HAIL-DIMM: Host Access Interleaved with Near-Data Processing on DIMM-based Memory System

Minkyu Lee, Sang-Seol Lee, Kyungho Kim, Eunchong Lee, Sung-Joon Jang*
Korea Electronics Technology Institute, South Korea
{minkyu.lee,sslee81,kyungho.kim,elee,sjjang0626}@keti.re.kr

Abstract

Near-data processing (NDP), a solution to reduce data movement overhead between host and memory, should not interfere with host access to ensure system fairness. We propose a cost-effective and energy-efficient LRDIMM-based NDP architecture (HAIL-DIMM) that can seamlessly interleave NDP and host access and is a drop-in replacement for existing main memory modules. The proposed NDP exploits the interleaving capability of the memory controller to interleave NDP and host access naturally. To take advantage of bank interleaving, an atomic operation of the proposed NDP, which consists of data movement and computation, is recognized by the memory controller as a DDR READ/WRITE but by the HAIL-DIMM as NDP based on the request's address. We implement a prototype of the proposed NDP architecture on an FPGA platform as proof of concept. The evaluation results show that the NDP system achieves up to 2.19x speedup in latency and up to 45.4% energy saving for data movement over the baseline system for memory-bound workloads.

Keywords: Near-data processing, DRAM, dual in-line memory module

1 Introduction

Near-data processing (NDP) is a technique that aims to reduce the data movement overhead of memory-bound workloads, which have low temporal locality and low arithmetic intensity. It can significantly decrease the latency and energy consumption of data movement for memory-bound workloads [2, 10]. In CPU systems, NDP-enabled memory is used as the main memory, serving both memory-bound workloads and traditional workloads. As a result, memory access for NDP and common memory access (i.e., host access) occurs concurrently. However, host accesses may be blocked (e.g., specific ranks or the entire memory) while performing NDP, which can potentially degrade the performance of the host workload. Therefore, the NDP-enabled main memory must interleave NDP and host access.

Previous studies on NDP-enabled memory have focused on DIMM-based memory to serve applications requiring high memory capacity [1, 3–7]. These NDP architectures are based on a Load-Reduced DIMM (LRDIMM) consisting of data buffers (DBs) [11] and a register clock driver (RCD) [14] for ease of architecture application. In LRDIMM-based NDP architectures, a media controller is built on the DIMM to issue a memory request by itself. While the NDP unit accelerates memory-bound workloads, host access is limited or blocked because the internal memory controller manages the memory bus. In addition, previous works have shown that each rank has an independent memory channel, so the host can access ranks that are not used by the NDP unit. LRDIMMs usually have multi-rank DRAM devices (e.g., double die package) with high capacity in a limited PCB area, and the ranks of the DRAM share I/O. Therefore, it is challenging for each rank of an LRDIMM to have an independent memory channel.

This paper proposes a HAIL-DIMM (**H**ost **A**ccess **I**nter**l**eaved with Near-Data Processing on **DIMM**), a cost-effective and energy-efficient LRDIMM-based NDP architecture, that can seamlessly interleave NDP and host access and serve as a drop-in replacement for existing main memory modules. The key idea of HAIL-DIMM is to use the existing memory controller's bank interleaving to interleave NDP and host access. An atomic NDP operation of HAIL-DIMM consists of data fetching from DRAM to DB and computation in DB's NDP unit, which is performed by the DDR READ/WRITE command. It is similar to the atomic operation of a hybrid memory cube (HMC) [8]. From the memory controller perspective, host access and NDP are typical DDR commands. Thus, NDP and host access are naturally interleaved by the memory controller. Additionally, we propose an NDP offloading method for fine-grained NDP operation of HAIL-DIMM by exploiting the CPU's direct memory access (DMA). We implement a prototype of the proposed NDP architecture on an FPGA platform as a proof of concept without memory controller modification. The evaluation results show that the NDP system achieves up to 2.19x speedup in the latency and up to 45.4% energy savings for data movement over the baseline system for memory-bound workloads. In summary, the contributions of this paper include:

- We propose an NDP architecture that supports a fine-grained NDP operation, enabling seamless NDP and host access interleaving.
- We propose a DMA-based NDP offloading technique for the fine-grained NDP operation of HAIL-DIMM.
- We implement the proposed NDP architecture on an FPGA platform containing an ARM processor and evaluate its feasibility and performance.

