Cache memory organization for processing in memory

Young-Kyu Kim¹, Byungin Moon², Dong-Sun Kim¹, and Young-Jong Jang¹a)

Abstract A promising solution for assuring ultra-low latency in data-intensive application processing systems is processing in memory (PIM). Although most studies that have examined PIM-based computing systems have used cache memory, few have adequately explored a reasonable cache management policy for PIM. Therefore, this paper studies cache management policies for PIM-based computing systems and classifies existing PIM policies according to where they are located and how they are managed. To evaluate the policies, we model three types of PIM-based computing systems used in an in-memory system architecture. One model employs an internal-single cache, another an external cache hierarchy, and the other internal multiple cache-based PIM. We also simulate the performance and power consumption of the three models by their workloads, each with diverse characteristics. The experimental results show how cache policies influence the performance and power of PIM-based in-memory computing systems.

Keywords: processing in memory, PIM, data-centric computing, near-data processing, cache management, cache architecture

Classification: Integrated circuits

1. Introduction

Big data have influenced the development of high-performance computing systems that support ultra-low latency services and real-time data analytics. The trend toward big data has involved other computing paradigms such as data-centric computing, near-data processing, and processing in memory (PIM) [1]. In particular, big-data processing technologies using in-memory computing technologies have entered the spotlight. An in-memory computing system drastically increases system performance because it holds entire datasets in the dynamic random access memory (DRAM)-based main memory of a computer system without disks and excludes overheads for accessing the slow mechanical hard drive I/O [2]. With such a system, numerous PIM research groups in both academia and industry have produced a number of promising results. The system is based on three-dimensional integrated circuit technology [3].

In recent years, PIM has been used in in-memory computing for processing large data-intensive applications such as machine learning, graph processing, and image-processing algorithms [4, 5, 6]. Unfortunately, while several in-memory computing systems have become a reality through commercialization, PIM has not, for several critical limitations must still be overcome [4, 7]. Specifically, we are interested in overcoming the performance bottleneck caused by processing PIM atomic operations. This paper, therefore, studies how cache management policies influence the performance and power of PIM-based computing systems. We classify existing cache policies for PIM into two types and propose a new cache management system for PIM. We also graft PIM onto an in-memory computing system in order to model and simulate the performance and power consumption of systems using PIM with various cache management policies.

2. Cache management policies for PIM

Although a typical PIM consists of a processing unit (PU), a DRAM controller, and at least one more DRAM, recent PIM proposals have not questioned the necessity of using a cache for PIM [5, 6, 8, 9, 10, 11]. Existing cache architectures for PIM may be classified under two large groups, one inside of PIM [5, 10, 11] and one outside [6, 8, 9]. Fig. 1 presents a simplified system block diagram based on a PIM subsystem.

Fig. 1. Simplified system block diagrams based on a PIM subsystem
referred to the cache management policies for PIM as an external cache hierarchy (ECH) and an internal single cache (ISC). Unfortunately, the cache architectures for PIM induce a performance bottleneck that results from processing PIM operations, which comprise a read-modify-write scheme based on atomic operations and require consecutive memory accesses in PIM [4]. Consequently, existing cache architectures such as the ECH and the ISC are not suitable for PIM. To overcome this drawback, we conduct simulations to find a reasonable cache management policy and propose an internal-multiple cache (IMC).

The proposed IMC-based PIM, illustrated in Fig. 2, consists of a packet parser, a PU, a cache array, and a DRAM. The packet parser receives PIM packets from the PMU, unpacks them for extracting PIM operations, and passes them to the PU, which processes them. To accomplish this process, the PU must repeatedly access the DRAM for either loading operands or writing the processing results. The location of the DRAM controller located in the PU is only one example. The cache array has at least one more transparent cache for the DRAM. Since the cache array is physically or logically separated inside a multi-port architecture [12, 13, 14], it allows access to the memory interleaving scheme. Therefore, the destination and source addresses for the PIM operations can be simultaneously accessed. As a result, enabling parallel memory access within PIM mitigates the PIM atomic operation overhead. In addition, since the hardware resource of each cache module in the IMC is much smaller than those of the ISC and ECH, the power consumption required for accessing the cache decreases [15]. The IMC, however, is still limited by the cache hit-ratio issue, particularly if the addresses in PIM operations consecutively point to the same cache of the IMC, which would exacerbate system performance [12]. To avoid this problem, the address mapping scheme for the IMC must be rearranged, as shown in Fig. 3.

