Optimizing Stream Separation Using Event Logs in FDP Storage Devices

Hyunbin Kang
Department of Computer Science and Engineering
Sungkyunkwan University
Suwon, Korea
khb84341@gmail.com

Dongkun Shin
Department of Computer Science and Engineering
Sungkyunkwan University
Suwon, Korea
dongkun@skku.edu

***Abstract*—The new Non-Volatile Memory express (NVMe) specification, Flexible Data Placement (FDP), introduces a method for improving storage efficiency and performance by reducing the write amplification and ensuring quality of service (QoS). FDP allows hosts to place data into the desired block by specifying the stream in write command. Hosts can reduce write amplification by placing data with similar lifetime in the same block. Additionally, FDP also ensures that each VM uses a different physical die group to minimize performance interference between VMs as much as possible. FDP also has four FDP-specific log pages, of which the Media Reallocated event in the FDP events log was used to experiment with reducing the WAF by reallocating streams by hosts that incorrectly predicted the lifetime of data. Experiments show that this method could significantly reduce the garbage collection cost to almost zero.**

***Index Terms*—SSD, NVMe, FDP, Event Log, Multi-Stream**

I. Introduction

Recently Non-Volatile Memory express (NVMe) standards introduced the Flexible Data Placement (FDP) [1] specification. FDP provides compatibility with existing conventional I/O and does not require significant modifications to the existing kernel stack. FDP targets a reduction in write amplification factor (WAF) by employing stream separation and enhances quality-of-service (QoS) by isolating physical dies. When creating a namespace with the Namespace Management command, the host can specify the number of Placement Handles (PHNDLs) that the namespace will use and the Reclaim Unit Handles (RUHs) that each PHNDL will use. A PHNDL serves as a stream designated by the host within that namespace, whereas an RUH represents a stream determined by the endurance group on the storage device.

During write operations, the host specifies a Placement ID (PID) within the write command, which is subsequently parsed by the FDP controller into a Reclaim Group (RG) and PHNDL. Then the PHNDL is transformed into a predefined RUH during namespace creation. The combination of the RG and RUH determines the Reclaim Unit (RU), where the data requested by the host is placed. RU is a superblock in the SSD which is a unit of erasure. By allocating data with similar update cycles to the same RU, hosts can effectively reduce WAF. Furthermore, FDP introduces RG, a group of multiple dies, to mitigate performance interference among VMs when a single storage device is utilized concurrently by multiple VMs.

FDP introduces two garbage collection (GC) schemes for each RUH. The Initially Isolated method allows the mixing of data moved by GC within the same RU, while Persistently Isolated ensures that data remains within the same stream during GC. With the appearance of FDP, four FDP-specific log pages were introduced: FDP configurations, RUH usage, FDP statistics, and FDP events. Among these, FDP events log comprises four host events and two controller events, recording various event types initiated by the host or controller within the endurance group, including information such as the timestamp of the event, namespace ID (NSID), PID, RGID, and RUHID. These details are stored in the device's internal ring buffer, which can hold up to 63 entries. Upon receiving a `get_log_page` command, the buffer's content is returned to the requester, although some information may not be provided depending on the controller.

Specifically, the Media Reallocated event which is one of the controller events provides additional information for an RU written by an Initially Isolated type of RUH, including the number of LBAs moved (NLBAM) and the address of one of these LBAs moved. As mentioned earlier, FDP allows hosts to improve WAF by placing data with similar lifetimes in the same RU via PID, and if the host incorrectly predicts the lifetime of the data, these Media Reallocated events can be leveraged to improve WAF. However, since the current NVMe standard says that the event can report only one LBA out of multiple LBAs moved by the GC, we need to report the most useful LBA for the host to utilize. With this purpose, we determined to report the first LBA address of the LBA chunk with the largest contiguous LBA area among the moved LBAs, which allowed us to reallocate the incorrectly predicted LBAs to other streams during runtime, significantly reducing the WAF during runtime.

II. GC Cost Reduction method Using FDP Events

The Media Reallocated event (Event type: 0x80) within the FDP events log (Log ID: 0x23) serves as a GC monitoring feature that provides hosts with real-time information related to physical page reallocations performed by the device controller. This feature enables hosts to consider repeatedly relocated LBAs as blocks with incorrectly predicted lifetimes and consider changes to other RUH. However, the current

