Intelligent Architectures for Intelligent Systems

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
14 October 2021
Huawei STW 2021: SW/HW Co-Design for Comp Arch
The Problem

Computing is Bottlenecked by Data
Data is Key for AI, ML, Genomics, …

- Important workloads are all data intensive

- They require rapid and efficient processing of large amounts of data

- Data is increasing
  - We can generate more than we can process
Data is Key for Future Workloads

**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]
Data Overwhelms Modern Machines

In-memory Databases

Graph/Tree Processing

Data → performance & energy bottleneck

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

Datacenter Workloads
[Kanev+ (Google), ISCA’15]
Data is Key for Future Workloads

Chrome
Google’s web browser

TensorFlow Mobile
Google’s machine learning framework

VP9
Video Playback
Google’s video codec

VP9
Video Capture
Google’s video codec
Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

VP9

Video Playback
Google’s video codec

Video Capture
Google’s video codec

SAFE
Data is Key for Future Workloads

Development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

Genome Analysis

1. Sequencing
2. Read Mapping
3. Variant Calling
4. Scientific Discovery

Data → performance & energy bottleneck
Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017
Published: 02 April 2018  Article history ▼

[Open arxiv.org version]
New Genome Sequencing Technologies

Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017
Published: 02 April 2018 Article history ▼

Data → performance & energy bottleneck
Accelerating Genome Analysis: A Primer on an Ongoing Journey

Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,
"Accelerating Genome Analysis: A Primer on an Ongoing Journey"
[Slides (pptx)(pdf)]
[Talk Video (1 hour 2 minutes)]
GenASM Framework [MICRO 2020]


[Lighting Talk Video (1.5 minutes)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (18 minutes)]
[Slides (pptx) (pdf)]
Future of Genome Sequencing & Analysis

Mohammed Alser, Zülal Bingöl, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, Onur Mutlu


Accelerating Genome Analysis: A Primer on an Ongoing Journey
DOI Bookmark: 10.1109/MM.2020.3013728

FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications
DOI Bookmark: 10.1109/MM.2021.3088396

MinION from ONT

SmidgION from ONT
More on Fast & Efficient Genome Analysis …

- Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  Invited Lecture at Technion, Virtual, 26 January 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hour 37 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]
Detailed Lectures on Genome Analysis

- **Computer Architecture, Fall 2020, Lecture 3a**
  - *Introduction to Genome Sequence Analysis* (ETH Zürich, Fall 2020)
  - [](https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5)

- **Computer Architecture, Fall 2020, Lecture 8**
  - *Intelligent Genome Analysis* (ETH Zürich, Fall 2020)
  - [](https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14)

- **Computer Architecture, Fall 2020, Lecture 9a**
  - *GenASM: Approx. String Matching Accelerator* (ETH Zürich, Fall 2020)
  - [](https://www.youtube.com/watch?v=XoLpzmN-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15)

- **Accelerating Genomics Project Course, Fall 2020, Lecture 1**
  - *Accelerating Genomics* (ETH Zürich, Fall 2020)
  - [](https://www.youtube.com/watch?v=rgjI8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqlgwIDRQDTyId)

[SAFARI](https://www.youtube.com/onurmutlulectures)
Data Overwhelms Modern Machines …

- Storage/memory capability
- Communication capability
- Computation capability

Greatly impacts robustness, energy, performance, cost
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.
Most of the system is dedicated to storing and moving data.

Yet, system is still bottlenecked by memory.
Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

Video Playback

Video Capture

VP9

Google’s video codec

Google’s video codec
Data Movement Overwhelms Modern Machines


62.7% of the total system energy is spent on data movement

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand\textsuperscript{1} Rachata Ausavarungrun\textsuperscript{1} Aki Kuusela\textsuperscript{3} Allan Knies\textsuperscript{3} Saugata Ghose\textsuperscript{1} Eric Shiu\textsuperscript{3} Rahul Thakur\textsuperscript{3} Parthasarathy Ranganathan\textsuperscript{3} Youngsok Kim\textsuperscript{2} Daehyun Kim\textsuperscript{4,3} Onur Mutlu\textsuperscript{5,1}

SAFARI
Axiom

An Intelligent Architecture
Handles Data Well
How to Handle Data Well

- Ensure data does not overwhelm the components
  - via intelligent algorithms
  - via intelligent architectures
  - via whole system designs: algorithm-architecture-devices

- Take advantage of vast amounts of data and metadata
  - to improve architectural & system-level decisions

- Understand and exploit properties of (different) data
  - to improve algorithms & architectures in various metrics
Corollaries: Architectures Today …

- **Architectures are terrible at dealing with data**
  - Designed to mainly store and move data vs. to compute
  - They are processor-centric as opposed to data-centric

- **Architectures are terrible at taking advantage of vast amounts of data** (and metadata) available to them
  - Designed to make simple decisions, ignoring lots of data
  - They make human-driven decisions vs. data-driven

- **Architectures are terrible at knowing and exploiting different properties of application data**
  - Designed to treat all data as the same
  - They make component-aware decisions vs. data-aware
Fundamentally Better Architectures

Data-centric

Data-driven

Data-aware
We Need to Revisit the Entire Stack

We can get there step by step
Axiom

To achieve the highest energy efficiency and performance:

we must take the expanded view
of computer architecture

Co-design across the hierarchy:
Algorithms to devices

Specialize as much as possible
within the design goals
"There's Plenty of Room at the Bottom: An Invitation to Enter a New Field of Physics" was a lecture given by physicist Richard Feynman at the annual American Physical Society meeting at Caltech on December 29, 1959. [1] Feynman considered the possibility of direct manipulation of individual atoms as a more powerful form of synthetic chemistry than those used at the time. Although versions of the talk were reprinted in a few popular magazines, it went largely unnoticed and did not inspire the conceptual beginnings of the field. Beginning in the 1980s, nanotechnology advocates cited it to establish the scientific credibility of their work.

https://en.wikipedia.org/wiki/There%27s_Plenty_of_Room_at_the_Bottom
Feynman considered some ramifications of a general ability to manipulate matter on an atomic scale. **He was particularly interested in the possibilities of denser computer circuitry, and microscopes that could see things much smaller than is possible with scanning electron microscopes.** These ideas were later realized by the use of the scanning tunneling microscope, the atomic force microscope and other examples of scanning probe microscopy and storage systems such as Millipede, created by researchers at IBM.

Feynman also suggested that it should be possible, in principle, to make nanoscale machines that "arrange the atoms the way we want", and do chemical synthesis by mechanical manipulation.

**He also presented the possibility of "swallowing the doctor",** an idea that he credited in the essay to his friend and graduate student Albert Hibbs. This concept involved building a tiny, swallowable surgical robot.
Historical: Opportunities at the Top

REVIEW

There’s plenty of room at the Top: What will drive computer performance after Moore’s law?

Charles E. Leiserson¹, Neil C. Thompson¹,², Joel S. Emer¹,³, Bradley C. Kuszmaul¹,⁴, Butler W. Lampson¹,⁴

See all authors and affiliations

Science 05 Jun 2020:
Vol. 368, Issue 6495, eaam9744
DOI: 10.1126/science.aam9744

Much of the improvement in computer performance comes from decades of miniaturization of computer components, a trend that was foreseen by the Nobel Prize-winning physicist Richard Feynman in his 1959 address, “There’s Plenty of Room at the Bottom,” to the American Physical Society. In 1975, Intel founder Gordon Moore predicted the regularity of this miniaturization trend, now called Moore’s law, which, until recently, doubled the number of transistors on computer chips every 2 years.

Unfortunately, semiconductor miniaturization is running out of steam as a viable way to grow computer performance—there isn’t much more room at the “Bottom.” If growth in computing power stalls, practically all industries will face challenges to their productivity. Nevertheless, opportunities for growth in computing performance will still be available, especially at the “Top” of the computing-technology stack: software, algorithms, and hardware architecture.
Axiom, Revisited

There is plenty of room both at the top and at the bottom

but much more so

when you

communicate well between and optimize across

the top and the bottom
Hence the Expanded View

Computer Architecture (expanded view)
Fundamentally Better Architectures

Data-centric

Data-driven

Data-aware
Data-Centric (Memory-Centric) Architectures
Data-Centric Architectures: Properties

- **Process data where it resides** *(where it makes sense)*
  - Processing in and near memory structures

- **Low-latency and low-energy data access**
  - Low latency memory
  - Low energy memory

- **Low-cost data storage and processing**
  - High capacity memory at low cost: hybrid memory, compression

- **Intelligent data management**
  - Intelligent controllers handling robustness, security, cost
Processing Data
Where It Makes Sense
Processing in/near Memory: An Old Idea


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.
Why In-Memory Computation Today?

- **Push from Technology**
  - **DRAM Scaling at jeopardy**
    - Controllers close to DRAM
    - Industry open to new memory architectures
Why In-Memory Computation Today?

- Push from Technology
- DRAM Scaling at jeopardy
- Controllers close to DRAM
- Industry open to new memory architectures
Memory Scaling Issues Were Real

- Onur Mutlu,
  "Memory Scaling: A Systems Architecture Perspective"
  *Proceedings of the 5th International Memory Workshop (IMW)*, Monterey, CA, May 2013. Slides (pptx) (pdf)
  EETimes Reprint

Memory Scaling: A Systems Architecture Perspective

Onur Mutlu  
Carnegie Mellon University  
onur@cmu.edu  
http://users.ece.cmu.edu/~omutlu/

As Memory Scales, It Becomes Unreliable

- Data from all of Facebook’s servers worldwide
- Meza+, “Revisiting Memory Errors in Large-Scale Production Data Centers,” DSN’15.
Large-Scale Failure Analysis of DRAM Chips

- Analysis and modeling of memory errors found in all of Facebook’s server fleet

- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, "Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field"

  *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN),* Rio de Janeiro, Brazil, June 2015.

  [Slides (pptx) (pdf)] [DRAM Error Model]

Revisiting Memory Errors in Large-Scale Production Data Centers:
Analysis and Modeling of New Trends from the Field

Justin Meza  Qiang Wu*  Sanjeev Kumar*  Onur Mutlu
Carnegie Mellon University  *Facebook, Inc.
Infrastructures to Understand Such Issues

An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study (Khan et al., SIGMETRICS 2014)

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case (Lee et al., HPCA 2015)

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems (Qureshi et al., DSN 2015)
Infrastructures to Understand Such Issues

PM, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.
SoftMC: Open Source DRAM Infrastructure


- Flexible
- Easy to Use (C++ API)
- Open-source

  github.com/CMU-SAFARI/SoftMC
SoftMC

- https://github.com/CMU-SAFARI/SoftMC

SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan\textsuperscript{1,2,3} Nandita Vijaykumar\textsuperscript{3} Samira Khan\textsuperscript{4,3} Saugata Ghose\textsuperscript{3} Kevin Chang\textsuperscript{3} Gennady Pekhimenko\textsuperscript{5,3} Donghyuk Lee\textsuperscript{6,3} Oguz Ergin\textsuperscript{2} Onur Mutlu\textsuperscript{1,3}

\textsuperscript{1}ETH Zürich \quad \textsuperscript{2}TOBB University of Economics & Technology \quad \textsuperscript{3}Carnegie Mellon University
\textsuperscript{4}University of Virginia \quad \textsuperscript{5}Microsoft Research \quad \textsuperscript{6}NVIDIA Research
A Curious Discovery [Kim et al., ISCA 2014]

One can predictably induce errors in most DRAM memory chips
The Story of RowHammer

- One can predictably induce bit flips in commodity DRAM chips
  - >80% of the tested DRAM chips are vulnerable

- First example of how a simple hardware failure mechanism can create a widespread system security vulnerability

Forget Software—Now Hackers Are Exploiting Physics
Repeatedly reading a row enough times (before memory gets refreshed) induces disturbance errors in adjacent rows in most real DRAM chips you can buy today.
Most DRAM Modules Are Vulnerable

A company
86%
(37/43)
Up to $1.0 \times 10^7$ errors

B company
83%
(45/54)
Up to $2.7 \times 10^6$ errors

C company
88%
(28/32)
Up to $3.3 \times 10^5$ errors

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, (Kim et al., ISCA 2014)
Recent DRAM Is More Vulnerable

All modules from 2012–2013 are vulnerable
One Can Take Over an Otherwise-Secure System

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors

Abstract. Memory isolation is a key property of a reliable and secure computing system — an access to one memory address should not have unintended side effects on data stored in other addresses. However, as DRAM process technology

Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

Project Zero

News and updates from the Project Zero team at Google

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn, 2015)
More Security Implications (I)

“We can gain unrestricted access to systems of website visitors.”

Not there yet, but ...

ROOT privileges for web apps!

Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript (DIMVA’16)

Source: https://lab.dsst.io/32c3-slides/7197.html
More Security Implications (II)

“Can gain control of a smart phone deterministically”

Hammer And Root

Millions of Androids

Source: https://fossbytes.com/drammer-rowhammer-attack-android-root-devices/
More Security Implications (VII)

USESIX Security 2019

Terminal Brain Damage: Exposing the Graceless Degradation in Deep Neural Networks Under Hardware Fault Attacks

Sanghyun Hong, Pietro Frigo†, Yiğitcan Kaya, Cristiano Giuffrida†, Tudor Dumitraş

University of Maryland, College Park
†Vrije Universiteit Amsterdam

A Single Bit-flip Can Cause Terminal Brain Damage to DNNs
One specific bit-flip in a DNN’s representation leads to an accuracy drop over 90%

Our research found that a specific bit-flip in a DNN’s bitwise representation can cause the accuracy loss up to 90%, and the DNN has 40-50% parameters, on average, that can lead to the accuracy drop over 10% when individually subjected to such single bitwise corruptions...
More Security Implications (VIII)

- USENIX Security 2020

DeepHammer: Depleting the Intelligence of Deep Neural Networks through Targeted Chain of Bit Flips

Fan Yao  
University of Central Florida  
fan.yao@ucf.edu

Adnan Siraj Rakin  
Arizona State University  
asrakin@asu.edu

Deliang Fan  

Degrade the **inference accuracy** to the level of **Random Guess**

Example: ResNet-20 for CIFAR-10, **10** output classes

Before attack, **Accuracy: 90.2%**  
After attack, **Accuracy: ~10% (1/10)**
Memory Scaling Issues Are Real

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,

"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"


[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data]
Memory Scaling Issues Are Real

- Onur Mutlu and Jeremie Kim, "RowHammer: A Retrospective"
  [Preliminary arXiv version]
  [Slides from COSADE 2019 (pptx)]
  [Slides from VLSI-SOC 2020 (pptx) (pdf)]
  [Talk Video (1 hr 15 minutes, with Q&A)]

RowHammer: A Retrospective

Onur Mutlu$^{§‡}$  Jeremie S. Kim$^{‡§}$
$^{§}$ETH Zürich  $^{‡}$Carnegie Mellon University
The Push from Circuits and Devices

Main Memory Needs

Intelligent Controllers
RowHammer in 2020 (I)

Jeremie S. Kim, Minesh Patel, A. Giray Yaglikci, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu,

"Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques"


[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (20 minutes)]
[Lightning Talk Video (3 minutes)]

Revisiting RowHammer: An Experimental Analysis of Modern DRAM Devices and Mitigation Techniques

Jeremie S. Kim$‡, Minesh Patel§, A. Giray Yağlıkçı§
Hasan Hassan§, Roknoddin Azizi§, Lois Orosa§, Onur Mutlu§‡

§ETH Zürich ‡Carnegie Mellon University
Key Takeaways from 1580 Chips

- Newer DRAM chips are more vulnerable to RowHammer

- There are chips today whose weakest cells fail after only 4800 hammers

- Chips of newer DRAM technology nodes can exhibit RowHammer bit flips 1) in more rows and 2) farther away from the victim row.

