SAFARI Project & Seminars Courses

Exploration of Emerging Memory Systems

Haocong Luo

Prof. Onur Mutlu

ETH Zürich

Fall 2022

4 October 2021

Meeting 1:
Introduction and Logistics

Haocong Luo
Prof. Onur Mutlu
ETH Zürich
Fall 2022
4 October 2021

Content

- You will learn in detail how **modern memory systems** operate.
- You will **design new DRAM and memory controller mechanisms** for improving overall system **performance**, **energy** consumption, and **reliability**.
- You will **simulate and understand** the memory system behavior of **modern workloads** such as machine learning, graph analytics, genome analysis.

Key Takeaways

- This P&S is aimed at improving your
  - Knowledge in Computer Architecture and Memory Systems
  - Technical skills in simulating memory systems
  - Critical thinking and analysis
  - Interaction with a nice group of researchers
  - Familiarity with key research directions
  - Technical presentation of your project

Key Goal

- Learn how state-of-the-art memory controllers operate
- Design new DRAM and memory controller mechanisms
- Evaluate your mechanisms using simulation

Prerequisites of the Course

- Digital Design and Computer Architecture (or equivalent course)
  - High-level understanding of how modern computers are organized.
  - Knows about how DRAM works.
- A comfortable knowledge in C++ programming language
- Interest in making things efficient and solving problems
- Interest in understanding software development and hardware design, and their interaction

Course Info: Who Are We? (I)

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

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

Course Info: Who Are We? (II)

- Lead Supervisor:
  - Haocong (Richard) Luo

- Supervisors:
  - Geraldo de Oliveira
  - Giray Yaglikci
  - Ataberk Olgun
  - Nisa Bostanci

- Get to know us and our research
  - https://safari.ethz.ch/safari-group/

Onur Mutlu's SAFARI Research Group

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

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

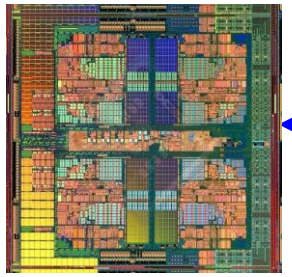
Think BIG, Aim HIGH!

https://safari.ethz.ch

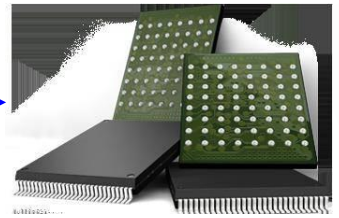
Current Research Focus Areas

***Research Focus:* Computer architecture, HW/SW, bioinformatics**

- *Memory and storage (DRAM, flash, emerging), interconnects*
- *Heterogeneous & parallel systems, GPUs, systems for data analytics*
- *System/architecture interaction, new execution models, new interfaces*
- *Energy efficiency, fault tolerance, hardware security, performance*
- *Genome sequence analysis & assembly algorithms and architectures*
- *Biologically inspired systems & system design for bio/medicine*

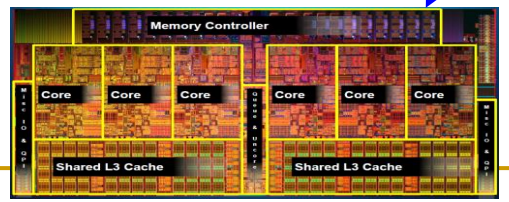


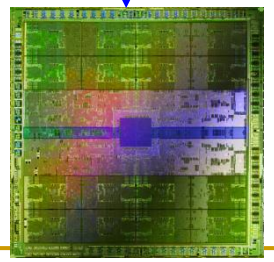
Hybrid Main Memory



Persistent Memory/Storage

Heterogeneous Processors and Accelerators





Graphics and Vision Processing

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

Course Website

- https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=ramulator
- Useful information about the course
- Check your email frequently for announcements

Course Schedule (Tentative)

- **Week 1 (This meeting):** Introduction and logistics.
  - Read the course materials.
  - Start to think about your own project ideas and write to us for comments.
- **Week 2:** Introduction to available projects defined by us.
- **Week 3:** Ramulator Tutorial
  - You should have picked or proposed a project by this meeting.
- **Week 4 - 8:** Past paper presentations.
  - Update meetings w/ your supervisor.
- **Week 9:** Milestone presentation.
  - Share your progress. What have you achieved? What challenges are you facing?
- **Week 10+:** Update meetings w/ your supervisor.
- **Date TBD:** Final presentation.

Project Proposal

- **Motivation**
  - Why do you want to work on this project?
  - Why is it important? Any insight from prior work? Is there a problem to solve?

- **Goal**
  - What do you want to achieve?
  - Be precise and practical.
  - If possible, define a primary goal (i.e., meet the goal minimally) and secondary goals.

- **Expected Results**
  - What are the deliverables of the project?
  - What do you plan to show in the final presentation?

Course Requirements and Expectations

- Attendance highly recommended for all meetings
  - You should actively schedule meetings w/ supervisors to discuss updates
- Study the learning materials
- Each student will carry out a hands-on project
  - Build, implement, code, and design with close engagement from the supervisors
- Participation
  - Ask questions, contribute thoughts/ideas
  - Read relevant papers

We will help in all projects!

If your work is really good, you may get it published!

Course Info: How About You?

- Let us know your background, interests
- Why did you join this P&S?
- **Please submit HW0**
  - **Due: Oct. 5**

Learning materials:

- An old version of Ramulator: https://github.com/CMU-SAFARI/ramulator
- Original Ramulator paper: https://people.inf.ethz.ch/omutlu/pub/ramulator_dram_simulator-ieee-cal15.pdf
- An example study of modern workloads and DRAM architectures using Ramulator: https://people.inf.ethz.ch/omutlu/pub/Workload-DRAM-Interaction-Analysis_sigmetrics19_pomacs19.pdf
- An example recent study of enhancing the interface between the DRAM and the memory controller using Ramulator: https://arxiv.org/abs/2207.13358
- An example recent study of a new DRAM architecture using Ramulator: https://people.inf.ethz.ch/omutlu/pub/CLR-DRAM_capacity-latency-reconfigurable-DRAM_isca20.pdf
- An example recent study of a new virtual memory system architecture using Ramulator: https://people.inf.ethz.ch/omutlu/pub/VBI-virtual-block-interface_isca20.pdf
- Three examples of new ideas enabled by Ramulator based evaluation:
  - https://people.inf.ethz.ch/omutlu/pub/rowclone_micro13.pdf
  - https://people.inf.ethz.ch/omutlu/pub/salp-dram_isca12.pdf
  - https://people.inf.ethz.ch/omutlu/pub/raidr-dram-refresh_isca12.pdf

Performance Assessment

We expect you to:

- **Learn** how DRAM operates and how to analyze performance of memory systems using simulation
- **Achieve the goals** of your project
- **Deliver** your **code and results** with sufficient documentation
- Prepare a **final presentation** and present your work to SAFARI

The Problem

Data access is the major performance and energy bottleneck

Our current

design principles

cause great energy waste

(and great performance loss)

