Computer Architecture

Lecture 11a: Memory Controllers

Prof. Onur Mutlu
ETH Zürich
Fall 2022
3 November 2022

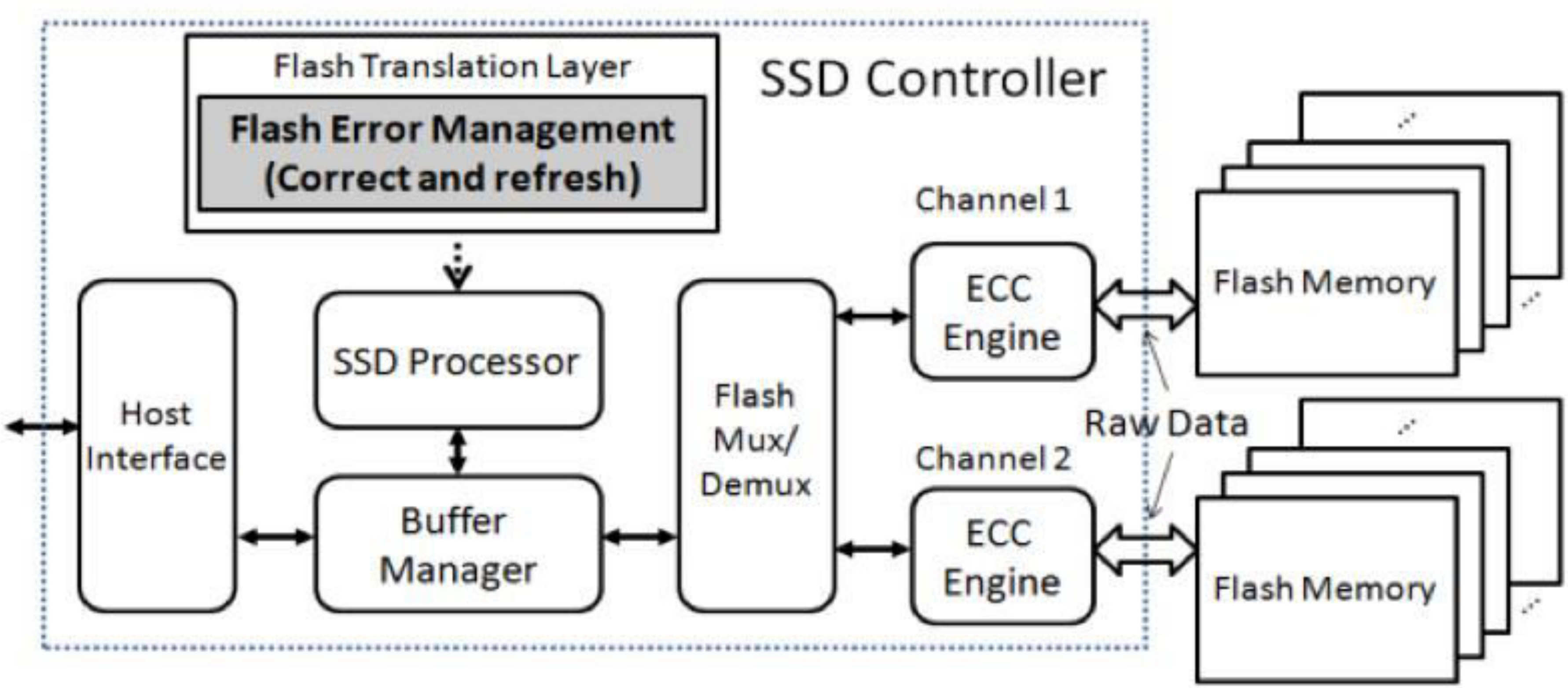
DRAM versus Other Types of Memories

- Long latency memories have similar characteristics that need to be controlled.

- This lecture will use DRAM as an example, but many scheduling and control issues are similar in the design of controllers for other types of memories
  - Flash memory
  - Other emerging memory technologies
    - Phase Change Memory
    - Spin-Transfer Torque Magnetic Memory
  - These other technologies can also place other demands on the controller

Flash Memory (SSD) Controllers

- Similar to DRAM memory controllers, except:
  - They are flash memory specific
  - They do much more: complex error correction, wear leveling, voltage optimization, garbage collection, page remapping, …



Cai+, "Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime", ICCD 2012.

Another View of the SSD Controller

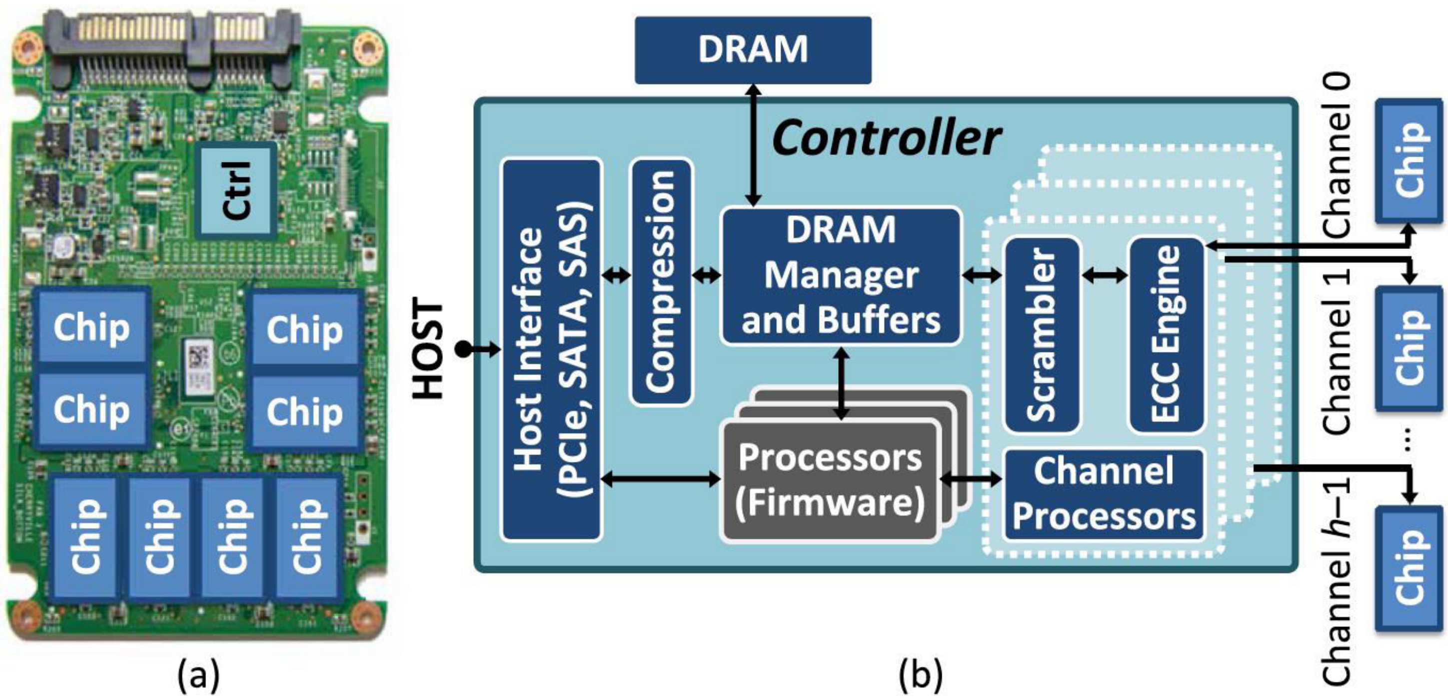


Fig. 1. ***(a) SSD system architecture, showing controller (Ctrl) and chips. (b) Detailed view of connections between controller components and chips.***

Cai+, "Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives," Proc. IEEE 2017.

https://arxiv.org/pdf/1711.11427.pdf

On Modern SSD Controllers (I)

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

Many Errors and Their Mitigation **[PIEEE'17]**

Table 3 List of Different Types of Errors Mitigated by NAND Flash Error Mitigation Mechanisms

Mitigation Mechanism	Error Type: *P/E Cycling* [32,33,42] (§IV-A)	*Program* [40,42,53] (§IV-B)	*Cell-to-Cell Interference* [32,35,36,55] (§IV-C)	*Data Retention* [20,32,34,37,39] (§IV-D)	*Read Disturb* [20,32,38,62] (§IV-E)
Shadow Program Sequencing [35,40] (Section V-A)			X		
Neighbor-Cell Assisted Error Correction [36] (Section V-B)			X		
Refresh [34,39,67,68] (Section V-C)				X	X
Read-Retry [33,72,107] (Section V-D)	X			X	X
Voltage Optimization [37,38,74] (Section V-E)	X			X	X
Hot Data Management [41,63,70] (Section V-F)	X	X	X	X	X
Adaptive Error Mitigation [43,65,77,78,82] (Section V-G)	X	X	X	X	X

Cai+, "Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives," Proc. IEEE 2017.

More Up-to-date Version

- Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu, **"Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery"**
  Invited Book Chapter in Inside Solid State Drives, 2018.
  [Preliminary arxiv.org version]

Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery

YU CAI, SAUGATA GHOSE
Carnegie Mellon University

ERICH F. HARATSCH
Seagate Technology

YIXIN LUO
Carnegie Mellon University

ONUR MUTLU
ETH Zürich and Carnegie Mellon University

On Modern SSD Controllers (II)

- Arash Tavakkol, Juan Gomez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu,
  "MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices"
  *Proceedings of the 16th USENIX Conference on File and Storage Technologies (**FAST**)*, Oakland, CA, USA, February 2018.
  [Slides (pptx) (pdf)]
  [Source Code]

MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices

Arash Tavakkol[†], Juan Gómez-Luna[†], Mohammad Sadrosadati[†], Saugata Ghose[‡], Onur Mutlu[†‡]
[†]*ETH Zürich* [‡]*Carnegie Mellon University*

On Modern SSD Controllers (III)

- Arash Tavakkol, Mohammad Sadrosadati, Saugata Ghose, Jeremie Kim, Yixin Luo, Yaohua Wang, Nika Mansouri Ghiasi, Lois Orosa, Juan G. Luna and Onur Mutlu,
"FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives"
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]

FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives

Arash Tavakkol[†] Mohammad Sadrosadati[†] Saugata Ghose[‡] Jeremie S. Kim[‡†] Yixin Luo[‡]
Yaohua Wang[†§] Nika Mansouri Ghiasi[†] Lois Orosa[†*] Juan Gómez-Luna[†] Onur Mutlu[†‡]
[†]*ETH Zürich* [‡]*Carnegie Mellon University* [§]*NUDT* [*]*Unicamp*

On Modern SSD Controllers (IV)

- Myungsuk Kim, Jisung Park, Geonhee Cho, Yoona Kim, Lois Orosa, Onur Mutlu, and Jihong Kim,
  "Evanesco: Architectural Support for Efficient Data Sanitization in Modern Flash-Based Storage Systems"
  *Proceedings of the 25th International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Lausanne, Switzerland, March 2020.
  [Slides (pptx) (pdf)]
  [Talk Video (20 mins)]

Evanesco: Architectural Support for Efficient Data Sanitization in Modern Flash-Based Storage Systems

Myungsuk Kim*
morssola75@davinci.snu.ac.kr
Seoul National University

Jisung Park*
jisung.park@inf.ethz.ch
ETH Zürich & Seoul National University

Geonhee Cho
ghcho@davinci.snu.ac.kr
Seoul National University

Yoona Kim
yoonakim@davinci.snu.ac.kr
Seoul National University

Lois Orosa
lois.orosa@inf.ethz.ch
ETH Zürich

Onur Mutlu
omutlu@gmail.com
ETH Zürich

Jihong Kim
jihong@davinci.snu.ac.kr
Seoul National University

On Modern SSD Controllers (V)

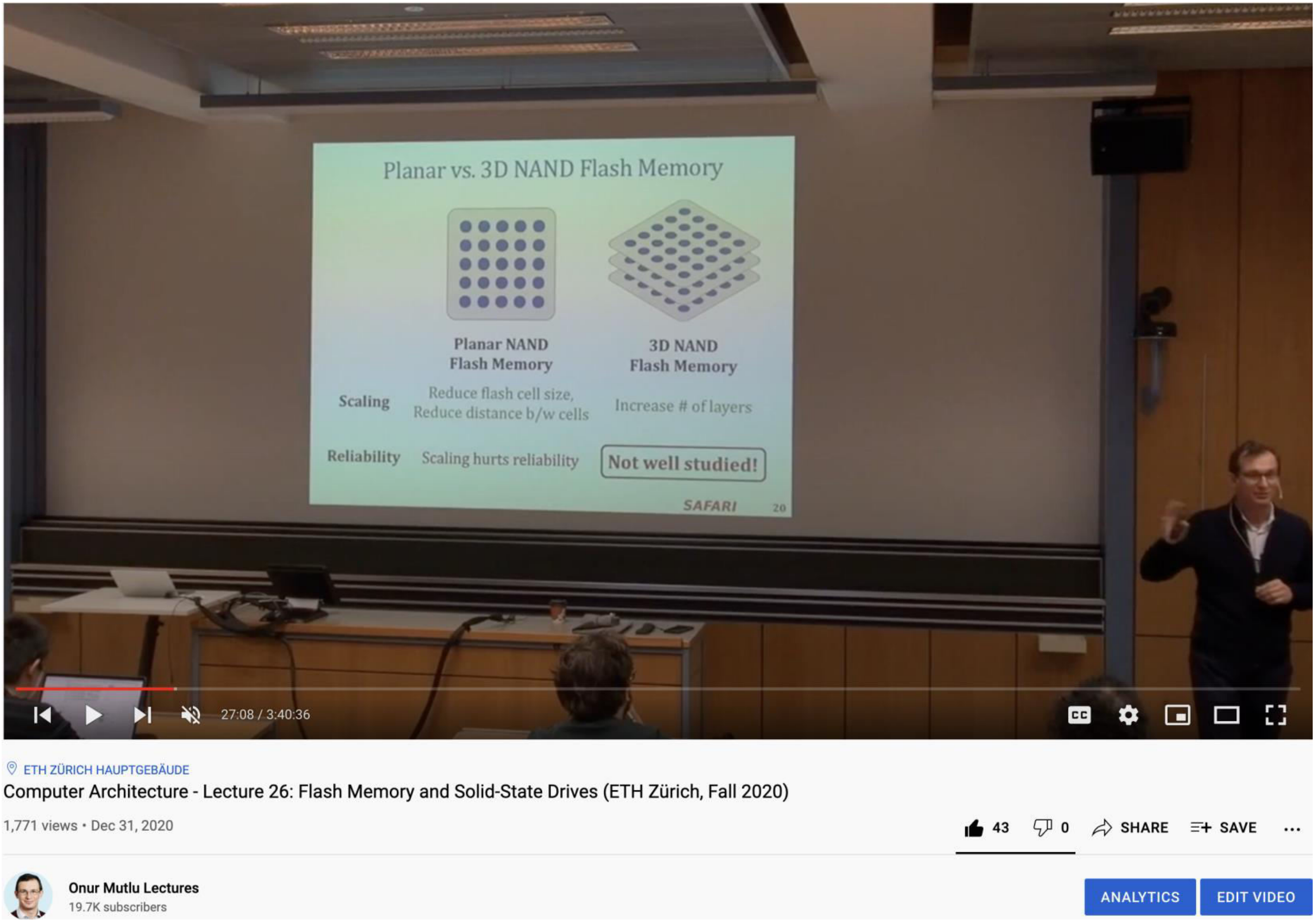
- Jisung Park, Myungsuk Kim, Myoungjun Chun, Lois Orosa, Jihong Kim, and Onur Mutlu,
  "Reducing Solid-State Drive Read Latency by Optimizing Read-Retry"
  *Proceedings of the 26th International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Virtual, March-April 2021.
  [2-page Extended Abstract]
  [Short Talk Slides (pptx) (pdf)]
  [Full Talk Slides (pptx) (pdf)]
  [Short Talk Video (5 mins)]
  [Full Talk Video (19 mins)]

Reducing Solid-State Drive Read Latency by Optimizing Read-Retry

Jisung Park[1] Myungsuk Kim[2,3] Myoungjun Chun[2] Lois Orosa[1] Jihong Kim[2] Onur Mutlu[1]

[1]ETH Zürich Switzerland [2]Seoul National University Republic of Korea [3]Kyungpook National University Republic of Korea

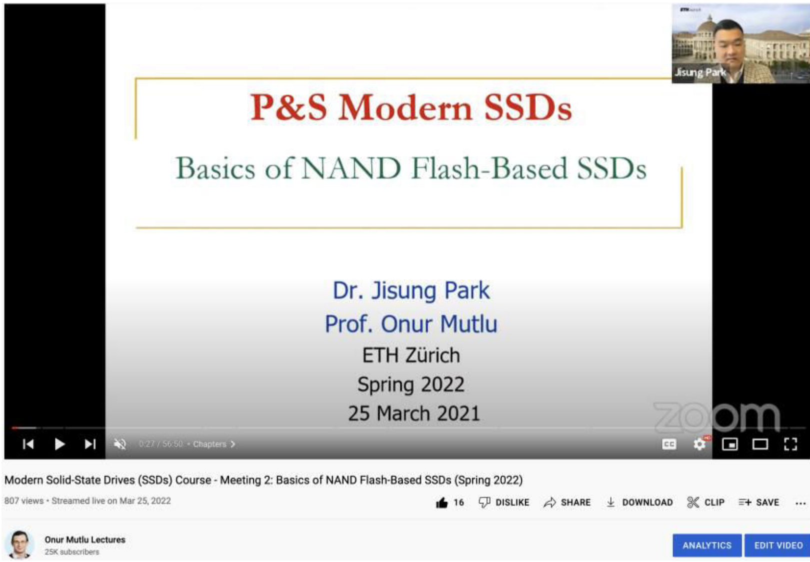
Lecture on Flash Memory & SSDs



Special Course on Flash Memory & SSDs

Solid-State Drives Course (Spring 2022)

- **Spring 2022 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=modern_ssds
- **Youtube Livestream:**
  - https://www.youtube.com/watch?v=_q4rm71DsY4&list=PL5Q2soXY2Zi8vabcse1kL22DEcgMl2RAq
- Project course
  - Taken by Bachelor's/Master's students
  - SSD Basics and Advanced Topics
  - Hands-on research exploration
  - Many research readings



https://www.youtube.com/onurmutlulectures

DRAM Types

- DRAM has different types with different interfaces optimized for different purposes
  - Commodity: DDR, DDR2, DDR3, DDR4, DDR5, …
  - Low power (for mobile): LPDDR1, …, LPDDR5, …
  - High bandwidth (for graphics): GDDR2, …, GDDR5, …
  - Low latency: eDRAM, RLDRAM, …
  - 3D stacked: WIO, HBM, HMC, HBM2.0, …
  - …
- Underlying microarchitecture is fundamentally the same
- A flexible memory controller can support various DRAM types
- This complicates the memory controller
  - Difficult to support all types (and upgrades)
  - Analog interface is different for different DRAM types

DRAM Types (circa 2015)

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

Table 1. Landscape of DRAM-based memory

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

Modern DRAM Types: Comparison to DDR3

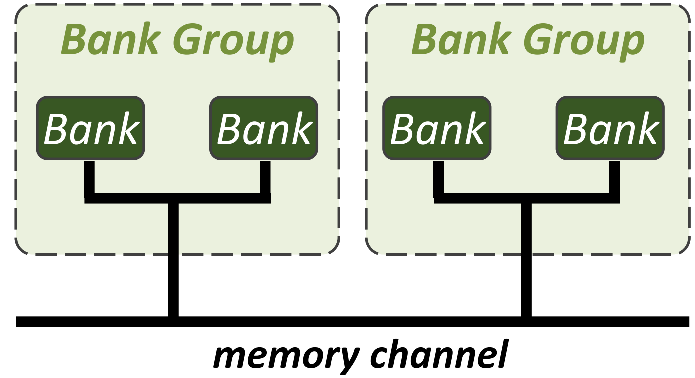
DRAM Type	Banks per Rank	Bank Groups	3D-Stacked	Low-Power
DDR3	8			
DDR4	16	✓		
GDDR5	16	✓		
HBM High-Bandwidth Memory	16		✓	
HMC Hybrid Memory Cube	256		✓	
Wide I/O	4		✓	✓
Wide I/O 2	8		✓	✓
LPDDR3	8			✓
LPDDR4	16			✓

increased latency

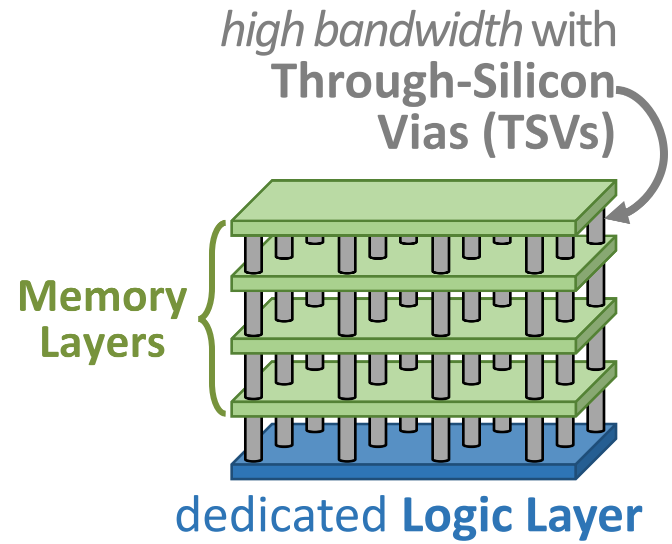
increased area/power

narrower rows, higher latency

- Bank groups



- 3D-stacked DRAM



Ramulator Paper and Source Code

- Yoongu Kim, Weikun Yang, and Onur Mutlu,
"Ramulator: A Fast and Extensible DRAM Simulator"
IEEE Computer Architecture Letters ***(CAL)***, March 2015.
[Source Code]

