Processing in memory: the tipping point

Petar Radojković, Paul Carpenter, Pouya Esmaili-Dokht, Rémy Cimadomo, Henri-Pierre Charles, Sebastian Abu, Paolo Amato

Processing in Memory: the Tipping Point

Why processing in memory is happening now and how to make it widely adopted in HPC

White Paper

Petar Radojković (BSC), Paul Carpenter (BSC),
Pouya Esmaili-Dokht (BSC), Rémy Cimadomo (UPMEM), Henri-Pierre Charles (CEA),
Abu Sebastian (IBM), Paolo Amato (Micron)

29/07/2021

etp4hpc.eu

@etp4hpc

EUROPEAN TECHNOLOGY PLATFORM FOR HIGH PERFORMANCE COMPUTING

Introduction

Decades after being initially explored in the 1970s, Processing in Memory (PIM) is currently experiencing a renaissance. By moving part of the computation to the memory devices, PIM addresses a fundamental issue in the design of modern computing systems, the mismatch between the von Neumann architecture and the requirements of important data-centric applications. A number of industrial prototypes and products are under development or already available in the marketplace, and these devices show the potential for cost-effective and energy-efficient acceleration of HPC, AI and data analytics workloads. This paper reviews the reasons for the renewed interest in PIM and surveys industrial prototypes and products, discussing their technological readiness.

Wide adoption of PIM in production, however, depends on our ability to create an ecosystem to drive and coordinate innovations and co-design across the whole stack. European companies and research centres should be involved in all aspects, from technology, hardware, system software and programming environment, to updating of the algorithm and application. In this paper, we identify the main challenges that must be addressed and we provide guidelines to prioritise the research efforts and funding. We aim to help make PIM a reality in production HPC, AI and data analytics.

Key insights

The tipping point for adoption of PIM is imminent for three main reasons:

- Firstly, PIM avoids the **von Neumann bottleneck**, a fundamental limitation to the effective use of computing systems for a large range of important **data-centric applications**.
- Secondly, it matches the community’s requirement for **efficient application acceleration** by reducing the amount of expensive data transfers, and demonstrates orders of magnitude improvements in **performance** and **energy**.
- Thirdly, PIM has reached a **high technological readiness** confirmed by various **industrial prototypes and products**.

Wide acceptance of PIM, however, depends on our ability to create an **ecosystem** in which a number of **PIM approaches** can be **designed** and **evaluated**, leading to the selection of **potential winners** and **adoption by system architects and end users**.

Key recommendations

To enable adoption of PIM technologies in production systems and applications, the community has to:

- **Build PIM software development vehicles, prototypes and commercial products** that will be used by early adopters to develop and evaluate system software and applications.
- **Manage PIM hardware** with advanced firmware, OS and system software.
- **Programme** target applications and kernels with enhanced **Application Programming Interfaces (APIs) and programming models** supported with PIM-aware **emulators, compilers, runtimes and libraries**.
- **Provide** users with **profiling and analysis tools** capable of detecting application phases and kernels that are suitable for offloading to PIM.
- **Evaluate** PIM proposals with recognised **benchmarks, simulators and HW platforms**. The evaluation metrics should include **performance, power, energy and total cost of ownership**, but also **system usability, resilience, security and economic viability**, including an analysis of the market size and production cost.

1. From von Neumann Bottleneck to Processing in Memory

From early designs, computer systems have maintained a strict separation between CPU and memory, forming the *von Neumann bottleneck*. This bottleneck, present from the very smallest scale embedded systems to the highest performance supercomputers, is especially detrimental to data-centric applications that poorly match the assumptions underlying a modern memory hierarchy, i.e. that have **large data volumes**, **low operational intensity, low spatial and**

temporal locality, and **unpredictable data access patterns**. Many of these characteristics arise in algorithms such as **sorting, filtering, hashing**, **associative memories**, **pointer-chasing**, **array and matrix address management**, **graph operations** and **time series analysis.** These algorithms are widely used in important application domains, especially **data analytics** (**big data** and **database** workloads), **machine learning**, **artificial intelligence**, **deep learning**, **bioinformatics** (e.g. **genome analysis**), and **scientific computing**.

The community has been trying to address the von Neumann bottleneck for decades. For example, high memory latency can often be mitigated using latency-hiding techniques, such as out-of-order execution, branch prediction, data prefetching, up to three levels of cache memory, data buffering, and more. These techniques have been successful in improving performance to a degree, but the downside has been increased complexity, high development and per-unit costs, and high power consumption. And yet, memory latency is still an important limitation to computer system performance [1]. This is because the problem to be solved is a fundamental one—the mismatch between the von Neumann architecture and the requirements of important data-centric applications. At this point, additional incremental opportunities to improve performance are close to be exhausted and it is time to look afresh at a different approach to the problem, the tight integration of data and processing. This approach is referred to as **Processing in Memory (PIM)**.

