# Digital Design & Computer Arch.

## Lecture 3a: Mysteries in Comp. Arch.

Prof. Onur Mutlu

ETH Zürich

Spring 2021

4 March 2021

# Some Example "Mysteries"

# Four Mysteries: Familiar with Any?

- **Meltdown & Spectre (2017-2018)**

- **Rowhammer (2012-2014)**

- **Memories Forget: Refresh (2011-2012)**

- **Memory Performance Attacks (2006-2007)**

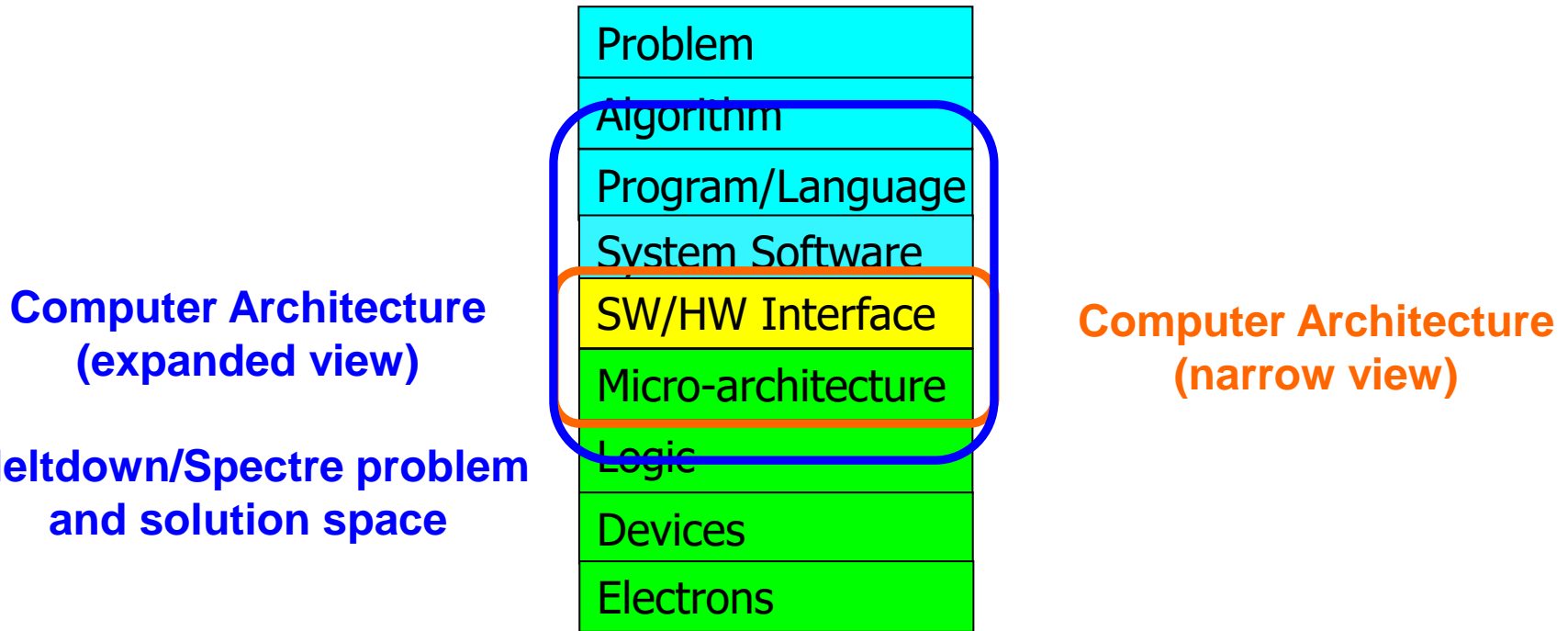# Mystery #1:
## Meltdown & Spectre

# What Are These?



MELTDOWN

SPECTRE

# Meltdown/Spectre Span Across the Hierarchy



**Computer Architecture (expanded view)**

**Meltdown/Spectre problem and solution space**

Problem
Algorithm
Program/Language
System Software
SW/HW Interface
Micro-architecture
Logic
Devices
Electrons

