



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Master's Thesis

A Study on the Effect of Access Patterns in Tiered Memory System

JungBeen Kim

Department of Computer Science and Engineering

Ulsan National Institute of Science and Technology

2022

A Study on the Effect of Access Patterns in Tiered Memory System

JungBeen Kim

Department of Computer Science and Engineering

Ulsan National Institute of Science and Technology


A Study on the Effect of Access Patterns in Tiered Memory System

A thesis/dissertation submitted to
Ulsan National Institute of Science and Technology
in partial fulfillment of the
requirements for the degree of
Master of Science

Jungbeen Kim

07.07.2222 of submission

Approved by

A handwritten signature in black ink, appearing to be 'S. H. Noh', is written over a horizontal line. The signature is stylized and includes a large flourish on the left side.

Advisor

Sam H. Noh


A Study on the Effect of Access Patterns in Tiered Memory System

Jungbeen Kim

This certifies that the thesis/dissertation of Jungbeen Kim is approved.

07.07.2022 of submission

Signature



Advisor: Sam H. Noh

Signature



Woongki Baek of Thesis Committee member #1

Signature



Myeongjae Jeon of Thesis Committee member #2

Abstract

High-capacity non-volatile memory is the new main memory. NVM provides up to 8x the memory capacity of DRAM, but can reduce bandwidth by up to 7x and increase latency by up to 2x. In case of using NVM alone, it provides a large capacity but has the disadvantage of low performance, so a system that is used with DRAM is used. However, if the two memories are not managed properly, the performance will be as bad as if NVM is used alone. A lot of optimization work is being done in the most studied tiered memory system to use the two memories. We found that before Intel Optane DC Persistent Memory (DCPMM) was commercialized, memory systems using both DRAM and NVM memory did not take DCPMM's performance into consideration.

We present High Probability Write Patterns (HPWP), an optimization policy for tiered memory systems, in consideration of the commercialized DCPMM performance. HPWP prevents DCPMM from generating write operations as much as possible through the fact that write performance of DCPMM is three times worse than read performance. In a tiered memory system equipped with DCPMM, HPWP provides up to 19% performance improvement in key-value store compared to previous studies.

Contents

I	Introduction	1
II	Background	3
	2.1 Application Memory Demands	3
	2.2 Intel Optane DC PMM	3
	2.3 Tiered Memory System	4
III	Related Work	5
	3.1 AutoTiering	5
	3.2 Thermostat	5
	3.3 Nimble Page Management	6
	3.4 HyMM	6
	3.5 HeMem	6
IV	Design	8
	4.1 Data Classification	9
	4.2 Data Placement	9
	4.3 Page Migration	10
V	Implementation	12
	5.1 Memory Management	12

5.2	Access Count Management	12
5.3	Memory Migration	12
VI	Evaluation	13
6.1	Experimental Setup	13
6.2	Microbenchmarks	13
6.3	Application Benchmarks	16
VII	Limitation and Future Works	18
7.1	Limitation	18
7.2	Future works	18
VIII	Conclusion	19
	References	20
	Acknowledgements	22

List of Figures

1	Tiered Memory System overview	4
2	Tiered main memory management system designs with HPWP	8
3	HPWP policy behavior	10
4	Hot memory threshold sensitivity	13
5	Data classified as hot and actual hot data	14
6	High probability write threshold sensitivity	15
7	Page classification according to the change of high probability write threshold	15
8	GUPS with different hot set sizes	16
9	FlexKVS throughput (Mops)	17

I Introduction

Recently, Intel launched Optane DCPMM, a commercial persistent memory [1]. DCPMM is a persistent medium and can be used as a storage device. Also, since it has the same interface as DRAM, it can be used as a main memory. In particular, DCPMM has a higher density than DRAM, which enables large capacity development per module, and many academics and industries are trying to use DCPMM as a main memory that can accommodate applications such as machine learning that require large memory [2–4]. However, since DCPMM has relatively lower latency and bandwidth than DRAM, system performance may be very slow when DCPMM alone is used as a main memory. Therefore, many research groups are not using DCPMM alone, but are actively conducting research on a Tiered Memory System that consists of DRAM and hybrid and drives the main memory [5–7].

Since the Tiered Memory System uses heterogeneous memory with different characteristics, it is necessary to efficiently manage data accessed to each memory. DCPMM has the advantage of providing high capacity, but has lower performance than DRAM, so if frequently accessed data is stored in the DCPMM, system performance may be very degraded. Therefore, in order to improve the performance of the tiered memory system, many studies have proposed techniques to accurately classify the hot/cold characteristics of data loaded from application programs, and to efficiently arrange and migrate the separated data to heterogeneous media [5–12].

AutoTiering [6], a representative related technology, makes it possible to manage the access history for each page where data is loaded in the Tiered Memory System using Persistent Memory and DRAM, and distinguish the access frequency for each page. In this way, pages exceeding a certain frequency are considered hot data and can be placed in DRAM. However, since these traditional studies developed policies without considering the architectural characteristics of the commercialized DCPMM, it is difficult to say that they accurately considered the characteristics of the actual PM.

We propose the High Probability Write Pattern Aware Migration Policy (HWPAP) technology, which classifies the hot/cold characteristics of data, efficiently deploys and migrates the classified data, considering the architect’s characteristics of DCPMM, a commercial persistent memory. HWPAP technology uses the characteristics of latency and bandwidth that are different from those of conventional DRAM in read/write operations of DCPMM, which is an actual PM module, to more precisely classify the hot/cold characteristics of data in a Tiered Memory System. In particular, considering that the write bandwidth of DCPMM is extremely poor compared to other patterns, it additionally classifies data that is likely to be accessed by write from hot/cold data. By placing the data in DRAM, we want to avoid the write operation of DCPMM as much as possible.