The Problem

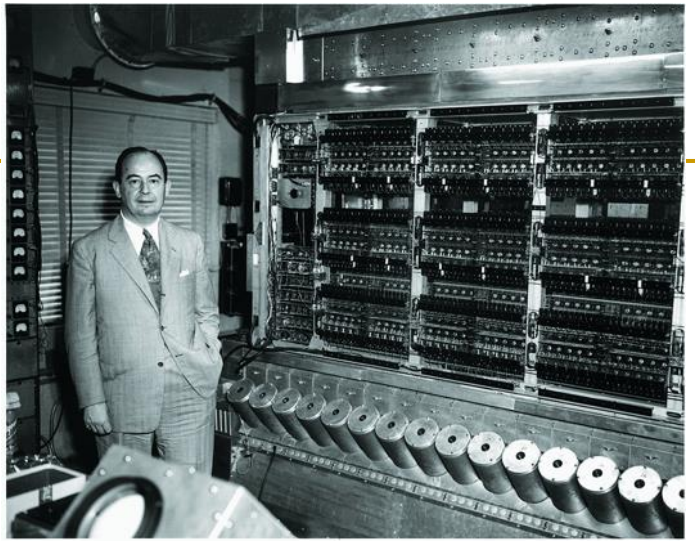
Processing of data

is performed

far away from the data

A Computing System

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



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

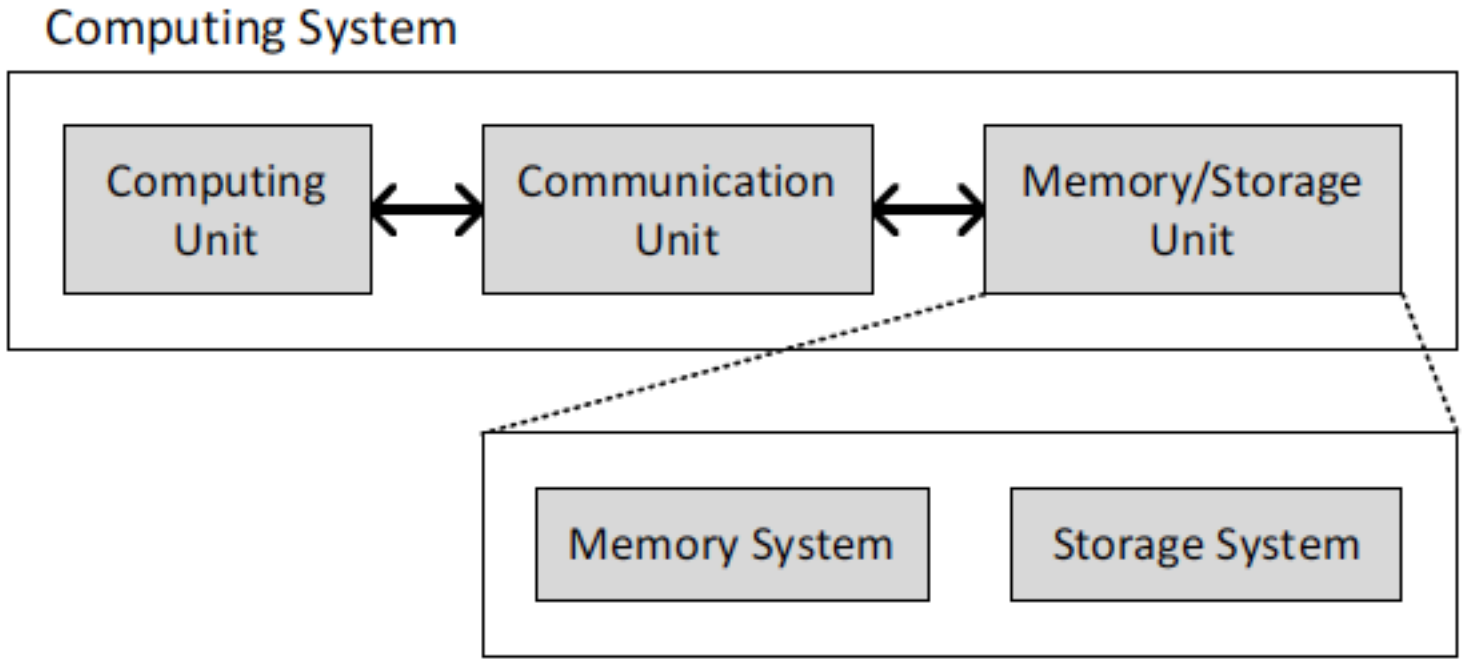
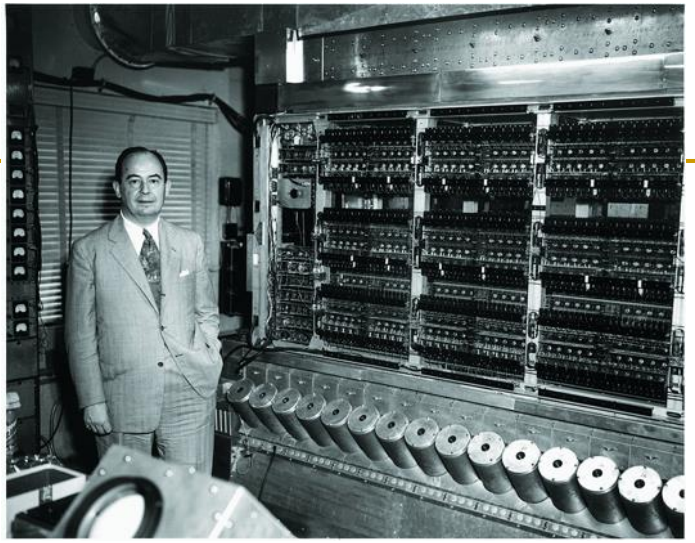


Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

A Computing System

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



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

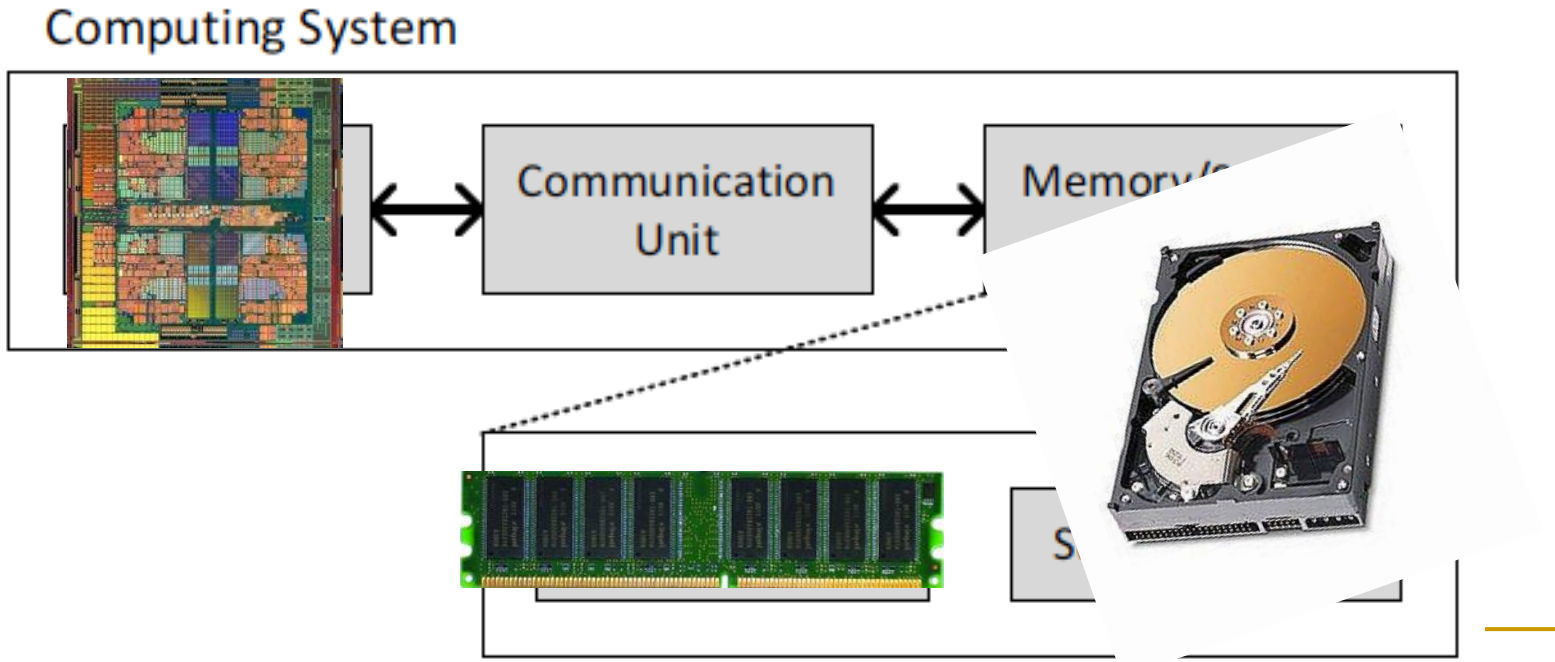
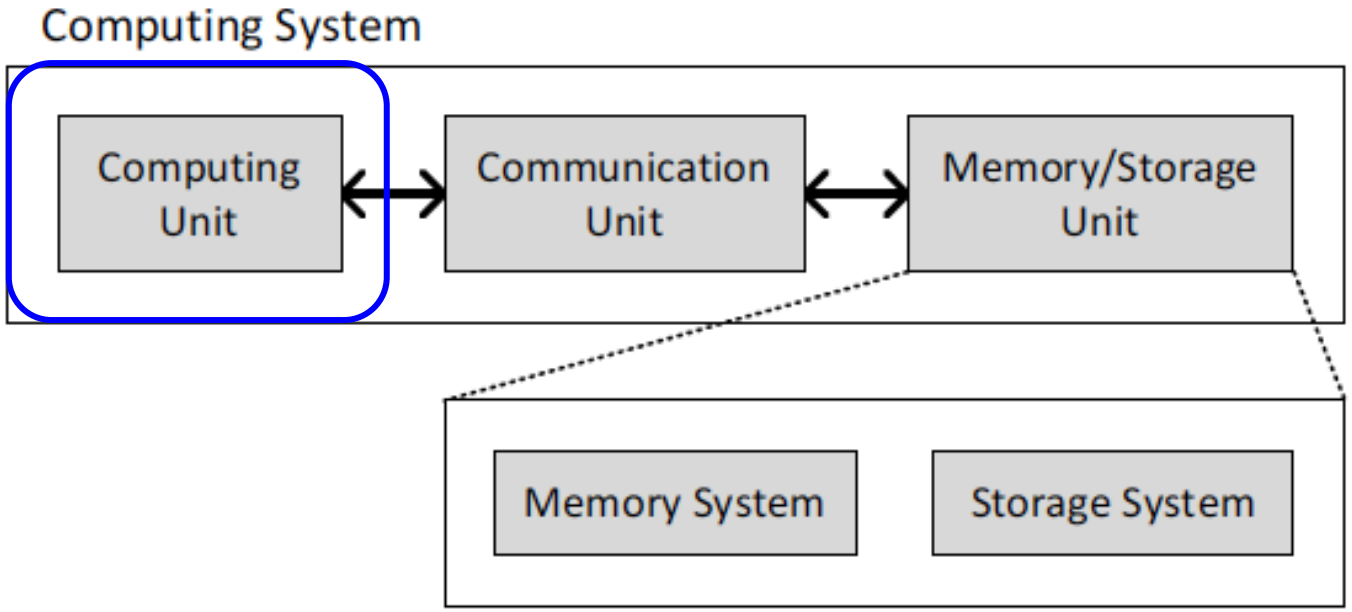


Image source: https://lbsitbytes2010.wordpress.com/2013/03/29/john-von-neumann-roll-no-15/

Today's Computing Systems

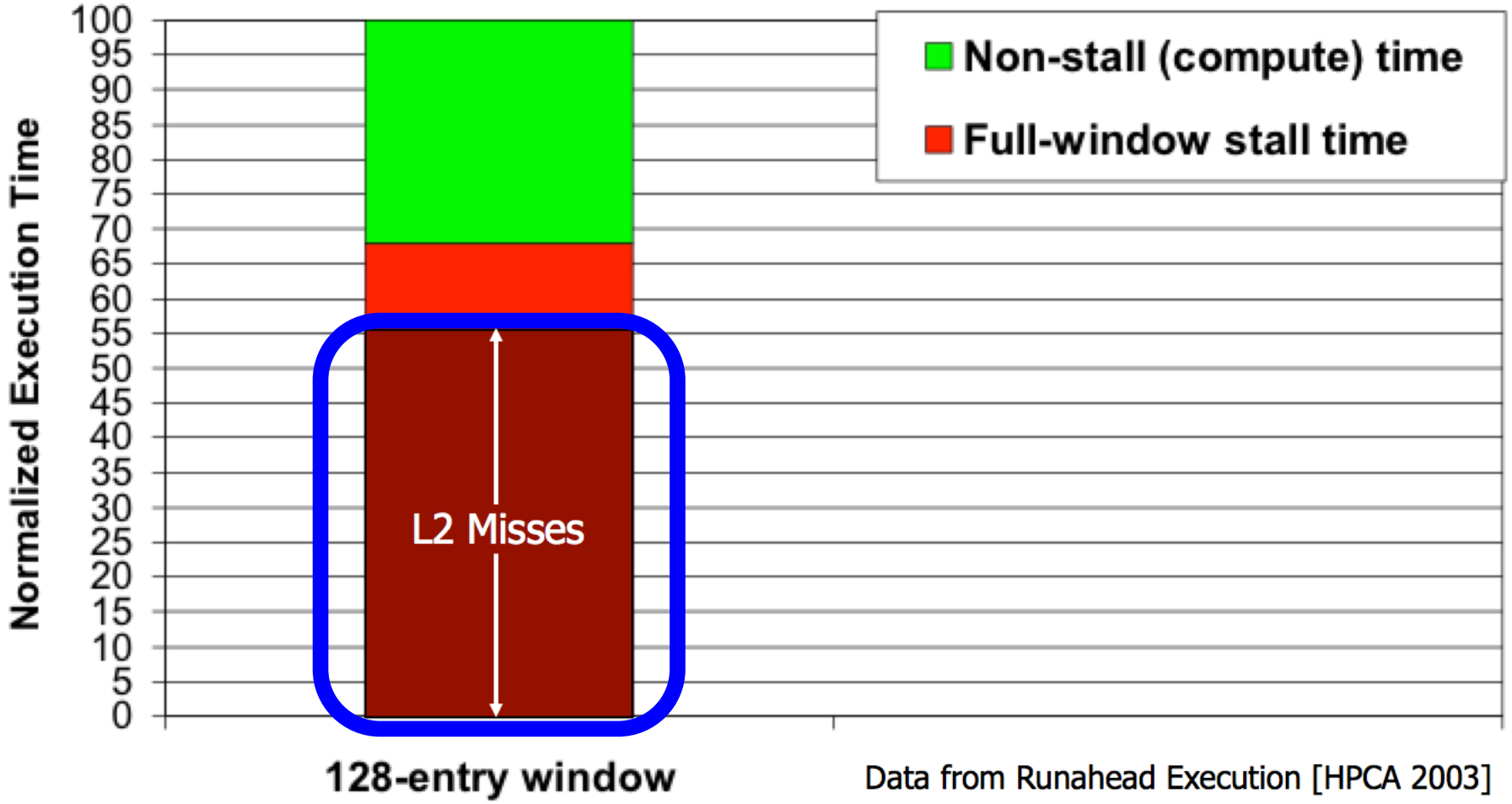
- Are overwhelmingly processor centric
- All data processed in the processor → at great system cost
- Processor is heavily optimized and is considered the master
- Data storage units are dumb and are largely unoptimized (except for some that are on the processor die)