**Computer Architecture (narrow view)**

# Two Major Goals of This Course

- ❑ Enable you to think critically

- ❑ Enable you to think broadly

# Mystery #2:  RowHammer

# The Story of RowHammer

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

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

**WIRED**  Forget Software—Now Hackers Are Exploiting Physics

| BUSINESS | CULTURE | DESIGN | GEAR | SCIENCE |
|---|---|---|---|---|

ANDY GREENBERG  SECURITY  08.31.16  7:00 AM

SHARE

# FORGET SOFTWARE—NOW HACKERS ARE EXPLOITING PHYSICS
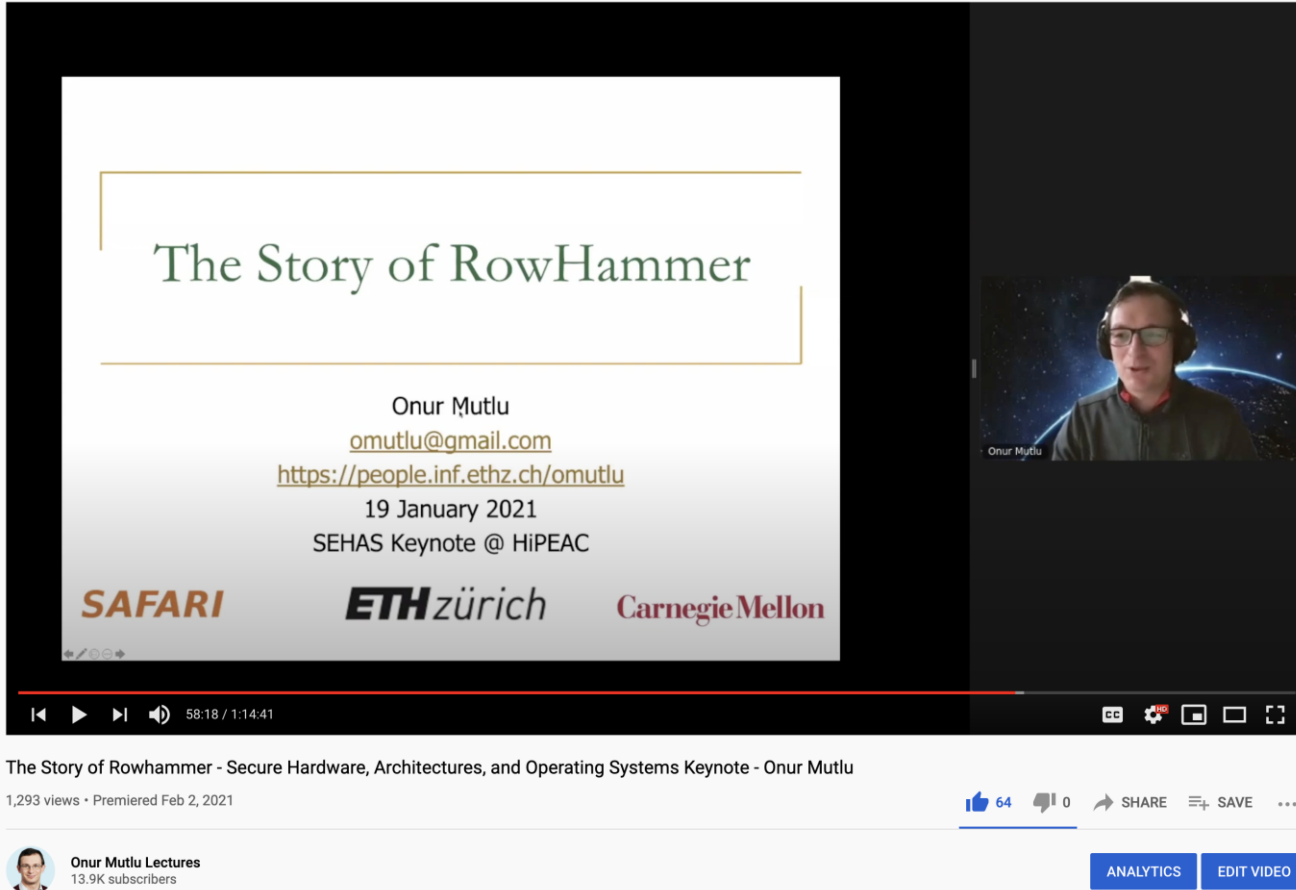
SHARE
18276

TWEET

# Security Implications



Rowhammer

It's like breaking into an apartment by repeatedly slamming a neighbor's door until the vibrations open the door you were after

# The Story of RowHammer Lecture ...

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

# Four Mysteries: Familiar with Any?
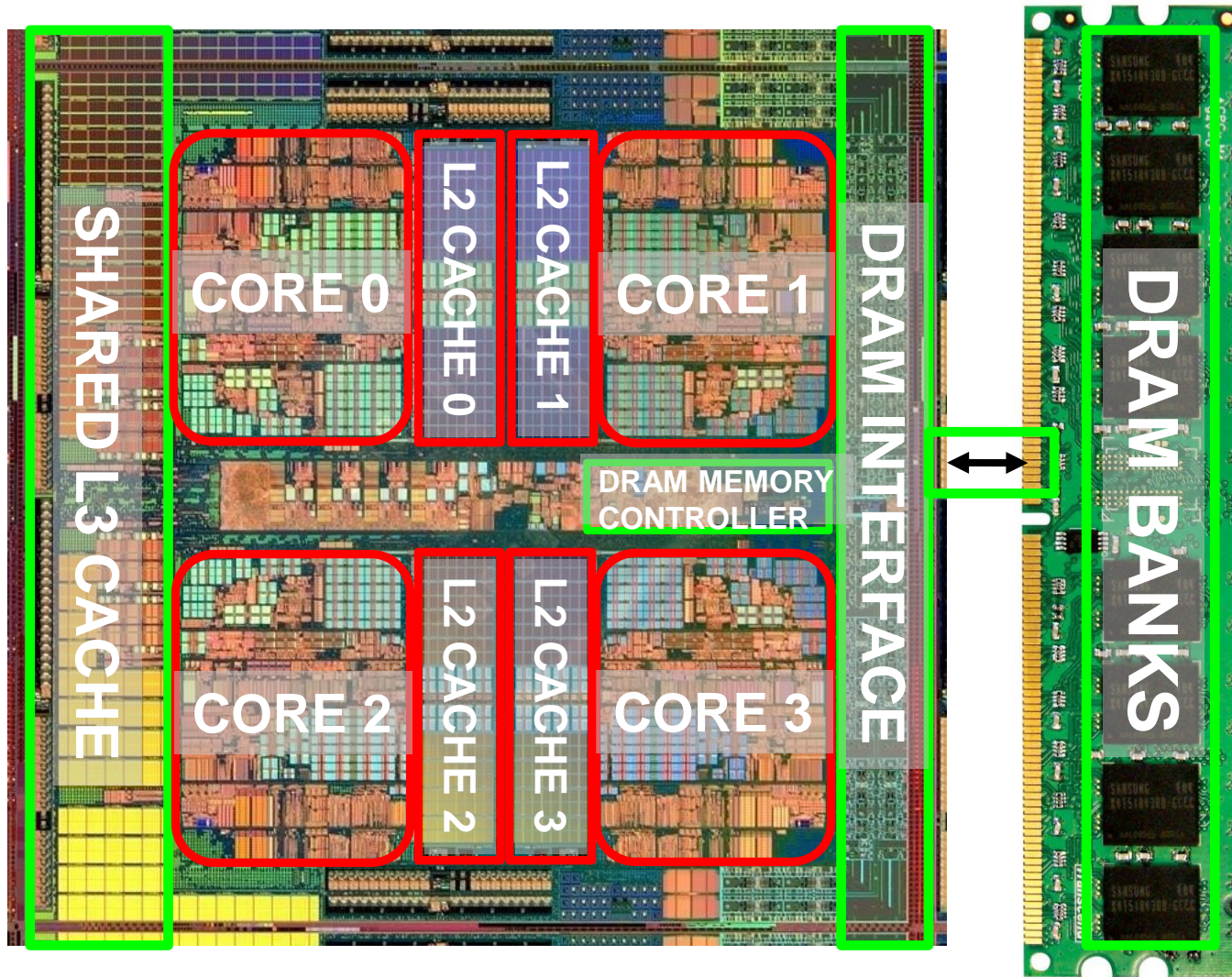
- Meltdown & Spectre (2017-2018)

- Rowhammer (2012-2014)

- Memories Forget: Refresh (2011-2012)

- Memory Performance Attacks (2006-2007)

# Mystery #3:  DRAM Refresh

# DRAM in the System

Multi-Core
Chip



SHARED L3 CACHE

CORE 0

L2 CACHE 0

L2 CACHE 1

CORE 1

DRAM MEMORY CONTROLLER

CORE 2

L2 CACHE 2

L2 CACHE 3

CORE 3

DRAM INTERFACE

DRAM BANKS

*Die photo credit: AMD Barcelona

# A DRAM Cell



wordline (row enable)

