Scalable High Performance Main Memory System Using Phase-Change Memory Technology

Moinuddin K. Qureshi   Vijayalakshmi Srinivasan   Jude A. Rivers

IBM Research
T. J. Watson Research Center, Yorktown Heights NY 10598

{mkquresh, viji, jarivers}@us.ibm.com

ABSTRACT

The memory subsystem accounts for a significant cost and power budget of a computer system. Current DRAM-based main memory systems are starting to hit the power and cost limit. An alternative memory technology that uses resistance contrast in phase-change materials is being actively investigated in the circuits community. *Phase Change Memory (PCM)* devices offer more density relative to DRAM, and can help increase main memory capacity of future systems while remaining within the cost and power constraints.

In this paper, we analyze a PCM-based hybrid main memory system using an architecture level model of PCM. We explore the trade-offs for a main memory system consisting of PCM storage coupled with a small DRAM buffer. Such an architecture has the latency benefits of DRAM and the capacity benefits of PCM. Our evaluations for a baseline system of 16-cores with 8GB DRAM show that, on average, PCM can reduce page faults by 5X and provide a speedup of 3X. As PCM is projected to have limited write endurance, we also propose simple organizational and management solutions of the hybrid memory that reduces the write traffic to PCM, boosting its lifetime from 3 years to 9.7 years.

Categories and Subject Descriptors:
B.3.1 [Semiconductor Memories]: Phase Change Memory

General Terms: Design, Performance, Reliability.

Keywords: Phase Change Memory, Wear Leveling, Endurance, DRAM Caching.

1. INTRODUCTION

Current computer systems consist of several cores on a chip, and sometimes several chips in a system. As the number of cores in the system increases, the number of concurrently running applications (or threads) increases, which in turn increases the combined working set of the system. The memory system must be capable of supporting this growth in the total working set. For several decades, DRAM has been the building block of the main memories of computer systems. However, with the increasing size of the memory system, a significant portion of the total system power and the total system cost is spent in the memory system. For example, Lefurgy et al. [15] report that as much as 40% of the total system energy is consumed by the main memory subsystem in a mid-range IBM eServer machine. Therefore, technology researchers have been studying new memory technologies that can provide more memory capacity than DRAM while still being competitive in terms of performance, cost, and power.

Two promising technologies that fulfill these criteria are *Flash* and *Phase Change Memory*(PCM). Flash is a solid-state technology that stores data using memory cells made of floating-gate transistors. PCM stores data using a phase-change material that can be in one of two physical states: crystalline or amorphous. While both Flash and PCM are much slower than DRAM, they provide superior density relative to DRAM. Therefore, they can be used to provide a much higher capacity for the memory system than DRAM can within the same budget.

Figure 1 shows the typical access latency (in cycles, assuming a 4GHz machine) of different memory technologies, and their relative place in the overall memory hierarchy. Hard disk drive (HDD) latency is typically about four to five orders of magnitude higher than DRAM [6]. A technology denser than DRAM and access latency between DRAM and hard disk can bridge this speed gap. Flash-based disk caches have already been proposed to bridge the gap between DRAM and hard disk, and to reduce the power consumed in HDD [13]. However, with Flash being 2^8 times slower than DRAM, it is still important to increase DRAM capacity to reduce the accesses to the Flash-based disk cache. The access latency of PCM is much closer to DRAM, and coupled with its density advantage, PCM is an attractive technology to increase memory capacity while remaining within the system cost and power budget. Furthermore, PCM cells can sustain 1000x more writes than Flash cells, which makes the lifetime of PCM-based memory system in the range of years as opposed to days for a Flash-based main memory system.

There are several challenges to overcome before PCM can become a part of the main memory system. First, PCM being much slower than DRAM, makes a memory system comprising exclusively of PCM, to have much increased memory access latency; thereby, adversely impacting system performance. Second, PCM devices are likely to sustain significantly reduced number of writes compared to DRAM, therefore the write traffic to these devices must be reduced. Otherwise, the short lifetime may significantly limit the usefulness of PCM for commercial systems.

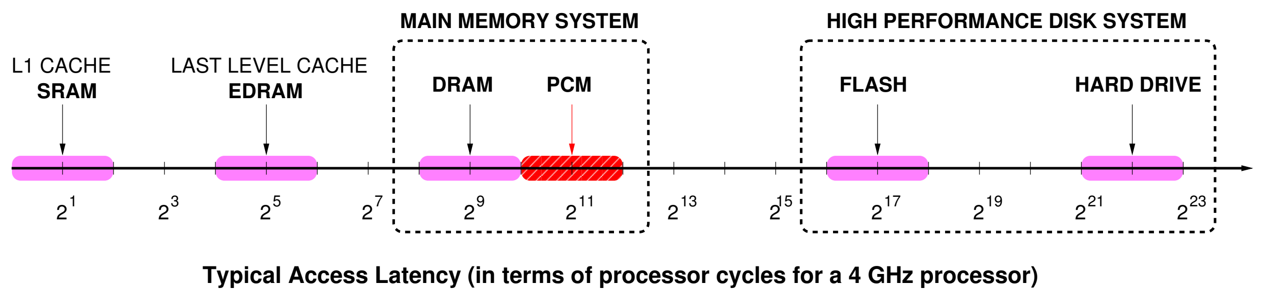

Figure 1: Latency of different technologies in memory hierarchy. Numbers accurate within a factor of two.

There is active research on PCM, and several PCM prototypes have been proposed, each optimizing for some important device characteristics (such as density, latency, bandwidth, or lifetime). While the PCM technology matures, and becomes ready to be used as a complement to DRAM, we believe that system architecture solutions can be explored to make these memories part of the main memory to improve system performance. The objective of this paper is to study the design trade-offs in integrating the most promising emerging memory technology, PCM, into the main memory system.

To be independent of the choice of a specific PCM prototype, we use an abstract memory model that is D times denser than DRAM and S times slower than DRAM. We show that for currently projected values of PCM ($S \approx 4, D \approx 4$), a main memory system using PCM can reduce page faults by 5X, and hence execute applications with much larger working sets. However, because PCM is slower than DRAM, main memory access time is likely to increase linearly with S, which increases the overall execution time. Therefore, we believe that PCM is unlikely to be a drop-in replacement for DRAM. We show that by having a small DRAM buffer in front of the PCM memory, we can make the effective access time and performance closer to a DRAM memory.

We study the design issues in such a hybrid memory architecture and show how a two-level memory system can be managed. Our evaluations for a baseline system of 16-cores with 8GB DRAM show that PCM-based hybrid memory can provide a speedup of 3X while incurring only 13% area overhead. The speedup is within 10% of an expensive DRAM only system which would incur 4X the area. We use an aggressive baseline that already has a large Flash-based disk cache. We show that PCM-based hybrid memory provides much higher performance benefits for a system without Flash or with limited Flash capacity.

As each cell in PCM can endure only a limited number of writes, we also discuss techniques to reduce write traffic to PCM. We develop an analytical model to study the impact of write traffic on the lifetime of PCM that shows how the "bytes per cycle" relates to average lifetime of PCM for a given endurance (maximum number of writes per cell). We show that architectural choices and simple enhancements can reduce the write traffic by 3X which can increase the average lifetime from 3 years to 9.7 years.

