


# ConZone: A Zoned Flash Storage Emulator for Consumer Devices

Dingcui Yu<sup>1,2</sup>, Jialin Liu<sup>1,2</sup>, Yumiao Zhao<sup>1,2</sup>, Wentong Li<sup>1,2</sup>, Ziang Huang<sup>1,2</sup>,  
Zonghuan Yan<sup>1,2</sup>, Mengyang Ma<sup>1,3</sup>,  Liang Shi<sup>1,2,4</sup>

<sup>1</sup>MoE Engineering Research Center of Software/Hardware Co-Design Technology and Application,  
East China Normal University, Shanghai, China

<sup>2</sup>College of Computer Science and Technology, East China Normal University, Shanghai, China

<sup>3</sup>Software Engineering Institute, East China Normal University, Shanghai, China

<sup>4</sup>Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China

**Abstract**—Considering the potential benefits to lifespan and performance, zoned flash storage is expected to be incorporated into the next generation of consumer devices. However, due to the limited volatile cache and heterogeneous flash cells of consumer-grade flash storage, adopting a zone abstraction requires additional internal hardware design to maximize its benefits. To understand and efficiently improve the hardware design on consumer-grade zoned flash storage, we present ConZone—the first emulator tailored to the characteristics of consumer-grade zoned flash storage. Users can explore the internal architecture and management strategies of consumer-grade zoned flash storage and integrate the optimization with software. We validate the accuracy of ConZone by realizing a hardware architecture for consumer-grade zoned flash storage and comparing it with the state-of-the-art. We also make a case study for read performance research with ConZone to explore the design of mapping mechanisms and cache management strategies.

## I. INTRODUCTION

The commercialization of high-density 3D NAND flash memory based on quad-level cell (QLC) technology has been widely adopted and is poised to be integrated into consumer-grade flash storage to further expand storage capacity [1] [2] [3]. Compared to triple-level cell (TLC) technology, QLC exhibits a significant reduction in write bandwidth, an increase in read latency by several tens of microseconds, and a decrease in the number of program/erase cycles [4] [5] [6] [7]. To maintain user experience and prolong the lifespan of storage devices, it is imperative to further optimize the storage systems of consumer devices [2] [3] [8] [9] [10] [11]. Traditional consumer-grade flash storage does not restrict the host from performing in-place updates at the granularity of pages and executes garbage collection within the flash storage. This is convenient for the host, but leads to two major issues. Firstly, flash storage requires to employ the page mapping [12] [13] to manage the flash media. As the capacity of flash storage increases, it is difficult for the limited logical-to-physical (L2P) cache in consumer-grade flash storage to cache all mapping table entries, resulting in degraded read performance and poor user experience [1]. Secondly, there is a time gap between the host invalidating data and the flash storage recognizing that the data is invalid, which may cause the flash storage to move

invalid data during garbage collection, thereby reducing the lifespan [14] [15].

The novel zone abstraction [1] [16] can address the aforementioned issues. Firstly, zone abstraction requires the host to perform sequential writes, allowing the flash storage to use larger granularity mapping entries, which occupies less L2P cache and mitigates the random read performance decline with increasing read range. Secondly, zone abstraction delegates the task of erasure to the host, avoiding the movement of invalidated data and thereby enhancing the device's lifespan. Mainstream consumer device manufacturers (e.g., Google, Samsung, and Huawei) have adopted the flash friendly file system (F2FS) in the storage systems, which can accommodate the stringent requirements of the zone abstraction [1] [17]. Currently, due to the potential benefits of zone abstraction and the compatibility of consumer devices with it, zoned flash storage is expected to be incorporated into the next generation of consumer devices. Recently, JEDEC has released zoned universal flash storage (UFS) standard for consumer storage [18]. Samsung explored I/O stack optimization based on zoned flash storage for consumer devices [1]. We believe that adopting zoned flash storage in consumer devices to optimize performance and lifespan is one of the future trends.

Compared to zoned namespace solid state drives (ZNS SSDs) in servers, zoned flash storage in consumer devices has limited volatile cache and programs some flash blocks as single-level cell (SLC) to serve as secondary buffers [19] [20] [21]. These characteristics result in distinct access pattern for consumer-grade zoned flash storage. For write accesses, the limited volatile write buffer may be flushed frequently and prematurely. Specifically, all the open zones have to share a small amount of write buffers. If the host switches the writing zone and the corresponding write buffer is occupied by another zone before, the data in the write buffer should be flushed. If this flush occurs prematurely, meaning the data being flushed is less than a programming unit, the data will be temporarily written to the SLC-based secondary write buffer through partial programming. For read accesses, due to the limited capacity of the L2P cache, each data read has a probability of requiring a mapping table lookup [22] [23]. When adopting zone abstraction, larger mapping granularity can be employed. However, due to the possibility of premature flush, data within the same zone may not be physically contiguous. Therefore, page mapping is still

This work is supported by the NSFC (62072177), Shanghai Science and Technology Project (22QA1403300) and the Open Project Program of Wuhan National Laboratory for Optoelectronics NO.2023WNLOKF004. The corresponding author is Liang Shi (shi.liang.hk@gmail.com).

required when data is written to SLC. In summary, on the one hand, it is necessary to reconsider the usage of write buffers and SLC-based buffers to avoid premature flushing that leads to superfluous writes, thereby reducing the impact on the write performance and lifespan of flash storage. On the other hand, it is essential to redesign the mapping table and L2P cache to fully leverage the advantages of the zone abstraction, thereby improving read performance. Adopting zoned abstraction for consumer-grade flash storage requires further design of internal hardware. However, transforming a conceptual idea into a real product takes time. Therefore, it is crucial to develop a zoned flash storage emulator for consumer devices to quickly validate their effectiveness.