bitline    bitline    bitline    bitline
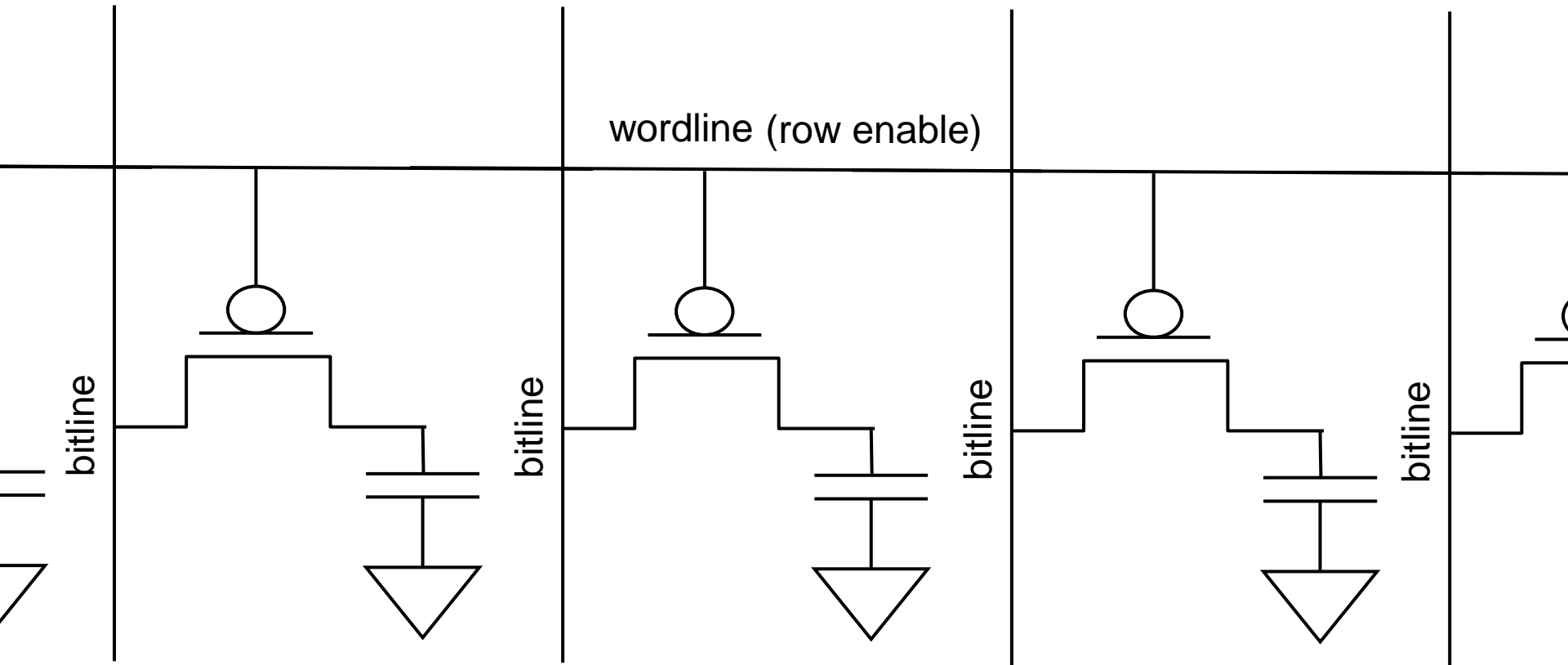
- A DRAM cell consists of a capacitor and an access transistor
- It stores data in terms of charge status of the capacitor
- A DRAM chip consists of (10s of 1000s of) rows of such cells

# DRAM Refresh

- DRAM capacitor charge leaks over time

- The memory controller needs to refresh each row periodically to restore charge
  - Activate each row every N ms
  - Typical N = 64 ms

- Downsides of refresh
  - -- Energy consumption: Each refresh consumes energy
  - -- Performance degradation: DRAM rank/bank unavailable while refreshed
  - -- QoS/predictability impact: (Long) pause times during refresh
  - -- Refresh rate limits DRAM capacity scaling

# First, Some Analysis

- Imagine a system with 1 ExaByte DRAM (2^60 bytes)
- Assume a row size of 8 KiloBytes (2^13 bytes)

- How many rows are there?
- How many refreshes happen in 64ms?
- What is the total power consumption of DRAM refresh?
- What is the total energy consumption of DRAM refresh during one day?

- A good exercise… Optional homework…
- Brownie points from me if you do it…

# Refresh Overhead: Performance



Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.
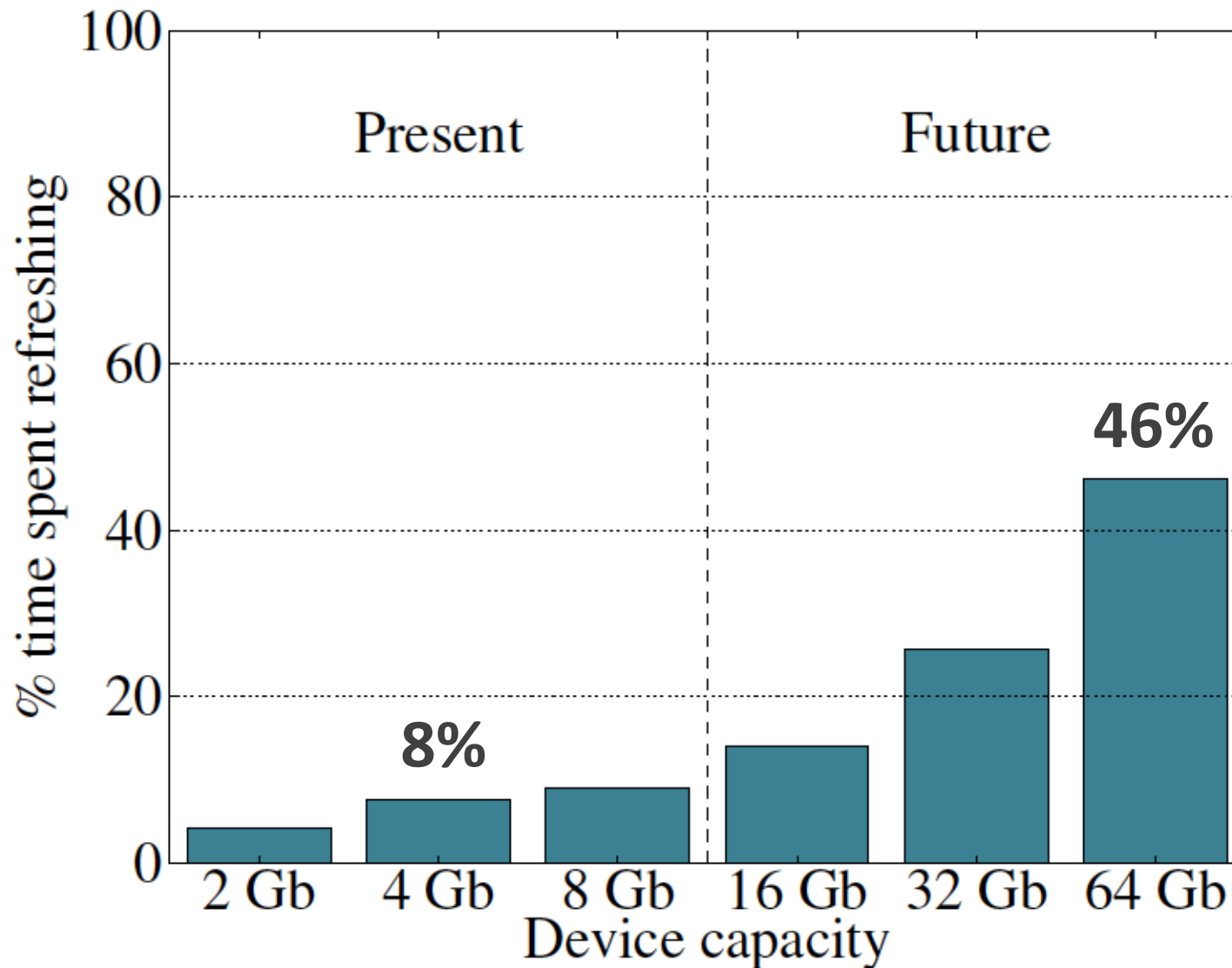
# Refresh Overhead: Energy



Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.
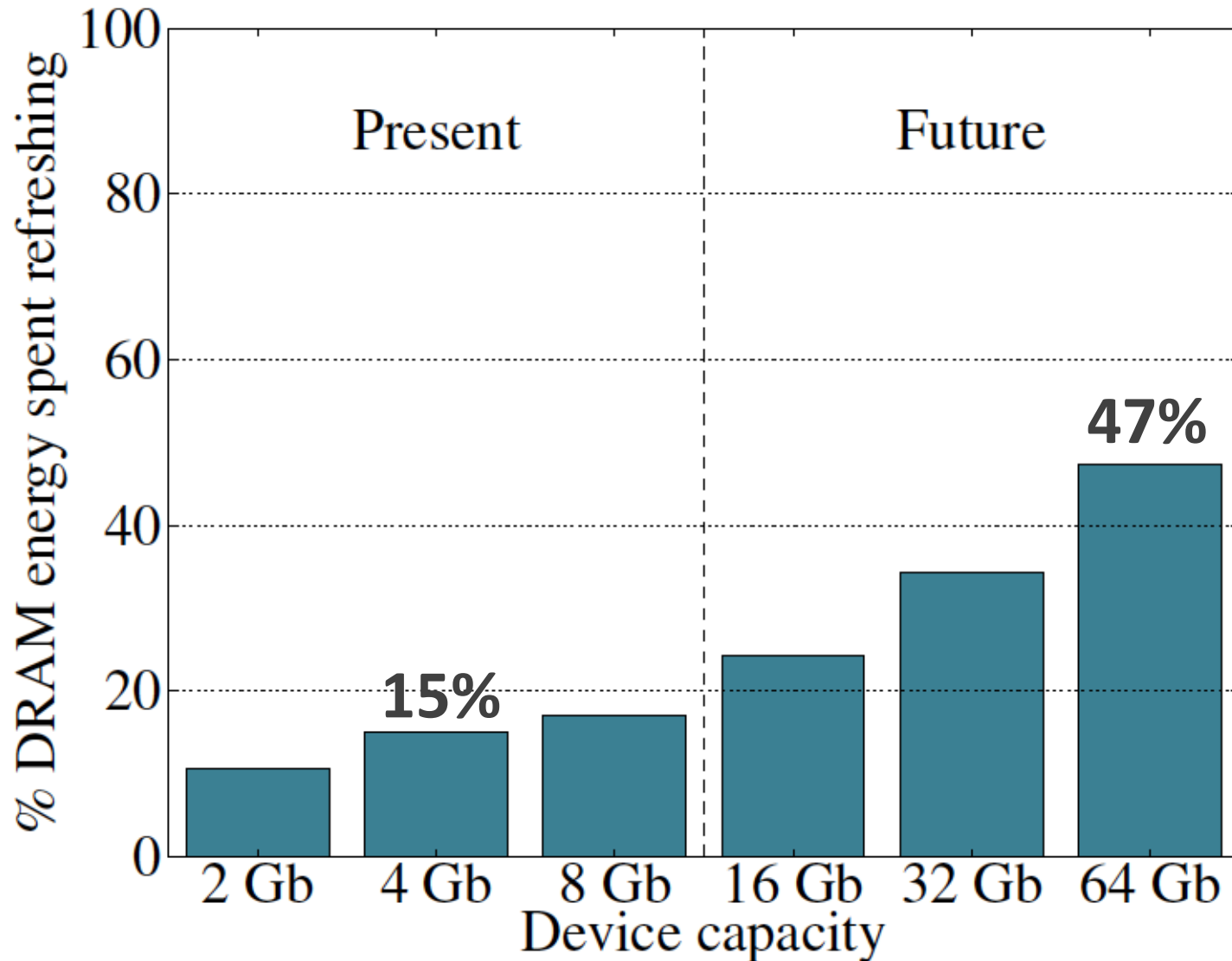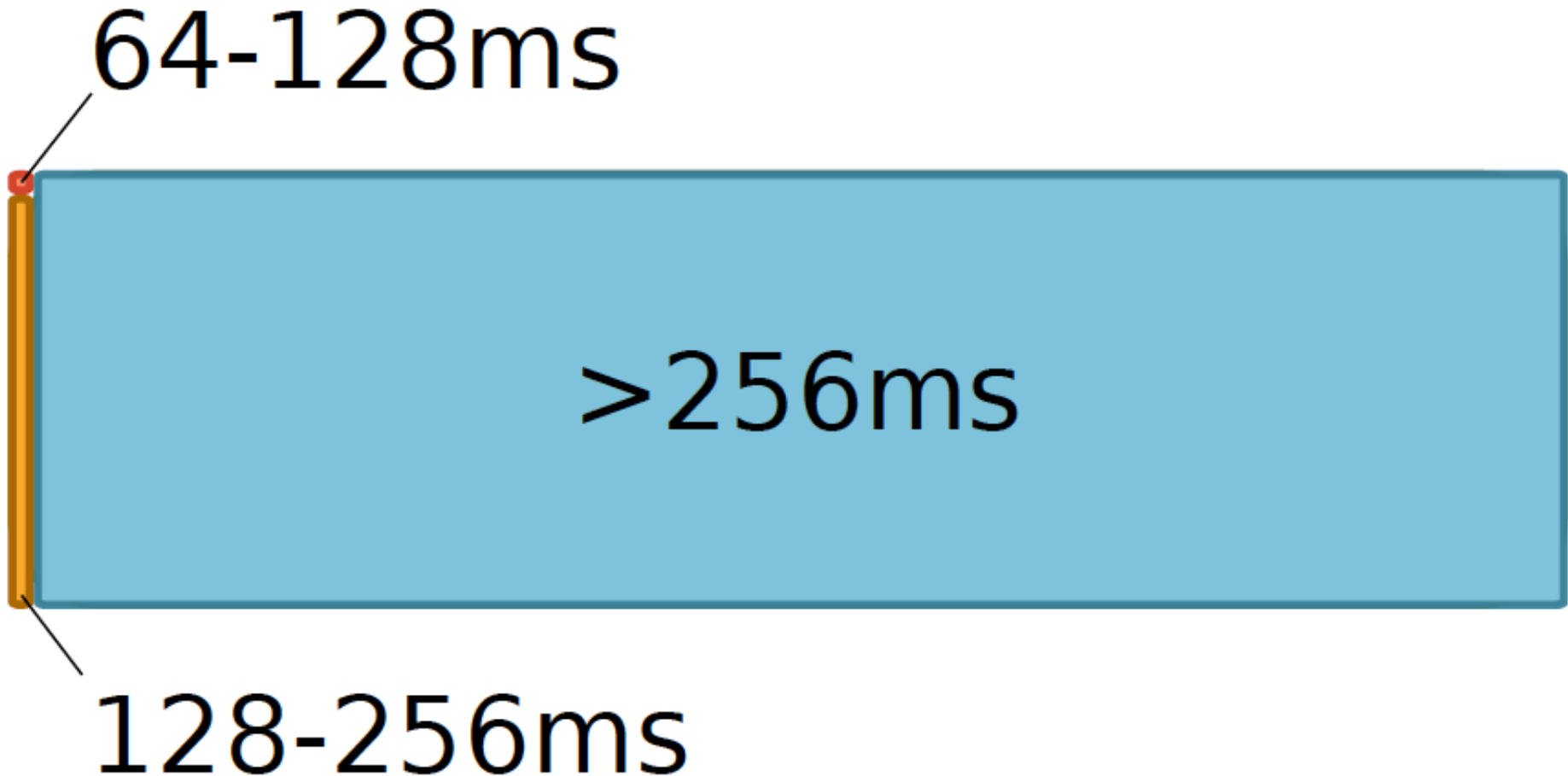
# How Do We Solve the Problem?

- Observation: All DRAM rows are refreshed every 64ms.

- Critical thinking: Do we need to refresh all rows every 64ms?

- What if we knew what happened underneath and exposed that information to upper layers?

# Underneath: Retention Time Profile of DRAM



64-128ms

\>256ms

128-256ms

Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

# Aside: Why Do We Have Such a Profile?

- Answer: Manufacturing is not perfect

- Not all DRAM cells are exactly the same

- Some are leakier than others

- This is called Manufacturing Process Variation

# Opportunity: Taking Advantage of This Profile

- Assume we know the retention time of each row exactly

- What can we do with this information?

