Processing-in-Memory

to conduct the simple computation in memory for energy-efficient computing.

Literature review by
Rahul Verma
PSID: 2251462, rverma6@uh.edu
Department of Electrical and Computer Engineering,
The University of Houston.

Abstract

The study focuses on several methods to processing-in-memory (PIM), a technique that claims to save energy while enhancing speed by doing computations in memory rather than transmitting data to the CPU. The NDA architecture utilizes 3D-stacked accelerators on DRAM devices to process data fast within memory, hence decreasing the requirement for data transmission between the CPU and memory. PIM-enabled instructions (PEIs) enable PIM operations to be executed in the same manner as traditional host processor instructions, and a PIM execution model that incorporates PEIs into the existing programming architecture is described. PRIME is a PIM architecture built on ReRAM for efficient neural network processing, and the Memory Channel Network (MCN) connects processors and memory to increase bandwidth.

Keywords

Processing in memory (PIM), Near data access (NDA), DRAM, ReRAM, Memory, Energy, Pipelayer, Neural network.

Introduction

Processing-in-Memory (PIM) is a modern approach that promises to progress in preparing productivity by conducting calculations straightforwardly in memory. Moving information from memory to the CPU for handling requires a noteworthy use of energy and time in conventional computer systems. Joining handling units specifically into memory structures with PIM technology avoids information mobility and permits computations to require put inside the memory itself. PIM innovation, which diminishes the sum of information sent between memory and CPU, can result in significant energy savings.

PIM innovation is classified into three sorts: logic-based memory, analog processing, and digital processing. Processing units inside the memory module utilize circuitry particularly built for different computations in logic-based memory. Analog processing units utilize the analog properties of memory cells to calculate, while digital processing units utilize advanced circuits to compute straightforward memory information.

PIM innovation is especially solid at crucial computations like matrix multiplication and convolution. With PIM-enabled systems, these operations may be done successfully in memory, coming about in impressive energy savings and improved execution. PIM innovation has the potential to revolutionize the computer industry and bring in a modern era of energy-efficient computing devices as it moves forward.

One of the foremost critical benefits of PIM technology is its potential to extend data-intensive movement execution. PIM systems can boost handling speed, decrease latency, and reduce energy utilization by lessening the requirement for information transmission, coming about in faster and more productive information processing. PIM innovation has appeared guaranteed in areas like manufactured insights, profound learning, and logical computing, where huge data of information must be processed quickly and legitimately.

In spite of its potential benefits, PIM innovation is still in its early stages, with different roadblocks to overcome before it can accomplish its full potential.

Challenges

A few challenges incorporate interaction between the data movement like CPU, and DRAM with the accelerator, issues with standard DRAM interface and DIMM design compatibility, dynamic profiling of the data locality of PEIs, memory bandwidth limitations, issues in handling graph systems in fundamental memory computation, memory wall issue, rising demand in deep learning, the physical size of chips, improvement of new programming models for in-memory processing, Equipment complexity.

Major Opportunities

A few of the openings of this innovation are the data increasing speed with 3D-stacked DRAM processors, proposed PRIME architecture, PRIME can too speed neural network applications by utilizing ReRAM-based primary memory, Pipelayer's innovation seem to revolutionize the field of deep learning, significant alter to DRAM dies due to the nearness of logic dies, Progressed execution and Scalability utilizing Memory Channel Network (MCN) engineering, Moved forward

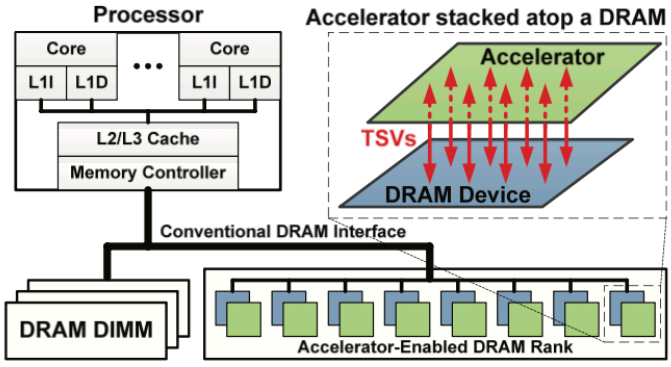
system execution, and Energy productivity utilizing Active-Routing.