However, existing ZNS emulators lack implementations of features in consumer devices, such as limited volatile write buffer and L2P cache, heterogeneous flash cells, and hybrid mapping, hindering the research on consumer-grade zoned flash storage. Motivated by these limitations, this paper presents ConZone, a consumer-grade zoned flash storage emulator. ConZone supports the hardware required for consumer-grade zoned flash storage, including write buffers, L2P caches, hybrid mapping, and SLC-based secondary buffer. Since applications and file systems can regard ConZone as a common storage device, users can further design and validate hardware-software co-optimization schemes for consumer-grade zoned flash storage devices. In the experiment, we validate the accuracy and features of ConZone by comparing an example of hardware configuration with the latest work that describing the organization and performance of zoned flash storage in consumer systems [1]. We also explain one case study to demonstrate that ConZone can be helpful for further research on hardware design of consumer-grade zoned flash storage. The source code of ConZone is publicly available at <https://github.com/DingcuiYu/ConZone>. The followings are the contributions of this paper:

- Develop a software framework to support research on consumer-grade zoned flash storage with diverse I/O characteristics.
- Enable the fast prototyping and development of hardware architecture within consumer-grade zoned flash storage through an accurate emulator.
- Make a case study for redesign the internal hardware to improve read performance of consumer-grade zoned flash storage.

## II. BACKGROUND AND RELATED WORK

### A. Flash Storage in Consumer Devices

In consumer devices, the size of a flash page is 16KiB. Multiple flash pages form a flash block. Flash blocks of different flash chips with the same offset form a superblock. Each open superblock is associated with a write pointer to point to new physical address for the next write. Several flash pages of multi-level flash cells need to be programmed (or written) simultaneously, such as TLC or QLC, while flash pages of single-level flash cells can be programmed partially with a programming unit of 4KiB, such as SLC. Program units of different flash chips with the same offset form a superpage.

As the program unit of multi-level cell flash storage is large, the data written by the host should be aggregated in volatile write buffers. Each writing superblock has a write buffer, and the size of a write buffer is one superpage to fully utilize the device's parallelism during flush operations. Due to the lack of power loss protection, consumer systems frequently issue synchronous writes [1]. The write buffer may be flushed prematurely without enough data to be programmed (Fig. 1 (a) W.1). The SLC flash blocks are organized as secondary write buffer for aggregation (Fig. 1 (a) W.2). Finally, the buffered data is written to normal multi-level cell flash blocks (Fig. 1 (a) W.3).

Flash memory must be erased before it can be written to, and the erasure unit is a flash block. When written data is updated, the storage controller marks the old page as invalid and allocates a new page according to the write pointer. Therefore, there are two functional modules within the storage: flash translation layer (FTL) and garbage collection (GC). The FTL is responsible for managing the mapping between logical addresses and physical addresses. To accelerate the read process, L2P entries are cached on demand in volatile L2P cache [12] [13]. When reading data, the L2P cache is first accessed to determine the current location of the data (Fig. 1 (a) R.1). If an L2P cache miss occurs, the L2P entry must be fetched from flash memory, leading to degraded read performance (Fig. 1 (a) R.2). Finally, the storage controller read data with its physical address (Fig. 1 (a) R.3). The GC module is responsible for moving valid pages during the erasure of flash blocks and updating the L2P mapping (Fig. 1 (a) E.1, E.2).

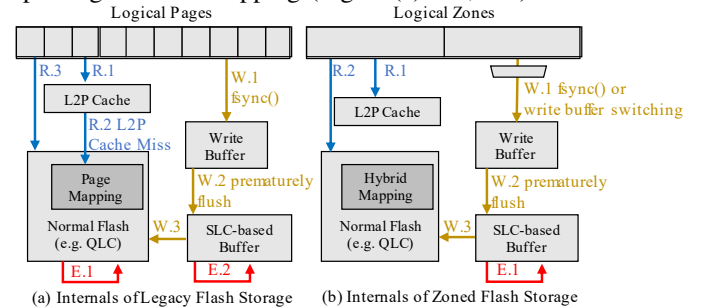


Fig. 1. Comparison of legacy flash storage and zoned flash storage.

### B. Zone Abstraction for Consumer Devices

Zone abstraction changes access patterns for flash storage. First, premature flushes are more frequent. Consumer devices use the F2FS file system, which accommodates zoned abstraction. F2FS can open up to 6 zones simultaneously, and each open zone requires a write buffer within the storage as one zone corresponds to one superblock. Assuming a superpage size of 384 KiB, a total write buffer size of  $384 \times 6 = 2304$  KiB is needed, whereas consumer-grade flash storage only have up to 1 MiB of write buffer [1]. Therefore, when switching the zone being written to, write buffer contention occurs, causing data from other zones in the buffer to be flushed prematurely (Fig. 1 (b) W.1, W.2, W.3). Secondly, the forced sequential writes of zone abstraction allow the storage to use a coarser-grained mapping table. As there are data temporarily written to SLC flash blocks, physical pages inside a zone is not always physically contiguous. Hybrid mapping

should be adopted. The limited L2P cache can cache larger range of logical address-physical address mappings. Therefore, L2P misses are reduced and the flash blocks are only read once during read process (Fig. 1 (b) R.1, R.2). Finally, zone abstraction delegate the responsibility of garbage collection to the host, thereby zoned flash blocks are not need to GC (Fig. 1 (b) E.1).

