Challenges and future directions for Energy, Latency, and Lifetime Improvements in NVMs

Saeed Kargar (skargar@ucsc.edu)
UC Santa Cruz
Faisal Nawab
UC Irvine

Systematic Review

Keywords: Non-Volatile Memory, write endurance, energy consumption, bit flipping

Posted Date: April 1st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1507900/v1

Challenges and future directions for Energy, Latency, and Lifetime Improvements in NVMs

Saeed Kargar[1*] and Faisal Nawab[2]

[1]CSE depatment, UC Santa Cruz, Santa Cruz, California, USA.
[2]CSE depatment, UC Irvine, Irvine, California, USA.

*Corresponding author(s). E-mail(s): skargar@ucsc.edu;
Contributing authors: nawabf@uci.edu;

Abstract

Recently, Non-Volatile Memory (NVM) technology has revolutionized the landscape of memory systems. With many advantages, such as non volatility and near zero standby power consumption, these byte-addressable memory technologies are taking the place of DRAMs. Nonetheless, they also present some limitations, such as limited write endurance, which hinders their widespread use in today's systems. Furthermore, adjusting current data management systems to embrace these new memory technologies and all their potential is proving to be a nontrivial task. Because of this, a substantial amount of research has been done, from both the database community and the storage systems community, that tries to improve various aspects of NVMs to integrate these technologies into the memory hierarchy. In this tutorial we survey state-of-the-art work on deploying NVMs in database and storage systems communities and the ways their limitations are being handled within these communities. In particular, we focus on the challenges that are related to high energy consumption, low write endurance and extending the lifetime of NVM devices.

Keywords: Non-Volatile Memory, write endurance, energy consumption, bit flipping

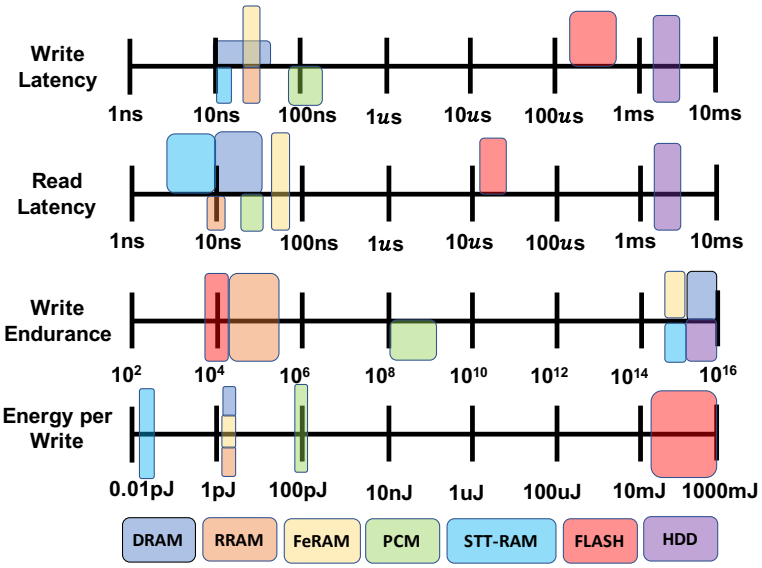

Fig. 1 Comparison of device properties of memory technologies [4]

1 Introduction

Nowadays, we are witnessing the emergence of new non-volatile memory (NVM) technologies that are remarkably changing the landscape of memory systems. They are persistent, have high density, byte addressable, and require near-zero standby power [1], which makes them of great interest in the database and storage systems communities. Furthermore, NVMs provide up to 10x higher bandwidth and 100x lower access latency compared to SSDs [2, 3]. However, because they also present some limitations, such as limited write endurance, which is significantly lower than DRAM write endurance, and high write energy consumption (Figure 1), adopting the current systems to use NVM while maximizing their potential is proving to be challenging. Additionally, unlike flash storage, cells are written individually in many NVM technologies such as PCM. This means that some cells may receive a much higher number of writes than others during a given period, and as a result wear out sooner. So, any system design needs to take these limitations into consideration before deciding to utilize NVMs.

The scientific community has conducted an extensive amount of research on integrating these new technologies in current systems. Furthermore, many companies are announcing the mass production of NVMs, which implies that these emerging technologies are carving out their own place in the memory hierarchy. So, in this tutorial we survey recent work in both storage and database communities, where a substantial amount of work has been done in the adoption of NVMs. In particular, we focus on the challenges related to the low write endurance of NVM. This area did not gain enough attention in the data management community and this tutorial will provide information on how to

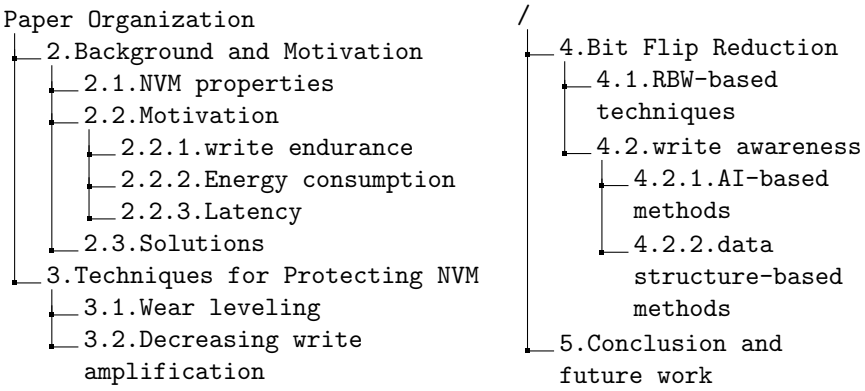

Fig. 2 Organization of the paper.

integrate recent advances from the NVM storage community into existing and future data management systems. Additionally, we discuss how the challenges of low write endurance are conflated with write amplification in data management systems and show that a specific treatment for low write endurance would lead to better longevity and to extending the lifetime of NVM devices.