In any case, as PIM innovation progresses, it might change the computing field and permit a new breed of energy-efficient computing systems.

There are different research directions pointing to decreasing energy utilization and enhancing execution through data processing.

1. Near-Data-Access.

The proposed NDA (Near-Data-Access) design is extraordinary to minimize energy utilization while improving processing execution by utilizing 3D-stacked accelerators on DRAM chips. Placing the processing unit close to memory permits data to be processed instantaneously within memory, minimizing the need to transport information back and forward between the processor and memory. The NDA design can save energy while improving handling execution by utilizing 3D-stacked accelerators [1] on DRAM chips. It is additionally economical and basic to execute since it makes utilize of standard DRAM modules and memory components.

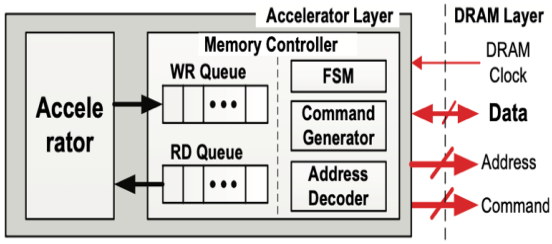


NDA Architecture

The NDA design is well-suited for data-intensive processes such as machine learning, which require the processing of huge volumes of data quickly and successfully. The NDA design can assist improve the speed of machine learning applications while decreasing energy utilization and consumption by eliminating the need to transport data between the CPU and memory [1]. In general, the NDA design offers a particular approach to bringing down energy utilization and improving processing speed, with the potential to have a major impact on a wide extend of applications and systems.

1.1. Near-memory processing using a memory controller (MC).

Near-memory processing that can be implemented is memory controller (MC) processing and incorporating processing units directly into memory devices such as accelerator CGRAs or SIMD/GPU/FPGA engines [1]. Dedicated hardware is added to the memory controller in the case of MC processing to allow data to be computed before being sent to the CPU or cache. Reducing the number of times data needs to be moved between different components, this can help reduce the amount of data transfer through the cache structure as well as the system's overall energy consumption [1].



Accelerator stacked on top of a DRAM device.

1.2. Near-data processing units directly into memory devices.

An alternative to near-memory processing is the integration of processors into memory components, which is additionally known as processing in memory (PIM) [1]. This approach includes joining processor logic and DRAM onto a single chip to empower processing in memory. PIM systems exploit lower access latency and higher total bandwidth to empower high-performance local data processing. There have been various endeavors to integrate processor logic and DRAM onto a single chip to empower PIM, such as the Computational RAM architecture, Cleverly RAM, and FlexRAM chip. In any case, these systems endure large manufacturing costs and low yields since of joining two distinctive technologies. Besides, they upgrade the DRAM design to efficiently integrate handling units within the memory devices, making their plan and verification challenging and time-consuming. In spite of these challenges, PIM has appeared promising comes about in reducing data exchange and maximizing system execution by cutting down on latency issues. Be that as it may, it requires careful consideration amid hardware design and computer program programming to attain proficient communication between the CPU, DRAM, and accelerators and create viable programming models for coordinates CPU-DRAM-accelerator systems.

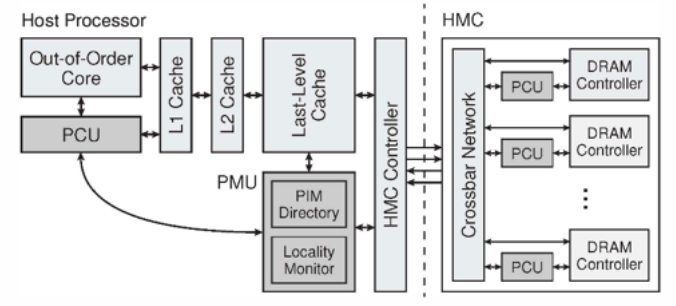
Both of these approaches have the potential to greatly augment computing performance and energy efficiency, in particular concerning data-heavy operations such as machine learning and data analysis. Nonetheless, they necessitate considerable alterations to both hardware and software designs, and a lot of research is still needed to unlock the full potential of near-memory computing.