Yet …

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

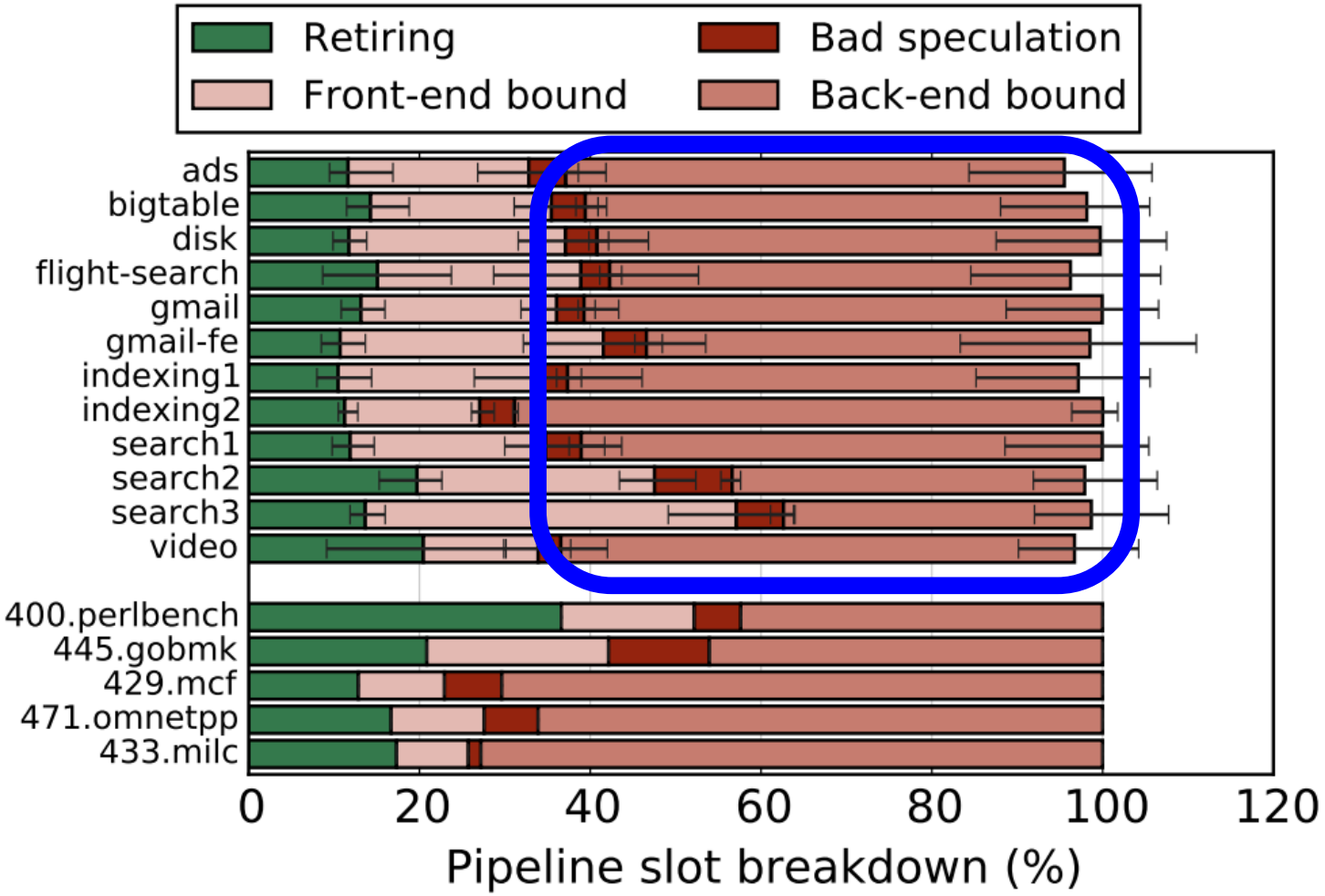
- "**It's the Memory, Stupid!**" (Richard Sites, MPR, 1996)



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

The Performance Perspective (Today)

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



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

The Performance Perspective (Today)

