A Survey of Software and Hardware techniques for using Non Volatile Memories

Tapojyoti Mandal, Sanket Agarwal and Dhiraj Kudva

Department of Electrical and Computer Engineering, Texas A&M University
{tapojyoti.mandal,agarwal.220196,dhirajkudva}@tamu.edu

Abstract—Non-volatile memories such as PCM, STT-RAM, ReRAM provide certain benefits over conventional memory storage technologies such as DRAM, SRAM and HDDs. But with the advantages come certain challenges which are specific to NVMs and need to be addressed in order to make them a viable competitior to the conventional technologies. In this paper, we survey a range of software and hardware techniques in order to address the disadvantages while extracting the advantages of NVM. The techniques address key issues with NVM such as endurance, access latencies and energy consumption to provide benefits which cannot be attained with conventional memories. With increasing demand of performance from computers, the memory system needs to scale along with processor performance so that it doesn't become a bottleneck. And NVMs are proving to be a popular and viable option to scale memory system performance. We hope that this survey will provide insights into emerging memory technologies and their application in modern high performance computer systems.

Index Terms—Non Volatile Memories, PCM, STT-RAM, SRAM, DRAM, SCM

1 Introduction

In modern computer systems, the memory system plays a crucial role in determining the performance. Processor performance has improved much faster as compared to memory technology, resulting in the memory becoming the bottleneck in many designs. Non-volatile memory technologies are beneficial as compared to traditional storage but in order to extract the best performance out of NVMs a range of architectural changes are required. The present software and hardware system were created keeping traditional memories in perspevtive, but with NVMs and their different properties, the software and hardware needs to change to tackle the challenges they pose and make NVM as memory storage viable. In Table 1, we have summarised a few impoartant parameters which shows how the NVMs differ from traditional technologies in terms of their physical device properties. We categorize the survey into three sections based on NVM usage as on-chip cache, main memory and storage.

2 NVM as Cache

SRAM has served as the dominant technology for on chip caches in processors for the past couple of decades. But SRAM has low density and very high leakage power consumption [9, 7] which makes it difficult to scale as we keep moving towards increasing number of cores and higher capacity LLC. NVMs have much higher density and lower leakage power as compared to SRAM with the added benefit of being non-volatile. And many prototypes have already shown promising results in support of NVM as on chip caches [13]. We discuss three key challenges that need to be adressed for utilizing their full potential.

2.1 Higher Access Latency

As shown in Table 1, NVMs usually have higher read/write access lattencies than SRAM. The objective here is to replace the L2 or L3 cache with NVM in a way that the AMAT(Average Memory Access Times) doesn't suffer a lot as compared to SRAM caches. Rucker et al [1] tested RRAM cache at L2 and L4 levels independently to study its impact on AMAT. With the L4 cache they didn't observe any significant gains, mostly because of the large 40MB L3 cache in their model absorbs most of the cache misses leaving very less for the L4 cache to make any impact. But their results are promising for NVM as L2 cache. Owing to high density of RRAM they were able to instantiate a 30 times larger L2 cache which helped reduce the miss rates at a slight higher penalty of hit time. But the overall AMAT was almost equivalent to that of SRAM cache. Zhao et al [18] used STT-RAM for their L3 cache and achieved an average performace in the range of 91% to 99% of that of SRAM cache. One important condition mentioned in both the research was that the NVM cache access latencies should not be greater than 5-10ns of that of SRAM, otherwise the higher hit time negates the performance gain obtained through higher capacity. The on-chip fabrication techninque plays a significant role in determining the acess latencies of NVM cache. Nevertheless both of the research papers provide promissing results for the idea of NVM L2 and L3 caches with AMAT similar to SRAM caches.

2.2 Write Endurance

One of the reasons NVMs are not preferred for L1 cache is the large number of writes that occurs at L1. Hence, most of the research focusses on L2 and L3 cache, but even here it is required that the amount of updates to cache blocks be controlled to avoid degradation of the durability of the caches. Wang et al [16] observed that the intra-set and inter-set variations in writes to L2 and L3 caches are two important factors which need to be addressed. For this they propose i2WAP, a wear leveling policy, which balances the writes to blocks not only between sets but also within a set. They use two techniques, one named Swap-Shift wherein they rotate the stored data between sets to address inter-set variation and the other technique name PoLF which maintains a counter to

