2025 IEEE International Symposium on High-Performance Computer Architecture (HPCA)

PIMnet: A Domain-Specific Network for Efficient Collective Communication in Scalable PIM

Hyojun Son[1] Gilbert Jonatan[1] Xiangyu Wu[1] Haeyoon Cho[1] Kaustubh Shivdikar[2]
José L. Abellán[3] Ajay Joshi[4] David Kaeli[2] John Kim[1]
[1]KAIST [2]Northeastern University [3]Universidad de Murcia [4]Boston University
{processor, gilbertjonatan, wuxiangyu, haeyoon.cho}@kaist.ac.kr, shivdikar.k@notheastern.edu
jlabellan@um.es, joshi@bu.edu, kaeli@ece.neu.edu, jjk12@kaist.edu

***Abstract*—Processing-in-memory (PIM), where compute is moved closer to memory or data, has been explored to accelerate emerging workloads. Different PIM-based systems have been announced, each offering a unique microarchitectural organization of their compute units, ranging from fixed functional units to programmable general-purpose compute cores near memory. However, one fundamental limitation of PIM is that each compute unit can only access its local memory; access to "remote" memory must occur through the host CPU – potentially limiting application performance scalability. In this work, we first characterize the scalability of real PIM architectures using the UPMEM PIM system. We analyze how the overhead of communicating through the host (instead of providing direct communication between the PIM compute units) can become a bottleneck for collective communications that are commonly used in many workloads. To overcome this inter-PIM bank communication, we propose *PIMnet* – a PIM interconnection network for PIM banks that provides direct connectivity between compute units and removes the overhead of communicating through the host. PIMnet exploits *bandwidth parallelism* where communication across the different PIM bank/chips can occur in parallel to maximize communication performance. PIMnet also matches the DRAM packaging hierarchy with a multi-tier network architecture. Unlike traditional interconnection networks, PIMnet is a PIM-*controlled* network where communication is managed by the PIM logic, optimizing collective communications and minimizing the hardware overhead of PIMnet. Our evaluation of PIMnet shows that it provides up to 85× speedup on collective communications and achieves a 11.8× improvement on real applications compared to the baseline PIM.**

I. Introduction

Emerging workloads, such as deep neural networks, demand computational capabilities beyond those provided by general-purpose architectures [28]. As a result, domain-specific architectures are commonly used to support the necessary computations [19], [40], [54], [73], [80], [86]. However, system performance is often bottlenecked by the memory system and by the movement of data to/from the main memory system [21], [45]. This bottleneck is becoming more problematic from data-intensive workloads that require large memory capacities and high memory bandwidth. Processing-in-memory (PIM) has been proposed as a potential solution to reduce the memory bandwidth gap and improve overall performance [64]. Recently, memory vendors have announced different PIM modules, including the Samsung HBM function-in-memory (FIM) [55], SK Hynix GDDR-based PIM [58], and the UPMEM DPU [25].

The type of compute across the different PIMs differ in terms of compute throughput and programmability. For example, SK Hynix PIM [58] has fixed functional units while Samsung FIM [59] provides some (limited) programmable compute logic. UPMEM [25] provides a general-purpose compute core near the memory banks. All PIM implementations pursue a similar goal of trying to minimize changes to the standard memory interface (e.g., DDRx protocol) while providing computation near the memory. However, the compute unit or logic only has access to its "local" memory (or memory bank) and cannot access "remote" memory or data located near other compute units or banks. Thus, ***one fundamental limitation of current PIM architectures is that remote memory accesses or accesses to a "non-local" PIM bank*** [1] ***occurs through the host CPU.*** This indirect PIM-to-PIM communication through the host impacts both the latency and bandwidth of inter-PIM communications.

In many parallel workloads, collective communication is commonly used between multiple nodes to communicate or exchange data following computation and is commonly used in CPUs [15], [84] and GPUs [3] through collective communication libraries. Modern PIM SDK (such as UPMEM [6]) provides some collective operations (e.g., gather, broadcast) and SimplePIM [16] provides an additional interface/wrapper to support collective communication. Recently, PID-Comm [67] proposed optimizations to improve collective communication for PIM by reducing the collective communication overhead in the host CPU. However, PID-Comm is a *software*-based approach to accelerate collective communication, and unlike other nodes such as CPUs or GPUs that have inter-node interconnect, the performance of collective communication in PIM is fundamentally limited by the communication bandwidth to/from the host CPU. Thus, in this work, we propose a PIMnet architecture where a domain-specific network [7] enables direct communication between PIM banks and efficiently accelerates collective communication.

Prior work [20], [83], [89] has proposed interconnecting DIMMs to provide direct communication between the DIMMs. While these work avoids communicating through the host, they assume that the near-memory compute unit is placed

[1]In this work, we refer to a PIM bank as the unit of compute and "local" bank memory. We use PIM bank and PIM node terminology interchangeably.

on a centralized buffer chip in the DIMM. In comparison, the benefits of PIM are maximized when near-memory compute is provided at a bank granularity, in order to exploit internal memory bandwidth [25], [58], [59] and prior work do not enable communication between near-bank PIM nodes. Given a large number of banks, introducing a *general-purpose* interconnect between the DIMM (or banks) is prohibitively expensive because of memory technology limitations (e.g. limited number of metal layers) and the hardware overhead; thus, we propose a *domain*-specific network [7] for PIM that targets collective communication which is commonly found in many emerging workloads [17], [65].

In this work, we first revisit the impact of inter-PIM communication by using a roofline model analysis and demonstrate the potential benefit of PIM interconnection network and how direct communication improves inter-PIM communication bandwidth and collective communication performance. We also show how the scalability of modern PIM systems from collective communication is limited, even with an *ideal* software-based collective communication on a PIM system. To enable PIM scalability when communication is required, we propose ***PIMnet – a processing-in-memory interconnection network to enable direct communication between the PIM banks*** across all DIMMs sharing the same memory bus channel. Thus, PIMnet avoids the high cost of communicating through the host (or the CPU). In particular, we exploit the communication or traffic pattern of collective communication to design a *PIM-controlled* interconnection network. The inter-PIM communication is carefully managed or "orchestrated" by the PIM to ensure on-chip communication contention is removed while minimizing the interconnection network hardware overhead. Two critical factors in any interconnection network are technology and the packaging constraints [23]. PIM interconnect presents unique challenges because of the memory packaging hierarchy and the memory technology constraints that limit the amount of logic and wire that can be introduced. PIMnet is a multi-tier topology that exploits both a direct and an indirect network to match the packaging hierarchy of PIM (or memory) systems and the collective communication pattern. PIMnet exploits PIM bandwidth *parallelism* (or parallel communication among different PIM banks) which can be exploited when local (e.g., inter-bank) communication can be performed in parallel. By providing direct communication between the PIM banks without support from the host, PIMnet enables performance to scale as the number of PIM banks increases. Effectively, the goal of PIMnet is to enable "One Gigantic PIM" architecture, instead of an architecture that is a collection of many PIM banks. In particular, the key contributions of this work include:

- We identify a bottleneck of communication in modern PIM architectures that limits scalability as communication occurs through the host and demonstrate the performance limitation on the UPMEM PIM system.
- We propose PIMnet – a *domain*-specific interconnection network for PIM that enables efficient collective communication through direct connectivity between the PIM nodes and avoid indirectly communicating through host.
- We propose a PIMnet architecture that matches the memory packaging constraints through a multi-tier interconnect. PIMnet is PIM-*controlled* interconnect as PIM logic orchestrates communication between the PIM banks to exploit the PIM bandwidth parallelism.
- We evaluate the benefits of PIMnet and show compared to communicating through the host, collective communication performance can be improved by up to 85×.

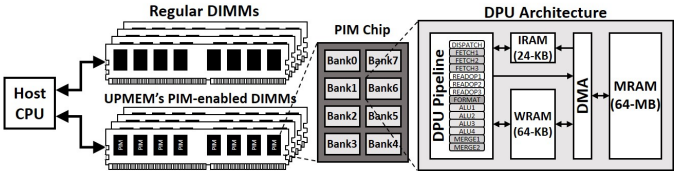


Fig. 1: High-level block diagram of a UPMEM system that consists of regular DIMM and PIM-enabled DIMMs.

II. Background

A. UPMEM PIM Architecture

To evaluate the scalability of a representative PIM-based memory system, we use the UPMEM architecture [25]. While other PIM architectures such as Samsung HBM PIM [59] or SK Hynix GDDR PIM [58] have been announced, they are not publicly available. Even more importantly, the UPMEM architecture provides the most flexible PIM solution in terms of general-purpose computation that can be exploited for near-data processing. Figure 1 presents a high-level block diagram of a UPMEM PIM system. Inside each chip, there are 8 DRAM Processing Units (DPUs) and 8 64-MB memory banks. Each DPU has 14-stage pipeline, 32-bit processor, that supports up to 24 hardware threads, called tasklet. [25] Each PIM bank consists of a DPU, 64-MB main memory bank (MRAM), a 24-KB instruction memory (IRAM), and a 64-KB scratchpad memory (a software-managed cache called WRAM). The PIM-enabled memory system is organized hierarchically – multiple PIM banks on a single PIM chip, multiple chips on a DIMM, and multiple DIMMs in the system. The host CPU can read/write data from/to MRAM through the DDR4 interface using UPMEM APIs [6]. Only data located in WRAM can be used for computation and the DMA module is responsible for moving the data between MRAM and WRAM within each bank.

B. Collective Communications

Collective communication is defined as communication that occurs in a coordinated manner between a group of processes or nodes in a parallel (distributed) system [15], [84]. Common collective communication patterns include AllReduce, where reduction operations (e.g., average, sum, etc.) are performed on partial sums across multiple nodes, and All-to-All where all pairs of nodes exchange data. Collective communication is a critical component of many parallel algorithms and applications in high-performance computing (HPC) [30], [69]. These

	PID-Comm [67]	DIMM-Link [89]	NDPBridge [85]	PIMnet (This work)
Inter-bank communication	CPU	Buffer chip	Buffer chip	Memory chip
Inter-chip communication	CPU	Buffer chip	Buffer chip	Buffer chip
Inter-rank communication	CPU	Dedicated link	CPU	Memory bus
Collective operation	CPU	Buffer chip	N/A	PIM bank

TABLE I: Qualitative comparison of PIMnet with prior work regarding where inter-PIM communication is done. Collective *operation* refers to the computation necessary in some collective communication (e.g., reduction operation in AllReduce).

operations have also become important in many emerging workloads, including deep neural networks [74], [78]. As the number of nodes increases, collective communication is a critical component in communication performance and has a direct impact on overall system performance.

Most prior work for PIM do not target workloads where communication is required between the PIM banks because of high overhead of PIM-to-PIM communication. Recent work have proposed improving PIM-to-PIM communication through a collective communication API interface (SimplePIM [16]) and optimizing the host CPU overhead for collective communication (PIM-Comm [67]). However, these work are effectively "software"-based approaches to collective communication and are limited by the bandwidth available to/from the host CPU. Other work [85], [89] have proposed hardware-based communication between DRAM but do not provide direct PIM bank-to-bank communication. In comparison, this work proposes an efficient hardware-based collective communication for PIM-to-PIM communication. To the best of our knowledge, this is one of the first work to propose a PIM-specific interconnection network. The qualitative comparison between PIMnet and prior work is summarized in table I.

