Hindawi
Mobile Information Systems
Volume 2022, Article ID 9552517, 19 pages
https://doi.org/10.1155/2022/9552517

Research Article

PCM-2R: Accelerating MLC PCM Writes via Data Reshaping and Remapping

Feng Hong,[1] **Jianquan Zhang**,[2,3] **Shigui Qi**,[4] **and Zheng Li**[5]

[1]*Institute of Electronics and Information Engineering, Hubei University of Science and Technology, Xianning, China*
[2]*Institute of Engineering and Technology, Hubei University of Science and Technology, Xianning, China*
[3]*Hubei Xiangcheng Intelligent Electromechanical Industry Technology Research Institute Co Ltd, Xianning, China*
[4]*Henan University of Economics and Law, Zhengzhou, China*
[5]*Huawei Technologies Co Ltd, Shenzhen, China*

Correspondence should be addressed to Jianquan Zhang; zhangjianquan@hbust.edu.cn

Received 7 May 2022; Revised 14 May 2022; Accepted 26 May 2022; Published 16 July 2022

Academic Editor: M. Praveen Kumar Reddy

Multilevel cell (MLC) phase change memory (PCM) shows great potential in terms of capacity and cost compared with single-level cell (SLC) PCM by storing multiple bits in one physical PCM cell. However, poor write performance is a huge challenge for MLC PCM. In general, write latency of MLC PCM is 10 to 100X longer compared with DRAM technology. Considerable write latency greatly degrades the overall system performance and restricts the application of MLC PCM. Actually, several chips compose a memory DIMM to match the wide interface of data bus. The data of a write request, i.e., a cache line block, are distributed to multiple PCM chips. As a result, the write service time is determined by the chips with the most data amount. Conventional PCM write schemes do not care for the modified-byte distribution among PCM chips and it just waits for the completion of the chip with the most amount of data. However, it is observed that (1) the conventional PCM write scheme suffers from unbalanced modified-byte distribution that some PCM chips bear too many modified bytes while some chips are kept idle for long times. (2) The modified-byte distribution shows some unique patterns that some bytes are changed more frequently compared with others. (3) MLC PCM shows significant asymmetry considering only MSB or LSB transitions. Based on these observations, in order to solve the poor write problem of PCM, this article presents a novel PCM write scheme called PCM-2R. The key ideas behind our proposed scheme are to reshape the data to evenly distribute the cache line blocks among all chips based on their modified-byte distribution pattern to avoid unbalanced distribution and then remap modified bytes to fast region after decoupling MLC PCM cells considering the state transition asymmetries. The evaluation results show that PCM-2R achieves 51% read latency reduction, 37% write latency reduction, 1.9X IPC improvement, 41% running time reduction, 2.2X throughout improvement, and 52% energy reduction compared with the baseline. Moreover, compared with the state-of-the-art write schemes, PCM-2R achieves 0.2X more IPC improvement and 0.2X throughout improvement.

1. Introduction

Modern data-intensive applications have exhibited a growing demand for storage-class memory, e.g., Internet of Things (IOT) and big data analytics. Phase Change Memory (PCM) is one of the most promising memory technologies since conventional DRAM technology faces some great challenges in scalability, energy consumption, and stability under the finer feature size [1–3]. Despite the advantages of nonvolatile and low energy consumption, multilevel cell (MLC) PCM shows great potential in terms of capacity and cost compared with single-level cell (SLC) PCM by storing multiple bits in one physical PCM cell [4, 5]. However, poor write performance and endurance are huge challenges for implementing PCM into the memory hierarchy [6–8]. In general, write latency of MLC PCM is 4 to 8X slower and 10 to 100X longer compared with SLC PCM and DRAM technology, respectively [9, 10]. The considerable write latency greatly degrades the overall system performance and restricts the application of MLC PCM. There are several

disadvantages and limitations in PCM write [11]. Firstly, due to the complex program-and-verify scheme, the programming time is quite long. Secondly, due to the high write power consumption constraint, the number of cells that can be written in parallel is restricted [12–14]. As a result, data can only be written in write division mode and it takes several write operations to finish the whole write request [15–17]. These limitations may lead to potential performance degradation if we add PCM to the memory hierarchy. In general, the computer system is usually organized as a hierarchical architecture, including CPUs, caches, main memory, etc. The exchange unit between LLC (last level cache) and main memory is cache line. In order to match the wide interface of data bus, several chips compose a memory DIMM. The data of a write request are distributed to multiple PCM chips. As we mentioned, the PCM-based main memory cannot program all data once due to circuit constraints and the size of cache line block is much larger than the size of the operation unit. As a result, it takes several write operations for each chip to finish the data write and the write service time is determined by the chip with the most data amount [18].

Conventional PCM write schemes do not care for the modified-byte distribution among PCM chips and it just waits for the completion of the chip with the most amount of data. However, we observe the following:

(i) The modified-byte distribution is quite unbalanced that some PCM chips bear too many modified bytes and the other chips are not. As a result, the idle chips have to wait for the busy chips and are kept idle for long times, which causes consequent low parallelism and high latency.

(ii) The modified-byte distribution shows some unique characteristics, such as cyclical, etc. In general, some bytes are changed more frequently compared with others and can be summarized into several common patterns, i.e., first half concentrated, second half concentrated, crossed, and others.

In addition, we also observe the following:

(iii) MLC PCM shows significant asymmetry considering only MSB or LSB transitions. The worst transition latency of LSB is only 0.84X compared with the latency of MSB.

Based on these key observations, to effectively reduce the write latency and improve the write parallelism of MLC PCM-based memory, we present a novel PCM write scheme called PCM-2R. The proposed scheme reshapes the data to evenly distribute the cache line block among all chips based on its modified-byte distribution pattern to avoid unbalanced distribution and then remaps modified bytes in fast region after decoupling MLC PCM cells considering the state transition asymmetries. The main contributions of this work are summarized as follows:

(i) We detect the modified-byte distribution patterns inside cache line write service and summarize it into several command cases. In general, the modified-byte distribution shows significantly cyclical pattern and the cycle length is typically 8 bytes. The modified-byte distribution shows some unique patterns that some bytes are changed more frequently compared with others. In addition, the modified-byte distribution is quite unbalanced under conventional scheme that some PCM chips bear too many modified bytes.

(ii) We propose a data reshaping method to distribute the modified bytes evenly and avoid unbalanced distribution among PCM chips to boost the write parallelism. Based on the three special modified-byte distribution patterns, data reshaping scheme reconstructs the relationship between the chips and data.

(iii) We propose a data remapping method to leverage MLC write asymmetries in state transition latency and remap modified bytes to fast region to reduce the write service time. The cache line is divided into fast write half line and slow write half line by decoupling MLC cells. Since all modified bytes are concentrated on the front position of the cache line after data reshaping, our proposed scheme can further improve the write performance via remapping most data to the fast region.

(iv) We build a multicore computer system on the full-system simulators and the evaluation results show that PCM-2R decreases 51% read latency and 37% write latency and earns 1.9X IPC improvement, 41% running time reduction, 2.2X throughput improvement, and 51% energy reduction compared with the baseline under multiprogrammed SPEC 2006 and multithreaded PARSEC 2.0 workloads. In addition, compared with the state-of-the-art write schemes [19, 20], PCM-2R achieves 0.2X more IPC improvement and 0.2X throughout improvement.

The remainder of this paper is structured as follows. Section 2 describes the background and presents the motivations and the key observations of this study. Section 3 introduces the architecture and implementation of proposed schemes. Section 4 presents and analyzes the experimental results. Section 5 summaries the related work. Finally, Section 6 offers conclusions.

2. Background and Motivations

2.1. Background. Phase Change Memory (PCM) is an extremely promising competitor to DRAM for some outstanding features such as better technology scaling, nonvolatility, and low idle power consumption [21]. Different from charge-based DRAM technology, PCM takes advantage of resistance of phase change material (chalcogenide glass, such as Ge2Sb2Te5) to store digital information and it does not need to do refresh operations. In general, amorphous state material exhibits higher resistance compared with the crystalline state material. By applying different intensity of current pulses and programming time to

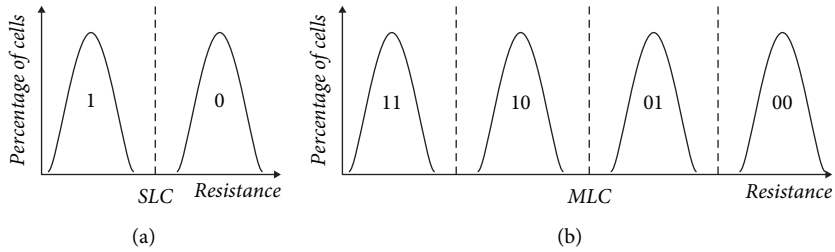

Figure 1: Resistance distributions of (a) SLC and (b) MLC PCM cells.

the phase change material, the chalcogenide material can be manipulated between the amorphous state and the crystalline state [22, 23].

As shown in Figure 1, PCM utilizes the tight resistance ranges to present digital information (logical "0" or "1"). The switching of logic "0" to "1" is defined as SET operation and the switching of logic "1" to "0" is defined as RESET operation. Single-level cell (SLC) stores only one bit inside one physical PCM cell and it takes two resistance levels in total. It is practical to store multiple bits in one cell by dividing more resistance levels, such as the multilevel cell (MLC) technology. Multilevel cell (MLC) PCM shows great potential in terms of capacity and cost compared with single-level cell (SLC) PCM by storing multiple bits in one PCM cell [24].

Program-and-verify (P&V) write strategy (or iterative write algorithm) is widely used in MLC PCM cell programming. According to the previous art Mercury [25], the write approaches can be concluded into two typical kinds, i.e., SET to RESET (referred to as S2R) and RESET to SET (referred to as R2S), as shown in Figure 2. By leveraging the adaptive programming techniques named State-aware Adaptive Programming, the transitions between different MLC cell states can be realized using S2R or R2S methods. R2S programming involves multiple short RESET pulses and each pulse gradually reduces the cell resistance. The RESET pulses continue until the cell reaches the given resistance value. In constant, S2R programming starts with a long SET pulse and the material is set to a lower resistance level state. After that, the resistance level is gradually increased with individual RESET pulses. In general, different MLC cell states take use of adaptive programming schemes and the service time depends on the initial state and the destination state, as shown in Table 1. Programming of PCM cells is more elaborate and complex than DRAM technology. As a result, the programming time is much longer. Even worse, MLC PCM involves more narrow resistance ranges of each data value, which increases the difficulty of programming and greatly expands the operation delay. Poor write performance has become a huge challenge for MLC PCM. In general, write latency of MLC PCM is 10 to 100X longer compared with DRAM technology. This poor write degrades the overall system performance and greatly restricts the application of MLC PCM [26].

In this paper, we consider 2-bit MLC PCM that one cell stores two logic bits. According to Mercury model [25], the state transition latency is summarized in Figure 3. The latency is normalized to the "00" to "10" latency. We highlight two novel insights that (1) MLC PCM shows significant asymmetry between state transitions. The programming time varies much and depends on the initial state and the destination state. In general, write latency of middle states "01"/"10" is much larger than states "00"/"11." (2) MLC PCM shows significant asymmetry considering only MSB or LSB state transitions. The worst transition latency of LSB is only 0.84X compared with the latency of MSB, as shown in the red part in Figure 3.