- Source code is released under the liberal MIT License
  - https://github.com/CMU-SAFARI/ramulator

Ramulator: A Fast and Extensible DRAM Simulator

Yoongu Kim[1] Weikun Yang[1,2] Onur Mutlu[1]
[1]Carnegie Mellon University [2]Peking University

DRAM Types vs. Workloads

- Saugata Ghose, Tianshi Li, Nastaran Hajinazar, Damla Senol Cali, and Onur Mutlu, **"Demystifying Workload–DRAM Interactions: An Experimental Study"** *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (**SIGMETRICS**)*, Phoenix, AZ, USA, June 2019.
  [Preliminary arXiv Version]
  [Abstract]
  [Slides (pptx) (pdf)]
  [MemBen Benchmark Suite]
  [Source Code for GPGPUSim-Ramulator]

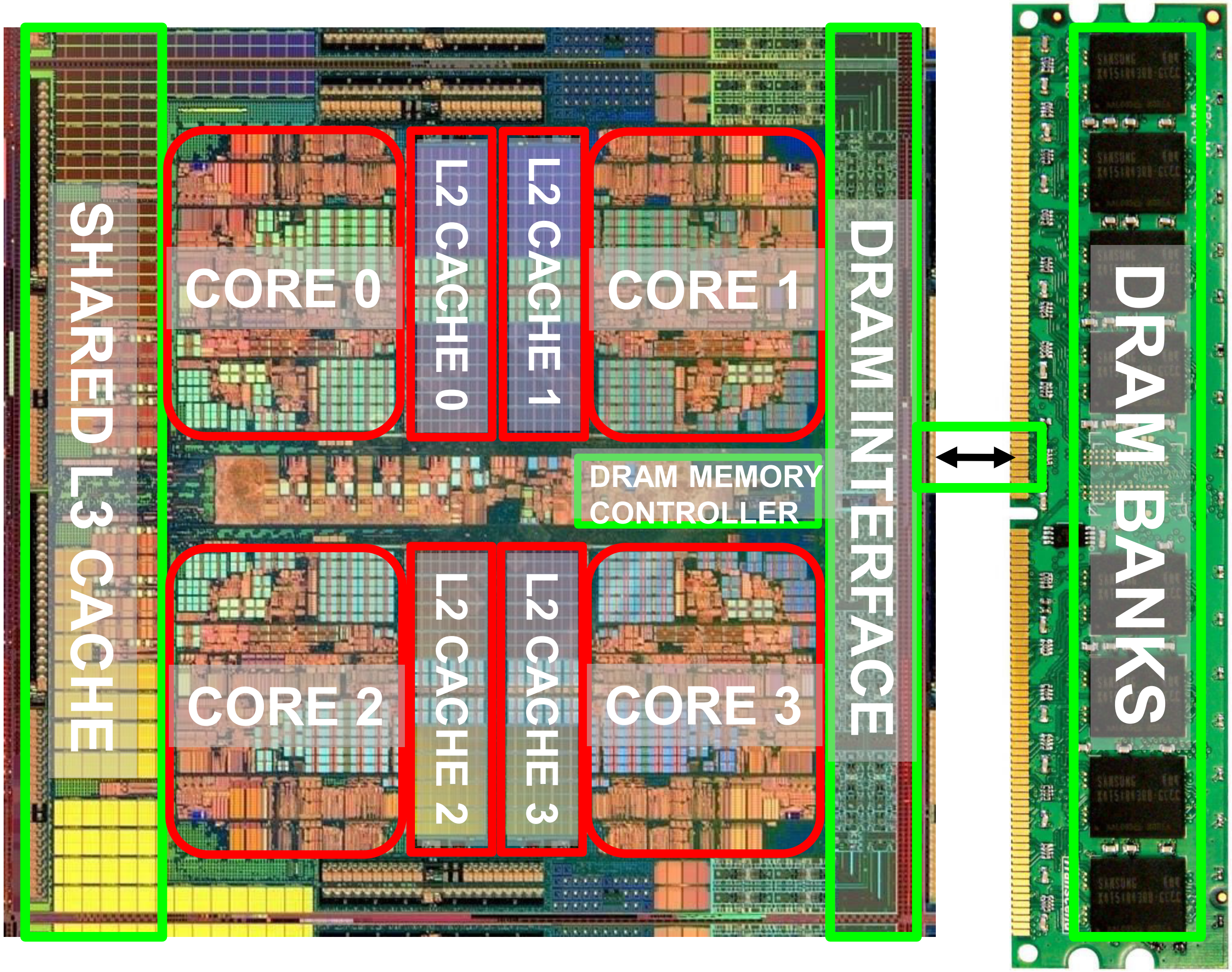
Demystifying Complex Workload–DRAM Interactions: An Experimental Study

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

†Carnegie Mellon University ‡Simon Fraser University §ETH Zürich

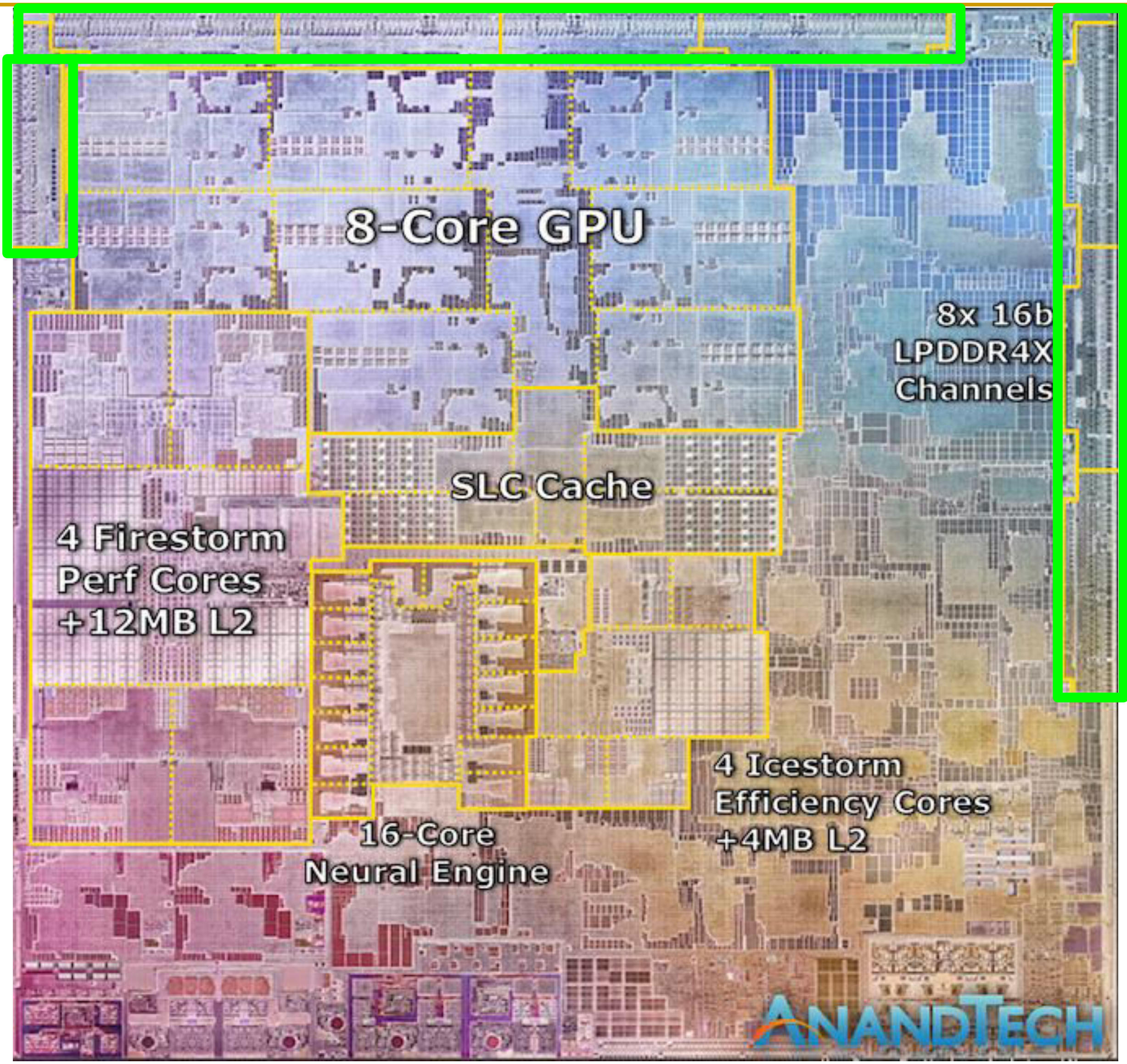
DRAM Control Logic Is Large

Multi-Core Chip



*Die photo credit: AMD Barcelona

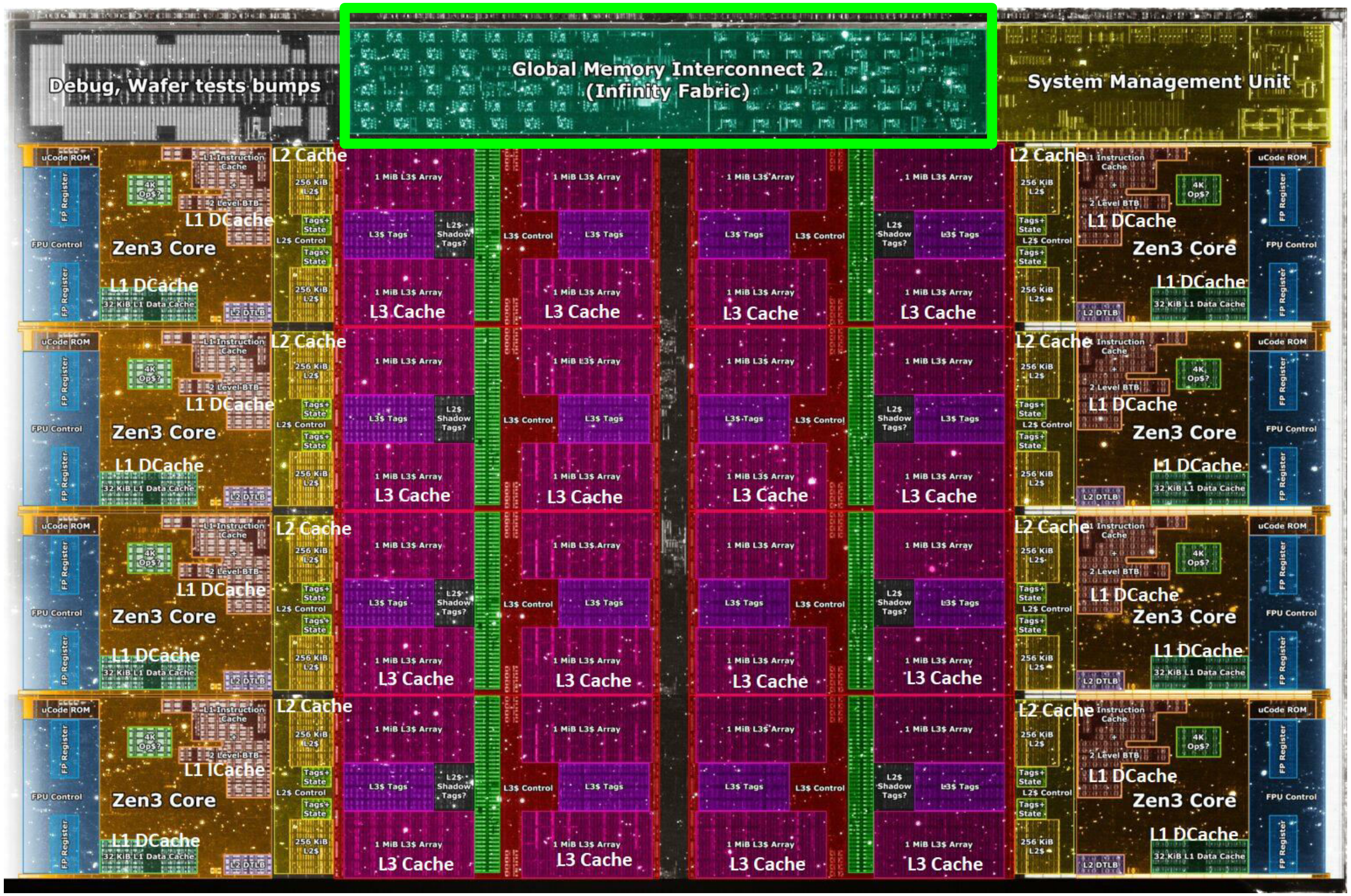
DRAM Control Logic Is Large



Apple M1, 2021

Source: https://www.anandtech.com/show/16252/mac-mini-apple-m1-tested

DRAM Control Logic Is Large



AMD Ryzen 5000, 2020

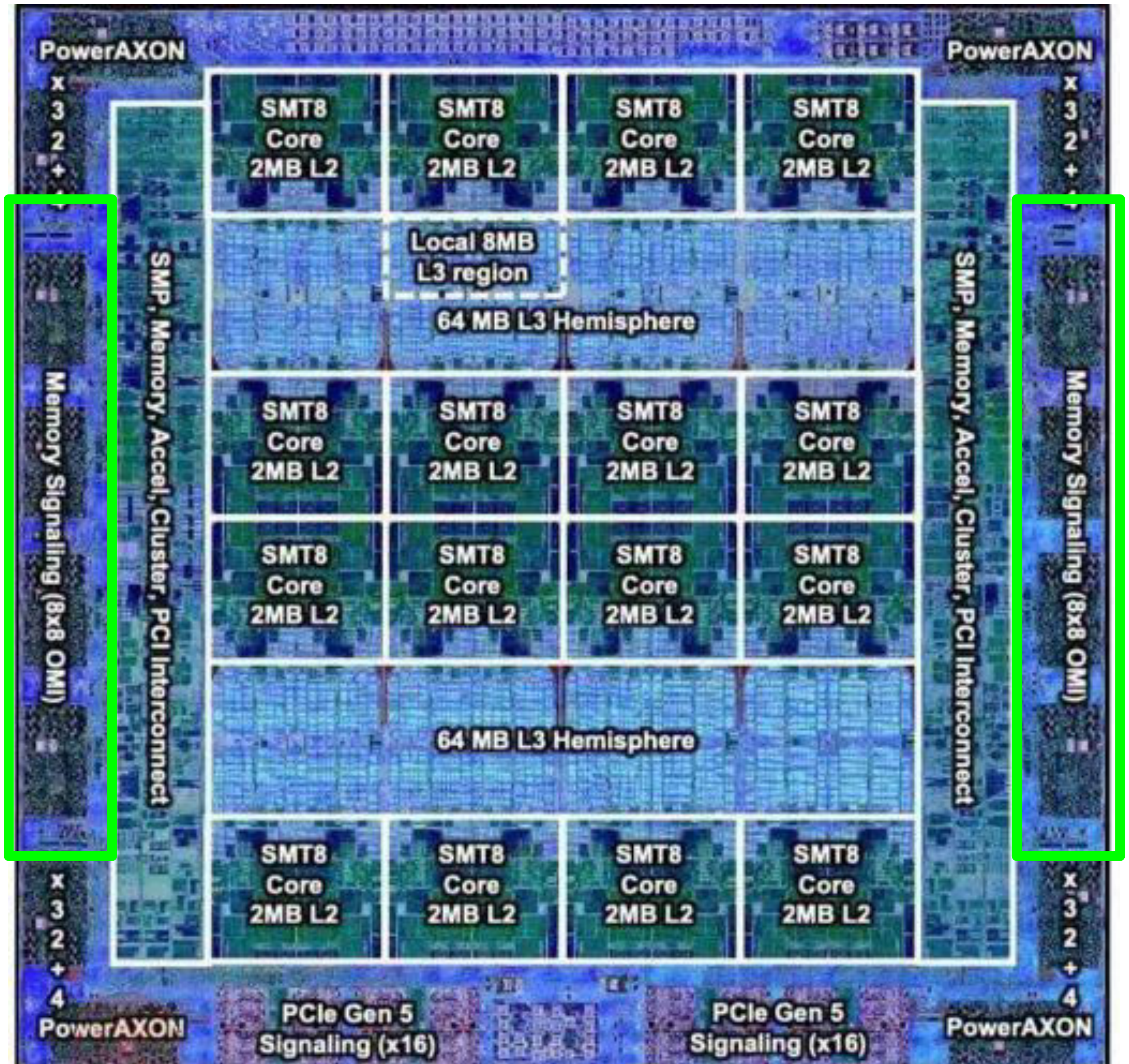
Core Count:
8 cores/16 threads

L1 Caches:
32 KB per core

L2 Caches:
512 KB per core

L3 Cache:
32 MB shared

DRAM Control Logic Is Large



IBM POWER10, 2020

Cores:
15-16 cores,
8 threads/core

L2 Caches:
2 MB per core

L3 Cache:
120 MB shared

https://www.it-techblog.de/ibm-power10-prozessor-mehr-speicher-mehr-tempo-mehr-sicherheit/09/2020/

DRAM Controller: Functions

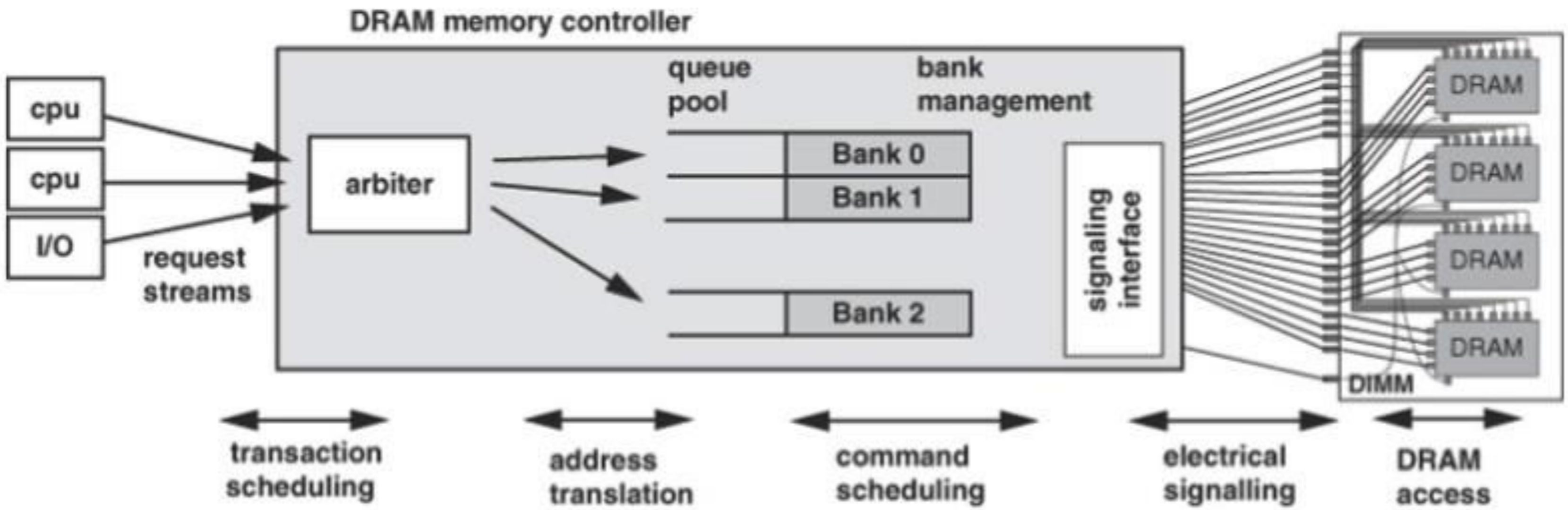
- Ensure correct operation of DRAM (refresh and timing)

- Service DRAM requests while obeying timing constraints of DRAM chips
  - Constraints: resource conflicts (bank, bus, channel), minimum write-to-read delays
  - Translate requests to DRAM command sequences

