SSD-iq: Uncovering the Hidden Side of SSD Performance

Gabriel Haas
Technische Universität München
University of Copenhagen
gabriel.haas@tum.de

Bohyun Lee
Technische Universität München
bohyun.lee@tum.de

Philippe Bonnet
University of Copenhagen
bonnet@di.ku.dk

Viktor Leis
Technische Universität München
leis@in.tum.de

ABSTRACT

SSDs are hardware and software systems whose design involves complex and undocumented trade-offs between cost, energy consumption and performance. This complexity is hidden behind standard interfaces and a few headline specifications, such as capacity, sequential, and random performance. As a result, database system designers often assume that SSDs are interchangeable commodities and regularly use a single SSD model to evaluate database performance. Does it matter which SSD model is provisioned for a database system? If yes, how to choose the right one? These are the questions we address in this paper. We study the performance characteristics of commercial data center SSDs, highlighting the limitations of current standard metrics in capturing their true behavior. We conduct experiments on nine SSDs from major vendors, revealing significant differences in performance despite similar headline specifications. We show that the choice of SSD matters for database system performance. We propose a new benchmark, SSD-iq, which introduces four additional metrics to better characterize SSD performance, particularly for write-intensive workloads. Incidentally, our work should encourage vendors to optimize SSD controllers using more comprehensive and transparent performance criteria.

PVLDB Reference Format:
Gabriel Haas, Bohyun Lee, Philippe Bonnet, and Viktor Leis. SSD-iq: Uncovering the Hidden Side of SSD Performance. PVLDB, 18(11): 4295 - 4308, 2025.
doi:10.14778/3749646.3749694

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/gabriel-haas/ssdiq.

Table 1: Specifications of five data center NVMe SSDs.

	Samsung PM9A3 960 GB	Hynix PE8110 960 GB	Intel D7 P5520 1,920 GB	Micron 7450PRO 960 GB	WD DC SN640 960 GB
Seq. Read [MB/s]	6,500	6,500	5,300	6,800	3,330
Seq. Write [MB/s]	1,500	1,700	1,900	1,400	1,190
Rnd. Read [MB/s]	580	900	700	530	434
Rnd. Write [MB/s]	70	70	114	85	49
Read Latency [µs]	?	75	75	80	78
Write Latency [µs]	?	?	15	15	?

1 INTRODUCTION

Flash has taken over. Over the past ten years, most data-intensive systems have switched from magnetic hard disk drives to flash-based SSDs. This transition has been fueled by increasing SSD performance and power efficiency at comparable cost, and eased by standards and software-level backward compatibility: SSDs emulate hard disk drives, even though the underlying architecture and storage media have nothing in common. While the characteristics of hard disk drives were uniform across vendors, the same is not true for SSDs. SSD models that have the same capacity and the same underlying NAND flash technology (e.g., MLC or TLC) embody varying undocumented trade-offs between cost, energy consumption and performance [41].

SSD models are different. Vendors downplay the idiosyncrasies of specific SSD models by marketing their devices using four "headline" throughput metrics: sequential read, sequential write, random read, and random write. The silent assumption is that SSDs can be meaningfully compared using these metrics. Table 1 shows the specifications for five data center NVMe SSDs containing TLC flash. All vendors specify the four aforementioned throughput metrics, and some additionally provide the latency of read and write operations, though this is far from universal. It is also often not clear under which settings one can achieve the stated performance, with some vendors qualifying their metrics as "Up To". While the superficial problems of existing metrics could be addressed by standardizing benchmarking settings, a more fundamental problem is that these standard metrics cannot capture the complex nature of flash SSDs. Neither write amplification nor latency under load is captured by these metrics, although being essential for the lifespan of the devices and the end-to-end performance of the system.

Write Amplification. Magnetic hard disk drives support in-place writes. NAND flash does not. Hence, writes are performed out-of-place in SSDs. To create the illusion of in-place writes, the SSD controller internally implements complex logic, which handles tasks such as logical-to-physical mapping and Garbage Collection (GC) [41]. GC kicks in once the internal capacity is exhausted. The GC process selects and erases multi-megabyte blocks consisting of hundreds of pages. If a block contains valid pages, these need to be moved to a new location before erasing, which results in write

Proceedings of the VLDB Endowment, Vol. 18, No. 11 ISSN 2150-8097.
doi:10.14778/3749646.3749694

amplification. In an analysis of 20 cloud storage workloads [44], Alibaba observed write amplification factors (WAF) as high as 8.0, with a median WAF of 2.5 and an average WAF of 3.0. A recent study of NetApp's enterprise storage systems reports WAFs of over 10 for more than a third of their SSDs [45]. Such high WAFs mean that every logical write causes multiple additional internal writes, reducing write throughput and device lifetime. SSD lifetime is directly linked to the sustainability of the entire system [50]. Thus, it is a crucial consideration for database system architects. Note that we do not claim that vendors ignore write amplification. Recent vendor white papers and research papers (e.g., [2, 17]) focus on it, in connection with the introduction of NVMe Flexible Data Placement (FDP). Our point is that write amplification is still not part of the standard metrics used in datasheets.

Latency under Load. Besides write amplification, SSDs also exhibit complex latency characteristics under load. For example, to hide high physical write latency on NAND flash (around 0.5ms), writes are buffered in DRAM on the SSD [41]. This results in a user-observable write latency of around 15 µs on the host. However, depending on the current write throughput and on how buffering is implemented, interferences occur that cause write tail latency to spike in unpredictable ways. This is especially problematic for database system write-ahead logging (WAL). Interferences not only impact other writes, but also reads [42]. One major source of tail latency is the flash erase operation, which takes several milliseconds. While reads normally take around 80 µs, if the target data resides on a flash chip currently performing an erase operation, read latency may increase to several milliseconds. To mitigate this issue, some SSDs have support for suspending ongoing erase commands [37]. How SSDs handle these complex interactions is also not captured by standard metrics.

Existing metrics are not enough. Can simple experiments be used to characterize the way a SSD handles write amplification and latency under load? Do they matter for database performance? How can one decide which SSD model is best-suited for a particular workload? The goal of this paper is to better understand the behavior of commercial data center SSDs, provide more transparency when choosing devices, and set better incentives for SSD vendors. Since commercial SSDs are largely black boxes, we perform carefully designed experiments to understand their internal behavior. In this endeavour, we are helped by the Open Compute Project (OCP) NVMe interface which provides insights into SSD internals, like physical media writes [57]. This interface is supported by recent data center SSD models.

Our key finding: SSDs behave very differently in practice. Our experiments with nine commercial data center SSDs from all five major vendors show that they behave very differently, despite similar headline metrics and flash technologies. For example, Table 1 shows five SSD models with comparable performance characteristics. However, our study shows that two of these five SSD models actually exhibit much worse latency behavior under load—making them suboptimal choices for latency-critical applications such as OLTP database systems. You could not tell which ones from the datasheet numbers. We also find that many SSD models rely on simple (Greedy-like) garbage collection algorithms rather than more sophisticated algorithms proposed in the literature. For skewed write-intensive workloads, the models that implement more intelligent algorithms benefit substantially in terms of performance and device lifetime.

Introducing the SSD-iq benchmark. Our findings indicate that for write-intensive workloads, the experiments associated to the six standard performance metrics are insufficient. Note that there is no standard industry benchmark for SSDs comparable to TPC-C [60] for OLTP. Customer-grade SSD benchmarks from specialized websites (e.g., Tom's Hardware, Anandtech, Storage Review) have evolved to become long-running processes, but they are designed to replicate I/O-intensive desktop usage, not write-intensive data center workloads. For instance, the Destroyer benchmark from Anandtech [61], would not overwrite any of the SSDs we evaluated and therefore fail to achieve a steady state. There is no popular testbed either, comparable to FileBench [59] for file systems, or Benchbase [14] for database systems. There is a popular tool, fio [4], for generating synthesized workloads as well as submitting I/Os and measuring their latency. This tool, developed by Jens Axboe, is widely used for SSD performance analysis in the industry and in academia [16, 20, 21, 36, 67]. It is trivial to use fio to measure the six standard metrics. However, as Bouganim et al. observed long ago [7], characterizing SSD performance in a meaningful and reproducible way is more challenging. We therefore propose the *SSD-iq* benchmark, which not only standardizes the benchmarking setup but also provides simple experiments and new metrics for evaluating and comparing SSDs.

2 RELATED WORK

Understanding SSD performance. Characterizing the performance of SSDs has been a topic of interest for many years. Works like [7, 25, 32] share a common goal with us in aiming to better understand the performance and internal behavior of black-box SSDs. They were introduced during the early development of flash SSDs, and are designed to assess SSD performance under various access settings such as alignment, granularity, locality, order, bursts. He et al. [25] define these access patterns in an *unwritten contract* that is based on the SSD specific design. Violations of this contract will lead to suboptimal SSD performance. They evaluated the performance of filesystems and storage engines on an SSD and looked at contract violations in their simulator. However, they only use 1 GB partition of an SSD and did not investigate the steady-state (*sustained*) impact of these access patterns. Kakaraparthy et al. [32] tried to learn the internal SSD design by black-box testing. For example, they do this by observing how block size and alignment influences latency and bandwidth. These learnings they then applied to optimize the write (and hence read) patterns of database systems. Our study on the other hand, evaluates state-of-the-art SSD models using workloads chosen to expose subtle differences in garbage collection and operation execution in steady-state condition.

Long-running Workloads. The workloads used to characterize SSD performance range from synthetic, to traces and applications in production. A key characteristic of these workloads is their duration, measured in hours, power-on years or number of times an SSD is overwritten. Long-running experiments have been of particular interest to exhibit interesting characteristics (e.g., throughput fluctuations [36], write amplification [27], or garbage collection behavior [67]).

