STRAW: A Stress-Aware WL-Based Read Reclaim Technique for High-Density NAND Flash-Based SSDs

Myoungjun Chun, Jaeyong Lee, Inhyuk Choi, Jisung Park, Myungsuk Kim and Jihong Kim

***Abstract*—Although read disturbance has emerged as a major reliability concern, managing read disturbance in modern NAND flash memory has not been thoroughly investigated yet. From a device characterization study using real modern NAND flash memory, we observe that reading a page incurs heterogeneous reliability impacts on each WL, which makes the existing block-level read reclaim extremely inefficient. We propose a new WL-level read-reclaim technique, called STRAW, which keeps track of the accumulated read-disturbance effect on each WL and reclaims only heavily-disturbed WLs. By avoiding unnecessary read-reclaim operations, STRAW reduces read-reclaim-induced page writes by 83.6% with negligible storage overhead.**

I. INTRODUCTION

NAND flash memory has successfully achieved continuous improvements in storage density over decades, but it has come with significant reliability degradation. Vertical wordline (WL) stacking and aggressive multi-level cell (MLC) technologies have effectively increased the bit density of flash chips by 2.4× every two years [1]. However, such capacity-oriented design decisions make modern flash memory significantly more susceptible to various error sources, such as retention loss, read disturbance, and program interference.

In particular, read disturbance has recently gained increasing attention as a major reliability concern in modern (and future) high-density flash memory. To read a page (from a target wordline (WL)), a flash chip applies a high *pass-through* voltage V_{pass} (e.g., $>$ 6 V) to all non-target WLs in the same block, which unintentionally programs all non-target WLs slightly and thus can potentially corrupt their stored data when repeated. As the block size rapidly increases with continuous vertical WL stacking (e.g., a 123-MB block [2]), reading a page disturbs a larger amount of data in the same block.

Despite its importance, how to efficiently manage read disturbance in modern SSDs has yet to be thoroughly investigated; to our knowledge, all prior works on read disturbance in the literature are based on a simple SSD-management task, called *read reclaim (RR)* [3], [4]. When a block's read count RC (i.e., the number of page reads to the block) exceeds a predefined threshold RC_{MAX}, the SSD controller triggers RR to eliminate read-disturbance-induced errors by rewriting (copying) all valid pages in the block to other free pages. The additional writes from RR can significantly affect the performance and lifetime of SSDs [5], so it is necessary to carefully set RC_{MAX} for preventing not only read-distrubance-induced data corruption but also unnecessary RR invocations.

In this work, we show that a conventional RR approach causes prohibitive performance overhead to guarantee data reliability in recent high-density 3D flash memory. In high-density 3D flash memory, reading a page incurs significantly higher disturbance to the target WL's *exact neighbors* compared to the other WLs in the same block [6], [7]. Such an asymmetry in read disturbance across WLs makes the existing *block-level* RR extremely inefficient. For example, when pages at the k-th WL WL_k are read repeatedly, pages at WL_{k-1} and WL_{k+1} may lose their data at a much lower RC value over when pages are randomly accessed over entire WL's. Although the worst-case access pattern (i.e., repeated reads for the same WL) may not be likely in practice, the existing RR approach should handle such a case safely, thus RC_{MAX} being set based on the worst-case pattern.

To mitigate RR overhead, we propose STRAW (STRess-Aware WL-based read reclaim technique for high-density NAND flash-based SSDs), a new *WL-level* RR technique for modern SSDs which effectively minimizes unnecessary RR at low cost. The key insight behind STRAW is that read disturbance should be managed at a finer granularity, i.e., per WL, not per block. Since the impact of read disturbance is substantially different depending on the location of a WL, we need a new approach that can selectively reclaim heavily-disturbed WLs only. To this end, we construct a read-disturbance model that can accurately estimate the impact of a page read on the reliability of each non-target WL in the same block, which enables STRAW to identify any heavily-disturbed WLs and reclaim them in a timely manner. For an efficient implementation of STRAW, we leverage the Space-Saving algorithm [8] to mitigate the space overhead for tracking the actual read-disturbance effect to WLs in each block. Our evaluation using the state-of-the-art SSD simulator [9] shows that STRAW reduces RR-induced writes by 83.6% compared to existing RR approaches with negligible space overhead.

