Computation in Memory
An Architectural Perspective

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
20 July 2020
DAC Tutorial on Toward New Era of Compute-in-Memory
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

Data → performance & energy bottleneck

In-memory Databases

Graph/Tree Processing

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
Data is Key for Future Workloads

The 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


[Preliminary arxiv.org version]
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 ▼

Oxford Nanopore MinION

Data → performance & energy bottleneck
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.

Perils of Processor-Centric Design

Most of the system is dedicated to storing and moving data
Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data $\rightarrow$ performance & energy bottleneck

VP9

Video Playback

Google’s video codec

VP9

Video Capture

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, 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"
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 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-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 & scalable 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
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?
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/

Memory Scaling Issues Are Real

- Onur Mutlu and Jeremie Kim, "RowHammer: A Retrospective"
  [Preliminary arXiv version]

RowHammer: A Retrospective

Onur Mutlu§‡, Jeremie S. Kim‡§
§ETH Zürich ‡Carnegie Mellon University
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

ANDY GREENBERG  SECURITY  08.31.16  7:00 AM

FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS
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...

Project Zero

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

Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn, 2015)
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)]
Key Takeaways from 1580 Chips

• Chips of newer DRAM technology nodes 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)]
[Talk Video (17 minutes)]
[Source Code]
[Web Article]

Best paper award.

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.
The Push from Circuits and Devices

Main Memory Needs
Intelligent Controllers
Data Retention in Memory [Liu et al., ISCA 2013]

- Retention Time Profile of DRAM looks like this:

  64-128ms

  >256ms

  128-256ms

Location dependent
Stored value pattern dependent
Time dependent
More on DRAM Refresh (I)

- Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu, "RAIDR: Retention-Aware Intelligent DRAM Refresh"

**RAIDR: Retention-Aware Intelligent DRAM Refresh**

Jamie Liu  Ben Jaiyen  Richard Veras  Onur Mutlu
Carnegie Mellon University
More on DRAM Refresh (II)


An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms

Jamie Liu*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
jamiel@alumni.cmu.edu

Ben Jaiyen*
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
bjaiyen@alumni.cmu.edu

Yoongu Kim
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
yoonguk@ece.cmu.edu

Chris Wilkerson
Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95054
chris.wilkerson@intel.com

Onur Mutlu
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213
onur@cmu.edu
More on DRAM Refresh (III)

- Samira Khan, Donghyuk Lee, Yoongu Kim, Alaa Alameldeen, Chris Wilkerson, and Onur Mutlu,

"The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study"

The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study

Samira Khan†*
samirakhan@cmu.edu

Donghyuk Lee†
donghyuk1@cmu.edu

Yoongu Kim†
yoongukim@cmu.edu

Alaa R. Alameldeen* alaa.r.alameldeen@intel.com

Chris Wilkerson* chris.wilkerson@intel.com

Onur Mutlu† onur@cmu.edu

†Carnegie Mellon University  *Intel Labs

SAFARI
More on DRAM Refresh (IV)

- Moinuddin Qureshi, Dae Hyun Kim, Samira Khan, Prashant Nair, and Onur Mutlu, "AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems"

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

AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems

Moinuddin K. Qureshi† Dae-Hyun Kim† Samira Khan‡ Prashant J. Nair† Onur Mutlu‡
†Georgia Institute of Technology
‡Carnegie Mellon University
{moin, dhkim, pnair6}@ece.gatech.edu
{samirakhan, onur}@cmu.edu
More on DRAM Refresh (V)

- Samira Khan, Donghyuk Lee, and Onur Mutlu, "PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM"
  Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Toulouse, France, June 2016. [Slides (pptx) (pdf)]

PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM

Samira Khan* Donghyuk Lee†‡ Onur Mutlu*†
*University of Virginia †Carnegie Mellon University ‡Nvidia *ETH Zürich
More on DRAM Refresh (VI)

- Samira Khan, Chris Wilkerson, Zhe Wang, Alaa R. Alameldeen, Donghyuk Lee, and Onur Mutlu,
  "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content"
  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)]

First experimental analysis of (mobile) LPDDR4 chips
Analyzes the complex tradeoff space of retention time profiling
Idea: enable fast and robust profiling at higher refresh intervals & temperatures
More on DRAM Refresh (VIII)

- Minesh Patel, Jeremie S. Kim, Hasan Hassan, and Onur Mutlu, "Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices"
  [Slides (pptx) (pdf)]
  [Talk Video (26 minutes)]
  [Full Talk Lecture (29 minutes)]
  [Source Code for EINSim, the Error Inference Simulator]
  Best paper award.

Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices

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

SAFARI
The Push from Circuits and Devices

Main Memory Needs

Intelligent Controllers
Industry Is Writing Papers About It, Too

DRAM Process Scaling Challenges

- **Refresh**
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance
  - Leakage current of cell access transistors increasing

- **tWR**
  - Contact resistance between the cell capacitor and access transistor increasing
  - On-current of the cell access transistor decreasing
  - Bit-line resistance increasing

- **VRT**
  - Occurring more frequently with cell capacitance decreasing
Call for Intelligent Memory Controllers

DRAM Process Scaling Challenges

- Refresh
  - Difficult to build high-aspect ratio cell capacitors decreasing cell capacitance

Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

Uksong Kang, Hak-soo Yu, Churoo Park, *Hongzhong Zheng, **John Halbert, **Kuljit Bains, SeongJin Jang, and Joo Sun Choi

Samsung Electronics, Hwasung, Korea / *Samsung Electronics, San Jose / **Intel
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 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 \( \rightarrow \) 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)

Data from Runahead Execution [HPCA 2003].

Onur Mutlu, Jared Stark, Chris Wilkerson, and Yale N. Patt, "Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors"

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
The Performance Perspective (Today)

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

The Performance Perspective (Today)

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

![Graph showing cache-bound cycles for different services](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
Most of the system is dedicated to storing and moving data.
The Energy Perspective

Communication Dominates Arithmetic

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


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 Ausavarungnirun\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
We Do Not Want to Move Data!

