Improving Phase Change Memory Performance with Data Content Aware Access

Shihao Song
Drexel University
USA

Anup Das
Drexel University
USA

Onur Mutlu
ETH Zürich
Switzerland

Nagarajan Kandasamy
Drexel University
USA

Abstract

Phase change memory (PCM) is a scalable non-volatile memory technology that has low access latency (like DRAM) and high capacity (like Flash). Writing to PCM incurs significantly higher *latency* and *energy* penalties compared to reading its content. A prominent characteristic of PCM's write operation is that its latency and energy are sensitive to the data to be written as well as the content that is *overwritten*. We observe that overwriting *unknown* memory content can incur significantly higher latency and energy compared to overwriting *known* all-zeros or all-ones content. This is because all-zeros or all-ones content is overwritten by programming the PCM cells *only* in one direction, i.e., using either SET or RESET operations, not both.

In this paper, we propose *data content aware PCM writes* (**DATACON**), a new mechanism that reduces the latency and energy of PCM writes by redirecting these requests to overwrite memory locations containing all-zeros or all-ones. DATACON operates in three steps. First, it estimates how much a PCM write access would benefit from overwriting known content (e.g., all-zeros, or all-ones) by comprehensively considering the number of set bits in the data to be written, and the energy-latency trade-offs for SET and RESET operations in PCM. Second, it translates the write address to a physical address within memory that contains the best type of content to overwrite, and records this translation in a table for future accesses. We exploit data access locality in workloads to minimize the address translation overhead. Third, it re-initializes *unused* memory locations with known all-zeros or all-ones content in a manner that does not interfere with regular read and write accesses. DATACON overwrites unknown content only when it is absolutely necessary to do so. We evaluate DATACON with workloads from state-of-the-art machine learning applications, SPEC CPU2017, and NAS Parallel Benchmarks. Results demonstrate that DATACON improves the effective access latency by 31%, overall system performance by 27%, and total memory system energy consumption by 43% compared to the best of performance-oriented state-of-the-art techniques.

CCS Concepts: • **Hardware → Memory and dense storage**; • **Software and its engineering → Main memory**.

Keywords: phase change memory (PCM), performance, energy, hybrid memory, DRAM, non-volatile memory (NVM)

ACM Reference Format:
Shihao Song, Anup Das, Onur Mutlu, and Nagarajan Kandasamy. 2020. Improving Phase Change Memory Performance with Data Content Aware Access. In *Proceedings of the 2020 ACM SIGPLAN International Symposium on Memory Management (ISMM '20), June 16, 2020, London, UK*. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3381898.3397210

ISMM '20, June 16, 2020, London, UK

ACM ISBN 978-1-4503-7566-5/20/06…$15.00
https://doi.org/10.1145/3381898.3397210

1 Introduction

DRAM has been the technology choice for implementing main memory due to its relatively low latency and low cost. However, DRAM is a fundamental performance and energy bottleneck in almost all computing systems [89, 95, 98, 131], and it is experiencing significant technology scaling challenges [41, 62, 67, 95–97]. Recently, DRAM-compatible, yet more technology scalable alternative memory technologies are being explored: phase-change memory (PCM) [7, 12, 23, 37, 40, 56, 57, 63, 76–78, 83, 106, 107, 109, 116, 127, 136, 142, 143, 145, 147–149], spin-transfer torque magnetic RAM (STT-MRAM) [70], metal-oxide resistive RAM (ReRAM) [5, 88, 133], conductive bridging RAM (CBRAM) [71], ferroelectric RAM (FeRAM) [18], etc. Of the various emerging memory technologies, PCM is the most mature [73, 132] due to its CMOS compatibility [59], deep material understanding [134], and availability of high-capacity working prototypes [4, 53, 61, 80, 100, 125].

However, PCM has finite write endurance, and its access latency and energy are higher than those of DRAM [29, 30,

58, 60, 77, 103, 107, 118]. One key characteristic of PCM is the *asymmetry* in read and write latencies [77, 86, 141].

This is because the SET operation (0 → 1 programming) in PCM requires longer latency than the RESET operation (1 → 0 programming). Due to memory bank interleaving, multiple requests can be served in parallel using different PCM banks [86]. Unfortunately, when two requests go to the *same bank*, they have to be served *serially*. This is known as *bank conflict*. Slow writes in PCM increase bank conflict latencies, degrading system performance [121].

Modern computing systems are embracing hybrid memories comprising of DRAM and PCM to combine the best properties of both memory technologies, achieving low latency, high reliability, and high density. 3D XPoint memory is one example of hybrid memory with DRAM and PCM connected to separate DIMMs [21]. The IBM POWER 9 processor [112] is another example, where embedded DRAM (eDRAM) is used as a write cache to PCM main memory. Many recent works mitigate the impact of slow and costly PCM writes in hybrid memories by migrating write-intensive pages from PCM to DRAM [13–17, 35, 45, 48, 52, 55, 81, 83, 91, 92, 107, 111, 140, 146]. Unfortunately, *none* of these approaches *fundamentally* reduce write latency at its source and PCM latency continues to be a performance bottleneck, especially for emerging data-intensive applications, where high volumes of data are accessed frequently.

One simple approach to reducing PCM latency is to use *only* RESET operations during all PCM writes [75, 110]. These schemes require the PCM cells in a memory location to be SET, prior to a write. Unfortunately, if a PCM write induces many RESET operations, the energy consumption of such schemes can be higher than simply overwriting the existing memory content as in a baseline system [77]. This is because a RESET operation in PCM requires higher energy than a SET operation. We observe that for some workloads, the energy overhead of such a RESET-only scheme can be up to 30% over Baseline (Section 6.1). Recent works also propose to reduce the energy overhead of PCM write operations by selectively programming the PCM cells [31, 33, 39, 54, 93, 137]. This is achieved by analyzing the data to be written (referred to as **write data**) and the content of the memory location that is overwritten (referred to as **overwritten content**). However, such schemes increase the effective write latency by over 15% due to the additional read request required to find out the overwritten content (Section 6.1).

Our goal is to design a mechanism to reduce the latency and energy of PCM writes. We achieve this goal by exploiting the following three major observations in this paper.

Observation 1: *We observe that the energy consumed to serve a PCM write depends on both the write data and the overwritten content.* Figure 1 plots the energy consumption of a PCM write as the fraction of SET bits in the write data is increased from 0% to 100% with the overwritten content (i.e., the initial content) set to all-0s and all-1s (see Section 5 for our simulation parameters). We observe that if the fraction of SET bits in the write data is less than 60%, overwriting all-0s is more energy efficient than overwriting all-1s. Conversely, if the fraction of SET bits in the write data is more than 60%, overwriting all-1s is more energy efficient. Therefore, *PreSET* [110], a mechanism that initializes a memory location to all-1s before overwriting it, is beneficial only in the latter scenario, where the fraction of SET bits in the write data is more than 60%.

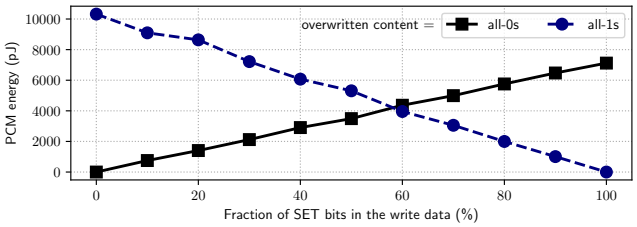

Figure 1. Energy of a PCM write for increasing fraction of SET bits in the write data when the overwritten content is all-0s and all-1s.

Observation 2: *We observe that the number of SET bits in the write data is workload dependent.* Figure 2 plots the total number of PCM writes distributed into write accesses having more than 60% and less than 60% SET bits in their write data for each of the evaluated workloads (Section 5). We observe that on average, only 33% of PCM writes in these workloads have more than 60% SET bits in their write data. Therefore, PreSET can bring energy benefit only for a small fraction of PCM write accesses, leaving behind a significant opportunity for energy improvement. Exploiting this opportunity will not only reduce energy but also improve performance by enabling more concurrency.

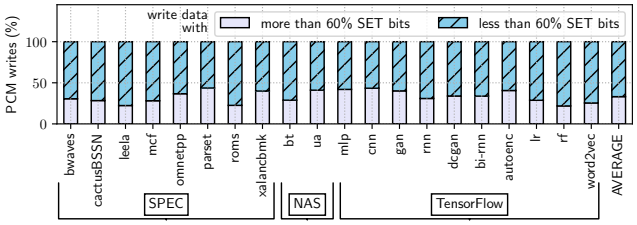

Figure 2. Total number of PCM writes distributed into write accesses having more than 60% and less than 60% SET bits in their write data for each of the evaluated workloads.

We propose *data content aware PCM writes* (**DATACON**), a new mechanism to reduce the latency and energy of PCM writes by exploiting these two observations. The **key idea** is to estimate how much a PCM write access would benefit from overwriting *known* all-0s or all-1s content by comprehensively considering the fraction of SET bits in the write data and the energy-latency trade-offs for SET and RESET operations in PCM. Based on this estimate, DATACON selects a memory location to redirect the write request to, containing the *best* overwritten content.

The operation of DATACON is straightforward. When a cache block is evicted from DRAM, the memory controller redirects the write request to a new physical address within memory, which contains the best overwritten content for the write data. When overwriting this content, the latency and energy of the write operation reduces due to programming the PCM cells *only* in one direction, i.e., using only SET or RESET operations, not both. This improves system performance. The memory controller maintains a table to record all such address translations. As memory locations are continuously re-mapped, DATACON periodically invalidates entries from the table and re-initializes *unused* memory locations with all-zeros or all-ones so that future write requests can be serviced using them. Section 4 describes the implementation of DATACON. To limit the size of DATACON's address translation table, we exploit the following observation.

Observation 3: We observe that a modern PCM bank is implemented as a collection of *partitions* that operate mostly *independently* while *sharing* a few global peripheral structures, which include the sense amplifiers (to read) and the write drivers (to write) [121]. Many workloads tend to frequently access rows that belong to the *same* memory partition in a bank. We refer to this form of spatial locality where rows from the same partition are read from and written to frequently as *Partition Level Spatial Locality (PLSL)*.

DATACON exploits the partition-level spatial locality in a workload to *cache* only a part of the address translation table inside the memory controller, while the full address translation table is stored in memory.

We evaluate DATACON with workloads from state-of-the-art machine learning applications [1], SPEC CPU2017 [22], and NAS Parallel Benchmarks [10]. Our results show that DATACON improves performance and energy efficiency over state-of-the-art techniques that reduce latency and energy of PCM writes. For multi-core workloads, DATACON improves the effective access latency by 31%, overall system performance by 27%, and total system energy consumption by 43% with a hardware cost of only 9KB for a PCM memory of 128GB capacity. As DATACON reduces the latency and energy of a PCM write access at its source, DATACON can be combined with other mechanisms that aim to reduce the *number* of write accesses to PCM [13–17, 35, 45, 48, 50–52, 81, 91, 135, 140, 143, 146].

We make the following **contributions**.

- We propose an efficient technique, DATACON, which lowers the PCM write latency and energy by overwriting the *best* content on a write access. DATACON overwrites unknown content only when it is absolutely necessary to do so. DATACON maintains a table for address translations and methodically re-initializes *unused* memory locations to service future write requests.
- We observe that many workloads exhibit a form of locality where rows from the *same* memory partition are frequently read from and written to frequently. We refer to this locality as *Partition Level Spatial Locality (PLSL)*. DATACON exploits this locality to minimize the overhead of its address translation table by *caching* address translations of only the frequently accessed PCM partitions.
- We comprehensively evaluate the performance and energy efficiency of DATACON. Our results show that DATACON significantly improves performance and energy efficiency across a wide variety of workloads.

2 Background

2.1 PCM Operation

Figure 3(a) illustrates how a chalcogenide semiconductor alloy is used to build a PCM cell. The amorphous phase (logic '0') in this alloy has higher resistance than the crystalline phase (logic '1'). $Ge_2Sb_2Te_5$ (GST) is the most commonly used alloy for PCM [11, 24, 72, 101, 115, 128] due to its high amorphous-to-crystalline resistance ratio, fast switching between phases, and high endurance. However, other chalcogenide alloys are also explored due to their better data retention properties [82, 94, 144]. Phase changes in a PCM cell are induced by injecting current into the resistor-chalcogenide junction and heating the chalcogenide alloy.

