Intelligent Architectures for Intelligent Machines

Onur Mutlu

omutlu@gmail.com

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

1 July 2019

Yale @ 80

Design Principles

- ***Critical path design***
- ***Bread and Butter design***
- ***Balanced design***

from Yale Patt’s EE 382N lecture notes

(Micro)architecture Design Principles

- Bread and butter design
  - Spend time and resources on where it matters (i.e. improving what the machine is designed to do)
  - Common case vs. uncommon case

- Balanced design
  - Balance instruction/data flow through uarch components
  - Design to eliminate bottlenecks

- Critical path design
  - Find the maximum speed path and decrease it
    - Break a path into multiple cycles?

from my ECE 740 lecture notes

This Talk

- Design Principles
- How We Violate Those Principles Today
- Principled Intelligent Architectures
- Concluding Remarks

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

VP9

YouTube

Video Playback

Google's **video codec**

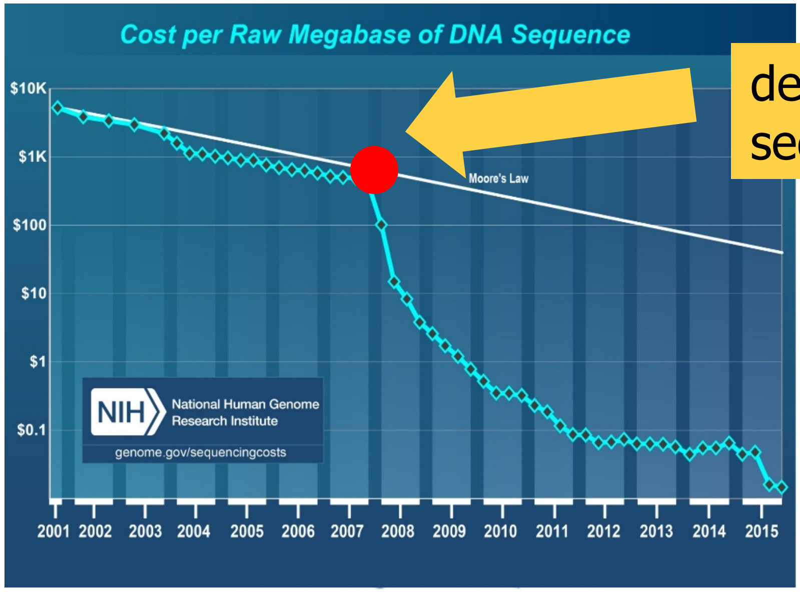
VP9

YouTube

Video Capture

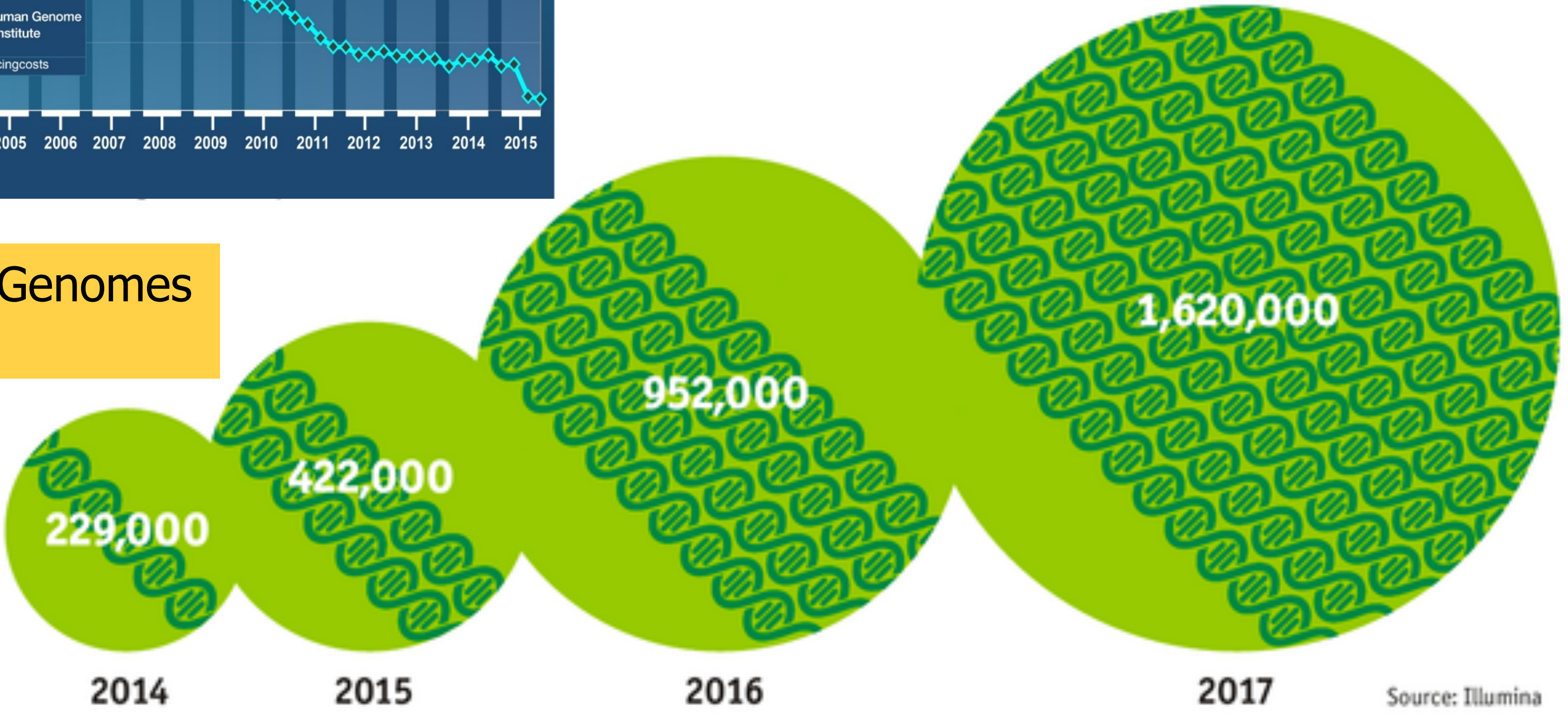
Google's **video codec**

Data is Key for Future Workloads



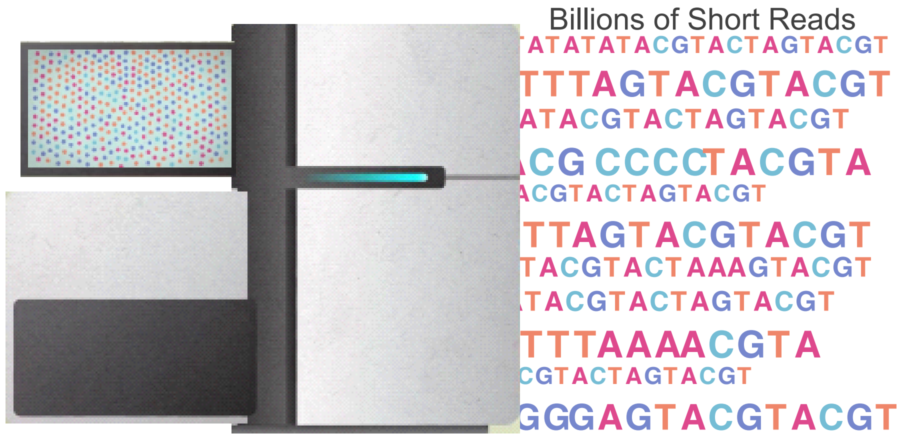
development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

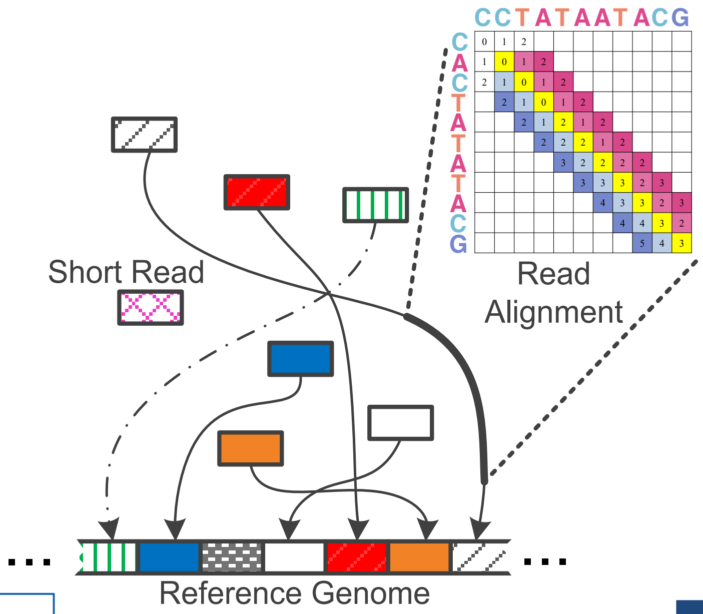


The Economist

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



1 **Sequencing**

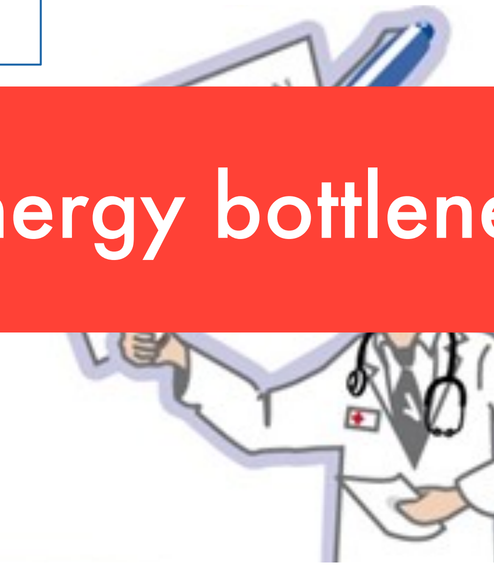


Read Mapping 2

Genome Analysis

Data → performance & energy bottleneck

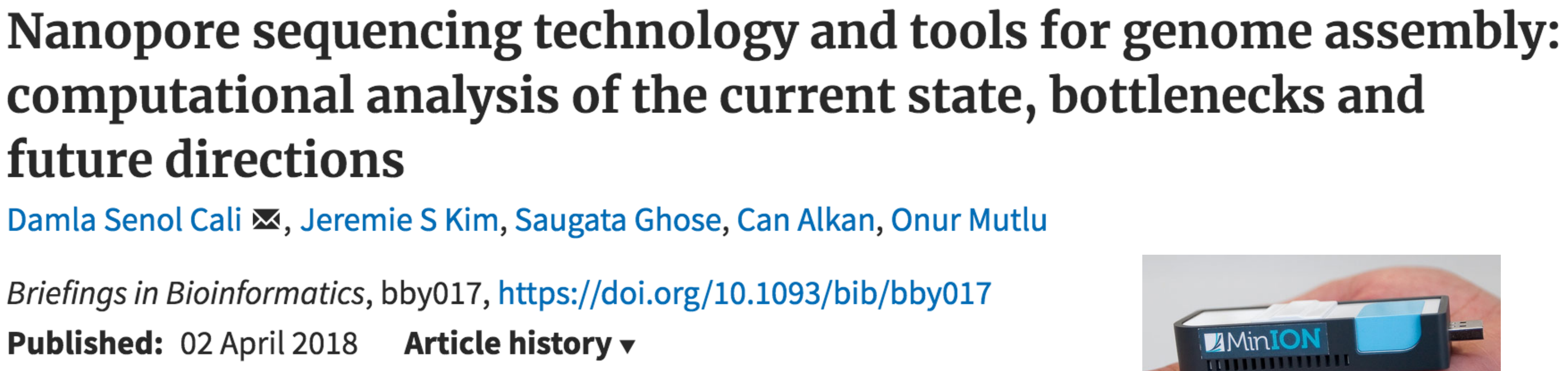
```
read4:          CGCTTCCAT
read5:               CCATGACGC
read6:             TTCCATGAC
```



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

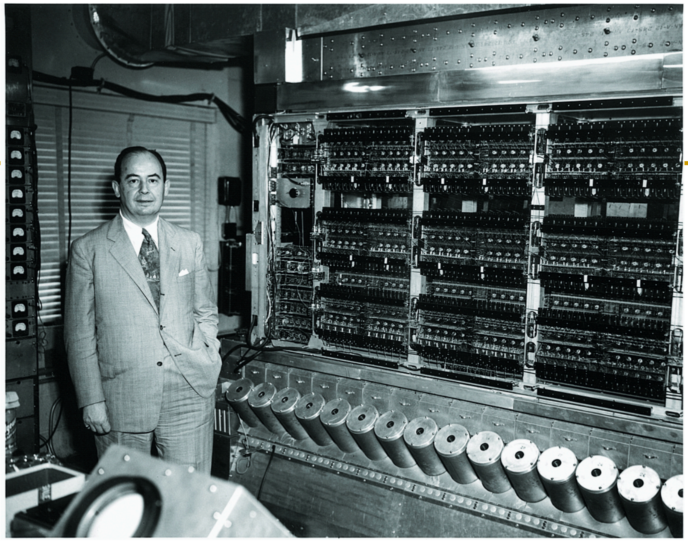
Data → performance & energy bottleneck

Data Overwhelms Modern Machines …

- Storage/memory capability
- Communication capability
- Computation capability
- Greatly impacts robustness, energy, performance, cost

A Computing System

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



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

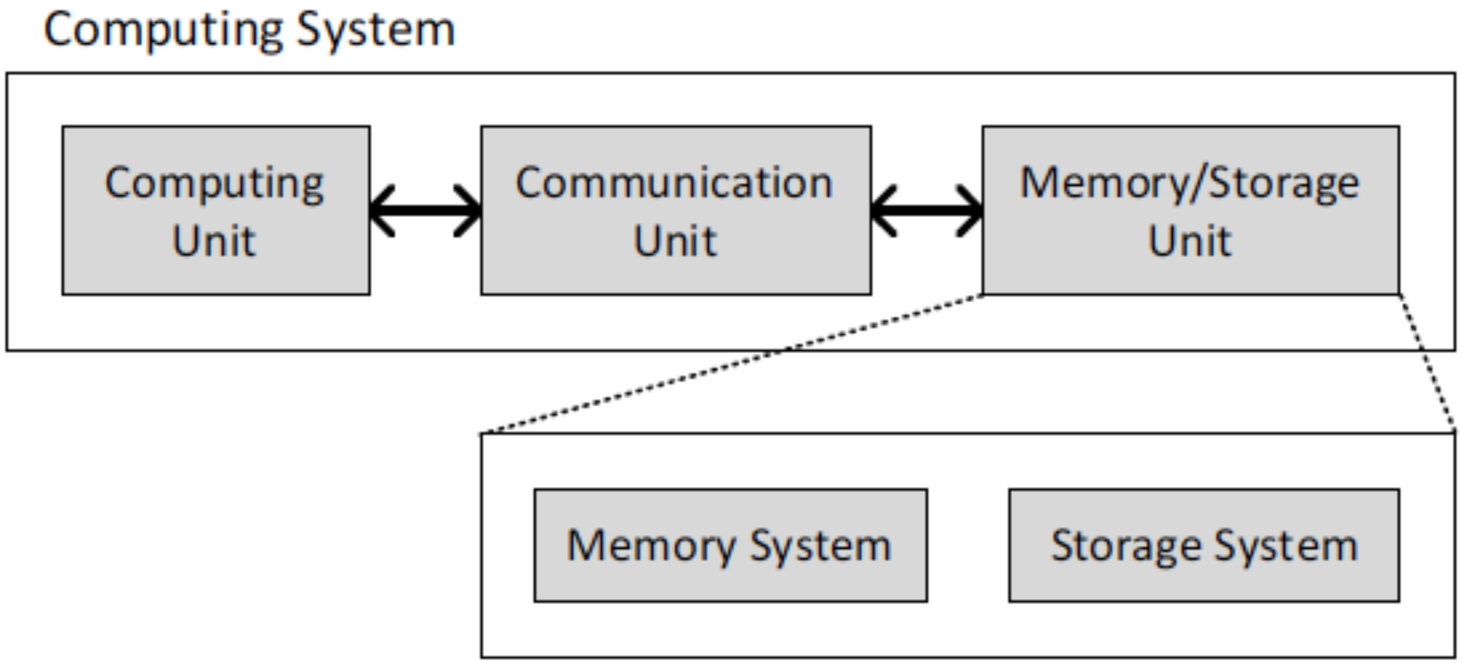
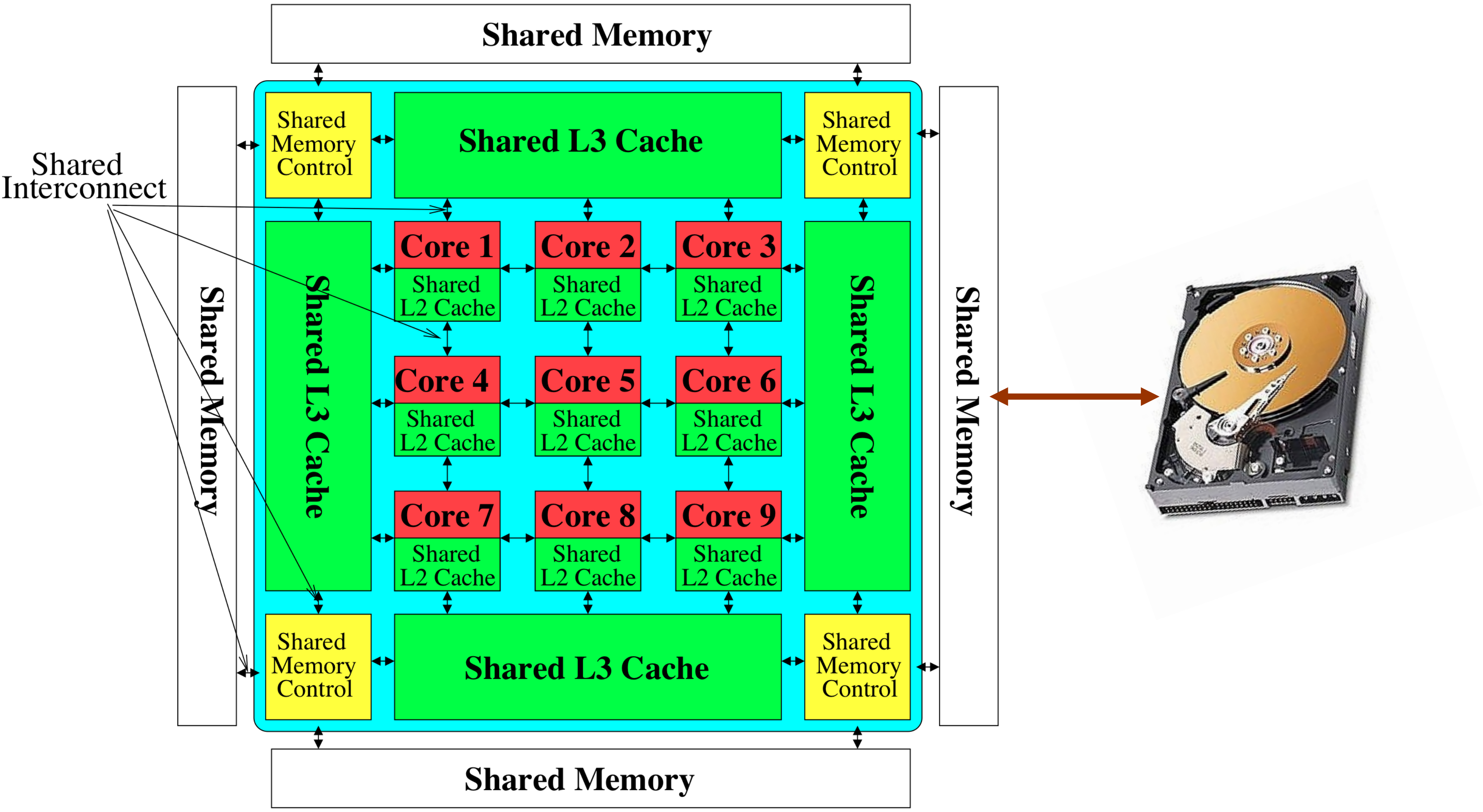


Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

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

VP9

YouTube

Video Playback

Google's video codec

VP9

YouTube

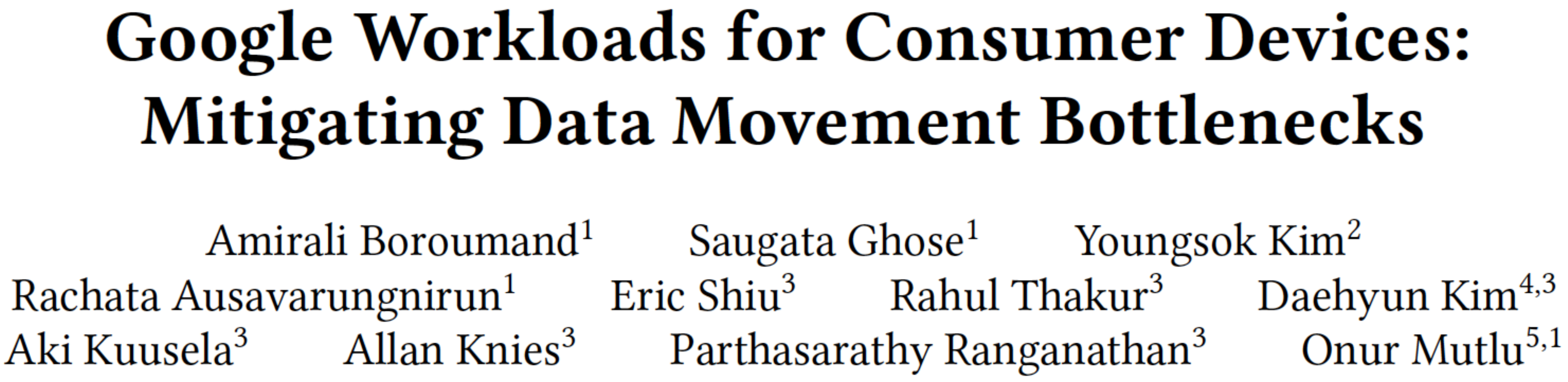
Video Capture

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]

SAFARI

Axiom

An Intelligent Architecture Handles Data Well

How to Handle Data Well

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

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

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

Corollaries: Architectures Today …

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

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

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

Data-Centric (Memory-Centric) Architectures

Data-Centric Architectures: Properties

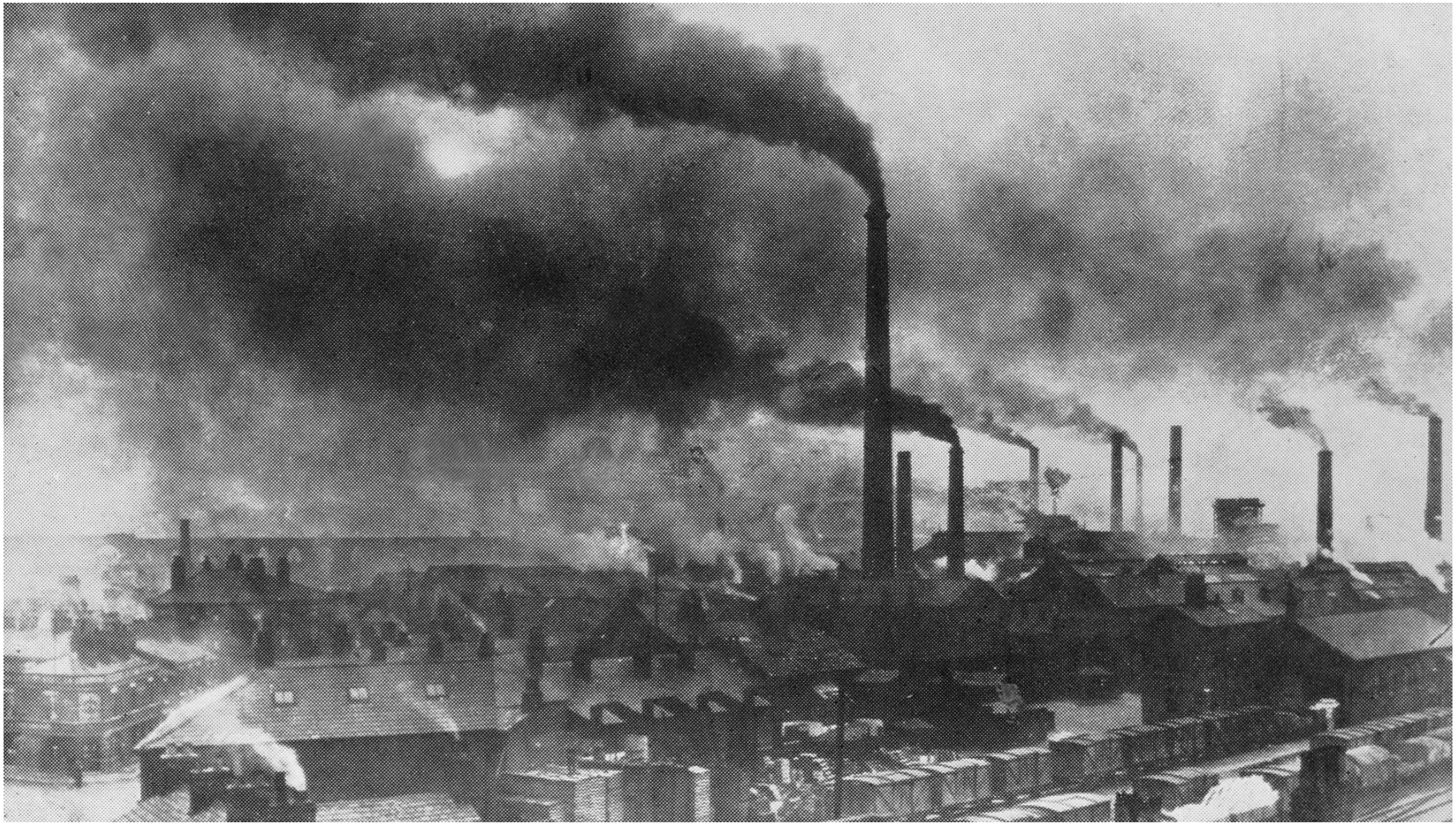
- **Process data where it resides** (where it makes sense)
  - Processing in and near memory structures
- **Low-latency and low-energy data access**
  - Low latency memory
  - Low energy memory
- **Low-cost data storage and processing**
  - High capacity memory at low cost: hybrid memory, compression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

Processing Data Where It Makes Sense

Do We Want This?

Source: V. Milutinovic

Or This?



SAFARI

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

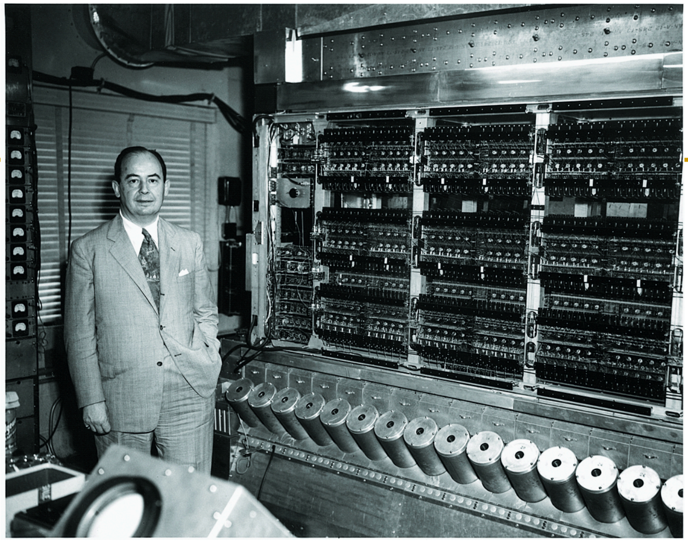
Processing of data

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.