2 Background and Previous Works

2.1 Main Memory System with LRDIMM

In a CPU system, the memory system consists of independently operating memory channels, and each memory channel consists of one or more dual in-line memory modules

*Corresponding author.

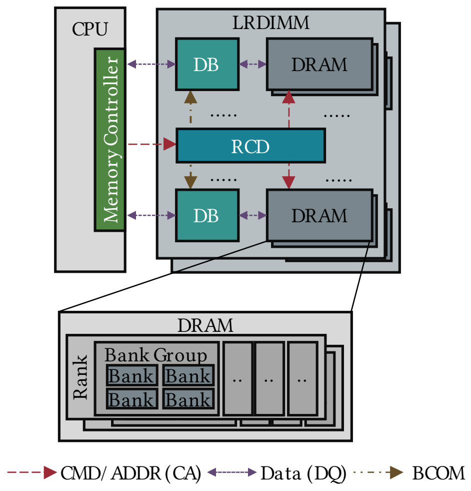

Figure 1. LRDIMM-based memory system. RCD is the register clock driver, DB is the data buffer, and BCOM is the data buffer command bus.

(DIMMs), as depicted in Fig 1. Each channel's memory controller governs the DIMMs within that channel. The DIMMs share a command/address bus (C/A) and a data bus (DQ) in each channel. DIMMs are organized into one or more ranks, with all DRAM devices in each rank operating in lockstep. DRAM devices consist of bank groups, each made up of independently operating DRAM banks.

In server-class CPU systems, load-reduced DIMMs (LRDIMMs) are commonly used to have higher capacity and data rates compared to typical DIMMs (i.e., Unbuffered DIMMs). As shown in Fig 1, an LRDIMM consists of a register clock driver (RCD), data buffers (DBs), and DRAM devices. Upon receiving command/address signals from the memory controller, the RCD forwards these to the DRAM and manages the DBs via the data buffer command (BCOM) bus. Meanwhile, the DBs facilitate data buffering between the memory controller and the DRAM.

The memory controller manages the DIMMs in the memory channel while satisfying the DRAM device's protocol and timing parameters [12, 13]. The memory controller reorders memory requests to increase memory performance. One of the scheduling techniques is a bank interleaving. In order to read data from DRAM, it is necessary to *open* a bank with a long wait time. To mitigate this latency, the memory controller changes the order of memory requests to alternate access to different banks.

2.2 Previous Works

Many previous works proposed DIMM-based NDP architectures to accelerate memory-bound workloads [1, 3–7]. In order to perform memory-bound workloads efficiently, the previous NDP architectures embed a media controller (i.e., memory controller) on DIMM. Consequently, the NDP accelerator on the DIMM can directly read from and write to the DRAM. It also means that the NDP accelerator occupies the memory bus, thereby limiting host access while the accelerator is active.

Each rank on DIMM has an independent memory channel, so NDP and host access can be possible simultaneously by assigning some ranks to the NDP accelerator. However, LRDIMMs are comprised of double-die package (DDP) DRAMs, which integrate two dies (or ranks) onto a single chip to have high memory capacity within a constrained PCB area. In these DRAMs, multiple ranks share a data bus and cannot have physically independent channels. Therefore, the number of independent memory channels on the DIMM is limited, and host access during NDP is also limited. Therefore, previous works are unsuitable for NDP-enabled main memory in CPU systems due to the difficulty of interleaving NDP and host access.