Figure 3 (b) shows the different current profiles needed to program and read in a PCM device [117]. To RESET a PCM cell, a high power pulse of short duration is applied and quickly terminated. This first raises the temperature of the chalcogenide alloy to 650°C, above its melting point. The melted alloy subsequently cools extremely quickly, locking into an amorphous phase. To SET a PCM cell, the chalcogenide alloy is heated above its crystallization temperature, but below its melting point for a sufficient amount of time. Finally, to read the content (i.e., know the phase) of a PCM cell, a small electrical pulse is applied that is sufficiently low so as not to induce phase change in the PCM cell.

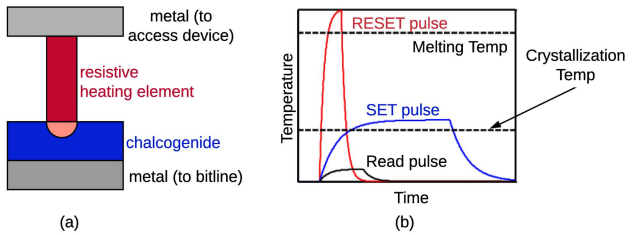

Figure 3. (a) A phase change memory (PCM) cell and (b) current needed to SET, RESET, and read a PCM cell.

2.2 PCM Organization

To ensure reasonable memory read and write latencies at high density, a PCM chip, like the DRAM, is organized hierarchically [124]. For example, a 128GB PCM can have 2 channels, 1 rank/channel and 8 banks/rank. A bank can have 64 partitions [121], which are similar to subarrays in DRAM [27, 66]. Within each partition, the PCM cells are organized as an array. A column of PCM cells is called a

bitline. A row of PCM cells is called a *wordline*. Each PCM bank can have 128 peripheral circuits to read and program 128 bits in the array in parallel.

2.3 PCM Timings

We briefly introduce the commands and timing parameters of PCM to understand DATACON. Without loss of generality, we describe them for the 28nm PCM design from Micron [124]. To serve a memory request that accesses data at a particular row and column address within a bank, a memory controller issues *three* commands to the bank [121].

- `ACTIVATE`: activate the wordline and enable the peripheral circuit for the memory cells to be accessed.
- `READ`/`WRITE`: drive read or write current through the cell. After this command executes, the data stored in the cell is available at the output terminal of peripheral circuit, or the write data is programmed to the cell.
- `PRECHARGE`: deactivate the wordline and bitline, and prepare the bank for the next access.

Figure 4 shows the different memory timing parameters used in DATACON. Table 1 provides the typical values of these timing parameters based on Micron's 28nm PCM design [124].[1] Table 1 also reports the timing parameters used in DATACON to service PCM reads and writes. DATACON uses the SET and RESET timing parameters when overwriting all-0s and all-1s content, respectively. DATACON uses the long-latency baseline write timing parameters only when overwriting unknown content. Our technique improves performance by minimizing the number of times unknown content is overwritten.

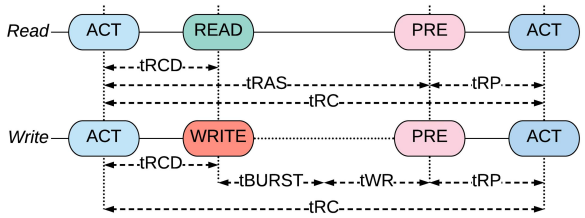

Figure 4. PCM timings for serving read and write requests.

3 Performance and Energy Improvements in DATACON

3.1 Quantifying Performance Improvement

We present some motivating examples to provide intuition for the performance improvement in DATACON. Figure 5 illustrates an 8-bit write operation in PCM. The write data '00100000' is to be programmed to a memory location, whose current content is '11011101' (❶). This content is unknown at the time of the write operation. The following four steps are performed in sequence to service the write request in a Baseline PCM chip [77].

1: The write data is compared with the overwritten content to select the PCM cells that need to be SET. In this example, only the third bit from the left needs to be SET. This is achieved by setting bit 3 in the write driver and masking others (❷).
2: The selected PCM cell is SET using a long-latency SET pulse generated by the write driver (❸). The memory content changes to '11111101'.
3: The write data is compared with the new memory content to select the PCM cells that need to be RESET. This is achieved by programming the corresponding bits in the write driver and masking others (❹).
4: The selected PCM cells are RESET by the write driver using short-latency RESET pulses (❺).

We observe that after executing these four steps the memory content is overwritten with the write data.

Table 1. PCM timing parameters based on [124].

Baseline Timing Parameters					
	tRCD	**tRAS**	**tRP**	**tRC**	
Read	3.75ns	55.25ns	1ns	56.25ns	
	tRCD	**tBURST**	**tWR**	**tRP**	**tRC**
Write	75ns	15ns	190ns	1ns	209.75ns
DATACON Timing Parameters					
	tRCD	**tRAS**	**tRP**	**tRC**	
Read	3.75ns	55.25ns	1ns	56.25ns	
	tRCD	**tBURST**	**tWR**	**tRP**	**tRC**
SET (all-0s)	3.75ns	15ns	150ns	1ns	169.75ns
RESET (all-1s)	3.75ns	15ns	40ns	1ns	59.75ns
Write (unknown)	75ns	15ns	190ns	1ns	209.75ns

Figure 6(a) and (b) illustrate, respectively, two scenarios – one where the write data is overwritten on all-1s (as in PreSET), and another one where it is overwritten on all-0s. We observe that for overwriting all-1s, step 3 (*compare*) ❹ and step 4 (*RESET*) ❺ are sufficient, while for overwriting all-0s, step 1 (*compare*) ❷ and step 2 (*SET*) ❸ are sufficient. Thus, in both these scenarios only two out of the conventional four steps are sufficient for servicing a PCM write, which improves both performance and energy consumption.

Using the timing parameters from Table 1, DATACON leads to 71.5% and 19% lower PCM write latency than Baseline, when using the RESET and SET timings, respectively. We conclude that overwriting all-1s (i.e., using only RESET operations) has a definitive performance advantage versus overwriting all-0s. In Section 6, we demonstrate an average 27% performance improvement with DATACON over PreSET [110] for all our evaluated workloads.

3.2 Quantifying Energy Improvement

Table 2 reports the energy consumption of the three write scenarios of Figures 5 & 6, i.e., overwriting the unknown content ('11011101'), overwriting all-0s ('00000000'), and overwriting all-1s ('11111111') for the same write data ('00100000').

[1] We expect these values to be similar for other PCM designs.

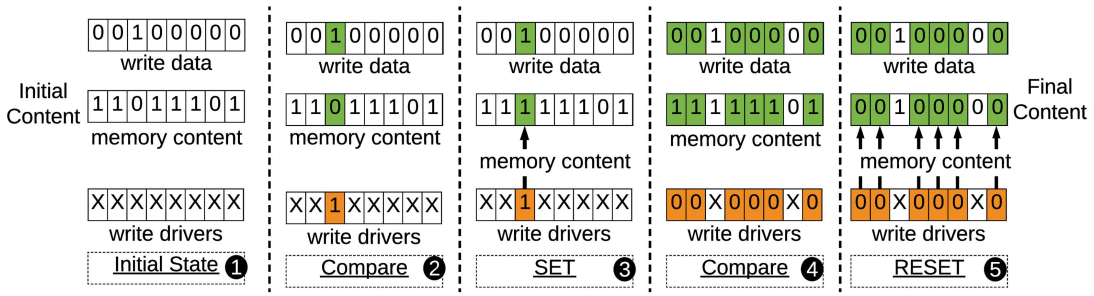

Figure 5. Servicing a PCM write request in a baseline system [77].

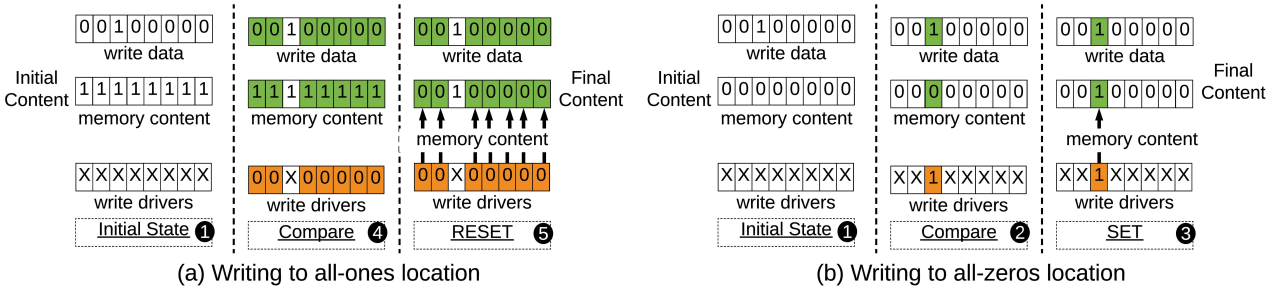

Figure 6. Servicing a PCM write by overwriting all-1s (a) and all-0s (b).

The total energy consumption includes (1) the energy to prepare the memory content (*preparation energy*), i.e., to program the memory content to all-0s or all-1s prior to servicing the write request, and (2) the energy to service the write request (*service energy*) by overwriting the memory content with the write data. The preparation step is *not* present when overwriting *unknown* content as in a Baseline system [77]. Preparing a memory content with all-1s (required by the PreSET technique of Qureshi et al. [110]) requires lower energy than preparing the location with all-0s (27pJ for the former vs. 115.2pJ for the latter). This is because only 2 SET operations are required in the former scenario, while 6 RESET operations are required in the latter scenario to prepare the data content for overwriting. Conversely, the service energy to overwrite all-1s is 10X higher than that to overwrite all-0s (column 4). This is because to overwrite all-1s, we need 7 RESET operations (see Figure 6(a)), which consume 134.4pJ of energy vs. 13.5pJ to overwrite all-0s, for which we only need 1 SET operation (see Figure 6(b)). We conclude that overwriting all-0s has a definitive energy advantage (20% lower energy than overwriting all-1s, and 11% lower energy than overwriting unknown content). In Section 6, we demonstrate an average 43% energy improvement with DATACON over PreSET [110] for all our evaluated workloads.

Table 2. Energy consumption of the three overwrite scenarios of Figures 5 and 6.

Write Data	Overwritten Content	Total Energy (pJ)		
		Preparation	Service	Total
00100000	11011101 (unknown)	0	144.7	**144.7**
	00000000	115.2	13.5	**128.7**
	11111111 (PreSET [110])	27	134.4	**161.4**

This example clearly demonstrates the latency-energy trade-offs in overwriting all-0s and all-1s content in PCM. Specifically, SET operations (to overwrite all-0s) are good for energy, while RESET operations (to overwrite all-1s) are good for performance. The exact energy and performance trade-off depends on the number of bits that are to be SET or RESET when overwriting content. Based on this motivating example, we now present our mechanism, DATACON, *data content aware PCM writes*.

4 Data Content Aware Access in PCM

We describe DATACON in the context of DRAM-PCM hybrid memory, where embedded DRAM (eDRAM) is used as a write cache to PCM main memory.[2] Figure 7 illustrates how an eDRAM cache line is mapped to 8 memory lines in a PCM rank. For completeness, we also show the shared last level cache (LLC) of the host CPU. For a read miss in LLC, the memory controller first checks to see if the read address is cached in eDRAM. If cached, the read data is sent from eDRAM to LLC. Otherwise, the memory controller issues a read request to PCM. The PCM read data is sent back to the requesting CPU, bypassing eDRAM. For a write miss in LLC, the memory controller checks to see if the write address is cached in eDRAM. If cached, the cache line is updated with the write data. Otherwise, a read request is issued to PCM. The PCM data is cached in eDRAM and also sent back to the requesting CPU. In doing so, the least-recently used eDRAM cache line is evicted, which generates a write request to PCM. These long-latency write requests to PCM are the

[2]Even though we use embedded DRAM as cache to PCM in our implementation and evaluations, DATACON is applicable to any type of hybrid memory or standalone PCM memory.

requests DATACON optimizes to improve both performance and energy consumption.

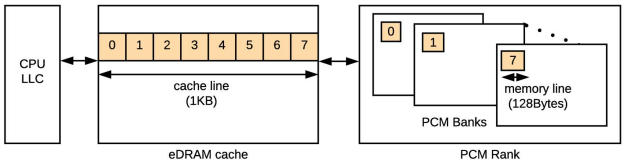

Figure 7. Mapping of an eDRAM cache line to 8 memory lines in a PCM rank, on our Baseline system.

We now describe DATACON's address translation mechanism, starting with a high-level overview.

