Computer Architecture

Lecture 17a: Emerging Memory Technologies II

Prof. Onur Mutlu

ETH Zürich

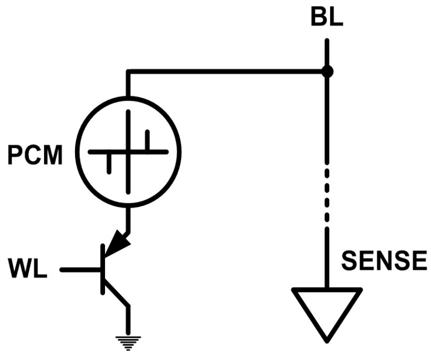
Fall 2021

25 November 2021

Solution 2: Emerging Memory Technologies

- Some emerging **resistive** memory technologies seem more scalable than DRAM (and they are non-volatile)

- Example: Phase Change Memory
  - Data stored by changing phase of material
  - Data read by detecting material’s resistance
  - Expected to scale to 9nm (2022 [ITRS 2009])
  - Prototyped at 20nm (Raoux+, IBM JRD 2008)
  - Expected to be denser than DRAM: can store multiple bits/cell



- But, emerging technologies have (many) shortcomings
  - Can they be enabled to replace/augment/surpass DRAM?

Solution 2: Emerging Memory Technologies

- Lee+, "Architecting Phase Change Memory as a Scalable DRAM Alternative," ISCA'09, CACM'10, IEEE Micro'10.
- Meza+, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters 2012.
- Yoon, Meza+, "Row Buffer Locality Aware Caching Policies for Hybrid Memories," ICCD 2012.
- Kultursay+, "Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative," ISPASS 2013.
- Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.
- Lu+, "Loose Ordering Consistency for Persistent Memory," ICCD 2014.
- Zhao+, "FIRM: Fair and High-Performance Memory Control for Persistent Memory Systems," MICRO 2014.
- Yoon, Meza+, "Efficient Data Mapping and Buffering Techniques for Multi-Level Cell Phase-Change Memories," TACO 2014.
- Ren+, "ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems," MICRO 2015.
- Chauhan+, "NVMove: Helping Programmers Move to Byte-Based Persistence," INFLOW 2016.
- Li+, "Utility-Based Hybrid Memory Management," CLUSTER 2017.
- Yu+, "Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation," MICRO 2017.
- Tavakkol+, "MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices," FAST 2018.
- Tavakkol+, "FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives," ISCA 2018.
- Sadrosadati+. "LTRF: Enabling High-Capacity Register Files for GPUs via Hardware/Software Cooperative Register Prefetching," ASPLOS 2018.
- Salkhordeh+, "An Analytical Model for Performance and Lifetime Estimation of Hybrid DRAM-NVM Main Memories," TC 2019.
- Wang+, "Panthera: Holistic Memory Management for Big Data Processing over Hybrid Memories," PLDI 2019.
- Song+, "Enabling and Exploiting Partition-Level Parallelism (PALP) in Phase Change Memories," CASES 2019.
- Liu+, "Binary Star: Coordinated Reliability in Heterogeneous Memory Systems for High Performance and Scalability," MICRO'19.
- Song+, "Improving Phase Change Memory Performance with Data Content Aware Access," ISMM 2020.
- Yavits+, "WoLFRaM: Enhancing Wear-Leveling and Fault Tolerance in Resistive Memories using Programmable Address Decoders," ICCD 2020.
- Song+, "Aging-Aware Request Scheduling for Non-Volatile Main Memory," ASP-DAC 2021.

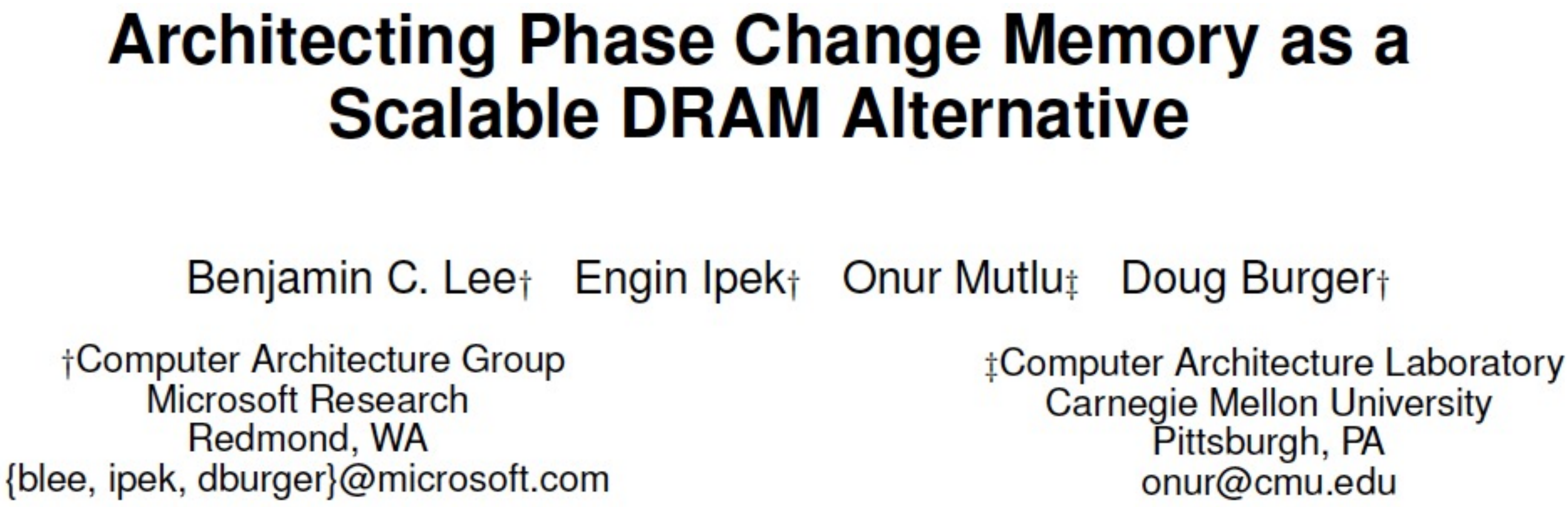
Intel Optane Persistent Memory (2019)

- Non-volatile main memory
- Based on 3D-XPoint Technology

https://www.storagereview.com/intel_optane_dc_persistent_memory_module_pmm

PCM as Main Memory: Idea in 2009

- Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger, **"Architecting Phase Change Memory as a Scalable DRAM Alternative"** *Proceedings of the 36th International Symposium on Computer Architecture* (***ISCA***), pages 2-13, Austin, TX, June 2009. Slides (pdf)
One of the 13 computer architecture papers of 2009 selected as Top Picks by IEEE Micro.
Selected as a CACM Research Highlight.



Architecting Phase Change Memory as a Scalable DRAM Alternative

Benjamin C. Lee† Engin Ipek† Onur Mutlu‡ Doug Burger†

†Computer Architecture Group
Microsoft Research
Redmond, WA
{blee, ipek, dburger}@microsoft.com

‡Computer Architecture Laboratory
Carnegie Mellon University
Pittsburgh, PA
onur@cmu.edu

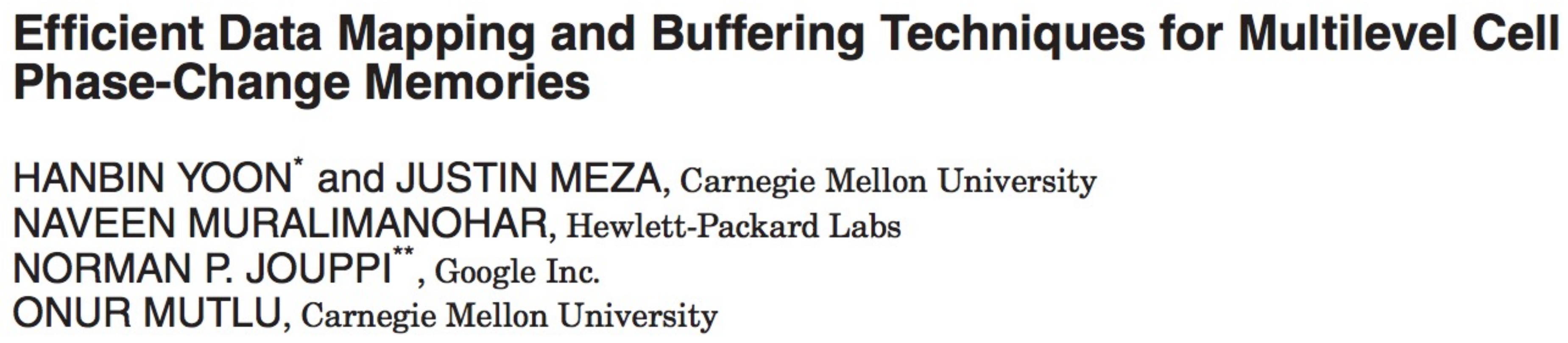
PCM as Main Memory: Idea in 2009

- Benjamin C. Lee, Ping Zhou, Jun Yang, Youtao Zhang, Bo Zhao, Engin Ipek, Onur Mutlu, and Doug Burger, **"Phase Change Technology and the Future of Main Memory"** *IEEE Micro, Special Issue: Micro's Top Picks from 2009 Computer Architecture Conferences (**MICRO TOP PICKS**)*, Vol. 30, No. 1, pages 60-70, January/February 2010.

PHASE-CHANGE TECHNOLOGY AND THE FUTURE OF MAIN MEMORY

More on PCM Based Main Memory

HanBin Yoon, Justin Meza, Naveen Muralimanohar, Norman P. Jouppi, and Onur Mutlu, **"Efficient Data Mapping and Buffering Techniques for Multi-Level Cell Phase-Change Memories"**
ACM Transactions on Architecture and Code Optimization (***TACO***), Vol. 11, No. 4, December 2014. [Slides (ppt) (pdf)]
Presented at the 10th HiPEAC Conference, Amsterdam, Netherlands, January 2015.
[Slides (ppt) (pdf)]
Best (student) presentation award.



Efficient Data Mapping and Buffering Techniques for Multilevel Cell Phase-Change Memories

HANBIN YOON[*] and JUSTIN MEZA, Carnegie Mellon University
NAVEEN MURALIMANOHAR, Hewlett-Packard Labs
NORMAN P. JOUPPI[**], Google Inc.
ONUR MUTLU, Carnegie Mellon University

More on STT-MRAM as Main Memory

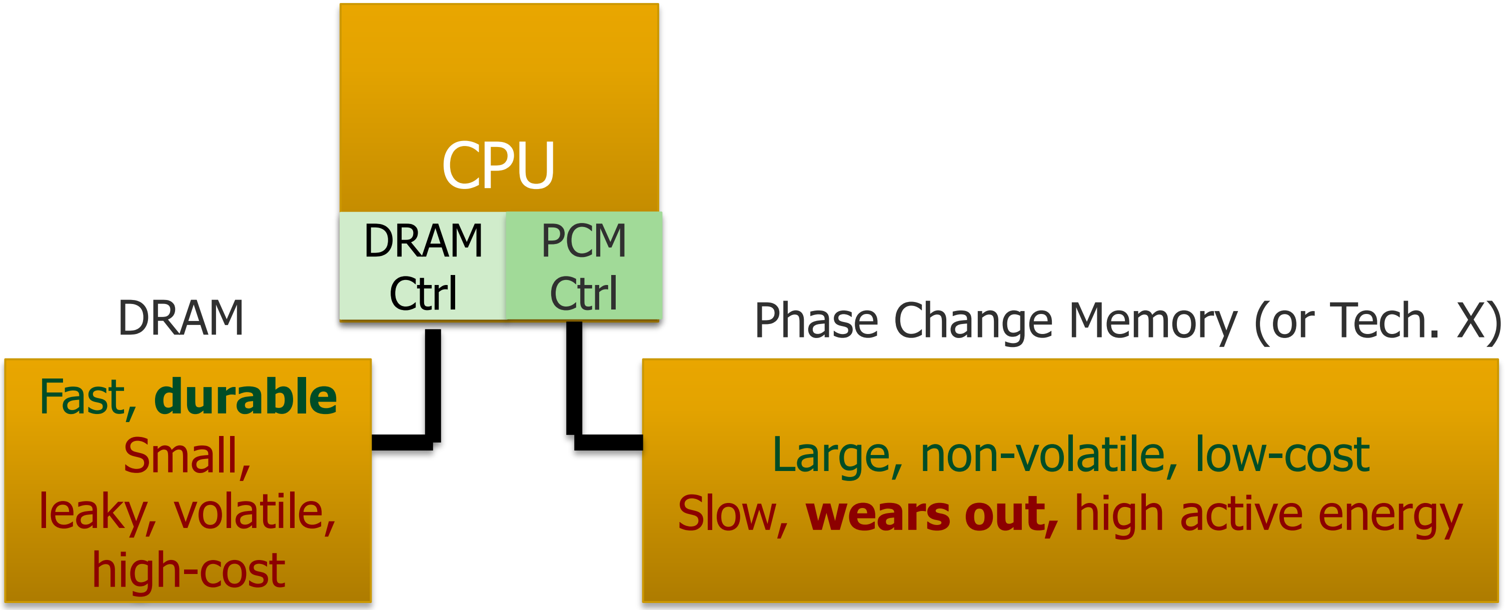
- Emre Kultursay, Mahmut Kandemir, Anand Sivasubramaniam, and Onur Mutlu, **"Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative"** *Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software* (***ISPASS***), Austin, TX, April 2013. Slides (pptx) (pdf)

Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative

Emre Kültürsay[*], Mahmut Kandemir[*], Anand Sivasubramaniam[*], and Onur Mutlu[†]
[*]The Pennsylvania State University and [†]Carnegie Mellon University

Hybrid Main Memory

A More Viable Approach: Hybrid Memory Systems


CPU
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.

Challenge and Opportunity

Providing the Best of
Multiple Metrics
with
Multiple Memory Technologies

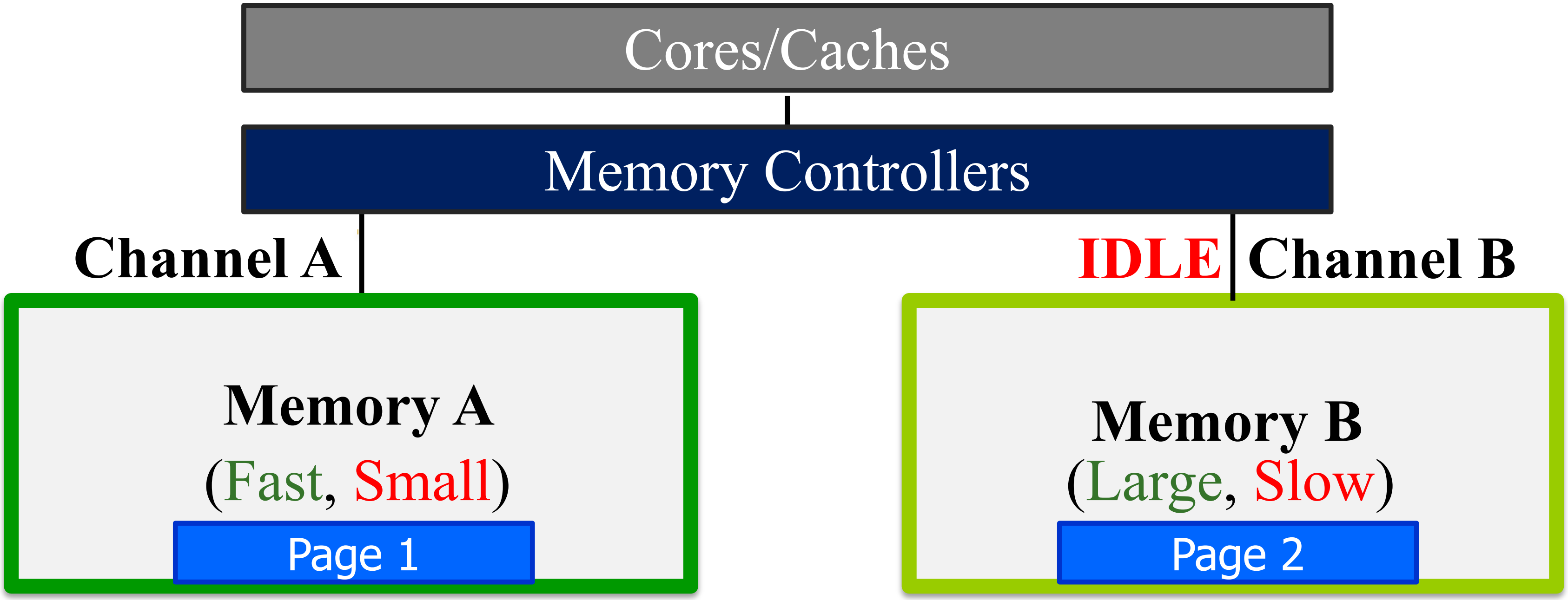
Challenge and Opportunity

Heterogeneous,
Configurable,
Programmable
Memory Systems

Hybrid Memory Systems: Issues

- Cache vs. Main Memory
- Granularity of Data Move/Manage-ment: Fine or Coarse
- Hardware vs. Software vs. HW/SW Cooperative
- When to migrate data?
- How to design a scalable and efficient large cache?
- …

Data Placement in Hybrid Memory



Which memory do we place each page in,
to **maximize system performance**?

- Memory A is fast, but small
- Load should be balanced on both channels?
- Page migrations have performance and energy overhead

Data Placement Between DRAM and PCM

- Idea: Characterize data access patterns and guide data placement in hybrid memory

- Streaming accesses: As fast in PCM as in DRAM

- Random accesses: Much faster in DRAM

- Idea: Place random access data with some reuse in DRAM; streaming data in PCM

- Yoon+, "Row Buffer Locality-Aware Data Placement in Hybrid Memories," ICCD 2012 Best Paper Award.

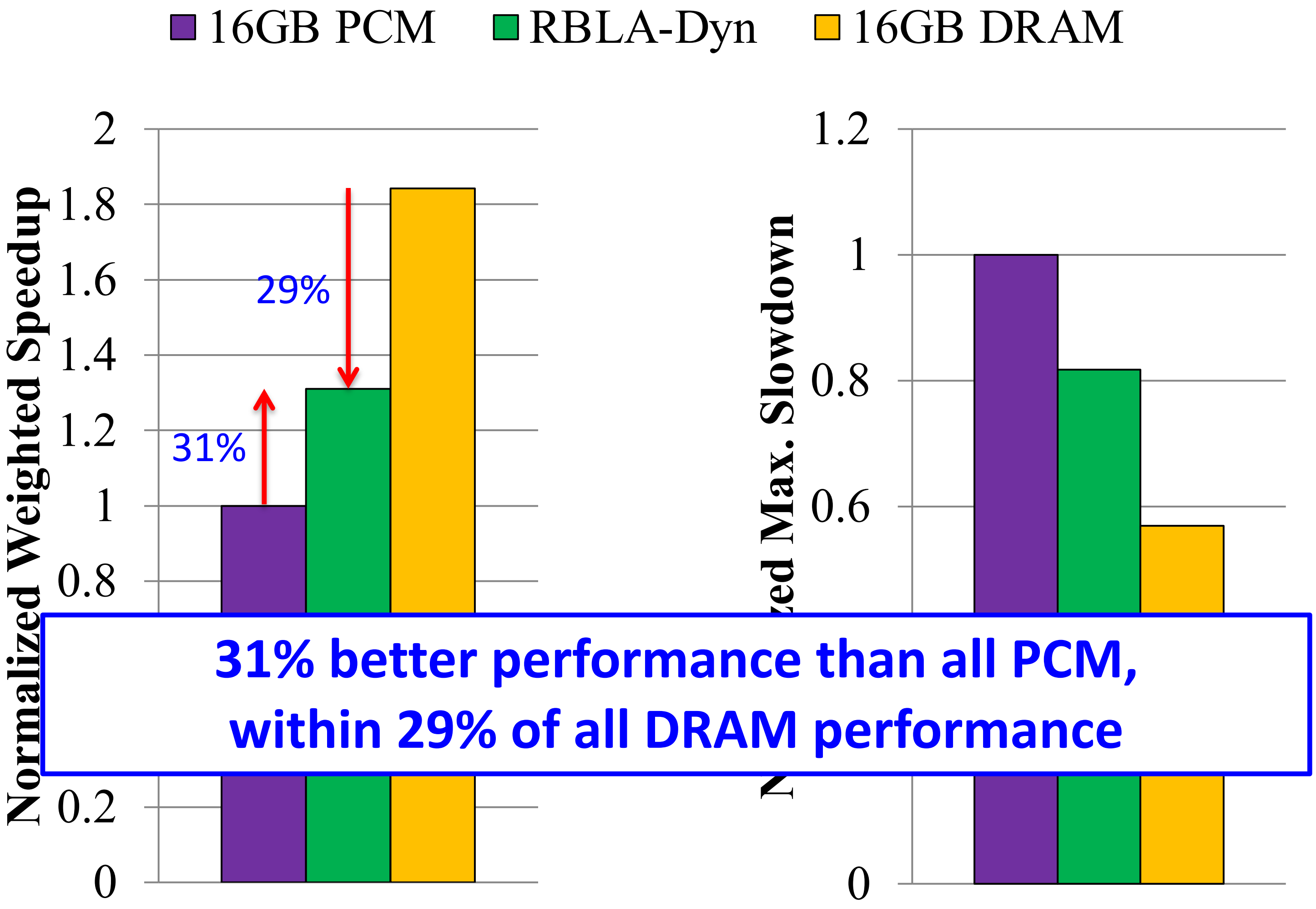
Key Observation & Idea

- Row buffers exist in both DRAM and PCM
  - Row hit latency **similar** in DRAM & PCM [Lee+ ISCA'09]
  - Row miss latency **small** in DRAM, **large** in PCM

- Place data in DRAM which
  - is likely to miss in the row buffer (low row buffer locality)→ miss penalty is smaller in DRAM

    AND
  - is reused many times → cache only the data worth the movement cost and DRAM space

Hybrid vs. All-PCM/DRAM [ICCD'12]

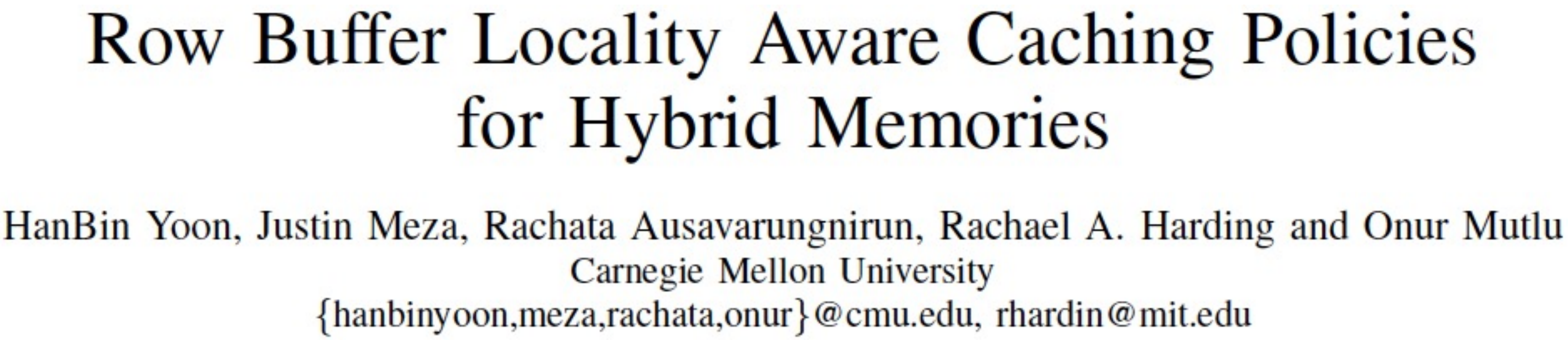


31% better performance than all PCM, within 29% of all DRAM performance

Yoon+, "Row Buffer Locality-Aware Data Placement in Hybrid Memories," ICCD 2012 Best Paper Award.

