

# Braid-ZNS: Leveraging Zone Random Write Area for Efficient In-Storage Compression on ZNS SSDs

Minkyu Choi<sup>\*§</sup>, Joonseong Hwang<sup>†§</sup>, Minjin Park<sup>‡§</sup>, Seokin Hong<sup>\*§</sup>

<sup>\*</sup>Department of Electrical and Computer Engineering, <sup>†</sup>Department of Semiconductor Convergence Engineering

<sup>‡</sup>Department of Semiconductor and Display Engineering, <sup>§</sup>Memory Division, Samsung Electronics, Hwaseong, Korea

<sup>§</sup>Sungkyunkwan University, Suwon, Republic of Korea

{cmk3071, henrly2, ppmjj2000, seokin}@skku.edu

**Abstract**—Zoned Namespace (ZNS) SSD is an emerging storage solution that reduces device-level garbage collection and write amplification. However, the sequential write constraint of ZNS SSDs poses a challenge to adopting in-storage compression, as data placement rules prevent compressed variable-length data from being packed into optimally sized chunks. In this paper, we propose *Braid-ZNS*, a novel in-storage compression framework that leverages the Zone Random Write Area (ZRWA) to avoid double reads during data compression on ZNS SSDs. By exploiting the ZRWA to enable temporary in-place updates, *Braid-ZNS* reorganizes compressed blocks in a size-aware manner and prevents cases where a single logical page is split into multiple fragments. Our evaluation shows *Braid-ZNS* improved compression efficiency by up to 47.0% and throughput by  $\times 2.24$  compared to a state-of-the-art in-storage compression on ZNS SSDs.

## I. INTRODUCTION

Zoned Namespace (ZNS) SSDs are storage solutions that have emerged to overcome the structural inefficiencies of conventional SSDs. It mitigates the block interface tax, the write amplification (WA) and lifetime degradation caused by device-level garbage collection (GC) [1]–[3]. The integration of in-storage compression into ZNS SSDs, however, introduces new challenges: the mismatch between variable data sizes after compression and the strict sequential write constraint enforced by the write pointer (WP). To overcome these limitations, we propose *Braid-ZNS*, an in-storage compression framework for ZNS SSDs that leverages the recently introduced Zone Random Write Area (ZRWA) feature in the NVMe standard [4]–[7].

## II. BACKGROUND AND MOTIVATION

In conventional SSDs, as shown in Fig. 1a, the Flash Translation Layer (FTL) can efficiently place variable-sized compressed data by exploiting random writes within a block [8]–[10]. In contrast, ZNS SSDs enforce coarse-grained address mapping within each zone, eliminating fine-grained page-level mapping and thus complicating compressed data placement [2]. As illustrated in Fig. 1b, when small fixed-size chunks are used, any compressed page that exceeds chunk boundaries must be stored separately. This leads to *double reads* during the procedure of reading a page, retrieving metadata, and extra reading the truncated data. This phenomenon, where more data is read than originally requested, is referred to as *read amplification*, and it directly increases read latency. Increasing the chunk size alleviates this issue but results in internal fragmentation, undermining the capacity gains from compression.

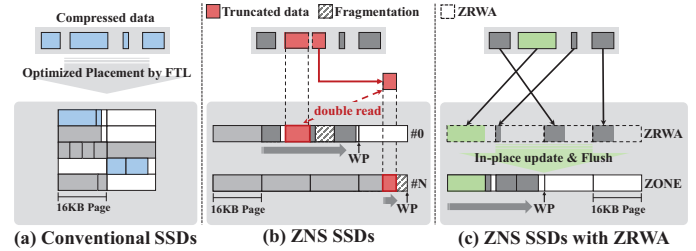


Fig. 1: Compressed data placement in (a) conventional SSDs, (b) ZNS SSDs under sequential write constraints, and (c) ZNS SSDs enhanced with ZRWA (Braid-ZNS).

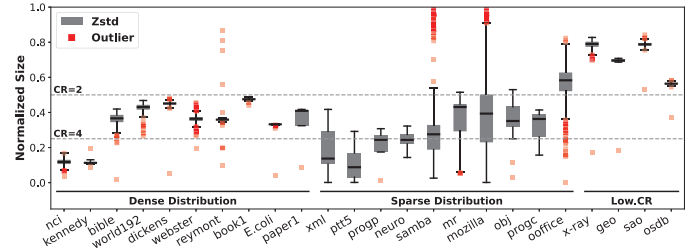


Fig. 2: Breakdown of compressed data size distribution per 16KB page. The results illustrate datasets with dense or sparse distributions, where red points denote outliers.

