Adaptive Page Grouping for Energy Efficiency in Hybrid PRAM-DRAM Main Memory

Dong-Jae Shin, Sung Kyu Park, Seong Min Kim, and Kyu Ho Park
Korea Advanced Institute of Science and Technology (KAIST)
305-701, Guseong-dong, Yuseong-gu, Daejeon, Korea
{djshin, skpark, smkim}@core.kaist.ac.kr, and kpark@ee.kaist.ac.kr

ABSTRACT

The recent technology faces the challenges to reduce energy consumption of DRAM, which consumes about 30% of total energy in data centers. Phase change RAM (PRAM), not requiring the charge current because of its non-volatility, has appeared for replacing DRAM. However, it has some disadvantages which are its low performance, high write power, and write endurance limitation compared to DRAM. To overcome these weak points of PRAM, the research related to the hybrid model combining PRAM with DRAM has been progressed. Unfortunately, previous works on hybrid memory is hard to be applied to the market because established hardware needs to be changed.

In this paper, we propose an Adaptive Page Grouping (APG) algorithm, which manages the hybrid PRAM-DRAM main memory for reducing energy consumption. We suggest the method to store the access information of pages without using additional space and make operating system (OS) can access it. We allocate pages effectively and reduce the migration among them through the grouping of pages which has similar access properties. We can apply our system immediately as a software patch when PRAM is released, because all these schemes are implemented in Linux OS without additional hardware. Thus, we have decreased average energy consumption by 36%, with maximum up to 42%, compared to a DRAM system while access time increases less than 8% including high latency of PRAM.

Categories and Subject Descriptors

D.4.2 [**Operating Systems**]: Storage Management; D.3.2 [**Memory Structure**]: Design Styles

General Terms

Algorithm, Management, Design

Keywords

Main memory, Power management, PRAM, PCM

RACS'12 October 23-26, 2012, San Antonio, TX, USA.

1. INTRODUCTION

As computer systems are widely used everywhere, energy consumption of computer is increasing. Data centers of IT enterprises such as cloud computing center possess the 1.5% of world's total energy consumption and it still doubles every five years [1]. Especially, dynamic random access memory (DRAM) has large portion of consumed energy. In Google data centers, about 30% of energy is consumed by DRAM main memory . [2]. The reason is that DRAM always spends the energy to keep stored data even when it does not work because of its volatile characteristic.

In recent years, Phase Change RAM (PRAM) technology has been researched to replace DRAM. PRAM is one of the promising candidates for future main memory because it is a byte-addressable and non-volatile memory. PRAM does not consume energy when it is in idle state because of its non-volatility as against DRAM. It is also believed that PRAM is more scalable than DRAM with multi level cell (MLC) capability [3] [4]. PRAM is on the verge to be massively produced; Samsung electronics has announced a prototype of 8Gb PRAM, larger than other non-volatile and byte-addressable memories [5]. However, PRAM has write limit of 10^8 times, high write latency (5-10 times higher than DRAM), and high write energy [6]. Because of these problems, it is difficult to use an only PRAM main memory.

Therefore, researchers consider a hybrid main memory architecture with DRAM and PRAM. However, previous works on hybrid memory is hard to be applied to the market because established hardware needs to be changed. To overcome this problem, power-aware management [7] proposed the main memory management mechanism of the operating system for the hybrid main memory, which combines both advantages of DRAM and PRAM. They suggest a migration scheme using access bits and second chance algorithm in page granularity between DRAM and PRAM. Unfortunately, some pages suffer from ping-pong migration, which means endless migrations every 2 cycles from DRAM to PRAM and from PRAM to DRAM in several workloads. Ping-pong migration is caused by limited access information and second chance algorithm. They only use 1 bit information to indicate whether page has been accessed or not, but it is not enough to represent the characteristics of pages. When second chance algorithm is combined with this limited information and a workload which has repetitive access patterns, then ping-pong migration occurs. When a page has a period of access, it is decided as hot page in access phase and as cold page in non-access phase repetitively. This patterns usually appears in sequential access pattern. In these

workloads, such as gzip benchmark, migration energy has more portion of energy than dynamic energy (read and write energy) because of ping-pong migration. These migrations increase not only power consumption but also access time.

In this paper, we propose a memory management technique using Adaptive Page Grouping (APG) which increases energy efficiency and maintains the performance degradation to the minimum. We can decrease overall migration counts and ping-pong migration using adaptive granularity of migration with stored access information. Increased granularity makes heavy migration of memory pages when migration policy is based on whether to be accessed or not. However, increased granularity would have large loss of energy if memory pages are misplaced. To prevent misprediction, we use the group of pages which have similar access count. Adaptive granularity is decided by physical distance of pages using grouping technique.

Our contribution is as follows.

- We designed and implemented APG system which is energy efficient while its performance degradation is minimized. We get 36% of energy reduction compared to only DRAM memory, 8% compared to the previous work for the various workload averagely.
- We prevent useless migration to the minimum and reduce write operation on PRAM effectively using access information with no space overhead.
- Proposed system is able to support DRAM-PRAM main memory by software patch without additional hardware, thereby implementing full software approach in Linux OS.