3 HAIL-DIMM

3.1 Overview

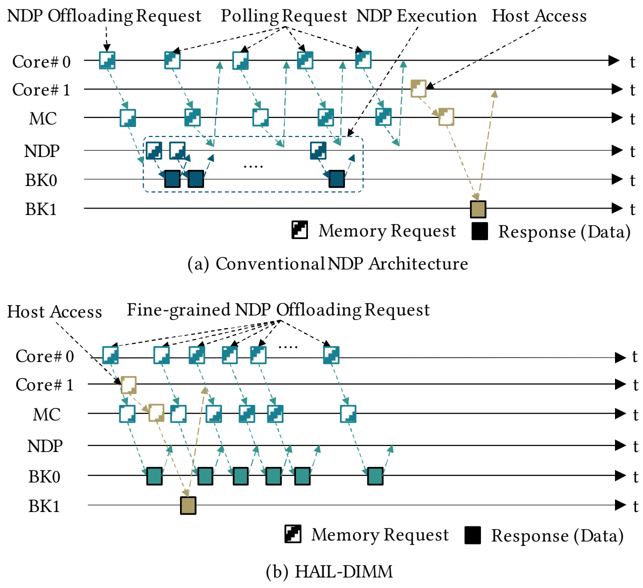

Figure 2. NDP offloading requests and NDP execution flow with host access in conventional NDP architecture and HAIL-DIMM. MC is a memory controller in CPU, and NDP is a NDP unit on DIMM.

The key idea of HAIL-DIMM is to allow NDP operations in the host's memory controller to be scheduled like host accesses, similar to HMC's atomic operation [8]. The NDP operation is an atomic operation consisting of data movement between DRAM and NDP unit and computation using it. This operation is executed by the DDR READ/WRITE command. Figure 2 illustrates the request flow of NDP and host access in conventional NDP architecture and HAIL-DIMM. In the conventional NDP architecture, Core#0 offloads the workload to NDP, and Core#1 waits until NDP execution is completed before accessing it. On the other hand, in HAIL-DIMM, when Core#0 sends a memory request to NDP-enabled memory to offload fine-grained NDP operation, NDP receives data from BK0 and performs the operation. NDP operation is performed by receiving data delivered in response to a memory request from the host without directly generating a memory request from the NDP unit. It allows Core#1 to send requests to memory regardless of the NDP operation in NDP-enabled memory.

The rest of this session is described as follows: we describe the architecture of HAIL-DIMM. Next, we introduce a DMA-based NDP offloading method to reduce the CPU's overhead to generate the find-grained NDP offloading requests. Finally, an FPGA-based prototype is described as a proof-of-concept of HAIL-DIMM.

3.2 Architecture of HAIL-DIMM

HAIL-DIMM, an LRDIMM-based NDP-enabled DIMM, necessitates only minimal modifications to the RCD and DBs to ensure compatibility with standard LRDIMMs. Figure 3 illustrates the architecture of HAIL-DIMM, featuring an NDP unit with computational capabilities within the DB. Furthermore, the existing BCOM controller (CTRL) is enhanced to differentiate between NDP and host accesses based on the memory request's address and to manage the NDP unit.

3.2.1 BCOM Controller in RCD. The original BCOM CTRL issues a *BCOM RD/WR* command to the DB for DDR READ/WRITE command. HAIL-DIMM distinguishes DDR command into host access or NDP operation based on the address of the requests. The access type of the NDP operation is further classified into *configuration register (CONF_REG)*, *instruction memory (Inst. Mem)*, *data memory (Data Mem)*, or *NDP execution (NDP_EXEC)*. Leveraging an *ACT-READ (or WRITE)-PRE* sequence of the memory request when accessing DRAM, HAIL-DIMM classifies the access type based on the ACT's address. Figure 3 demonstrates how the RCD identifies a memory request's access type. (❶) Upon receiving an ACT containing rank, bank group, bank, and row address, the RCD determines the access type using a predefined address space in an address table and updates the bank state table accordingly. The address space allocated for accessing the dedicated NDP region, such as CONF_REG, is tiny (only 0.00047% of a 128GB LRDIMM). The remaining address space is segmented into coarse-grained regions, each of which can be configured for either host access or NDP operation.

HAIL-DIMM expands the BCOM command set to facilitate control over the NDP unit. Two original BCOM commands are left as Reserved for Future (RFU) [14]. HAIL-DIMM exploits these remaining commands as two NDP commands (*NDP_RD* and *NDP_WR*). The BCOM commands follow a multi-cycle sequence. For instance, BCOM RD/WR requires three cycles, including of rank address and parity bit. The memory controller can issue a read or write command at a minimum interval of *tCCD* (i.e., four tCK) [12], making the maximum sequence length for BCOM commands four cycles. Therefore, the sequence of NDP_RD/WR BCOM commands takes four cycles. The sequence includes rank, bank group, bank address, and NDP-type information to control the NDP

