electronics

Article

Selective Power-Loss-Protection Method for Write Buffer in ZNS SSDs

Junseok Yang, Seokjun Lee and Sungyong Ahn *

School of Computer Science and Engineering, Pusan National University, Busan 46241, Korea; junseokyang@pusan.ac.kr (J.Y.); sky_lark0401@pusan.ac.kr (S.L.)
* Correspondence: sungyong.ahn@pusan.ac.kr

Abstract: Most SSDs (solid-state drives) use an internal DRAM (Dynamic Random Access Memory) to improve the I/O performance and extend SSD lifespan by absorbing write requests. However, this volatile memory does not guarantee the persistence of buffered data in the event of sudden power-off. Therefore, highly reliable enterprise SSDs employ power-loss-protection (PLP) logic to ensure the durability of buffered data using the back-up power of capacitors. The SSD must provide enough capacitors for the PLP in proportion to the size of the volatile buffer. Meanwhile, emerging ZNS (Zoned Namespace) SSDs are attracting attention because they can support many I/O streams that are useful in multi-tenant systems. Although ZNS SSDs do not use an internal mapping table unlike conventional block-interface SSDs, a large write buffer is required to provide many I/O streams. The reason is that each I/O stream needs its own write buffer for write buffering where the host can allocate separate zones to different I/O streams. Moreover, the larger capacity and more I/O streams the ZNS SSD supports, the larger write buffer is required. However, the size of the write buffer depends on the amount of capacitance, which is limited not only by the SSD internal space, but also by the cost. Therefore, in this paper, we present a set of techniques that significantly reduce the amount of capacitance required in ZNS SSDs, while ensuring the durability of buffered data during sudden power-off. First, we note that modern file systems or databases have their own solutions for data recovery, such as WAL (Write-ahead Log) and journal. Therefore, we propose a selective power-loss-protection method that ensures durability only for the WAL or journal required for data recovery, not for the entire buffered data. Second, to minimize the time taken by the PLP, we propose a balanced flush method that temporarily writes buffered data to multiple zones to maximize parallelism and preserves the data in its original location when power is restored. The proposed methods are implemented and evaluated by modifying FEMU (QEMU-based Flash Emulator) and RocksDB. According to experimental results, the proposed selective-PLP reduces the amount of capacitance by 50 to 90% while retaining the reliability of ZNS SSDs. In addition, the balanced flush method reduces the PLP latency by up to 96%.

Keywords: power-loss protection; energy efficiency; reliability; solid-state drives; Zoned Namespace; RocksDB; ZenFS

Citation: Yang, J.; Lee, S.; Ahn, S. Selective Power-Loss-Protection Method for Write Buffer in ZNS SSDs. *Electronics* **2022**, *11*, 1086. https://doi.org/10.3390/electronics11071086

Academic Editor: Marco Vacca

Received: 25 December 2021
Accepted: 28 March 2022
Published: 30 March 2022

1. Introduction

SSDs (solid-state drives) based on NAND flash memory are attractive due to various benefits such as high I/O performance, low power consumption, and shock resistance [1]. However, because of erase-before-write constraints and limited erase count of NAND flash memory, the increase in write requests adversely affects the lifespan of the SSD. Therefore, most SSDs use DRAM-based volatile write buffers inside the SSD to improve the write performance and the life of the SSD by absorbing the number of writes in NAND flash memory. However, the volatile write buffers do not guarantee the persistence of buffered data in the event of a sudden power-off [2,3].

Therefore, the fsync() system call should be invoked after executing the write command to flush buffered data from SSD internal write buffer to NAND flash memory. However, frequent flushing causes the degradation of I/O performance and reduces the efficiency of internal write buffers [4]. Therefore, SSDs employ the power-loss-protection (PLP) logic, which safely writes data from the internal write buffer to NAND flash memory using back-up power of SSD-internal capacitors in the event of a sudden power-off. It is mainly used for enterprise SSDs in data centers that require high reliability [5].

However, the existing PLP method has two limitations. The first is that, as the capacity of the SSD increases, the capacity of the internal write buffer also increases, but the number of capacitors cannot be increased infinitely due to the limitation of the space inside the SSD and the unstable price of capacitor material (Tantalum) [6,7]. The second is that in the existing block device interface it is not possible to distinguish data that require persistence in the internal buffer. Therefore, to achieve high reliability, persistence must be guaranteed for the entire area of the internal buffer in case of power failure, which greatly increases the required capacity of the capacitor.

The SSD internal buffer is used to store not only user data but also meta information of the FTL (Flash Translation Layer), such as a mapping table. In particular, legacy SSDs typically consume most of the in-storage DRAM for a page-level mapping table. Therefore, existing studies have mainly focused on minimizing the capacitance required to ensure the persistence of the mapping table in the event of a sudden power-off. SpartanSSD [6], the latest study on PLP, reduced the amount of the capacitance dramatically by recording a journal for mapping table updates and guaranteeing persistence only for the mapping table journal in case of sudden power-off. On the other hand, since the size of user data is negligible compared to the mapping table, it was not seriously considered in previous studies.

However, in ZNS SSDs [8,9], the amount of capacitance required for PLP mainly depends on the size of internal write buffer because of two reasons as follows. First, unlike conventional block-interface SSDs, ZNS SSDs do not need to retain a large internal mapping table in SSD internal DRAM. ZNS (Zoned Namespace) is a new storage interface that divides NAND flash memory into fixed-size zones that must be sequentially written. Additionally, because the ZNS interface exposes the zone to the host, the SSD internal mapping table is not required. Therefore, ZNS SSDs can utilize the DRAM buffer as a large write buffer for user data instead of using it to store the mapping table. The second reason is that ZNS SSDs need a much larger write buffer than legacy SSDs to support many I/O streams. The ZNS interface supports performance isolation and minimizes garbage collection overhead by allowing the host to allocate separate zones to different I/O streams. However, because of the size difference between the program unit of NAND flash (typically, multiple NAND pages aggregated across several NAND planes to maximize internal parallelism) and the page size of the host (typically, 4 KB), host writes should be buffered before NAND write. Therefore, each stream needs its own buffer space for write buffering [7]. Recently, ZNS SSDs have been required to support more than thousands of I/O streams. As SSD capacity increases, the number of multi-streams also increases, so ZNS SSDs will require larger write buffers in the future. We need to consider that NVMe can support 65 K streams and write unit size of NAND flash will increase to maximize the internal parallelism of SSDs. Therefore, we need to reduce the amount of capacitance while ensuring durability of write buffer, because capacitors cannot be added indefinitely due to the limitation of SSD internal space and the cost. Although there have been studies on reducing the number of capacitors by limiting the size of dirty data in the write buffer, the efficiency of the write buffer can be reduced due to frequent flushing [10].

Therefore, in this paper, we propose a set of techniques that can minimize the amount of capacitance required for PLP while retaining data integrity during sudden power-off. At first, we propose the selective power-loss-protection method, which selectively guarantees the persistence of user data in the SSD internal buffer. This technique was based on the following observation: most file systems or databases have their own way of ensuring data

integrity, such as WAL (Write-Ahead Log) and journaling techniques [11,12]. Therefore, it is sufficient to ensure the durability of the journal or WAL stored in the volatile buffer for data recovery. So, we modified the ZNS (Zoned Namespace) write command to deliver PLP-enable request for specific data to an SSD. The proposed selective-PLP was implemented by modifying FEMU (QEMU-based Flash Emulator) and evaluated using RocksDB. The experimental results show that it reduced the amount of SSD-internal capacitance by 50 to 90%. The second method is to shorten the PLP execution time by writing the user data in the SSD internal buffer to the NAND flash memory in parallel as much as possible. For ZNS SSD, because the host determines the zone to which data is written, the data in the write buffer must be recorded in the pre-designated zone during PLP. However, when write requests are concentrated in a specific NAND plane, the PLP latency increases, which means that the SSD-internal capacitance should be increased. Therefore, in this paper, we propose a technique for temporarily recording buffered data in multiple planes to maximize parallelism. Here, the temporarily recorded data is copied to the original location when power is restored. This technique was also implemented by modifying the FEMU, and according to experimental results, the PLP latency was reduced by up to 96%.