PIM is not a new idea. It was first explored in the 1970s [2, 3] and is currently experiencing a renaissance due to a combination of the community requirement for cost- and energy-efficient application acceleration and technological readiness. The interest in PIM has grown dramatically over the last years. In 2020, the EuroLab4HPC Vision [4] already included a survey of the approaches for near- and in-memory processing and it recommended funding in this direction. In the same year, ETP4HPC's Strategic Research Agenda [5] called near and in-memory architectures the "ultimate option" to improve energy efficiency, but correctly saw the approach as not mature enough for short-term adoption in production systems. Now is, however, the time to make the necessary research investments, across the whole stack, from hardware to programming environment to applications, in order to drive the adoption of PIM technologies.

For more perspective on the historical evolution of PIM concepts from the 1970s till today, don't miss *„Data-Centric Computing Frontiers: A Survey On Processing-In-Memory"* by Siegl et al. [3].

2. The Hidden Power of Memory

The high latency and low bandwidth of main memory is often thought to be a consequence of the memory technology, e.g. the timing parameters associated to the memory cells or cell-arrays. But this is wrong. In fact, most of the main memory access latency does not come from the time required to access the memory cells, but from the overall complexity of the memory system in both hardware (e.g. CPU cache hierarchy) and software (e.g. virtual memory management). Main memory bandwidth is also not limited by the internal architecture of the memory device and its timings, but by the narrow external CPU-to-memory interface.

We illustrate the problem using an example DRAM DIMM architecture. Figure 1 shows one side of the DIMM, holding one rank. Each rank contains multiple DRAM chips that work in sync, and each chip comprises multiple banks that can be accessed independently. Banks hold memory cells organised into two-dimensional arrays of rows and columns managed by the peripheral circuitry.

Internal memory latency in current DDR3 and DDR4 devices could be as low as 15 ns [6, 7]. This is the time required to read data from the memory array to the sense amplifiers, defined by the JEDEC standard as the Activate to internal Read delay (tRCD) timing parameter. The **external memory access latency**, the main memory latency perceived by the user, is the time required for a load instruction to access data in main memory. This timing is platform-specific as it includes the time spent in the CPU load/store queues, cache hierarchy, network-on-chip and on-chip memory controller. In current architectures the minimum external memory access latency, corresponding to the simplest case of a single memory load in an idle system, starts at 90 ns [8], exceeding the internal device latency by **6x**.

The difference between internal and external DIMM performance is even more pronounced if we consider the **memory bandwidth**. DIMMs are connected to the CPU over narrow memory channels that can transfer only **64 bits** (8 bytes) of data on each edge of the clock, i.e. each half cycle. Bandwidth of conventional DDR channels is in the order of **tens of gigabytes per second**, e.g. **21 GB/s** for DDR4-2666 channels. Internally, however, each per-rank DRAM row operation is performed on 8 KB of data,[1] and multiple ranks and banks access their memory arrays independently. This means that a conventional 2-rank 8-bank DIMM can simultaneously handle **128 KB** of data. The memory array row cycle has a total latency of tens of cycles, e.g. 62 cycles for DDR4-2666 [6]. This means that the internal DRAM memory bandwidth of 2-rank DDR4-2666 DIMM is **3884 GB/s**.[2] Overall, the cumulative internal memory bandwidth exceeds the external bandwidth by more than **two orders of magnitude**.