### C. Current Zoned Flash Storage Emulators

Currently available zoned flash storage emulators are designed for enterprise ZNS SSDs, including FEMU [24], ConfZNS [25], and NVMeVirt [26]. None of them can fully emulate the internal hardware of consumer-grade zoned flash storage. Table I compares these emulators and ConZone. FEMU supports write buffer, but lacks L2P cache and FTL in ZNS mode. ConfZNS, developed based on FEMU, supports zone mapping in FTL but lacks write buffer and L2P cache. Both FEMU and ConfZNS are based on virtual machines, which introduce additional latency of tens of microseconds in their I/O stack. This drawback makes FEMU and ConfZNS incapable of simulating low-latency media like SLC [26], failing to accurately reflect the performance of premature write buffer flushes. NVMeVirt is not based on virtualization technology. It is a Linux kernel module that provides the system with a virtual NVMe device, capable of emulating low-latency media. However, its ZNS mode does not support heterogeneous flash cells. The L2P cache and hybrid mapping is also not supported. ConZone proposed by this paper realize the necessary internal hardware and is tailored for emulating the consumer-grade zoned flash storage.

TABLE I  
EXISTING ZONED FLASH STORAGE EMULATORS AND CONZONE

	FEMU [24]	ConfZNS [25]	NVMeVirt [26]	ConZone
Low-latency media	No	No	Yes	Yes
Heterogeneous media	No	No	No	Yes
# of write buffers	Yes	No	No	Yes
L2P cache	No	No	No	Yes
L2P mapping	No	Zone	No	Hybrid

## III. CONZONE INTERNALS

### A. Overview

Fig. 2 illustrates the key components of ConZone. ConZone, based on NVMeVirt [26], implements the necessary hardware modules for consumer-grade zoned flash storage during read, write, and erase (or zone reset) operations. These modules include L2P caches and hybrid mapping tables, write buffers and heterogeneous flash cells, and composite garbage collection.

For write operations, we limit the number of write buffers and establish a mapping relationship between the write buffers and zones (W.1). When switching the zone being written to, the data in the write buffer is flushed to the normal flash blocks or temporarily stored in SLC flash blocks (W.2) and finally written to normal flash blocks (W.3) as described in Section II-B. For read operations, we still use a page mapping table to index all data. However, as the consecutive physical data of normal flash blocks within a zone gradually reach a chunk (4MiB) or a zone, we aggregate the mapping entries as one entry with larger range and store it to L2P cache. We call this mechanism

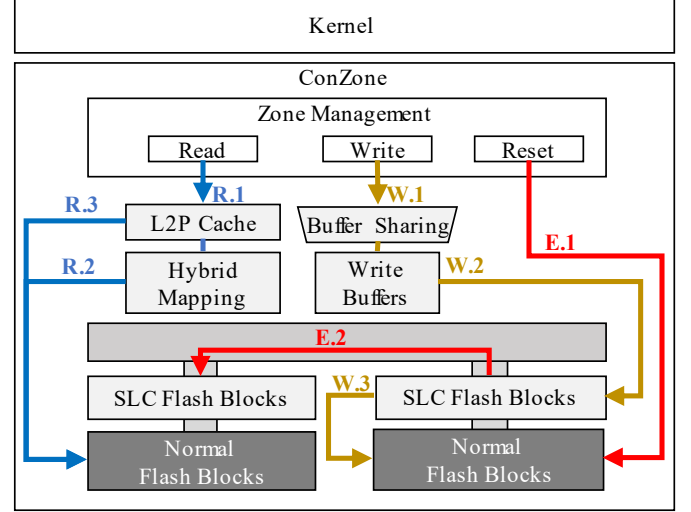


Fig. 2. Internals of ConZone.

as hybrid mapping. Each read operation from the host requires querying the L2P cache (R.1). If L2P cache miss occurs, due to the use of hybrid mapping, multiple flash read for the mapping table are required, which may lead to unstable read performance (R.2). Finally, the data is transferred back to the host, and the L2P cache is updated (R.3). For erase operations, we embed a complete garbage collection mechanism in the SLC flash blocks (E.1), including the selection of victim blocks, the migration of valid pages, block erasure, and mapping table updates. On the other hand, since zones are mapped to the normal flash blocks, we directly erase the flash blocks and update the mapping table when the zone reset is handled (E.2).

### B. Write Path: Hybrid Media and Limited Write Buffer

Fig. 3 illustrates the write path of ConZone. Writes from different zones are directed to their corresponding write buffers for aggregation. The flush path is determined based on the data volume and the physical layout of the data within the current zone. If the data volume is sufficient for programming, it is directly flushed to a normal flash block (①). Otherwise, it is flushed to an SLC flash block (②). If the data stored in the SLC flash block combined with the new writing data reaches a programming unit, the data in the SLC is read out and invalidated (striped blocks in Fig. 3), and then flushed together with the new writing data to the normal flash block (③). Write pointers are used to point to new physical addresses. Specifically, ConZone binds the write pointer to a free superblock and the write pointer iterates within the superblock according to fixed rules. After traversing to the end of a superblock, ConZone rebinds the write pointer to a new free superblock. By iterating the write pointer multiple times, a certain amount of contiguous physical addresses can be reserved. Note that we reserve a corresponding number of normal flash blocks for each zone (square-patterned blocks in Fig. 3) to ensure that when the data within the zone is in the normal flash area, they are physically contiguous. When writing to the reserved area, the physical address can be calculated based on the logical address offset within the zone.