The rest of the paper is organized as follows. Section 2 briefly describes the related background topics, such as a ZNS SSD and PLP. Section 3 reviews previous studies on supporting battery-backed durable cache to the SSD and reducing the number of capacitors required for PLP. Section 4 presents the selective-PLP method to ensure the durability only for WAL and the balanced flush method to write buffered data to multiple NAND planes in parallel to minimize the latency of PLP. In Section 5, we evaluate the proposed methods with FEMU and RocksDB. Finally, we conclude this paper in Section 6.

2. Background

In this section, we introduce some backgrounds related to our paper. Note that some important abbreviations are explained in Abbreviations part.

2.1. Conventional SSDs vs. ZNS SSDs

NAND flash memory-based SSDs achieve a significant performance improvement compared to conventional HDDs (hard disk drives) [13]. However, because of erase-before-write constraints of NAND flash memory, SSDs employ firmware-level software FTL, which manages the L2P(Logical-to-Physical) mapping table to support out-place updates and garbage collection. As a result, a host can use an SSD as traditional block device through FTL [14,15]. However, because the erase unit (i.e., NAND block) is much larger than read/write unit (i.e., NAND page) in NAND flash memory, SSDs suffer from write amplification caused by valid page copies during garbage collection. Moreover, this can be worsened because SSD cannot receive any useful information from the host, allowing it to differentiate between hot and cold data. Therefore, host-managed SSD interfaces such as ZNS [8] and open-channel SSD [16] get attraction because they allow the host to perform data placement to minimize the garbage collection overhead.

In particular, the ZNS is an emerging storage interface that was recently defined by NVMe (NVM Express) specification [17]. The ZNS SSD divides NAND flash memory into fixed-size zones, which must be written sequentially(Figure 1) [9]. Moreover, zones are exposed to the host, allowing the host to control the data placement on SSDs. Consequently, the ZNS SSD has several advantages compared to conventional SSDs. At first, the ZNS SSD does not require an SSD internal mapping table, whereas the legacy block-interface SSDs require a large mapping table, typically equivalent to 0.1% of the storage capacity. Therefore, the ZNS SSD can utilize SSD internal DRAM as a large write buffer instead of the internal mapping table. This improves the SSD's I/O performance and lifespan by reducing the number of NAND writes [18]. Second, the garbage collection overhead can be reduced dramatically by data placement considering data hotness and I/O stream. On the other hand, a user-level library (e.g., libzbd) or ZNS-aware file system (e.g., zonefs, F2FS and BtrFS) is required to manipulate ZNS SSDs [9]. For example, RocksDB, a representative

key-value store, can use ZNS SSDs through ZenFS, which is a file system plugin for RocksDB developed by Western Digital [19].

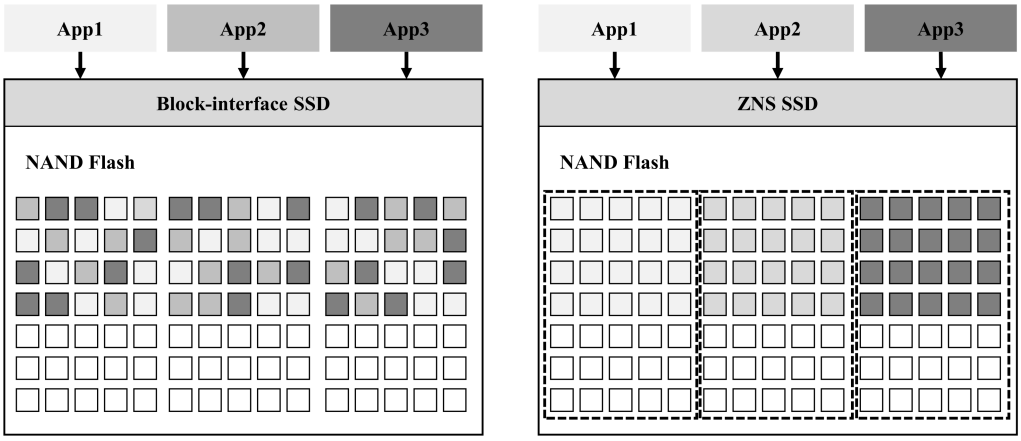

Figure 1. Comparison of a block-interface SSD and a ZNS SSD.

2.2. Power-Loss Protection

Most SSDs have a DRAM-based volatile buffer to improve I/O performance and expand the lifespan of SSDs. However, a volatile buffer does not guarantee the durability of the data during sudden power-off [2,3]. Therefore, enterprise SSDs employ power-loss protection, which writes all buffered data to the NAND flash memory by consuming back-up power of the capacitors [20]. As you can see in Figure 2, power supply is switched from the external power source to the internal capacitor bank when a power-loss event is detected. Note that the amount of the capacitors required for the PLP is proportional to the size of the volatile buffer [6,7,18].

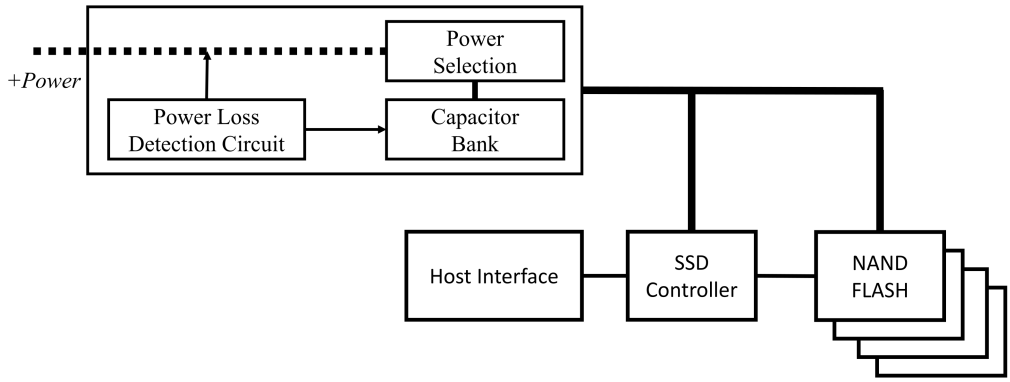

Figure 2. Block diagram of power-loss protection logic.

Typically, SSDs use an internal buffer to store not only user data but also meta data for FTL. However, the conventional block-interface SSD consumes most of volatile buffer to store a page-level mapping table with a size of 0.1% of the storage capacity. Therefore, the conventional SSDs use more than 97% of the DRAM buffer for the mapping table, while only a few megabytes can be used as a user data buffer [6]. As a result, the amount of capacitance required for PLP depends primarily on the size of the mapping table, not the size of the write buffer.

In other words, ZNS SSDs do not need to be concerned with the capacitance for guaranteeing the durability of mapping table because it does not use the internal mapping table. Instead, ZNS SSDs need a much larger size of write buffer to support many I/O streams. In multi-tenant systems, ZNS SSDs can provide performance isolation between different I/O streams and minimize garbage collection overhead by allowing the host to allocate separate zones to each I/O streams. However, because of the size difference

between the program unit of NAND flash (typically, multiple NAND pages aggregated across several NAND planes to maximize internal parallelism) and the page size of the host (typically, 4 KB), host writes should be buffered before NAND write. Therefore, each I/O stream needs its own buffer space for write buffering. Note that each ZNS SSD defines *maximum active zones*, which determines the maximum number of I/O streams that can be used simultaneously. The *maximum active zones* depends on the size of write buffer and back-up capacitance. In addition, as the SSD capacity increases, the maximum active zone will increase, so the demand for the write buffer of the ZNS SSD will also increase in the future. However, because the back-up capacitance cannot increase indefinitely due to the limitation of SSD internal space and the cost, it is worth finding a way to reduce the amount of capacitance while ensuring the durability of the write buffer.