II. READ DISTURBANCE IN MODERN SSDS

Read-Disturbance Variations in Modern Flash Memory. Unlike in planar (2D) flash memory, where a page read disturbs all non-target WLs in the block almost equally [3], the reliability impact of read disturbance significantly varies across WLs in high-density 3D flash memory. Fig. 1 illustrates two key factors contributing to the read-disturbance variations.

First, when reading a page, a high-density 3D flash chip applies a higher V_{pass} (V_{passH}, approximately 0.4V higher than V_{passL} [6]) to the two adjacent WLs compared to the non-adjacent WLs (Fig. 1(a)). Based on the widely-known Fowler–Nordheim (FN) tunneling equation, the impact of

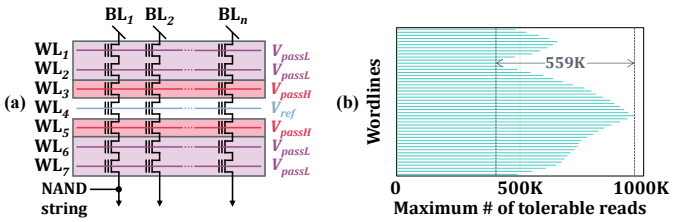

Fig. 1. Key factors for asymmetry in read disturbance across WLs.

read disturbance is exponentially proportional to V_{pass} [3]. Consequently, reading a page leads to a higher reliability impact on the data stored in adjacent WLs [6], [7]. Second, read-disturbance tolerance varies significantly among WLs due to inherent process variations in high-density 3D flash memory [10]. Fig. 1(b) shows the maximum tolerable read counts for WLs within a block when the block's pages are accessed uniformly. As shown in Fig. 1(b), the worst WL in a block can tolerate only 403K reads before data corruption, while the best WL can reliably endure 559K additional reads.

To quantify the asymmetry in read disturbance within a high-density 3D block, we measure the raw bit error rate (RBER) of WLs in a block after repeated read operations on a real TLC flash chip. Fig. 2 illustrates the change in RBER for 48 representative WLs in a block under two distinct read patterns, PA and PB, where PA reads only the page at WL_{35}, which is adjacent to the worst WL (WL_{36}) while PB reads sequentially all pages in the block. As expected, the maximum read count that a block can endure without data loss varies significantly based on the read pattern. Under the pattern PA, reading WL_{35} leads to a significant increase in the RBER of its adjacent WLs (WL_{34} and WL_{36}), resulting in uncorrectable errors (i.e., RBER exceeding the ECC correction capability) after just 54,560 reads. In contrast, the block can tolerate up to 518,420 reliable reads under the pattern PB.

Limitations of Existing Solutions. To evaluate the effectiveness of the conventional RR approach, we measure the number of RR-induced page copies in two SSDs, both employing the block-level RR, SSD_{real} and SSD_{ideal}. SSD_{real} is an SSD composed of our tested high-density 3D flash memory, while SSD_{ideal} is a hypothetical SSD where read disturbance is symmetric; all non-target WLs in a block experience uniform disturbance, with the tolerable read counts of all WLs set equal to the median value shown in Fig. 1(b). In both SSDs, RC_{MAX} is conservatively set to ensure data reliability even under the worst-case access pattern.

Fig. 3 compares the number of page copies from RR, normalized to SSD_{ideal}, across two distinct P/E cycles under six workloads [11]. (For a detailed description of the workloads, see §IV.) From the results, we observe that the block-level RR imposes significant lifetime/performance overhead to ensure data reliability in modern high-density 3D flash memory. The number of RR-induced page copies increases by 10.5×, and 15.4× on average compared to when read disturbance is symmetric, at 1K, and 2K P/E cycles (PEC), respectively. Our evaluation results highlight the fundamental limitations of the block-level RR. Due to the heterogeneous reliability impact of read disturbance, data loss can occur at significantly lower RC values under worst-case access patterns (Fig. 2). Consequently, block-level RR must conservatively set RC_{MAX}, leading to frequent and unnecessary RR operations.

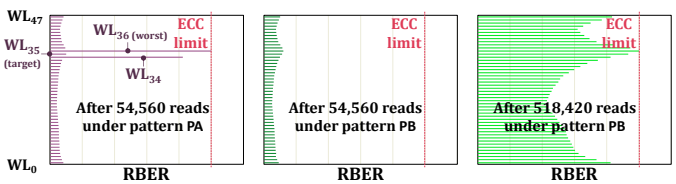

Fig. 2. Heterogeneous disturbance impact under different read patterns.