To our knowledge, this is the first study on architectural analysis of PCM based main memory systems. We believe this will serve as a starting point for system architects to address the challenges posed by PCM, making PCM attractive to be integrated in the main memory of future systems.

2. BACKGROUND AND MOTIVATION

With increasing number of processors in the computer system, the pressure on the memory system to satisfy the demand of all concurrently executing applications (threads) has increased as well. Furthermore, critical computing applications are becoming more data-centric than compute-centric [9]. One of the major challenges in the design of large-scale, high-performance computer systems is maintaining the performance growth rate of the system memory. Typically, the disk is five orders of magnitude slower than the rest of the system [6] making frequent misses in system main memory a major bottleneck to system performance. Furthermore, main memory consisting entirely of DRAM is already hitting the power and cost limits [15]. Exploiting emerging memory technologies, such as Phase-Change Memory (PCM) and Flash, become crucial to be able to build larger capacity memory systems in the future while remaining within the overall system cost and power budgets. In this section, we first present a brief description of the Phase-Change Memory technology, and highlight the strengths of PCM that makes it a promising candidate for main memory of high-performance servers. We present a simple model that is useful in describing such emerging memory technologies for use in computer architecture studies.

2.1 What is Phase-Change Memory?

PCM is a type of non-volatile memory that exploits the property of chalcogenide glass to switch between two states, amorphous and crystalline, with the application of heat using electrical pulses. The phase change material can be switched from one phase to another reliably, quickly, and a large number of times. The amorphous phase has low optical reflexivity and high electrical resistivity. Whereas, the crystalline phase (or phases) has high reflexivity and low resistance. The difference in resistance between the two states is typically about five orders of magnitude [24] and can be used to infer logical states of binary data.

While the principle of using phase change materials for memory cell was demonstrated in 1960s [23], the technology was too slow to be of practical use. However, the discovery of fast crystallizing material such as $Ge_2Sb_2Te_5$(GST) [27] and *Ag*- and *In-doped* Sb_2Te(AIST) [25] has renewed industrial interest in PCM and the first commercial PCM products are about to enter the market. Both GST and AIST can crystallize in less than 100ns compared to $10\mu s$ or more for earlier materials [23]. PCM devices with extremely small dimensions as low as $3nm \times 20nm$ have been fabricated and tested. A good discussion on scaling characteristics of PCM is available in [24].

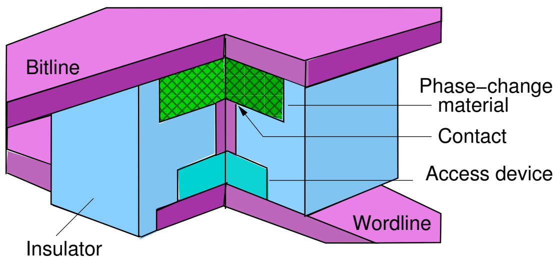

Figure 2: Typical Phase-Change Memory Device

Figure 2 shows the basic structure of a PCM device [24]. The PCM material is between a top and a bottom electrode with a heating element that extends from the bottom electrode, and establishes contact with the PCM material. When current is injected into the junction of the material and the heating element, it induces the phase change. Crystallizing the phase-change material by heating it above the crystallization temperature (but below the melting temperature) is called the SET operation. The SET operation is controlled by moderate power, and long duration of electrical pulses and this returns the cell to a low-resistance state, and logically stores a 1. Melt-quenching the material is called the RESET operation, and it makes the material amorphous. The RESET operation is controlled by high-power pulses which places the memory cell in high-resistance state. The data stored in the cell is retrieved by sensing the device resistance by applying very low power. The Current-Voltage curve for PCM is shown in Figure 3 [24].

One of the critical properties that enables PCM is *threshold switching* [24]. To convert from crystalline to amorphous state, very high voltage is required to deliver the high power. However, above a particular threshold voltage, the conductivity of the material in the amorphous state increases rapidly thereby generating large current flows, and consequently heats the material. If the current pulse is switched off as soon as the threshold voltage is reached, the material returns to the high-resistance, amorphous state.

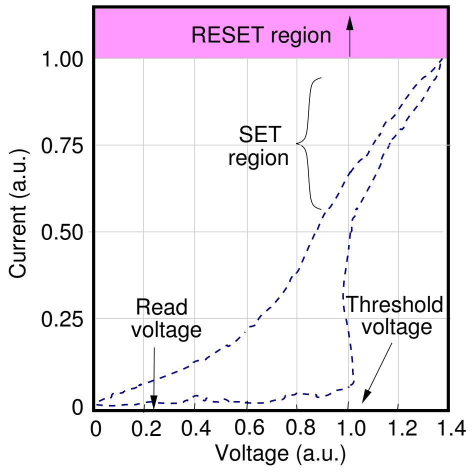

Figure 3: Typical I-V curve for PCM device, demonstrating threshold switching (a.u.= arbitrary units)

2.2 Why PCM for Main Memory?

PCM is a dense technology with feature size comparable to DRAM cells. Furthermore, a PCM cell can be in different degrees of partial crystallization thereby enabling more than one bit to be stored in each cell, Recently, a prototype [4] with two logical bits in each physical cell has been demonstrated. This means four states with different degrees of partial crystallization are possible, which allows twice as many bits to be stored in the same physical area. Research is under way to store more bits per PCM cell.

Table 1 summarizes the properties of different memory technologies based on the data obtained from the literature [11][12][4][22][10][5][26][1]. Write endurance is the maximum number of writes for each cell. Data retention is the duration for which the non-volatile technologies can retain data. As PCM is still being researched, different sources quote different relative merits of their PCM prototypes. Here we use the values of density, speed, endurance for PCM as a summary from the different prototype projections, and this will serve as a reference point for a PCM design.

Table 1: Comparison of Memory Technologies

Parameter	DRAM	NAND Flash	NOR Flash	PCM
Density	1X	4X	0.25X	2X-4X
Read Latency	60ns	25 us	300 ns	200-300 ns
Write Speed	≈1 Gbps	2.4 MB/s	0.5 MB/s	≈100 MB/s
Endurance	N/A	10^4	10^4	10^6 to 10^8
Retention	Refresh	10yrs	10yrs	10 yrs

From Table 1, it is obvious that the poor density of NOR Flash makes it uncompetitive compared to DRAM for designing main memory. NAND Flash has much better density than DRAM but has significantly slow random access time (≈ 25us). The 400x latency difference between DRAM and Flash means that DRAM capacity will have to continue increasing in the future main memories. Furthermore, the poor write endurance of Flash would result in unacceptably low lifetime (in days) for main memory system. However, Flash has much better latency and power than HDD, and is finding widespread use as a disk cache [13]. In our studies, we assume that the disk system already has a large Flash-based disk cache. However, we show that even with such an aggressive disk system, main memory capacity is still a problem.

Among the emerging memory technologies, PCM has the most promising characteristics. PCM offers a density advantage similar to NAND Flash, which means more main memory capacity for the same chip area. The read latency of PCM is similar to NOR Flash, which is only about 4X slower compared to DRAM. The write latency of PCM is about an order of magnitude slower than read latency. However, write latency is typically not in the critical path and can be tolerated using buffers. Finally, PCM is also expected to have higher write endurance (10^6 to 10^8 writes) relative to Flash (10^4 writes). We believe that these capabilities of PCM combined with lower memory cost makes PCM a promising candidate for main memory of high-performance servers.

