Xia F, Jiang DJ, Xiong J *et al.* A survey of phase change memory systems. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 30(1): 121–144 Jan. 2015. DOI 10.1007/s11390-015-1509-2

A Survey of Phase Change Memory Systems

Fei Xia[1,2] (夏 飞), *Student Member, CCF, ACM, IEEE*, De-Jun Jiang[1] (蒋德钧), *Member, CCF, ACM, IEEE*
Jin Xiong[1] (熊 劲), *Senior Member, CCF, Member, ACM, IEEE*, and
Ning-Hui Sun[1](孙凝晖), *Fellow, CCF, Member, ACM, IEEE*

[1] *State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences Beijing 100190, China*

[2] *University of Chinese Academy of Sciences, Beijing 100049, China*

E-mail: {xiafei2011, jiangdejun, xiongjin, snh}@ict.ac.cn

Received April 17, 2014; revised September 17, 2014.

Abstract As the scaling of applications increases, the demand of main memory capacity increases in order to serve large working set. It is difficult for DRAM (dynamic random access memory) based memory system to satisfy the memory capacity requirement due to its limited scalability and high energy consumption. Compared to DRAM, PCM (phase change memory) has better scalability, lower energy leakage, and non-volatility. PCM memory systems have become a hot topic of academic and industrial research. However, PCM technology has the following three drawbacks: long write latency, limited write endurance, and high write energy, which raises challenges to its adoption in practice. This paper surveys architectural research work to optimize PCM memory systems. First, this paper introduces the background of PCM. Then, it surveys research efforts on PCM memory systems in performance optimization, lifetime improving, and energy saving in detail, respectively. This paper also compares and summarizes these techniques from multiple dimensions. Finally, it concludes these optimization techniques and discusses possible research directions of PCM memory systems in future.

Keywords phase change memory, memory system, performance, lifetime, energy

1 Introduction

DRAM (dynamic random access memory) has been used as main memory for the last decades. As the scaling of applications (such as bioinformatics and search engine) increases, the demand of memory capacity increases in order to serve increasing working sets. On the other hand, the number of concurrent running applications on individual servers increases as the number of processor cores increases, which also results in the increasing demand of memory capacity. However, DRAM technology is facing two issues: energy and scalability. First, DRAM uses capacitor to store data, and thus DRAM needs to refresh periodically to avoid the leakage of capacitor. This increases the energy consumption of DRAM memory. Generally, current DRAM memory system consumes 20% to 40% energy of the total server energy[1–3]. Second, it is difficult for DRAM to scale down to 20 nm due to limitations, such as capacitor placement, device leakage, and chip packaging①. Therefore, DRAM memory system incurs excessive cost to satisfy the increasing memory demands of applications.

Recently, emerging non-volatile memories (NVMs) have shown potential to be adopted as main memory, such as phase change memory (PCM), resistive random access memory (RRAM), and magnetoresistive random access memory (MRAM). NVMs are expected to have better scalability, lower energy consumption, and comparable performance with DRAM. PCM is a promising technology among these NVMs[4–7]. First, PCM technology has better scalability than DRAM. For example,

Survey

This work was supported by the National Basic Research 973 Program of China under Grant No. 2011CB302502, the National Natural Science Foundation of China under Grant No. 61379042, Huawei Research Program under Grant No. YB2013090048, and the Strategic Priority Research Program of Chinese Academy of Sciences under Grant No. XDA06010401.

① International technology roadmap for semiconductors (ITRS 2013). http://www.itrs.net/Links/2013ITRS/Home2013.htm, Nov. 2014.

a PCM device prototype with 3 nm × 20 nm has been fabricated and tested[8]. Second, the density of PCM is 2 to 4 times higher than that of DRAM. Thus, it is possible to produce PCM chip with larger capacity than DRAM chip. At last, since PCM does not need to refresh periodically, the static energy consumption of PCM is also lower than that of DRAM. PCM has been evaluated as main memory in context of GPUs, embedded systems, real-time systems, video applications and so on[9].

Unfortunately, compared to DRAM, PCM technology has three major drawbacks: longer write latency, limited write endurance, and higher write energy, which challenges its adoption as main memory. First, the write latency of PCM is longer than that of DRAM, which affects the performance of memory system. Second, the write endurance of PCM is limited. PCM cells can be worn out after a large number of writes, which shortens the lifetime of PCM memory. At last, the write energy of PCM is apparently higher than that of DRAM. The high write energy not only degrades write performance, but also increases system running cost. Therefore, it is vital to address these issues of PCM for its practical usage. A large number of techniques have been proposed to optimize PCM memory system from the perspectives of device, architecture, and system.

There are already a few studies that summarize the optimization techniques of PCM memory systems. Querish *et al.*[10] investigated architectural techniques to enable PCM for main memory. Zilberberg *et al.*[11] surveyed architecture and system level techniques when PCM is used to replace DRAM and NAND flash. Mittal[9] listed energy saving techniques for PCM memory systems. The studies [12] and [13] summarize device-level optimization techniques for PCM devices. However, these researches were finished several years ago, and cannot reflect the latest research progress on PCM memory systems. Moreover, these researches focus on introducing each technique and lack classification, comparison, and analysis over these techniques.

In this paper, we survey the *architecture-level* techniques of PCM memory systems to address PCM's issues and enable its actual adoption. This paper makes the following contributions.

1) We survey architectural techniques for PCM memory systems from the perspectives of performance, lifetime, and energy, respectively. Moreover, We reveal the advantages and limitations of these techniques by classifying and comparing them from multiple dimensions.

2) We propose a few possible research directions for the future work of PCM memory systems. Hybrid DRAM/PCM memory system and PCM-based storage are promising usage models of PCM. On the other hand, optimizing the performance of persistent memory system also requires more research efforts.

The remainder of this paper is organized as follows. Section 2 presents the background of PCM memory. Section 3 surveys the techniques to address the long write latency to improve PCM memory systems performance. Section 4 summarizes the techniques to overcome the limited write endurance to improve the lifetime of PCM memory systems. Section 5 describes the techniques to reduce the energy consumption of PCM memory systems. Section 6 concludes this paper and discusses possible directions of future research.

2 Phase Change Memory

2.1 Physical Mechanism

Phase change memory (PCM) is a type of non-volatile memory that exploits the phase change property of chalcogenide glasses to store bit information[14-15]. Although the principle of using phase change material to store data was proven in 1968[16], the lack of suitable material with low energy consumption prevents its actual usage. With the discovery and development of crystallizing material, such as Ge2Sb2Te5 (GST), phase change technology renews industry interest including IBM, Micron, Samsung and so on. For example, IBM produced a PCM device prototype in 2008. Micron announced in massive production of 1 Gbit PCM chips in 2012②. Samsung produced an 8 Gbit PCM chip using 20 nm process in 2012[17].

Fig. 1(a) shows the structure of a conventional PCM cell, which consists of top and bottom electrodes, physical change material and heater[18]. Phase change material can switch back and forth between two states: amorphous state that has high resistance and polycrystalline state that has low resistance. PCM utilizes the resistance difference to store bit information. Fig.1(b) shows the read and write mechanisms of PCM technology[18]. To RESET (writing bit "0") a PCM cell, a short but high voltage pulse is applied to heat the

② Micron announces availability of phase change memory for mobile devices. http://investors.micron.com/releasedetail.cfm?ReleaseID=692563, Nov. 2014.

phase change material and switch it from the polycrystalline state to the amorphous state. To SET (writing bit "1") a PCM cell, a sustained but low voltage pulse is applied to switch the material back to the polycrystalline state. Therefore, RESET takes short latency and consumes high power, while SET takes long latency and consumes low power. The energy consumption of RESET is also higher than that of SET. To read the state of phase change material, a low enough voltage pulse is applied to the material. The bit information is distinguished according to the current difference. Both the read latency and the read energy of PCM are low.

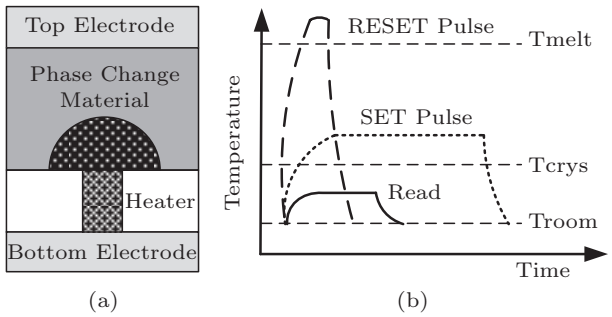


Fig.1. (a) PCM cell and (b) its read and write mechanisms[18].

The large resistance difference between the amorphous state and the polycrystalline state makes it possible to store multiple bits per PCM cell, which is called multi-level cell (MLC). Fig.2 shows a typical 2-bit MLC that has four states. "10" and "01" are in the intermediate state compared with "11" and "00".

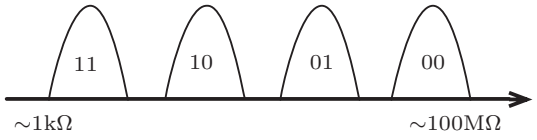


Fig.2. 2-bit MLC with 4 states.

The *iterative programming* technique (or called *programming-and-verify*) is usually used to write MLC PCM[19-20], as shown in Fig.3[21]. After every write, a read is used to decide whether to stop or continue to write. If the resistance does not reach the target range, then the controller calculates new programming pulse and continues to write. This process iterates multiple times until the resistance of MLC reaches the target range. Therefore, the write latency and the write energy of MLC PCM are higher than those of SLC PCM.

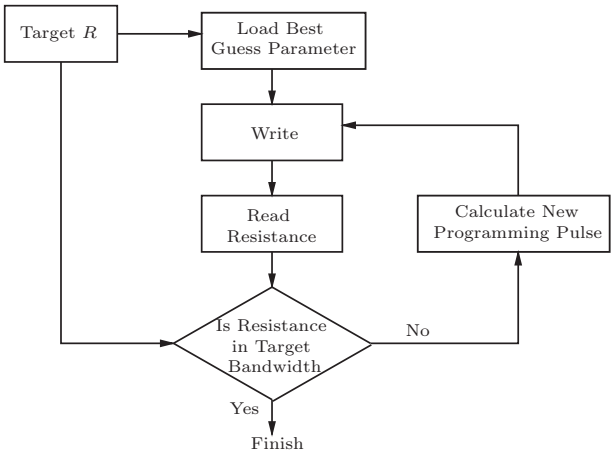


Fig.3. Iterative programming of MLC PCM[21].

2.2 Comparison with Other Storage and Memory Technologies

Table 1 compares the characteristics of PCM with those of other storage and memory technologies, including HDD, NAND flash, DRAM, and SRAM. Note that the parameters of PCM are varied among different prototypes and literatures. The exact parameters are affected by many factors, such as the material and the process. The parameters in Table 1 derive from widely cited studies③[4].

The feature size of PCM can be scaled down to 8 nm④, which is smaller than those of other technologies. Thus, PCM has better scalability than other technologies. The read and the write latencies of PCM are much longer than those of DRAM, especially the write latency. Therefore, it is important to improve write performance when PCM is adopted in memory. Compared to NAND flash and HDD, both the read and the write latencies of PCM are shorter by several orders of magnitude. Using PCM in storage systems can greatly improve the storage system performance. For endurance, PCM cell can only endure 10^9 writes on average. Although the write endurance is larger than that of NAND flash, it is much smaller than 10^{16} of HDD, DRAM, and SRAM. It is essential to address the endurance issue of PCM to construct a stable system. The write energy of PCM is higher than that of other technologies. Reducing write energy is important

③ International technology roadmap for semiconductors (ITRS 2012). http://www.itrs.net/Links/2012ITRS/2012Tables/PIDS_2012Tables.xlsx, Dec. 2014.

④ The feature size of PCM is projected to be 8 nm in future according to ITRS2013.

Table 1. Characteristics Comparison of Memory and Storage Technologies

Storage/Memory	Feature Size (nm)	Read Latency	Write Latency	Endurance	Write Energy (J/b)	Volatility
HDD	–	3 ms∼5 ms	3 ms∼5 ms	> 1E+16	2E-16	No
NAND flash	16	35 µs⑤	350 µs⑤	1E+5	>2E-16	No
PCM	8[2]	55 ns[4]	150 ns[4]	1E+9	6E-12	No
DRAM	20⑥	<10 ns	<10 ns	>1E+16	4E-15	Yes
SRAM	10	0.2 ns	0.2 ns	>1E+16	5E-16	Yes

to build an energy-efficient PCM system. DRAM and SRAM are volatile, while PCM, HDD and NAND flash are non-volatile. Thus, PCM can be used to store persistent data.

2.3 Chip-Level Write Process of PCM

The write latency of PCM in Table 1 is called *programming latency* or *cell-level write latency*, which is the latency to write (or program) a PCM cell. Actually, the latency to write a data block to a PCM chip, which is called *chip-level write latency*, is much longer than programming latency. In this subsection, we show the chip-level write process of PCM to present the detail of chip-level write latency.

For DRAM, the whole row buffer data can be written back to memory array once. However, the PCM chip has the maximum power constraint, which is determined by the area of its charge pump[22]. The PCM chip only supports a limited number of bits to be written concurrently due to high write power and the maximum power constraint[23-24]. The number of bits that can be written to a PCM chip concurrently is called *writing bits parallelism* in this paper. Fig.4 shows the latency to write a 64-byte data to eight PCM chips, which is called *write command latency*⑦. Assuming the maximum writing bits parallelism is 8, then only 64 bits data can be written to eight PCM chips concurrently, which takes one programming latency. Writing a 64-byte data needs eight times of programming latency. For this example, the write command latency is eight times of the programming latency.

