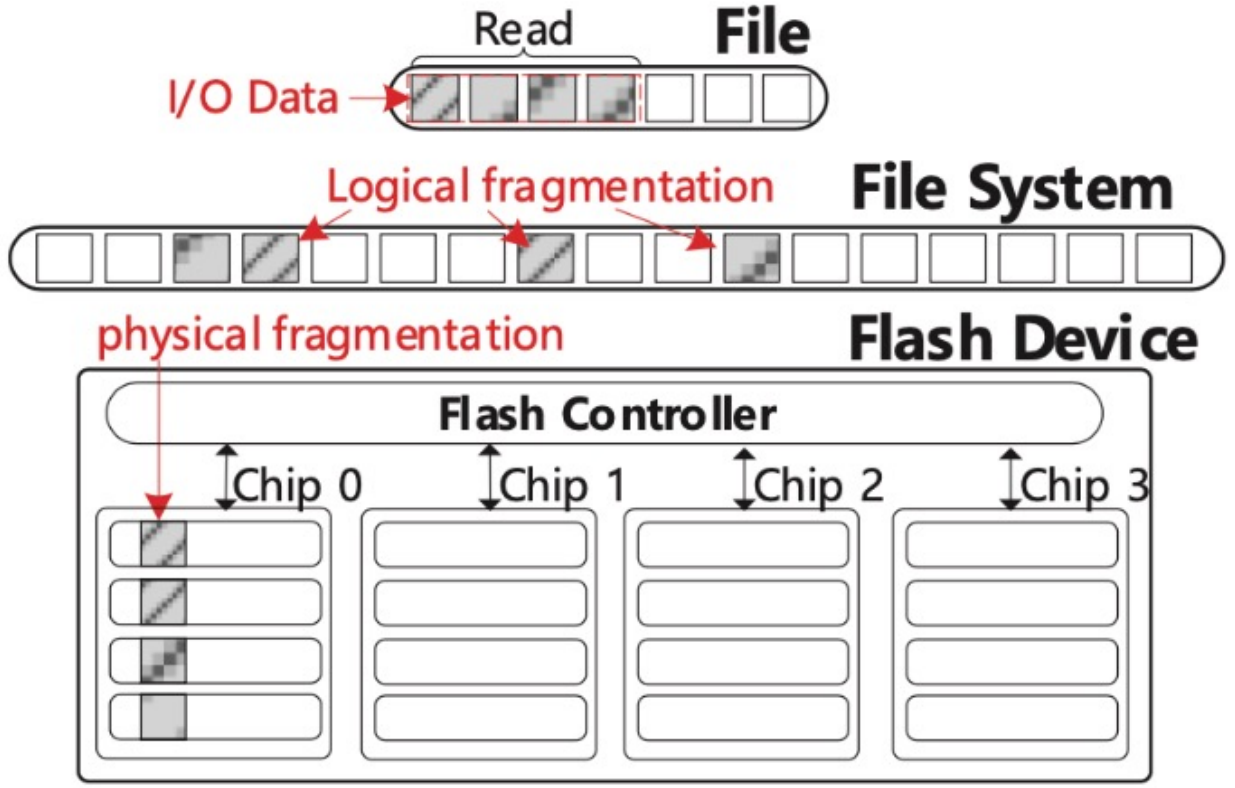


Fig. 2. The logical and physical fragmentation for the I/O data.

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

Definitions:

- **DoLF:** The number of logical fragments in an I/O range.
- **DoPF:** We measure the degree of physical fragmentation of I/O requests by the even distribution of I/O-wide data across flash parallel units.

$$DoPF = \frac{\sum_{i=1}^{L}(N_i - \frac{M}{L})^2}{L}$$

- M: The number of data blocks accessed by the I/O request
- L: Number of parallel chips of flash memory
- N_i : N represents the number of data blocks located in i parallel unit (parallel chip)

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

How to find suitable fragments to process?

- Defragmenting cold data that is poorly fragmented or rarely accessed provides little performance gain.
- I/O data defragmentation priority (IODP):

$$IODP = (\alpha \cdot DoLF + \beta \cdot DoPF) \cdot readcount$$

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

The overhead of traditional SSD fragmentation processing

- Memory Page Allocation
- Data read
- Contiguous Space Allocation
- Data Write
- Metadata update