A memory access consumes $\sim 100-1000X$ 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?
Processing in Memory:
Two Approaches

1. Minimally changing memory chips
2. Exploiting 3D-stacked memory
Approach 1: Minimally Changing Memory

- DRAM has great capability to perform **bulk data movement and computation** internally with small changes
  - 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

SAFARI
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

Transfer row
Transfer row

Row Buffer (4 Kbytes)

DRAM subarray

Data Bus

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)]
Memory as an Accelerator

Memory similar to a “conventional” accelerator
In-Memory Bulk Bitwise Operations

- We can support **in-DRAM COPY, ZERO, AND, OR, NOT, MAJ**
- At low cost
- Using analog computation capability of DRAM
  - Idea: activating multiple rows performs computation
- **30-60X performance and energy improvement**

- **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

\[ \frac{1}{2}V_{DD} + \delta \]

Final State

\[ AB + BC + AC \]

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

In-DRAM Bulk Bitwise AND/OR Operation

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

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

1. RowClone A into D1
2. RowClone B into D2
3. RowClone R0 into D3
4. ACTIVATE D1,D2,D3
5. RowClone Result into C
In-DRAM NOT: Dual Contact Cell

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

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

Ambit vs. DDR3: Performance and Energy

Performance Improvement
Energy Reduction

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

>5.4-6.6X Performance Improvement

Performance: BitWeaving on Ambit

Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

More on In-DRAM Bulk AND/OR

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

"Fast Bulk Bitwise AND and OR in DRAM"


---

Fast Bulk Bitwise AND and OR in DRAM

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

*Carnegie Mellon University  †Intel Pittsburgh
More on In-DRAM Bitwise Operations

More on In-DRAM Bulk Bitwise Execution

Vivek Seshadri and Onur Mutlu,
"In-DRAM Bulk Bitwise Execution Engine"
[Preliminary arXiv version]

In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
RowClone & Bitwise Ops in Real DRAM Chips

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

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

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

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

Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories

Shuangchen Li\textsuperscript{1,*}, Cong Xu\textsuperscript{2}, Qiaosha Zou\textsuperscript{1,5}, Jishen Zhao\textsuperscript{3}, Yu Lu\textsuperscript{4}, and Yuan Xie\textsuperscript{1}

University of California, Santa Barbara\textsuperscript{1}, Hewlett Packard Labs\textsuperscript{2}
University of California, Santa Cruz\textsuperscript{3}, Qualcomm Inc.\textsuperscript{4}, Huawei Technologies Inc.\textsuperscript{5}
{shuangchenli, yuanxie}@ece.ucsb.edu\textsuperscript{1}
Processing in Memory: Two Approaches

1. Minimally changing memory chips
2. Exploiting 3D-stacked memory
Opportunity: 3D-Stacked Logic+Memory

Other “True 3D” technologies under development
# DRAM Landscape (circa 2015)

<table>
<thead>
<tr>
<th>Segment</th>
<th>DRAM Standards &amp; Architectures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commodity</td>
<td>DDR3 (2007) [14]; DDR4 (2012) [18]</td>
</tr>
<tr>
<td>Performance</td>
<td>eDRAM [28], [32]; RLDDRAM3 (2011) [29]</td>
</tr>
</tbody>
</table>

Table 1. Landscape of DRAM-based memory

Two Key Questions in 3D-Stacked PIM

- What are the performance and energy benefits of using 3D-stacked memory as a coarse-grained accelerator?
  - By changing the entire system
  - By performing simple function offloading

- What is the minimal processing-in-memory support we can provide?
  - With minimal changes to system and programming
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
  - 32 Cores
  - 128...

  Speedup

  +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
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

In-Order Core

DRAM Controller

NI

PF Buffer

MTP

Message Queue

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

- Logic
  - Crossbar Network

- Memory
  - In-Order Core

- Communications via Remote Function Calls
  - Message Queue
Tesseract System for Graph Processing

Host Processor

Memory-Mapped Accelerator Interface
Noncacheable, Physically Addressed

Crossbar Network

Logic

Memory

Prefetching

LP
PF Buffer
MTP

Message Queue

DRAM Controller

NI
Evaluated Systems

<table>
<thead>
<tr>
<th>DDR3-OoO</th>
<th>HMC-OoO</th>
<th>HMC-MC</th>
<th>Tesseract</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>32 Tesseract Cores</td>
</tr>
<tr>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
<td>128 In-Order 2GHz</td>
<td>8TB/s</td>
</tr>
<tr>
<td>8 OoO 4GHz</td>
<td>8 OoO 4GHz</td>
<td>128 In-Order 2GHz</td>
<td></td>
</tr>
<tr>
<td>102.4GB/s</td>
<td>640GB/s</td>
<td>640GB/s</td>
<td></td>
</tr>
</tbody>
</table>

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

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

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

Memory Bandwidth Consumption (TB/s)
Tesseract Graph Processing System Energy

> 8X Energy Reduction

HMC-OoO vs. Tesseract with Prefetching

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

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi,

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


[Slides (pdf)] [Lightning Session Slides (pdf)]
Two Key Questions in 3D-Stacked PIM

- What are the performance and energy benefits of using 3D-stacked memory as a coarse-grained accelerator?
  - By changing the entire system
  - By performing simple function offloading

- What is the minimal processing-in-memory support we can provide?
  - With minimal changes to system and programming
PIM on Mobile Devices


Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand¹ Rachata Ausavarunngirun¹ Aki Kuusela³ Allan Knies³
Saugata Ghose¹ Eric Shiu³ Rahul Thakur³ Parthasarathy Ranganathan³
Youngsok Kim²
Daehyun Kim⁴,³ Onur Mutlu⁵,¹

SAFARI
Consumer Devices

Consumer devices are everywhere!

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

- Chrome: Google’s web browser
- TensorFlow Mobile: Google’s machine learning framework
- VP9: Google’s video codec (used in YouTube)
  - Video Playback
  - Video Capture
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

2nd 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

Small embedded low-power core

PIM Core

Small fixed-function accelerators

PIM Accelerator

Offloading to PIM logic reduces energy and improves performance, on average, by 55.4% and 54.2%
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
Packing

Reorders elements of matrices to minimize cache misses during matrix multiplication

Up to 40% of the inference energy and 31% of inference execution time

Packing’s data movement accounts for up to 35.3% of the inference energy

A simple data reorganization process that requires simple arithmetic
Quantization

Converts 32-bit floating point to 8-bit integers to improve inference execution time and energy consumption.

- Up to 16.8% of the inference energy and 16.1% of inference execution time.
- Majority of quantization energy comes from data movement.