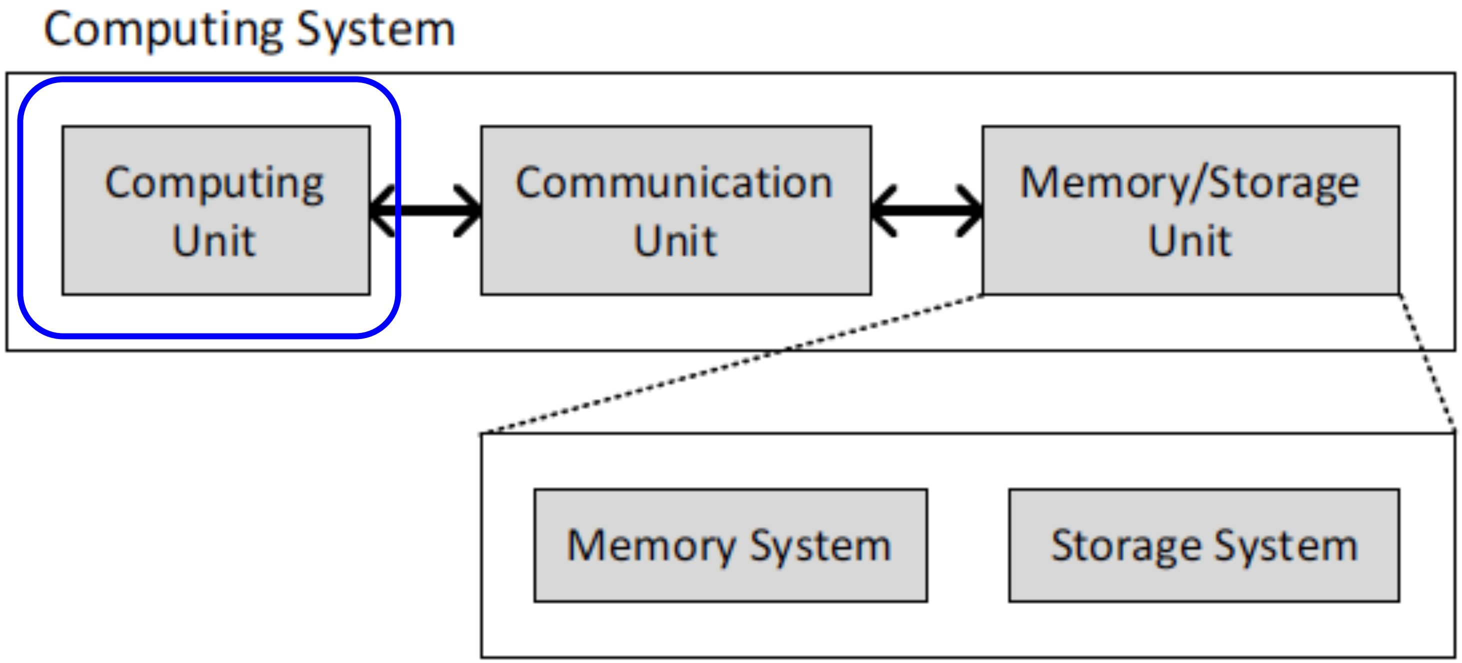
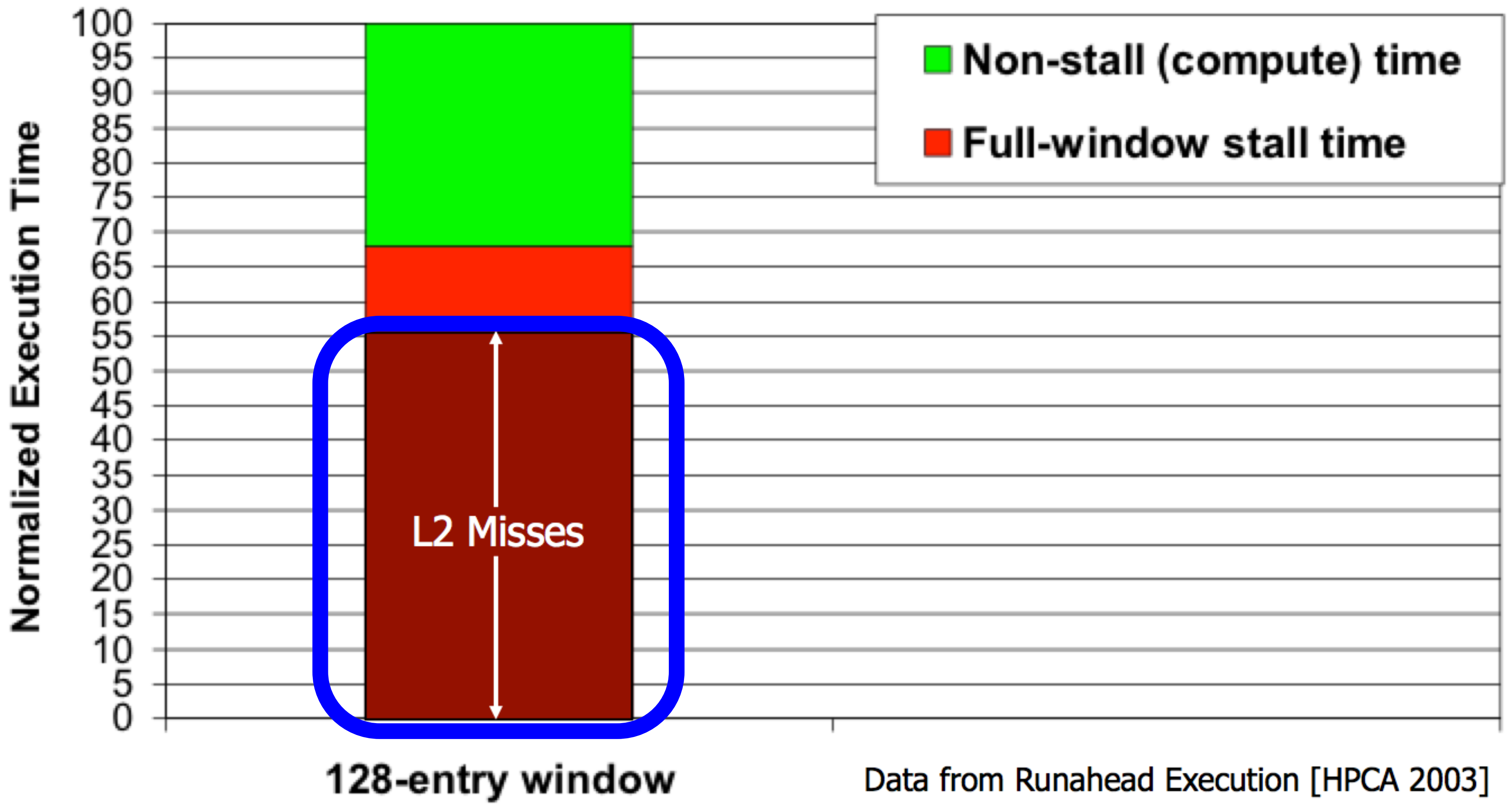


Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

Yet …

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

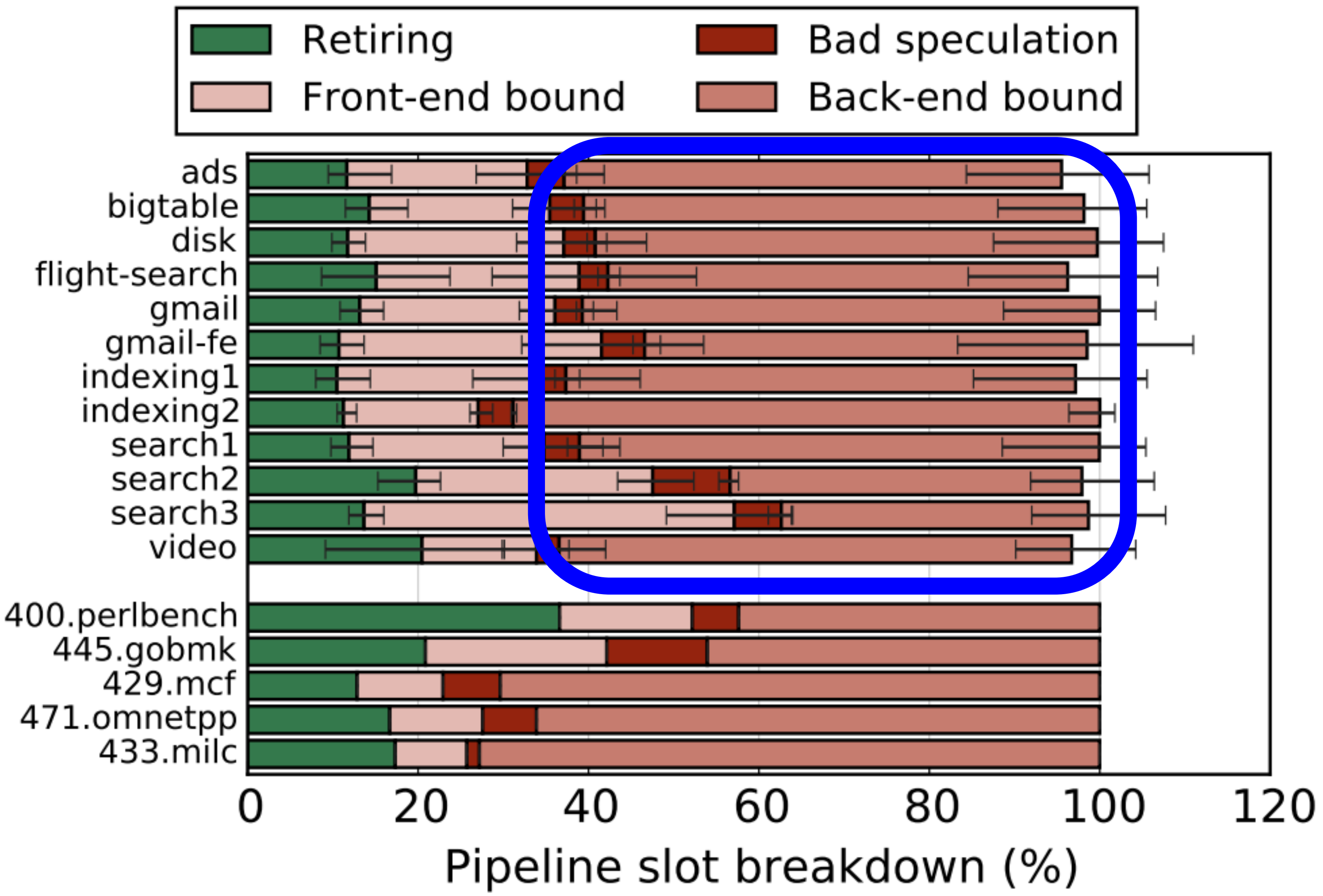


Data from Runahead Execution [HPCA 2003]

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

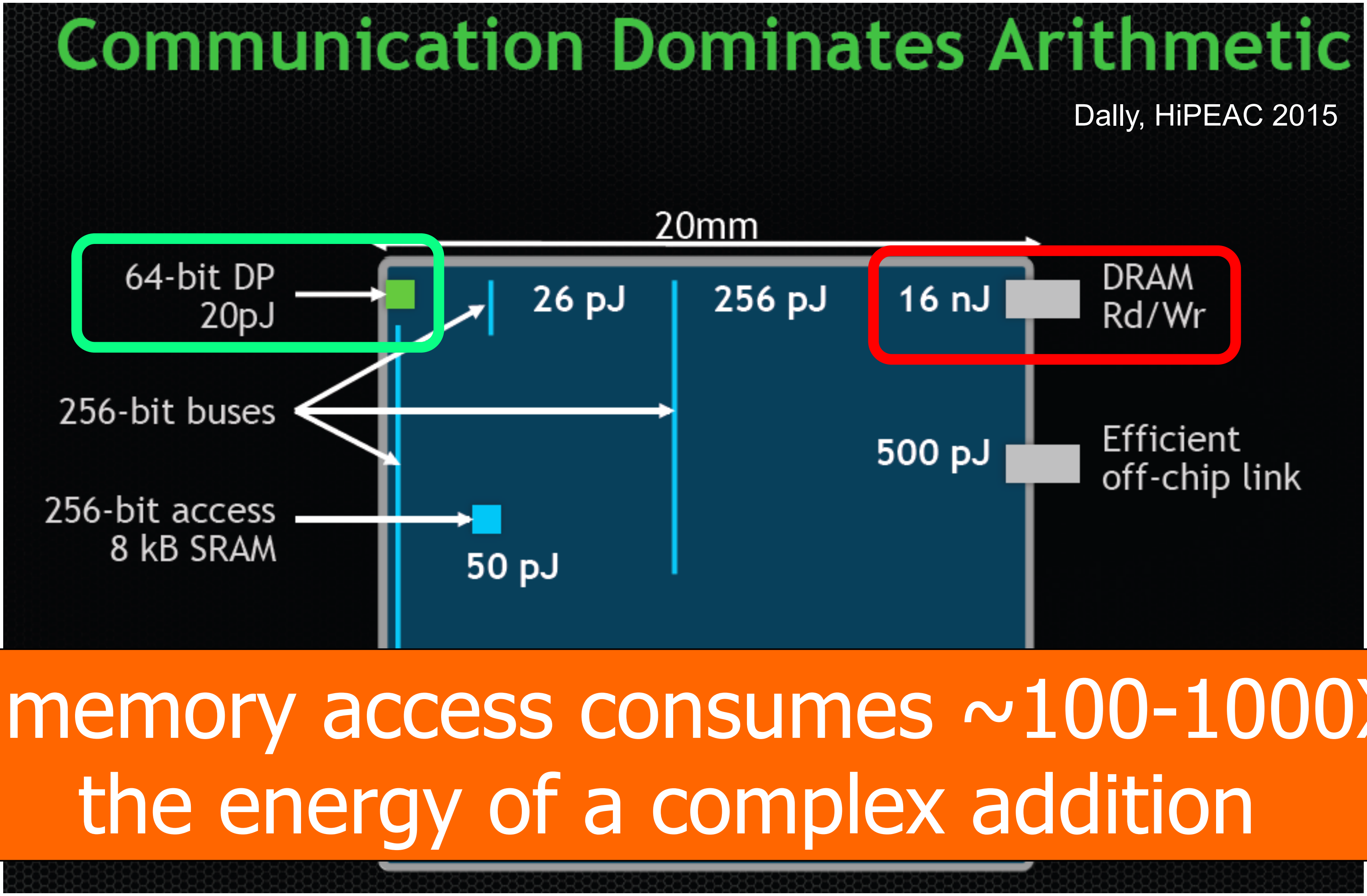
The Performance Perspective (Today)

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



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

Data Movement vs. Computation Energy

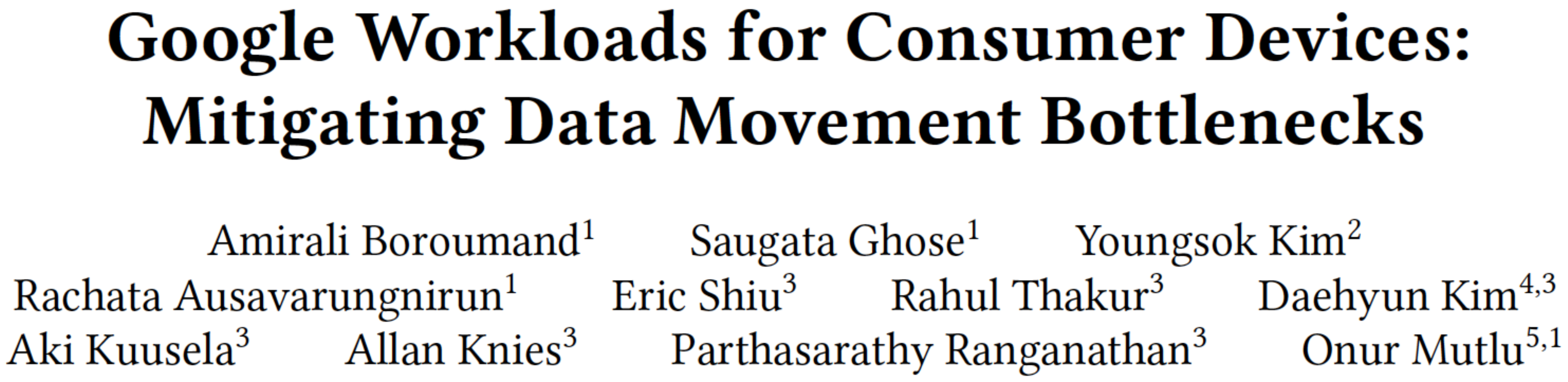


A memory access consumes ~100-1000X the energy of a complex addition

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.

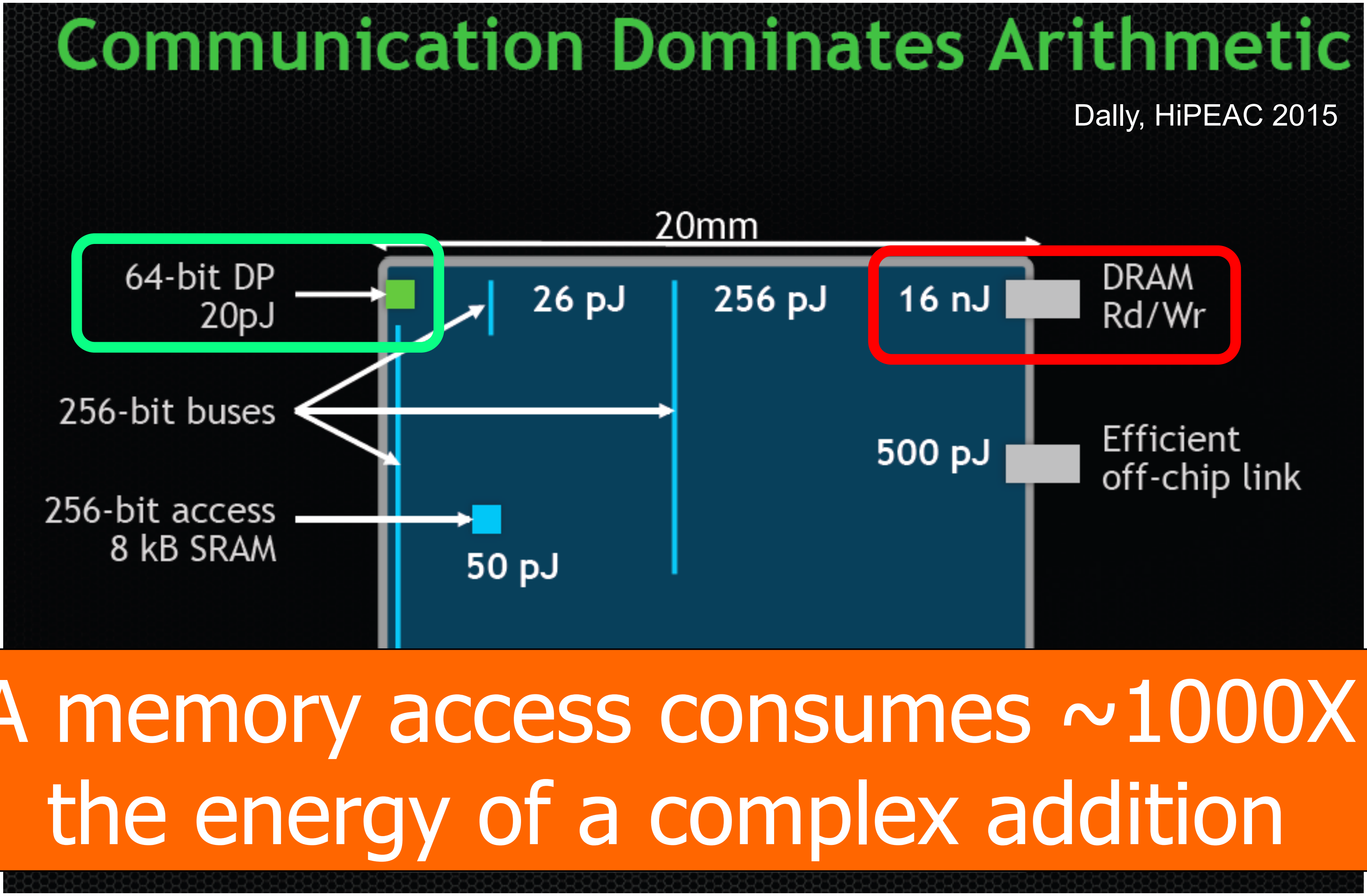
62.7% of the total system energy is spent on data movement



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!

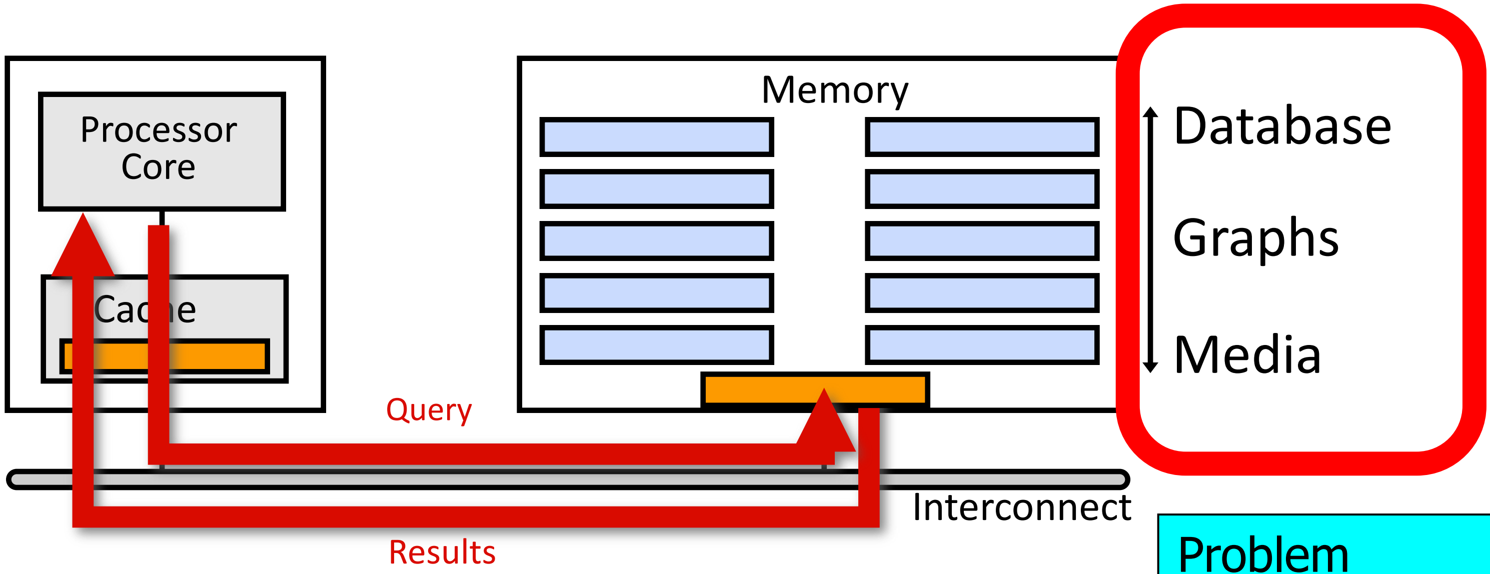


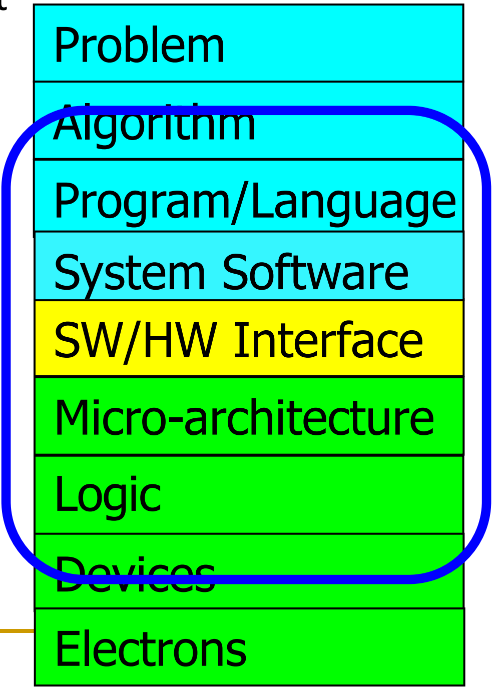
A memory access consumes ~1000X the energy of a complex addition

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





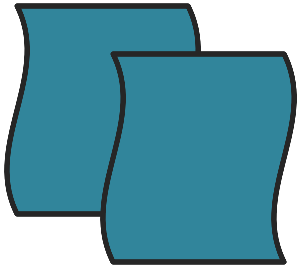
- Many questions … How do we design the:
  - compute-capable memory & controllers?
  - processor chip and in-memory units?
  - software and hardware interfaces?
  - system software and languages?
  - algorithms?

