Mitigating High-Density SSD Shortage via Overhead-Aware Lifetime Optimization

Darong Yang*, Yanqi Pan*, Hao Huang*, Wen Xia*†§, Cai Deng*, Haijun Zhang‡
*Harbin Institute of Technology, Shenzhen, China
†Pengcheng Laboratory, Shenzhen, China
‡Jinan Inspur Data Technology Co., Ltd., China

***Abstract*—The sharply reduced program/erase cycles of high-density SSDs, combined with an AI-driven SSD shortage, have made their lifetime a critical concern. WOM-v codes mitigate this issue via multiple in-place reprogramming to reduce erasure. However, existing uniform encoding strategies ignore an inherent lifetime-space trade-off, leading to either lifetime potential loss or excessive space waste.**

We propose RouterSSD, an architecture that co-optimizes lifetime and space efficiency on high-density SSDs. RouterSSD precisely tracks the update characteristics of logical pages and intelligently routes heterogeneous WOM-v codes to the pages based on update frequency. Experiments on real-world traces show that RouterSSD consistently approaches the Overhead-Aware Lifetime Optimization (OALO) metric upper bound, with average OALO gains of up to 103% over WOM-v(1,4)-SSD and 17.0% over the best-performing uniform alternative.

I. Introduction

Driven by the exponential growth of data-intensive applications (e.g., AI), high-density, cost-effective solid-state drives (e.g., QLC/PLC SSDs) [1], [2], [3], [4] have become critical for next-generation storage infrastructures. However, the deployment of such drives is currently constrained by an unprecedented, industry-wide SSD shortage. At the same time, their increased density comes at a steep physical price: substantially reduced program/erase (P/E) cycles that cause rapid wear-out and frequent replacements, which are unsustainable under the ongoing shortage [5], [6], [7], [8].

To address this lifetime challenge, WOM-v codes [9] introduce a transformative programming paradigm that enables reprogramming without erase on high-density SSDs. Conventionally, the 4-bit (16-level) states of a QLC cell encode 4 bits of data and must be fully erased before new data can be written. In contrast, WOM-v codes intentionally map these 16 levels to fewer than 4 bits, providing redundancy that allows multiple reprogrammings prior to erase. This redundancy translates into substantial endurance improvement, offering up to a 3.75× lifetime gain on QLC SSDs.

To pursue this maximal lifetime gain, a straightforward approach adopted by current works [9] is to uniformly apply single WOM-v(1,4) (encoding 1-bit data over 4-bit voltage states) with maximum lifetime optimization potential to high-density SSDs. However, this indiscriminate strategy is fundamentally inefficient in practice. As shown in Figure 1, we observe that there exists an inherent trade-off between lifetime and space efficiency across different WOM-v codes. A real-world trace in the OLTP scenario [10] shows that this strategy fails to deliver the expected maximal lifetime gains at 3.75×, while resulting in substantial unnecessary space waste. Conversely, the trace in the Mail scenario [10] shows that the conservative single WOM-v(2,4) strategy misses substantial lifetime optimization potential.

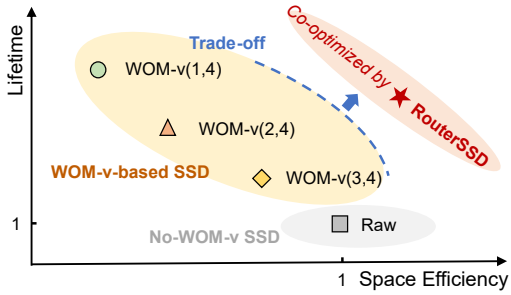

Fig. 1. **Lifetime-space tradeoff on high-density SSDs.** Different WOM-v codes entail inherent trade-offs between lifetime and space efficiency.

Our realistic workload analysis (in §III) reveals that significant I/O skewness is prevalent in the real world: across smartphones, PCs, and cloud infrastructures, a large fraction of logical page addresses (LPAs) receive very few updates [11], [12], [13], [14], [15], while a small fraction of LPAs accounts for the majority of updates. These real-world features expose the root causes behind the inefficiency of the indiscriminate encoding strategies. On cold pages, aggressive assignment (e.g., in WOM-v(1,4)) incurs substantial unnecessary space waste, whereas on hot pages, conservative reprogramming-capability assignment (e.g., in WOM-v(2,4)) misses potential lifetime gains. As a result, any uniform encoding strategy fails to achieve efficient lifetime optimization under realistic dynamic workloads.

We propose RouterSSD, a lifetime-space co-optimization architecture on high-density SSDs. Based on our observations and analysis, our key insight is that, rather than applying a uniform encoding to all pages, an effective SSD architecture should adaptively route heterogeneous WOM-v codes according to per-LPA I/O behavior to jointly optimize lifetime and space efficiency. We further introduce the Overhead-Aware Lifetime Optimization (OALO) metric as a unified co-optimization objective to guide RouterSSD's design.

However, this is non-trivial and directly raises two practical challenges. First, accurately maintaining per-LPA update be-

§Corresponding author. Email: xiawen@hit.edu.cn.

havior requires careful design. Different LPAs exhibit distinct update patterns, and an individual LPA's pattern can change over time as workloads evolve. Capturing these dynamics efficiently without incurring excessive DRAM overhead poses a challenge. Second, determining the optimal WOM-v code for LPAs in dynamic workloads on high-density SSDs is challenging. The limited resources [16], [17] (DRAM, CPU) and strict I/O latency constraints render sophisticated optimization algorithms impractical.