JEDEC. The industry standard JESD218 [28] by JEDEC defines endurance testing requirements and methods for SSD vendors to evaluate SSD endurance. JESD219 [29] contains detailed descriptions and clearly defines workloads for enterprise-grade and customer-grade SSDs. The enterprise workload is synthetic, with varying access granularity from 512 Byte to 64 KB, with most access being 4 KB. Further, it divides the LBA space in three *constant* groups with different access frequencies, similar to our Two-Zone and Read-Only workload. The advantage of our patterns is that it can be adjusted using a single parameter (e.g., shifting from 60/40 to 90/10) to expose resulting variations in performance and WAF. The client endurance workload is based on a trace which we discuss in Section 3.2. These standards were created to establish a defined way to measure SSD endurance with two static workloads. They were not intended to show other differences in SSD behavior. We further consider the role of the workload in SSD benchmarking methodology in Section 3.

WA studies. Since write amplification is one of the most influential factors in SSD performance, various efforts have been made to understand its behavior within SSDs. The foundational works like [27] explores write amplification and the impact of spare factor on WAF. In an attempt to better understand the impact of garbage collection on write amplification, extensive research [11, 13, 40, 68] provides theoretical models of write amplification, while others [10, 34, 36, 58, 67] propose improved garbage collection strategies or data placement techniques to mitigate WA. While our paper also simulates several garbage collection algorithms, our main objective is to demonstrate that these algorithms yield different WAF values under the same workloads. We also present opportunities for improving these strategies by comparing WAF with the optimal value (refer to Figure 3b and Figure 4b). Additionally, we show that the effectiveness of garbage collection algorithms is highly dependent on workload skewness, as evidenced not only in simulated WAF but also on real SSDs.

Wear leveling. Although we left wear leveling out of the scope of this paper, as we focus on write-intensive workloads, it is worth noting that wear leveling can significantly impact WAF under read-heavy workloads. This is because cold data must be relocated to balance wear on flash blocks during periods of low write activity. As a result, various studies have proposed wear leveling-aware data placement strategies from the application level [8, 9], or studied wear leveling algorithms for flash SSDs [31, 49].

Host-side hinting interfaces. SSD vendors have also proposed various architectures and data placement interfaces to reduce WAF at the SSD level. Open-channel SSDs [6] provide full host visibility into device topology, while ZNS SSDs [5] allow the host to manage data placement and garbage collection. Both designs aim to improve performance and predictability by leveraging the host's greater knowledge of the stored data. In terms of data placement, multistream SSDs [33] utilize stream IDs provided by the user as hints for more intelligent data placement. FDP-enabled SSDs [47] also focus on host-aware data placement, offering even more information to the host. We believe benchmarking these SSDs would also provide valuable insights, and hence leave this exploration as future work.

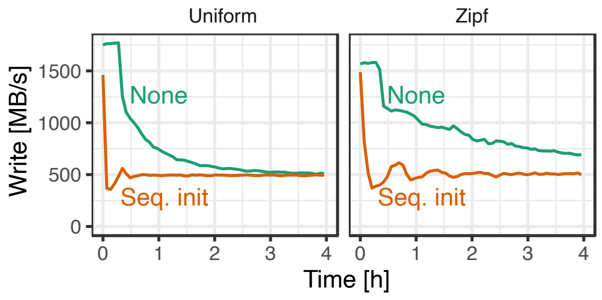


Figure 1: Steady-state performance vs. initialization.

SSD tail latency studies. Various studies have focused on storage tail latency, as it is a key concern for end users. To address this issue, several works [6, 15, 24, 35] present measured tail latency on SSDs in both on-premise and cloud environments. To improve SSD latency itself, TTFlash [66] attempts to reduce garbage collection-induced tail read latency, while [62] proposes ways to reduce read latency by reducing the overhead of the P/E suspension scheme.

Summary. Despite this large body of work, there is still no industry benchmark for data center SSDs. With SSD-iq, our goal is to define a framework for systematically comparing commercial SSDs in a way that is relevant for database architects.

3 METHODOLOGY

In this Section, we describe how write-intensive SSD benchmarks must be done to achieve conclusive and reproducible results.

3.1 How to Benchmark SSDs

Complex internal state. SSDs have complex internal logic to manage NAND flash memory and to try to hide the resulting undesirable properties from users. This is the responsibility of the flash translation layer (FTL). The internal state determines performance characteristics; for instance, an empty SSD will have much higher write bandwidth compared to a fully written SSD that must perform Garbage Collection (GC) before new data can be written.

Reproducibility. To achieve reproducible results when experimenting with SSDs, it is essential to control the initial state of the experiment. However, SSDs do not provide any control over physical data placement or garbage collection, which makes it impossible to know the current state. Therefore, the only way to ensure a known and consistent state across all SSDs is to completely reset it. Hence, before every experiment, the SSD must be fully erased, including *all* flash blocks and the FTL mappings. This can be done using the NVMe sanitize block erase action [54].

Steady state. By erasing the SSD, we ensure that all benchmarks start from a well-defined and comparable state. The next consideration is how to run benchmarks to achieve consistent and conclusive results, specifically it is important to determine the time required for workloads to converge to a steady state. Figure 1 shows the write performance over several hours for random and skewed workloads. When the SSD is fully written sequentially ("Seq. init" in the figure) before the actual workload, the performance converges quickly.

Without this initialization step ("None"), convergence takes significantly longer. In particular, when running the Zipf (0.8) distribution, it takes an unfeasibly long time. For uniform access patterns, convergence without initialization takes more than 4 hours, equivalent to over 6 full disk writes. With sequential initialization, steady state is reached in less than 1 hour with fewer than 1 additional disk overwrite beyond the initial sequential write. Sequential initialization is beneficial, as it prevents GC from making use of unwritten pages. Unwritten pages (Logical Block Addresses, or LBAs) can otherwise be utilized by the GC process just like Over-Provisioned (OP) space. In all experiments in this paper, we use the sequential initialization method up to the desired fill level, after successful erasure. This approach allows the workloads to reach steady state much faster, reducing benchmark duration and wear on the SSD.

3.2 Traces and Workloads

I/O traces. To bridge the gap between the internal behavior of SSDs and the applications that run on them, many studies rely on I/O block traces. These traces are utilized under the assumption that they represent real-world workloads. However, many commonly used open-source traces [30, 39, 52, 56] were created when hard disks were primarily used as storage devices and have been used ever since. To better reflect modern SSD-based I/O workloads, more recent traces have been provided [44, 45, 63, 65, 68], including those from cloud storage environments [43, 55].

Choosing synthetic workloads over I/O traces. We analyzed existing traces to determine whether they are suited for observing how each SSD behaves under such workloads. The following table shows the most important characteristics of these traces (total number of writes, the unique pages that are accessed in the trace, and the difference between highest and lowest page):

Trace	Total Writes (GB)	Dataset Size (GB)	Data Range (GB)
MSR [48]	144.5	1.7	16.2
FIU [38]	1,561.0	8.4	278.5
RocksDB [64]	1,342.8	161.9	476.5
Alibaba [55]	7,443.8	346.9	500.0
JEDEC client [29]	781.3	49.0	128.0

Based on these numbers, we can conclude that these traces are inadequate for our purposes. First, the traces have a static size, making them unsuitable for testing SSDs with varying capacities. For instance, running the same Alibaba trace on a 1,920 GB Intel D7 P5520 and a 960 GB Micron 7450PRO SSD would lead to an unfair comparison due to differences in the relative fill factor. Second, except for the Alibaba trace, none of the traces contained sufficient number of write requests to reach a steady state, or even cover a significant portion of the LBA range. The JEDEC JESD219A [29] SSD Endurance Workloads specification defines a synthetic enterprise and a trace-based client workload. The client workload shown in the table, is based on 7 months of SSD activity on a 128 GB SSD. The specification also includes instructions on how the trace can be scaled to larger SSDs, by expanding the cold (read-only) areas, keeping the written data constant at 49 GB. In the era of multi-Tera-Byte SSDs, these workloads are unsuitable for our research,

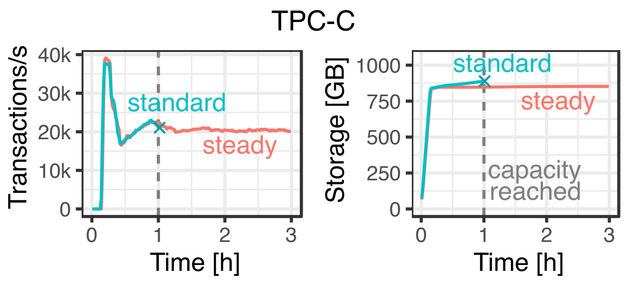


Figure 2: TPC-C with LeanStore on Samsung PM9A3 (64 GB buffer pool, 128 worker threads, 6550 warehouses ≈ 800 GB).

as we are interested in write-heavy, enterprise workloads. Interestingly, the JESD219A standard itself also defines the enterprise workload in a synthetic manner with multiple groups of different access frequency. In conclusion, every trace is obtained from a different application with varying levels of skewness, I/O request sizes, and read/write ratios; this makes it difficult to reason about internal SSD behavior from it.

Micro-benchmarks vs. macro-benchmarks. TPC-C is a write-heavy workload that models an order-processing system for a wholesale supplier and YCSB mimics a simple key/value workload. While both are often used to benchmark OLTP database systems, we argue that they are not ideal for SSD benchmarking. The TPC-C database grows during the benchmark run, which makes it impossible to measure steady-state write amplification. Figure 2 shows this behavior, with the system running the growing TPC-C workload for only an hour until the SSD is full. To get around this, we developed a variant of TPC-C that truncates growing tables as part of the workload, so that space usage remains constant. The figure also shows that *Steady TPC-C* can be run indefinitely and we can get to a steady-state, where database throughput settles. While this is an interesting result, it is difficult to further adapt TPC-C to expose SSD-internal behavior. Further, this introduces application specific noise that is difficult to disentangle from SSD behavior. For example, most database system will be CPU bound before an NVMe drive will be saturated, which makes it unclear what is actually being measured. Similarly for the YCSB workload, which uses the Zipf distribution for skewed accesses, it is easier to directly model the Zipf access pattern directly on the SSD without a database on top. Thus, instead of using complex workloads, we intentionally opt to use well-defined and understandable synthetic workloads in the first part of the paper. This will enable us to expose and understand differences in SSD behavior. We will then confirm these behaviors in Section 6.3 using end-to-end TPC-C and YCSB benchmarks.