Contributions. In this paper, we present a survey of techniques for improving NVM's lifetime and energy consumption. Fig. 2 shows an overview of this paper. In Section 2, we present a background on NVM key terms, concepts, and properties. In Section 2.2, we underscore the motivation for and challenges in improving NVM lifetime. Further, we classify the importance of works based on three main views to give an overall view of the whole field. We discuss three main methods for overcoming NVM limitations in Sections. 3 and 4. In Section. 4 we elaborate bit flip reduction technique, which is the main focus of this survey. We conclude this paper in Section. 5 with a brief mention of future research challenges.

Scope. For sake of a concise presentation, we limit the scope of this paper as follows. We discuss the storage and hardware-level techniques focused on decreasing bit flips, and not wear-leveling or write amplification, although some of the ideas/techniques proposed for them are presented in Section.3, which may also have some positive effects on decreasing the number of bit flips in NVM. We believe that this paper will be useful for memory designers and database and storage researchers who are new to this field and are interested to integrate NVMs in their work.

2 Background and Motivation

We now introduce some concepts which will be useful throughout this paper. We refer the reader to previous works for a detailed background on NVM architecture [5–7].

2.1 NVM Properties

Non-Volatile Memory (NVM) has the potential of transforming the memory architecture in data management systems due to their characteristics such as persistency, high density, byte addressability, and requiring near-zero standby power [1]. However, NVM technologies suffer from two main challenges that needs to be taken into consideration to efficiently utilize them. 1) NVM has low write endurance (the number of writes that can be applied to a segment of storage media before it becomes unreliable.) NVM write endurance is on the order of 10^8–10^9 writes, which is significantly lower than DRAM write endurance which is on the order of 10^{15} writes [4, 5]. 2) NVM write operations demand a significant amount of current and power. For flipping an individual bit in PCM, for instance, it requires around 50 pJ/b compared to writing a DRAM page, which needs only 1 pJ/b [8]. Figure. 1 shows the device properties of different memory technologies including some NVM technologies. As it is clear in this figure, the asymmetric read and write properties of NVMs are the main concerns of the research community who are working on NVMs. These limitations make researchers either redesign the conventional data structures that require fewer writes or use hardware techniques such as caching to reduce writes.

2.2 Motivation for bit flip reduction

2.2.1 Write endurance

Recently, the topic of improving NVM write endurance has become controversial among NVM research community, especially with all the commercial claims of facts from the vendors, which state that the recently released NVMs can tolerate years of intensive writes before they wear out [9].

Nevertheless, many recent studies have shown that although this might be true to some extent, it does not mean that this problem is solved altogether [8, 10–14]. For example, there is no information on the type of data sets that are used to get to these results. Using certain data sets, such as sequential, which have predictable bit patterns, can extend the lifetime of the system tens of times compared to some other data sets, such as uniform, which has unpredictable patterns.

Furthermore, unlike flash storage, cells are written individually in many NVM technologies such as PCM. This means that some cells may receive a much higher number of writes than others during a given period, and as a result wear out sooner. For example, in some extreme cases, even with the protection of state-of-the-art wear-leveling schemes, wear-out attacks such as [15] and [16] can wear out the modern NVMs as fast as 137 seconds [16]. So, improving the lifetime of NVMs is still an open topic in NVM technologies, which can be achieved through reducing the number of written cells in the system as much as possible. Not only does reducing the number of written cells increase the average lifetime of the device, but also it decrease the system's power consumption significantly.

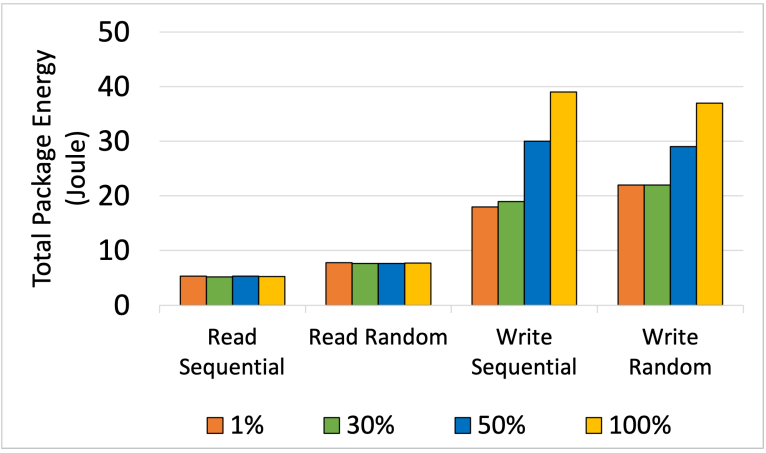

Fig. 3 Total memory energy consumption on a real Intel Optane memory device for read and write operations with different percentages of hamming distance.

2.2.2 Energy Consumption

Bit flip reduction represents the design principle of minimizing how many bits are flipped from 0 to 1 or 1 to 0 when a write is applied to a memory segment. As the number of bit flips decreases, as energy efficiency and write endurance improve.

Reducing the number of written cells also means that the energy consumption of the system drops significantly. For instance, based on an experiment conducted by [17], when the number of different bits in the write data and the overwritten content varies from 0% to 100%, the energy consumption can vary from nearly 0 to over 10000 pJ. So, targeting bit flip reduction is a worthy investment in the NVM context, and there are a large body of research, such as [8, 10–14, 17–19], targeting bit flip reduction to extend the NVM lifetime and decrease their energy consumption.

To see how this difference affects the system's energy consumption and performance, we have conducted a simple experiment on a real Optane memory device. We used the Persistent Memory Development Kit (PMDK) [20], formerly known as NVML. In this test, first, we allocate a contiguous region of N Optane blocks of 256B. During each "round" of the experiment, we first initialize all the blocks with random data, and then update the blocks with new data with content that is x% different than the data that is already in the block (hamming distance). We use PMDK's transactions to persist writes. We measure the energy consumption of the socket for each round. Figure 3 shows that by overwriting similar content, which needs less bit flipping, we can save a huge amount of energy.

