# Suitable Data Structures for Heterogeneous Memory

Jae Woo Joo

## 1. INTRODUCTION

Nowadays data center applications such as in-memory databases [1] and key-value stores [2, 3] require increasingly large memory capacities to handle exponentially expanding datasets, but confront the challenges due to the difficulties in scaling DRAM and the cost of DRAM. Because of the limitations, the current DRAM structure will be unsuitable in the future. In order to overcome this challenge, researchers are exploring new memory technologies such as non-volatile memory (NVM). Table 1 summarizes the key attributes of byte-addressable NVM such as phase change memory (PCM) and DRAM. NVM provides higher capacity (5×) than DRAM, but lower read (2× to 4×) and write (5× to 10×) latency, and lower bandwidth (4× to 8×) [4, 5, 6]. This implies that current memory management technology is inefficient for data center and thus future systems demand heterogeneous memory architecture by combining DRAM and NVM.

Attaching NVM and DRAM directly as NUMA nodes to the systems reusing the existing OS could be efficient in terms of the cost and scaling benefits [5]. However, there exists a concern about the performance overheads due to the slower latency compared to the DRAM. The goal of homogeneous memory system is to maximize temporal and spatial locality when accessing data in order to reduce the latency. Compared to the homogeneous memory system, heterogeneous memory system make demands on placing appropriate data into the fastest memory. Placing the critical data in inappropriate memory could lead to a performance overhead with less than 50% overheads [6]. Therefore, taking full advantage of heterogeneous memory technologies requires changes in OS-level.

Even though many data center applications try to optimize the data to take advantages of memory efficiency [5], OS kernel plays an important role in large memory systems performance such as memory allocation, page migration, and network. Thus it is crucial to understand the OS-level data structure in order to leverage the performance in heterogeneous memory systems. To gain insight into understanding the kernel data structure in OS-level, we pose the following questions: (1) What is the usage pattern and the access pattern when stressing out a specific subsystem? (2) Are current data structures suitable for heterogeneous memory systems? (3) If not, how could we make changes to the OS kernel to improve the performance?

To answer these questions, we conduct experiments with microbenchmarks and we present the modification of the OS design which can be adapted to heterogeneous memory systems. This paper makes the following contributions:

| | DRAM | NVM |
|---|---|---|
| Cost | 5× | 1× |
| Capacity per CPU | 100s of GB | Terabytes |
| Read Latency | 1× | 2× - 4× |
| Write Latency | 1× | 5× - 10× |
| Bandwidth | 4× - 8× | 1× |

**Table 1: Comparison of new memory technologies [4, 5, 6]**

- **New profiling mechanism for memory management in OS-level.** We build a tool to collect and trace the data inside the kernel to see how the kernel data structures are used for NVM.
- **Modification of data structures in OS-level.** We revise several kernel data structures which are commonly used in the kernel in order to improve the performance of heterogeneous memory systems. (More information needed.)
- **Comprehensive evaluation.** We evaluate our OS design with various data center applications which concentrates on memory, storage, and network respectively. (More information needed.)

To the best of our knowledge, this is the first in-depth study of profiling the usage and the characteristics of various data structures in the OS-level. We conjecture that this study will benefit commercial vendors. The results of this study can help the researchers to understand and to overcome the overheads when designing the heterogeneous memory systems.

(Additional paragraph) In large memory systems, not only the data from the application but also the kernel data structures should be allocated to certain memory (e.g., DRAM, NVM). When the application has different memory access patterns, it could touch the same page multiple times or it could touch different pages. Hence, placing the kernel data structures in an appropriate place in heterogeneous memory systems is critical. We extract the information from the OS-level how kernel data structures use different memory pages.

## 2. MOTIVATION

## 3. RELATED WORK

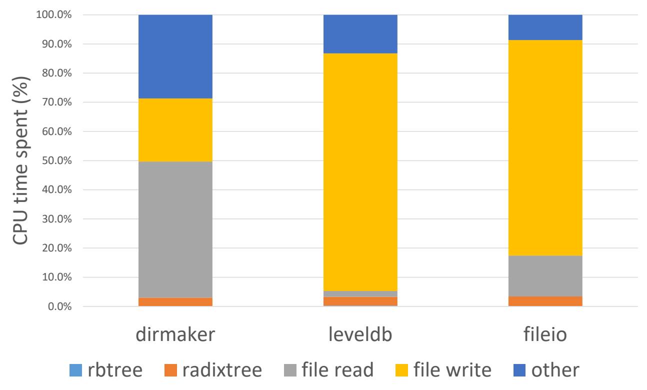

**Figure 1: CPU time spent on major functions in applications**

Agarwal et al. [7] proposed a new page placement policy that improves GPU performance in heterogeneous memory systems which differs the memory bandwidth capabilities. The proposed page placement policy places the critical data in the appropriate memory which improves the GPU performance. They use the profiling mechanism to collect the information of GPU application data structure memory access patterns and the size of the data structures allocated in the program.

## 4. REFERENCES

[1] M. Stonebraker and A. Weisberg, "The voltdb main memory dbms.," *IEEE Data Eng. Bull.*, vol. 36, no. 2, pp. 21–27, 2013.

[2] *Redis*. `https://redis.io/`.

[3] *Memcached*. `https://memcached.org/`.

[4] S. Venkataraman, N. Tolia, P. Ranganathan, R. H. Campbell, *et al.*, "Consistent and durable data structures for non-volatile byte-addressable memory.," in *FAST*, vol. 11, pp. 61–75, 2011.

[5] S. R. Dulloor, A. Roy, Z. Zhao, N. Sundaram, N. Satish, R. Sankaran, J. Jackson, and K. Schwan, "Data tiering in heterogeneous memory systems," in *Proceedings of the Eleventh European Conference on Computer Systems*, p. 15, ACM, 2016.

[6] S. Kannan, A. Gavrilovska, V. Gupta, and K. Schwan, "Heteroos: Os design for heterogeneous memory management in datacenter," in *ACM SIGARCH Computer Architecture News*, vol. 45, pp. 521–534, ACM, 2017.

[7] N. Agarwal, D. Nellans, M. Stephenson, M. O'Connor, and S. W. Keckler, "Page placement strategies for gpus within heterogeneous memory systems," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '15, (New York, NY, USA), pp. 607–618, ACM, 2015.