The packet for the PIM consists of a packet head, an opcode, operands, and a packet tail. The packet head includes information about the packet, such as the packet number, the packet length, and the PIM identifier (ID). The packet tail indicates the end of the packet, and the opcode is an operation code processed by the PU in the PIM. Depending on the opcode, operands will become either the immediate data or the physical address of the PIM. The PIM physical address maps to several sub-fields such as the tag, the index, the internal cache ID, and offset fields for checking the IMC. The tag, the index, and offset fields are similar to those of a traditional cache. The physical address, however, includes an internal cache ID that matches one cache of the IMC in the current working PIM. Operands in the PIM packet must be stored separately in the IMC. After all, if operands in the PIM packet reside in just one cache of the IMC, sequential cache accesses will be incurred. Thus, as lower bits of a physical address with typical computer systems are relatively more variable than higher bits, this paper proposes using lower bits of the physical address as the internal cache ID field for the IMC [16].

3. Modeling and workloads

To evaluate the performance and the power of the IMC-based system, we use the C programming language to separately model three types of in-memory computing systems with a simplified approach because the system has only one memory subsystem without disk-based sub-memory. The three models are the IMC model, the ISC model, and the ECH model, illustrated in Figs. 4–6, respectively. Each model is a multi-processor-based system with sixteen processors. In addition, since we are in an era of the terabyte-sized hard disk or solid-state memory [1, 17], targeted PIM-based in-memory computing systems deserve to have sufficient capacity. Thus, each model possesses 1,024 PIMs, each with a 4-GB DRAM. The size of the main memory is four terabytes.
In the IMC model, a PIM has four 16-kB caches. In the ISC model, a PIM has only one 64-kB cache. The IMC and ISC models are managed in eight-way set associative mapping and a private cache. In the ECH model, PIMs have no cache memory, but each processor in the ECH model has level 1 (L1) and level 2 (L2) caches, and a last level cache (LLC) is situated between the processors and the PMU. The L1 and L2 caches in the ECH model are 64 kB and 512 kB private caches, respectively, and mapped as 8- and 16-way, respectively. The LLC, with a capacity of 64 MB, is used as a shared and a 32-way set associative cache. We also assumed and modeled the two-level caches integrated in processors to be exclusive and the relationship between the two-level caches and the LLC to be inclusive [18]. As Al-Zoubi et al. [19] revealed no common wisdom about the best algorithm to use as a replacement policy for the caches of the three models, we use a pseudo least recently used (PLRU) algorithm, one of the options among the various algorithms for cache block replacement. The selection of the PLRU algorithm allowed us to exclude overheads from the simulations. The size of the cache line in each model is 64 bytes. We configured the parameters of the simulation as shown in Table I. We used CACTI 6.5 [20] to estimate the access times and the energies of the caches in each model and the MICRON DDR4-2400 specification [21] to calculate those of the DRAM.

For experiments with large-capacity in-memory computing systems, we prepared sixteen workloads, extracted from selected applications among the SPEC CPU2006 and SPEC OMP2012 benchmarks by Pin [22, 23, 24], shown in Table II. We examined the selected sixteen applications in terms of their memory characterization and memory footprint size for generating workloads [25, 26, 27, 28].

Each workload consisted of 12 billion packets. Since addresses in the packets were made by a 64-bit operating system (OS), they are 48-bit virtual addresses. The addresses were assumed and used as physical memory addresses for PIMs.