2.2.3 Latency

Not only does reducing the number of bit flips save the energy consumption of the system, but also it can improve the latency of write operations. Figure. 4

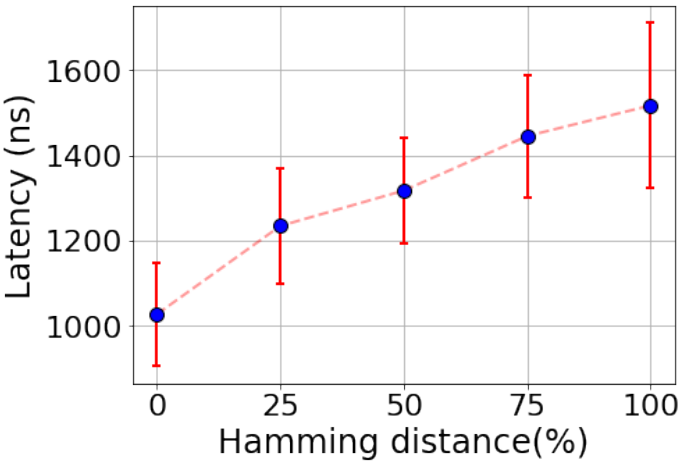

Fig. 4 The write latency in a real Intel Optane memory device of write operation for different percentages of hamming distance.

shows that write latency also improves when bit flips are reduced. The main reason behind this is when the content of the old data and the data that is being written is similar, the total number of writes can also be reduced, which results in improving write latency. This process can reduce the number of writes in two ways: (1) the first way is by writing new items in-place to replace a similar old value in terms of hamming distance. This leads to decreasing NVM word writes (i.e., the number of modified words in a cache line.) (2) In the second way, writing similar contents decreases the number of NVM line writes, respectively cache lines needed to be written per item. For example, suppose that the page size in a system is 4KB as shown in Figure. 1 [11]. In this scenario, if the items are similar to each other in terms of the hamming distance, fewer number of cache lines are needed to fulfill the request (suppose each part in Figure. 5 is a cache line). This enables PNW to decrease NVM word writes in addition to NVM line writes.

2.3 Solutions

There have been plenty of methods proposed for improving NVM write endurance and energy consumption from hardware-focused methods to software-focused ones. According to their general strategies, these methods can be categorized into three main groups:

1) the storage community developed write optimization techniques that are mostly based on a *Read-Before-Write* (RBW) pattern [21]. In RBW, a write operation w to a memory location x is always preceded with a read of x. The value to be written by w is compared with the old content of x, and only the bits that are different are written. This reduces the number of flipped bits, which increases write endurance [21]. Other approaches built on RBW to increase write endurance by masking or changing the value to be written if it

	Content				
Old item	0xABCDA23B	BCDABCDA	BCDAB01A ... 1CDA23BA	CCDAAB00	ACCDAB01
New item	0xABCDABC1	BCDABCDA	BCDAB01A ... 1CDA23BA	CCDAAB00	ABCDABC8

Fig. 5 An example of replacing a memory content with a similar content used in PNW [11]

leads to reducing bit flips [22–26]. Flip-N-Write (FNW), for example, checks whether flipping the bits of the write operation would lead to decreasing the number of bit flips [22].

2) the data management community tackled the problem of write endurance by minimize write operations via techniques such as caching [27–29] and delayed merging [30, 31], or by designing specialized data structures that require fewer writes [10]. Therefore, data structures that are designed to be deployed on NVM should be designed in a way to exploit the advantages and avoid the disadvantages of the technologies. For example, data structures for disks are block-oriented and work the best for sequential access. However, those designed for flash reduce write amplification, which is the main concern in flash technologies [8]. As we will explain later, techniques that focus on reducing write amplification can result in increasing write endurance although this is not always the case [8].

3) The proposed methods in the third group also increase the write endurance of NVMs through a new approach called *memory-aware*. The main idea behind this approach is to take advantage of having knowledge of the memory content in advance because read operations are less expensive than write operations in NVMs. Prior methods pick the memory location for a write operation arbitrarily (new data items select an arbitrary location in memory, and updates to data items overwrite the previously-chosen location.) The methods in this group judiciously pick a memory location that is similar to the value to be written. When the new value and the value to be overwritten are similar, this means that the number of bit flips is going to be lower. There are lots of methods that use this approach as the backbones of their methods, which will be explored in Section. 4.2.

3 Techniques for Protecting NVM

In this section, we present the main concerns, challenges, and limitations of state-of-the-art methods that have utilized NVMs in their designs by dividing them into three main groups based on the trends and solutions they propose to solve the problem of write endurance and energy consumption. We also identify the short- and long-term research opportunities in this space.

3.1 Wear leveling

Wear leveling has been studied extensively for Flash-based storage devices [32–36]. In these methods, the wear-leveling algorithms usually keep track of the storage blocks and remap those blocks that are written heavily in a given time

quanta to the lowest wear-out blocks. In storage class memory, wear leveling has almost the same objective, which is extending the lifetime of NVM devices by distributing the writes evenly across the memory blocks of NVM so that no *hot* area reaches its maximum lifespan by extremely high concentration of write operations [13, 17, 22, 23, 37–39]. These methods usually are implemented in the memory controller level to protect NVMs. Wear-leveling is usually transparent for upper-level applications and they can simply access to the same content using the same logical address (they are unaware of the physical address where the data are stored.) According to the mapping strategies, existing wear-leveling schemes can be divided into two main groups:

(1) table-based wear-leveling (TBWL) techniques that store the logical addresses (LA), their corresponding physical addresses (PA), and the frequency of accesses. In this way, the storage table of the wear-leveling can be eliminated. When the number of writes to a specific physical address (PA) goes beyond a threshold, its content is swapped by the wear-leveling method [13, 40].

