

A COMPARATIVE STUDY OF PHASE CHANGE MEMORY(PCM):
ACHIEVING SIGNIFICANT REDUCTIONS IN ENERGY CONSUMPTION

by

Johny Rufus

A Thesis Submitted to the Faculty of the
DEPARTMENT OF COMPUTER SCIENCE
In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE
In the Graduate College
THE UNIVERSITY OF ARIZONA

2 0 1 1

FINAL EXAMINING COMMITTEE APPROVAL FORM

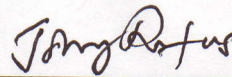
Replace this page with the correct approval form.

STATEMENT BY AUTHOR

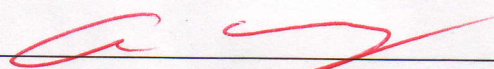
This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: _____

**APPROVAL BY THESIS DIRECTOR**

This thesis has been approved on the date shown below:



Chris Gniady
Assistant Professor, Department of Computer Science

05.12.2011
Date

ACKNOWLEDGEMENTS

It was a pleasure to work with Dr. Chris Gniady. His insightful guidance and inputs have been invaluable to me. Working with my Professor has been a truly enriching experience. I remember once I said I had a problem with the code, and Professor was patient enough to look through the code and help me with the issue. I thank my Professor for his unwavering support and guidance.

I thank Dr. Beichuan Zhang and Dr. David Lowenthal for being on my committee and supporting me through the process.

No amount of words can express the gratitude I have for my family and friends for their support and encouragement. I would like to extend my special thanks to Swami who helped me in writing the document and with latex, with his valuable time. I would like to thank Ajeya my friend for his support and encouraging words.

As a final word, I would like to thank the Almighty, who gave me the strength and hope, to proceed when things seemed impossible.

DEDICATION

Dedicated to my Dad, Mom, Sister, and the Love of my Life

TABLE OF CONTENTS

LIST OF TABLES	7
LIST OF FIGURES	8
ABSTRACT	9
CHAPTER 1 Introduction	10
CHAPTER 2 Simulators	13
2.1 Buffer Cache and Memory Simulator	14
2.2 Disksim	15
2.3 Flashsim	16
2.4 FAST based FTL for PCM	16
2.5 PTL - Design of new FTL for PCM	17
2.5.1 Design	17
2.5.2 Implementation	18
CHAPTER 3 Performance and Energy characteristics of Devices	25
3.1 Hard Disk Drives	25
3.2 Solid State Drives	26
3.3 Phase Change Memory	36
CHAPTER 4 Related Work	39
4.1 PCM as DRAM	39
4.2 PCM DRAM Hybrid Memory	40
4.3 PCM Related	41
CHAPTER 5 Experiments and Results	42
5.1 Methodology	42
5.2 Hard Disk vs PCM	44
5.3 PCM vs SSD	48
5.3.1 Delay	48
5.3.2 Device Energy	50
5.3.3 Memory Energy	51
5.3.4 Total Energy	53
5.3.5 Energy Delay	54
CHAPTER 6 Conclusion and Future Work	57
6.1 Conclusion	57
6.2 Future Work	58
REFERENCES	59

LIST OF TABLES

2.1	SSD Parameters and Values	16
2.2	PCM Parameters and Values	17

LIST OF FIGURES

2.1	Simulator Framework	14
2.2	PCM Translation Layer	24
3.1	Flash Translation Layer	29
3.2	Page Level mapping FTL	30
3.3	Block Level mapping FTL	31
3.4	Hybrid mapping FTL	32
3.5	PCM Comparison	37
5.1	Total Energy and Delay with Hard Disk results included for Postmark, TPC-C, TPC-H	46
5.2	Memory Energy and Device Energy with Hard Disk results included for Postmark, TPC-C, TPC-H	47
5.3	Delay - Postmark	48
5.4	Delay - TPC-C	49
5.5	Delay - TPC-H	49
5.6	Device Energy Postmark	50
5.7	Device Energy TPC-C	50
5.8	Device Energy TPC-H	51
5.9	Memory Energy Postmark	52
5.10	Memory Energy TPC-C	52
5.11	Memory Energy TPC-H	53
5.12	Total Energy Postmark	53
5.13	Total Energy TPC-C	54
5.14	Total Energy TPC-H	54
5.15	Energy Delay - Postmark	55
5.16	Energy Delay - TPC-C	55
5.17	Energy Delay - TPC-H	56

ABSTRACT

Phase Change Memory(PCM) is an emerging technology in the storage hierarchy. PCM is full of promises with low latencies and low energy consumption and high scalability. Most of the research done regarding PCM focuses on using PCM as a DRAM alternative or by using PCM as a hybrid component with DRAM as a part of the primary storage.

Our work focuses on using PCM as a Hard Disk/Flash based SSD alternative. We focus on reducing the total energy consumption of the system, by using the high performance PCM as a disk alternative and experiment with different buffer cache configurations to figure out a way of reducing the memory needed by the system. In the process we develop a new Translation Layer for PCM called PCM Translation Layer (PTL) and develop a simulator based on PTL to conduct our experiments. We try to develop a system with less memory and PCM based secondary storage and strive to maintain the same performance given by a conventional high performance system that uses larger memory and a Disk or Flash based secondary storage. Thus without compromising performance, we are trying to reduce the energy consumption of the system by using PCM as the secondary storage media.

CHAPTER 1

Introduction

As traditional storage devices like hard disk drives and flash based solid state drives keep improving their performance, the inevitable result is the increase in energy consumption. It is difficult to achieve energy consumption by sacrificing performance as the demand for high performance devices keeps increasing constantly. When we look at the total energy consumption of a system, the Random Access Memory consumes a sizable portion, due to the fact that the idle energy consumption of DRAM itself is high. We need to switch to a device that has low energy consumption and very low latencies, which enable us to reduce the total amount of memory used and Phase Change Memory seems to be the most promising technology that can meet the above criteria.

Phase Change Memory has existed for quite some period of time, but recently, the research and work on Phase Change Memory for usage in the storage hierarchy has seen a sudden surge due to the recent developments in the underlying technology of PCM (27) (30). PCM is a non volatile memory that uses chalcogenide glass to switch between two states crystalline and amorphous by applying heat through electrical pulses. The states are identified by figuring out the difference in resistivity between the two states, which differs by a few orders of magnitude.

(22)

A lot of work has been done to study the use of alternate memory technology

to either replace DRAM or augment it, in order to reduce the energy consumption of the main memory (15), (14), (22), (31), (18). Our study focuses on replacing the secondary storage medium with low energy consuming and high performance device like PCM. We then experiment with different buffer cache sizes, to analyze how PCM with less memory configuration fares with the hard disk and flash based SSD with high memory configuration.

PCM has many advantages over hard disk drives and flash based SSDs. PCM has similar energy consumption and lesser read and write latency, when compared to SSDs. PCM also has better endurance than SSDs and is more dense and scalable. When compared to hard disk drives, PCM read and write latencies are orders of magnitude lesser and so are the power consumption values of PCM. This makes PCM the ideal candidate to replace the existing technologies at the secondary storage level (21).

In our experimental study we focus on reducing the the total energy consumption of the system, by using PCM. Since PCM has lower latencies, the total execution time of the system reduces considerably using PCM as the secondary storage, which results in lesser memory idle energy consumption. To measure the performance and energy consumption of PCM, we build a simulator for PCM using the same Flash Translation Layer as our SSD simulator. This simulator can be considered as a drop in replacement PCM simulator which resembles a flash device with the same controller, but the flash hardware alone being replaced by PCM. We refer to this simulator as PCM-FAST, where FAST refers to Fully Associative Sector Translation (16) FTL algorithm used in our SSD simulator.

In our next step, we build a new Translation Layer for PCM called PCM Translation Layer(PTL), which takes into account the specific characteristics of the PCM device. Using our new PTL we build a second PCM simulator PCM-PTL to

be used in our experiments. Additionally we also run our results by wearing the device before running the traces, to study the impact of wear leveling algorithm in PTL.

We then run the standard system traces Postmark, TPC-C and TPC-H on five simulators,

1. Hard Disk simulator
2. Flash based SSD simulator
3. PCM simulator based on Flash FTL algorithm
4. PCM simulator based on our new PTL algorithm‘
5. PCM-PTL simulator thats already pre-worn.

We then analyze the results based on the data obtained by comparing the best case scenarios of HDD, SSD with worst case scenarios of PCM, to find out the real benefits of using PCM.

CHAPTER 2

Simulators

To study the energy consumption and performance aspects of the various devices, we integrate a host of simulators to simulate the functioning of the buffer cache, memory, hard disk, SSD and PCM. We integrated a buffer cache and memory simulator with the latest version of Disksim to build our simulator for the hard disk. To build the flash simulator we took the FTL portion of Flashsim (13) and integrated it on top of our hard disk simulator. We build the simulator for PCM in two ways. The first simulator is built on top of our flash simulator, by using the same FTL algorithm as SSD, but modified to suite the device characteristics of PCM. For the second PCM simulator, we wrote a new FTL algorithm, which we specifically designed for PCM and integrated it on top of Flashsim.

Figure 2.1 provides an overview of the simulator framework used in our experimental study.

We use trace driven simulations as inputs to our simulator for all the three devices. The traces used in our study, PostMark, TPC-C, Viewperf and TPC-H (28) are obtained from the previous work done by Bi et al. (1). PostMark is a standard file system benchmark that operates on a large number of small files. TPC-C is an On-line Transaction Processing benchmark that is heavily read oriented and issues a majority of random I/Os. TPC-H benchmark is based on decision support systems that examines large volume of data in a random fashion.

