2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems

Making Write Less Blocking for Read Accesses in Phase Change Memory

Jianhui Yue, Yifeng Zhu

Electrical and Computer Engineering, University of Maine
{*jianhui.yue, yifeng.zhu*}*@maine.edu*

***Abstract*— Phase-change Memory (PCM) is a promising alternative or complement to DRAM for its non-volatility, scalable bit density, and fast read performance. Nevertheless, PCM has two serious challenges including extraordinarily slow write speed and less-than-desirable write endurance. While recent research has improved the write endurance significantly, slow write speed become a more prominent issue and prevents PCM from being widely used in real systems.**

To improve write speed, this paper proposes a new memory micro-architecture, called Parallel Chip PCM(PC2M), which leverages the spatial locality of memory accesses and trades bank-level parallelism for larger chip-level parallelism. We also present a micro-write scheme to reduce the blocking for read accesses caused by uninterrupted serialized writes. Micro-write breaks a large write into multiple smaller writes and timely schedules newly arriving reads immediately after a small write completes. Our design is orthogonal to many existing PCM write hiding techniques, and thus can be used to further optimize PCM performance. Based on simulation experiments of a multi-core processor under SPEC CPU 2006 multi-programmed workloads, our proposed techniques can reduce the memory latency of standard PCM by 68.5% and improve the system performance by 30.3% on average. PC2M and Micro-write significantly outperform existing approaches.

I. Introduction

Increasingly larger amounts of memory are required to alleviate I/O performance bottleneck. Unfortunately, DRAM technology is facing the transistor scaling limitation and it is a great challenge to fabricate high density of DRAM beyond 22*nm* [1]. In addition, DRAM technology also has a challenging thermal issue on most large servers since DRAM chips are energy-hungry and current green-computing technology achieves much less power saving on memory chips than processors. While the power dissipation of an idle processor can be lowered to a very small value via technologies such as dynamic voltage scaling, the background power of DRAM chips stills accounts for more than 40% of DRAM power [2]. Fortunately, phase-change memory (PCM) has a better scalability than DRAM and a PCM prototype is fabricated with a feature size as small as 3*nm* [3]. In addition, PCM consumes less leakage current and requires no refresh operations due to its non-volatile property. Compared with DRAM, PCM is much more energy efficient. As a result, PCM is emerging as a promising memory alternative and complementation to DRAM.

However, PCM has two major weaknesses: slow write performance and weak write endurance. A PCM cell endures around $10^8 - 10^{10}$ write cycles while a DRAM cell can support over 10^{15} writes. Recent research work has successfully [4] extended the PCM lifetime to over 7 years via fine-grained wear-leveling methods. The significant improvement in write endurance has made the other weakness, slow write, more prominent and urgent to solve.

Writes are slow in PCM because the crystallization process is time-consuming and the number of concurrent writes is limited due to circuit constraints. When writing data to a PCM storage cell, large electric current is drawn to heat up the cell's GST material in order to change its resistance. Different resistances represent different logic values stored in each cell. Compared with reading PCM, writing operation requires an electric current pulse with a much larger width and amplitude [3]. Furthermore, the noise at the power line restricts charge pump to provide a large electric current instantaneously for writing PCM cells, and thus the number of concurrent writes in a chip is limited to N bits, which is referred as xN write division mode [5]. The typical value for N can be 2, 4, 8 and 16 bits. This constrain limits the number of bytes that can be written to a bank each time, which is referred as write unite. The size of write unit is proportional to the size of write division mode. Accordingly, writing a cache line, typically 64 bytes, needs multiple serially executed write units. These serially executed write units of a cache line exacerbate the performance of PCM write. For example, writing a cache line of 64 bytes to a PCM bank is about 30 times slower than reading 64 bytes from the same bank [6]. Thus an on-going slow PCM write often unnecessarily blocks newly arriving read requests for a long time, significantly degrading the overall performance and preventing it from being widely deployed in real systems.

We propose a new memory micro-architecture, called Parallel Chip PCM (PC2M), to leverage the spatial locality of memory accesses, increase the speed of memory writes, and reduce the chance of a write blocking outstanding reads. We augment a PCM bank with extra chips to increase the size of write unit without violating power constrains at the chip level. Our novelty lies in that we leverage the fact that write requests tend to cluster to a small set of banks due to spatial access locality. Accordingly we trade bank-level parallelism for more chip-level parallelism to reduce the number of write units for each cache line. Compared with conventional memory architecture with the same number of PCM chips, our design takes less time to write a cache line, with the sacrifice that the number of concurrent writes to different banks is reduced.

1526-7539 2012

DOI 10.1109/MASCOTS.2012.39

To further mitigate the performance degradation of write blocking for read accesses in PCM, we propose another optimization technique, called micro-write, which allows the memory controller to schedule a waiting read request whenever a write unit completes, rather than waiting for the completion of writing a whole cache line. Typically in PCM a cache line is broken multiple write units and these unites are written into a bank in a serial order. Our micro-write technique converts a batch of uninterrupted serial writes into multiple interruptible micro-writes so that time-critical outstanding read requests are blocked for a shorter amount of time.

We evaluate our proposed techniques by simulating a multi-core system under the SPEC CPU 2006 benchmark suite. Experimental results under ten multi-programmed workloads show that PC2M and micro-write reduce the read latency of the standard PCM systems by 65.8% and 25.3% on average, and the total running time by 30.3% and 7.89%, respectively. We also compare our design with two state-of-the-art write-latency hidden technology, including Flip-N-Write and Write Cancellation. Compared with recently proposed Flip-N-Write, PC2M achieves on average 8.8% performance improvement. In addition, micro-write outperforms Write Cancellation by 7.41%.

