CARAM: A Content-Aware Hybrid PCM/DRAM Main Memory System Framework

Yinjin Fu, Yang Wu

CARAM: A Content-Aware Hybrid PCM/DRAM Main Memory System Framework

Yinjin Fu[1, 2] [0000-0001-9107-1338] and Yang Wu[2]

[1] PengCheng Laboratory, Shenzhen 518055, China
[2] Army Engineering University, Nanjing 210007, China
yinjinfu@gmail.com

Abstract. The emergence of Phase-Change Memory (PCM) provides opportunities for directly connecting persistent memory to main memory bus. While PCM achieves high read throughput and low standby power, the critical concerns are its poor write performance and limited durability, especially when compared to DRAM. A naturally inspired design is the hybrid memory architecture that fuses DRAM and PCM, so as to exploit the positive aspects of both types of memory. Unfortunately, existing solutions are seriously challenged by the limited main memory size, which is the primary bottleneck of in-memory computing. In this paper, we introduce a novel Content Aware hybrid PCM/DRAM main memory system framework—*CARAM*, which exploits deduplication to improve line sharing with high memory efficiency. CARAM effectively reduces write traffic to hybrid memory by removing unnecessary duplicate line writes. It also substantially extends available free memory space by coalescing redundant lines in hybrid memory, thereby further improving the wear-leveling efficiency of PCM. To obtain high data access performance, we also design a set of acceleration techniques to minimize the overhead caused by extra computation costs. Our experiment results show that CARAM effectively reduces 15%~42% of memory usage and improves I/O bandwidth by 13%~116%, while saving 31%~38% energy consumption, compared to the state-of-the-art of hybrid systems.

Keywords: Phase Change Memory, Hybrid Memory Management, Deduplication, Content Awareness, Line Sharing.

1 Introduction

The limited main memory capacity has always been a critical issue for multi/many-core systems to meet the needs of concurrent access to working sets. Unfortunately, conventional DRAM is not the ideal storage medium for in-memory computing due to its high power consumption, even though it achieves low access latency. Moreover, Phase change memory (PCM) is attracting an increasing attention as a promising candidate for next-generation memory[1]. However, there are some crippling limitations that prevent PCM from completely replacing DRAM in future systems, such as low write performance, high power cost of write access, and limited long-term endurance. These drawbacks have led designers toward the adoption of hybrid main memory architectures [2,3,4,5,6,7], which couple the large-capacity PCM with the

small-capacity DRAM, in order to combine the best of both memory media. Furthermore, *deduplication* can improve space efficiency by replacing redundant data with references to a unique copy in storage systems[8], due to its excellent ability in removing redundancy with higher throughput than lossless compression techniques[9]. Hence, we can enable deduplication for a hybrid PCM/DRAM memory structure that fits the characteristic of storage class memory due to their merits in DRAM-like performance and lower power consumption than DRAM.

In this paper, we present *CARAM*, a content-aware hybrid DRAM/PCM main memory system design framework by leveraging the deduplication technique at the line level. We use DRAM buffering for unique line writes to PCM, and also exploit the available DRAM space for memory address mapping to store deduplication metadata. We also introduce a deduplication-based hybrid memory line write processing to elevate space efficiency by enabling line sharing. Finally, we evaluate the space saving, I/O performance, and power consumption with real-world traces using a simulator that we build for content-aware hybrid memory evaluation. *To the best of our knowledge, this is the first study on the architectural design of content-aware hybrid PCM/DRAM main memory system by enabling line sharing to the most extent using deduplication.*

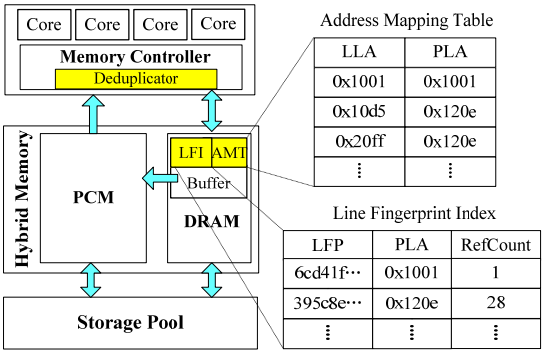

Fig. 1. The architecture of CARAM.