This solution aims to reduce the amount of data that needs to be sent between components of the cache hierarchy by incorporating specialized hardware [1] into the memory controller. By enabling calculations to be done at the memory controller level before sending data to the CPU or cache, this approach could decrease the need for data

to be transferred between multiple components, a key issue in PIM/NMP systems.

2. PIM-Enabled Instructions.

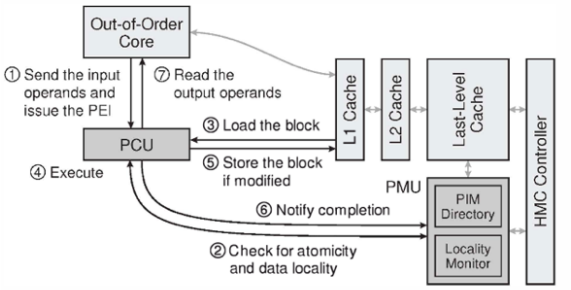
PEIs, or PIM-enabled instructions, are a type of instruction that enables PIM operations to be executed in the same way as conventional host processor instructions are [2]. These instructions are designed to operate on either the host processor or the in-memory computing logic, allowing current systems to gain PIM capabilities without changing their programming interface. PEIs enable older systems to reap the benefits of PIM without requiring large changes by embedding PIM aspects into the current programming architecture [2].



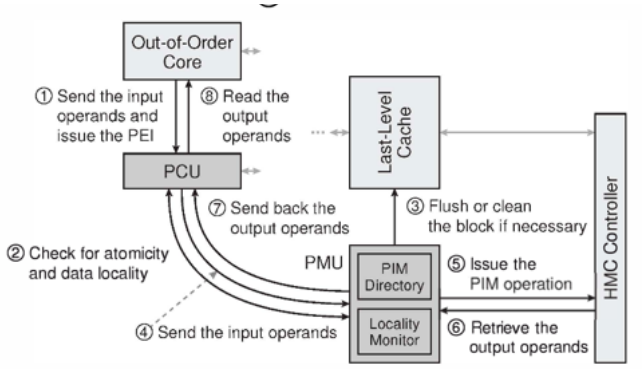
Proposed architecture

2.1. Processing-in-memory (PIM) execution model.

A PIM execution demonstrates that's in line with current virtual memory and cache coherence technologies, as well as modern programming models. This will be fulfilled through the creation of specialized instructions (PEIs) that can be executed on the have processor or the in-memory computation logic, the usage of straightforward in-memory computation utilizing compute-capable memory commands, and the extension of the have processor's ISA with PEIs to empower straightforward PIM operations whereas minimizing overheads. Besides, PIM operations can be performed in memory or on processors based on the area of the data employing a mechanism known as locality-aware PEI execution [2]. There are different advantages to the proposed approach. PIM operations may presently be completely consistent with present-day programming standards, cache coherence conventions, and virtual memory strategies. It diminishes the estimate, control, and warm overheads by implanting the handling unit’s interior memory whereas too disentangling equipment back for interoperability with existing cache coherence and virtual memory methods. At last, it accomplishes the benefits of both memory-side and host-side execution by conducting PIM operations in memory or on processors according to data arrangement [2].



Host-side PEI execution.



Memory-side PEI execution.

To create a processing-in-memory execution demonstration that's consistent with cutting-edge programming models and existing mechanisms for cache coherence and virtual memory, the straightforward in-memory computation must be executed utilizing compute-capable memory commands, specialized instructions (PEIs) must be characterized, the processor's ISA must be extended with PEIs to empower straightforward PIM operations whereas minimizing overheads, and a component known as locality-aware PEI execution must be introduced.

2.2. Locality-aware PEI execution.

Based on data locality, the concept of locality-aware PEI execution points to adaptively decide the perfect place for executing PEIs, either in memory or on the host processor [2]. To do this, the system is designed to screen the application's information get to designs and decide whether to run a PEI in memory or on the processor dependent on the area of the data being gotten to. Based on data locality, the concept of locality-aware PEI execution points to adaptively decide the perfect place for executing PEIs, either in memory or on the host processor. To do this, the system is designed to screen the application's information get to designs and decide whether to run a PEI in memory or on the processor dependent on the area of the data being gotten to.