A simple data conversion operation that requires shift, addition, and multiplication operations.
Normalized Energy

PIM core and PIM accelerator reduce energy consumption on average by 49.1% and 55.4%
Offloading these kernels to **PIM core and PIM accelerator** improves **performance** on average by **44.6%** and **54.2%**
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 Boroumand\textsuperscript{1} Rachata Ausavarungnirun\textsuperscript{1} Aki Kuusela\textsuperscript{3} Allan Knies\textsuperscript{3}
Saugata Ghose\textsuperscript{1} Eric Shiu\textsuperscript{3} Rahul Thakur\textsuperscript{3}
Youngsok Kim\textsuperscript{2} Parthasarathy Ranganathan\textsuperscript{3} Daehyun Kim\textsuperscript{4,3} Onur Mutlu\textsuperscript{5,1}

SAFARI
Truly Distributed GPU Processing with PIM?

3D-stacked memory (memory stack)

Main GPU

SM (Streaming Multiprocessor)

Logic layer

Crossbar switch

Vault Ctrl

Vault Ctrl

--

```c
__global__
void applyScaleFactorsKernel( uint8_t * const out,
uint8_t const * const in, const double *factor,
size_t const numRows, size_t const numCols )
{
    // Work out which pixel we are working on.
    const int rowIdx = blockIdx.x * blockDim.x + threadIdx.x;
    const int colIdx = blockIdx.y;
    const int sliceIdx = threadIdx.z;

    // Check this thread isn't off the image
    if( rowIdx >= numRows ) return;

    // Compute the index of my element
    size_t linearIdx = rowIdx + colIdx*numRows +
        sliceIdx*numRows*numCols;
```
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 Pattnaik1  Xulong Tang1  Adwait Jog2  Onur Kayiran3
Asit K. Mishra4  Mahmut T. Kandemir1  Onur Mutlu5,6  Chita R. Das1

1Pennsylvania State University  2College of William and Mary
3Advanced Micro Devices, Inc.  4Intel Labs  5ETH Zürich  6Carnegie 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
Two Key Questions in 3D-Stacked PIM

- What are the performance and energy benefits of using 3D-stacked memory as a coarse-grained accelerator?
  - By changing the entire system
  - By performing simple function offloading

- What is the minimal processing-in-memory support we can provide?
  - With minimal changes to system and programming
PEI: PIM-Enabled Instructions (Ideas)

- **Goal:** Develop mechanisms to get the most out of near-data processing with minimal cost, minimal changes to the system, no changes to the programming model

- **Key Idea 1:** Expose each PIM operation as a cache-coherent, virtually-addressed host processor instruction (called PEI) that operates on only a single cache block
  - e.g., `__pim_add(&w.next_rank, value) → pim.add r1, (r2)`
  - No changes sequential execution/programming model
  - No changes to virtual memory
  - Minimal changes to cache coherence
  - No need for data mapping: Each PEI restricted to a single memory module

- **Key Idea 2:** Dynamically decide where to execute a PEI (i.e., the host processor or PIM accelerator) based on simple locality characteristics and simple hardware predictors
  - Execute each operation at the location that provides the best performance
Simple PIM Operations as ISA Extensions (II)

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

Host Processor

Main Memory

![Image showing the interaction between Host Processor and Main Memory](image_url)

Conventional Architecture

**64 bytes in**

**64 bytes out**
for (v: graph.vertices) {
    value = weight * v.rank;
    for (w: v.successors) {
        __pim_add(&w.next_rank, value);
    }
}

In-Memory Addition

8 bytes in
0 bytes out

Main Memory

Host Processor

value

w.next_rank

pim.add r1, (r2)
PEI: PIM-Enabled Instructions (Example)

```c
for (v: graph.vertices) {
    value = weight * v.rank;
    for (w: v.successors) {
        __pim_add(&w.next_rank, value);
    }
}
pfence();
```

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

Table 1: Summary of Supported PIM Operations

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

Host Processor
Out-Of-Order Core

PCU (PEI Computation Unit)

L1 Cache
L2 Cache

Last-Level Cache

HMC Controller

3D-stacked Memory

DRAM Controller

PCU

Network

PCU

PIM Directory

Locality Monitor

PMU (PEI Mgmt Unit)

Example PEI uArchitecture
PEI: Initial Evaluation Results

- Initial evaluations with **10 emerging data-intensive workloads**
  - Large-scale graph processing
  - In-memory data analytics
  - Machine learning and data mining
  - Three input sets (small, medium, large) for each workload to analyze the impact of data locality

- Pin-based cycle-level x86-64 simulation

**Performance Improvement and Energy Reduction:**
- 47% average speedup with large input data sets
- 32% speedup with small input data sets
- 25% avg. energy reduction in a single node with large input data sets

---

**Table 2: Baseline Simulation Configuration**

<table>
<thead>
<tr>
<th>Component</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>16 out-of-order cores, 4 GHz, 4-issue</td>
</tr>
<tr>
<td>L1 I/D-Cache</td>
<td>Private, 32 KB, 4/8-way, 64 B blocks, 16 MSHRs</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>Private, 256 KB, 8-way, 64 B blocks, 16 MSHRs</td>
</tr>
<tr>
<td>L3 Cache</td>
<td>Shared, 16 MB, 16-way, 64 B blocks, 64 MSHRs</td>
</tr>
<tr>
<td>On-Chip Network</td>
<td>Crossbar, 2 GHz, 144-bit links</td>
</tr>
<tr>
<td>Main Memory</td>
<td>32 GB, 8 HMCs, daisy-chain (80 GB/s full-duplex)</td>
</tr>
<tr>
<td>HMC</td>
<td>4 GB, 16 vaults, 256 DRAM banks [20]</td>
</tr>
<tr>
<td>–DRAM</td>
<td>FR-FCFS, tCL = tRCD = tRP = 13.75 ns [27]</td>
</tr>
<tr>
<td>–Vertical Links</td>
<td>64 TSVs per vault with 2 Gb/s signaling rate [23]</td>
</tr>
</tbody>
</table>
PEI Performance Delta: Large Data Sets

(Large Inputs, Baseline: Host-Only)