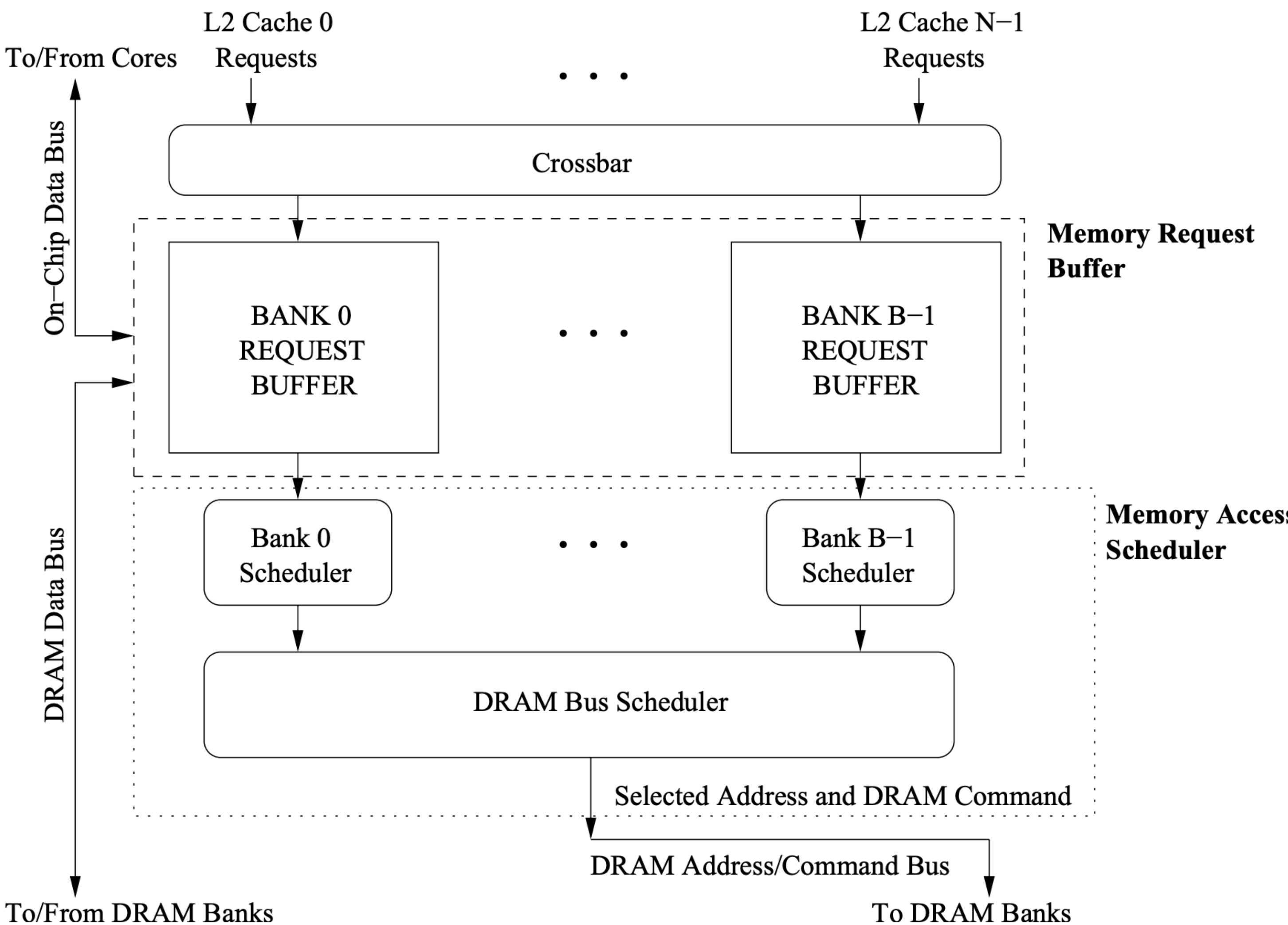
- Buffer and schedule requests for high performance + QoS
  - Reordering, row-buffer, bank, rank, bus management

- Manage power consumption and thermals in DRAM
  - Turn on/off DRAM chips, manage power modes

A Modern DRAM Controller (I)



A Modern DRAM Controller



Mutlu+, "Stall-Time Fair Memory Scheduling," MICRO 2007.

DRAM Scheduling Policies (I)

- FCFS (first come first served)
  - Oldest request first

- FR-FCFS (first ready, first come first served)

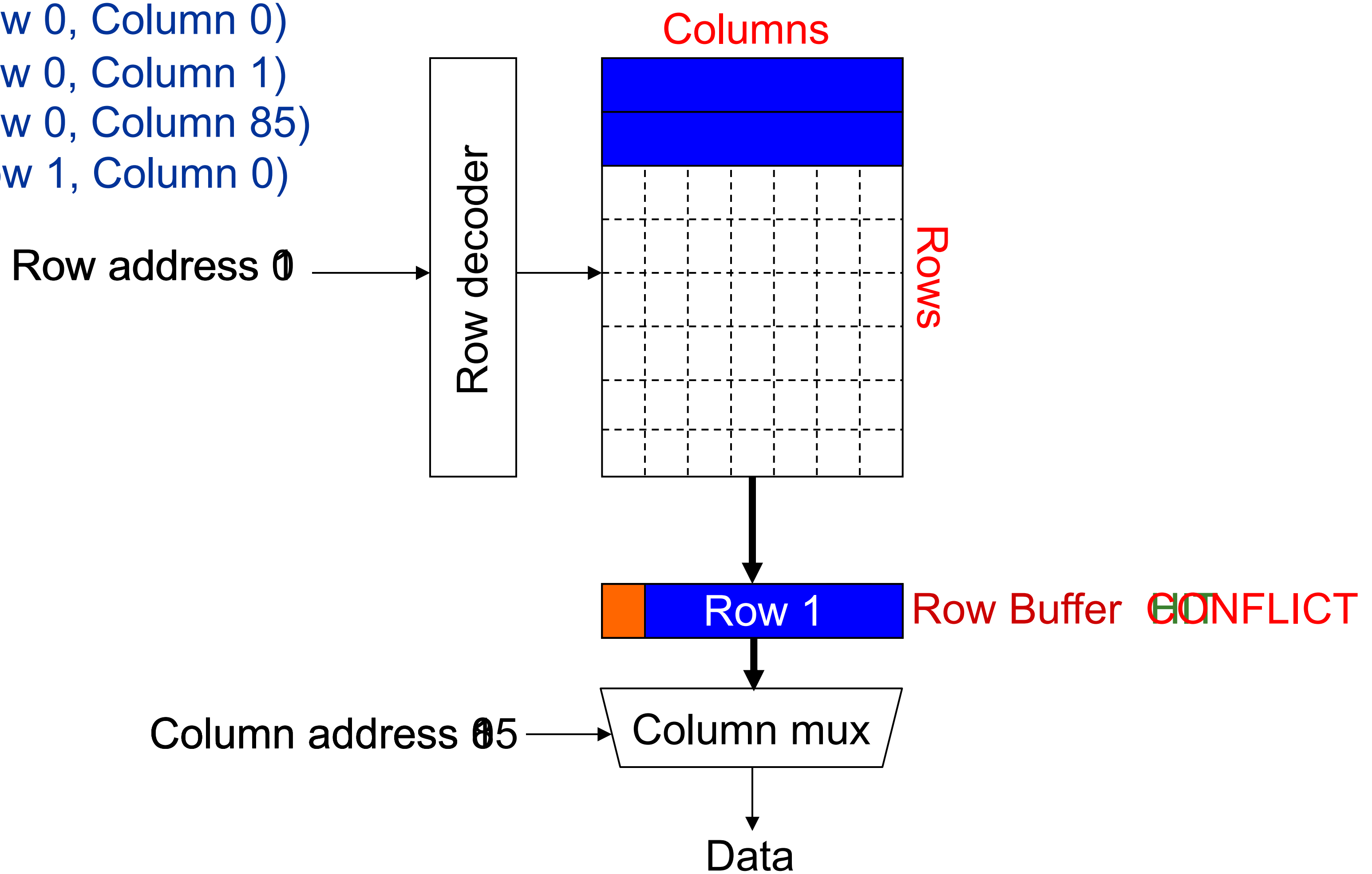
  1. Row-hit first
  2. Oldest first

  Goal: Maximize row buffer hit rate → maximize DRAM throughput

  - Actually, scheduling is done at the command level
    - Column commands (read/write) prioritized over row commands (activate/precharge)
    - Within each group, older commands prioritized over younger ones

Review: DRAM Bank Operation

Access Address:
(Row 0, Column 0)
(Row 0, Column 1)
(Row 0, Column 85)
(Row 1, Column 0)



DRAM Scheduling Policies (II)

- A scheduling policy is a request prioritization order

- Prioritization can be based on
  - Request age
  - Row buffer hit/miss status
  - Request type (prefetch, read, write)
  - Requestor type (load miss or store miss)
  - Request criticality
    - Oldest miss in the core?
    - How many instructions in core are dependent on it?
    - Will it stall the processor?
  - Interference caused to other cores
  - ...

Row Buffer Management Policies

- Open row
  - Keep the row open after an access

  \+ Next access might need the same row → row hit

  -- Next access might need a different row → row conflict, wasted energy

- Closed row
  - Close the row after an access (if no other requests already in the request buffer need the same row)

  \+ Next access might need a different row → avoid a row conflict

  -- Next access might need the same row → extra activate latency

- Adaptive policies
  - Predict whether or not the next access to the bank will be to the same row and act accordingly

Open vs. Closed Row Policies

Policy	First access	Next access	Commands needed for next access
Open row	Row 0	Row 0 (row hit)	Read
Open row	Row 0	Row 1 (row conflict)	Precharge + Activate Row 1 + Read
Closed row	Row 0	Row 0 – access in request buffer (row hit)	Read
Closed row	Row 0	Row 0 – access not in request buffer (row closed)	Activate Row 0 + Read + Precharge
Closed row	Row 0	Row 1 (row closed)	Activate Row 1 + Read + Precharge

DRAM Power Management

- DRAM chips have power modes
- Idea: When not accessing a chip power it down

- Power states
  - Active (highest power)
  - All banks idle
  - Power-down
  - Self-refresh (lowest power)

- Tradeoff: State transitions incur latency during which the chip cannot be accessed

Difficulty of DRAM Control

Why Are DRAM 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 DRAM 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.

More on DRAM Operation

- 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.

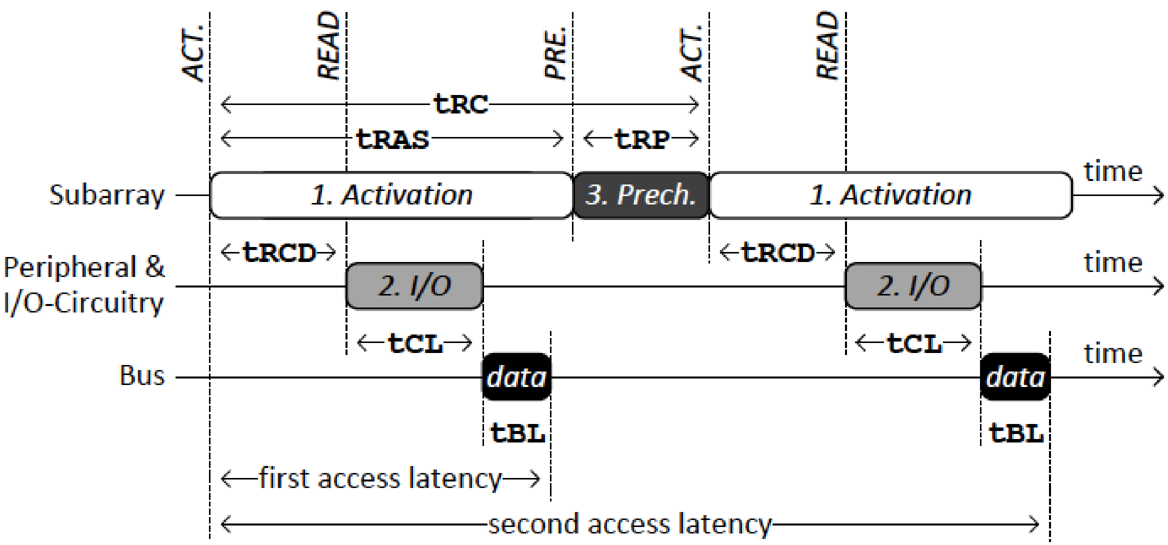
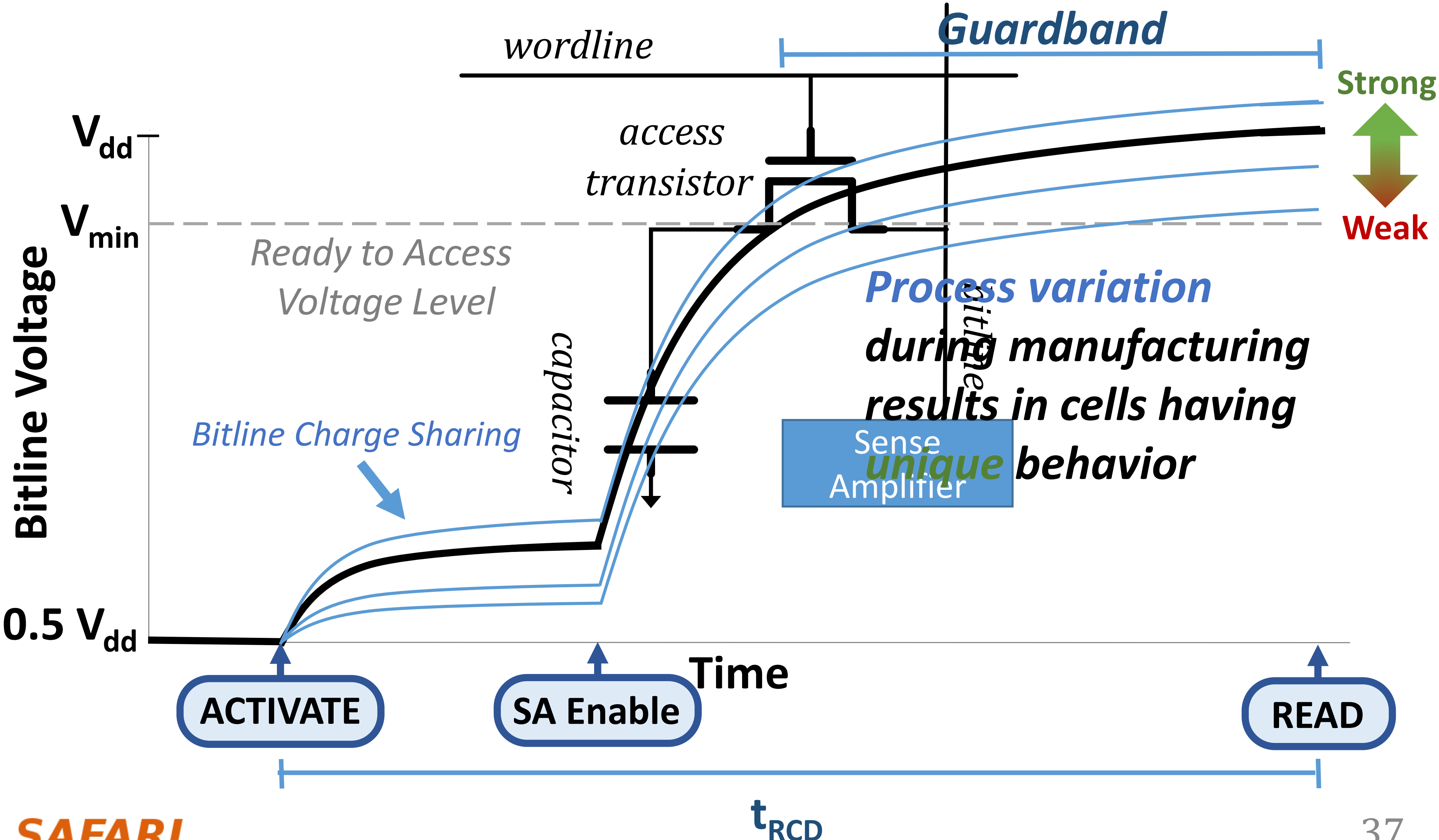


Figure 5. Three Phases of DRAM Access

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

Why Timing Constraints?



***Process variation* during manufacturing results in cells having *unique* behavior**

Why So Many Timing Constraints? (I)

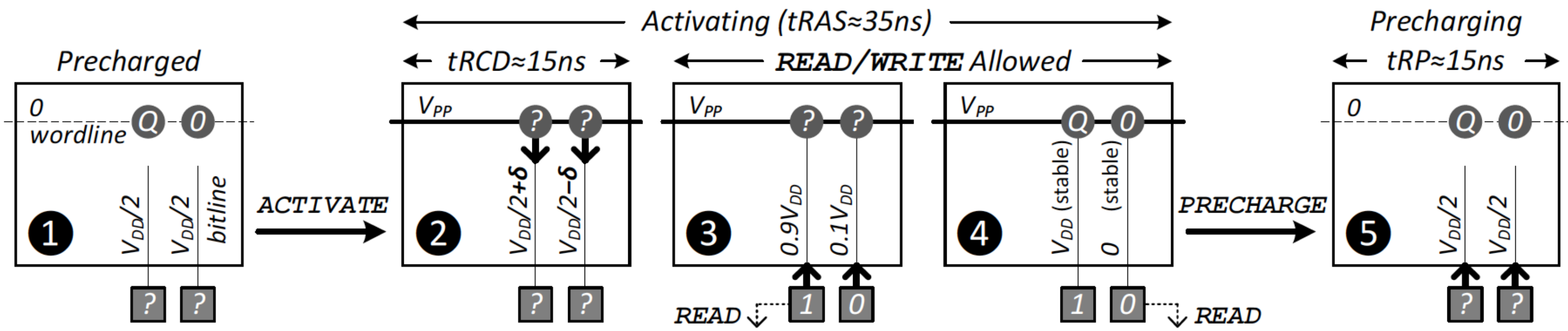


Figure 4. DRAM bank operation: Steps involved in serving a memory request [17] $(V_{PP} > V_{DD})$

Category	RowCmd↔RowCmd			RowCmd↔ColCmd			ColCmd↔ColCmd			ColCmd→DATA	
Name	*tRC*	*tRAS*	*tRP*	*tRCD*	*tRTP*	*tWR**	*tCCD*	*tRTW*†	*tWTR**	*CL*	*CWL*
Commands	A→A	A→P	P→A	A→R/W	R→P	W*→P	R(W)→R(W)	R→W	W*→R	R→DATA	W→DATA
Scope	Bank	Bank	Bank	Bank	Bank	Bank	Channel	Rank	Rank	Bank	Bank
Value (ns)	**~50**	~35	13-15	13-15	~7.5	**15**	5-7.5	11-15	~7.5	13-15	10-15

A: ACTIVATE– P: PRECHARGE– R: READ– W: WRITE

∗ Goes into effect after the last write *data*, not from the WRITE command

† Not explicitly specified by the JEDEC DDR3 standard [18]. Defined as a function of other timing constraints.

Table 1. Summary of DDR3-SDRAM timing constraints (derived from Micron's 2Gb DDR3-SDRAM datasheet [33])

Kim et al., "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM," ISCA 2012.