There are several limitations in PCM write. The number of cells that can be programmed concurrently is restricted due to the high programming power consumption and write circuit constraints. As a result, the whole data can only be written in write division mode and it takes several write operations to finish the whole write request [13, 14]. When PCM is incorporated into the main memory, these limitations will further amplify the shortcomings of PCM. In the memory hierarchy, the exchange unit between LLC (last level cache) and main memory is cache line. In general, several chips compose a memory DIMM in order to match the wide interface of the data bus. The data of a write request, i.e., a cache line block, are distributed to multiple PCM chips to improve the write parallelism. However, the PCM-based main memory cannot program all data in one operation due to circuit constraints. As a result, it takes several write operations to complete the write service for the cache line and the write service time is determined by the chips with the most data amount.

2.2. Motivations and Observations. Conventional PCM write schemes use the naive DIMM organization and the cache line data are sequentially assigned to multiple chips. It does not care for the modified-byte distribution among PCM chips and it just waits for the completion of the chip with the most amount of data. Giving a simple example as shown in Figure 4, assuming the cache line size is 64 bytes, a memory DIMM is composed of four 16-bit wide PCM chips (the size of the write operation per chip is 16 bits); i.e., one chip can only operate two bytes in parallel. Under conventional byte distribution method, the cache line block is sequentially assigned to multiple chips. Conventional method does not care for the modified-byte distribution, which causes potential parallelism and performance degradation. In this example, chip 1 consumes 8 write units to complete the entire data write while the other chips take only 1 write unit to finish the service. Chip 0, chip 2, and chip 3 are idle and

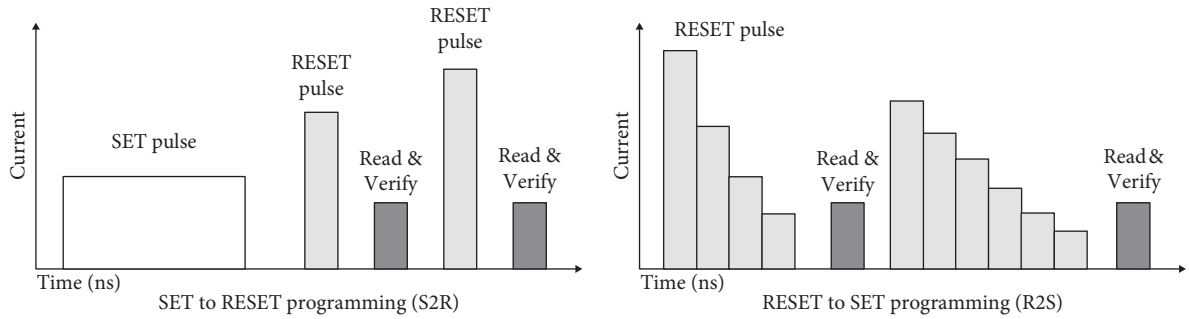

Figure 2: MLC PCM P&V strategies.

Table 1: MLC PCM state transitions.

Destination initial	00	01	10	11
00	—	Init + R2S	Init + R2S	Init + R2S
01	Init + R2S	—	Init + R2S	Init + R2S
10	Init + S2R	Init + S2R	—	S2R (without init)
11	Init + S2R	Init + S2R	Init + S2R	—

have to wait for the completion of chip 1 since it has the most data amount among all PCM chips. If the data are distributed evenly across all chips, the write parallelism can be significantly improved and the write latency can be greatly reduced.

In order to measure the degree of uneven distribution, we count the modified-byte distribution of memory requests (we assume the cache line is 64-byte size) and the results are shown in Figure 5. The detailed system parameters and benchmark information can be found in Section 4. We draw three key observations from the experimental results:

(1) *Cyclical*. The modified-byte distribution shows significantly cyclical pattern and the cycle length is typically 8 bytes.

(2) *Unique*. The modified-byte distribution shows some unique patterns that some bytes are changed more frequently compared with others.

(3) *Unbalanced*. The modified-byte distribution is quite unbalanced under conventional scheme that some PCM chips bear too many modified bytes.

Moreover, based on the MLC transition latency shown in Figure 3, we observe the following:

(4) *Asymmetries*. The worst transition latency of LSB is only 0.84X compared with the latency of MSB considering only MSB or LSB.

According to observation "cyclical," the cyclical pattern implies that if we can solve the problem in one or two cycles, the whole problem can be solved. Observation "unique" shows that if we can effectively deal with the problem according to the unique data distribution patterns, we can get significant endurance and performance improvement of PCM-based main memory. According to observation unbalanced, the quite unbalanced modified-byte distribution results in the circumstance that some PCM chips bear too many modified bytes. The write service time is determined by the chips with the most data amount, which causes write parallelism and performance degradation. Observation "asymmetries" hints that we can get further performance improvement by decoupling the PCM cells and distributing data to the fast regions (regions with only LSB state transitions).

3. PCM-2R Design

Keeping these key insights in mind, we present a novel write scheme named PCM-2R for MLC PCM-based memory. We first characterize the unique distribution patterns and then introduce our data reshaping and remapping methods as well as the architectural design.

3.1. Modified-Byte Distribution Patterns. According to our observations, the modified-byte distribution patterns can be classified into several cases, as shown in Table 2 (*C* refers to the byte that is changed and *U* refers to state unchanged).

(i) *First half concentrated*: This pattern implies that the modified bytes are mainly concentrated on the first half of the 8 bytes' cycle. The distribution profile shape is similar to letter "Z." In other words, the distribution of one cycle is like "CCCCUUUU," where "C" and "U" present that the byte is changed (modified) and unchanged (unmodified) compared with the original data, respectively.

(ii) *Second half concentrated*: This pattern is opposite to the first half concentrated that the modified bytes are concentrated on the second half of the 8 bytes' cycle. The distribution looks like "UUUUCCCC."

(iii) *Crossed*: The modified-byte distribution is crossed over the 8 bytes' cycle and the distribution type is "CCUUCCUU."

(iv) *Others*: The other remaining cases that we do not care for in this work are, for example, the uniform distribution and other irregular cases.

The percentage of distribution patterns is shown in Figure 6. We observe that these three distribution patterns occupy more than 20% of memory accesses in some workloads such as canneal, astar, bwaves, gobmk, leslie3d,

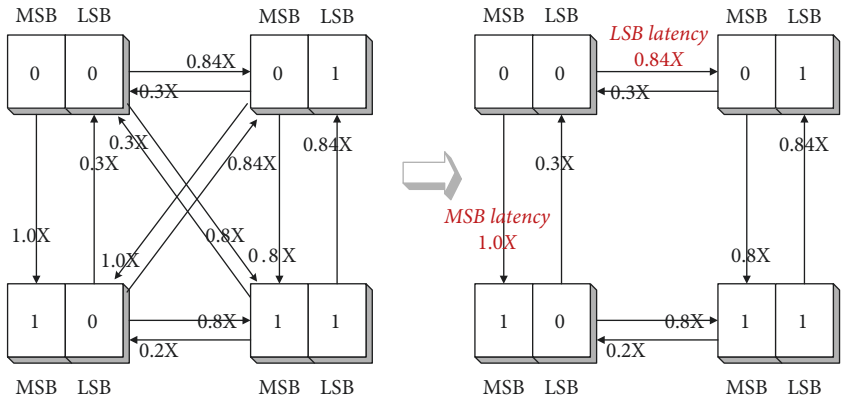

Figure 3: MLC PCM state transition latency according to prototype Mercury [25].

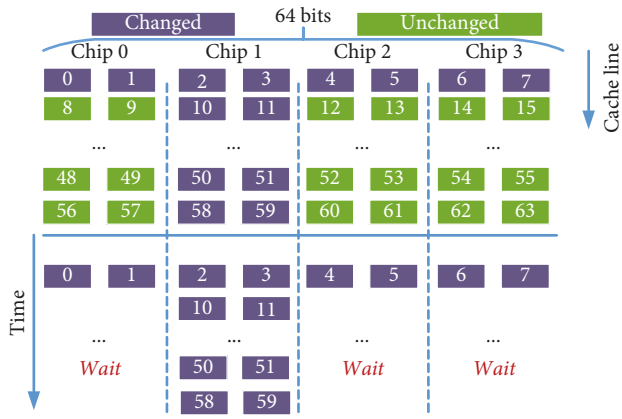

Figure 4: Conventional byte distribution and PCM write division scheme.

libquantum, and zeusmp. However, in some benchmarks, the circumstance is not obvious, such as dedup, bzip2, and x264. Since the modified-byte distribution shows some cyclical patterns, the unbalanced distribution results in potential performance and endurance problems. The service time is determined by the chip with most write work, which results in the parallelism degradation. In addition, some bytes are changed more frequently and these cells are more likely to be worn out. Keeping these typical patterns in mind, PCM-2R enhances the write parallelism and performance by leveraging data reshaping and remapping strategies.

3.2. Data Reshaping and Remapping

3.2.1. Data Reshaping. In order to avoid unbalanced data distribution, we propose a novel data reshaping method by leveraging the special characteristics shown in the modified-byte distribution. The primary job of reshaping scheme is to reconstruct the data and distribute the modified bytes to all chips evenly.

The data reshaping scheme is based on modified-byte distribution patterns and is divided into three cases, as shown in Figure 7. Under the conventional method, chip 0 corresponds to byte $(8 \times i)$ and byte $(8 \times i + 1)$, where $i = \{0, 1, 2, \ldots, 7\}$. Data reshaping scheme reconstructs the relationship between the chips and data. Since the distribution has periodic characteristics, we take 2 write cycles for example. In detail, (1) if the distribution pattern is first half concentrated, the proposed reshaping scheme uses static data reorganization that exchanges the second half of the first 8 bytes and the first half of the second 8 bytes. Under this circumstance, all chips have the same amount of modified bytes, i.e., 2 modified bytes, and the programming time of each time is almost equal. Relatively, the write latency is halved (two write units are reduced to one write unit) as shown in Figure 7(a). (2) If the distribution pattern is second half concentrated, similarly, our proposed reshaping method exchanges the first half of the first 8 bytes and the second half of the second 8 bytes. Thus, the modified-byte is evenly distributed to all chips, as shown in Figure 7(b). (3) If the distribution pattern is crossed, however, exchange occurs between the first half of first 8 bytes and second 8 bytes, and second half of first 8 bytes and second 8 bytes. Different from previous circumstances, we only exchange part of the data. As shown in Figure 7(c), bytes 2 and 3 are exchanged with bytes 8 and 9. In the same way, bytes 6 and 7 are exchanged with bytes 12 and 13. The data are equally distributed among the individual chips.