Processing in Memory: Two Approaches

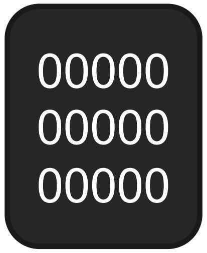
1. Minimally changing memory chips
2. Exploiting 3D-stacked memory

Starting Simple: Data Copy and Initialization

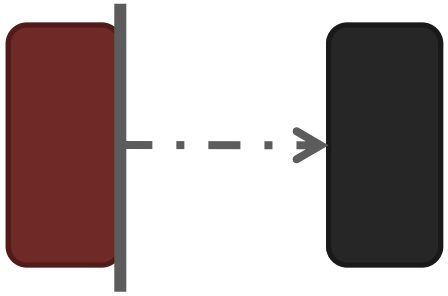
memmove & *memcpy*: 5% cycles in Google's datacenter [Kanev+ ISCA'15]



Forking

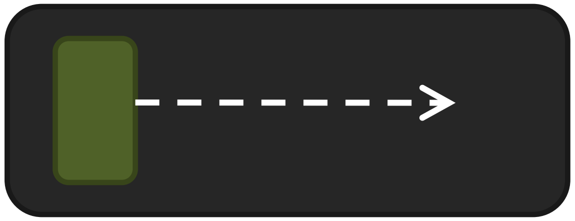


Zero initialization (e.g., security)



Checkpointing

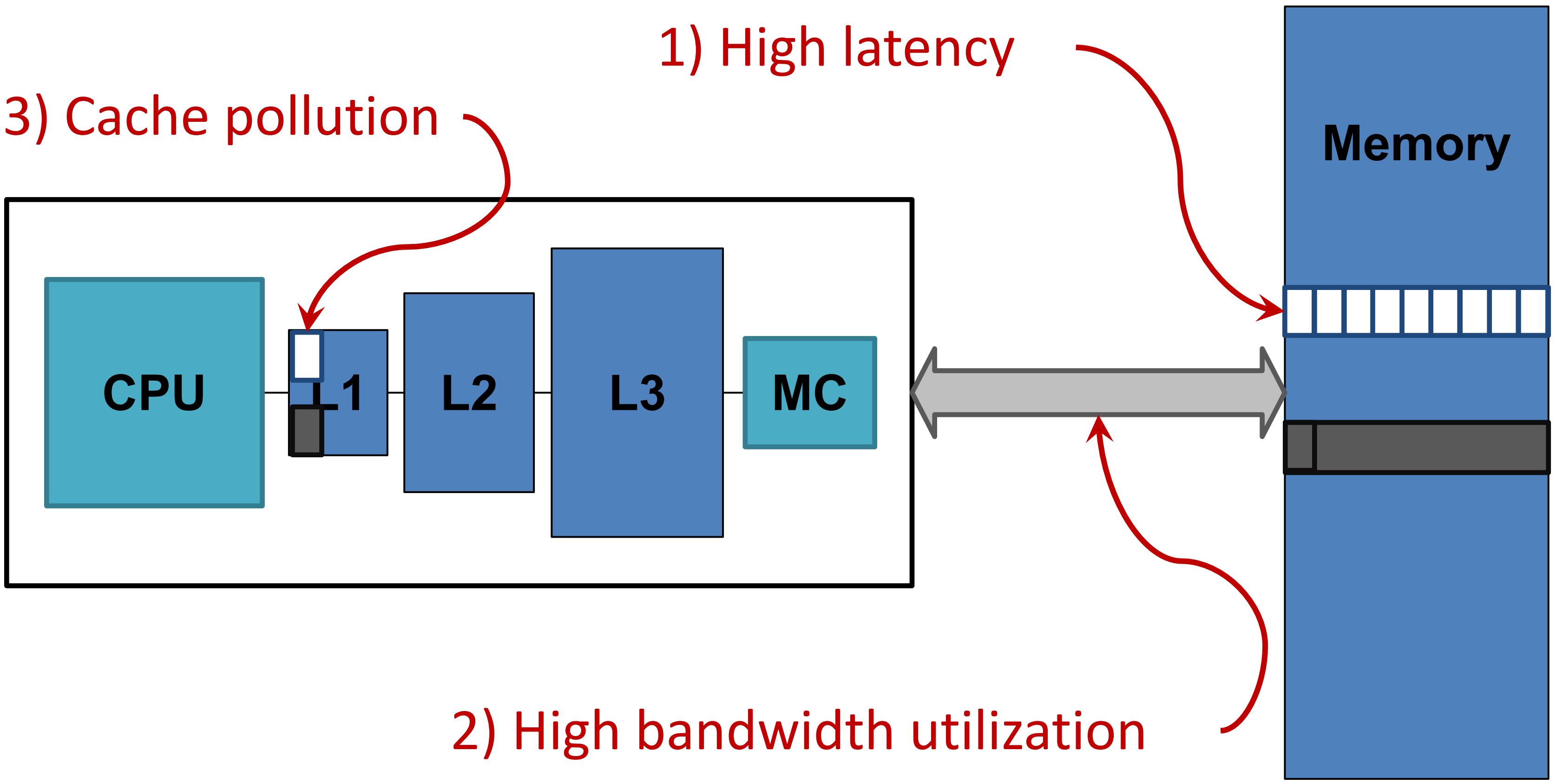
VM Cloning Deduplication



Page Migration

• • •
Many more

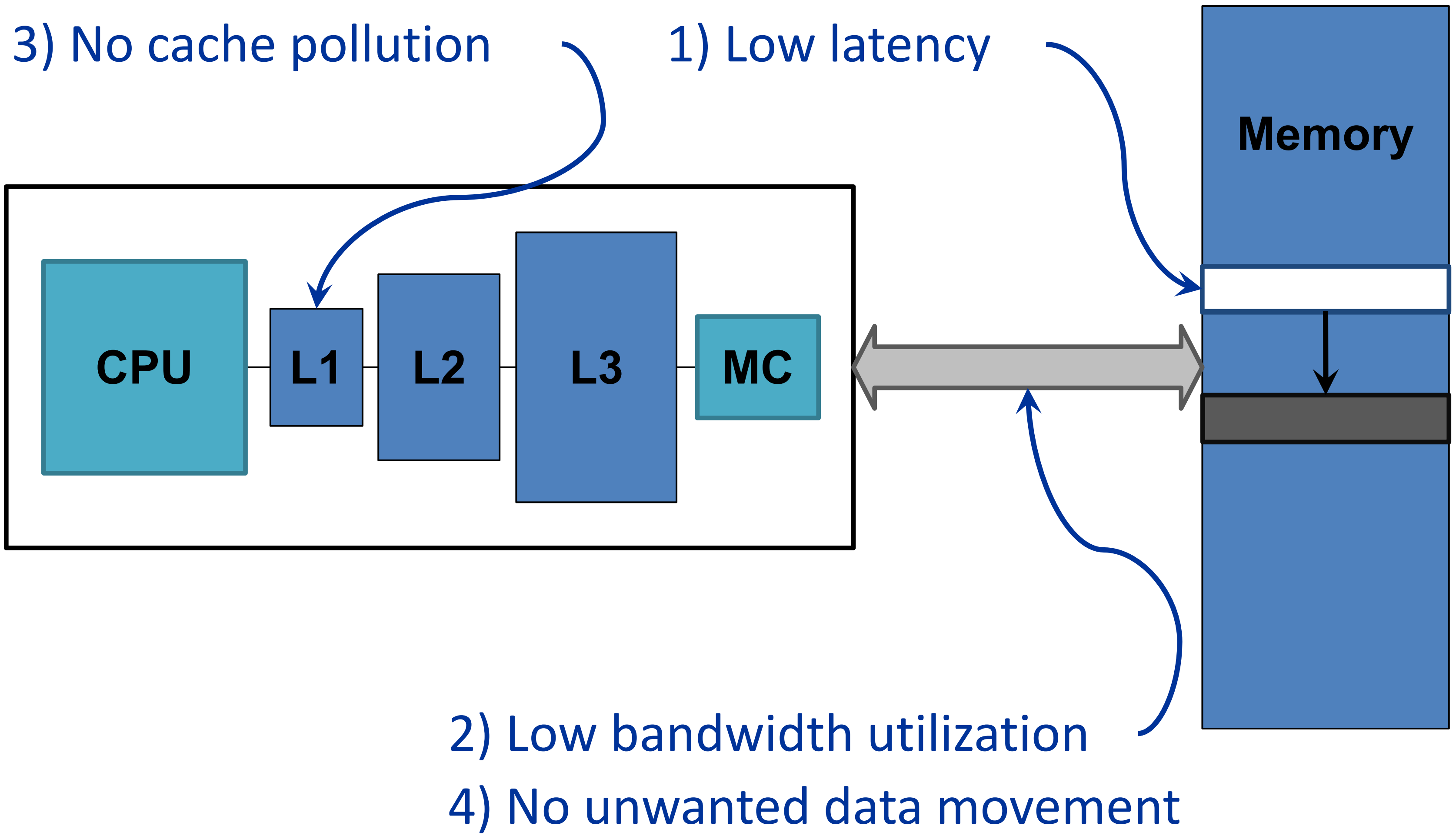
Today's Systems: Bulk Data Copy



4) Unwanted data movement

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

Future Systems: In-Memory Copy

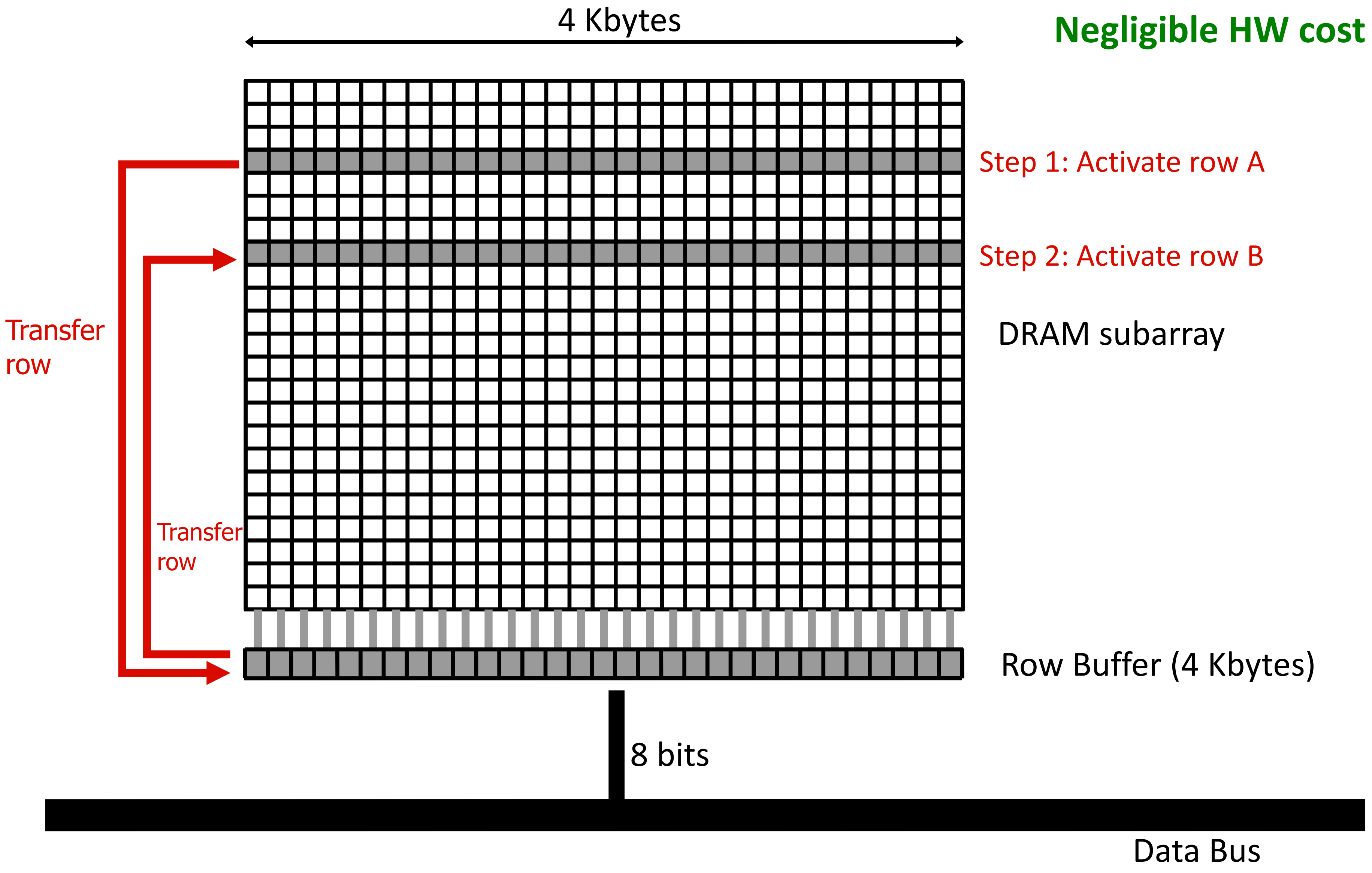


1046ns, 3.6uJ → 90ns, 0.04uJ

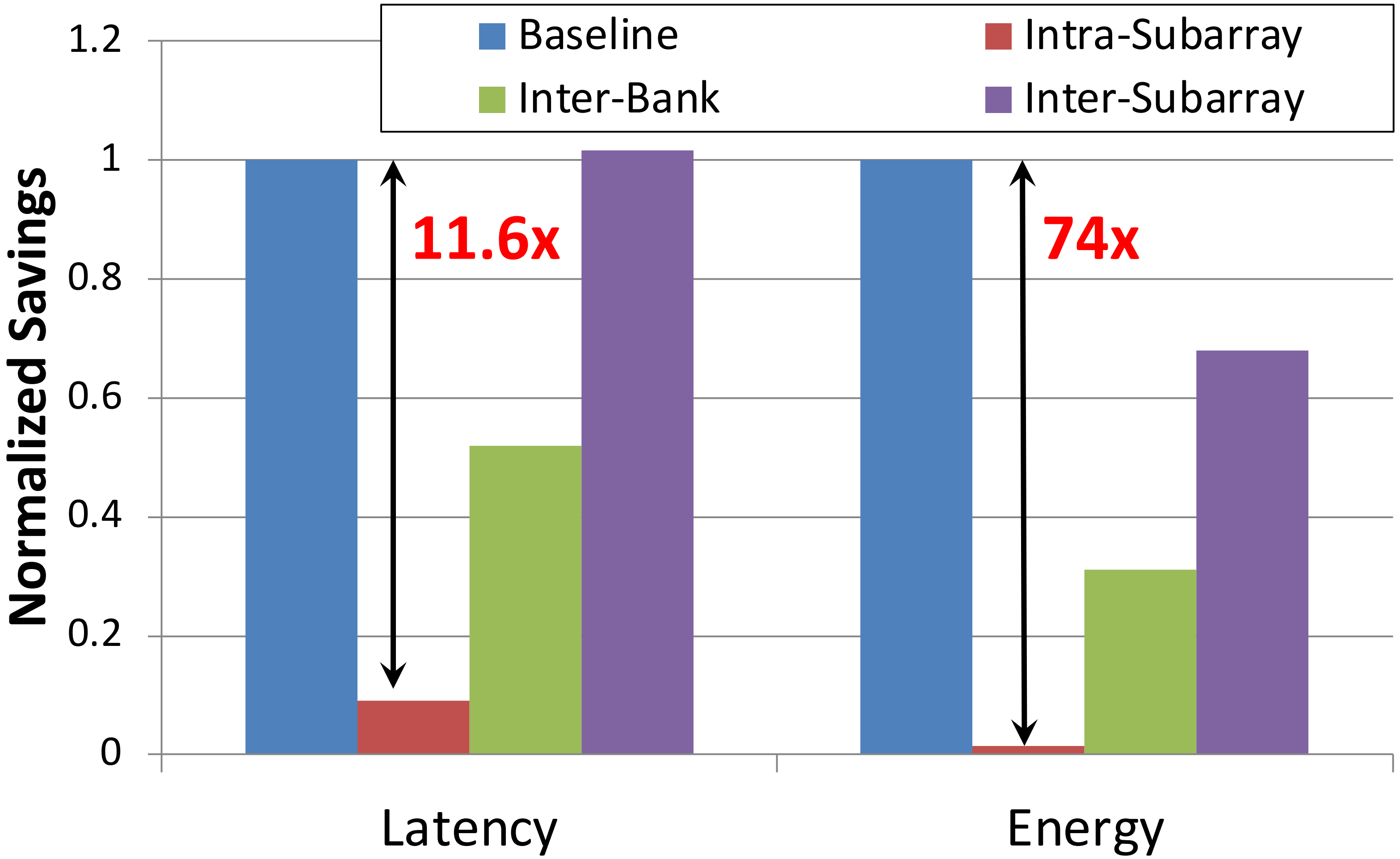
RowClone: In-DRAM Row Copy

Idea: Two consecutive ACTivates

Negligible HW cost



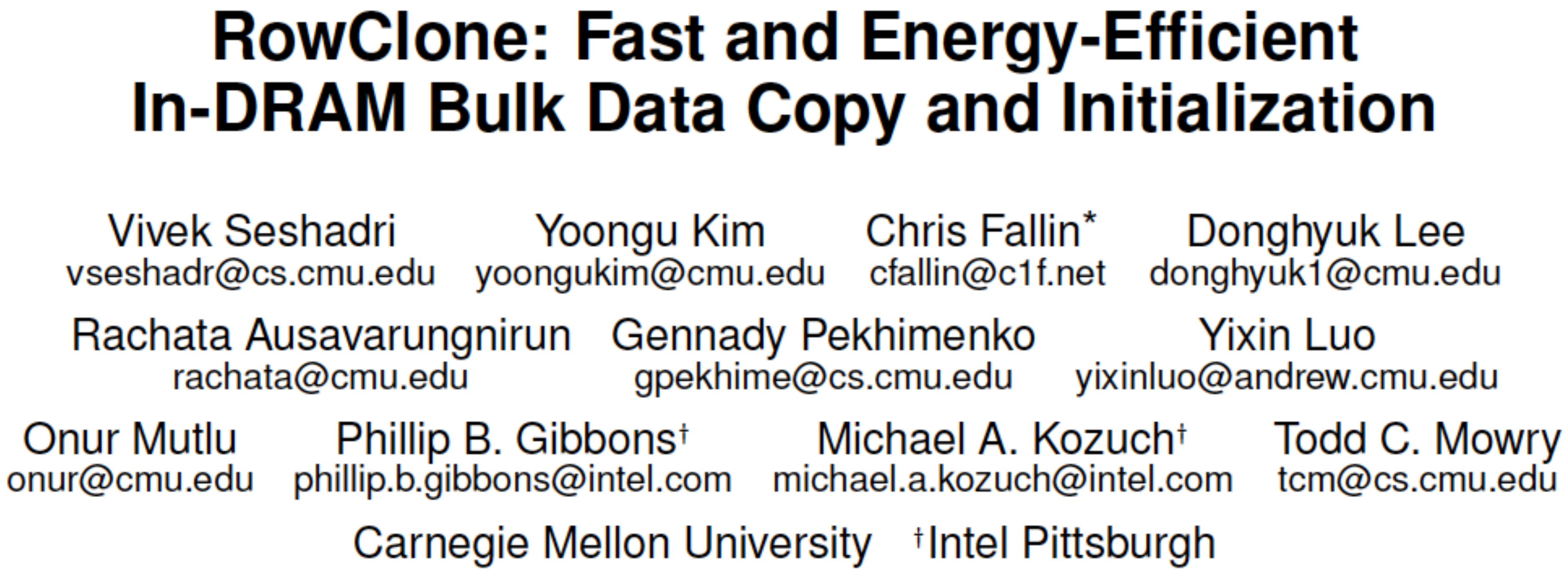
RowClone: Latency and Energy Savings



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)]

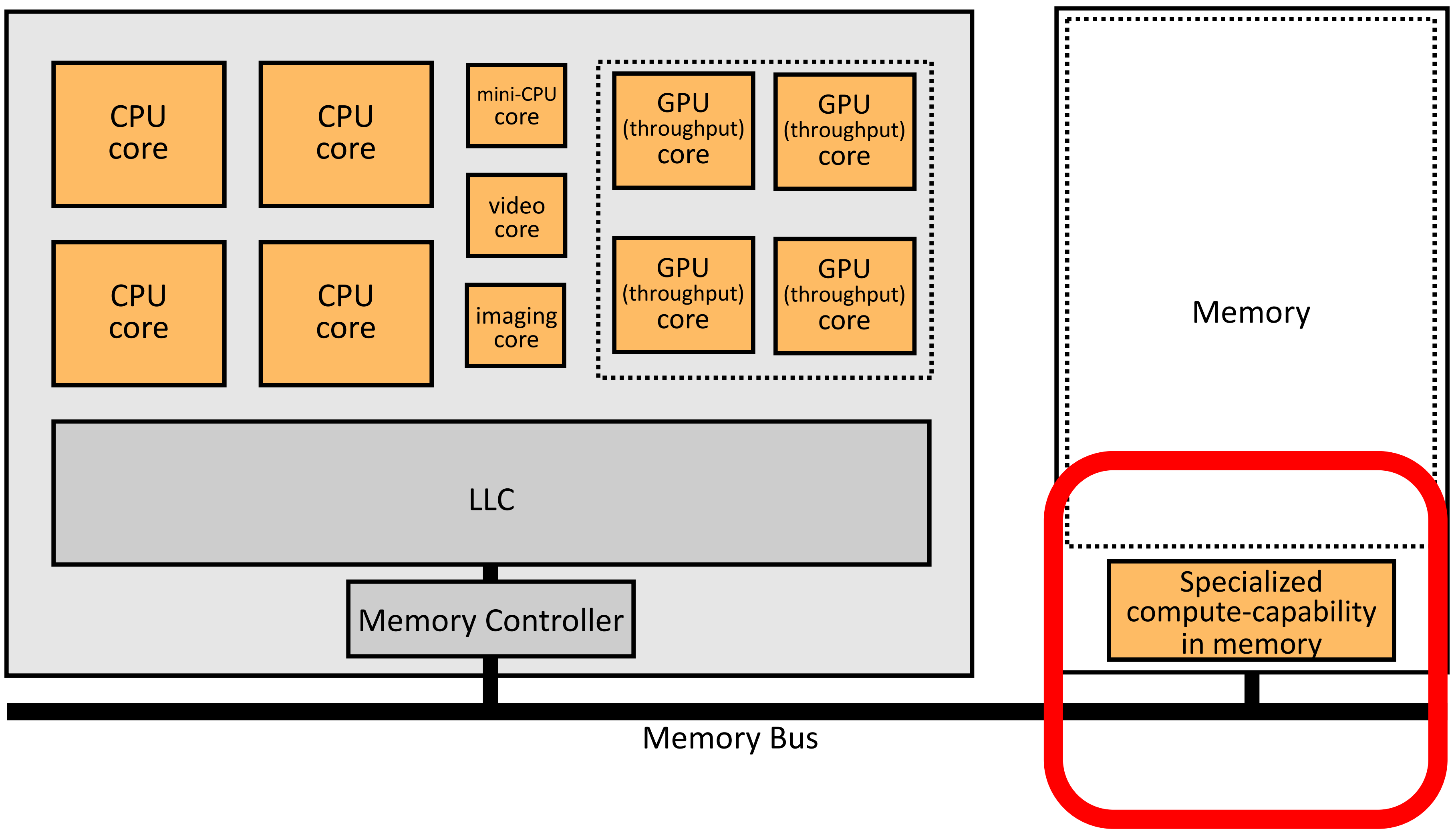


RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization

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

Memory as an Accelerator



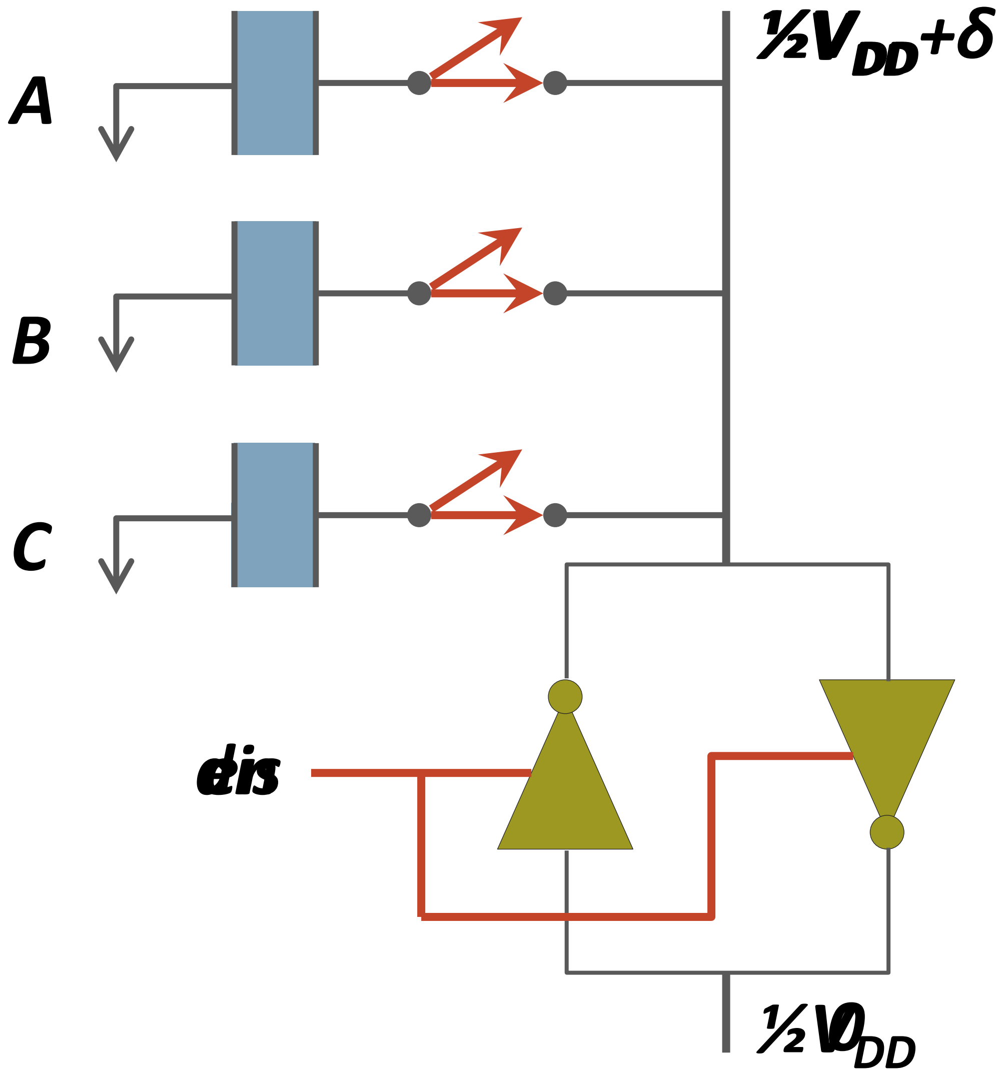
Memory similar to a "conventional" accelerator

In-Memory Bulk Bitwise Operations

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

In-DRAM NOT: Dual Contact Cell

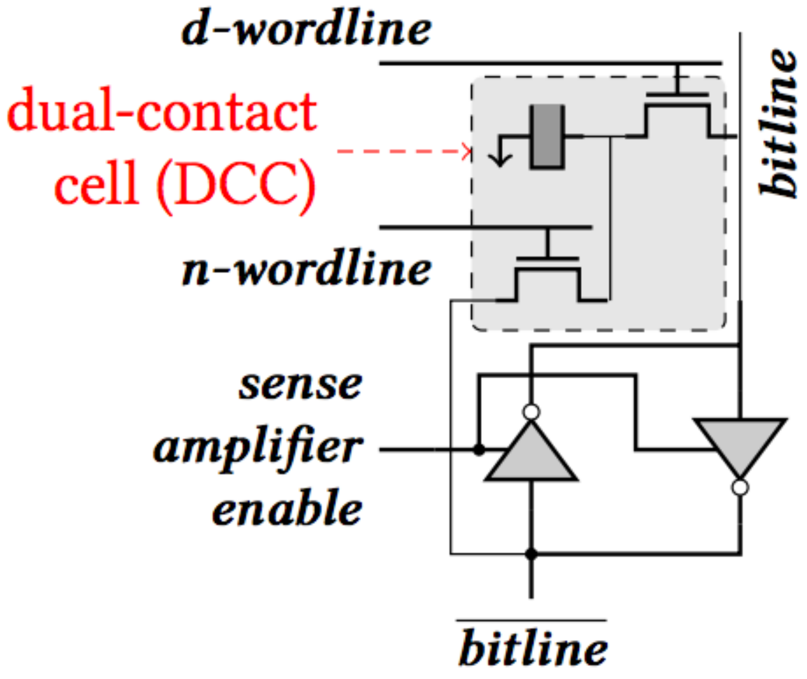
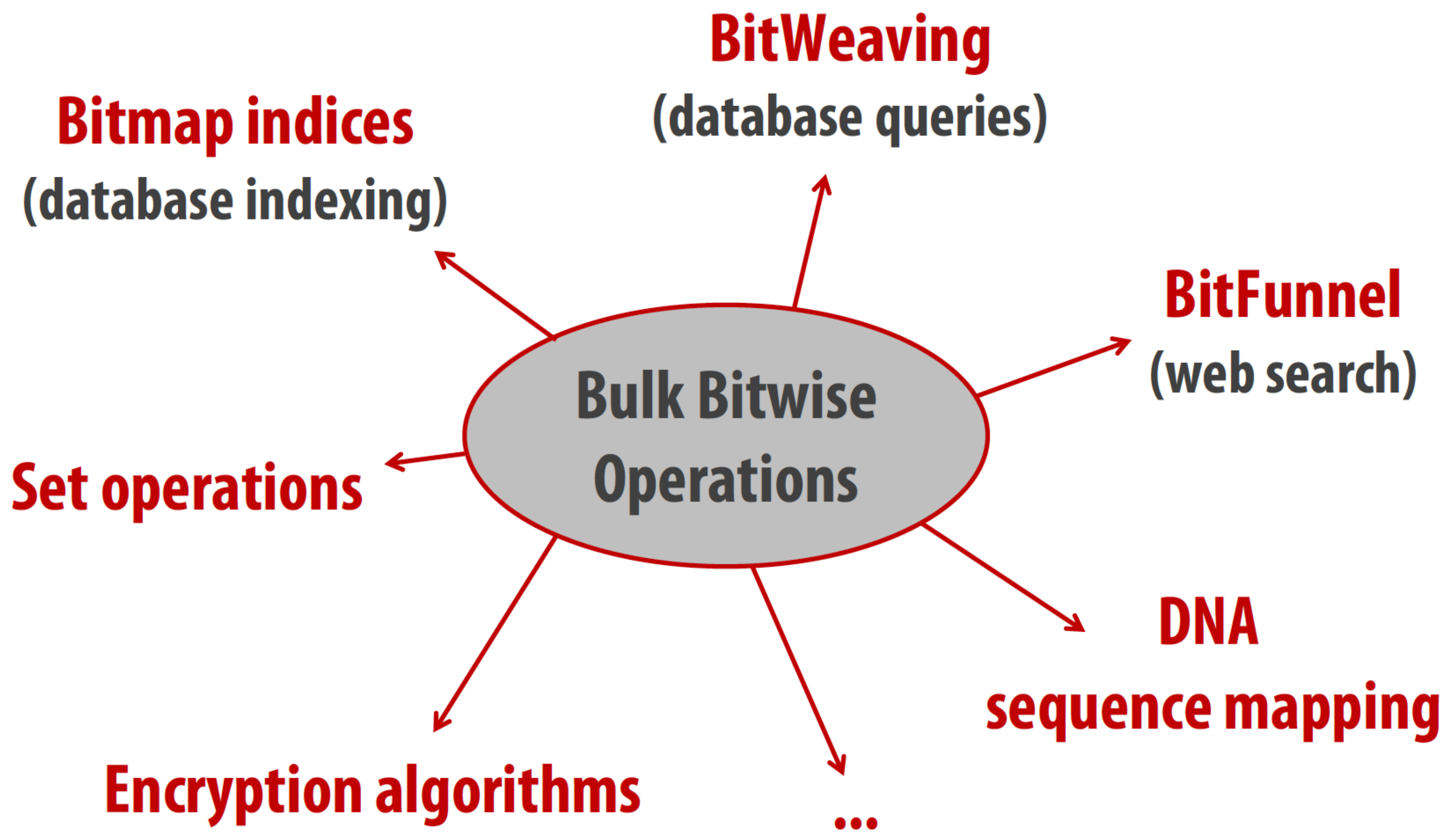