Why So Many Timing Constraints? (II)

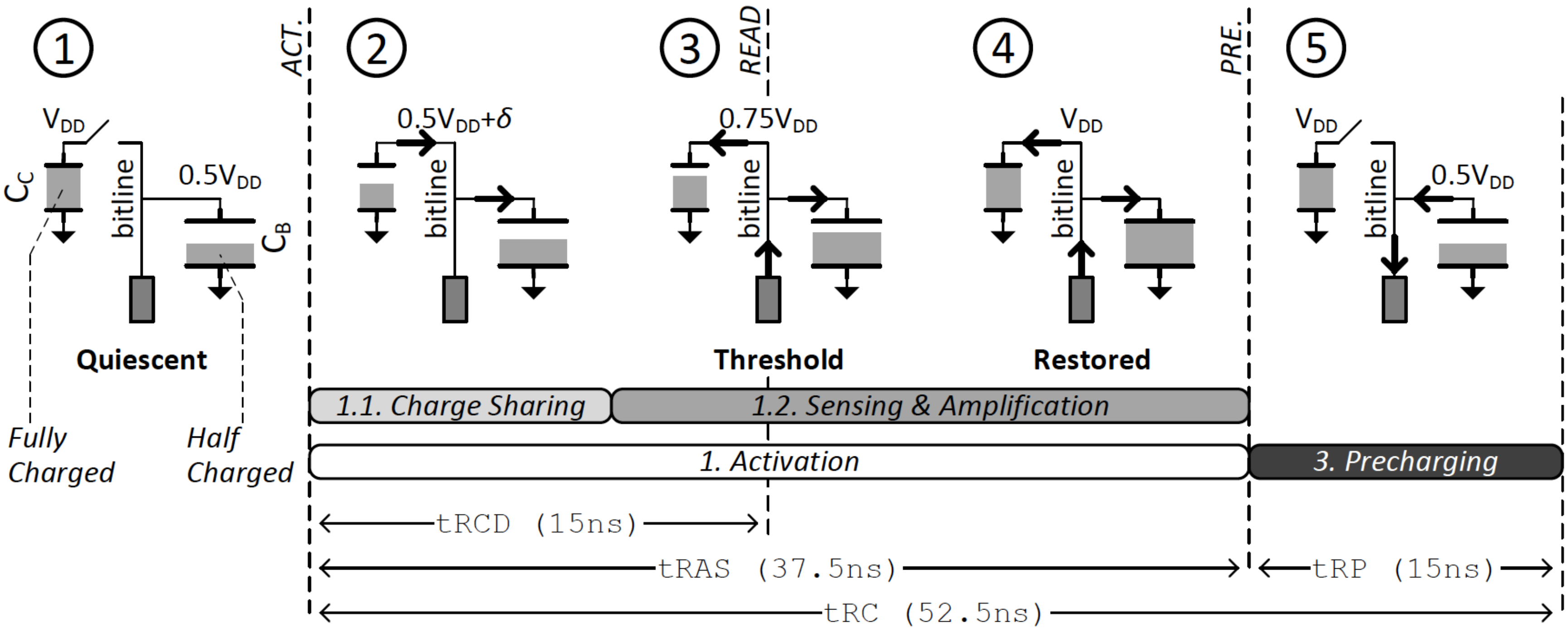


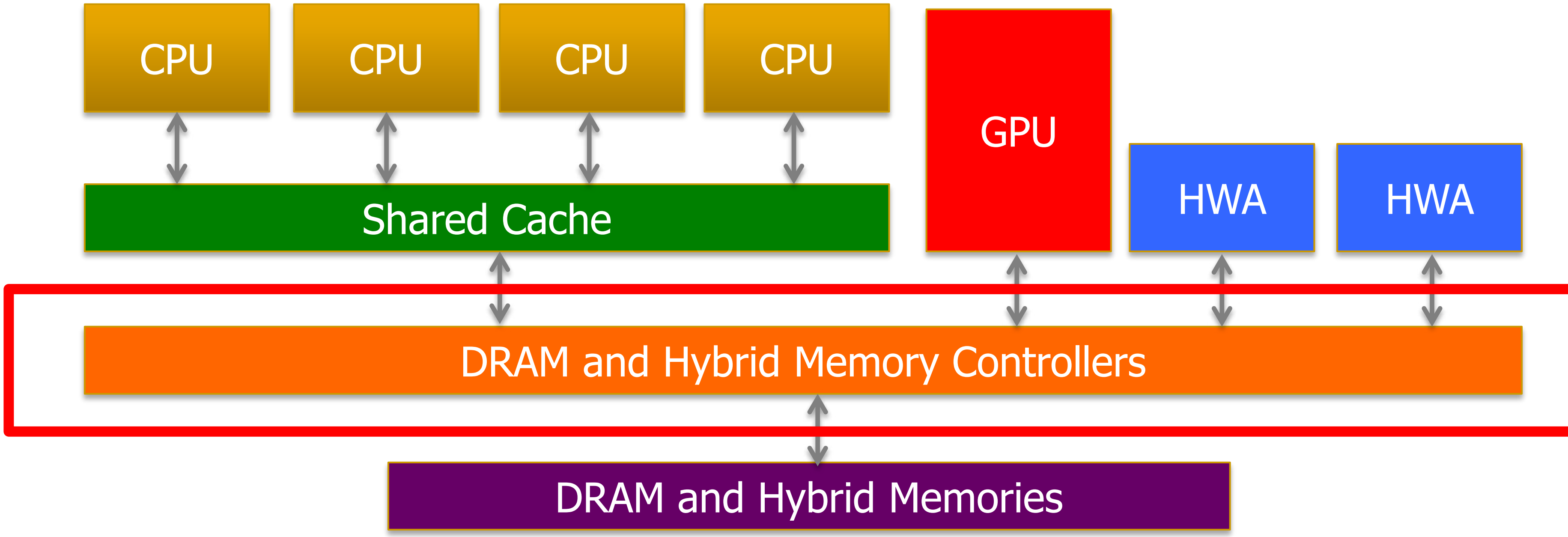
Figure 6. Charge Flow Between the Cell Capacitor (C_C), Bitline Parasitic Capacitor (C_B), and the Sense-Amplifier ($C_B \approx 3.5C_C$ [39])

Lee et al., "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture," HPCA 2013.

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

DRAM Controller Design Is Becoming More Difficult



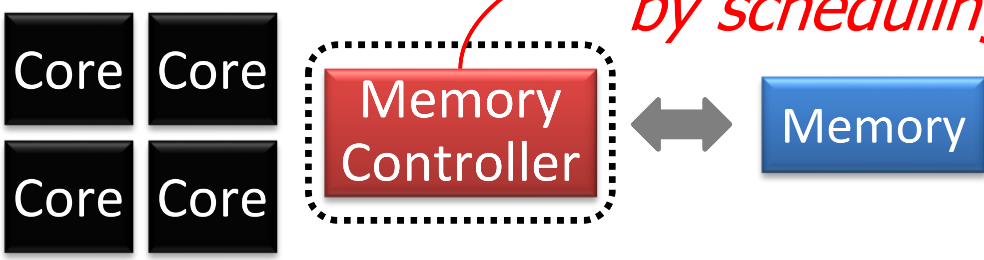
- Heterogeneous agents: CPUs, GPUs, and HWAs
- Main memory interference between CPUs, GPUs, HWAs
- Many timing constraints for various memory types
- Many goals at the same time: performance, fairness, QoS, energy efficiency, ...

Reality and Dream

- Reality: It is 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?

Memory Controller: Performance Function



Resolves memory contention by scheduling requests

How to schedule requests to maximize system performance?

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

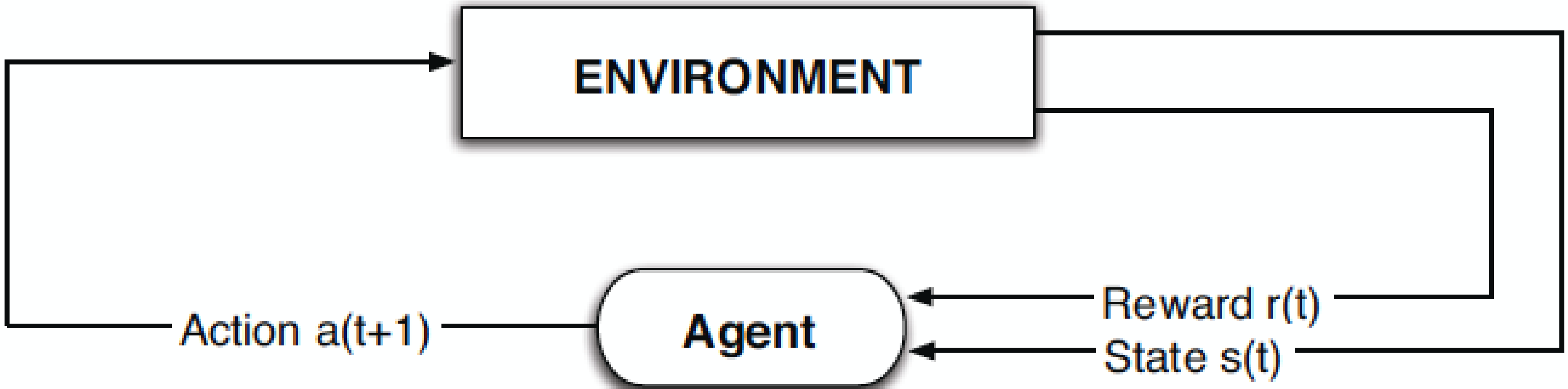
- Idea: A memory controller that adapts its scheduling policy to workload behavior and system conditions using machine learning.

- 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

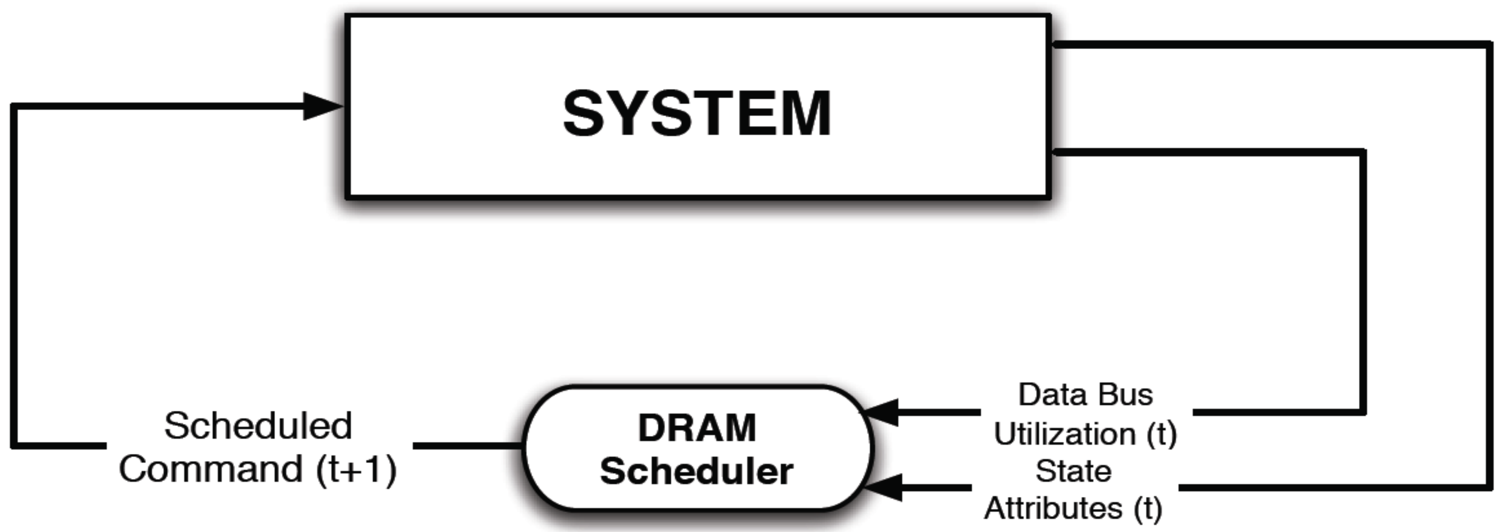


Goal: Learn to choose actions to maximize $r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots$ ($0 \leq \gamma < 1$)

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.

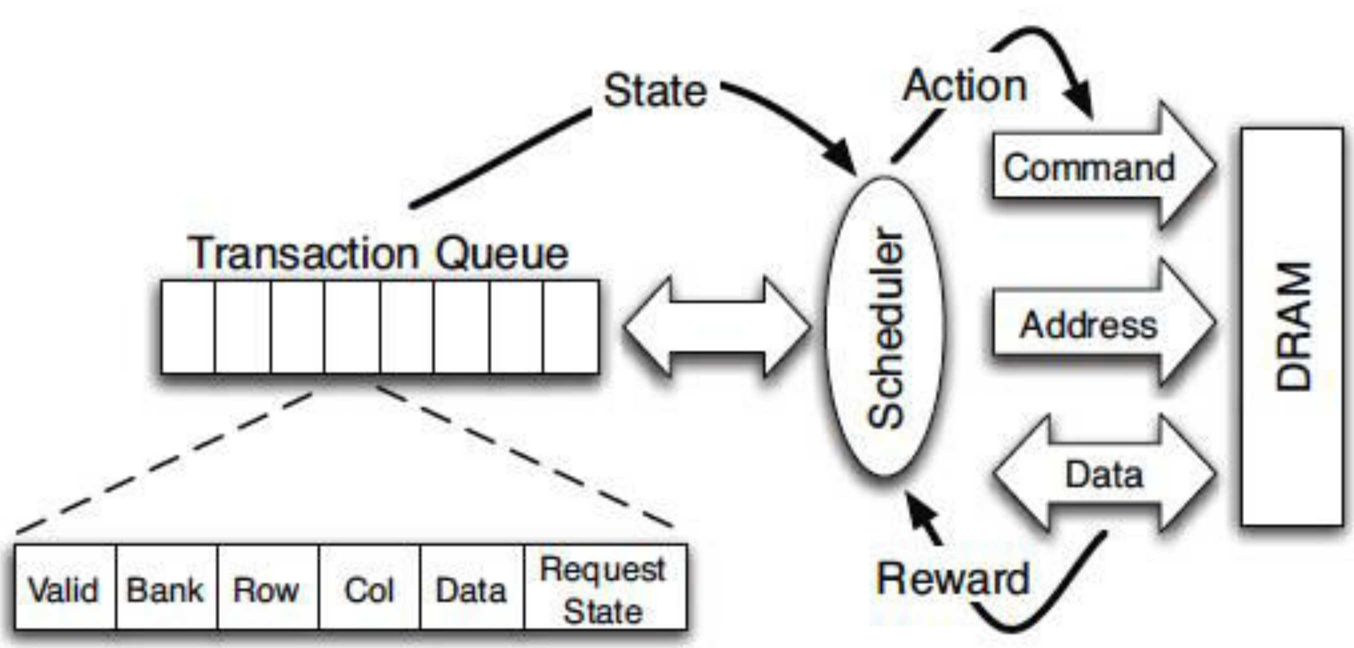


Figure 4: High-level overview of an RL-based scheduler.

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

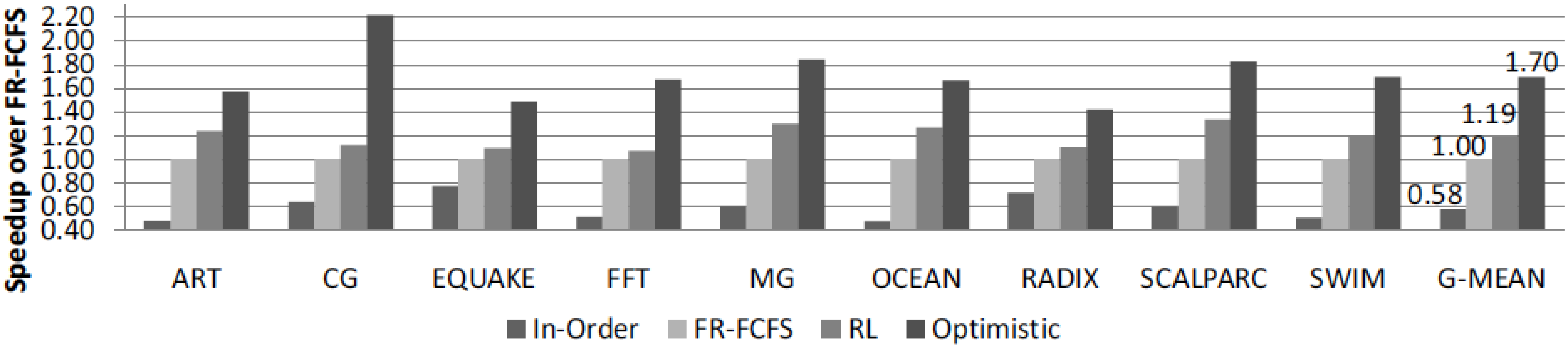


Figure 7: Performance comparison of in-order, FR-FCFS, RL-based, and optimistic memory controllers

Large, robust performance improvements over many human-designed policies

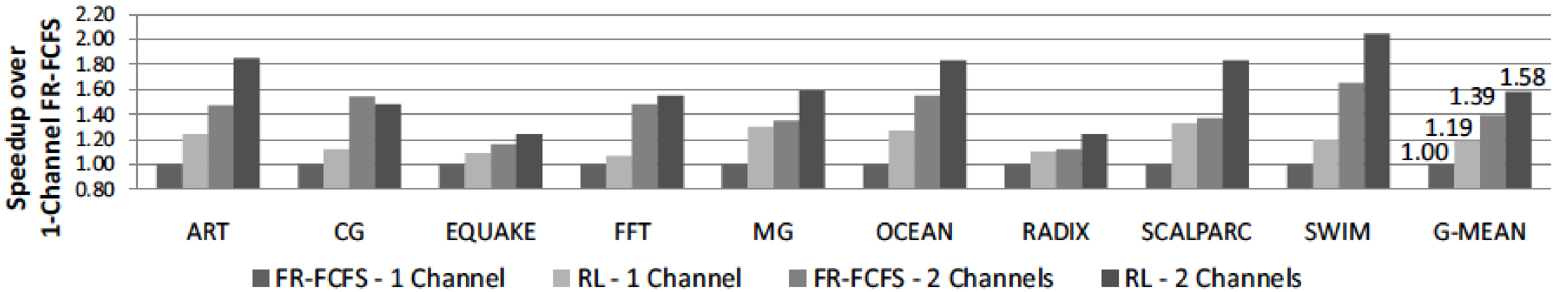


Figure 15: Performance comparison of FR-FCFS and RL-based memory controllers on systems with 6.4GB/s and 12.8GB/s peak DRAM bandwidth

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

Challenge and Opportunity for Future

Self-Optimizing

(Data-Driven)

Computing 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

We need to rethink design (of all controllers)

Self-Optimizing Memory Prefetchers

- Rahul Bera, Konstantinos Kanellopoulos, Anant Nori, Taha Shahroodi, Sreenivas Subramoney, and Onur Mutlu,
"Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning"
Proceedings of the 54th International Symposium on Microarchitecture (***MICRO***), Virtual, October 2021.
[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Pythia Source Code (Officially Artifact Evaluated with All Badges)]
[arXiv version]

Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning

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

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

Self-Optimizing Hybrid SSD Controllers

Gagandeep Singh, Rakesh Nadig, Jisung Park, Rahul Bera, Nastaran Hajinazar, David Novo, Juan Gomez-Luna, Sander Stuijk, Henk Corporaal, and Onur Mutlu,
"Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning"
Proceedings of the 49th International Symposium on Computer Architecture ***(ISCA)***, New York, June 2022.
[Slides (pptx) (pdf)]
[arXiv version]
[Sibyl Source Code]
[Talk Video (16 minutes)]

Sibyl: Adaptive and Extensible Data Placement in Hybrid Storage Systems Using Online Reinforcement Learning

Gagandeep Singh[1] Rakesh Nadig[1] Jisung Park[1] Rahul Bera[1] Nastaran Hajinazar[1]
David Novo[3] Juan Gómez-Luna[1] Sander Stuijk[2] Henk Corporaal[2] Onur Mutlu[1]
[1]ETH Zürich [2]Eindhoven University of Technology [3]LIRMM, Univ. Montpellier, CNRS

https://arxiv.org/pdf/2205.07394.pdf

Learning-Based Off-Chip Load Predictors

- **Best Paper Award at MICRO 2022**

Hermes: Accelerating Long-Latency Load Requests via Perceptron-Based Off-Chip Load Prediction

Rahul Bera[1] Konstantinos Kanellopoulos[1] Shankar Balachandran[2] David Novo[3]
Ataberk Olgun[1] Mohammad Sadrosadati[1] Onur Mutlu[1]

[1]ETH Zürich [2]Intel Processor Architecture Research Lab [3]LIRMM, Univ. Montpellier, CNRS

https://arxiv.org/pdf/2209.00188.pdf

Architectures for Intelligent Machines

Data-centric

Data-driven

Data-aware

Key Problems with Today's Architectures

- 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**

Fundamentally Better Architectures

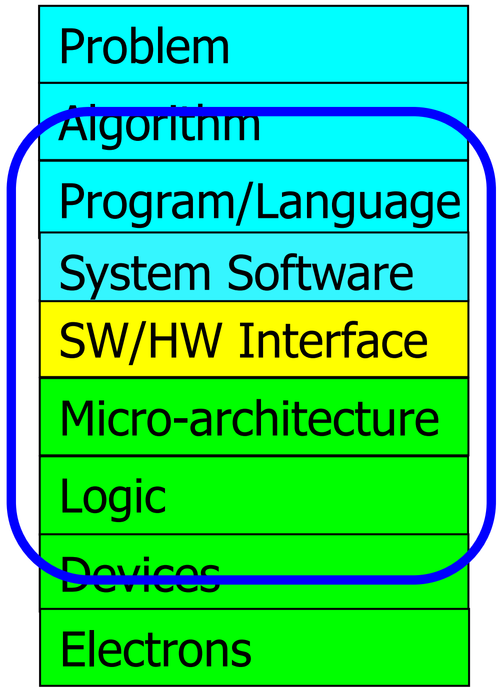
Data-centric

Data-driven

Data-aware

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

We Need to Think Across the Entire Stack



We can get there step by step

A Blueprint for Fundamentally Better Architectures

- Onur Mutlu,
  "Intelligent Architectures for Intelligent Computing Systems"
  *Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**)*, Virtual, February 2021.
  [Slides (pptx) (pdf)]
  [IEDM Tutorial Slides (pptx) (pdf)]
  [Short DATE Talk Video (11 minutes)]
  [Longer IEDM Tutorial Video (1 hr 51 minutes)]