3.3 Tool & experimental setup

***Benchmarking Tool:* `iob`.** We are using our own, custom benchmarking tool `iob`, which collects a wide range of statistics, including detailed latency statistics, SMART [23] and OCP counters. iob is a lightweight and efficient tool that handles high IOPS per thread. `iob` is very flexible and supports many access patterns and customizations like threads, I/O-depth, buffered I/O, block size, etc. It also supports multiple I/O backends, like, libaio, io_uring, and

SPDK. SSD-iq is composed of bash scripts that call iob with specific I/O patterns and characteristics. The bash scripts we used for the paper and the iob tool are available in the SSD-iq repository: https://github.com/gabriel-haas/ssdiq. The extensibility and maintainability of SSD-iq is based on the flexibility of the iob benchmarking tool and the simplicity of the scripts.

NVMe setup & aging. We consider NVMe SSDs directly attached to the host over PCIe. In addition to the five PCIe 4.0 SSDs with TLC NAND flash, listed in Table 1, the server contains Kioxia CM7-R (PCIe 5.0) and Micron 7450 Max (PCIe 4.0) SSDs, which are included in the results in Table 2. At the start of this work, all SSDs were essentially new. Throughout this work they were used extensively with their *percentage used* in SMART now ranging from 40% to 60%. To understand the effect of aging we run a long running experiment on a 2 TB Intel D7 P5520 SSD until it switched into read-only mode. This happened after more than 6 months, with more than 4000 logical, uniform random drive writes, or 8 PB written to the SSD. The most interesting fact about this was that it achieved double the guaranteed 1800 drive writes (5 years * 1 DWPD) and this with 26 PB of physical writes (WAF of 3.2). For the duration of the experiment we did not observe any significant differences in read or write latency, nor write-amplification, until it went into read-only mode (where writing at 2 MB/s is still possible).

Hardware & software setup. Most of the experiments in this paper are conducted on a server equipped with an AMD EPYC 9654P 96-Core processor and 384 GB DRAM running Linux v6.8.0. Some additional experiments were run on a similar remote server equipped with Samsung PM1733 SSDs (PCIe 4.0) and i4i instances in AWS with NVMe instance storage. For all experiments in this paper we used `iob` with the io_uring backend and non-buffered I/O (`O_DIRECT`) directly on the block device using 4 KiB pages.

4 WRITE AMPLIFICATION

Most of the complex performance characteristics exhibited by SSDs are directly or indirectly caused by writes and how they are managed by the SSD controller. In this Section, we experimentally investigate the write amplification factors (WAF) on data center SSDs caused by different workloads. For this we designed a series of experiments to dissect the behavior of SSDs.

4.1 Background

Device lifetime. Since flash storage has a limited number of program/erase cycles, the total volume of physical writes determines the device's lifetime. With a write amplification factor (WAF) of 2, the SSD internally writes twice as much data as it receives from the host, which effectively halves the SSD lifespan. SSD manufacturers generally provide warranties in terms of Disk Writes Per Day (DWPD) over a period of five years. For example, most devices in this paper guarantee 1 logical DWPD, implying that customers may overwrite the disk $1 \times 365 \times 5 = 1825$ times. However, the true number of overwrites may substantially vary depending on WAF.

Causes of write amplification. Write amplification is determined by the workload, the over-provisioning capacity, and the garbage collection algorithm. While the workload and the over-provisioning capacity are outside the control of the SSD controller, the garbage collection algorithm is a key component of the FTL and has a significant impact on WAF.

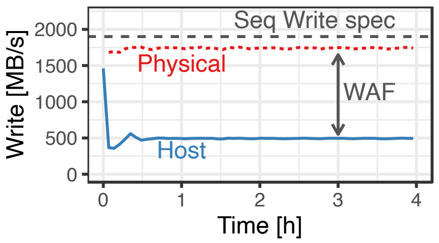
Measuring write amplification with OCP. The Open Compute Project (OCP) [57] recently introduced an NVMe extension that provides insights into SSD internals. Many data center SSDs support OCP, including several devices that we compare in this paper. OCP specifies metrics for physical flash writes, flash block erases, number of error corrections at various levels, and bad blocks. For our purposes, the physical write counter is particularly useful, as it allows calculating WAF as follows:

$$\text{WAF}_{\text{measured}} = \frac{physWrites}{hostWrites}$$

Implied write amplification. For SSDs that do not support OCP, we can estimate WAF by observing the drives write performance. Specifically, we compare the maximum write bandwidth for a particular access pattern with the maximum sequential write bandwidth:

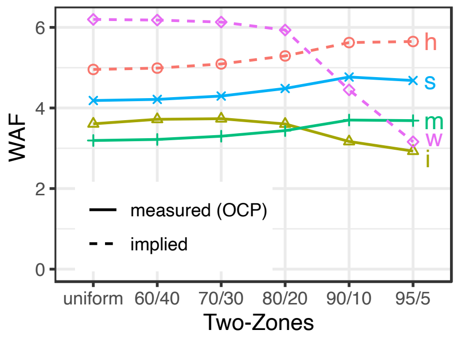
$$\text{WAF}_{\text{implied}} = \frac{bw_{seq}}{bw_{measured}}$$

This can be done because sequential write performance is close to the internal write bandwidth ($bw_{seq} \approx bw_{phy}$). SSD vendors want to achieve high sequential write performance in micro-benchmarks and their datasheet, close to what is physically possible. There are other factors that reduce host writes throughput, like wear leveling, but that has a small impact on WAF [26]. Hence, the formula assumes that WAF is close to 1 for sequential writes and that the slowdown in the measured workload is caused by write amplification. Using the SSDs that do support OCP, we found the former assumption to be true for all tested SSDs, and the latter assumption holds well enough to be useful. The following figure visualizes this on the example of the Intel SSD:

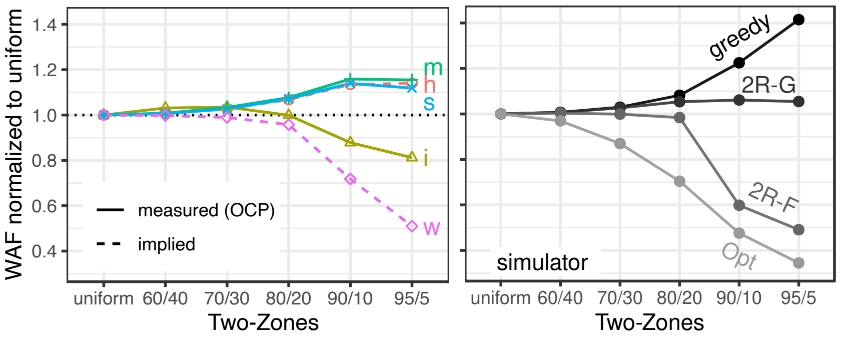


In this case, the sequential write spec and the physical writes measured using OCP counters are comparable. The estimation results in an implied WAF of 3.81, compared to the WAF measured through OCP of 3.47. Thus, while this estimation technique is not perfect, it is still useful and works for any SSD.

GC algorithms. The most salient aspect of a GC algorithm is the decision on which block to erase next. A straightforward approach is to select the block with the fewest valid pages. This *greedy* algorithm has been shown to be optimal for uniform random workloads [40] but does not perform well for skewed workloads [34]. More sophisticated algorithms based on hot/cold data separation, explicit data placement, and explicit partitioning of over-provisioning capacity [10, 11, 13, 34, 40, 58, 68] exploit skew in the access distribution and have been shown to reduce WAF substantially. Given the existence of these algorithms, the importance of WAF, and the fact that many real-world workloads are skewed [65], one would expect that state-of-the-art data center SSDs employ intelligent GC algorithms.



(a) Absolute WAF on data center SSDs.



(b) Normalized WAF for SSDs and GC algorithms in simulator.

Figure 3: Write amplification with increasing skew on two-zones workload.

4.2 Two-Zones Workload

Workload definition. To investigate how skew affects GC, we define a simple workload consisting of two zones and a single parameter that determines the level of skew. This parameter governs both the write frequency and the size of each zone. For example, a 70/30 setting means that 70% of all writes are directed to the first zone, which occupies 30% of the storage space. A 50/50 two-zone setting is equivalent to uniform random writes, where 50% of the writes are directed at 50% of the SSD capacity, and the other half receives the rest. As a result, writes are uniformly distributed across the entire SSD capacity. A 95/5 setting represents the other extreme with very high skew, there, 95% of all writes target only 5% of the storage. Within each zone, writes are uniformly distributed, but the two zones are not placed sequentially on the SSD. Instead, they are randomly scattered across the entire SSD range. The scrambling prevents frequently accessed pages from clustering in the same LBA range, making the workload slightly more challenging for SSDs.

Underwhelming SSDs. Figure 3a shows the absolute WAF for the five SSDs listed in Table 1 across increasingly skewed two-zone workloads. The SSDs are abbreviated by manufacturer name: Samsung (s), SK Hynix (h), Intel (i), Micron (m), and WD (w). Even for uniform workloads, we observe significant WAF differences, primarily due to varying over-provisioning levels. This is expected and discussed further in Section 4.6. More notably, except for the Intel and WD drives, all SSDs exhibit progressively higher WAF as skew increases. This is a surprising result and contradicts the expectation that modern SSDs employ intelligent GC algorithms. Despite decades of research into GC algorithms, many data center SSDs fail to exploit skew, even in this simple two-zone workload.

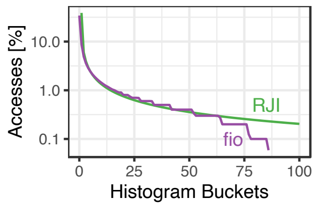
Simulator. To better understand this unexpected result, we developed an SSD simulator with the following algorithms:

- Naive Greedy: Selects the block with the fewest valid pages, compacts it, and then adds new host writes to the same block.
- 2R-Greedy (2R-G) [34]: Similar to Naive Greedy, but uses two regions; data moved by GC is written to different blocks than host writes. It performs garbage collection on one region at a time to prevent mixing hot host writes with cold GC data.
- 2R-FIFO (2R-F) [34]: Expands on 2R-G by managing SSD blocks in a FIFO list to avoid the false promotion of cold blocks, while giving hot blocks more time for pages to become invalidated.
- Optimal: This is a theoretical lower bound, obtained by exhaustively enumerating all possible ways of partitioning available over-provisioning space to groups with similar access frequency.