For example, In [41], the authors propose Fine-Grain Wear Leveling (FGWL), which shifts cache lines within a page to achieve uniform wear out of all lines in the page. Also, when a PCM page is read, it is realigned. The pages are written from the Write Queue to the PCM in a line-shifted format. For a system with 4 lines per page, for instance, the rotate amount is between 0 and 3 lines. The rotate value of 0 means the page is stored in a traditional manner. If it is 1, then the Line 0 of the address is being shifted one line and stored in Line 1 of the physical PCM page, line 1 of the address in stored in line 2, line 2 in 3, and finally, line 3 of address space is stored in Line 0. When a PCM page is read, it is realigned. The pages are written from the Write Queue to the PCM in a line-shifted format.

Self-adaptive wear-leveling (SAWL) algorithm [13] is another wear-leveling scheme that dynamically adjusts the wear-leveling granularity to accommodate more useful addresses in the cache, thus improving cache hit rate. This method distributes the writes across the cells of entire memory, thus achieving suitable tradeoff between the lifetime and cache hit rate.

(2) Algebraic-based wear-leveling methods [38, 42, 43] are another group that try to distribute the incoming writes to avoid concentrating writes on specific physical locations and creating the hot areas. To this end, they usually replace the address-mapping table in the table-based wear-leveling algorithms with hardware structure, which are more space efficient [38]. In this way, they can increase the lifetime of NVMs orders of magnitude compared to the based line where there is no wear-leveling.

Start-Gap [38] is one of the first methods that proposed a wear-leveling method based on the algebraic mapping between the logical and physical addresses. In this technique, there are two registers (Start and Gap) that do the wear-leveling. When a new write comes to the memory, Start-Gap moves one line from its location to a neighboring location. While the Gap register keeps track of the number of lines moved, Start register counts the number of times that all the available lines have moved. Finally, the mapping between

logical and physical address is done by a simple arithmetic operation of Gap and Start registers, which eliminates the need for storing the address-mapping table in the memory.

In [44], the authors propose a hardware-based wear-leveling scheme named Security Refresh, which performs dynamic randomization for placing PCM data. In this method, an embedded controller inside each PCM is responsible for preventing adversaries from tampering the bus interface or aggregating meaningful information via side channels [44]. They also applied designed some attacks to analyze the wear-out distribution using Security Refresh.

Although wear-leveling strategies have been successful in preventing creation of hot locations and extending the lifetime of NVMs, the controller cannot guarantee that some cells will not wear out much faster than the average. The reason is that distributing writes evenly across the memory space does not necessarily mean that the individual cells within the words also be flipped/written evenly. That is why, in some extreme cases, even with the protection of state-of-the-art wear-leveling schemes, wear-out attacks such as Remapping Timing Attack[15] and Row Buffer Hit [16] can wear out NVM as fast as 137 seconds [16]. Therefore, hardware techniques such as FNW [22], CDE [23], FPC [45], and Flip-Mirror-Rotate [25], which will be explained in the next section, have been proposed to focus on reducing the number of bit flips within a given word instead of just distributing writes uniformly across the device.

3.2 Reducing write amplification

Many data storage and indexing solutions target the reduction of write amplification to optimize the utilization of I/O bandwidth. This is done via various techniques, including delaying the consolidation of writes [30, 31], caching [27–29], and others [10]. With the introduction of NVM to the memory hierarchy, it turns out that reducing write amplification can have the positive side-effect of increasing NVM write endurance since less data is written. However, this is not an easy task to do due to the fact that all the existing data structures and database systems have been designed for DRAMs and HDDs, where the challenges of the lifespan of memory segments and the energy consumption of writes are not as significant in DRAM/HDD as they are in NVM. However, as discussed before, when it comes to NVMs, write operation needs to be performed wisely. So, the proposed methods in this group reduce the write amplification in an attempt to decrease the average number of updated cells and as a result increases the lifetime of NVMs.

To achieve this, many methods re-design existing data structures and database systems to mitigate the write amplification issue caused by them instead of designing and building new ones from scratch. The reason behind this is that existing data structures and database systems have undergone decades of research that makes them extremely efficient and makes building alternatives from scratch an arduous task.

Log-Structured Merge-tree (LSM-tree) is one of those data structures that has been widely adopted for use in the storage layer of modern NoSQL systems, and as a result, has attracted a large body of research, from both the database community and the storage systems community, that try to improve various aspects of LSM-trees by using NVMs [31, 46, 47]. NoveLSM [31] is one of these methods. This method is a persistent LSM-based key-value storage system designed to take advantage of having a non-volatile memory in its design. To tackle NVM's limited write endurance, NoveLSM comes up with a new design, where only the parts of the key/value store that do not need to be changed frequently, such as immutable memtables, are handled by NVM. On the other hand, other parts, such as mutable memtables, which need constant updates and data movements, are placed on DRAM, which do not have any restrictions on write operation. WiscKey [46] is another work, which proposes a persistent LSM-tree-based key-value store, which has been derived from the popular LSM-tree implementation, LevelDB. Although, like the other methods in this category, WiscKey focuses on decreasing write amplification, it achieve this through a different and simple way, which is separating keys from values. This method observes that since the indexing is done by keys, and not values, they do not need to be bundled together when they are stored in the LSM-tree. So, in this method, only keys are kept sorted in the LSM-tree, while values are stored separately in a log. Through this insight, they have reduced write amplification by avoiding the unnecessary movement of values while sorting. Although this technique is originally proposed for SSDs, it can be generalized to storage class memories, which suffer from the same limitation.