**Conflicting Zone-Write Buffer Mapping:** Considering the limited resources of mobile storage devices, it is not feasible to

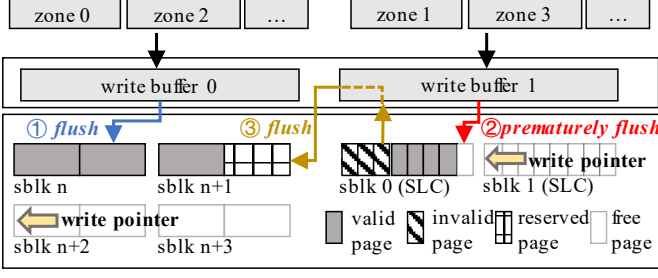


Fig. 3. Write Path of ConZone

allocate a write buffer for each open zone. Users can configure the number of write buffers. The mapping relationship between zones and write buffers is determined by taking the modulo of the zone index with the total number of write buffers.

**Heterogeneous Media and Extended Timing Model:** Due to the difference in programming units between SLC (4 KiB) and normal flash blocks (e.g. QLC, 64 KiB), the iteration of the write pointers differs. We maintain a separate write pointer for each of these regions. We extend the timing model to distinguish heterogeneous flash cells and present the default access latency of them in Table II. Values in Table II are mostly based on the published data in academic work. The read latency of SLC is determined after discussions with relevant engineers. We allow users to uniformly designate the first  $n$  flash blocks of each chip as SLC flash blocks and configure access latency of different media.

TABLE II  
LATENCY FOR DIFFERENT MEDIA IN CONZONE

	SLC	TLC	QLC
Program	75us [27]	937.5us [28]	6400us [29]
Read	20us	32us [28]	85us [29]

### C. Read Path: Hybrid Mapping and L2P Cache Management

Fig. 4 shows the read path of ConZone. When processing a read request, the L2P cache is first queried. ConZone sequentially translates the logical address into a logical zone address (LZA), logical chunk address (LCA), and logical page address (LPA), and attempts to match them in the L2P cache in turn (I). If the match is successful (II), the physical address (III) is calculated based on the offset of original logical address and translated logical address. Instead, the L2P mapping entry needs to be fetched from flash memory. Similarly, the mapping table is looked up by corresponding LZA, LCA and LPA in turn (①, ②, ③). ConZone utilizes two reserved bits in the mapping table entry to indicate the aggregation of L2P entries. If the magnitude of aggregated entries is the same as that of the mapping granularity (②), the L2P cache entry is generated and insert into the L2P cache (④). If the L2P cache is full, the cached entries are evicted based on the LRU rule.

**Hybrid Mapping:** ConZone supports aggregating mapping table entries with contiguous logical and physical addresses into a single L2P cache entry to improve read performance. Note that FTL still uses page mapping to record all mapping information. As shown in Fig. 5, to distinguish the magnitude of aggregation, we use two reserved bits as map bits in the

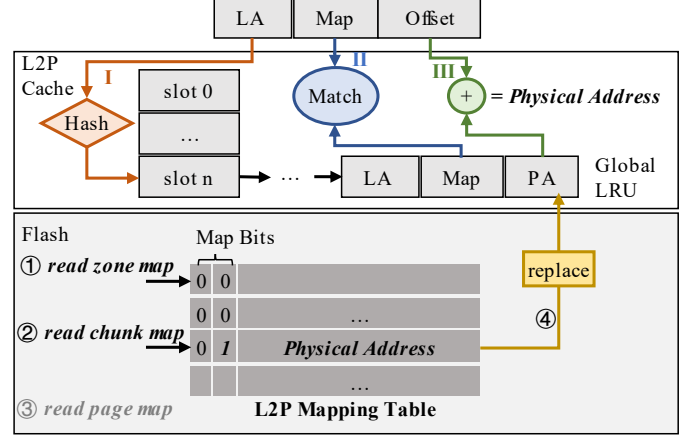
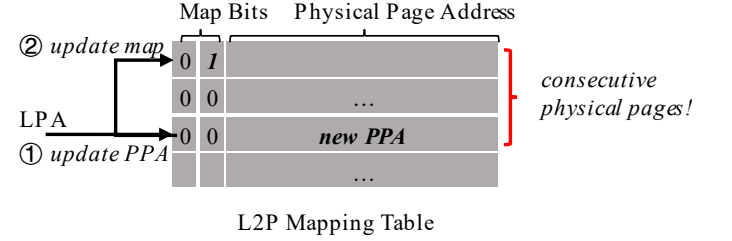


Fig. 4. Read Path of ConZone

mapping table entry to indicate three levels of aggregation, which is page, chunk (1024 pages) and zone. When the write buffer is flushed back, the logical-physical mapping relationship is established and the mapping table is updated (①). Since consecutive normal flash blocks are reserved for each zone, ConZone can determine whether the mapping table entries can be aggregated simply by comparing the physical address to the physical chunk/physical zone boundary (②). Data that is temporarily written to SLC flash blocks cannot then be aggregated because there is no guarantee of its continuity.