47% Performance Improvement
25% Energy Reduction

PEI Energy Consumption

- Host-Only
- PIM-Only
- Locality-Aware

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

Medium

Large

25% Energy Reduction
Simpler PIM: PIM-Enabled Instructions


PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture

Junwhan Ahn  Sungjoo Yoo  Onur Mutlu†  Kiyoungh Choi
junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr

Seoul National University  †Carnegie Mellon University

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

1. Functionality of and applications & software for PIM

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

3. System support: coherence & virtual memory

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

5. Infrastructures to assess benefits and feasibility

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

We can get there step by step
Onur Mutlu$^{a,b}$, Saugata Ghose$^b$, Juan Gómez-Luna$^a$, Rachata Ausavarungnirun$^{b,c}$

$^a$ETH Zürich
$^b$Carnegie Mellon University
$^c$King Mongkut’s University of Technology North Bangkok


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

Saugata Ghose† Amirali Boroumand† Jeremy 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

[Image of DRAM module]

Key Challenge 1: Code Mapping

- **Challenge 1:** Which operations should be executed in memory vs. in CPU?
Key Challenge 2: Data Mapping

- **Challenge 2**: How should data be mapped to different 3D memory stacks?
How to Do the Code and Data Mapping?


[Slides (pptx) (pdf)]
[Lightning Session Slides (pptx) (pdf)]
How to Schedule Code? (I)


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

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

\textsuperscript{1}Pennsylvania State University \textsuperscript{2}College of William and Mary \textsuperscript{3}Advanced Micro Devices, Inc. \textsuperscript{4}Intel Labs \textsuperscript{5}ETH Zürich \textsuperscript{6}Carnegie Mellon University
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
How to Schedule Code? (III)

- 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
How to Maintain Coherence? (I)

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,
  "LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory"

LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory

Amirali Boroumand†, Saugata Ghose†, Minesh Patel†, Hasan Hassan†§, Brandon Lucia†, Kevin Hsieh†, Krishna T. Malladi*, Hongzhong Zheng*, and Onur Mutlu††

†Carnegie Mellon University  *Samsung Semiconductor, Inc.  §TOBB ETÜ  ††ETH Zürich
How to Maintain Coherence? (II)

- Amirali Boroumand, Saugata Ghose, Minesh Patel, Hasan Hassan, Brandon Lucia, Kevin Hsieh, Krishna T. Malladi, Hongzhong Zheng, and Onur Mutlu,

"CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators"


CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

Amirali Boroumand†
Brandon Lucia†
Nastaran Hajinazar¢
Saugata Ghose†
Rachata Ausavarungnirun†‡
Krishna T. Malladi§
Minesh Patel*
Hasan Hassan*
Kevin Hsieh†
Hongzhong Zheng§
Onur Mutlu*†

†Carnegie Mellon University
¢Simon Fraser University
§Samsung Semiconductor, Inc.

*ETH Zürich
‡KMUTNB

SAFARI
How to Support Virtual Memory?

- 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
How to Design Data Structures for PIM?


Concurrent Data Structures for Near-Memory Computing

Zhiyu Liu  
Computer Science Department  
Brown University  
zhiyu.liu@brown.edu

Irina Calciu  
VMware Research Group  
icalciu@vmware.com

Maurice Herlihy  
Computer Science Department  
Brown University  
mph@cs.brown.edu

Onur Mutlu  
Computer Science Department  
ETH Zürich  
onur.mutlu@inf.ethz.ch
Simulation Infrastructures for PIM

- **Ramulator** extended for PIM
  - Flexible and extensible DRAM simulator
  - Can model many different memory standards and proposals
  - [Source Code for Ramulator-PIM](https://github.com/CMU-SAFARI/ramulator-pim)
  - [Source Code for Ramulator](https://github.com/CMU-SAFARI/ramulator)
Performance & Energy Models for PIM

- Gagandeep Singh, Juan Gomez-Luna, Giovanni Mariani, Geraldo F. Oliveira, Stefano Corda, Sander Stuijk, Onur Mutlu, and Henk Corporaal, "NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning"

Proceedings of the 56th Design Automation Conference (DAC), Las Vegas, NV, USA, June 2019.

[Slides (pptx) (pdf)]
[Poster (pptx) (pdf)]
[Source Code for Ramulator-PIM]

NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning

Gagandeep Singh\textsuperscript{a,c} \quad Juan Gómez-Luna\textsuperscript{b} \quad Giovanni Mariani\textsuperscript{c} \quad Geraldo F. Oliveira\textsuperscript{b}

Stefano Corda\textsuperscript{a,c} \quad Sander Stuijk\textsuperscript{a} \quad Onur Mutlu\textsuperscript{b} \quad Henk Corporaal\textsuperscript{a}

\textsuperscript{a}Eindhoven University of Technology
\textsuperscript{b}ETH Zürich
\textsuperscript{c}IBM Research - Zurich
Challenge and Opportunity for Future Computing Architectures with Minimal Data Movement
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 architectures
  - 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
  - ...
Architectures for Intelligent Machines

Data-centric

Data-driven

Data-aware
We Need to Revisit the Entire Stack

We can get there step by step
Computation in Memory
An Architectural Perspective

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
20 July 2020
DAC Tutorial on Toward New Era of Compute-in-Memory
Backup Slides
A Bit About Myself

Onur Mutlu

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

Research and Teaching in:

- Computer architecture, computer systems, hardware security, bioinformatics
- Memory and storage systems
- Hardware security, safety, predictability
- Fault tolerance
- Hardware/software cooperation
- Architectures for bioinformatics, health, medicine
- ...

...
Related Overview Talks

- Future Computing Architectures
  - https://www.youtube.com/watch?v=kqiZlSOcGFMy&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=hPnSmfwu2-A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJhnV2YFBJl&index=9

- 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=n8Aj_A0WSg8&list=PL5Q2soXY2Zi8D_5MGV6EnXEJhnV2YFBJl&index=22
Accelerated Memory Course (~6.5 hours)

- **ACACES 2018**
  - Memory Systems and Memory-Centric Computing Systems
  - Taught by Onur Mutlu July 9-13, 2018
  - ~6.5 hours of lectures

- **Website for the Course including Videos, Slides, Papers**
  - [https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpM JM05P6J9x](https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x)

- **All Papers are at:**
  - [https://people.inf.ethz.ch/omutlu/projects.htm](https://people.inf.ethz.ch/omutlu/projects.htm)
  - Final lecture notes and readings (for all topics)
An Interview on Research and Education

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

- Maurice Wilkes Award Speech (10 minutes)
  - https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15
Data-Centric Architectures: Properties

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

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

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

- **Intelligent data management**
  - Intelligent controllers handling robustness, security, cost, scaling
Low-Latency & Low-Energy Data Access
Main Memory Latency Lags Behind

Memory latency remains almost constant
A Closer Look …

Figure 1: DRAM latency trends over time [20, 21, 23, 51].