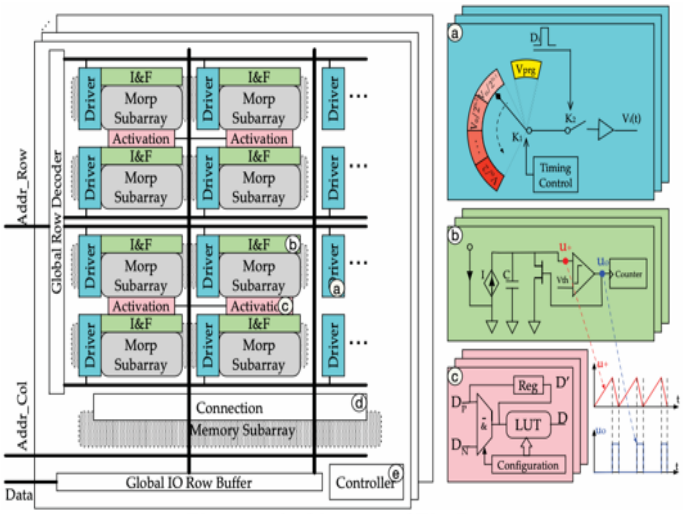
Memory is separated into different segments that compare to cache squares in this approach. The framework at that point keeps up track of the zones that the program gets to and uses this information to discover the leading execution location for a PEI. If the data being processed by a PEI is as of now in memory, it'll be executed there. In any case, on the off chance that the data isn't accessible in memory, the PEI is executed on the processor.

To summarize, locality-aware PEI execution [2] incorporates dividing memory into zones that compare to cache blocks, following the regions utilized by an application, and powerfully selecting whether to execute a PEI in memory or on the processor based on data locality. This approach gives fine-grained control over the computation area and optimizes both memory-side and host-side execution. Data exchange between memory and processors may be minimized by doing calculations in memory at whatever point conceivable, but running computations on processors, when necessary, helps to maintain execution within the face of cache misses or other conditions.

The concept of locality-aware PEI execution handles the center challenge of dynamically observing the locality of PEI data. By checking the application's data getting to designs and deciding whether to run a PEI in memory or on the processor based on data area, this strategy decreases overheads and progresses speed.

3. Pipelayer- ReRAM-based processing-in-memory (PIM) accelerator.

ReRAM is a customized processing-in-memory (PIM) accelerator created by PipeLayer for convolutional neural networks (CNNs) with weight upgrades [3]. Both the testing stage and the training stage involved this acceleration. PipeLayer utilizes the compute and capacity capabilities of ReRAM to execute matrix multiplication and weight updates in memory. The design of the accelerator uses a clear intra and inter-layer pipeline to extend throughput for CNNs with weight updates while minimizing the sum of information that must be moved all through the memory hierarchy and improving energy efficiency. In tests, PipeLayer consumed up to 5 times less energy per operation and performed up to 3.8 times way better than the foremost progressed GPU solutions. As illustrated, PipeLayer's exceptional performance and energy economy make it a profitable device for boosting deep learning applications.



Architecture of PipeLayer

3.1. Optimization of PipeLayer for different types of neural networks.

PipeLayer optimization for different sorts of neural networks requires testing with different pipelining calculations, memory access designs, and weight upgrading strategies that are suited to each neural network's topology [3]. Certain neural networks, for example, may have more inter-layer parallelism than intra-layer parallelism, though others may have the inverse. PipeLayer tuning for these different sorts of neural systems may hence require adjusting the pipeline depth and width to extend parallelism and decrease latency. Moreover, different memory access designs may be required in certain neural networks to proficiently handle information conditions between layers. At last, weight update plans may need to be custom-made to particular neural network designs in order to decrease overhead whereas progressing execution. In general, optimizing PipeLayer for different sorts of neural networks may result in moved-forward execution and energy effectiveness for a broader extent of deep learning applications.