4.1 High-level Overview

Figure 8 shows our system architecture. For a write request from the *write queue*, DATACON performs three operations. First, it selects the content to overwrite (all-0s or all-1s) based on the write data and the energy-latency trade-off for overwriting all-0s and all-1s (Section 4.2.2). Second, it selects a physical address containing the appropriate content to overwrite for servicing the write request and records this logical-to-physical address translation in a table (Section 4.2.1). Third, it schedules the write request to PCM.

For a read request from the *read queue*, DATACON performs two operations. First, it performs the logical-to-physical address translation for the read address using the table (Section 4.2.1). Second, it schedules the read request to PCM. DATACON also methodically re-initializes *unused* memory locations with all-0s or all-1s content in a manner that minimizes interference with PCM accesses (Section 4.2.3).

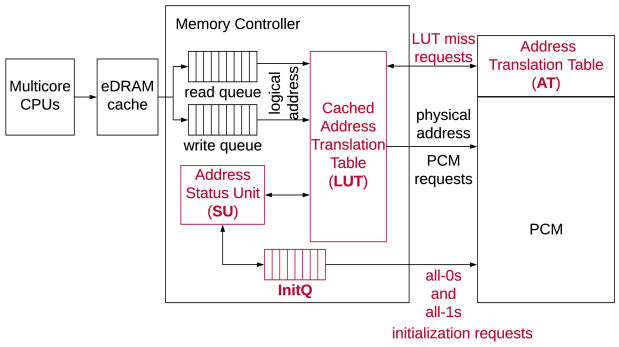

Figure 8. DATACON's address translation mechanism.

4.2 Detailed Design of DATACON

Figure 8 shows the detailed design of the memory controller to support DATACON's address translation mechanism. DATACON adds four new components to the baseline memory controller design, which are highlighted in the figure. We describe these components in detail.

The *first* component is the *address translation table* (**AT**). DATACON uses this table to record logical-to-physical address translations, which are needed to redirect write requests to the best overwritten content in PCM. AT requires one translation entry for every possible eDRAM cache line (e.g., of 1KB size). With 8GB/1KB = 8,388,608 cache lines per 8GB rank (see our simulation parameters in Table 3), DATACON requires 23MB of storage per rank for AT. We use a PCM partition inside the rank to store the AT.

To translate a logical address on demand, an access to the PCM partition storing the AT is required, which can increase the address translation latency. To address this, DATACON uses a second component, the *lookup table* (**LUT**), which reduces the latency of address translation by caching recently-used address translation information in the memory controller. We exploit the data access locality of a workload to cache the address translation information of a small number of recently-accessed cache lines in the LUT. Under this new caching mechanism, the memory controller categorizes each PCM request from eDRAM into one of two cases: 1) *LUT hit:*, i.e., the translation of the logical address is in LUT, and ii) *LUT miss:*, i.e., the translation of the logical address is not in LUT. In the first case, the logical address is translated using information from LUT and the request is scheduled to PCM. In the second case, the memory controller first issues a PCM request to read the address translation table (AT), caches the required AT entry in the LUT, and then translates the logical address using this entry. To make space inside LUT, address translation of the least recently used (LRU) cache line is evicted from LUT. If the evicted content is dirty, a PCM request is generated to write the updated content into the PCM partition storing the AT. In Section 6.5, we evaluate the size of LUT and find that storing the address translation information of only two PCM partitions in LUT (requiring 32KB of storage) is sufficient for the evaluated workloads.

The *third* component is the *address status unit* (**SU**). DATACON uses this unit to select a physical address for the logical address of a write request. Internally, SU implements two queues: 1) a 32-entry *ResetQ* queue, which stores physical addresses that are initialized to all-0s content and 2) a 32-entry *SetQ*, which stores physical addresses that are initialized to all-1s content. The size of the SU is 256B.

The *fourth* component is the *initialization queue* (**InitQ**). DATACON uses this queue to record unused physical locations in PCM, such that they can be re-initialized methodically. In terms of structure, InitQ is similar to the write queue but much simpler, in that each InitQ entry stores only address information and a single bit to indicate if the address is to be re-initialized with all-zeros or all-ones, while each write queue entry stores the write data as well the address.

We now describe the three key operations of DATACON: 1) address translation, 2) overwritten content selection, and 3) re-initialization to all-0s and all-1s content.

4.2.1 Address Translation. Figure 9 shows the details of the address translation operation for an example read and an example write request, using the newly-introduced DATACON components of the memory controller. To service a read request, the logical read address (0x30a in this

example) is translated to the physical address 0x11fe and scheduled to PCM. To service a write request, DATACON first selects between overwriting all-0s and all-1s content based on the write data 0x0005 using the *Overwritten Content Selection* unit (which we describe in Section 4.2.2). In this example, without loss of generality, we assume the overwritten content to be all-0s. Therefore, DATACON selects address 0x910b from the head of the ResetQ to redirect the write request to. The translation of logical address 0x1078 to physical address 0x910b is recorded in LUT. The newly-remapped address 0x1078 is also inserted in the InitQ to be re-initialized to all-0s or all-1s content (Section 4.2.3).

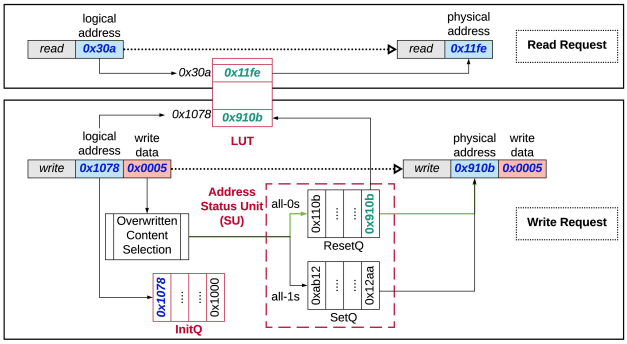

Figure 9. Illustration of DATACON's address translation mechanism for servicing an example PCM read and write.

4.2.2 Overwritten Content Selection. Figure 10 illustrates DATACON's policy to select the overwritten content for every PCM write request. If the number of SET bits in the write data is more than 60%, DATACON checks the status table to see if there is all-0s or all-1s locations available in PCM. If an all-1s location is available, DATACON selects it as the overwritten content. This is because overwriting all-1s using only RESET operations is both more energy efficient and higher performance than overwriting any other content when the fraction of SET bits in the write data is more than 60%. In this way, DATACON optimizes both *performance and energy*. Otherwise, if an all-0s location is available, DATACON selects it as the overwritten content. This is because overwriting all-0s using only SET operations has lower latency than overwriting unknown content. In this way, DATACON optimizes *performance*.

Conversely, if the number of SET bits in the write data is less than 60% and there is an all-0s location available, DATACON selects it as the overwritten content. This is because overwriting all-0s using only SET operations is more energy efficient than overwriting any other content when the fraction of SET bits in the write data is less than 60%. In this way, DATACON optimizes *energy*. Otherwise, if an all-1s location is available, DATACON selects it as the overwritten content. This is because overwriting all-1s using only RESET operations has lower latency than overwriting unknown content. In this way, DATACON optimizes *performance*.

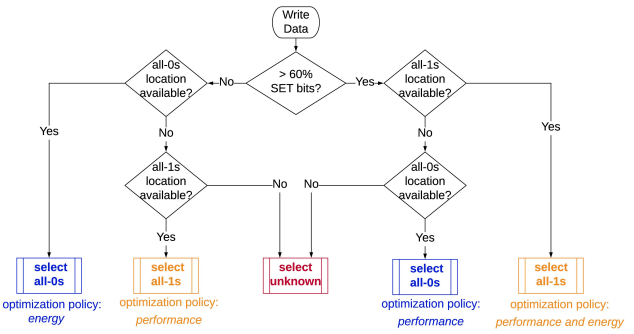

Figure 10. Flowchart for overwritten content selection.

DATACON overwrites the existing content, which is unknown at the time of a PCM write only if there are no all-0s or all-1s locations available in PCM to service the write request.

4.2.3 Re-Initialization of Unused Memory Locations. As write requests are continuously re-mapped to different locations, DATACON re-initializes *unused* memory locations with all-0s or all-1s so that future PCM write requests can be serviced using all-0s or all-1s locations. To trigger such all-0s and all-1s re-initialization, we use *thresholding*. Any time the number of entries in the ResetQ (which stores all-0s locations) or the SetQ (which stores all-1s locations) falls below the initialization threshold (th_{init}), DATACON issues a re-initialization request of a memory address stored in the InitQ. However, the re-initialization request may not be serviced immediately. DATACON schedules re-initialization of memory locations off the critical path to minimize their interference with regular read and write accesses by exploiting a PCM bank's partition-level parallelism. Re-initialization requests in DATACON are serviced when

- the read and the write queues are both empty, or
- the write queue is empty and a read request from the read queue is being serviced by a memory partition that is different from the one from which the re-initialization request is to be serviced.

By using a threshold to trigger memory re-initializations, DATACON ensures that the re-initialization overhead is reasonable. Furthermore, by scheduling these requests off the critical path, DATACON's impact on system performance is also minimized (See Section 6.4).

4.3 Other Considerations

We discuss two main issues that may effect DATACON's benefits in modern systems: data encryption and irregular memory accesses.

4.3.1 DATACON with Memory Encryption. Many persistent memory systems [6, 43, 84, 90, 92, 111, 126, 146] now include support for data encryption [8, 9, 32, 42, 47, 113, 114, 123, 138, 139]. Enabling encryption inside the memory chip can lead to the following two issues. First, the write data going out of the CPU chip can change before overwriting

the content at a memory location inside PCM. Second, 0s and 1s in the write data may be equally distributed for most write data. DATACON can be easily adapted to provide performance gain for such systems where encryption is enabled *inside* the memory chip. We observe that RESET operations are *always* better than SET operations in terms of latency, while SET operations may be more energy efficient than RESET operations depending on the number of 0s and 1s in the write data. This follows directly from the two observations we made in Section 3.

Therefore, when the write data content is unknown and/or equally distributed across 0s and 1s, we propose to configure DATACON to prioritize only performance by overwriting all-1s using only RESET operations for every PCM write. We analyze the performance improvement of this all-1s configuration of DATACON in Section 6.6. We observe that by always overwriting all-1s, DATACON can always improve performance 1) when the fraction of 0s and 1s in the write data are equal (as expected with encryption) and 2) *independently* of the write data content.

4.3.2 DATACON with Irregular Memory Accesses.

If a workload has many irregular memory accesses (e.g., as common in graph analytics [2, 3, 36, 99, 120] or pointer chasing workloads [19, 20, 34, 38, 46, 49, 64, 69, 102, 130]), the workload would likely generate many LUT misses. This is because PCM requests in such workloads tend to access many PCM partitions. As DATACON caches the address translation of only 2 PCM partitions inside LUT, most irregular accesses will miss in the LUT, generating extra PCM traffic. To address this, we store the full AT in a separate dedicated PCM partition, which can be accessed in parallel with other partitions within a PCM bank using the bank's partition-level parallelism [121]. Doing so reduces the performance overhead of servicing the LUT misses. However, the energy overhead of LUT misses remains a concern. To address this, we store the AT in the eDRAM cache, which provides lower-latency and lower-energy access compared to PCM.

5 Evaluation Methodology

5.1 System Configuration

To evaluate DATACON, we designed a cycle-accurate DRAM-PCM hybrid memory simulator with the following:

- A cycle-level in-house x86 multi-core simulator, whose front-end is based on Pin [85]. We configure this to simulate 8 out-of-order cores.
- A main memory simulator, closely matching the JEDEC Nonvolatile Dual In-line Memory Module (NVDIMM)-N/F/P Specifications [74]. This simulator is composed of Ramulator [68], to simulate DRAM, and a cycle-level PCM simulator based on NVMain [104].
- Power and latency for DRAM and PCM are based on Intel/Micron's 3D Xpoint specification [21, 124]. Energy is modeled for DRAM using DRAMPower [26] and for PCM using NVMain, with parameters from [124].

Table 3 shows our simulation parameters. The memory controller implements a 16-entry read queue, a 16-entry write queue, and an 8-entry InitQ per each bank.

Table 3. Major Simulation Parameters.

