2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium

Using PCM in Next-generation Embedded Space Applications

Alexandre P. Ferreira, Bruce Childers, Rami Melhem and Daniel Mossé
University of Pittsburgh
{*apf75,childers,melhem,mosse*}*@cs.pitt.edu*

Mazin Yousif
Chief technology Officer, IBM Canada
Cloud Computing Global Technology service
ymazin@ca.ibm.com

***Abstract*—Dynamic RAM (DRAM) has been the best technology for main memory for over thirty years. In embedded space applications, radiation hardened DRAM is needed because gamma rays cause transient errors; such rad-hard memories are extremely expensive and power hungry, leading to lower life (or increased battery weight) for satellite and other devices operating in space. Despite these problems, DRAM has been the technology of choice because it has better performance and it scales well. New, more energy efficient, non-volatile, scalable, radiation resistant memory technologies are now available, namely phase-change memory (PCM), making the DRAM choice much less compelling. However, current approaches require changes to PCM device internal circuity, the operating system and/or the CPU cache-memory organization/interface.**

This paper presents a new, practical, detailed architecture, called PMMA, to effectively use PCM for main memory in next-generation embedded space systems. We designed PMMA avoiding changes to commodity PCM devices, the operating system, and the existing CPU cache-memory interface, enabling plug-in replacement of a conventional DRAM main memory by one constructed with PMMA. Our architecture incorporates novel mechanisms to address PCM's limitations including expensive write operations, asymmetric read/write latency, and limited endurance. In our evaluation we show that PMMA achieves a 60% improvement in energy-delay over a conventional DRAM main memory.

I. Introduction

The design of main memory in embedded space applications has become the newest challenge for system designers due to aggressive SWEPT requirements (that is, Size, Weight, Energy, Performance and Time). In addition, next generation space applications have two main requirements. First, high demands for memory capacity, due to the large images being captured and small bandwidth and high power cost of links to move data off the space devices; this leads to high memory utilization for storage and high memory footprint for the application that processes this large amount of data. Second, the high performance of CPUs and the need to use single platforms for multiple purposes puts more strain on the memory subsystem.

Due to such increased demands, main memory has now become quite large, and as a result, it uses a significant portion of system power [1], [2] For example, 2GB of DRAM with 2GB DIMMs consumes 3W to 6W [3], which is equivalent to the total power consumption of the Atom processor [4] or a RAD750 [5].

DRAM has some undesirable properties (e.g., destructive reads and low retention time) that require specific architecture solutions (e.g., write after read operations and refresh control). The future scalability of DRAM has also been called into question beyond 40*nm* sizes [6], [7].

In particular, in space, radiation in the form of ionizing rays is the main cause of SEUs (single event upsets, or transient errors), which typically change the value of bits in DRAM arrays [1]. Thus, DRAM in space devices must be radiation-hardened (rad-hard), by using physically larger devices that are less susceptible to radiation or that requires higher energy radiation to corrupt bits; these rad-hard devices also require higher voltages and charges, and therefore higher power consumption. Thus, minimizing the amount of DRAM in space devices is one of the main goals of system designers.

Space devices are migrating to newer technologies (e.g., Flash [8] and MRAM [9]) to address the issue of reliability within a certain budget (rad-hard DRAM is too expensive). Because the next generation space devices will be untethered and will require large memories (the latest Mars Rovers uses 256MB of RAM and 2GB Flash, but would use more if it were available [8]), some technologies are ruled out due to its small sizes (MRAM is limited to 4Mb chips), or due to cost and power (e.g. rad-hard DRAM).

PCM is a relatively new memory technology [10]–[13] that changes the physical state of the material, offering low power consumption and scalability in comparison to DRAM [7], [14]–[16]. PCM is **low power**, **non-volatile** [17] and **radiation resistant** [18]), with **asymmetric timing and energy for reads/writes** (writes are five to ten times slower than reads and consume more energy due to the need to melt the cell to change its physical state [12]). Another feature of PCM, unlike NAND Flash, is that PCM can do a read/write on a single memory location rather than on memory block (byte addressability). Other operations, such as block erasure before a write (as in Flash) are not required. PCM suffers from limited endurance because a write degrades a cell by reducing the cell's ability to reliably achieve physical state changes. PCM prototypes support at least 10^7 writes, which is orders of magnitude better than Flash [10], [17].

Due to PCM's promise of low power consumption and scalability, there have been recent proposals to use it for main memory (i.e., replacing DRAM) [7], [14], [15], [19].

1080-1812/10 $26.00 © 2010 IEEE
DOI 10.1109/RTAS.2010.40

Technique	PCM Device	Cache Design	Memory Controller	OS	Target
Zhou et al. [15]	Modified		Modified		3-D Stacking
Lee et al. [7]	Modified	Modified	Modified		Conventional
Qureshi et al. [14]	Modified		Modified	Modified	Conventional
PMMA			Modified		Conventional

Table I
COMPONENTS MODIFIED BY DIFFERENT ALTERNATIVES TO USE PCM AS MAIN MEMORY

Although these recent proposals pave the way for PCM main memory, they require modifications in several components, as shown in Table I. Naturally, all approaches require changes to the memory controller to support PCM. The existing proposals require modifications to the internal circuit or structure of PCM [7], [14], [15], which might be appropriate for a custom 3-D stack [15], but are more difficult to justify in commodity PCM devices (e.g., Samsung storage devices [17]). The processor caches would also be changed to support PCM [7]. Unfortunately, many of these modifications could hinder the quick adoption of PCM as main memory.

In this paper, we propose a new and practical architecture, called Phase Change Main Memory Architecture (PMMA), that restricts *all* changes to the memory controller (see Table I and Figure 1). It does not require information or policy support from the OS. Wear-leveling and page management policies are incorporated into the memory controller with our architecture. Furthermore, commodity PCM devices can be used with PMMA, such as the current ones from Samsung. In addition, devices built with some of the proposed PCM circuit changes [7], [15], if they are adopted, are complementary to PMMA and can be incorporated directly. PMMA keeps the existing interface to the CPU-caches and does not modify the cache design. PMMA cleverly subsumes the cache modification in [7] by checking whether data is dirty before it is written.