Figure 5: A dual-contact cell connected to both ends of a sense amplifier

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

Bulk Bitwise Operations in Workloads



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

Performance: Bitmap Index on Ambit

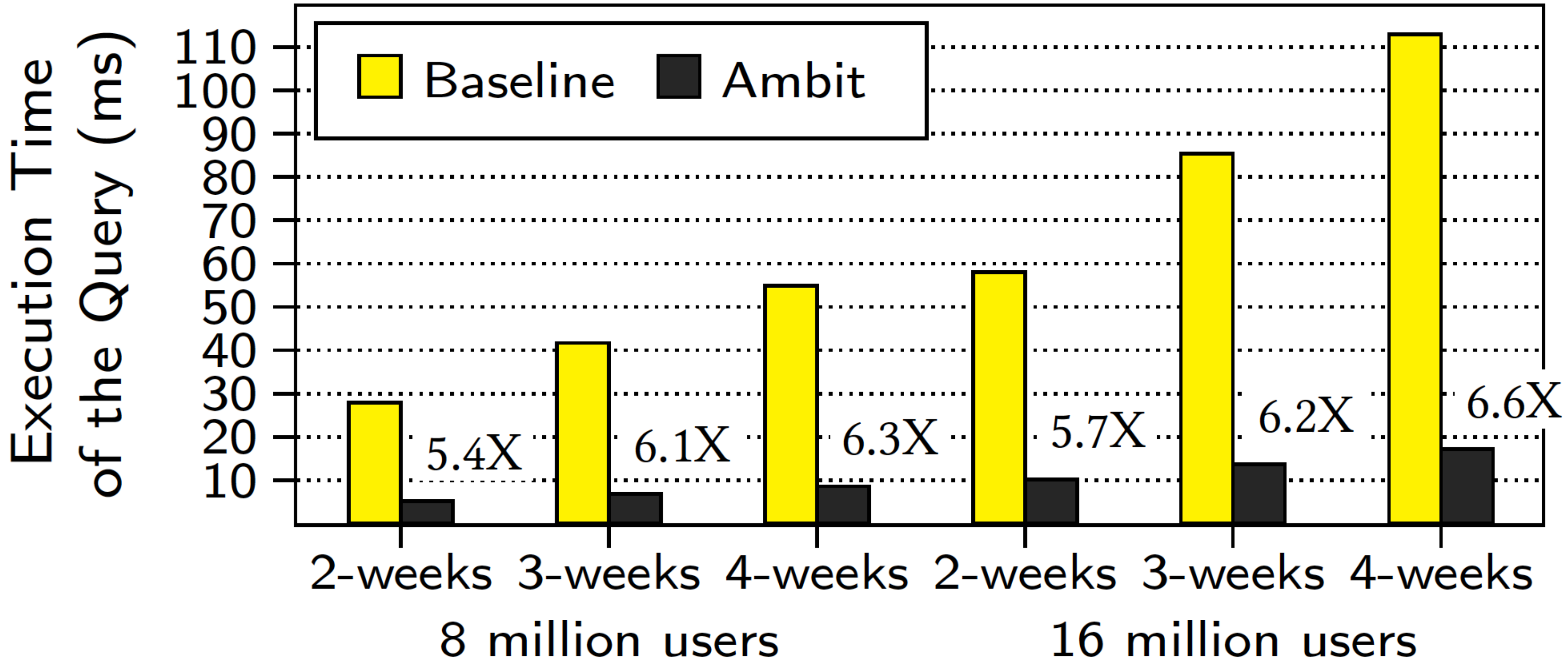


Figure 10: Bitmap index performance. The value above each bar indicates the reduction in execution time due to Ambit.

>5.4-6.6X Performance Improvement

Performance: BitWeaving on Ambit

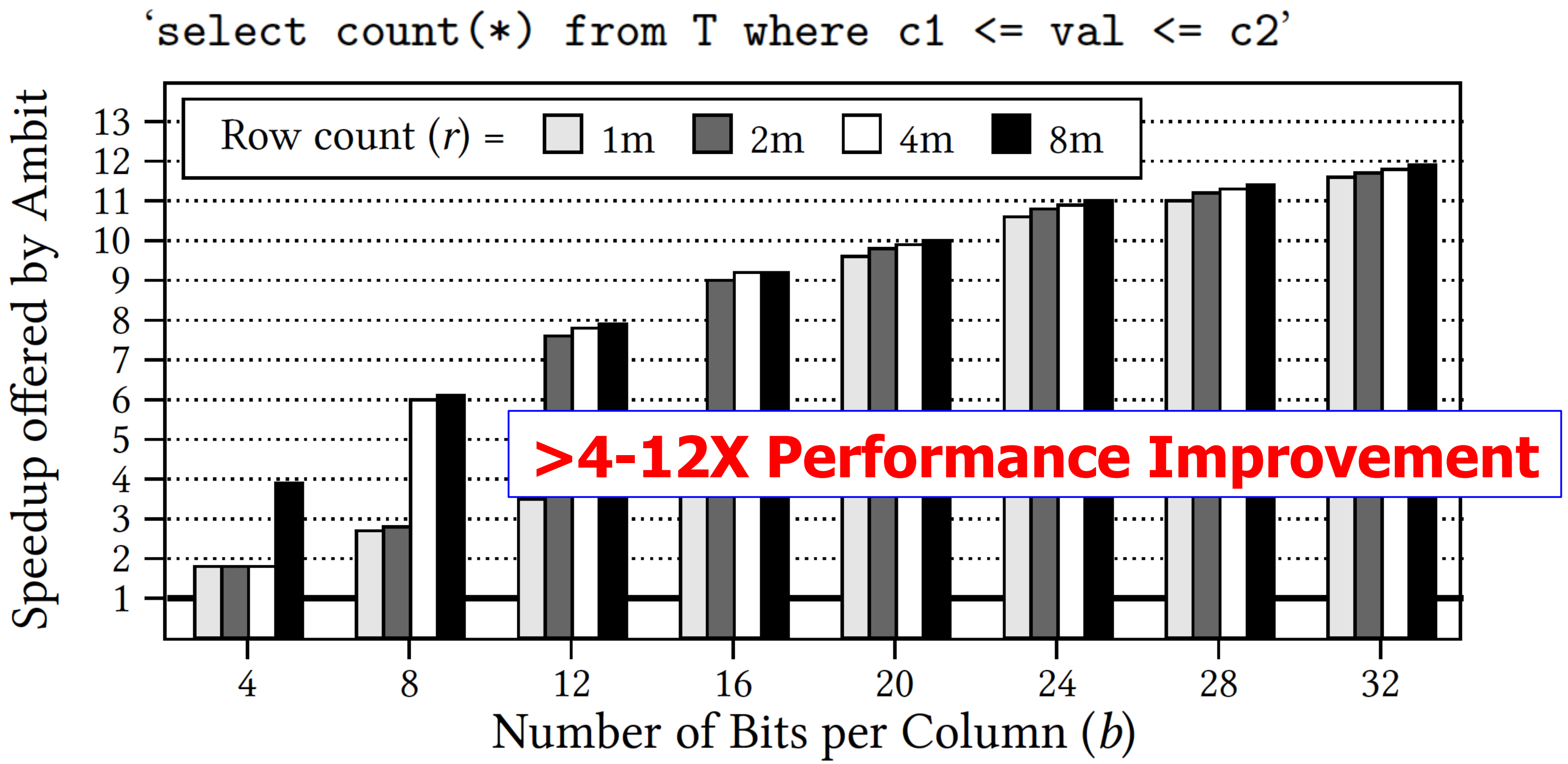
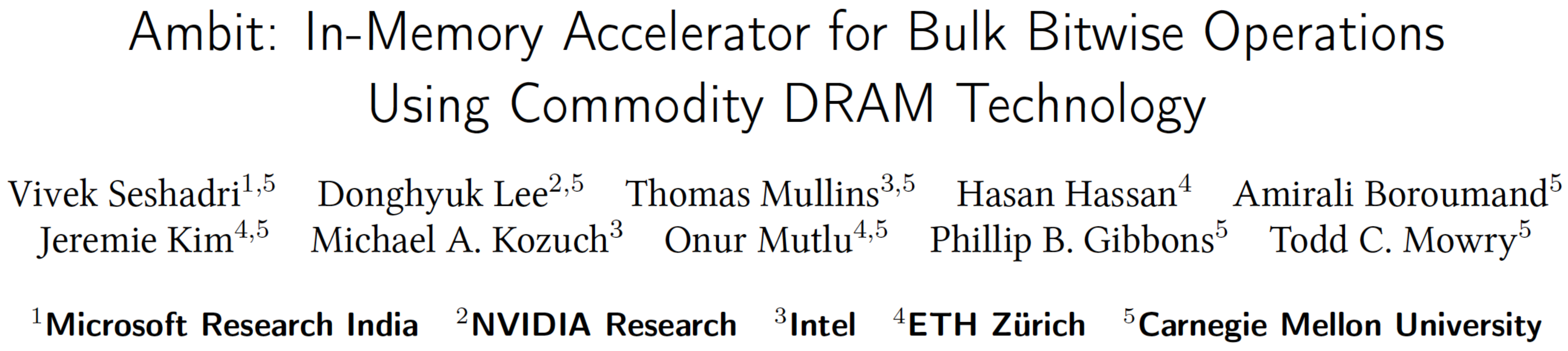


Figure 11: Speedup offered by Ambit over baseline CPU with SIMD for BitWeaving

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

More on Ambit

- Vivek Seshadri et al., "**Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology**," MICRO 2017.



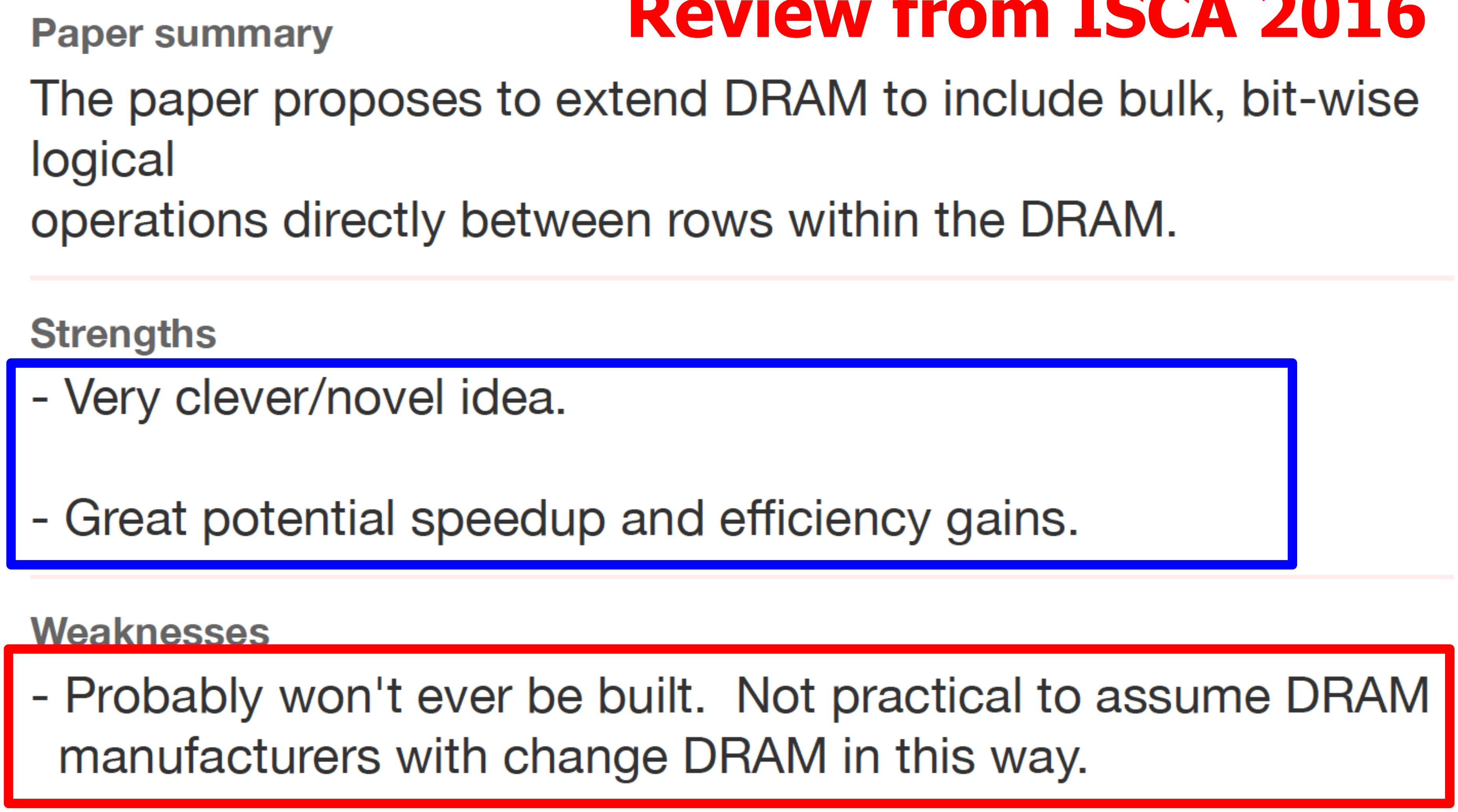
Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology

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

Sounds Good, No?

Review from ISCA 2016



Paper summary

The paper proposes to extend DRAM to include bulk, bit-wise logical
operations directly between rows within the DRAM.

Strengths

- Very clever/novel idea.

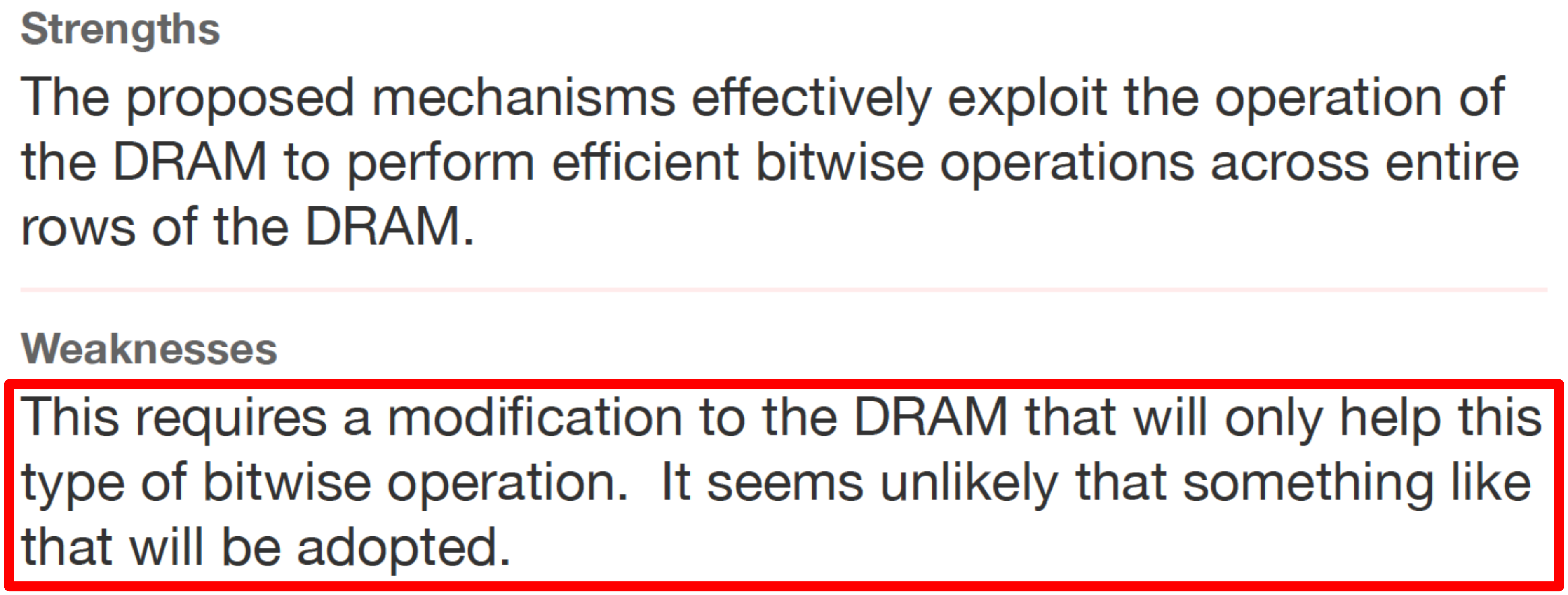
- Great potential speedup and efficiency gains.

Weaknesses

- Probably won't ever be built. Not practical to assume DRAM manufacturers with change DRAM in this way.

Another Review

Another Review from ISCA 2016



Strengths

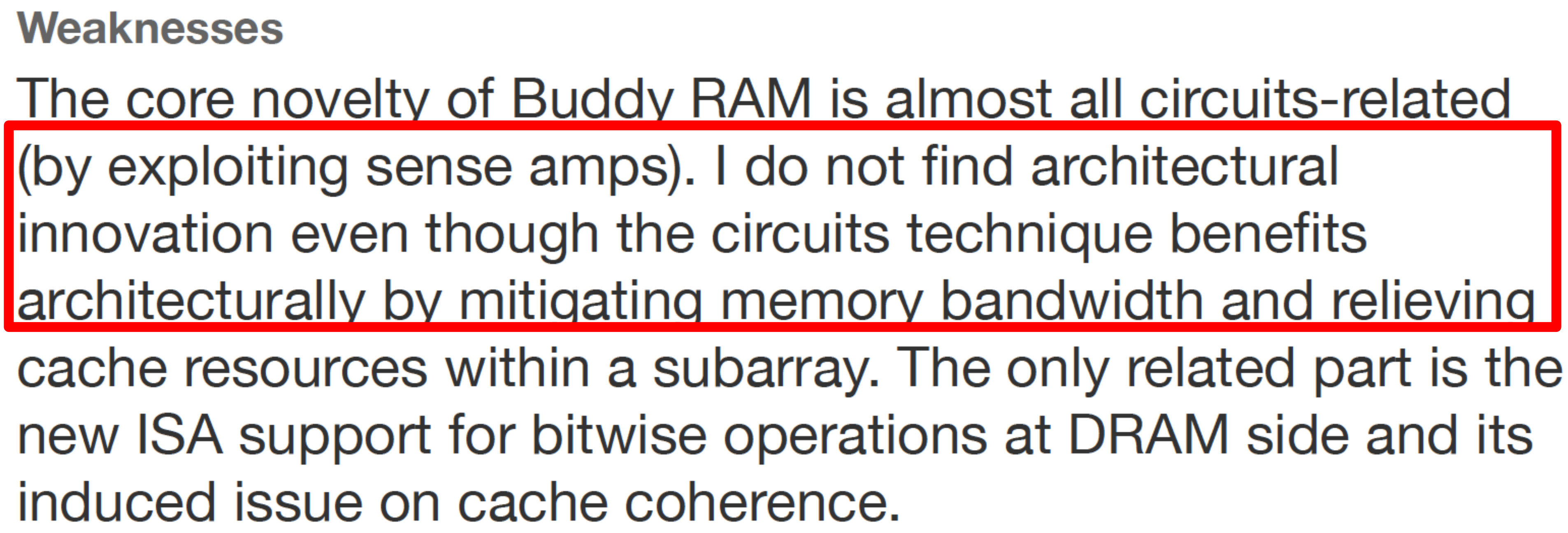
The proposed mechanisms effectively exploit the operation of the DRAM to perform efficient bitwise operations across entire rows of the DRAM.

Weaknesses

This requires a modification to the DRAM that will only help this type of bitwise operation. It seems unlikely that something like that will be adopted.

Yet Another Review

Yet Another Review from ISCA 2016



Weaknesses

The core novelty of Buddy RAM is almost all circuits-related (by exploiting sense amps). I do not find architectural innovation even though the circuits technique benefits architecturally by mitigating memory bandwidth and relieving cache resources within a subarray. The only related part is the new ISA support for bitwise operations at DRAM side and its induced issue on cache coherence.

We Have a Mindset Issue…

- There are many other similar examples from reviews…
  - For many other papers…
- And, we are not even talking about JEDEC yet…
- How do we fix the mindset problem?
- By doing more research, education, implementation in alternative processing paradigms

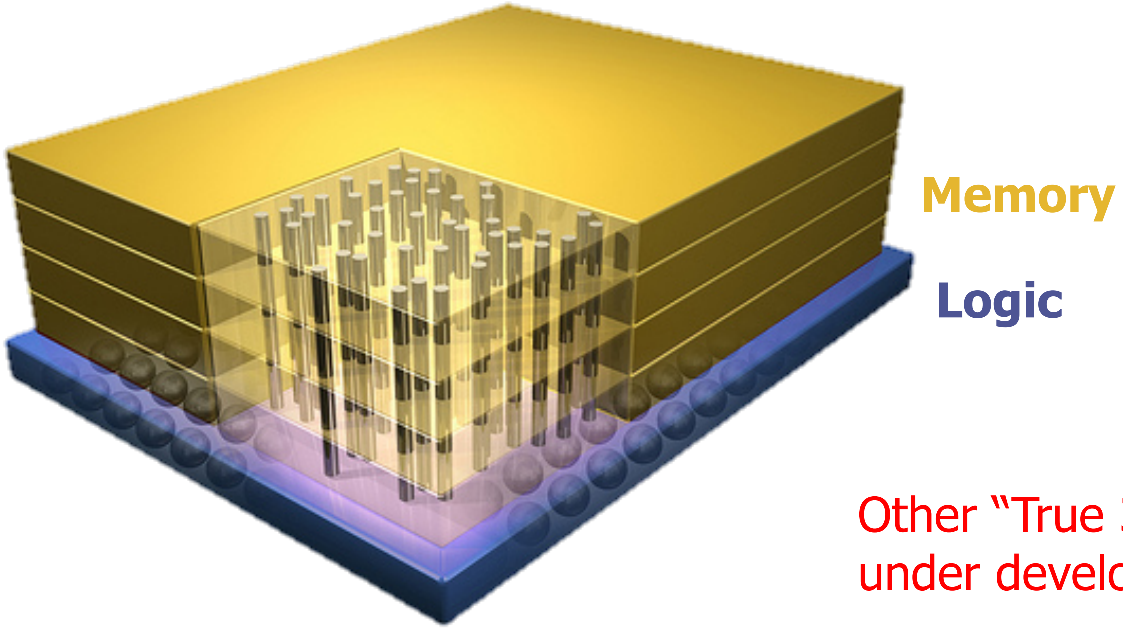
We need to work on enabling the better future…

Processing in Memory: Two Approaches

1. Minimally changing memory chips
2. Exploiting 3D-stacked memory

Opportunity: 3D-Stacked Logic+Memory

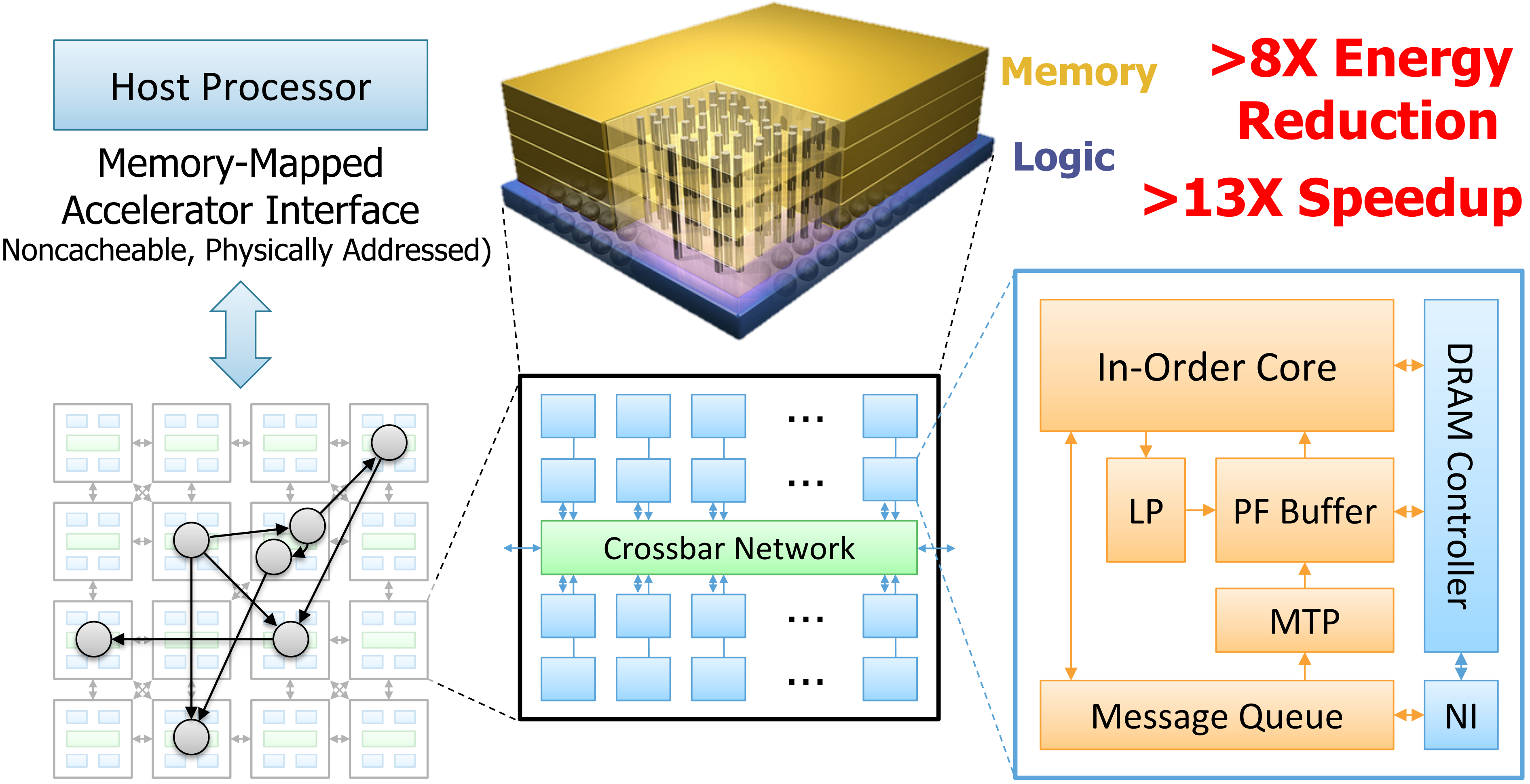
Hybrid Memory Cube
CONSORTIUM



Other "True 3D" technologies
under development

Tesseract System for Graph Processing

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

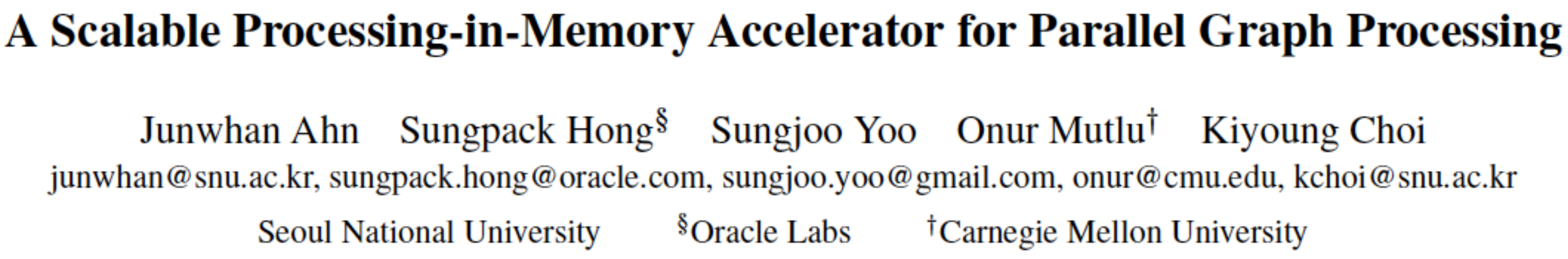


>8X Energy Reduction

>13X Speedup

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

Another Example: 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.