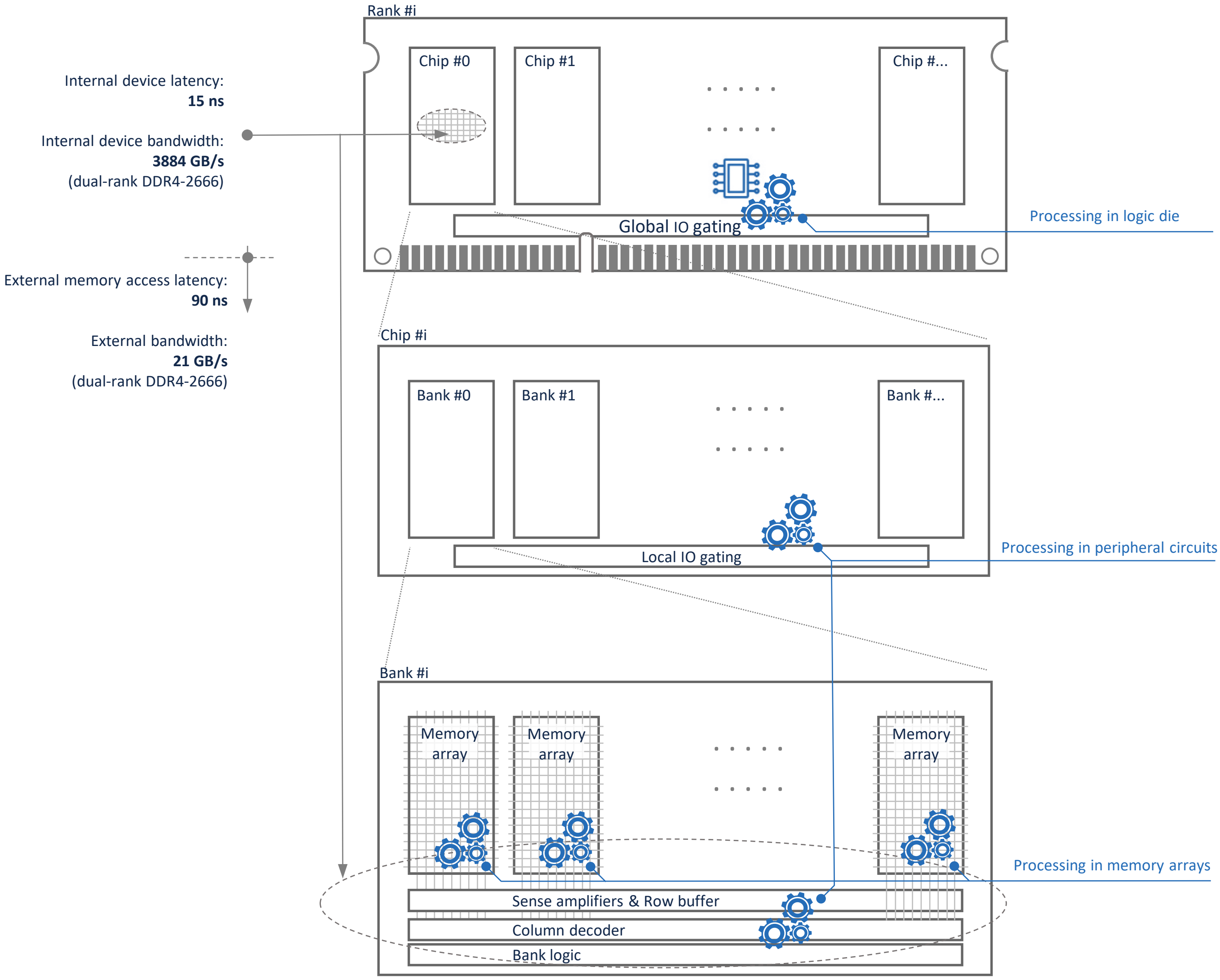


Figure 1: Internal DDR DRAM memory device performance significantly exceeds the performance seen externally by the user: starting from 6x in terms of latency and two orders of magnitude in terms of bandwidth. Huge internal data movement capabilities motivate the integration of processing directly into the memory devices. Processing in memory (PIM) can be performed at various levels: directly in the memory arrays, in peripheral circuitry or in logic layers and dies integrated into memory devices.

[1] The actual DRAM chip row size is assumed here to be 1 KB [6]. In the text by "row" we mean instead the combination of the rows activated in all the eight distinct DRAM chips that work in sync (lockstep mode).

[2] Internal bandwidth of a dual-rank DDR4-2666 DIMM is $\frac{128*1024\ [\text{Bytes}]}{62\ [\text{Cycles}]} = 2114\ \frac{\text{Bytes}}{\text{Cycle}} = 3884\ \frac{\text{GB}}{\text{s}}$.

3. Technology Readiness

As discussed in Section 2, integration of processing directly into memory devices has been motivated by the potential for huge internal data movement capabilities, low latency and high bandwidth. Processing in memory (PIM) is a generic term that covers any form of integration of processing capabilities into the memory system, of which there are many approaches; e.g. computation in memory arrays, peripheral circuits, and logic layers and dies integrated into the memory devices (see Figure 1). PIM has been explored by a large number of academic studies [3, 9] and it has already been demonstrated in various industrial prototypes and products:

Micron's Automata Processor is based on an adaptation of **DRAM memory arrays and peripheral circuitry** to achieve a hardware implementation of non-deterministic finite automata [10, 11]. The Automata Processor accelerates pattern-based algorithms, e.g. pattern matching and pattern recognition, which are common in many application domains including data mining, cybersecurity, bioinformatics, image and text processing. Another Micron's PIM architecture is **Micron In-Memory Intelligence (IMI)** [12]. The IMI architecture is a standard DRAM in form and function with the ability for massively parallel computation, by supporting vector instructions over the entire bank. The IMI targets big-data problems in imaging, vision, convolutional neural networks, content addressable memory, encryption, compression, search and sorting.

UPMEM's advanced DRAM chips embed **DRAM processing units (DPUs)** next to each DRAM bank. The current 500 MHz DPU implementation is successfully integrated in DDR4-2400 DIMMs manufactured in a 2x-nm DRAM process, and its first evaluations show significant performance improvements for pattern matching and database applications [13].

Through-silicon vias (TSVs) and 3D packaging of DRAM and logic dies enable a new approach to memory system architecture. **Hybrid Memory Cube (HMC)** [14] is the first architecture that exploits the concept of DRAM die stacking on top of a high-performance **logic die**. The logic die is responsible for DRAM sequencing, refresh, data routing, error correction and high-speed SERDES interconnect to the host CPU. Other natural candidates for inclusion in the HMC logic layer are low-operational high-data-intensity functions such as atomic memory operations, cache coherence management, scatter/gather operations and meta-data processing. **High Bandwidth Memory (HBM)** [15] is a JEDEC standard for 3D-stacked DRAM dies connected with TSVs. The HBM standard explicitly allows an optional **logic base layer** that can redistribute signals and implement logic functions.[3] HBM devices are already widely-used in the GPU domain and are adopted by some emerging processor architectures [16, 17, 18]. The first mainstream servers with high-end x86 processors and HBM are soon to hit the market.