The rest of paper is organized as follows. Section II introduces the PCM background and memory architecture. Section III presents our PC2M architecture design and Section IV presents our micro-write scheme. Section IV discusses the experimental results. Related work is summarized in Section VI and conclusions are given in Section VII.

II. Background

A. Phase Change Memory

Phase change memory exploits remarkably different properties of phase change material in a memory cell to store digital information. Phase change material, such Ge2Sb2Te5(GST), has two phases: an amorphous phase that has a high resistance in $M\Omega$s and a crystalline phase that has low resistance in $K\Omega$s. PCM can exploit the different resistances associated with the amorphous or crystalline phase to represent bit 0 and bit 1, respectively. Essentially, PCM is a one-transistor, one-resistor (1T1R) device shown in Figure 1, while DRAM is a one transistor, one-capacitor (1T1C) device.

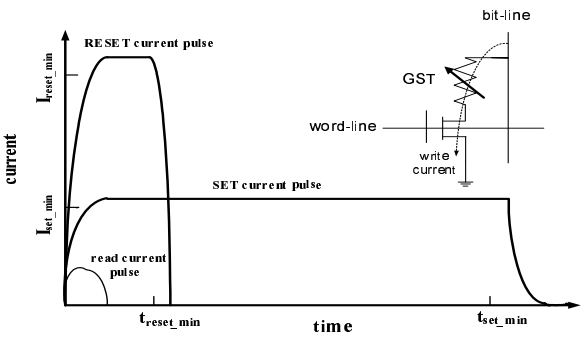

Fig. 1. Phase Change Memory

Write latency of PCM is much larger than its read latency. Reading a PCM cell is to sense the current flowing from the bit line, which is determined by the resistance of the cell. However, writing a PCM cell is more complex than reading, since GST needs to be heated up to change its phase, as shown in Figure 1. If bit 0 is written, a large current is applied to the cell for a short duration in order to heat GST abruptly and leave it in the amorphous phase. On the other hand, if bit 1 is written, a relatively smaller current is applied to the cell for a longer duration. After such a slow heat, GST remains in the crystalline phase. Since bit 0 and 1 are non-deterministically distributed among memory cells, the memory controller chooses the slow time of writing 1 as the basic time required to write a bit.

B. PCM Division Write Modes

The write performance is also limited by the electronic circuits constraints inside a PCM chip. As discussed previously, writing 0 needs a large amount of electric current to heat GST. Upon a write, the bit line is raised to a voltage higher than the phase change voltage, typically ranging from Vdd+1 to Vdd+3. With higher programming voltage and current, writing PCM consumes much power. The Dickson charge pump, widely used inside the PCM chip, provides the current to write driver. The noises at the power line restricts charge pump to provide the large amount of instantaneous current for writing PCM cells [5] inside a chip. In addition, the poor efficiency of charge pump further limits parallel writings [7]. This limitation of current provision constraints the number of concurrent written bits to 2, 4, and 8 typically in a chip [5]. This writing scheme is refereed to write division mode [7], resulting in increased time to write large data. For example, writing 16 bits to a PCM chip takes 8, 4 and 2 unit time when writing under x2, x4 and x8 write division mode, respectively.

C. Memory Architecture

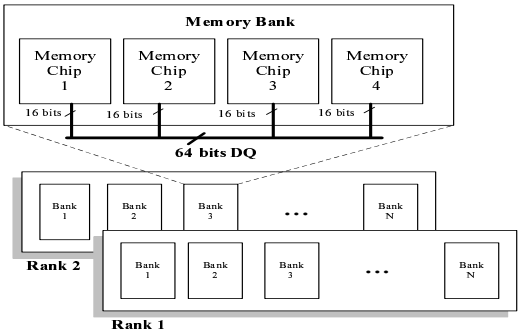

Fig. 2. Memory Architecture

Memory is often organized in a hierarchical way, as shown in Figure 2. A memory system usually consists of multiple memory ranks and each rank includes several banks. A bank is an independent device to serve data accesses. A bank is composed by a set of memory chips which include the array of storage cells. Each chip can provide multiple I/O bits

and multiple chips are connected to concurrently feed the data bus to provide data to the processor. The width of data bus determines the number of chips required to construct a memory bank.

III. Parallel Chip PCM (PC2M)

Due to the circuit constraints within PCM chips, the number of bits that can be written concurrently to a PCM bank is often limited and is referred to write unit size [8]. If the data bus has 64 bits, the write unit size is typically 1 byte, 2 bytes, 4 bytes and 8 bytes respectively in the x2, x4, x8 and x16 division write node [5], [8]. Thus writing a cache line of 64 bytes needs to perform 64, 32, 16, and 8 writes in the x2, x4, x8 and x16 division write node, respectively. Small size of write unit is one of the key reasons for slow write speed in PCM.