- Existing mitigation mechanisms are NOT effective
RowHammer in 2020 (II)

- Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi,

"TRRespass: Exploiting the Many Sides of Target Row Refresh"


[Slides (pptx) (pdf)]
[Lecture Slides (pptx) (pdf)]
[Talk Video (17 minutes)]
[Lecture Video (59 minutes)]
[Source Code]
[Web Article]

Best paper award.
Pwnie Award 2020 for Most Innovative Research. Pwnie Awards 2020

TRRespass: Exploiting the Many Sides of Target Row Refresh

Pietro Frigo*† Emanuele Vannacci*† Hasan Hassan§ Victor van der Veen¶
Onur Mutlu§ Cristiano Giuffrida* Herbert Bos*
Kaveh Razavi*

*Vrije Universiteit Amsterdam
§ETH Zürich
¶Qualcomm Technologies Inc.
RowHammer in 2020 (III)

- Lucian Cojocar, Jeremie Kim, Minesh Patel, Lillian Tsai, Stefan Saroiu, Alec Wolman, and Onur Mutlu,

"Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers"


[Slides (pptx) (pdf)]
[Talk Video (17 minutes)]

Are We Susceptible to Rowhammer?
An End-to-End Methodology for Cloud Providers

Lucian Cojocar, Jeremie Kim§‡, Minesh Patel§, Lillian Tsai‡,
Stefan Saroiu, Alec Wolman, and Onur Mutlu§‡
Microsoft Research, §ETH Zürich, ‡CMU, †MIT
BlockHammer Solution in 2021

- A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu,

"BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows"


[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Talk Video (22 minutes)]
[Short Talk Video (7 minutes)]

BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows

A. Giray Yaşlıkçı¹ Minesh Patel¹ Jeremie S. Kim¹ Roknoddin Azizi¹ Ataberk Olgun¹ Lois Orosa¹ Hasan Hassan¹ Jisung Park¹ Konstantinos Kanellopoulos¹ Taha Shahroodi¹ Saugata Ghose² Onur Mutlu¹

¹ETH Zürich ²University of Illinois at Urbana–Champaign
Two Upcoming RowHammer Papers at MICRO 2021

- Lois Orosa, Abdullah Giray Yaglıkçı, Haocong Luo, Ataberk Olgun, Jisung Park, Hasan Hassan, Minesh Patel, Jeremie S. Kim, Onur Mutlu,

"A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses"

MICRO 2021

A Deeper Look into RowHammer’s Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses

Lois Orosa*
ETH Zürich

A. Giray Yaglıkçı*
ETH Zürich

Haocong Luo
ETH Zürich

Ataberk Olgun
ETH Zürich, TOBB ETÜ

Jisung Park
ETH Zürich

Hasan Hassan
ETH Zürich

Minesh Patel
ETH Zürich

Jeremie S. Kim
ETH Zürich

Onur Mutlu
ETH Zürich
Two Upcoming RowHammer Papers at MICRO 2021

- Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, Onur Mutlu,

"Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications"

MICRO 2021

Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implications

Hasan Hassan† Yahya Can Tuğrul‡‡ Kaveh Razavi† Jeremie S. Kim† Onur Mutlu† Victor van der Veenσ

†ETH Zürich ‡‡TOBB University of Economics & Technology †Qualcomm Technologies Inc.
Detailed Lectures on RowHammer

- **Computer Architecture, Fall 2020, Lecture 4b**
  - RowHammer (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=K Dy632z23UE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=8](https://www.youtube.com/watch?v=K Dy632z23UE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=8)

- **Computer Architecture, Fall 2020, Lecture 5a**
  - RowHammer in 2020: TRRespass (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=pwRw7QqK_qA&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=9](https://www.youtube.com/watch?v=pwRw7QqK_qA&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=9)

- **Computer Architecture, Fall 2020, Lecture 5b**
  - RowHammer in 2020: Revisiting RowHammer (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=qR7XR-Eepcg&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=10](https://www.youtube.com/watch?v=qR7XR-Eepcg&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=10)

- **Computer Architecture, Fall 2020, Lecture 5c**
  - Secure and Reliable Memory (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=HvswnsfG3oQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=11](https://www.youtube.com/watch?v=HvswnsfG3oQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=11)

[https://www.youtube.com/onurmutlulectures](https://www.youtube.com/onurmutlulectures)
Onur Mutlu,
"The Story of RowHammer"
Keynote Talk at Secure Hardware, Architectures, and Operating Systems Workshop (SeHAS), held with HiPEAC 2021 Conference, Virtual, 19 January 2021.
[Slides (pptx) (pdf)]
[Talk Video (1 hr 15 minutes, with Q&A)]
How Reliable/Secure/Safe is This Bridge?

Source: http://www.technologystudent.com/struct1/tacom1.png
Collapse of the “Galloping Gertie” (1940)

Source: AP
http://www.wsdot.wa.gov/tnbhistory/connections/connections3.htm
Another Example (1994)

Source: By 최광모 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=35197984
Yet Another Example (2007)

A More Recent Example (2018)

How Safe & Secure Is This Platform?

Security is about preventing unforeseen consequences
How Safe & Secure Is This Platform?

Source: https://taxistartup.com/wp-content/uploads/2015/03/UK-Self-Driving-Cars.jpg
Fundamentally Secure, Reliable, Safe Computing Architectures
Solution Direction: Principled Designs

Design fundamentally secure computing architectures

Predict and prevent safety & security issues
The Push from Circuits and Devices

Computing Systems Need Intelligent Memories
In-Field Patch-ability (Intelligent Memory) Can Avoid Many Failures
Why In-Memory Computation Today?

- **Push from Technology**
  - DRAM Scaling at jeopardy
    - Controllers close to DRAM
    - Industry open to new memory architectures

- **Pull from Systems and Applications**
  - Data access is a major system and application bottleneck
  - Systems are energy limited
  - Data movement much more energy-hungry than computation
Three Key Systems & Application 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
Do We Want This?

Source: V. Milutinovic
Or This?

Source: V. Milutinovic
Challenge and Opportunity for Future

High Performance, Energy Efficient, Sustainable
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.

Today’s Computing Systems

- Are overwhelmingly processor centric
- All data processed in the processor ⇒ at great system cost
- Processor is heavily optimized and is considered the master
- Data storage units are dumb and are largely unoptimized (except for some that are on the processor die)
Yet …

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

I expect that over the coming decade memory subsystem design will be the only important design issue for microprocessors.

Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors

Onur Mutlu §  Jared Stark †  Chris Wilkerson ‡  Yale N. Patt §

§ECE Department
The University of Texas at Austin
{onur,patt}@ece.utexas.edu

†Microprocessor Research
Intel Labs
jared.w.stark@intel.com

‡Desktop Platforms Group
Intel Corporation
chris.wilkerson@intel.com


One of the 15 computer arch. papers of 2003 selected as Top Picks by IEEE Micro.
HPCA Test of Time Award (awarded in 2021).
The Performance Perspective (Today)

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

The Performance Perspective (Today)

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

![Box plot showing cache-bound cycles](image)

**Figure 11: Half of cycles are spent stalled on caches.**

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
Perils of Processor-Centric Design

Most of the system is dedicated to storing and moving data.

Yet, system is still bottlenecked by memory.
The Energy Perspective

Communication Dominates Arithmetic

Dally, HiPEAC 2015
A memory access consumes $\sim 100$-$1000X$ the energy of a complex addition.
Data Movement vs. Computation Energy

- **Data movement** is a major system energy bottleneck
  - Comprises **41%** of mobile system energy during web browsing [2]
  - Costs ~**115** times as much energy as an ADD operation [1, 2]

[1]: Reducing data Movement Energy via Online Data Clustering and Encoding (MICRO’16)
[2]: Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms (IISWC’14)
Energy Waste in Mobile Devices

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu,
  "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"

62.7% of the total system energy is spent on data movement

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand¹
Rachata Ausavarungnirun¹
Aki Kuusela³
Allan Knies³
Saugata Ghose¹
Eric Shiu³
Rahul Thakur³
Parthasarathy Ranganathan³
Youngsok Kim²
Daehyun Kim⁴,³
Onur Mutlu⁵,¹

SAFARI
We Do Not Want to Move Data!

Communication Dominates Arithmetic

Dally, HiPEAC 2015

A memory access consumes $\sim 100-1000\times$ the energy of a complex addition
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**
Goal: Processing Inside Memory

Many questions ... How do we design the:
- compute-capable memory & controllers?
- processor chip and in-memory units?
- software and hardware interfaces?
- system software, compilers, languages?
- algorithms and theoretical foundations?
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

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

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 all 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 computing systems and applications [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-
Processing in Memory: Two Approaches

1. Processing using Memory
2. Processing near Memory
Approach 1: Processing Using Memory

- Take advantage of operational principles of memory to perform bulk data movement and computation in memory
  - Can exploit internal connectivity 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)
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 → 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)

Data Bus

Transfer row

Transfer row

8 bits

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)]
Lecture on RowClone & Processing using DRAM

Mindset: Memory as an Accelerator

- CPU core
- CPU core
- mini-CPU core
- GPU (throughput) core
- GPU (throughput) core
- GPU (throughput) core
- Memory

Memory similar to a “conventional” accelerator

https://www.youtube.com/watch?v=n6Pwg1qax_E&list=PL5Q2soXY2Zi_7UBNmC9B8Yr5JSwTG9yH4&index=4
RowClone Extensions and Follow-Up Work

- Can we do faster inter-subarray copy?
  - Yes, see LISA [Chang et al., HPCA 2016]

- Can we enable data movement at smaller granularities within a bank?
  - Yes, see FIGARO [Wang et al., MICRO 2020]

- Can we do better inter-bank copy?
  - Yes, see Network-on-Memory [CAL 2020]

- Can similar ideas and DRAM properties be used to perform computation on data?
  - Yes, see Ambit [Seshadri et al., CAL 2015, MICRO 2017]

Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM

Kevin K. Chang†, Prashant J. Nair*, Donghyuk Lee†, Saugata Ghose†, Moinuddin K. Qureshi*, and Onur Mutlu†

†Carnegie Mellon University  *Georgia Institute of Technology
FIGARO: Fine-Grained In-DRAM Copy

- Yaohua Wang, Lois Orosa, Xiangjun Peng, Yang Guo, Saugata Ghose, Minesh Patel, Jeremie S. Kim, Juan Gómez Luna, Mohammad Sadrosadati, Nika Mansouri Ghiasi, and Onur Mutlu,

"FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching"


FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching

Yaohua Wang*  Lois Orosa†  Xiangjun Peng‡*  Yang Guo*  Saugata Ghose○†  Minesh Patel†
Jeremie S. Kim†  Juan Gómez Luna†  Mohammad Sadrosadati‡  Nika Mansouri Ghiasi†  Onur Mutlu‡‡

*National University of Defense Technology  †ETH Zürich  ○Chinese University of Hong Kong
§University of Illinois at Urbana–Champaign  ‡Carnegie Mellon University  ¶Institute of Research in Fundamental Sciences
Network-On-Memory: Fast Inter-Bank Copy

- Seyyed Hossein SeyyedAghaei Rezaei, Mehdi Modarressi, Rachata Ausavarungnirun, Mohammad Sadrosadati, Onur Mutlu, and Masoud Daneshtalab,

"NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories"

Mindset: Memory as an Accelerator

Memory similar to a “conventional” accelerator
(Truly) In-Memory Computation

- We can support in-DRAM 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

- New memory technologies enable even more opportunities
  - Memristors, resistive RAM, phase change mem, STT-MRAM, ...
  - Can operate on data with minimal movement
In-DRAM AND/OR: Triple Row Activation

Final State

$$AB + BC + AC$$

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

Seshadri+, "Fast Bulk Bitwise AND and OR in DRAM", IEEE CAL 2015.
Bulk Bitwise Operations in Workloads

- Bitmap indices (database indexing)
- Set operations
- Encryption algorithms
- BitWeaving (database queries)
- BitFunnel (web search)
- DNA sequence mapping

[1] Li and Patel, BitWeaving, SIGMOD 2013
Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

More on Ambit

- Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,

"Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology"
Proceedings of the 50th International Symposium on Microarchitecture (MICRO), Boston, MA, USA, October 2017.
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]

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
In-DRAM Bulk Bitwise Execution


In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
SIMDRAM Framework


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

*Nastaran Hajinazar\(^1,2\)  *Geraldo F. Oliveira\(^1\)  Sven Gregorio\(^1\)  João Dinis Ferreira\(^1\)
Nika Mansouri Ghiasi\(^1\)  Minesh Patel\(^1\)  Mohammed Alser\(^1\)  Saugata Ghose\(^3\)
Juan Gómez-Luna\(^1\)  Onur Mutlu\(^1\)

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

• **SIMDRAM**: An end-to-end processing-using-DRAM framework that provides the **programming interface**, the **ISA**, and the **hardware support** for:

  - **Efficiently** computing **complex** operations in DRAM
  - Providing the ability to implement **arbitrary** operations as required
  - Using an **in-DRAM massively-parallel SIMD substrate** that requires **minimal** changes to DRAM architecture
SIMDRAM Framework: Overview

**Step 1: Generate MAJ logic**
- Desired operation
  - AND/OR/NOT logic

**Step 2: Generate sequence of DRAM commands**
- MAJ logic
  - MAJ/NOT logic
  - New SIMDRAM instruction
  - bbop_new

**Step 3: Execution according to µProgram**
- User Input
  - SIMDRAM-enabled application
    - foo () {
      bbop_new
    }

- Control Unit
  - µProgram

- Memory Controller
  - µProgram

- SIMDRAM Output
  - Main memory
  - New SIMDRAM µProgram
  - ISA
SIMDRAM Key Results

Evaluated on:
- 16 complex in-DRAM operations
- 7 commonly-used real-world applications

SIMDRAM provides:

• 88× and 5.8× the throughput of a CPU and a high-end GPU, respectively, over 16 operations

• 257× and 31× the energy efficiency of a CPU and a high-end GPU, respectively, over 16 operations

• 21× and 2.1× the performance of a CPU and a high-end GPU, over seven real-world applications
SIMDRAM Conclusion

• **SIMDRAM:**
  - Enables efficient computation of a flexible set and wide range of operations in a PuM massively parallel SIMD substrate
  - Provides the hardware, programming, and ISA support, to:
    - Address key system integration challenges
    - Allow programmers to define and employ new operations without hardware changes

**SIMDRAM is a promising PuM framework**
- Can ease the adoption of processing-using-DRAM architectures
- Improves the performance and efficiency of processing-using-memory architectures
SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

Nastaran Hajinazar*
Sven Gregorio
Minesh Patel
Juan Gómez–Luna

Geraldo F. Oliveira*
Joao Ferreira
Mohammed Alser
Onur Mutlu

Safari

SFU Simon Fraser University

ETH Zürich

University of Illinois
Urbana–Champaign
In-DRAM Physical Unclonable Functions


The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

Jeremie S. Kim†§ Minesh Patel§ Hasan Hassan§ Onur Mutlu§†

†Carnegie Mellon University §ETH Zürich
In-DRAM True Random Number Generation

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, and Onur Mutlu, "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput"


[Slides (pptx) (pdf)]
[Full Talk Video (21 minutes)]
[Full Talk Lecture Video (27 minutes)]

Top Picks Honorable Mention by IEEE Micro.

