Allocation Policies Matter for Hybrid Memory Systems

Adnan Maruf
amaru009@fiu.edu
Florida International University
Miami, Florida, USA

Daniel Carlson
dcarl026@fiu.edu
Florida International University
Miami, Florida, USA

Ashikee Ghosh
ghashike@amazon.com
Amazon Web Services
New York, New York, USA

Manoj Saha
msaha002@fiu.edu
Florida International University
Miami, Florida, USA

Janki Bhimani
jbhimani@fiu.edu
Florida International University
Miami, Florida, USA

Raju Rangaswami
raju@cs.fiu.edu
Florida International University
Miami, Florida, USA

ABSTRACT

Existing tiered memory systems all use DRAM-Preferred as their allocation policy, whereby pages get allocated from higher-performing DRAM until it is filled, after which all future allocations are made from lower-performing persistent memory (PM). The novel insight of this work is that the right page allocation policy for a workload can help to lower the access latencies for the newly allocated pages. We design, implement, and evaluate three page allocation policies within the real system deployment of the state-of-the-art dynamic tiering system. We observe that the right page allocation policy can improve the performance of a tiered memory system by as much as 17x for certain workloads.

KEYWORDS

hybrid memory systems, memory allocation, memory tiering

ACM Reference Format:
Adnan Maruf, Daniel Carlson, Ashikee Ghosh, Manoj Saha, Janki Bhimani, and Raju Rangaswami. 2023. Allocation Policies Matter for Hybrid Memory Systems. In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing (HPDC '23), June 16–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3588195.3595946

1 INTRODUCTION

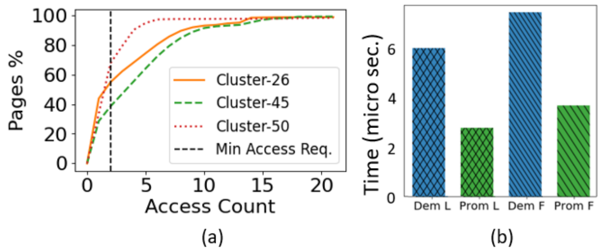


Figure 1: Page Selection (a) and migration (b) Overhead.

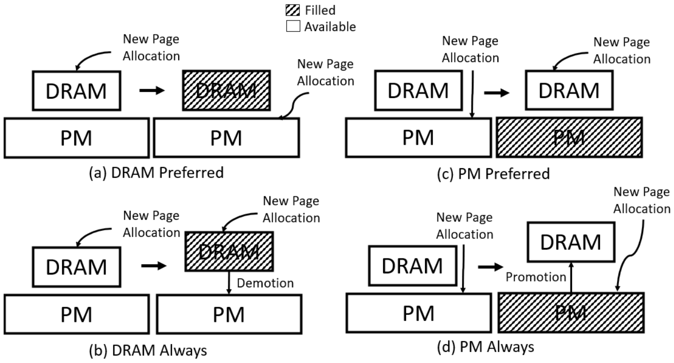
In tiered memory systems, optimizing dynamic page migration has received substantial attention. The state-of-the-art tiered memory systems such as MULTI-CLOCK, AutoTiering, AMP, Nimble, and others [6] dynamically reorganize the pages across memory tiers based upon the accesses to the pages. These systems improve the overall performance of the dynamic workloads by periodically scanning/sampling the page accesses to determine page importance and perform *page selection* followed by *page migration* to move the page(s) to an appropriate tier. However, to our surprise, we noticed that all the existing tired memory systems allocate the new pages in DRAM until full and then allocate the remaining new pages throughout the workload from the lower tier (i.e., the tier with higher access latency compared to DRAM). In the rest of this paper, we refer to this as the DRAM-Preferred allocation policy. Hence, in the steady state of any workload when DRAM is fully utilized, the important pages get accessed from the slower tier until the page is selected and promoted to the faster tier. *The novel insight of this research is that with the right page allocation policy in addition to tiering algorithm, the workload performance can be further improved by the lower access latencies as well as reduced page selection and migration overheads for the newly allocated pages.*

The *page allocation policy* of a tiered memory system determines which tier new page allocations are made from prior to engaging page migration mechanisms. Fig. 1(a) shows the page selection overhead of the dynamic tiered systems, with the number of page access on the X-axis and the percentages of total pages on the Y-axis for Twitter cluster traces [5]. The dashed line presents the required number of page accesses (i.e., 2) for Multi-Clock to select the page to migrate. *From Fig. 1(a), we can see that the accesses to the 67% of the total pages for cluster 50, 57% for cluster 26, and 38% for cluster 45 depend on the initial placement of the pages due to the page allocation policy used.* These pages would not even be considered for migrations as these pages have a total access count less than the required threshold. Fig. 1(b) shows the time taken to demote and promote a page in most of the dynamic tiering systems [6]. *With the default DRAM-Preferred allocation policy, hot pages that are allocated after the DRAM is filled will trigger migration. Thus, the number of initial demotions and promotions can be hundreds of millions for workloads like the Twitter clusters where the working set size of a cluster can reach terabytes [1]. These demotions and promotions would require a significant amount of time just for the migrations. The above overheads can be significantly reduced by choosing the right tier to allocate new pages.*

In this work, first, we design and implement three different allocation policies. Second, we integrate the above allocation policies with the state-of-the-art dynamic tiering systems deployed using

HPDC '23, June 16–23, 2023, Orlando, FL, USA

ACM ISBN 979-8-4007-0155-9/23/06.
https://doi.org/10.1145/3588195.3595946



(a) DRAM Preferred

(b) DRAM Always

(c) PM Preferred

(d) PM Always

Figure 2: Page allocation policies.