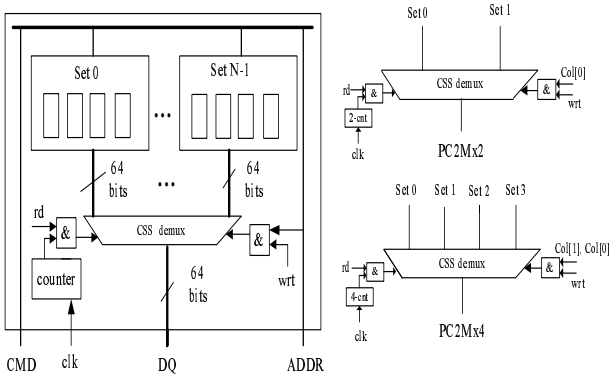

Fig. 3. Parallel Chip PCM Bank

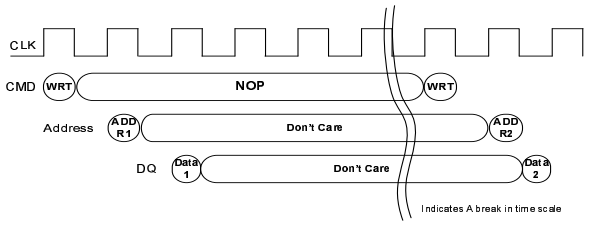

Fig. 4. Time Sequence of Conventional PCM Writing

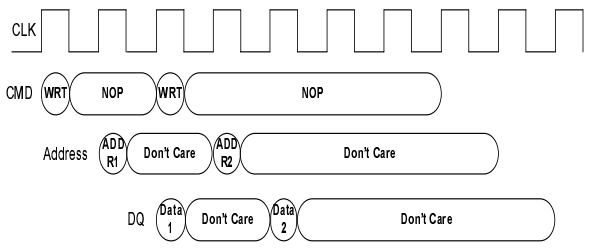

Fig. 5. Time Sequence of PC2M Writing

In conventional bank architecture, the number of chips is determined by the number of I/O bits of this chip and the width of the data bus. For example, if the data bus has a width of 64 bits, a bank needs to concurrently feed 64 bits data to the data bus and thus consists of four x16 PCM chips. Each chip can write 16 bits simultaneously under x16 write division mode each time and hence a bank can write 4×16 bits, i.e. 8 bytes, concurrently. Since the slow write speed, the next write unit waits long time, 430*ns* in our experiments, until the previous write unit is finished, as shown in Figure 4.

In order to increase the write unit size, we propose the Parallel Chips PCM (PC2M) that re-architects PCM banks, as shown in Figure 3. We augment the PCM bank with extra chips to a PCM bank to increase the number of bits that can be written to this bank in parallel. For example, the size of write unit of the bank with 8 chips is increased to 16 bytes. Accordingly, the increased write unit size reduces the number of write units when writing a cache line to PCM and hence shortens the time to complete writing a cache line to PCM. Each chip in the PC2M bank still comply the circuit constraints.

A set of chips inside PC2M are organized into a chip set and the width of a set of chips is the same as the width of the data bus. Chips in a set are concurrently accessed to serve requests from/to data bus. In order to simplify the design, the total number of chips added to PC2M is a multiple of the chip number within a chip set and thus they can provide sufficient amount of bits to feed the data bus. The PC2M bank is referred to as PC2Mxn if it has n chip sets.Data are interleaved among different chip sets with the granularity of column and thus neighbouring columns of data are stored at different chip sets which are determined by least significant bits of column address. In order to avoid modification of bank interface to data bus, we add a chip set selector (CSS) to access different chip sets. The CSS includes read part and write part and they are used to select chip sets for read and write operation respectively. Upon a memory write, the memory controller sends several pieces of data to different chip sets within a very short time as shown in Figure 5. The data size is limited by the number of parallel write bits for a chip set. Assume a chip set has the same number of chips as a conventional bank and a PC2M bank has 2 chip sets, then the size of write unit for this PC2M bank is 2 times of the conventional bank. In this way, we increase the size of a write unit of a PC2M bank. These data have different column addresses, which differ only on least significant bits and these different address bits are used to control the selection of chip set for writing. For example, PC2Mx2 and PC2Mx4 use the col[0] and col[0..1] respectively, which are least significant collum address bit, to select the chip set for different data, as shown in Figure 3.

We also add a circuit to manage reading data from various chip sets in PC2M. Firstly, we need to change memory prefetch length, which is a parameter stored in a configuration register in the memory device. Secondly, we add a clock counter to control the switching of data bus between different chip sets in a PC2M bank. The edge signal of the clock triggers the clock counter and activates the corresponding chip set

according to its value. This counter is based on module-N, where N is the total number of chip sets in a bank, as shown in Figure 3. Assume each bank has two chip sets. On rising edge, the counter value is 0 and correspondingly the chip set 0 is selected; on the falling edge, the counter value is 1 and thus the chip set 1 is selected. Note that each time the memory controller sends the same amount of data as the write unit of a chip set. Our design overhead is very small since only a counter and a switch is added for each bank.

Figure 4 and Figure 5 compares the difference of write operations between PCM and PC2Mx2. In this example, when writing two units, the second write has to wait for 430*ns* before issuing the second write command WRT, the second address A2. However, in PC2M, while the first write is being served in one chip set, the second write command WRT can be issued immediately after the target address of the first write has been successfully decoded. As a result, these two writes can be simultaneously serve by two different chip sets.

In our design, we make a tradeoff between the bank level parallelism (BLP) and chip level parallelism (CLP). Specifically, we trade BLP for a larger CLP. The high density of PCM usually offers much more banks than DRAM. For example, the product grade PCM prototype [6] has 16 banks while DRAM typically has only 8 or 4 banks. However, due to spatial access locality, most workloads studied in a multi-core environment have a BLP less than 7, as presented in the result section later. So PC2M aims to reduce the opportunity of chips being idle by reorganizing them to improve CLP to accelerate PCM write. When the number of chips in a bank increases, the number of banks is reduced proportionally. As a result, our design does not need more chips than conventional memory organization.

IV. Micro-Write

In DRAM, memory arrays are operated in a burst mode to improve the utilization of data bus. In the burst mode, memory accesses for a cache line continuously come or go to memory device without interruption. Combined with the fact that the latency of read and writes has no noticeable difference in DRAM, it is natural that the memory controller schedules the next request after all data bursts for a request are finished.

However, in PCM the burst mode is suitable for reads but ill-suitable for writes. The write burst is subject to chip's constrains and only a write unit is written at a time. Thus a cache line is broken into several write units. However, the conventional PCM controller still follows DRAM's scheduling policy and only schedules a waiting read request only after all bytes of a cache line are written to PCM, i.e., the conventional scheme schedules an outstanding read request only after all write units of a cache line are written. These serialized write units of a cache line make all newly arrived read requests waiting for a large amount of time.