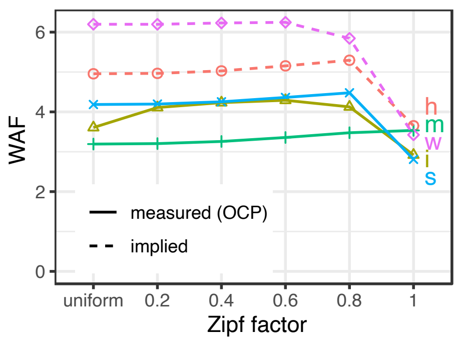
Revealing the hidden side. Figure 3b compares the SSDs with the GC algorithms implemented in the simulator. To highlight algorithmic differences rather than the influence of OP, we normalize the WAF by the WAF observed in the uniform workload (WAF_x/WAF_{unif}). For most SSDs (Micron (m), SK Hynix (h), and Samsung (s)), WAF increases with higher skew, exhibiting behavior that falls between the Naive Greedy and semi-naive (2R-Greedy) algorithms. Only the Intel (i) and WS (w) SSDs manage to somewhat exploit the skew of the two-zone workload to reduce WAF.

4.3 Zipfian Workload

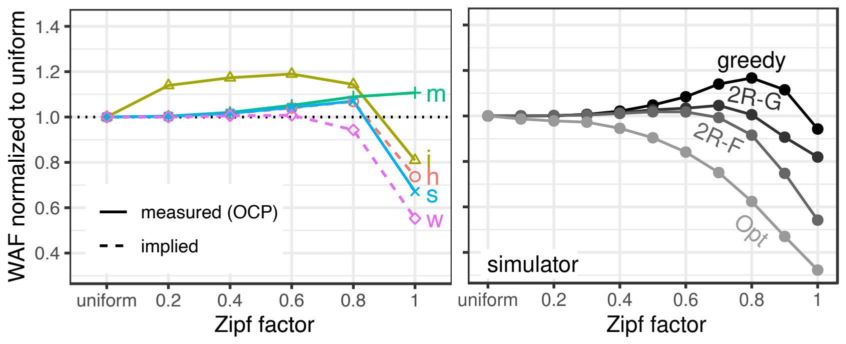
Zipf implementation. A frequently used distribution to emulate real-world access patterns is Zipf. While many implementations approximate it, the lack of detailed explanations in most research papers makes reproducibility difficult. We use the Rejection Inversion Zipf (RJI) sampler [12], ported from the Apache Commons implementation [3]. We also experimented with the Zipf generator from the fio I/O benchmarking tool [4], but found that its implementation does not access all pages. The access patterns generated by fio and RJI Zipf samplers are shown in a histogram in the figure below, based on 500 million pages (equivalent to a 2 TB SSD capacity) and 10×500M accesses:



While the distribution from the RJI sampler resembles the expected Zipfian distribution, the fio Zipf implementation introduces noticeable discrete steps. More critically, it leaves 14% of the pages untouched, which results in inaccurate performance and WAF.



(a) Absolute WAF on data center SSDs.



(b) Normalized WAF for SSDs and GC algorithms in simulator.

Figure 4: Write amplification with increasing skew on Zipf workload.

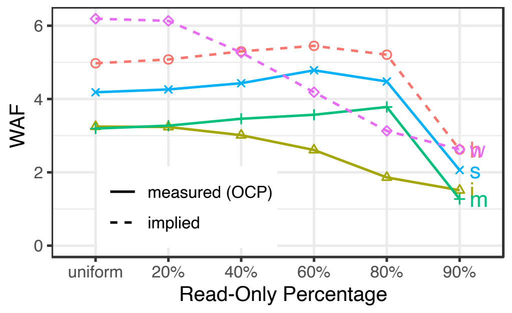


Figure 5: Write amplification with increasing read-only area.

Zipf results. Figure 4 shows the WAF with increasing Zipf factors. Similar to the two-zone workload, higher Zipf factors initially result in higher WAF on most SSDs. The Western Digital SSD (w) again shows good performance, while the Intel SSD (i), which performed well in the two-zone workload, fares particularly poorly here. However, the biggest difference between the two workloads is that at a high Zipf factor of 1.0, WAF suddenly drops for all SSDs except for Micron (m). We hypothesize that this occurs because most SSDs have a DRAM write buffer capable of capturing frequently written pages. With a Zipf factor of 1.0, 52% of accesses target only 0.05% of the pages, which can be cached by the SSD's small write buffer, drastically reducing WAF. We validated our hypothesis by implementing an LRU-like write buffer in the simulator, setting its size to 0.02% of the SSD's total capacity. The resulting curves in Figure 4b follow the pattern observed in real SSDs. Overall, as with the two-zone workload, the comparison with the simulator indicates that smarter algorithms could significantly reduce WAF. However, under very high skew, write buffering can offer similar benefits for SSDs that invest in and implement such buffers.

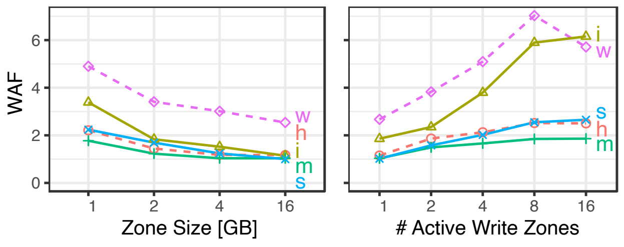
4.4 Read-Only Area Workload

Another important workload pattern in database systems involves data that is only written once and then remains unchanged, effectively making it read-only. To capture this behavior, we design a simplified workload model that simulates this scenario. The experiment begins with a full sequential write to the SSD, ensuring an initialized state. After this, we randomly select a subset of the logical block address (LBA) range (scrambled and aligned to page size, i.e., 4 KB) that will be updated uniformly, while the remaining data is never modified. Figure 5 illustrates the impact of increasing read-only area on SSD behavior, starting from a uniform random write pattern (as in previous experiments). We only simulate writes; no actual read operations are performed in this workload. Ideally, GC should benefit from the fact that read-only data can be grouped into the same erase blocks, allowing the SSD to allocate more OP space to manage the data that is updated. However, the results show that only the Intel (i) and WD (w) SSDs adapt, benefiting from read-only areas (~ 40%). The other SSDs only start showing improvements when large portions of the data (~ 80%) are read-only, suggesting that their GC algorithms are not tuned to effectively exploit this.

4.5 Sequential Workloads

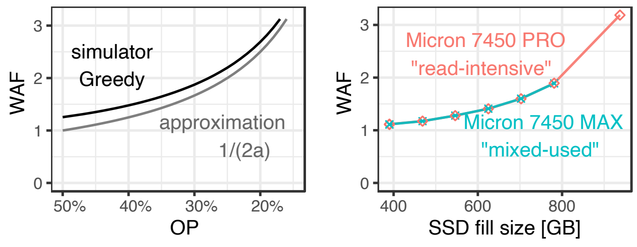
WAF = 1. The observations so far have been sobering — few SSDs can effectively leverage high skew to reduce Write Amplification Factor (WAF). This raises the question: is there any other write pattern that consistently results in predictably low WAF? Conventional wisdom suggests that large sequential writes are beneficial not only for HDDs but also for flash-based SSDs. Sequential writes are prevalent in various applications, including database systems, where they are commonly used for log writing and LSM-tree data structures. To assess the SSD's ability to exploit sequential writes, we designed an experiment where the Logical Block Addressing (LBA) range is divided into large zones, with writes occurring sequentially within each zone. In the first experiment, a random zone is selected, trimmed (BLKDISCARD), and then written sequentially. This process is repeated by selecting zones randomly, with only one zone written at a time (i.e., a single write front) As shown in Figure 6a, this approach results in a lower WAF compared to our baseline scenario of uniform random writes. However, achieving significant WAF reduction requires very large zones (>1 GB).

ZNS-like workload. This pattern essentially emulates a Zoned Namespace (ZNS) write pattern on top of a standard SSD. Considering that this is exactly the pattern that ZNS enforces, it is clear that this pattern should result in a WAF of 1 regardless of zone size, as long as the zones are large enough. The results are promising, showing WAF values below 2 for all SSDs for which we have access



(a) One active write front. **(b) Multiple active 8GB zones.**

Figure 6: ZNS-like write workload (write size: 512 KB, io-depth: 1, threads: 1, 4×capacity).



(a) Simulator and approximation. **(b) Comparison of Micron SSDs.**

Figure 7: Impact of over provisioning (OP) / SSD fill level on write amplification.

to OCP counters when the zone size exceeds 2 GB. However, unlike ZNS, there is no guarantee that WAF will remain consistently low. While this pattern could be relatively easy to adopt for log writing and LSM-tree merging, applications requiring in-place updates would need to implement their own garbage collection—just like when using ZNS devices.

Multiple writers. A major limitation of using a single active write front is that only one application or tenant can write at a time. This constraint poses challenges even for single-application scenarios, particularly in database systems. Buffer managers can optimize write patterns to some extent, but they still need to manage concurrent writes, such as the Write-Ahead Log (WAL) and buffer evictions. For LSM-tree-based systems, this restriction implies that only one merge operation can be performed at any given moment. To address this, we conducted a follow-up experiment with multiple concurrent write fronts, as illustrated in Figure 6b. Even with a large zone size of 8 GB, increasing the number of active write zones substantially worsens WAF. Notably, on the Intel SSD (i), WAF degrades beyond that observed with uniform random writes. For other SSDs, WAF remains slightly better but is still far from the optimal value (WAF = 1).