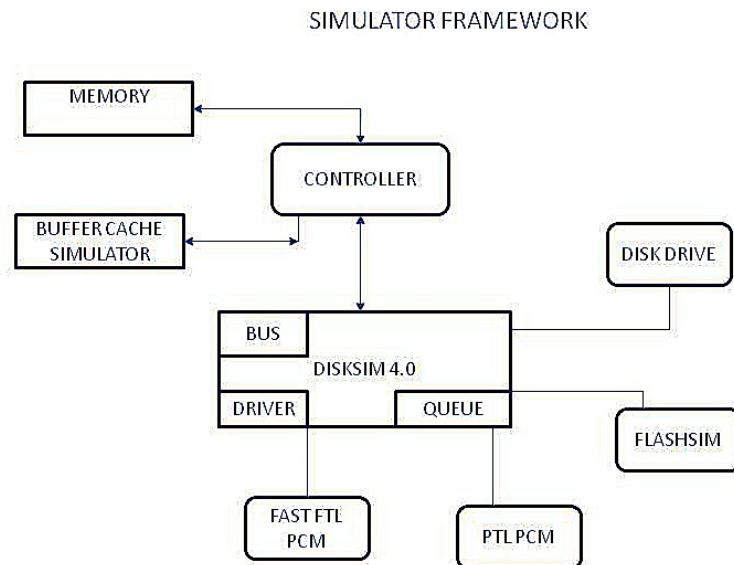


Figure 2.1: Simulator Framework

2.1 Buffer Cache and Memory Simulator

The buffer cache simulator we use is taken from the earlier work done by Butt et al. (3). The simulator implements eight different buffer caching algorithms of which we use only the LRU algorithm for our experiments. We also disable the kernel prefetching mechanism implemented in the simulator. The memory simulator which is taken from the work done by Bi et al. (1) includes a memory controller and configurable DDR2-800 ranks. We vary the number of ranks and the devices per rank, to simulate memory of different capacities. The entire memory is used for the buffer cache, which suites our purpose of study. We disable all the energy saving mechanisms implemented by the memory simulator to try to make the simulator

behave as close to the real world memory devices. In our study whenever we refer to memory, we refer to the memory allocated for the buffer cache.

2.2 Disksim

Our simulation of the disk is based on Disksim 4.0 (2) which is a highly efficient, accurate and configurable disk simulator. Disksim includes simulation of different storage components including device drivers, buses, controllers, and disk drives. The main advantage of disksim is that, it includes hooks for use in a large level system simulator such as the one we develop. Even though the memory and buffer cache simulators are already interfaced with Disksim 3.0, we integrated those simulators with Disksim 4.0, as some of the latest hard drive simulations are supported only in the 4.0 version.

We chose the parameters for the Seagate Cheetah 15k.5 disk with model number ST3146855FC (25), which has a capacity of 146 GB and is a 15000 RPM drive. This is a high performance drive which is relatively new and representative of other competitive hard drives of this generation. The drawback of choosing a high performance hard disk, is the increase in energy consumption. We take advantage of the bus functionality supported by Disksim, to include the latency delay of the transfer of data to and from the device. Based on the Seagate specification for this model, we use the maximum bandwidth value of 125 MB/s for data transfers. The parameters for latency values for the drive are derived by using DIXtrac (29), a program for disk parameter extraction. These parameters provide sufficient characterization to allow extremely accurate simulation of disk performance. From the Seagate specification for Cheetah 15k.5 drive, we use the read and write power as 12 W and idle power as 10.56 W, for energy calculations.

SSD Parameters	Values
Read Latency	.0325 ms
Write Latency	.0425 ms
Read Power	0.150 W
Write Power	0.150 W
Idle Power	0.06 W

Table 2.1: SSD Parameters and Values

2.3 Flashsim

Our simulation of the SSD, is based on the Flashsim developed at PSU which implements the FAST FTL algorithm. Flashsim was written and integrated on top of Disksim 3.0. We made necessary modifications to the Flashsim, to support Disksim 4.0. The parameter values used in the simulator were also changed to reflect specifications from Numonyx and Intel X 25 M (10) SSD device which is an enterprise class high performance SSD device.

The following are the delay and power values we use in our simulator based on Intel X 25 M specification.

The Flashsim supports many FTLs, including Page mapping FTL, DFTL (8) and FAST hybrid FTL (16). Since Page mapping FTL and DFTL are both page mapping based, we have used FAST which is a hybrid FTL for our experimentation purposes.

2.4 FAST based FTL for PCM

Our first simulator for PCM is directly based on the FAST based Flashsim. Since an erase operation is not needed for PCM, we mask the effects of an erase operation in

PCM Parameters	Values
Read Latency	.0032 ms
Write Latency	.0320 ms
Read Power	0.150 W
Write Power	0.150 W
Idle Power	0.06 W

Table 2.2: PCM Parameters and Values

the simulator, to reflect the behavior change. Also the latency and power values are changed to reflect the values for the PCM. This simulator serves two purposes in our study. Even if the performance of SSD increases dramatically in the coming years and comes closer to the latencies of PCM, the values simulated by this simulator reflect the upper limit, such an hypothetical SSD can achieve. Along similar lines, the simulator gives us the lower limit of PCM devices, if when scaled, they are subjected to the same restrictions of SSD.

We derive our performance and power values from Numonyx (21) and Micron specifications for PCM (19) and the values are as follows

2.5 PTL - Design of new FTL for PCM

2.5.1 Design

To take advantages of the characteristics of the PCM, we design a new FTL for PCM devices called PTL which is a PCM Translation Layer. The main design characteristics are

a) Low write amplification All writes to a PCM device are in place. There is no need for an erase operation before a write operation. This is a huge difference

between FTLs written for SSDs, which include a lot of complex merge and log block and erase operations for writes.

b) Low overhead of memory, by using Block level mapping We chose to proceed with the Block level metadata management scheme, by dividing the whole device into blocks. Each block has a fixed number of pages, which contain a fixed set of sectors.

c) Wear Leveling Even though PCM has a better lifetime than SSD, it becomes imperative to employ an efficient wear leveling scheme to distribute the writes across the whole device and prevent premature wearing out of blocks that are heavily written.

2.5.2 Implementation

Figure 2.2 provides an overview of the simulator framework used in this study.

i) Address Translation Layer :

The basic unit of addressing in the device is a page. A page is configured to be of size 2kB and contains four sectors of 512 bytes each. The whole device is divided into blocks. Each block is made up of 256 pages.

LPMAP is a table that stores the logical block to physical block mapping. PBT is a physical block table that contains statistics information on each block. When `pcm_read()` or `pcm_write()` functions are called using logical sector number, the address translation layer decodes the logical sector number into logical block, and retrieves the corresponding physical block. Also it calculates the number of pages to be written based on the input size which is in sectors.

ii) Wear Level Layer :

We use a two level hash table WLT, to store the wear level information. The granularity of wear leveling is in terms of blocks. Each block maintains a variable `curr_wear` which gets incremented when ever a page gets written. There is also a variable called `max_wear` which defines the maximum wear the block can reach, before it needs to be switched with another block. The hash table is indexed based on the current wear.

Each block exists in the WLT based on its current wear. The physical block table and the Logical mapping table point to the appropriate block in the wear level table. As the pages get written, based on the setting of `WEAR_JUMP`, the block gets re positioned in the second level hash table. Once a block reaches its maximum allowed wear, the block's data is copied to a free block or a block with minimum current wear. The block's max wear is incremented by `MAX_WEAR` and the block moves on to the next level in the first level hash table.

Block switching happens when a block has reached its max wear level. If a free block exists, then the current block is switched with the free block and the current block returns to unused state with no logical mapping associated with it. When no free block exists, because all the blocks have been written at least once, then the victim block selected is the one that has the minimum wear level. The two level hash table comes handy in this case, as we need not search for the block with minimum hash individually, since any block that belongs to the lowest non empty first and second level hash will satisfy the purpose. We use a doubly linked list to store the blocks in each bucket of the hash. All operations on the wear level table are performed in constant $O(1)$ time.

iii) Wear Leveling Algorithm :

The following steps describes the wear leveling algorithm in detail.

```

if block current wear + pages to write  $\geq$  block max wear then
    update the block's current wear
    if block needs to be reposition in the second level hash then
        reposition block in second level hash table
    return;
end-if

```

```

end-if

```

```

replacement block = find free block

```

```

if free block exists then
    copy the block contents to replacement block
    update the wear level of replacement block
    update the logical mapping table to point to the replacement block
    mark the block as used
    insert the replacement block first time in the wear level table

    increment the block's current wear
    move the block to the next first level hash
    update the block's max wear to the current hash's max wear
    mark the block as unused
    update the block's logical mapping to -1
    return
end-if

```

```

if free block does not exist then
    replacement block = find minimum wear block in the wear level table
    if replacement block is not in use then
        copy the block contents to replacement block
        update the wear level of replacement block
        update the logical mapping table for the replacement block
        mark the block as used
        reinsert the block in the wear level table

        increment the block's current wear
        move the block to the next first level hash
        update the block's max wear to the current hash's max wear
        mark the block as unused
        update the block's logical mapping to -1
        return
    end-if

    if replacement block is already in use then
        switch the contents in both the blocks
        update the wear level information appropriately
        update the logical mapping table to reflect the switch
        reinsert both the blocks in the wear table using the new values
        return
    end-if
endif

```

iv) **Write Algorithm :**

logical blocks to be written = logical blocks calculated using ATL

foreach block to be written *do*

if logical mapping for block already exists *then*

 physical block = retrieve from the mapping table

else

 physical block = get new block from free list

 make an entry for this new block in the logical table

end-else

 pages to be written = determine offset of pages to write in block

 increment the current wear of the block by number of pages written

 calculate and update the write delay in the total write delay

 do wear leveling for the block

 update the wear level delay in the total wear delay

end-foreach

v) Read Algorithm :

logical blocks to be read = calculate the logical blocks using Address Translation Layer

foreach each block to be read *do*

 physical block = retrieve physical block from the mapping table

 pages to be read = determine offset of pages to read in this block

 calculate and update the read delay in the total write delay

end-foreach

The read and the write algorithms use the address translation layer and the wear leveling algorithm to read or write the appropriate number of pages as per the request.

vi) Delay and Energy Measurements :

Similar to our FAST based PCM simulator, we derive our performance and power values from Numonyx and Micron documents and specifications for PCM as listed in the table 2.4