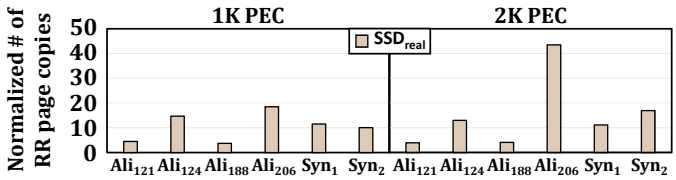

Fig. 3. A comparison of the RR overhead between SSD_{real} and SSD_{ideal}.

III. STRAW: STRESS-AWARE WL-BASED READ RECLAIM

To overcome the limitations of existing solutions, we propose STRAW, a novel WL-level RR technique. Unlike existing RR techniques that invoke block-level RR based on a conservative RC_{MAX}, STRAW reclaims individual WLs only when necessary, thereby significantly reducing the performance and lifetime overheads from RR. To this end, we develop (*i*) a new read-disturbance model that quantifies the heterogeneous reliability impact of read disturbance (§III-A) and (*ii*) a STRAW-enabled flash translation layer (FTL) that efficiently estimates the actual read disturbance accumulated to each WL (§III-B) by leveraging an approximate counting algorithm [8].

A. New Read-Disturbance Model

We develop a new read-disturbance model through comprehensive characterization of 160 real 3D TLC flash chips from Samsung. Our proposed model quantifies two key factors that contribute to the heterogeneous disturbance impact on non-target WLs during a read operation: (*i*) the inherent process variations across WLs [10] and (*ii*) read-disturbance asymmetry between adjacent and non-adjacent WLs. To account for the first factor with minimal overhead, we classify the WLs within each block into four groups, Best, Good, Bad, and Worst, based on their initial RBER values. We then measure the maximum read count for each group across 19,200 blocks, while varying PEC and read patterns.

Fig. 4 compares the maximum read counts of four WL groups under different PEC and read patterns. A coordinate (x, y) for each WL group indicates that the group's worst WL can endure up to y page reads on the same block (i.e., read disturbance from the reads) when x% ($(100-x)$%) of the reads are performed to (non-)adjacent WLs. We make three key observations. First, reading a WL causes significantly more disturbance (8.4× on average) to adjacent WLs compared to non-adjacent WLs. Second, under the same operating condition, the ratio of read disturbance impact between non-adjacent and adjacent WL reads maintains a consistent disturbance rate α. For instance, at 2K PEC, the disturbance rate α is 8.7 for the Best group, indicating that reading the adjacent WL of the worst WL in the Best group (WL_w^{Best}) incurs 8.7× more disturbance stress on WL_w^{Best} compared to reading the non-adjacent WL of WL_w^{Best}. Third, the read-disturbance tolerance of a WL group and the disturbance rate α vary significantly depending on operating conditions, such as the inherent reliability characteristics of the WLs and the PEC.

Based on our observations, we derive a model with two key parameters for each WL group: (*i*) the effective maximum read count, ERC_{MAX}, which denotes the maximum number of reads the worst WL in a group can tolerate from non-adjacent WL reads, and (*ii*) the disturbance rate α, which

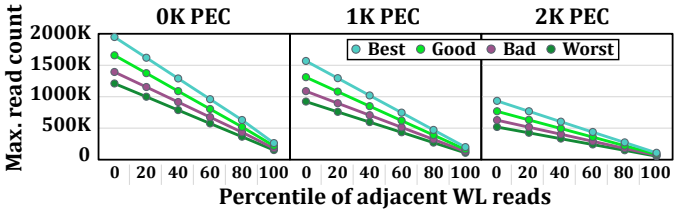

Fig. 4. Comparisons of the maximum read counts of each WL Group at different PEC under various read patterns.