No universal solution. Importantly, we conduct all experiments directly on the raw block device, bypassing the file system. In real-world deployments, file systems introduce additional complexities such as metadata writes, allocation policies, and fragmentation, further increasing WAF unpredictability. Even under our controlled setup, only a few SSDs could leverage sequential writes to reduce WAF. When additional real-world factors come into play, achieving consistently low WAF becomes even more difficult. Ultimately, finding a write pattern that consistently delivers low and predictable WAF on standard SSDs remains a challenge.

4.6 Over-Provisioning

OP. All previous experiments utilized the full logical capacity of the device. However, over-provisioning (OP) is a critical factor that significantly impacts write amplification. OP refers to the extra capacity reserved on the SSD to support garbage collection. For workloads where the application can influence OP, such as multi-tenant services that limit storage usage (e.g., to 80% of the SSD capacity), this can greatly improve garbage collection efficiency. In single-node database instances, SSDs are typically not fully occupied, since spare space is needed for data growth. This can benefit GC when unused storage is correctly trimmed by the application.

Estimating OP. Interestingly, OP is never mentioned in SSD specifications, despite its significant impact on SSD performance. Similar to how implied WAF can be estimated, OP can also be inferred by running a uniform random write workload. Consider a scenario in which a SSD has an additional 10% of OP capacity, which is hidden from the host. With random writes, the user can overwrite data until the additional 10% OP is nearly filled. At this point, the GC algorithm has to pick blocks containing invalidated pages (i.e., logically overwritten) and compact those, to create new empty blocks. The simplest GC strategy is to pick random blocks. Then, one would expect each selected block on average to contain 10% invalid pages. Compacting 10 blocks would then yield 1 empty block, resulting in a WAF of 10. In reality, GC algorithms should implement a more advanced strategy than picking random blocks, like picking the ones with the highest number of invalid pages, which is exactly what Greedy does. In this scenario, the expected number of invalid pages per block increases to approximately 20%, meaning that compacting 5 blocks will produce 1 empty block. This results in a reduced WAF of 5. To estimate OP, we can use the WAF $\approx 1/(2 * a)$ formula from Desnoyers [13], with a representing OP. By rearranging the formula and substituting a with OP we get $OP \approx 1/(2 * \text{WAF})$, e.g., a WAF of 5 results in an OP of 10%. Lange et. al. [40] proved that for uniform random writes there is no strategy better than Greedy. Our estimated OP is therefore a lower bound, the real OP could be higher if the implemented GC algorithm performs worse than Greedy in uniform random writes. Figure 7a shows the difference between the formula and the simulated Greedy algorithm. With low values of OP the difference is small, showing it can be used as a simple approximation between WAF and OP.

OP on write-intensive SSD. SSD vendors often offer two versions of SSDs with similar hardware specifications, where the lower-capacity model is typically marketed as “write-optimized” or “mixed-use”. One might expect that such write-optimized SSDs would demonstrate improved WAF characteristics due to specialized internal designs. To investigate this, we compared two Micron SSD models: the Micron 7450 PRO, designed for “read-intensive” workloads with a capacity of 960 GB, and the Micron 7450 MAX, intended for “mixed-use” workloads with a capacity of 800 GB. Both SSDs were tested under identical workloads and dataset sizes, as shown in Figure 7b. The WAF results for both models were identical and closely matched the results from the simulator. This suggests that these Micron SSDs, despite being marketed for different workloads, are essentially identical in performance, with the only difference being a larger OP on the "mixed-use" model. For these SSD models, there appear to be no other hardware or algorithmic improvements. As a result, users can achieve similar performance by manually reserving free space on the "read-intensive" SSD, offering a practical alternative to purchasing the "mixed-use" model.

4.7 Take-Aways

Sequential writes aren't enough. The common wisdom is that sequential writes result in lower WAF. While our experiments using OCP counters confirm this, it is only true for very large sequential writes when run in isolation. Once multiple write streams are introduced, WAF worsens significantly—even when all writes are sequential.

Implications for LSM-trees. This is problematic for database systems, where write patterns are rarely purely sequential. For instance, LSM-tree-based databases seem optimal for this, as they continuously perform merging operations that produce large sequential writes. However, multiple merge operations may occur simultaneously, inevitably interleaving writes with each other and with write-ahead log (WAL) writes, deteriorating the potential benefit. In practice, the expectation that sequential writes alone can minimize WAF is overly simplistic.

Skew can worsen WAF. A surprising finding from this section is that skewed write distributions can worsen WAF rather than improve it. This is the case with very simple, static access patterns (e.g., Zipf, Two-Zone) and even when substantial fractions of data is read-only. This suggests that the garbage collection algorithms implemented in most commercial SSDs are essentially greedy variants that do not efficiently leverage write skew. The exception is the WD (w) SSD, whose behavior suggests that a more intelligent algorithm is used.

5 LATENCY

5.1 Background

Users notice latency. Latency is critical for applications running on SSDs because it directly influences throughput and it is also what users can observe. In particular, cloud and web service providers may prioritize tail latency over maximum bandwidth, as they must ensure a certain degree of response time for their services. Latency is essentially a result of interference within the SSD [41] as well as the duration of flash operations, particularly program and erase [66]. Consider a scenario where a database application is running on an SSD: In such systems, the I/O workload generally involves background writes, latency-critical reads, and writes from the write-ahead log (WAL). Background writes, such as evictions from the buffer pool, are typically performed asynchronously and are not latency-critical. In contrast, latency-critical reads are often involved in index lookups, which exhibit data dependencies—meaning each lookup depends on the result of the previous one. As a result, transactions involving multiple index lookups or I/O operations are particularly susceptible to tail latency. Finally, writes from the WAL are also latency-critical, as data is only considered committed once the WAL entry is written to non-volatile storage. Therefore, it is essential for SSDs to minimize tail latency.

Controlling latency. Despite the importance of latency, most SSD specifications only include idle latency metrics for reads and writes, and some omit latency metrics entirely, as shown in Table 1. Two key factors contribute to I/O latency: the duration of operations (especially flash program/write and flash erase times) and queuing effects, which occur due to limited parallelism within the SSD. As with any queuing system, latency is expected to increase when the device approaches its saturation point. On the other hand, latency should remain stable and predictable under moderate workloads.

Write buffering. For synchronous writes, it is crucial to mitigate the impact of long flash program times. To achieve durability without these delays, data is initially written to a volatile on-SSD write cache in DRAM. However, when a flush operation occurs, the data must be committed to non-volatile storage, which still involves high program latency. Enterprise SSDs address this issue by incorporating capacitors that provide sufficient power during power loss to flush the write cache to flash memory. This design ensures data durability without requiring the system to wait for the longer program times associated with flash writes. As a result, write latency is typically lower than read latency by an order of magnitude.

Suspension. Flash program durations are typically in the millisecond range, while erase operations can take multiple milliseconds. Erase operations occur asynchronously during GC, but their effects become apparent when other operations are queued behind them. This can lead to significant read latency when a read operation must wait for a program or erase operation to complete. In such cases, read latency can increase from an average of less than 100 µs to several milliseconds [66]. Flash controllers can mitigate this issue by suspending write and erase operations to allow low-latency read operations to proceed [37, 62]. This approach permits the controller to temporarily pause a program or erase operation, execute the read, and subsequently resume the suspended operation—significantly reducing read tail latency.

5.2 Latency Under Load

Writes under load workload. To investigate the write latency of our SSDs under load, we measured latency while running a write workload with varying write bandwidth. The results for writes with average and tail latency (at the 99th and 99.999th percentiles) are shown in the first two rows of Figure 8. The x-axis represents the relative write speed compared to the maximum write bandwidth specified for each SSD model. The SSDs are ordered in the plot from lowest/predictable to highest/unpredictable latency. The second row of the figure, highlighted with a gray background, provides a zoomed-in view of the gray-shaded section from the first row of plots. First and foremost, it is evident that significant differences exist between SSD models that are not reflected in their specifications. For instance, although the random write performance of SK Hynix (h) and Samsung (s) appear similar in their specifications (see Table 1), the (h) SSD maintains lower average and P99 write latencies (<25 µs) for higher write bandwidths compared to the other SSDs relative to the SSD's specified random write throughput. The (s) SSD exhibits comparable performance and trends to the (h) SSD as the write speed increases. In contrast, the WD (w) and Intel (i) SSDs

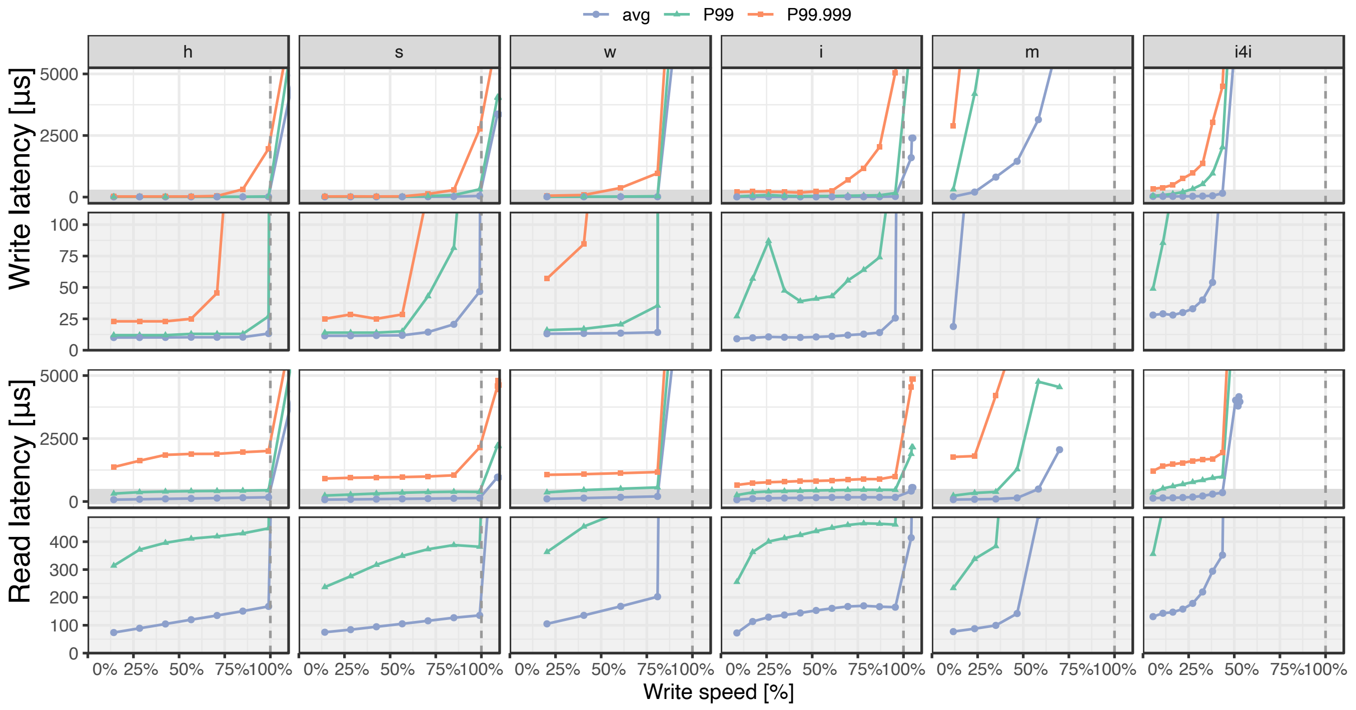


Figure 8: Write and read latency vs. normalized write bandwidth based on SSD specification. (100% fill, 4 KiB uniform writes)