3.2.2. Data Remapping. Inspired by prior art [27–30], we propose a novel bit-mapping scheme named data remapping to improve write performance and reduce energy consumption. The primary goal of data remapping is to remap the data after reshaping to fast region of MLC PCM by decoupling the cells considering the asymmetry between MSB and LSB state transitions. For illustrative purpose, we assume the cache line is 512 bits with 256 physical cells. The conventional bit-mapping scheme is shown in Figure 8. In general, one PCM cell stores two consequent bits and the whole cache line is distributed in continuous physical PCM cells. Under this circumstance, writing a cache block involves all of the bits (both the LSB and MSB bits) of the 256 PCM cells. Recently, SPCM mapping scheme [19, 20] is proposed to utilize the asymmetries of MLC PCM. By consolidating the physical units of the two cache lines, the memory lines are divided into two parts, i.e., one stored in MSBs in two PCM physical lines and the other stored in

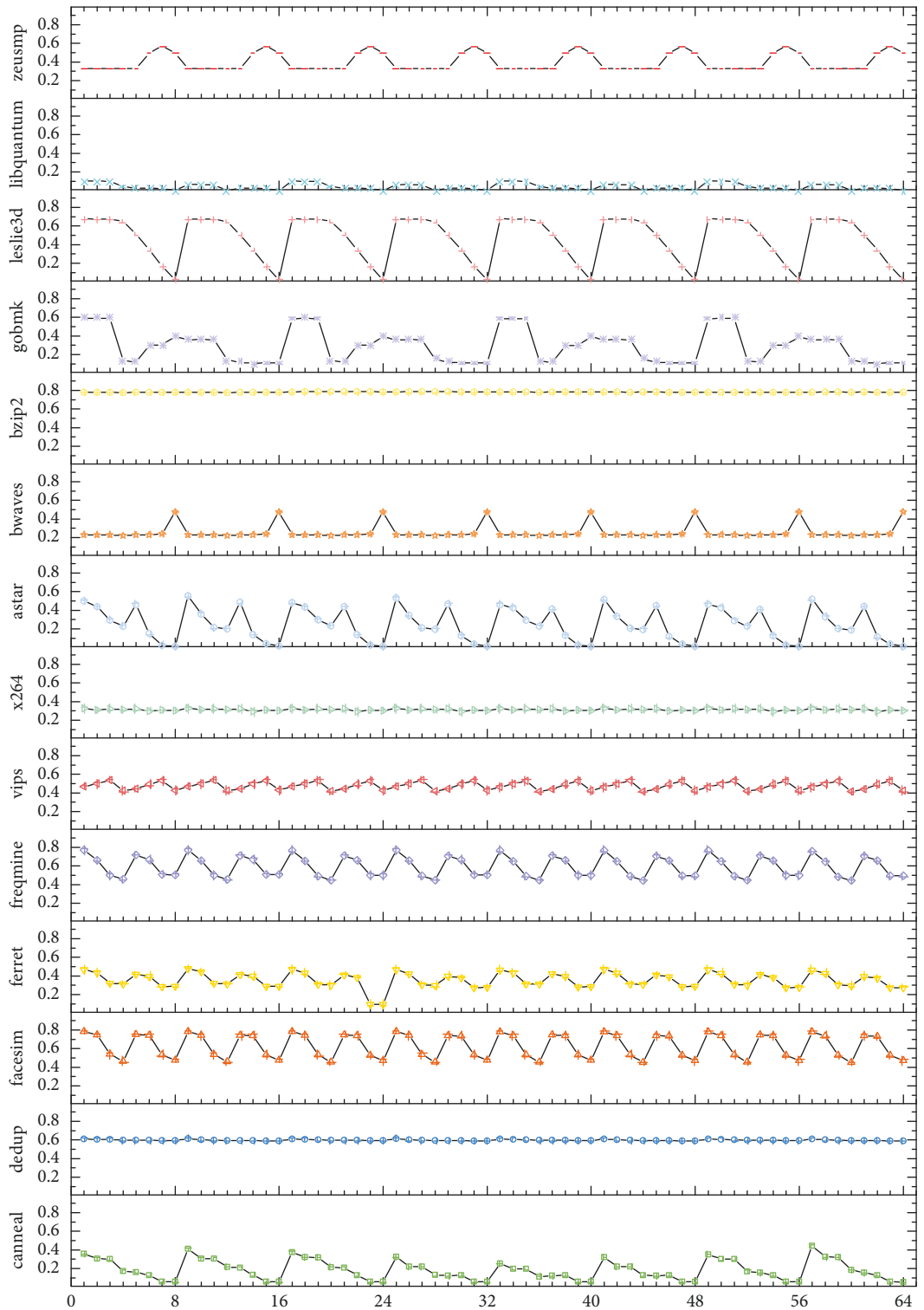

Figure 5: Modified-byte distribution of memory requests using SPEC CPU2006 and PARSEC 2.0 benchmarks. Each cache line is composed of 64 bytes. *Y*-axis represents the proportion of modified bytes of different benchmarks.

LSBs in two PCM lines. As shown in Figure 8, cache line block A is stored in the MSBs of two physical lines and cache line block B involves all LSBs of two physical lines. Since the LSBs write latency is much shorter than MSBs latency, the write service time of cache block B is reduced. In other words, only half of the cache lines can be accelerated.

To explore and exploit the asymmetries between LSB and MSB transitions, we propose the data remapping scheme to speed up the write operation after implementing data reshaping scheme. As shown in Figure 8, our proposed remapping scheme involves only one physical PCM line. The cache line has both MSB and LSB bits similar to

Table 2: Modified-byte distribution patterns.

Patterns	Description	Examples	Instances
First half concentrated	Modified bytes are concentrated in the first half of each 8 bytes	CCCCUUUU	*leslie3d, gobmk, libquan-tum, canneal, and vips*
Second half concentrated	Modified bytes are concentrated in the second half of each 8 bytes	UUUUCCCC	*zeusmp and bwaves*
Crossed	Modified bytes are crossed in each 8 bytes	CCUUCCUU	*astar, freqmine, ferret, and facesim*
Others	The other remaining cases	—	*dedup, bzip2, and x264*

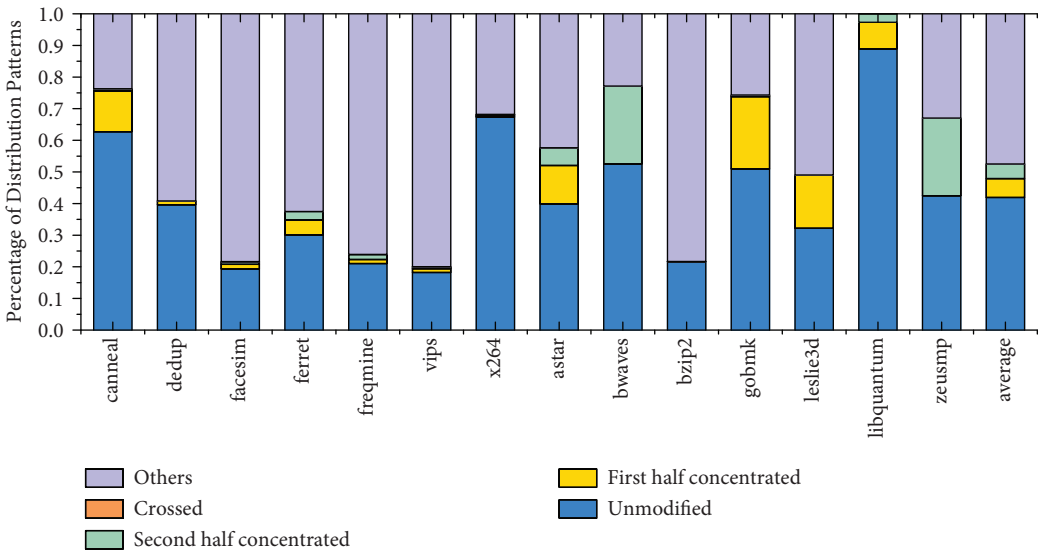

Figure 6: Percentage of distribution patterns using SPEC CPU2006 and PARSEC 2.0 benchmarks.

conventional cache line. Differently, our design exploits interleaved mapping, and the first half of the cache line is stored in the LSBs of one physical line while the second half of the cache line is stored in the MSBs of the same PCM line. In other words, the cache line is divided into fast write half line and slow write half line by decoupling MLC cells. In general, our design differs from the conventional scheme and SPCM in the following aspects. On one hand, unlike conventional mapping scheme, the continuous bits are mapped to different physical cells. Compared with SPCM, data remapping scheme involves only one physical line for one cache line while SPCM has to write two different PCM lines for one cache block. On the other hand, the first half of the cache line is composed of all LSB bits to leverage its fast transition speed (fast write half line). Since all modified bytes are centered on the front position of the cache line after data reshaping, our proposed scheme can further improve the write performance via remapping the data to the fast "region" for all cache lines. However, SPCM can only accelerate half of the cache line write based on the write address (odd address or even address).

As shown in Figure 9, conventional PCM write scheme involves 8 write units without considering the actual modified-byte patterns. PCM-2R tries to reshape the data according to its data pattern. After the data reshaping method (first half concentrated in this case), the data amount of all chips is balanced and most modified bytes are centered on the front position. The number of write units is reduced since there are no modified bytes in some write units after data reshaping. Then the first half cache line is remapped to fast write region and the second half is remapped to slow write region. Fast region has shorter write latency and overall write performance is accelerated again by leveraging data remapping scheme.

3.3. Architectural Design. The architectural design of PCM-2R, including proposed data reshaping and remapping schemes, is illustrated in Figure 10. In general, PCM-2R involves the innovations in both memory controller and PCM array. The memory controller-side includes write buffer, modified-byte distribution checking, data reshaping and write control modules for write operations and read buffer, and data reconstruction and read control modules for read operations. The PCM array-side includes write control logic, write circuits, row/column decoders, and physical cells.

In general, the DCW (Data-Comparison-Write) scheme [31] is adopted to reduce the energy consumption. During a write operation, DCW first reads out the old data and then writes the new data only when the old and new data are different. We add a middle layer named modified-byte distribution checking to provide necessary meta information to the data reshaping module designed in the memory controller. The memory controller first reads the old data stored in the PCM chips and compares the old data with the