L2P Mapping Table

Fig. 5. ConZone's hybrid mapping mechanism, LPA for logical page addresses and PPA for physical page addresses

**Management of L2P Cache:** L2P cache entries include three domains: logical address, mapping granularity and physical address. To reduce the time of querying the L2P cache, logical addresses are mapped into multiple hash buckets and ConZone traverses the cache entries within the hash buckets in corresponding LZA, LCA and LPA turn. When both the logical address and the mapping granularity match, the L2P cache hit. If the L2P cache misses, the L2P mapping entries need to be fetched from flash memory. There is a challenge: *how to know the magnitude of aggregation of current logical address before reading the mapping table*. One possible solution is to use a bitmap to record the map bits of all logical addresses with a capacity overhead of about 0.006%. 1TB SSD requires about 64MB of SRAM, which is unacceptable in consumer devices. Another option is to perform multiple fetches in similar way that querying the L2P cache. Specifically, firstly assume one zone is aggregated, fetch the mapping entry of corresponding LZA, check its map bits. If the check fails, fetch the mapping entry of LCA and so on. This scheme makes L2P cache misses require multiple flash reads, resulting in degraded read performance. We make a case study on this issue in Section

IV-D to explore the negative effects of multiple fetching and propose a feasible design.

#### D. Erase Path: Composite Garbage Collection

SLC flash blocks and normal flash blocks have different access modes. Validation and invalidation of the former are fully managed by the storage controller, while the latter is fully managed by the host. Based on this, we design a GC mechanism for SLC flash blocks and normal flash blocks respectively. For SLC flash blocks, ConZone uses the full GC process. First, ConZone selects the victim superblock according to the number of valid pages, then moves the valid pages in SLC space, and finally erases the victim superblock and adds it to the free superblock list of SLC. For normal flash blocks, ConZone uses a simplified GC process. When the host resets the zone, ConZone directly erases the normal flash blocks allocated to the zone in the past. If the zone has some data in SLC, ConZone invalidates the corresponding data also.

#### E. Limitations and Discussions

**Non-power-of-2 Zone Size:** Zone abstraction based on Non-Volatile Memory Express (NVMe) standard does not allow for non-power-of-two zone sizes. Therefore, it is not allowed to set up a compliant zone when the flash media is TLC. Our temporary solution is to align the zone size and write the patched data to SLC flash pages. Note that these SLC flash pages are also reserved so that the corresponding L2P mapping table entries can be aggregated. Developers are working on supporting non-power-of-two zone sizes in NVMe, and we believe this restriction will be solved [30].

**Persistence of L2P Mapping Table Updates:** Since the L2P mapping table needs to be persisted into flash memory, how to update the mapping table entries is a design challenge. Currently L2P log is used within consumer-grade flash storage to accumulate L2P mapping table updates. The L2P log is flushed to flash memory when several updates are accumulated, and the flushing back of the L2P log may block host requests. In addition, how to seek the address of the L2P mapping entries after flushing back, and whether other structures of page tables need to be used are also topics that need to be explored. This feature will be realized in the future work.

**Conventional Zones:** In consumer devices, a certain amount of conventional zones need to be present to allow necessary in-place updates from the host, such as updating the metadata of F2FS. How to isolate such zones from sequential write zones and how to manage the resources of these two types of zones, such as write buffers and SLC flash blocks, remains an open topic. Currently ConZone does not implement such zones, but it is still possible to test the performance of F2FS by mounting the metadata area to a real flash storage. We also plan to extend the design in the future.

### IV. EVALUATION

In this section we will evaluate the functionality of ConZone. The followings are mainly explored:

- 1) How precisely can ConZone emulate the performance of zoned flash storage for consumer devices?

- 2) What are the benefits and challenges in zoned flash storage for consumer devices?
- 3) How the design of zoned flash storage internals affects I/O performance?

#### A. Evaluation Setup

**Evaluation Environment:** We implemented ConZone on an HP Z8 G4 workstation equipped with two Intel Xeon Silver 4114 2.20 GHz processors and 94 GiB of memory, based on the Linux kernel 6.8.0-40-generic. Due to the lack of consumer devices equipped with zoned flash storage, we extensively referenced and compared information disclosed in the latest academic work [1]. In our paper, we subsequently refer to it as “ZMS”. Due to the undisclosed internal structure of flash storage in existing consumer devices, we additionally implemented traditional flash storage in consumer devices based on descriptions from the recent work [1]. In our paper, we subsequently refer to this as “Legacy”. Besides, we also compare our work with the latest version of the FEMU simulator tool [24] with the same hardware configuration. We use flexible I/O tester (FIO) [31] to generate micro benchmark to evaluate ConZone.

**Configuration:** To mitigate the impact of virtualization on emulator performance, we reserved two dedicated cores for executing ConZone. We configured ConZone’s parameters by referencing ZMS. Specifically, we set the storage medium to TLC and configured the number of SLC flash blocks according to the requirements for write aggregation and alignment. Additionally, we set up two parallel channels, each with two chips. After discussions with corporate engineers, we set the channel bandwidth to 3200 MiB/s, referencing the standard bandwidth of UFS 4.0 and adding various redundant read data. The zone size and superblock size are the same to maximize read and write performance. For writes, we set the programming unit to 96 KiB. All zones share two write buffers, each with a size of 384 KiB, consistent to that of ZMS. For reads, due to the insufficient memory capacity, it is challenging to emulate a large flash storage. To approximate real conditions, we proportionally scaled down the L2P cache, setting the flash capacity to 1.5 GB and the L2P cache size to 12 KiB.