Processor	8 cores per socket, 3.32 GHz, out-of-order
L1 cache	Private 32KB per core, 8-way
L2 cache	Private 512KB per core, 8-way
L3 cache	Shared 8MB, 16-way
eDRAM cache	Shared 64MB per socket, 16-way, on-chip
Main memory	128GB PCM. 4 channels, 4 ranks/channel, 8 banks/rank, 8 partitions/bank, 128 tiles/partition, 4096 rows/tile. Memory interface = DDR4 Memory clock = 1066MHz PCM Timings = See Table 1

5.2 Evaluated Workloads

We evaluate the following workloads.

- 8 SPEC CPU2017 workloads [22]: bwaves, cactusBSSN, leela, mcf, omnetpp, parset, roms, and xalancbmk.
- 2 NAS Parallel workloads [10]: bt and ua.
- 10 TensorFlow machine learning workloads [1]: mlp, cnn, gan, rnn, dcgan, bidirectional rnn (bi-rnn), autoencoder (autoenc), logistic regression (lr), random forest (rf), and word2vec.

All workloads are executed for 10 billion instructions. Figure 11 plots the eDRAM Misses Per Kilo Instruction (MPKI) of the evaluated workloads. For other workloads with low MPKI (not reported in Section 6), DATACON neither significantly improves nor hurts performance and energy.

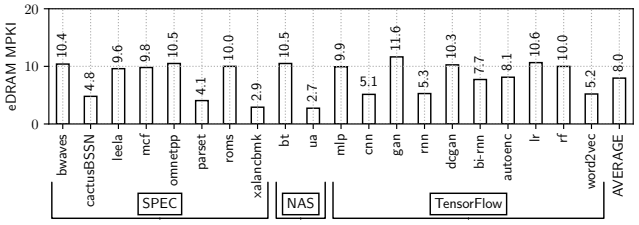

Figure 11. eDRAM cache MPKI for the evaluated workloads.

5.3 Evaluated Systems

We evaluate the following systems.

- *Baseline* [77] services PCM writes by overwriting *unknown* content.
- *PreSET* [110], which services PCM writes by *always* overwriting all-1s.
- *Flip-N-Write* [33] services PCM writes by *finding out* the memory content using additional reads and programming only bits that are different from the write data.
- *DATACON* redirects write requests to overwrite all-0s or all-1s content, depending on the write data.

5.4 Figures of Merit

We report the following figures of merit in this work.

- *Access Latency:* The sum of queuing delay and service latency, averaged across all PCM accesses.
- *System Energy Consumption:* The sum of DRAM and PCM energy for all accesses.
- *Execution Time:* The time to complete a workload.

6 Results and Discussions

6.1 Overall System Performance

Figure 12 plots the execution time of each workload for each of the evaluated systems, normalized to Baseline. We make two main observations.

First, between PreSET and Flip-N-Write, PreSET has higher performance. Average execution time of PreSET is 18% lower than Baseline, while that of Flip-N-Write is 12% higher than Baseline. PreSET improves performance over Baseline by programming PCM cells only in the RESET direction. Flip-N-Write, which finds out the memory content before overwriting it, has lower performance than Baseline due to the extra PCM read accompanying *every* PCM write.

Second, DATACON has the highest performance among all the evaluated techniques. Average execution time of DATACON is 40% lower than Baseline, 47% lower than Flip-N-Write, and 27% lower than PreSET. DATACON outperforms PreSET because it greatly reduces the probability of overwriting unknown content on a write access. It does so in two ways. First, DATACON overwrites all-0s as well, using only RESETs, when there is no all-1s location to overwrite, yet PreSET is limited to only overwriting all-1s, and it overwrites unknown content in the absence of opportunity to overwrite all-1s. As we show, overwriting all-0s provides 19% latency reduction per access than overwriting unknown content (Section 3.1). Second, DATACON methodically reinitializes unused memory locations to all-0s and all-1s content, whereas PreSET does so only opportunistically. We find that the probability of overwriting an all-0s or all-1s location is therefore much higher in DATACON than in PreSET.

To illustrate this, Figure 13 plots the distribution of total PCM writes into the type of content they overwrite in DATACON and PreSET. We observe that PreSET overwrites all-1s content for 41% of PCM write accesses and unknown content for 59%. DATACON overwrites all-0s content for 54% of write accesses, all-1s content for 42%, and unknown content for only 4% of all PCM write accesses. DATACON has higher performance and lower energy than PreSET because DATACON greatly reduces the number of times unknown content is overwritten by also overwriting all-0s content in addition to all-1s content.

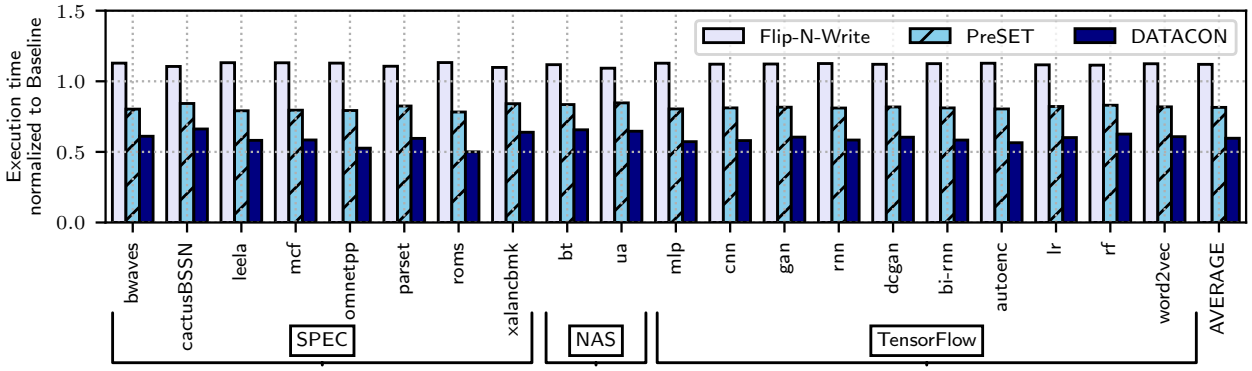

Figure 12. Execution time normalized to Baseline for the evaluated workloads.

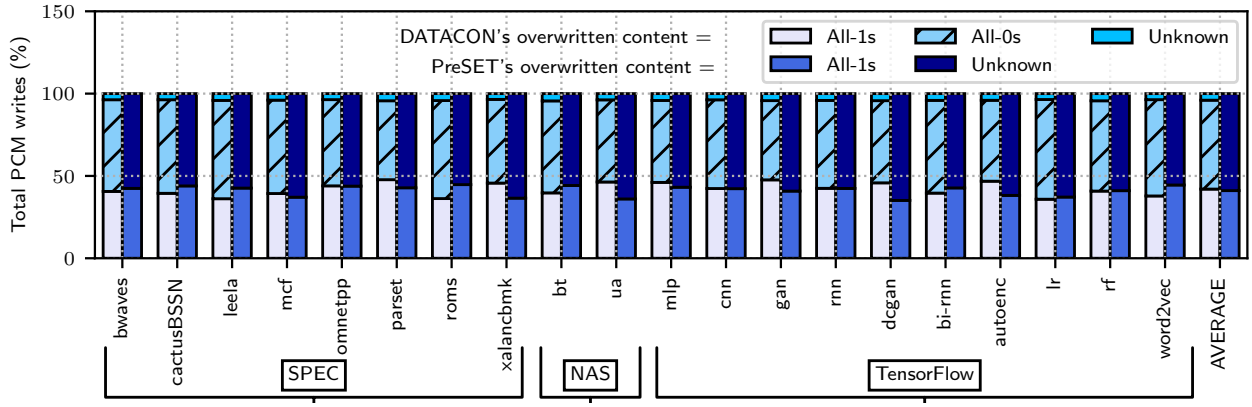

Figure 13. Total PCM writes distributed into overwriting all-0s, all-1s, and unknown content in DATACON, compared to overwriting all-1s and unknown content in PreSET [110].

6.2 Access Latency

Figure 14 plots average access latency of the workloads for each of our evaluated systems, normalized to Baseline. We make two main observations.

First, between PreSET and Flip-N-Write, PreSET has lower access latency than Flip-N-Write. Access latency of PreSET is 19% lower than Baseline and that of Flip-N-Write is 10% higher than Baseline. Flip-N-Write, which increases the queuing delay of a PCM write request to serve the extra read request needed to first know the memory content, has higher access latency than Baseline. PreSET, on the other hand, reduces access latency by programming PCM cells only in the SET direction. Second, average access latency of DATACON is 43% lower than Baseline, 50% lower than Flip-N-Write, and 31% lower than PreSET. The reason for DATACON's improvements are the same as explained in Section 6.1.

6.3 Energy Consumption

Figure 15 plots the total energy consumption of each of the workloads for each of our evaluated systems, normalized to Baseline. We make three main observations.

First, the average energy consumption of Flip-N-Write is lower than Baseline by only 0.5%. Although Flip-N-Write saves PCM write energy by minimizing the programming of PCM cells in a memory location based on the write data and the overwritten content, such energy saving is compensated by the energy increase due to the PCM read, which is issued before every write access to find out the overwritten data.

Second, between PreSET and Flip-N-Write, PreSET has an average 30% higher energy consumption than Flip-N-Write. High energy consumption in PreSET is due to three reasons. First, PreSET consumes energy to prepare a memory location with all-1s content before overwriting it. This preparation energy is higher than the energy consumed in reading the content as required by Flip-N-Write. Second, as PreSET does not analyze the write data, the energy consumption in PreSET can be high when there are many RESET operations to be performed to service a write. This is the case where the number of SET bits in the write data is low. As we have discussed in Observation 2 and reported in Figure 2, on average, only 33% of PCM writes in the evaluated workloads has more than 60% SET bits in the write data. Flip-N-Write has lower energy than PreSET in scenarios where the number of SET bits is low. This is because Flip-N-Write minimizes programming of PCM cells during a write by analyzing both the write data and the overwritten content. Finally, PreSET overwrites unknown content when there is no opportunity to prepare the memory content due to outstanding requests in the request queue. Overwriting unknown content increases energy consumption.

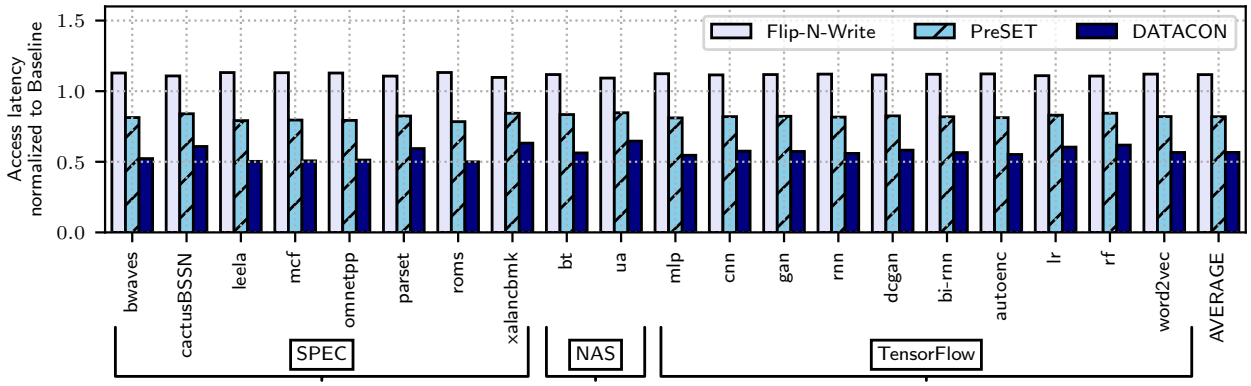

Figure 14. Access latency normalized to Baseline for the evaluated workloads.

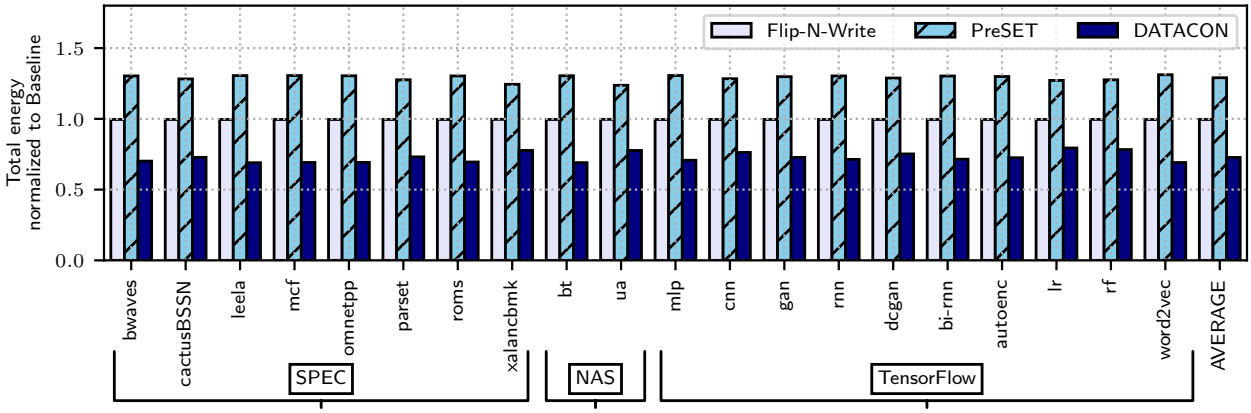

Figure 15. Energy consumption normalized to Baseline for the evaluated workloads.