are less effective at leveraging their write cache, as indicated by their consistently high tail latency even with low write throughput. The Micron (m) SSD performs poorly from the outset, likely due to either a lack of a write buffer or an inadequately sized one.

Reads under load. Figure 8 also shows the same experiment but for read latency under write load in the two lower rows. As anticipated, read latency is generally higher than write latency, and all SSDs are relatively similar, as they use the same NAND technology. The average read latency is approximately 100 µs, while the 99th percentile latency ranges between 200 µs to 500 µs. Notably, the increase in latency as write speed rises mirrors the trend observed between different SSD models in write latency. Additionally, the high tail latency (99.999th percentile) is significantly elevated for the SK Hynix (h) and Micron (m) SSDs, suggesting that these models might not support erase/write suspension.

Take-aways. This section shows that latency under load cannot be inferred from the specifications, and that it differs significantly across SSD models.

6 ACTIONABLE SSD METRICS

6.1 SSD-iq Results

SSD-iq. We now present the results of the SSD-iq benchmark for 9 SSDs from all major vendors, including state-of-the-art PCIe 5.0 drives and a cloud-based AWS instance. Table 2 shows both the existing throughput and latency metrics from the datasheet (denoted as *Spec* with a white background) and the numbers that we measured (gray background).

New performance metrics. Table 2 also shows new metrics: (1) Throughput under skewed access patterns, measured using both the Two-Zone (*TwoZ* 80/20) pattern and Zipf (0.8) workloads, *providing insights into how well the SSD manages non-uniform data access.* (2) Latency under load (25% of the SSD specified random write bandwidth), with a dedicated probe thread (io-depth=1) for read and write latency (average and 99.9th percentile).

Internal metrics. Table 3 additionally shows the following characteristics: (3) Write Amplification: Measured directly via the OCP interface or estimated based on throughput reduction under uniform and skewed workloads as described in Section 4.1. (4) Over-Provisioning (OP): Inferred from WA, OP can be estimated using the method described in Section 4.6, revealing one of the most important SSD characteristics.

6.2 SSD-iq Interpretation

Specs replication. For both sequential reads and writes, our measurements match the numbers from the *Spec*. The random read throughput we observe in our measurements with a uniform workload is largely similar to performance advertised in the datasheets. If anything our measurements are slightly better than the advertised performance. The random write throughput we observe for a uniform workload is also higher than the *Spec* by a few percent. One hypothesis is that SSD manufacturers provide slightly conservative numbers to account for variance in manufacturing quality.

Throughput under skewed access patterns. Throughput obtained with a skewed write workload is lower than with a uniform workload for most SSDs, except for the Intel (i) and WD (w) SSDs, as shown in Section 4. Out of nine state-of-the-art data center SSDs, only two appear to implement a more sophisticated GC algorithm

Table 2: SSD Performance Comparison

		Samsung	SK Hynix	Intel	WD Ultra.	Micron	Micron	Samsung	Kioxia	AWS EC2
		PM9A3	PE8110	D7-P5520	DC SN640	7450 PRO	7450 MAX	PM1733	CM7-R	i4i.4xlarge
		s	h	i	w	m	-	-	-	-
Capacity *[GB]*		960	960	1920	960	960	800	3840	3840	3750
Seq. Read	*Spec*	6500	6500	5300	3330	6800	6800	7000	14000	2800
[MB/s]		6696	6564	5236	3083	6700	6732	6973	14312	2844
Seq. Write	*Spec*	1500	1700	1900	1190	1400	1400	3800	6750	–
[MB/s]		1441	1731	1957	1189	1393	1391	3538	6746	2258
Rnd. Read	*Spec*	580	900	700	434	530	530	1500	2700	400
[k IOPS]		582	1016	731	460	549	549	1483	2775	393
Rnd. Write	*Spec*	70	70	114	49	85	145	135	310	220
[k IOPS]	*Unif*	71	84	118	47	94	160	146	330	146
	TwoZ	67	79	129	50	85	141	134	301	134
	Zipf	67	79	117	50	84	143	133	296	133
Read Lat.	*Spec*	–	75	75	78	80	80	–	–	–
under load	*Avg*	62	58	61	75	66	67	59	65	126
[us]	*P99.9*	487	521	734	471	532	485	571	1153	1179
Write Lat.	*Spec*	–	–	15	–	15	15	–	–	–
under load	*Avg*	13	11	11	17	10	10	16	8	27
[us]	*P99.9*	17	15	178	36	10790	9504	131	272	715

Background colors: white: data based on the spec; gray: measured value. Other colors have different interpretations depending on the metric: (1) For "Rnd. Write Unif" it means conformity to the spec. (2) For "Rnd. Write TwoZ/Zipf" it is a comparison to the uniform workload. (3) For latency it is absolute measure: For "Read Avg.": < 100 µs green else red; for "P99.9": < 500 µs green, < 1000 µs yellow, else red. For "Write Avg.": < 20 µs green, else yellow; for "P99.9": < 20 µs green, < 100 µs yellow, else red.

Table 3: Additional SSD Characteristics

		Samsung	SK Hynix	Intel	WD Ultra.	Micron	Micron	Samsung	Kioxia	AWS EC2
		PM9A3	PE8110	D7-P5520	DC SN640	7450 PRO	7450 MAX	PM1733	CM7-R	i4i.4xlarge
OCP Support		yes	no	yes	no	yes	yes	no	yes	no
WA	*Unif*	4.2	5.3	3.5	6.5	3.2	1.9	6.2	4.4	4.0
	TwoZ	4.5	5.6	3.4	6.1	3.4	2.1	6.7	4.8	4.3
	Zipf	4.5	5.6	3.6	6.1	3.5	2.1	6.8	4.9	4.3
Estimated OP		11.9%	9.5%	14.3%	7.7%	15.7%	26.5%	8.1%	11.3%	12.6%

Background coloring interpretation: gray: measured or estimated value. For "WA TwoZ/Zipf" it is a comparison to the uniform workload (green: better, red: worse).

capable of exploiting skewed data access. Overall though, *the observed throughput under skewed access patterns is within a few percent of the Spec for all SSDs*, except for the AWS instance (i4i), which is significantly below the advertised random write bandwidth.

Latency under load. The average read latency under load that we observe is lower than the latency numbers reported in the datasheets. However, the *P99.9* latency is an order of magnitude higher. For writes, the *P99.9* latency is substantially higher than average latency on most SSDs. Only the SSD from Samsung (PM9A3), SK Hynix, and WD perform well. The Micron SSDs exhibit a *P99.9* latency that is two orders of magnitude higher than average. The SSD in the AWS instance has particularly problematic properties in terms of latency (as is the case for throughput as we noted above).

Write Amplification. WAF varies by a factor of 2.5 between the SSDs with highest WAF (WD Ultra and Samsung PM1733) and the SSD with the lowest WAF (Micron 7450 MAX).

Over-Provisioning. As explained in Section 4.6, the estimated Over-Provisioning (OP) differ across SSD models. Most interestingly, the "mixed-used" SSD from Micron (26.5%) has 10% more OP than the "read-intensive" model (15.7%).

6.3 Choosing an SSD: Implications for DBMS

Let us compare the Samsung PM9A3 (*s*) and Micron 7450 PRO (*m*) SSDs in more detail. These two models have similar specifications, the same capacity (960 GB), and, at the time of writing, similar prices. In terms of sequential throughput, *m* is slightly better for reads,

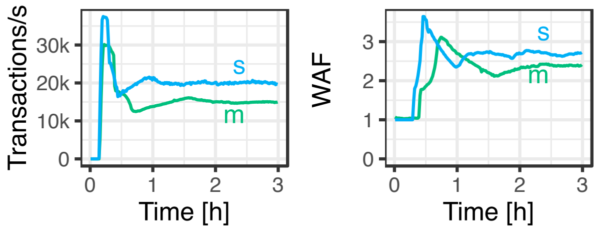


Figure 9: Performance and WAF of steady TPC-C in LeanStore (64 GB buffer pool, 128 threads, 6550 warehouses ≈ 800 GB).

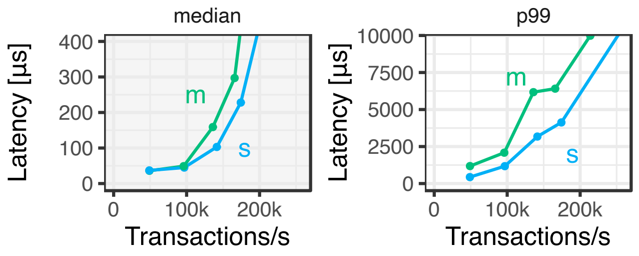


Figure 10: Latency of YCSB-A in LeanStore. (64 GB buffer pool, 96 threads, 3 billion tuples, 811 GB data, 40 GB WAL).

while s is slightly better for writes. For random I/O, m performs better on writes, whereas s is better for reads. From the specifications alone, it is impossible to make an informed decision regarding which SSD to choose for a database workload. SSD-iq shows two very significant differences between these SSDs: (1) write latency under load degrades by two orders of magnitude for m, while it remains stable for s; (2) WAF is 25% higher for s than for m (WAF is between 4.2 and 4.4 for s, while it is between 3.2 and 3.5 for m). According to SSD-iq, the choice between these two SSDs matters and it is a trade-off between performance under load (s is better) and write amplification, i.e., lifetime (m is better).