Technology	Density(F^2)	Read Latency	Write Latency	Endurance
DRAM	6-10	50ns	1-5ns	$> 10^{15}$
Flash	4-6	25 µs	500 µs	$10^4 - 10^5$
SRAM	120-200	1-5ns	1-5ns	$> 10^{15}$
PCM	4-12	10-50ns	50-200ns	$10^8 - 10^9$
RRAM	4-10	10ns	25-50ns	10^{11}
STT-RAM	6-50	10ns	25-50ns	$4 * 10^{12}$

TABLE 1: Memory Technology Properties

monitor writes to blocks and flush the blocks when the counter saturates at a predetermined value. They achieved imrpovement in lifetime of on chip caches by 75% on average. Rucker et al [1] follows an objective of write minimization instead of wear leveling. They test multiple inclusion policies and replacement policies in their L2 cache with server workloads. Surprisingly, random inclusion policy and LRU replacement policy, the typical conventinal candidates, provided the best results for the lowest number of updates to cache on average. The authors speculate that the complicated access patterns in server workloads could be one of the reasons for such results. Rucker et al [1] use conventional replacement and inclusion policies to minimize the updates to cache while Wang et al [16] analyse the cache line access patterns in order to make microarchitecture changes to balance out the writes across cache lines. In both of the papers, authors have mentioned that due to the inter-set and intra-set variations, the wear leveling techniques used in NVM as main memory and storage hierarchy would not provide beneficial results at the cache hierarchy, thus requiring further cache line write variance analysis.

2.3 Dynamic Energy

Dynamic energy refers to the energy consumed during reading and writing memory cells in NVMs, which is usually calculated in nanoJoules/byte. NVMs have greater write dynamic energy than that of SRAM. For example, SRAM has 2.21 nJ/access while STT-RAM has 20.25 nJ/access which is a 4x increase [3].Smullen et al [14] use STT-RAM to implement L2 and L3 caches but reduces the dynamic energy by decreasing the planar area of each memory cell thus requiring less current to perform read and write operations. The sacrifice was the reduced retention time for each memory cell for which they had to implement a refresh scheme, something similar to refresh mechanism in DRAMs, to periodically refresh each memory cell. Through this they were able to obtain 70% improvement in dynamic power consumption but the paper hasn't made an analysis of the impact of refreshes on the write endurance of the cache. Hu et al [6] uses a hybrid combination of SRAM and NVM to implement their SPM(Scratch Pad Memory) which is a software controlled on-chip cache, typically implemented in embedded systems. They introduce Optimal Data Allocation(ODA), a dynamic data management algorithm which moves blocks having high write accesses to SRAM while blocks having high read acceses are placed in NVM. This allows taking advantage of low read latencies and avoid costly writes to NVM by shifting it to SRAM thus leading to improved lifetime for their product. Through their hybrid model they were able to reduce dynamic power by 27% and leakage power by 34% as compared to a fully SRAM design. The AMAT was reduced by 18% due to the higher capacity of NVM recuing the miss rate. However we need to keep in mind that it is the low leakage power of NVM, unlike SRAM, that make them the ideal technology for high capacity situations. Smullen et al's [14] idea provides a hardware based solution while Hu et al's [6] idea is to use software optimization to manage writes to the hybrid cache model. Both of the ideas can be implemented in combination to supplement each other.

3 SCM as Main Memory

3.1 Need of SCM as Main Memory

DRAM has served as the universal standard for memory in mainframes, laptops, and data centers [11]. But DRAMs account for significant cost and power budget of a computer system [12]. Also, as the number of cores in system increases, memory should be capable of supporting that growth. Thus, to be competitive in terms of performance, cost & power and simultaneously provide more memory capacity than DRAM, Storage Class Memories (SCM) seems to be a prime candidate to serve the next generation. Relative latencies of DRAM, SCM and FLASH are explained in [11] & [12] with DRAM being the quickest and FLASH being slowest. The access latency of PCM is close to DRAM but still slower. Thus, a hybrid memory solution is the best approach that achieves results very close to an expensive (4X more) only DRAM memory [15] [12] [11].