3.2. Integrate PipeLayer with other hardware accelerators.

One conceivable approach to joining PipeLayer with other equipment accelerators is to utilize a heterogeneous computing architecture that combines different sorts of processors, such as CPUs, GPUs, and FPGAs [3]. PipeLayer can serve as a PIM accelerator for deep learning applications, whereas other equipment accelerators can be utilized for other assignments requiring high execution or energy efficiency.

Another choice is to utilize a secluded design approach, empowering diverse components of the computing system to be effectively replaced or upgraded. In this approach, PipeLayer can be created as a standalone module that can be coordinated into different computing systems based on their particular necessities.

In common, the integration of PipeLayer with other equipment accelerators requests cautious thought of system necessities and design limitations [3]. It may require the creation of modern programming models and runtime systems that can productively use the distinctive sorts of processors within the system whereas moreover addressing concerns with respect to workload diversity and scalability.

4. PRIME: ReRAM-based processing in-memory solution to accelerate neural network (NN) applications.

A PIM based on ReRAM crossbar arrays is called PRIME. It could be a shorthand for Processing-in-memory architecture for effective neural network processing. The proposed design utilizes ReRAM

crossbar arrays to perform matrix-vector multiplication, which is essential in neural network applications [4]. PRIME is composed of memory arrays that, depending on the circumstance, may be sent as either NN accelerators or customary memory. This includes permitting data mobility inside memory and moving forward the speed of ReRAM-based handling. The design comprises a collection of circuits and microarchitecture that empower NN computation in memory while decreasing space overhead. This is often fulfilled by cautious circuit plans, such as the utilization of peripheral circuits for memory and computing. To summarize, the recommended PRIME design gives a novel and imaginative arrangement to the "memory wall" issues of future computer systems [4]. It moves forward by and large system proficiency whereas upgrading neural network applications by effectively utilizing ReRAM-based primary memory.

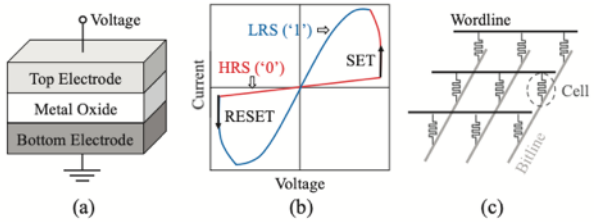


Figure 1. (a) Conceptual view of a ReRAM cell; (b) I-V curve of bipolar switching; (c) schematic view of a crossbar architecture.

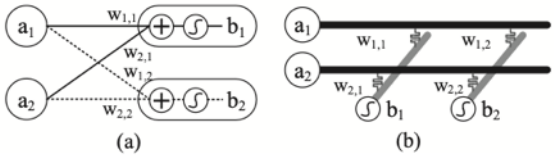
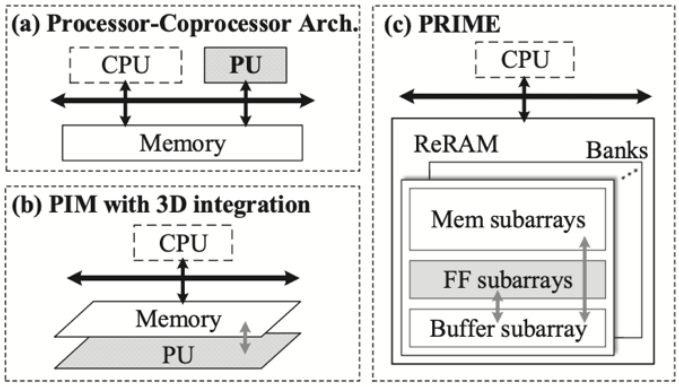


Figure 2. (a) An ANN with one input/output layer; (b) using a ReRAM crossbar array for neural computation.



(a) Traditional shared memory-based processor-coprocessor architecture,
(b) PIM approach using 3D integration technologies,
(c) PRIME design.