The rest of paper is organized as follows. Section 2 describes background and related work including hybrid main memory architecture; Section 3 details proposed system and its hybrid main memory management operating techniques; the results of evaluation and discussion is shown in Section 4; Section 5 concludes this thesis and section 6 describes our future work.

2. BACKGROUND AND RELATED WORK

2.1 Hybrid Main Memory Architecture

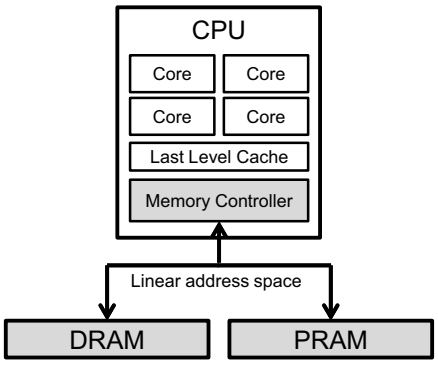


Figure 1: Hybrid Main Memory Architecture

One of hybrid architecture is a hierarchical architecture to alleviate disadvantages of PRAM [8]. This architecture is composed of a small amount of DRAM memory as a buffer, and large sized PRAM as main memory to exploit short latency of DRAM and low idle energy of PRAM. These are tightly coupled with H/W logic between DRAM and PRAM memory devices, while the memory controller manages it. There is a controller to manage this architecture, that checks whether data is in DRAM or not at every request similar to cache system of modern processor. If data is in DRAM, then it is read from the buffer. However, it is read and transferred from PRAM to DRAM when data is not in DRAM. When write request is received, the controller checks dirty bits and data is written to DRAM or both of memories. If DRAM is full, then eviction algorithm is executed to free space.

Figure 1 shows our target architecture. This hybrid main memory architecture was proposed in [9]. Both of PRAM and DRAM memory devices are directly connected to memory controller in parallel with each other. In this architecture, DRAM and PRAM are assigned to a single physical address space. It has several advantages compared to the hierarchical architecture. Firstly, the total capacity is the sum of both memories, indeed size of DRAM is shown to user because DRAM is not a buffer any more and then both of memories operate as a main memory respectively. Secondly, memory operation is a simple access, without having write-backs and data passing between DRAM and PRAM. This causes the performance improvement for workloads which has poor locality.

However, elaborate management is needed for this architecture. If write-intensive pages are allocated in PRAM, this system consumes a lot of write energy due to its characteristics. In addition to this problem, many write operations wear out the cell of PRAM rapidly. Therefore, placement of page is a challenging issue in this architecture.

2.2 Related Work

Several researches have proposed in the hierarchical architecture. Qureshi et. al. [8] proposed this architecture to exploit both benefits of DRAM and PRAM. In this architecture Park et. al. [10] suggested a power saving technique reducing the refresh energy of DRAM. There was several works to enhance lifetime of PRAM caused by endurance limitation [11] [12] [13]. However, this architecture has several disadvantages due to its own property. Firstly, the capacity of DRAM is not shown to user because DRAM is used only as a buffer. Moreover, data is duplicated in both memories due to its consistency. Secondly, it degrades performance significantly for workloads which has poor locality [14]. If a workload has poor locality, memory operation performs 2 steps of access and many writeback operations with high probability; then its performance degrades.

PDRAM [9] proposed a parallel architecture based on Phase Change RAM (PRAM) and DRAM. The paper explores the challenges involved in incorporating PRAM into the main memory hierarchy of computing systems, and proposes a low overhead hybrid hardware-software solution to manage it. This paper introduces a wear-leveling scheme using additional free list and achieve equational wear-level in page granularity. However, it is hard to apply this approach to industry because the memory controller, located in the processor, has to be changed. According to our experiments, PRAM pages consumes substantial dynamic energy until write count reach the threshold; these are the problems to solve.

Rank-based Page Placement (RaPP) [14] is proposed as a memory management mechanism designed for hybrid main memory, using rank-based migration techniques to provide lower energy-delay^2. It also requires architectural change of hardware to provide rank decision policy. Since our management system aims to apply it as full software approach, we did not compare our result to the RaPP.

Park et. al. [7] proposed the main memory management mechanism of the operating system for the hybrid main memory which combines both advantages of DRAM and PRAM. They proposed the migration scheme using access bits and power-off technique of DRAM chunk. However, there is a problem that makes some pages endure endless migrations from DRAM to PRAM and from PRAM to DRAM repeatedly. We denote this problem with ping-pong migration. Ping-pong migration of pages is caused by limited access information and second chance algorithm. Only 1 bit information is not enough to decide the hotness and coldness of page. When second chance algorithm is combined with this limited information and a workload which has repetitive access patterns, then ping-pong migration occurs. If a page has a period of access, it is decided as hot page in access phase and as cold page in non-access phase repetitively. Because of ping-pong migration, migration energy is lager than dynamic energy (read and write energy) in some workload, such as gzip benchmark. Both migration energy and access time are increased by these migrations. Moreover, they obtained reduction of energy using power-off mode of DRAM, but this technique is needed for the algorithm to be supported by hardware.

