Leveraging Nonvolatility for Architecture Design with Emerging NVM

Shuangchen Li* Ping Chi* Jishen Zhao† Kwang-Ting Cheng* Yuan Xie*
Dept. of Electrical and Computer Engineering, University of California, Santa Barbara*
Computer Engineering Department, University of California, Santa Cruz†
{shuangchenli, pingchi, timcheng, yuanxie}@ece.ucsb.edu* jishen.zhao@ucsc.edu†

***Abstract*—Emerging nonvolatile memory (NVM), such as spin-transfer torque magnetic Memory (STT-RAM), phase-change memory (PCM), and resistive memory (ReRAM), are widely expected to become the next generation cache and main memory, in order to migrate the "power wall" and overcome the DRAM stability challenge. Previous effort has well explored NVM's feature of ultra-low leakage and high density at various memory hierarchy. Furthermore, challenges such as asymmetric read/write, expensive write operation, and limited lifetime have also been tackled. However, the benefit from NVM's nonvolatility has never been fully exploited. This paper points out the potential benefit by leveraging nonvolatility for architecture design. Two case studies are described. The first one is to leveraging multi-level cell (MLC) STT-RAM for ultra-low overhead local checkpointing. The second one is persistent memory design, which support persistency in NVM based main memory. Potential benefit and design challenge for those two cases are described. Future research topic around exploring NVM's nonvolatility is also discussed.**

I. Introduction

In conventional computer architecture design, SRAM and DRAM memories are the common memory embodiments at different memory levels, as the cache or main memory. Technology scaling of SRAM and DRAM is increasingly constrained by fundamental technology limits. In particular, there are three urgent challenges. First, the increasing leakage power of SRAM has been pushing us closer to the "power wall", and the increasing refresh dynamic power of DRAM has been a major issue for the energy hungry data center or mobile devise. Second, at the big era, memory with larger capacity and higher density are required by applications like in-memory-database. However, according to ITRS [1], the scalability of DRAM beyond 16nm is still unclear. Third, the significant degradation of the reliability of SRAM and DRAM have also posed challenges for circuit/architecture designers for future memory hierarchy design.

For decades, we have seen a lot of efforts to address the research and development of emerging non-volatile memory (NVM) technologies, e.g., Phase-Change RAM (PCM) [2], Spin-Transfer-Torque Magnetoresistive RAM (STT-MRAM) [3], and Resistive RAM (ReRAM) [4], [5]. By combining the speed of SRAM, the density of DRAM, and the nonvolatility of FLASH memory, these emerging NVM technologies have a great potential to be the universal memories of the future. Recently, lots of prototype chips have been demonstrated [6], [7]. Intel and Micron have just unveiled 3D Xpoint [8], the NVM technology for future product, with 1000× faster than FLASH and 10× denser then DRAM. It is anticipated that the emerging NVM technologies will break important ground and move closer to the market in the near future [9], [10].

As such emerging memory technologies are getting mature, it is important for computer architects to understand both their benefits and limitations, in order to better utilize them for the performance, power, and reliability improvement. We have seen an increasing interest on memory architecture research with focus on such emerging memory technologies in the last several years. The existing architecture research works are mainly focused on the following two aspects.

Exploring the performance/capacity/power benefits: There is a plenty of work using such emerging NVM as replacement for traditional SRAM-based cache [11] and DRAM-based main memory [12]. For example, Dong *et al.* [13] proposed a 3D cache architecture design using STT-MRAM as L2 or L3 cache, which can reduce system power significantly and improve system performance moderately. Xu *et al.* [14] architects the cross-point ReRAM as the main memory, which provides high density, promising scalability, and low energy consumption.

Overcoming the expensive-write/endurance etc. challenges: Such emerging NVM memory share common design challenges such as relatively longer write latency, larger write energy, limited write endurance (as compared with SRAM), and high variation. Therefore, many of the mitigation techniques are developed for NVMs. For example, hybrid DRAM/NVM main memory architecture [12] is proposed, where DRAM works as a buffer for the PCM. Read-before-write scheme [15] is proposed to reduce unnecessary write to the endurance limited PCM-based main memory. Similarly, other technique such as adaptive scheduling policy [16], data encoding [17] are proposed to deal with the endurance/latency challenges.