4.1. Designing a set of circuits and microarchitecture to enable the neural network computation in memory.

This proposed design incorporates peripheral circuits, a memory gets-to controller, a computational engine, and a data exchange engine, among other equipment components. The peripheral circuits are aiming to be flexible, and able of performing memory and calculation operations while involving less overall space. This design strategy makes great utilization of ReRAM-based primary memory for NN calculation [4]. The memory access controller guarantees that data streams easily between system components. To conduct matrix-vector multiplication successfully, the computation engine uses ReRAM crossbar arrays. It is expected to be adaptable since it may serve as an accelerator for NN applications or as ordinary memory for a more extensive working memory zone. The data movement engine, which permits in-memory information transmission, is basic for high execution and energy effectiveness.

These equipment components collaborate to empower productive NN computation inside memory whereas utilizing the slightest sum of space [4]. By making productive utilization of ReRAM-based primary memory, the recommended circuit and microarchitecture solutions eliminate the "memory wall" issue for future computer systems.

This addresses the "memory wall" issue in computer systems by permitting effective neural network (NN) computation on ReRAM-based primary memory. It does this through the utilization of adaptable peripheral circuits, a memory access controller, and a compute engine that utilizes ReRAM crossbar arrays for matrix-vector multiplication.

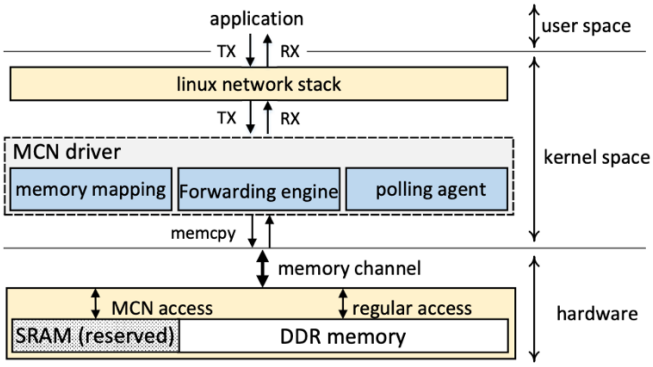
4.2. Utilizing the same peripheral circuitry for processing and memory.

A key design concept for enabling effective neural network computation in memory is the reuse of peripheral circuits for both memory and computation exercises. In neuromorphic computer systems and ReRAM-based memory systems, the peripheral circuits are regularly required for read and write operations. The recommended architecture, in any case, utilizes the same peripheral circuits for both jobs, for instance, utilizing ADCs for memory and calculation duties, individually. The complete overhead space is decreased as a result. This strategy makes it simpler to utilize ReRAM-based primary memory for NN calculation [4]. Through cautious planning, the recommended system accomplishes low area overhead, empowering successful NN computation in memory with a negligible range overhead. In general, the plan procedure offers an innovative reaction to the troubles of successfully utilizing ReRAM-based primary memory to speed up neural network applications.

This architecture overcomes the issue of space overhead in ReRAM-based neuromorphic computing gadgets by reusing peripheral circuits for both memory and computation exercises [4]. This minimizes the general area required and empowers effective neural network computation in memory, improving performance and energy efficiency.

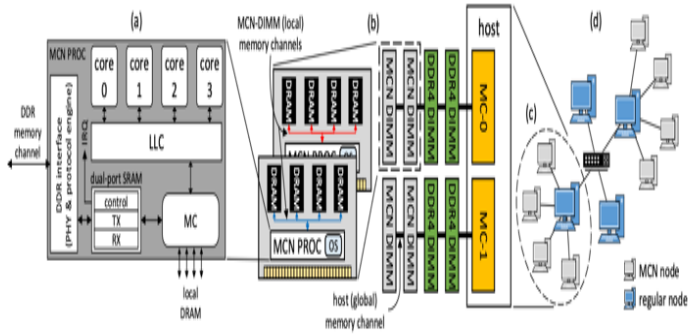
5. Near memory processing using Memory Channel Network (MCN).

The MCN design (Memory Channel Network) may be a one-of-a-kind innovation that advances handling in memory (PIM) by firmly integrating a processor with memory and uncovering the processor to extra bandwidth [5]. The architecture is made up of MCN DIMMs (dual in-line memory modules) and MCN drivers, which permit programs based on existing distributed computing systems to operate without requiring any changes to the processor equipment, distributed computing middleware, or application software.