RouterSSD *solves the first challenge* by using an I/O Tracker to record and manage per-LPA update frequency by adding only minimal metadata to the logical-to-physical (L2P) mapping table. In each traffic window, RouterSSD resets the hotness counters of LPAs to accommodate workload changes. RouterSSD *addresses the second challenge* by employing a stepwise transition scheme: I/O Router transitions pages through increasingly aggressive WOM-v codes, progressively moving from unencoded to WOM-v(1,4) as their update frequency increases. The transition thresholds are derived from our theoretical modeling. RouterSSD is designed to be lightweight and practical, adding only minimal metadata and using constant-time routing logic.

We implement RouterSSD on the LightNVM platform [18]. Extensive evaluations based on real-world traces demonstrate that RouterSSD approaches the OALO upper bound, achieving up to 103% higher OALO than WOM-v(1,4)-SSD and 17.0% higher than the best-performing uniform alternative.

In summary, this paper makes the following contributions to address the lifetime challenge of high-density SSDs:

- We identify an opportunity to exploit I/O patterns to fully unlock the fundamental lifetime-space trade-off (§III).
- We propose the overhead-aware lifetime optimization methodology on high-density SSDs for the first time (§II-C), and design RouterSSD, a lifetime-space co-optimization architecture, intelligently routing heterogeneous WOM-v codes to LPAs based on update frequency (§IV).
- Experiments show that RouterSSD surpasses state-of-the-art solutions in co-optimizing lifetime and space (§V).

II. Background

A. P/E Characteristics on High-Density SSDs

NAND flash memory is organized in a hierarchical structure consisting of channels, chips, planes, blocks, pages, and cells.

Flash cell. NAND flash memory stores data by controlling the threshold voltage (V_{th}) of the transistors. High-density flash cells (e.g., QLC) use multiple voltage levels (e.g., V_0–V_{15}) to represent data. Figure 2a shows a flash cell internal structure. To store 4-bit data at V_{11} in a QLC cell, the programming process monotonically injects electrons into the trapping layer [19] with program voltage (V_{pgm}, e.g., +15V), incrementally raising V_{th} to the target V_{11} level. As Figure 2b shows, to determine whether the electrons reach V_{11}, the programming circuit injects charge in small incremental pulses and verifies the resulting V_{th} after each step, which is termed incremental step pulse programming (ISPP).

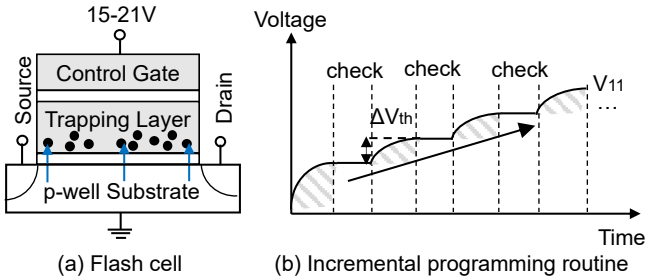

Fig. 2. **Programming on flash memory**.

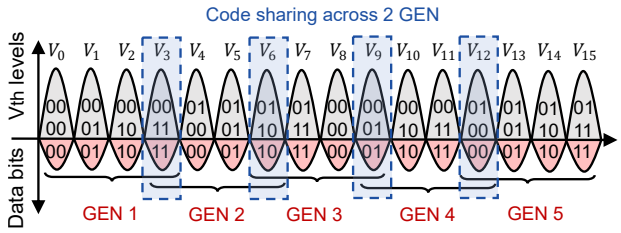

Fig. 3. **WOM-v code scheme.** GEN denotes generation.

One-directionality. A NAND cell's programming is one-directional [9], as decreasing V_{th} requires a destructive erase operation. This process imposes significant stress on the tunnel oxide, leading to gradual degradation and ultimately serving as the primary limiter of cell endurance [20]. Prior studies [21], [22] show that erasure is the dominant source of wear-out, accounting for roughly 80%; the damage introduced by programming is comparatively slight.

Lifetime challenge. Unfortunately, increasing flash cell density significantly reduces P/E cycle endurance due to narrower error margins for detecting correct voltage states. Prior studies demonstrate that the P/E cycle limit drops from approximately 11,000 cycles in two-state SLC to just 800 in 16-state QLC [9].

B. WOM-v: A New Programming Paradigm

To address the lifetime issue, WOM-v codes [9] introduce a transformative programming paradigm on high-density SSDs.

Principle. To allow page reprogramming, the key is to avoid voltage conflicts (i.e., from high voltage to low voltage) under the one-directional constraints (in §II-A) when programming new data. This is achieved by representing fewer valid bits using voltage states [23], [24], [25], [9]. WOM-v(x,y) codes [9], [26] use a single lookup table to map y-bits (2^y) voltage states to x-bits data ($x < y$). As shown in Figure 3, WOM-v(2,4) maps 4-bit (2^4=16) voltage levels of a QLC cell to 2-bit data; with this representation, the 2-bit data can be programmed at least 5 times. A cell that originally stores only 4-bit data can represent 10-bit (2 bits×5 times) data in WOM-v(2,4), theoretically achieving a 1.5× lifetime improvement.

Why WOM-v codes matter. WOM-v codes can provide substantial lifetime potential. On QLC, WOM-v(1,4) can achieve a 3.75× lifetime. We theoretically model the lifetime and space efficiency of WOM-v codes. As shown in Figure 4a, it enables exponential improvement in lifetime as density continues to scale, while the space efficiency loss is bounded

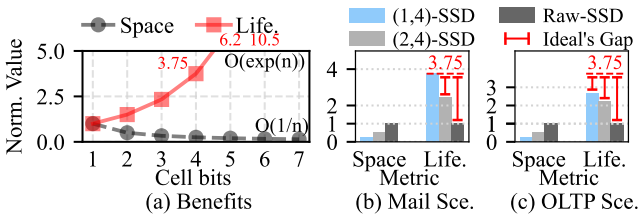

Fig. 4. **Analysis of WOM-v-based approaches.** We evaluate lifetime and efficiency under representative real-world scenarios, the Mail scenario, and the OLTP scenario. We model the ideal lifetime as the theoretical lifetime promised by WOM-v(1,4), representing the upper bound on QLC.

by $O(1/n)$. This capability is significant for extending the lifetime of current and future high-density flash [27], [7], [8].