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

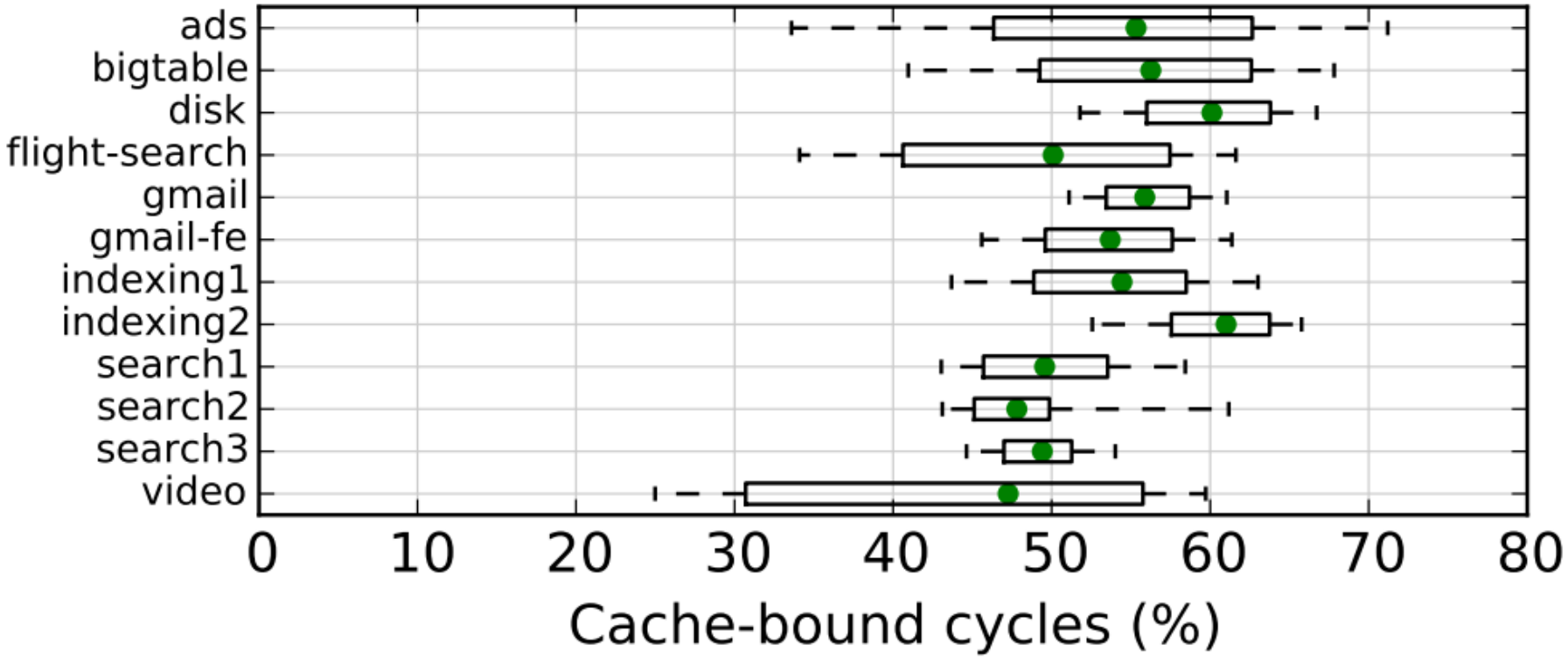
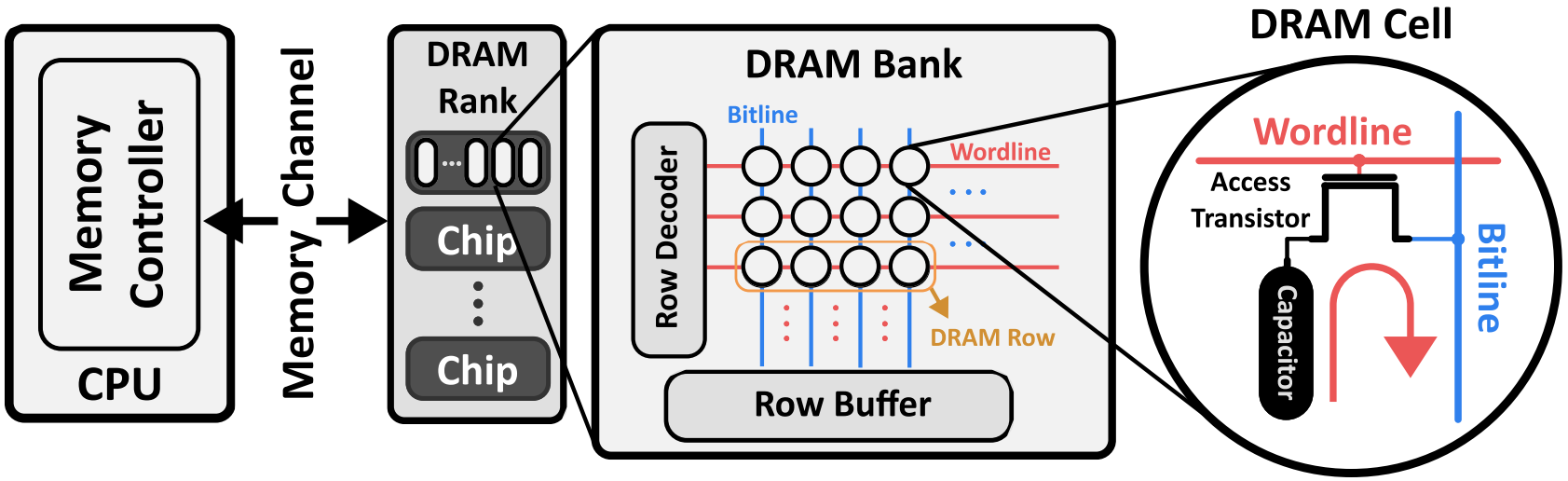


Figure 11: Half of cycles are spent stalled on caches.