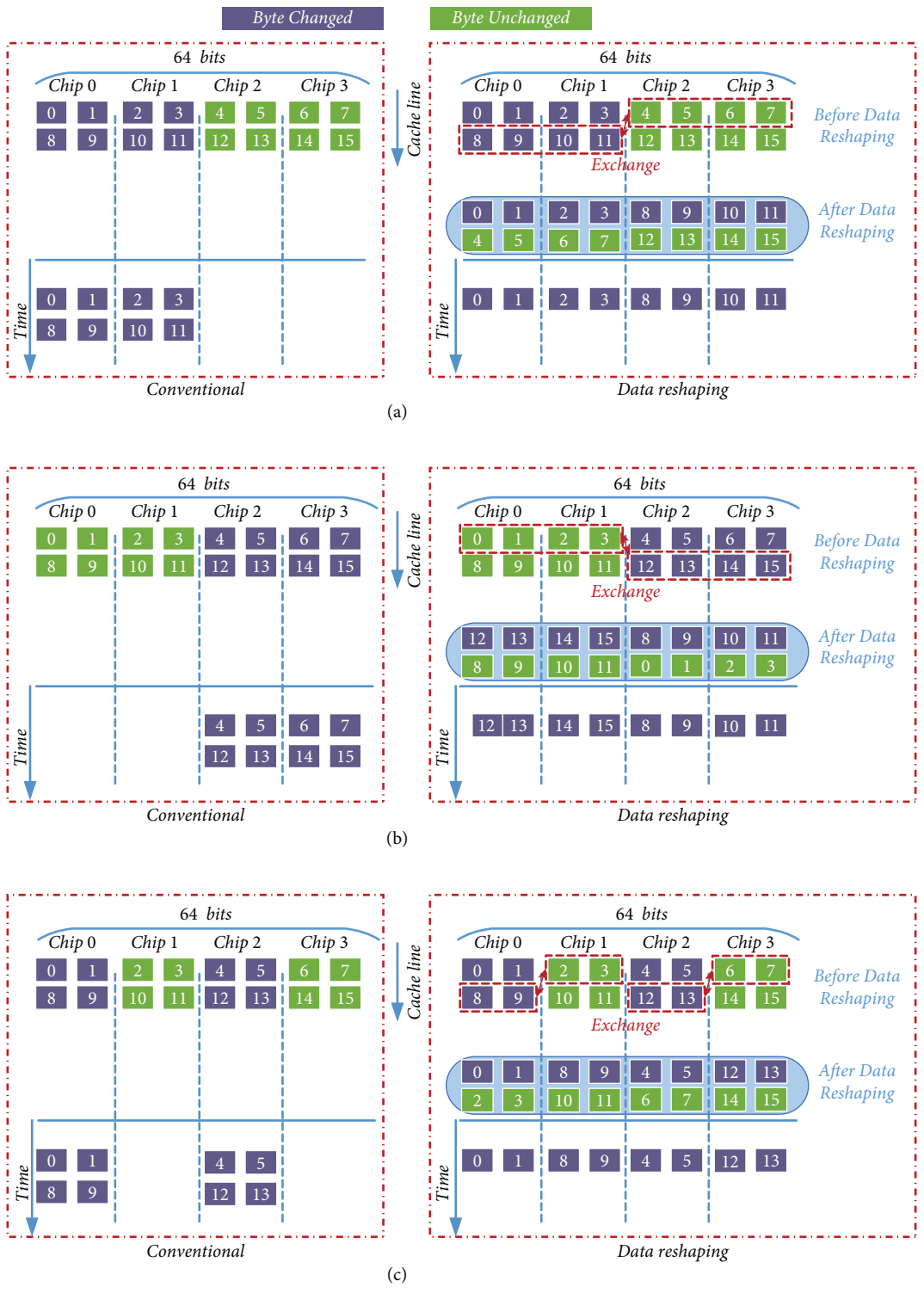

FIGURE 7: Data reshaping scheme based on different modified-byte distribution. (a) Data reshaping for first half concentrated; (b) data reshaping for second half concentrated; (c) data reshaping for crossed.

new data to be written and then records the modified-byte distribution. For the write request to be processed, the memory controller maintains a 64-bit array and each bit indicates the state of the 64 bytes' cache line data. If one byte is modified, i.e., the new data are different from the stored old data, the value is labeled with "1." In detail, PCM-2R uses the violent crack method and tries to find the most suitable data reshaping method or keep the original data. The modified-byte array is transformed according to each data reshaping method and our design estimates the number of write units of all proposed methods in parallel. For each data chunk (8 bytes per chunk and the cache line is 64 bytes), if

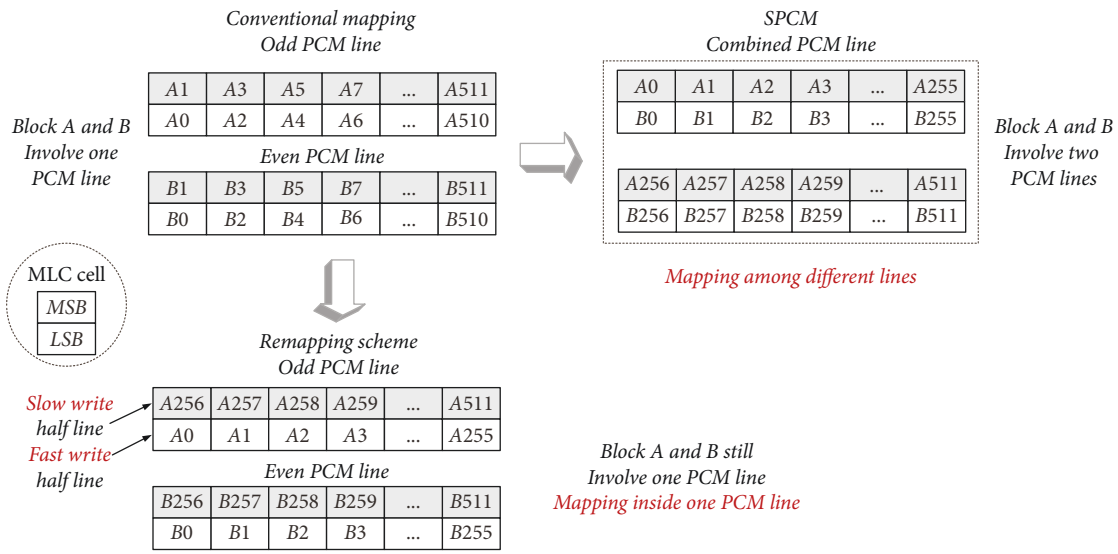

Figure 8: Remapping scheme.

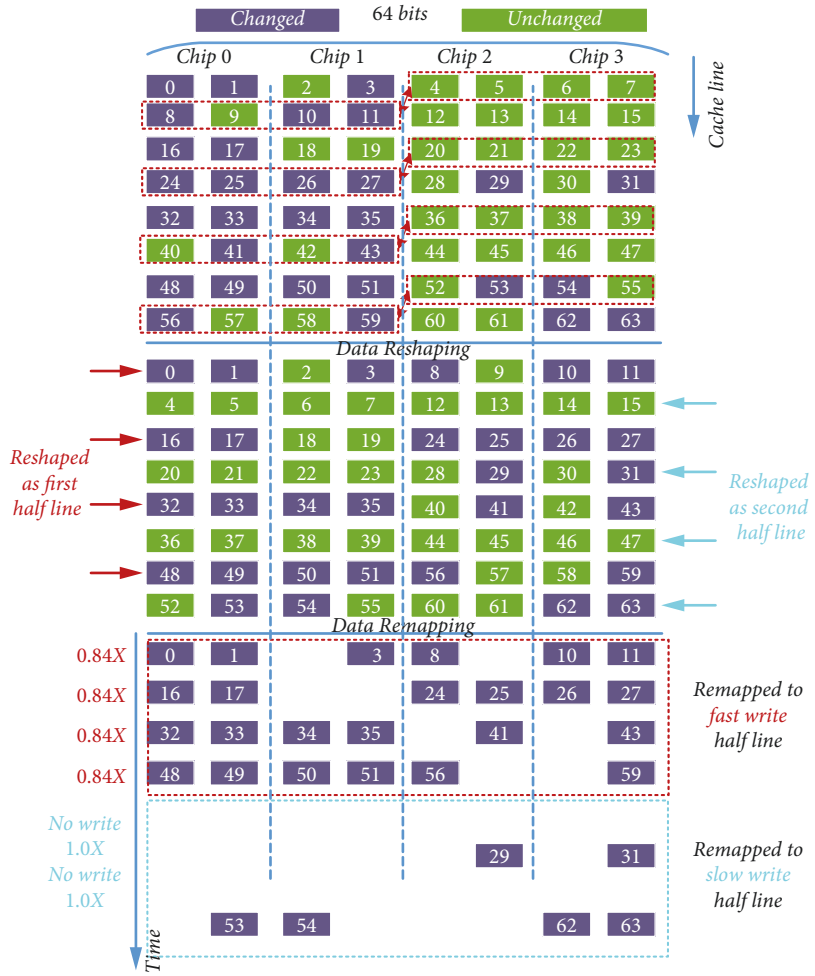

Figure 9: Putting data reshaping and remapping all together.

any bytes need to be modified, i.e., the corresponding bits in the modified-byte array are 1, the write unit counter increases by one. Through the prior estimates about the time consumption based on the modified-byte distribution results, the reshaping selection can be released into very simple operations, as shown in Figure 11. Once the data reshaping

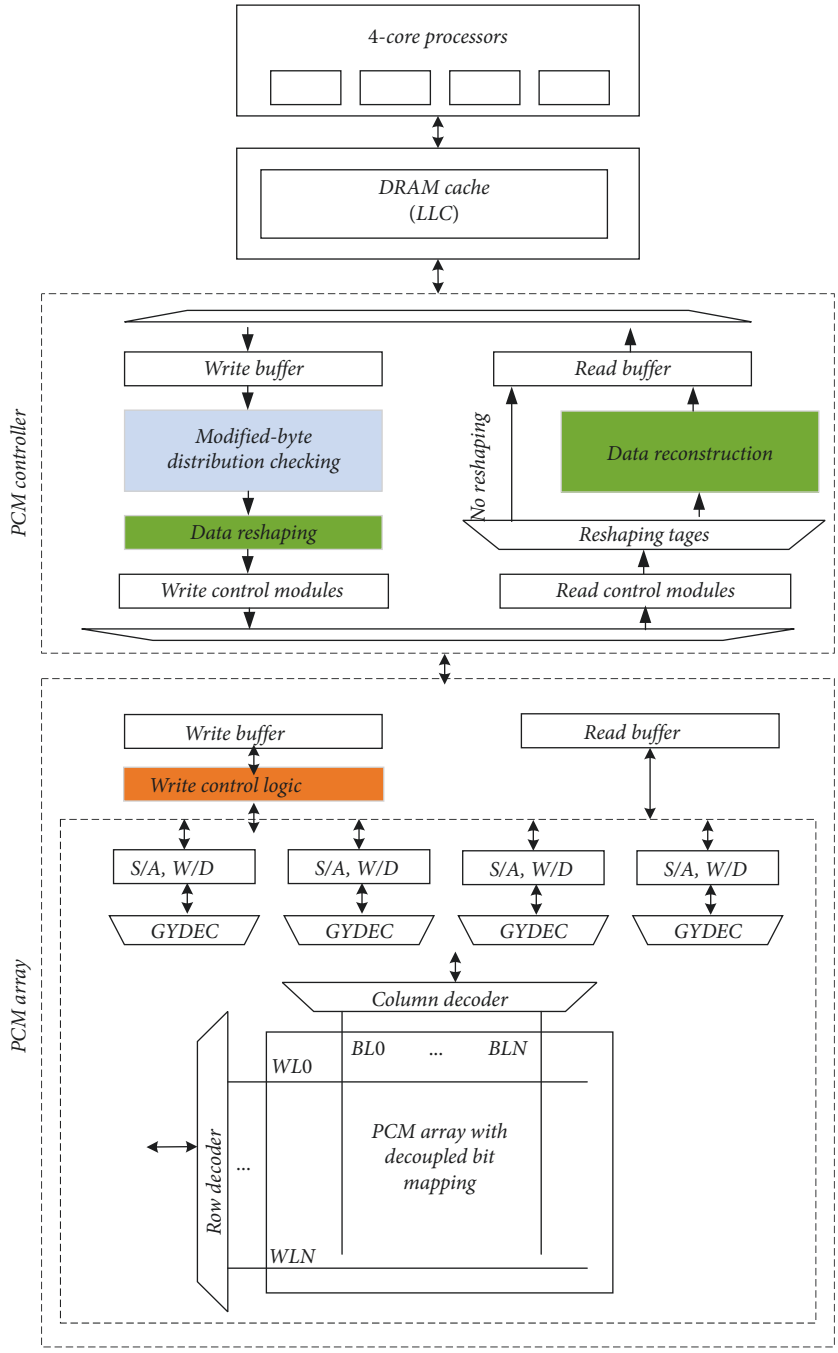

Figure 10: Architecture of PCM-2R.

mechanism is determined, an additional tag per cache line is introduced to indicate the pattern.