3.2 Feasibility

Dong Li [8] and Lloyd [10] evaluate the feasibility of using SCM as the main memory. Dong Li [8] states that SCM not only saves the leakage power issue but also improves performance. Using a hybrid DRAM-SCM model, NV-SCAVENGER instrumentation tool and four different types of scientific applications, they find PCM uses the least power compared to STTRAM, MRAM & DRAM. Varying access latencies (Table 1), they find performance loss is negligible until an increase of 20%. Lloyd [10] architecture replaces DRAM with SCM and evaluates the performance by varying system cache - SCM memory ratio (C:M). It just places an upper boundary on the ratio of architecture parameters. It was found that the runtime is least when C:M ratio was 1:1.

3.3 Architecture Design

Frietas, Wilcke et.al [5] & Qureshi [12] defines the hybrid memory system using Phase Change Memory (PCM) as the main memory and a DRAM cache which is other than the system cache.Qureshi [12] uses a normal 8GB DRAM with 16 cores. They suggest PCM is managed by the operating system using a Page Table like the current DRAM main memory systems. Whereas the DRAM cache is organized like a hardware cache that is not visible to the OS and is managed by the DRAM controller. In their model, they also suggest a write queue which overcomes the write latency & write endurance problem of PCM. They suggest three techniques to overcome the write latency and write endurance issues: Lazy-Write Organization, Line-Level Writes & Page Level Bypass. Using Lazy-Write Organization, they reduce the number of writes to PCM memory by not writing the page into memory in case of a Page Fault. Instead, the page is written directly to the DRAM cache. The

page is updated in PCM memory only when the page is evicted from the DRAM which is not present in PCM, identified using a 'P' tag bit, or when the Dirty Bit tag of the page in DRAM is set and is evicted. Using Line-Level Writes, they reduce the granularity of updating data to line level, instead of page level. Thus, if the dirty bit is set, the entire page is not updated rather only the dirty line is updated in the PCM memory. This is possible because of the byte accessibility of PCM memory. To avoid a specific line being rewritten always, they adopted a rotation policy that ensured each line is written an average number of times. Page-Level Bypass leverage the fact that not all applications have a reuse functionality. If a page of such an application sets the Page Level Bypass (PLB) bit then, that page is not written in the PCM memory even if the dirty bit is set. Using this configuration and techniques they found hybrid configuration provides performance benefits equivalent to a 4X DRAM only system (32 GB). Also, a hybrid system accounts for only about 13% area overhead while the DRAM-only system would require 4X area. Also, using the above-mentioned writing techniques, they reduced the write traffic by 3X and increase the average lifetime of PCM from 3 years to 9.7 years. Besides, the hybrid memory system consumes 17% less power compared to a 4X DRAM only system. These results confirm that the PCM-based hybrid memory is a practical power-performance efficient architecture to increase memory capacity.

3.4 Cache Design

Dong [4] and Ustiugov [15] also suggested a similar hybrid model as compared to [12]. However, instead of using a normal DRAM cache, they used a 3D stacked DRAM cache. By using 3D RAM as a cache, checkpoint overhead reduces to less than 5% even in exascale systems. In addition, they found an improvement in performance by 31% when compared to a single-level DRAM configuration which is 4 times larger in capacity than the 3D-RAM. In contrast to paper [12], they suggest amortizing the latency of SCM using bulk transfers. Thus, write endurance is not considered. It is assumed that the data blocks requested have spatial locality and bulk data present in DRAM cache would avoid the need to activate SCM read/write. Limitation occurs due to over fetching with 4KB blocks, causing bandwidth contention in the SCM. Further,Ustiugov [15] suggested that to design SCM memory, three parameters namely read latency, write latency and row buffer size control end-to-end applications. Besides, Dong [4] finds improvement in power consumption by using 3D-DRAM as PCM memory is kept switched off more than 95% of times.

4 SCM AS STORAGE MEMORY

SCM is being looked upon currently as the panacea for the bottlenecking of the memory performance when compared with the processor performance. However, the replacement of SCM as storage memory has its own advantages and disadvantages. This section focusses on these aspects of using SCM as storage memory, its limitation and a special management system to gain maximum performance from SCM.