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

DRAM Organization

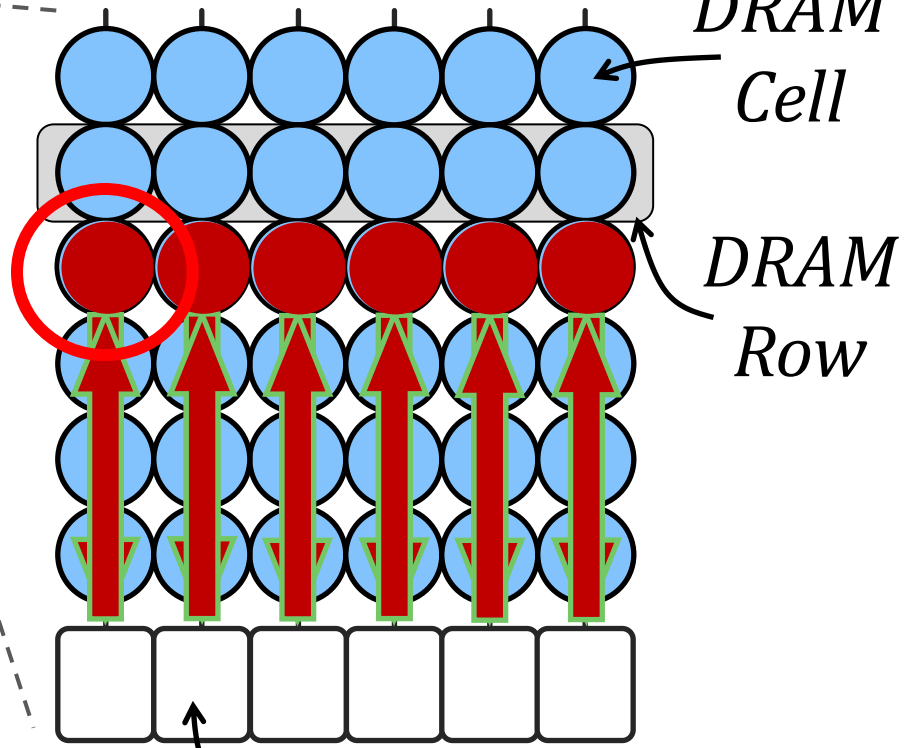


DRAM Operations

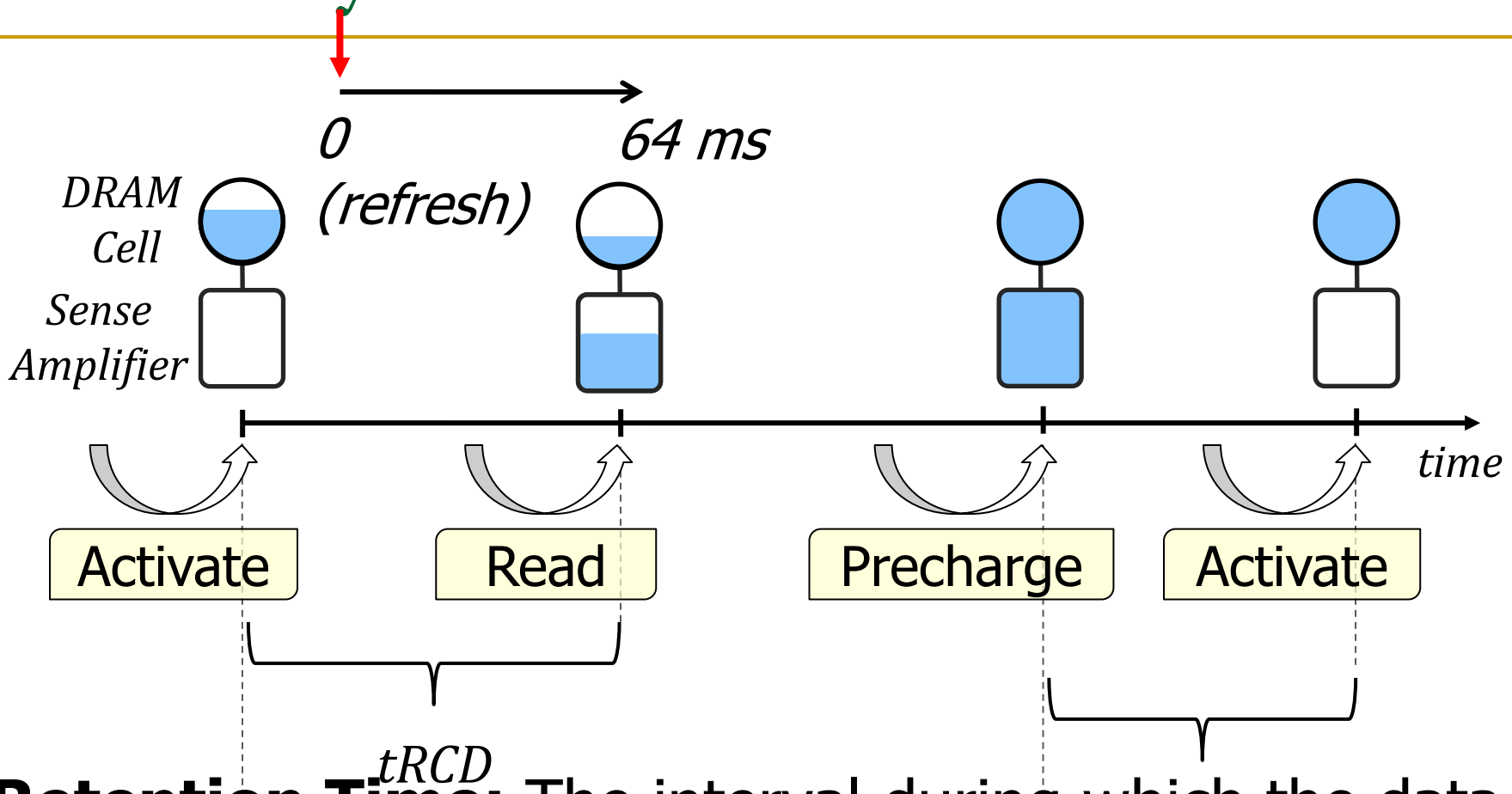
Memory Bus

Precharge

CPU



DRAM Latency



Retention Time: The interval during which the data is retained correctly in the DRAM cell without accessing it

