Memory-Centric Computing

Onur Mutlu

omutlu@gmail.com

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

9 October 2021

ESWEEK Education Class

SAFARI ETH zürich Carnegie Mellon

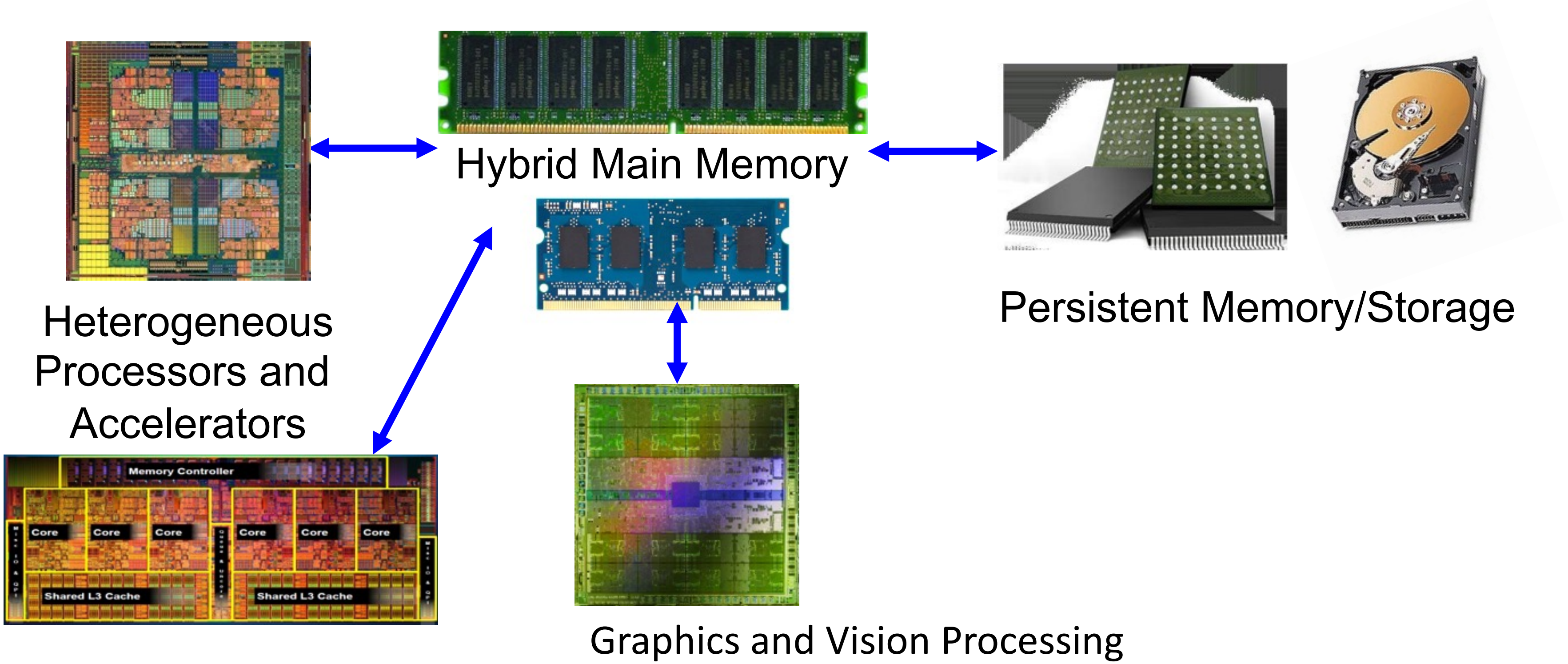
Brief Self Introduction

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

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

Current Research Mission

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



Hybrid Main Memory

Persistent Memory/Storage

Heterogeneous Processors and Accelerators

Graphics and Vision Processing

Build fundamentally better architectures

Four Key Current Directions

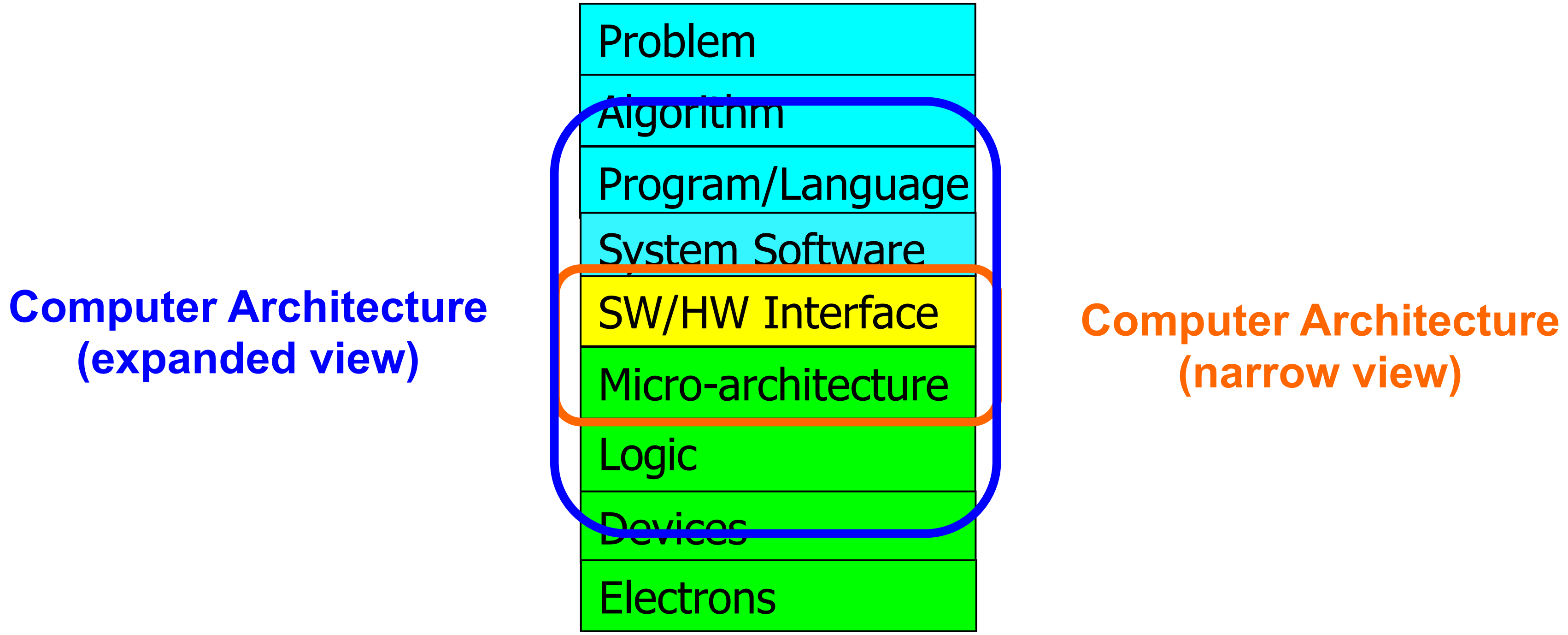
- Fundamentally Secure/Reliable/Safe Architectures

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

- Fundamentally Low-Latency and Predictable Architectures

- Architectures for AI/ML, Genomics, Medicine, Health

The Transformation Hierarchy

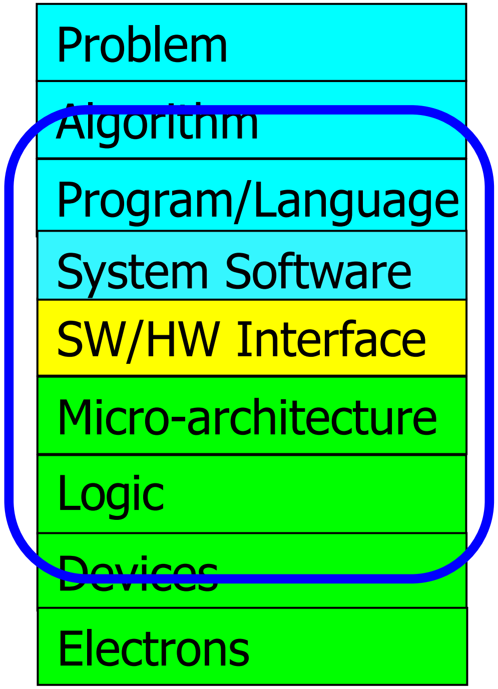


Axiom

To achieve the highest energy efficiency and performance:

we must take the expanded view

of computer architecture

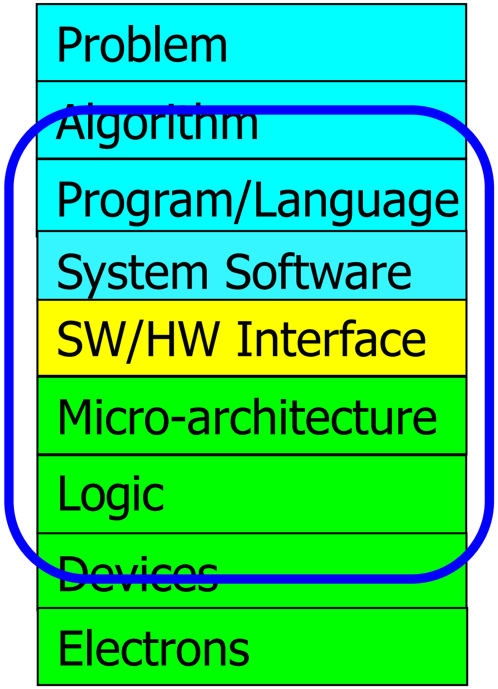


Co-design across the hierarchy: Algorithms to devices

Specialize as much as possible within the design goals

Current Research Mission & Major Topics

Build fundamentally better architectures



Problem

Algorithm

Program/Language

System Software

SW/HW Interface

Micro-architecture

Logic

Devices

Electrons

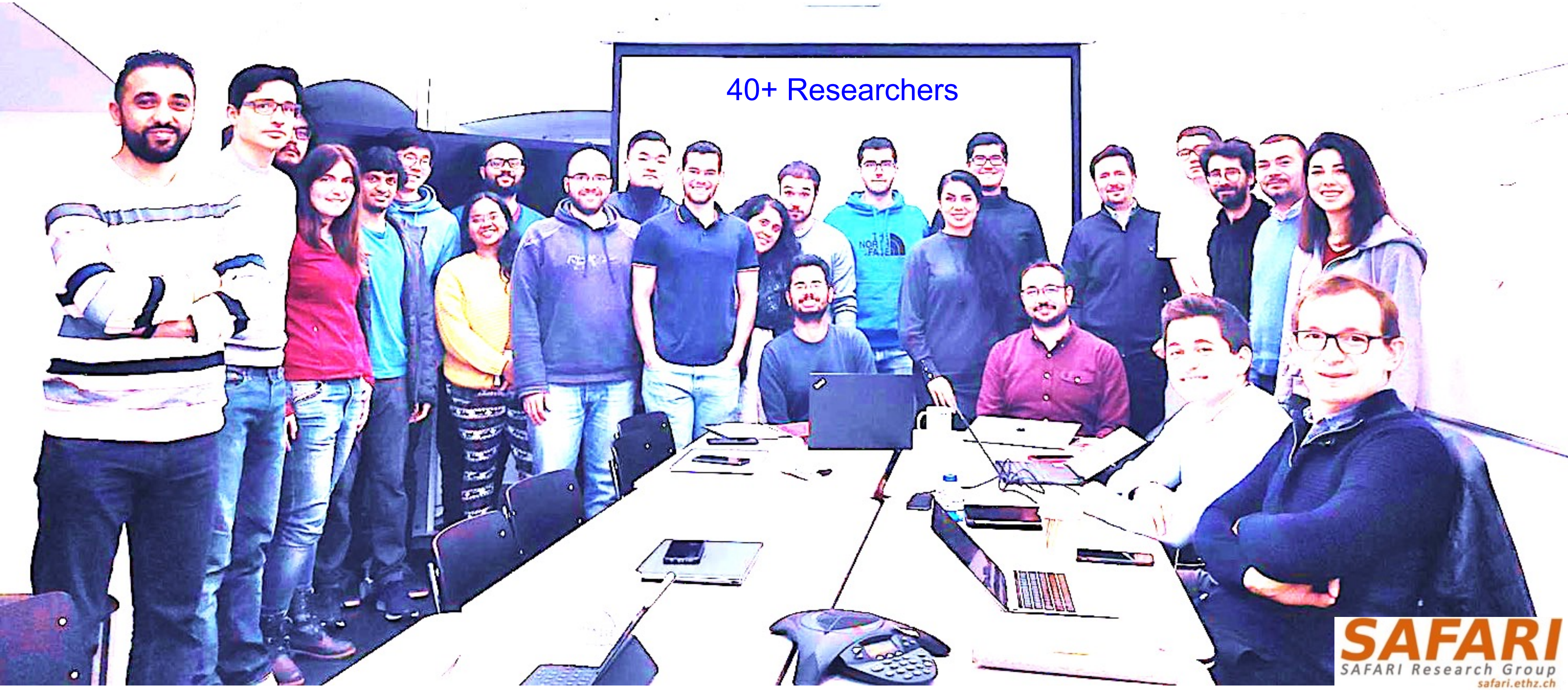
Broad research spanning apps, systems, logic with architecture at the center

- Data-centric arch. for low energy & high perf.
  - Proc. in Mem/DRAM, NVM, unified mem/storage
- Low-latency & predictable architectures
  - Low-latency, low-energy yet low-cost memory
  - QoS-aware and predictable memory systems
- Fundamentally secure/reliable/safe arch.
  - Tolerating all bit flips; patchable HW; secure mem
- Architectures for ML/AI/Genomics/Graph/Med
  - Algorithm/arch./logic co-design; full heterogeneity
- Data-driven and data-aware architectures
  - ML/AI-driven architectural controllers and design
  - Expressive memory and expressive systems

Onur Mutlu's SAFARI Research Group

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

https://safari.ethz.ch/safari-newsletter-april-2020/



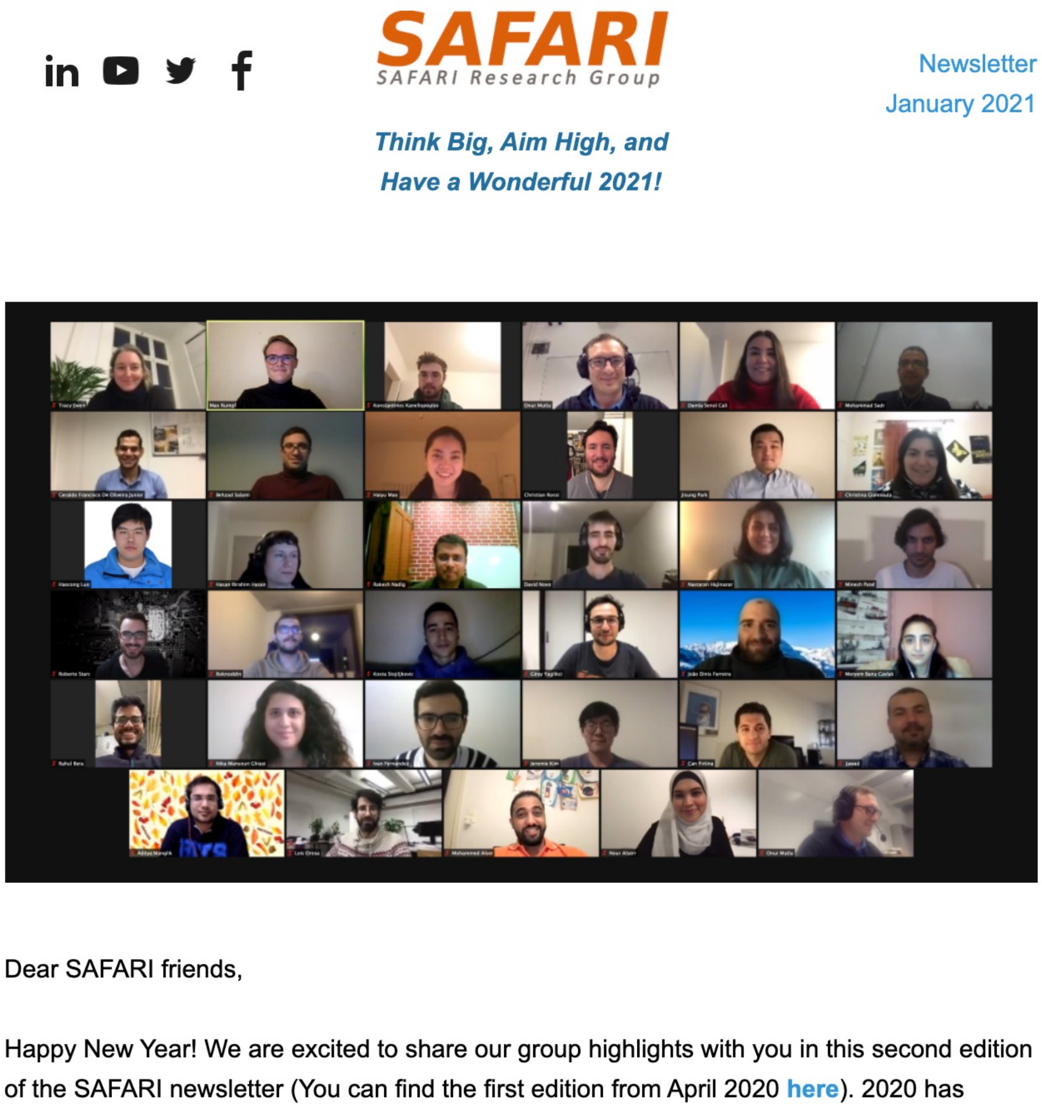
Think BIG, Aim HIGH!

SAFARI

https://safari.ethz.ch

SAFARI Newsletter January 2021 Edition

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



SAFARI
SAFARI Research Group

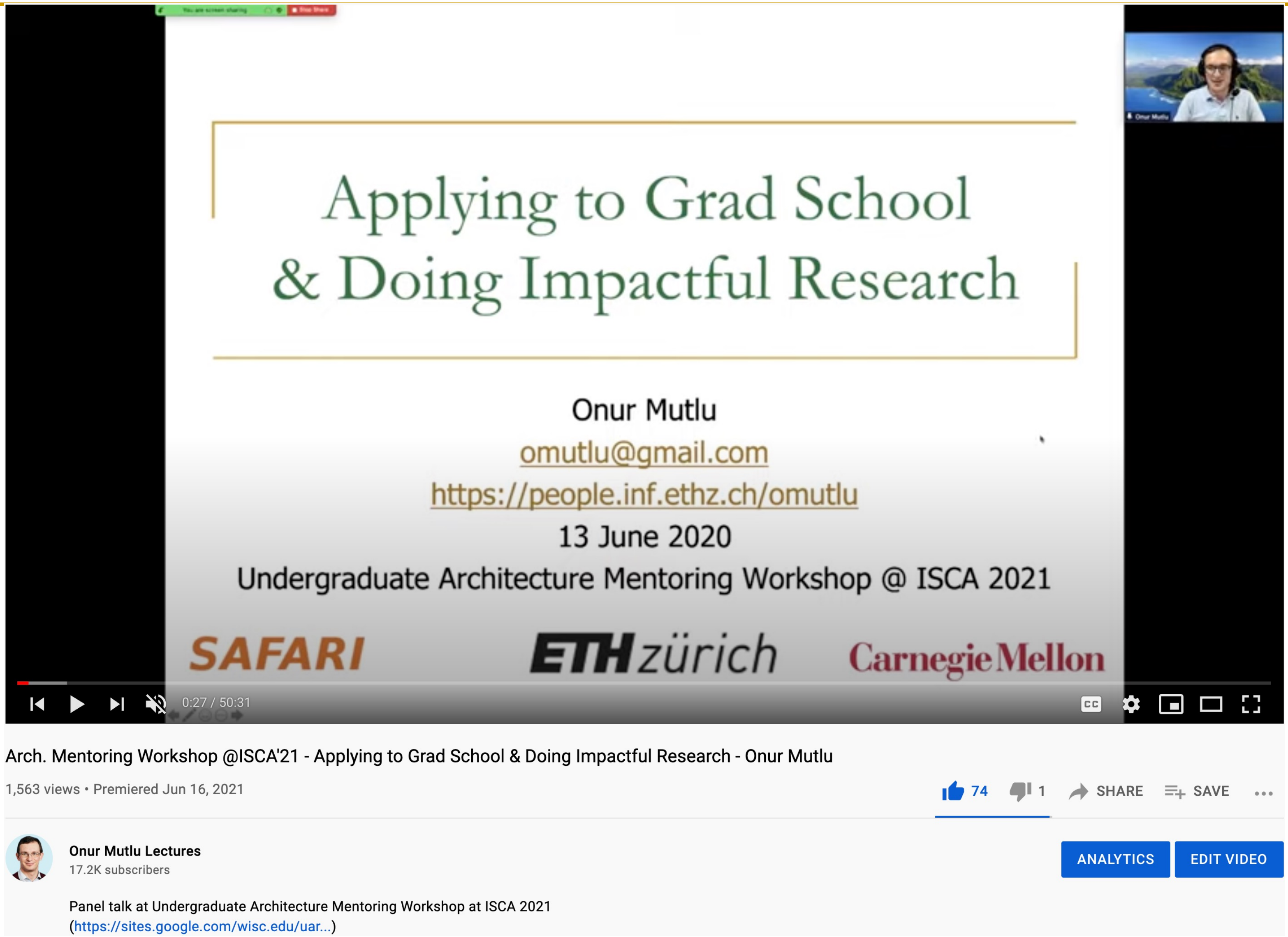
Newsletter
January 2021

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

Dear SAFARI friends,

Happy New Year! We are excited to share our group highlights with you in this second edition of the SAFARI newsletter (You can find the first edition from April 2020 **here**). 2020 has

A Talk on Impactful Research & Teaching



Applying to Grad School
& Doing Impactful Research

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

Arch. Mentoring Workshop @ISCA'21 - Applying to Grad School & Doing Impactful Research - Onur Mutlu

1,563 views • Premiered Jun 16, 2021

Onur Mutlu Lectures
17.2K subscribers

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

SAFARI

https://www.youtube.com/watch?v=83tlorht7Mc&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=54

Principle: Teaching and Research

…

Teaching drives Research

Research drives Teaching

…

Principle: Insight and Ideas

Focus on Insight

Encourage New Ideas

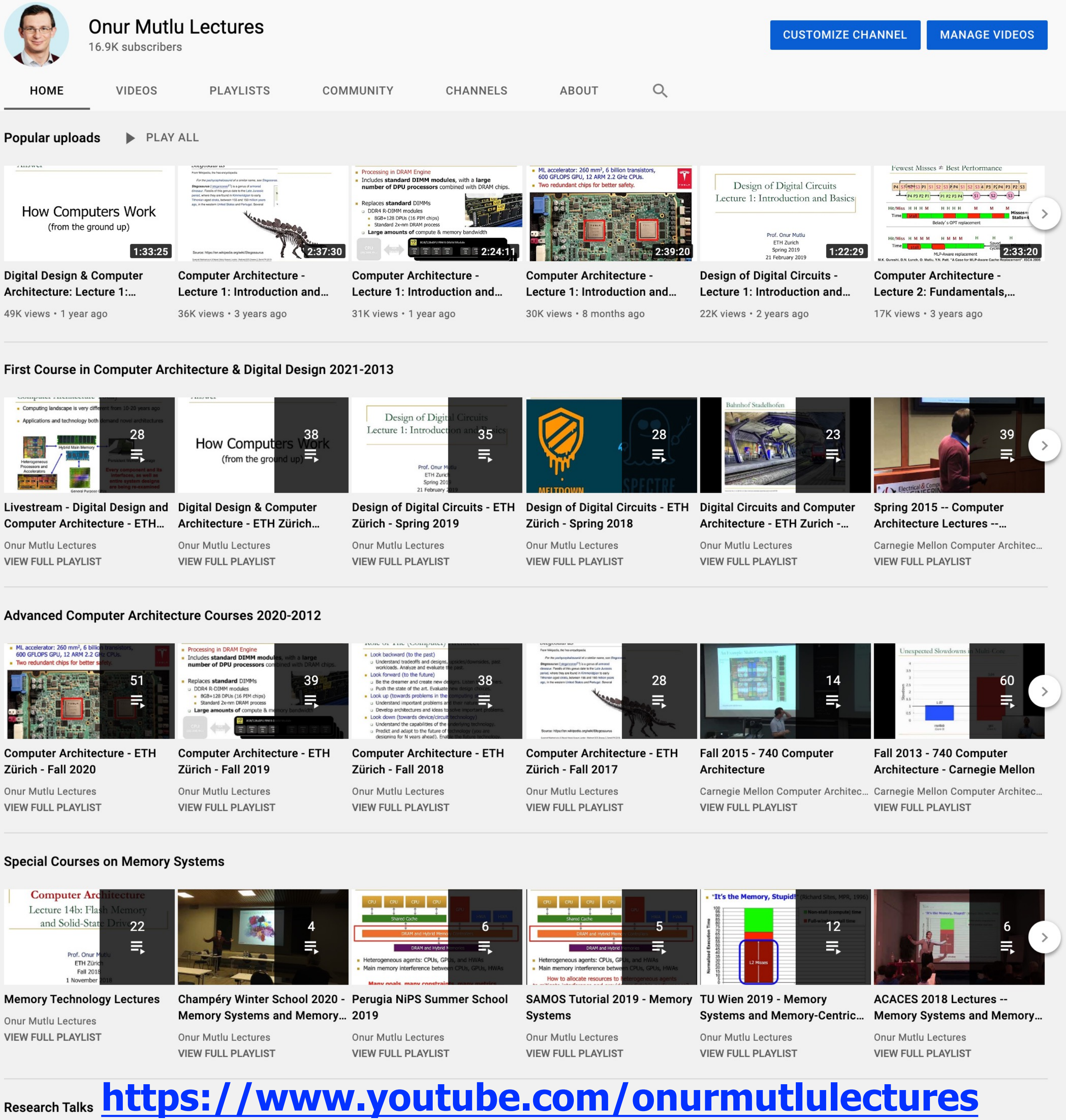
Principle: Learning and Scholarship

Focus on

learning and scholarship

Principle: Good Mindset, Goals & Focus

You can make a
good impact
on the world



Onur Mutlu Lectures

16.9K subscribers

Popular uploads ▶ PLAY ALL

- Digital Design & Computer Architecture: Lecture 1:... — 1:33:25 — 49K views • 1 year ago
- Computer Architecture - Lecture 1: Introduction and... — 2:37:30 — 36K views • 3 years ago
- Computer Architecture - Lecture 1: Introduction and... — 2:24:11 — 31K views • 1 year ago
- Computer Architecture - Lecture 1: Introduction and... — 2:39:20 — 30K views • 8 months ago
- Design of Digital Circuits - Lecture 1: Introduction and... — 1:22:29 — 22K views • 2 years ago
- Computer Architecture - Lecture 2: Fundamentals,... — 2:33:20 — 17K views • 3 years ago

First Course in Computer Architecture & Digital Design 2021-2013

- Livestream - Digital Design and Computer Architecture - ETH... — 28 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Digital Design & Computer Architecture - ETH Zürich... — 38 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Design of Digital Circuits - ETH Zürich - Spring 2019 — 35 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Design of Digital Circuits - ETH Zürich - Spring 2018 — 28 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Digital Circuits and Computer Architecture - ETH Zurich -... — 23 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Spring 2015 -- Computer Architecture Lectures --... — 39 — Carnegie Mellon Computer Architec... — VIEW FULL PLAYLIST

Advanced Computer Architecture Courses 2020-2012

- Computer Architecture - ETH Zürich - Fall 2020 — 51 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Computer Architecture - ETH Zürich - Fall 2019 — 39 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Computer Architecture - ETH Zürich - Fall 2018 — 38 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Computer Architecture - ETH Zürich - Fall 2017 — 28 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Fall 2015 - 740 Computer Architecture — 14 — Carnegie Mellon Computer Architec... — VIEW FULL PLAYLIST
- Fall 2013 - 740 Computer Architecture - Carnegie Mellon — 60 — Carnegie Mellon Computer Architec... — VIEW FULL PLAYLIST

Special Courses on Memory Systems

- Memory Technology Lectures — 22 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Champéry Winter School 2020 - Memory Systems and Memory... — 4 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- Perugia NiPS Summer School 2019 — 6 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- SAMOS Tutorial 2019 - Memory Systems — 5 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- TU Wien 2019 - Memory Systems and Memory-Centric... — 12 — Onur Mutlu Lectures — VIEW FULL PLAYLIST
- ACACES 2018 Lectures -- Memory Systems and Memory... — 6 — Onur Mutlu Lectures — VIEW FULL PLAYLIST

Research Talks

https://www.youtube.com/onurmutlulectures

Online Courses & Lectures

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

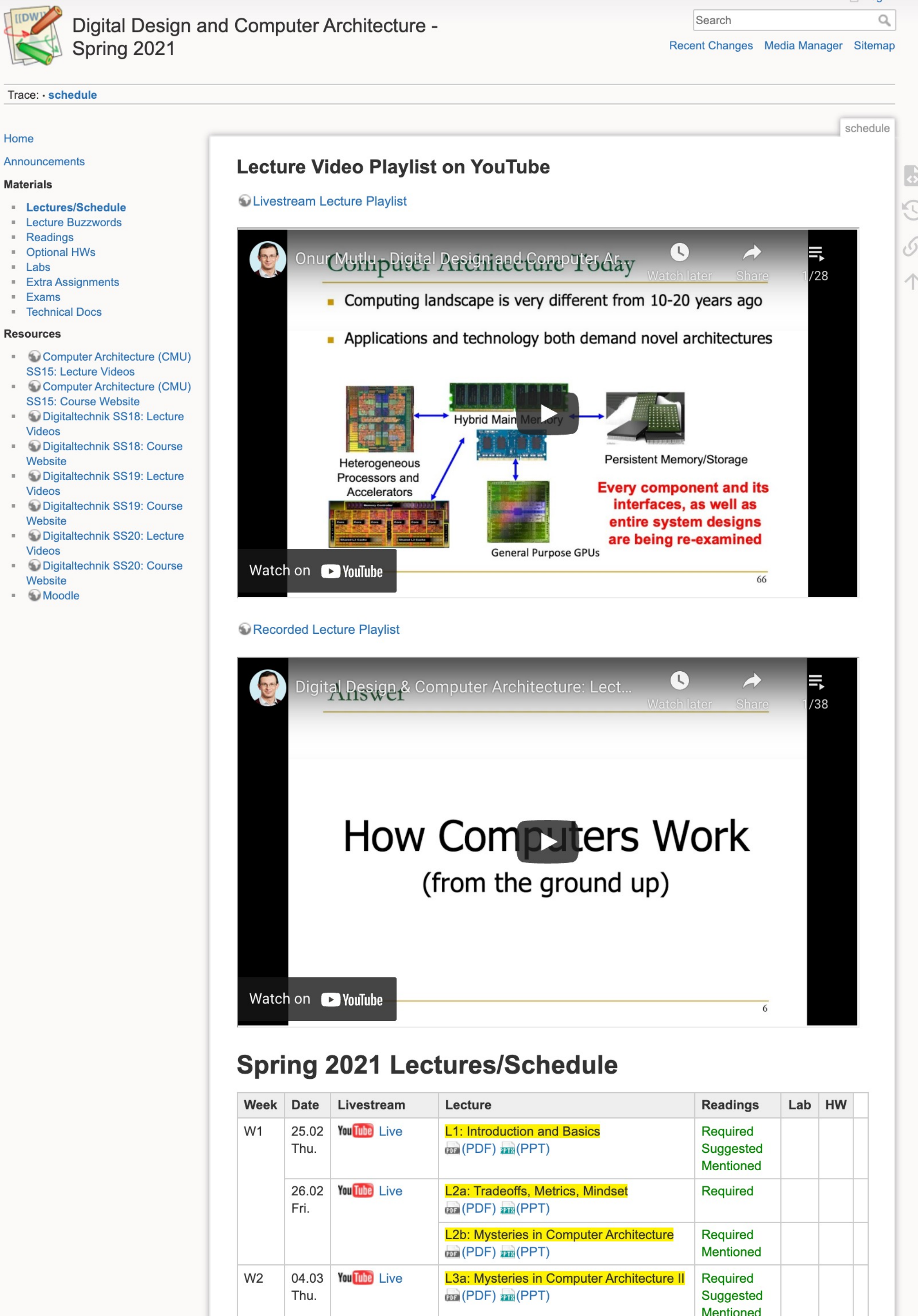
- **Advanced Computer Architecture Course**
  - Computer Architecture
  - Fall 2020 Edition: https://www.youtube.com/watch?v=c3mPdZA-Fmc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN

DDCA (Spring 2021)

- https://safari.ethz.ch/digitaltechnik/spring2021/doku.php?id=schedule

- https://www.youtube.com/watch?v=LbC0EZY8yw4&list=PL5Q2soXY2Zi_uej3aY39YB5pfW4SJ7LlN

- Bachelor's course
  - 2nd semester at ETH Zurich
  - Rigorous introduction into "How Computers Work"
  - Digital Design/Logic
  - Computer Architecture
  - 10 FPGA Lab Assignments


Search
Recent Changes
Media Manager
Sitemap
schedule
Onur Mutlu - Digital Design and Computer Ar...
Computer Architecture Today
Watch later
Share
1/28
Computing landscape is very different from 10-20 years ago
Applications and technology both demand novel architectures
Hybrid Main Memory
Heterogeneous Processors and Accelerators
Persistent Memory/Storage
General Purpose GPUs
Every component and its interfaces, as well as entire system designs are being re-examined
66
Watch on YouTube
Digital Design & Computer Architecture: Lect...
Answer
Watch later
Share
1/38
How Computers Work
(from the ground up)
6
Watch on YouTube

Digital Design and Computer Architecture - Spring 2021

Trace: · **schedule**

Home

Announcements

Materials

- **Lectures/Schedule**
- Lecture Buzzwords
- Readings
- Optional HWs
- Labs
- Extra Assignments
- Exams
- Technical Docs

Resources

- Computer Architecture (CMU) SS15: Lecture Videos
- Computer Architecture (CMU) SS15: Course Website
- Digitaltechnik SS18: Lecture Videos
- Digitaltechnik SS18: Course Website
- Digitaltechnik SS19: Lecture Videos
- Digitaltechnik SS19: Course Website
- Digitaltechnik SS20: Lecture Videos
- Digitaltechnik SS20: Course Website
- Moodle

Lecture Video Playlist on YouTube

Livestream Lecture Playlist

Recorded Lecture Playlist

Spring 2021 Lectures/Schedule

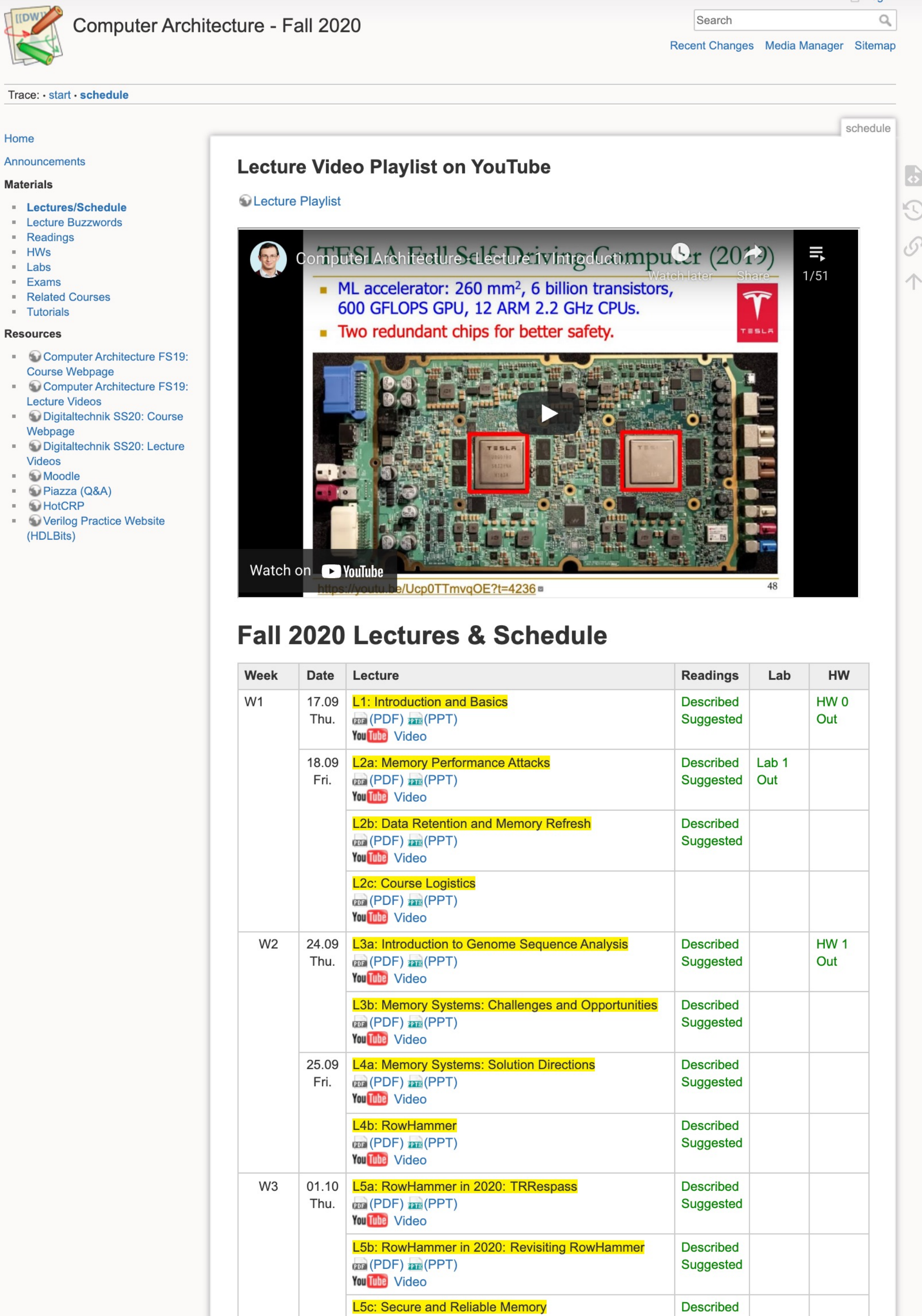
Week	Date	Livestream	Lecture	Readings	Lab	HW
W1	25.02 Thu.	YouTube Live	L1: Introduction and Basics (PDF) (PPT)	Required Suggested Mentioned		
	26.02 Fri.	YouTube Live	L2a: Tradeoffs, Metrics, Mindset (PDF) (PPT)	Required		
			L2b: Mysteries in Computer Architecture (PDF) (PPT)	Required Mentioned		
W2	04.03 Thu.	YouTube Live	L3a: Mysteries in Computer Architecture II (PDF) (PPT)	Required Suggested Mentioned		

Comp Arch (Fall 2020)

- https://safari.ethz.ch/architecture/fall2020/doku.php?id=schedule

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

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



Computer Architecture - Fall 2020

Search

Recent Changes Media Manager Sitemap

Trace: • start • **schedule**

schedule

Home

Announcements

Materials

- **Lectures/Schedule**
- Lecture Buzzwords
- Readings
- HWs
- Labs
- Exams
- Related Courses
- Tutorials

Resources

- Computer Architecture FS19: Course Webpage
- Computer Architecture FS19: Lecture Videos
- Digitaltechnik SS20: Course Webpage
- Digitaltechnik SS20: Lecture Videos
- Moodle
- Piazza (Q&A)
- HotCRP
- Verilog Practice Website (HDLBits)

Lecture Video Playlist on YouTube

Lecture Playlist

Computer Architecture - Lecture 1: Introducti...
Watch later
Share
1/51
TESLA Full Self-Driving Computer (2019)
ML accelerator: 260 mm², 6 billion transistors, 600 GFLOPS GPU, 12 ARM 2.2 GHz CPUs.
Two redundant chips for better safety.
TESLA
Watch on YouTube
https://youtu.be/Ucp0TTmvqOE?t=4236
48

Fall 2020 Lectures & Schedule

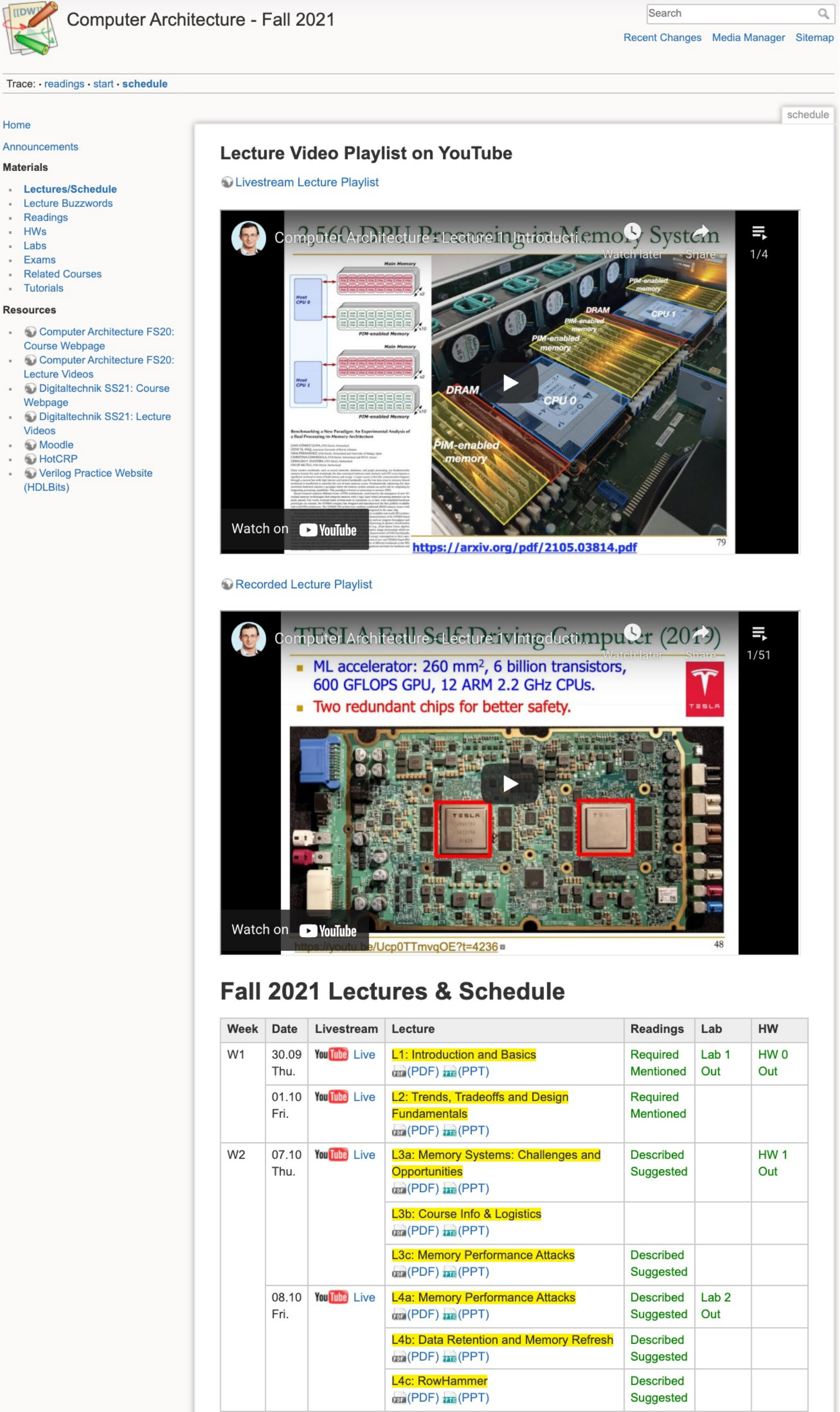
Week	Date	Lecture	Readings	Lab	HW
W1	17.09 Thu.	L1: Introduction and Basics (PDF) (PPT) YouTube Video	Described Suggested		HW 0 Out
	18.09 Fri.	L2a: Memory Performance Attacks (PDF) (PPT) YouTube Video	Described Suggested	Lab 1 Out	
		L2b: Data Retention and Memory Refresh (PDF) (PPT) YouTube Video	Described Suggested		
		L2c: Course Logistics (PDF) (PPT) YouTube Video			
W2	24.09 Thu.	L3a: Introduction to Genome Sequence Analysis (PDF) (PPT) YouTube Video	Described Suggested		HW 1 Out
		L3b: Memory Systems: Challenges and Opportunities (PDF) (PPT) YouTube Video	Described Suggested		
	25.09 Fri.	L4a: Memory Systems: Solution Directions (PDF) (PPT) YouTube Video	Described Suggested		
		L4b: RowHammer (PDF) (PPT) YouTube Video	Described Suggested		
W3	01.10 Thu.	L5a: RowHammer in 2020: TRRespass (PDF) (PPT) YouTube Video	Described Suggested		
		L5b: RowHammer in 2020: Revisiting RowHammer (PDF) (PPT) YouTube Video	Described Suggested		
		L5c: Secure and Reliable Memory	Described		

Comp Arch (Current)

- https://safari.ethz.ch/architecture/fall2021/doku.php?id=schedule

- **Youtube Livestream:**
  - https://www.youtube.com/watch?v=4yfkM_5EFgo&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF

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


Search
Recent Changes
Media Manager
Sitemap
Computer Architecture - Lecture 1: Introducti...
2,560 DPU Processing-in-Memory System
Watch later
Share
1/4
Main Memory
Host CPU 0
Host CPU 1
PIM-enabled Memory
DRAM
CPU 0
CPU 1
PIM-enabled memory
Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture
Watch on YouTube
https://arxiv.org/pdf/2105.03814.pdf
79
Computer Architecture - Lecture 1: Introducti...
TESLA Full Self-Driving Computer (2019)
Watch later
Share
1/51
ML accelerator: 260 mm², 6 billion transistors, 600 GFLOPS GPU, 12 ARM 2.2 GHz CPUs.
Two redundant chips for better safety.
TESLA
Watch on YouTube
https://youtu.be/Ucp0TTmvqOE?t=4236
48

Computer Architecture - Fall 2021

Trace: · readings · start · **schedule**

schedule

Home

Announcements

Materials

- **Lectures/Schedule**
- Lecture Buzzwords
- Readings
- HWs
- Labs
- Exams
- Related Courses
- Tutorials

Resources

- Computer Architecture FS20: Course Webpage
- Computer Architecture FS20: Lecture Videos
- Digitaltechnik SS21: Course Webpage
- Digitaltechnik SS21: Lecture Videos
- Moodle
- HotCRP
- Verilog Practice Website (HDLBits)

Lecture Video Playlist on YouTube

Livestream Lecture Playlist

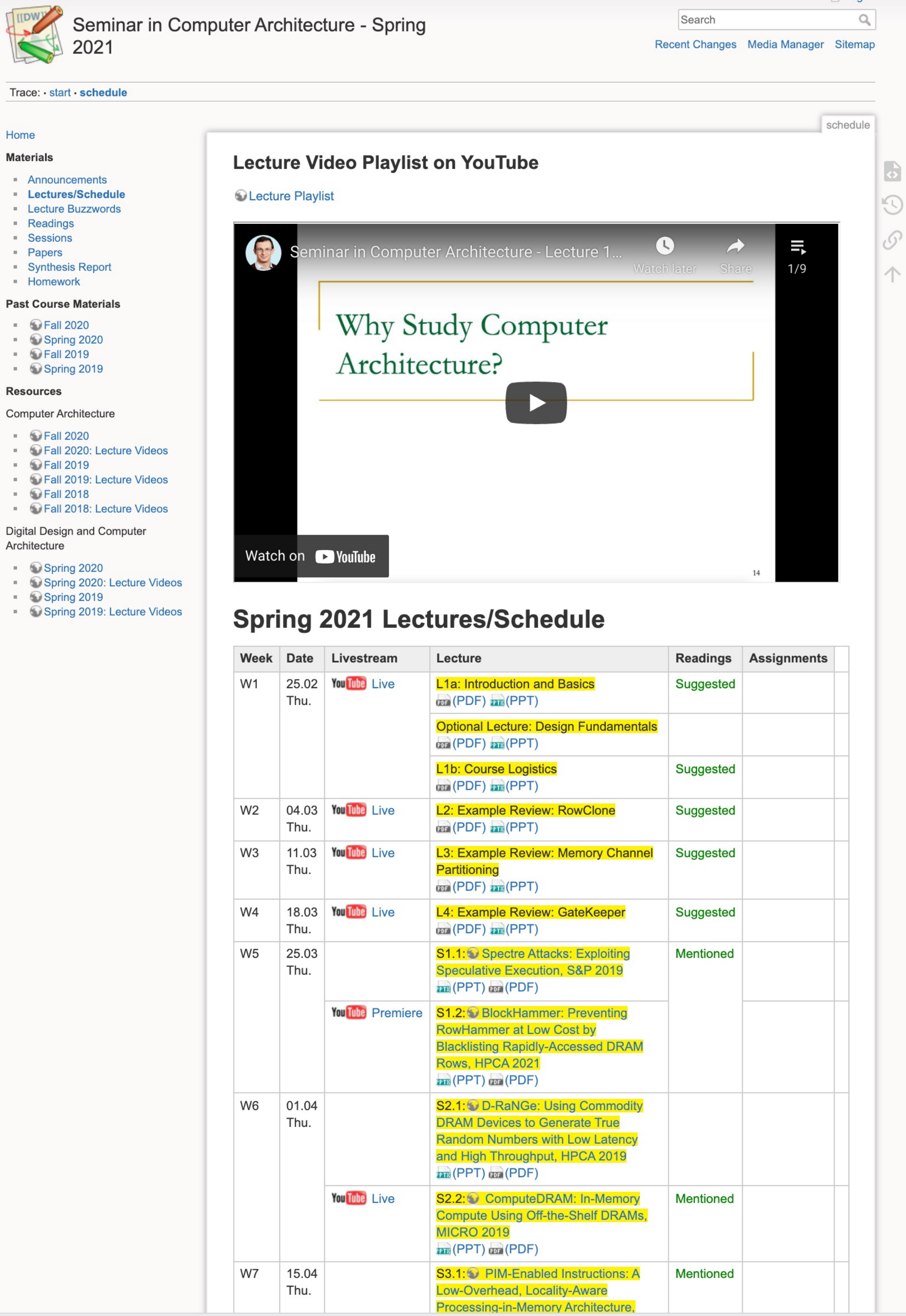
Recorded Lecture Playlist

Fall 2021 Lectures & Schedule

Week	Date	Livestream	Lecture	Readings	Lab	HW
W1	30.09 Thu.	YouTube Live	L1: Introduction and Basics (PDF) (PPT)	Required Mentioned	Lab 1 Out	HW 0 Out
	01.10 Fri.	YouTube Live	L2: Trends, Tradeoffs and Design Fundamentals (PDF) (PPT)	Required Mentioned		
W2	07.10 Thu.	YouTube Live	L3a: Memory Systems: Challenges and Opportunities (PDF) (PPT)	Described Suggested		HW 1 Out
			L3b: Course Info & Logistics (PDF) (PPT)			
			L3c: Memory Performance Attacks (PDF) (PPT)	Described Suggested		
	08.10 Fri.	YouTube Live	L4a: Memory Performance Attacks (PDF) (PPT)	Described Suggested	Lab 2 Out	
			L4b: Data Retention and Memory Refresh (PDF) (PPT)	Described Suggested		
			L4c: RowHammer (PDF) (PPT)	Described Suggested		

Seminar (Spring'21)

- https://safari.ethz.ch/architecture_seminar/spring2021/doku.php?id=schedule
- https://www.youtube.com/watch?v=t3m93ZpLOyw&list=PL5Q2soXY2Zi_awYdjmWVIUegsbY7TPGW4
- Critical analysis course
  - Taken by Bachelor's/Masters/PhD students
  - Cutting-edge research topics + fundamentals in Computer Architecture
  - 20+ research papers, presentations, analyses



Seminar in Computer Architecture - Spring 2021

Search

Recent Changes Media Manager Sitemap

Trace: • start • schedule

schedule

Home

Materials

- Announcements
- Lectures/Schedule
- Lecture Buzzwords
- Readings
- Sessions
- Papers
- Synthesis Report
- Homework

Past Course Materials

- Fall 2020
- Spring 2020
- Fall 2019
- Spring 2019

Resources

Computer Architecture

- Fall 2020
- Fall 2020: Lecture Videos
- Fall 2019
- Fall 2019: Lecture Videos
- Fall 2018
- Fall 2018: Lecture Videos

Digital Design and Computer Architecture

- Spring 2020
- Spring 2020: Lecture Videos
- Spring 2019
- Spring 2019: Lecture Videos

Lecture Video Playlist on YouTube

Lecture Playlist

Seminar in Computer Architecture - Lecture 1...

Spring 2021 Lectures/Schedule

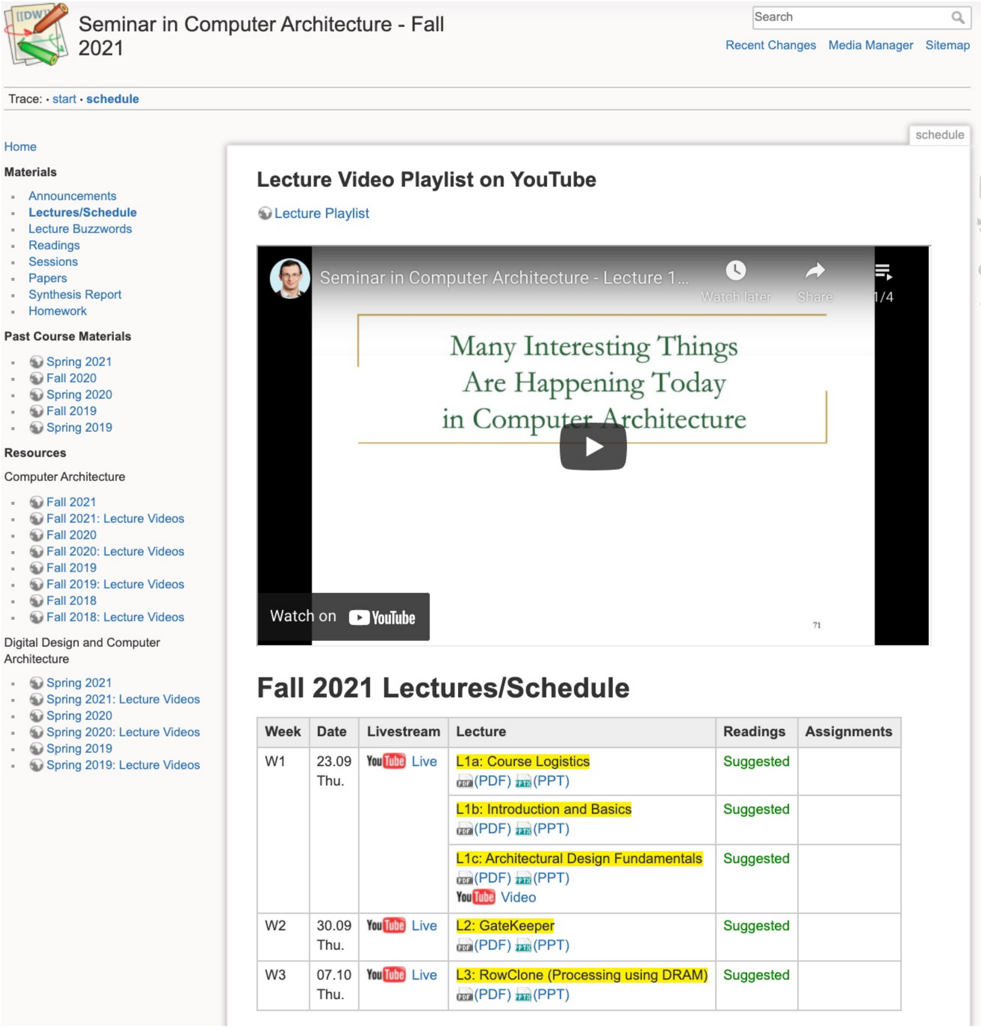
Week	Date	Livestream	Lecture	Readings	Assignments
W1	25.02 Thu.	YouTube Live	L1a: Introduction and Basics (PDF) (PPT)	Suggested	
			Optional Lecture: Design Fundamentals (PDF) (PPT)		
			L1b: Course Logistics (PDF) (PPT)	Suggested	
W2	04.03 Thu.	YouTube Live	L2: Example Review: RowClone (PDF) (PPT)	Suggested	
W3	11.03 Thu.	YouTube Live	L3: Example Review: Memory Channel Partitioning (PDF) (PPT)	Suggested	
W4	18.03 Thu.	YouTube Live	L4: Example Review: GateKeeper (PDF) (PPT)	Suggested	
W5	25.03 Thu.		S1.1: Spectre Attacks: Exploiting Speculative Execution, S&P 2019 (PPT) (PDF)	Mentioned	
		YouTube Premiere	S1.2: BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows, HPCA 2021 (PPT) (PDF)		
W6	01.04 Thu.		S2.1: D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput, HPCA 2019 (PPT) (PDF)		
		YouTube Live	S2.2: ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs, MICRO 2019 (PPT) (PDF)	Mentioned	
W7	15.04 Thu.		S3.1: PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture,	Mentioned	

Seminar (Current)

- https://safari.ethz.ch/architecture_seminar/fall2021/doku.php?id=schedule

- **Youtube Livestream:**
  - https://www.youtube.com/watch?v=4TcP297mdsI&list=PL5Q2soXY2Zi_7UBNmC9B8Yr5JSwTG9yH4

- Critical analysis course
  - Taken by Bachelor's/Masters/PhD students
  - Cutting-edge research topics + fundamentals in Computer Architecture
  - 20+ research papers, presentations, analyses



Seminar in Computer Architecture - Fall 2021

Home

Materials

- Announcements
- **Lectures/Schedule**
- Lecture Buzzwords
- Readings
- Sessions
- Papers
- Synthesis Report
- Homework

Past Course Materials

- Spring 2021
- Fall 2020
- Spring 2020
- Fall 2019
- Spring 2019

Resources

Computer Architecture

- Fall 2021
- Fall 2021: Lecture Videos
- Fall 2020
- Fall 2020: Lecture Videos
- Fall 2019
- Fall 2019: Lecture Videos
- Fall 2018
- Fall 2018: Lecture Videos

Digital Design and Computer Architecture

- Spring 2021
- Spring 2021: Lecture Videos
- Spring 2020
- Spring 2020: Lecture Videos
- Spring 2019
- Spring 2019: Lecture Videos

Lecture Video Playlist on YouTube

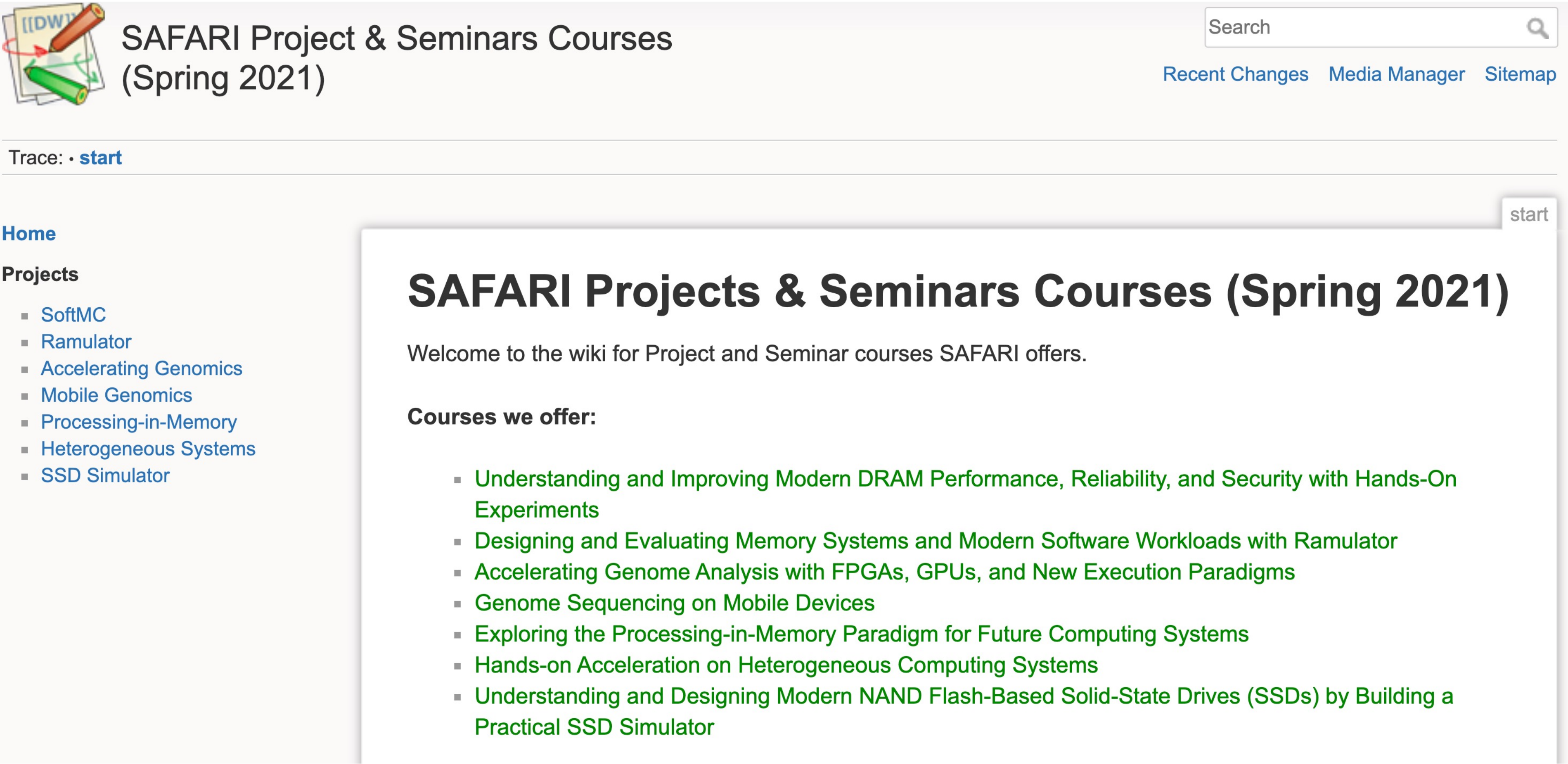
Lecture Playlist

Fall 2021 Lectures/Schedule

Week	Date	Livestream	Lecture	Readings	Assignments
W1	23.09 Thu.	YouTube Live	L1a: Course Logistics (PDF) (PPT)	Suggested	
			L1b: Introduction and Basics (PDF) (PPT)	Suggested	
			L1c: Architectural Design Fundamentals (PDF) (PPT) YouTube Video	Suggested	
W2	30.09 Thu.	YouTube Live	L2: GateKeeper (PDF) (PPT)	Suggested	
W3	07.10 Thu.	YouTube Live	L3: RowClone (Processing using DRAM) (PDF) (PPT)	Suggested	

Hands-On Projects & Seminars Courses

- https://safari.ethz.ch/projects_and_seminars/doku.php



SAFARI Project & Seminars Courses (Spring 2021)

Home

Projects

- SoftMC
- Ramulator
- Accelerating Genomics
- Mobile Genomics
- Processing-in-Memory
- Heterogeneous Systems
- SSD Simulator

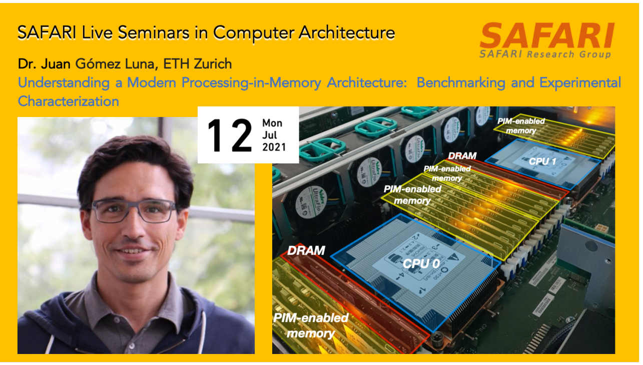
SAFARI Projects & Seminars Courses (Spring 2021)

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

Courses we offer:

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

SAFARI Live Seminars



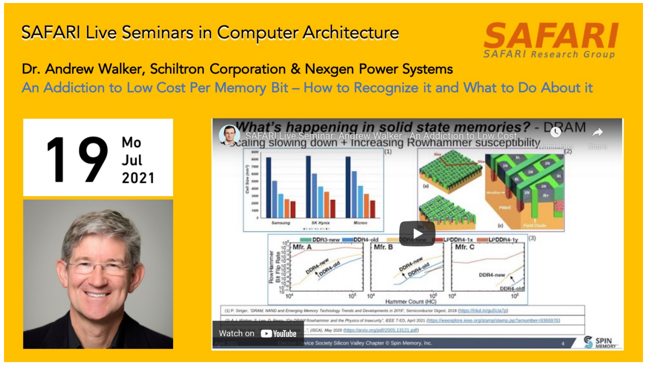
SAFARI Live Seminars in Computer Architecture

SAFARI
SAFARI Research Group

Dr. Juan Gómez Luna, ETH Zurich

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

12 Mon Jul 2021



SAFARI Live Seminars in Computer Architecture

SAFARI
SAFARI Research Group

Dr. Andrew Walker, Schiltron Corporation & Nexgen Power Systems

An Addiction to Low Cost Per Memory Bit – How to Recognize it and What to Do About it

19 Mo Jul 2021

SAFARI Live Seminar: Andrew Walker - An Addiction to Low Cost ...

Watch later

Share

What's happening in solid state memories? - DRAM

Scaling slowing down + Increasing Rowhammer susceptibility

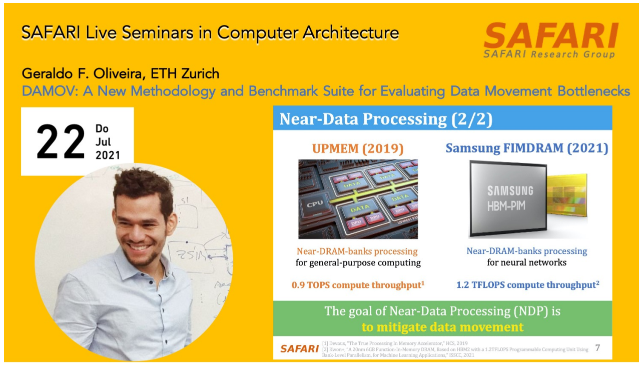
(1) P. Singer, "DRAM, NAND and Emerging Memory Technology Trends and Developments in 2019", Semiconductor Digest, 2019 (https://lnkd.in/guSUa7p)

(2) A. J. Walker, S. Lee, D. Beery, "On DRAM Rowhammer and the Physics of Insecurity", IEEE T-ED, April 2021 (https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9366976)

(3) ... (ISCA), May 2020 (https://arxiv.org/pdf/2005.13121.pdf)

Watch on YouTube

Electron Device Society Silicon Valley Chapter © Spin Memory, Inc. 4 SPIN MEMORY



SAFARI Live Seminars in Computer Architecture

SAFARI
SAFARI Research Group

Geraldo F. Oliveira, ETH Zurich

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

22 Do Jul 2021

Near-Data Processing (2/2)

UPMEM (2019)

Near-DRAM-banks processing for general-purpose computing

0.9 TOPS compute throughput[1]

Samsung FIMDRAM (2021)

Near-DRAM-banks processing for neural networks

1.2 TFLOPS compute throughput[2]

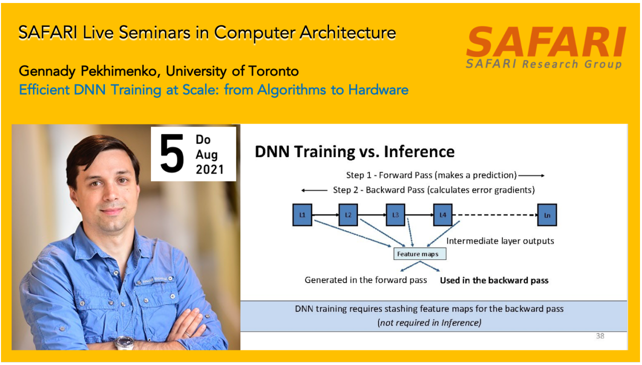
The goal of Near-Data Processing (NDP) is **to mitigate data movement**

SAFARI

[1] Devaux, "The True Processing In Memory Accelerator," HCS, 2019

[2] Kwon+, "A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications," ISSCC, 2021

7



SAFARI Live Seminars in Computer Architecture

SAFARI
SAFARI Research Group

Gennady Pekhimenko, University of Toronto

Efficient DNN Training at Scale: from Algorithms to Hardware

5 Do Aug 2021

DNN Training vs. Inference

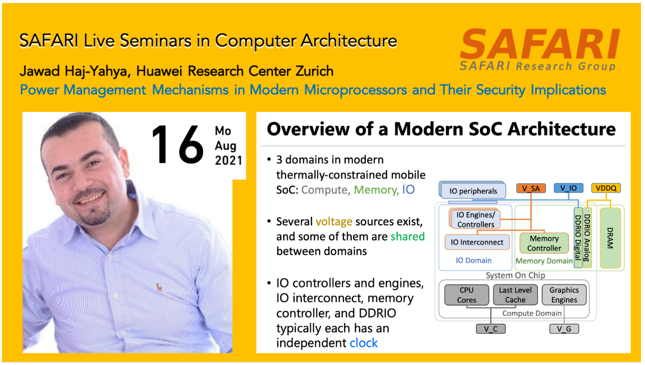
Step 1 - Forward Pass (makes a prediction) →

← Step 2 - Backward Pass (calculates error gradients)

Generated in the forward pass **Used in the backward pass**

DNN training requires stashing feature maps for the backward pass
(*not required in Inference*)

38



SAFARI Live Seminars in Computer Architecture

SAFARI
SAFARI Research Group

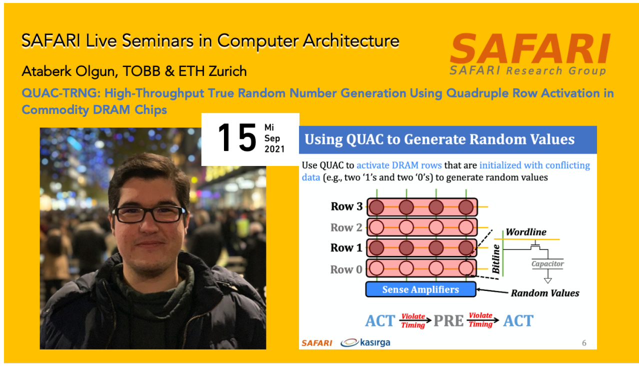
Jawad Haj-Yahya, Huawei Research Center Zurich

Power Management Mechanisms in Modern Microprocessors and Their Security Implications

16 Mo Aug 2021

Overview of a Modern SoC Architecture

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



SAFARI Live Seminars in Computer Architecture

SAFARI
SAFARI Research Group

Ataberk Olgun, TOBB & ETH Zurich

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

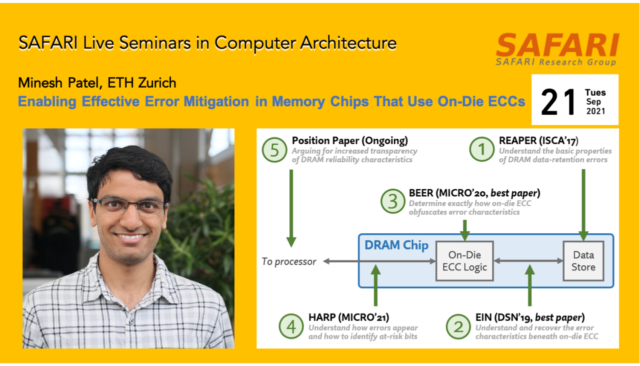
15 Mi Sep 2021

Using QUAC to Generate Random Values

Use QUAC to activate DRAM rows that are initialized with conflicting data (e.g., two '1's and two '0's) to generate random values

ACT $\xrightarrow[\text{Timing}]{\text{Violate}}$ PRE $\xrightarrow[\text{Timing}]{\text{Violate}}$ ACT

SAFARI kasırga 6



SAFARI Live Seminars in Computer Architecture

SAFARI
SAFARI Research Group

Minesh Patel, ETH Zurich

Enabling Effective Error Mitigation in Memory Chips That Use On-Die ECCs

21 Tues Sep 2021

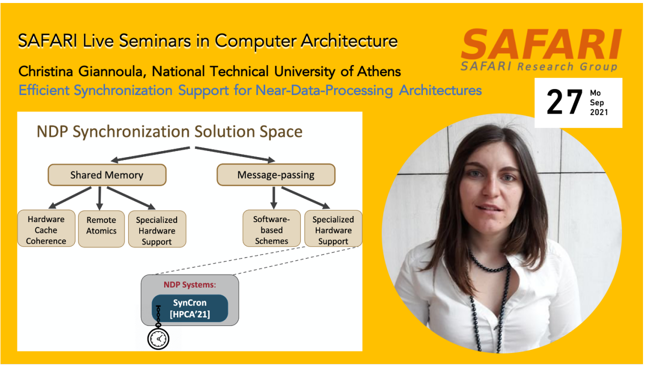
5 Position Paper (Ongoing)
Arguing for increased transparency of DRAM reliability characteristics

1 REAPER (ISCA'17)
Understand the basic properties of DRAM data-retention errors

3 BEER (MICRO'20, *best paper*)
Determine exactly how on-die ECC obfuscates error characteristics

4 HARP (MICRO'21)
Understand how errors appear and how to identify at-risk bits

2 EIN (DSN'19, *best paper*)
Understand and recover the error characteristics beneath on-die ECC



SAFARI Live Seminars in Computer Architecture

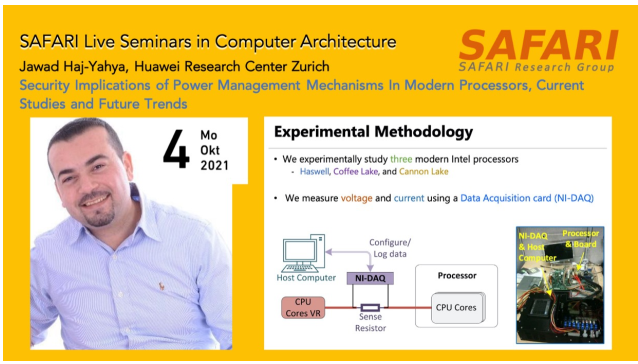
SAFARI
SAFARI Research Group

Christina Giannoula, National Technical University of Athens

Efficient Synchronization Support for Near-Data-Processing Architectures

27 Mo Sep 2021

NDP Synchronization Solution Space



SAFARI Live Seminars in Computer Architecture

SAFARI
SAFARI Research Group

Jawad Haj-Yahya, Huawei Research Center Zurich

Security Implications of Power Management Mechanisms In Modern Processors, Current Studies and Future Trends

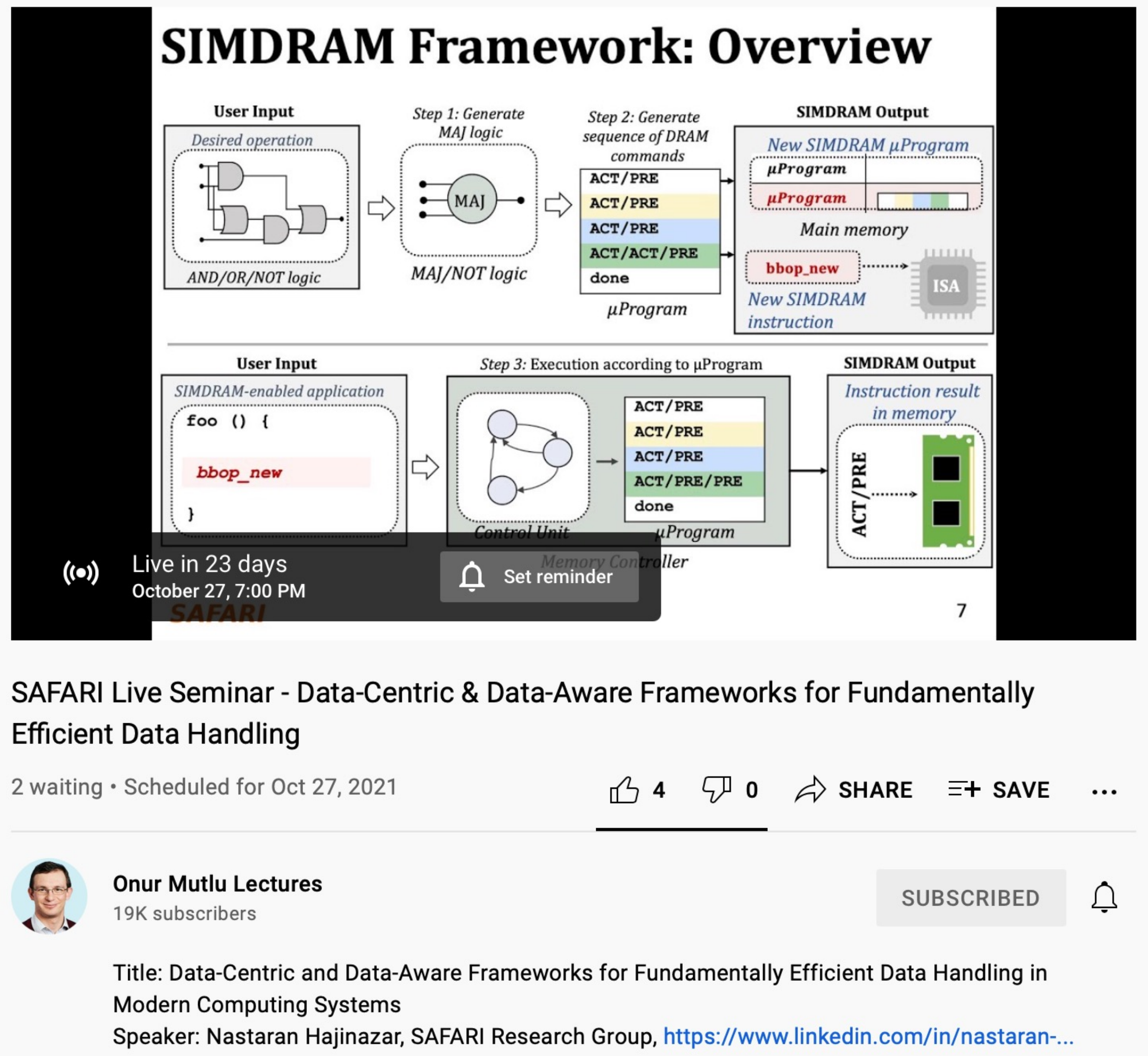
4 Mo Okt 2021

Experimental Methodology

- We experimentally study three modern Intel processors
  - Haswell, Coffee Lake, and Cannon Lake
- We measure voltage and current using a Data Acquisition card (NI-DAQ)

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

Upcoming SAFARI Live Seminar: Oct 27



SAFARI Live Seminar - Data-Centric & Data-Aware Frameworks for Fundamentally Efficient Data Handling

2 waiting • Scheduled for Oct 27, 2021

Onur Mutlu Lectures

19K subscribers

Title: Data-Centric and Data-Aware Frameworks for Fundamentally Efficient Data Handling in Modern Computing Systems

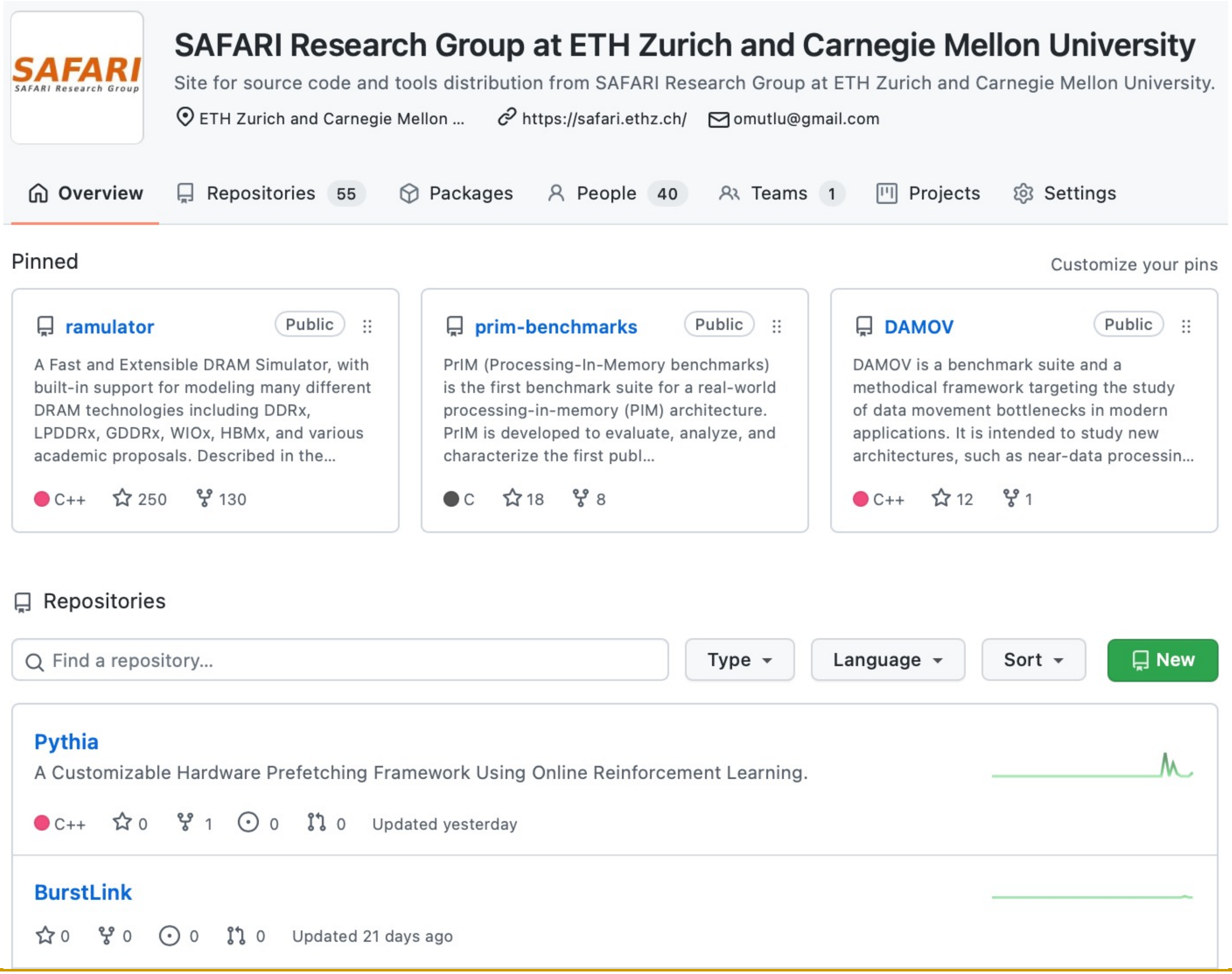
Speaker: Nastaran Hajinazar, SAFARI Research Group, https://www.linkedin.com/in/nastaran-...

https://www.youtube.com/watch?v=XIfPHtvA9rw&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9&index=10

Open-Source Artifacts

https://github.com/CMU-SAFARI

Open Source Tools: SAFARI GitHub



SAFARI Research Group at ETH Zurich and Carnegie Mellon University

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

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

Pinned

ramulator Public

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...

● C++ ☆ 250 Forks 130

prim-benchmarks Public

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

● C ☆ 18 Forks 8

DAMOV Public

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

● C++ ☆ 12 Forks 1

Repositories

Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning.

● C++ ☆ 0 1 0 0 Updated yesterday

BurstLink

☆ 0 0 0 0 Updated 21 days ago

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

Research & Teaching: Some Overview Talks

https://www.youtube.com/onurmutlulectures

- Future Computing Architectures
  - https://www.youtube.com/watch?v=kgiZlSOcGFM&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=r7sn41lH-4A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=41
- Rethinking Memory System Design
  - https://www.youtube.com/watch?v=F7xZLNMIY1E&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=3
- Intelligent Architectures for Intelligent Machines
  - https://www.youtube.com/watch?v=c6_LgzuNdkw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=25
- The Story of RowHammer
  - https://www.youtube.com/watch?v=sgd7PHQQ1AI&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=39

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

More Thoughts and Suggestions

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

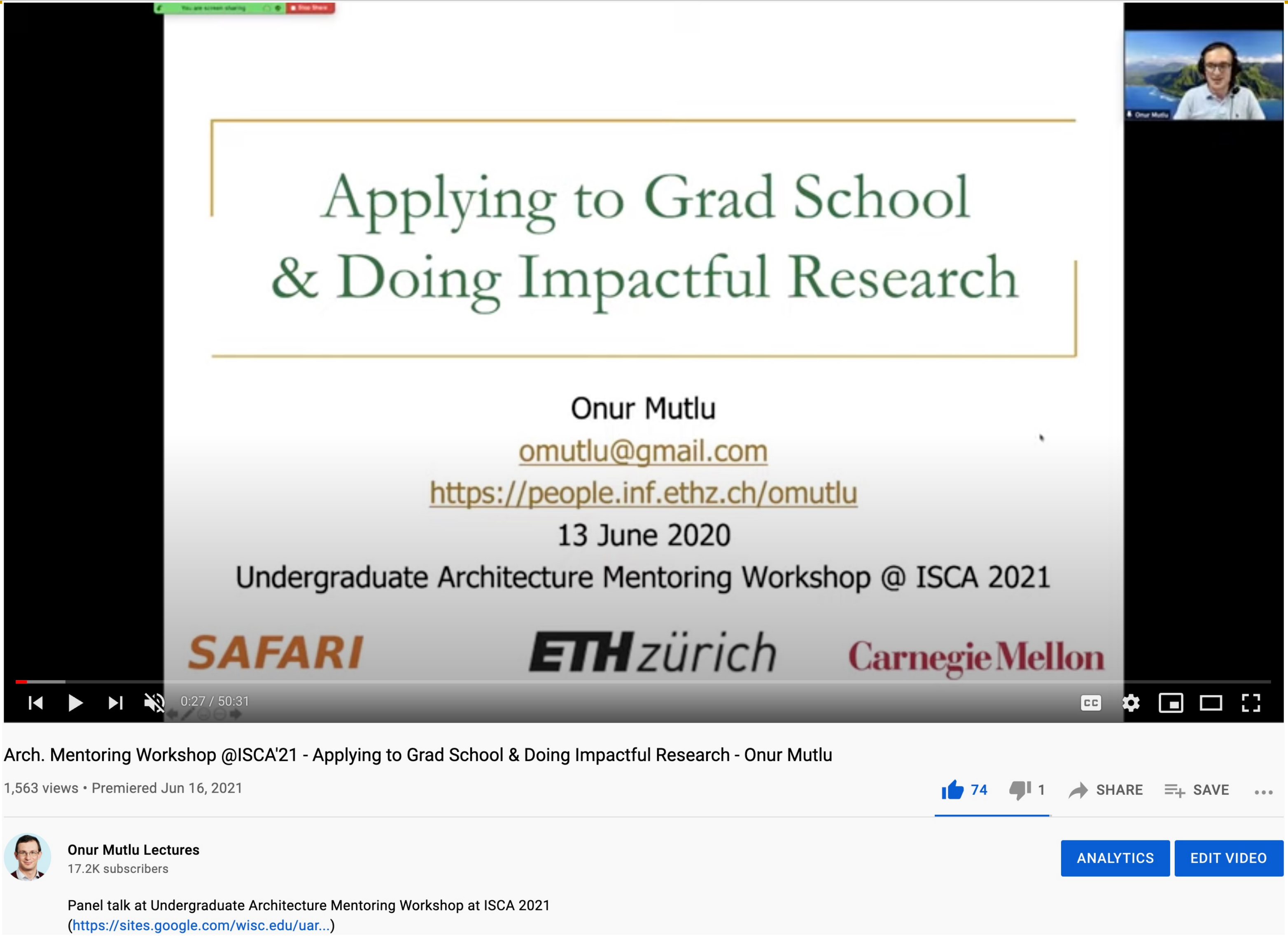
- Onur Mutlu,
  "How to Build an Impactful Research Group"
  *57th Design Automation Conference Early Career Workshop (**DAC**)*, Virtual, 19 July 2020.
  [Slides (pptx) (pdf)]

More Thoughts and Suggestions (II)

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

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

A Talk on Impactful Research & Teaching



Applying to Grad School & Doing Impactful Research

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

Arch. Mentoring Workshop @ISCA'21 - Applying to Grad School & Doing Impactful Research - Onur Mutlu

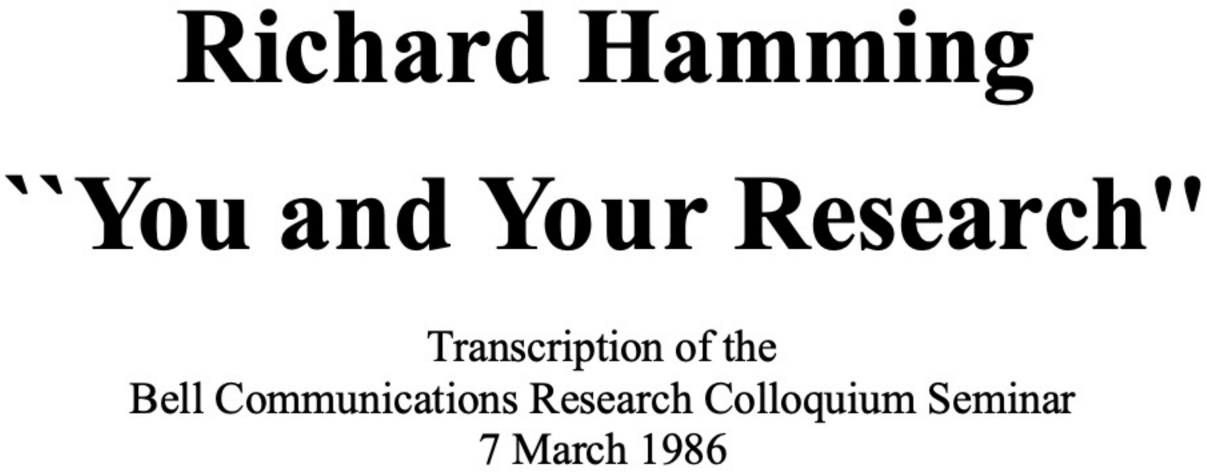
1,563 views • Premiered Jun 16, 2021

Onur Mutlu Lectures
17.2K subscribers

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

https://www.youtube.com/watch?v=83tlorht7Mc&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=54

Highly Recommended Reading



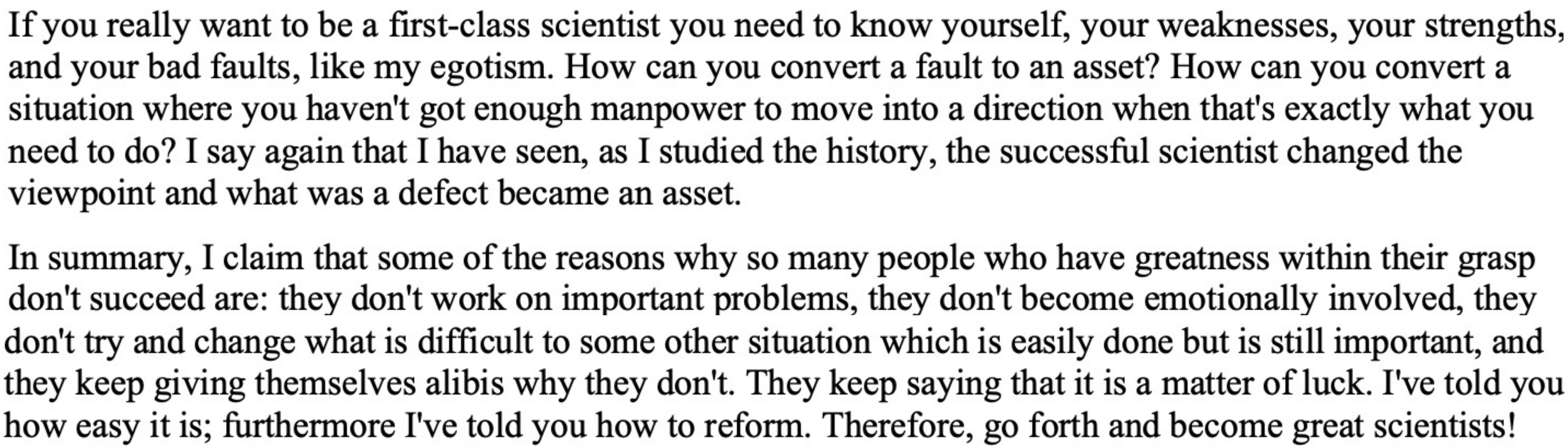
Richard Hamming

``You and Your Research''

Transcription of the
Bell Communications Research Colloquium Seminar
7 March 1986

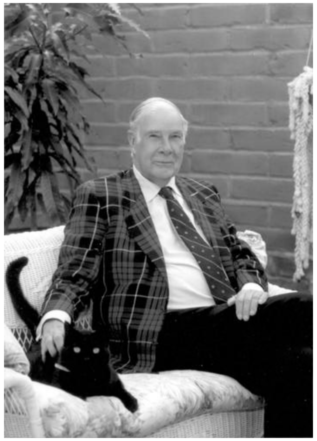
https://safari.ethz.ch/architecture/fall2021/lib/exe/fetch.php?media=youandyourresearch.pdf

Suggested Reading on Mindset & More



If you really want to be a first-class scientist you need to know yourself, your weaknesses, your strengths, and your bad faults, like my egotism. How can you convert a fault to an asset? How can you convert a situation where you haven't got enough manpower to move into a direction when that's exactly what you need to do? I say again that I have seen, as I studied the history, the successful scientist changed the viewpoint and what was a defect became an asset.

In summary, I claim that some of the reasons why so many people who have greatness within their grasp don't succeed are: they don't work on important problems, they don't become emotionally involved, they don't try and change what is difficult to some other situation which is easily done but is still important, and they keep giving themselves alibis why they don't. They keep saying that it is a matter of luck. I've told you how easy it is; furthermore I've told you how to reform. Therefore, go forth and become great scientists!



https://safari.ethz.ch/architecture/fall2021/lib/exe/fetch.php?media=youandyourresearch.pdf

Memory-Centric Computing Systems

The Problem

Computing

is Bottlenecked by Data

Data is Key for AI, ML, Genomics, …

- Important workloads are all data intensive

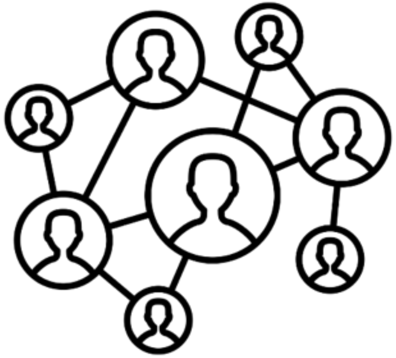
- They require rapid and efficient processing of large amounts of data

- Data is increasing
  - We can generate more than we can process

Data is Key for Future Workloads

In-memory Databases

[Mao+, EuroSys'12;
Clapp+ (**Intel**), IISWC'15]

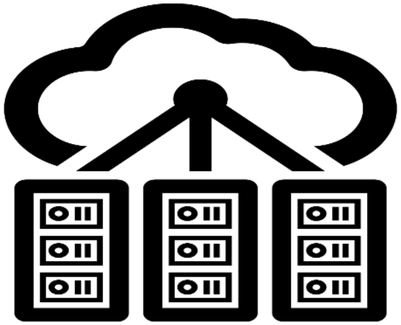


Graph/Tree Processing

[Xu+, IISWC'12; Umuroglu+, FPL'15]

In-Memory Data Analytics

[Clapp+ (**Intel**), IISWC'15;
Awan+, BDCloud'15]

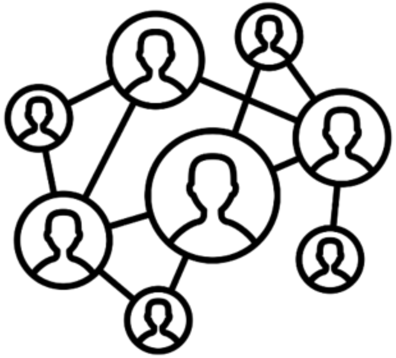


Datacenter Workloads

[Kanev+ (**Google**), ISCA'15]

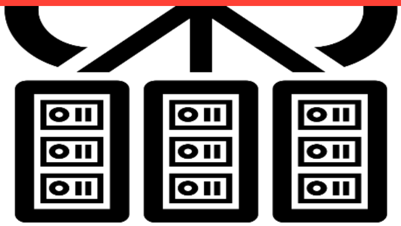
Data Overwhelms Modern Machines

In-memory Databases



Graph/Tree Processing

Data → performance & energy bottleneck



In-Memory Data Analytics

[Clapp+ (**Intel**), IISWC'15;
Awan+, BDCloud'15]

Datacenter Workloads

[Kanev+ (**Google**), ISCA'15]

Data is Key for Future Workloads

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

Video Playback

Google's video codec

Video Capture

Google's video codec

Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

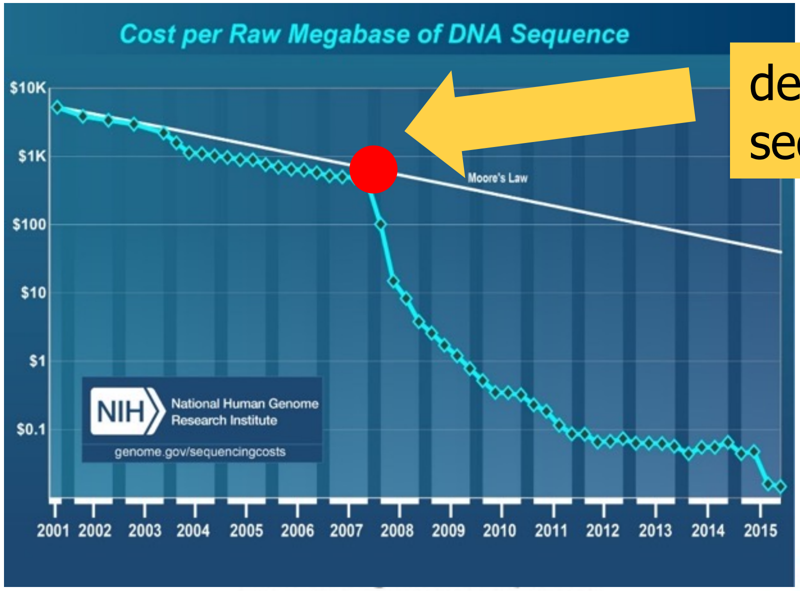
Video Playback

Google's **video codec**

Video Capture

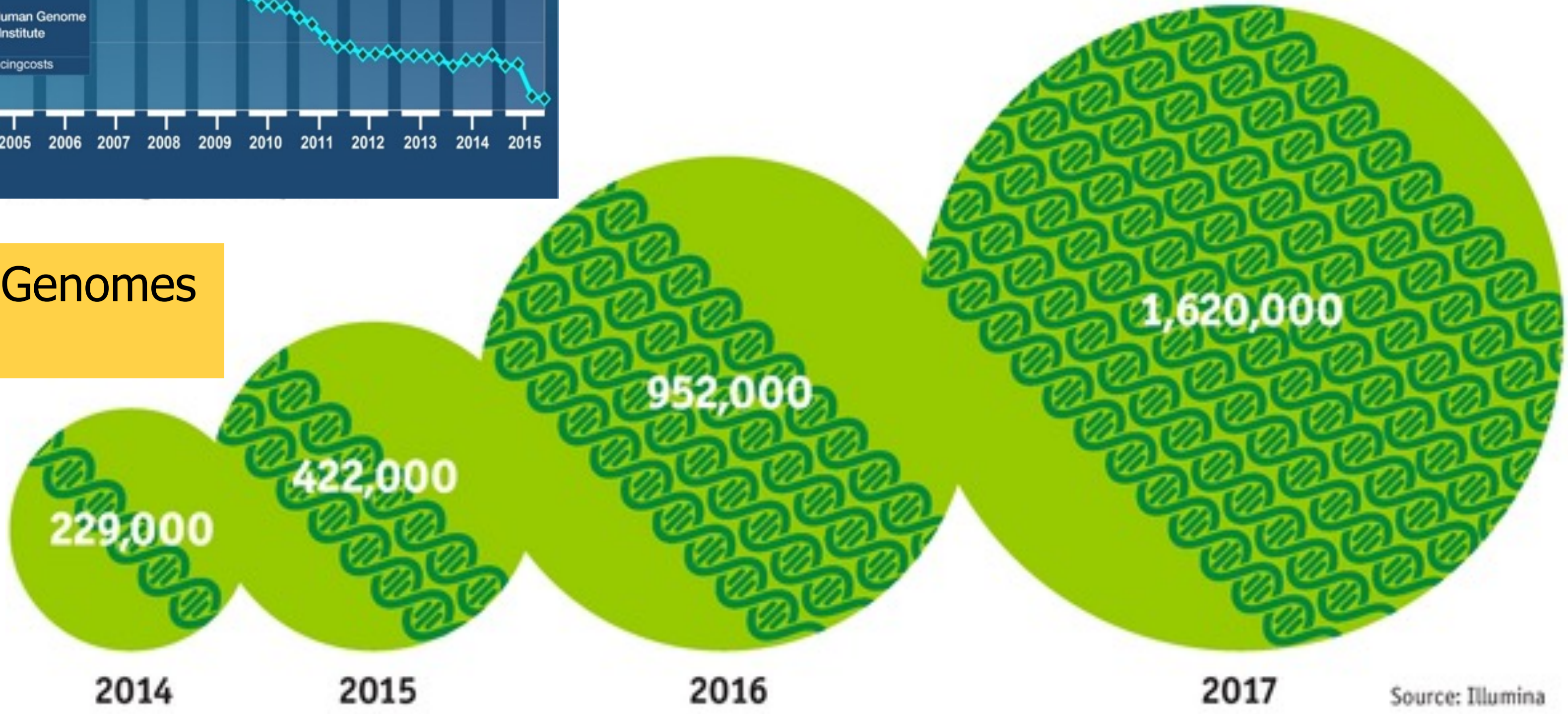
Google's **video codec**

Data is Key for Future Workloads



development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

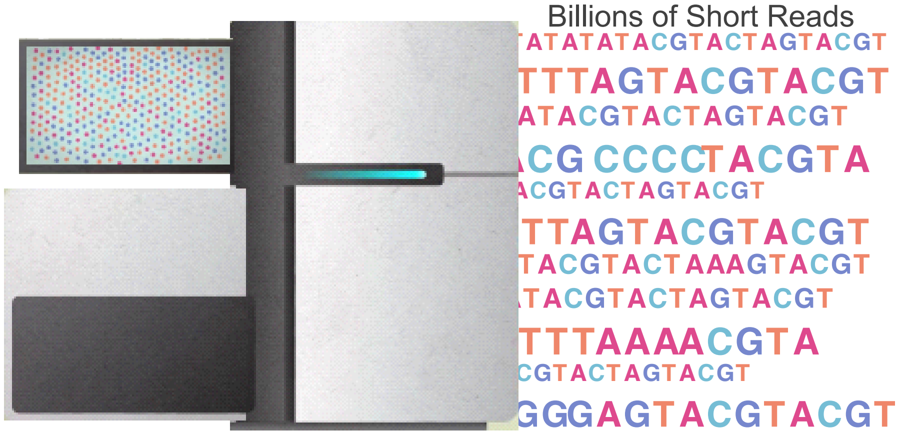


Source: Illumina

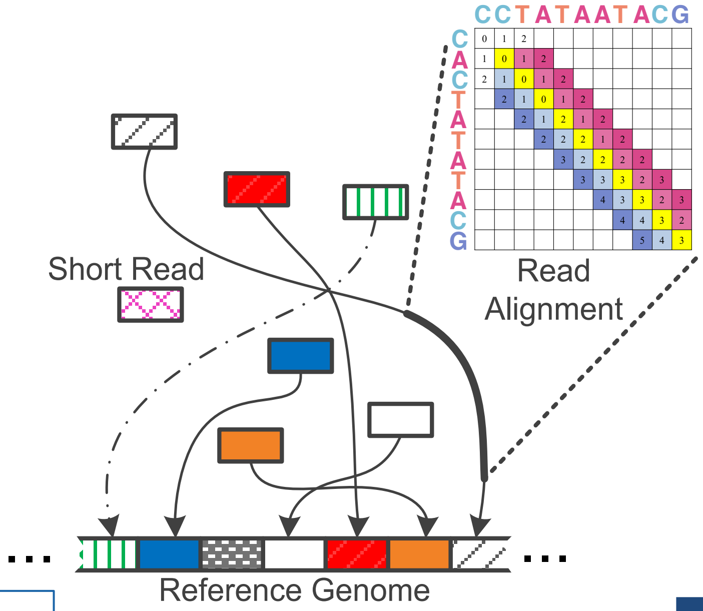
The Economist

SAFARI

http://www.economist.com/news/21631808-so-much-genetic-data-so-many-uses-genes-unzipped



1 Sequencing



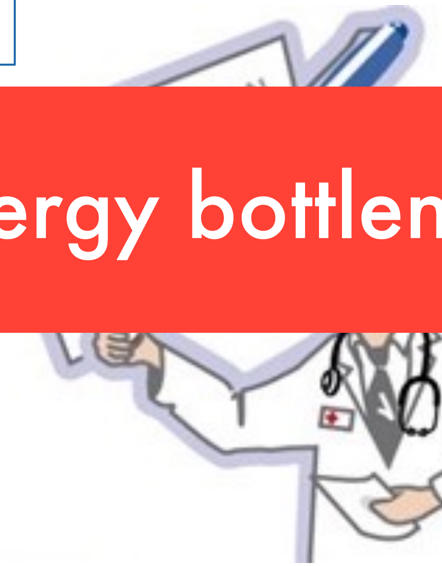
Read Mapping 2

Genome Analysis

Data → performance & energy bottleneck

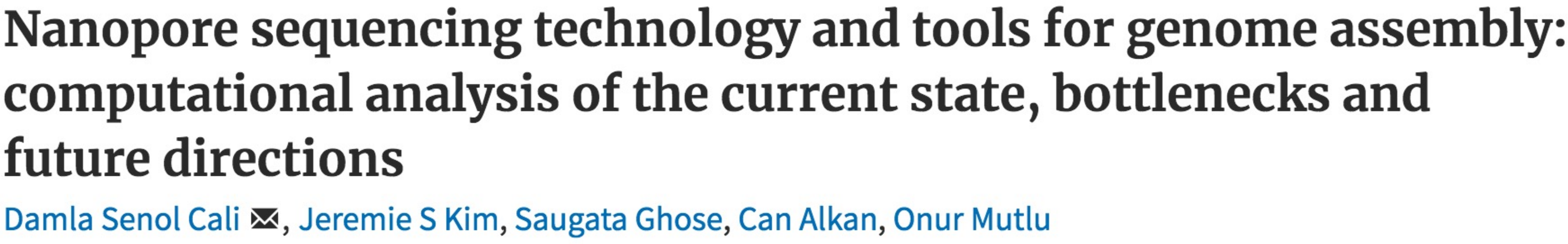
```
read4:          CGCTTCCAT
read5:              CCATGACGC
read6:            TTCCATGAC
```

3 Variant Calling



Scientific Discovery 4

New Genome Sequencing Technologies



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

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

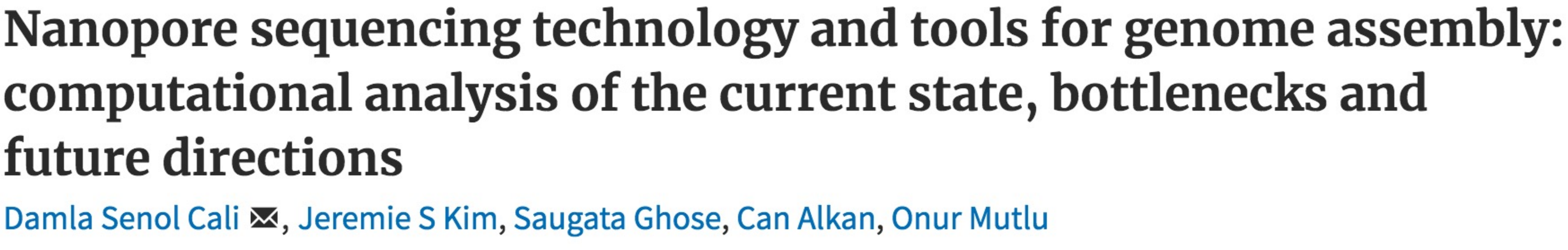
Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017

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

Oxford Nanopore MinION

Senol Cali+, "**Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions**," Briefings in Bioinformatics, 2018.
[Open arxiv.org version]

New Genome Sequencing Technologies



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

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

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

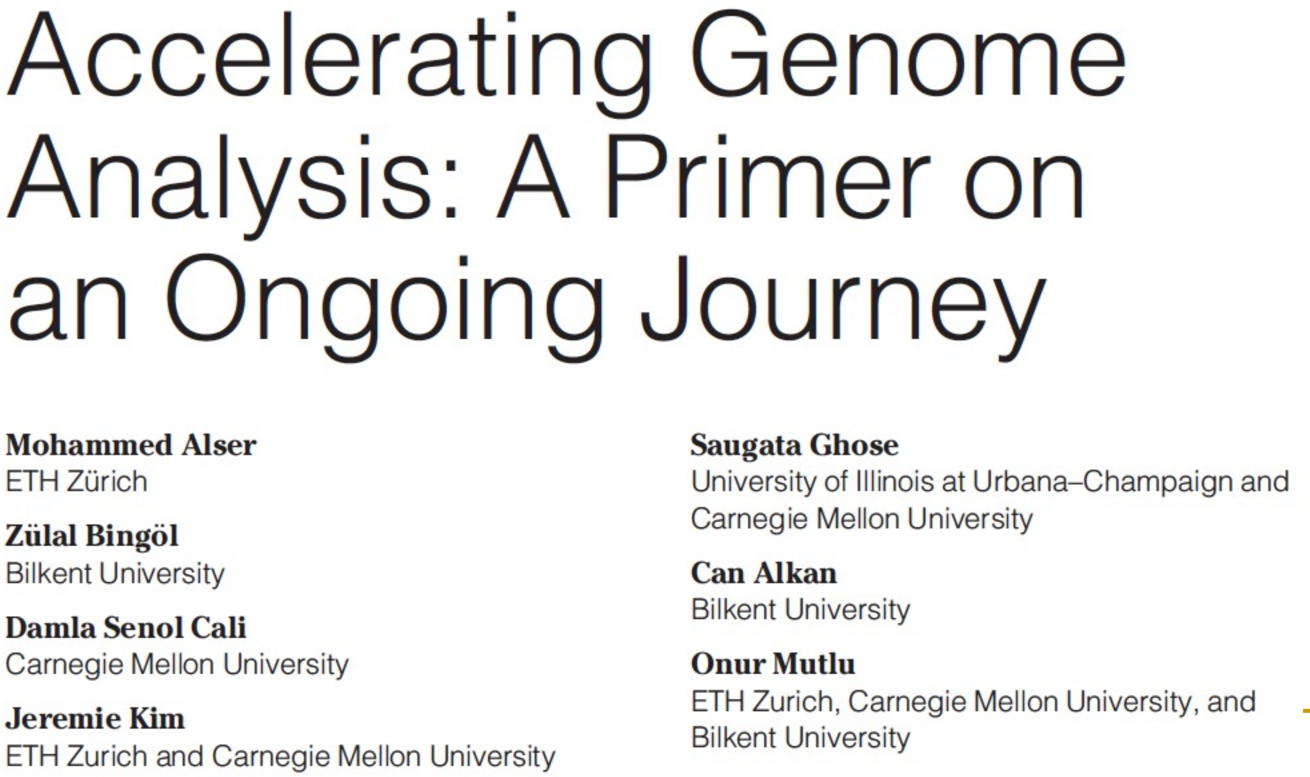
Published: 02 April 2018 **Article history** ▾

Oxford Nanopore MinION

Data → performance & energy bottleneck

Accelerating Genome Analysis

- Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  IEEE Micro (***IEEE MICRO***), Vol. 40, No. 5, pages 65-75, September/October 2020.
  [Slides (pptx)(pdf)]
  [Talk Video (1 hour 2 minutes)]



Accelerating Genome Analysis: A Primer on an Ongoing Journey

Mohammed Alser
ETH Zürich

Zülal Bingöl
Bilkent University

Damla Senol Cali
Carnegie Mellon University

Jeremie Kim
ETH Zurich and Carnegie Mellon University

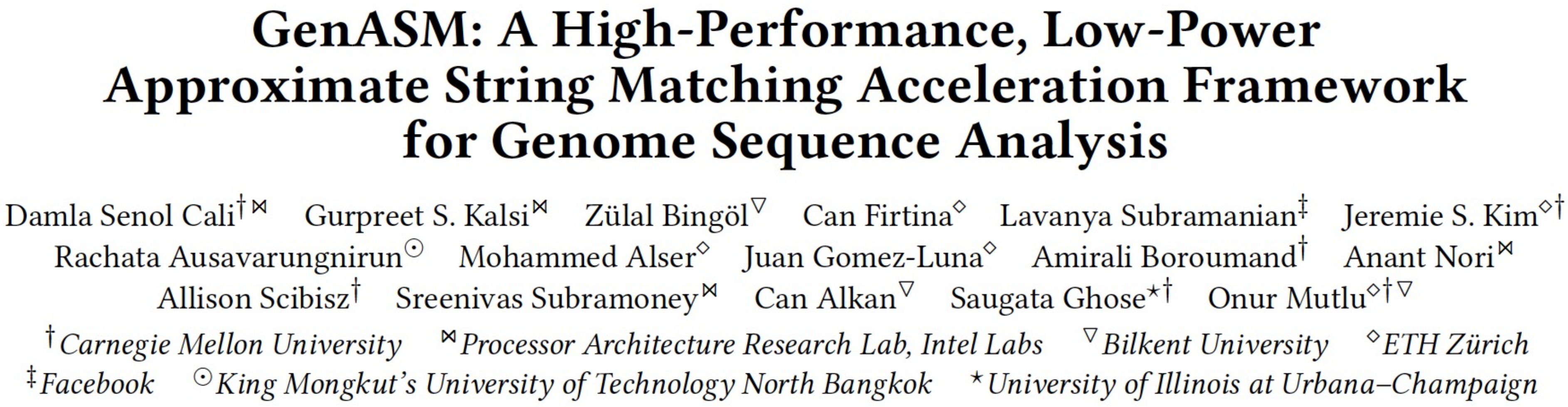
Saugata Ghose
University of Illinois at Urbana–Champaign and Carnegie Mellon University

Can Alkan
Bilkent University

Onur Mutlu
ETH Zurich, Carnegie Mellon University, and Bilkent University

GenASM Framework **[MICRO 2020]**

- Damla Senol Cali, Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,
  "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"
  Proceedings of the 53rd International Symposium on Microarchitecture (***MICRO***), Virtual, October 2020.
  [Lighting Talk Video (1.5 minutes)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (18 minutes)]
  [Slides (pptx) (pdf)]



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

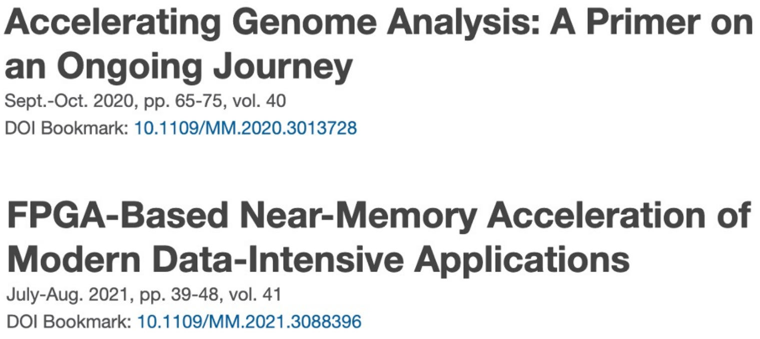
Damla Senol Cali[†⋈] Gurpreet S. Kalsi[⋈] Zülal Bingöl[▽] Can Firtina[◇] Lavanya Subramanian[‡] Jeremie S. Kim[◇†]
Rachata Ausavarungnirun[⊙] Mohammed Alser[◇] Juan Gomez-Luna[◇] Amirali Boroumand[†] Anant Nori[⋈]
Allison Scibisz[†] Sreenivas Subramoney[⋈] Can Alkan[▽] Saugata Ghose[⋆†] Onur Mutlu[◇†▽]

[†] *Carnegie Mellon University* [⋈] *Processor Architecture Research Lab, Intel Labs* [▽] *Bilkent University* [◇] *ETH Zürich*
[‡] *Facebook* [⊙] *King Mongkut's University of Technology North Bangkok* [⋆] *University of Illinois at Urbana–Champaign*

Future of Genome Sequencing & Analysis

Mohammed Alser, Zülal Bingöl, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, Onur Mutlu
"Accelerating Genome Analysis: A Primer on an Ongoing Journey" IEEE Micro, August 2020.

