Runtime Solutions to Apply Non-volatile Memories in Future Computer Systems

Hoda Aghaei Khouzani

Introduction

Memory system is an indispensable part of any computer system from large supercomputers to embedded devices. Its performance, energy, capacity, cost, and management algorithms must scale as computer systems scale to ensure the desired performance growth and the possibility of running new applications. Unfortunately, observable trends of current memory technologies from on-chip Static-RAM (SRAM) caches to Dynamic-RAM (DRAM) main memory and Hard Disk Drive (HDD) storage are failing to respond to these existing and emerging requirements, which lead research groups in both academia and industry to look for alternative solutions. A number of promising non-volatile memory (NVM) technologies are introduced to replace the current devices on various levels of the memory hierarchy. Due to the extent of memory hierarchy subject and the diversity of NVM technologies, the main focus of this study is main memory. Specifically, it explores the feasibility of applying two of the promising NVMs, namely, phase change memory (PCM) and domain wall memory (DWM).

Challenges of Current Main Memories

Main memories are facing challenges from three distinctive perspectives of system, applications, and technology. Today's systems tend to let more heterogeneous processing cores share the memory system [1,2], which results in rising demand for more capacity, bandwidth and quality of service (QoS) from the memory system. This in turn makes the well-known problem of memory wall[1] more critical. More importantly, energy and power consumption have become key system design limiting factors [3]. Some studies show DRAM, the mainstream main memory technology, consumes up to 40% of system energy due to the need of consistent refresh cycles [4].

Applications, on the other hand, are becoming very data intensive and require the manipulation of great amount of data. For instance, scientific research increasingly relies on computing over large data sets [5]. Meanwhile, the existence of sensors, networks, and storage technology makes it possible to collect and store wide range of information for processing. Creation of new application capable of processing this vast amount of collected data depends on how much the memory system scale and how well its management unit is designed.

From the technology perspective, DRAM is predicted to face the end of its reign in near future. According to ITRS, existing DRAMs suffer from limited scalability below 20nm due to the constraints of leakage current and capacitor placement [6]. Furthermore, the smaller size of transistors increases the frequency of hard and soft errors [7], which reduces the reliability of system.

Emergence of New Memory Technologies

While DRAM fails to fulfill requirements of system and applications, many new memory technologies are introduced. The major advantage of these technologies is their non-volatility nature which eliminates the need for power hungry refresh cycles, and results in near zero standby power. Among these technologies, Phase Change Memory (PCM) and Domain Wall Memory (DWM) show better scalability than DRAM, which makes them promising candidates for implementing next generation main memory. Both PCM and DWM has comparable read latency and energy to DRAM, while DWM additionally has comparable write latency and energy. Yet, neither of them are perfect.

PCM's applicability as main memory is challenged by its limited write endurance, long write latency, and high write energy. Without addressing three problems, a PCM-based memory system either fails to work within the expected performance and energy budget of future main memory, or dies in matter of days [8, 9, 10]. Although many research and development groups target these limitations through improving the physical attributes of PCM, there is still a need for architectural-level solutions and runtime supports.

Unlike PCM, DWM's write attributes are comparable to DRAM [11], which makes it even more attractive. The main challenge to employ DWM is due to its sequential access structure. Specifically, to read/write data from/to DWM, it is necessary to shift the track containing the data to align it with an access port. As a result, the data access latency and energy of DWM are considerably affected by the shift operations.

Proposed Techniques

The goal of this study is to explore the applicability of PCM and DWM as main memory. Compared to DWM, PCM is a well-studied device. So, this proposal aims at providing more effective solutions on top of previous works to address its write limitation. On the other hand, DWM is rather new and less explored. Here, the main focus is the access latency of metadata, in particular page table, which is on the critical path of memory access latency.

I. Prolonging PCM Limited Lifetime

Wear-leveling is a technique aiming at evenly spreading write operations over the PCM at various granularities in order to prevent some cells from being worn out quickly [9, 12, 13, 14].

[1] The increasing gap between processor and memory speed is known as memory wall, which means the performance is going to be limited by the time it takes to get data from main memory into the CPU.

While these techniques can effectively extend PCM lifetime, a lot of extra write operations need to be performed to periodically swap the mappings of hot (i.e., frequently written) and cold (i.e., rarely written) data. To achieve a more balanced write distribution, more extra writes need to be performed, and a larger performance and energy overhead will be induced.