2 System Design of CARAM

2.1 The Overview of System Architecture

Our CARAM design aims to improve space efficiency, power efficiency, and the endurance limit of traditional DRAM/PCM main memory. Fig. 1 presents our architectural design of CARAM. The on-chip memory controller replays memory requests from the core's LLCs to the DRAM controller or the PCM controller. Line-level deduplication is performed in a deduplicator module, which generates line fingerprints with light-weight hashing SuperFastHash [12]. To support the deduplication process for line fingerprint management and line address mapping in duplicate identification, we store a line fingerprint index (LFI) and an address mapping table (AMT) in a persistent battery-backed DRAM, along with a write buffer for the PCM write accesses to overcome the slow write speed of PCM. We assign all the PCM and the remaining part of DRAM to a single physical memory address space used for the unique line writes after

deduplication. The hybrid main memory is used for page cache to hide the access latency in the underlying persistent storage pool, and all the unique pages in both DRAM and PCM are managed by page caching algorithms, such as LRU[6] and CLOCK[7].

In our CARAM design, the AMT is an in-memory table that consists of multiple entries, each of which is a key-value pair {logical line address(LLA), physical line address (PLA)}. Each entry requires 4B for storing the LLA and another 4B for PLA. The pair is a many-to-one mapping to support line sharing after deduplication. We need to update the AMT when there are new lines or line updates in hybrid main memory. The LFI is responsible for the fingerprint management of memory lines. Each of its entries contains a mapping between a line fingerprint (LFP) and a pair: {physical line address (PLA), RefCount}. RefCount presents the corresponding reference count in the hybrid main memory. Each fingerprint is 4B long for SuperFastHash value, while each RefCount is 2B long. Each entry is a one-to-one mapping to record the metadata information of a unique line in main memory at that time. The LFI can be updated when a memory line is renewed or swapped.

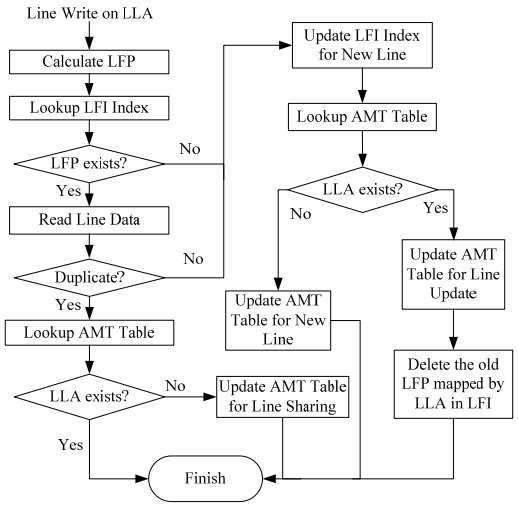

Fig. 2. Line deduplication processing in CARAM.

2.2 Line Deduplication Processing

As shown in Fig. 2, the line deduplication processing of our CARAM is performed in the deduplicator module of the memory controller. When a line write request on a LPA is issued from the LLC, the corresponding line fingerprint LFP is calculated in CPU using weak hashing SuperFastHash. Then it queries the LFI in DRAM to check whether its line fingerprint exists or not. If yes, it further reads and compares the data in PLA line with the writing line data, if it is duplicate, we can find the mapping of LLA to the same PLA in the AMT, and if it exists in the table, we can deduce that the line write is a duplicate request and drop it; otherwise, the write operation is a duplicate line write, and it updates the AMT for line sharing in hybrid memory. On the other hand, if the line fingerprint LFP does not exist in the LFI or not a duplicate, which means a new line is issued, and it needs to add the new LFP and its metadata into the

LFI after the line write is finished in the hybrid main memory. Then it also queries the LLA in the AMT. If it is found, then it updates the AMT for the line update after line edition or page swapping, and deletes the old LFP mapped by LLA in the PFI; Otherwise, it means a new line write is issued, and adds the new LLA into the AMT.

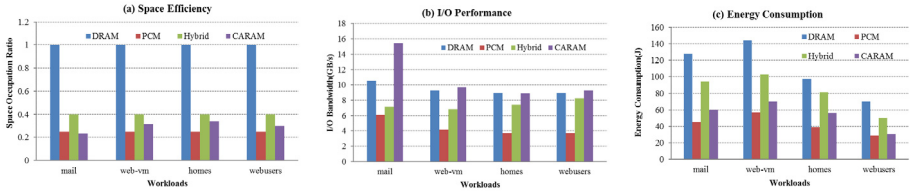

Fig. 3. The experiment results of various main memory architecture.