MinION from ONT



Accelerating Genome Analysis: A Primer on an Ongoing Journey
Sept.-Oct. 2020, pp. 65-75, vol. 40
DOI Bookmark: 10.1109/MM.2020.3013728

FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications
July-Aug. 2021, pp. 39-48, vol. 41
DOI Bookmark: 10.1109/MM.2021.3088396

SmidgION from ONT

Detailed Lectures on Genome Analysis

- Computer Architecture, Fall 2020, Lecture 3a
  - **Introduction to Genome Sequence Analysis** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5
- Computer Architecture, Fall 2020, Lecture 8
  - **Intelligent Genome Analysis** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14
- Computer Architecture, Fall 2020, Lecture 9a
  - **GenASM: Approx. String Matching Accelerator** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=XoLpzmN-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15
- Accelerating Genomics Project Course, Fall 2020, Lecture 1
  - **Accelerating Genomics** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqLgwiDRQDTyId

More on Fast & Efficient Genome Analysis …

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

Population-Scale Microbiome Profiling

SAFARI https://blog.wego.com/7-crowded-places-and-events-that-you-will-love/ 30

Onur Mutlu - Invited Lecture @Technion: Accelerating Genome Analysis: A Primer on an Ongoing Journey

740 views • Premiered Feb 6, 2021

Onur Mutlu Lectures
15.9K subscribers

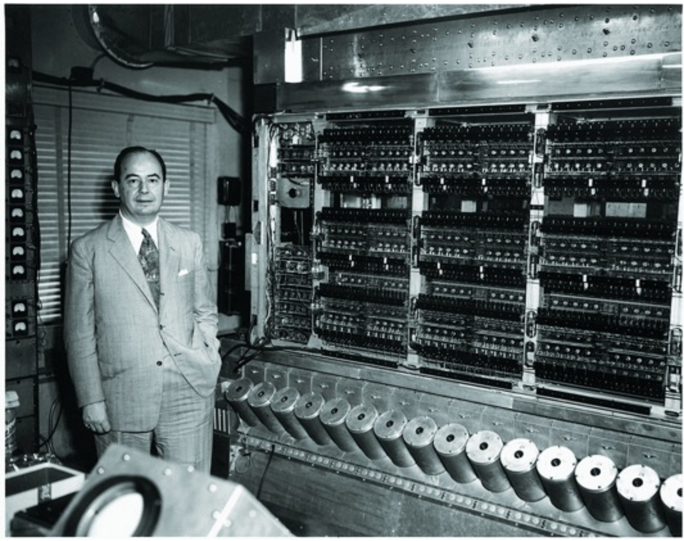
https://www.youtube.com/watch?v=r7sn41lH-4A

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.

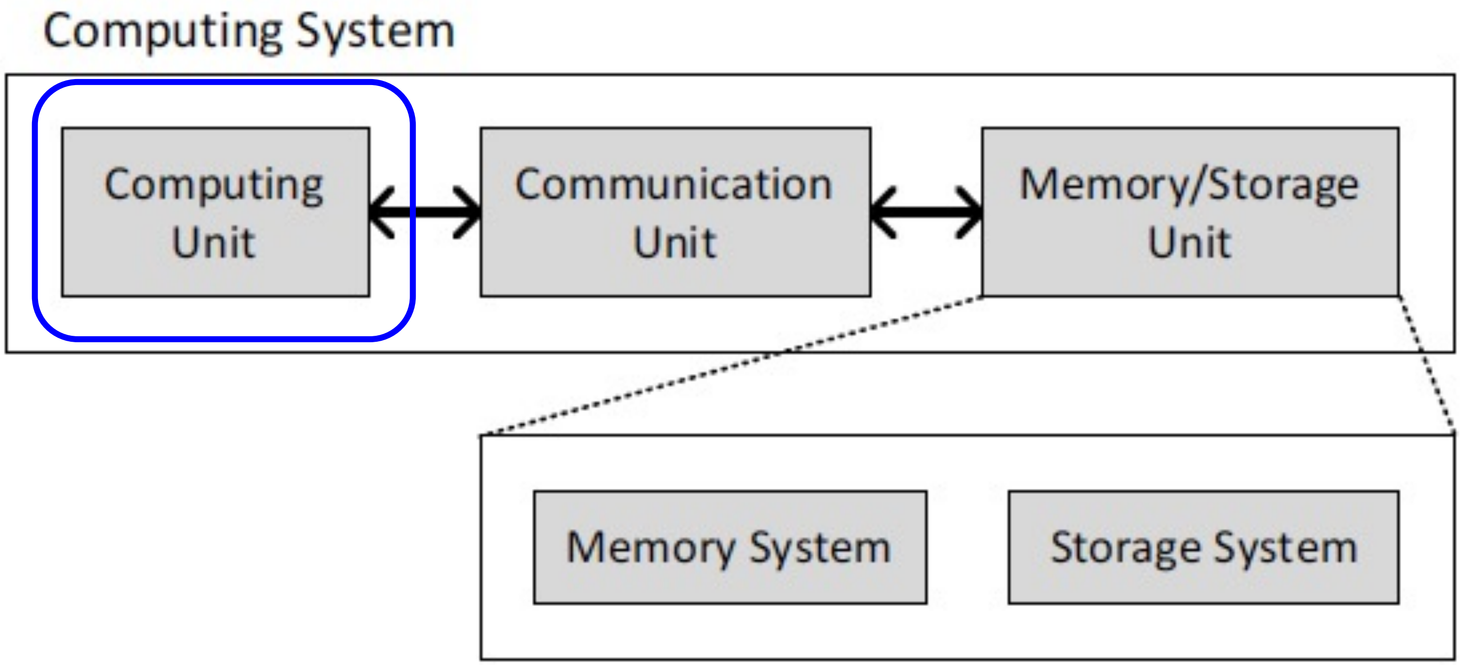
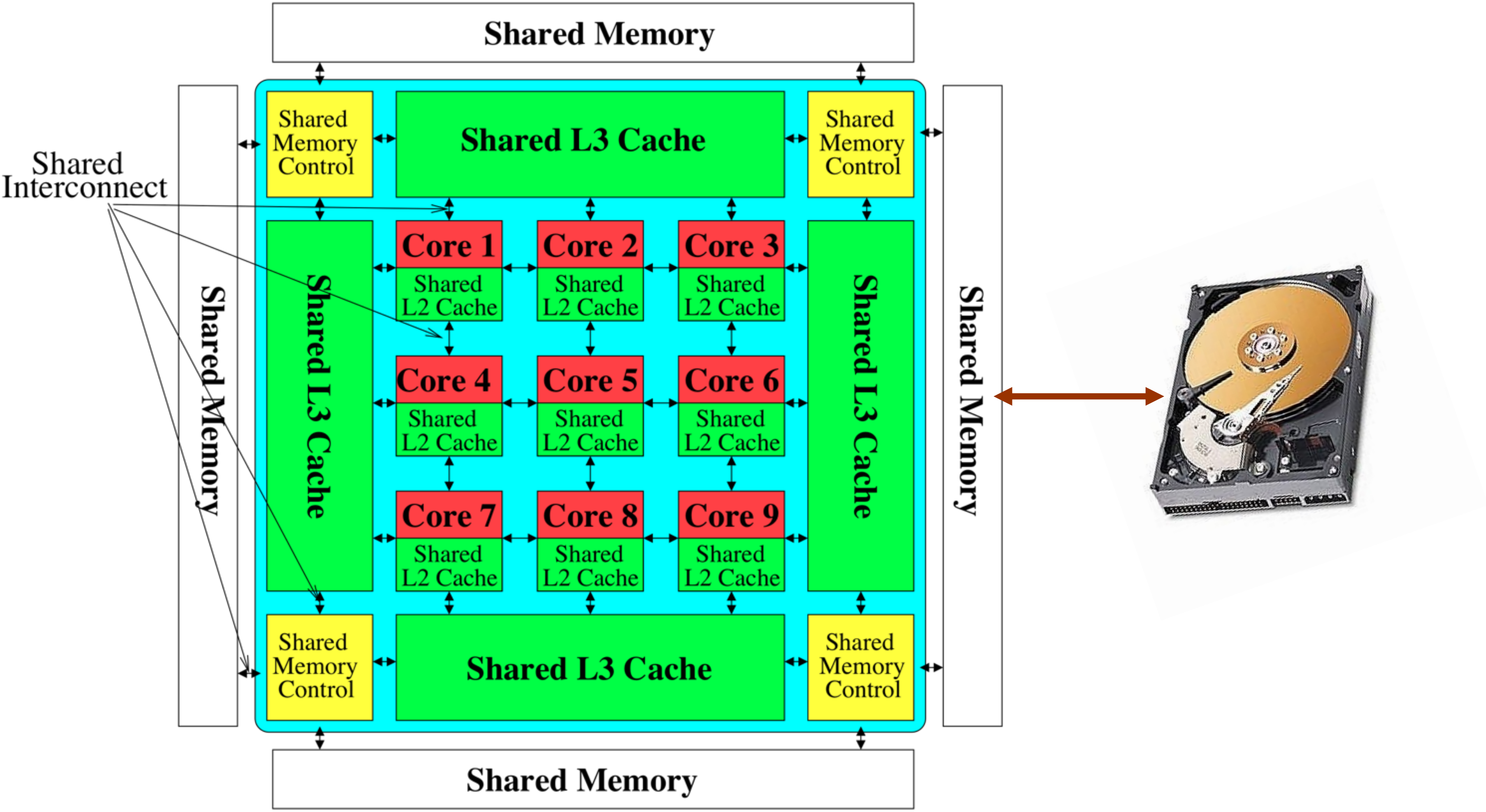


Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

Perils of Processor-Centric Design



Most of the system is dedicated to storing and moving data

Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

Video Playback

Google's **video codec**

Video Capture

Google's **video codec**

Data Movement Overwhelms Modern Machines

- 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"** *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems* (***ASPLOS***), Williamsburg, VA, USA, March 2018.

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

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

Axiom

An Intelligent Architecture

Handles Data Well

How to Handle Data Well

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

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

- Understand and exploit properties of (different) data
  - to improve algorithms & architectures in various metrics

Corollaries: Architectures Today …

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

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

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

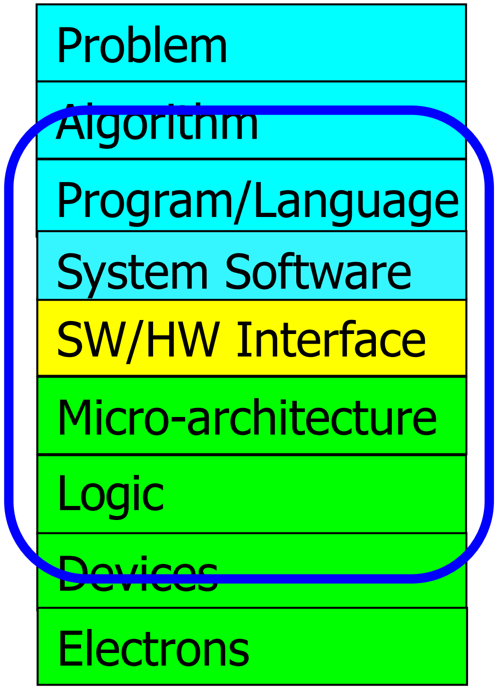
Fundamentally Better Architectures

Data-centric

Data-driven

Data-aware

We Need to Revisit the Entire Stack



We can get there step by step

Data-Centric (Memory-Centric) Architectures

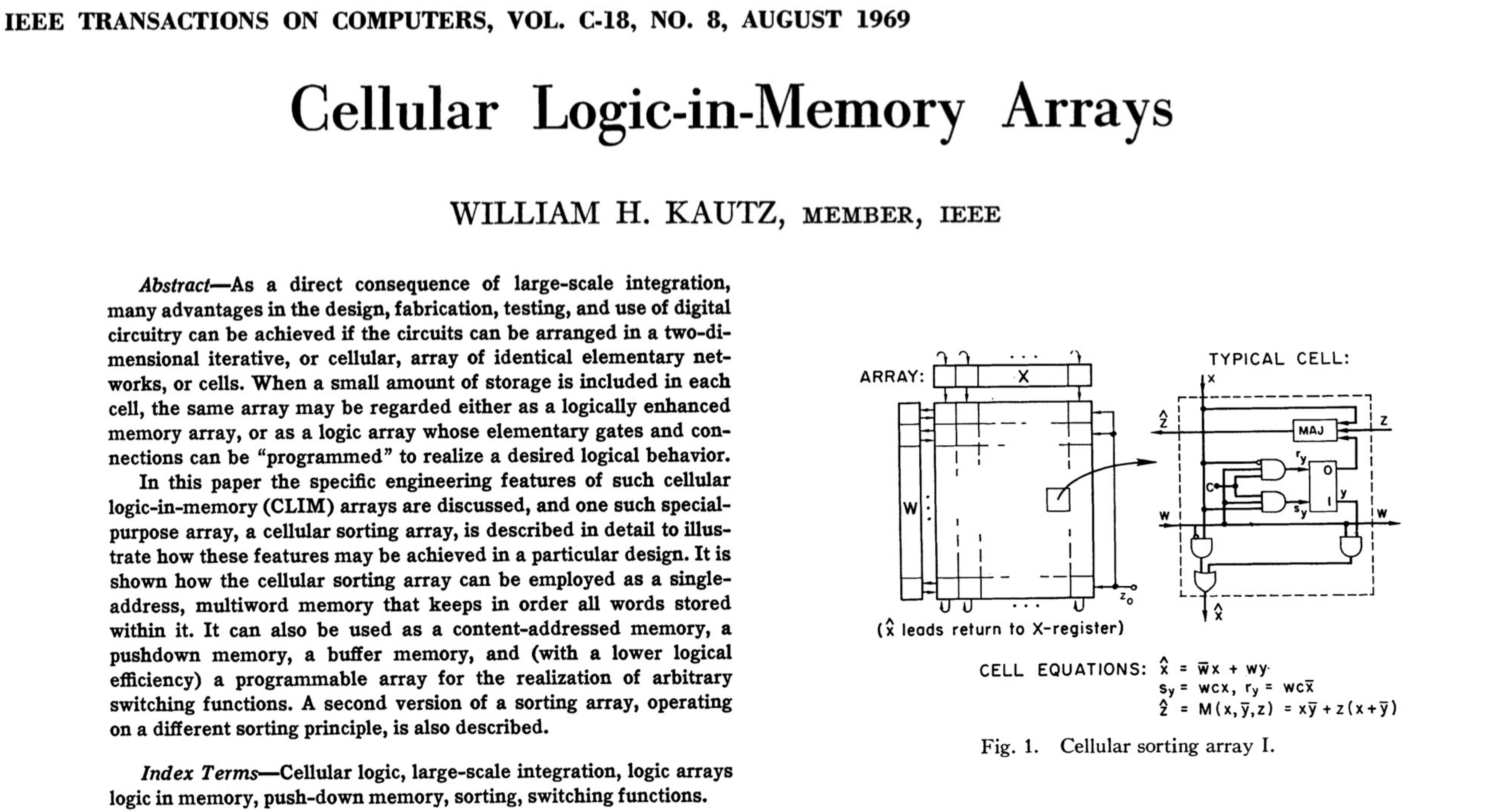
Data-Centric Architectures: Properties

- **Process data where it resides** (where it makes sense)
  - Processing in and near memory structures
- **Low-latency and low-energy data access**
  - Low latency memory
  - Low energy memory
- **Low-cost data storage and processing**
  - High capacity memory at low cost: hybrid memory, compression
- **Intelligent data management**
  - Intelligent controllers handling robustness, security, cost

Processing Data Where It Makes Sense

Processing in/near Memory: An Old Idea

- Kautz, "Cellular Logic-in-Memory Arrays", IEEE TC 1969.



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

Cellular Logic-in-Memory Arrays

WILLIAM H. KAUTZ, MEMBER, IEEE

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

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

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

($\hat{x}$ leads return to X-register)

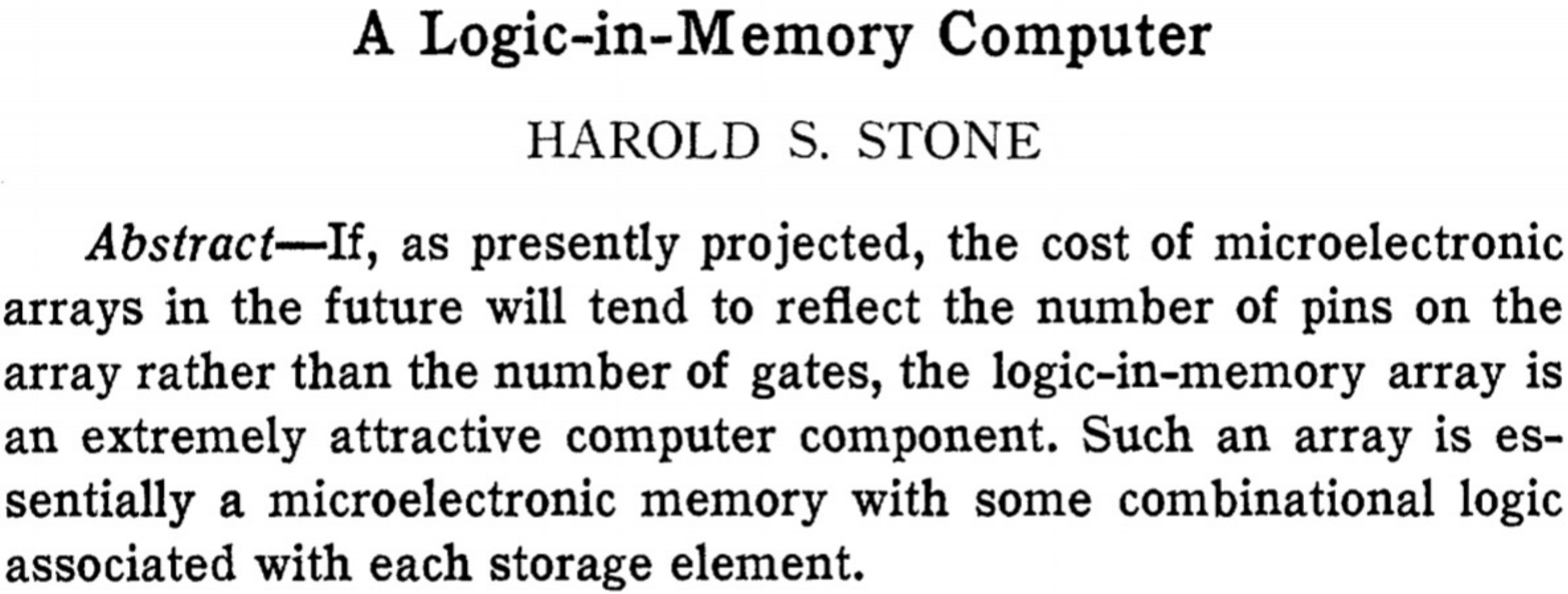
CELL EQUATIONS: $\hat{x} = \bar{w}x + wy$
$s_y = wcx, \; r_y = wc\bar{x}$
$\hat{z} = M(x, \bar{y}, z) = x\bar{y} + z(x + \bar{y})$

Fig. 1. Cellular sorting array I.

https://doi.org/10.1109/T-C.1969.222754

Processing in/near Memory: An Old Idea

- Stone, “A Logic-in-Memory Computer,” IEEE TC 1970.



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.

https://safari.ethz.ch/architecture/fall2020/lib/exe/fetch.php?media=stone_logic_in_memory_1970.pdf

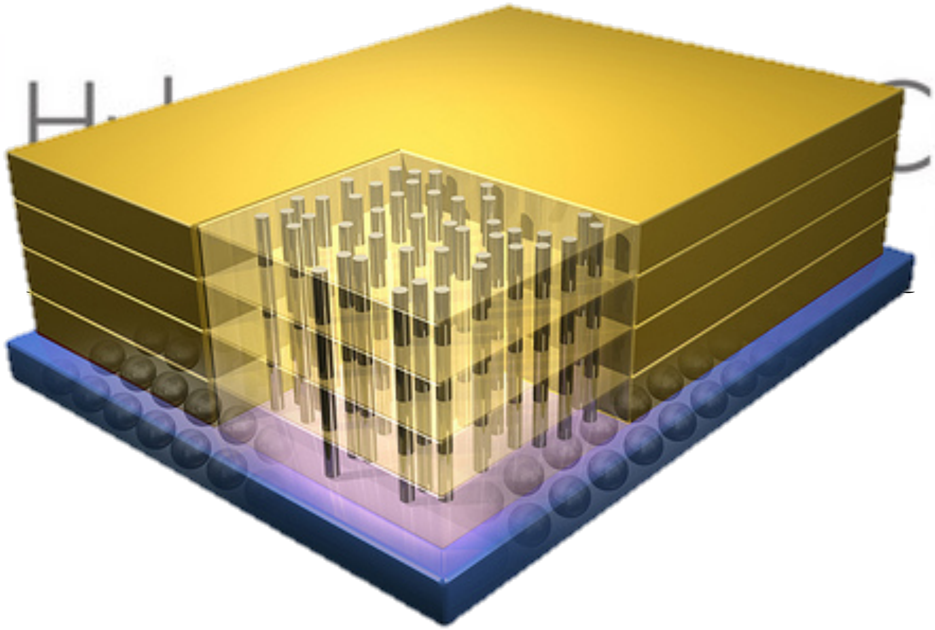
Why In-Memory Computation Today?

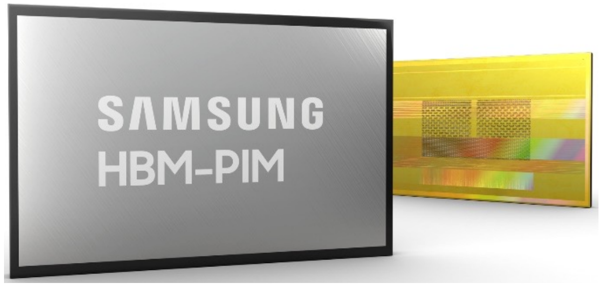
- Push from Technology
  - DRAM Scaling at jeopardy

    → Controllers close to DRAM

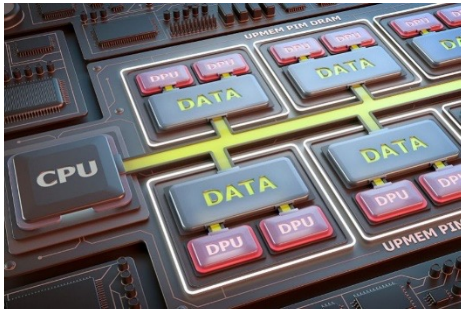
    → Industry open to new memory architectures

Why In-Memory Computation Today?





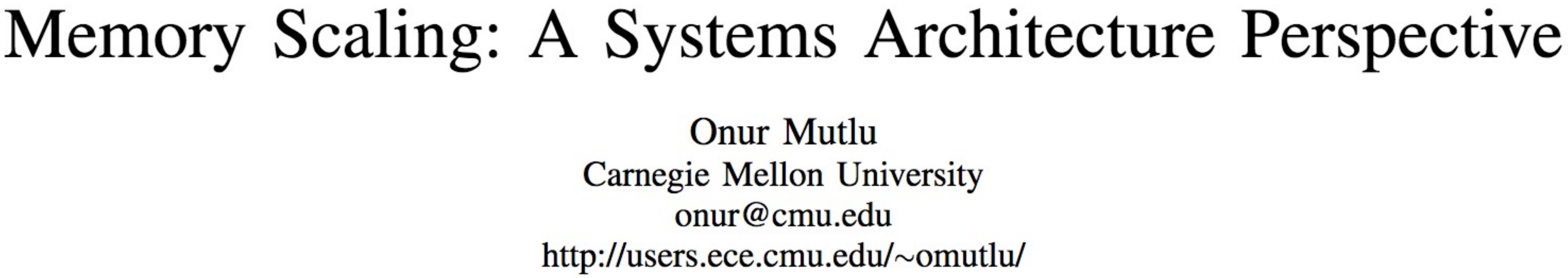
[Samsung 2021]



[UPMEM 2019]

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



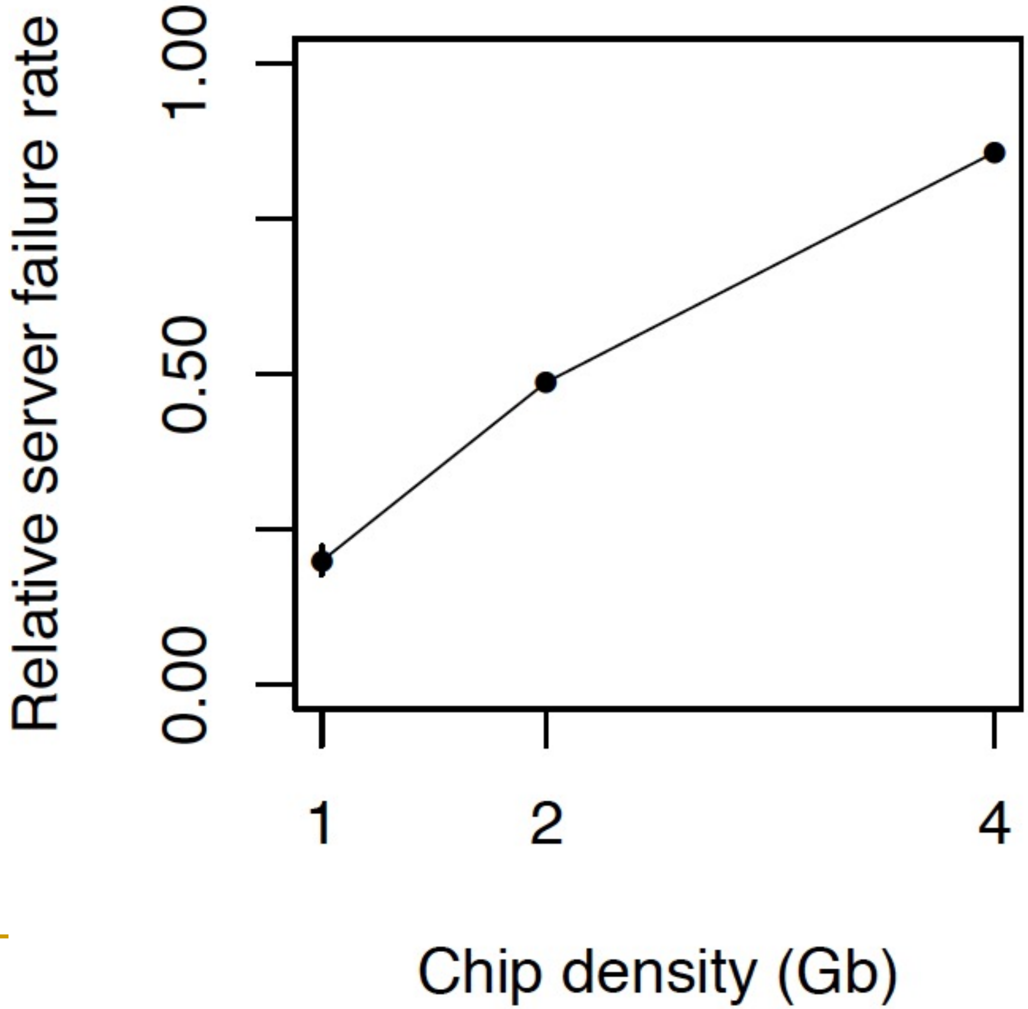
Memory Scaling: A Systems Architecture Perspective

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

https://people.inf.ethz.ch/omutlu/pub/memory-scaling_memcon13.pdf

As Memory Scales, It Becomes Unreliable

- Data from all of Facebook's servers worldwide
- Meza+, "Revisiting Memory Errors in Large-Scale Production Data Centers," DSN'15.

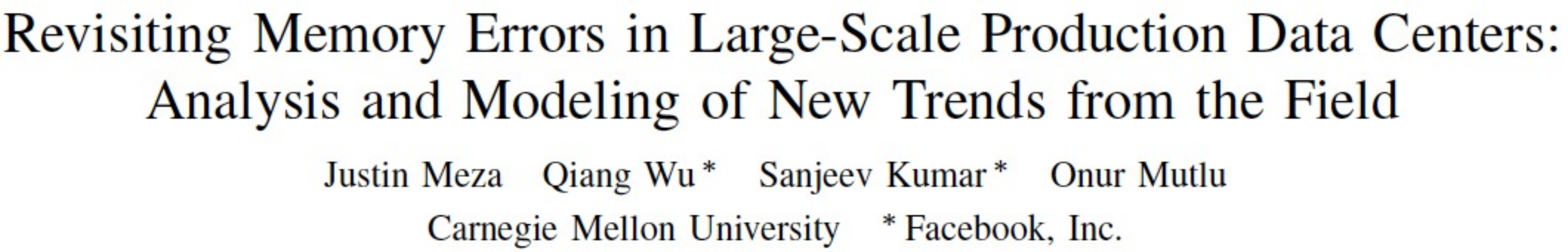


Intuition: quadratic increase in capacity

Large-Scale Failure Analysis of DRAM Chips

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

- Justin Meza, Qiang Wu, Sanjeev Kumar, and Onur Mutlu, **"Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field"** *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**)*, Rio de Janeiro, Brazil, June 2015.
  [Slides (pptx) (pdf)] [DRAM Error Model]

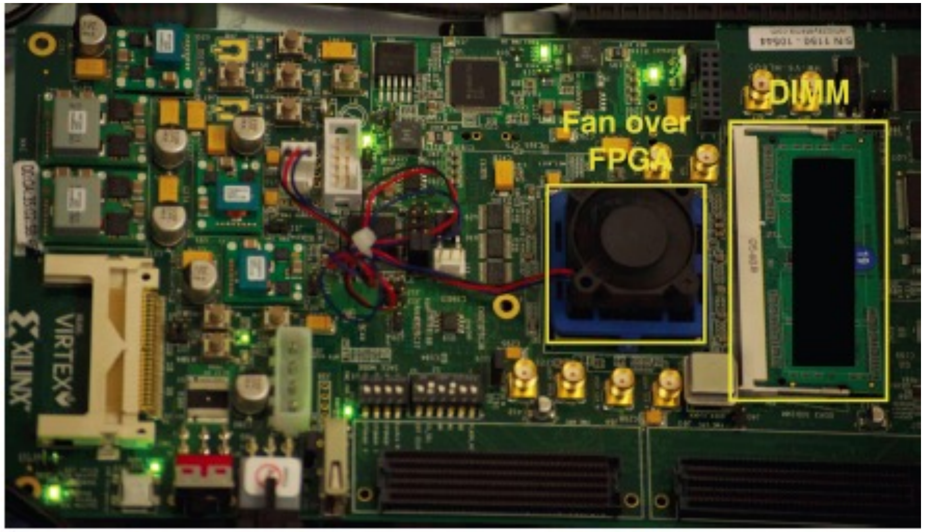


Revisiting Memory Errors in Large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field

Justin Meza Qiang Wu * Sanjeev Kumar * Onur Mutlu

Carnegie Mellon University * Facebook, Inc.

Infrastructures to Understand Such Issues



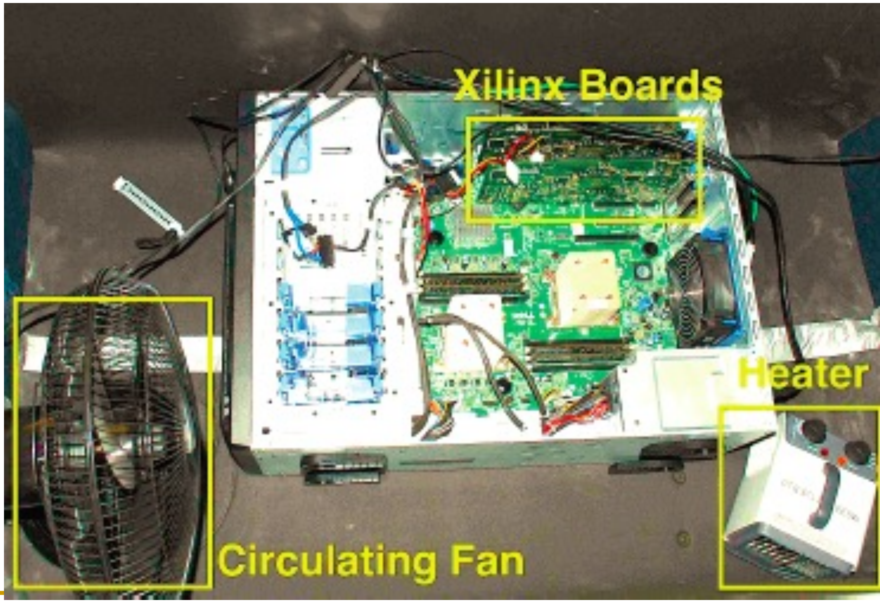
An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms (Liu et al., ISCA 2013)

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

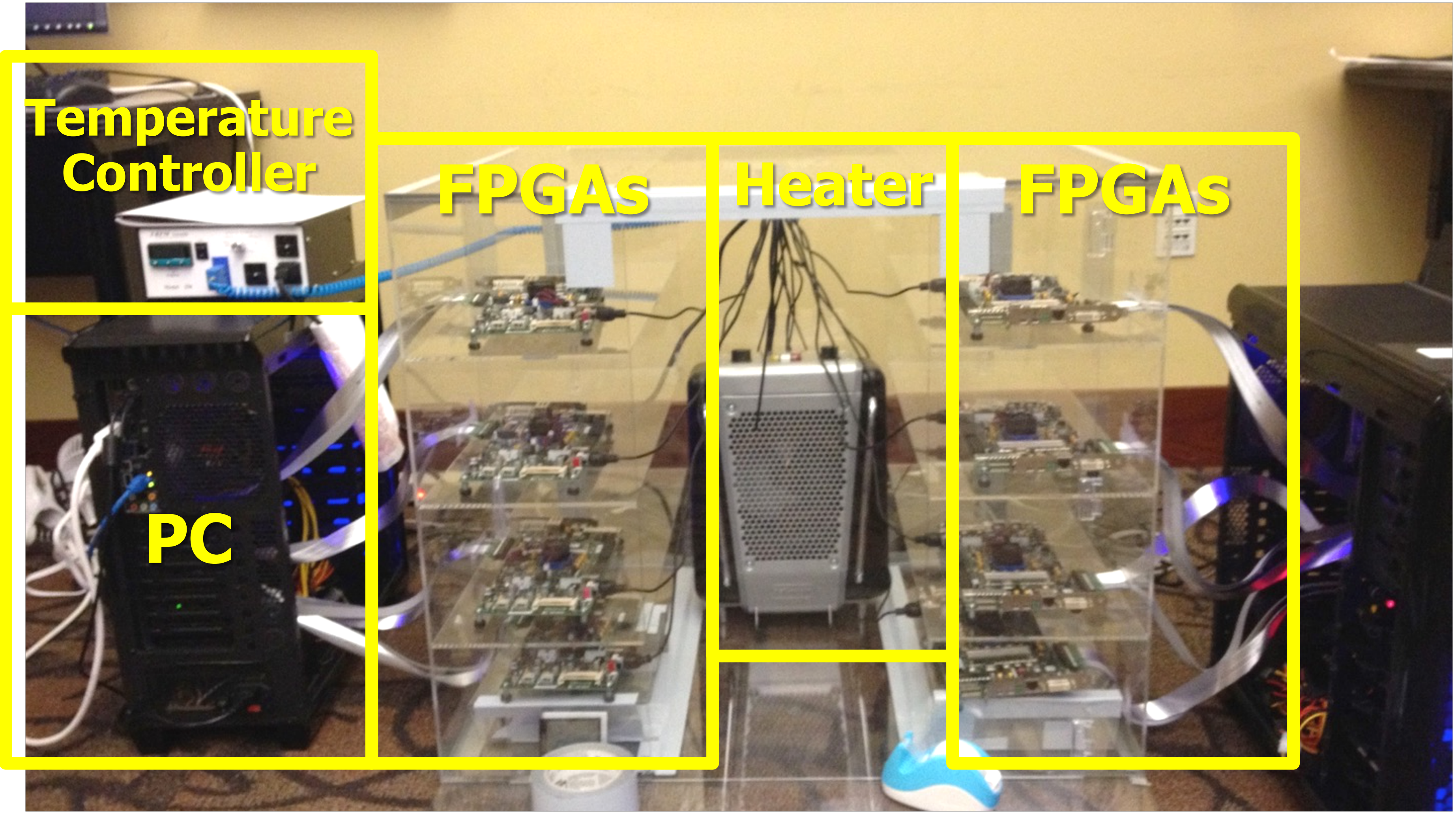
Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)

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

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



Infrastructures to Understand Such Issues



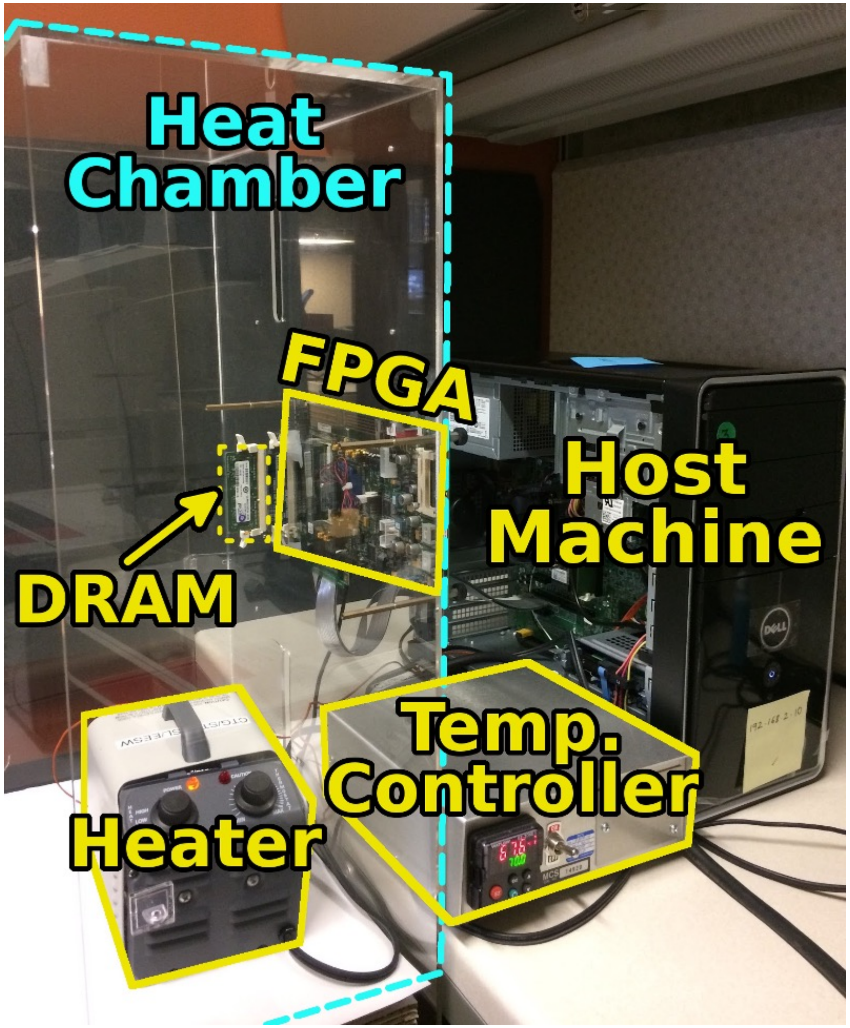
Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

SoftMC: Open Source DRAM Infrastructure

- Hasan Hassan et al., "**SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies,**" HPCA 2017.

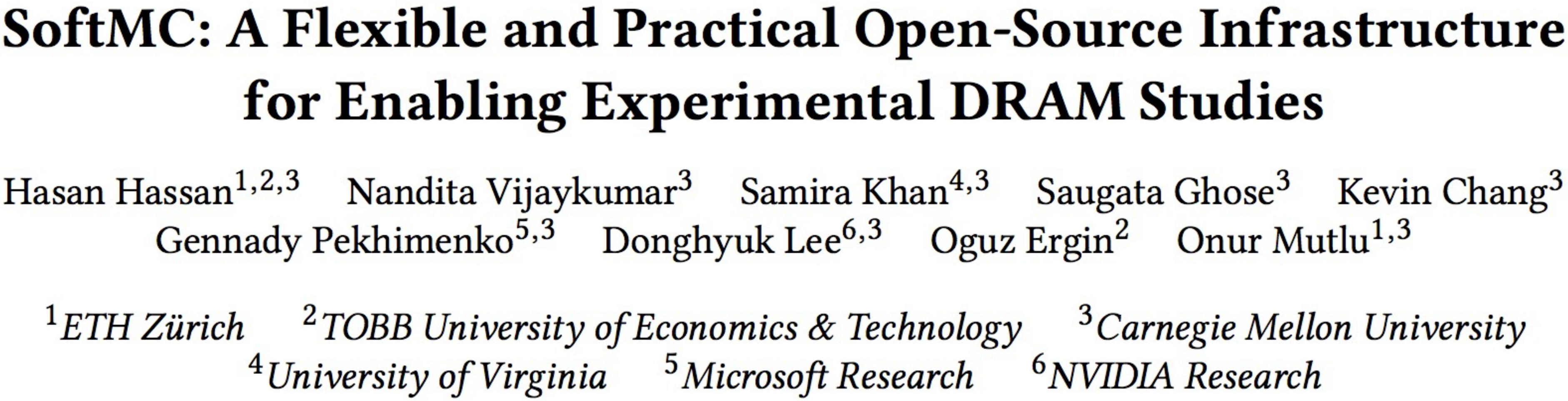
- **Flexible**
- **Easy to Use (C++ API)**
- **Open-source**

  github.com/CMU-SAFARI/SoftMC



SoftMC

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



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

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

[1]*ETH Zürich* [2]*TOBB University of Economics & Technology* [3]*Carnegie Mellon University*
[4]*University of Virginia* [5]*Microsoft Research* [6]*NVIDIA Research*

A Curious Discovery **[Kim et al., ISCA 2014]**

One can

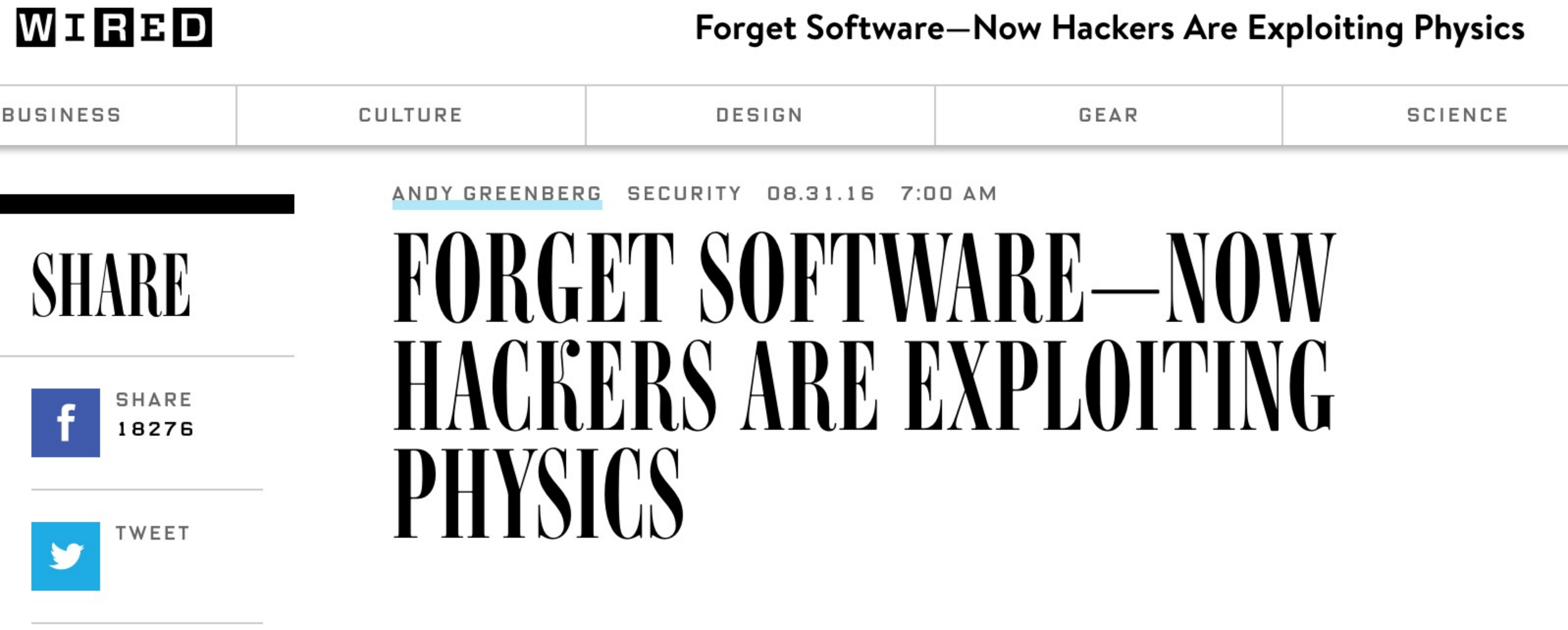
predictably induce errors

in most DRAM memory chips

The Story of RowHammer

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

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



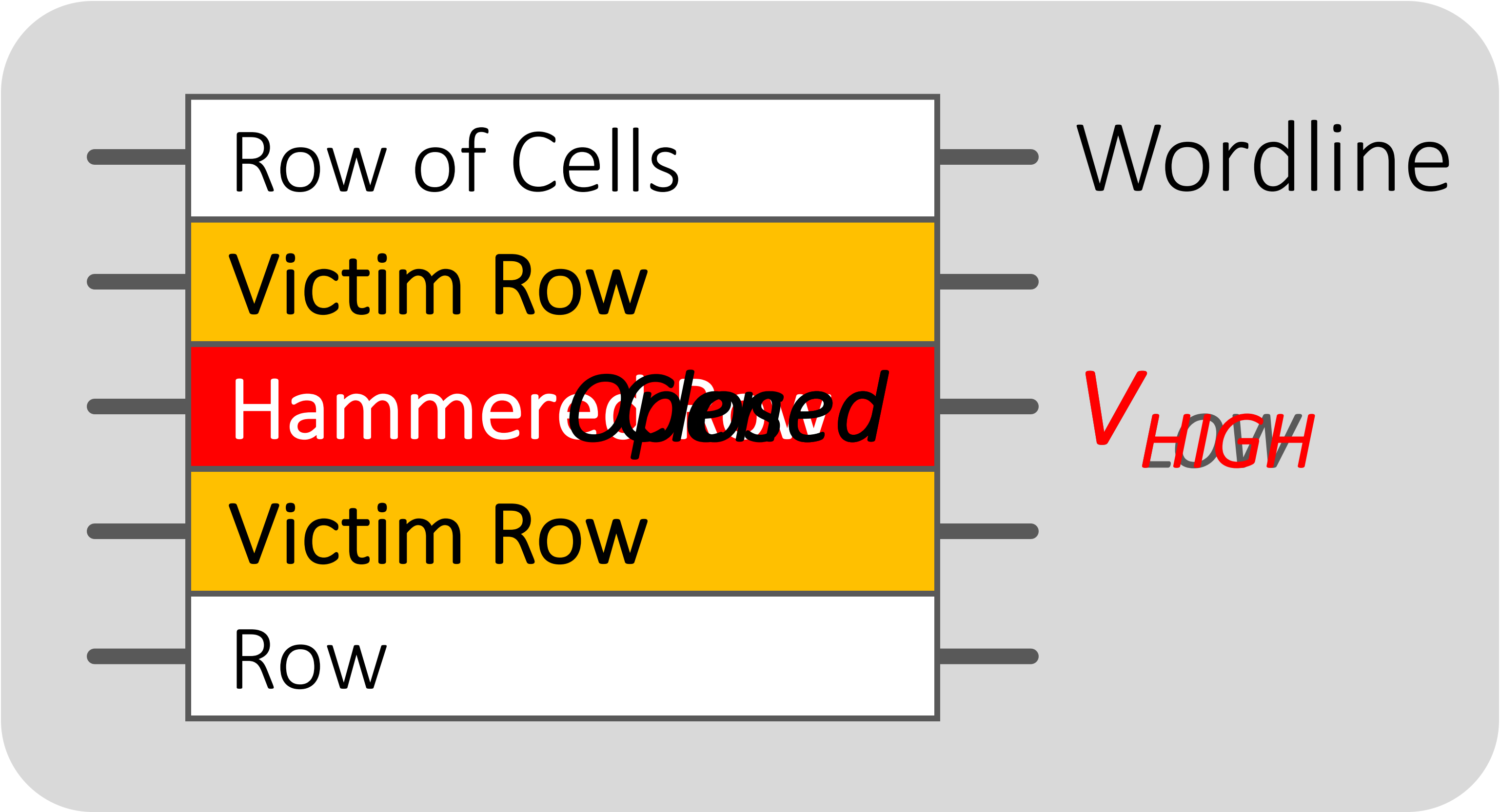
WIRED

Forget Software—Now Hackers Are Exploiting Physics

ANDY GREENBERG SECURITY 08.31.16 7:00 AM

FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS

Modern DRAM is Prone to Disturbance Errors

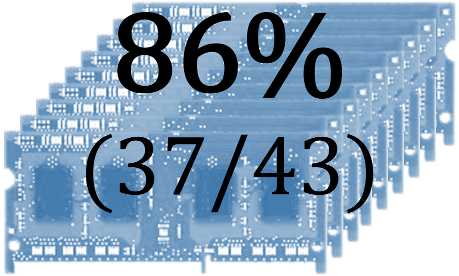


Repeatedly reading a row enough times (before memory gets refreshed) induces disturbance errors in **adjacent rows** in most real DRAM chips you can buy today

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

Most DRAM Modules Are Vulnerable

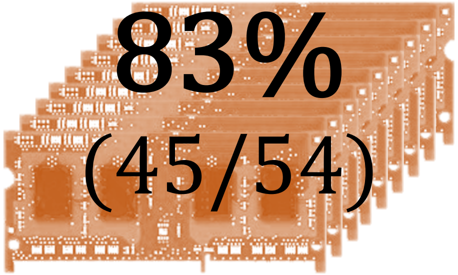
A company



86%
(37/43)

Up to
1.0×10^{7}
errors

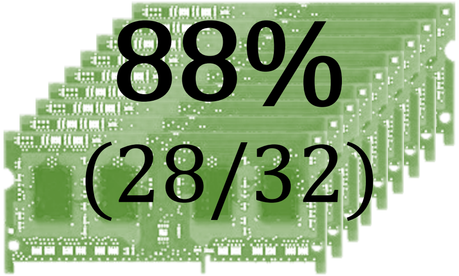
B company



83%
(45/54)

Up to
2.7×10^{6}
errors

C company

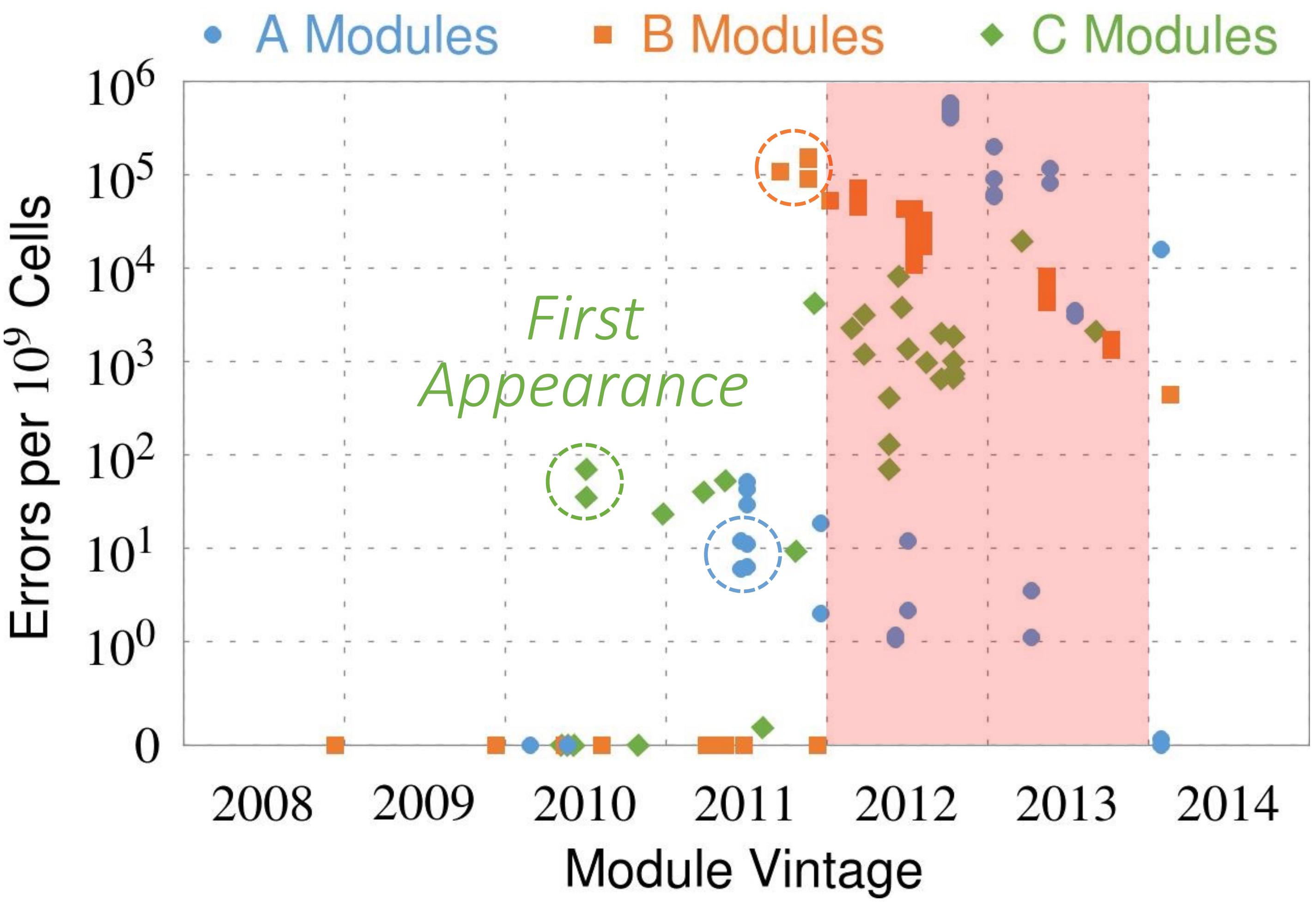


88%
(28/32)

Up to
3.3×10^{5}
errors

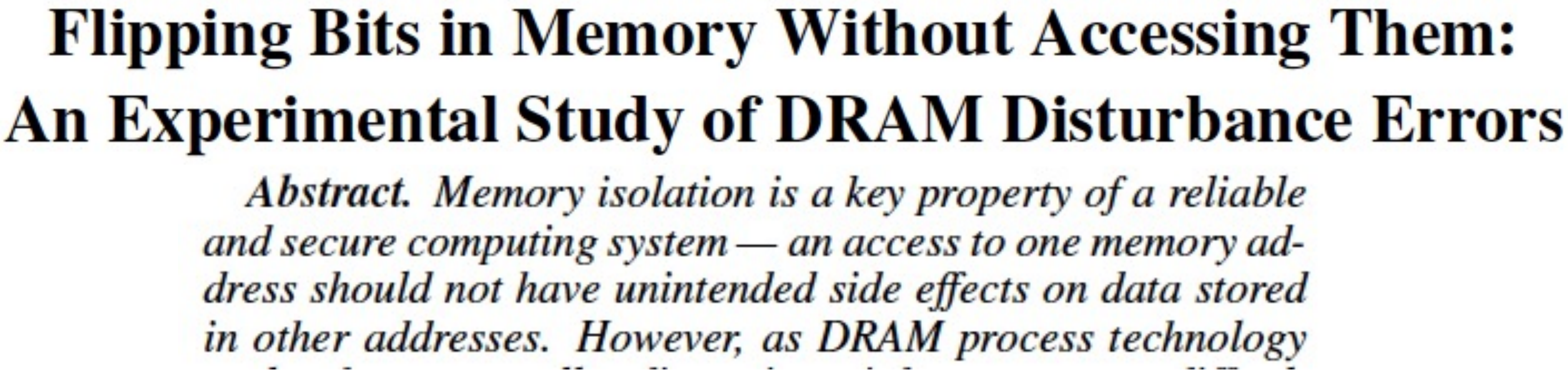
Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors, (Kim et al., ISCA 2014)

Recent DRAM Is More Vulnerable



All modules from 2012–2013 are vulnerable

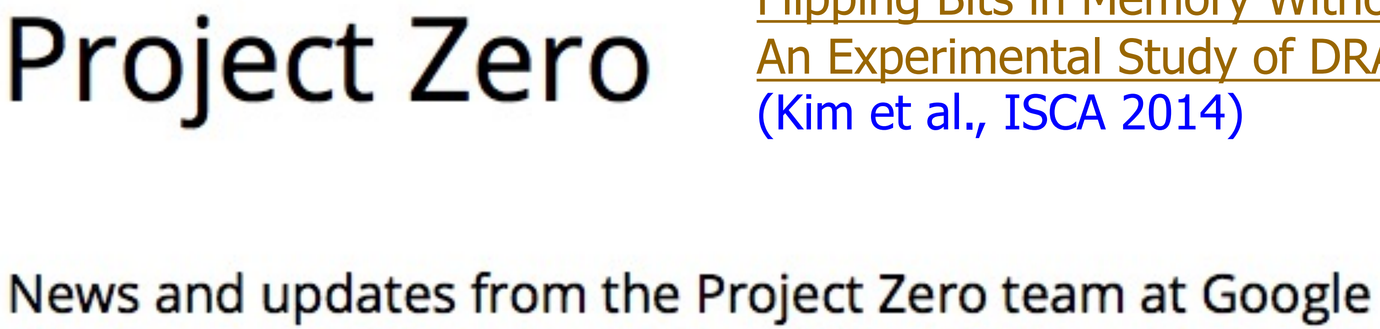
One Can Take Over an Otherwise-Secure System



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

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

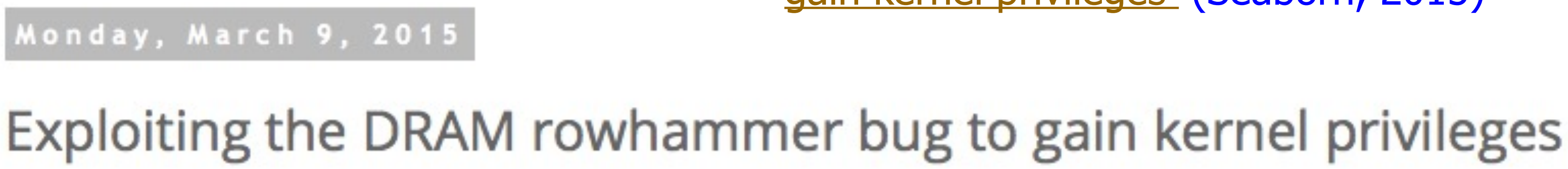
Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors (Kim et al., ISCA 2014)



Project Zero

News and updates from the Project Zero team at Google

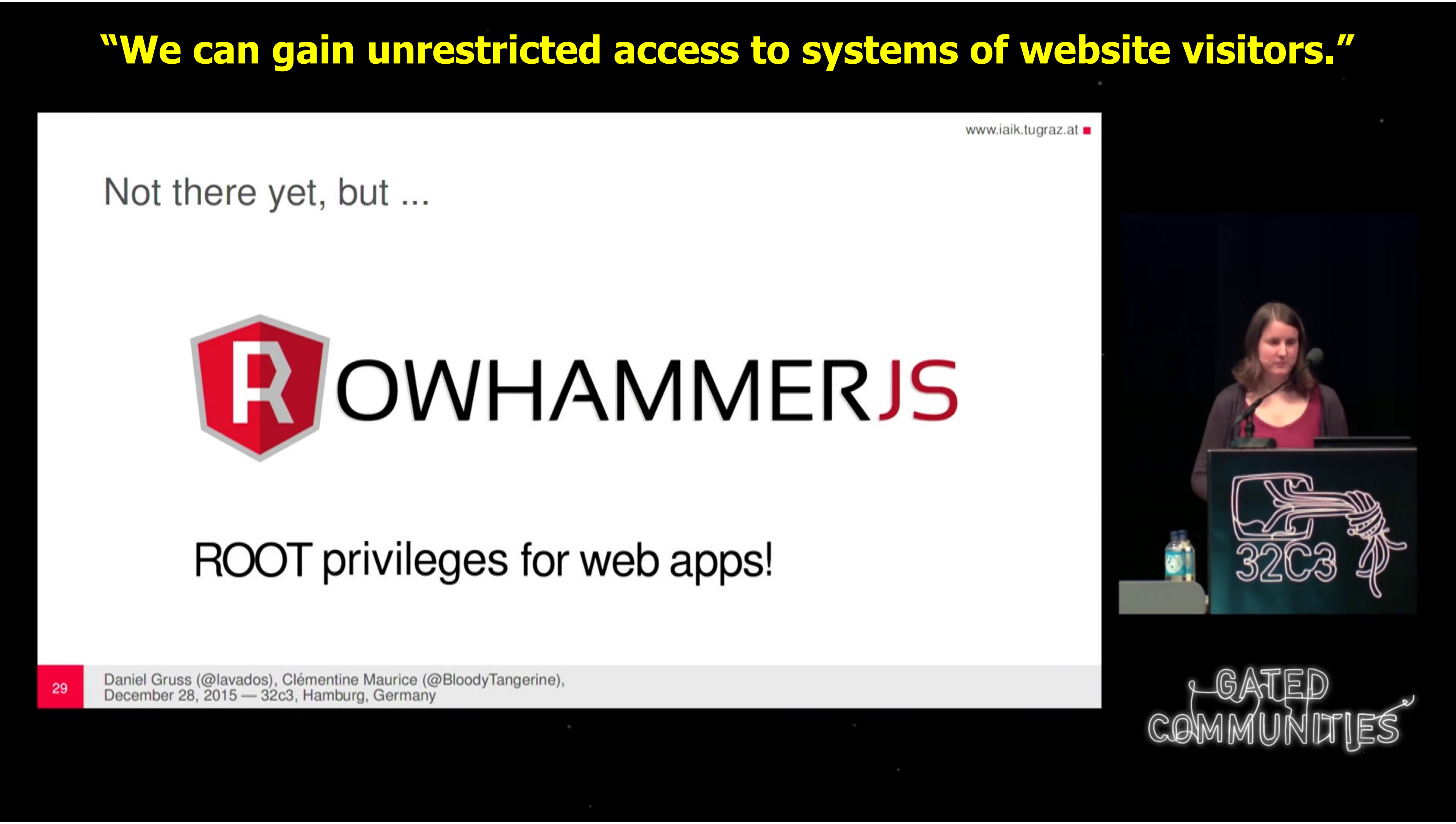
Exploiting the DRAM rowhammer bug to gain kernel privileges (Seaborn, 2015)



Monday, March 9, 2015

Exploiting the DRAM rowhammer bug to gain kernel privileges

More Security Implications (I)



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

www.iaik.tugraz.at

Not there yet, but ...

ROWHAMMERJS

ROOT privileges for web apps!

29

Daniel Gruss (@lavados), Clémentine Maurice (@BloodyTangerine),
December 28, 2015 — 32c3, Hamburg, Germany

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

Source: https://lab.dsst.io/32c3-slides/7197.html

More Security Implications (II)

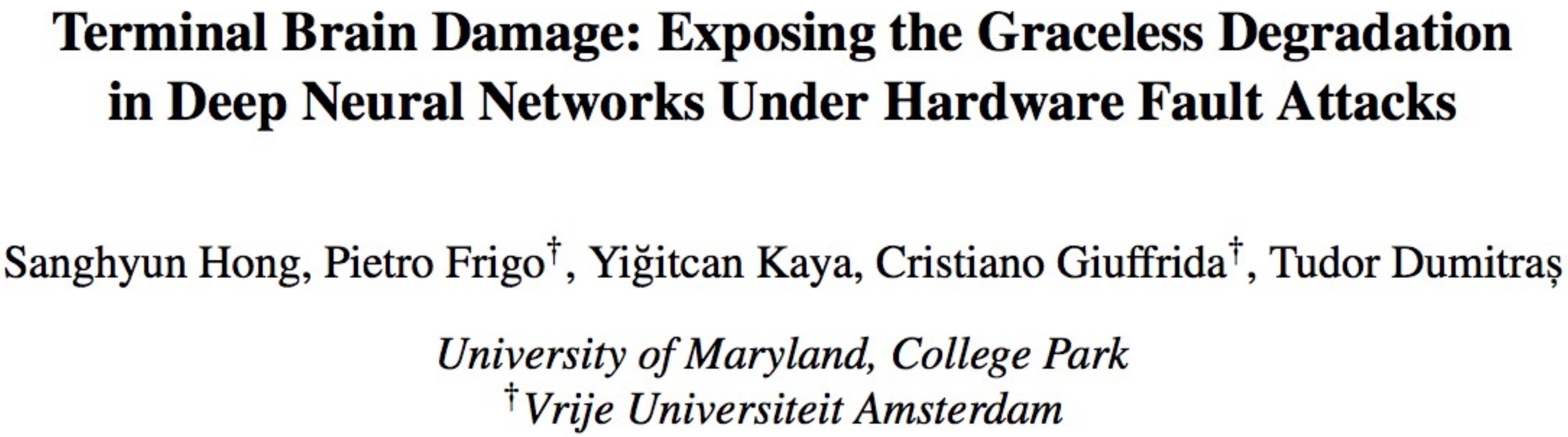
"Can gain control of a smart phone deterministically"

Drammer: Deterministic Rowhammer Attacks on Mobile Platforms, CCS'16

Source: https://fossbytes.com/drammer-rowhammer-attack-android-root-devices/

More Security Implications (VII)

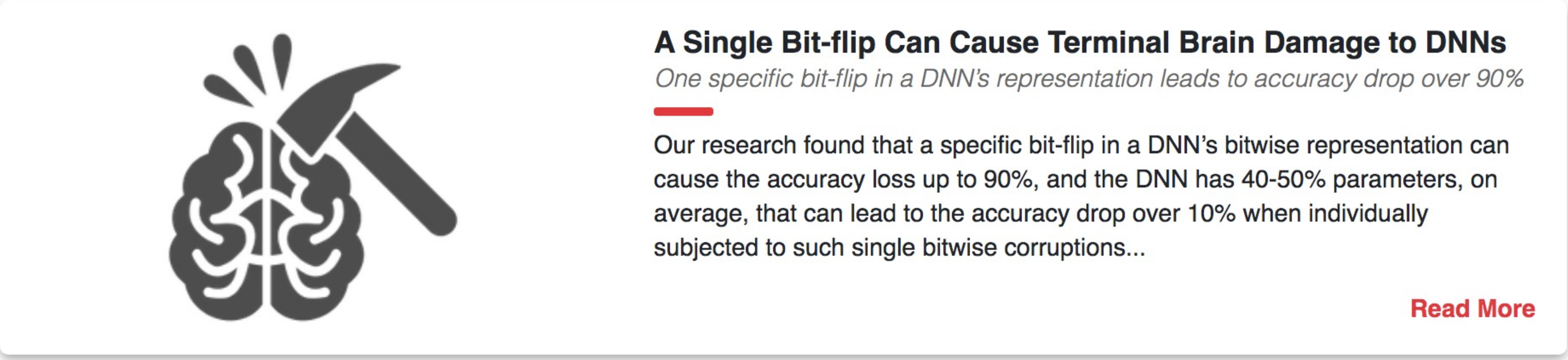
- USENIX Security 2019



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

Sanghyun Hong, Pietro Frigo[†], Yiğitcan Kaya, Cristiano Giuffrida[†], Tudor Dumitraș

University of Maryland, College Park
[†]Vrije Universiteit Amsterdam



A Single Bit-flip Can Cause Terminal Brain Damage to DNNs

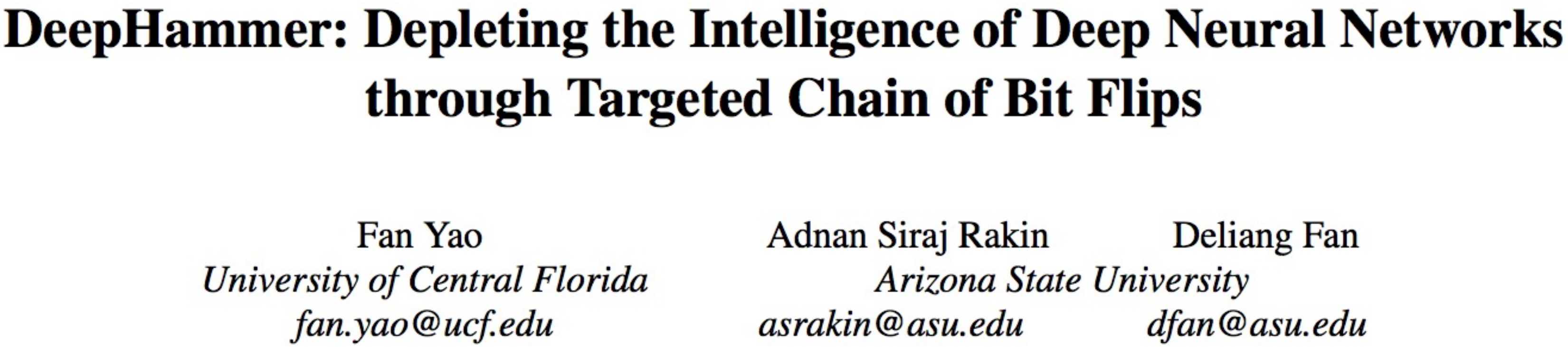
One specific bit-flip in a DNN's representation leads to accuracy drop over 90%

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

Read More

More Security Implications (VIII)

- USENIX Security 2020



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

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

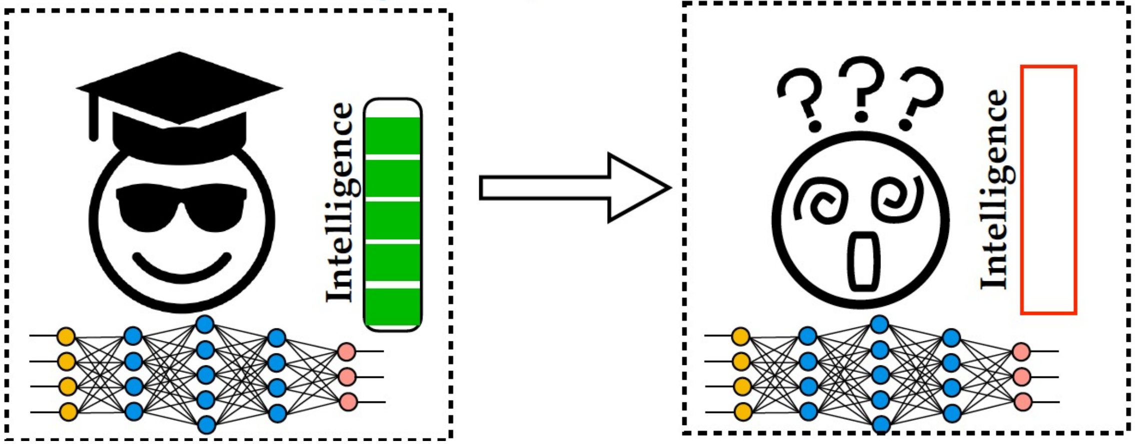
Adnan Siraj Rakin
Arizona State University
asrakin@asu.edu

Deliang Fan
Arizona State University
dfan@asu.edu

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

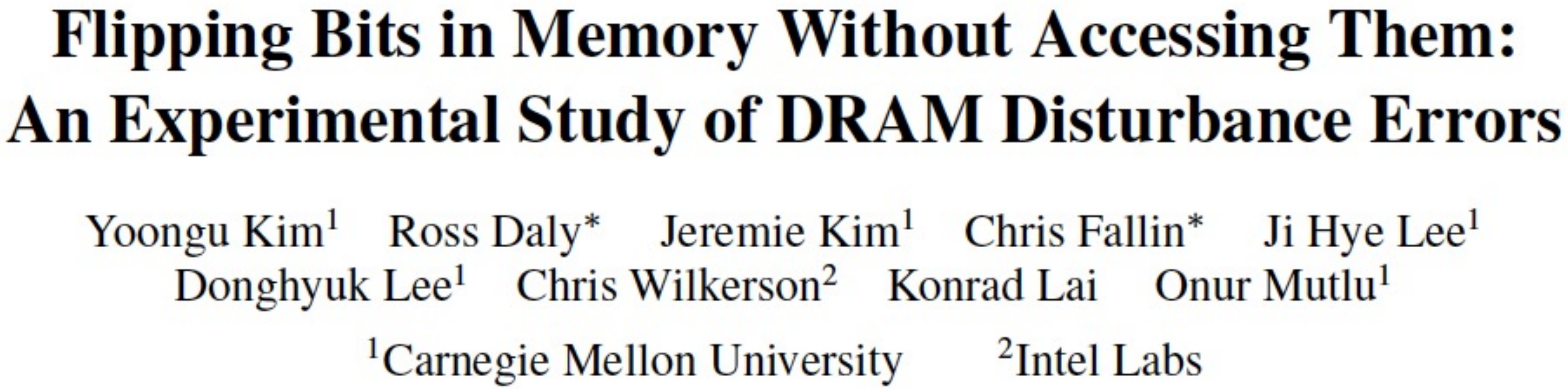
Example: ResNet-20 for CIFAR-10, **10** output classes

Before attack, **Accuracy: 90.2%** After attack, **Accuracy: ~10% (1/10)**



Memory Scaling Issues **Are** Real

- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu,
"Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors"
Proceedings of the 41st International Symposium on Computer Architecture (***ISCA***), Minneapolis, MN, June 2014.
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Source Code and Data]



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

Yoongu Kim[1] Ross Daly* Jeremie Kim[1] Chris Fallin* Ji Hye Lee[1]
Donghyuk Lee[1] Chris Wilkerson[2] Konrad Lai Onur Mutlu[1]

[1]Carnegie Mellon University [2]Intel Labs

Memory Scaling Issues **Are** Real

- Onur Mutlu and Jeremie Kim,
  "RowHammer: A Retrospective"
  IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (***TCAD***) *Special Issue on Top Picks in Hardware and Embedded Security*, 2019.
  [Preliminary arXiv version]
  [Slides from COSADE 2019 (pptx)]
  [Slides from VLSI-SOC 2020 (pptx) (pdf)]
  [Talk Video (1 hr 15 minutes, with Q&A)]

RowHammer: A Retrospective

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

The Push from Circuits and Devices

Main Memory Needs

Intelligent Controllers

RowHammer in 2020 (I)

- Jeremie S. Kim, Minesh Patel, A. Giray Yaglikci, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu,
"Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques"
Proceedings of the 47th International Symposium on Computer Architecture (***ISCA***), Valencia, Spain, June 2020.
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (20 minutes)]
[Lightning Talk Video (3 minutes)]

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

Jeremie S. Kim[§†] Minesh Patel[§] A. Giray Yağlıkçı[§]
Hasan Hassan[§] Roknoddin Azizi[§] Lois Orosa[§] Onur Mutlu[§†]

[§]*ETH Zürich* [†]*Carnegie Mellon University*

Key Takeaways from 1580 Chips

- **Newer DRAM chips are more vulnerable to RowHammer**

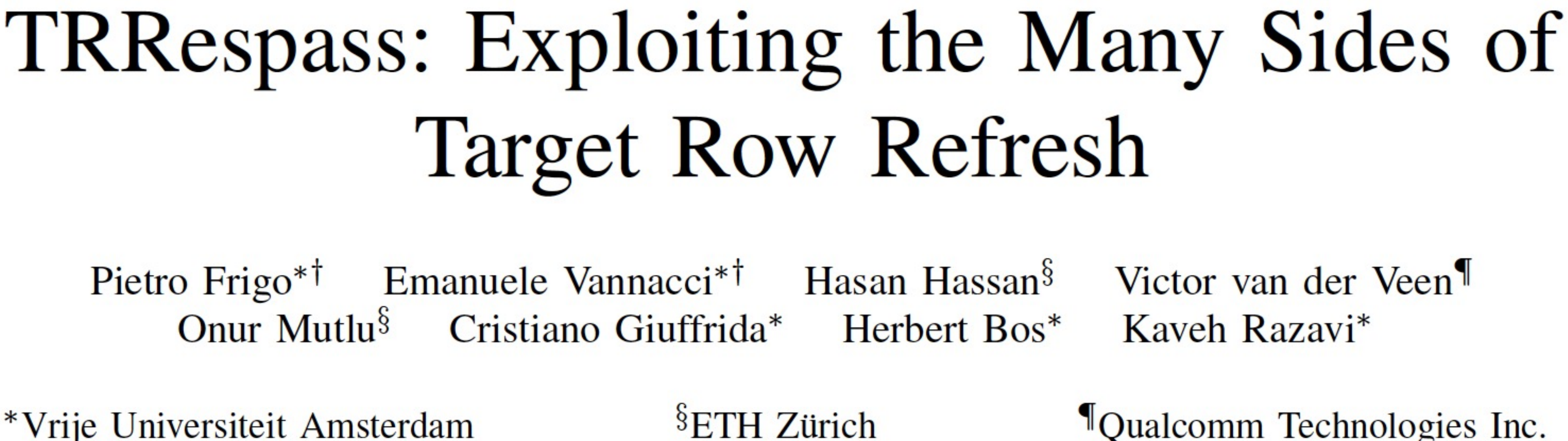
- There are chips today whose weakest cells fail after **only 4800 hammers**

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

- **Existing mitigation mechanisms are NOT effective**

RowHammer in 2020 (II)

- Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi,
"TRRespass: Exploiting the Many Sides of Target Row Refresh"
*Proceedings of the 41st IEEE Symposium on Security and Privacy (**S&P**)*, San Francisco, CA, USA, May 2020.
[Slides (pptx) (pdf)]
[Lecture Slides (pptx) (pdf)]
[Talk Video (17 minutes)]
[Lecture Video (59 minutes)]
[Source Code]
[Web Article]
Best paper award.
Pwnie Award 2020 for Most Innovative Research. Pwnie Awards 2020



TRRespass: Exploiting the Many Sides of Target Row Refresh

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

[*]Vrije Universiteit Amsterdam [§]ETH Zürich [¶]Qualcomm Technologies Inc.

RowHammer in 2020 (III)

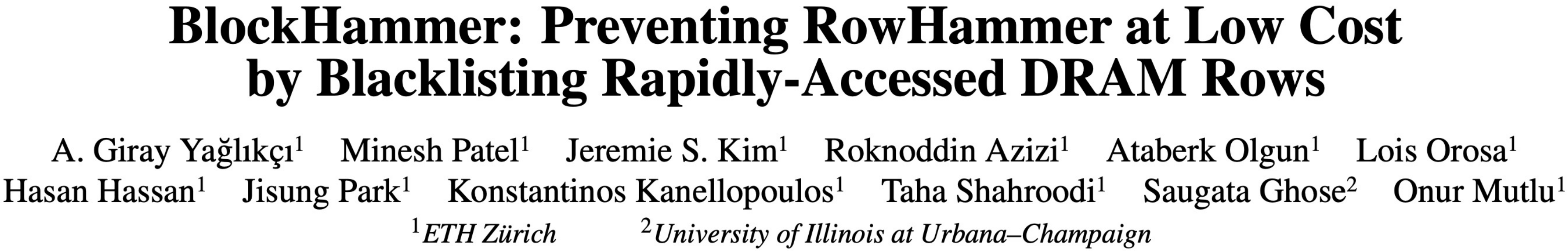
- Lucian Cojocar, Jeremie Kim, Minesh Patel, Lillian Tsai, Stefan Saroiu, Alec Wolman, and Onur Mutlu, **"Are We Susceptible to Rowhammer? An End-to-End Methodology for Cloud Providers"** *Proceedings of the 41st IEEE Symposium on Security and Privacy (**S&P**)*, San Francisco, CA, USA, May 2020.
  [Slides (pptx) (pdf)]
  [Talk Video (17 minutes)]

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

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

BlockHammer Solution in 2021

- A. Giray Yaglikci, Minesh Patel, Jeremie S. Kim, Roknoddin Azizi, Ataberk Olgun, Lois Orosa, Hasan Hassan, Jisung Park, Konstantinos Kanellopoulos, Taha Shahroodi, Saugata Ghose, and Onur Mutlu,
"BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows"
*Proceedings of the 27th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Virtual, February-March 2021.
[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Talk Video (22 minutes)]
[Short Talk Video (7 minutes)]



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

A. Giray Yağlıkçı[1] Minesh Patel[1] Jeremie S. Kim[1] Roknoddin Azizi[1] Ataberk Olgun[1] Lois Orosa[1]
Hasan Hassan[1] Jisung Park[1] Konstantinos Kanellopoulos[1] Taha Shahroodi[1] Saugata Ghose[2] Onur Mutlu[1]
[1]*ETH Zürich* [2]*University of Illinois at Urbana–Champaign*

Two Upcoming RowHammer Papers at MICRO 2021

- Lois Orosa, Abdullah Giray Yaglikci, Haocong Luo, Ataberk Olgun, Jisung Park, Hasan Hassan, Minesh Patel, Jeremie S. Kim, Onur Mutlu,
  "A Deeper Look into RowHammer's Sensitivities: Experimental Analysis of Real DRAM Chips and Implications on Future Attacks and Defenses"
  MICRO 2021

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

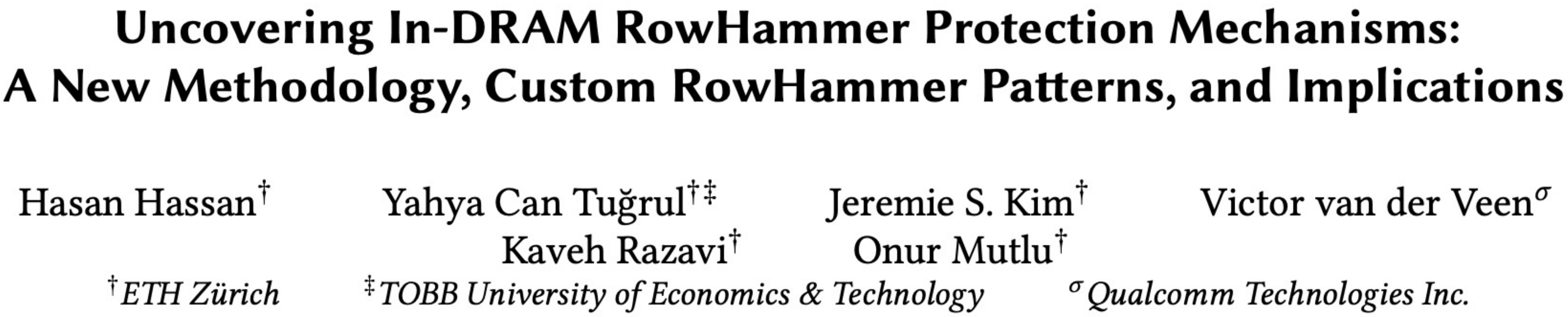
Lois Orosa* — ETH Zürich
A. Giray Yağlıkçı* — ETH Zürich
Haocong Luo — ETH Zürich
Ataberk Olgun — ETH Zürich, TOBB ETÜ
Jisung Park — ETH Zürich
Hasan Hassan — ETH Zürich
Minesh Patel — ETH Zürich
Jeremie S. Kim — ETH Zürich
Onur Mutlu — ETH Zürich

Two Upcoming RowHammer Papers at MICRO 2021

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

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

  MICRO 2021



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

Hasan Hassan[†] Yahya Can Tuğrul[†‡] Jeremie S. Kim[†] Victor van der Veen[σ]
Kaveh Razavi[†] Onur Mutlu[†]
[†] *ETH Zürich* [‡] *TOBB University of Economics & Technology* [σ] *Qualcomm Technologies Inc.*

Key Takeaways in 2021

RowHammer is still
an open problem

Security by obscurity
is not a good solution

Detailed Lectures on RowHammer

- Computer Architecture, Fall 2020, Lecture 4b
  - RowHammer (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=KDy632z23UE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=8
- Computer Architecture, Fall 2020, Lecture 5a
  - RowHammer in 2020: TRRespass (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=pwRw7QqK_qA&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=9
- Computer Architecture, Fall 2020, Lecture 5b
  - RowHammer in 2020: Revisiting RowHammer (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=gR7XR-Eepcg&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=10
- Computer Architecture, Fall 2020, Lecture 5c
  - Secure and Reliable Memory (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=HvswnsfG3oQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=11

The Story of RowHammer Lecture …

- Onur Mutlu,
  "The Story of RowHammer"
  Keynote Talk at *Secure Hardware, Architectures, and Operating Systems Workshop (**SeHAS**), held with HiPEAC 2021 Conference*, Virtual, 19 January 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hr 15 minutes, with Q&A)]

The Story of RowHammer

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
19 January 2021
SEHAS Keynote @ HiPEAC

The Story of Rowhammer - Secure Hardware, Architectures, and Operating Systems Keynote - Onur Mutlu

1,293 views • Premiered Feb 2, 2021

https://www.youtube.com/watch?v=sgd7PHQQ1AI

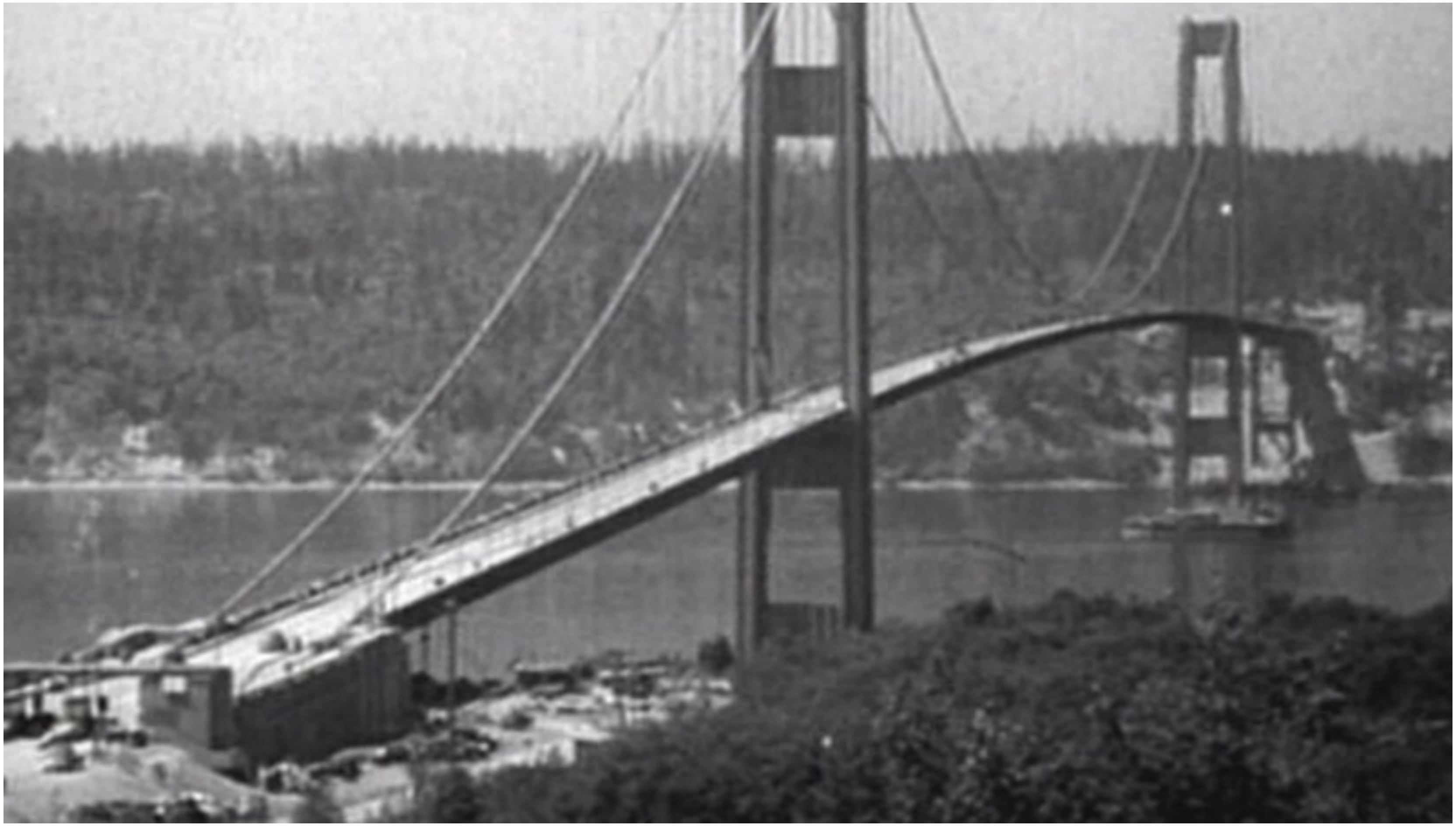
Rowhammer

The Push from Circuits and Devices

Main Memory Needs

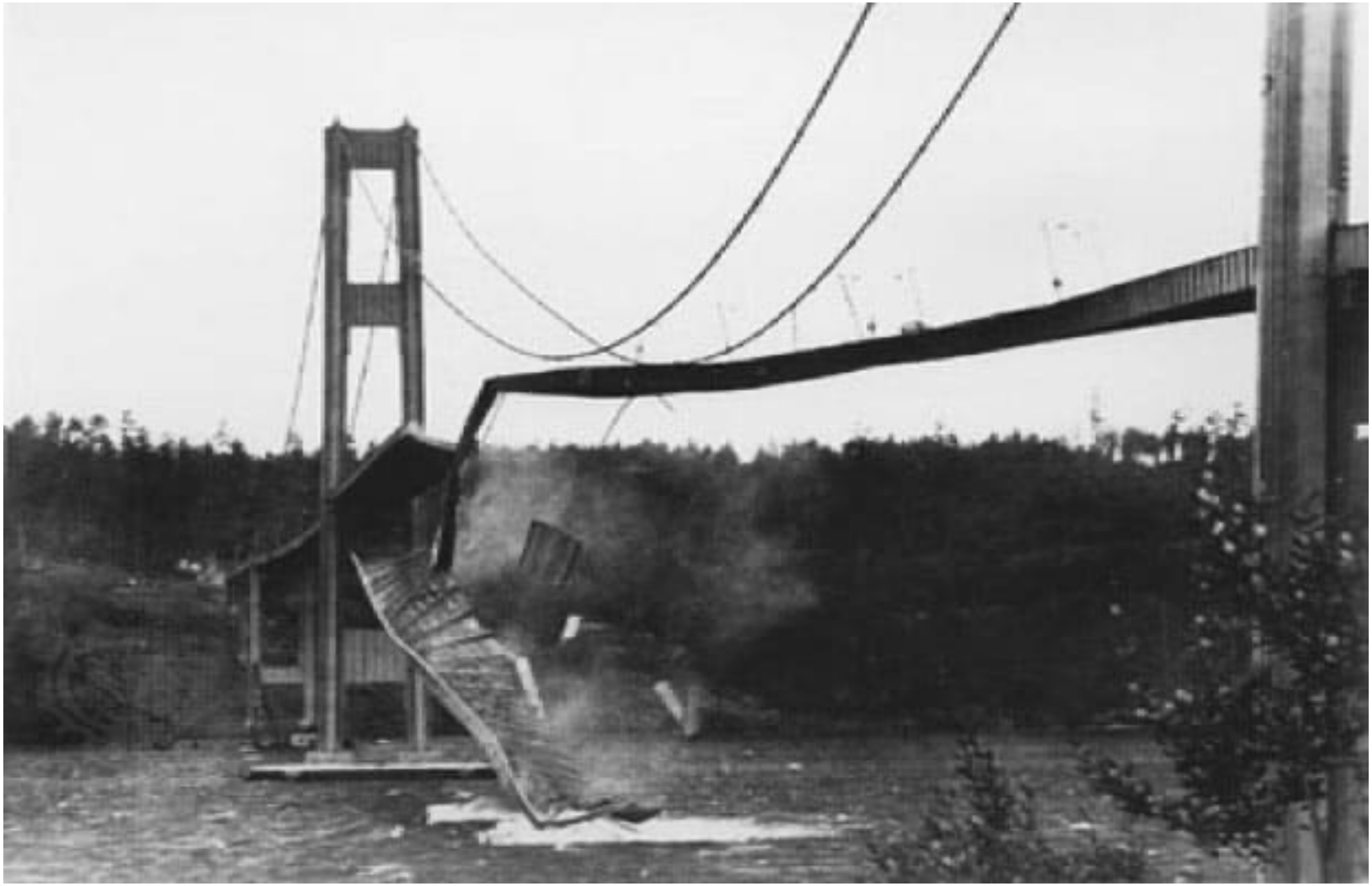
Intelligent Controllers

How Reliable/Secure/Safe is This Bridge?



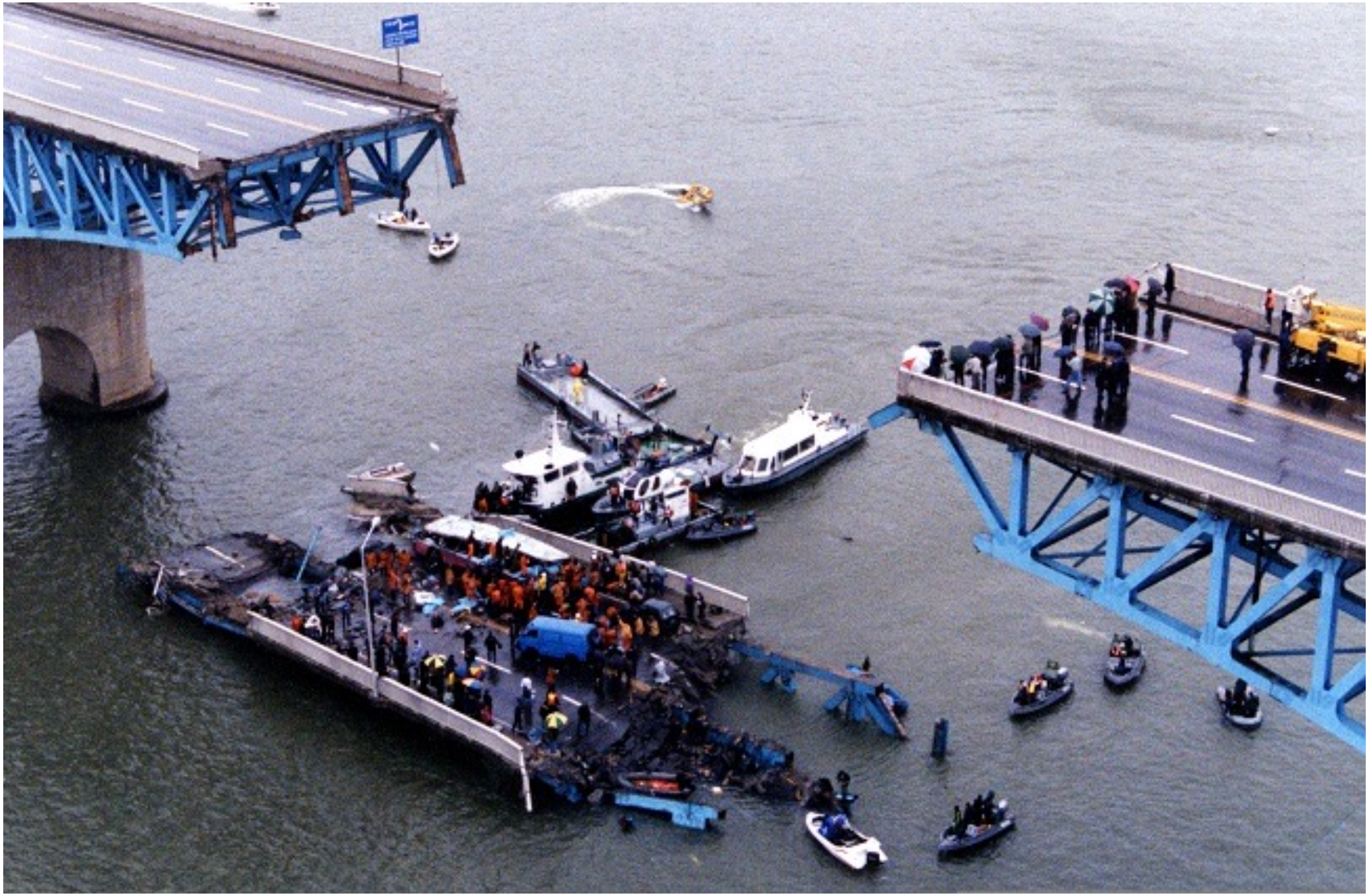
Source: http://www.technologystudent.com/struct1/tacom1.png

Collapse of the "Galloping Gertie" (1940)



Source: AP
http://www.wsdot.wa.gov/tnbhistory/connections/connections3.htm

Another Example (1994)

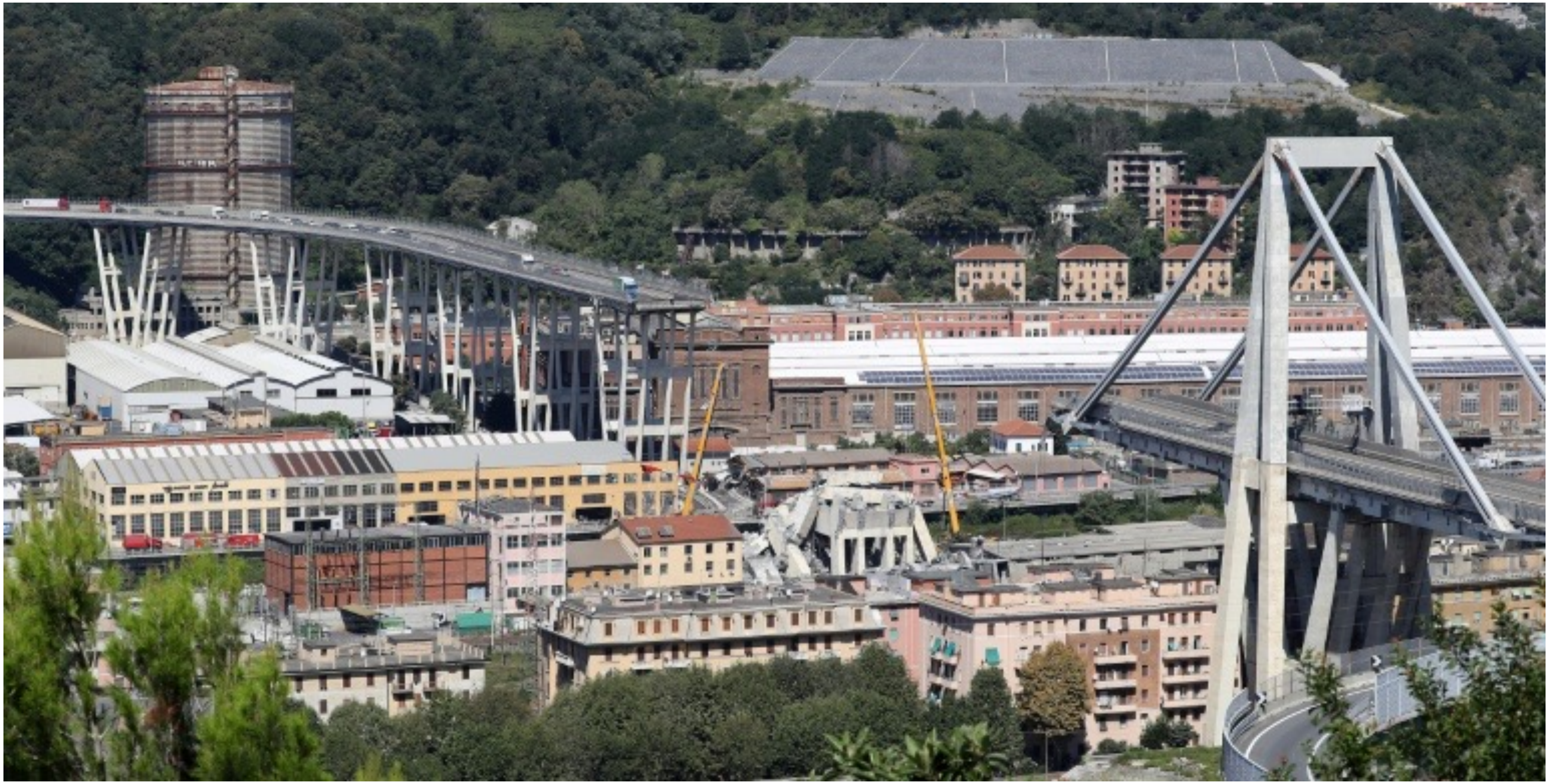


Source: By 최광모 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=35197984

Yet Another Example (2007)

Source: Morry Gash/AP,
https://www.npr.org/2017/08/01/540669701/10-years-after-bridge-collapse-america-is-still-crumbling?t=1535427165809

A More Recent Example (2018)



Source: AFP / Valery HACHE, https://www.capitalfm.co.ke/news/2018/08/genoa-bridge-collapse-what-we-know/

How Safe & Secure Is This Platform?

Security is about preventing unforeseen consequences

Source: https://s-media-cache-ak0.pinimg.com/originals/48/09/54/4809543a9c7700246a0cf8acdae27abf.jpg

How Safe & Secure Is **This** Platform?