As each program has its own hot/cold pages, most wear-leveling techniques balance the write operations to different pages during the execution of each program. However, the observation here is that this is unnecessary. Since endurance is a lifetime factor, it is unnecessary to fully balance the number of writes to each page during the execution of each program. Instead, the imbalanced access characteristics of different programs can be utilized. The hot virtual pages of different processes can be mapped to different physical page frames in the PCM, thus balancing write counts to different PCM pages without inducing extra write operations.

Balancing writes across different processes requires the operating system (OS) to predict, upon allocating a virtual page, whether it will be hot or cold. This can be achieved by exploiting segment information. Different segments have diverse write characteristics. The text segment is usually read-only, while the data and stack segments are write-intensive and their accesses display high spatial and temporal localities. Leveraging this property, I proposes a low-cost, proactive wear leveling technique. More specifically, a segment-based and age-aware page allocation strategy is developed, enabling the OS to allocate page frames of different ages to different segments. This is further enhanced by a wear-resistant page replacement procedure and a wear out prevention procedure, both of which deallocate page frames based on their age information. These three procedures are compatible with existing virtual memory management, with minimum software and hardware modifications needed.

To evaluate the proposed scheme, I developed a memory management system and incorporated the proposed wear-leveling procedures in it. I also implemented three representative wear leveling approaches as the comparison set, segment swapping [8], random swapping [12], and start-gap [13]. Twelve benchmarks that produce a high volume of memory writes are selected from the SPEC 2000 and 2006 suites. Their memory access traces are collected with Pin. Main memory is 128 MB with 10^6 wear out limit per cell.

The normalized PCM lifetime for benchmarks is shown in Figure 1. All the wear-leveling techniques are able to prolong PCM lifetime to some extent. On average, segment swapping, start-gap, random swapping and the proposed scheme are able to utilize 2%, 38%, 89% and 98% of the potential writes, respectively.

Table 1 shows the average overhead in terms of write volume. The overhead of no-WL is caused by page faults, while for the four WL approaches, both page faults and remapping operations are considered.

Figure 1 Normalized Lifetime

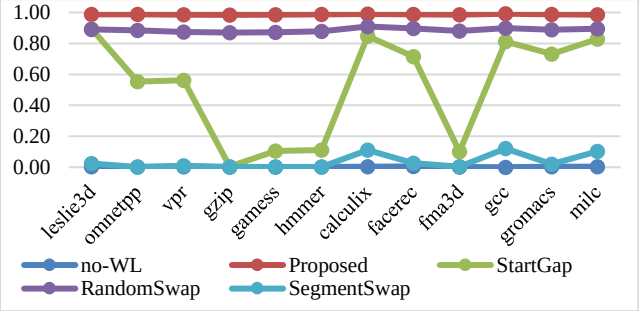


Table 1. The overhead

no-WL	Proposed	StartGap	Random Swap	Segment Swap
6.75%	7.25%	146%	67.25%	36.5%

II. Overcoming PCM Long Write Latency and High Write Energy

The high write energy and long write latency of PCM challenge its complete replacement to DRAM as main memory. To hide these shortcomings of PCM, one attractive solution is a DRAM-PCM hybrid architecture, which offers the merit of combining the advantages of DRAM and PCM and hiding their shortcomings. Specifically, as write operations are critical to the performance, energy, and endurance of PCM [15, 8], it is preferable to place pages with more read operations in PCM while storing write-intensive pages in DRAM [4]. Usually two types of hybrid architectures can be adopted. The first one organizes DRAM and PCM in parallel and places a page exclusively either in PCM or in DRAM [16, 17, 18]. Since in these architectures the size of DRAM is in the same order as the size of PCM, the scalability limitation and the high standby power of DRAM are still dominant. The alternative is to place DRAM and PCM hierarchically and uses DRAM as a small cache for PCM [19, 20, 12]. These architectures are more efficient, since a DRAM of 3% total memory size is about to deliver similar performance as a DRAM-only system and absorb most of memory writes [9], thus largely reducing the energy-expensive and lifetime-hurting PCM writes. However, a small DRAM size results in a relatively higher DRAM miss rate, which in turn hurts the performance and endurance of the hybrid memory since upon a DRAM miss, PCM is accessed, and pages need to be migrated between the two devices.