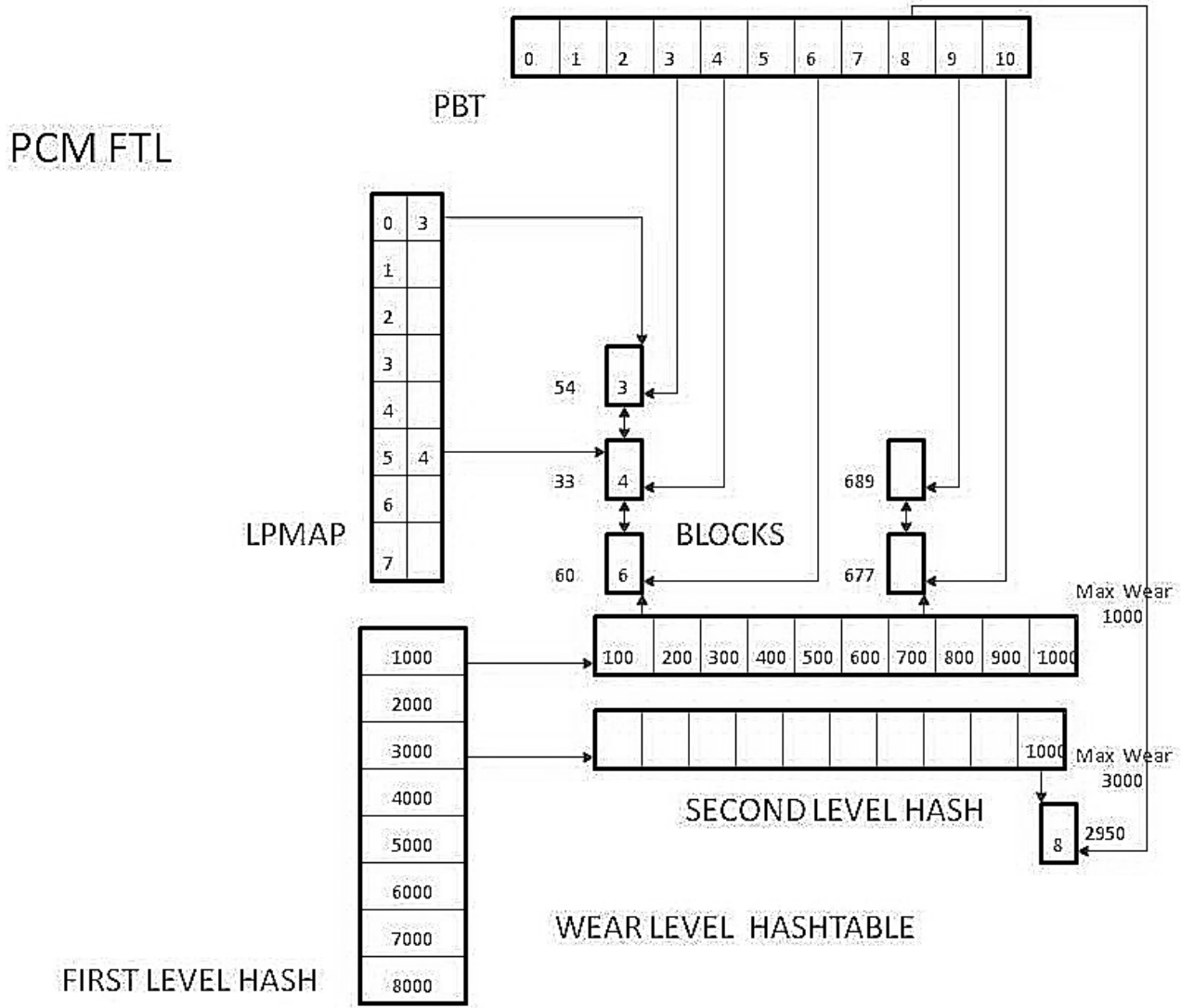


Figure 2.2: PCM Translation Layer

CHAPTER 3

Performance and Energy characteristics of Devices

3.1 Hard Disk Drives

Hard disk drives are random access, non volatile mechanical devices, that has rotating platters mounted on a spindle and read/write heads between the platters. Due to the mechanical nature of the device, the access time and energy consumptions of hard disks are high when compared to other non volatile recording media like SSDs. The biggest advantage of hard disks is the fact that the cost-per-byte for a hard disk remains an order of magnitude lower than for flash memory.

Performance: The access time of HDD is the measure of latency, seek time and data transfer rate. Latency is directly proportional to the revolutions per minute - RPM of the device. The higher the RPM, the lower the value of the latency. On an average a 5400 RPM device incurs a latency of 5.58 ms, whereas a 15000 RPM device has a latency of just 2 ms. Seek time refers to the time taken by the head assembly to travel to the specific track to satisfy a I/O request. Average seek time ranges from 3 ms to 15 ms. Data transfer rate is a measure of how much information can be transferred from the disk to buffer and is a function of the number of blocks transferred. Average data transfer rate is around 128 MB/s. The biggest advantage of hard disks is, they are not limited by the number of writes. There are no limitations on the number of cycles a block can be written before it becomes unusable.

Energy Consumption: Energy consumption of hard disks is a very important factor that affects the total energy consumption of the whole system. In general the active power consumption varies from 4 W to 17 W, and this value is high for high performance drives. So when high performance is needed from hard disks, the increase in power consumption becomes inevitable. The much more hurting factor is the fact that the idle state power consumption of hard disk is also high ranging from 3 W to 14.5 W (9). This implies that not only from a performance perspective, but also from energy consumption perspective, we need to move to other alternatives for hard disks.

3.2 Solid State Drives

Solid State Drives are non volatile block I/O devices like hard drives, but they are made of electronic parts without incurring the disadvantages of a electromechanical device. They retain data in non volatile memory chips. The most prevalently used SSDs are DRAM and NAND-flash based devices.

NOR and NAND Flash: Flash memory is based on the technology of storing the bits in an array of memory cells made of floating-gate tansistors. There are two main types of flash memory, NOR flash and NAND flash. NOR flash is byte accessible, faster but expensive. It's density is also lower compared to NAND flash. NOR flash has a high erase latency due to the restriction that a block has to be filled with zeros before it can be erased. NOR flash has a fast random access and is more suited for executing program code. As discussed in the article (26) NAND flash is suited for mass storage devices and is a better replacement for hard disk drives for the following reasons

1. NAND writes significantly faster than NOR.

2. NAND erases much faster than NOR—4 ms vs. 5 s, respectively.
3. NAND has smaller erase units, so fewer erases are needed.
4. NOR flash has an SRAM interface, whereas NAND uses a block device interface‘
5. Due to significantly lesser cell size, NAND is more affordable in price
6. Life span of NAND is 10 times greater than the cell life span of NOR

SLC and MLC: Single Level Cell devices store only one bit per cell whereas Multi Level Cell devices can store more than one bit per cell. SLC devices have more endurance than MLC devices, but MLC devices have higher capacity. SLC NAND offers high performance and reliability, lower power consumption than MLC NAND devices.

Erase and Write Amplification in NAND Flash: The biggest limitation of NAND devices is due to the restriction that a page has to be first erased before it can be written again. And erasure is not supported for a single page, but erasures are to be done for the whole block. There is no in place update without an erase operation first. Erasure just sets all the bits to 1, after which the cells can be written again. In a device without any efficient controller, a naive write algorithm will work as follows

```

if page A in block B has to be re-written then
    block C = new unwritten block or fully erased block
    foreach page in block B that precedes page A do
        copy the page to the same offset in block C
  
```

```

    end-foreach
    rewrite page A in the correct location in block C
    erase page A and make it available
    update the logical to physical mapping for block C
end-if

```

If we assume a block has 16 pages and block B was partially full, then to do one re write of a page A in a block we ideally have to perform one erase operation and 9 page write operations. This operation is called as a merge operation and it leads to the write amplification problem associated with flash devices and is usually defined as the ratio of the size of data requested to the size of the data that was actually written.

Endurance: The other major factor to consider is the expected lifetime of a flash device. The endurance of a flash device is affected by the number of times a flash cell can be programmed and erased. This value is typically 1000,000 program/erase cycles. So the important factor is, if a particular file gets re-written repeatedly, then the physical pages holding the file gets worn out sooner than the rest, thereby creating unusable blocks. So some kind of wear leveling is needed, to distribute the writes evenly among the physical blocks to ensure the integrity and proper functioning of the device.

FTL - Flash Translation Layer: In simple terms a FTL makes a flash drive appear to the operating system like a disk drive, so that the OS can issue reads and writes in sectors and the FTL handles them through flash specific algorithms. The conversion from virtual address to physical address happens at the FTL. The major difference in the addressing scheme between disk drives and flash drives is that, a flash drive is usually divided into blocks, which is further divided into pages. This is illustrated in the following figure Figure 3.1 taken from (11)

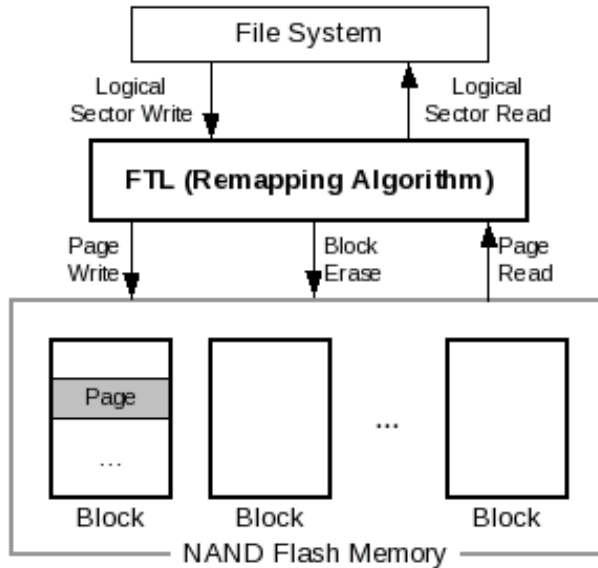


Figure 3.1: Flash Translation Layer

Apart from handling the address translation, the FTL also has two main responsibilities.

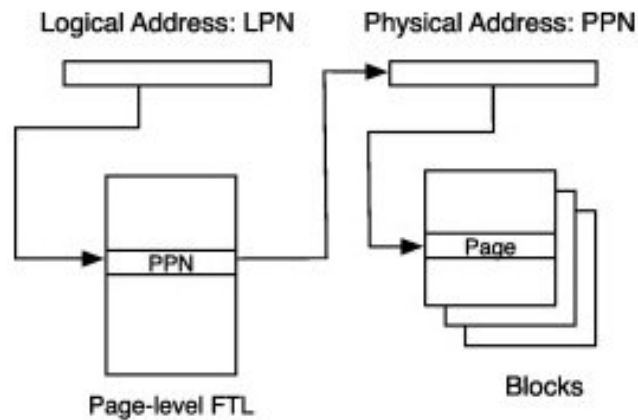
1. To reduce write amplification
2. To maintain wear leveling

FTL Classification: The three broad classifications of FTL are

1. Page level mapping FTL
2. Block level mapping FTL
3. Hybrid mapping FTL

Page-level FTL scheme : In a page-level FTL scheme, the logical page number of the virtual address from the operating system can be mapped into any physical

page number within the flash. The physical page number is then used to calculate the physical block number and the offset of the requested page within the block. Despite the most efficient FTL algorithm possible, this scheme suffers from the obvious issue of maintaining the page table in the cache. For a flash drive with a capacity of 150 GB, which is common today, the amount of SRAM cache needed is approximately 320 MB (8). It becomes infeasible to increase the size of the cache as the capacity increases. The figure 3.2 provides an overview of the page level mapping scheme.