In the IMC model, a PIM has four 16-kB caches. In the ISC model, a PIM has only one 64-kB cache. The IMC and ISC models are managed in eight-way set associative mapping and a private cache. In the ECH model, PIMs have no cache memory, but each processor in the ECH model has level 1 (L1) and level 2 (L2) caches, and a last level cache (LLC) is situated between the processors and the PMU. The L1 and L2 caches in the ECH model are 64 kB and 512 kB private caches, respectively, and mapped as 8- and 16-way, respectively. The LLC, with a capacity of 64 MB, is used as a shared and a 32-way set associative cache. We also assumed and modeled the two-level caches integrated in processors to be exclusive and the relationship between the two-level caches and the LLC to be inclusive [18]. As Al-Zoubi et al. [19] revealed no common wisdom about the best algorithm to use as a replacement policy for the caches of the three models, we use a pseudo least recently used (PLRU) algorithm, one of the options among the various algorithms for cache block replacement. The selection of the PLRU algorithm allowed us to exclude overheads from the simulations. The size of the cache line in each model is 64 bytes. We configured the parameters of the simulation as shown in Table I. We used CACTI 6.5 [20] to estimate the access times and the energies of the caches in each model and the MICRON DDR4-2400 specification [21] to calculate those of the DRAM.

For experiments with large-capacity in-memory computing systems, we prepared sixteen workloads, extracted from selected applications among the SPEC CPU2006 and SPEC OMP2012 benchmarks by Pin [22, 23, 24], shown in Table II. We examined the selected sixteen applications in terms of their memory characterization and memory footprint size for generating workloads [25, 26, 27, 28].

Each workload consisted of 12 billion packets. Since addresses in the packets were made by a 64-bit operating system (OS), they are 48-bit virtual addresses. The addresses were assumed and used as physical memory addresses for PIMs.

In the IMC model, a PIM has four 16-kB caches. In the ISC model, a PIM has only one 64-kB cache. The IMC and ISC models are managed in eight-way set associative mapping and a private cache. In the ECH model, PIMs have no cache memory, but each processor in the ECH model has level 1 (L1) and level 2 (L2) caches, and a last level cache (LLC) is situated between the processors and the PMU. The L1 and L2 caches in the ECH model are 64 kB and 512 kB private caches, respectively, and mapped as 8- and 16-way, respectively. The LLC, with a capacity of 64 MB, is used as a shared and a 32-way set associative cache. We also assumed and modeled the two-level caches integrated in processors to be exclusive and the relationship between the two-level caches and the LLC to be inclusive [18]. As Al-Zoubi et al. [19] revealed no common wisdom about the best algorithm to use as a replacement policy for the caches of the three models, we use a pseudo least recently used (PLRU) algorithm, one of the options among the various algorithms for cache block replacement. The selection of the PLRU algorithm allowed us to exclude overheads from the simulations. The size of the cache line in each model is 64 bytes. We configured the parameters of the simulation as shown in Table I. We used CACTI 6.5 [20] to estimate the access times and the energies of the caches in each model and the MICRON DDR4-2400 specification [21] to calculate those of the DRAM.

For experiments with large-capacity in-memory computing systems, we prepared sixteen workloads, extracted from selected applications among the SPEC CPU2006 and SPEC OMP2012 benchmarks by Pin [22, 23, 24], shown in Table II. We examined the selected sixteen applications in terms of their memory characterization and memory footprint size for generating workloads [25, 26, 27, 28].

Each workload consisted of 12 billion packets. Since addresses in the packets were made by a 64-bit operating system (OS), they are 48-bit virtual addresses. The addresses were assumed and used as physical memory addresses for PIMs.

In the IMC model, we did not account for IMC-access scheduling among the received physical addresses, which would require additional hardware resources such as a received-address buffer, scheduling logic, and a re-ordering buffer. In addition, as the use of a scheduling algorithm would impact IMC model performance, the IMC model in this paper sequentially processed the received physical addresses, illustrated in Fig. 7. Thus, the degree of accessibility of the multiple cache is influenced by the address locality of the internal-cache-ID field, including the source or destination addresses for PIM applications. We fixed the
internal cache ID field to the middle two bits between the tag field and the offset field according to the experimental results of the address mapping, shown in Fig. 8. For the sake of simplicity, however, we omitted detailed descriptions of the experiments, which we left for future work.

| MSB | Tag | Upper bit index | Internal cache ID | Lower bit index | Offset |
|-----|-----|-----------------|-------------------|-----------------|-------|

Destination or source address in the PIM packet

**Fig. 8.** The address mapping scheme for the IMC model

### 4. Simulations and results

Because the simulations in this paper focused on the use of PIM-based in-memory computing systems for high-performance computing, we overburdened the simulations as much as possible. Accordingly, each of the three modeled systems executed exactly sixteen workloads, each of which was executed by sixteen processors in each model. Thus, the workloads were processed in parallel on multiple processors. We determined the processor scheduling according to the OS executing the simulation and implemented multi-processor synchronization on a mutual exclusion object function [29, 30] for shared resources such as the LLC and DRAMs. These multi-thread-based simulations ensure that simulation conditions, such as the natural contentions of shared resources and a large memory load, are similar to those of real systems.