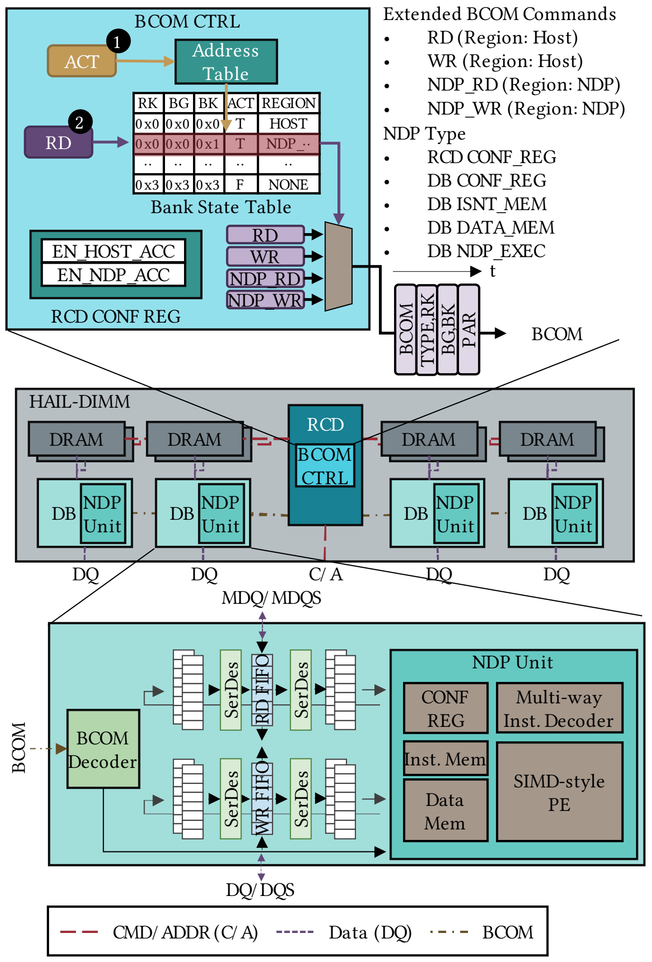

Figure 3. Architecture of HAIL-DIMM with access flow for multi-cycles BCOM command sequence in extended BCOM controller (CTRL) of RCD and NDP units in DB.

unit. To summarize, ❷ When RCD receives a DDR READ (or WRITE) command, it issues either an RD (or WR) or NDP_RD (or NDP_WR) BCOM command to the DB, determined by the region information in the bank state table.

3.2.2 NDP Unit in DB. The NDP unit can simultaneously operate host access and NDP, specifically designed to support bank-interleaving for enhanced NDP throughput. Figure 3 shows the structure of the DB. The NDP unit independently facilitates data movement and computation operation, positioned in parallel with the data path (RD_FIFO and WR_FIFO) between the memory controller and DRAM. RD_FIFO/WR_FIFO and NDP unit are interconnected through serializer-deserializer (SerDes). The NDP unit's clock frequency is four times lower than that of DDR4 DRAM (and eight times for DDR5). Since the clock of DRAM is up to 1.6 GHz for DDR4 [12] and up to 4 GHz for DDR5 [13], the frequency of NDP units implemented as digital logic should be lowered for implementation feasibility.

Upon receiving the NDP_RD BCOM command with NDP EXEC type, the NDP unit decodes and executes instructions stored in its instruction memory. HAIL-DIMM employs a

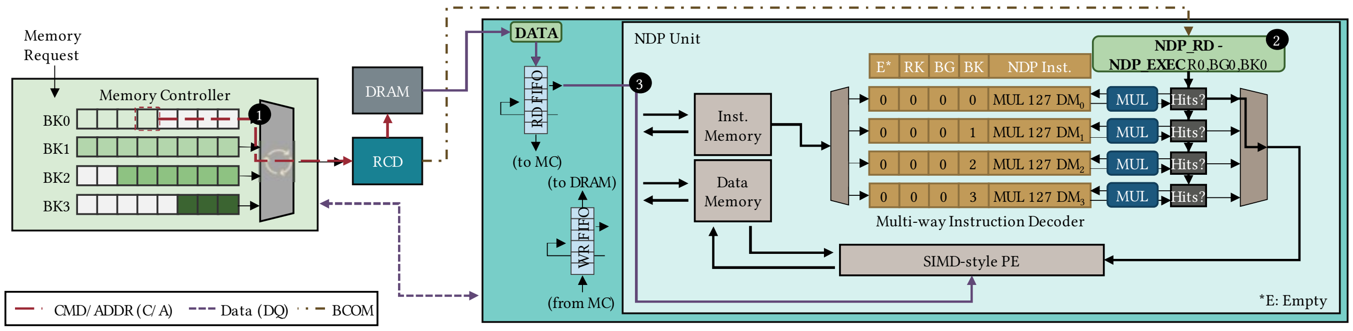

Figure 4. Execution flow of NDP operation on HAIL-DIMM. The example performs an element-wise multiplication on data stored in data memory and data stored in four banks of DRAM.