The data reshaping and reconstruction are relative, as shown in Figure 10. For write operation, the memory controller reshapes the data based on its pattern and stores the tags with reshaped data into the array. For read operation, the memory controller has to reconstruct the data based on the tags. Since the schemes involved in PCM-2R are quite simple, the data reshaping and reconstruction can be achieved through simple circuits, as, respectively, shown in Figures 12 and 13. The RTL designs of reshaping and reconstruction schemes involve only one multiplexer and several OBUFs; the time and space overhead caused by reshaping and reconstruction is small.

In addition, the data remapping mechanisms are designed at the hardware level and implemented in the MLC PCM array control logic. The key design of proposed remapping scheme is the write control logic. Write control logic determines which physical cells are adopted to store the data to be written. As shown in Figure 14, the write control logic includes clock, write buffer, finite state machine, and write signals generation modules. The finite state machine selectively chooses the data through the multiplexer and generates write control signals to the actual write circuits. By

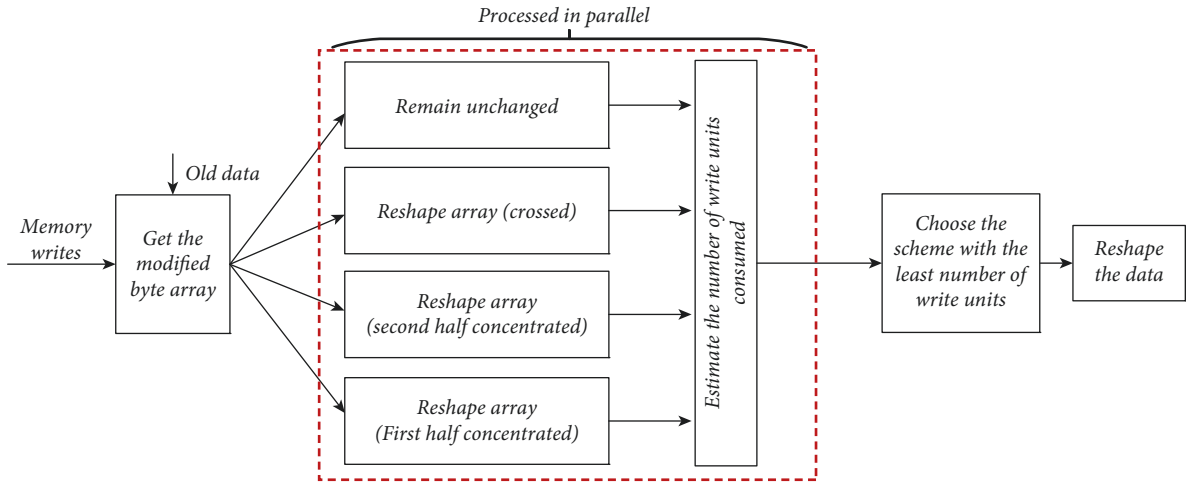

Figure 11: Reshaping methods selection.

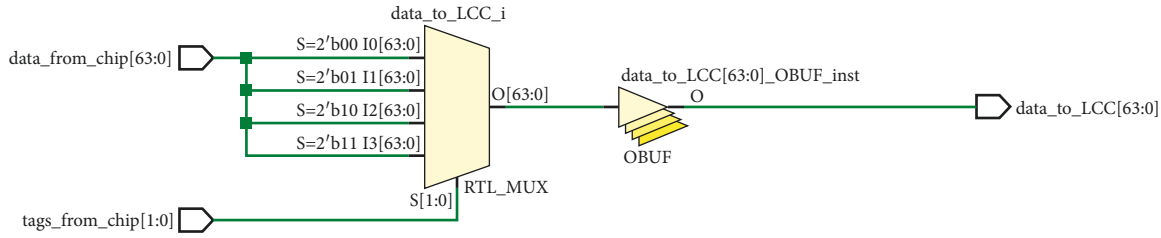

Figure 12: Reshaping RTL designs.

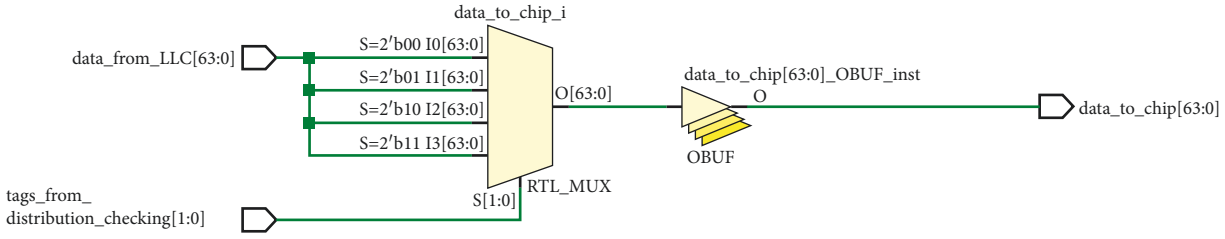

Figure 13: Reconstruction RTL designs.

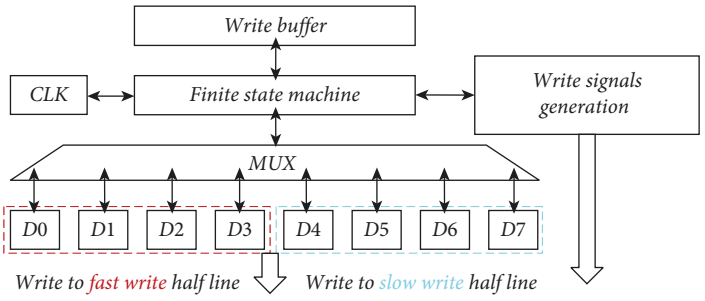

Figure 14: Write control logic.

leveraging the write control logic, the first half data are written into fast write half line and the second half data are written into slow write half line.

3.4. Overhead. In PCM-2R, a read operation is issued to compare the old data and the original data and triggers an extra time overhead. However, compared with the slow write operation, the overhead raised by an extra read operation is acceptable. It is worth noting that the key component of PCM-2R is the data reshaping mechanism selection, which affects the overall performance and introduces extra storage overhead. For each write request, the memory controller maintains a 64-bit array to indicate the state of the cache line

data. Since the calculation involved is not complicated and can be released into simple gate operations, the time overhead is quite low. Moreover, the data reshaping and reconstruction methods deliver 64 LUTS (look-up-tables) overhead according to the experimental results on Xilinx FPGA [32]. There are four selected distribution patterns in PCM-2R, so the space overhead is 2 bits per 64B cache line (0.4%). In addition, the reshaping scheme has 1.8 to 4.3 ns overhead and the reconstruction scheme has extra 1.8 to 5.2 ns overhead. Since the write control logic only controls the write data selection, the overhead to support the data control is not significant. It is worth noting that PCM-2R does not have any PCM DIMM or PCM array changes, so there is no extra time, area, and energy overhead.

4. Performance Evaluation

4.1. Experimental Environment. In order to evaluate the efficiency and effectiveness of the proposed schemes, we use the full-system GEM5 simulator [33] combined with the cycle-level NVMain module [34], which is a cycle accurate memory simulator for NVMs. We use the same configuration parameters from prior similar studies [35] and the evaluation parameters used in this study are shown in Table 3. In general, we simulate a 4-core system with ALPHA-like processors and model the whole memory hierarchy including L1, L2, last level DRAM caches, and 16 GB MLC PCM main memory. DRAM cache is adopted for improving the performance and mitigating the endurance of PCM-based main memory [36–38]. The main memory has 4 channels in total, with 2 ranks per channel and 8 banks per rank. The memory controller adopts the FRFCFS-WQF scheduling proposed by previous art [39, 40]. The workloads used in this study are summarized in Table 4. We adopt a set of multithreaded and multiprogrammed programs in PARSEC 2.0 [41] and SPEC 2006 suites [42], respectively. All of these benchmarks involve intensive and nonintensive memory accesses and have different memory Read Per Kilo Instructions (RPKI) and memory Write Per Kilo Instructions (WPKI) rates. In addition, five billion instructions are simulated for each workload after fast-forwarding one billion instructions considering program initialization and cache warm-up. In general, we use the DCW [31] as the baseline and combine the DCW with the silentstore scheme [43, 44] as the comparison. All the comparison techniques have the same technology node (22 nm) and simulation configurations. PCM-R combines proposed reshaping method with DCW while PCM-2R puts reshaping and remapping all together. We also try to compare our proposed designs with state-of-the-art bit-mapping [3] and SPCM designs [19, 20]. Specifically, the row buffer hit/miss is considered in the evaluation and the time overhead caused by data reshaping, reconstruction, and remapping is also involved in our evaluation.

4.2. Memory Read Latency. The write latency reduction significantly benefits the read latency due to the wait time reduction of read requests [45]. Although PCM-2R does not improve the read performance directly, it improves the write parallelism and read latency benefits from the write service time reduction. Figure 15 shows the experimental results of read latency. On average, PCM-R shows 48% read latency reduction compared with the baseline. In comparison, silentstore and bit-mapping + SPCM show, respectively, 40% and 45% read latency improvement and PCM-2R shows almost 51% read latency reduction by leveraging data remapping scheme. One workload shows significant performance improvement (facesim) and outperforms the state-of-the-art scheme by 25% more read latency reduction because the modified-byte pattern is obvious and the amount is balanced after data reshaping and remapping methods. It is noteworthy that our schemes are not effective in some workloads with balanced modified-byte distribution, such as dedup, bzip2, and x264. In other words, PCM-R and PCM-2R are not effective for all benchmarks and efficient for some workloads with special unbalanced distribution patterns.

Table 3: Parameters of simulation.

Parameter	Value
Processor	4 cores, 2 GHz, ALPHA-like
L1 cache	32 kB I-cache, 32 kB D-cache, 2 cycles' access latency
L2 cache	8-way, 2 MB private per core, 20 cycles' access latency
L3 DRAM cache	16-way, 32 MB shared, 50 cycles' access latency
Cache organization	64-byte line size, write back, LRU replacement
Memory controller	FRFCFS-WQF scheduling, 32-entry R/W queues
Memory organization	16 GB MLC PCM, 4 channels with 2 ranks each, 8 banks per rank
Memory timing	Read 250 ns, write 2 μs ("00" to "10," 1.0X)
PCM organization	4 chips per DIMM, the size of write unit is 64 bits (32 2 bit MLC cells)

Table 4: Characteristics of the evaluated workloads.

Workload	RPKI	WPKI
SPEC 2006, multiprogrammed		
Astar: 4 copies of astar	0.53	0.38
bwaves: 4 copies of bwaves	11.19	5.67
bzip2: 4 copies of bzip2	8.81	4.64
gobmk: 4 copies of gobmk	0.14	0.13
leslie3d: 4 copies of leslie3d	7.11	2.59
libquantum: 4 copies of libquantum	10.95	6.93
zeusmp: 4 copies of zeusmp	13.56	3.18
PARSEC 2.0, multithreaded		
canneal	1.88	1.31
dedup	0.35	0.33
facesim	0.86	0.71
ferret	0.23	0.15
freqmine	0.32	0.33
vips	1.27	1.18
x264	0.18	0.20

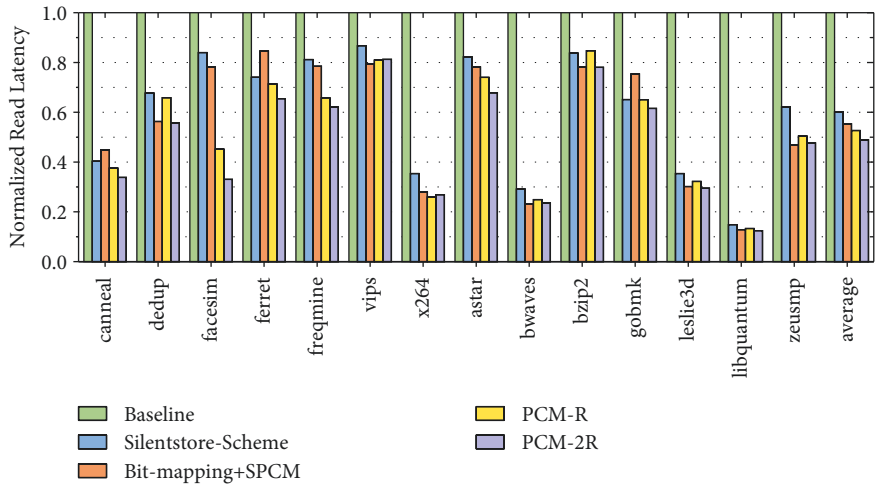

Figure 15: Read latency.