Linux kernel to study the impact of allocation policies within tiering systems using various real workload traces and several popular benchmarks.

We find that the page allocation policy is as important as the dynamic tiering policy. Allocation policies can impact the tiering system performance by up to 17x. While all of the state-of-the-art tiering mechanisms use the *DRAM-Preferred* allocation policy, we find that no single allocation policy performs the best for all workloads.

2 ALLOCATION POLICIES

DRAM-Preferred: All new pages are allocated from the DRAM tier as long as it has free space. Once the DRAM is filled, new pages are allocated from the PM tier. Fig. 2(a) illustrates this allocation policy.
DRAM-Always: All new pages are allocated from the DRAM, even if the DRAM is already filled. As Fig. 2(b) shows, when DRAM is filled, new page allocations force the demotion of cold pages to the PM tier.
PM-Preferred: New pages are allocated from the PM tier as long as the PM has free space. Once it is filled, new pages are then allocated from the DRAM tier. Fig. 2(c) shows how page allocations get made with the *PM-Preferred* policy.
PM-Always: Always allocates new pages from the PM tier as shown in Fig. 2(d). With *PM-Always*, in the Linux kernel, we observe that allocating new pages from the swap space while DRAM space is free often causes an Out Of Memory (OOM) error which kills the running application. Hence, in the rest of the paper, we focus our experiments on the remaining three allocation policies.

3 EXPERIMENT SETUP

All experiments are performed using an Intel Xeon Gold 5218 dual-socket processor with 16 cores per socket, i.e., 32 cores in total. The system is configured with twelve DDR4 DIMMs, totaling 192GB of DRAM, and 4 Intel Optane DC Persistent Memory (DCPM) DIMMs totaling 512GB of PM. We implemented the allocation policies in Linux kernel version 5.3.1. We used SPEC [7], NAS [2], YCSB [4], and GAPBS [3] benchmarks to evaluate the performance of the allocation policies.

4 IMPACT OF ALLOCATION POLICIES

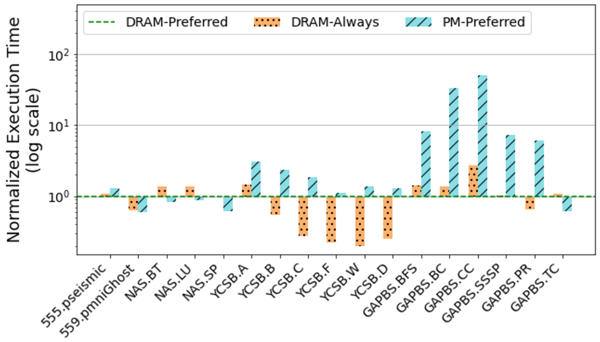


Figure 3: Impact of allocation policy on the dynamic tiered memory system.

Fig. 3 shows the large variation in performance that we can obtain by changing the allocation policy for various workloads. For example, in Fig 3, the performance of the dynamic tiering can significantly improve if the default allocation policy of *DRAM-Preferred* is replaced by the *DRAM-Always* allocation policy for the YCSB.D workload. In YCSB workload D, new items are added, and the most recent items are the most popular items [4]. With *DRAM-Preferred*, since new pages are allocated from the PM once the DRAM is filled, the newer popular pages would get accessed from the PM. On the other hand, with *DRAM-Always* allocation policy, newer pages containing popular items are allocated and accessed from the DRAM. Thus, *the allocation policies can significantly impact the performance of the tiered memory systems*. Furthermore, we observed that *no single allocation policy always performs best for different types of workloads*.

ACKNOWLEDGMENTS

This work was supported in part by NSF grants CCF-1718335, CNS-1956229, and CNS-2008324.

5 CONCLUSION

In this work, we investigated the problem of page allocation in tiered memory systems, which was previously unexplored. We introduced several allocation policies and evaluated the performance of these allocation policies in dynamic tiered memory systems by using a variety of workloads. We observed that allocation policies can significantly impact the performance of tiered memory systems.

REFERENCES

[1] 2020. Twitter Cache Trace Stat. https://github.com/twitter/cache-trace/blob/master/stat/2020Mar.md.

[2] David Bailey, Tim Harris, William Saphir, Rob Van Der Wijngaart, Alex Woo, and Maurice Yarrow. 1995. *The NAS parallel benchmarks 2.0.* Technical Report. Technical Report NAS-95-020, NASA Ames Research Center.

[3] Scott Beamer, Krste Asanović, and David Patterson. 2015. The GAP benchmark suite. *arXiv preprint arXiv:1508.03619* (2015).

[4] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking Cloud Serving Systems with YCSB. In *Proceedings of the ACM symposium on Cloud computing (SoCC '10).*

[5] Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a Social Network or a News Media?. In *Proceedings of the International World Wide Web Conference (WWW '10).*

[6] Adnan Maruf, Ashikee Ghosh, Janki Bhimani, Daniel Campello, Andy Rudoff, and Raju Rangaswami. 2022. MULTI-CLOCK: Dynamic Tiering for Hybrid Memory Systems. In *Proceedings of the 2012 IEEE 28th International Symposium on High Performance Computer Architecture (HPCA '22).*

[7] Standard Performance Evaluation Corporation (SPEC). 2018. SPEC Benchmarks. http://www.spec.org/benchmarks.html.