Vol.2017-ARC-224 No.33
Vol.2017-SLDM-178 No.33
2017/1/25

Near Memory Processing on Hybrid Memories (Non-Refereed Workshop Manuscript)

Eishi Arima[1,2,3,a)]

Abstract: The data transfer between memories and processors is a major performance/power bottleneck for current computer systems. Particularly, the bottleneck is more severe while executing workloads who produce less regular and more frequent accesses to larger memory space. To overcome this problem, near memory processing that off-loads data processing tasks to the logic close to memories was proposed. However, that is not optimized accordingly for hybrid memory system, which is one of the best technique to sustain increasing demand for larger memory space with minimal performance impact. Therefore, this report explores near memory processing functionalities on hybrid main memories for several data intensive workloads.

Acknowledgments I would like to thank Dr. Maya Gokhale, Dr. Scott Lloyd and Dr. Martin Schulz at Lawrence Livermore National Laboratory for their assistance. Part of this work is performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This work is partly supported by MEXT Grant-in-Aid for Research Activity Start-up (JP16H06677) and Research on Processor Architecture, Power Management, System Software and Numerical Libraries for the Post K Computer System of RIKEN.

1. Introduction

The performance/power overheads of communication between memories and processors are very critical for modern computer systems. Because the off-chip latency/bandwidth are long/limited, frequent accesses to main memory can cause severe performance degradation. Moreover, it is reported that the power consumption of data transfer between them is dominant in total power consumption for several systems [1].

Particularly, the overheads are quite critical while executing modern workloads that produce irregular and frequent accesses to large memory space. The reason for this is conventional cache hierarchies do not work well for such workloads. Firstly, data are placed on caches based on spatial/temporal access localities to efficiently use their limited capacity and thus such workloads cause very frequent cache misses, which significantly affects performance. Secondly, even the data that are referenced only once are usually placed on caches, thus they waste significant amount of energy for such workloads.

To overcome this problem, near memory processing, also known as processing-in-memory, was proposed [2] and has been studied [3], [4], [5], [6], [7], [8], [9]. The key idea of it is off-loading data processing like data rearrangement tasks to logic which is close to memories. By doing so, it is possible to reduce the number of data transfer between processors and memories, which is going to reduce waste of energy and improve performance. Recently, emerging 3D-stacked memories such as HBM (High Bandwidth Memory) [10] and HMC (Hybrid Memory Cube) [11] began to spread rapidly in many systems. In these memory technologies, multiple memory layers are stacked on a logic layer and tightly connected each other by TSV (Through Silicon Via) technique. Therefore, it is practical to use the logic layer for the near memory processing purpose.

However, that is not well-studied for a hybrid memory system, which is one of the best way to sustain increasing demand for larger memory space [12], [13], [14], [15]. A hybrid memory system consists of two regions; one is faster but smaller region, the other is slower but larger one. By adopting emerging non-volatile memories like Phase Change RAM (PCRAM) for the larger region and optimizing data placement between them, it is possible to sustain quite large memory space with very small performance overhead [13], [14], [15]. Thus, future memory systems should be hybrid.

Therefore, this report explores near memory processing functionalities on the hybrid memory systems for data intensive workloads. More specifically, it points out the research opportunities on how to cope with several tasks like "copy and rearrangement" and "pointer chasing" on such memory systems.

The rest of this report is organized as follows. In the next section, this report describes the concept of near memory processing and hybrid memory systems. In section3, this report explains the

[1] The University of Tokyo, Tokyo, Japan
[2] Lawrence Livermore National Laboratory, CA, USA
[3] RIKEN AICS, Hyogo, Japan
[a)] arima@cc.u-tokyo.ac.jp

This is an early-stage non-refereed technical report and should not preclude later publication at any journals, conferences, symposiums, etc.

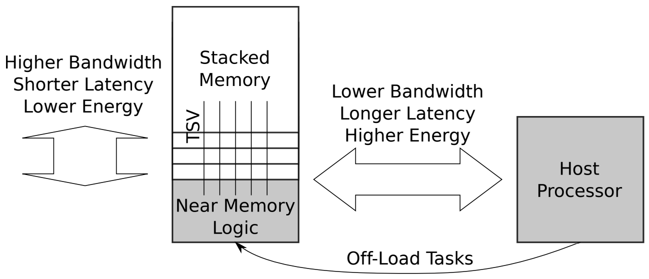

Fig. 1 Concept of near memory processing

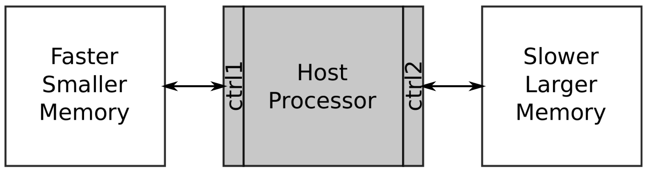

Fig. 2 A hybrid memory system

problem settings for some data intensive workloads and future research plans. In section4, it introduces prior work. In section 5, this report concludes the work.