D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

Jeremie S. Kim‡$ Minesh Patel$ Hasan Hassan$ Lois Orosa$ Onur Mutlu‡$
‡Carnegie Mellon University $ETH Zürich

SAFARI
In-DRAM True Random Number Generation

- Ataberk Olgun, Minesh Patel, A. Giray Yaglikci, Haocong Luo, Jeremie S. Kim, F. Nisa Bostanci, Nandita Vijaykumar, Oguz Ergin, and Onur Mutlu,

"QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips"


[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Talk Video (25 minutes)]
[Safari Live Seminar Video (1 hr 26 mins)]

**QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips**

- Ataberk Olgun$^{§†}$
- Minesh Patel$^§$
- A. Giray Yağlıkçı$^§$
- Haocong Luo$^§$
- Jeremie S. Kim$^§$
- F. Nisa Bostanci$^{§†}$
- Nandita Vijaykumar$^{§⊙}$
- Oğuz Ergin$^†$
- Onur Mutlu$^§$

$^{§}$ETH Zürich  $^{†}$TOBB University of Economics and Technology  $^{⊙}$University of Toronto

SAFARI
Processing in Memory: Two Approaches

1. Processing using Memory
2. Processing near Memory
Another Example: In-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
  - Speedup
    - 32 Cores
    - 128...
    - +42%
Key Bottlenecks in Graph Processing

```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
Opportunity: 3D-Stacked Logic+Memory

Other “True 3D” technologies under development
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)

Crossbar Network

Memory

Logic

In-Order Core

DRAM Controller

PF Buffer

MTP

Message Queue

NI

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

Host Processor
Memory-Mapped Accelerator Interface
(Noncacheable, Physically Addressed)

Memory

Logic

Crossbar Network

In-Order Core

Communications via Remote Function Calls

Message Queue
Tesseract System for Graph Processing

Logic

Crossbar Network

Message Queue

Host Processor

Memory-Mapped Accelerator Interface
Noncacheable, Physically Addressed

Memory

Prefetching

LP
PF Buffer
MTP

DRAM Controller

SAFARI
Evaluated Systems

**DDR3-OoO**
- 8 OoO 4GHz
- 8 OoO 4GHz
- 102.4GB/s

**HMC-OoO**
- 8 OoO 4GHz
- 8 OoO 4GHz
- 640GB/s

**HMC-MC**
- 128 In-Order 2GHz
- 128 In-Order 2GHz
- 640GB/s

**Tesseract**
- 32 Tesseract Cores
- 8TB/s

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

>13X Performance Improvement

On five graph processing algorithms

<table>
<thead>
<tr>
<th>Speedup</th>
<th>DDR3-OoO</th>
<th>HMC-OoO</th>
<th>HMC-MC</th>
<th>Tesseract</th>
<th>Tesseract-LP</th>
<th>Tesseract-LP-MTP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>+56%</td>
<td>+25%</td>
<td>9.0x</td>
<td>11.6x</td>
<td>13.8x</td>
</tr>
</tbody>
</table>

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

Memory Bandwidth Consumption

- DDR3-OoO: 80GB/s
- HMC-OoO: 190GB/s
- HMC-MC: 243GB/s
- Tesseract: 1.3TB/s
- Tesseract-LP: 2.2TB/s
- Tesseract-LP-MTP: 2.9TB/s

SAFARI
Tesseract Graph Processing System Energy

> 8X Energy Reduction

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)]

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
Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand
Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, Onur Mutlu
Consumer Devices

Consumer devices are everywhere!

Energy consumption is a first-class concern in consumer devices.
Popular Consumer Workloads

**Chrome**
Google’s web browser

**TensorFlow Mobile**
Google’s machine learning framework

**VP9**
YouTube Video Playback
Google’s video codec

**VP9**
YouTube Video Capture
Google’s video codec
Energy Cost of Data Movement

1st key observation: 62.7% of the total system energy is spent on data movement.

Potential solution: move computation close to data.

Challenge: limited area and energy budget.
Using PIM to Reduce Data Movement

2\textsuperscript{nd} key observation: a significant fraction of the data movement often comes from simple functions

We can design lightweight logic to implement these simple functions in memory

Offloading to PIM logic reduces energy and improves performance, on average, by 2.3X and 2.2X
Workload Analysis

Chrome
Google’s web browser

TensorFlow Mobile
Google’s machine learning framework

Video Playback
Google’s video codec

Video Capture
Google’s video codec
57.3% of the inference energy is spent on data movement.

54.4% of the data movement energy comes from packing/unpacking and quantization.
More on PIM for Mobile Devices

- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu,

"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)] [Lightning Talk Video (2 minutes)] [Full Talk Video (21 minutes)]

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand¹
Rachata Ausavarungnirun¹
Aki Kuusela³
Saugata Ghose¹
Eric Shiu³
Youngsok Kim²
Rahul Thakur³
Parthasarathy Ranganathan³
Daehyun Kim⁴,³
Onur Mutlu⁵,¹

SAFARI
Truly Distributed GPU Processing with PIM

3D-stacked memory (memory stack)

SM (Streaming Multiprocessor)

Main GPU

Logic layer

Crossbar switch

Vault Ctrl

Vault Ctrl
Accelerating GPU Execution with PIM (I)

Kevin Hsieh, Eiman Ebrahimi, Gwangsun Kim, Niladrish Chatterjee, Mike O'Connor, Nandita Vijaykumar, Onur Mutlu, and Stephen W. Keckler,
"Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems"
[Slides (pptx) (pdf)]
[Lightning Session Slides (pptx) (pdf)]
Accelerating GPU Execution with PIM (II)


Proceedings of the 25th International Conference on Parallel Architectures and Compilation Techniques (PACT), Haifa, Israel, September 2016.

Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities

Ashutosh Pattnaik\(^1\) Xulong Tang\(^1\) Adwait Jog\(^2\) Onur Kayiran\(^3\)
Asit K. Mishra\(^4\) Mahmut T. Kandemir\(^1\) Onur Mutlu\(^5,6\) Chita R. Das\(^1\)

\(^1\)Pennsylvania State University \(^2\)College of William and Mary
\(^3\)Advanced Micro Devices, Inc. \(^4\)Intel Labs \(^5\)ETH Zürich \(^6\)Carnegie Mellon University
Accelerating Linked Data Structures

Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu,
"Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"
Proceedings of the 34th IEEE International Conference on Computer Design (ICCD), Phoenix, AZ, USA, October 2016.

Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh† Samira Khan‡ Nandita Vijaykumar†
Kevin K. Chang† Amirali Boroumand† Saugata Ghose† Onur Mutlu§†
†Carnegie Mellon University ‡University of Virginia §ETH Zürich
Accelerating Dependent Cache Misses

- Milad Hashemi, Khubaib, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt, "Accelerating Dependent Cache Misses with an Enhanced Memory Controller"


[Slides (pptx) (pdf)]
[Lightning Session Slides (pptx) (pdf)]

Accelerating Dependent Cache Misses with an Enhanced Memory Controller

Milad Hashemi*, Khubaib†, Eiman Ebrahimi‡, Onur Mutlu§, Yale N. Patt*

*The University of Texas at Austin †Apple ‡NVIDIA §ETH Zürich & Carnegie Mellon University
Accelerating Runahead Execution

- Milad Hashemi, Onur Mutlu, and Yale N. Patt, "Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads". Proceedings of the 49th International Symposium on Microarchitecture (MICRO), Taipei, Taiwan, October 2016. [Slides (pptx) (pdf)] [Lightning Session Slides (pdf)] [Poster (pptx) (pdf)]

Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads

Milad Hashemi*, Onur Mutlu§, Yale N. Patt*

*The University of Texas at Austin  §ETH Zürich
Accelerating Climate Modeling

- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,
  "NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"
  Proceedings of the 30th International Conference on Field-Programmable Logic and Applications (FPL), Gothenburg, Sweden, September 2020.
  [Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (23 minutes)]
  Nominated for the Stamatis Vassiliadis Memorial Award.

NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

Gagandeep Singh$^{a,b,c}$ Dionysios Diamantopoulos$^c$ Christoph Hagleitner$^c$ Juan Gómez-Luna$^b$
Sander Stuijk$^a$ Onur Mutlu$^b$ Henk Corporaal$^a$

$^a$Eindhoven University of Technology
$^b$ETH Zürich
$^c$IBM Research Europe, Zurich
Accelerating Approximate String Matching

- Damla Senol Cali, Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,

"GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"


[Lighting Talk Video (1.5 minutes)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (18 minutes)]
[Slides (pptx) (pdf)]

GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali†M, Gurpreet S. Kalsi†M, Zülal Bingöl√, Can Firtina†, Lavanya Subramanian†, Jeremie S. Kim††, Rachata Ausavarungnirun©, Mohammed Alser•, Juan Gomez-Luna•, Amirali Boroumand†, Anant Nori†M, Allison Scibisz†, Sreenivas Subramoney‡, Can Alkan√, Saugata Ghose‡†, Onur Mutlu‡†V

†Carnegie Mellon University   ‡Processor Architecture Research Lab, Intel Labs   √Bilkent University   •ETH Zürich
©Facebook   ©King Mongkut’s University of Technology North Bangkok   *University of Illinois at Urbana–Champaign

SAFARI
Accelerating Time Series Analysis

- Ivan Fernandez, Ricardo Quislant, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, Eladio Gutiérrez, Oscar Plata, and Onur Mutlu,

"NATSA: A Near-Data Processing Accelerator for Time Series Analysis"
[Slides (pptx) (pdf)]
[Talk Video (10 minutes)]
[Source Code]

NATSA: A Near-Data Processing Accelerator for Time Series Analysis

Ivan Fernandez§  Ricardo Quislant§  Christina Giannoula†  Mohammed Alser‡
Juan Gómez-Luna‡  Eladio Gutiérrez§  Oscar Plata§  Onur Mutlu‡

§University of Malaga  †National Technical University of Athens  ‡ETH Zürich
Accelerating Neural Network Inference

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,

"Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (PACT), Virtual, September 2021.
[Slides (pptx) (pdf)]
[Talk Video (14 minutes)]

Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand†‡
Geraldo F. Oliveira* Saugata Ghose‡‡ Xiaoyu Ma§ Berkin Akin§ Eric Shiu§ Ravi Narayanaswami§ Onur Mutlu*†

†Carnegie Mellon Univ.  ‡Stanford Univ.  ‡‡Univ. of Illinois Urbana-Champaign  §Google  *ETH Zürich
Executive Summary

**Context:** We extensively analyze a state-of-the-art edge ML accelerator (Google Edge TPU) using 24 Google edge models

- Wide range of models (CNNs, LSTMs, Transducers, RCNNs)

**Problem:** The Edge TPU accelerator suffers from three challenges:

- It operates significantly below its peak throughput
- It operates significantly below its theoretical energy efficiency
- It inefficiently handles memory accesses

**Key Insight:** These shortcomings arise from the monolithic design of the Edge TPU accelerator

- The Edge TPU accelerator design does not account for layer heterogeneity

**Key Mechanism:** A new framework called Mensa

- Mensa consists of heterogeneous accelerators whose dataflow and hardware are specialized for specific families of layers

**Key Results:** We design a version of Mensa for Google edge ML models

- Mensa improves performance and energy by 3.0X and 3.1X
- Mensa reduces cost and improves area efficiency
FPGA-based Processing Near Memory


FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh♦ Mohammed Alser♦ Damla Senol Cali♦
Dionysios Diamantopoulos∇ Juan Gómez-Luna♦
Henk Corporaal* Onur Mutlu♦∇

♦ETH Zürich ♦Carnegie Mellon University
*Eindhoven University of Technology ∇IBM Research Europe
We Need to Revisit the Entire Stack

We can get there step by step
PEI: Simple Processing in Memory

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


SAFARI

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

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 \textit{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) \textit{processing using memory} by exploiting analog operational properties of DRAM chips to perform massively-parallel operations in memory, with low-cost changes, (2) \textit{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.

\textit{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

SAFARI
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 computing 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, 7, 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 unimportant 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-
A Workload and Programming Ease Driven Perspective of Processing-in-Memory

Saugata Ghose† Amirali Boroumand† Jeremie S. Kim†§ Juan Gómez-Luna§ Onur Mutlu§†
†Carnegie Mellon University §ETH Zürich

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu, "Processing-in-Memory: A Workload-Driven Perspective"
[Preliminary arXiv version]

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

UPMEM Memory Modules

- E19: 8 chips DIMM (1 rank). DPUs @ 267 MHz
- P21: 16 chips DIMM (2 ranks). DPUs @ 350 MHz
Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

Juan Gómez-Luna, ETH Zürich, Switzerland
IZZAT EL HAJI, American University of Beirut, Lebanon
IVAN FERNANDEZ, ETH Zürich and University of Malaga, Spain
CHRISTINA CIANNOULA, 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 manifestly 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 work evaluates these architectures in simulation or, at best, with simplified hardware prototypes. In contrast, the UPNEM company has designed and manufactured the first publicly available real-world PIM architecture. The UPNEM PIM architecture combines traditional DRAM as 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 UPNEM-based PIM system using microbenchmarks to assess various architecture limits such as compute throughput and memory bandwidth, yielding new insights. Second, we present PIM (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 PIM benchmarks on the UPNEM 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 UPNEM-based PIM systems with 140 and 2,560 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.
More on the UPMEM PIM System

https://www.youtube.com/watch?v=Sscy1Wrr22A&list=PL5Q2soXY2Zi9xidy1gBxUz7xRPS-wisBN&index=26
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.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Benchmark</th>
<th>Short name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dense linear algebra</td>
<td>Vector Addition</td>
<td>VA</td>
</tr>
<tr>
<td></td>
<td>Matrix-Vector Multiply</td>
<td>GEMV</td>
</tr>
<tr>
<td>Sparse linear algebra</td>
<td>Sparse Matrix-Vector Multiply</td>
<td>SpMV</td>
</tr>
<tr>
<td>Databases</td>
<td>Select</td>
<td>SEL</td>
</tr>
<tr>
<td></td>
<td>Unique</td>
<td>UNI</td>
</tr>
<tr>
<td>Data analytics</td>
<td>Binary Search</td>
<td>BS</td>
</tr>
<tr>
<td></td>
<td>Time Series Analysis</td>
<td>TS</td>
</tr>
<tr>
<td>Graph processing</td>
<td>Breadth-First Search</td>
<td>BFS</td>
</tr>
<tr>
<td>Neural networks</td>
<td>Multilayer Perceptron</td>
<td>MLP</td>
</tr>
<tr>
<td>Bioinformatics</td>
<td>Needleman-Wunsch</td>
<td>NW</td>
</tr>
<tr>
<td>Image processing</td>
<td>Image histogram (short)</td>
<td>HST-S</td>
</tr>
<tr>
<td></td>
<td>Image histogram (large)</td>
<td>HST-L</td>
</tr>
<tr>
<td>Parallel primitives</td>
<td>Reduction</td>
<td>RED</td>
</tr>
<tr>
<td></td>
<td>Prefix sum (scan-scan-add)</td>
<td>SCAN-SSA</td>
</tr>
<tr>
<td></td>
<td>Prefix sum (reduce-scan-scan)</td>
<td>SCAN-RSS</td>
</tr>
<tr>
<td></td>
<td>Matrix transposition</td>
<td>TRNS</td>
</tr>
</tbody>
</table>
PrIM Benchmarks are Open Source

- All microbenchmarks, benchmarks, and scripts
- [https://github.com/CMU-SAFARI/prim-benchmarks](https://github.com/CMU-SAFARI/prim-benchmarks)

**PrIM (Processing-In-Memory Benchmarks)**

PrIM is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publicly-available real-world processing-in-memory (PIM) architecture, the UPMEM 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.