We solve these problems, migration energy and hardware change, using Adaptive Page Grouping system implemented by only software approach. We use storing write history technique to get more information, and we use grouped migration to reduce the number of total migration.

3. ADAPTIVE PAGE GROUPING SYSTEM

Figure 2 shows the overall architecture of system we proposed and it consists of three modules which are monitoring & shifting module, page grouping module and migration module. These modules are implemented as the form of kernel daemon and carry out the tasks periodically.

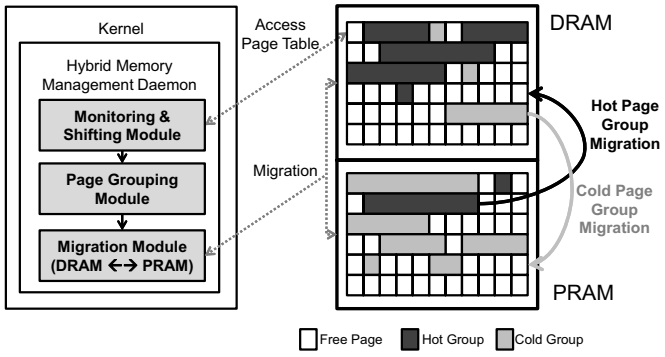


Figure 2: Overall architecture of APG System for hybrid PRAM-DRAM main memory

3.1 Storing Write History

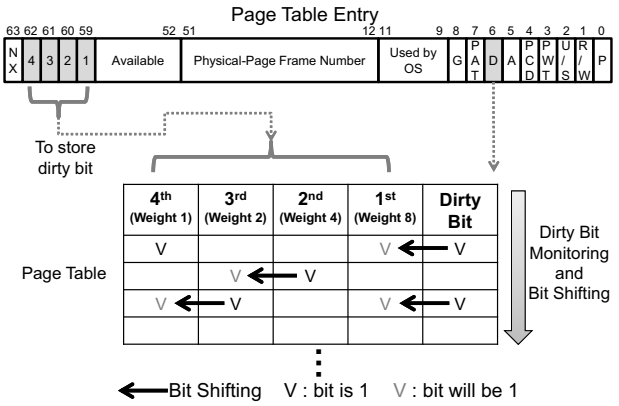


Figure 3: Page Table Monitoring and Bit Shifting Recent bit has more weight to exploit temporal locality.

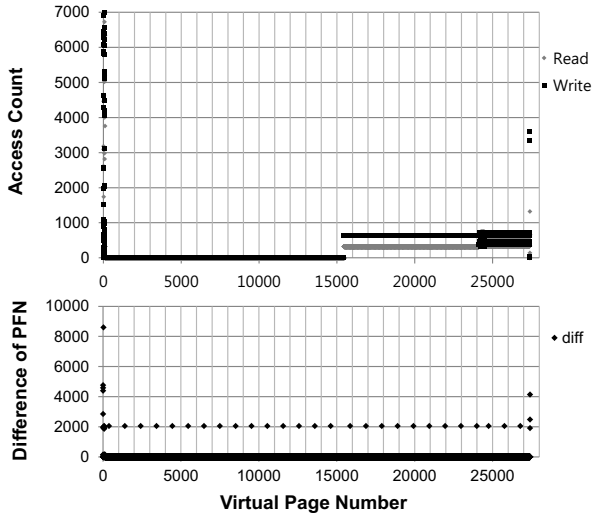
In hybrid architecture, it is important to decide the location of page due to their different characteristics. Essentially, hot pages should be in DRAM because of its lower latency and write energy. To distinguish the hot pages and cold pages accurately, we have to judge from the saved access information of pages. We use dirty bit checked by H/W in page table entry (PTE). From the experiment, we show that deciding the properties of pages by using dirty bit is efficient rather than using access bit because of critical write property of PRAM. Unfortunately, decision from the information of one bit is not accurate due to lack of information.

We propose the bit shifting method to store access information while other works use a portion of PRAM or SRAM cache to store access information. We use unused bits of page table entry without additional storage overhead. Figure 3 shows how the bit shifting operates. To store access information, we use available 59–62 bits in a PTE. The 6th bits, denoted with 'D', means dirty bit which is changed to '1' by processor when a write operation occurs. If dirty bit is 1, it shifts to the first location. Similarly, in the case of 0, it shifts to the first location. Through this process, recently written histories are saved in the pages and this information can be used to determine whether the pages are hot or cold.