As shown in Fig 2, our analysis shows that compressed chunk sizes often vary significantly, resulting in sparse distributions or appear mostly uniform while still containing outliers [11]–[13]. Sparse datasets incur frequent double reads under sequential write constraints, while even a single outlier in otherwise dense datasets can disrupt profiling and lead to repeated misplacement. These findings highlight the need for a more robust data placement strategy that can tolerate both highly variable compressed sizes and outliers to avoid read amplification when employing the in-storage compression for the ZNS SSDs.

## III. BRAID-ZNS

We propose *Braid-ZNS* framework, which leverages the Zone Random Write Area (ZRWA) to achieve optimized placement of compressed data on ZNS SSDs. *Braid-ZNS* exploits ZRWA's ability to support random writes and temporary in-place updates, allowing compressed data to be reorganized before being flushed to NAND flash (Fig. 1c). Fig. 3 presents an overview of the framework. Depending on the compressed-size distribution, *Braid-ZNS* applies one of two schemes:

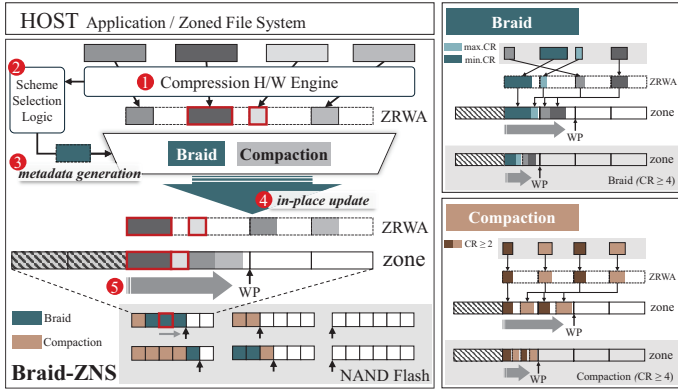


Fig. 3: Braid-ZNS Overview and Compression Schemes

**Compaction**, which is applied to dense patterns to store data without truncation, or **Braid**, which exploits ZRWA reordering under sparse patterns to prevent truncation and double reads.

Upon receiving a write request from the host, the data is compressed by the compression engine and temporarily written into the ZRWA (1). The compression engine forwards the post-compression size metadata to the scheme selection logic, where the maximum and minimum values are recorded (2). After four adjacent pages have been compressed, the scheme selection logic estimates the average compression ratio and its dispersion, and determines the compression scheme with corresponding metadata (e.g., the Braid scheme in Fig. 3) (3). Using this metadata, the compressed data are then reorganized within the ZRWA (4). Finally, the data are flushed from the ZRWA to physical NAND flash pages, and the write pointer (WP) advances by one unit of the ZRWA flush granularity (5).

**Implementation Details:** Braid-ZNS metadata consists of the compressed data size (CPSZ) and a compression scheme header (CPSH) that encodes the applied compression scheme and the ordering of compressed blocks. The metadata required by Braid-ZNS is approximately 9 bytes ( $4 \times 2B + 1B$ ) per 16 KB physical page, and stored in the out-of-band (OOB) area of each page. Since the OOB capacity is about 2 KB per 16 KB page [14], [15], the overhead introduced by Braid-ZNS is negligible. The selection logic is implemented with a small number of combinational components (e.g., comparators, arithmetic units, and multiplexers) and requires roughly 1,500 logic gates in UMC’s 28 nm process. This corresponds to less than 0.02% of the logic area in computational storage drives (CSDs) such as SmartSSD, which integrate about 1.14M logic cells for programmable hardware accelerators [16]–[18].