C. Evaluation Methodology

In this work, we use the UPMEM PIM system [25] since it is publicly available and programmability (or flexibility) is provided with a general-purpose compute logic. However, our observations on scalability (or limited inter-bank PIM communication) apply to other commercial PIM architectures [58], [59]. The configuration of the UPMEM system used in our evaluation is shown in Table II. For evaluation, we developed kernel code using the UPMEM API [6] (which is based on C) and the source code is partitioned into host code that is executed on the CPU, and kernel code that is offloaded to the PIM DPU. While there has been recent work on collective communication in UPMEM systems [16], [67], benchmarks with collective communication were not readily accessible. Thus, we implemented similar workloads as prior work including BFS, CC, MLP, and GEMV. The baseline collective communication (using the host) was implemented used UPMEM API – identical to how collective communication was implemented in SimplePIM [16]. We also evaluated two emerging workloads – the embedding table (`EMB`) found

	Configuration
CPU	Intel(R) Xeon(R) Silver 4215R CPU @ 3.20GHz
Number of PIM DIMM	20 DIMMs (20 Ranks)
Number of regular DIMM	4 DIMMs
Number of DPUs	2560 DPUs
DPU Frequency	350 MHz
PIM-Enabled Memory	171 GB
DRAM Memory	96 GB

TABLE II: System configuration of the UPMEM server.

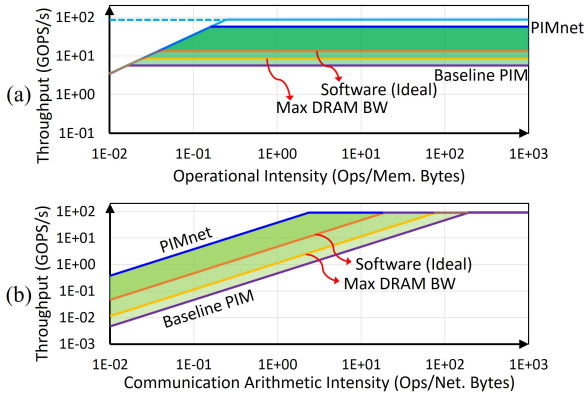


Fig. 2: Roofline models to demonstrate the potential benefits from a direct PIM-to-PIM interconnect, including (a) traditional roofline model [87] and (b) modified roofline model using communication arithmetic intensity [14].

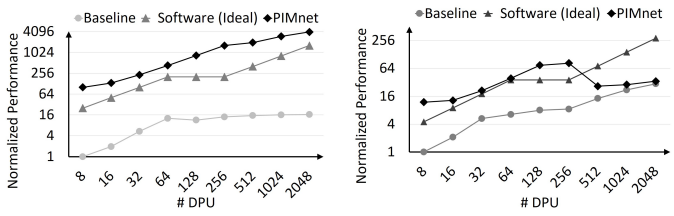
in deep learning recommendation models (DLRM) [65] and Number Theoretic Transform (NTT), used in Fully Homomorphic Encryption [17]. NTT was implemented based on iterative FFT algorithm, and merged Cooley–Tukey NTT optimization [79] with $N = 2^{16}$ (number of coefficients). 2D NTT [12] was also implemented – i.e., two computation NTT steps with each step consisting of 256 256-point NTTs using 16 tasklets on each DPU. The computation steps in 2D NTT are column-wise NTTs followed by row-wise NTTs that required All-to-All collective communication.

III. Motivation: PIM Scalability Limitation

Prior work [39], [47], [67] have identified challenges from the lack of inter-PIM communication in modern PIM systems. In this section, we revisit the challenges of inter-PIM communication and the limitations of PIM scalability caused by the communication between the PIM and the host CPU. In particular, we use the roofline model to present the potential performance benefits from a dedicated PIM interconnect. We then present the scalability limitations of the collective communications on the UPMEM PIM system.

A. Roofline Model

Two different roofline models are shown in Figure 2 to highlight the potential benefits of a dedicated PIM interconnect (shown as `PIMnet`). We compare `PIMnet` with the baseline PIM based on the UPMEM [25] and two alternative PIM collective communication modeling – `Max DRAM BW` and `Software(Ideal). Max DRAM BW` assumes that the



(a) AllReduce (b) All-to-all

Fig. 3: Scalability comparison of (a) AllReduce and (b) All-to-all collective communication across different PIM implementations.

DRAM bandwidth (e.g., 19.2 GB/s) can be fully utilized for collective communication [2] while `Software(Ideal)` models an *idealized* software-based collective communication where only communication to/from the host are modeled and the host overhead is assumed to be zero. [3]

Figure 2(a) shows a conventional roofline model that plots operational intensity against compute throughput. Since all implementations have the same internal memory bandwidth, the sloped line is identical with the maximum horizontal line showing the compute limit maximum compute throughput of the PIM system. However, workloads that require inter-PIM communication result in lower system (compute) utilization from the communication overhead (`Baseline PIM`). Increasing the DDR bandwidth utilization (`Max DRAM BW`) improves the compute throughput slightly while using an optimized software collective communication can further increase the compute throughput (`Software(Ideal)`). However, all of these approaches require communication through the host and are limited by the bandwidth (and the serialized communication) through the shared memory channel. In comparison, a dedicated PIM interconnect (`PIMnet`) can theoretically achieve throughput that nearly matches the throughput of the underlying PIM architecture, achieving approximately 8× more compute throughput than `Software(Ideal)`.

An alternative roofline model that uses *communication* arithmetic intensity [14] or the number of operations per amount of bytes sent over the network is shown in Figure 2(b). Using this model, the difference in the collective communication bandwidth is shown through the different slopes of the roofline plot. For PIM workloads that require communication, the roofline model shows how most of the region will be communication-bound, shown by the sloped line for `Baseline`. The other two approaches improve the communication bound but with a hardware-based `PIMnet`, the PIM workloads become less communication bound and maximize PIM compute throughput.

[2]Prior work [39] demonstrated that the DRAM bandwidth that can be achieved on the DDR4 UPMEM system is between 4.7 and 16.9 GB/s.

[3]An example of software-based collective communication includes PID-Comm [67] which optimizes collective communication compared to the baseline PIM; however, such software approach only reduces the host overhead but PIM-to-PIM communication still requires communication through the host.

Design Goals	PIMnet Design
Low-radix network	Ring bank-to-bank network
Simplify arbitration	No hardware arbitration through scheduling
No network buffers	No network contention
Minimize pins	Utilize existing channels

TABLE III: PIMnet design goals and how PIMnet achieves the design objectives highlighted.

B. Collective Communication Scalability

Performance scalability of the collective communication across different PIM-based implementations is shown in Figure 3. The results are shown for weak scaling as the message size is proportionally increased as the number of PIM banks (or DPUs) is increased. The results are normalized to the baseline system at 8 PIM banks. Similar to the roofline model analysis, `PIMnet` is compared to `Baseline` [16], [25] and `Software(Ideal)` – an ideal implementation of PID-Comm [67] where all host overhead is removed. The baseline PIM results in the lowest performance and an ideal software implementation provides significant performance improvement compared to the baseline implementation for AllReduce. However, both are still "software" implementations, and the scalability is limited beyond 64 banks when multiple DIMMs (or ranks) within the same channel are utilized. In comparison, a dedicated PIM interconnect provides not only higher performance but also better scalability as `PIMnet` does not communicate through the host CPU. In addition, PIM bandwidth parallelism can be exploited as the number of ranks increases beyond one as local AllReduce within each rank or chip can be done in parallel. While the global reduction is limited to the DDR memory channel bandwidth, the amount of data that needs to be communicated after local reduction is much smaller – thus, effectively improving overall scalability. The benefit of `PIMnet` for All-to-All is lower since all traffic is global and bottlenecked by the DDR memory channel bandwidth; however, the benefit of `PIMnet` is still approximately 2× compared to an ideal software implementation through 256 DPUs. In this work, we assume 64 DPUs per rank and 4 ranks per memory channel; thus, PIMnet only provides connectivity between the 256 DPUs within a single memory channel. When DPUs across multiple channels need to communicate, PIMnet still requires communication through the host – similar to the baseline PIM system. It remains to be seen if PIMnet can be extended to inter-memory channel communication.

IV. PIMnet Architecture

An interconnection network between PIM nodes presents new challenges and opportunities. Because of memory packaging constraints and limited amount of logic in memory technology, a conventional hardware-based interconnect router microarchitecture is not feasible. Hence, we propose PIMnet – a domain-specific interconnection network [7] for inter-PIM bank collective communication between the PIM nodes. An important characteristic of collective communication is *determinism* in the communication pattern – i.e., the source, the destination, and the message size are known ahead of

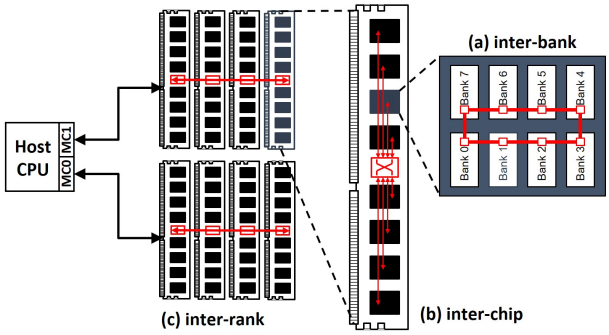


Fig. 4: High-level overview of the PIMnet that consists of (a) inter-bank, (b) inter-chip, and (c) inter-rank networks to enable direct inter-PIM communication.

time [8]. PIMnet exploits this behavior to enable a PIM-controlled interconnect that orchestrates the communication. In this section, we describe the necessary software changes to enable efficient collective communication through PIMnet and present the PIMnet microarchitecture in the following section.

A. PIM-constrained Interconnect

The cost of an interconnection network and routers is often defined by the buffers and crossbar datapath as well as the arbitration that is often on the critical path [48]. A unique constraint of PIM interconnect is that the technology is based on memory (or DRAM) process technology – resulting in limited availability of logic and wires/bandwidth. As a result, the design objectives of PIMnet include the following:

- **Keep radix low:** Maintain low port count to enable low switch/crossbar complexity, which is proportional to $O(k^2)$ where k is the switch radix.
- **Simplify arbitration:** Reduce or avoid switch arbitration to minimize impact on the critical path and network throughput.
- **No Buffers:** Minimize the amount of switch input buffers to reduce router overhead.
- **Minimize "pins":** Adding more pins to existing DRAM packaging is infeasible while introducing more wires within a chip is also very costly because of limited metal routing resources.

To achieve these objectives, general-purpose interconnection network is prohibitively expensive given the DRAM constraints. We propose a domain-specific interconnection network where communication is "scheduled" to avoid the complexity of a conventional interconnection network. Instead of dynamic flow control with buffering and arbitration, communication is scheduled such that contention does not occur – thus, removing the need for any input buffers and arbitration. The key observation is that the ***communication traffic pattern is known ahead of time or is deterministic***. For example, collective communication such as AllReduce and All-to-all communications for global communication between PIM banks have known communication patterns (i.e., source/destination and traffic amount) and can be scheduled.

	Inter-bank	**Inter-chip**	**Inter-rank**
Physical channels	Bank I/O bus	DQ pins	DDR bus
# channel	4	2	1 (half-duplex)
Width per channel (b)	16	4	64
Bandwidth per channel (GB/s)	0.7	1.05	16.8
Physical topology	ring	crossbar	bus
Router	PIMnet stop	Buffer chip	Buffer chip

TABLE IV: Comparison of different network hierarchy in PIMnet.

In this section, we describe PIMnet which requires changes to the PIM architecture, both in terms of the hardware and the software. We use the UPMEM as our baseline and avoid modifyi ng the DDR interface to the host to ensure compatibility with the existing host/CPU interface.

B. PIMnet Overview

To avoid the communication performance overhead through the host, PIMnet provides a dedicated inter-PIM interconnect for communication. A high-level block diagram of the PIMnet is shown in Figure 4, which consists of a multi-tier interconnect to match the DRAM packaging hierarchy. At the lowest level, *inter-bank* network provides connectivity between banks within a single chip. The next level of interconnect is the inter-chip network that connects multiple chips together within a single rank. The last level of interconnect is the inter-rank network across the different ranks (or DIMMs) that share the same memory channel. To address the PIM-constrained challenges, PIMnet has the following principles (Table III):

- Utilize existing DRAM pins and/or on-chip wires to maximize DRAM constraints.
- PIM-*controlled* interconnect, where the interconnect or communication is managed by the PIM units.
- Data movement is "scheduled" to avoid contention and minimize interconnect complexity.