4.1 SCM versus Traditional Disks

Frietas and Wilcke et.al [5] in their journal have described SCM as a disk drive replacement. SCM has random and sequential I/O performance that is orders of magnitude better than that of traditional disk-based systems and require much less space and power in the data center. Flash can be considered as an early form of SCM. However, its low write endurance (10^4 to 10^5) is a limitation in the path of replacement of disks as a storage memory. So in this section, we will be focussing on some other trends in SCM like Phase Change Memory, etc. PCM has three major advantages over flash: better size scaling, write-in place functionality and better write endurance. Frietas and Wilcke in their journal have provided a straight comparison of SCM with disk over various parameters like CPU cycles, Input Output operations per second, memory density and power. The access time in a disk is 10^7 to 10^8 CPU cycles, whereas in SCM it is 10^3 CPU cycles . To handle compute-centric workloads (used by researchers to solve complex problems), traditional disk can draw up to 25 MW of power [5]. Whereas the same performance is achieved by SCM using just 173 kW of power. All these comparison highlights that the SCM performs better than a traditional disk, with the only parameter where disk standout is in write endurance. Write endurance problem is more severe when it is used as main memory then as a storage memory. But can be avoided by using wear guarding techniques

4.2 Architecture with SCM

Apart from the component or device level, in order to extract better performance from SCM, we need to modify the architecture as well. In the scope of this survey, two architectures would be briefly discussed. The first one is called as the Dynamic Temperature-Aware Bufferpool Extension [2], where the SCM is moved up in the Storage level hierarchy, intermediate between the magnetic disk and the main memory. In this architecture, the SCM acts as an extension to the main memory bufferpool and adopts the role of a second level page cache. The second architecture, which is proposed by Freitas and Wilcke et.al. [5], is more focussed on reducing the hierarchy and suggests a flatter approach. They suggest an architecture where the SCM is used both as the main memory and storage memory and DRAM of gigabyte scale is used as cache. This realization of a flattened architecture greatly increases the speed and simplifies performance tuning.

4.3 Storage Class File Management System

This architecture can further be improved if we optimize the file management system. The system proposed by Wu and Reddy et.al. [17] is an ideal system for further improving the memory performance. Wu and Reddy proposed a new file management system by making sure that space allocation would be contiguous for each file. This file management system is named as SCFMS (Storage Class File Management System). It is necessary because the existing file systems are for volatile memory system and it does not consider the feature to restore the memory after boot. The SCFMS utilizes the existing memory management module in the OS. The traditional file systems assume the underlying storage devices are input bus attached block devices and not memory. Traditional file systems access storage devices through generic block layer and emulated block input out operations and hence contain overload. This overload is not necessary as the file system specially designed for memory devices can be built on top of the memory access interface directly. This would result in less overhead. In the SCFMS, file system is on virtual memory space and it utilizes the memory management unit to map the file system address to physical address on SCM. The concept of space pre allocation

reduces the number of allocation and deallocation operations significantly. In this concept, whenever a file is deleted or shrunk it does not deallocate that memory space but marks as null space. This null space is first considered whenever a new memory location is needed. The SCFMS contains garbage collection feature. In this, extra null files are removed in case it reaches the threshold performance overhead. This paper by Wu and Reddy covers major aspects but it does not consider some important conditions like defragmentation, large mapping table, TLB misses and wear levelling techniques

5 Conclusion and Future Work

NVM as Cache section covered research focusing on write minimization and wear leveling techniques to improve NVM on chip cache lifetime. Also we realized the fabrication of NVM on chip and its integration with the processor core will play a significant role in access times. NVM with its low leakage power and dynamic write management techniques can provide signifcant improvement in power consumption of caches. As seen above, SCM provides performance equivalent to a 4X DRAM only Main Memory system. However, the architecture to implement SCM depends on the type of application. A high spatial locality data can use denser DRAM cache thereby improving the performance/cost even further. Future improvements in the access latencies and the density would further push the usage of SCM as the main memory. SCM as a storage memory is better than disk when it comes latency, memory density and power. But a major improvement is needed in the wear guarding techniques to overcome the limitations of low write endurance. The flatter architecture ensures high performance. If this architecture is coupled with SCFMS, it will further reduce the access times and ensure proper utilization of storage. With this survery, we are able to depict that although NVMs have their challenges with proper technqiues the challenges can be mitigated to provider performance on par, and sometimes better, with conventional technologies. NVMs will play a significant role in improving modern computer systems and we hope that through further research we can leverage the potential of NVMs.

References

[1] C. C. Alexander Rucker, Andrew Bartolo, "NVM Cache with Predictive Allocation," https://web.stanford.edu/~bartolo/assets/nvm-cache.pdf, [Online; accessed 29-Sep-2019].

[2] B. Bhattacharjee, M. Canim, C. A. Lang, G. Mihaila, and K. A. Ross, "Storage class memory aware data management," in *IEEE Data Engineering Bulletin*, 2010, p. 33(4).