PEC	ERC_{MAX}	α	ERC_{MAX}	α	ERC_{MAX}	α	ERC_{MAX}	α
0	1947K	7.4	1657K	7.6	1391K	7.8	1210K	8.0
0.5K	1806K	7.7	1521K	7.9	1270K	8.0	1094K	8.3
1K	1567K	7.9	1309K	8.2	1087K	8.4	922K	8.6
1.5K	1310K	8.3	1086K	8.6	899K	8.8	749K	9.0
2K	933K	8.7	767K	9.0	627K	9.2	518K	9.5
2.5K	539K	9.2	438K	9.5	354K	9.7	288K	10.1
3K	111K	9.7	89K	10.0	71K	10.3	58K	10.7
	Best		Good		Bad		Worst	

Fig. 5. Final Model of RC_{MAX} and α under different PEC.

quantifies the relative impact of adjacent WL reads compared to non-adjacent WL reads. The proposed model allows for determining whether a WL is heavily disturbed by using its current effective read count, derived from the read counts of its adjacent and non-adjacent WLs. Fig. 5 shows the parameters of the final model for the tested flash chips under different PEC. For example, at 2K PEC, $\text{WL}_w^{\text{Good}}$ can tolerate 767K non-adjacent reads, and the disturbance rate α is 9.0, respectively.

B. STRAWFTL

We implement an STRAW-enabled FTL, called STRAWFTL, by extending the conventional page-level FTL [9] with two key data structures: (*i*) Read-reclaim Parameter Table (RPT) and (*ii*) Resource-Efficient Counters (REC). The RPT is a table to store ERC_{MAX} and α for each PEC, which can be built through offline profiling of target chips (Fig. 5). The REC is a set of per-block counters that keep tracks the RC values of individual WLs within a block, as well as the RC value of the block itself.

Fig. 6 illustrates how STRAWFTL estimates the accumulated disturbance impact on individual WLs. For WL_i, which is located in the i-th WL in the k-th block, it first looks up the RC values of WL_{i-1}, WL_i, WL_{i+1}, and BLK_k from the REC (❶). Based on the obtained RC values, STRAWFTL determines the number of reads to adjacent and non-adjacent WLs of WL_i (❷). Then it queries the RPT with the PEC of BLK_k and the WL group to which WL_i belongs (❸). STRAWFTL converts the number of reads to adjacent and non-adjacent WLs of WL_i into the ERC, using the disturbance rate α from the query result (❹). The remaining process is straightforward. If the ERC of WL_i (including the possible additional reads by the next interval) exceeds ERC_{MAX} from the RPT, STRAWFTL identifies WL_i as a heavily-disturbed WL (❺).

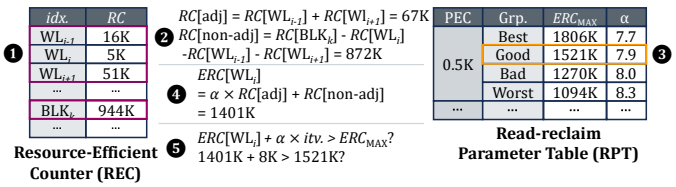

Fig. 6. A procedure for identifying heavily-disturbed WLs in STRAW.

Fig. 7 shows how STRAWFTL manages read-disturbance at a WL-granularity. Whenever a page is read, STRAWFTL updates the REC for the target block and WL. Every predefined interval (e.g., every 1K reads to the block), STRAWFTL checks all valid WLs in the block to determine whether the accumulated disturbance impact on any valid WL exceeds the threshold or if there is a possibility it will exceed the threshold by the next interval. For such WLs, STRAWFTL copies the valid pages to free pages before the next interval, thereby preventing read-disturbance-induced data corruption. If the block contains no valid pages after the checking procedure, STRAWFTL erases the block and resets all associated counters.

Overhead Optimization. STRAWFTL only requires simple modifications over the conventional FTL at high level, but a naive implementation of per-WL counters introduces non-trivial storage overhead. For example, assuming a 2-TiB SSD comprising a block with 2,568 WLs [2], the REC requires approximately 125 MB of internal DRAM space (one 3-byte counter per 48-KB WL), which is more than 2,568 times the space required for per-block counting. Even though modern SSDs typically employ internal DRAM equivalent to 0.1% of the total SSD capacity, most of this DRAM is dedicated to the address mapping table [9], leaving only a few tens of MB available for other metadata and cache management.

To minimize the storage overhead of per-WL counters, the REC incorporates the Space-Saving (SS) algorithm [8], which efficiently estimates the frequency of elements in a data stream using a limited number of counters. Each counter entry consists of an element index and its corresponding count value. In our context, the data stream consists of a sequence of reads to a block between successive block erases.