2.3 Abstract Memory Model

The focus of this work is to study the effect of overall system performance by adding PCM as a complement to the DRAM memory. In order to not restrict our evaluation of PCM to currently available prototypes, we have adopted the approach of a generic memory model that combines existing DRAM memory with a PCM memory. Therefore, in order to study the impact of adding PCM to main memory we first develop a simple abstract model using DRAM as

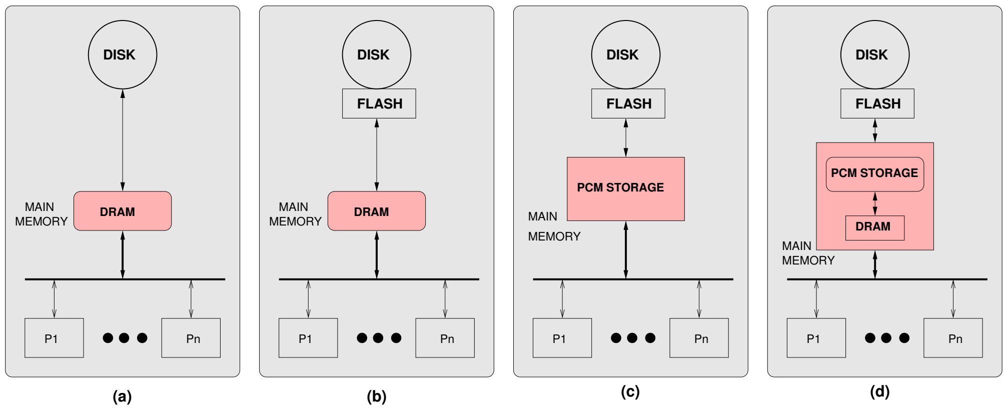

Figure 4: Candidate main memory organizations (a) Traditional system (b) Aggressive system with Flash-based disk cache (c) System with PCM (d) System with Hybrid memory system

a reference. The PCM technology is described as $\{D, S, Wmax\}$ which means the PCM is D times denser than DRAM, S times slower read latency than DRAM and can endure a maximum of $Wmax$ writes per cell. For the purposes of our evaluation, we use currently projected values[1] for PCM technology and describe it as $\{D = 4, S = 4, Wmax = 10Million\}$.

Using a PCM with $S = 4$ as a replacement of DRAM significantly increases the memory access latency, which can have an adverse impact on system performance. Therefore, we explore the trade-offs of using a combination of DRAM and PCM as part of the main memory. Furthermore, the write endurance of PCM, although orders of magnitude better than Flash, can still limit main memory lifetime. The next section describes the organization and system performance challenges in having such a hybrid main memory system comprising of different technologies.

3. HYBRID MAIN MEMORY

Figure 4 (a) shows a traditional system in which DRAM main memory is backed by a disk. Flash memory is finding widespread use to reduce the latency and power requirement of disks. In fact, some systems have only Flash-based storage without the hard disks; for example, the MacBook Air [3] laptop has DRAM backed by a 64GB Flash drive. It is therefore reasonable to expect future high-performance systems to have Flash-based disk caches [13] such as shown in Figure 4(b). However, because there is still two orders of magnitude difference in the access latency of DRAM memories and the next level of storage, a large amount of DRAM main memory is still needed to avoid going to the disks. PCM can be used instead of DRAM to increase main memory capacity as shown in Figure 4 (c). However, the relatively higher latency of PCM compared to DRAM will significantly decrease the system performance. Therefore, to get the best capacity and latency, Figure 4(d) shows the hybrid system we foresee emerging for future high-performance systems. The larger PCM storage will have the capacity to hold most of the pages needed during program execution, thereby reducing disk accesses due to paging. The fast DRAM memory will act as both a buffer for main memory, and as an interface between the PCM main memory and the processor system. We show that a relatively small DRAM buffer (3% size of the PCM storage) can bridge most of the latency gap between DRAM and PCM.

[1]PCM cells are smaller in size (4.8 F^2 per cell) compared to DRAM cells (6-8 F^2 per cell) [2]. Each PCM cell can store multiple bits [4]. PCM is expected to scale better than DRAM [9]. Therefore, we assume PCM has 4X density compared to DRAM.

3.1 Hybrid Main Memory Organization

In a hybrid main memory organization, the PCM storage is managed by the Operating System (OS) using a Page Table, in a manner similar to current DRAM main memory systems. The DRAM buffer is organized similar to a hardware cache that is not visible to the OS, and is managed by the DRAM controller. Although, the DRAM buffer can be organized at any granularity, we assume that both the DRAM buffer and the PCM storage are organized at a page granularity.

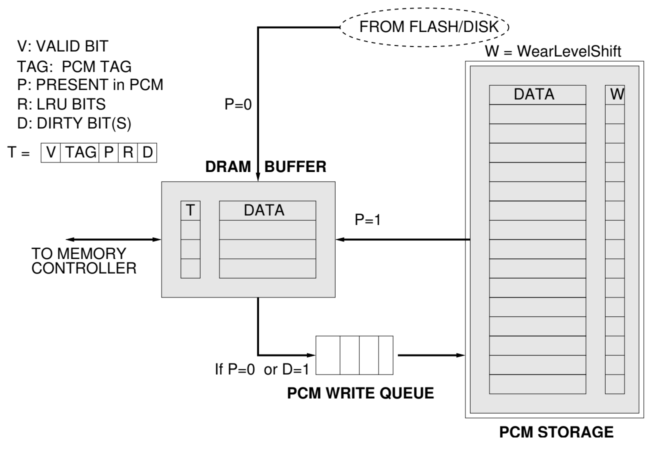

Figure 5: Lazy Write Organization

Figure 5 shows our proposed hybrid main memory organization. In addition to covering the larger read latency of PCM using a DRAM buffer, this organization tolerates an order of magnitude slower write latency of PCM using a write queue, and overcomes the endurance limit of PCM using techniques to limit the number of writes to the PCM memory. These mechanisms are discussed in detail in the next few sections, along with a description of the relevant portions of Figure 5.

3.2 Lazy-Write Organization

We propose the Lazy-Write organization which reduces the number of writes to the PCM and overcomes the slow write speed of the PCM, both without incurring any performance overhead. When a page fault is serviced, the page fetched from the hard disk (HDD) is written only to the DRAM cache. Although allocating a page table entry at the time of page fetch from HDD automatically allocates the space for this page in the PCM, the allocated PCM page is not written with the data brought from the HDD. This eliminates the overhead of writing the PCM. To track the pages present only in the DRAM, and not in the PCM, the DRAM tag directory is extended with a "presence" (P) bit. When the page from HDD is stored in the DRAM cache, the P bit in the DRAM tag directory is set to 0. In the "lazy write" organization, a page is written to the PCM only when it is evicted from the DRAM storage, and the P bit is 0, or the dirty bit is set. If on a DRAM miss, the page is fetched from the PCM then the P bit in the DRAM tag directory entry of that page is set to 1. When a page with P bit set is evicted from the DRAM, it is not written back to the PCM unless it is dirty. Furthermore, to account for the larger write latency of the PCM a write queue is associated with the PCM. We assume that tags of both the write queue and the DRAM buffer are made of SRAM in order to help in probing these structures while incurring low latency. Given the PCM write latency, a write queue of 100 pages is sufficient to avoid stalls due to write queue being full.