Source: https://taxistartup.com/wp-content/uploads/2015/03/UK-Self-Driving-Cars.jpg

Challenge and Opportunity for Future

Fundamentally

Secure, Reliable, Safe

Computing Architectures

Solution Direction: Principled Designs

Design fundamentally secure computing architectures

Predict and prevent safety & security issues

The Push from Circuits and Devices

Computing Systems Need

Intelligent Memories

The Takeaway, Again

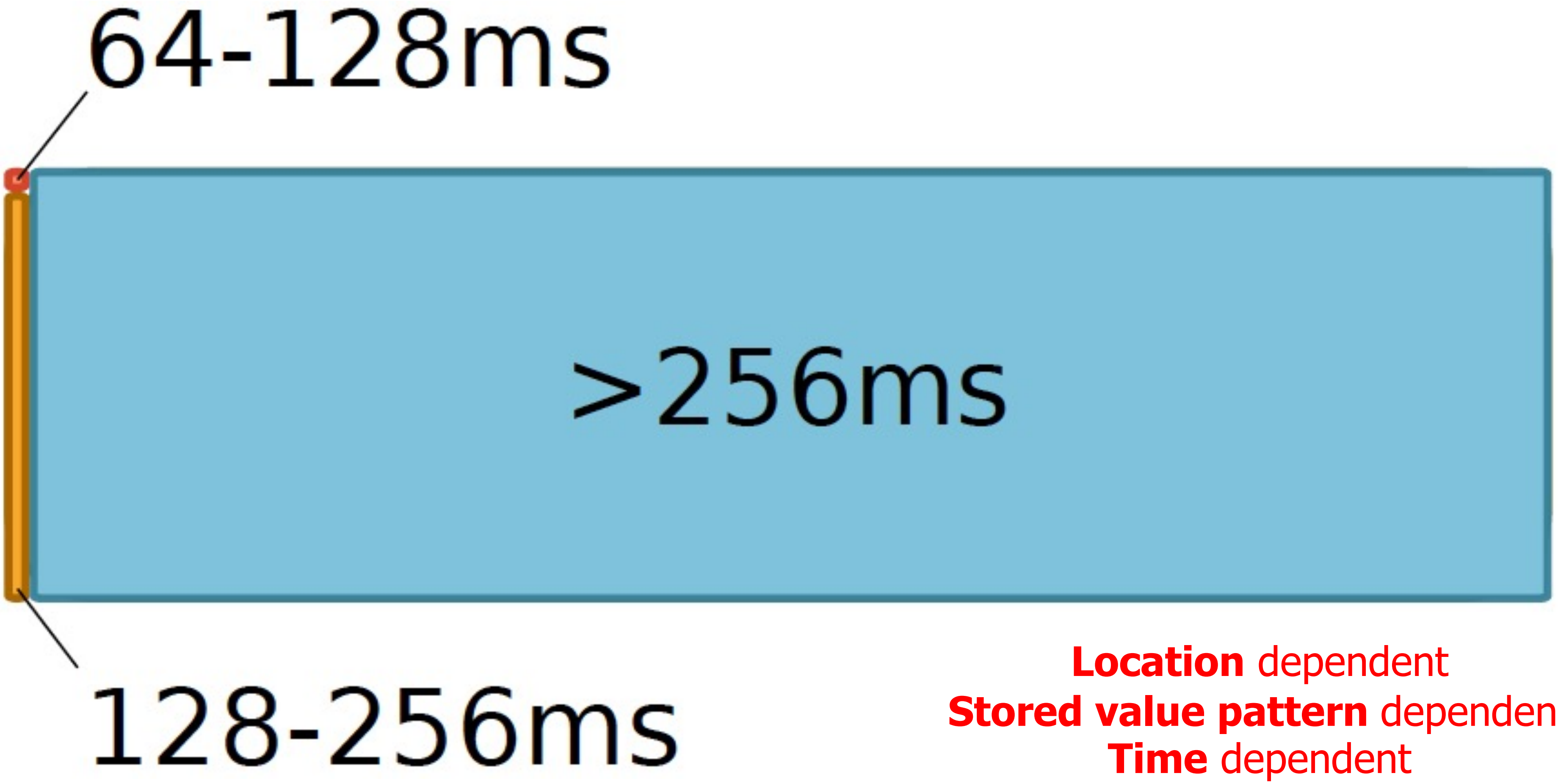
In-Field Patch-ability

(Intelligent Memory)

Can Avoid Many Failures

Data Retention in Memory **[Liu et al., ISCA 2013]**

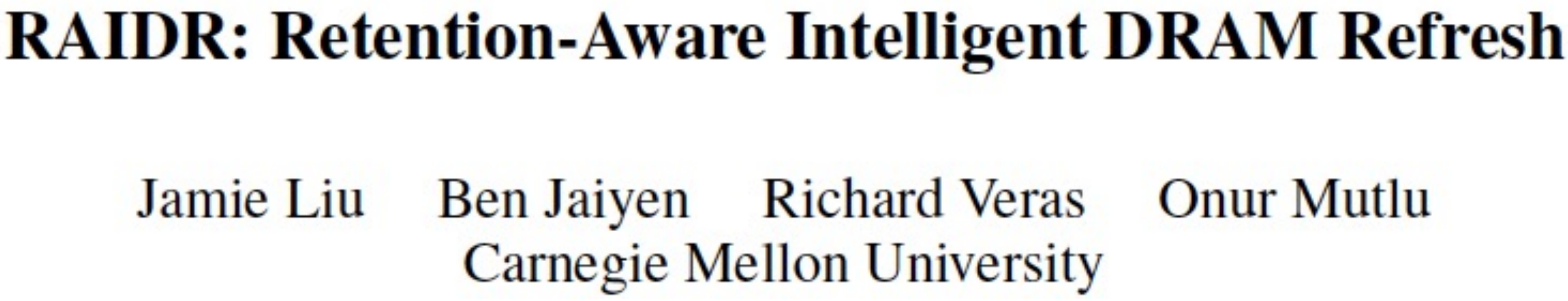
- Retention Time Profile of DRAM looks like this:



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"** *Proceedings of the 39th International Symposium on Computer Architecture* (***ISCA***), Portland, OR, June 2012. Slides (pdf)

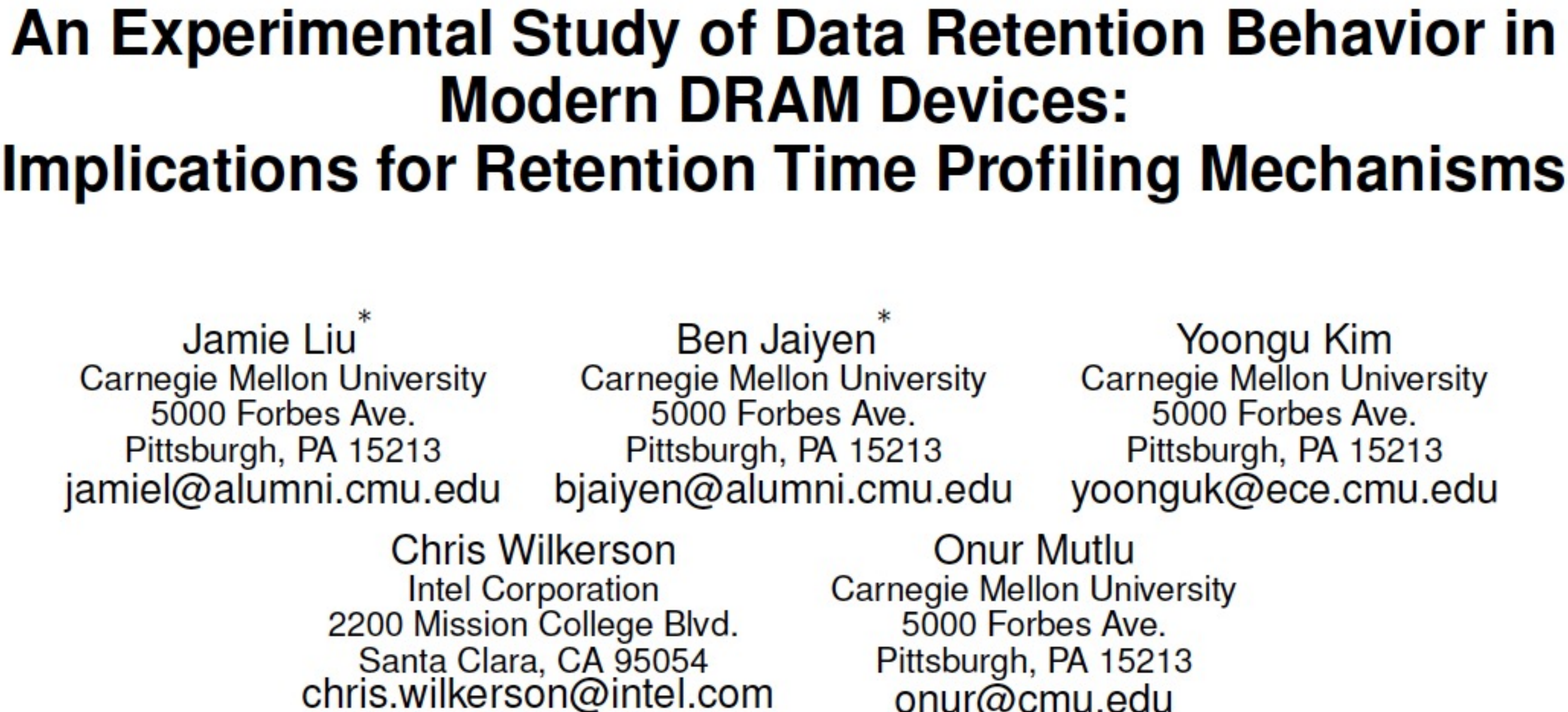


RAIDR: Retention-Aware Intelligent DRAM Refresh

Jamie Liu   Ben Jaiyen   Richard Veras   Onur Mutlu
Carnegie Mellon University

More on DRAM Refresh (II)

- Jamie Liu, Ben Jaiyen, Yoongu Kim, Chris Wilkerson, and Onur Mutlu, **"An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms"** *Proceedings of the 40th International Symposium on Computer Architecture (**ISCA**)*, Tel-Aviv, Israel, June 2013. Slides (ppt) Slides (pdf)



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"
*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, Austin, TX, June 2014. [Slides (pptx) (pdf)] [Poster (pptx) (pdf)] [Full data sets]

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

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
{moin, dhkim, pnair6}@ece.gatech.edu

[‡]Carnegie Mellon University
{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)]

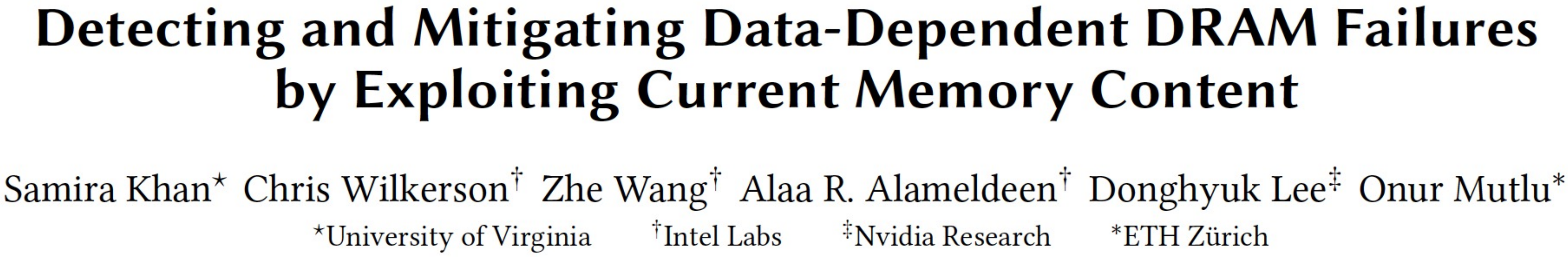
PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM

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)]



Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content

Samira Khan[⋆] Chris Wilkerson[†] Zhe Wang[†] Alaa R. Alameldeen[†] Donghyuk Lee[‡] Onur Mutlu[*]
[⋆]University of Virginia [†]Intel Labs [‡]Nvidia Research [*]ETH Zürich

More on DRAM Refresh (VII)

- Minesh Patel, Jeremie S. Kim, and Onur Mutlu,
  "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions"
  *Proceedings of the 44th International Symposium on Computer Architecture (**ISCA**)*, Toronto, Canada, June 2017.
  [Slides (pptx) (pdf)]
  [Lightning Session Slides (pptx) (pdf)]

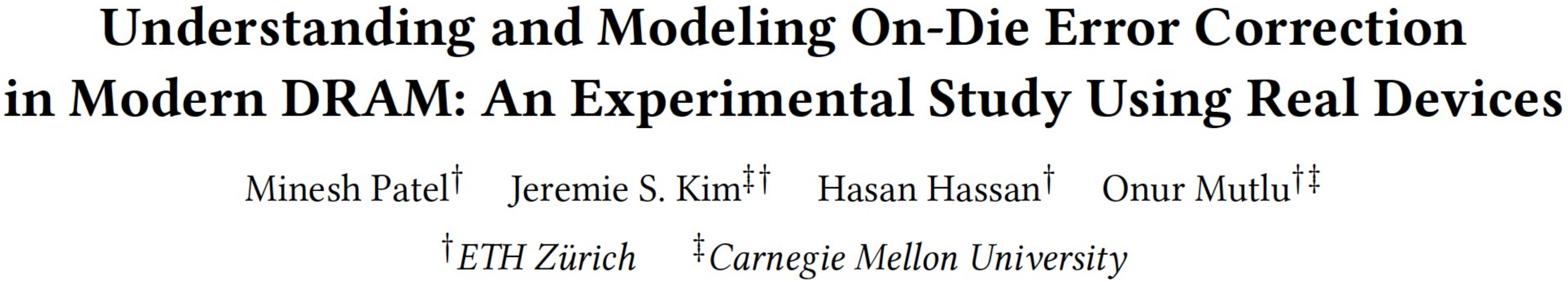
- First experimental analysis of (mobile) LPDDR4 chips
- Analyzes the complex tradeoff space of retention time profiling
- Idea: enable fast and robust profiling at higher refresh intervals & temperatures

The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions

Minesh Patel[§‡] Jeremie S. Kim[‡§] Onur Mutlu[§‡]
[§]ETH Zürich [‡]Carnegie Mellon University

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"** *Proceedings of the 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (**DSN**)*, Portland, OR, USA, June 2019.
[Slides (pptx) (pdf)]
[Talk Video (26 minutes)]
[Full Talk Lecture (29 minutes)]
[Source Code for EINSim, the Error Inference Simulator]
Best paper award.



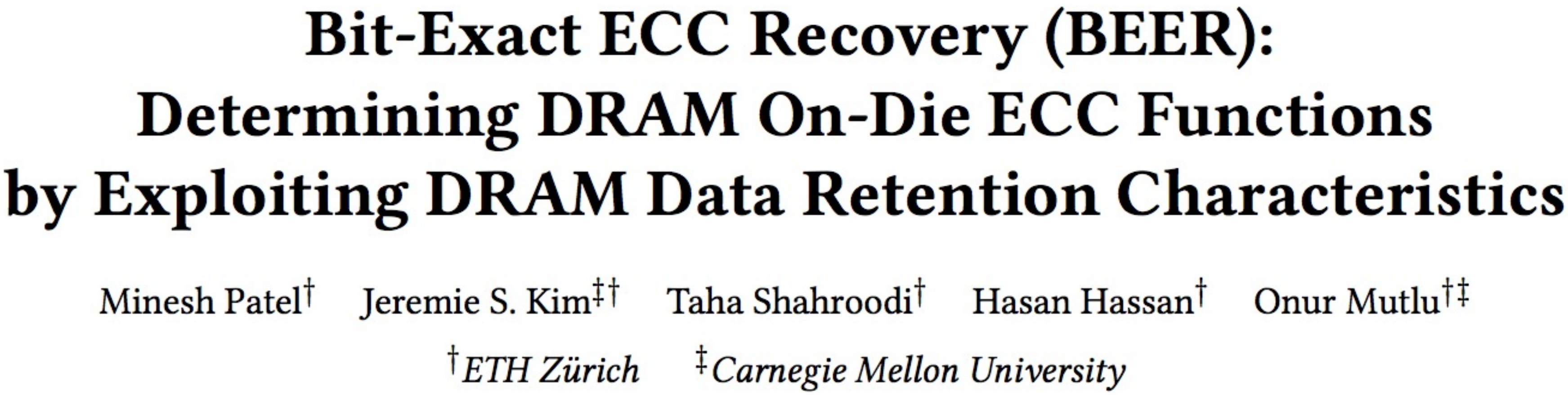
Understanding and Modeling On-Die Error Correction in Modern DRAM: An Experimental Study Using Real Devices

Minesh Patel[†] Jeremie S. Kim[‡†] Hasan Hassan[†] Onur Mutlu[†‡]

[†]*ETH Zürich* [‡]*Carnegie Mellon University*

More on DRAM Refresh (IX)

- Minesh Patel, Jeremie S. Kim, Taha Shahroodi, Hasan Hassan, and Onur Mutlu,
"Bit-Exact ECC Recovery (BEER): Determining DRAM On-Die ECC Functions by Exploiting DRAM Data Retention Characteristics"
Proceedings of the 53rd International Symposium on Microarchitecture (***MICRO***), Virtual, October 2020.
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (15 minutes)]
[Lightning Talk Video (1.5 minutes)]
Best paper award.



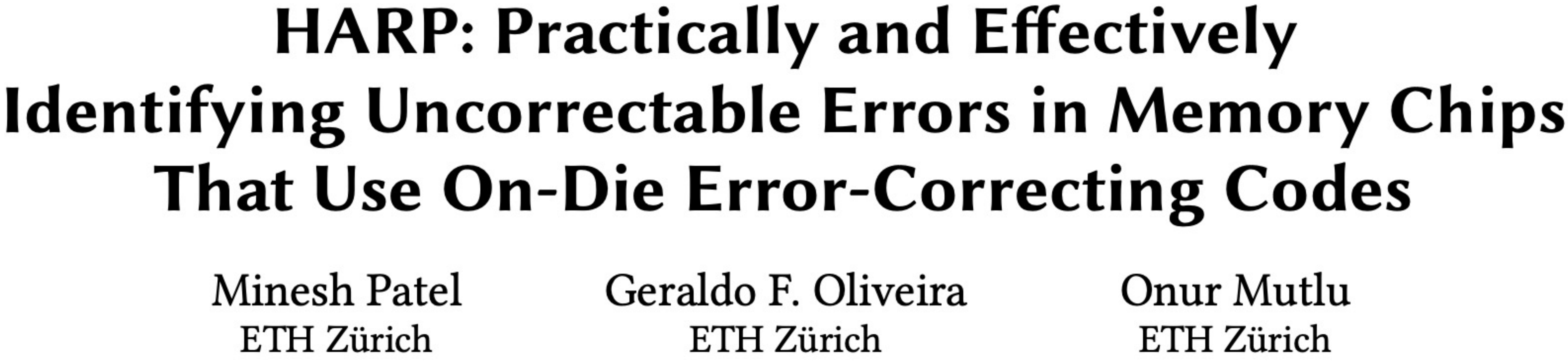
**Bit-Exact ECC Recovery (BEER):
Determining DRAM On-Die ECC Functions
by Exploiting DRAM Data Retention Characteristics**

Minesh Patel[†] Jeremie S. Kim[‡†] Taha Shahroodi[†] Hasan Hassan[†] Onur Mutlu[†‡]

[†]*ETH Zürich* [‡]*Carnegie Mellon University*

More on DRAM Refresh (X)

- To Appear in MICRO 2021



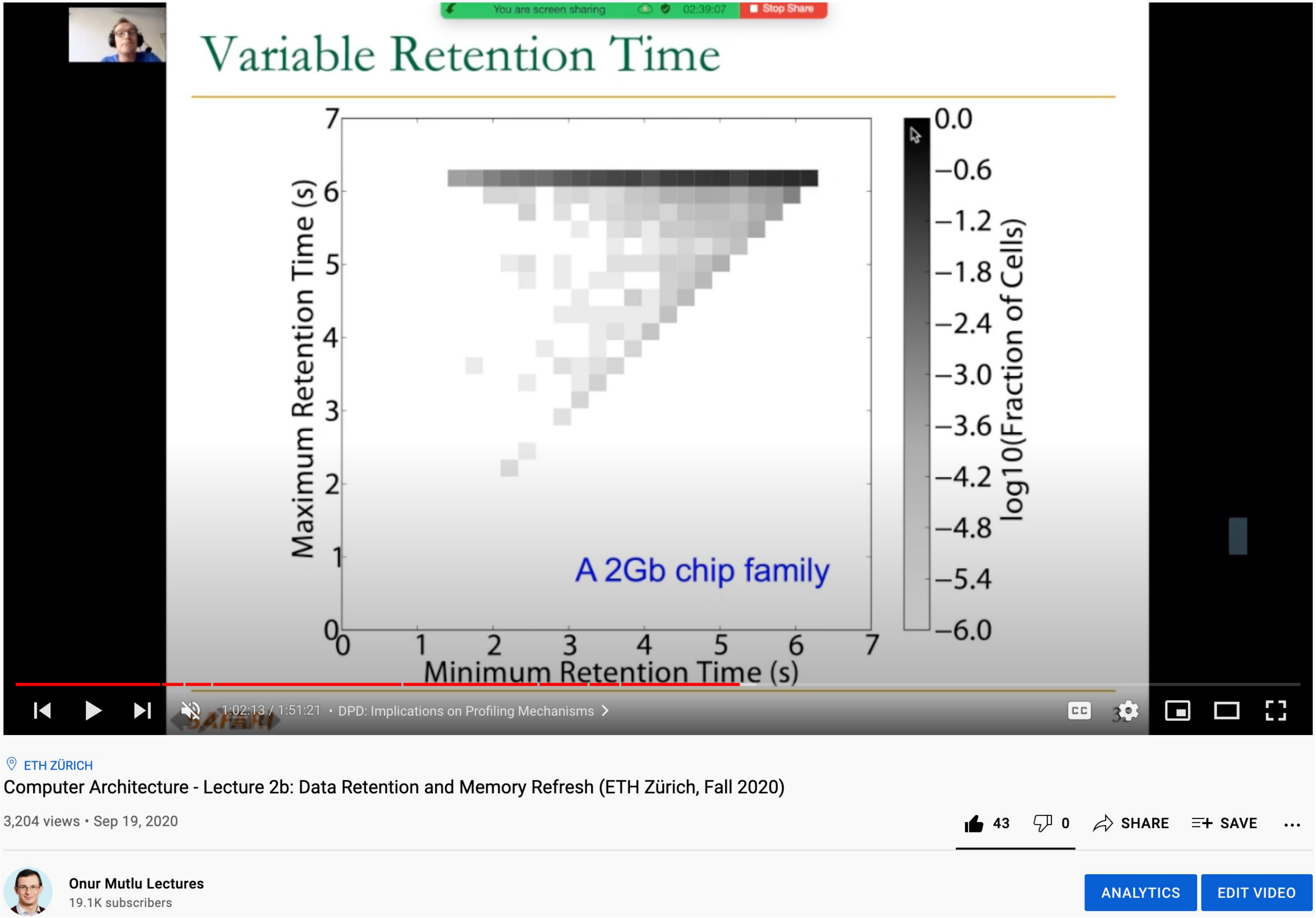
HARP: Practically and Effectively Identifying Uncorrectable Errors in Memory Chips That Use On-Die Error-Correcting Codes

Minesh Patel
ETH Zürich

Geraldo F. Oliveira
ETH Zürich

Onur Mutlu
ETH Zürich

More on DRAM Refresh & Data Retention



ETH ZÜRICH

Computer Architecture - Lecture 2b: Data Retention and Memory Refresh (ETH Zürich, Fall 2020)

3,204 views • Sep 19, 2020

Onur Mutlu Lectures

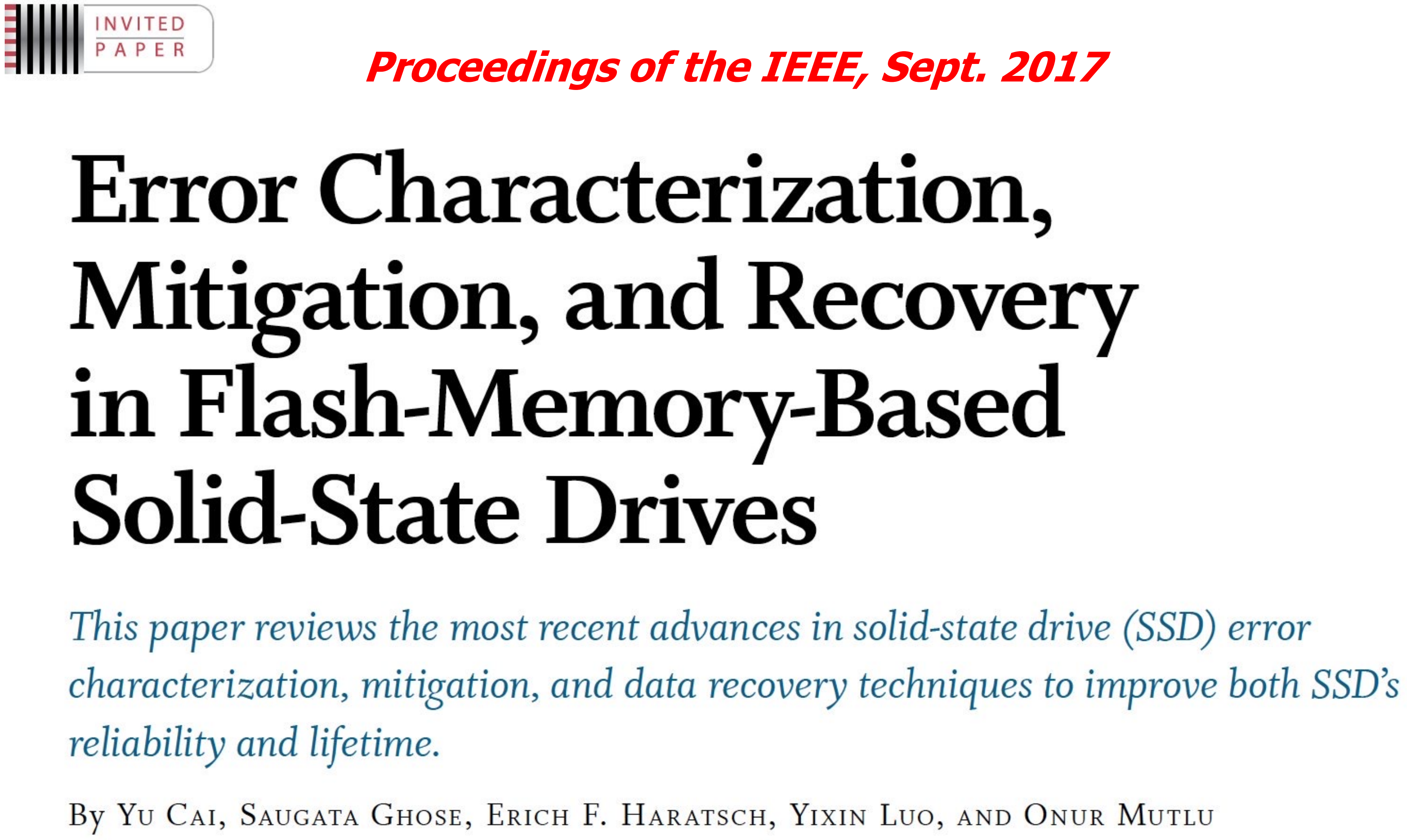
19.1K subscribers

https://www.youtube.com/watch?v=v702wUnaWGE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=3

The Push from Circuits and Devices

Main Memory Needs Intelligent Controllers

An Example Intelligent Controller



INVITED PAPER

Proceedings of the IEEE, Sept. 2017

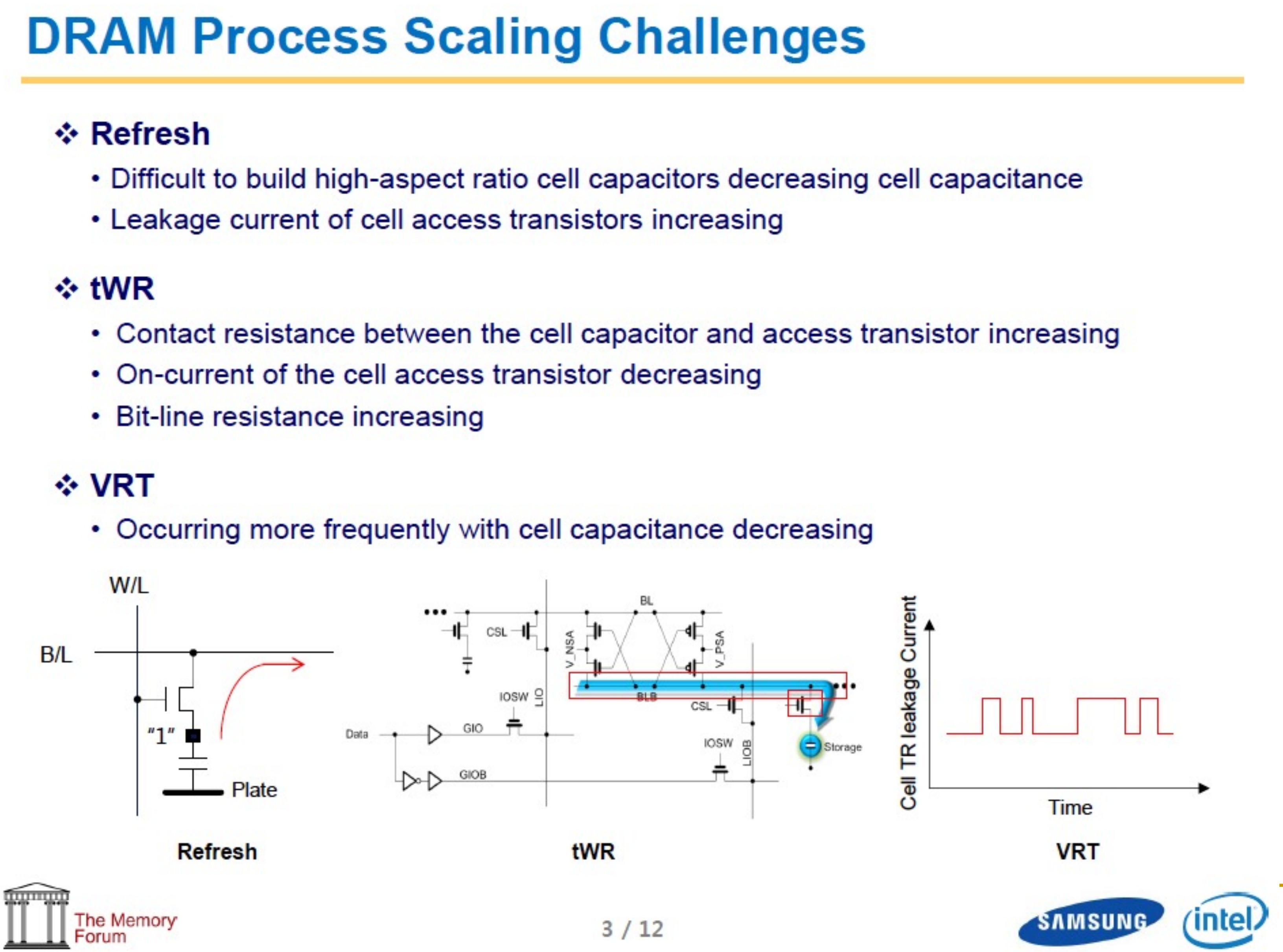
Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

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

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**

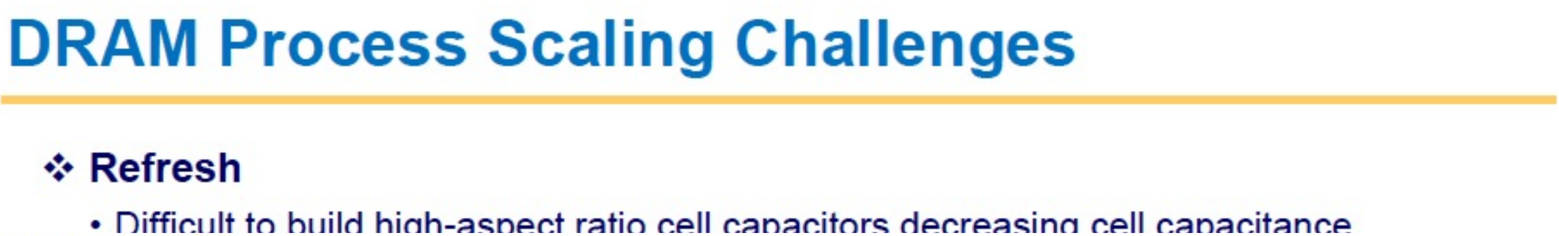
- Occurring more frequently with cell capacitance decreasing

Refresh **tWR** **VRT**

The Memory Forum

3 / 12

Call for Intelligent Memory Controllers

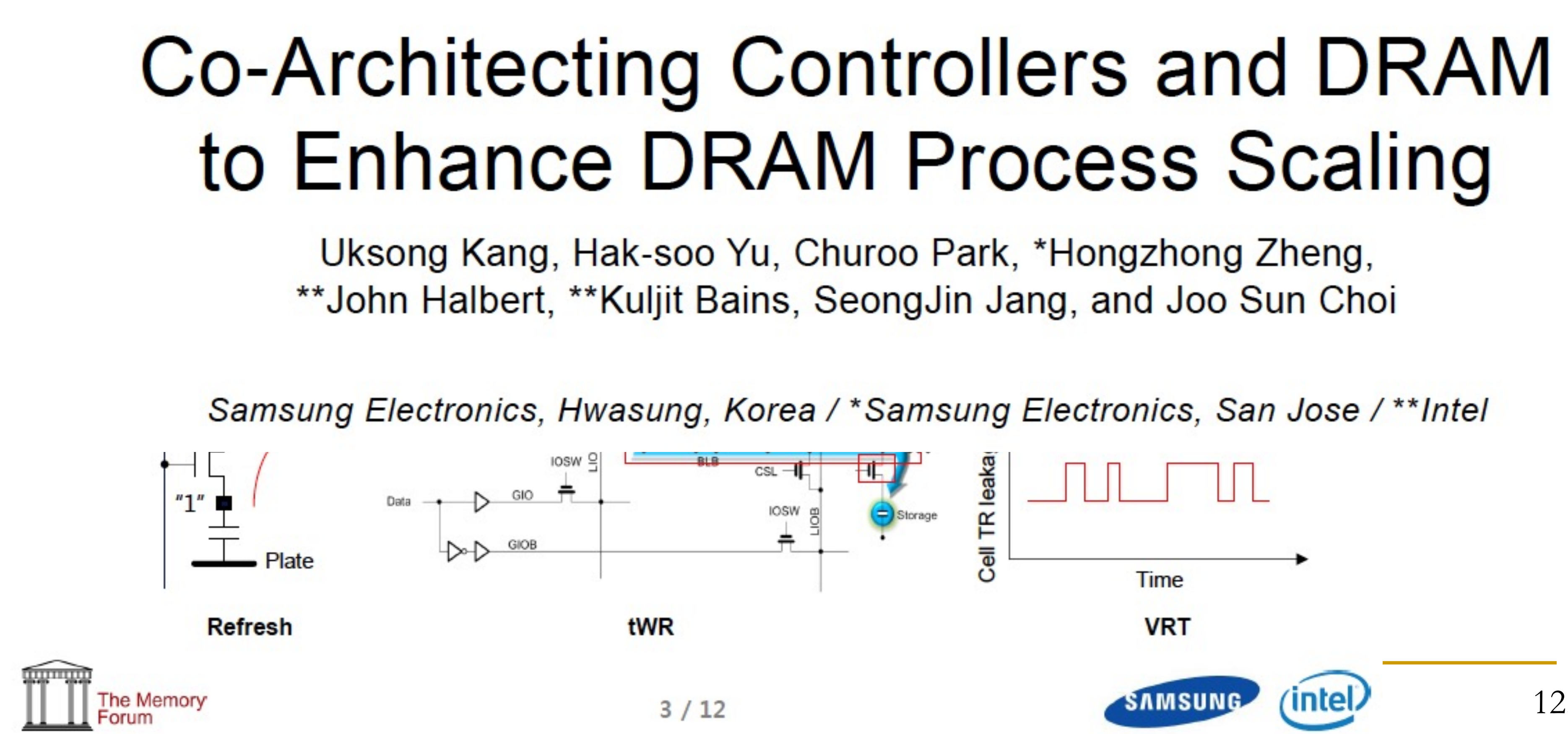


DRAM Process Scaling Challenges

❖ **Refresh**

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

THE MEMORY FORUM 2014

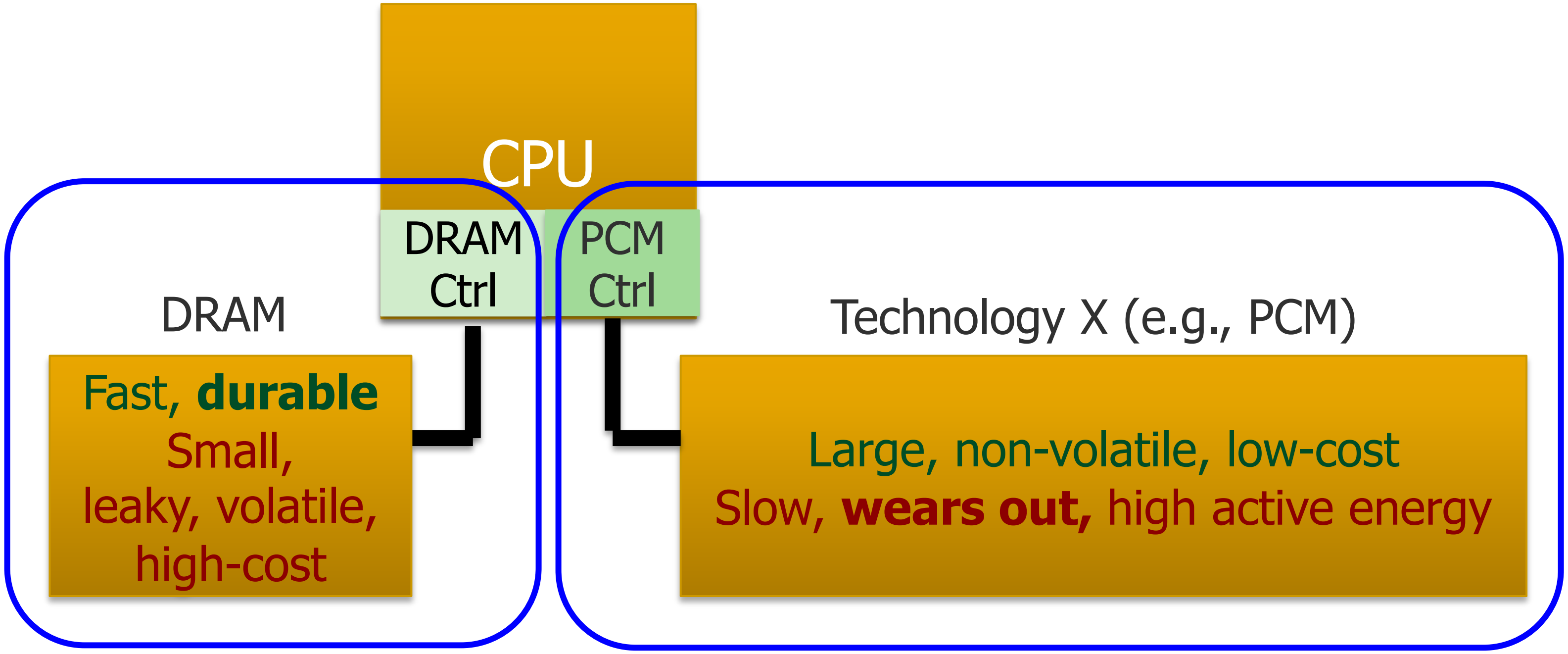


Co-Architecting Controllers and DRAM to Enhance DRAM Process Scaling

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*

Promising Direction: Hybrid Memory Systems



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

Meza+, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters, 2012.

Yoon, Meza et al., "Row Buffer Locality Aware Caching Policies for Hybrid Memories," ICCD 2012 Best Paper Award.

The Push from Circuits and Devices

Main Memory Needs

Intelligent Controllers

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

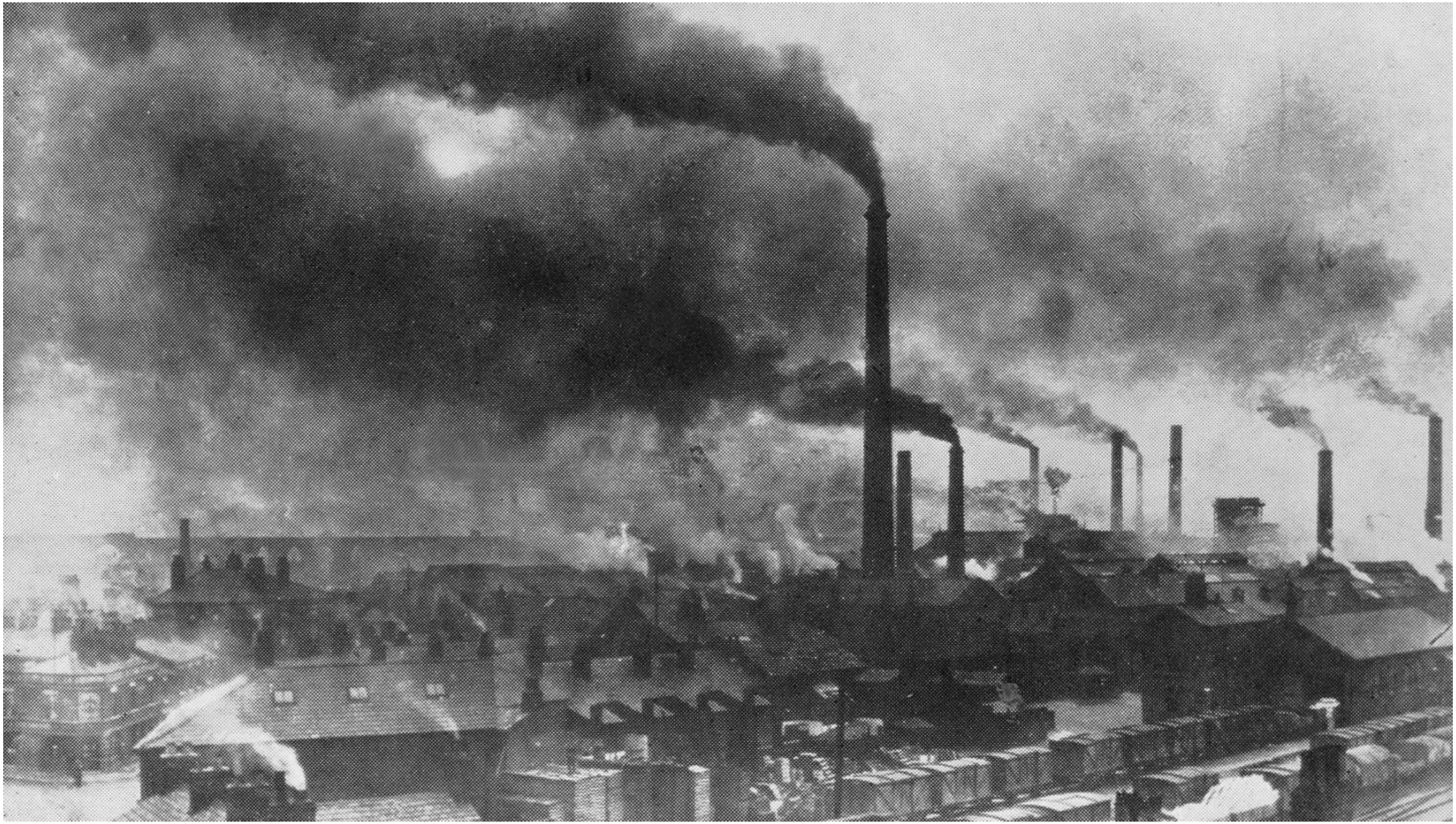
2. Energy consumption is a key limiter

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

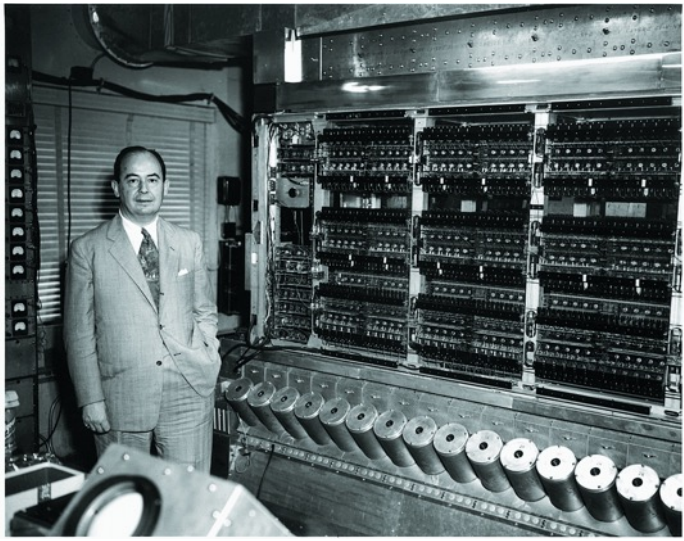
Processing of data

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.

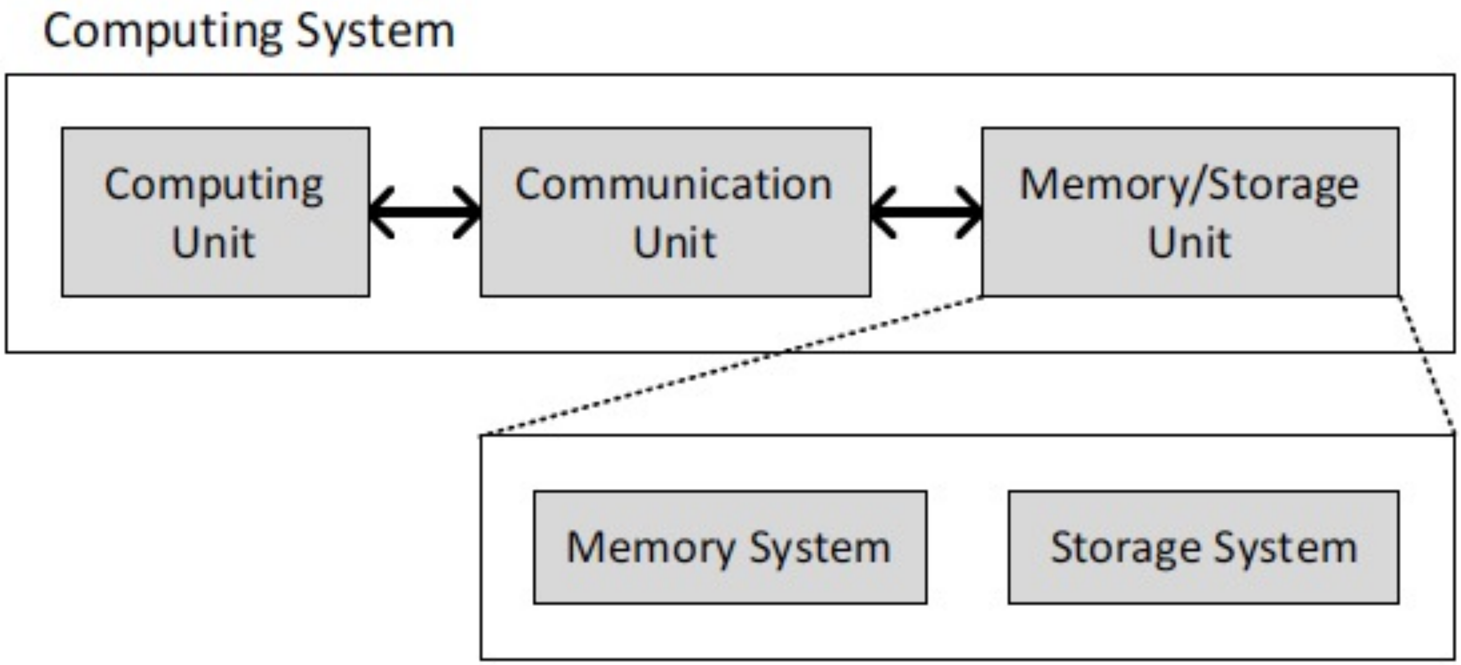
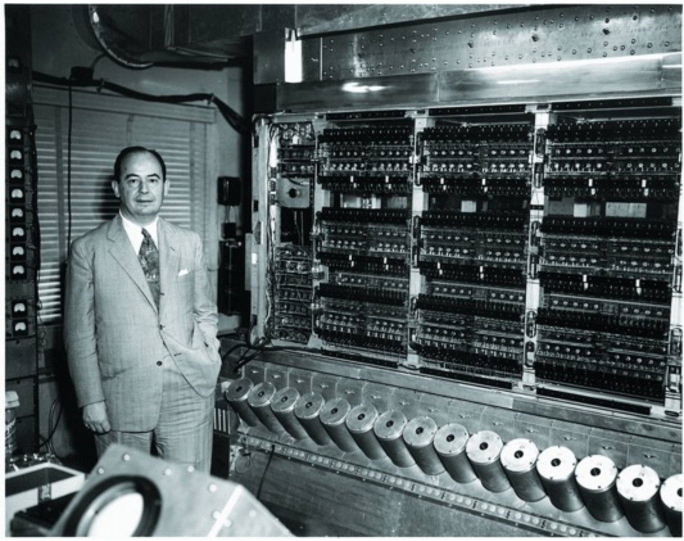


Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

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.

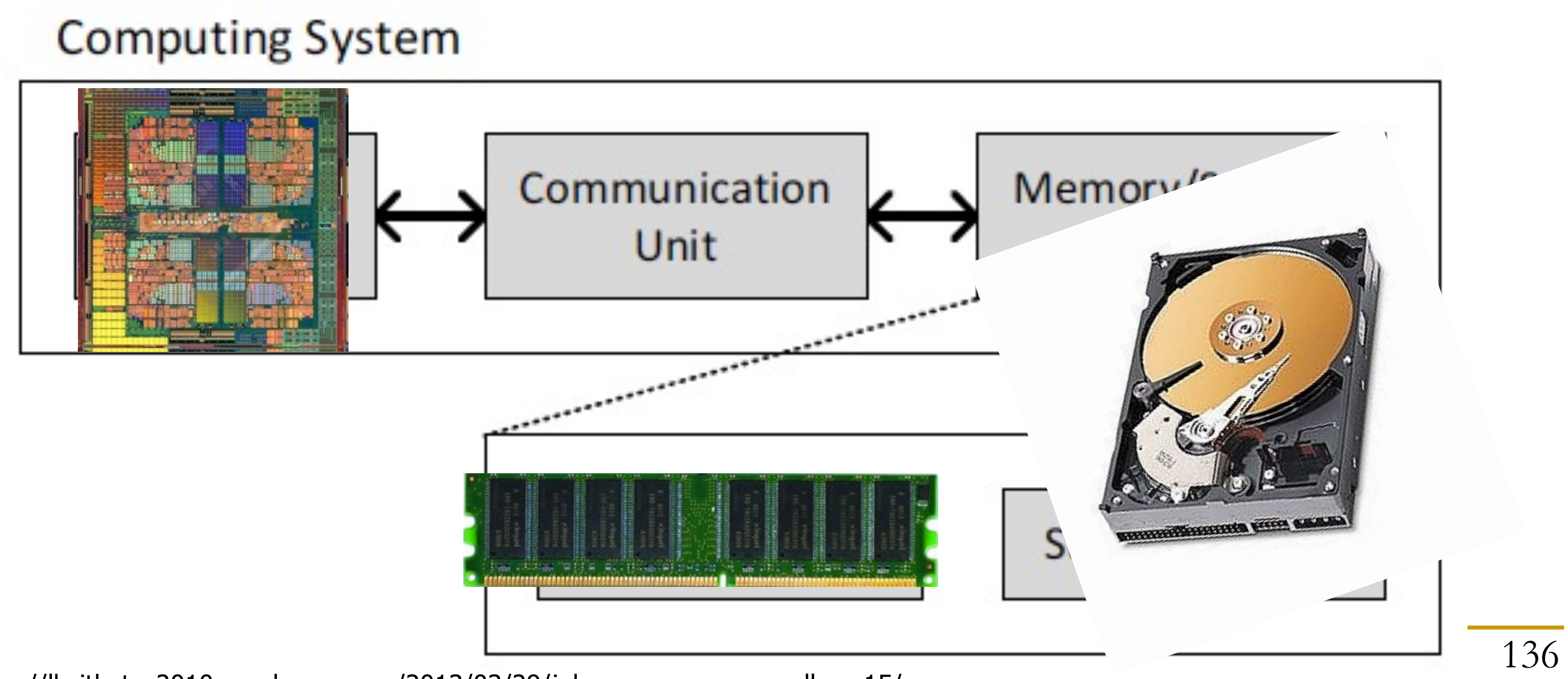
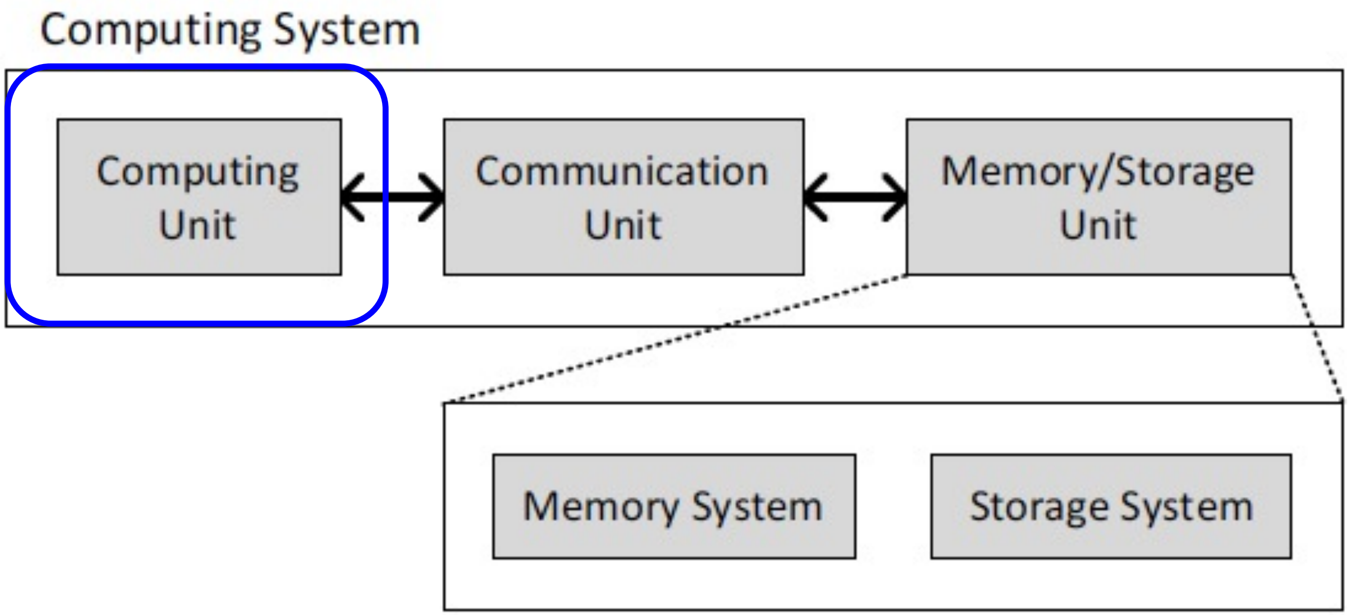


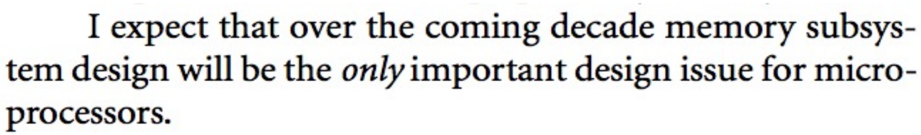
Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

Today's Computing Systems

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

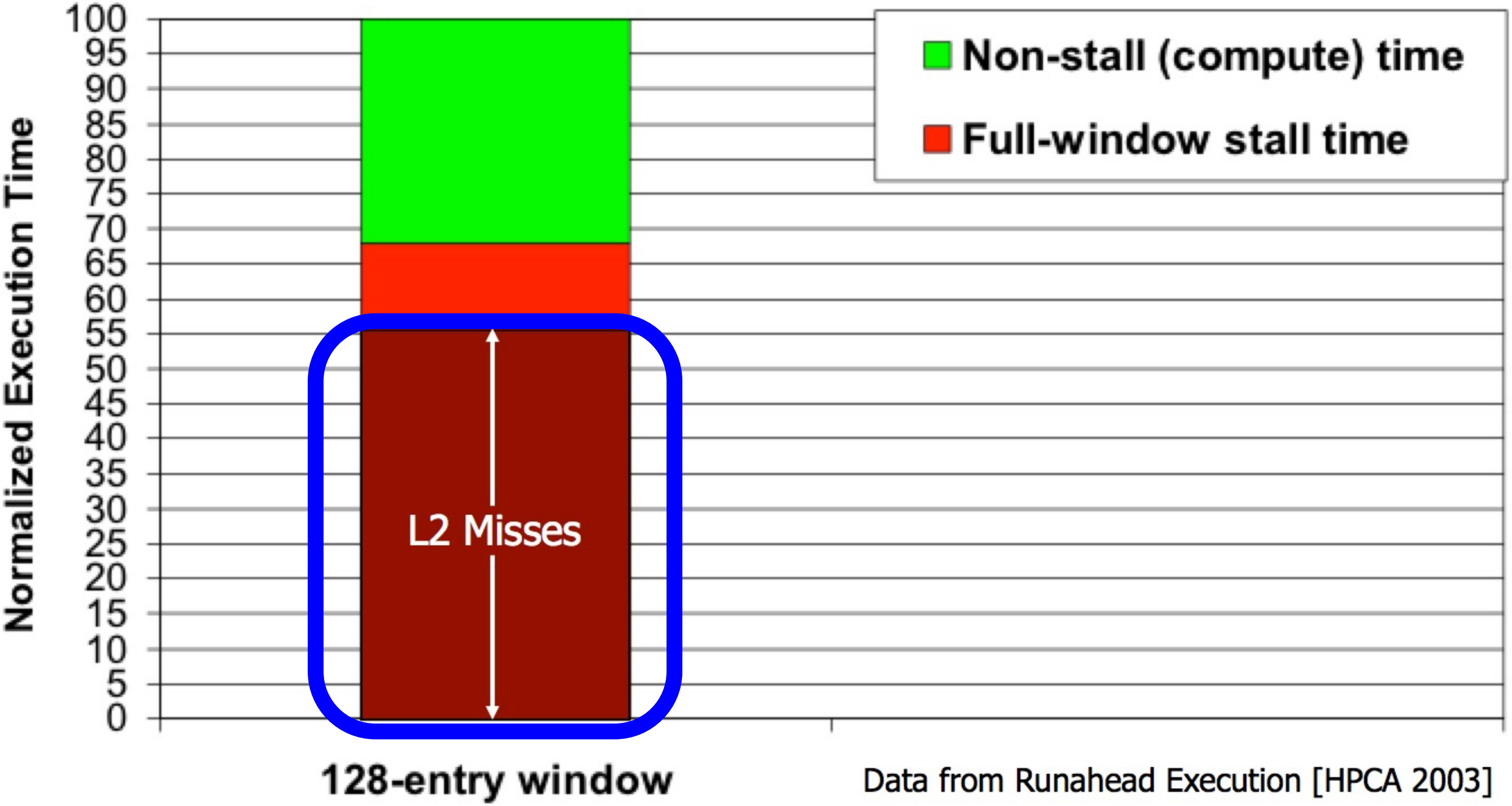


Yet …



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

- "**It's the Memory, Stupid!**" (Richard Sites, MPR, 1996)

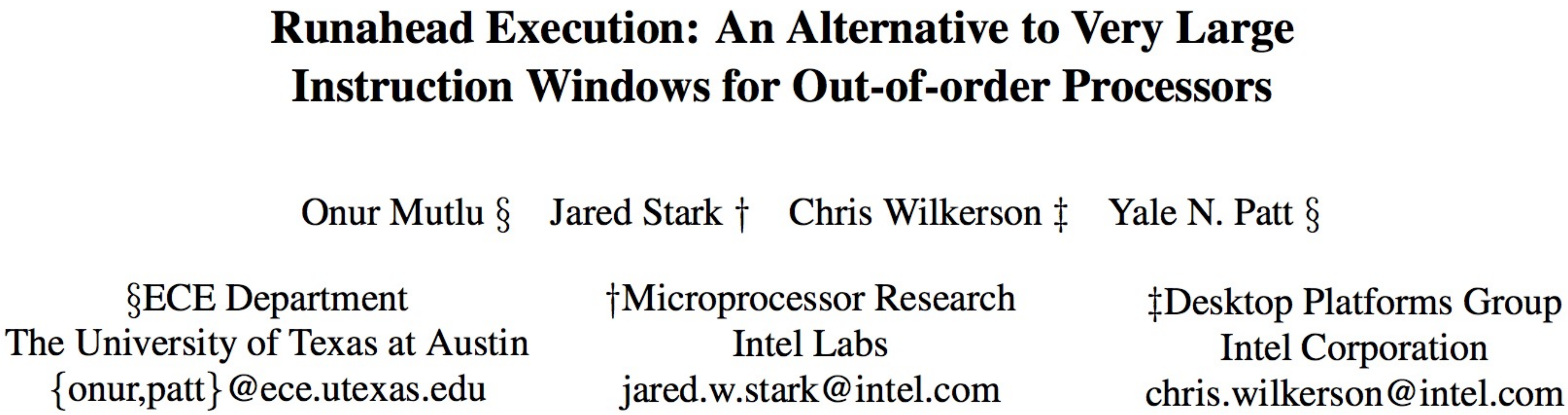


Data from Runahead Execution [HPCA 2003]

Mutlu+, "Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-Order Processors," HPCA 2003.

The Performance Perspective

- Onur Mutlu, Jared Stark, Chris Wilkerson, and Yale N. Patt, **"Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors"** *Proceedings of the 9th International Symposium on High-Performance Computer Architecture* (***HPCA***), pages 129-140, Anaheim, CA, February 2003. Slides (pdf)
  One of the 15 computer arch. papers of 2003 selected as Top Picks by IEEE Micro. HPCA Test of Time Award (awarded in 2021).



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

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

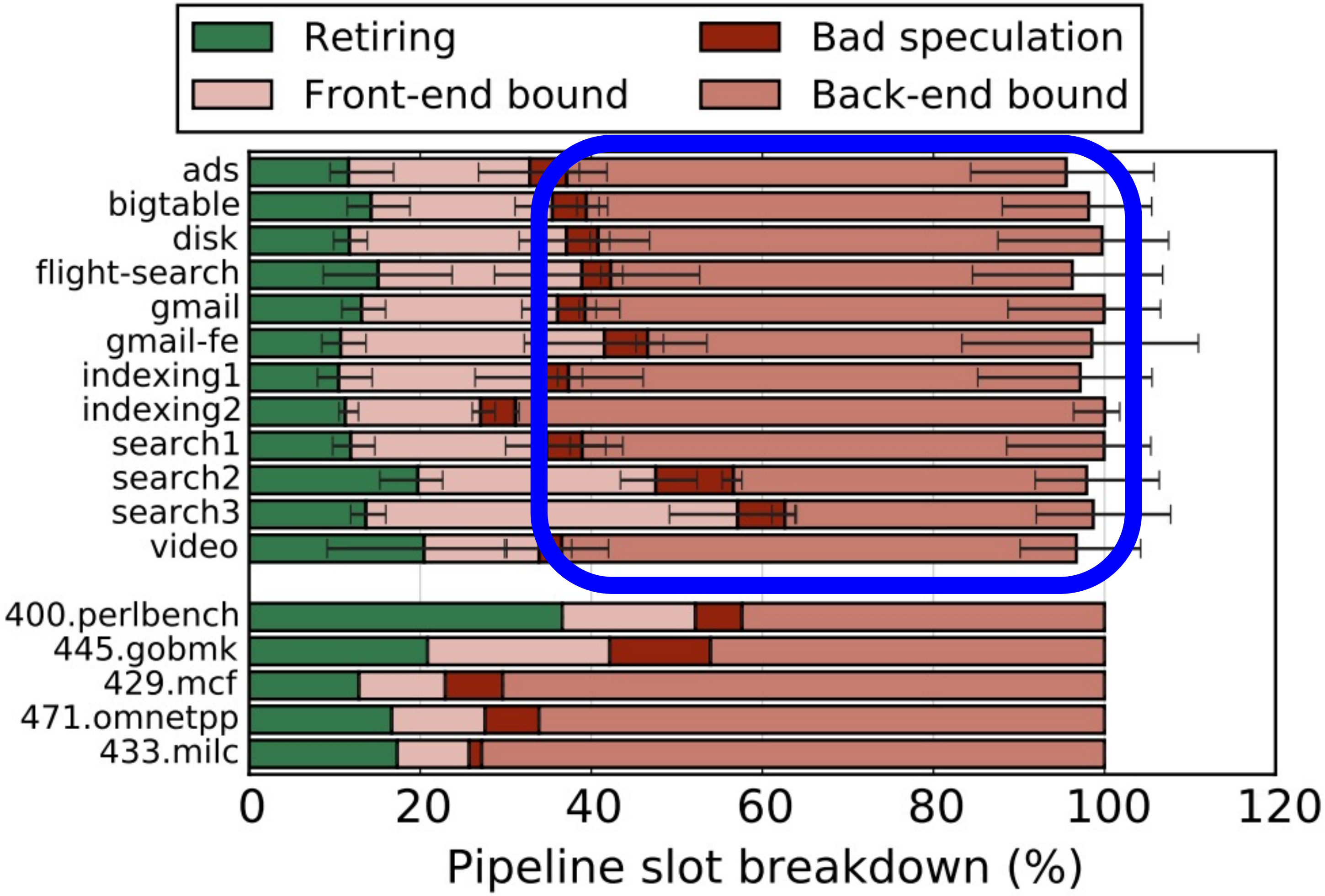
§ECE Department
The University of Texas at Austin
{onur,patt}@ece.utexas.edu

†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):



Kanev+, "Profiling a Warehouse-Scale Computer," ISCA 2015.

The Performance Perspective (Today)

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

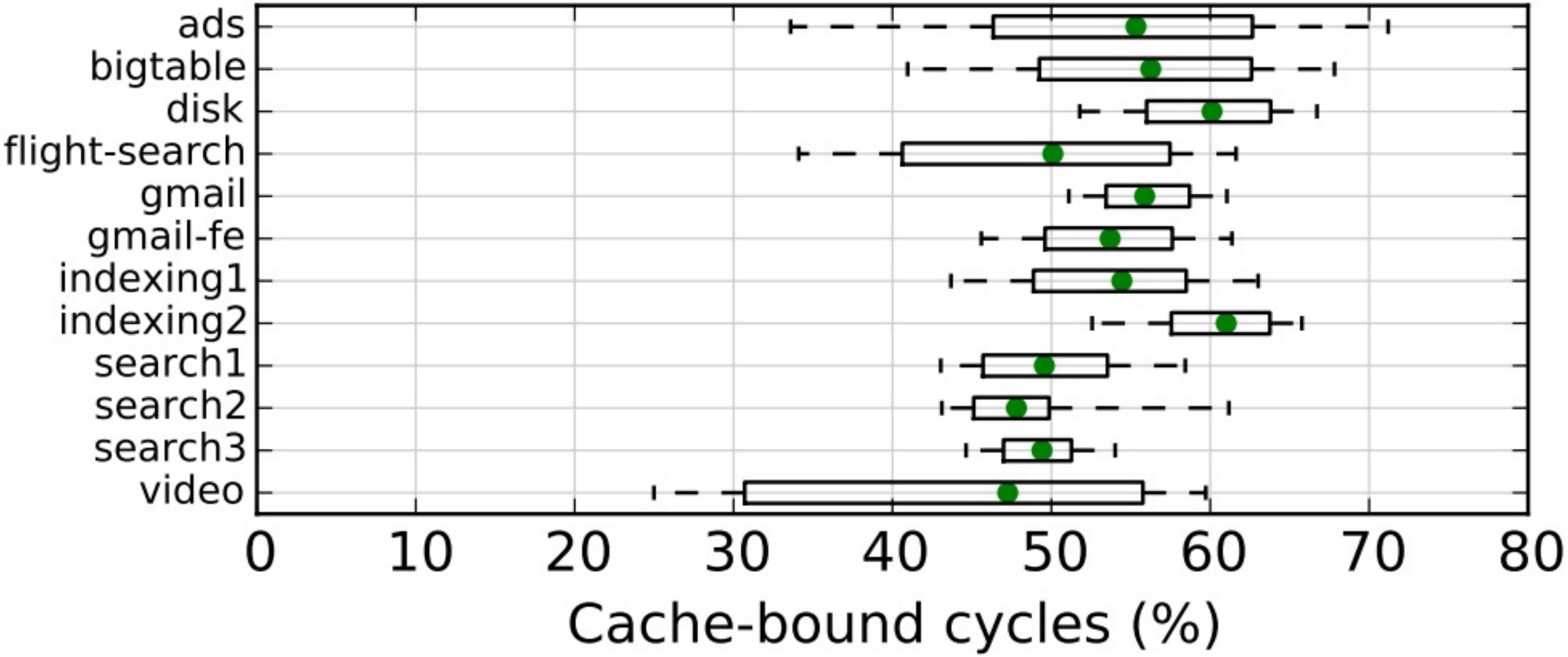


Figure 11: Half of cycles are spent stalled on caches.

Kanev+, "Profiling a Warehouse-Scale Computer," ISCA 2015.

Perils of Processor-Centric Design

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

  → Energy inefficient

  → Low performance

  → Complex

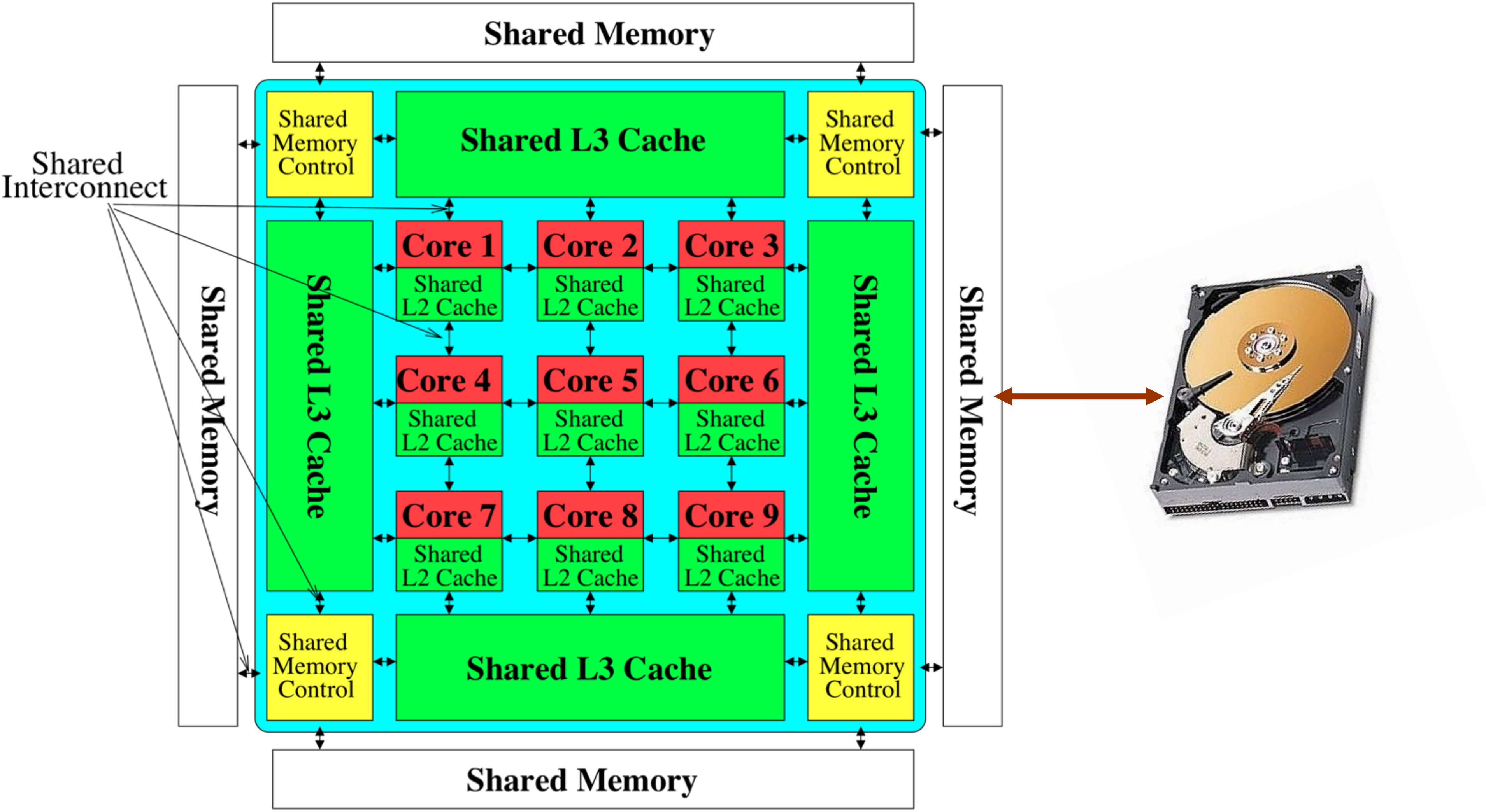
- Overly complex and bloated processor (and accelerators)
  - To tolerate data access from memory
  - Complex hierarchies and mechanisms

  → Energy inefficient

  → Low performance

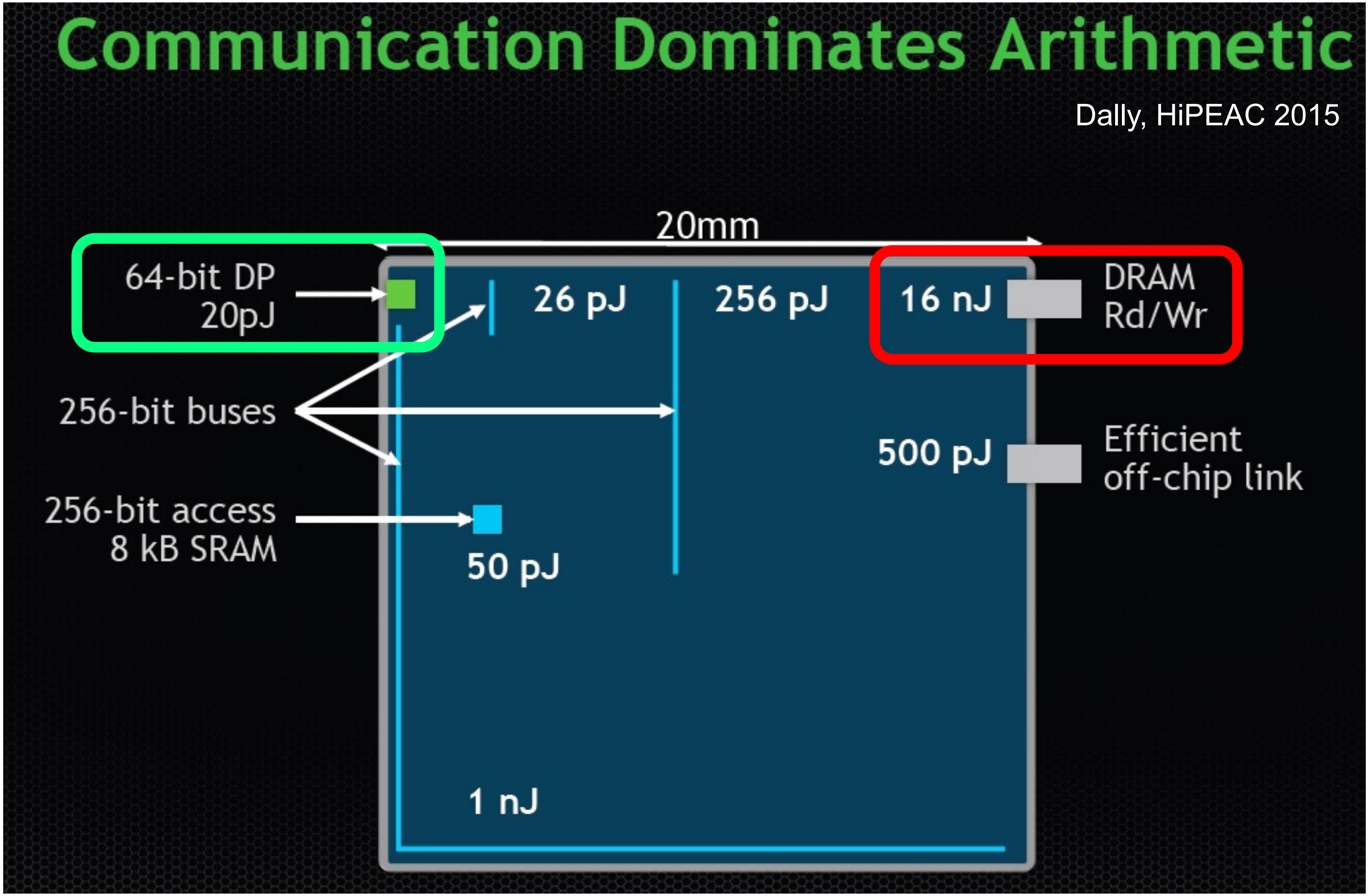
  → Complex

Perils of Processor-Centric Design

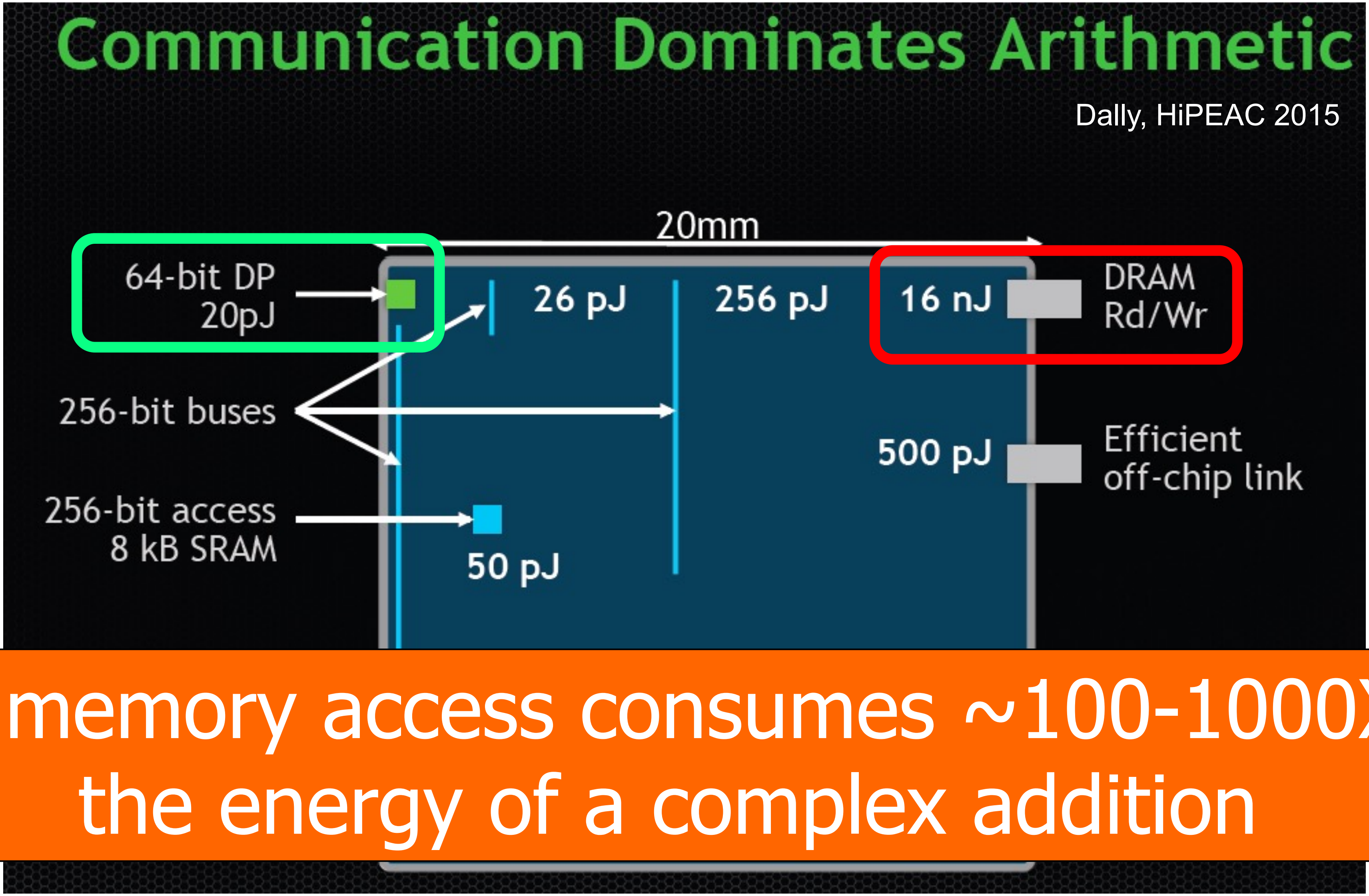


Most of the system is dedicated to storing and moving data

The Energy Perspective



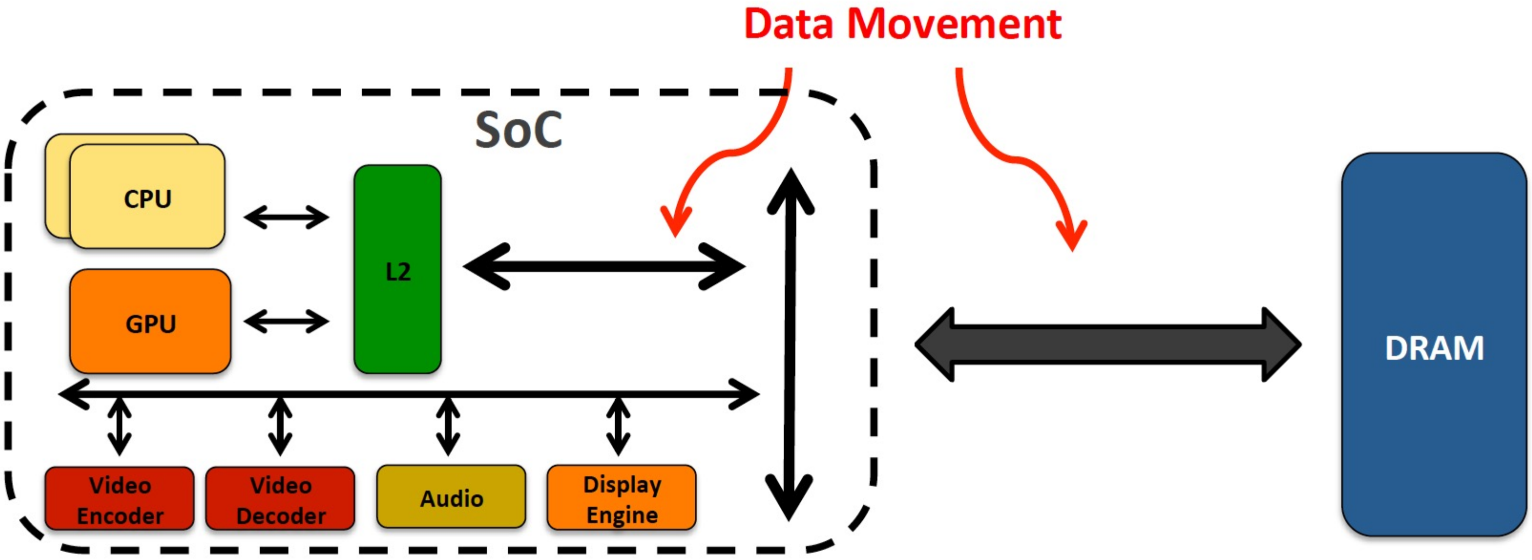
Data Movement vs. Computation Energy



A memory access consumes ~100-1000X the energy of a complex addition

Data Movement vs. Computation Energy

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



[1]: Reducing data Movement Energy via Online Data Clustering and Encoding (MICRO'16)

[2]: Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms (IISWC'14)

Energy Waste in Mobile Devices

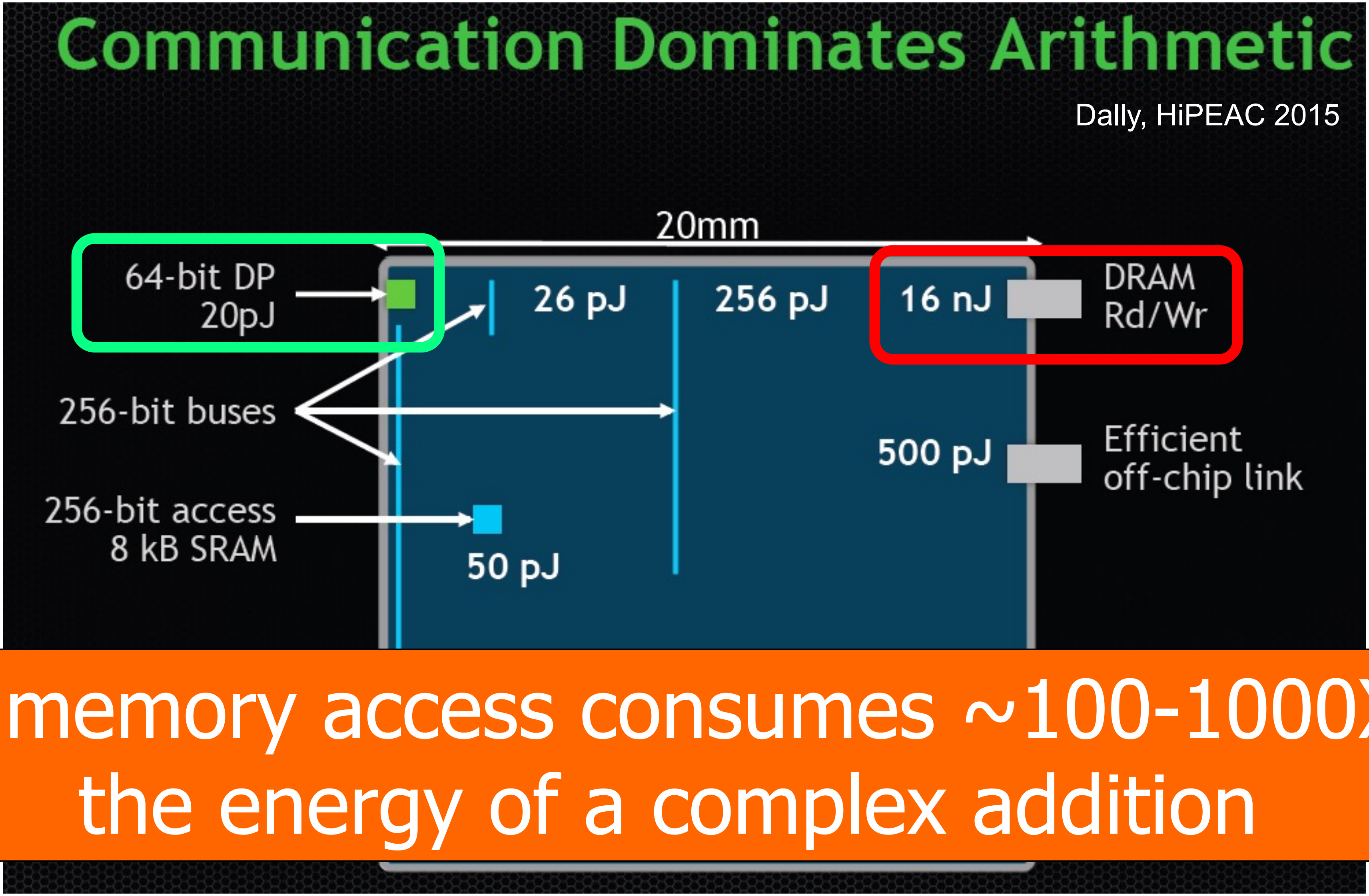
- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu, **"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"** *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems* (***ASPLOS***), Williamsburg, VA, USA, March 2018.

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

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

We Do Not Want to Move Data!



A memory access consumes ~100-1000X the energy of a complex addition

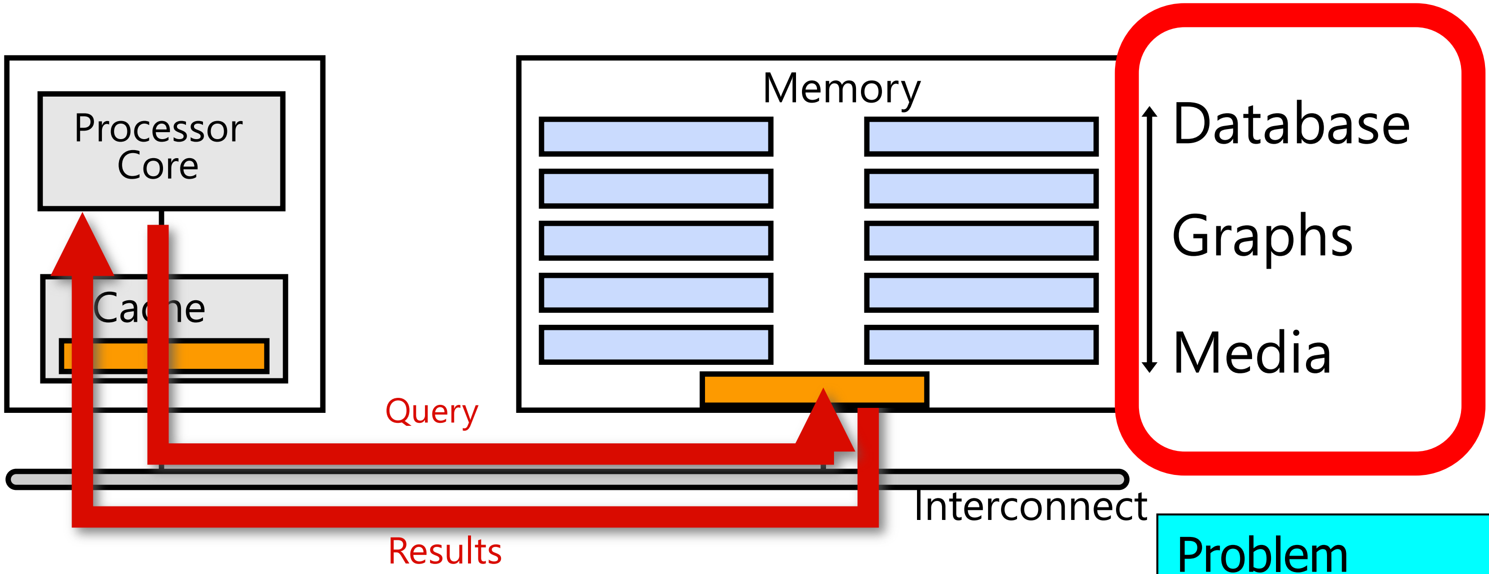
We Need A Paradigm Shift To …

- Enable computation with minimal data movement

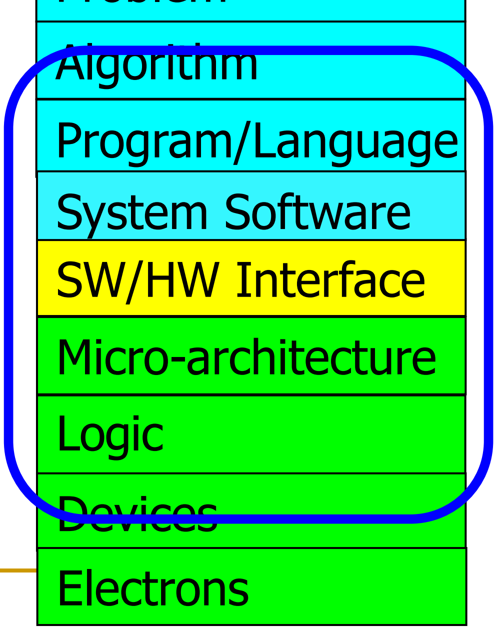
- Compute where it makes sense (where data resides)

- Make computing architectures more data-centric

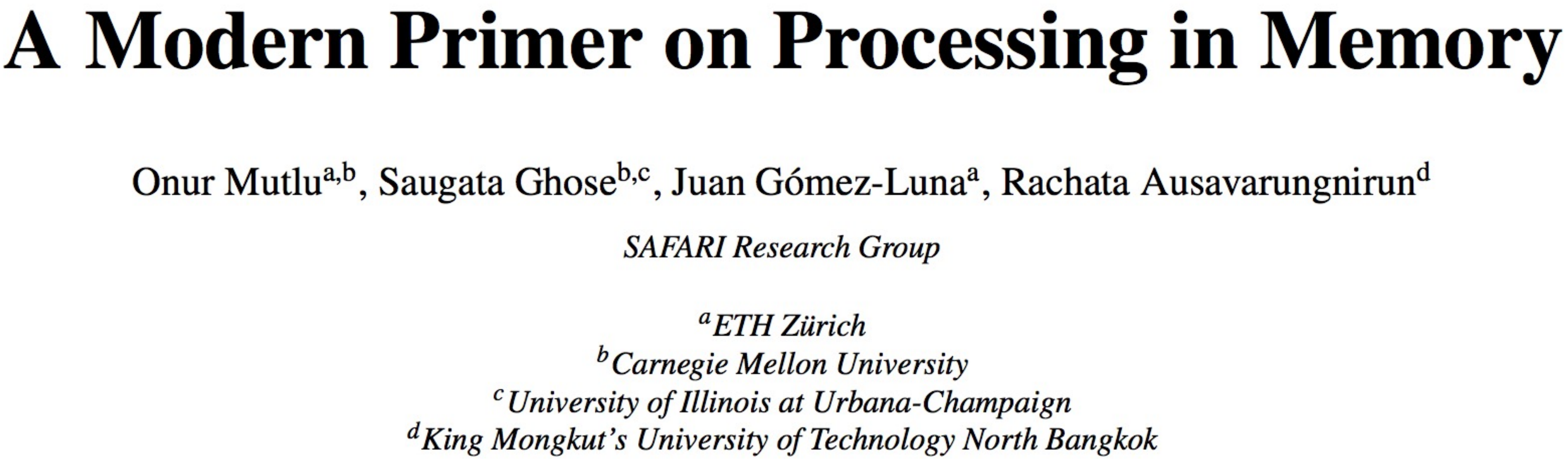
Goal: Processing Inside Memory



- 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?



PIM Review and Open Problems



A Modern Primer on Processing in Memory

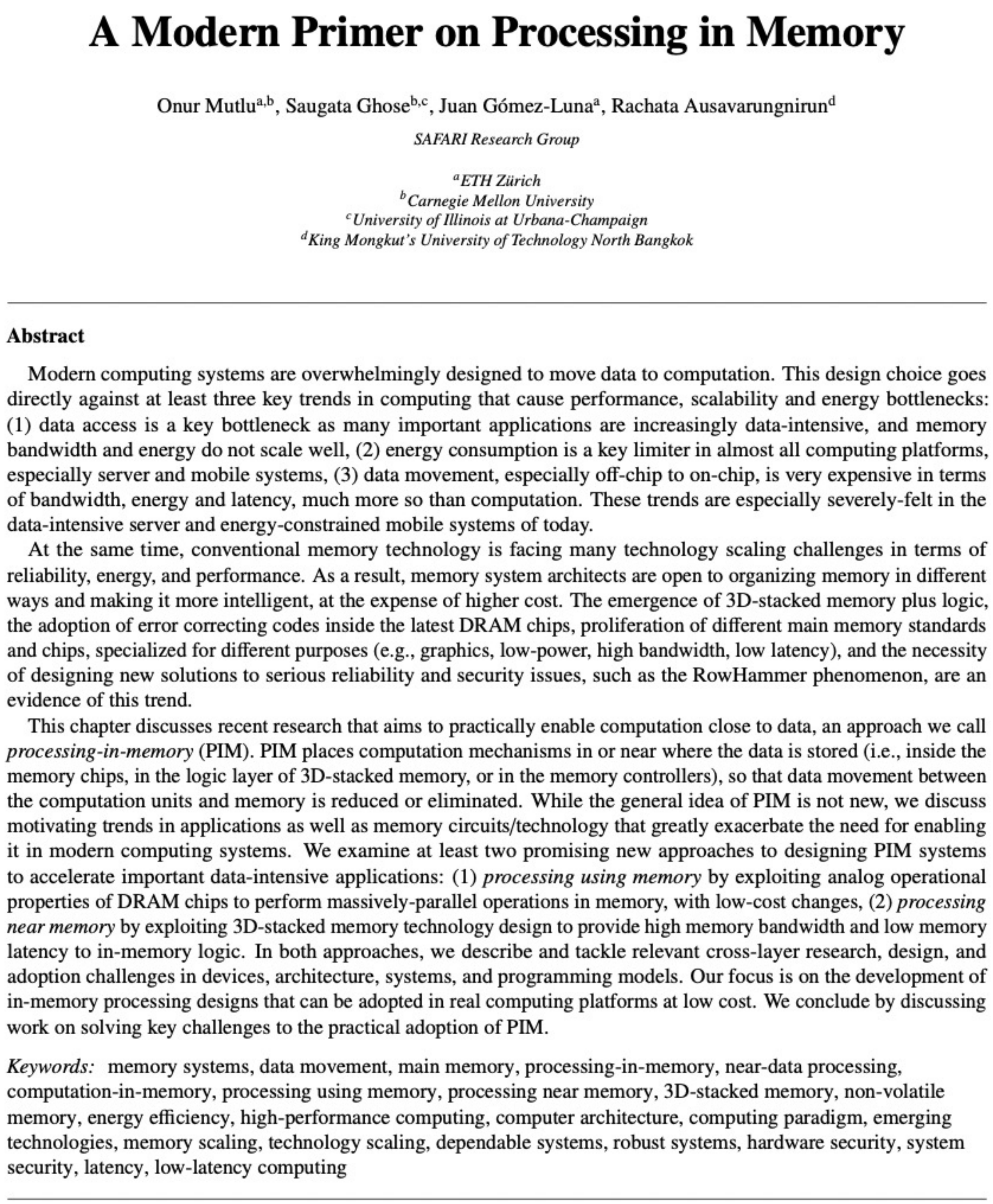
Onur Mutlu[a,b], Saugata Ghose[b,c], Juan Gómez-Luna[a], Rachata Ausavarungnirun[d]

SAFARI Research Group

[a] *ETH Zürich*
[b] *Carnegie Mellon University*
[c] *University of Illinois at Urbana-Champaign*
[d] *King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,
"A Modern Primer on Processing in Memory"
Invited Book Chapter in ***Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann***, Springer, to be published in 2021.

https://arxiv.org/pdf/1903.03988.pdf



A Modern Primer on Processing in Memory

Onur Mutlu[a,b], Saugata Ghose[b,c], Juan Gómez-Luna[a], Rachata Ausavarungnirun[d]

SAFARI Research Group

[a]*ETH Zürich*
[b]*Carnegie Mellon University*
[c]*University of Illinois at Urbana-Champaign*
[d]*King Mongkut's University of Technology North Bangkok*

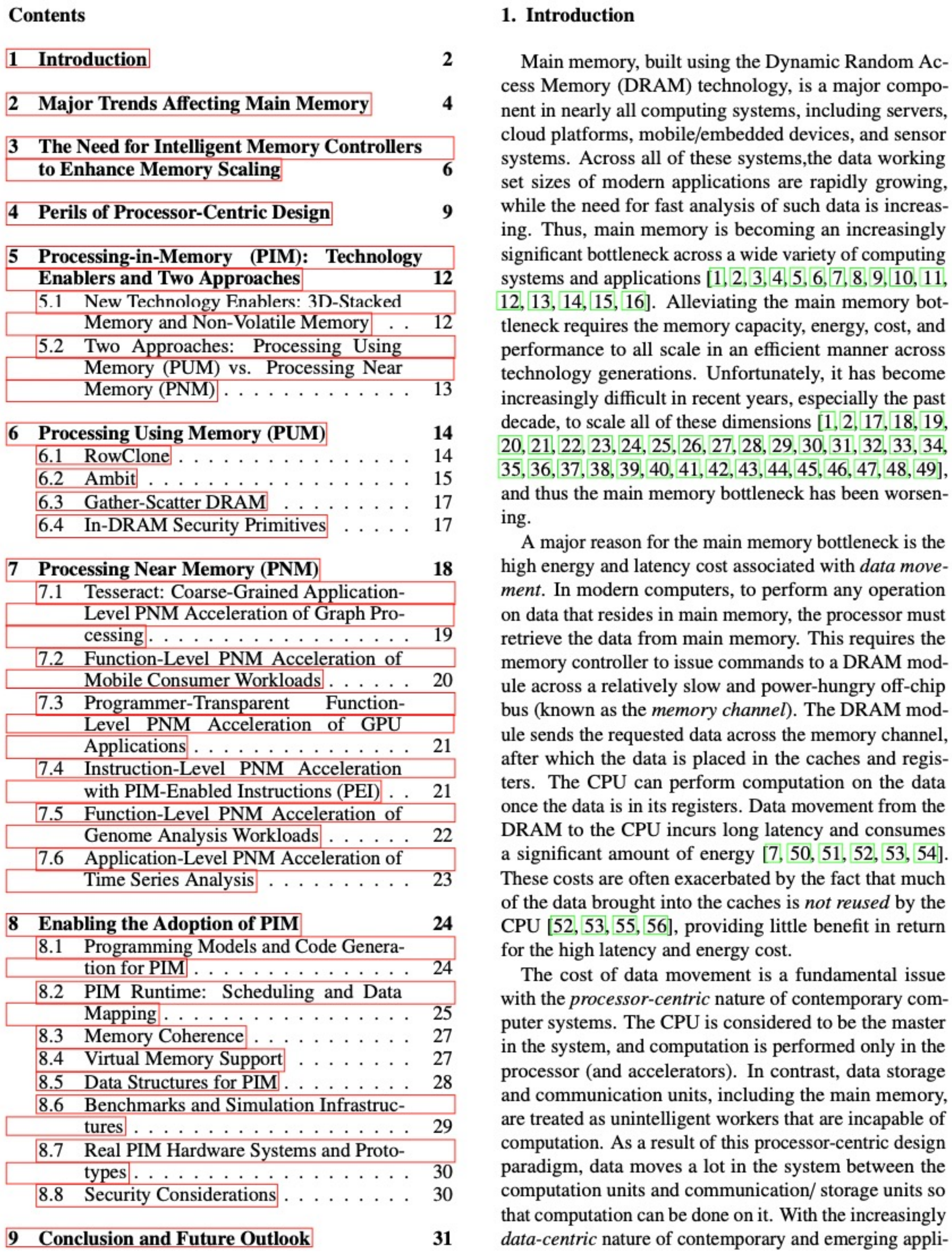
Abstract

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

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

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

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



Contents

1. Introduction

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

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

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

Processing in Memory: Two Approaches

1. Processing using Memory
2. Processing near Memory

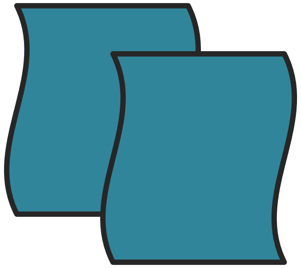
Approach 1: Processing Using Memory

- Take advantage of operational principles of memory to perform bulk data movement and computation in memory
  - Can exploit internal connectivity to move data
  - Can exploit analog computation capability
  - ...

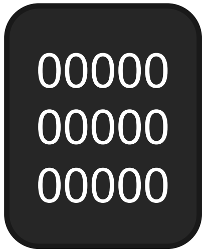
- Examples: RowClone, In-DRAM AND/OR, Gather/Scatter DRAM
  - RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data (Seshadri et al., MICRO 2013)
  - Fast Bulk Bitwise AND and OR in DRAM (Seshadri et al., IEEE CAL 2015)
  - Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses (Seshadri et al., MICRO 2015)
  - "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology" (Seshadri et al., MICRO 2017)

Starting Simple: Data Copy and Initialization

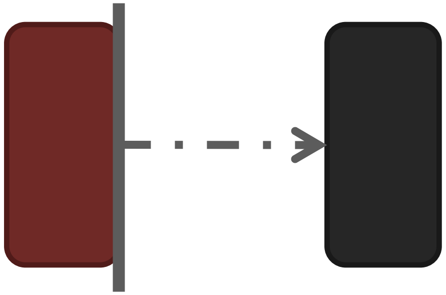
memmove & *memcpy*: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking

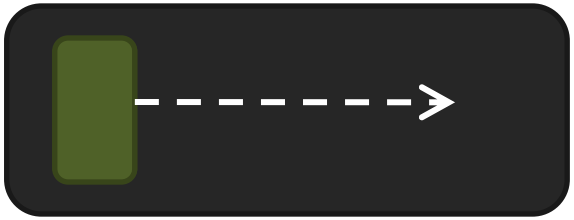


Zero initialization
(e.g., security)



Checkpointing

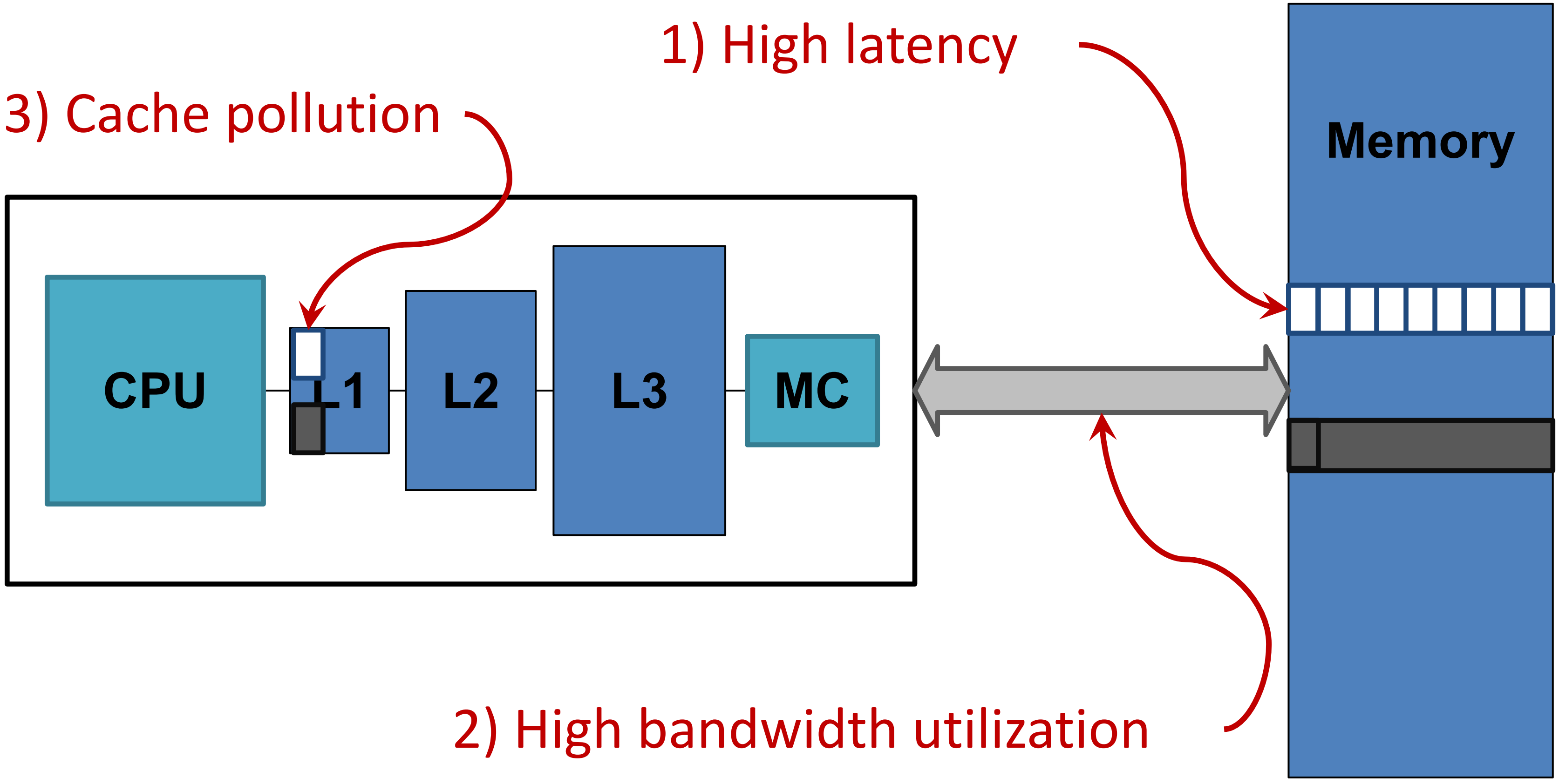
VM Cloning
Deduplication



Page Migration

• • •
Many more

Today's Systems: Bulk Data Copy



1) High latency

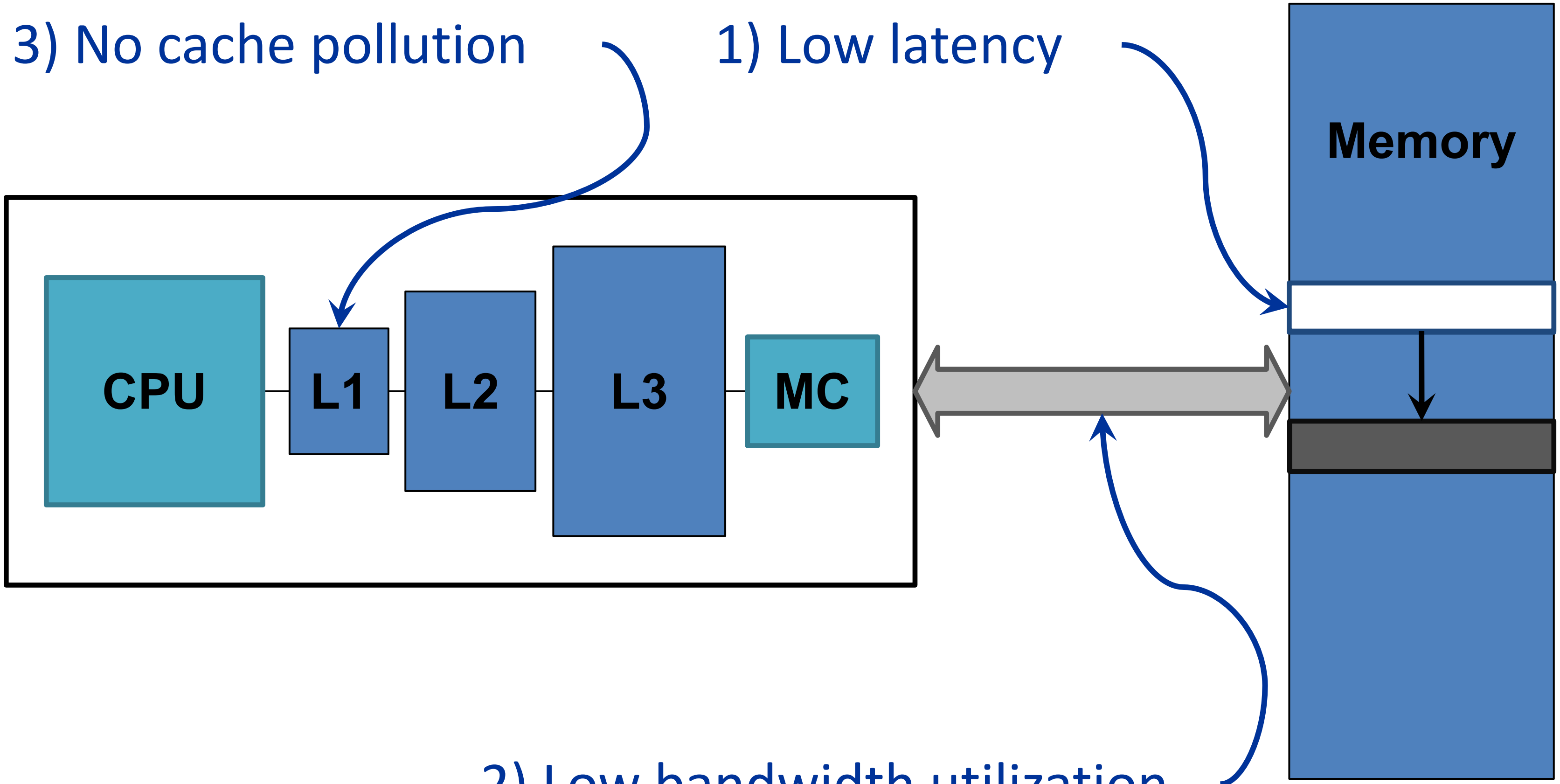
3) Cache pollution

2) High bandwidth utilization

4) Unwanted data movement

1046ns, 3.6uJ (for 4KB page copy via DMA)

Future Systems: In-Memory Copy



3) No cache pollution

1) Low latency

2) Low bandwidth utilization

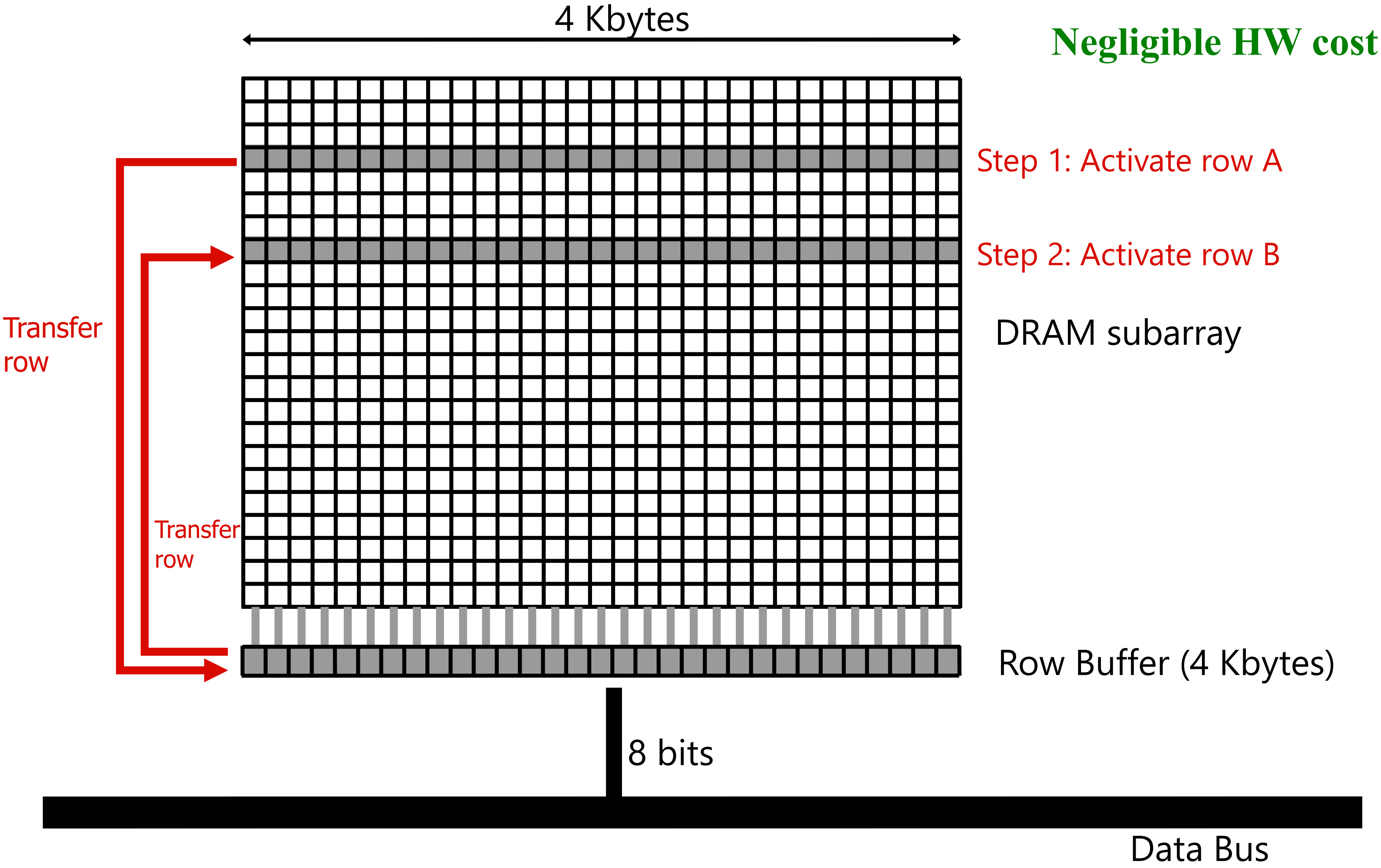
4) No unwanted data movement

1046ns, 3.6uJ → 90ns, 0.04uJ

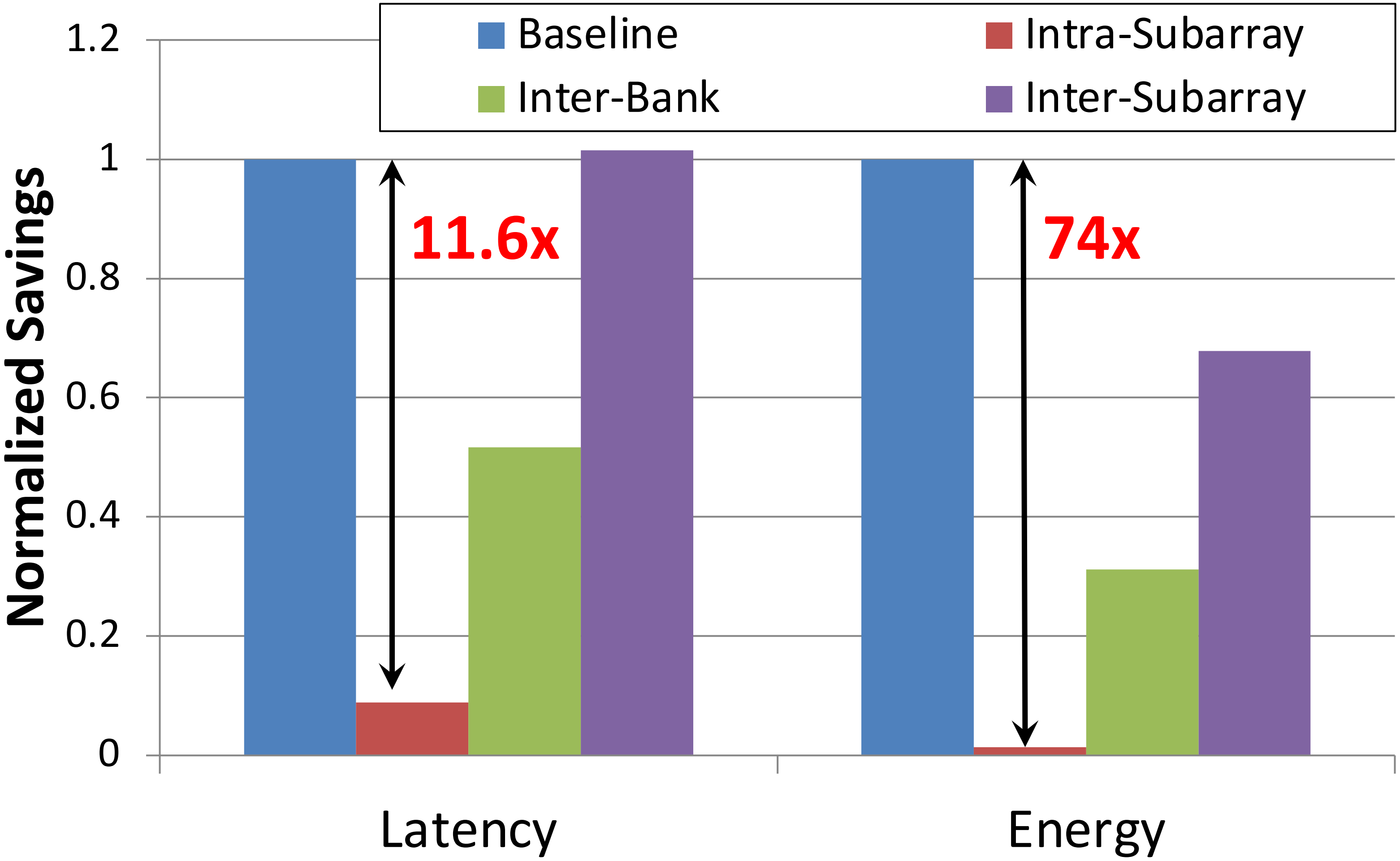
RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates

Negligible HW cost



RowClone: Latency and Energy Savings



Seshadri et al., "RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data," MICRO 2013.