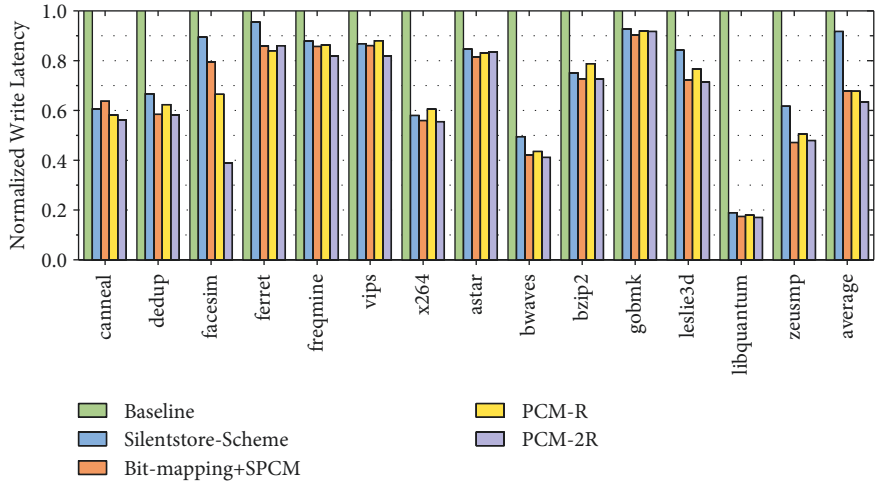

Figure 16: Write latency.

4.3. Memory Write Latency. Figure 16 illustrates the comparison of memory write latency of different PCM write schemes. The improvement of write latency is not significant compared with the increase of read latency since the memory controller we used in this study adopts the read priority scheduling and the write requests are delayed until reaching the HighWaterMark [39, 40]. On average, we observe that the write latency of PCM-R and PCM-2R is only 67% and 63% compared with the baseline, respectively. The write latency of silentstore is 78% of the baseline. In comparison, the write latency of bit-mapping + SPCM is 68% compared with the baseline and PCM-2R outperforms the state-of-the-art PCM write scheme greatly. Moreover, our designs achieve better performance improvement with the memory intensiveness increases with more obvious modified-byte distribution patterns (among the workloads canneal, facesim, astar, bwaves, leslie3d, and zeusmp). However, some workloads show less performance improvement, such as dedup, ferret, vips, x264, bwaves, and bzips, since the modified-byte distribution is almost balanced in these benchmarks and our proposed schemes are less effective for the balanced distribution.

4.4. IPC Speedup. The metric IPC speedup is used to evaluate the system performance of different schemes. IPC speedup is equal to the IPC of existing scheme divided by the IPC of baseline. The results of IPC speedup are shown in Figure 17. PCM-2R has the best performance in general and shows about 1.9X speedup compared with the baseline. PCM-R presents 1.8X speedup and silentstore scheme indicates 1.6X speedup compared with the DCW on average. In comparison, bit-mapping + SPCM shows 1.7X speedup compared with baseline on average. Our proposed schemes achieve IPC improvement due to the reduction of memory write service time. Since we adopt O3 (out-of-order) CPUs, more instructions are sent and finished and the IPC improves subsequently. Similar to the results of write latency, two memory-intensive workloads show significant speedup (more than 3.5X, as shown in workloads bwaves, leslie3d, and libquantum). It is demonstrated that the

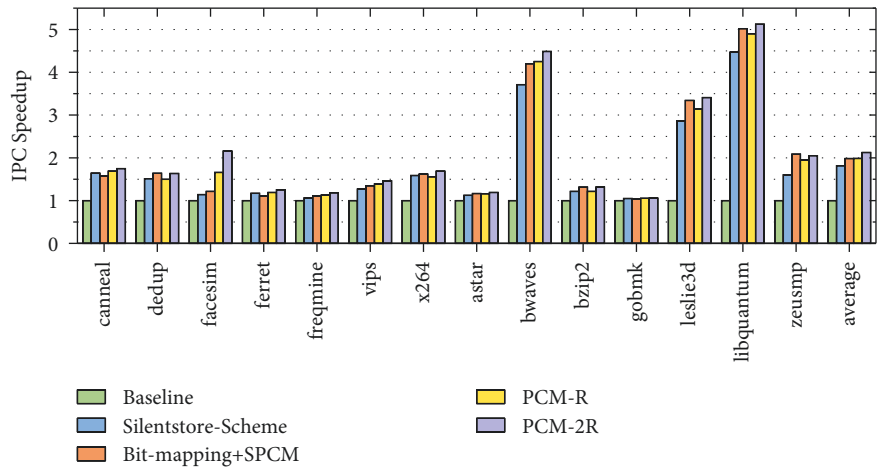

Figure 17: IPC speedup.

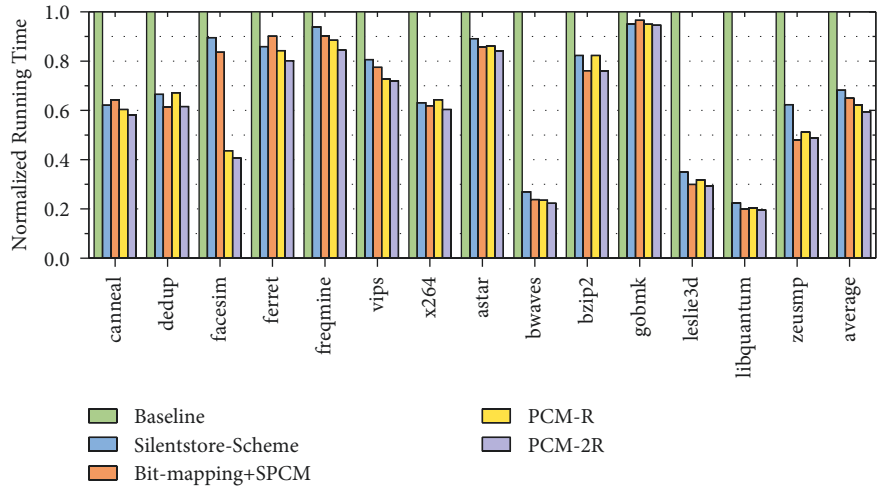

Figure 18: Running time.

reduction of read and write latency can lead to significant increase in system performance and our designs are effective.

4.5. Program Running Time. The program running time is also an important indicator of the performance of the main memory system. The faster the load/store instructions finish, the better problem running time can get. Our designs can get program running time reduction since both reshaping and remapping methods accelerate the write operation and the service time is reduced, as shown in Figure 18. Silentstore scheme and PCM-R schemes show 32% and 38% running time reduction compared with the baseline scheme, respectively. In comparison, bit-mapping + SPCM and PCM-2R show 35% and 41% running time reduction compared with the DCW scheme, respectively. PCM-2R outperforms silentstore scheme, bit-mapping + SPCM, and PCM-R by 9%, 6%, and 3%, respectively.

4.6. Memory Throughput. Write throughput is one of the most important metrics of the PCM-based main memory. Figure 19 shows the results of throughput normalized to the baseline. Our mechanisms show significant throughput improvement in all workloads. On average, the throughput improvement under PCM-R scheme is 2.0X. Four workloads show more than 2.0X improvement (canneal, bwaves, libquantum, and zeusmp). When employing the PCM-2R scheme, the throughput is increased significantly. The average improvement is 2.2X and 5 out of 14 workloads show more than 2.1X improvement over the baseline (canneal, ferret, bwaves, libquantum, and zeusmp) since their modified-byte distribution is very close to the given three patterns in this work. In comparison, silentstore and bit-mapping + SPCM schemes can get 1.8X and 1.9X throughput enhancement on average, respectively.

4.7. Energy Consumption. Although our mechanisms do not reduce the data amount directly, the memory energy consumption is reduced since data reshaping and remapping significantly decrease the number of write operations and reduce the overall write service time. Figure 20 illustrates the energy consumption of compared schemes normalized to

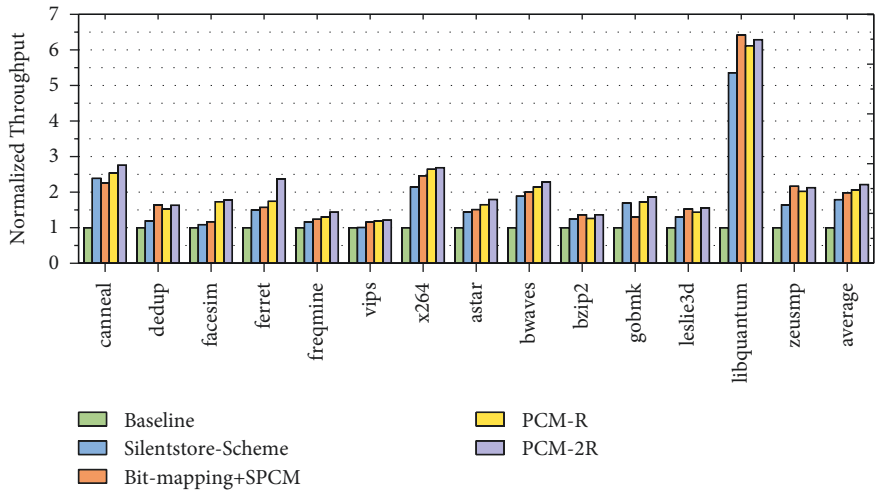

Figure 19: Throughput enhancement.

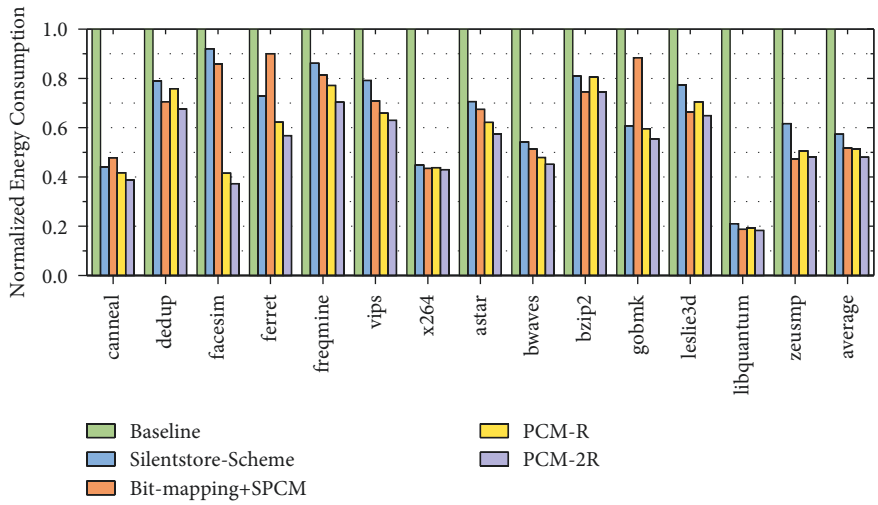

Figure 20: Energy consumption.

the baseline. PCM-R reduces energy consumption by up to 49% for all 14 workloads on average. PCM-2R decreases the energy consumption by 52% on average. The compared silentstore and bit-mapping + SPCM schemes can get 43% and 48% energy consumption on average compared with the baseline, respectively.