PrIM provides a common set of workloads to evaluate the UPMEM PIM architecture with and can be useful for programming, architecture and system researchers all alike to improve multiple aspects of future PIM hardware and software. The workloads have different characteristics, exhibiting heterogeneity in their memory access patterns, operations and data types, and communication patterns. This repository also contains baseline CPU and GPU implementations of PrIM benchmarks for comparison purposes.

PrIM also includes a set of microbenchmarks can be used to assess various architecture limits such as compute throughput and memory bandwidth.
Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization

Juan Gómez-Luna$^1$  Izzat El Hajj$^2$  Ivan Fernandez$^{1,3}$  Christina Giannoula$^{1,4}$
Geraldo F. Oliveira$^1$  Onur Mutlu$^1$

$^1$ETH Zürich  $^2$American University of Beirut  $^3$University of Malaga  $^4$National Technical University of Athens

https://github.com/CMU-SAFARI/prim-benchmarks
Understanding a Modern PIM Architecture

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

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
Inter-DPU Communication

- There is *no direct communication channel between DPUs*

Inter-DPU communication takes place via the host CPU using CPU-DPU and DPU-CPU transfers

Example communication patterns:
- Merging of partial results to obtain the final result
- Only DPU-CPU transfers
- Redistribution of intermediate results for further computation
- DPU-CPU transfers and CPU-DPU transfers

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

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

Dr. Juan Gómez-Luna, SAFARI Research Group, D-ITET, ETH Zurich

https://www.youtube.com/watch?v=D8Hjy2iU9j4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9
Data Movement in Computing Systems

- Data movement dominates performance and is a major system energy bottleneck
- Total system energy: data movement accounts for
  - 62% in consumer applications
  - 40% in scientific applications
  - 35% in mobile applications

---

Understanding a Modern Processing-in-Memory Arch: Benchmarking & Experimental Characterization; 21m

3,482 views • Premiered Jul 25, 2021

https://www.youtube.com/watch?v=Pp9jSU2b9oM&list=PL5Q2soXY2Zi8_VVChACnON4sfh2bJ5IRD&index=159
FPGA-based Processing Near Memory

Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu,
"FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"

FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh\textsuperscript{\circled{1}}  Mohammed Alser\textsuperscript{\circled{1}}  Damla Senol Cali\textsuperscript{\circled{2}}
Dionysios Diamantopoulos\textsuperscript{\textdagger}  Juan Gómez-Luna\textsuperscript{\circled{1}}
Henk Corporaal\textsuperscript{*}  Onur Mutlu\textsuperscript{\textdagger\textcircled{1}}

\textsuperscript{\circled{1}}ETH Zürich  \textsuperscript{\textdagger}Carnegie Mellon University
\textsuperscript{*}Eindhoven University of Technology  \textsuperscript{\textdagger}IBM Research Europe
DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, Institute for Research in Fundamental Sciences (IPM), Iran & ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Data movement between the CPU and main memory is a first-order obstacle against improving performance, scalability, and energy efficiency in modern systems. Computer systems employ a range of techniques to reduce overheads tied to data movement, spanning from traditional mechanisms (e.g., deep multi-level cache hierarchies, aggressive hardware prefetchers) to emerging techniques such as Near-Data Processing (NDP), where some computation is moved close to memory. Prior NDP works investigate the root causes of data movement bottlenecks using different profiling methodologies and tools. However, there is still a lack of understanding about the key metrics that can identify different data movement bottlenecks and their relation to traditional and emerging data movement mitigation mechanisms. Our goal is to methodically identify potential sources of data movement over a broad set of applications and to comprehensively compare traditional compute-centric data movement mitigation techniques (e.g., caching and prefetching) to more memory-centric techniques (e.g., NDP), thereby developing a rigorous understanding of the best techniques to mitigate each source of data movement.

With this goal in mind, we perform the first large-scale characterization of a wide variety of applications, across a wide range of application domains, to identify fundamental program properties that lead to data movement to/from main memory. We develop the first systematic methodology to classify applications based on the sources contributing to data movement bottlenecks. From our large-scale characterization of 77K functions across 345 applications, we select 144 functions to form the first open-source benchmark suite (DAMOV) for main memory data movement studies. We select a diverse range of functions that (1) represent different types of data movement bottlenecks, and (2) come from a wide range of application domains. Using NDP as a case study, we identify new insights about the different data movement bottlenecks and use these insights to determine the most suitable data movement mitigation mechanism for a particular application. We open-source DAMOV and the complete source code for our new characterization methodology at https://github.com/CMU-SAFARI/DAMOV.
When to Employ Near-Data Processing?

Near-Data Processing

- Mobile consumer workloads
  - (GoogleWL²)
- Neural networks
  - (GoogleWL²)
- DNA sequence mapping
  - (GenASM³; GRIM-Filter⁴)
- Graph processing
  - (Tesseract¹)
- Databases
  - (Polynesia⁵)
- Time series analysis
  - (NATSA⁶)

Step 1: Application Profiling

- We analyze **345 applications** from distinct domains:
  - Graph Processing
  - Deep Neural Networks
  - Physics
  - High-Performance Computing
  - Genomics
  - Machine Learning
  - Databases
  - Data Reorganization
  - Image Processing
  - Map-Reduce
  - Benchmarking
  - Linear Algebra
  ...

SAFARI
Step 3: Memory Bottleneck Analysis

Six classes of data movement bottlenecks:

each class $\leftrightarrow$ data movement mitigation mechanism
DAMOV is Open Source

- We open-source our benchmark suite and our toolchain
DAMOV is Open Source

- We open-source our **benchmark suite** and our **toolchain**

---

**DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks**

DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing.

The DAMOV benchmark suite is the first open-source benchmark suite for main memory data movement-related studies, based on our systematic characterization methodology. This suite consists of 144 functions representing different sources of data movement bottlenecks and can be used as a baseline benchmark set for future data-movement mitigation research. The applications in the DAMOV benchmark suite belong to popular benchmark suites, including **BWA**, **Chai**, **Darknet**, **GASE**, **Hardware Effects**, **Hashjoin**, **HPCC**, **HPCG**, **Ligra**, **PARSEC**, **Parboil**, **PolyBench**, **Phoenix**, **Rodinia**, **SPLASH-2**, and **STREAM**.

---

Get DAMOV at: [https://github.com/CMU-SAFARI/DAMOV](https://github.com/CMU-SAFARI/DAMOV)
More on DAMOV Analysis Methodology & Workloads

**Goal:** identify the specific sources of data movement bottlenecks

- **Scalability Analysis:**
  1. 1, 4, 16, 64, and 256 out-of-order/in-order host and NDP CPU cores
  2. 3D-stacked memory as main memory

SAFARI Live Seminar: DAMOV: A New Methodology & Benchmark Suite for Data Movement Bottlenecks

[View Video](https://www.youtube.com/watch?v=GWideVyo0nM&list=PL5Q2soXY2Zi_TOTAYm--dYByNPL7JhwR9&index=3)
More on DAMOV

- Geraldo F. Oliveira, Juan Gomez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan Fernandez, Mohammad Sadrosadati, and Onur Mutlu, "DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks"
  Preprint in arXiv, 8 May 2021.
  [arXiv preprint]
  [DAMOV Suite and Simulator Source Code]
  [SAFARI Live Seminar Video (2 hrs 40 mins)]
  [Short Talk Video (21 minutes)]

DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÔMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland
Samsung Develops Industry’s First High Bandwidth Memory with AI Processing Power

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 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications


*Samsung Electronics, Hwasung, Korea  
Samsung Electronics, San Jose, CA  
Samsung Electronics, Suwon, Korea
Programmable Computing Unit

- Configuration of PCU block
  - Interface unit to control data flow
  - Execution unit to perform operations
  - Register group
    - 32 entries of CRF for instruction memory
    - 16 GRF for weight and accumulation
    - 16 SRF to store constants for MAC operations

---

**Samsung Function-in-Memory DRAM (2021)**

---

ISSCC 2021 / SESSION 25 / DRAM / 25.4

A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

- Young-Choon Kwon, Suk Ho Lee, Jeonhoon Lee, Sang-Hyuk Kim,
  Je Min Ryu, Jong-Pil Son, Soon-Bong Lee, Han-Soo Lee,
  Seo Young Kim, Youngmin Choi, Jin Suk Kim, Jeongsoo Choi,
  Heesung Shin, Jin Kim, Bengsang Park, HyoungMin Kim,
  Myeong Jun Song, Min Cho, Dooho Kim, Seo-Young Kim, Eun-Bong Kim,
  David Wang, Shikung Kang, Youseop Yoo, Seungwoo Son, Jeonho Song,
  Jeonhoon Yoon, Kyunghoon Seo, Nam Sung Kim

*Samsung Electronics, Hwasung, Korea
*Samsung Electronics, San Jose, CA
*Samsung Electronics, Suwon, Korea
## Available instruction list for FIM operation

<table>
<thead>
<tr>
<th>Type</th>
<th>CMD</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating Point</td>
<td>ADD</td>
<td>FP16 addition</td>
</tr>
<tr>
<td></td>
<td>MUL</td>
<td>FP16 multiplication</td>
</tr>
<tr>
<td></td>
<td>MAC</td>
<td>FP16 multiply-accumulate</td>
</tr>
<tr>
<td></td>
<td>MAD</td>
<td>FP16 multiply and add</td>
</tr>
<tr>
<td>Data Path</td>
<td>MOVE</td>
<td>Load or store data</td>
</tr>
<tr>
<td></td>
<td>FILL</td>
<td>Copy data from bank to GRFs</td>
</tr>
<tr>
<td>Control Path</td>
<td>NOP</td>
<td>Do nothing</td>
</tr>
<tr>
<td></td>
<td>JUMP</td>
<td>Jump instruction</td>
</tr>
<tr>
<td></td>
<td>EXIT</td>
<td>Exit instruction</td>
</tr>
</tbody>
</table>
Chip Implementation

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

[Image: TSV & Peri Control Block]

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

- DDR5-PIM
  - DLRM recommendation system
Detailed Lectures on PIM (I)

- Computer Architecture, Fall 2020, Lecture 6
  - Computation in Memory (ETH Zürich, Fall 2020)
    - https://www.youtube.com/watch?v=oGcZAGwfEUE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=12

- Computer Architecture, Fall 2020, Lecture 7
  - Near-Data Processing (ETH Zürich, Fall 2020)
    - https://www.youtube.com/watch?v=j2GIigqn1Qw&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=13

- Computer Architecture, Fall 2020, Lecture 11a
  - Memory Controllers (ETH Zürich, Fall 2020)
    - https://www.youtube.com/watch?v=TeG773OgiMQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=20

- Computer Architecture, Fall 2020, Lecture 12d
  - Real Processing-in-DRAM with UPMEM (ETH Zürich, Fall 2020)
    - https://www.youtube.com/watch?v=Sscy1Wrr22A&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=25
Detailed Lectures on PIM (II)

- Computer Architecture, Fall 2020, Lecture 15
  - **Emerging Memory Technologies** (ETH Zürich, Fall 2020)
  - [YouTube Link](https://www.youtube.com/watch?v=AlE1rD9G_YU&list=PL5Q2soXY2Zi9idayyIgBxUz7xRPS-wisBN&index=28)

- Computer Architecture, Fall 2020, Lecture 16a
  - **Opportunities & Challenges of Emerging Memory Technologies** (ETH Zürich, Fall 2020)
  - [YouTube Link](https://www.youtube.com/watch?v=pmLszWGmMGQ&list=PL5Q2soXY2Zi9idayyIgBxUz7xRPS-wisBN&index=29)

- Computer Architecture, Fall 2020, Guest Lecture
  - **In-Memory Computing: Memory Devices & Applications** (ETH Zürich, Fall 2020)
  - [YouTube Link](https://www.youtube.com/watch?v=wNmqQHHiEZNK&list=PL5Q2soXY2Zi9idayyIgBxUz7xRPS-wisBN&index=41)
A Longer & Detailed Tutorial on PIM

Onur Mutlu,
"Memory-Centric Computing Systems"
[Slides (pptx) (pdf)]
[Executive Summary Slides (pptx) (pdf)]
[Tutorial Video (1 hour 51 minutes)]
[Executive Summary Video (2 minutes)]
[Abstract and Bio]
[Related Keynote Paper from VLSI-DAT 2020]
[Related Review Paper on Processing in Memory]

https://www.youtube.com/watch?v=H3sEaINPBOE

https://www.youtube.com/onurmutlulectures
Memory-Centric Computing Systems

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
12 December 2020
Iedm Tutorial

SAFARI
ETH Zürich
Carnegie Mellon

https://www.youtube.com/watch?v=H3sEaINPBOE
https://www.youtube.com/onurmutlulectures
A Recent Short Talk on PIM

https://www.youtube.com/onurmutlulectures

https://www.youtube.com/watch?v=jVYCchBGNVc

https://www.youtube.com/onurmutlulectures
Challenge and Opportunity for Future

Fundamentally
Energy-Efficient
(Data-Centric)
Computing Architectures
Fundamentally High-Performance (Data-Centric) Computing Architectures
Computing Architectures with Minimal Data Movement
Eliminating the Adoption Barriers

How to Enable Adoption of Processing in Memory
Potential Barriers to Adoption of PIM

1. **Functionality** and **applications & software** for PIM

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

3. **System** and **security** support: coherence, synchronization, virtual memory, isolation, ...

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
DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GOSE, University of Illinois at Urbana-Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, Institute for Research in Fundamental Sciences (IPM), Iran & ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Data movement between the CPU and main memory is a first-order obstacle against improving performance, scalability, and energy efficiency in modern systems. Computer systems employ a range of techniques to reduce overheads tied to data movement, spanning from traditional mechanisms (e.g., deep multi-level cache hierarchies, aggressive hardware prefetchers) to emerging techniques such as Near-Data Processing (NDP), where some computation is moved close to memory. Prior NDP works investigate the root causes of data movement bottlenecks using different profiling methodologies and tools. However, there is still a lack of understanding about the key metrics that can identify different data movement bottlenecks and their relation to traditional and emerging data movement mitigation mechanisms. Our goal is to methodically identify potential sources of data movement over a broad set of applications and to comprehensively compare traditional compute-centric data movement mitigation techniques (e.g., caching and prefetching) to more memory-centric techniques (e.g., NDP), thereby developing a rigorous understanding of the best techniques to mitigate each source of data movement.