Third, although DATACON incurs energy overhead for methodically re-initializing unused memory locations, the average energy consumption of DATACON is still 27% lower than Baseline, 26% lower than Flip-N-Write, and 43% lower than PreSET. This improvement is because DATACON exploits the energy-latency trade-offs of SET and RESET operations of the PCM cells in selecting between using only SET or only RESET operations during a PCM write, unlike PreSET, which always uses *only* SET operations, or Flip-N-Write, which uses *both* SET and RESET operations during a PCM write. As explained in Section 6.1, DATACON minimizes the probability of overwriting unknown content among the three mechanisms, which also reduces energy consumption.

6.4 Re-Initialization Overhead

Figure 16 plots the total PCM energy distributed into energy to service reads, writes, and re-initialization requests in DATACON for the evaluated workloads. We make the following two main observations from this data.

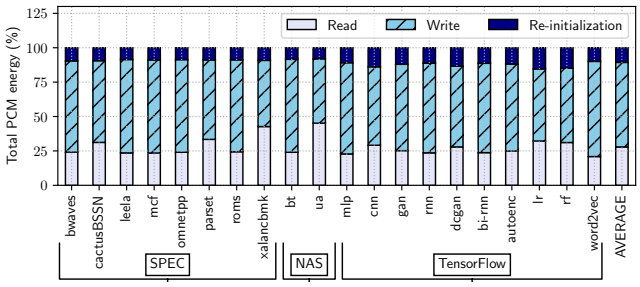

Figure 16. Total PCM energy distributed into energy to service reads, writes, and re-initialization requests.

First, PCM reads and writes constitute, on average, 89% of the total PCM energy. The distribution of read and write energy within a workload depends on the relative proportion of PCM reads and writes in the workload. NAS_ua, which has a higher proportion of PCM reads, has a higher fraction of total energy spent on reads than NAS_bt, which has higher proportion of PCM writes. Second, re-initialization requests constitute, on average, 11% of the total PCM energy. This overhead is to service the re-initialization of unused memory locations in PCM, every time the available number of initialized all-0s or all-1s memory locations falls below the initialization threshold (th_{init}), which is set to 16.

6.5 LUT Sizing

Figure 17 plots DATACON's execution time normalized to Baseline for each of the evaluated workloads when the LUT size is increased from 2 partitions (our default configuration) to 8 partitions. We make two main observations.

First, when the LUT size is increased, the number of LUT misses reduces and thus performance increases. With only 4 and 8 recently-used PCM partitions cached in the LUT, the average execution time of DATACON is respectively, only 3% and 5% lower than DATACON with the default configuration. Second, for most workloads, caching the address translation of 2 recently-used PCM partitions is sufficient to provide high performance and there is only marginal performance improvement from caching more partitions. This is because workloads exhibit a high degree of partition-level spatial locality in accesses, which DATACON exploits to its benefit in keeping the address translation overhead reasonable.

6.6 Overwriting Only All-0s or All-1s Content in DATACON

We configure DATACON to overwrite only all-0s and only all-1s content for every single PCM write, to evaluate the benefits of having the ability to decide between all-0s and all-1s depending on overwritten content. We call these two modes of DATACON as *DATACON-all-0s* and *DATACON-all-1s*, respectively. DATACON-all-1s could also be a good mechanism to employ when memory content is encrypted inside the memory chip, as explained in Section 4.3.1.

Figure 18 plots the execution time of DATACON, DATACON-all-0s, and DATACON-all-1s normalized to Baseline for the evaluated workloads. We make the following three main observations.

First, between DATACON-all-0s and DATACON-all-1s, DATACON-all-1s has higher performance. The average execution time of DATACON-all-1s is 58.5% lower than Baseline and that of DATACON-all-0s is 34% lower than Baseline. As discussed in Section 3.1, to overwrite all-1s content using DATACON-all-1s, PCM cells are programmed in the RESET direction, which has lower latency than programming the PCM cells in the SET direction as required by DATACON-all-0s to overwrite all-0s content.

Second, although both PreSET and DATACON-all-1s overwrite all-1s at every PCM write, the performance of DATACON-all-1s is, on average, 50% higher than PreSET. This is because PreSET requires the memory location to be initialized to all-1s content first before overwriting it during a write. If there are outstanding requests in the request queue, PreSET skips the initialization step and proceeds to overwrite unknown content, thereby hurting performance. DATACON-all-1s, on the other hand, overwrites any available all-1s location in PCM by translating the write address. It also methodically re-initializes locations to all-1s to maximize the probability of overwriting them when servicing write requests. Therefore, DATACON-all-1s overwrites all-1s content for more PCM writes (on average, 2.3x more) than PreSET does, thereby achieving higher performance.

Third, DATACON, which selects between overwriting all-0s and all-1s based on the write data and the energy-performance trade-offs, has higher performance than DATACON-all-0s (average 10% lower execution time) and lower performance than DATACON-all-1s (average 46% higher execution time). Despite this relatively lower performance of DATACON compared to DATACON-all-1s, DATACON outperforms PreSET by achieving 27% lower average execution.

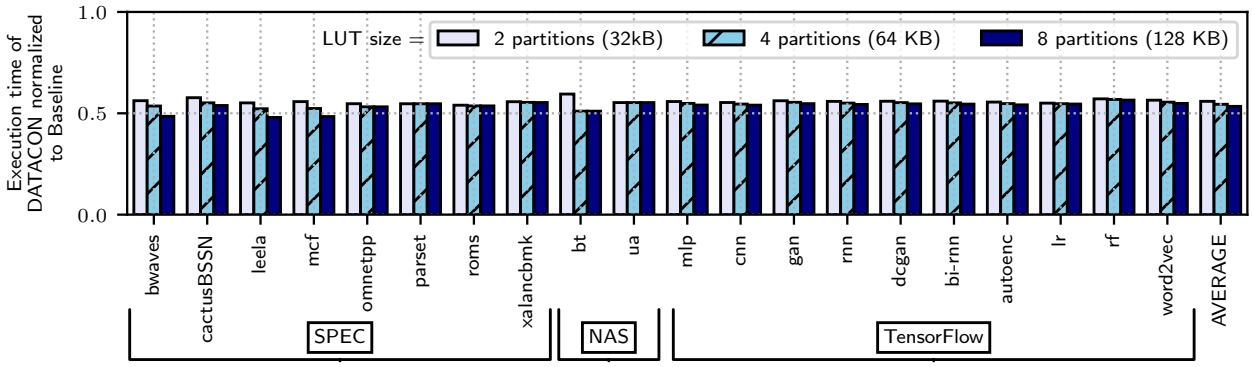

Figure 17. Execution time of DATACON normalized to Baseline with three address translation cache sizes – 2 partitions (32KB) (our default configuration), 4 partitions (64KB), and 8 partitions (128KB).

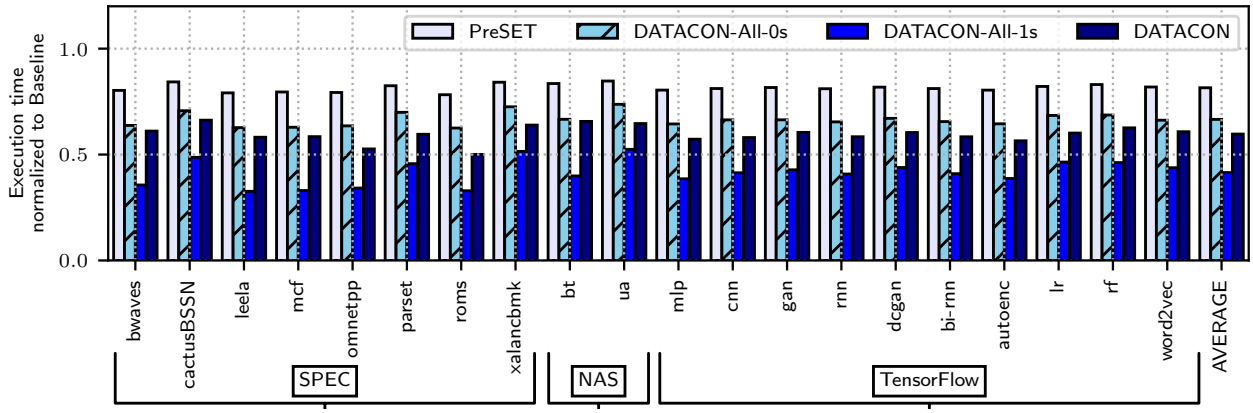

Figure 18. Execution time of DATACON-all-0s and DATACON-all-1s normalized to Baseline for the evaluated workloads.

Figure 19 plots the total energy consumption of the four mechanisms for the evaluated workloads. We make two main observations. First, DATACON-all-1s leads to higher energy consumption than DATACON-all-0s because RESET operations used by DATACON-all-1s require higher energy than SET operations used by DATACON-all-0s. Second, DATACON has lower energy consumption than the other mechanisms because DATACON selects between SET and RESET operations by effectively considering the fraction of 1s versus 0s in the write data and the associated energy-performance trade-offs of SET and RESET operations (as explained in Section 3).

6.7 Performance and Energy Trade-off

To analyze the performance and energy trade-off evaluation performed by DATACON in response to the write data, we designed a microbenchmark, where the *same* write data is used for *every* PCM write access. We ran the microbenchmark repeatedly while increasing the fraction of set bits in the write data every time. Figure 20 plots the execution time

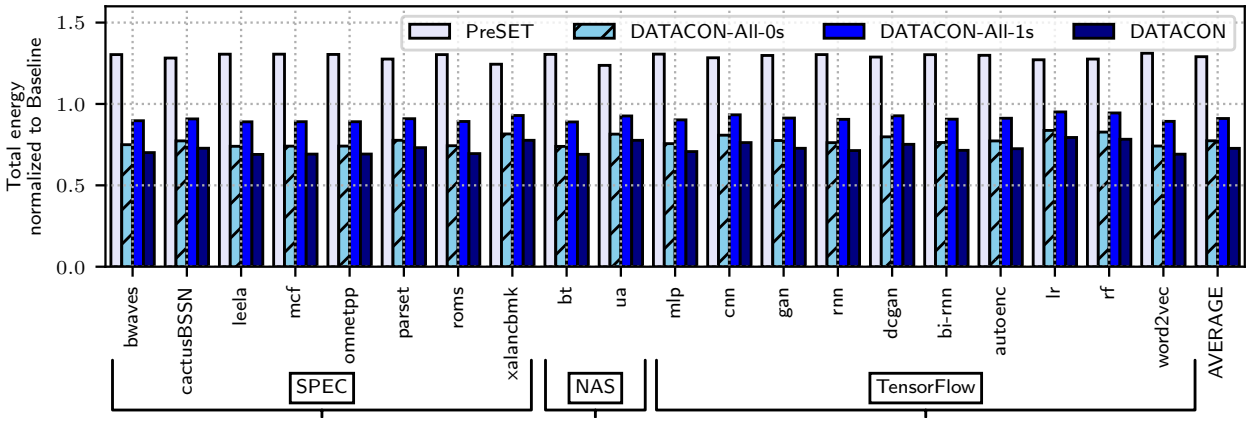

Figure 19. Energy consumption of DATACON-all-0s and DATACON-all-1s normalized to Baseline for the evaluated workloads.

and energy of DATACON normalized to peak points in each curve when increasing the fraction of set bits from 0% (write data is all-0s) to 100% (write data is all-1s). We make the following two main observations.