We propose micro-write to reduce the long latency caused by uninterrupted serialized write units. Micro-write breaks a large write, typically a cache line, into a number of small writes, which is a micro-write in this paper. A micro-write is a basic schedule unit when serving a write request. When a micro-write completes, the memory controller can immediately schedule a waiting read request, rather than waiting all write units for a cache line are finished. Essentially, micro-write converts a batch of uninterrupted serialized write units into multiple interruptible micro-writes in order to block outstanding read requests less. Only after all micro writes of a write request complete does memory controller release its data entry in the request queue. If a read request hits the request queue in the memory controller, the memory controller retrieves data directly from the corresponding data entry in the queue, which avoid creating data consistency issues. The memory controller performs bookmarking to record partially completed write requests so that these writes can continue executing the rest micro-writes when PCM is idle. Note that the micro-write only changes the scheduling granularity for write, not for read.

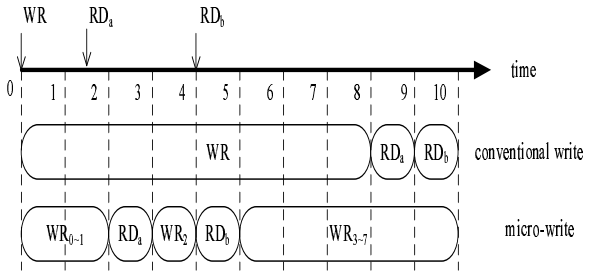

Fig. 6. Comparison of Conventional PCM Write and Micro-Write

Figure 6 uses a simple example to illustrate the difference between the conventional PCM write and our micro-write. In the conventional write, the slow write WR blocks the timing-critical read request RD_a and RD_b for 6.5 time units and 5 time units, respectively. In the micro-write, the memory controller will stop processing the next write unit immediately after a write unit completes. In this example, a cache line is broken into 8 write units, identified by the subscript for a write request. In our micro-write scheme, the memory controller schedules RD_a at time unite 2, which only blocks RD_a for 0.5 time unit. After RD_a finishes, WR continues to execute the next write unit. Again, when a write unit finishes, the read request RD_b arrives and then is executed immediately after the second write units completes. While the average waiting time of read requests in the conventional scheme is 5.75 time units, the average of waiting for read requests in our micro-write approach is 0.25 time unit.

The micro-write shares the same sprite with write pausing [9] which pauses PCM writing and schedules a waiting read request in order to reduce the read latency. However, our micro-write differs from write pause in two major aspects. Firstly, they operate at different time granularity. Write pause pauses between different write-verify iterations, while the micro-write pauses between different serial write units. Secondly, while write pausing highly relies on PCM to add a new interface through which memory controller has knowledge of programming and verification iteration and has ability to pause

the on-going iteration, our micro-write does not require any modification of PCM chip. As a result, micro-write is much easier to implement in real systems.

V. Experimental Methodology

We evaluate the performance by using the execution-driven processor simulator M5 [10] and the cycle-level memory simulator DRAMsim [11]. Table I shows the parameters of simulated processor and product grade PCM [6]. In order to accurately model memory access, we use the out-order core because it produces more parallel and independent memory accesses than in-order core. After passing through cache, the memory accesses go to PCM. The simulated processor has four cores with 32 MB L3 cache. The PCM read latency and write latency for a cell are set as 57*ns* and 430*ns* respectively [6].

Taking PCM chip constrains into account, we model the PCM write unit to be 8 bytes. As a result, writing a cache line of 64 bytes to PCM needs $8 \times 430ns = 3440ns$ while reading a cache line is less than 100*ns* due to the chip-level prefetch. To tolerate slow PCM write, we add a large off-chip L3 DRAM cache. In addition, since the memory write back is not in the performance critical path, we use the Read Instruction and Fetch First (RIFF) scheduling algorithm to improve the performance [9].

The SPEC CPU 2006 benchmark suit is used to construct 10 multi-programmed workloads with intensive memory accesses. All test applications in each workload runs in parallel and each application is fast-forwarded 5 billion instructions and then simulated 250 million instructions. Table II summarizes memory Read Per Kilo Instructions (RPKI), memory Write Per Kilo Instructions(WPKI), Memory Level Parallelism (MLP) [12] and Bank Level Parallelism (BLP) [13] for each workload. RPKI and WPKI are indicators of memory access intensity of a workload and most of them are larger than 1. While our L3 cache has 32MB and is smaller than the one used in other studies [9], [14], the intensity of memory accesses measured in our workloads is very close to theirs and thus we believe a larger L3 cache is unnecessary.

Importantly, in Table II, we find that BLP for most workloads is less than 7, which is much smaller than the number of banks available in most conventional memory system. This observation motivates our PC2M design that exploits the otherwise idle PCM chips to accelerate the slow PCM writing by increasing the chip-level parallelism.

We compare our design with the baseline scheme that uses the same parameters as listed in Table I but does not have any PCM optimization. We also compare ours against two recently proposed PCM write optimization techniques including Write Cancellation (WC) [9] and Flip-N-Write (FNW) [8]. Through comparison with old data, Flip-N-Write reduces the number of actual written bits by less than half and doubles the size of write unit, speeding up write. Write cancellation aborts an on-going write for a newly-arriving read request targeted to the same bank if the write operation is not close to completion. When there are no read requests, the cancelled write requests are re-executed. The static threshold value for write cancellation is set to be 75%, which was reported as the optimal value under the SPEC 2006 workloads [9]. We also compare these algorithms against an ideal PCM whose write latency is very small (57*ns*, the same latency as read), which acts as an upper bound of the performance improvement for PCM writing.