With this goal in mind, we perform the first large-scale characterization of a wide variety of applications, across a wide range of application domains, to identify fundamental program properties that lead to data movement to/from main memory. We develop the first systematic methodology to classify applications based on the sources contributing to data movement bottlenecks. From our large-scale characterization of 77K functions across 345 applications, we select 144 functions to form the first open-source benchmark suite (DAMOV) for main memory data movement studies. We select a diverse range of functions that (1) represent different types of data movement bottlenecks, and (2) come from a wide range of application domains. Using NDP as a case study, we identify new insights about the different data movement bottlenecks and use these insights to determine the most suitable data movement mitigation mechanism for a particular application. We open-source DAMOV and the complete source code for our new characterization methodology at https://github.com/CMU-SAFARI/DAMOV.
DAMOV is Open Source

• We open-source our **benchmark suite** and our **toolchain**

**DAMOV:** A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks

DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing. The applications in the DAMOV benchmark suite belong to popular benchmark suites, including BWA, Chai, Darknet, GASE, Hardware Effects, Hashjoin, HPCC, HPCG, Ligra, PARSEC, Parboil, PolyBench, Phoenix, Rodinia, SPLASH-2, STREAM.
PIM Can Enable New Medical Platforms

Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali+, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

*Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017*

**Published:** 02 April 2018  **Article history** ▼

[Preliminary arxiv.org version]
Future of Genome Sequencing & Analysis

MinION from ONT

SmidgION from ONT
Accelerating Genome Analysis: A Primer on an Ongoing Journey

Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,

"Accelerating Genome Analysis: A Primer on an Ongoing Journey"


[Slides (pptx)(pdf)]
[Talk Video (1 hour 2 minutes)]
FPGA-based Processing Near Memory


FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh\textsuperscript{○}  Mohammed Alser\textsuperscript{○}  Damla Senol Cali\textsuperscript{●}

Dionysios Diamantopoulos\textsuperscript{▼}  Juan Gómez-Luna\textsuperscript{○}

Henk Corporaal\textsuperscript{*}  Onur Mutlu\textsuperscript{●●}

\textsuperscript{○}ETH Zürich  \textsuperscript{●●}Carnegie Mellon University

\textsuperscript{*}Eindhoven University of Technology  \textsuperscript{▼}IBM Research Europe
More on Fast & Efficient Genome Analysis …

- Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  [Slides (pptx) (pdf)]
  [Talk Video (1 hour 37 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]
Detailed Lectures on Genome Analysis

- **Computer Architecture, Fall 2020, Lecture 3a**
  - *Introduction to Genome Sequence Analysis* (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5](https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5)

- **Computer Architecture, Fall 2020, Lecture 8**
  - *Intelligent Genome Analysis* (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14](https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14)

- **Computer Architecture, Fall 2020, Lecture 9a**
  - *GenASM: Approx. String Matching Accelerator* (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=XoLpzmN-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15](https://www.youtube.com/watch?v=XoLpzmN-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15)

- **Accelerating Genomics Project Course, Fall 2020, Lecture 1**
  - *Accelerating Genomics* (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqLgwiDRQDTyId](https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqLgwiDRQDTyId)

[SAFARI](https://www.youtube.com/onurmutlulectures)
We Will Focus More
On the Following Two Parts
Challenge and Opportunity for Future

Data-Driven (Self-Optimizing) Computing Architectures
Challenge and Opportunity for Future

Data-Aware
(Expressive)
Computing Architectures
More Info in This Longer Tutorial…

- Onur Mutlu,
  "Memory-Centric Computing Systems"

[Slides (pptx) (pdf)]
[Executive Summary Slides (pptx) (pdf)]
[Tutorial Video (1 hour 51 minutes)]
[Executive Summary Video (2 minutes)]
[Abstract and Bio]
[Related Keynote Paper from VLSI-DAT 2020]
[Related Review Paper on Processing in Memory]

https://www.youtube.com/watch?v=H3sEaINPBOE

https://www.youtube.com/onurmutlulectures
Data-Driven Architectures
Corollaries: Architectures Today …

- Architectures are **terrible at dealing with data**
  - Designed to mainly store and move data vs. to compute
  - They are processor-centric as opposed to **data-centric**

- Architectures are **terrible at taking advantage of vast amounts of data** (and metadata) available to them
  - Designed to make simple decisions, ignoring lots of data
  - They make **human-driven decisions vs. data-driven decisions**

- Architectures are **terrible at knowing and exploiting different properties of application data**
  - Designed to treat all data as the same
  - They make **component-aware decisions vs. data-aware decisions**
Exploiting Data to Design Intelligent Architectures
System Architecture Design Today

- Human-driven
  - Humans design the policies (how to do things)

- Many (too) simple, short-sighted policies all over the system

- No automatic data-driven policy learning

- (Almost) no learning: cannot take lessons from past actions

Can we design fundamentally intelligent architectures?
An Intelligent Architecture

- Data-driven
  - Machine learns the “best” policies (how to do things)

- Sophisticated, workload-driven, changing, far-sighted policies

- Automatic data-driven policy learning

- All controllers are intelligent data-driven agents

How do we start?
Self-Optimizing Memory Controllers
Memory Controller

Resolves memory contention by scheduling requests

How to schedule requests to maximize system performance?
Why are Memory Controllers Difficult to Design?

- Need to obey DRAM timing constraints for correctness
  - There are many (50+) timing constraints in DRAM
  - tWTR: Minimum number of cycles to wait before issuing a read command after a write command is issued
  - tRC: Minimum number of cycles between the issuing of two consecutive activate commands to the same bank
  - ...

- Need to keep track of many resources to prevent conflicts
  - Channels, banks, ranks, data bus, address bus, row buffers, ...

- Need to handle DRAM refresh

- Need to manage power consumption

- Need to optimize performance & QoS (in the presence of constraints)
  - Reordering is not simple
  - Fairness and QoS needs complicates the scheduling problem

- ...

Many Memory Timing Constraints

Many Memory Timing Constraints


![Diagram of three phases of DRAM access](Figure 5. Three Phases of DRAM Access)

| Table 2. Timing Constraints (DDR3-1066) [43] |
|-----|-----|-----|-----|
| Phase | Commands | Name  | Value |
| 1     | ACT → READ  
      | ACT → WRITE | tRCD | 15ns |
|       | ACT → PRE   | tRAS | 37.5ns |
| 2     | READ → data | tCL  | 15ns |
|       | WRITE → data | tCWL | 11.25ns |
|       | data burst  | tBL  | 7.5ns  |
| 3     | PRE → ACT   | tRP  | 15ns  |
| 1 & 3 | ACT → ACT   | tRC (tRAS+tRP) | 52.5ns |
Memory Controller Design Is Becoming More Difficult

- Heterogeneous agents: CPUs, GPUs, and HWAs
- Main memory interference between CPUs, GPUs, HWAs
- Many timing constraints for various memory types
- Many goals at the same time: performance, fairness, QoS, energy efficiency, ...
Reality and Dream

- **Reality**: It difficult to design a policy that maximizes performance, QoS, energy-efficiency, ... 
  - Too many things to think about
  - Continuously changing workload and system behavior

- **Dream**: Wouldn’t it be nice if the DRAM controller automatically found a good scheduling policy on its own?
Self-Optimizing DRAM Controllers

- Problem: DRAM controllers are difficult to design
  - It is difficult for human designers to design a policy that can adapt itself very well to different workloads and different system conditions

- Idea: A memory controller that adapts its scheduling policy to workload behavior and system conditions using machine learning.

- Observation: Reinforcement learning maps nicely to memory control.

- Design: Memory controller is a reinforcement learning agent
  - It dynamically and continuously learns and employs the best scheduling policy to maximize long-term performance.
Self-Optimizing DRAM Controllers

Goal: Learn to choose actions to maximize $r_0 + \gamma r_1 + \gamma^2 r_2 + ...$ ( $0 \leq \gamma < 1$)

Figure 2: (a) Intelligent agent based on reinforcement learning principles;
Self-Optimizing DRAM Controllers

- Dynamically adapt the memory scheduling policy via interaction with the system at runtime
  - Associate system states and actions (commands) with long term reward values: each action at a given state leads to a learned reward
  - Schedule command with highest estimated long-term reward value in each state
  - Continuously update reward values for <state, action> pairs based on feedback from system
Self-Optimizing DRAM Controllers


Figure 4: High-level overview of an RL-based scheduler.
States, Actions, Rewards

❖ Reward function
- +1 for scheduling Read and Write commands
- 0 at all other times

Goal is to maximize long-term data bus utilization

❖ State attributes
- Number of reads, writes, and load misses in transaction queue
- Number of pending writes and ROB heads waiting for referenced row
- Request’s relative ROB order

❖ Actions
- Activate
- Write
- Read - load miss
- Read - store miss
- Precharge - pending
- Precharge - preemptive
- NOP
Performance Results

Large, robust performance improvements over many human-designed policies

Figure 7: Performance comparison of in-order, FR-FCFS, RL-based, and optimistic memory controllers

Figure 15: Performance comparison of FR-FCFS and RL-based memory controllers on systems with 6.4GB/s and 12.8GB/s peak DRAM bandwidth
Self Optimizing DRAM Controllers

+ **Continuous learning** in the presence of changing environment

+ **Reduced designer burden** in finding a good scheduling policy. Designer specifies:
  1) What system variables might be useful
  2) What target to optimize, but not how to optimize it

  -- How to specify **different objectives**? (e.g., fairness, QoS, ...)

  -- **Hardware complexity**?

  -- Design **mindset** and flow
More on Self-Optimizing DRAM Controllers

Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana,
"Self Optimizing Memory Controllers: A Reinforcement Learning Approach"

Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

Engin İpek\textsuperscript{1,2} \quad Onur Mutlu\textsuperscript{2} \quad José F. Martínez\textsuperscript{1} \quad Rich Caruana\textsuperscript{1}

\textsuperscript{1}Cornell University, Ithaca, NY 14850 USA
\textsuperscript{2}Microsoft Research, Redmond, WA 98052 USA
Self-Optimizing Memory Prefetchers

- To appear at MICRO 2021

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera\textsuperscript{1}  Konstantinos Kanellopoulos\textsuperscript{1}  Anant V. Nori\textsuperscript{2}  Taha Shahroodi\textsuperscript{3,1}
Sreenivas Subramoney\textsuperscript{2}  Onur Mutlu\textsuperscript{1}

\textsuperscript{1}ETH Zürich  \textsuperscript{2}Processor Architecture Research Labs, Intel Labs  \textsuperscript{3}TU Delft

Pythia
A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera, Konstantinos Kanellopoulos, Anant V. Nori, Taha Shahroodi, Sreenivas Subramoney, Onur Mutlu

https://github.com/CMU-SAFARI/Pythia
# Executive Summary

**Background**: Prefetchers predict addresses of future memory requests by associating memory access patterns with program context (called **feature**)

**Problem**: Three key shortcomings of prior prefetchers:
- Predict mainly using a **single program feature**
- Lack **inherent system awareness** (e.g., memory bandwidth usage)
- Lack **in-silicon customizability**

**Goal**: Design a prefetching framework that:
- Learns from **multiple features** and **inherent system-level feedback**
- Can be **customized in silicon** to use different features and/or prefetching objectives

**Contribution**: Pythia, which formulates prefetching as reinforcement learning problem
- Takes **adaptive** prefetch decisions using multiple features and system-level feedback
- Can be **customized in silicon** for target workloads via simple configuration registers
- Proposes a **realistic and practical** implementation of RL algorithm in hardware

**Key Results**:
- Evaluated using a wide range of workloads from SPEC CPU, PARSEC, Ligra, Cloudsuite
- Outperforms best prefetcher (in 1-core config.) by **3.4%, 7.7% and 17%** in 1/4/bw-constrained cores
- Up to **7.8% more performance** over basic Pythia across Ligra workloads via simple customization

---

[GitHub link: https://github.com/CMU-SAFARI/Pythia](https://github.com/CMU-SAFARI/Pythia)
Key Shortcomings in Prior Prefetchers

- We observe **three key shortcomings** that significantly limit performance benefits of prior prefetchers

1. **Predict mainly using a single program feature**

2. **Lack inherent system awareness**

3. **Lack in-silicon customizability**
A **prefetching framework** that can:

1. Learn to prefetch using *multiple features* and *inherent system-level feedback* information

2. Be **easily customized in silicon** to use different features and/or change prefetcher’s objectives
Our Proposal

Pythia

Formulates prefetching as a reinforcement learning problem

Pythia is named after the oracle of Delphi, who is known for her accurate prophecies
https://en.wikipedia.org/wiki/Pythia
Basics of Reinforcement Learning (RL)

• Algorithmic approach to learn to take an action in a given situation to maximize a numerical reward

Agent

Environment

• Agent stores Q-values for every state-action pair
  - Expected return for taking an action in a state
  - Given a state, selects action that provides highest Q-value
Formulating Prefetching as RL

Agent

Environment

State (S_t)

Action (A_t)

Reward (R_t + 1)

Prefetch from address A + offset (O)

Features of memory request to address A (e.g., PC)
Pythia Overview

- **Q-Value Store**: Records Q-values for all state-action pairs
- **Evaluation Queue**: A FIFO queue of recently-taken actions

![Diagram of Pythia Overview]

1. **Demand Request**
   - Assign reward to corresponding EQ entry

2. **State Vector**
   - Look up QVStore

3. **Q-Value Store (QVStore)**
   - Find the Action with max Q-Value

4. **Memory Hierarchy**
   - Generate prefetch

5. **Evaluation Queue (EQ)**
   - Insert prefetch action & State-Action pair in EQ

6. **Evict EQ entry and update QVStore**

7. **Set filled bit**
Simulation Methodology

- **Champsim** [3] trace-driven simulator

- **150** single-core memory-intensive workload traces
  - SPEC CPU2006 and CPU2017
  - PARSEC 2.1
  - Ligra
  - Cloudsuite

- Homogeneous and heterogeneous multi-core mixes

- **Five** state-of-the-art prefetchers
  - SPP [Kim+, MICRO’16]
  - Bingo [Bakhshali, HPCA’19]
  - MLOP [Shakerinava+, 3rd Prefetching Championship, 2019 ]
  - SPP+DSPatch [Bera+, MICRO’19]
  - SPP+PPF [Bhatia+, ISCA’20]


SAFARI
Basic Pythia Configuration

• Derived from **automatic design-space exploration**

**State:** 2 features
- PC+Delta
- Sequence of last-4 deltas

**Actions:** 16 prefetch offsets
- Ranging between -6 to +32. Including 0.

**Rewards:**
- $R_{AT} = +20$; $R_{AL} = +12$; $R_{NP-H} = -2$; $R_{NP-L} = -4$
- $R_{IN-H} = -14$; $R_{IN-L} = -8$; $R_{CL} = -12$
Performance with Varying Core Count

Geomean speedup over no prefetching

Number of cores

- Pythia
- SPP
- MLOP
- Bingo

3.4%
7.7%
Performance with Varying Core Count

1. Pythia consistently provides the highest performance in all core configurations.

2. Pythia’s gain increases with core count.