2.3. RocksDB and ZenFS

RocksDB is an LSM tree-based key-value engine [12]. RocksDB uses a write buffer implemented as a Skip List called MemTable. Write buffers first try to improve performance by buffering I/O going to storage. All writes are written to the MemTable in memory and the WAL in a storage. When the size of the MemTable reaches a threshold, it is written to a storage in the format of a Sorted String Table (SST) file. Here, the WAL is created for data recovery in case of sudden power-off, and when the SST file is safely written to the SSD, it is deleted or transferred to other persistent storage.

ZenFS is an on-disk file system for Zoned Block Devices that comes as a plugin to RocksDB. It is a file system for transferring FTL functions to the host and uses ZNS SSDs operating in user space [8]. ZenFS uses journal zones and data zones to manage file systems. The journal zone stores data on maintaining ZenFS file system, such as Super Block Extent Map, and the data zone stores file contents.

3. Related Work

DuraSSD [18] proposed a battery-backed buffer cache to ensure atomicity of writes to the SSD. However, the traditional database engines use the fsync() system call to achieve write endurance, and SSD devices store cache as storage on command. In this process, typical database engines stop working while fsync() is being processed, which incurs a significant overhead in latency. Thus, DuraSSD achieved the performance improvement by eliminating flush commands from the file system with battery-backed durable cache.

Additionally, for fast writing, only the dirty data in the buffer are written. Since the buffer pool and in-memory data structures are small, all the data are written to the dump area and the abnormal end state is saved for recovery. Since the dump area must be created quickly in the event of a power loss, garbage collection is not performed, so it always remains available. When a system is rebooted after sudden power-off, the recovery process proceeds while charging the capacitors. If a power-loss situation is recognized by checking an abnormal shutdown state during the booting process, the recovery manager is called to recover the data stored in the dump area.

Viyojit [10] aims to reduce the correlation between DRAM capacity and capacitor capacity by exploiting the distortion in the application working configuration to reduce the amount of capacitor required. To reduce the flush operation time, this study focused on only writing dirty pages in an actual power-loss situation. To prove this, the actual workload of Microsoft's Azure blob storage, Cosmos, Page rank, and Search index serving production were analyzed. As a result, it was confirmed that only about 15% of the data of the entire file system were written within 1 h. The write-pattern analysis shows that the ratio of writes was small, but most of the writes consumed unique pages, or the ratio of writes was high and most of the unique pages were written for the overall workload. The study addresses this issue by limiting the number of dirty pages in the DRAM buffer in SSDs. Here, the dirty pages are managed using a list sorted by update order, similarly to the LRU. When the number of dirty pages reaches a threshold due to a high number of

writes, the oldest dirty pages in the buffer are written to storage and removed from the dirty page list.

SpartanSSD [6] is one of the most recent studies to reduce the capacitance for PLP of SSD. In this paper, the authors proposed a technique that guarantees durability only for the information necessary for the mapping table recovery in sudden power-off recovery (SPOR) instead of the entire mapping table. According to the paper, a technique called *elastic journaling* was inspired by journaling or write-ahead logging used in filesystems or databases. SpartanSSD writes an update log of the mapping table in the journal area, which is maintained in a battery-backed part of the internal DRAM buffer to ensure durability. When the journal area is full, all the mapping table entries in the volatile memory are flushed to NAND flash memory and the journal area is reclaimed. Because SpartanSSD needs to guarantee durability only for the mapping table journal, not the entire mapping table, the required capacitance for PLP can be reduced by more than 97%. However, ZNS SSDs cannot enjoy the capacitance reduction effect of *elastic journaling* because most of the internal DRAM buffer is used for user data. Therefore, in this paper, we propose a method to reduce the capacitance required for PLP while maintaining data integrity.

4. Selective Power-Loss Protection

4.1. Motivation

As mentioned above, conventional SSDs allocate most of their DRAM buffer to storing the mapping table. Therefore, the previous study focused on minimizing the capacitance required for maintaining durability of mapping table. However, in ZNS SSDs, the total capacitance required for PLP depends on the amount of user data in the volatile memory buffer, because ZNS SSDs do not use an internal mapping table. We are motivated by the fact that most filesystems and databases guarantee data integrity through their own recovery scheme, such as WAL and journaling. This means that there is no need to write the entire user data in the volatile memory buffer to the NAND flash memory in the event of sudden power-off. Even if the user data in the volatile memory buffer are lost during a sudden power-off, they can be recovered by using WAL or journal during SPOR. Therefore, in this section, we present a selective power-loss-protection method that guarantees durability only for the WAL or journal in the event of sudden power-off. As you can see in Figure 3, the proposed method provides new PLP interface and PLP-support buffer management policy for ZNS SSD. Because of the WAL or the journal being much smaller than other user data, the capacitance required to persist with them during sudden power-off can be significantly reduced in comparison to the previous full-protection PLP.

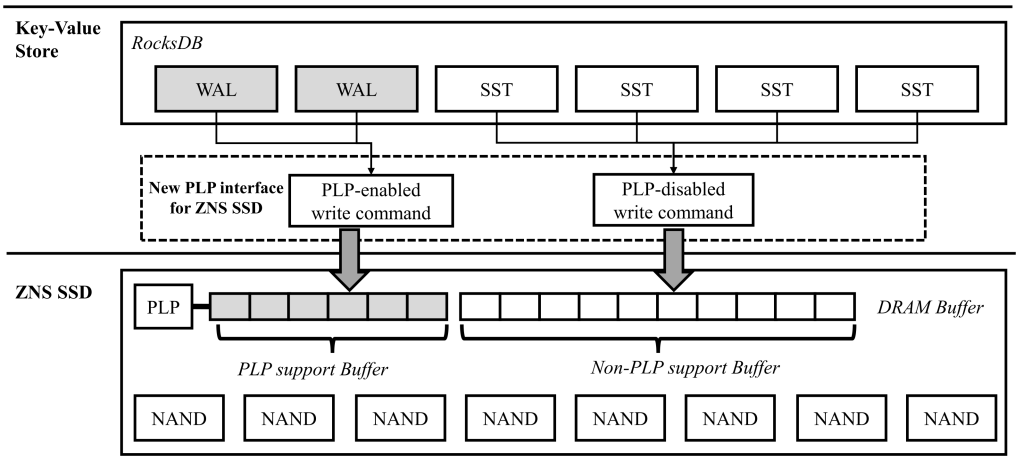

Figure 3. Overview of selective power-loss protection.

4.2. Selective Power-Loss-Protection (Selective-PLP) Interface

To ensure durability of the WAL or journal during sudden power-off, they should be separated from the user data. Therefore, we present a new write interface to transfer a PLP

flag with a write command. Here, the PLP flag indicates whether the data requested to be written needs to be durable during a sudden power-off. The latest NVMe specification supports PMR (Persistent Memory Region), persistent memory which is located on the NVMe SSD. According to the specification, the PMR feature enables the host to access persistent memory such as PRAM (Phase-change Random Access Memory) and MRAM (Magnetic RAM) or battery-backed DRAM installed in NVMe SSD via memory-mapped I/O. Therefore, we can consider that NVMe SSDs selectively guarantee the durability of user data by writing a journal or WAL to the PMR of NVMe. However, because the existing databases or filesystems usually record the WAL or journal through file I/O, replacing them with memory reads/writes to the PMR might require extensive code modifications. Moreover, since the correlation between PMR and zones is not yet clearly defined in the NVMe specification, implementation using PMR may be tricky. Therefore, to achieve an intuitive and simple implementation, we utilized the unused bit of the ZNS write command to transmit the PLP flag to ZNS SSDs. Moreover, we can customize a vendor-specific NVMe admin command to request durability for specific write requests. Because NVMe standard allows customizing NVMe admin commands, it may be a more practical way to apply the proposed Selective-PLP to the real products. However, in the case of using the additional admin command, the additional overhead should be evaluated. Therefore, in this paper, we focused on verifying our idea with the proof of concept implemented by utilizing an unused bit of existing write command.