3 Evaluation

We build a trace-driven hybrid memory simulator based on DRAMsim2[10] for our studies. To support the heterogeneous design, we use multiple channels to simulate DRAM and PCM. We calculate the performance and energy metrics by referring to the energy and performance models used in [2]. There are two channels in the current version: one channel for 8GB PCM, and another channel for 2GB DRAM. We implement the deduplicator module to enable deduplication in hybrid main memory by modifying some components in DRAMsim2 to support simulation for PCM memory. We can easily modify the simulator to support pure DRAM, pure PCM, and the naïve hybrid memory simulation for comparisons. We assume that the byte price of DRAM is four times that of PCM, and compare CARAM with the three kinds of main memory configuration with the same cost: 4GB DRAM(DRAM), 16GB PCM(PCM), and the hybrid 2GB DRAM+8GB PCM memory (Hybrid). We feed the simulator with modified I/O traces down-stream of an active cache from four types of application systems: a mail VM server(mail), a web VM server for online course(web-vm), a file server(homes) and a web server for personal pages of users(web-users), in the CS department of FIU[11]. The four workloads were obtained by adding hash calculation time with 1 byte/cycle on each volume as a downstream of a last-level cache. It records the same line fingerprint for all 256B lines from every 4 KB block, and the fingerprint value is the first 4B value of the MD5 value of the block.

As shown in Fig. 3, we evaluate various main memory architectures in terms of space efficiency, I/O performance, and energy consumption. Limited main memory size is the primary bottleneck for in-memory computing. Here, we assume the space occupation ratio of DRAM is 1 in all four applications. As shown in Fig. 3(a), the ratio of PCM is very low, and it has only a quarter of DRAM's value due to its low unit price. The ratio of the naïve hybrid memory is more than 0.4, while CARAM can improve it to approach or even better than that of PCM, by saving 15%~42% memory space of hybrid main memory systems via deduplication.

CARAM enables deduplication in hybrid main memory, and it can significantly reduce line writes to enhance I/O performance through line sharing. Fig. 3(b) shows that CARAM can achieve higher performance range from 13% to 116% than the naïve hybrid main memory system, since it performs a large number of low-overhead metadata updates only instead of duplicate line writes. Also, it performs the best under the mail workload mainly due to its lowest space occupation ratio.

We evaluate the energy consumption of the four memory architectures with the I/O traces first, but their values are almost the same under the four different workloads due to the domination of idle time. To differentiate these schemes, we stress-test their energy consumption by continuously issuing the read or write line request without the greatest common idle intervals. Results in Fig. 3(c) show that our CARAM can save 31%~38% energy consumption than that of the naïve hybrid main memory, since it can significantly reduce the number of write operations in main memory.

4 Conclusions

In this paper, we present a content aware hybrid DRAM/PCM main memory system, called CARAM, by exploiting line sharing with deduplication technique, and implement it in a trace-driven hybrid memory simulator based on DRAMsim2. Specifically, we introduce line-level deduplication processing of write access in our hybrid structure to balance space saving and system performance. Evaluation results show that CARAM constantly outperforms the existing hybrid memory systems in terms of space saving, I/O bandwidth, and power consumption. We will study the combination of deduplication and memory compression running real data as a direction of future work.

Acknowledgments

This research was supported by the NSF-Jiangsu grant BK20191327. We would like to thank Prof. Patrick P.C. Lee, the Chinese University of Hong Kong for his help on the initial design of the system.

References

1. Lee, B.C., Zhou, P., Yang, J., et al: Phase change technology and the future of main memory. IEEE Micro, 2010, 30(1):131-141.
2. Lee, H., Baek, S., Nicopoulos, C., et al: An energy-and performance-aware DRAM cache architecture for hybrid DRAM/PCM main memory systems. IEEE ICCD2011, pp.381-387.
3. Ham, T.J., Chelepalli, B.K., Xue, N., et al: Disintegrated control for energy-efficient and heterogeneous memory systems. IEEE HPCA 2013, pp. 424−435.
4. Qureshi, M.K., Srinivasan, V., Rivers, J.A.: Scalable high performance main memory system using phase-change memory technology. IEEE ISCA, 2009, 37(3), pp.24-33.
5. Dhiman, G., Ayoub, R., Rosing, T.: PDRAM: a hybrid PRAM and DRAM main memory system. IEEE DAC, 2009, pp. 664-669.
6. Ramos, L., Gorbatov, E., Bianchini, R.: Page placement in hybrid memory systems. International Conference on Supercomputing, 2011:85-95.
7. Lee, S., Bahn, H., Noh, S.H.: CLOCK-DWF: a write-history-aware page replacement algorithm for hybrid PCM and DRAM memory architectures. IEEE Trans. on Computers, 2014, 63(9):2187-2200.
8. Wang, Q., Li, J., Xia, W., et al: Austere flash caching with deduplication and compression. USENIX ATC 2020, pp.713-726.
9. Baek, S., Lee, H.G., Nicopoulos, C., et al: Designing hybrid DRAM/ PCM main memory systems utilizing dual-phase compression. ACM TODAES, 2014, 20(1):1-31.
10. DRAMsim2. https://github.com/ericlove/DRAMSim2.
11. FIU Traces. http://iotta.snia.org/traces/390.
12. Hsieh, P.: The superfasthash function. http://www.azillionmonkeys.com/qed/hash.html. 2004.