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]

Four Important Workloads

Chrome

Google's web browser

TensorFlow Mobile

Google's machine learning framework

Video Playback

Google's video codec

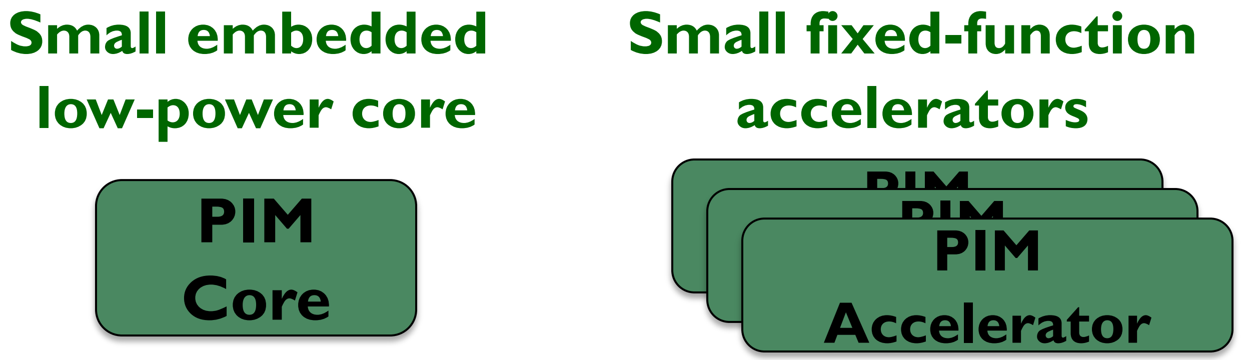
Video Capture

Google's video codec

Simple PIM on Mobile Workloads

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

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


Small embedded low-power core
PIM Core
Small fixed-function accelerators
PIM
PIM
PIM Accelerator

Offloading to PIM logic reduces energy and execution time, on average, by 55.4% and 54.2%

Eliminating the Adoption Barriers

How to Enable Adoption of Processing in Memory

Barriers to Adoption of PIM

1. Functionality of and applications & software for PIM

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

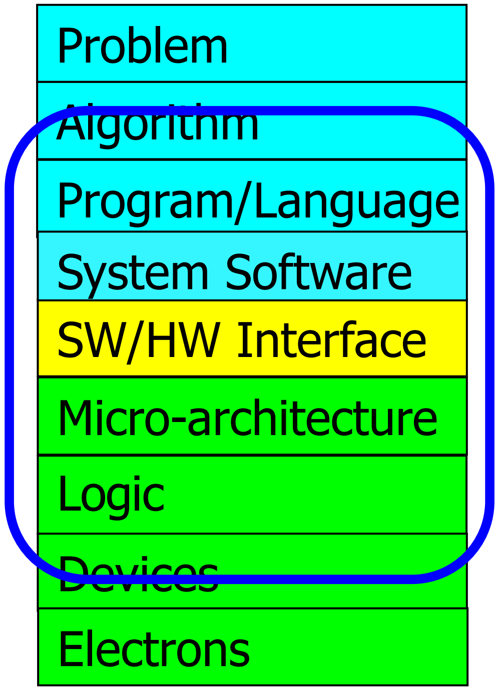
3. System support: coherence & virtual memory

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

5. Infrastructures to assess benefits and feasibility

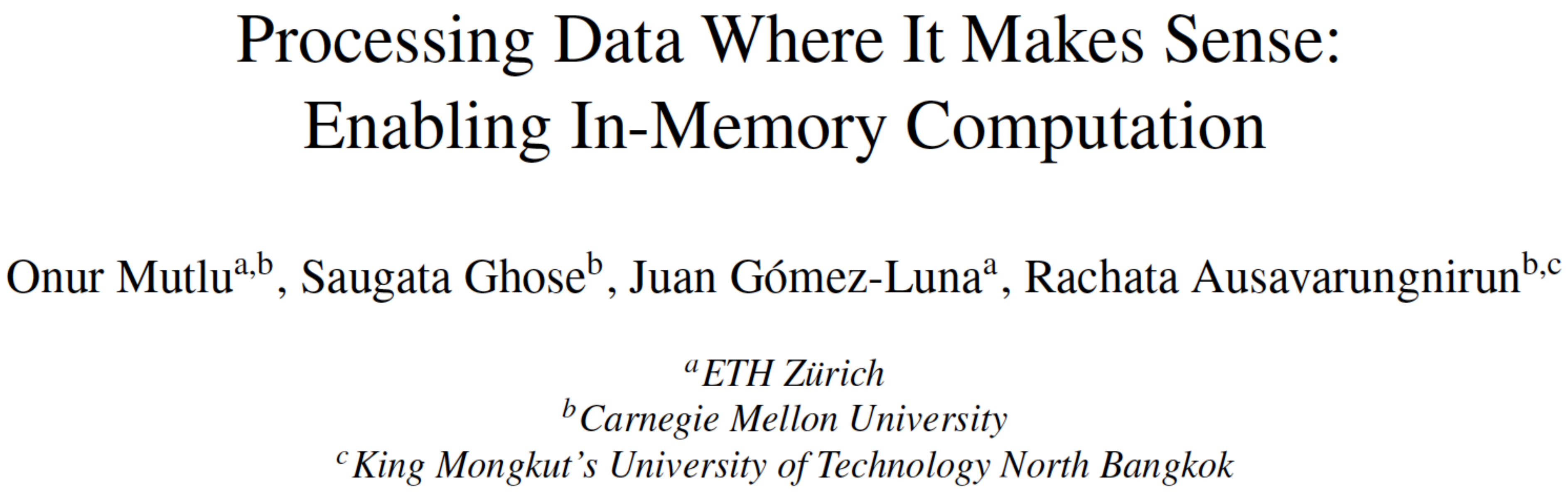
All can be solved with change of mindset

We Need to Revisit the Entire Stack



We can get there step by step

PIM Review and Open Problems



Processing Data Where It Makes Sense: Enabling In-Memory Computation

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

[a] *ETH Zürich*
[b] *Carnegie Mellon University*
[c] *King Mongkut's University of Technology North Bangkok*

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]

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

Challenge and Opportunity for Future

Computing Architectures

with

Minimal Data Movement

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

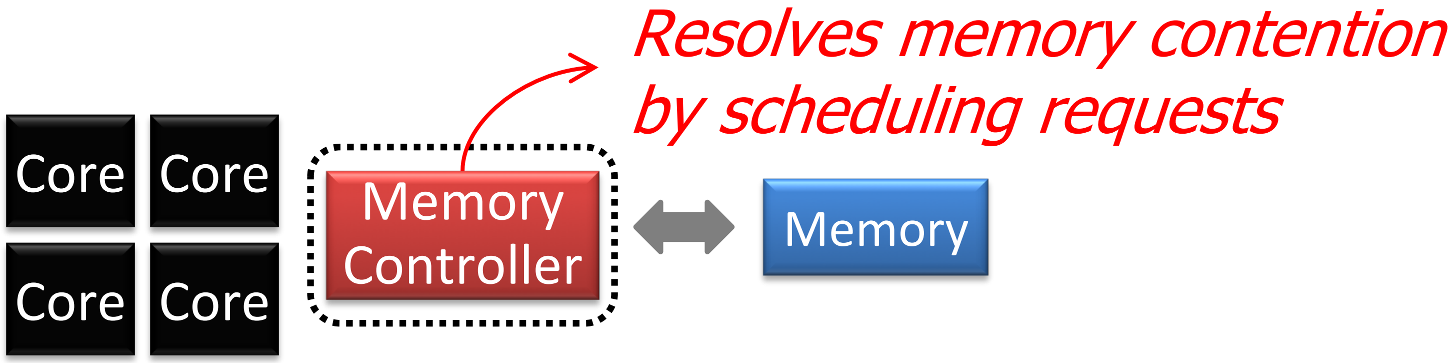
- Automatic data-driven policy learning

- All controllers are intelligent data-driven agents

How do we start?

Self-Optimizing Memory Controllers

Memory Controller

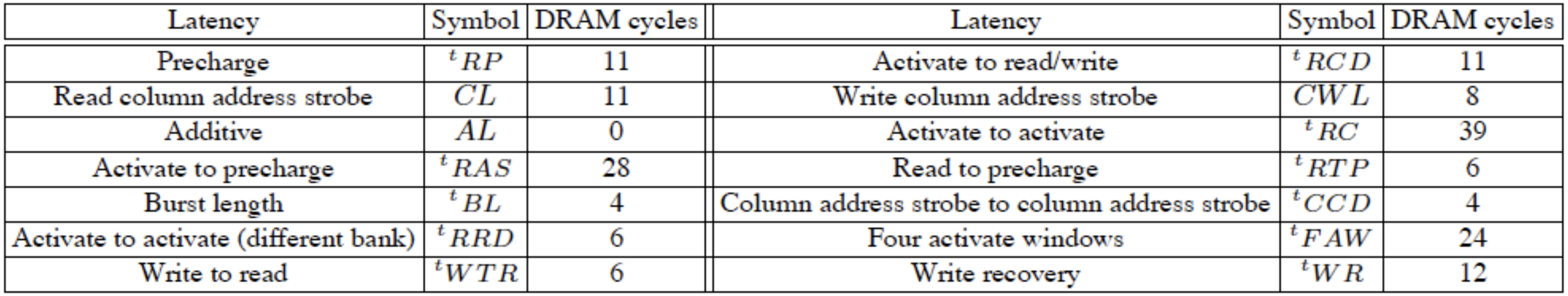


How to schedule requests to maximize system performance?

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	^tRP	11	Activate to read/write	^tRCD	11
Read column address strobe	CL	11	Write column address strobe	CWL	8
Additive	AL	0	Activate to activate	^tRC	39
Activate to precharge	^tRAS	28	Read to precharge	^tRTP	6
Burst length	^tBL	4	Column address strobe to column address strobe	^tCCD	4
Activate to activate (different bank)	^tRRD	6	Four activate windows	^tFAW	24
Write to read	^tWTR	6	Write recovery	^tWR	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.

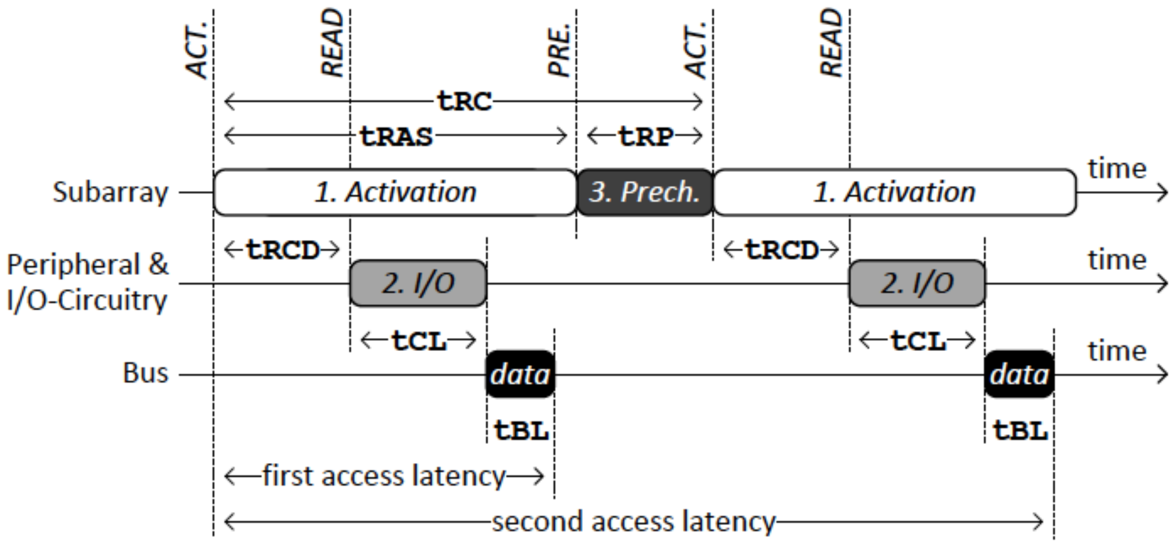


Figure 5. Three Phases of DRAM Access

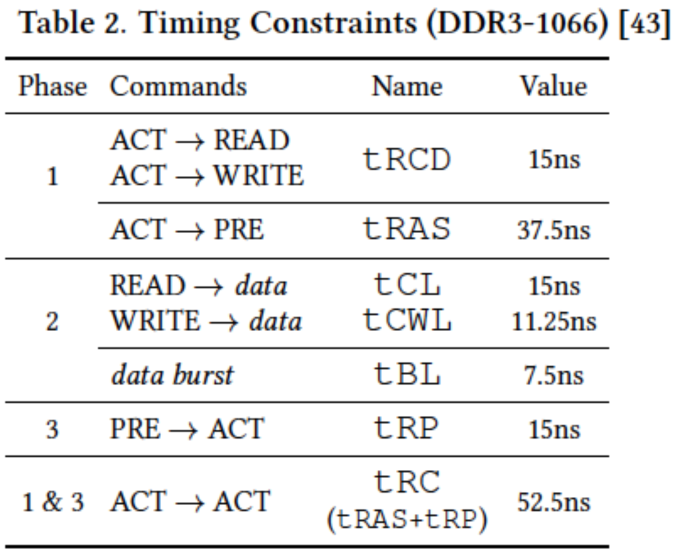
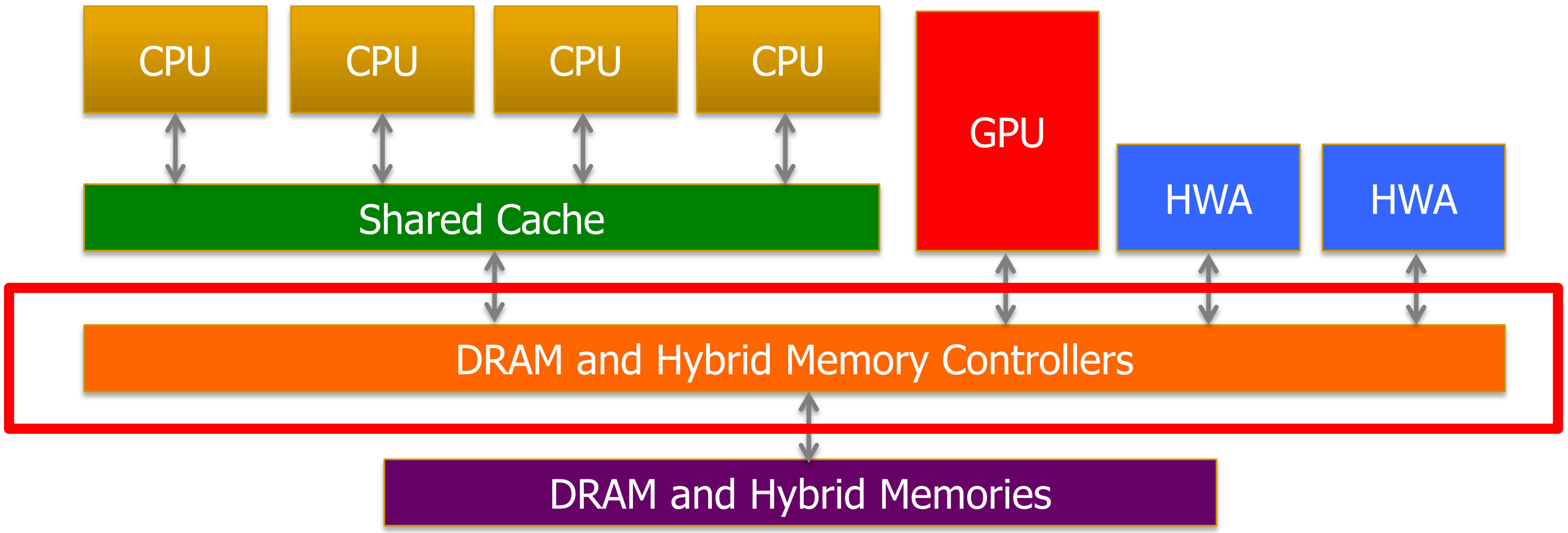


Table 2. Timing Constraints (DDR3-1066) [43]

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

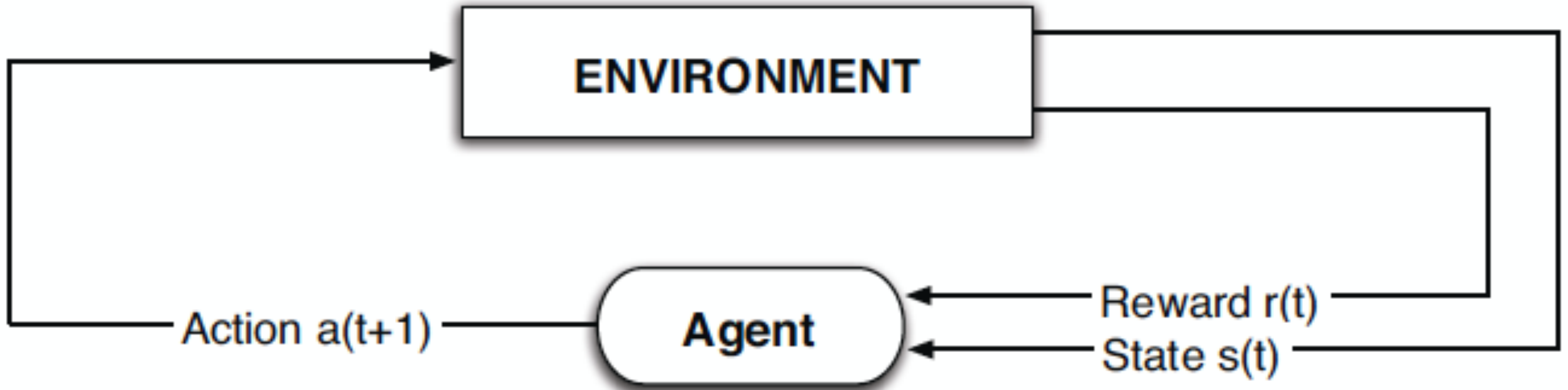
- Idea: A memory controller that adapts its scheduling policy to workload behavior and system conditions using machine learning.

- 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

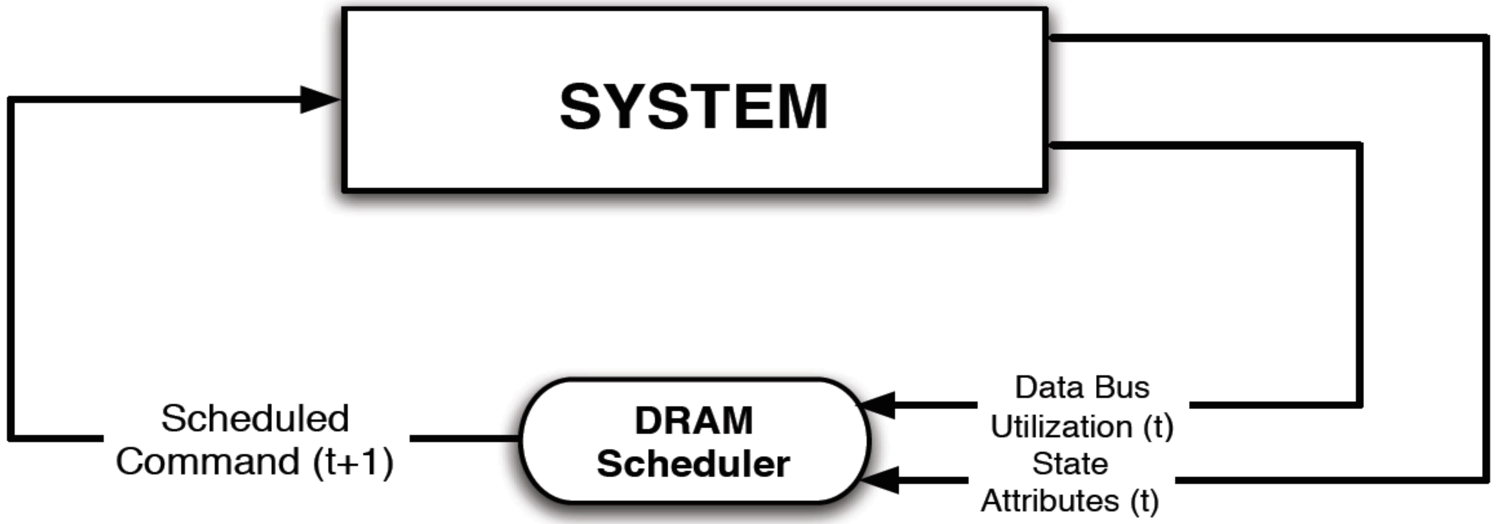


Goal: Learn to choose actions to maximize $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$ ($0 \leq \gamma < 1$)

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.

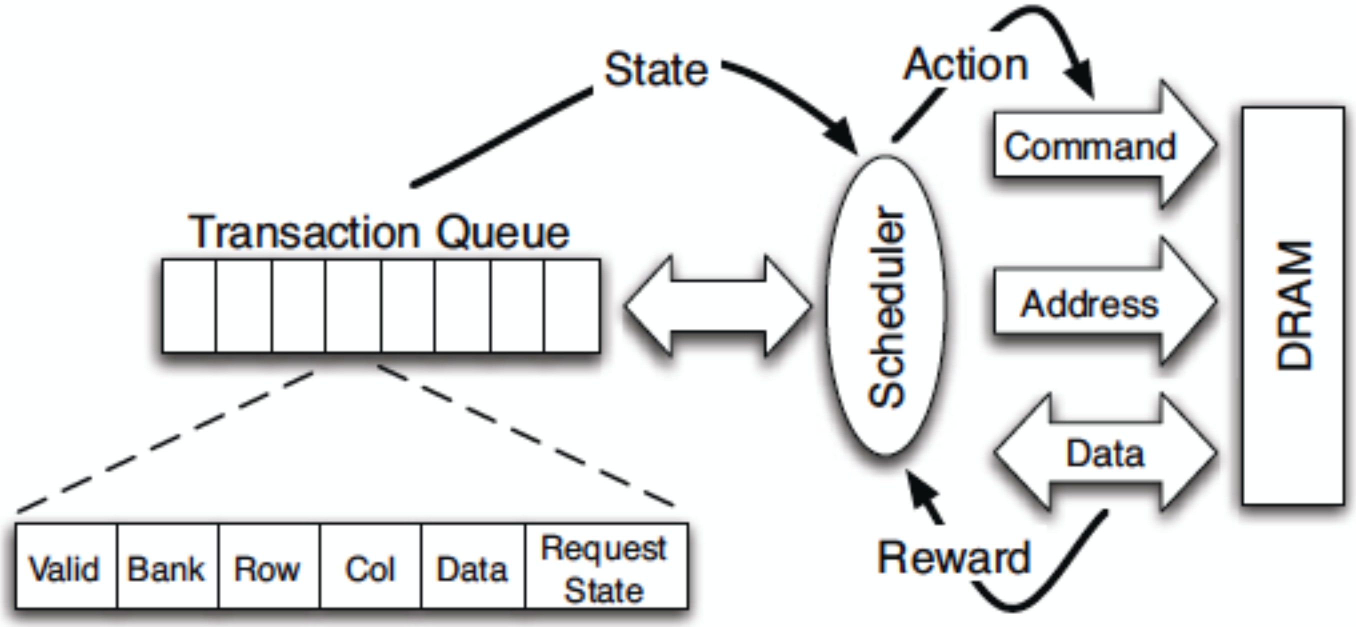


Figure 4: High-level overview of an RL-based scheduler.