Upon block erasure, the REC initializes a predefined number of counter entries for that block. Whenever a WL is read, the REC first checks for an existing counter entry associated with that WL index. If an entry exists, the REC increments the corresponding count value. If no entry exists, the REC identifies the entry with the *lowest* count value, increments its count value, and replaces the element index with the current read WL index. When the STRAWFTL queries the read count for a particular WL, the REC returns the corresponding count value if an entry exists. If no entry is found, the REC returns the minimum count value among all entries.

Due to the limited number of counters, the estimated read counts by the REC may introduce some error, but SS ensures that the estimated count value for any element is never underestimated [8]. This guarantees that errors in estimation do not result in read-disturbance-induced data corruption, although they may cause premature RR invocations.

IV. EVALUATION

Evaluation Setup. We evaluate the effectiveness of our pro-

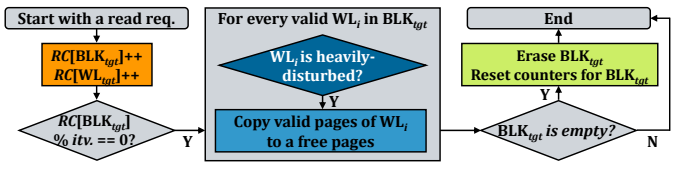

Fig. 7. Fine-grained read-disturbance management in STRAW.

posal using MQSim-E [9], a state-of-the-art SSD simulator. We extend MQSim-E to trigger RR operations according to the read-disturbance characteristics observed in our 19,200 tested blocks. We configure the architecture and key parameters of the emulated SSDs to closely match those of modern high-performance SSDs employing high-density 3D flash memory. Table I summarizes the simulated SSD configuration. We evaluate two synthetic workloads with different I/O patterns, as well as four real-world workloads obtained from Alicloud traces [11]. Table II compares the key I/O characteristics of six workloads with varying read ratios and read patterns.

We compare four SSD configurations with different read-disturbance management techniques, BLOCK, PAGETYPE, STRAW-SS, and STRAW-WL. BLOCK is our baseline SSD that employs block-level RR. PAGETYPE is an SSD that adopts a state-of-the-art read-disturbance management technique [12]. Unlike block-level RR, PAGETYPE classifies pages within a block according to their page types (e.g., MSB, CSB, and LSB pages in TLC flash memory) and migrates them based on their vulnerability to read disturbance. Both STRAW-SS and STRAW-WL are SSDs that implement STRAWFTL; however, STRAW-SS utilizes the SS algorithms [8] for WL-level counters (32 counter entries for a block), while STRAW-WL employs naive WL-level counters.

Evaluation Results. We first measure the number of pages copies from RR, which is directly related to the effectiveness of read-disturbance management. Fig. 8 compares the number of RR-induced page copies in four SSD configurations, normalized to BLOCK, under two different PECs. We make three observations. First, both STRAW-SS and STRAW-WL significantly reduce the number of RR-induced page copies compared to BLOCK, by preventing premature RR invocations. For example, STRAW-SS (STRAW-WL) reduces the number of RR-induced page copies over BLOCK by 83.8% (91.5%) on average at 2K PEC. Second, PAGETYPE also reduces the number of RR-induced page writes considerably (by 29.4% on average) compared to BLOCK, but its benefits are limited compared to both STRAW-SS and STRAW-WL. Third, with significantly less space overhead, STRAW-SS achieves efficiency comparable to STRAW-WL under random-read patterns and remains competitive under sequential-read patterns.

We evaluate the impact of reduced RR invocations on read tail latency, which is a crucial performance factor for many data-intensive apps. Fig. 9 depicts a comparison of the 99.9th percentile read latencies across the four SSDs at two

TABLE I
EVALUATED SSD CONFIGURATIONS.

Configuration	2-TiB total capacity; 8 channels; 4 dies/channel; 4 planes/die; 321 vertical WLs/block; 141 blocks/plane; 7704 pages/block
Latencies (μs)	tREAD = 40; tPROG = 380; tERASE = 3500;
Bandwidth	8.0 GB/s external I/O bandwidth (PCIe 4.0, 4-lane); 2.0 GB/s channel I/O bandwidth

TABLE II
KEY I/O CHARACTERISTICS OF SIX WORKLOADS.