Parameter	Value
System	4-core CMP, 4 GHz
Execution Core	Alpha-like out-order processor
L1 Cache	32KB I-cache, 32KB D-cache
L2 Cache	Latency 20*ns*, 2MB, 4-way, 64B cache line
L3 Cache	Latency 50*ns*, 32MB, 8-way, 64B cache line
Memory Controller	RIFF request scheduling algorithm, page level interleaving address mapping
Width of data bus	64 bits
Number of Ranks	2
Number of Banks	16
Number of Chips per Bank	4
Width of a PCM Chip	16 bits
Time to write a PCM cell	430*ns*
Time to read a PCM cell	53*ns*
PCM write unit size	8 bytes

TABLE I
Simulation Parameters

We do not compare ours with write pause [9] and write truncation [14] since they depend on hard-to-obtain information about PCM writing statistic behavior. On the other hand, our proposed techniques are orthogonal to the previous techniques.

A. PC2M

In the baseline system, each PCM bank has four x16 chips to match the 64-bit data bus. In order to eliminate the issue of a small number of bits written in parallel in PCM, we augment a bank with more PCM chips. We add N times of chips into a bank and thus increase the size of write unit by N times, which is denoted as PC2MxN. In our experiments, we choose N to be 2 and 4. The write unit size is 8 bytes in baseline and the write unit size is increased to 16 bytes and 32 bytes respectively in PC2Mx2 and PC2Mx4. In order to fairly compare PC2M against the baseline, the total number of chips in PC2M and the baseline is exactly the same. For example, if the baseline has 2 ranks, the total number of chips is $2 \times 16 \times 4 = 128$ for the baseline with 2 ranks. Correspondingly, PC2Mx2 also has two ranks, each rank has 8 banks, and each bank has $2 \times 8 = 16$ chips. Thus the total number of chips in PC2Mx2 is $2 \times 8 \times 8 = 128$.

1) PC2M Read Latency: Figure 7 presents PC2M's and the other schemes' read latency reduction compared with the baseline. In all ten workloads studied, PC2Mx4 can successfully reduce the read latency more than Write Cancellation (WC) and Flip-N-Write (FNW). On average, the read latency reduction of PC2Mx2 and PC2Mx4 is 45.1% and 65.8% smaller than the baseline respectively, while Write Cancellation and Flip-N-Write achieve only 12.3% and 50.8% respectively. Although

Benchmark	Description	RPKI	WPKI	MLP	BLP
MIX1	astar, astar, asta, rastar	52.3677	40.8805	51.8186	6.1961
MIX2	astar, bzip2, milc, leslie3d	2.8253	2.7557	32.7038	4.7073
MIX3	leslie3d, leslie3d, milc, milc	3.3761	2.1986	26.8651	4.6363
MIX4	leslie3d, leslie3d, soplex, soplex	4.3020	2.4331	47.2699	3.7872
MIX5	libquantum, libquantum, libquantum, libquantum	49.4452	49.5580	68.0637	4.0746
MIX6	milc, libquantum, lbm, GemsFDTD	3.1718	2.5785	40.8732	4.7492
MIX7	milc, astar, milc, astar	4.5038	3.1436	17.6526	6.3863
MIX8	milc, milc, milc, milc	2.6383	2.3019	24.0918	5.6910
MIX9	sjeng, sjeng, sjeng, sjeng	1.0531	0.8473	6.2170	4.9882
MIX10	soplex, soplex, soplex, soplex	2.5687	1.5518	42.4891	3.6368

TABLE II
WORKLOADS

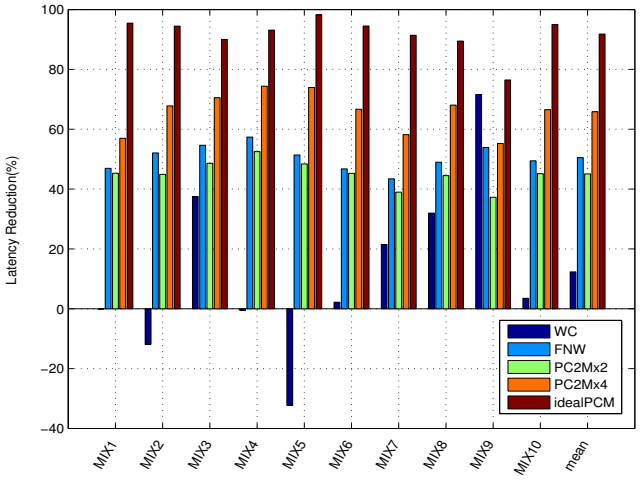

Fig. 7. PC2M Read Latency Reduction

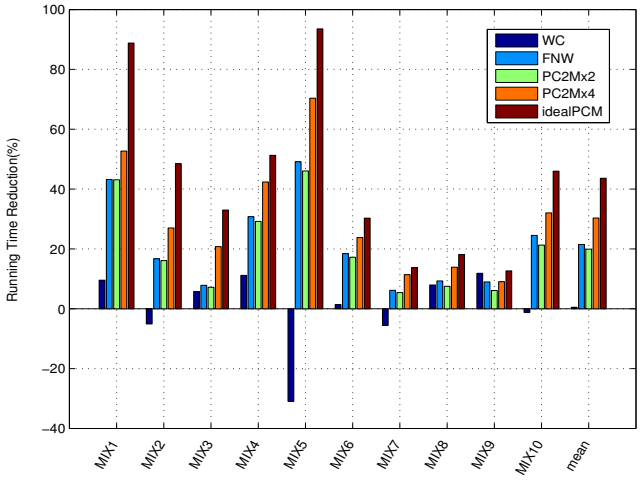

Fig. 8. PC2M Total Running Time Reduction

the number of bits that can be concurrently read is the same, PC2M allows more bits to be written simultaneously. As a result, PC2M reduces the number of write units for a cache line write, resulting in shorter waiting time for outstanding read requests.