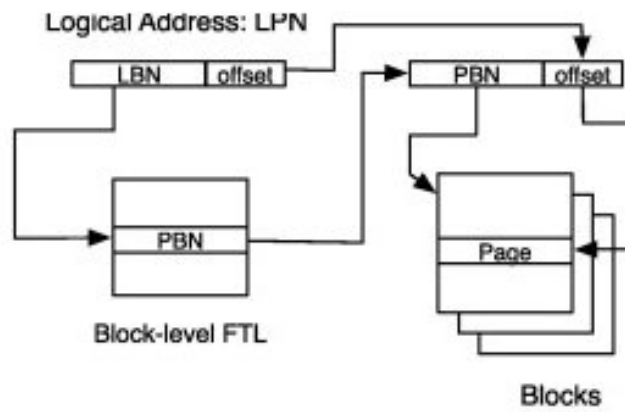


(a) Page-level Address Translation Scheme

Figure 3.2: Page Level mapping FTL

Block-level FTL scheme : In the block level FTL scheme(8), a virtual address is first translated into a logical block number and offset. The logical block number is then used to retrieve a physical block number from the block level mapping table and the offset calculated in the first step is used to retrieve the requested page. The advantage of this scheme is that the block level mapping table contains only one entry per block and hence the size of the block level mapping table reduces drastically. But the disadvantage of this scheme is that, a given logical page, can exist only in a specific offset within a block. Let us assume a write request of one

page was issued and the page has an offset that corresponds to the first page of a block. Now if a rewrite request is issued to the same virtual address, then even though the rest of the pages of this block are unwritten, to overwrite the requested page, a new or erased block has to be selected and its first page has to be written for the given request. The old block that had the page previously, now has to be erased before it can be used again. This leads to extreme write amplification, because if the writes are not sequential then an erase has to be performed in every step. The figure 3.3 taken from (8) provides an overview of the block level mapping scheme.



(b) Block-level Address Translation Scheme

Figure 3.3: Block Level mapping FTL

Hybrid FTL scheme : In the hybrid FTL scheme (8), log buffers are used to address the issues with block level and page level FTL schemes. There are two type of blocks used.

1. Data Blocks : Data blocks are mapped using block level mapping and form the majority of the total number of blocks
2. Log Blocks : A small percentage of the total blocks are designated as log blocks and they are mapped using page level mapping scheme. The logs store

the pages temporarily and each log block can store pages from different data blocks. When an overwrite of a page in the data block occurs, the page is written in the log block and the old page in the data block is invalidated. So each data block is associated a specific log block. So the overwrites are handled this way until the log block spaces are completely used. Then one of the log blocks is chosen as a victim and is merged with all the data blocks for which the log block contains valid pages.

The figure 3.4 taken from taken from (8) provides an overview of the hybrid level mapping scheme.

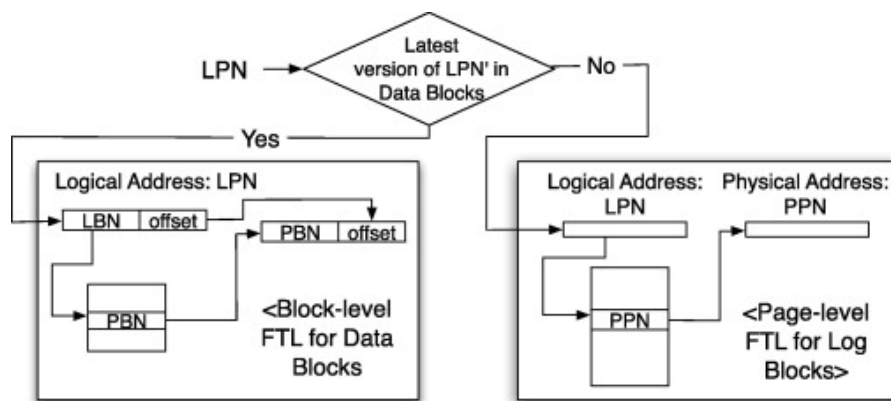


Figure 3.4: Hybrid mapping FTL

The most recent and efficient FTL schemes are all some variant of the Hybrid FTL scheme. The following pages will give a brief overview of the core idea used in these schemes

BAST : Block Associative Sector Translation (12) scheme dedicates a log block to a single data block. Whenever a collision is detected BAST writes the new data to the log block, thereby reducing the number of merge operations totally performed. The algorithm works as follows,


```

if collision exists in trying to write a page to a data block then
    if log block already available for data block in sector level mapping table then
        write the page to the first empty sector in the log block
        update the sector level mapping table
    else
        if free list of log blocks is not empty then
            allocate a log block from free list
            write the page to the first sector in the log block
            update the sector level mapping table
        else
            select a victim log block to make it free
            merge the victim log block and its associated data block
            return the erased blocks in merge to free list
            use one of the blocks in the free list as the required log block
        end-if
    end-if
end-if

```

The merge operation that occurs in log block schemes consists of three steps

1. from the log block and the corresponding data block, select the latest sectors and copy them to a free block
2. update the mapping information
3. erase the data block and the log block and return them to the free list

The major disadvantage of BAST scheme is the log block thrashing, which happens if when an application runs, a large number of collisions occur and the log block cache is not provisioned enough to handle all the collisions. Log block victims have to be selected periodically and each victim replacement will cost an expensive merge operation.

FAST :Fully Associative Sector Translation (16) is a scheme that uses log block FTL scheme to reduce write amplification by associating a log block with multiple data blocks thereby making it fully associative in contrast to the BAST scheme. It reduces the log block thrashing problem of BAST by using sequential write log blocks and random write log blocks.

The algorithm is as follows as given by (16) :

```

logical block number lbn = calculate from the input sector number
page offset, pb n= calculate from the input sector number
physical block number pbn = gete from physical block table
if no overwrite at the offset in the block then
    write the data at the offset of the pbn
    return;
end-if
if offset == 0 then
    if there are no empty sectors in the SW log block then
        perform a switch operation between the SW log block and
        its corresponding data block;
    else
        merge the SW log block with its corresponding data block;
    end-if

```

```

get a block from the free block list and use it as a SW log block;
append data to the SW log block;
update the SW log block part of the sector mapping table;
else
  if the current owner of the SW log block is the same with lbn then
    last'lsn := getLastlsnFromSMT(lbn);
    if lsn is equivalent with (last'lsn+1) then
      append data to the SW log block;
    else
      merge the SW log block with its corresponding data block;
      get a block from the free block list and use it as a SW log block;
    end-if
  end-if

  update the SW log block part of the sector mapping table;
else
  if there are no rooms in the RW log blocks to write data then
    select the first block of the RW log block list as a victim;
    merge the victim with its corresponding data block;
    get a block from the free block list and add it to the end of
    the RW log block list;
    update the RW log block part of the sector mapping table;
  end-if
end-if

append data to the RW log blocks;

end-if
end-if

```

LAST:

Locality-Aware Sector Translation FTL scheme (17) employs multiple multiple sequential log blocks to exploit the spatial locality in workloads. It maintains separate hot and cold regions for random log blocks to reduce full merge cost. The problem is that it depends on external locality detection mechanism to maintain the hot and cold regions and the accuracy of the detection mechanism affects the efficiency of the algorithm.

3.3 Phase Change Memory

Phase Change Memory is a new and progressing technology that has many attractive features making it suitable to be used as either primary memory or as secondary storage. PCM uses the crystalline and amorphous states of chalcogenide glass to store the bits. Two more additional distinct states, are also possible which enables PCM to have double the capacity with the same size. Even though the technology dates back to 1960s, recent advancements from Intel and Micron has paved way for PCM to enter into the mainstream storage arena (4).

In a PCM, the amorphous, high resistance state is used to represent a binary 1, and the crystalline, low resistance state represents a binary 0. They can be switched rapidly back and forth between amorphous and crystalline phases by applying appropriate heat pulses. Also reading the values is fairly simple by measuring the resistance of the cell, as there is a huge difference in the resistance value between the two states (4).

PCM vs other technologies : The figure 3.5 taken from (4) provides a comparison of PCM with DRAM and flash based SSD.

	DRAM	PCM	NAND Flash	HDD
Read energy	0.8 J/GB	1 J/GB	1.5 J/GB [28]	65 J/GB
Write energy	1.2 J/GB	6 J/GB	17.5 J/GB [28]	65 J/GB
Idle power	~100 mW/GB	~1 mW/GB	1-10 mW/GB	~10 W/TB
Endurance	∞	$10^6 - 10^8$	$10^4 - 10^5$	∞
Page size	64B	64B	4KB	512B
Page read latency	20-50ns	~ 50ns	~ 25 μ s	~ 5 ms
Page write latency	20-50ns	~ 1 μ s	~ 500 μ s	~ 5 ms
Write bandwidth	~GB/s per die	50-100 MB/s per die	5-40 MB/s per die	~200MB/s per drive
Erase latency	N/A	N/A	~ 2 ms	N/A
Density	1 \times	2 - 4 \times	4 \times	N/A

Figure 3.5: PCM Comparison

PCM vs DRAM : When compared to DRAM, PCM is byte-addressable similar to DRAM, but the idle power consumption of PCM is significantly lower than DRAM. The read energy consumption of PCM is almost similar to DRAM, but the write energy consumption is 6x higher than DRAM. Similarly PCM read latency is comparable to DRAM, whereas the write delay is 20x slower than DRAM. Also the life cycle endurance of PCM is in the order of $10^6 - 10^8$ (4) , whereas a DRAM has theoretically infinite endurance. PCM used as a main memory can last only for around 100 days running a typical SPEC CPU program (26). Because of the delay, energy consumption and restricted endurance of performing writes in PCM, it becomes increasingly difficult to replace DRAM with PCM, unless special algorithms are used in PCM and the memory management to overcome the limitations. The biggest advantage that PCM has over DRAM as mentioned earlier is the low idle power consumption of PCM, which can be effectively used to reduce the overall energy consumption of the system. The technology of PCM also offers the following advantages when compared to DRAM (14)

1. High Scalability and density, at least 4x times more than DRAM
2. Zero Leakage power due to non-volatile storage
3. Low burst read latencies - due to the peripheral logic of PCM
4. Immune to cross talk, whereas DRAM cross talk is high when scaled ; 65 nm
5. No need for periodic refresh

PCM vs SSD :

PCM seems to fare well when compared to SSDs in almost every aspect. The read latency of PCM is 12x lesser than SSD, the write latency of PCM is 1.5x times lesser than SSD. The read, write and energy consumption of PCM is similar and comparable to SSD. But the endurance and lifetime of PCM is 10^3 x better than SSD, which makes PCM an excellent choice as a secondary storage device. Also the other big advantage of PCM is the fact that no erase is needed before a write, making writes faster and this obviates the need for complex algorithms used in FTL for SSD. There is no erase operation in PCM and also the write amplification of PCM is very less as the only factor contributing to write amplification is the wear leveling algorithm whose impact is significantly less as we will see later in our simulator results.