#### B. The Accuracy of ConZone

We use FIO to perform 512 KiB sequential read and write operations on ConZone, referencing the experiment of ZMS. Fig. 6 (a) shows the bandwidth of sequential I/O, where ST denotes single-threaded and MT denotes multi-threaded (with 4 threads). In terms of write performance, both single-threaded and multi-threaded, ConZone is very close to ZMS, indicating the accuracy of ConZone’s write emulation. Regarding read performance, the multi-threaded read bandwidth of ConZone is comparable to that of ZMS, whereas the single-threaded read bandwidth is lower. There are two possible reasons for this. First, the CPU difference may reduce the single-threaded read bandwidth. The CPU used in the ZMS (SM8350) may have better single core performance than our CPU (e.g., SM8350 has a higher Geekbench 6 single-core score than Xeon 4114). When FIO runs in the multi-threaded mode, it can use multiple cores to send requests, compensating for the bandwidth difference



caused by the cores' performance gap. Second, ZMS has not disclosed the details of its L2P cache implementation, and differences in the L2P cache implementation between ConZone and ZMS may also lead to bandwidth differences. The write performance of FEMU is slightly higher than that of ZMS, probably because FEMU can not simulate the channel bandwidth of the UFS interface. There is a large gap between the read latency of FEMU and ZMS, which is due to the fact that the implementation of FEMU is based on Linux KVM, and the need for host/client switching during I/O accesses brings indispensable latency fluctuations, which are difficult to simulate the read latency of flash, which is in the tens of microseconds [26].

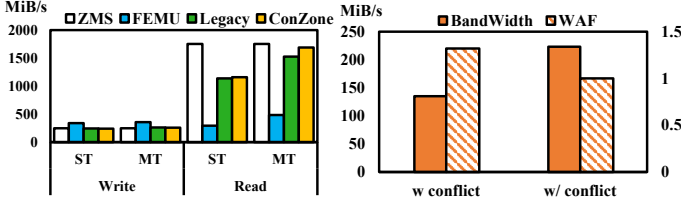


Fig. 6. Comparison of Sequential I/Os with Different Configurations

### C. The Benefits and Challenges of Zoned Flash Storage

Fig. 6 (a) also demonstrates the performance advantages of zoned flash storage. ConZone's write bandwidth is comparable to that of Legacy, while its read bandwidth is 1% higher in single-threaded and 10% higher in multi-threaded compared to Legacy. Note that to leverage the characteristics of sequential reads, we enable prefetching for the L2P cache under Legacy, with a prefetch window size of 1023 entries, allowing the L2P entries of a chunk (4 MiB) to be read in upon an L2P miss. For fairness, ConZone only aggregates mapping table entries with a mapping range of a chunk. The read bandwidth advantage of ConZone stems from the reduction of L2P cache capacity usage with chunk-level L2P aggregation. More L2P mappings can be cached, thereby lowering the L2P cache miss rate.

As describing in Section I, there are two main challenges in consumer-grade zoned flash storage. One is write buffer conflicts, the other is L2P cache and mapping table design. The latter will be presented as a case study in Section IV-D. Fig. 6 (b) presents the negative influence of write buffer conflicts. Here we designed the following test scheme. First, we split odd-numbered zones and even-numbered zones into two separate write buffers. Then, we used dual threads to write data the size of one zone each, setting the write granularity to 48KiB to trigger prematurely flushing of the write buffer. When the zone numbers being written to have the same parity, write buffer conflicts occur; otherwise, there are no conflicts. The write bandwidth without buffer conflicts is 65% higher than with conflicts, and write amplification is reduced by 24%. Therefore, it is crucial to avoid write buffer conflicts when designing zoned flash storage.

### D. Case for Read Performance with Different Internals

**Impact of Mapping Mechanisms:** In this section, we test the impact of using page mapping and hybrid mapping on the 4KiB random read performance with zoned flash storage. All

the tests are executed under the same data volume but different read ranges. As shown in the left graph of Fig. 7, when the read range is 1MiB, the KIOPS of both map mechanisms is 20.2K. All mapping entries can be stored in the L2P cache for both page mapping and hybrid mapping in this case. However, when the read range is expanded to 16MiB and 1GiB, the KIOPS for page mapping decreases by 16.5% and 33.5% compared to the 1MiB range respectively. The advantages of hybrid mapping is also reflected in the tail latency. In the right graph of Fig. 7, the tail latency of random reads under hybrid mapping remains around 50us, as all mapping entries can be cached.

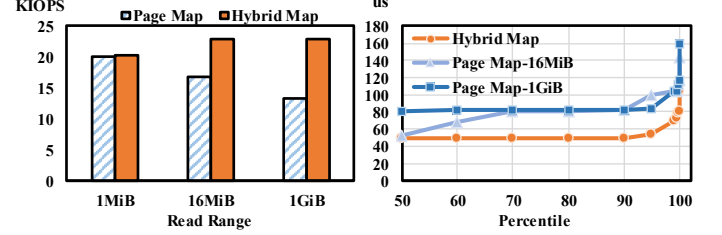


Fig. 7. Impact of Mapping Mechanisms on Random Reads

**Impact of L2P Search Strategy:** The L2P cache miss cost is higher in hybrid mapping due to the multiple fetches of L2P mapping entries. Fig. 8 compares the impact of L2P misses under performance-optimized (BITMAP) and capacity-optimized (MULTIPLE). When the L2P miss rate reaches 27.4%, the KIOPS of MULTIPLE is 10% lower than that of BITMAP and the tail latency is also higher. One feasible solution is to pin the aggregated L2P mapping entries in the L2P cache once they are generated. When the L2P mapping entry with larger mapping range is generated, the covered L2P mapping entries are evicted. In the hybrid mapping mechanism, all the mapping entries can be aggregated as a zone mapping entry when a zone is full. Assuming the zone size is 16 MiB and the size of L2P cache entry is 4 B, 1 TiB flash storage needs only 256 KiB SRAM to cache all the entries. The capacity overhead is acceptable and this solution is realized as configure option in ConZone.