Samsung has recently announced its **Function-In-Memory DRAM (FIMDRAM)** for HBM2, which integrates into the Single-Instruction Multiple-Data (SIMD) programmable processing units (**PCUs**) in each memory bank [19]. The HBM2 FIMDRAM chips are manufactured in a 20-nm DRAM process and the PCUs operate at 300 MHz. The PCUs exploit bank-level parallelism, with eight PCUs per pseudo-channel. They benefit from 4x higher data bandwidth than is possible with an off-chip memory solution and overall provide a total processing power of **1.2 TFLOPS** for FP16 matrix-vector calculations targeting machine learning applications.

PIM is traditionally associated with mainstream DRAM devices. In addition, numerous recent and ongoing studies are actively exploring the potential for **PIM in various levels of the memory hierarchy and in emerging memory technologies**. Processing in memory can also be performed in the **SRAM** used in on-chip caches or stand-alone accelerators. In Europe, computational SRAM design, chip manufacturing and programming is explored by **CEA**, the French Alternative Energies and Atomic Energy Commission [20, 21], **IBM Research Zurich** and **ETH Zurich** [22]. PIM capabilities are also demonstrated by various emerging memory technologies: phase-change memory (**PCM**) [23], **resistive-RAM (RRAM)** [24, 25, 26] and **Flash** [27].

[3] High-bandwidth memory devices, HMC and HBM, address the von Neumann bottleneck in two orthogonal ways. First, they increase the bandwidth between CPU and memory, and second, they support PIM extensions which delegate part of the computation to the memory devices.

In addition to demonstrating the capabilities and high performance of PIM, these studies also consider the complexity and the associated cost of integration. The costs of processing directly in the memory devices include additional silicon area, potentially leading reduced storage capacity, increased power and energy consumption, and device cost. The trade-off analyses clearly show superior performance-per-watt and significantly better total cost of ownership compared with conventional architectures.

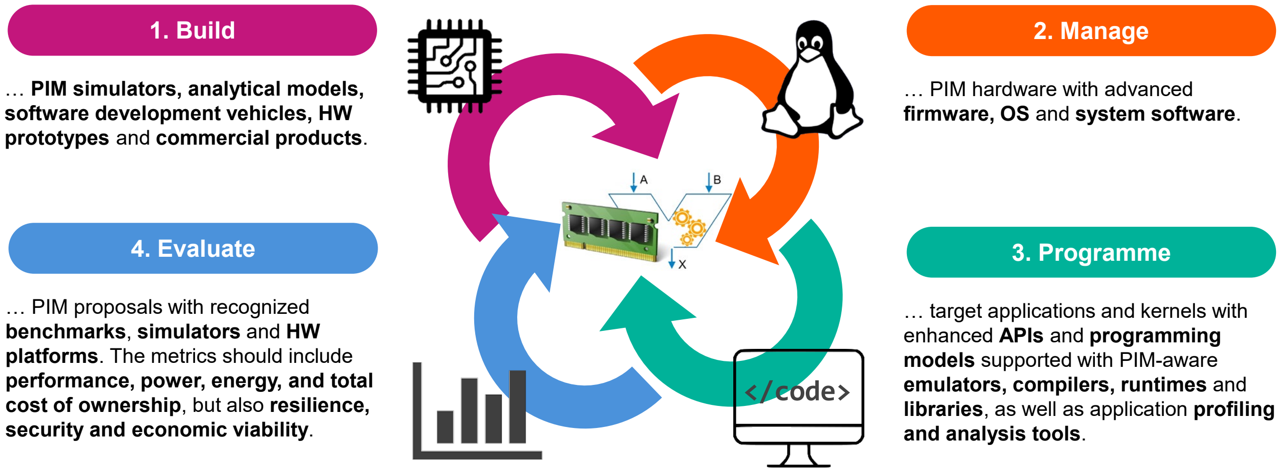
Finally, a commonly forgotten example of PIM that is already **widely implemented** in **production memory devices** is an **on-die Error Correction Code (ECC)**. Although on-die ECC is a fixed computation not programmable by the user, it is an important PIM example since it demonstrates capability to integrate computation directly in the memory peripheral circuits. The on-die ECC improves data integrity within many current **DDR, LPDDR and PCM devices** [28, 29, 30].

A simple form of Processing in Memory is already incorporated into numerous commercial memory devices. On-die ECC is an effective way to provide high memory density, capacity and bandwidth without compromising power and reliability.

4. Challenges

PIM is approaching a tipping point for three reasons. Firstly, it avoids the von Neumann bottleneck, a fundamental limitation to the effective use of computing systems for a large range of important data-centric applications. Secondly, it matches the community's requirement for efficient application acceleration by reducing the amount of expensive data transfers and, for some applications, it already demonstrates orders of magnitude improvements in performance and energy [10, 11, 12, 13]. Thirdly, it has reached a high technological readiness confirmed by various industrial prototypes and products.