Named as "nonvolatile memory", however, the **nonvolatile characteristics of such emerging memory technologies are not fully explored in prior study**. Some researchers investigated methods to tradeoff the nonvolatility for better performance/power for NVM-based memory architecture (i.e., by sacrificing the data retention time, STT-MRAM cache can be made faster with lower write energy) [18]. Instead of tradeoff nonvolatility for something else, this paper points out that by smartly re-design the memory architecture, the nonvolatility itself brings performance or energy benefit. Two case studies are described in this paper. First, we show that by leveraging the special physical structure of the MLC

Li, Chi, and Xie were supported in part by NSF1500848, 1461698.

978-1-4673-6688-5/15/$31.00

STT-RAM, local checkpointing in main memory is able to be implement in an ultra-low overhead approach with large internal bandwidth [19]. Aware of the nonvolatility, this work utilizes the hard-bit in the MLC for backup while the soft-bit as working memory, so that a huge backup bandwidth is provided. Second, we show the design of persistent memory, which supports data persistency in a NVM-based memory hierarchy [20]. The proposed NVM-based persistent memory takes the advantage of the nature copies of data in both NV-chace and NV-main memory, and provides multi-version without any logging or copy-on-write overhead. Both of those two cases well exploit the nonvolatility of NVM and gains performance/energy benefit.

The remainder of this paper is organized as follows. Section II describes leveraging MLC STT-RAM for high speed local checkpointing. Section III describes how the nonvolatility of NVM benefits data persistency, as well as the design challenge for the proposed persistent memory. Section IV shows the vision for research that leveraging nonvolatilty for NVM. Section V draws the conclusion.

II. Checkpointing with Emerging NVM

In this section, we describe a local checkpointing architecture for large-scale computing system design based on the MLC STT-RAM.

A. Challenges for Checkpointing

In modern large-scale computing systems, high reliability, availability and serviceability (RAS) are mandatory. However, as the scale goes larger, it is very difficult and challenging to provide sufficient RAS. Even though for each node the mean time to failure (MTTF) is long, the entire system's MTTF could be very short, due the large scale. It is reported that for exascale era, the MTTF could be as short as few minutes [21]. Checkpoint is the most efficient way to deal improve RAS. It backups the current data to disk or other NVM, in order to rollback while encountering with system crash caused by, e.g., power loss, hardware/software error, or even human fault. The problem of checkpoint is the performance loss. For example, checkpointing takes about 50% performance overhead in a petaFLOPS system [22]. The checkpointing operation is typically bounded by the IO bandwidth between the working memory and backup memory/storage. How to implement checkpointing with large IO bandwidth and little performance overhead is challenging.

B. Using MLC STT-SRAM for Local Checkpointing

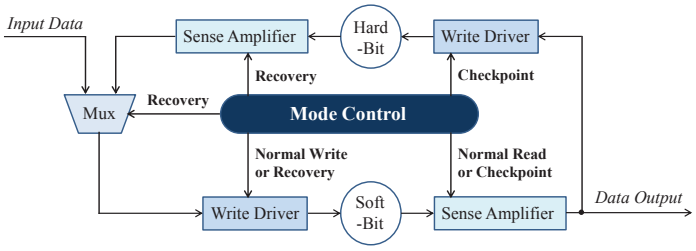

Fig. 1. The data flow diagram of the mode control of sense amplifiers and write drivers. [19]

The design [19] takes advantage of the MLC STT-MRAM's unique characteristics of write operations and ensures that only one-step writes occur during the entire execution time (including checkpoint and recovery) by using the soft-bit of each cell to store the working data and the hard-bit to save a local checkpoint. In this design, the sense amplifiers and write drivers are controlled by the "Mode Control" signal, as shown in Fig. 1. It can work in four modes: normal read, normal write, checkpoint, and recovery. In the normal read/write mode, only the soft-bit are accessed. In the checkpoint mode, the soft-bits (working data) are first sensed, and then values of "00" or "11" are written to the cells according to the values of the soft-bits. In recovery mode, the hard-bits (checkpoint data) are first sensed, and then values of "00" or "11" are written to the cells accordingly.