More on Hybrid Memory Data Placement

- HanBin Yoon, Justin Meza, Rachata Ausavarungnirun, Rachael Harding, and Onur Mutlu,
  "Row Buffer Locality Aware Caching Policies for Hybrid Memories"
  Proceedings of the 30th IEEE International Conference on Computer Design (***ICCD***), Montreal, Quebec, Canada, September 2012. Slides (pptx) (pdf)
  Best paper award (in Computer Systems and Applications track).



Row Buffer Locality Aware Caching Policies for Hybrid Memories

HanBin Yoon, Justin Meza, Rachata Ausavarungnirun, Rachael A. Harding and Onur Mutlu
Carnegie Mellon University
{hanbinyoon,meza,rachata,onur}@cmu.edu, rhardin@mit.edu

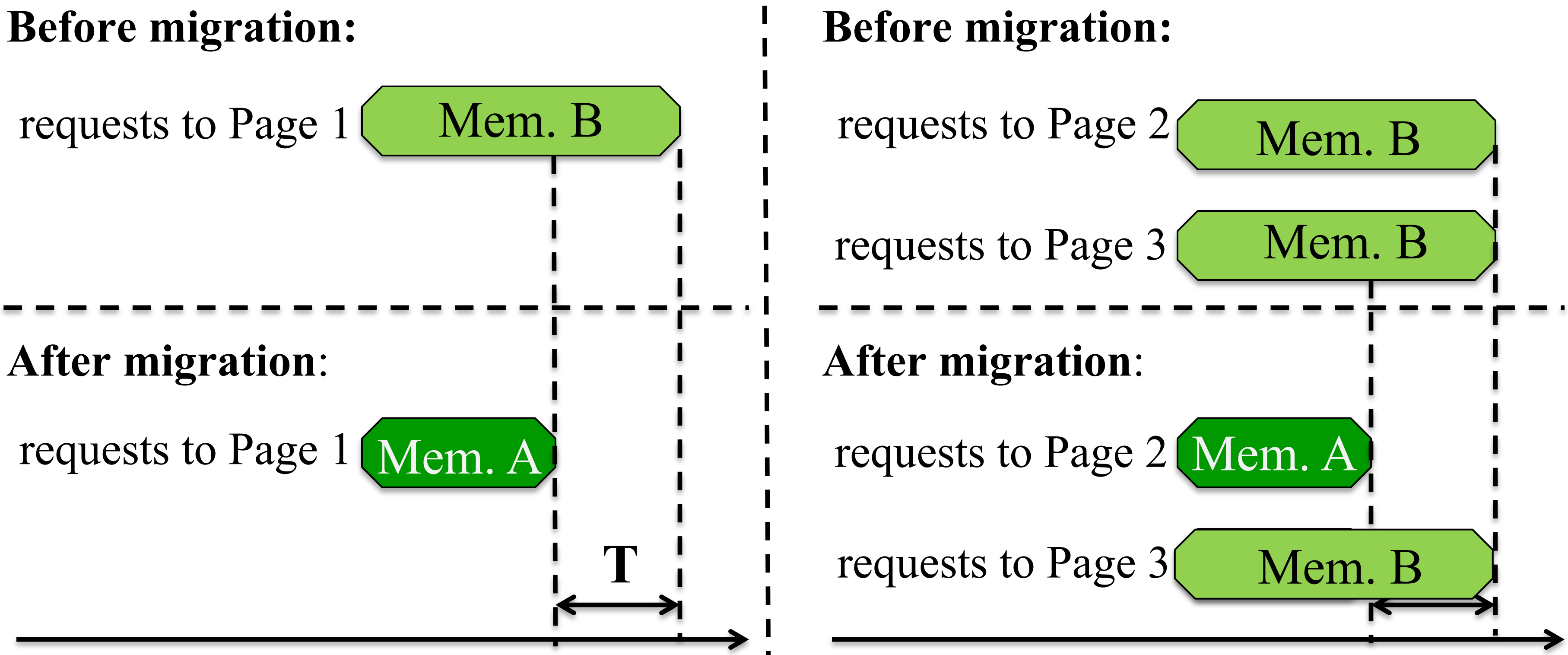
Weaknesses of Existing Solutions

- They are all heuristics that consider only a ***limited part* of memory access behavior**

- **Do not *directly* capture the overall system performance impact** of data placement decisions

- Example: None capture **memory-level parallelism** (MLP)
  - Number of ***concurrent* memory requests** from the same application when a page is accessed
  - Affects how much page migration helps performance

Importance of Memory-Level Parallelism



Before migration:

requests to Page 1 Mem. B

After migration:

requests to Page 1 Mem. A

T

time

Migrating one page reduces stall time by T

Before migration:

requests to Page 2 Mem. B

requests to Page 3 Mem. B

After migration:

requests to Page 2 Mem. A

requests to Page 3 Mem. B

time

Must migrate two pages to reduce stall time by T: migrating one page alone does not help

Page migration decisions **need to consider MLP**

Our Goal [CLUSTER 2017]

A **generalized** mechanism that

1. Directly estimates the **performance benefit of migrating a page** between **any two types of memory**

2. Places **only** the **performance-critical data** in the fast memory

Utility-Based Hybrid Memory Management

- A memory manager that works for ***any*** hybrid memory
  - e.g., DRAM-NVM, DRAM-RLDRAM

- **Key Idea**
  - For each page, use **comprehensive** characteristics to calculate estimated ***utility*** (i.e., performance impact) of migrating page from one memory to the other in the system
  - **Migrate only pages with the highest utility** (i.e., pages that improve system performance the most when migrated)

- Li+, "Utility-Based Hybrid Memory Management", CLUSTER 2017.

Key Mechanisms of UH-MEM

- For each page, estimate **utility** using a **performance model**
  - **Application stall time reduction**

    How much would migrating a page benefit the performance of the application that the page belongs to?
  - **Application performance sensitivity**

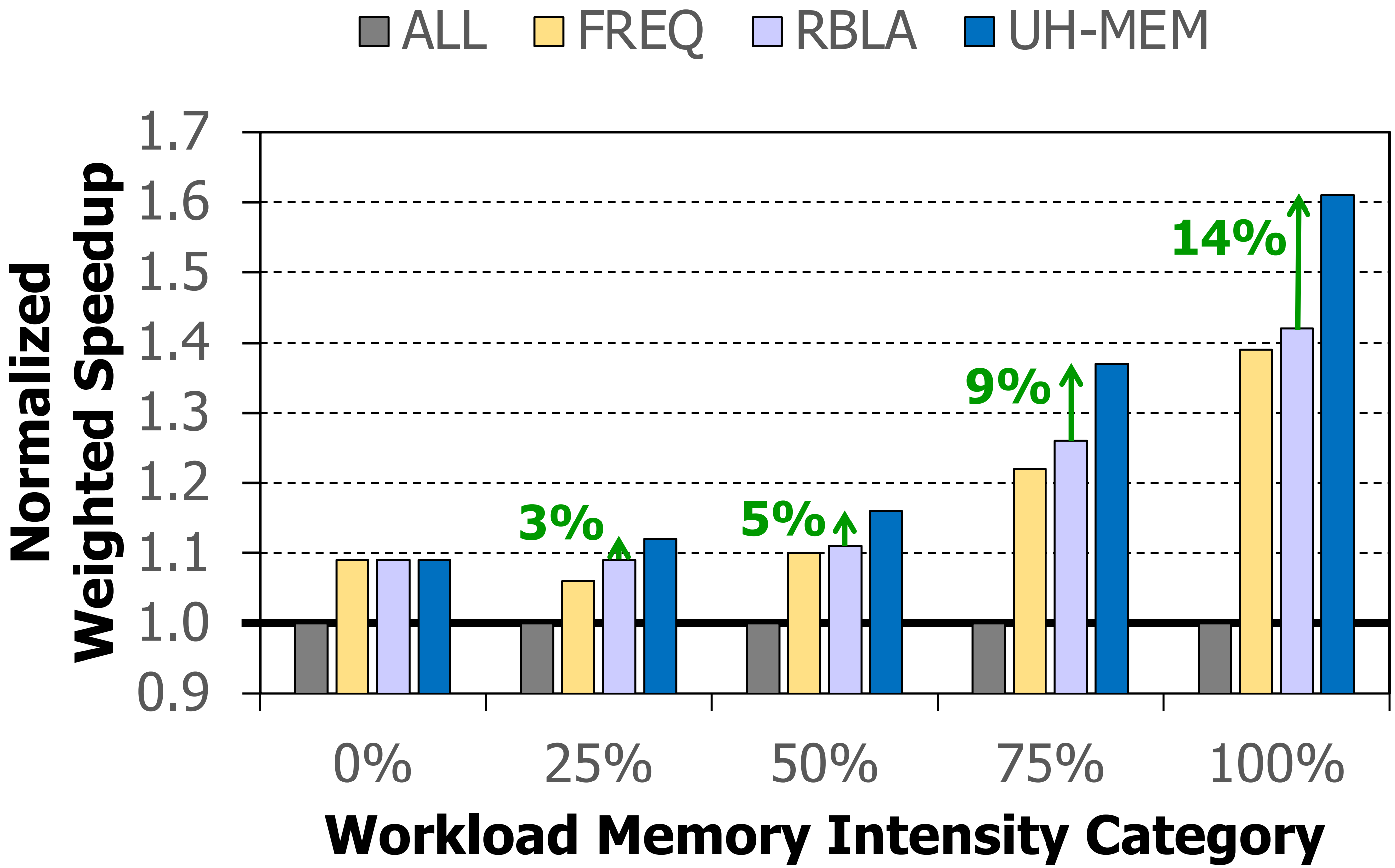
    How much does the improvement of a single application's performance increase the *overall* system performance?

$$Utility = \Delta StallTime_i \times Sensitivity_i$$

- **Migrate** only pages whose utility exceed the migration threshold from slow memory to fast memory

- Periodically **adjust migration threshold**

Results: System Performance

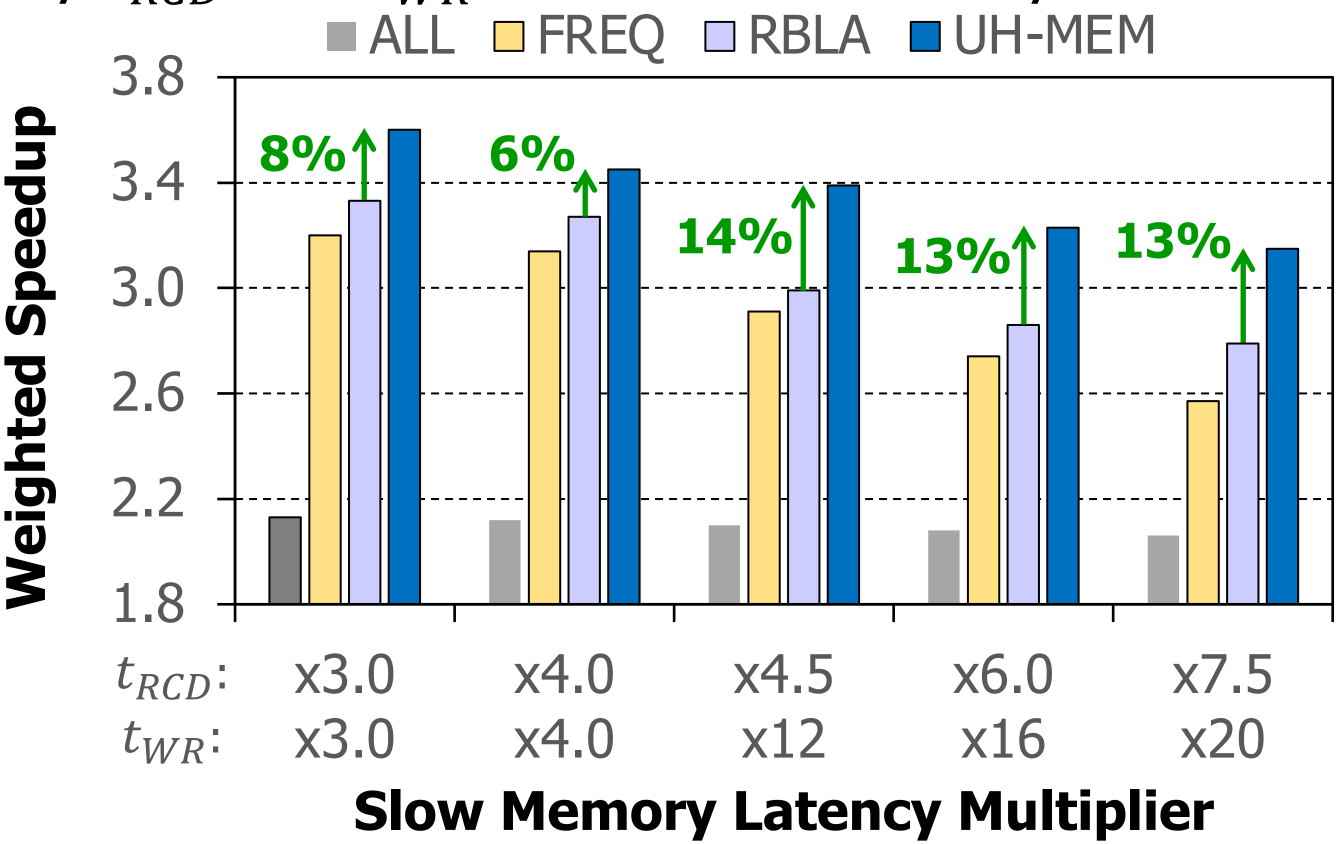


UH-MEM improves system performance
over the best state-of-the-art hybrid memory manager

SAFARI

Results: Sensitivity to Slow Memory Latency

- We vary t_{RCD} and t_{WR} of the slow memory



UH-MEM improves system performance for a wide variety of hybrid memory systems

More on UH-MEM

- Yang Li, Saugata Ghose, Jongmoo Choi, Jin Sun, Hui Wang, and Onur Mutlu,
  "Utility-Based Hybrid Memory Management"
  Proceedings of the 19th IEEE Cluster Conference (***CLUSTER***), Honolulu, Hawaii, USA, September 2017.
  [Slides (pptx) (pdf)]

Utility-Based Hybrid Memory Management

Yang Li[†] Saugata Ghose[†] Jongmoo Choi[‡] Jin Sun[†] Hui Wang[⋆] Onur Mutlu[+][†]

[†] *Carnegie Mellon University* [‡] *Dankook University* [⋆] *Beihang University* [+] *ETH Zürich*

Challenge and Opportunity

Enabling

an Emerging Technology

to Augment DRAM

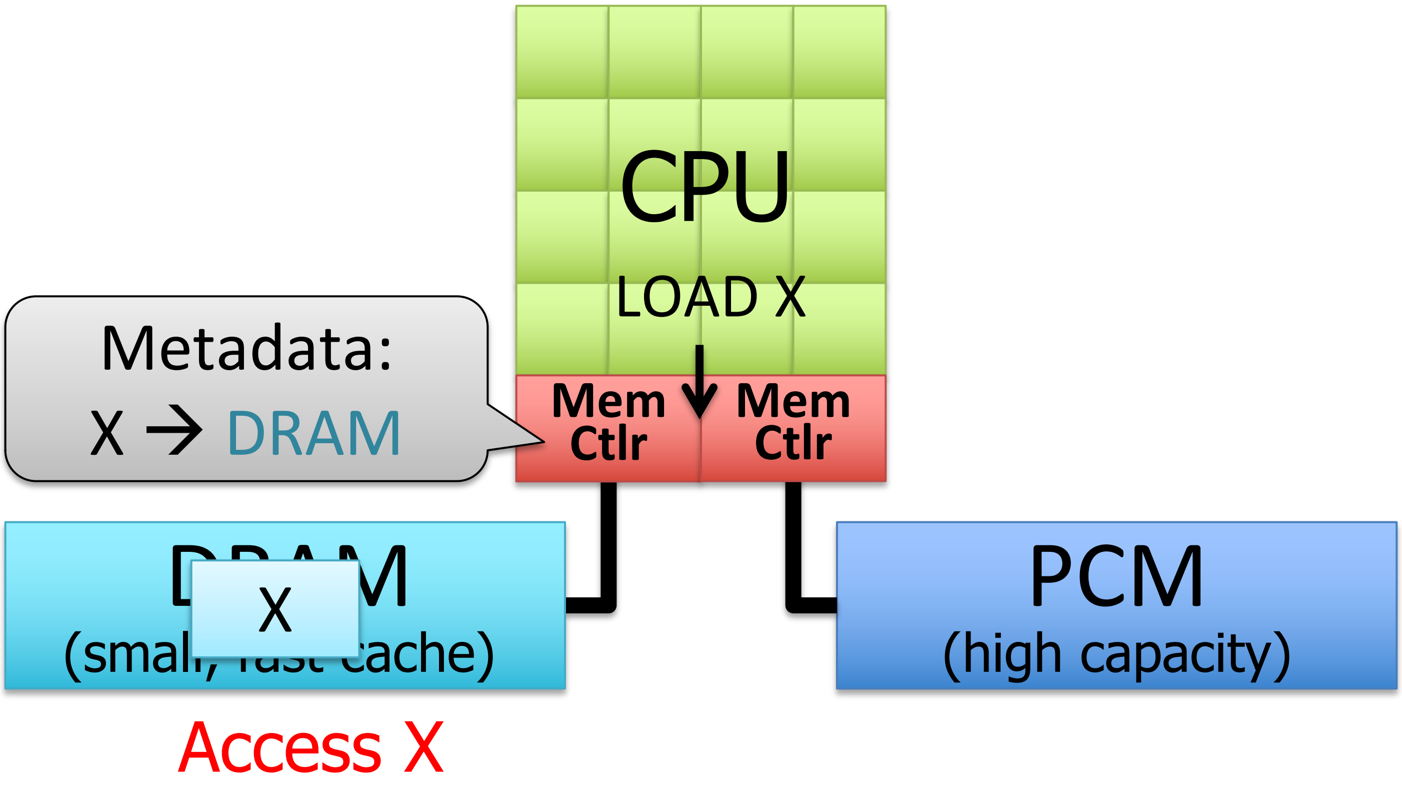
Managing Hybrid Memories

Another Challenge

Designing Effective
Large (DRAM) Caches

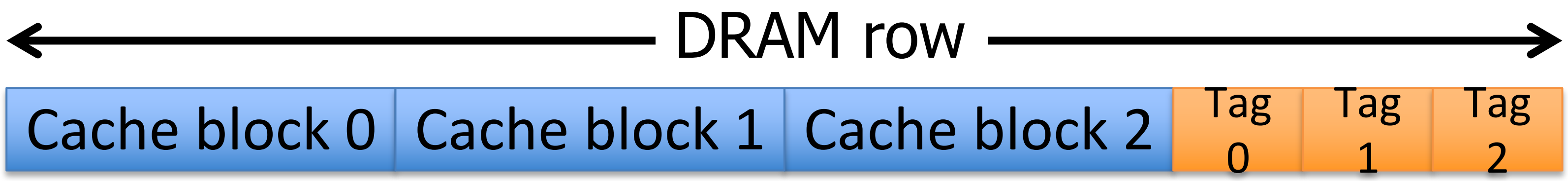
One Problem with Large DRAM Caches

- A large DRAM cache requires a large metadata (tag + block-based information) store
- How do we design an efficient DRAM cache?