Wide adoption of PIM, however, depends on our ability to create an ecosystem in which a number of PIM approaches can be designed and evaluated, leading to the selection of potential winners and their adoption by HPC system architects and end users. Creating the PIM ecosystem will require innovations and co-design across the whole stack, from technology, hardware, system, software and programming environment, to updating of algorithms and applications. To address these challenges, the HPC community has to:



1. Build

… **PIM simulators, analytical models, software development vehicles, HW prototypes** and **commercial products**.

2. Manage

… PIM hardware with advanced **firmware, OS** and **system software**.

3. Programme

… target applications and kernels with enhanced **APIs** and **programming models** supported with PIM-aware **emulators, compilers, runtimes** and **libraries**, as well as application **profiling and analysis tools**.

4. Evaluate

… PIM proposals with recognized **benchmarks**, **simulators** and **HW platforms**. The metrics should include **performance, power, energy, and total cost of ownership**, but also **resilience, security and economic viability**.

Figure 2: We have to create an ecosystem in which the community will be able to build, manage, program and evaluate PIM approaches. Only after a number of these proposals is designed and evaluated, will it be possible to select potential winners and enable their adoption by system architects and end users.

1. **Build PIM software development vehicles, prototypes and commercial products** that will be used by early adopters to develop and evaluate system software and applications. Design space exploration of the PIM proposals has to be supported with **PIM simulators** and **analytical models**. In addition to the PIM engines, the community has to consider the interfaces between the PIM accelerator and the rest of the system (e.g., host CPU). These interfaces can be designed specifically for PIM accelerators or they could be adopted from a range of emerging industrial solutions, such as CXL [31] and CCIX [32].

   Exploration of PIM should cover **all technology variants**, including **DRAM**, **SRAM, Flash** and **emerging technologies,** and the four **key domains** of **HPC, cloud, embedded** and **edge**. Also, the community should continue exploring PIM at all system levels, including on-chip caches, off-chip main memory and near storage (not covered in this document). Although, some of the emerging PIM technologies are more complex to integrate and not yet production ready, we should keep investing in their exploration as they open new systems design trade-offs and opportunities.

2. **Manage PIM hardware** with advanced **firmware, OS and system software**. PIM support in system software has to be tightly co-designed with PIM hardware to meet the requirements of production HPC, AI and data analytics systems. These requirements include efficient resource utilisation, performance, power, energy, but also system usability, resilience and security. To address these requirements, we need to consider the following problems:

   - **Applicability for HPC**. PIM devices must be capable of efficiently implementing common operations and patterns in HPC, most notably by supporting double-precision (and single-precision) floating-point arithmetic. Any constraints on code structure or size should be carefully considered in the context of HPC. Many current PIM devices are targeted towards ML/AI models, which are already important in HPC, but they are not well matched to classical scientific/engineering computing. Since not all HPC applications will be able to benefit from PIM in the short term, PIM devices should be able to be used either as conventional memory or a PIM accelerator, on an application-by-application basis, without rebooting the system.

   - **Data (cache) coherency**, if required in a host–accelerator PIM system. Data coherency adds complexity to the design, downgrades performance and limits scalability. However, the lack of coherency in systems in which the users expect it substantially limits the system usability. Therefore, data coherency should be considered from an early design stage for any host–accelerator system, and PIM accelerators should not be an exception.

   - **Data placement and alignment** in the memory system, e.g. at the level of channel, device, rank, bank, subarray, row and column. The main idea of PIM is to avoid data transfers between the location where the data initially resides and the location at which it is processed. Therefore, efficient PIM systems should ensure that all dependent operands are located (at least) in the same memory device, e.g. DIMM. Some PIM approaches even require perfect operand alignment at the level of memory subarray, row and column. In conventional systems, this is not possible.

   - The **security features** have to assure that PIM systems are as secure as conventional ones. This requires a special focus on any potential security issues that could be a consequence of an advanced PIM-specific designs. For example, as mentioned before, efficient use of PIM requires system software, and indirectly users, to control data placement and alignment within the physical devices. Before passing this control to the users, however, we have to understand and prevent any security exploits, such as RowHammer [33], that may be facilitated by this new ability. On the other hand, PIM solutions can also have native characteristics that make them exceptionally fit for secure computing [34].

   - The **resilience features** implemented in PIM systems should ensure that the failure rate of the system is below an acceptable threshold, representative of the technology, system size and target application [35]. This would require adjustments to current error detection and correction mechanisms to consider idiosyncratic PIM features such as analog computing directly in the memory arrays, or wide row-level operations that process kilobytes of data at a time. Quantitative analysis of the PIM device error rates and the effectiveness of the implemented resilience features should be performed from early and small-scale lab tests to the full-scale production systems, as soon as they are deployed.

> To build truly usable systems, resilience and security have to be considered from an early design stage, not as an afterthought.

3. **Programme** target applications and kernels with enhanced **Application Programming Interfaces (APIs) and programming models** supported with PIM-aware **emulators, compilers, runtimes and libraries**.