#### IV. EVALUATION

We emulate Braid-ZNS and other ZNS SSDs using FEMU, which is the most widely used full-system NVMe SSD emulator [19]. For evaluation, we use Base-ZNS without compression as the baseline and compare it against Balloon-ZNS [20], the state-of-the-art work for in-storage compression on ZNS SSDs, as well as our proposed Braid-ZNS. In each dataset, H, M, and L denote the compression ratio (CR) levels, high, medium, and low, respectively, while D and S indicate whether the size distribution of the dataset is dense or sparse. Overall,

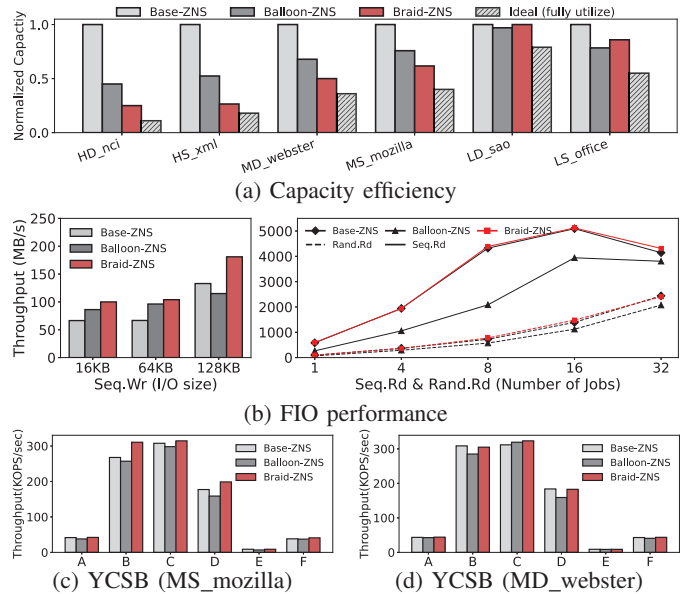


Fig. 4: Braid-ZNS evaluation: (a) Capacity efficiency with six datasets. (b) FIO performance with different I/O sizes and I/O concurrency for MS\_mozilla. RocksDB throughput in six YCSB workloads using (c) MS\_mozilla and (d) MD\_webster.

the results demonstrate that Braid-ZNS improves both capacity efficiency and I/O performance across diverse real-world datasets. Compared to Balloon-ZNS, it achieves up to 47% and 22.5% higher capacity efficiency for high- and medium-compression-ratio datasets without data truncation (Fig. 4a). Performance evaluations with the FIO micro-benchmark [21] show consistent improvements for both reads and writes, with particularly significant gains for sparse datasets and sequential read access, achieving up to 2.24× higher throughput compared to Balloon-ZNS (Fig. 4b). In the case of RocksDB throughputs under six YCSB I/O workloads [22], Braid-ZNS improves on average 18.7% compared to Balloon-ZNS in the MS\_mozilla dataset. Especially, workloads that access consecutive keys repeatedly, such as D and E, achieve an average improvement of 27.2% (Fig. 4c). In MD\_webster dataset, 6.0% improvements are shown on average compared to Balloon-ZNS (Fig. 4d).

These benefits stem from eliminating page fragmentation and double-reads caused by truncated data, allowing Braid-ZNS to deliver efficient data placement and stable performance.

#### ACKNOWLEDGMENT

This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.10692981, 50%), the IITP (Institute of Information & Communications Technology Planning & Evaluation)-ITRC (Information Technology Research Center) (RS-2021-II212052, 25%) grant funded by the Korea government (Ministry of Science and ICT), and Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.00228970, 25%). Seokin Hong is the corresponding author.