2. Background

In this section, this report firstly explains the concept of near memory processing. Then, it describes hybrid memory systems.

2.1 Concept of near memory processing

Fig. 1 shows the concept of near memory processing on a 3D stacked memory. The host processor and the logic layer are connected each other by silicon interposer. Although the bandwidth between host processor and memory is improved by adopting this technology, the latency is still high [12] and the data transfer energy is not negligible for many workloads. On the other hand, the memory chips and the logic layer are connected with TSV technology and the internal bandwidth, latency and data transfer energy of the stack are higher, shorter and lower.

Therefore, it is important to off-load memory processing tasks to the logic layer to decrease the number of data transfer between the host processor and the logic layer. By doing so, it is possible to significantly improve the energy-efficiency of the system for various workloads.

2.2 Hybrid memory systems

To sustain larger memory space with smaller performance impact, hybrid memory systems were proposed. In general, a hybrid memory system consists of two regions as shown in **Fig. 2**; one is faster but smaller region, the other is slower but larger one. By optimizing data placement between them, it is possible to suppress performance degradation caused by using large but slow memory.

The hybrid main memories are adopted in recent systems. Intel Knights Landing processors support hybrid main memories which consist of 3D stacked MCDRAMs and conventional DDR4 DIMMs [12]. Because emerging non-volatile RAMs such as PCRAM are much more denser than conventional DRAM, thus non-volatile RAM based hybrid main memory will be adopted in the future systems to significantly increase the total main memory capacity.

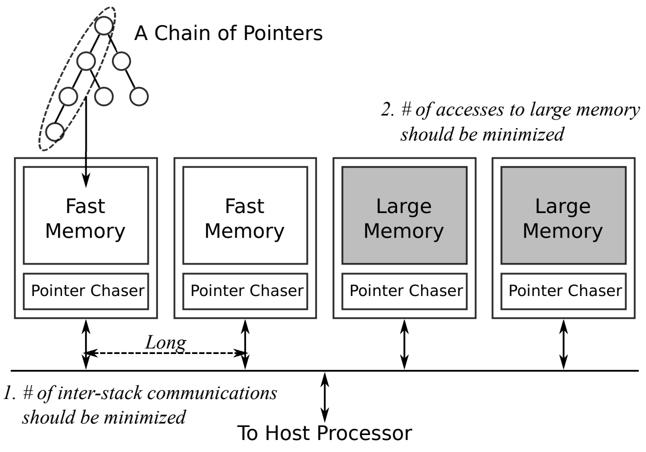

Fig. 3 Optimizing pointer chasing

3. Problems and future plans

In this section, this report firstly clarifies the problems on near data processing for hybrid main memory systems. Then, it describes future plans.

3.1 Pointer chasing

Pointers are ubiquitous in various data structures such as lists, trees and hash tables. Thus, pointer chasing is a widely-used task in various workloads including in-memory databases like Memcached [16], graph processing [17] and so on. However, because the task is very memory intensive, it also incurs very frequent data transfer between processors and memories when the data size is much lager than that of caches.

To overcome this problem, accelerating pointer chasing with near memory logic is regarded as a promising approach [9]. More specifically, in the technique, a host processor off-loads the pointer chasing tasks to the logic and receive only the necessary values that can be gotten after chasing pointer chains. By doing so, the number of data transfer between them can be significantly reduced.

However, the approach does not work well if it is applied to the systems that have multiple and hybrid stacks like **Fig. 3**. Therefore, several optimizations should be applied to the approach. Firstly, the number of data transfer between stacks should be minimized. This is because an inter-stack data movement takes much longer time and consumes much larger energy than intra-stack one. However, if a chain of pointers is distributed to different stacks, pointer chasing on the chain incurs multiple inter-stack communications. Secondly, the number of accesses to large but slow memory such as PCRAM should be minimized. This is because the access latency/energy of non-volatile memories are much longer/larger than conventional DRAM, thus frequent accesses to them significantly degrades/increases the performance/power of the system. Thus, it is also important to find out frequently accessed pointer chains and place them on DRAM stacks.