PMMA introduces more complexity in the memory controller than the alternative approaches. However, the benefits of isolating the mechanisms into a single component to provide *direct plug-in replacement* for the expensive and power-hungry rad-hard ECC DRAM main memory outweighs the downside. Ultimately, an isolated approach may better enable the adoption of PCM for main memory.

PMMA addresses the problems associated with PCM, namely endurance, write vs. read asymmetries, and performance, so that it can be easily used as a replacement for DRAM in embedded systems. PMMA has an enhanced memory controller that interfaces with PCM devices and uses a small, fast memory as metadata storage and a staging area for data transfers to/from the actual PCM devices. PMMA seamlessly replaces the existing DRAM memory controller by presenting the same interface to the CPU as a DRAM memory controller.

The contributions of this paper are: (a) A novel architecture (PMMA) that uses PCM as energy efficient main memory and other memory technologies for metadata and data staging area; (b) A study of the design space to identify and resolve the primary constraints of using PCM for main memory, including design choices and algorithms that achieve impressive power savings and limited performance impact comparing to conventional DRAM; (c) An investigation of the feasibility of current/future use of PCM as main memory; and (d) Mechanisms to improve endurance of PCM main memory.

II. PHASE CHANGE MAIN MEMORY ARCHITECTURE

PMMA uses PCM as a DRAM replacement for main memory. Figure 1 contrasts a current DRAM architecture (left) with PMMA (right). PMMA has two auxiliary components, in addition to PCM devices: the Acceleration and Endurance Buffer (AEB) and the Memory Manager (MM). The AEB is a much smaller and faster but also more power consuming memory than PCM. The MM manages the internal operation of PMMA. It has similar functionality as a memory controller in a conventional DRAM architecture. The MM uses the AEB and PCM as randomly accessible storage and controls the information flow between CPUs, PCM and AEB. PMMA supports multiple pending requests and a split-transaction bus.

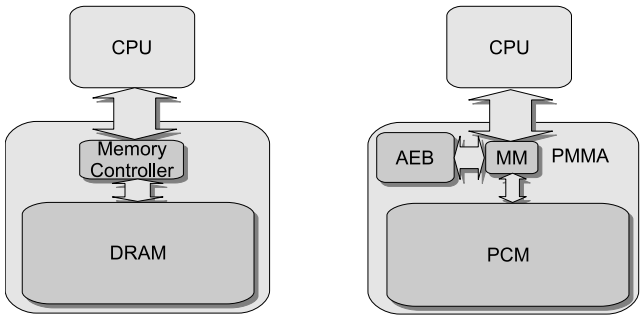

Figure 1. Typical DRAM Architecture and PMMA Architecture

PMMA stores requested pages in the AEB. All CPU write transactions are made to the AEB, not to the PCM. Writes are done to the PCM devices only when a page is evicted from the AEB (i.e., a writeback), reducing the total number of expensive PCM writes, leading to better performance (writes are expensive) and improved endurance. Thus, PCM cells will "live" longer.

We describe the PMMA components below.

PCM: Commodity PCM devices are managed by the MM are connected to the MM via a PCM controller. Current PCM prototypes use an interface similar to DDR DRAM [17] form factor, which is used by PMMA for compatibility. In our design, we use a single PCM bus to reduce controller costs and implementation complexity.

The PCM memory is divided into two areas: (a) The *pages area* holds the actual data stored. The data is arranged in relatively large chunks, called pages. PMMA uses page sizes appropriate to reduce cost and improve performance, as described and evaluated below. (b) The *spares* area holds spare pages that can replace worn-out pages in the PCM devices. In addition, the spares area holds metadata (i.e., a spare table) to map a CPU address to a replacement page if the original page was damaged (see Section III-B).

AEB: The AEB is a module that uses fast memory as a write buffer and high-speed data cache (relative to PCM). However, the AEB is much smaller than a conventional DRAM main memory. We limit the AEB to a single, fast DIMM to reduce physical bus size, load and capacitance, which enables a fast bus for DRAM. The AEB uses DRAM rather SRAM due to capacity and energy considerations. It is around 1% of total PCM capacity (see Section V-A).

The AEB has two areas: a *page cache* and a *spare table*. The page cache acts as a large data cache, used to improve performance and endurance; it holds application data at page granularity. The tags for CPU addresses in the page cache are kept by the faster MM (not in the AEB) because the tag check is on the critical path for any operation.

The AEB also stores metadata – the spare table – for endurance management of the PCM components: it maps CPU addresses to PCM device addresses. It duplicates the spares table on the PCM devices to permit fast lookups. It is accessed only when a request is made to PCM. Note that the page size can have a dramatic impact on the spare table size: a very small page size (e.g., the same L2/L3 cache block size) would result in an exceedingly large table. Thus, less storage space would be left to the pages area.

MM: The internal architecture of the Memory Manager (MM) is shown in Figure 2. The MM includes a request controller, a request buffer, an In-Flight Buffer (IFB), a PCM controller and an AEB (DRAM) controller. The **request controller** receives requests from the CPU interface, allocates resources for the requests and executes the memory transactions on behalf of the CPU. It is the primary component introduced into the memory controller – it is the component that manages the AEB and the PCM devices, while exposing the same memory interface to the CPU. The **request buffer** maintains information about pending requests. It stores the current state of a request, including the request's CPU/DRAM/PCM addresses, size of the transaction and the resources used to handle the request (e.g., buffers and tag array entries). The **In-Flight Buffer** is a temporary data storage area used by the request controller. The **PCM and AEB controllers** contain DMA engines to read and write data from the DRAM and PCM devices at page granularity.