This method takes use of emerging NVM's nonvolatility, and therefore benefits from the large internal bandwidth for local checkpointing. The experiment result 63.16GB/s checkpointing bandwidth compared with 12.8GB/s for DDR and 0.467GB/s for using separated PCM as checkpointing. Fig. 2 shows that, compared with DRAM+PCRAM (DRAM as working memory while PCRAM as local storage for checkpointing), the proposed method reduces the performance overhead from $\sim$15% to $\sim$3%. The energy consumption is reduced from 0.116J to 0.025J.

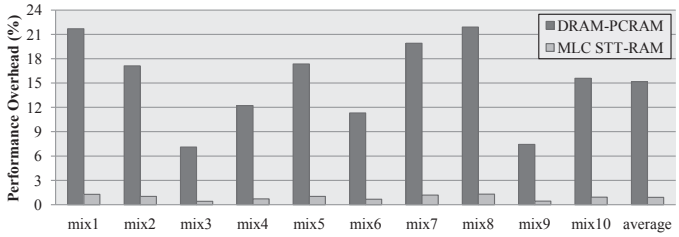

Fig. 2. Performance overhead of local checkpointing during error-free execution in a multiprogrammed four-core process node with checkpointing interval as 1s. [19]

C. Discussion and Future Work

There are few paper exploring fast and efficient checkpointing that taking advantages of the nonvolatility of emerging NVM. Future work could exploring the flexibility between working memory and backup memory. When part of the memory does not require checkpoint, the memory can morphic from 1-bit for working data and 1-bit for checkpoint data per MLC to 2-bit for working data per MLC, in order to provide larger capacity. Further improvement on optimizing checkpoint capacity overhead is also interesting.

III. Persistent Memory Architecture with Emerging NVM

In this section, we describe how the nonvolatility benefit the design of persistent memory. The Klin [20] system is described as an example.

A. Introduction to the Persistent Memory

In the conventional computer architecture, the working main memory requires fast access and byte-addressable. However, it losts data on a system crash/halt typically caused by

TABLE I. COMPARISON OF KILN WITH PREVIOUS WORK. (⋆ MEANS IN-PLACE UPDATES ARE ONLY PERFORMED FOR MEMORY STORES TO A SINGLE VARIABLE OR AT THE GRANULARITY OF THE BUS WIDTH. ◇ MEANS ORDERING IS MAINTAINED AMONG THE WRITES TO THE DISK OR FLASH BY FLUSH OR CHECKPOINTING. [20])

Designs	Mechanisms					Persistence Support	
	In-place	Logging	COW	clflush/fsync	mfence/barrier	Atomicity	Ordering
BPFS [23]	⋆	**No**	Yes	**No**	Yes	√	√
Mnemosyne [24]	⋆	Yes	Yes	Yes	Yes	√	√
NV-heaps [25]	No	Yes	**No**	**No**	Yes	√	√
CDDS [26]	No	**No**	Yes	Yes	Yes	√	√
UBJ [27]	**Yes**	Yes	Yes	◇	◇	√	√
eNVy [28]	No	**No**	Yes	◇	◇	√	√
Native System	**Yes**	**No**	**No**	**No**	**No**	×	×
Kiln [20]	**Yes**	**No**	**No**	**No**	**No**	√	√

power failure, soft-error, software bugs or even human faults. In order to keep data persistent, the permanent storage class memory is required, but it is slow and block-addressable. With the nonvolatility of the emerging NVM, a new tier as persistent memory is proposed, which is between memory and storage with attributes of both. With the memory attribute, it provides fast access through load/store interface; with the storage memory attribute, it supports data persistence.

The persistent memory requires the properties including atomicity, consistency, and durability as follows.