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

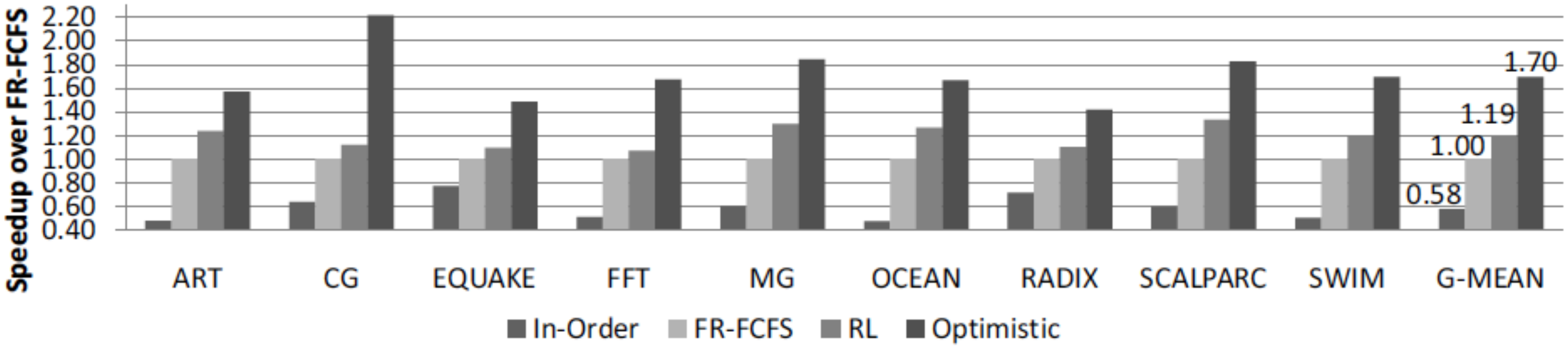


Figure 7: Performance comparison of in-order, FR-FCFS, RL-based, and optimistic memory controllers

Large, robust performance improvements over many human-designed policies

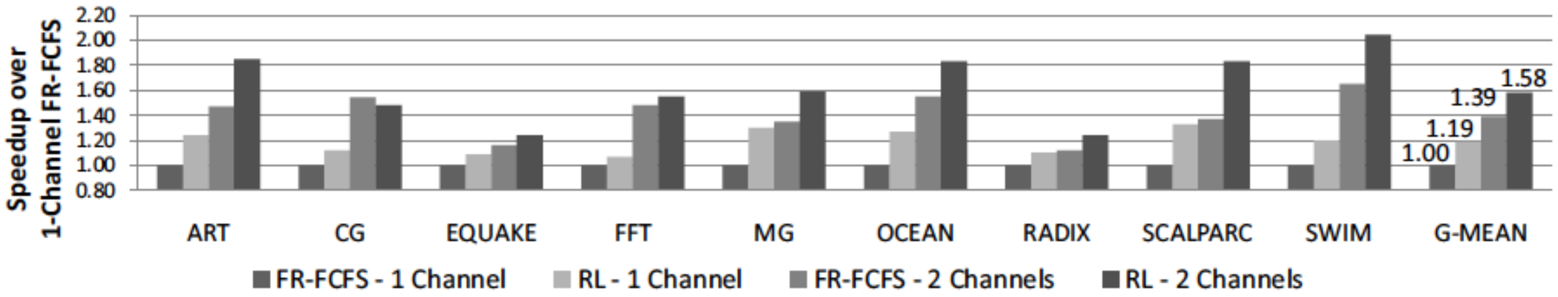


Figure 15: Performance comparison of FR-FCFS and RL-based memory controllers on systems with 6.4GB/s and 12.8GB/s peak DRAM bandwidth

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

An Intelligent Architecture

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

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

- Automatic data-driven policy learning

- All controllers are intelligent data-driven agents

We need to rethink design
(of all controllers)

Challenge and Opportunity for Future

Self-Optimizing

(Data-Driven)

Computing Architectures

Corollaries: Architectures Today …

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

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

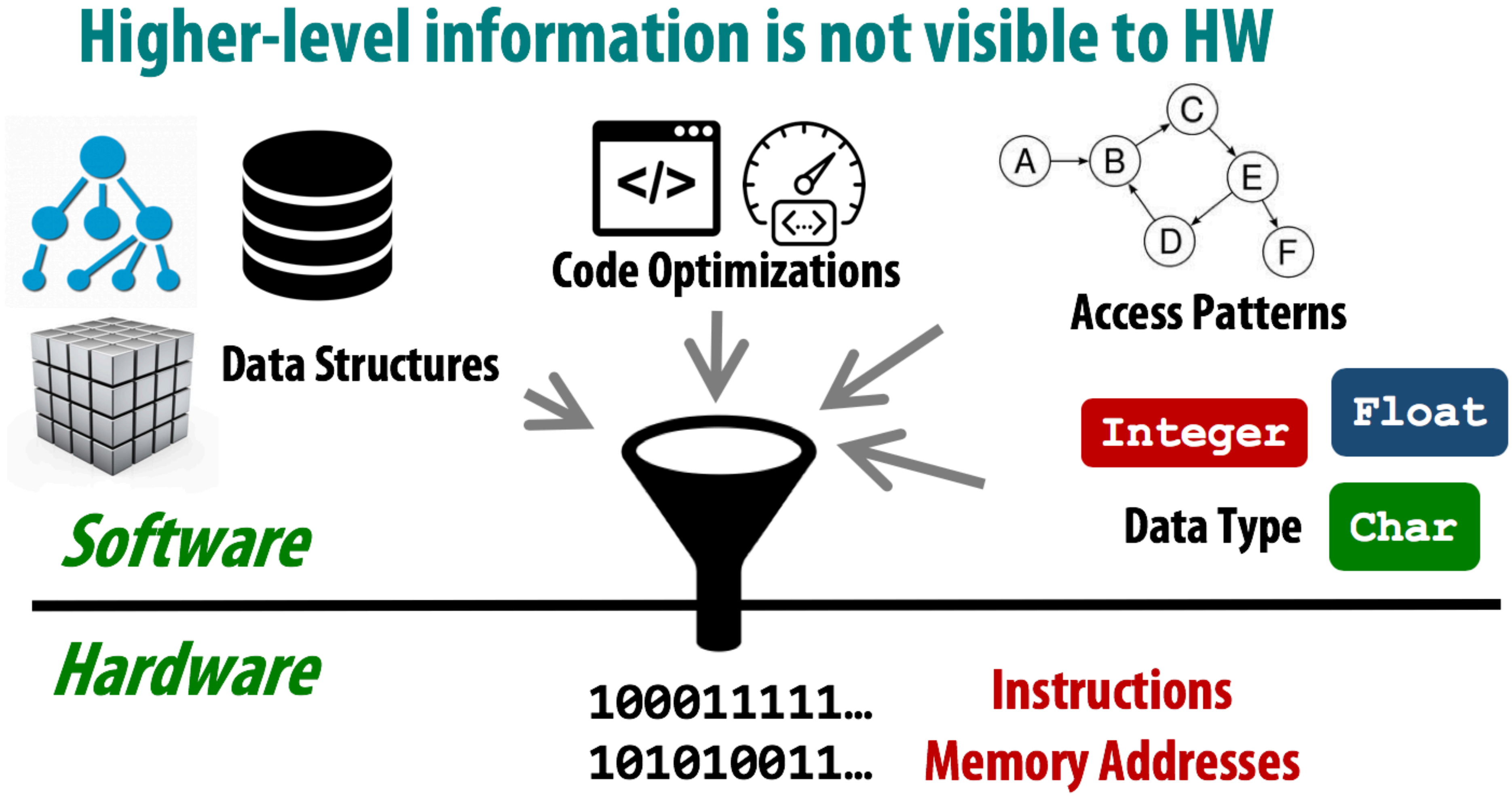
- Architectures are terrible at knowing and exploiting different properties of application data
  - Designed to treat all data as the same
  - They make component-aware decisions vs. **data-aware**

Data-Aware Architectures

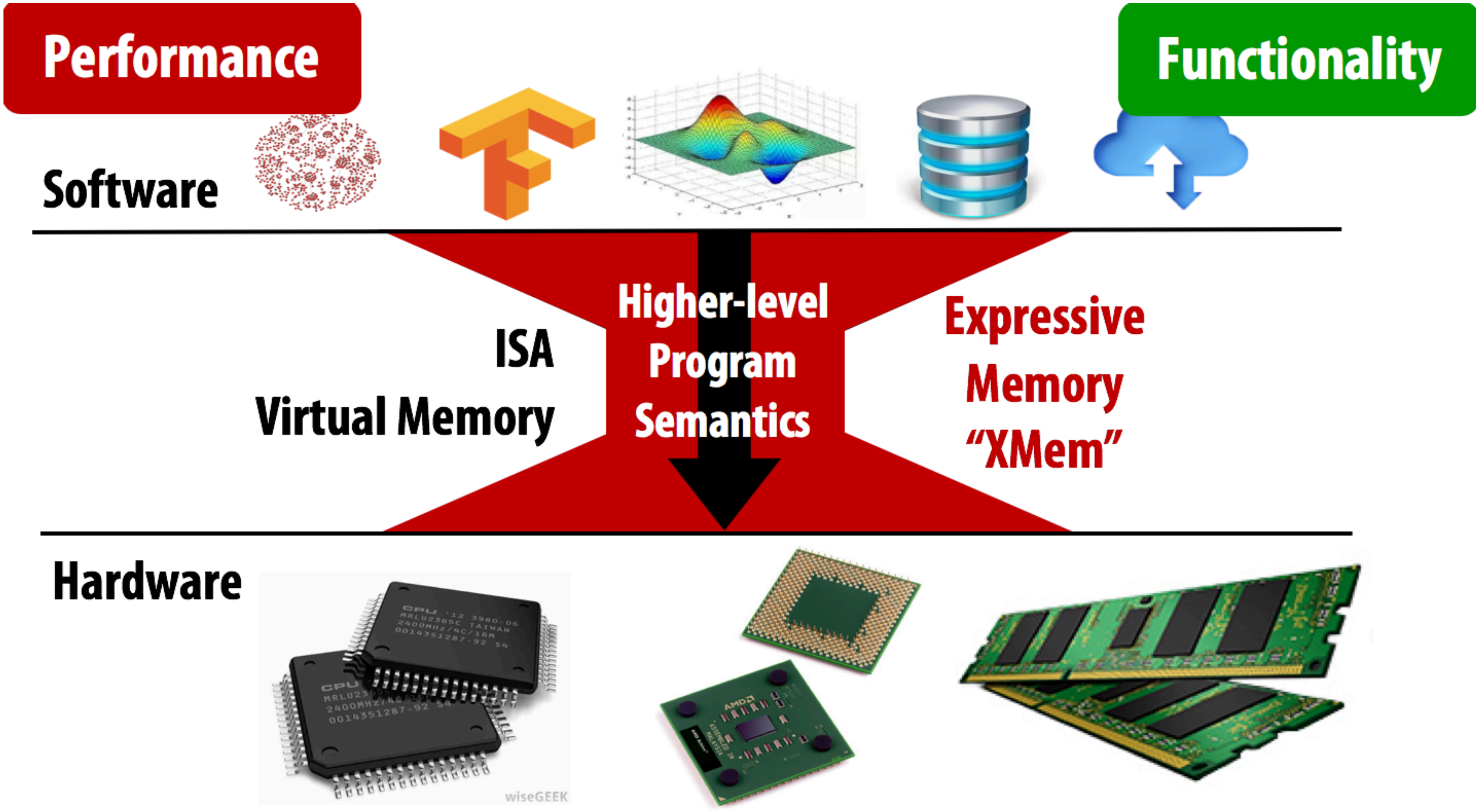
- A data-aware architecture understands what it can do with and to each piece of data

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

One Problem: Limited Interfaces

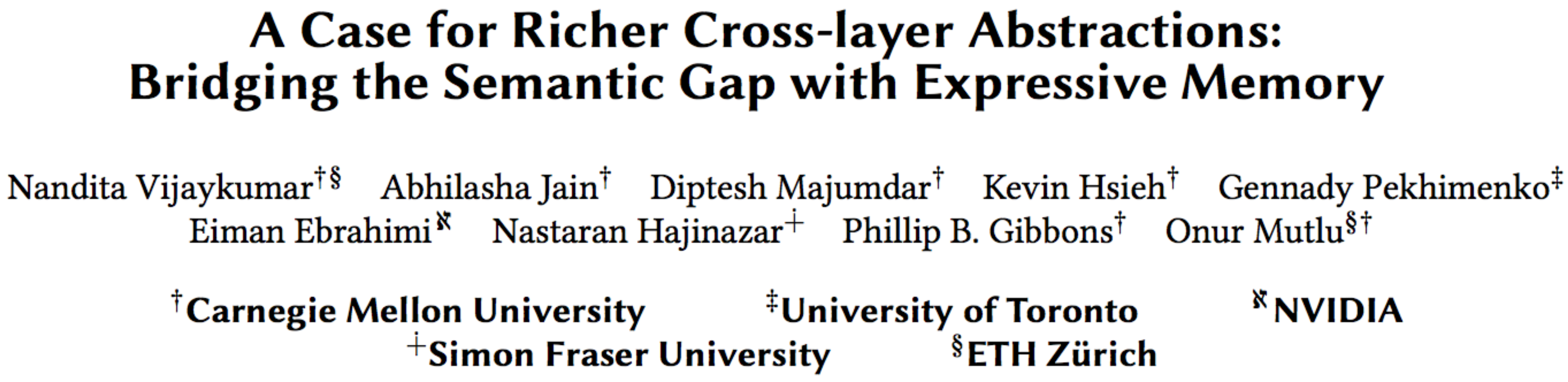


A Solution: More Expressive Interfaces



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]



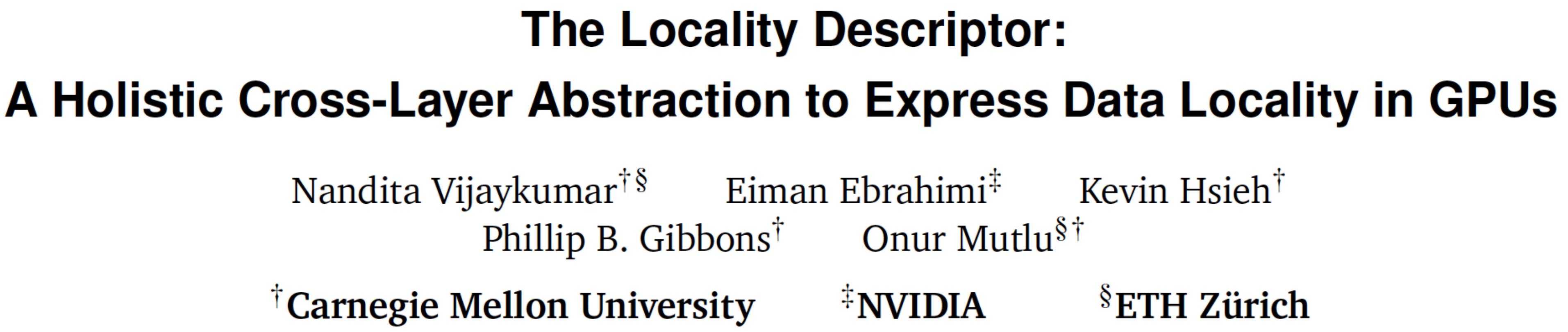
A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory

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

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]

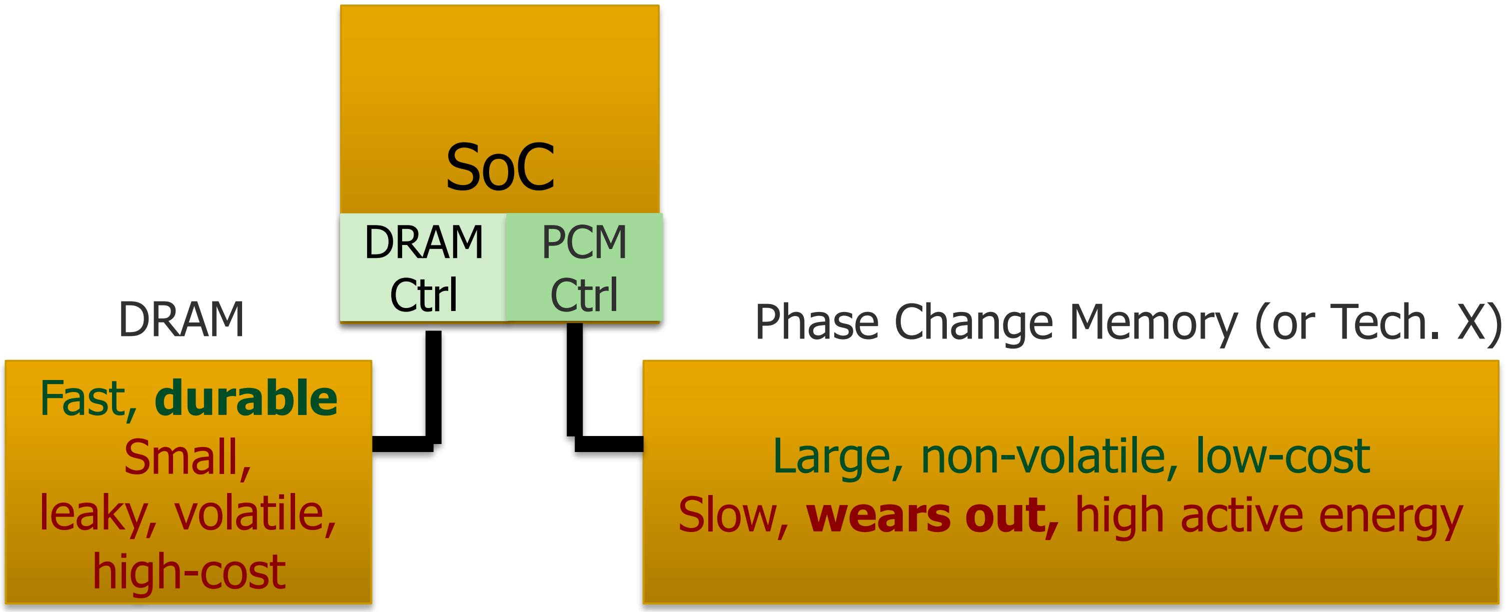


The Locality Descriptor:
A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs

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

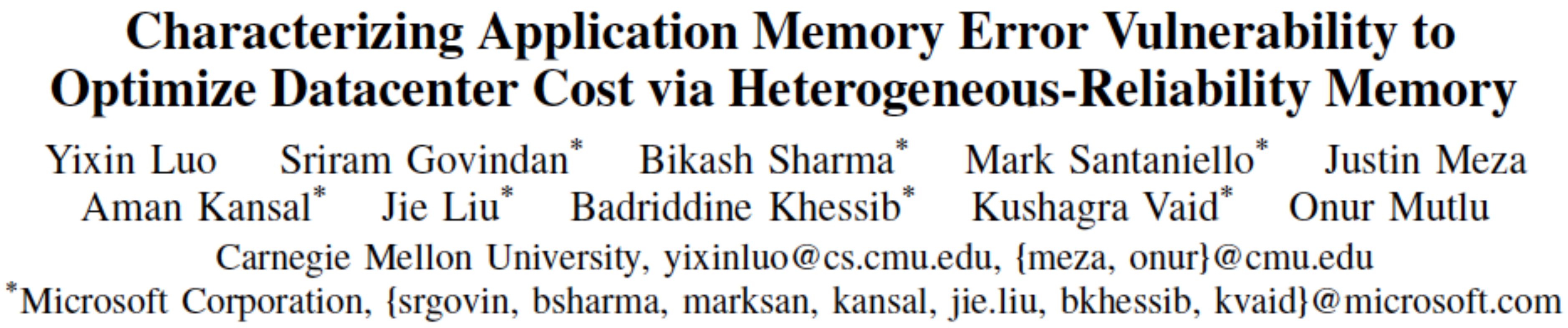
Hardware/software manage data allocation and movement
to achieve the best of multiple technologies

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

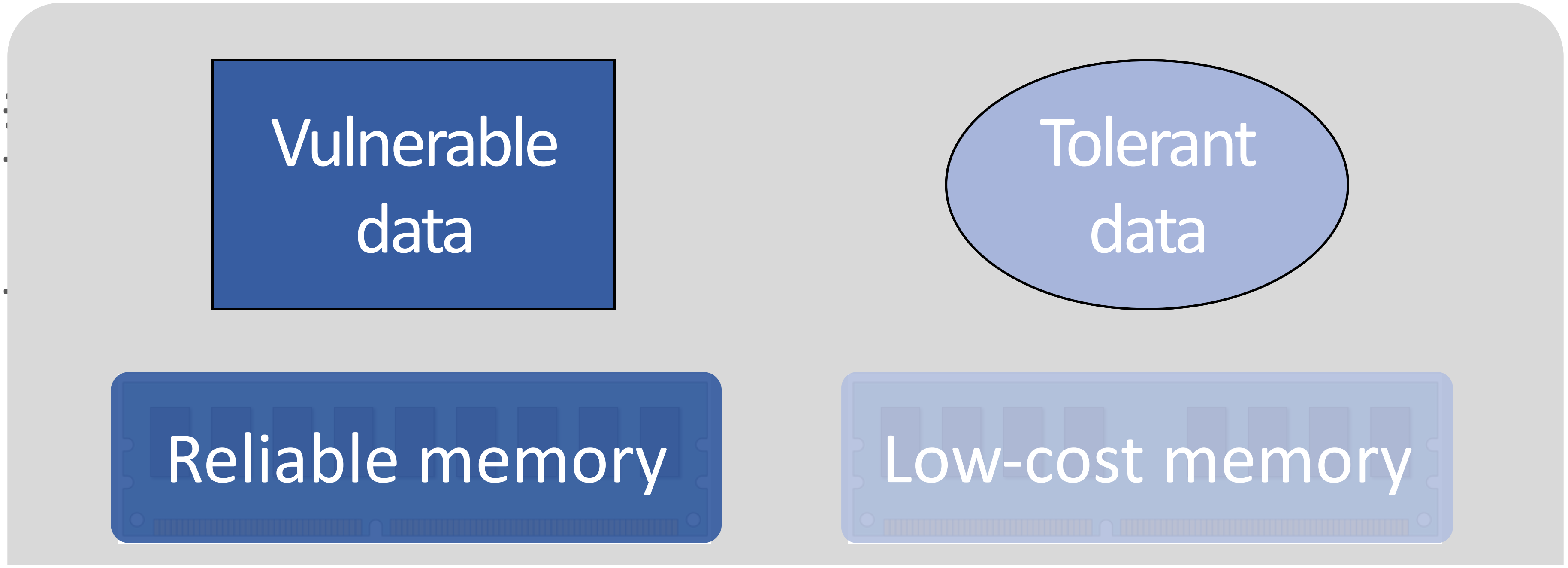
- Yixin Luo, Sriram Govindan, Bikash Sharma, Mark Santaniello, Justin Meza, Aman Kansal, Jie Liu, Badriddine Khessib, Kushagra Vaid, and Onur Mutlu, **"Characterizing Application Memory Error Vulnerability to Optimize Data Center Cost via Heterogeneous-Reliability Memory"** *Proceedings of the 44th Annual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

Another Example: EDEN

- 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. We can reduce DRAM latency and voltage on such data and layers (intermediate feature maps and weights)

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

Data-aware management of DRAM latency and voltage

Challenge and Opportunity for Future

Data-Aware

(Expressive)

Computing Architectures

Recap: Corollaries: Architectures Today

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

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

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

Concluding Remarks

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

- Design complete systems to be truly balanced, high-performance, and energy-efficient → intelligent architectures

- **Data-centric, data-driven, data-aware**

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

Architectures for Intelligent Machines

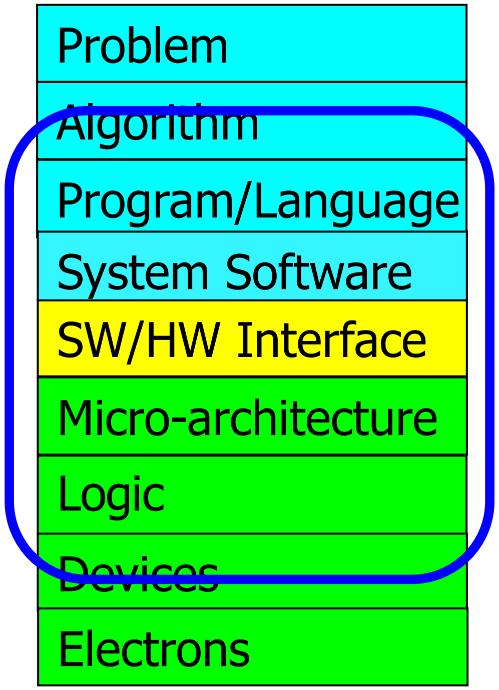
Data-centric

Data-driven

Data-aware

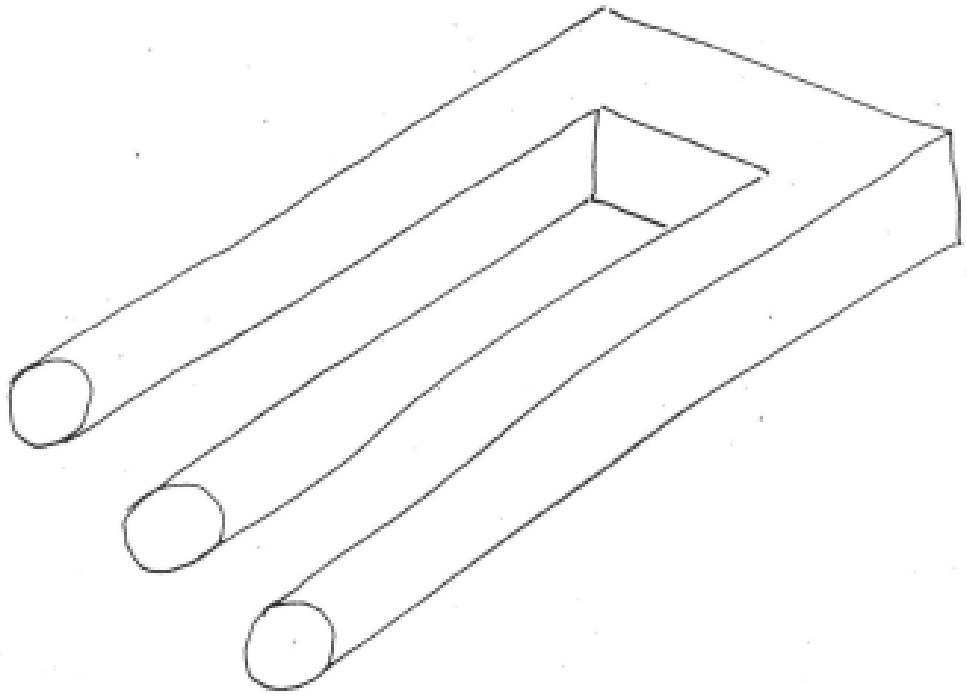
Source: http://spectrum.ieee.org/image/MjYzMzAyMg.jpeg

We Need to Revisit the Entire Stack



We can get there step by step

Finally, people are always telling you: *Think outside the box*



from Yale Patt's EE 382N lecture notes

I prefer: Expand the box

from Yale Patt’s EE 382N lecture notes

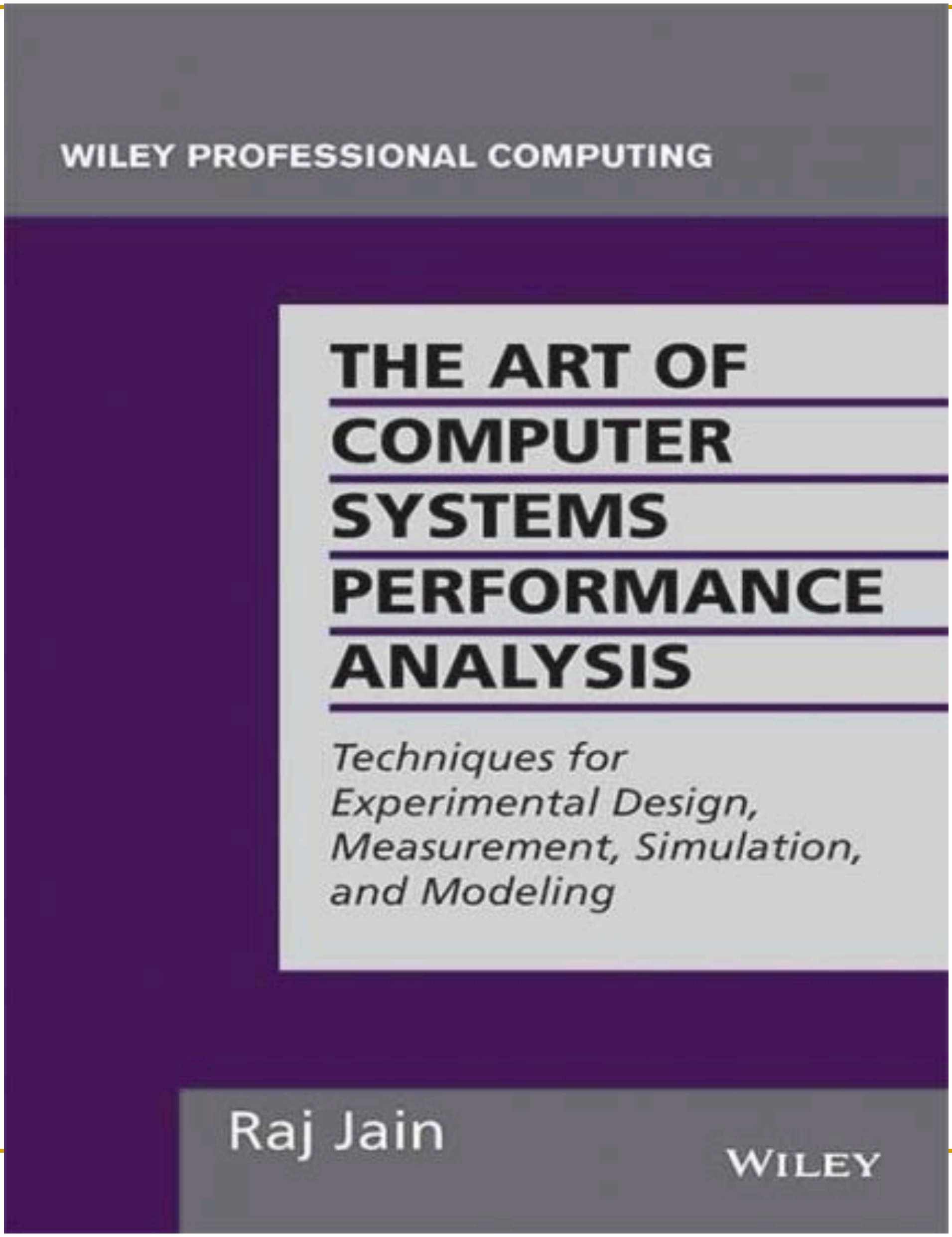
Principled Architectures for Intelligent Machines