In this paper, we counted the hits and misses of the internal and external caches of the modeled systems during the processing of the workload packets and then calculated the access time and power consumption of the caches and DRAMs in each system for processing all of the packets of each workload. A workload in this simulation has twelve billion packets, two billion of which are dedicated for initializing the caches. Thus, we used only ten billion packets to measure the performance and power consumption of the system, avoiding phase-incurred intensive-compulsory misses in the caches. Figs. 9–11 show the simulation results of the ECH, ISC, and IMC models.

**Fig. 9.** Total times for processing all packets in each workload

**Fig. 10.** Total number of cache misses in each model for the simulation

**Fig. 11.** Total power consumption for the completion of all workloads

models incurred similar numbers of cache misses, such as the bzip, GemsFDTD, hmmer, lbm, mcf, mile, and omnetpp, but the IMC model outperformed the ISC model because the cache-access time of the IMC model was overlapped by the proposed IMC.

A comparison of the power consumption required to process all of the workloads in each model is shown in Fig. 11. The simulation results showed that the number of L1 cache misses of the ECH model was similar to the number of L1 cache misses of the ISC and ISC models, but the ECH model employed redundant hardware resources for the L2 cache and LLC and consumed more power for the caches than the ISC and IMC models. Thus, the power consumption of the ECH model was much greater than that of the ISC and IMC models. Compared to the workloads of the ISC and IMC models, several workloads of the ECH model such as the GemsFDTD, omnetpp, and swim, consumed a great deal of power. After all, the workloads in the ECH model incurred many more LLC misses than those of the ISC and IMC models; in other words, the workloads in the ECH model consumed much more power for accessing the DDR4 than the other models. Since the counts of cache accesses in the ISC model were approximately equal to those of the IMC model, the results of the comparison of the power consumption of these models exhibited a pattern similar to the counts of the cache misses.
The results of the simulation showed that the proposed IMC model outperformed the ISC and ECH models and consumed less power. Although the ISC model also had an internal cache, overall, the cache misses of the ISC model were greater than those of the IMC model, a result of the disparity between the cache management and address mapping schemes of these models; and, since the ECH model had an external cache hierarchy, which lengthened the processing time, it consumed a great deal more power than the IMC model.

5. Conclusions

This study explored how cache management policies influence the performance and power consumption of PIM-based systems. To verify the cache management policies for PIM, we modeled and simulated ECH, ISC, and a proposed IMC-based in-memory computing system. The simulations revealed the superiority of the proposed IMC-based system in terms of performance and power consumption. Although this paper studied cache management for a hypothetical PIM-based system, we expect to contribute to the design of a PIM subsystem in future work and to implement an actual PIM-based system that will involve a compiler and programming model.

Acknowledgments

This work was financially supported by the Industrial Strategic Technology Development Program of MOTIE/KEIT (10052653).