Compared to an architecture that installs in both storage (PCM and DRAM), the "lazy write" architecture avoids the first write in case of dirty pages. For example, consider the kernel of the DAXPY application $Y[i] = a \cdot X[i] + Y[i]$. In this case, "lazy write" policy fetches page(s) for array Y only to DRAM. The PCM gets a copy of the pages of Y only after it has been read, modified, and evicted from DRAM. If on the other hand page(s) for Y were installed in the PCM at fetch time, then they would have been written twice, once on fetch and again on writeback at the time of eviction from DRAM. The number of PCM writes for the read only page for X remains unchanged in both configurations.

3.3 Line-Level Writes

Typically, the main memory is read and written in pages. However, "endurance" limits of the PCM require exploring mechanisms to reduce the number of writes to the PCM. We propose writing to the PCM memory in smaller chunks instead of a whole page. For example, if writes to a page can be tracked at the granularity of a processor's cache line, the number of writes to the PCM page can be minimized by writing only "dirty" lines within a page. We propose *Line Level WriteBack (LLWB)*, that tracks the writes to pages held in the DRAM on the basis of processor's cache lines. To do so, the DRAM tag directory shown in Figure 5 is extended to hold a "dirty" bit for each cache line in the page. In this organization, when a dirty page is evicted from the DRAM, if the P bit is 1 (i.e., the page is already present in the PCM), only the dirty lines of the page are written to the PCM. When the P bit of a dirty page chosen for eviction is 0, all the lines of the page will have to be written to the PCM. LLWB significantly reduces wasteful writes from DRAM to PCM for workloads which write to very few lines in a dirty page. To support LLWB we need dirty bits per line of a page. For example, for the baseline system with 4096B page and 256B linesize, we need 16 dirty bits per page in the tag store of DRAM buffer.

3.4 Fine-Grained Wear-Leveling for PCM

Memories with limited endurance typically employ wear-leveling algorithms to extend their life expectancy. For example, in Flash memories, wear-leveling algorithms arrange data in a manner so that sector erasures are distributed more evenly across the Flash cell array and single sector failures due to high concentration of erase cycles are minimized. Wear leveling enables the host system to perform its reads and writes to logical sector addresses, while the wear leveling algorithm remaps logical sector addresses to different physical sector addresses in the Flash array depending on write traffic. For example, the wear-leveling algorithm of TrueFFS [16] file system tracks the least used sectors in the Flash to identify where to next write data.

We also assume that the baseline PCM system uses a wear-leveling mechanism to ensure uniform usage of different regions of the PCM memory. However, wear-leveling is typically done at a granularity larger than page-size (e.g. 128KB regions) in order to reduce wear-out tracking overhead [13].

LLWB reduces write traffic to PCM. However, if only some cache lines within a page are written to frequently, they will wear out sooner than the other lines in that page. We analyze the distribution of write traffic to each line in a PCM page. Figure 6 shows the total writeback traffic per dirty page for the two database applications, db1 and db2. The average number of writes per line is also shown. The page size is 4KB and line size is 256B, giving a total of 16 lines per page, numbered from 0 to 15.

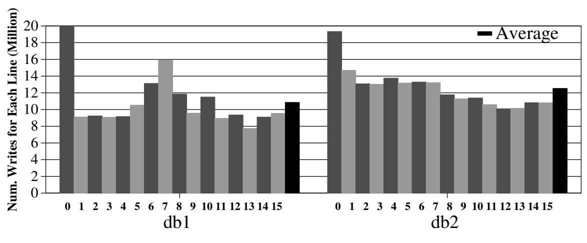

Figure 6: Total write traffic for each of the sixteen lines (0-15) for dirty pages in baseline system

For both db1 and db2 there is significant non-uniformity in which lines in the page are written back. For example, in db1, Line0 is written twice as often as average. This means Line0 may get endurance related failure in half the time. The lifetime of PCM can be increased if the writes can be made uniform across all lines in the page. This can be done by tracking number of writes on a per line basis, however, this would incur huge tracking overhead. We propose a technique, *Fine Grained Wear-Leveling (FGWL)*, for making the writes uniform (in the average case) while avoiding per line storage. In FGWL, the lines in each page are stored in the PCM in a rotated manner. For a system with 16 lines per page the rotate amount is between 0 and 15 lines. If the rotate value is 0, the page is stored in a traditional manner. If it is 1, then the Line 0 of the address space is stored in Line 1 of the physical PCM page, each line is stored shifted, and Line 15 of address space is stored in Line 0. When a PCM page is read, it is realigned. The pages are written from the Write Queue to the PCM in a line-shifted format. On a page fault, when the page is fetched from the hard disk, a Pseudo Random Number Generator (PRNG) is consulted to get a random 4-bit rotate value, and this value is stored in the WearLevelShift (W) field associated with the PCM page as shown in Figure 5. This value remains constant until the page is replaced, at which point the PRNG is consulted again for the new page allocated in the same physical space of the PCM.

A PCM page is occupied by different virtual pages at different times and is replaced often (several times an hour). Therefore, over the lifetime of the PCM page (in years) the random rotate value associated with each page will have a uniform distribution for the rotate value of 0 through 15. Therefore, the average stress on each physical page will be uniform. Figure 7 shows the write traffic per dirty DRAM page with FGWL, for db1 and db2. FGWL makes the write traffic uniform for all the lines in the PCM page, which means the lifetime of PCM page is determined by the average-case write traffic and not the worst-case write traffic to a line. Implementing FGWL requires 4-bit storage per page (4KB).

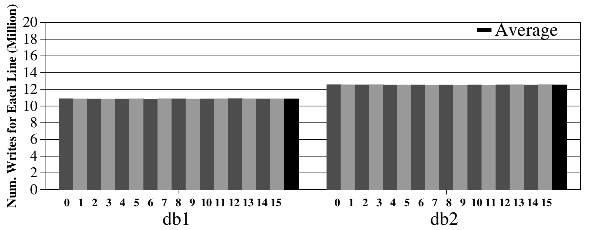

Figure 7: Total write traffic for the sixteen lines (0-15) for dirty pages using FGWL

3.5 Page Level Bypass for Write Filtering

Not all applications benefit from more memory capacity. For example, streaming applications typically access a large amount of data but have poor reuse. Such applications do not benefit from the capacity boost provided by PCM. In fact, storing pages of such applications only accelerates the endurance related wear-out of PCM. As PCM serves as the main memory, it is necessary to allocate space in PCM when a page table entry is allocated for a page. But, the actual writing of such pages in the PCM can be avoided by leveraging the lazy write architecture. We call this *Page Level Bypass (PLB)*. When a page is evicted from DRAM, PLB invalidates the Page Table Entry associated with the page, and does not install the page in PCM. We assume that the OS enables/disables PLB for each application using a configuration bit. If the PLB bit is turned on, all pages of that application bypass the PCM storage.