We implemented a new write interface PLP_write(), which the host application will use to pass the PLP flag with the ZNS write command instead of pwrite(). As shown in Figure 4, PLP_write() uses ioctl() to deliver PLP flag via the existing NVMe ZNS write command [17]. Command Dword3 (CDW3), which is not used in the existing NVMe write command, is used to represent PLP flag. Note that durability of the write request is guaranteed if it is 1, otherwise it is not guaranteed. When a ZNS SSD receives a write request, CDW3 is checked to determine whether durability should be guaranteed. In other words, if PLP flag is 1, the user data is stored in the PLP-supported durable buffer. The proposed interface is implemented by using emulated ZNS SSD supported by FEMU [21], an NVMe SSD emulator. Figure 4 describes how the PLP_write() is handled through ZNS emulated by FEMU.

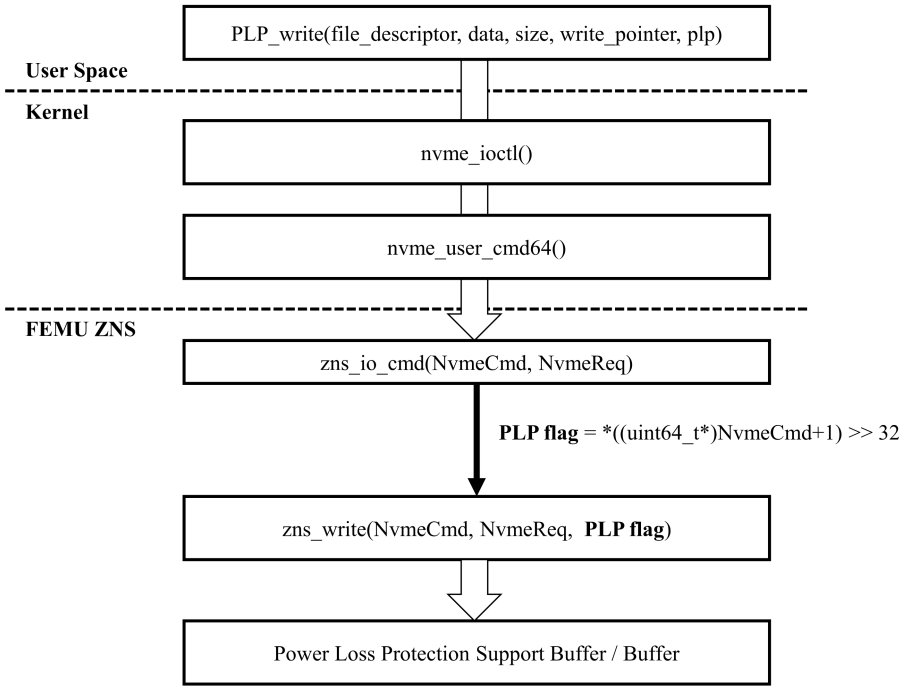

Figure 4. Flowchart of PLP_write() command operation in FEMU ZNS.

4.3. Buffer Management Scheme for Selective-PLP

In this section, we present the buffer management scheme of the selective-PLP. As shown in Figure 5, to support the selective-PLP, the DRAM buffer should be divided into PLP-support buffer region(PLP buffer) and non-PLP-support buffer region(non-PLP buffer). Moreover, because the host determines the data placement in ZNS SSDs, the zone associated with each buffered data has already been determined. Therefore, it is necessary to manage the buffer by zone. To manage buffer space, the *buffer hit table* and the *buffer list* are employed(described in Figure 6). At first, the *buffer hit table* maintains a reference bit for each sector because the LRU approximation algorithm based on a reference bit is used for buffer space management. Therefore, when a read command is received from the host, the ZNS SSD first scans the *buffer hit table* to find the requested sector with sector number. Additionally, buffer units associated with each zone are maintained in a separated list. Second, the *buffer list* is a data structure used to maintain the buffers associated with each zone. As mentioned above, because the buffered data should be written sequentially to the associated zone, it is efficient to maintain buffers per associated zone.

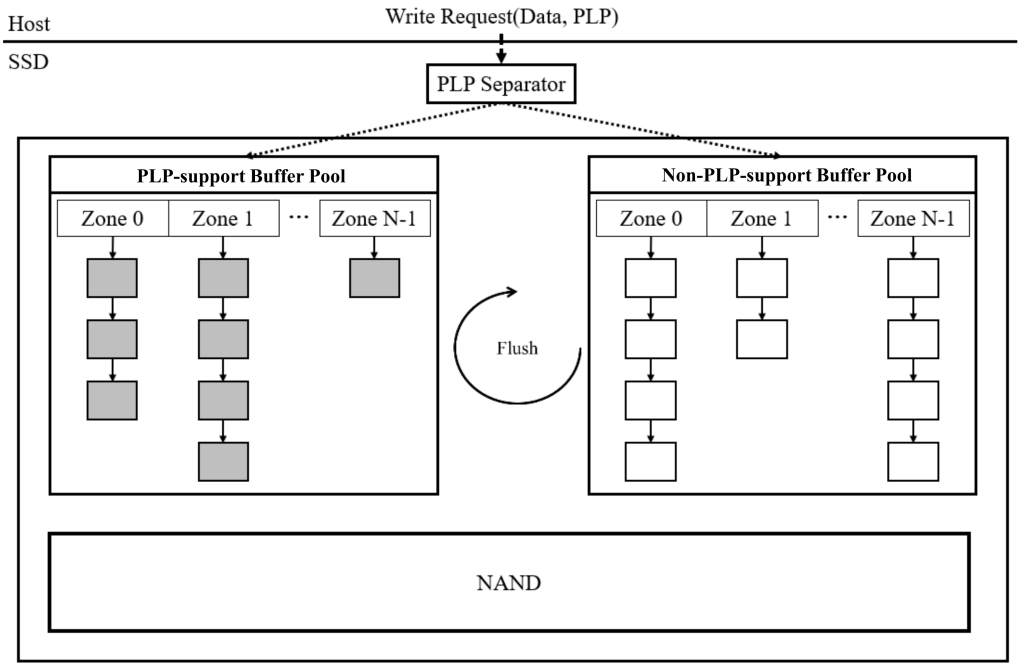

Figure 5. ZNS SSD buffer management scheme for selective power-loss protection.

In addition, we modified the operation of the flush() command issued from the host for the PLP buffer based on the idea of DuraSSD [18], which improved performance by removing the flush of the battery-backed durable cache. Essentially, the host calls the flush() command to ensure the durability of all the data in the volatile memory buffer. However, in the case of the PLP buffer, durability is already guaranteed due to the pre-loaded capacitors. Therefore, the number of NAND writes for the PLP buffer can be reduced by ignoring the flush() command for the PLP buffer. Moreover, because the flushing buffer means that new incoming requests are suspended until the flush operation is finished to ensure write ordering, the I/O performance significantly decreases. Therefore, the I/O performance is significantly improved by ignoring the flush() command for the PLP buffer. In addition, if the buffer usage exceeds 70% of the total capacity of each buffer, all the buffered data are written to the NAND flash memory. Note that the threshold of the buffer usage is a configurable parameter.

The buffer flush is implemented in two ways. First is to write the data in the buffer to the NAND flash memory, and the other is to discard the data without writing. The former is used when the buffer usage exceeds a predetermined threshold, or when a flush() command is issued by the host. The latter is used for the zones that are already reset. This

means that all data contained in the zone are invalid due to deletion or out-place updates. Therefore, we can omit writing invalid data of the reset zone to the NAND flash memory.

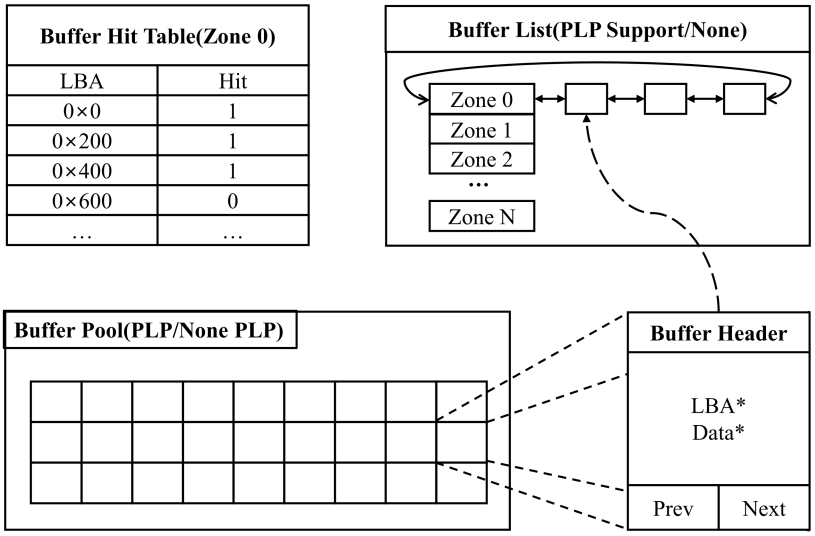

Figure 6. Buffer hit table and Buffer list for managing PLP and Non-PLP buffer.