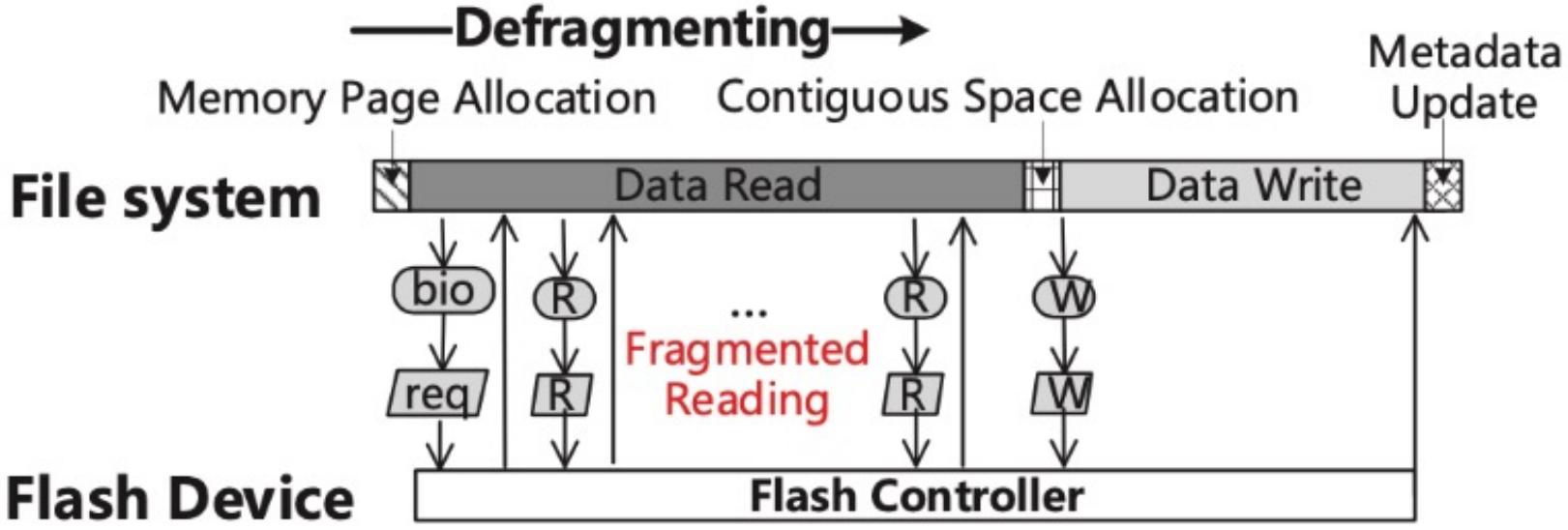


Fig. 5. The overall process of conventional defragmenter.

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

Approaches: InDeF

- Minimize the amount of migrated data used for defragmentation to reduce write traffic to the underlying device.
- Take advantage of the ZNS SSD's internal parallelism to reduce defragmentation time.

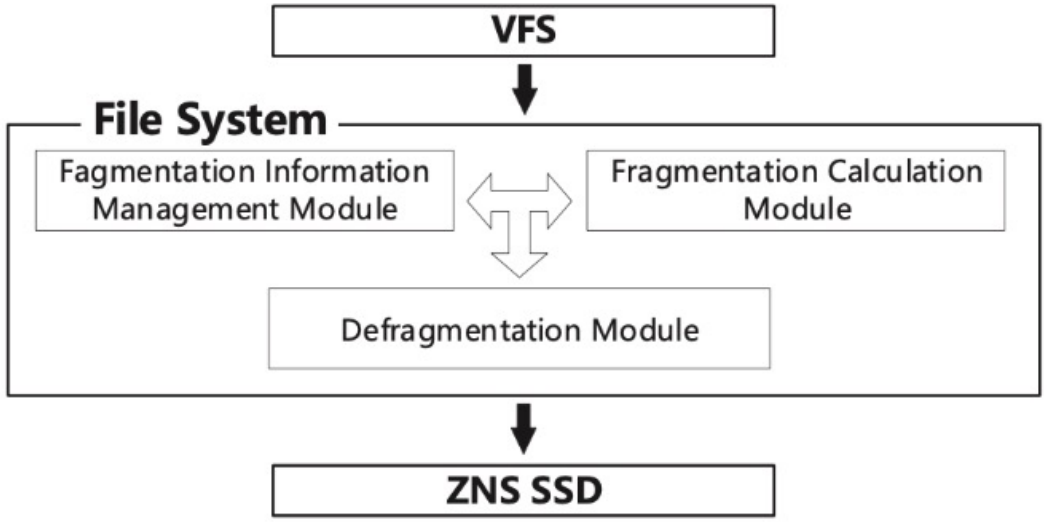


Fig. 6. The overview of InDeF.