Idea 1: Tags in Memory

- Store tags in the same row as data in DRAM
  - Store metadata in same row as their data
  - Data and metadata can be accessed together



DRAM row

Cache block 0 | Cache block 1 | Cache block 2 | Tag 0 | Tag 1 | Tag 2

- Benefit: No on-chip tag storage overhead
- Downsides:
  - Cache hit determined only after a DRAM access
  - Cache hit requires two DRAM accesses

Idea 2: Cache Tags in SRAM

- Recall Idea 1: Store all metadata in DRAM
  - To reduce metadata storage overhead

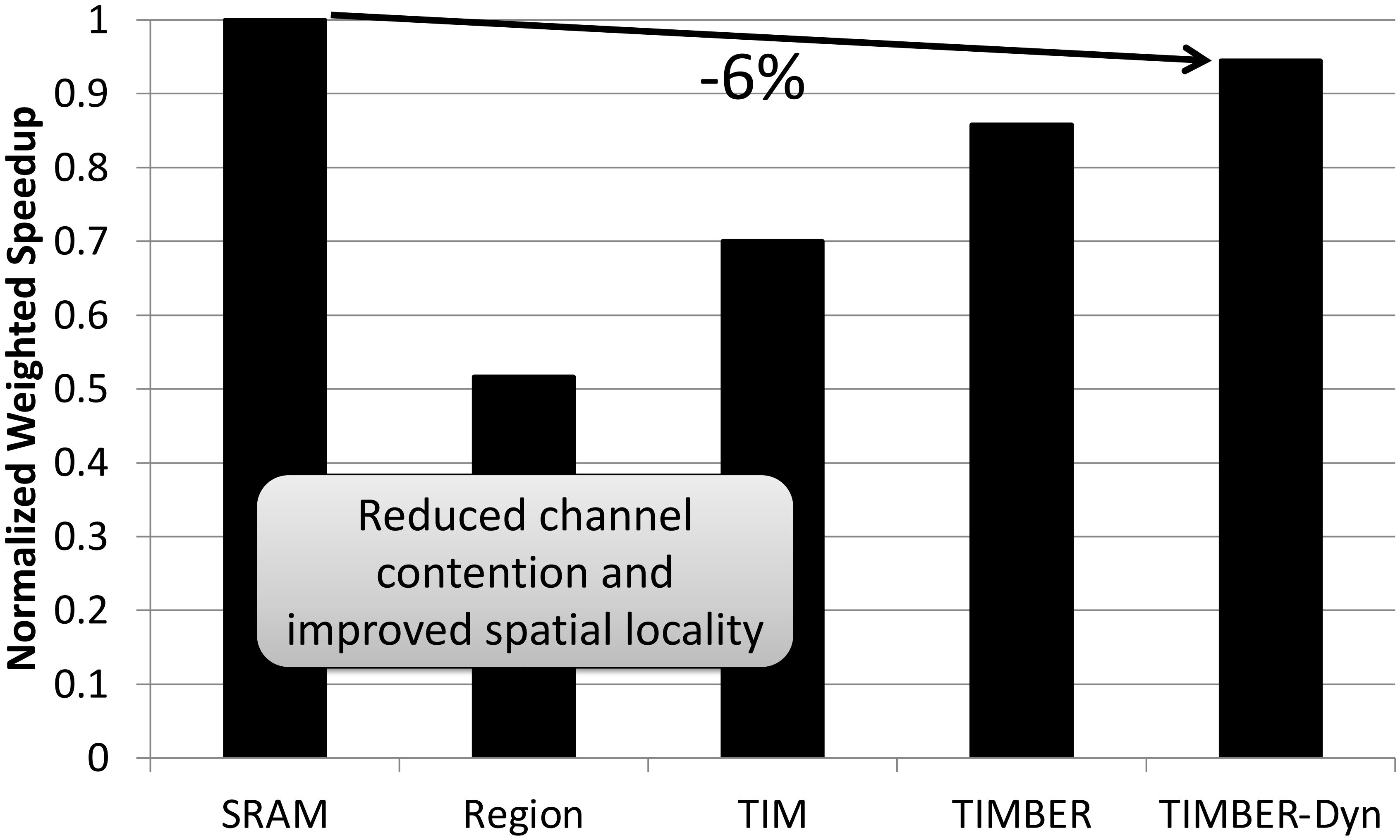
- Idea 2: Cache in on-chip SRAM frequently-accessed metadata
  - Cache only a small amount to keep SRAM size small

Idea 3: Dynamic Data Transfer Granularity

- Some applications benefit from caching more data
  - They have good spatial locality
- Others do not
  - Large granularity wastes bandwidth and reduces cache utilization

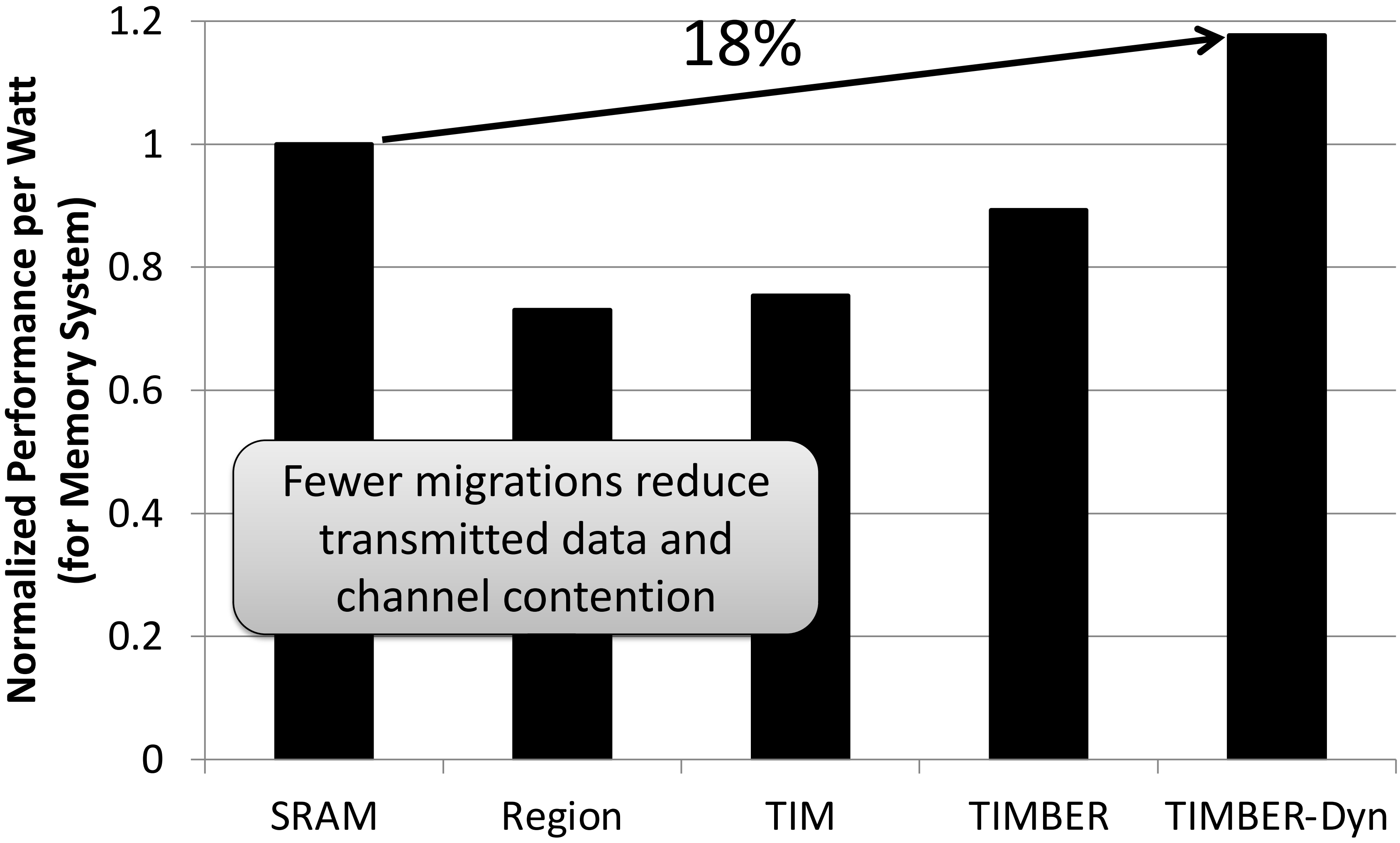
- Idea 3: Simple dynamic caching granularity policy
  - Cost-benefit analysis to determine best DRAM cache block size
  - Group main memory into sets of rows
  - Different sampled row sets follow different fixed caching granularities
  - The rest of main memory follows the best granularity
    - Cost–benefit analysis: access latency versus number of cachings
    - Performed every quantum

TIMBER Performance



Meza, Chang, Yoon, Mutlu, Ranganathan, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters, 2012.

TIMBER Energy Efficiency

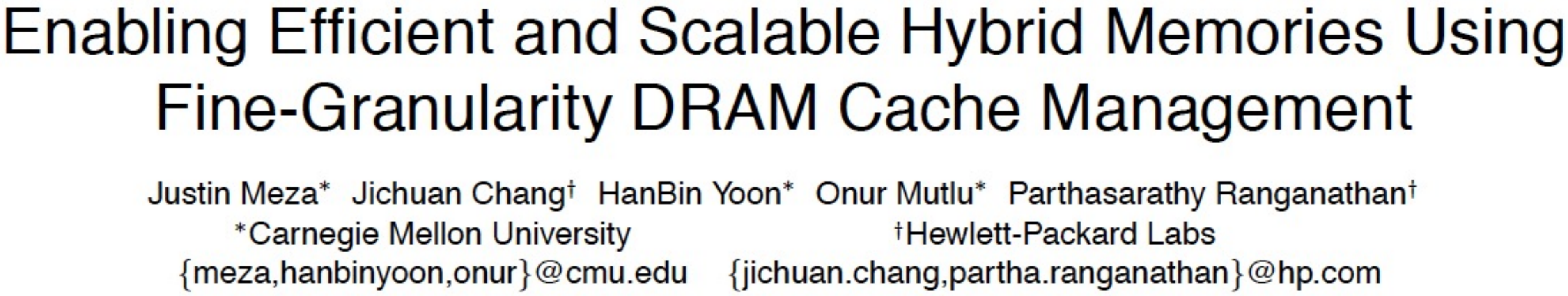


Fewer migrations reduce transmitted data and channel contention

Meza, Chang, Yoon, Mutlu, Ranganathan, "Enabling Efficient and Scalable Hybrid Memories," IEEE Comp. Arch. Letters, 2012.

On Large DRAM Cache Design

- Justin Meza, Jichuan Chang, HanBin Yoon, Onur Mutlu, and Parthasarathy Ranganathan, **"Enabling Efficient and Scalable Hybrid Memories Using Fine-Granularity DRAM Cache Management"** *IEEE Computer Architecture Letters* (***CAL***), February 2012.



Enabling Efficient and Scalable Hybrid Memories Using Fine-Granularity DRAM Cache Management

Justin Meza[*] Jichuan Chang[†] HanBin Yoon[*] Onur Mutlu[*] Parthasarathy Ranganathan[†]
[*]Carnegie Mellon University [†]Hewlett-Packard Labs
{meza,hanbinyoon,onur}@cmu.edu {jichuan.chang,partha.ranganathan}@hp.com

DRAM Caches: Many Recent Options

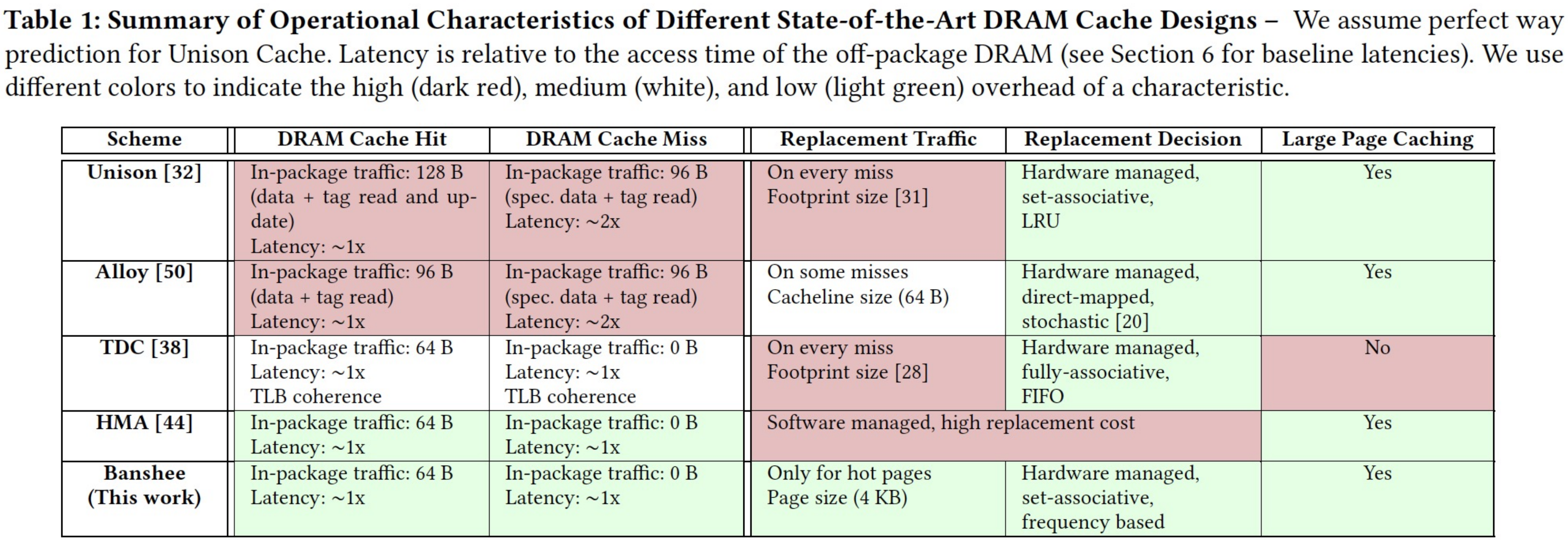


Table 1: Summary of Operational Characteristics of Different State-of-the-Art DRAM Cache Designs – We assume perfect way prediction for Unison Cache. Latency is relative to the access time of the off-package DRAM (see Section 6 for baseline latencies). We use different colors to indicate the high (dark red), medium (white), and low (light green) overhead of a characteristic.

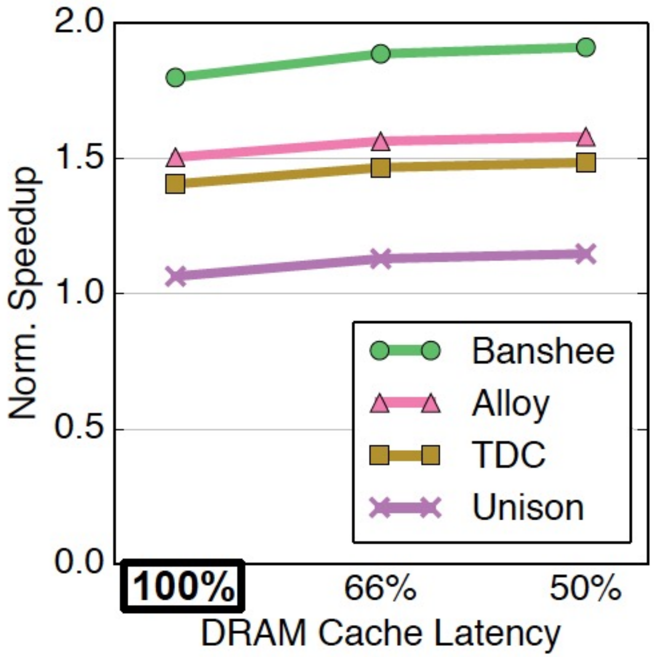
Scheme	DRAM Cache Hit	DRAM Cache Miss	Replacement Traffic	Replacement Decision	Large Page Caching
Unison [32]	In-package traffic: 128 B (data + tag read and update) Latency: ~1x	In-package traffic: 96 B (spec. data + tag read) Latency: ~2x	On every miss Footprint size [31]	Hardware managed, set-associative, LRU	Yes
Alloy [50]	In-package traffic: 96 B (data + tag read) Latency: ~1x	In-package traffic: 96 B (spec. data + tag read) Latency: ~2x	On some misses Cacheline size (64 B)	Hardware managed, direct-mapped, stochastic [20]	Yes
TDC [38]	In-package traffic: 64 B Latency: ~1x TLB coherence	In-package traffic: 0 B Latency: ~1x TLB coherence	On every miss Footprint size [28]	Hardware managed, fully-associative, FIFO	No
HMA [44]	In-package traffic: 64 B Latency: ~1x	In-package traffic: 0 B Latency: ~1x	Software managed, high replacement cost		Yes
Banshee (This work)	In-package traffic: 64 B Latency: ~1x	In-package traffic: 0 B Latency: ~1x	Only for hot pages Page size (4 KB)	Hardware managed, set-associative, frequency based	Yes

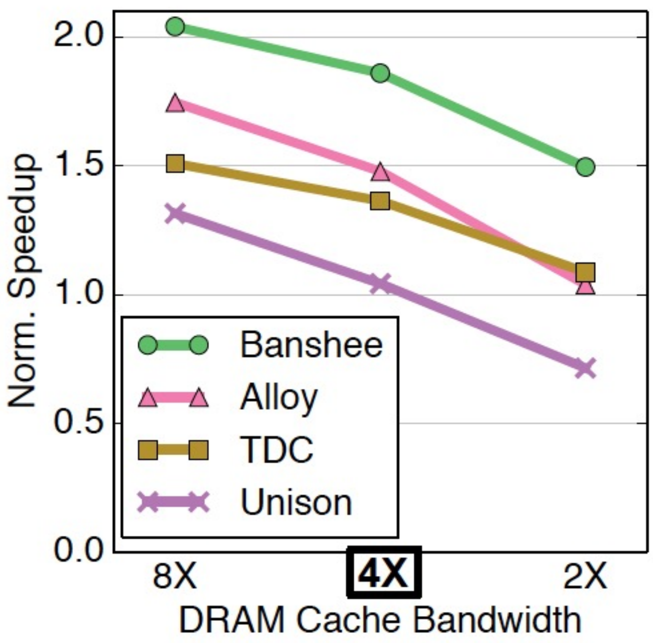
Yu+, "Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation," MICRO 2017.

Banshee [MICRO 2017]

- Tracks presence in cache using TLB and Page Table
  - No tag store needed for DRAM cache
  - Enabled by a new lightweight lazy TLB coherence protocol

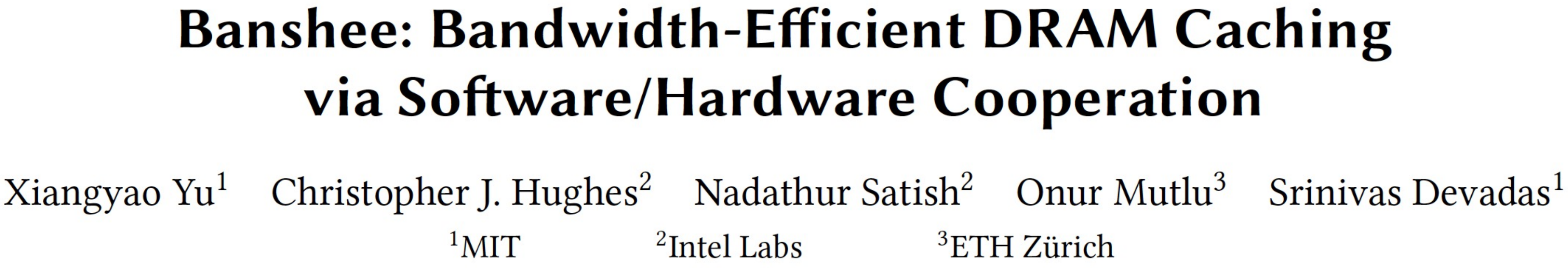
- New bandwidth-aware frequency-based replacement policy





More on Banshee

- Xiangyao Yu, Christopher J. Hughes, Nadathur Satish, Onur Mutlu, and Srinivas Devadas,
  "Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation"
  Proceedings of the 50th International Symposium on Microarchitecture (***MICRO***), Boston, MA, USA, October 2017.



Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation

Xiangyao Yu[1] Christopher J. Hughes[2] Nadathur Satish[2] Onur Mutlu[3] Srinivas Devadas[1]

[1]MIT [2]Intel Labs [3]ETH Zürich

Other Opportunities with Emerging Technologies

- Merging of memory and storage
  - e.g., a single interface to manage all data
- New applications
  - e.g., ultra-fast checkpoint and restore
- More robust system design
  - e.g., reducing data loss
- Processing tightly-coupled with memory
  - e.g., enabling efficient search and filtering

Recall: Processing Using Memory

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 Bulk Bitwise AND/OR

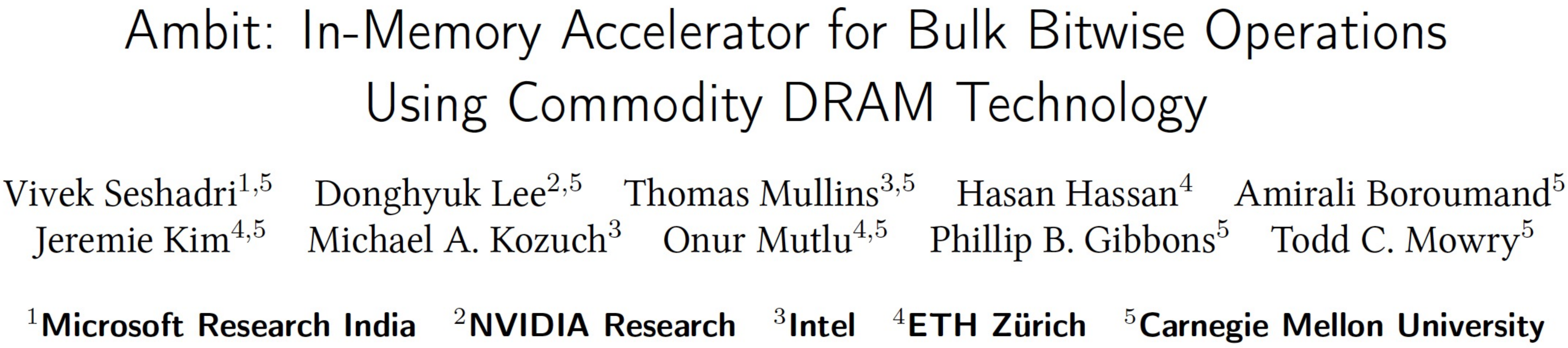
- Vivek Seshadri, Kevin Hsieh, Amirali Boroumand, Donghyuk Lee, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
"Fast Bulk Bitwise AND and OR in DRAM"
IEEE Computer Architecture Letters (***CAL***), April 2015.

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]

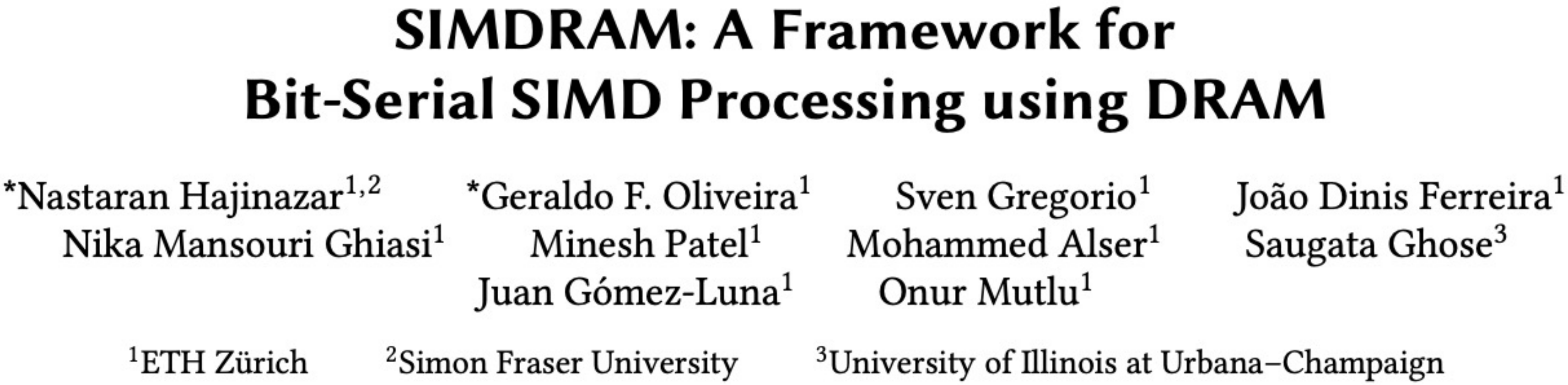
In-DRAM Bulk Bitwise Execution Engine

Vivek Seshadri
Microsoft Research India
visesha@microsoft.com

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

SIMDRAM Framework for in-DRAM Computing

- 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)]

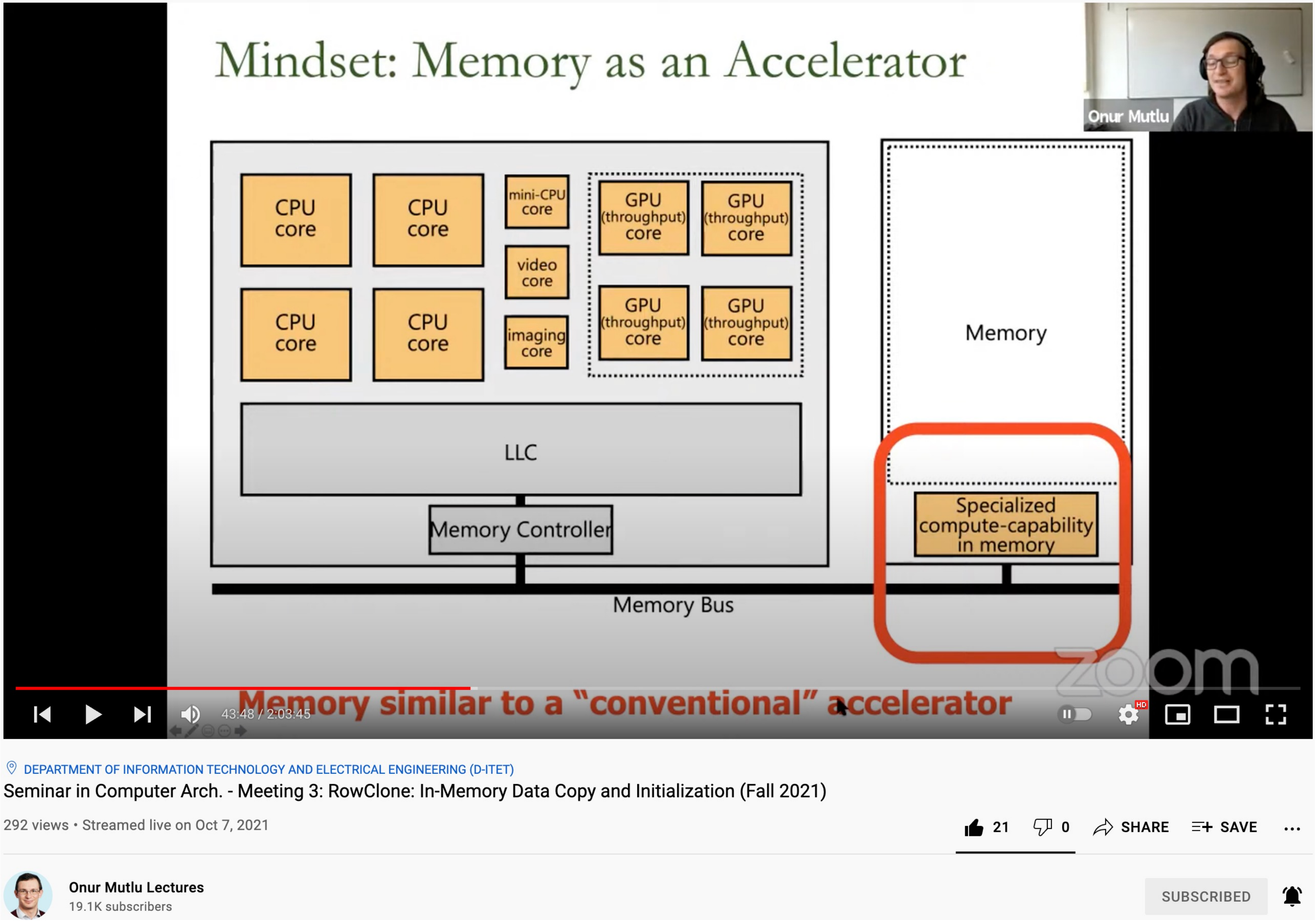


SIMDRAM: A Framework for
Bit-Serial SIMD Processing using DRAM

*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

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)

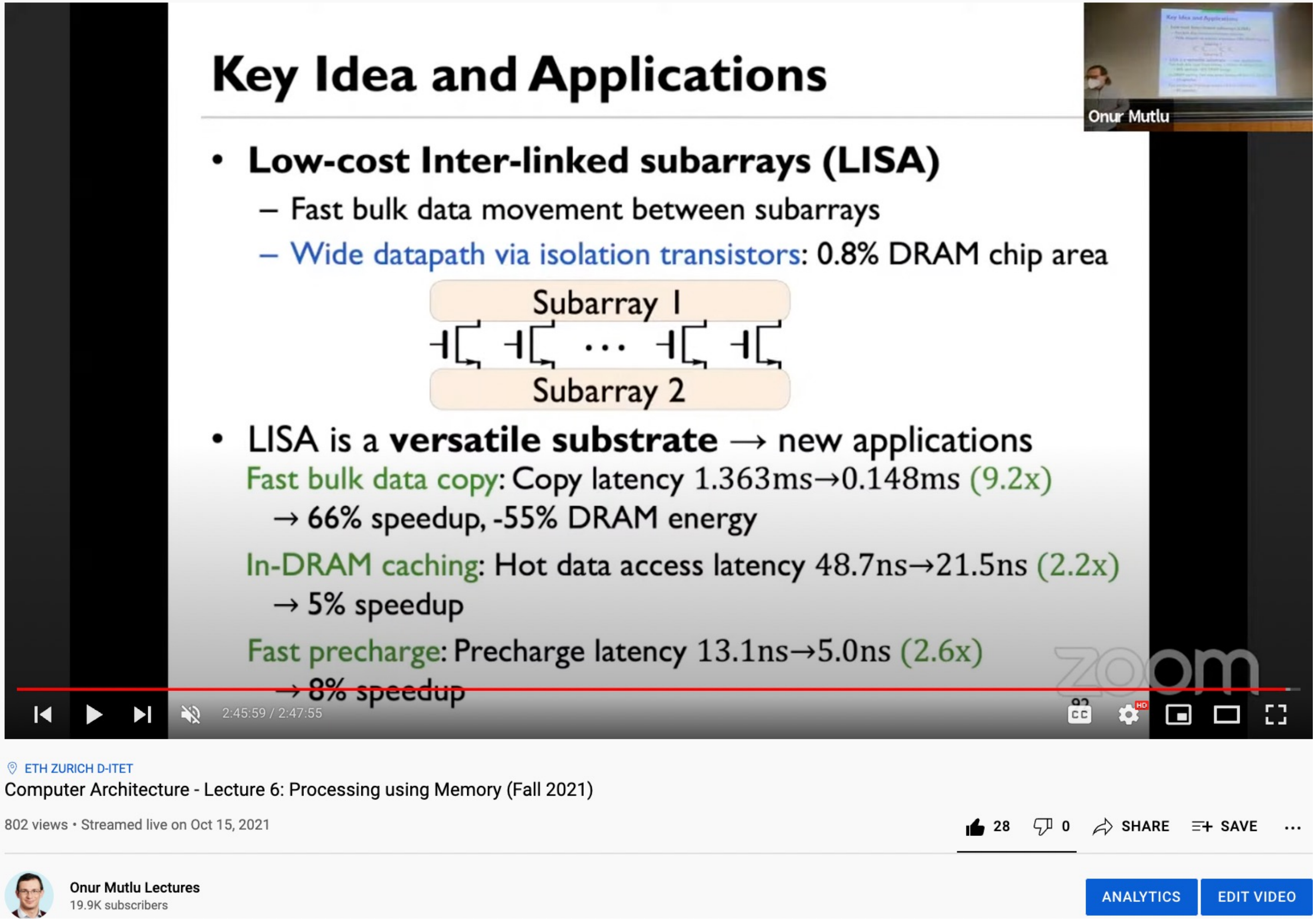
292 views • Streamed live on Oct 7, 2021

Onur Mutlu Lectures

19.1K subscribers

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

Lecture on Processing using Memory (I)



Key Idea and Applications

- **Low-cost Inter-linked subarrays (LISA)**
  - Fast bulk data movement between subarrays
  - Wide datapath via isolation transistors: 0.8% DRAM chip area

- LISA is a **versatile substrate** → new applications

  Fast bulk data copy: Copy latency 1.363ms→0.148ms (9.2x)
  → 66% speedup, -55% DRAM energy

  In-DRAM caching: Hot data access latency 48.7ns→21.5ns (2.2x)
  → 5% speedup

  Fast precharge: Precharge latency 13.1ns→5.0ns (2.6x)
  → 8% speedup

ETH ZURICH D-ITET

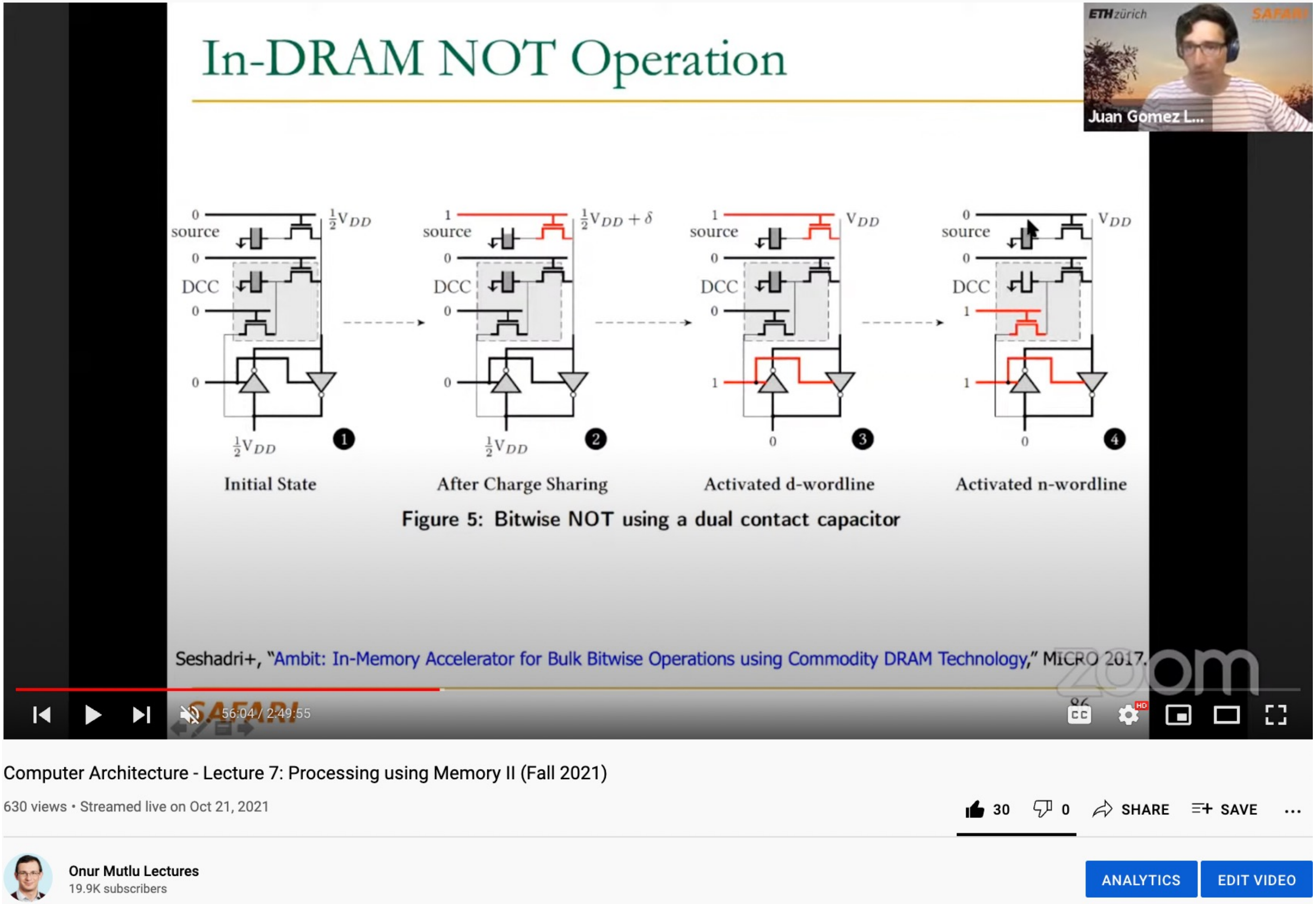
Computer Architecture - Lecture 6: Processing using Memory (Fall 2021)

802 views • Streamed live on Oct 15, 2021

Onur Mutlu Lectures
19.9K subscribers

https://www.youtube.com/watch?v=HNd4skQrt6I&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF&index=6

Lecture on Processing using Memory (II)



In-DRAM NOT Operation

Figure 5: Bitwise NOT using a dual contact capacitor

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

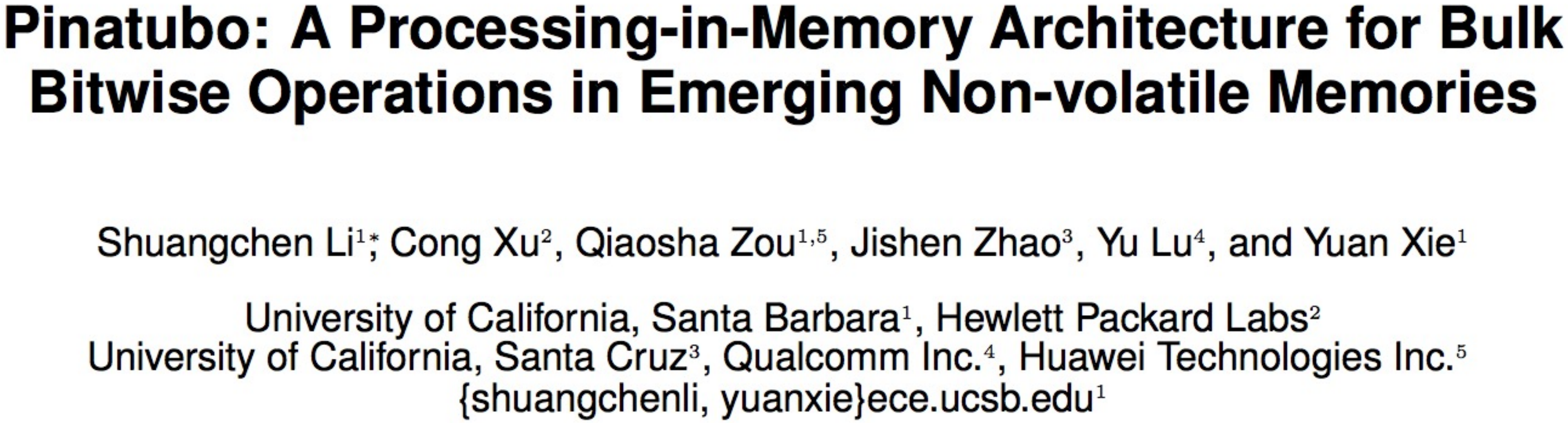
Computer Architecture - Lecture 7: Processing using Memory II (Fall 2021)

630 views • Streamed live on Oct 21, 2021

Onur Mutlu Lectures
19.9K subscribers

https://www.youtube.com/watch?v=k56x2qcaXWY&list=PL5Q2soXY2Zi-Mnk1PxjEIG32HAGILkTOF&index=7

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

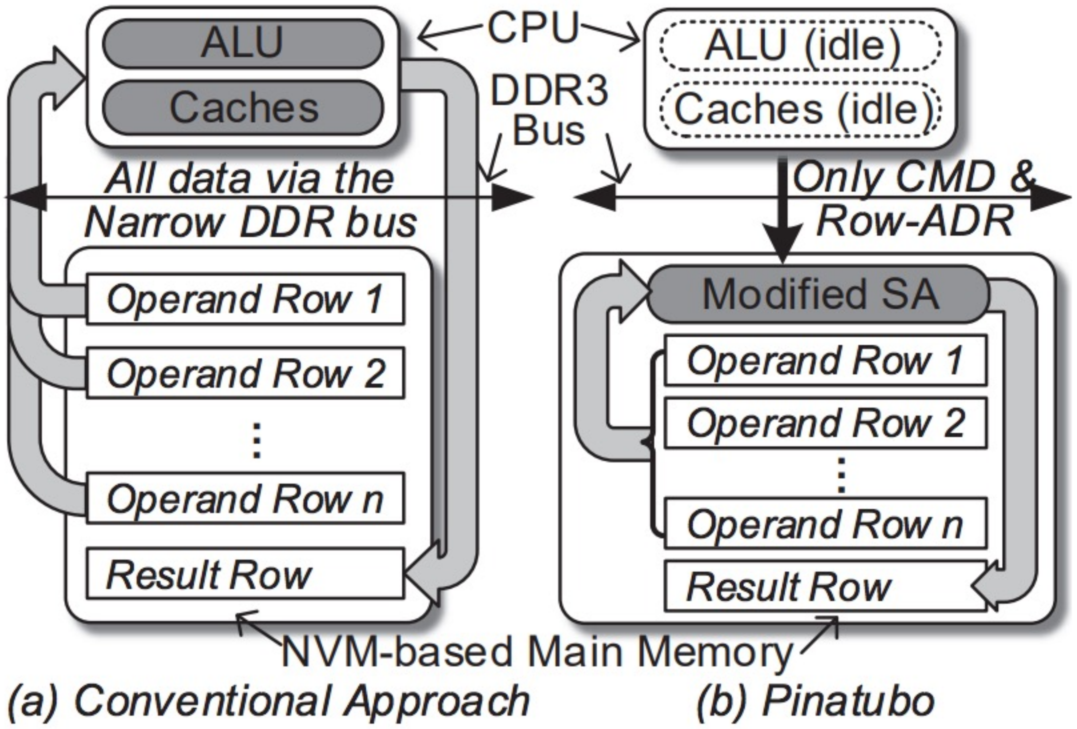


Figure 2: Overview: (a) Computing-centric approach, moving tons of data to CPU and write back. (b) The proposed Pinatubo architecture, performs n-row bitwise operations inside NVM in one step.