4. EXPERIMENTAL METHODOLOGY

4.1 Workloads

We use a diverse set of applications from various suites, including data-mining, numerical computation, Unix utilities, streaming applications, and database workloads. Six out of the eight benchmarks are data-parallel applications, each containing sixteen identical threads. These benchmarks are simulated to completion. Benchmarks *qsort* and *bsearch* (binary search) are Unix utilities. *Gauss* is a numerical algorithm used to solve linear equations. *Kmeans* is a data-mining algorithm used primarily to classify data into multiple clusters. In our studies, *kmeans* simulates 8 iterations of clustering and re-centering between two cluster centers. *Daxpy* and *vdotp* are streaming benchmarks derived from the stream suite [19]. These two benchmarks spend more time in array initialization than in computation. So we do not collect statistics in the initialization phase for these two benchmarks. We also use scaled versions of two industry-standard database workloads, derived from a mainframe sever. For both database workloads, db1 and db2, we simulate four billion memory references (corresponding to a total transfer of 1 Tera Byte of data between memory and processors). Table 2 shows the relevant characteristics of the benchmarks used in our studies, including data-set size, page faults per million instructions for the baseline 8GB memory, instructions/thread, and memory accesses per thousand instructions (MAPKI).

4.2 Configuration

We use an in-house system simulator for our studies. The baseline configuration is a sixteen-core system comprising of four 4-core CMPs with the parameters given in Table 3. We use a simple in-order processor model so that we can evaluate our proposal for several hundred billion instructions in order to stress the multi-GB main memory. Each core consists of private L1 and L2 caches, each with a linesize of 256B. We keep the processor parameters constant in our study.

Table 3: Baseline Configuration

System	Sixteen cores (Four quad-core chips)
Processor Core	single issue in-order, five stage pipeline, 4GHz
L1 caches (Private)	I-cache and D-cache : 64 KB, 4-way 256B line
L2 cache (Private)	2MB, 16-way, LRU, 256B linesize
DRAM Memory	8GB, 320 cycles access latency, 8 ranks of 8 banks each, bank conflict modeled
Off-chip Bus	16B wide split-transaction bus, 2:1 speed ratio
Disk Cache (Flash)	32μs (128K cycles) avg. latency, 99% hit rate
Hard disk drive	2ms (8M cycles) average access latency

The baseline consists of an 8GB DRAM main memory which can be accessed in 320 cycles if there are no bank conflicts. The off-chip bus can transfer a cache line in 32 cycles and bus queuing delays are modeled. A page size of 4KB is assumed. Virtual to physical address translation is performed using a page table built in our simulator. A clock style algorithm is used to perform page replacements. If there is PCM storage, it is accessible at 4X the latency of DRAM. In the hybrid memory configuration, the DRAM buffer is 16-way with 4KB lines (same as page size), and is managed using LRU replacement. The interconnect for PCM is assumed to be identical to that between DRAM and processor.

We also assume that our memory system has a very aggressive Flash-based disk cache (99% hit rate) so that the latency of page faults is reduced considerably. Although such an aggressive configuration will diminish the benefits of larger capacity PCM main memory, we expect such efficient Flash caches to be used in future high-performance systems. We discuss systems without a Flash cache in Section 5.4.

Table 2: Benchmark Characteristics (Bn=Billions).

Name	Description	Data set for each thread	$\frac{PageFaults}{Million\ inst}$	Inst/thread	MAPKI
qsort	Quick sort	14M entries, 128B each	21.5	114Bn	1.66
bsearch	Binary search	7M entries, 256B each, 14M queries	2507	12Bn	20.8
kmeans	Clustering, Data Mining	256K pts * 1K attrib of 8B each	48	175Bn	0.82
gauss	Gauss Siedal method	matrix 64K x 4K, 8B entry	97	97Bn	1.82
daxpy	Do $Y_i = a \cdot X_i + Y_i$	X, Y are 100M entries x 8B each	156	2.6Bn	3.75
vdotp	Vector dot product $X_i \cdot Y_i$	X, Y are 100M entries x 8B each	205	2Bn	3.29
db1	Database	OLTP	39.5	23Bn	10.86
db2	Database	web-based database	119.5	22.9Bn	10.93

5. RESULTS AND ANALYSIS

5.1 Page Faults vs. Size of Main Memory

Figure 8 shows the number of page faults as the size of the main memory is increased from 4GB to 32GB, normalized to the baseline with 8GB. The bar labeled *Gmean* denotes the geometric mean. Both database benchmarks, *db1* and *db2*, continue to benefit from increasing the size of main memory. For *bsearch*, there are about 100X more page faults at 8GB than at 32GB. *Gauss* and *Kmeans* frequently reuse a data-set of greater than 16GB. These benchmarks suffer a large number of page misses, unless the memory size is 32GB. *Daxpy* and *vdotp* do not reuse the page after initial accesses, therefore the number of page faults is independent of main memory size. Overall, a 4X increase (8GB to 32GB) in the memory capacity due to the density advantage of PCM reduces the average number of page faults by 5X.

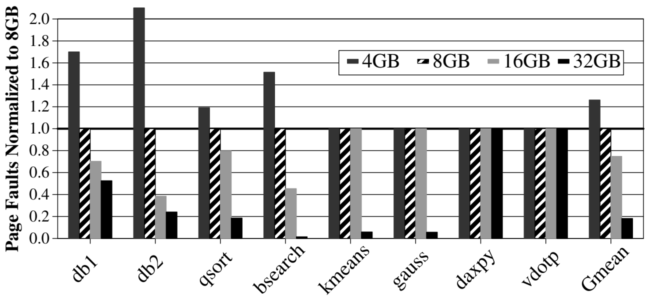

Figure 8: Number of page faults (normalized to 8GB) as size of main memory is increased.

5.2 Cycles Per Memory Access

PCM can provide 4X more memory capacity than DRAM but at 4X more latency. If the data-set resident in main memory is frequently reused then the page-fault savings of the PCM system may be offset by increased memory access penalty on each access. Figure 9 shows the average cycles per memory access for four systems: baseline 8GB DRAM, 32GB PCM, 32GB DRAM (an expensive alternative), and 32GB PCM with 1 GB DRAM. For *db2*, *qsort*, *bsearch*, *kmeans*, and *gauss*, the increased capacity of PCM reduces average memory access time because of fewer page faults. For *db1*, even though the increased capacity of PCM reduced page faults by 46%, the average memory access time increases by 59% because the working set is frequently reused. Both *daxpy* and *vdopt* do not benefit from more capacity but suffer because of increased memory latency. Having a 1GB DRAM buffer along-with PCM makes the average memory access time much closer to the expensive 32GB DRAM system.

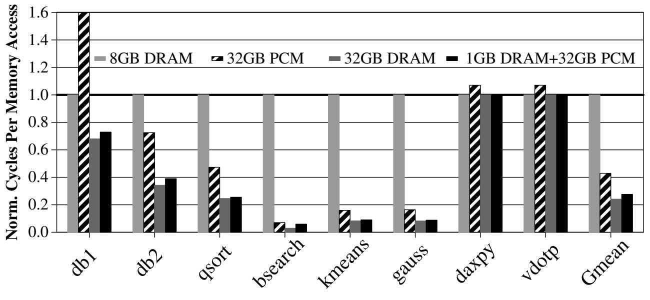

Figure 9: Average cycles per memory access (normalized to the baseline 8GB DRAM system).