C. Overhead-Aware Lifetime Optimization

To quantify lifetime-space co-optimization, we introduce the Overhead-Aware Lifetime Optimization (OALO) metric for high-density SSDs, inspired by the Energy-Delay Product widely used in computer architecture. We adopt a multiplicative form $\text{OALO}(n) = L \times S^{1/n}$, where L and S denote the lifetime (e.g., terabytes written) and space efficiency (e.g., usable capacity ratio) normalized against a baseline, and $n \geq 1$ controls the relative weight of space efficiency.

III. Observation and Motivation

A. Revisiting the WOM-v-based Approaches

Observation #1: Lifetime-space trade-off. Our study reveals a previously ignored property: different WOM-v codes offer inherently different lifetime-space trade-offs. For WOM-v(x, y), y typically corresponds to the NAND cell density (e.g., $y = 4$ for QLC). With a fixed y, decreasing x increases lifetime by enabling more reprogramming times, but at the cost of reduced space efficiency with fewer data bits per cell.

Unfortunately, current integrations of WOM-v codes are straightforward and overlook this property. They apply uniform encoding to every page. Therefore, to apply the WOM-v(x,y) code, the SSD controller maintains y/x physical pages (PP) to represent one logical page (LP).

Problems. We observe that current WOM-v-based approaches suffer from two practical problems in real-world workloads.

- First, to pursue the maximal lifetime gain promised by WOM-v(1,4), current works [9], [28], [29] often adopt a straightforward strategy that uniformly applies WOM-v(1,4) to all pages. However, as shown in Figure 4c, in update-intensive workloads (e.g., OLTP), uniformly using WOM-v(1,4) fails to realize the ideal 3.75× lifetime in practice, while its space efficiency remains fixed at 1/4 of Raw-SSD, resulting in substantial and persistent space waste.
- Second, in the FIU-Mail workload (Figure 4b), a conservative choice such as WOM-v(2,4)-SSD or Raw-SSD avoids the worst space overhead but forgoes the full reprogramming capability, leaving significant lifetime potential untapped. As a result, both WOM-v(2,4)-SSD and Raw-SSD remain far below the ideal upper bound.

TABLE I
Inter-page skewness.

Workload	Cold (1)	Warm (2-10)	Hot (>10)
OLTP	3.51%	18.22%	78.27%
FIU-Web	45.36%	20.13%	34.51%
FIU-Mail	0%	25.29%	74.71%
Phone	46.70%	46.44%	6.86%
Home	40.37%	32.52%	27.11%
RocksDB	45.88%	53.85%	0.27%

B. Root Causes and Opportunities

To reveal the root causes of these problems, we analyze I/O traces from diverse real-world workloads, including FIU-Home, Mail, Web [10], [30], RocksDB [31], [32], Phone [33], and OLTP [34], covering personal PCs and servers, key-value stores, smartphones, and financial OLTP systems.

Observation #2: Update skewness. Our study observes a highly skewed update pattern in real-world workloads. We categorize SSD pages based on their update counts into three groups: Cold (1 update), Warm (2-10 updates), and Hot (>10 updates). As shown in Table I, the logical page addresses (LPAs) exhibit varying update frequencies. For example, in the FIU-Web workload, 45.36% of the LPAs are updated only once (Cold), while 34.51% are updated more than 10 times (Hot). Phone, as another example, shows that 46.70% of LPAs are Cold, while only 6.86% of LPAs are frequently updated.

Root causes and opportunities. A uniform aggressive assignment wastes reprogramming capability on infrequently updated pages while still incurring substantial space overhead, whereas a uniform conservative assignment of reprogramming capability misses lifetime gains on frequently updated pages. This analysis yields a key design insight: rather than applying a uniform encoding to all pages, an effective high-density SSD architecture should intelligently assign heterogeneous WOM-v codes based on per-LPA I/O behavior. Fortunately, modern SSDs already maintain an internal L2P table that records every LPA entry, offering a low-overhead substrate for workload-aware hotness tracking and adaptive routing.

IV. Design and Implementation

A. RouterSSD Overview

RouterSSD architecture. Fig. 5 illustrates the RouterSSD architecture. RouterSSD employs a lightweight system design and algorithm, adding only minimal metadata and using constant-time routing logic, which enables efficient deployment in high-density SSDs. First, *Hotness Tracker* (§IV-B) records the update frequency (i.e., hotness) of each LPA during runtime. Second, *I/O Router* (§IV-C) adaptively encodes data with different WOM-v codes by considering the hotness of each LPA, employing a stepwise transition strategy guided by theoretical modeling of different WOM-v codes.

- *New write.* If the requested LPA is not indexed, RouterSSD allocates a new physical page address (PPA) and writes the data directly.
- *In-place update.* If the LPA is already indexed, RouterSSD records its update frequency and, based on this frequency,

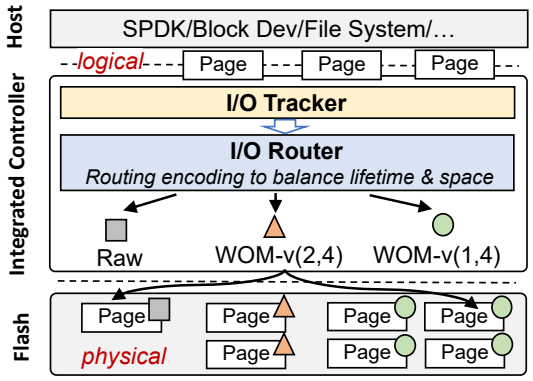

Fig. 5. **RouterSSD architecture.**

determines the encoding scheme using a lightweight routing algorithm and updates the data in-place.