A high-level qualitative comparison of the three tiers of PIMnet is provided in Table IV. The configuration shown is only one possible implementation of PIMnet and different configurations as possible, as long as the DRAM packaging constraints are met – e.g., instead of a bi-directional ring for inter-bank, a unidirectional ring with 2 channels and 32 bits per channel can be created. All three tiers utilize existing interconnect/pins as much as possible. Because of the packaging constraints, the bandwidth provided across each tier differs. Within each chip, approximately 0.7 GB/s per bank channel results in a bisection bandwidth of 2.8 GB/s. With 8 chips per rank, the total inter-bank bisection is approximately $2.8 \times 8 = 22.4$GB/s. When all banks communicate in parallel for bandwidth-optimal collective communication (e.g., Ring AllReduce), PIMnet provides $2.8 \times 64 = 179.2$GB/s of aggregated send and receive bandwidth per rank. Each tier also differs in the topology as the inter-bank is a *direct* network where each PIM node has its own "router" while both the inter-chip and inter-rank are *indirect* network by exploiting the buffer chip to enable PIMnet.

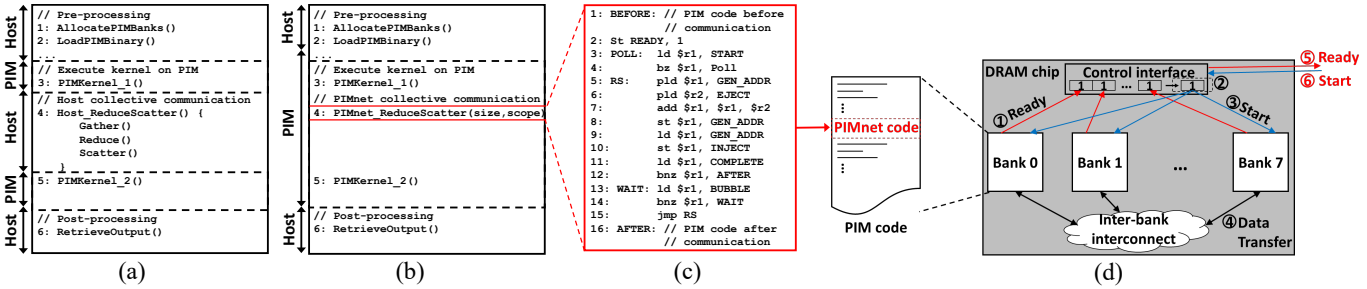


Fig. 5: High-level pseudo-code of (a) baseline collective communication for PIM, and (b) proposed PIMnet, (c) PIM instructions that are offloaded for collective communication, and (d) the execution flow of PIMnet communication across multiple banks.

C. PIMnet Software/Interface

An overview of the collective communication in the baseline PIM systems is shown in Figure 5(a) which illustrates a pseudo-code that is executed across the host and the PIM. Recent work has proposed alternative collective communications, including SimplePIM [16], a software framework to support PIM programming and provide communication primitives, and PID-Comm [67] which optimizes host overhead for collective communication. However, the high-level behavior shown in Figure 5(a) is valid as the collective communication (e.g., `Host_ReduceScatter()`) is executed on the host while the PIM kernel is offloaded to the PIM. Once the PIM kernel is launched (and executed), collective communication (i.e., Reduce-Scatter) is executed on the host. In the proposed PIMnet, changes to the pseudo-code are shown in Figure 5(b). The PIM kernel code remains the same but the PIMnet API is called for collective communication. The key difference is that the collective communication results in a sequence of instructions that is also offloaded to the PIM, similar to the PIM kernel code itself. [4] In the example, the arguments for `PIMnet_ReduceScatter()` include `size` which is the number of elements that need to be communicated, and `scope` which defines how many banks (or DPUs) are participating in the collective communication.

The corresponding PIM code for the collective communication is shown in Figure 5(c). The PIM assembly instructions consist of multiple phases including `POLL` phase which provides synchronization between the DPUs before the execution of the collective communication and `RS` (Reduce-Scatter) which iterates across the data from the WRAM to perform the local reduction and move data using the PIMnet. `WAIT` phase is used to ensure access to the shared inter-chip (or inter-rank) channels is properly scheduled to avoid contention between the banks on the same DRAM chip. The execution flow of the collective communication on the PIM is shown in Figure 5(d). During the `POLL` phase, each PIM bank sends the `READY` signal to the control interface unit that is shared by all banks on the same chip. The control interface aggregates the `READY` signals and if the scope of the communication is beyond a single DRAM chip, `READY` signal is sent to the inter-chip switch (and then, to the inter-rank switch if the scope consists of multiple ranks.) Once the `READY` signals are all aggregated, `START` signal is sent back to the appropriate banks such that the PIM banks can begin their collective communication.

[4]The behavior is similar to GPUs where collective communication kernels are launched by the host and executed on the GPU.

V. PIMnet Microarchitecture

In this section, we describe the details of the PIMnet microarchitecture and the direct PIM-to-PIM interconnect across the memory packaging hierarchy, including inter-bank, inter-chip, and inter-rank.

A. Inter-Bank Network

A detailed block diagram of PIMnet architecture is shown in Figure 6. The main microarchitectural change for the inter-bank interconnect (Figure 6(a)) is the *PIMnet stop* or a "router" that is added to interconnect the PIM banks together. Since all communication is "scheduled" to avoid any contention, no input buffers or arbitration (or hardware flow control) is necessary within the PIMnet stop. Hardware routing is also not necessary since the packets or data movement are scheduled or "orchestrated," similar to software-scheduled network [8]. With the simplified architecture, PIMnet becomes a ***deterministic*** interconnect architecture as the latency between a source and a destination is fixed as there is no source of non-determinism, including queuing or arbitration. However, PIMnet introduces two requirements to ensure the communication is done deterministically – the launch of communication needs to be synchronized and the "scheduling" of the communication is needed to ensure there is no contention for communication resources. As described earlier in Section IV-C, `READY` signal is leveraged for synchronization. Based on the type of collective communication, the scheduling is done differently. For example, for AllReduce, hierarchical Ring-based AllReduce is used while for All-to-all communication, the scheduling is done based on ring algorithm in inter-bank communication and permuted injection pattern for inter-chip and inter-rank communication to avoid any contention in inter-chip switch or on the multi-drop memory bus.

The changes to the datapath of the inter-bank network is also shown in Figure 6(a). Additional datapath is added between the WRAM and PIMnet stop (for inter-bank communication) and from the WRAM to the DDR interface for inter-chip/rank communication as the data for communication is provided from the WRAM. The added datapath is only utilized when

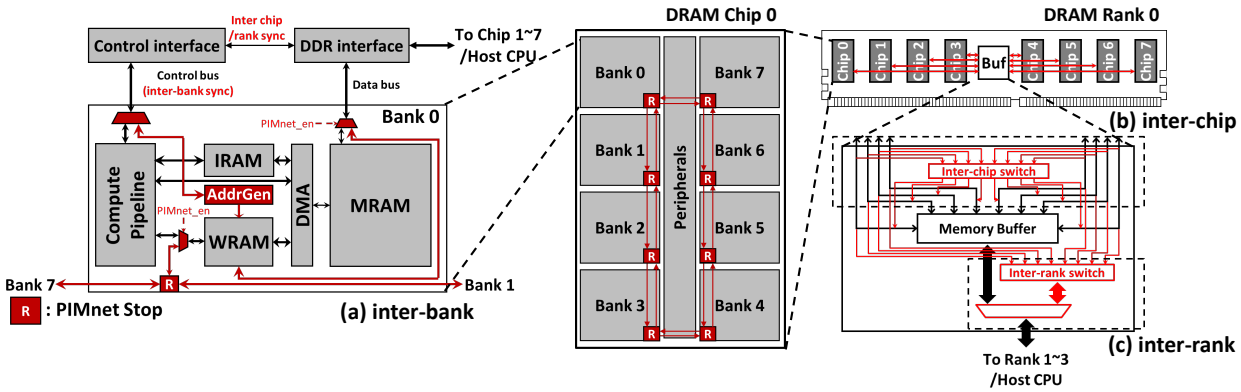


Fig. 6: A detailed block diagram of PIMnet architecture across multi-tier interconnect consisting of (a) inter-bank, (b) inter-chip, and (c) inter-rank networks. The red components highlight the new components introduced with PIMnet.

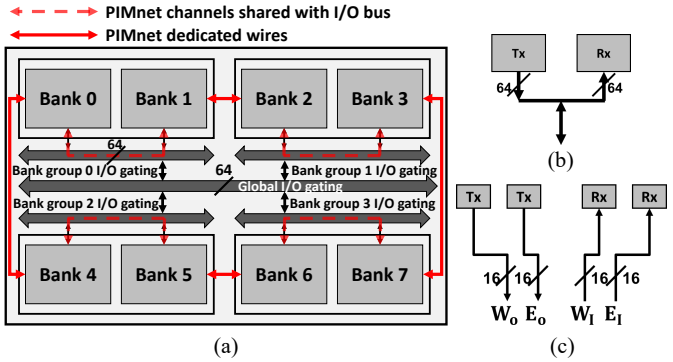


Fig. 7: (a) DRAM chip diagram showing hierarchical, bank I/O bus with the PIMnet topology connectivity shown in red, (b) conventional DDR I/O bus (DQ) interface, and (c) proposed PIMnet I/O that splits the I/O bus into multiple channels.

PIMnet communication is enabled (i.e., PIMnet_en is asserted). The physical connectivity of the inter-bank is a ring topology in our PIMnet implementation. To minimize the impact on wire routing resources which are limited to 3 metal layers [77], [82], we leverage the existing I/O bus in modern DDR DRAM chips [1], [2], [4]. A block diagram of the internal, hierarchical DRAM I/O bus structure is shown in Figure 7(a) with the logical connectivity of the PIMnet topology overlaid on top. The DDR I/O bus used for normal memory access can be shared for inter-bank (PIMnet) communication since communication does not overlap with memory accesses or host-to-memory communication. We assume a baseline I/O bus of 64 bits [62] and the I/O bus is hierarchical with two banks or a bank group sharing one set of I/O bus and the bank group sharing a global I/O bus. The bank group I/O bus, commonly organized as a 64-bit bidirectional bus (Figure 7(b)) is partitioned into four 16-bit unidirectional channels for the PIMnet ring connectivity (Figure 7(c)) – to provide the four channels for In/Out from the East/West ports.

B. Inter-Chip Network

Inter-chip communication of PIMnet occurs through the existing DRAM pins (i.e. DQ pins). Since there are 8 DQ pins on a DDR4 DRAM chip, we partition them into two groups for PIMnet – 4 pins to send data to the buffer chip and 4 pins to receive data from the buffer chip. For normal memory operations, the DQ pins are used as a bidirectional bus, but for PIMnet, the bus is partitioned into two unidirectional channels. Similar to inter-bank channels, sharing of the DQ pins is possible since inter-chip communication does not overlap with any host-to-memory communication. We also assume a DRAM with a memory buffer chip in the middle of the DIMM [5] (Figure 6(b)) – thus, DQ pins are routed to the memory buffer chip before being driven off the DIMM through the memory channel bus. For PIMnet, the DQ pins routed to the memory buffer chip are connected to an 8×8 inter-chip crossbar switch that provides connectivity between the PIM chips on the same rank. In addition to the crossbar, a switch control unit is necessary to manage the communication between the PIM chips and the synchronization. The READY signals from each chip are collected and when all chips are ready, the START signal is sent to all DRAM chips to begin inter-chip communication.