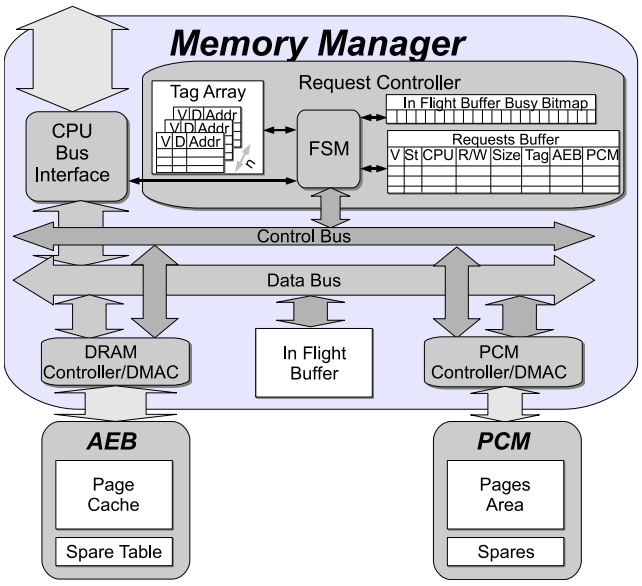

Figure 2. MM Internal Architecture

During the writeback of a dirty page, the MM checks for PCM device wear-out. A *write-read-verify* process is done: an evicted page is written to the PCM device, it is read back and the read data is compared to the evicted page. If page wear-out is detected, a spare is used to replace the "broken" page and the spare tables in the AEB and the PCM are updated. Note that both tables are updated in parallel to ensure fault tolerance (i.e., the mapping is not lost) and good performance (i.e., the table is accessed from the AEB).

Because the AEB acts as a data cache, tags are needed. A **tag array** is incorporated in the MM as part of the request controller to permit a fast tag check (on the critical path of *all* memory requests). We use a large (compared to cache line) page size in PMMA to reduce the size of the tag array.

The In-Flight Buffer (IFB) is used to increase parallelism and decouple the PCM and AEB controllers. It is a small storage area on the MM that receives pages to be transfered to/from the AEB and PCM. The IFB allows multiple transfers to occur simultaneously on the AEB and PCM buses. The write-read-verify process to detect wear-out also uses the IFB. This arrangement decouples wear-out detection from the operation of the AEB and servicing of other requests. The IFB size is based on the number of requests that can be executed simultaneously in the system. The IFB is managed by the request controller through a busy bitmap.

PMMA has a finite-state machine (FSM) that controls the operation of the MM. The FSM uses the current request state and events coming from the DRAM and PCM controllers to determine the next action to execute. The FSM uses the request buffer to track memory requests. The request buffer stores, for each CPU memory request, (1) information about the original transaction (read or write, CPU address, size of the transaction, CPU interface tag); and (2) internal MM information (AEB physical address, IFB buffers allocated and PCM physical address).

PCM and AEB controllers interface PMMA with the actual memory devices. These controllers also schedule requests to their devices and implement acceleration techniques (see Section III-C). Race conditions, hazards are avoided *in this initial implementation* by requiring that all requests to the same page be executed in order.

III. PMMA Operation

A. Basic Operation

PMMA uses the physical address provided by the CPU to identify the page that will be used to fulfill a request.

During operation, the PCM page address of a new CPU request is checked against the request buffer to identify if the page used by this request is cached at the AEB or is being used (processed) by another request. A request can be blocked by a previous request that uses the same page or if a cached copy of the page is being evicted from the AEB. In both situations, the new request will be restarted when the previous requests finishes.

The tag array is checked next. A hit implies that the AEB contains a valid copy of the needed page and the CPU transaction can use the cached version of the page. A miss requires that the page in PCM be brought to the AEB before the CPU transaction is executed. A miss with writeback requires that the evicted page be written back to the PCM before the AEB cache page can be reused. One buffer is allocated per request in the In-Flight Buffer because there can be only one transfer at a time.

Failure detection and correction are implemented as follows. After each write to PCM is completed, a read is done for the written page. If an inconsistency is found, a new spare page is allocated and the process is repeated. PCM reads the spare mapping table to determine where the correct page is located at the PCM.

Further existing optimizations in our current design and implementation:

- The MM checks the tag array speculatively in parallel with checking if page is in (a) use or (b) being evicted. The tag check is ignored if either condition is true.
- For a CPU read, if the needed information is available inside the MM (at the IFB), the data can be immediately forwarded to the CPU.
- For write operations, upon reaching the IFB, the MM sends an acknowledgment to the CPU, because the transaction has already started, even though the full transaction is not completed by using the MM as a write buffer.
- For a miss with writeback, both read and writeback can be done in parallel, using different IFB buffers to store the data. This requires adding intermediate states to the FSM. It also requires that two IFB buffers be allocated (one for reading the AEB and the other for reading the PCM) and the IFB busy bitmap to be updated.

Support for parallelism in the FSM increases the number of requests pending since long latency operations are hidden from the CPU but can still be executed. The IFB must be appropriately sized to allow for number of pending requests.

B. Endurance Management

PCM cells have limited endurance, around 10^7 writes for existing devices. This limited endurance allows a cell to be damaged in seconds if that cell is written continuously at the maximum rate. Even a rate of only one write per second will damage a cell in 110 days. The acceptance of PCM as main memory hinges on its lifetime not being the bottleneck in space applications (must be comparable to a DRAM-based system). Due to the limited endurance of PCM and high write rate of main memory in space devices, PMMA requires mechanisms for endurance management to achieve the desired lifetime.

PMMA uses two approaches to enhance endurance: *write minimization* and *write management*. Write minimization helps endurance by reducing the number of writes that are destined to PCM. One way to minimize writes is to cache pages and coalesce writes at the AEB, avoiding repetitive writes to the same PCM location. The choice of a writeback policy for the AEB and the replacement policy are also designed to improve endurance (see Section III-C).

Write minimization mechanisms reduce but can not eliminate all writes to PCM and, even a small write rate is sufficient to damage a cell during expected lifetime. A complementary scheme, namely write management, avoids damaging a page and/or mitigates the impact of a damaged page. PMMA uses sparing as the chosen mechanism to deal with damaged pages. In sparing, an initial excess capacity is reserved in PCM to replace damaged cells when needed, similar to the extra unadvertised blocks in hard drives. Two major aspects of sparing are determining the necessary spare capacity and the granularity at which the PCM memory will be managed. The required spare capacity depends on the rate at which cells are damaged. This rate depends on the application: those that have poor locality and a high number of writes are the most damaging.

