P&S Processing-in-Memory

Exploring the Processing-in-Memory Paradigm for Future Computing Systems

Dr. Juan Gómez Luna
Prof. Onur Mutlu

ETH Zürich
Spring 2022
10 March 2022

**Semester** Spring Semester 2022  
**Lecturers** J. Gómez Luna  
**Periodicity** every semester recurring course  
**Language of instruction** English  
**Comment** Only for Electrical Engineering and Information Technology BSc.  
The course unit can only be taken once. Repeated enrollment in a later semester is not creditable.

<table>
<thead>
<tr>
<th>Courses</th>
<th>Catalogue data</th>
<th>Performance assessment</th>
<th>Learning materials</th>
<th>Groups</th>
<th>Restrictions</th>
<th>Offered in</th>
</tr>
</thead>
</table>

**Abstract**
The category of "Laboratory Courses, Projects, Seminars" includes courses and laboratories in various formats designed to impart practical knowledge and skills. Moreover, these classes encourage independent experimentation and design, allow for explorative learning and teach the methodology of project work.

**Objective**
Data movement between the memory units and the compute units of current computing systems is a major performance and energy bottleneck. From large-scale servers to mobile devices, data movement costs dominate computation costs in terms of both performance and energy consumption. For example, data movement between the main memory and the processing cores accounts for 62% of the total system energy in consumer applications. As a result, the data movement bottleneck is a huge burden that greatly limits the energy efficiency and performance of modern computing systems. This phenomenon is an undesired effect of the dichotomy between memory and the processor, which leads to the data movement bottleneck.

Many modern and important workloads such as machine learning, computational biology, graph processing, databases, video analytics, and real-time data analytics suffer greatly from the data movement bottleneck. These workloads are exemplified by irregular memory accesses, relatively low data reuse, low cache line utilization, low arithmetic intensity (i.e., ratio of operations per accessed byte), and large datasets that greatly exceed the main memory size. The computation in these workloads cannot usually compensate for the data movement costs. In order to alleviate this data movement bottleneck, we need a paradigm shift from the traditional processor-centric design, where all computation takes place in the compute units, to a more data-centric design where processing elements are placed closer to or inside where the data resides. This paradigm of computing is known as Processing-in-Memory (PIM).

This is your perfect P&S if you want to become familiar with the main PIM technologies, which represent "the next big thing" in Computer Architecture. You will work hands-on with the first real-world PIM architecture, will explore different PIM architecture designs for important workloads, and will develop tools to enable research of future PIM systems. Projects in this course span software and hardware as well as the software/hardware interface. You can potentially work on developing and optimizing new workloads for the first real-world PIM hardware or explore new PIM designs in simulators, or do something else that can forward our understanding of the PIM paradigm.

---

Data movement between the memory units and the compute units of current computing systems is a major performance and energy bottleneck. From large-scale servers to mobile devices, data movement costs dominate computation costs in terms of both performance and energy consumption. For example, data movement between the main memory and the processing cores accounts for 62% of the total system energy in consumer applications. As a result, the data movement bottleneck is a huge burden that greatly limits the energy efficiency and performance of modern computing systems. This phenomenon is an undesired effect of the dichotomy between memory and the processor, which leads to the data movement bottleneck.

Many modern and important workloads such as machine learning, computational biology, graph processing, databases, video analytics, and real-time data analytics suffer greatly from the data movement bottleneck. These workloads are exemplified by irregular memory accesses, relatively low data reuse, low cache line utilization, low arithmetic intensity (i.e., ratio of operations per accessed byte), and large datasets that greatly exceed the main memory size. The computation in these workloads cannot usually compensate for the data movement costs. In order to alleviate this data movement bottleneck, we need a paradigm shift from the traditional processor-centric design, where all computation takes place in the compute units, to a more data centric design where processing elements are placed closer to or inside where the data resides. This paradigm of computing is known as Processing-in Memory (PIM).

This is your perfect P&S if you want to become familiar with the main PIM technologies, which represent "the next big thing" in Computer Architecture. You will work hands-on with the first real-world PIM architecture, will explore different PIM architecture designs for important workloads, and will develop tools to enable research of future PIM systems. Projects in this course span software and hardware as well as the software/hardware interface. You can potentially work on developing and optimizing new workloads for the first real world PIM hardware or explore new PIM designs in simulators, or do something else that can forward our understanding of the PIM paradigm.
A memory access consumes \( \sim 1000X \) the energy of a complex addition.
Goals of this P&S Course
We will introduce the **data movement bottleneck**, which is a major threat to high performance and energy efficiency of current computing systems.

You will learn what are **key workload characteristics** that make them more prone to the data movement bottleneck.

You will review traditional approaches to alleviating data movement and will **get familiar with new research proposals**: processing-in-memory solutions.

You will **work hands-on**: analyzing workloads, programming PIM architectures, simulating new PIM proposals, etc.
A +50-Year-Old Paradigm

Kautz, “Cellular Logic-in-Memory Arrays”, IEEE TC 1969

IEEE TRANSACTIONS ON COMPUTERS, VOL. C-18, NO. 8, AUGUST 1969

Cellular Logic-in-Memory Arrays

WILLIAM H. KAUTZ, MEMBER, IEEE

Abstract—As a direct consequence of large-scale integration, many advantages in the design, fabrication, testing, and use of digital circuitry can be achieved if the circuits can be arranged in a two-dimensional iterative, or cellular, array of identical elementary networks, or cells. When a small amount of storage is included in each cell, the same array may be regarded either as a logically enhanced memory array, or as a logic array whose elementary gates and connections can be “programmed” to realize a desired logical behavior.

In this paper the specific engineering features of such cellular logic-in-memory (CLIM) arrays are discussed, and one such special-purpose array, a cellular sorting array, is described in detail to illustrate how these features may be achieved in a particular design. It is shown how the cellular sorting array can be employed as a single-address, multiword memory that keeps in order all words stored within it. It can also be used as a content-addressed memory, a pushdown memory, a buffer memory, and (with a lower logical efficiency) a programmable array for the realization of arbitrary switching functions. A second version of a sorting array, operating on a different sorting principle, is also described.

Index Terms—Cellular logic, large-scale integration, logic arrays logic in memory, push-down memory, sorting, switching functions.