   The ideal scenario would be to develop backward-compatible PIM systems that provide full functionality and performance for existing codebases. Any modifications required to existing codes should be as small and local as possible, and should be done in a performance-portable and vendor-independent way controlled in the long term by open standards. As such, the community should focus in the short to medium term on efficient and productive PIM programming based on generic high-level (not device specific) and future-proof PIM APIs.

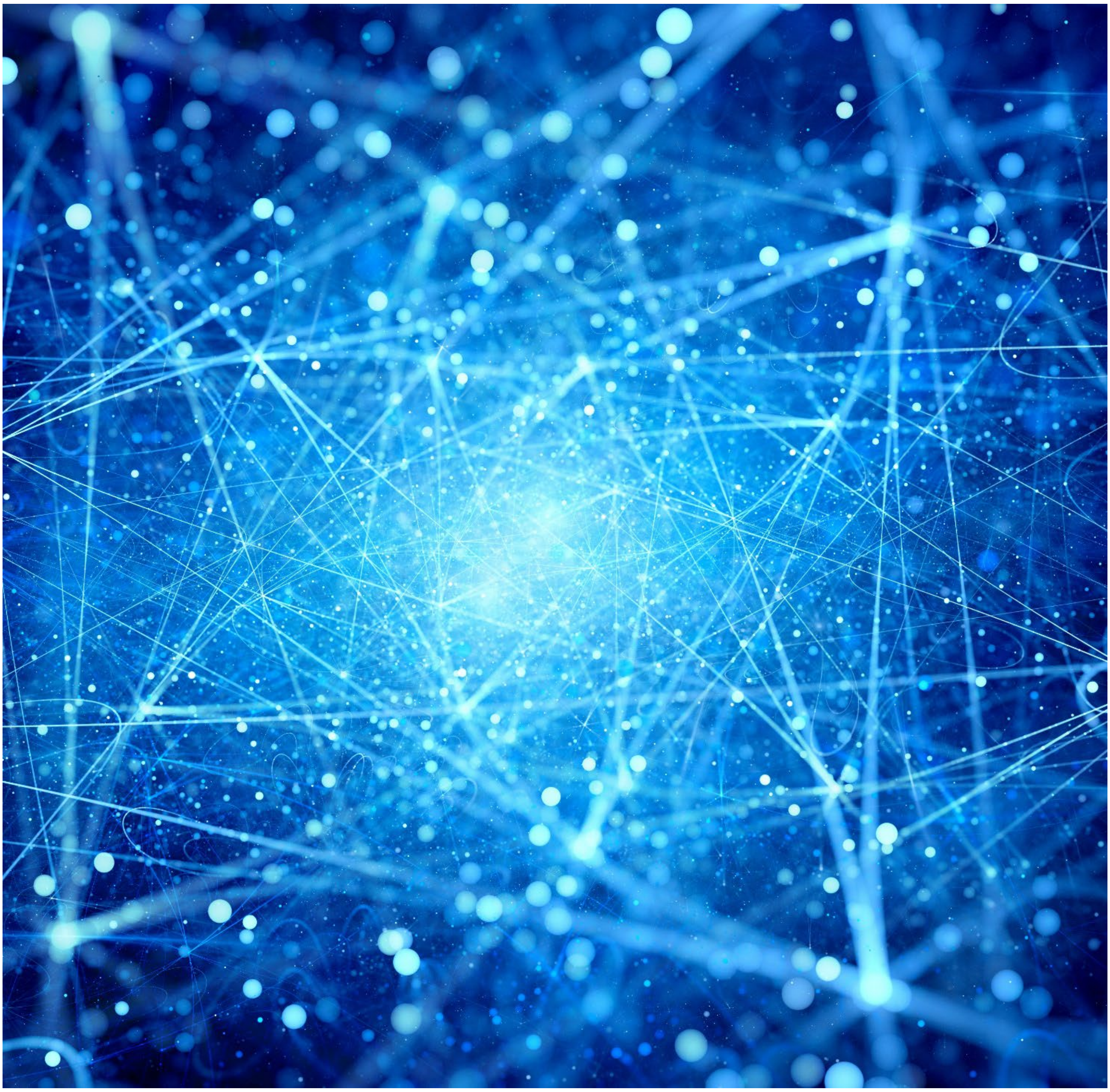
   The programmer should be able to interleave conventional and PIM APIs, being focused on the desired functionality and aware of the characteristics of generic PIM devices, but with the minimum exposure to device implementation details. This requires considerable effort to build an advanced software stack that will translate the high-level API into the correct and efficient PIM execution, scheduling functions, tasks and/or threads onto PIM devices in a composable and dynamic way, automating any necessary data transfers, conversions or coherency operations. The task is especially challenging for some PIM approaches, such as analog computing or bulk row-level operations, that require novel data types, instruction sets and arithmetic operations. Additionally, we have to provide users with **debugging tools** providing visibility into PIM execution and **profiling and analysis tools** capable of detecting application phases and kernels that are suitable for offloading to PIM.

   PIM programmability is the key for its wide adoption.