MCN Driver

The MCN idea utilizes a memory channel organized to put through a few processors and memories, permitting efficient communication. Each MCN CPU has its local memory channel, which it utilizes to get to DRAM devices on the same MCN DIMM, as restricted to the worldwide memory channel shared by other DIMMs. This design permits high-bandwidth communication between processors and memories while lessening data exchange necessities.



MNC Architecture

It proposes software and hardware optimization techniques for improving transmission capacity utilization and decreasing communication time between MCN DIMMs [5]. These optimization options incorporate updating the MCN driver and other OS organize layers that make utilization of MCN's special features over standard Ethernet. Besides, by utilizing an extraordinary signal used by recent and future memory interfacing to interrupt the MC when an MCN DIMM has active packets, changing the host-side MC can progress communication effectiveness.

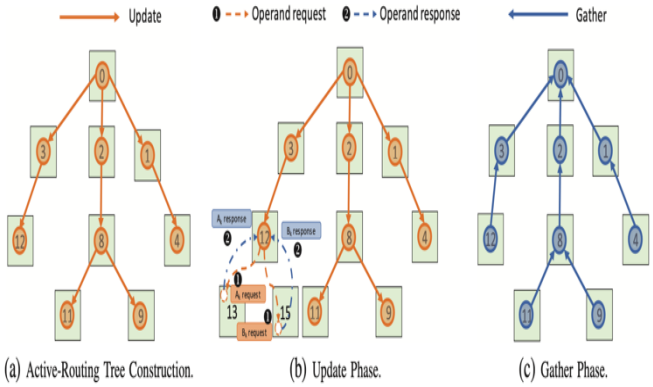
In general, the MCN design may be a progressive strategy for memory-based computing since it firmly integrates a processor with memory whereas lessening data transit between them. The utilization of an arrangement of memory channels empowers more successful communication between processors and memories, whereas optional optimization strategies can encourage increment bandwidth effectiveness and minimize communication delay.

The MCN design overcomes the communication proficiency trouble within the processor in memory (PIM) frameworks. MCN encourages successful communication between processors and recollections whereas diminishing information exchange by interfacing them by means of a arrange of memory channels. This plan empowers high-bandwidth communication between processors and memory, which is fundamental for PIM frameworks like MCN. Furthermore, the paper proposes discretionary program and equipment optimization procedures to extend transfer speed utilization and diminish communication idleness, tending to a major challenge of PIM models.

6. Active-Routing Architecture.

The Active-Routing PIM architecture has a three-phase processing schedule [6]. The first stage classifies compute kernel memory access patterns, and calculations are offloaded in various granularities by using their proximity attributes to reduce offloading overhead. In the second step, the computation is offloaded to the memory network for execution close to the data. In the final step, the findings are consolidated along their routing path.

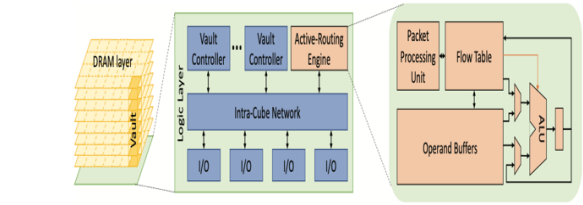
PIM architectures such as Active-Routing can help address the challenge of high CPU compute density vs. inadequate memory bandwidth by minimizing data transmission requirements and improving system performance [6]. They do concede, however, that applying these techniques in current memory subsystems and communication fabrics may have downsides or limits that require further investigation.



The Active-Routing microarchitecture is proposed in the study as a solution for efficient processing in memory. It is implemented in the Hybrid Memory Cube (HMC) logic layer as a connected module to the router switch and

consists of various components such as a packet processing unit, a flow table, a pool of operand buffers, and an arithmetic logic unit (ALU).

The flow table is a hash table where Active-Routing flows are stored. Each item contains information on the flow's source and destination addresses, as well as its current state. An example flow table entry is shown in the image. The Active-Routing microarchitecture was designed to allow high-performance data processing activities by offloading computation to memory network data-resident locations. It aims to successfully manage processing-in-memory difficulties with its packet processing capabilities, flow tracking features, operand storage, and calculation capabilities [6].



(a) HMC Logic Layer for Active-Routing Engine.