The overall buffer management scheme consists of two steps. First, a buffer pool is created during the ZNS SSD initialization process. The buffer pool is composed of sector-sized buffers, and each buffer has its own metadata structure, a buffer header. Through this, the overhead of buffer allocation is reduced because the frequent memory allocation and deallocation operations can be omitted by using a pre-allocation buffer pool. Second, the buffer allocation and return process is as follows. When a buffer is requested for a new write request, a free buffer is allocated from the buffer pool. After that, data are written to the allocated buffer, and the buffer is inserted into the PLP or the non-PLP *buffer list* according to the PLP flag of the write request. If the buffer is used up, the buffer is removed from the *buffer list* and returned to the buffer pool.

4.4. ZenFS Modification for Selective-PLP

To apply selective-PLP to ZenFS of RocksDB, we modified ZenFS as follows. First, zones are divided into two categories: durable zones and normal zones. Here, a durable zone indicates that data written into the zone should be guaranteed to be durable by using PLP buffer. As shown in Figure 7, we have two kinds of durable zone: *journal zone* and *WAL zone*. A *journal zone* is used for metadata of ZenFS, such as Super Block, Zone Extent Mapping Table, and Extent Management. A *WAL zone* is used for write-ahead log of RocksDB. On the other side, data written to normal zones are buffered in non-PLP buffer. STT files of RocksDB, which have key-value data, are buffered in non-PLP buffer. Therefore, in a sudden power-off, key-value data in the non-PLP buffer cannot have guaranteed durability. However, because the WAL in the PLP buffer is guaranteed a durability, RocksDB can recover the lost key-value data during sudden power-off recovery (SPOR) process.

The existing ZenFS stores files with a similar life cycle in the same zone. Therefore, to separate durable zones and normal zones, we added a criterion to distinguish whether it is a WAL file or ZenFS metadata. As a result, WAL file and ZenFS metadata are simply classified and allocated to the durable zone. Moreover, it was confirmed that ZenFS uses two zones for WAL. Additionally, because WAL is deleted after key-value data is normally written, NAND write for WAL can be eliminated by using PLP buffer with a size of two zones or more. In fact, in the experimental environment, since the size of a zone is 16 MB, only the 32 MB PLP buffer is enough to store WAL of RocksDB. Since the two WAL zones are written alternately, there is never a case where both have no free space simultaneously.

As a result, in our experiments, a PLP buffer of size 32 MB is sufficient to ensure durability for WAL as well as metadata for ZenFS.

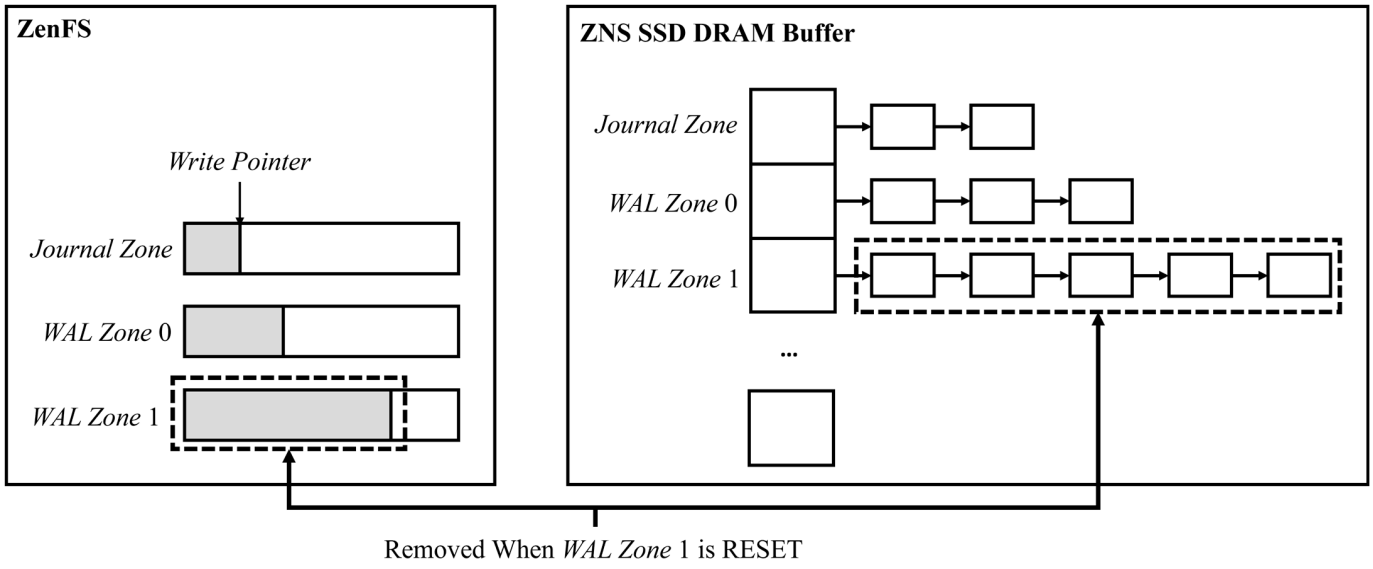

Figure 7. ZenFS optimization storing WAL and metadata in the PLP buffer.

4.5. Balanced Flush to Minimize Flush Latency

Due to the limited capacity of the capacitor, the buffered data in PLP buffer needs to be flushed quickly to NAND flash in a sudden power-loss situation. However, as you can see in Figure 8, in a power-loss situation, the time taken for flushing increases if the physical addresses of the buffered data are concentrated in a specific NAND plane. To solve this problem, we propose a balanced flush method to minimize the time required for flushing PLP buffer to NAND flash.

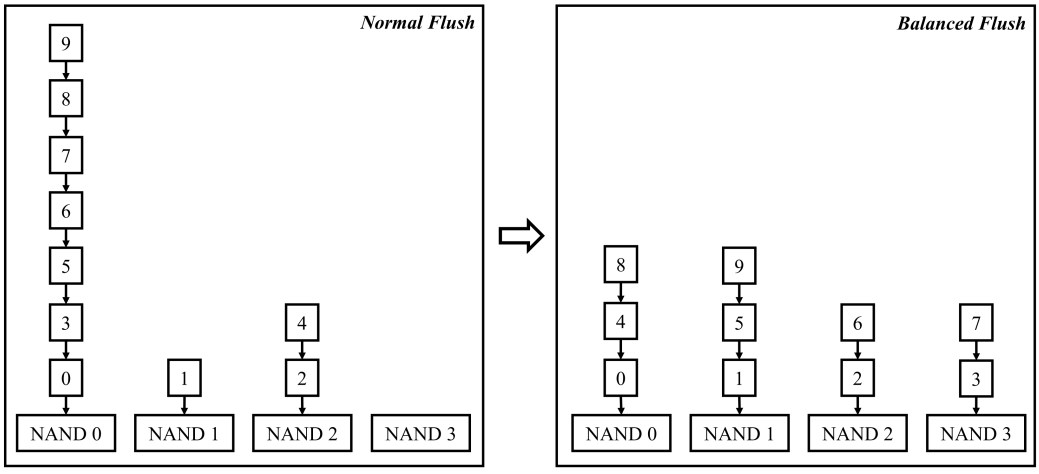

Figure 8. Comparison of normal and balanced flush in case of power loss.