TPC-C. Figure 9 illustrates the impact of SSD behavior on database performance using TPC-C on LeanStore [1, 22, 53]. LeanStore uses a B^+-Tree based buffer manager, conceptually similar to those in PostgreSQL and MySQL. The advantage of LeanStore is that it is a storage engine optimized for multi-core CPUs and NVMe SSDs, and will therefore not be CPU-bound. Like with our previous microbenchmarks, SSD are erased and the experiment is run for multiple hours until the steady state is reached. The code for steady TPC-C and experiment can be found in [19] and for the YCSB latency experiment in [18]. In this configuration, the TPC-C workload results in about 60% reads and 40% writes. The Figure shows that the choice of SSD impacts database performance. First, the SSD is responsible for the evolution of throughput over time. Initially, throughput is high (about 35k transactions per second for the Samsung SSD (s) and 30k transactions per second for the Micron SSD (m)). Then Garbage Collection (GC) kicks in. which causes a sharp decline in database throughput. Performance recovers when WAF starts to stabilize until it reaches a steady state. Second, the choice of SSD impacts the steady state performance. LeanStore achieves a throughput of 20K transactions per second with s and 15K transactions per second with m. At the same time, we observe that the WAF of s is about 2.8, while the WAF of m is about 2.4.

YCSB. SSD-iq shows significant differences in latency under load. To confirm this, we measure end-to-end database latency using the YCSB-A workload (50% lookups, 50% updates) under varying throughput levels with LeanStore. Figure 10 shows that there is a clear difference, with the m SSD showing higher median latency under load. The p99 latency for m is especially problematic, with values consistently worse than the s SSD at low throughput. These results emphasize that the performance gap between m and s is not just observable in synthetic I/O benchmarks but has real-world implications under OLTP workloads. Overall, these experiments provides evidence to support the predictions we made with SSD-iq.

7 OUTLOOK

Insights. We proposed SSD-iq, a benchmark designed to measure SSD write amplification and latency under load with skewed and mixed access patterns. We evaluated SSD-iq on nine commercial SSDs and showed that the choice of SSD matters for database performance. SSD-iq is well-suited to inform this choice. Our findings also unveil fundamental inefficiencies in existing SSDs when handling write-intensive workloads. In particular, most of the SSDs we tested behave as if they implement a simple, greedy garbage collection algorithm, which negatively impacts write amplification.

Future Interfaces. How about upcoming SSD architectures? How will they impact database systems? What are the implications for SSD-iq? Most of the performance penalty we observe in Section 4 and Section 5 stems from interferences within SSDs with a traditional block device interface. We expect that newer NVMe interfaces, like ZNS [5] and FDP [47] will be useful to reduce WAF for classes of workloads, especially those based on sequential writes [2, 46]. ZNS and FDP allow the host some degree of control over data placement. For instance, database systems could manually manage data placement and GC for different tables and objects or use pre-existing statistics about page access. However, just moving the task of data placement and GC from the device to the host is not a panacea. If the garbage collector is not capable of exploiting placement information, it does not matter whether the host or the SSD firmware is responsible for it. Quantifying the impact of ZNS and FDP with SSD-iq remains a topic for future work.

Sustainability. SSD sustainability has become a key issue for SSD designers. Recent work has shown that the number of SSDs used in a system is the main factor with respect to sustainability (far more important than energy consumption at rest or in operation) [51]. As a result, there is increased focus on SSD utilization and SSD lifetime. The latter is directly related to write amplification. A question for future work is whether the SSD-iq benchmark uncovers all relevant aspects of write amplification, or whether new developments of the benchmark are needed to uncover SSD sustainability issues.

ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 447457559.

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] Adnan Alhomssi and Viktor Leis. 2023. Scalable and Robust Snapshot Isolation for High-Performance Storage Engines. *Proc. VLDB Endow.* 16, 6 (2023), 1426–1438.

[2] Michael Allison, Arun George, Javier Gonzalez, Dan Helmick, Roshan Nair, and Vivek Shah. 2025. Towards Efficient Flash Caches with Emerging NVMe Flexible Data Placement SSDs. In *Eurosys*.

[3] Apache. 2016. Apache Commons Random Number Generator. https://github.com/apache/commons-rng/blob/master/commons-rng-sampling/src/main/java/org/apache/commons/rng/sampling/distribution/RejectionInversionZipfSampler.java.

[4] Jens Axboe. 2025. fio: Flexible I/O Tester. https://github.com/axboe/fio.

[5] Matias Bjørling, Abutalib Aghayev, Hans Holmberg, Aravind Ramesh, Damien Le Moal, Gregory R. Ganger, and George Amvrosiadis. 2021. ZNS: Avoiding the Block Interface Tax for Flash-based SSDs. In *USENIX ATC*. 689–703.

[6] Matias Bjørling, Javier González, and Philippe Bonnet. 2017. LightNVM: The Linux Open-Channel SSD Subsystem. In *FAST*. 359–374.

[7] Luc Bouganim, Björn Þór Jónsson, and Philippe Bonnet. 2009. uFLIP: Understanding Flash IO Patterns. In *CIDR*.

[8] Li-Pin Chang. 2007. On efficient wear leveling for large-scale flash-memory storage systems. In *SAC*. 1126–1130.

[9] Yuan-Hao Chang, Jen-Wei Hsieh, and Tei-Wei Kuo. 2010. Improving Flash Wear-Leveling by Proactively Moving Static Data. *IEEE Trans. Computers* 59, 1 (2010), 53–65.

[10] Niv Dayan and Philippe Bonnet. 2015. Garbage Collection Techniques for Flash-Resident Page-Mapping FTLs. *CoRR* abs/1504.01666 (2015).

[11] Niv Dayan, Luc Bouganim, and Philippe Bonnet. 2015. Modelling and Managing SSD Write-amplification. *CoRR* abs/1504.00229 (2015).

[12] Gerhard Derflinger, Wolfgang Hörmann, and Josef Leydold. 2010. Random variate generation by numerical inversion when only the density is known. *ACM Trans. Model. Comput. Simul.* 20, 4 (2010), 18:1–18:25.

[13] Peter Desnoyers. 2014. Analytic Models of SSD Write Performance. *ACM Trans. Storage* 10, 2 (2014), 8:1–8:25.

[14] Djellel Eddine Difallah, Andrew Pavlo, Carlo Curino, and Philippe Cudré-Mauroux. 2013. OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases. *PVLDB* 7, 4 (2013), 277–288. http://www.vldb.org/pvldb/vol7/p277-difallah.pdf

[15] Thanh Do, Mingzhe Hao, Tanakorn Leesatapornwongsa, Tiratat Patana-anake, and Haryadi S. Gunawi. 2013. Limplock: understanding the impact of limpware on scale-out cloud systems. In *SoCC*. 14:1–14:14.

[16] Krijn Doekemeijer, Nick Tehrany, Balakrishnan Chandrasekaran, Matias Bjørling, and Animesh Trivedi. 2023. Performance characterization of NVMe flash devices with zoned namespaces (ZNS). In *2023 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 118–131. https://doi.org/10.1109/cluster52292.2023.00018

[17] Arun George. 2025. Introduction to Flexible Data Placement: A New Era of Optimized Data Management. https://download.semiconductor.samsung.com/resources/white-paper/FDP_Whitepaper_102423_Final.pdf.

[18] Gabriel Haas. 2025. Commit 2a96057: ssd latency experiment. https://github.com/leanstore/leanstore/commit/2a96057

[19] Gabriel Haas. 2025. Commit e0c5a7d: SSD WAF experiment with steady TPC-C. https://github.com/leanstore/leanstore/commit/e0c5a7d

[20] Gabriel Haas, Adnan Alhomssi, and Viktor Leis. 2025. Managing Very Large Datasets on Directly Attached NVMe Arrays. *Scalable Data Management for Future Hardware* (2025), 223.

[21] Gabriel Haas, Michael Haubenschild, and Viktor Leis. 2020. Exploiting directly-attached NVMe arrays in DBMS. In *CIDR*. https://www.cidrdb.org/cidr2020/papers/p16-haas-cidr20.pdf

[22] Gabriel Haas and Viktor Leis. 2023. What Modern NVMe Storage Can Do, And How To Exploit It: High-Performance I/O for High-Performance Storage Engines. *Proc. VLDB Endow.* 16, 9 (2023), 2090–2102.

[23] Shujie Han, P Lee, Fan Xu, Yi Liu, Cheng He, and Jiongzhou Liu. 2021. An in-depth study of correlated failures in production SSD-based data centers. *File Storage Technol* (2021), 417–429. https://www.usenix.org/conference/fast21/presentation/han

[24] Mingzhe Hao, Gokul Soundararajan, Deepak R. Kenchammana-Hosekote, Andrew A. Chien, and Haryadi S. Gunawi. 2016. The Tail at Store: A Revelation from Millions of Hours of Disk and SSD Deployments. In *FAST*. 263–276.

[25] Jun He, Sudarsun Kannan, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. 2017. The Unwritten Contract of Solid State Drives. In *EuroSys*. 127–144.

[26] Benny Van Houdt. 2023. On the Cost of Near-Perfect Wear Leveling in Flash-Based SSDs. *ACM Trans. Model. Perform. Evaluation Comput. Syst.* 8, 1-2 (2023), 1–22.

[27] Xiao-Yu Hu, Evangelos Eleftheriou, Robert Haas, Ilias Iliadis, and Roman A. Pletka. 2009. Write amplification analysis in flash-based solid state drives. In *SYSTOR*. 10.