CHAPTER 4

Related Work

A lot of work has been done recently in trying to use Phase Change Memory at different levels of the storage hierarchy. Most of the work done focuses on overcoming the following three main disadvantages of using PCM

1. high write latency
2. high write energy
3. finite endurance

4.1 PCM as DRAM

Lee et al. have proposed schemes to use PCM as a DRAM alternative. They propose buffer re-organization and partial write methodologies and focus on addressing the three main issues that limit PCM in using it as a scalable DRAM alternative (15)

Krishnaswami et al. try to make architectural changes in PCM to overcome its limitations. They try to eliminate redundant Writes, use row shifting and Segment Swapping and do Partial writes to overcome the limitations of PCM in using it as a DRAM alternative. (14)

Alexandre P. Ferreira et al. use similar techniques to overcome the limitations of PCM by using new cache replacement policies, reducing writes and using

novel endurance management techniques.(7)

Dong et al. developed a simulator based PCRAM model called PCRAMsim to study PCM-based main memory and cache. Their simulator is based on CACTI (24) to provide a system-level tool beyond the device-level research of PCM by automating the process of finding an optimal PCM array organization. Their focus is on evaluating PCM as a main memory and cache system.(6)

4.2 PCM DRAM Hybrid Memory

Qureshi et al. propose a hybrid memory technology where they use a PCM based main memory coupled with a small DRAM buffer. They use Lazy Write Technique , line-level writes, page level bypass and fine-grained wear leveling to address the issues with PCM.(22)

Ramos et al. propose a new DRAM+PCM memory system design that is robust across a wide range of workloads by using a sophisticated memory controller that implements a page placement policy called Rank-based Page Placement. The policy efficiently ranks pages according to popularity (access frequency) and write intensity, migrating top-ranked pages to DRAM. (23)

Mangalagiri et al. propose a hybrid-cache architecture, that address challenges associated with the write behavior of PCM and develop a PCM based cache simulator to evaluate their results.(18)

Mogul et al. have investigated operating system support for placing either flash or PCM on the memory bus alongside DRAM. They focus on FLAM a hybrid of Flash and DRAM and investigate the consequences of PCM replacing flash in FLAM. (20)

4.3 PCM Related

Ping Zhou et.al mainly concentrate on improving the endurance of the Phase Change Memory technology by using row shifting and Segment Swapping and try to increase the lifetime of all PCM cell type to 1322 years.(31)

Chen et al. analyze the unique characteristics of PCM, and their potential impact on database system design. They illustrate that current approaches for common database algorithms such as B+ -trees and Hash Joins are suboptimal for PCM and present improved algorithms that reduce both execution time and energy on PCM while increasing write endurance.(4)

Condit et al. present a file system and a hardware architecture that are designed around the properties of persistent, byte addressable memory like PCM. They have designed the PCM system to be used on the memory bus directly and focus on the file system aspects of the design.(5)

From all the related work done on PCM, we can deduct that most of the work is focused on using PCM as memory or cache in the storage hierarchy. None of the work treats PCM as a replacement for hard disk and SSDs as secondary storage and ours is the first attempt in evaluating PCM as a secondary storage device to reduce the total memory used and achieve significant reduction in energy consumption.

CHAPTER 5

Experiments and Results

5.1 Methodology

We have compared the results of running the following traces Postmark, TPC-C and TPC-H on the following devices : Hard Disk simulator, Flash based SSD simulator, PCM simulator based on Flash FTL algorithm, PCM simulator based on our new PTL algorithm and PCM-PTL simulator that's already pre-worn with the following Memory configurations 512 MB, 1 GB, 2 GB, 4 GB

Our goal is to compare the best case performance and energy consumption of HDD and SSD with the worst case performance and energy consumption of PCM and try to minimize energy consumption by reducing the size of the memory. During the run of the traces, we measure the Device Active time and the Memory Active time, and calculate the memory and device idle times based on best and worst case scenarios for different devices and finally compute the energy consumption. We calculate our upper bound and lower bound idle times using the following algorithm

Memory and Device Idle Time Calculation.

M_{at} = Total Memory Active time

D_{at} = Total Device Active time

M_{ie} = Memory Idle Energy per unit

D_{ie} = Device Idle Energy per unit

M_{ae} = Memory Active Energy per unit

D_{ae} = Device Active Energy per unit

ME_{LB} = Total Memory Energy Lower Bound

DE_{LB} = Total Device Energy Lower Bound

ME_{UB} = Total Memory Energy Upper Bound

DE_{UB} = Total Device Energy Upper Bound

Memory Energy Calculation

Condition : When either memory or device is active

Case 1 : $M_{at} < D_{at}$

$$ME_{LB} = ((D_{at} - M_{at}) * M_{ie}) + M_{at} * M_{ae}$$

$$ME_{UB} = (D_{at} * M_{ie}) + M_{at} * M_{ae}$$

end-Case 1

Case 2 : $M_{at} > D_{at}$

$$ME_{LB} = M_{at} * M_{ae}$$

$$ME_{UB} = (D_{at} * M_{ie}) + M_{at} * M_{ae}$$

end-Case 2

Device Energy Calculation

Condition : When either memory or device is active

Case 1 : $M_{at} < D_{at}$

$$DE_{LB} = D_{at} * D_{ae}$$

$$DE_{UB} = (M_{at} * D_{ie}) + D_{at} * D_{ae}$$

end-Case 1

Case 2 : $M_{at} > D_{at}$

$$DE_{LB} = D_{at} * D_{ae} + ((M_{at} - D_{at}) * D_{ie})$$

$$DE_{UB} = (M_{at} * D_{ie}) + D_{at} * D_{ae}$$

end-Case 2

We know the exact M_{at} and D_{at} from our run of the traces. The memory and the device idle time depends on a number of factors including the processor time. We are trying look at the factors that get affected when a Disk is replaced by PCM. When the processor is active and both the device and the memory are idle, the idle time will not vary much by changing the devices. So this idle time cannot be reduced when disk is replaced by PCM. Similarly how long the Disk was idle when memory was being accessed and how long the memory was idle when disk was accessed will vary between different executions. Hence we concentrate on the upper bound and lower bound of memory and device idle time. By using the lower bound for Disk and SSD and by using the upper bound for PCM, we take the best case scenarios for Disk and SSD and the worst case scenario for PCM as described in our algorithm, and try to figure out the energy savings under these circumstances.

5.2 Hard Disk vs PCM

Since the values obtained for Hard disks are very large when compared to the other devices, we include the hard disk values in 5.1 and 5.2, and discuss its impact first and in the later graphs we discuss the results based on SSD and PCM which have more comparable values.

When we look at the Total energy consumption of hard disks, from the

graph we can deduce that if we replace a 4 GB memory + HDD configuration with 512 MB + PCM , PCM consumes energy many orders of magnitude lesser than HDD for all the three traces and still incurs a delay at least 6 times lesser than the HDD. This is on expected lines for all the three traces because the idle power consumption of HDD is around 10 W whereas PCM consumes only 0.06 W, and the delay incurred by HDD to satisfy individual request is in order of a few , whereas PCM delays are in the order of nano seconds.

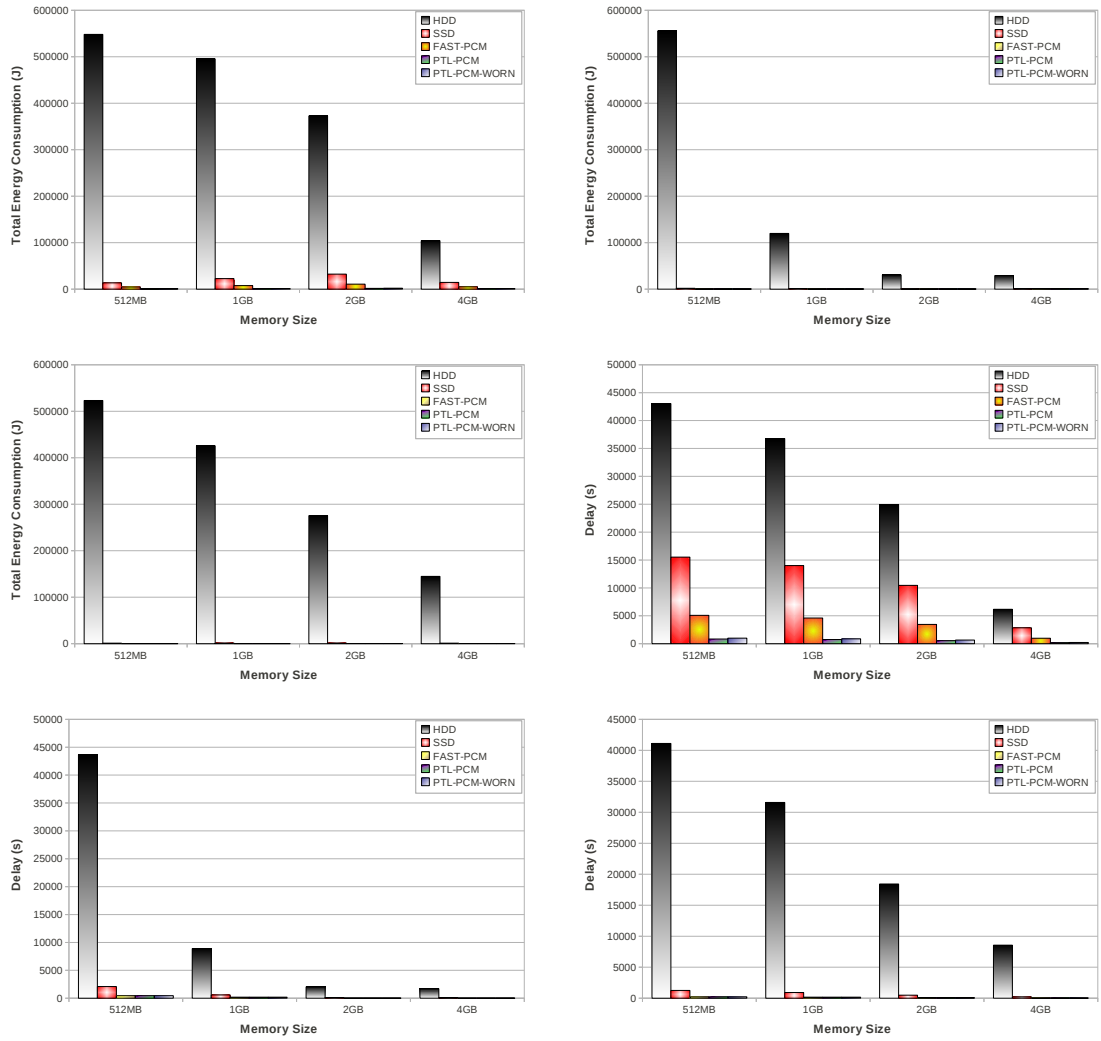


Figure 5.1: Total Energy and Delay with Hard Disk results included for Postmark, TPC-C, TPC-H