More on RowClone

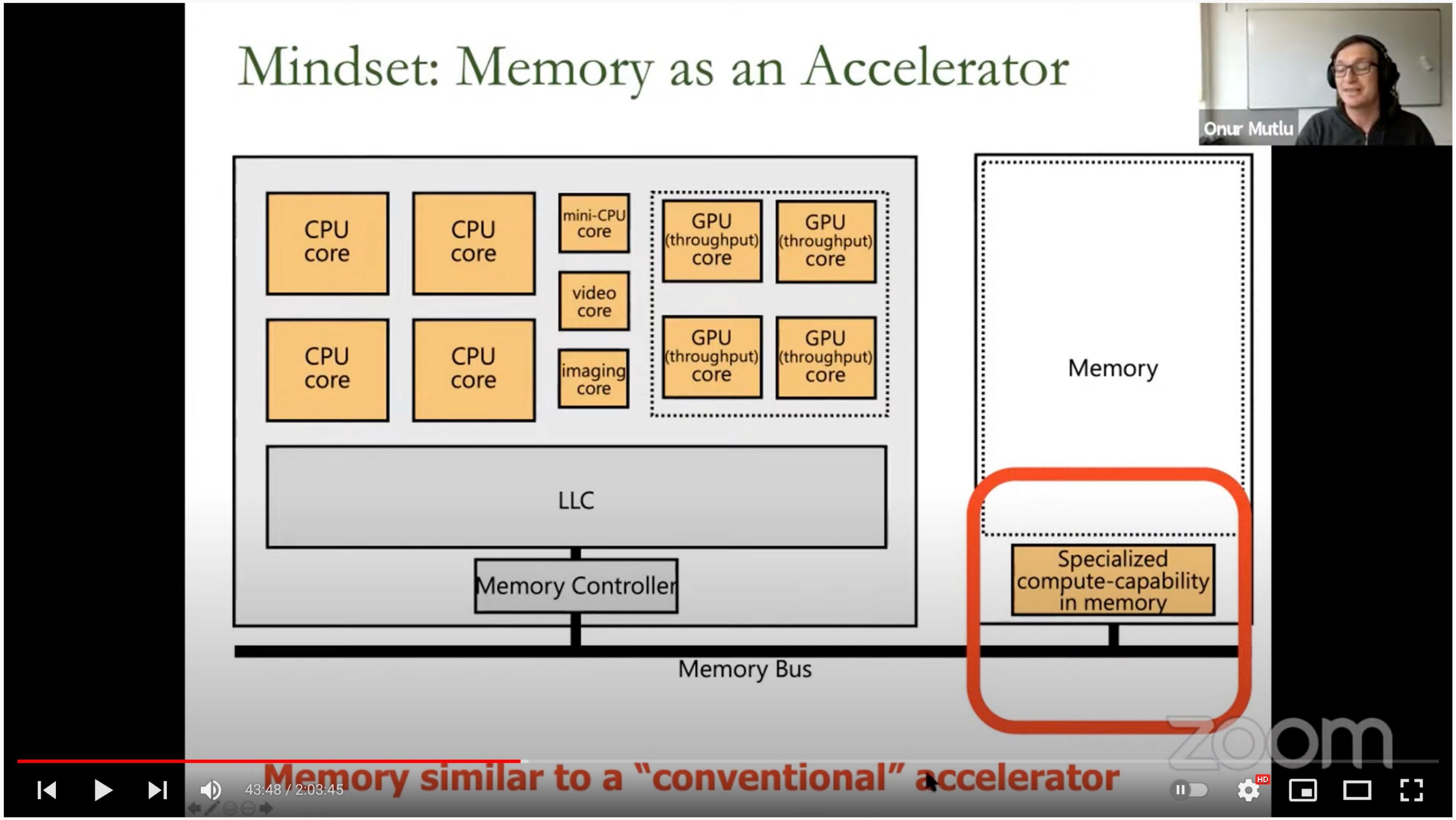
- Vivek Seshadri, Yoongu Kim, Chris Fallin, Donghyuk Lee, Rachata Ausavarungnirun, Gennady Pekhimenko, Yixin Luo, Onur Mutlu, Michael A. Kozuch, Phillip B. Gibbons, and Todd C. Mowry, **"RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization"** *Proceedings of the 46th International Symposium on Microarchitecture (**MICRO**)*, Davis, CA, December 2013. [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]

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

Vivek Seshadri vseshadr@cs.cmu.edu
Yoongu Kim yoongukim@cmu.edu
Chris Fallin* cfallin@c1f.net
Donghyuk Lee donghyuk1@cmu.edu
Rachata Ausavarungnirun rachata@cmu.edu
Gennady Pekhimenko gpekhime@cs.cmu.edu
Yixin Luo yixinluo@andrew.cmu.edu
Onur Mutlu onur@cmu.edu
Phillip B. Gibbons† phillip.b.gibbons@intel.com
Michael A. Kozuch† michael.a.kozuch@intel.com
Todd C. Mowry tcm@cs.cmu.edu

Carnegie Mellon University †Intel Pittsburgh

Lecture on RowClone & Processing using DRAM



Mindset: Memory as an Accelerator

Memory similar to a "conventional" accelerator

DEPARTMENT OF INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING (D-ITET)

Seminar in Computer Arch. - Meeting 3: RowClone: In-Memory Data Copy and Initialization (Fall 2021)

292 views • Streamed live on Oct 7, 2021

Onur Mutlu Lectures
19.1K subscribers

https://www.youtube.com/watch?v=n6Pwg1qax_E&list=PL5Q2soXY2Zi_7UBNmC9B8Yr5JSwTG9yH4&index=4

RowClone Extensions and Follow-Up Work

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

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

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

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

LISA: Increasing Connectivity in DRAM

- 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"
*Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (**HPCA**)*, Barcelona, Spain, March 2016.
[Slides (pptx) (pdf)]
[Source Code]

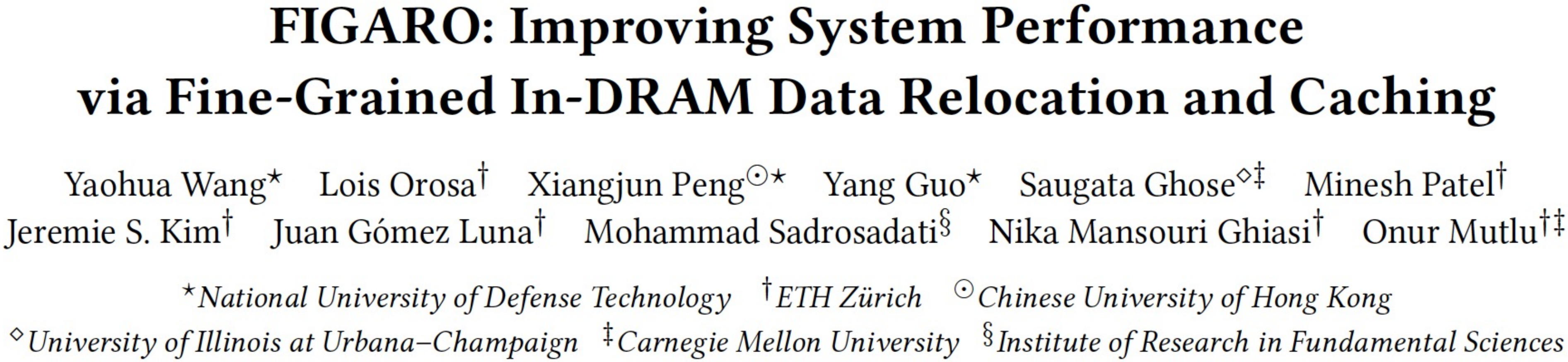
Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM

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

[†]*Carnegie Mellon University* [*]*Georgia Institute of Technology*

FIGARO: Fine-Grained In-DRAM Copy

- Yaohua Wang, Lois Orosa, Xiangjun Peng, Yang Guo, Saugata Ghose, Minesh Patel, Jeremie S. Kim, Juan Gómez Luna, Mohammad Sadrosadati, Nika Mansouri Ghiasi, and Onur Mutlu, **"FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching"** *Proceedings of the 53rd International Symposium on Microarchitecture* (***MICRO***), Virtual, October 2020.



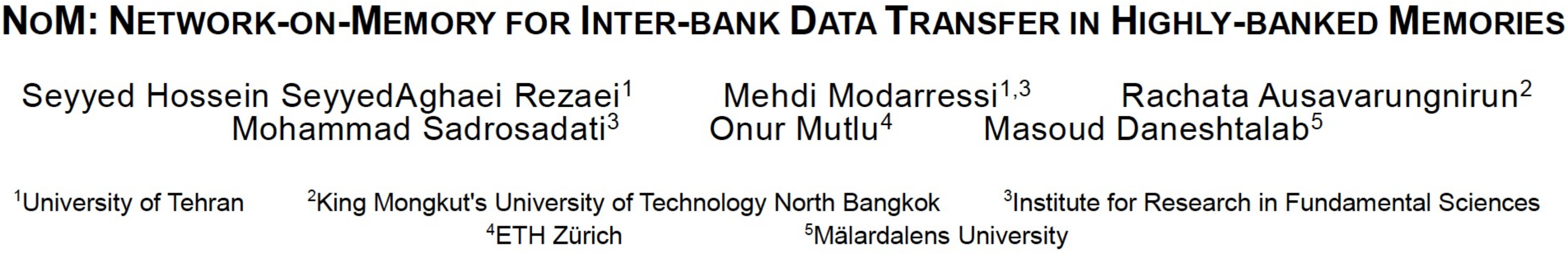
FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching

Yaohua Wang[⋆] Lois Orosa[†] Xiangjun Peng[⊙⋆] Yang Guo[⋆] Saugata Ghose[⋄‡] Minesh Patel[†]
Jeremie S. Kim[†] Juan Gómez Luna[†] Mohammad Sadrosadati[§] Nika Mansouri Ghiasi[†] Onur Mutlu[†‡]

[⋆]*National University of Defense Technology* [†]*ETH Zürich* [⊙]*Chinese University of Hong Kong*
[⋄]*University of Illinois at Urbana–Champaign* [‡]*Carnegie Mellon University* [§]*Institute of Research in Fundamental Sciences*

Network-On-Memory: Fast Inter-Bank Copy

- Seyyed Hossein SeyyedAghaei Rezaei, Mehdi Modarressi, Rachata Ausavarungnirun, Mohammad Sadrosadati, Onur Mutlu, and Masoud Daneshtalab,
  "NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories"
  IEEE Computer Architecture Letters ***(CAL)***, to appear in 2020.

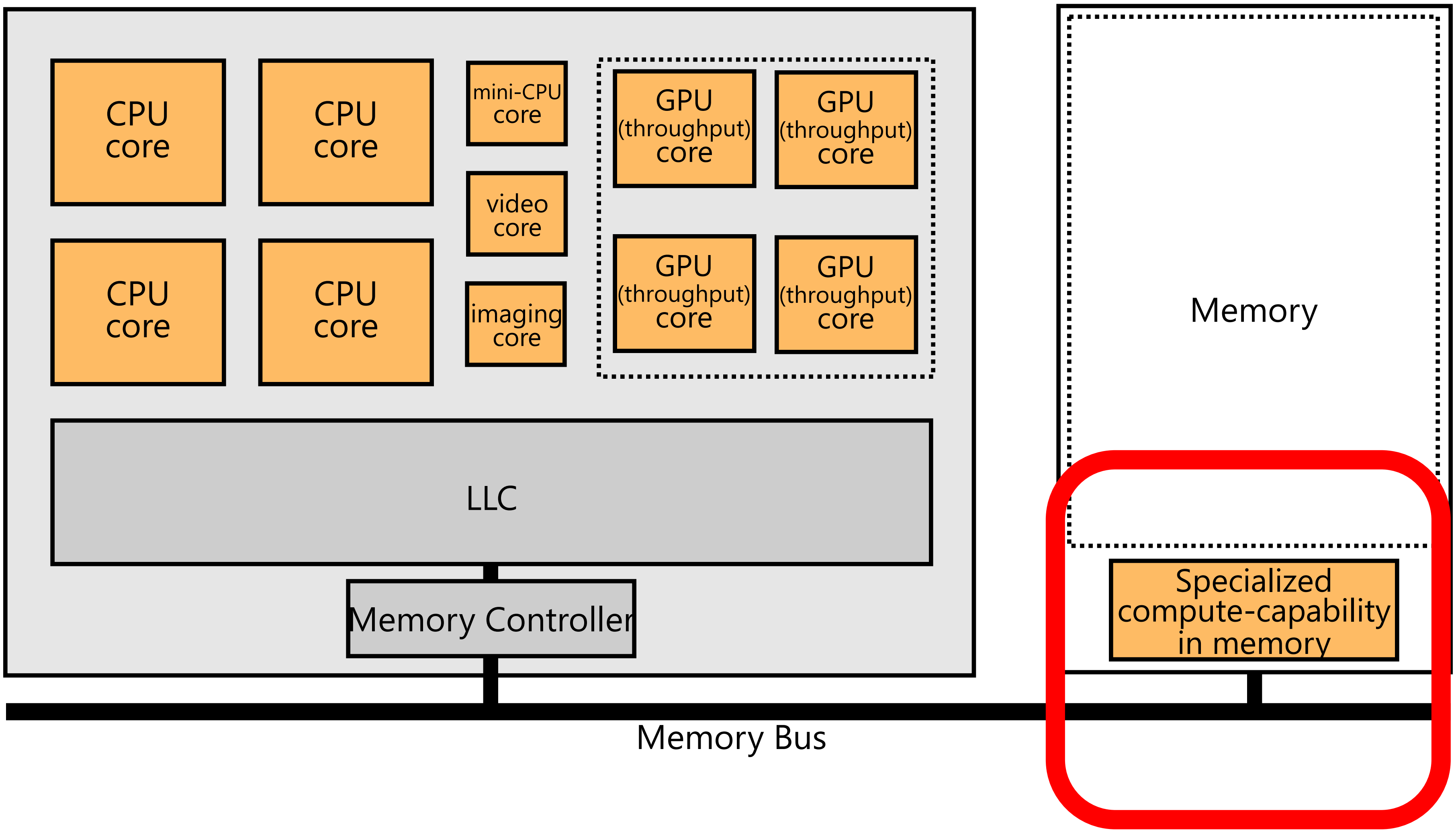


NoM: Network-on-Memory for Inter-bank Data Transfer in Highly-banked Memories

Seyyed Hossein SeyyedAghaei Rezaei[1] Mehdi Modarressi[1,3] Rachata Ausavarungnirun[2]
Mohammad Sadrosadati[3] Onur Mutlu[4] Masoud Daneshtalab[5]

[1]University of Tehran [2]King Mongkut's University of Technology North Bangkok [3]Institute for Research in Fundamental Sciences
[4]ETH Zürich [5]Mälardalens University

Mindset: Memory as an Accelerator



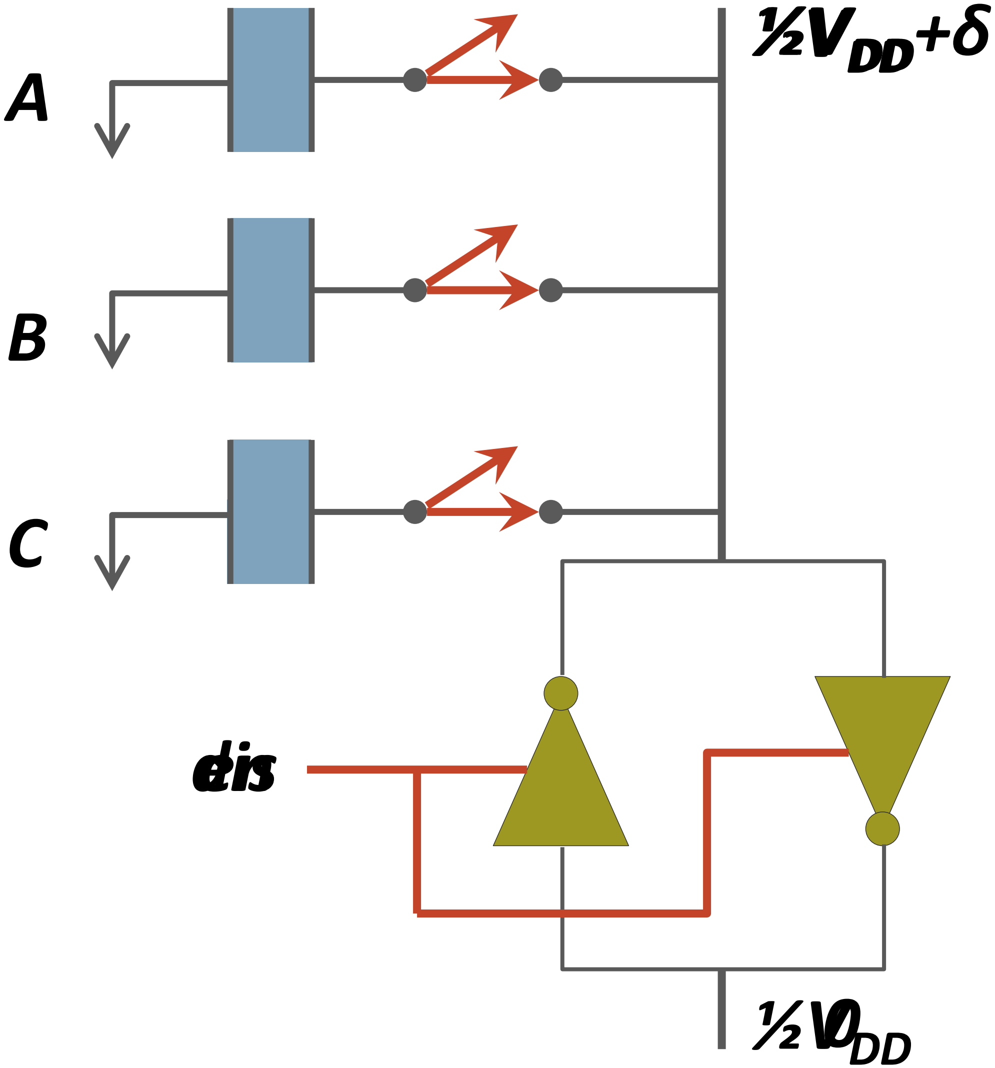
Memory similar to a "conventional" accelerator

(Truly) In-Memory Computation

- We can also support in-DRAM AND, OR, NOT, MAJ
- At low cost
- Using analog computation capability of DRAM
  - Idea: activating multiple rows performs computation
- 30-60X performance and energy improvement
  - Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology," MICRO 2017.

- New memory technologies enable even more opportunities
  - Memristors, resistive RAM, phase change mem, STT-MRAM, …
  - Can operate on data with minimal movement

In-DRAM AND/OR: Triple Row Activation



Final State

AB + BC + AC

C(A + B) + ~C(AB)

Seshadri+, “Fast Bulk Bitwise AND and OR in DRAM”, IEEE CAL 2015.

In-DRAM Bulk Bitwise AND/OR Operation

- BULKAND A, B → C
- Semantics: Perform a bitwise AND of two rows A and B and store the result in row C

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

1. RowClone A into D1
2. RowClone B into D2
3. RowClone R0 into D3
4. ACTIVATE D1,D2,D3
5. RowClone Result into C

In-DRAM NOT: Dual Contact Cell

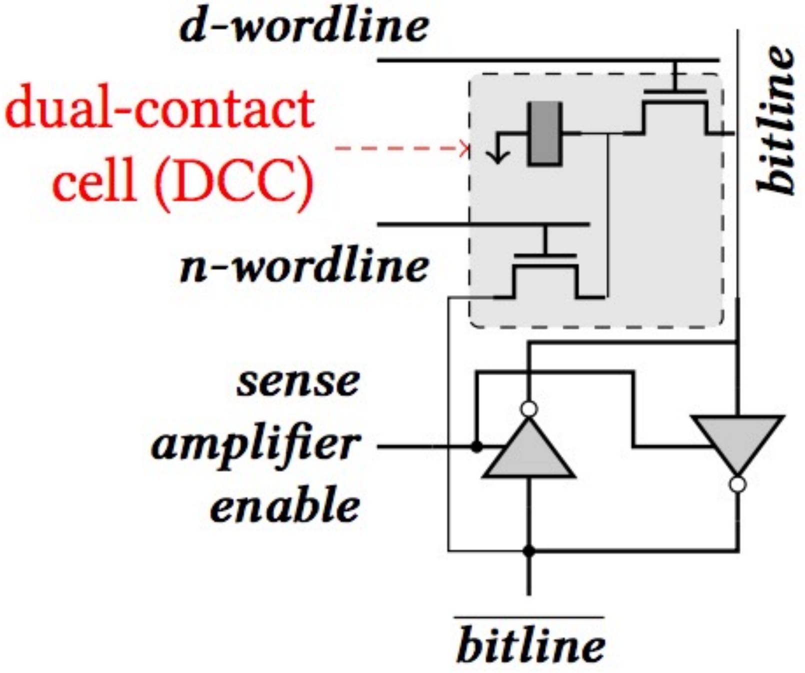
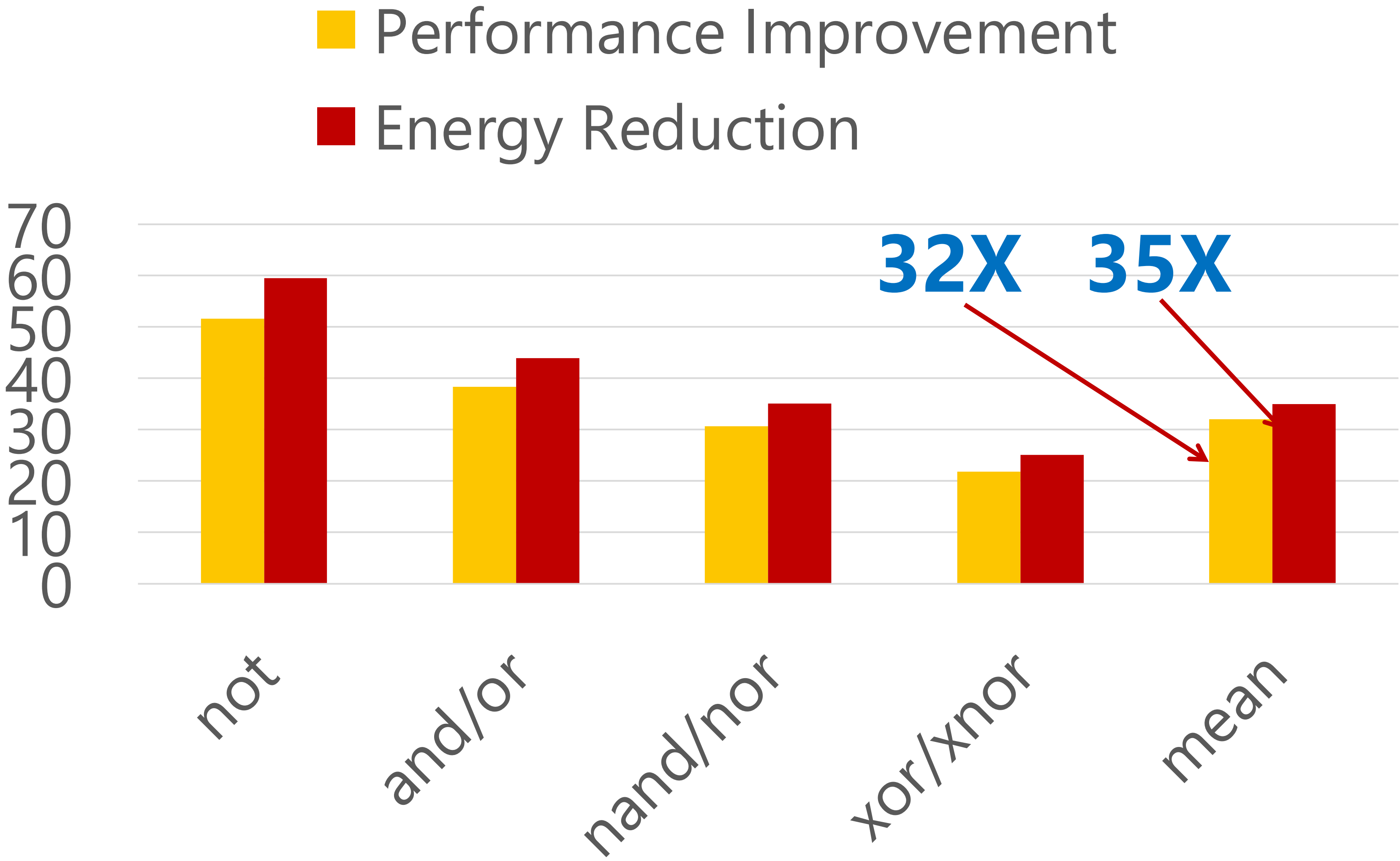


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

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

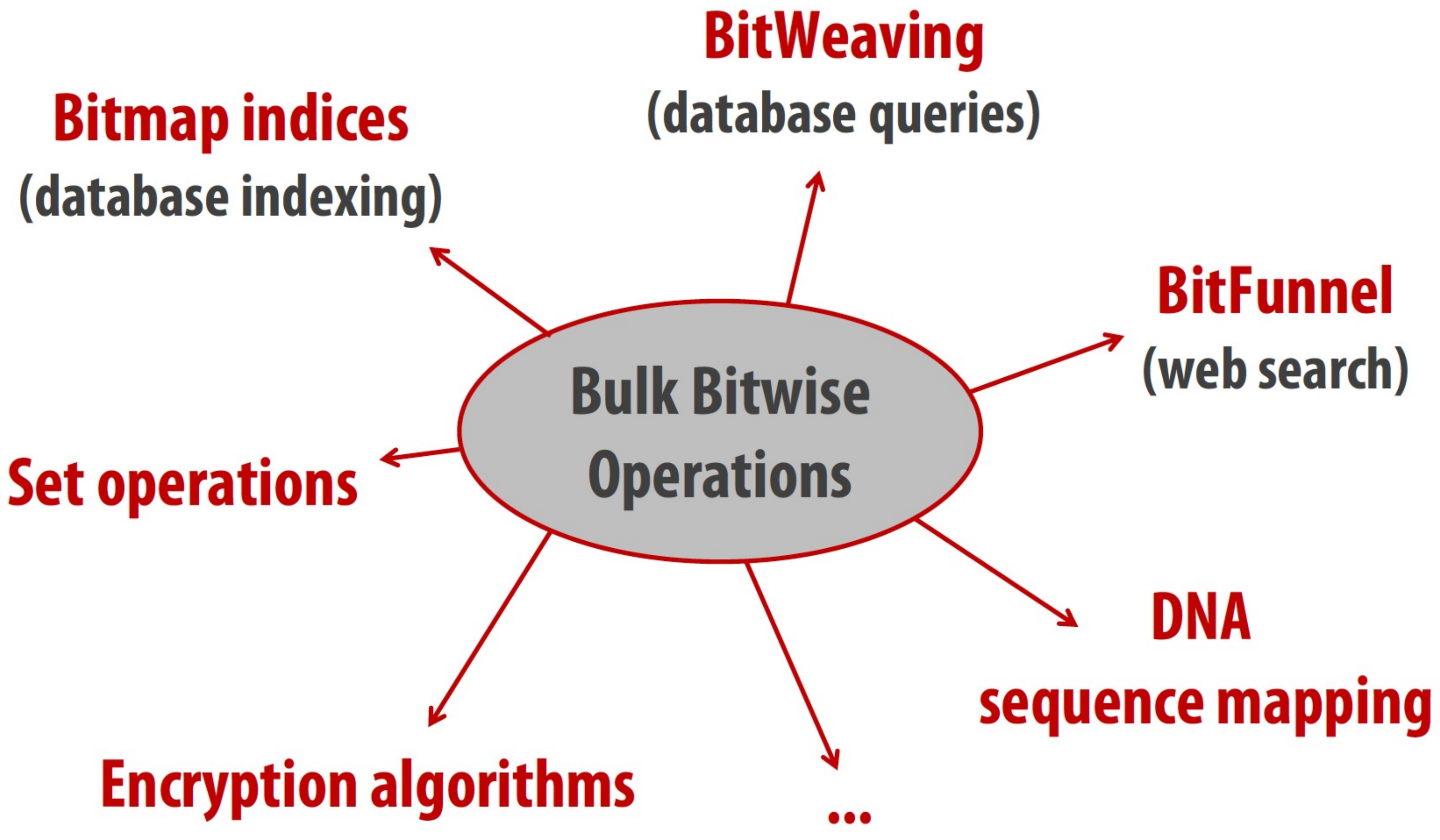
Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

Ambit vs. DDR3: Performance and Energy



Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

Bulk Bitwise Operations in Workloads



[1] Li and Patel, BitWeaving, SIGMOD 2013
[2] Goodwin+, BitFunnel, SIGIR 2017

Performance: Bitmap Index on Ambit

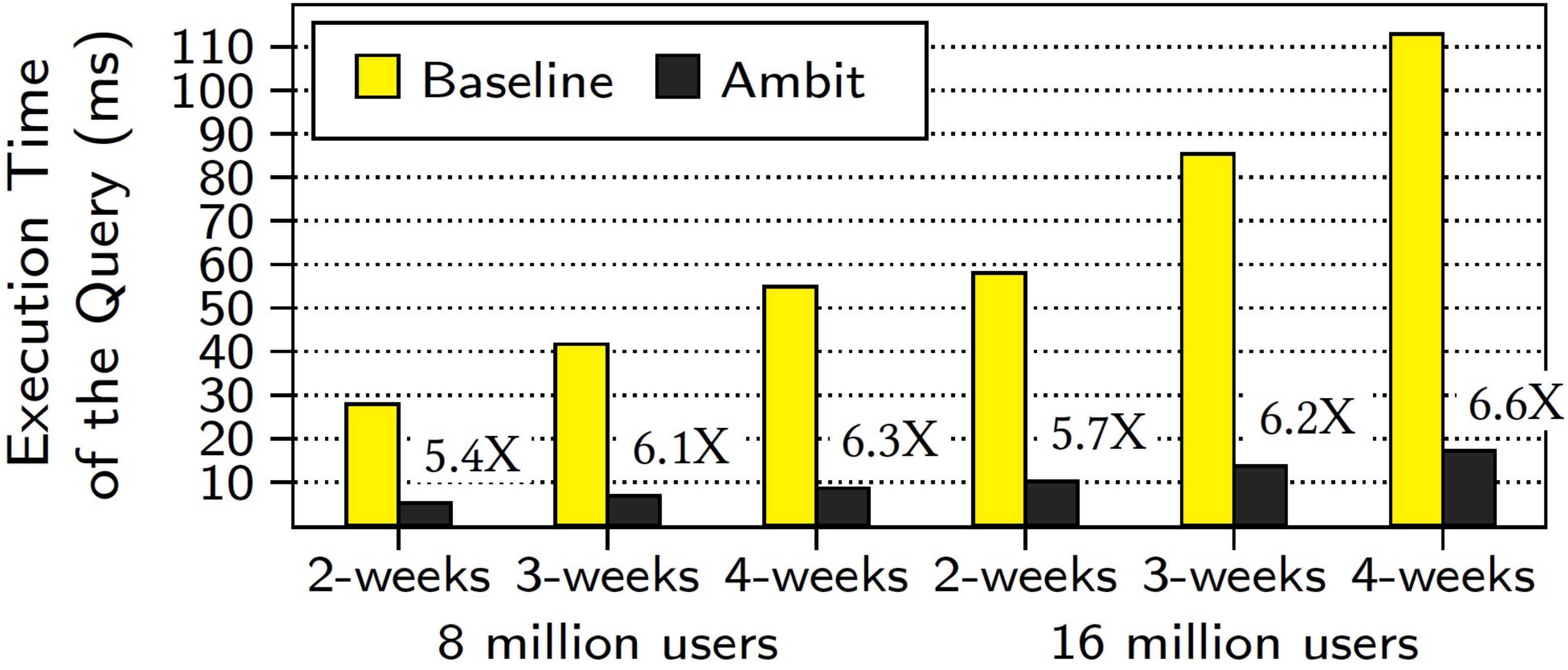
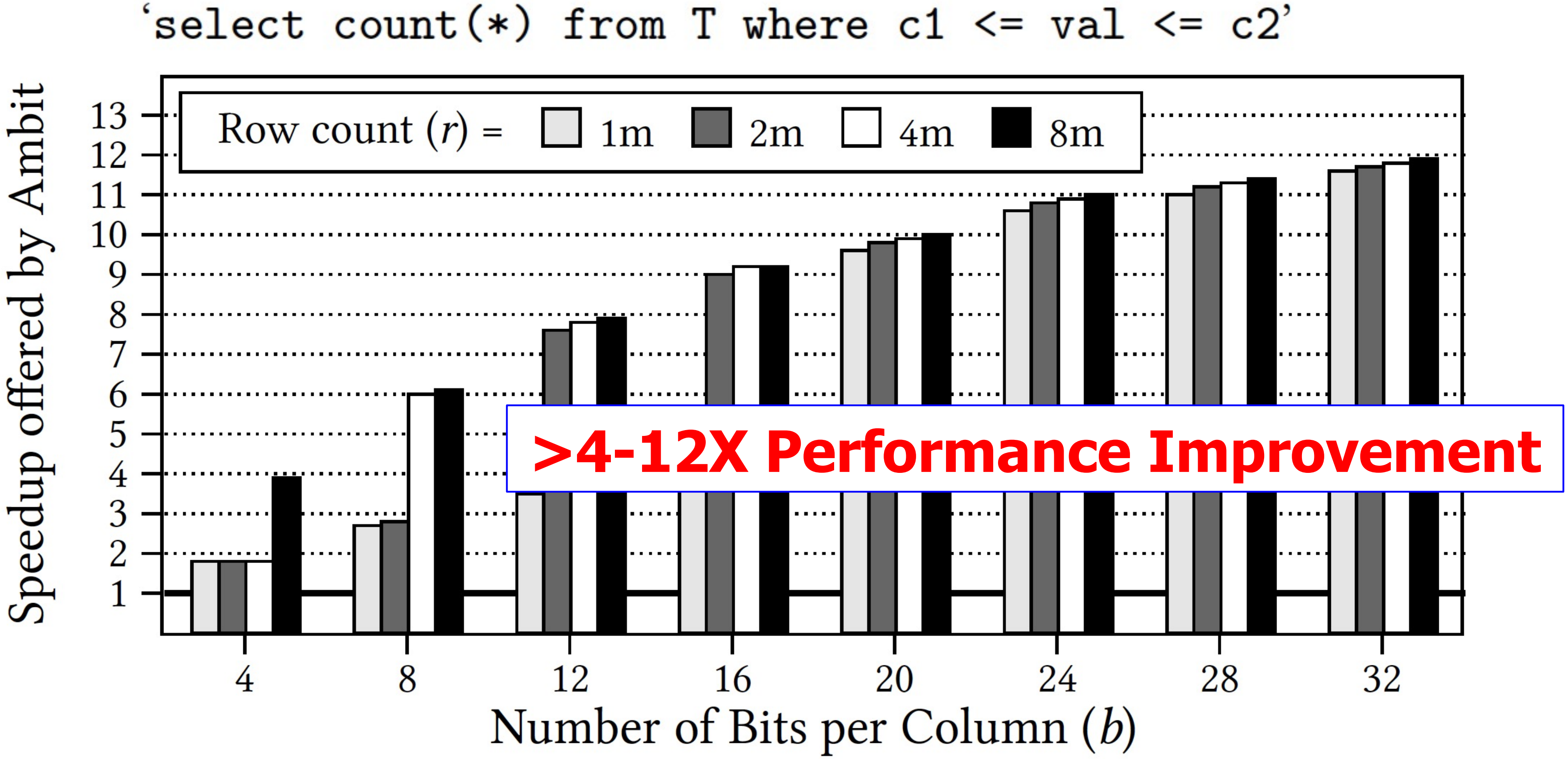


Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.

>5.4-6.6X Performance Improvement

Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

Performance: BitWeaving on Ambit



>4-12X Performance Improvement

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

Seshadri+, "Ambit: In-Memory Accelerator for Bulk Bitwise Operations using Commodity DRAM Technology," MICRO 2017.

In-DRAM Bulk Bitwise AND/OR

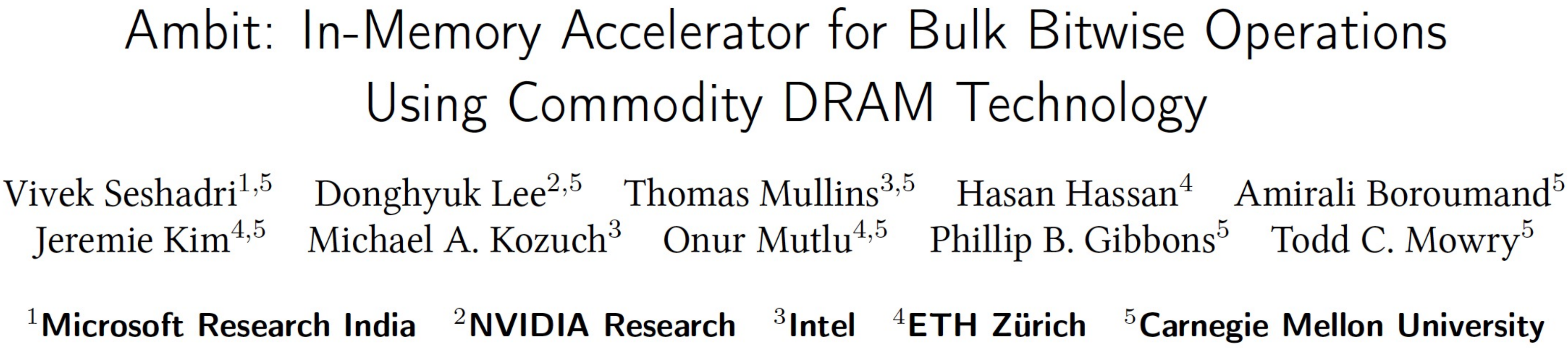
- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
"Fast Bulk Bitwise AND and OR in DRAM"
IEEE Computer Architecture Letters (***CAL***), April 2015.

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

Ambit: Bulk-Bitwise in-DRAM Computation

- Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
  "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology"
  Proceedings of the 50th International Symposium on Microarchitecture (***MICRO***), Boston, MA, USA, October 2017.
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]



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

Vivek Seshadri[1,5] Donghyuk Lee[2,5] Thomas Mullins[3,5] Hasan Hassan[4] Amirali Boroumand[5]
Jeremie Kim[4,5] Michael A. Kozuch[3] Onur Mutlu[4,5] Phillip B. Gibbons[5] Todd C. Mowry[5]

[1]Microsoft Research India [2]NVIDIA Research [3]Intel [4]ETH Zürich [5]Carnegie Mellon University

In-DRAM Bulk Bitwise Execution Paradigm

- Vivek Seshadri and Onur Mutlu,
"In-DRAM Bulk Bitwise Execution Engine"
Invited Book Chapter in Advances in Computers, to appear in 2020.
[Preliminary arXiv version]

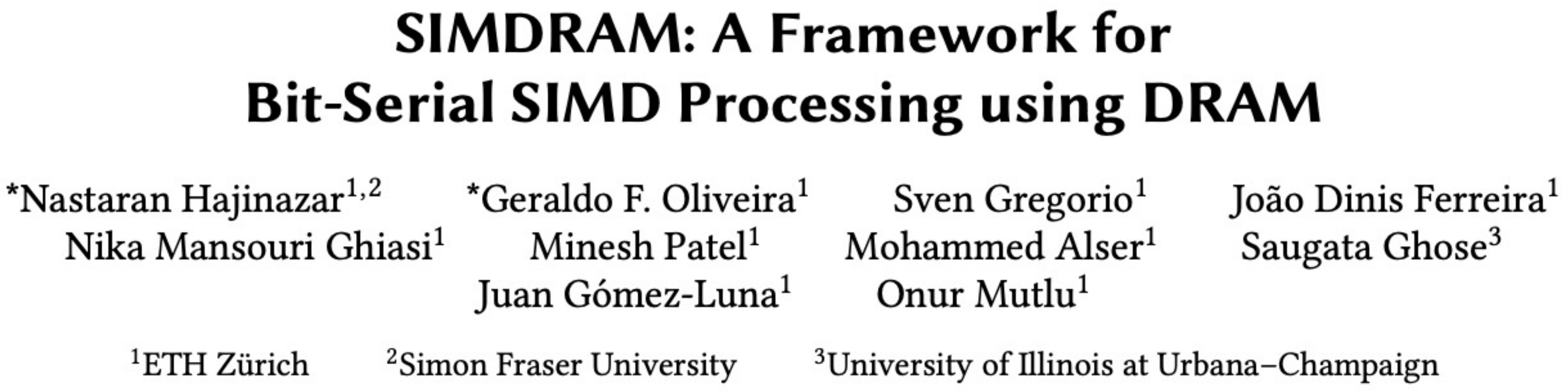
In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

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

SIMDRAM Framework

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu, **"SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"** *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Virtual, March-April 2021.
  [2-page Extended Abstract]
  [Short Talk Slides (pptx) (pdf)]
  [Talk Slides (pptx) (pdf)]
  [Short Talk Video (5 mins)]
  [Full Talk Video (27 mins)]



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

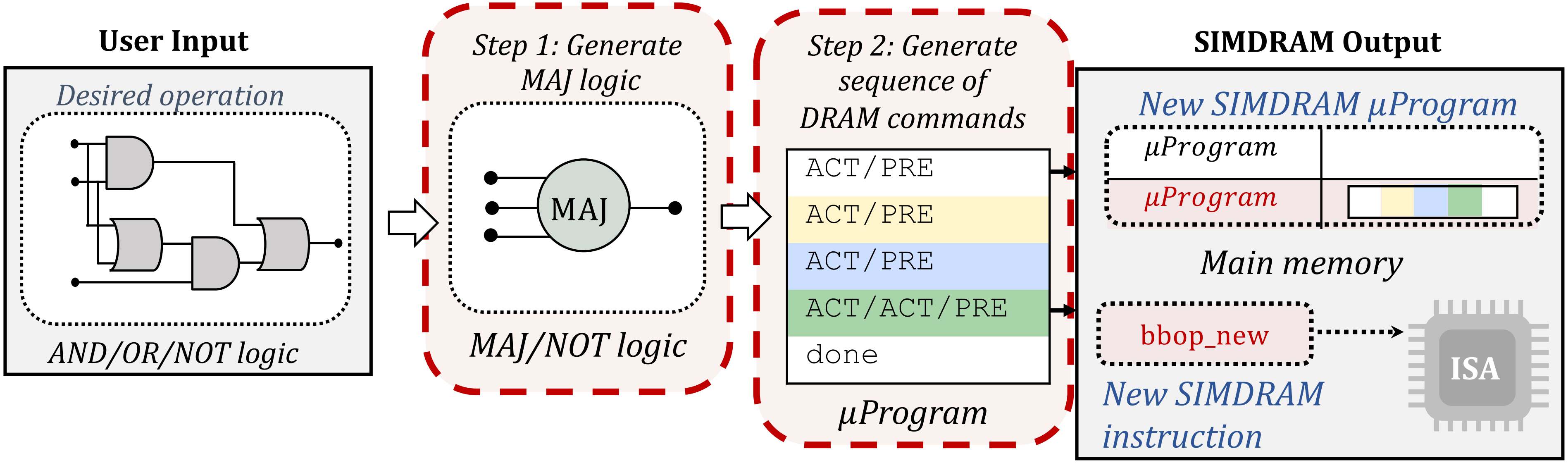
*Nastaran Hajinazar[1,2] *Geraldo F. Oliveira[1] Sven Gregorio[1] João Dinis Ferreira[1]
Nika Mansouri Ghiasi[1] Minesh Patel[1] Mohammed Alser[1] Saugata Ghose[3]
Juan Gómez-Luna[1] Onur Mutlu[1]

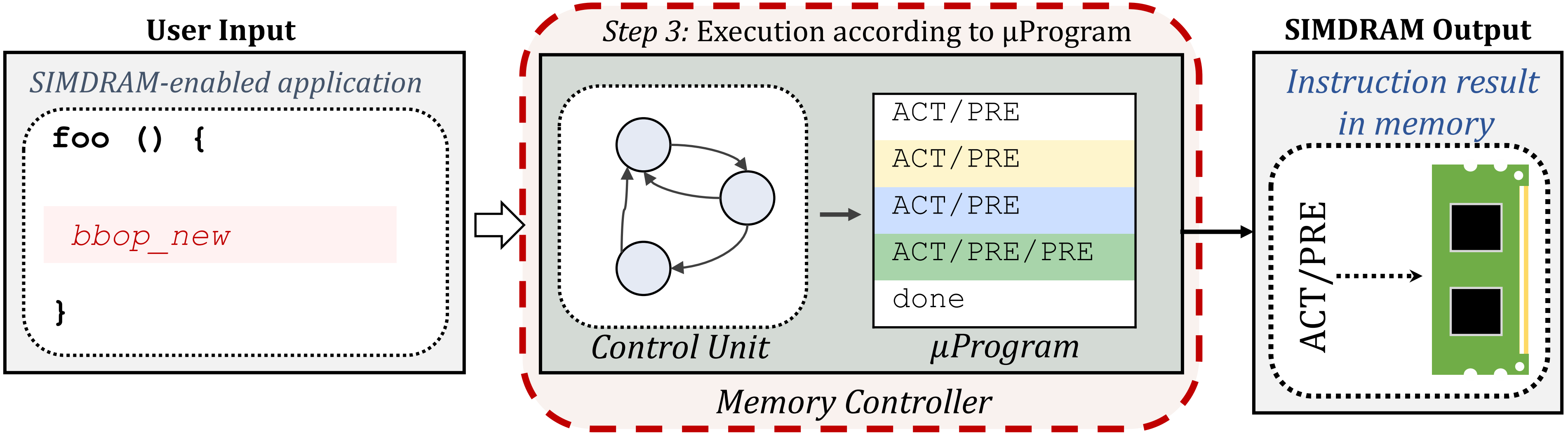
[1]ETH Zürich [2]Simon Fraser University [3]University of Illinois at Urbana–Champaign

SIMDRAM Key Idea

- **SIMDRAM:** An end-to-end processing-using-DRAM framework that provides the programming interface, the ISA, and the hardware support for:
  - **Efficiently** computing **complex** operations in DRAM
  - Providing the ability to implement **arbitrary** operations as required
  - Using an **in-DRAM massively-parallel SIMD substrate** that requires **minimal** changes to DRAM architecture

SIMDRAM Framework: Overview





```
foo () {

  bbop_new

}
```

SIMDRAM Key Results

Evaluated on:

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

SIMDRAM provides:

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

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

- **21×** and **2.1×** the **performance** of a **CPU** an a **high-end GPU**, over **seven real-world applications**

SIMDRAM Conclusion

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

SIMDRAM is a promising PuM framework

- Can **ease the adoption** of processing-using-DRAM architectures
- Improves the **performance** and **efficiency** of processing-using-memory architectures

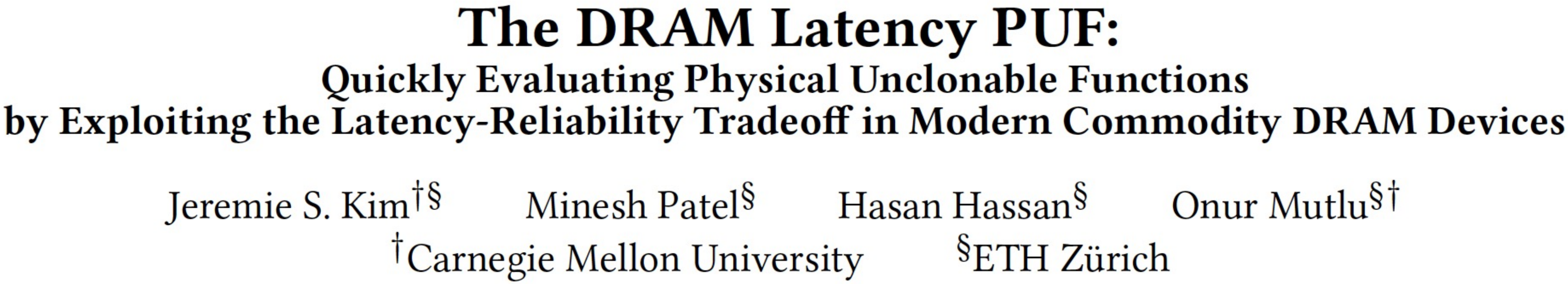
SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

Nastaran Hajinazar* Geraldo F. Oliveira*
Sven Gregorio Joao Ferreira Nika Mansouri Ghiasi
Minesh Patel Mohammed Alser Saugata Ghose
Juan Gómez–Luna Onur Mutlu

SAFARI
ETH Zürich
SFU SIMON FRASER UNIVERSITY
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

In-DRAM Physical Unclonable Functions

- 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"
*Proceedings of the 24th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Vienna, Austria, February 2018.
[Lightning Talk Video]
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]
[Full Talk Lecture Video (28 minutes)]

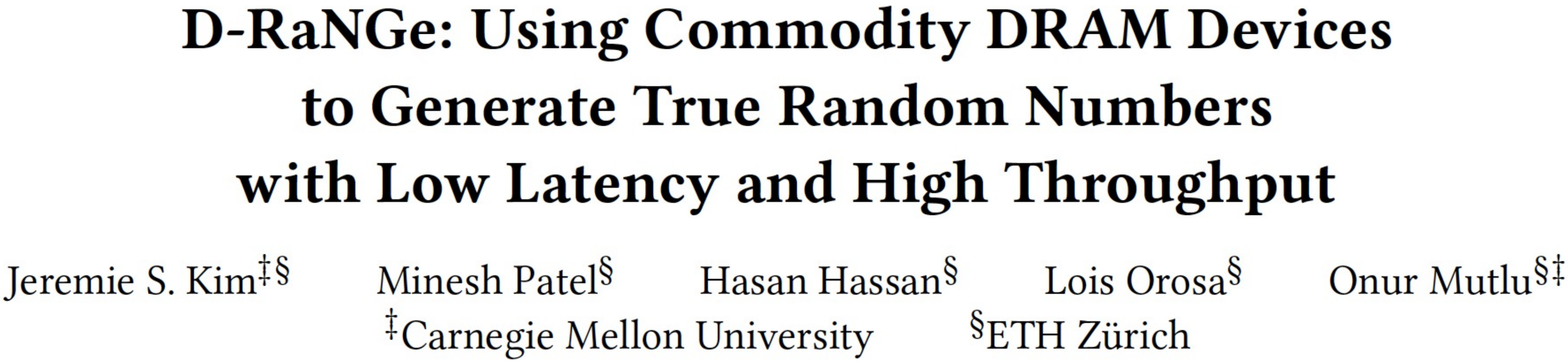


The DRAM Latency PUF:
Quickly Evaluating Physical Unclonable Functions
by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

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

In-DRAM True Random Number Generation

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, and Onur Mutlu,
"D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput"
*Proceedings of the 25th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Washington, DC, USA, February 2019.
[Slides (pptx) (pdf)]
[Full Talk Video (21 minutes)]
[Full Talk Lecture Video (27 minutes)]
Top Picks Honorable Mention by IEEE Micro.

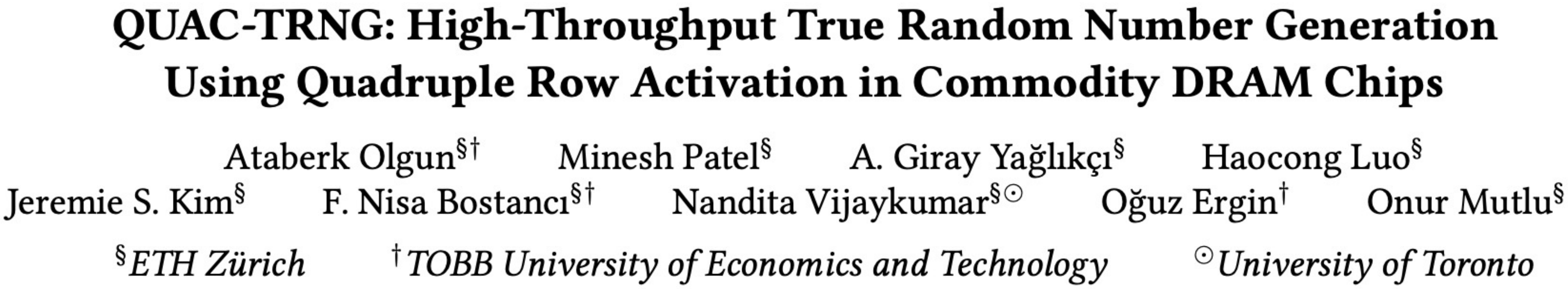


D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

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

In-DRAM True Random Number Generation

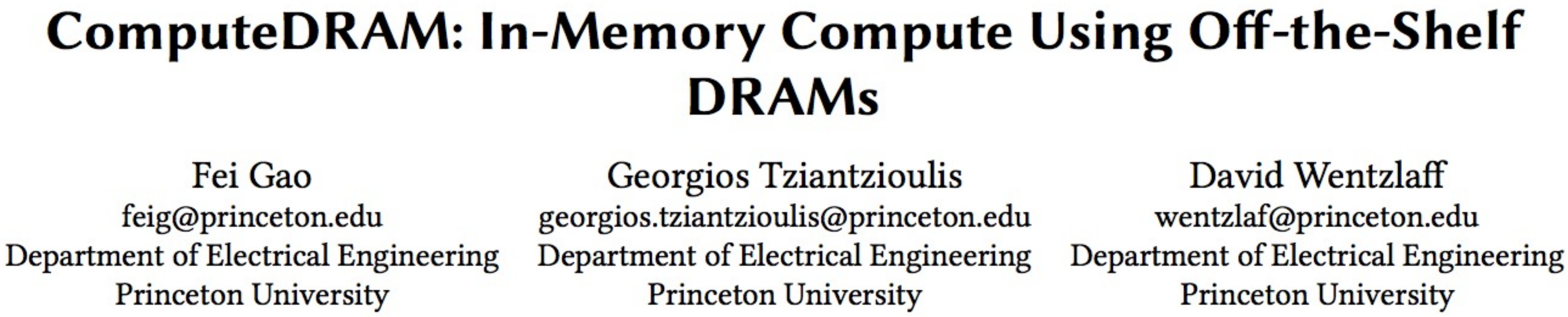
- Ataberk Olgun, Minesh Patel, A. Giray Yaglikci, Haocong Luo, Jeremie S. Kim, F. Nisa Bostanci, Nandita Vijaykumar, Oguz Ergin, and Onur Mutlu,
  "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips"
  Proceedings of the 48th International Symposium on Computer Architecture (***ISCA***), Virtual, June 2021.
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [Talk Video (25 minutes)]
  [SAFARI Live Seminar Video (1 hr 26 mins)]



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

Ataberk Olgun[§†] Minesh Patel[§] A. Giray Yağlıkçı[§] Haocong Luo[§]
Jeremie S. Kim[§] F. Nisa Bostancı[§†] Nandita Vijaykumar[§⊙] Oğuz Ergin[†] Onur Mutlu[§]
[§]*ETH Zürich* [†]*TOBB University of Economics and Technology* [⊙]*University of Toronto*

RowClone & Bitwise Ops in Real DRAM Chips



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

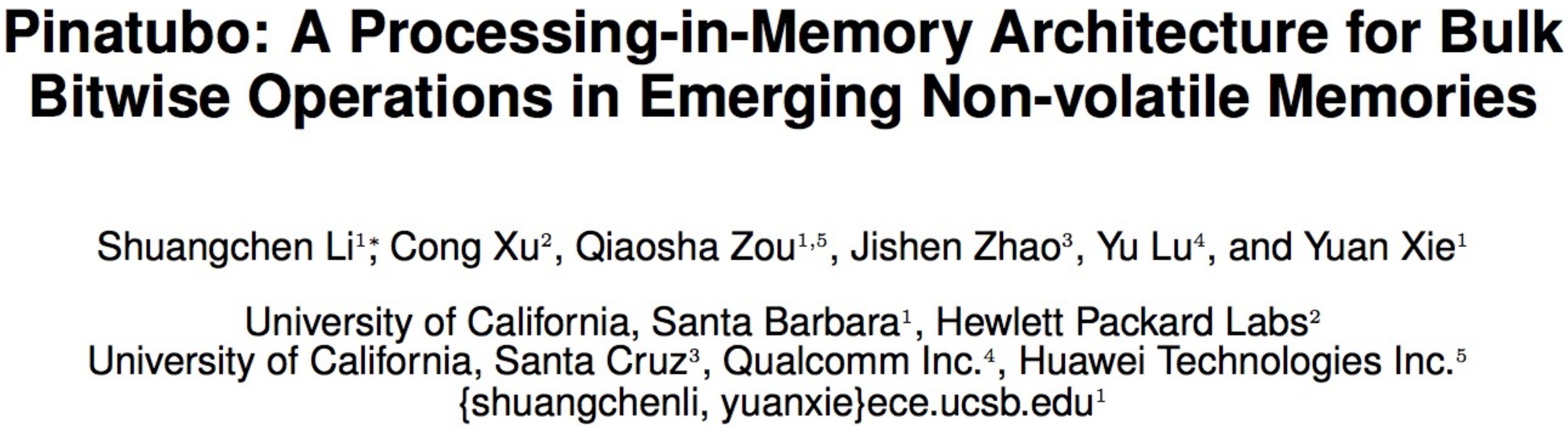
Fei Gao
feig@princeton.edu
Department of Electrical Engineering
Princeton University

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

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

https://parallel.princeton.edu/papers/micro19-gao.pdf

Pinatubo: RowClone and Bitwise Ops in PCM



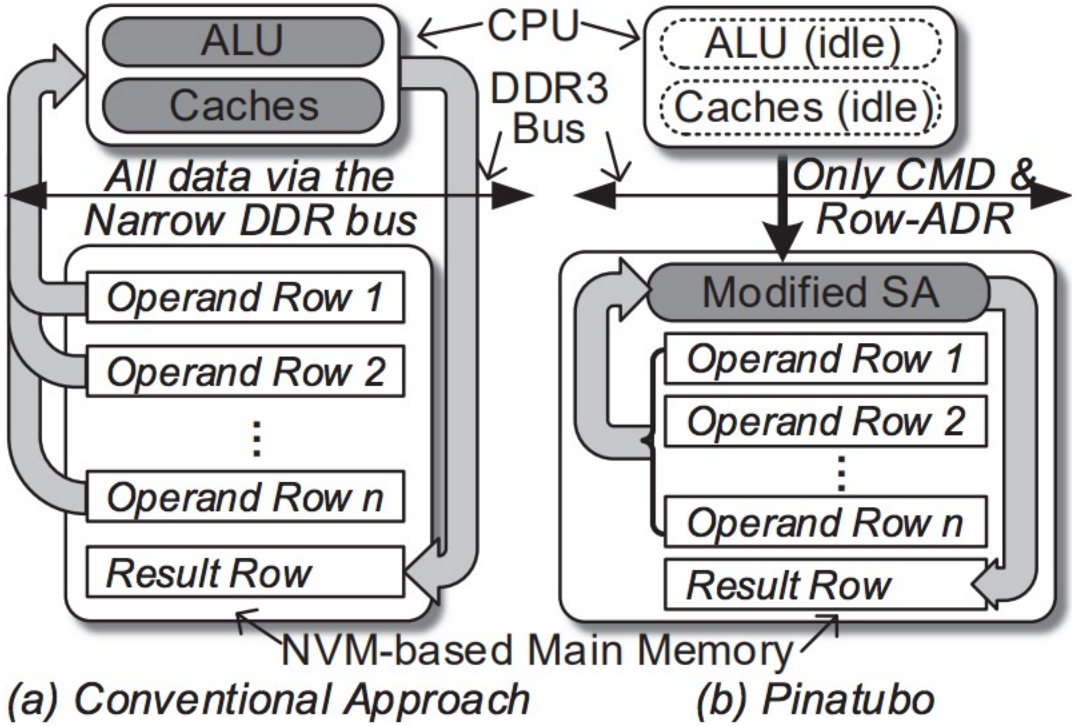
Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories

Shuangchen Li[1]*, Cong Xu[2], Qiaosha Zou[1,5], Jishen Zhao[3], Yu Lu[4], and Yuan Xie[1]

University of California, Santa Barbara[1], Hewlett Packard Labs[2]
University of California, Santa Cruz[3], Qualcomm Inc.[4], Huawei Technologies Inc.[5]
{shuangchenli, yuanxie}ece.ucsb.edu[1]

https://cseweb.ucsd.edu/~jzhao/files/Pinatubo-dac2016.pdf

Pinatubo: RowClone and Bitwise Ops in PCM



(a) Conventional Approach (b) Pinatubo

Figure 2: Overview: (a) Computing-centric approach, moving tons of data to CPU and write back. (b) The proposed Pinatubo architecture, performs n-row bitwise operations inside NVM in one step.

https://cseweb.ucsd.edu/~jzhao/files/Pinatubo-dac2016.pdf

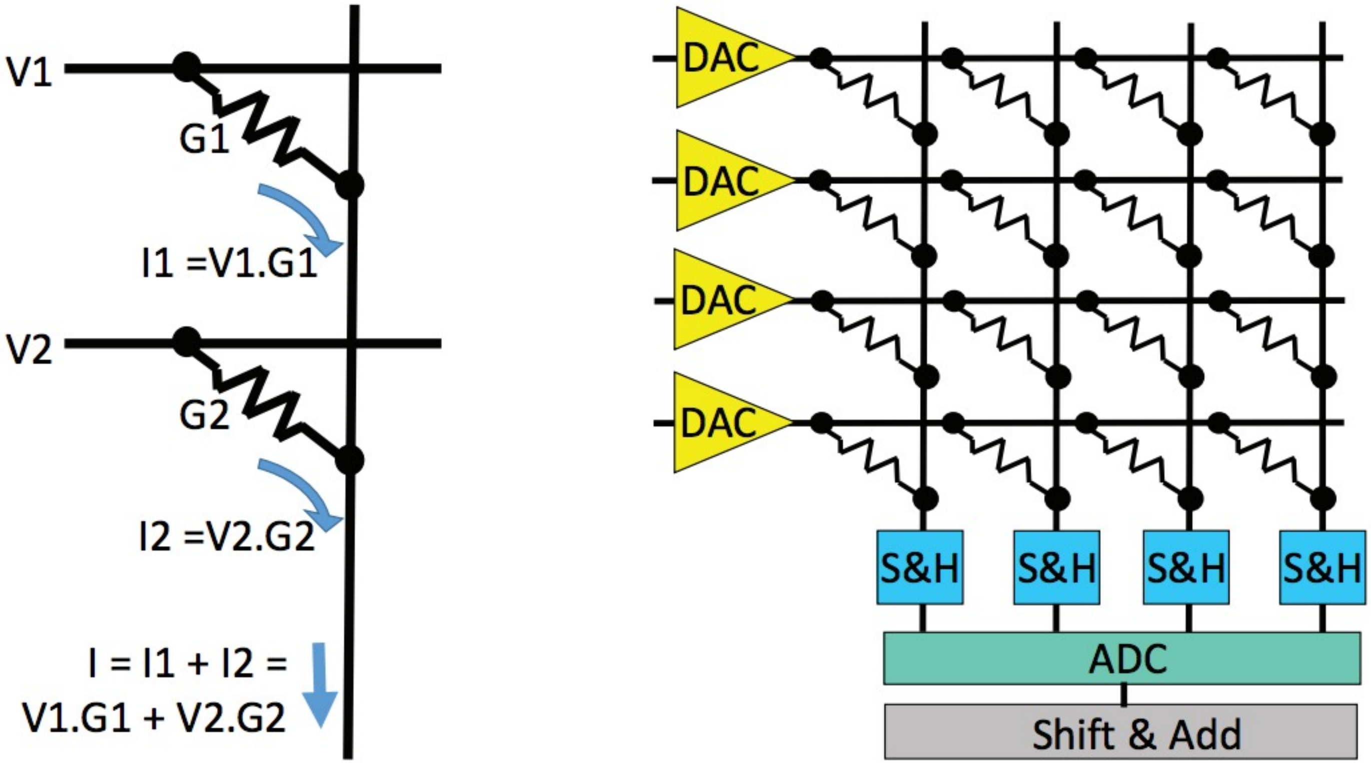
In-Memory Crossbar Array Operations

- Some emerging NVM technologies have crossbar array structure
  - Memristors, resistive RAM, phase change mem, STT-MRAM, …

- Crossbar arrays can be used to perform dot product operations using "analog computation capability"
  - Can operate on multiple pieces of data using Kirchoff's laws
    - Bitline current is a sum of products of wordline V x (1 / cell R)
  - Computation is in analog domain inside the crossbar array

- Need peripheral circuitry for D→A and A→D conversion of inputs and outputs

In-Memory Crossbar Computation



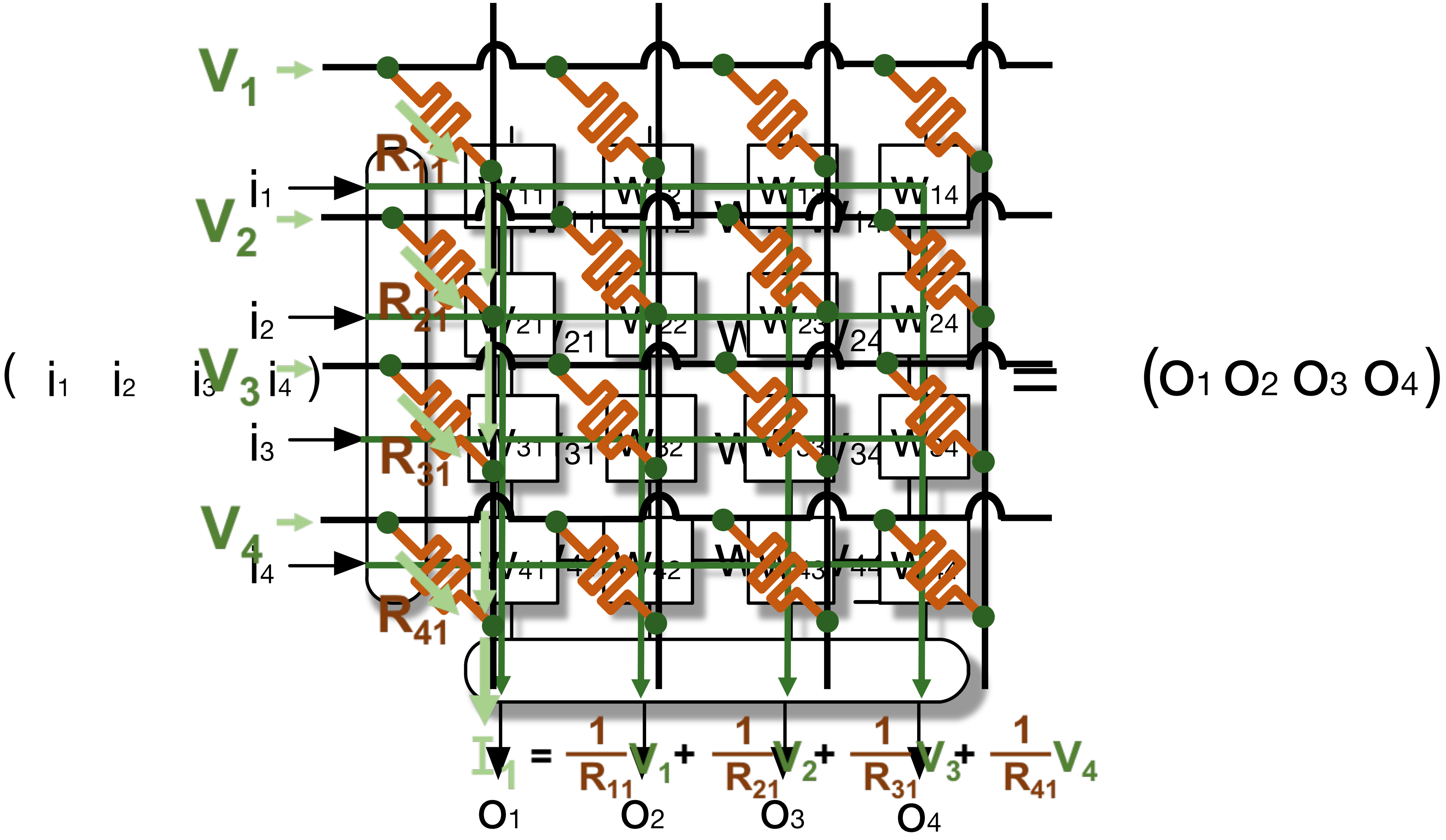
(a) Multiply-Accumulate operation

(b) Vector-Matrix Multiplier

Fig. 1. (a) Using a bitline to perform an analog sum of products operation. (b) A memristor crossbar used as a vector-matrix multiplier.

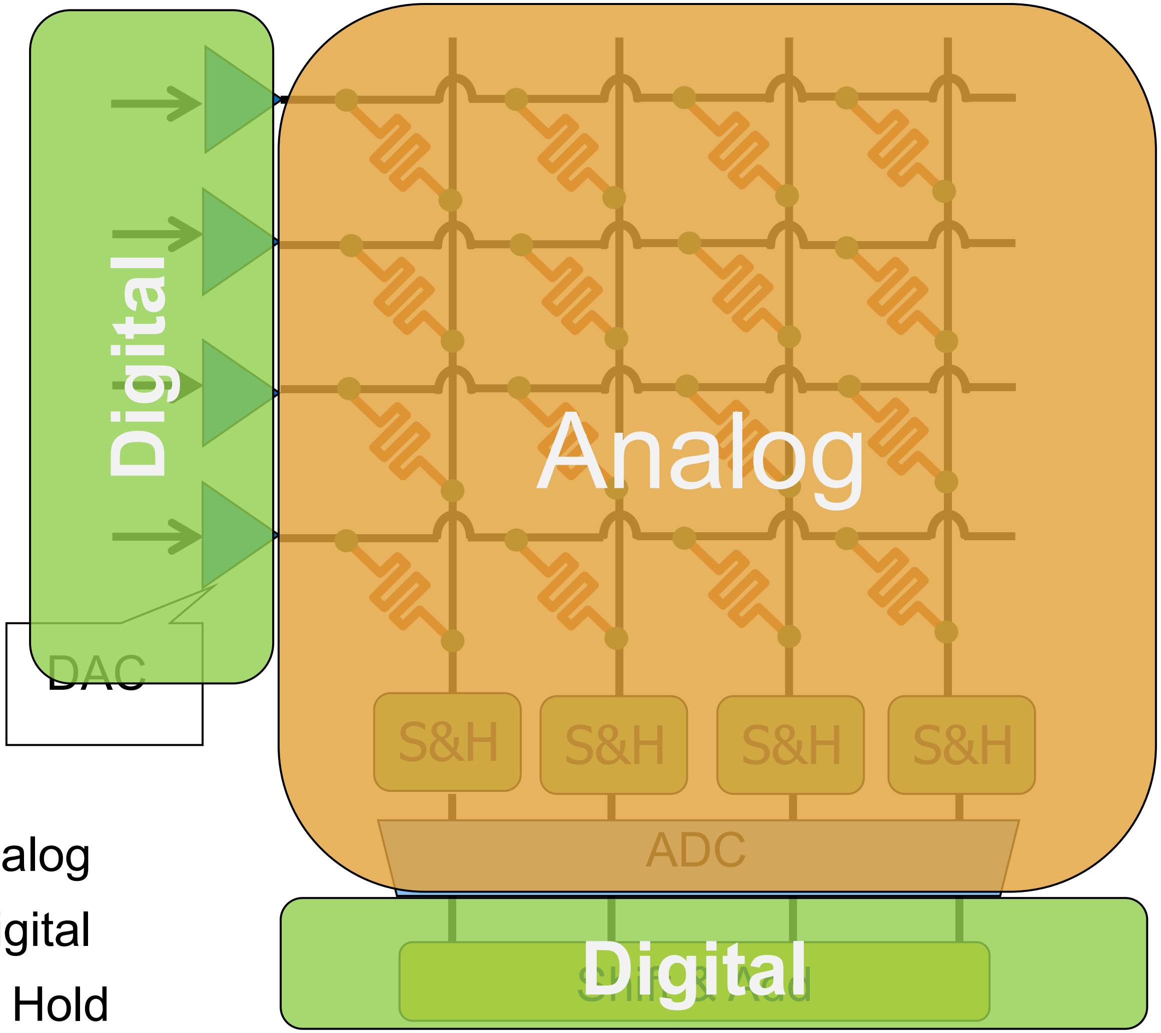
Shafiee+, “ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars”, ISCA 2016.

In-Memory Crossbar Computation



$$I_1 = \frac{1}{R_{11}}V_1 + \frac{1}{R_{21}}V_2 + \frac{1}{R_{31}}V_3 + \frac{1}{R_{41}}V_4$$

Required Peripheral Circuitry



DAC: Digital to Analog

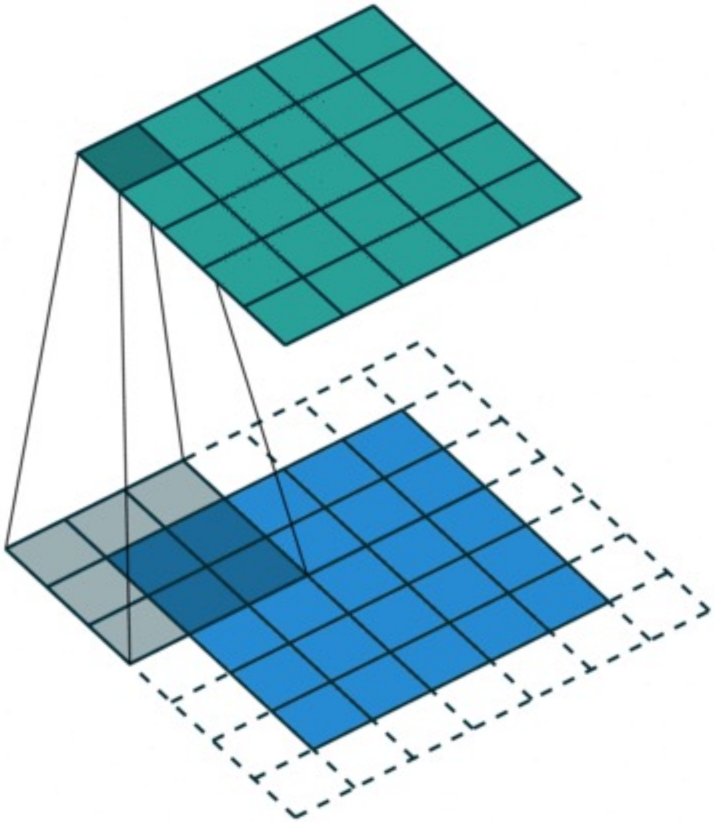
ADC: Analog to Digital

S&H: Sample and Hold

Shift and add: used to summarize the final output

An Example of 2D Convolution

Output feature map



Input feature map

Structure information

Input: 5*5 (blue)

Kernel (filter): 3*3 (grey)

Output: 5*5 (green)

Computation information

Stride: 1

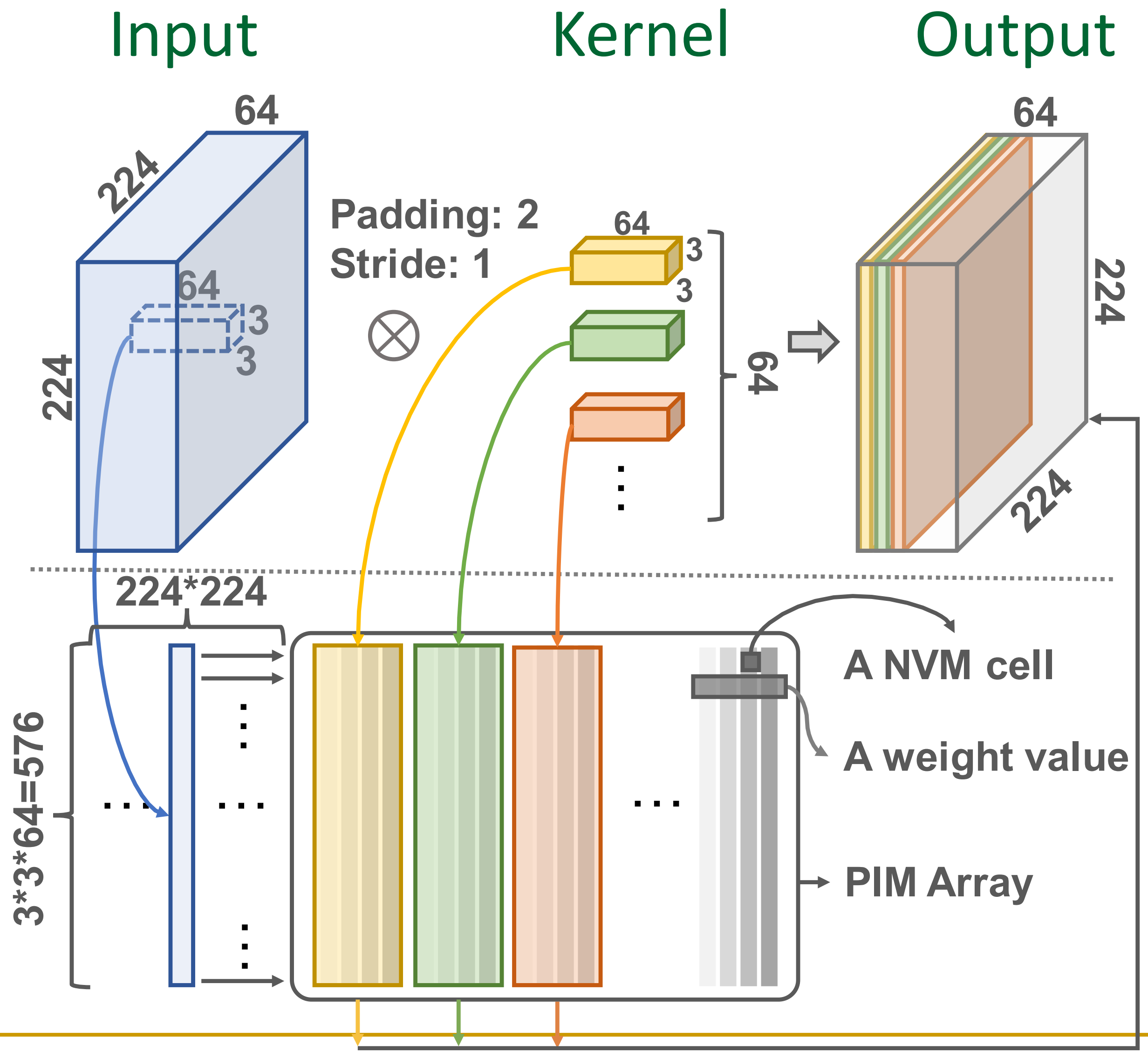
Padding: 1 (white)

Output Dim = (Input + 2*Padding - Kernel) / Stride + 1

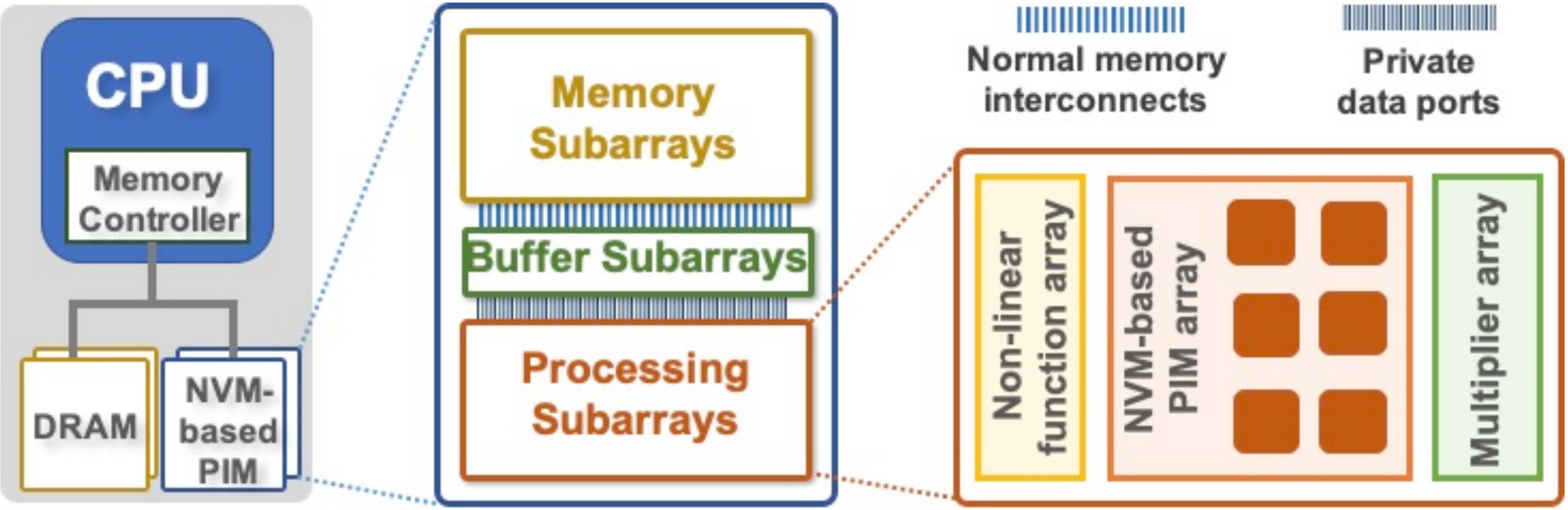
Mapping Computation onto the Crossbar

A convolution operation in neural network application

An NVM-based PIM array



An Overview of NVM-Based PIM System



NVM-based PIM array:

core processing unit for vector-matrix multiplication

Non-linear function array:

processing unit for non-linear functions (e.g., ReLU operations in neural networks)

Multiplier array:

handles element-wise operations

Example Readings on NVM-Based PIM

- Shafiee+, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars", ISCA 2016.

- Chi+, "PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory", ISCA 2016.

- Prezioso+, "Training and Operation of an Integrated Neuromorphic Network based on Metal-Oxide Memristors", Nature 2015

- Ambrogio+, "Equivalent-accuracy accelerated neural-network training using analogue memory", Nature 2018.

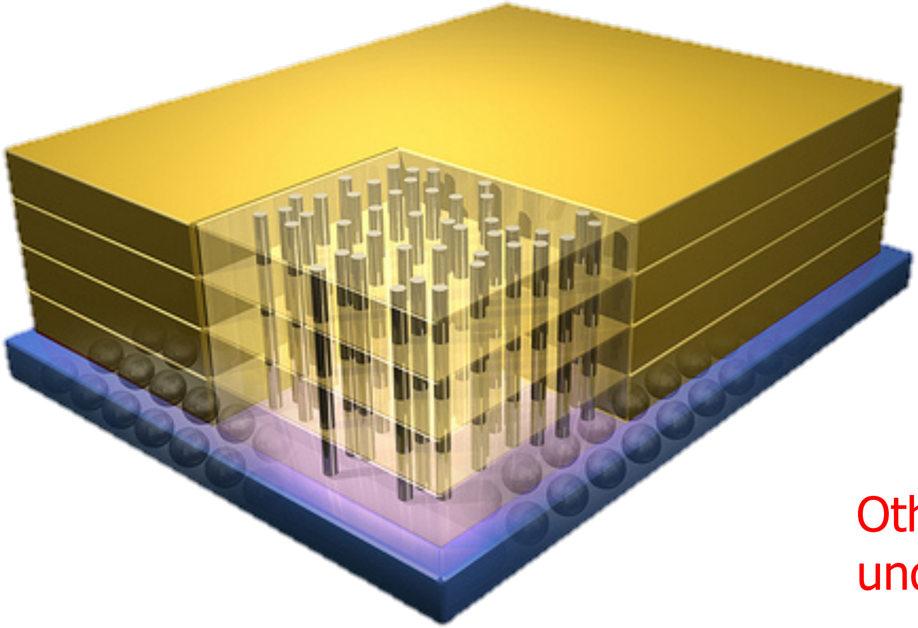
Processing in Memory: Two Approaches

1. Processing using Memory
2. Processing near Memory

Opportunity: 3D-Stacked Logic+Memory

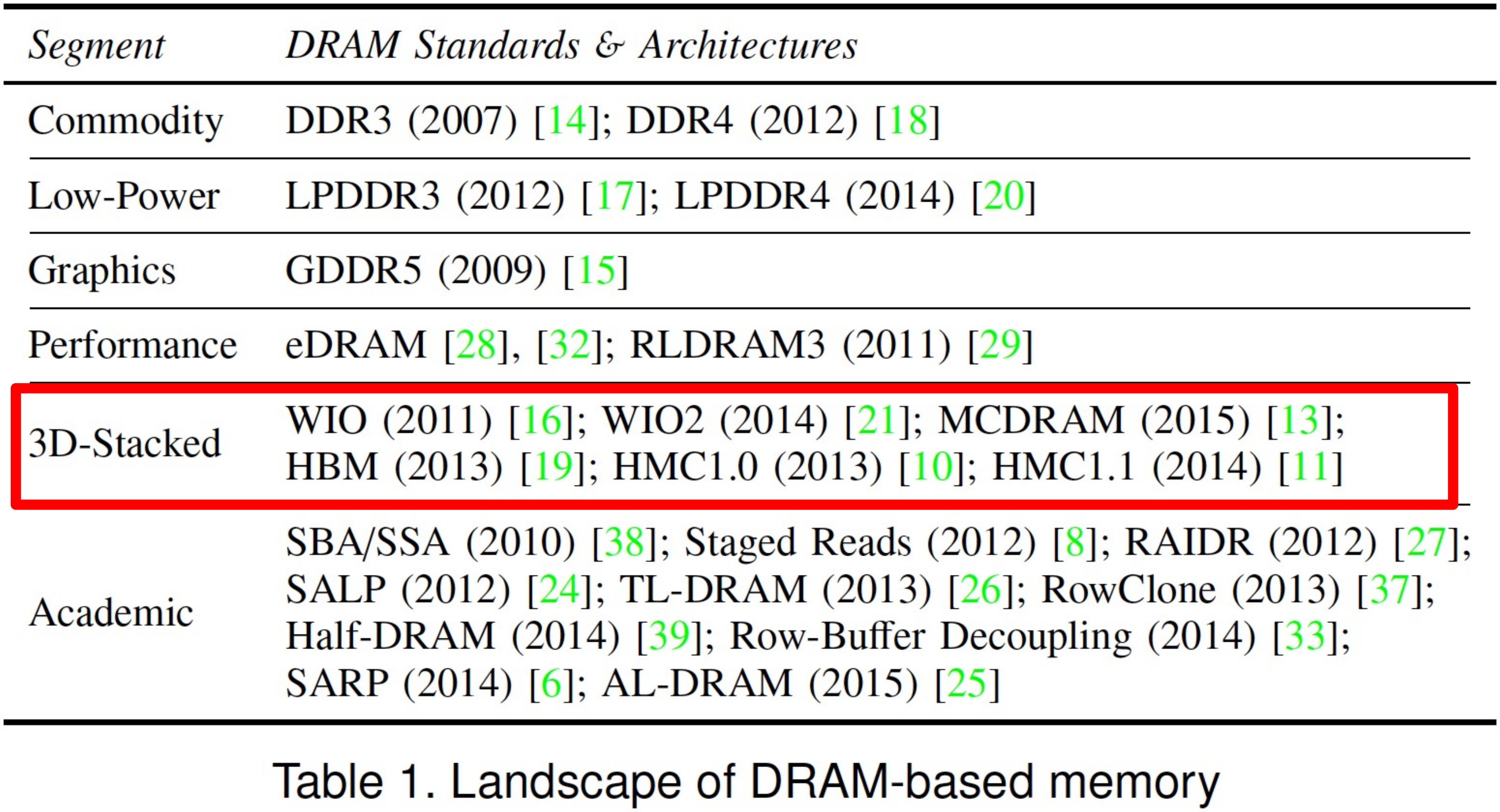
Hybrid Memory Cube

CONSORTIUM



Other “True 3D” technologies under development

DRAM Landscape (circa 2015)



Segment	*DRAM Standards & Architectures*
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLDRAM3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

Kim+, "Ramulator: A Flexible and Extensible DRAM Simulator", IEEE CAL 2015.

Two Key Questions in Processing Near Memory

- 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)

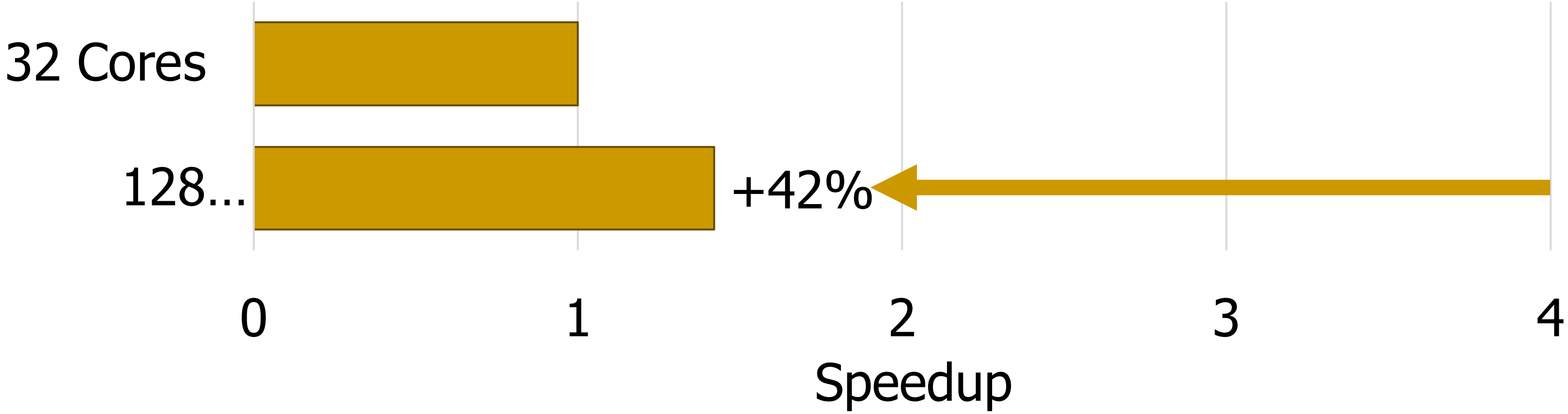
36 Million
Wikipedia Pages

1.4 Billion
Facebook Users

300 Million
Twitter Users

30 Billion
Instagram Photos

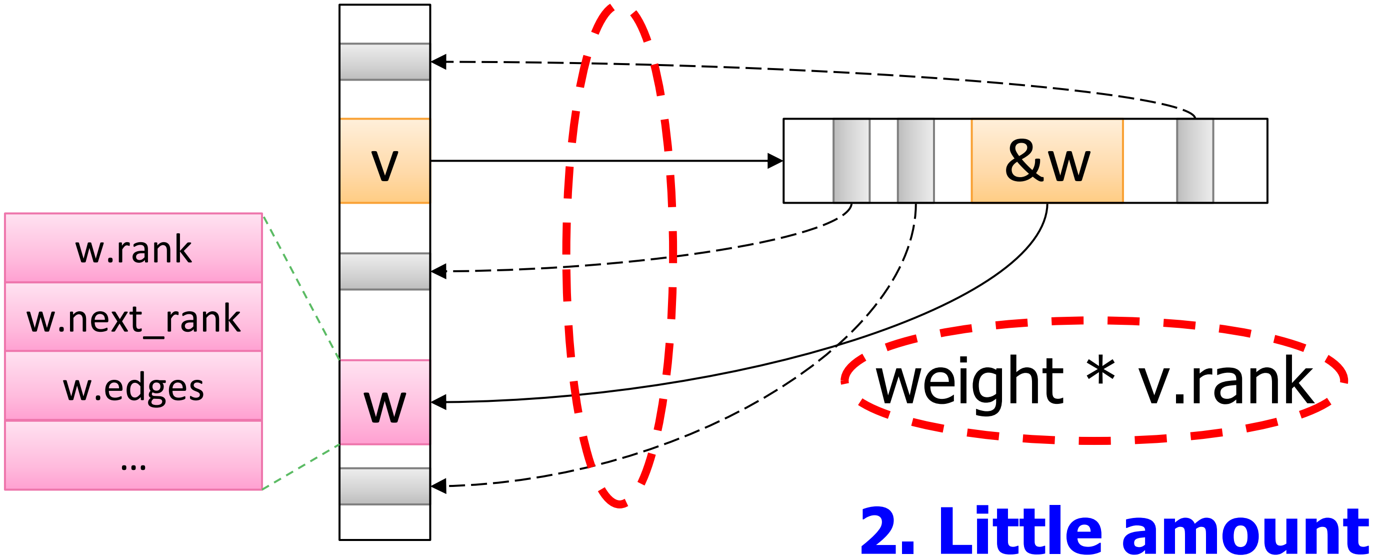
- Scalable large-scale graph processing is challenging



Key Bottlenecks in Graph Processing

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

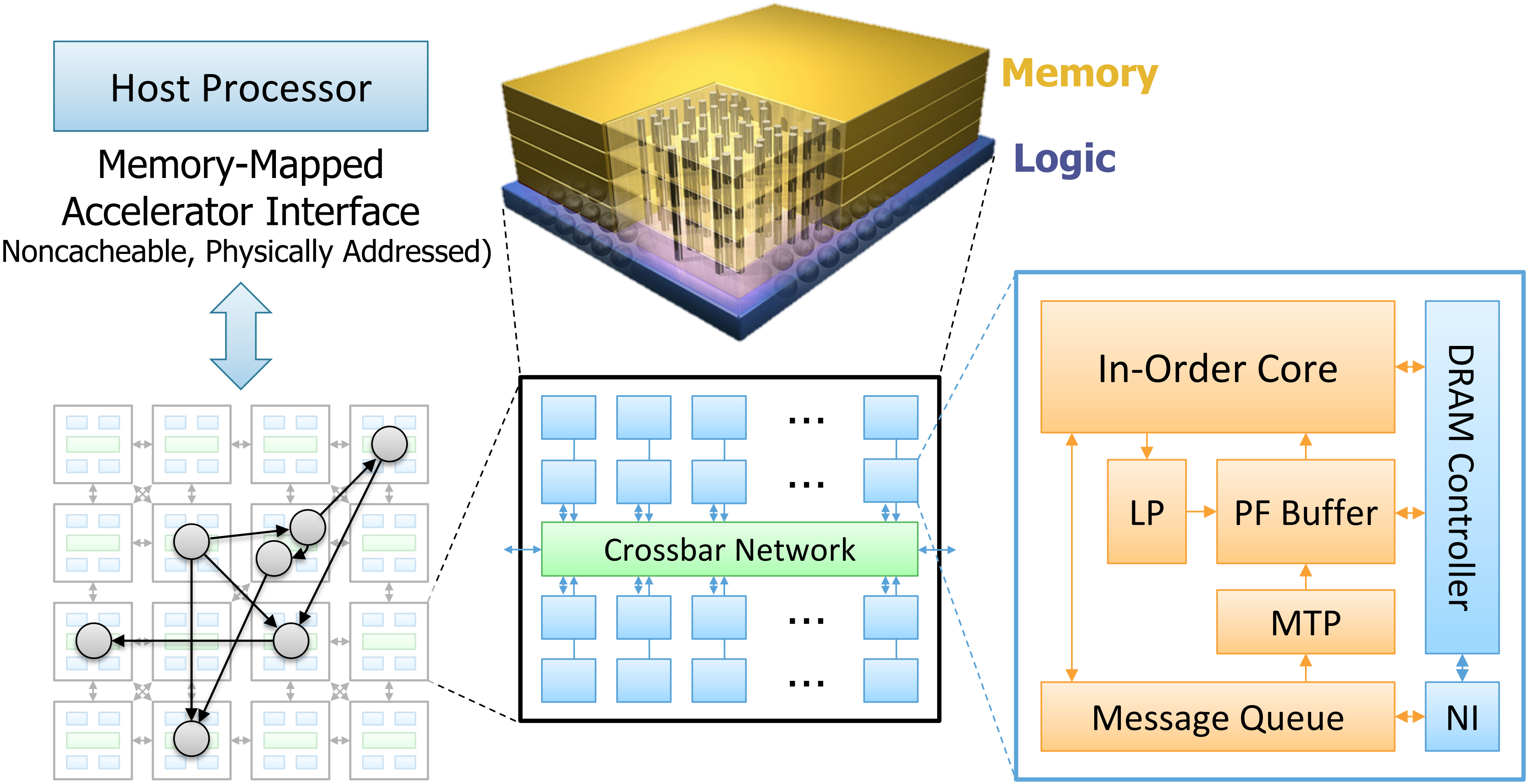
1. Frequent random memory accesses



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)

Memory

Logic

Crossbar Network

In-Order Core

LP

PF Buffer

MTP

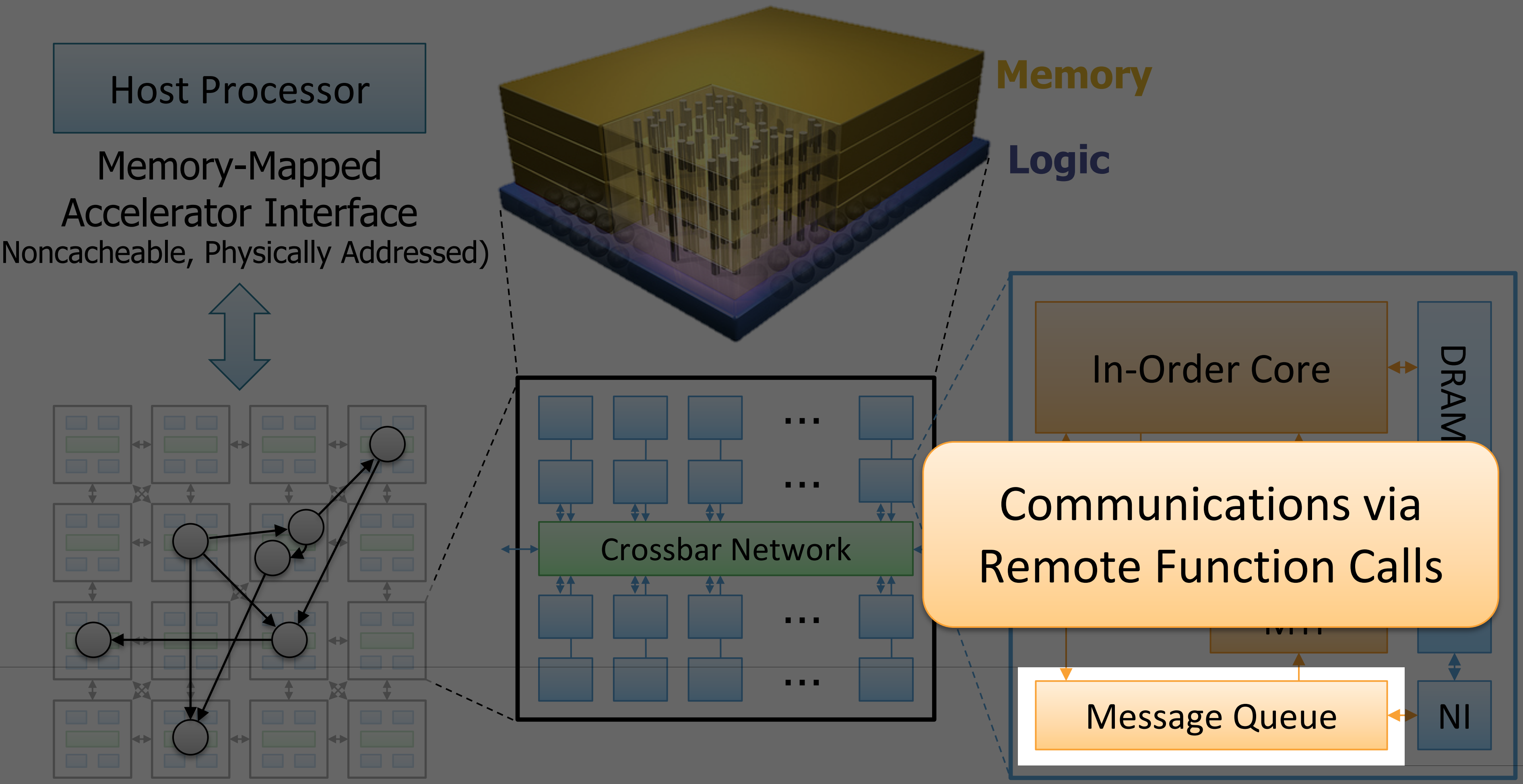
Message Queue

DRAM Controller

NI

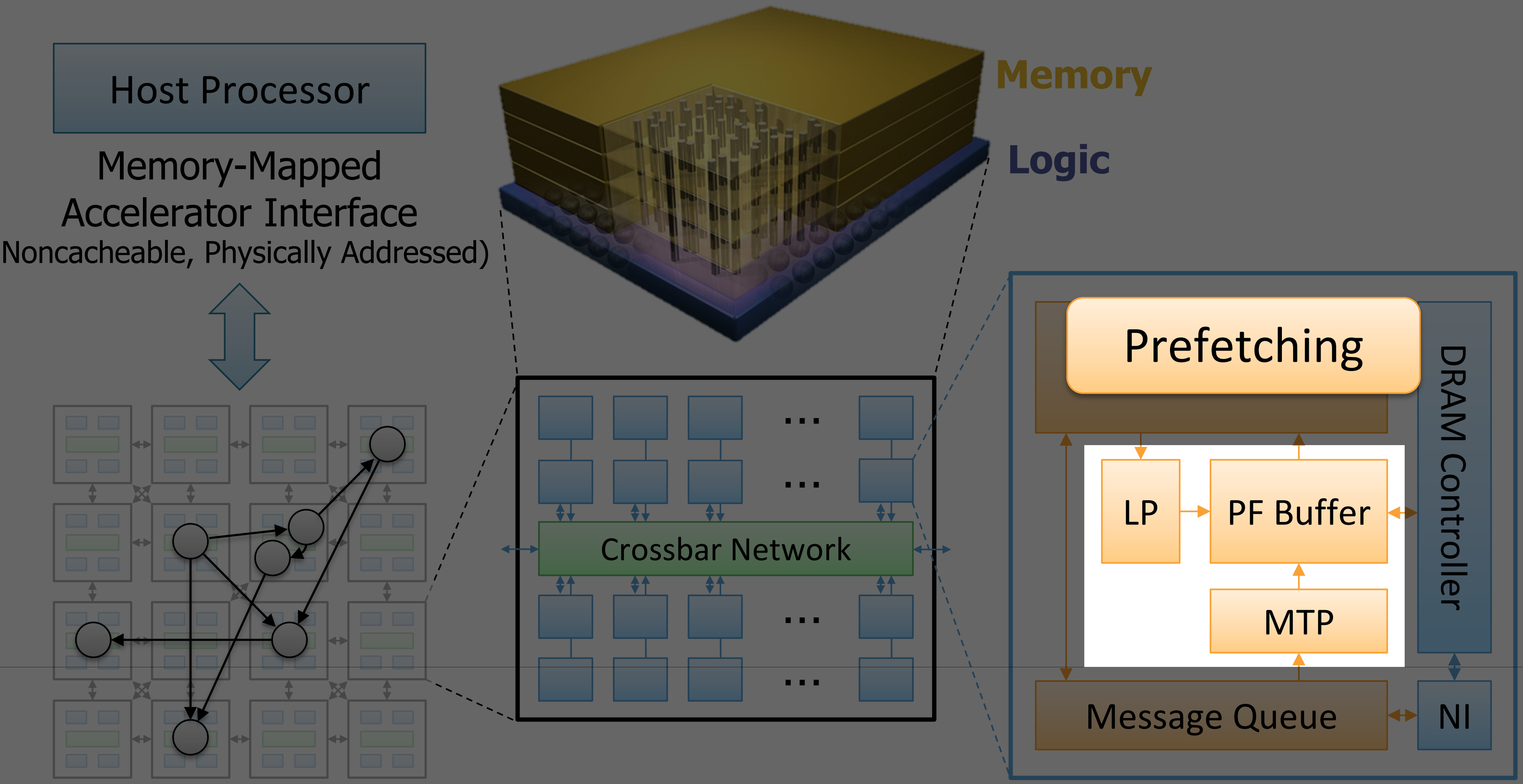
SAFARI Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

Tesseract System for Graph Processing



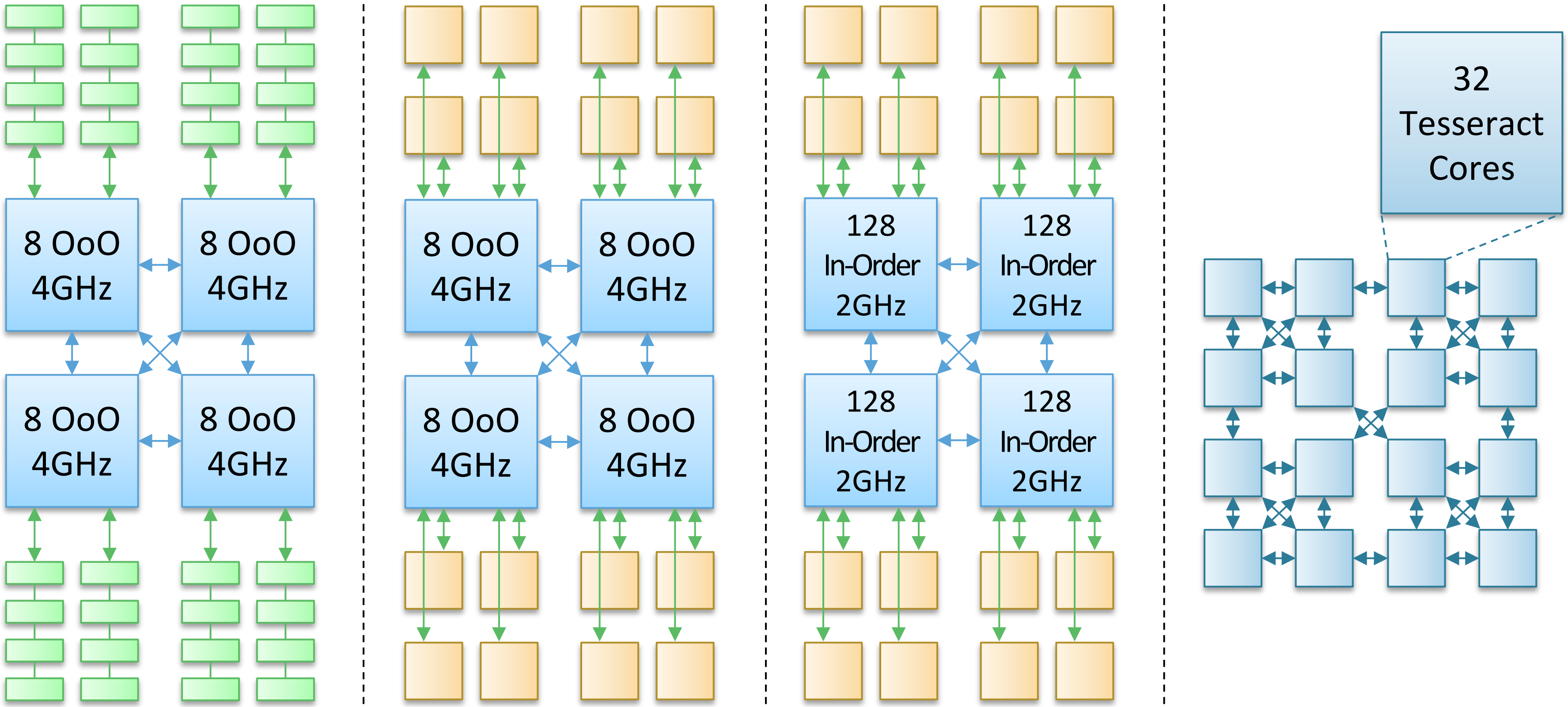
Communications via Remote Function Calls

Tesseract System for Graph Processing



Evaluated Systems

DDR3-OoO | HMC-OoO | HMC-MC | **Tesseract**

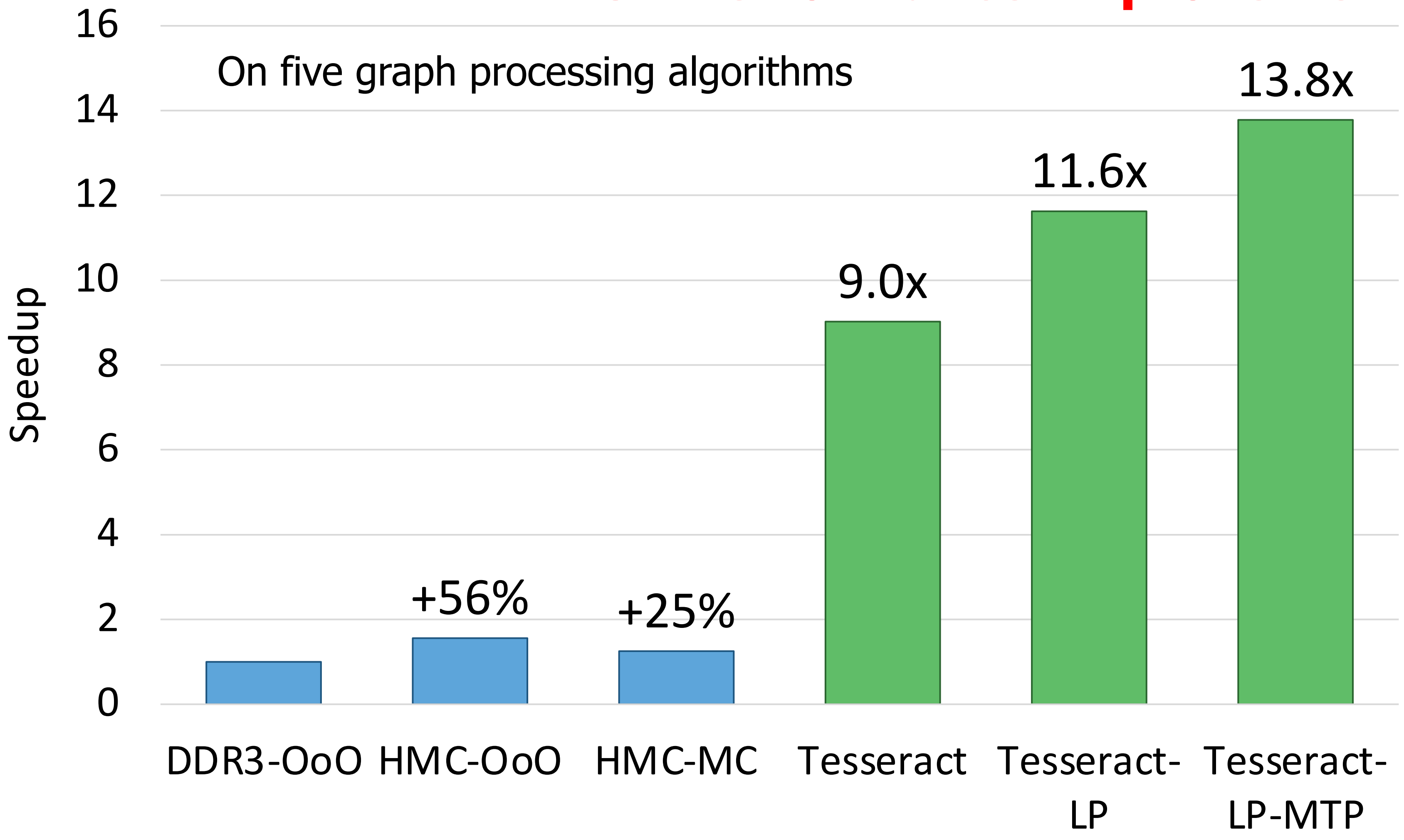


102.4GB/s | 640GB/s | 640GB/s | **8TB/s**

Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

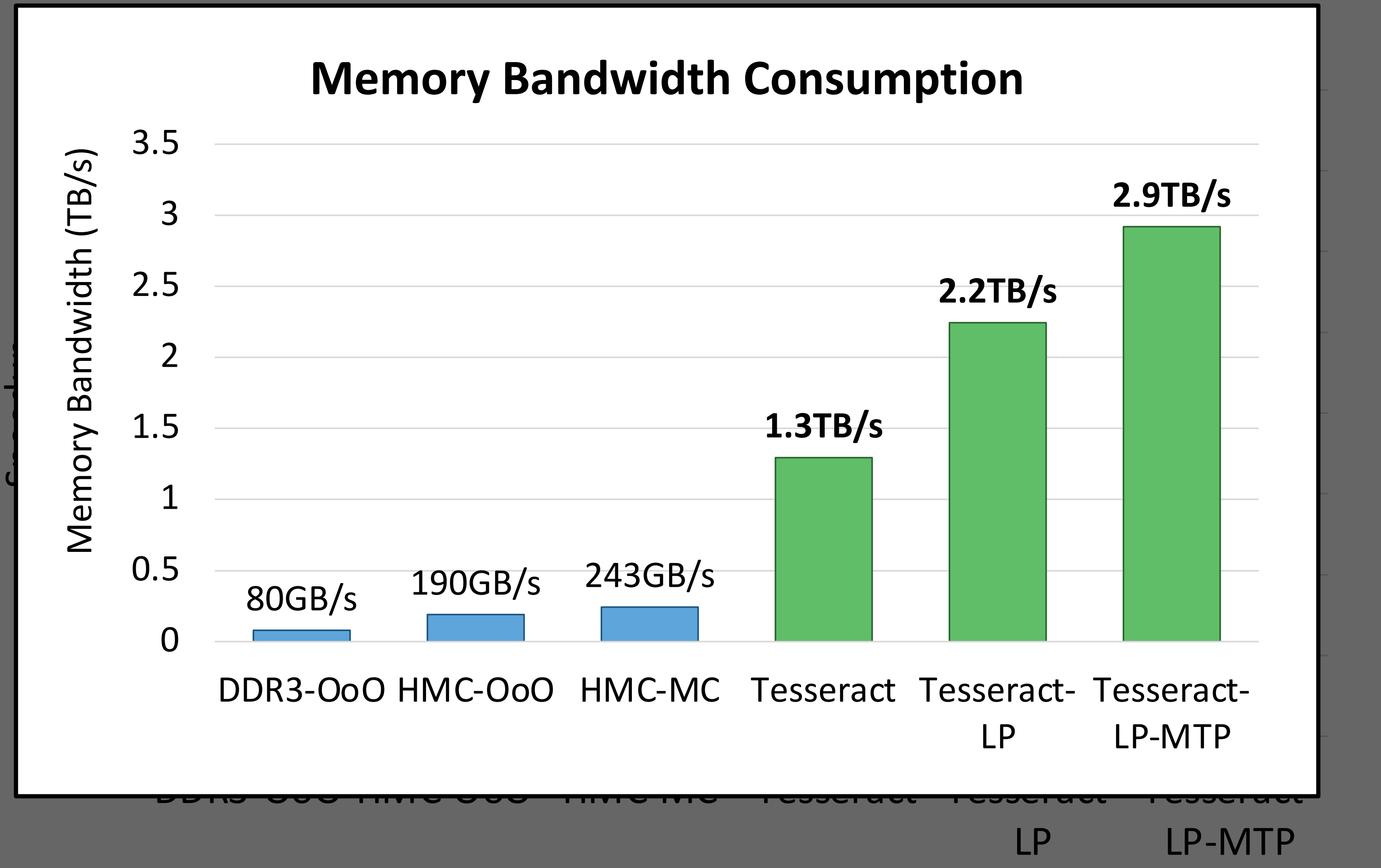
Tesseract Graph Processing Performance

>13X Performance Improvement

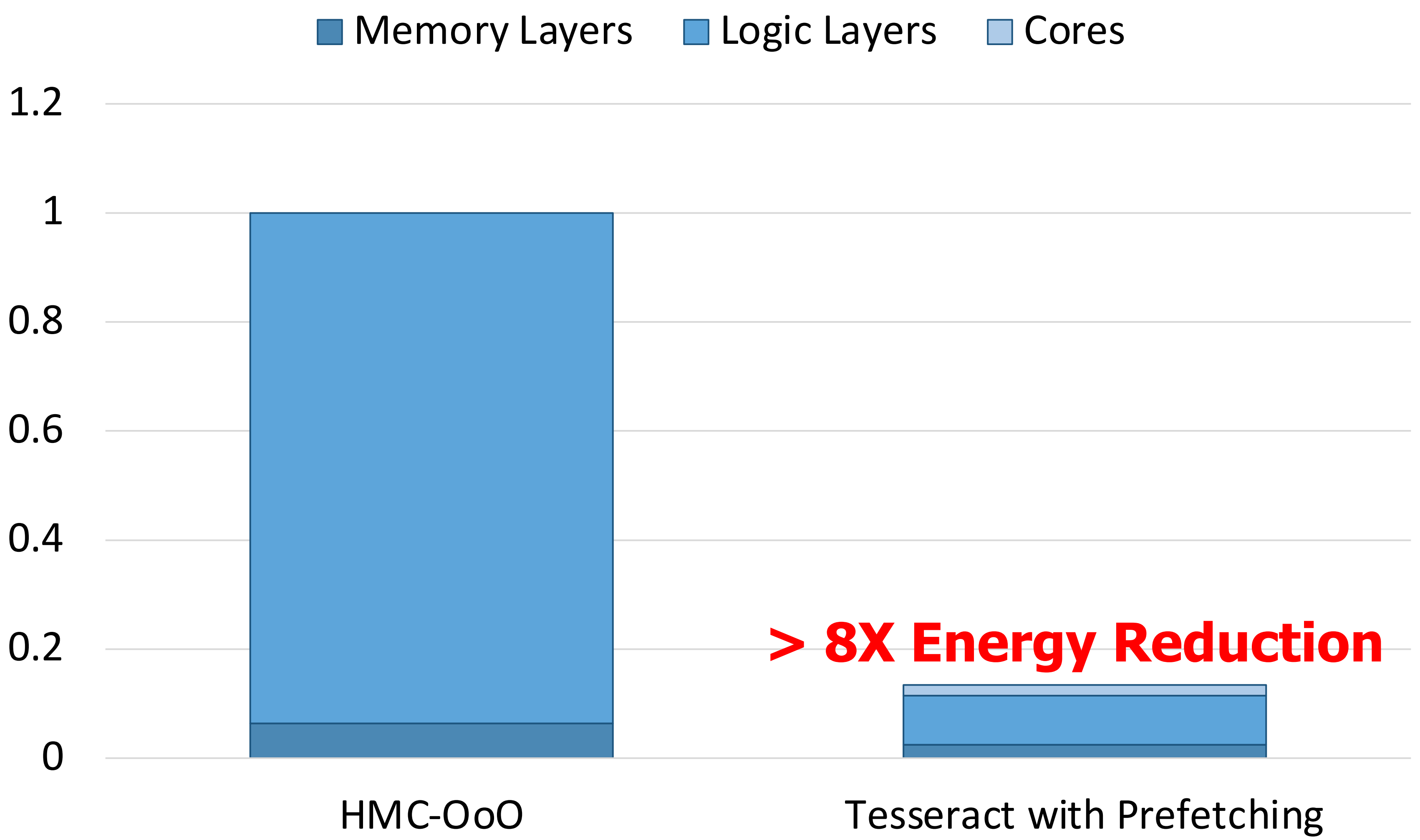


Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

Tesseract Graph Processing Performance



Tesseract Graph Processing System Energy

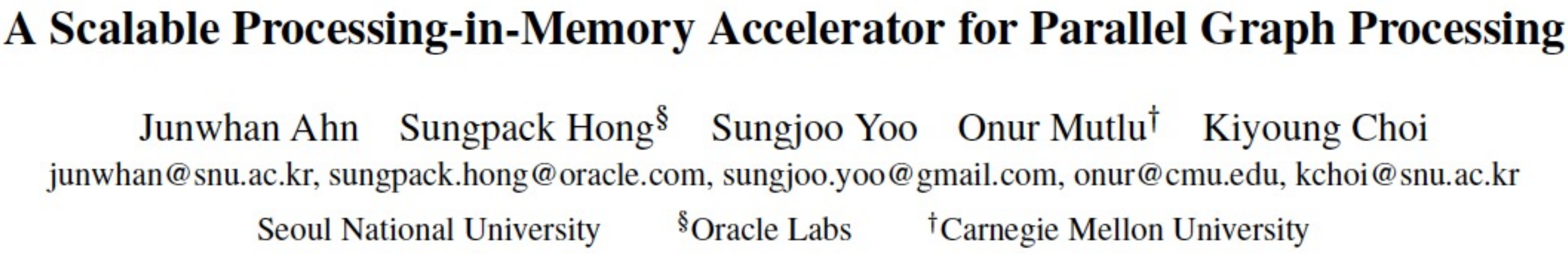


> 8X Energy Reduction

SAFARI Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

More on Tesseract

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



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

Junwhan Ahn Sungpack Hong[§] Sungjoo Yoo Onur Mutlu[†] Kiyoung Choi
junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
Seoul National University [§]Oracle Labs [†]Carnegie Mellon University

Two Key Questions in Processing Near Memory

- 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

- 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"
  *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Williamsburg, VA, USA, March 2018.
  [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[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand

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

SAFARI Carnegie Mellon Google

SAMSUNG SEOUL NATIONAL UNIVERSITY ETH Zürich

Consumer Devices

Consumer devices are everywhere!