We evaluate through microbenchmark and application benchmark by adding the HPWP policy, which is a policy that considers performance changes according to access patterns. We configure parameters to apply to HPWP through GUPS, a microbenchmark, and measure

performance results through application benchmarks graph500 and FlexKVS. It shows up to 19% better performance compared to the existing policy of HeMem.

This thesis proceeds in the following order. Section II describes the background. Section III summarizes related studies. Section IV summarizes the policies presented in this paper, and Section V summarizes the implementation method of HPWP. Section VI evaluates and analyzes the proposed technology. Section VII summarizes the limitations of HPWP and future work. Finally, Section VIII presents the conclusions about this study.

II Background

2.1 Application Memory Demands

Recently, an application that requires a large amount of memory has emerged. For example, data centers are generating large amounts of data with the spread of the Internet, and in order to smoothly access large in-memory datasets, its are searched using a database and graph processing system in memory. In addition, the machine learning systems train on huge in-memory datasets to increase accuracy in the process of analyzing data and self-learning [13]. If the capacity of the memory is increased, it is possible to satisfy the application that requires the use of a large amount of memory. The number of DIMM slots per server can be increased to expand memory capacity, but increasing DRAM density is currently limited.

2.2 Intel Optane DC PMM

Intel’s Optane DCPMM is used as main memory using the same DIMM interface as DRAM. DCPMM offers high capacity up to 512GB per module, supporting larger capacities than DRAM modules. Unlike DRAM, DCPMM also provides persistence, allowing data to be retained even when power is turned off, protecting the system from data loss. DCPMM has a larger capacity than DRAM, but overall DRAM is better than DCPMM in terms of performance. This makes DCPMM attractive for expanding DRAM in tiered memory configurations. Applications running on a server with DCPMM installed can utilize a larger main memory pool. However, to simply use DRAM and DCPMM together, the behavior of the two memories is different. DCPMM has asymmetric read and write bandwidth, larger cache lines than DRAM, and wears out faster than DRAM. These factors show that DCPMM is much more sensitive to memory access patterns than DRAM. Therefore, it is necessary to accurately understand the operation of DCPMM for memory access, which is one of the important elements in the Tiered Memory System.

We compare the performance of DRAM and DCPMM in detail. Table 1 compares the latency and bandwidth of DRAM and DCPMM. Overall, Overall, DRAM performs better than DCPMM. Looking in detail, DRAM’s read latency is about 2.1 times faster than DCPMM, and DRAM’s write latency is 1.1 times faster than DCPMM. Also, DRAM has about 3.3 times better read bandwidth than DCPMM, and DRAM has about 7.2 times better write bandwidth

Memory	Latency (ns)		Bandwidth (GB/s)		Capacity (GB)
	Read	Write	Read	Write	
DRAM	82		107	80	Up to 128
PM	175	94	32	11	Up to 512

Table 1: Comparison between DRAM and PM

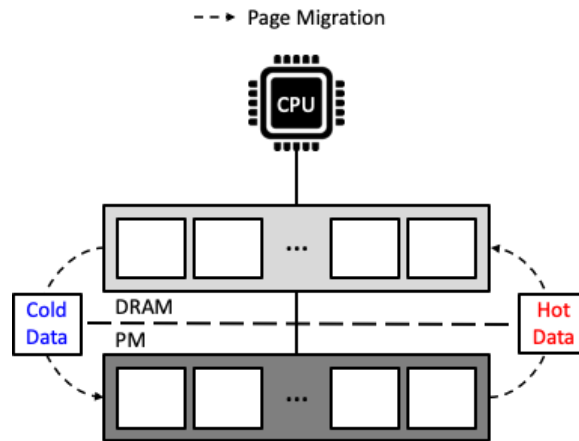


Figure 1: Tiered Memory System overview

than DCPMM.

2.3 Tiered Memory System

The goal of the Tiered Memory System, which uses two memories with different characteristics, is to place as much data that requires a lot of access in fast memory as possible. Figure 1 shows the structure of a tiered memory system with a single CPU socket. Basically, if (hot) data that is heavily accessed exists in the PM, it is migrated to DRAM, and if (cold) data that is less accessed exists in the DRAM, it is migrated to the PM. Data classified as hot represents data to be accessed in the future and is probabilistic. Therefore, studies aimed at increasing the accuracy by identifying the pattern of the data being accessed have been conducted [5, 14]. However, before DCPMM was commercialized, accessibility was checked in Tiered Memory System without accurately considering the characteristics of DCPMM. Therefore, the accuracy of the policy to determine the data accessed to the Tiered Memory System created before DCPMM is commercialized can be questioned. Therefore, it is essential to study the policy to classify hot/cold data considering the characteristics of DCPMM.

Previous studies on tiered memory systems have evaluated through emulated non-volatile memory (NVM) and do not accurately take into account the characteristics of commercialized DCPMMs. Looking at Table 1, it can be seen that DRAM shows higher performance than DCPMM for both read and write operations, but DCPMM's write bandwidth is extremely poor compared to other operations. Therefore, in order to make a tiered memory system using DCPMM and DRAM, DCPMM should accurately identify the difference from DRAM and consider the page migration policy.

III Related Work

There are several studies for Tiered Memory System. The first is to decide "which page" should be placed in "what memory". By predicting the pages to be accessed in the future and placing them in the fast memory, the performance of the fast memory can be exhibited. Since it is impossible to accurately predict which pages will be accessed in the future, information on whether or not recently accessed pages have been accessed is stored for each page, and the most accessed pages are placed in a fast memory. The second is to reduce the overhead that occurs when page migration. *When a page migration occurs, a memory copy is made to move from memory to memory. For this reason, inappropriate page migration causes unnecessary overhead, resulting in overall performance degradation. Third, the memory layer must be distinguished. Since it is a single memory system in which only DRAM exists in the past, memory in a multi-node is only distinguished from local and remote. However, with the release of DCPMM, it is necessary to divide the layers by separating DRAM and DCPMM as well as local and remote. Below, each representative tiered memory system-related research is summarized in more detail.