Another data structure that has been redesigned to utilize NVMs is B+-Trees, which is used widely in K/V data stores [28, 48]. Fingerprinting Persistent Tree (FPTree) [29] is a hybrid SCM-DRAM persistent and concurrent B+-Tree that is designed specifically for NVMs. This method aims to decrease write amplification on NVMs. To do so, in this method, leaf nodes are persisted in SCM while inner nodes are placed in DRAM and rebuilt upon recovery.

Hash-based indexing structures have also been good candidates to utilize NVMs due to their nature of typically causing high write amplification and that they are vastly used in various applications and systems [10, 14, 49, 50]. A lot of effort has been made to improve hash-based indexing structures for byte-addressable persistent memory, and almost all of them focus on decreasing the write amplification to reach their goal. Path hashing [10] is an example of these hash-based indexes, which is designed specifically for NVMs. The basic idea of path hashing is to leverage a position sharing method to resolve the hash-collision problem, which usually results in a high number of extra writes or write amplifications.

All these methods offer different advantages and disadvantages. These methods invite exploring how they can be integrated with existing data management systems to enable them to improve the lifetime of NVMs. Some of these methods are independent from the application (and often implemented as

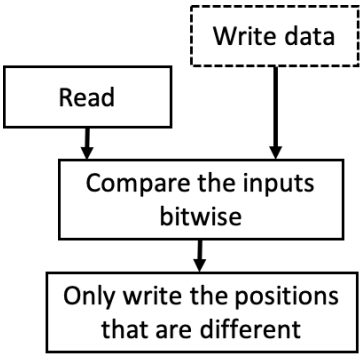

Fig. 6 Read Before Write (RBW) technique.

a hardware method) which means that augmenting them within existing data management systems is a straight-forward task. Other approaches—especially ones based on masking—require domain knowledge on the application using them. There is an opportunity for data management researchers to find ways to adapt these methods to work with existing data management systems. This would entail learning the write patterns of data management systems and translating this knowledge into appropriate masking techniques that are based on the methods presented above.

4 Bit Flip Reduction

Although reducing write amplification is a promising way to extend the NVMs' lifespan, it does not necessarily lead to the best opportunities to reduce bit flipping and increasing write endurance [8, 11]. This is because—unlike flash—NVM cells are written individually, which means that the number of flipped bits is more important to optimize than the number of written words [8]. Therefore, focusing on reducing bit flips is a viable solution that can both save energy and extend the life of NVMs. For example, in [8], the authors modifies the common data structures based on the idea of pointer distance to minimize the number of bit flips on NVMs. In this method, instead of building a doubly-linked list, for instance, XOR linked lists are used, which allows each node to store only the XOR between the previous and next node instead of storing the previous and next nodes. The results show that storing the XOR of two pointers, which are likely to contain similar higher-order bits, reduces the number of bit flips, which can lead to reducing power consumption.

Generally, the existing bit flipping reduction methods can be divided into two main categories: RBW-based techniques and Memory-awareness.

4.1 RBW-based techniques

In the NVM storage community, this method has been one of the simplest and most effective in dealing with the limitations of NVMs. So, there has been a

large body of research that uses various types of this method in their works. In this category, there are various techniques, such as caching [27–29] and the Read-Before-Write (RBW) technique [21], to decrease the number of bit flips.

RBW is one of the most popular techniques, which has been widely utilized by various approaches [22–26], to reduce the number of bit flips is the Read-Before-Write (RBW) technique [21], in which the content of an old memory block is read before it is overwritten with the new data (Figure. 4.1). This technique replaces each NVM write operation with a more efficient read-modify-write operation. Reading before writing allows comparing the bits of the old and new data, updating only the bits that differ.

Flip-n-Write (FNW) [22] is one of the most popular methods and became the building block of many other techniques in this area. This method compares the current content of the memory location (the old data) with the content to-be-written (the new data). This enables FNW to decide whether to write the new data in its original format or to flip it before writing it if that leads to reducing the number of bit flips. (A flag is used so that future operations know whether to flip the content before reading.) This method guarantees that the number of bit flips in NVM is always less than half the total number of written bits (excluding the flag bit).

DCW [23] finds common patterns and then compresses data to reduce the number of bit flips in NVM. Like Flip-N-Write, DCW replaces a write operation with a read-modify-write process. It starts comparing the new data and the old data from the first bit to the last one. The most significant difference between DCW and Frip-N-Write is that in DCW, the maximum number of bit flips is still N (the word width).

Captopril [24] is another recent proposal for reducing bit flips in NVMs. This method masks some "hot locations", where bits are flipped more, to reduce the number of bit flips. In this method, the authors compare every write with 4 predefined sequences of bits to decide which bits need to be flipped and which ones need to be written in their original form. This method suffers from relatively high overhead. More importantly, it is rigid and would only work on predefined applications.

Flip-Mirror-Rotate [25] is another method that is built upon Flip-N-Write [22] and FPC [26] to reduce the number of flipped bits. Like Captopril, this method uses only predefined patterns to mask some bits, which means it would only work on predefined applications.

MinShift [51] proposes reduces the total number of update bits to SCMs. The main idea of this method is that if the hamming distance falls between two specific bounds, the new data is rotated to change the hamming distance. Although this method is simple, it suffers from high overhead.

In [52], the authors use a combination of MinShift and Flip-N-Write to decrease the number of written bits. They compute the minimum amount of some possible states to choose a pattern to encode the data. This method has advantages and disadvantages of both methods.

4.2 Memory-awareness

As we discussed earlier, the writing operations in NVM takes much more energy than reading operation. To save energy, the PCM controller can avoid writing to a cell whose content is the desired value. It means that the lifetime of the cell and the consumed energy in the write operation depends on the number of bits that are actually being flipped by the write rather than the number of words or bits that are written.