- Who do we expose this information to?

- How much information do we expose?
  - Affects hardware/software overhead, power consumption, verification complexity, cost

- How do we determine this profile information?
  - Also, who determines it?

| Problem |
| Algorithm |
| Program/Language |
| Runtime System (VM, OS, MM) |
| ISA (Architecture) |
| Microarchitecture |
| Logic |
| Devices |
| Electrons |

# Retention Time of DRAM Rows

- Observation: Overwhelming majority of DRAM rows can be refreshed much less often without losing data
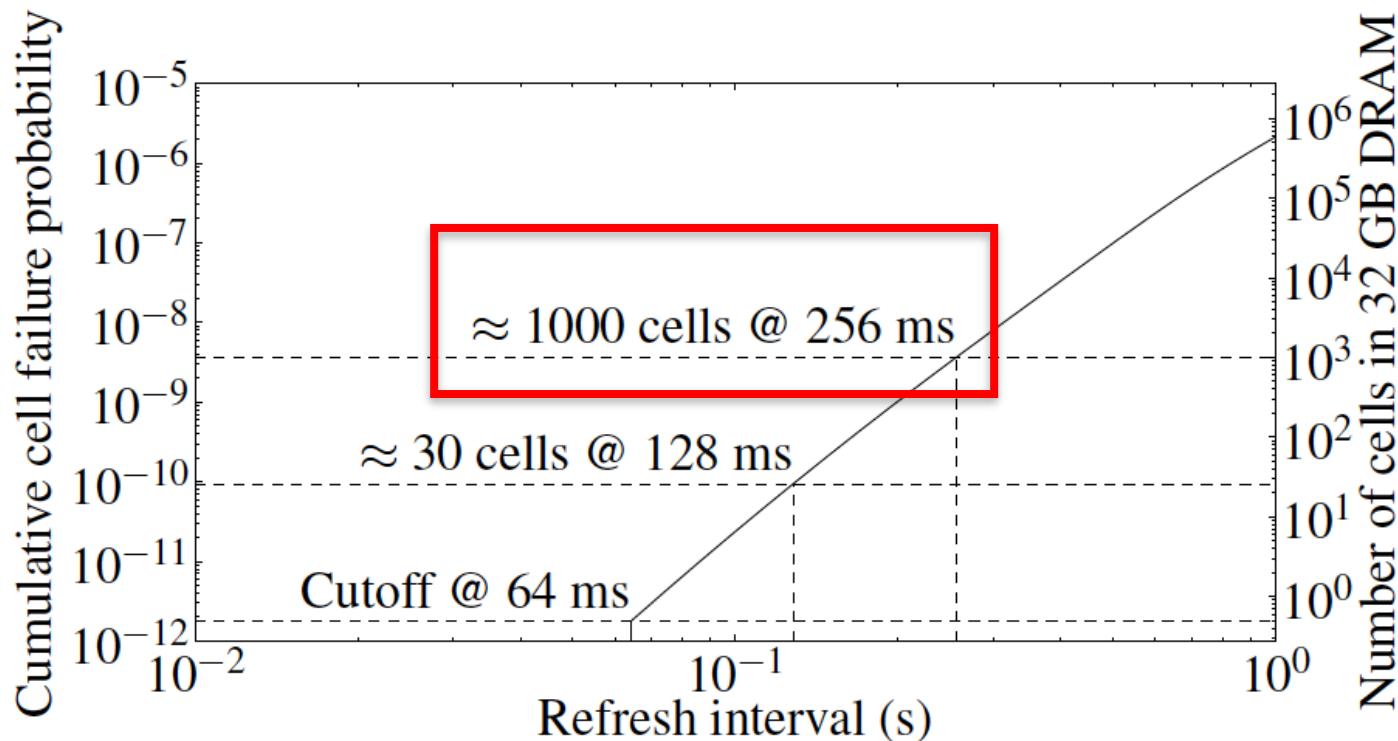


Key Idea of RAIDR: Refresh weak rows more frequently, all other rows less frequently

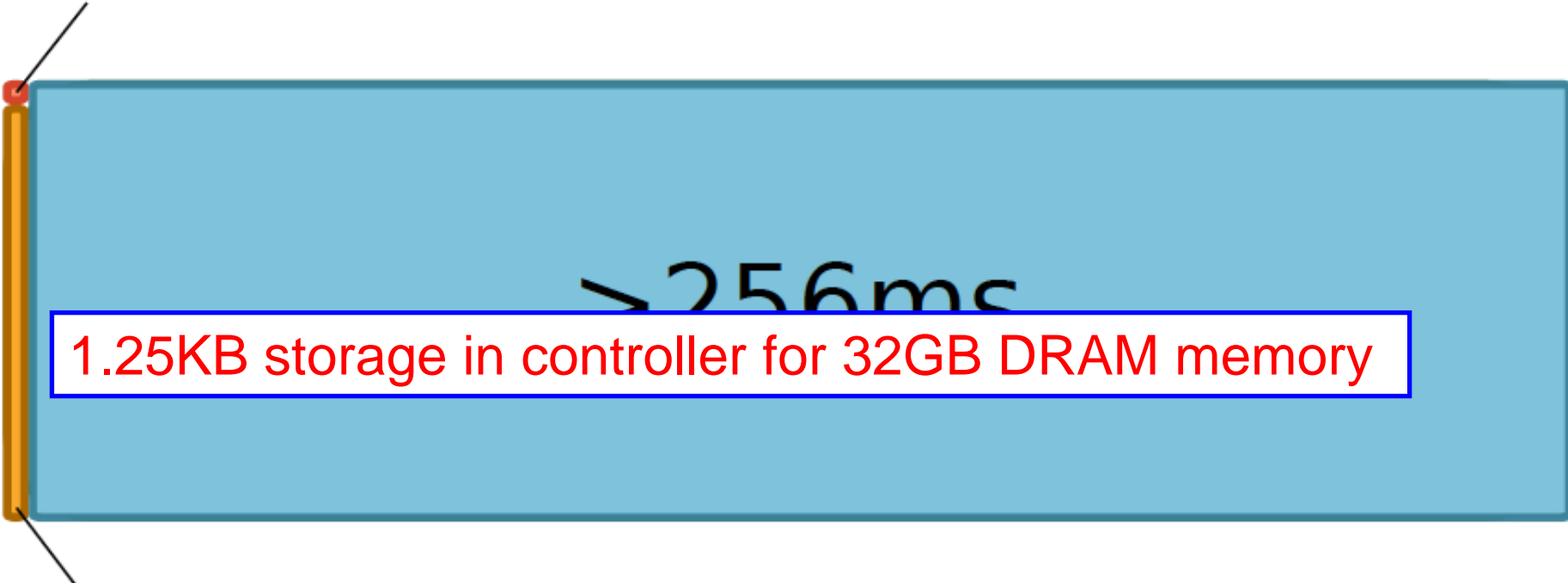Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

# RAIDR: Eliminating Unnecessary DRAM Refreshes

Liu, Jaiyen, Veras, Mutlu,
[RAIDR: Retention-Aware Intelligent DRAM Refresh](#)
ISCA 2012.