Actually, the resistance of a PCM cell still increases with time after a write operation is finished, which is known as resistance drift. The resistance drift latency is typically tens of microseconds. The recently written data can only be accessed after the resistance is stable[25]. Thus, the chip-level write latency is the sum of write command latency and the drift latency. As a result, the chip-level write latency of PCM is apparently longer than the cell-level write latency. Taking Micron's first-generation PCM chip (128Mb: P8P Parallel PCM) as an example, writing 64 bytes data typically takes 120 µs⑧.

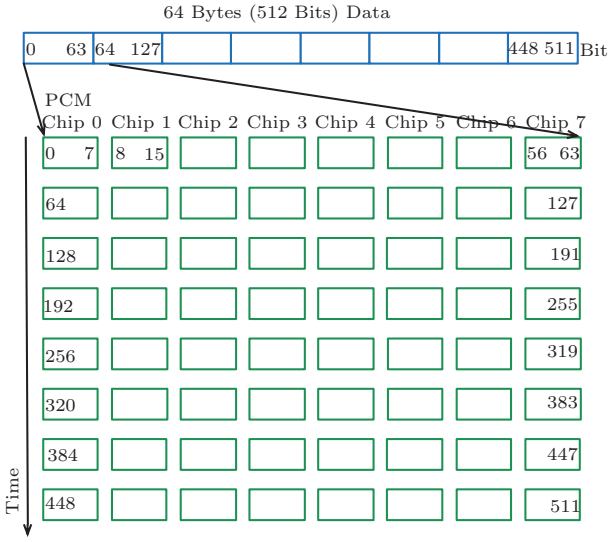


Fig.4. Write command latency. Assuming each chip can support 8 bits writing concurrently (the writing bits parallelism is 8), then writing 64-byte data to 8 chips needs 8 times of programming latency.

2.4 PCM Memory System

As shown in Table 1, the read latency and the cell-level write latency of PCM cell are comparable to those of DRAM. Thus, a number of research literatures explore to adopt PCM as main memory. Although some studies do not consider the chip-level write latency of

⑤ Micron. SLC NAND flash products. http://www.micron.com/products/nand-flash/slc-nand#fullPart, Dec. 2014.

⑥ Samsung computing DRAM. http://www.samsung.com/global/business/semiconductor/product/computing-dram/overview, Nov. 2014.

⑦ In memory architecture, a cache line data usually spans multiple memory chips, which can be operated in parallel. A write command writes a cache line data to memory.

⑧ Micron. P8P parallel phase change memory (PCM). http://www.micron.com/∼/media/documents/products/data-sheet/pcm/p8p_parallel_pcm_ds.pdf, Dec. 2014.

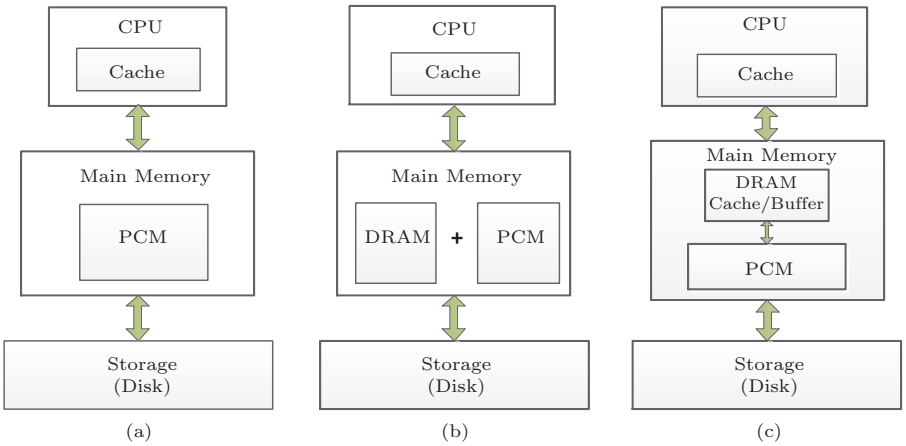


Fig.5. Three categories of PCM memory system architecture. (a) Replacing DRAM. (b) Parallel hybrid memory. (c) Stratified hybrid memory.

PCM, they still provide valuable explorations to optimize PCM memory systems. In this paper, we survey architectural optimizations for PCM memory systems.

There are three categories of PCM main memory architecture, as shown in Fig.5. The first one is using PCM to replace DRAM, as shown in Fig.5(a)[4,6,21]. The second one is the parallel hybrid main memory consisting of DRAM and PCM, as shown in Fig.5(b)[26-28]. The physical address spaces of DRAM and PCM are addressable by the processor. The last one is stratified hybrid main memory that DRAM acts as cache or buffer of PCM memory, as shown in Fig.5(c)[5,29-30]. The address space of DRAM is transparent to the operating system (OS). Hybrid main memory allows applications to exploit the advantages of DRAM and PCM.

Compared to DRAM, PCM has advantages in scalability and non-volatility. However, PCM also has three drawbacks: long write latency, limited write endurance, and high write energy. Therefore, optimizing performance, improving lifetime, and saving energy are three key techniques of PCM memory systems[⑨]. In this paper, we survey research on PCM memory system from these three aspects.

3 Performance Optimization Techniques

Memory accessing latency is a major factor that affects memory system performance. Although write is not on the critical path of memory system, serving a write request blocks subsequent read requests that access different lines of the same bank until the write request completes. Therefore, adopting PCM as the main memory without any optimizations degrades the system performance due to the long write latency of PCM, especially the chip-level write latency.

This section surveys the key techniques to overcome the long write latency and optimize the performance of PCM memory system. First, reducing chip-level write latency can directly improve memory system performance. Since resistance drift is the physical property of phase change material, its latency is hard to be reduced in architecture. Therefore, the main technique to reduce chip-level write latency is by reducing write command latency. Second, the increasing parallelism of write commands and hybrid memory techniques can hide the long write latency of PCM, and thus improve performance. Third, reducing the impact of writes on reads can also improve performance, as reads are on the critical path of memory system performance. Fourth, the iterative programming characteristic of MLC PCM provides opportunity to optimize the performance of MLC PCM memory. At last, in addition to store memory data, PCM can also store persistent data utilizing its non-volatility. Differentiating the persistent data from memory data in PCM can improve the performance of persistent memory.

Subsection 3.1 presents the techniques to reduce write command latency. Subsection 3.2 and Subsection 3.3 show the techniques increasing the parallelism of write commands and hybrid main memory, respectively. Subsection 3.4 presents techniques to reduce the impact of writes on reads. Subsection 3.5 describes the techniques to improve MLC PCM memory performance. Subsection 3.6 presents techniques to optimize persistent memory.

[⑨] In this paper, the PCM memory system includes the three architectures as Fig.5 shows.

3.1 Reducing Write Command Latency

The write command latency is the number of bits to write dividing the writing bits parallelism, as explained in Subsection 2.3. Therefore, both reducing the number of writing bits and improving writing bits parallelism can reduce the write command latency. The latency asymmetry of SET and RESET provides opportunity to reduce the write command latency. At last, consolidating multiple write commands into one can also reduce the average write command latency.

3.1.1 Reducing Writing Bits

DRAM always writes the whole row buffer data to the memory array. Reading data from non-volatile PCM array does not destroy the original value. Therefore, the number of writing bits of a write request can be reduced by avoiding writing unmodified bits to PCM chips.

Yang *et al.*[31] proposed data-comparison-write (DCW) to reduce the number of writing bits. DCW only writes the modified data by reading the old data and comparing them with the new data to write before writing. Zhou *et al.*[6] proposed reducing-bit-writes that only write modified bits of row buffer to PCM array, which is similar to DCW. The number of writing bits can be further reduced using Flip-N-Write[32]. Flip-N-Write flips the data to write if the modified bits are larger than half of total bits. Flip-N-Write only writes the modified bits of the flipped data or the original one. Fig.6 shows an example of Flip-N-Write that writes 8-bit data. In Fig.6(a), 2 bits of the data to write (new data) are modified, less than 4 bits. Thus, Flip-N-Write writes the original data. However, 6 bits of new data are modified, more than 4 bits, in Fig.6(b). Thus, Flip-N-Write writes the flipped data. The flip-bit is used to identify whether the data is flipped or not. Flip-N-Write guarantees that at most half data bits are actually written to PCM chips. As such, Flip-N-Write can reduce the write command latency by half of the original latency at least.

3.1.2 Increasing Writing Bits Parallelism

The maximum writing bits parallelism per PCM chip is due to the maximum power constraints. As mentioned in Subsection 2.1, RESET consumes higher power than SET. However, conversional write scheme assumes that writing a bit consumes power as high as RESET, which limits the writing bits parallelism. On the other hand, SET takes longer than RESET. Assuming that writing a bit takes as long as SET also increases write command latency. Yue and Zhu[33] proposed two-stage-write scheme that divides a write into two stages: write-0 stage and write-1 stage. Two-stage-write differentiates RESET and SET by exploiting their latency and power asymmetries. In write-0 stage, all zeros are written at an accelerated speed. In write-1 stage, all ones are written with increased writing bits parallelism without violating the power constraint. Thus, two-stage-write can reduce the write command latency of PCM. Fig.7 shows an example of two-stage-write. Assume the maximum writing bits parallelism is 2 in the conventional PCM write scheme. Thus, writing 16-bit data to a chip needs 8 times of SET latency. Assuming RESET power is twice of SET power, then one chip can write 4 bits of "1" concurrently. Assume SET latency is 4 times of RESET latency. Two-stage-write only needs 4 times of SET latency to write 16-bit data.

The data of a write request is mapped to multiple chips, as shown in Fig.4. Du *et al.*[34] observed that the distribution of modified bits in different chips is unbalanced under the conventional data mapping policy, which limits the aggregate writing bits parallelism. The write command latency is determined by the chip that needs the largest number of writing bits. Based on this observation, a double XOR mapping (D-XOR)[34] that uses a mapping function to distribute modified

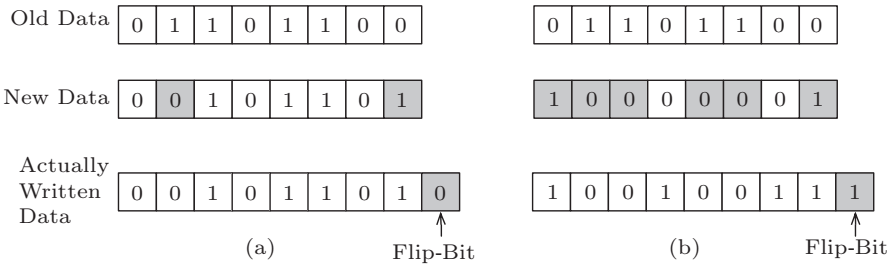


Fig.6. Example of Flip-N-Write. (a) Flip-N-Write writes the originally new data. (b) Flip-N-Write writes the flipped data. The flip-bit identifies whether the data is flipped or not.

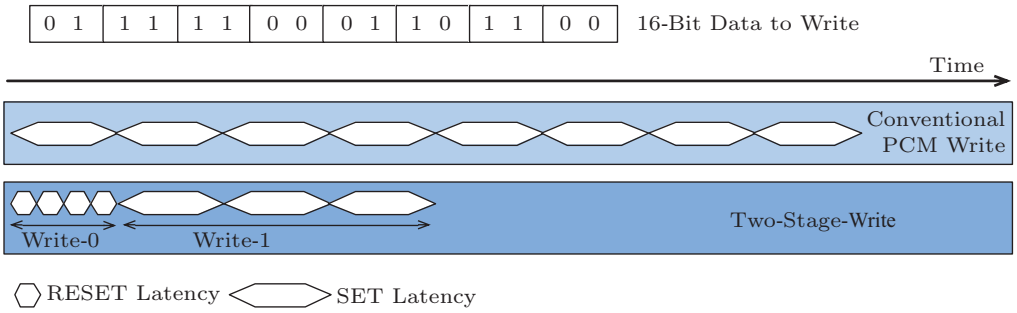


Fig.7. Write command latencies of different write schemes. RESET power is twice of SET power.

bits evenly to chips is proposed to reduce write command latency. As a result, the aggregate writing bits parallelism of multiple chips is increased. Fig.8 shows an example of write command latencies with different mappings. The programming process is divided into two phases: a RESET phase and a SET phase. The maximum writing bits parallelism per chip is 2. Mapping adjacent bits to the same cell group under the conventional mapping scheme limits the aggregate writing bits parallelism, as shown in Fig.8(a). Fig.8(b) shows that mapping adjacent bits to different cell groups increases aggregate writing bits parallelism, and thus reduces write command latency.

3.1.3 *Pro-Actively SET the Memory Line and Write Consolidation*

The programming latency of PCM is asymmetric in that SET takes longer than RESET. On the other hand, when a cache line is written, the corresponding data in memory is invalidated. Based on these two insights, Qureshi *et al.*[35] proposed PreSET to improve the write performance of PCM memory. PreSET proactively sets all the corresponding bits in the memory row immediately after a cache line becomes dirty. Then, PreSET only needs to RESET some bits when actual write is issued. By doing so, write command latency is reduced as RESET takes less time than SET, and the memory performance is improved.