It is interesting that PC2Mx4 achieves more read latency reduction than PC2Mx2. Compared with PC2Mx2, PC2Mx4 augments twice amount of PCM chips into a bank, doubles the write unit size to 32 bytes, and spends only half of the time to write a cache line. The experiment results show that the benefits of increased chip-level parallelism outweigh the sacrifice of bank-level parallelism.

PC2Mx2's read latency is close to Flip-N-Write but PC2Mx4's read latency is better than Flip-N-Write. With flipping bits, Flip-N-Write doubles size of write unit and achieves read latency reduction without changing memory organization. However, PC2Mx2 doubles the size of write unit but its number of banks is reduced by half. With a less number of banks in the memory system, we could potentially lose opportunities to concurrently serve more requests at different banks, which depends on how many concurrent requests go to different banks on average. This explains why PC2Mx2 is close to Flip-N-Write. On the other hand, Flip-N-Write cannot increase the write unit larger than 2 times of the baseline while PC2M can further increase the write unit size by adding more chips into a bank. Increasing the number of chips for a bank leads to less banks, which is potentially harmful to performance. But the average BLP for each workload is less than 7, as shown in Table II, and PC2Mx4 has 8 banks. So PC2Mx4 can meet the workload's BLP and thus enjoy the benefit of reduced write time of a cache line.

It is also noted that Write Cancellation provides the lowest read latency in the workload MIX9. This is because Write Cancellation works well if the BLP of a workload is close to its MLP. For example, as shown in Table II, BLP and MLP are 6.2 and 4.9 respectively in MIX9. In workloads where requests are evenly distributed over different banks, Write Cancellation can reduce the possibility of re-executing cancelled writes and hence effectively reduce read latency. We call these workloads Write Cancellation friendly workloads. However, Write Cancellation is less effective for unfriendly workloads with remarkably different BLP and MLP. In Write Cancellation, an on-going write still blocks read requests if it has finished more than $K\%$ of the time of writing a cache line, where $K\%$ is a predefined threshold. In our experiments, with a threshold of 75%, Write Cancellation can block a

read request for 860*ns* in the worst case. This possibility of blocking read increases with the more occurrences of re-execution of cancelled writes. As a result, for unfriendly workloads with very different BLP and MLP, most requests are clustered to a smaller number of banks, resulting in high frequent occurrences of write cancellations. This increases the possibility of read blocking.

2) PC2M Performance: Figure 8 shows PC2M's and other schemes' total running time reduction compared with the baseline. PC2M's reduced read latency is directly translated into performance gain. On average, PC2Mx2 and PC2MX4 provide 19.9% and 30.3% performance improvement over the baseline, respectively. Note that the performance of PC2Mx2 is close to Flip-N-Write due to its reduced number of banks but on average PC2Mx4 is 8.8% better than Flip-N-Write due to its larger write unit.

It is noted that under the MIX1 workload Write Cancellation reduces the total running time by 9.56% (see Figure 8) but has a larger read latency than the baseline (see Figure 7). This is because the re-execution of cancelled writes increases the possibility of blocking read request when an executing write finishes more than 75%. On the other hand, by cancelling write requests, Write Cancellation can improve BLP. For example, the BLP of MIX1 is 7.59 and 6.17 respectively in Write Cancellation and the baseline. The performance influenced by both read latency and BLP. So it is possible that the running time is reduced but read latency is increased.

B. Micro-write

Our micro-write scheme breaks a cache line write into a number of micro-writes and creates an opportunity for the memory controller to stop unfinished cache line writing and schedules a waiting read targeted to the same bank. However, micro-write introduces performance overhead caused by the loss of pipelined operations. In conventional system, the address transmission, address decoding, data transmission and data writing to cell for each write unit are executed in a pipeline way. Since a micro-write is the basic schedule unit, micro-write cannot perform these operations in a pipeline way across the boundary of each micro-write unit. Generally more micro-writes incur a larger overhead. We evaluate the micro-write schemes with the size of one write unite and two write units. Correspondingly each cache line write has 8 and 4 micro-writes, which are denoted as *microWritex8* and *microWritex4* respectively.

1) Micro-Write Latency: Figure 9 compares the read latency reduction over the baseline for four schemes, including Micro-Write, Write Cancellation (WC), Flip-N-Write (FNW), and the ideal PCM case.

Micro-Write achieves more reduction of read latency than Write Cancellation. On average, the latency reduction for micro-writex8 and micro-writex4 is 25.3% and 20.9% respectively, while Write Cancellation only has 12.3%. Compared with Micro-Write, Write Cancellation has a larger overhead associated with re-execution of cancelled writes. This is because Micro-Write chooses a micro write to be a basic write schedule unit and has a smaller opportunity to block a read request. However, micro-write achieves less latency reduction than Flip-N-Write under all workloads expect under the workload MIX9. Flip-N-Write reduces the time to write a cache line by half almost with small performance overhead while micro-write has some overhead caused by loss of pipelined operations across micro writes when writing a cache line.

We also note that microWritex8 performs worse than microWritex4 under some workloads in terms of read latency. A smaller size of micro-write implies that each micro-write occupies the bank for a shorter time and thus blocks waiting read less. However, microWritex8's overhead is two times of microWritex4. In the workloads in which more writes are clustered to a bank, such as MIX6, write requests are likely to be serially executed and the performance overhead of micro-write is accumulated more seriously. So its negative impact to read latency is more manifested under a larger number of micro-writes. Conversely, in the workloads where less writes are clustered to a bank, such as MIX3, the overhead of micro-write becomes smaller. The workload MIX6 and MIX3 have similar memory access intensity but different extents to which accesses are clustered to the same bank. This is a good example to show that when memory accesses are more concentrated to a small set of memory banks, the performance gain of micro-write tends to decrease with large number of micro writes while the performance gain of micro-write with larger number of micro writes increase when memory accesses are less skewed to a small set of banks.