As described in Figure 8, the balanced flush temporarily writes the buffered data in multiple NAND planes to maximize parallelism. Here, the temporarily recorded data is copied to the original location when power is restored. The original address of the buffered data is stored in the OOB (Out-Of-Band) area of NAND page. As described in Figure 9, each physical page of NAND flash memory has an OOB (Out-Of-Band) area [22]. OOB can be used to store an additional information such as Bad Block Marking and ECC (Error Correction Codes), and is mainly used for error checking or recovery [23]. Since OOB has sufficient space (typically, 1/32 of data area of NAND page) and no additional write latency,

it is suitable for storing original address of buffered data. In the event of sudden power-off, the data in the PLP buffer is flushed to the NAND chip, which currently shows the lowest latency, not the original location specified by the host. Then, the original address is stored in the OOB area for data restoring. In SPOR, the OOB area is checked to determine whether the page is written by the balanced flush, then the page is restored to original location by using the address stored in OOB area.

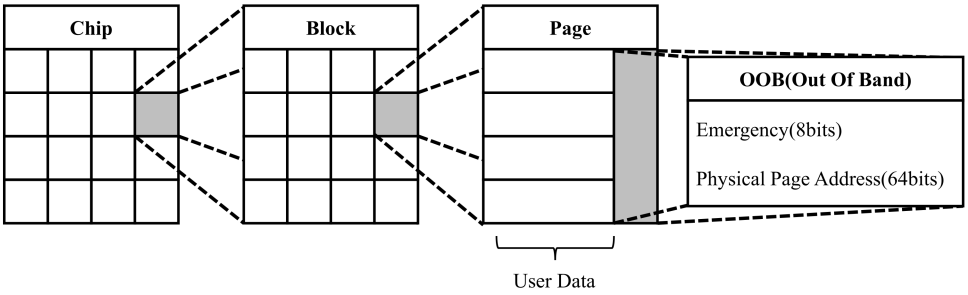

Figure 9. OOB area in NAND flash memory.

Since FEMU [21], the ZNS SSD emulator, operates based on memory, all data are removed when the actual emulator is shut down. Therefore, the proposed balanced flush method was verified by randomly generated sudden power-off event. First, when a sudden power-off event occurs at any point in time, locks are acquired for all buffer units. After that, all read commands are ignored while only write commands for the data in the PLP buffer are processed. After making two copies of the PLP buffer, they are used for the normal flush and the balanced flush, respectively. Moreover, since the delay time model of FEMU must also not interfere with each measurement, the delay time model is also copied for calculating flush latency. In verification, the data in the copied buffer are applied to the copied latency model to return the oldest timeline of each chip. After that, compared with the oldest timeline just before copying, the time equal to the difference is calculated as the time taken for flushing.

5. Results and Discussion

5.1. Experimental Setup

The experiment is conducted in FEMU [21], a ZNS SSD emulator, and the experimental environment for the host and the virtual machine is shown in Table 1. To use the ZNS SSD-enabled virtual machine, we used kernel version 5.10, and allocated 32 GB of memory to the FEMU virtual machine and 8 GB of memory to the ZNS SSD. The FEMU virtual machine uses memory as a backend storage for its operating system and the emulated ZNS SSD. The existing FEMU ZNS does not consider channel- and chip-level parallelism. Therefore, we adopted the latency model of existing FEMU black-box mode to FEMU ZNS because it already supports channel- and chip-level parallelism. Note that FEMU black-box mode is used to emulate a conventional block-interface NVMe SSD. In the latency model, each channel and chip maintain their own timeline of the most recent I/O request completed. Therefore, when a new I/O request arrives in a particular NAND chip, it calculates the I/O completion time considering currently processing I/O requests in the NAND chip. The latency for each NAND operation and configuration of the emulated ZNS SSD are described in Table 2. The experiment was conducted while varying the size of the DRAM write buffer from 64 MB to 512 MB. Note that the size of the PLP buffer is fixed at 32 MB because only twice the size of the WAL file is required, as mentioned in Section 4.4.

Table 1. Host and Virtual Machine Specifications.

Classification	Component	Specification
Host	CPU	Intel Xeon CPU E5-2620 v4 × 2 (16 cores)
	OS	Ubuntu 18.04.5 LTS (kernel 4.15)
	Memory	Samsung DDR4 2400 Mbps 16 GB × 4
Virtual Machine	OS	Ubuntu 20.04.1 LTS (kernel 5.10)
	Memory	32 GB
	Storage	80 GB

Table 2. ZNS SSD Specifications.

Component	Specification
Page Size/Block Size	4 KB/4 MB
Number of Channel	8
Number of Chips per Channel	8
Read/Write Delay	40 us/100 us
Data Transfer Delay	40 us
Capacity	8 GB
Zone Size	16 MB
Write Buffer Size	64–512 MB

The benchmark uses the *db_bench* benchmark provided by RocksDB, and the workload settings are shown in Table 3. Here, the *fillseq* workload writes a specified number of key-value pairs in key order. Furthermore, the *overwrite* workload must perform the fillseq workload first, followed by overwriting key-value pairs in random key order.

Table 3. *db_bench* Workload Configuration.

Component	Specification
Number of Keys	3,000,000
Key Size/Value Size	100 bytes/800 bytes
write_buffer_size	8 MB
target_file_size_base	8 MB
benchmark	fillseq, overwrite

5.2. I/O Performance with Varying Buffer Size

In this section, we evaluate the I/O performance of the proposed Selective-PLP. As mentioned in Section 4.3, because flush() command is ignored for PLP buffer, I/O performance can be improved. Therefore, we compared the I/O performance of three different versions of ZNS SSD as follows: None-PLP(*None*), Full-PLP(*Full*), Selective-PLP(*Selective*). None-PLP is ZNS SSD without PLP, so any flush() command cannot be ignored. Full-PLP is ZNS SSD, which has sufficient capacitance to support PLP for the entire volatile DRAM buffer. Therefore, in the Full-PLP, every flush() command issued by the host can be ignored. Here, Full-PLP represents current commercial enterprise SSDs. Finally, Selective-PLP supports PLP only for the durable buffer in which WAL and ZenFS metadata are stored.

Figure 10 shows the I/O bandwidth measured with *fillseq* workload of *db_bench*. According to Figure 10, Selective-PLP improves the I/O performance by an average of 15% compared to None-PLP, while showing comparable performance to Full-PLP. As mentioned above, because the PLP buffer is fixed at 32 MB and capacitance required for PLP is proportional to the amount of buffered data, Selective-PLP achieves performance improvement comparable to Full-PLP with only 6–50% of capacitance of Full-PLP. These results mean that the same level of reliability and performance improvement as Full-PLP can be achieved with much less capacitance by selectively supporting PLP only for data essential for data recovery such as WAL or journal. Figure 11 shows the results of the

overwrite workload of *db_bench*. According to the figure, Selective-PLP shows that the I/O performance is improved by up to 6.5% and 8.7% compared to None-PLP and Full-PLP, respectively. Furthermore, we can see that the I/O performance improvement are reduced in the *overwrite* workload compared to *fillseq*. The reason is that the effect of the PLP support buffer is reduced due to increasing compaction overhead of RocksDB incurred by random key-value updates.

To evaluate the effect of PLP buffer size on RocksDB, we measured the performance of RocksDB with *fillseq* workload by varying the size of the PLP buffer. In the Selective-PLP, the size of the non-PLP buffer was fixed at 32 MB and the size of the PLP buffer was increased from 8 MB to 128 MB. Figure 12 shows the performance evaluation results for varying PLP buffer size. According to the figure, the performance of RocksDB increases as the PLP buffer size increases, but there is no further improvement in the performance of RocksDB beyond 32 MB. The reason is that PLP buffer of size 32 MB is sufficient to store two zones used for WAL of RocksDB.