979-8-3503-8855-8/24/$31.00 ©2024 IEEE
NVMSA.2024

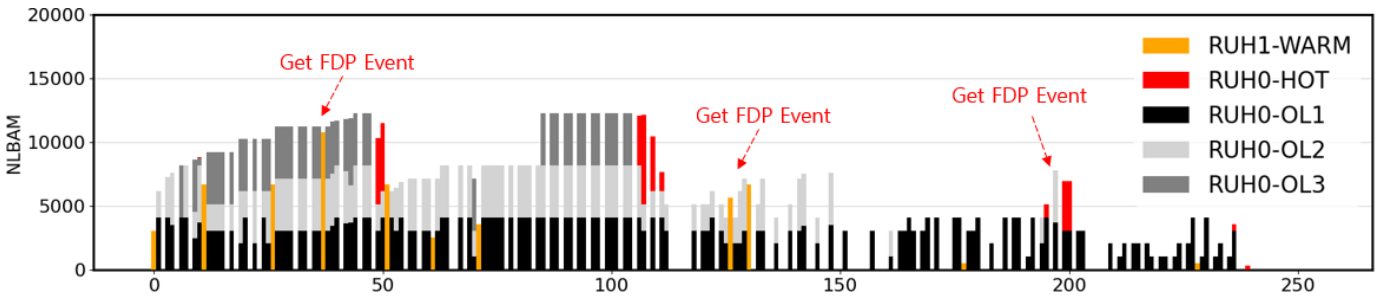

Fig. 1. Block Copy Cost per GC

NVMe specification mandates that among the numerous LBAs moved by GC, only a single LBA is reported.

Given this limitation, it is imperative for the device to report the most potentially useful LBA to the host. We have thus implemented a practical method where the largest contiguous LBA chunk moved during GC is probabilistically considered an outlier, and its first LBA is reported. For example, during a GC operation if LBA chunks (100-101), (107-120), and (125-130) are moved, the first LBA of the second chunk, 107, would be returned because the length is longest. This allows the host to identify the outlier, and by assigning a different PID to the corresponding file or data in subsequent writes, it can reduce the WAF during runtime.

III. Evaluation

In the absence of a physical FDP device, we implemented FDP emulator based on the Blackbox mode (conventional page mapping FTL) of FEMU [2], a QEMU-based SSD emulator. FEMU divides its functionality into a controller logic that handles data I/O from the host and an FTL logic that calculates latency post-processing. The FTL logic includes managing the mapping table and executing GC operations. As QEMU-NVMe [3] already has an FDP Controller logic implemented, this component was ported to fit FEMU, and the internal FTL logic was entirely custom-developed. The emulator supports all two GC schemes and four FDP-specific log pages. The number of RGs can be adjusted by modifying the `RG_DEGREE` value, which indicates how many physical dies are interleaved within a single RG.

We used the Linux kernel v6.3 to utilize I/O passthru [4] for FDP I/O operations. For our experiments, the workloads were executed using the `io_uring_cmd` I/O engine of Fio [5]. The device used has a total capacity of 16 GB, with 8 channels and 8 dies per channel. We configured the endurance group with 1 RG and 4 RUHs, all set to Initially Isolated type. HOT workload involves performing 4 KB random writes at maximum speed on 10% of the total logical area. WARM workload consists of creating files of three sizes—512 KB, 1 MB, and 2 MB—occupying 40% of the total logical area, with random selection and completely overwriting of the selected files. COLD workload involves creating four large files of 600 MB each, with no updates occurring during workload execution. HOT, WARM, and COLD basically use RUH 1, RUH 2, and RUH 3, respectively. However, to demonstrate the gradual improvement in GC cost through FDP events, we intentionally configured part of the WARM data as an outlier (OL) for RUH 0 at the start of the workload.

Fig. 1 illustrates the change in valid block copy cost per stream according to GC sequence. OL1, OL2, and OL3 represent the 512 KB, 1 MB, and 2 MB files, respectively. When the first FDP events log is obtained, it is expected that the LBAs corresponding to the 2 MB files will be predominantly reported among the dozens of Media Reallocated events, allowing the host to consider swapping the RUH. It is assumed here that all OLs are reassigned to RUH 1 immediately after the `get_log_page` command. Although a copy of OL3 written before obtaining the first FDP events log appears later, no such copy occurs after the 105th GC. Subsequent reassignments of RUH through two additional FDP events gradually filter out these OLs, ultimately achieving a nearly zero copy cost.

IV. Conclusion

We proposed a GC performance optimization technique that leverages a new FDP-specific log, the FDP events. By encapsulating the most beneficial LBA information within the Media Reallocated event, which provides only a single unique LBA, and utilizing this LBA information, we can readjust streams during workload runtime whose lifetimes were incorrectly predicted. This strategic approach has significantly reduced the GC cost.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2021R1A2C2095125)

References

[1] Flexible Data Placement (FDP). https://nvmexpress.org/wp-content/uploads/Hyperscale-Innovation-Flexible-Data-Placement-Mode-FDP.pdf

[2] H. Li, M. Hao, M. H. Tong, S. Sundararaman, M. Bjørling, and H. S. Gunawi, "The CASE of FEMU: Cheap, Accurate, Scalable and Extensible Flash Emulator." 16th USENIX Conference on File and Storage Technologies (FAST '18). pp. 83–90, 2018.

[3] NVMe Emulation. "qemu-nvme," https://qemu-project.gitlab.io/qemu/system/devices/nvme.html

[4] K. Joshi, A. Gupta, J. González, A. Kumar, K. K. Reddy, A. George, S. Lund, and J. Axboe, "I/O Passthru: Upstreaming a flexible and efficient I/O Path in Linux." 22th USENIX Conference on File and Storage Technologies (FAST '24). pages 107–122, 2024.

[5] "Fio benchmark," 2014. [Online]. Available: http://freecode.com/projects/fio