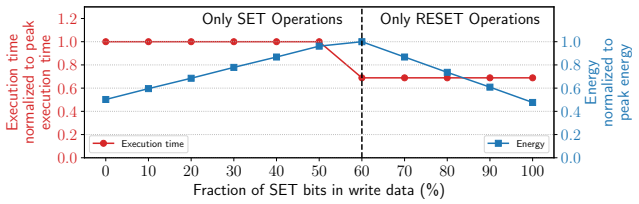

Figure 20. Execution time and energy of DATACON normalized to their peak values with increasing fraction of SET bits in the write data of a microbenchmark.

First, as the fraction of set bits in write data is increased, energy consumption increases, reaching a peak at around 60%, and then decreases. This is due to the energy difference of SET and RESET operations, which we have analyzed in Section 3.1 and illustrated in our Observation 1 in Figure 1. DATACON lowers the energy consumption by switching from only SET operations (overwriting all-0s) to only RESET operations (overwriting all-1s) as the fraction of SET bits in the write data exceeds 60%. Second, as the RESET operation in PCM has lower latency than the SET operation (see Table 1), switching to using only RESET operations also leads to lower execution time, i.e., higher performance.

6.8 Effect on PCM Lifetime

We compare the PCM lifetime (in years) obtained with DATACON against Baseline and Baseline+SecurityRefresh, which is our Baseline extended with a dynamic address translation scheme called SecurityRefresh [118], a state-of-the-art wear-leveling mechanism for PCM. We assume a pessimistic PCM cell endurance of 10^7 writes. Figure 21 shows the impact of DATACON on PCM lifetime for the evaluated workloads, normalized to the lifetime obtained using Baseline+SecurityRefresh. We also plot the lifetime with Baseline (without SecurityRefresh). We make two main observations.

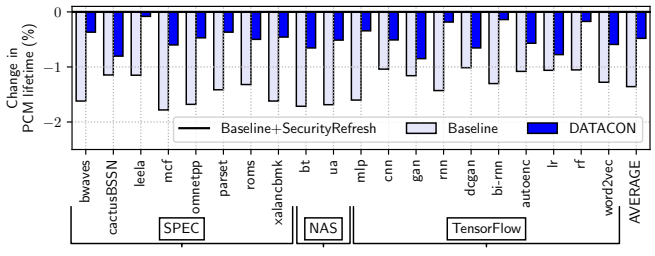

Figure 21. Effect of DATACON on PCM Lifetime relative to Baseline with SecurityRefresh [118].

First, Baseline has an average 1.35% lower lifetime than Baseline+SecurityRefresh. This is because the dynamic address translation mechanism in Baseline+SecurityRefresh distributes the PCM write requests of a workload to other locations, balancing the wear-out of PCM cells. Baseline, on the other hand, writes to only the memory locations requested by the program, inducing higher wear-out in the PCM cells in those requested locations.[3] Second, the PCM lifetime with DATACON is slightly lower than Baseline with SecurityRefresh by, on average, 0.48%, despite both techniques performing dynamic address translation when servicing PCM writes.[4] This is due to the different optimization objectives of the two techniques. When selecting a physical address for a write request, DATACON searches for a memory location that contains the best data pattern for the write data that would result in the lowest latency and energy. For Baseline with SecurityRefresh, the objective is to redirect a write request to a memory location that has the *least wear-out*, but the content is unknown at the time of the write. We observe that the impact of PreSET and Flip-N-Write on PCM lifetime are both similar to Baseline *without* SecurityRefresh. This is because both PreSET and Flip-N-Write do not use address translation, which helps to level the wear-out in PCM. DATACON achieves slightly better PCM lifetime than Baseline, Flip-N-Write, and PreSET. We believe DATACON can be combined with SecurityRefresh in the future to achieve even smaller impact on PCM lifetime.

7 Related Works

To our knowledge, this is the first work that improves both performance and energy consumption in PCM by analyzing only the data to be written, without having to first read the overwritten content.

7.1 Related Concepts in Flash Memory

The Erase and Program operations in Flash memory are equivalent to the SET and RESET operations in PCM with similar asymmetries in energy and latency. To facilitate erase-before-write in Flash memory, logical-to-physical address translations are performed to hide the long latency of erase operations [44, 65, 87]. Invalidated physical locations are erased periodically [25, 28].

DATACON is different from the erase-before-write in Flash memory for three reasons. First, DATACON selects between overwriting all-0s and all-1s based on energy-latency trade-offs, whereas writing in Flash memory is *always* performed on erased content (i.e., on all-1s). Second, the granularity of read, write and re-initialization operations are the same in DATACON, whereas the granularity of an erase operation in Flash memory is much *larger* than the granularity of a write operation, which leads to different performance and energy trade-offs. Third, DATACON is an *optional* operation and tries to maximize overwriting all-0s or all-1s, whereas

[3] SecurityRefresh's lifetime improvement over Baseline is lower than that reported in [118] because of the eDRAM write cache used in our system, which mitigates PCM's endurance problem by greatly reducing the number of write accesses to PCM.

[4] DATACON has 25% higher performance and 40% lower energy than Baseline+SecurityRefresh.

an erase is a *required* operation before writing a block in Flash memory. Inspired from the erase-before-write in Flash memory, block erase is proposed for PCM [75].

7.2 Writeback Optimizations

Line-level writeback is proposed in [77, 105, 107], which tracks the lines in a memory page that are dirty and selectively writes only those lines to PCM when a DRAM/eDRAM cache line is evicted. DATACON implements the line-level writeback policy *only* to overwrite unknown content. Dynamic write consolidation is proposed in [79, 119, 122, 129, 135] to consolidate multiple writes targeting the same row into one write operation. Write activity reduction is proposed in [50, 51], which uses registers for servicing costly PCM writes. Multi-stage PCM write is proposed in [108, 109, 142, 143] to improve PCM performance by completing a long-latency write access to PCM in several steps, with each step scheduled off the critical path of read accesses. DATACON is complementary to all these approaches.

7.3 Read Before Write

Read before write is proposed to reduce the number of SET operations when servicing PCM write accesses. One example is the Flip-N-Write [33], which compares the overwritten content with the *write data* to decide if writing an inverted version of the data would result in a lower number of SET operations than writing the requested data itself. Other works propose variants of this approach [31, 39, 54, 93, 137], where the write data is differentially encoded with the overwritten content. The differentially-encoded data is used to overwrite the content. All these techniques convert a PCM write operation into a PCM read, followed by a PCM write. We compare DATACON to Flip-N-Write and find that DATACON performs significantly better (Section 6).

7.4 Related Works in Multi-Level PCM

In recent PCM devices, PCM cells are used to store multiple bits per cell (referred to as multilevel cell or MLC). MLC PCM offers greater capacity per unit area at the cost of asymmetric latency and energy in accessing the bits in a cell. Yoon et al. propose an architectural technique for data placement in MLC PCM [141], exploiting latency and energy asymmetries. Qureshi et al. propose a morphable PCM system [108], dynamically adapting between high-density and high-latency MLC PCM and low-density and low-latency single-level cell PCM. Qureshi et al. propose write cancellation and pausing to allow reads to be serviced faster by interrupting long write operations in PCM [109]. Jiang et al. propose write truncation [56], where a write operation is truncated to allow read operations, compensating for the loss in data integrity with stronger ECC. These techniques are also complementary to and can be combined with DATACON.

8 Conclusions

We introduce DATACON, a simple and effective mechanism to reduce the latency and energy of write operations in Phase Change Memory (PCM). DATACON is based on the key observation that overwriting *unknown* memory content can incur significantly higher latency and energy compared to overwriting *known* all-zeros or all-ones data pattern.

Based on this observation, DATACON redirects PCM write requests to overwrite memory locations containing all-zeros or all-ones, and overwrites unknown content only when it is absolutely necessary to do so. The key idea is to estimate how much a PCM write access would benefit from a particular *known* content (i.e., all zeros or all ones) by comprehensively considering the fraction of set bits in the write data and the energy-latency trade-offs for SET and RESET operations in PCM. Based on this estimate, DATACON translates the logical write address to a physical address in PCM containing the best overwritten content and records this address translation in a table to service future requests. DATACON exploits the data access locality in workloads to cache a small portion of address translation table inside the memory controller, lowering the address translation latency. DATACON methodically re-initializes *unused* memory locations to minimize the probability of overwriting unknown content, which improves both performance and energy efficiency. Results of our experiments with workloads from SPEC CPU2017, NAS Parallel Benchmarks, and state-of-the-art machine learning applications show that DATACON improves the effective access latency by 31%, system performance by 27%, and overall system energy consumption by 43% compared to the best of previously-proposed state-of-the-art techniques.

We **conclude** that DATACON is a simple yet efficient technique to reduce the latency and energy of PCM writes, which significantly improves performance and energy efficiency of modern memory systems.

Acknowledgments

This work is supported by 1) the National Science Foundation Faculty Early Career Development Award CCF-1942697 (CAREER: Facilitating Dependable Neuromorphic Computing: Vision, Architecture, and Impact on Programmability) and 2) the National Science Foundation Award CCF-1937419 (RTML: Small: Design of System Software to Facilitate Real-Time Neuromorphic Computing).

References

[1] "TensorFlow models and datasets," *https://www.tensorflow.org/resources/models-datasets*, 2020.
[2] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in *ISCA*, 2015.
[3] J. Ahn, S. Yoo, O. Mutlu, and K. Choi, "PIM-enabled instructions: a low-overhead, locality-aware processing-in-memory architecture," in *ISCA*, 2015.

[4] S. J. Ahn, Y. N. Hwang, Y. J. Song, S. H. Lee, S. Y. Lee, J. H. Park, C. W. Jeong, K. C. Ryoo, J. M. Shin, Y. Fai, J. H. Oh, G. H. Koh, G. T. Jeong, S. H. Joo, S. H. Choi, Y. H. Son, J. C. Shin, Y. T. Kim, H. S. Jeong, and Kinam Kim, "Highly reliable 50nm contact cell technology for 256Mb PRAM," in *VLSI Technology*, 2005.

[5] H. Akinaga and H. Shima, "Resistive random access memory (ReRAM) based on metal oxides," *Proceedings of the IEEE*, 2010.

[6] M. Alshboul, J. Tuck, and Y. Solihin, "Lazy persistency: A high-performing and write-efficient software persistency technique," in *ISCA*, 2018.

[7] G. Atwood, "PCM applications and an outlook to the future," in *Phase Change Memory*, 2018.

[8] A. Awad, Y. Wang, D. Shands, and Y. Solihin, "Obfusmem: A low-overhead access obfuscation for trusted memories," in *ISCA*, 2017.

[9] A. Awad, M. Ye, Y. Solihin, L. Njilla, and K. A. Zubair, "Triad-NVM: Persistency for integrity-protected and encrypted non-volatile memories," in *ISCA*, 2019.

[10] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga, "The NAS parallel benchmarks," *Supercomputing Applications*, 1991.

[11] R. Bez, "Chalcogenide PCM: A memory technology for next decade," in *IEDM*, 2009.

[12] S. Bock, B. Childers, R. Melhem, D. Mosse, and Y. Zhang, "Analyzing the impact of useless write-backs on the endurance and energy consumption of PCM main memory," in *ISPASS*, 2011.

[13] S. Bock, B. R. Childers, R. Melhem, and D. Mossé, "Concurrent page migration for mobile systems with OS-managed hybrid memory," in *CF*, 2014.

[14] S. Bock, B. R. Childers, R. Melhem, and D. Mossé, "Characterizing the overhead of software-managed hybrid main memory," in *MASCOTS*, 2015.

[15] S. Bock, B. R. Childers, R. Melhem, and D. Mosse, "HMMSim: A simulator for hardware-software co-design of hybrid main memory," in *NVMSA*, 2015.

[16] S. Bock, B. R. Childers, R. Melhem, and D. Mossé, "Understanding the limiting factors of page migration in hybrid main memory," in *CF*, 2015.

[17] S. Bock, B. R. Childers, R. Melhem, and D. Mossé, "Concurrent migration of multiple pages in software-managed hybrid main memory," in *ICCD*, 2016.

[18] D. Bondurant, "Ferroelectronic RAM memory family for critical data storage," *Ferroelectrics*, 1990.

[19] A. Boroumand, S. Ghose, M. Patel, H. Hassan, B. Lucia, K. Hsieh, K. T. Malladi, H. Zheng, and O. Mutlu, "LazyPIM: An efficient cache coherence mechanism for processing-in-memory," *CAL*, 2016.

[20] A. Boroumand, S. Ghose, Y. Kim, R. Ausavarungnirun, E. Shiu, R. Thakur, D. Kim, A. Kuusela, A. Knies, P. Ranganathan *et al.*, "Google workloads for consumer devices: Mitigating data movement bottlenecks," in *ASPLOS*, 2018.