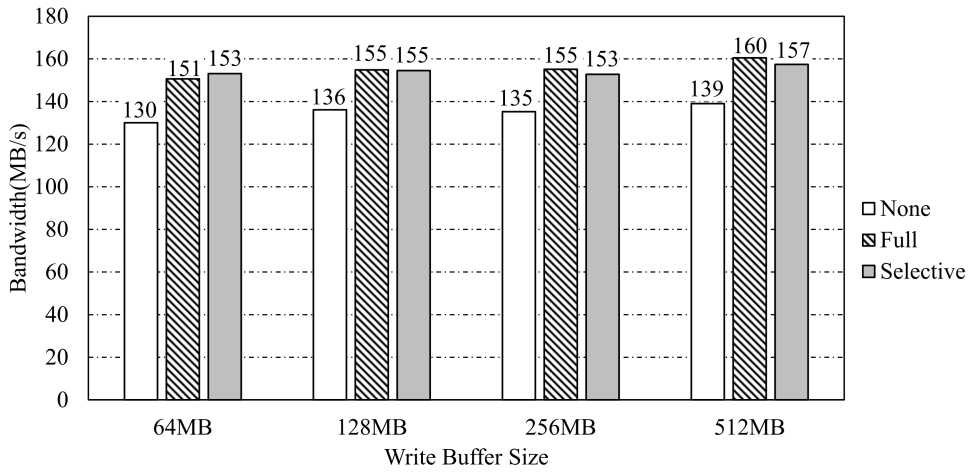

Figure 10. Performance evaluation results of *fillseq* workload with varying buffer size.

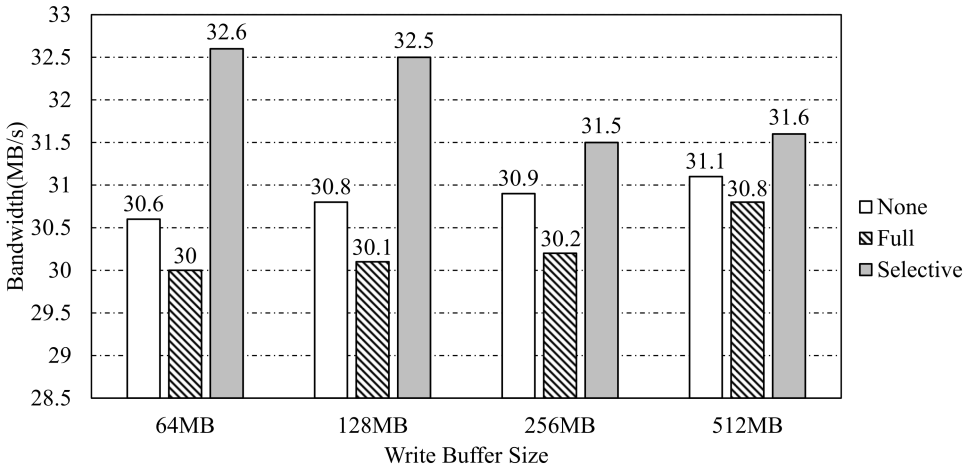

Figure 11. Performance evaluation results of *overwrite* workload with varying buffer size.

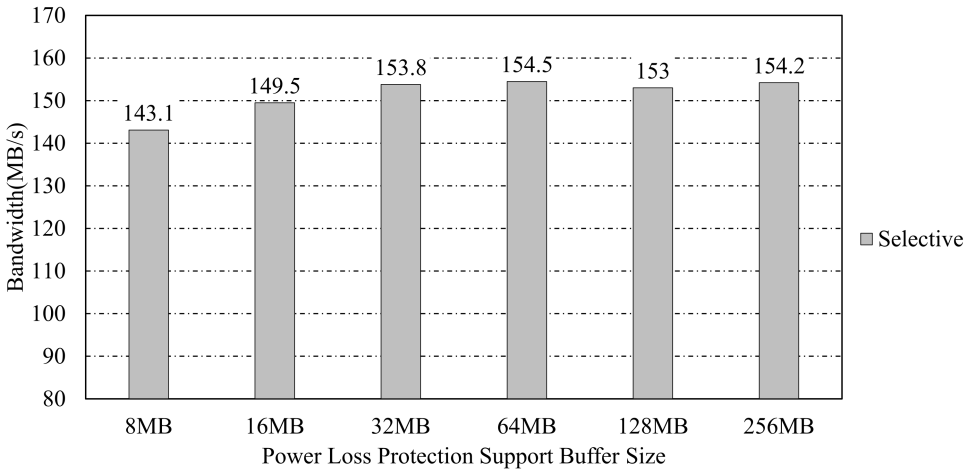

Figure 12. Performance evaluation results of *fillseq* workload with varying PLP buffer size.

5.3. Balanced Flush and Recovery

To evaluate the balanced flush described in Section 4.5, a power-loss event was randomly generated by using a background thread while performing *db_bench* workload. A random number was generated every 0.1 s to generate a power-loss event with a probability of 50%, and the time taken for PLP was measured. In the event of a power loss, all commands are stopped, and information on the current status of ZNS SSD, such as buffer and simulated latency value, is copied safely. Afterwards, data in the PLP buffer are written to the NAND flash memory by using the balanced flush or the normal flush, and the time taken for PLP is measured. It then resumes *db_bench* after performing a recovery process using the copied buffers and the simulated delay value.

We compared the latency of the normal flush and the balanced flush under sudden power-off by varying the size of the PLP buffer from 32 MB to 256 MB. The size of the non-PLP buffer is fixed to 32 MB. As you can see in Figure 13, the balanced flush decreased the time taken for PLP by about 96% on average compared to the normal flush. This is because intensive write requests for a specific NAND chip are distributed across multiple NAND chips, reducing overall latency.

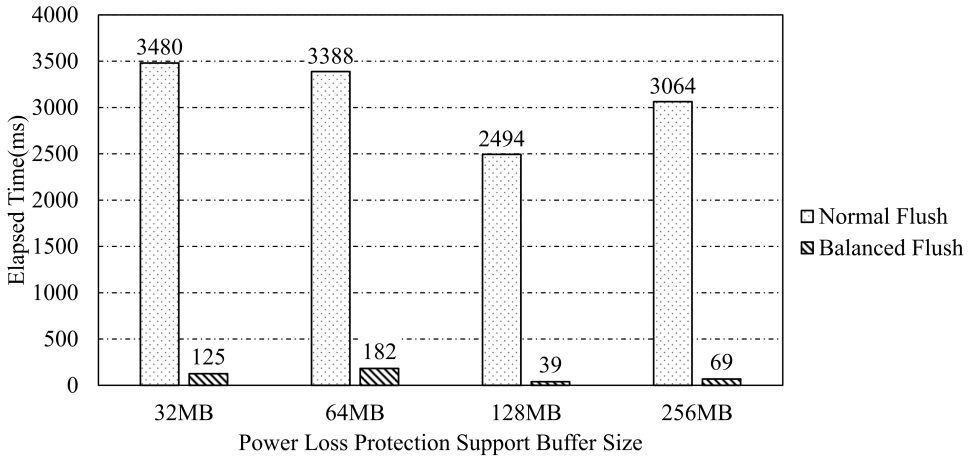

Figure 13. The time taken by PLP with varying PLP buffer size.

In the Section 5, we evaluated the proposed Selective-PLP by using emulated ZNS SSD and modified RocksDB. According to the experimental results, we can achieve the same reliability and performance as Full-PLP while reducing capacitance of ZNS SSDs by 6~50% by guaranteeing durability only for the WAL or journal required for data recovery. In addition, the balanced flush was evaluated by using a randomly generated sudden

power-off event. The experimental results show that the balanced flush can significantly reduce the PLP latency.