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

Fragmentation Information Management Module:

- I/O Monitor. Determines the range of file blocks through the inode of the file accessed by I/O.
- I/O Information Management: sort and merge file list
- Fragmentation Information Management: manage IODP value

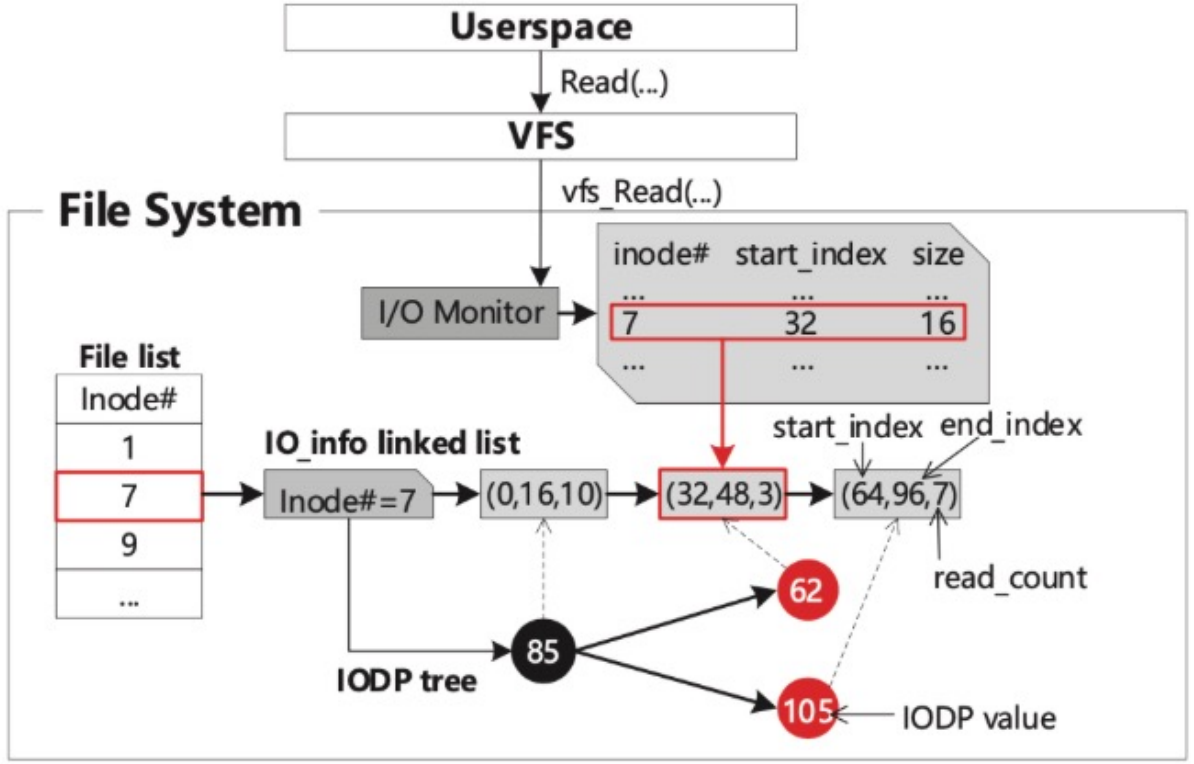


Fig. 7. The fagmentation information management module of InDeF.

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

Fragmentation Calculation Module:

- Zone Mapping in ZNS SSD

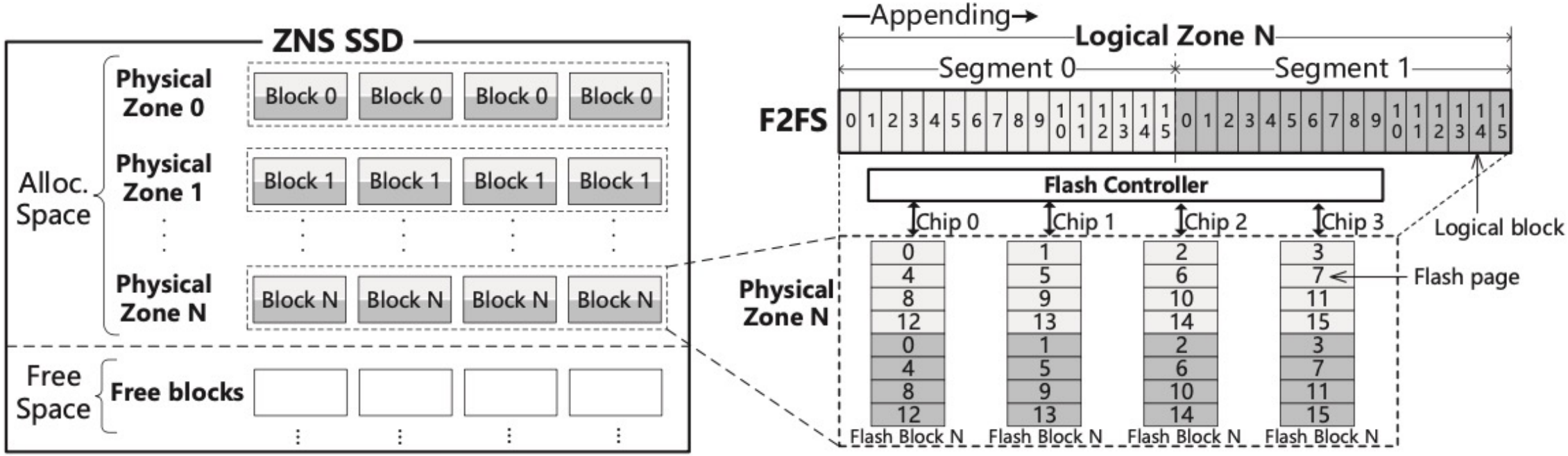


Fig. 8. An example of zone and flash blocks mapping.

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

Fragmentation Calculation Module:

- Zone Mapping in ZNS SSD
- Calculating the degree of Fragmentation.

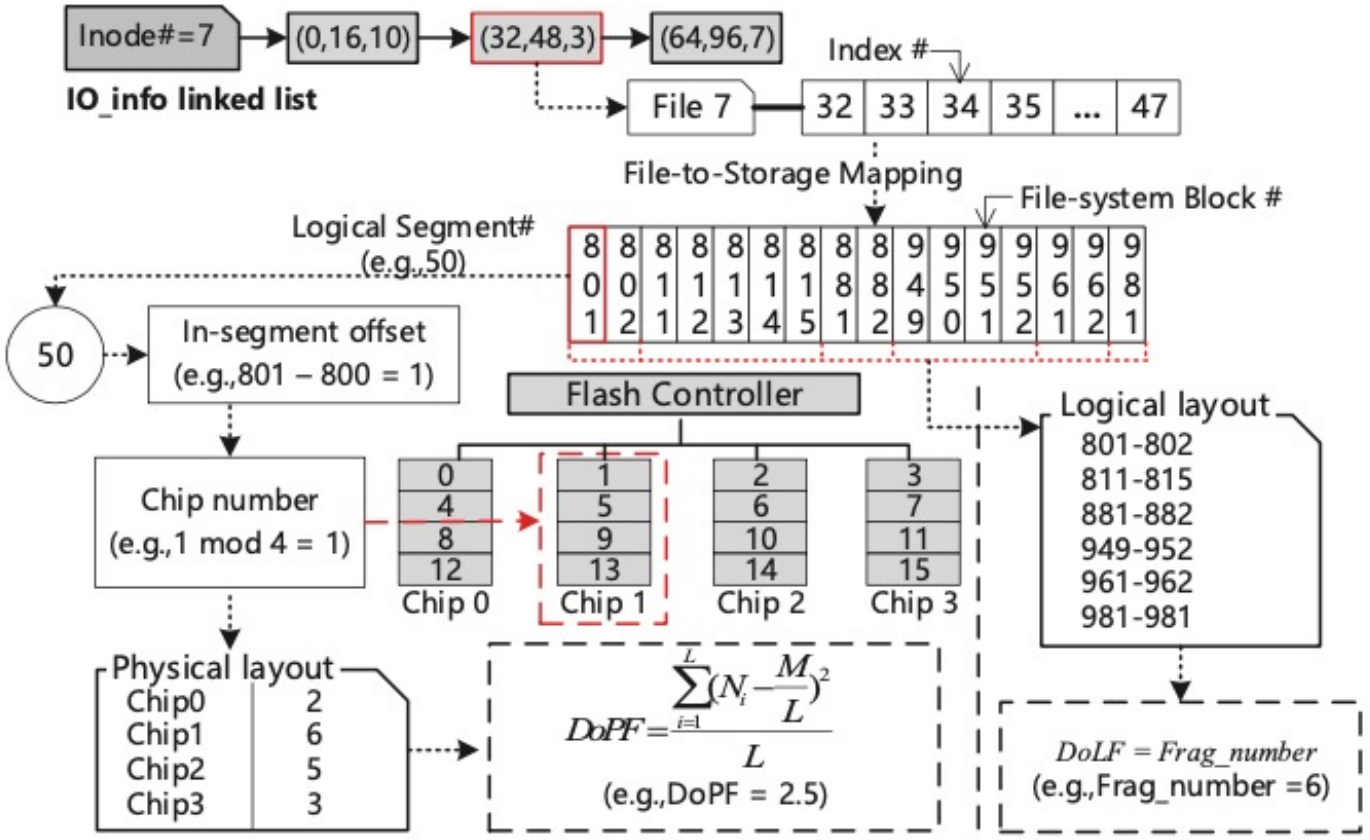


Fig. 9. An example of calculating the DoLF and the DoPF.