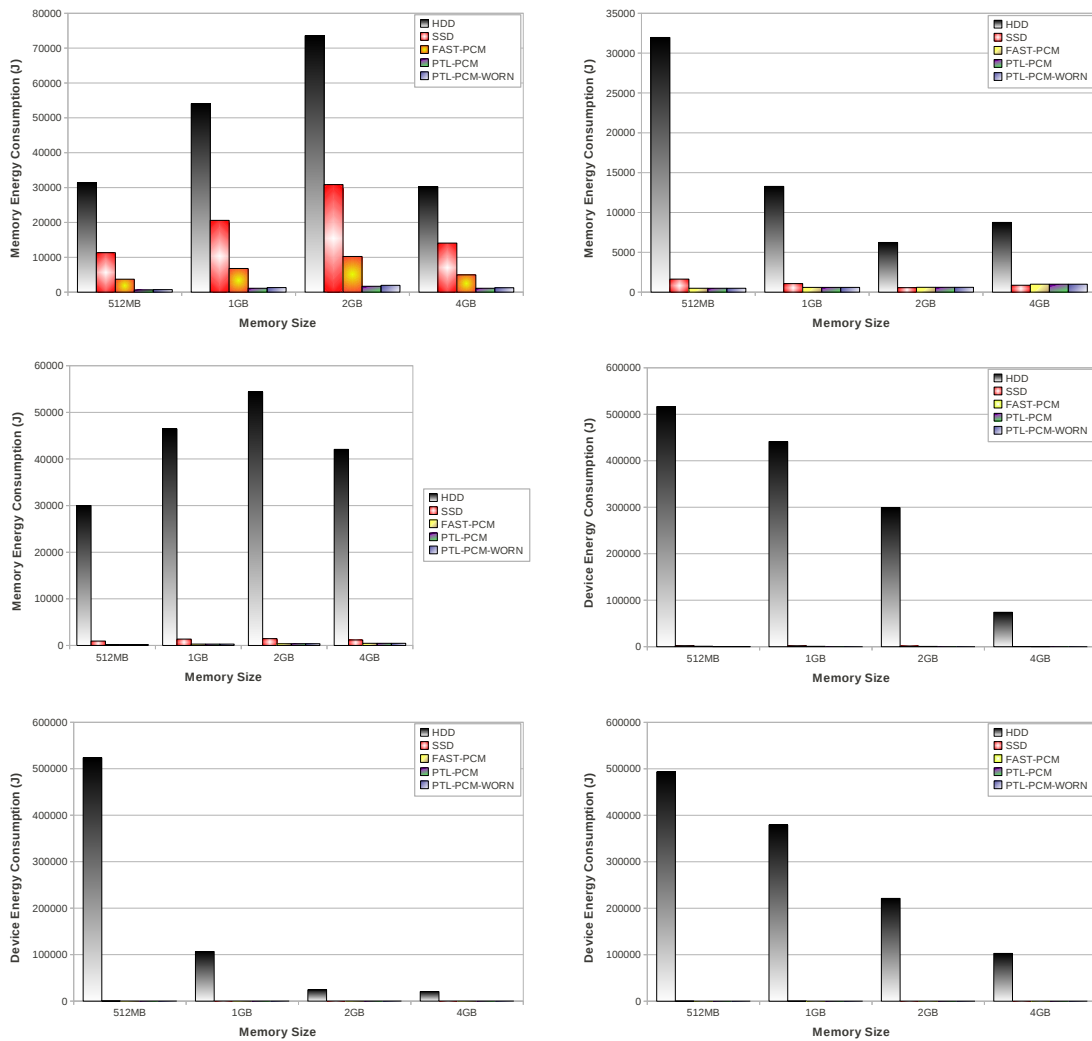


Figure 5.2: Memory Energy and Device Energy with Hard Disk results included for Postmark, TPC-C, TPC-H

5.3 PCM vs SSD

5.3.1 Delay

When we look at the delay graph for Postmark 5.3, there is a huge performance difference between the PCM-FAST and PCM-PTL. Postmark issues a significant amount of writes, and the write cost of the algorithm in FAST is high. When we use our PTL algorithm, the delay reduces a lot, as we use a simplified wear leveling scheme and do in place writes which is suitable for PCM. Also the PCM-PTL-WORN slightly incurs more delay, because we wear out the device completely before executing the trace, which leads to block switching when the worn out block needs to be exchanged with a less worn block. The difference is not huge implying that the wear level algorithm increases the delay only by a factor of 1.8x in the worst case. When PCM is compared with SSD, we get a delay reduction of at least 12x. This is a direct result of the low latency of PCM and the in place write supported by PTL.

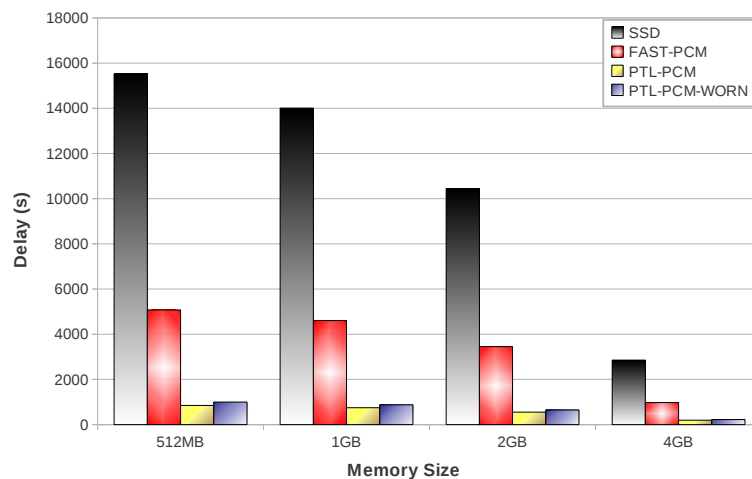


Figure 5.3: Delay - Postmark

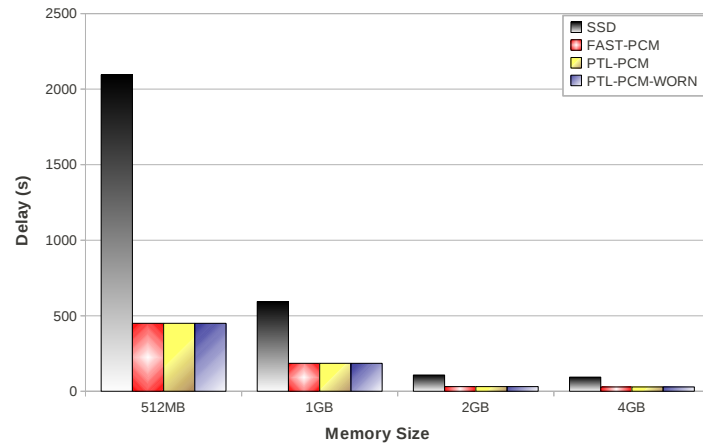


Figure 5.4: Delay - TPC-C

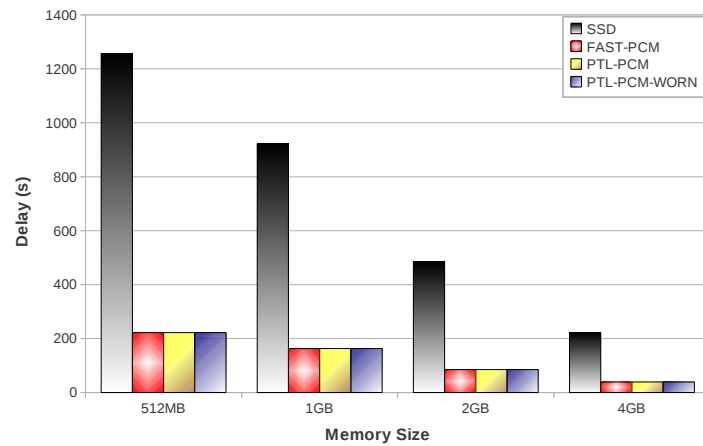


Figure 5.5: Delay - TPC-H

In the delay graph for TPC-C 5.4, shows that the difference in delay between using 2 GB and 4 GB of memory is not huge, implying that the memory requirements of TPC-C gets satisfied with 2 GB of memory itself. The delay graph for TPC-H 5.5 is on expected lines, and since this is a heavily read oriented workload, all the three PCM simulators exhibit similar delays.

5.3.2 Device Energy

The device energy consumption for all the three traces are a direct result of the delay incurred by the devices as seen in the above graphs and they follow a similar pattern.