6. Conclusions

Most SSDs have a DRAM-based volatile buffer to improve I/O performance and expand the lifespan of SSDs by absorbing write requests. However, because a volatile buffer does not guarantee the durability of the buffered data during a sudden power-off, enterprise SSD employs power-loss-protection logic by using internal capacitors. However, as capacity and parallelism of SSDs increase, a lager write buffer becomes required. Especially, emerging ZNS SSDs require a much larger write buffer than legacy block-interface SSDs to support many I/O streams in multi-tenant systems because each stream needs its own buffer space for write buffering. However, the amount of capacitance required to ensure the durability of the write buffer cannot be increased infinitely due to the limitations of internal space and the cost. In this paper, we proposed a selective power-loss-protection method for ZNS SSDs to reduce the capacitance required for PLP. By using a new write interface, PLP_write(), that can deliver a PLP flag to ZNS SSD, the ZNS SSD can ensure the durability of the WAL or journal. Because file systems and databases have their own recovery mechanisms using WAL or journal, the Selective-PLP achieves the same level of reliability while dramatically reducing the amount of capacitance for PLP. Moreover, ZenFS of RocksDB is modified to guarantee durability for only WAL files of RocksDB by using PLP_write(). According to our evaluation results, the performance of RocksDB increased by an average of 15% compared to None-PLP due to ignoring the flush() command for the PLP buffer. In addition, the Selective-PLP achieves performance improvement comparable to Full-PLP with only 6–50% of the capacitance of Full-PLP. These experimental results indicate that the same level of reliability and performance improvement as Full-PLP can be achieved with much less capacitance by selectively supporting PLP only for data essential for data recovery, such as WAL or journal. Moreover, the balanced flush decreased the time taken for PLP by about 96% on average by distributing write requests across multiple NAND chips.

Author Contributions: Conceptualization, J.Y. and S.A.; Data curation, S.L.; Formal analysis, J.Y.; Funding acquisition, S.A.; Investigation, S.L.; Methodology, J.Y.; Project administration, S.A.; Resources, S.L.; Software, J.Y.; Supervision, S.A.; Validation, J.Y.; Writing—original draft, J.Y.; Writing—review & editing, S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (2019-0-01343, Regional strategic industry convergence security core talent training business) and the National Research Foundation of Korea (NRF) grant funded by the Korea government(MSIT) (No. NRF-2020R1A4A4079859).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

SSDs	Solid-State Drives
HDDs	Hard Disk Drives
PLP	Power-Loss Protection
SPOR	Sudden Power-Off Recovery
FTL	Flash Translation Layer
NVMe	NVM Express
ZNS	Zoned Namespace interface
PMR	Persistent Memory Region
WAL	Write-ahead Log
SST	Sorted String Table of RocksDB
OOB	Out-Of-Band area of NAND flash memory
ECC	Error Correction Codes

References

1. Kim, H.; Ahn, S. BPLRU: A Buffer Management Scheme for Improving Random Writes in Flash Storage. In Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST 08), San Jose, CA, USA, 26–29 February 2008; pp. 239–252.
2. Tseng, H.-W.; Grupp, L.; Swanson, S. Understanding the impact of power loss on flash memory. In Proceedings of the 48th Design Automation Conference (DAC 11), San Diego, CA, USA, 5−10 June 2011; pp. 35–40.
3. Zheng, M.; Tucek, J.; Qin, F.; Lillibridge, M. Understanding the robustness of SSDs under power fault. In Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST 13), San Jose, CA, USA, 12–15 February 2013; pp. 271–284.
4. Lee, T.H.; Lee, M.; Eom, Y.I. An insightful write buffer scheme for improving SSD performance in home cloud server. In Proceedings of the 2017 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 8–10 January 2017; pp. 164–165.
5. Gao, C.; Shi, L.; Li, Q.; Liu, K.; Xue, C.J.; Yang, J.; Zhang, Y. Aging Capacitor Supported Cache Management Scheme for Solid-State Drives. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2019**, *39*, 2230–2239. [CrossRef]
6. Lee, H.G.; Lee, J.W.; Kim, M.W.; Shin, D.H.; Lee, S.J.; Kim, B.S.; Lee, E.J.; Min, S.L. SpartanSSD: A reliable SSD under capacitance constraints. In Proceedings of the 2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), Boston, MA, USA, 26–28 July 2021; pp. 1–6.
7. Kim, T.; Hong, D.; Han, S.S.; Chun, M.; Lee, S.; Hwang, J.; Lee, J.; Kim, J. Fully Automatic Stream Management for Multi-Streamed SSDs Using Program Contexts. In Proceedings of the 17th USENIX Conference on File and Storage Technologies (FAST 19), Boston, MA, USA, 25–28 February 2019; pp. 295–308.
8. Bjørling, M.; Aghayev, A.; Holmberg, H.; Ramesh, A.; Moal, D.; Ganger, G.; Amvrosiadis, G. ZNS: Avoiding the Block Interface Tax for Flash-based SSDs. In Proceedings of the 2021 USENIX Annual Technical Conference (USENIX ATC 21), Virtual Event, 14–16 July 2021; pp. 689–703.
9. Zoned Namespaces (ZNS) SSDs. Available online: https://zonedstorage.io/docs/introduction (accessed on 1 August 2021).
10. Kateja, R.; Badam, A.; Govindan, S.; Sharma, B.; Ganger, G. Viyojit: Decoupling battery and DRAM capacities for battery-backed DRAM. *ACM SIGARCH Comput. Archit. News* **2017**, *45*, 613–626. [CrossRef]
11. Lee, E.; Bahn, H.; Noh, S.H. Unioning of the buffer cache and journaling layers with non-volatile memory. In Proceedings of the 11th USENIX Conference on File and Storage Technologies (FAST 13), San Jose, CA, USA, 12–15 February 2013; pp. 73–80.
12. Dong, S.; Kryczka, A.; Jin, Y.; Stumm, M. RocksDB: Evolution of Development Priorities in a Key-value Store Serving Large-scale Applications. *ACM Trans. Storage* **2021**, *17*, 1–32. [CrossRef]
13. Rizvi, S.S.; Chung, T.-S. Flash SSD vs. HDD: High performance oriented modern embedded and multimedia storage systems. In Proceedings of the 2010 2nd International Conference on Computer Engineering and Technology, Chengdu, China, 16–18 April 2010; Volume 7, pp. V7-297–V7-299.
14. Lee, S.-W.; Choi, W.-K.; Park, D.-J. FAST: An efficient flash translation layer for flash memory. In Proceedings of the International Conference on Embedded and Ubiquitous Computing (EUC 2006), Seoul, Korea, 1–4 August 2006; pp. 879–887.
15. Jung, D.; Kang, J.-U.; Jo, H.; Kim, J.-S.; Lee, J. Superblock FTL: A superblock-based flash translation layer with a hybrid address translation scheme. *ACM Trans. Embed. Comput. Syst.* **2010**, *9*, 1–41. [CrossRef]
16. Bjørling, M.; González, J.; Bonnet, P. Lightnvm: The linux open-channel SSD subsystem. In Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST 17), Santa Clara, CA, USA, 27 February–2 March 2017; pp. 359–374.
17. NVMe Zoned Namespace Command Set Specification. Available online: https://nvmexpress.org/wp-content/uploads/NVMe-Zoned-Namespace-Command-Set-Specification-1.1a-2021.07.26-Ratified.pdf (accessed on 1 August 2021).
18. Kang, W.; Lee, S.W.; Moon, B.; Kee, Y.S.; Oh, M. Durable write cache in flash memory SSD for relational and NoSQL databases. In Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD'14), Snowbird, UT, USA, 22–27 June 2014; pp. 529–540.
19. ZenFS. Available online: https://github.com/westerndigitalcorporation/zenfs (accessed on 1 March 2021).
20. Huang, M.; Wang, Y.; Qiao, L.; Liu, D.; Shao, Z. SmartBackup: An efficient and reliable backup strategy for solid state drives with backup capacitors. In Proceedings of the 2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC 15), New York, NY, USA, 24–26 August 2015; pp. 746–751.
21. Li, H.; Hao, M.; Tong, M.H.; Sundararaman, S.; Bjørling, M.; Gunawi, H.S. The CASE of FEMU: Cheap, accurate, scalable and extensible flash emulator. In Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST 18), Oakland, CA, USA, 12–15 February 2018; pp. 83–90.
22. Mativenga, R.; Hamandawana, P.; Chung, T.-S.; Kim, J. FTRM: A cache-based fault tolerant recovery mechanism for multi-channel flash devices. *Electronics* **2020**, *9*, 1581. [CrossRef]
23. Min, D.; Park, D.; Ahn, J.; Walker, R.; Lee, J.; Park, S.; Kim, Y. Amoeba: An autonomous backup and recovery SSD for ransomware attack defense. *IEEE Comput. Archit. Lett.* **2018**, *17*, 245–248. [CrossRef]