3.1 AutoTiering

AutoTiering [6] reconfigures the memory tier on NUMA systems. Because the existing NUMA system uses only a single memory, local access is given priority. However, in a NUMA system using heterogeneous memory, it is difficult to say that local access always optimizes performance. AutoTiering suggests a new memory tier based on remote DRAM showing higher performance than local DCPMM. In addition, we suggest a policy to optimize page migration. When page migration from slow memory to fast memory occurs, if all the fast memory space is being used, the least frequently accessed page should be migrated to slow memory first. In order to migrate from slow memory to fast memory, there is a disadvantage that a prior work is required. AutoTiering optimizes page migration operations by hiding page demote latency by reserving some space in fast memory so that it doesn't use all of the space in fast memory.

3.2 Thermostat

Thermostat [15] is a policy study to efficiently judge the hotness of Transparent Huge Page (THP) in a system using THP. Conventionally, in order to determine the hotness of THP, all 512 4KB pages are scanned. This method accurately identifies the hotness of THP, but can be a huge overhead. To solve this, Thermostat samples some of the 512 4KB pages that make up THP to check the page access. Accuracy decreases because only a portion is checked, but it has the advantage of providing a small scanning overhead to determine the hotness of THP.

3.3 Nimble Page Management

Nimble Page Management [7] improves inefficient page migration of 2MB THP. Previous studies have divided THP into 512 4KB pages when migrating each of them. Therefore, the overhead of separating and merging THP occurs, and at the same time, a bottleneck of sequentially migrating the divided data occurs. Nimble Page Management solves this problem by adding a policy to migrate by THP without separating THP. In addition, this study improves the problem that data in both memories must be copied in one direction in order to migrate. In order to perform a one-way copy operation, independent page allocation and deallocation must be performed, which leads to overhead. Nimble Page Management solves this problem by eliminating allocation and deallocation tasks by adding a policy that only exchanges data with pages that already exist.

3.4 HyMM

HyMM [14] studies a new hot & cold classification method of data accessing memory in a tiered memory system. In the previous study of HyMM, the page access_bit of the page table entry (PTE) was checked to distinguish between hot and cold pages. However, this method incurs the overhead of scanning numerous PTEs to access pages in applications that use large amounts of memory. To solve this problem, HyMM distinguishes between hot & cold by monitoring the number of TLB misses instead of access bits. A small number of TLB misses will occur in the case of a cold page that is rarely accessed, and a large number of TLB misses in a hot page that receives a lot of access. However, since "very hot" pages can be kept in the TLB, there can be fewer TLB misses than cold pages, which makes accurate predictions impossible. Since the distribution of "very hot" pages has a very small proportion, by additionally examining the access bits of the corresponding page, pages that can be incorrectly classified as cold pages can be classified with relatively little overhead. In addition, HyMM distinguishes page access patterns in addition to hot cold classification of pages. HyMM predicts whether a page will be accessed as a read or a write in the future by examining the dirty bit of the PTE. The study predicts the read/write of a page with up to 96% accuracy.

3.5 HeMem

HeMem [5] improves the overhead that occurs in the page access process for page migration. The method of tracking PTE to distinguish between hot and cold pages has high CPU overhead for page table search and TLB shutdown. To solve this problem, HeMem analyzes access patterns by sampling memory locations accessed through CPU events. Through this method, frequent page table scans and TLB shutdowns are not required, which greatly reduces overhead. To further reduce CPU overhead, DMA is used to replace the memory copy operation performed by the CPU operation during migration. Also, when a page is accessed through the page table, page table scan and page migration are performed in the same thread. Therefore, page migration

and page table scan are not processed at the same time. Because HeMem manages page access through CPU events, it separates page access work and page migration so that other threads can perform it, preventing delays caused by the two tasks.

IV Design

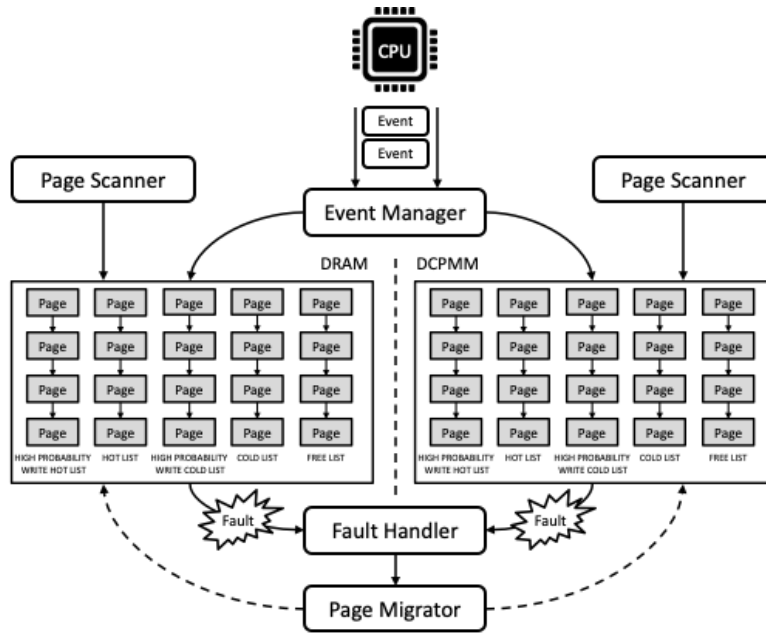


Figure 2: Tiered main memory management system designs with HPWP

In this section, we describe the design of the tiered memory system that is the basis of the current study and explain the newly added High Probability Write Pattern (HPWP) policy.

Page migration is essential for performance improvement in tiered memory systems [5–7,15]. The goal of page migration is to hide the performance of DCPMM as much as possible, bring out the performance of DRAM, and provide users with large memory. Basically, pages with many accesses are placed in fast memory, and pages with few accesses are placed in slow memory. However, since it is impossible to predict which pages will be accessed in the future, it chooses the suboptimal method of increasing accuracy.