PCM employs burst writes mode to transfer a cache line data to memory. However, the cache line data is partly modified when it is written back to memory. Thus, some burst writes of a write command are wasted to transfer unmodified data. Based on this observation, Xia *et al.*[36] proposed dynamic write consolidation (DWC) to consolidate multiple write commands, and thus improve the performance of PCM memory systems. The key idea of DWC is utilizing the unmodified data burst writes of one write command to transfer the modified data of other write commands. By doing so, multiple write commands can be proposed within one

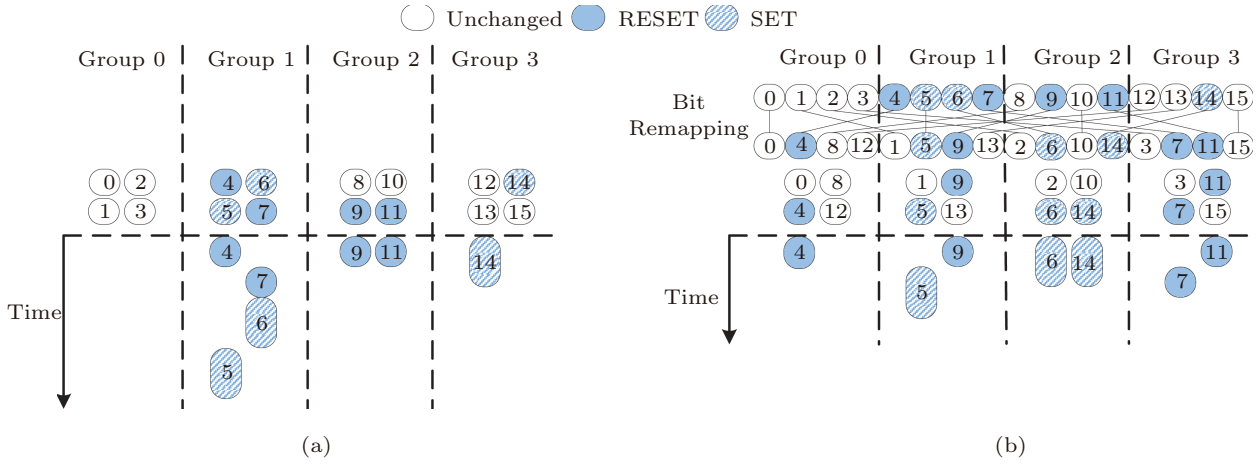


Fig.8. Write command latencies under different mappings[34]. A chip is called a cell group in the figure. (a) Adjacent bits are mapped to the same cell group. (b) Adjacent bits are mapped to different cell groups.

write command latency. As a result, DWC reduces the average write command latency, and thus improves the performance.

3.2 Increasing Write Commands Parallelism

The aforementioned techniques improve PCM memory performance by reducing write command latency. This subsection surveys techniques to increase the parallelism of write commands (called *write commands parallelism*), which can hide the long write latency of PCM to some extent. Larger write commands parallelism can also accelerate the processing of write commands, and thus improve system performance.

The maximum power constraint limits the write commands parallelism. Power-token[37] calculates the actual power requirement of every write command by only writing the modified bits of the data to write. By recording the rest power that every chip can support, power-token can issue more write commands concurrently without violating the power constraint.

Power-token works well for SLC PCM memory, while it does not for MLC PCM memory. First, the iterative write of MLC PCM starts with a RESET pulse and is followed by a varying number of SET pulses. Assuming all iterations consume the same power as RESET is inefficient. Second, one heavily written (hot) PCM chip may block write commands even though other chips are idle. Jiang *et al.*[22] proposed two fine-grained power budgeting (FPB) schemes to address these two issues. FPB-IPM reduces the maximum power of a write command by splitting the first RESET iteration into multiple SET iterations. FPB-GCP integrates a global charge pump on a DIMM to provide extra power for hot chips to support more writes. Both FPB-IPM and FPB-GCP increase the write commands parallelism of MLC PCM memory, and thus improve performance.

The write commands parallelism can also be improved by increasing the parallelism of PCM chips. Although Mini-Rank[38] increases write commands parallelism by dividing a DRAM rank into multiple subranks, it cannot increase the number of parallel PCM chips. This is because conventional electrical bus cannot support a large number of memory chips due to its limitations of insufficient load capacity and signal traversing speed. Li *et al.*[39] proposed OptiPCM to overcome the bus issue by utilizing the photonics to link the memory controller and PCM chips. Ham *et al.*[30] proposed to add memory buffer (MB) in memory to keep the integrity of signal. By doing so, the LR-DIMM[⑩] buffers can be connected by parallel bus structure and extended to two-level buffer structure, as shown in Fig.9[30]. By using two-level 20 memory buffers, 64 memory ranks can share a memory bus.

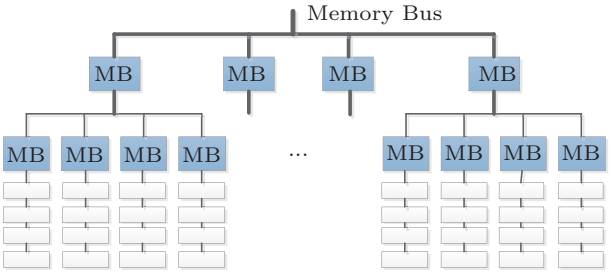


Fig.9. Buffering with hierarchical topology. With 20 memory buffers (gray), 64 memory ranks (white) share a memory bus[30].

3.3 Hybrid Main Memory

Hybrid DRAM/PCM memory system can exploit advantages of both DRAM and PCM. Hybrid memory techniques that place frequently accessed data in DRAM can avoid writes to PCM as many as possible, and thus hide the long write latency of PCM. To utilize the scalability advantage of PCM, hybrid memory consists of DRAM with small capacity and PCM with large capacity.

3.3.1 Parallel Hybrid Memory

For parallel hybrid memory architecture, appropriate data placement can improve memory system performance. The key is estimating the future access pattern of data and then determining its placement.

Zhang and Li[40] proposed OS-level paging scheme that takes page accesses into consideration and migrates hot pages from PCM to DRAM. Ramos *et al.*[27] proposed a page placement policy called "Rank-based Page Placement" (RaPP) in the memory controller. RaPP ranks pages according to their access frequencies and write intensities. Top-ranked pages in PCM are migrated to DRAM. Both [40] and [27] use a modified multi-queue algorithm to identify hot and cold pages in memory controller.

Aforementioned two techniques use memory reads and writes to estimate future access pattern. However, Lee *et al.*[41] observed that using writes alone performs better than using both reads and writes. Besides, the frequency is a better estimator than temporal locality.

⑩ Inphi basic of LRDIMM. http://www.edn.com/design/systems-design/4368420/Basics-of-LRDIMM, Dec. 2014.

Based on the two observations, Lee *et al.*[41] proposed a new page placement policy CLOCK-DWF[41], which is a modified CLOCK algorithm considering the dirty bits and the write frequency. When a write occurs to a PCM page (the page then becomes dirty), CLOCK-DWF migrates it to DRAM. When DRAM is full, a DRAM page is selected using CLOCK-DWF algorithm and replaced to PCM.

The latencies of accesses that are row buffer hits are similar in DRAM and PCM, whereas the latencies of accesses that are row buffer misses are longer in PCM. Based on this observation, Yoon *et al.*[28] proposed row buffer locality-aware (RBLA) mechanism to reduce PCM accesses. RBLA places the data of high row buffer miss rate and frequent accessing in DRAM by recording the count of the row buffer misses of PCM in the memory controller.

Previous techniques migrate data between DRAM and PCM at the page granularity[27,40-41]. However, migrating a page consumes long latency and blocks subsequent memory requests, which affects memory performance. To reduce migration costs, Chelepalli *et al.*[30] proposed fine-grained migration. Instead of migrating data at 4KB page granularity, fine-grained migration moves data at 64B block granularity. To improve bandwidth utilization, Chelepalli *et al.*[30] further proposed to dynamically tune the number of blocks according to successive accesses to one page. If the memory controller accesses successive blocks of one page, it increases blocks for migration. Otherwise, it decreases them.

3.3.2 Stratified Hybrid Memory

For stratified hybrid memory architecture, DRAM cache or buffer can filter data that are accessed frequently, and thus avoid writes to PCM. Lazy-Write[5] was proposed to reduce writes to PCM and thus overcome the long write latency of PCM. When serving a page fault, Lazy-Write fetches page from HDD and only writes it to DRAM cache. This is because the read latency of PCM is similar to that of DRAM. Lee *et al.*[29] proposed a threshold-based DRAM cache that combines the benefits of using DRAM as a write buffer. A part of DRAM data blocks are written back to PCM pro-actively, which guarantees that there are free blocks in DRAM cache to serve requests due to on-chip cache missing. It avoids the long latency to write PCM after on-chip cache missing, which results in improved performance. Since the capacity of DRAM is limited, Meza *et al.*[42] explored fine-granularity DRAM cache management to improve the efficiency of hybrid memory. To accelerate DRAM cache accessing, Meza *et al.*[42] proposed to cache the metadata for recently accessed rows in a small on-chip buffer.

3.4 Reducing Impact of Writes on Reads

Read requests are in the critical path of memory system, which affects memory system performance significantly. Serving a write request blocks subsequent read requests that access different lines of the same bank until the write request completes. The long write latency of PCM aggravates the impact for PCM memory system.

To reduce the impact of writes on reads, Qureshi *et al.*[21] proposed write cancellation and write pausing. Write cancellation cancels the processing of a scheduled write request to serve the pending read request that accesses the same bank. The cancelled write request is scheduled again until the read request completes. Write cancellation increases write time, as well as wastes the time of the partially completed write. By utilizing the iterative writing of MLC PCM, write pausing pauses the write iteration and serves the read when detecting a pending read. The write is resumed from the point where it was paused.

3.5 Improving MLC PCM Performance

Compared to SLC PCM, MLC PCM has higher density and larger capacity, while its programming latency is also longer. The long programming latency degrades memory system performance. This subsection surveys research work that aims to improve MLC PCM memory performance.

MLC PCM adopts an iterative programming technique to write data, as shown in Fig.3. The programming latency of MLC increases as the number of iterations increases. Jiang *et al.*[43] observed that only a small number of cells need significantly more iterations than most of the other cells. However, the programming latency is determined by the cell that needs the maximum iterations. Based on this observation, write-truncation[43] was proposed to accelerate the write process of MLC PCM. When most cells have completed iterations and error correction coding (ECC) can correct the uncompleted cells, write-truncation truncates the uncompleted iterations of cells and finishes the programming.

Besides the number of iterations, the programming latency of MLC PCM is affected by the initial state

of the MLC cell, the target resistance, and process variation (PV). There are several schemes to program MLC PCM, such as PV-aware control programming using staircase down current pluses, programming using increasing reset current pulses. Joshi *et al.*[44] proposed an MLC PCM memory system with low programming latency, called Mercury. Mercury employs adaptive program scheme to reduce programming latency. For example, Mercury selectively uses R2S (RESET-to-SET) and S2R (SET-to-RESET) programming algorithms based on the target resistance level.

Write-truncation and Mercury do not consider the actual memory capacity requirements of applications. Actually, the memory requirement varies from one application to another. It is unnecessary to provide the largest memory capacity at any time. MLC PCM cell can be programmed to two different densities. The high-density cell stores as many bits as those permitted by the technology, which has long latency (called HDPCM). The low-density cell stores half the number of bits but with low latency (called LLPCM). Morphable memory system (MMS)[45] that divides the main memory into two regions was proposed to trade off the capacity and the latency. MMS uses a memory monitoring circuit (MMON) to estimate the capacity requirement of workloads by observing the memory traffic. Then, it periodically tunes the number of pages that must be in LLPCM mode and HDPCM mode.

3.6 Persistent Memory

The non-volatility of PCM makes it possible to store persistent data, called persistent memory. In this subsection, we survey architectural techniques to improve the performance of persistent memory. Note that system-level techniques of persistent memory, such as file system[46-48] and programming model[49-50], are out of the scope of this paper.

The emerging of PCM provides the opportunity of implementing single level storage architecture that PCM acts as memory and storage simultaneously. However, storage needs to guarantee consistency and durability, which limits the performance of persistent memory. Consistency is achieved by ordering write commands. As a result, the scheduler cannot exploit bank-level parallelism of PCM memory that is critical to memory performance. PCM's write command latency is related to the retention time of non-volatility. Durability is achieved by guaranteeing the non-volatility of PCM cell, and thus increases write command latency. Liu *et al.* proposed NVM Duet[51], a unified memory and storage architecture, to improve the performance. NVM Duet guarantees the consistency and the durability for storage and relaxes the constraints for memory. NVM Duet differentiates memory accesses and storage accesses using a new hardware/software interface. The memory scheduler fully exploits bank-level parallelism for memory accesses while guaranteeing the write order for storage accesses. NVM Duet provides dual retention to guarantee the durability for storage while relaxing the retention requirement for memory to reduce write command latency.

3.7 Summary

Table 2 summaries the techniques optimizing the performance of PCM memory systems described in this section.

These techniques improve the performance of PCM memory systems from different perspectives, including reducing the number of writing bits, reducing the number of write commands, increasing parallelism, changing the programming mode, and reducing the impact of writes on reads (*WR-RD impact*), considering persistence. The *memory type* column refers to whether a technique is applicable to SLC, MLC, or both of them (PCM). The *implement* column refers to the main implementation hierarchy of the proposed technique.