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

New: In-Memory Crossbar Array Operations

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

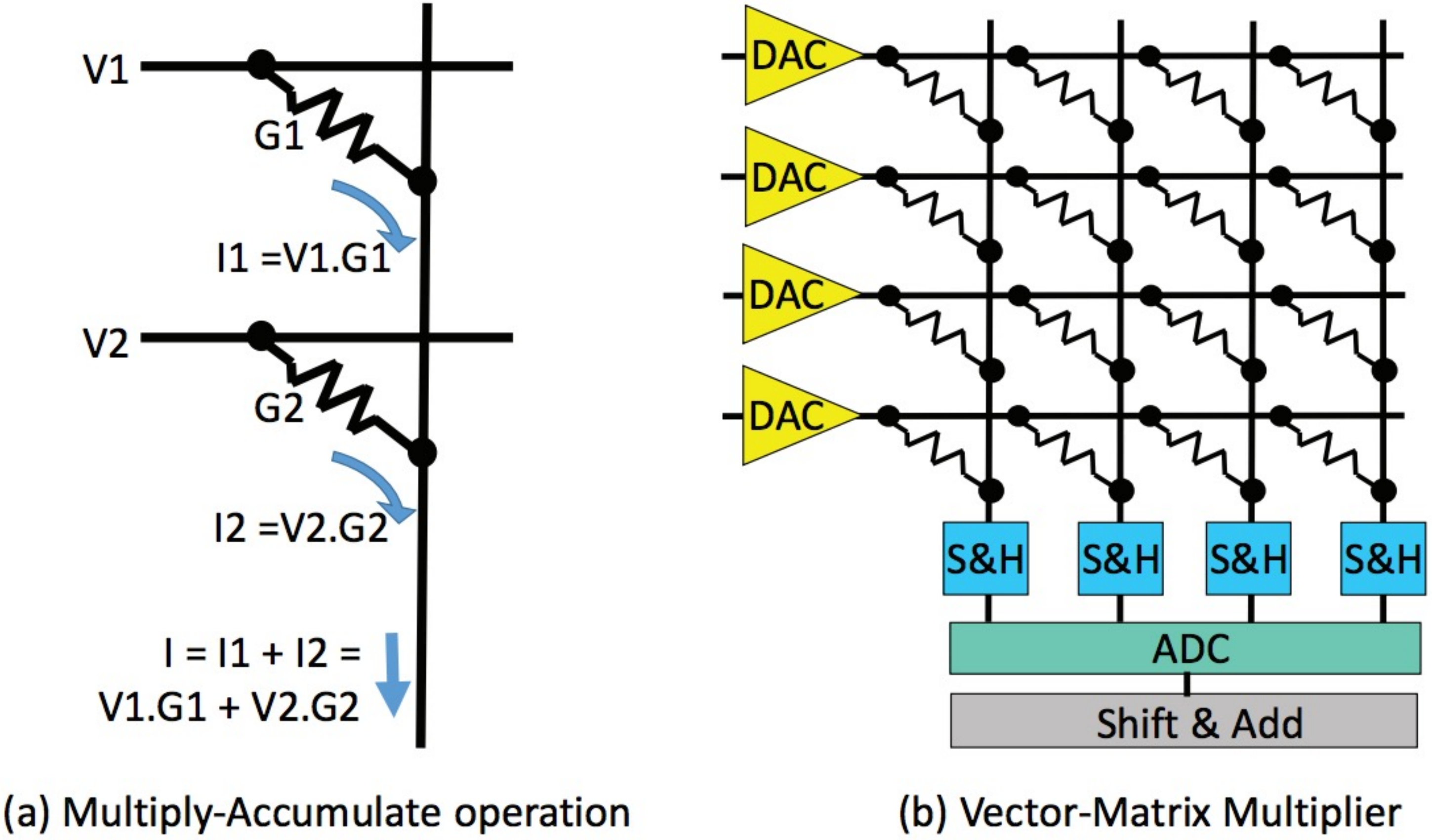
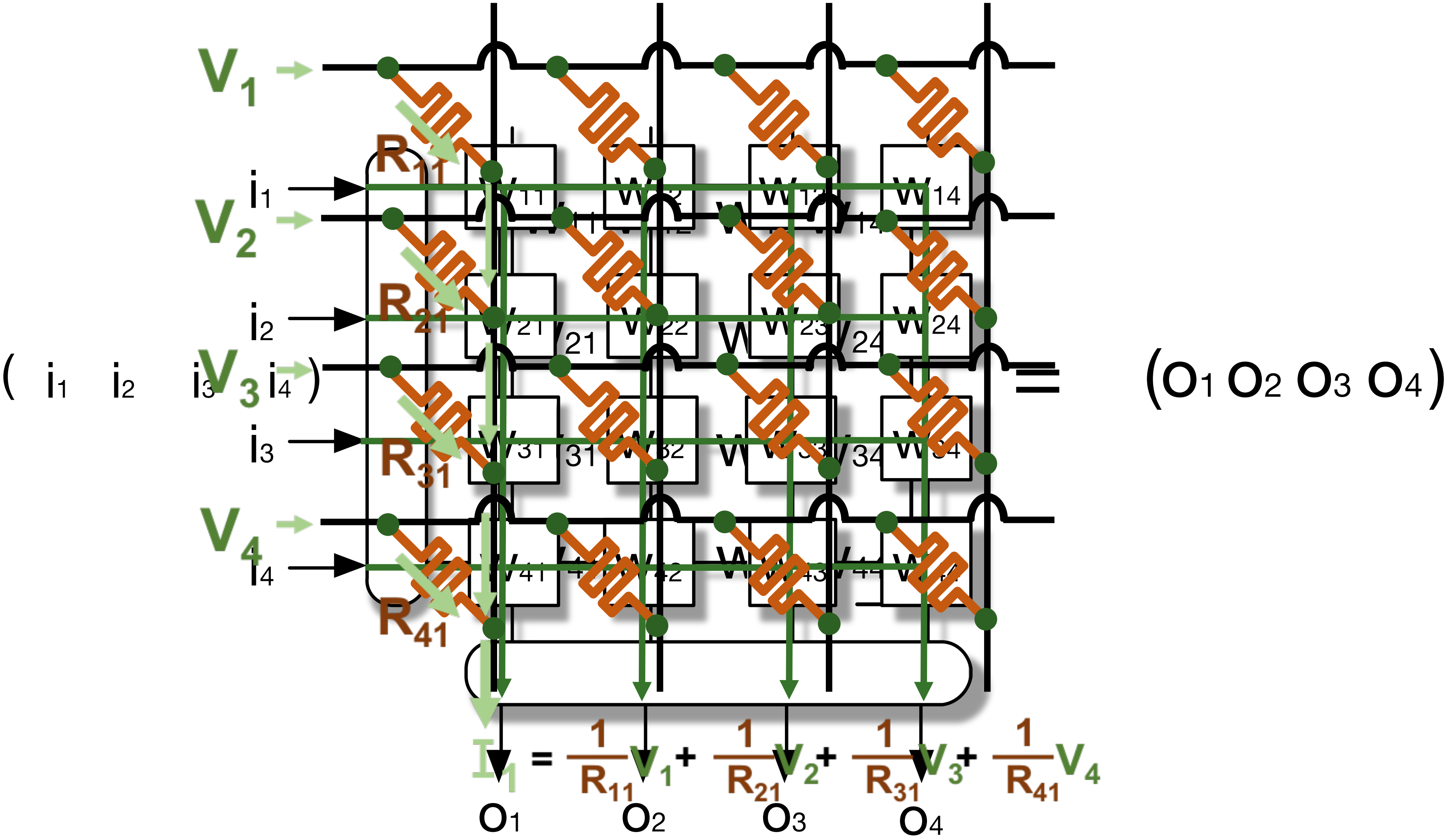


Fig. 1. (a) Using a bitline to perform an analog sum of products operation. (b) A memristor crossbar used as a vector-matrix multiplier.

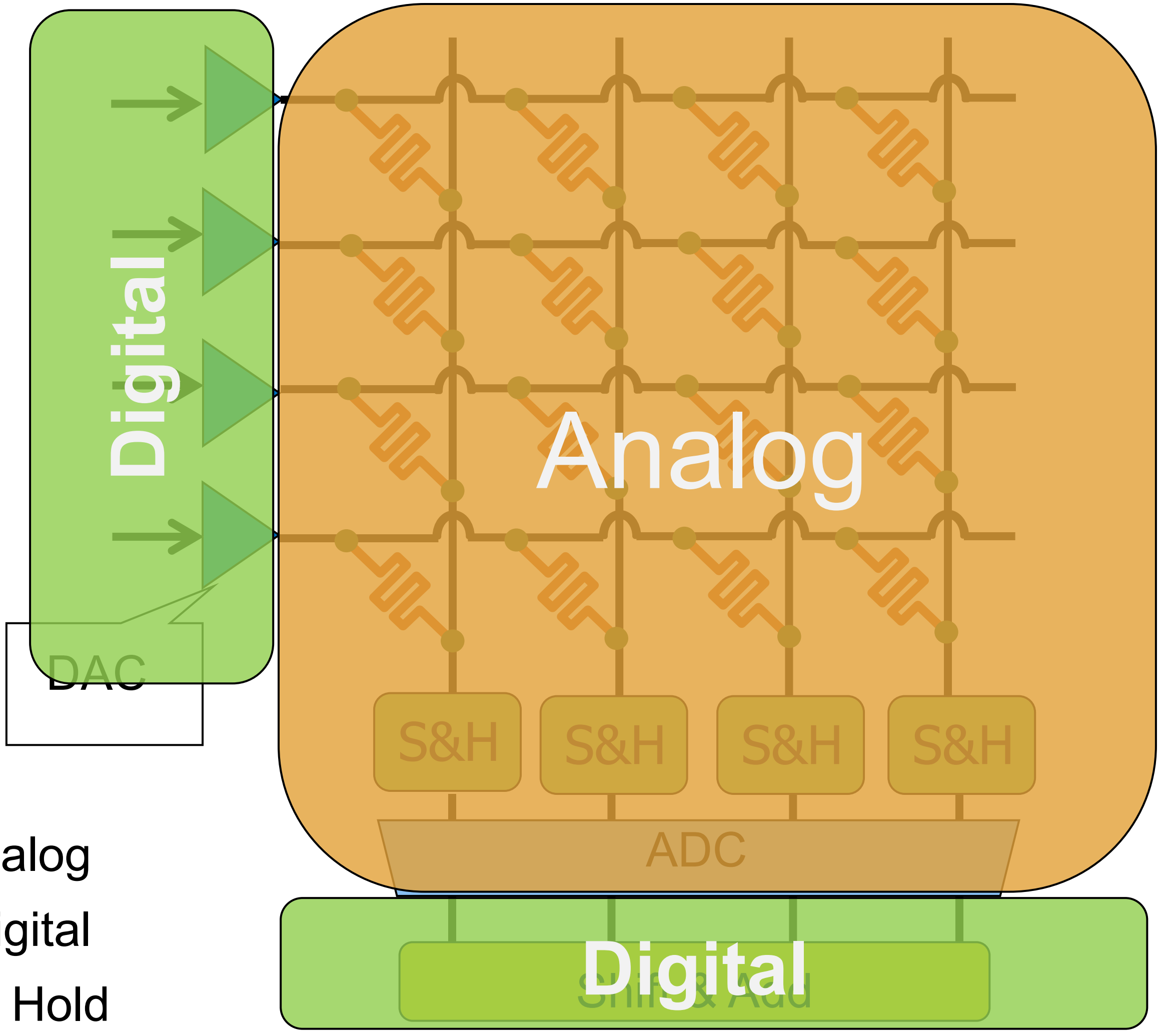
Shafiee+, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars", ISCA 2016.

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

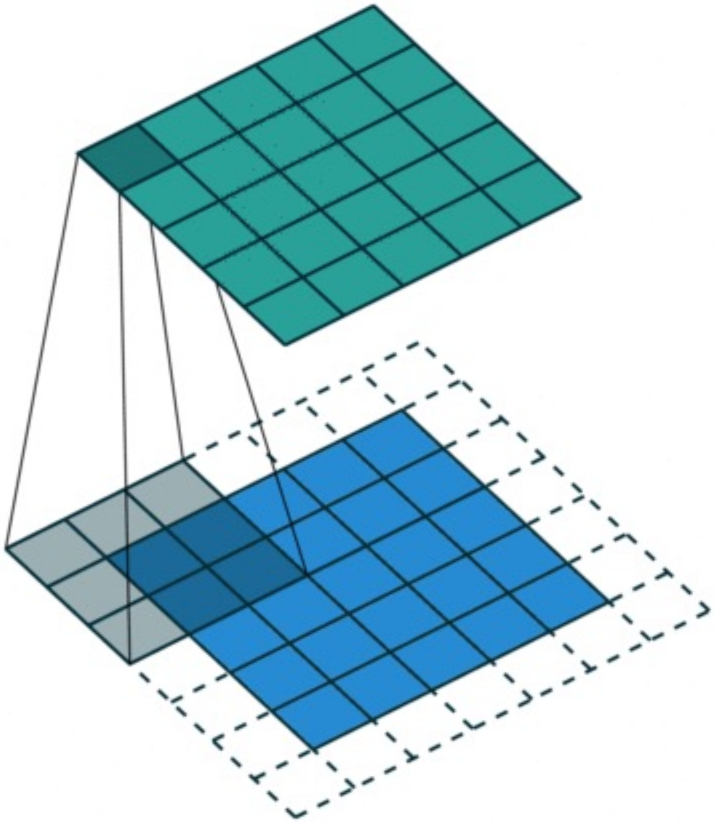
ADC: Analog to Digital

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

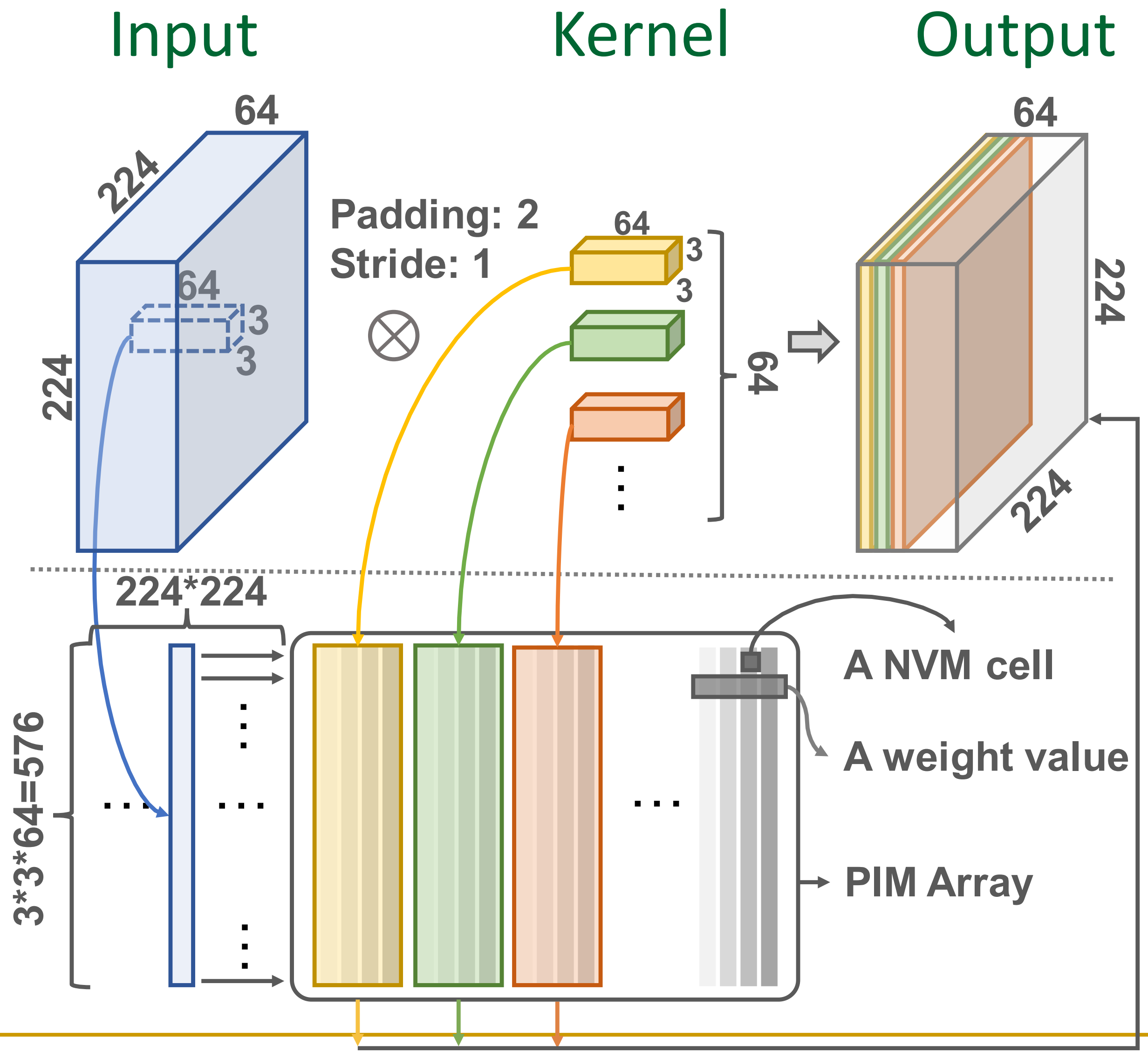
Padding: 1 (white)

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

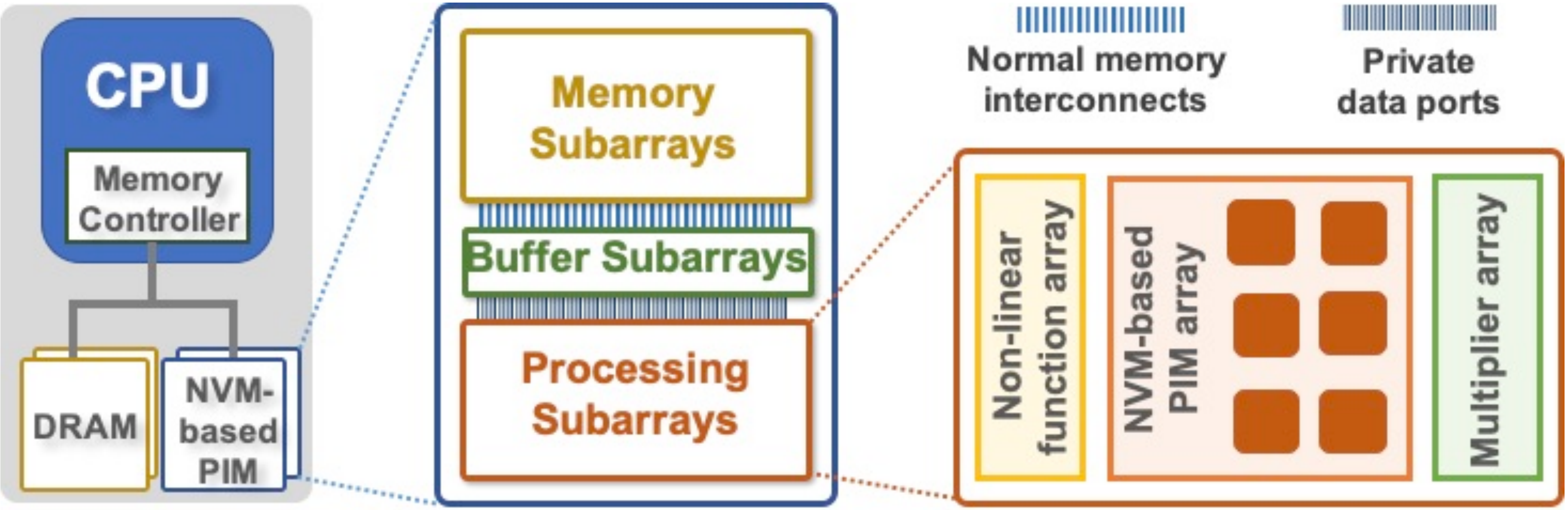
Mapping Computation onto the Crossbar

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.