Performance with Varying DRAM Bandwidth

Geomean speedup over no prefetching

- Pythia
- Bingo
- MLOP
- SPP

Baseline

~Intel Xeon 6258R
~AMD EPYC Rome 7702P
~AMD Threadripper 3990x

DRAM MTPS (in log scale)
Performance with Varying DRAM Bandwidth

Pythia outperforms prior best prefetchers for a wide range of DRAM bandwidth configurations.
Pythia’s Overhead

• **25.5 KB** of total metadata storage *per core*
  - Only simple tables

• We also model functionally-accurate Pythia with full complexity in **Chisel** [4] HDL

- **1.03% area overhead**
- **0.4% power overhead**
- Satisfies prediction latency

*of a desktop-class 4-core Skylake processor (Xeon D2132IT, 60W)*

More in the Paper

• Performance comparison with unseen traces
  - Pythia provides equally high performance benefits

• Comparison against multi-level prefetchers
  - Pythia outperforms prior best multi-level prefetchers

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera¹  Konstantinos Kanellopoulos¹  Anant V. Nori²  Taha Shahroodi³,¹
Sreenivas Subramoney²  Onur Mutlu¹
¹ETH Zürich  ²Processor Architecture Research Labs, Intel Labs  ³TU Delft


• Performance sensitivity towards different features and hyperparameter values

• Detailed single-core and four-core performance
Pythia is Open Source

https://github.com/CMU-SAFARI/Pythia

• MICRO’21 artifact evaluated
• Champsim source code + Chisel modeling code
• All traces used for evaluation
Pythia
A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera, Konstantinos Kanellopoulos, Anant V. Nori, Taha Shahroodi, Sreenivas Subramoney, Onur Mutlu

https://github.com/CMU-SAFARI/Pythia
Self-Optimizing Memory Prefetchers

- To appear at MICRO 2021

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera¹ Konstantinos Kanellopoulos¹ Anant V. Nori² Taha Shahrroodi³,¹
Sreenivas Subramoney² Onur Mutlu¹

¹ETH Zürich  ²Processor Architecture Research Labs, Intel Labs  ³TU Delft

An Intelligent Architecture

- Data-driven
  - Machine learns the “best” policies (how to do things)

- Sophisticated, workload-driven, changing, far-sighted policies

- Automatic data-driven policy learning

- All controllers are intelligent data-driven agents

We need to rethink design (of all controllers)
Challenge and Opportunity for Future

Data-Driven
(Self-Optimizing)
Computing Architectures
Data-Aware Architectures
Corollaries: Architectures Today …

- Architectures are **terrible at dealing with data**
  - Designed to mainly store and move data vs. to compute
  - They are processor-centric as opposed to **data-centric**

- Architectures are **terrible at taking advantage of vast amounts of data** (and metadata) available to them
  - Designed to make simple decisions, ignoring lots of data
  - They make **human-driven decisions vs. data-driven decisions**

- Architectures are **terrible at knowing and exploiting different properties of application data**
  - Designed to treat all data as the same
  - They make **component-aware decisions vs. data-aware**
Data-Aware Architectures

- A data-aware architecture understands what it can do with and to each piece of data.

- It makes use of different properties of data to improve performance, efficiency and other metrics:
  - Compressibility
  - Approximability
  - Locality
  - Sparsity
  - Criticality for Computation
  - Access Semantics
  - ...
One Problem: Limited Expressiveness

Higher-level information is not visible to HW

Software

Hardware

Data Structures

Code Optimizations

Access Patterns

Data Type

Integer

Float

Char

100011111...

101010011...

Instructions

Memory Addresses
A Solution: More Expressive Interfaces

Performance
Software

Functionality

Hardware

ISA
Virtual Memory

Higher-level Program Semantics

Expressive Memory “XMem”
Expressive (Memory) Interfaces

  [Slides (pptx) (pdf)] [Lightning Talk Slides (pptx) (pdf)]
  [Lightning Talk Video]

A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory

Nandita Vijaykumar†§ Abhilasha Jain† Diptesh Majumdar† Kevin Hsieh† Gennady Pekhimenko‡
Eiman Ebrahimi§ Nastaran Hajinazar† Phillip B. Gibbons† Onur Mutlu§†

†Carnegie Mellon University ‡University of Toronto §NVIDIA
‡Simon Fraser University §ETH Zürich
SW provides key program information to HW

Data Structures

Access Patterns

Data Type/Layout

Software

Hardware

Data Placement

Prefetcher

Data Compression
Broader goal: Enable many cross-layer optimizations

Express:
- Data structures
- Access semantics
- Data types
- Working set
- Reuse
- Access frequency

Optimizations:
- Cache Management
- Data Placement in DRAM
- Data Compression
- Approximation
- DRAM Cache Management
- NVM Management
- NUCA/NUMA Optimizations

Benefits:
- More efficient HW:
  - ✔ Performance
- Reduced SW burden:
  - ✔ Programmability
  - ✔ Portability
Our approach: Rich cross-layer abstractions

1. Generality: Enable a wide range of cross-layer approaches
2. Minimize programmer effort
3. Overhead

Approach: Flexibly associate specific semantic information with any data & code
Example: XMem

• **Goal**: convey data semantics to the hardware enables more intelligent management of resources.

• **XMem**: introduces a new HW/SW abstraction, called *Atom*, for conveying data semantics

## XMem Aids/Enables Many Optimizations

### Table 1: Summary of the example memory optimizations that XMem aids.

<table>
<thead>
<tr>
<th>Memory optimization</th>
<th>Example semantics provided by XMem (described in §3.3)</th>
<th>Example Benefits of XMem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache management</td>
<td>(i) Distinguishing between data structures or pools of similar data; (ii) Working set size; (iii) Data reuse</td>
<td>Enables: (i) applying different caching policies to different data structures or pools of data; (ii) avoiding cache thrashing by knowing the active working set size; (iii) bypassing/prioritizing data that has no/high reuse. (§5)</td>
</tr>
<tr>
<td>Page placement in DRAM, e.g., [23, 24]</td>
<td>(i) Distinguishing between data structures; (ii) Access pattern; (iii) Access intensity</td>
<td>Enables page placement at the data structure granularity to (i) isolate data structures that have high row buffer locality and (ii) spread out concurrently-accessed irregular data structures across banks and channels to improve parallelism. (§6)</td>
</tr>
<tr>
<td>Cache/mem compression, e.g., [25–32]</td>
<td>(i) Data type: integer, float, char; (ii) Data properties: sparse, pointer, data index</td>
<td>Enables using a different compression algorithm for each data structure based on data type and data properties, e.g., sparse data encodings, FP-specific compression, delta-based compression for pointers [27].</td>
</tr>
<tr>
<td>Data prefetching, e.g., [33–36]</td>
<td>(i) Access pattern: strided, irregular, irregular but repeated (e.g., graphs), access stride; (ii) Data type: index, pointer</td>
<td>Enables (i) highly accurate software-driven prefetching while leveraging the benefits of hardware prefetching (e.g., by being memory bandwidth-aware, avoiding cache thrashing); (ii) using different prefetcher types for different data structures: e.g., stride [33], tile-based [20], pattern-based [34–37], data-based for indices/pointers [38, 39], etc.</td>
</tr>
<tr>
<td>DRAM cache management, e.g., [40–46]</td>
<td>(i) Access intensity; (ii) Data reuse; (iii) Working set size</td>
<td>(i) Helps avoid cache thrashing by knowing working set size [44]; (ii) Better DRAM cache management via reuse behavior and access intensity information.</td>
</tr>
<tr>
<td>Approximation in memory, e.g., [47–53]</td>
<td>(i) Distinguishing between pools of similar data; (ii) Data properties: tolerance towards approximation</td>
<td>Enables (i) each memory component to track how approximable data is (at a fine granularity) to inform approximation techniques; (ii) data placement in heterogeneous reliability memories [54].</td>
</tr>
<tr>
<td>Data placement, NUMA systems, e.g., [55, 56]</td>
<td>(i) Data partitioning across threads (i.e., relating data to threads that access it); (ii) Read-Write properties</td>
<td>Reduces the need for profiling or data migration (i) to co-locate data with threads that access it and (ii) to identify Read-Only data, thereby enabling techniques such as replication.</td>
</tr>
<tr>
<td>Data placement, hybrid memories, e.g., [16, 57, 58]</td>
<td>(i) Read-Write properties (Read-Only/Read-Write); (ii) Access intensity; (iii) Data structure size; (iv) Access pattern</td>
<td>Avoids the need for profiling/migration of data in hybrid memories to (i) effectively manage the asymmetric read-write properties in NVM (e.g., placing Read-Only data in the NVM) [16, 57]; (ii) make tradeoffs between data structure &quot;hotness&quot; and size to allocate fast/high bandwidth memory [14]; and (iii) leverage row-buffer locality in placement based on access pattern [45].</td>
</tr>
<tr>
<td>Managing NUCA systems, e.g., [15, 59]</td>
<td>(i) Distinguishing pools of similar data; (ii) Access intensity; (iii) Read-Write or Private-Shared properties</td>
<td>(i) Enables using different cache policies for different data pools (similar to [15]); (ii) Reduces the need for reactive mechanisms that detect sharing and read-write characteristics to inform cache policies.</td>
</tr>
</tbody>
</table>
Expressive (Memory) Interfaces

- Nandita Vijaykumar, Abhilasha Jain, Diptesh Majumdar, Kevin Hsieh, Gennady Pekhimenko, Eiman Ebrahimi, Nastaran Hajinazar, Phillip B. Gibbons and Onur Mutlu,
"A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory"
[Slides (pptx) (pdf)] [Lightning Talk Slides (pptx) (pdf)]
[Lightning Talk Video]

A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory

Nandita Vijaykumar†§  Abhilasha Jain†  Diptesh Majumdar†  Kevin Hsieh†  Gennady Pekhimenko‡
Eiman Ebrahimi‡  Nastaran Hajinazar†  Phillip B. Gibbons†  Onur Mutlu§†

†Carnegie Mellon University  ‡University of Toronto  §ETH Zürich
The Locality Descriptor: A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs

Nandita Vijaykumar†§ Eiman Ebrahimi‡ Kevin Hsieh†
Phillip B. Gibbons† Onur Mutlu§†
†Carnegie Mellon University ‡NVIDIA §ETH Zürich
Exploiting data locality in GPUs is a challenging task.

Flexible, architecture-agnostic interface

Performance Benefits:
26.6% (up to 46.6%) from cache locality
53.7% (up to 2.8x) from NUMA locality
An Example: Hybrid Memory Management

Hardware/software manage data allocation and movement to achieve the best of multiple technologies.

Yoon+, “Row Buffer Locality Aware Caching Policies for Hybrid Memories,” ICCD 2012 Best Paper Award.
An Example: Heterogeneous-Reliability Memory

- Yixin Luo, Sriram Govindan, Bikash Sharma, Mark Santaniello, Justin Meza, Aman Kansal, Jie Liu, Badriddine Khessib, Kushagra Vaid, and Onur Mutlu, "Characterizing Application Memory Error Vulnerability to Optimize Data Center Cost via Heterogeneous-Reliability Memory" Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Atlanta, GA, June 2014. [Summary] [Slides (pptx) (pdf)] [Coverage on ZDNet]
Exploiting Memory Error Tolerance with Hybrid Memory Systems

Vulnerable data

Tolerant data

Reliable memory

Low-cost memory

On Microsoft’s Web Search workload
Reduces server hardware cost by 4.7 %
Achieves single server availability target of 99.90 %

Heterogeneous-Reliability Memory [DSN 2014]
Step 1: Characterize and classify application memory error tolerance

Step 2: Map application data to the HRM system enabled by SW/HW cooperative solutions
More on Heterogeneous-Reliability Memory

- Yixin Luo, Sriram Govindan, Bikash Sharma, Mark Santaniello, Justin Meza, Aman Kansal, Jie Liu, Badriddine Khessib, Kushagra Vaid, and Onur Mutlu,

"Characterizing Application Memory Error Vulnerability to Optimize Data Center Cost via Heterogeneous-Reliability Memory"
Proceedings of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Atlanta, GA, June 2014. [Summary] [Slides (pptx) (pdf)] [Coverage on ZDNet]
Data-Aware Cross-Layer Hybrid System Management

- Heterogeneous agents: CPUs, GPUs, and HWAs
- Main memory interference between CPUs, GPUs, HWAs
- Many timing constraints for various memory types
- Many goals at the same time: performance, fairness, QoS, energy efficiency, ...
Another Example: EDEN for DNNs

- Deep Neural Network evaluation is very DRAM-intensive (especially for large networks)

1. Some data and layers in DNNs are very tolerant to errors

2. Reduce DRAM latency and voltage on such data and layers

3. While still achieving a user-specified DNN accuracy target by making training DRAM-error-aware

Data-aware management of DRAM latency and voltage for Deep Neural Network Inference
Example DNN Data Type to DRAM Mapping

Mapping example of ResNet-50:

Map more error-tolerant DNN layers to DRAM partitions with lower voltage/latency

4 DRAM partitions with different error rates
**EDEN: Overview**

**Key idea:** Enable **accurate, efficient** DNN inference using approximate DRAM

EDEN is an **iterative** process that has **3 key steps**
Average 21% DRAM energy reduction maintaining accuracy within 1% of original
CPU: Performance Evaluation

Average 8% system speedup
Some workloads achieve 17% speedup

EDEN achieves close to the ideal speedup possible via tRCD scaling
GPU, Eyeriss, and TPU: Energy Evaluation

- **GPU**: average 37% energy reduction
- **Eyeriss**: average 31% energy reduction
- **TPU**: average 32% energy reduction
EDEN: Data-Aware Efficient DNN Inference

- Skanda Koppula, Lois Orosa, A. Giray Yaglikci, Roknoddin Azizi, Taha Shahroodi, Konstantinos Kanellopoulos, and Onur Mutlu,
"EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM"
Proceedings of the 52nd International Symposium on Microarchitecture (MICRO), Columbus, OH, USA, October 2019.
[Lightning Talk Slides (pptx) (pdf)]
[Lightning Talk Video (90 seconds)]
SMASH: SW/HW Indexing Acceleration

Konstantinos Kanellopoulos, Nandita Vijaykumar, Christina Giannoula, Roknoddin Azizi, Skanda Koppula, Nika Mansouri Ghiasi, Taha Shahroodi, Juan Gomez-Luna, and Onur Mutlu,

"SMASH: Co-designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations"

Proceedings of the 52nd International Symposium on Microarchitecture (MICRO), Columbus, OH, USA, October 2019.

[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Poster (pptx) (pdf)]
[Lightning Talk Video (90 seconds)]
[Full Talk Lecture (30 minutes)]
The Virtual Block Interface: A Flexible Alternative to the Conventional Virtual Memory Framework


*ETH Zürich †Simon Fraser University *University of Washington †Carnegie Mellon University
©King Mongkut’s University of Technology North Bangkok ∇Boston University ∨Microsoft Research India
SW/HW Climate Modeling Accelerator

- Gagandeep Singh, Dionysios Diamantopoulos, Christoph Hagleitner, Juan Gómez-Luna, Sander Stuijk, Onur Mutlu, and Henk Corporaal,

"NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling"

Proceedings of the 30th International Conference on Field-Programmable Logic and Applications (FPL), Gothenburg, Sweden, September 2020.

[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (23 minutes)]

Nominated for the Stamatis Vassiliadis Memorial Award.

NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

Gagandeep Singh\textsuperscript{a,b,c} \quad Dionysios Diamantopoulos\textsuperscript{c} \quad Christoph Hagleitner\textsuperscript{c} \quad Juan Gómez-Luna\textsuperscript{b}

Sander Stuijk\textsuperscript{a} \quad Onur Mutlu\textsuperscript{b} \quad Henk Corporaal\textsuperscript{a}