These techniques can be applied to PCM memory simultaneously. For example, D-XOR accelerates single write by improving the writing bits parallelism, while power-token accelerates multiple writes by improving write commands parallelism. Among these optimization techniques, DCW is a basic technique for PCM, which has been implemented in PCM chip prototypes[52]. The evaluation in [33] shows that DCW can reduce the number of writing bits by 70% on average. Many techniques are based on DCW, such as Flip-N-Write, D-XOR and power-token. Compared to DCW, Flip-N-Write is a cost-effective technique to further reduce writing bits to PCM.

The long chip-level write latency of PCM makes PCM-only memory systems difficult to satisfy the memory performance requirement of applications. Hybrid DRAM/PCM memory is a promising solution to provide comparable performance to DRAM-only memory. To fully exploit the advantages of DRAM and PCM, data placement and migration are key issues. However, current data placement and migration policies, such as RAPP, have unavoidable overheads. How to place and

Table 2. Performance Optimization Techniques of PCM Memory

Technique	Reduce Bits	Reduce Commands	Increase Parallelism	Change Program	WR-RD Impact	Consider Persistence	Memory Type	Implement
DCW[31]	Y			Y			PCM	Chips
Flip-N-Write[32]	Y			Y			PCM	Chips
DWC[36]		Y					PCM	MC
Two-stage-write[33]			Y	Y			SLC	Chips
D-XOR[34]			Y	Y			SLC	Chips
PreSET[35]				Y			SLC	MC
Power-token[37]			Y				PCM	MC
FPB[22]			Y				MLC	MC, DIMM
OptiPCM[39]			Y				PCM	Memory bus
Memory buffer[30]			Y				PCM	DIMM
Write cancellation[21]					Y		PCM	MC, chips
Write pausing[21]					Y		MLC	MC, chips
Write truncation[43]				Y			MLC	MC, chips
Mercurcy[44]				Y			MLC	Chips
Morphable MS[45]				Y			MLC	OS, chips
NVM Duet[51]						Y	PCM	MC

migrate data between DRAM and PCM with low overhead remains a problem.

4 Lifetime Improving Techniques

One of the major weaknesses of PCM is its limited write endurance. This is because converting the state of phase change material repeatedly would result in the permanent stuck-at fault of PCM cell. SLC PCM can endure about 10^9 writes[11], and MLC PCM can only endure 10^7 writes. Fault blocks due to too many writes not only result in data losing, but also reduce the available capacity and lifetime of memory system.

In this section, we survey techniques to improve the lifetime of PCM memory, which can be classified into three categories. First, improving the endurance of PCM itself can efficiently improve its lifetime. Second, delaying the worn out of PCM cells can also improve PCM's lifetime, including reducing writes to PCM and wear leveling. At last, fault block reusing that continuously uses partly faulty blocks rather than discards them is also effective to improve the lifetime of PCM memory.

Subsection 4.1 surveys techniques to improve the endurance of PCM. Subsection 4.2 and subsection 4.3 survey techniques of reducing writes and wear leveling, respectively. Subsection 4.4 presents fault block reusing techniques.

4.1 Improving Endurance

The endurance of PCM is affected by many factors, such as programming current and resistance drift. In this subsection, we survey architectural techniques to improve PCM's endurance, and thus improve its lifetime.

MLC PCM requires large resistance range to store multiple states per cell. In order to reduce readout error rate due to resistance drift, conversional MLC design increases the resistance margin between two adjacent states, which further increases the maximum resistance requirement. As a result, large RESET current is needed to initialize its maximum resistance. However, the endurance of MLC decreases as the RESET current increases[53].

Jiang *et al.*[53] proposed elastic RESET (ER) to reduce the RESET current and thus improve the endurance of MLC. ER first uses frequent pattern compression (FPC)[54] to store data in PCM line. Thus, the memory capacity requirements are varied from the data to write. ER adaptively initializes a PCM line according to the memory capacity requirement at runtime. If the compressed data is less than or equal to 50% of the original size, then ER programs each cell as 2-state SLC. If the compressed data is larger than 50% but less than or equal to 75%, then ER programs each cell as 3-state MLC. If the compressed data is larger than 75%, then ER programs each cell as 4-state MLC.

[11] International technology roadmap for semiconductors (ITRS 2013). http://www.itrs.net/Links/2013ITRS/Home2013.htm, Nov. 2014.

The readout error rate of MLC is related to the data pattern. The readout error rates of "10" and "01" (drift-sensitive) are higher than those of "11" and "00" (drift-insensitive). This is because "10" and "01" are in the intermediate resistance state. Zhang and Li[55] proposed bit-inversion and rotation operations to reduce the readout error rate of MLC. The key idea is to store more data in drift-insensitive state rather than drift-sensitive state by applying bit-inversion, rotation, or a combination of both, as shown in Fig.10. Fig.10(a) shows bit-inversion that inverts data from highly drift-sensitive state "01" to less drift-sensitive state "10". Fig.10(b) shows data rotation that changes the data from drift-sensitive states "0110" to drift-insensitive states "0011". Fig.10(c) shows an example of combining bit-inversion and rotation. Bit-inversion and rotation can guarantee low error rate but narrow the resistance margin. As a result, the initial RESET current is reduced, which improves the endurance of MLC. Bit-inversion and rotation techniques were proposed to tolerate resistance drift, and ER is a general technique. They can be combined to further improve MLC PCM's endurance.

4.2 Reducing Writes

Hybrid DRAM/PCM memory described in Subsection 3.3 can reduce writes to PCM by placing frequently accessed data in DRAM. This subsection surveys other research work that aims to reduce writes to PCM memory, including reducing redundant writes, data compression, cache replacement and partition policy.

4.2.1 Reducing Redundant Writes

There are two situations that memory writes are redundant. First, the data to write is unmodified. Second, the data to write is useless, if 1) it will not be used in future, or 2) it is similar to the old data, which does not affect the applications. In this subsection, we survey techniques to reduce redundant writes to PCM memory.

• *Reducing Unmodified Writes.* Unmodified data does not need to be written back to PCM memory. DCW[31] and reducing-bit-writes[6] techniques only write modified data bits to PCM (their details have been presented in Subsection 3.1.1). These techniques identify unmodified data in the memory chip. Actually, we can also identify whether the data is modified or not in the cache hierarchy. Qureshi *et al.*[5] proposed Line Level WriteBack (LLWB) to write the dirty lines of a page when it is evicted from DRAM cache. Lee *et al.*[4] proposed partial writes to reduce writes to PCM memory. Instead of writing the whole cache line, partial writes track the modified data of cache line and only write the modified data to PCM.

Aforementioned techniques[4-6,31] identify unmodified data by comparing the new data with the old data. The number of unmodified data can be increased by encoding the new data or choosing the old data. As a result, writes to PCM memory can be further reduced. Flip-N-Write[32] described in Subsection 3.1.1 is a kind of flipping manipulation, which flips the new data or not. It guarantees to write half data at most by writing the flipped or not-flipped data. Zhao and Zhu[56] further proposed three data manipulations, which are flipping, XOR, and OR. The manipulation that needs to write least bits compared to the old data is chosen. Jacobvitz *et al.*[57] proposed to choose old data that has the most unmodified data compared to the new data. Coset coding performs a one-to-many mapping from each dataword to a coset of vectors. FlipMin[57] was

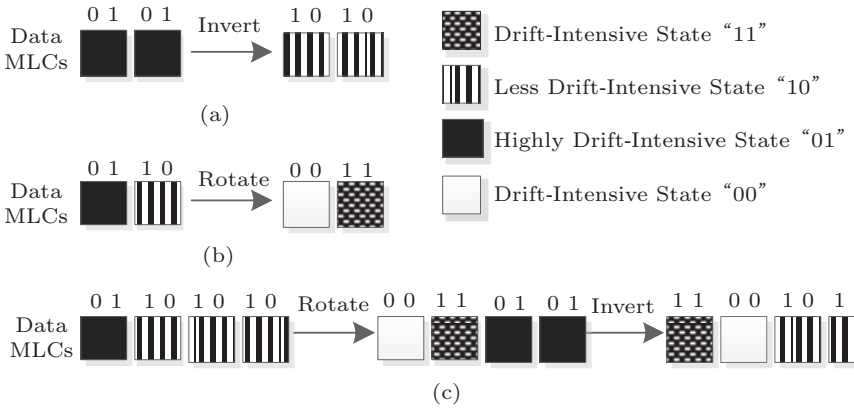


Fig.10. Using (a) bit-inversion, (b) rotation, or (c) the combination of both to convert the original data to drift-tolerant one[55].

proposed to reduce writes by exploiting coset coding to select a vector that minimizes the number of bits write.

• *Reducing Useless Writes.* In case of writing data that will not be used again, these writes to PCM are also unnecessary. Based on this observation, Bock *et al.*[58] proposed to reduce useless write-backs of cache line data. If the corresponding memory region is useless for program, such as heap and stack space, then the cache line data are not written back to PCM. By exploiting the error tolerance characteristic of video applications, SoftPCM[59] relaxes the accuracy of write operations to reduce writes to PCM. SoftPCM cancels write operation if the new data to write are similar to the original data in PCM.

4.2.2 Data Compression

Compressing data can reduce the data volume. Thus, writing the compressed data rather than the original data can reduce writes to PCM memory.

There are frequent values that are accessed frequently in applications. For some applications, mere ten values occupy 50% of total memory accesses[60]. Therefore, reducing writes of frequent values can effectively reduce memory writes. Sun *et al.*[61] proposed frequent value compression (FVC) that compresses the frequent values and writes the compressed data to PCM memory. FVC uses static profiling and dynamic profiling techniques to find the frequent values of applications, considering the implementation overheads.

The FVC technique only compresses memory data when it is written to PCM. For hybrid DRAM/PCM memory, compressing data in DRAM cache can reduce cache misses, and thus further reduce writes to PCM memory. Baek *et al.*[62] proposed Dual-Phase Compression (DPC) to reduce writes in hybrid DRAM/PCM memory. Fig.11 shows an overview of the DPC mechanism. In the first phase, DPC uses a successive matching algorithm to compress data and stores the compressed data in DRAM cache. In the second phase, DPC uses bit-based frequent pattern compression to further compress the data of DRAM cache to reduce writes to PCM memory. Du *et al.*[63] proposed delta compression that only compresses the modified data of DRAM cache and writes them back to PCM memory.

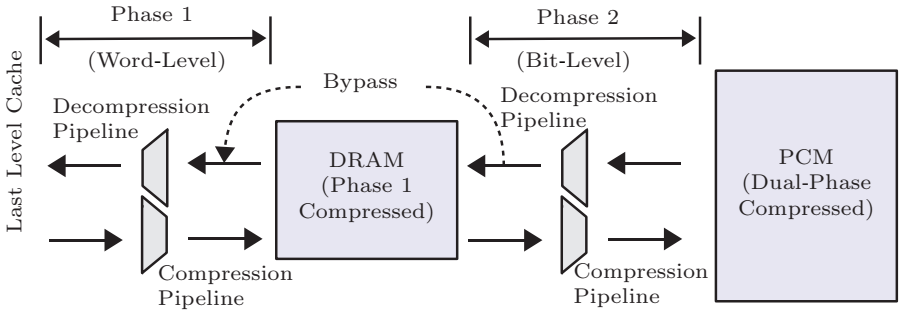


Fig. 11. Overview of Dual-Phase Compression mechanism[62].

4.2.3 Cache Replacement and Partition Policy

Conventional cache replacement and partition policies aim to improve the cache hit rate, disregarding reads or writes. However, writing data from cache to PCM memory increases energy consumption and affects the lifetime of PCM. Moreover, the highest cache hit rate does not mean the best performance definitely for PCM memory. This is because the write latency of PCM is longer than its read latency. Thus, cache write-back may degrade system performance more significantly than cache miss.

Replacing a dirty cache block to memory needs to write memory, whereas replacing a clean cache block does not induce actual write to memory. Several cache replacement policies were proposed to reduce writes to PCM memory. Their key idea is replacing clean cache blocks with high priority based on existing cache replacement policies. Ferreia *et al.*[64] proposed a new clean-preferred victim selection policy with N chances (CLP-N). CLP-N selects the oldest clean cache block among the N least recently used cache blocks as a vic-

⑫ Re-reference interval prediction (RRIP) is a cache replacement policy that selects the eviction block by predicting the re-reference interval of a cache block.

tim. If such a clean cache block does not exist, the LRU cache block is evicted. Based on RRIP⑫, Rodriguez *et al.*[65] proposed to evict the clean block among blocks that are predicted to be re-referenced at long time later.

Both [64] and [65] do not quantize the costs of read and write, while the following cache replacement policies do. Zhang *et al.*[66] proposed a read-write aware cache replacement policy (RWA) by adding a counter per cache block. RWA assigns a large value to the corresponding counter when a cache block is written and assigns a small value if a cache block is read. The counters of other cache blocks decrease by one when a request arrives at one block. RWA selects to evict the cache block with the minimum counter value. As such, RWA reduces writes of dirty cache line data to PCM. Barcelo *et al.*[67] proposed variable aging (VA) cache replacement policy that considers the cost of PCM read and write. Instead of using reference times as the age of a block, VA uses the rate of reference times to reference cost as the age. As such, clean blocks age at a rate of 1, and dirty blocks age at a rate of $1/c$, where c is the average cost of writing a block to PCM. VA replaces the oldest block as LRU.

To reduce writes to PCM, Zhou *et al.*[68] proposed write-back-aware cache partition (WCP) policy that considers the trade-off of cache writing back and cache miss. WCP is a run-time mechanism that partitions a shared LLC among multiple applications. WCP considers the reduction in cache misses, as well as the cost of write-back to PCM.

