P&S Processing-in-Memory

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

Dr. Juan Gómez Luna
Prof. Onur Mutlu
ETH Zürich
Fall 2021
5 October 2021
# P&S: Processing-in-Memory (I)


<table>
<thead>
<tr>
<th>Semester</th>
<th>Autumn Semester 2021</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecturers</td>
<td>J. Gómez Luna</td>
</tr>
<tr>
<td>Periodicity</td>
<td>every semester recurring course</td>
</tr>
<tr>
<td>Language of instruction</td>
<td>English</td>
</tr>
<tr>
<td>Comment</td>
<td>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.</td>
</tr>
</tbody>
</table>

<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>
<th>Overview</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 ~1000X the energy of a complex addition.
Goals of this P&S Course
P&S Processing-in-Memory: Contents

- 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
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 GIANNOLA, 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 (DPUs), 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 DPUs 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.

Understanding a Modern PIM Architecture

SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture

2,579 views • Streamed live on Jul 12, 2021

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

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

https://www.youtube.com/watch?v=D8Hjy2iU9I4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9
Samsung Develops Industry’s First High Bandwidth Memory with AI Processing Power

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**

![3D Chip Structure of HBM with FIMDRAM]

**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)

**ISSCC 2021 / SESSION 25 / DRAM / 25.4**

25.4 A 28nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Heon Kwon¹, Suk Han Lee¹, Jaehoon Lee¹, Sang-Hyuk Kwon¹, Je Min Ryu¹, Jong-Hi Son¹, Seongil O¹, Hak-Soo Yu¹, Haesuk Lee¹, Soo Young Kim¹, Youngmin Cho¹, Jin Guk Kim¹, Jongyoon Choi¹, Hyeon-Sung Shin¹, Jin Kim¹, BangSeng Phua¹, HyungiMin Kim¹, Myeong Jun Song¹, Ahn Choi¹, Daeho Kim¹, SooYoung Kim¹, Eun-Bong Kim¹, David Wang¹, Shinhae Kang¹, Yuhwan Ro¹, Seungwoo Seo¹, JoonHo Song¹, Jeryoun Youn¹, Kyomin Sohn¹, Nam Sung Kim¹

¹Samsung Electronics, Hwasung, Korea
²Samsung Electronics, San Jose, CA
³Samsung Electronics, Suwon, Korea
Chip Implementation

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

[Digital RTL design for PCU block]

A 20nm 8GB Function-in-Memory DRAM, Based on HBM2 with 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young Choon Kee1, Suk Hak Lee1, Jaehoon Lee1, Sang Hyuk Kee1, Ja Min Ryu1, Jong P-H Choi2, Seong-Ho Han3, Sung Ju Yu4, Hamak Lee1, Soo Young Kim1, Youngmin Cho1, Jin Suk Kim1, Jongyoo Cho1, Hyeon-Sang Shin1, Jin Kerr, Bengtae Phan1, Honggin Kim1, Myung Jin Song5, Ahn Choi5, Daeho Kim5, Soo Young Kim1, Eun-Bong Kim1, Daeil Wang5, Shin-Haeng Kong5, Yuhwan polo, Seogyeong Sree1, Jongho Song1, Jaeyoul You1, Hyunje Seo1, Nam Sung Kim1

1Samsung Electronics, Hwasung, Korea
2Samsung Electronics, San Jose, CA
3Samsung Electronics, Seongnam, Korea
Samsung AxDIMM (2021)

- DIMM-based PIM
  - DLRM recommendation system

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.
(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)

- 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/
- omutlu@gmail.com (Best way to reach me)
- 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-april-2020/

Think BIG, Aim HIGH!

https://safari.ethz.ch
Dear SAFARI friends,

Happy New Year! We are excited to share our group highlights with you in this second edition of the SAFARI newsletter (You can find the first edition from April 2020 here). 2020 has...
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, Schiliton 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 Pekhimenko, 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
Gerald 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-MRG: High Throughput Trace Random Number Generation Using Quadruple Base Activation in Context of DRAM Chips

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

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

SAFARI Live Seminars in Computer Architecture
Jawad Haj-Yahya, Huawei Research Center Zurich

https://safari.ethz.ch/safari-seminar-series/
SAFARI Live Seminar - Data-Centric & Data-Aware Frameworks for Fundamentally Efficient Data Handling

2 waiting • Scheduled for Oct 27, 2021

Onur Mutlu Lectures
19K subscribers

Title: Data-Centric and Data-Aware Frameworks for Fundamentally Efficient Data Handling in Modern Computing Systems
Speaker: Nastaran Hajinazar, SAFARI Research Group, https://www.linkedin.com/in/nastaran-...
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

Hybrid Main Memory

Heterogeneous Processors and Accelerators

Persistent Memory/Storage

Graphics and Vision Processing

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

- Required materials:


- 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, lecture, Fall 2020). (PDF) (PPT)Video
Meeting 2 (October 12\textsuperscript{th})

- We will 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 1-1 meetings to match your interests, skills, and background with a suitable project

- It is important that you 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
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.
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.
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

- **Capacity**
- **Bandwidth**
- **Latency**

Memory latency remains almost constant.