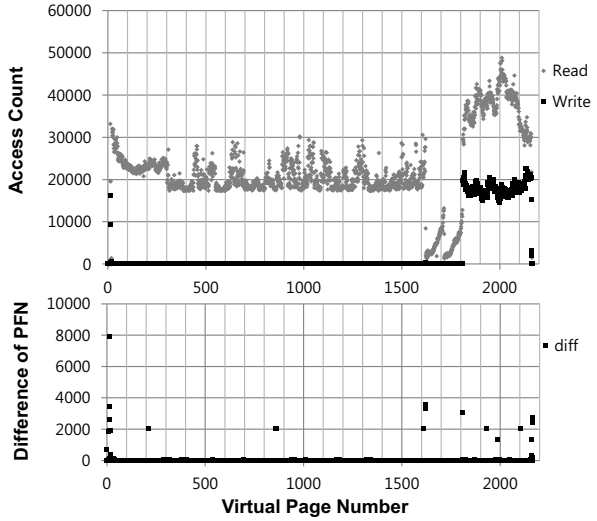
3.2 Adaptive Page Grouping

We proposed page grouping due to the following reasons. Firstly, it prevents from individual pages ping-pong migrations by grouping pages with similar properties. Ping-pong migration of pages is caused by repetitive access pattern. Thus increased granularity reduces the number of migration. Therefore, with page grouping, ping-pong migration can be solved by preventing migration of pages in group separately. Secondly, it is possible to compute fine-grained hot/coldness in case of grouped compared to the policy which is based on individual page. We can get more specific values through observing the stored dirty bit of grouped pages rather than page granularity because it has few bits of information.

Grouping process is based on the physical distance of memory pages. In our experiments (Figure 4), we confirmed that physically adjacent pages have similar accessed count. The difference of page frame number (PFN) means a physical distance between memory pages ordered by virtual address in page table. Figure 4 shows that most of are physically close and a rapid change of access counts occurs on the



(a) gzip benchmark of SPEC CPU 2000

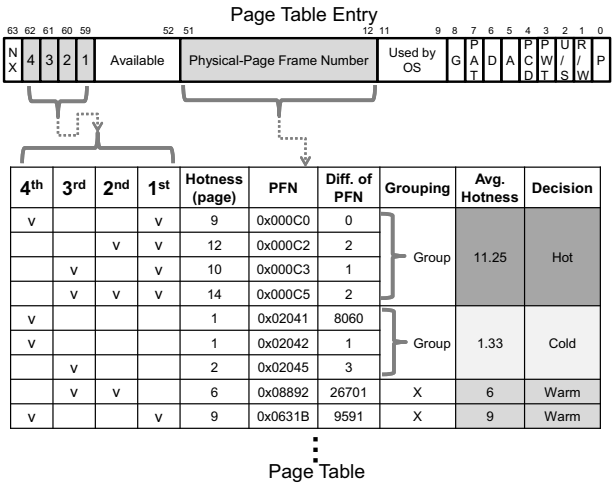


(b) mcf benchmark of SPEC CPU 2000

Figure 4: Correlation between access count and physical distance. Physically adjacent pages have similar accessed count.

point which have large difference of PFN. This comes from the buddy system which is the memory management method in Linux OS. If possible buddy system allocates continuous blocks to prevent memory fragmentation. The memory region, allocated by a user, has a own purpose using a allocation function such as *malloc*. For instance, there exist some regions for using buffer, which has a lot of writes, and others for data region, which has a lot of reads. According to this, physically continuous pages have similar access counts. Therefore, it is possible to manage the pages which have similar properties by using this tendency.

Figure 5 shows the Adaptive Page Grouping algorithm. To measure the distance, the difference between current and previous page frame numbers in page table entry is used. This is because the difference of PFN means the difference of physically allocated addresses. If this value does not exceed threshold value, they are considered as the same group.



4th	3rd	2nd	1st	Hotness (page)	PFN	Diff. of PFN	Grouping	Avg. Hotness	Decision
v			v	9	0x000C0	0	Group	11.25	Hot
		v	v	12	0x000C2	2			
	v		v	10	0x000C3	1			
	v	v	v	14	0x000C5	2			
v				1	0x02041	8060	Group	1.33	Cold
v				1	0x02042	1			
	v			2	0x02045	3			
	v	v		6	0x08892	26701	X	6	Warm
v			v	9	0x0631B	9591	X	9	Warm

Figure 5: Adaptive Page Grouping and Hot/Cold Separation. Group is based on physical distance because physically near pages have similar access counts.

This grouping is performed differently each time on the periodically working system. Even though there is some changes like memory allocation and deallocation, the certain grouping is executed at next turn. The minimum size of group is 1 (same as page level) and the maximum size is 512, because it is the maximum page table entries for a page middle directory. Using this Adaptive Page Grouping, it is possible to migrate grouped pages which has similar properties. We use the threshold value 1000 as a good indicator which divides the group to have similar access characteristics as shown in Figure 4.

Figure 5 also shows the mechanism to decide the hotness of page groups. In a page table entry, it has a write history which uses the dirty bit shifting method explained before. The first bit is recent bit and fourth bit is the oldest. We set weights for these bits to exploit temporal locality. The first bit codes for a value of 8, the second bit for 4, the third one for 2, and fourth one for 1. The total summation of above values are the page hotness. Hotness of the group is expressed as the average hotness of included pages and hot/cold decision is determined by it. If hotness of the group is higher than hot_threshold, it is hot and if it is smaller than the cold_threshold, it is cold. If it is not included in both cases, these groups become warm group. We use the hot_threshold value as 12, cold_threshold value as 2 which are good trade-off between accuracy and migration overhead. In our evaluation, the variation of threshold value does not affect the energy consumption widely because hotness of a group is distributed extremely.