4.2.4 Summary

Table 3 summarizes the techniques to improve the lifetime of PCM memory by reducing writes to PCM memory. The *implement* column shows that writes to PCM memory can be reduced from the software, cache to memory controller, and memory chip perspectives. These techniques can be classified into five categories, including reducing unmodified writes, reducing useless writes, data compression, write-aware cache replacement, and cache partition.

Among these techniques, SoftPCM is only applicable to video applications, while the other techniques are effective for more applications. Reducing useless write-backs needs the support of OS to identify useless write-backs, which is more complicated than the other hardware-only techniques. For data compression techniques, dual-phase compression compresses all data. The other two techniques selectively compress data that improve the efficiency of compression.

We can see that the aforementioned architectural techniques to reduce writes to PCM memory are relatively comprehensive. However, these architectural optimizations reduce the volume of writes by only a factor about 3[69]. Some researchers aimed to reduce writes at software level, such as compiler[70], and data structure[69]⑬. They are important complements to the architectural optimizations. These software techniques are only preliminary explorations. There is still large space to further reduce writes to PCM memory. For instance, database systems have more information about how memories are used than hardware, which provides opportunity to reduce writes to PCM memory. We expect more research work from these aspects.

4.3 Wear Leveling

Reducing writes to PCM can improve its lifetime. However, it is not enough to overcome the limited write endurance of PCM. Because the locality of memory accessing results in unbalanced writes that some cells are written more than others. By address remapping, wear leveling (WL) can distribute writes to the whole PCM address space, which can further improve the lifetime of PCM memory. Based on whether to record the write times of data blocks, the wear leveling techniques can be divided into two categories: write times based wear leveling and random wear leveling. The following two subsections present these two kinds of wear leveling techniques respectively.

4.3.1 Write Times Based Wear Leveling

Write times based wear leveling records the write times of every data block, and swaps "hot" blocks (more writes) with "cold" blocks (less writes) to achieve balanced writing.

Segment swapping[6] records the write times of every memory segment in the memory controller. It swaps segments of large writes with segments of small writes after several writes. Segment swapping assumes that different PCM cells have the same write endurance. However, the actual write endurance is varied among different cells due to process variation[71]. Based on this insight, Dong *et al.*[72] proposed wear rate leveling (WRL) considering the distinct write endurance. WRL uses the ratio of write times to write endurance as the metric and swaps the data blocks of high ratio with

⑬ The details of software techniques are out of the scope of this paper.

Table 3. Reducing Writes to PCM Memory

Technique	Reduce Unmodified Writes	Reduce Useless Writes	Data Compression	Cache Replace	Cache Partition	Implement
DCW[31]	Y					Chips
Reducing-bit-writes[6]	Y					Chips
LLWB[5]	Y					Cache
Partial writes[4]	Y					Cache
Flip-N-Write[32]	Y					Chips
FlipMin[57]	Y					MC
Reducing useless write-backs[58]		Y				OS, MC
SoftPCM[59]		Y				MC
Dual-phase compression[62]			Y			Cache, MC
Frequent values compression[61]			Y			MC
Delta compression[63]			Y			Cache
CLP-N[64]				Y		Cache
RRIP-variation[65]				Y		Cache
RWA[66]				Y		Cache
VA[67]				Y		Cache
WCP[68]					Y	Cache

blocks of low ratio. To reduce the overhead of recording every data block, Yun *et al.*[73] proposed bloom filter based wear leveling technique. Bloom filters are used to identify hot and cold data blocks, which reduces the number of write counters.

Dhiman *et al.*[26] proposed PDRAM that considers wear leveling in the memory management of operating system. Fig.12 shows the PDRAM architecture. The memory controller uses an access map to count the write times of every physical page. The access map is stored in PRAM, and the memory controller keeps a cache of the access map. When a page has been written several times, PDRAM writes the data of the page to a new allocated page. PDRAM preferentially allocates a free page of less write times. Age-based page allocation[74] exploits a similar idea with PDRAM. To reduce the overhead of finding free page with low writes, research work of [74] explores the organization

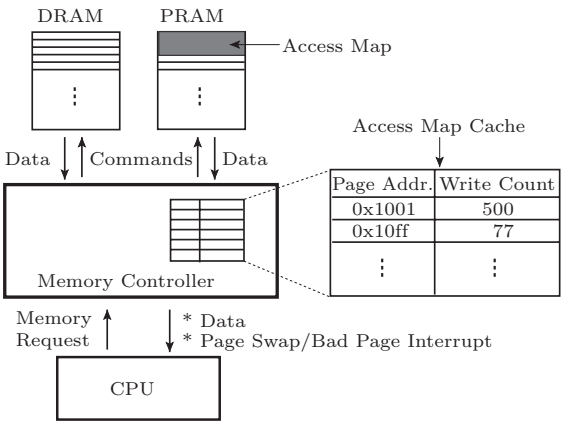


Fig.12. Overview of PDRAM architecture[26].

of physical pages considering write times. PDRAM and age-based page allocation can be the complements of architecture-only wear leveling techniques, and thus reduce data migrations and their costs.

4.3.2 Random Wear Leveling

Write times based wear leveling introduces extra storage overhead to record write times of every data block. Instead of recording write times, random wear leveling randomly swaps the data of different positions after a number of writes.

Row shifting[6] randomly shifts the data of a PCM row at the word granularity periodically. Fine-grained wear leveling (FGWL)[5] makes the writes uniform at the line granularity. FGWL uses a pseudo random number generator to get a random value when the OS allocates a physical page. FGWL stores the lines of each page in a rotated manner according to the random value until the page is reclaimed. Start-gap[75] utilizes an extra line (called GapLine) to simplify data swap between different lines. Start-gap moves one line that neighbors the GapLine to the GapLine after several writes. The neighboring line becomes the new GapLine which is used to move data next time. Only a start register and a gap register are needed to remap address in start-gap. Fig.13 shows an example of start-gap wear leveling.

PCM design has to consider not only the durability under normal application behavior, but also security issues due to malicious attack. Security refresh[76] avoids information leak by migrating physical locations inside the PCM. Security refresh controller (SRC), which is embedded in PCM bank, dynamically remaps

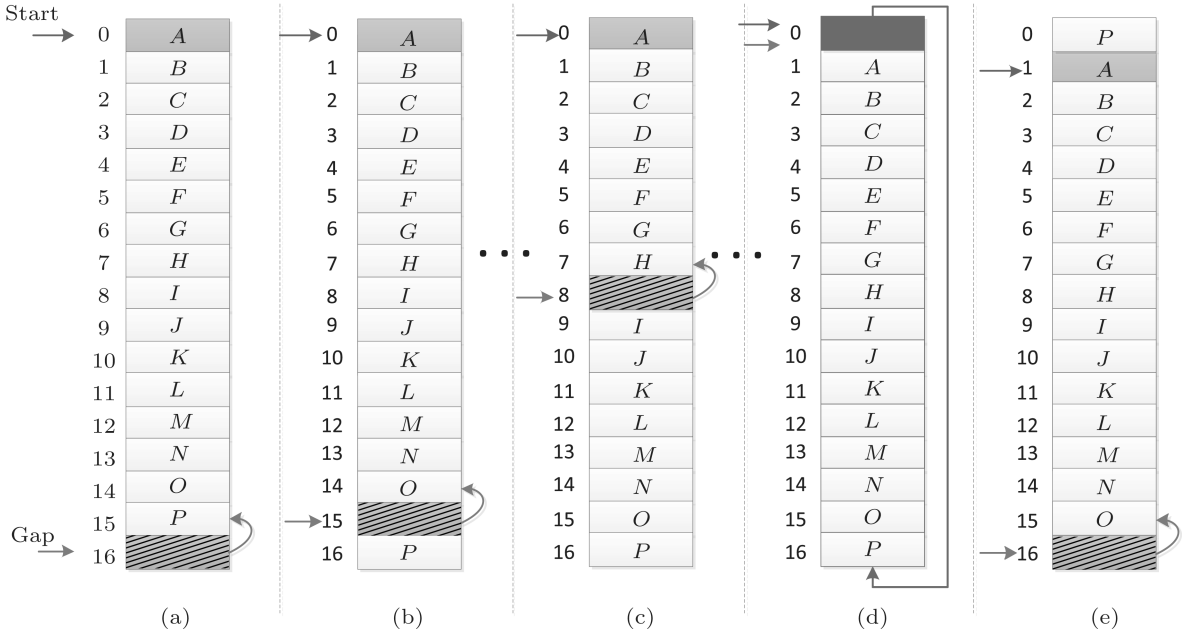


Fig.13. Start-gap wear leveling on a memory containing 16 lines[75].

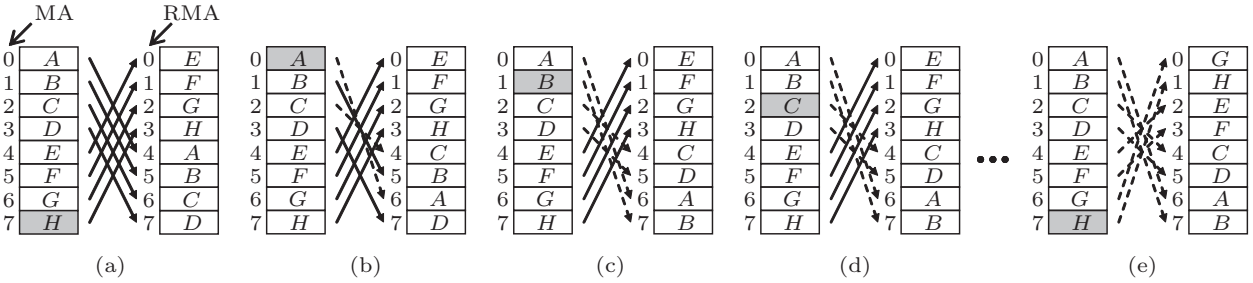


Fig.14. Example of one complete security refresh round[76]. (a) Initial state. (b) 1st refresh. (c) 2nd refresh. (d) 3rd refresh. (e) Final state.

data block memory address (MA) to refreshed memory address (RMA) inside a PCM bank using a randomly generated key (called *refresh*). SRC generates a new random key after several writes and refreshes the mapping again. Fig.14 shows an example of one complete *security refresh round*, similar to DRAM's refresh period[76]. At the first refresh, MA 0 is refreshed. Assuming the random generated key is k_1 ($k_1 = 6$), then MA 0 is remapped from RMA 4 to RMA 6 (000 XOR $k_1 = 110$). Since the data (A) of MA 0 is moved to RMA 6, the original data (C) in RMA 6 is remapped to RMA 2 (110 XOR $k_1 = 010$). The next security refresh is similar to the first refresh.

4.3.3 Summary

Table 4 summaries the wear leveling techniques from the perspective of the key idea, granularity, randomness, and implementation hierarchy.

We can see that most techniques implement wear leveling in the memory controller. PDRAM[26] and age-based page allocation[74] consider wear leveling in the OS memory management, which needs the support of memory controller. Random wear leveling does not record the write times of every data block, and thus has lower overhead. However, random wear leveling cannot achieve the best wear leveling because it does not differentiate "hot" and "cold" data blocks. Write times based wear leveling performs better but has higher storage overhead compared with random techniques. Bloom filter-based WL[73] was proposed to reduce the storage overhead of recording write times.

The granularity of wear leveling affects the efficiency and implementation overhead. Coarse-grained wear leveling usually needs low implementation overhead, but it is difficult to provide global wear leveling, which is contrary to fine-grained wear leveling. In the-

Table 4. Wear Leveling Techniques of PCM Memory

Technique	Key Idea	Implement	Randomness	Granularity
Row shifting[6]	Random shift within a line	MC	Y	Word
FGWL[5]	Store the lines of page in a rotated manner	MC	Y	Line
Start-gap[75]	Swap data with neighboring line	MC	Y	Line
Segment swapping[6]	Segments swap based on write times	MC	N	Segment
Security refresh[76]	Remap a block to another one	MC	Y	Block
Wear rate leveling[72]	Swap blocks considering endurance variations	MC	N	Block
Bloom filter-based WL[73]	Use bloom filter to reduce storage overhead	MC	N	Block
PDRAM[26]	Memory management considering writes	MC, OS	N	Page
Age-based page allocation[74]	Memory management considering writes	MC, OS	N	Page

ory, combining fine-grained WL and coarse-grained WL at different levels, such as OS and hardware, can utilize their advantages and avoid their drawbacks.

Wear rate leveling considers the endurance variations of different blocks. However, the difficulty of identifying the endurance variations limits its adoption. Due to the simplicity and efficiency, the start-gap algorithm has been used in actual PCM-based SSD prototypes, such as Onyx[77] and PSS[78].

4.4 Tolerating Wear-Out Faults

Although improving endurance, reducing writes and wear leveling can delay the wear-out of PCM cells, some PCM cells still wear out before other cells unavoidably. This is because the absolute wear leveling of all cells is impossible. This subsection surveys techniques that tolerate wear-out faults to improve the lifetime of PCM memory. One method is error correcting that corrects fault bits using well bits. The other method is fault block reusing that reuses the well bits of the fault block. Subsection 4.4.1 and Subsection 4.4.2 present techniques of error correcting and fault block reusing, respectively.