multi-way instruction decoder to synchronize the order of NDP instructions with the reordered memory request order by the memory controller. Figure 4 shows the flow of an NDP operation in the NDP unit. In this example, four NDP instructions carry out element-wise multiplication using data from four different banks and the NDP unit's data memory. The instruction decoder decodes the four NDP instructions in parallel. The host issues memory requests to four banks for NDP operation using the data stored in the four banks. ❶ The memory controller selects one of the memory requests and issues a corresponding READ command. ❷ Subsequently, the RCD sends an NDP EXEC BCOM command to the DB. The DB then selects a decoded instruction corresponding to the rank, bank group, and bank address specified in the BCOM command. Given the instruction decoder's limited width, programmers must carefully manage the number of banks simultaneously contributing to the NDP operation. ❸ After a specified delay (tCL), the DRAM transmits the data to the DB, and the NDP unit operates on the SIMD-style PE. In summary, The NDP unit that considers the memory controller's scheduling capability can have high NDP throughput by fully utilizing the bandwidth of DRAM through bank-interleaving.

3.3 DMA-based Fine-grained NDP Offloading

The granularity of the NDP operation of the HAIL-DIMM is the same as the access granularity of the DIMM (i.e., 64B). HAIL-DIMM necessitates many load instructions equivalent to the data volume required for the NDP operation. However, to execute these load instructions back-to-back, they must be distributed across multiple cores for simultaneous execution, owing to the limited entry capacity in each core's load-store unit. This means that NDP offloading requires many CPU resources. Therefore, this paper proposes a fine-grained NDP offloading technique using direct access memory (DMA) to generate burst memory requests. As DMA generates memory requests with consecutive addresses, the consecutive NDP behavior is affected by the memory controller's address mapping scheme. For example, if the address mapping scheme is *RK:RO:CO:BK:BG*, successive memory requests sequentially access all banks. With its multi-way instruction decoder, HAIL-DIMM can fully support NDP operations by interleaving access across multiple banks. Additionally, DMA is not limited to accessing consecutive addresses; it can generate diverse patterns of memory requests, utilizing features like scatter-gather. In summary, DMA-based fine-grained NDP offloading can significantly enhance NDP throughput by efficiently issuing numerous memory requests to HAIL-DIMMs while saving CPU resources.

3.4 Prototype on FPGA Platform

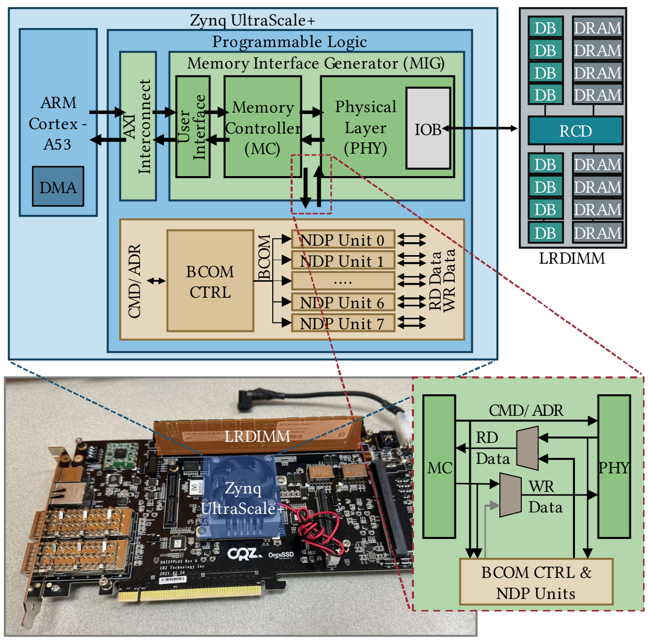

Figure 5. Prototype of HAIL-DIMM on Zynq UltraScale+ MPSoC FPGA platform with LRDIMM.