3.2 Copy and rearrangement

Some of current HPC systems support both DIMM based large memories and 3D stacked RAM based memories like **Fig. 4**.

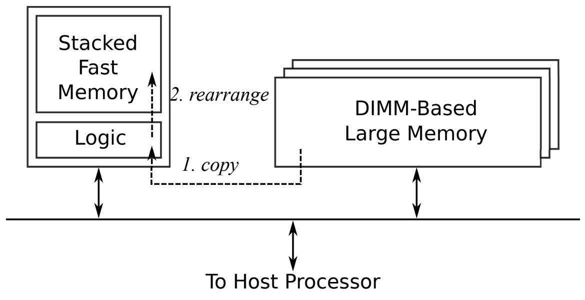

Fig. 4 Copy and rearrangement

This is because DIMM based memories enable users to flexibly change the configuration of memory systems and also the cost of a DIMM is lower than using silicon interposer based one. Actually, recent Intel Knights Landing processors support both of them as described before.

In such memory systems, the data placement between these two different technologies is more important for system performance. In Knights Landing based systems, copying a bunch of data from slower memories to faster memories just before it is referenced is an often used approach. This technique is usually implemented with special threads that are dedicated to data copy. However, inefficient data transfer between processor cores and memories is necessary for the implementation. Also, such data are stored in caches which can cause cache conflicts and thus may degrade/increase performance/power.

Therefore, this report considers data copy between the memories by using only the near memory logic as shown in the figure. By doing so, the host processor is not used for the data movements and thus the number of cache accesses/conflicts in it can be significantly reduced.

In addition, the report also considers data rearrangement in the near memory logic just before/after sending/receiving data. One application example of this technique is matrix-vector multiplication for very large sparse matrices with CSR (Compressed Sparse Row) structure [18]. The task requires gather/scatter accesses to very large vectors whose elements are mostly stored in the large memory. Therefore, it is effective to fetch a part of a vector from the large memory just before referenced and gather it in the logic and send it to the faster memory is promising. The logic can also be used any data arrangement tasks such as matrix transpose, shuffle and data format change.

Such copy and rearrangement tasks can also be implemented in current Knights Landing systems by programming the special threads which are dedicated to the tasks as described above. So, the author is going to implement them for certain applications, confirm the effectiveness and then evaluate how much efficiency can be improved by off-loading such tasks to the near memory logic.

3.3 Future plans

The future plans of this work are to define the problems in more detail and propose solutions for the problems and quantitatively prove the effectiveness of the solutions. In the workshop presentation, the author is going to talk about the problem definitions, the solutions and the results of preliminary evaluation for one or both of what the report introduced in this section.

4. Related work

In this section, this report firstly describes prior work about near data processing. Then, it introduces related work about hybrid memories.

4.1 Near memory processing

The idea of near memory processing firstly appeared in 1970 [2]. After that, several papers proposed the detailed designs based on the concept in 1990s [3]. But, the industry did not adopt the concept because it took quite high cost to integrate computational units inside memory dies.

However, the situation has changed after the industry decided to adopt 3D stacking memory technologies for sustaining increasing demand for larger memory without scaling DRAM cells. Because a stacked memory have a logic layer under memory layers, it suits for near dada processing.

Therefore, several papers proposed some functionality for near data processing on the logic layer. Seshadri et al. proposed RowClone that tries to process bulk data copy and initialization within memory [4]. Akin et al. proposed in-memory data reorganization technique for common operations including shuffle, pack/unpack, swap, transpose, and layout transformations [5]. Gokhale et al. demonstrated the effectiveness of in-memory gather/scatter technique with their emulation infrastructure for it [6]. Ahn et al. proposed an architecture of the logic layer to execute parallel graph processing efficiently [7]. Mirzadeh et al. analyzed and characterized a Join workload for near-data execution and clarified that algorithms and microarchitecture should be revisited [8]. Hsieh et al. proposed in-memory accelerator for pointer chasing workloads [9].

However, these papers did not consider solutions for hybrid main memories. In contrast to the prior work, this report explores functionalities of near memory processing for hybrid main memories. As far as I know, this is the first work which attempts to tackle that.