5.3 Normalized Execution Time

Figure 10 shows the normalized execution time of the four systems discussed in the previous section. The reduction in average memory access time correlates well with the reduction in execution time. On average, relative to the baseline 8GB system, the PCM-only 32GB system reduces execution time by 53% (speedup of 2.12X) and the 32GB DRAM only system reduces it by 70% (speedup of 3.3X). Whereas, the hybrid configuration reduces it by 66.6% (speedup of 3X). For five out of the eight benchmarks, execution time reduces by more than 50% with the hybrid memory system. Thus, the hybrid configuration provides the performance benefit similar to increasing the memory capacity by 4X using DRAM, while incurring only about 13% area overhead while the DRAM-only system would require 4X the area.

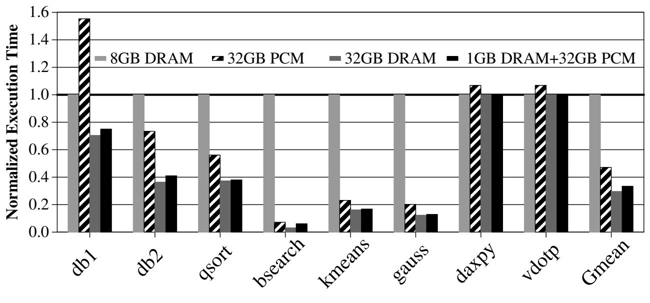

Figure 10: Execution time (normalized to 8GB DRAM).

5.4 Impact of Hard-Disk Optimizations

In our baseline we used an aggressive Flash-based disk cache with 99% hit rate. Figure 11 shows the speedup obtained with the hybrid memory (1GB DRAM + 32GB PCM) over 8GB DRAM for four systems: a Flash only disk system, a Flash-cache with 99% hit rate, 90% hit rate, and a traditional system with no Flash-based disk cache. As, the benchmark *bsearch* has very large speedups the geometric mean without *bsearch*, namely *GmeanNoB*, is also shown. For current systems with limited size (or without) Flash-based disk cache the speedups are much higher than reported with our aggressive baseline. For the Flash-only disk system, the hybrid memory system provides an average speedup of 2.5X (2X excluding *bsearch*). For a limited sized disk cache with 90% hit-rate, the hybrid memory system provides an average speedup of 4.46X (3.27X excluding *bsearch*). Whereas, for a traditional disk-only system, the hybrid memory provides an average speedup of 5.3X (3.7X excluding *bsearch*). Thus, regardless of the HDD optimizations, hybrid main memory provides significant speedups.

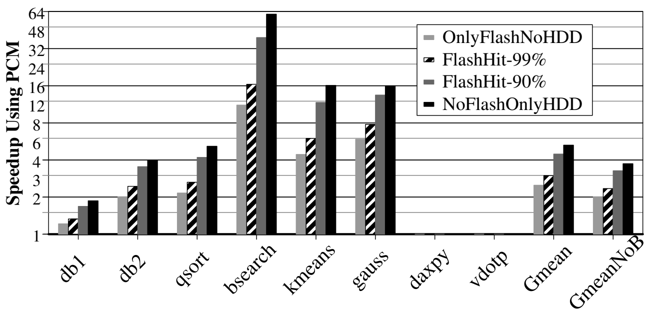

Figure 11: Speedup of Hybrid Memory over baseline as Flash cache is varied. Note: Y-axis is in log scale.

5.5 Impact of PCM Latency

We have assumed that PCM has 4X higher read latency than DRAM. In this section, we analyze the benefit of hybrid memory system as the read latency of PCM is varied. Figure 12 shows the average execution time of the eight workloads normalized to the 8GB DRAM system for PCM-only system and the hybrid memory system as PCM read latency is increased from 2X to 16X. For reference, the average execution time for the 32GB DRAM-only system is also shown. As the read latency of PCM is increased from 2X to 16X, the average execution time of the PCM-only system increases from 0.35 to 1.09. Thus, the higher read latency of PCM hurts performance even with 4X boost in capacity. The PCM-based hybrid system, however, is much more robust to PCM read latency. As the read latency of PCM is increased from 2X to 16X, the average execution time of the hybrid system increases from 0.33 to only 0.38.

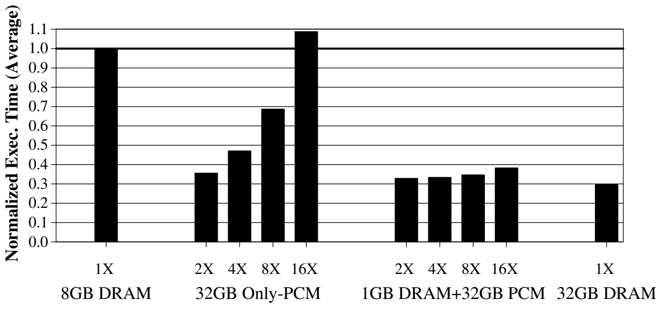

Figure 12: Normalized Execution Time of PCM-only and hybrid systems as PCM read latency is varied (Note: PCM latency is defined relative to DRAM latency (1X)).

5.6 Impact of DRAM-Buffer Size

The DRAM buffer reduces the effective memory access time of the PCM-based memory system relative to an equal capacity DRAM memory system. A large DRAM buffer reduces the latency difference between the two memory systems, but incurs more area overhead. Figure 13 shows the execution time of the hybrid system with 32GB PCM and DRAM buffer size varying from 0.25GB to 1GB to 2 GB to 8GB. Note that the execution time is normalized to the 32GB DRAM-only system and not the baseline. Except *bsearch*, all benchmarks have execution time with 1GB DRAM buffer very close to the 32GB DRAM system. Thus, a 1GB DRAM buffer provides a good trade-off between performance benefit and area overhead.

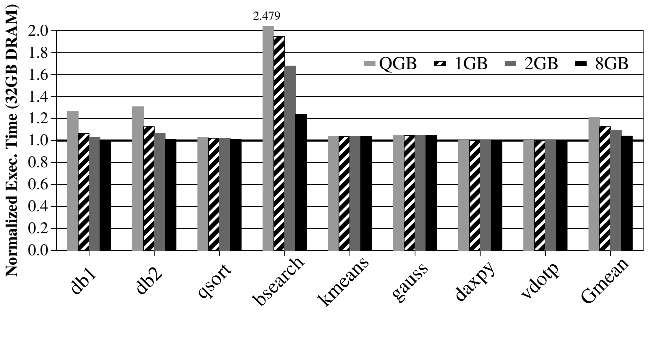

Figure 13: Execution time of hybrid memory (with 32GB PCM) when the size of the DRAM buffer is varied. All values are normalized to the 32GB DRAM system.

5.7 Storage Overhead of Hybrid Memory

Table 4 shows the storage overhead for the hybrid memory consisting of 32GB PCM and a 1GB DRAM. The significant additional area overhead is the DRAM buffer. In addition to data, each entry in the DRAM buffer requires 31 bits of tag-store (1 valid bit + 16 dirty bits + 9-bit tag + 4-bit LRU + P bit). We assume that the tags of the DRAM buffer are made of SRAM in order to have fast latency. Relative to the 8GB of DRAM main memory for the baseline, all the overhead is still less that 13% (We pesimistically assume SRAM cells incur 20 times the area of DRAM cells [2]).