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

Defragmentation Module:

I. Select the data to be migrated and filter out fragments that have little impact on I/O performance.

II. adjoining space allocation allocates adjoining free space from the area.

III. the data migration task of the host is offloaded to the SSD.

IV. Finally, the file system retains the metadata blocks of the migrated data.

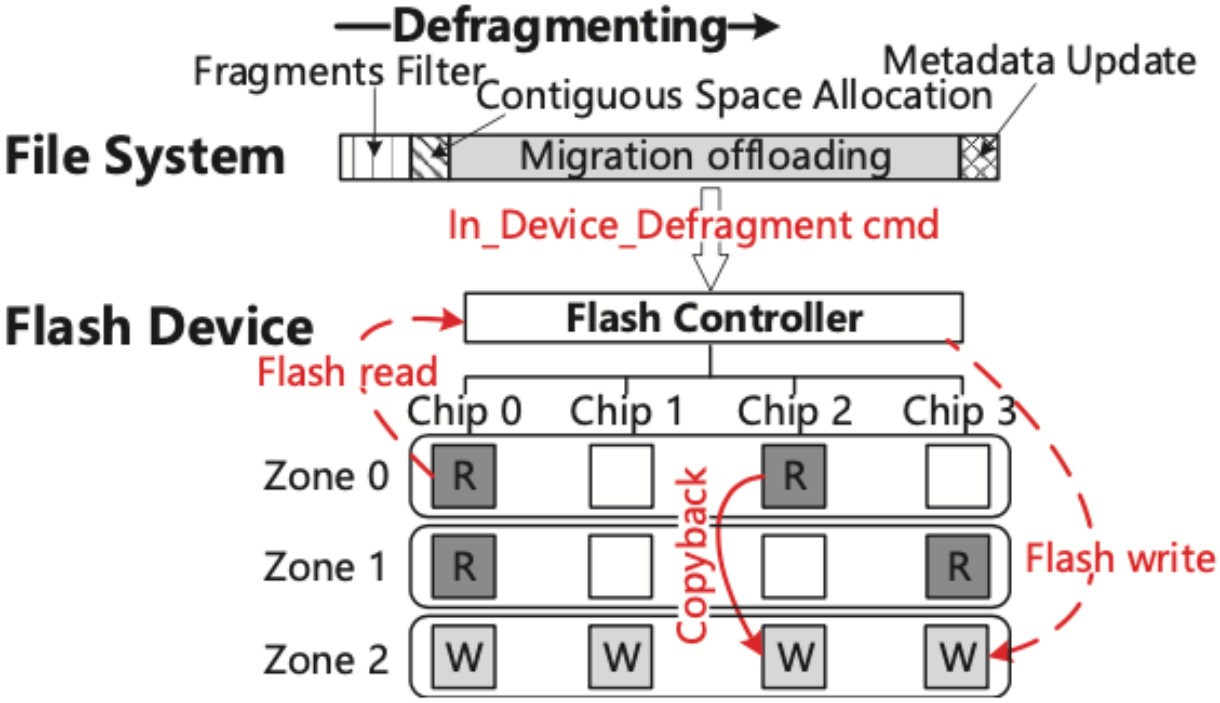


Fig. 10. The overall process of The Defragmentation Module.

4. Related literatures

4.3. InDeF: An Advanced Defragmenter Supporting Migration Offloading on ZNS SSD. ICCD 2022

Defragmentation Module:

- **Fragments filter.**
- **Migration offloading.**
  I. Command generating: InDeF divides the source data into two groups, one of which needs to be transferred from the host memory to the device and the other is processed through migration offload.
  II. Command sending: In_Device_Defragment command implements in-storage data migration
  III. Command processing: The SSD internal controller handles migration offloads and host requests based on the zone's write policy.
    - If the source block and target block belong to the same chip, use the copyback command to transfer within the chip.
    - Otherwise, the data block is first read into the DRAM in the SSD and then written to the target location.

5. Future work

- **1. Read the zns code in FEMU**

- **2. Read more literatures**

Thank you for listening

Presented by Chen Haodong

Supervised by Zhao Yumiao

HDU – Center of Cloud computing