Fig. 1. Cellular sorting array I.

https://doi.org/10.1109/T-C.1969.222754
Processing in/near Memory: An Old Idea


A Logic-in-Memory Computer

HAROLD S. STONE

Abstract—If, as presently projected, the cost of microelectronic arrays in the future will tend to reflect the number of pins on the array rather than the number of gates, the logic-in-memory array is an extremely attractive computer component. Such an array is essentially a microelectronic memory with some combinational logic associated with each storage element.
UPMEM Processing-in-DRAM Engine (2019)

- Processing in DRAM Engine
  - Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.

- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth

CPU (x86, ARM, RV...)

[Link to AnandTech article](https://www.anandtech.com/show/14750/hot-chips-31-analysis-inmemory-processing-by-upmem)
Experimental Analysis of the UPMEM PIM Engine

Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
IZZAT EL HAJJ, American University of Beirut, Lebanon
IVAN FERNANDEZ, ETH Zürich, Switzerland and University of Malaga, Spain
CHRISTINA GIANNOULA, ETH Zürich, Switzerland and NTUA, Greece
GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Many modern workloads, such as neural networks, databases, and graph processing, are fundamentally memory-bound. For such workloads, the data movement between main memory and CPU cores imposes a significant overhead in terms of both latency and energy. A major reason is that this communication happens through a narrow bus with high latency and limited bandwidth, and the low data reuse in memory-bound workloads is insufficient to amortize the cost of main memory access. Fundamentally addressing this data movement bottleneck requires a paradigm where the memory system assumes an active role in computing by integrating processing capabilities. This paradigm is known as processing-in-memory (PIM).

Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPMEM company has designed and manufactured the first publicly-available real-world PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Units (DPU), integrated in the same chip.

This paper provides the first comprehensive analysis of the first publicly-available real-world PIM architecture. We make two key contributions. First, we conduct an experimental characterization of the UPMEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present PrIM (Processing-In-Memory benchmarks), a benchmark suite of 16 workloads from different application domains (e.g., dense/sparse linear algebra, databases, data analytics, graph processing, neural networks, bioinformatics, image processing), which we identify as memory-bound. We evaluate the performance and scaling characteristics of PrIM benchmarks on the UPMEM PIM architecture, and compare their performance and energy consumption to their state-of-the-art CPU and GPU counterparts. Our extensive evaluation conducted on two real UPMEM-based PIM systems with 640 and 2,556 DPs provides new insights about suitability of different workloads to the PIM system, programming recommendations for software designers, and suggestions and hints for hardware and architecture designers of future PIM systems.

UPMEM PIM System Summary

- Juan Gomez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu,

"Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware"


[arXiv version]
[PrIM Benchmarks Source Code]
[Slides (pptx) (pdf)]
[Talk Video (37 minutes)]
[Lightning Talk Video (3 minutes)]
Understanding a Modern PIM Architecture

Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization

Juan Gómez Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, Onur Mutlu

https://github.com/CMU-SAFARI/prim-benchmarks

SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture
2,579 views • Streamed live on Jul 12, 2021

https://www.youtube.com/watch?v=D8Hjy2iU9l4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9
Samsung Develops Industry’s First High Bandwidth Memory with AI Processing Power

Korea on February 17, 2021

The new architecture will deliver over twice the system performance and reduce energy consumption by more than 70%

Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry’s first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power — the HBM-PIM. The new processing-in-memory (PIM) architecture brings powerful AI computing capabilities inside high-performance memory, to accelerate large-scale processing in data centers, high performance computing (HPC) systems and AI-enabled mobile applications.

Kwangil Park, senior vice president of Memory Product Planning at Samsung Electronics stated, “Our groundbreaking HBM-PIM is the industry’s first programmable PIM solution tailored for diverse AI-driven workloads such as HPC, training and inference. We plan to build upon this breakthrough by further collaborating with AI solution providers for even more advanced PIM-powered applications.”

Samsung Function-in-Memory DRAM (2021)

- FIMDRAM based on HBM2

Chip Specification
- 128DQ / 8CH / 16 banks / BL4
- 32 PCU blocks (1 FIM block/2 banks)
- 1.2 TFLOPS (4H)
- FP16 ADD / Multiply (MUL) / Multiply-Accumulate (MAC) / Multiply-and-Add (MAD)

[3D Chip Structure of HBM with FIMDRAM]
Chip Implementation

- Mixed design methodology to implement FIMDRAM
  - Full-custom + Digital RTL

[Digital RTL design for PCU block]
Samsung AxDIMM (2021)

- DIMM-based PIM
  - DLRM recommendation system

SK hynix Develops PIM, Next-Generation AI Accelerator

February 16, 2022

Seoul, February 16, 2022

SK hynix (or "the Company", www.skhynix.com) announced on February 16 that it has developed PIM*, a next-generation memory chip with computing capabilities.

*PIM (Processing in Memory): A next-generation technology that provides a solution for data congestion issues for AI and big data by adding computational functions to semiconductor memory

It has been generally accepted that memory chips store data and CPU or GPU, like human brain, process data. SK hynix, following its challenge to such notion and efforts to pursue innovation in the next-generation smart memory, has found a breakthrough solution with the development of the latest technology.

SK hynix plans to showcase its PIM development at the world's most prestigious semiconductor conference, 2022 ISSCC*, in San Francisco at the end of this month. The company expects continued efforts for innovation of this technology to bring the memory-centric computing, in which semiconductor memory plays a central role, a step closer to the reality in devices such as smartphones.

*ISSCC: The International Solid-State Circuits Conference will be held virtually from Feb. 20 to Feb. 24 this year with a theme of "Intelligent Silicon for a Sustainable World"

For the first product that adopts the PIM technology, SK hynix has developed a sample of GDDR6-AiM (Accelerator in memory). The GDDR6-AiM adds computational functions to GDDR6* memory chips, which process data at 16Gbps. A combination of GDDR6-AiM with CPU or GPU instead of a typical DRAM makes certain computation speed 16 times faster. GDDR6-AiM is widely expected to be adopted for machine learning, high-performance computing, and big data computation and storage.