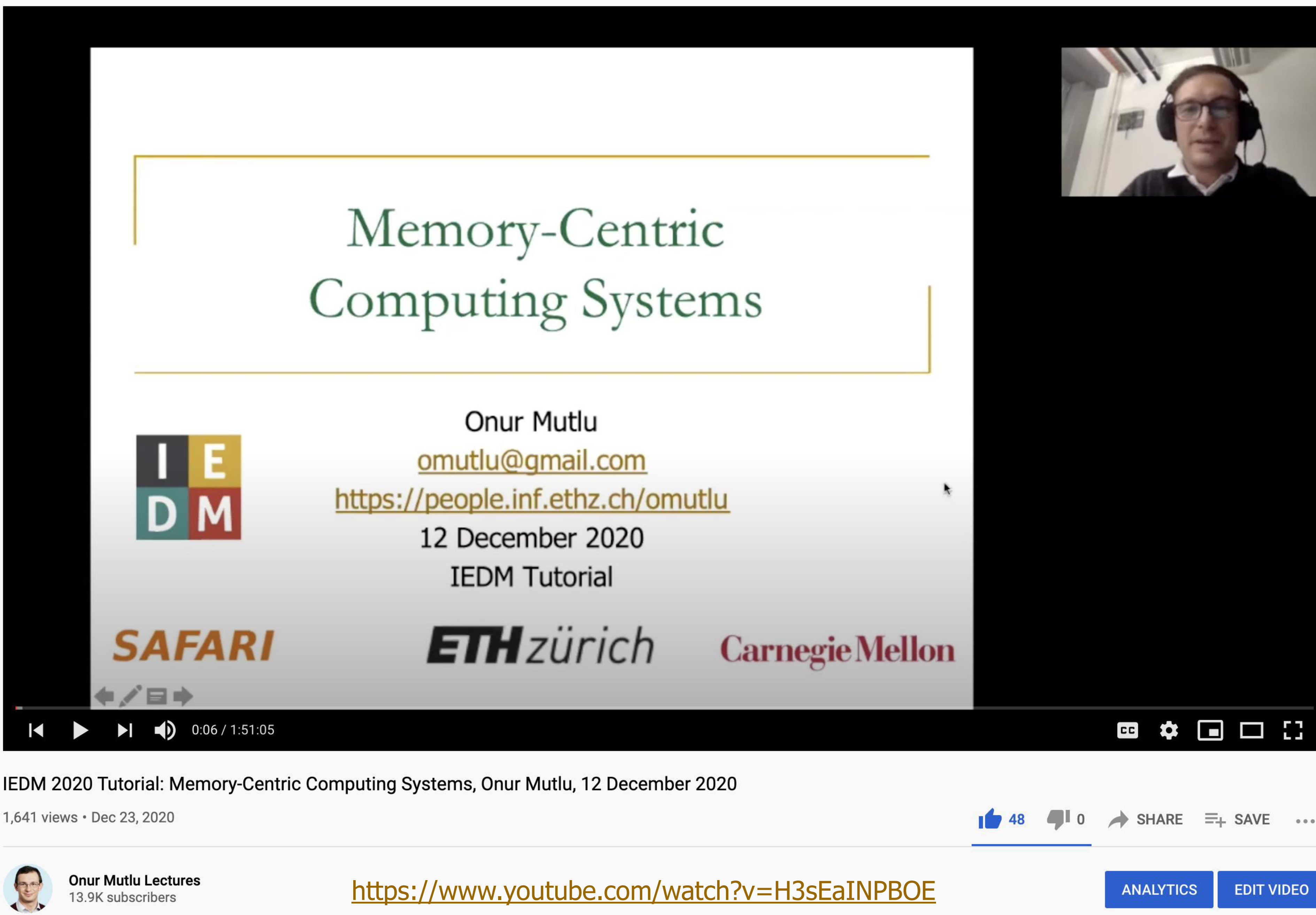
Intelligent Architectures for Intelligent Computing Systems

Onur Mutlu
ETH Zurich
omutlu@gmail.com

A Tutorial on Fundamentally Better Architectures

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

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



https://www.youtube.com/onurmutlulectures

Computer Architecture

Lecture 11a: Memory Controllers

Prof. Onur Mutlu
ETH Zürich

Fall 2022

3 November 2022

Backup Slides

Data-Aware Architectures

Corollaries: Architectures Today …

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

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

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

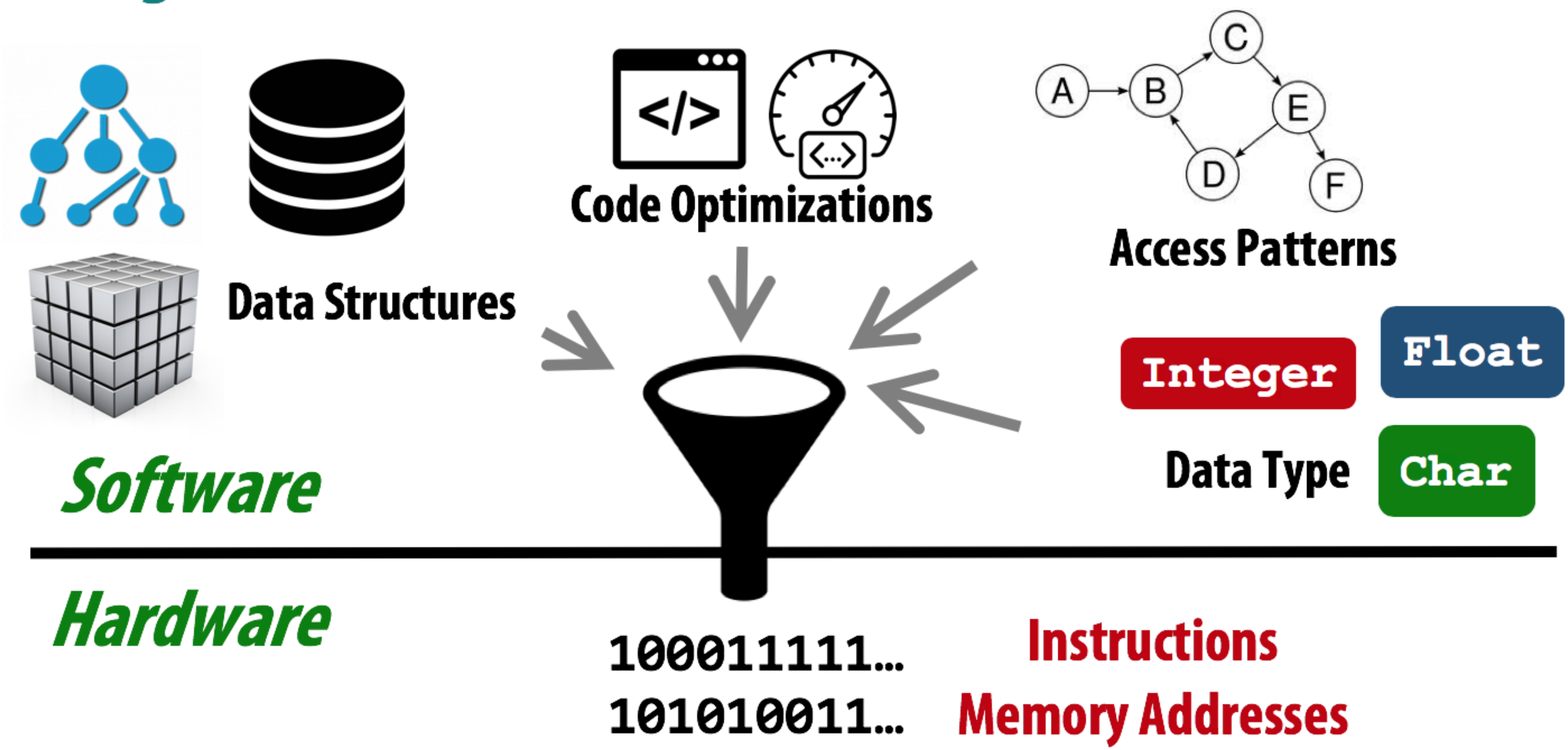
Data-Aware Architectures

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

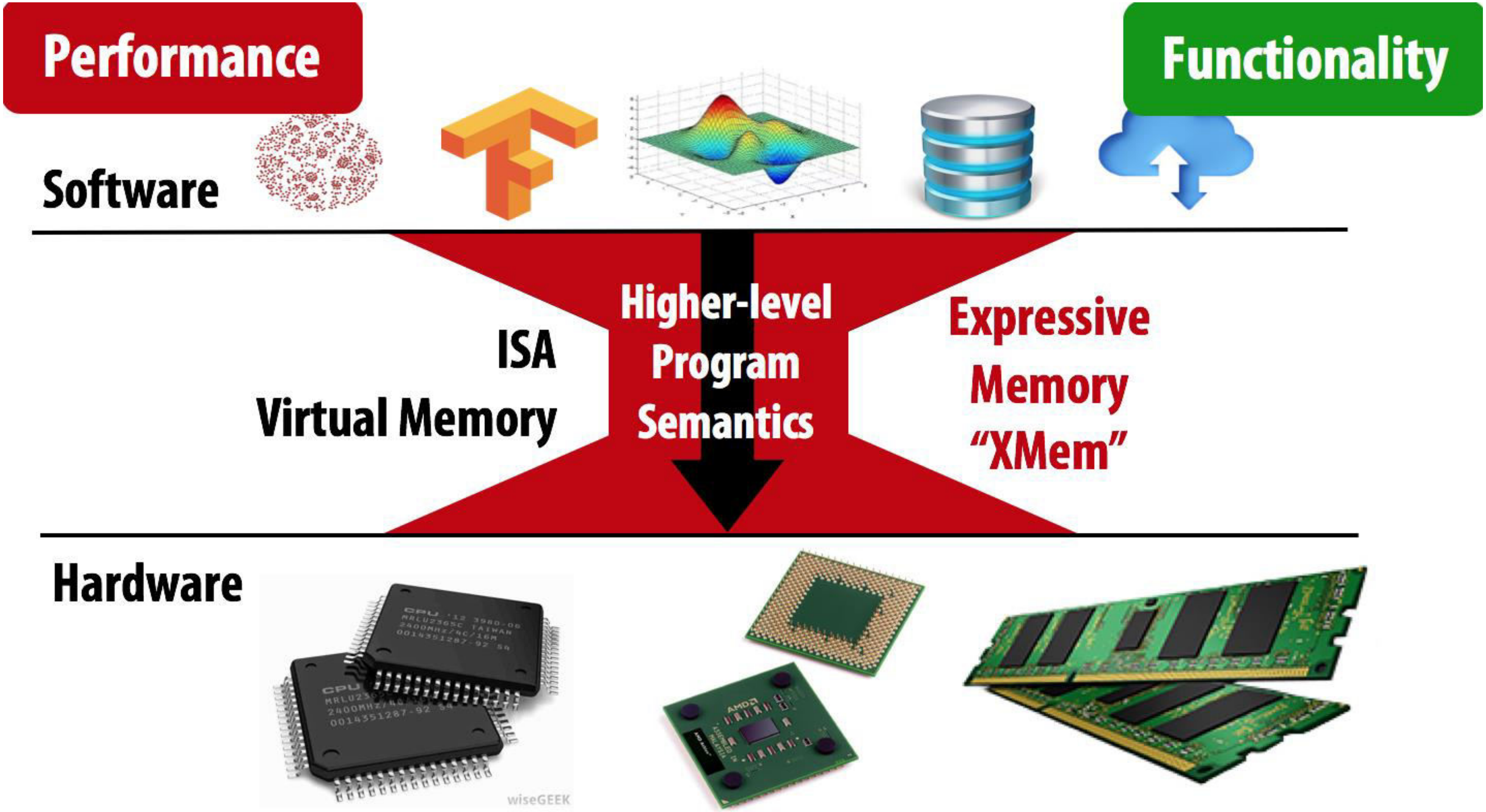
- It makes use of different properties of data to improve performance, efficiency and other metrics
  - Compressibility
  - Approximability
  - Locality
  - Sparsity
  - Criticality for Computation X
  - Access Semantics
  - …

One Problem: Limited Expressiveness

Higher-level information is not visible to HW



A Solution: More Expressive Interfaces



Expressive (Memory) Interfaces

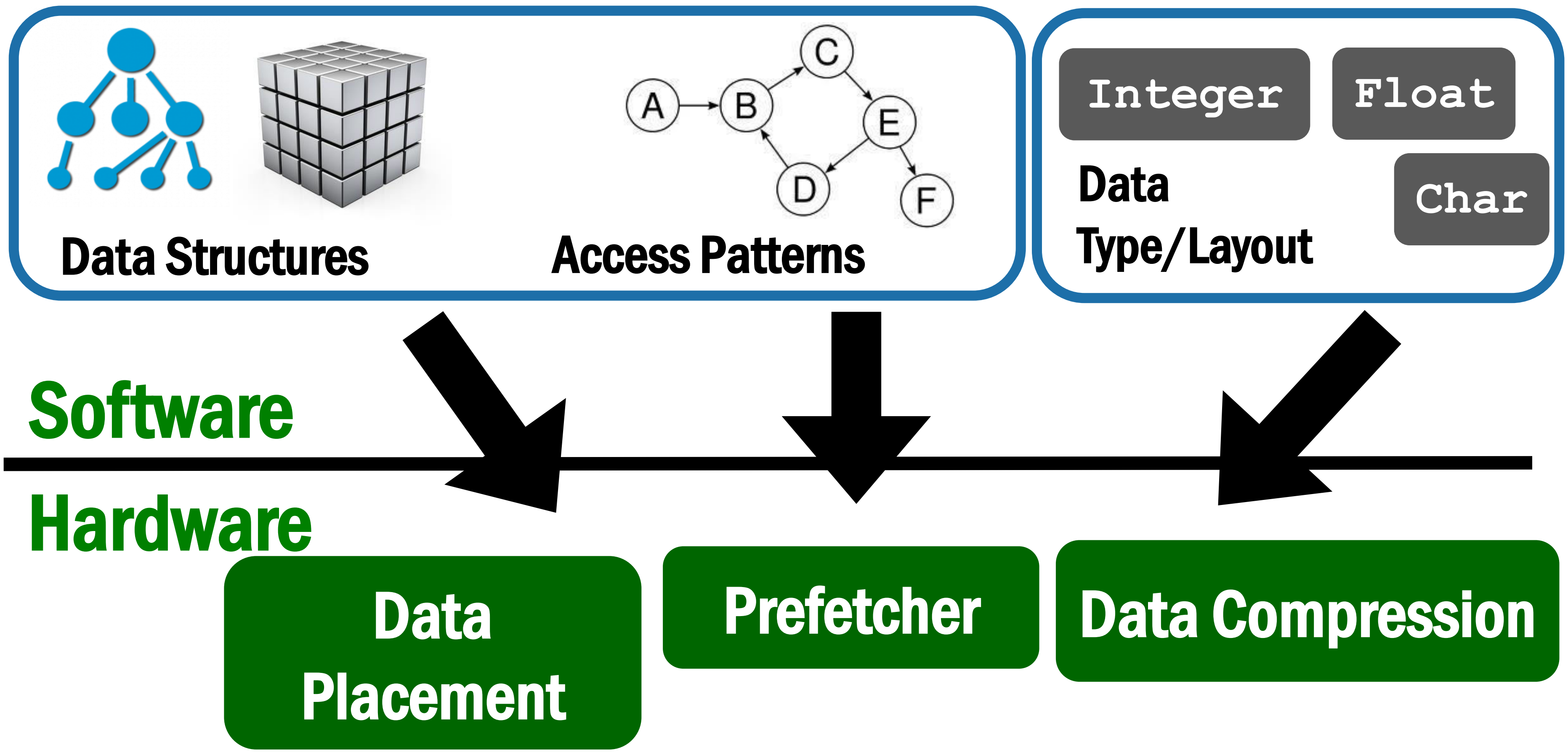
- Nandita Vijaykumar, Abhilasha Jain, Diptesh Majumdar, Kevin Hsieh, Gennady Pekhimenko, Eiman Ebrahimi, Nastaran Hajinazar, Phillip B. Gibbons and Onur Mutlu,
"A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory"
Proceedings of the 45th International Symposium 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

SW provides key program information to HW



Broader goal: Enable many cross-layer optimizations

Express:

Data structures

Access semantics

Data types

Working set

Reuse

Access frequency

...

Optimizations:

Cache Management

Data Placement in DRAM

Data Compression

Approximation

DRAM Cache Management

NVM Management

NUCA/NUMA Optimizations

...

Benefits:

More efficient HW:

✓ Performance

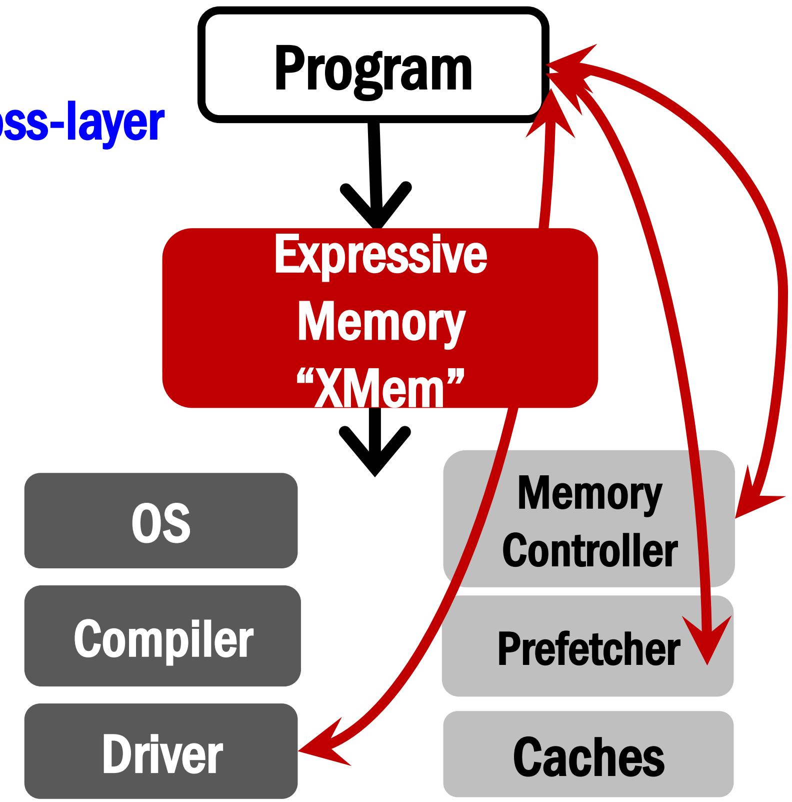
Reduced SW burden:

✓ Programmability

✓ Portability

Our approach: Rich cross-layer abstractions

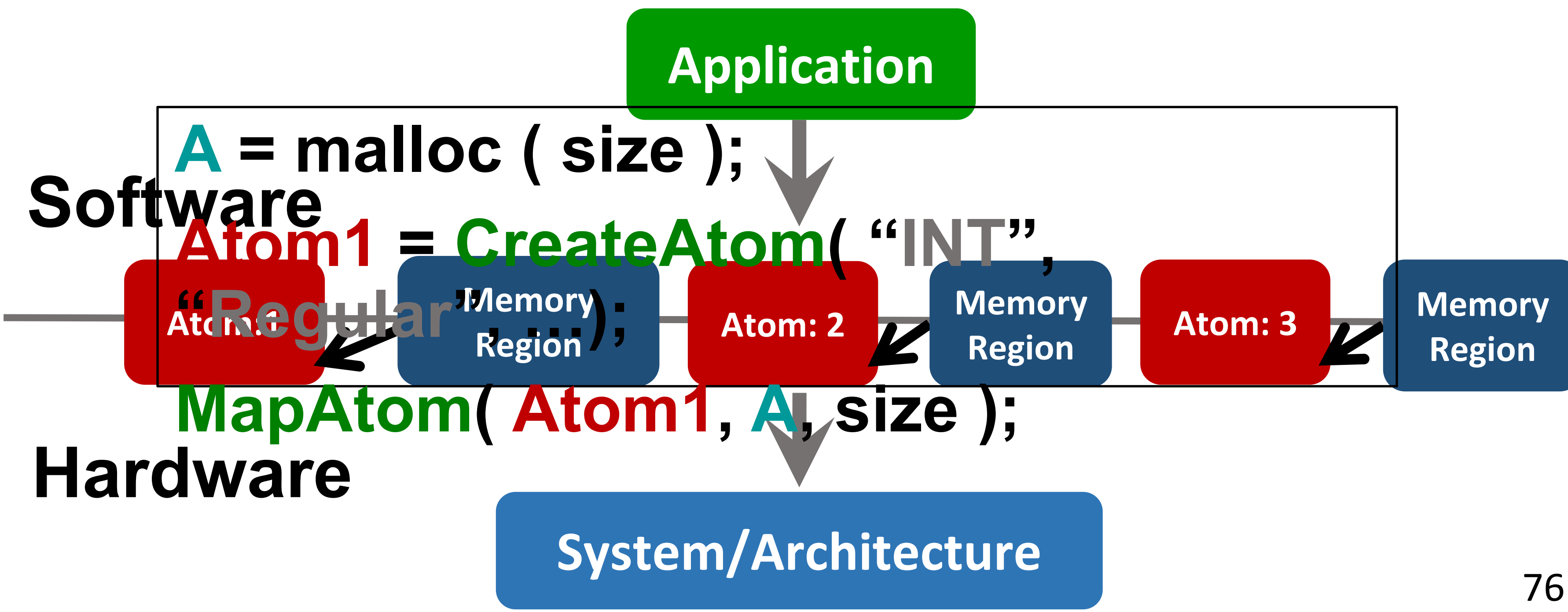
1. Generality: Enable a wide range of cross-layer approaches
2. Minimize programmer effort
3. Overhead