- *Read.* RouterSSD identifies the encoding type of the LPA and obtains data from the corresponding physical pages.

B. I/O Tracker

We extend SSD's L2P table to support the I/O Tracker. Each L2P entry is extended to (LPA, `idx`, `hotness`) → PPA.

We use the `hotness` field to store the hotness of the LPA. Each time an LPA is updated, its corresponding counter is incremented by one. Due to the temporal locality of workloads, the hotness of an LPA may change over time. To address this, we introduce a cool-down mechanism: after the SSD has processed N pages (by default, covering the entire physical capacity), the hotness values are reset to zero during garbage collection, allowing a new round of hotness tracking. We do not need to explicitly record the coding type, as it can be inferred from the `hotness` field via the I/O Router's routing algorithm, which provides a fixed mapping from hotness to encoding type.

The usage of WOM-v code enables one-to-many mapping from LPAs to PPAs. We use the `idx` field to indicate the logical order of retrieving the PPA to restore the original logical page. The `idx` field is set when new PPAs are allocated for the same LPA.

C. I/O Router

We propose I/O Router to encode updates by considering LPA hotness for its desired reprogramming capability. For each update, we consider three levels of WOM-v codes: No WOM-v, WOM-v(2,4), and WOM-v(1,4). Due to page misalignment issues, we do not consider WOM-v(3,4) at the LPA level. In the initial state, RouterSSD does not apply any encoding to the pages (i.e., no WOM-v). When an update occurs on an LPA, its hotness counter increases by one. RouterSSD tracks hotness for all valid LPAs in the SSD logical to physical (L2P) page mapping table.

Vertical-first strategy. For each physical page, we aim to provide the appropriate reprogramming capability (i.e., considering the vertical dimension first) to minimize the redundancy caused by encoding. Hence, we primarily use the update frequency to determine the encoding for each logical page.

Algorithm 1 Vertical-First Strategy for WOM-v Coding

Input: Logical Page p
Output: Appropriate WOM code Routing for p

```
Load LPA's hotness h via Hotness Tracker
if h < TH1 then                                  // Low freq
    Select No-WOM-v coding
else if TH1 ≤ h < TH2 then                       // Medium freq
    Select WOM-v(2,4) coding
else                                             // High freq
    Select WOM-v(1,4) coding
end if
Program/Read page p with selected WOM code
if Program then
    Increment h via Hotness Tracker
end if
```

As shown in Fig. 6, we model the cumulative physical pages each encoding consumes as hotness grows. The optimal strategy (red line) picks the encoding with the fewest pages at each hotness. Each threshold is set at the first integer past the crossover point of two curves; when curves overlap, the code with lower per-update overhead is preferred. This yields TH_1=2 and TH_2=10. As shown in Algorithm 1, I/O Router switches No-WOM-v, WOM-v(2,4), and WOM-v(1,4) codes based on when the LPA is updated in $[0, 2)$, $[2, 10)$, and $[10, \infty)$ times, respectively.

Encoding on programming. Based on the latest appropriate WOM-v code, the I/O Router dynamically encodes input data directly into higher voltage states according to the current cell voltage of the previous WOM-v code, instead of immediately allocating new physical pages, thereby maximizing voltage utilization. If encoding the data with the latest WOM-v code would exceed the maximum voltage level (i.e., V_{15} in QLC), a new page is allocated, signifying the exhaustion of the current page's reprogramming capacity.

Decoding on reading. When reading data, the I/O Router determines its corresponding encoding scheme based on the latest hotness of the LPA and decodes the data accordingly (note that reads do not affect the LPA's hotness in RouterSSD).

D. Managing SSD Components

The flexible one-to-many mapping and reprogramming introduce new designs for key SSD components.

Write cache. Since WOM-v encoding maps one logical page to multiple physical pages, RouterSSD extends each write-cache entry (`w_ctx`) to the physical-page granularity, so that each encoded physical page is enqueued separately and aligns with GC migration. A non-empty PPA field in `w_ctx` marks a reprogramming operation; an empty one triggers a conventional write.

Consistency and recovery. RouterSSD relies on the SSD's supercapacitors [35], [36] to atomically flush all physical pages of a logical page on power failure. For recovery, it extends the OOB area with (LPA, `idx`) → PPA metadata; scanning the

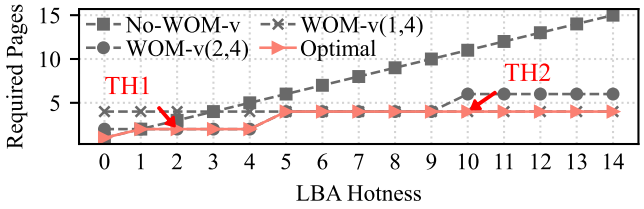

Fig. 6. **Model optimal encoding with hotness.**

OOB rebuilds the L2P table, where the PPA count per LPA reveals the encoding level and `idx` restores page ordering.

Capacity management. RouterSSD can operate within the over-provisioning space to offer a standard fixed-capacity interface. Alternatively, analogous to *compressed SSDs* whose usable capacity varies with data compressibility, RouterSSD can expose variable usable capacity in elastic mode (the more aggressively WOM-v encoding is applied, the more capacity is reclaimed), following host–SSD co-designed capacity-variant stacks [37], [38], [27].

V. Evaluation

A. Evaluation Method

We implement RouterSSD based on LightNVM [18], following established methodologies [9].