\textsuperscript{a}Eindhoven University of Technology \quad \textsuperscript{b}ETH Zürich \quad \textsuperscript{c}IBM Research Europe, Zurich
## HW/SW Time Series Analysis Accelerator

- Ivan Fernandez, Ricardo Quislant, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, Eladio Gutiérrez, Oscar Plata, and Onur Mutlu,

"**NATSA: A Near-Data Processing Accelerator for Time Series Analysis**"


[Slides (pptx) (pdf)]

[Talk Video (10 minutes)]

[Source Code]

---

### NATSA: A Near-Data Processing Accelerator for Time Series Analysis

<table>
<thead>
<tr>
<th>Ivan Fernandez§</th>
<th>Ricardo Quislant§</th>
<th>Christina Giannoula†</th>
<th>Mohammed Alser†</th>
</tr>
</thead>
<tbody>
<tr>
<td>Juan Gómez-Luna‡</td>
<td>Eladio Gutiérrez§</td>
<td>Oscar Plata§</td>
<td>Onur Mutlu‡</td>
</tr>
</tbody>
</table>

§University of Malaga

†National Technical University of Athens

‡ETH Zürich
FPGA-based Processing Near Memory


FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh♦ Mohammed Alser♦ Damla Senol Cali♦ Dionysios Diamantopoulos▼ Juan Gómez-Luna♦ Henk Corporaal* Onur Mutlu♦▼

♦ETH Zürich ▼Carnegie Mellon University *Eindhoven University of Technology ▼IBM Research Europe
Accelerating Linked Data Structures

Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu,
"Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"
Proceedings of the 34th IEEE International Conference on Computer Design (ICCD), Phoenix, AZ, USA, October 2016.

Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation

Kevin Hsieh† Samira Khan‡ Nandita Vijaykumar†
Kevin K. Chang† Amirali Boroumand‡ Saugata Ghose† Onur Mutlu§
†Carnegie Mellon University ‡University of Virginia §ETH Zürich
Accelerating Approximate String Matching


[Lighting Talk Video (1.5 minutes)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (18 minutes)]
[Slides (pptx) (pdf)]
Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,
"Accelerating Genome Analysis: A Primer on an Ongoing Journey"
[Slides (pptx)(pdf)]
[Talk Video (1 hour 2 minutes)]
Challenge and Opportunity for Future Data-Aware (Expressive) Computing Architectures
We Need to **Rethink** the Entire Stack

<table>
<thead>
<tr>
<th>Problem</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program/Language</td>
<td>System Software</td>
</tr>
<tr>
<td>SW/HW Interface</td>
<td>Micro-architecture</td>
</tr>
<tr>
<td>Logic</td>
<td>Devices</td>
</tr>
<tr>
<td>Electrons</td>
<td></td>
</tr>
</tbody>
</table>

We can get there case by case
Concluding Remarks
Recap: Corollaries: Architectures Today

- Architectures are terrible at dealing with data
  - Designed to mainly store and move data vs. to compute
  - They are processor-centric as opposed to data-centric

- Architectures are terrible at taking advantage of vast amounts of data (and metadata) available to them
  - Designed to make simple decisions, ignoring lots of data
  - They make human-driven decisions vs. data-driven

- Architectures are terrible at knowing and exploiting different properties of application data
  - Designed to treat all data as the same
  - They make component-aware decisions vs. data-aware
Concluding Remarks

- It is time to design **principled system architectures** to solve the **data handling (i.e., memory/storage) problem**

- Design complete systems to be truly balanced, high-performance, and **energy-efficient** → intelligent systems
  - **Data-centric, data-driven, data-aware**

- Enable computation capability inside and close to memory

- **This can**
  - Lead to **orders-of-magnitude** improvements
  - Enable new applications & computing platforms
  - Enable better understanding of nature
  - ...


Fundamentally Better Architectures

Data-centric
Data-driven
Data-aware
We Need to Revisit the Entire Stack

We can get there step by step
We Need to Exploit Good Principles

- Data-centric system design
- All components intelligent
- Better cross-layer communication, better interfaces
- Better-than-worst-case design
- Heterogeneity
- Flexibility, adaptability

Open minds
PIM Review and Open Problems

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{b,c}, Juan Gómez-Luna\textsuperscript{a}, Rachata Ausavarungrun\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

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 computing 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, 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-
A Workload and Programming Ease Driven Perspective of Processing-in-Memory

Saugata Ghose† Amirali Boroumand† Jeremie S. Kim†‡ Juan Gómez-Luna§ Onur Mutlu§†

†Carnegie Mellon University
‡ETH Zürich

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu, "Processing-in-Memory: A Workload-Driven Perspective"
[Preliminary arXiv version]

A Longer Tutorial Version of This Talk

Onur Mutlu,
"Memory-Centric Computing Systems"
[Slides (pptx) (pdf)]
[Executive Summary Slides (pptx) (pdf)]
[Tutorial Video (1 hour 51 minutes)]
[Executive Summary Video (2 minutes)]
[Abstract and Bio]
[Related Keynote Paper from VLSI-DAT 2020]
[Related Review Paper on Processing in Memory]

https://www.youtube.com/watch?v=H3sEaINPBOE

https://www.youtube.com/onurmutlulectures
Memory-Centric Computing Systems

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
12 December 2020
IEDM Tutorial

https://www.youtube.com/watch?v=H3sEaINPBOE
https://www.youtube.com/onurmutlulectures
A Recent Short Talk on PIM

https://www.youtube.com/watch?v=jVYCchBGNVc

https://www.youtube.com/onurmutlulectures
Funding Acknowledgments

- Alibaba, AMD, ASML, Google, Facebook, Hi-Silicon, HP Labs, Huawei, IBM, Intel, Microsoft, Nvidia, Oracle, Qualcomm, Rambus, Samsung, Seagate, VMware
- NSF
- NIH
- GSRC
- SRC
- CyLab
- EFCL
Acknowledgments

Think BIG, Aim HIGH!

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

2019 and the first three months of 2020 have been very positive eventful times for SAFARI.
SAFARI Newsletter January 2021 Edition

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

Think Big, Aim High, and Have a Wonderful 2021!

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
Referenced Papers, Talks, Artifacts

- All are available at

  https://people.inf.ethz.ch/omutlu/projects.htm

  https://www.youtube.com/onurmutlulectures

  https://github.com/CMU-SAFARI/
Intelligent Architectures for Intelligent Systems

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
14 October 2021
Huawei STW 2021: SW/HW Co-Design for Comp Arch
Backup Slides
A Quote from A Famous Architect

“architecture [...] based upon principle, and not upon precedent”
Precedent-Based Design?

“architecture [...] based upon principle, and not upon precedent”
Principled Design

- “architecture [...] based upon principle, and not upon precedent”
The Overarching Principle

Organic architecture

From Wikipedia, the free encyclopedia

Organic architecture is a philosophy of architecture which promotes harmony between human habitation and the natural world through design approaches so sympathetic and well integrated with its site, that buildings, furnishings, and surroundings become part of a unified, interrelated composition.

A well-known example of organic architecture is Fallingwater, the residence Frank Lloyd Wright designed for the Kaufmann family in rural Pennsylvania. Wright had many choices to locate a home on this large site, but chose to place the home directly over the waterfall and creek creating a close, yet noisy dialog with the rushing water and the steep site. The horizontal striations of stone masonry with daring cantilevers of colored beige concrete blend with native rock outcroppings and the wooded environment.
Another Example: Precedent-Based Design

Source: http://cookiemagik.deviantart.com/art/Train-station-207266944
Principled Design

Source: By Toni_V, CC BY-SA 2.0, https://commons.wikimedia.org/w/index.php?curid=4087256
Another Principled Design
Another Principled Design

Principle Applied to Another Structure

The Overarching Principle

Zoomorphic architecture

From Wikipedia, the free encyclopedia

**Zoomorphic architecture** is the practice of using animal forms as the inspirational basis and blueprint for architectural design. "While animal forms have always played a role adding some of the deepest layers of meaning in architecture, it is now becoming evident that a new strand of *biomorphism* is emerging where the meaning derives not from any specific representation but from a more general allusion to biological processes."[^1]

Some well-known examples of Zoomorphic architecture can be found in the TWA Flight Center building in New York City, by Eero Saarinen, or the Milwaukee Art Museum by Santiago Calatrava, both inspired by the form of a bird’s wings.[^3]
Overarching Principles for Computing?

Source: http://spectrum.ieee.org/image/MjYzMzAyMg.jpeg
Readings, Videos, Reference Materials
More on My Research & Teaching
Brief Self Introduction

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
- ...
Current Research Mission

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

Heterogeneous Processors and Accelerators

Hybrid Main Memory

Persistent Memory/Storage

Graphics and Vision Processing

Build fundamentally better architectures
Four Key Current Directions

- Fundamentally **Secure/Reliable/Safe** Architectures

- Fundamentally **Energy-Efficient** Architectures
  - **Memory-centric** (Data-centric) Architectures

- Fundamentally **Low-Latency and Predictable** Architectures

- Architectures for **AI/ML, Genomics, Medicine, Health**
The Transformation Hierarchy

Computer Architecture (expanded view)

- Problem
- Algorithm
- Program/Language
- System Software

- SW/HW Interface
- Micro-architecture
- Logic
- Devices
- Electrons

Computer Architecture (narrow view)
Axiom

To achieve the highest energy efficiency and performance:

we must take the expanded view
of computer architecture

Co-design across the hierarchy:
Algorithms to devices

Specialize as much as possible
within the design goals
Current Research Mission & Major Topics

Build fundamentally better architectures

- Data-centric arch. for low energy & high perf.
  - Proc. in Mem/DRAM, NVM, unified mem/storage

- Low-latency & predictable architectures
  - Low-latency, low-energy yet low-cost memory
  - QoS-aware and predictable memory systems

- Fundamentally secure/reliable/safe arch.
  - Tolerating all bit flips; patchable HW; secure mem

- Architectures for ML/AI/Genomics/Health/Med
  - Algorithm/arch./logic co-design; full heterogeneity

- Data-driven and data-aware architectures
  - ML/AI-driven architectural controllers and design
  - Expressive memory and expressive systems

Broad research spanning apps, systems, logic with architecture at the center
Think BIG, Aim HIGH!

https://safari.ethz.ch
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!

SAFARI

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 PhD and Post-Doc Alumni

- [https://safari.ethz.ch/safari-alumni/](https://safari.ethz.ch/safari-alumni/)
- Damla Senol Cali (Bionano Genomics)
- Nastaran Hajinazar (ETH Zurich)
- Gagandeep Singh (ETH Zurich)
- Amirali Boroumand (Stanford Univ)
- Jeremie Kim (ETH Zurich)
- Nandita Vijaykumar (Univ. of Toronto, Assistant Professor)
- Kevin Hsieh (Microsoft Research, Senior Researcher)
- Justin Meza (Facebook)
- Mohammed Alser (ETH Zurich)
- Yixin Luo (Google)
- Kevin Chang (Facebook)
- Rachata Ausavarungnirun (KMUNTB, Assistant Professor)
- Gennady Pekhimenko (Univ. of Toronto, Assistant Professor)
- Vivek Seshadri (Microsoft Research)
- Donghyuk Lee (NVIDIA Research, Senior Researcher)
- Yoongu Kim (Google)
- Lavanya Subramanian (Intel Labs → Facebook)
- Samira Khan (Univ. of Virginia, Assistant Professor)
- Saugata Ghose (Univ. of Illinois, Assistant Professor)
- Jawad Haj-Yahya (Huawei Research Zurich, Principal Researcher)
SAFARI Research Group: Introduction and Research

- Onur Mutlu,

"SAFARI Research Group: Introduction & Research"

Talk at ETH Future Computing Laboratory Welcome Workshop (EFCL), Virtual, 6 July 2021.

[Slides (pptx) (pdf)]
A Talk on Impactful Research & Teaching

Applying to Grad School & Doing Impactful Research

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
13 June 2020
Undergraduate Architecture Mentoring Workshop @ ISCA 2021

Arch. Mentoring Workshop @ISCA’21 - Applying to Grad School & Doing Impactful Research - Onur Mutlu
1,563 views • Premiered Jun 16, 2021

Panel talk at Undergraduate Architecture Mentoring Workshop at ISCA 2021
(https://sites.google.com/wisc.edu/uar...)

https://www.youtube.com/watch?v=83tlorht7Mc&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=54
Teaching drives Research
Research drives Teaching
Focus on learning and scholarship
Principle: Insight and Ideas

Focus on Insight
Encourage New Ideas
Principle: Learning and Scholarship

The quality of your work defines your impact
You can make a good impact on the world
Research & Teaching: Some Overview Talks

https://www.youtube.com/onurmutlulectures

- Future Computing Architectures
  - https://www.youtube.com/watch?v=kgiZISOcGFM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=1

- Enabling In-Memory Computation
  - https://www.youtube.com/watch?v=njX_14584Jw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=16

- Accelerating Genome Analysis
  - https://www.youtube.com/watch?v=r7sn41IH-4A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=41

- Rethinking Memory System Design
  - https://www.youtube.com/watch?v=F7xZLNMIY1E&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=3

- Intelligent Architectures for Intelligent Machines
  - https://www.youtube.com/watch?v=c6_LqzuNdkw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=25

- The Story of RowHammer
  - https://www.youtube.com/watch?v=sgd7PHQQ1AI&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=39
Online Courses & Lectures

- **First Computer Architecture & Digital Design Course**
  - Digital Design and Computer Architecture
  - Spring 2021 Livestream Edition:
    https://www.youtube.com/watch?v=LbC0EZY8yw4&list=PL5Q2soXY2Zi_uej3aY39YB5pfW4SJ7LlN

- **Advanced Computer Architecture Course**
  - Computer Architecture
  - Fall 2020 Edition:
    https://www.youtube.com/watch?v=c3mPdZA-Fmc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN

SAFARI  https://www.youtube.com/onurmutlulectures
Onur Mutlu Lectures
16.9K subscribers

Popular uploads

- How Computers Work
  49K views • 1 year ago

- Computer Architecture - Lecture 1: Introduction and Basics
  36K views • 3 years ago

- Computer Architecture - Lecture 1: Introduction and Basics
  31K views • 1 year ago

- Computer Architecture - Lecture 1: Introduction and Basics
  30K views • 8 months ago

- Design of Digital Circuits
  22K views • 2 years ago

- Computer Architecture - Lecture 2: Fundamentals
  17K views • 3 years ago

First Course in Computer Architecture & Design 2021-2013

- Livestream - Digital Design and Computer Architecture - ETH Zurich
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Digital Design & Computer Architecture - ETH Zurich - Spring 2019
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Design of Digital Circuits - ETH Zurich - Spring 2018
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Design of Digital Circuits - ETH Zurich - Spring 2017
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Digital Circuits and Computer Architecture - ETH Zurich
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Spring 2015 -- Computer Architecture Lectures...
  Carnegie Mellon Computer Architecture