4. **Evaluate** PIM proposals with recognised **benchmarks, simulators and HW platforms**. This work requires the porting and tuning of benchmarks and production applications to a number of PIM platforms, with the objective of understanding which PIM designs are most promising and for which application domains. The evaluation metrics should include **performance, power, energy**, and **total cost of ownership**, but also **system usability, resilience** and **security.** Once a given PIM design is evaluated with benchmarks from various application domains, its potential market size should be contrasted with the production cost, leading to the overall estimate of its **economic viability**. This would also detect application domains that are the most suitable for PIM acceleration. These application domains would be the best candidates for early PIM adoption that would motivate **further investments** in the PIM ecosystem.

5. Conclusions

This white paper has reviewed why Processing in Memory is happening right now. We have discussed the fundamental motivations for PIM and surveyed the industrial prototypes and products that already demonstrate a high level of technological readiness. PIM is gaining momentum, but its wide acceptance in high-performance computing depends on our ability to create an ecosystem in which a number of PIM approaches can be designed and evaluated, leading to the selection of potential winners and adoption by system architects and end users. This paper identifies the main PIM challenges and provides guidelines to prioritise the research effort and funding, encompassing development and evaluation platforms, system software to manage the hardware, APIs and programming models to program it, profiling and analysis tools to analyse it, and benchmarks, simulators and platforms to evaluate the overall benefits. Only coordinated innovations and co-design across the whole stack can make PIM a reality in production.



References

[1] M. Radulovic, D. Zivanovic, D. Ruiz, B. R. d. Supinski, S. A. McKee, P. Radojković and E. Ayguadé, “Another Trip to the Wall: How Much Will Stacked DRAM Benefit HPC?,” in *Proceedings of the International Symposium on Memory Systems (MEMSYS)*, 2015.

[2] H. S. Stone, “A Logic-in-Memory Computer,” *IEEE Transactions on Computers,* vol. 19, 1970.

[3] P. Siegl, R. Buchty and M. Berekovic, “Data-Centric Computing Frontiers: A Survey on Processing-In-Memory,” in *Proceedings of the Second International Symposium on Memory Systems (MEMSYS)*, 2016.

[4] Eurolab4HPC Long-Term Vision on High-Performance Computing (2nd Edition), 2020.

[5] ETP4HPC’s SRA 4, “Strategic Research Agenda for High-performance Computing in Europe,” White Paper, 2020.

[6] Samsung Electronics Co., Ltd., “288pin Registered DIMM based on 4Gb E-die,” DDR4 SDRAM Datasheet, 2017.

[7] S. Li, Z. Yang, D. Reddy, A. Srivastava and B. Jacob, “DRAMsim3: A Cycle-Accurate, Thermal-Capable DRAM Simulator,” *IEEE Computer Architecture Letters,* vol. 19, 2020.

[8] M. Radulovic, K. Asifuzzaman, D. Zivanovic, N. Rajovic, G. C. d. Verdiére, D. Pleiter, M. Marazakisl, N. Kallimanis, P. Carpenter, P. Radojković and E. Ayguadé, “Mainstream vs. Emerging HPC: Metrics, Trade-Offs and Lessons Learned,” in *Proceedings of 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 2018.

[9] O. Mutlu, S. Ghose, J. Gómez-Luna and R. Ausavarungnirun, “A Modern Primer on Processing in Memory,” in *arXiv*, 2020.

[10] K. Wang, K. Angstadt, C. Bo, N. Brunelle, E. Sadredini, T. Tracy, J. Wadden, M. Stan and K. Skadron, “An Overview of Micron's Automata Processor,” in *Proceedings of the Eleventh IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES)*, 2016.

[11] P. Dlugosch, D. Brown, P. Glendenning, M. Leventhal and H. Noyes, “An Efficient and Scalable Semiconductor Architecture for Parallel Automata Processing,” *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, no. 12, 2014.

[12] T. Finkbeiner, G. Hush, T. Larsen, P. Lea, J. Leidel and T. Manning, “In-Memory Intelligence,” *IEEE Micro,* vol. 37, no. 4, 2017.

[13] F. Devaux, “The True Processing in Memory Accelerator,” *IEEE Hot Chips Symposium (HCS),* 2019.

[14] J. Jeddeloh and B. Keeth, “Hybrid Memory Cube New DRAM Architecture Increases Density and Performance,” in *Proceedings of Symposium on VLSI Technology (VLSIT)*, 2012.

[15] JEDEC Solid State Technology Association, “High Bandwidth Memory (HBM) DRAM,” White Paper, 2013.

[16] J. Jeffers, J. Reinders and a. A. Sodani, Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition (2nd ed.), 2016.

[17] FUJITSU LIMITED, “FUJITSU Supercomputer PRIMEHPC Specifications,” White Paper, 2020.

[18] FUJITSU LIMITED, “FUJITSU Supercomputer PRIMEHPC FX1000,” White Paper, 2020.

[19] Y. Kwon et al., “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 *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, 2021.