Testbed. Our experiment is carried out on FEMU [39], a widely used NVMe SSD emulator, to model a QLC SSD in the white-box mode. The emulated QLC SSD is configured with 64 GiB capacity, and the page size is configured to 4 KiB[0]. Based on the available QLC SSD latency [42], we set the flash read, write, and erase latencies to 85 μs, 1630 μs, and 3 ms, respectively; channel transfer takes 52.4 μs, and encode/decode overhead with a mature hardware computing engine is estimated within 5 μs based on prior art [12], [37], [38]. Other configurations remain as the default. The emulated QLC SSD runs on a QEMU virtual machine with 4 CPU cores, 16 GiB of DRAM, and Linux kernel 5.4.

Evaluated workload. To provide a comprehensive evaluation, we consider workloads in various scenarios, including FIU-Web, FIU-Mail, OLTP, Phone, and RocksDB, with detailed descriptions presented in §III-B. These workloads are also used in prior WOM-v work [9], covering a wide range of scenarios, including PCs, cloud servers, and mobile devices.

B. Overhead-Aware Lifetime Optimization

Lifetime and space efficiency. Figure 7 evaluates lifetime and space efficiency separately. RouterSSD, WOM-v(1,4)-SSD, and WOM-v(2,4)-SSD achieve average lifetime factors of 2.19×, 2.10×, and 1.80× over Raw-SSD, respectively. RouterSSD matches WOM-v(1,4)-SSD's lifetime while avoiding its degradation on cold-update workloads (e.g., Phone). For space efficiency, WOM-v(1,4)-SSD and WOM-v(2,4)-SSD are fixed at 0.25× and 0.5× of Raw-SSD. RouterSSD achieves 0.64× of Raw-SSD (2.6× of WOM-v(1,4)-SSD) while maintaining near-best lifetime.

OALO metric evaluation. Figure 8 reports normalized OALO under three space-sensitivity settings: small n favors space efficiency (e.g., mobile storage), while large n favors lifetime (e.g., enterprise SSDs).

- *Robust superiority across workloads.* RouterSSD consistently achieves the highest OALO, while uniform schemes degrade on specific workloads. In Fig. 8(b), WOM-v(1,4)-SSD drops to 0.28 on RocksDB yet reaches 1.01 on Mail; Raw-SSD shows the opposite trend. RouterSSD avoids both extremes via dynamic per-LPA routing, achieving gains of up to 103% over WOM-v(1,4)-SSD and consistently outperforming the best-performing uniform alternative by 3.3%, 17.0%, and 15.9% across the three settings.
- *Robust superiority across space sensitivity.* RouterSSD maintains superiority under all space-weighting settings. Raw-SSD reaches 0.97 in space-sensitive settings but declines to 0.61 in lifetime-sensitive settings; WOM-v(1,4)-SSD rises from 0.49 to 0.77 in the opposite direction, yet remains suboptimal throughout.
- *Scalability.* Figure 4 shows that as flash density scales (e.g., PLC), the theoretical OALO upper bound grows, which will widen RouterSSD's advantage compared to Raw-SSD.

C. Performance Overhead Analysis

The performance overhead of WOM-v codes is an unavoidable cost for lifetime extension. RouterSSD minimizes this cost through its *Routing and Tracking* mechanisms that add only constant-time DRAM lookups.

Routing & Tracking overhead. *(i) latency.* Routing and Tracking perform two constant-time DRAM lookups per I/O. Our measurements show the combined cost is stably $<1\,\mu$s, bounded by memory access. *(ii) memory.* RouterSSD appends two metadata fields to each L2P entry (§IV-B), adding 6 bits in total (∼15% over a 4-byte PPA). Large pages [43] and on-demand FTLs [16], [44] can further reduce this overhead.

Per-page I/O latency. Table II breaks down the single 4 KB page latency. The WOM-v encode/decode adds $\leq 5\,\mu$s [12], [37], [38]; Routing and Tracking add $<1\,\mu$s as analyzed above. Consequently, RouterSSD's per-page latency is nearly identical to that of WOM-v SSD, with <0.7% difference on both read and write paths (Table II). RouterSSD's read latency overhead relative to Raw-SSD is 4.4%; write overhead is only 0.66%. All schemes share the same pre-read cost[1].

Bandwidth trade-off analysis. Since RouterSSD is optimized for update-skewed scenarios, we replay real-world traces with random I/O for bandwidth evaluation. As shown in Figure 9, RouterSSD consistently outperforms WOM-v(1,4)-SSD and matches or approaches WOM-v(2,4)-SSD in both read and write bandwidth. For update-intensive workloads (e.g., Homes), RouterSSD trades bandwidth for lifetime while maintaining competitive average performance.

Figure 10 further confirms this adaptivity over time: RouterSSD's throughput dynamically adjusts between Raw-

[0]We use 4KiB due to emulator restrictions, which is the same as many prior works [40], [12], [37], [38], [41]. Note that RouterSSD does not rely on the exact size of the SSD page; it also works on the emerging large page.

[1]The read-modify-write pattern is common in SSDs due to page-unaligned writes and out-of-place updates; thus, pre-read latency is inherited from the read path for all schemes.