<u>Approach</u>: Flexibly associate specific semantic information with any data & code

Example: XMem

- Goal: convey data semantics to the hardware enables more intelligent management of resources.
- XMem: introduces a new HW/SW abstraction, called *Atom,* for conveying data semantics




```
A = malloc ( size );
Atom1 = CreateAtom( "INT", "Regular", ... );
MapAtom( Atom1, A, size );
```


Vijaykumar+, "**A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory**",

XMem Aids/Enables Many Optimizations

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

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

Expressive (Memory) Interfaces

- 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]

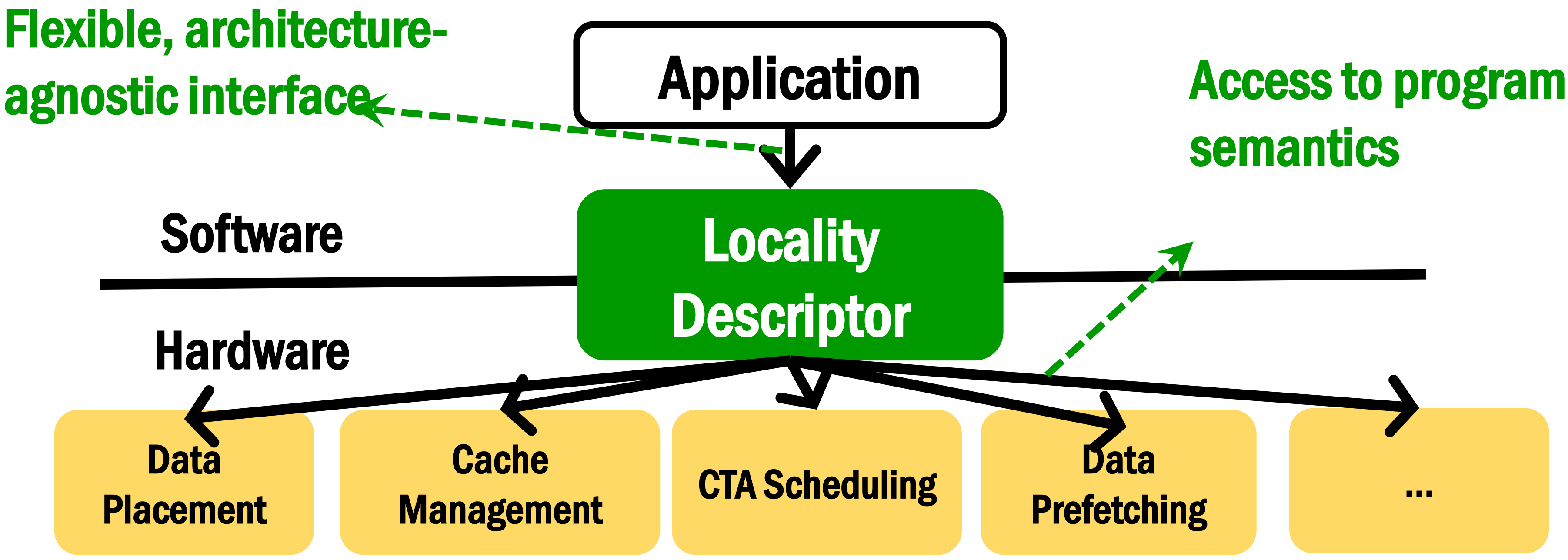
The Locality Descriptor:
A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs

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

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

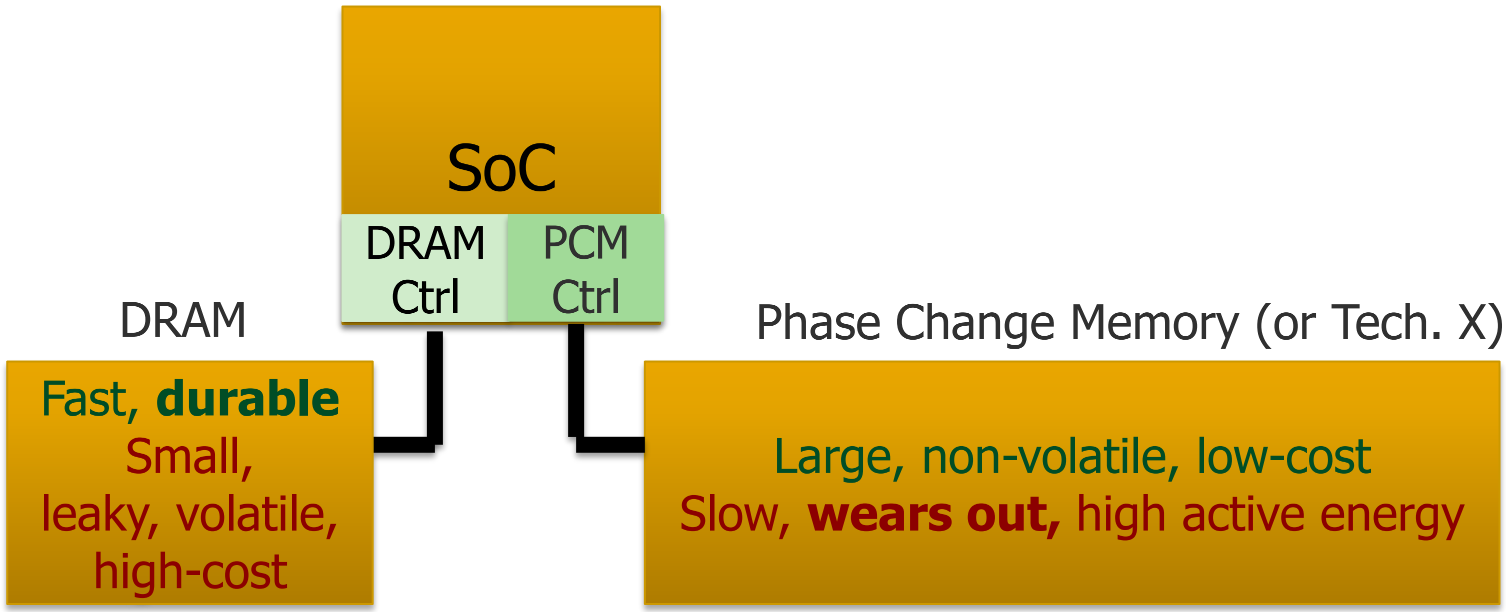
Locality Descriptor: Executive Summary

Exploiting data locality in GPUs is a challenging task



Performance Benefits:
26.6% (up to 46.6%) from cache locality
53.7% (up to 2.8x) from NUMA locality

An Example: Hybrid Memory Management



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

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

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

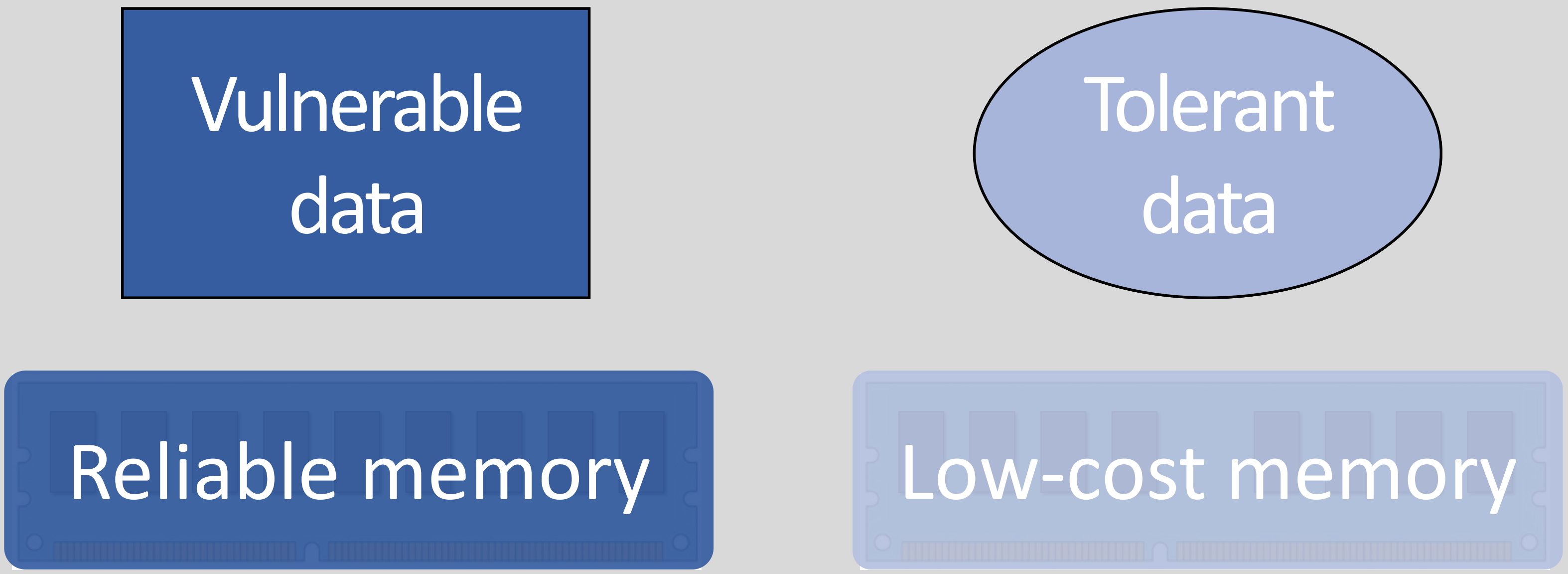
An Example: Heterogeneous-Reliability Memory

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

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

Yixin Luo Sriram Govindan* Bikash Sharma* Mark Santaniello* Justin Meza
Aman Kansal* Jie Liu* Badriddine Khessib* Kushagra Vaid* Onur Mutlu
Carnegie Mellon University, yixinluo@cs.cmu.edu, {meza, onur}@cmu.edu
*Microsoft Corporation, {srgovin, bsharma, marksan, kansal, jie.liu, bkhessib, kvaid}@microsoft.com

Exploiting Memory Error Tolerance with Hybrid Memory Systems



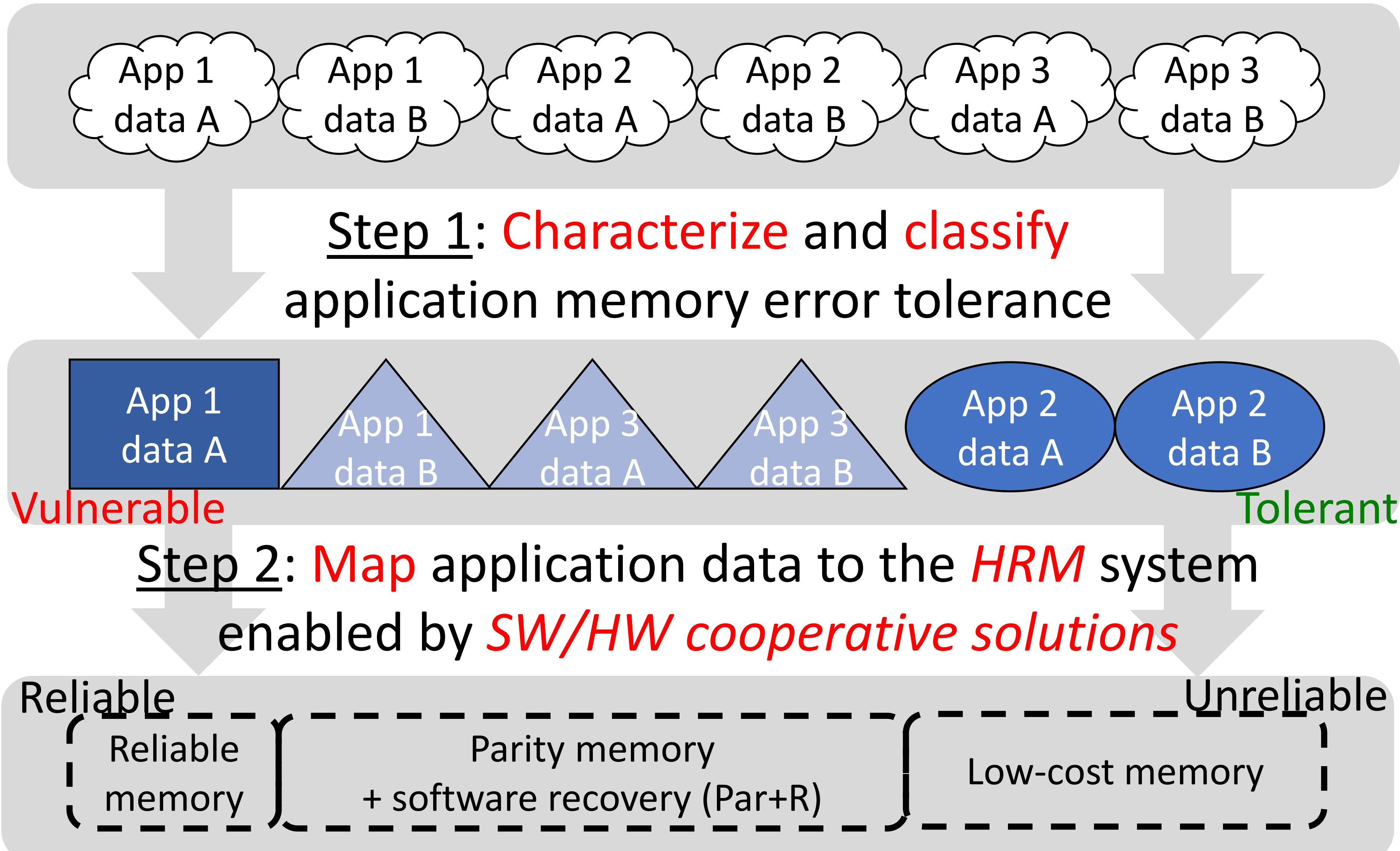
On Microsoft's Web Search workload

Reduces server hardware cost by 4.7 %

Achieves single server availability target of 99.90 %

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

Heterogeneous-Reliability Memory



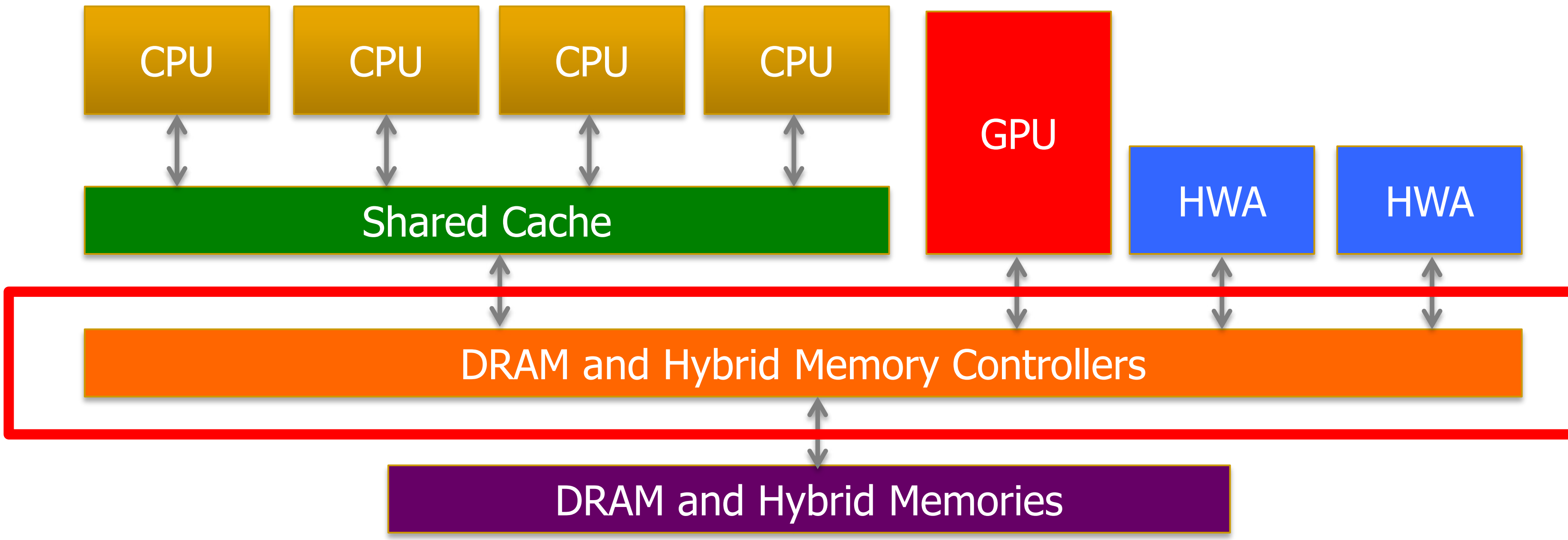
More on Heterogeneous-Reliability Memory

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

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

Yixin Luo Sriram Govindan* Bikash Sharma* Mark Santaniello* Justin Meza
Aman Kansal* Jie Liu* Badriddine Khessib* Kushagra Vaid* Onur Mutlu
Carnegie Mellon University, yixinluo@cs.cmu.edu, {meza, onur}@cmu.edu
*Microsoft Corporation, {srgovin, bsharma, marksan, kansal, jie.liu, bkhessib, kvaid}@microsoft.com

Data-Aware Cross-Layer Hybrid System Management



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

Another Example: EDEN for DNNs

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

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

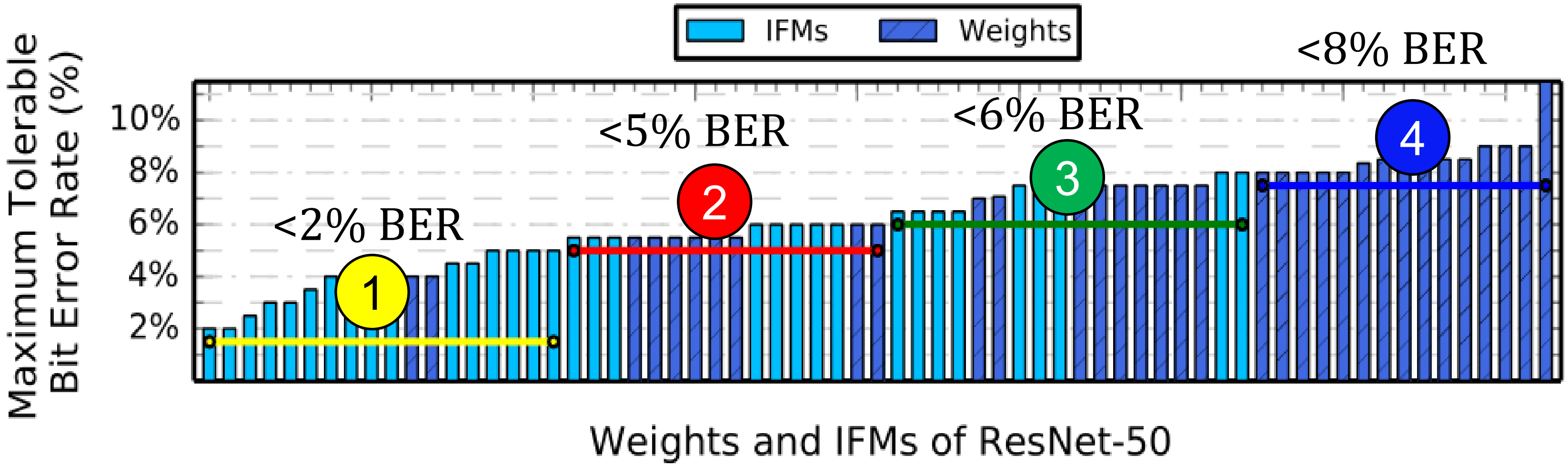
2. Reduce DRAM latency and voltage on such data and layers