The design of the tiered memory system to which the HPWP policy is applied is shown in Figure 2. By using the RWAP policy, data that has been previously divided only into hot cold is classified more precisely, and performance improvement is guaranteed when an application using a large memory is operated. In the HPWP policy, information is stored in the page considering not only whether the page is accessed, but also the pattern when accessing the page. Data is classified in detail and each memory has 5 FIFO queues to manage both Hot Cold as well as Read Write separately. The following explains the roles of the elements shown in Figure 2.

Event Manager

Event Manager monitors memory access data via CPU events. Many CPU events occur, but only two events (all load instruction and all store instruction) are considered for HPWP . Page stores information when an event occurs and determines whether it is accessed by read or write.

Fault Handler

Fault Handler is called to perform page migration when a page accessed due to a CPU event has a fault. Because the system is configured at the user-level, the virtual address of the memory map is registered in `userfaultfd` to handle page faults. So, if a page fault occurs within the range registered in `userfaultfd`, it is transferred to a dedicated thread that handles only the page fault to handle the page fault.

Page Scanner

Page Scanner continuously scans pages where events occur. Page checks the total number of times the page was accessed for migration and how recently it was accessed. In addition, access patterns can be distinguished through the Event Manager, so it is possible to determine whether it is read or write. Page Scanner requests that the updated information be moved to the appropriate list when certain conditions are met.

Page Migrator

Page Migrator migrates the page faulted by the Fault Handler. When the system decides to migrate a page, it uses `userfaultfd` to mark the page as write-protected. While migration is in progress, reads are possible, but writes are prohibited. Migration does not occur if page access does not meet certain conditions.

4.1 Data Classification

For efficient data placement in DRAM and DCPMM, HPWP requires analysis of access patterns as well as the number of times data is accessed in memory. In order to distinguish between hot and cold, HPWP uses a mechanism that can observe whether memory is accessed and how often it is accessed as the most used method []. As described in 3.1, the existing kernel uses the page table scan method for page access, which incurs a huge overhead when large memory is used. To solve this, HPWP uses Processor Event Based Sampling (PEBS), a method used in HeMem, to minimize its overhead. In PEBS, a process writes an event to be tracked in advance among events occurring in the CPU to an allocated memory buffer. Through this method, it is possible to distinguish whether the page accessed through the event is accessed by reading or writing.

4.2 Data Placement

Each memory organizes data types into Read Write lists as well as Hot Cold, and classifies data into four criteria. When a CPU event is accessed to a part of the memory area managed by `mmap`, the system increases the number of page accesses. If the access frequency of a page exceeds a certain threshold, the page is judged as Hot. In addition, by classifying access

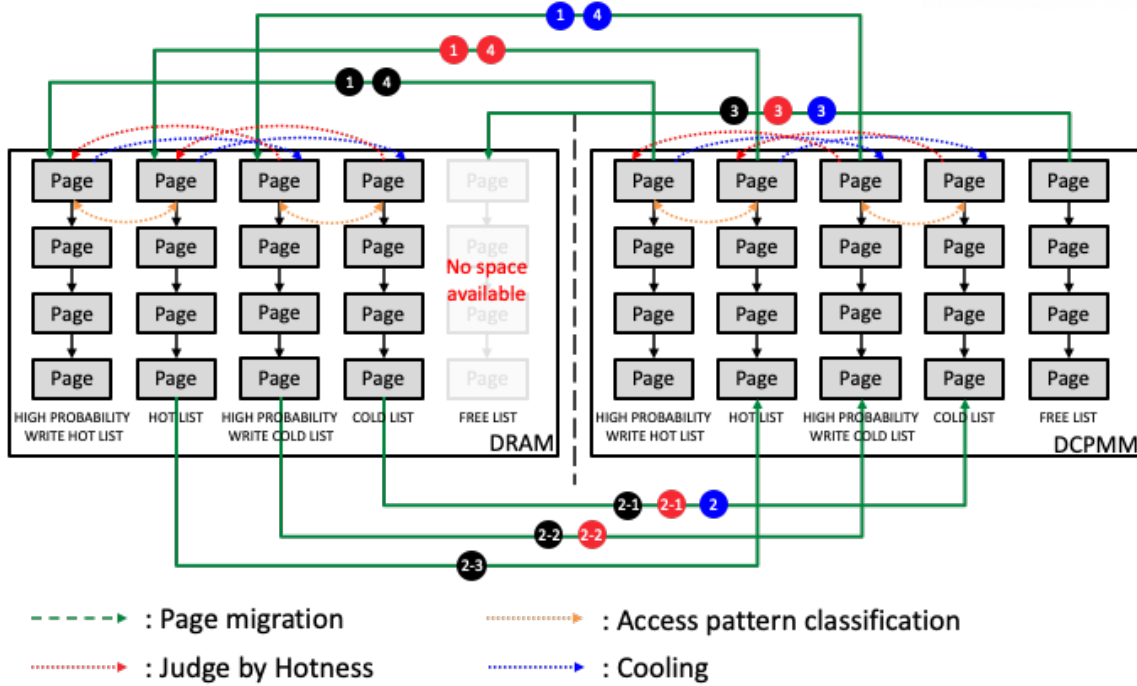


Figure 3: HPWP policy behavior

patterns, it is determined whether access is accessed by read or write access. Through this information, it records how many writes were accessed recently for each page. If the number of write accesses satisfies a specific condition, the next access is predicted as write.