4.4.1 Error Correcting

Conventional error correction coding cannot correct fault bits because the states of the fault cells cannot be converted any more. Schechter *et al.*[79] proposed error-correcting pointers (ECP) that permanently encode the locations of fault cells into a table and assign well cells to replace them. Fig.15 shows the ECP-5 scheme that corrects up to five failed cells for a 512-bit data block (or line). The high nine bits of the correction entry indicate the location of the failed bits and the low one bit stores the right value of the failed bit.

ECP uniformly allocates six correction entries for each PCM data block (512 bits) that can correct up to 6-bit faults. However, most data blocks are correct or have one bit fault, and only a few data blocks have more bits faults. As a result, ECP induces high storage overhead. Based on this observation, Qureshi[80] proposed pay-as-you-go (PAYG) to improve the lifetime of PCM while reducing storage overheads. PAYG only allocates one error correction entry per data block to correct 1-bit fault. In order to correct data blocks that have more faults, PAYG uses a global correction entries pool to provide extra entries for these data blocks.

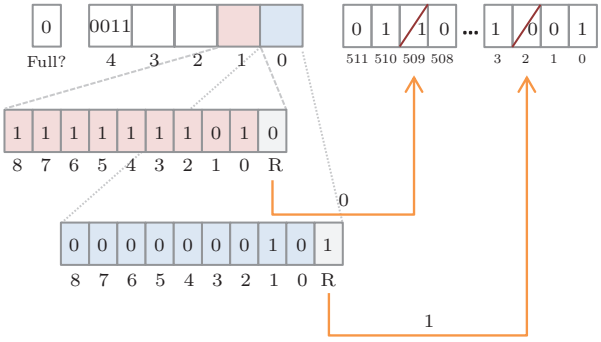


Fig.15. ECP5 scheme that corrects up to 5 failed cells. There are 5 correction entries in each data block[79].

ECP and PAYG need extra space to store error correction entries, which suffer from large storage overhead. To reduce the overhead, Yoon *et al.*[81] proposed fine-grained remapping with ECC and embedded pointer (FREE-p) that remaps worn-out PCM block to well block using a pointer. The granularity of block is 64B. Fig.16 shows an example of FREE-p. The well cells of a partly faulty block are used to store the pointer that points to the well data block. The 1-bit D/P (data/pointer) flag indicates whether the block has been remapped. FREE-p does not need dedicated space to store the pointer.

ECP discards the data line when the number of fault bits is out of the scope that it can correct, which

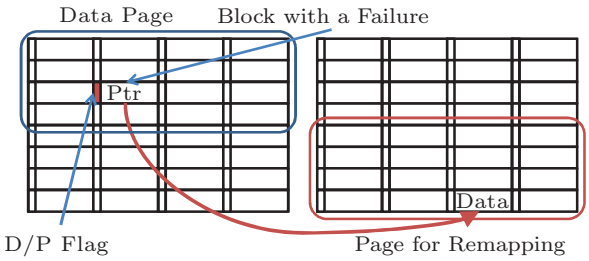


Fig.16. Example of FREE-p[81].

results in non-contiguous memory space. Jiang *et al.*[82] proposed Line-Leveling Mapping and Salvaging (LLS) to achieve a longer PCM lifetime. By allocating a dynamic portion of total space in a PCM device as backup space, LLS remaps fault lines to backup space. As a result, LLS constructs a contiguous PCM space and masks lower-level failure from the OS and applications. Fig.17 shows the memory splitting of LLS. The percentage of backup space increases as fault lines increase, and at most half of the total memory space is allocated as backup space. LLS uses ECP to correct data line with few fault bits.

Although a failed cell with a stuck-at value cannot be written, it is still readable. SAFER[83] exploits this key attribute and uses failed cells to store data, thereby reducing the overhead of error recovery. SAFER partitions a data block dynamically while ensuring that there is at most one failed bit per partition. SAFER unceasingly uses the failed bit by utilizing bit inversion technique. If a new failure occurs in a partition that already has a fail bit, then SAFER needs to re-partition the data block. To improve the efficiency of block partition, Fan *et al.*[84] proposed Aegis, a systematical partition scheme that uses fewer groups to accommodate more faults. Ageis utilizes the property of Cartesian plane that any two different points on a line uniquely determine the slope of the line. If the slope is changed, then at most one point on the original line will stay on a new line. Therefore, Ageis guarantees that any two bits in the same group will not be in the same group after a repartition, which reduces the average overhead of tolerating one-bit fault. Fig.18 shows how Aegis partitions a data block. Bits that have the same identification consist of a group.

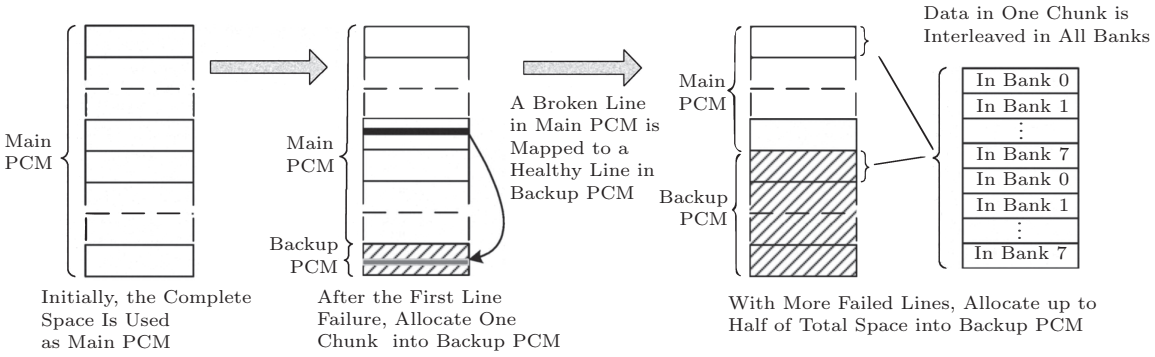


Fig.17. Splitting the PCM memory to the main and backup space (to achieve graceful degradation, each chunk contains data from all banks)[82].

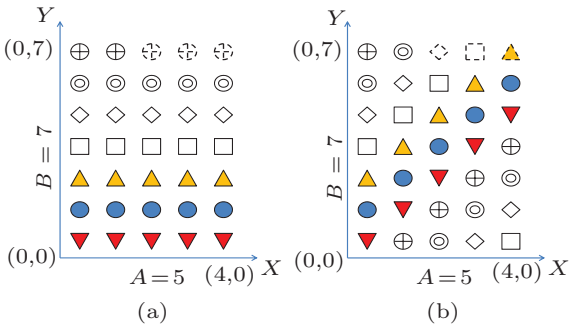


Fig.18. Illustrating how bits in a 32-bit data block are partitioned into 7 groups, each of 5 bits. (a) The partition method uses slope $k = 0$. (b) A different method uses slope $k = 1$. In total, there are 7 partition methods of the 5×7 rectangle[84].

4.4.2 *Fault Block Reusing*

Discarding the data block with partly worn-out cells not only wastes memory capacity, but also reduces the lifetime of PCM memory. This subsection surveys fault block reusing techniques that reuse the well bits of a fault block.

Ipek *et al.*[85] proposed dynamically replicated memory (DRM) that reuses memory pages that contain hard faults. DRM dynamically forms pairs of complementary pages that act as a single page. Fig.19 shows an example of compatible pages and incompatible pages. If there is no same fault bit between two pages, then the

two pages are compatible. Otherwise, the two pages are incompatible. Since a page has many bits, the probability that two pages have the same fault bits is low. Therefore, the probability of finding two compatible pages is high. DRM uses two fault pages as a single page, which results in low memory space utilization. Chen *et al.*[86] proposed re-cycling PRAM (RePRAM) to improve the utilization of PCM memory. RePRAM leverages a group of fault pages that are compatible and uses a well page to store the fault bits of these pages in the group. As the number of faults per page increases to a threshold, RePRAM uses DRM to tolerate faults.

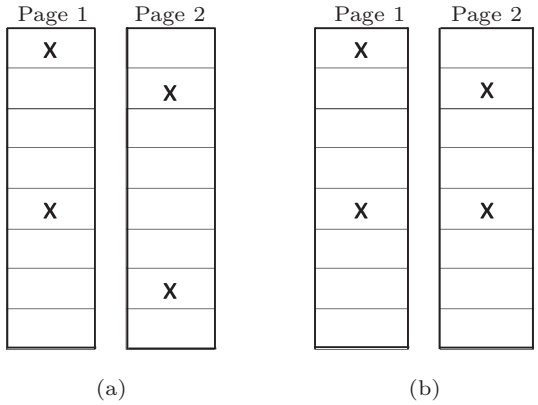


Fig.19. Example of (a) compatible and (b) incompatible pages[85].

Both DRM and RePRAM reuse fault block at the page granularity, which results in low memory space utilization. Azevedo *et al.*[87] proposed Zombie to extend the lifetime of pages by reusing well blocks in discarded pages. Zombie uses a block, or even a subblock, sourced from disabled pages (spare block) to pair with a block in software-visible pages (primary block). Zombie enables dynamically increasing spare subblock size as the faults in primary block increase to improve the lifetime of primary block. Two new error correction mechanisms, ZombieMLC and ZombieXOR, were proposed in [87]. ZombieMLC is designed specifically for MLC to tolerate both stuck-at faults and drift. ZombieXOR is suit for SLC and only tolerates stuck-at faults.

4.4.3 Summary

Table 5 summaries the wear-out fault tolerance techniques.

The *key idea* column shows how fault tolerance is achieved: by replacing partly faulty block with well block (replacing); by using compatible page or block to pair well memory region (pairing); or by partitioning partly faulty blocks with bit inversion per partition (partition+inversion). There are two granularities to tolerate the wear-out faults: block (or called line) granularity and page granularity. The typical size of one block is 512 bits. Page granularity techniques have low storage overhead but result in low utilization of PCM memory capacity. Therefore, the latest research efforts usually use block granularity. All techniques are implemented in hardware. FREE-p, DRM, and RePRAM techniques need the support of OS. The *reusing* column refers to whether a technique reuses the partly faulty page or block.

These error correction techniques can correct variable faults with different storage overheads. To compare these techniques, the *error correction overhead* column shows the storage overhead to tolerate 6-bit faults for a 512-bit block. Since LLS uses ECP to tolerate faults when the number of fault bits is less than 6, its error correction overhead is the same as that of ECP. We can see that the overhead of PAYG and Aegis is lower than that of the other techniques.

Wear-out fault tolerance is essential for real memory system, especially for PCM memory that has limited write endurance. Although there are many studies that aim to improve the lifetime of PCM by tolerating wear-out faults, they have not been used in real systems or PCM chips. Further exploring wear-out fault

Table 5. Tolerating Wear-Out Faults of PCM Memory

Technique	Key Idea	Granularity	Implement	Reusing	Error Correction Overhead
ECP[79]	Replacing	Block	Hardware	No	61 bits/block for ECP-6
PAYG[80]	Replacing	Block	Hardware	No	19.5 bits/block
FREE-p[81]	Replacing+pairing	Block	OS, hardware	No	64 bits/block
LLS[82]	Replacing	Block	Hardware	No	61 bits/block
SAFER[83]	Partition+inversion	Block	PCM chips	No	55 bits/block
Ageis[84]	Partition+inversion	Block	PCM chips	No	27 bits/block
DRM[85]	Pairing	Page	OS, hardware	Yes	–
RePRAM[86]	Pairing	Page	OS, hardware	Yes	–
Zombie[87]	Pairing	Block	Hardware	Yes	–

tolerance techniques with higher utilization of memory capacity and lower overhead is still an open issue.

5 Energy Saving Techniques

Energy consumption has become one of the bottlenecks of memory systems[88-90]. PCM is a non-volatile memory without refresh operation, and thus has lower static energy than DRAM. However, the write energy of PCM is largely higher than that of DRAM. The high write energy limits the write parallelism of PCM which results in poor write performance. On the other hand, high write energy also introduces logical error, uncompleted state conversion, and read error[9]. Therefore, reducing energy consumption is critical to construct energy-efficient and reliable PCM memory system. We survey energy saving techniques of PCM memory in this section.

5.1 Reducing Writes

Reducing writes to PCM summarized in Subsection 4.2 can not only improve the lifetime of PCM, but also save energy. MLC PCM consumes more energy than SLC PCM. The latency optimization techniques of MLC PCM summarized in Subsection 3.5 can also reduce energy consumption, because these techniques actually reduce writes to MLC PCM. We do not repeat the details in this subsection.

5.2 Exploiting Energy Asymmetry

The write energy of PCM is asymmetric, which depends on the writing values. For example, writing bit "0" consumes more energy than writing bit "1". Writing "01" and "10" consumes more energy than writing "00" and "11" for MLC PCM cells. This subsection describes techniques to save energy by exploiting the energy asymmetry.

Various research studies exploit the energy asymmetry of writing bit "1" and writing bit "0" to reduce the energy consumption of PCM memory[91-94]. Xu *et al.*[91] proposed XOR-masked write to reduce write energy. For the data to write and the original data, their technique finds an optimal bit-pattern such that writing an XOR-masked value leads to minimum write energy. Fig.20 shows the data flow diagram of the XOR-masked write strategy. Mirhoseini *et al.*[92] also proposed a coding-based technique to reduce energy consumption. Integer linear programming and dynamic programming approaches are used to find the optimal codes. Chen *et al.*[93] proposed out-of-position write that selectively writes the free page with the lowest energy consumption by comparing writing energies of different pages. Yue and Zhu[94] proposed to invert a cache line when there are more "0" bits than "1" bits in the cache line. Since writing "00" and "11" consumes less energy than writing "01" and "10", Wang *et al.*[95] proposed to increase writes of "00" and "11" and reduce writes of "01" and "10" using data encoding technique.