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

Data-aware management of DRAM latency and voltage for Deep Neural Network Inference

Example DNN Data Type to DRAM Mapping

Mapping example of ResNet-50:



Map more error-tolerant DNN layers

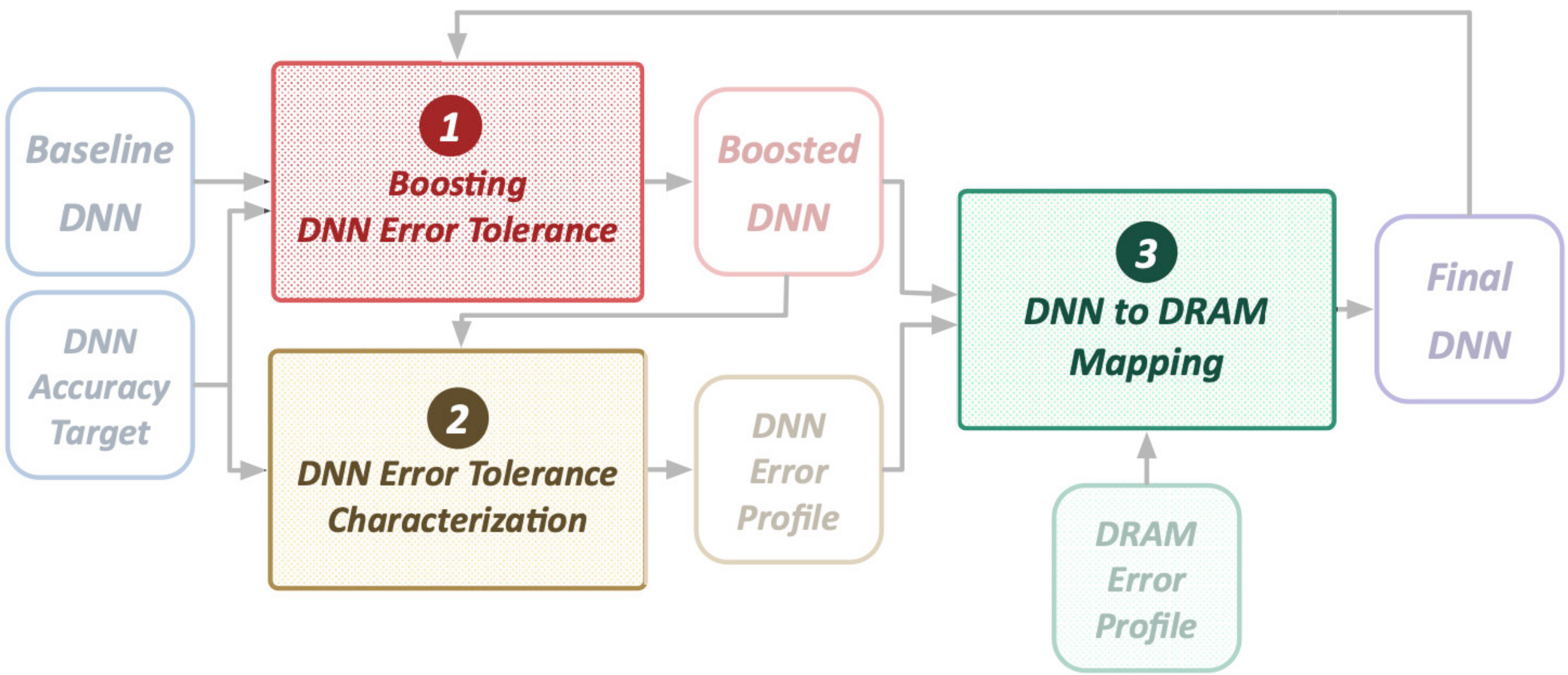
to DRAM partitions **with lower voltage/latency**

4 DRAM partitions with different error rates

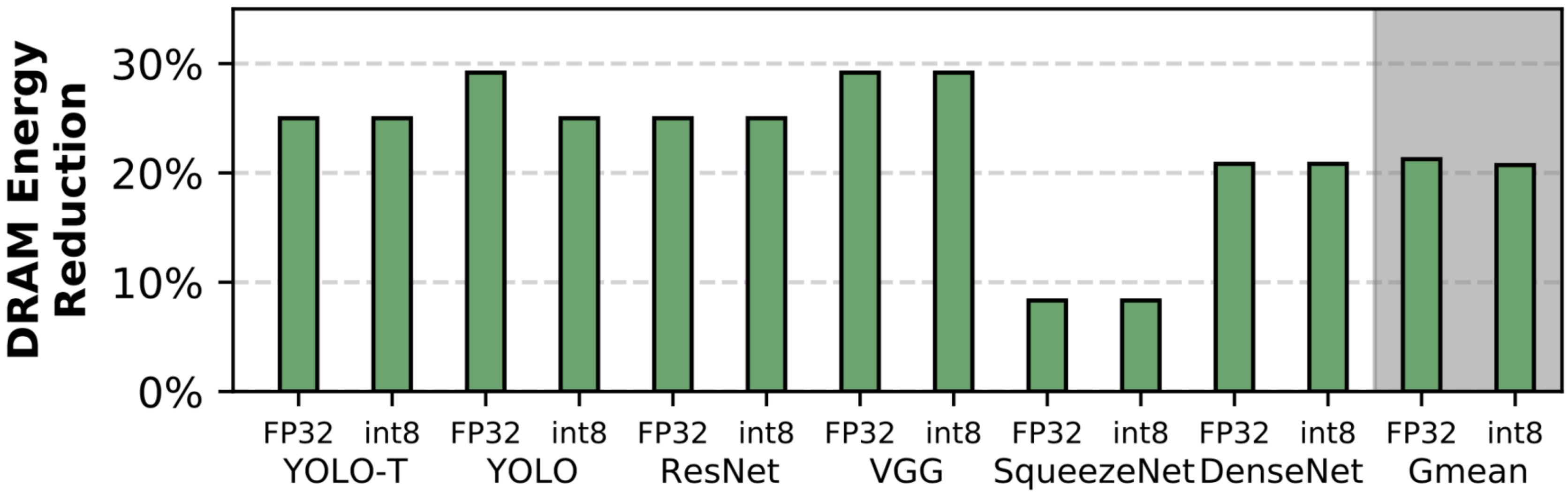
EDEN: Overview

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

EDEN is an **iterative** process that has **3 key steps**



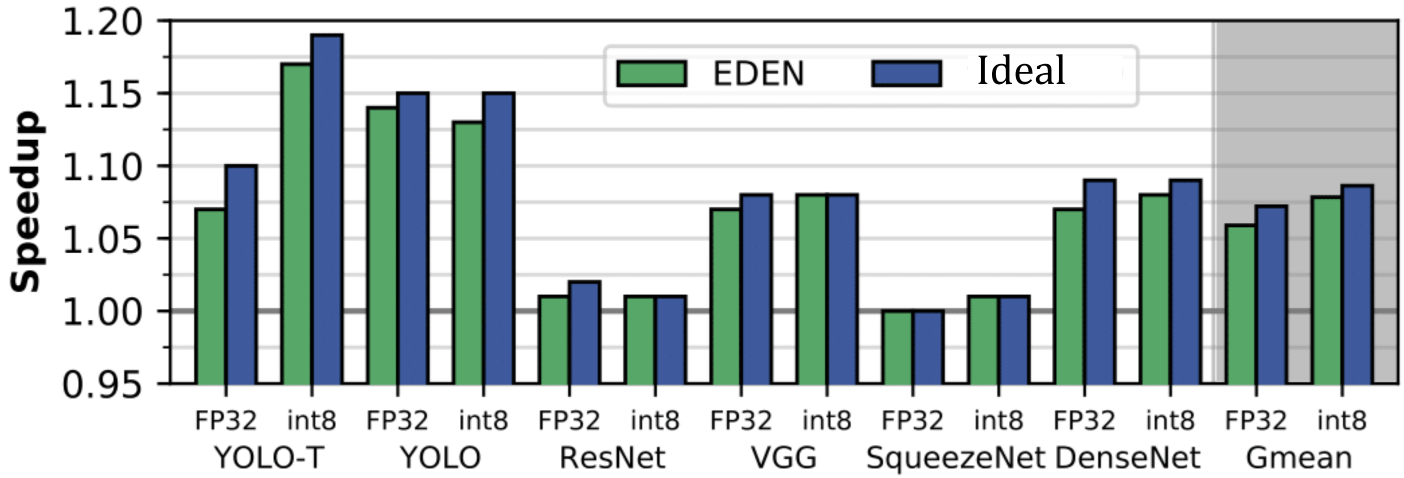
CPU: DRAM Energy Evaluation



Average 21% DRAM energy reduction

maintaining accuracy within 1% of original

CPU: Performance Evaluation



Average 8% system speedup

Some workloads achieve **17% speedup**

EDEN achieves **close to the ideal** speedup possible via tRCD scaling

GPU, Eyeriss, and TPU: Energy Evaluation

- **GPU:** average **37% energy reduction**
- **Eyeriss**: average **31% energy reduction**
- **TPU**: average **32% energy reduction**

EDEN: Data-Aware Efficient DNN Inference

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

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

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

SMASH: SW/HW Indexing Acceleration

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

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

Konstantinos Kanellopoulos[1] Nandita Vijaykumar[2,1] Christina Giannoula[1,3] Roknoddin Azizi[1] Skanda Koppula[1] Nika Mansouri Ghiasi[1] Taha Shahroodi[1] Juan Gomez Luna[1] Onur Mutlu[1,2]

[1]ETH Zürich [2]Carnegie Mellon University [3]National Technical University of Athens

Data-Aware Virtual Memory Framework

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

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

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

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

SW/HW Climate Modeling Accelerator

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

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

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

HW/SW Time Series Analysis Accelerator

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

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

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

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

FPGA-based Processing Near Memory

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

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

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

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

Accelerating Linked Data Structures

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

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

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

Accelerating Approximate String Matching

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

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

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

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

Accelerating Genome Analysis [IEEE MICRO 2020]

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

Accelerating Genome Analysis: A Primer on an Ongoing Journey

Mohammed Alser
ETH Zürich

Zülal Bingöl
Bilkent University

Damla Senol Cali
Carnegie Mellon University

Jeremie Kim
ETH Zurich and Carnegie Mellon University

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

Can Alkan
Bilkent University

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

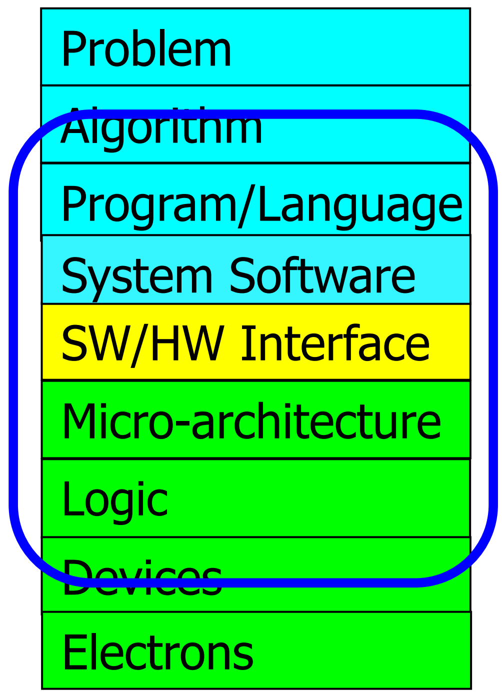
Challenge and Opportunity for Future

Data-Aware

(Expressive)

Computing Architectures

We Need to **Rethink** the Entire Stack



We can get there case by case

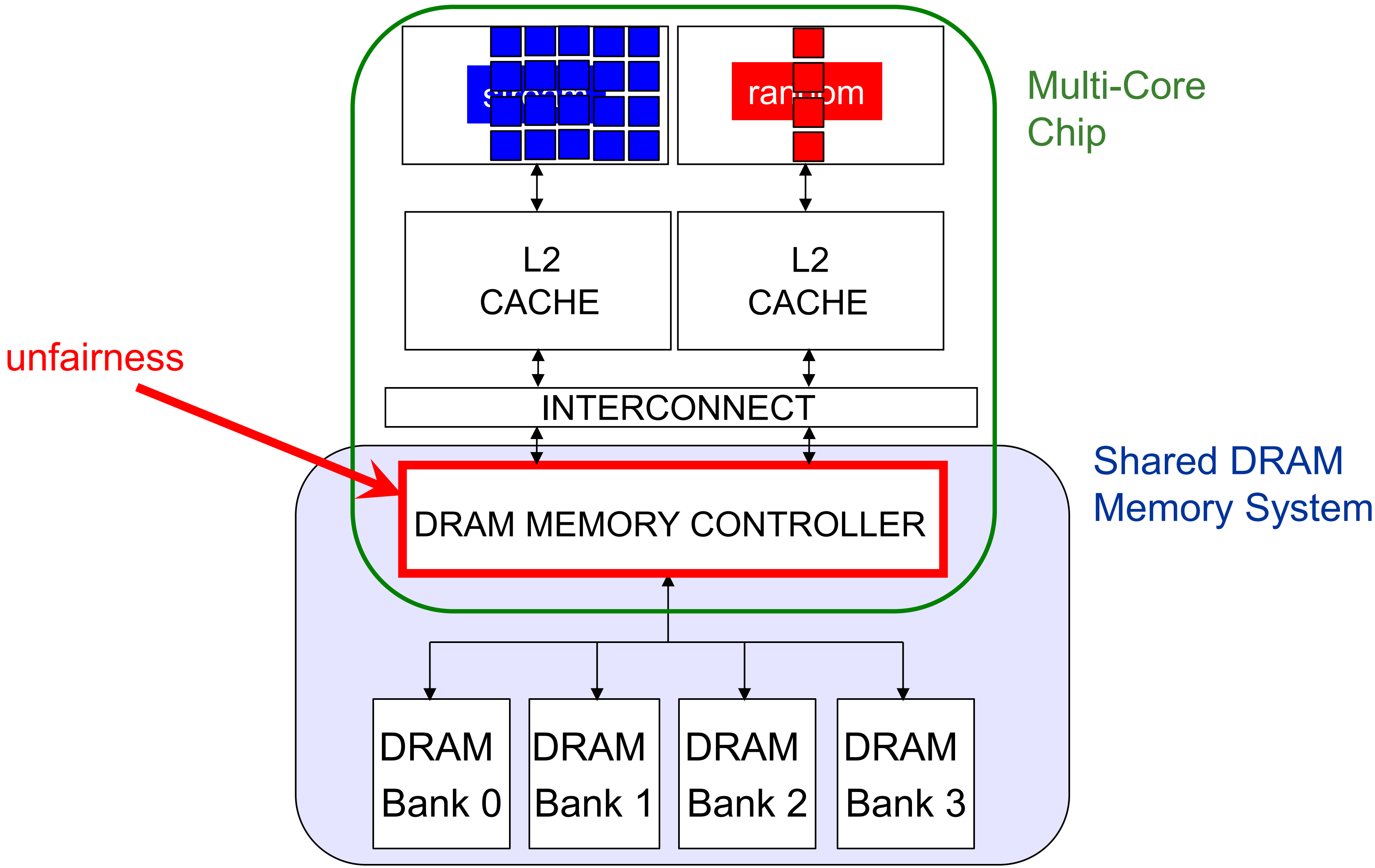
Memory Interference

Inter-Thread/Application Interference

- Problem: Threads share the memory system, but memory system does not distinguish between threads' requests

- Existing memory systems
  - Free-for-all, shared based on demand
  - Control algorithms thread-unaware and thread-unfair
  - Aggressive threads can deny service to others
  - Do not try to reduce or control inter-thread interference

Uncontrolled Interference: An Example



A Memory Performance Hog

```
// initialize large arrays A, B

for (j=0; j<N; j++) {
    index = j*linesize;   streaming
    A[index] = B[index];
    …
}
```

```
// initialize large arrays A, B

for (j=0; j<N; j++) {
    index = rand();   random
    A[index] = B[index];
    …
}
```

STREAM

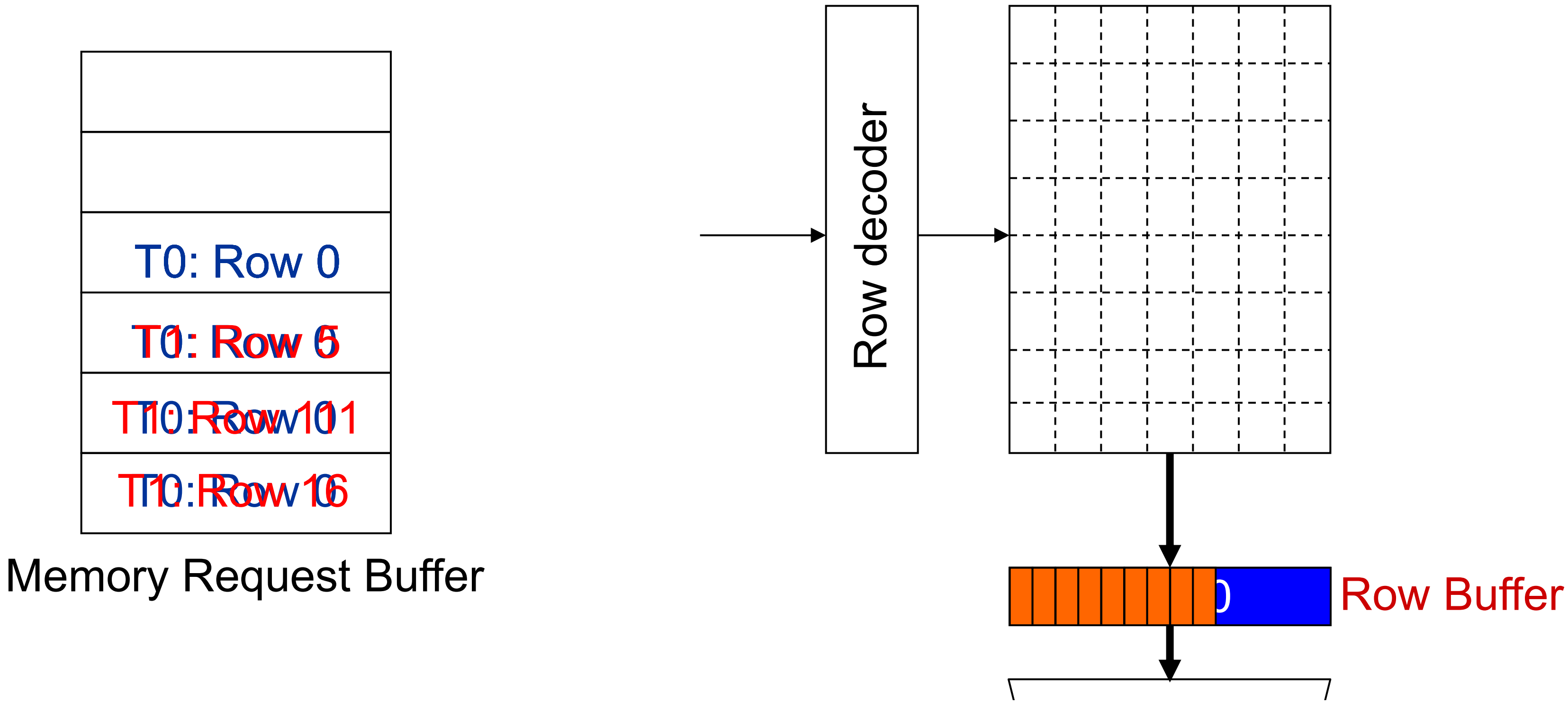
- Sequential memory access
- Very high row buffer locality (96% hit rate)
- Memory intensive

RANDOM

- Random memory access
- Very low row buffer locality (3% hit rate)
- Similarly memory intensive

Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

What Does the Memory Hog Do?