4.3 Page Migration

Pages are divided into Hot / Cold Read / Write, and each memory has a total of 5 FIFO queues: *HIGH PROBABILITY WRITE HOT LIST*, *HOT LIST*, *HIGH PROBABILITY WRITE COLD LIST*, *COLD LIST*, and *FREE LIST*. Figure 3 shows in detail how page migration works for each memory. First, the green lines indicate all paths where page migration occurs. The page migration with the highest priority is *HIGH PROBABILITY WRITE HOT LIST*. If a page exists in the *HIGH PROBABILITY WRITE HOT LIST* of DCPMM, it requests migration to the *HIGH PROBABILITY WRITE HOT LIST* of DRAM. If all the space in DRAM is used and there are no pages in *FREE LIST*, migrate the pages in *COLD LIST* of DRAM to DCPMM first. If there is no page to migrate from *COLD LIST* of DRAM to DCPMM, the page of *HIGH PROBABILITY WRITE COLD LIST* of DRAM is migrated to DCPMM. Similarly, if the DRAM *HIGH PROBABILITY WRITE COLD LIST* page does not exist, the DRAM *HOT LIST* page is moved to DCPMM. The second priority page migration is *HOT LIST*. Migration proceeds in the same order as *HIGH PROBABILITY WRITE HOT*, but pages in the *HIGH PROBABILITY WRITE HOT LIST* of DRAM have the highest priority and therefore cannot be migrated to DCPMM. The page migration with the lowest priority is *HIGH PROBABILITY WRITE COLD LIST*. Unlike the existing system, COLD data migrates

to DRAM. It proceeds in the same way as the page migration described above. If there are no pages in the DRAM *FREE LIST*, only the DRAM *COLD LIST* pages can be migrated to DCPMM to create space.

The red line represents moving in the same memory if a page is judged to be hot when accessed. The blue line represents the regular cooling of the page being tracked. If there is no cooling operation, even if the page marked as Hot is no longer accessed, it continues to remain in the *HIGH PROBABILITY WRITE HOT LIST* or *HOT LIST*, so cooling is required on a regular basis. After a cooling operation, if the number of accesses of a page is below the threshold to be considered hot, it is marked cold and placed in the cold list depending on the memory type and *HIGH PROBABILITY WRITE COLD* or *COLD*. Finally, the orange line distinguishes pages that are likely to be accessed by write and pages that are not, through access patterns. When a page is traced due to a CPU event, it can be determined whether it is accessed as read or write.

V Implementation

This section describes the overall implementation method of the system to which the newly added HPWP policy is applied.

5.1 Memory Management

DCPMM is used in App-Direct Mode and exposed to user space through DAX file. DRAM is managed as a DAX file like DCPMM using the `mmap` command in the kernel.

5.2 Access Count Management

Perf of the kernel is used to manage the event, which is the access information of the page. Among the functions of *Perf*, the system can receive necessary event information through *perf_event_open*. To distinguish between read and write, we only record specific events. Read is recognized through *MEM_INST_RETIRED.ALL_LOADS* that can recognize all load information, and write is recognized through *MEM_INST_RETIRED.ALL_STORES* that can recognize all store information. In accordance with the existing policy, we classify hot/cold according to the total number of accesses, regardless of access patterns. In addition, cooling is performed regularly to maintain the freshness of hot data. Cooling is performed using a clock so that the FIFO queues of all tracked memories do not have to be checked each time the threshold that separates hot/cold is reached. When any page meets the cooling threshold, the clock is incremented. The next time the page is accessed, if the last time the page cooled down does not match the current clock, the page is halved before the number of accesses increases. We show a study in which parameters were adjusted in 6.2.

5.3 Memory Migration

In HPWP, memory migration runs in the background every 10ms. When the migration starts, mark the page as write-protected using `userfaultfd`. When migration is complete, the page returns to the writable state. When memory migration occurs, copy occurs for data movement, and the corresponding operation is carried out on the CPU, which puts a burden on the CPU. So, if the I/OAT DMA engine can be used in the system, the memory migration is freed from the CPU by utilizing DMA.

VI Evaluation

We conduct various evaluations based on the micro-benchmark GUPS to analyze the performance of the system to which the HPWP policy is applied. After that, using a graph processing system such as graph500 and a key-value store, evaluate how the HPWP policy affects the actual application. The HPWP policy is compared to the HeMem, a system that has been recently studied and is the basis of the policy.

6.1 Experimental Setup

We run the evaluation on a single socket server equipped with an Intel(R) Xeon(R) Gold 6242 processor. The socket is equipped with 192GB DRAM ($32GB \times 6$) and 512GB Intel Optane DC PMM ($256GB \times 2$). In real experiments, we force only 48GB DRAM and 256GB DCPMM to be used for workloads up to 200GB. We use Linux kernel 5.1 and Ubuntu 20.04 server. The NUMA effect of tiered memory is not covered in this paper. Therefore, each experiment is conducted on a single NUMA node. The benchmark uses GUPS [16], a micro-benchmark, for basic testing, and FlexKVS [17], an in-memory key-value store. The size of the page used in the experiment is THP, which is a 2MB page.

6.2 Microbenchmarks

We use the microbenchmark GUPS [16] to evaluate the basic operation of Tiered Memory System. GUPS executes read-modify-write operations in a random pattern with uniform or skewed objects of fixed size and measures Giga Update Per Second (GUPS). We run each benchmark 5 times and report the average GUPS. We run GUPS with 16 threads, and perform 1 billion updates on objects of 8 bytes per thread.