[3] M. Chang, P. Rosenfeld, S. Lu, and B. Jacob, "Technology comparison for large last-level caches (l3cs): Low-leakage sram, low write-energy stt-ram, and refresh-optimized edram," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2013, pp. 143–154.

[4] X. Dong, N. Muralimanohar, N. Jouppi, R. Kaufmann, and Y. Xie, "Leveraging 3d pcram technologies to reduce checkpoint overhead for future exascale systems," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, ser. SC '09. New York, NY, USA: ACM, 2009, pp. 57:1–57:12. [Online]. Available: http://doi.acm.org/10.1145/1654059.1654117

[5] R. F. Freitas and W. W. Wilcke, "Storage-class memory: The next storage system technology," in *IBM Journal of Research and Development 2008*. IBM, 2008, pp. 439–447.

[6] J. Hu, C. J. Xue, Q. Zhuge, W. Tseng, and E. H. . Sha, "Towards energy efficient hybrid on-chip scratch pad memory with non-volatile memory," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.

[7] C. H. Kim, Jae-Joon Kim, S. Mukhopadhyay, and K. Roy, "A forward body-biased-low-leakage sram cache: device and architecture considerations," in *Proceedings of the 2003 International Symposium on Low Power Electronics and Design, 2003. ISLPED '03.*, Aug 2003, pp. 6–9.

[8] D. Li, J. S. Vetter, G. Marin, C. McCurdy, C. Cira, Z. Liu, and W. Yu, "Identifying opportunities for byte-addressable non-volatile memory in extreme-scale scientific applications," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, May 2012, pp. 945–956.

[9] L. Li, I. Kadayif, Y. . Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and A. Sivasubramaniam, "Leakage energy management in cache hierarchies," in *Proceedings.International Conference on Parallel Architectures and Compilation Techniques*, Sep. 2002, pp. 131–140.

[10] S. Lloyd and M. Gokhale, "Evaluating the feasibility of storage class memory as main memory," in *Proceedings of the Second International Symposium on Memory Systems*, ser. MEMSYS '16. New York, NY, USA: ACM, 2016, pp. 437–441. [Online]. Available: http://doi.acm.org/10.1145/2989081.2989118

[11] S. Mittal and J. S. Vetter, "A survey of software techniques for using non-volatile memories for storage and main memory systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1537–1550, May 2016.

[12] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ser. ISCA '09. New York, NY, USA: ACM, 2009, pp. 24–33. [Online]. Available: http://doi.acm.org/10.1145/1555754.1555760

[13] S. Sheu, M. Chang, K. Lin, C. Wu, Y. Chen, P. Chiu, C. Kuo, Y. Yang, P. Chiang, W. Lin, C. Lin, H. Lee, P. Gu, S. Wang, F. T. Chen, K. Su, C. Lien, K. Cheng, H. Wu, T. Ku, M. Kao, and M. Tsai, "A 4mb embedded slc resistive-ram macro with 7.2ns read-write random-access time and 160ns mlc-access capability," in *2011 IEEE International Solid-State Circuits Conference*, Feb 2011, pp. 200–202.

[14] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, Feb 2011, pp. 50–61.

[15] D. Ustiugov, A. Daglis, J. Picorel, M. Sutherland, E. Bugnion, B. Falsafi, and D. Pnevmatikatos, "Design guidelines for high-performance scm hierarchies," in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS '18. New York, NY, USA: ACM, 2018, pp. 3–16. [Online]. Available: http://doi.acm.org/10.1145/3240302.3240310

[16] J. Wang, X. Dong, Y. Xie, and N. P. Jouppi, "i2wap: Improving non-volatile cache lifetime by reducing inter- and intra-set write variations," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2013, pp. 234–245.

[17] X. Wu and A. Reddy, "Scmfs: A file system for storage class memory," in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2011.

[18] J. Zhao, S. Li, D. H. Yoon, Y. Xie, and N. P. Jouppi, "Kiln: Closing the performance gap between systems with and without persistence support," in *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Dec 2013, pp. 421–432.

Appendix

1] Tapojyoti Mandal: Worked on Section 2 NVM as Cache
2] Sanket Agarwal: Worked on Section 3 SCM as Main Memory
3] Dhiraj Kudva: Worked on Section 4 SCM as Storage
4] Worked together on Abstract, Introduction and Conclusion