5.3 Summary

Writes to PCM not only result in the worn-out PCM cells, but also increase the energy consumption of the memory system. Thus, various studies aim to save the energy of PCM memory systems by reducing writes to PCM. On the other hand, the asymmetric write energy of PCM provides opportunities to reduce the energy consumption of PCM memory. Actually, the energy saving by utilizing energy asymmetry is limited compared with reducing writes. The high write energy of PCM not only increases system running cost, but also limits its performance. Therefore, how to reduce writes to PCM as many as possible remains a critical issue of PCM energy saving research.

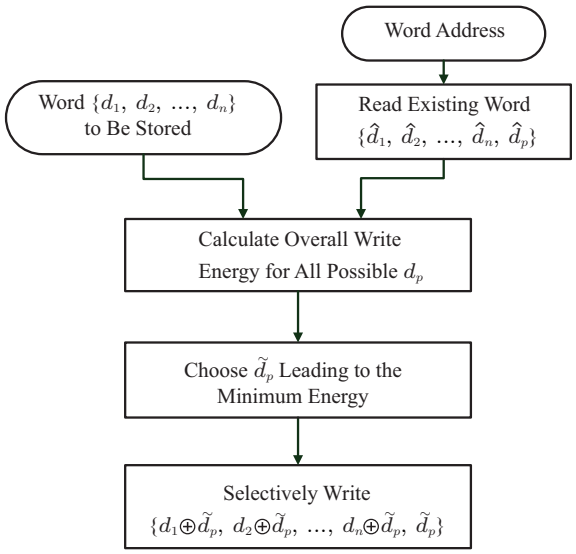


Fig.20. Data flow diagram of the XOR-masked write strategy[91].

6 Conclusions

The long write latency, limited write endurance, and high write energy of PCM challenge its adoption as main memory. In this paper, we extensively surveyed

the key techniques of PCM memory systems, including performance optimization, lifetime improving, and energy saving. Since the issues of PCM are caused by writes, reducing writes is a fundamental technique to optimize PCM memory systems.

Actually, the chip-level write latency of PCM makes PCM-only memory difficult to satisfy the latency requirement of real memory systems. Although most techniques do not take the fact into consideration, these architectural research efforts are still valuable. First, the hybrid DRAM/PCM memory system is promising to provide comparable performance with DRAM memory. Thus, addressing the issues of PCM is still necessary. Second, many techniques, such as wear-out fault tolerance techniques, are also applicable to other NVMs that have endurance issue. At last, a few techniques, such as start-gap, are applicable to PCM-based storage devices. Adopting PCM as storage is a practical choice as its latency is shorter than that of NAND flash.

Compared to DRAM, a key advantage of PCM is its non-volatility. Just using PCM to replace DRAM or construct hybrid DRAM/PCM memory system cannot utilize this feature. How to exploit the non-volatility of PCM to build persistent and high-performance memory system is a valuable research direction. Recently, Kiln[96] adopts a non-volatile cache and a nonvolatile main memory to support persistence from an architecture perspective, which is a good exploration. We expect more efforts in this aspect.

References

[1] Lefurgy C, Rajamani K, Rawson F, Felter W, Kistler M, Keller T W. Energy management for commercial servers. *Computer*, 2003, 36(12): 39–48.

[2] Lim K, Ranganathan P, Chang J, Patel C, Mudge T, Reinhardt S. Understanding and designing new server architectures for emerging warehouse-computing environments. In *Proc. the 35th Annual Int. Symp. Computer Architecture*, Jun. 2008, pp.315-326.

[3] Udipi A N, Muralimanohar N, Chatterjee N, Balasubramonian R, Davis A, Jouppi N P. Rethinking DRAM design and organization for energy-constrained multi-cores. *ACM SIGARCH Comput. Archit. News*, 2010, 38(3): 175–186.

[4] Lee B C, Ipek E, Mutlu O, Burger D. Architecting phase change memory as a scalable dram alternative. In *Proc. the 36th Annual Int. Symp. Computer Architecture*, Jun. 2009, pp.2–13.

[5] Qureshi M K, Srinivasan V, Rivers J A. Scalable high performance main memory system using phase-change memory technology. In *Proc. the 36th Annual Int. Symp. Computer Architecture*, Jun. 2009, pp.24–33.

[6] Zhou P, Zhao B, Yang J, Zhang Y. A durable and energy efficient main memory using phase change memory technology. In *Proc. the 36th Int. Symp. Computer Architecture*, Jun. 2009, pp.14–23.

[7] Lee B C, Zhou P, Yang J, Zhang Y, Zhao B, Ipek E, Mutlu O, Burger D. Phase-change technology and the future of main memory. *IEEE Micro*, 2010, 30(1): 131–141.

[8] Raoux S, Burr G W, Breitwisch M J *et al.* Phase-change random access memory: A scalable technology. *IBM Journal of Research and Development*, 2008, 52(4.5): 465–479.

[9] Mittal S. Energy saving techniques for phase change memory (PCM). arXiv:1309.3785, 2013. http://arxiv.org/abs/1309.3785, Nov. 2014.

[10] Qureshi M K, Gurumurthi S, Rajendran B. Phase Change Memory: From Devices to Systems (1st edition). Morgan & Claypool Publishers, 2011.

[11] Zilberberg O, Weiss S, Toledo S. Phase-change memory: An architectural perspective. *ACM Computing Survey*, 2013, 45(3): 1–33.

[12] Burr G W, Breitwisch M J, Franceschini M *et al.* Phase change memory technology. *Journal of Vacuum Science and Technology B*, 2010, 28(2): 223–262.

[13] Li H, Chen Y. Nonvolatile Memory Design: Magnetic, Resistive, and Phase Change. CRC Press, 2011.

[14] Yamada N, Ohno E, Nishiuchi K *et al.* Rapid-phase transitions of GeTe-Sb2Te3 pseudobinary amorphous thin films for an optical disk memory. *Journal of Applied Physics*, 1991, 69(5): 2849–2856.

[15] Tominaga J, Kikukawa T, Takahashi M, Phillips R T. Structure of the optical phase change memory alloy, Ag-V-In-Sb-Te, determined by optical spectroscopy and electron diffraction. *Journal of Applied Physics*, 1997, 82(7): 3214-3218.

[16] Ovshinsky S R. Reversible electrical switching phenomena in disordered structures. *Physical Review Letters*, 1968, 21: 1450–1453.

[17] Choi Y, Song I, Park M H *et al.* A 20nm 1.8V 8Gb PRAM with 40MB/s program bandwidth. In *Digest of Technical Papers of IEEE Int. Solid-State Circuits Conf.*, Feb. 2012, pp.46–48.

[18] Wong H, Raoux S, Kim S B, Liang J, Reifenberg J P, Rajendran B, Asheghi M, Goodson K E. Phase change memory. *Proceedings of the IEEE*, 2010, 98(12): 2201–2227.

[19] Nirschl T, Phipp J B, Happ T D *et al.* Write strategies for 2 and 4-bit multi-level phase-change memory. In *Proc. IEEE Int. Electron Devices Meeting*, Dec. 2007, pp.461–464.

[20] Pozidis H, Papandreou N, Sebastian A *et al.* Enabling technologies for multilevel phase-change memory. In *Proc. European Phase Change & Ovonics Symposium*, Sept. 2011. http://www.epcos.org/library/papers/pdf_2011/Oral-Papers/S7-03.pdf, Dec. 2014.

[21] Qureshi M K, Franceschini M M, Lastras-Montano L A. Improving read performance of phase change memories via write cancellation and write pausing. In *Proc. the 16th IEEE Int. Symp. High Performance Computer Architecture*, Jan. 2010.

[22] Jiang L, Zhang Y, Childers B R, Yang J. FPB: Fine-grained power budgeting to improve write throughput of multi-level cell phase change memory. In *Proc. the 45th Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2012, pp.1–12.

[23] Kang S, Cho W Y, Cho B H *et al.* A 0.1-μm 1.8-V 256-Mb phase-change random access memory (PRAM) with 66-MHz synchronous burst-read operation. *IEEE Journal of Solid-State Circuits*, 2007, 42(1): 210-218.

[24] Hanzawa S, Kitai N, Osada K *et al.* A 512kB embedded phase change memory with 416kB/s write throughput at 100μA cell write current. In *Digest of Technical Papers of IEEE Int. Solid-State Circuits Conf.*, Feb. 2007, pp.474–616.

[25] Kwon S, Kim D, Kim Y, Yoo S, Lee S. A case study on the application of real phase-change RAM to main memory subsystem. In *Proc. Conf. Design, Automation and Test in Europe*, Mar. 2012, pp.264–267.

[26] Dhiman G, Ayoub R, Rosing T. PDRAM: A hybrid PRAM and DRAM main memory system. In *Proc. the 46th ACM/IEEE Design Automation Conf.*, Jul. 2009, pp.664–669.

[27] Ramos L E, Gorbatov E, Bianchini R. Page placement in hybrid memory systems. In *Proc. the 25th Int. Conf. Supercomputing*, May 31–June 4, 2011, pp.85–95.

[28] Yoon H B, Meza J, Ausavarungnirun R, Harding R A, Mutlu O. Row buffer locality aware caching policies for hybrid memories. In *Proc. the 30th IEEE Int. Conf. Computer Design*, Sept.30–Oct.3, 2012, pp.337-344.

[29] Lee H G, Baek S, Nicopoulos C, Kim J. An energy- and performance-aware DRAM cache architecture for hybrid DRAM/PCM main memory systems. In *Proc. the 29th IEEE Int. Conf. Computer Design*, Oct. 2011, pp.381–387.

[30] Ham T J, Chelepalli B K, Xue N, Lee B C. Disintegrated control for energy-efficient and heterogeneous memory systems. In *Proc. the 19th Int. Symp. High Performance Computer Architecture*, Feb. 2013, pp.424–435.

[31] Yang B D, Lee J E, Kim J S, Cho J, Lee S Y, Yu B G. A low power phase-change random access memory using a data-comparison write scheme. In *Proc. IEEE Int. Symp. Circuits and Systems*, May 2007, pp.3014–3017.

[32] Cho S, Lee H. Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance. In *Proc. the 42nd Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2009, pp.347–357.

[33] Yue J, Zhu Y. Accelerating write by exploiting PCM asymmetries. In *Proc. the 19th IEEE Int. Symp. High Performance Computer Architecture*, Feb. 2013, pp.282–293.

[34] Du Y, Zhou M, Childers B R, Mossé D, Melhem R. Bit mapping for balanced PCM cell programming. In *Proc. the 40th Annual Int. Symp. Computer Architecture*, Jun. 2013, pp.428–439.

[35] Qureshi M K, Franceschini M M, Jagmohan A, Lastras L A. PreSET: Improving performance of phase change memories by exploiting asymmetry in write times. In *Proc. the 39th Annual Int. Symp. Computer Architecture*, Jun. 2012, pp.380–391.

[36] Xia F, Jiang D, Xiong J, Chen M, Zhang L, Sun N. DWC: Dynamic write consolidation for phase change memory systems. In *Proc. the 28th ACM Int. Conf. Supercomputing*, Jun. 2014, pp.211–220.

[37] Hay A, Strauss K, Sherwood T, Loh G H, Burger D. Preventing PCM banks from seizing too much power. In *Proc. the 44th Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2011, pp.186–195.

[38] Zheng H, Lin J, Zhang Z, Gorbatov E, David H, Zhu Z. Mini-rank: Adaptive DRAM architecture for improving memory power efficiency. In *Proc. the 41st IEEE/ACM Int. Symp. Microarchitecture*, Nov. 2008, pp.210-221.

[39] Li Z, Zhou R, Li T. Exploring high-performance and energy proportional interface for phase change memory systems. In *Proc. the 19th Int. Symp. High Performance Computer Architecture*, Feb. 2013, pp.210–221.

[40] Zhang W, Li T. Exploring phase change memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures. In *Proc. the 18th Int. Conf. Parallel Architectures and Compilation Techniques*, Sept. 2009, pp.101-112.

[41] Lee S, Bahn H, Noh S H. Characterizing memory write references for efficient management of hybrid PCM and DRAM memory. In *Proc. the 19th IEEE Int. Symp. Modeling, Analysis & Simulation of Computer and Telecommunication Systems*, July 2011, pp.168–175.

[42] Meza J, Chang J, Yoon H, Mutlu O, Ranganathan P. Enabling efficient and scalable hybrid memories using fine-granularity DRAM cache management. *IEEE Computer Architecture Letters*, 2012, 11(2): 61–64.

[43] Jiang L, Zhao B, Zhang Y, Yang J, Childers B R. Improving write operations in MLC phase change memory. In *Proc. the 18th Int. Symp. High Performance Computer Architecture*, Feb. 2012.

[44] Joshi M, Zhang W, Li T. Mercury: A fast and energy-efficient multi-level cell based phase change memory system. In *Proc. the 17th Int. Symp. High Performance Computer Architecture*, Feb. 2011, pp.345-356.

[45] Qureshi M K, Franceschini M M, Lastras-Montañ L A, Karidis J P. Morphable memory system: A robust architecture for exploiting multi-level phase change memories. In *Proc. the 37th Annual Int. Symp. Computer Architecture*, Jun. 2010, pp.153–162.