Unlike conventional crossbar [23] that has hardware arbitration, the inter-chip crossbar is not hardware-controlled but managed or scheduled before PIM execution to avoid contention. When inter-chip communication is needed, the host sends control information to the switch control unit during PIM kernel launch. The control information includes the traffic pattern (e.g., communication type and scope) and message size. The switch control unit includes memory-mapped registers that contain switch configuration information for PIMnet communication between the chips. An example of inter-chip switch configuration is shown in Figure 8 for a 4×4 inter-chip switch with 4 DRAM chips for the All-to-all communication. To enable connectivity between all chips during inter-chip communication, 3 steps (or $N-1$ steps) are needed where N is the number of chips that are participating in the All-to-all communication.

C. Inter-Rank Network

The inter-rank interconnect (Figure 6(c)) shares a multi-drop DDR bus that exists in modern DRAM between the ranks; thus, building a "network" on top of a bus becomes a

Algorithm 1 AllReduce scheduling & addressing algorithm

$N_B,\ N_C,\ N_R$ ←Number of banks/chips/ranks
$I_B,\ I_C,\ I_R$ ←Bank/chip/rank ID
D ←Data size
$Addr_B$ ← Base address of communication
$T_{RS_B},\ T_{AG_B}$ ←Inter-bank ReduceScatter/AllGather time
$T_{RS_C},\ T_{AG_C}$ ←Inter-chip ReduceScatter/AllGather time
$T_{RS_R},\ T_{AG_R}$ ←Inter-rank ReduceScatter/AllGather time
procedure Schedule_AllReduce(domain, phase)
  if (domain == bank) **then**
    if (phase == RS) **then**
      offset = 0
      $Addr_s$ = start_address = $Addr_B + (\frac{D}{N_B} \times I_B)$
    else if (phase == AG) **then**
      offset = $T_{RS_B} + T_{RS_C} + T_{RS_R} + T_{AG_R} + T_{AG_C}$
      $Addr_s = Addr_B + (\frac{D}{N_B} \times ((I_B + N_B - 1)\%N_B))$
  else if (domain == chip) **then**
    if (phase == RS) **then**
      ...
    else if (phase == AG) **then**
      ...
  else if (domain == rank) **then**
    ...
  return offset, start_address

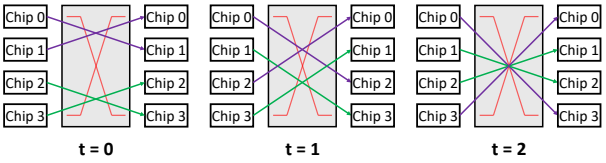


Fig. 8: An example configuration of 4×4 inter-chip switch for All-to-all communication across three time-steps.

challenge without physically modifying the bus. For PIMnet, we also leverage the DDR bus as a "broadcast" bus. However, since the source and destination are pre-determined based on the traffic pattern in PIMnet, once the source rank (DIMM) injects a packet, only the destination rank will receive the packet without any support from the host. By leveraging a bus, the communication is not necessarily as efficient as a point-to-point link since multiple communications cannot occur simultaneously on a bus. However, by avoiding communication through the host which utilizes the bus twice (once to reach the host and once to send data back to the PIM nodes), there is still approximately 2× improvement in bandwidth by enabling direct communication between the ranks (or DIMMs). Similar to inter-chip switch, inter-rank switches also has their own memory-mapped control registers that can be accessed by the host CPU and used to configure the switch.

D. Address Generation/Traffic Scheduling

Since PIM interconnect does not involve the host during communication, each PIM bank needs the memory address

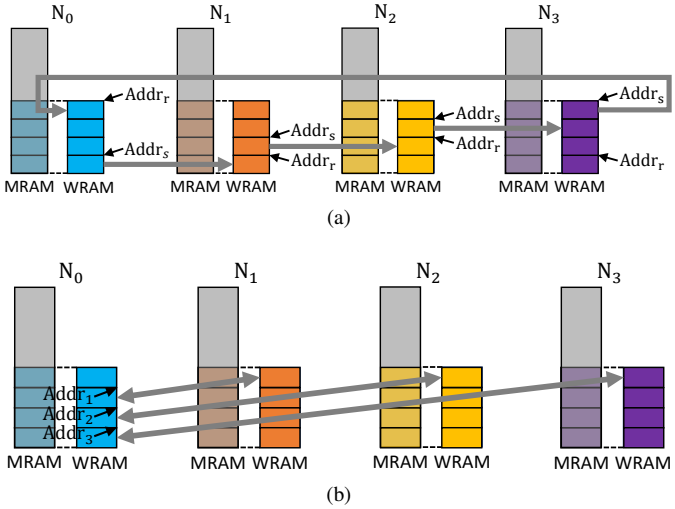


Fig. 9: High-level block diagram that illustrates the addresses that need to be generated for (a) AllReduce (Reduce-Scatter) and (b) All-to-all collective communication. The data for collective communication for PIMnet needs to be accessed from the local scratchpad memory. For simplicity, only addresses for node N_0 is shown in All-to-all communication.

of the data that is involved in the collective communications. Similar to GPU collective communication libraries (e.g., NCCL [3]), PIMnet collective communications occur across multiple "steps" and the number of steps depends on the (1) collective communication pattern, (2) number of PIM banks, and (3) PIM interconnect topology. Since these parameters are known prior to PIM execution, all addresses used during communication can be produced by the CPU during compilation. The address necessary depends on the collective communication and the number of banks participating in the communication and differs for each bank (or PIM node). For example, for Reduce-Scatter within a single DRAM chip, the address necessary for node N_i includes the start (read) address for data to be sent to node N_{i+1} ($Addr_s$) while another address is necessary to determine the data that needs to be combined with the data received from N_{i-1} ($Addr_r$) as shown in Figure 9(a). In our PIMnet implementation, the local scratchpad (i.e., WRAM) is used which contains a copy of the data that is located in the main memory (i.e., MRAM). For hierarchical AllReduce (or Reduce-Scatter) when the scope extends beyond a single chip, additional memory addresses are needed for the communication of data across the different hierarchies, based on the collective communication algorithm. For All-to-all communication, to minimize the memory requirement, PIMnet implements a pair-wise communication such that data can effectively be "swapped" between the pair of nodes and the data do not need to be stored in an intermediate location. For example, as shown earlier in Figure 8, the configuration for All-to-all for each step is based on pair-wise communication – e.g., if N_i is sending data to N_j, then, N_j also sends data to N_i. In Figure 9(b), an example of the addresses that need to be generated for All-to-all communication is shown for N_0. The

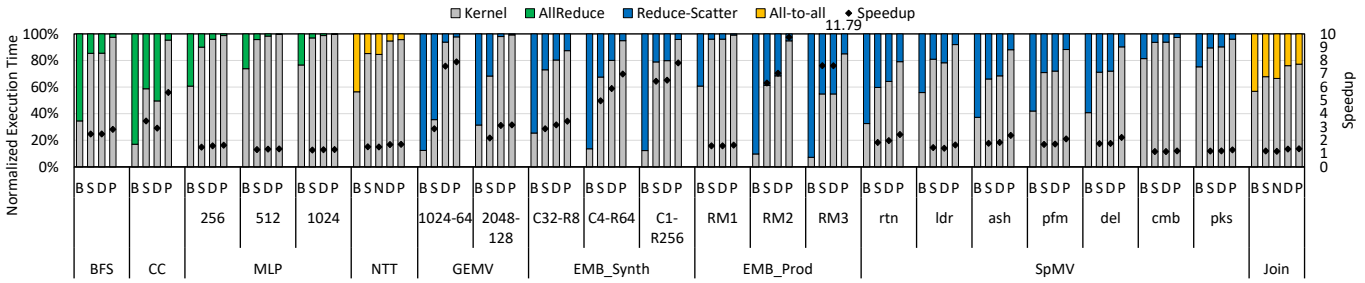


Fig. 10: Performance comparison and execution time breakdown of real-world applications. (**B**: Baseline PIM, **S**: Ideal software, **N**: NDPBridge, **D**: DIMM-Link, **P**: PIMnet.) Input graphs of `SpMV` are from SparseP [31].

Communication pattern	PIMnet implementation
Reduce-Scatter	Ring(inter-bank)→ Ring(inter-chip)→Broadcast(inter-rank)
AllGather	Broadcast(inter-rank)→ Ring(inter-chip)→Ring(inter-bank)
AllReduce	Ring(inter-bank)→ Ring(inter-chip)→Broadcast(inter-rank)→Ring(inter-chip)→Ring(inter-bank)
All-to-all	Ring(inter-bank)→ Permutation(inter-chip)→Unicast(inter-rank)
Broadcast	Ring(inter-chip)→ Broadcast(inter-rank)→Ring(inter-bank)

TABLE V: Collective communication primitives and their implementation on PIMnet.

address that needs to be generated is proportional to N or the number of nodes participating in the All-to-all communication. For example, $Addr_1$ corresponds to the send address for data to N_1 from N_0 while $Addr_2$ corresponds to data that will be sent to N_2 etc.

In addition to the local memory address, the latency (or timing) information needs to be known ahead of time to ensure deterministic data movement. Algorithm 1 provides a summary of the algorithm for calculating the address and timing information for each PIM bank during an AllReduce communication. The address is used to determine the local memory address while the timing information is used to determine when the collective communication phase should begin. The exact value of these two parameters depends on the phase of the collective communication (e.g., Reduce-Scatter (RS) and All-Gather (AG) for AllReduce) as well as the hierarchy (e.g., bank, chip, rank) and latency parameters, including latency to other bank/chip/ranks. Timing *offset* value is necessary to determine when communication can begin – e.g., for `RS`, communication can begin when synchronization completes but for `AG`, communication needs to wait until the previous `RS` phase completes. Address generation and traffic scheduling of All-to-all is needed, similar to AllReduce except more communication steps are needed for scheduling of inter-chip (and rank) permutation. From the programmer's perspective, the detailed address and timing calculation procedure during compilation does not need to be exposed to the programmer and only providing high-level library function (e.g. PIMnet_AllReduce() or PIMnet_Alltoall()) is sufficient to enable programmers to utilize PIMnet functionalities.

E. Routing

Given a topology, the routing algorithm determines the path taken by a packet and is an important aspect of any interconnection network [23]. However, since communication is scheduled in PIMnet with a known *physical* topology and a pre-determined communication algorithm (or the *logical* topology [18]), no hardware-supported routing is needed as the routing or movement of data is based on the "logical" topology of the collective communication. Implementation of different collective communication for PIMnet is summarized in Table V. For Reduce-Scatter, the logical ring communication algorithm is used for the inter-bank and the inter-chip hierarchy while "broadcast" is used to reduce across the different ranks. Since the PIM banks within the same DRAM chip perform ring-based communication, each PIM bank only communicates with its adjacent bank. For inter-chip communication, the inter-chip switch is configured to form a ring for AllReduce and Reduce-Scatter, or a different permutation of source-destination connection is established across each time step for All-to-all communication (Figure 8). Similarly, the inter-rank switch accepts and rearranges data from remote ranks based on static schedule defined by the communication pattern. Therefore, once synchronization occurs, no congestion or contention within the network occurs. In this work, we focused mostly on two commonly used collective communications including AllReduce (which consists of Reduce-Scatter and AllGather) and All-to-all for PIMnet. However, PIMnet can be extended to support other collective communications – e.g., for collectives with N-to-1 communication (e.g., Reduce, Gather), a single DPU can be used to enable such communication.