2) MicroWrite Performance: The running time reduction over the baseline is shown in Figure 10. While Micro-Write outperforms Write Cancellation by 7.41% on average, it is worse than Flip-N-Write except under workload MIX9. BLP and MLP are almost the same in MIX9. Accordingly Micro-Write can schedule more waiting read requests to different banks. On the other hand, when write requests are concentrated to a small set of banks, i.e., when MLP is significantly larger than BLP, the overhead of micro-write is larger than Flip-N-Write and experiences poor performance. However, note that micro-write is orthogonal to Flip-N-Write and can be used together with Flip-N-Write to further improve the performance.

VI. Related Work

Due to the advantage of scalability, PCM has emerged as a promising non-volatility memory technology which can compensate and potential replace DRAM. Most existing research work focuses solve issues of its write endurance issue and slow writes in order to make it practical in real systems.

The write endurance issue has received extensive attentions recently. Ref. [15] presents removing data bit redundant, row shifting and segment swapping to prolong the PCM lifetime. Ref. [16] shows a start-gap wear leveling technique to improve the PCM endurance with negligible overhead. Ref. [17] proposes dynamically replicating memory writes data to different pages with disjoint failures and reading data from both pages in case of data corruption based on the observation that it

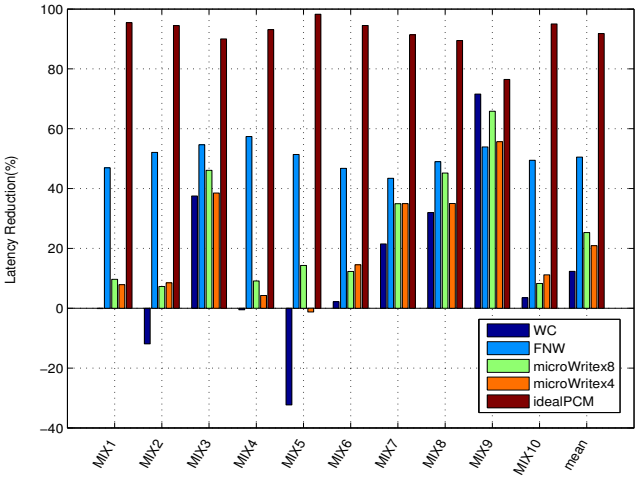

Fig. 9. MicroWrite Read Latency

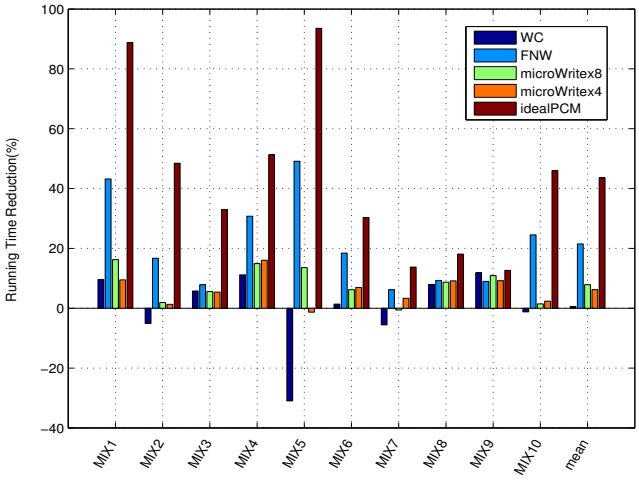

Fig. 10. MicroWrite Total Running Time

is easy to find two pages with the same failure distribution over storage space. Ref. [18] designs Error-Correcting Pointers (ECP) that permanently encodes the locations of failed bits into a table and replaces failed bits with healthy ones in order to correct corrupted bits. Ref. [19] proposes to partition memory into partitions with at most one failed bit and then use error correction codes to for each partition. Ref. [4] exploits the healthy bits in faulty block to store remapping information without any storage overhead in PCM and extend the PCM lifetime to over 7 years. These achievements have successfully made PCM more reliable to be used as main memory.

Several research projects have aimed to hide the long write latency of PCM. Ref. [20] adds a buffer to each PCM bank and exploits the data locality to mitigate the slow write. They further proposes a partial write strategy for a cache line to reduce the amount of data written to PCM.

Flip-N-Write [8] is a simple read-modify-write technique to write either flipped or unflipped to reduce write time. During a write, Flip-N-Write first reads the old value out of PCM and then calculate the number of different bits between the new data and the old data, as well as the number of different bits between the bit-flipped new data and old data, and finally choose to write either the flipped or unflipped new data depending on which has more unmodified data. This scheme requires an extra bit to record whether associated data have been flipped or not.

Write cancellation and write pausing [9] are proposed to indirectly improve the PCM read performance. Write cancellation aborts an on-going write for a newly-arriving read request targeted to the same bank if the write operation is not close to completion. When there are no read requests, the cancelled write requests are re-executed. Write pausing is a similar technique that pauses a PCM write at the end of a PCM write iteration and start to serve a waiting read request. Our micro-write differs from write pausing. Write pausing uses a much smaller time granularity to switch to reads and requires more bookmarking overhead. While write pausing stops a write between write-verify iterations, the micro-write pauses on different serial write units. In addition, while write pausing highly relies on PCM to add a new interface through which memory controller has knowledge of programming and verification iteration and has ability to pause the on-going iteration, our micro-write does not require any modification of PCM chip.Recently, write truncation and form switch [14] are proposed to improve write performance for the multiple-level-cell PCM. Based on the observation that not all bits for a block of data need the same number of write iterations, the write truncation early terminates the write iteration when most bits have been successfully written and then recover the data with extra error correction code during reading. The form switch compresses data to reduce the storage space overhead of write truncation.