In PMMA, a remapping from the damaged region to the spare region is done at the AEB-resident hash table. For the size of the remapping table to be manageable, a granularity larger than a cell is needed. Care must be taken that the granularity is not too large since a set of cells is replaced

each time any individual cell is damaged. By using the same granularity as the AEB, a miss in the AEB can be mapped to a PCM address with a single query to the table.

The latency and energy *overhead* of the read-after-write failure detection mechanism (i.e., the *read* part) is small because reads are an order of magnitude less costly than writes in terms of energy and delay.

C. Performance and Endurance Enhancements

A **page** is the logical unit of data managed by PMMA. A **block** is the amount of data that is transfered to/from PCM in a single transaction. A block is transfered with multiple bus transfers, each of which is called a **bus transfer unit**. The bus width, block and page sizes are defined by PMMA. Some constraints affect the choice of sizes: a page is larger than a block, which is larger than a bus transfer unit.

PMMA performance and endurance are improved by using a number of techniques described below. These techniques are designed to address the major limitations of PCM, such as large latency and limited PCM bandwidth.

Critical word first. This means that the MM starts reading PCM from the specific address requested by the CPU instead of starting from the beginning of the page. The PCM controller also signals the request controller when all the data necessary to fulfill the request is available, usually before the entire page is available. The use of critical word first can lead to a large reduction in latency.

AEB bypass. If the requested data is available at the IFB, the request controller sends the data from the IFB instead of reading it from the AEB or PCM. The data can be available at the IFB either because the buffer was not yet reclaimed for another transfer operation or the block is still in use by a transfer to the AEB or PCM.

Page Partitioning. As discussed before, too large a page size reduces the amount of metadata; however, it has the negative effect of increasing the transfer time, bandwidth and energy consumption at the PCM and AEB interfaces. Endurance is also negatively affected since a whole page is written even when only a small portion of the page was modified. Dividing pages into subpages (page partitioning) lowers metadata size and offsets the costs of a large page size, as follows. A single entry in the tag array is maintained for each page and valid and dirty bits are kept for each subpage. Because PCM reads are much less expensive (in both energy and time) than writes, we use relatively large subpages for reads and small ones for writes. A small write subpage reduces the number of bytes written to PCM, which lowers the time to write, consumes less energy and increases endurance (the number of writes per subpage will be at most the number of writes per page). This optimization requires each entry of the tag array to have a valid bit per read subpage and a dirty bit per write subpage (recall that sizes of read and write subpages can be different).

Clean-preferred page replacement. We devised a PCM-aware victim selection for AEB page replacement that prefers to evict a clean page. This is because a dirty page requires a writeback to PCM, which is slow and energy expensive. However, avoiding writebacks at all costs may increase the number of misses and impact negatively the performance. In our case we use a 2-chance clean-preferred algorithm that evicts the second oldest victim if the oldest is dirty but the second oldest is not (and does LRU otherwise).

Read-write-read (RWR). Rewriting cells with the same value damages the cell with no benefit. We take advantage of PCM's byte addressability through a novel RWR scheme that adds a read before a write: the original page stored in the PCM is read and compared with the new page to be stored, and only the differences are written back to PCM. We still perform the read after the write, which is the usual read-verify for failure detection. The endurance is increased for applications that write different portions of a page, since the amount of writes per cell is typically much smaller than that per page.

RWR has the most impact on applications that have a high miss rate. In these applications, each page lives in the AEB for a short time period, which reduces the chance to coalesce/combine multiple writes before the page is evicted. RWR is similar to write page partitioning in that both techniques are trying to reduce the number of useless writes, but RWR acts at a smaller granularity. The cost for partitioning is a larger tag array and the cost for RWR is a larger latency and increased PCM bus utilization for writes of highly modified pages.

Critical word first and AEB bypass are very useful and have imperceptible overhead. Thus, they are implemented in PMMA. The effect of page partitioning on performance, effects of AEB replacement policy, RWR impact, and application characteristics on endurance are examined below.

IV. Evaluation Infrastructure

We wrote a simulator using accurate timing and energy models to evaluate PMMA's performance and energy use. The input to the simulator is a memory reference trace, obtained with Simics. The memory trace contains, for each memory request by the CPU, a timestamp (assuming zero memory latency), read/write type and a physical address.

The PMMA simulator has a memory activity generator that processes the memory trace and initiates memory requests to the memory hierarchy. The simulator models the memory manager, PCM controller and devices, and a DRAM controller and devices. The simulator schedules all shared resources and accounts for latency due to resource contention. PCM and DRAM controllers have internal queues of finite size (parameterizable).

The parameters for the results in Section V are in Table II. Two DRAM devices were modeled: (a) the largest capacity

Parameter	PCM	Large DRAM	Fast DRAM
Bus Size (Bits)	8	8	16
# Banks	8	8	8
# Rows	32768	16384	16384
# Columns (Bytes)	1024	1024	1024
Bus Cycle Time (ns)	16.7	3	1.87
Read Latency (ns)	66.8	15	15
Write Latency (ns)	334	15	15
Read Bus (MHz)	66	333	533
Write Bus (MHz)	33	333	533
Idle Current (mA)	1	7	7
Read Current (mA)	10	160	170
Write Current (mA)	70	160	170
Vdd (V)	1.8	1.8	1.8
Max requests in queue	16	16	8

Table II
PARAMETERS USED IN THE SIMULATION

chip available today, and (b) a fast and wide device that can be used for the AEB (which requires a single fast DRAM device). The simulator supports Dynamic Power Management for memory [3]. The PCM timing and power figures are conservative [17].

A request is only issued to the MM after the previous requests finished and the computing time has elapsed. Given the zero-delay memory traces, the memory activity generator determines the request issue time assuming that the CPU sustains a single memory request at a time and the difference between the time stamps is due to computing time and L1/L2 latency. This models a CPU that can sustain a single pending memory request and no write-buffer at the lowest level cache. Recall that the MM/IFB cannot "respond" to some operations before they are completed.

The request buffer and the tag array only process a single event at a time (FIFO) and all events consume a fixed amount of time. Executing an iteration of the FSM is very fast compared to an access to DRAM.