VI. Evaluation

A. Methodology

A detailed processing-in-memory (PIM) simulator [43], validated against UPMEM PIM hardware, was used in our evaluation of PIMnet. The simulator was modified to model PIMnet in our evaluation and the PIM system configuration is summarized in Table VI. Both synthetic collective communication and real-world workloads were used in our evaluation. Description of the workloads are summarized in Table VII, including the workload inputs as well as the type of collective communication used in the workloads. Our evaluations also include two kernels that are commonly found in emerging

	Configurations
Host CPU	16 Cores @ 4GHz,32KB L1 I/D,256KB L2,8MB L3
PIM Core	350MHz,24KB IRAM,64KB WRAM
Memory System	DDR4-2400, 4 ranks/channel
Host-PIM bandwidth	4.74GB/s (PIM→CPU), 6.68GB/s (CPU→PIM), 16.88GB/s (CPU→PIM broadcast) [39]
Buffer chip-PIM bandwidth	19.2GB/s ([89])

TABLE VI: System configuration for PIMnet evaluation.

Workload	Description	Comm.	Input/Config
Embedding table lookup (EMB) [65]	Table lookup in Deep Learning Recommendation Model	RS	Pooling size 8, Batch size 256
Number Theoretic Transform (NTT) [88]	Key operation in Homomorphic Encryption (HE)	A2A	$N = 2^{16}$
Matrix-vector multiplication (GEMV) [39]	Common operation in linear algebra and neural networks	RS	1024 × 64, 2048 × 128
Multi-layer perception (MLP) [39]	Fully connected layers used in neural networks	AR	256 × 256, 512 × 512, 1024 × 1024
Sparse matrix-vector Multiplication (SpMV) [31]	GEMV operation using sparse weight matrix	RS	DBCOO partitioning, 32 vertical partitions
Breadth-first search (BFS) [39]	Graph traverse algorithm from a single start vertex	AR	*log-gowalla*
Connected components (CC) [39]	Find maximal subsets of a graph where there is a path between any two vertices within this set	AR	*log-gowalla*
Hash join (Join) [61]	Join operation used in database applications	A2A	64M tuples

TABLE VII: Description of real-world workloads used in the evaluation. RS: Reduce-Scatter, AR: AllReduce, A2A: All-to-all collective communication patterns.

workloads – homomorphic encryption (HE) and deep-learning recommendation models (DLRM). For HE, we evaluate the potential benefits of PIMnet for NTT (Number Theoretic Transform) [88] that is widely used and requires All-to-all communication between the nodes. In addition, embedding tables are widely used in DLRM for categorical data and we evaluate both synthetic embedding tables (EMB_Synth) and production-level (EMB_Prod) embedding tables (RM1, RM2, RM3) [63]. Since we focused only on the embedding lookup within DLRM, Reduce-scatter communication is necessary. For the synthetic embedding tables, Cx-Ry embedding table partitioning was used where x is the amount of column-wise partitions and y is the amount of row-wise partitioning of the embedding tables [49], assuming 4M entries, 64 embedding dimension, and 8 of polling factor.

Performance is compared with a baseline PIM system with host communication (**B**) and DIMM-Link [89] (**D**), which proposed direct point-to-point connections between the different DRAM ranks. To ensure a fair comparison, we assume the same "global" (or inter-rank) bandwidth between DIMM-Link and PIMnet and ignore any overhead from the bridge

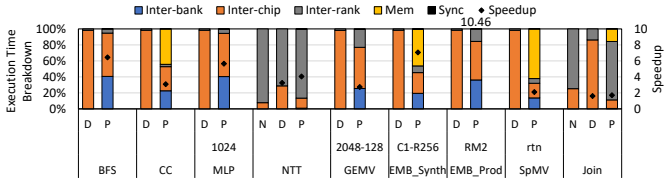


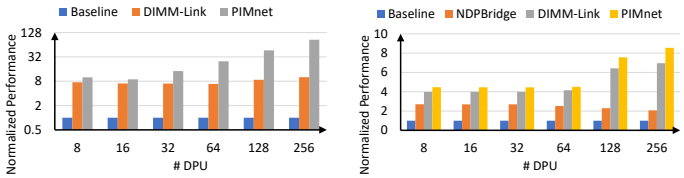
Fig. 11: Breakdown of PIM communication time across evaluated workloads.

required in DIMM-Link. We also compare against an ideal software collective communication implementation (**S**) and NDPBridge [85] (**N**) for All-to-all collective communication since it does not have support for AllReduce. To ensure a fair comparison, we assume DIMM-Link [89] provides bank-level PIM and assume each rank can perform collective communication in parallel within its buffer chip for its local data. Thus, the compute implementation across the different alternative implementations is constant but the key difference is how collective communication is performed. The initial evaluations used 256 DPUs on a single memory channel and additional scalability results are also presented.

Note that some of the workloads that we evaluate can be implemented without collective communication (e.g., GEMV, MLP). However, alternative implementations or parallelism approaches lead to different trade-offs and some implementations do require collective communication. For example, data parallelism can be used for GEMV/MLP which duplicates data across the PIM nodes and does not require communication. However, if tensor parallelism, similar to prior work [67], [72], is used, collective communication is needed based on the tensor partitioning.

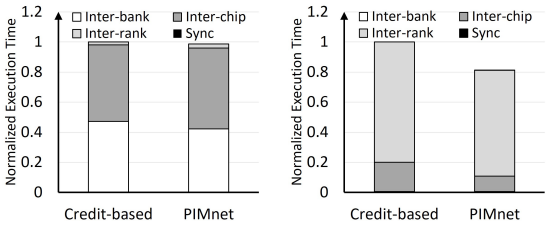
B. Results

Application Performance Comparison: Figure 10 shows the performance benefits of PIMnet on real-world applications. In the baseline system, the performance of the graph workloads (i.e, BFS, CC) is limited by the low communication bandwidth through the host as AllReduce represents up to 83% of the total execution time. However, PIMnet reduces the communication time to 5% and results in a 5.6× speedup for CC. The larger amount of communication for CC (compared to BFS) results in higher performance improvement. Compared to the baseline, alternative implementations (**S**, **D**) provide improvement compared to the baseline, but the benefits are limited. MLP and NTT result in a relatively small speedup due to a high proportion of compute time in the baseline PIM system. The large compute time in these workloads is caused by the low multiplication compute throughput of UPMEM data processing units (DPUs) [25] as UPMEM does not support native multiplication but is emulated through software. GEMV involves the same amount of multiplication per input as MLP; however, it results in higher speedup than MLP since communication occurs after every layer. For EMB, RM3 results in the biggest improvement from PIMnet because of a higher amount of communication and a relatively low amount of



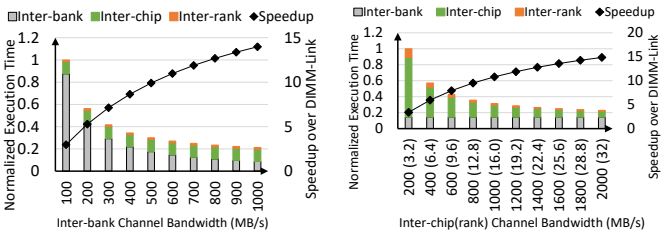
(a) AllReduce (b) All-to-all

Fig. 12: Scalability comparison for (a) AllReduce and (b) All-to-all communication, with results normalized to the baseline.



(a) AllReduce (b) All-to-all

Fig. 13: Execution time comparison of credit-based flow control and PIM-controlled traffic scheduling for (a) AllReduce and (b) All-to-all collective communication.



(a) Inter-bank (b) Inter-chip/rank

Fig. 14: PIMnet's AllReduce performance scalability over different (a) inter-bank and (c) inter-chip/rank channel bandwidth.

memory access. Retrieval and accumulation of partial sums in 2D-partitioned `SpMV`, before output retrieval from the host, is accelerated by performing Reduce-Scatter using PIMnet and thus, 2.43× speedup compared to the host-managed communication. `Join` operation in bank-level PIM system incurs an All-to-all communication across all PIM banks after global partitioning of tuples [61] and PIMnet provides 36% improve in performance with 64M tuples compared to the baseline.

PIM Communication Analysis: Figure 11 shows the breakdown of PIM communication for representative configuration of each workload and *PIM communication* speedup of PIMnet normalized to DIMM-Link (**D**) is shown, except for `NTT` and `Join` which is normalized to NDPBridge (**N**). The PIM communication time is broken up into the three PIM hierarchy (inter-bank, inter-chip, and inter-rank) as well overhead for PIMnet, including `Sync` and `Mem`. `Sync` refers to the synchronization overhead while `Mem` refers to the memory transfer overhead between the scratchpad memory (i.e., WRAM) and DRAM bank (i.e., MRAM) when the communication amount (or data) cannot fit within the scratchpad and has to be copied from the DRAM memory bank. Prior work (**D**) do not have support for direct inter-bank communication but with PIMnet, not only is direct inter-bank communication possible but for workloads with AllReduce or Reduce-Scatter, PIMnet significantly improves performance by performing inter-bank communication in parallel. For some of the workloads (`CC`, `EMB_Synth`, `SpMV`, and `Join`), the overhead from `Mem` is non-negligible but PIMnet is still able to provide communication performance improvement over DIMM-Link.

Collective Communication Scalability: Scalability analysis of AllReduce and All-to-all collective communication is shown in Figure 12 as the number of DPUs are increased from 8 to 256 DPUs. For both analysis, we assume weak scaling with a message size of 32KB and the results are normalized to the performance of baseline PIM system for each number of DPUs. For AllReduce, as discussed earlier, local AllReduce can be done in parallel and provides more speedup as the number of DPUs increases. The benefit of All-to-All with PIMnet is limited as the number of DPUs increases since All-to-All requires global communication of all of its data. However, compared to NDPBridge, PIMnet does not require communication to/from the host and thus, provides better scalability. In addition, PIMnet provides some performance improvement over DIMM-Link because of direct data exchange between PIM banks without the overhead of re-arranging data.

Comparison to Hardware-based Flow Control: PIMnet is a *PIM-controlled* interconnect where data moves across the interconnection network by exploiting the deterministic traffic pattern. While this greatly simplifies the microarchitecture, it can potentially have a performance impact because of the synchronization overhead. To analyze the impact of static-scheduled traffic management against a fully functional interconnection network architecture, we implemented PIMnet's topology and static-scheduled traffic model in the cycle-accurate NoC simulator Booksim 2.0 [46]. We then compared the performance of collective communication. To measure the performance overhead of statically scheduled communication, we used execution time data of individual DPUs from the real UPMEM system. This data was used as input to determine when communication is ready for each node. In credit-based flow control, each DPU starts communication right after finishing its computation, and PIMnet starts communication simultaneously after the last computation finishes across all of the DPUs – thus, adding some synchronization overhead. As the result in Figure 13 shows, for AllReduce, the performance of both approaches is very similar within a 1% reduction of the execution time of PIMnet. However, for All-to-all communication, PIM-controlled scheduling (or PIMnet) shows an 18.7% time reduction over credit-based flow control. Since there are many independent point-to-point communications in All-to-all, heavy contention occurs at the inter-chip crossbar, and communication time actually *increases*.

Hardware Overhead of PIMnet: We implemented the PIMnet in Verilog, including the PIMnet stop and the address

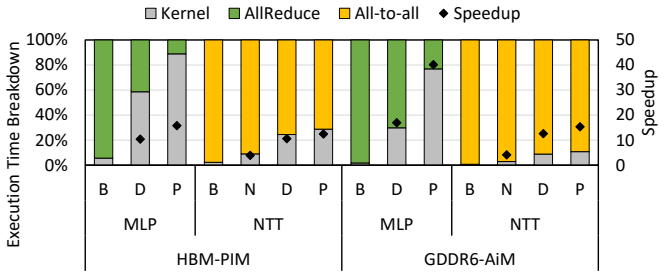


Fig. 15: PIMnet performance benefit with different compute implementations (HBM-PIM [59] and GDDR6-AiM [58]).