3.3 Migration policy

The total system performance and energy consumption are influenced by the location of the classified group of pages. According to characteristics of PRAM and DRAM, locating hot pages in DRAM and cold pages in PRAM is suitable especially for the case of write operations. For example, if a page group which has many write operations is allocated in PRAM, it causes lots of energy consumption and performance loss. Even though hot group is allocated in DRAM and cold group is allocated in PRAM successively, total en-

ergy consumption and performance cannot be improved if there are a lot of migrations. For the effective placement, we establish the placement policies as follows.

- **Hot groups are allocated to DRAM, while cold groups are migrated to PRAM** Hot groups, which have many write operations, need to be allocated on DRAM in terms of both energy and performance. Cold pages, however, is better to be allocated on PRAM. Cold pages have a little difference on energy and performance whether PRAM or DRAM. Therefore, it is efficient to allocate cold pages on PRAM due to the limitations of spaces in DRAM.
- **Warm groups do not migrate** From the experiment, we conclude that it is better not to migrate groups which are neither hot nor cold until the states of each group are recognized. If we migrate the warm pages, it can increase the migration energy with mispredictions.
- **The first allocated memory region is DRAM.** Firstly allocated page has a high probability for immediate use. This is the main difference from PDRAM [9] and the experiment shows that this policy has an energy gain. Migrating cold groups to PRAM is needed to procure the space of DRAM when using this policy.
- **Kernel memory region is allocated to DRAM.** It has to be allocated to DRAM since it is highly dependent on the performance. Therefore, we suppose that user process can only use PRAM.

4. EVALUATION

In this section, we evaluate the energy and performance of Adaptive Page Grouping (APG) system. First, this part explains the experimental environment and detailed implementation. Then, we show the result of power saving and performance of the experiments and discuss its reason. We select the comparable system as PDRAM [9] and Power-aware Management [7] (refers second chance algorithm).

4.1 Environment

Table 1 summarizes our experimental environment including both hardware and software. We implement the APG system by only applying software on the Linux operating system, and we used commercialized hardware for the case of processor and memory. For PRAM, however, there is no DDR3 product yet; to alternate this, we use DRAM instead and compensate its latency in the result. To calculate the power consumption and performance, counting information based on pintool [15] is used.

These calculations are comprised of cited memory characteristics. Table 2 shows the power consumption and latency information of both DRAM and PRAM. Based on this information, we estimate the power consumption and access frequency of DRAM and PRAM for each workload. When memory controller requests the access, it consumes the active power while selecting certain row of memory device. After this, row read power or row write power is consumed depending on the operation. Standby power and refresh power is the constantly consumed power in the memory module while the computer is turned on. Note that row write power of PRAM is evidently higher than others. Because of this,

Feature	Value
Processor	Intel Core2 CPU E6600 2.4 GHz 32kB, 64B line, 8-way, L1 cache 4MB, 64B line, 16-way, L2 cache
Memory	DDR3 1066MHz DRAM 1GB PRAM 1GB or 4GB
Operating System	Linux 64bit 2.6.31 kernel

Table 1: Experimental Specification

Power characteristics		
Parameter	**DRAM**	**PRAM**
Row Read Power	210mW	78mW
Row Write Power	195mW	773mW
Active Power	75mW	25mW
Standby Power	90mW	45mW
Refresh Power	4mW	0mW
Timing characteristics		
Initial row read	15ns	28ns
Same row read	15ns	15ns
Initial Row write	22ns	150ns
Same row write	15ns	15ns

Table 2: Memory Characteristics [9]

it is energy-efficient for write request to be accomplished on DRAM. Since both DRAM and PRAM have a row buffer, they operate differently depending on the cases: same row and initial row. For the case of same row, read and write operations are fast irrespectively of memory because row buffer is made of fast circuit. However, the operation for initial row in PRAM is slower than that of DRAM and it is remarkable for the write operation.

We select the workload to evaluate the performance as SPEC CPU 2000 benchmark [16] and Matching as a Service application (Maas) [17] which requires a large scale of memory. SPEC CPU 2000 is also used in [9] and power-aware management [7] respectively. For all SPEC benchmarks (mcf, gzip, applu, gcc, equake, ammp, mgrid), we evaluate the values by executing the reference set.

4.2 Implementation

Memory layout is composed with 1GB DRAM and 1GB PRAM or 1GB DRAM and 4GB PRAM. And we use 4KB of page size which is the default size of Linux. These memories are located in a physically linear address space. In this composition, we use virtual non-uniform memory access (NUMA) of Linux to distinguish between DRAM and PRAM. Virtual NUMA is able to be set in the boot-loader using parameters. Therefore, it is able to discriminate DRAM and PRAM from the OS which deals with memory since DRAM and PRAM have different node id.

We implement the APG system as the form of kernel daemon. This daemon frequently wakes up and does monitoring, bit-shifting, page grouping, and migration. After that, it waits until the next period. We use a period of 1 second which is good trade-off between accuracy and migration

overhead.