DRAM Latency

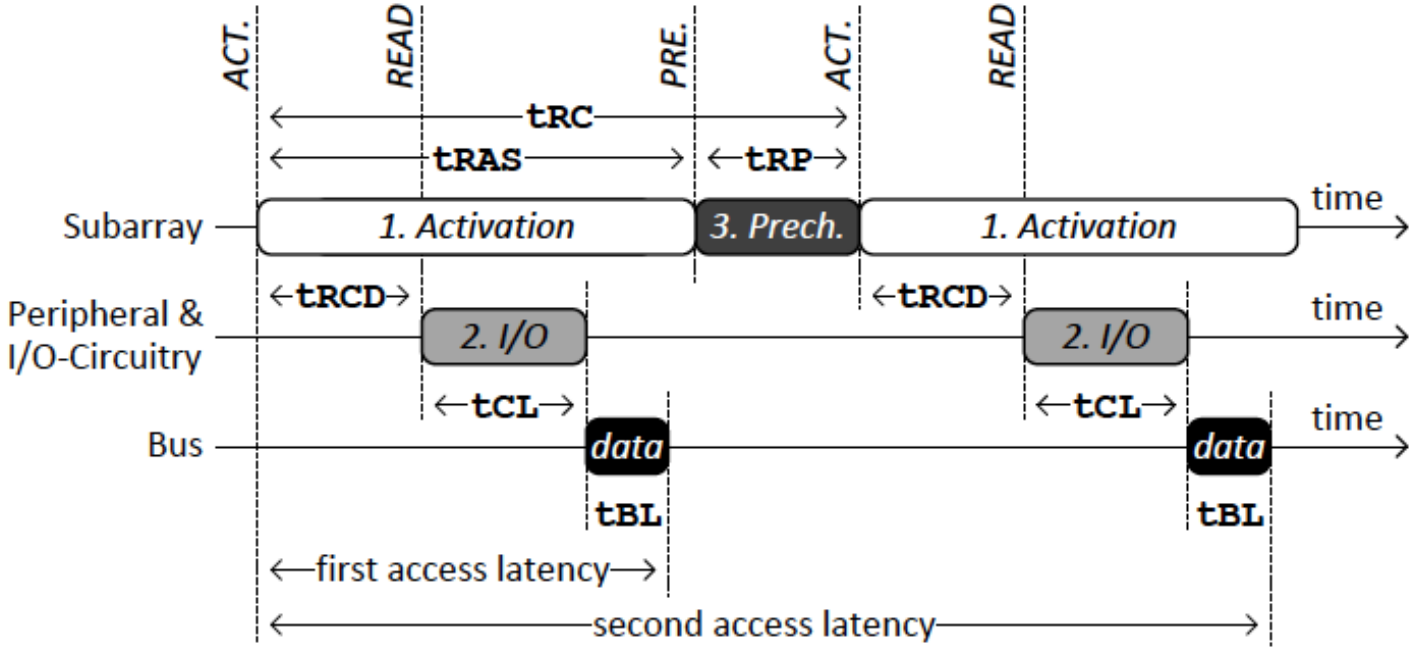


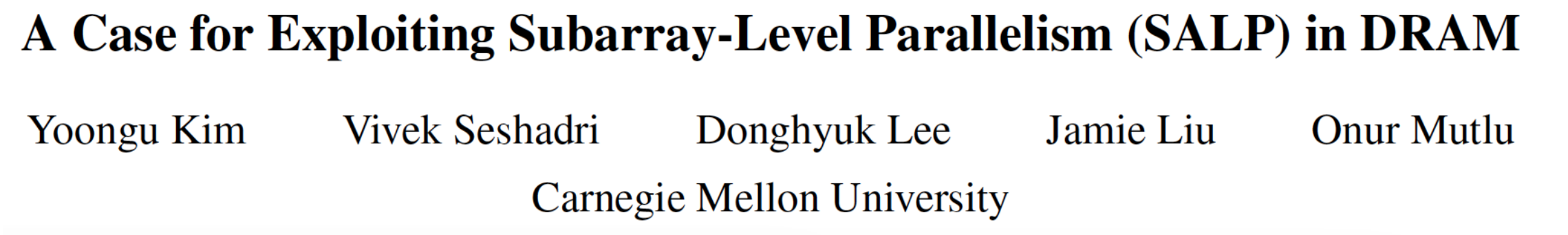
Figure 5. Three Phases of DRAM Access

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

SALP

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



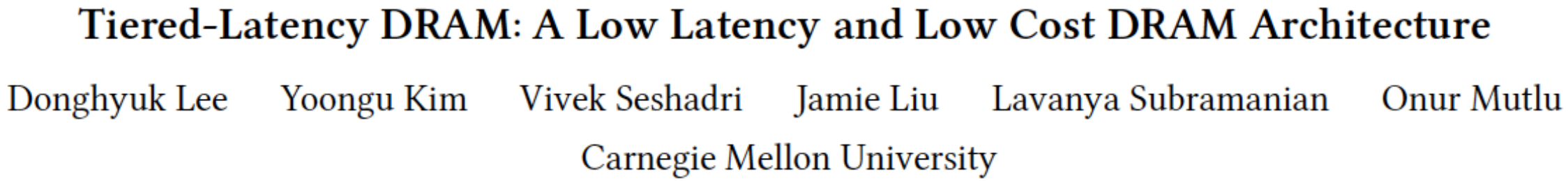
A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM

Yoongu Kim Vivek Seshadri Donghyuk Lee Jamie Liu Onur Mutlu

Carnegie Mellon University

TL-DRAM

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



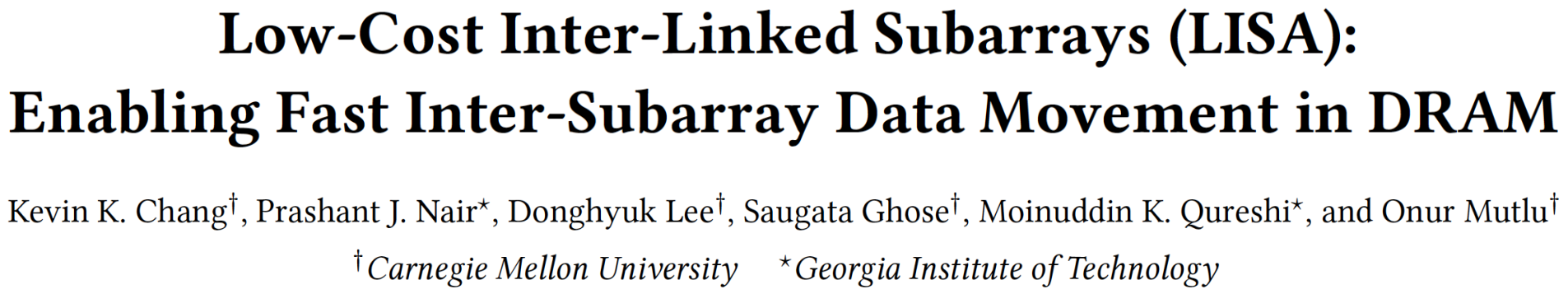
Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture

Donghyuk Lee Yoongu Kim Vivek Seshadri Jamie Liu Lavanya Subramanian Onur Mutlu

Carnegie Mellon University

LISA

- Kevin K. Chang, Prashant J. Nair, Saugata Ghose, Donghyuk Lee, Moinuddin K. Qureshi, and Onur Mutlu,
  "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM"
  *Proceedings of the 22nd International Symposium on High-Performance Computer Architecture (**HPCA**)*, Barcelona, Spain, March 2016.
  [Slides (pptx) (pdf)]
  [Source Code]



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

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

†*Carnegie Mellon University* ⋆*Georgia Institute of Technology*

CROW

- Hasan Hassan, Minesh Patel, Jeremie S. Kim, A. Giray Yaglikci, Nandita Vijaykumar, Nika Mansourighiasi, Saugata Ghose, and Onur Mutlu,
  "CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability"
  *Proceedings of the 46th International Symposium on Computer Architecture (**ISCA**)*, Phoenix, AZ, USA, June 2019.
  [Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Poster (pptx) (pdf)]
  [Lightning Talk Video (3 minutes)]
  [Full Talk Video (16 minutes)]
  [Source Code for CROW (Ramulator and Circuit Modeling)]

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

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

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

CLR-DRAM: Capacity-Latency Reconfigurability

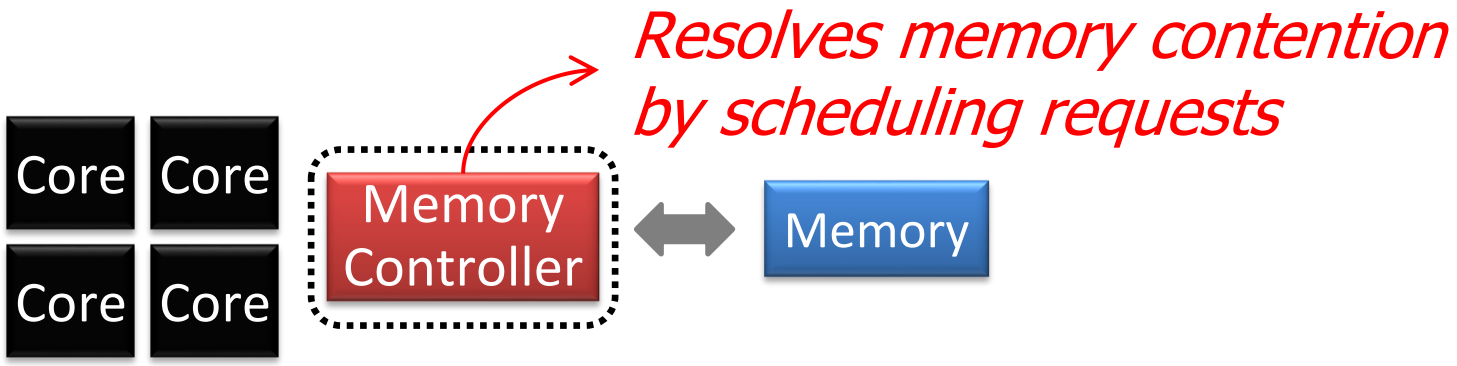
- Haocong Luo, Taha Shahroodi, Hasan Hassan, Minesh Patel, A. Giray Yaglikci, Lois Orosa, Jisung Park, and Onur Mutlu,
  "CLR-DRAM: A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-Off"
  *Proceedings of the 47th International Symposium on Computer Architecture (**ISCA**)*, Valencia, Spain, June 2020.
  [Slides (pptx) (pdf)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (20 minutes)]
  [Lightning Talk Video (3 minutes)]

CLR-DRAM: A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-Off

Haocong Luo[§†] Taha Shahroodi[§] Hasan Hassan[§] Minesh Patel[§]
A. Giray Yağlıkçı[§] Lois Orosa[§] Jisung Park[§] Onur Mutlu[§]

[§]*ETH Zürich* [†]*ShanghaiTech University*

Memory Controller



How to schedule requests to maximize system performance?