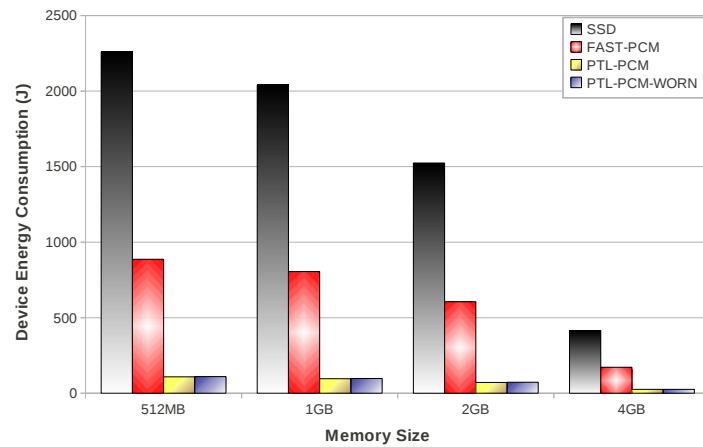


Figure 5.6: Device Energy Postmark

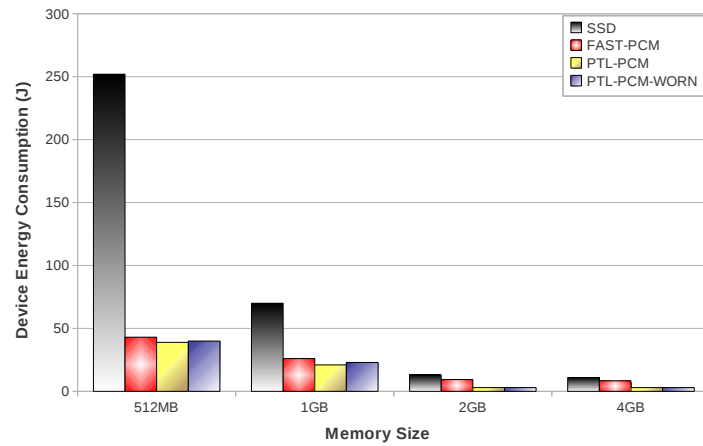


Figure 5.7: Device Energy TPC-C

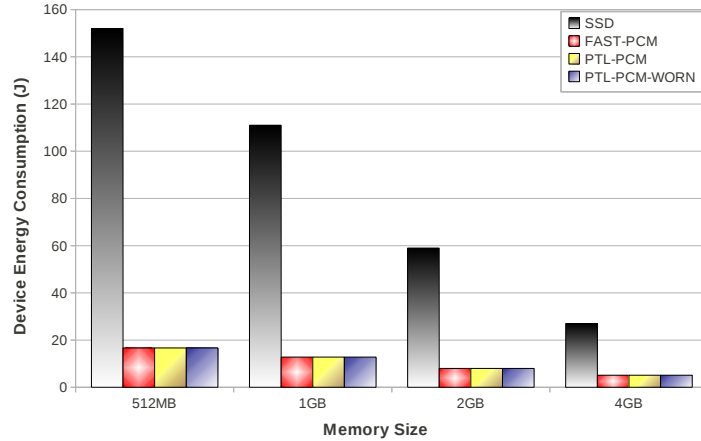


Figure 5.8: Device Energy TPC-H

5.3.3 Memory Energy

The memory energy consumption for postmark looks a bit different. As we increase the memory size from 512 MB to 1 GB to 2 GB, the memory energy consumed increases. This is because the available memory is not good enough to satisfy the current set of requests and a lot of calls still go to the device and so the idle memory energy is high. But when we move to 4GB of memory, there is a sudden drop in the delay incurred as seen in 5.3, which leads to a drastic reduction in the memory idle time, and hence the total memory energy decreases. In the memory energy for TPC-C for SSDs, the sudden drop in delay is seen when we move from 512 MB to 1GB and from 1 GB to 2 GB, hence we see the memory energy decreasing. But from 2 GB to 4 GB, the extra memory added does not significantly decrease the delay and hence the total memory energy increases. For PCM, the memory active time is higher than the device active time, the memory energy directly depends on the amount of memory used and is reflected in the graph 5.10. The memory energy of TPC-H, follows similar pattern to postmark trace.

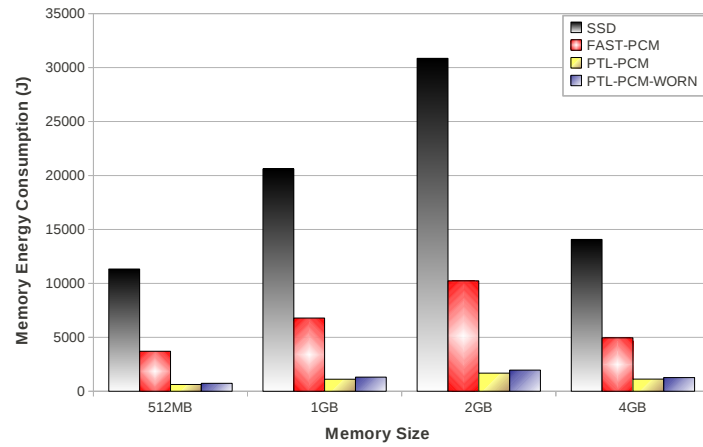


Figure 5.9: Memory Energy Postmark

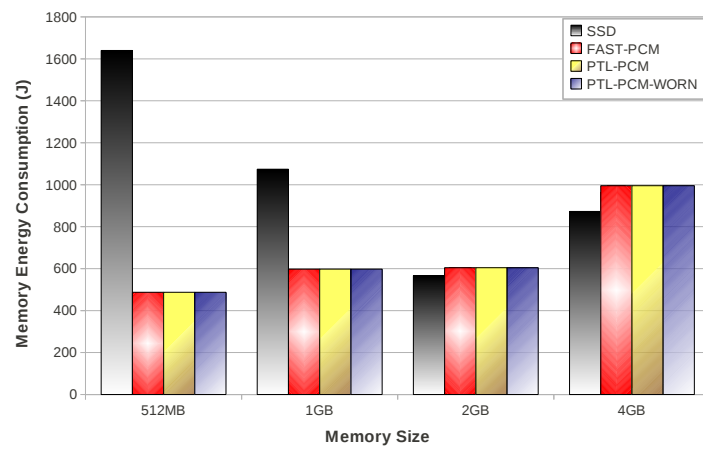


Figure 5.10: Memory Energy TPC-C

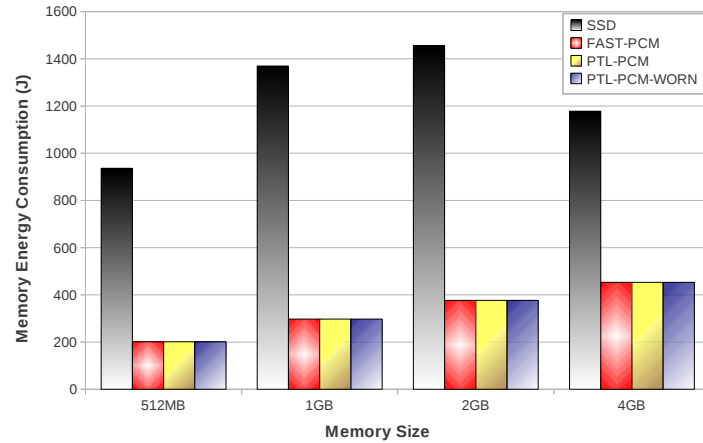


Figure 5.11: Memory Energy TPC-H

5.3.4 Total Energy

The total energy consumption for all the traces reflects the memory energy values, as the memory energy incurred is high. This implies that the memory energy consumed dominates the total energy consumption and adds strength to our goal of reducing the memory size, to achieve the same performance.

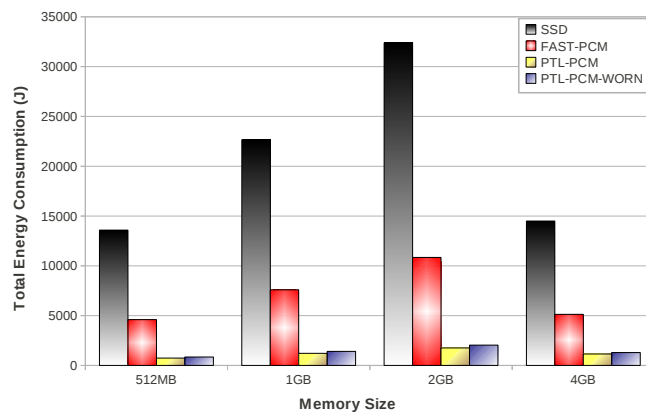


Figure 5.12: Total Energy Postmark

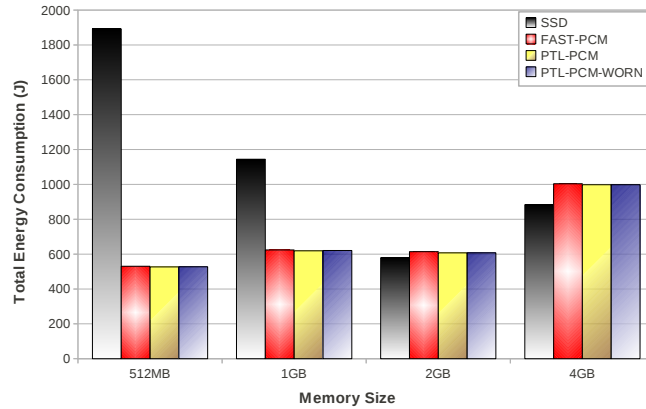


Figure 5.13: Total Energy TPC-C

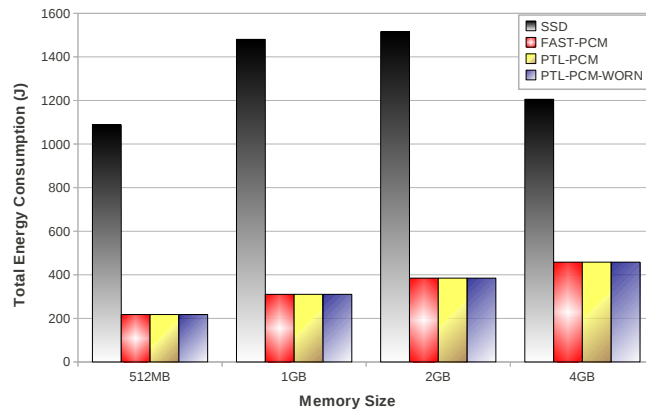


Figure 5.14: Total Energy TPC-H

5.3.5 Energy Delay

The energy delay graphs represent the combined performance and energy values, and gives us a platform to decide the best configuration based on the results obtained. Looking at all the three graphs, we can deduce that using 512 MB with PCM outperforms SSD with 4 GB and HDD with 4 GB configurations. The only exception is in the case of TPC-C, where a 2 GB + SSD performs better than 512 MB + PCM. This is possible, as all our results are based on the best case performance of

SSD/HDD vs worst case performance of PCM.