Although focusing on bit flipping reduction technique seems a reasonable choice, the methods in this category fail to achieve its full potential because the existing methods miss a crucial opportunity. Prior methods pick the memory location for a write operation arbitrarily (new data items select an arbitrary location in memory, and updates to data items overwrite the previously-chosen location.) This misses the opportunity to judiciously pick a memory location that is similar to the value to be written. When the new value and the value to be overwritten are similar, this means that the number of bit flips is going to be lower. Reducing the number of bit flips increases write endurance and reduces power consumption. This approach is called memory awareness.

4.2.1 content-aware methods

The methods in this group, by being aware of the memory content, try to redirect write requests to overwrite specific memory locations based on their content-aware replacement policies to decrease the energy consumption of the system and extend the lifetime of the device. For example, the authors in [39] proposes an encoded content-aware cache replacement policy to reduce the total switched bits in spin-torque magnetic random-access memory (STT-MRAM) caches. To do this, instead of replacing the LRU block, the victim block is chosen among the blocks whose contents are most similar to the missed one. To avoid the comparison of the entire 512 bits of the blocks, each block is encoded using 8 bits, which incur low space overhead. The reason behind this encoding is that when the contents of two blocks are dominated by certain bit value, there is a good chance that the content similarity of the two blocks is high, hence may lower the switch bits when one double word replaces the other [39].

Data Content-aware (DATACON) [17] is a recent mechanism that reduces the latency and energy of PCM writes by redirecting the write requests to a new physical address within memory to overwrite memory locations containing all-zeros or all-ones depending on the content of the incoming writes. DATACON is implemented inside the memory controller. To keep track of the all-zeros and all-ones memory locations and their address translations, the memory controller needs to maintain a table. Although this method takes advantage of being content aware, it needs to be implemented inside the existing memory controller, which is non-trivial task. Moreover, the average number of bit flips it can save is highly dependent on the workload since it uses two fixed patterns to save bit flips.

Another method that leverages memory awareness in its structure is called Hamming-Tree [12], which is an auxiliary data structure that can be augmented with existing indexes. Hamming-Tree is a data structure that organizes free memory locations based on their hamming distance. It can be built upon any existing tree-based data structure—whether they are designed for NVM or not—to improve their performance in terms of NVM write endurance. One of the unique qualities of this method, which makes it highly adoptable by any existing key/value stores, is its ability to be augmented with a data indexing structure from B+-tree to LSM-based persistent K/V stores to cache optimized NVM index, and write-friendly hashing schemes. In this method, the data indexing structure handles the regular indexing of keys and values, and Hamming-Tree handles the mapping of free memory locations for future writes and updates. This method also reduces bit flipping considerably.

4.2.2 AI-based methods

Machine learning and deep learning are changing the world by transforming all segments from power systems to transport to storage systems and database systems [53–55]. Using machine learning in the field of NVMs is not new, but utilizing a machine learning-based method to extend the lifetime of NVMs was introduced in [11].

Predict and Write (PNW) [11] is a memory-aware mechanism that uses machine learning to extend the lifetime of NVMs. PNW is a K/V store that is designed specifically for NVMs. This method uses a clustering-based approach to extend the lifetime of NVMs using machine learning. Writes are directed to clusters with similar content to reduce the number of bit flips. Like the previous methods in Section 4.1, PNW also targets bit flip reduction but through software techniques. PNW decreases the number of bit flips for PUT/UPDATE operations by determining the best memory location an updated value should be written to. This method leverages the indirection level of K/V-stores to freely choose the target memory location for any given write based on its value. In this method, NVM addresses are organized in a dynamic address pool clustered by the similarity of the data values they refer to.

This recent AI-based direction has a high potential to improve the performance of the existing wear-leveling methods that use conventional techniques such as fixed patterns. Also, this new direction is in its infancy and can be improved in many ways, such as using efficient autonomous clustering methods [53, 56] or utilize advanced deep learning methods that are capable of learning the existing patterns among the existing data dynamically [57].

5 Conclusion and future work

NVMs promise to be an indispensable part of future memory systems due to their unique characteristics such as non volatility, byte addressability, high density, high scalability, and requiring near-zero standby power. They can revolutionize the performance, energy efficiency, and processing footprint of

existing systems from storage systems to edge and cloud environments to distributed database systems and blockchain decentralized applications [5, 58–63]. However, their main limitations, especially limited write endurance and high write energy consumption, pose serious challenges for their full adoption, which needs to be taken into consideration to leverage their full potential.

There is an opportunity now for researchers in data management systems to adopt solutions to overcome these limitations of NVMs that would be essential for their adoption and success. Specifically, in this paper, we present the low-hanging fruits and approaches of augmenting existing techniques from the NVM storage community to be adopted in data management systems. Also, we outline future opportunities in the area of memory-awareness that promises to increase the efficacy of existing techniques to improve the lifetime and energy efficiency of NVM devices.

References

[1] Hameed, F., *et al.*: Efficient stt-ram last-level-cache architecture to replace dram cache. In: MEMSYS 2017, pp. 141–151 (2017)

[2] Caulfield, A.M., De, A., Coburn, J., Mollow, T.I., Gupta, R.K., Swanson, S.: Moneta: A high-performance storage array architecture for next-generation, non-volatile memories. In: 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 385–395 (2010). IEEE

[3] Dulloor, S.R., Kumar, S., Keshavamurthy, A., Lantz, P., Reddy, D., Sankaran, R., Jackson, J.: System software for persistent memory. In: Proceedings of the Ninth European Conference on Computer Systems, pp. 1–15 (2014)

[4] Kargar, S., Nawab, F.: Extending the lifetime of nvm: challenges and opportunities. Proceedings of the VLDB Endowment **14**(12), 3194–3197 (2021)

[5] Mittal, S., Vetter, J.S.: A survey of software techniques for using non-volatile memories for storage and main memory systems. TPDS 2015 **27**(5), 1537–1550 (2015)