Key Takeaways

- This P&S is aimed at improving your
  - Knowledge in Computer Architecture and Processing-in-Memory
  - Technical skills in programming parallel (PIM) architectures and CompArch simulation
  - Critical thinking and analysis
  - Interaction with a nice group of researchers
  - Familiarity with key research directions
  - Technical presentation of your project
Key Goal

(Learn how to) overcome the data movement bottleneck by programming, benchmarking, exploring different designs of the PIM computing paradigm
Prerequisites of the Course

- Digital Design and Computer Architecture (or equivalent course)
  - https://safari.ethz.ch/digitaltechnik/spring2022/doku.php?id=schedule

- Familiarity with C/C++ programming
  - FPGA implementation or GPU programming (desirable)

- Interest in
  - future computer architectures and computing paradigms
  - discovering why things do or do not work and solving problems
  - making systems efficient and usable
Course Info: Who Are We? (I)

- **Onur Mutlu**
  - Full Professor @ ETH Zurich ITET (INFK), since September 2015
  - Strecker Professor @ Carnegie Mellon University ECE/CS, 2009-2016, 2016-...
  - PhD from UT-Austin, worked at Google, VMware, Microsoft Research, Intel, AMD
  - [https://people.inf.ethz.ch/omutlu/](https://people.inf.ethz.ch/omutlu/)
  - [omutlu@gmail.com](mailto:omutlu@gmail.com) (Best way to reach me)
  - [https://people.inf.ethz.ch/omutlu/projects.htm](https://people.inf.ethz.ch/omutlu/projects.htm)

- **Research and Teaching in:**
  - Computer architecture, computer systems, hardware security, bioinformatics
  - Memory and storage systems
  - Hardware security, safety, predictability
  - Fault tolerance
  - Hardware/software cooperation
  - Architectures for bioinformatics, health, medicine
  - ...
Course Info: Who Are We? (II)

- **Lead Supervisor:**
  - Dr. Juan Gómez Luna

- **Supervisors:**
  - Dr. Haiyu Mao
  - Geraldo F. Oliveira
  - Konstantinos Kanellopoulos
  - Nika Mansouri Ghiasi

- Get to know us and our research
  - [https://safari.ethz.ch/safari-group/](https://safari.ethz.ch/safari-group/)
Onur Mutlu’s SAFARI Research Group

Computer architecture, HW/SW, systems, bioinformatics, security, memory

https://safari.ethz.ch/safari-newsletter-january-2021/

Think BIG, Aim HIGH!

https://safari.ethz.ch
SAFARI Live Seminars (I)

SAFARI Live Seminars in Computer Architecture
Dr. Juan Gómez Luna, ETH Zurich
Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization

SAFARI Live Seminars in Computer Architecture
Dr. Andrew Walker, Schlüter Corporation & Nexgen Power Systems
An Addiction to Low Cost Per Memory Bit – How to Recognize it and What to Do About It

SAFARI Live Seminars in Computer Architecture
Gennady Palshenko, University of Toronto
Efficient DNN Training at Scale: from Algorithms to Hardware

SAFARI Live Seminars in Computer Architecture
Jawad Haj-Yahya, Huawei Research Center Zurich
Power Management Mechanisms in Modern Microprocessors and Their Security Implications

SAFARI Live Seminars in Computer Architecture
Minas Patel, ETH Zurich
Enabling Effective Error Mitigation in Memory Chips That Use On-Die ECCs

SAFARI Live Seminars in Computer Architecture
Christina Giannoulou, National Technical University of Athens
Efficient Synchronisation: Support for Near-Data-Processing Architectures

SAFARI Live Seminars in Computer Architecture
Geraldo F. Oliveira, ETH Zurich
DAMON: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

SAFARI Live Seminars in Computer Architecture
Ataberk Olgun, TOBB & ETH Zurich
QUAC-PRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Contemporary DRAM Chips

https://safari.ethz.ch/safari-seminar-series/
Overview

- CODIC substrate enables greater control over DRAM internal circuit timings
- CODIC is an efficient and low-cost way to enable new functionalities and optimizations in DRAM
- CODIC controls four key signals that orchestrate DRAM internal circuit timings
  - wordline (wl): Connects DRAM cells to bitlines
  - sense_p and sense_n: Trigger sense amplifiers
  - EQ: Triggers the logic that prepares a DRAM bank for the next access

SAFARI Live Seminar: Lois Orosa, 10 Feb 2022

Posted on January 16, 2022 by ewent

Join us for our next SAFARI Live Seminar with Lois Orosa.
Thursday, February 10 at 5:00 pm Zurich time (CET)

Lois Orosa, SAFARI Research Group, ETH Zurich
CODIC: A Low-Cost Substrate for Enabling Custom In-DRAM Functionalities and Optimizations

Livestream on YouTube Link

Current Research Focus Areas

**Research Focus:** Computer architecture, HW/SW, bioinformatics

- Memory and storage (DRAM, flash, emerging), interconnects
- Heterogeneous & parallel systems, GPUs, systems for data analytics
- System/architecture interaction, new execution models, new interfaces
- Energy efficiency, fault tolerance, hardware security, performance
- Genome sequence analysis & assembly algorithms and architectures
- Biologically inspired systems & system design for bio/medicine

Broad research spanning apps, systems, logic with architecture at the center
Course Info: How About You?

- Let us know your background, interests
- Why did you join this P&S?
Course Requirements and Expectations

- Attendance required for all meetings
- Study the learning materials
- Each student will carry out a hands-on project
  - Build, implement, code, and design with close engagement from the supervisors
- Participation
  - Ask questions, contribute thoughts/ideas
  - Read relevant papers

We will help in all projects!
If your work is really good, you may get it published!
Course Website

- [https://safari.ethz.ch/projects_and_seminars/doku.php?id=processing_in_memory](https://safari.ethz.ch/projects_and_seminars/doku.php?id=processing_in_memory)

- Useful information about the course

- Check your email frequently for announcements

- We will also have Moodle for Q&A
Meeting 1: Learning Materials

Required materials:
1. Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun, "A Modern Primer on Processing in Memory"
   *[Tutorial Video on "Memory-Centric Computing Systems"]* (1 hour 51 minutes)

2. Onur Mutlu, "Memory-Centric Computing"
   *Education Class at Embedded Systems Week (ESWEEK), Virtual, 9 October 2021.*
   *[Slides (pptx) (pdf)] [Abstract (pdf)] [Talk Video (2 hours, including Q&A)] [Invited Paper at DATE 2021] ["A Modern Primer on Processing in Memory" paper]*

Recommended materials:

4. Computation in Memory (Professor Onur Mutlu, lecture, Fall 2020).
   *(PDF) (PPT) Video*

5. Near-data Processing (Professor Onur Mutlu, lecture, Fall 2020).
   *(PDF) (PPT) Video*

6. Real Processing-in-DRAM with UPMEM (Dr. Juan Gomez Luna, SAFARI Live Seminar, July 2021).
   "Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture"
   *[PrIM Benchmarks Source Code] [Slides (pptx) (pdf)] [Long Talk Slides (pptx) (pdf)] [Short Talk Slides (pptx) (pdf)] [SAFARI Live Seminar Slides (pptx) (pdf)] [SAFARI Live Seminar Video (2 hrs 57 mins)] [Lightning Talk Video (3 minutes)]
Meeting 2 (March 15\textsuperscript{th})

- We will \textbf{announce the projects} and will give you some description about them

- We will give you a chance to select a project

- Then, we will have \textbf{1-1 meetings} to match your interests, skills, and background with a suitable project

- It is important that you \textbf{study the learning materials} before our next meeting!
Next Meetings

- Individual meetings with your mentor/s

- Tutorials and short talks
  - PIM programming
  - Recent research works

- Presentation of your work
PIM Course (Fall 2021)

- **Fall 2021 Edition:**

- **Youtube Livestream:**
  - [https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi841fUYUK9EsXKhQKRPyX](https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi841fUYUK9EsXKhQKRPyX)

- **Project course**
  - Taken by Bachelor’s/Master’s students
  - Processing-in-Memory lectures
  - Hands-on research exploration
  - Many research readings

---

Fall 2021 Meetings/Schedule

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Livestream</th>
<th>Meeting</th>
<th>Learning Materials</th>
<th>Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W1</td>
<td>05.10</td>
<td>Live</td>
<td>M1: P&amp;S PIM Course Presentation</td>
<td>(PDF) (PPT)</td>
<td>HW 0 Out</td>
</tr>
<tr>
<td>W2</td>
<td>12.10</td>
<td>Live</td>
<td>M2: Real-World PIM Architectures</td>
<td>(PDF) (PPT)</td>
<td></td>
</tr>
<tr>
<td>W3</td>
<td>19.10</td>
<td>Live</td>
<td>M3: Real-World PIM Architectures II</td>
<td>(PDF) (PPT)</td>
<td></td>
</tr>
<tr>
<td>W4</td>
<td>26.10</td>
<td>Live</td>
<td>M4: Real-World PIM Architectures III</td>
<td>(PDF) (PPT)</td>
<td></td>
</tr>
<tr>
<td>W5</td>
<td>02.11</td>
<td>Live</td>
<td>M5: Real-World PIM Architectures IV</td>
<td>(PDF) (PPT)</td>
<td></td>
</tr>
<tr>
<td>W6</td>
<td>09.11</td>
<td>Live</td>
<td>M6: End-to-End Framework for Processing-using-Memory</td>
<td>(PDF) (PPT)</td>
<td></td>
</tr>
<tr>
<td>W7</td>
<td>16.11</td>
<td>Live</td>
<td>M7: How to Evaluate Data Movement Bottlenecks</td>
<td>(PDF) (PPT)</td>
<td></td>
</tr>
<tr>
<td>W8</td>
<td>23.11</td>
<td>Live</td>
<td>M8: Programming PIM Architectures</td>
<td>(PDF) (PPT)</td>
<td></td>
</tr>
<tr>
<td>W9</td>
<td>30.11</td>
<td>Live</td>
<td>M9: Benchmarking and Workload Suitability on PIM</td>
<td>(PDF) (PPT)</td>
<td></td>
</tr>
<tr>
<td>W10</td>
<td>07.12</td>
<td>Live</td>
<td>M10: Bit-Serial SIMD Processing using DRAM</td>
<td>(PDF) (PPT)</td>
<td></td>
</tr>
</tbody>
</table>
An Introduction to Processing-in-Memory
Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor.

Main memory system must scale (in size, technology, efficiency, cost, and management algorithms) to maintain performance growth and technology scaling benefits.
The Main Memory System

- Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor

- Main memory system must scale (in size, technology, efficiency, cost, and management algorithms) to maintain performance growth and technology scaling benefits
Main memory is a critical component of all computing systems: server, mobile, embedded, desktop, sensor.

Main memory system must scale (in size, technology, efficiency, cost, and management algorithms) to maintain performance growth and technology scaling benefits.
Most of the system is dedicated to storing and moving data
Three Key Systems Trends

1. Data access is a major bottleneck
   - Applications are increasingly data hungry

2. Energy consumption is a key limiter

3. Data movement energy dominates compute
   - Especially true for off-chip to on-chip movement
Example: Capacity, Bandwidth & Latency

Memory latency remains almost constant
The Need for More Memory Performance

**In-memory Databases**
[Mao+, EuroSys’12; Clapp+ (Intel), IISWC’15]

**Graph/Tree Processing**
[Xu+, IISWC’12; Umuroglu+, FPL’15]

**In-Memory Data Analytics**
[Clapp+ (Intel), IISWC’15; Awan+, BDCloud’15]

**Datacenter Workloads**
[Kanev+ (Google), ISCA’15]
DRAM Latency Is Critical for Performance

In-memory Databases

Graph/Tree Processing

Long memory latency $\rightarrow$ performance bottleneck

In-Memory Data Analytics
[Clapp+ (Intel), IISWC’15; Awan+, BDCloud’15]

Datacenter Workloads
[Kanev+ (Google), ISCA’15]
The Energy Perspective

Communication Dominates Arithmetic

Dally, HiPEAC 2015

- 64-bit DP 20pJ
- 256-bit buses
- 256-bit access 8 kB SRAM
- 26 pJ
- 256 pJ
- 16 nJ
- 500 pJ
- 50 pJ
- 1 nJ
- DRAM Rd/Wr
- Efficient off-chip link

20mm
A memory access consumes $\sim 1000X$ the energy of a complex addition
The Performance Perspective (1996-2005)

- “It’s the Memory, Stupid!” (Richard Sites, MPR, 1996)

The Performance Perspective (Today)

- All of Google’s Data Center Workloads (2015):

The Problem

Data access is the major performance and energy bottleneck

Our current design principles cause great energy waste
(and great performance loss)
The Problem

Processing of data is performed far away from the data
A Computing System

- Three key components
  - Computation
  - Communication
  - Storage/memory

Burks, Goldstein, von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument,” 1946.
A Computing System

- Three key components
  - Computation
  - Communication
  - Storage/memory

Burks, Goldstein, von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument,” 1946.