Table 4: Storage Overhead wrt 8GB DRAM Memory

Structure	Storage overhead
1 GB DRAM Buffer Data	1GB DRAM
1GB DRAM Buffer Tag (4byte/page * 256K pages)	1MB SRAM
100-entry PCM write queue	400KB SRAM
Wear-leveling counters (4bit/page * 8M pages)	4MB DRAM
Total storage overhead (%)	1.005 GB (12.9%)

5.8 Power and Energy Implications

Figure 14 shows the three key metrics: power, energy, and energy-delay product for the memory system of the baseline (8GB DRAM), hybrid memory (1GB DRAM + 32GB DRAM) and 32GB DRAM-only memory. All values are reported normalized to the baseline and are computed as geometric mean over the 8 benchmarks. For the DRAM storage we assumed DDR3 type memories and used the Micron System Power Calculator [21]. For the PCM system we used values of idle-power, read-power and write-power of PCM relative to DRAM, based on the PCM prototypes [14].

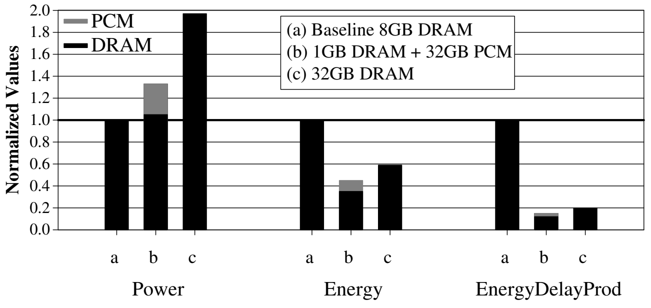

Figure 14: Power, Energy, and Energy Delay Product of memory systems normalized to the Baseline.

The hybrid memory system consumes 33% more power than the baseline but most (80%) of the power is consumed by the DRAM buffer. In comparison, the system with 32 GB DRAM, consumes twice the power. The power increase for both of these systems come from the reduced overall execution time relative to the baseline. For the hybrid memory PCM consumes only 20% of the power. This is because the DRAM buffer filters out most of the PCM accesses and our proposed mechanisms further reduce write accesses to PCM.

Both the hybrid memory and 32GB DRAM-only systems consume much less energy than the baseline. On average, the hybrid system consumes 45% the energy, compared to 59% for the 32GB DRAM system. These results confirm that the PCM-based hybrid memory is a practical power-performance efficient architecture to increase memory capacity.

6. IMPACT OF WRITE ENDURANCE

PCM has significantly less write endurance compared to DRAM. PCM storage in a hybrid memory system receives write traffic when a newly fetched page from disk is installed in PCM, or when a page already present in PCM gets writeback updates from the DRAM buffer. To analyze the impact of write traffic on the lifetime of PCM, we first develop an analytical model.

For a FHz processor that operates for Y years, the PCM main memory must last for $Y \cdot F \cdot 2^{25}$ processor cycles, given that there are $\approx 2^{25}$ seconds in a year. Let PCM be of size S bytes which is written at a rate of B bytes per cycle. Let $Wmax$ be the maximum number of writes that can be done to any PCM cell. Assuming writes can be made uniform for the entire PCM capacity, we have

$$\frac{S}{B} \cdot Wmax = Y \cdot F \cdot 2^{25} \tag{1}$$

$$Wmax = \frac{Y \cdot F \cdot B}{S} \cdot 2^{25} \tag{2}$$

Thus, a 4GHz system with S=32GB written at 1 byte/cycle, must have $Wmax \geq 2^{24}$ to last 4 years. Table 5 shows the average bytes per cycle (BPC) written to the 32 GB PCM storage. The PCM-only system has BPC of 0.317 (average lifetime of 7.6 years), but that is primarily because the system takes much more time than DRAM to execute the same program. Therefore, the lifetime of the PCM-only system is not useful, and in-fact misleading. The more practical PCM memory system is the hybrid with 1GB DRAM which has an average BPC of 0.807 (lifetime of 3 years). To improve PCM lifetime we proposed three[2] techniques to reduce the write traffic from DRAM to PCM. First, *Lazy Write* (Section 3.2) which avoids the first write to PCM for dirty pages. Second, *Line Level WriteBack (LLWB)* (Section 3.3), that tracks which lines in the page are dirty and writes only those lines from DRAM to PCM when the DRAM page is evicted. Third, *Page Level Bypass (PLB)* (Section 3.5) for applications that have poor reuse in PCM. We enable PLB only for *daxpy* and *vdotp*, for which PLB completely eliminates writes to PCM.

Table 5 also shows the write traffic and average lifetime when each of the three optimizations are added one after another to the hybrid memory configuration. Lazy Write reduces BPC from 0.807 to 0.725 (lifetime of 3.4 years). LLWB reduces BPC to 0.316 (average lifetime of 7.6 years) and enabling PLB on top of LLWB increases lifetime to 9.7 years. Thus, design choices and architecture of main memory significantly affects the lifetime of PCM.

[2]The Fine-Grained Wear Leveling (FGWL) technique described in Section 3.4 does not reduce write traffic. It tries to ensure uniform wear-out of lines within a PCM page so that we can use average write traffic in calculating the lifetime (assuming writes across different pages are made uniform using some page-level wear leveling algorithm).

7. RELATED WORK

A NAND Flash based file buffer cache is used in [13] to reduce the main memory power. However, Flash is still 2^8 times slower than DRAM, implying that increasing DRAM capacity is still important to reduce accesses to the Flash-based disk cache. Our work is complementary to [13], and in fact in our evaluations we do assume that the HDD is supported by an aggressive FLASH cache. Furthermore, [13] uses improved error correction for FLASH memories, which does not increase the lifetime of each individual cell. Whereas, we focus on reducing the write traffic to PCM so that each cell wears out after a longer time. Both approaches can be used together for greater benefit than either scheme standalone.

Challenges and issues of using SRAMs for main memory is discussed in [17]. The first level of main memory is an SRAM, followed by the traditional DRAM memory. The goal of the study was to improve the memory speed, and not address the cost, or power limits of the DRAM memory. The SRAM memory is in fact lest denser, and consumes more power than the DRAM memory. However, the challenges highlighted in their work, namely, efficient address translation for a 2-level memory organization, page size optimizations for the SRAM and DRAM, changes in the OS to accommodate two level memories are still valid for our hybrid DRAM and PCM memory systems.

In a related work [7, 8], a multi-level main memory organization is evaluated showing that allocating only 30% of the memory to be accessed in DRAM speed, and the rest at a slower rate does not significantly impact performance, and allows the use of techniques such as memory compression, and dynamic power management of different regions of memory. Their work still uses DRAM as the memory for the multiple levels in the hierarchy, and does not use hybrid memory technologies. However, the key finding in [7] that it is possible to tolerate a longer access latency to a major part of the main memory inspired our work on using hybrid memory technologies to increase the overall main memory capacity.

A virtualization-based methodology to study hybrid memories is presented in [28]. However, that work focuses only on methodology, whereas we focus on management and organization of hybrid memories to get larger capacity while improving the overall system cost, power, and lifetime.

Mangalagiri et al. [18] propose to use PCM based on-chip caches to reduce power. However, we believe that the slower latency and limited endurance of PCM makes it useful only after DRAM and not as on-chip caches.