We implement a prototype of HAIL-DIMM on an FPGA platform as a proof-of-concept. We use Xilinx's memory controller IP, Memory Interface Generator (MIG), to validate the NDP operation without necessitating modifications to the memory controller. Figure 5 depicts the prototype implementation on an LRDIMM-equipped FPGA. To intercept DDR commands issued to the LRDIMM from the MIG, the HAIL-DIMM's BCOM controller and NDP units are located between the MIG's memory controller (MC) and the physical layer (PHY). The prototype receives data from the MC (or

DRAM) and the scheduled DDR command and emulates the NDP operation. The results of the NDP operation can be passed to the host through the RD Data Bus. This prototype uses the ARM A53 processor [16] in the Zynq FPGA platform to generate memory requests to control the HAIL-DIMM.

4 Evaluation

4.1 Evaluation Environment

We evaluate the performance and energy of the HAIL-DIMM on an ARM-based CPU system, including the prototype of HAIL-DIMM. Table 1 shows the details of the evaluation environment. The baseline CPU system comprises an *ARM A53 processor* with LPDDR4. However, as the LRDIMM is connected to the CPU via a Programmable Logic (PL) interface, it is unsuitable as the main memory in the baseline system due to its prolonged access latency. In the HAIL-DIMM system, the host processor uses *Xilinx's ZDMA driver* to offload NDP operation to the HAIL-DIMM [16]. The energy model of the off-chip bus in both systems is modeled by referring to [6]. This energy model assumes that both systems use LRDIMMs. We evaluate the performance and energy of both systems using memory-bound microbenchmarks. The microbenchmarks are hand-written code optimized using *ARM NEON instructions* [9]. The precision of the HAIL-DIMM is *FP16*, while the ARM A53 does not support *FP16*, so to use the same amount of data, the data type of the benchmark for the baseline is *INT16*.

Table 1. Evaluation environment. PS is a processing system, and PL is programmable logic. The values in parentheses are theoretical peak bandwidth.

PS	CPU	Quad-core ARM Cortex-A53 1.5GHz
	DMA	Xilinx's ZDMA
	DRAM	4GB LPDDR4-3200 (6.25 GB/s)
PS-PL Interface		Xilinx's FPD AXI Master Bus (4.68 GB/s)
PL	Memory Controller	Xilinx's MIG (12.5 GB/s)
	HAIL-DIMM	BCOM CTRL, 8 NDP units, FP16x4 SIMD unit per NDP unit
	DIMM	LRDIMM 64GB DDR4-2400

4.2 Evaluation Result

4.2.1 Performance. In Fig. 6 (a), we sweep the size of the vector and matrix. The HAIL-DIMM system achieves speeds of up to 1.76x, 1.43x, and 2.20x for element-wise addition, vector reduction, and matrix-vector multiplication (GEMV), respectively. The baseline system outperforms the HAIL-DIMM system in vector operations at smaller vector sizes because the HAIL-DIMM system uses DMA for NDP offloading, requiring much time to prepare DMA. If the CPU system and the HAIL-DIMM use identical DIMMs, the peak performances of both systems for a memory-bound application are the same due to the same off-chip bandwidth of both systems. However, to fully utilize the off-chip bandwidth in a CPU system, it is necessary to utilize numerous processes. On the other hand, HAIL-DIMM can utilize enough memory bandwidth without using much CPU resources by using the DMA-based NDP offloading technique.