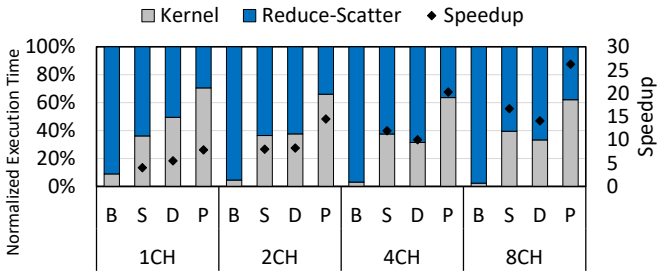


Fig. 16: Performance comparsion of embedding table lookup with memory channel scaling.

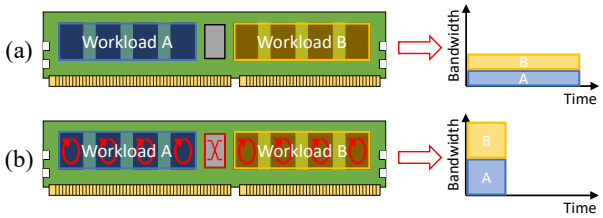


Fig. 17: PIM bank allocation for multi-tenancy using spatial mapping with (a) host-based communication, and (b) PIMnet.

generator, and synthesized using OpenROAD [10] with 45nm technology (Nangate45). The metal layers were restricted to 3 layers, similar to DRAM technology. The overhead introduced was very negligible – an increase of only 0.09% area, compared to a baseline PIM bank [25] while the power overhead was estimated to be only 1.6%. Compared to a traditional NoC router (e.g., a ring router), PIMnet stop results in over 60× reduction in area. Inter-chip and inter-rank switch was also synthesized and resulted in 0.013mm^2 and 17mW of power consumption overhead, which is negligible compared to buffer chip resource [49]. One source of performance overhead for PIMnet is the synchronization overhead (i.e., `READY/START` signals) which is proportional to the propagation latency across the PIMnet. In the PIMnet system that we consider, the worst-case propagation latency across PIMnet is estimated to be approximately 15 ns – or approximately 6 PIM (DPU) cycles). Considering even a small message size (1KB) AllReduce across 256 DPUs can take over 1000 cycles, the synchronization overhead is relatively small.

Bandwidth Scaling Analysis of PIMnet: One important factor in PIMnet performance is the interconnect bandwidth. In Figure 14(a), the inter-bank PIMnet channel bandwidth is varied from 0.1 to 1 GB/s. In this work we assumed inter-bank bandwidth of 0.7 GB/s but even if the bandwidth is reduced to 0.1 GB/s, PIMnet can still outperform DIMM-Link by 3× because of the bandwidth parallelism that can be exploited across the different DRAM chips. In Figure 14(b), the inter-bank bandwidth is fixed at 0.7 GB/s and provides the scalability analysis of inter-chip and inter-rank channel bandwidth. Since PIMnet enables hierarchical AllReduce and reduces the amount of global data communicated across the different ranks, PIMnet is still able to outperform DIMM-Link even with a smaller amount of global (inter-chip/inter-rank) bandwidth.

Multi-channel Scaling Analysis: Figure 16 shows `EMB_Synth` performance trend of different methods with different number of PIM channels. Since the scope of PIMnet is interconnecting PIM banks within the same memory channel, data transfer across different memory channels needs to go through the host CPU. However, PIMnet can greatly reduce the transfer amount that goes to the host CPU by performing channel-wise data reduction. Therefore, the overhead of host communication is lower for PIMnet as the number of channels increases – resulting in higher speedup compared to the baseline system with more memory channels.

Alternative PIM implementation: This work was based on UPMEM PIM architecture because of its programmable compute but other PIM architectures [58], [59] have different compute characteristics (and throughput); however, the need for inter-PIM communication also exists for other PIM architectures. One key difference of the other PIM, compared with UPMEM, is the higher compute throughput (or FLOPS) provided by dedicated hardware logic, including HBM-PIM [59] and GDDR6-AiM [58]. In Figure 15, we provide an analysis of the potential benefit of PIMnet if the compute throughput of PIM is increased to match that of alternative PIMs. For the workloads, we selected the two most compute-intensive tasks in our evaluations: `MLP` and `NTT`. The results show that PIMnet has the potential to provide more speedup. For example, the performance improvement from PIMnet on `MLP` was only 1.3× in Figure 10 but if we assume PIM logic with custom multiply/accumulate units (i.e., GDDR6-AiM [58] PIM) that provides 180× higher compute throughput than UPMEM [39], the benefit of PIMnet increases to approximately 40×. The next generation of DPUs from UPMEM [68] is described to have support for native floating point operations, offering more than several orders of magnitude higher FLOPS (5-8TFLOPS/chip) compared to the current DPUs. As a result, the benefits from PIMnet will likely be higher with future DPUs from UPMEM.

Multi-Tenancy: Current PIM assumes a single workload or a single user but it is possible future PIMs can have support for multi-tenancy [50]. While PIMnet can support multi-tenancy across the memory hierarchy, the key benefit of PIMnet is the ability to provide bandwidth isolation across the different tenants. For example, in the baseline system, spatial mapping

of multiple workloads is shown in Figure 17(a); however, the bandwidth to/from the host need to be shared for PIM-to-PIM communication and degrades performance. In comparison, with PIMnet (Figure 17(b)), the PIM-to-PIM collective communication can be isolated and not only result in high bandwidth but also exploits PIM bandwidth parallelism and accelerate both workloads.

VII. Related work

There have been many prior work on accelerating different workloads on the UPMEM PIM architecture, including machine learning workloads [24], [34], [38], [60], transcendental functions [44], sequence alignment [26], graph pattern matching [13], homomorphic operations [35], sparse matrix-vector multiplications [32], and join algorithms [61]. In addition, to accelerate data-intensive workloads that are memory-intensive, different processing-in-memory architectures have been proposed, including PIM architecture for recommendation systems and embedding tables [49], [56], [57], [71] as well as fully homomorphic encryption (FHE) and various kernels within FHE [36], [37], [66], [75], [90]. In this work, we target similar workloads but unlike prior work, this work explores potential benefits from introducing a *domain*-specific (or PIM-specific) interconnection network to accelerate collective communication. Other work have also identified potential limitations of PIM scalability caused by the PIM-to-PIM communication overhead through the host CPU [16], [39], [47] and software-based optimization of collective communications have been proposed [67]. In comparison, this is one of the first work to propose a dedicated (hardware) interconnect to accelerate collective communication for PIM architectures.

To scale PIM architecture, different multi-PIM systems have been proposed [70] including heterogeneous NPU-PIM systems [41], [81]. The performance benefits of these systems are often determined by the inter-PIM communication and collective communication; however, the DRAM module-to-module constraints are different from a PIM bank-to-bank communication. To provide communication between DIMMs without communicating through the host, different architectures have been proposed including AIM [20] and DIMM-Link [89] which have proposed external connector or a bridge to interconnect the DIMMs together. ABC-DIMM [83] proposed to re-use the memory bus to broadcast traffic and enable DIMM-to-DIMM communication. NDPBridge [85] proposes hardware "bridges" across the DRAM hierarchy to accelerate message transfer between memory banks. In comparison, this work addresses the challenge of collective communication in PIM-to-PIM data movement and proposes a PIM-*controlled* interconnect architecture.

There has been a significant amount of work done on different types of interconnection networks, including network-on-chip [22], [23] to large-scale networks [53]. In addition, interconnection networks have been leveraged to enable scalable near-memory or near-data processing architectures. Memory channel network [11] proposes a practical near-memory processing system where each near-memory processor is viewed as "node" that runs an OS. While no hardware changes are necessary, it can introduce additional performance overhead. Memory-centric network for CPUs [51] and CPU-GPU systems [52] explore interconnect between the memory modules and CPU/GPU as well as intra-module interconnect. The memory-centric organization has been exploited for PIM or NDP accelerations [9], [27], [29], [33], [76]. However, the constraints of PIMnet (and PIM-to-PIM communication) that we exploit present unique opportunities and challenges compared to other types of interconnection networks. Active routing [42] proposes computation within the network for near-data processing; however, PIMnet does not require similar type of "routing." Modern PIM architectures [58] often have broadcast bus structures where common data can be shared between different banks for computation; as a result, different approaches to accelerating broadcasting has been proposed (e.g., ring-based broadcasting for Transformers [91]). PIMnet presented in this work represents one possible implementation and it remains to be seen how PIMnet can be optimized.

VIII. Conclusion

In this work, a *domain*-specific interconnection network for PIM nodes, referred to as PIMnet, was proposed to enable efficient collective communication between PIM banks and provide scalable PIM architecture. PIMnet exploits the memory packaging hierarchy to enable direct data movement for PIM-to-PIM communication without sending data through the host. In addition, PIMnet leverages communication characteristics of collective communication to schedule communication and create a PIM-*controlled* interconnection network architecture to accelerate collective communication.

Acknowledgement

We thank the anonymous reviewers for their comments and feedback that helped to improve the paper and thank Hyeonwoo Sung for his support in implementing PIMnet in the simulator. This work was supported in part by NSF CNS 2312275 and 2312276, NSF IUCRC Center for Hardware and Embedded Systems Security and Trust, IITP-RS202300228255, NRF-2023R1A2C200422912, and IITP-RS202400402898. Additionally, we acknowledge the financial assistance from grants RYC2021-031966-I and CNS2023-144241 funded by MICIU/AEI/10.13039/501100011033 and the "European Union NextGenerationEU/PRTR", and project grant PID2020-112827GB-I00 funded by MICIU/AEI/10.13039/501100011033.

References

[1] "8gb c-die ddr4 sdram x16," https://download.semiconductor.samsung.com/resources/user-manual/x16%20only_8G_C_DDR4_Samsung_Spec_Rev1.5_Apr.17.pdf.

[2] "Micron ddr4 sdram," https://www.micron.com/-/media/client/global/documents/products/datasheet/dram/ddr4/8gb_ddr4_sdram.pdf.

[3] "NCCL documentation," https://docs.nvidia.com/deeplearning/nccl/user-guide/docs/index.html, accessed: 2023-04-28.

[4] "Sk hynix ddr4," https://product.skhynix.com/products/dram/ddr/ddr4sd.go.

[5] "Sk hynix ddr4 lrdimm chip datasheet," https://product.skhynix.com/products/dram/module/ddr4dm.go.

[6] "UPMEM documentation," https://sdk.upmem.com/2021.3.0/.

[7] D. Abts and J. Kim, "The case for domain-specific networks," in *2023 IEEE Symposium on High-Performance Interconnects (HOTI)*, 2023, pp. 49–52.

[8] D. Abts, G. Kimmell, A. Ling, J. Kim, M. Boyd, A. Bitar, S. Parmar, I. Ahmed, R. DiCecco, D. Han, J. Thompson, M. Bye, J. Hwang, J. Fowers, P. Lillian, A. Murthy, E. Mehtabuddin, C. Tekur, T. Sohmers, K. Kang, S. Maresh, and J. Ross, "A software-defined tensor streaming multiprocessor for large-scale machine learning," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 567–580. [Online]. Available: https://doi.org/10.1145/3470496.3527405

[9] J. Ahn, S. Hong, S. Yoo, O. Mutlu, and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, 2015, pp. 105–117.

[10] T. Ajayi and D. Blaauw, "Openroad: Toward a self-driving, open-source digital layout implementation tool chain," in *Proceedings of Government Microcircuit Applications and Critical Technology Conference*, 2019.

[11] M. Alian, S. W. Min, H. Asgharimoghaddam, A. Dhar, D. K. Wang, T. Roewer, A. McPadden, O. O'Halloran, D. Chen, J. Xiong, D. Kim, W.-m. Hwu, and N. S. Kim, "Application-transparent near-memory processing architecture with memory channel network," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2018, pp. 802–814.