Atomicity: Each memory update must be "all or nothing", which means it either finishes the update operation successfully or fails completely. This is because that the granularity of user-defined data updates can be larger than the persistent memory interface's width. Therefore, one update is serviced by multiple memory requests. In this sense, if system crash (e.g., power losses) happens during one single update, only part of the date is updated, which violates the data persistency.

Consistency: Each memory update must convert the persistent data from one consistent state to another. For instance, if an application needs to insert a new node to a link list in the persistent memory, the initial values of the new node must be written into the persistent memory before the insertion operations are executed. Otherwise, if the system crash happens during the insertion, the pointer to the link list might be lost, which is a permanent corruption that cannot be recovered by simply reboot.

Durability: Each memory update must be retained during any system failure such as power loss, crashes.

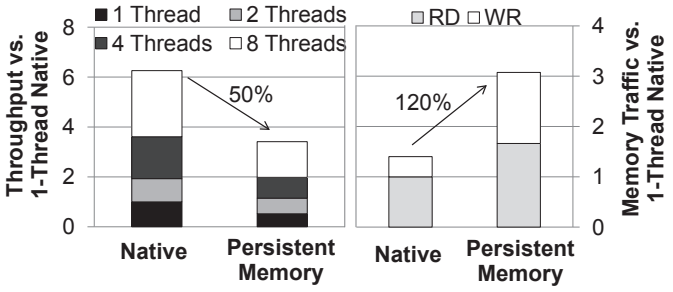

Fig. 3. Comparison between a native system with no persistence support (Native) and log-based persistent memory (Persistent Memory). [20]

There is a plenty of work looking at this direction. Researches have tried to tackle this problem with software solutions. They require either logging/coyp-on-write (CoW) support or rigorous write-order control. Logging/CoW are used to maintain the multi-version data, in order to meet the atomicity requirement for the persistent memory. It updates the data in an explicitly allocated log, instead of overwriting. Therefore, if the system crash, the original data is available to recovery. For instance, Mnemosyne [24] and NV-heaps [25] applies durable software transactional memory to support persistence for in-memory data objects with logging. Similarly, Venkataraman *et al.* [26] uses CoW to support multi-version in order to provide atomicity property. However, logging or CoW doubles the memory traffic with extra data movement and increases the demand of storage. For example, a persistent memory implementation using off-chip NVM and write-ahead logging incurs a 120% increase in memory traffic and only achieves 50% of native system throughput, as it is shown in Fig. 3. Besides atomicity, other persistent memory work also study providing consistency. They require rigorous write-order control and ensure consistency by flushing all processor caches at the barrier of each persistent memory update, using flush (`clflush`) and memory fence (`mfence`) instructions. For instance, previous persistent memory studies [25], [24] flush dirty data blocks out of cache between two order-restricted data updates. However, frequent cache flushes and memory fence lead to significantly performance overhead. BPFS [23] adopted an epoch barrier mechanism to minimize the cache flush frequency. However, it sacrifices durability to some degree that leads to potential data loss.

B. Redesign the Persistent Memory with Emerging NVM

Recently, persistent memory design received substantial attention in hardware community [20], [29], [30], [31]. The key idea is to leveraging novolatility for the persistence service in NVM-based memory hierarchy. This paper takes the persistent memory system of Klin [20] for example. In Klin, both NVM-based cache and NVM-based memory are adopted Therefore the multi-version is already a nature feature: the new updated version is in the cache line with a dirty bit, and the old version remains in the memory. The version update relies on the cache write-back. Therefore, the Klin does not need any overhead (like logging or CoW) to maintain multi-version and provide persistent memory's atomicity property. For the consistency, Klin adopts the clean-on-commit, which optimized the cache flushing operations. Clean-on-commit allows the cache controller to issue flushing request without the `clflush` instruction. The cache controller traces the

dirty cache line all the time, and flushes them during the in-flight persistent memory transactions. Therefore, when the transaction commits, there are only a small portion of cache lines that are still waiting to flush.

Table I compares Klin with existing software solutions and that support no persistence (Native System), in terms of memory update mechanisms and support of atomicity and consistency. The Klin is able to achieve the performance very close to the native system while providing the persistency. The performance of Kiln is 1.5× of system with redo logs and 3× of system with undo logs with workloads of small footprints. Then 1.2× and 1.5× with the workload of large footprints. Furthermore, Kiln only leads to 5% extra total NV cache traffic compared with native system.