## REFERENCES

- [1] M. Björling, A. Aghayev, H. Holmberg, A. Ramesh, D. L. Moal, G. R. Ganger, and G. Amvrosiadis, “ZNS: Avoiding the block interface tax for flash-based SSDs,” in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021.
- [2] K. Han, H. Gwak, D. Shin, and J. Hwang, “Zns+: Advanced zoned namespace interface for supporting in-storage zone compaction,” in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, 2021.
- [3] T. Kim, J. Jeon, N. Arora, H. Li, M. Kaminsky, D. G. Andersen, G. R. Ganger, G. Amvrosiadis, and M. Björling, “Raizn: Redundant array of independent zoned namespaces,” in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2023.
- [4] “Nvm express® zoned namespace command set specification, revision 1.3,” NVM Express, Inc., Tech. Rep., 2025. [Online]. Available: <https://nvmexpress.org/wp-content/uploads/NVMe-Zoned-namespace-Command-Set-Specification-1.3-2025.03.11-Ratified.pdf>
- [5] NVM Express, Inc., “What’s new in NVMe® technology: Ratified technical proposals to enable the future of storage,” <https://nvmexpress.org/wp-content/uploads/Whats-New-in-NVMe-Technology-Ratified-Technical-Proposals-to-Enable-the-Future-of-Storage-1.pdf>, 2021.
- [6] M. Kim, S. Jeong, and J.-S. Kim, “Zraid: Leveraging zone random write area (zrwa) for alleviating partial parity tax in zns raid,” in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2025.
- [7] T. Jiang, G. Zhang, X. Liao, and Y. Zhou, “Zebra: Efficient redundant array of zoned namespace ssds enabled by zone random write area (zrwa),” in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025.
- [8] Y. Qiao, X. Chen, J. Hao, J. Li, Q. Wu, J. Wang, Y. Liu, and T. Zhang, “Improving relational database upon the arrival of storage hardware with built-in transparent compression,” in *2021 IEEE International Conference on Networking, Architecture and Storage (NAS)*, 2021.
- [9] T. Makatos, Y. Klonatos, M. Marazakis, M. D. Flouris, and A. Bilas, “Using transparent compression to improve ssd-based i/o caches,” in *Proceedings of the 5th European Conference on Computer Systems*, ser. EuroSys ’10, 2010.
- [10] H. Kim, S. Kim, J. Park, G. Byeon, and S. Hong, “Don’t cache, speculate!: Speculative address translation for flash-based storage systems,” *IEEE Access*, 2025.
- [11] *Silesia compression corpus*, Silesia corpus, 2003. [Online]. Available: <https://sun.aei.polsl.pl/~sdeor/index.php?page=silesia>
- [12] University of Canterbury, “The canterbury corpus,” <https://corpus.canterbury.ac.nz/descriptions/calgary>, accessed: 2025-09-12.
- [13] NeuroElectro, “Neuroelectro: Organizing information on cellular neurophysiology,” <https://neuroelectro.org/>, 2016, accessed: 2025-09-15.
- [14] H. Qin, D. Feng, W. Tong, Y. Zhao, S. Qiu, F. Liu, and S. Li, “Better atomic writes by exposing the flash out-of-band area to file systems,” in *Proceedings of the 22nd ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, and Tools for Embedded Systems*, ser. LCTES 2021, 2021.
- [15] J. Hwang, M. Choi, M. Park, J. Yoon, Y. Jang, and S. Hong, “Minimizing Read Disturb via Localized Page Allocation for Modern NAND Flash-Based SSDs,” in *2025 IEEE 43rd International Conference on Computer Design (ICCD)*, 2025.
- [16] I. Xilinx, “Samsung smartssd computational storage drive product brief,” Tech. Rep., 2023. [Online]. Available: <https://www.xilinx.com/publications/product-briefs/xilinx-smartssd-computational-storage-drive-product-brief.pdf>
- [17] I. ScaleFlux, “Csd-5000 computational storage drive,” 2025, accessed: 2025-09-12. [Online]. Available: <https://scaleflux.com/products/csd-5000/>
- [18] DapuStor, “Roalsen6 r6x00 product brief version 1.2,” Technical Report / Product Brief, 2025, version 1.2. [Online]. Available: [https://en.dapustor.com/uploads/pdf/DapuStor%20-%20Roalsen6%20R6X00\(EN\)\\_V1.2.pdf](https://en.dapustor.com/uploads/pdf/DapuStor%20-%20Roalsen6%20R6X00(EN)_V1.2.pdf)
- [19] H. Li, M. Hao, M. H. Tong, S. Sundararaman, M. Björling, and H. S. Gunawi, “The case of femu: Cheap, accurate, scalable and extensible flash emulator,” in *16th USENIX Conference on File and Storage Technologies (FAST 18)*, 2018.
- [20] Y. Wang, Z. Sun, Y. Zhou, T. Lu, C. Xie, and F. Wu, “Balloon-zns: Constructing high-capacity and low-cost zns ssds with built-in compression,” in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024.
- [21] J. Axboe, “fio: Flexible i/o tester,” <https://github.com/axboe/fio>, 2006, accessed: 2025-04-10.
- [22] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Ycsb: Yahoo! cloud serving benchmark,” <https://github.com/brianfrankcooper/YCSB>, 2010, accessed: 2025-09-12.