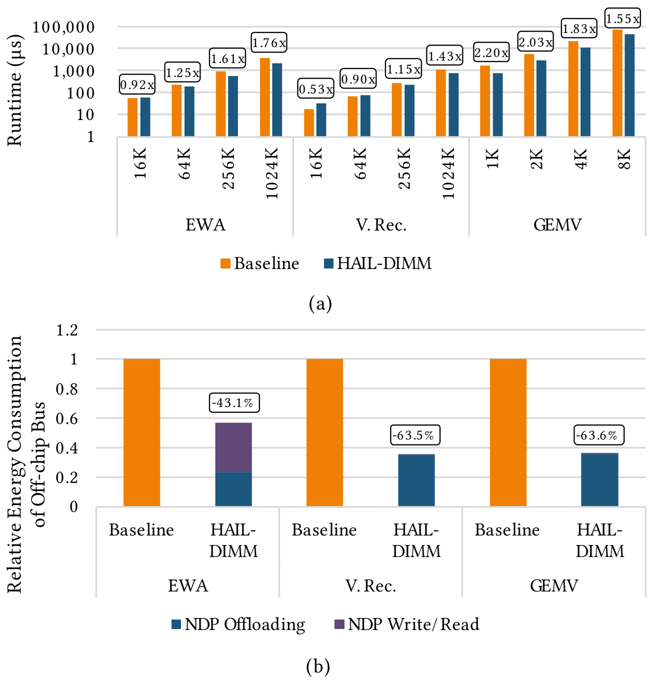

Figure 6. (a) Performance of baseline and HAIL-DIMM with different vector and matrix sizes. (b) Relative off-chip energy consumption of baseline and HAIL-DIMM. EWA is element-wise addition, V. Rec. is vector reduction, and GEMV is matrix-vector multiplication.

4.2.2 Energy Consumption of Off-chip Bus. As shown in Fig. 6 (b), HAIL-DIMM achieves energy savings ranging from 43.1% to 63.6% compared to the baseline. The HAIL-DIMM system achieves higher energy savings of vector reduction or GEMV than element-wise addition due to less data movement between the host and the DIMM. The energy consumption of data movement between the DB and DRAM on the DIMM is less than that between the memory controller and the DB [6]. Therefore, the HAIL-DIMM system is effective in saving energy on the off-chip bus by performing operations on the NDP unit in the DB and reducing data movement to the host.

4.2.3 Interleaving Host Access and NDP Operation. To evaluate the impact of HAIL-DIMM's burst access on host access, we issue memory requests with the same pattern as NDP operation to the main memory of the baseline. Figure 7 shows the performance of GEMV on the CPU when performing NDP operation. The performance of two workloads on one CPU and one NDP (oCoN) is 31.4% to 36.0% lower than a single workload on one CPU (oC). It shows that while burst memory access for NDP operation impacts host access, simultaneous access is still feasible during the NDP operation. In addition, the overhead of *close* and *open* a bank while accessing the same bank is substantial, which causes significant performance degradation. Therefore, if host access and NDP are assigned to different ranks, the performance degradation can be reduced by mitigating this latency overhead.

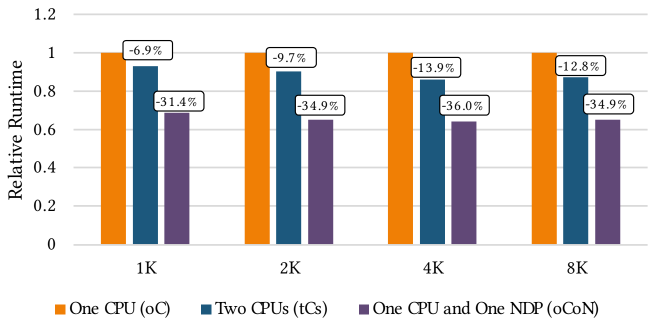

Figure 7. Performance of GEMV with different vector sizes on CPU with NDP operation.

5 Discussion and Future Work

Scale-Out with multi-DIMMs per memory channel: The HAIL-DIMM cannot utilize rank-level parallelism. However, HAIL-DIMM has the potential to utilize DIMM-level parallelism. The ABC-DIMM proposed an inter-DIMM broadcasting technique to utilize DIMM-level parallelism [15]. Adopting the DIMM broadcast technique could enable HAIL-DIMM to enhance its effective bandwidth. Within a memory system that supports inter-DIMM broadcasting, a HAIL-DIMM system with 2/4/8 DIMMs per channel (DPC) has 1.9/3.8/6.9x performance improvement over 1 DPC for GEMV, respectively.
Error correction code (ECC): Based on LRDIMMs, the NDP architecture currently does not support ECC for NDP operations due to the independence of DBs in LRDIMMs and the lack of a data transfer path between them. As part of our future work, we plan to explore the ECC-enabled NDP architecture within our proposed architecture by exploiting a new data path between DBs.