Other Opportunities with Emerging Technologies

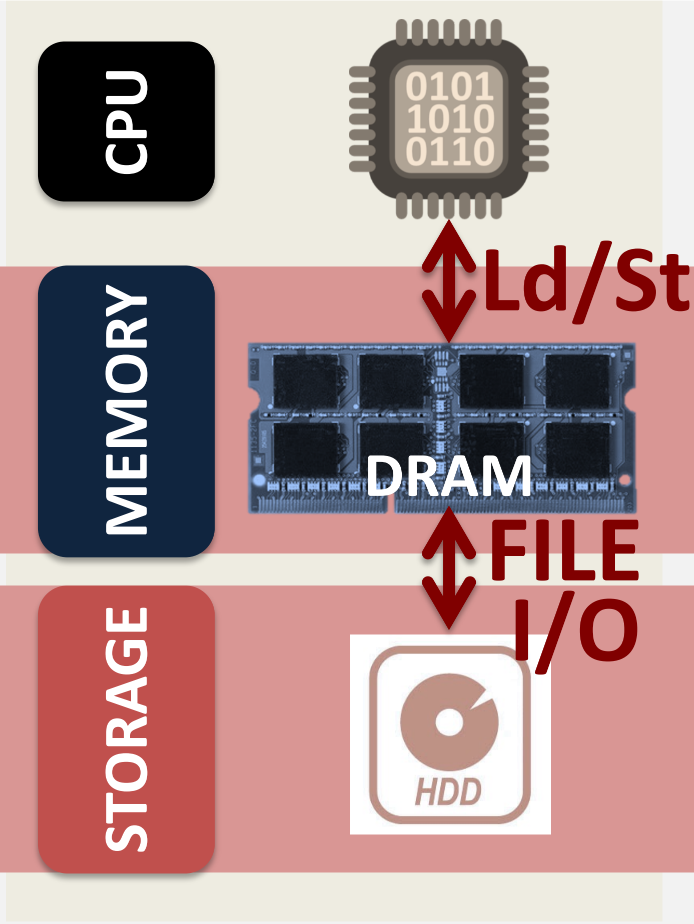
- Merging of memory and storage
  - e.g., a single interface to manage all data

- New applications
  - e.g., ultra-fast checkpoint and restore

- More robust system design
  - e.g., reducing data loss

- Processing tightly-coupled with memory
  - e.g., enabling efficient search and filtering

TWO-LEVEL STORAGE MODEL



VOLATILE

FAST

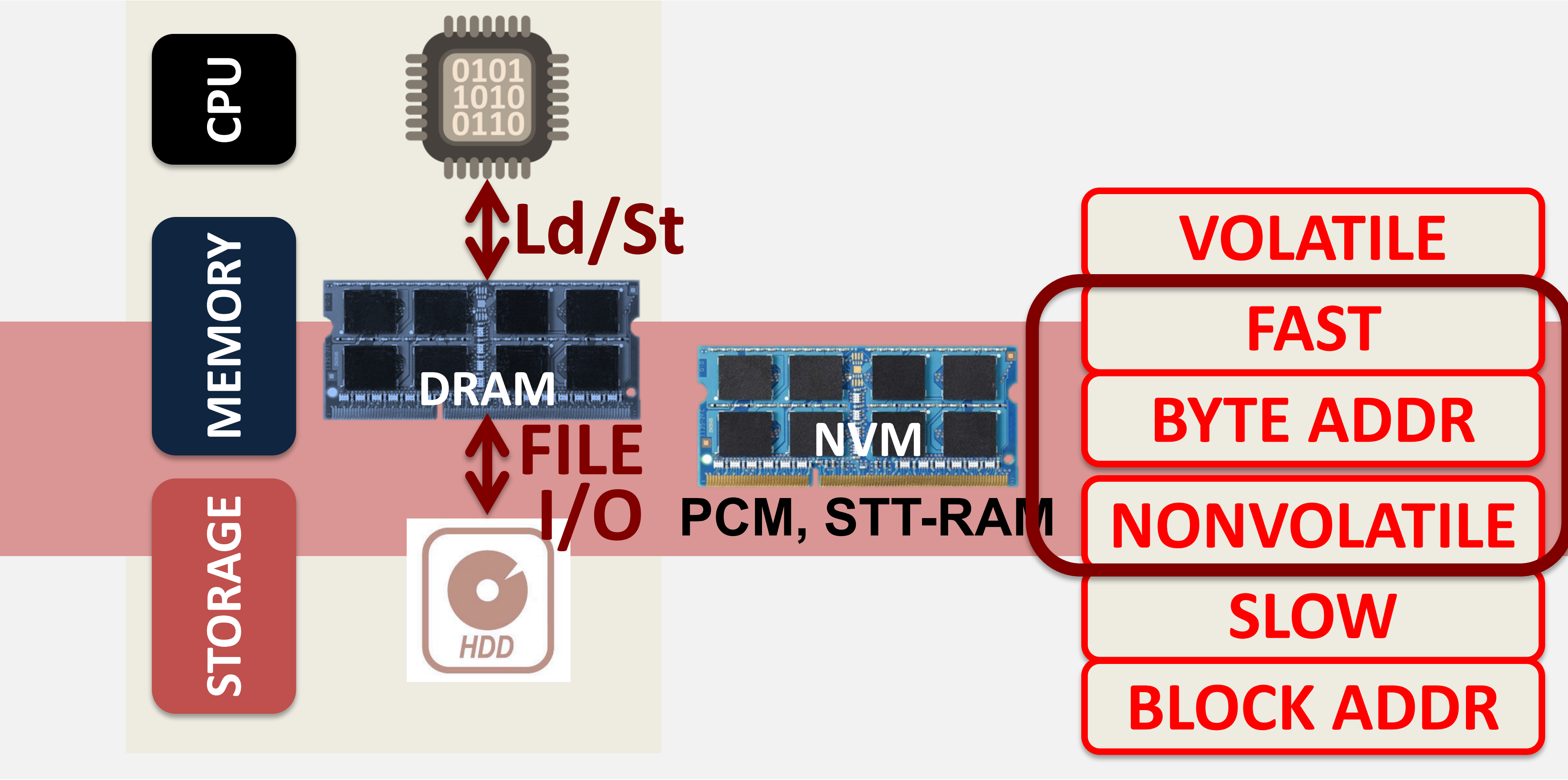
BYTE ADDR

NONVOLATILE

SLOW

BLOCK ADDR

TWO-LEVEL STORAGE MODEL



CPU

MEMORY

STORAGE

Ld/St

DRAM

FILE I/O

NVM

PCM, STT-RAM

VOLATILE

FAST

BYTE ADDR

NONVOLATILE

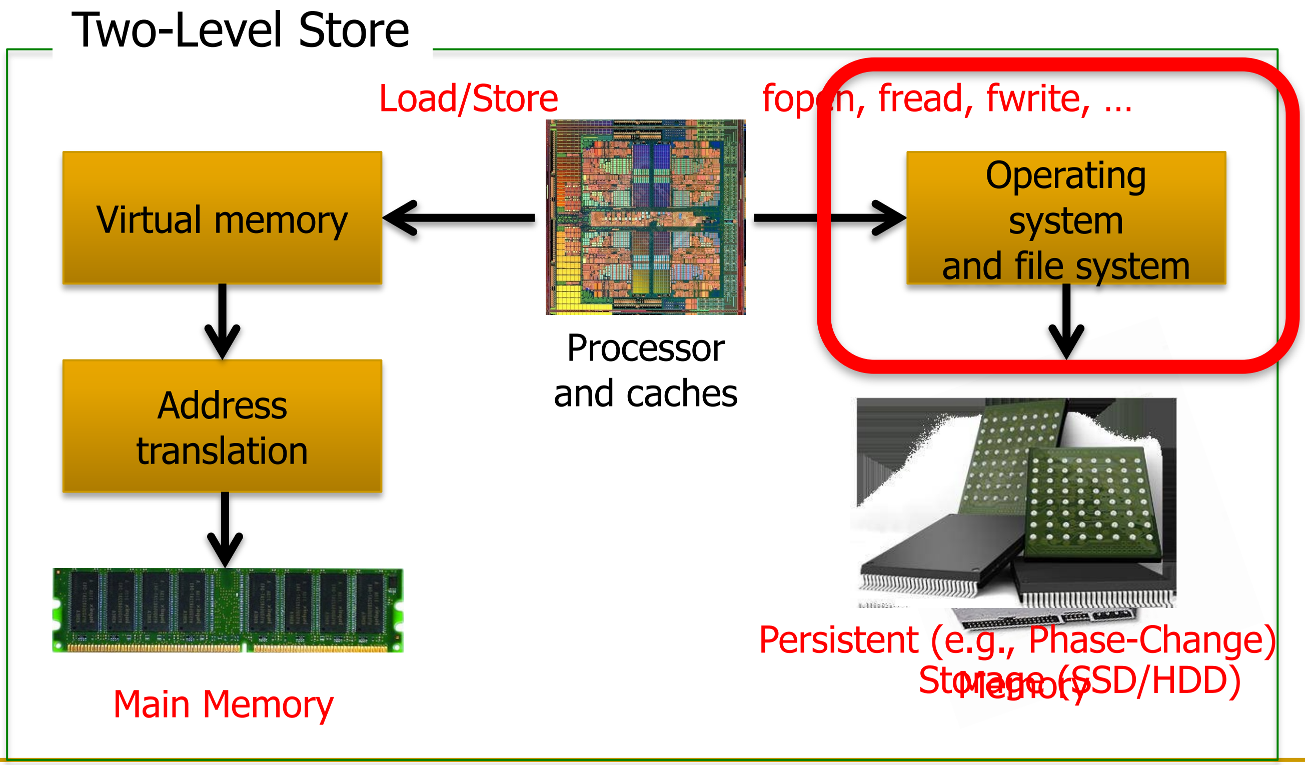
SLOW

BLOCK ADDR

Non-volatile memories combine characteristics of memory and storage

Two-Level Memory/Storage Model

- The traditional two-level storage model is a bottleneck with NVM
  - **Volatile** data in memory → a **load/store** interface
  - **Persistent** data in storage → a **file system** interface
  - Problem: Operating system (OS) and file system (FS) code to locate, translate, buffer data become performance and energy bottlenecks with fast NVM stores



Two-Level Store

Load/Store

fopen, fread, fwrite, …

Virtual memory

Address translation

Main Memory

Processor and caches

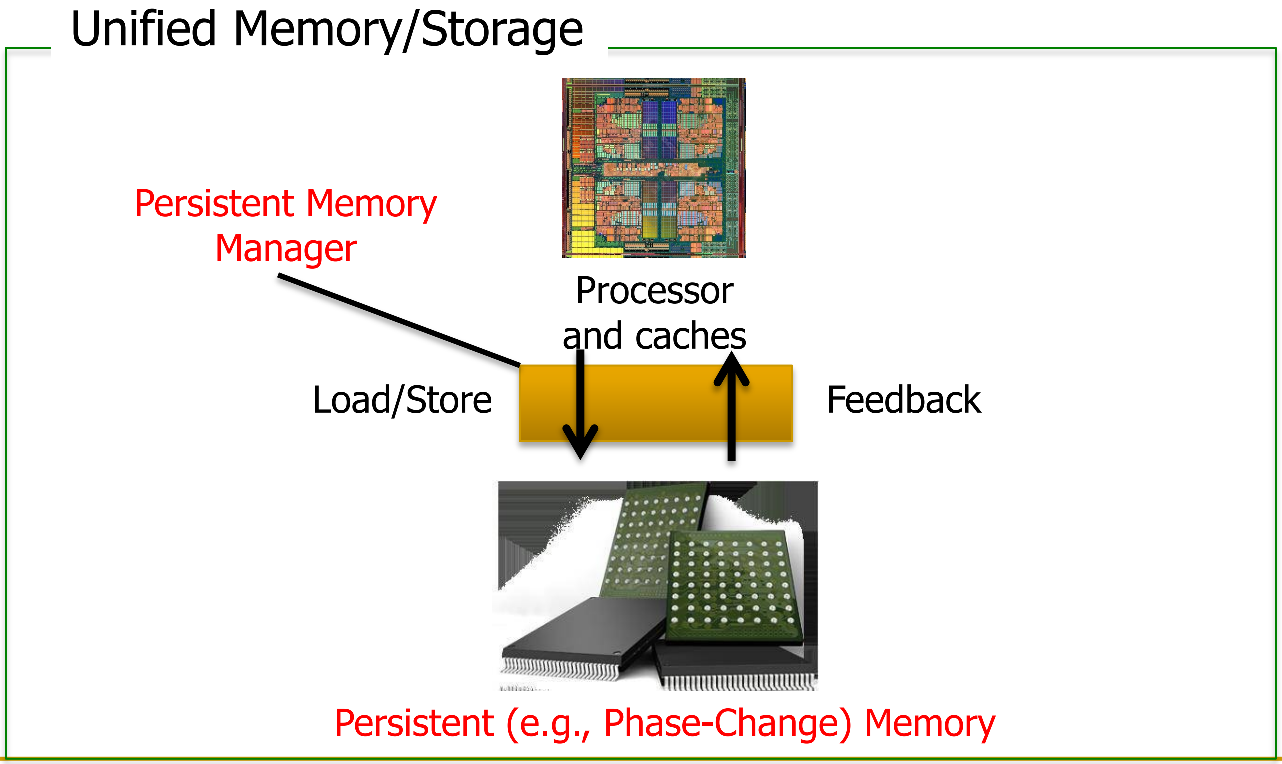
Operating system and file system

Persistent (e.g., Phase-Change) Memory

Storage (SSD/HDD)

Unified Memory and Storage with NVM

- Goal: Unify memory and storage management in a single unit to eliminate wasted work to locate, transfer, and translate data
  - Improves both energy and performance
  - Simplifies programming model as well



Unified Memory/Storage

Persistent Memory Manager

Processor and caches

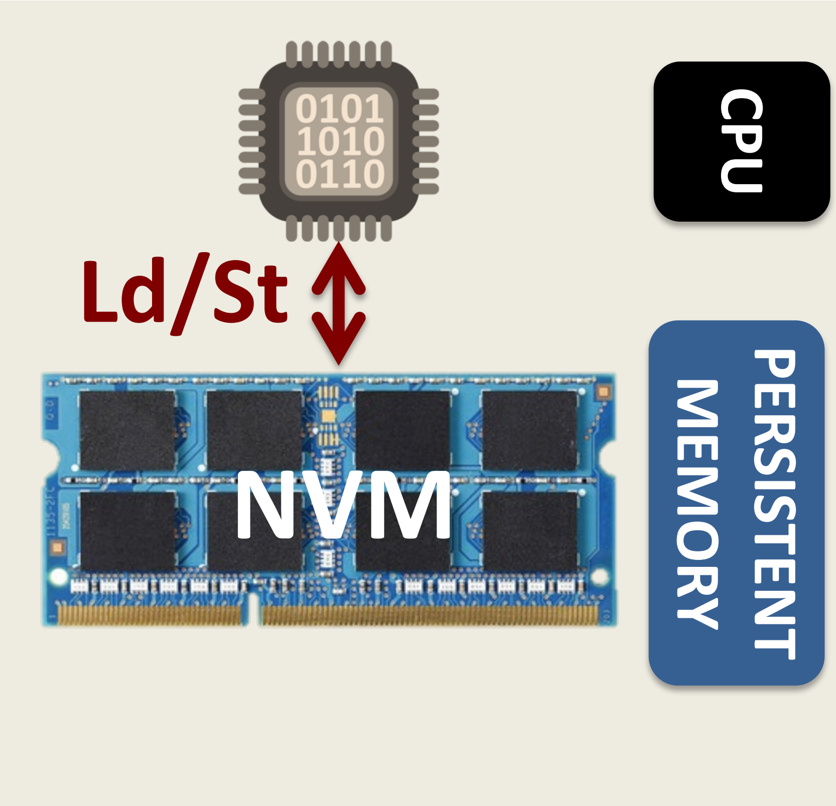
Load/Store

Feedback

Persistent (e.g., Phase-Change) Memory

Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.

PERSISTENT MEMORY

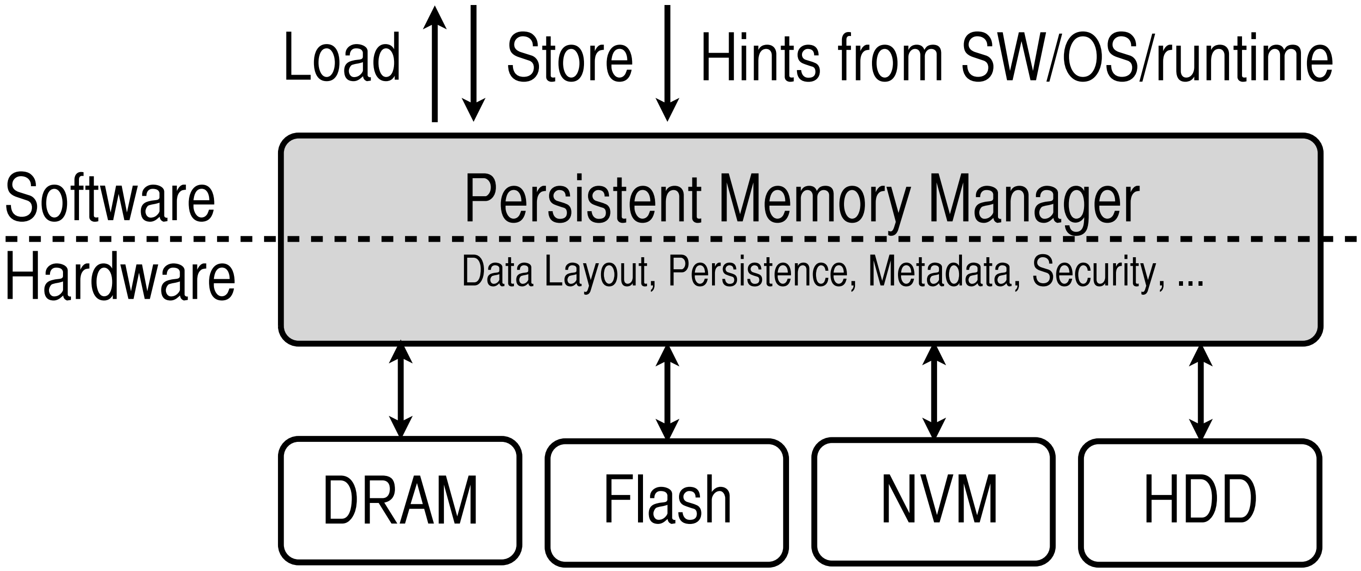


Provides an opportunity to manipulate persistent data directly

The Persistent Memory Manager (PMM)

```
int main(void) {
  // data in file.dat is persistent
  FILE myData = "file.dat";
  myData = new int[64];
}
void updateValue(int n, int value) {
  FILE myData = "file.dat";
  myData[n] = value; // value is persistent
}
```

Persistent objects


Load
Store
Hints from SW/OS/runtime
Software
Hardware
Persistent Memory Manager
Data Layout, Persistence, Metadata, Security, ...
DRAM
Flash
NVM
HDD

PMM uses access and hint information to allocate, locate, migrate and access data in the heterogeneous array of devices

The Persistent Memory Manager (PMM)

- Exposes a load/store interface to access persistent data
  - Applications can directly access persistent memory → no conversion, translation, location overhead for persistent data

- Manages data placement, location, persistence, security
  - To get the best of multiple forms of storage

- Manages metadata storage and retrieval
  - This can lead to overheads that need to be managed

- Exposes hooks and interfaces for system software
  - To enable better data placement and management decisions

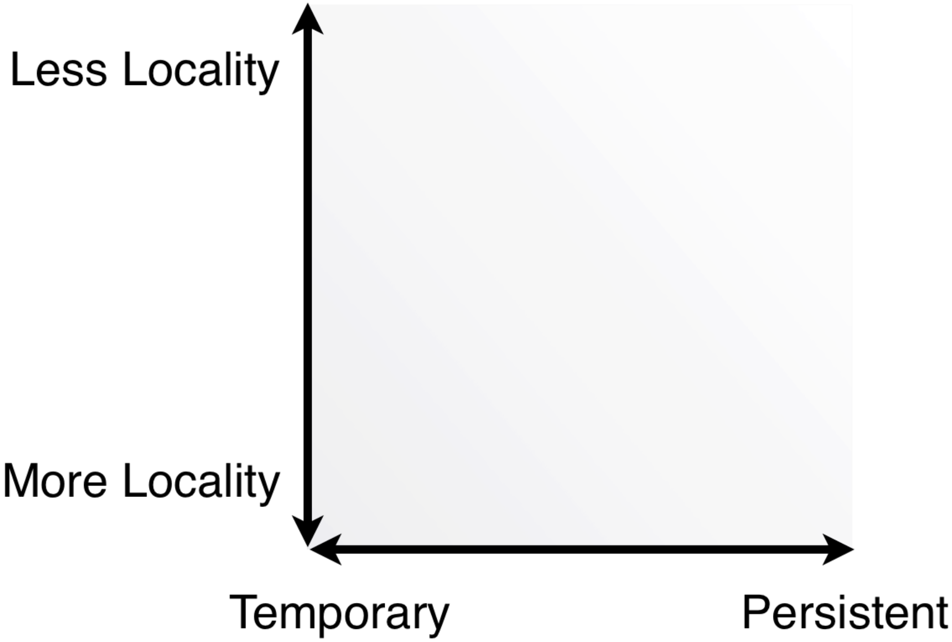
- Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.