Workload	**Ali**$_{121}$	**Ali**$_{124}$	**Ali**$_{188}$	**Ali**$_{206}$	**Syn**$_1$	**Syn**$_2$
Read ratio	0.55	0.98	0.85	0.99	1.0	1.0
Read pattern	Seq.	Mixed.	Seq.	Rand.	Rand.	Mixed.

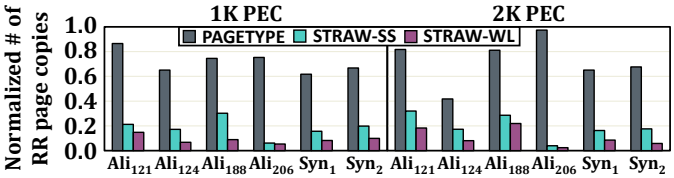

Fig. 8. Comparison of RR-induced page copies under two PEC.

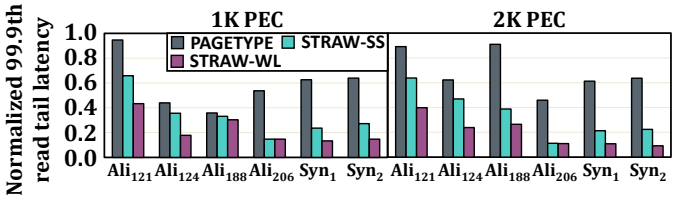

Fig. 9. Comparison of 99-th percentile read tail latencies under two PEC.

distinct PEC values. All values are normalized to BLOCK. We make two observations. First, STRAW-SS (STRAW-WL) significantly reduces the 99.9th percentile read latencies compared to BLOCK by 70.4% (81.3%), on average across all the evaluated workloads and PEC. Second, in modern SSDs, the extra operations by RR substantially impact read tail latencies, highlighting the need for efficient read-disturbance management (such as our proposed solution) to meet the strict service level agreements of modern data-intensive apps.

V. CONCLUSION

We have proposed a new WL-level read reclaim technique, STRAW, which significantly improves SSD lifetime and performance by reducing the frequency of RR invocation. We identified that the existing block-level RR is extremely inefficient in modern SSDs due to the heterogeneous impact of read disturbance in high-density flash memory. Unlike block-level RR that performs RR at block granularity, STRAW identifies heavily-disturbed WLs within blocks and reclaims them in a timely manner. Our evaluation results showed that STRAW effectively enhances SSD lifetime and performance.

REFERENCES

[1] A. Goda. Recent Progress on 3D NAND Flash Technologies. *Electronics*, 2021.
[2] B. Kim et al. 28.2 A High-Performance 1Tb 3b/Cell 3D-NAND Flash with a 194MB/s Write Throughput on over 300 Layers i. In *ISSCC*, 2023.
[3] K. Ha et al. An Integrated Approach for Managing Read Disturbs in High-Density NAND Flash Memory. *IEEE TCAD*, 2015.
[4] G. Zhang et al. Cocktail: Mixing Data with Different Characteristics to Reduce Read Reclaims for NAND Flash Memory. *IEEE TCAD*, 2022.
[5] C. Liu et al. Prolonging 3D NAND SSD Lifetime via Read Latency Relaxation. In *ASPLOS*, 2021.
[6] Qin Xiong et al. Characterizing 3D Floating Gate NAND Flash: Observations, Analyses, and Implications. *ACM TOS*, 2018.
[7] T. Ren et al. Read Disturb and Reliability: The Complete Story for 3D CT NAND Flash. In *NVMSA*, 2023.
[8] A. Metwally et al. Efficient Computation of Frequent and Top-k Elements in Data Streams. In *ICDT*, 2005.
[9] D. Lee et al. MQSim-E: An Enterprise SSD Simulator. *IEEE CAL*, 2022.
[10] Y. Shim et al. Exploiting Process Similarity of 3D Flash Memory for High Performance SSDs. In *MICRO*, 2019.
[11] Jinhong Li et al. An In-Depth Comparative Analysis of Cloud Block Storage Workloads: Findings and Implications. *ACM TOS*, 2023.
[12] Sangwoo Han et al. Page Type-Aware Data Migration Technique for Read Disturb Management of NAND Flash Memory. *IEEE TVLSI*, 2023.