  VIEW FULL PLAYLIST

Advanced Computer Architecture Courses 2020-2012

- Computer Architecture - ETH Zurich - Fall 2020
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Computer Architecture - ETH Zurich - Fall 2019
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Computer Architecture - ETH Zurich - Fall 2018
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Computer Architecture - ETH Zurich - Fall 2017
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Fall 2015 - 740 Computer Architecture
  Carnegie Mellon Computer Architecture
  VIEW FULL PLAYLIST

- Fall 2013 - 740 Computer Architecture - Carnegie Mellon
  Carnegie Mellon Computer Architecture
  VIEW FULL PLAYLIST

Special Courses on Memory Systems

- Memory Technology Lectures
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Champery Winter School 2020 - Memory Systems and Memory
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- Perugia NIPS Summer School 2019
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- SAMOS Tutorial 2019 - Memory Systems
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- TU Wien 2019 - Memory Systems and Memory-Centric...
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

- ACACES 2018 Lectures -- Memory Systems and Memory...
  Onur Mutlu Lectures
  VIEW FULL PLAYLIST

Research Talks

https://www.youtube.com/onurmutlulectures
DDCA (Spring 2021)

- [https://www.youtube.com/watch?v=LbC0EZY8yw4&list=PL5Q2soXY2Zi_uej3aY39YB5pfW4SJ7LIN](https://www.youtube.com/watch?v=LbC0EZY8yw4&list=PL5Q2soXY2Zi_uej3aY39YB5pfW4SJ7LIN)

- Bachelor’s course
  - 2nd semester at ETH Zurich
  - Rigorous introduction into “How Computers Work”
  - Digital Design/Logic
  - Computer Architecture
  - 10 FPGA Lab Assignments
Comp Arch (Fall 2020)

- https://www.youtube.com/watch?v=c3mPdZA-Fmc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN

- Master’s level course
  - Taken by Bachelor’s/Masters/PhD students
  - Cutting-edge research topics + fundamentals in Computer Architecture
  - 5 Simulator-based Lab Assignments
  - Potential research exploration
  - Many research readings
Seminar (Spring’21)

- [https://www.youtube.com/watch?v=t3m93ZpLOyw&list=PL5Q2soXY2Zi_awYdjmWVIUegsbY7TPGW4](https://www.youtube.com/watch?v=t3m93ZpLOyw&list=PL5Q2soXY2Zi_awYdjmWVIUegsbY7TPGW4)

- Critical analysis course
  - Taken by Bachelor’s/Masters/PhD students
  - Cutting-edge research topics + fundamentals in Computer Architecture
  - 20+ research papers, presentations, analyses
SAFARI Project & Seminars Courses (Spring 2021)

Welcome to the wiki for Project and Seminar courses SAFARI offers.

Courses we offer:

- Understanding and Improving Modern DRAM Performance, Reliability, and Security with Hands-On Experiments
- Designing and Evaluating Memory Systems and Modern Software Workloads with Ramulator
- Accelerating Genome Analysis with FPGAs, GPUs, and New Execution Paradigms
- Genome Sequencing on Mobile Devices
- Exploring the Processing-in-Memory Paradigm for Future Computing Systems
- Hands-on Acceleration on Heterogeneous Computing Systems
- Understanding and Designing Modern NAND Flash-Based Solid-State Drives (SSDs) by Building a Practical SSD Simulator

https://safari.ethz.ch/projects_and_seminars/doku.php
More on the UPMEM PIM System
More on Analysis of the UPMEM PIM Engine

Inter-DPU Communication

- There is no direct communication channel between DPUs

- Inter-DPU communication takes place via the host CPU using CPU-DPU and DPU-CPU transfers

- Example communication patterns:
  - Merging of partial results to obtain the final result
  - Only DPU-CPU transfers
  - Redistribution of intermediate results for further computation
  - DPU-CPU transfers and CPU-DPU transfers

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

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

Dr. Juan Gómez-Luna, SAFARI Research Group, D-TTET, ETH Zurich

https://www.youtube.com/watch?v=D8Hjy2iU9j4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9
SAFARI Live Seminars (Past Talks)

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

DRAM non-cumulative memory

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

Near-Data Processing (2/2)

SAFARI Live Seminars in Computer Architecture
Gennady Pavelchenko, 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: Mechanics in Modern Microprocessors and Their Security Implications

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

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

SAFARI Live Seminars in Computer Architecture
Christine Giannoulis, National Technical University of Athens
Efficient Synchronization Support for Near-Data-Processing Architectures

Overview of a Modern SoC Architecture

- 3 domains in modern thermally-constrained mobile SoC: Compute, Memory, ID
- Several voltage sources exist, and some of them are shared between domains
- ID controllers and engines, ID interconnect, memory controller, and DRAM, typically each has an independent clock

SAFARI Live Seminars in Computer Architecture

https://safari.ethz.ch/safari-seminar-series/
SAFARI Live Seminars (Upcoming Talk)

SAFARI Live Seminar: Security Implications of Power Management... IChannels
Exploiting Current Management Mechanisms to Create Covert Channels in Modern Processors

Jawad Haj-Yahya
Jeremie S. Kim  A. Giray Yaşlıkçı  Ivan Puddu  Lois Orosa
Juan Gómez Luna  Mohammed Alser  Onur Mutlu

SAFARI Live Seminar: Jawad Haj-Yahya 4 October 2021
Posted on September 18, 2021 by event

Join us for our SAFARI Live Seminar with Jawad Haj-Yahya.

Monday, October 4 at 5:30 pm Zurich time (CEST)


Jawad Haj-Yahya, Huawei Research Center Zurich

https://safari.ethz.ch/safari-seminar-series/
Open-Source Artifacts

https://github.com/CMU-SAFARI
Open Source Tools: SAFARI GitHub

SAFARI Research Group at ETH Zurich and Carnegie Mellon University
Site for source code and tools distribution from SAFARI Research Group at ETH Zurich and Carnegie Mellon University.

ETH Zurich and Carnegie Mellon ... https://safari.ethz.ch/ omutlu@gmail.com

Pinned

ramulator
A Fast and Extensible DRAM Simulator, with built-in support for modeling many different DRAM technologies including DDRx, LPDDRx, GDDRx, WIOx, HBMx, and various academic proposals. Described in the...

Public 

prim-benchmarks
PrIM (Processing-In-Memory benchmarks) is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publi...

Public

DAMOV
DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processin...

Public

Repositories

Find a repository...

Type 
Language 
Sort 

New

Pythia
A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning.

C++ 0 1 0 Updated yesterday

BurstLink

0 0 0 0 Updated 21 days ago

https://github.com/CMU-SAFARI/
Some Open Source Tools (I)

- **Rowhammer – Program to Induce RowHammer Errors**
  - [https://github.com/CMU-SAFARI/rowhammer](https://github.com/CMU-SAFARI/rowhammer)
- **Ramulator – Fast and Extensible DRAM Simulator**
  - [https://github.com/CMU-SAFARI/ramulator](https://github.com/CMU-SAFARI/ramulator)
- **MemSim – Simple Memory Simulator**
  - [https://github.com/CMU-SAFARI/memsim](https://github.com/CMU-SAFARI/memsim)
- **NOCulator – Flexible Network-on-Chip Simulator**
  - [https://github.com/CMU-SAFARI/NOCulator](https://github.com/CMU-SAFARI/NOCulator)
- **SoftMC – FPGA-Based DRAM Testing Infrastructure**
  - [https://github.com/CMU-SAFARI/SoftMC](https://github.com/CMU-SAFARI/SoftMC)

- **Other open-source software from my group**
  - [https://github.com/CMU-SAFARI/](https://github.com/CMU-SAFARI/)
Some Open Source Tools (II)

- MQSim – A Fast Modern SSD Simulator
  - https://github.com/CMU-SAFARI/MQSim
- Mosaic – GPU Simulator Supporting Concurrent Applications
  - https://github.com/CMU-SAFARI/Mosaic
- IMPICA – Processing in 3D-Stacked Memory Simulator
  - https://github.com/CMU-SAFARI/IMPICA
- SMLA – Detailed 3D-Stacked Memory Simulator
  - https://github.com/CMU-SAFARI/SMLA
- HWASim – Simulator for Heterogeneous CPU-HWA Systems
  - https://github.com/CMU-SAFARI/HWASim

- Other open-source software from my group
  - https://github.com/CMU-SAFARI/
**Pythia**
A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning.
- machine-learning
- reinforcement-learning
- prefetcher
- cache-replacement
- branch-predictor
- chmpsi-simulator
- chmpsi-tracer

- C++
- MIT
- 1
- 6
- 0
- 0
- Updated yesterday

**BurstLink**

- 1
- 0
- 0
- 0
- Updated 21 days ago

**MIG-7-PHY-DDR3-Controller**
A DDR3 Controller that uses the Xilinx MIG-7 PHY to interface with DDR3 devices.
- Verilog
- 1
- 1
- 0
- 0
- Updated on Aug 22

**Pythia-HDL**
Implementation of Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning in Chisel HDL.
- machine-learning
- scala
- reinforcement-learning
- chisel
- chisel3
- frrt
- hdl

- Scala
- MIT
- 0
- 0
- 0
- 0
- Updated on Jul 31

**HARP**
Private
- 0
- 0
- 0
- 0
- Updated on Jul 31

**EINSim**
- simulator
- reliability
- statistical-inference
- dram
- error-correcting-codes
- map-estimation
- error-correction

- C++
- MIT
- 0
- 5
- 0
- 0
- Updated on Jul 29

**DAMOV**
DAMOV is a benchmark suite and a methodological framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing. Described by Oliveira et al. (preliminary version at https://arxiv.org/pdf/2105.03725.pdf).

- C++
- 1
- 12
- 0
- 0
- Updated on Jul 13

**MetaSys**
Metasys is the first open-source FPGA-based infrastructure with a prototype in a RISC-V core, to enable the rapid implementation and evaluation of a wide range of cross-layer software/hardware cooperative techniques in real hardware. Described in our pre-print: https://arxiv.org/abs/2105.08123

- C++
- 1
- 0
- 0
- 0
- Updated on Jul 9

**NATSA**
NATSA is the first near-data-processing accelerator for time series analysis based on the Matrix Profile (SCMP) algorithm. NATSA exploits modern 3D-stacked High Bandwidth Memory (HBM) to enable efficient and fast matrix profile computation near memory. Described in ICCD 2020 by Fernandez et al. https://people.inf.ethz.ch/omutlu/pub/NATSA_time...
- accelerator
- hbm
- time-series-analysis
- matrix-profile
- near-data-processing

- C++
- 1
- 0
- 0
- 0
- Updated on Jun 28

**COVIDHunter**
COVIDHunter is an accurate and flexible COVID-19 outbreak simulation model that forecasts the strength of future mitigation measures and the numbers of cases, hospitalizations, and deaths for a given day, while considering the potential effect of environmental conditions. Described by Alser et al. (preliminary version at https://arxiv.org/abs/22...
- simulation
- epidemiology
- covid-19
- covid-19-data
- covid-19-tracker
- reproduction-number
- covidhunter

- Swift
- MIT
- 1
- 5
- 0
- 0
- Updated on Jun 27

**prim-benchmarks**
Prim (Processing-In-Memory benchmarks) is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. Prim is developed to evaluate, analyze, and characterize the first publicly-available real-world PIM architecture, the UPMEM PIM architecture. Described by Gómez-Luna et al. (preliminary version at https://arxiv.org/abs/22...

- C
- MIT
- 0
- 0
- 0
- 0
- Updated on Jun 16

**SNP-Selective-Hiding**
An optimization-based mechanism to selectively hide the minimum number of overlapping SNPs among the family members who participated in the genomic studies (i.e. GWAS). Our goal is to disturb the dependencies among the family members in the original database for achieving better privacy without significantly degrading the data utility.
- gwas
- genomics
- data-privacy
- differential-privacy
- genomic-data-analysis
- laplace-distribution
- genomic-privacy

- MATLAB
- 0
- 0
- 0
- 0
- Updated on Jun 16

**SneakySnake**
SneakySnake is the first and the only pre-alignment filtering algorithm that works efficiently and fast on modern CPU, FPGA, and GPU architectures. It greatly (by more than two orders of magnitude) expedites sequence alignment calculation for both short and long reads. Described in the Bioinformatics (2020) by Alser et al. https://arxiv.org/abs/...

- fpca
- gpus
- smith-waterman
- Needleman-Wunsch
- sequence-alignment
- long-reads
- minimap2

- VHDL
- GPL-3.0
- 0
- 0
- 0
- 0
- Updated on May 12

**ramulator**
A Fast and Extensible DRAM Simulator, with built-in support for modeling many different DRAM technologies including DDRx, LPDDRx, GDDRx, WiOx, HBMx, and various academic proposals. Described in the IEEE CAL 2015 paper by Kim et al. at http://users.ece.cmu.edu/~omutlu/pub/ramulator_dram_simulator-ieee-cal15.pdf

- C++
- MIT
- 130
- 0
- 250
- 49
- 4
- Updated on May 11

**GenASIM**
- approximate-string-matching
- read-mapping
- hw-sw-co-design
- read-alignment
- bitmap-algorithm
- pre-alignment-filtering
- genome-sequence-analysis

- C
- GPL-3.0
- 0
- 20
- 0
- 0
- Updated on Mar 22

**AirLift**
AirLift is a tool that updates mapped reads from one reference genome to another. Unlike existing tools, it can do this in an efficient manner.

- 0
- 0
- 0
- 0
- Updated on Aug 8
An Interview on Research and Education

- **Computing Research and Education (@ ISCA 2019)**
  - [https://www.youtube.com/watch?v=8ffSEKZhmv0&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz](https://www.youtube.com/watch?v=8ffSEKZhmv0&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz)

- **Maurice Wilkes Award Speech (10 minutes)**
  - [https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15](https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15)
More Thoughts and Suggestions

- Onur Mutlu, 
  "Some Reflections (on DRAM)"
  Award Speech for ACM SIGARCH Maurice Wilkes Award, at the ISCA Awards Ceremony, Phoenix, AZ, USA, 25 June 2019.
  [Slides (pptx) (pdf)]
  [Video of Award Acceptance Speech (Youtube; 10 minutes) (Youku; 13 minutes)]
  [Video of Interview after Award Acceptance (Youtube; 1 hour 6 minutes) (Youku; 1 hour 6 minutes)]
  [News Article on "ACM SIGARCH Maurice Wilkes Award goes to Prof. Onur Mutlu"]

- Onur Mutlu,
  "How to Build an Impactful Research Group"
  57th Design Automation Conference Early Career Workshop (DAC), Virtual, 19 July 2020.
  [Slides (pptx) (pdf)]
More Thoughts and Suggestions (II)

- Onur Mutlu, "Computer Architecture: Why Is It So Important and Exciting Today?"
  Invited Lecture at Izmir Institute of Technology (IYTE), Virtual, 16 October 2020.
  [Slides (pptx) (pdf)]
  [Talk Video (2 hours 12 minutes)]

- Onur Mutlu, "Applying to Graduate School & Doing Impactful Research"
  Invited Panel Talk at the 3rd Undergraduate Mentoring Workshop, held with the 48th International Symposium on Computer Architecture (ISCA), Virtual, 18 June 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (50 minutes)]
A Talk on Impactful Research & Teaching

Applying to Grad School & Doing Impactful Research

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
13 June 2020
Undergraduate Architecture Mentoring Workshop @ ISCA 2021

SAFARI
ETH zürich
Carnegie Mellon

Arch. Mentoring Workshop @ISCA'21 - Applying to Grad School & Doing Impactful Research - Onur Mutlu
1,563 views • Premiered Jun 16, 2021

Onur Mutlu Lectures
17.2K subscribers

Panel talk at Undergraduate Architecture Mentoring Workshop at ISCA 2021
(https://sites.google.com/wisc.edu/uar...)

https://www.youtube.com/watch?v=83tlorht7Mc&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=54
Papers, Talks, Videos, Artifacts

- All are available at

  https://people.inf.ethz.ch/omutlu/projects.htm

  http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en

  https://www.youtube.com/onurmutlulectures

  https://github.com/CMU-SAFARI/
End of Backup Slides