[12] D. H. Bailey, "FFTs in external or hierarchical memory," in *Supercomputing '89: Proceedings of the 1989 ACM/IEEE Conference on Supercomputing*, 1989, pp. 234–242.

[13] S. Cai, B. Tian, H. Zhang, and M. Gao, "Pimpam: Efficient graph pattern matching on real processing-in-memory hardware," *Proc. ACM Manag. Data*, vol. 2, no. 3, May 2024. [Online]. Available: https://doi.org/10.1145/3654964

[14] D. Cardwell and F. Song, "An extended roofline model with communication-awareness for distributed-memory hpc systems," in *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, ser. HPCAsia '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 26–35. [Online]. Available: https://doi.org/10.1145/3293320.3293321

[15] E. Chan, M. Heimlich, A. Purkayastha, and R. van de Geijn, "Collective communication: theory, practice, and experience," *Concurrency and Computation: Practice and Experience*, vol. 19, no. 13, pp. 1749–1783, 2007. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1206

[16] J. Chen, J. Gomez-Luna, I. E. Hajj, Y. Guo, and O. Mutlu, "Simplepim: A software framework for productive and efficient processing-in-memory," in *2023 32nd International Conference on Parallel Architectures and Compilation Techniques (PACT)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2023, pp. 99–111. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/PACT58117.2023.00017

[17] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.

[18] S. Cho, H. Son, and J. Kim, "Logical/physical topology-aware collective communication in deep learning training," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2023, pp. 56–68.

[19] J. Choquette, W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky, "Nvidia a100 tensor core gpu: Performance and innovation," *IEEE Micro*, vol. 41, no. 2, pp. 29–35, 2021.

[20] J. Cong, Z. Fang, M. Gill, F. Javadi, and G. Reinman, "Aim: Accelerating computational genomics through scalable and noninvasive accelerator-interposed memory," in *Proceedings of the International Symposium on Memory Systems*, ser. MEMSYS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 3–14. [Online]. Available: https://doi.org/10.1145/3132402.3132406

[21] A. Dakkak, C. Li, J. Xiong, I. Gelado, and W. Hwu, "Accelerating reduction and scan using tensor core units," in *Proceedings of the ACM International Conference on Supercomputing*, 2019, pp. 46–57.

[22] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," ser. DAC '01. New York, NY, USA: ACM, 2001, pp. 684–689. [Online]. Available: http://doi.acm.org/10.1145/378239.379048

[23] W. Dally and B. Towles, "Principles and practices of interconnection network," 01 2004.

[24] P. Das, P. R. Sutradhar, M. Indovina, S. M. P. Dinakarrao, and A. Ganguly, "Implementation and evaluation of deep neural networks in commercially available processing in memory hardware," in *2022 IEEE 35th International System-on-Chip Conference (SOCC)*, 2022, pp. 1–6.

[25] F. Devaux, "The true processing in memory accelerator," in *2019 IEEE Hot Chips 31 Symposium (HCS), Cupertino, CA, USA, August 18-20, 2019*. IEEE, 2019, pp. 1–24. [Online]. Available: https://doi.org/10.1109/HOTCHIPS.2019.8875680

[26] S. Diab, A. Nassereldine, M. Alser, J. Gómez Luna, O. Mutlu, and I. El Hajj, "A framework for high-throughput sequence alignment using real processing-in-memory systems," *Bioinformatics*, vol. 39, no. 5, p. btad155, 2023.

[27] M. Drumond, A. Daglis, N. Mirzadeh, D. Ustiugov, J. Picorel, B. Falsafi, B. Grot, and D. Pnevmatikatos, "The mondrian data engine," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, p. 639–651, jun 2017. [Online]. Available: https://doi.org/10.1145/3140659.3080233

[28] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, 2011, pp. 365–376.

[29] M. Gao, G. Ayers, and C. Kozyrakis, "Practical near-data processing for in-memory analytics frameworks," in *2015 International Conference on Parallel Architecture and Compilation (PACT)*, 2015, pp. 113–124.

[30] M. Gao, M. Coletti, R. B. Davidson, R. Prout, S. Abraham, B. Hernández, and A. Sedova, "Proteome-scale deployment of protein structure prediction workflows on the summit supercomputer," in *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2022, pp. 206–215.

[31] C. Giannoula, I. Fernandez, J. Gómez-Luna, N. Koziris, G. Goumas, and O. Mutlu, "Sparsep: Towards efficient sparse matrix vector multiplication on real processing-in-memory systems," 2022.

[32] C. Giannoula, I. Fernandez, J. G. Luna, N. Koziris, G. Goumas, and O. Mutlu, "Sparsep: Towards efficient sparse matrix vector multiplication on real processing-in-memory architectures," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 6, no. 1, Feb. 2022. [Online]. Available: https://doi.org/10.1145/3508041

[33] C. Giannoula, N. Vijaykumar, N. Papadopoulou, V. Karakostas, I. Fernandez, J. Gómez-Luna, L. Orosa, N. Koziris, G. Goumas, and O. Mutlu, "Syncron: Efficient synchronization support for near-data-processing architectures," 2021.

[34] K. Gogineni, S. S. Dayapule, J. Gómez-Luna, K. Gogineni, P. Wei, T. Lan, M. Sadrosadati, O. Mutlu, and G. Venkataramani, "Swiftrl: Towards efficient reinforcement learning on real processing-in-memory systems," in *2024 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2024, pp. 217–229.

[35] H. Gupta, M. Kabra, J. Gómez-Luna, K. Kanellopoulos, and O. Mutlu, "Evaluating homomorphic operations on a real-world processing-in-memory system," in *2023 IEEE International Symposium on Workload Characterization (IISWC)*, 2023, pp. 211–215.

[36] S. Gupta, R. Cammarota, and T. Šimunić, "Memfhe: End-to-end computing with fully homomorphic encryption in memory," *ACM Trans. Embed. Comput. Syst.*, vol. 23, no. 2, Mar. 2024. [Online]. Available: https://doi.org/10.1145/3569955

[37] S. Gupta and T. Š. Rosing, "Accelerating fully homomorphic encryption with processing in memory," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 1335–1338.

[38] J. Gómez-Luna, Y. Guo, S. Brocard, J. Legriel, R. Cimadomo, G. F. Oliveira, G. Singh, and O. Mutlu, "Evaluating machine learningworkloads on memory-centric computing systems," in *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2023, pp. 35–49.

[39] J. Gómez-Luna, I. E. Hajj, I. Fernandez, C. Giannoula, G. F. Oliveira, and O. Mutlu, "Benchmarking a new paradigm: Experimental analysis and characterization of a real processing-in-memory system," *IEEE Access*, vol. 10, pp. 52 565–52 608, 2022.

[40] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. IEEE Press, 2016, p. 243–254. [Online]. Available: https://doi.org/10.1109/ISCA.2016.30

[41] G. Heo, S. Lee, J. Cho, H. Choi, S. Lee, H. Ham, G. Kim, D. Mahajan, and J. Park, "Neupims: Npu-pim heterogeneous acceleration for batched llm inferencing," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 722–737. [Online]. Available: https://doi.org/10.1145/3620666.3651380

[42] J. Huang, R. Reddy Puli, P. Majumder, S. Kim, R. Boyapati, K. H. Yum, and E. J. Kim, "Active-routing: Compute on the way for near-data processing," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 674–686.

[43] B. Hyun, T. Kim, D. Lee, and M. Rhu, " Pathfinding Future PIM Architectures by Demystifying a Commercial PIM Technology ," in *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Mar. 2024, pp. 263–279. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/HPCA57654.2024.00029

[44] M. Item, G. F. Oliveira, J. Gomez-Luna, M. Sadrosadati, Y. Guo, and O. Mutlu, "Transpimlib: Efficient transcendental functions for processing-in-memory systems," in *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. Los Alamitos, CA, USA: IEEE Computer Society, apr 2023, pp. 235–247. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/ISPASS57527.2023.00031

[45] Z. Jia, M. Maggioni, B. Staiger, and D. P. Scarpazza, "Dissecting the nvidia volta gpu architecture via microbenchmarking," *arXiv preprint arXiv:1804.06826*, 2018.

[46] N. Jiang, D. U. Becker, G. Michelogiannakis, J. Balfour, B. Towles, D. E. Shaw, J. Kim, and W. J. Dally, "A detailed and flexible cycle-accurate network-on-chip simulator," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 86–96.

[47] G. Jonatan, H. Cho, H. Son, X. Wu, N. Livesay, E. Mora, K. Shivdikar, J. L. Abellán, A. Joshi, D. Kaeli, and J. Kim, "Scalability limitations of processing-in-memory using real system evaluations," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 8, no. 1, Feb. 2024. [Online]. Available: https://doi.org/10.1145/3639046

[48] A. B. Kahng, B. Lin, and S. Nath, "Orion3.0: A comprehensive noc router estimation tool," *IEEE Embedded Systems Letters*, vol. 7, no. 2, pp. 41–45, 2015.

[49] L. Ke, U. Gupta, B. Y. Cho, D. Brooks, V. Chandra, U. Diril, A. Firoozshahian, K. Hazelwood, B. Jia, H.-H. S. Lee, M. Li, B. Maher, D. Mudigere, M. Naumov, M. Schatz, M. Smelyanskiy, X. Wang, B. Reagen, C.-J. Wu, M. Hempstead, and X. Zhang, "Recnmp: Accelerating personalized recommendation with near-memory processing," in *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ser. ISCA '20. IEEE Press, 2020, p. 790–803. [Online]. Available: https://doi.org/10.1109/ISCA45697.2020.00070

[50] D. Kim, T. Kim, I. Hwang, T. Park, H. Kim, Y. Kim, and Y. Park, "Virtual pim: Resource-aware dynamic dpu allocation and workload scheduling framework for multi-dpu pim architecture," in *2023 32nd International Conference on Parallel Architectures and Compilation Techniques (PACT)*. Los Alamitos, CA, USA: IEEE Computer Society, oct 2023, pp. 112–123. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/PACT58117.2023.00018

[51] G. Kim, J. Kim, J. H. Ahn, and J. Kim, "Memory-centric system interconnect design with hybrid memory cubes," in *Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques*, 2013, pp. 145–155.

[52] G. Kim, M. Lee, J. Jeong, and J. Kim, "Multi-gpu system design with memory networks," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 2014, pp. 484–495.

[53] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *2008 International Symposium on Computer Architecture*, 2008, pp. 77–88.

[54] S. Kim, J. Kim, M. J. Kim, W. Jung, J. Kim, M. Rhu, and J. H. Ahn, "Bts: An accelerator for bootstrappable fully homomorphic encryption," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 711–725.

[55] Y.-C. Kwon, J. Lee, Suk Han fhand Lee, S.-H. Kwon, J. M. Ryu, J.-P. Son, O. Seongil, H.-S. Yu, H. Lee, S. Y. Kim, Y. Cho, J. G. Kim, J. Choi, H.-S. Shin, J. Kim, B. Phuah, H. Kim, M. J. Song, A. Choi, D. Kim, S. Kim, E.-B. Kim, D. Wang, S. Kang, Y. Ro, S. Seo, J. Song, J. Youn, K. Sohn, and N. S. Kim, "25.4 a 20nm 6gb function-in-memory dram, based on hbm2 with a 1.2tflops programmable computing unit using bank-level parallelism, for machine learning applications," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 64, 2021, pp. 350–352.