[21] K. Bourzac, "Has Intel created a universal memory technology?" *Spectrum*, 2017.

[22] J. Bucek, K.-D. Lange, and J. v. Kistowski, "SPEC CPU2017: Next-generation compute benchmark," in *ICPE*, 2018.

[23] G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," *IBM JRD*, 2008.

[24] G. W. Burr, M. J. Breitwisch, M. Franceschini, D. Garetto, K. Gopalakrishnan, B. Jackson, B. Kurdi, C. Lam, L. A. Lastras, A. Padilla, B. Rajendran, S. Raoux, and R. S. Shenoy, "Phase change memory technology," *Vacuum Science & Technology B*, 2010.

[25] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo, and O. Mutlu, "Error characterization, mitigation, and recovery in flash-memory-based solid-state drives," *Proceedings of the IEEE*, 2017.

[26] K. Chandrasekar, C. Weis, Y. Li, B. Akesson, N. Wehn, and K. Goossens, "DRAMPower: Open-source DRAM power & energy estimation tool," *http://www. drampower. info*, 2012.

[27] K. K. Chang, A. G. Yağlıkçı, S. Ghose, A. Agrawal, N. Chatterjee, A. Kashyap, D. Lee, M. O'Connor, H. Hassan, and O. Mutlu, "Understanding reduced-voltage operation in modern DRAM devices: Experimental characterization, analysis, and mechanisms," *SIGMETRICS*, 2017.

[28] L.-P. Chang, T.-W. Kuo, and S.-W. Lo, "Real-time garbage collection for flash-memory storage systems of real-time embedded systems," *TECS*, 2004.

[29] C. Chen, A. Schrott, M. H. Lee, S. Raoux, Y. H. Shih, M. Breitwisch, F. H. Baumann, E. K. Lai, T. M. Shaw, P. Flaitz, R. Cheek, E. A. Joseph, S. H. Chen, B. Rajendran, H. L. Lung, and C. Lam, "Endurance improvement of Ge2Sb2Te5-based phase change memory," in *IMW*, 2009.

[30] J. Chen, Z. Winter, G. Venkataramani, and H. H. Huang, "rpram: Exploring redundancy techniques to improve lifetime of PCM-based main memory," in *PACT*, 2011.

[31] J. Chen, R. C. Chiang, H. H. Huang, and G. Venkataramani, "Energy-aware writes to non-volatile main memory," *OSR*, 2012.

[32] S. Chhabra and Y. Solihin, "i-NVMM: A secure non-volatile main memory system with incremental encryption," in *ISCA*, 2011.

[33] S. Cho and H. Lee, "Flip-N-Write: a simple deterministic technique to improve PRAM write performance, energy and endurance," in *MICRO*, 2009.

[34] J. Collins, S. Sair, B. Calder, and D. M. Tullsen, "Pointer cache assisted prefetching," in *MICRO*, 2002.

[35] G. Dhiman, R. Ayoub, and T. Rosing, "PDRAM: A hybrid PRAM and DRAM main memory system," in *DAC*, 2009.

[36] H. Dogan, F. Hijaz, M. Ahmad, B. Kahne, P. Wilson, and O. Khan, "Accelerating graph and machine learning workloads using a shared memory multicore architecture with auxiliary support for in-hardware explicit messaging," in *IPDPS*, 2017.

[37] Y. Du, M. Zhou, B. R. Childers, D. Mossé, and R. Melhem, "Bit mapping for balanced PCM cell programming," in *ISCA*, 2013.

[38] E. Ebrahimi, O. Mutlu, and Y. N. Patt, "Techniques for bandwidth-efficient prefetching of linked data structures in hybrid prefetching systems," in *HPCA*, 2009.

[39] Y. Fang, H. Li, and X. Li, "SoftPCM: Enhancing energy efficiency and lifetime of phase change memory in video applications via approximate write," in *ATS*, 2012.

[40] A. P. Ferreira, M. Zhou, S. Bock, B. Childers, R. Melhem, and D. Mossé, "Increasing PCM main memory lifetime," in *DATE*, 2010.

[41] P. Frigo, E. Vannacci, H. Hassan, V. van der Veen, O. Mutlu, C. Giuffrida, H. Bos, and K. Razavi, "TRRespass: Exploiting the Many Sides of Target Row Refresh," *S&P*, 2020.

[42] S. Gueron, "Memory encryption for general-purpose processors," *S&P*, 2016.

[43] J. Guerra, L. Mármol, D. Campello, C. Crespo, R. Rangaswami, and J. Wei, "Software persistent memory," in *ATC*, 2012.

[44] A. Gupta, Y. Kim, and B. Urgaonkar, "DFTL: A flash translation layer employing demand-based selective caching of page-level address mappings," in *ASPLOS*, 2009.

[45] T. J. Ham, B. K. Chelepalli, N. Xue, and B. C. Lee, "Disintegrated control for energy-efficient and heterogeneous memory systems," in *HPCA*, 2013.

[46] M. Hashemi, Khubaib, E. Ebrahimi, O. Mutlu, and Y. N. Patt, "Accelerating dependent cache misses with an enhanced memory controller," in *ISCA*, 2016.

[47] M. Henson and S. Taylor, "Memory encryption: A survey of existing techniques," *CSUR*, 2014.

[48] J.-W. Hsieh and Y.-H. Kuan, "Double circular caching scheme for DRAM/PRAM hybrid cache," in *RTCSA*, 2012.

[49] K. Hsieh, S. Khan, N. Vijaykumar, K. K. Chang, A. Boroumand, S. Ghose, and O. Mutlu, "Accelerating pointer chasing in 3D-stacked

memory: Challenges, mechanisms, evaluation," in *ICCD*, 2016.
[50] J. Hu, C. J. Xue, Q. Zhuge, W.-C. Tseng, and E. H.-M. Sha, "Write activity reduction on non-volatile main memories for embedded chip multiprocessors," *TECS*, 2013.
[51] Y. Huang, T. Liu, and C. J. Xue, "Register allocation for write activity minimization on non-volatile main memory," in *ASP-DAC*, 2011.
[52] W. Hwang and K. H. Park, "Hmmsched: Hybrid main memory-aware task scheduling on multicore systems," in *Future Computing*, 2013.
[53] D. H. Im, J. I. Lee, S. L. Cho, H. G. An, D. H. Kim, I. S. Kim, H. Park, D. H. Ahn, H. Horii, S. O. Park, U. Chung, and J. T. Moon, "A unified 7.5nm dash-type confined cell for high performance PRAM device," in *IEDM*, 2008.
[54] A. N. Jacobvitz, R. Calderbank, and D. J. Sorin, "Coset coding to extend the lifetime of memory," in *HPCA*, 2013.
[55] Y. Jia, F. Zhou, X. Gao, S. Wu, H. Jin, X. Liao, and P. Yuan, "VAIL: A victim-aware cache policy for improving lifetime of hybrid memory," *Parallel Computing*, 2018.
[56] L. Jiang, Y. Zhang, B. R. Childers, and J. Yang, "FPB: Fine-grained power budgeting to improve write throughput of multi-level cell phase change memory," in *MICRO*, 2012.
[57] L. Jiang, Y. Du, B. Zhao, Y. Zhang, B. R. Childers, and J. Yang, "Hardware-assisted cooperative integration of wear-leveling and salvaging for phase change memory," *TACO*, 2013.
[58] L. Jiang, Y. Zhang, and J. Yang, "Mitigating write disturbance in super-dense phase change memories," in *DSN*, 2014.
[59] S. H. Jo and W. Lu, "CMOS compatible nanoscale nonvolatile resistance switching memory," *Nano Letters*, 2008.
[60] Y. Joo, D. Niu, X. Dong, G. Sun, N. Chang, and Y. Xie, "Energy-and endurance-aware design of phase change memory caches," in *DATE*, 2010.
[61] M. J. Kang, T. J. Park, Y. W. Kwon, D. H. Ahn, Y. S. Kang, H. Jeong, S. J. Ahn, Y. J. Song, B. C. Kim, S. W. Nam, H. K. Kang, G. T. Jeong, and C. H. Chung, "PRAM cell technology and characterization in 20nm node size," in *IEDM*, 2011.
[62] U. Kang, H.-s. Yu, C. Park, H. Zheng, J. Halbert, K. Bains, S. Jang, and J. S. Choi, "Co-architecting controllers and DRAM to enhance DRAM process scaling," in *The Memory Forum*, 2014.
[63] S. Kannan, M. Qureshi, A. Gavrilovska, and K. Schwan, "Energy aware persistence: Reducing energy overheads of memory-based persistence in NVMs," in *PACT*, 2016.
[64] M. Karlsson, F. Dahlgren, and P. Stenstrom, "A prefetching technique for irregular accesses to linked data structures," in *HPCA*, 2000.
[65] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A space-efficient flash translation layer for CompactFlash systems," *TCE*, 2002.
[66] Y. Kim, V. Seshadri, D. Lee, J. Liu, and O. Mutlu, "A case for exploiting subarray-level parallelism (SALP) in DRAM," in *ISCA*, 2012.
[67] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai, and O. Mutlu, "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors," in *ISCA*, 2014.
[68] Y. Kim, W. Yang, and O. Mutlu, "Ramulator: A fast and extensible DRAM simulator." *CAL*, 2016.
[69] N. Kohout, S. Choi, D. Kim, and D. Yeung, "Multi-chain prefetching: Effective exploitation of inter-chain memory parallelism for pointer-chasing codes," in *PACT*, 2001.
[70] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating STT-RAM as an energy-efficient main memory alternative," in *ISPASS*, 2013.
[71] M. Kund, G. Beitel, C.-U. Pinnow, T. Rohr, J. Schumann, R. Symanczyk, K. Ufert, and G. Muller, "Conductive bridging RAM (CBRAM): An emerging non-volatile memory technology scalable to sub 20nm," in *IEDM*, 2005.
[72] S. Lai, "Current status of the phase change memory and its future," in *IEDM*, 2003.
[73] S. Lai, "Non-volatile memory technologies: The quest for ever lower cost," in *IEDM*, 2008.
[74] A. Lalam and A. C. Shen, "Non-volatile dual in-line memory module (NVDIMM) multichip package," *US Patent 10,199,364*, 2019.
[75] C. H. Lam and H.-L. Lung, "Block erase for phase change memory," in *US Patent 7,755,935*, 2010.
[76] B. C. Lee, P. Zhou, J. Yang, Y. Zhang, B. Zhao, E. Ipek, O. Mutlu, and D. Burger, "Phase-change technology and the future of main memory," *IEEE Micro*, 2010.
[77] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable DRAM alternative," in *ISCA*, 2009.
[78] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Phase change memory architecture and the quest for scalability," *CACM*, 2010.
[79] C. J. Lee, V. Narasiman, E. Ebrahimi, O. Mutlu, and Y. N. Patt, "DRAM-aware last-level cache writeback: Reducing write-caused interference in memory systems," 2010.
[80] J. I. Lee, H. Park, S. L. Cho, Y. L. Park, B. J. Bae, J. H. Park, J. S. Park, H. G. An, J. S. Bae, D. H. Ahn, Y. T. Kim, H. Horii, S. A. Song, J. C. Shin, S. O. Park, H. S. Kim, U. Chung, J. T. Moon, and B. I. Ryu, "Highly scalable phase change memory with CVD GeSbTe for sub 50nm generation," in *VLSI Technology*, 2007.
[81] Y. Li, S. Ghose, J. Choi, J. Sun, H. Wang, and O. Mutlu, "Utility-based hybrid memory management," in *CLUSTER*, 2017.
[82] Y. Y. Lin, H. B. Lv, P. Zhou, M. Yin, F. F. Liao, Y. F. Cai, T. A. Tang, J. Feng, Y. Zhang, Z. F. Zhang, B. W. Qiao, Y. F. Lai, B. C. Cai, and B. Chen, "Nano-crystalline phase change memory with composite Si-Sb-Te film for better data retention and lower operation current," in *NVSMW*, 2007.
[83] S. Liu, A. Kolli, J. Ren, and S. Khan, "Crash consistency in encrypted non-volatile main memory systems," in *HPCA*, 2018.
[84] Y. Lu, J. Shu, L. Sun, and O. Mutlu, "Loose-ordering consistency for persistent memory," in *ICCD*, 2014.
[85] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *PLDI*, 2005.
[86] H. Lung, C. P. Miller, C. Chen, S. C. Lewis, J. Morrish, T. Perri, R. C. Jordan, H. Ho, T. Chen, W. Chien, M. Drapa, T. Maffitt, J. Heath, Y. Nakamura, J. Okazawa, K. Hosokawa, M. BrightSky, R. Bruce, H. Cheng, A. Ray, Y. Ho, C. Yeh, W. Kim, S. Kim, Y. Zhu, and C. Lam, "A double-data-rate 2 (DDR2) interface phase-change memory with 533MB/s read-write data rate and 37.5ns access latency for memory-type storage class memory applications," in *IMW*, 2016.
[87] D. Ma, J. Feng, and G. Li, "A survey of address translation technologies for flash memories," *CSUR*, 2014.
[88] A. Mallik, D. Garbin, A. Fantini, D. Rodopoulos, R. Degraeve, J. Stuijt, A. K. Das, S. Schaafsma, P. Debacker, G. Donadio, H. Hody, L. Goux, G. S. Kar, A. Furnemont, A. Mocuta, and P. Raghavan, "Design-technology co-optimization for OxRRAM-based synaptic processing unit," in *VLSI Technology*, 2017.
[89] J. A. Mandelman, R. H. Dennard, G. B. Bronner, J. K. DeBrosse, R. Divakaruni, Y. Li, and C. J. Radens, "Challenges and future directions for the scaling of dynamic random-access memory (DRAM)," *IBM JRD*, 2002.
[90] V. J. Marathe, M. Seltzer, S. Byan, and T. Harris, "Persistent memcached: Bringing legacy code to byte-addressable persistent memory," in *HotStorage*, 2017.
[91] J. Meza, J. Chang, H. Yoon, O. Mutlu, and P. Ranganathan, "Enabling efficient and scalable hybrid memories using fine-granularity DRAM cache management," *CAL*, 2012.
[92] J. Meza, Y. Luo, S. Khan, J. Zhao, Y. Xie, and O. Mutlu, "A case for efficient hardware/software cooperative management of storage and memory," in *WEED*, 2013.
[93] A. Mirhoseini, M. Potkonjak, and F. Koushanfar, "Coding-based energy minimization for phase change memory," in *DAC*, 2012.
[94] T. Morikawa, K. Kurotsuchi, M. Kinoshita, N. Matsuzaki, Y. Matsui, Y. Fujisaki, S. Hanzawa, A. Kotabe, M. Terao, H. Moriya, T. Iwasaki,