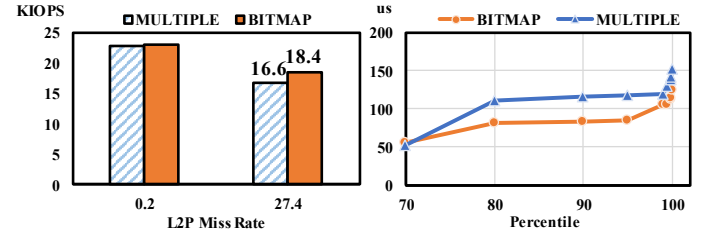


Fig. 8. Impact of L2P Search Strategy on Random Reads with Hybrid Map

### V. CONCLUSION

This paper introduces ConZone, the first zoned flash storage emulator tailored for consumer devices. ConZone features novel hardware internals distinct from existing emulators. It opens a new opportunity for developers to redesign the flash storage internal to better utilize the zone abstraction. We demonstrate the usefulness of ConZone for consumer-grade zoned flash storage research. We are planning to support more features inside a consumer-grade zoned flash storage, such as conventional zones. Furthermore, we are working on redesigning hardware management to solve the write buffer conflicts issue.

## REFERENCES

- [1] Joo-Young Hwang, Seokhwan Kim, Daejun Park, Yong-Gil Song, Junyoung Han, Seunghyun Choi, Sangyeun Cho, and Youjip Won. {ZMS}: Zone abstraction for mobile flash storage. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 173–189, 2024.
- [2] Longfei Luo, Dingcui Yu, Yina Lv, and Liang Shi. Critical data backup with hybrid flash-based consumer devices. *ACM Transactions on Architecture and Code Optimization*, 21(1):1–23, 2023.
- [3] Longfei Luo, Han Wang, Dingcui Yu, Yina Lv, and Liang Shi. Cpf: A cross-layer prefetching framework for high-density flash-based storage. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2024.
- [4] Wontaek Jung, Hyunggon Kim, Do-Bin Kim, Tae-Hyun Kim, Namhee Lee, Dongjin Shin, Minyoung Kim, Youngsik Rho, Hun-Jong Lee, Yujin Hyun, et al. 13.3 a 280-layer 1tb 4b/cell 3d-nand flash memory with a 28.5 gb/mm<sup>2</sup> areal density and a 3.2 gb/s high-speed io rate. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 67, pages 236–237. IEEE, 2024.
- [5] Koichi Kawai, Yuichi Einaga, Yoko Oikawa, Yankang He, Biagio Iorio, Shigekazu Yamada, Yoshihiko Kamata, Tomoko Iwasaki, Andrea D’alessandro, Erwin Yu, et al. 13.7 a 1tb density 3b/cell 3d-nand flash on a 2yy-tier technology with a 300mb/s write throughput. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 67, pages 244–246. IEEE, 2024.
- [6] Longfei Luo, Dingcui Yu, Hang Li, Yunpeng Song, Yina Lv, Edwin H-M Sha, and Liang Shi. Revisiting trim on high-density flash-based hybrid storage systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.
- [7] Yina Lv, Liang Shi, Qiao Li, Congming Gao, Yunpeng Song, Longfei Luo, and Youtao Zhang. Mgc: Multiple-gray-code for 3d nand flash based high-density ssds. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 122–136. IEEE, 2023.
- [8] Yunpeng Song, Yina Lv, and Liang Shi. Adaptive differential wearing for read performance optimization on high-density nand flash memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.
- [9] Ben Gu, Longfei Luo, Yina Lv, Changlong Li, and Liang Shi. Dynamic file cache optimization for hybrid ssds with high-density and low-cost flash memory. In *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pages 170–173. IEEE, 2021.
- [10] Wentong Li, Dingcui Yu, Yunpeng Song, Longfei Luo, and Liang Shi. Elasticzram: Revisiting zram for swapping on mobile devices. In *Proceedings of the 61st ACM/IEEE Design Automation Conference*, pages 1–6, 2024.
- [11] Wentong Li, Liang Shi, Hang Li, Changlong Li, and Edwin Hsing-Mean Sha. Iosr: Improving i/o efficiency for memory swapping on mobile devices via scheduling and reshaping. *ACM Transactions on Embedded Computing Systems*, 22(5s):1–23, 2023.
- [12] Aayush Gupta, Youngjae Kim, and Bhuvan Ugaonkar. Dftl: a flash translation layer employing demand-based selective caching of page-level address mappings. *Acm Sigplan Notices*, 44(3):229–240, 2009.
- [13] Zhiguang Chen, Nong Xiao, Fang Liu, and Yimo Du. Hot data-aware ftl based on page-level address mapping. In *2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC)*, pages 713–718. IEEE, 2010.
- [14] Kirock Kwon, Dong Hyun Kang, Jonggyu Park, and Young Ik Eom. An advanced trim command for extending lifetime of tlc nand flash-based storage. In *2017 IEEE International Conference on Consumer Electronics (ICCE)*, pages 424–425. IEEE, 2017.
- [15] Yu Liang, Cheng Ji, Chenchen Fu, Rachata Ausavarungnirun, Qiao Li, Riwei Pan, Siyu Chen, Liang Shi, Tei-Wei Kuo, and Chun Jason Xue. itrim: I/o-aware trim for improving user experience on mobile devices. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(9):1782–1795, 2020.