The DRAM and PCM models each have a controller with a finite size input queue, but can reorder requests. The physical memory devices (chips) can be configured with multiple buses, DIMMs and ranks. Each chip can have multiple independent banks and the bus width is configurable in multiples of device bus width. cusses, devices and banks are FIFO scheduled with reordering, which can only occur when a later request can be scheduled without changing the schedule of previous requests or when a higher priority request (a request in the critical path) arrives. This reordering allows low latency requests for banks that are free to proceed when a long latency operation (e.g., PCM write) is happening in another bank.

A PCM or DRAM memory request is mapped to a single bank, so a single transaction will involve only a single bank. The models allow as much parallelism as possible in both PCM and DRAM by interleaving accesses to different bank and buses.

A. Experimental Setup

We used Simics to generate the memory traces with the following configurations: four 1.6GHz x86 processors, each with dedicated L1I and L1D caches (4-way, 32 KBytes, cache line size of 64 bytes and a hit latency of 1 cycle) and a L2 cache shared by core pairs (16-way, 4 MBytes, cache line size of 64 bytes and a hit latency of 6 cycles). The traces were collected until 200 Million main memory accesses were seen (i.e., 200M L2 misses), because a large page makes the AEB warm with a few million requests.

The baseline DRAM main memory system is configured with a 333MHz 64bit DDR2 bus with 16 DIMMs.

The AEB interconnection to the MM is a single 64bit DDR2 bus. In the experiments, we used two different bus speeds: 333 MHz and 533 MHz. The AEB is configured with 4 1Gbit DRAM chips and a bus width of 16 bits.

The PCM memory is arranged as a 16 GB logical address space, with 16 GB reserved as excess capacity. It has a similar configuration as the DRAM-only baseline. The PCM devices are interconnected to the MM via a single 128-bit DDR2 bus, which varies from 66 to 133 MHz. 8 Mbytes are reserved in both the AEB and the PCM to hold the spare mapping table. Each entry in the table is 8 bytes, and the table can map up to 1 million pages.

We chose a set of benchmarks that would represent *future workloads* of satellite systems, and not current small embedded systems benchmarks. All benchmarks have large memory footprints, to represent for example image manipulation and compression. Multithreaded applications and a mix of applications were used to evaluate the design with multicore systems and increased memory pressure. Two PARSEC benchmarks, Canneal and Facesim, were used since they are multithreaded and are very memory intensive applications that represent image manipulation applications. Four SPECcpu2006 benchmarks (GCC, mcf, bzip2, and bwaves) were chosen based on main memory usage [20], representing compute/memory intensive applications. A mix of the SPECcpu2006 applications was used as a mixed workload for a multitasking benchmark (labeled as "SPECmix" in the results). SPECjbb2005 was also used because (a) it has a large memory footprint; (b) it is main memory intensive; (c) it is multithreaded; and (d) it tests memory access patterns influenced by garbage collection (representing hidden system interference).

In the simulations, we record how long it took to serve all memory requests in the trace and the energy used for each PMMA component. We compare PMMA and a conventional DRAM architecture with energy-delay product (i.e., a small performance loss is acceptable if that generates large energy gains).

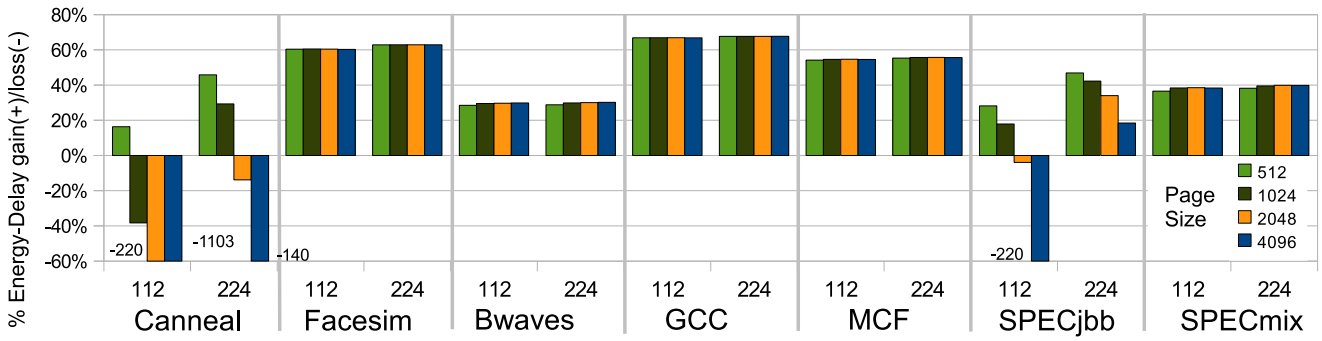

Figure 3. Impact of AEB size (112-224 MBytes) and page size on energy-delay, compared to DRAM-only system

V. Evaluation Results

A. AEB size and Page Size

All values presented in Figures 3 and 4 are relative to a baseline, namely a DRAM main memory subsystem. Basic PMMA includes the use of critical word first and AEB bypass. Positive results indicate PMMA improvements.

Figure 3 compares the impact of different AEB page sizes on energy-delay. Most configurations (AEB and page sizes) have a large gain, up to 65% in energy-delay. Larger AEBs are always beneficial, with Canneal, Bwaves and SPECjbb benefiting more than other applications.

In contrast, page size has a mixed effect. When page sizes are increased, we can see: (a) marginal increase in energy-delay (SPECmix, Bwaves and MCF), (b) much worse energy-delay (Canneal and SPECjbb), and (c) no variation. In other words, increasing page size yields either small gains or extreme negative effects. This clearly shows that small pages should be used.

Figure 3 can be better understood by analyzing each metric separately. Although not shown for lack of space, most applications have energy gains of up to 68% on all configurations and most applications show latency gains.