To handle the page table entries (PTEs), we use the kernel functions of Linux such as *pte_dirty* and *set_pte_at*. In addition to the these functions, we implement several functions to deal with unused bits in PTEs. These macro functions are implemented in *pgtable_types.h* with definition of bits. The kernel function *migrate_pages* is used for migration with node id which indicates that page is in DRAM or PRAM.

4.3 Experimental Results

4.3.1 Comparable System

All evaluations are executed by following settings.

- 2GB of Total Memory
  - **DRAM Only** : 2GB DRAM memory
  - **PRAM Only** : 2GB PRAM memory
  - **PDRAM** [9] : 1GB DRAM and 1GB PRAM with PDRAM system. When write counts on PRAM pages exceed the threshold, migrate pages to DRAM. We use threshold value as 1000 as mentioned in the paper.
  - **S.chance** [7] : 1GB DRAM and 1GB PRAM with second chance algorithm. A page with two consecutive memory references is migrated to DRAM while a page with two consecutive non-reference is migrated to PRAM.
  - **APG** : 1GB DRAM and 1GB PRAM with Adaptive Page Grouping explained in previous sections.
- 5GB of Total Memory
  - **DRAM Only** : 5GB DRAM memory
  - **PRAM Only** : 5GB PRAM memory
  - **PDRAM**, **S.chance**, **APG** : Same policy as above with 1GB DRAM and 4GB PRAM

PDRAM [9] used 1GB of DRAM and 3GB of PRAM, while power-aware management [7] used both 1GB of DRAM and PRAM. The setting, we stated above, covers the range of these configuration. Thus, we set the memory size to 2GB and 5GB for fairly comparing with related works.

4.3.2 Energy

Figure 6 refers energy consumption of the memory system. It includes static power (standby power, refresh power), dynamic power (active power, read power, write power), and migration power. PRAM has lower static power but high write power and write latency, as we show in Table 2. This is the reason why PRAM only system spends more energy than DRAM only system, in the cases of mcf, applu and mgrid. Because workloads which have high write operations per instructions spend more time for writing, this additional time cancels out profit of low static power. For 5GB hybrid memory system, which is comprised of 1GB of DRAM and 4GB of PRAM, saves more energy than 2GB memory system (1GB of DRAM and 1GB of PRAM). It is because that low static power of PRAM take up the more portion of total energy consumption.

For the case of mcf, there are a lot of memory requests from the memory controller because it has low locality. Therefore, energy consumption of only PRAM system has more

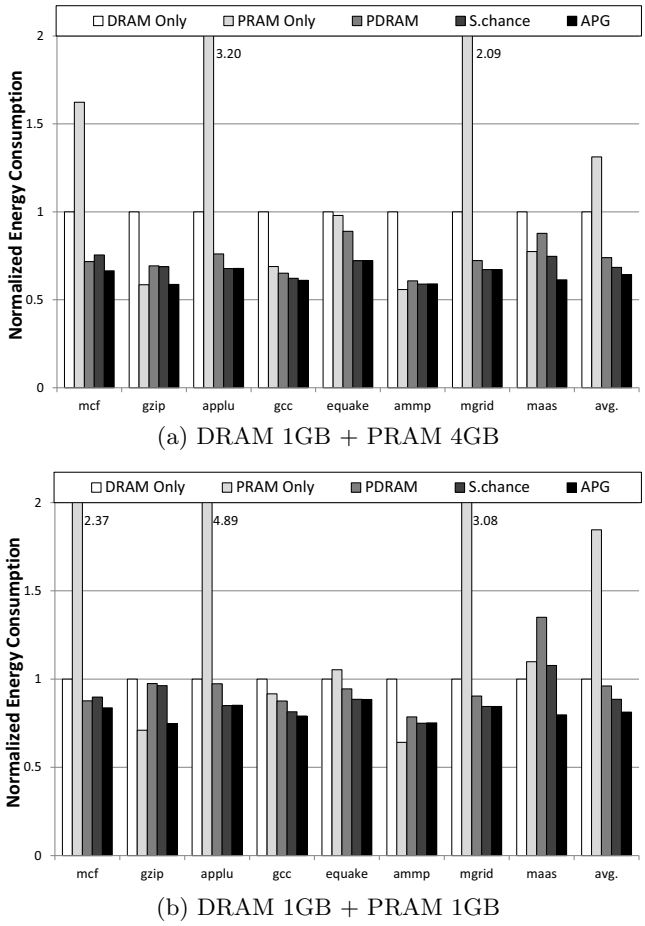
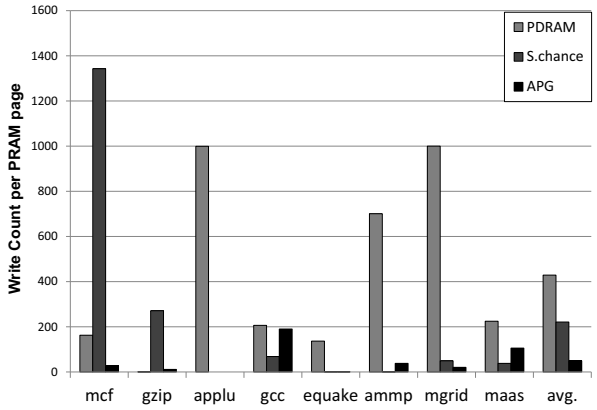
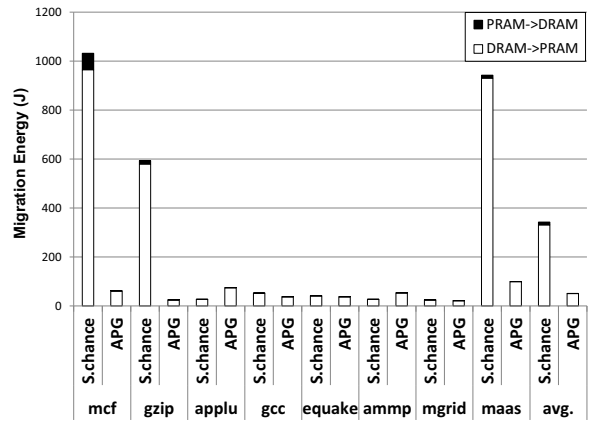