The Problem

- Multiple applications share the DRAM controller
- DRAM controllers designed to maximize DRAM data throughput

- DRAM scheduling policies are unfair to some applications
  - Row-hit first: unfairly prioritizes apps with high row buffer locality
    - Threads that keep on accessing the same row
  - Oldest-first: unfairly prioritizes memory-intensive applications

- DRAM controller vulnerable to denial-of-service (DOS) attacks
  - Can write programs to exploit unfairness

QoS-Aware Memory Scheduling: Evolution

- Stall-time fair memory scheduling [Mutlu+ MICRO'07]
  - Idea: Estimate and balance thread slowdowns
  - Takeaway: Proportional thread progress improves performance, especially when threads are "heavy" (memory intensive)

- Parallelism-aware batch scheduling [Mutlu+ ISCA'08, Top Picks'09]
  - Idea: Rank threads and service in rank order (to preserve bank parallelism); batch requests to prevent starvation
  - Takeaway: Preserving within-thread bank-parallelism improves performance; request batching improves fairness

- ATLAS memory scheduler [Kim+ HPCA'10]
  - Idea: Prioritize threads that have attained the least service from the memory scheduler
  - Takeaway: Prioritizing "light" threads improves performance

QoS-Aware Memory Scheduling: Evolution

- Thread cluster memory scheduling [Kim+ MICRO'10, Top Picks'11]
  - Idea: Cluster threads into two groups (latency vs. bandwidth sensitive); prioritize the latency-sensitive ones; employ a fairness policy in the bandwidth sensitive group
  - Takeaway: Heterogeneous scheduling policy that is different based on thread behavior maximizes both performance and fairness

- Integrated Memory Channel Partitioning and Scheduling [Muralidhara+ MICRO'11]
  - Idea: Only prioritize very latency-sensitive threads in the scheduler; mitigate all other applications' interference via channel partitioning
  - Takeaway: Intelligently combining application-aware channel partitioning and memory scheduling provides better performance than either

QoS-Aware Memory Scheduling: Evolution

- Parallel application memory scheduling [Ebrahimi+ MICRO'11]
  - Idea: Identify and prioritize limiter threads of a multithreaded application in the memory scheduler; provide fast and fair progress to non-limiter threads
  - Takeaway: Carefully prioritizing between limiter and non-limiter threads of a parallel application improves performance

- Staged memory scheduling [Ausavarungnirun+ ISCA'12]
  - Idea: Divide the functional tasks of an application-aware memory scheduler into multiple distinct stages, where each stage is significantly simpler than a monolithic scheduler
  - Takeaway: Staging enables the design of a scalable and relatively simpler application-aware memory scheduler that works on very large request buffers

QoS-Aware Memory Scheduling: Evolution

- MISE: Memory Slowdown Model [Subramanian+ HPCA'13]
  - Idea: Estimate the performance of a thread by estimating its change in memory request service rate when run alone vs. shared → use this simple model to estimate slowdown to design a scheduling policy that provides predictable performance or fairness
  - Takeaway: Request service rate of a thread is a good proxy for its performance; alone request service rate can be estimated by giving high priority to the thread in memory scheduling for a while

- ASM: Application Slowdown Model [Subramanian+ MICRO'15]
  - Idea: Extend MISE to take into account cache+memory interference
  - Takeaway: Cache access rate of an application can be estimated accurately and is a good proxy for application performance

QoS-Aware Memory Scheduling: Evolution

- BLISS: Blacklisting Memory Scheduler [Subramanian+ ICCD'14, TPDS'16]
  - Idea: Deprioritize (i.e., blacklist) a thread that has consecutively serviced a large number of requests
  - Takeaway: Blacklisting greatly reduces interference enables the scheduler to be simple without requiring full thread ranking

- DASH: Deadline-Aware Memory Scheduler [Usui+ TACO'16]
  - Idea: Balance prioritization between CPUs, GPUs and Hardware Accelerators (HWA) by keeping HWA progress in check vs. deadlines such that HWAs do not hog performance and appropriately distinguishing between latency-sensitive vs. bandwidth-sensitive CPU workloads
  - Takeaway: Proper control of HWA progress and application-aware CPU prioritization leads to better system performance while meeting HWA deadlines

QoS-Aware Memory Scheduling: Evolution

- Prefetch-aware shared resource management [Ebrahimi+ ISCA'11] [Ebrahimi+ MICRO'09] [Ebrahimi+ HPCA'09] [Lee+ MICRO'08'09]
  - Idea: Prioritize prefetches depending on how they affect system performance; even accurate prefetches can degrade performance of the system
  - Takeaway: Carefully controlling and prioritizing prefetch requests improves performance and fairness

- DRAM-Aware last-level cache policies and write scheduling [Lee+ HPS Tech Report'10] [Seshadri+ ISCA'14]
  - Idea: Design cache eviction and replacement policies such that they proactively exploit the state of the memory controller and DRAM (e.g., proactively evict data from the cache that hit in open rows)
  - Takeaway: Coordination of last-level cache and DRAM policies improves performance and fairness; writes should not be ignored

QoS-Aware Memory Scheduling: Evolution

- FIRM: Memory Scheduling for NVM [Zhao+ MICRO'14]
  - Idea: Carefully handle write-read prioritization with coarse-grained batching and application-aware scheduling
  - Takeaway: Carefully controlling and prioritizing write requests improves performance and fairness; write requests are especially critical in NVMs

- Criticality-Aware Memory Scheduling for GPUs [Jog+ SIGMETRICS'16]
  - Idea: Prioritize latency-critical cores' requests in a GPU system
  - Takeaway: Need to carefully balance locality and criticality to make sure performance improves by taking advantage of both

- Worst-case Execution Time Based Memory Scheduling for Real-Time Systems [Kim+ RTAS'14, JRTS'16]

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

Meeting 1:
Introduction and Logistics

Haocong Luo
Prof. Onur Mutlu
ETH Zürich
Fall 2022
4 October 2021

SAFARI Project & Seminars Courses

Exploration of Emerging Memory Systems

Haocong Luo

Prof. Onur Mutlu

ETH Zürich

Fall 2022

4 October 2021