C. Discussion and Future Work

Still at the early stages of development, persistent memory research has a long way to go. First, the conventional memory controller losses it efficient while supporting persistent memory [29]. For example, existing memory controller optimizations are all focusing on critical read operation. However, in persistent memory, the write operation (data update) turns to be the bottleneck. Also, existing persistent memory only supports applications with persistent requirement. How to share the persistent memory with both applications with and without persistent requirement, while achieves fairness and efficiency? Second, the current persistent memory are only focusing on performance optimization. Energy-aware persistent memory design has not been explore yet. Third, as an emerging research topic, there is still no published benchmark for persistent memory study. The benchmark suite that comprehensively reflects the key metrics of persistent memory design and cover typical cases of persistent memory systems will benefit the future study.

IV. Discussion and Future Work

Besides what are described above, there are a lot of potentials to exploring NVM's nonvolatlity. In this section, we discuss the future research topics on leveraging nonvolatility for NVM architecture design.

A. Exploring the Resistive Cell Feature

Different from charge-based DRAM cell, the emerging NVM technologies share the common of resistive-based memory cell. As one benefit of the nonvolatility, the NVM based main memory do not need refresh operation as it does in DRAM. Also, it is non-destructive read, so that the refresh after read operation is not necessary. Besides the naive power and performance benefit by eliminating refresh operations, we are able to achieve more by re-design the NVM-based main memory while fully exploiting the nonvolatility. First, the local row buffer in DRAM is dedicated for each column for refreshing purpose. In NVM-based main memory, the local row buffer is not even necessary. Accordingly, the globe row buffer's design also need rethinking. Second, the memory controller design also need optimization in this scenario, i.e., no refresh, fewer or no row buffers. Third, the row buffer hit rate is able to be improved. Without the burden to refresh after read, it can eliminate the restriction that only one row in each bank can be opened at any given time, and that all the data in this row must be sensed. Therefore, it is able to collect data from multiple rows to the row buffer for higher row buffer hit rate.

B. Architecting NVM for Normally-Off Applications

Normally-off working mode power gates the idle part for power and/or energy saving. Conventionally, logic circuit are free to power gating but the memory part, in order to keep data retention, can only be set to a low voltage supply, such as drowsy mode [32]. The NVM with nonvolatility is a prefect candidate for the normally-off computation. There are many potential topics. First, by further architecting NVM to lower level memory hierarchy such as register files, it is possible to make the entire processor nonvolatile [33]. Such nonvolatile processors arise design challenges [34], [35] such as whether to use NVM as working devise or for backup only, trade-offs between backup and re-computing, and backup circuit optimization etc. Third, levering the NVM for reconfigurable logic such as FPGA provides another opportunity for normally-off computing. Fourth, if the NVM-based cache is design for normally-off computing, design exploration is required to determine each cache line needs power gating. Also, the cache coherency in this scenario is challenging.

C. Reliability and Security for NVM-based System

Due to the charge storage nature, SRAM and DRAM are susceptible to soft error strikes. As the technology node scales down, their reliability is becoming an increasing challenge in modern computer systems design. Moreover, reducing supply voltage (such as drowsy cache [32]) make the memory cells are more susceptible to particle strikes. As a result, error detection and correction mechanisms, such as multi-bit Error Correction Code (ECC), are require. However, a strong ECC stores extra information to detect and correct error, which induces overhead in area, timing, and power consumption. A few studies have shown that the SERs of emerging NVMs, caused by particle strikes, are several orders lower than that of SRAM or DRAM because the data in an NVM cell is much harder to be changed by particle radiation strikes [36], [37]. Evaluating and exploring the advantage of emerging NVM's immunity to soft error strikes over traditional SRAM/DRAM at architecture level is interesting.