DRAM Latency Is Critical for Performance

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

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

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

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

In-memory Databases

Graph/Tree Processing

Long memory latency $\rightarrow$ performance bottleneck

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

Datacenter Workloads
[Kanev+ (Google), ISCA’15]
New DRAM Types Increase Latency!

- Saugata Ghose, Tianshi Li, Nastaran Hajinazar, Damla Senol Cali, and Onur Mutlu, "Demystifying Workload–DRAM Interactions: An Experimental Study"


[Abstract]
[Slides (pptx) (pdf)]

Demystifying Complex Workload–DRAM Interactions: An Experimental Study

Saugata Ghose† Tianshi Li† Nastaran Hajinazar‡†
Damla Senol Cali† Onur Mutlu§†

†Carnegie Mellon University ‡Simon Fraser University §ETH Zürich
The Memory Latency Problem

- High memory latency is a significant limiter of system performance and energy-efficiency.

- It is becoming increasingly so with higher memory contention in multi-core and heterogeneous architectures:
  - Exacerbating the bandwidth need
  - Exacerbating the QoS problem

- It increases processor design complexity due to the mechanisms incorporated to tolerate memory latency.
Retrospective: Conventional Latency Tolerance Techniques

- Caching [initially by Wilkes, 1965]
  - Widely used, simple, effective, but inefficient, passive
  - Not all applications/phases exhibit temporal or spatial locality

- Prefetching [initially in IBM 360/91, 1967]
  - Works well for regular memory access patterns
  - Prefetching irregular access patterns is difficult, inaccurate, and hardware-intensive

- Multithreading [initially in CDC 6600, 1964]
  - Works well if there are multiple threads
  - Improving single thread performance using multithreading hardware is an ongoing research effort

- Out-of-order execution [initially by Tomasulo, 1967]
  - Tolerates cache misses that cannot be prefetched
  - Requires extensive hardware resources for tolerating long latencies

None of These Fundamentally Reduce Memory Latency
Two Major Sources of Latency Inefficiency

- Modern DRAM is **not** designed for low latency
  - Main focus is cost-per-bit (capacity)

- Modern DRAM latency is determined by **worst case** conditions and **worst case** devices
  - Much of memory latency is unnecessary

**Our Goal:** Reduce Memory Latency at the Source of the Problem
Why is Memory Latency High?

- **DRAM latency**: Delay as specified in DRAM standards
  - Doesn’t reflect true DRAM device latency
- Imperfect manufacturing process $\rightarrow$ latency variation
- **High standard latency** chosen to increase yield

![Diagram showing DRAM latency variations](image)

Manufacturing Variation

**Low** $\rightarrow$ **High**

**DRAM Latency**

**Standard Latency**
Adaptive-Latency DRAM

• **Key idea**
  – Optimize DRAM timing parameters online

• **Two components**
  – DRAM manufacturer provides multiple sets of reliable DRAM timing parameters at different temperatures for each DIMM
  – System monitors DRAM temperature & uses appropriate DRAM timing parameters

Infrastructures to Understand Such Issues

**SoftMC: Open Source DRAM Infrastructure**


- **Flexible**
- **Easy to Use (C++ API)**
- **Open-source**
  
  [github.com/CMU-SAFARI/SoftMC](https://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
Latency Reduction Summary of 115 DIMMs

• *Latency reduction for read & write (55°C)*
  – Read Latency: **32.7%**
  – Write Latency: **55.1%**

• *Latency reduction for each timing parameter (55°C)*
  – Sensing: **17.3%**
  – Restore: **37.3%** (read), **54.8%** (write)
  – Precharge: **35.2%**
AL-DRAM: Real-System Performance

Performance Improvement

Single Core  Multi Core

soplex  mcf  milc  libq  lbm  gems  copy  s.cluster  gups

Average Improvement

Non-intensive   Intensive   All-35-workload

14.0%  10.4%  2.9%

AL-DRAM provides high performance on memory-intensive workloads
Reducing Latency Also Reduces Energy

- AL-DRAM reduces DRAM power consumption
- Major reason: reduction in row activation time
More on Adaptive-Latency DRAM

- Donghyuk Lee, Yoongu Kim, Gennady Pekhimenko, Samira Khan, Vivek Seshadri, Kevin Chang, and Onur Mutlu,

"Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case"


[Slides (pptx) (pdf)] [Full data sets]

Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case

Donghyuk Lee  Yoongu Kim  Gennady Pekhimenko
Samira Khan  Vivek Seshadri  Kevin Chang  Onur Mutlu

Carnegie Mellon University
Tackling the Fixed Latency Mindset

- Reliable operation latency is actually very heterogeneous
  - Across temperatures, chips, parts of a chip, voltage levels, ...

- Idea: Dynamically find out and use the lowest latency one can reliably access a memory location with
  - Adaptive-Latency DRAM [HPCA 2015]
  - Flexible-Latency DRAM [SIGMETRICS 2016]
  - Design-Induced Variation-Aware DRAM [SIGMETRICS 2017]
  - Voltron [SIGMETRICS 2017]
  - DRAM Latency PUF [HPCA 2018]
  - DRAM Latency True Random Number Generator [HPCA 2019]
  - ...