While most of existing optimizations on DRAM-PCM hybrid memory focus on improving PCM endurance by reducing the number of PCM writes in PCM [12], this study improves the performance, energy and the endurance of the hybrid memory at the same time, through directly reducing the number of DRAM misses and the resultant writebacks to PCM. However, to efficiently achieve this goal, straightforward solutions such as enlarging the DRAM size or increasing the way-

associativity of the DRAM cache are not desirable. The former increases the standby power and the footprint of the main memory, while the latter complicates the process of tag matching and hence increases the latency and energy of each DRAM access.

To reduce DRAM misses without imposing too much overhead or sacrificing memory performance, a proactive solution is proposed that exploits the flexibility of mapping virtual pages to physical pages. Specifically, the page allocation process is tuned to consider segment and conflict information. The proposed technique leverages program segment information to differentiate write-intensive and read-only pages. By placing read-only pages in PCM and furthermore bypassing DRAM when accessing those pages, capacity misses in DRAM can be reduced. Moreover, as memory accesses of most programs are highly clustered, a high variation in the number of conflict misses across different DRAM sets is expected. This study develops a novel hierarchical clock algorithm, capable of allocating pages to less conflicting DRAM sets, thus effectively reducing the conflict misses in DRAM.

A memory management system is developed to simulate the proposed DRAM-PCM hybrid architecture and proactive page allocation algorithm. The sizes of PCM and DRAM are set to 64MB and 2MB (DRAM is 3% of PCM), respectively. DRAM is 4-way associative. The evaluation set is composed of 8 representative benchmarks from the SPEC2000&2006 suites. These benchmarks are selected because they produce a high volume of memory write operations and a varying number of DRAM misses. The hybrid DRAM-PCM structure proposed in [9] is considered as the baseline architecture during our study. To show the impact of different replacement algorithms, both the LRU (least-recently-used) and the CF (clean-first) [12] algorithm have been implemented.

As conflict misses are more evenly distributed across different sets, the total number of DRAM misses can be reduced. Figure 2 shows the miss reduction achieved by the proposed technique when it is applied to different replacement algorithms. On average, the proposed technique reduces the number of DRAM misses by 27% for LRU, and by 29% for the CF replacement algorithm. More reduction is achieved for the latter case because the algorithm originally causes more conflict misses than LRU, thus providing more potential for reduction.

Reducing writes in PCM is important for prolonging PCM lifetime. Figure 3 shows the reduction in the number of PCM writes achieved by the proposed method. When LRU is used, the proposed scheme can reduce 25% of total PCM writes on average. More specifically, by comparing Fig. 3 to Fig. 2, it can be seen that when LRU is used, the reduction in PCM writes has the same pattern as the reduction in DRAM misses across different benchmarks, indicating strongly correlation between the two values.

Figure 2. DRAM Miss Reduction

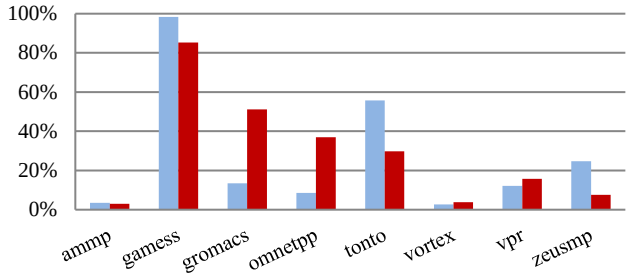
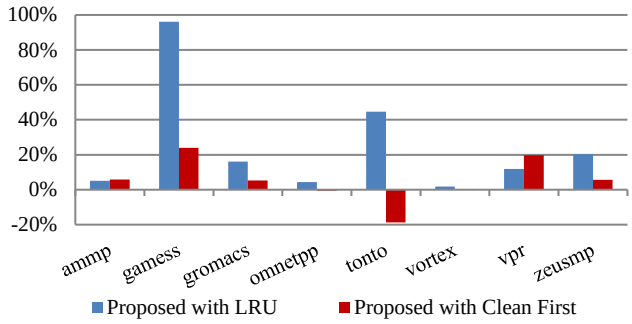


Figure 3. PCM Write Reduction



III. Reducing DWM Access Latency

This study also focuses on DWM. As mentioned before, the main challenge to apply DWM is its sequential access structure, which requires a costly shift operation to access data of interest. Previous research has proposed different ways to alleviate the impact of shift operations on system metrics [21, 22, 23, 24]. However, no work has addressed the performance-critical page table accesses in DWM-based main memory.