Yet ...

- “It’s the Memory, Stupid!” (Richard Sites, MPR, 1996)

Perils of Processor-Centric Design

- Grossly-imbalanced systems
  - Processing done only in **one place**
  - Everything else just stores and moves data: **data moves a lot**
    - Energy inefficient
    - Low performance
    - Complex

- Overly complex and bloated processor (and accelerators)
  - To tolerate data access from memory
  - Complex hierarchies and mechanisms
    - Energy inefficient
    - Low performance
    - Complex
Data Movement in Computing Systems

- **Data movement** dominates performance and is a major system **energy bottleneck**
  - Comprises **41% of mobile system energy** during web browsing

**Compute systems should be more data-centric**

**Processing-In-Memory** proposes **computing where it makes sense** (where data resides)

*Reducing data Movement Energy via Online Data Clustering and Encoding (MICRO’16)*

**Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms (IISWC’14)**
62.7% of the total system energy is spent on data movement
We Need A Paradigm Shift To …

- Enable computation with minimal data movement
- Compute where it makes sense (where data resides)
- Make computing architectures more data-centric
Why In-Memory Computation Today?

- Pull from systems/applications for data-centric execution
- It can be practical today
  - 3D-stacked memories combine logic and memory functionality (relatively) tightly + industry open to new architectures
Challenge and Opportunity for Future

High Performance and Energy Efficiency
Goal: Processing Inside Memory

Many questions... How do we design the:
- compute-capable memory & controllers?
- processor chip?
- software and hardware interfaces?
- system software and languages?
- algorithms?
- **Near-Data Processing or Processing In-Memory (PIM)**
  - Move *computation* closer to *where the data resides*

---

**Logic layer**
3D stacked DRAM

**Through-Silicon Via (TSV)**

**Memory controller**

**Memory module (DIMM)**
UPMEM Processing-in-DRAM Engine (2019)

- Processing in DRAM Engine
  - Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.

- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth

---

Samsung AxDIMM (2021)

- DIMM-based PIM
  - DLRM recommendation system

Possible Designs

- **Fixed-function** units
- **Reconfigurable** architectures
  - FPGAs, CGRA
- **General-purpose** programmable cores
  - E.g., ARM Cortex R-8, ARM Cortex A-35 (+SIMD units)
  - Possibility of running any workload
- **Processing-using-memory**:
  - Ambit: In-DRAM bulk bitwise operations (Seshadri+, MICRO’17)
  - SIMDREAM: End-to-end framework for SIMD in DRAM (Hajinazar+, ASPLOS’21)

![Diagram of possible designs including fixed-function units, reconfigurable architectures, low power cores, and processing-in-memory operations.](image-url)
5.2. Two Approaches: Processing Using Memory (PUM) vs. Processing Near Memory (PNM)

Many recent works take advantage of the memory technology innovations that we discuss in Section 5.1 to enable and implement PIM. We find that these works generally take one of two approaches, which are categorized in Table 1: (1) processing using memory or (2) processing near memory. We briefly describe each approach here. Sections 6 and 7 will provide example approaches and more detail for both.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Enabling Technologies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Using Memory</td>
<td>SRAM</td>
</tr>
<tr>
<td></td>
<td>DRAM</td>
</tr>
<tr>
<td></td>
<td>Phase-change memory (PCM)</td>
</tr>
<tr>
<td></td>
<td>Magnetic RAM (MRAM)</td>
</tr>
<tr>
<td></td>
<td>Resistive RAM (RRAM)/memristors</td>
</tr>
<tr>
<td>Processing Near Memory</td>
<td>Logic layers in 3D-stacked memory</td>
</tr>
<tr>
<td></td>
<td>Silicon interposers</td>
</tr>
<tr>
<td></td>
<td>Logic in memory controllers</td>
</tr>
</tbody>
</table>

Processing using memory (PUM) exploits the existing memory architecture and the operational principles of the memory circuitry to enable operations within main memory with minimal changes. PUM makes use

Processing in Memory: Two Approaches

1. Processing-using-Memory
2. Processing-near-Memory
Agenda

- Major Trends Affecting Memory
- Processing in Memory: Two Directions
  - Processing-using-Memory (PuM)
    - Minimally Changing Memory Chips
  - Processing-near-Memory (PnM)
    - Exploiting 3D-Stacked Memory
Approach 1: Minimally Changing DRAM

- DRAM has great capability to perform **bulk data movement and computation** internally with small changes
  - Can exploit internal bandwidth to move data
  - Can exploit analog computation capability
  - ...

**Examples: RowClone, In-DRAM AND/OR, Gather/Scatter DRAM**

- **RowClone**: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data (Seshadri et al., MICRO 2013)
- **Fast Bulk Bitwise AND and OR in DRAM** (Seshadri et al., IEEE CAL 2015)
- **Gather-Scatter DRAM**: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses (Seshadri et al., MICRO 2015)
- "**Ambit**: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology" (Seshadri et al., MICRO 2017)
- "**SIMDRAM**: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM" (Hajinazar et al., ASPLOS 2021)
RowClone: In-Memory Copy and Initialization
Starting Simple: Data Copy and Initialization

*memmove & memcpy: 5% cycles in Google's datacenter [Kanev+ ISCA'15]*

Forking

Zero initialization (e.g., security)

Checkpointing

VM Cloning

Deduplication

Page Migration

Many more
Today’s Systems: Bulk Data Copy

1) High latency
2) High bandwidth utilization
3) Cache pollution
4) Unwanted data movement

1046ns, 3.6uJ  (for 4KB page copy via DMA)
Future Systems: In-Memory Copy

1) Low latency
2) Low bandwidth utilization
3) No cache pollution
4) No unwanted data movement

1046ns, 3.6uJ $\rightarrow$ 90ns, 0.04uJ
RowClone: In-DRAM Row Copy

- Step 1: Activate row A
- Step 2: Activate row B

Idea: Two consecutive ACTivates

Negligible HW cost

Data Bus

Row Buffer (4 Kbytes)