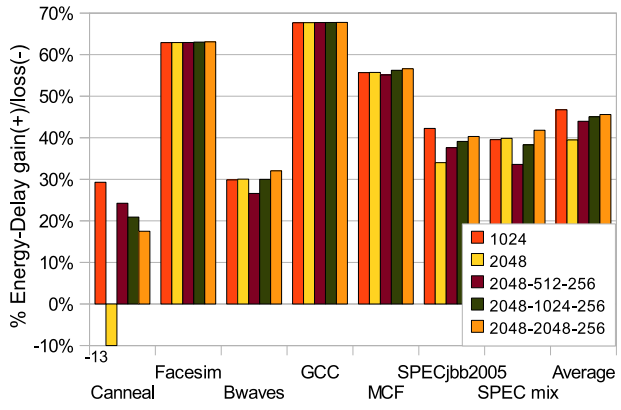

Figure 4. Read/Write Page Partitioning, compared to DRAM-only system; labels are pagesize-read_subpage-write_subpage

The cause of the performance loss for some applications is the high miss rate in the AEB page cache. Canneal has the largest measured AEB miss rate and is the only application that shows an increase in AEB miss rate for larger pages, due to its large memory footprint and lack of spatial locality. Bwaves is also an interesting case: even with a large AEB, the miss rate is large. However, the miss rate is reduced by 75% with a large page size (4KB page). This result indicates strong short distance locality with poor global locality. We can conclude that the beneficial effect of using a large page size (essentially prefetching) is only realized if the miss rate is substantially reduced when a large page size is used. Otherwise, the extra bandwidth use at the PCM interface will offset all gains. Only a few of the benchmarks (facesim, bwaves, and MCF) have sufficient short distance spatial locality to benefit significantly from large pages.

From a design perspective, a small AEB allows for a smaller DRAM memory. From our results, 224MB satisfies the needs of almost all the benchmarks. Page size and partitioning are considered below, since small page sizes are more energy and delay efficient, but require more metadata.

B. Page Partitioning

In Figure 4, the effect of page partitioning on the PMMA is examined, compared to a DRAM-only system. Each configuration with page partitioning is labeled by *page_size–read_subpage_size–write_subpage_size*. Note that write subpages are smaller than read subpages. A smaller subpage for writes always lowers energy consumption and decreases latency; this result can be seen by comparing the results of 2048 (no partitioning) and 2048-2048-256. A smaller subpage for writes increase the tag size since each tag entry size has to store more dirty bits. Results are mixed when subpaging is applied to reads. Applications that have less spatial locality (have high miss rates such as Canneal and SPECjbb) benefit from read subpages because reducing the congestion at the PCM bus improves performance. The

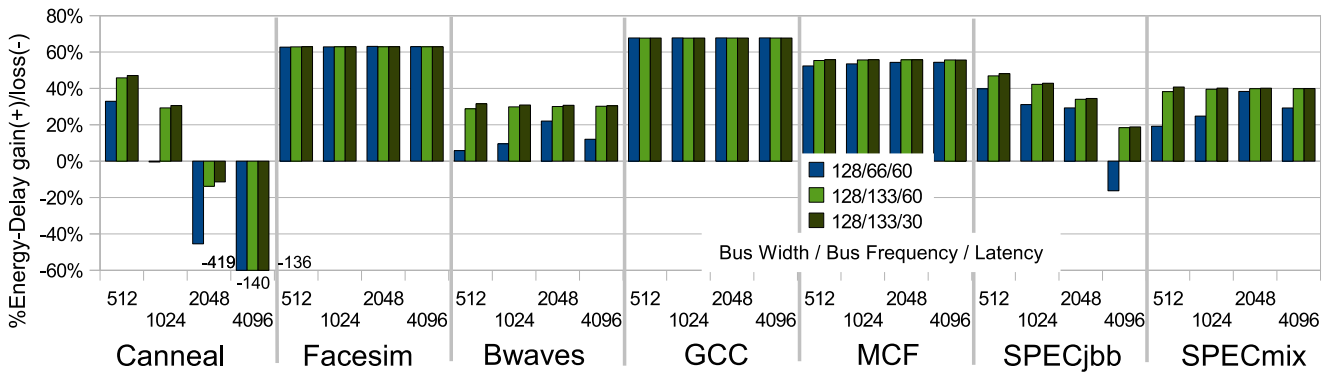

Figure 5. Impact of PCM bus speed and latency at various page sizes, compared to DRAM-only system

other applications (e.g., gcc, facesim, MCF) suffer since any eviction will remove a whole set of subpages but a miss will only bring in one subpage, which loses the prefetching effect of large pages. A 2KB-1KB-256B configuration was chosen as the best compromise among all configurations and will be used as the default configuration below.

C. Technology Constraints

PCM technology: PCM devices prototypes available are slow compared to DRAM devices in production today. In this study a 133MHz device was used, but we carried out a sensitivity analysis to find the impact of PCM technology. Figure 5 shows the impact of reducing by half the speed of the PCM bus and the latency of the PCM devices. Clearly, a lower-speed PCM would hurt some applications that are more sensitive to bandwidth, high miss rate and lower locality (e.g., Canneal). A low-speed PCM would be useful in a small page configuration (ruling out large pages). This result also shows that reducing PCM latency has minimal effect on energy-delay. PMMA is less sensitive to latency than to bandwidth because PCM portion of PMMA is bandwidth limited. This is because a latency reduction only reduces the time to the first byte, having a smaller impact on total transfer time.

Scalability: A performance and energy-delay analysis of a larger PMMA can be estimated based on the assumption that scaling the AEB and PCM proportionally would keep miss rate at a similar value and a larger DRAM would have more idle time. The delay ratio between PMMA delay and DRAM delay would be constant if the scaled PMMA has the same miss rate. The energy ratio would be improved because PCM has a lower energy in idle state than DRAM. Thus, for larger memories, PMMA would increase its advantage in energy and energy-delay over DRAM.

D. Endurance Results

The lifetime of PCM in PMMA is determined by the amount of spares and the rate that an application is able to damage pages. The spare area has to support PCM operation over the specified lifetime. We measured the number of spare pages needed for lifetimes of three and seven years. Two modes of operation were used to measure the spare use rate, as follows.

In Figure 6, all seven applications were run in parallel with each application occupying a different portion of the physical address space. Every time a page reaches a specified number of writes a new spare is allocated to that cell (we simulated 10^6 to 10^8 cell endurance).