[6] Xia, F., *et al.*: A survey of phase change memory systems. Journal of Computer Science and Technology **30**(1), 121–144 (2015)

[7] Boukhobza, J., Rubini, S., Chen, R., Shao, Z.: Emerging nvm: A survey on architectural integration and research challenges. ACM Transactions on Design Automation of Electronic Systems (TODAES) **23**(2), 1–32 (2017)

[8] Bittman, D., *et al.*: Optimizing systems for byte-addressable {NVM} by reducing bit flipping. In: 17th {USENIX} Conference on File and Storage

Technologies ({FAST} 19), pp. 17–30 (2019)

[9] Intel® Optane™ Technology Delivers New Levels of Endurance. https://www.intel.com/content/www/us/en/architecture-and-technology/optane-technology/delivering-new-levels-of-endurance-article-brief.html (2022-03-28)

[10] Zuo, P., Hua, Y.: A write-friendly hashing scheme for non-volatile memory systems. In: Proc. MSST (2017)

[11] Kargar, S., Litz, H., Nawab, F.: Predict and write: Using k-means clustering to extend the lifetime of nvm storage. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 768–779 (2021). IEEE

[12] Kargar, S., Nawab, F.: Hamming tree: The case for memory-aware bit flipping reduction for nvm indexing. In: CIDR (2021)

[13] Huang, J., Hua, Y., Zuo, P., Zhou, W., Huang, F.: An efficient wear-level architecture using self-adaptive wear leveling. In: 49th International Conference on Parallel Processing-ICPP, pp. 1–11 (2020)

[14] Huang, K., Yan, Y., Huang, L.: Revisiting persistent hash table design for commercial non-volatile memory. In: 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 708–713 (2020). IEEE

[15] Huang, F., Feng, D., Xia, W., Zhou, W., Zhang, Y., Fu, M., Jiang, C., Zhou, Y.: Security rbsg: Protecting phase change memory with security-level adjustable dynamic mapping. In: 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 1081–1090 (2016). IEEE

[16] Mao, H., Zhang, X., Sun, G., Shu, J.: Protect non-volatile memory from wear-out attack based on timing difference of row buffer hit/miss. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017, pp. 1623–1626 (2017). IEEE

[17] Song, S., Das, A., Mutlu, O., Kandasamy, N.: Improving phase change memory performance with data content aware access. In: Proceedings of the 2020 ACM SIGPLAN International Symposium on Memory Management, pp. 30–47 (2020)

[18] Kamath, A.K., et al.: Storage class memory: Principles, problems, and possibilities. arXiv preprint arXiv:1909.12221 (2019)

[19] Song, S., Das, A., Mutlu, O., Kandasamy, N.: Aging-aware request scheduling for non-volatile main memory. In: Proceedings of the 26th Asia and South Pacific Design Automation Conference, pp. 657–664 (2021)

[20] https://pmem.io (accessed: 2022-02)

[21] Yang, B.-D., *et al.*: A low power phase-change random access memory using a data-comparison write scheme. In: ISCAS 2007, pp. 3014–3017 (2007). IEEE

[22] Cho, S., Lee, H.: Flip-n-write: A simple deterministic technique to improve pram write performance, energy and endurance. In: MICRO 2009, pp. 347–357 (2009)

[23] Dgien, D.B., *et al.*: Compression architecture for bit-write reduction in non-volatile memory technologies. In: NANOARCH 2014, pp. 51–56 (2014). IEEE

[24] Jalili, M., Sarbazi-Azad, H.: Captopril: Reducing the pressure of bit flips on hot locations in non-volatile main memories. In: DATE 2016, pp. 1116–1119 (2016). IEEE

[25] Palangappa, P.M., Mohanram, K.: Flip-mirror-rotate: An architecture for bit-write reduction and wear leveling in non-volatile memories. In: GLSVLSI 2015, pp. 221–224 (2015)

[26] Alameldeen, A., Wood, D.: Frequent pattern compression: A significance-based compression scheme for l2 caches. Technical report, University of Wisconsin-Madison Department of Computer Sciences (2004)

[27] Arulraj, J., *et al.*: Let's talk about storage & recovery methods for non-volatile memory database systems. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 707–722 (2015)

[28] Chen, S., Jin, Q.: Persistent b+-trees in non-volatile main memory. Proceedings of the VLDB Endowment **8**(7), 786–797 (2015)

[29] Oukid, I., *et al.*: Fptree: A hybrid scm-dram persistent and concurrent b-tree for storage class memory. In: Proceedings of the 2016 International Conference on Management of Data, pp. 371–386 (2016)

[30] Li, W., *et al.*: Hilsm: an lsm-based key-value store for hybrid nvm-ssd storage systems. In: Proceedings of the 17th ACM International Conference on Computing Frontiers, pp. 208–216 (2020)

[31] Kannan, S., *et al.*: Redesigning lsms for nonvolatile memory with novelsm. In: 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18), pp. 993–1005 (2018)

[32] Liu, Z., Liu, T., Guo, J., Wu, N., Wen, W.: An ecc-free mlc stt-ram based

approximate memory design for multimedia applications. In: 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 142–147 (2018). IEEE

[33] Kokolis, A., Skarlatos, D., Torrellas, J.: Pageseer: Using page walks to trigger page swaps in hybrid memory systems. In: 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 596–608 (2019). IEEE

[34] Kültürsay, E., Kandemir, M., Sivasubramaniam, A., Mutlu, O.: Evaluating stt-ram as an energy-efficient main memory alternative. In: 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 256–267 (2013). IEEE

[35] Gleixner, B., Pellizzer, F., Bez, R.: Reliability characterization of phase change memory. In: 2009 10th Annual Non-Volatile Memory Technology Symposium (NVMTS), pp. 7–11 (2009). IEEE

