Extending the Lifetime of NVM: Challenges and Opportunities

Saeed Kargar
UC Santa Cruz
Santa Cruz, USA
skargar@ucsc.edu

Faisal Nawab
UC Irvine
Irvine, USA
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 low write endurance and extending the lifetime of NVM devices.

PVLDB Reference Format:
Saeed Kargar and Faisal Nawab. Extending the Lifetime of NVM: Challenges and Opportunities. PVLDB, 14(12): 3194 - 3197, 2021.
doi:10.14778/3476311.3476406

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 [11], 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 [4, 10]. 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.

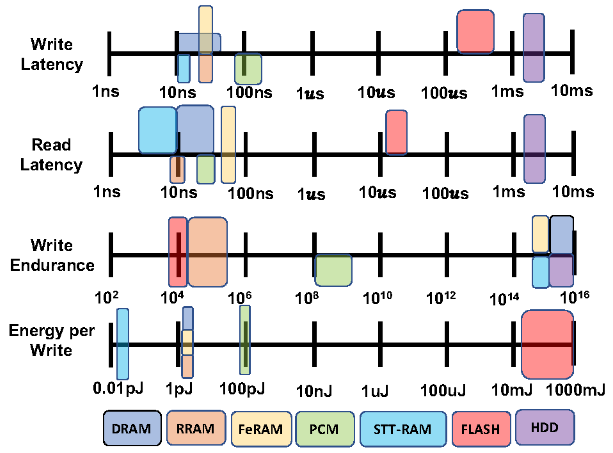

Figure 1: Comparison of device properties of memory technologies.

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 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.

2 TUTORIAL INFORMATION

Target audience. The target audience are database and storage researchers with basic knowledge of memory and database concepts. Some knowledge of well-known data structures, such as B-Trees and hash tables, read and write amplification, and basic operations of key/value stores are helpful but not necessary. For newcomers, the tutorial introduces NVMs, their main characteristics, and the methods used to integrate them into data management systems. For researchers experienced in NVMs, the tutorial provides a broader

Proceedings of the VLDB Endowment, Vol. 14, No. 12 ISSN 2150-8097.
doi:10.14778/3476311.3476406

view of NVM research beyond the work within the data management community. Generally, by introducing NVMs and their huge potential, we aim to enable researchers from different fields, such as Edge computing, data analysis, AI, and others, to extend their work and experience by using NVMs, which is possible by having better understanding of NVMs and their main challenges, especially in terms of their limited write endurance and high write energy consumption. To provide more depth to the presented topics, for each part we highlight and discuss in more details one or two representative systems.

Tutorial time and structure. The target length of the tutorial is 1.5 hours and it is divided into four sections: (1) Introduction (Section 1): This section introduces the unique characteristics of NVMs and the salient challenges of low write endurance and high energy consumption. It also showcases how NVMs are used in current data management systems. (2) Overcoming low write endurance via write amplification reduction (Section 3.1): This section shows the method that is typically adopted in data management systems which conflates write amplification reduction with improving write endurance. We will show how write amplification reduction helps in overcoming low write endurance and how it misses opportunities to improve write endurance further by conflating the two problems. (3) Overcoming low write endurance via local write optimizations (Section 3.2): This section consists of storage and software techniques that are designed specifically to overcome low write endurance via tecniques that aims at minimizing bit flips in the NVM device. (4) Overcoming low write endurance via memory-awareness (section 3.3): This section presents a recent technique that aims to overcome the problem of low write endurance by judiciously selecting memory locations for new writes with the goal of reducing bit flips.

3 TUTORIAL OUTLINE

In this work, 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 low write endurance. We also identify the short- and long-term research opportunities in this space.

3.1 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 [15, 18], caching [2, 5, 26], and others [30]. 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 [7, 15, 20]. NoveLSM [15] 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 [20] 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 [5, 12]. Fingerprinting Persistent Tree (FPTree) [26] 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 [13, 22, 28, 30]. 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 [30] 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.

In [3], the authors designed new data structures based on the idea of pointer distance. 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. Storing the XOR of two pointers, which are likely to contain similar higher-order bits, can lead to reducing the number of bit flips.