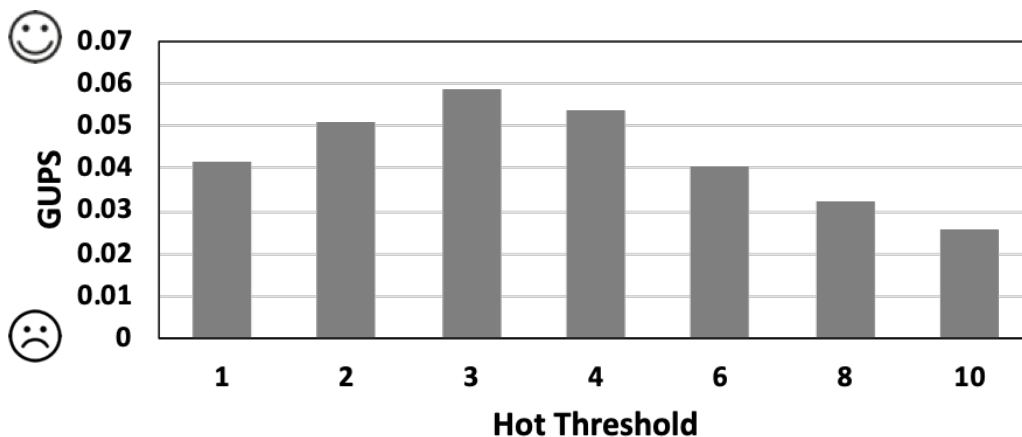


Figure 4: Hot memory threshold sensitivity

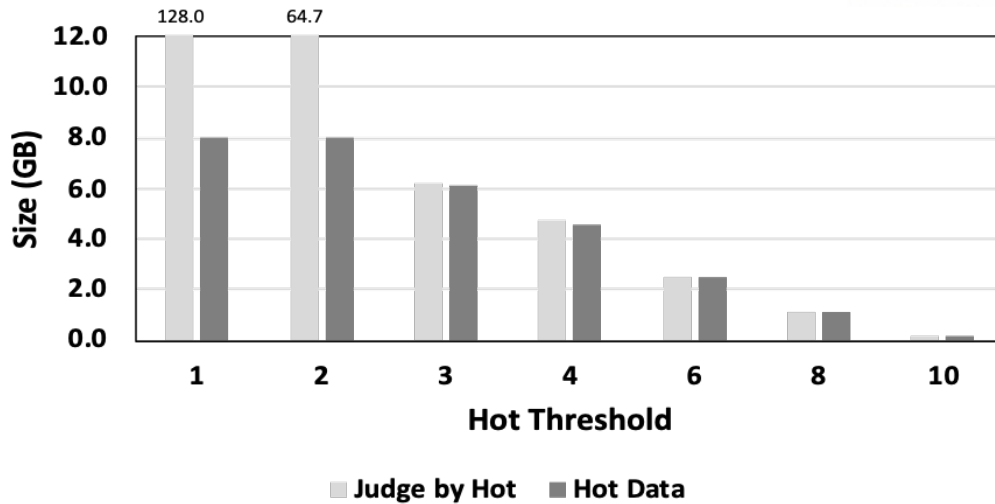


Figure 5: Data classified as hot and actual hot data

Hot Memory Threshold

We set the criteria for judging the hotness of a page through GUPS. To experiment, randomly create a *hot set* among the data. *Hot set* means data that is accessed uniformly by 90% of each thread's work. The remaining 10% is uniformly accessed to objects that are not classified as hot sets. A tiered memory system to which HPWP is applied measures how many hot sets are held in DRAM by setting hot sets. We set the total working set size to 128 GB and the hot set size to 8 GB.

Figure 4 shows the result of changing the hot memory threshold. A low threshold (1-2) will overestimate the hot set and degrade performance. Thresholds greater than 3 can differentiate between hot and cold pages, but the hot sets are underestimated, which reduces performance.

Figure 5 details the reasons for the results shown in Figure 4. A low threshold value (1-2) judges more data than a hot set as hot and does not accurately distinguish hotness. Therefore, a hot set may exist in DCPMM, and performance degradation occurs. If the threshold value is 2, the data determined to be hot is 64.7 GB. Based on this, if the DRAM size is sufficient, it can cover all hot sets and show better performance. However, it cannot be judged as good because it cannot accurately distinguish between hot / cold. If the threshold value is greater than 3, hotness is accurately classified because most of the data determined to be hot is included in the hot set. However, as the threshold increases, only a small amount of data among the hot sets is judged as hot, so actual hot data may exist in the DCPMM, resulting in performance degradation. Based on this, we set the hot threshold to 3.

High Probability Write Threshold

Before evaluating the HPWP, the criterion of the number of recent writes is established to make a judgment to predict that the page will be accessed by write next time it is accessed. As in