M. Matsuoka, F. Nitta, M. Moniwa, T. Koga, and N. Takaura, "Doped In-Ge-Te phase change memory featuring stable operation and good data retention," in *IEDM*, 2007.

[95] O. Mutlu, "Memory scaling: A systems architecture perspective," in *IMW*, 2013.

[96] O. Mutlu, "The RowHammer problem and other issues we may face as memory becomes denser," in *DATE*, 2017.

[97] O. Mutlu and J. S. Kim, "Rowhammer: A retrospective," *TCAD*, 2019.

[98] O. Mutlu and L. Subramanian, "Research problems and opportunities in memory systems," *Supercomputing Frontiers and Innovations*, 2015.

[99] L. Nai, R. Hadidi, J. Sim, H. Kim, P. Kumar, and H. Kim, "Graphpim: Enabling instruction-level pim offloading in graph computing frameworks," in *HPCA*, 2017.

[100] J. H. Oh, J. H. Park, Y. S. Lim, H. S. Lim, Y. T. Oh, J. S. Kim, J. M. Shin, J. H. Park, Y. J. Song, K. C. Ryoo, D. W. Lim, S. S. Park, J. I. Kim, J. H. Kim, J. Yu, F. Yeung, C. W. Jeong, J. H. Kong, D. H. Kang, G. H. Koh, G. T. Jeong, H. S. Jeong, and K. Kim, "Full integration of highly manufacturable 512Mb PRAM based on 90nm technology," in *IEDM*, 2006.

[101] S. R. Ovshinsky, "Reversible electrical switching phenomena in disordered structures," *PRL*, 1968.

[102] L. Peled, U. Weiser, and Y. Etsion, "A neural network prefetcher for arbitrary memory access patterns," *TACO*, 2019.

[103] A. Pirovano, A. Redaelli, F. Pellizzer, F. Ottogalli, M. Tosi, D. Ielmini, A. L. Lacaita, and R. Bez, "Reliability study of phase-change nonvolatile memories," *TDMR*, 2004.

[104] M. Poremba, T. Zhang, and Y. Xie, "Nvmain 2.0: A user-friendly memory simulator to model (non-) volatile memory systems," *CAL*, 2015.

[105] B. Pourshirazi, M. V. Beigi, Z. Zhu, and G. Memik, "WALL: A writeback-aware LLC management for PCM-based main memory systems," in *DATE*, 2018.

[106] M. K. Qureshi, "Pay-As-You-Go: Low-overhead hard-error correction for phase change memories," in *MICRO*, 2011.

[107] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *ISCA*, 2009.

[108] M. K. Qureshi, M. M. Franceschini, L. A. Lastras-Montaño, and J. P. Karidis, "Morphable memory system: A robust architecture for exploiting multi-level phase change memories," in *ISCA*, 2010.

[109] M. K. Qureshi, M. M. Franceschini, and L. A. Lastras-Montano, "Improving read performance of phase change memories via write cancellation and write pausing," in *HPCA*, 2010.

[110] M. K. Qureshi, M. M. Franceschini, A. Jagmohan, and L. A. Lastras, "PreSET: Improving performance of phase change memories by exploiting asymmetry in write times," in *ISCA*, 2012.

[111] J. Ren, J. Zhao, S. Khan, J. Choi, Y. Wu, and O. Mutiu, "ThyNVM: Enabling software-transparent crash consistency in persistent memory systems," in *MICRO*, 2015.

[112] S. K. Sadasivam, B. W. Thompto, R. Kalla, and W. J. Starke, "IBM Power9 processor architecture," *IEEE Micro*, 2017.

[113] G. Saileshwar, P. Nair, P. Ramrakhyani, W. Elsasser, J. Joao, and M. Qureshi, "Morphable counters: Enabling compact integrity trees for low-overhead secure memories," in *MICRO*, 2018.

[114] G. Saileshwar, P. J. Nair, P. Ramrakhyani, W. Elsasser, and M. K. Qureshi, "Synergy: Rethinking secure-memory design for error-correcting memories," in *HPCA*, 2018.

[115] M. Salinga, B. Kersting, I. Ronneberger, V. P. Jonnalagadda, X. T. Vu, M. Le Gallo, I. Giannopoulos, O. Cojocaru-Mirédin, R. Mazzarello, and A. Sebastian, "Monatomic phase change memory," *Nature Materials*, 2018.

[116] A. Sebastian, T. Tuma, N. Papandreou, M. Le Gallo, L. Kull, T. Parnell, and E. Eleftheriou, "Temporal correlation detection using computational phase-change memory," *Nature Communications*, 2017.

[117] J. Secco, F. Corinto, and A. Sebastian, "Flux–charge memristor model for phase change memory," *TCAS II: Express Briefs*, 2018.

[118] N. H. Seong, D. H. Woo, and H.-H. S. Lee, "Security Refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping," in *ISCA*, 2010.

[119] V. Seshadri, A. Bhowmick, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "The dirty-block index," in *ISCA*, 2014.

[120] L. Song, Y. Zhuo, X. Qian, H. Li, and Y. Chen, "GraphR: Accelerating graph processing using ReRAM," in *HPCA*, 2018.

[121] S. Song, A. Das, O. Mutlu, and N. Kandasamy, "Enabling and exploiting partition-level parallelism (PALP) in phase change memories," *TECS*, 2019.

[122] J. Stuecheli, D. Kaseridis, D. Daly, H. C. Hunter, and L. K. John, "The virtual write queue: Coordinating DRAM and last-level cache policies," in *ISCA*, 2010.

[123] S. Swami, J. Rakshit, and K. Mohanram, "Secret: Smartly encrypted energy efficient non-volatile memories," in *DAC*, 2016.

[124] C. Villa, "PCM array architecture and management," in *Phase Change Memory*, 2018.

[125] C. Villa, D. Mills, G. Barkley, H. Giduturi, S. Schippers, and D. Vimercati, "A 45nm 1Gb 1.8 V phase-change memory," in *ISSCC*, 2010.

[126] H. Volos, A. J. Tack, and M. M. Swift, "Mnemosyne: Lightweight persistent memory," in *ASPLOS*, 2011.

[127] R. Wang, L. Jiang, Y. Zhang, L. Wang, and J. Yang, "Exploit imbalanced cell writes to mitigate write disturbance in dense phase change memory," in *DAC*, 2015.

[128] Y. Wang, Y. Zheng, G. Liu, T. Li, T. Guo, Y. Cheng, S. Lv, S. Song, K. Ren, and Z. Song, "Scandium doped Ge2Sb2Te5 for high-speed and low-power-consumption phase change memory," *APL*, 2018.

[129] Z. Wang, S. Shan, T. Cao, J. Gu, Y. Xu, S. Mu, Y. Xie, and D. A. Jiménez, "WADE: Writeback-aware dynamic cache management for NVM-based main memory system," *TACO*, 2013.

[130] Z. Wang and T. Nowatzki, "Stream-based memory access specialization for general purpose processors," in *ISCA*, 2019.

[131] M. V. Wilkes, "The memory gap and the future of high performance memories," *Computer Architecture News*, 2001.

[132] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase change memory," *Proceedings of the IEEE*, 2010.

[133] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–oxide RRAM," *Proceedings of the IEEE*, 2012.

[134] M. Wuttig and N. Yamada, "Phase-change materials for rewriteable data storage," *Nature Materials*, 2007.

[135] F. Xia, D. Jiang, J. Xiong, M. Chen, L. Zhang, and N. Sun, "DWC: Dynamic write consolidation for phase change memory systems," in *ICS*, 2014.

[136] N. Yamada, R. Kojima, T. Nishihara, A. Tsuchino, Y. Tomekawa, and H. Kusada, "100GB rewritable triple-layer optical disk having Ge-Sb-Te films," in *Phase-Change and Ovonics Symposium*, 2009.

[137] B.-D. Yang, J.-E. Lee, J.-S. Kim, J. Cho, S.-Y. Lee, and B.-G. Yu, "A low power phase-change random access memory using a data-comparison write scheme," in *ISCAS*, 2007.

[138] J. Yang, L. Gao, and Y. Zhang, "Improving memory encryption performance in secure processors," *TC*, 2005.

[139] M. Ye, C. Hughes, and A. Awad, "Osiris: A low-cost mechanism to enable restoration of secure non-volatile memories," in *MICRO*, 2018.

[140] H. Yoon, J. Meza, R. Ausavarungnirun, R. A. Harding, and O. Mutlu, "Row buffer locality aware caching policies for hybrid memories," in *ICCD*, 2012.

[141] H. Yoon, J. Meza, N. Muralimanohar, N. P. Jouppi, and O. Mutlu, "Efficient data mapping and buffering techniques for multilevel cell phase-change memories," *TACO*, 2014.

[142] J. Yue and Y. Zhu, "Accelerating write by exploiting PCM asymmetries," in *HPCA*, 2013.

[143] L. Zhang, B. Neely, D. Franklin, D. Strukov, Y. Xie, and F. T. Chong, "Mellow writes: Extending lifetime in resistive memories through selective slow write backs," in *ISCA*, 2016.
[144] T. Zhang, Z. Song, F. Wang, B. Liu, S. Feng, and B. Chen, "Te-free SiSb phase change material for high data retention phase change memory application," *JJAP*, 2007.
[145] Y. Zhang, J. Yang, A. Memaripour, and S. Swanson, "Mojim: A reliable and highly-available non-volatile memory system," in *ASPLOS*, 2015.
[146] J. Zhao, O. Mutlu, and Y. Xie, "FIRM: Fair and high-performance memory control for persistent memory systems," in *MICRO*, 2014.
[147] M. Zhou, Y. Du, B. Childers, R. Melhem, and D. Mossé, "Writeback-aware partitioning and replacement for last-level caches in phase change main memory systems," *TACO*, 2012.
[148] M. Zhou, Y. Du, B. R. Childers, R. Melhem, and D. Mosse, "Writeback-aware bandwidth partitioning for multi-core systems with PCM," in *PACT*, 2013.
[149] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *ISCA*, 2009.