Efficient Data Mapping among Heterogeneous Devices

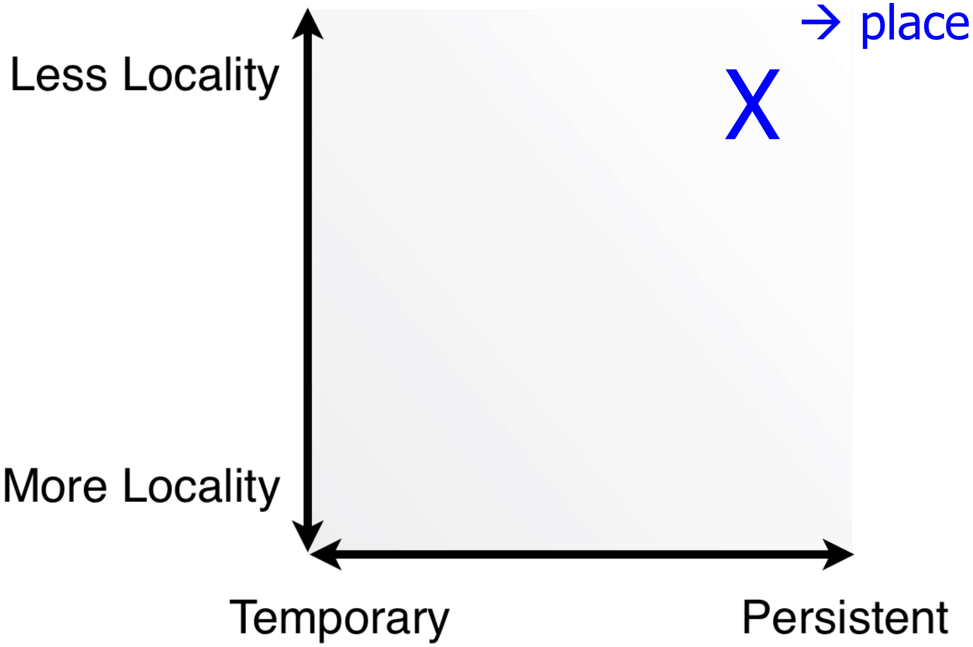
- A persistent memory exposes a large, persistent address space
  - But it may use many different devices to satisfy this goal
  - From fast, low-capacity volatile DRAM to slow, high-capacity non-volatile HDD or Flash
  - And other NVM devices in between

- Performance and energy can benefit from good placement of data among these devices
  - Utilizing the strengths of each device and avoiding their weaknesses, if possible
  - For example, consider two important application characteristics: locality and persistence

Efficient Data Mapping among Heterogeneous Devices

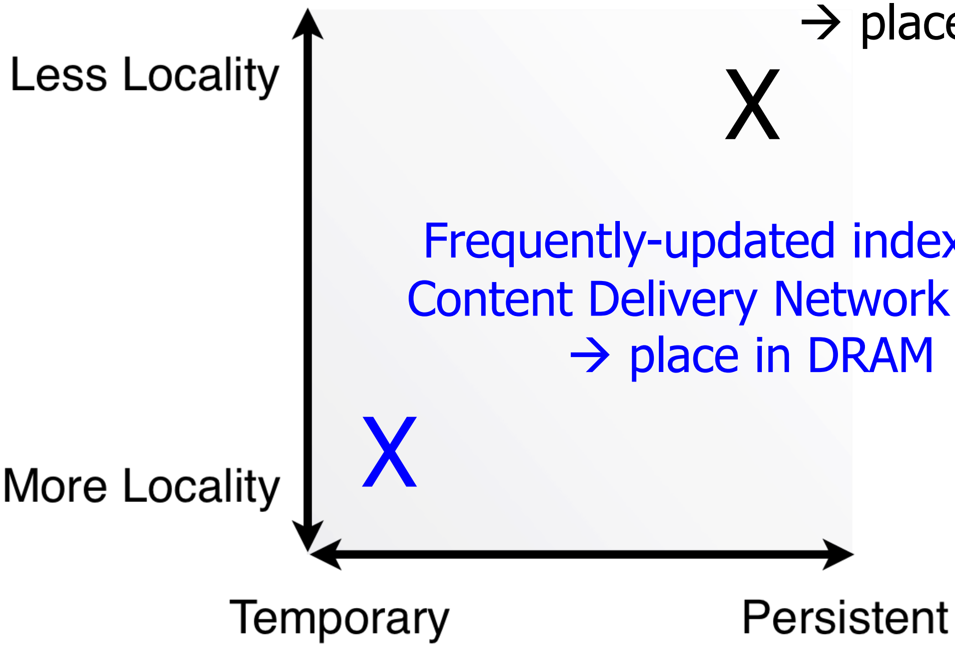


Efficient Data Mapping among Heterogeneous Devices



Columns in a column store that are scanned through only infrequently → place on Flash

Efficient Data Mapping among Heterogeneous Devices



Columns in a column store that are scanned through only infrequently
→ place on Flash

Frequently-updated index for a Content Delivery Network (CDN)
→ place in DRAM

Applications or system software can provide hints for data placement

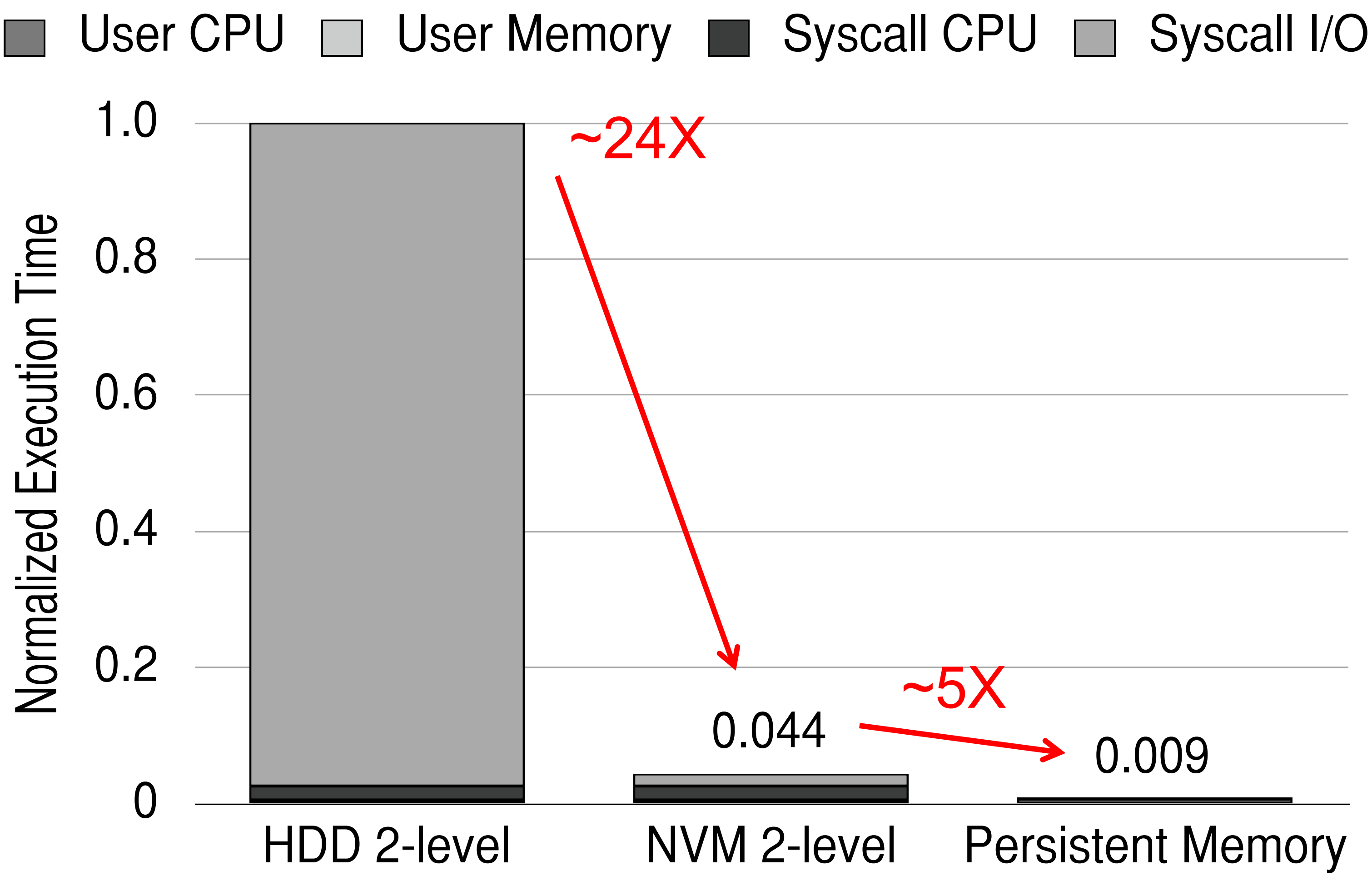
Evaluated Systems

- HDD Baseline
  - Traditional system with volatile DRAM memory and persistent HDD storage
  - Overheads of operating system and file system code and buffering

- NVM Baseline (NB)
  - Same as HDD Baseline, but HDD is replaced with NVM
  - Still has OS/FS overheads of the two-level storage model

- Persistent Memory (PM)
  - Uses only NVM (no DRAM) to ensure full-system persistence
  - All data accessed using loads and stores
  - Does not waste time on system calls
  - Data is manipulated directly on the NVM device

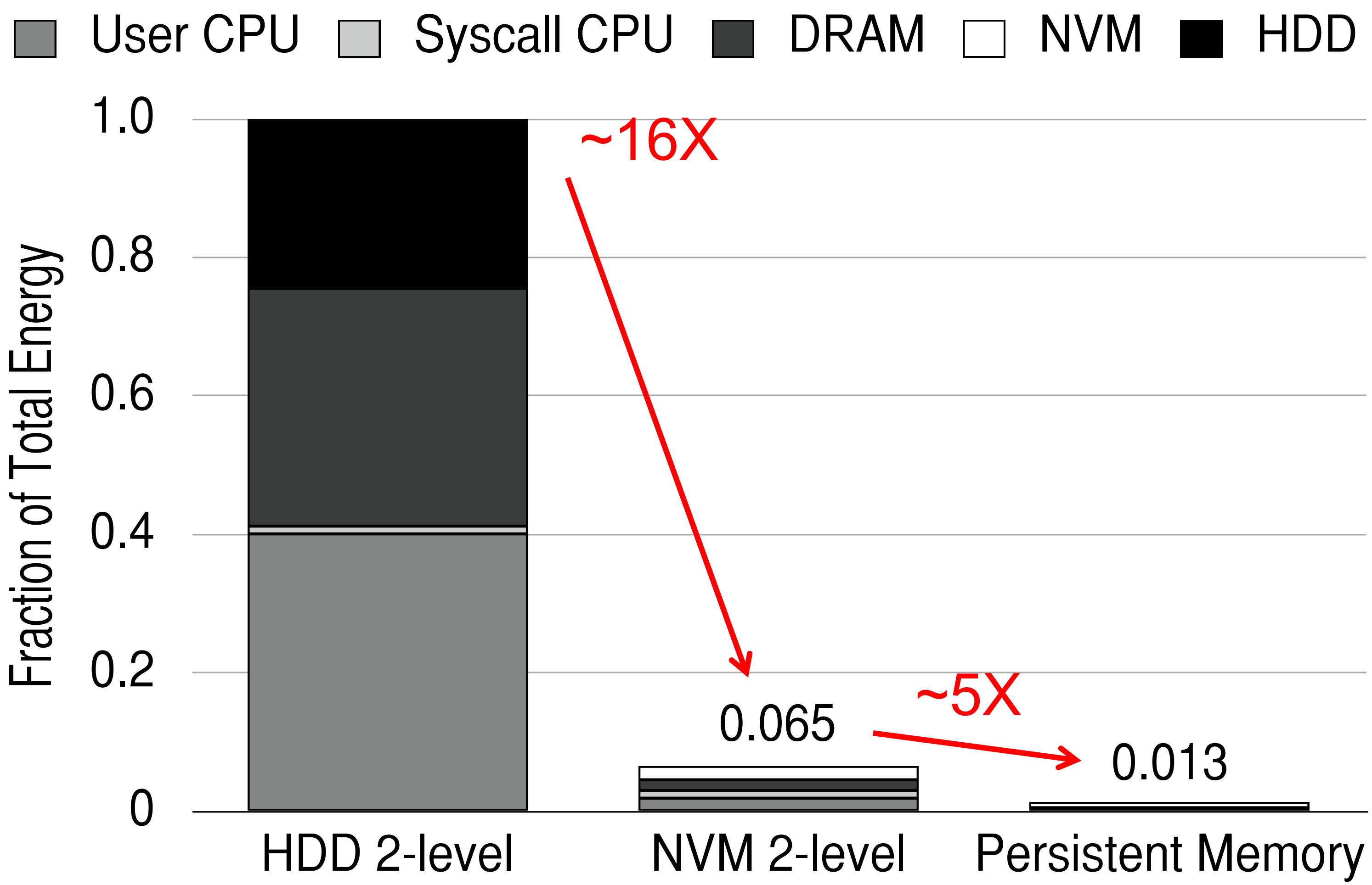
Performance Benefits of a Single-Level Store



SAFARI

Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.

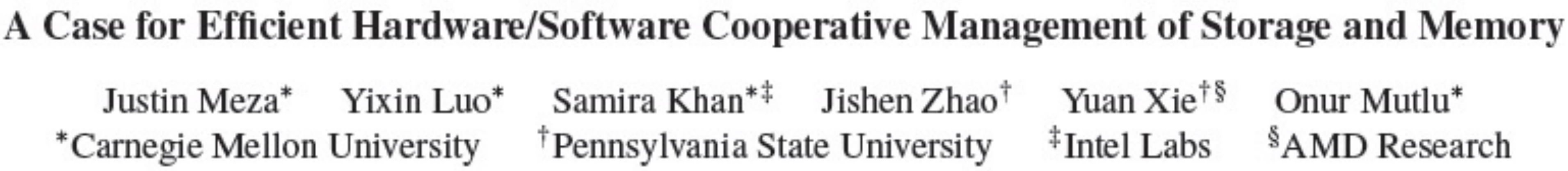
Energy Benefits of a Single-Level Store



Meza+, "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory," WEED 2013.

On Persistent Memory Benefits & Challenges

- Justin Meza, Yixin Luo, Samira Khan, Jishen Zhao, Yuan Xie, and Onur Mutlu, **"A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory"** *Proceedings of the 5th Workshop on Energy-Efficient Design* (***WEED***), Tel-Aviv, Israel, June 2013. Slides (pptx) Slides (pdf)



A Case for Efficient Hardware/Software Cooperative Management of Storage and Memory

Justin Meza* Yixin Luo* Samira Khan*‡ Jishen Zhao† Yuan Xie†§ Onur Mutlu*
*Carnegie Mellon University †Pennsylvania State University ‡Intel Labs §AMD Research

Challenge and Opportunity

Combined

Memory & Storage

Challenge and Opportunity

A Unified Interface to

All Data

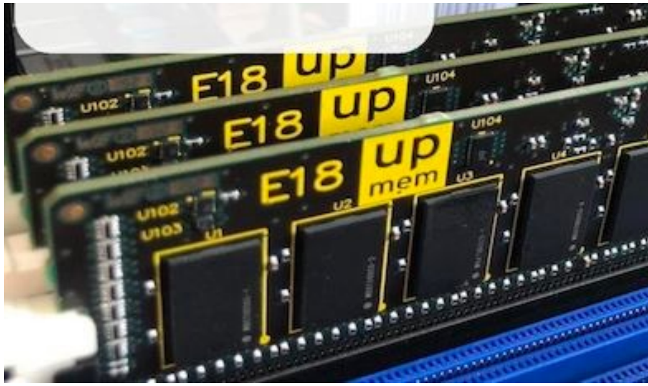
Intel Optane Persistent Memory (2019)

- Non-volatile main memory
- Based on 3D-XPoint Technology

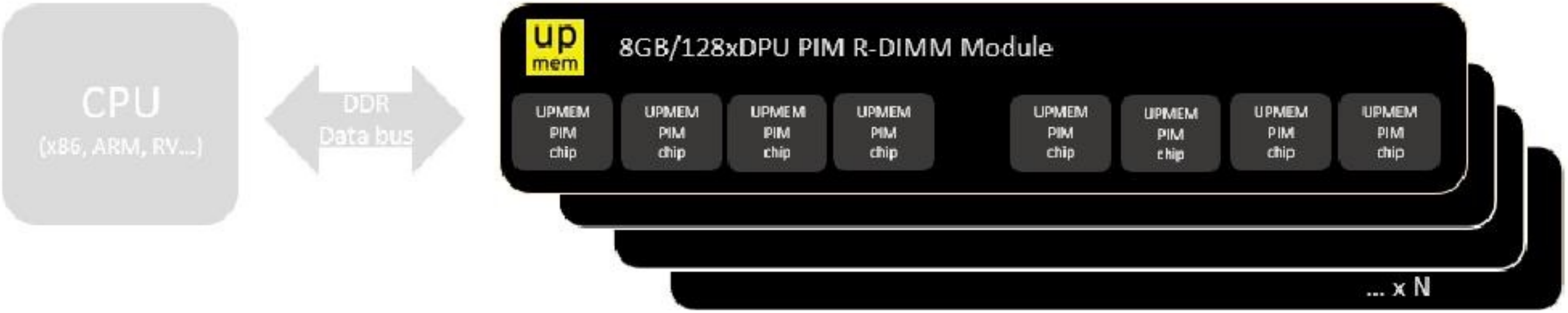
https://www.storagereview.com/intel_optane_dc_persistent_memory_module_pmm

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/

One Key Challenge in Persistent Memory

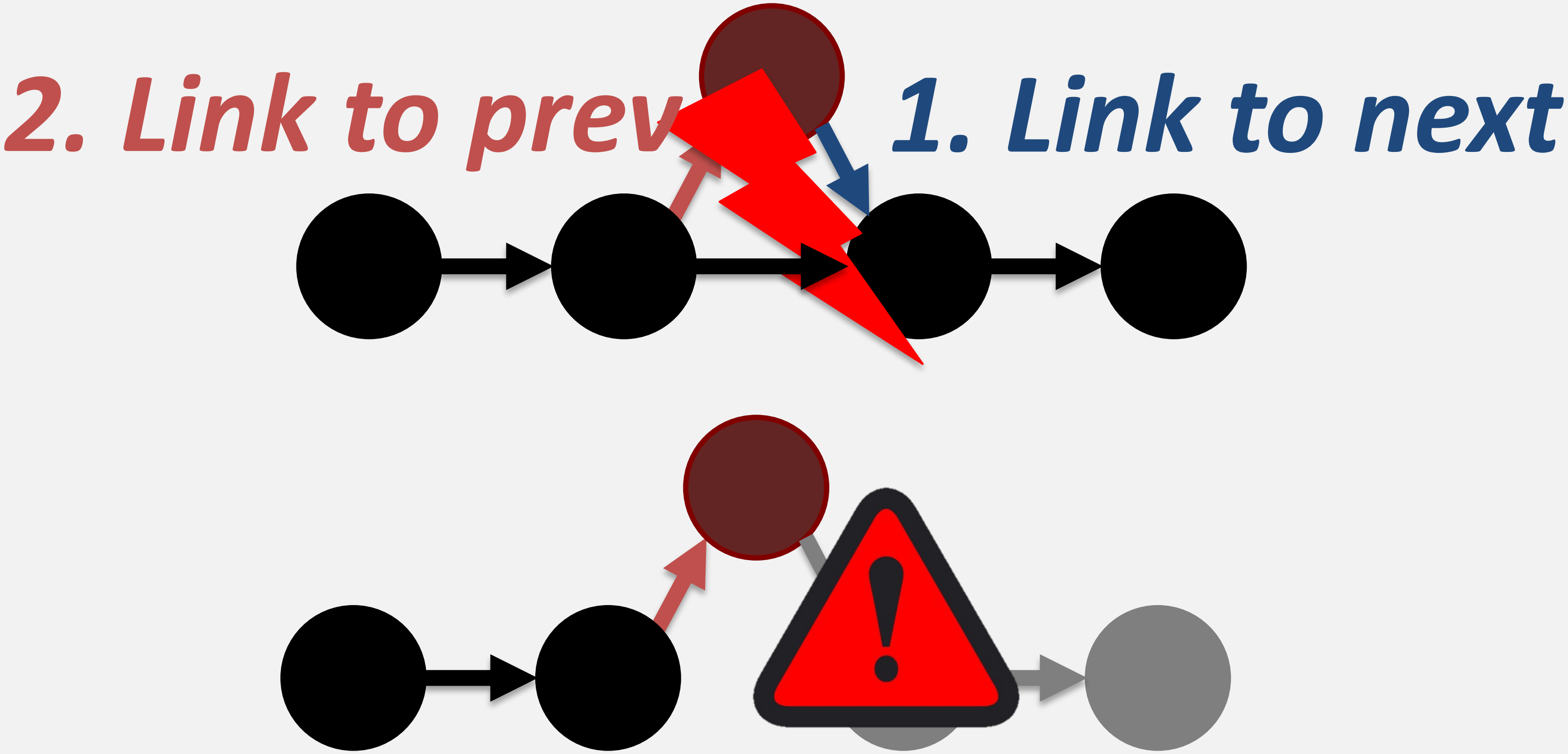
- How to ensure consistency of system/data if all memory is persistent?