Figure 6: Normalized energy consumption

than twice energy consumption compared to only DRAM system due to large increment in dynamic energy. The case that uses second chance algorithm has inefficient energy performance because the number of writes on PRAM is larger than that of PDRAM and APG. For gzip, it has a few memory requests due to its high locality. Therefore only PRAM system has the smallest energy consumption because of its low static power. Since gzip has a lot of sequential access pattern which means that it is the workload frequently exhibits ping-pong problem, while APG has a little energy consumption. For the case of applu, it has a lot of write operations while it also has many memory requests. Therefore, only PRAM system has a huge energy consumption while s.chance and APG has a small energy consumption. On the other hand, gcc has a few write operation ratio and memory requests. The results shows that APG system has the smallest energy consumption because it allocates the page which will be written in DRAM very often. For equake, it has lots of read operations and relatively little write operations. PDRAM, which allocates the pages on PRAM until the number of write operations per page is 1000, has higher energy consumption compared to the other two systems. Since ammp has a small memory requests and write operation ratio similar to gcc, only PRAM system offers the best energy saving because the gain from low static energy is larger than other systems. The case that uses mgrid has



(a) Reduction of write count per PRAM page



(b) Reduction of Migration Energy

Figure 7: The reason for saving energy. Reduction of write count in PRAM and reduced number of migration save the energy.

a lot of memory requests but small write operation ratio. This is because three systems distinguish the hot pages and cold pages clearly, every cases saved energy consumption. Finally, Maas has small memory accesses while it uses large scale of memory system. For the second chance algorithm, it needs a lot of migration energy with frequent ping-pong problem because access bits are checked even few usages of page. PDRAM system shows that it saves energy where most of pages are allocated in PRAM while APG system saves energy by allocating DRAM hot groups only.

We get 36% and 19% average energy reduction compared to only DRAM memory in 2GB memory and 5GB memory respectively, which is 8% of reduction compared to previous work. We analyze above energy saving results into two main reasons. First, we lowered the write count in PRAM. As the write count becomes larger, large write energy of PRAM causes bad influence on the total energy consumption significantly. Also, we decrease the migration energy by reducing the number of migrations through Adaptive Page Grouping.

In Figure 7(a), write count per PRAM page is declined mostly in APG on average. Since PDRAM writes on PRAM until the threshold, it usually has more write counts. However, APG decreases the number of writes on PRAM by detailed hot/cold separation compared to s.chance algorithm.

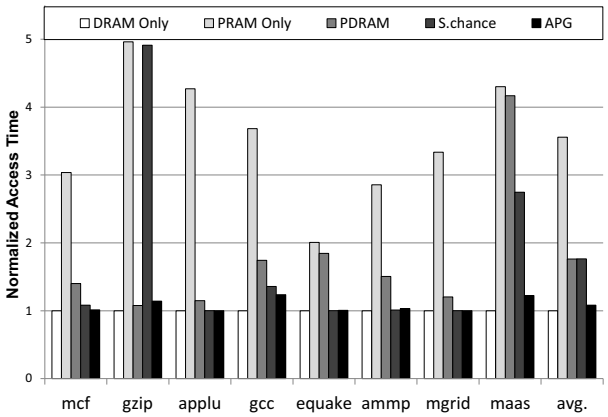


Figure 8: Normalized access time

Workload	Overhead (% of time)
mcf	0.79
gzip	0.52
applu	0.41
gcc	0.88
equake	0.32
ammp	0.15
mgrid	0.23
maas	1.14
avg.	0.55

Table 3: Overhead

Figure 7(b) refers the migration energy comparison between s.chance and APG. It shows that total migration energy including ping-pong migration is declined.

4.3.3 Performance

Figure 8 refers the memory access time due to the read, write, and migration operation. The most impactive variable which determines the performance of memory system is the number of write operations on PRAM. Only DRAM system has the smallest execution time while only PRAM system has the largest execution time. In case of gzip using s.chance algorithm, the reason why it has long spend time is because of ping-pong migration of second chance algorithm. APG system has a latency delay incremented by 8% compared to only DRAM system on average. This is a value decreased by 38% compared to PDRAM or second chance algorithm.