Figure 6, at 7 years, shows that the use of clean-preferred page replacement (described in III-C) with RWR at the expected PCM endurance of 10^7 needs about 630K spare pages, accounting for approximately 8% of the visible PCM area. RWR has a large impact and its removal increases the number of spares to about 2.8M pages (about 30%). Using LRU instead of the clean-preferred degrades the results by up to 10%. The results for 3 years are similar. The performance impact of RWR is negligible (less than 0.5%).

In the second mode, all applications were loaded and executed one at a time (that is, they are loaded in the same physical address space). Similar results are shown on Figure 7; about 50% more spares, 963K pages, are needed for a 7-year lifetime. These results show that a spare area of 11% of the PCM visible area is enough to run applications continuously for 7 years.

The number of spares for endurance of 10^6 are quite high, being around 10M of spares, or 120% of the visible area! This result shows that the endurance of PCM has to be equal to or higher than 10^7. For 10^8 endurance, the spare area size (1-2%) is negligible. Sparing is required only due to the outchance thta some cells will still be damaged.

VI. Related Work

Recent works [7], [14], [15] propose to change several components to use PCM as main memory. We aimed to

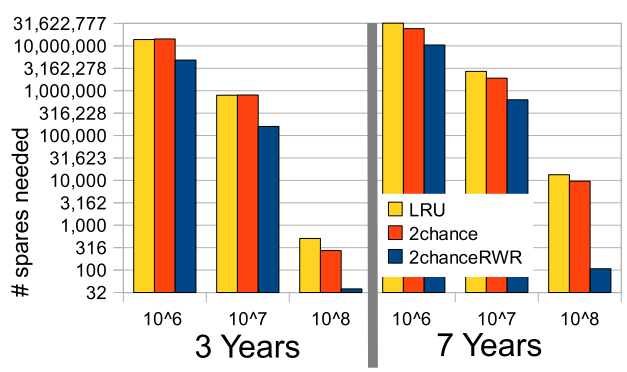

Figure 6. Number of spares needed for parallel execution of all benchmarks, with cell endurance between 10^6 and 10^8

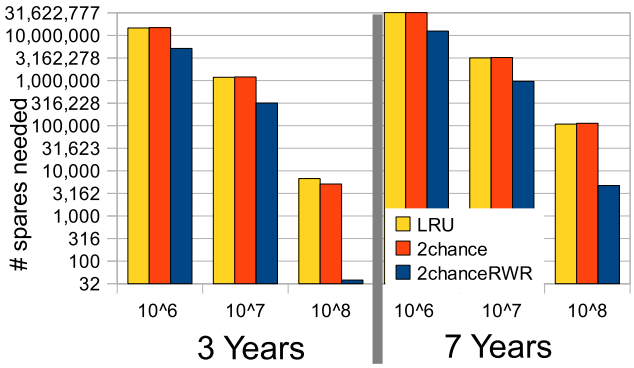

Figure 7. Number of spares needed for sequential execution of all benchmarks, with cell endurance between 10^6 and 10^8

modify a single component, the memory controller, to facilitate the use of commodity PCM and DRAM components.

Qureshi et al.'s proposal [14] has the most similarity to PMMA, where a DRAM cache is used. They rely on a large main memory to achieve good performance by reducing the number of page faults. They modify the OS to accommodate wear-leveling and performance management. In PMMA, the request controller manages the AEB (the DRAM cache) and the PCM devices and uses several techniques to improve performance and endurance: critical word first, AEB bypass, page partitioning, and read-write-read (RWR). These mechanisms ensure good performance from PMMA and minimizes writes to PCM.

Our RWR mechanism is similar to line-level writes [14] and partial writes [7]. It avoids unnecessary writes to PCM. However, RWR does not track the dirty state of cache lines in the CPU cache or the DRAM cache (AEB). Instead, it performs a difference operation on the page to be written and the contents currently stored in the PCM. Only the differences are written. For endurance, fine-grain wear leveling has been proposed by rotating the bytes in a "memory row" in the PCM device [15] or rotating cache blocks in a PCM page [14]. For coarse-grain wear-leveling, swapping heavily written "memory segments" (groups of PCM pages) with lightly written memory segments was proposed [15] or unrealistically assume perfect wear-leveling on page granularity [14]. To improve endurance, PMMA uses sparing (with write minimization). PMMA relies on the future scalability of PCM to provide "extra capacity" (a modest 11%) for cost-effective sparing. The extra capacity is reserved until needed to replace a failed PCM page. RWR detects failure in our approach. For a rotation based mechanism [14], because writes in our benchmarks are relatively evenly distributed over a page, page-level rotation have minimal benefit (but may be useful because much data is read into memory, but only a small portion (i.e., a record) are typically accessed per page. Their rotate mechanism could be incorporated into PMMA for these kinds of applications.

In comparison to circuit techniques [7], [15], RWR in PMMA is related. It aims to minimize the writes like appropriately designed buffers [7] and avoiding redundant bit writes [15]. However, RWR is hosted in the memory controller, rather than integrated into the PCM devices, and does not change PCM's operation, while other techniques assume that PCM memory devices will have a similar organization as DRAM [7], [15]. It is not yet certain whether this will be the case. If PCM devices become available that include these circuit-level techniques, the devices could be directly incorporated in PMMA.

The use of PCM as Flash replacement for file system storage or disk accelerators is well explored. In [21] an algorithm to improve error resilience as a file system accelerator is proposed. Similar work explores the storage of file system metadata on NVRAM (Phase Change Memory or similar) but not as main memory [22]. In [23] non-volatile memory is explored. Dedicated cache replacement algorithms that explore the asymmetric nature of read and writes of Flash are described in [24] for the typical Flash application as secondary storage. Virtual memory is affected by the use of a NVRAM or Flash memories and the work in [25] explores this to created a more efficient secondary storage using NAND Flash.

Finally, DRAM power management is important due to the increase in power consumption caused by larger memory sizes and faster busses. It is a major energy consumer in most systems [26]–[28]. DRAM power management may be done for thermal purposes [28] and for peak power management, where a power budget is allocated and enforced [26]. New memory interconnection technologies, such as FBDRAM, have been introduced to improve performance. [27]. These interconnects, however, are power hungry.