8 bits

DRAM subarray

Transfer row

Transfer row

4 Kbytes
RowClone: Latency and Energy Savings

More on RowClone

- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry,

"RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"

Proceedings of the 46th International Symposium on Microarchitecture (MICRO), Davis, CA, December 2013. [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
RowClone Demonstration in Real DRAM Chips

ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs

Fei Gao  
feig@princeton.edu  
Department of Electrical Engineering  
Princeton University

Georgios Tziantzioulis  
georgios.tziantzioulis@princeton.edu  
Department of Electrical Engineering  
Princeton University

David Wentzlaff  
wentzlaf@princeton.edu  
Department of Electrical Engineering  
Princeton University

Ambit: In-Memory Bulk Bitwise Operations
In-Memory Bulk Bitwise Operations

- We can support in-DRAM COPY, ZERO, AND, OR, NOT, MAJ
- At low cost

- Using analog computation capability of DRAM
  - Idea: activating multiple rows performs computation

- 30-60X performance and energy improvement
In-DRAM AND/OR: Triple Row Activation

Final State
$AB + BC + AC$

$C(A + B) + \sim C(AB)$

In-DRAM Bulk Bitwise AND/OR Operation

- **BULKAND A, B \(\rightarrow C\)**
- **Semantics:** Perform a bitwise AND of two rows A and B and store the result in row C

- **R0** – reserved zero row, **R1** – reserved one row
- **D1, D2, D3** – Designated rows for triple activation

1. RowClone A into D1
2. RowClone B into D2
3. RowClone R0 into D3
4. ACTIVATE D1,D2,D3
5. RowClone Result into C
More on In-DRAM Bulk AND/OR

- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,

"Fast Bulk Bitwise AND and OR in DRAM"


---

**Fast Bulk Bitwise AND and OR in DRAM**

Vivek Seshadri*, Kevin Hsieh*, Amirali Boroumand*, Donghyuk Lee*, Michael A. Kozuch†, Onur Mutlu*, Phillip B. Gibbons†, Todd C. Mowry*

*Carnegie Mellon University †Intel Pittsburgh
In-DRAM NOT: Dual Contact Cell

Idea:
Feed the negated value in the sense amplifier into a special row

Figure 5: A dual-contact cell connected to both ends of a sense amplifier

In-DRAM NOT Operation

Figure 5: Bitwise NOT using a dual contact capacitor

Performance: In-DRAM Bitwise Operations

![Graph showing throughput of bitwise operations on various systems]

Figure 9: Throughput of bitwise operations on various systems.
Energy of In-DRAM Bitwise Operations

<table>
<thead>
<tr>
<th>Design</th>
<th>not</th>
<th>and/or</th>
<th>nand/nor</th>
<th>xor/xnor</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM &amp; DDR3</td>
<td>93.7</td>
<td>137.9</td>
<td>137.9</td>
<td>137.9</td>
</tr>
<tr>
<td>Ambit Channel Energy</td>
<td>1.6</td>
<td>3.2</td>
<td>4.0</td>
<td>5.5</td>
</tr>
<tr>
<td>(nJ/KB) (↓)</td>
<td>59.5X</td>
<td>43.9X</td>
<td>35.1X</td>
<td>25.1X</td>
</tr>
</tbody>
</table>

Table 3: Energy of bitwise operations. (↓) indicates energy reduction of Ambit over the traditional DDR3-based design.

Example Data Structure: Bitmap Index

- Alternative to B-tree and its variants
- Efficient for performing *range queries* and *joins*
- Many bitwise operations to perform a query

<table>
<thead>
<tr>
<th>Age Range</th>
<th>Bitmap</th>
</tr>
</thead>
<tbody>
<tr>
<td>age &lt; 18</td>
<td>Bitmap 1</td>
</tr>
<tr>
<td>18 &lt; age &lt; 25</td>
<td>Bitmap 2</td>
</tr>
<tr>
<td>25 &lt; age &lt; 60</td>
<td>Bitmap 3</td>
</tr>
<tr>
<td>age &gt; 60</td>
<td>Bitmap 4</td>
</tr>
</tbody>
</table>
Performance: Bitmap Index on Ambit

Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.

More on Ambit


Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology

Vivek Seshadri\textsuperscript{1,5} Donghyuk Lee\textsuperscript{2,5} Thomas Mullins\textsuperscript{3,5} Hasan Hassan\textsuperscript{4} Amirali Boroumand\textsuperscript{5}

Jeremie Kim\textsuperscript{4,5} Michael A. Kozuch\textsuperscript{3} Onur Mutlu\textsuperscript{4,5} Phillip B. Gibbons\textsuperscript{5} Todd C. Mowry\textsuperscript{5}

\textsuperscript{1}Microsoft Research India \textsuperscript{2}NVIDIA Research \textsuperscript{3}Intel \textsuperscript{4}ETH Zürich \textsuperscript{5}Carnegie Mellon University
SIMDRAM Framework

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu,

"SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"

[2-page Extended Abstract]
[Short Talk Slides (pptx) (pdf)]
[Talk Slides (pptx) (pdf)]
[Short Talk Video (5 mins)]
[Full Talk Video (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar\(^1,2\)
Nika Mansouri Ghiasi\(^1\)

Geraldo F. Oliveira\(^1\)
Minesh Patel\(^1\)
Juan Gómez-Luna\(^1\)

Sven Gregorio\(^1\)
Mohammed Alser\(^1\)
Onur Mutlu\(^1\)
João Dinis Ferreira\(^1\)
Saugata Ghose\(^3\)

\(^1\)ETH Zürich  \(^2\)Simon Fraser University  \(^3\)University of Illinois at Urbana–Champaign
Agenda

- Major Trends Affecting Memory

- Processing in Memory: Two Directions
  - Processing-using-Memory (PuM)
    - Minimally Changing Memory Chips
  - Processing-near-Memory (PnM)
    - Exploiting 3D-Stacked Memory
Approach 2: 3D-Stacked Logic+Memory
Graph Processing

- Large graphs are everywhere (circa 2015)
  - 36 Million Wikipedia Pages
  - 1.4 Billion Facebook Users
  - 300 Million Twitter Users
  - 30 Billion Instagram Photos

- Scalable large-scale graph processing is challenging

<table>
<thead>
<tr>
<th>Cores</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>+42%</td>
</tr>
<tr>
<td>128</td>
<td></td>
</tr>
</tbody>
</table>

Only +42% for 4x more cores!!!
Key Bottlenecks in Graph Processing

**PageRank algorithm** (Page et al. 1999)

```cpp
for (v: graph.vertices) {
    for (w: v.successors) {
        w.next_rank += weight * v.rank;
    }
}
```

1. Frequent random memory accesses
2. Little amount of computation

PageRank algorithm (Page et al. 1999)
Two Key Questions in 3D-Stacked PIM

- How can we accelerate important applications if we use 3D-stacked memory as a coarse-grained accelerator?
  - what is the architecture and programming model?
  - what are the mechanisms for acceleration?