# RAIDR: Mechanism

**1. Profiling:** Identify the retention time of all DRAM rows

64-128ms

>256ms

1.25KB storage in controller for 32GB DRAM memory

128-256ms

→ check the bins to determine refresh rate of a row

Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.

# RAIDR: Results and Takeaways

- System: 32GB DRAM, 8-core; Various workloads

- RAIDR hardware cost: 1.25 kB (2 Bloom filters)
- Refresh reduction: 74.6%
- Dynamic DRAM energy reduction: 16%
- Idle DRAM power reduction: 20%
- Performance improvement: 9%

- Benefits increase as DRAM scales in density
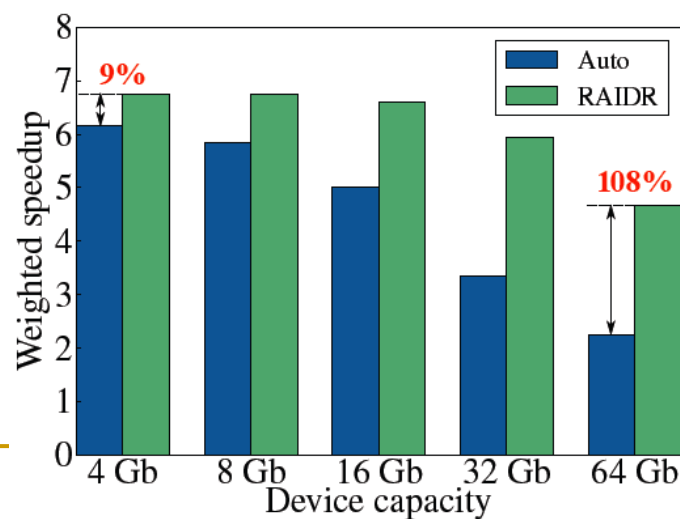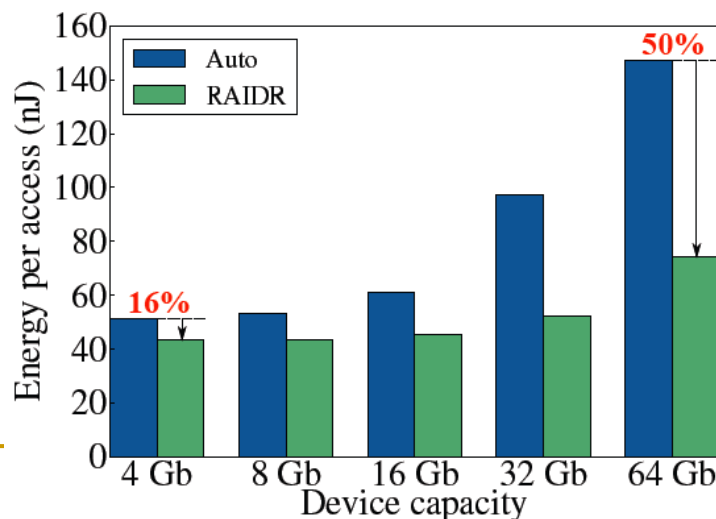
# Takeaway

Breaking the abstraction layers (between components and transformation hierarchy levels)

and knowing what is underneath

enables you to **understand** and **solve** problems

# Reading for the Really Interested

- Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu,
  **"RAIDR: Retention-Aware Intelligent DRAM Refresh"**
  *Proceedings of the 39th International Symposium on Computer Architecture*
  (**ISCA**), Portland, OR, June 2012. Slides (pdf)

# RAIDR: Retention-Aware Intelligent DRAM Refresh

Jamie Liu   Ben Jaiyen   Richard Veras   Onur Mutlu
Carnegie Mellon University
{jamiel,bjaiyen,rveras,onur}@cmu.edu

# Really Interested? … Further Readings

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

- Kevin Chang, Donghyuk Lee, Zeshan Chishti, Alaa Alameldeen, Chris Wilkerson, Yoongu Kim, and Onur Mutlu,
  **"Improving DRAM Performance by Parallelizing Refreshes with Accesses"**
  *Proceedings of the 20th International Symposium on High-Performance Computer Architecture* (**HPCA**), Orlando, FL, February 2014. Slides (pptx) (pdf)

# Detailed Lectures on Memory Refresh

- **Computer Architecture, Fall 2020, Lecture 2b**
  - Data Retention and Memory Refresh (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=v702wUnaWGE&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=3

- **Computer Architecture, Fall 2020, Lecture 3b**
  - Memory Systems: Challenges & Opportunities (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=Q2FbUxD7GHs&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=6

- **Computer Architecture, Fall 2020, Lecture 4a**
  - Memory Systems: Solution Directions (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=PANTCVTYe8M&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=7

**https://www.youtube.com/onurmutlulectures**

# Memory Refresh Lecture …

- Computer Architecture, Fall 2020, Lecture 2b
  - Data Retention and Memory Refresh (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=v702wUnaWGE&list=PL5Q2soXY2Zi9xidyIgB xUz7xRPS-wisBN&index=3