- We would like to find sources of latency heterogeneity and exploit them to minimize latency (or create other benefits)
Analysis of Latency Variation in DRAM Chips

Kevin Chang, Abhijith Kashyap, Hasan Hassan, Samira Khan, Kevin Hsieh, Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Tianshi Li, and Onur Mutlu,
"Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization"
[Slides (pptx) (pdf)]
[Source Code]
Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms

Donghyuk Lee, NVIDIA and Carnegie Mellon University
Samira Khan, University of Virginia
Lavanya Subramanian, Saugata Ghose, Rachata Ausavarungnirun, Carnegie Mellon University
Gennady Pekhimenko, Vivek Seshadri, Microsoft Research
Onur Mutlu, ETH Zürich and Carnegie Mellon University
Solar-DRAM: Exploiting Spatial Variation

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu, "Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines"  

Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines

Jeremie S. Kim‡§ Minesh Patel§ Hasan Hassan§ Onur Mutlu§†  
‡Carnegie Mellon University §ETH Zürich
DRAM Latency PUFs

Jeremie S. Kim, Minesh Patel, Hasan Hassan, and Onur Mutlu, "The DRAM Latency PUF: Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern DRAM Devices"
[Lightning Talk Video]
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]

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
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
ChargeCache: Exploiting Access Patterns

- Hasan Hassan, Gennady Pekhimenko, Nandita Vijaykumar, Vivek Seshadri, Donghyuk Lee, Oguz Ergin, and Onur Mutlu,

"ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality"


[Slides (pptx) (pdf)]
[Source Code]

ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality

Hasan Hassan†*, Gennady Pekhimenko†, Nandita Vijaykumar†
Vivek Seshadri†, Donghyuk Lee†, Oguz Ergin*, Onur Mutlu†

†Carnegie Mellon University    *TOBB University of Economics & Technology
Exploiting Subarray Level Parallelism

- Yoongu Kim, Vivek Seshadri, Donghyuk Lee, Jamie Liu, and Onur Mutlu,
"A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM"
Proceedings of the 39th International Symposium on Computer Architecture (ISCA), Portland, OR, June 2012. Slides (pptx)
Tiered-Latency DRAM

Donghyuk Lee, Yoongu Kim, Vivek Seshadri, Jamie Liu, Lavanya Subramanian, and Onur Mutlu,
"Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture"
Proceedings of the 19th International Symposium on High-Performance Computer Architecture (HPCA), Shenzhen, China, February 2013. Slides (pptx)
LISA: Low-cost Inter-linked Subarrays

Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,
"Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"
[Slides (pptx) (pdf)]
[Source Code]

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

SAFARI 193
The CROW Substrate for DRAM

- Hasan Hassan, Minesh Patel, Jeremie S. Kim, A. Giray Yaglikci, Nandita Vijaykumar, Nika Mansourighiasi, Saugata Ghose, and Onur Mutlu,

"CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability"

CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability

Hasan Hassan†  Minesh Patel†  Jeremie S. Kim†§  A. Giray Yaglikci†
Nandita Vijaykumar†§  Nika Mansouri Ghiasi†  Saugata Ghose§  Onur Mutlu†§

†ETH Zürich  §Carnegie Mellon University
Reducing Refresh Latency

- Anup Das, Hasan Hassan, and Onur Mutlu, "VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency"

Proceedings of the 55th Design Automation Conference (DAC), San Francisco, CA, USA, June 2018.

VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency

Anup Das
Drexel University
Philadelphia, PA, USA
anup.das@drexel.edu

Hasan Hassan
ETH Zürich
Zürich, Switzerland
hhasan@ethz.ch

Onur Mutlu
ETH Zürich
Zürich, Switzerland
omutlu@gmail.com
Parallelizing Refreshes and Accesses

Kevin Chang, Donghyuk Lee, Zeshan Chishti, Alaa Alameldeen, Chris Wilkerson, Yoongu Kim, and Onur Mutlu,
"Improving DRAM Performance by Parallelizing Refreshes with Accesses"
[Summary] [Slides (pptx) (pdf)]

Reducing Performance Impact of DRAM Refresh by Parallelizing Refreshes with Accesses

Kevin Kai-Wei Chang  Donghyuk Lee  Zeshan Chishti†
Alaa R. Alameldeen†  Chris Wilkerson†  Yoongu Kim  Onur Mutlu
Carnegie Mellon University  †Intel Labs
Eliminating Refreshes

- Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu, "RAIDR: Retention-Aware Intelligent Intelligent DRAM Refresh"
  Slides (pdf)

**RAIDR: Retention-Aware Intelligent Intelligent DRAM Refresh**

Jamie Liu   Ben Jaiyen   Richard Veras   Onur Mutlu
Carnegie Mellon University
Analysis of Latency-Voltage in DRAM Chips

Kevin Chang, A. Giray Yaglikci, Saugata Ghose, Aditya Agrawal, Niladrish Chatterjee, Abhijith Kashyap, Donghyuk Lee, Mike O'Connor, Hasan Hassan, and Onur Mutlu,
"Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms"

Understanding Reduced-Voltage Operation in Modern DRAM Chips: Characterization, Analysis, and Mechanisms

Kevin K. Chang† Abdullah Giray Yaglıkçı† Saugata Ghose† Aditya Agrawal‡ Niladrish Chatterjee‡
Abhijith Kashyap† Donghyuk Lee‡ Mike O’Connor‡,† Hasan Hassan§ Onur Mutlu§,†

†Carnegie Mellon University ‡NVIDIA †The University of Texas at Austin §ETH Zürich
VAMPIRE DRAM Power Model

- Saugata Ghose, A. Giray Yaglikci, Raghav Gupta, Donghyuk Lee, Kais Kudrolli, William X. Liu, Hasan Hassan, Kevin K. Chang, Niladrish Chatterjee, Aditya Agrawal, Mike O'Connor, and Onur Mutlu,

"What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study"


[Abstract]

What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study

Saugata Ghose† Abdullah Giray Yağlıkçı‡† Raghav Gupta† Donghyuk Lee§
Kais Kudrolli† William X. Liu† Hasan Hassan‡ Kevin K. Chang†
Niladrish Chatterjee§ Aditya Agrawal§ Mike O’Connor§‖ University of Texas at Austin

†Carnegie Mellon University ‡ETH Zürich §NVIDIA ‖University of Texas at Austin
Takeaway I

We Can Reduce Memory Latency with Change of Mindset
Takeaway II

Main Memory Needs Intelligent Controllers to Reduce Latency
Data-Driven and 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
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
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
  - \( t_{WTR} \): Minimum number of cycles to wait before issuing a read command after a write command is issued
  - \( t_{RC} \): 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