3.2 Local write optimizations

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 [2, 5, 26] and the Read-Before-Write (RBW) technique [29], to decrease the number of bit flips.

RBW is one of the most popular techniques, which has been widely utilized by various approaches [1, 6, 8, 14, 27], to reduce the number of bit flips is the Read-Before-Write (RBW) technique [29], in which the content of an old memory block is read before it is overwritten with the new data. 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) [6] 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 [8] 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 [14] 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 [27] is another method that is built upon Flip-N-Write [6] and FPC [1] 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 [21] 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 [9], 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.

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 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.

3.3 Memory-awareness

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 [3, 16]. 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 [3]. Therefore, focusing on reducing bit flips is a viable solution that can both save energy and extend the life of NVMs. Although focusing on bit flipping reduction technique seems a reasonable choice, the methods in this category (Section 3.2) 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.

The first memory-aware method that has been proposed to extend the lifespan of NVMs is called Predict and Write (PNW) [16], which 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 3.2, 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. In this paper, it has been shown that, by choosing the right target memory location for a given PUT/UPDATE operation, the number of total bit flips and cache lines can be reduced significantly.

Another method that leverages memory awareness in its structure is called Hamming-Tree [17], 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 OUTLOOK

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 tutorial, 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.

5 BIOGRAPHICAL SKETCHES

Saeed Kargar is a doctoral student at the University of California at Santa Cruz. His current research work is in the areas of data management on emerging non-volatile memory technology, machine learning and deep learning, and big data analytics.

Faisal Nawab is an Assistant Professor of Computer Science at the Computer Science Department at the University of California Irvine. He is interested in building efficient distributed data management systems that span edge and cloud infrastructures. Relevant to this tutorial, Faisal studies the design of data management systems that enable edge components to have a longer lifetime and better energy efficiency [16]. He also worked on the design of indexing technology over emerging memory technology [19, 23–25].

REFERENCES

[1] Alaa Alameldeen and David Wood. 2004. *Frequent pattern compression: A significance-based compression scheme for L2 caches.* Technical Report. University of Wisconsin-Madison Department of Computer Sciences.

[2] Joy Arulraj et al. 2015. 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.* 707–722.

[3] Daniel Bittman et al. 2019. Optimizing Systems for Byte-Addressable {NVM} by Reducing Bit Flipping. In *17th {USENIX} Conference on File and Storage Technologies ({FAST} 19).* 17–30.

[4] Adrian M Caulfield, Arup De, Joel Coburn, Todor I Mollow, Rajesh K Gupta, and Steven Swanson. 2010. Moneta: A high-performance storage array architecture for next-generation, non-volatile memories. In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture.* IEEE, 385–395.

[5] Shimin Chen and Qin Jin. 2015. Persistent b+-trees in non-volatile main memory. *Proceedings of the VLDB Endowment* 8, 7 (2015), 786–797.

[6] Sangyeun Cho and Hyunjin Lee. 2009. Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance. In *MICRO 2009.* 347–357.

[7] Yifan Dai, Yien Xu, Aishwarya Ganesan, Ramnatthan Alagappan, Brian Kroth, Andrea Arpaci-Dusseau, and Remzi Arpaci-Dusseau. 2020. From wisckey to bourbon: A learned index for log-structured merge trees. In *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20).* 155–171.

[8] David B Dgien et al. 2014. Compression architecture for bit-write reduction in non-volatile memory technologies. In *NANOARCH 2014.* IEEE, 51–56.

[9] Wei Dong et al. 2015. Minimizing update bits of NVM-based main memory using bit flipping and cyclic shifting. In *HPCC 2015, CSS 2015, and ESS 2015.* IEEE, 290–295.

[10] Subramanya R Dulloor, Sanjay Kumar, Anil Keshavamurthy, Philip Lantz, Dheeraj Reddy, Rajesh Sankaran, and Jeff Jackson. 2014. System software for persistent memory. In *Proceedings of the Ninth European Conference on Computer Systems.* 1–15.