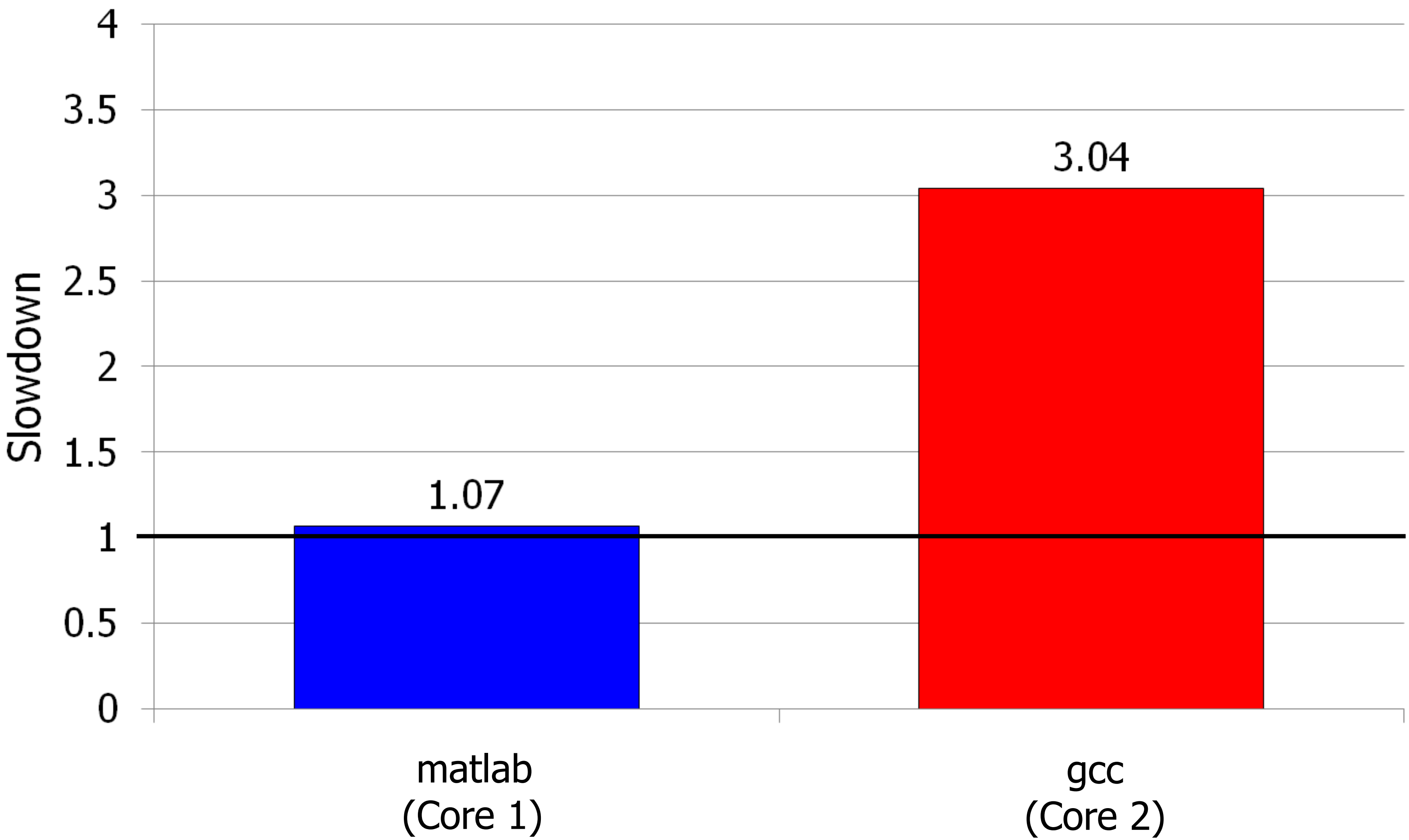


Row size: 8KB, cache block size: 64B

128 (8KB/64B) requests of T0 serviced before T1

Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

Unfair Slowdowns due to Interference



Moscibroda and Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," USENIX Security 2007.

DRAM Controllers

- A row-conflict memory access takes significantly longer than a row-hit access

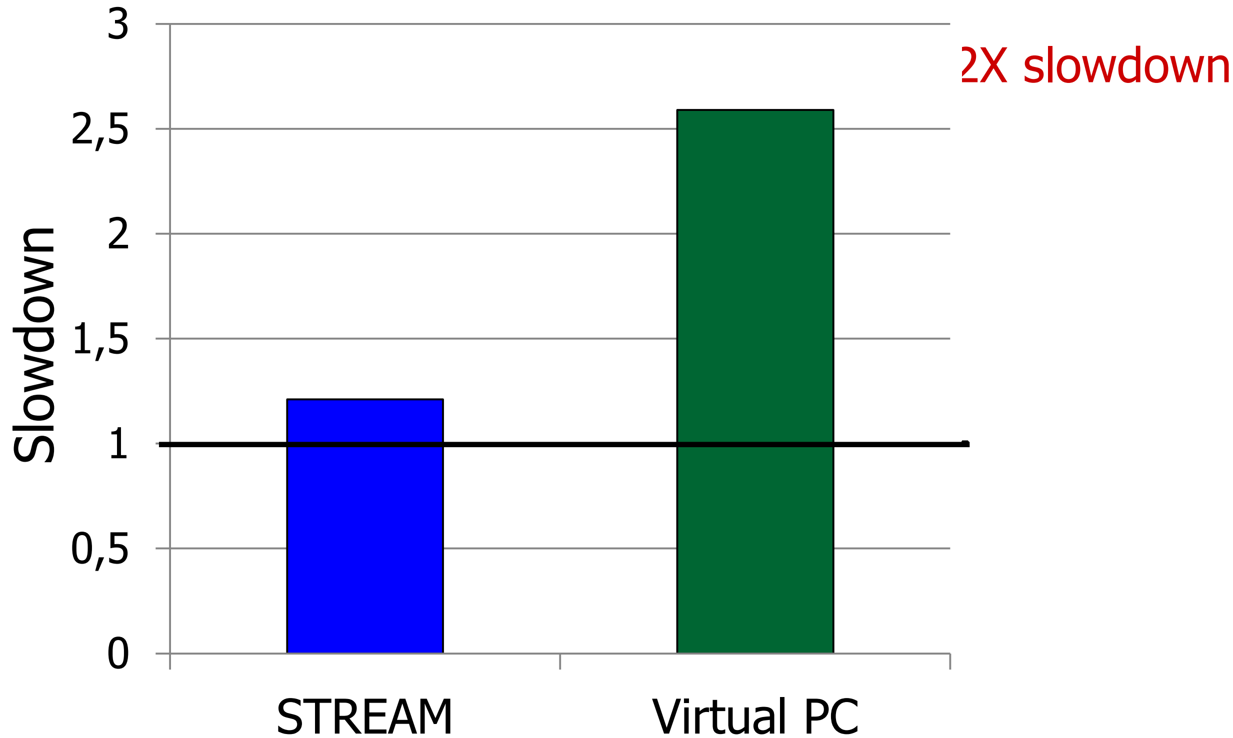
- Current controllers take advantage of the row buffer

- Commonly used scheduling policy (FR-FCFS) [Rixner 2000]*
  (1) Row-hit first: Service row-hit memory accesses first
  (2) Oldest-first: Then service older accesses first

- This scheduling policy aims to maximize DRAM throughput
  - But, it is unfair when multiple threads share the DRAM system

*Rixner et al., "Memory Access Scheduling," ISCA 2000.
*Zuravleff and Robinson, "Controller for a synchronous DRAM …," US Patent 5,630,096, May 1997.

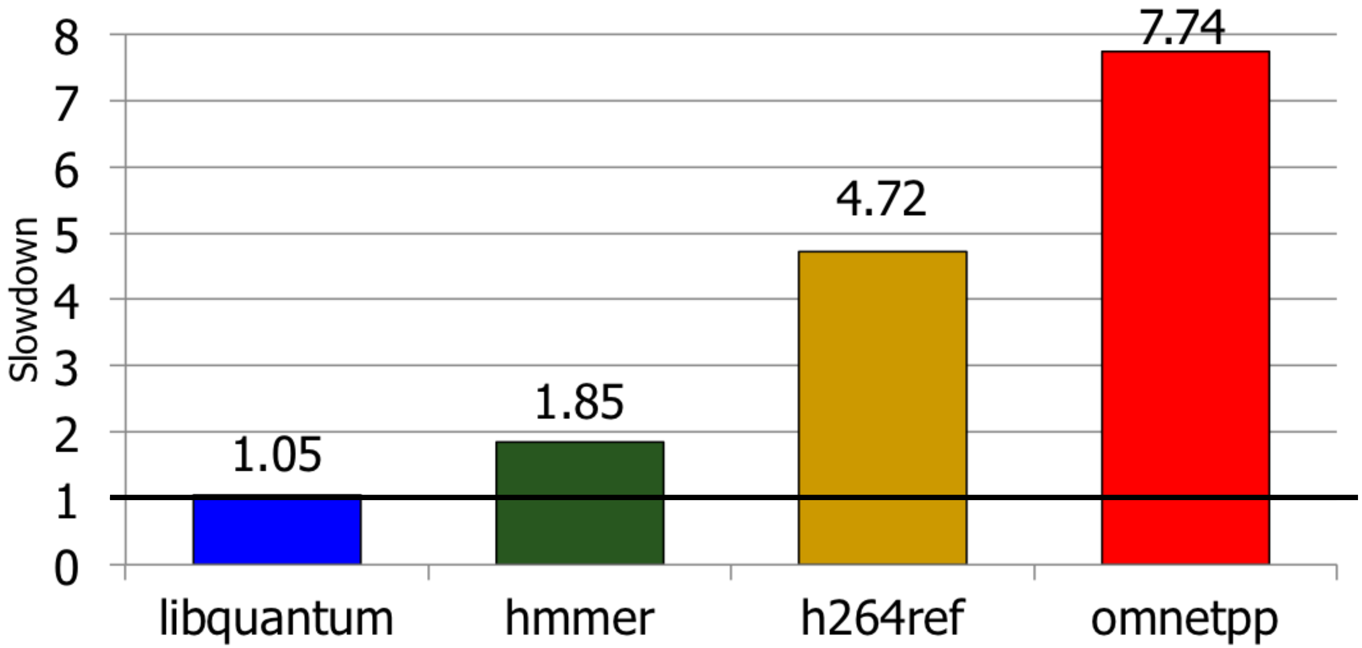
Effect of the Memory Performance Hog



Results on Intel Pentium D running Windows XP
(Similar results for Intel Core Duo and AMD Turion, and on Fedora Linux)

Moscibroda and Mutlu, "Memory Performance Attacks," USENIX Security 2007.

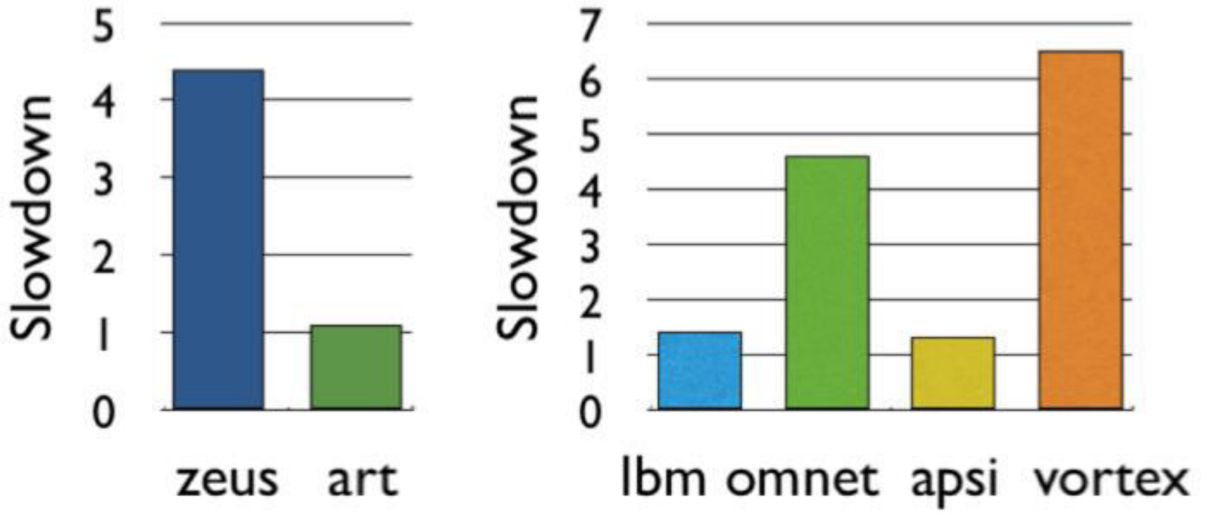
Greater Problem with More Cores



- Vulnerable to denial of service (DoS)
- Unable to enforce priorities or SLAs
- Low system performance

Uncontrollable, unpredictable system

Greater Problem with More Cores



- Vulnerable to denial of service (DoS)
- Unable to enforce priorities or SLAs
- Low system performance

Uncontrollable, unpredictable system

More on Memory Performance Attacks

- Thomas Moscibroda and Onur Mutlu,
  "Memory Performance Attacks: Denial of Memory Service in Multi-Core Systems"
  *Proceedings of the 16th USENIX Security Symposium (**USENIX SECURITY**)*, pages 257-274, Boston, MA, August 2007. Slides (ppt)

Memory Performance Attacks:
Denial of Memory Service in Multi-Core Systems

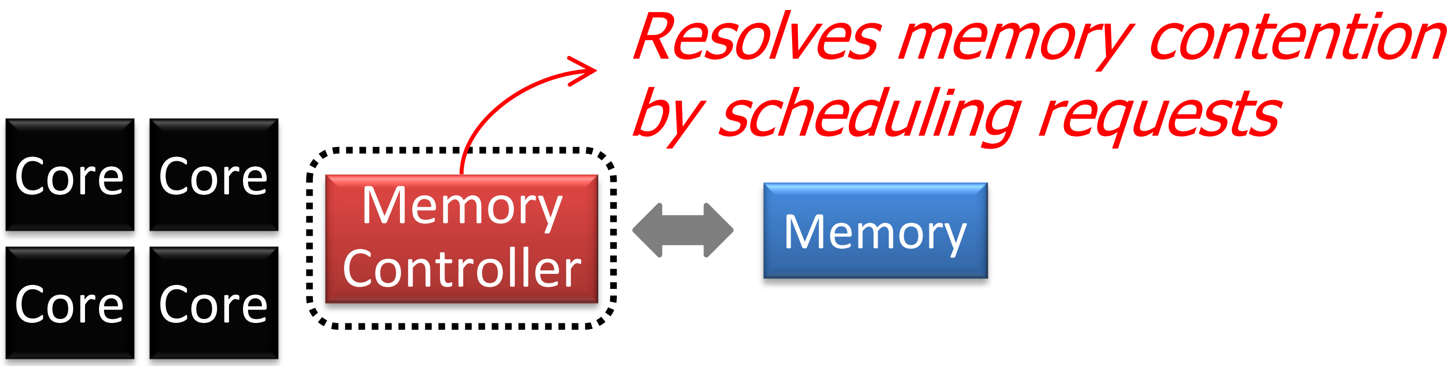
Thomas Moscibroda   Onur Mutlu
Microsoft Research
{*moscitho,onur*}@*microsoft.com*

How Do We Solve The Problem?

- Inter-thread interference is uncontrolled in all memory resources
  - Memory controller
  - Interconnect
  - Caches

- We need to control it
  - i.e., design an interference-aware (QoS-aware) memory system

QoS-Aware Memory Scheduling



- How to schedule requests to provide
  - High system performance
  - High fairness to applications
  - Configurability to system software

- Memory controller needs to be aware of threads

QoS-Aware Memory: Readings (I)

- Onur Mutlu and Thomas Moscibroda,
  "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors"
  Proceedings of the 40th International Symposium on Microarchitecture (***MICRO***), pages 146-158, Chicago, IL, December 2007. [Summary] [Slides (ppt)]

Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors

Onur Mutlu   Thomas Moscibroda

Microsoft Research
{onur,moscitho}@microsoft.com

QoS-Aware Memory: Readings (II)

- Onur Mutlu and Thomas Moscibroda,
  "Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems"
  *Proceedings of the 35th International Symposium on Computer Architecture (**ISCA**)*, pages 63-74, Beijing, China, June 2008.
  [Summary] [Slides (ppt)]
  [Lecture Slides (pptx) (pdf)]
  [Lecture Video (1 hr 16 mins), 8 October 2020]
  One of the 12 computer architecture papers of 2008 selected as Top Picks by IEEE Micro.

Parallelism-Aware Batch Scheduling:
Enhancing both Performance and Fairness of Shared DRAM Systems

Onur Mutlu Thomas Moscibroda
Microsoft Research
{onur,moscitho}@microsoft.com

QoS-Aware Memory: Readings (III)

- Yoongu Kim, Dongsu Han, Onur Mutlu, and Mor Harchol-Balter, **"ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers"**
Proceedings of the 16th International Symposium on High-Performance Computer Architecture (***HPCA***), Bangalore, India, January 2010. Slides (pptx)
Best paper session. One of the four papers nominated for the Best Paper Award by the Program Committee.

ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers

Yoongu Kim Dongsu Han Onur Mutlu Mor Harchol-Balter

Carnegie Mellon University

QoS-Aware Memory: Readings (IV)

- Yoongu Kim, Michael Papamichael, Onur Mutlu, and Mor Harchol-Balter, **"Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior"**
  *Proceedings of the 43rd International Symposium on Microarchitecture (**MICRO**)*, pages 65-76, Atlanta, GA, December 2010. Slides (pptx) (pdf)
  One of the 11 computer architecture papers of 2010 selected as Top Picks by IEEE Micro.

Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior

Yoongu Kim yoonguk@ece.cmu.edu
Michael Papamichael papamix@cs.cmu.edu
Onur Mutlu onur@cmu.edu
Mor Harchol-Balter harchol@cs.cmu.edu

Carnegie Mellon University

QoS-Aware Memory: Readings (V)

- Lavanya Subramanian, Donghyuk Lee, Vivek Seshadri, Harsha Rastogi, and Onur Mutlu,
"The Blacklisting Memory Scheduler: Achieving High Performance and Fairness at Low Cost"
*Proceedings of the 32nd IEEE International Conference on Computer Design (**ICCD**)*, Seoul, South Korea, October 2014.
[Slides (pptx) (pdf)]

The Blacklisting Memory Scheduler: Achieving High Performance and Fairness at Low Cost

Lavanya Subramanian, Donghyuk Lee, Vivek Seshadri, Harsha Rastogi, Onur Mutlu
Carnegie Mellon University
{lsubrama,donghyu1,visesh,harshar,onur}@cmu.edu

QoS-Aware Memory: Readings (VI)

- Lavanya Subramanian, Donghyuk Lee, Vivek Seshadri, Harsha Rastogi, and Onur Mutlu,
"BLISS: Balancing Performance, Fairness and Complexity in Memory Access Scheduling"
IEEE Transactions on Parallel and Distributed Systems (***TPDS***), to appear in 2016. arXiv.org version, April 2015.
An earlier version as *SAFARI Technical Report*, TR-SAFARI-2015-004, Carnegie Mellon University, March 2015.
[Source Code]

BLISS: Balancing Performance, Fairness and Complexity in Memory Access Scheduling

Lavanya Subramanian, Donghyuk Lee, Vivek Seshadri, Harsha Rastogi, and Onur Mutlu

QoS-Aware Memory: Readings (VII)

- Rachata Ausavarungnirun, Kevin Chang, Lavanya Subramanian, Gabriel Loh, and Onur Mutlu,
  "Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems"
  *Proceedings of the 39th International Symposium on Computer Architecture (**ISCA**)*, Portland, OR, June 2012. Slides (pptx)

Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems

Rachata Ausavarungnirun† Kevin Kai-Wei Chang† Lavanya Subramanian† Gabriel H. Loh‡ Onur Mutlu†

†Carnegie Mellon University
{rachata,kevincha,lsubrama,onur}@cmu.edu

‡Advanced Micro Devices, Inc.
gabe.loh@amd.com

QoS-Aware Memory: Readings (VIII)

- Hiroyuki Usui, Lavanya Subramanian, Kevin Kai-Wei Chang, and Onur Mutlu,
  "DASH: Deadline-Aware High-Performance Memory Scheduler for Heterogeneous Systems with Hardware Accelerators"
  ACM Transactions on Architecture and Code Optimization (***TACO***), Vol. 12, January 2016.
  Presented at the 11th HiPEAC Conference, Prague, Czech Republic, January 2016.
  [Slides (pptx) (pdf)]
  [Source Code]

DASH: Deadline-Aware High-Performance Memory Scheduler for Heterogeneous Systems with Hardware Accelerators

HIROYUKI USUI, LAVANYA SUBRAMANIAN, KEVIN KAI-WEI CHANG, and ONUR MUTLU, Carnegie Mellon University

QoS-Aware Memory: Readings (IX)

- Lavanya Subramanian, Vivek Seshadri, Yoongu Kim, Ben Jaiyen, and Onur Mutlu,
"MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems"
Proceedings of the 19th International Symposium on High-Performance Computer Architecture (***HPCA***), Shenzhen, China, February 2013. Slides (pptx)

MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems

Lavanya Subramanian Vivek Seshadri Yoongu Kim Ben Jaiyen Onur Mutlu
Carnegie Mellon University

QoS-Aware Memory: Readings (X)

- Lavanya Subramanian, Vivek Seshadri, Arnab Ghosh, Samira Khan, and Onur Mutlu,
"The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory"
*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]

The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory

Lavanya Subramanian*§ Vivek Seshadri* Arnab Ghosh*† Samira Khan*‡ Onur Mutlu*

*Carnegie Mellon University §Intel Labs †IIT Kanpur ‡University of Virginia