- [16] Matias Björling, Abutalib Aghayev, Hans Holmberg, Aravind Ramesh, Damien Le Moal, Gregory R Ganger, and George Amvrosiadis. {ZNS}: Avoiding the block interface tax for flash-based {SSDs}. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 689–703, 2021.
- [17] Wenxin Wang, Yaqi Li, Liang Shi, and Edwin H-M Sha. Eliminate critical fragmentation of f2fs in mobile devices with controller co-design. In *2024 13th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pages 1–6. IEEE, 2024.
- [18] JEDEC. Zoned storage for ufs. <https://www.jedec.org/standards-documents/docs/jesd220-5>.
- [19] Liang Shi, Longfei Luo, Yina Lv, Shicheng Li, Changlong Li, and Edwin Hsing-Mean Sha. Understanding and optimizing hybrid ssd with high-density and low-cost flash memory. In *2021 IEEE 39th International Conference on Computer Design (ICCD)*, pages 236–243. IEEE, 2021.
- [20] Yina Lv, Liang Shi, Yunpeng Song, and Chun Jason Xue. Access characteristic guided partition for nand flash-based high-density ssds. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(12):4643–4656, 2023.
- [21] Shicheng Li, Longfei Luo, Yina Lv, and Liang Shi. Latency aware page migration for read performance optimization on hybrid ssds. In *2022 IEEE 11th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pages 33–38. IEEE, 2022.
- [22] Wookhan Jeong, Hyunsoo Cho, Yongmyung Lee, Jaegy Lee, Songho Yoon, Jooyoung Hwang, and Donggi Lee. Improving flash storage performance by caching address mapping table in host memory. In *9th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 17)*, 2017.
- [23] Shengzhe Wang, Zihang Lin, Suzhen Wu, Hong Jiang, Jie Zhang, and Bo Mao. Learnedftl: A learning-based page-level ftl for reducing double reads in flash-based ssds. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 616–629. IEEE, 2024.
- [24] Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Matias Björling, and Haryadi S Gunawi. The {CASE} of {FEMU}: Cheap, accurate, scalable and extensible flash emulator. In *16th USENIX Conference on File and Storage Technologies (FAST 18)*, pages 83–90, 2018.
- [25] Inho Song, Myoungsoon Oh, Bryan Suk Joon Kim, Seehwan Yoo, Jaedong Lee, and Jongmoo Choi. Confzns: A novel emulator for exploring design space of zns ssds. In *Proceedings of the 16th ACM International Conference on Systems and Storage*, pages 71–82, 2023.
- [26] Sang-Hoon Kim, Jaehoon Shim, Euidong Lee, Seongyeop Jeong, Ilkueon Kang, and Jin-Soo Kim. {NVMeVirt}: A versatile software-defined virtual {NVMe} device. In *21st USENIX Conference on File and Storage Technologies (FAST 23)*, pages 379–394, 2023.
- [27] Toshiyuki Kouchi, Noriyasu Kumazaki, Masashi Yamaoka, Sanad Bushnaq, Takuyo Kodama, Yuki Ishizaki, Yoko Deguchi, Akio Sugahara, Akihiro Imamoto, Norichika Asaoka, Ryosuke Isomura, Takaya Handa, Junichi Sato, Hiromitsu Komai, Atsushi Okuyama, Naoaki Kanagawa, Yasufumi Kajiyama, Yuri Terada, Hidekazu Ohnishi, Hiroki Yabe, Cynthia Hsu, Mami Kakoi, and Masahiro Yoshihara. 13.5 a 128gb 1b/cell 96-word-line-layer 3d flash memory to improve random read latency with tprog=75µs and tr=4µs. In *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, pages 226–228, 2020.
- [28] Koichi Kawai, Yuichi Einaga, Yoko Oikawa, Yankang He, Biagio Iorio, Shigekazu Yamada, Yoshihiko Kamata, Tomoko Iwasaki, Andrea D’alessandro, Erwin Yu, et al. 13.7 a 1tb density 3b/cell 3d-nand flash on a 2yy-tier technology with a 300mb/s write throughput. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 67, pages 244–246. IEEE, 2024.
- [29] Wontaek Jung, Hyunggon Kim, Do-Bin Kim, Tae-Hyun Kim, Namhee Lee, Dongjin Shin, Minyoung Kim, Youngsik Rho, Hun-Jong Lee, Yujin Hyun, et al. 13.3 a 280-layer 1tb 4b/cell 3d-nand flash memory with a 28.5 gb/mm<sup>2</sup> areal density and a 3.2 gb/s high-speed io rate. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 67, pages 236–237. IEEE, 2024.
- [30] Pankaj Raghav. Support zoned block devices with non-power-of-2 zone sizes. <https://lore.kernel.org/lkml/860fb643-8a1a-225e-13e7-e68a4b6f3842@opensource.wdc.com/T/>.
- [31] fio. Flexible i/o tester. [https://fio.readthedocs.io/en/latest/fio\\_doc.html](https://fio.readthedocs.io/en/latest/fio_doc.html).