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu

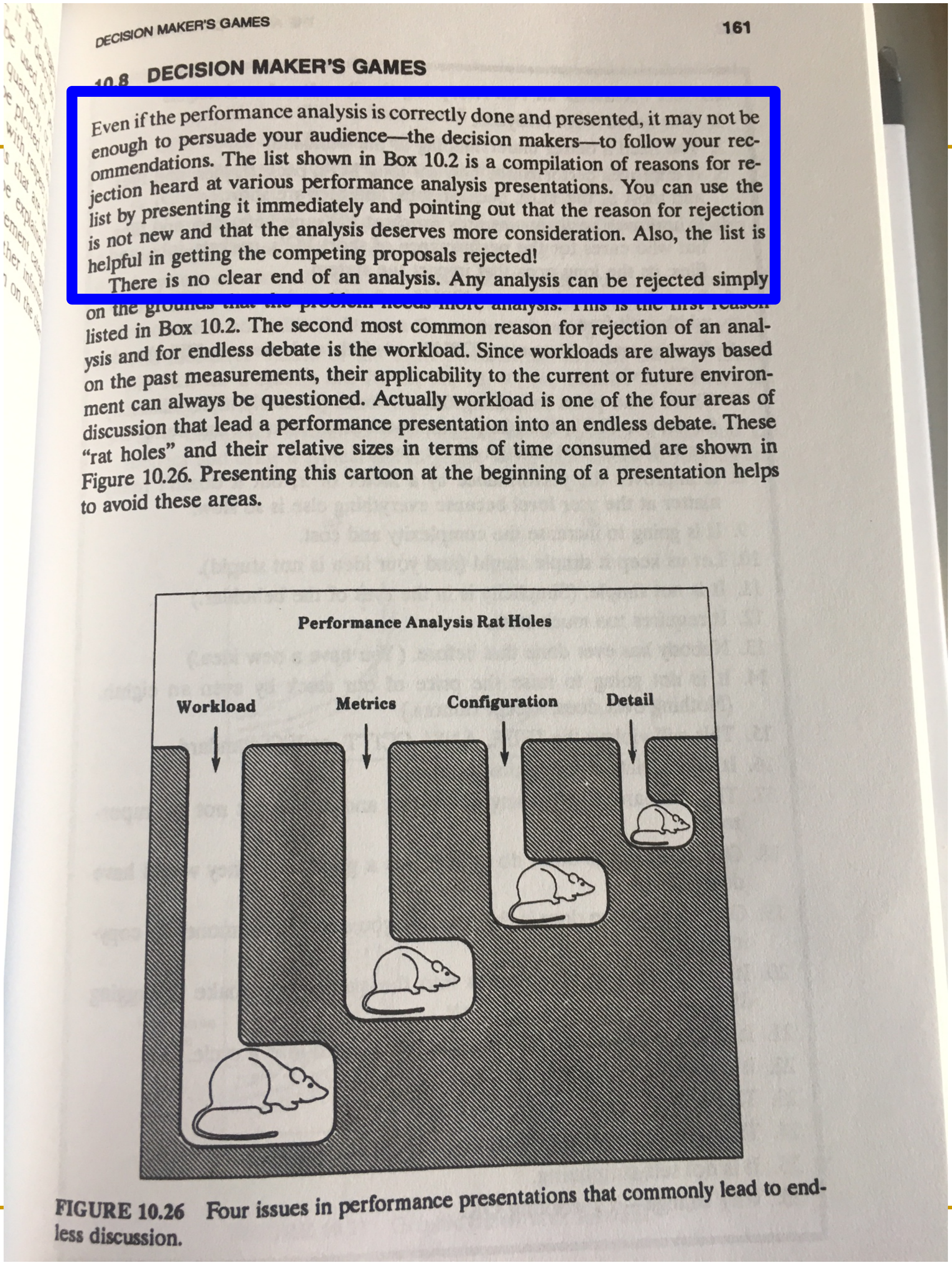
1 July 2019

Yale @ 80

Aside: A Recommended Book


WILEY PROFESSIONAL COMPUTING
THE ART OF COMPUTER SYSTEMS PERFORMANCE ANALYSIS
Techniques for Experimental Design, Measurement, Simulation, and Modeling
Raj Jain
WILEY

Raj Jain, "The Art of Computer Systems Performance Analysis," Wiley, 1991.



10.8 DECISION MAKER'S GAMES

Even if the performance analysis is correctly done and presented, it may not be enough to persuade your audience—the decision makers—to follow your recommendations. The list shown in Box 10.2 is a compilation of reasons for rejection heard at various performance analysis presentations. You can use the list by presenting it immediately and pointing out that the reason for rejection is not new and that the analysis deserves more consideration. Also, the list is helpful in getting the competing proposals rejected!

There is no clear end of an analysis. Any analysis can be rejected simply on the grounds that the problem needs more analysis. This is the first reason listed in Box 10.2. The second most common reason for rejection of an analysis and for endless debate is the workload. Since workloads are always based on the past measurements, their applicability to the current or future environment can always be questioned. Actually workload is one of the four areas of discussion that lead a performance presentation into an endless debate. These "rat holes" and their relative sizes in terms of time consumed are shown in Figure 10.26. Presenting this cartoon at the beginning of a presentation helps to avoid these areas.

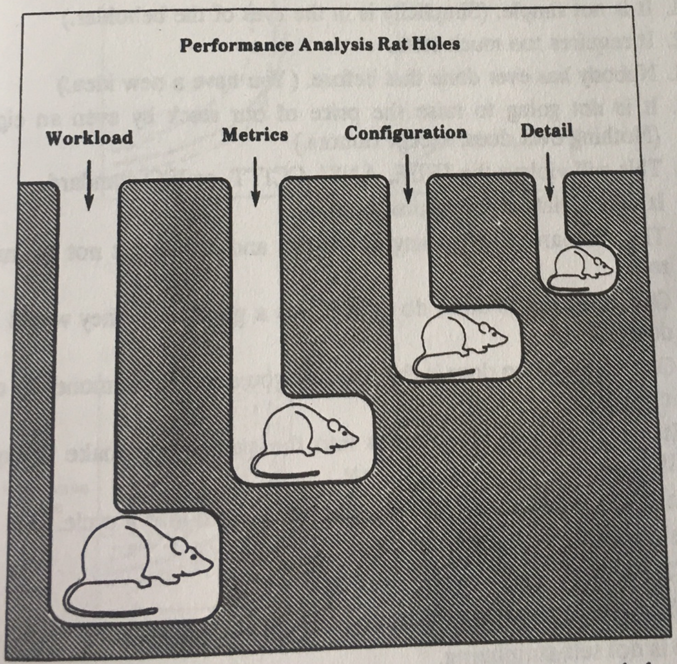
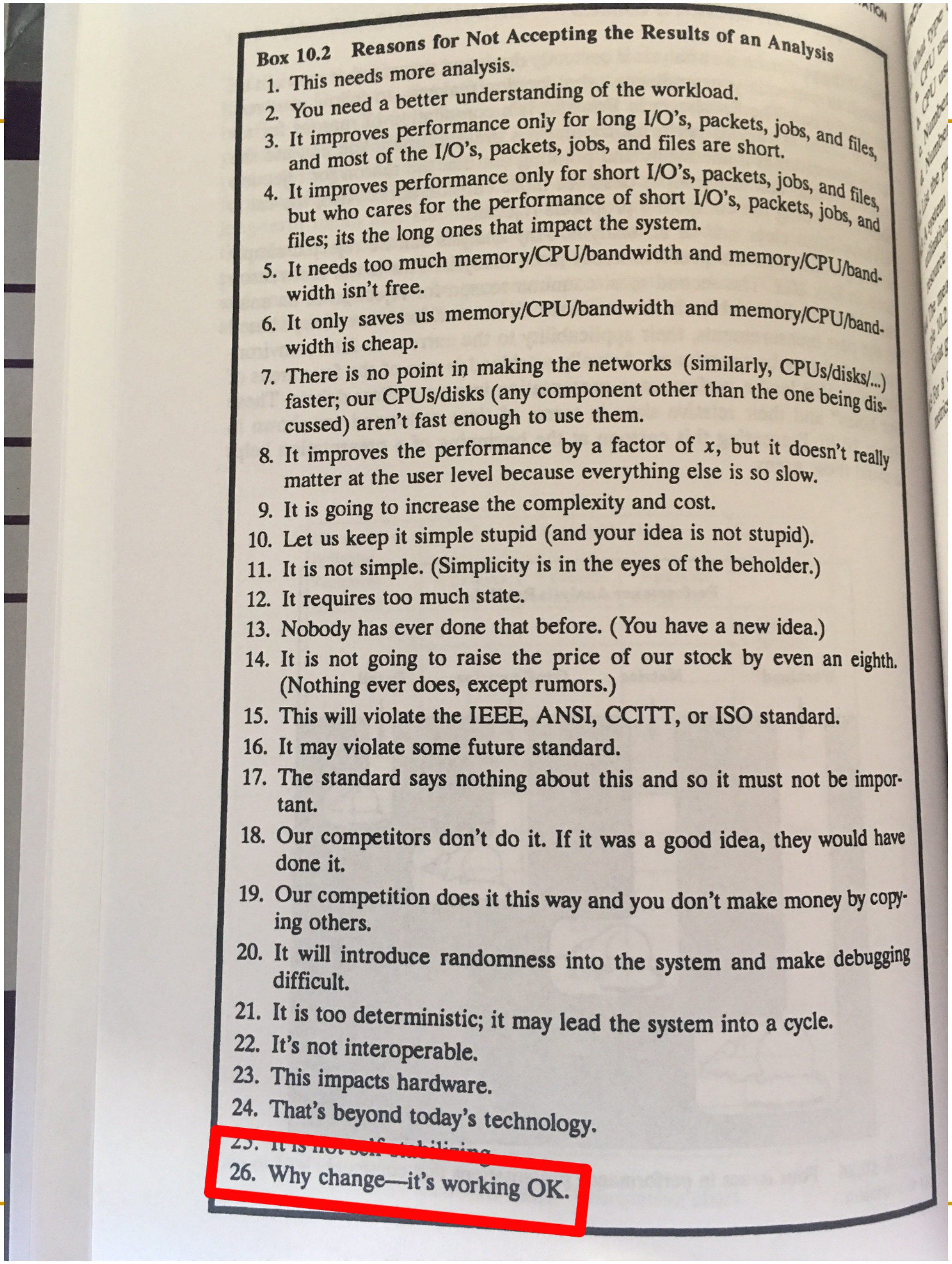


FIGURE 10.26 Four issues in performance presentations that commonly lead to endless discussion.

Raj Jain, "The Art of Computer Systems Performance Analysis," Wiley, 1991.



Box 10.2 Reasons for Not Accepting the Results of an Analysis

1. This needs more analysis.
2. You need a better understanding of the workload.
3. It improves performance only for long I/O's, packets, jobs, and files, and most of the I/O's, packets, jobs, and files are short.
4. It improves performance only for short I/O's, packets, jobs, and files, but who cares for the performance of short I/O's, packets, jobs, and files; its the long ones that impact the system.
5. It needs too much memory/CPU/bandwidth and memory/CPU/bandwidth isn't free.
6. It only saves us memory/CPU/bandwidth and memory/CPU/bandwidth is cheap.
7. There is no point in making the networks (similarly, CPUs/disks/...) faster; our CPUs/disks (any component other than the one being discussed) aren't fast enough to use them.
8. It improves the performance by a factor of x, but it doesn't really matter at the user level because everything else is so slow.
9. It is going to increase the complexity and cost.
10. Let us keep it simple stupid (and your idea is not stupid).
11. It is not simple. (Simplicity is in the eyes of the beholder.)
12. It requires too much state.
13. Nobody has ever done that before. (You have a new idea.)
14. It is not going to raise the price of our stock by even an eighth. (Nothing ever does, except rumors.)
15. This will violate the IEEE, ANSI, CCITT, or ISO standard.
16. It may violate some future standard.
17. The standard says nothing about this and so it must not be important.
18. Our competitors don't do it. If it was a good idea, they would have done it.
19. Our competition does it this way and you don't make money by copying others.
20. It will introduce randomness into the system and make debugging difficult.
21. It is too deterministic; it may lead the system into a cycle.
22. It's not interoperable.
23. This impacts hardware.
24. That's beyond today's technology.
25. It is not self-stabilizing.
26. Why change—it's working OK.

Raj Jain, "The Art of Computer Systems Performance Analysis," Wiley, 1991.

Low Latency Data Access

Data-Centric Architectures: Properties

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

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

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

- **Intelligent data management**
  - Intelligent controllers handling robustness, security, cost, scaling

Low-Latency & Low-Energy Data Access

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

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

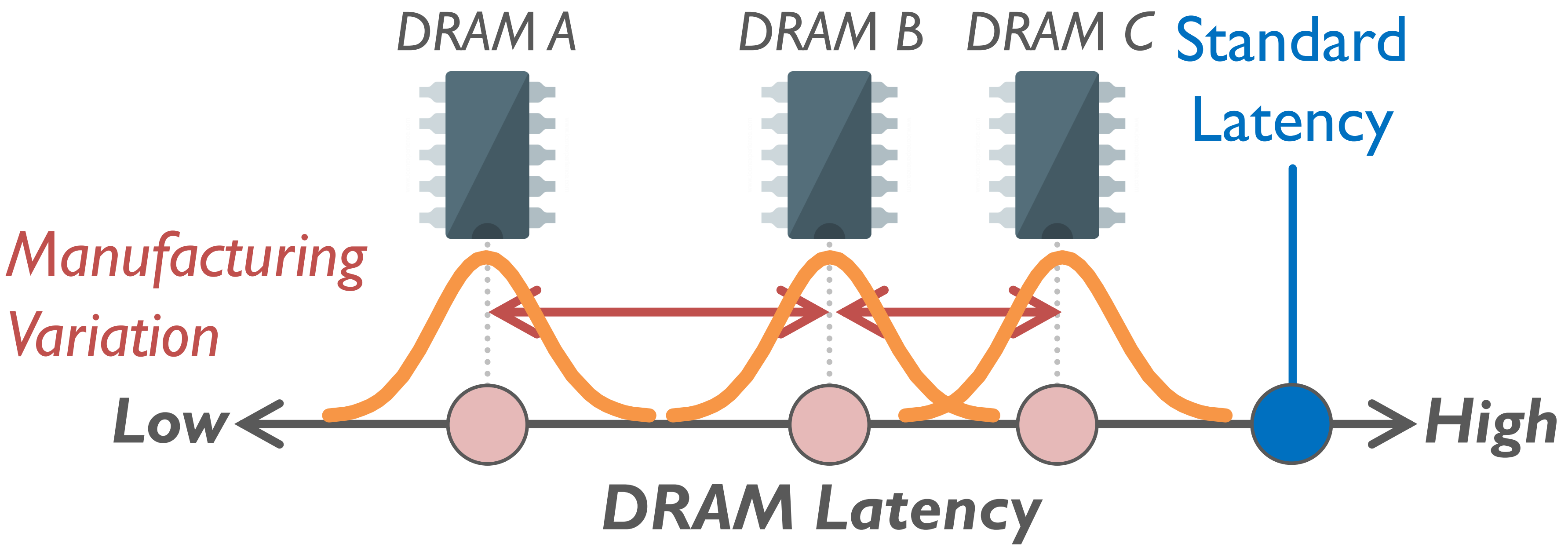
- Modern DRAM is **not** designed for low latency
  - Main focus is cost-per-bit (capacity)

- 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 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

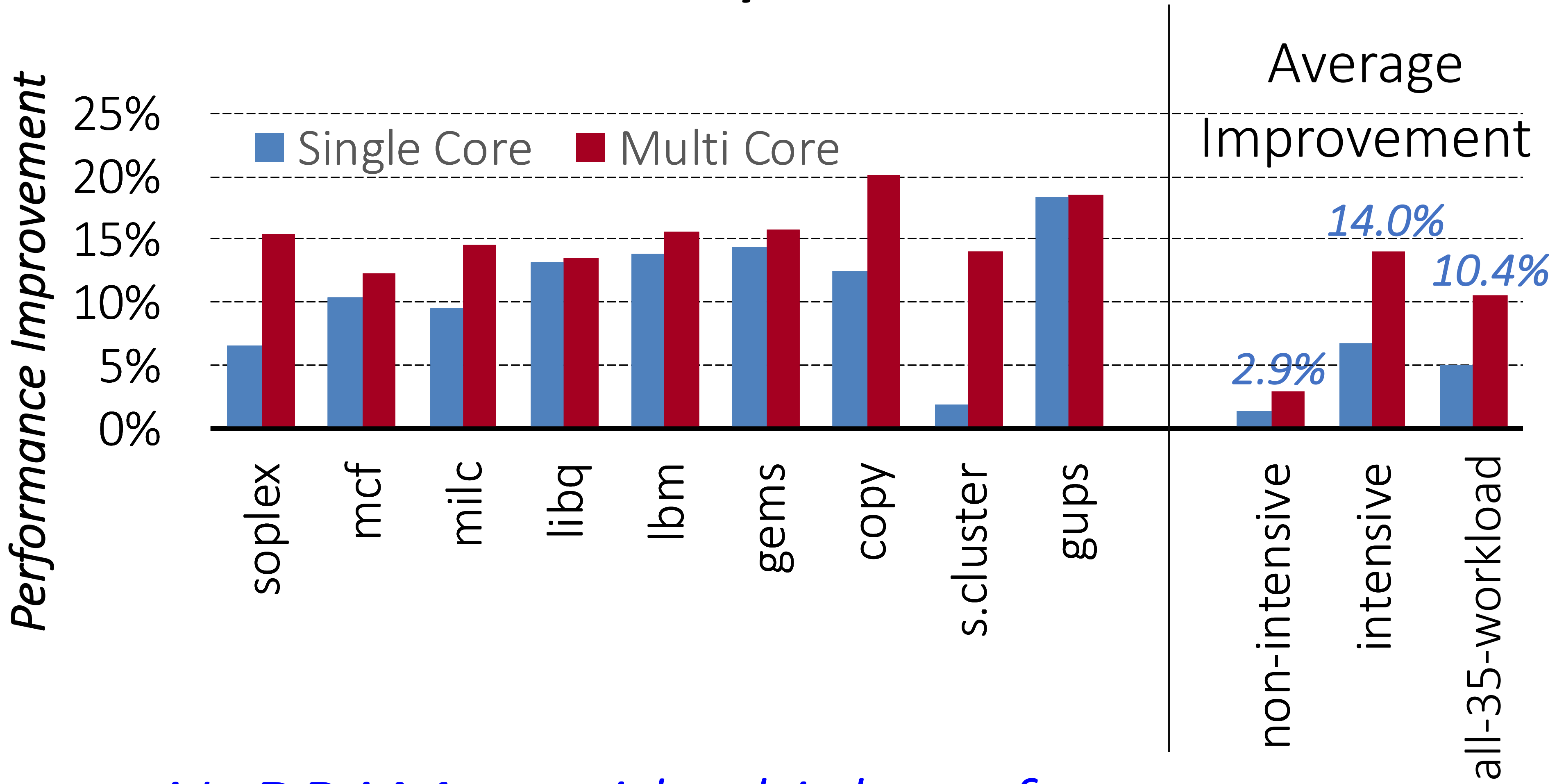
- *Key idea*
  - Optimize DRAM timing parameters online
- *Two components*
  - DRAM manufacturer provides multiple sets of reliable DRAM timing parameters at different temperatures for each DIMM
  - System monitors DRAM temperature & uses appropriate DRAM timing parameters

Lee+, "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case," HPCA 2015.

Latency Reduction Summary of 115 DIMMs

- *Latency reduction for read & write (55°C)*
  - *Read Latency: **32.7%***
  - *Write Latency: **55.1%***
- *Latency reduction for each timing parameter (55°C)*
  - *Sensing: **17.3%***
  - *Restore: **37.3%** (read), **54.8%** (write)*
  - *Precharge: **35.2%***

AL-DRAM: Real-System Performance



AL-DRAM provides high performance on ***memory-intensive workloads***

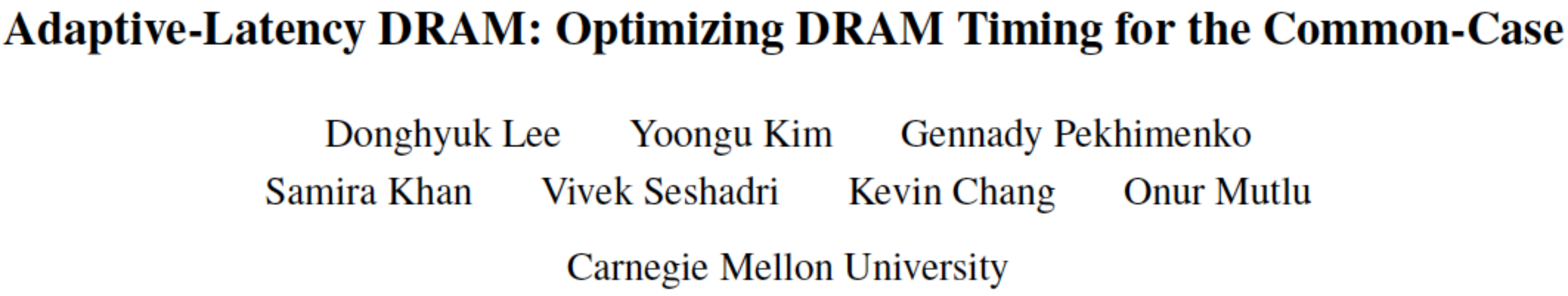
SAFARI

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

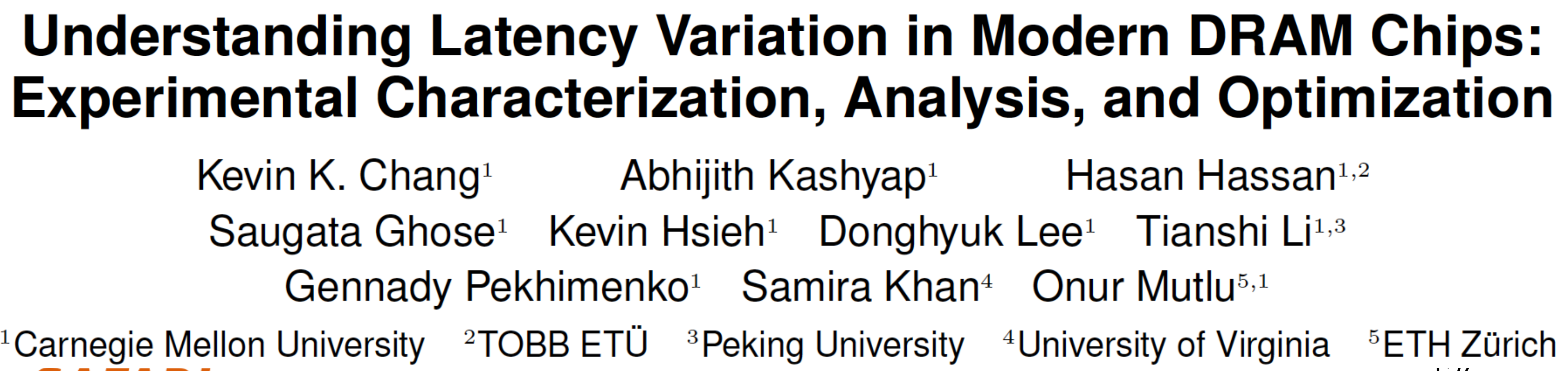
- Reliable operation latency is actually very heterogeneous
  - Across temperatures, chips, parts of a chip, voltage levels, …

- 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]
  - Solar DRAM [ICCD 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]

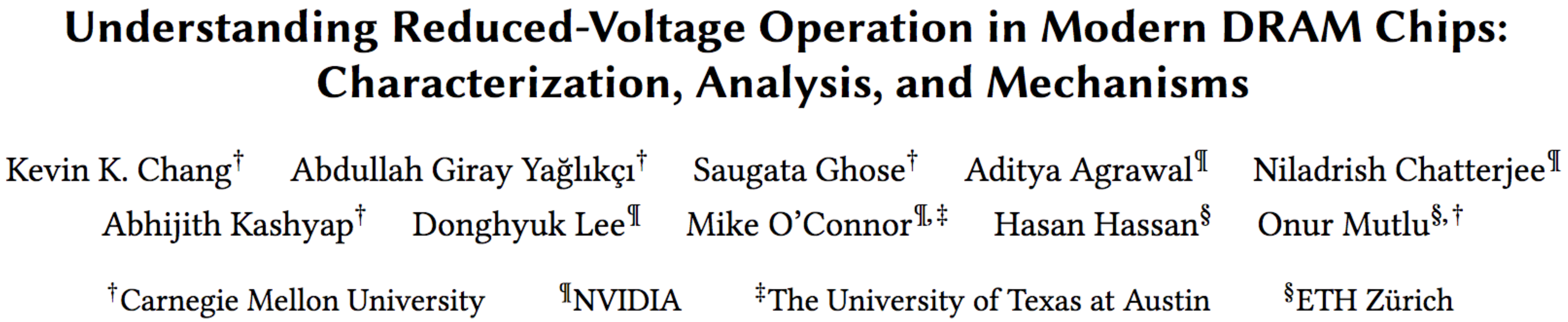


Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization

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

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

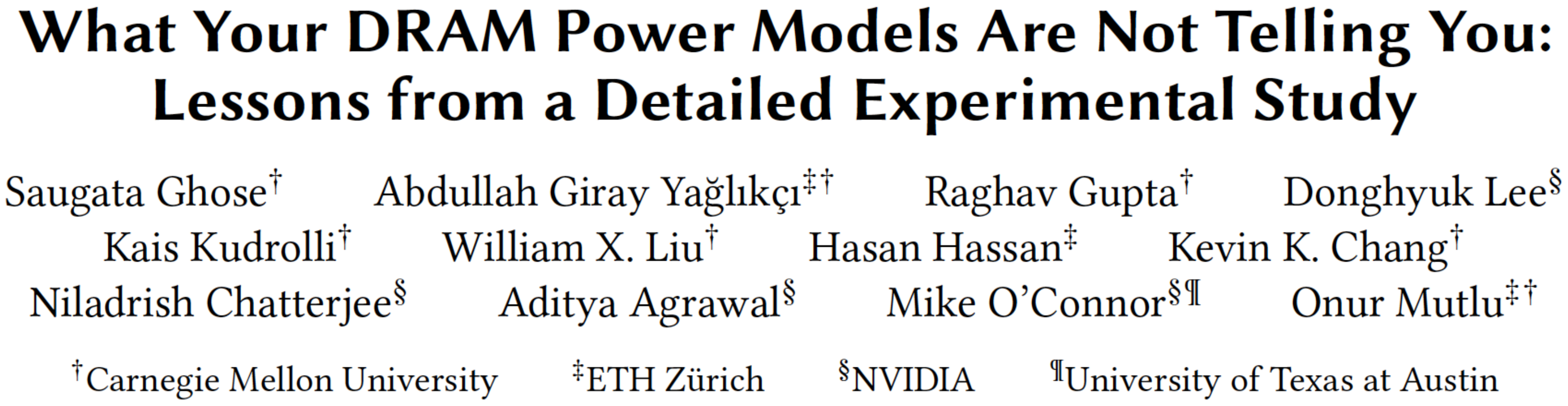
Kevin K. Chang[†] Abdullah Giray Yağlıkçı[†] Saugata Ghose[†] Aditya Agrawal[¶] Niladrish Chatterjee[¶]
Abhijith Kashyap[†] Donghyuk Lee[¶] Mike O'Connor[¶,‡] Hasan Hassan[§] Onur Mutlu[§,†]