- Two extremes
  - Programmer transparent: Let the system handle it
  - Programmer only: Let the programmer handle it

- Many alternatives in-between…

CRASH CONSISTENCY PROBLEM

Add a node to a linked list



System crash can result in inconsistent memory state

CURRENT SOLUTIONS

Explicit interfaces to manage consistency

- NV-Heaps [ASPLOS'11], BPFS [SOSP'09], Mnemosyne [ASPLOS'11]

```
AtomicBegin {
     Insert a new node;
} AtomicEnd;
```

Limits adoption of NVM

Have to rewrite code with clear partition between volatile and non-volatile data

Burden on the programmers

CURRENT SOLUTIONS

Explicit interfaces to manage consistency

– **NV-Heaps** [ASPLOS’11], **BPFS** [SOSP’09], **Mnemosyne** [ASPLOS’11]

Example Code

update a node in a persistent hash table

```
void hashtable_update(hashtable_t* ht,
               void *key, void *data)
{
   list_t* chain = get_chain(ht, key);
   pair_t* pair;
   pair_t updatePair;
   updatePair.first = key;
   pair = (pair_t*) list_find(chain,
                        &updatePair);
   pair->second = data;
}
```

CURRENT SOLUTIONS

```
void TMhashtable_update(TMARCGDECL
hashtable_t* ht, void *key,
void*data){
  list_t* chain = get_chain(ht, key);
  pair_t* pair;
  pair_t updatePair;
  updatePair.first = key;
  pair = (pair_t*) TMLIST_FIND(chain,
                            &updatePair);
  pair->second = data;
}
```

CURRENT SOLUTIONS

Manual declaration of persistent components

```
void TMhashtable_update(TMARCGDECL
hashtable_t* ht, void *key,
void*data){
   list_t* chain = get_chain(ht, key);
   pair_t* pair;
   pair_t updatePair;
   updatePair.first = key;
   pair = (pair_t*) TMLIST_FIND(chain,
                           &updatePair);
   pair->second = data;
}
```

CURRENT SOLUTIONS

Manual declaration of persistent components

```
void TMhashtable_update(TMARCGDECL
hashtable_t* ht, void *key,
void*data){
   list_t* chain = get_chain(ht, key);
   pair_t* pair;
   pair_t updatePair;
   updatePair.first = key;
   pair = (pair_t*) TMLIST_FIND(chain,
                      &updatePair);
   pair->second = data;
}
```

Need a new implementation

CURRENT SOLUTIONS

Manual declaration of persistent components

```
void TMhashtable_update(TMARCGDECL
hashtable_t* ht, void *key,
void*data){
   list_t* chain = get_chain(ht, key);
   pair_t* pair;
   pair_t updatePair;
   updatePair.first = key;
   pair = (pair_t*) TMLIST_FIND(chain,
                                &updatePair);
   pair->second = data;
}
```

Need a new implementation

Third party code
can be inconsistent

CURRENT SOLUTIONS

Manual declaration of persistent components

```
void TMhashtable_update(TMARCGDECL
hashtable_t* ht, void *key,
void*data){
  list_t* chain = get_chain(ht, key);
  pair_t* pair;
  pair_t updatePair;
  updatePair.first = key;
  pair = (pair_t*) TMLIST_FIND(chain,
                          &updatePair);
  pair->second = data;
}
```

Need a new implementation

Third party code can be inconsistent

Prohibited Operation

=

Burden on the programmers

OUR APPROACH: ThyNVM

Goal:
Software transparent consistency in persistent memory systems

Key Idea:
Periodically checkpoint state; recover to previous checkpt on crash

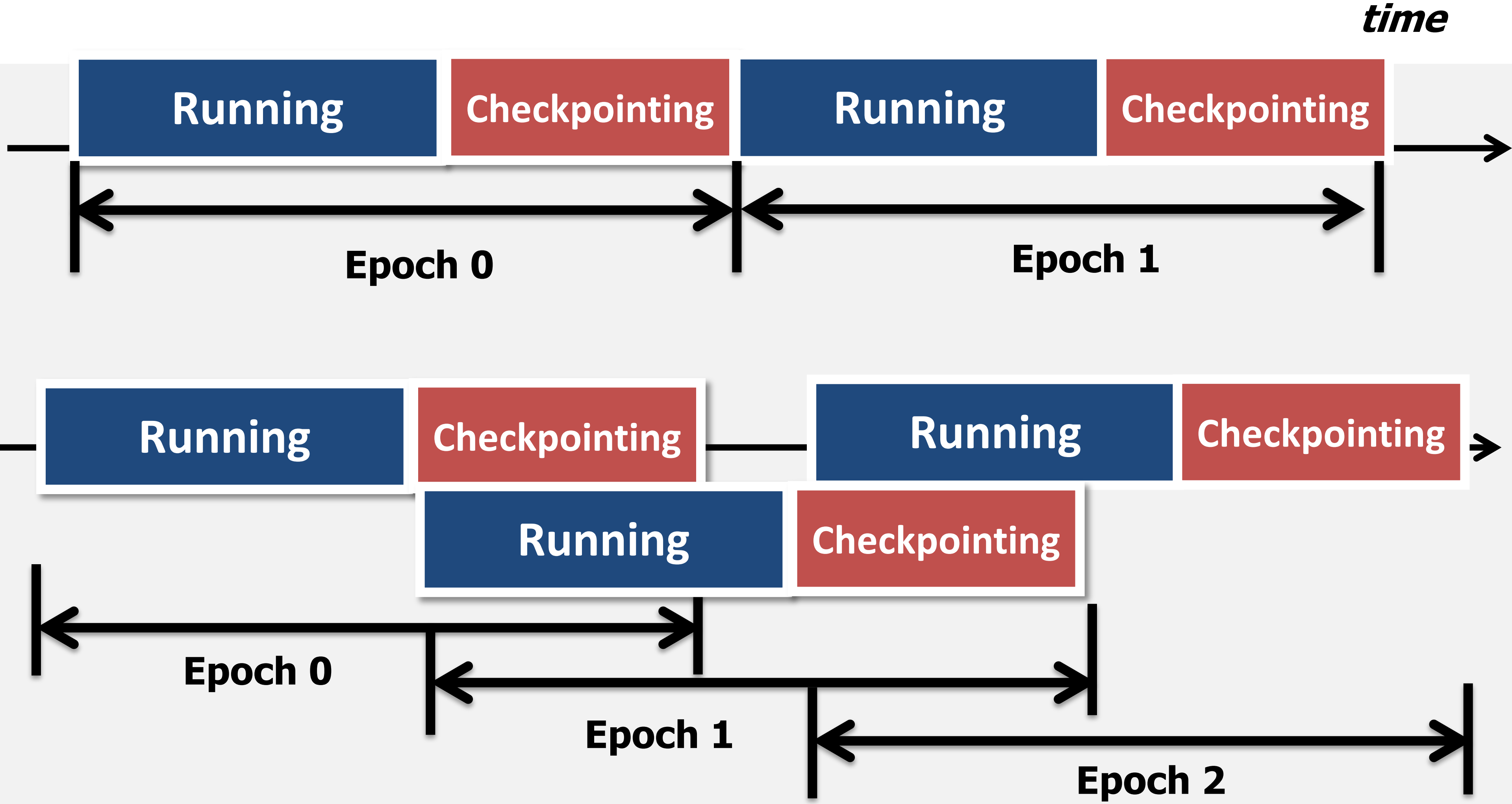
ThyNVM: Summary

A new hardware-based checkpointing mechanism

- **Checkpoints** at *multiple granularities* to reduce both checkpointing latency and metadata overhead
- **Overlaps** *checkpointing* and *execution to* reduce checkpointing latency
- **Adapts** to *DRAM and NVM* characteristics

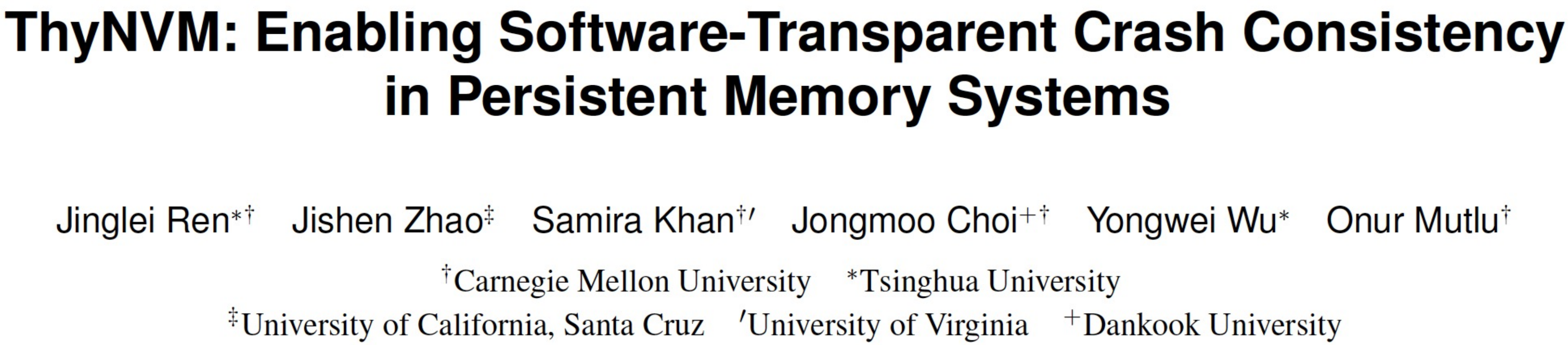
Performs within 4.9% of an *idealized DRAM* with zero cost consistency

2. OVERLAPPING CHECKPOINTING AND EXECUTION



More About ThyNVM

- Jinglei Ren, Jishen Zhao, Samira Khan, Jongmoo Choi, Yongwei Wu, and Onur Mutlu,
  "ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems"
  Proceedings of the 48th International Symposium on Microarchitecture (***MICRO***), Waikiki, Hawaii, USA, December 2015.
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
  [Source Code]



ThyNVM: Enabling Software-Transparent Crash Consistency in Persistent Memory Systems

Jinglei Ren[*†] Jishen Zhao[‡] Samira Khan[†′] Jongmoo Choi[+†] Yongwei Wu[*] Onur Mutlu[†]

[†]Carnegie Mellon University [*]Tsinghua University
[‡]University of California, Santa Cruz [′]University of Virginia [+]Dankook University

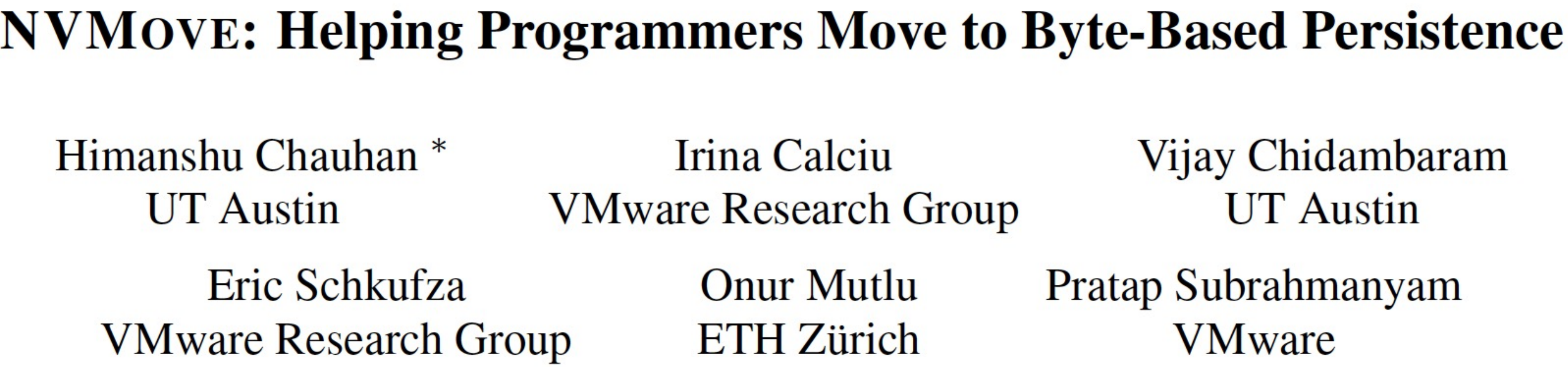
Another Key Challenge in Persistent Memory

Programming Ease

to Exploit Persistence

Tools/Libraries to Help Programmers

- Himanshu Chauhan, Irina Calciu, Vijay Chidambaram, Eric Schkufza, Onur Mutlu, and Pratap Subrahmanyam,
"NVMove: Helping Programmers Move to Byte-Based Persistence"
*Proceedings of the 4th Workshop on Interactions of NVM/Flash with Operating Systems and Workloads (**INFLOW**)*, Savannah, GA, USA, November 2016.
[Slides (pptx) (pdf)]



NVMove: Helping Programmers Move to Byte-Based Persistence

Himanshu Chauhan *
UT Austin

Irina Calciu
VMware Research Group

Vijay Chidambaram
UT Austin

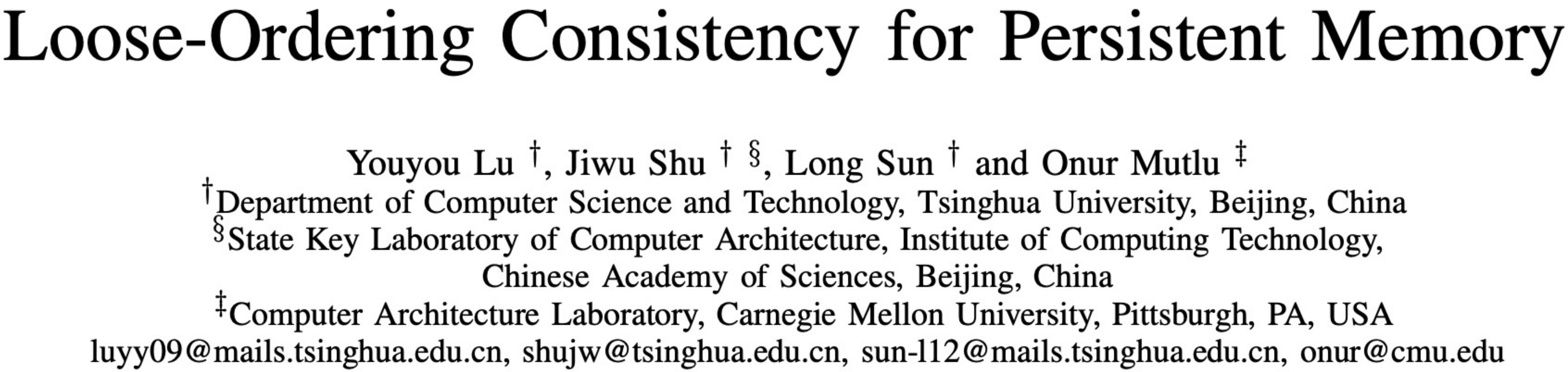
Eric Schkufza
VMware Research Group

Onur Mutlu
ETH Zürich

Pratap Subrahmanyam
VMware

Consistency Support for Persistent Memory

- Youyou Lu, Jiwu Shu, Long Sun, and Onur Mutlu,
  "Loose-Ordering Consistency for Persistent Memory"
  *Proceedings of the 32nd IEEE International Conference on Computer Design (**ICCD**)*, Seoul, South Korea, October 2014.
  [Slides (pptx) (pdf)]
  [Erratum]



Loose-Ordering Consistency for Persistent Memory

Youyou Lu[†], Jiwu Shu[†][§], Long Sun[†] and Onur Mutlu[‡]
[†]Department of Computer Science and Technology, Tsinghua University, Beijing, China
[§]State Key Laboratory of Computer Architecture, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
[‡]Computer Architecture Laboratory, Carnegie Mellon University, Pittsburgh, PA, USA
luyy09@mails.tsinghua.edu.cn, shujw@tsinghua.edu.cn, sun-l12@mails.tsinghua.edu.cn, onur@cmu.edu

Another Key Challenge in Persistent Memory

Security and
Data Privacy Issues

Security and Privacy Issues of NVM

- Endurance problems → Wearout attacks

- Hybrid memories → Performance attacks

- Data not erased after power-off → Privacy breaches

Conclusion

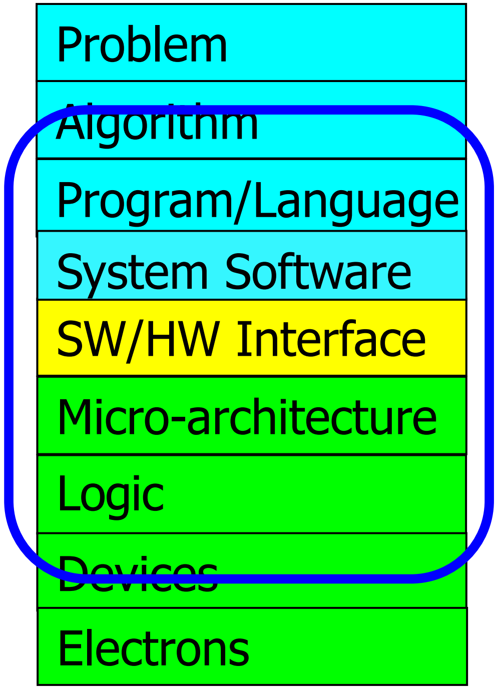
The Future of Emerging Technologies is Bright

- Regardless of challenges
  - in underlying technology and overlying problems/requirements

Can enable:

- Orders of magnitude improvements

- New applications and computing systems



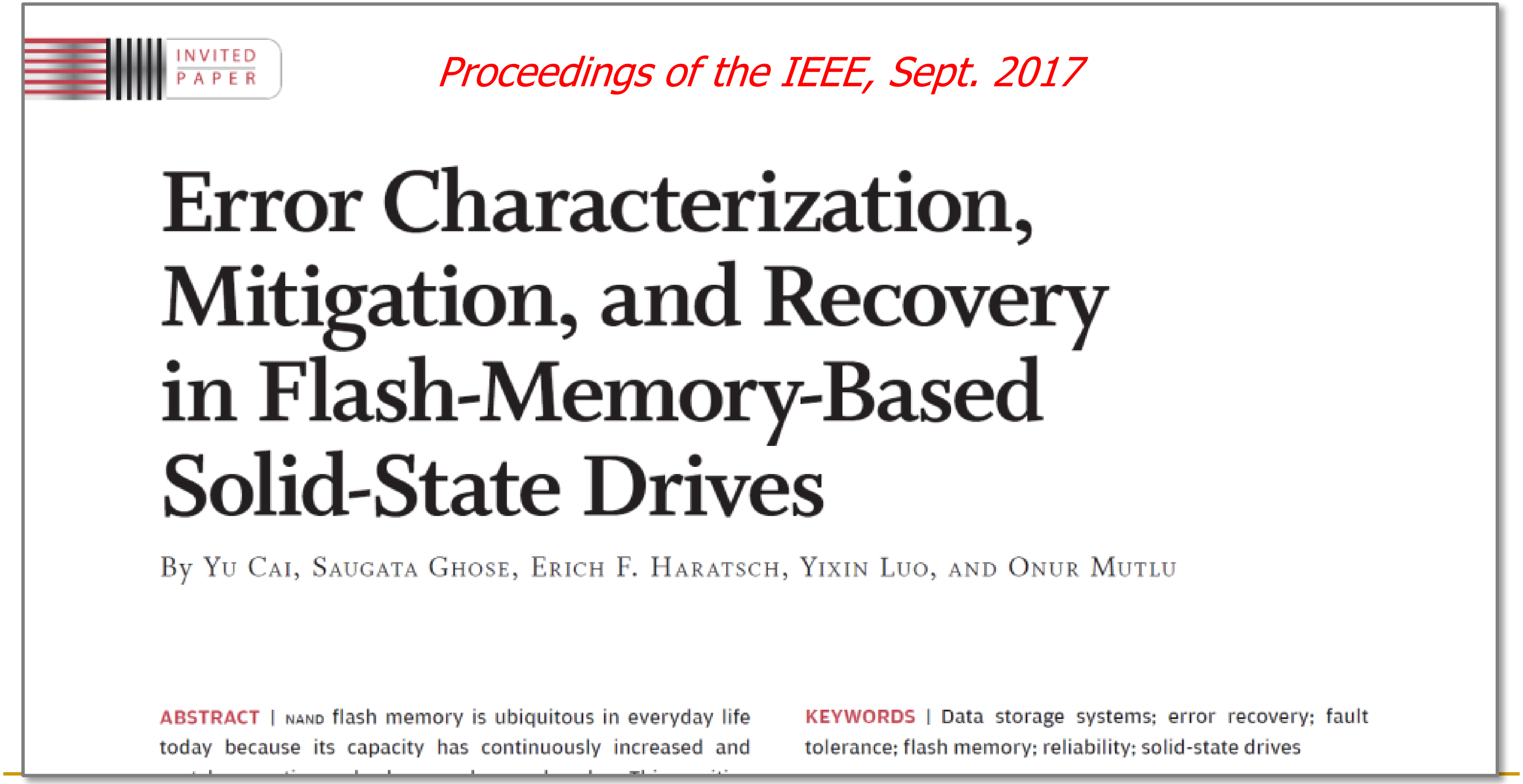
Yet, we have to

- Think across the stack

- Design enabling systems

If In Doubt, Refer to Flash Memory

- A very "doubtful" emerging technology
  - for at least two decades



INVITED PAPER

Proceedings of the IEEE, Sept. 2017

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

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

ABSTRACT | NAND flash memory is ubiquitous in everyday life today because its capacity has continuously increased and

KEYWORDS | Data storage systems; error recovery; fault tolerance; flash memory; reliability; solid-state drives

https://arxiv.org/pdf/1706.08642

Many Research & Design Opportunities

- Enabling completely persistent memory
- Computation in/using NVM based memories
- Hybrid memory systems
- Security and privacy issues in persistent memory
- Reliability and endurance related problems
- Virtual memory systems for NVM → virtual block interface
- …

Computer Architecture

Lecture 17a: Emerging Memory Technologies II

Prof. Onur Mutlu

ETH Zürich

Fall 2021

25 November 2021