VII. Conclusion

This paper presents and evaluates two new optimization techniques, including Parallel Chip PCM (PC2M) and micro-writes, to reduce the negative impact of slow writes on time-critical reads in PCM. PC2M re-architects a PCM bank with extra memory chips and increases the size of write unit to speed up writing a cache line to PCM. PC2M leverages the spatial locality of memory accesses and accordingly trades bank-level parallelism for increased chip-level parallelism. The other technique, micro-write, enables memory controller to early schedule the waiting read requests after a subset unit of a cache line is written to PCM rather than until completion of the whole cache line. Unlike other write hiding techniques, micro-write does not require to make any modification to PCM chips. Using SPEC CPU 2006 benchmark suit to evaluate a four-core system with PCM memory, our experiment results based on ten different multi-programmed workloads show that PC2M and micro-write can reduce the read latency respectively by 65.8% and 25.3% on average over a standard PCM baseline.

PC2M and micro-write reduce more read latency by 15% and 13% than Flip-N-Write and Write Cancellation respectively, two state-of-the-art PCM optimization techniques.

A. Acknowledgement

This work was supported by National Science Foundation under grants IIS 091663, CNS 1117032, EAR 1027809, CCF 0937988, CCF 0621493, and EPS 0904155.

REFERENCES

[1] International technology roadmap for semiconductors 2009.

[2] H. Zheng and Z. Zhu, "Power and performance trade-offs in contemporary dram system designs for multicore processors," *Computers, IEEE Transactions on*, vol. 59, no. 8, pp. 1033 –1046, Aug. 2010.

[3] S. Raoux, G. W. Burr, M. J. Breitwisch, C. T. Rettner, Y.-C. Chen, R. M. Shelby, M. Salinga, D. Krebs, S.-H. Chen, H.-L. Lung, and C. H. Lam, "Phase-change random access memory: A scalable technology," *IBM Journal of Research and Development*, vol. 52, no. 4.5, pp. 465 –479, July 2008.

[4] D. H. Yoon, N. Muralimanohar, J. Chang, P. Ranganathan, N. P. Jouppi, and M. Erez, "Free-p: Protecting non-volatile memory against both hard and soft errors," in *HPCA*, 2011, pp. 466–477.

[5] S. Kang and e. a. , "A 0.1-um 1.8-v 256-mb phase-change random access memory (pram) with 66-mhz synchronous burst-read operation," *Solid-State Circuits, IEEE Journal of*, vol. 42, no. 1, pp. 210 –218, Jan. 2007.

[6] K.-J. Lee and et. al., "A 90 nm 1.8 v 512 mb diode-switch pram with 266 mb/s read throughput," *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 1, pp. 150 –162, Jan. 2008.

[7] S. Hanzawa, N. Kitai, K. Osada, A. Kotabe, Y. Matsui, N. Matsuzaki, N. Takaura, M. Moniwa, and T. Kawahara, "A 512kb embedded phase change memory with 416kb/s write throughput at 100a cell write current," in *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, Feb. 2007, pp. 474 –616.

[8] S. Cho and H. Lee, "Flip-n-write: a simple deterministic technique to improve pram write performance, energy and endurance," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42, 2009, pp. 347–357.

[9] M. K. Qureshi, M. M. Franceschini, and L. A. Lastras-monta?o, "Improving read performance of phase change memories via write cancellation and write pausing," in *16th International Conference on High-Performance Computer Architecture (HPCA-16 2010), 9-14 January 2010, Bangalore, India*, ser. HPCA. IEEE Computer Society, 2010.

[10] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The m5 simulator: Modeling networked systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, 2006.

[11] D. Wang, B. Ganesh, N. Tuaycharoen, K. Baynes, A. Jaleel, and B. Jacob, "Dramsim: a memory system simulator," *SIGARCH Comput. Archit. News*, vol. 33, no. 4, pp. 100–107, 2005.

[12] A. Glew., "Mlp yes! ilp no! in wild and crazy ideas session, 8th international conference on architectural support for programming languages and operating systems," 1998.

[13] O. Mutlu and T. Moscibroda, "Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared dram systems," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ser. ISCA '08, 2008, pp. 63–74.

[14] L. Jiang, B. Zhao, Y. Zhang, J. Yang, and B. R. Childers, "Improving write operations in mlc phase change memory," in *HPCA*, 2012, pp. 201–210.

[15] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Proceedings of the 36th annual international symposium on Computer architecture*, ser. ISCA '09, 2009, pp. 14–23.

[16] M. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, Dec. 2009, pp. 14 –23.

[17] E. Ipek, J. Condit, E. B. Nightingale, D. Burger, and T. Moscibroda, "Dynamically replicated memory: building reliable systems from nanoscale resistive memories," in *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems*, ser. ASPLOS '10, 2010, pp. 3–14.

[18] S. Schechter, G. H. Loh, K. Straus, and D. Burger, "Use ecp, not ecc, for hard failures in resistive memories," in *Proceedings of the 37th annual international symposium on Computer architecture*, ser. ISCA '10, 2010, pp. 141–152.

[19] N. H. Seong, D. H. Woo, V. Srinivasan, J. A. Rivers, and H.-H. S. Lee, "Safer: Stuck-at-fault error recovery for memories," in *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '43, 2010, pp. 115–124.

[20] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *Proceedings of the 36th annual international symposium on Computer architecture*, ser. ISCA '09, 2009, pp. 2–13.