64-bit	6-bit	64-bit	64-bit	64-bit	2-bit	4-bit	1-bit
flowID	opcode	result	req_counter	resp_counter	parent	children flags	Gflag

(b) Flow Table Entry.

64-bit	6-bit	64-bit	1-bit	64-bit	1-bit
flowID	opcode	operand1	op1_ready	operand2	op2_ready

(c) Operand Buffer Entry.

The Active-Routing PIM design overcomes the problem of high CPU compute density with insufficient memory bandwidth by reducing data transmission needs and boosting system performance. This is accomplished by a three-phase processing schedule that offloads computing to data-resident sites in the memory network, decreasing data travel across the memory hierarchy.

Future scope

Technological improvements in Processing-in-Memory (PIM) have opened new prospects for improving computer speed and energy efficiency. More research and development will be needed in the future to overcome the challenges of implementing PIM technology, such as hardware complexity and memory bandwidth limitations. However, several promising approaches are already available, including the Near-Data-Access (NDA) design, PRIME architecture, Memory Channel Network (MCN), and Active-Routing architecture. These strategies have the ability to boost data processing while decreasing energy consumption and enhancing system efficiency. Furthermore, the creation of PIM-enabled instructions (PEIs) and execution models may aid in the seamless integration of PIM capabilities into existing systems, making them more accessible and widely used.

Conclusion

In summary, Processing-in-Memory (PIM) technology has the potential to revolutionize the computer industry by improving processing speed, latency, and energy consumption. PIM capabilities can be achieved through different technologies such as logic-based memory, analog processing, and digital processing. The Near-Data-Access (NDA) architecture is an interesting concept that uses 3D-stacked accelerators on DRAM chips to process data in memory instantly. PIM-enabled instructions (PEIs) and a PIM execution model have been proposed to integrate PIM operations into the current programming architecture.

PipeLayer is a ReRAM-based PIM accelerator that offers exceptional performance and energy efficiency for deep learning applications. Optimizing PipeLayer for different neural network topologies and integrating it with other hardware accelerators can improve performance and energy efficiency. PRIME and Memory Channel Network (MCN) are other PIM topologies that can improve neural network computation while improving bandwidth, and the Active-Routing design addresses the issue of high CPU compute density and limited memory bandwidth.

While PIM technology provides numerous opportunities for improving data processing, there are challenges such as memory bandwidth constraints, DRAM interface issues, and hardware complexity. However, ongoing research and development in this field are expected to bring further advances and improvements in computing systems. In conclusion, the potential benefits of PIM technology make it an exciting area of research, and its integration with other technologies will likely shape the future of computing systems.

Acknowledgment

I would like to express my sincere gratitude to Professor Dr. Xin Fu and teaching assistant Mr. Chen Zhang for their valuable guidance and support throughout this project. Their expertise in the field of advanced computer architecture has been instrumental in shaping the direction and outcomes of this literature review. Thank you for your unwavering commitment and dedication to our academic pursuits.

References

1. Farmahini-Farahani, Amin, et al. "NDA: Near-Dram Acceleration Architecture Leveraging Commodity DRAM Devices and Standard Memory Modules." 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), 2015, https://doi.org/10.1109/hpca.2015.7056040.

2. Ahn, Junwhan, et al. "Pim-Enabled Instructions." *Proceedings of the 42nd Annual International Symposium on Computer Architecture, 2015, https://doi.org/10.1145/2749469.2750385.*

3. Song, L., Qian, X., Li, H., & Chen, Y. (2017). Pipelayer: A pipelined reram-based accelerator for deep

learning. *2017 IEEE International Symposium on HighPerformance Computer Architecture (HPCA)*. https://doi.org/10.1109/hpca.2017.55

4. Chi, Ping, et al. "Prime: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory." 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), 2016, https://doi.org/10.1109/isca.2016.13.

5. Alian, Mohammad, et al. "Application-Transparent near-Memory Processing Architecture with Memory Channel Network." 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2018, https://doi.org/10.1109/micro.2018.00070.

6. Huang, Jiayi, et al. "Active-Routing: Compute on the Way for near-Data Processing." 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2019, https://doi.org/10.1109/hpca.2019.00018.