Previously when the application footprint was small, accesses to the page table were limited since more than 99% of address translations are handled by the translation lookaside buffer (TLB) [25]. However, recent applications tend to access more page table entries (PTEs), thus engendering more TLB misses. Meanwhile, the TLB miss penalty also increases as multiple memory accesses are required to travel through the multi-level page table. For example, in a 64-bit architecture such as x86_64 and modern ARM, at least four memory accesses are needed to handle a TLB miss. The latency of virtual to physical address translation can reach to 50% of the total execution time [26]. Since page table is on the critical path of data accesses, its access latency is crucial to the overall system performance. Such latency furthermore affects context switching, since upon each context switch the entire TLB has to be flushed back to the page table. This effect becomes more significant as users expect to run more and more applications on their devices simultaneously.

This study reduces the latency of page table accesses in a DWM-based main memory. The observation is that not all the fields in a PTE are required per each access. Instead, based on the current state of a PTE, one can determine the upcoming access as well as the exact fields to be accessed, and pre-shift

the tracks to align those bits with the access port. This approach accelerates page table accesses from two aspects. First, the alignment shifts are eliminated from the critical path. Second, without reading the PTE, its state is known a priori since the position of the access port can be used to interpret the state of PTE to reading the data. Also a new approach to place page table pages in DWM main memory is proposed. To minimize the potential conflicts between the PTEs that share the same track, only PTEs with non-overlapping access time are placed on the same track.

A memory management system is developed to simulate the proposed DWM-based main memory and page table organization. The memory size is 4GiB, while the upper limit of page table size within the memory is set to 2MiB. The evaluation set is composed of 24 SPEC 2006 benchmarks, which are relatively memory-intensive and engender non-trivial amount of TLB misses. The page table access trace of each benchmark is collected with Pin. Every four benchmarks are grouped together to simulate a multi-process environment. A round robin policy is used to simulate OS context switches at the frequency of once per million cycles. Table III lists benchmarks in each group.

Figure 4 shows the total number of shift operations that affect the latency of accessing the page table. The lower bound only includes necessary shifts and fully eliminates alignment shifts, extra shifts, as well as inter-PTE conflicts. The overall reduction in shift operations in 69%, whose difference to the theoretical lower bound is negligible. The number of shifts that affect the energy consumed in page table accesses are shown in Figure 5. Compared to latency reduction, the amount of energy reduction is less because pre-alignment does not completely eliminate alignment shifts, but instead performs them prior to the access. Overall, the average energy reduction achieved by proposed technique is 15%.The shift operations performed to flush the TLB during context switches are presented in Figure 9. These shifts constrain both the latency and energy consumption of context switches.

Figure 4. Shifts that affect latency of address translation

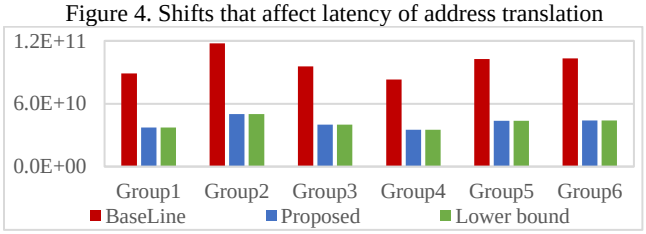


Figure 5. Shifts that affect energy of address translation

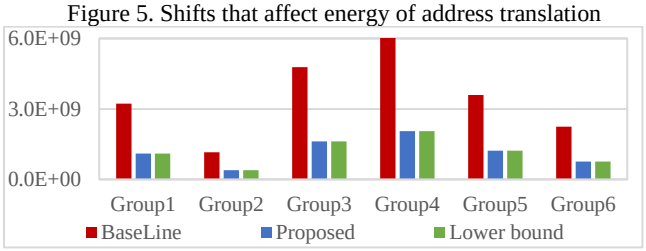
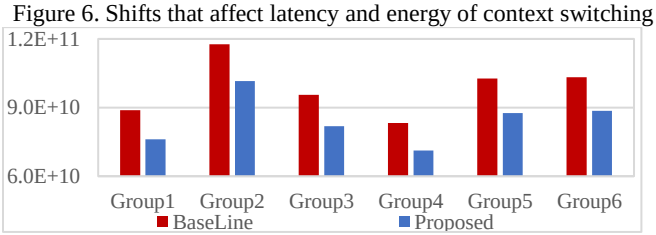