**SAFARI**

# Mystery #4:
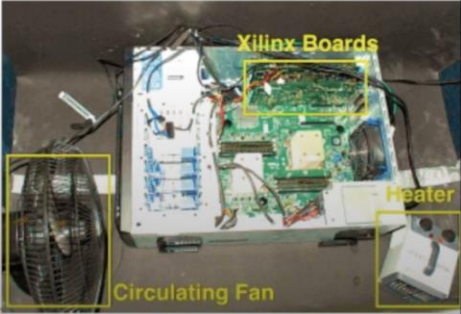# Memory Performance Attacks

# Multi-Core Systems

Multi-Core
Chip



SHARED L3 CACHE

CORE 0

L2 CACHE 0

L2 CACHE 1

CORE 1

DRAM MEMORY CONTROLLER

CORE 2

L2 CACHE 2

L2 CACHE 3

CORE 3

DRAM INTERFACE

DRAM BANKS
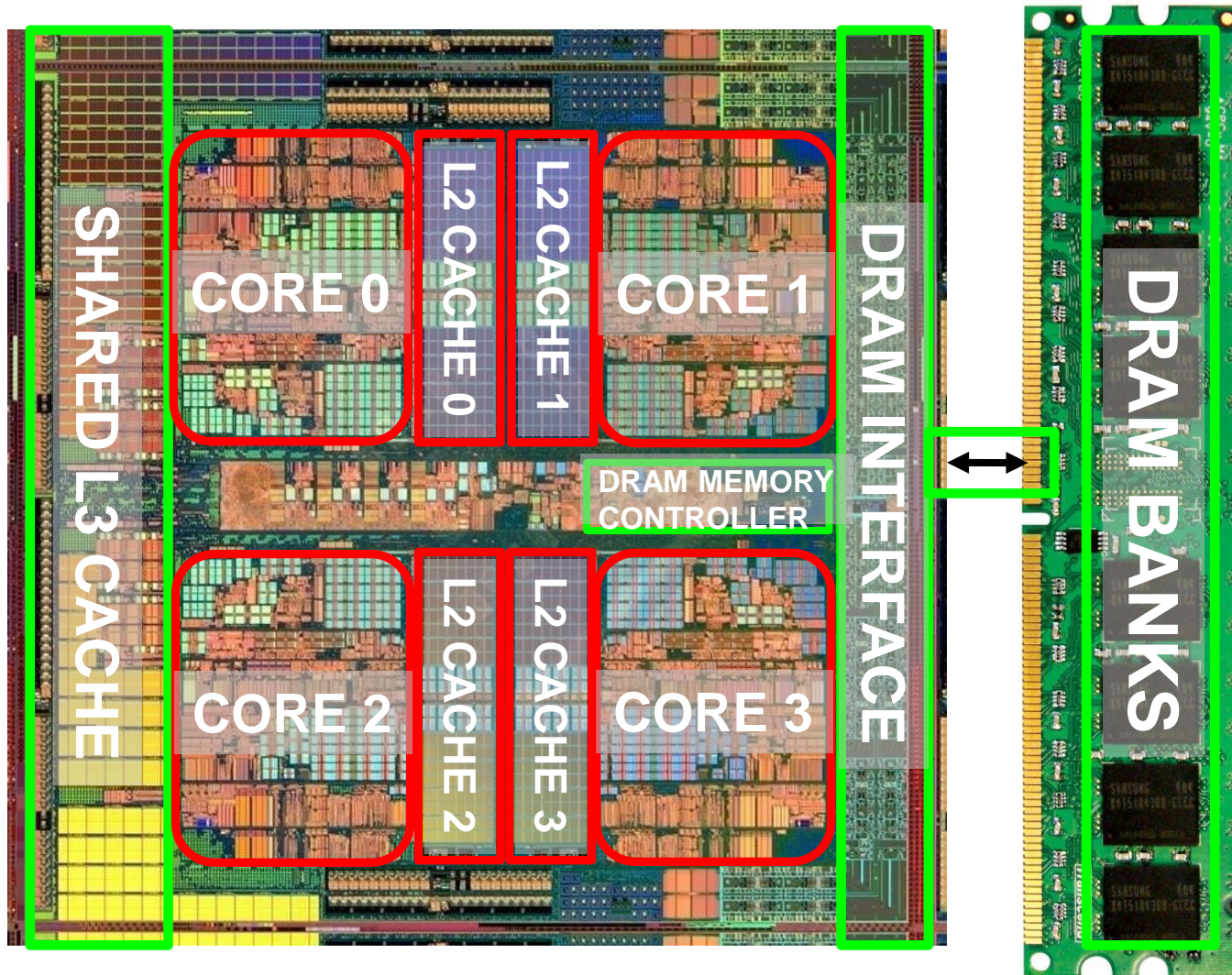
*Die photo credit: AMD Barcelona

# A Trend: Many Cores on Chip

- Simpler and lower power than a single large core
- Parallel processing on single chip → faster, new applications
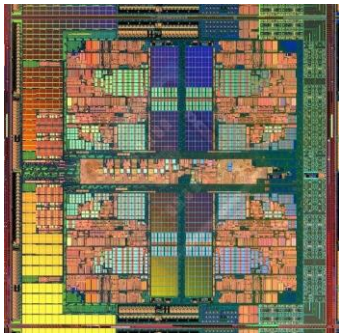
AMD Barcelona
4 cores
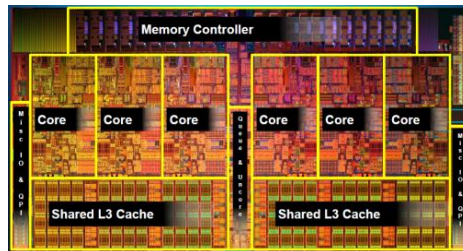
Intel Core i7
8 cores
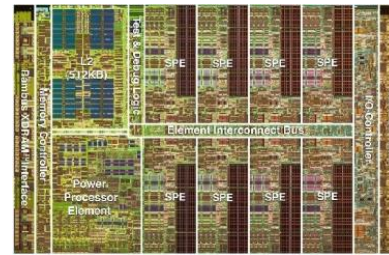
IBM Cell BE
8+1 cores
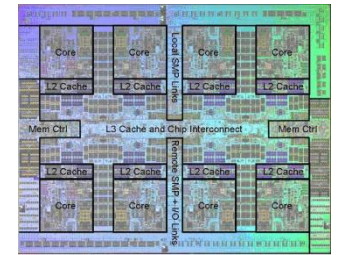
IBM POWER7
8 cores
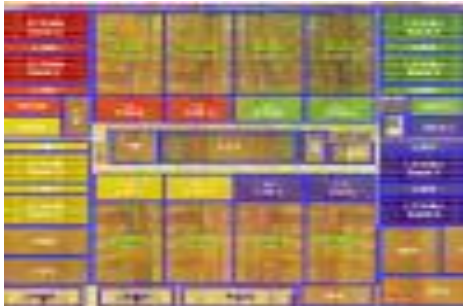
Sun Niagara II
8 cores
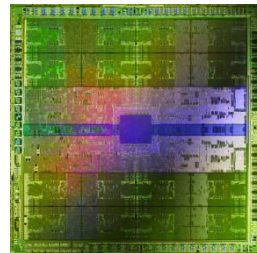
Nvidia Fermi
448 "cores"
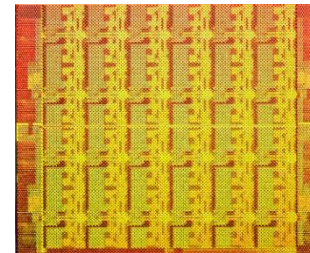
Intel SCC
48 cores, networked
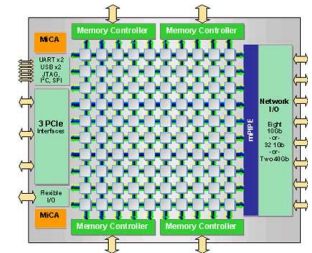
Tilera TILE Gx
100 cores, networked

# Many Cores on Chip

- What we want:
  - N times the system performance with N times the cores

- What do we get today?