4.2 Hybrid main memories

Hybrid main memories are widely studied and adopted in latest systems. Intel Knights Landing based systems support hybrid main memories which consist of 3D stacked MCDRAMs and conventional DDR4 DIMMs [12]. On the other hand, Dhiman et al. proposed non-volatile memory based hybrid main memory [13]. Then, Ramos et al. proposed memory controller based page placement technique for such memory systems [14]. Yoon et al. proposed row buffer locality aware page placement algorithm based on the technique [15].

Although these techniques were proved to be effective, they do not consider near data processing for hybrid main memories. By adopting it, it is possible to improve the energy-efficiency of such memory systems.

5. Conclusion

It is critical to mitigate the memory access overheads for modern computer systems to efficiently execute memory-intensive applications. To this end, near memory processing that off-loads data processing tasks to the logic close to memories were proposed. However, it is not well studied for hybrid memory systems. Thus, this report considered several near memory processing functionalities for such systems. The future plans of the work are to propose detailed implementation and to quantitatively prove the effectiveness of our proposal.

References

[1] Dally, W.: Challenges for Future Computing Systems, *HiPEAC Keynote* (2015).

[2] Stone, H. S.: A Logic-in-Memory Computer, *IEEE Transactions on Computers*, Vol. 100, No. 1, pp. 73–78 (1970).

[3] Gokhale, M., Holmes, B. and Iobst, K.: Processing in Memory: The Terasys Massively Parallel PIM Array, *Computer*, Vol. 28, No. 4, pp. 23–31 (1995).

[4] Seshadri, V., Kim, Y., Fallin, C., Lee, D., Ausavarungnirun, R., Pekhimenko, G., Luo, Y., Mutlu, O., Gibbons, P. B., Kozuch, M. A. et al.: RowClone: Fast and Energy-Efficient in-DRAM Bulk Data Copy and Initialization, *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, ACM, pp. 185–197 (2013).

[5] Akin, B., Franchetti, F. and Hoe, J. C.: Data Reorganization in Memory Using 3D-Stacked DRAM, *ACM SIGARCH Computer Architecture News*, Vol. 43, No. 3, ACM, pp. 131–143 (2015).

[6] Gokhale, M., Lloyd, S. and Hajas, C.: Near Memory Data Structure Rearrangement, *Proceedings of the 2015 International Symposium on Memory Systems*, ACM, pp. 283–290 (2015).

[7] Ahn, J., Hong, S., Yoo, S., Mutlu, O. and Choi, K.: A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing, *ACM SIGARCH Computer Architecture News*, Vol. 43, No. 3, ACM, pp. 105–117 (2015).

[8] Mirzadeh, N., Kocberber, O., Falsafi, B. and Grot, B.: Sort vs. Hash Join Revisited for Near-Memory Execution, *5th Workshop on Architectures and Systems for Big Data (ASBD)*, No. EPFL-CONF-209121 (2015).

[9] Hsieh, K., Khan, S., Vijaykumar, N., Chang, K. K., Boroumand, A., Ghose, S. and Mutlu, O.: Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation, *Computer Design (ICCD), 2016 IEEE 34th International Conference on*, IEEE, pp. 25–32 (2016).

[10] Standard, J.: High Bandwidth Memory (HBM) DRAM, *JESD235* (2013).

[11] Consortium, H. M. C. et al.: Hybrid Memory Cube Specification 2.1, *Last Revision Jan* (2015).

[12] Reinders, J., Jeffers, J. and Sodani, A.: Intel Xeon Phi Processor High Performance Programming Knights Landing Edition (2016).

[13] Dhiman, G., Ayoub, R. and Rosing, T.: PDRAM: A Hybrid PRAM and DRAM Main Memory System, *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*, IEEE, pp. 664–669 (2009).

[14] Ramos, L. E., Gorbatov, E. and Bianchini, R.: Page Placement in Hybrid Memory Systems, *Proceedings of the international conference on Supercomputing*, ACM, pp. 85–95 (2011).

[15] Yoon, H., Meza, J., Ausavarungnirun, R., Harding, R. A. and Mutlu, O.: Row Buffer Locality Aware Caching Policies for Hybrid Memories, *Computer Design (ICCD), 2012 IEEE 30th International Conference on*, IEEE, pp. 337–344 (2012).

[16] Fitzpatrick, B.: Distributed caching with memcached, *Linux journal*, Vol. 2004, No. 124, p. 5 (2004).

[17] Murphy, R. C., Wheeler, K. B., Barrett, B. W. and Ang, J. A.: Introducing the graph 500, *Cray Users Group (CUG)* (2010).

[18] Saad, Y.: SPARSKIT: A Basic Tool Kit for Sparse Matrix Computations (1990).