<table>
<thead>
<tr>
<th>Latency</th>
<th>Symbol</th>
<th>DRAM cycles</th>
<th>Latency</th>
<th>Symbol</th>
<th>DRAM cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precharge</td>
<td>( t_{RP} )</td>
<td>11</td>
<td>Activate to read/write</td>
<td>( t_{RCD} )</td>
<td>11</td>
</tr>
<tr>
<td>Read column address strobe</td>
<td>( CL )</td>
<td>11</td>
<td>Write column address strobe</td>
<td>( CWL )</td>
<td>8</td>
</tr>
<tr>
<td>Additive</td>
<td>( AL )</td>
<td>0</td>
<td>Activate to activate</td>
<td>( RC )</td>
<td>39</td>
</tr>
<tr>
<td>Activate to precharge</td>
<td>( t_{RAS} )</td>
<td>28</td>
<td>Read to precharge</td>
<td>( RTP )</td>
<td>6</td>
</tr>
<tr>
<td>Burst length</td>
<td>( t_{BL} )</td>
<td>4</td>
<td>Column address strobe to column address strobe</td>
<td>( CCD )</td>
<td>4</td>
</tr>
<tr>
<td>Activate to activate (different bank)</td>
<td>( t_{RRD} )</td>
<td>6</td>
<td>Four activate windows</td>
<td>( FAW )</td>
<td>24</td>
</tr>
<tr>
<td>Write to read</td>
<td>( t_{WTR} )</td>
<td>6</td>
<td>Write recovery</td>
<td>( WR )</td>
<td>12</td>
</tr>
</tbody>
</table>

Table 4. DDR3 1600 DRAM timing specifications
Many Memory Timing Constraints


Table 2. Timing Constraints (DDR3-1066) [43]

<table>
<thead>
<tr>
<th>Phase</th>
<th>Commands</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACT → READ</td>
<td>tRC</td>
<td>15ns</td>
</tr>
<tr>
<td></td>
<td>ACT → WRITE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACT → PRE</td>
<td>tRAS</td>
<td>37.5ns</td>
</tr>
<tr>
<td>2</td>
<td>READ → data</td>
<td>tCL</td>
<td>15ns</td>
</tr>
<tr>
<td></td>
<td>WRITE → data</td>
<td>tCWL</td>
<td>11.25ns</td>
</tr>
<tr>
<td></td>
<td>data burst</td>
<td>tBL</td>
<td>7.5ns</td>
</tr>
<tr>
<td>3</td>
<td>PRE → ACT</td>
<td>tRP</td>
<td>15ns</td>
</tr>
<tr>
<td>1 &amp; 3</td>
<td>ACT → ACT</td>
<td>tRC</td>
<td>52.5ns</td>
</tr>
<tr>
<td></td>
<td>(tRAS+tRP)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5. Three Phases of DRAM Access
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.
Goal: Learn to choose actions to maximize $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \ (0 \leq \gamma < 1)$
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
Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana, "Self Optimizing Memory Controllers: A Reinforcement Learning Approach"


Figure 4: High-level overview of an RL-based scheduler.
<table>
<thead>
<tr>
<th>Reward function</th>
<th>State attributes</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1 for scheduling Read and Write commands</td>
<td>Number of reads, writes, and load misses in transaction queue</td>
<td>Activate</td>
</tr>
<tr>
<td>0 at all other times</td>
<td>Number of pending writes and ROB heads waiting for referenced row</td>
<td>Write</td>
</tr>
<tr>
<td>Goal is to maximize long-term data bus utilization</td>
<td>Request’s relative ROB order</td>
<td>Read - load miss</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Read - store miss</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Precharge - pending</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Precharge - preemptive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOP</td>
</tr>
</tbody>
</table>
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"
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

Self-Optimizing (Data-Driven) Computing 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 X
  - Access Semantics
  - ...

One Problem: Limited Interfaces

Higher-level information is not visible to HW

Software

Hardware

100011111...
101010011...

Instructions
Memory Addresses

Data Types: Integer, Float, Char

Data Structures

Code Optimizations

Access Patterns

A → B → C
D → E → F

226
A Solution: More Expressive Interfaces

Performance
Software

Functionality

Expressive Memory “XMem”

Higher-level Program Semantics

ISA Virtual Memory

Hardware

wiseGEEK
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  +Simon Fraser University  §ETH Zürich  

‡NVIDIA
## X-MeM Aids 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 <em>”hotness”</em> 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 for GPUs

Nandita Vijaykumar, Eiman Ebrahimi, Kevin Hsieh, Phillip B. Gibbons and Onur Mutlu, "The Locality Descriptor: A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs"


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

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
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]

Characterizing Application Memory Error Vulnerability to Optimize Datacenter Cost via Heterogeneous-Reliability Memory

Yixin Luo* Sriram Govindan* Bikash Sharma* Mark Santaniello* Justin Meza Aman Kansal* Jie Liu* Badriddine Khessib* Kushagra Vaid* Onur Mutlu Carnegie Mellon University, yixinluo@cs.cmu.edu, {meza, onur}@cmu.edu

*Microsoft Corporation, {srgovin, bsharma, marksan, kansal, jie.liu, bkhessib, kvaid}@microsoft.com

SAFARI
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]
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: Data-Aware Efficient DNN Inference

- Skanda Koppula, Lois Orosa, A. Giray Yaglıkçı, 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)]
Challenge and Opportunity for Future Data-Aware (Expressive) Computing Architectures
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 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
Readings, Videos, Reference Materials
Some Overview Talks

https://www.youtube.com/onurmutlulectures

- **Future Computing Architectures**
  - [https://www.youtube.com/watch?v=kqiZISOcGFM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=1](https://www.youtube.com/watch?v=kqiZISOcGFM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=1)

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

- **Accelerating Genome Analysis**
  - [https://www.youtube.com/watch?v=hPnSmfwu2-A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=9](https://www.youtube.com/watch?v=hPnSmfwu2-A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=9)

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

- **Intelligent Architectures for Intelligent Machines**
  - [https://www.youtube.com/watch?v=n8Aj_A0WSq8&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=22](https://www.youtube.com/watch?v=n8Aj_A0WSq8&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=22)
Accelerated Memory Course (~6.5 hours)

- **ACACES 2018**
  - Memory Systems and Memory-Centric Computing Systems
  - Taught by Onur Mutlu July 9-13, 2018
  - ~6.5 hours of lectures