- Capacity improvement: 128x
- Bandwidth improvement: 20x
- Latency improvement: 1.3x
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]
Communication Dominates Arithmetic

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

Dally, HiPEAC 2015
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)*
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?

Diagram:
- Processor Core
- Cache
- Memory
- Database
- Graphs
- Media
- Interconnect
- Query
- Results

List:
- Problem
- Algorithm
- Program/Language
- System Software
- SW/HW Interface
- Micro-architecture
- Logic
- Devices
- Electrons
Processing In-Memory (PIM)

- Near-Data Processing or Processing In-Memory (PIM)
  - Move *computation* closer to *where the data resides*

---

**Logic layer**
3D stacked DRAM

**Memory controller**

**Memory module (DIMM)**
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

---

**UPMEM Processing-in-DRAM Engine (2019)**

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)
  - SIMDRAM: End-to-end framework for SIMD in DRAM (Hajinazar+, ASPLOS’21)
Agenda

- Major Trends Affecting Memory

- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips
  - 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
- Many more
- VM Cloning
- Deduplication
- Page Migration
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

Idea: Two consecutive ACTivates

Negligible HW cost

Step 1: Activate row A

Step 2: Activate row B

DRAM subarray

Row Buffer (4 Kbytes)

4 Kbytes

8 bits

Data Bus
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 → 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"

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

Seshadri+，“Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology,” MICRO 2017
In-DRAM NOT Operation

Figure 5: Bitwise NOT using a dual contact capacitor

Seshadri+, “Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology,” MICRO 2017
Performance: In-DRAM Bitwise Operations

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>DDR3</td>
<td>93.7</td>
<td>137.9</td>
<td>137.9</td>
<td>137.9</td>
</tr>
<tr>
<td>Ambit (↓)</td>
<td>1.6</td>
<td>3.2</td>
<td>4.0</td>
<td>5.5</td>
</tr>
<tr>
<td>59.5X</td>
<td>43.9X</td>
<td>35.1X</td>
<td>25.1X</td>
<td></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

age < 18  18 < age < 25  25 < age < 60  age > 60

Bitmap 1  Bitmap 2  Bitmap 3  Bitmap 4
Performance: Bitmap Index on Ambit

![Bar chart showing performance comparison between Baseline and Ambit. The chart includes execution times for different query scenarios with varying user counts.](chart)

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

More on Ambit

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)]
Agenda

- Major Trends Affecting Memory
- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips
  - 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>0</td>
</tr>
<tr>
<td>128</td>
<td>1</td>
</tr>
</tbody>
</table>

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

PageRank algorithm (Page et al. 1999)

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

1. Frequent random memory accesses
2. Little amount of computation
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
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
Ahn+, “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing” ISCA 2015.
Tesseract Graph Processing System Energy

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"


[Slides (pdf)] [Lightning Session Slides (pdf)]
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 \textit{minimal processing-in-memory support} we can provide?
  - without changing the system significantly
  - while achieving significant benefits
PIM-Enabled Instructions for Graph Processing
for (v: graph.vertices) {
    value = weight * v.rank;
    for (w: v.successors) {
        w.next_rank += value;
    }
}

PageRank algorithm (Page et al. 1999)
for (v: graph.vertices) {
    value = weight * v.rank;
    for (w: v.successors) {
        __pim_add(&w.next_rank, value);
    }
}

PageRank algorithm (Page et al. 1999)

Main Memory

8 bytes in
0 bytes out

In-Memory Addition
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

Host Processor

- Out-Of-Order Core
- PCU (PEI Computation Unit)
- L1 Cache
- L2 Cache
- Last-Level Cache
- PMU (PEI Mgmt Unit)
  - PIM Directory
  - Locality Monitor
- PMU
  - PCU
  - PMU
- HMC Controller

3D-stacked Memory

- PCU
- DRAM Controller
- DRAM Controller
- DRAM Controller

Example PEI uArchitecture
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

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

Agenda

- Major Trends Affecting Memory
- Processing in Memory: Two Directions
  - Minimally Changing Memory Chips
  - Exploiting 3D-Stacked Memory
Eliminating the Adoption Barriers

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

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

Dr. Juan Gómez Luna
Prof. Onur Mutlu

ETH Zürich
Fall 2021
5 October 2021