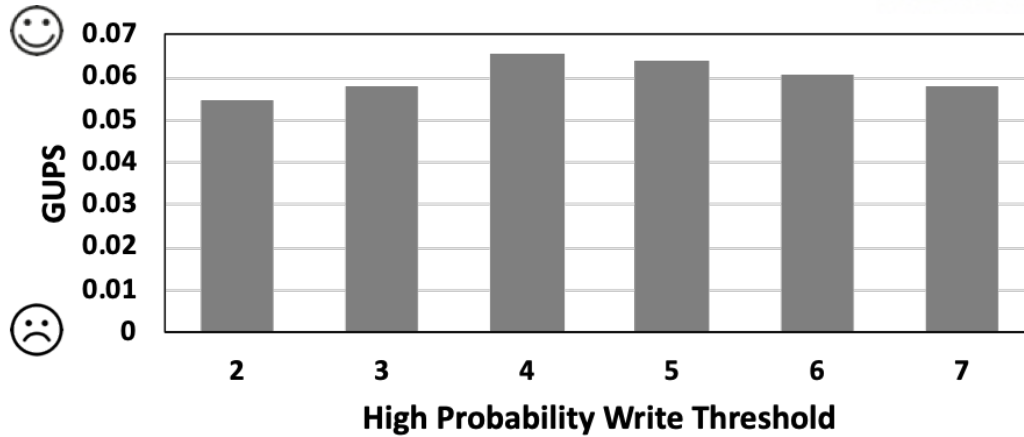


Figure 6: High probability write threshold sensitivity

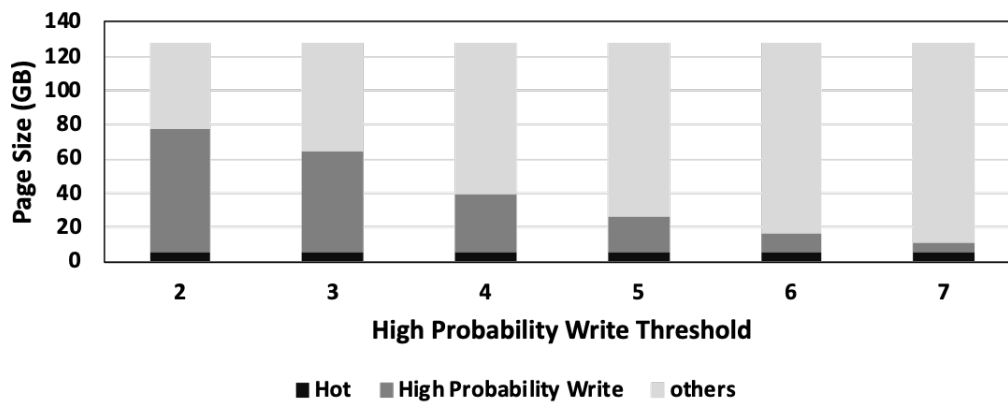


Figure 7: Page classification according to the change of high probability write threshold

the previous experiment, set the working set size to 128 GB and the hot set size to 8 GB. Also, the hot threshold is fixed at 3.

Figure 6 shows the results by changing the high probability write threshold. A low threshold (2-3) causes pages to be considered writes overestimated and degrades performance. A threshold value greater than 4 clearly distinguishes the pages to be accessed as writes, but the pages to be judged as writes are underestimated, resulting in poor performance.

Figure 7 details the reasons for the results shown in Figure 6. In all cases, since the hot threshold is fixed, the data classified as hot is constant. However, if the threshold for determining which pages are judged to be accessed by write is low, it is judged that many pages will be accessed by write. However, since the accuracy of the page judged as such is down to 95%, it cannot be said that it has been accurately judged. In addition, as the threshold increases, pages that are expected to be accessed by write are classified with a maximum accuracy of 99.5%. But it doesn't classify many pages, so it doesn't bring any performance benefit. If the size of the DRAM can cover all the pages to be accessed by writes, it will give the best performance, but the system cannot guarantee that this is always the case. That's why we set the threshold

by balancing it.

Hot set

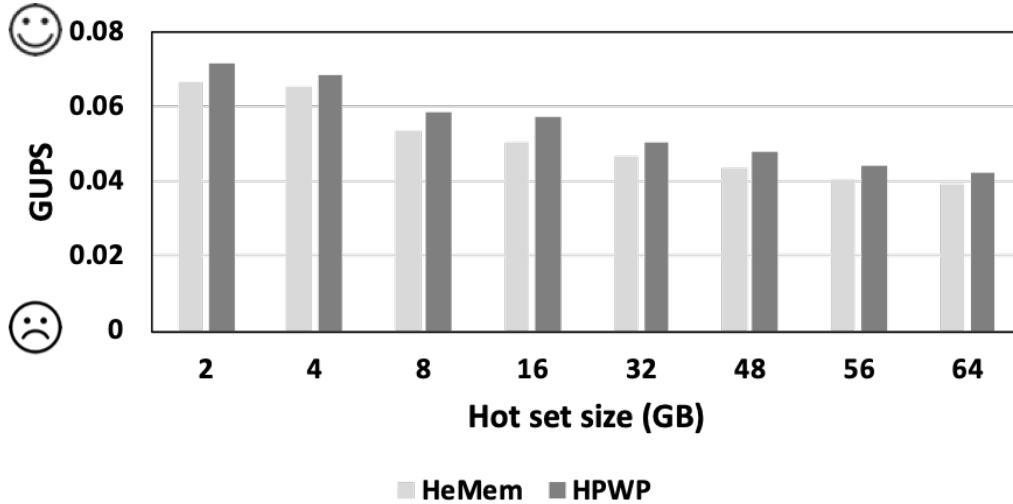


Figure 8: GUPS with different hot set sizes

Figure 1 shows the performance difference between HeMem and HPWP according to the change in the size of the hot set. If the hot set is smaller than the size of the DRAM, the system tries to keep all the hot set in the DRAM as much as possible. HPWP hides the write performance of DCPMM as much as possible by additionally keeping data that can be accessed by write as well as hot set in DRAM. If the hot set is larger than the DRAM, data that can be accessed by write from the hot set is maintained in the DRAM. HPWP occurs twice as many page migrations as HeMem, but performs up to 13% better.

6.3 Application Benchmarks

We now evaluate how applications using big data run using a tiered memory system with HPWP applied. First, using the in-memory key-value store, FlexKVS, evaluates the operational throughput and latency according to the ratio of read and write. Second, the runtime is evaluated with graph500, a graph processing system.

Key-Value Store

We evaluate FlexKVS [17], a key-value store, using tiered memory. FlexKVS is compatible with MemCache, but uses logs to reduce synchronization overhead and uses blockchain hash tables to minimize cache coherence overhead. We evaluate the throughput of FlexKVS using a server running 8 threads and 1 client machine running 16 threads. Also, a key-value pair with a size of 4KB is used. The total working set size is 200 GB and the hot set size is 40 GB. The hot set accounts for 90% of the total throughput.