[28] JEDEC Solid State Technology Association. 2010. Solid-State Drive (SSD) Requirements and Endurance Test Method. JESD218A.

[29] JEDEC SOLID STATE TECHNOLOGY ASSOCIATION. 2022. Solid-State Drive (SSD) Endurance Workloads. JESD219.A01.

[30] Song Jiang, Lei Zhang, XinHao Yuan, Hao Hu, and Yu Chen. 2011. S-FTL: An efficient address translation for flash memory by exploiting spatial locality. In *MSST*. 1–12.

[31] Dawoon Jung, Yoon-Hee Chae, Heeseung Jo, Jinsoo Kim, and Joonwon Lee. 2007. A group-based wear-leveling algorithm for large-capacity flash memory storage systems. In *CASES*. 160–164.

[32] Aarati Kakaraparthy, Jignesh M. Patel, Kwanghyun Park, and Brian Kroth. 2019. Optimizing Databases by Learning Hidden Parameters of Solid State Drives. *PVLDB* 13, 4 (2019), 519–532.

[33] Jeong-Uk Kang, Jeeseok Hyun, Hyunjoo Maeng, and Sangyeun Cho. 2014. The Multi-streamed Solid-State Drive. In *HotStorage*.

[34] Minji Kang, Soyee Choi, Gihwan Oh, and Sang Won Lee. 2020. 2R: Efficiently Isolating Cold Pages in Flash Storages. *PVLDB* 13, 11 (2020), 2004–2017.

[35] Wonkyung Kang, Dongkun Shin, and Sungjoo Yoo. 2017. Reinforcement Learning-Assisted Garbage Collection to Mitigate Long-Tail Latency in SSD. *ACM Trans. Embed. Comput. Syst.* 16, 5s, Article 134 (sep 2017), 20 pages. https://doi.org/10.1145/3126537

[36] Jaeho Kim, Kwanghyun Lim, Youngdon Jung, Sungjin Lee, Changwoo Min, and S Noh. 2019. Alleviating garbage collection interference through spatial separation in all flash arrays. *USENIX Annu Tech Conf* (2019), 799–812. https://www.usenix.org/conference/atc19/presentation/kim-jaeho

[37] Shine Kim, Jonghyun Bae, Hakbeom Jang, Wenjing Jin, Jeonghun Gong, SeungYeon Lee, Tae Jun Ham, and Jae W. Lee. 2019. Practical Erase Suspension for Modern Low-latency SSDs. In *USENIX ATC*. USENIX Association, 813–820.

[38] Ricardo Koller and Raju Rangaswami. 2008. FIU IODedup Traces (SNIA IOTTA Trace Set 391). In *SNIA IOTTA Trace Repository*, Geoff Kuenning (Ed.). Storage Networking Industry Association. http://iotta.snia.org/traces/block-io/390?only=391

[39] Ricardo Koller and Raju Rangaswami. 2010. I/O Deduplication: Utilizing content similarity to improve I/O performance. *ACM Trans. Storage* 6, 3 (2010), 13:1–13:26.

[40] Tomer Lange, Joseph (Seffi) Naor, and Gala Yadgar. 2023. Offline and Online Algorithms for SSD Management. *Commun. ACM* 66, 7 (2023), 129–137.

[41] Alberto Lerner and Philippe Bonnet. 2025. *Principles of Database and Solid-State Drive Co-Design*. Springer. https://link.springer.com/book/9783031578762

[42] Alberto Lerner, Jaewook Kwak, Sangjin Lee, Kibin Park, Yong Ho Song, and Philippe Cudré-Mauroux. 2020. It takes two: Instrumenting the interaction between in-memory databases and solid-state drives. *CIDR 2020, 10th Conference on Innovative Data Systems Research, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings* (2020), 1–8. https://folia.unifr.ch/rerodoc/330495

[43] Jinhong Li, Qiuping Wang, Patrick P. C. Lee, and Chao Shi. 2020. An In-Depth Analysis of Cloud Block Storage Workloads in Large-Scale Production. In *IISWC*. 37–47.

[44] Ruiming Lu, Erci Xu, Yiming Zhang, Zhaosheng Zhu, Mengtian Wang, Zongpeng Zhu, Guangtao Xue, Minglu Li, and Jiesheng Wu. 2022. NVMe SSD Failures in the Field: the Fail-Stop and the Fail-Slow. In *USENIX ATC*. 1005–1020.

[45] Stathis Maneas, Kaveh Mahdaviani, Tim Emami, and Bianca Schroeder. 2022. Operational Characteristics of SSDs in Enterprise Storage Systems: A Large-Scale Field Study. In *FAST*. 165–180.

[46] Sara McAllister, Yucong Wang, Benjamin Berg, Daniel S. Berger, George Amvrosiadis, Nathan Beckmann, and Gregory R. Ganger. 2024. FairyWREN: A Sustainable Cache for Emerging Write-Read-Erase Flash Interfaces. In *OSDI*. 745–764.

[47] John Rudelic Michael Allison. 2023. What is the NVM Express® Flexible Data Placement (FDP)? https://www.snia.org/educational-library/what-nvm-express%C2%AE-flexible-data-placement-fdp-2023.

[48] MSR. 2018. MSR Cambridge Traces (SNIA IOTTA Trace Set 388). In *SNIA IOTTA Trace Repository*, Geoff Kuenning (Ed.). Storage Networking Industry Association. http://iotta.snia.org/traces/block-io?only=388

[49] Muthukumar Murugan and David Hung-Chang Du. 2011. Rejuvenator: A static wear leveling algorithm for NAND flash memory with minimized overhead. In *MSST*. 1–12.

[50] Roshan Nair and Arun George. 2024. The Promise of NVMe Flexible Data Placement in Data Center Sustainability. In *SNIA Storage Developer Conference (SDC24)*.

[51] Roshan Nair and Arun George. 2024. The Promise of NVMe Flexible Data Placement in Data Center Sustainability. URL: https://www.sniadeveloper.org/events/agenda/session/698.

[52] Dushyanth Narayanan, Austin Donnelly, and Antony I. T. Rowstron. 2008. Write off-loading: Practical power management for enterprise storage. *ACM Trans. Storage* 4, 3 (2008), 10:1–10:23.

[53] Lam-Duy Nguyen, Adnan Alhomssi, Tobias Ziegler, and Viktor Leis. 2025. Moving on From Group Commit: Autonomous Commit Enables High Throughput and Low Latency on NVMe SSDs. *Proceedings of the ACM on Management of*

Data 3, 3 (2025), 1–24.

[54] NVM Express, Inc 2023. *NVM Command Set Specification*. NVM Express, Inc. Revision 1.0d.

[55] Phitchaya Mangpo Phothilimthana, Saurabh Kadekodi, Soroush Ghodrati, Selene Moon, and Martin Maas. 2024. Thesios: Synthesizing Accurate Counterfactual I/O Traces from I/O Samples. In *ASPLOS*. 1016–1032.

[56] Alma Riska and Erik Riedel. 2006. Disk Drive Level Workload Characterization. In *USENIX ATC*. 97–102.

[57] Ross Stenfort, Ta-Yu Wu, and Lee Prewitt. 2020. *NVMe Cloud SSD Specification*. Open Compute Project (OCP). Version 1.0a (06262020).

[58] Radu Stoica and Anastasia Ailamaki. 2013. Improving Flash Write Performance by Using Update Frequency. *PVLDB* 6, 9 (2013), 733–744.

[59] Vasily Tarasov, E Zadok, and S Shepler. 2016. Filebench: A Flexible Framework for File System Benchmarking. *login - The Usenix Magazine* 41 (2016). https://www.semanticscholar.org/paper/Filebench%3A-A-Flexible-Framework-for-File-System-Tarasov-Zadok/b5c260b4cc4110abd7f05302b3e6109011a772bb

[60] TPC. 2025. TPC-C Homepage. https://www.tpc.org/tpcc/. Accessed: 2025-4-29.

[61] Kristian Vättö. 2025. The OCZ Vector 180 SSD Review. https://www.anandtech.com/show/9009/ocz-vector-180-240gb-480gb-960gb-ssd-review.

[62] Guanying Wu and Xubin He. 2012. Reducing SSD read latency via NAND flash program and erase suspension. In *FAST*. USENIX Association, 10.

[63] Gala Yadgar and Moshe Gabel. 2016. Avoiding the Streetlight Effect: I/O Workload Analysis with SSDs in Mind. In *HotStorage*.

[64] Gala Yadgar, Moshe Gabel, Shehbaz Jaffer, and Bianca Schroeder. 2018. YCSB RocksDB SSD Traces (SNIA IOTTA Trace Set 28568). In *SNIA IOTTA Trace Repository*, Geoff Kuenning (Ed.). Storage Networking Industry Association. http://iotta.snia.org/traces/block-io?only=28568

[65] Gala Yadgar, Moshe Gabel, Shehbaz Jaffer, and Bianca Schroeder. 2021. SSD-based Workload Characteristics and Their Performance Implications. *ACM Trans. Storage* 17, 1 (2021), 8:1–8:26.

[66] Shiqin Yan, Huaicheng Li, Mingzhe Hao, Michael Hao Tong, Swaminathan Sundararaman, Andrew A. Chien, and Haryadi S. Gunawi. 2017. Tiny-Tail Flash: Near-Perfect Elimination of Garbage Collection Tail Latencies in NAND SSDs. *ACM Trans. Storage* 13, 3 (2017), 22:1–22:26.

[67] Pan Yang, Ni Xue, Yuqi Zhang, Yangxu Zhou, Li Sun, Wenwen Chen, Zhonggang Chen, Wei Xia, Junke Li, and Kihyoun Kwon. 2019. Reducing Garbage Collection Overhead in {SSD} Based on Workload Prediction. In *11th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 19)*. https://www.usenix.org/conference/hotstorage19/presentation/yang

[68] You Zhou, Fei Wu, Ping Huang, Xubin He, Changsheng Xie, and Jian Zhou. 2015. An efficient page-level FTL to optimize address translation in flash memory. In *EuroSys*. 12:1–12:16.