Energy consumption is a first-class concern in consumer devices

Four Important Workloads

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

Video Playback

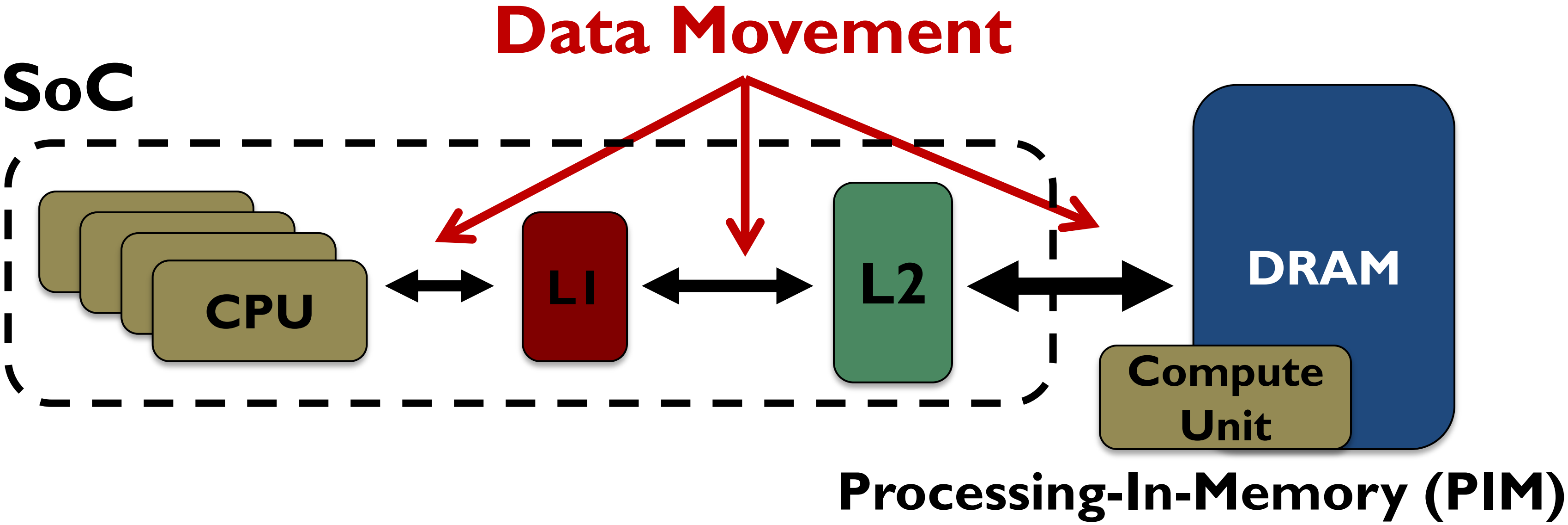
Google's **video codec**

Video Capture

Google's **video codec**

Energy Cost of Data Movement

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



Potential solution: move computation close to data

Challenge: limited area and energy budget

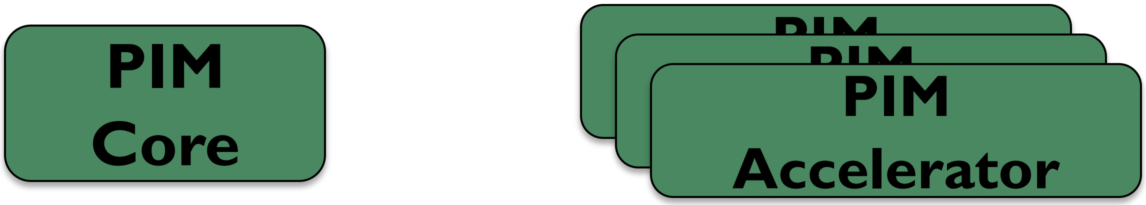
Using PIM to Reduce Data Movement

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

Small fixed-function accelerators



PIM Core

PIM Accelerator

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

Workload Analysis

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

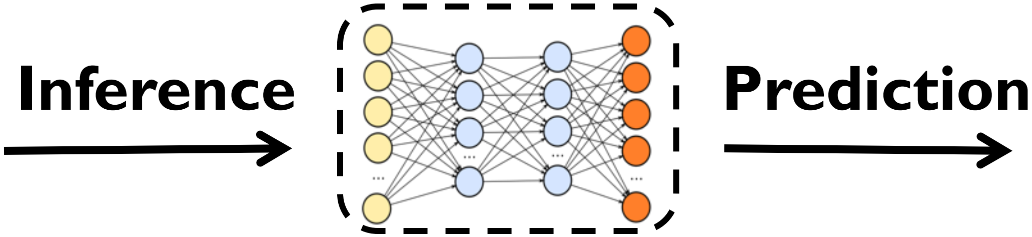
Video Playback

Google's video codec

Video Capture

Google's video codec

TensorFlow Mobile



Inference

Prediction

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

Matrix
Packing
Packed Matrix

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

floating point
Quantization
integer

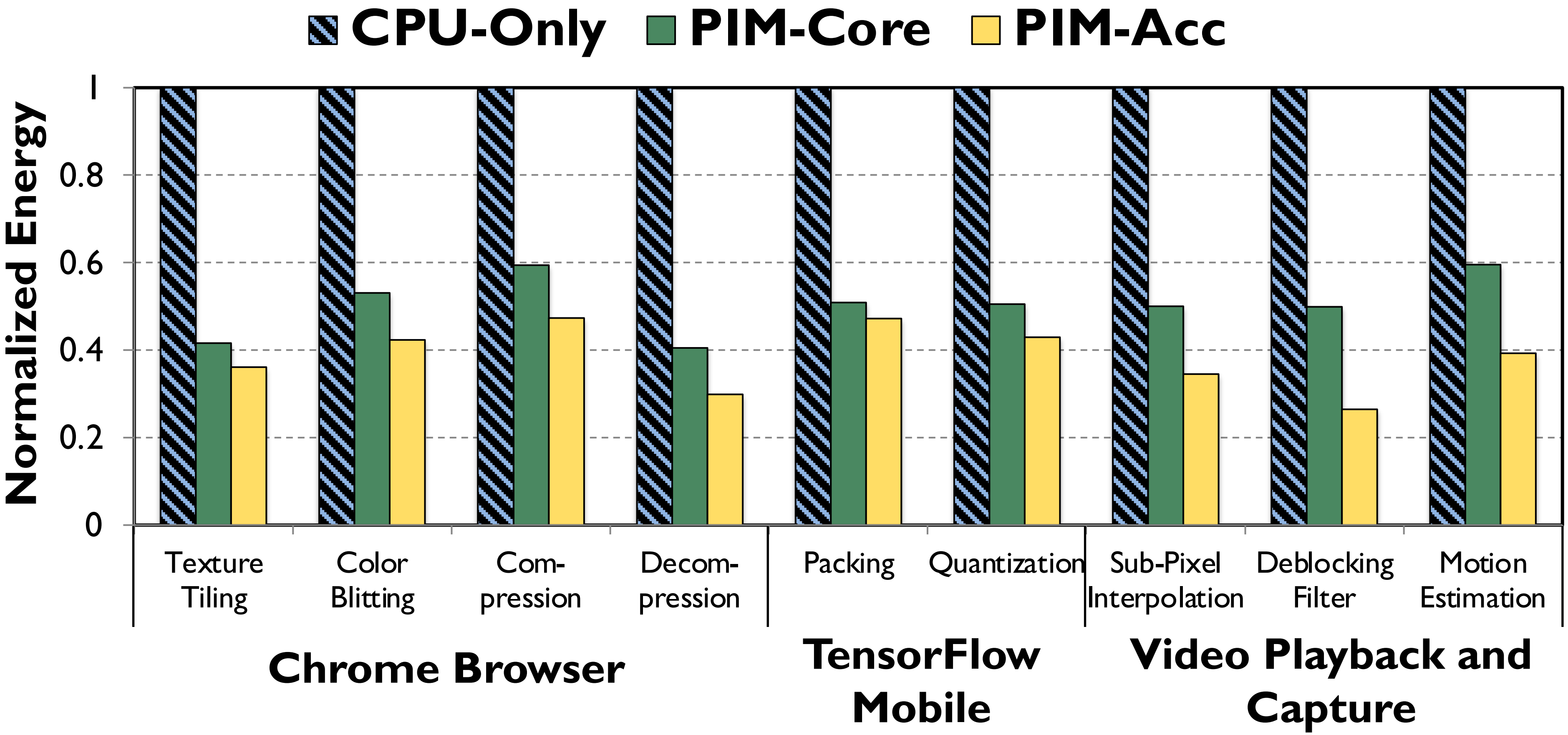
Converts 32-bit floating point to 8-bit integers to improve inference execution time and energy consumption

Up to 16.8% of the inference energy and 16.1% of inference execution time

Majority of quantization energy comes from data movement

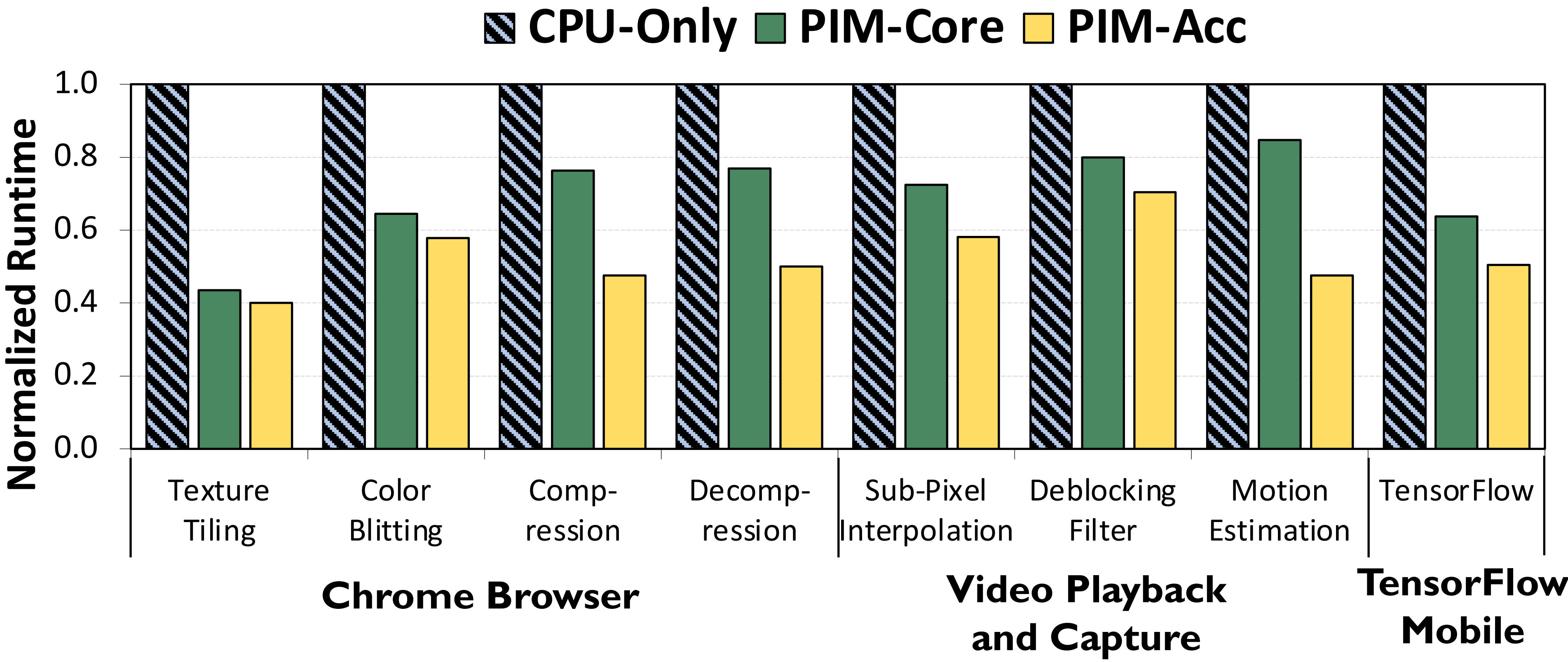
A simple data conversion operation that requires shift, addition, and multiplication operations

Normalized Energy



PIM core and PIM accelerator reduce energy consumption on average by 2.0X and 2.3X

Normalized Runtime



Offloading these kernels to PIM core and PIM accelerator reduces program runtime on average by 1.8X and 2.2X

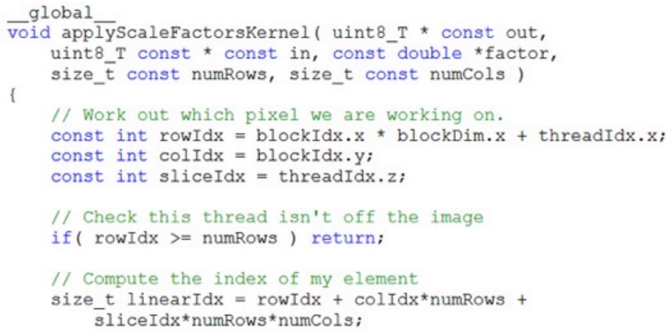
More on PIM for Mobile Devices

- 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"
*Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Williamsburg, VA, USA, March 2018.
[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

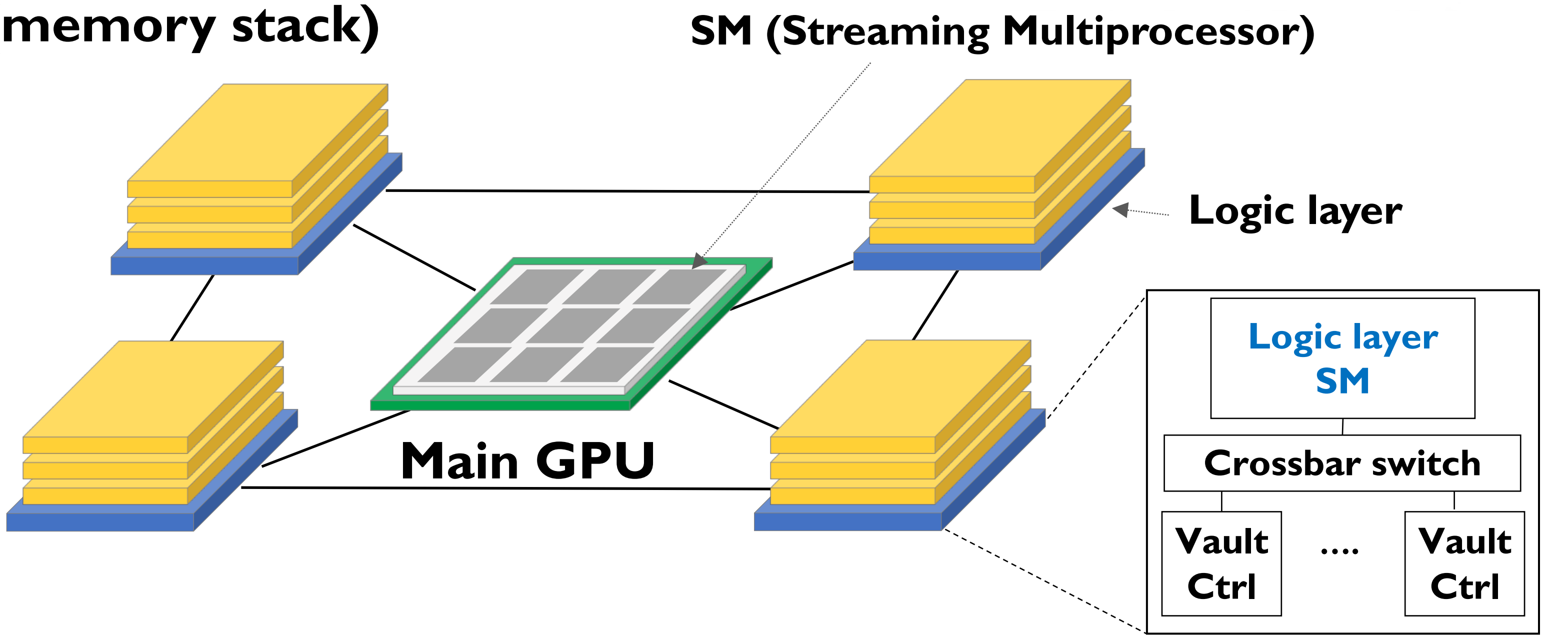
Amirali Boroumand[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

Truly Distributed GPU Processing with PIM?



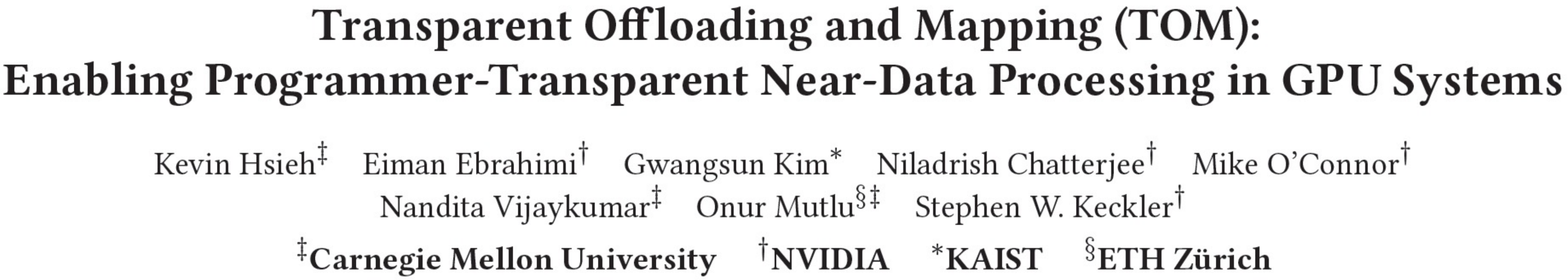
```
__global__
void applyScaleFactorsKernel( uint8_T * const out,
    uint8_T const * const in, const double *factor,
    size_t const numRows, size_t const numCols )
{
    // Work out which pixel we are working on.
    const int rowIdx = blockIdx.x * blockDim.x + threadIdx.x;
    const int colIdx = 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;
```

3D-stacked memory
(memory stack)



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"**
  Proceedings of the 43rd International Symposium on Computer Architecture (***ISCA***), Seoul, South Korea, June 2016.
  [Slides (pptx) (pdf)]
  [Lightning Session Slides (pptx) (pdf)]

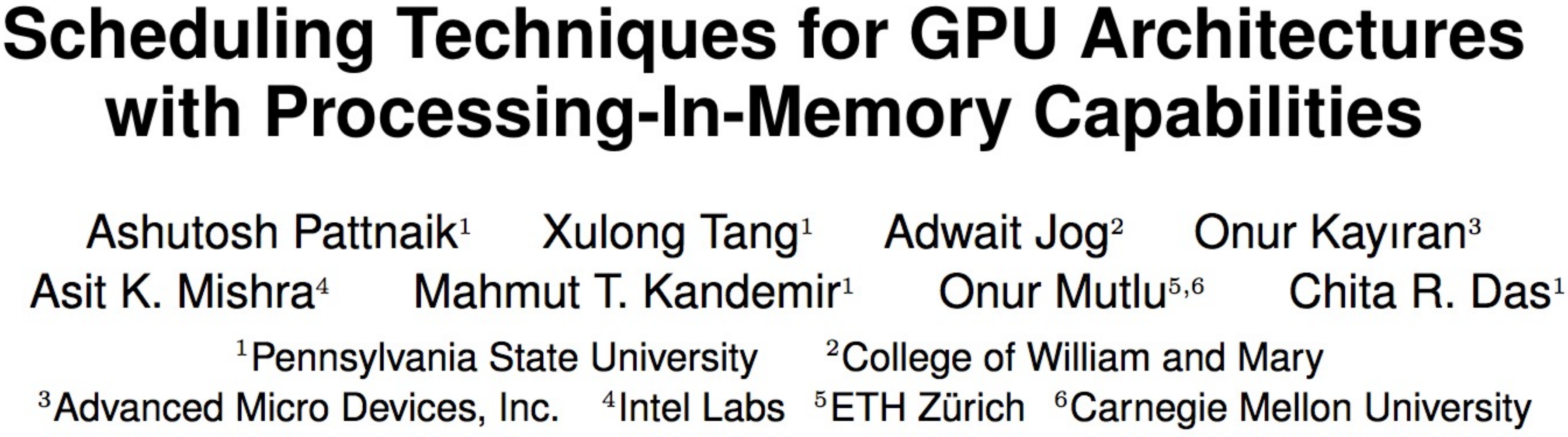


Transparent Offloading and Mapping (TOM):
Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh[‡] Eiman Ebrahimi[†] Gwangsun Kim[*] Niladrish Chatterjee[†] Mike O'Connor[†]
Nandita Vijaykumar[‡] Onur Mutlu[§‡] Stephen W. Keckler[†]
[‡]**Carnegie Mellon University** [†]**NVIDIA** [*]**KAIST** [§]**ETH Zürich**

Accelerating GPU Execution with PIM (II)

- Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, and Chita R. Das, **"Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities"** *Proceedings of the 25th International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Haifa, Israel, September 2016.



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

Ashutosh Pattnaik[1] Xulong Tang[1] Adwait Jog[2] Onur Kayıran[3]
Asit K. Mishra[4] Mahmut T. Kandemir[1] Onur Mutlu[5,6] Chita R. Das[1]

[1]Pennsylvania State University [2]College of William and Mary
[3]Advanced Micro Devices, Inc. [4]Intel Labs [5]ETH Zürich [6]Carnegie Mellon University

Accelerating Linked Data Structures

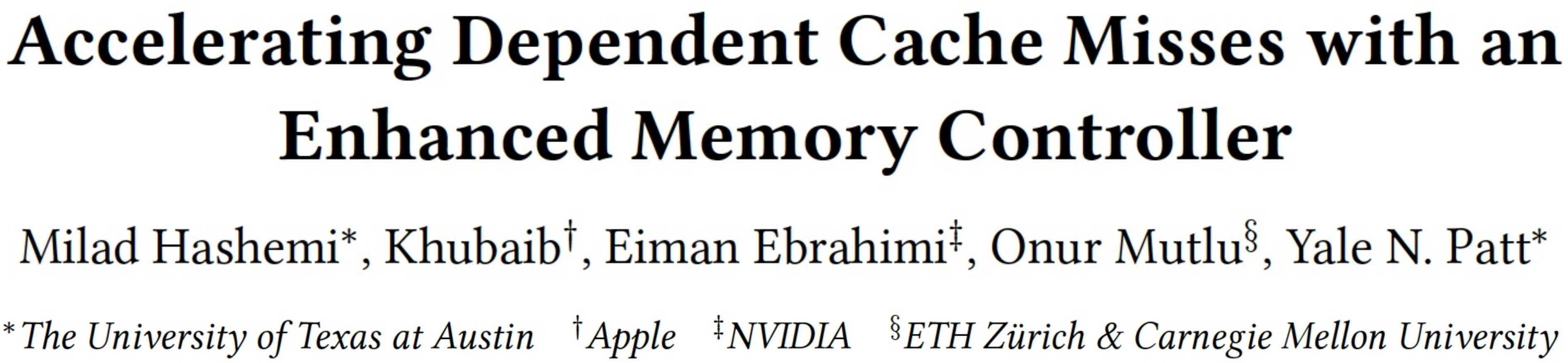
- Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu, **"Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"** *Proceedings of the 34th IEEE International Conference on Computer Design (**ICCD**)*, Phoenix,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

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

Kevin Hsieh[†] Samira Khan[‡] Nandita Vijaykumar[†]
Kevin K. Chang[†] Amirali Boroumand[†] Saugata Ghose[†] Onur Mutlu[§†]
[†]*Carnegie Mellon University* [‡]*University of Virginia* [§]*ETH Zürich*

Accelerating Dependent Cache Misses

- Milad Hashemi, Khubaib, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt, **"Accelerating Dependent Cache Misses with an Enhanced Memory Controller"**
  Proceedings of the 43rd International Symposium on Computer Architecture (***ISCA***), Seoul, South Korea, June 2016.
  [Slides (pptx) (pdf)]
  [Lightning Session Slides (pptx) (pdf)]



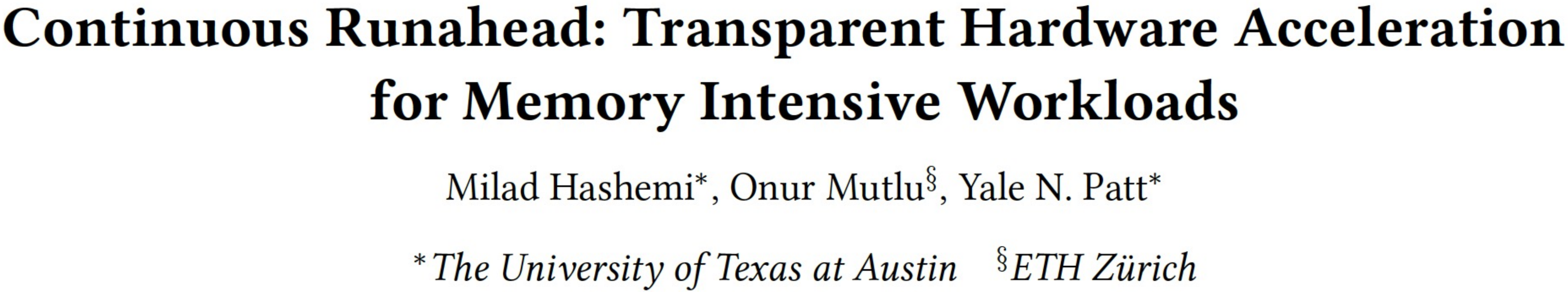
Accelerating Dependent Cache Misses with an Enhanced Memory Controller

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

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

Accelerating Runahead Execution

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



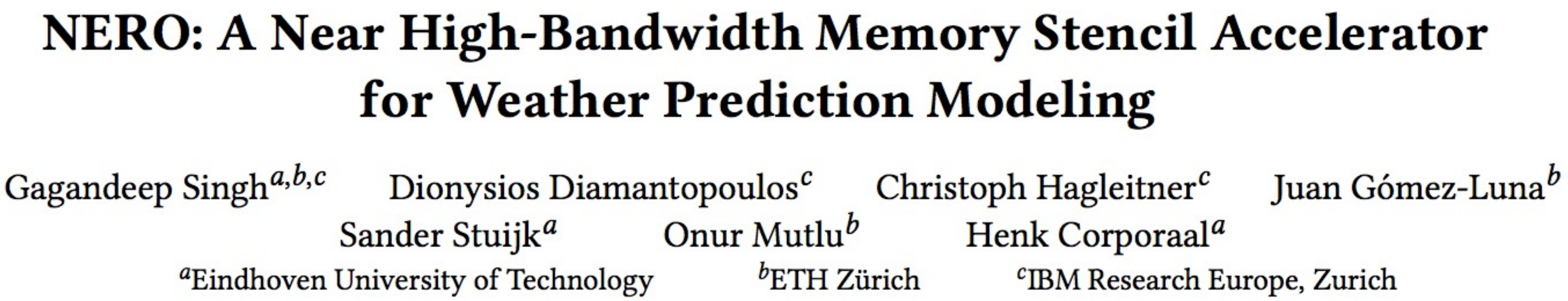
Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads

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

[*] *The University of Texas at Austin* [§] *ETH Zürich*

Accelerating Climate Modeling

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

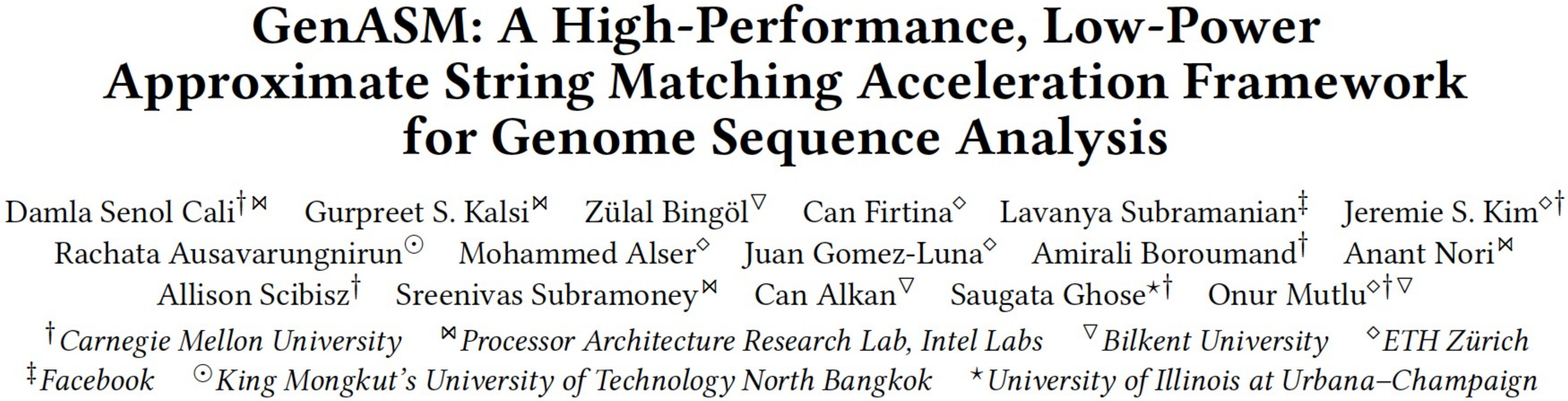


NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling

Gagandeep Singh[a,b,c] Dionysios Diamantopoulos[c] Christoph Hagleitner[c] Juan Gómez-Luna[b]
Sander Stuijk[a] Onur Mutlu[b] Henk Corporaal[a]
[a]Eindhoven University of Technology [b]ETH Zürich [c]IBM Research Europe, Zurich

Accelerating Approximate String Matching

- Damla Senol Cali, Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,
  "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"
  Proceedings of the 53rd International Symposium on Microarchitecture (***MICRO***), Virtual, October 2020.
  [Lighting Talk Video (1.5 minutes)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (18 minutes)]
  [Slides (pptx) (pdf)]



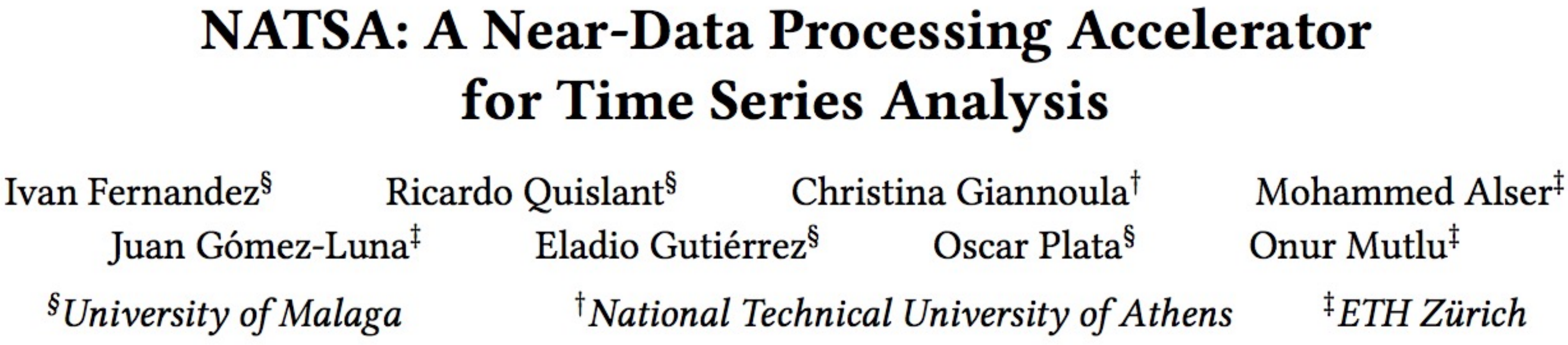
GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

Damla Senol Cali[†⋈] Gurpreet S. Kalsi[⋈] Zülal Bingöl[▽] Can Firtina[◇] Lavanya Subramanian[‡] Jeremie S. Kim[◇†]
Rachata Ausavarungnirun[⊙] Mohammed Alser[◇] Juan Gomez-Luna[◇] Amirali Boroumand[†] Anant Nori[⋈]
Allison Scibisz[†] Sreenivas Subramoney[⋈] Can Alkan[▽] Saugata Ghose[⋆†] Onur Mutlu[◇†▽]

[†] *Carnegie Mellon University* [⋈] *Processor Architecture Research Lab, Intel Labs* [▽] *Bilkent University* [◇] *ETH Zürich*
[‡] *Facebook* [⊙] *King Mongkut's University of Technology North Bangkok* [⋆] *University of Illinois at Urbana–Champaign*

Accelerating Time Series Analysis

- Ivan Fernandez, Ricardo Quislant, Christina Giannoula, Mohammed Alser, Juan Gómez-Luna, Eladio Gutiérrez, Oscar Plata, and Onur Mutlu,
  "NATSA: A Near-Data Processing Accelerator for Time Series Analysis"
  *Proceedings of the 38th IEEE International Conference on Computer Design (**ICCD**)*, Virtual, October 2020.
  [Slides (pptx) (pdf)]
  [Talk Video (10 minutes)]
  [Source Code]



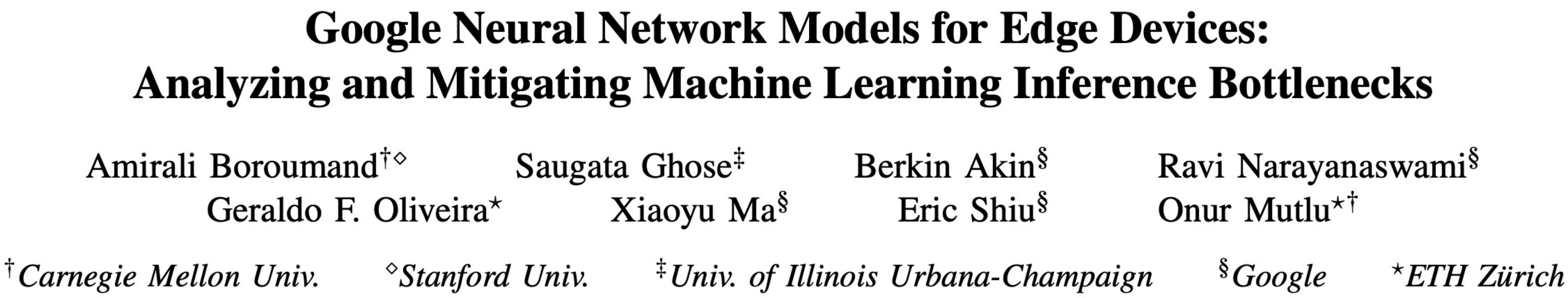
NATSA: A Near-Data Processing Accelerator for Time Series Analysis

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

[§]*University of Malaga* [†]*National Technical University of Athens* [‡]*ETH Zürich*

Accelerating Neural Network Inference

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
"Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
*Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Virtual, September 2021.
[Slides (pptx) (pdf)]



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

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

[†]*Carnegie Mellon Univ.* [◇]*Stanford Univ.* [‡]*Univ. of Illinois Urbana-Champaign* [§]*Google* [⋆]*ETH Zürich*

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

Amirali Boroumand Saugata Ghose Berkin Akin

Ravi Narayanaswami Geraldo F. Oliveira Xiaoyu Ma

Eric Shiu Onur Mutlu

PACT 2021

SAFARI

Carnegie Mellon UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN Google ETH Zürich

Executive Summary

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

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

Problem: The Edge TPU accelerator suffers from three challenges:

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

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

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

Key Mechanism: A new framework called Mensa

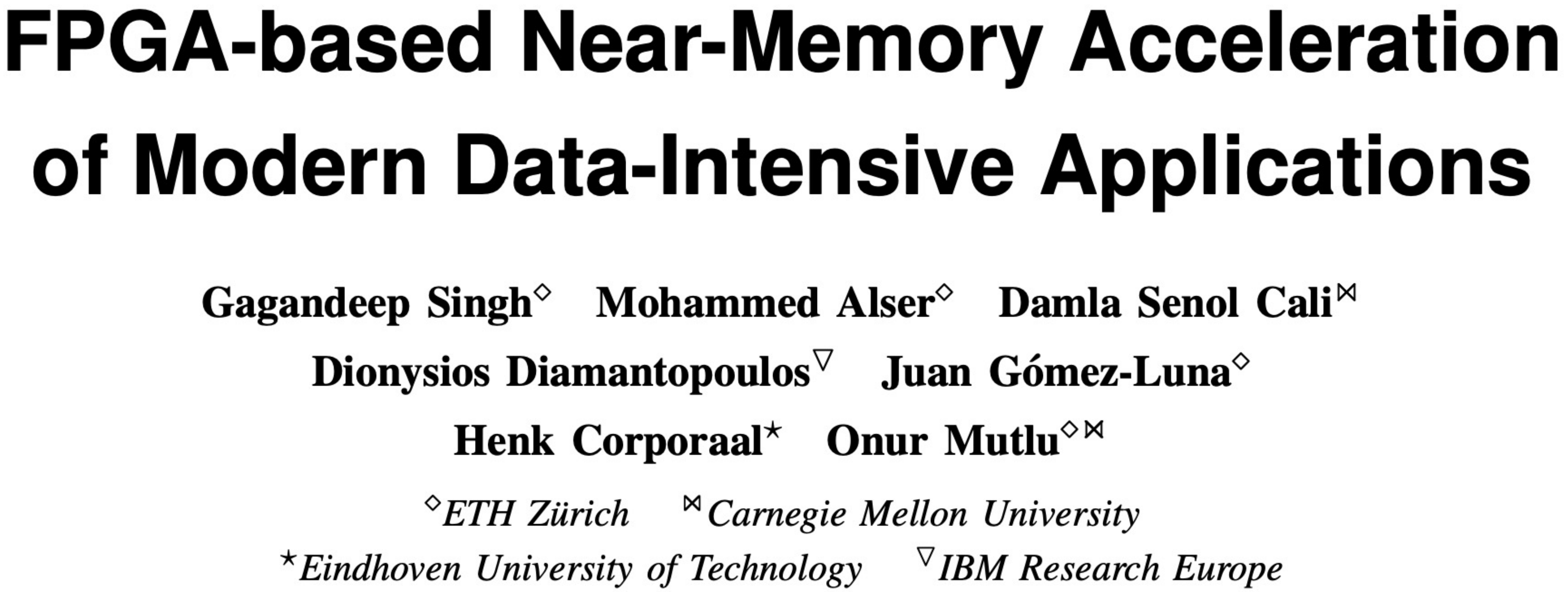
- Mensa consists of heterogeneous accelerators whose dataflow and hardware are specialized for specific families of layers

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

- Mensa improves performance and energy by 3.0X and 3.1X
- Mensa reduces cost and improves area efficiency

FPGA-based Processing Near Memory

- Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu, **"FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"**
  IEEE Micro ***(IEEE MICRO)***, 2021.

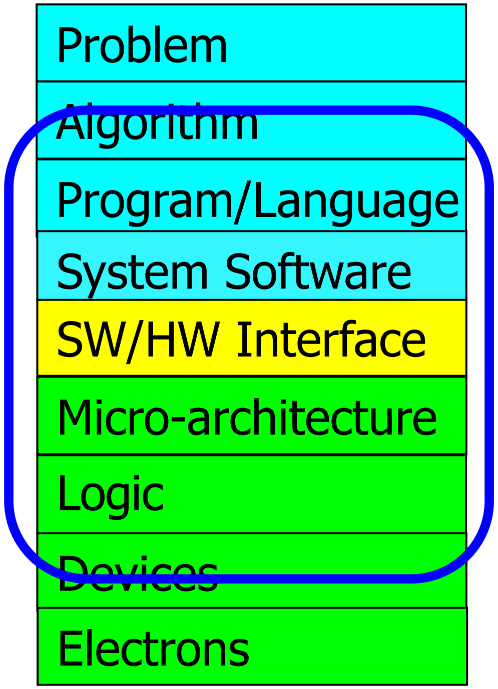


FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh[◇] **Mohammed Alser**[◇] **Damla Senol Cali**[⋈]
Dionysios Diamantopoulos[▽] **Juan Gómez-Luna**[◇]
Henk Corporaal[⋆] **Onur Mutlu**[◇⋈]

[◇]*ETH Zürich* [⋈]*Carnegie Mellon University*
[⋆]*Eindhoven University of Technology* [▽]*IBM Research Europe*

We Need to Revisit the Entire Stack



Problem

Algorithm

Program/Language

System Software

SW/HW Interface

Micro-architecture

Logic

Devices

Electrons

We can get there step by step

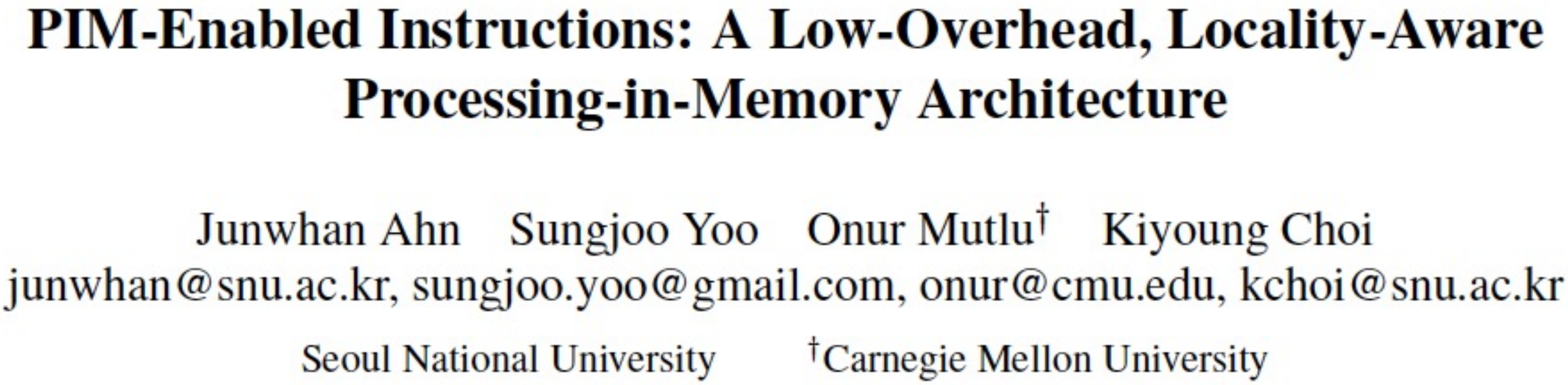
Two Key Questions in Processing Near Memory

- 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: Simple Processing in Memory

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"** *Proceedings of the 42nd International Symposium on Computer Architecture* (***ISCA***), Portland, OR, June 2015. [Slides (pdf)] [Lightning Session Slides (pdf)]



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

Junwhan Ahn Sungjoo Yoo Onur Mutlu[†] Kiyoung Choi
junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
Seoul National University [†]Carnegie Mellon University

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

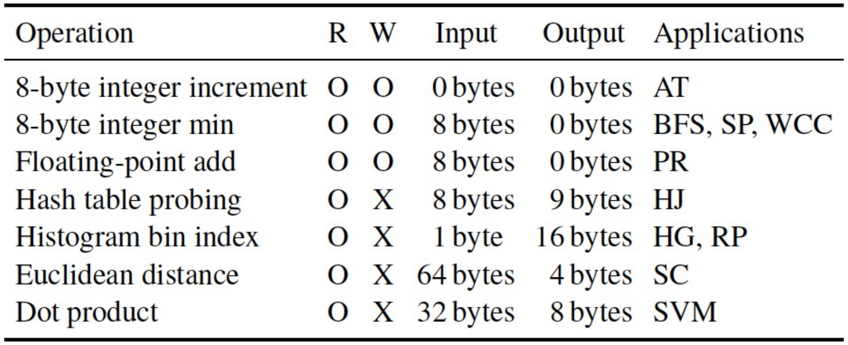
PEI: PIM-Enabled Instructions (Example)

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

pim.add r1, (r2)

pfence

Table 1: Summary of Supported PIM Operations



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

- 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)

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

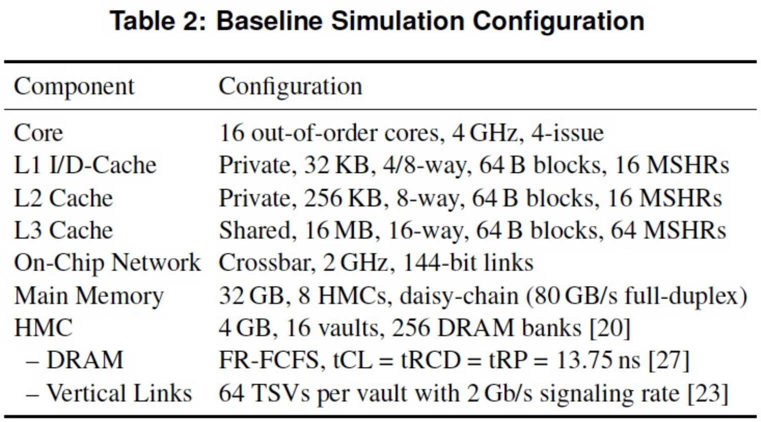


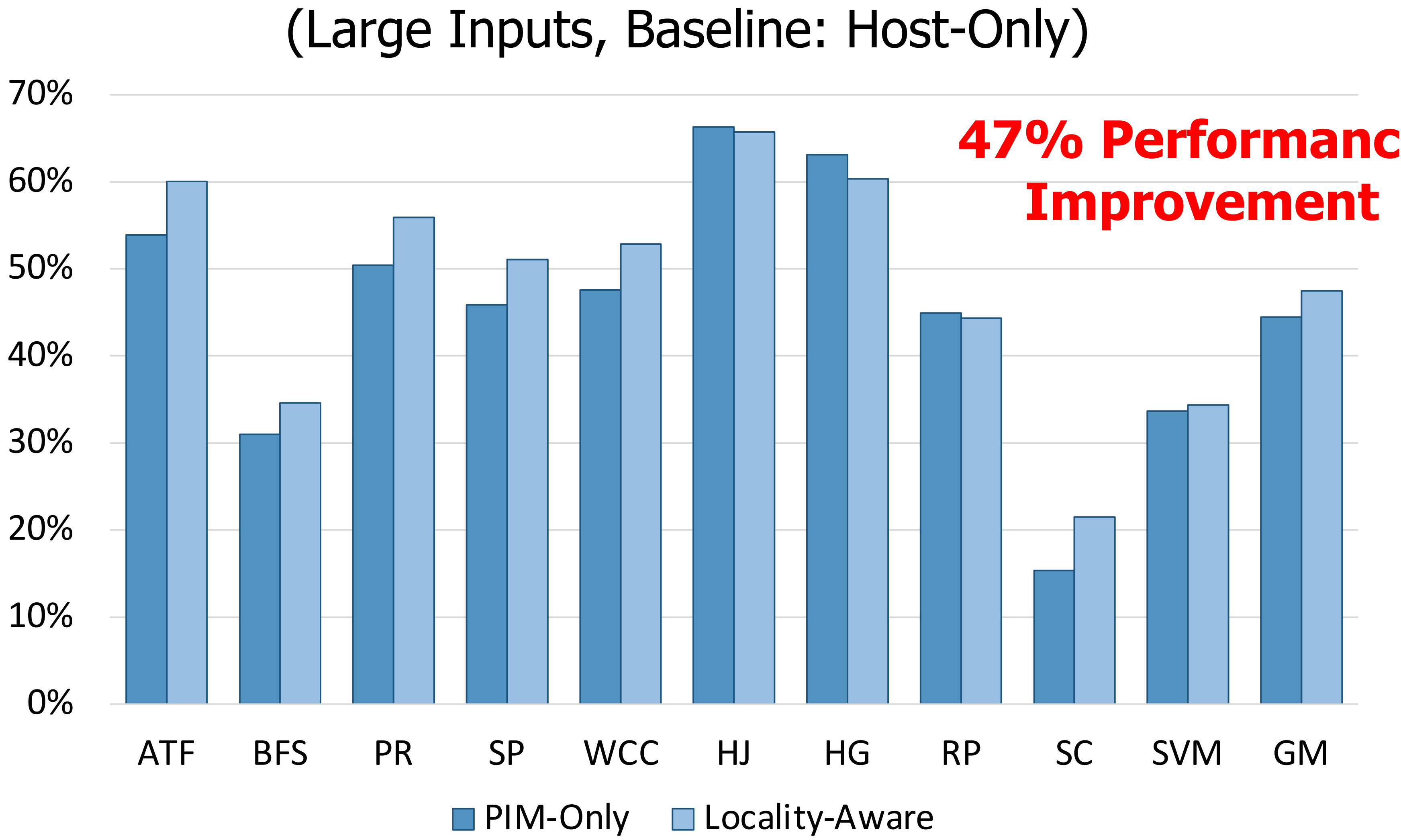
Table 2: Baseline Simulation Configuration

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

- 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

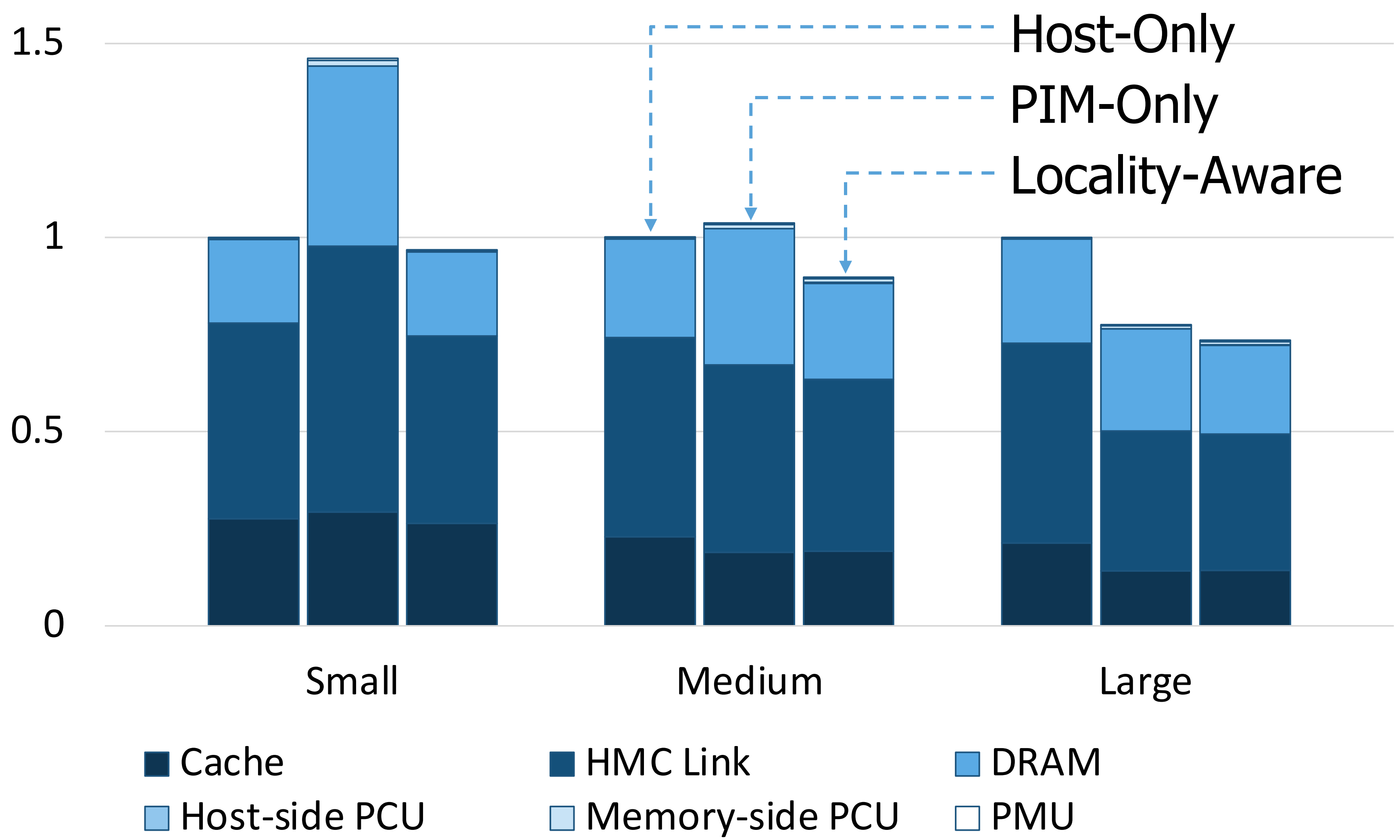
PEI Performance Delta: Large Data Sets



47% Performance Improvement

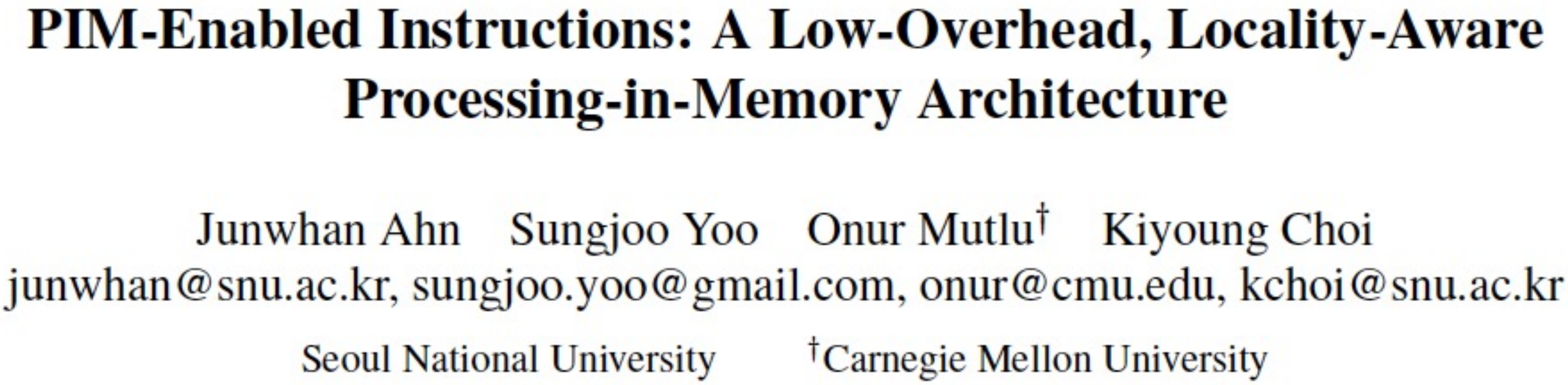
PEI Energy Consumption

25% Energy Reduction



Simpler PIM: PIM-Enabled Instructions

- Junwhan Ahn, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi, **"PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture"** *Proceedings of the 42nd International Symposium on Computer Architecture* (***ISCA***), Portland, OR, June 2015. [Slides (pdf)] [Lightning Session Slides (pdf)]



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

Junwhan Ahn Sungjoo Yoo Onur Mutlu[†] Kiyoung Choi
junwhan@snu.ac.kr, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
Seoul National University [†]Carnegie Mellon University

Processing in Memory: Two Approaches

1. Processing using Memory
2. Processing near Memory

Eliminating the Adoption Barriers

How to Enable Adoption of Processing in Memory

Potential Barriers to Adoption of PIM

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

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

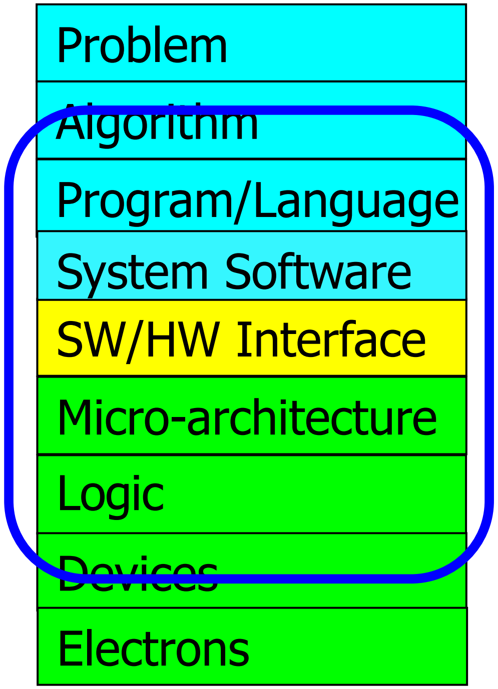
3. **System** support: coherence, synchronization, virtual memory

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

5. **Infrastructures** to assess benefits and feasibility

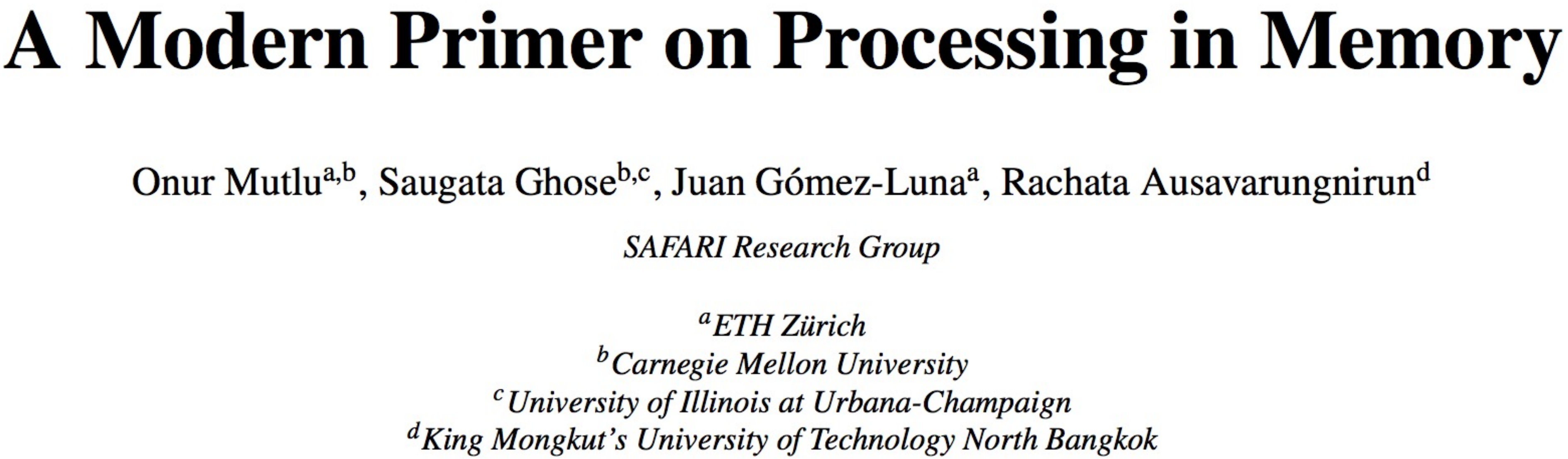
All can be solved with change of mindset

We Need to Revisit the Entire Stack



We can get there step by step

PIM Review and Open Problems



A Modern Primer on Processing in Memory

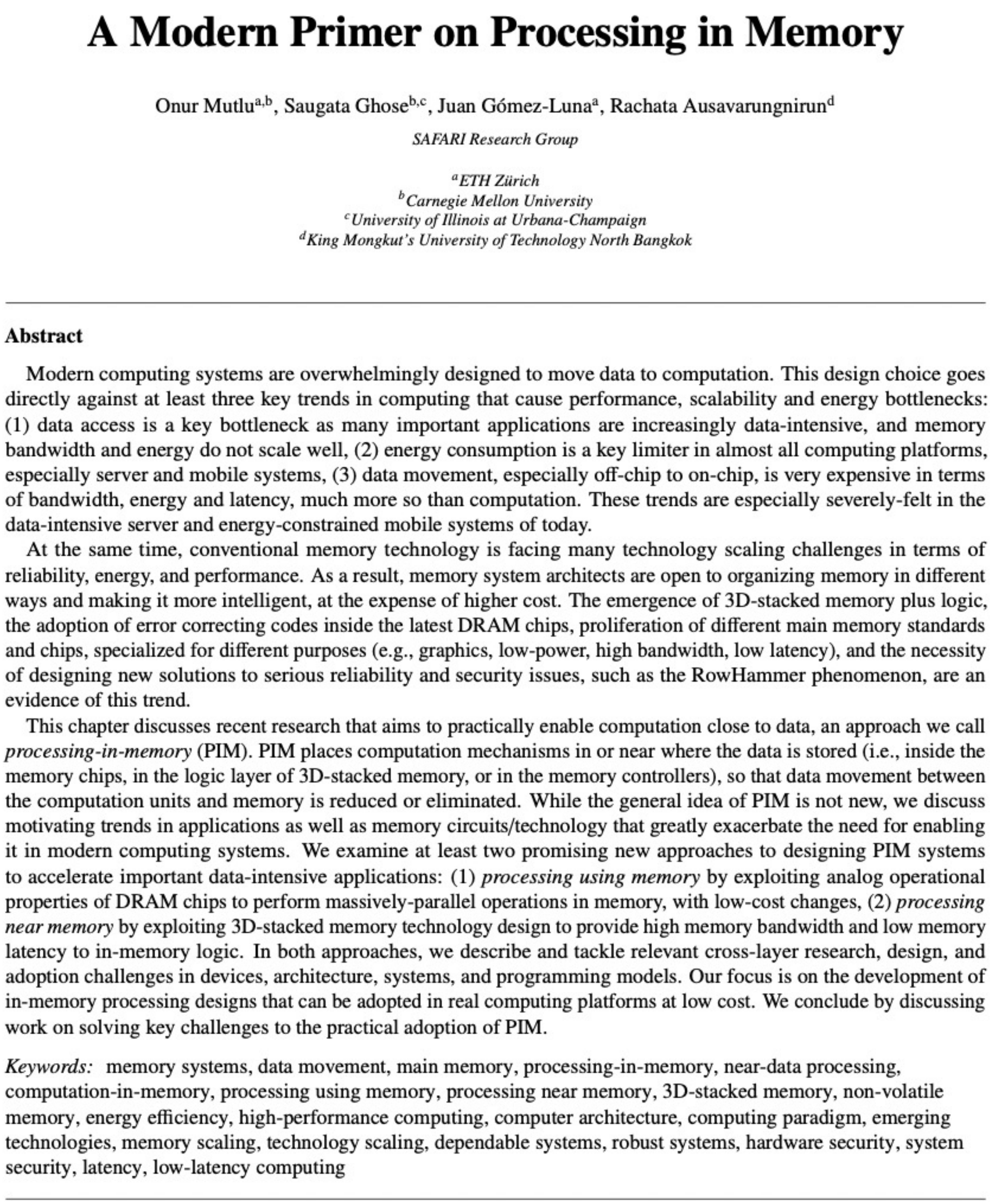
Onur Mutlu[a,b], Saugata Ghose[b,c], Juan Gómez-Luna[a], Rachata Ausavarungnirun[d]

SAFARI Research Group

[a] *ETH Zürich*
[b] *Carnegie Mellon University*
[c] *University of Illinois at Urbana-Champaign*
[d] *King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,
"A Modern Primer on Processing in Memory"
Invited Book Chapter in ***Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann***, Springer, to be published in 2021.

https://arxiv.org/pdf/1903.03988.pdf



A Modern Primer on Processing in Memory

Onur Mutlu[a,b], Saugata Ghose[b,c], Juan Gómez-Luna[a], Rachata Ausavarungnirun[d]

SAFARI Research Group

[a]*ETH Zürich*
[b]*Carnegie Mellon University*
[c]*University of Illinois at Urbana-Champaign*
[d]*King Mongkut's University of Technology North Bangkok*

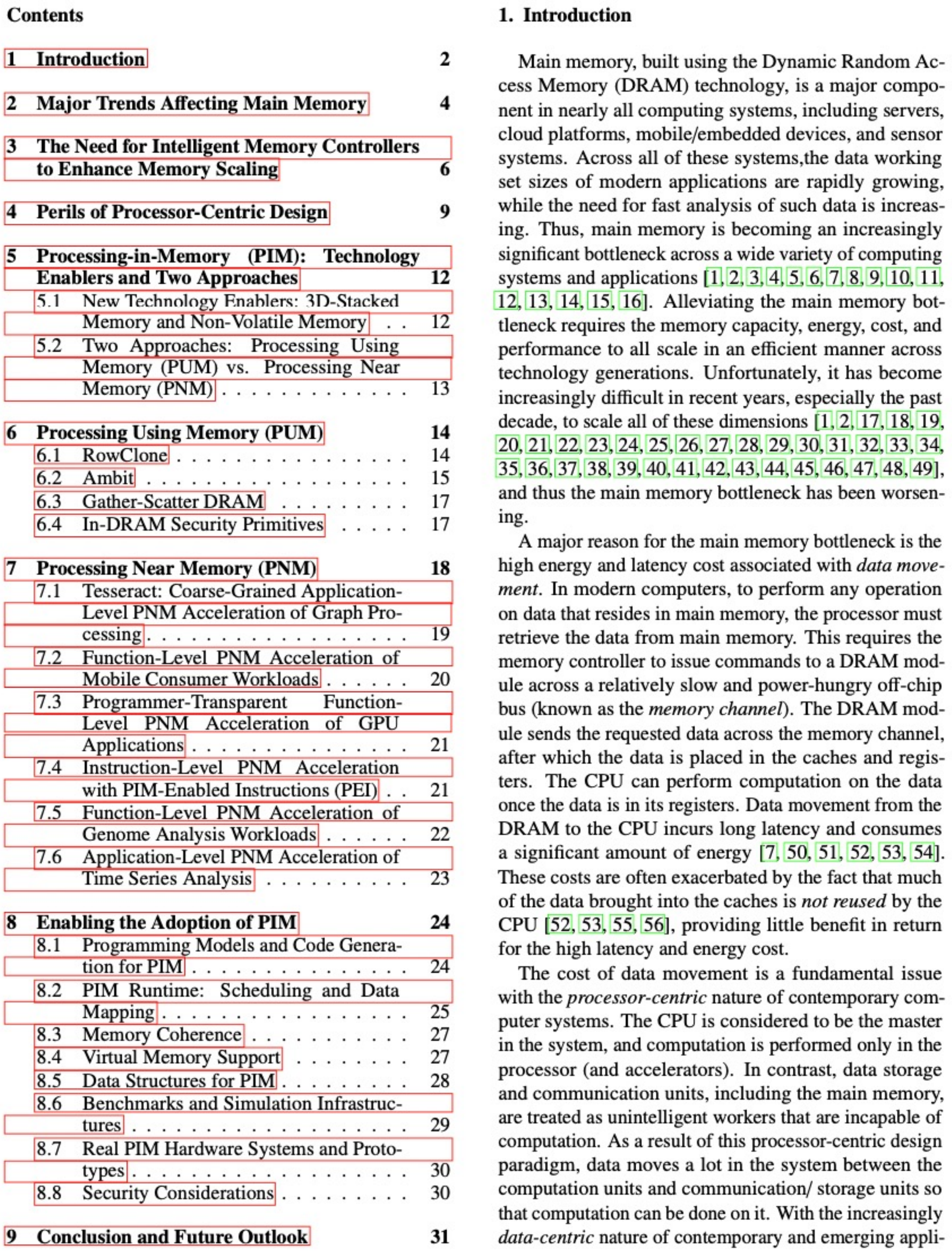
Abstract

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

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

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

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



Contents

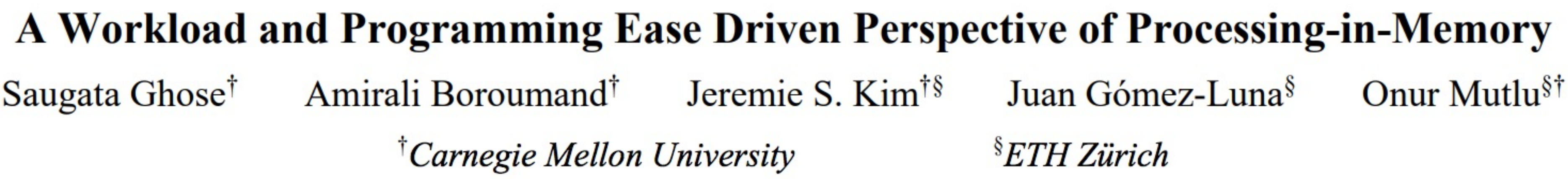
1. Introduction

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

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

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

PIM Review and Open Problems (II)



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

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

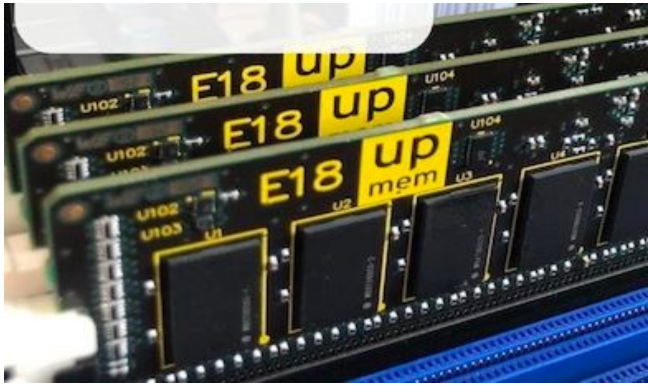
[†]*Carnegie Mellon University* [§]*ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu, **"Processing-in-Memory: A Workload-Driven Perspective"** *Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence*, to appear in November 2019. [Preliminary arXiv version]

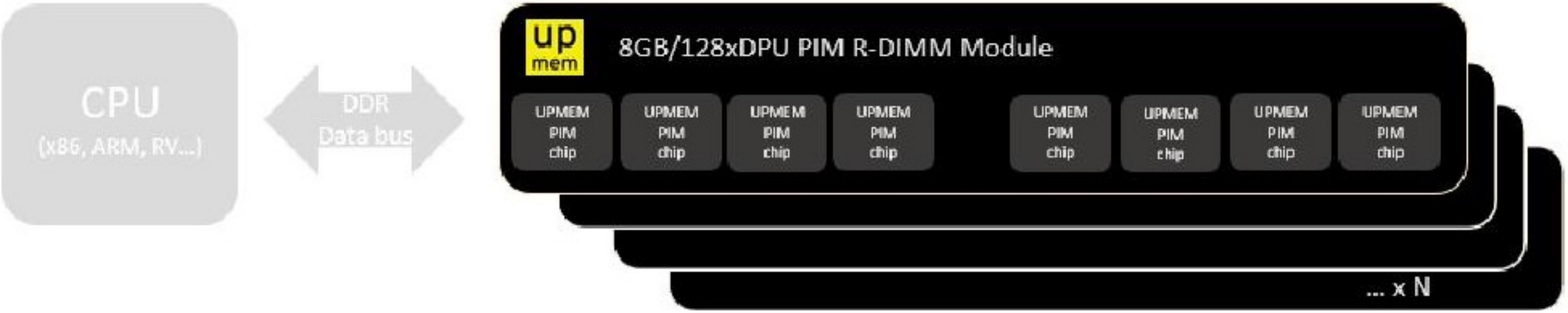
https://arxiv.org/pdf/1907.12947.pdf

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

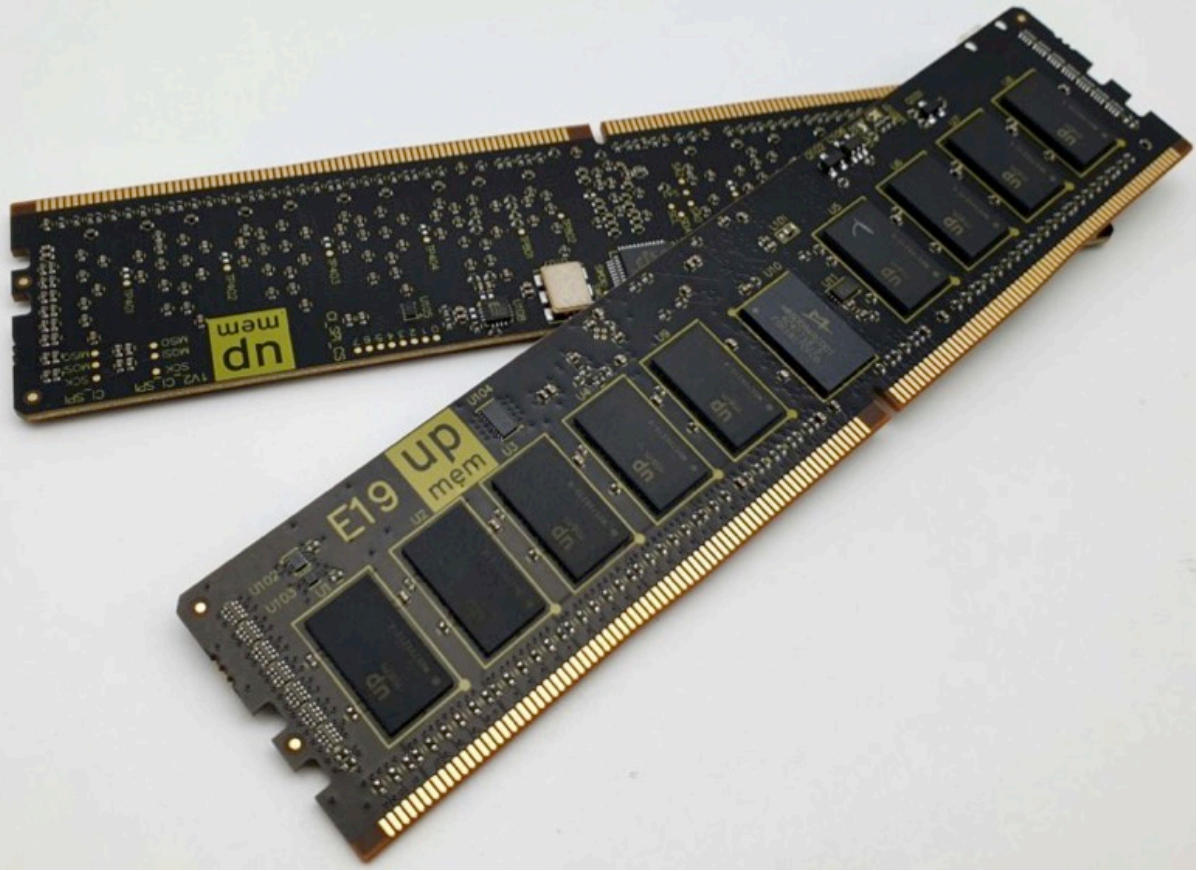


https://www.anandtech.com/show/14750/hot-chips-31-analysis-inmemory-processing-by-upmem

https://www.upmem.com/video-upmem-presenting-its-true-processing-in-memory-solution-hot-chips-2019/

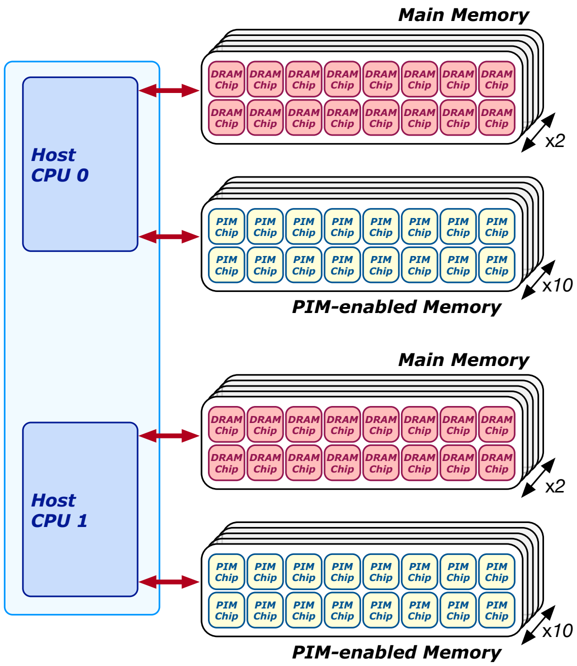
UPMEM Memory Modules

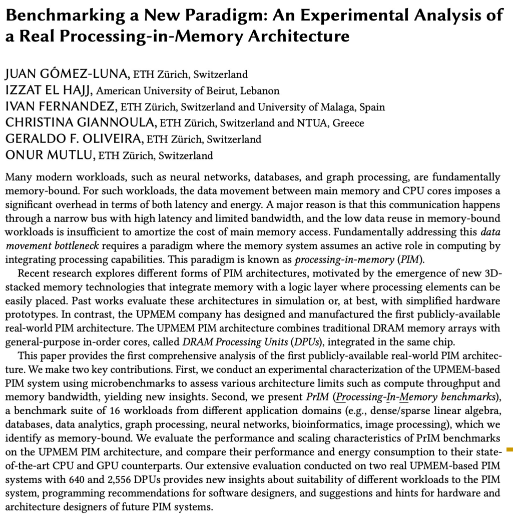
- E19: 8 chips DIMM (1 rank). DPUs @ 267 MHz
- P21: 16 chips DIMM (2 ranks). DPUs @ 350 MHz



www.upmem.com

2,560-DPU Processing-in-Memory System





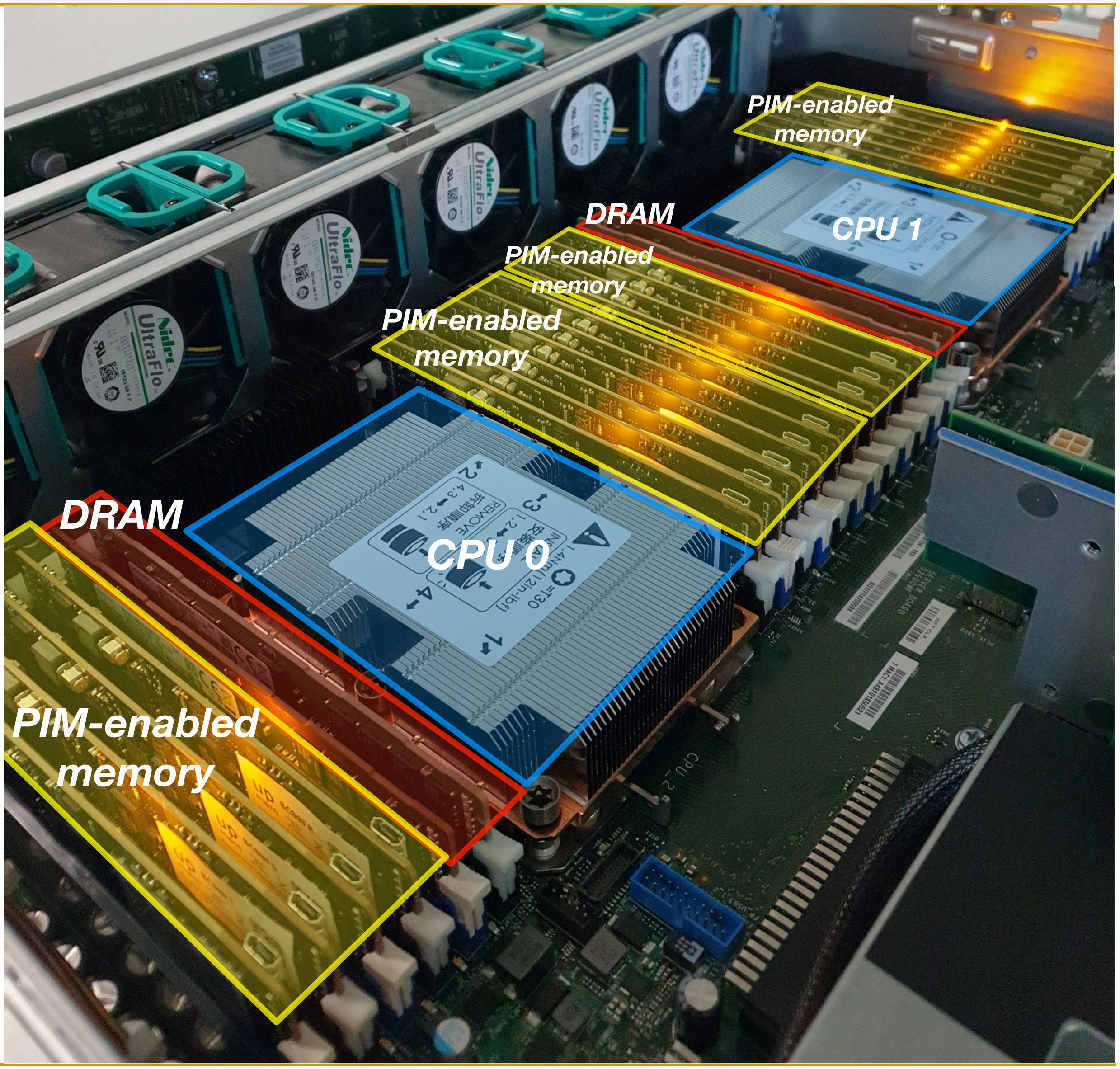
Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture

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

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

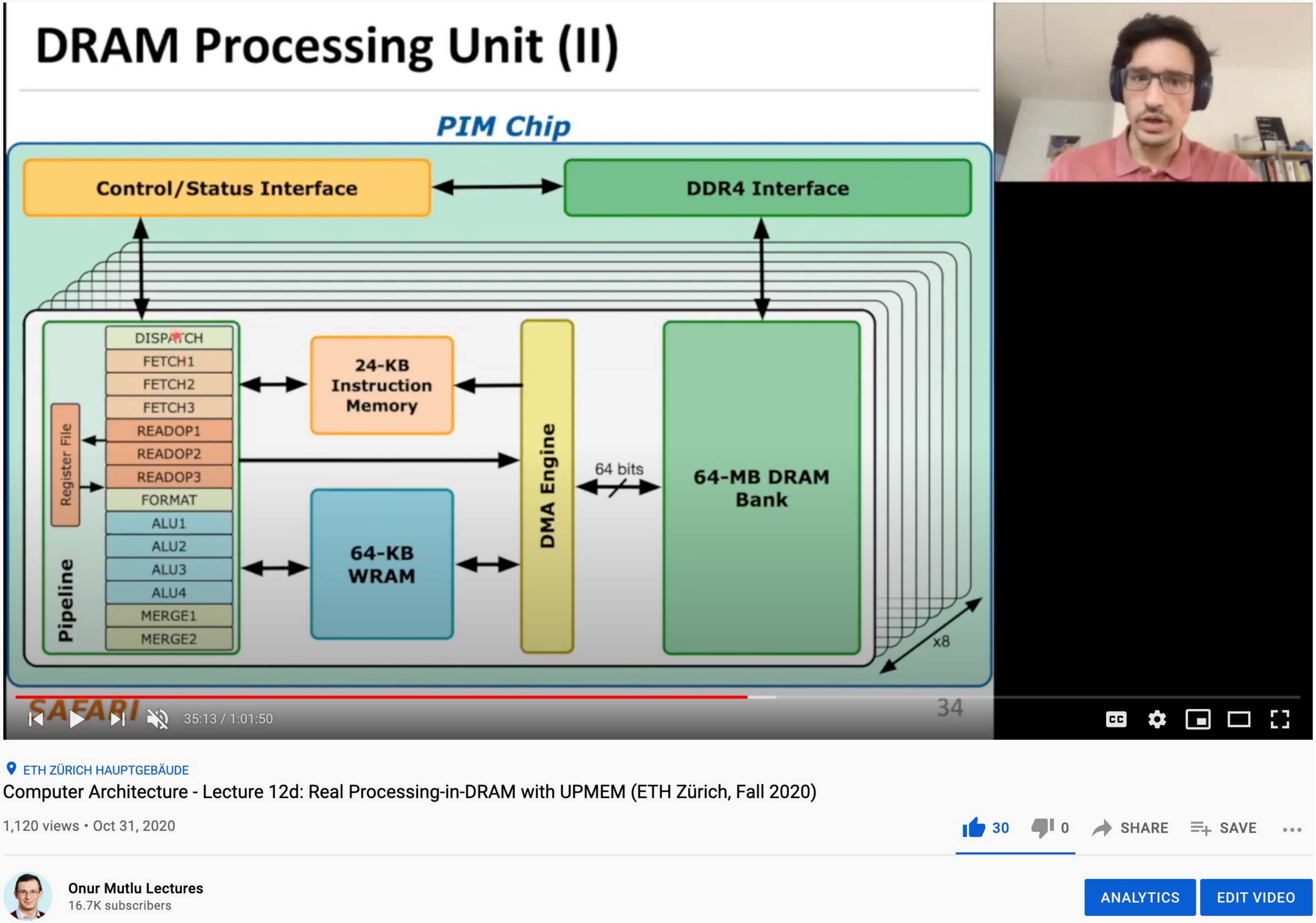
Recent research explores different forms of PIM architectures, motivated by the emergence of new 3D-stacked memory technologies that integrate memory with a logic layer where processing elements can be easily placed. Past works evaluate these architectures in simulation or, at best, with simplified hardware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

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



https://arxiv.org/pdf/2105.03814.pdf

More on the UPMEM PIM System



DRAM Processing Unit (II)

ETH ZÜRICH HAUPTGEBÄUDE

Computer Architecture - Lecture 12d: Real Processing-in-DRAM with UPMEM (ETH Zürich, Fall 2020)

1,120 views • Oct 31, 2020

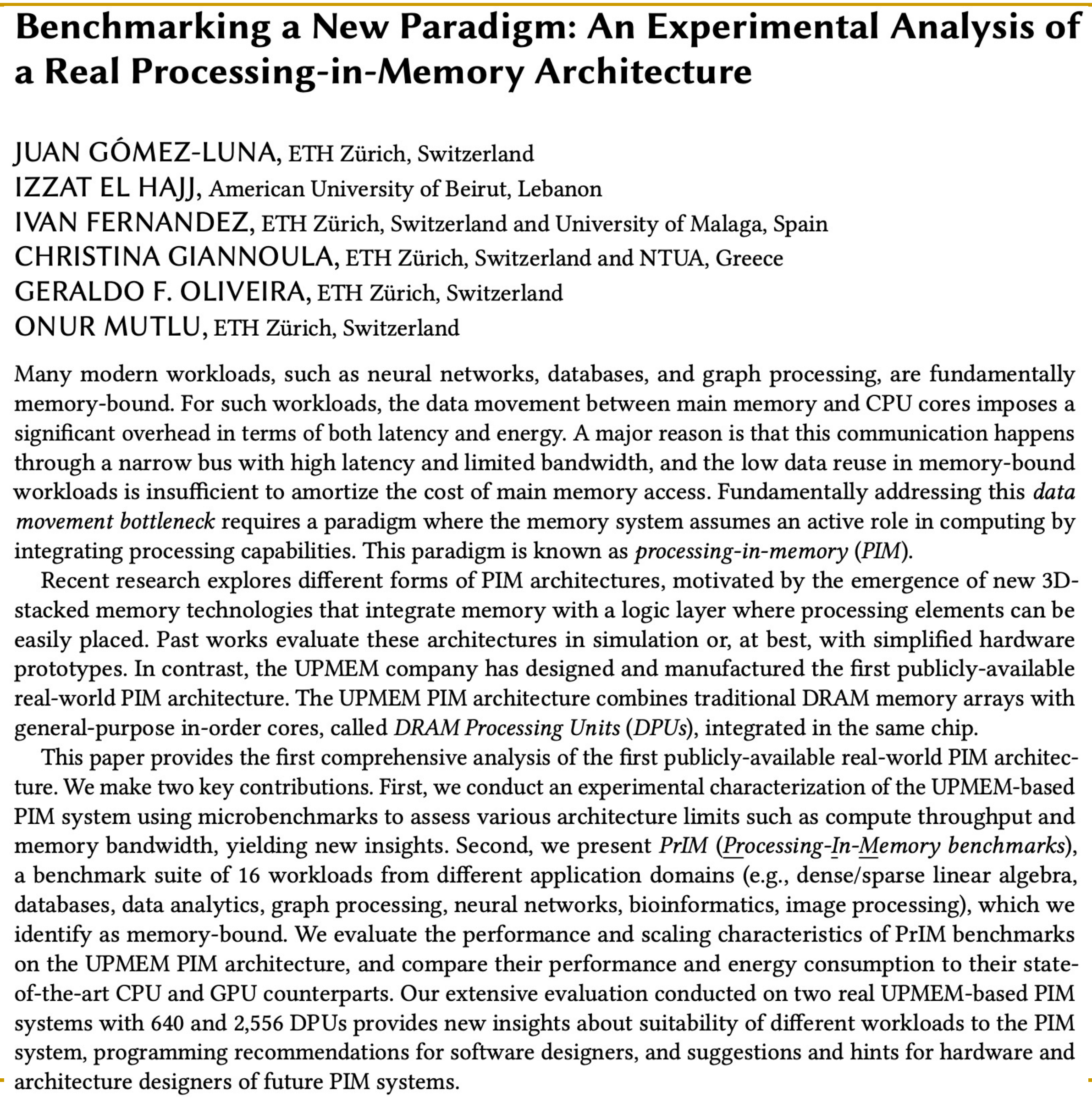
30 0

Onur Mutlu Lectures

16.7K subscribers

https://www.youtube.com/watch?v=Sscy1Wrr22A&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=26

Experimental Analysis of the UPMEM PIM Engine



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

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

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

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

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

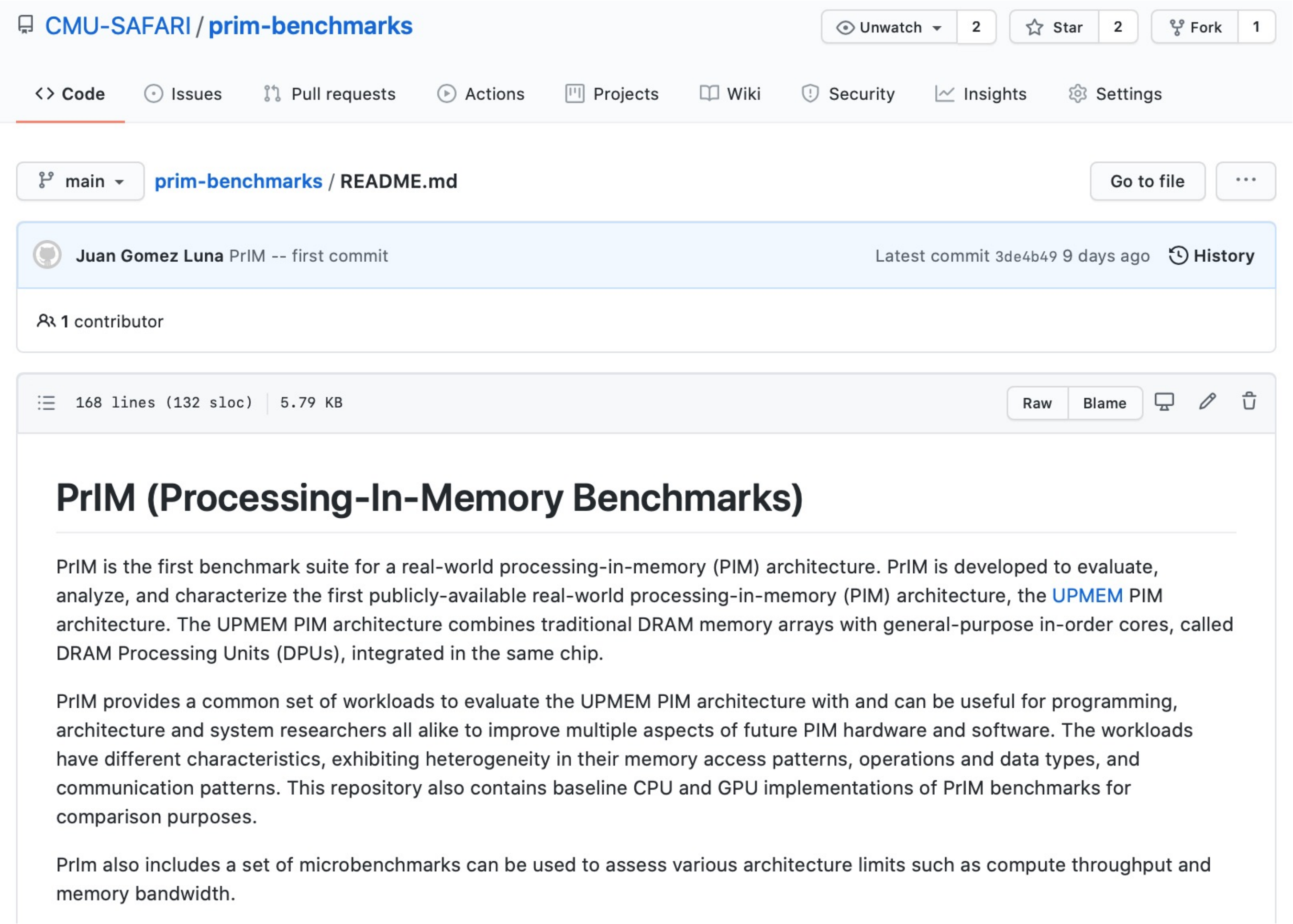
https://arxiv.org/pdf/2105.03814.pdf

PrIM Benchmarks: Application Domains

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

PrIM Benchmarks are Open Source

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



CMU-SAFARI / prim-benchmarks
Unwatch 2
Star 2
Fork 1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
main
prim-benchmarks / README.md
Go to file
Juan Gomez Luna PrIM -- first commit
Latest commit 3de4b49 9 days ago
History
1 contributor
168 lines (132 sloc) | 5.79 KB
Raw
Blame

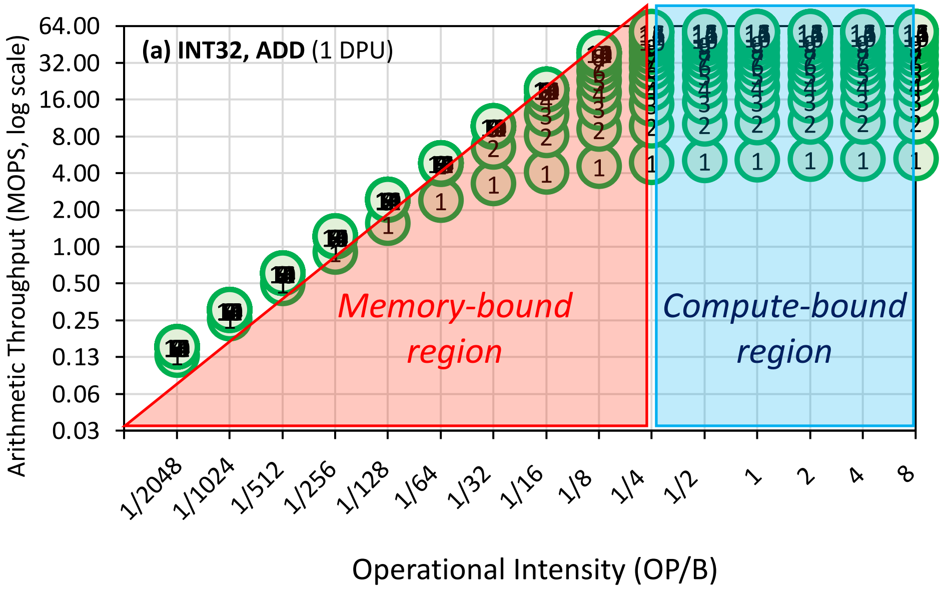
PrIM (Processing-In-Memory Benchmarks)

PrIM is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publicly-available real-world processing-in-memory (PIM) architecture, the UPMEM PIM architecture. The UPMEM PIM architecture combines traditional DRAM memory arrays with general-purpose in-order cores, called DRAM Processing Units (DPUs), integrated in the same chip.

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

PrIm also includes a set of microbenchmarks can be used to assess various architecture limits such as compute throughput and memory bandwidth.