[56] Y. Kwon, Y. Lee, and M. Rhu, "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*, ser. MICRO '52. New York, NY, USA: Association for Computing Machinery, 2019, p. 740–753. [Online]. Available: https://doi.org/10.1145/3352460.3358284

[57] Y. Kwon, Y. Lee, and M. Rhu, "Tensor casting: Co-designing algorithm-architecture for personalized recommendation training," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 235–248.

[58] S. Lee, K. Kim, S. Oh, J. Park, G. Hong, D. Ka, K. Hwang, J. Park, K. Kang, J. Kim, J. Jeon, N. Kim, Y. Kwon, K. Vladimir, W. Shin, J. Won, M. Lee, H. Joo, H. Choi, J. Lee, D. Ko, Y. Jun, K. Cho, I. Kim, C. Song, C. Jeong, D. Kwon, J. Jang, I. Park, J. Chun, and J. Cho, "A 1ynm 1.25v 8gb, 16gb/s/pin gddr6-based accelerator-in-memory supporting 1tflops mac operation and various activation functions for deep-learning applications," in *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, vol. 65, 2022, pp. 1–3.

[59] S. Lee, S.-h. Kang, J. Lee, H. Kim, E. Lee, S. Seo, H. Yoon, S. Lee, K. Lim, H. Shin, J. Kim, O. Seongil, A. Iyer, D. Wang, K. Sohn, and N. S. Kim, "Hardware architecture and software stack for pim based on commercial dram technology : Industrial product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 43–56.

[60] C. Li, Z. Zhou, Y. Wang, F. Yang, T. Cao, M. Yang, Y. Liang, and G. Sun, "Pim-dl: Expanding the applicability of commodity dram-pims for deep learning via algorithm-system co-optimization," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 879–896. [Online]. Available: https://doi.org/10.1145/3620665.3640376

[61] C. Lim, S. Lee, J. Choi, J. Lee, S. Park, H. Kim, J. Lee, and Y. Kim, "Design and analysis of a processing-in-dimm join algorithm: A case study with upmem dimms," *Proc. ACM Manag. Data*, vol. 1, no. 2, Jun. 2023. [Online]. Available: https://doi.org/10.1145/3589258

[62] S. Lym, H. Ha, Y. Kwon, C.-k. Chang, J. Kim, and M. Erez, "Eruca: Efficient dram resource utilization and resource conflict avoidance for memory system parallelism," in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 670–682.

[63] D. Mudigere, Y. Hao, J. Huang, Z. Jia, A. Tulloch, S. Sridharan, X. Liu, M. Ozdal, J. Nie, J. Park, L. Luo, J. A. Yang, L. Gao, D. Ivchenko, A. Basant, Y. Hu, J. Yang, E. K. Ardestani, X. Wang, R. Komuravelli, C.-H. Chu, S. Yilmaz, H. Li, J. Qian, Z. Feng, Y. Ma, J. Yang, E. Wen, H. Li, L. Yang, C. Sun, W. Zhao, D. Melts, K. Dhulipala, K. Kishore, T. Graf, A. Eisenman, K. K. Matam, A. Gangidi, G. J. Chen, M. Krishnan, A. Nayak, K. Nair, B. Muthiah, M. khorashadi, P. Bhattacharya, P. Lapukhov, M. Naumov, A. Mathews, L. Qiao, M. Smelyanskiy, B. Jia, and V. Rao, "Software-hardware co-design for fast and scalable training of deep learning recommendation models," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 993–1011. [Online]. Available: https://doi.org/10.1145/3470496.3533727

[64] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "Processing data where it makes sense: Enabling in-memory computation," *Microprocessors and Microsystems*, vol. 67, pp. 28–41, 2019.

[65] M. Naumov, D. Mudigere, H.-J. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C.-J. Wu, A. G. Azzolini, D. Dzhulgakov, A. Mallevich, I. Cherniavskii, Y. Lu, R. Krishnamoorthi, A. Yu, V. Kondratenko, S. Pereira, X. Chen, W. Chen, V. Rao, B. Jia, L. Xiong, and M. Smelyanskiy, "Deep learning recommendation model for personalization and recommendation systems," 2019. [Online]. Available: https://arxiv.org/abs/1906.00091

[66] H. Nejatollahi, S. Gupta, M. Imani, T. S. Rosing, R. Cammarota, and N. Dutt, "Cryptopim: In-memory acceleration for lattice-based cryptographic hardware," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[67] S. U. Noh, J. Hong, C. Lim, S. Park, J. Kim, H. Kim, Y. Kim, and J. Lee, "Pid-comm: A fast and flexible collective communication framework

for commodity processing-in-dimm devices," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024, pp. 245–260.

[68] C. Ortega, "Next generation upmem pim dram for ai applications," 2024. [Online]. Available: https://www.upmem.com/wp-content/uploads/2024/09/240826-ABUMPIMP-2024-Keynote_-UPMEM-PIM-platform-for-Data-Intensive-Applications.pdf

[69] D. K. Panda, H. Subramoni, C.-H. Chu, and M. Bayatpour, "The mvapich project: Transforming research into high-performance mpi library for hpc community," *Journal of Computational Science*, vol. 52, p. 101208, 2021, case Studies in Translational Computer Science. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877750320305093

[70] J. Park, J. Choi, K. Kyung, M. J. Kim, Y. Kwon, N. S. Kim, and J. H. Ahn, "Attacc! unleashing the power of pim for batched transformer-based generative model inference," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 103–119. [Online]. Available: https://doi.org/10.1145/3620665.3640422

[71] J. Park, B. Kim, S. Yun, E. Lee, M. Rhu, and J. H. Ahn, "Trim: Enhancing processor-memory interfaces with scalable tensor reduction in memory," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 268–281. [Online]. Available: https://doi.org/10.1145/3466752.3480080

[72] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, J. Heek, K. Xiao, S. Agrawal, and J. Dean, "Efficiently scaling transformer inference," in *Proceedings of Machine Learning and Systems*, D. Song, M. Carbin, and T. Chen, Eds., vol. 5. Curan, 2023, pp. 606–624. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2023/file/c4be71ab8d24cdfb45e3d06dbfca2780-Paper-mlsys2023.pdf

[73] W. Qadeer, R. Hameed, O. Shacham, P. Venkatesan, C. Kozyrakis, and M. Horowitz, "Convolution engine: Balancing efficiency and flexibility in specialized computing," *Commun. ACM*, vol. 58, no. 4, p. 85–93, mar 2015. [Online]. Available: https://doi.org/10.1145/2735841

[74] S. Rashidi, W. Won, S. Srinivasan, S. Sridharan, and T. Krishna, "Themis: A network bandwidth-aware collective scheduling policy for distributed training of dl models," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ser. ISCA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 581–596. [Online]. Available: https://doi.org/10.1145/3470496.3527382

[75] D. Reis, J. Takeshita, T. Jung, M. Niemier, and X. S. Hu, "Computing-in-memory for performance and energy-efficient homomorphic encryption," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 11, pp. 2300–2313, 2020.

[76] S. H. S. Rezaei, M. Modarressi, R. Ausavarungnirun, M. Sadrosadati, O. Mutlu, and M. Daneshtalab, "Nom: Network-on-memory for inter-bank data transfer in highly-banked memories," *IEEE Comput. Archit. Lett.*, vol. 19, no. 1, p. 80–83, jan 2020. [Online]. Available: https://doi.org/10.1109/LCA.2020.2990599

[77] Y. Ro, H. Cho, E. Lee, D. Jung, Y. H. Son, J. H. Ahn, and J. W. Lee, "SOUP-N-SALAD: allocation-oblivious access latency reduction with asymmetric DRAM microarchitectures," in *2017 IEEE International Symposium on High Performance Computer Architecture, HPCA 2017, Austin, TX, USA, February 4-8, 2017*. IEEE Computer Society, 2017, pp. 517–528. [Online]. Available: https://doi.org/10.1109/HPCA.2017.31

[78] J. Romero, J. Yin, N. Laanait, B. Xie, M. T. Young, S. Treichler, V. Starchenko, A. Borisevich, A. Sergeev, and M. Matheson, "Accelerating collective communication in data parallel training across deep learning frameworks," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 1027–1040.

[81] M. Seo, X. T. Nguyen, S. J. Hwang, Y. Kwon, G. Kim, C. Park, I. Kim, J. Park, J. Kim, W. Shin, J. Won, H. Choi, K. Kim,

[79] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede, "Compact ring-lwe cryptoprocessor," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2014, pp. 371–391.

[80] N. Samardzic, A. Feldmann, A. Krastev, S. Devadas, R. Dreslinski, C. Peikert, and D. Sanchez, "F1: A Fast and Programmable Accelerator for Fully Homomorphic Encryption," in *MICRO-54: 54th Annu. IEEE/ACM Int. Symp. on Microarchitecture*, ser. MICRO '21. New York, NY, USA: ACM, 2021, pp. 238–252.
D. Kwon, C. Jeong, S. Lee, Y. Choi, W. Byun, S. Baek, H.-J. Lee, and J. Kim, "Ianus: Integrated accelerator based on npu-pim unified memory system," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ser. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 545–560. [Online]. Available: https://doi.org/10.1145/3620666.3651324

[82] S. Shim, S. Kim, J. Bae, K. Ko, E. Lee, K. Kim, K. Kim, S. Lee, J. Hyun, I. Koh, J. Park, M. Kim, S. Shin, D. Lee, Y. Lee, S. Hyun, W. Choi, D. Im, D. Lee, J. Jang, S. Lee, J. Chun, J. Oh, J. Kim, and S.-H. lee, "A 16gb 1.2v 3.2gb/s/pin ddr4 sdram with improved power distribution and repair strategy," in *2018 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2018, pp. 212–214.

[83] W. Sun, Z. Li, S. Yin, S. Wei, and L. Liu, "Abc-dimm: Alleviating the bottleneck of communication in dimm-based near-memory processing with inter-dimm broadcast," in *Proceedings of the 48th Annual International Symposium on Computer Architecture*, ser. ISCA '21. IEEE Press, 2021, p. 237–250. [Online]. Available: https://doi.org/10.1109/ISCA52012.2021.00027

[84] R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in mpich," *Int. J. High Perform. Comput. Appl.*, vol. 19, no. 1, p. 49–66, feb 2005. [Online]. Available: https://doi.org/10.1177/1094342005051521

[85] B. Tian, Y. Li, L. Jiang, S. Cai, and M. Gao, "NDPBridge: Enabling Cross-Bank Coordination in Near-DRAM-Bank Processing Architectures," in *51st International Symposium on Computer Architecture (ISCA)*, Jun 2024, pp. 628–643.

[86] Y. Turakhia, G. Bejerano, and W. J. Dally, "Darwin: A genomics co-processor provides up to 15,000x acceleration on long read assembly," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 199–213. [Online]. Available: https://doi.org/10.1145/3173162.3173193

[87] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, p. 65–76, Apr. 2009. [Online]. Available: https://doi.org/10.1145/1498765.1498785

[88] N. Zhang, B. Yang, C. Chen, S. Yin, S. Wei, and L. Liu, "Highly efficient architecture of newhope-nist on fpga using low-complexity ntt/intt," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 49–72, 2020.

[89] Z. Zhe, L. Cong, Y. Fan, and S. Guangyu, "Dimm-link: Enabling efficient inter-dimm communication for near-memory processing," in *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, ser. HPCA '23. IEEE Computer Society, 2023.

[90] M. Zhou, Y. Nam, P. Gangwar, W. Xu, A. Dutta, K. Subramanyam, C. Wilkerson, R. Cammarota, S. Gupta, and T. Rosing, "Fhemem: A processing in-memory accelerator for fully homomorphic encryption," 2023. [Online]. Available: https://arxiv.org/abs/2311.16293

[91] M. Zhou, W. Xu, J. Kang, and T. Rosing, "Transpim: A memory-based acceleration via software-hardware co-design for transformer," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 1071–1085.