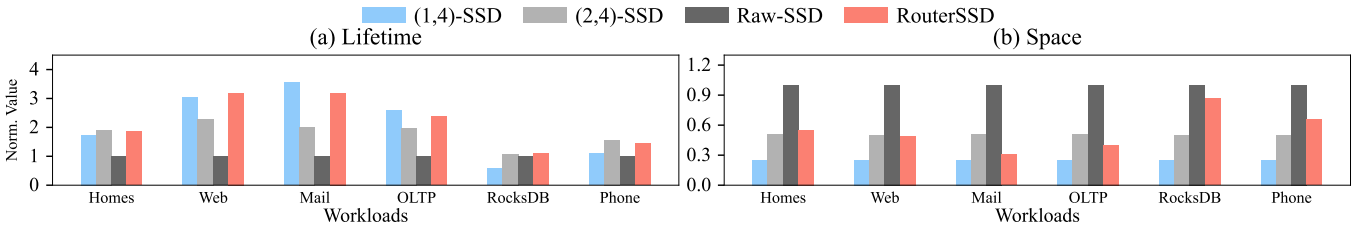

Fig. 7. **Separate evaluation of lifetime and space efficiency.** All reported results are averaged over multiple trace replays to reduce run-to-run variation.

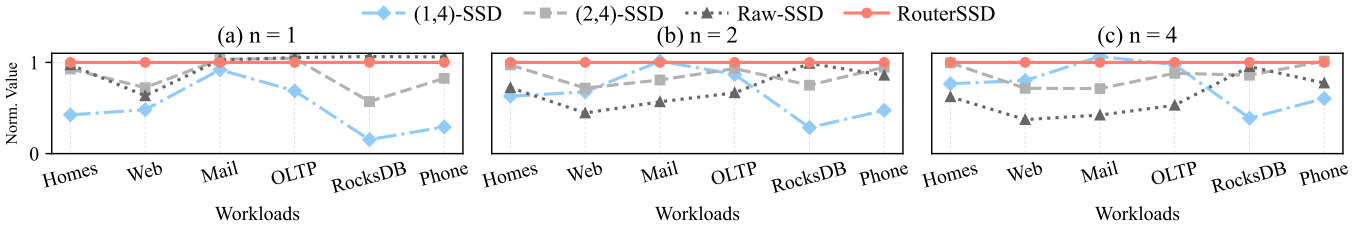

Fig. 8. **OALO metric evaluation.** We evaluate OALO(n) (§II-C) under three space-sensitivity settings, representing high, medium, and low space sensitivity. All values are normalized to RouterSSD.

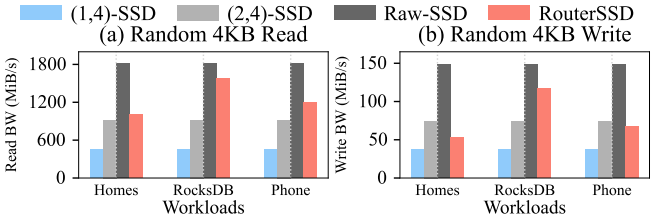

Fig. 9. **Random 4 KB read and write bandwidth across workloads.**

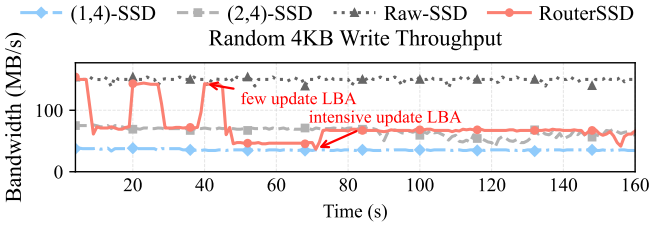

Fig. 10. **Random 4 KB write throughput over time (Homes workload).** Temporal throughput variations qualitatively reflect tail-latency behavior.

SSD level during cold phases and WOM-v level during hot bursts, remaining consistently above WOM-v-based SSDs.

VI. Limitations and Future Work

Unavoidable amplification. The overhead-aware framework in RouterSSD still incurs inevitable read/write amplification from the WOM-v's multi-page encoding. One promising direction to mitigate both overheads is *opportunistic in-storage compression*: extracting and compressing only the changed portion before applying WOM-v codes.

Hardware prototype. Realizing WOM-v codes naturally motivates a *short-pulse* ISPP (SP-ISPP) at the flash level. By injecting small charges per step, SP-ISPP enables multiple programming operations on the same wordline, tightens V_{th} distributions, and reduces program disturbance. This fine-grained hardware primitive, in turn, allows controllers to bypass rigid sequential programming constraints and unlocks new opportunities, such as in-place data refreshes via generation advancement without migrating pages.

TABLE II
Single Physical-Page I/O Latency Breakdown.

Path	Stage	Raw-SSD	WOM-v SSD	RouterSSD
Read	Flash Read	85 μs	85 μs	85 μs
	Channel Transfer	52.4 μs	52.4 μs	52.4 μs
	Decoding	n/a	$\leq$5 μs	$\leq$5 μs
	Tracking & Routing	n/a	n/a	$<$1 μs
	Total	**137.4 μs**	**$\leq$142.4 μs**	**$\leq$143.4 μs**
Write	Pre-read	137.4 μs	$\leq$142.4 μs	$\leq$143.4 μs
	Flash Program	1630 μs	1630 μs	1630 μs
	Channel Transfer	52.4 μs	52.4 μs	52.4 μs
	Encoding	n/a	$\leq$5 μs	$\leq$5 μs
	Tracking & Routing	n/a	n/a	$<$1 μs
	Total	**1819.8 μs**	**$\leq$1829.8 μs**	**$\leq$1831.8 μs**

VII. Related Work

Some studies [23], [24], [25] adopt bit-based WOM code originally designed for PROM to achieve page reuse on MLC SSDs. However, these approaches fail to work on high-density SSDs. To address this, recent research [9], [26] introduces voltage-aware WOM-v codes and applies WOM-v(2,4) to SSD secure deletion [28], [29]. Unlike these studies with uniform encoding, RouterSSD first focuses on the lifetime-space trade-off inherent to WOM-v codes, overlooked by prior work, achieving effective co-optimization.