Key Takeaway 1

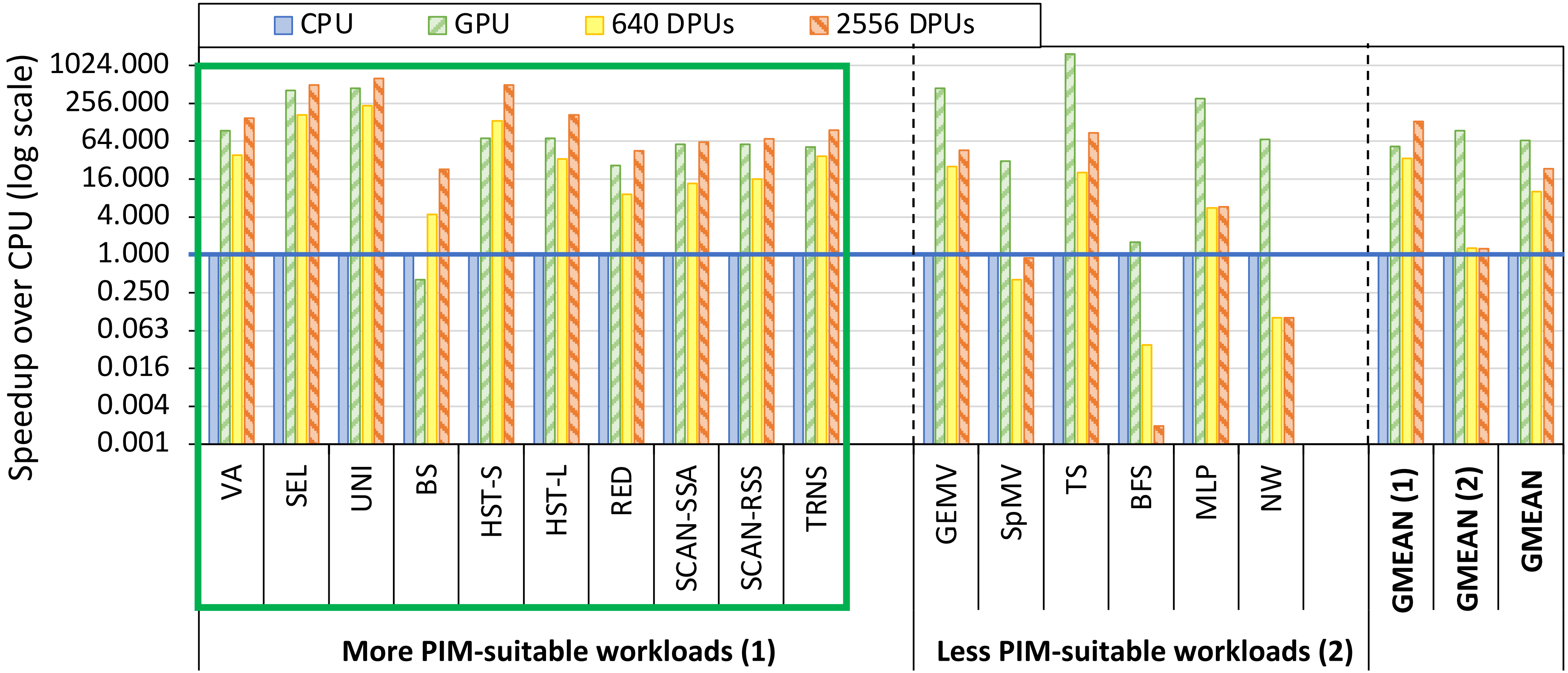


The throughput saturation point is as low as ¼ OP/B, i.e., 1 integer addition per every 32-bit element fetched

KEY TAKEAWAY 1

The UPMEM PIM architecture is fundamentally compute bound. As a result, **the most suitable workloads are memory-bound.**

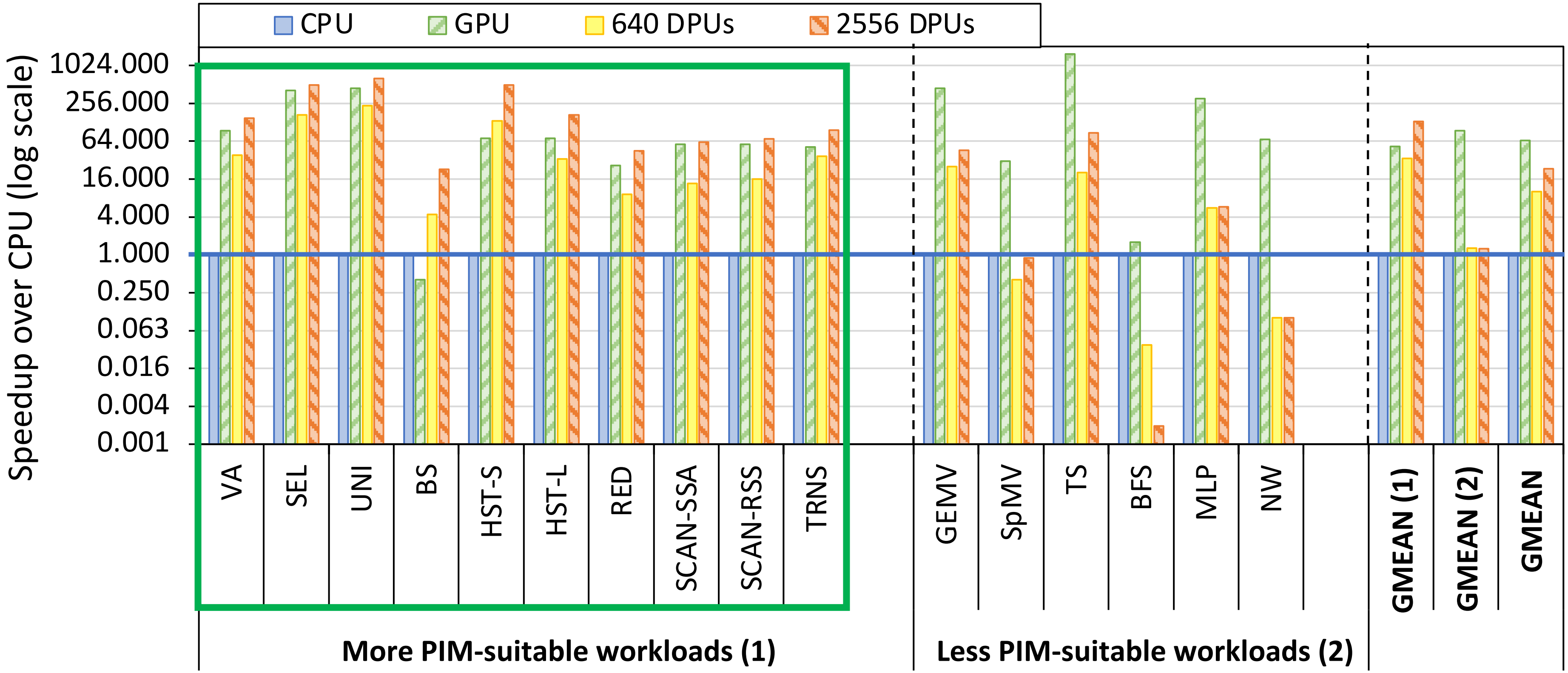
Key Takeaway 2



KEY TAKEAWAY 2

The most well-suited workloads for the UPMEM PIM architecture use no arithmetic operations or use only simple operations (e.g., bitwise operations and integer addition/subtraction).

Key Takeaway 3



KEY TAKEAWAY 3

The most well-suited workloads for the UPMEM PIM architecture require little or no communication across DPUs (inter-DPU communication).

Key Takeaway 4

KEY TAKEAWAY 4

- UPMEM-based PIM systems **outperform state-of-the-art CPUs in terms of performance and energy efficiency on most of PrIM benchmarks.**

- UPMEM-based PIM systems **outperform state-of-the-art GPUs on a majority of PrIM benchmarks**, and the outlook is even more positive for future PIM systems.

- UPMEM-based PIM systems are **more energy-efficient than state-of-the-art CPUs and GPUs on workloads that they provide performance improvements** over the CPUs and the GPUs.

More on UPMEM System & Analysis

- Juan Gomez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu, **"Benchmarking a New Paradigm: An Experimental Analysis of a Real Processing-in-Memory Architecture"**
  Preprint in ***arXiv***, 9 May 2021.
  [arXiv preprint]
  [PrIM Benchmarks Source Code]
  [Slides (pptx) (pdf)]
  [Long Talk Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [SAFARI Live Seminar Slides (pptx) (pdf)]
  [SAFARI Live Seminar Video (2 hrs 57 mins)]
  [Lightning Talk Video (3 minutes)]
  [Short Talk Video (21 minutes)]
  [1-hour Talk Video (58 minutes)]

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

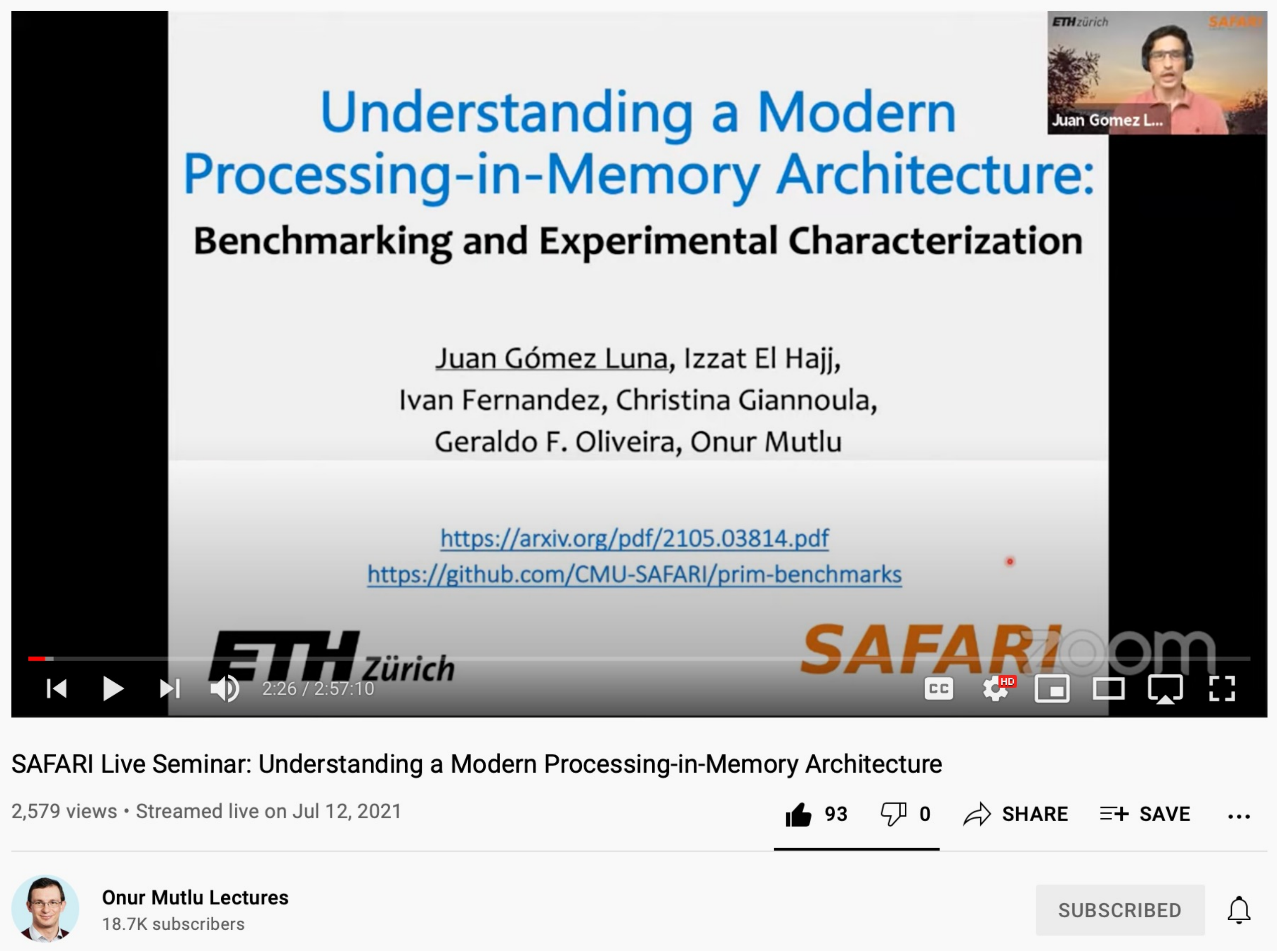
Juan Gómez-Luna[1] Izzat El Hajj[2] Ivan Fernandez[1,3] Christina Giannoula[1,4]
Geraldo F. Oliveira[1] Onur Mutlu[1]

[1]ETH Zürich [2]American University of Beirut [3]University of Malaga [4]National Technical University of Athens

https://arxiv.org/pdf/2105.03814.pdf

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

Understanding a Modern PIM Architecture



Understanding a Modern Processing-in-Memory Architecture:

Benchmarking and Experimental Characterization

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

https://arxiv.org/pdf/2105.03814.pdf
https://github.com/CMU-SAFARI/prim-benchmarks

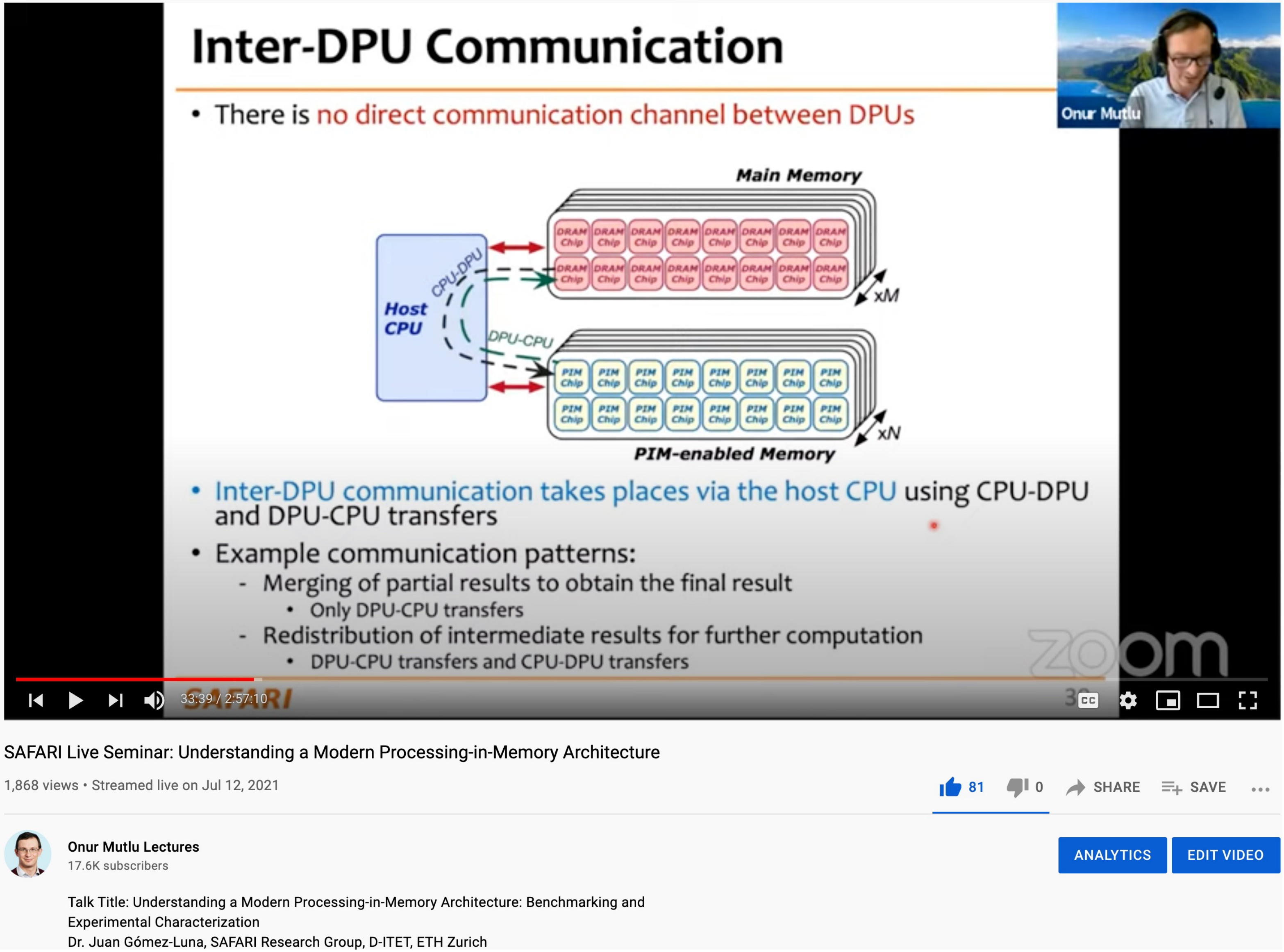
SAFARI Live Seminar: Understanding a Modern Processing-in-Memory Architecture

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

Onur Mutlu Lectures
18.7K subscribers

https://www.youtube.com/watch?v=D8Hjy2iU9l4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9

More on Analysis of the UPMEM PIM Engine



Inter-DPU Communication

- There is no direct communication channel between DPUs

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

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

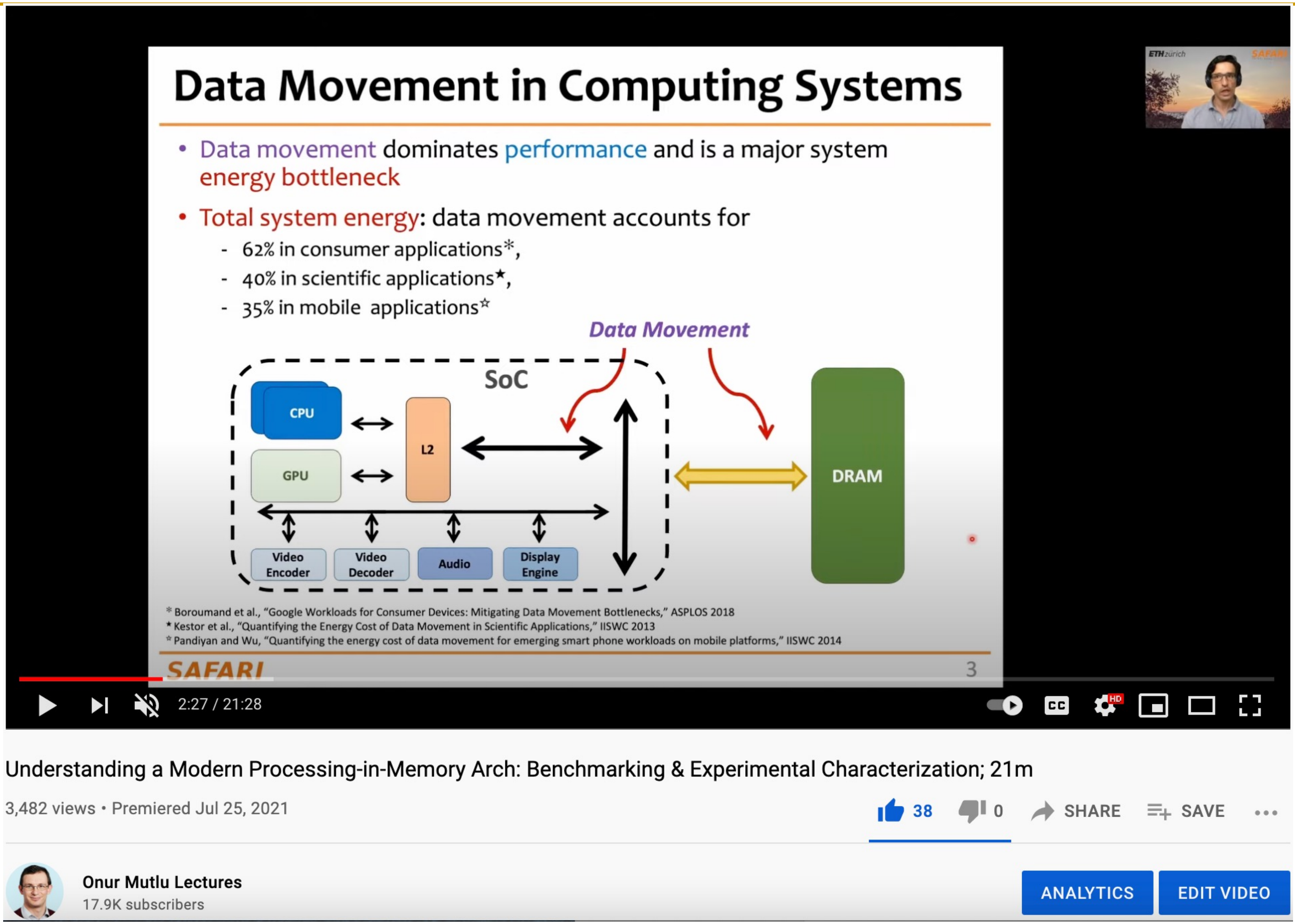
1,868 views • Streamed live on Jul 12, 2021

Onur Mutlu Lectures
17.6K subscribers

Talk Title: Understanding a Modern Processing-in-Memory Architecture: Benchmarking and Experimental Characterization
Dr. Juan Gómez-Luna, SAFARI Research Group, D-ITET, ETH Zurich

https://www.youtube.com/watch?v=D8Hjy2iU9l4&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9

More on Analysis of the UPMEM PIM Engine



Data Movement in Computing Systems

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

* Boroumand et al., "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," ASPLOS 2018

★ Kestor et al., "Quantifying the Energy Cost of Data Movement in Scientific Applications," IISWC 2013

☆ Pandiyan and Wu, "Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms," IISWC 2014

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

3,482 views • Premiered Jul 25, 2021

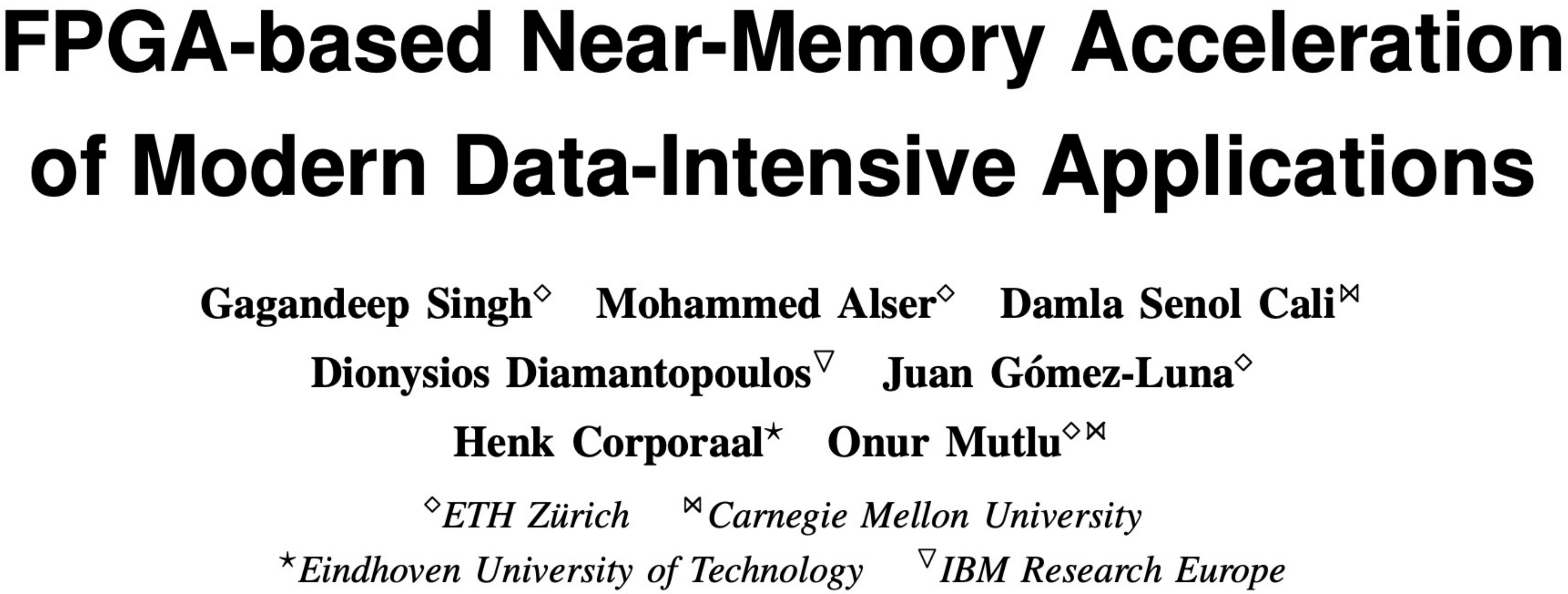
Onur Mutlu Lectures

17.9K subscribers

https://www.youtube.com/watch?v=Pp9jSU2b9oM&list=PL5Q2soXY2Zi8_VVChACnON4sfh2bJ5IrD&index=159

FPGA-based Processing Near Memory

- Gagandeep Singh, Mohammed Alser, Damla Senol Cali, Dionysios Diamantopoulos, Juan Gómez-Luna, Henk Corporaal, and Onur Mutlu, **"FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications"** *IEEE Micro* ***(IEEE MICRO)***, to appear, 2021.

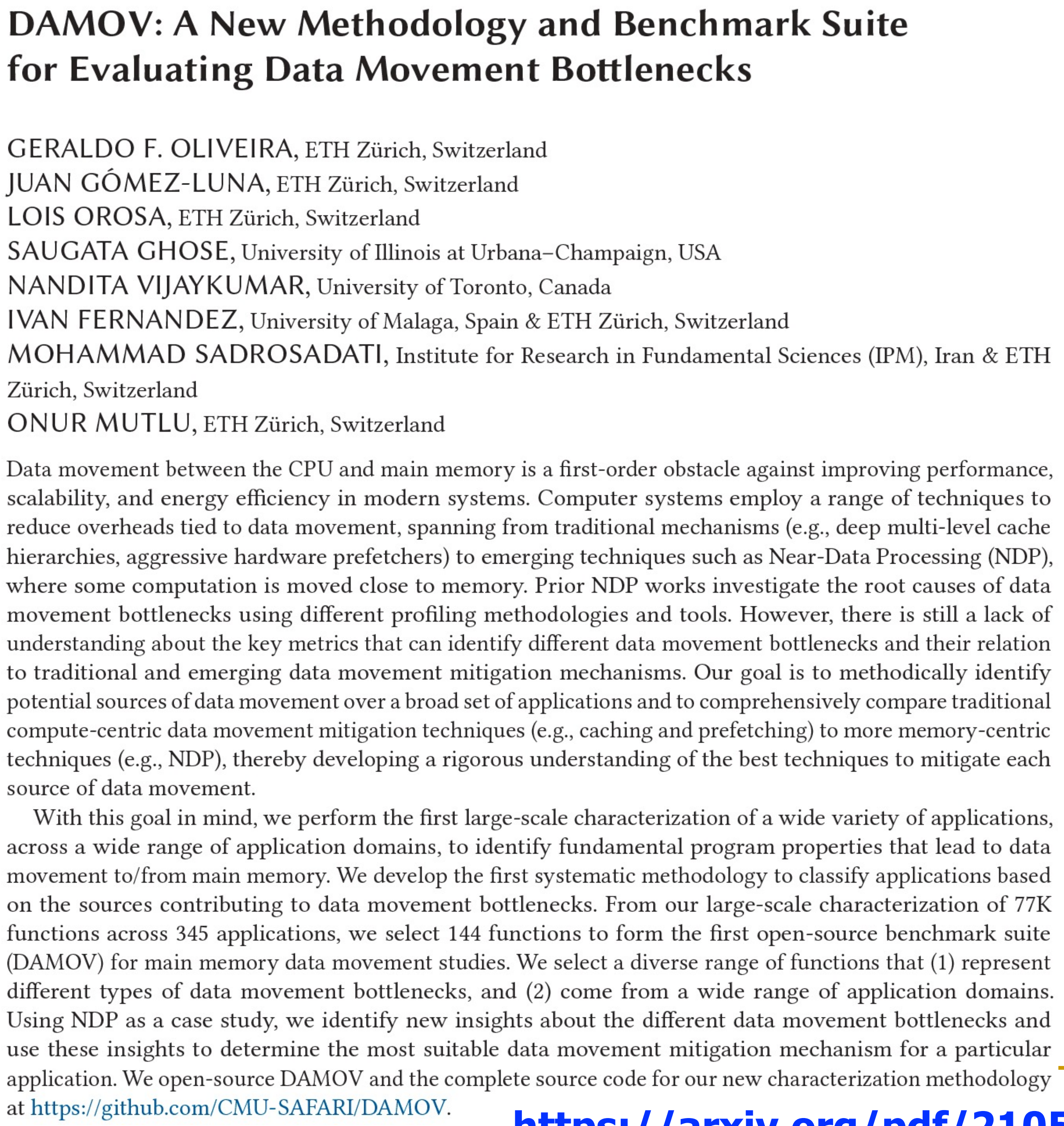


FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh[◇] **Mohammed Alser**[◇] **Damla Senol Cali**[⋈]
Dionysios Diamantopoulos[▽] **Juan Gómez-Luna**[◇]
Henk Corporaal[⋆] **Onur Mutlu**[◇⋈]

[◇]*ETH Zürich* [⋈]*Carnegie Mellon University*
[⋆]*Eindhoven University of Technology* [▽]*IBM Research Europe*

DAMOV Analysis Methodology & Workloads



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

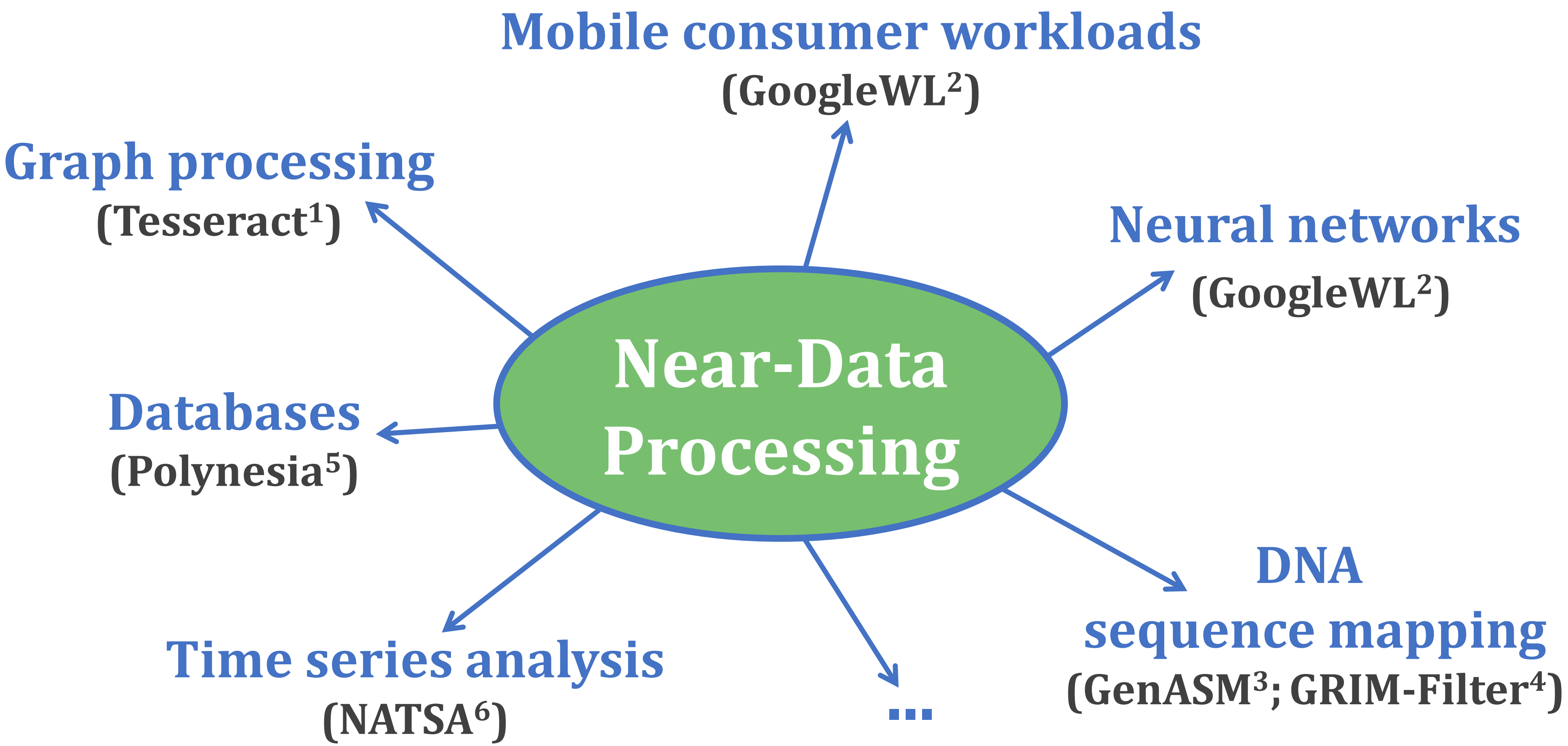
GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland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

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

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

https://arxiv.org/pdf/2105.03725.pdf

When to Employ Near-Data Processing?



Near-Data Processing

Mobile consumer workloads (GoogleWL[2])

Graph processing (Tesseract[1])

Neural networks (GoogleWL[2])

Databases (Polynesia[5])

DNA sequence mapping (GenASM[3]; GRIM-Filter[4])

Time series analysis (NATSA[6])

...

[1] Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing," ISCA, 2015

[2] Boroumand+, "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," ASPLOS, 2018

[3] Cali+, "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis," MICRO, 2020

[4] Kim+, "GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies," BMC Genomics, 2018

[5] Boroumand+, "Polynesia: Enabling Effective Hybrid Transactional/Analytical Databases with Specialized Hardware/Software Co-Design," arXiv:2103.00798 [cs.AR], 2021

[6] Fernandez+, "NATSA: A Near-Data Processing Accelerator for Time Series Analysis," ICCD, 2020

Key Approach

- New workload characterization methodology to analyze:
  - data movement bottlenecks
  - suitability of different data movement mitigation mechanisms

- Two main profiling strategies:

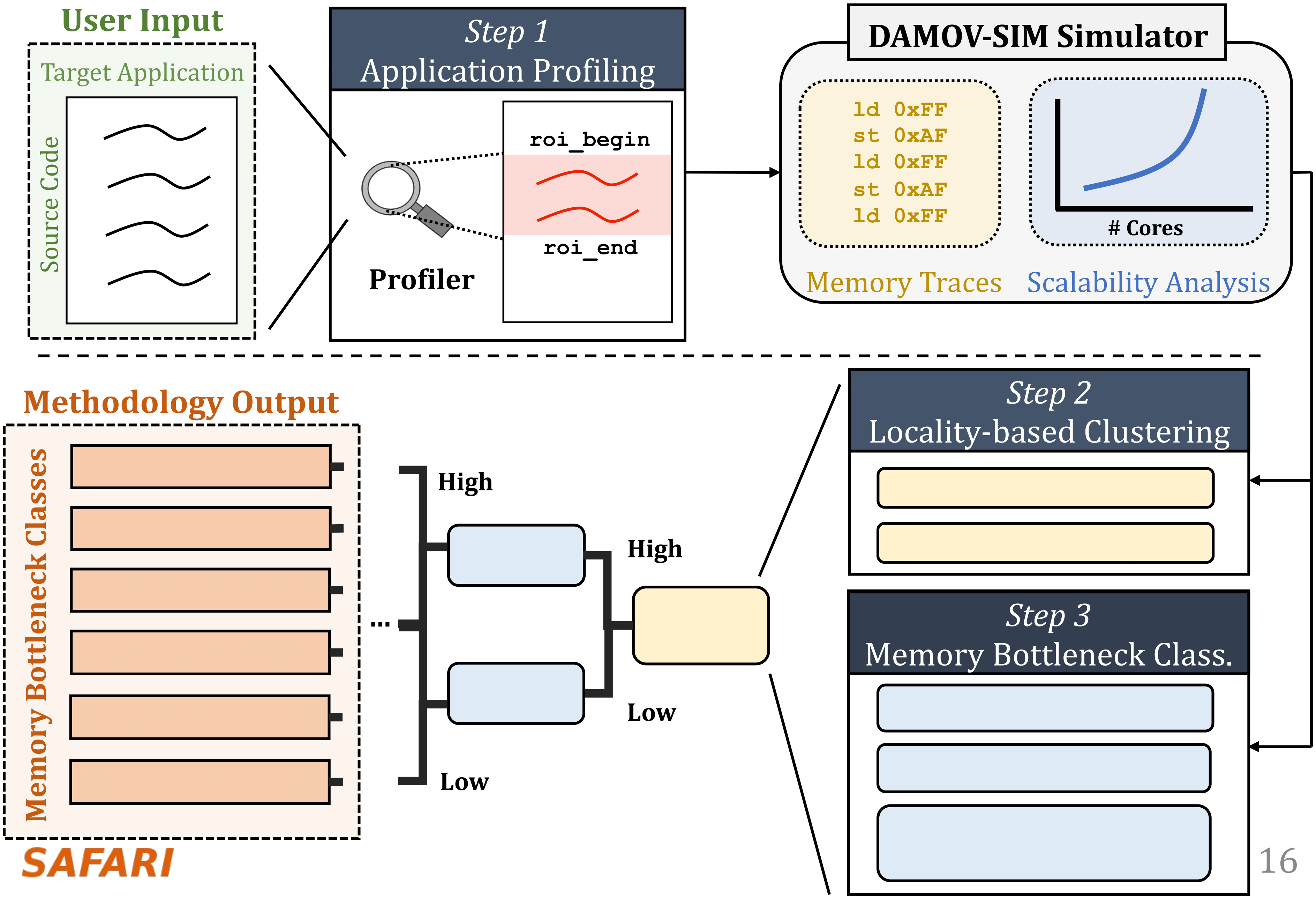
Architecture-independent profiling:

characterizes the memory behavior independently of the underlying hardware

Architecture-dependent profiling:

evaluates the impact of the system configuration on the memory behavior

Methodology Overview



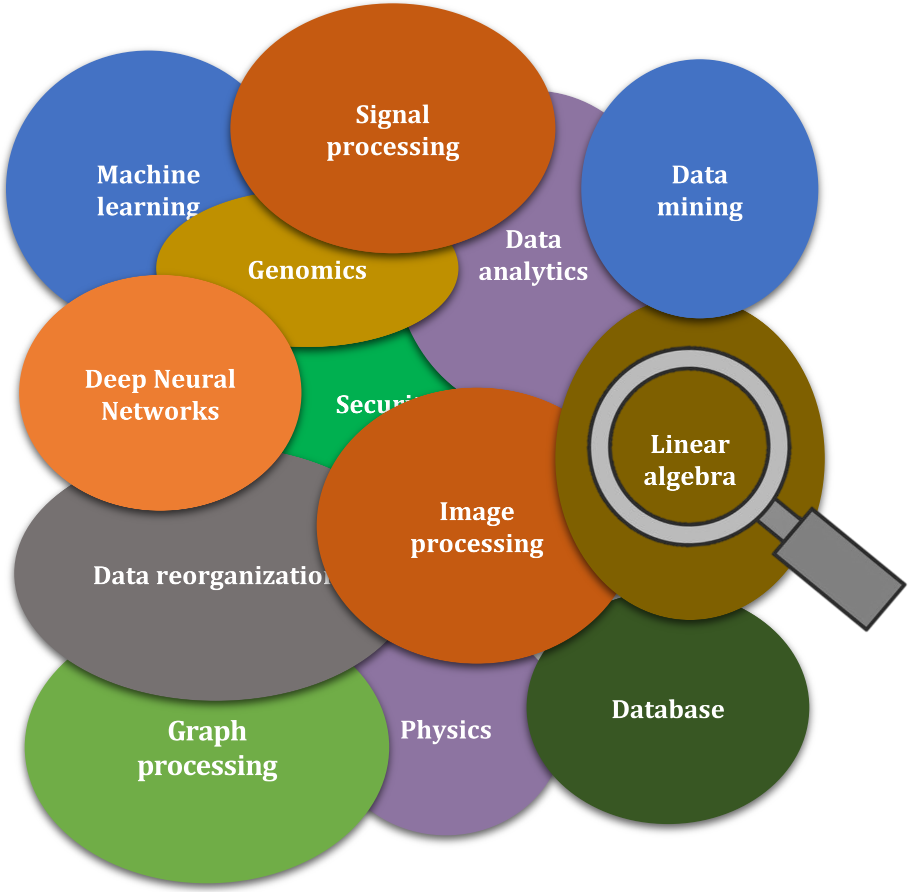
```
ld 0xFF
st 0xAF
ld 0xFF
st 0xAF
ld 0xFF
```

Step 1: Application Profiling

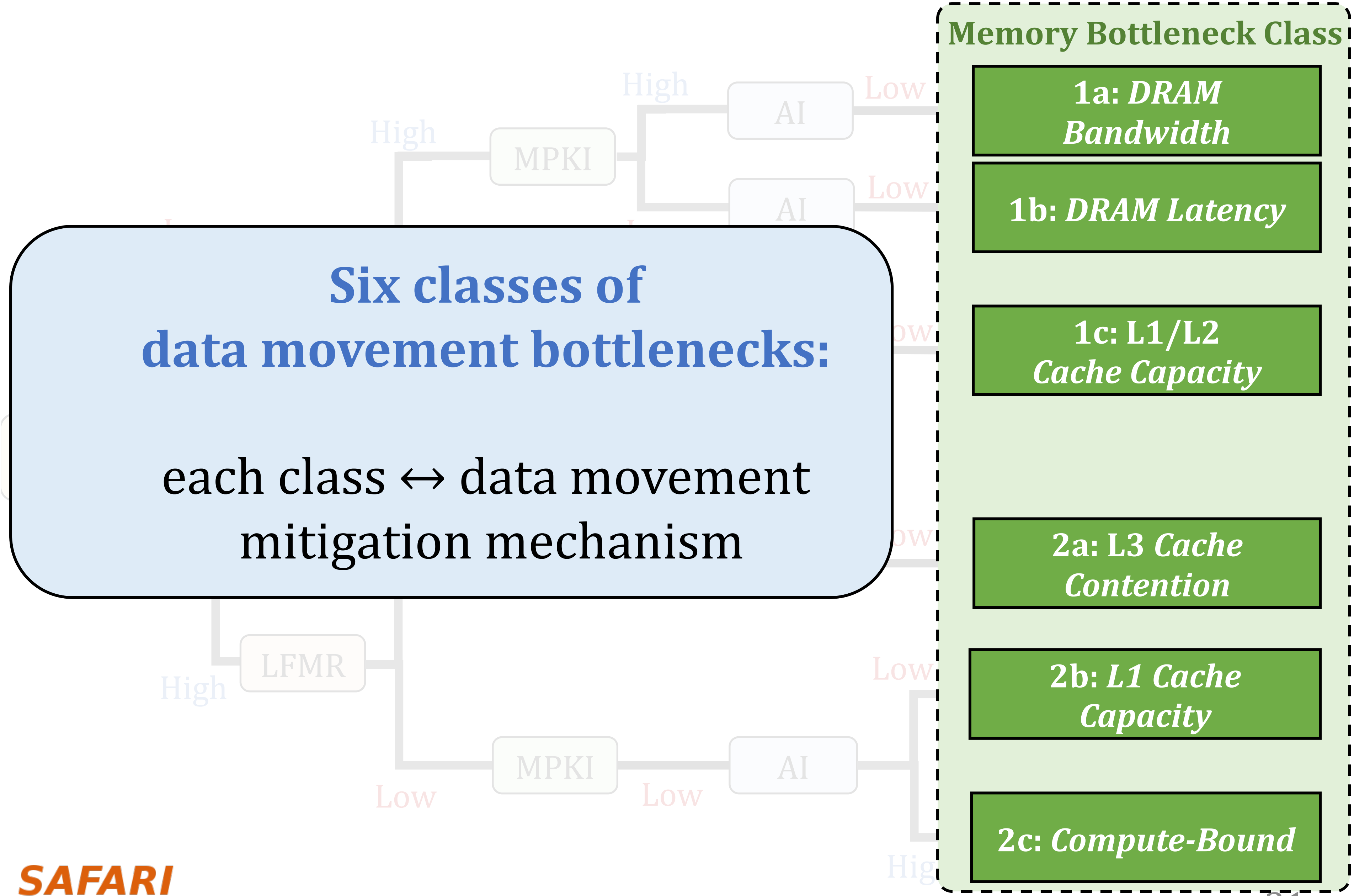
- We analyze 345 applications from distinct domains:

- Graph Processing
- Deep Neural Networks
- Physics
- High-Performance Computing
- Genomics
- Machine Learning
- Databases
- Data Reorganization
- Image Processing
- Map-Reduce
- Benchmarking
- Linear Algebra

…



Step 3: Memory Bottleneck Analysis



Memory Bottleneck Class

1a: *DRAM Bandwidth*

1b: *DRAM Latency*

1c: L1/L2 *Cache Capacity*

2a: L3 *Cache Contention*

2b: *L1 Cache Capacity*

2c: *Compute-Bound*

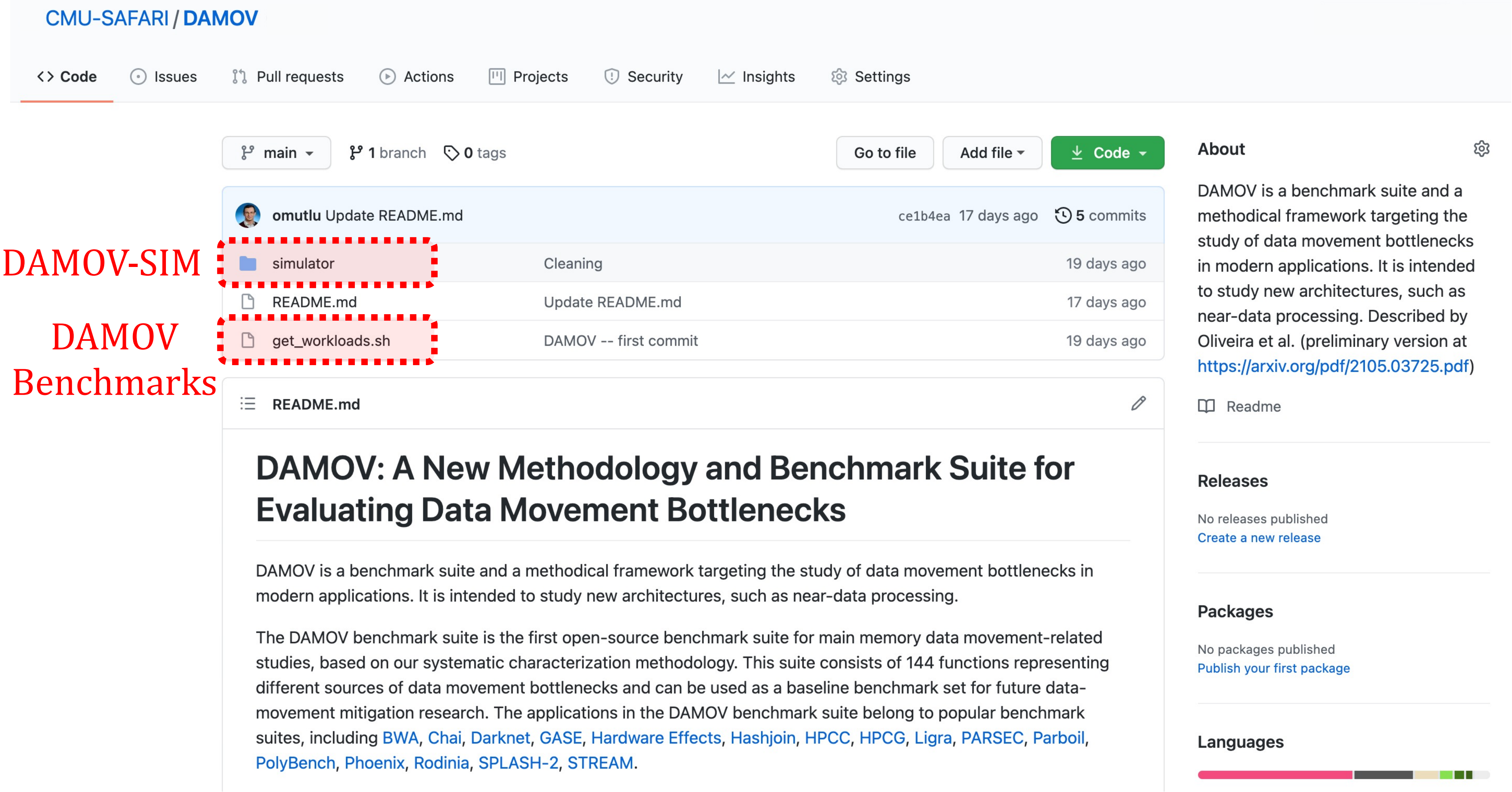
Six classes of data movement bottlenecks:

each class ↔ data movement mitigation mechanism

SAFARI

DAMOV is Open Source

- We open-source our benchmark suite and our toolchain



CMU-SAFARI / **DAMOV**

omutlu Update README.md ce1b4ea 17 days ago 5 commits

simulator	Cleaning	19 days ago
README.md	Update README.md	17 days ago
get_workloads.sh	DAMOV -- first commit	19 days ago

README.md

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

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

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

About

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

Readme

Releases

No releases published
Create a new release

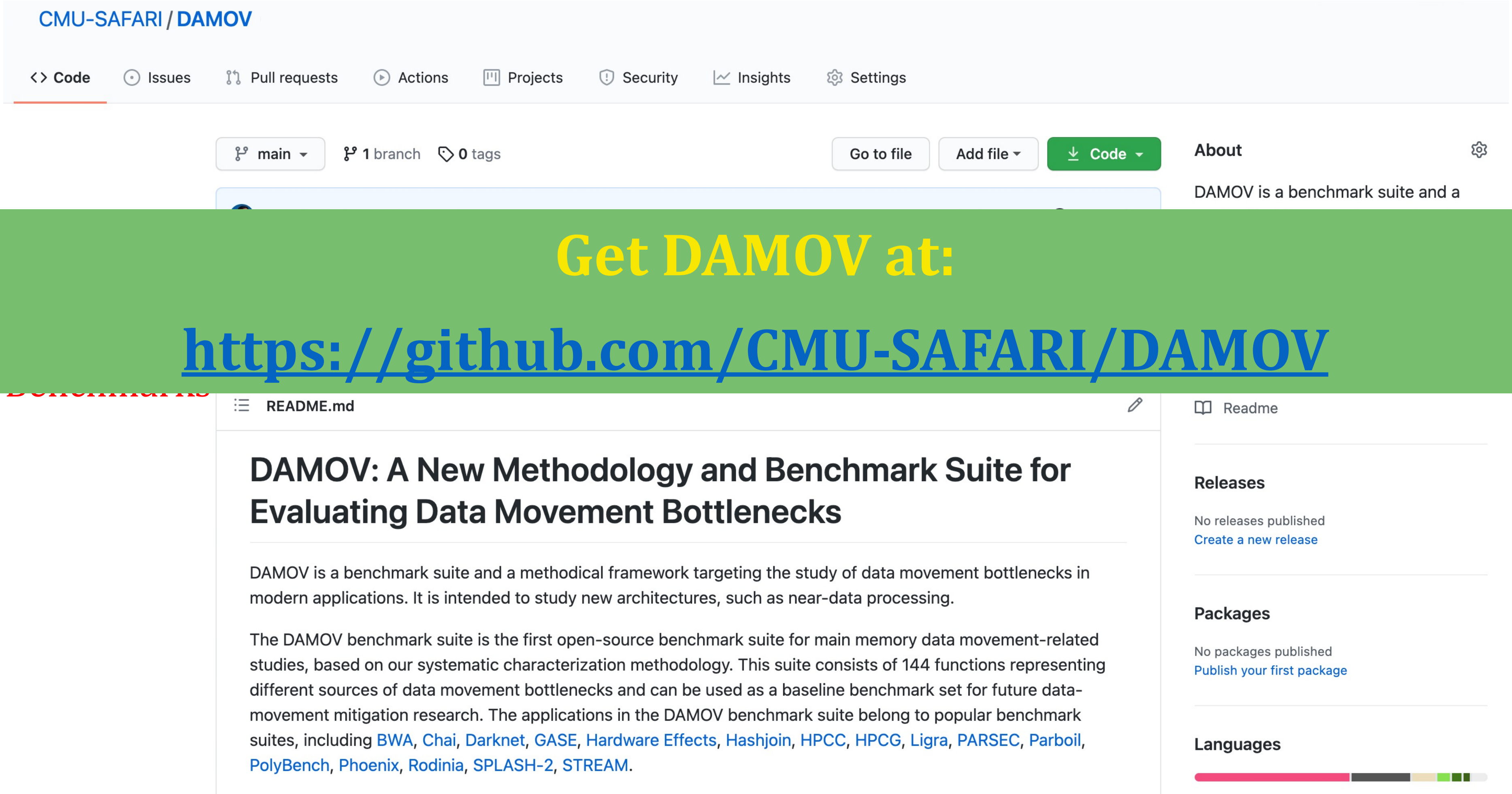
Packages

No packages published
Publish your first package

Languages

DAMOV is Open Source

- We open-source our benchmark suite and our toolchain



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

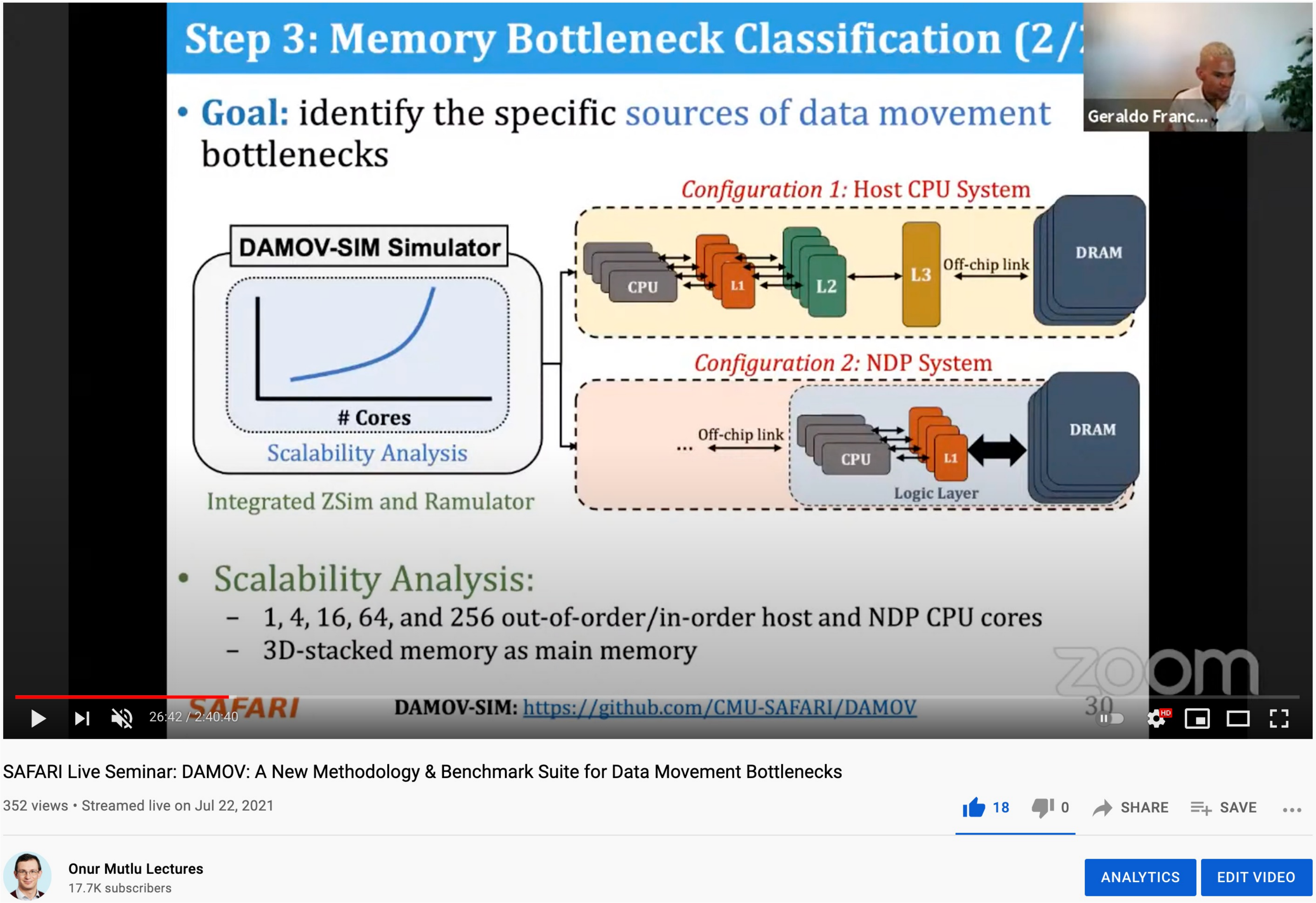
DAMOV is a benchmark suite and a methodical framework targeting the study of data movement bottlenecks in modern applications. It is intended to study new architectures, such as near-data processing.

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

Get DAMOV at:

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

More on DAMOV Analysis Methodology & Workloads



Step 3: Memory Bottleneck Classification (2/

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

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

SAFARI

DAMOV-SIM: https://github.com/CMU-SAFARI/DAMOV

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

352 views • Streamed live on Jul 22, 2021

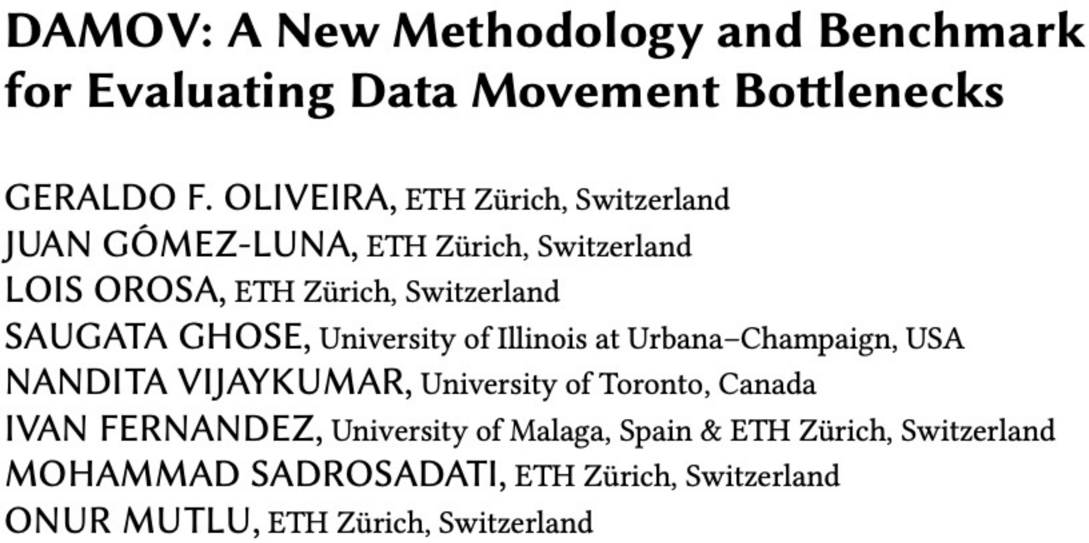
Onur Mutlu Lectures

17.7K subscribers

https://www.youtube.com/watch?v=GWideVyo0nM&list=PL5Q2soXY2Zi_tOTAYm--dYByNPL7JhwR9&index=3

More on DAMOV

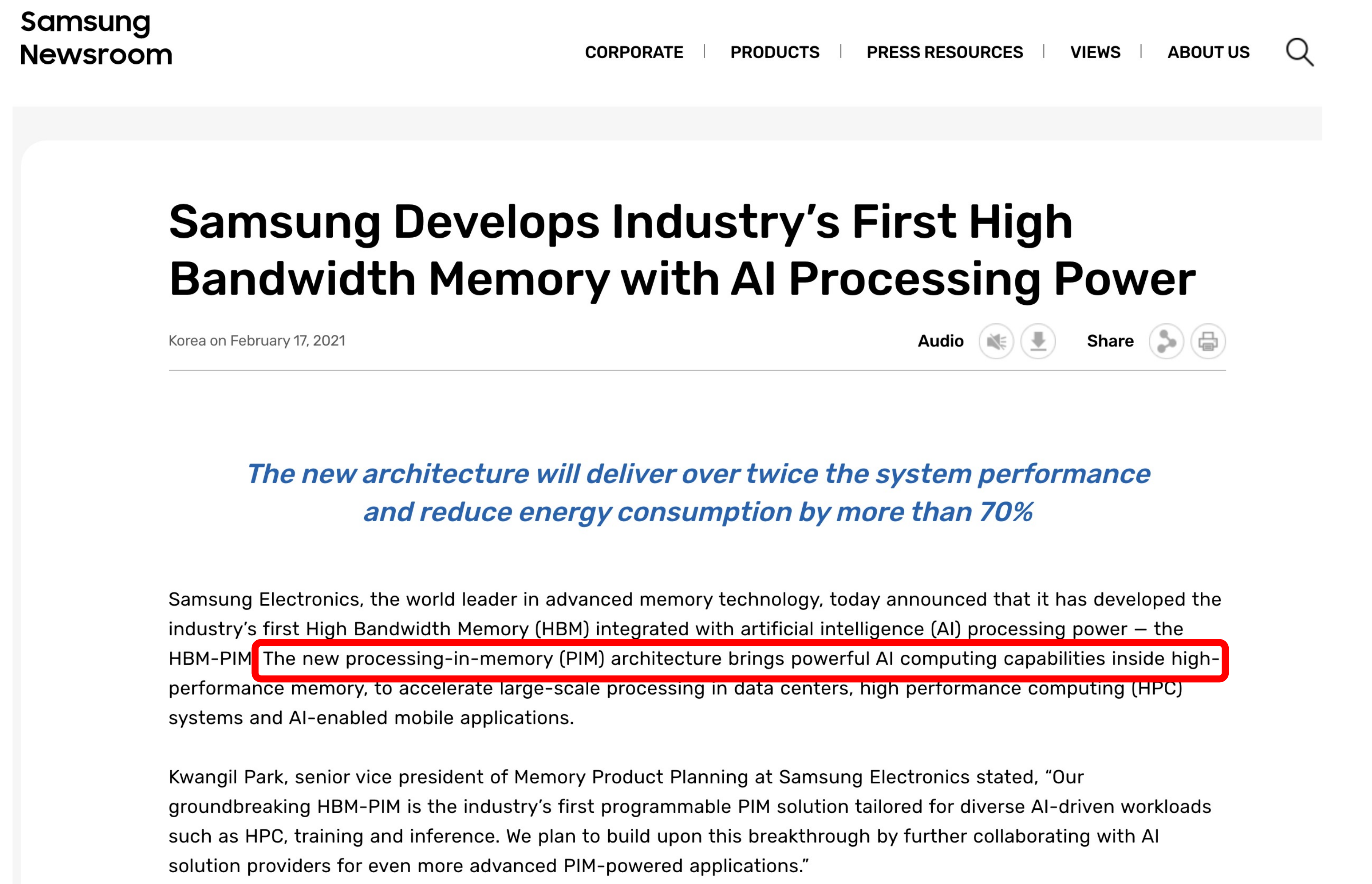
- Geraldo F. Oliveira, Juan Gomez-Luna, Lois Orosa, Saugata Ghose, Nandita Vijaykumar, Ivan fernandez, Mohammad Sadrosadati, and Onur Mutlu,
  "DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks"
  *Preprint in **arXiv***, 8 May 2021.
  [arXiv preprint]
  [DAMOV Suite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



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

GERALDO F. OLIVEIRA, ETH Zürich, Switzerland
JUAN GÓMEZ-LUNA, ETH Zürich, Switzerland
LOIS OROSA, ETH Zürich, Switzerland
SAUGATA GHOSE, University of Illinois at Urbana–Champaign, USA
NANDITA VIJAYKUMAR, University of Toronto, Canada
IVAN FERNANDEZ, University of Malaga, Spain & ETH Zürich, Switzerland
MOHAMMAD SADROSADATI, ETH Zürich, Switzerland
ONUR MUTLU, ETH Zürich, Switzerland

Samsung Function-in-Memory DRAM (2021)



Samsung Develops Industry's First High Bandwidth Memory with AI Processing Power

Korea on February 17, 2021

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

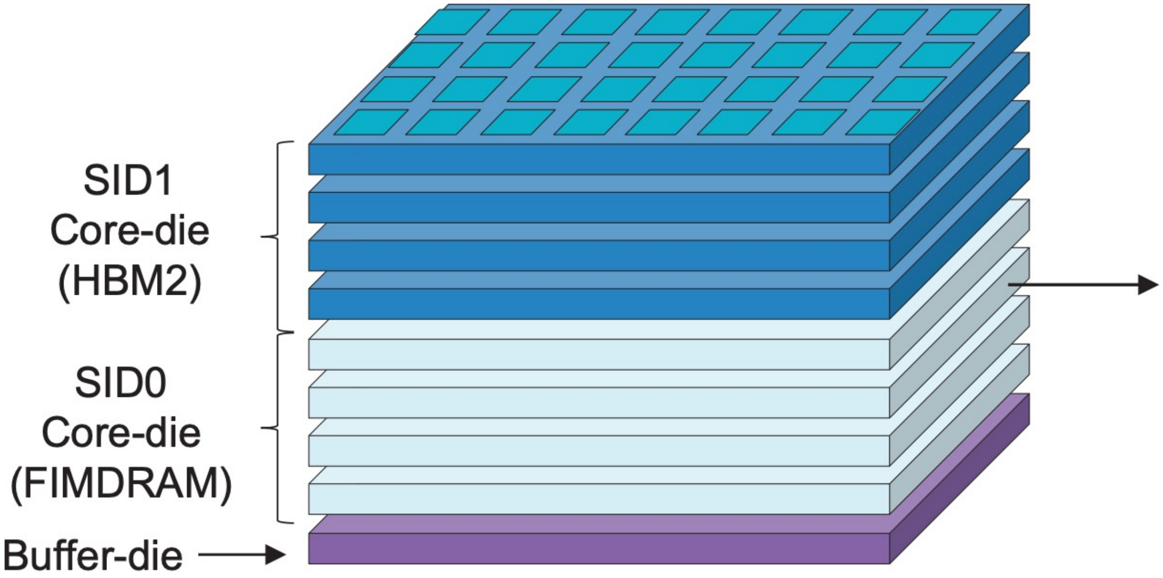
Samsung Electronics, the world leader in advanced memory technology, today announced that it has developed the industry's first High Bandwidth Memory (HBM) integrated with artificial intelligence (AI) processing power — the HBM-PIM The new processing-in-memory (PIM) architecture brings powerful AI computing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

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

https://news.samsung.com/global/samsung-develops-industrys-first-high-bandwidth-memory-with-ai-processing-power

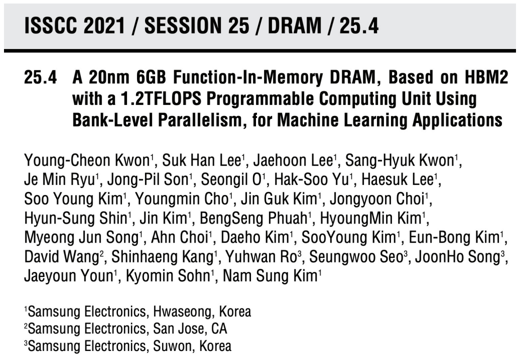
Samsung Function-in-Memory DRAM (2021)

- FIMDRAM based on HBM2



[3D Chip Structure of HBM with FIMDRAM]

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



ISSCC 2021 / SESSION 25 / DRAM / 25.4

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

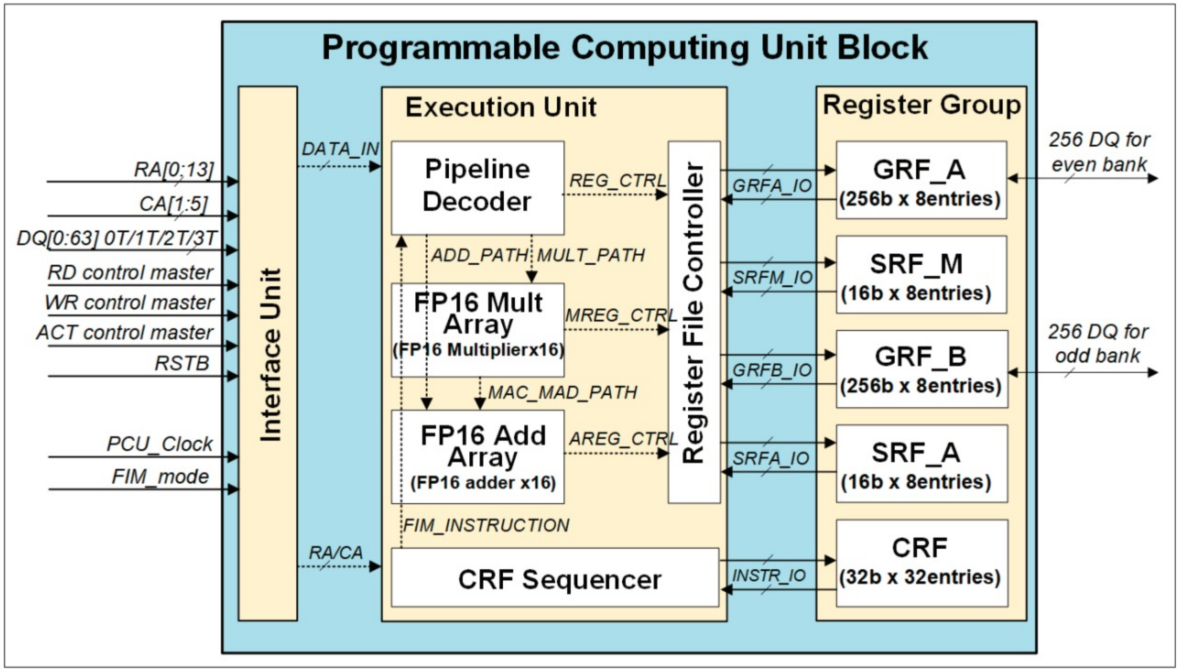
Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1],
Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1],
Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1],
Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1],
Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1],
David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3],
Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

[1]Samsung Electronics, Hwaseong, Korea
[2]Samsung Electronics, San Jose, CA
[3]Samsung Electronics, Suwon, Korea

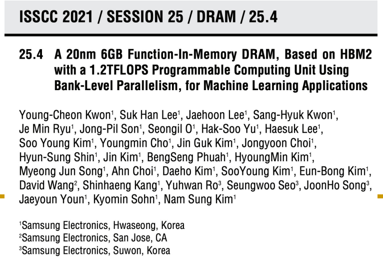
Samsung Function-in-Memory DRAM (2021)

Programmable Computing Unit

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



[Block diagram of PCU in FIMDRAM]



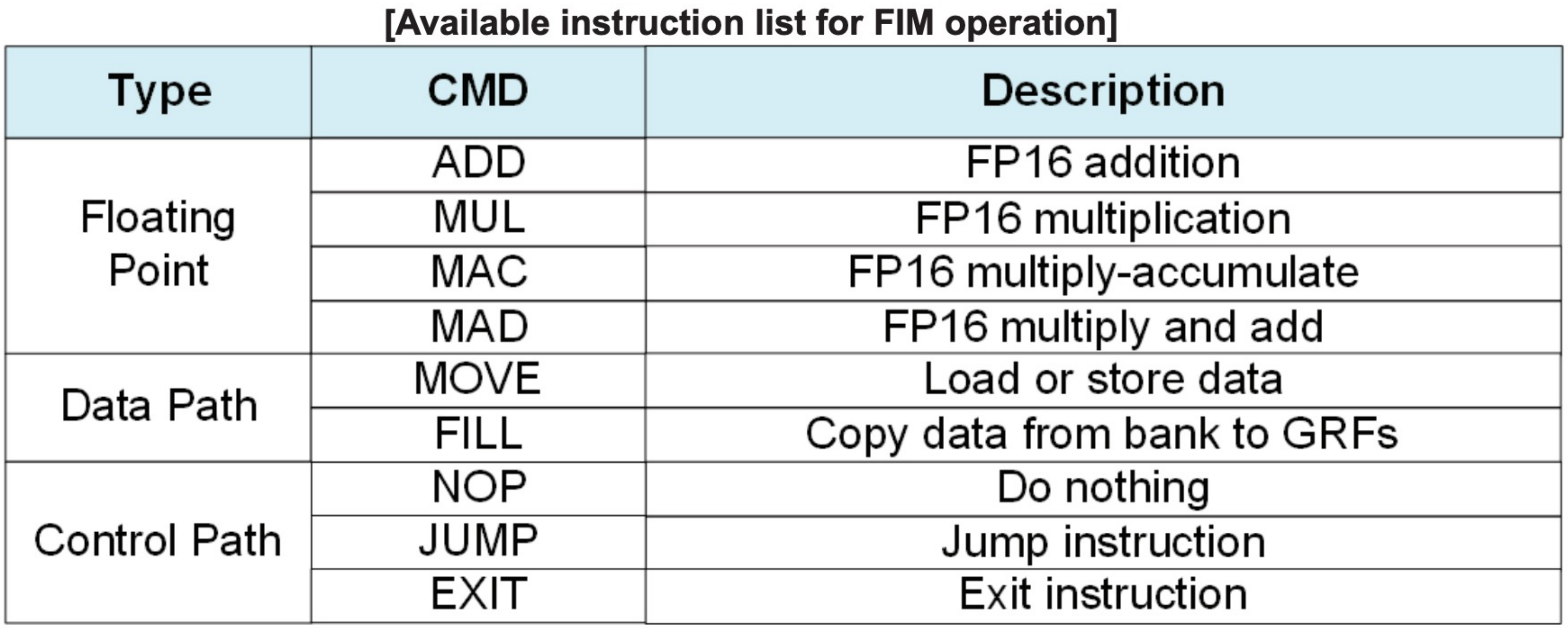
ISSCC 2021 / SESSION 25 / DRAM / 25.4

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

Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1], Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1], Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1], Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1], Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1], David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3], Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

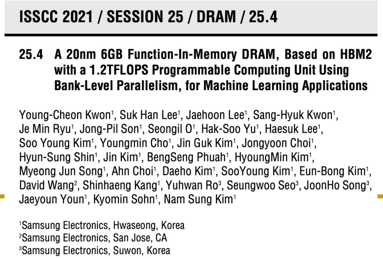
[1]Samsung Electronics, Hwaseong, Korea
[2]Samsung Electronics, San Jose, CA
[3]Samsung Electronics, Suwon, Korea

Samsung Function-in-Memory DRAM (2021)



[Available instruction list for FIM operation]

Type	CMD	Description
Floating Point	ADD	FP16 addition
	MUL	FP16 multiplication
	MAC	FP16 multiply-accumulate
	MAD	FP16 multiply and add
Data Path	MOVE	Load or store data
	FILL	Copy data from bank to GRFs
Control Path	NOP	Do nothing
	JUMP	Jump instruction
	EXIT	Exit instruction



ISSCC 2021 / SESSION 25 / DRAM / 25.4

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

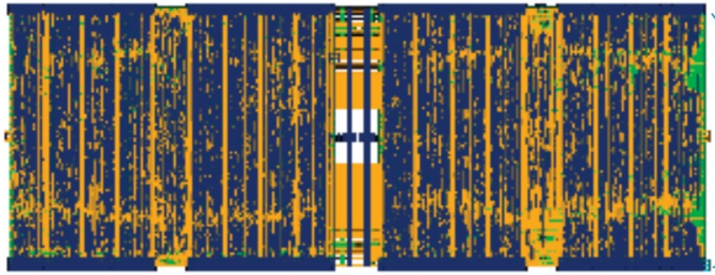
Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1],
Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1],
Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1],
Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1],
Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1],
David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3],
Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

[1]Samsung Electronics, Hwaseong, Korea
[2]Samsung Electronics, San Jose, CA
[3]Samsung Electronics, Suwon, Korea

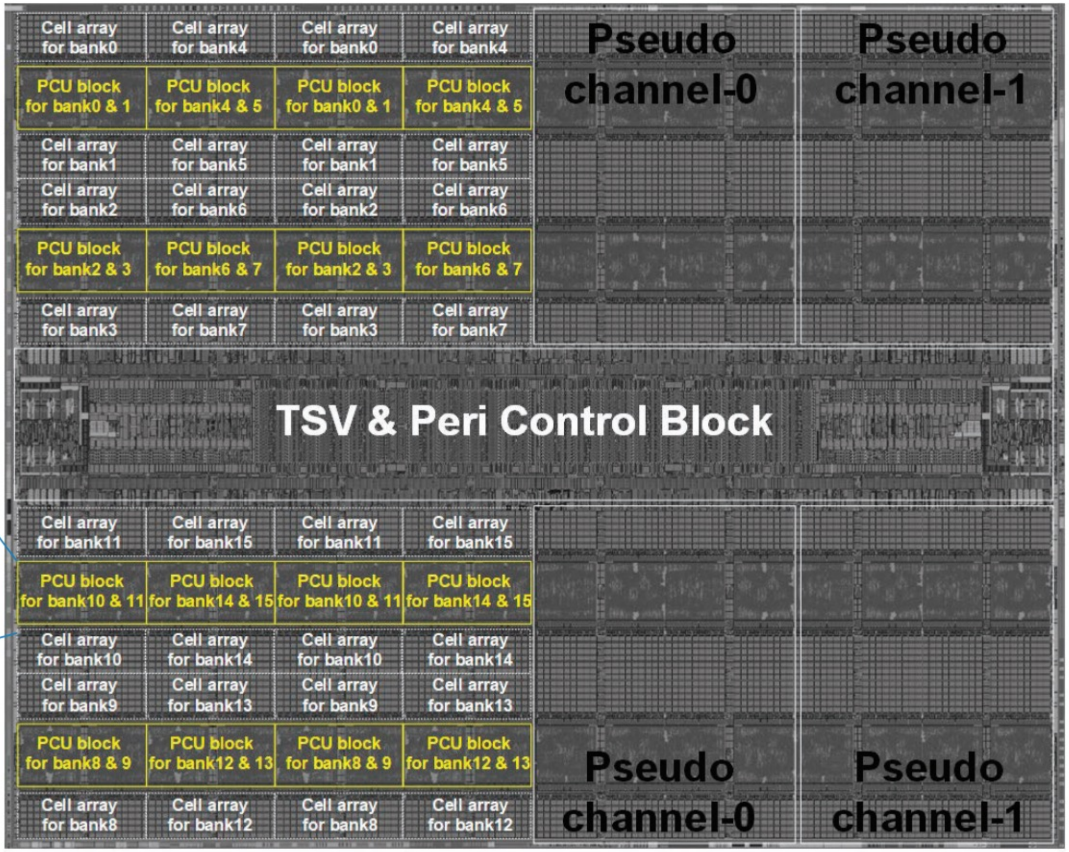
Samsung Function-in-Memory DRAM (2021)

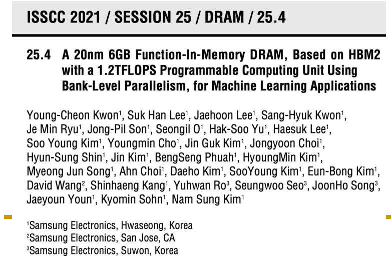
Chip Implementation

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



[Digital RTL design for PCU block]





ISSCC 2021 / SESSION 25 / DRAM / 25.4

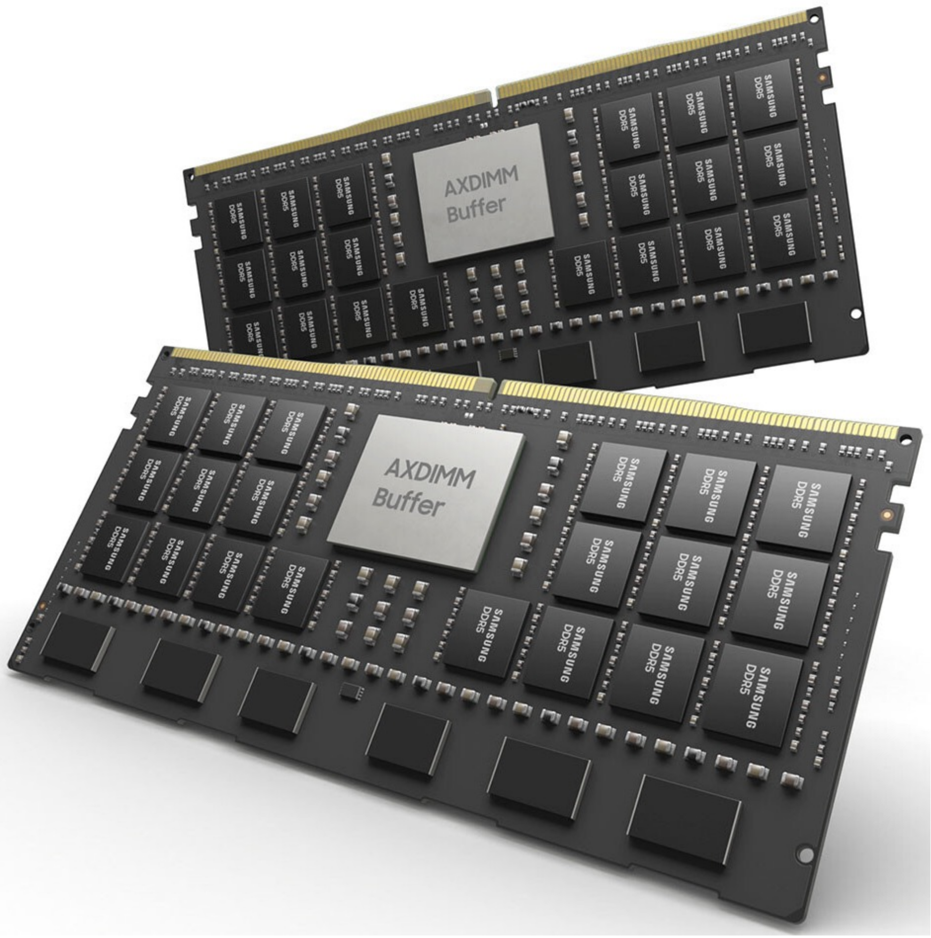
25.4 A 20nm 6GB Function-In-Memory DRAM, Based on HBM2 with a 1.2TFLOPS Programmable Computing Unit Using Bank-Level Parallelism, for Machine Learning Applications

Young-Cheon Kwon[1], Suk Han Lee[1], Jaehoon Lee[1], Sang-Hyuk Kwon[1], Je Min Ryu[1], Jong-Pil Son[1], Seongil O[1], Hak-Soo Yu[1], Haesuk Lee[1], Soo Young Kim[1], Youngmin Cho[1], Jin Guk Kim[1], Jongyoon Choi[1], Hyun-Sung Shin[1], Jin Kim[1], BengSeng Phuah[1], HyoungMin Kim[1], Myeong Jun Song[1], Ahn Choi[1], Daeho Kim[1], SooYoung Kim[1], Eun-Bong Kim[1], David Wang[2], Shinhaeng Kang[1], Yuhwan Ro[3], Seungwoo Seo[3], JoonHo Song[3], Jaeyoun Youn[1], Kyomin Sohn[1], Nam Sung Kim[1]

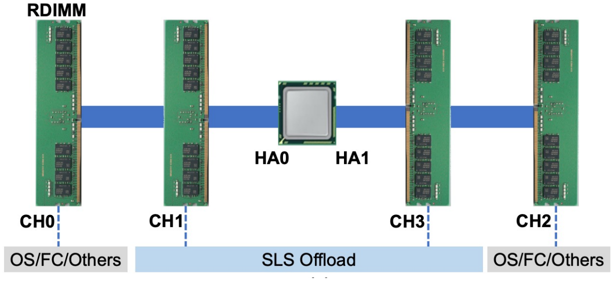
[1]Samsung Electronics, Hwaseong, Korea
[2]Samsung Electronics, San Jose, CA
[3]Samsung Electronics, Suwon, Korea

Samsung AxDIMM (2021)

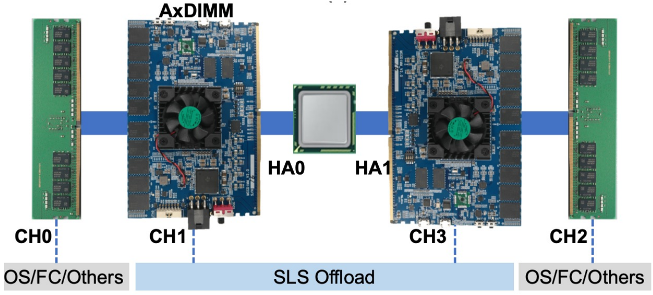
- DDR5-PIM
  - DLRM recommendation system

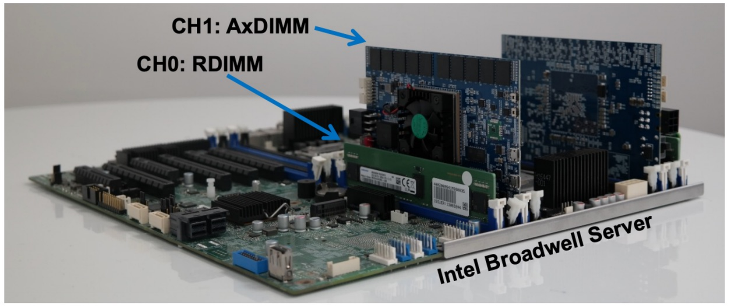


Baseline System



AxDIMM System





Ke et al. "Near-Memory Processing in Action: Accelerating Personalized Recommendation with AxDIMM", IEEE Micro (2021)

Detailed Lectures on PIM (I)

- Computer Architecture, Fall 2020, Lecture 6
  - **Computation in Memory** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=oGcZAGwfEUE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=12
- Computer Architecture, Fall 2020, Lecture 7
  - **Near-Data Processing** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=j2GIigqn1Qw&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=13
- Computer Architecture, Fall 2020, Lecture 11a
  - **Memory Controllers** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=TeG773OgiMQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=20
- Computer Architecture, Fall 2020, Lecture 12d
  - **Real Processing-in-DRAM with UPMEM** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=Sscy1Wrr22A&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=25

https://www.youtube.com/onurmutlulectures

Detailed Lectures on PIM (II)

- Computer Architecture, Fall 2020, Lecture 15
  - **Emerging Memory Technologies** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=AlE1rD9G_YU&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=28

- Computer Architecture, Fall 2020, Lecture 16a
  - **Opportunities & Challenges of Emerging Memory Technologies** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=pmLszWGmMGQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=29

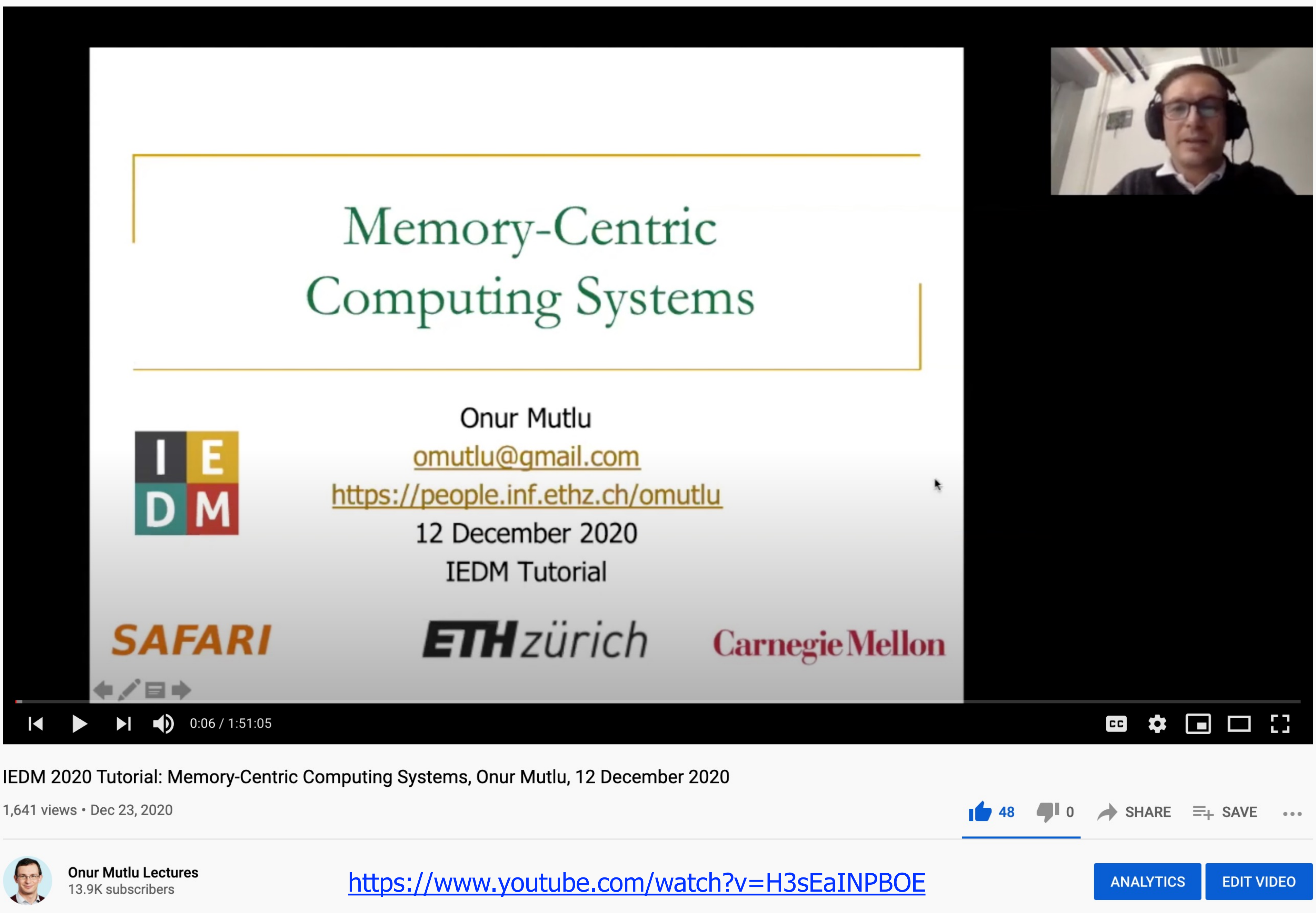
- Computer Architecture, Fall 2020, Guest Lecture
  - **In-Memory Computing: Memory Devices & Applications** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=wNmqQHiEZNk&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=41

https://www.youtube.com/onurmutlulectures

A Longer & Detailed Tutorial on PIM

- Onur Mutlu,
  "Memory-Centric Computing Systems"
  Invited Tutorial at *66th International Electron Devices Meeting (**IEDM**)*, Virtual, 12 December 2020.
  [Slides (pptx) (pdf)]
  [Executive Summary Slides (pptx) (pdf)]
  [Tutorial Video (1 hour 51 minutes)]
  [Executive Summary Video (2 minutes)]
  [Abstract and Bio]
  [Related Keynote Paper from VLSI-DAT 2020]
  [Related Review Paper on Processing in Memory]

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



Memory-Centric Computing Systems

Onur Mutlu

omutlu@gmail.com

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

12 December 2020

IEDM Tutorial

IEDM 2020 Tutorial: Memory-Centric Computing Systems, Onur Mutlu, 12 December 2020

1,641 views • Dec 23, 2020

Onur Mutlu Lectures

13.9K subscribers

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

https://www.youtube.com/onurmutlulectures

Challenge and Opportunity for Future

Fundamentally
Energy-Efficient
(Data-Centric)
Computing Architectures

Challenge and Opportunity for Future

Fundamentally
High-Performance
(Data-Centric)
Computing Architectures

Challenge and Opportunity for Future

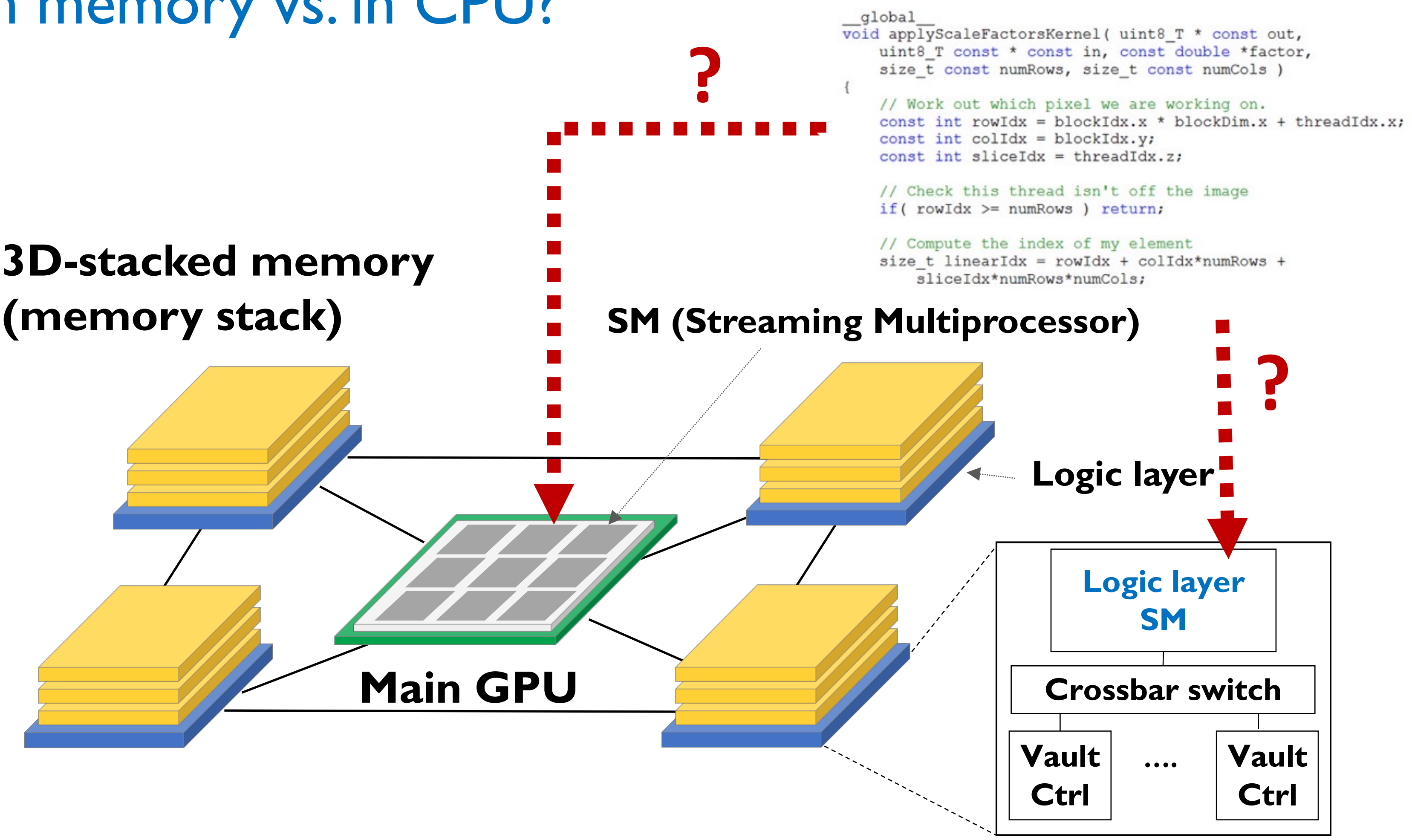
Computing Architectures

with

Minimal Data Movement

Key Challenge 1: Code Mapping

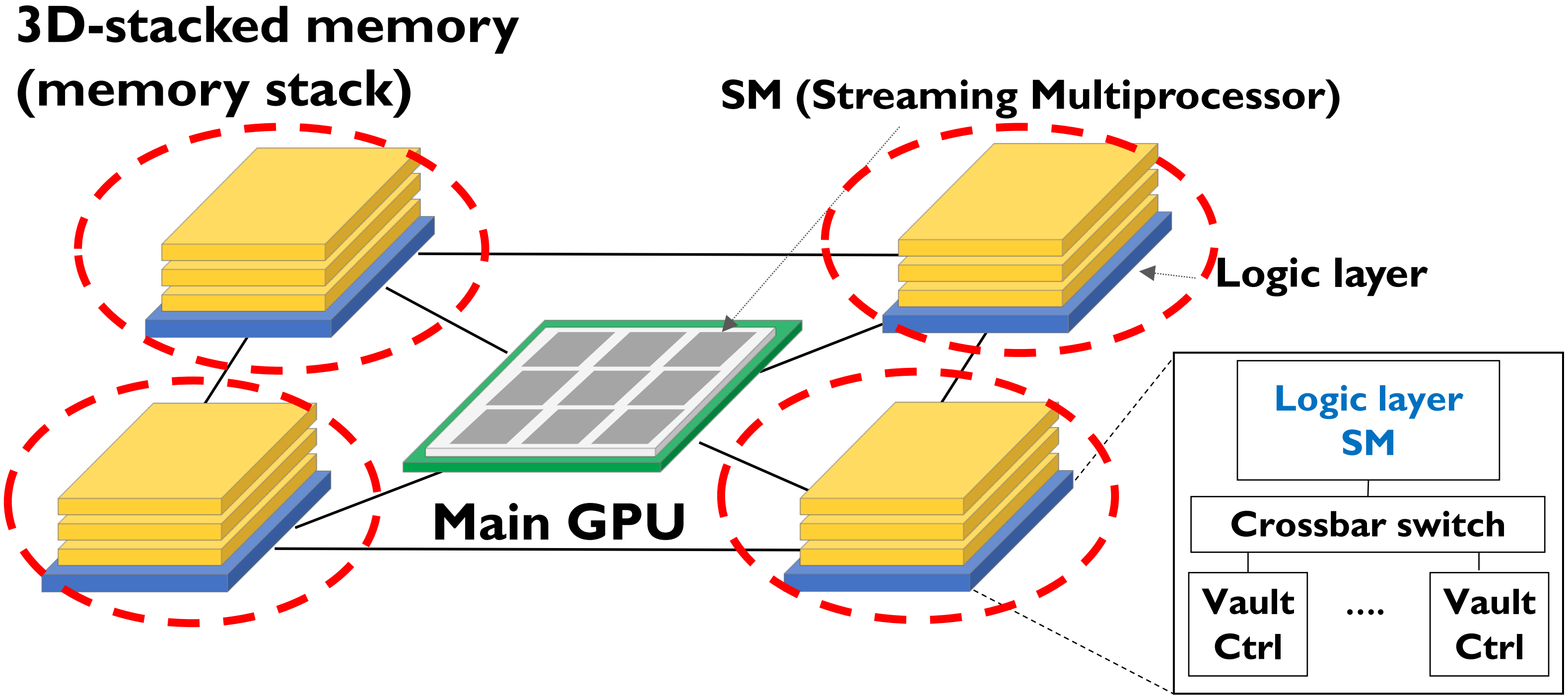
- **Challenge 1:** Which operations should be executed in memory vs. in CPU?



```
__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;
```

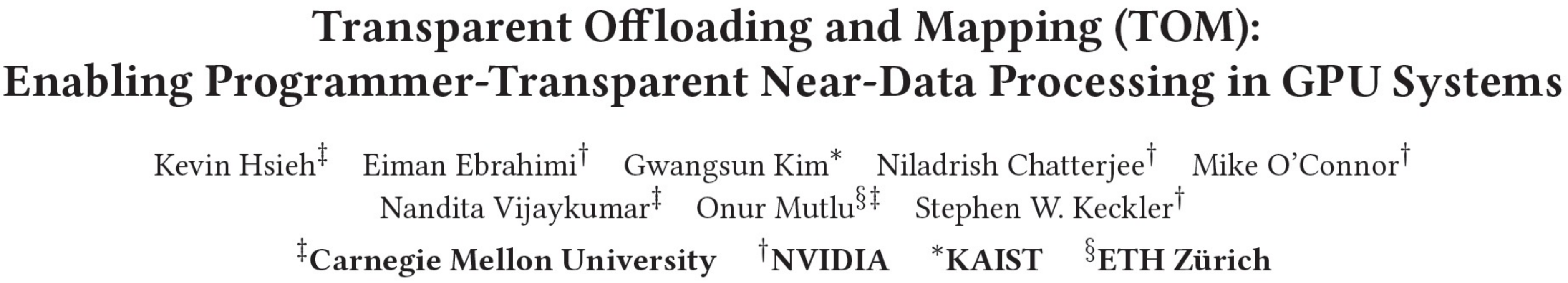
Key Challenge 2: Data Mapping

- **Challenge 2:** How should data be mapped to different 3D memory stacks?



How to Do the Code and Data Mapping?

- 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"**
  Proceedings of the 43rd International Symposium on Computer Architecture (***ISCA***), Seoul, South Korea, June 2016.
  [Slides (pptx) (pdf)]
  [Lightning Session Slides (pptx) (pdf)]

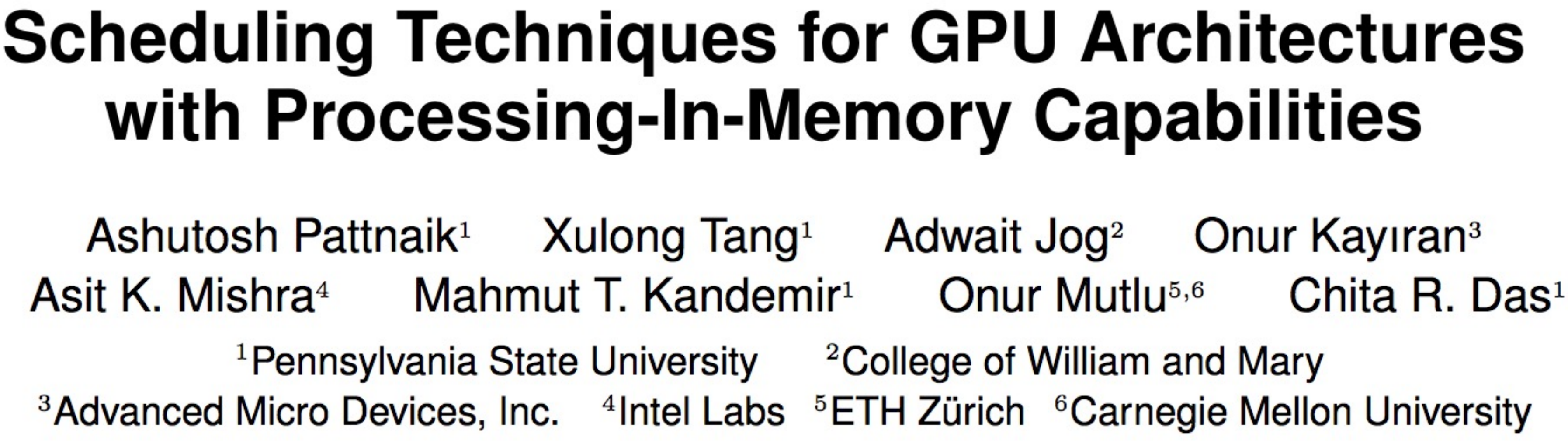


Transparent Offloading and Mapping (TOM):
Enabling Programmer-Transparent Near-Data Processing in GPU Systems

Kevin Hsieh[‡] Eiman Ebrahimi[†] Gwangsun Kim[*] Niladrish Chatterjee[†] Mike O'Connor[†]
Nandita Vijaykumar[‡] Onur Mutlu[§‡] Stephen W. Keckler[†]
[‡]**Carnegie Mellon University** [†]**NVIDIA** [*]**KAIST** [§]**ETH Zürich**

How to Schedule Code? (I)

- Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K. Mishra, Mahmut T. Kandemir, Onur Mutlu, and Chita R. Das, **"Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities"** *Proceedings of the 25th International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Haifa, Israel, September 2016.



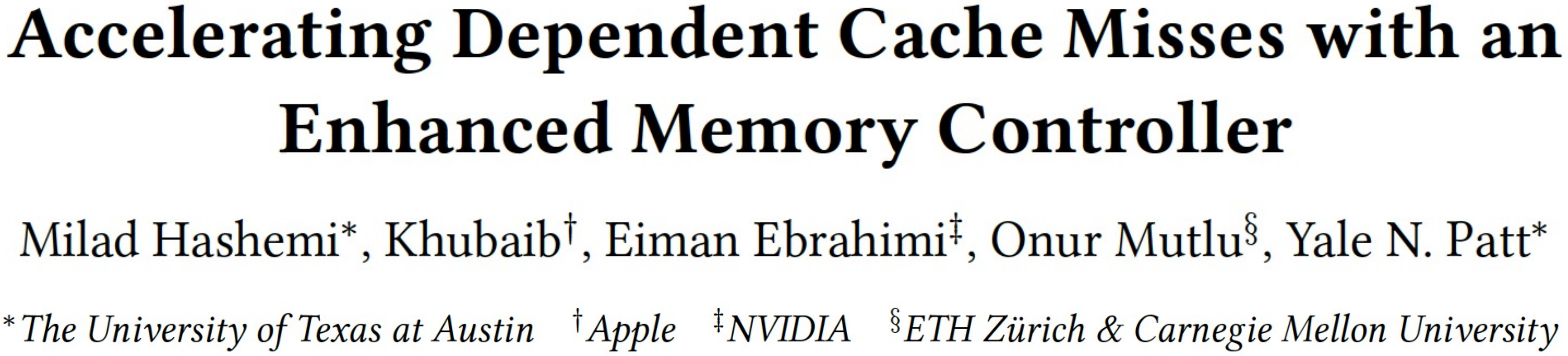
Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities

Ashutosh Pattnaik[1] Xulong Tang[1] Adwait Jog[2] Onur Kayıran[3]
Asit K. Mishra[4] Mahmut T. Kandemir[1] Onur Mutlu[5,6] Chita R. Das[1]

[1]Pennsylvania State University [2]College of William and Mary
[3]Advanced Micro Devices, Inc. [4]Intel Labs [5]ETH Zürich [6]Carnegie Mellon University

How to Schedule Code? (II)

- Milad Hashemi, Khubaib, Eiman Ebrahimi, Onur Mutlu, and Yale N. Patt, **"Accelerating Dependent Cache Misses with an Enhanced Memory Controller"**
  Proceedings of the 43rd International Symposium on Computer Architecture (***ISCA***), Seoul, South Korea, June 2016.
  [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*

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"
IEEE Computer Architecture Letters ***(CAL)***, June 2016.

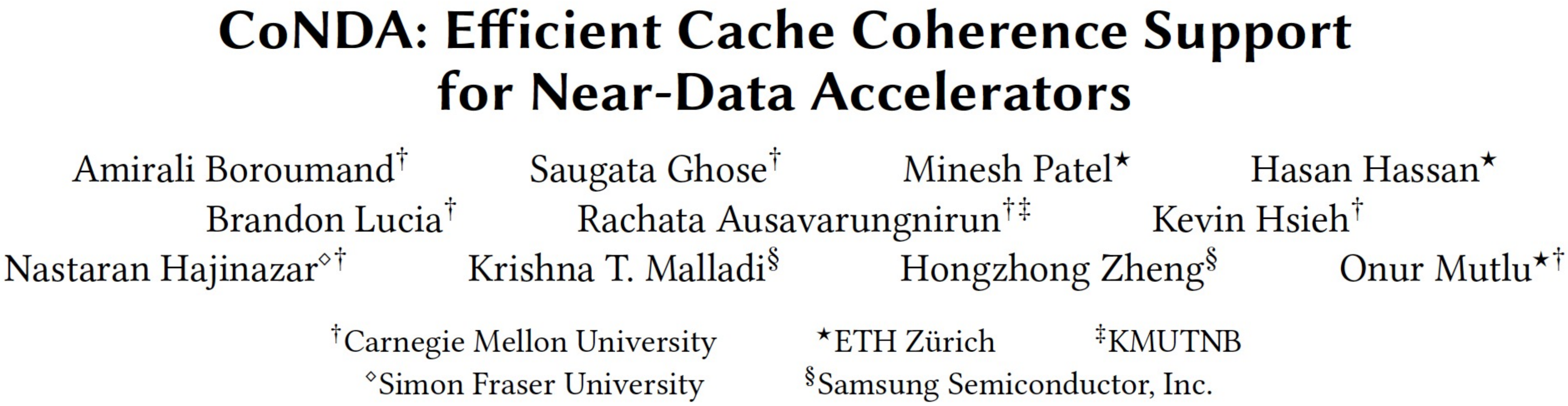
LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory

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"** *Proceedings of the 46th International Symposium on Computer Architecture* (***ISCA***), Phoenix, AZ, USA, June 2019.



CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators

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

[†]Carnegie Mellon University [⋆]ETH Zürich [‡]KMUTNB
[⋄]Simon Fraser University [§]Samsung Semiconductor, Inc.

How to Support Synchronization?

- Christina Giannoula, Nandita Vijaykumar, Nikela Papadopoulou, Vasileios Karakostas, Ivan Fernandez, Juan Gómez-Luna, Lois Orosa, Nectarios Koziris, Georgios Goumas, Onur Mutlu, **"SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures"**
  *Proceedings of the 27th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Virtual, February-March 2021.
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [Talk Video (21 minutes)]
  [Short Talk Video (7 minutes)]

SynCron: Efficient Synchronization Support for Near-Data-Processing Architectures

Christina Giannoula[†‡] Nandita Vijaykumar[*‡] Nikela Papadopoulou[†] Vasileios Karakostas[†] Ivan Fernandez[§‡]
Juan Gómez-Luna[‡] Lois Orosa[‡] Nectarios Koziris[†] Georgios Goumas[†] Onur Mutlu[‡]
[†]*National Technical University of Athens* [‡]*ETH Zürich* [*]*University of Toronto* [§]*University of Malaga*

How to Support Virtual Memory?