[46] Condit J, Nightingale E B, Frost C, Ipek E, Lee B, Burger D, Coetzee D. Better I/O through byte-addressable, persistent memory. In *Proc. the 22nd ACM SIGOPS Symp. Operating Systems Principles*, Oct. 2009, pp.133–146.

[47] Wu X, Reddy A L N. SCMFS: A file system for storage class memory. In *Proc. Int. Conf. High Performance Computing, Networking, Storage and Analysis*, Nov. 2011, pp.39:1–39:11.

[48] Dulloor S R, Kumar S, Keshavamurthy A, Lantz P, Reddy D, Sankaran R, Jackson J. System software for persistent memory. In *Proc. the 9th European Conf. Computer Systems*, April 2014, Article No. 15.

[49] Volos H, Tack A J, Swift M M. Mnemosyne: Lightweight persistent memory. In *Proc. the 16th Int. Conf. Architectural Support for Programming Languages and Operating Systems*, March 2011, pp.91–104.

[50] Coburn J, Caulfield A M, Akel A, Laura M, Gupta R K, Jhala R, Swanson S. NV-Heaps: Making persistent objects fast and safe with next-generation, non-volatile memories. In *Proc. the 16th Int. Conf. Architectural Support for Programming Languages and Operating Systems*, March 2011, pp.105–118.

[51] Liu R S, Shen D Y, Yang C L, Yu S C, Wang C Y M. NVM Duet: Unified working memory and persistent store architecture. In *Proc. the 19th Int. Conf. Architectural Support for Programming Languages and Operating Systems*, Feb. 2014, pp.455–470.

[52] Chung H, Jeong B H, Min B *et al.* A 58nm 1.8V 1Gb PRAM with 6.4MB/s program BW. In *Digest of Technical Papers of IEEE Int. Solid-State Circuits Conf.*, Feb. 2011, pp.500–502.

[53] Jiang L, Zhang Y, Yang J. ER: Elastic RESET for low power and long endurance MLC based phase change memory. In *Proc. ACM/IEEE Int. Symp. Low Power Electronics and Design*, Aug. 2012, pp.39–44.

[54] Alameldeen A R, Wood D A. Adaptive cache compression for high-performance processors. In *Proc. the 31st Annual Int. Symp. Computer Architecture*, Jun. 2004, pp.212–223.

[55] Zhang W, Li T. Helmet: A resistance drift resilient architecture for multi-level cell phase change memory system. In *Proc. the 41st IEEE/IFIP Int. Conf. Dependable Systems Networks*, Jun. 2011, pp.197-208.

[56] Zhao P, Zhu L. A scheme for protecting confidentially of non-volatile main memory based on phase-change memory. *Chinese Journal of Computers*, 2011, 34(11):2114–2120. (in Chinese)

[57] Jacobvitz A N, Calderbank R, Sorin D J. Coset coding to extend the lifetime of memory. In *Proc. the 19th IEEE Int. Symp. High Performance Computer Architecture*, Feb. 2013, pp.222–233.

[58] Bock S, Childers B, Melhem R, Mosse D, Zhang Y. Analyzing the impact of useless write-backs on the endurance and energy consumption of PCM main memory. In *Proc. IEEE Int. Symp. Performance Analysis of Systems and Software*, Apr. 2011, pp.56-65.

[59] Fang Y, Li H, Li X. SoftPCM: Enhancing energy efficiency and lifetime of phase change memory in video applications via approximate write. In *Proc. the 21st IEEE Asian Test Symp.*, Nov. 2012, pp.131-136.

[60] Zhang Y, Yang J, Gupta R. Frequent value locality and value-centric data cache design. In *Proc. the 9th Int. Conf. Architectural Support for Programming Languages and Operating Systems*, Mar. 2000, pp.150–159.

[61] Sun G, Niu D, Ouyang J, Xie Y. A frequent-value based PRAM memory architecture. In *Proc. the 16th Asia and South Pacific Design Automation Conf.*, Jan. 2011, pp.211–216.

[62] Baek S, Lee H G, Nicopoulos C, Kim J. A dual-phase compression mechanism for hybrid DRAM/PCM main memory architectures. In *Proc. the 22nd Great Lakes Symp. VLSI*, May 2012, pp.345–350.

[63] Du Y, Zhou M, Childers B, Melhem R, Mossé D. Delta-compressed caching for overcoming the write bandwidth limitation of hybrid main memory. *ACM Transaction Architecture Code Optimization*, 2013, 9(4): 55:1–55:20.

[64] Ferreira A P, Zhou M, Bock S, Childers B, Melhem R, Mosse D. Increasing PCM main memory lifetime. In *Proc. Design, Automation & Test in Europe Conf. & Exhibition*, Mar. 2010, pp.914-919.

[65] Rodriguez-Rodriguez R, Castro F, Chaver D, Pinuel L, Tirado F. Reducing writes in phase-change memory environments by using efficient cache replacement policies. In *Proc. Design, Automation & Test in Europe Conf. & Exhibition*, Mar. 2013, pp.93-96.

[66] Zhang X, Hu Q, Wang D, Li C, Wang H. A read-write aware replacement policy for phase change memory. In *Advanced Parallel Processing Technologies*, Teman O, Yew P C, Zang B (eds), Springer, 2011, pp.31–45.

[67] Barcelo N, Zhou M, Cole D, Nugent M, Pruhs K. Energy efficient caching for phase-change memory. In *Proc. the 1st Conf. Algorithms*, Dec. 2012, pp.67–81.

[68] Zhou M, Du Y, Childers B, Melhem R, Mossé D. Writeback-aware partitioning and replacement for last-level caches in phase change main memory systems. *ACM Transaction Architecture Code Optimization*, 2012, 8(4): 53:1–53:21.

[69] Chen S, Gibbons P B, Nath S. Rethinking database algorithms for phase change memory. In *Proc. the 5th Biennial Conference on Innovative Data Systems Research*, Jan. 2011, pp.21–31.

[70] Hu J, Xue C J, Zhuge Q, Tseng W C, Sha E H M. Write activity reduction on non-volatile main memories for embedded chip multiprocessors. *ACM Transaction on Embedded Computer System*, 2013, 12(3): Article No. 77.

[71] Zhang W, Li T. Characterizing and mitigating the impact of process variations on phase change based memory systems. In *Proc. the 42nd Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2009, pp.2–13.

[72] Dong J, Zhang L, Han Y, Wang Y, Li X. Wear rate leveling: Lifetime enhancement of PRAM with endurance variation. In *Proc. the 48th ACM/EDAC/IEEE Design Automation Conf.*, Jun. 2011, pp.972–977.

[73] Yun J, Lee S, Yoo S. Bloom filter-based dynamic wear leveling for phase-change RAM. In *Proc. Design, Automation & Test in Europe Conf. & Exhibition*, Mar. 2012, pp.1513-1518.

[74] Chen C H, Hsiu P C, Kuo T W, Yang C L, Wang C Y M. Age-based PCM wear leveling with nearly zero search cost. In *Proc. the 49th ACM/EDAC/IEEE Design Automation Conf.*, Jun. 2012, pp.453–458.

[75] Qureshi M K, Karidis J, Franceschini M, Srinivasan V, Lastras L, Abali B. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Proc. the 42nd Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2009, pp.14–23.

[76] Seong N H, Woo D H, Lee H H S. Security refresh: Prevent malicious wear-out and increase durability for phase-change memory with dynamically randomized address mapping. In *Proc. the 37th Int. Symp. Computer Architecture*, Jun. 2010, pp.383–394.

[77] Akel A, Caulfield A M, Mollov T I, Gupta R K, Swanson S. Onyx: A prototype phase change memory storage array. In *Proc. the 3rd USENIX Workshop on Hot Topics in Storage and File systems*, Jun. 2011.

[78] Koltsidas I, Pletka R, Mueller P *et al.* PSS: A prototype storage subsystem based on PCM. In *Proc. the 5th Annual Non-Volatile Memories Workshop*, Mar. 2014.

[79] Schechter S, Loh G H, Straus K, Burger D. Use ECP, not ECC, for hard failures in resistive memories. In *Proc. the 37th Annual Int. Symp. Computer Architecture*, Jun. 2010, pp.141–152.

[80] Qureshi M K. Pay-As-You-Go: Low-overhead hard-error correction for phase change memories. In *Proc. the 44th Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2011, pp.318–328.

[81] Yoon D H, Muralimanohar N, Chang J, Ranganathan P, Jouppi N P, Erez M. FREE-p: Protecting non-volatile memory against both hard and soft errors. In *Proc. the 17th Int. Symp. High Performance Computer Architecture*, Feb. 2011, pp.466–477.

[82] Jiang L, Du Y, Zhang Y, Childers B R, Yang J. LLS: Cooperative integration of wear-leveling and salvaging for PCM main memory. In *Proc. the 41st IEEE/IFIP Int. Conf. Dependable Systems and Networks*, Jun. 2011, pp.221-232.

[83] Seong N H, Woo D H, Srinivasan V, Rivers J A, Lee H H S. SAFER: Stuck-at-fault error recovery for memories. In *Proc. the 43rd Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2010, pp.115–124.

[84] Fan J, Jiang S, Shu J *et al.* Aegis: Partitioning data block for efficient recovery of stuck-at-faults in phase change memory. In *Proc. the 46th Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2013, pp.433–444.

[85] Ipek E, Condit J, Nightingale E B, Burger D, Moscibroda T. Dynamically replicated memory: Building reliable systems from nanoscale resistive memories. In *Proc. the 15th Int. Conf. Architectural Support for Programming Languages and Operating Systems*, Mar. 2010, pp.3–14.

[86] Chen J, Venkataramani G, Huang H H. RePRAM: Recycling PRAM faulty blocks for extended lifetime. In *Proc. the 42nd Annual IEEE/IFIP Int. Conf. Dependable Systems and Networks*, Jun. 2012.

[87] Azevedo R, Davis J D, Strauss K, Gopalan P, Manasse M, Yekhanin S. Zombie memory: Extending memory lifetime by reviving dead blocks. In *Proc. the 40th Annual Int. Symp. Computer Architecture*, Jun. 2013, pp.452–463.

[88] Mutlu O. Memory systems in the many-core era: Challenges, opportunities, and solution directions. In *Proc. Int. Symp. Memory Management*, Jun. 2011, pp.77–78.

[89] Chatterjee N, Shevgoor M, Balasubramonian R, Davis A, Fang Z, Illikkal R, Iyer R. Leveraging heterogeneity in DRAM main memories to accelerate critical word access. In *Proc. the 45th Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2012, pp.13–24.

[90] Artes A, Ayala J L, Huisken J, Catthoor F. Survey of low-energy techniques for instruction memory organisations in embedded systems. *Journal of Signal Processing Systems*, 2013, 70(1): 1–19.

[91] Xu W, Liu J, Zhang T. Data manipulation techniques to reduce phase change memory write energy. In *Proc. ACM/IEEE Int. Symp. Low Power Electronics and Design*, Aug. 2009, pp.237–242.

[92] Mirhoseini A, Potkonjak M, Koushanfar F. Coding-based energy minimization for phase change memory. In *Proc. the 49th ACM/EDAC/IEEE Design Automation Conf.*, Jun. 2012, pp.68–76.

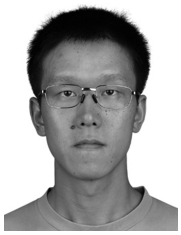
[93] Chen J, Chiang R C, Huang H H, Venkataramani G. Energy-aware writes to non-volatile main memory. *ACM SIGOPS Operating Systems Review*, 2012, 45(3): 48–52.

[94] Yue J, Zhu Y. Exploiting subarrays inside a bank to improve phase change memory performance. In *Proc. Design, Automation & Test in Europe Conf. & Exhibition*, Mar. 2013, pp.386-391.

[95] Wang J, Dong X, Sun G, Niu D, Xie Y. Energy-efficient multi-level cell phase-change memory system with data encoding. In *Proc. the 29th IEEE Int. Conf. Computer Design*, Oct. 2011, pp.175-182.

[96] Zhao J, Li S, Yoon D H, Xie Y, Jouppi N P. Kiln: Closing the performance gap between systems with and without persistence support. In *Proc. the 46th Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2013, pp.421–432.

Fei Xia received his B.S. degree in computer science and technology from Xi'an Jiaotong Unversity in 2011. He is now a Ph.D. candidate in the Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His research interests include computer architecture, non-volatile memory system, and storage systems.



De-Jun Jiang received his B.S. degree in electronic engineering from Beihang University, M.S. degree in software engineering from Tsinghua University, and Ph.D. degree in computer science from VU University Amsterdam, Netherlands. He is currently an assistant professor of Institute of Computing Technology, Chinese Academy of Sciences. His current research interests include storage system, non-volatile memory, cloud computing, and virtualization. He is a member of CCF, ACM, and IEEE.

Jin Xiong received her B.S. degree from Sichuan University, Chengdu, in 1990, M.S. degree from Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, in 1993, and Ph.D. degree from Graduate University of Chinese Academy of Sciences, Beijing, in 2006, all in computer science. She is currently a professor at ICT, CAS. Her research interests include big data storage and management, storage systems, and file systems. She is a senior member of CCF and a member of ACM and IEEE.

Ning-Hui Sun is a professor and the director of Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). He received his B.S. degree from Peking University in 1989 and M.S. and Ph.D. degrees both in computer science from ICT, CAS in 1992 and 1999, respectively. He is the architect and main designer of the Dawning series high performance computers, from Dawning2000 to Dawning Nebulae. His research interests include computer architecture, operating system, and parallel algorithm. He is a fellow of CCF and a member of ACM and IEEE.