- What is the minimal processing-in-memory support we can provide?
  - without changing the system significantly
  - while achieving significant benefits
Tesseract: An In-Memory Accelerator for Graph Processing
Tesseract System for Graph Processing

Interconnected set of 3D-stacked memory+logic chips with simple cores

Host Processor

Memory-Mapped Accelerator Interface
(Noncacheable, Physically Addressed)

In-Order Core

Message Queue

Logic

Crossbar Network

Memory

DRAM Controller

LP

PF Buffer

MTP

Ahn+, “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing” ISCA 2015
Tesseract System for Graph Processing

- **Evaluation on**
  - DDR3 DRAM, computation on Out-of-Order (OoO) core
  - Hybrid Memory Cube (HMC) DRAM, computation on Out-of-Order (OoO) core
  - HMC DRAM, computation on the Memory Controller (MC)

- **Tesseract**
  - With or without List Prefetching (LP)
  - With or without Message Triggered Prefetching (MTP), specified by the programmer
Tesseract Graph Processing Performance

>13X Performance Improvement

On five graph processing algorithms

Speedup

- DDR3-OoO
- HMC-OoO
- HMC-MC
- Tesseract
- Tesseract-LP
- Tesseract-LP-MTP

Ahn+, “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing” ISCA 2015
Ahn+, “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing” ISCA 2015
More on Tesseract

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi,
  "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing"
  Proceedings of the 42nd International Symposium on Computer Architecture (ISCA),
  Portland, OR, June 2015.
  [Slides (pdf)] [Lightning Session Slides (pdf)]

A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Junwhan Ahn  Sungpack Hong§  Sungjoo Yoo  Onur Mutlu†  Kiyoung Choi
junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
Seoul National University  §Oracle Labs  †Carnegie Mellon University
Two Key Questions in 3D-Stacked PIM

- How can we accelerate important applications if we use 3D-stacked memory as a coarse-grained accelerator?
  - what is the architecture and programming model?
  - what are the mechanisms for acceleration?

- What is the minimal processing-in-memory support we can provide?
  - without changing the system significantly
  - while achieving significant benefits
PIM-Enabled Instructions for Graph Processing
Simple PIM Operations as ISA Extensions (I)

```java
for (v: graph.vertices) {
    value = weight * v.rank;
    for (w: v.successors) {
        w.next_rank += value;
    }
}

PageRank algorithm (Page et al. 1999)
```

Host Processor

Main Memory

Conventional Architecture

64 bytes in
64 bytes out
for (v: graph.vertices) {
    value = weight * v.rank;
    for (w: v.successors) {
        pim.add_r1, (r2)
    }
}

PageRank algorithm (Page et al. 1999)

In-Memory Addition

8 bytes in
0 bytes out

Host Processor

Main Memory

value

w.next_rank
PEI: Benchmarks

- **Graph processing**
  - Average Teenage Follower (AT)
  - Breadth-First Search (BFS)
  - PageRank (PR)
  - Single-Source Shortest Path (SP)
  - Weakly Connected Components (WCC)

- **Other benchmarks** that can benefit from PEI
  - **Data analytics**
    - Hash Join (HJ)
    - Histogram (HG)
    - Radix Partitioning (RP)
  - **Machine learning and data mining**
    - Streamcluster (SC)
    - Support Vector Machine (SVM)
PEI: PIM-Enabled Instructions: Examples

Table 1: Summary of Supported PIM Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>R</th>
<th>W</th>
<th>Input</th>
<th>Output</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-byte integer increment</td>
<td>O</td>
<td>O</td>
<td>0 bytes</td>
<td>0 bytes</td>
<td>AT</td>
</tr>
<tr>
<td>8-byte integer min</td>
<td>O</td>
<td>O</td>
<td>8 bytes</td>
<td>0 bytes</td>
<td>BFS, SP, WCC</td>
</tr>
<tr>
<td>Floating-point add</td>
<td>O</td>
<td>O</td>
<td>8 bytes</td>
<td>0 bytes</td>
<td>PR</td>
</tr>
<tr>
<td>Hash table probing</td>
<td>O</td>
<td>X</td>
<td>8 bytes</td>
<td>9 bytes</td>
<td>HJ</td>
</tr>
<tr>
<td>Histogram bin index</td>
<td>O</td>
<td>X</td>
<td>1 byte</td>
<td>16 bytes</td>
<td>HG, RP</td>
</tr>
<tr>
<td>Euclidean distance</td>
<td>O</td>
<td>X</td>
<td>64 bytes</td>
<td>4 bytes</td>
<td>SC</td>
</tr>
<tr>
<td>Dot product</td>
<td>O</td>
<td>X</td>
<td>32 bytes</td>
<td>8 bytes</td>
<td>SVM</td>
</tr>
</tbody>
</table>

- Executed either in memory or in the processor: dynamic decision
- Low-cost locality monitoring for a single instruction
- Cache-coherent, virtually-addressed, single cache block only
- Atomic between different PEIs
- Not atomic with normal instructions (use pfence for ordering)
Example PEI Microarchitecture
PEI Performance Delta: Large Data Sets

(Large Inputs, Baseline: CPU-Only)

Percentage of Performance Improvement wrt Baseline (CPU-only)

- ATF
- BFS
- PR
- SP
- WCC
- HJ
- HG
- RP
- SC
- SVM
- GeoMean

PIM-Only
Locality-Aware

Locality-Aware = PIM or CPU depending on data location
PEI Energy Consumption

Breakdown of Energy Consumption on Different System Components:

- **Cache**
- **HMC Link**
- **DRAM**
- **Host-side PCU**
- **Memory-side PCU**
- **PMU**