Apart from the advantage of the NVM's nonvolatility, we should also take care of its downside. The nonvolitility arise the new challenge for security. For instance, how to provide data security when the memory is physically stolen, considering the nonvolatility [38]. Another example, how to protect data under magnetic attack for STT-RAM [39].

V. Conclusion

For decades, the emerging NVM has been studied to replace SRAM and DRAM in the computer memory hierarchy. However, existing effort narrowed the topic to exploring the low power and high density benefit while overcome the expensive write operation and limited lifetime challenges. This paper justifies that the basic feature of the emerging NVM, i.e., the nonvolatility itself, is able to be beneficial, which opens up a new research direction for NVM. The local checkpointing and

persistent memory design are described as examples that utilizing nonvolatility to earn performance/energy improvement. We believe that in the future, there will be more interesting topics that take use of the nonvolitility, and promote the development of emerging NVM.

REFERENCES

[1] "International technology roadmap for semiconductors (itrs)," 2013. [Online]. Available: http://www.itrs.net/ITRS%201999-2014%20Mtgs,%20Presentations%20&%20Links/2013ITRS/Summary2013.htm

[2] H. P. Wong, S. Raoux *et al.*, "Phase Change Memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201–2227, 2010.

[3] B. J.-g. Zhu, "Magnetoresistive Random Access Memory: The Path to Competitiveness and Scalability," *Proceedings of the IEEE*, vol. 96, no. 11, pp. 1786–1798, Nov. 2008.

[4] H. S. P. Wong, H.-Y. Lee *et al.*, "Metal Oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[5] J. J. Yang, D. B. Strukov *et al.*, "Memristive devices for computing." *Nature nanotechnology*, vol. 8, no. 1, pp. 13–24, Jan. 2013.

[6] H.-C. Yu, K.-C. Lin *et al.*, "Cycling endurance optimization scheme for 1Mb STT-MRAM in 40nm technology," in *ISSCC*, Feb 2013, pp. 224–225.

[7] T. yi Liu, T. H. Yan *et al.*, "A 130.7mm2 2-layer 32Gb ReRAM memory device in 24nm technology," in *ISSCC*, Feb 2013, pp. 210–211.

[8] "Intel and micron have new class of non-volatile memory that is 1000 times faster and 10 times denser than nand flash memory," July 2015. [Online]. Available: http://nextbigfuture.com/2015/07/intel-and-micron-have-new-class-of-non.html

[9] Q. Zou, T. Zhang *et al.*, "Thermomechanical stress-aware management for 3d ic designs," in *DATE*, March 2013, pp. 1255–1258.

[10] Y. Xie, "Modeling, architecture, and applications for emerging memory technologies," *Design Test of Computers, IEEE*, vol. 28, no. 1, pp. 44–51, 2011.

[11] G. Sun, X. Dong, and other, "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," in *HPCA*, 2009, pp. 239–249.

[12] M. K. Qureshi, V. Srinivasan *et al.*, "Scalable high performance main memory system using phase-change memory technology," in *ISCA*, 2009, pp. 24–33.

[13] X. Dong, X. Wu *et al.*, "Circuit and microarchitecture evaluation of 3d stacking magnetic RAM (MRAM) as a universal memory replacement," in *DAC*. ACM, 2008, pp. 554–559.

[14] C. Xu, D. Niu *et al.*, "Overcoming the challenges of crossbar resistive memory architectures," in *HPCA*, Feb 2015, pp. 476–488.

[15] P. Zhou, B. Zhao *et al.*, "A durable and energy efficient main memory using phase change memory technology," in *ISCA*, 2009, pp. 14–23.

[16] J. Yue and Y. Zhu, "Accelerating write by exploiting PCM asymmetries," in *HPCA*, 2013, pp. 282–293.

[17] Y. Du, M. Zhou *et al.*, "Bit mapping for balanced PCM cell programming," in *ISCA*, 2013, pp. 428–439.

[18] A. Jog, A. K. Mishra *et al.*, "Cache revive: Architecting volatile stt-ram caches for enhanced performance in cmps," in *DAC*. ACM, 2012, pp. 243–252.