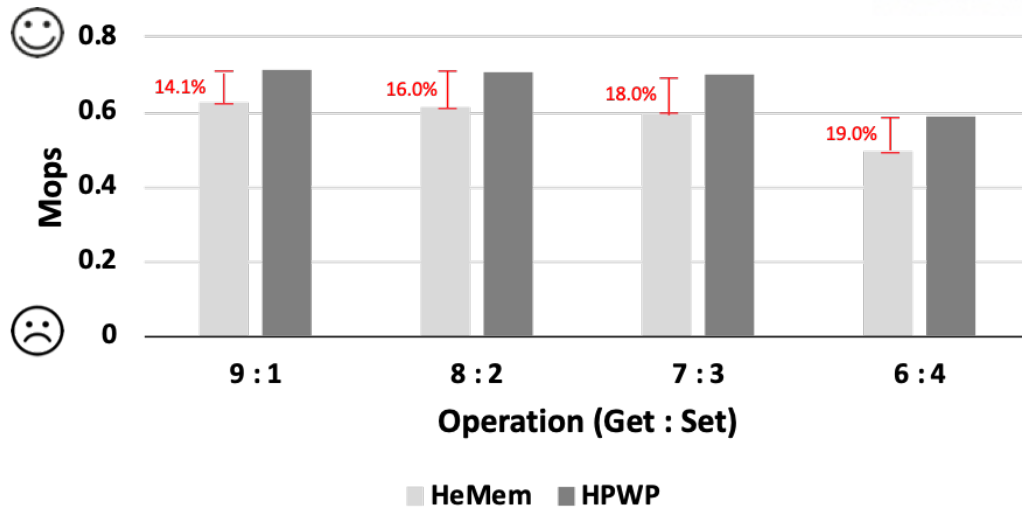


Figure 9: FlexKVS throughput (Mops)

Figure 9 shows the result of comparing throughput according to the ratio of GET and SET requests. HPWP provides up to 19% higher throughput than HeMem. As the ratio of set requests increases, the number of write operations increases and the overall performance decreases. However, it has a greater performance advantage by preventing more write operations from running in DCPMM.

VII Limitation and Future Works

7.1 Limitation

HPWP is a policy implemented considering only the write bandwidth of Intel's DCPMM. In Table 1, when the latency of DRAM and DCPMM is compared, write operates similarly, but DCPMM is delayed by twice that of DRAM in read. HPWP has a performance advantage when there are a lot of requests, but performance can suffer when dealing with a small number of relatively large data. We need to address the policy contradictions to deal with this.

7.2 Future works

We summarize the following to supplement the limitations described above.

- Pages are placed in memory by analyzing more detailed information through additional CPU events.
- It is implemented so that the priority of the operation can be changed according to the size of the data.

VIII Conclusion

By using the PM with DRAM, it provides a large amount of memory. However, due to the performance difference between the two memories, if memory management is not carefully managed, the performance may be adversely affected. This leads to studies on policies that use DRAM and PM efficiently. In previous studies, such a memory environment was referred to as a tiered memory system, and recent studies aimed to place only data that is accessed a lot in DRAM, and did not accurately consider the performance of commercialized PMs. In this paper, HPWP is presented and a new data classification method of tiered memory system is considered in consideration of the performance of DCPMM, a commercialized PM. As a result of testing the performance of the tiered memory system by applying the newly implemented HPWP policy, a result of up to 19% improvement was obtained compared to the policy of the existing system.

References

- [1] Intel optane dc persistent memory. [Online]. Available: <http://www.intel.com/optanedcpersistentmemory>
- [2] O. Patil, L. Ionkov, J. Lee, F. Mueller, and M. Lang, “Performance characterization of a dram-nvm hybrid memory architecture for hpc applications using intel optane dc persistent memory modules,” in *Proceedings of the International Symposium on Memory Systems*, 2019, pp. 288–303.
- [3] B. Lu, J. Ding, E. Lo, U. F. Minhas, and T. Wang, “Apex: A high-performance learned index on persistent memory,” *arXiv preprint arXiv:2105.00683*, 2021.
- [4] H. Huang, Z. Wang, J. Kim, S. Swanson, and J. Zhao, “Ayudante: A deep reinforcement learning approach to assist persistent memory programming,” in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021, pp. 789–804.
- [5] A. Raybuck, T. Stamler, W. Zhang, M. Erez, and S. Peter, “Hemem: Scalable tiered memory management for big data applications and real nvm,” in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, 2021, pp. 392–407.
- [6] J. Kim, W. Choe, and J. Ahn, “Exploring the design space of page management for {Multi-Tiered} memory systems,” in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 2021, pp. 715–728.
- [7] Z. Yan, D. Lustig, D. Nellans, and A. Bhattacharjee, “Nimble page management for tiered memory systems,” in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019, pp. 331–345.
- [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*, 2009, pp. 24–33.
- [9] H. Yoon, J. Meza, R. Ausavarungnirun, R. A. Harding, and O. Mutlu, “Row buffer locality aware caching policies for hybrid memories,” in *2012 IEEE 30th International Conference on Computer Design (ICCD)*. IEEE, 2012, pp. 337–344.

- [10] N. Agarwal, D. Nellans, M. Stephenson, M. O'Connor, and S. W. Keckler, "Page placement strategies for gpu within heterogeneous memory systems," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, 2015, pp. 607–618.
- [11] H. Wang, J. Zhang, S. Shridhar, G. Park, M. Jung, and N. S. Kim, "Duang: Fast and lightweight page migration in asymmetric memory systems," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2016, pp. 481–493.
- [12] H. Liu, Y. Chen, X. Liao, H. Jin, B. He, L. Zheng, and R. Guo, "Hardware/software cooperative caching for hybrid dram/nvm memory architectures," in *Proceedings of the International Conference on Supercomputing*, 2017, pp. 1–10.
- [13] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 2014, pp. 583–598.
- [14] L. Liu, S. Yang, L. Peng, and X. Li, "Hierarchical hybrid memory management in os for tiered memory systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 10, pp. 2223–2236, 2019.
- [15] N. Agarwal and T. F. Wenisch, "Thermostat: Application-transparent page management for two-tiered main memory," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 631–644.
- [16] Gups. [Online]. Available: <http://icl.cs.utk.edu/projectsfiles/hpcc/RandomAccess/>
- [17] A. Kaufmann, S. Peter, N. K. Sharma, T. Anderson, and A. Krishnamurthy, "High performance packet processing with flexnic," in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, 2016, pp. 67–81.

Acknowledgements

We would like to thank Prof. Sam H. Noh and Dr. Hyunsub Song, who greatly helped to complete the thesis.