4.8. Sensitivity Analysis. DRAM buffer is widely used in PCM-based main memory system to hide the long write latency and a suitable DRAM cache size can effectively improve the performance of PCM main memory system. In this section, we try to explore the design space of proposed PCM-2R schemes with various DRAM cache and evaluate the impact of different LLC DRAM size on the effectiveness and efficiency of PCM-2R. Figure 21(a) shows the running time reduction (compared with the baseline) sensitivity to varying LLC DRAM size (32 MB, 64 MB, and up to 128 MB). On average, the running time reduction decreases from 37% to 25% with the LLC size increases. For some workloads, the degradation is quite obvious, such as canneal, facesim, freqmine, vips, x264, gobmk, leslie3d, and zeusmp. This is because larger LLC DRAM cache absorbs most of the main memory access requests; the service time of different schemes is almost the same in DRAM cache. For some workloads, the effect of different DRAM size is small, such as bzip2 and libquantum. Moreover, dedup shows performance improvement with DRAM size increases. This is because the memory accesses patterns are strengthened and have more expected patterns shown in Table 2. The experimental results of IPC speedup sensitivity to different DRAM cache size are similar to running time reduction, as shown in Figure 21(b). The IPC speedup results are also compared with the baseline. On average, the IPC speedup is 1.9X, 1.8X, and 1.6X when DRAM is 32 MB, 64 MB, and 128 MB, respectively. In general, IPC speedup decreases in most workloads expect dedup. For freqmine, astar, and gobmk, there are almost no memory accesses when DRAM is set to 128 MB and IPC speedup is 1X compared with the baseline, i.e., no improvement. In summary, our proposed schemes are still effective and efficient with varying LLC DRAM size.

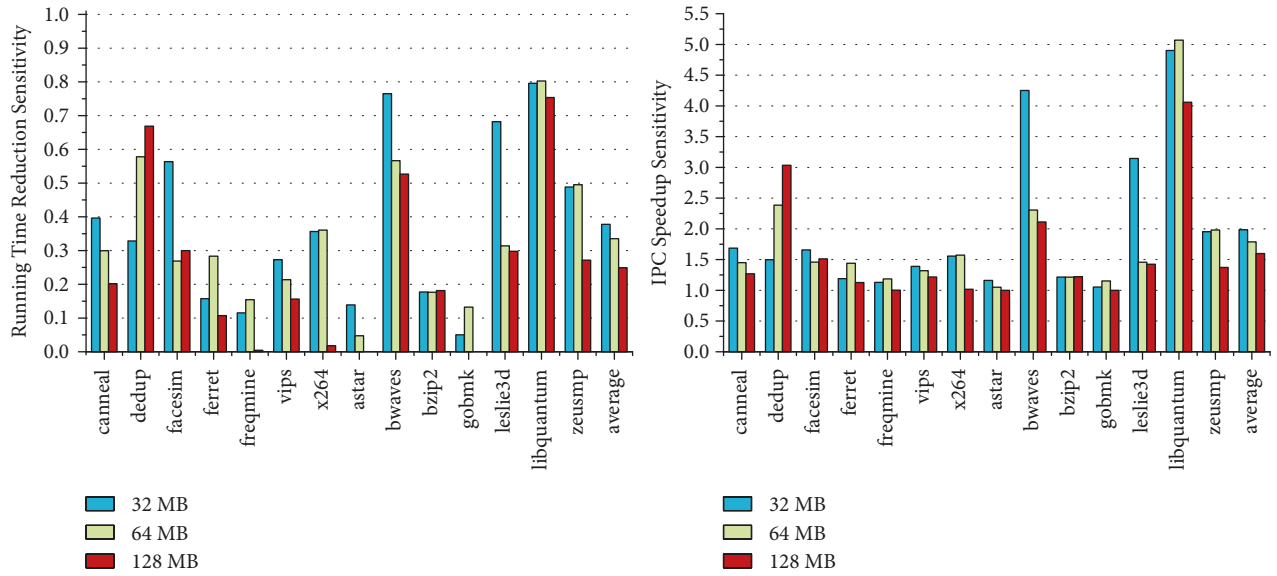

Figure 21: Sensitivity analysis: (a) running time reduction sensitivity; (b) IPC speedup sensitivity.

5. Related Work

In this section, we focus on the recent research related to this work. Du et al. [14] observed that the bit distribution is quite unbalanced among PCM groups and proposed a low-overhead D-XOR bit-mapping strategy to balance the distribution of modified bits. The write service time is reduced due to the balanced workload distribution. Jiang et al. [46] proposed a mechanism to decouple the 2-bit STT-RAM cells and improve the performance by leveraging the cell asymmetries. The cache lines are divided into RFWS lines (read fast and write slow) and RSWF lines (read low and write fast). Using efficiency prediction method, the data are placed into the right position and the write time is reduced. Hoseinzadeh et al. [19, 20] tried to improve the critical read latency by leveraging the asymmetries between the MSB and LSB bits of MLC cells. By decoupling the cells, the memory is divided into two parts. One part has almost the same read latency compared with SLC cell and the other remains the same as the MLC cell. The authors also proposed a pairing write queue to address the write problem raised by the line stripping method. Similar to this mechanism, Yoon et al. [28] proposed data mapping and buffering techniques to apply the key ideas to MLC PCM including cells decoupling and data prediction. The difference is that this strategy takes advantage of the row buffer space and divides the row buffer in two parts for RFWS regions and RSWF regions. Yue and Zhu [27] present a different decoupling method compared with prior art. This strategy divides one cache line into two half lines rather than dividing the whole memory into two different-type lines. Based on the key insights that the read operation usually involved the data of the first half of a cache line, the proposed bit-mapping scheme reduces the critical read latency. Zhang et al. [35] proposed a PCM write scheme named TriState-SET to improve the poor PCM write performance by choosing the fastest state transition among all possible transitions. Palangappa and Mohanram [47, 48] proposed compression expansion coding by combining the data compression algorithm and the energy and latency asymmetries in MLC write. Qureshi et al. [6] proposed write cancellation method by cancelling the currently executing write operation and then performing the critical read request first to improve the overall system performance. The authors also modeled the MLC PCM write and proposed an extended model for iterative write. Based on the proposed modeling, write pausing tries to pause the ongoing write request and reperform it after all read requests are issued. Jiang et al. [49] proposed a write truncation method to improve the write operations in MLC PCM. MLC PCM write involves multiple write iterations, which results in long write time. Finding that there is a long tail effect in the write iterations, write truncation method stops the iterations after reaching a certain number of times. In order to solve the problem of data errors, ECC methods are introduced. Jiang et al. [39] presented FPB method to improve the write parallelism of MLC PCM. The authors assumed and proved that the required power is reduced with the number of iterations increases. By implementing a fine-grained power budgeting method, two writes can be served in parallel under constraints and the service time is reduced. Huang and Hua [50] proposed a write-friendly and fast-recovery scheme for security metadata in nonvolatile memories by instantly persisting the modifications of security metadata without extra memory writes. Zhang et al. [51, 52] proposed two microarchitectural designs to enhance the write operations of nonvolatile memories. These methods can also be applied in conjunction with our techniques.

6. Conclusion

In this paper, we proposed PCM-2R, a novel MLC PCM write scheme based on the key observations that the modified-byte distribution is quite unbalanced among all chips and shows some unique patterns. Keeping these special

patterns in mind, we try to boost the write by reshaping the data to distribute the cache line block among all chips to avoid unbalanced distribution and then remapping modified bytes to fast region after decoupling MLC PCM cells considering the asymmetries in state transitions. Evaluation results of read latency, write latency, IPC improvement, and running time reduction of PARSEC 2.0 and SPEC 2006 workloads demonstrate the effectiveness and efficiency of PCM-2R. Our results show that PCM-2R reduces 51% read latency, 37% write latency, 41% running time, and 52% energy and gets 1.9X IPC and 2.2X throughput improvement on average compared with the baseline scheme. The experimental results also show that PCM-2R outperforms the state-of-the-art bit-mapping and SPCM schemes by 4.7% and 3.6% more read latency and write latency, 0.2X more IPC improvement, 2.8% more running time reduction, 0.2X more throughput improvement, and 3.5% more energy reduction.

Actually, there is an increasing demand for large capacity memory to run modern data-intensive applications, such as IOT, artificial intelligence algorithms, big data analytics, and so on. The proposed PCM-2R design can offer storage-class memory; therefore it is well suited for these use-case scenarios. As PCM has been partly commercially used, e.g., Intel 3D X-Point, we believe PCM-2R can be used in these real world use-case scenarios in near future.

Data Availability

The datasets used and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by Major Project of Scientific and Technological Innovation in Hubei (Grant nos. 2018ABA076, 2019AAA057, and 2020BGC028) and Key R&D and Promotion Projects of Science and Technology in Henan Province (no. 192102210112). This work was also supported by TC210804V-1 Project in Guangdong Province.

References

[1] L. Wilson, *International Technology Roadmap for Semiconductors (ITRS)*, Semiconductor Industry Association, Washington, DC, USA, 2013.

[2] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," *ACM SIGARCH - Computer Architecture News*, vol. 37, no. 3, pp. 24–33, 2009.

[3] E. Kültürsay, M. Kandemir, S. Anand, and O. Mutlu, "Evaluating STT-RAM as an energy-ecient main memory alternative," in *Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 256–267, IEEE, Austin TX, USA, April 2013.

[4] M. K. Qureshi, M. M Franceschini, L. A. Lastras-montaño, and J. P. Karidis, "Morphable memory system: a robust architecture for exploiting multi-level phase change memories," in *Proceedings of the 37th International Symposium on Computer Architecture (ISCA 2010)*, pp. 153–162, Citeseer, Saint-Malo, France, J2010.

[5] M. Zhang, L. Zhang, L. Jiang, Z. Liu, and F. T. Chong, "Balancing performance and Lifetime of MLC PCM by Using a Region Retention Monitor," in *Proceedings of the 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 385–396, IEEE, Austin, TX, USA, February 2017.

[6] M. K. Qureshi, M. M Franceschini, and L. A. Lastras-Monta, "Improving read performance of phase change memories via write cancellation and write pausing," in *Proceedings of the HPCA - 16 2010 The 16th International Symposium on High-Performance Computer Architecture*, pp. 1–11, IEEE, Bangalore India, January 2010.

[7] F. Huang, D. Feng, X. Wen et al., "Security RBSG: Protecting phase change memory with security-level adjustable dynamic mapping," in *Proceedings of the 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1081–1090, IEEE, Chicago IL USA, May 2016.