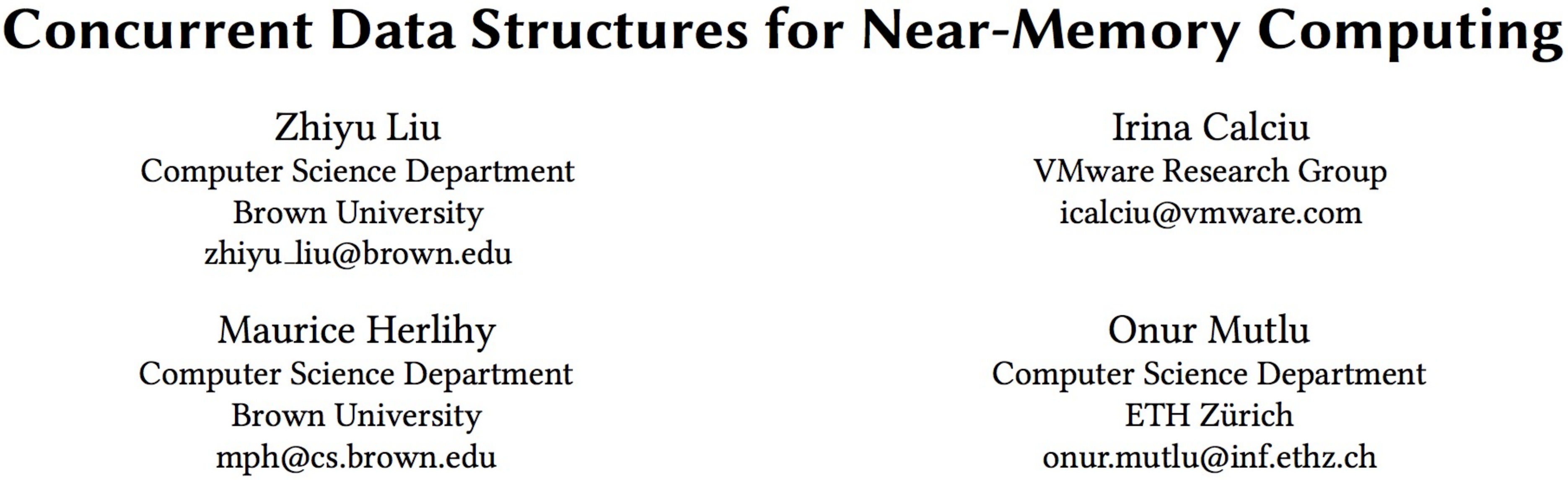
- Kevin Hsieh, Samira Khan, Nandita Vijaykumar, Kevin K. Chang, Amirali Boroumand, Saugata Ghose, and Onur Mutlu, **"Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation"** *Proceedings of the 34th IEEE International Conference on Computer Design (**ICCD**)*, Phoenix,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

- Zhiyu Liu, Irina Calciu, Maurice Herlihy, and Onur Mutlu, **"Concurrent Data Structures for Near-Memory Computing"** *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures (**SPAA**)*, Washington, DC, USA, July 2017.
  [Slides (pptx) (pdf)]



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
  - Kim+, **"Ramulator: A Fast and Extensible DRAM Simulator"**, IEEE CAL 2015.
  - https://github.com/CMU-SAFARI/ramulator-pim
  - https://github.com/CMU-SAFARI/ramulator
  - [Source Code for Ramulator-PIM]

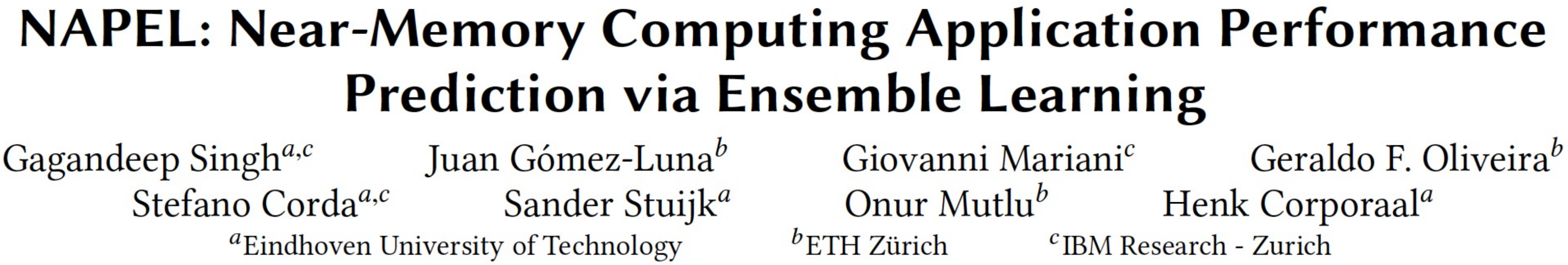
Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim[1] Weikun Yang[1,2] Onur Mutlu[1]

[1]Carnegie Mellon University [2]Peking University

Performance & Energy Models for PIM

- Gagandeep Singh, Juan Gomez-Luna, Giovanni Mariani, Geraldo F. Oliveira, Stefano Corda, Sander Stujik, 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[a,c] Juan Gómez-Luna[b] Giovanni Mariani[c] Geraldo F. Oliveira[b]
Stefano Corda[a,c] Sander Stuijk[a] Onur Mutlu[b] Henk Corporaal[a]
[a]Eindhoven University of Technology [b]ETH Zürich [c]IBM Research - Zurich

Challenge and Opportunity for Future

Fundamentally

Energy-Efficient

(Data-Centric)

Computing Architectures

Challenge and Opportunity for Future

Fundamentally

High-Performance

(Data-Centric)

Computing Architectures

Challenge and Opportunity for Future

Computing Architectures

with

Minimal Data Movement

What We Have Less Time For

Challenge and Opportunity for Future

Data-Driven

(Self-Optimizing)

Computing Architectures

Challenge and Opportunity for Future

Data-Aware

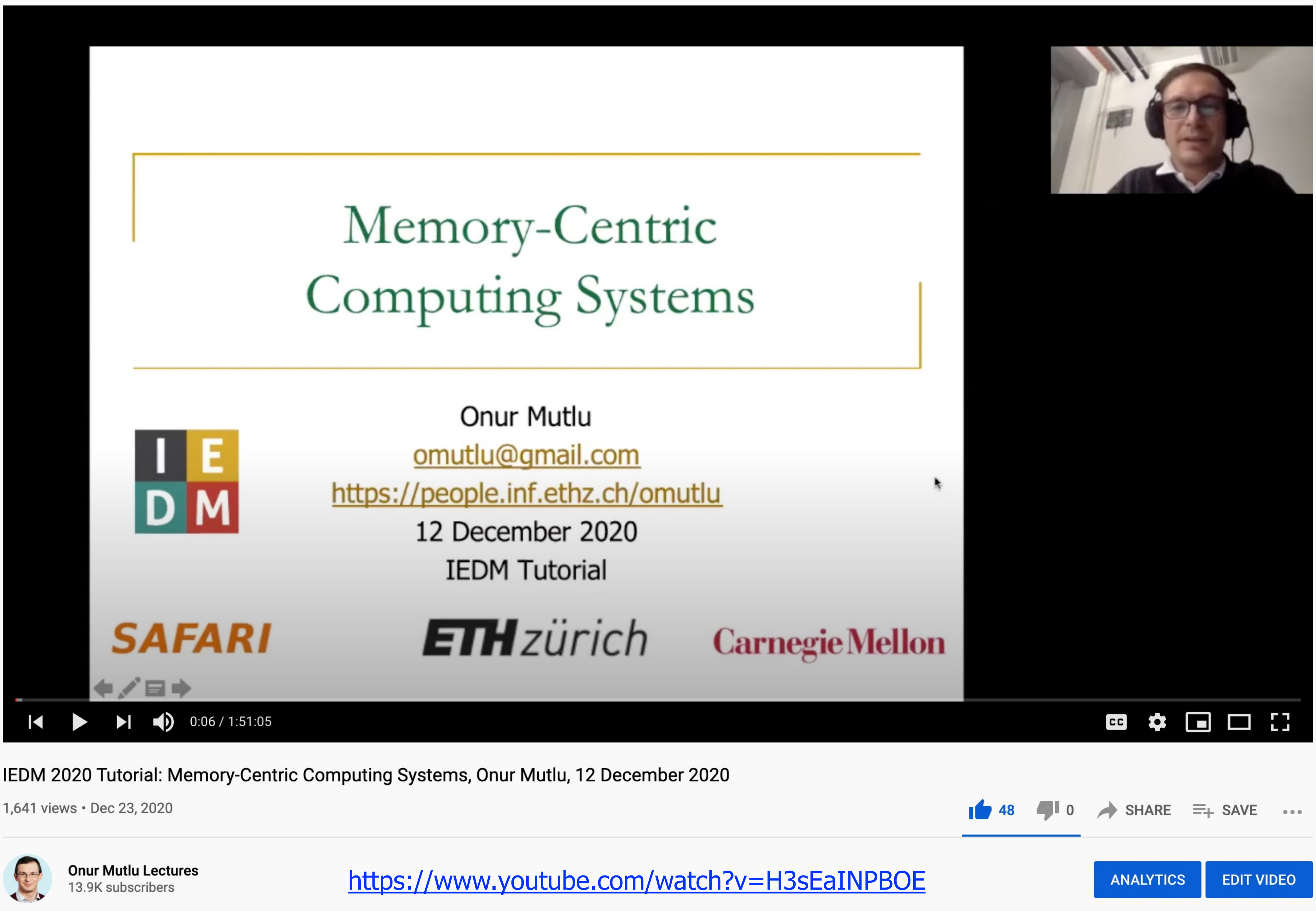
(Expressive)

Computing Architectures

More Info in This Longer Tutorial…

- Onur Mutlu,
  "Memory-Centric Computing Systems"
  Invited Tutorial at *66th International Electron Devices Meeting (**IEDM**)*, Virtual, 12 December 2020.
  [Slides (pptx) (pdf)]
  [Executive Summary Slides (pptx) (pdf)]
  [Tutorial Video (1 hour 51 minutes)]
  [Executive Summary Video (2 minutes)]
  [Abstract and Bio]
  [Related Keynote Paper from VLSI-DAT 2020]
  [Related Review Paper on Processing in Memory]

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



Memory-Centric Computing Systems

Onur Mutlu

omutlu@gmail.com

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

12 December 2020

IEDM Tutorial

IEDM 2020 Tutorial: Memory-Centric Computing Systems, Onur Mutlu, 12 December 2020

1,641 views • Dec 23, 2020

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

https://www.youtube.com/onurmutlulectures

Data-Driven Architectures

Corollaries: Architectures Today …

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

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

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

Exploiting Data to Design Intelligent Architectures

System Architecture Design Today

- Human-driven
  - Humans design the policies (how to do things)
- Many (too) simple, short-sighted policies all over the system
- No automatic data-driven policy learning
- (Almost) no learning: cannot take lessons from past actions

Can we design fundamentally intelligent architectures?

An Intelligent Architecture

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

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

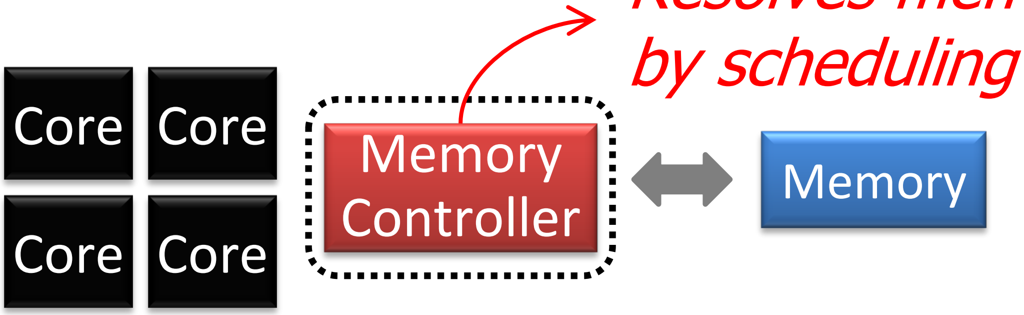
- Automatic data-driven policy learning

- All controllers are intelligent data-driven agents

How do we start?

Self-Optimizing Memory Controllers

Memory Controller



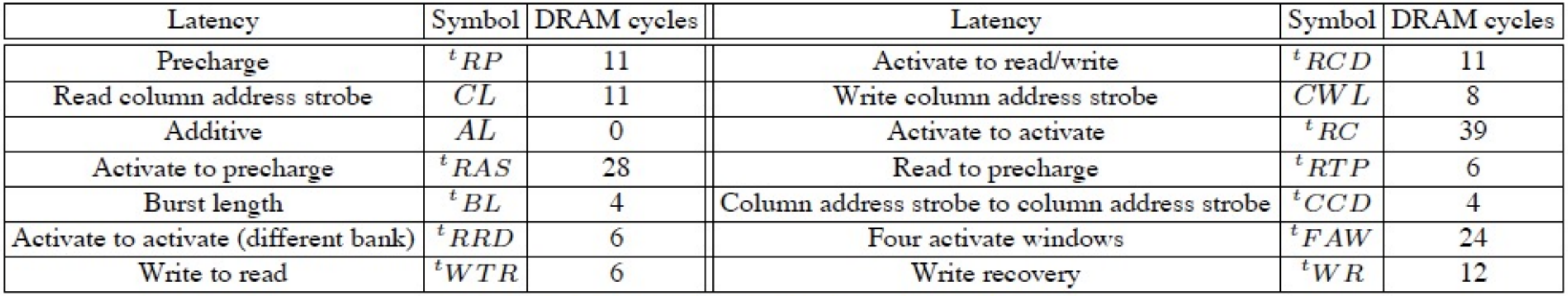
Resolves memory contention by scheduling requests

How to schedule requests to maximize system performance?

Why are Memory Controllers Difficult to Design?

- Need to obey DRAM timing constraints for correctness
  - There are many (50+) timing constraints in DRAM
  - tWTR: Minimum number of cycles to wait before issuing a read command after a write command is issued
  - tRC: Minimum number of cycles between the issuing of two consecutive activate commands to the same bank
  - …
- Need to keep track of many resources to prevent conflicts
  - Channels, banks, ranks, data bus, address bus, row buffers, …
- Need to handle DRAM refresh
- Need to manage power consumption
- Need to optimize performance & QoS (in the presence of constraints)
  - Reordering is not simple
  - Fairness and QoS needs complicates the scheduling problem
- …

Many Memory Timing Constraints



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

Table 4. DDR3 1600 DRAM timing specifications

- From Lee et al., "DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems," HPS Technical Report, April 2010.

Many Memory Timing Constraints

- Kim et al., "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," ISCA 2012.
- Lee et al., "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.

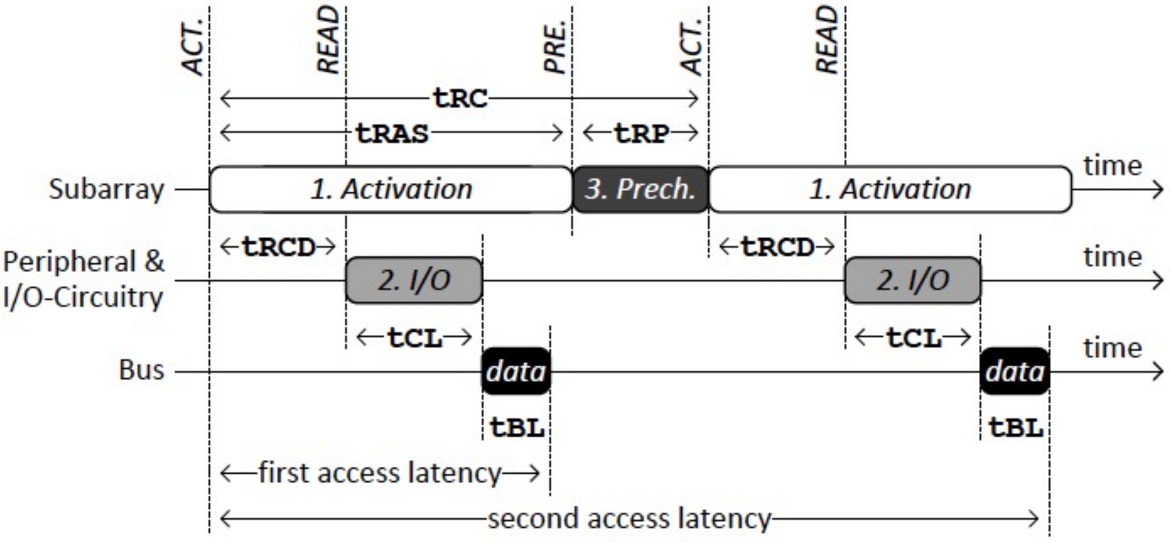


Figure 5. Three Phases of DRAM Access

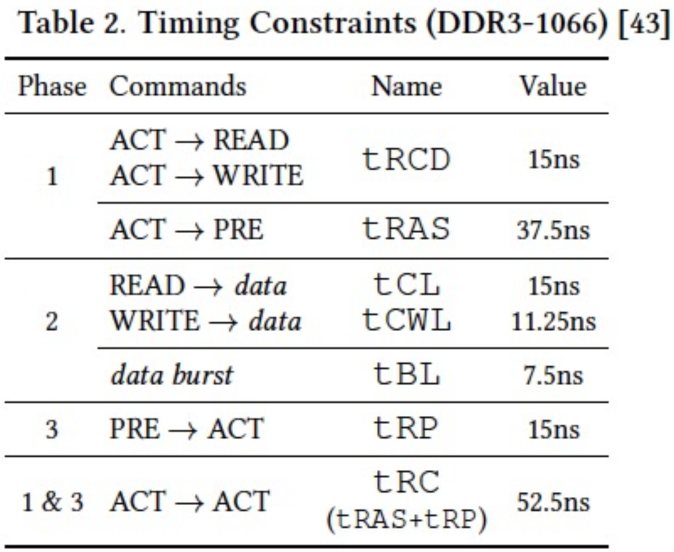
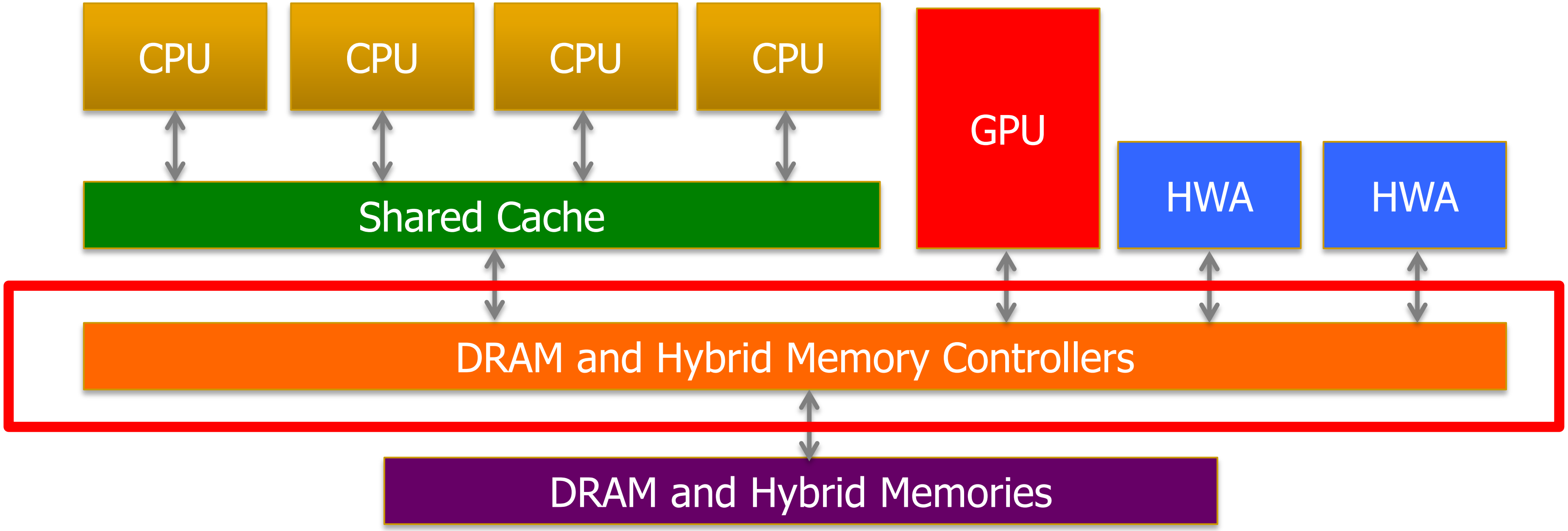


Table 2. Timing Constraints (DDR3-1066) [43]

Phase	Commands	Name	Value
1	ACT → READ ACT → WRITE	tRCD	15ns
	ACT → PRE	tRAS	37.5ns
2	READ → *data* WRITE → *data*	tCL tCWL	15ns 11.25ns
	data burst	tBL	7.5ns
3	PRE → ACT	tRP	15ns
1 & 3	ACT → ACT	tRC (tRAS+tRP)	52.5ns

Memory Controller Design Is Becoming More Difficult



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

Reality and Dream

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

- Dream: Wouldn't it be nice if the DRAM controller automatically found a good scheduling policy on its own?

Self-Optimizing DRAM Controllers

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

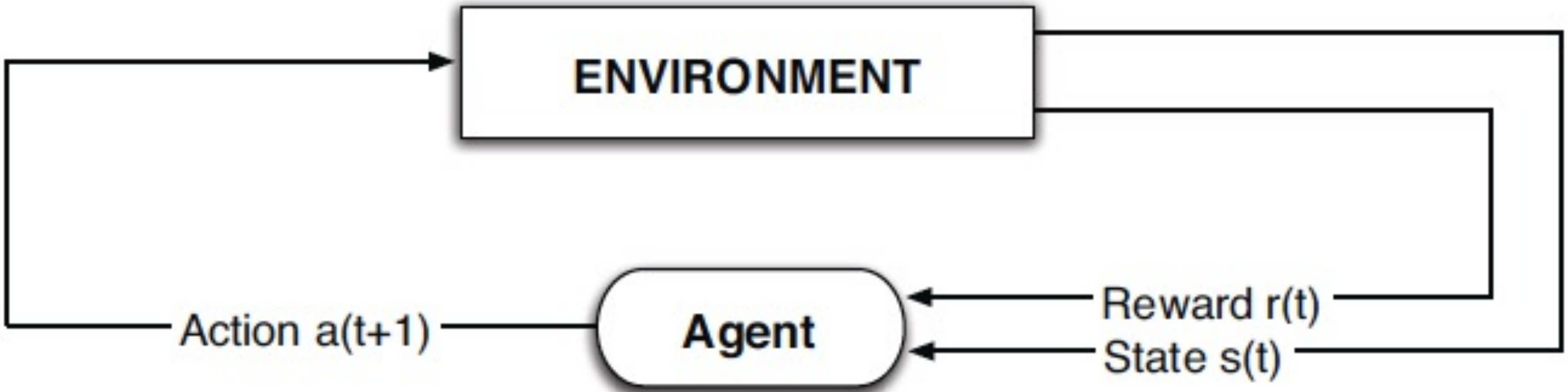
- Idea: A memory controller that adapts its scheduling policy to workload behavior and system conditions using machine learning.

- Observation: Reinforcement learning maps nicely to memory control.

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

Ipek+, "Self Optimizing Memory Controllers: A Reinforcement Learning Approach," ISCA 2008.

Self-Optimizing DRAM Controllers

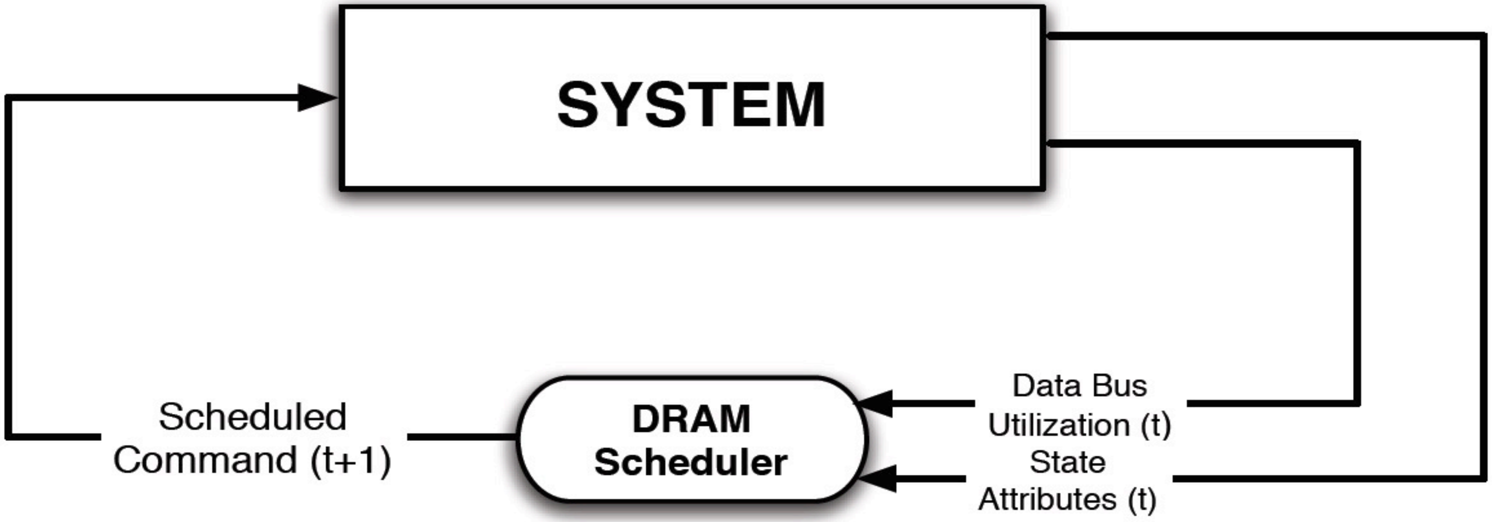


Goal: Learn to choose actions to maximize $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots \ (0 \leq \gamma < 1)$

Figure 2: (a) Intelligent agent based on reinforcement learning principles;

Self-Optimizing DRAM Controllers

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



Self-Optimizing DRAM Controllers

- Engin Ipek, Onur Mutlu, José F. Martínez, and Rich Caruana, **"Self Optimizing Memory Controllers: A Reinforcement Learning Approach"** *Proceedings of the 35th International Symposium on Computer Architecture (**ISCA**)*, pages 39-50, Beijing, China, June 2008.

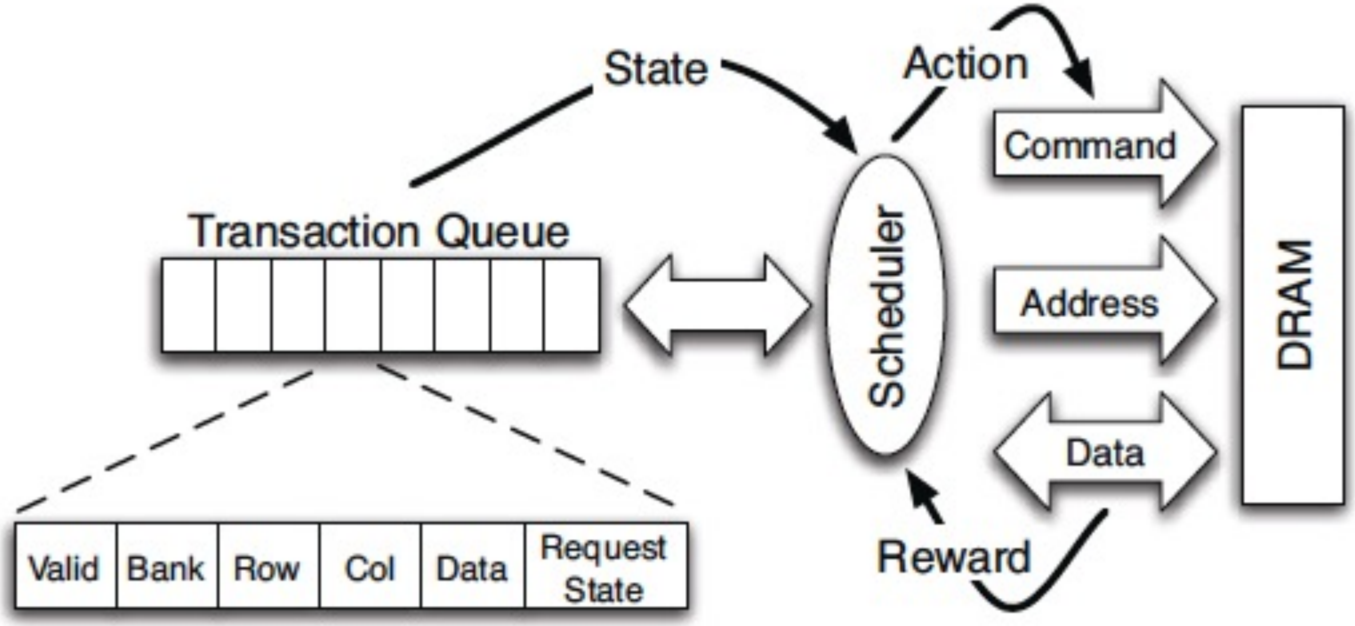


Figure 4: High-level overview of an RL-based scheduler.

States, Actions, Rewards

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

  Goal is to maximize long-term data bus utilization

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

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

Performance Results

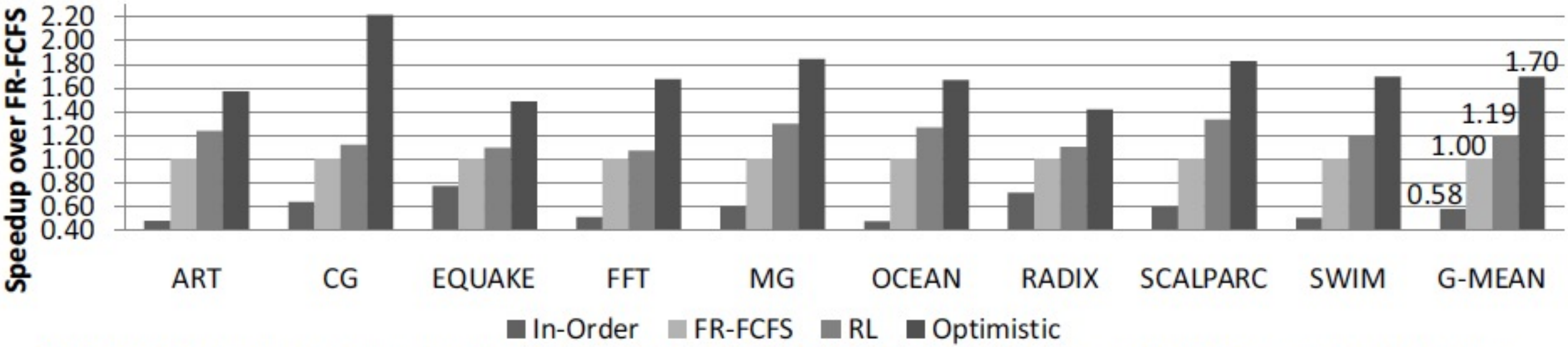


Figure 7: Performance comparison of in-order, FR-FCFS, RL-based, and optimistic memory controllers

Large, robust performance improvements over many human-designed policies

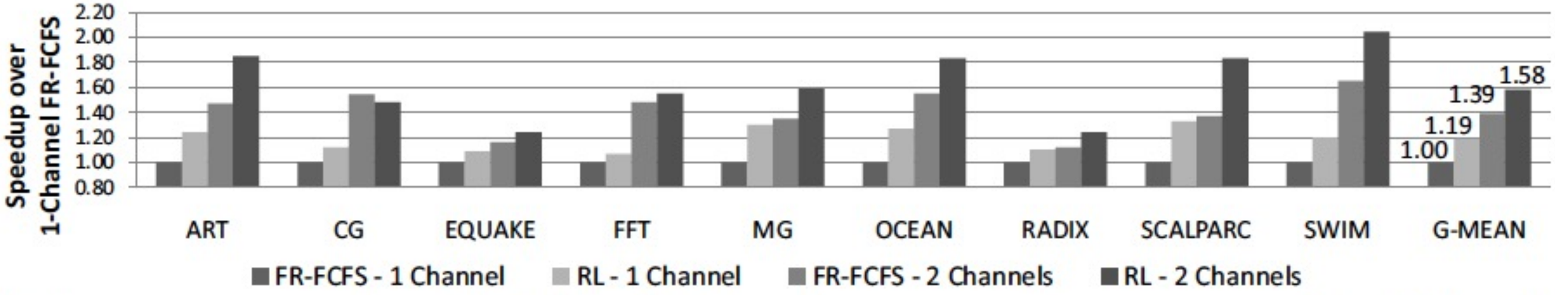


Figure 15: Performance comparison of FR-FCFS and RL-based memory controllers on systems with 6.4GB/s and 12.8GB/s peak DRAM bandwidth

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"** *Proceedings of the 35th International Symposium on Computer Architecture (**ISCA**)*, pages 39-50, Beijing, China, June 2008.

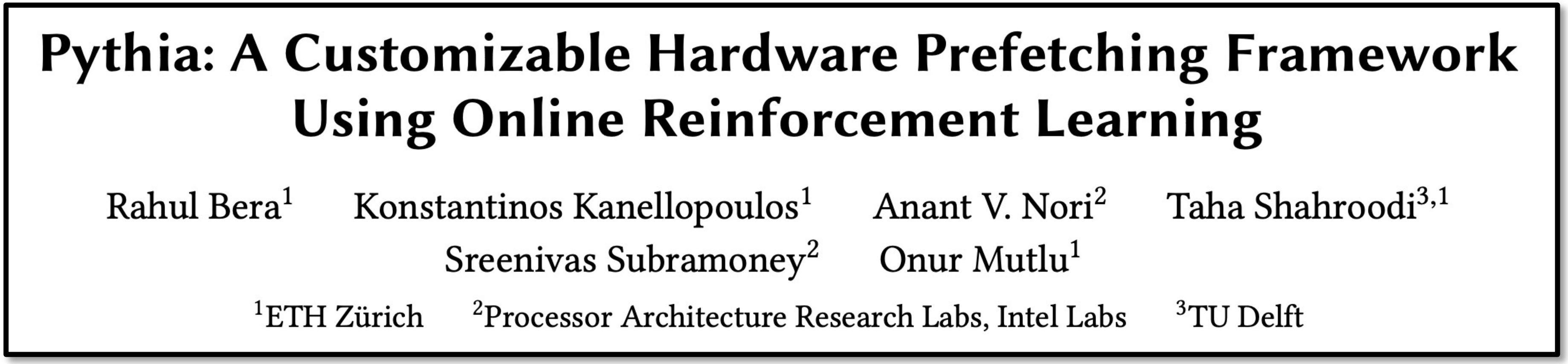
Self-Optimizing Memory Controllers: A Reinforcement Learning Approach

Engin İpek[1,2] **Onur Mutlu**[2] **José F. Martínez**[1] **Rich Caruana**[1]

[1] Cornell University, Ithaca, NY 14850 USA
[2] Microsoft Research, Redmond, WA 98052 USA

Self-Optimizing Memory Prefetchers

- To appear at MICRO 2021



Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera[1] Konstantinos Kanellopoulos[1] Anant V. Nori[2] Taha Shahroodi[3,1]
Sreenivas Subramoney[2] Onur Mutlu[1]
[1]ETH Zürich [2]Processor Architecture Research Labs, Intel Labs [3]TU Delft

https://arxiv.org/pdf/2109.12021.pdf

Pythia

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

SAFARI
SAFARI Research Group
safari.ethz.ch

ETH zürich

intel

TU Delft

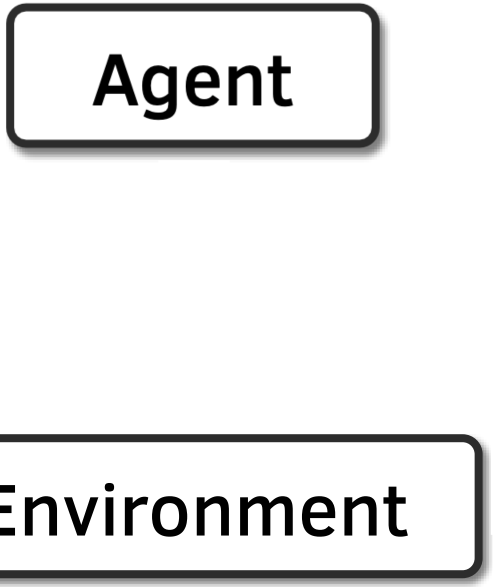
Our Goal

A prefetching **framework** that:

1. Can learn to prefetch using **multiple features** and **inherent system-level feedback** information
2. Can be **easily customized in silicon** to change feature type and/or prefetcher's objective

Basics of Reinforcement Learning (RL)

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



Agent

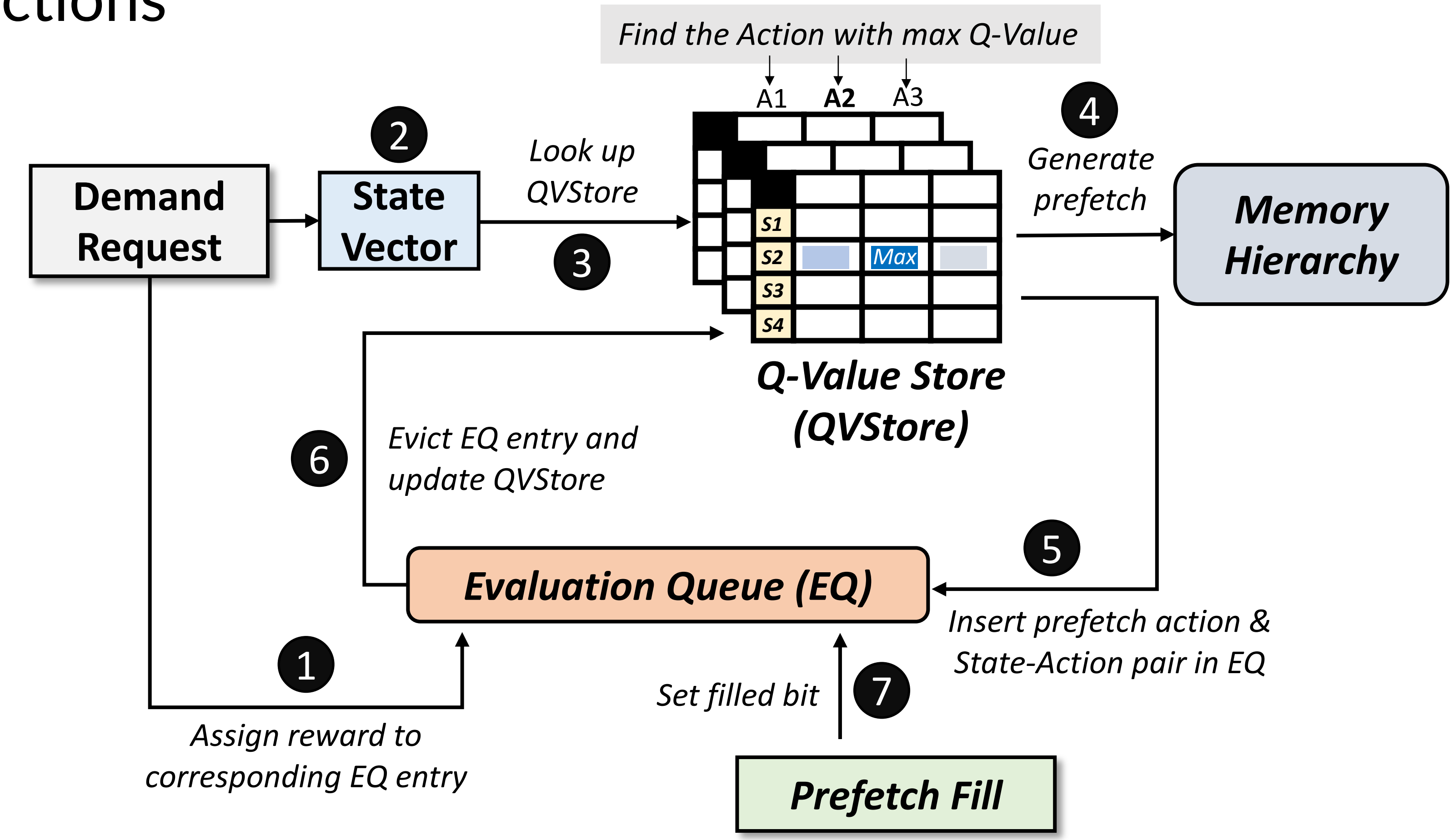
Environment

- Agent stores **Q-values** for *every* state-action pairs
  - Given a state, selects action that provides **highest** Q-value

Formulating Prefetching as RL

Pythia Overview

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



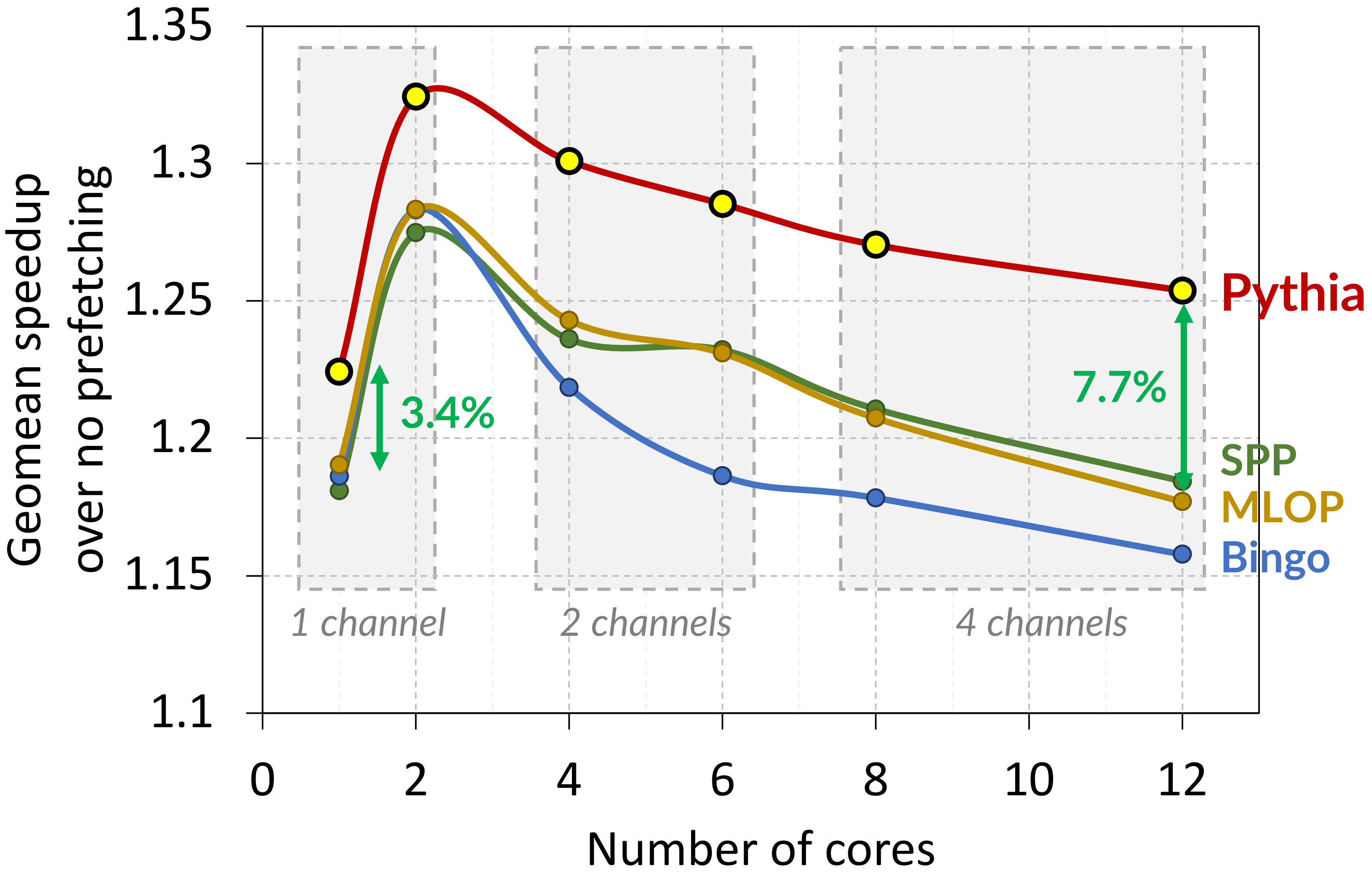
Simulation Methodology

- **Champsim** trace-driven simulator

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

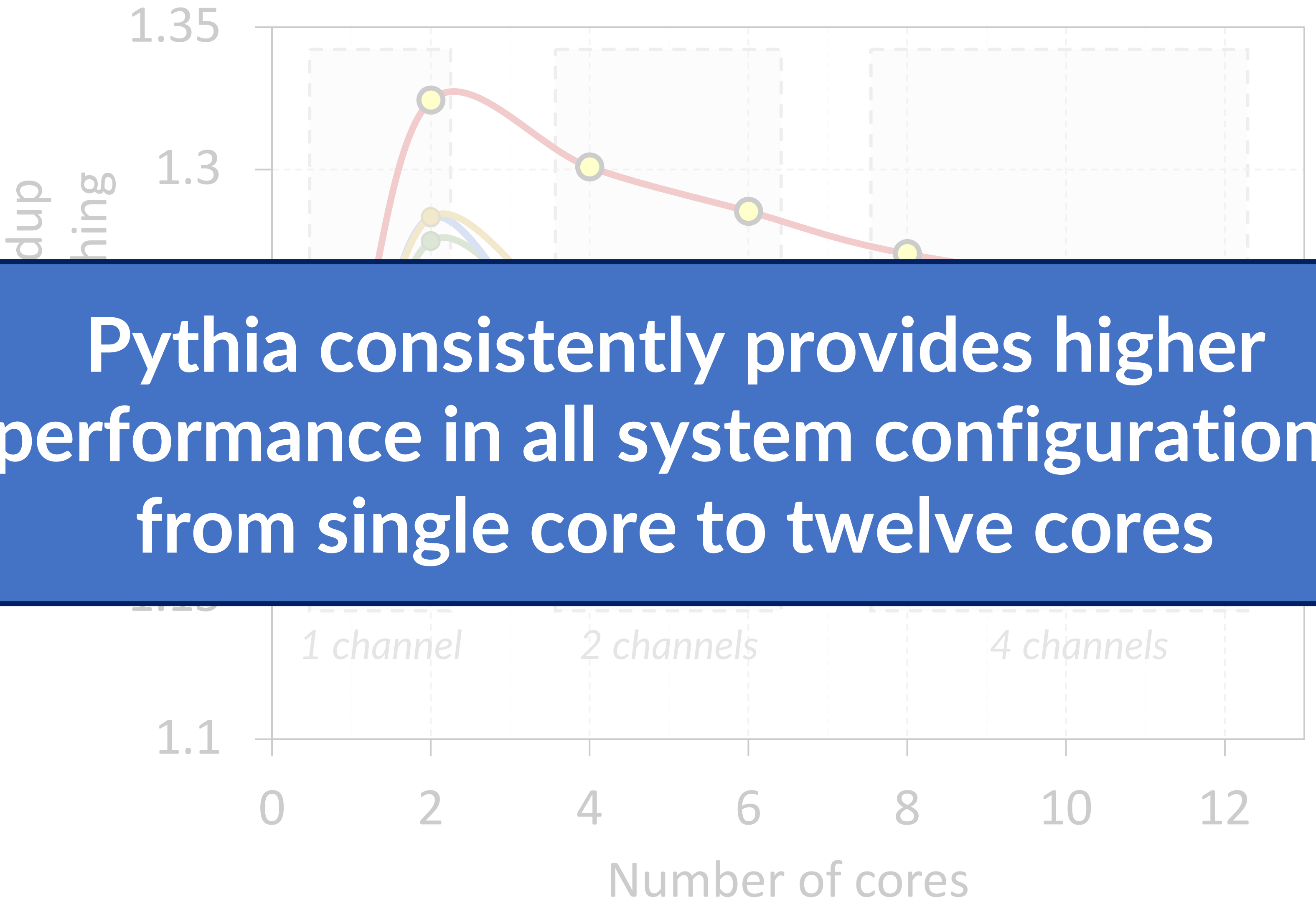
- **Five** state-of-the-art prefetchers
  - SPP [Kim+, MICRO'16]
  - Bingo [Bakhshalipour+, HPCA'19]
  - MLOP [Shakerinava+, Prefetching Championship-3]
  - SPP+DSPatch [Bera+, MICRO'19]
  - SPP+PPF [Bhatia+, ISCA'20]

Performance with Varying Core Count



SAFARI

Performance with Varying Core Count



Pythia consistently provides higher performance in all system configurations from single core to twelve cores

Pythia is Completely Open Source

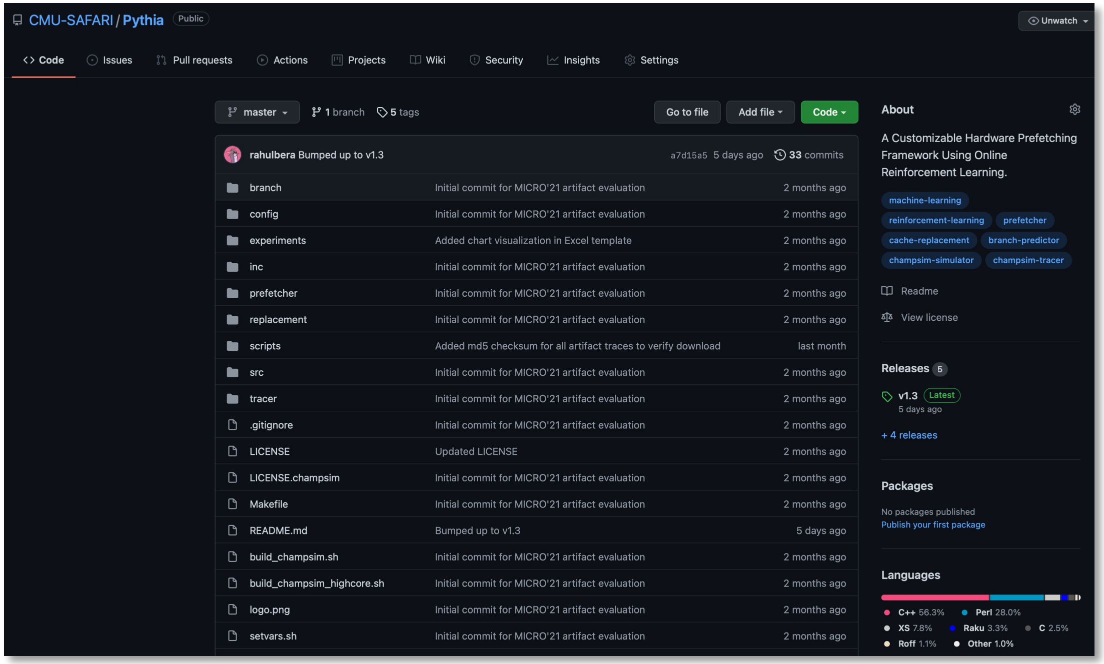
https://github.com/CMU-SAFARI/Pythia

- MICRO’21 **artifact evaluated**
- **Champsim source** code + **Chisel** modeling code
- **All traces** used for evaluation

Artifacts Evaluated acm Functional V1.1

Artifacts Available acm V1.1

Results Reproduced acm V1.1



rahulbera Bumped up to v1.3 a7d15a5 5 days ago 33 commits

branch	Initial commit for MICRO'21 artifact evaluation	2 months ago
config	Initial commit for MICRO'21 artifact evaluation	2 months ago
experiments	Added chart visualization in Excel template	2 months ago
inc	Initial commit for MICRO'21 artifact evaluation	2 months ago
prefetcher	Initial commit for MICRO'21 artifact evaluation	2 months ago
replacement	Initial commit for MICRO'21 artifact evaluation	2 months ago
scripts	Added md5 checksum for all artifact traces to verify download	last month
src	Initial commit for MICRO'21 artifact evaluation	2 months ago
tracer	Initial commit for MICRO'21 artifact evaluation	2 months ago
.gitignore	Initial commit for MICRO'21 artifact evaluation	2 months ago
LICENSE	Updated LICENSE	2 months ago
LICENSE.champsim	Initial commit for MICRO'21 artifact evaluation	2 months ago
Makefile	Initial commit for MICRO'21 artifact evaluation	2 months ago
README.md	Bumped up to v1.3	5 days ago
build_champsim.sh	Initial commit for MICRO'21 artifact evaluation	2 months ago
build_champsim_highcore.sh	Initial commit for MICRO'21 artifact evaluation	2 months ago
logo.png	Initial commit for MICRO'21 artifact evaluation	2 months ago
setvars.sh	Initial commit for MICRO'21 artifact evaluation	2 months ago

About

A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning.

machine-learning reinforcement-learning prefetcher cache-replacement branch-predictor champsim-simulator champsim-tracer

Readme

View license

Releases 5

v1.3 Latest
5 days ago

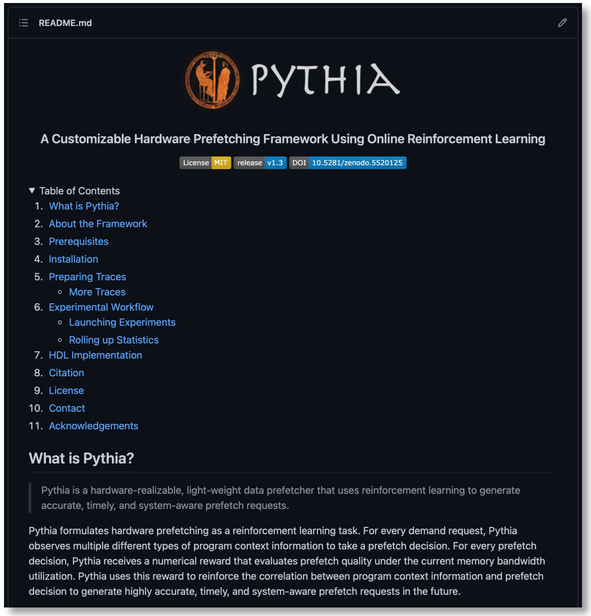
+ 4 releases

Packages

No packages published
Publish your first package

Languages

C++	56.3%
Perl	28.0%
XS	7.8%
Raku	3.3%
C	2.5%
Roff	1.1%
Other	1.0%



A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

License MIT release v1.3 DOI 10.5281/zenodo.5520125

▼ Table of Contents

1. What is Pythia?
2. About the Framework
3. Prerequisites
4. Installation
5. Preparing Traces
   - More Traces
6. Experimental Workflow
   - Launching Experiments
   - Rolling up Statistics
7. HDL Implementation
8. Citation
9. License
10. Contact
11. Acknowledgements

What is Pythia?

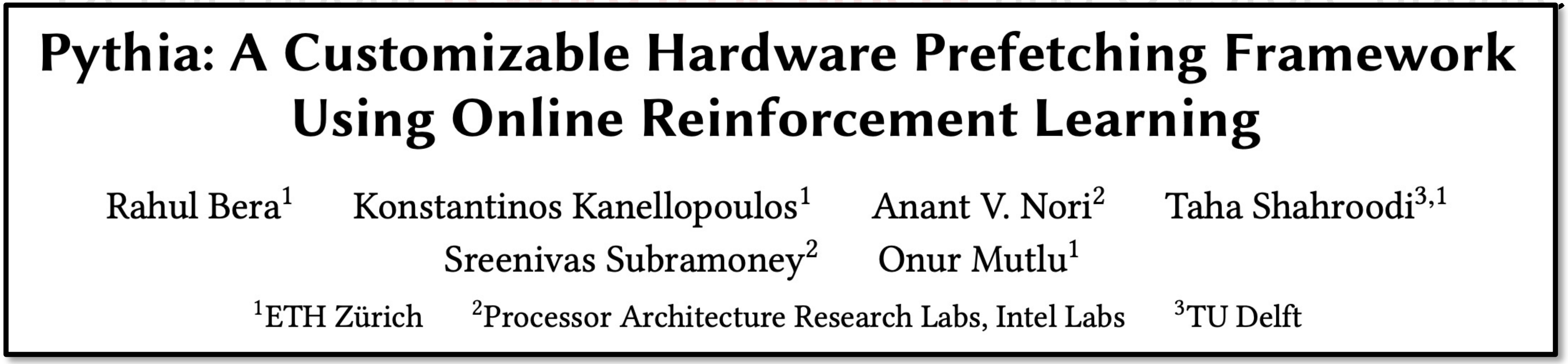
> Pythia is a hardware-realizable, light-weight data prefetcher that uses reinforcement learning to generate accurate, timely, and system-aware prefetch requests.

Pythia formulates hardware prefetching as a reinforcement learning task. For every demand request, Pythia observes multiple different types of program context information to take a prefetch decision. For every prefetch decision, Pythia receives a numerical reward that evaluates prefetch quality under the current memory bandwidth utilization. Pythia uses this reward to reinforce the correlation between program context information and prefetch decision to generate highly accurate, timely, and system-aware prefetch requests in the future.

More in the Pythia Paper

- Automatic **design-space exploration** for Pythia

- Details about **reward assignment** and QVStore update



Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

Rahul Bera[1] Konstantinos Kanellopoulos[1] Anant V. Nori[2] Taha Shahroodi[3,1]
Sreenivas Subramoney[2] Onur Mutlu[1]

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

✓Performance comparison with **unseen traces**
✓Understanding Pythia's learning with **a case study**
✓Performance benefits via **customization**

https://arxiv.org/pdf/2109.12021.pdf

An Intelligent Architecture

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

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

- Automatic data-driven policy learning

- All controllers are intelligent data-driven agents

We need to rethink design (of all controllers)

Challenge and Opportunity for Future

Data-Driven

(Self-Optimizing)

Computing Architectures

Data-Aware Architectures

Corollaries: Architectures Today …

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

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

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

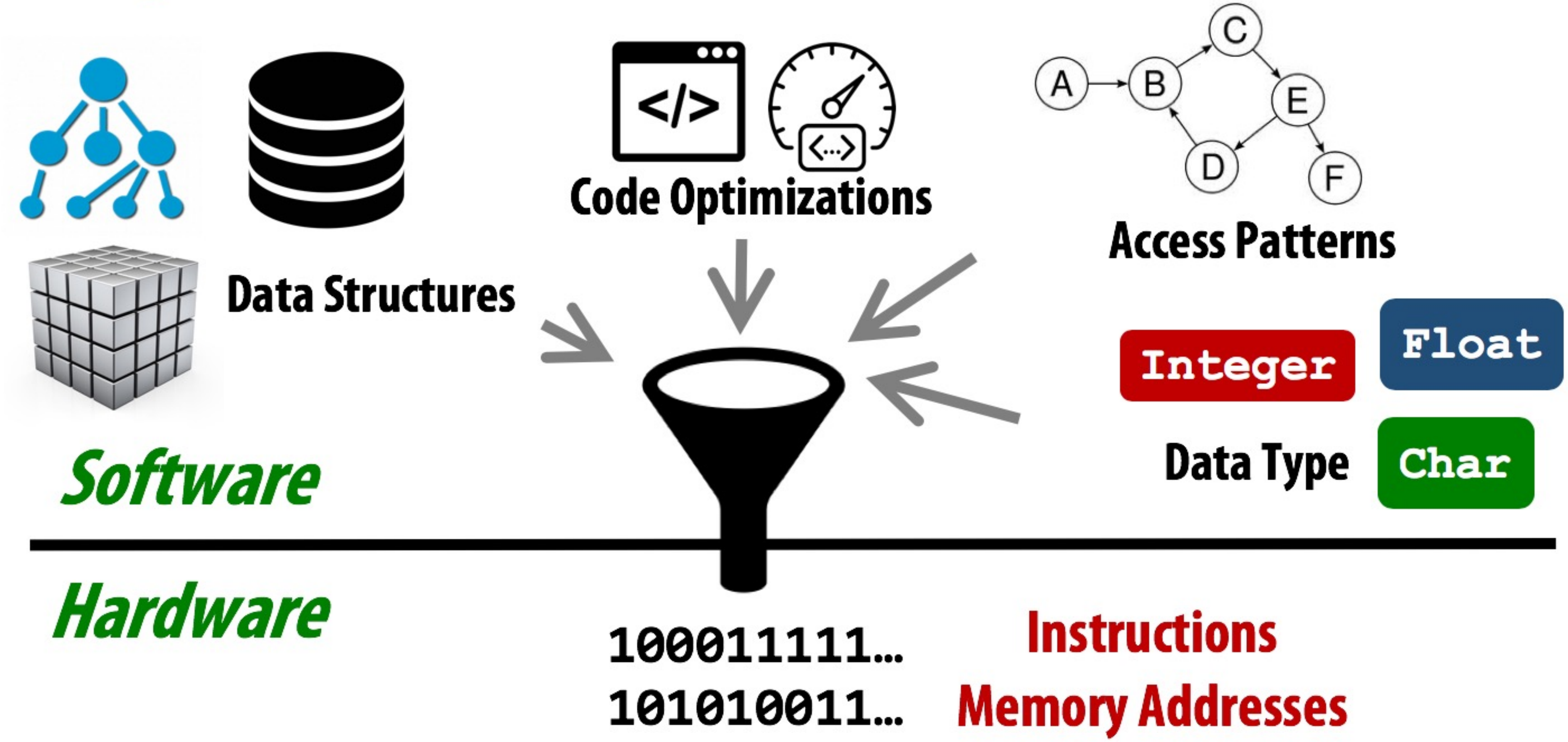
Data-Aware Architectures

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

- It makes use of different properties of data to improve performance, efficiency and other metrics
  - Compressibility
  - Approximability
  - Locality
  - Sparsity
  - Criticality for Computation X
  - Access Semantics
  - …

One Problem: Limited Expressiveness

Higher-level information is not visible to HW



Data Structures

Code Optimizations

Access Patterns

Integer

Float

Data Type

Char

Software

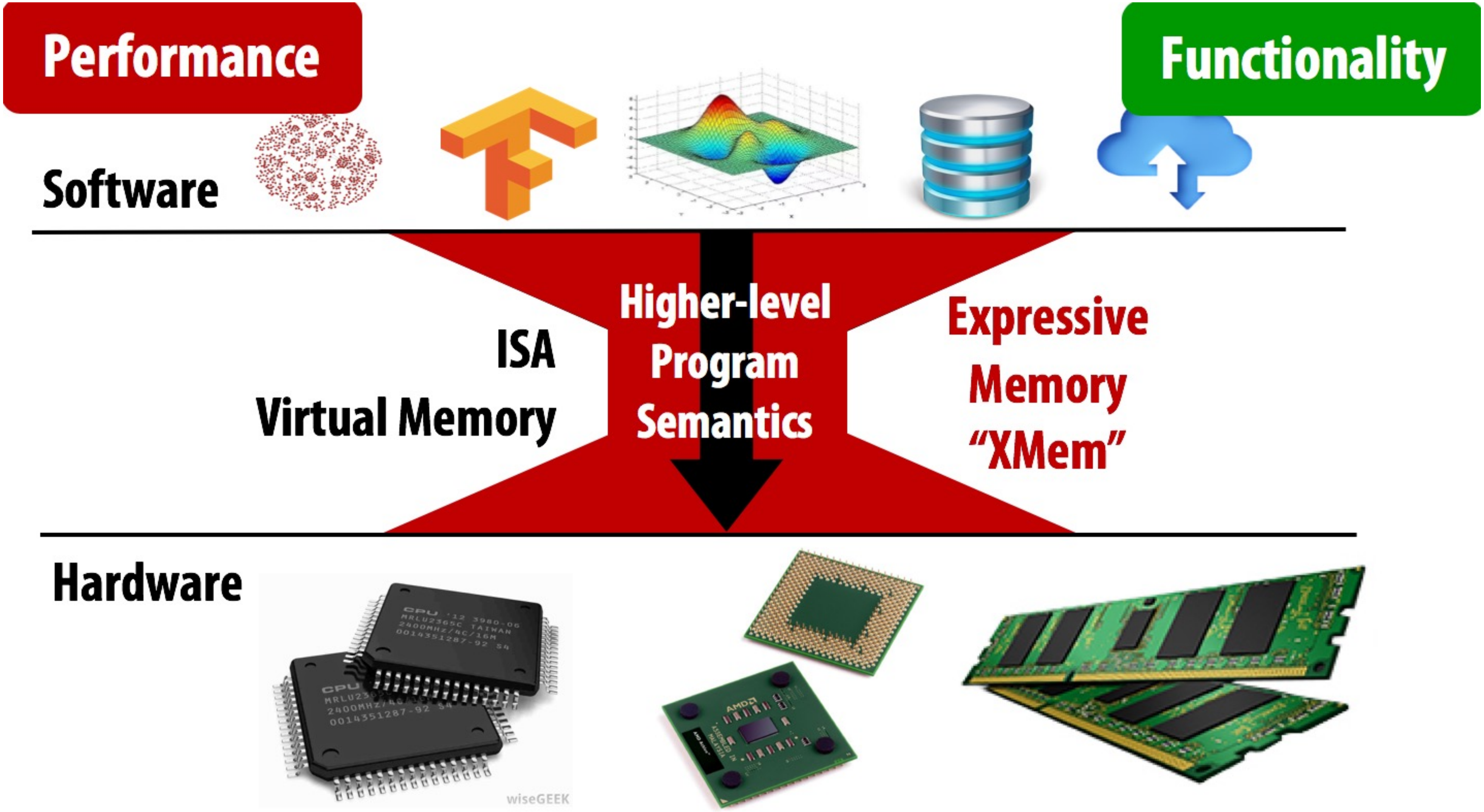
Hardware

100011111...
101010011...

Instructions

Memory Addresses

A Solution: More Expressive Interfaces



Performance

Functionality

Software

ISA

Virtual Memory

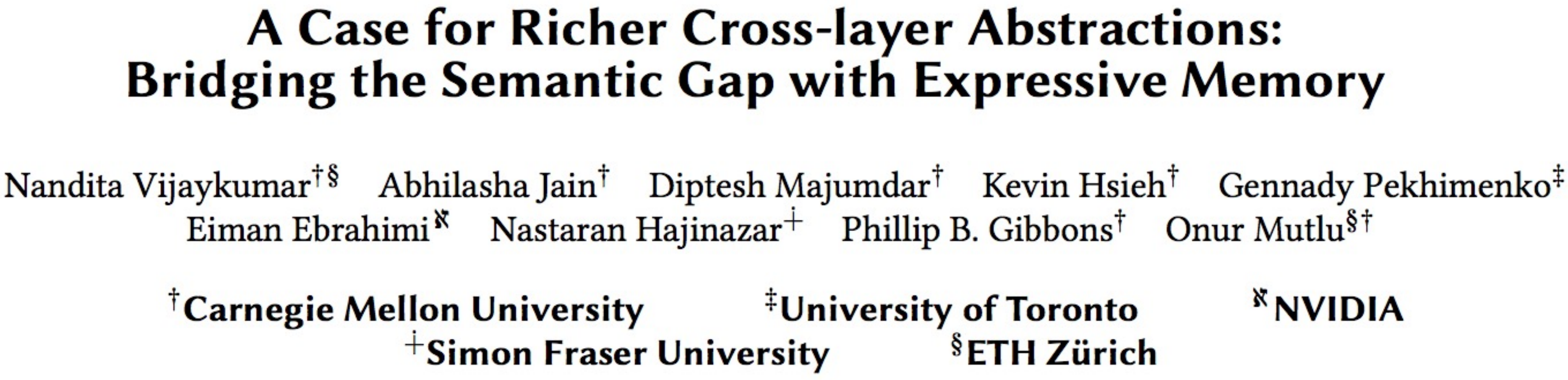
Higher-level Program Semantics

Expressive Memory "XMem"

Hardware

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"**
Proceedings of the 45th International Symposium on Computer Architecture (***ISCA***), Los Angeles, CA, USA, June 2018.
[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[ℵ] Nastaran Hajinazar[+] Phillip B. Gibbons[†] Onur Mutlu[§†]

[†]Carnegie Mellon University [‡]University of Toronto [ℵ]NVIDIA
[+]Simon Fraser University [§]ETH Zürich

X-MeM Aids Many Optimizations

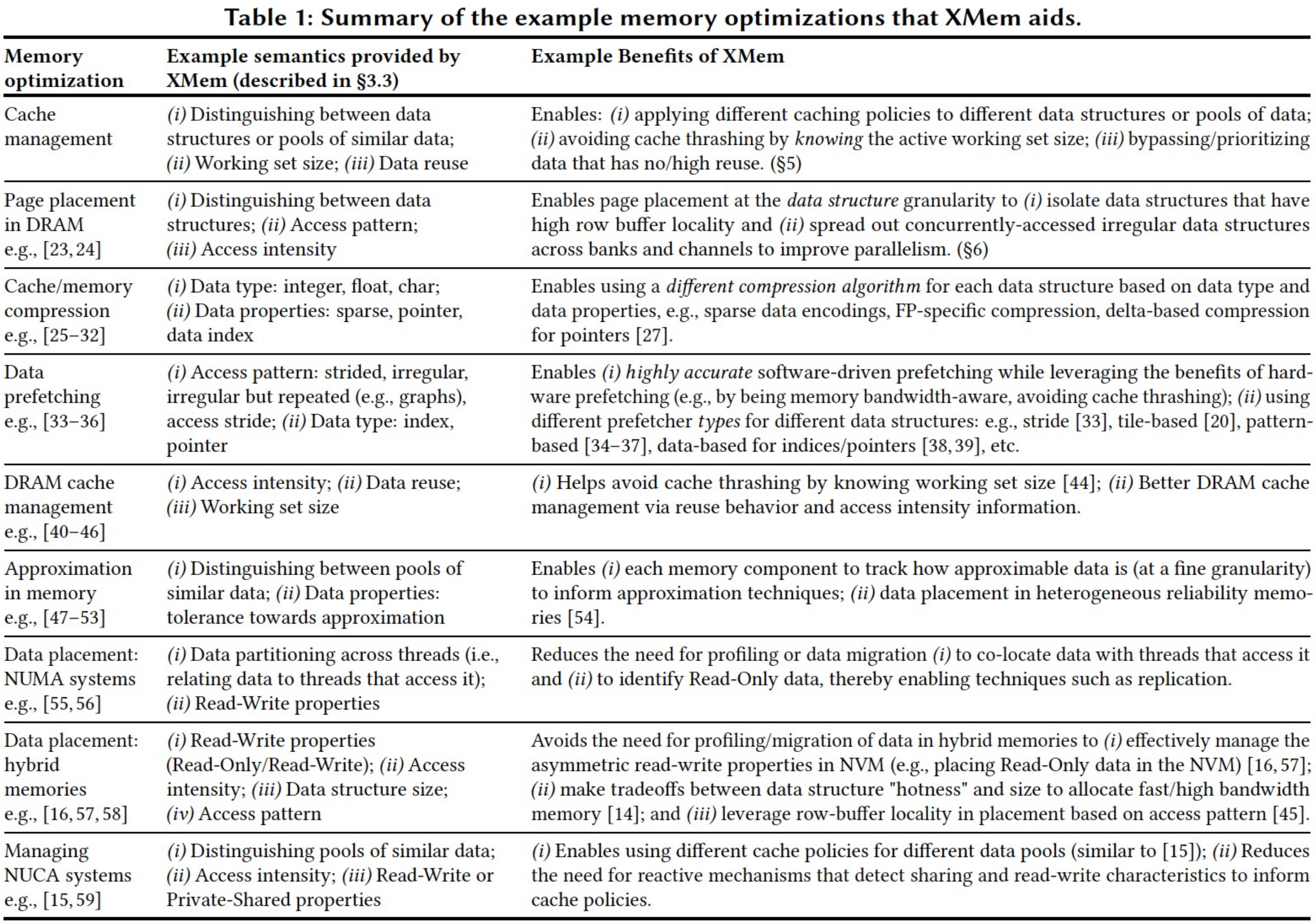
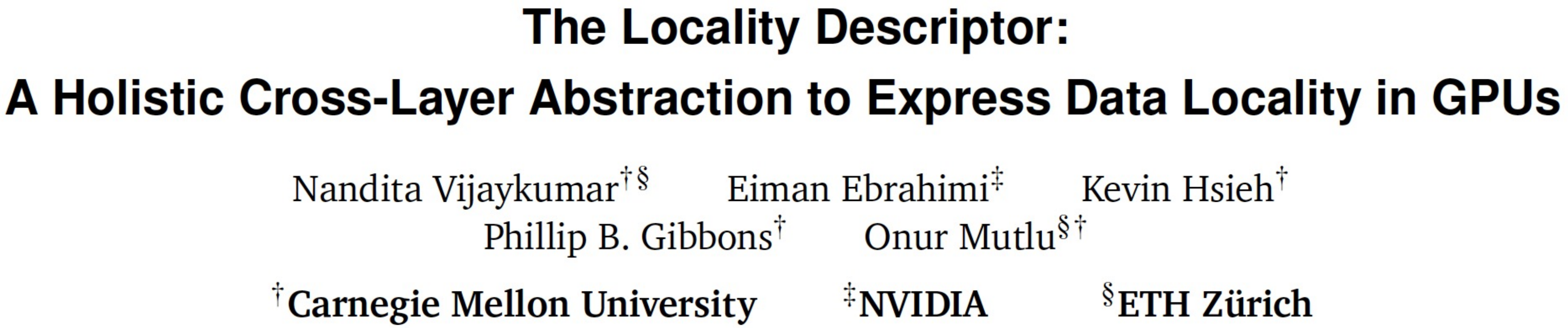


Table 1: Summary of the example memory optimizations that XMem aids.

Memory optimization	**Example semantics provided by XMem (described in §3.3)**	**Example Benefits of XMem**
Cache management	*(i)* Distinguishing between data structures or pools of similar data; *(ii)* Working set size; *(iii)* Data reuse	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)
Page placement in DRAM e.g., [23, 24]	*(i)* Distinguishing between data structures; *(ii)* Access pattern; *(iii)* Access intensity	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)
Cache/memory compression e.g., [25–32]	*(i)* Data type: integer, float, char; *(ii)* Data properties: sparse, pointer, data index	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].
Data prefetching e.g., [33–36]	*(i)* Access pattern: strided, irregular, irregular but repeated (e.g., graphs), access stride; *(ii)* Data type: index, pointer	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.
DRAM cache management e.g., [40–46]	*(i)* Access intensity; *(ii)* Data reuse; *(iii)* Working set size	*(i)* Helps avoid cache thrashing by knowing working set size [44]; *(ii)* Better DRAM cache management via reuse behavior and access intensity information.
Approximation in memory e.g., [47–53]	*(i)* Distinguishing between pools of similar data; *(ii)* Data properties: tolerance towards approximation	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].
Data placement: NUMA systems e.g., [55, 56]	*(i)* Data partitioning across threads (i.e., relating data to threads that access it); *(ii)* Read-Write properties	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.
Data placement: hybrid memories e.g., [16, 57, 58]	*(i)* Read-Write properties (Read-Only/Read-Write); *(ii)* Access intensity; *(iii)* Data structure size; *(iv)* Access pattern	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 "hotness" and size to allocate fast/high bandwidth memory [14]; and *(iii)* leverage row-buffer locality in placement based on access pattern [45].
Managing NUCA systems e.g., [15, 59]	*(i)* Distinguishing pools of similar data; *(ii)* Access intensity; *(iii)* Read-Write or Private-Shared properties	*(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.

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"
  Proceedings of the 45th International Symposium on Computer Architecture (***ISCA***), Los Angeles, CA, USA, June 2018.
  [Slides (pptx) (pdf)] [Lightning Talk Slides (pptx) (pdf)]
  [Lightning Talk Video]

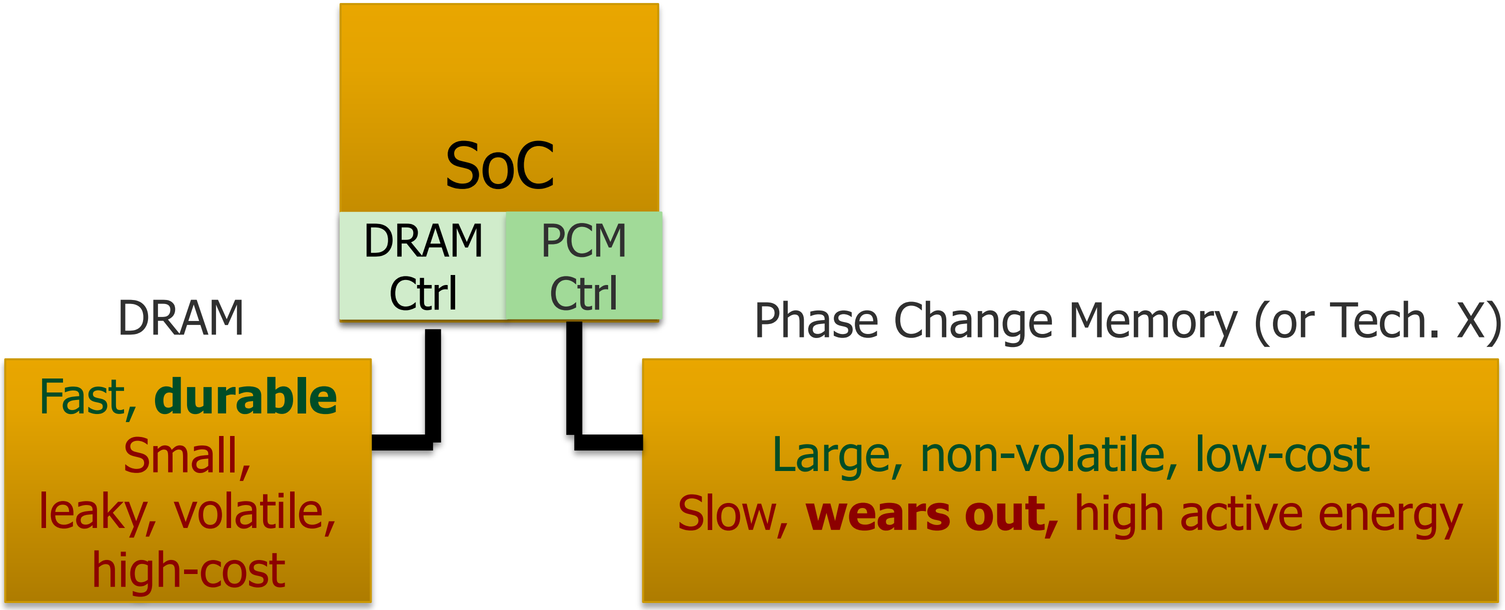


The Locality Descriptor:
A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs

Nandita Vijaykumar[†§] Eiman Ebrahimi[‡] Kevin Hsieh[†]
Phillip B. Gibbons[†] Onur Mutlu[§†]

[†]**Carnegie Mellon University** [‡]**NVIDIA** [§]**ETH Zürich**

An Example: Hybrid Memory Management


SoC
DRAM Ctrl
PCM Ctrl
DRAM
Fast, **durable**
Small,
leaky, volatile,
high-cost
Phase Change Memory (or Tech. X)
Large, non-volatile, low-cost
Slow, **wears out,** high active energy

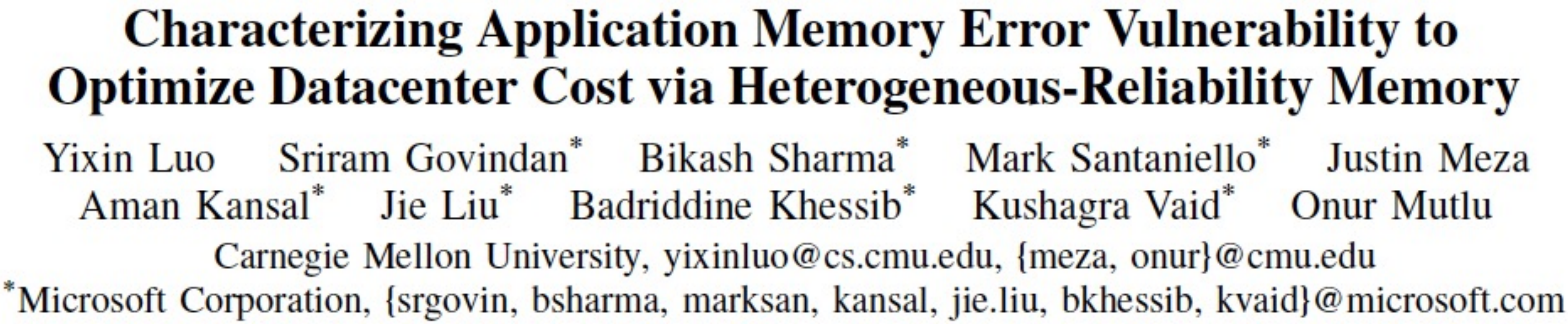
Hardware/software manage data allocation and movement
to achieve the best of multiple technologies

Meza+, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters, 2012.

Yoon+, "Row Buffer Locality Aware Caching Policies for Hybrid Memories," ICCD 2012 Best Paper Award.

An Example: Heterogeneous-Reliability Memory

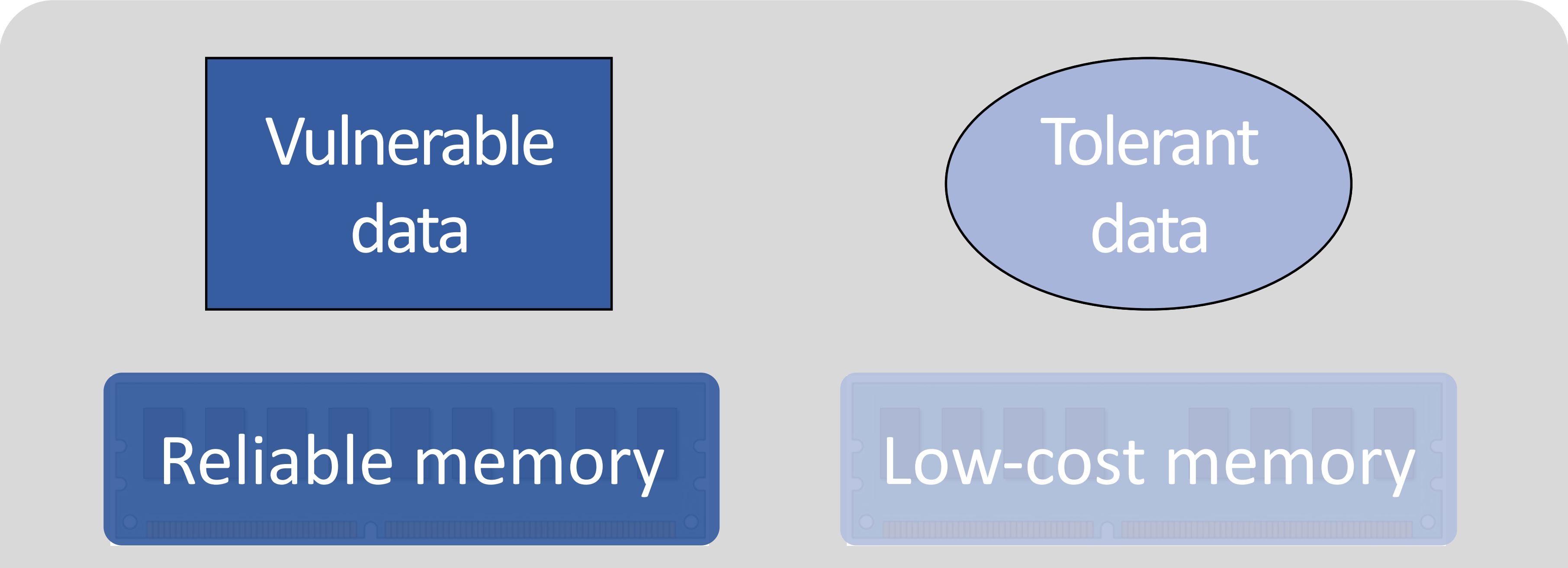
- Yixin Luo, Sriram Govindan, Bikash Sharma, Mark Santaniello, Justin Meza, Aman Kansal, Jie Liu, Badriddine Khessib, Kushagra Vaid, and Onur Mutlu, **"Characterizing Application Memory Error Vulnerability to Optimize Data Center Cost via Heterogeneous-Reliability Memory"** *Proceedings of the 44th Annual 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

Exploiting Memory Error Tolerance with Hybrid Memory Systems



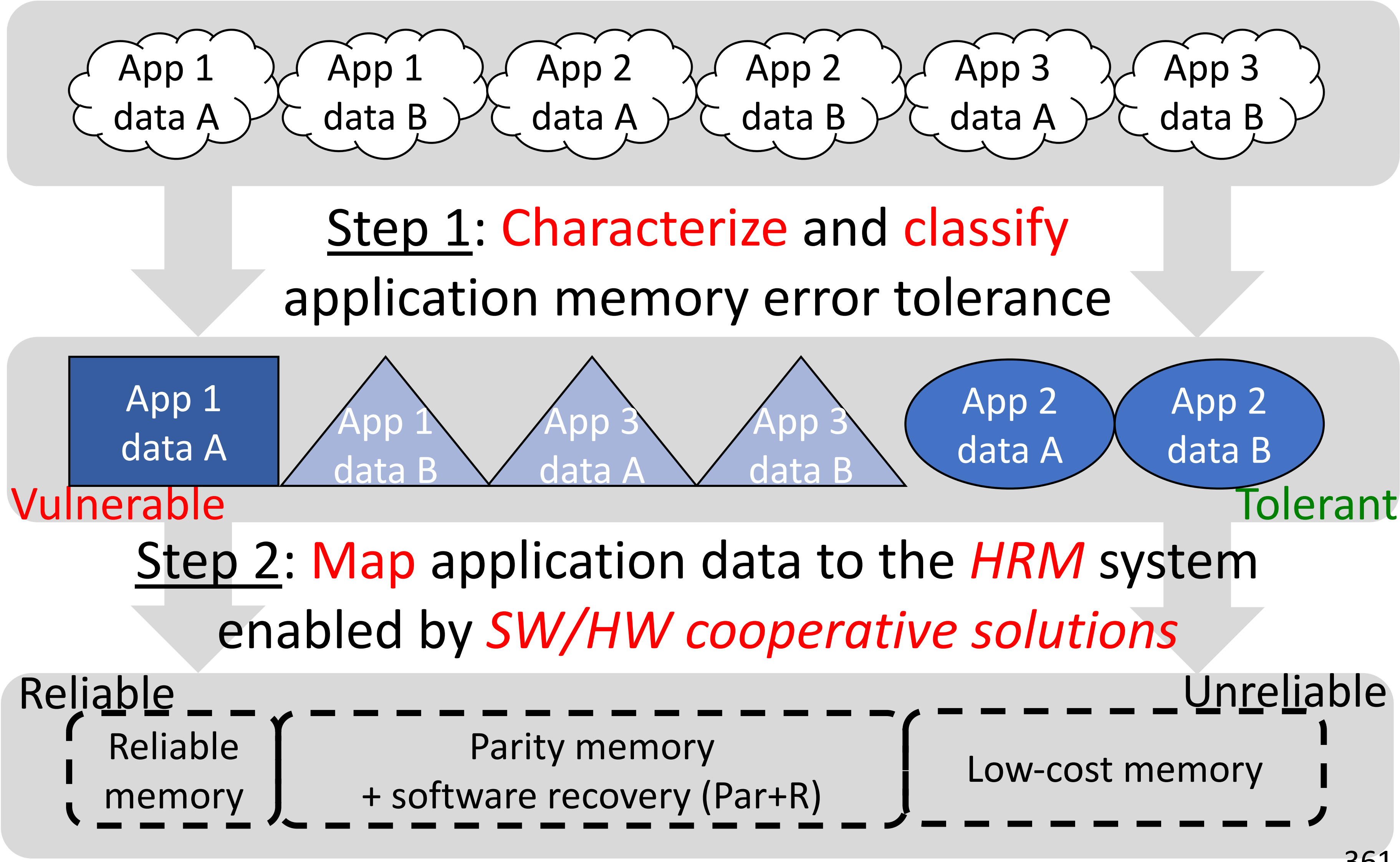
On Microsoft's Web Search workload

Reduces server hardware cost by 4.7 %

Achieves single server availability target of 99.90 %

Heterogeneous-**R**eliability **M**emory **[DSN 2014]**

Heterogeneous-Reliability Memory



App 1 data A

App 1 data B

App 2 data A

App 2 data B

App 3 data A

App 3 data B

Step 1: Characterize and classify application memory error tolerance

App 1 data A

App 1 data B

App 3 data A

App 3 data B

App 2 data A

App 2 data B

Vulnerable

Tolerant

Step 2: Map application data to the *HRM* system enabled by *SW/HW cooperative solutions*

Reliable

Unreliable

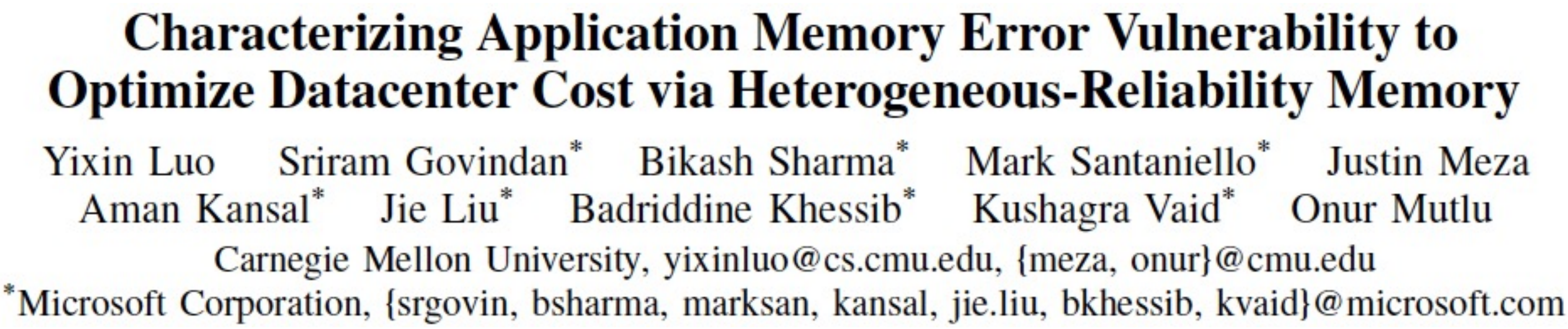
Reliable memory

Parity memory + software recovery (Par+R)

Low-cost memory

More on Heterogeneous-Reliability Memory

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



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

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

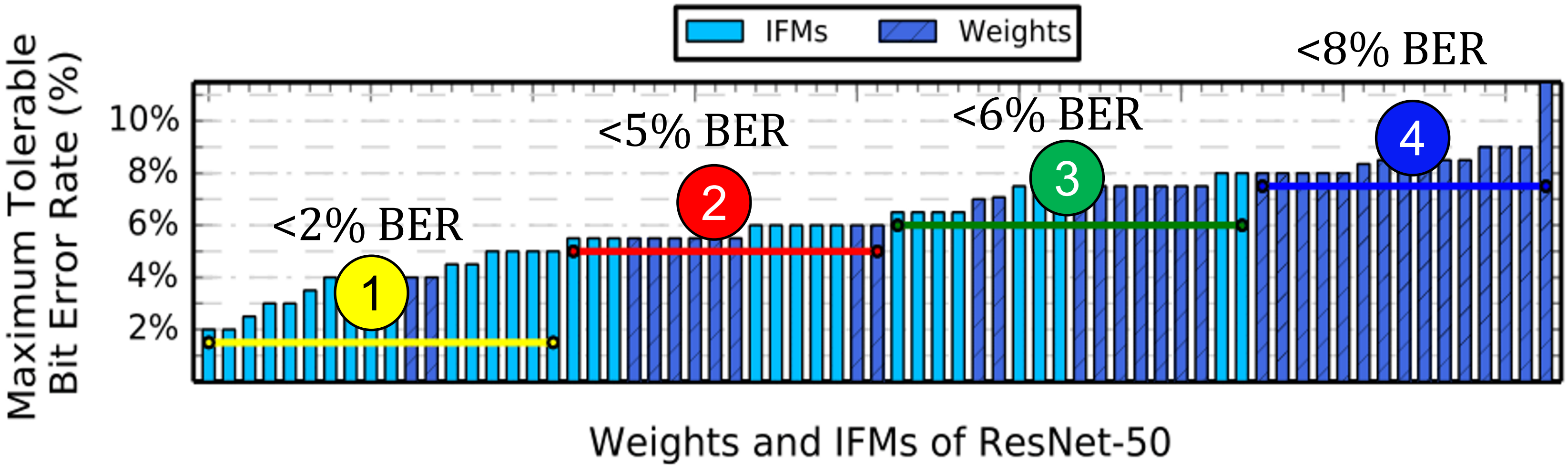
2. Reduce DRAM latency and voltage on such data and layers

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:



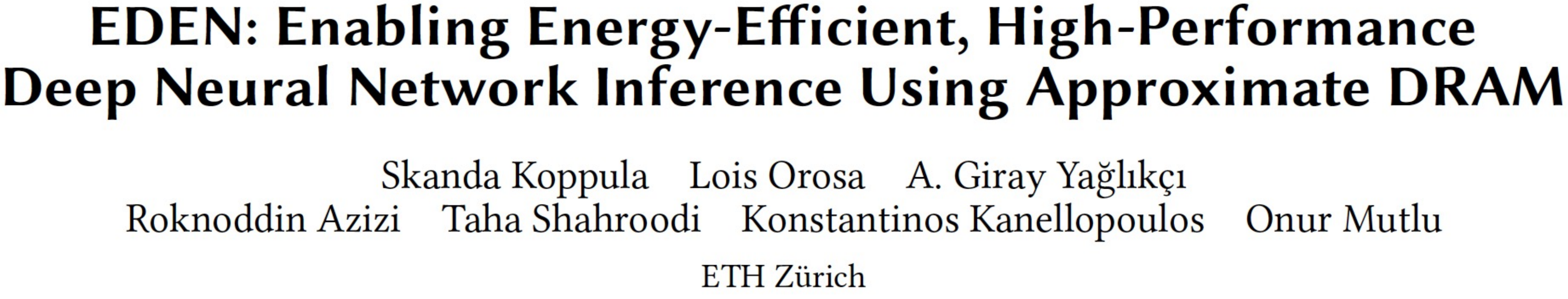
Map more error-tolerant DNN layers

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

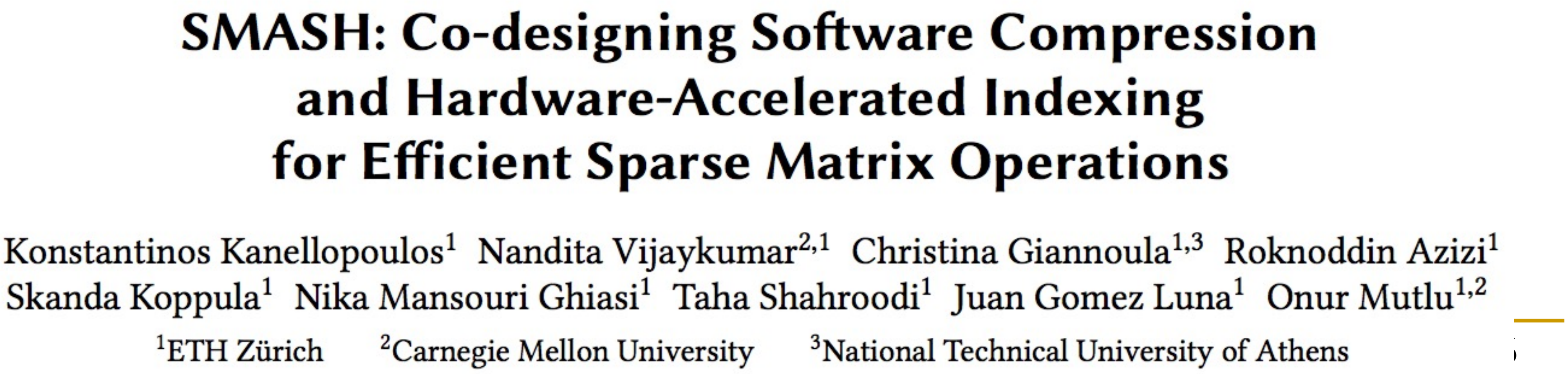


EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM

Skanda Koppula Lois Orosa A. Giray Yağlıkçı
Roknoddin Azizi Taha Shahroodi Konstantinos Kanellopoulos Onur Mutlu
ETH Zürich

SMASH: SW/HW Indexing Acceleration

- Konstantinos Kanellopoulos, Nandita Vijaykumar, Christina Giannoula, Roknoddin Azizi, Skanda Koppula, Nika Mansouri Ghiasi, Taha Shahroodi, Juan Gomez-Luna, and Onur Mutlu,
"SMASH: Co-designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations"
Proceedings of the 52nd International Symposium on Microarchitecture (***MICRO***), Columbus, OH, USA, October 2019.
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Poster (pptx) (pdf)]
[Lightning Talk Video (90 seconds)]
[Full Talk Lecture (30 minutes)]

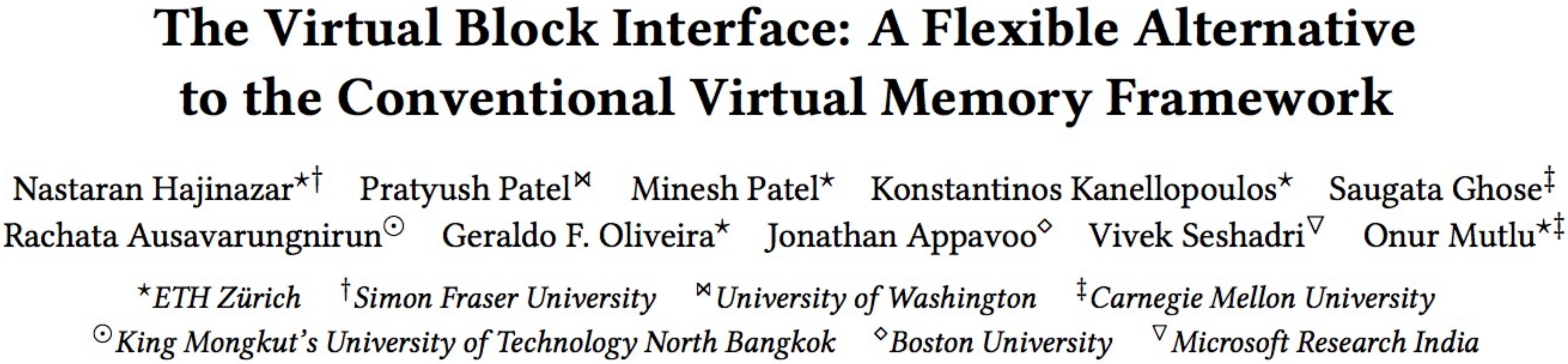


SMASH: Co-designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations

Konstantinos Kanellopoulos[1] Nandita Vijaykumar[2,1] Christina Giannoula[1,3] Roknoddin Azizi[1]
Skanda Koppula[1] Nika Mansouri Ghiasi[1] Taha Shahroodi[1] Juan Gomez Luna[1] Onur Mutlu[1,2]
[1]ETH Zürich [2]Carnegie Mellon University [3]National Technical University of Athens

Data-Aware Virtual Memory Framework

Nastaran Hajinazar, Pratyush Patel, Minesh Patel, Konstantinos Kanellopoulos, Saugata Ghose, Rachata Ausavarungnirun, Geraldo Francisco de Oliveira Jr., Jonathan Appavoo, Vivek Seshadri, and Onur Mutlu,
"The Virtual Block Interface: A Flexible Alternative to the Conventional Virtual Memory Framework"
Proceedings of the 47th International Symposium on Computer Architecture (***ISCA***), Virtual, June 2020.
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[ARM Research Summit Poster (pptx) (pdf)]
[Talk Video (26 minutes)]
[Lightning Talk Video (3 minutes)]
[Lecture Video (43 minutes)]



The Virtual Block Interface: A Flexible Alternative to the Conventional Virtual Memory Framework

Nastaran Hajinazar[⋆†] Pratyush Patel[⋈] Minesh Patel[⋆] Konstantinos Kanellopoulos[⋆] Saugata Ghose[‡]
Rachata Ausavarungnirun[⊙] Geraldo F. Oliveira[⋆] Jonathan Appavoo[⋄] Vivek Seshadri[▽] Onur Mutlu[⋆‡]

[⋆]*ETH Zürich* [†]*Simon Fraser University* [⋈]*University of Washington* [‡]*Carnegie Mellon University*
[⊙]*King Mongkut's University of Technology North Bangkok* [⋄]*Boston University* [▽]*Microsoft Research India*

Challenge and Opportunity for Future

Data-Aware

(Expressive)

Computing Architectures

Concluding Remarks

Recap: Corollaries: Architectures Today

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

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

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

Concluding Remarks

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

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

- Enable computation capability inside and close to memory

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

Fundamentally Better Architectures

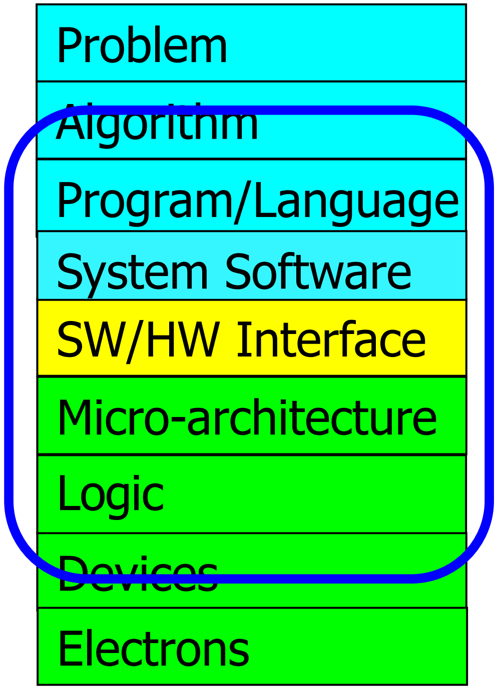
Data-centric

Data-driven

Data-aware

Source: http://spectrum.ieee.org/image/MjYzMzAyMg.jpeg

We Need to Revisit the Entire Stack



Problem

Algorithm

Program/Language

System Software

SW/HW Interface

Micro-architecture

Logic

Devices

Electrons

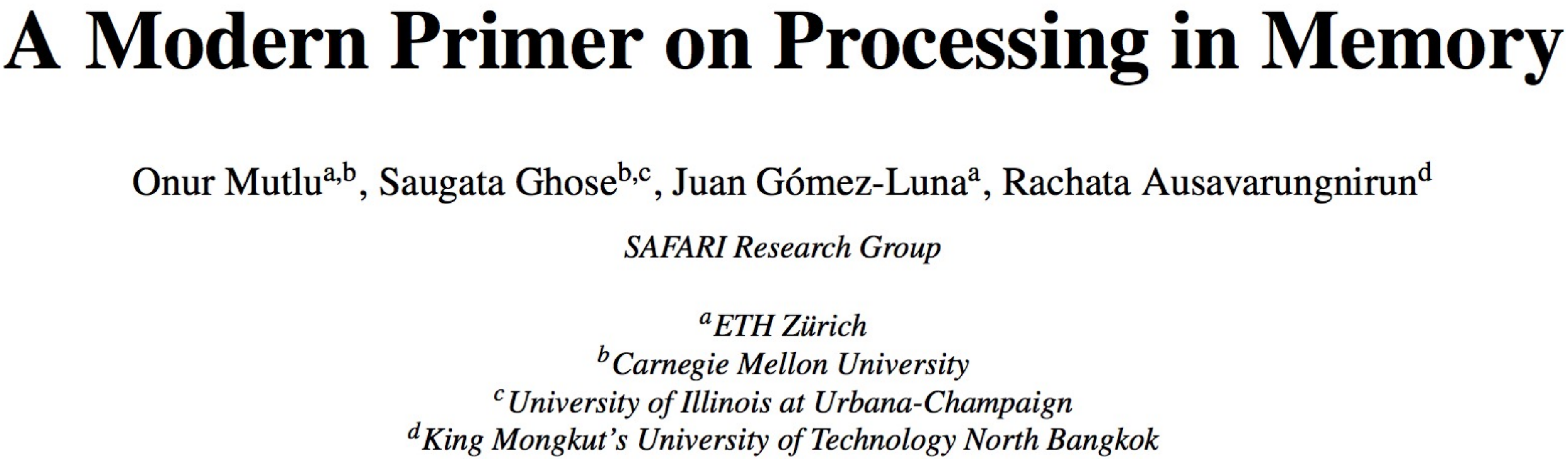
We can get there step by step

We Need to Exploit Good Principles

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

Open minds

PIM Review and Open Problems



A Modern Primer on Processing in Memory

Onur Mutlu[a,b], Saugata Ghose[b,c], Juan Gómez-Luna[a], Rachata Ausavarungnirun[d]

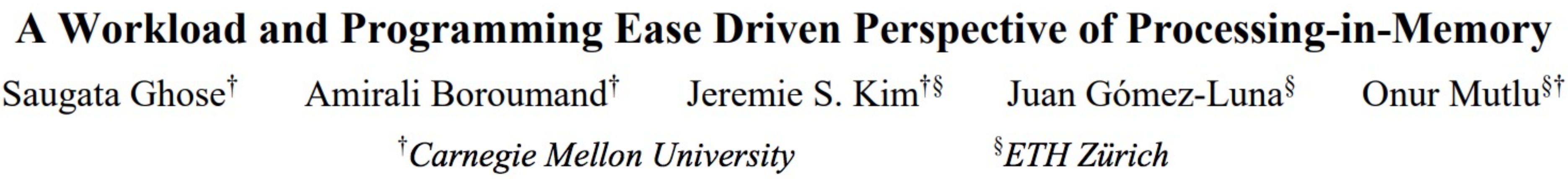
SAFARI Research Group

[a] *ETH Zürich*
[b] *Carnegie Mellon University*
[c] *University of Illinois at Urbana-Champaign*
[d] *King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,
"A Modern Primer on Processing in Memory"
Invited Book Chapter in ***Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann***, Springer, to be published in 2021.

https://arxiv.org/pdf/1903.03988.pdf

PIM Review and Open Problems (II)



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

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

[†]*Carnegie Mellon University* [§]*ETH Zürich*

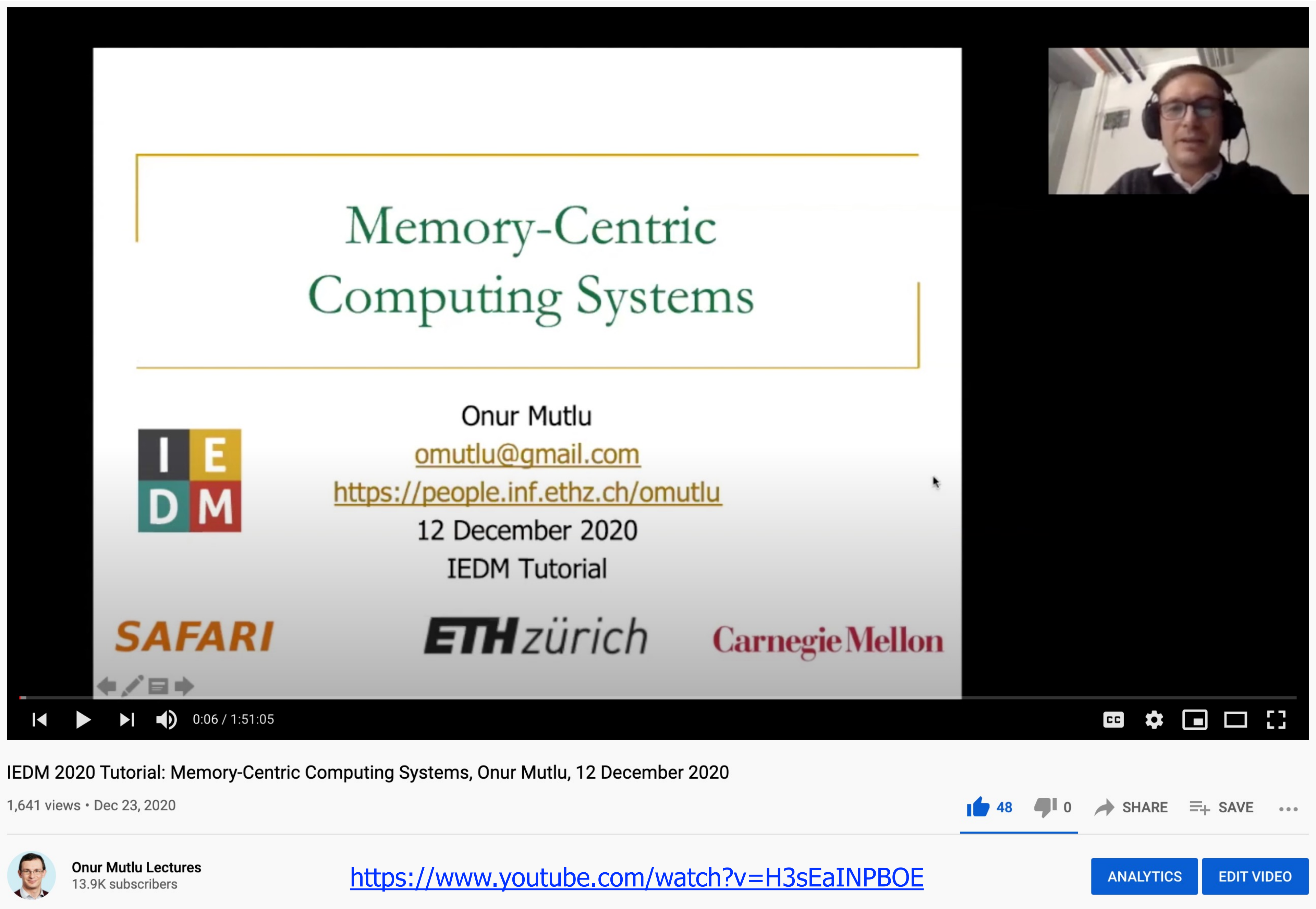
Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu, **"Processing-in-Memory: A Workload-Driven Perspective"** *Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence*, to appear in November 2019. [Preliminary arXiv version]

https://arxiv.org/pdf/1907.12947.pdf

A Longer Tutorial Version of This Talk

- Onur Mutlu,
  "Memory-Centric Computing Systems"
  Invited Tutorial at *66th International Electron Devices Meeting (**IEDM**)*, Virtual, 12 December 2020.
  [Slides (pptx) (pdf)]
  [Executive Summary Slides (pptx) (pdf)]
  [Tutorial Video (1 hour 51 minutes)]
  [Executive Summary Video (2 minutes)]
  [Abstract and Bio]
  [Related Keynote Paper from VLSI-DAT 2020]
  [Related Review Paper on Processing in Memory]

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



Memory-Centric Computing Systems

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

IEDM 2020 Tutorial: Memory-Centric Computing Systems, Onur Mutlu, 12 December 2020

1,641 views • Dec 23, 2020

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

https://www.youtube.com/onurmutlulectures

Detailed Lectures on PIM (I)

- Computer Architecture, Fall 2020, Lecture 6
  - **Computation in Memory** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=oGcZAGwfEUE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=12
- Computer Architecture, Fall 2020, Lecture 7
  - **Near-Data Processing** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=j2GIigqn1Qw&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=13
- Computer Architecture, Fall 2020, Lecture 11a
  - **Memory Controllers** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=TeG773OgiMQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=20
- Computer Architecture, Fall 2020, Lecture 12d
  - **Real Processing-in-DRAM with UPMEM** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=Sscy1Wrr22A&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=25

https://www.youtube.com/onurmutlulectures

Detailed Lectures on PIM (II)

- Computer Architecture, Fall 2020, Lecture 15
  - **Emerging Memory Technologies** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=AlE1rD9G_YU&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=28

- Computer Architecture, Fall 2020, Lecture 16a
  - **Opportunities & Challenges of Emerging Memory Technologies** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=pmLszWGmMGQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=29

- Computer Architecture, Fall 2020, Guest Lecture
  - **In-Memory Computing: Memory Devices & Applications** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=wNmqQHiEZNk&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=41

https://www.youtube.com/onurmutlulectures

Funding Acknowledgments

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

Acknowledgments

SAFARI

SAFARI Research Group

safari.ethz.ch

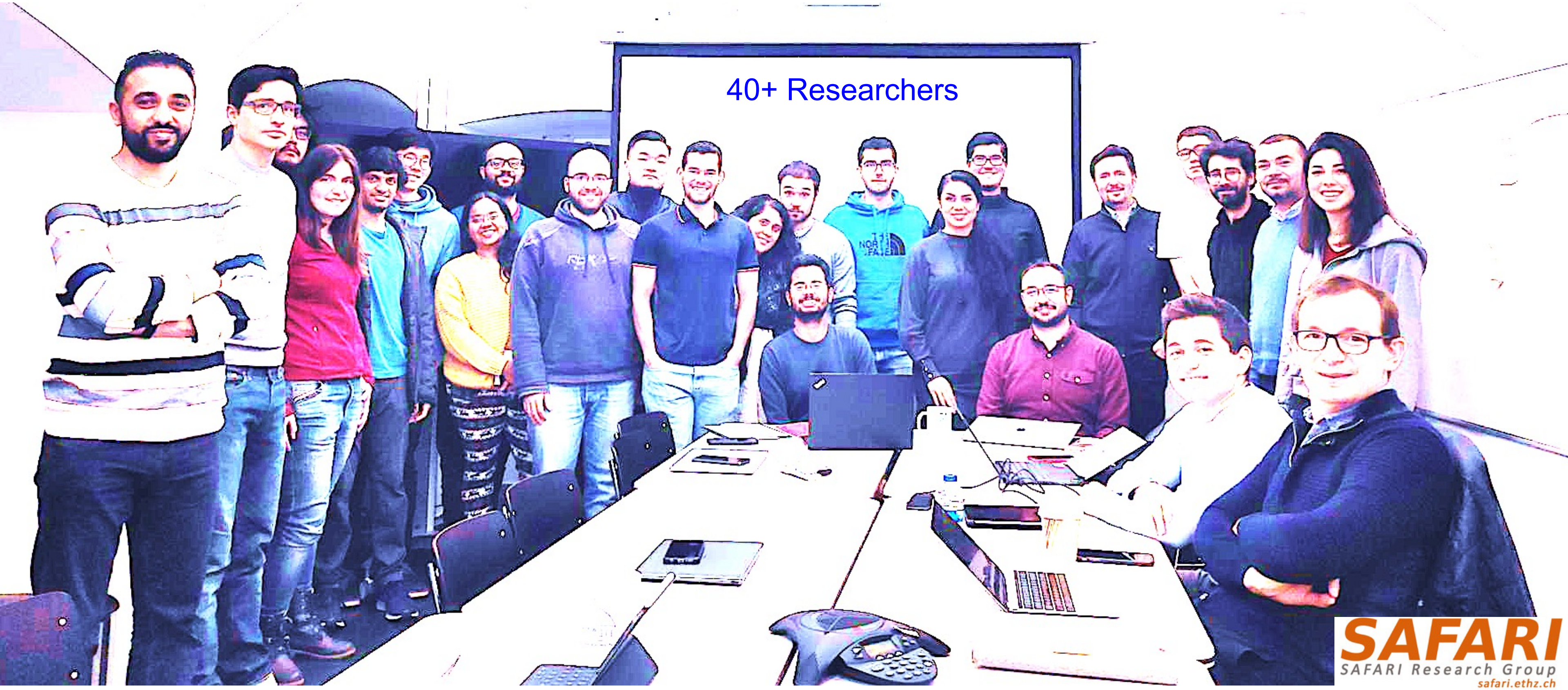
Think BIG, Aim HIGH!

https://safari.ethz.ch

Onur Mutlu's SAFARI Research Group

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

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

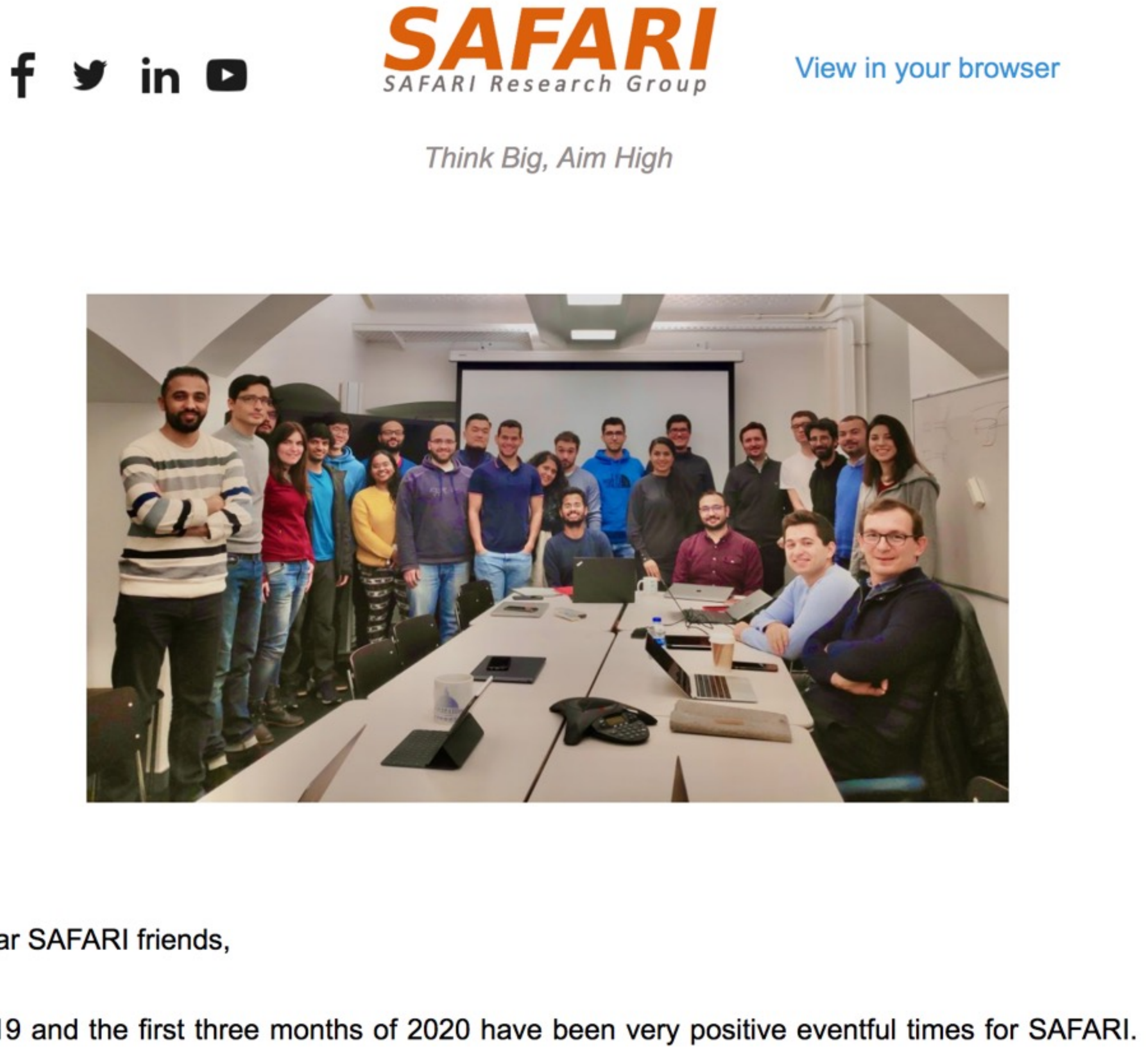


Think BIG, Aim HIGH!

https://safari.ethz.ch

SAFARI Newsletter April 2020 Edition

- https://safari.ethz.ch/safari-newsletter-april-2020/



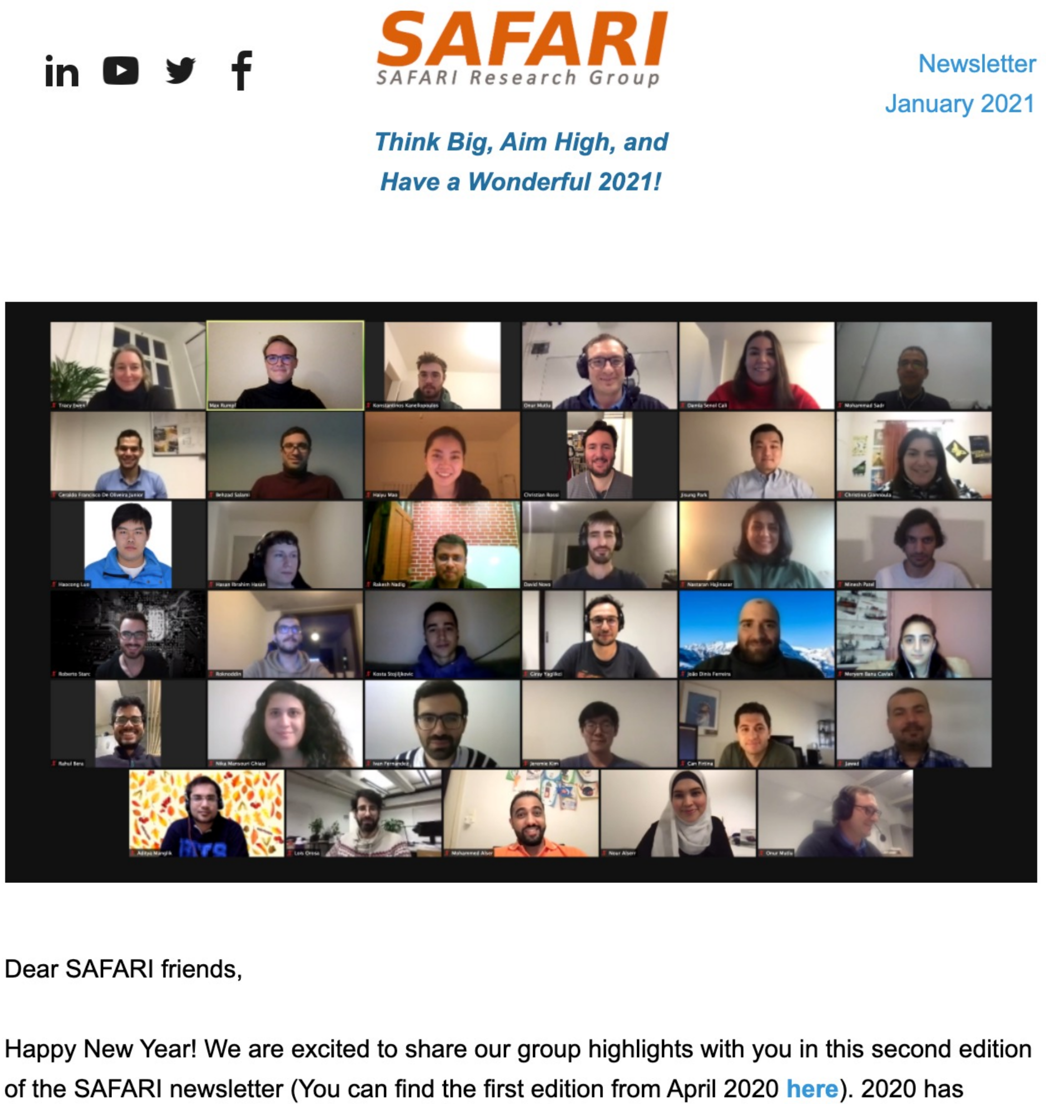
Think Big, Aim High

Dear SAFARI friends,

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

SAFARI Newsletter January 2021 Edition

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



SAFARI
SAFARI Research Group

Newsletter
January 2021

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

Dear SAFARI friends,

Happy New Year! We are excited to share our group highlights with you in this second edition of the SAFARI newsletter (You can find the first edition from April 2020 **here**). 2020 has

Referenced Papers, Talks, Artifacts

- All are available at

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

  https://www.youtube.com/onurmutlulectures

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

Memory-Centric Computing Systems

Onur Mutlu

omutlu@gmail.com

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

9 October 2021

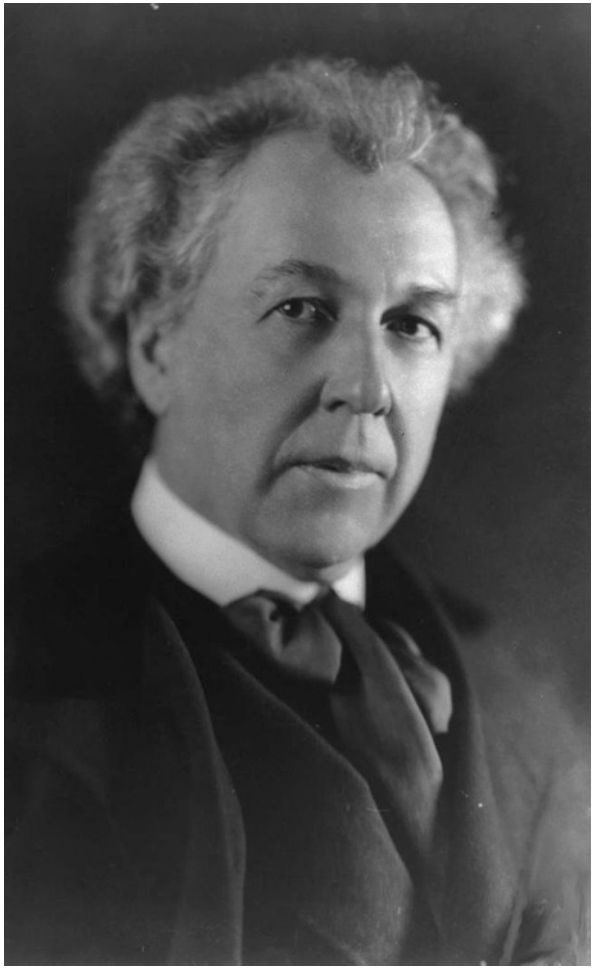
ESWEEK Education Class

SAFARI ETH zürich Carnegie Mellon

Backup Slides

A Quote from A Famous Architect

- “architecture […] based upon principle, and not upon precedent”

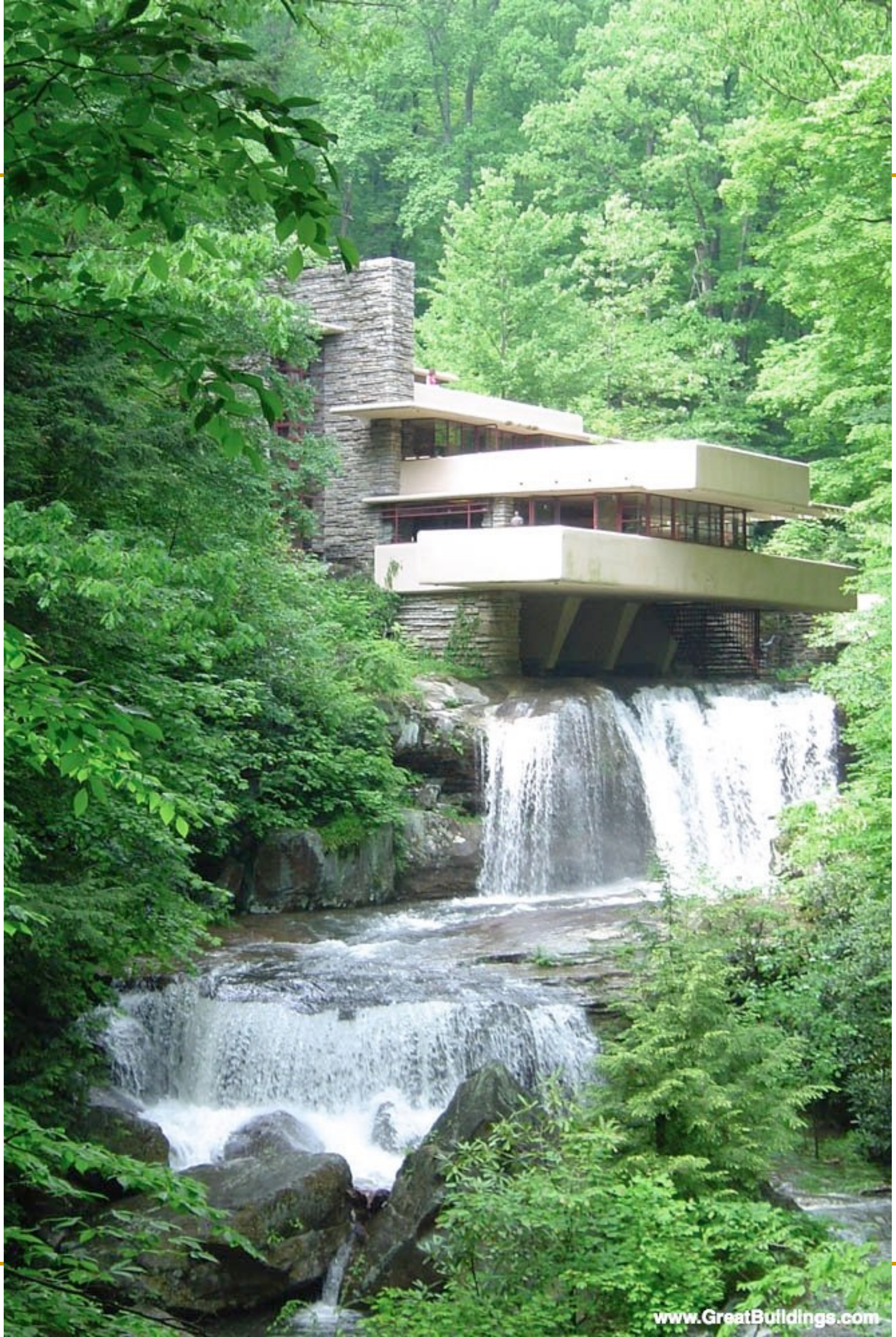


Precedent-Based Design?