6 Conclusion

This paper proposes HAIL-DIMM, which can seamlessly interleave NDP and host access, and an efficient DMA-based NDP offloading method for fine-grained HAIL-DIMM operation. We implement a prototype of HAIL-DIMM on an FPGA platform to evaluate its feasibility and performance. Our evaluation shows that the proposed NDP architecture achieves performance improvements and energy savings over the CPU-based system for memory-bound workloads.

Acknowledgments

This work was financially supported by the Industrial Core Technology Development Program of MOTIE/KEIT (RS-2022-00144288, Development of DRAM Interface Semiconductor Integrated with Deep Learning Accelerator)

References

[1] Hadi Asghari-Moghaddam, Young Hoon Son, Jung Ho Ahn, and Nam Sung Kim. 2016. Chameleon: Versatile and practical near-DRAM acceleration architecture for large memory systems. In *2016 49th annual IEEE/ACM international symposium on Microarchitecture (MICRO)*. IEEE, 1–13.
[2] Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, et al. 2018. Google workloads for consumer devices: Mitigating data movement bottlenecks. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*. 316–331.
[3] Wenqin Huangfu, Xueqi Li, Shuangchen Li, Xing Hu, Peng Gu, and Yuan Xie. 2019. Medal: Scalable dimm based near data processing accelerator for dna seeding algorithm. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 587–599.
[4] Wenqin Huangfu, Krishna T Malladi, Shuangchen Li, Peng Gu, and Yuan Xie. 2020. Nest: Dimm based near-data-processing accelerator for k-mer counting. In *Proceedings of the 39th International Conference on Computer-Aided Design*. 1–9.
[5] Liu Ke, Udit Gupta, Benjamin Youngjae Cho, David Brooks, Vikas Chandra, Utku Diril, Amin Firoozshahian, Kim Hazelwood, Bill Jia, Hsien-Hsin S Lee, et al. 2020. Recnmp: Accelerating personalized recommendation with near-memory processing. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 790–803.
[6] Liu Ke, Xuan Zhang, Jinin So, Jong-Geon Lee, Shin-Haeng Kang, Sukhan Lee, Songyi Han, YeonGon Cho, Jin Hyun Kim, Yongsuk Kwon, et al. 2021. Near-memory processing in action: Accelerating personalized recommendation with axdimm. *IEEE Micro* 42, 1 (2021), 116–127.
[7] Youngeun Kwon, Yunjae Lee, and Minsoo Rhu. 2019. Tensordimm: A practical near-memory processing architecture for embeddings and tensor operations in deep learning. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 740–753.
[8] Lifeng Nai and Hyesoon Kim. 2015. Instruction offloading with hmc 2.0 standard: A case study for graph traversals. In *Proceedings of the 2015 International Symposium on Memory Systems*. 258–261.
[9] Venu Gopal Reddy. 2008. Neon technology introduction. *ARM Corporation* 4, 1 (2008), 1–33.
[10] Patrick Siegl, Rainer Buchty, and Mladen Berekovic. 2016. Data-centric computing frontiers: A survey on processing-in-memory. In *Proceedings of the Second International Symposium on Memory Systems*. 295–308.
[11] JEDEC Standard. 2019. DDR4 Data Buffer (DDR4DB02) JESD82-32A. *JEDEC Solid State Technology Association* (Aug 2019).
[12] JEDEC Standard. 2021. DDR4 SDRAM JESD79-4D. *JEDEC Solid State Technology Association* (July 2021).
[13] JEDEC Standard. 2022. DDR5 SDRAM JESD79-5B. *JEDEC Solid State Technology Association* (Aug 2022).
[14] JEDEC Standard. 2023. DDR4 Registering Clock Driver (DDR4RCD02) JESD82-31A.01. *JEDEC Solid State Technology Association* (Jan 2023).
[15] Weiyi Sun, Zhaoshi Li, Shouyi Yin, Shaojun Wei, and Leibo Liu. 2021. ABC-DIMM: alleviating the bottleneck of communication in dimm-based near-memory processing with inter-dimm broadcast. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 237–250.
[16] Xilinx 2023. *Zynq UltraScale+ Device Technical Reference Manual (UG1085)*. Retrieved November 19, 2023 from https://docs.xilinx.com/r/en-US/ug1085-zynq-ultrascale-trm/Zynq-UltraScale-Device-Technical-Reference-Manual