[36] Ban, A.: Wear leveling of static areas in flash memory. Google Patents. US Patent 6,732,221 (2004)

[37] Che, Y., Yang, Y., Awad, A., Wang, R.: A lightweight memory access pattern obfuscation framework for nvm. IEEE Computer Architecture Letters **19**(2), 163–166 (2020)

[38] Qureshi, M.K., Karidis, J., Franceschini, M., Srinivasan, V., Lastras, L., Abali, B.: Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling. In: 2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 14–23 (2009). IEEE

[39] Zeng, Q., Peir, J.-K.: Content-aware non-volatile cache replacement. In: 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 92–101 (2017). IEEE

[40] Zhou, P., Zhao, B., Yang, J., Zhang, Y.: A durable and energy efficient main memory using phase change memory technology. ACM SIGARCH computer architecture news **37**(3), 14–23 (2009)

[41] Qureshi, M.K., *et al.*: Scalable high performance main memory system using phase-change memory technology. In: ISCA '09, pp. 24–33 (2009)

[42] Seong, N.H., Woo, D.H., Lee, H.-H.S.: Security refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. ACM SIGARCH computer architecture news **38**(3), 383–394 (2010)

[43] Thoziyoor, S., Ahn, J.H., Monchiero, M., Brockman, J.B., Jouppi, N.P.: A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies. ACM SIGARCH Computer Architecture News **36**(3), 51–62 (2008)

[44] Young, V., Nair, P.J., Qureshi, M.K.: Deuce: Write-efficient encryption for non-volatile memories. ACM SIGARCH Computer Architecture News **43**(1), 33–44 (2015)

[45] Guo, Y., Hua, Y., Zuo, P.: Dfpc: A dynamic frequent pattern compression scheme in nvm-based main memory. In: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1622–1627 (2018). IEEE

[46] Lu, L., Pillai, T.S., Gopalakrishnan, H., Arpaci-Dusseau, A.C., Arpaci-Dusseau, R.H.: Wisckey: Separating keys from values in ssd-conscious storage. ACM Transactions on Storage (TOS) **13**(1), 1–28 (2017)

[47] Dai, Y., Xu, Y., Ganesan, A., Alagappan, R., Kroth, B., Arpaci-Dusseau, A., Arpaci-Dusseau, R.: From wisckey to bourbon: A learned index for log-structured merge trees. In: 14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20), pp. 155–171 (2020)

[48] Hu, J., et al.: Understanding and analysis of b+ trees on nvm towards consistency and efficiency. CCF Transactions on High Performance Computing, 1–14 (2020)

[49] Ma, Z., Sha, E.H.-M., Zhuge, Q., Jiang, W., Zhang, R., Gu, S.: Towards the design of efficient hash-based indexing scheme for growing databases on non-volatile memory. Future Generation Computer Systems **105**, 1–12 (2020)

[50] Xia, F., *et al.*: Hikv: A hybrid index key-value store for dram-nvm memory systems. In: USENIX ATC 17, pp. 349–362 (2017)

[51] Luo, X., *et al.*: Enhancing lifetime of nvm-based main memory with bit shifting and flipping. In: RTCSA 2014, pp. 1–7 (2014). IEEE

[52] Dong, W., *et al.*: Minimizing update bits of nvm-based main memory using bit flipping and cyclic shifting. In: HPCC 2015, CSS 2015, and ESS 2015, pp. 290–295 (2015). IEEE

[53] Gu, B., Kargar, S., Nawab, F.: Efficient dynamic clustering: Capturing patterns fromhistorical cluster evolution. arXiv preprint arXiv:2203.00812 (2022)

[54] Andalibi, M., Hajihosseini, M., Teymoori, S., Kargar, M., Gheisarnejad, M.: A time-varying deep reinforcement model predictive control for dc

power converter systems. In: 2021 IEEE 12th International Symposium on Power Electronics for Distributed Generation Systems (PEDG), pp. 1–6 (2021). IEEE

[55] Kraska, T., *et al.*: The case for learned index structures. In: SIGMOD 2018, pp. 489–504 (2018)

[56] Huang, S., Kang, Z., Xu, Z., Liu, Q.: Robust deep k-means: An effective and simple method for data clustering. Pattern Recognition **117**, 107996 (2021)

[57] Sharif, A., Li, J.P., Saleem, M.A., Manogran, G., Kadry, S., Basit, A., Khan, M.A.: A dynamic clustering technique based on deep reinforcement learning for internet of vehicles. Journal of Intelligent Manufacturing **32**(3), 757–768 (2021)

[58] Gazzaz, S., Chakraborty, V., Nawab, F.: Croesus: Multi-stage processing and transactions for video-analytics in edge-cloud systems. arXiv preprint arXiv:2201.00063 (2021)

[59] Kargar, S., Mohammad-Khanli, L.: Fractal: An advanced multidimensional range query lookup protocol on nested rings for distributed systems. Journal of Network and Computer Applications **87**, 147–168 (2017)

[60] Nawab, F., Chakrabarti, D.R., Kelly, T., Morrey III, C.B.: Procrastination beats prevention: Timely sufficient persistence for efficient crash resilience. In: EDBT, pp. 689–694 (2015)

[61] Nawab, F.: Wedgechain: A trusted edge-cloud store with asynchronous (lazy) trust. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 408–419 (2021). IEEE

[62] Nawab, F., Sadoghi, M.: Blockplane: A global-scale byzantizing middleware. In: 2019 IEEE 35th International Conference on Data Engineering (ICDE), pp. 124–135 (2019). IEEE

[63] Gu, B., Li, Z., Liu, A., Xu, J., Zhao, L., Zhou, X.: Improving the quality of web-based data imputation with crowd intervention. IEEE Transactions on Knowledge and Data Engineering **33**(6), 2534–2547 (2019)

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- mybib.bib