References

[1] P. Siegl, et al.: “Data-centric computing frontiers: A survey on processing-in-memory,” MEMSYS Symp. (2016) 295 (DOI: 10.1145/2989081.2989087).
[2] H. Zhang, et al.: “In-memory big management and processing: A survey,” IEEE Trans. Knowl. Data Eng. 27 (2015) 1920 (DOI: 10.1109/TKDE.2015.2427795).
[3] R. Balasubramonian, et al.: “Near-data processing: Insights from a MICRO-46 workshop,” IEEE Micro 34 (2014) 36 (DOI: 10.1109/MM.2014.55).
[4] L. Nai, et al.: “GraphPIM: Enabling instruction-level PIM offloading in graph computing frameworks,” HPCA Symp. (2017) 457 (DOI: 10.1109/HPCA.2017.54).
[5] P. Chi, et al.: “PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory,” ISCA Symp. (2016) 27 (DOI: 10.1109/ISCA.2016.13).
[6] J. Ahn, et al.: “PIM-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture,” ISCA Symp. (2015) 336 (DOI: 10.1145/2749469.2750385).
[7] M. L. Chu, et al.: “High-level programming model abstractions for processing in memory,” Workshop on Near-Data Processing (2013).
[8] D. Zhang, et al.: “TOP-PIM: Throughput-oriented programmable processing in memory,” HPDC Symp. (2014) 85 (DOI: 10.1145/2600212.2600213).
[9] M. Serbak, et al.: “Processing-in-memory: Exploring the design space,” ARCS (2015) 43 (DOI: 10.1007/978-3-319-61086-3.4).
[10] D. Kim, et al.: “Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory,” ISCA Symp. (2016) 380 (DOI: 10.1109/ISCA.2016.41).
[11] A. Borosam, et al.: “Lazy-PIM: An efficient cache coherence mechanism for processing-in-memory,” IEEE Comput. Archit. Lett. 16 (2016) 46 (DOI: 10.1109/LCA.2016.2577557).
[12] S. Niar, et al.: “Comparing multiported cache schemes,” PDPTA 3 (2003) 1179.
[13] K. Huang, et al.: “Curve fitting based shared cache partitioning scheme for energy saving,” IEICE Electron. Express (2018) (DOI: 10.1587/elex.15.20180886).
[14] X. Zhang, et al.: “Towards practical page coloring-based multicore cache management,” EuroSys 9 (2009) 89 (DOI: 10.1145/1519065.1519076).
[15] J. Kim, et al.: “The filter cache: An energy efficient memory structure,” Proc. 30th Annual International Symposium on Microarchitecture (1997) 184 (DOI: 10.1109/MICRO.1997.645809).
[16] Z. Zhang, et al.: “A permutation-based page interleaving scheme to reduce row-buffer conflicts and exploit data locality,” MICRO 33 (2000) 32 (DOI: 10.1145/360128.360134).
[17] K. Therdsteerasukdi, et al.: “Utilizing radio-frequency interconnect for a many-DIMM DRAM system,” IEEE J. Emerg. Sel. Topics Circuits Syst. 2 (2012) 210 (DOI: 10.1109/JETCAS.2012.2193843).
[18] Y. Zheng, et al.: “Performance evaluation of exclusive cache hierarchies,” ISPASS Symp. (2004) 89 (DOI: 10.1109/ISPASS.2004.1291359).
[19] H. Al-Zoubi, et al.: “Performance evaluation of cache replacement policies for the SPEC CPU2000 benchmark suite,” ACM-SE 42 (2004) 267 (DOI: 10.1145/986537.986601).
[20] HP Labs: “CACTI,” (2015) http://www.hpl.hp.com/research/cacti.
[21] Micron Technology, Inc.: “DDR4 SDRAM,” 4gb_ddr4.dram.pdf - Rev. K 06/18 EN.
[22] HP Labs: “CACTI,” (2015) http://www.hpl.hp.com/research/cacti.
[23] Standard Performance Evaluation Corporation: “SPEC CPU2006,” https://www.spec.org/cpu2006/.
[24] Standard Performance Evaluation Corporation: “SPEC OMP 2012,” https://www.spec.org/omp2012/.
[25] Intel Software: “Pin,” https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool.
[26] X. Jin, et al.: “A simple cache partitioning approach in a virtualized environment,” ISPA Symp. (2009) 519 (DOI: 10.1109/ISPA.2009.47).
[27] A. A. Nair and L. K. John: “Simulation points for SPEC CPU 2006,” ICCD (2008) 397 (DOI: 10.1109/ICCD.2008.4751891).
[28] H. Saito, et al.: “Large system performance of SPEC OMP benchmark suites,” Int. J. Parallel Program. 31 (2003) 197 (DOI: 10.1023/A:1023086618401).
[29] M. S. Müller, et al.: “SPEC OMP2012—An application benchmark suite for parallel systems using OpenMP,” IWOMP (2012) 223 (DOI: 10.1007/978-3-642-30961-8.17).
[30] MSDN: “CreateMutex function,” https://docs.microsoft.com/ko-kr/windows/desktop/api/synchapi/nf-synchapi-createmutexa.
[31] MSDN: “WaitForSingleObject function,” https://docs.microsoft.com/ko-kr/windows/desktop/api/synchapi/nt-synchapi-waitforsingleobject.