Figure 6. Shifts that affect latency and energy of context switching



References

[1] E. S. Chung, et.al, “Single-chip heterogeneous computing: Does the future include custom logic, FPGAs, and GPUs?” in MICRO, 2010, pp. 225–236.
[2] R. Ausavarungnirun, et. al, “Staged memory scheduling: achieving high performance and scalability in heterogeneous systems,” in ISCA, 2012, pp. 416–427.
[3] O. Mutlu, “Memory scaling: A systems architecture perspective,” in IEEE Intl. Memory Workshop, 2013, pp. 21–25.
[4] A. N. Udipi, “Rethinking DRAM Design and Organization for Energy-Constrained Multi-Cores,” in ISCA, 2010, pp. 175–186.
[5] A. J. G. Hey and A. E. Trefethen, “The Data Deluge: An e-Science Perspective,” 2003, chapter: 36.
[6] ITRS, “International Technology Roadmap for Semiconductors,” http://www.itrs2.net/2013-itrs.html, Emerging Research Devices (ERD).
7] J. H. Ahn, et. al, “Future Scaling of Processor-Memory Interfaces,” in SC, 2009, pp. 1–12.
[8] P. Zhou, et. al, “A Durable and Energy Efficient Main Memory Using Phase Change Memory Technology,” in ISCA, 2009, pp. 14–23.
[9] M. K. Qureshi, et al, “Scalable High Performance Main Memory System Using Phase-Change Memory Technology,” in ISCA, 2009, pp. 24–33.
[10] S. Cho and H. Lee, “Flip-N-Write: A Simple Deterministic Technique to Improve PRAM Write Performance, Energy and Endurance,” in MICRO, 2009, pp. 347–357.
[11] M. H. Kryder and C. S. Kim, “After Hard DrivesWhat Comes Next?” IEEE Trans. On Magnetics, vol. 45, pp. 3406–3413, Oct. 2009.
[12] A. Ferreira, et. al, “Increasing PCM main memory lifetime,” in DATE, 2010, pp. 914–919.
[13] M. Qureshi, et. al, “Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling,” in MICRO, 2009, pp. 14–23.
[14] N. Seong, et. al, “Security refresh: prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping,” in ISCA, 2010, pp. 383–394.
[15] S. Raoux, et. al, “Phase-change random access memory: A scalable technology,” IBM Journal of Research and Development, vol. 52, Jul. 2008.
[16] G. Dhiman, et. al, “PDRAM: A Hybrid PRAM and DRAM Main Memory System,” in DAC, 2009, pp. 664–669.
[17] Y. Park, et al, “Power-Aware Memory Management for Hybrid Main Memory,” 2011, pp. 82– 85.
[18] S. Lee, et al, “CLOCK-DWF: A Write-History-Aware Page Replacement Algorithm for Hybrid PCM and DRAM Memory Architectures,” IEEE Trans. Comput., vol. 63, pp. 2187–2200, 2014.
[19] H. G. Lee, et al, “An Energy- and Performance-Aware DRAM Cache Architecture for Hybrid DRAM/PCM Main Memory Systems,” in ICCD, 2011, pp. 381–387.
[20] T. J. Ham, et. al, “Disintegrated Control for Energy-Efficient and Heterogeneous Memory Systems,” in HPCA, 2013, pp. 424–435.
[21] Q. Hu, et al., “Exploring Main Memory Design Based on Racetrack Memory Technology,” in GLVLSI, 2016, pp. 397–402.
[22] R. Venkatesan, et. al, “TapeCache: A High Density, Energy Efficient Cache Based on Domain Wall Memory,” in ISLPED, 2012.
[23] S. Motaman, et. al, “Synergistic Circuit and System Design for Energy-efficient and Robust Domain Wall Caches,” in ISLPED, 2014.
[24] M. Mao, et. al, “Exploration of GPGPU Register File Architecture Using Domain-wall-shiftwrite based Racetrack Memory,” in DAC, 2014.
[25] D. A. Patterson and J. L. Hennessy, Computer Organization And Design. Hardware/Software Interface. Morgan Kaufmann Publishers, 2009.
[26] V. Karakostas, et. al, “Redundant Memory Mappings for Fast Access to Large Memories,” in ISCA, 2015, pp. 66–78.