[†]Carnegie Mellon University [¶]NVIDIA [‡]The University of Texas at Austin [§]ETH Zürich

SAFARI

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]



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

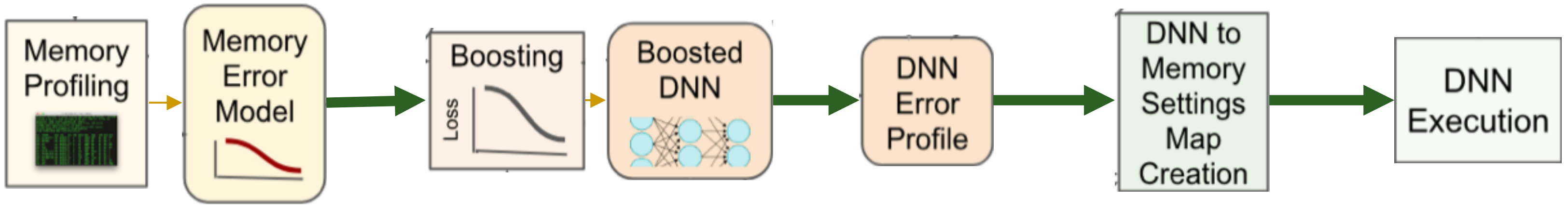
Saugata Ghose[†] Abdullah Giray Yağlıkçı[‡†] Raghav Gupta[†] Donghyuk Lee[§]
Kais Kudrolli[†] William X. Liu[†] Hasan Hassan[‡] Kevin K. Chang[†]
Niladrish Chatterjee[§] Aditya Agrawal[§] Mike O'Connor[§¶] Onur Mutlu[‡†]

[†]Carnegie Mellon University [‡]ETH Zürich [§]NVIDIA [¶]University of Texas at Austin

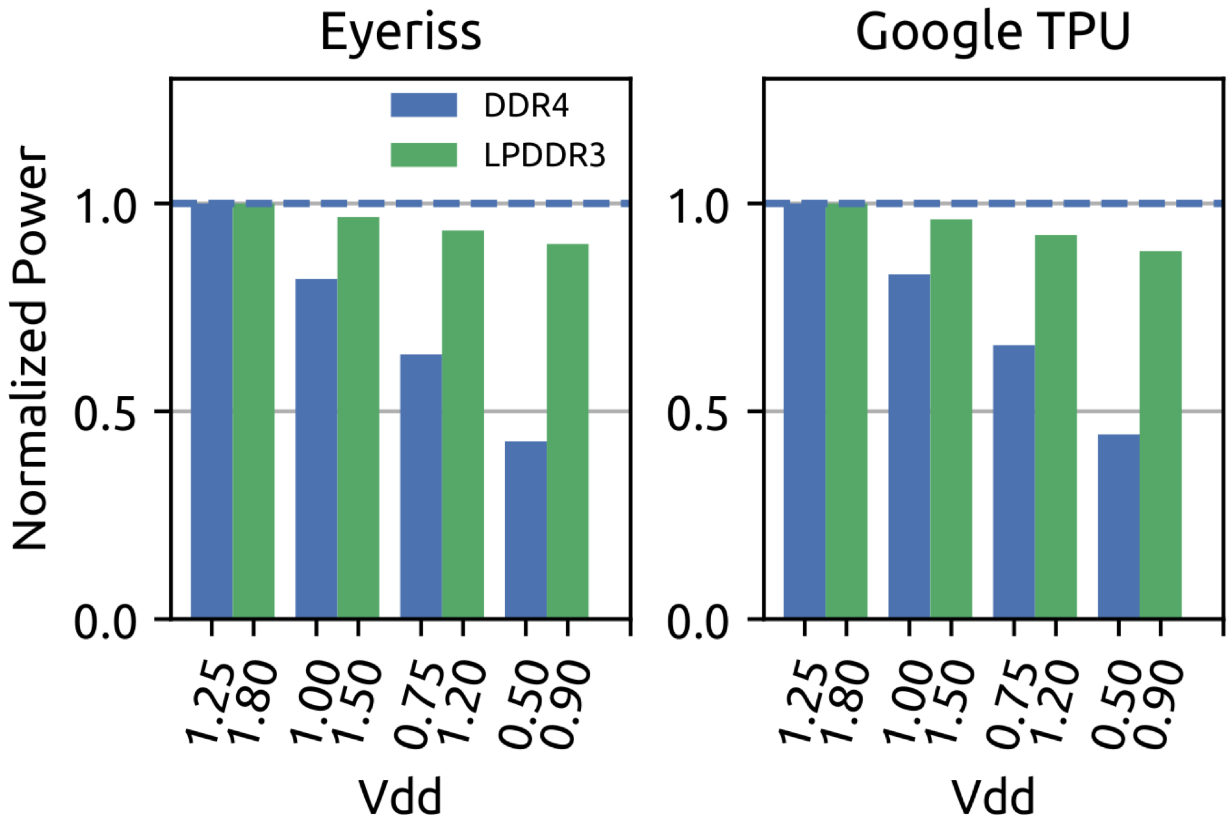
EDEN

EDEN Flow

- Goal: reduce energy consumption and improve performance of DNNs by reducing voltage and timing parameters in DRAM
- Key Ideas:
  - Build an error model by profiling the target DRAM module with reduced voltage and timing
  - Boost DNN accuracy by introducing the error model in training
  - Profile the boosted DNN to understand the network's error tolerance
  - Map DNN components to DRAM partitions and DRAM settings



EDEN Power, Performance, Accuracy



- ~15-20% power savings, 8% perf improvement, <1% accuracy loss

Other Backup Slides

Readings, Videos, Reference Materials

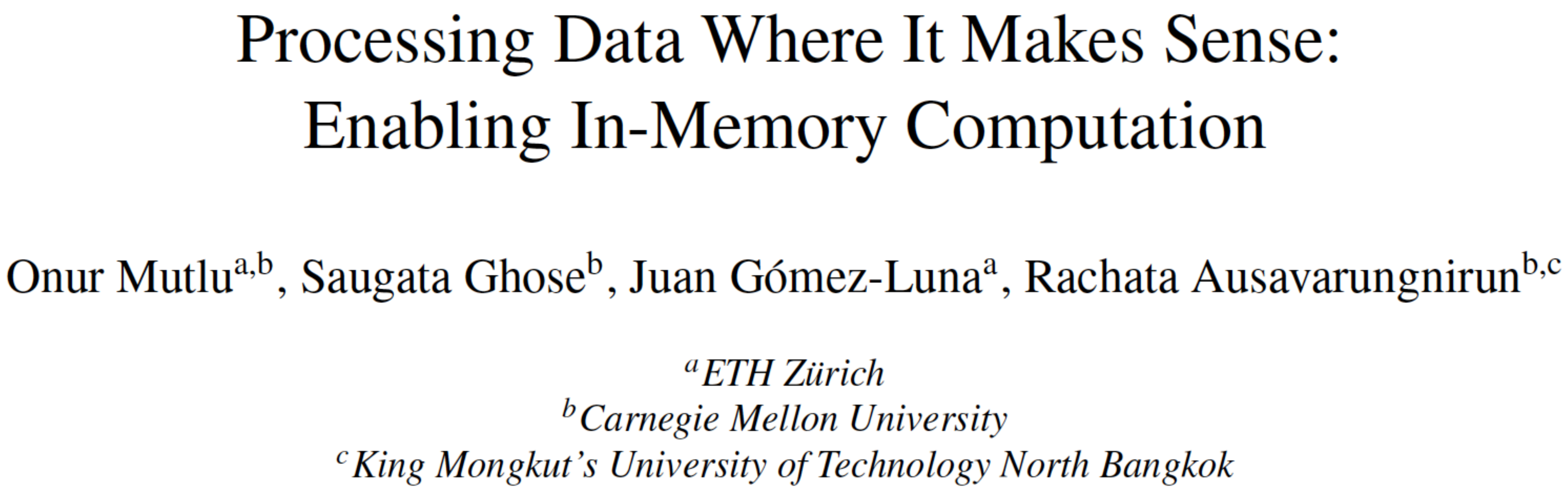
Accelerated Memory Course (~6.5 hours)

- 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)

Reference Overview Paper I



Processing Data Where It Makes Sense: Enabling In-Memory Computation

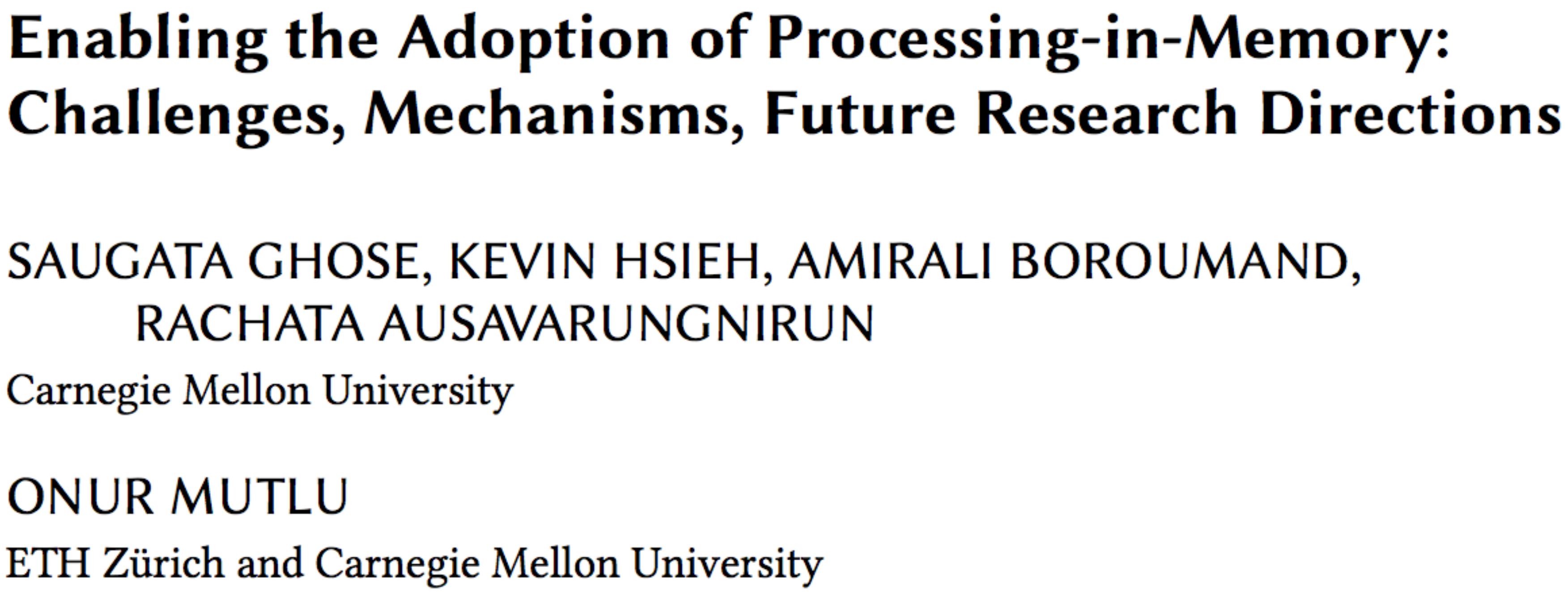
Onur Mutlu[a,b], Saugata Ghose[b], Juan Gómez-Luna[a], Rachata Ausavarungnirun[b,c]

[a] *ETH Zürich*
[b] *Carnegie Mellon University*
[c] *King Mongkut's University of Technology North Bangkok*

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]

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

Reference Overview Paper II



Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions

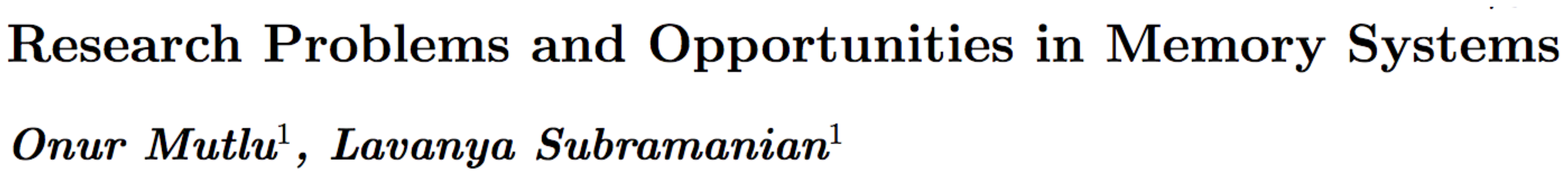
SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND, RACHATA AUSAVARUNGNIRUN
Carnegie Mellon University

ONUR MUTLU
ETH Zürich and Carnegie Mellon University

Saugata Ghose, Kevin Hsieh, Amirali Boroumand, Rachata Ausavarungnirun, Onur Mutlu, **"Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions"**
Invited Book Chapter, to appear in 2018.
[Preliminary arxiv.org version]

Reference Overview Paper III

- Onur Mutlu and Lavanya Subramanian,
"Research Problems and Opportunities in Memory Systems"
*Invited Article in Supercomputing Frontiers and Innovations (**SUPERFRI**)*, 2014/2015.

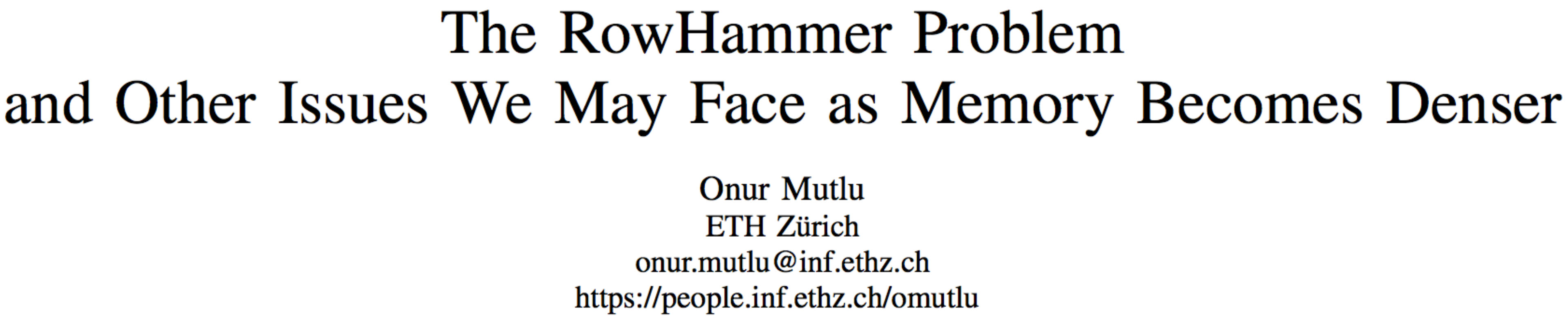

Research Problems and Opportunities in Memory Systems

Onur Mutlu[1], *Lavanya Subramanian*[1]

https://people.inf.ethz.ch/omutlu/pub/memory-systems-research_superfri14.pdf

Reference Overview Paper IV

- 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)]

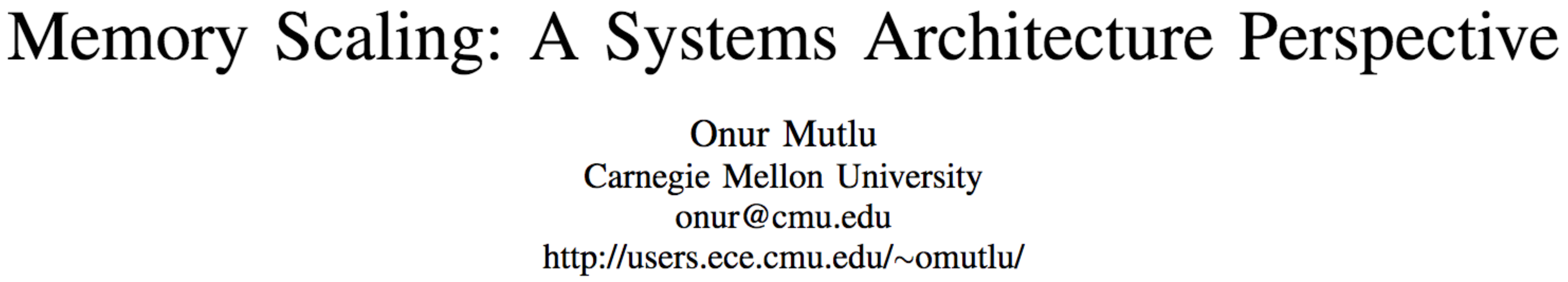


The RowHammer Problem
and Other Issues We May Face as Memory Becomes Denser

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

Reference Overview Paper V

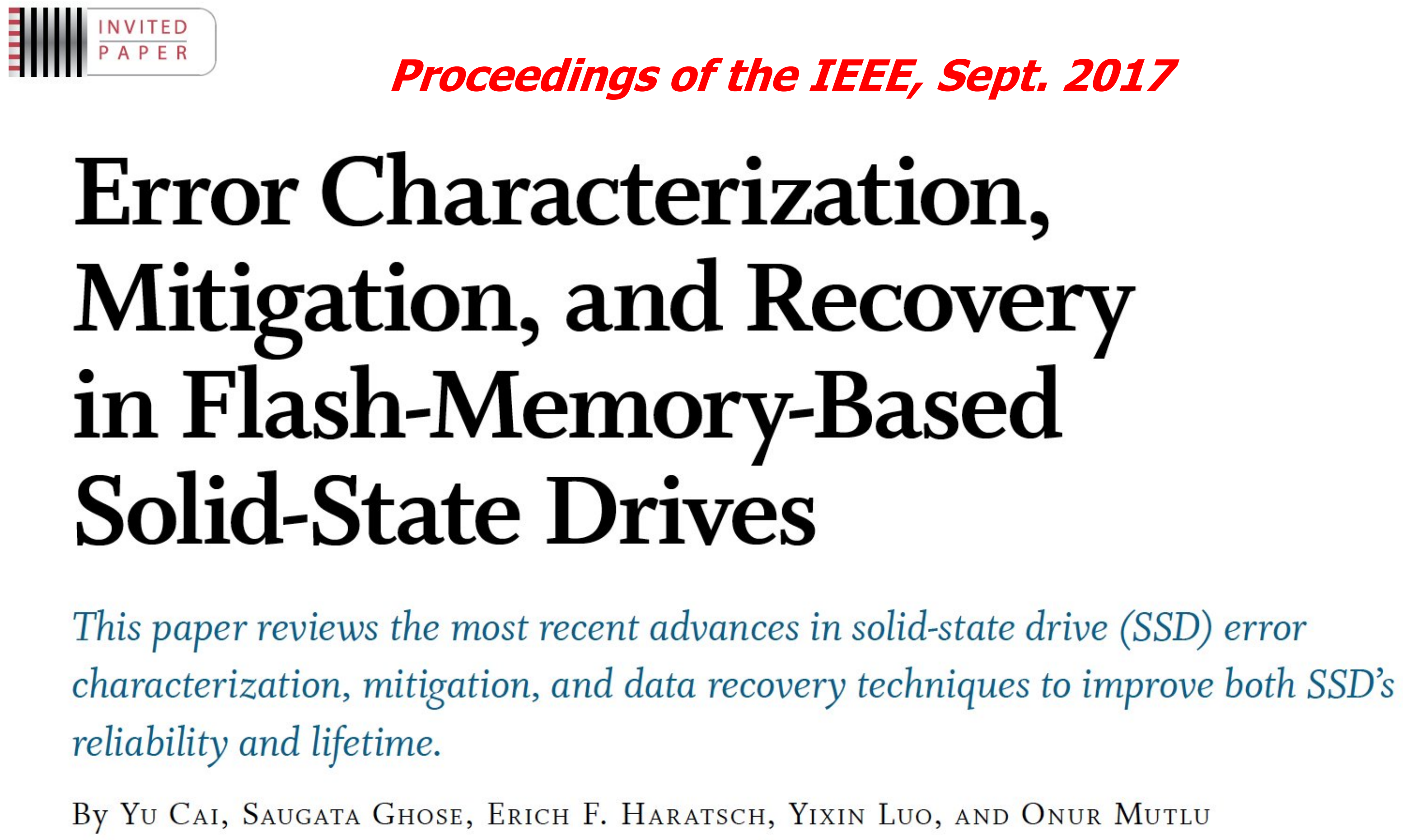
- Onur Mutlu,
  "Memory Scaling: A Systems Architecture Perspective"
  *Technical talk at MemCon 2013 (**MEMCON**)*, Santa Clara, CA, August 2013. [Slides (pptx) (pdf)]
  [Video] [Coverage on StorageSearch]



Memory Scaling: A Systems Architecture Perspective

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

Reference Overview Paper VI



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

Related Videos and Course Materials (I)

- Undergraduate Computer Architecture Course Lecture Videos (2015, 2014, 2013)
- Undergraduate Computer Architecture Course Materials (2015, 2014, 2013)

- Graduate Computer Architecture Course Lecture Videos (2018, 2017, 2015, 2013)
- Graduate Computer Architecture Course Materials (2018, 2017, 2015, 2013)

- Parallel Computer Architecture Course Materials (Lecture Videos)

Related Videos and Course Materials (II)

- **Freshman Digital Circuits and Computer Architecture Course Lecture Videos (2018, 2017)**
- **Freshman Digital Circuits and Computer Architecture Course Materials (2018)**

- **Memory Systems Short Course Materials**

  (Lecture Video on Main Memory and DRAM Basics)

Some Open Source Tools (I)

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

- Other open-source software from my group
  - **https://github.com/CMU-SAFARI/**
  - **http://www.ece.cmu.edu/~safari/tools.html**

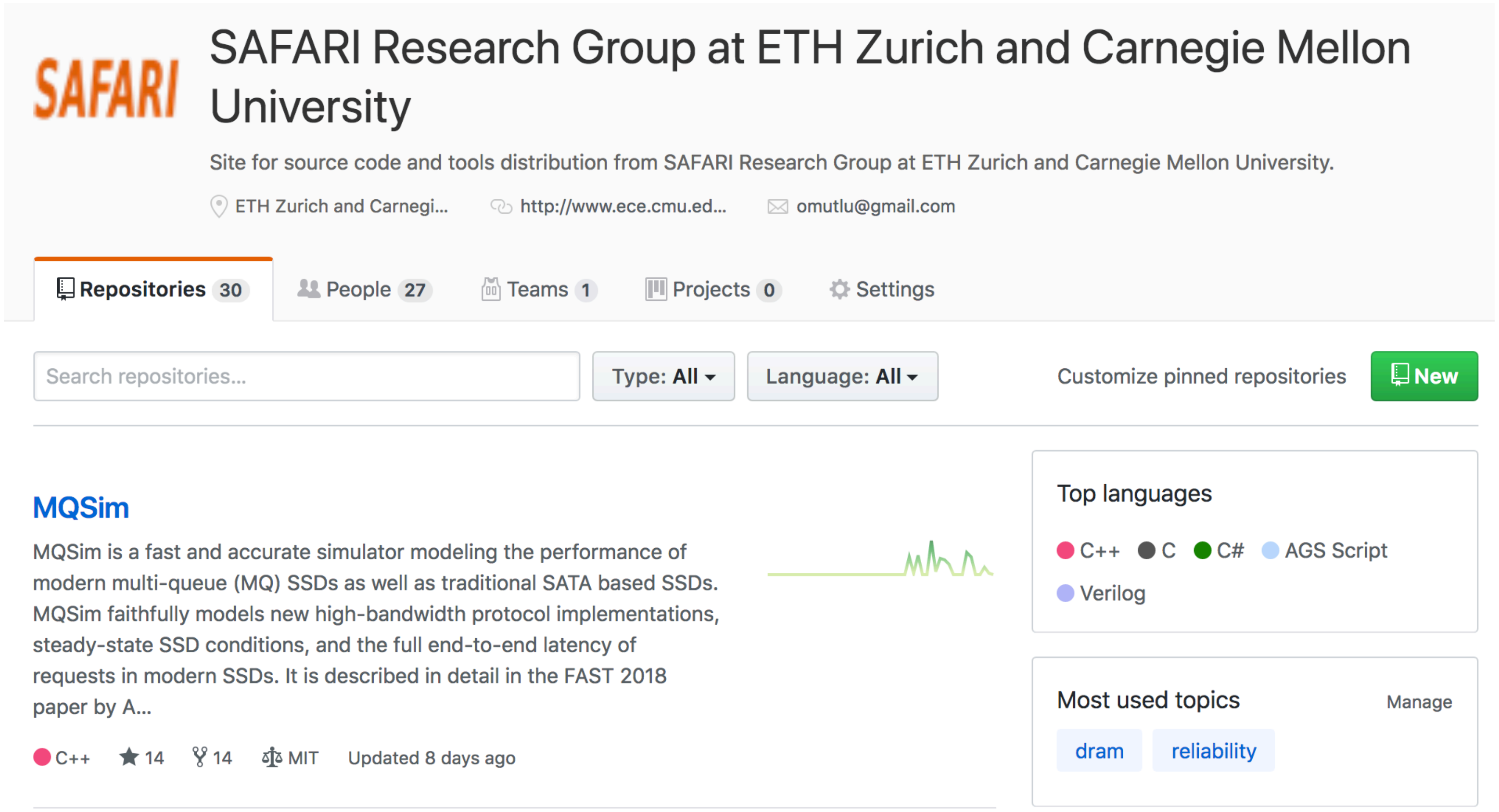
Some Open Source Tools (II)

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

- Other open-source software from my group
  - **https://github.com/CMU-SAFARI/**
  - **http://www.ece.cmu.edu/~safari/tools.html**

More Open Source Tools (III)

- A lot more open-source software from my group
  - https://github.com/CMU-SAFARI/
  - http://www.ece.cmu.edu/~safari/tools.html



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 Carnegi... http://www.ece.cmu.ed... omutlu@gmail.com

Repositories 30 | People 27 | Teams 1 | Projects 0 | Settings

Search repositories... | Type: All | Language: All | Customize pinned repositories | New

MQSim

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

● C++ ★ 14 14 MIT Updated 8 days ago

Top languages

● C++ ● C ● C# ● AGS Script ● Verilog

Most used topics Manage

dram reliability

Referenced Papers

- All are available at

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

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

  https://people.inf.ethz.ch/omutlu/acaces2018.html