4.3.4 Overhead

Table 3 shows the overhead of Adaptive Page Grouping(APG) system. The overheads contain the spent time for page table monitoring, big shifting, grouping, and migration. The portion of daemon overhead for the total execution time is 1.14% maximum which is small enough. In our experiments, most of overheads occurs not grouping but from migrating pages. Therefore, reducing the number of migration is the major factor to obtain low overhead. We show the reduction of migration energy of APG System in Figure 7(b).

5. CONCLUSION & FURTHER WORK

We designed and implemented Adaptive Page Grouping system which is energy efficient while minimizing its performance degradation. Dirty bit shifting is used to store access information without additional storage. We use weighted dirty bit value to exploit temporal locality. APG system gathers physically near pages to exploit spatial locality over pages. An it prevents useless migration to its minimum and reduces write operation on PRAM effectively. Besides, all of this work are implemented with fully software approach in Linux OS. The experiments show that we can get 36% energy saving compared to only DRAM system and 8% of saving compared to previous power-aware memory management [7] with less than 8% of access delay which includes low latency of PRAM.

We will examine various wear leveling schemes related to write endurance, which is one of the fatal problem that PRAM faced. The lifetime of PRAM will increase while using same wear leveling algorithm because we reduce the write count on PRAM. It is possible to apply our work to the mobile system such as android OS, where also power consumption is a significant factor.

6. ACKNOWLEDGMENTS

The work presented in this paper was supported by MKE (Ministry of Knowledge Economy, Republic of Korea), Project No. 10035231-2010-01.

7. REFERENCES

[1] S. Murugesan, "Harnessing green it: Principles and practices," *IT Professional*, vol. 10, pp. 24–33, Jan. 2008.

[2] U. Hoelzle and L. A. Barroso, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st ed., 2009.

[3] F. Bedeschi, R. Fackenthal, C. Resta, E. Donze, M. Jagasivamani, E. Buda, F. Pellizzer, D. Chow, A. Cabrini, G. Calvi, R. Faravelli, A. Fantini, G. Torelli, D. Mills, R. Gastaldi, and G. Casagrande, "A multi-level-cell bipolar-selected phase-change memory," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pp. 428 –625, Feb. 2008.

[4] N. Papandreou, H. Pozidis, T. Mittelholzer, G. Close, M. Breitwisch, C. Lam, and E. Eleftheriou, "Drift-tolerant multilevel phase-change memory," in *Memory Workshop (IMW), 2011 3rd IEEE International*, pp. 1–4, May 2011.

[5] Y. Choi, I. Song, M.-H. Park, H. Chung, S. Chang, B. Cho, J. Kim, Y. Oh, D. Kwon, J. Sunwoo, J. Shin, Y. Rho, C. Lee, M. G. Kang, J. Lee, Y. Kwon, S. Kim, J. Kim, Y.-J. Lee, Q. Wang, S. Cha, S. Ahn, H. Horii, J. Lee, K. Kim, H. Joo, K. Lee, Y.-T. Lee, J. Yoo, and G. Jeong, "A 20nm 1.8v 8gb pram with 40mb/s program bandwidth," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pp. 46–48, Feb. 2012.

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

[7] Y. Park, D.-J. Shin, S. K. Park, and K. H. Park, "Power-aware memory management for hybrid main memory," in *Next Generation Information Technology (ICNIT), 2011 The 2nd International Conference on*, pp. 82–85, June 2011.

[8] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, "Scalable high performance main memory system using phase-change memory technology," in *Proceedings of the 36th annual international symposium on Computer architecture*, ISCA '09, pp. 24–33, ACM, 2009.

[9] G. Dhiman, R. Ayoub, and T. Rosing, "Pdram: A hybrid pram and dram main memory system," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, pp. 664–669, July 2009.

[10] H. Park, S. Yoo, and S. Lee, "Power management of hybrid dram/pram-based main memory," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pp. 59 –64, June 2011.

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

[12] A. Ferreira, M. 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*, pp. 914 –919, March 2010.

[13] S. K. Park, H. Seok, D.-J. Shin, and K. H. Park, "Pram wear-leveling for hybrid main memory based on data buffering, swapping, and shifting," in *Proceedings of the 2012 ACM Symposium on Applied Computing*, SAC '12, pp. 1643–1644, ACM, 2012.

[14] L. E. Ramos, E. Gorbatov, and R. Bianchini, "Page placement in hybrid memory systems," in *Proceedings of the international conference on Supercomputing*, ICS '11, pp. 85–95, ACM, 2011.

[15] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation," in *Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '05, pp. 190–200, ACM, 2005.

[16] J. Henning, "Spec cpu2000: measuring cpu performance in the new millennium," *Computer*, vol. 33, pp. 28 –35, July 2000.

[17] J. H. Choi, K. W. Park, S. K. Park, and K. H. Park, "Multimedia matching as a service: Technical challenges and blueprints," in *Proceedings of the 26th international technical conference on circuits/systems, computers and communications*, ITC-CSCC'11, pp. 632–635, 2011.