[11] Fazal Hameed et al. 2017. Efficient STT-RAM last-level-cache architecture to replace DRAM cache. In *MEMSYS 2017.* 141–151.

[12] Jiangkun Hu et al. 2020. Understanding and analysis of B+ trees on NVM towards consistency and efficiency. *CCF Transactions on High Performance Computing* (2020), 1–14.

[13] Kaixin Huang, Yan Yan, and Linpeng Huang. 2020. Revisiting persistent hash table design for commercial non-volatile memory. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE).* IEEE, 708–713.

[14] Majid Jalili and Hamid Sarbazi-Azad. 2016. Captopril: Reducing the pressure of bit flips on hot locations in non-volatile main memories. In *DATE 2016.* IEEE, 1116–1119.

[15] Sudarsun Kannan et al. 2018. Redesigning LSMs for nonvolatile memory with NoveLSM. In *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18).* 993–1005.

[16] Saeed Kargar, Heiner Litz, and Faisal Nawab. 2021. Predict and Write: Using K-Means Clustering to Extend the Lifetime of NVM Storage. In *2021 IEEE 37th International Conference on Data Engineering (ICDE).* IEEE, 768–779.

[17] Saeed Kargar and Faisal Nawab. 2021. Hamming Tree: The Case for Memory-Aware Bit Flipping Reduction for NVM Indexing. (2021).

[18] Wenjie Li et al. 2020. HiLSM: an LSM-based key-value store for hybrid NVM-SSD storage systems. In *Proceedings of the 17th ACM International Conference on Computing Frontiers.* 208–216.

[19] David B Lomet and Faisal Nawab. 2015. High performance temporal indexing on modern hardware. In *2015 IEEE 31st International Conference on Data Engineering.* IEEE, 1203–1214.

[20] Lanyue Lu, Thanumalayan Sankaranarayana Pillai, Hariharan Gopalakrishnan, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. 2017. Wisckey: Separating keys from values in ssd-conscious storage. *ACM Transactions on Storage (TOS)* 13, 1 (2017), 1–28.

[21] Xianlu Luo et al. 2014. Enhancing lifetime of NVM-based main memory with bit shifting and flipping. In *RTCSA 2014.* IEEE, 1–7.

[22] Zhulin Ma, Edwin H-M Sha, Qingfeng Zhuge, Weiwen Jiang, Runyu Zhang, and Shouzhen Gu. 2020. Towards the design of efficient hash-based indexing scheme for growing databases on non-volatile memory. *Future Generation Computer Systems* 105 (2020), 1–12.

[23] Faisal Nawab, Dhruva Chakrabarti, Terence Kelly, and Charles Morrey. 2015. Zero-Overhead NVM Crash Resilience. In *Non-Volatile Memories Workshop.*

[24] Faisal Nawab, Dhruva R Chakrabarti, Terence Kelly, and Charles B Morrey III. 2015. Procrastination Beats Prevention: Timely Sufficient Persistence for Efficient Crash Resilience.. In *EDBT.* 689–694.

[25] Faisal Nawab, Joseph Izraelevitz, Terence Kelly, Charles B Morrey III, Dhruva R Chakrabarti, and Michael L Scott. 2017. Dalí: A periodically persistent hash map. In *31st International Symposium on Distributed Computing (DISC 2017).* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[26] Ismail Oukid et al. 2016. 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.* 371–386.

[27] Poovaiah M Palangappa and Kartik Mohanram. 2015. Flip-Mirror-Rotate: An architecture for bit-write reduction and wear leveling in non-volatile memories. In *GLSVLSI 2015.* 221–224.

[28] Fei Xia et al. 2017. Hikv: A hybrid index key-value store for dram-nvm memory systems. In *USENIX ATC 17.* 349–362.

[29] Byung-Do Yang et al. 2007. A low power phase-change random access memory using a data-comparison write scheme. In *ISCAS 2007.* IEEE, 3014–3017.

[30] Pengfei Zuo and Yu Hua. 2017. A write-friendly hashing scheme for non-volatile memory systems. In *Proc. MSST.*