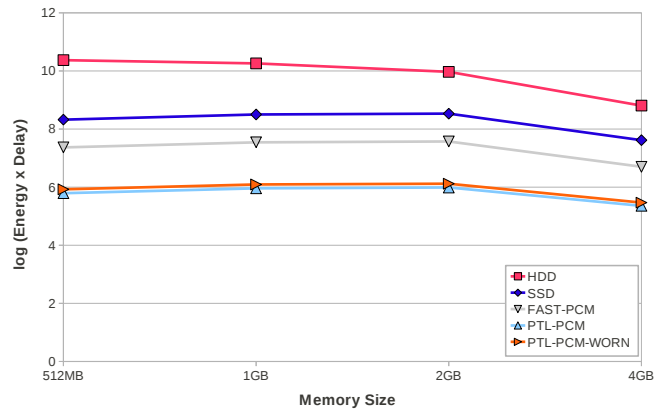


Figure 5.15: Energy Delay - Postmark

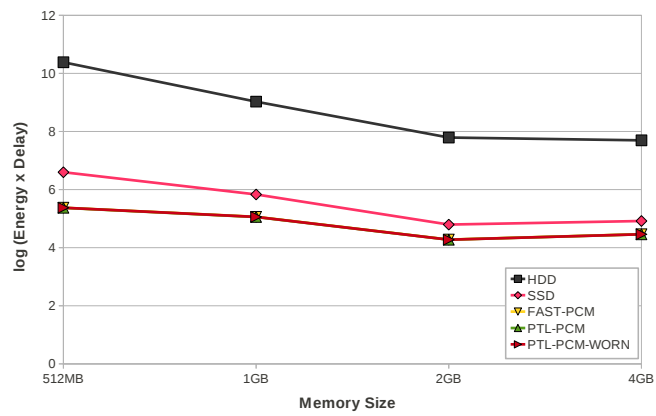


Figure 5.16: Energy Delay - TPC-C

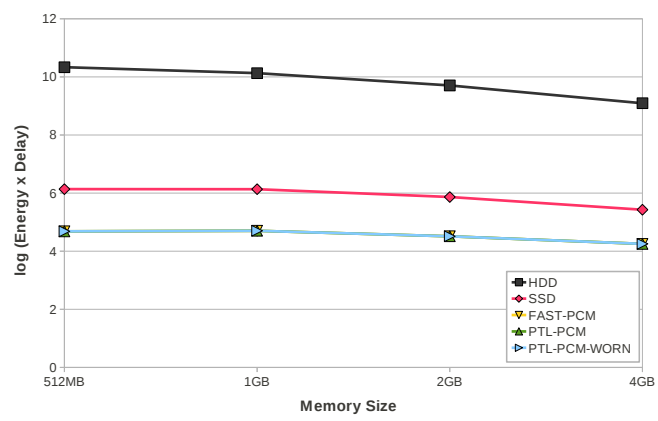


Figure 5.17: Energy Delay - TPC-H

CHAPTER 6

Conclusion and Future Work

6.1 Conclusion

To reduce the energy consumption of a system, we have identified that the bottleneck is the memory energy consumption. To reduce the memory energy consumption by reducing the memory idle time, we use a high performance PCM device which incurs less delays and thus reduces the total memory idle time. We develop a new PTL algorithm for PCM and implement it in our simulator. For different traces we get different energy savings and performance boost. By replacing a 4 GB + SSD with 512 MB + PCM, the performance increases by a factor of 3x and the energy consumption reduces by a factor of 15x. For applications that resemble Postmark, with low hit ratio in the buffer cache, and similar amount of reads and writes this summarizes the gain of using PCM. For TPC-C, which is memory intensive, and also at 2GB the memory requirements saturate, SSD + 2GB performs better by a factor 4x than PCM + 512 MB, even though energy wise the PCM + 512 MB consumes less energy. For TPC-H, PCM + 512 MB gives the same performance as SSD + 4GB, but the energy consumption using PCM + 512MB is reduced by a factor of 6x.

Thus we infer that using a high performing and low consuming device like PCM, we can get away with just using 512 MB of Buffer Cache and achieve similar or better performance than a hard disk or flash based SSD with 4Gb of buffer

cache and reduce the total energy consumption of the system by a huge factor which depends on the specific workload as demonstrated by our experiments with different traces.

6.2 Future Work

The PCM simulator can be augmented with meta data management schemes and even though the delay will be increased only by a small factor, the total delay values can be made more accurate. A bitmap page level wear leveling algorithm should be added with our wear leveling algorithm which is block based, so that a few extreme scenarios like only a few pages in the block being worn out, can be handled better

REFERENCES

- [1] Mingsong Bi, Ran Duan, and C. Gniady. Delay-hiding energy management mechanisms for dram. In *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*, pages 1–10, jan. 2010.
- [2] John S. Bucy, Jiri Schindler, Steven W. Schlosser, Gregory R. Ganger, and Contributors. The disksim simulation environment version 4.0 reference manual. 2008.
- [3] Ali R. Butt, Chris Gniady, and Y. Charlie Hu. The performance impact of kernel prefetching on buffer cache replacement algorithms. *SIGMETRICS Perform. Eval. Rev.*, 33:157–168, June 2005.
- [4] Shimin Chen, Phillip B. Gibbons, and Suman Nath. Rethinking database algorithms for phase change memory. In *CIDR*, pages 21–31, 2011.
- [5] Jeremy Condit, Edmund B. Nightingale, Christopher Frost, Engin Ipek, Benjamin Lee, Doug Burger, and Derrick Coetzee. Better i/o through byte-addressable, persistent memory. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, SOSP '09*, pages 133–146, New York, NY, USA, 2009. ACM.
- [6] Xiangyu Dong, N.P. Jouppi, and Yuan Xie. Pcransim: System-level performance, energy, and area modeling for phase-change ram. In *Computer-Aided Design - Digest of Technical Papers, (ICCAD) 2009. IEEE/ACM International Conference on*, pages 269–275, nov. 2009.
- [7] A.P. Ferreira, Miao Zhou, S. Bock, B. Childers, R. Melhem, and D. Mosse. Increasing pcm main memory lifetime. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pages 914–919, march 2010.
- [8] Aayush Gupta, Youngjae Kim, and Bhuvan Urgaonkar. Dftl: a flash translation layer employing demand-based selective caching of page-level address mappings. *SIGPLAN Not.*, 44:229–240, March 2009.
- [9] Tom’s Hardware. All enterprise hard drive charts 2010. <http://www.tomshardware.com/charts/enterprise-hard-drive-charts-2010/benchmarks,105.html>.

- [10] Intel. Intel x18-m/x25-m sata solid state drive - 34 nm product line.
http://download.intel.com/newsroom/kits/ssd/pdfs/X25-M_34nm_DataSheet.pdf.
- [11] Hyojun Kim and Seongjun Ahn. Bplru: a buffer management scheme for improving random writes in flash storage. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies*, FAST'08, pages 16:1–16:14, Berkeley, CA, USA, 2008. USENIX Association.
- [12] Jesung Kim, Jong Min Kim, S.H. Noh, Sang Lyul Min, and Yookun Cho. A space-efficient flash translation layer for compactflash systems. *Consumer Electronics, IEEE Transactions on*, 48(2):366–375, may 2002.
- [13] Youngjae Kim, Brendan Tauras, Aayush Gupta, and Bhuvan Uргаonkar. Flashsim: A simulator for nand flash-based solid-state drives. In *Proceedings of the 2009 First International Conference on Advances in System Simulation*, pages 125–131, Washington, DC, USA, 2009. IEEE Computer Society.
- [14] Venkataraman Krishnaswami and Venkatasubramanian Viswanathan. Exploring the potential of phase change memories as an alternative to dram technology. *World Academy of Science, Engineering and Technology*, Sep 2010.
- [15] Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. Architecting phase change memory as a scalable dram alternative. *SIGARCH Comput. Archit. News*, 37:2–13, June 2009.
- [16] Sang-Won Lee, Dong-Joo Park, Tae-Sun Chung, Dong-Ho Lee, Sangwon Park, and Ha-Joo Song. A log buffer-based flash translation layer using fully-associative sector translation. *ACM Trans. Embed. Comput. Syst.*, 6, July 2007.
- [17] Sungjin Lee, Dongkun Shin, Young-Jin Kim, and Jihong Kim. Last: locality-aware sector translation for nand flash memory-based storage systems. *SIGOPS Oper. Syst. Rev.*, 42:36–42, October 2008.
- [18] Prasanth Mangalagiri, Karthik Sarpatwari, Aditya Yanamandra, VijayKrishnan Narayanan, Yuan Xie, Mary Jane Irwin, and Osama Awadel Karim. A low-power phase change memory based hybrid cache architecture. In *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, GLSVLSI '08, pages 395–398, New York, NY, USA, 2008. ACM.
- [19] Micron. Comparing 128mb p8p parallel pcm and parallel nor flash memory.
<http://www.micron.com/get-document/?documentId\=5649>.

- [20] Jeffrey C. Mogul, Eduardo Argollo, Mehul Shah, and Paolo Faraboschi. Operating system support for nvm+dram hybrid main memory. In *Proceedings of the 12th conference on Hot topics in operating systems*, HotOS'09, pages 14–14, Berkeley, CA, USA, 2009. USENIX Association.
- [21] Numonyx. http://www.micron.com/get-document/?documentId=5814&file=PhaseChangeMemory_WhitePaper.pdf.
- [22] Moinuddin K. Qureshi, Vijayalakshmi Srinivasan, and Jude A. Rivers. Scalable high performance main memory system using phase-change memory technology. *SIGARCH Comput. Archit. News*, 37:24–33, June 2009.
- [23] Luiz Ramos, Eugene Gorbatov, and Ricardo Bianchini. Page placement in hybrid memory systems. In *Proceedings of the 25th ACM International Conference on Supercomputing*, ICS '11, New York, NY, USA, 2011. ACM.
- [24] G Reinman and N P Jouppi. Cacti 2.0: An integrated cache timing and power model. (2000/7), 2000.
- [25] Seagate. Seagate cheetah 15k.5 fc product manual. <http://www.seagate.com/staticfiles/support/disc/manuals/enterprise/cheetah/15K.5/FC/100384772f.pdf>.
- [26] Arie Tal. Nand vs. nor flash technology. http://www2.electronicproducts.com/NAND_vs_NOR_flash_technology-article-FEBMSY1-feb2002-html.aspx.
- [27] J. Tominaga, T. Kikukawa, M. Takahashi, and R. T. Phillips. Structure of the optical phase change memory alloy, ag-v-in-sb-te, determined by optical spectroscopy and electron diffraction. *Journal of Applied Physics*, 82(7):3214–3218, oct 1997.
- [28] Transaction Processing Council (TPC). Transaction processing performance council (tpc) benchmarks. <http://www.tpc.org/information/benchmarks.asp>.
- [29] Carnegie Mellon University. Dixtrac: Automated disk drive characterization. <http://www.pdl.cmu.edu/Dixtrac/index.html>.
- [30] Noboru Yamada, Eiji Ohno, Kenichi Nishiuchi, Nobuo Akahira, and Masatoshi Takao. Rapid-phase transitions of gete-sb₂te₃ pseudobinary amorphous thin films for an optical disk memory. *Journal of Applied Physics*, 69(5):2849–2856, 1991.
- [31] Ping Zhou, Bo Zhao, Jun Yang, and Youtao Zhang. A durable and energy efficient main memory using phase change memory technology. *SIGARCH*

Comput. Archit. News, 37:14–23, June 2009.