VIII. Conclusion

Focusing on AI-driven SSD shortages, this paper finds that uniform WOM-v code leaves substantial lifetime-space co-optimization potential untapped on high-density SSDs. RouterSSD addresses this gap with hotness-aware routing of heterogeneous WOM-v codes, delivering near-optimal lifetime improvement while preserving space efficiency.

IX. Acknowledgment

This research was partly supported by the Major Key Project of PCL under Grant PCL2024A05, the National Natural Science Foundation of China under Grant 62472127, Jinan Inspur Data Technology Co., Ltd. and Shandong Information Storage System Technology Innovation Center under Grant LCCCQLJJ-2609.

REFERENCES

[1] "Micron 2500 NVMe QLC SSD," https://www.micron.com/products/storage/ssd/client-ssd/2500-ssd, 2024.

[2] "Samsung 870-qvo qlc ssd," https://semiconductor.samsung.com/us/consumer-storage/internal-ssd/870qvo/, 2024.

[3] "Solidigm D5-P5430 QLC SSD," https://www.solidigm.com/products/data-center/d5/p5430.html, 2024.

[4] A. Khakifirooz, E. Anaya, S. Balasubrahmanyam, G. Bennett, D. Castro, J. Egler, K. Fan, R. Ferdous, K. Ganapathi, O. Guzman, C. W. Ha, R. Haque, V. Harish, M. Jalalifar, O. W. Jungroth, S.-t. Kang, G. Karbasian, J.-Y. Kim, S. Li, A. S. Madraswala, S. Maddukuri, A. Mohammed, S. Mookiah, S. Nagabhushan, B. Ngo, D. Patel, S. K. Poosarla, N. V. Prabhu, C. Quiroga, S. Rajwade, A. Rahman, J. Shah, R. S. Shenoy, E. Tachie Menson, A. Tankasala, S. K. Thirumala, S. Upadhyay, K. Upadhyayula, A. Velasco, N. K. B. Vemula, B. Venkataramaiah, J. Zhou, B. M. Pathak, and P. Kalavade, "A 1.67 tb, 5b/cell flash memory fabricated in 192-layer floating gate 3d-nand technology and featuring a 23.3 gb/mm 2 bit density," *IEEE Solid-State Circuits Letters (SSCL)*, vol. 6, pp. 161–164, 2023.

[5] S. Liang, Z. Qiao, S. Tang, J. Hochstetler, S. Fu, W. Shi, and H.-B. Chen, "An empirical study of quad-level cell (qlc) nand flash ssds for big data applications," in *Proceedings of the IEEE International Conference on Big Data (Big Data)*, 2019, pp. 3676–3685.

[6] C.-Y. Liu, Y. Lee, M. Jung, M. T. Kandemir, and W. Choi, "Prolonging 3d nand ssd lifetime via read latency relaxation," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 730–742.

[7] S. McAllister, B. Berg, D. S. Berger, G. Amvrosiadis, N. Beckmann, G. R. Ganger, and Y. S. Wang, "FairyWREN: A Sustainable Cache for Emerging Write-Read-Erase Flash Interfaces," in *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024, pp. 745–764.

[8] S. McAllister, F. Kazhamiaka, D. S. Berger, R. Fonseca, K. Frost, A. Ogus, M. Sah, R. Bianchini, G. Amvrosiadis, N. Beckmann, and G. R. GANGER, "A call for research on storage emissions," *ACM SIGENERGY Energy Informatics Review*, vol. 4, no. 5, pp. 67–75, 2024.

[9] S. Jaffer, K. Mahdaviani, and B. Schroeder, "Improving the Reliability of Next Generation SSDs using WOM-v Codes," in *Proceedings of the 20th USENIX Conference on File and Storage Technologies (FAST)*, 2022, pp. 117–132.

[10] "FIU IODedup traces," http://iotta.snia.org/traces/block-io/391, 2010.

[11] H. Wu, E. Xu, L. Wang, Y. Hong, C. Niu, B. Shi, L. Zhu, J. He, D. Wu, W. Zhang, Q. Wang, C. Wang, X. Chen, G. Xue, Y.-C. Chen, and D. Ding, "Hey hey, my my, skewness is here to stay: Challenges and opportunities in cloud block store traffic," in *Proceedings of the Twentieth European Conference on Computer Systems (Eurosys)*, 2025, pp. 736–752.

[12] X. Zhang, J. Li, H. Wang, K. Zhao, and T. Zhang, "Reducing Solid-State Storage Device Write Stress through Opportunistic In-place Delta Compression," in *Proceedings of 14th USENIX Conference on File and Storage Technologies (FAST)*, 2016, pp. 111–124.

[13] J.-W. Hsieh, T.-W. Kuo, and L.-P. Chang, "Efficient identification of hot data for flash memory storage systems," *ACM Transactions on Storage (TOS)*, vol. 2, no. 1, pp. 22–40, 2006.

[14] C. Lee, D. Sim, J. Hwang, and S. Cho, "F2FS: A new file system for flash storage," in *Proceedings of the 13th USENIX Conference on File and Storage Technologies (FAST)*, 2015, pp. 273–286.

[15] S. McAllister, B. Berg, J. Tutuncu-Macias, J. Yang, S. Gunasekar, J. Lu, D. S. Berger, N. Beckmann, and G. R. Ganger, "Kangaroo: Caching billions of tiny objects on flash," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP)*, 2021, pp. 243–262.

[16] J. Sun, S. Li, Y. Sun, C. Sun, D. Vucinic, and J. Huang, "Leaftl: A learning-based flash translation layer for solid-state drives," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2023, pp. 442–456.

[17] S. Wang, Z. Lin, S. Wu, H. Jiang, J. Zhang, and B. Mao, "Learnedftl: A learning-based page-level ftl for reducing double reads in flash-based ssds," in *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024, pp. 616–629.