[20] J.-P. Noel, M. Pezzin, R. Gauchi, J.-F. Christmann, M. Kooli, H.-P. Charles, L. Ciampolini, M. Diallo, F. Lepin, B. Blampey, P. Vivet, S. Mitra and B. Giraud, “A 35.6 TOPS/W/mm² 3-Stage Pipelined Computational SRAM with Adjustable Form Factor for Highly Data-Centric Applications,” *IEEE Solid-State Circuits Letters,* vol. 3, 2020.

[21] M. Kooli, H.-P. Charles, C. Touzet, B. Giraud and J.-P. Noel, “Smart Instruction Codes for In-Memory Computing Architectures Compatible with Standard SRAM Interfaces,” in *Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018.

[22] R. Khaddam-Aljameh, P.-A. Francese, L. Benini and E. Eleftheriou, “An SRAM-Based Multibit In-Memory Matrix-Vector Multiplier with a Precision that Scales Linearly in Area, Time, and Power,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 29, 2021.

[23] R. Khaddam-Aljameh et al., “HERMES Core – A 14nm CMOS and PCM-based In-Memory Compute Core using an array of 300ps/LSB Linearized CCO-based ADCs and local digital processing,” in *Proc. Symposium on VLSI Circuits*, 2021.

[24] A. Sebastian, M. L. Gallo, R. Khaddam-Aljameh and E. Eleftheriou, “Memory devices and applications for in-memory computing,” *Nature Nanotechnology,* no. July, p. 529–544, 2020.

[25] M. Giordano, K. Prabhu, K. Koul, R. M. Radway, A. Gural, R. Doshi, Z. F. Khan, J. W. Kustin, T. Liu, G. B. Lopes, V. Turbiner, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, G. Lallement, B. Murmann, S. Mitra and P. Raina, “CHIMERA: A 0.92 TOPS, 2.2 TOPS/W Edge AI Accelerator with 2 MByte On-Chip Foundry Resistive RAM for Efficient Training and Inference,” *Symposium on VLSI Circuits (VLSI),* 2021.

[26] A. Valentian, F. Rummens, E. Vianello, T. Mesquida, C. L.-M. d. Boissac, O. Bichler and C. Reita, “Fully Integrated Spiking Neural Network with Analog Neurons and RRAM Synapses,” *IEEE International Electron Devices Meeting (IEDM),* pp. 14.3.1-14.3.4, 2019.

[27] Microchip Technology Inc., “Enhancing System Architecture Implementation for AI Applications, Microchip Delivers its Analog Embedded SuperFlash Technology,” News Release, 2019.

[28] Micron Technology, Inc., “ECC Brings Reliability and Power Efficiency to Mobile Devices,” White Paper, 2017.

[29] M. Patel, J. S. Kim, H. Hassan and O. Mutlu, “Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices,” in *Proceedings of 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2019.

[30] P. Amato, C. Laurent, M. Sforzin, S. Bellini, M. Ferrari and A. Tomasoni, “Ultra fast, two-bit ECC for Emerging Memories,” in *IEEE 6th International Memory Workshop (IMW)*, 2014.

[31] D. D. Sharma, “Compute express link,” White Paper, 2019.

[32] B. Benton, “CCIX, GEN-Z, OpenCAPI: Overview and Comparison.,” White Paper, 2017.

[33] Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai and O. Mutlu, “Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors,” in *Proceedings of ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, 2014.

[34] UPMEM, “UPMEM PIM Security Benefits - Architecture and Features Overview,” White Paper, 2020.

[35] P. Radojković et al., “Towards Resilient EU HPC Systems: A Blueprint,” *European HPC resilience initiative,* 2020.

Authors:

Petar Radojković is leader of the Memory systems team at the Barcelona Supercomputing Center (BSC).

Paul Carpenter is leader of the Microservers and System Software team at the Barcelona Supercomputing Center (BSC).

Pouya Esmaili-Dokht is a senior engineer in the BSC Memory systems team.

Rémy Cimadomo leads technological partnerships and business development at UPMEM, Grenoble.

Henri-Pierre Charles is Research Director at CEA research center, Grenoble.

Abu Sebastian is a Distinguished Research Staff Member and Technical Manager at IBM Research - Zurich.

Paolo Amato is Distinguished Member of Technical Staff and Director of the Memory System Pathfinding team at Micron, Vimercate, Italy.

This work was supported by the Spanish Government (contract PID2019-107255GB), Generalitat de Catalunya (contracts 2017-SGR-1328 and 2017-SGR-1414), and the European Union's Horizon 2020 research and innovation programme under grant agreements No 955606 (DEEP-SEA) and No 682675 (Projected Memristor European Research Council grant). Paul Carpenter holds the Ramon y Cajal fellowship under contracts RYC2018-025628-I of the Ministry of Economy and Competitiveness of Spain. This work was also supported by the Collaboration Agreement between Micron Technology, Inc. and BSC. The authors wish to thank Xavier Martorell from BSC for his technical support, and Manolis Marazakis and André Brinkmann for their feedback.

Cite as: P. Radojković et al., « Processing in Memory: the Tipping Point », ETP4HPC White Paper, 2021, doi 10.5281/zenodo.4767489.

DOI: 10.5281/zenodo.4767489