[19] P. Chi, C. Xu *et al.*, "Using multi-level cell STT-RAM for fast and energy-efficient local checkpointing," in *ICCAD*, Nov 2014, pp. 301–308.

[20] J. Zhao, S. Li *et al.*, "Kiln: Closing the performance gap between systems with and without persistence support," in *Micro*. ACM, 2013, pp. 421–432.

[21] S. Amarasinghe, D. Campbell, W. Carlson, A. Chien, W. Dally, E. Elnohazy, M. Hall, R. Harrison, W. Harrod, K. Hill *et al.*, "Exascale software study: Software challenges in extreme scale systems," 2009.

[22] R. Oldfield, S. Arunagiri *et al.*, "Modeling the impact of checkpoints on next-generation systems," in *Mass Storage Systems and Technologies, IEEE Conference on*, Sept 2007, pp. 30–46.

[23] J. Condit, E. B. Nightingale *et al.*, "Better I/O through byte-addressable, persistent memory," in *SOSP*. ACM, 2009, pp. 133–146.

[24] H. Volos, A. J. Tack, and M. M. Swift, "Mnemosyne: Lightweight persistent memory," in *ASPLOS*. ACM, 2011, pp. 91–104.

[25] J. Coburn, A. M. Caulfield *et al.*, "NV-heaps: making persistent objects fast and safe with next-generation, non-volatile memories," in *ASPLOS*, 2011, pp. 105–118.

[26] S. Venkataraman, N. Tolia *et al.*, "Consistent and durable data structures for non-volatile byte-addressable memory," in *Proceedings of the 9th USENIX Conference on File and Storage Technologies*, 2011, pp. 1–15.

[27] E. Lee, H. Bahn *et al.*, "Unioning of the buffer cache and journaling layers with non-volatile memory," in *Proceedings of the USENIX Conference on File and Storage Technologies*, 2013, pp. 73–80.

[28] M. Wu and W. Zwaenepoel, "eNVy: A non-volatile, main memory storage system," in *ASPLOS*, 1994, pp. 86–97.

[29] J. Zhao, O. Mutlu, and Y. Xie, "FIRM: Fair and high-performance memory control for persistent memory systems," in *MICRO*. IEEE, 2014, pp. 153–165.

[30] S. Pelley, P. M. Chen *et al.*, "Memory persistency," in *ISCA*, 2014, pp. 1–12.

[31] Y. Lu, J. Shu *et al.*, "Loose-ordering consistency for persistent memory," in *ICCD*, Oct 2014, pp. 216–223.

[32] K. Flautner, N. S. Kim *et al.*, "Drowsy caches: simple techniques for reducing leakage power," in *ISCA*, 2002, pp. 148–157.

[33] Y. Wang, Y. Liu *et al.*, "A 3us wake-up time nonvolatile processor based on ferroelectric flip-flops," in *ESSCIRC*, Sept 2012, pp. 149–152.

[34] Y. Liu, Z. Li *et al.*, "Ambient energy harvesting nonvolatile processors: From circuit to system," in *DAC*. ACM, 2015, pp. 150:1–150:6.

[35] K. Ma, Y. Zheng *et al.*, "Architecture exploration for ambient energy harvesting nonvolatile processors," in *HPCA*, Feb 2015, pp. 526–537.

[36] D. Mahalanabis, H. Barnaby *et al.*, "Investigation of single event induced soft errors in programmable metallization cell memory," *Nuclear Science, IEEE Transactions on*, vol. 61, no. 6, pp. 3557–3563, Dec 2014.

[37] W. Bennett, N. Hooten *et al.*, "Single- and Multiple-Event Induced Upsets in HfO2Hf 1T1R RRAM," *Nuclear Science, IEEE Transactions on*, vol. 61, no. 4, pp. 1717–1725, Aug 2014.

[38] V. Young, P. J. Nair *et al.*, "Deuce: Write-efficient encryption for non-volatile memories," in *ASPLOS*. ACM, 2015, pp. 33–44.

[39] J.-W. Jang, J. Park *et al.*, "Self-correcting STTRAM under magnetic field attacks," in *DAC*, June 2015, pp. 1–6.