- **Website for the Course including Videos, Slides, Papers**
  - [https://safari.ethz.ch/memory_systems/ACACES2018/](https://safari.ethz.ch/memory_systems/ACACES2018/)
  - [https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x](https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x)

- **All Papers are at:**
  - [https://people.inf.ethz.ch/omutlu/projects.htm](https://people.inf.ethz.ch/omutlu/projects.htm)
  - Final lecture notes and readings (for all topics)
Longer Memory Course (~18 hours)

- **Tu Wien 2019**
  - Memory Systems and Memory-Centric Computing Systems
  - Taught by Onur Mutlu June 12-19, 2019
  - ~18 hours of lectures

- **Website for the Course including Videos, Slides, Papers**
  - [https://www.youtube.com/playlist?list=PL5Q2soXY2Zi_gntM55VoMlKlw7YrXOhbl](https://www.youtube.com/playlist?list=PL5Q2soXY2Zi_gntM55VoMlKlw7YrXOhbl)

- **All Papers are at:**
  - [https://people.inf.ethz.ch/omutlu/projects.htm](https://people.inf.ethz.ch/omutlu/projects.htm)
  - Final lecture notes and readings (for all topics)
Processing Data Where It Makes Sense: Enabling In-Memory Computation

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

\textsuperscript{a}ETH Zürich
\textsuperscript{b}Carnegie Mellon University
\textsuperscript{c}King Mongkut’s University of Technology North Bangkok

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun, "\textit{Processing Data Where It Makes Sense: Enabling In-Memory Computation}"

\textit{Invited paper in Microprocessors and Microsystems (MICPRO), June 2019.} [arXiv version]
Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions

SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND, RACHATA AUSAVARUNGNIRUN
Carnegie Mellon University

ONUR MUTLU
ETH Zürich and Carnegie Mellon University

[Preliminary arxiv.org version]
Onur Mutlu and Lavanya Subramanian, "Research Problems and Opportunities in Memory Systems" 
Invited Article in Supercomputing Frontiers and Innovations (SUPERFRI), 2014/2015.

Research Problems and Opportunities in Memory Systems

Onur Mutlu¹, Lavanya Subramanian¹

Onur Mutlu,
"The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
[Slides (pptx) (pdf)]

---

The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
https://people.inf.ethz.ch/omutlu

Onur Mutlu,
"Memory Scaling: A Systems Architecture Perspective"
Technical talk at MemCon 2013 (MEMCON), Santa Clara, CA, August 2013. [Slides (pptx) (pdf)] [Video] [Coverage on StorageSearch]
Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD’s reliability and lifetime.

By Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu

https://arxiv.org/pdf/1706.08642
Onur Mutlu and Jeremie Kim,
"RowHammer: A Retrospective"
[Preliminary arXiv version]
Related Videos and Course Materials (I)


- Parallel Computer Architecture Course Materials (Lecture Videos)
Related Videos and Course Materials (II)

- Seminar in Computer Architecture Course Lecture Videos (Spring 2020, Fall 2019, Spring 2019, 2018)
- Seminar in Computer Architecture Course Materials (Spring 2020, Fall 2019, Spring 2019, 2018)

- Memory Systems Course Lecture Videos (Sept 2019, July 2019, June 2019, October 2018)
- Memory Systems Short Course Lecture Materials (Sept 2019, July 2019, June 2019, October 2018)
- ACACES Summer School Memory Systems Course Lecture Videos (2018, 2013)
- ACACES Summer School Memory Systems Course Materials (2018, 2013)
Some Open Source Tools (I)

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

- Other open-source software from my group
  - [https://github.com/CMU-SAFA/](https://github.com/CMU-SAFA/)
  - [http://www.ece.cmu.edu/~safari/tools.html](http://www.ece.cmu.edu/~safari/tools.html)
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/
  - http://www.ece.cmu.edu/~safari/tools.html
More Open Source Tools (III)

- A lot more open-source software from my group
  - https://github.com/CMU-SAFARI/
  - http://www.ece.cmu.edu/~safari/tools.html
ramulator-pim
A fast and flexible simulation infrastructure for exploring general-purpose processing-in-memory (PIM) architectures. Ramulator-PIM combines a widely-used simulator for out-of-order and in-order processors (ZSim) with Ramulator, a DRAM simulator with memory models for DDRx, LPDDRx, GDDRx, WIOx, HBMx, and HMCx. Ramulator is described in the IEEE ...

SMASH
SMASH is a hardware-software cooperative mechanism that enables highly-efficient indexing and storage of sparse matrices. The key idea of SMASH is to compress sparse matrices with a hierarchical bitmap compression format that can be accelerated from hardware. Described by Kanellopoulos et al. (MICRO '19)
https://people.inf.ethz.ch/omutlu/pub/SMA...

MQSim
MQSim is a fast and accurate simulator modeling the performance of modern multi-queue (MQ) SSDs as well as traditional SATA based SSDs. MQSim faithfully models new high-bandwidth protocol implementations, steady-state SSD conditions, and the full end-to-end latency of requests in modern SSDs. It is described in detail in the FAST 2018 paper by A...

Apollo
Apollo is an assembly polishing algorithm that attempts to correct the errors in an assembly. It can take multiple sets of reads in a single run and polish the assemblies of any size. Described in the Bioinformatics journal paper (2020) by Firtina et al. at https://people.inf.ethz.ch/omutlu/pub/apollo-technology-independent-genome-asse...

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

Shifted-Hamming-Distance

SneakySnake
The first and the only pre-alignment filtering algorithm that works on all modern high-performance computing architectures. It works efficiently and fast on CPU, FPGA, and GPU architectures and that greatly (by more than two orders of magnitude) expedites sequence alignment calculation. Described by Alser et al. (preliminary version at https://a...

AirLift
AirLift is a tool that updates mapped reads from one reference genome to another. Unlike existing tools, it accounts for regions not shared between the two reference genomes and enables remapping across all parts of the references. Described by Kim et al. (preliminary version at http://arxiv.org/abs/1912.08735)

GPGPUSim-Ramulator
The source code for GPGPUSim+Ramulator simulator. In this version, GPGPUSim uses Ramulator to simulate the DRAM. This simulator is used to produce some of the ...
Referenced Papers and Talks

- 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
An Interview on Research and Education

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

- Maurice Wilkes Award Speech (10 minutes)
  - https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15