- “architecture […] based upon principle, and not upon precedent”

Principled Design

- “architecture […] based upon principle, and not upon precedent”



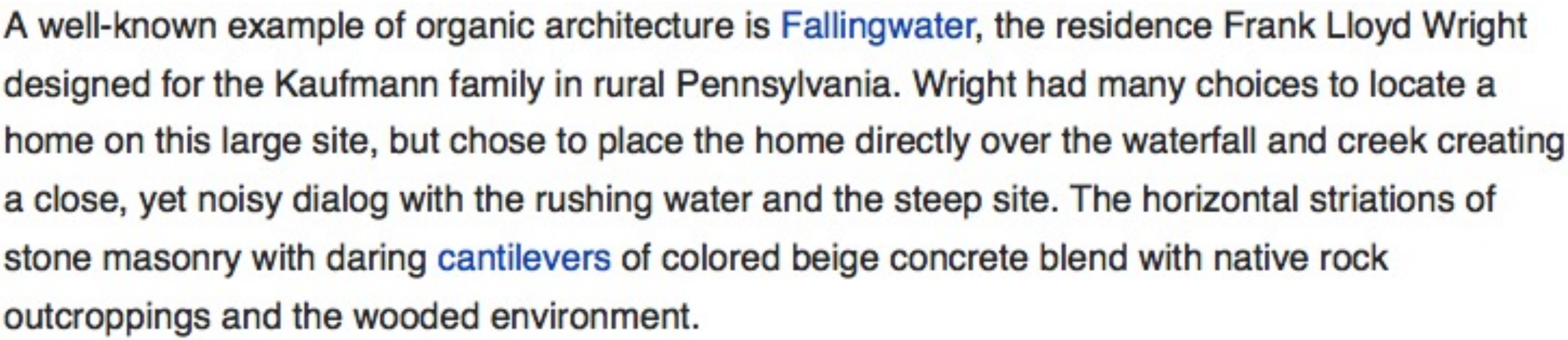
www.GreatBuildings.com

The Overarching Principle

Organic architecture

From Wikipedia, the free encyclopedia

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



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

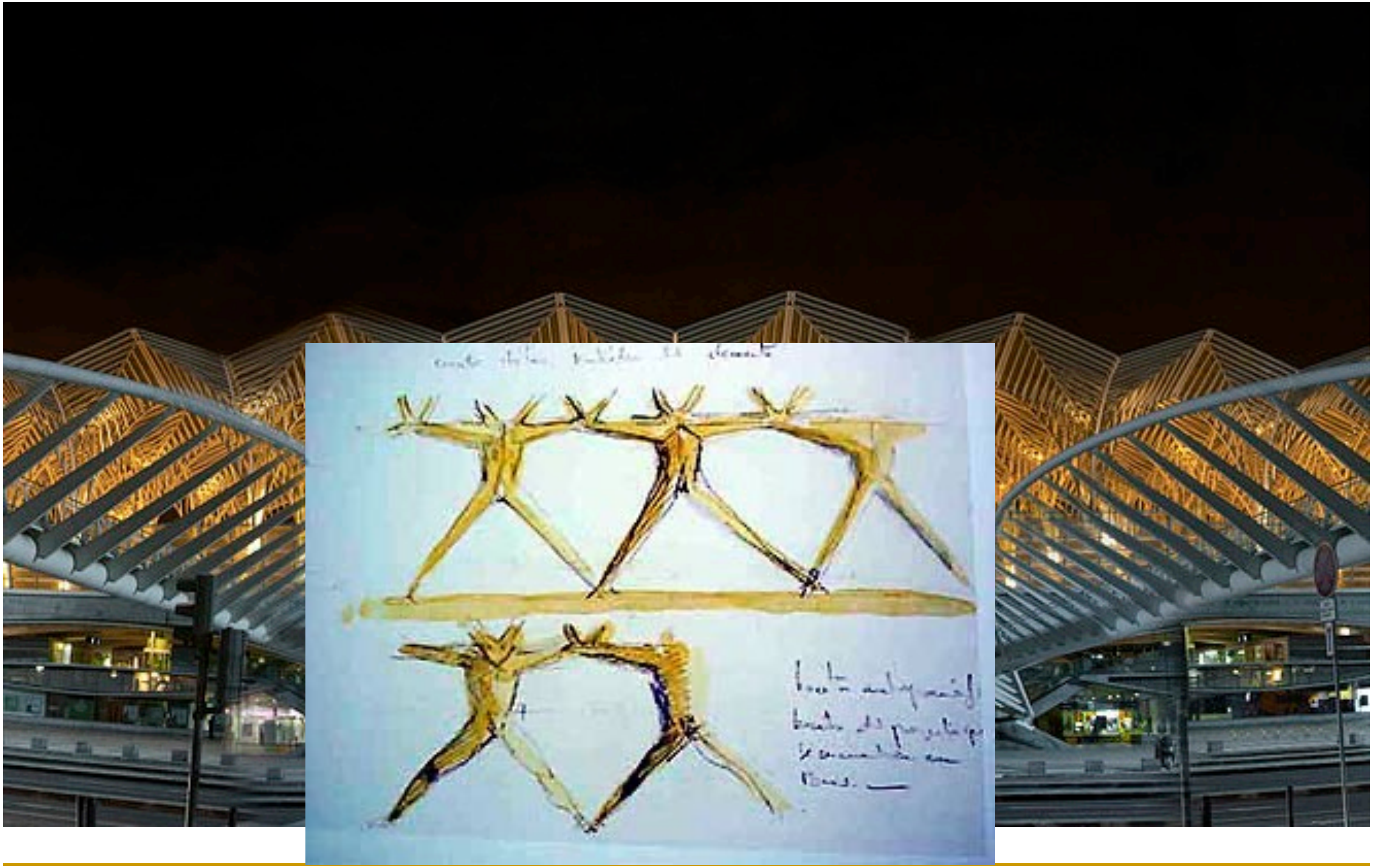
Another Example: Precedent-Based Design

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

Principled Design

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

Another Principled Design



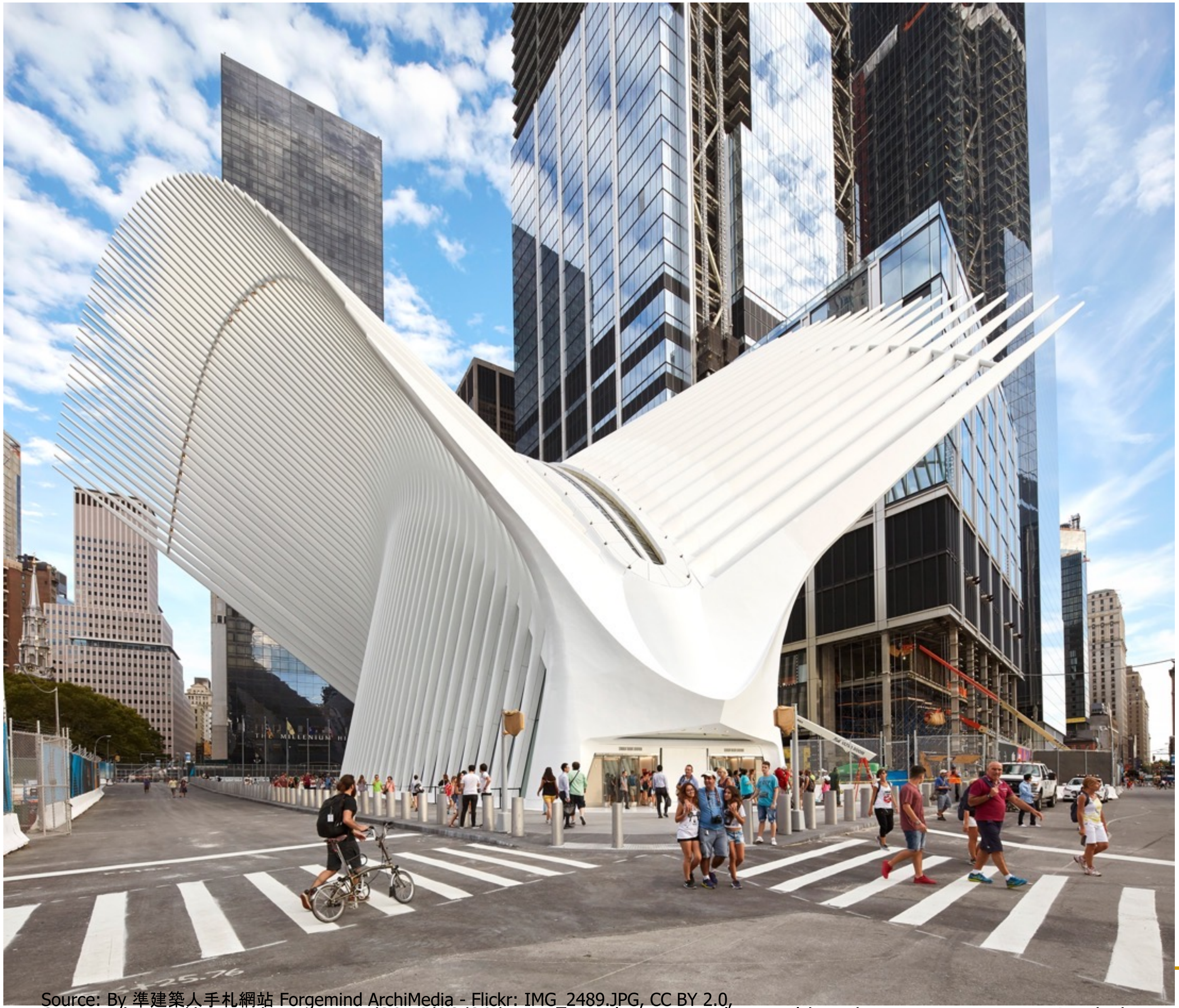
Source: By Martín Gómez Tagle - Lisbon, Portugal, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=13764903

Source: http://www.arcspace.com/exhibitions/unsorted/santiago-calatrava/

Another Principled Design

Source: CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=172107

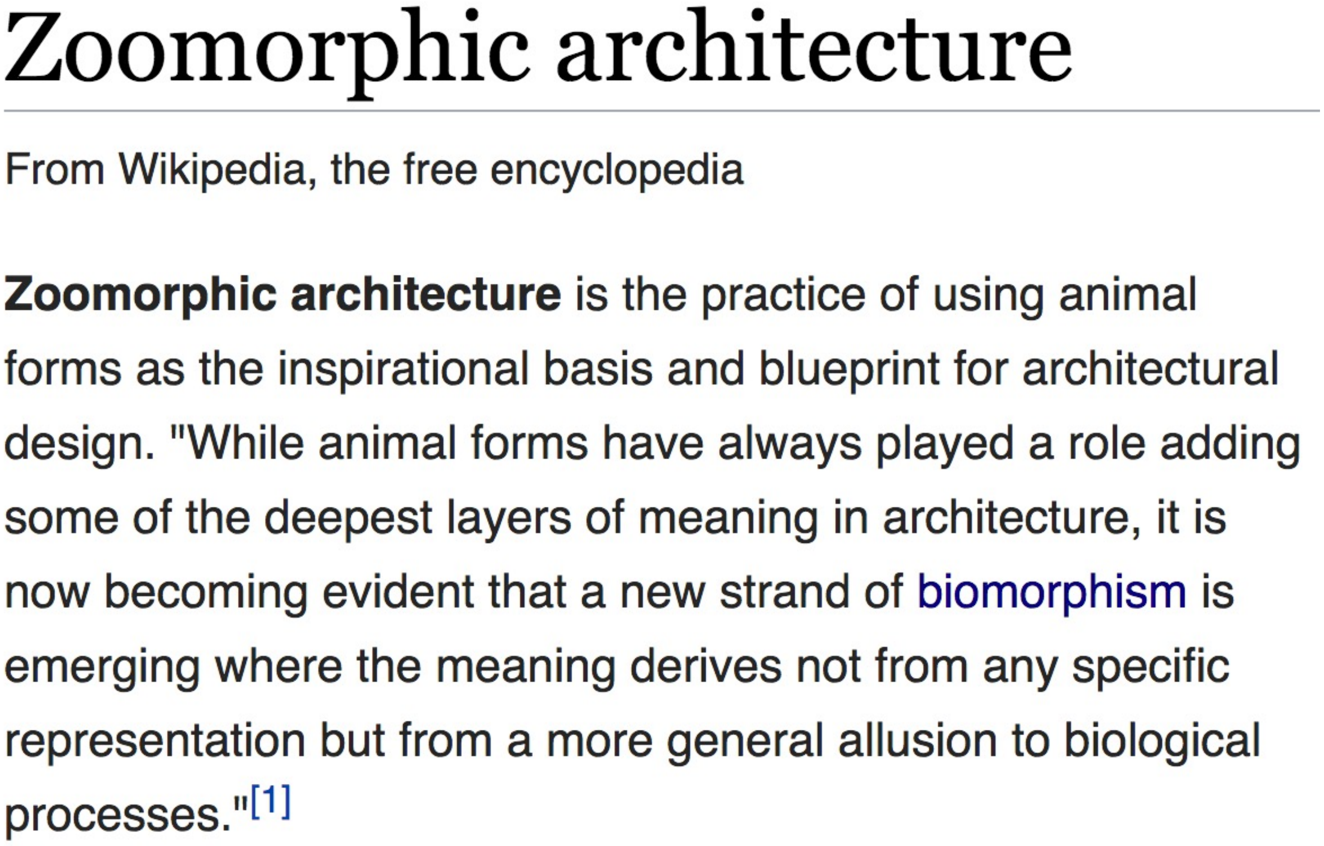
Principle Applied to Another Structure



Source: By 準建築人手札網站 Forgemind ArchiMedia - Flickr: IMG_2489.JPG, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=51493396, https://en.wikipedia.org/wiki/Santiago_Calatrava

Source: https://www.dezeen.com/2016/08/29/santiago-calatrava-oculus-world-trade-center-transportation-hub-new-york-photographs-hufton-crow/

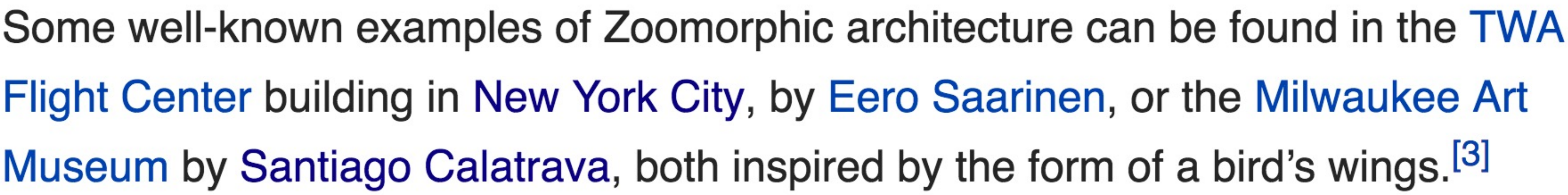
The Overarching Principle



Zoomorphic architecture

From Wikipedia, the free encyclopedia

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



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

Overarching Principles for Computing?

Source: http://spectrum.ieee.org/image/MjYzMzAyMg.jpeg

Readings, Videos, Reference Materials

List of References
(Incomplete but Hopefully Useful)

Overview Readings (I)

- Onur Mutlu,
  "Intelligent Architectures for Intelligent Machines"
  *Invited Keynote Paper in Proceedings of the 2020 International Symposia on VLSI (**VLSI**)*, Hsinchu City, Taiwan, August 2020.
  [Slides (pptx) (pdf)]
  [Keynote Talk Video (55 minutes)]

- Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun,
  "Processing Data Where It Makes Sense: Enabling In-Memory Computation"
  *Invited paper in Microprocessors and Microsystems (**MICPRO**)*, June 2019.
  [arXiv version]
  [Slides (pptx)]
  [Talk Video]

- Vivek Seshadri and Onur Mutlu,
  "In-DRAM Bulk Bitwise Execution Engine"
  Invited Book Chapter in Advances in Computers, to appear in 2020.
  [Preliminary arXiv version]

Overview Readings (II)

- Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu,
  "Processing-in-Memory: A Workload-Driven Perspective"
  Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence, to appear in November 2019.
  [Preliminary arXiv version]

- Onur Mutlu and Jeremie Kim,
  "RowHammer: A Retrospective"
  *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (**TCAD**) Special Issue on Top Picks in Hardware and Embedded Security*, 2019.
  [Preliminary arXiv version]
  [Slides from COSADE 2019 (pptx)]
  [Slides from VLSI-SOC 2020 (pptx) (pdf)]
  [Talk Video (30 minutes)]

- Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu,
  "Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery"
  Invited Book Chapter in Inside Solid State Drives, 2018.
  [Preliminary arxiv.org version]

Overview Readings (III)

- Onur Mutlu,
  "The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
  *Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**)*, Lausanne, Switzerland, March 2017.
  [Slides (pptx) (pdf)]

- Onur Mutlu,
  "Main Memory Scaling: Challenges and Solution Directions"
  Invited Book Chapter in More than Moore Technologies for Next Generation Computer Design, pp. 127-153, Springer, 2015.

- Onur Mutlu and Lavanya Subramanian,
  "Research Problems and Opportunities in Memory Systems"
  *Invited Article in Supercomputing Frontiers and Innovations (**SUPERFRI**)*, 2014.

- 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]

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://people.inf.ethz.ch/omutlu/acaces2018.html
  - https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x

- All Papers are at:
  - 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://safari.ethz.ch/memory_systems/TUWien2019
  - https://www.youtube.com/playlist?list=PL5Q2soXY2Zi_gntM55VoMlKlw7YrXOhbl

- All Papers are at:
  - https://people.inf.ethz.ch/omutlu/projects.htm
  - Final lecture notes and readings (for all topics)

All Referenced Works Can Be Found At

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

- Includes PDFs, presentations, talk videos, etc.

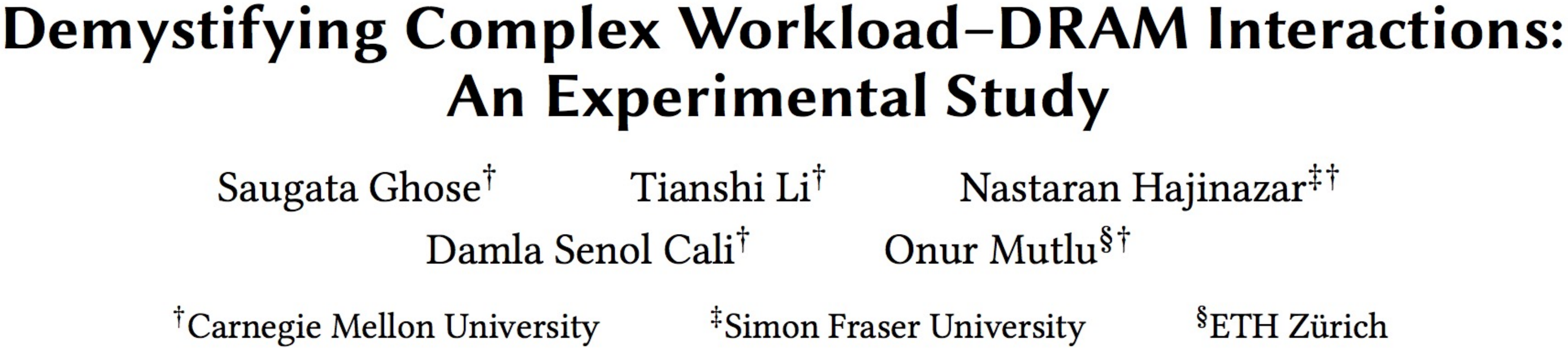
- Many paper and course lecture videos are here:
  - **https://www.youtube.com/OnurMutluLectures**

- Please email me with any questions, feedback, etc.
  - **omutlu@gmail.com**

Low-Latency Memory

Workload-DRAM Interaction Analysis

- Saugata Ghose, Tianshi Li, Nastaran Hajinazar, Damla Senol Cali, and Onur Mutlu,
"Demystifying Workload–DRAM Interactions: An Experimental Study"
*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, Phoenix, AZ, USA, June 2019.
[Preliminary arXiv Version]
[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

Why Study Workload–DRAM Interactions?

- Manufacturers are developing many new types of DRAM
  - **DRAM limits performance, energy improvements:** new types may overcome some limitations
  - Memory systems now serve a **very diverse set of applications:** can no longer take a one-size-fits-all approach

- **So which DRAM type works best with which application?**
  - Difficult to understand intuitively due to the complexity of the interaction
  - Can't be tested methodically on real systems: new type needs a new CPU

- We perform a wide-ranging experimental study to uncover the combined behavior of workloads and DRAM types
  - **115 prevalent/emerging applications and multiprogrammed workloads**
  - **9 modern DRAM types:** DDR3, DDR4, GDDR5, HBM, HMC, LPDDR3, LPDDR4, Wide I/O, Wide I/O 2

Modern DRAM Types: Comparison to DDR3

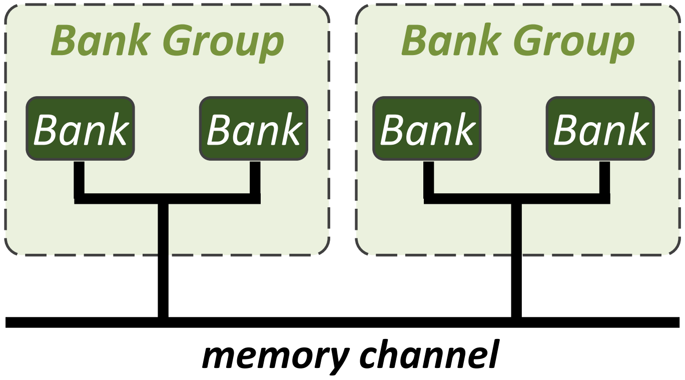
DRAM Type	Banks per Rank	Bank Groups	3D-Stacked	Low-Power
DDR3	8			
DDR4	16	✓		
GDDR5	16	✓		
HBM High-Bandwidth Memory	16		✓	
HMC Hybrid Memory Cube	256		✓	
Wide I/O	4		✓	✓
Wide I/O 2	8		✓	✓
LPDDR3	8			✓
LPDDR4	16			✓

increased latency

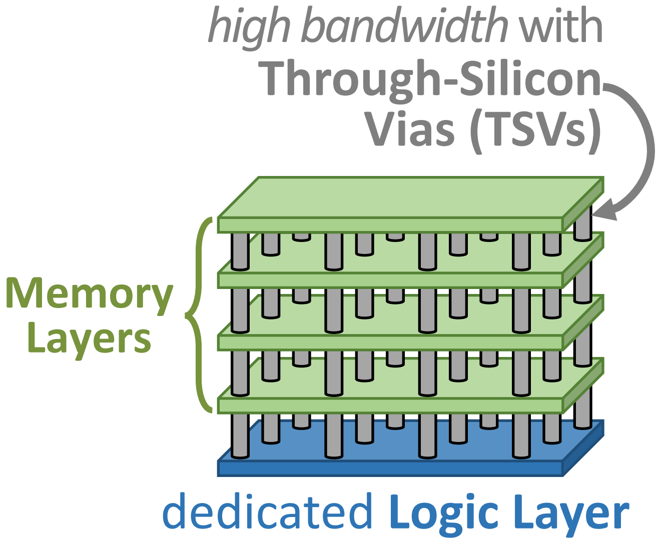
increased area/power

narrower rows, higher latency

- Bank groups

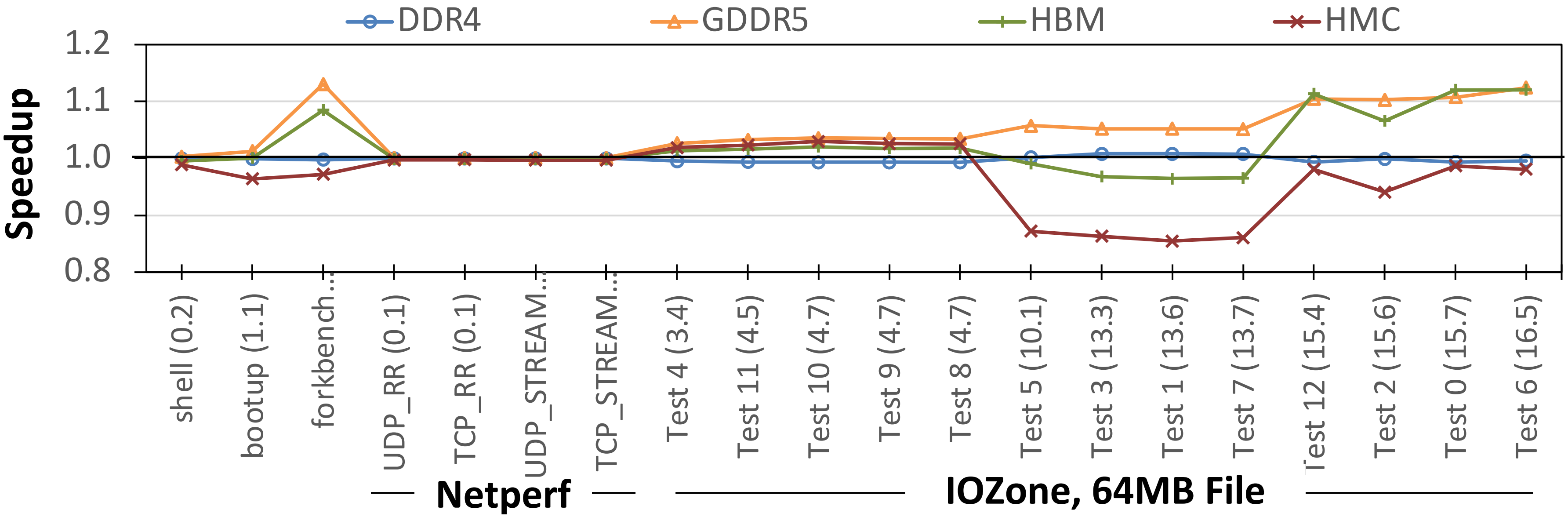


- 3D-stacked DRAM



4. Need for Lower Access Latency: Performance

- New DRAM types often increase access latency in order to provide more banks, higher throughput
- Many applications can't make up for the increased latency
  - Especially true of common OS routines (e.g., file I/O, process forking)



  - A variety of desktop/scientific, server/cloud, GPGPU applications

Several applications don't benefit from more parallelism

Key Takeaways

1. DRAM latency remains a critical bottleneck for many applications

2. Bank parallelism is not fully utilized by a wide variety of applications

3. Spatial locality continues to provide significant performance benefits if it is exploited by the memory subsystem

4. For some classes of applications, low-power memory can provide energy savings without sacrificing significant performance

Conclusion

- Manufacturers are developing many new types of DRAM
  - DRAM limits performance, energy improvements: new types may overcome some limitations
  - Memory systems now serve a very diverse set of applications: can no longer take a one-size-fits-all approach
  - Difficult to intuitively determine which DRAM–workload pair works best

- We perform a wide-ranging experimental study to uncover the combined behavior of workloads, DRAM types
  - 115 prevalent/emerging applications and multiprogrammed workloads
  - 9 modern DRAM types

- 12 key observations on DRAM–workload behavior

Open-source tools: https://github.com/CMU-SAFARI/ramulator

Full paper: https://arxiv.org/pdf/1902.07609

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]

None of These Fundamentally Reduce Memory Latency

e-

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

Two Major Sources of Latency Inefficiency

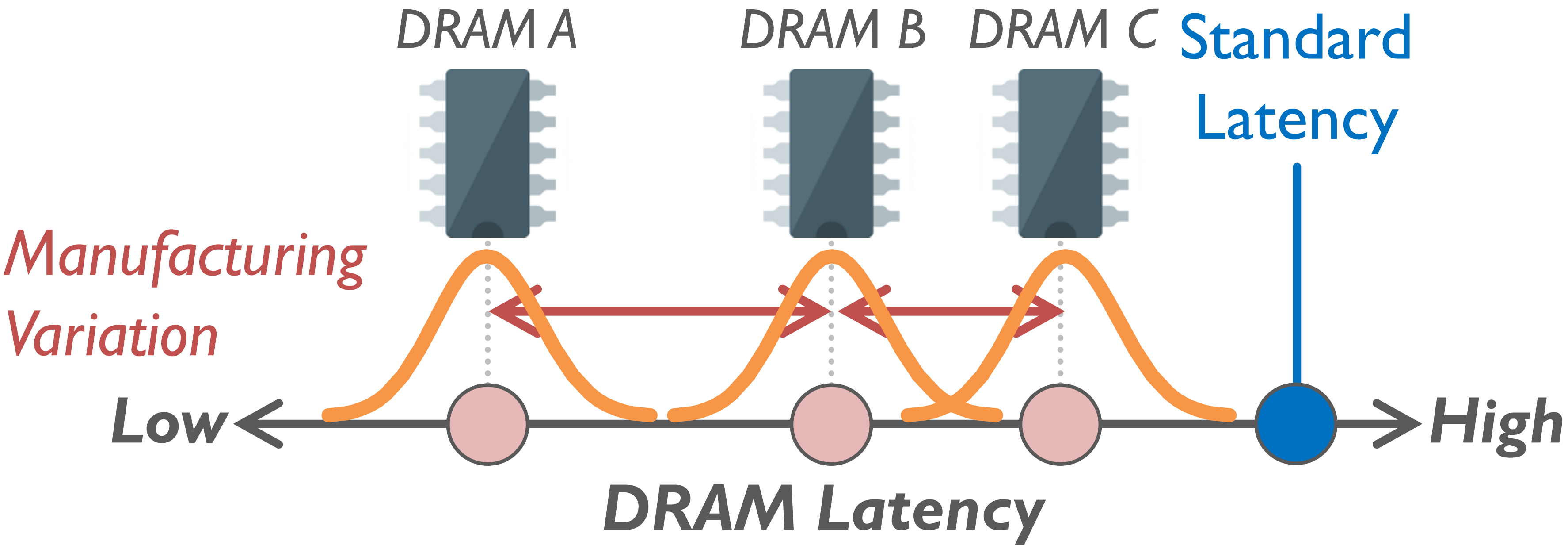
- Modern DRAM is **not** designed for low latency
  - Main focus is cost-per-bit (capacity)

- 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 → latency variation
- **High standard latency** chosen to increase yield



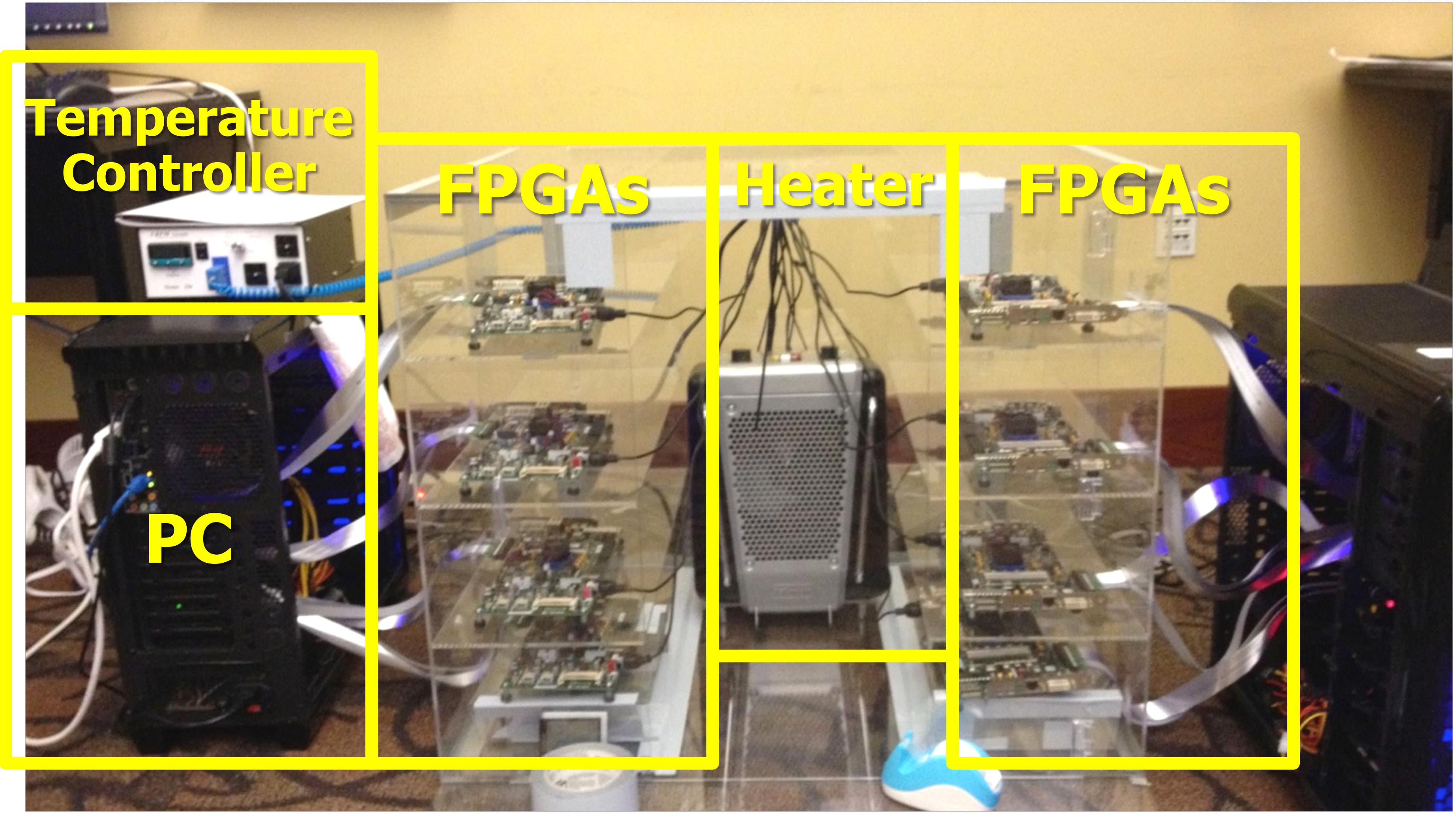
Adaptive-Latency DRAM

- *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

Lee+, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," HPCA 2015.

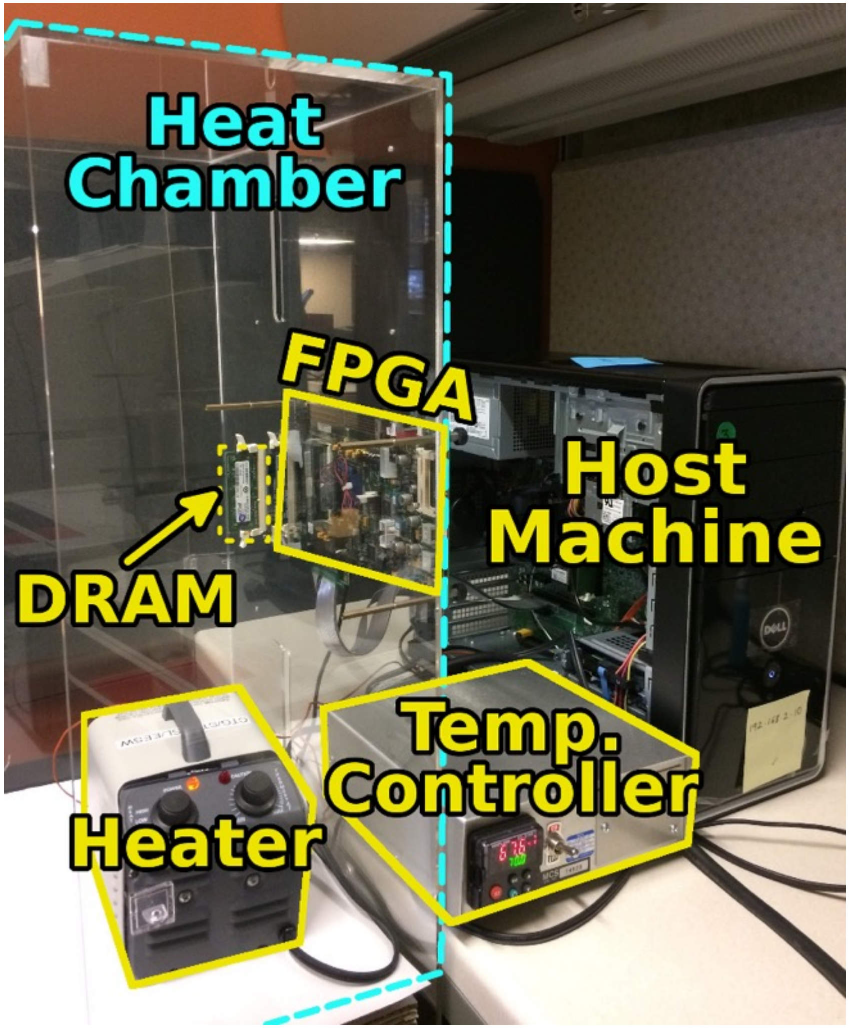
Infrastructures to Understand Such Issues



Kim+, "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors," ISCA 2014.

SoftMC: Open Source DRAM Infrastructure

- Hasan Hassan et al., "**SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies**," HPCA 2017.

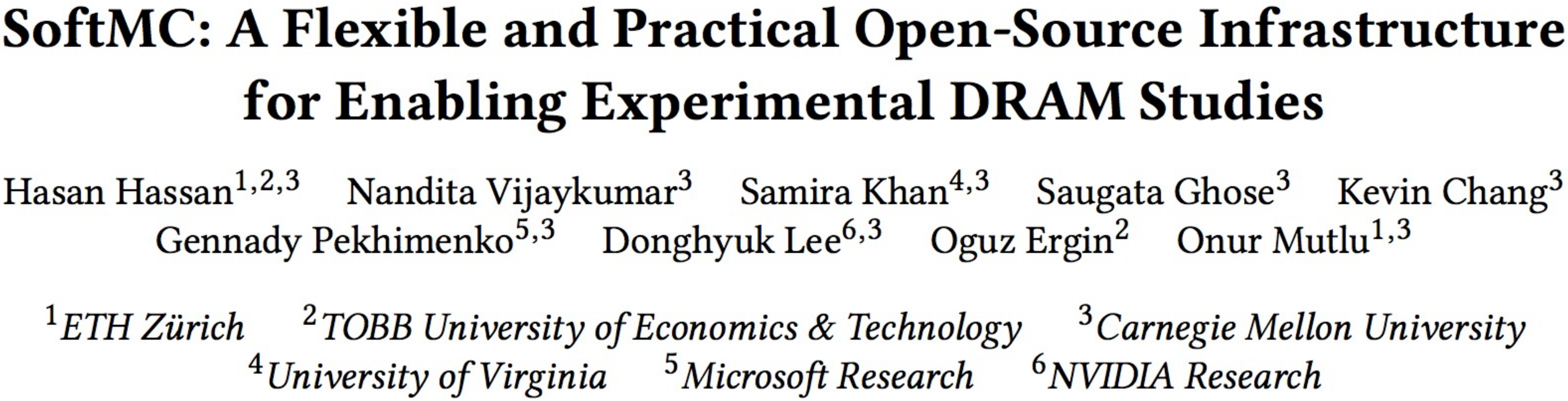


- **Flexible**
- **Easy to Use (C++ API)**
- **Open-source**

github.com/CMU-SAFARI/SoftMC

SoftMC

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



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

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

[1]*ETH Zürich* [2]*TOBB University of Economics & Technology* [3]*Carnegie Mellon University*
[4]*University of Virginia* [5]*Microsoft Research* [6]*NVIDIA Research*

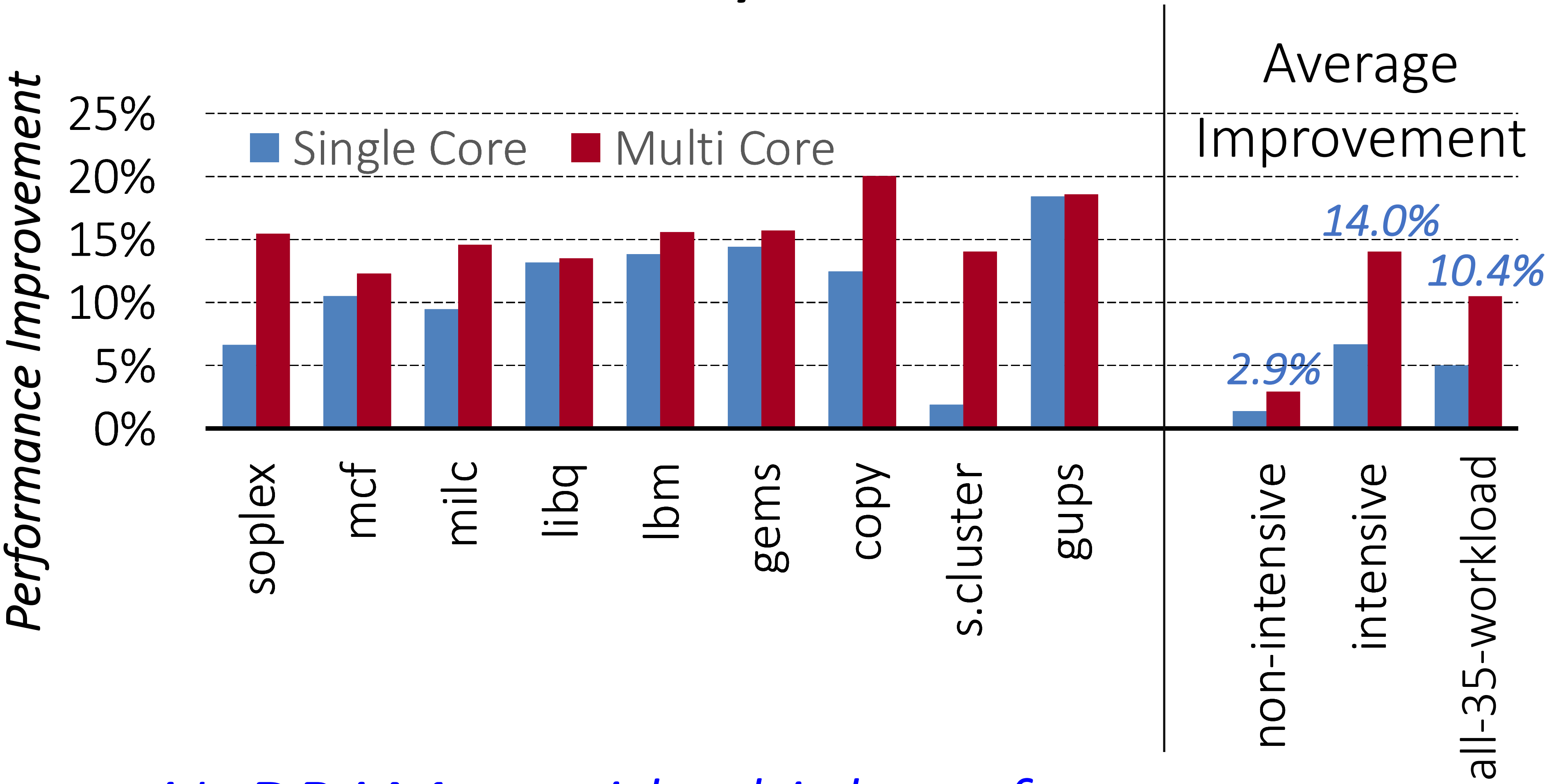
Latency Reduction Summary of 115 DIMMs

- *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%***

Lee+, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," HPCA 2015.

AL-DRAM: Real-System Performance



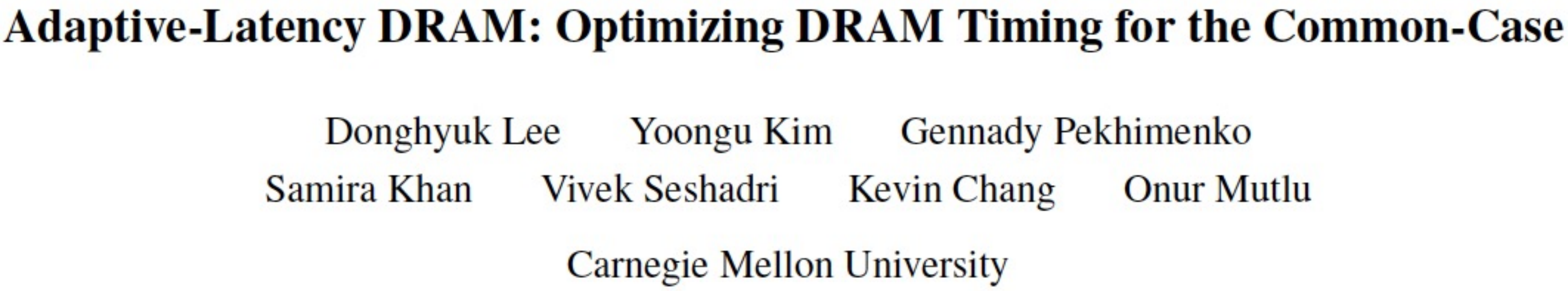
AL-DRAM provides high performance on ***memory-intensive workloads***

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"** *Proceedings of the 21st International Symposium on High-Performance Computer Architecture (**HPCA**)*, Bay Area, CA, February 2015.
  [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

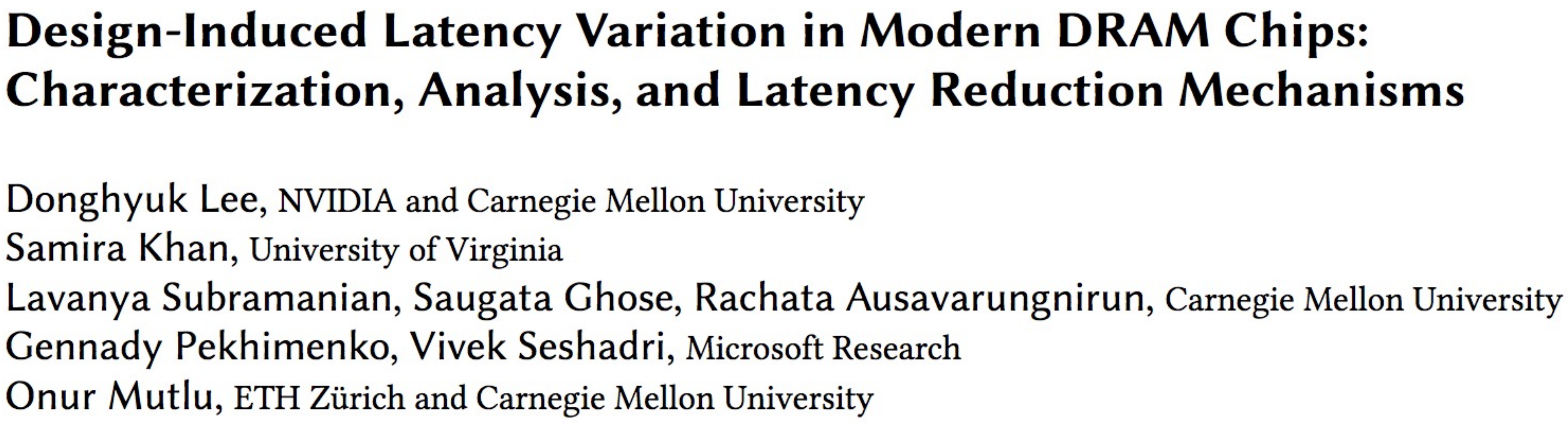
- Kevin Chang, Abhijith Kashyap, Hasan Hassan, Samira Khan, Kevin Hsieh, Donghyuk Lee, Saugata Ghose, Gennady Pekhimenko, Tianshi Li, and Onur Mutlu,
"Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization"
Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (***SIGMETRICS***), Antibes Juan-Les-Pins, France, June 2016.
[Slides (pptx) (pdf)]
[Source Code]

Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization

Kevin K. Chang[1] Abhijith Kashyap[1] Hasan Hassan[1,2]
Saugata Ghose[1] Kevin Hsieh[1] Donghyuk Lee[1] Tianshi Li[1,3]
Gennady Pekhimenko[1] Samira Khan[4] Onur Mutlu[5,1]
[1]Carnegie Mellon University [2]TOBB ETÜ [3]Peking University [4]University of Virginia [5]ETH Zürich

Design-Induced Latency Variation in DRAM

- Donghyuk Lee, Samira Khan, Lavanya Subramanian, Saugata Ghose, Rachata Ausavarungnirun, Gennady Pekhimenko, Vivek Seshadri, and Onur Mutlu, **"Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms"** *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, Urbana-Champaign, IL, USA, June 2017.

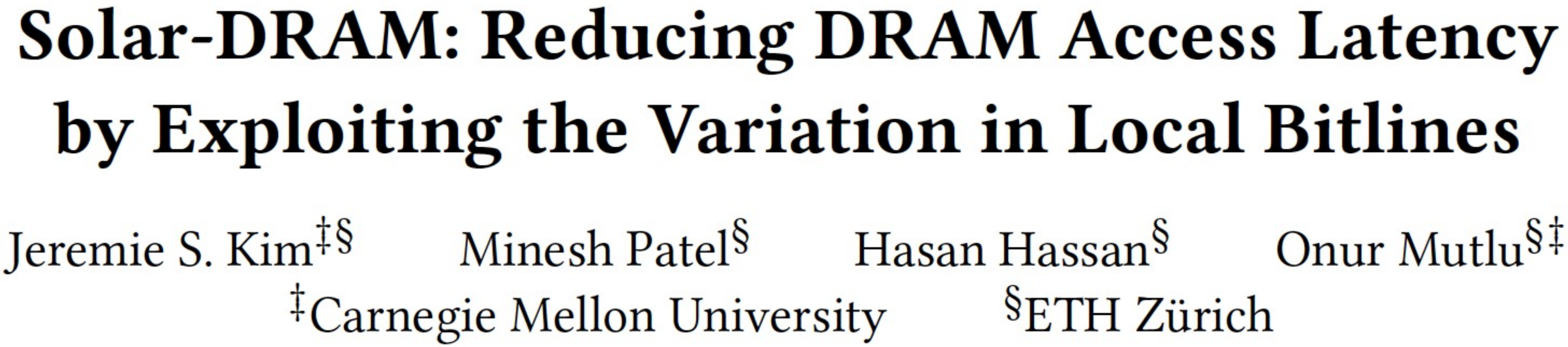


Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms

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"** *Proceedings of the 36th IEEE International Conference on Computer Design (**ICCD**)*, Orlando, FL, USA, October 2018.

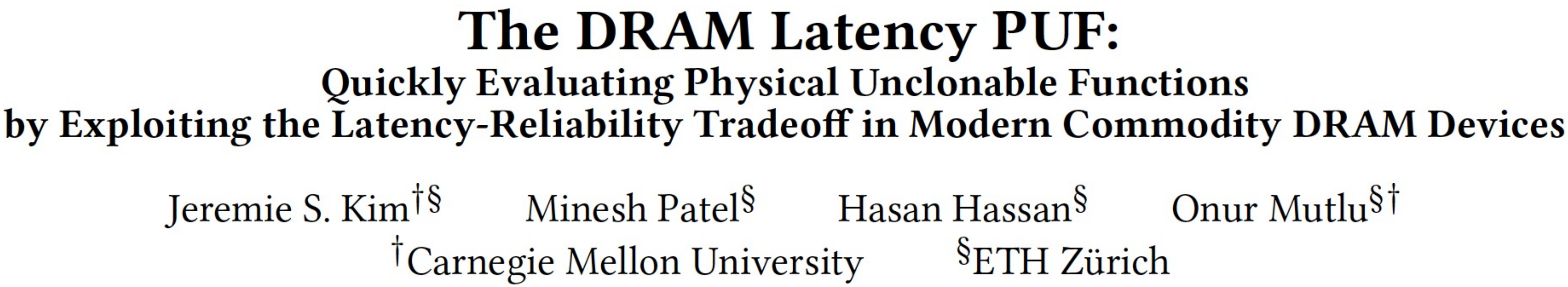


Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines

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"**
  Proceedings of the 24th International Symposium on High-Performance Computer Architecture (***HPCA***), Vienna, Austria, February 2018.
  [Lightning Talk Video]
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)]



The DRAM Latency PUF:

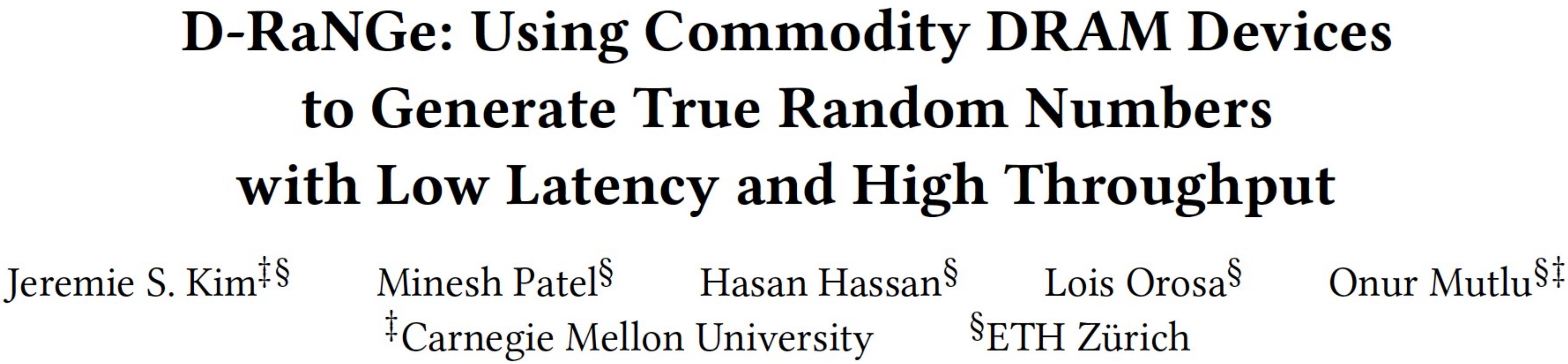
Quickly Evaluating Physical Unclonable Functions by Exploiting the Latency-Reliability Tradeoff in Modern Commodity DRAM Devices

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

[†]Carnegie Mellon University [§]ETH Zürich

DRAM Latency True Random Number Generator

- Jeremie S. Kim, Minesh Patel, Hasan Hassan, Lois Orosa, and Onur Mutlu, **"D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput"** *Proceedings of the 25th International Symposium on High-Performance Computer Architecture* (***HPCA***), Washington, DC, USA, February 2019.



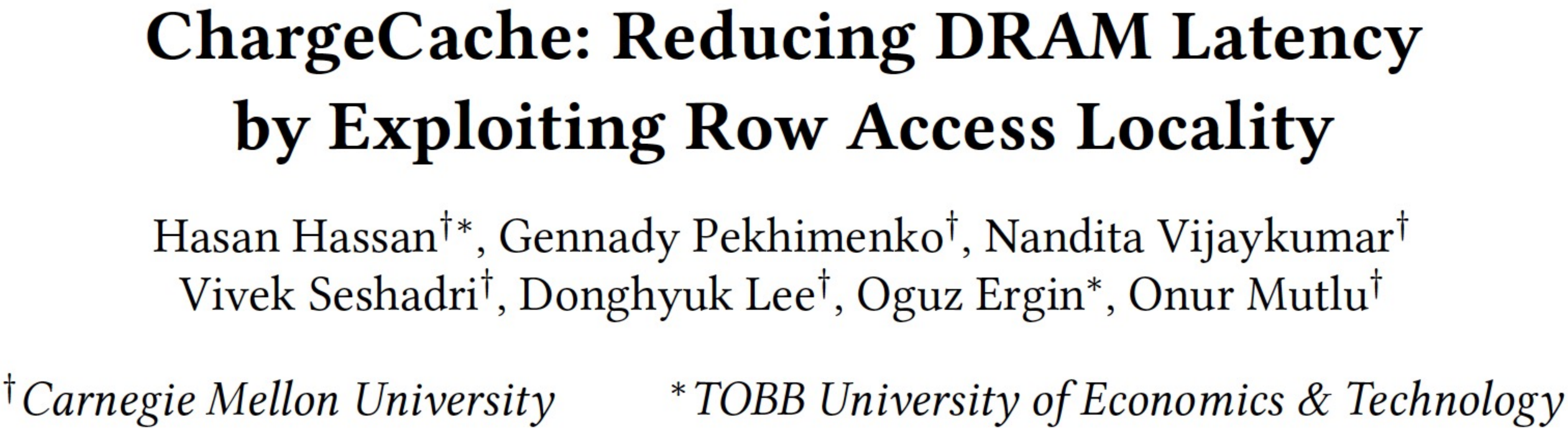
D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput

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"**
Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (***HPCA***), Barcelona, Spain, March 2016.
[Slides (pptx) (pdf)]
[Source Code]



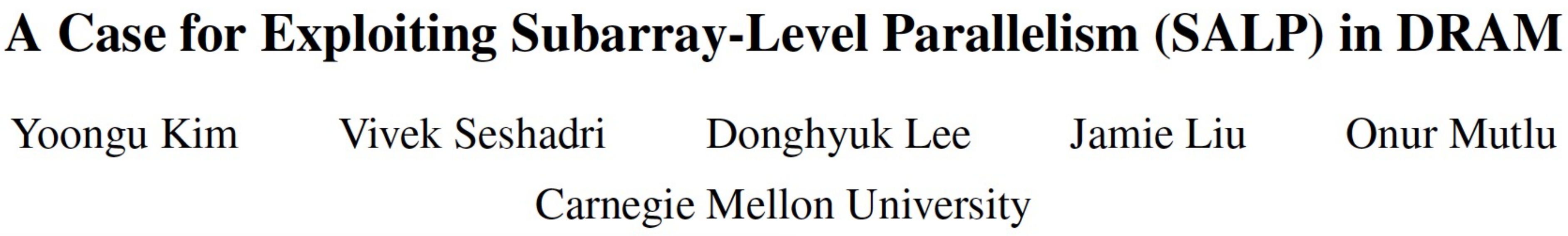
ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality

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)

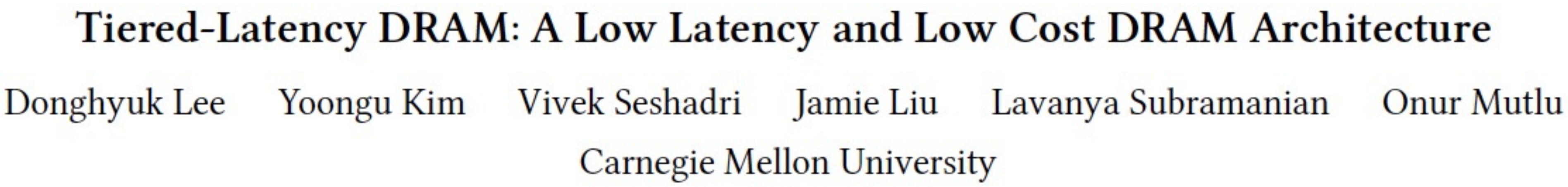


A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM

Yoongu Kim   Vivek Seshadri   Donghyuk Lee   Jamie Liu   Onur Mutlu
Carnegie Mellon University

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)



Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture

Donghyuk Lee Yoongu Kim Vivek Seshadri Jamie Liu Lavanya Subramanian Onur Mutlu
Carnegie Mellon University

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"
Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (***HPCA***), Barcelona, Spain, March 2016.
[Slides (pptx) (pdf)]
[Source Code]

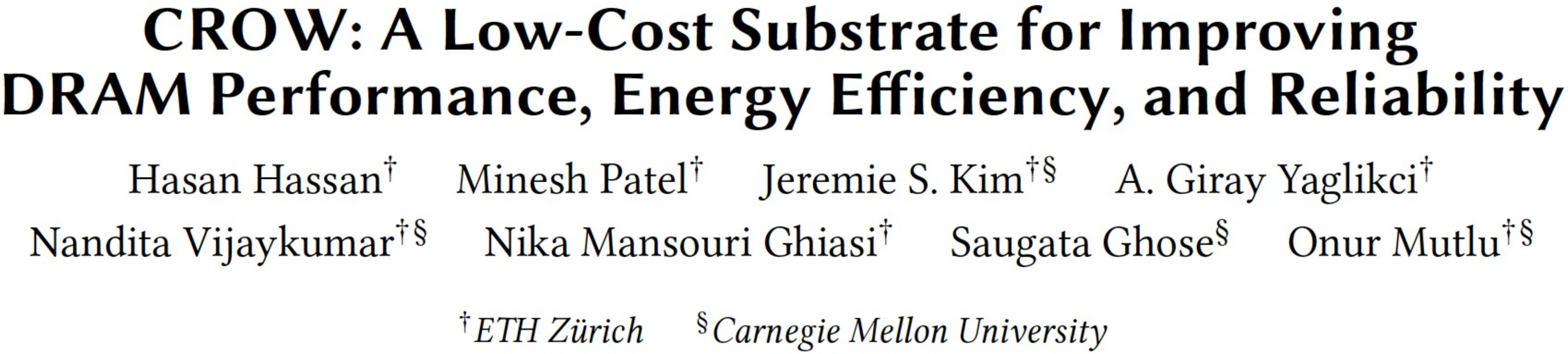
Low-Cost Inter-Linked Subarrays (LISA):
Enabling Fast Inter-Subarray Data Movement in DRAM

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

[†]*Carnegie Mellon University* [*]*Georgia Institute of Technology*

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"
Proceedings of the 46th International Symposium on Computer Architecture (***ISCA***), Phoenix, AZ, USA, June 2019.



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

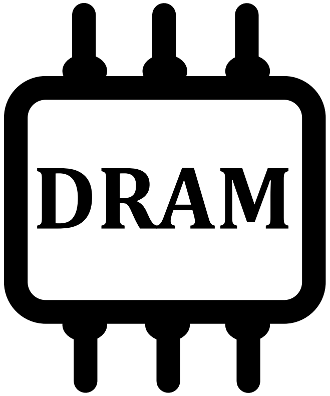
Hasan Hassan[†] Minesh Patel[†] Jeremie S. Kim[†§] A. Giray Yaglikci[†]
Nandita Vijaykumar[†§] Nika Mansouri Ghiasi[†] Saugata Ghose[§] Onur Mutlu[†§]

[†]*ETH Zürich* [§]*Carnegie Mellon University*

CROW: The Copy Row Substrate

[ISCA 2019]

Challenges of DRAM Scaling

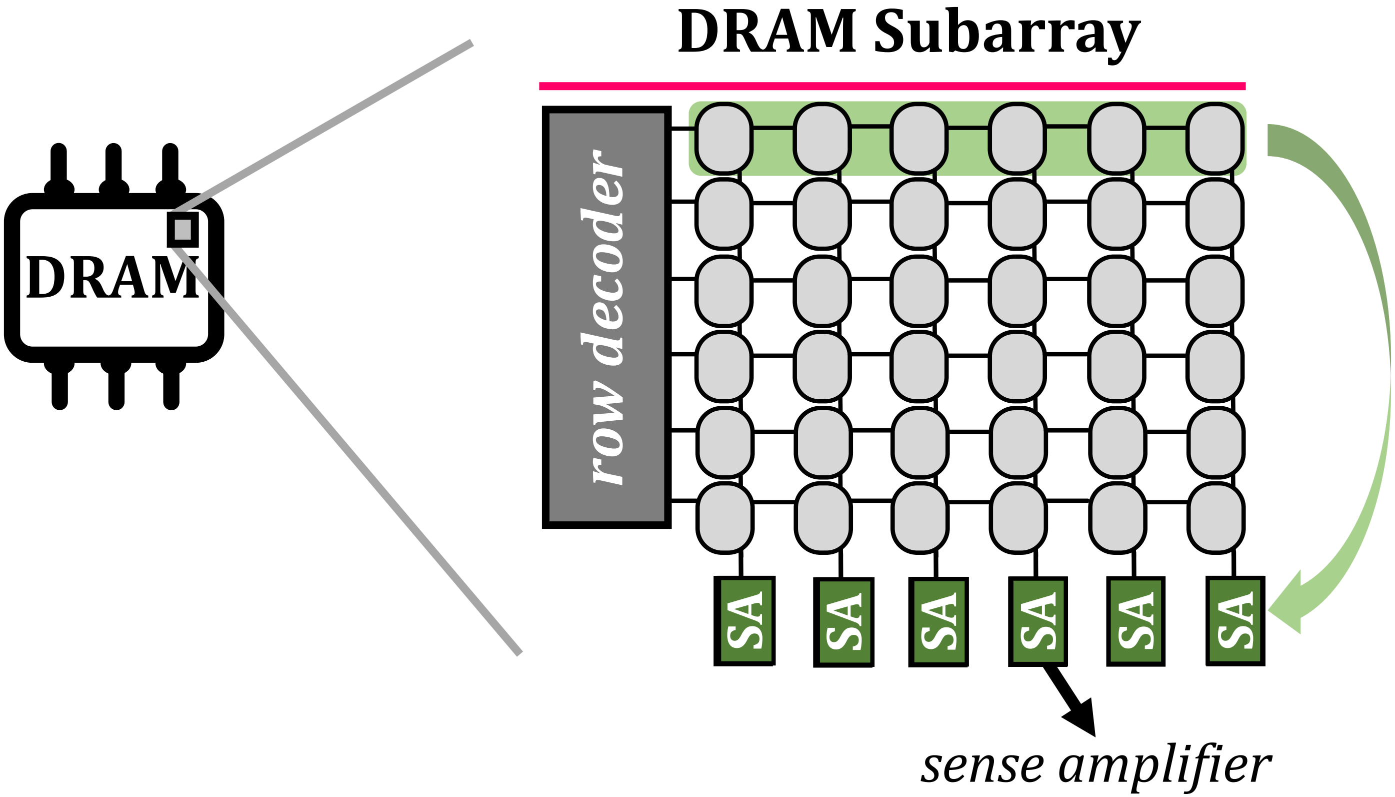


❶ access latency

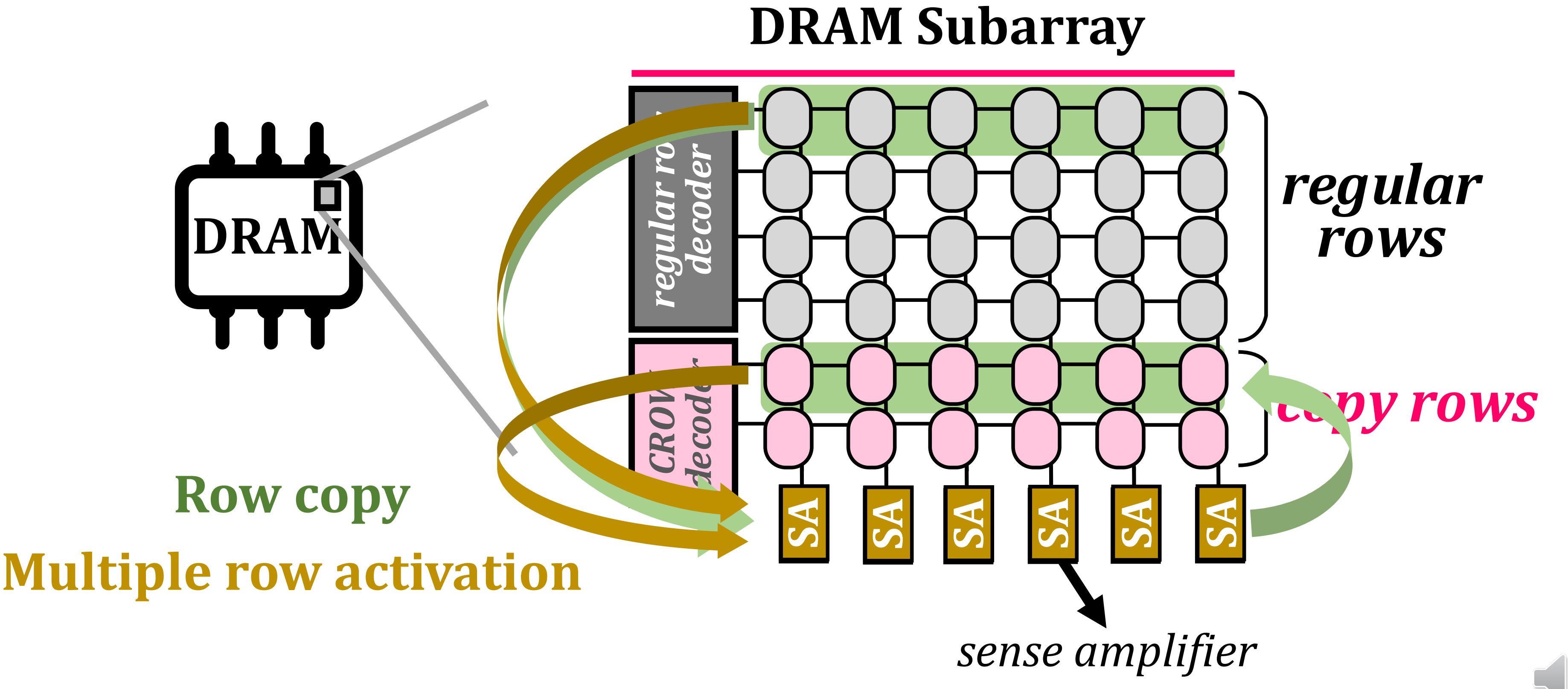
❷ refresh overhead

❸ exposure to vulnerabilities

Conventional DRAM



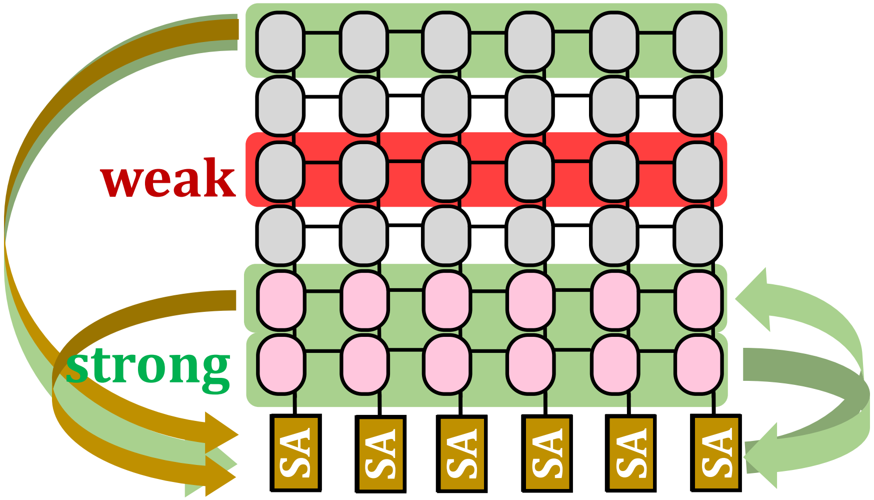
Copy Row DRAM (CROW)



Row copy

Multiple row activation

Use Cases of CROW



- **CROW-cache**
  - ✓reduces *access latency*
- **CROW-ref**
  - ✓reduces DRAM *refresh overhead*
- A mechanism for protecting against *RowHammer*

Key Results

CROW-cache + CROW-ref

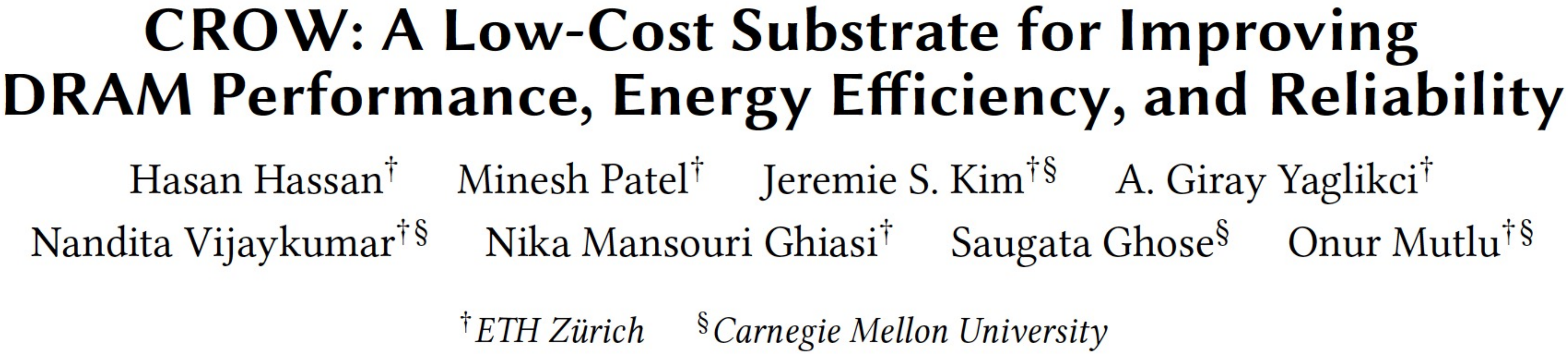
- 20% speedup
- 22% less DRAM energy

Hardware Overhead

- 0.5% DRAM chip area
- 1.6% DRAM capacity
- 11.3 KiB memory controller storage

More on CROW

- 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"
Proceedings of the 46th International Symposium on Computer Architecture (***ISCA***), Phoenix, AZ, USA, June 2019.
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Poster (pptx) (pdf)]
[Lightning Talk Video (3 minutes)]
[Full Talk Video (16 minutes)]
[Source Code for CROW (Ramulator and Circuit Modeling)]



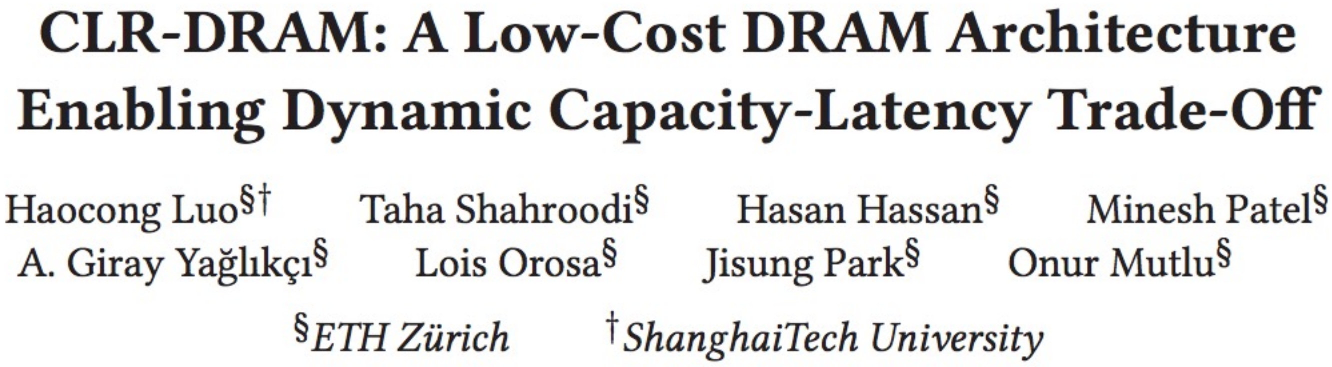
CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability

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

[†]*ETH Zürich* [§]*Carnegie Mellon University*

CLR-DRAM: Capacity-Latency Reconfigurability

- Haocong Luo, Taha Shahroodi, Hasan Hassan, Minesh Patel, A. Giray Yaglikci, Lois Orosa, Jisung Park, and Onur Mutlu, **"CLR-DRAM: A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-Off"**
*Proceedings of the 47th International Symposium on Computer Architecture (**ISCA**)*, Valencia, Spain, June 2020.
[Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Talk Video (20 minutes)]
[Lightning Talk Video (3 minutes)]



CLR-DRAM: A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-Off

Haocong Luo[§†] Taha Shahroodi[§] Hasan Hassan[§] Minesh Patel[§]
A. Giray Yağlıkçı[§] Lois Orosa[§] Jisung Park[§] Onur Mutlu[§]

[§]*ETH Zürich* [†]*ShanghaiTech University*

CLR-DRAM: Capacity-Latency Reconfigurable DRAM **[ISCA 2020]**

CLR-DRAM:
A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-off

Haocong Luo   Taha Shahroodi   Hasan Hassan   Minesh Patel
A. Giray Yaglıkçı   Lois Orosa   Jisung Park Onur Mutlu

ETH Zürich

SAFARI
SAFARI Research Group
safari.ethz.ch

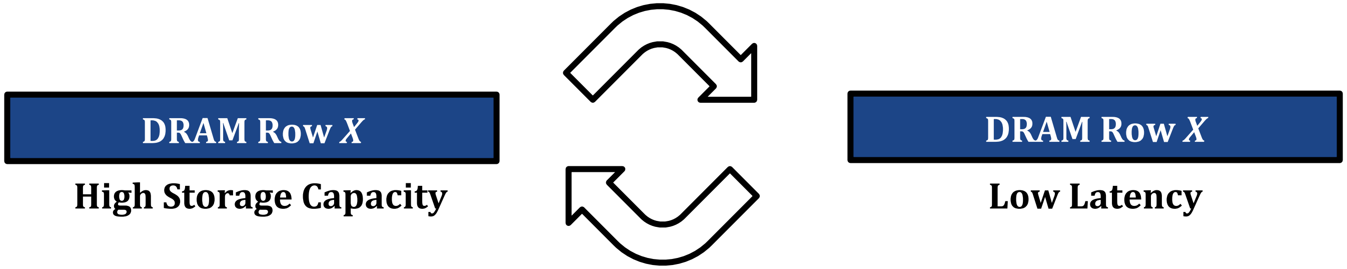
上海科技大学
ShanghaiTech University

Motivation & Goal

- Workloads and systems have varying main memory capacity and latency demands.
- Existing commodity DRAM makes static capacity-latency trade-off at design time.

- Systems miss opportunities to improve performance by adapting to changes in main memory capacity and latency demands.

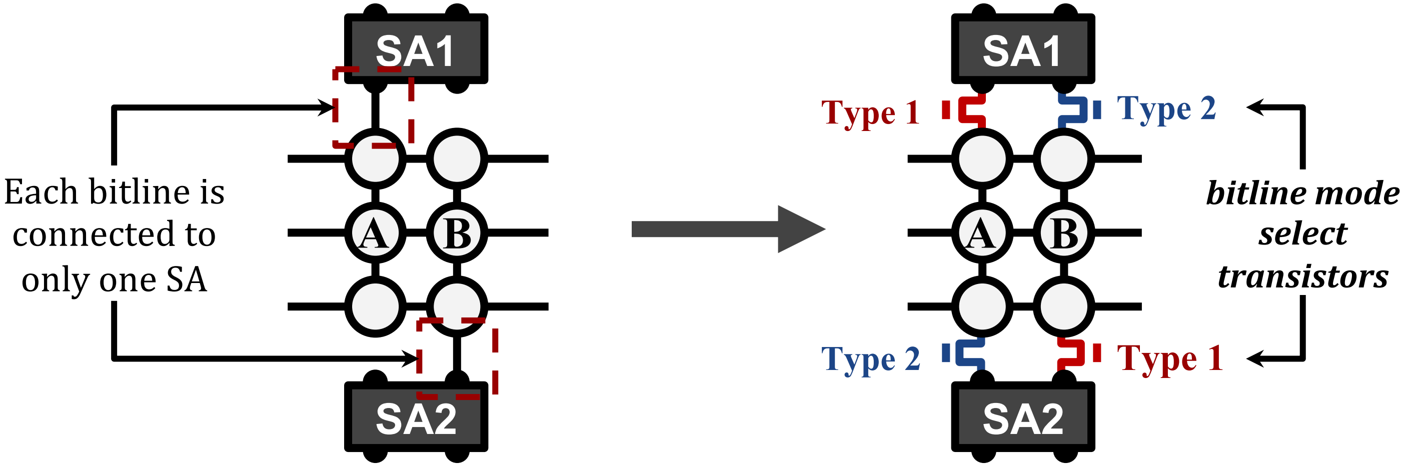
- **Goal**: Design a low-cost DRAM architecture that can be dynamically configured to have high capacity or low latency at a fine granularity (i.e., at the granularity of a row).



CLR-DRAM (Capacity-Latency-Reconfigurable DRAM)

- **CLR-DRAM (Capacity-Latency-Reconfigurable DRAM)**:
  - A low cost DRAM architecture that enables a single DRAM row to *dynamically* switch between **max-capacity mode** or **high-performance mode**.
- **Key Idea**:

  Dynamically configure the connections between DRAM cells and sense amplifiers in the density-optimized open-bitline architecture.



Each bitline is connected to only one SA

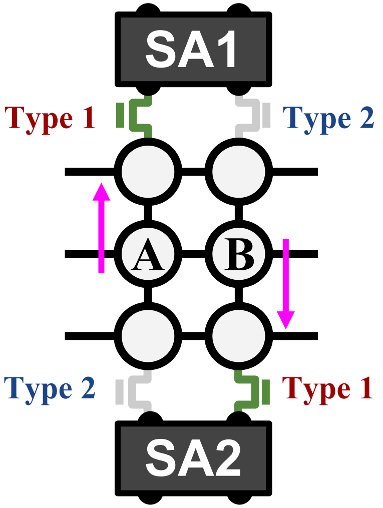
bitline mode select transistors

Open-bitline (Baseline)

CLR-DRAM

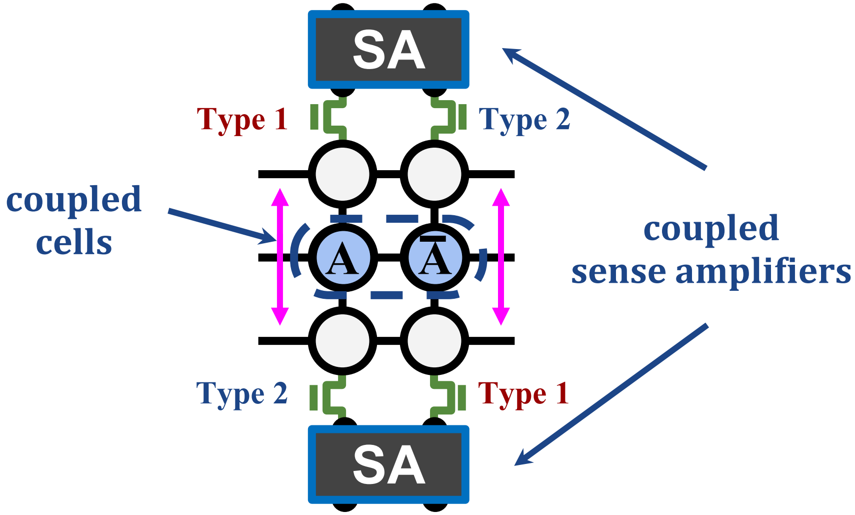
CLR-DRAM (Capacity-Latency-Reconfigurable DRAM)

- Max-capacity mode



mimics the cell-to-SA connections as in the open-bitline architecture

- High-performance mode

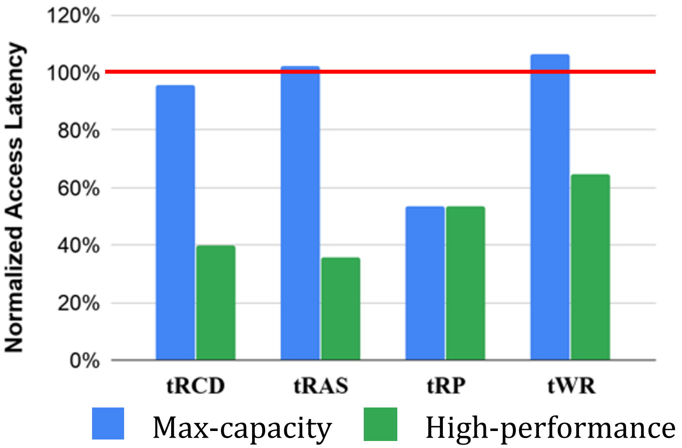


The same storage capacity as the conventional open-bitline architecture

Reduced latency and refresh overhead via coupled cell/SA operation

Key Results

- **DRAM Latency Reduction**:
  - Activation latency (**tRCD**) by **60.1%**
  - Restoration latency (**tRAS**) by **64.2%**
  - Precharge latency (**tRP**) by **46.4%**
  - Write-recovery latency (**tWR**) by **35.2%**

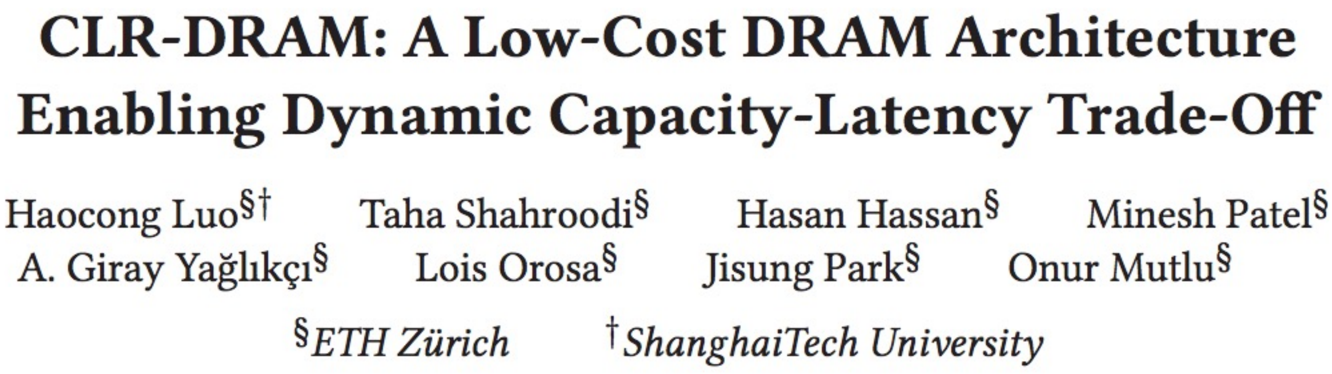


- **System-level Benefits**:
  - Performance improvement: **18.6%**
  - DRAM energy reduction: **29.7%**
  - DRAM refresh energy reduction: **66.1%**

We hope that CLR-DRAM can be exploited to develop more flexible systems that can adapt to the diverse and changing DRAM capacity and latency demands of workloads.

More on CLR-DRAM

- Haocong Luo, Taha Shahroodi, Hasan Hassan, Minesh Patel, A. Giray Yaglikci, Lois Orosa, Jisung Park, and Onur Mutlu, **"CLR-DRAM: A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-Off"** *Proceedings of the 47th International Symposium on Computer Architecture (**ISCA**)*, Valencia, Spain, June 2020. [Slides (pptx) (pdf)] [Lightning Talk Slides (pptx) (pdf)] [Talk Video (20 minutes)] [Lightning Talk Video (3 minutes)]



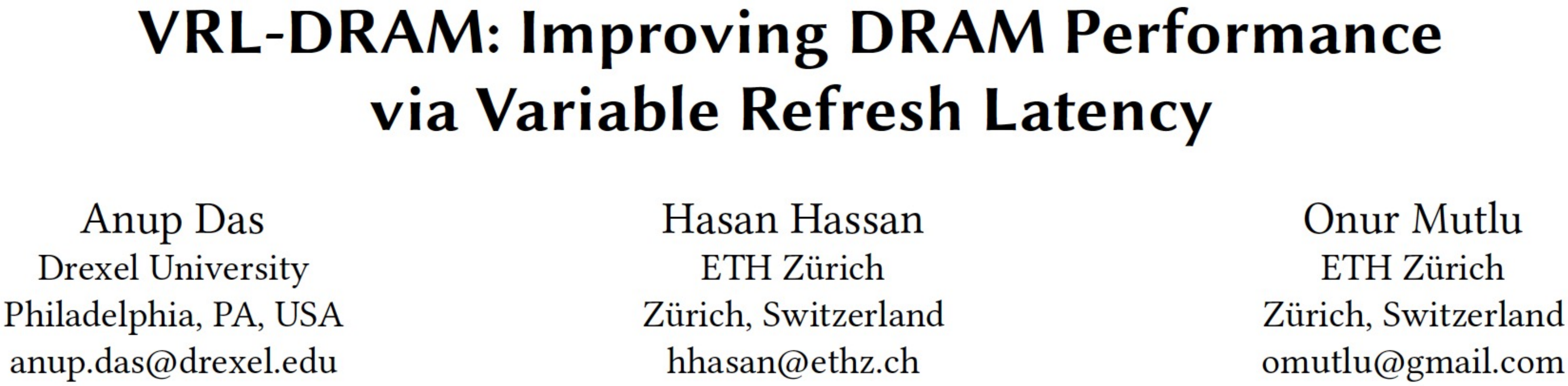
CLR-DRAM: A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-Off

Haocong Luo[§†] Taha Shahroodi[§] Hasan Hassan[§] Minesh Patel[§]
A. Giray Yağlıkçı[§] Lois Orosa[§] Jisung Park[§] Onur Mutlu[§]

[§]*ETH Zürich* [†]*ShanghaiTech 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

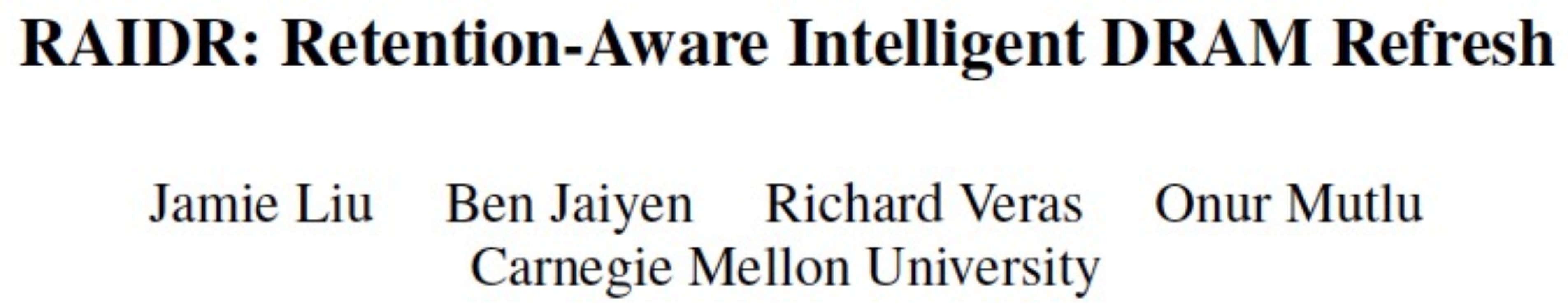
- Kevin Chang, Donghyuk Lee, Zeshan Chishti, Alaa Alameldeen, Chris Wilkerson, Yoongu Kim, and Onur Mutlu,
"Improving DRAM Performance by Parallelizing Refreshes with Accesses"
Proceedings of the 20th International Symposium on High-Performance Computer Architecture (***HPCA***), Orlando, FL, February 2014.
[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 DRAM Refresh"** *Proceedings of the 39th International Symposium on Computer Architecture* (***ISCA***), Portland, OR, June 2012. Slides (pdf)

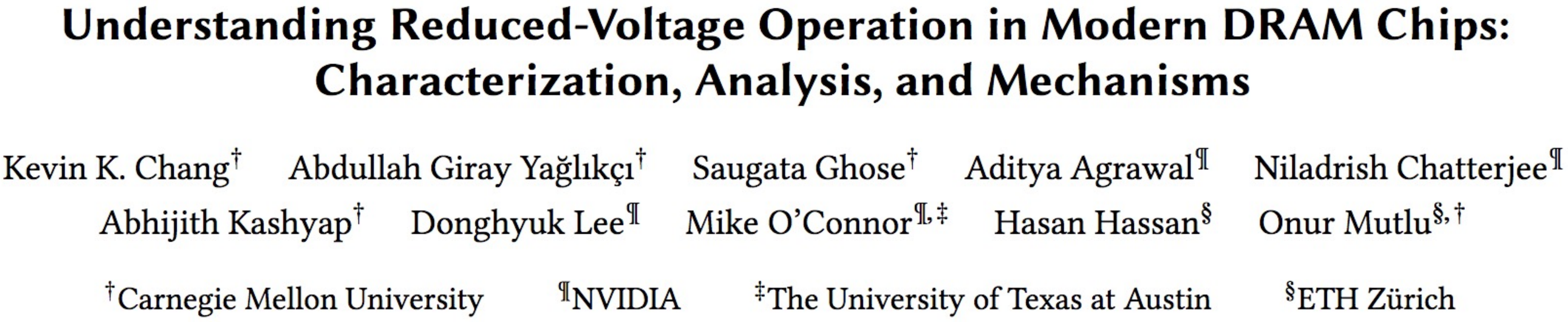


RAIDR: Retention-Aware Intelligent DRAM Refresh

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"
*Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, Urbana-Champaign, IL, USA, June 2017.



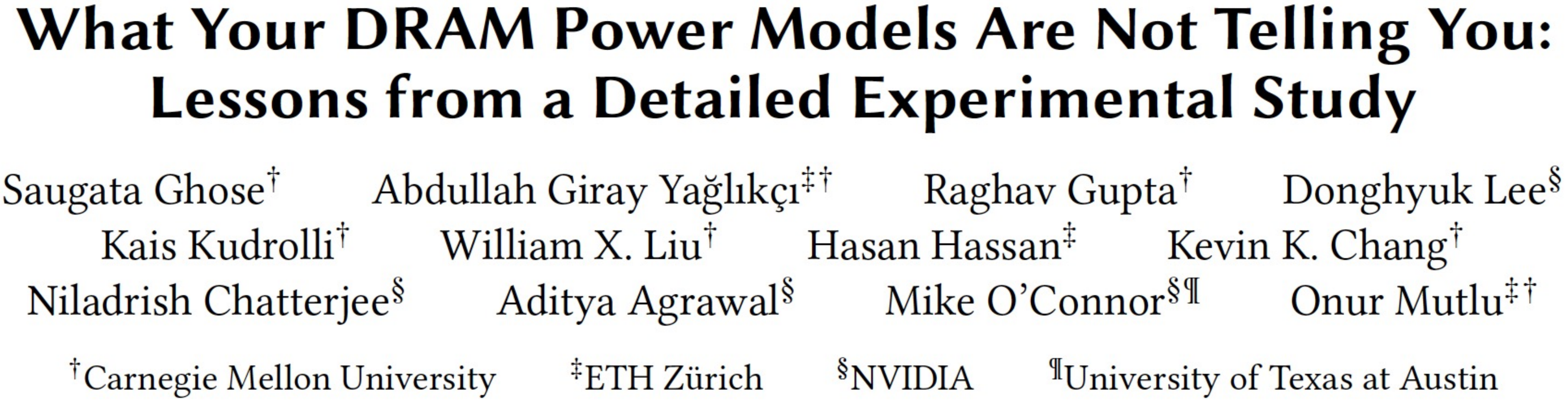
Understanding Reduced-Voltage Operation in Modern DRAM Chips: Characterization, Analysis, and Mechanisms

Kevin K. Chang[†] Abdullah Giray Yağlıkçı[†] 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"
  Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (***SIGMETRICS***), Irvine, CA, USA, June 2018.
  [Abstract]
  [POMACS Journal Version (same content, different format)]
  [Slides (pptx) (pdf)]
  [VAMPIRE DRAM Power Model]



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[§¶] Onur Mutlu[‡†]

[†]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