VII. Conclusion

Next-generation space applications will require very large memories, which cannot be achieved with radiation-hardened DRAMs, because of the limitations in size, cost,

and energy. PCM is the new technology of choice in these scenarios, but cannot be used directly as main memory due to performance issues. We proposed a new memory architecture, PMMA, that shows an average of 65% improvement in energy-delay with less than 5% loss in performance. Several performance and endurance enhancements are introduced, such as bypass, page partitioning and read-write-read. A sensitivity analysis of performance impact of the PCM main memory is presented showing that bus bandwidth is the major constraint on performance requiring either a wide bus or higher frequency devices. The use of page partitioning is very beneficial, but we notice that write page partitioning yields the most performance gains due to the write costs and that reads and writes should have different sizes of subpages. PMMA's endurance mechanisms show that with 11% spares and our read-write-read mechanism, a lifetime of seven years is achieved using PCM devices that have a write endurance of only 10^7. Overall, our results show that PMMA has a significant energy improvement over conventional DRAM architectures, while achieving competitive performance and lifetime that is far beyond the usefulness of most space-born systems.

Acknowledgments: Work supported by NSF CCF-0811295, CCF-0811352, ANI-0325353, CNS-0702236

REFERENCES

[1] I. Troxel, "Memory technology for space," in *Military and Aerospace Programmable Logic Devices (MAPLD)*, 2009.

[2] N. AbouGhazaleh, B. Childers, D. Mossé, and R. Melhem, "Power management in external memory using PA-CDRAM," in *The Int'l. Journal for Embedded Systems (IJES)*, vol. 3-1, 2007, pp. 65–72.

[3] Micron, "DDR2 memory power calculator," http://download.micron.com/downloads/misc/ddr2_power_calc_web.xls.

[4] Intel, "Intel microprocessor quick reference guide," http://www.intel.com/pressroom/kits/quickreffam.htm.

[5] L. Burcin, "Rad750 experience: The challenge of see hardening a high performance commercial processor," in *Microelectronics Reliability & Qualification Workshop (MRQW)*, 2002.

[6] "Process integration, devices and structures," in *Int'l. Technology Roadmap for Semiconductors*, 2007.

[7] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, "Architecting phase change memory as a scalable dram alternative," in *Int'l. Symp. on Computer Architecture (ISCA)*, 2009, pp. 2–13.

[8] M. Bajracharya, M. W. Maimone, and D. Helmick, "Autonomy for mars rovers: Past, present, and future," *Computer*, vol. 41, no. 12, pp. 44–50, 2008.

[9] "Japanese research satellite to use magnetic read only memory (mram)," Allbusiness website: http://www.allbusiness.com/electronics/computer-equipment-computer/10196054-1.html, March 2008.

[10] E. Prinz, "The Zen of Nonvolatile Memories," in *Design Automation Conf. (DAC)*, 2006.

[11] Lai and T. Lowrey, "OUM - A 180nm Nonvolatile Memory Cell Element Technology for Stand Alone and Embedded Applications," in *IEEE IEDM*, 2001.

[12] C. Lam, "Phase-change Memory," in *Device Research Conf.*, 2007.

[13] F. Bedeschi et al., "An 8mb demonstrator for high-density 1.8v phase-change memories," in *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symp. on*, 2004, pp. 442–445.

[14] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Int'l. Symp. on Computer Architecture (ISCA)*, 2009, pp. 24–33.

[15] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, "A durable and energy efficient main memory using phase change memory technology," in *Int'l. Symp. on Computer Architecture (ISCA)*, 2009, pp. 14–23.

[16] S. Raoux et al., "Phase-change random access memory: A scalable technology," no. 4-52, 2008, pp. 465–479.

[17] Kang et al, "A 0.1 μm 1.8V 256Mb 66MHz Synchronous Burst PRAM," in *IEEE Int'l. Solid-State Circuits Conf. (ISSCC)*, 2006.

[18] J. Maimon, K. Hunt, J. Rodgers, L. Burcin, and K. Knowles, "Radiation Hardened Phase Change Chalcogenide Memory: Progress and Plans," in *Non-Volatile Memory Technology Symp.*, 2003.

[19] W. Zhang and T. Li, "Exploring phase change memory and 3d die-stacking for power/thermal friendly, fast and durable memory architectures," in *Int'l. Conf. on Parallel Architectures and Compilation Techniques (PACT)*, 2009.

[20] D. Gove, "CPU2006 working set size," *SIGARCH Comput. Archit. News*, vol. 35, no. 1, pp. 90–96, 2007.

[21] K. M. Greenan and E. L. Miller, "PRIMS: Making NVRAM suitable for extremely reliable storage," in *Workshop on Hot Topics in System Dependability (HotDep '07)*, 2007.

[22] I. H. Doh, J. Choi, D. Lee, and S. H. Noh, "Exploiting non-volatile RAM to enhance flash file system performance," in *Int'l. Conf. on Embedded Sofware EMSOFT*, 2007.

[23] E. L. Miller, S. A. Brandt, and D. D. E. Long, "HeRMES: High performance reliable MRAM-enabled storage," in *IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, 2001.

[24] S. yeong Park, D. Jung, J. uk Kang, J. soo Kim, and J. Lee, "CFLRU: A replacement algorithm for flash memory," in *Int'l. Conf. on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, 2006.

[25] H.-W. Tseng, H.-L. Li, and C.-L. Yang, "An energy-efficient virtual memory system with flash memory as the secondary storage," in *Int'l. Symp. on Low Power Electronics and Design (ISLPED)*, 2006.

[26] B. Diniz, D. Guedes, W. M. Jr., and R. Bianchini, "Limiting the power consumption of main memory," in *Int'l. Symp. on Computer Architecture (ISCA)*, 2007, pp. 290–301.

[27] B. Ganesh, A. Jaleel, D. Wang, and B. Jacob, "Fully-buffered dimm memory architectures: Understanding mechanisms, overheads and scaling," in *Int'l. Symp. on High Performance Computer Architecture (HPCA)*, 2007, pp. 109–120.

[28] J. Lin, H. Zheng, Z. Zhu, H. David, and Z. Zhang, "Thermal modeling and management of dram memory systems," in *Int'l. Symp. on Computer Architecture (ISCA)*, 2007, pp. 312–322.