**Legend:**
- **Host-Only (CPU)**
- **PIM-Only**
- **Locality-Aware**
More on PIM-Enabled Instructions

Agenda

- Major Trends Affecting Memory

- Processing in Memory: Two Directions
  - Processing-using-Memory (PuM)
    - Minimally Changing Memory Chips
  - Processing-near-Memory (PnM)
    - Exploiting 3D-Stacked Memory
How to Enable Adoption of Processing in Memory
Barriers to Adoption of PIM

1. Functionality of and applications & software for PIM

2. Ease of programming (interfaces and compiler/HW support)

3. System support: coherence & virtual memory

4. Runtime and compilation systems for adaptive scheduling, data mapping, access/sharing control

5. Infrastructures to assess benefits and feasibility

All can be solved with change of mindset
We Need to Revisit the Entire Stack

We can get there step by step
A Modern Primer on Processing in Memory

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{b,c}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungnirun\textsuperscript{d}

SAFARI Research Group

\textsuperscript{a} ETH Zürich
\textsuperscript{b} Carnegie Mellon University
\textsuperscript{c} University of Illinois at Urbana-Champaign
\textsuperscript{d} King Mongkut's University of Technology North Bangkok

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun, "A Modern Primer on Processing in Memory"

A Modern Primer on Processing in Memory

Onur Mutlu\textsuperscript{a,b}, Saugata Ghose\textsuperscript{c}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungnirun\textsuperscript{d}

SAFARI Research Group

\textsuperscript{a}\textit{ETH Zürich}
\textsuperscript{b}\textit{Carnegie Mellon University}
\textsuperscript{c}\textit{University of Illinois at Urbana-Champaign}
\textsuperscript{d}\textit{King Mongkut's University of Technology North Bangkok}

Abstract

Modern computing systems are overwhelmingly designed to move data to computation. This design choice goes directly against at least three key trends in computing that cause performance, scalability and energy bottlenecks: (1) data access is a key bottleneck as many important applications are increasingly data-intensive, and memory bandwidth and energy do not scale well, (2) energy consumption is a key limiter in almost all computing platforms, especially server and mobile systems, (3) data movement, especially off-chip to on-chip, is very expensive in terms of bandwidth, energy and latency, much more so than computation. These trends are especially severely-felt in the data-intensive server and energy-constrained mobile systems of today.

At the same time, conventional memory technology is facing many technology scaling challenges in terms of reliability, energy, and performance. As a result, memory system architects are open to organizing memory in different ways and making it more intelligent, at the expense of higher cost. The emergence of 3D-stacked memory plus logic, the adoption of error correcting codes inside the latest DRAM chips, proliferation of different main memory standards and chips, specialized for different purposes (e.g., graphics, low-power, high bandwidth, low latency), and the necessity of designing new solutions to serious reliability and security issues, such as the RowHammer phenomenon, are an evidence of this trend.

This chapter discusses recent research that aims to practically enable computation close to data, an approach we call processing-in-memory (PIM). PIM places computation mechanisms in or near where the data is stored (i.e., inside the memory chips, in the logic layer of 3D-stacked memory, or in the memory controllers), so that data movement between the computation units and memory is reduced or eliminated. While the general idea of PIM is not new, we discuss motivating trends in applications as well as memory circuits/technology that greatly exacerbate the need for enabling it in modern computing systems. We examine at least two promising new approaches to designing PIM systems to accelerate important data-intensive applications: (1) processing using memory by exploiting analog operational properties of DRAM chips to perform massively-parallel operations in memory, with low-cost changes, (2) processing near memory by exploiting 3D-stacked memory technology design to provide high memory bandwidth and low memory latency to in-memory logic. In both approaches, we describe and tackle relevant cross-layer research, design, and adoption challenges in devices, architecture, systems, and programming models. Our focus is on the development of in-memory processing designs that can be adopted in real computing platforms at low cost. We conclude by discussing work on solving key challenges to the practical adoption of PIM.

Keywords: memory systems, data movement, main memory, processing-in-memory, near-data processing, computation-in-memory, processing using memory, processing near memory, 3D-stacked memory, non-volatile memory, energy efficiency, high-performance computing, computer architecture, computing paradigm, emerging technologies, memory scaling, technology scaling, dependable systems, robust systems, hardware security, system security, latency, low-latency computing
1. Introduction

Main memory, built using the Dynamic Random Access Memory (DRAM) technology, is a major component in nearly all computing systems, including servers, cloud platforms, mobile/embedded devices, and sensor systems. Across all of these systems, the data working set sizes of modern applications are rapidly growing, while the need for fast analysis of such data is increasing. Thus, main memory is becoming an increasingly significant bottleneck across a wide variety of systems and applications [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Alleviating the main memory bottleneck requires the memory capacity, energy, cost, and performance to all scale in an efficient manner across technology generations. Unfortunately, it has become increasingly difficult in recent years, especially the past decade, to scale all of these dimensions [1, 2, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49], and thus the main memory bottleneck has been worsening.

A major reason for the main memory bottleneck is the high energy and latency cost associated with data movement. In modern computers, to perform any operation on data that resides in main memory, the processor must retrieve the data from main memory. This requires the memory controller to issue commands to a DRAM module across a relatively slow and power-hungry off-chip bus (known as the memory channel). The DRAM module sends the requested data across the memory channel, after which the data is placed in the caches and registers. The CPU can perform computation on the data once the data is in its registers. Data movement from the DRAM to the CPU incurs long latency and consumes a significant amount of energy [7, 50, 51, 52, 53, 54]. These costs are often exacerbated by the fact that much of the data brought into the caches is not reused by the CPU [52, 53, 55, 56], providing little benefit in return for the high latency and energy cost.

The cost of data movement is a fundamental issue with the processor-centric nature of contemporary computer systems. The CPU is considered to be the master in the system, and computation is performed only in the processor (and accelerators). In contrast, data storage and communication units, including the main memory, are treated as unintelligent workers that are incapable of computation. As a result of this processor-centric design paradigm, data moves a lot in the system between the computation units and communication/storage units so that computation can be done on it. With the increasingly data-centric nature of contemporary and emerging appli-
P&S Processing-in-Memory

Exploring the Processing-in-Memory Paradigm for Future Computing Systems

Dr. Juan Gómez Luna
Prof. Onur Mutlu
ETH Zürich
Spring 2022
10 March 2022