[18] M. Bjorling, J. Gonzalez, and P. Bonnet, "LightNVM: The Linux Open-Channel SSD Subsystem," in *Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST)*, 2017, pp. 359–374.

[19] S. Liu and X. Zou, "QLC NAND study and enhanced Gray coding methods for sixteen-level-based program algorithms," *Microelectronics Journal*, vol. 66, pp. 58–66, 2017.

[20] A. Buck, K. Ganesan, and N. E. Jerger, "Flipbit: Approximate flash memory for iot devices," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2024, pp. 876–890.

[21] S. Cho, B. Kim, H. Cho, G. Seo, O. Mutlu, M. Kim, and J. Park, "AERO: Adaptive Erase Operation for Improving Lifetime and Performance of Modern NAND Flash-Based SSDs," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2024, pp. 101–118.

[22] D. Hong, M. Kim, G. Cho, D. Lee, and J. Kim, "Guardederase: Extending ssd lifetimes by protecting weak wordlines," in *20th USENIX Conference on File and Storage Technologies (FAST 22)*, 2022, pp. 133–146.

[23] G. Yadgar, E. Yaakobi, and A. Schuster, "Write once, get 50% free: Saving SSD erase costs using WOM codes," in *Proceedings of the 13th USENIX Conference on File and Storage Technologies (FAST)*, 2015, pp. 257–271.

[24] F. Margaglia and A. Brinkmann, "Improving MLC flash performance and endurance with extended P/E cycles," in *Proceedings of the 31st Symposium on Mass Storage Systems and Technologies (MSST)*, 2015, pp. 1–12.

[25] F. Margaglia, G. Yadgar, E. Yaakobi, Y. Li, A. Schuster, and A. Brinkmann, "The Devil Is in the Details: Implementing Flash Page Reuse with WOM Codes," in *Proceedings of the 14th USENIX Conference on File and Storage Technologies (FAST)*, 2016, pp. 95–109.

[26] S. Jaffer, K. Mahdaviani, and B. Schroeder, "Improving the Endurance of Next Generation SSD's using WOM-v Codes," *ACM Transactions on Storage (TOS)*, vol. 18, no. 4, pp. 1–32, 2022.

[27] Z. Jiao, X. Zhang, H. Shin, J. Choi, and B. S. Kim, "The design and implementation of a capacity-variant storage system," in *22nd USENIX Conference on File and Storage Technologies (FAST)*, 2024, pp. 159–176.

[28] J. Cui, K. Tang, and L. T. Yang, "A fast secure deletion strategy for high-density flash memory through wom-v codes," in *Proceedings of the 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[29] K. Tang, J. Cui, C. Wen, S. Nie, D. Liu, Y. Zhao, and L. T. Yang, "Wom-ftl: An efficient ftl for high-density flash memory through wom-v codes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2025.

[30] R. Koller and R. Rangaswami, "I/O deduplication: Utilizing content similarity to improve I/O performance," *ACM Transactions on Storage (TOS)*, vol. 6, no. 3, pp. 1–26, 2010.

[31] G. Yadgar, M. Gabel, S. Jaffer, and B. Schroeder, "SSD-based workload characteristics and their performance implications," *ACM Transactions on Storage (TOS)*, vol. 17, no. 1, pp. 1–26, 2021.

[32] "YCSB RocksDB SSD traces," http://iotta.snia.org/traces/28568, 2020.

[33] Q. Yang, R. Jin, and M. Zhao, "Smartdedup: Optimizing deduplication for resource-constrained devices," in *USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 633–646.

[34] "OLTP Application I/O," https://traces.cs.umass.edu/docs/traces/storage/, 2002.

[35] X. Liao, Y. Lu, E. Xu, and J. Shu, "Write dependency disentanglement with HORAE," in *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2020, pp. 549–565.

[36] V. Chidambaram, T. Sharma, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Consistency without ordering," in *Proceedings of the 10th USENIX Conference on File and Storage Technologies (FAST)*, vol. 12, 2012, pp. 101–116.

[37] Y. Wang, T. Lu, Y. Liang, X. Chen, and M.-C. Yang, "Reviving in-storage hardware compression on zns ssds through host-ssd collaboration," in *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2025, pp. 608–623.

[38] 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 (DAC)*, 2024, pp. 1–6.

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

[40] G. Wu and X. He, "Delta-FTL: Improving SSD lifetime via exploiting content locality," in *Proceedings of Proceedings of the 7th ACM European Conference on Computer Systems (EuroSys)*, 2012, pp. 253–266.

[41] Y. Park and J.-S. Kim, "zFTL: Power-efficient data compression support for NAND flash-based consumer electronics devices," *IEEE Transactions on Consumer Electronics (TCE)*, vol. 57, no. 3, pp. 1148–1156, 2011.

[42] A. Khakifirooz, S. Balasubrahmanyam, R. Fastow, and K. Gaewsky, "30.2 A 1Tb 4b/Cell 144-Tier Floating-Gate 3D-NAND Flash Memory with 40MB/s Program Throughput and 13.8Gb/mm2 Bit Density," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 424–426.

[43] Y. Zhou, E. Xu, L. Zhang, K. Karkra, M. Barczak, W. Gao, W. Malikowski, M. Kozlowski, Ł. Łasek, R. Lu, F. Yang, L. Huang, X. Zhang, K. Niu, J. Zhu, and J. Wu, "CSAL: the Next-Gen Local Disks for the Cloud," in *Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys)*, 2024, pp. 608–623.

[44] S. Wang, Z. Lin, S. Wu, H. Jiang, J. Zhang, and B. Mao, "Learnedftl: A learning-based page-level ftl for reducing double reads in flash-based ssds," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2024, pp. 616–629.