[8] Z. Wen, D. Feng, H. Yu, J. Liu, F. Huang, and P. Zuo, "Increasing lifetime and security of phase-change memory with endurance variation," in *Proceedings of the 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 861–868, IEEE, Wuhan, China, December 2016.

[9] Z. Wen, D. Feng, H. Yu, J. Liu, F. Huang, and Y. Chen, "An efficient parallel scheduling scheme on multi-partition PCM architecture," in *Proceedings of the ISLPED '16: Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pp. 344–349, ACM, Boston, MA, USA, August 2022.

[10] N. H. Seong, S. Yeo, H. Hsin, and S. Lee, "Tri-level-cell phase change memory: Toward an efficient and reliable memory system," in *Proceedings of the ISCA '13: Proceedings of the 40th Annual International Symposium on Computer Architectures*, no. 3, pp. 440–451, ACM, Philadelphia, PA, USA, July 2022.

[11] P. Zhou, Bo Zhao, J. Yang, and Y. Zhang, "Throughput enhancement for phase change memories," *IEEE Transactions on Computers*, vol. 63, no. 8, pp. 2080–2093, 2014.

[12] Z. Li, F. Wang, D. Feng, Y. Hua, J. Liu, and W. Tong, "MaxPB: Accelerating PCM write by maximizing the power budget utilization," *ACM Transactions on Architecture and Code Optimization*, vol. 13, no. 1, pp. 26–46, 2016.

[13] J. Yue and Y. Zhu, "Accelerating write by exploiting PCM asymmetries," in *Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 282–293, IEEE, Shenzhen, China, February 2013.

[14] Y. Du, M. Zhou, B. R. Childers, D. Mossé, and R. Melhem, "Bit mapping for balanced PCM cell programming," in *Proceedings of the SIGARCH Computer Architecture News*, pp. 428–439, Philadelphia, PA, USA, July 2022.

[15] C.-H. Lai, S.-C. Yu, C.-L. Yang, and H.-P. Li, "Fine-grained write scheduling for pcm performance improvement under write power budget," in *Proceedings of the 2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 19–24, IEEE, Rome, Italy, July 2015.

[16] Li Zheng, F. Wang, H. Yu et al., "Exploiting more parallelism from write operations on PCM," in *Proceedings of the 2016*

Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 768–773, IEEE, Dresden, Germany, March 2016.

[17] Li Zheng, F. Wang, D. Feng et al., "Tetris Write: Exploring more write parallelism considering PCM asymmetries," in *Proceedings of the 2016 45th International Conference on Parallel Processing (ICPP)*, pp. 159–168, IEEE, Philadelphia, PA, USA, August 2016.

[18] M. Arjomand, M. T. Kandemir, A. Sivasubramaniam, and C. R. Das, "Boosting access parallelism to PCM-based main memory," in *Proceedings of the ACM SIGARCH Computer Architecture News*, pp. 695–706, IEEE, Seoul, Korea (South), June 2016.

[19] M. Hoseinzadeh, M. Arjomand, and H. Sarbazi-Azad, "Reducing access latency of MLC PCMs through line striping," in *Proceedings of the ACM SIGARCH Computer Architecture News*, pp. 277–288, IEEE, Minneapolis, MN, USA, June 2014.

[20] M. Hoseinzadeh, M. Arjomand, and H. Sarbazi-Azad, "SPCM: The striped phase change memory," *ACM Transactions on Architecture and Code Optimization*, vol. 12, no. 1, pp. 38–25, 2015.

[21] B. C. Lee, P. Zhou, J. Yang et al., "Phase-change technology and the future of main memory," *IEEE micro*, vol. 30, p. 143, 2010.

[22] S. Raoux, G. W. Burr, M. J. Breitwisch et al., "Phase-change random access memory: A scalable technology," *IBM Journal of Research and Development*, vol. 52, no. 4, pp. 465–479, 2008.

[23] M. K. Qureshi, M. M Franceschini, A. Jagmohan, and L. A. Lastras, "PreSET: improving performance of phase change memories by exploiting asymmetry in write times," in *Proceedings of the 2012 39th Annual International Symposium on Computer Architecture (ISCA) IEEE, Portland, OR, USA*, June 2012.

[24] T. Nirschl, J. B. Philipp, T. D. Happ et al., "Write strategies for 2 and 4-bit multi-level phase-change memory," in *Proceedings of the 2007 IEEE International Electron Devices Meeting*, pp. 461–464, IEEE, Washington, DC, USA, 10-12 December 2007.

[25] M. Joshi, W. Zhang, and T. Li, "Mercury: A fast and energy-ecient multi-level cell based phase change memory system," in *Proceedings of the 2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pp. 345–356, IEEE, San Antonio, TX, USA, 12-16 February 2011.

[26] M. Zhao, L. Jiang, Y. Zhang, and C. J. Xue, "SLC-enabled wear leveling for MLC PCM considering process variation," in *Proceedings of the 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, San Francisco, CA, USA, 01-05 June 2014.

[27] J. Yue and Y. Zhu, "Reducing read latency in MLC PCM," in *Proceedings of the 2016 IEEE International Conference on Networking, Architecture and Storage (NAS)*, pp. 1–6, IEEE, Long Beach, CA, USA, August 2016.

[28] H. B. Yoon, N. Muralimanohar, J. Meza, O. Mutlu, and N. P. Jouppi, "Data mapping for higher performance and energy eciency in multi-level phase change memory," in *Proceedings of the Proc.NVMW'12*, pp. 1-2, Amherst, MA, USA, October 2012.

[29] M. Arjomand, M. T. Kandemir, A. Sivasubramaniam, C. R. Das, and C. Das, "Boosting access parallelism to PCM-based main memory," in *Proceedings of the ACM SIGARCH Computer Architecture News*, pp. 695–706, Seoul, Korea (South), June 2016.

[30] L. Liu, P. Chi, S. Li, Y. Cheng, and X. Yuan, "Building energy-ecient multi-level cell STT-RAM caches with data compression," in *Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 751–756, IEEE, Chiba Japan, January 2017.

[31] B.-Do Yang, J.-E. Lee, J.-Su Kim, J. Cho, S.-Y. Lee, and B.-G. Yu, "A low power phase-change random access memory using a data-comparison write scheme," in *Proceedings of the 2007 IEEE International Symposium on Circuits and Systems*, pp. 3014–3017, IEEE, New Orleans, LA, USA, 27-30 May 2007.

[32] X. April, "Zynq-7000 All Programmable SoC DS191 (v1.18)," 2017, https://www.xilinx.com/support/documentation/data_sheets/ds191-XC7Z030-XC7Z045-data-sheet.pdf.

[33] N. Binkert, B. Beckmann, G. Black et al., "The gem5 simulator," *ACM SIGARCH - Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[34] M. Poremba, T. Zhang, and Y. Xie, "NVMain 2.0: A user-friendly memory simulator to model (non-)volatile memory systems," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 140–143, 2015.

[35] X. Zhang, Y. Zhang, and J. Yang, "TriState-SET: Proactive SET for improved performance of MLC phase change memories," in *Proceedings of the 2015 33rd IEEE International Conference on Computer Design (ICCD)*, pp. 659–665, IEEE, New York, NY, USA, 18-21 October 2015.

[36] J. Meza, J. Chang, H. B. Yoon, O. Mutlu, and P. Ranganathan, "Enabling Efficient and Scalable Hybrid Memories Using Fine-Granularity DRAM Cache Management," *IEEE Computer Architecture Letters*, vol. 11, no. 2, pp. 61–64, 2012.

[37] H. B. Yoon, J. Meza, R. Ausavarungnirun, R. A. Harding, and O. Mutlu, "Row buffer locality aware caching policies for hybrid memories," in *Proceedings of the 2012 IEEE 30th International Conference on Computer Design (ICCD)*, pp. 337–344, IEEE, Montreal, QC, Canada, September 2012.

[38] J. Meza, Li Jing, and O. Mutlu, "Evaluating Row Buffer Locality in Future Non-volatile Main Memories," SAFARI Technical Report TR-SAFARI-2012-002, Carnegie Mellon University report , Pittsburgh, PA, USA, 2012.

[39] L. Jiang, Y. Zhang, B. R. Childers, and J. Yang, "FPB: Fine-grained power budgeting to improve write throughput of multi-level cell phase change memory," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 1–12, IEEE, Vancouver, BC, Canada, 01-05 December 2012.

[40] A. Hay, K. Strauss, T. Sherwood, G. H. Loh, and D. Burger, "Preventing PCM banks from seizing too much power," in *Proceedings of the Proc.Micro'11. ACM*, pp. 186–195, IEEE, Porto Alegre, Brazil, 03-07 December 2011.

[41] C. Bienia, *Benchmarking Modern Multiprocessors*, Ph.D. Dissertation, Princeton University thesis, Princeton, NJ, USA, 2011.

[42] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH - Computer Architecture News*, vol. 34, no. 4, pp. 1–17, 2006.

[43] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable DRAM alternative," in *Proceedings of the ISCA '09 36th annual international symposium on Computer architecture*, pp. 2–13, ACM, Philadelphia PA USA, July 11-14 2022.

[44] K. M. Lepak and M. H. Lipasti, "On the value locality of store instructions," in *Proceedings of the ISCA '00 27th annual international symposium on Computer architecture*, pp. 12–17, IEEE, Vancouver, BC, Canada, 4-14 June 2000.

[45] J. Zhao, O. Mutlu, and X. Yuan, "FIRM: Fair and high-performance memory control for persistent memory systems," in

Proceedings of the 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 153–165, IEEE, Cambridge, UK, 13-17 December 2014.

[46] L. Jiang, Bo Zhao, Y. Zhang, and J. Yang, "Constructing large and fast multi-level cell STT-MRAM based cache for embedded processors," in *Proceedings of the DAC Design Automation Conference 201*, pp. 907–912, IEEE, San Francisco, CA, USA, 03-07 June 2012.

[47] P. M. Palangappa and K. Mohanram, "CompEx: Compression-expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVM," in *Proceedings of the 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 90–101, IEEE, Barcelona, Spain, 12-16 March 2016.

[48] P. M. Palangappa and K. Mohanram, "CompEx++: Compression-expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVMs," *ACM Transactions on Architecture and Code Optimization*, vol. 14, pp. 1–30, 2017.

[49] L. Jiang, Bo Zhao, Y. Zhang, J. Yang, and B. R. Childers, "Improving write operations in MLC phase change memory," in *Proceedings of the IEEE International Symposium on High-Performance Comp Architecture*, pp. 1–10, IEEE, New Orleans, LA, USA, 25-29 February 2012.

[50] J. Huang and Y. Hua, "Write-Friendly and Fast-Recovery Scheme for Security Metadata in Non-Volatile Memories," in *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture*, pp. 359–370, IEEE, Seoul, Korea (South), February 2021.

[51] Y. Zhang, D. Feng, W. Tong, J. Liu, C. Wang, and J. Xu, "Tiered-ReRAM: A Low Latency and Energy Efficient TLC Crossbar ReRAM Architecture," in *Proceedings of the 35th Symposium on Mass Storage Systems and Technologies*, pp. 92–102, IEEE, Santa Clara, CA, USA, 20-24 May 2019.

[52] Y. Zhang, Z. Yu, L. Gu, C. Wang, and D. Feng, "EnTiered-ReRAM: An Enhanced Low Latency and Energy Efficient TLC Crossbar ReRAM Architecture," *IEEE Access*, vol. 9, pp. 167173–167189, 2021.