For increasing DRAM capacity, MetaRAM [20] use the MetaSDRAM chipset between the memory controller and the DRAM which allows up to 4X more mainstream DRAM to be integrated into existing DIMMs without any changes to hardware. However, this still does not address the issue of growth in the cost or power for main memory. It only allows the users who are willing to pay a higher cost for DRAM to have higher capacity within the same motherboard space.

Table 5: Average number of bytes per cycle written to PCM (and PCM lifetime if $Wmax = 10^7$)

Configuration	db1	db2	qsort	bsearch	kmeans	gauss	daxpy	vdotp	Average	Average Lifetime
PCM 32GB	0.161	0.186	0.947	0.127	0.138	0.305	0.400	0.269	0.317	7.6 yrs
+1GB DRAM	1.909	1.907	1.119	0.150	0.190	0.468	0.422	0.288	0.807	3.0 yrs
+ Lazy Write	1.854	1.866	1.004	0.078	0.096	0.354	0.274	0.276	0.725	3.4 yrs
+ LLWB	0.324	0.327	0.797	0.078	0.096	0.354	0.274	0.276	0.316	7.6 yrs
+ PLB	0.324	0.327	0.797	0.078	0.096	0.354	0	0	0.247	9.7 yrs

8. SUMMARY

The need for memory capacity continues to increase while the main memory system consisting of DRAM has started hitting the cost and power wall. An emerging memory technology, Phase Change Memory (PCM), promises much higher density than DRAM and can increase the main memory capacity substantially. However, PCM comes with the drawback of increased access latency and limited number of writes. In this paper we studied the impact of using PCM as main memory and make the following contributions:

1. To our knowledge, this is the first architectural study that proposes and evaluates PCM for main memory systems. We provide an abstract model for PCM and show that for currently projected values, PCM can increase main memory capacity by 4X and reduce page faults by 5X on average.

2. As PCM is slower than DRAM, we recommend that it be used in conjunction with a DRAM buffer. Our evaluations show that a small DRAM buffer (3% the size of PCM storage) can bridge the latency gap between PCM and DRAM. We also show that for a wide variety of workloads the PCM-based hybrid system provides an average speedup of 3X while requiring only 13% area overhead.

3. We propose three techniques: *Lazy Write*, *Line Level Writeback*, and *Page Level Bypass* to reduce the write traffic to PCM. We show that these simple techniques can reduce the write traffic by 3X and increase the average lifetime of PCM from 3 years to 9.7 years.

4. We propose Fine Grained Wear Leveling (FGWL), a simple and low-overhead technique to make the wear-out of PCM storage uniform across all lines in a page. FGWL requires 4-bit per 4KB page.

We believe the architecture-level abstractions used in this work will serve as a starting point for system architects to address the challenges posed by PCM, which will make PCM attractive for the main memory of future systems.

Acknowledgments

The authors thank Ravi Nair, Robert Montoye, Partha Ranganathan and the anonymous reviewers for their comments and feedback. We also thank Pradip Bose for his support during this work.

9. REFERENCES

[1] *The Basics of Phase Change Memory Technology*. http://www.numonyx.com/Documents/WhitePapers/PCM_Basics_WP.pdf.

[2] *International Technology Roadmap for Semiconductors, ITRS 2007*.

[3] Apple Computer Inc. *Apple Products*. http://www.apple.com.

[4] F. Bedeschil et al. A multi-level-cell bipolar-selected phase-change memory. In *2008 IEEE International Solid-State Circuits Conference*, pages 428–430, Feb. 2008.

[5] E. Doller. *Flash Memory Trends and Technologies*. Intel Developer Forum, 2006.

[6] E.Grochowski and R. Halem. Technological impact of magnetic hard disk drives on storage systems. *IBM Systems. Journal*, 42(2):338–346, 2003.

[7] M. Ekman and P. Stenstrom. A case for multi-level main memory. In *WMPI '04: Proceedings of the 3rd workshop on Memory performance issues*, pages 1–8, 2004.

[8] M. Ekman and P. Stenstrom. A cost-effective main memory organization for future servers. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, 2005.

[9] R. Freitas and W. Wilcke. Storage-class memory: The next storage system technology. *IBM Journal of R. and D.*, 52(4/5):439–447, 2008.

[10] HP. *Memory technology evolution: an overview of system memory technologies, technology brief, 7th edition*, 1999.

[11] C.-G. Hwang. Semiconductor memories for IT era. In *2002 IEEE International Solid-State Circuits Conference*, pages 24–27, Feb. 2002.

[12] J. Javanifard et al. A 45nm Self-Aligned-Contact Process 1Gb NOR Flash with 5MB/s Program Speed. In *2008 IEEE International Solid-State Circuits Conference*, pages 424–426, Feb. 2008.

[13] T. Kgil, D. Roberts, and T. Mudge. Improving NAND Flash Based Disk Caches. In *ISCA '08: Proceedings of the 35th annual international symposium on Computer architecture*, pages 327–338, 2008.

[14] K. J. Lee et al. A 90nm 1.8V 512Mb Diode-Switch PRAM with 266 MB/s Read Throughput. *isscc08*, 43(1):150–162, 2008.

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

[16] M-Systems. *TrueFFS Wear-leveling Mechanism*. http://www.dataio.com/pdf/NAND/MSystems/TrueFFS_Wear_Leveling_Mechanism.pdf.

[17] P. Machanick. The Case for SRAM Main Memory. *Computer Architecture News*, 24(5):23–30, 1996.

[18] P. Mangalagiri, K. Sarpatwari, A. Yanamandra, V. Narayanan, Y. Xie, M. J. Irwin, and O. A. Karim. A low-power phase change memory based hybrid cache architecture. In *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, pages 395–398, 2008.

[19] J. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, Dec. 1995.

[20] MetaRAM, Inc. *MetaRAM*. http://www.metaram.com/.

[21] Micron. *Micron System Power Calculator*. http://www.micron.com/support/part_info/powercalc.

[22] D. Nobunagal et al. A 50nm 8Gb NAND Flash Memory with 100MB/s Program Throughput and 200MB/s DDR Interface. In *2008 IEEE International Solid-State Circuits Conference*, pages 426–427, Feb. 2008.

[23] S. R. Ovshinsky. Reversible electrical switching phenomena in disordered structures. *Phys. Rev. Lett.*, 21(20), 1968.

[24] 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 R. and D.*, 52(4/5):465–479, 2008.

[25] J. Tominaga, T. Kikukawa, M. Takahashi, and R. T. Phillips. Structure of the Optical Phase Change Memory Alloy, AgVInSbTe, Determined by Optical Spectroscopy and Electron Diffraction,. *J. Appl. Phys.*, 82(7), 1997.

[26] C. Weissenberg. *Current & Future Main Memory Technology for IA Platforms*. Intel Developer Forum, 2006.

[27] N. Yamada, E. Ohno, K. Nishiuchi, and N. Akahira. Rapid-Phase Transitions of GeTe-Sb2Te3 Pseudobinary Amorphous Thin Films for an Optical Disk Memory. *J. Appl. Phys.*, 69(5), 1991.

[28] D. Ye, A. Pavuluri, C. Waldspurger, B. Tsang, B. Rychlik, and S. Woo. Prototyping a hybrid main memory using a virtual machine monitor. In *Proceedings of the IEEE International Conference on Computer Design, 2008.*