# Unexpected Slowdowns in Multi-Core
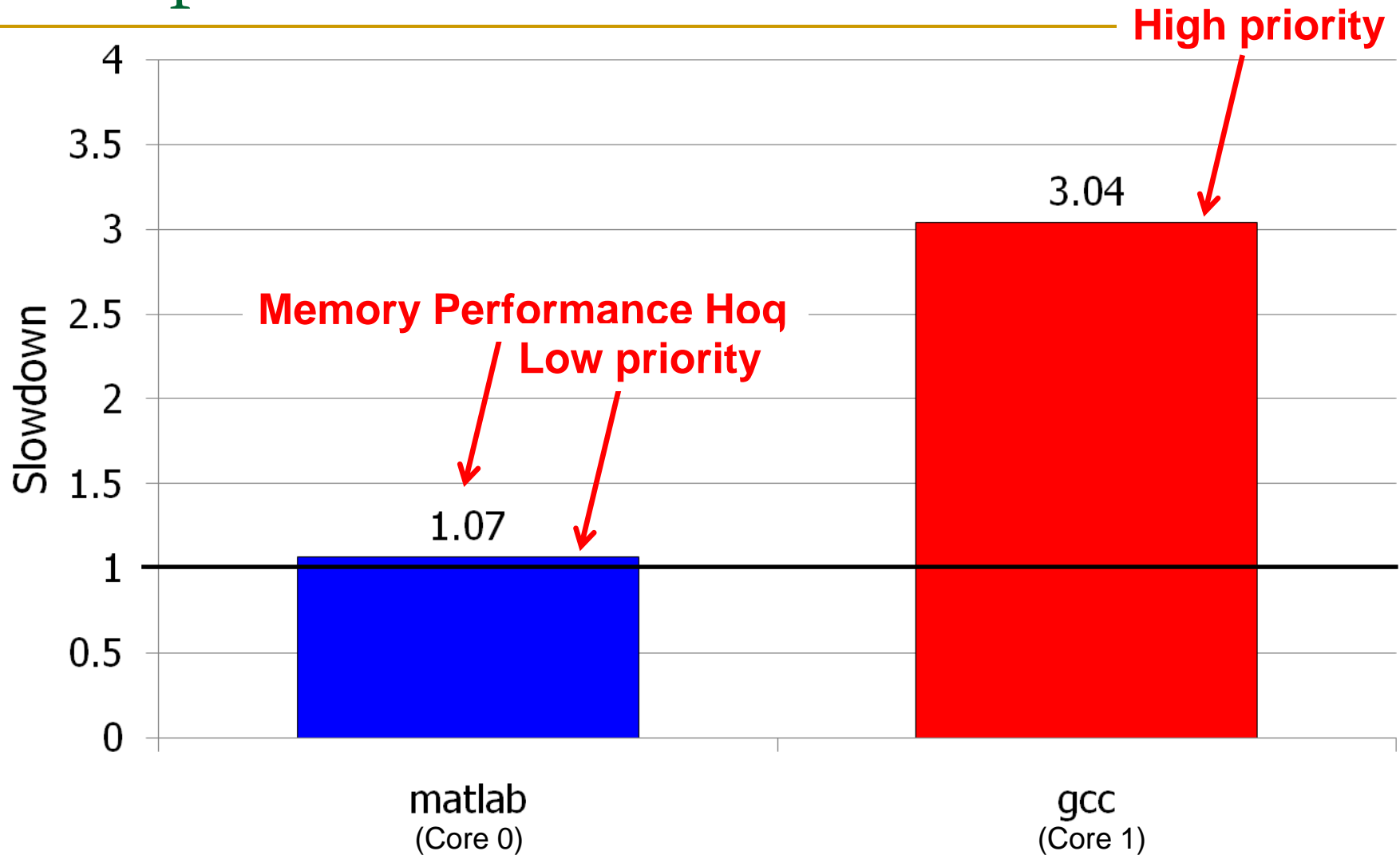


Moscibroda and Mutlu, "Memory performance attacks: Denial of memory service in multi-core systems," USENIX Security 2007.

# Three Questions

- Can you figure out why the applications slow down if you do not know the underlying system and how it works?

- Can you figure out why there is a disparity in slowdowns if you do not know how the system executes the programs?

- Can you fix the problem without knowing what is happening "underneath"?

# Three Questions

- Why is there any slowdown?

- Why is there a disparity in slowdowns?

- How can we solve the problem if we do not want that disparity?
  - What do we want (the system to provide)?

# Why Is This Important?

- We want to execute applications in parallel in multi-core systems → consolidate more and more
  - Cloud computing
  - Mobile phones

- We want to mix different types of applications together
  - those requiring QoS guarantees (e.g., video, pedestrian detection)
  - those that are important but less so
  - those that are less important

- We want the system to be controllable **and** high performance

# Why the Disparity in Slowdowns?



Multi-Core Chip

Shared DRAM Memory System

CORE 1    CORE 2

L2 CACHE    L2 CACHE

INTERCONNECT

DRAM MEMORY CONTROLLER

DRAM Bank 0    DRAM Bank 1    DRAM Bank 2    DRAM Bank 3

# Why the Disparity in Slowdowns?



Multi-Core Chip

unfairness

Shared DRAM Memory System

matlab  gcc

L2 CACHE    L2 CACHE

INTERCONNECT

DRAM MEMORY CONTROLLER
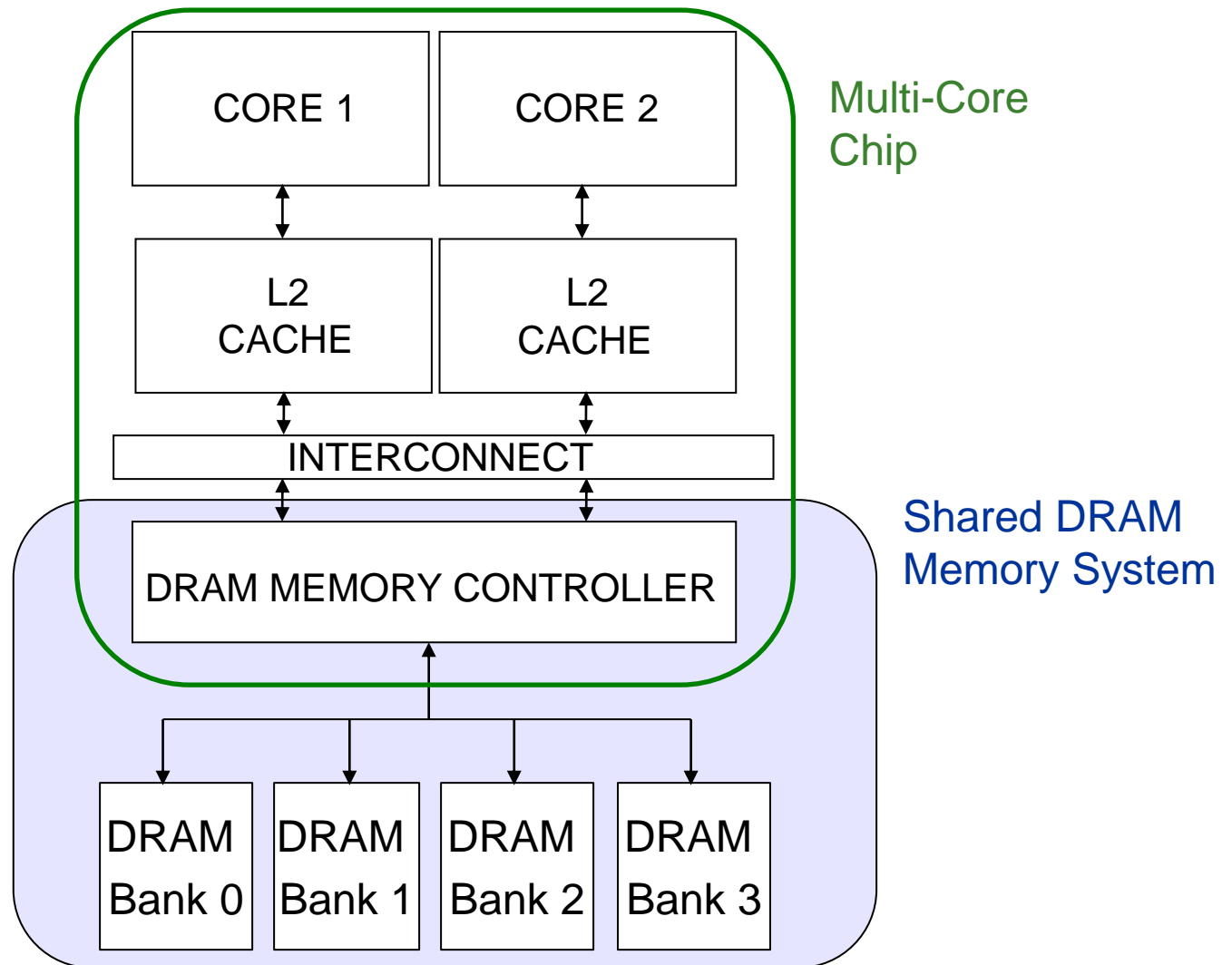
DRAM Bank 0   DRAM Bank 1   DRAM Bank 2   DRAM Bank 3
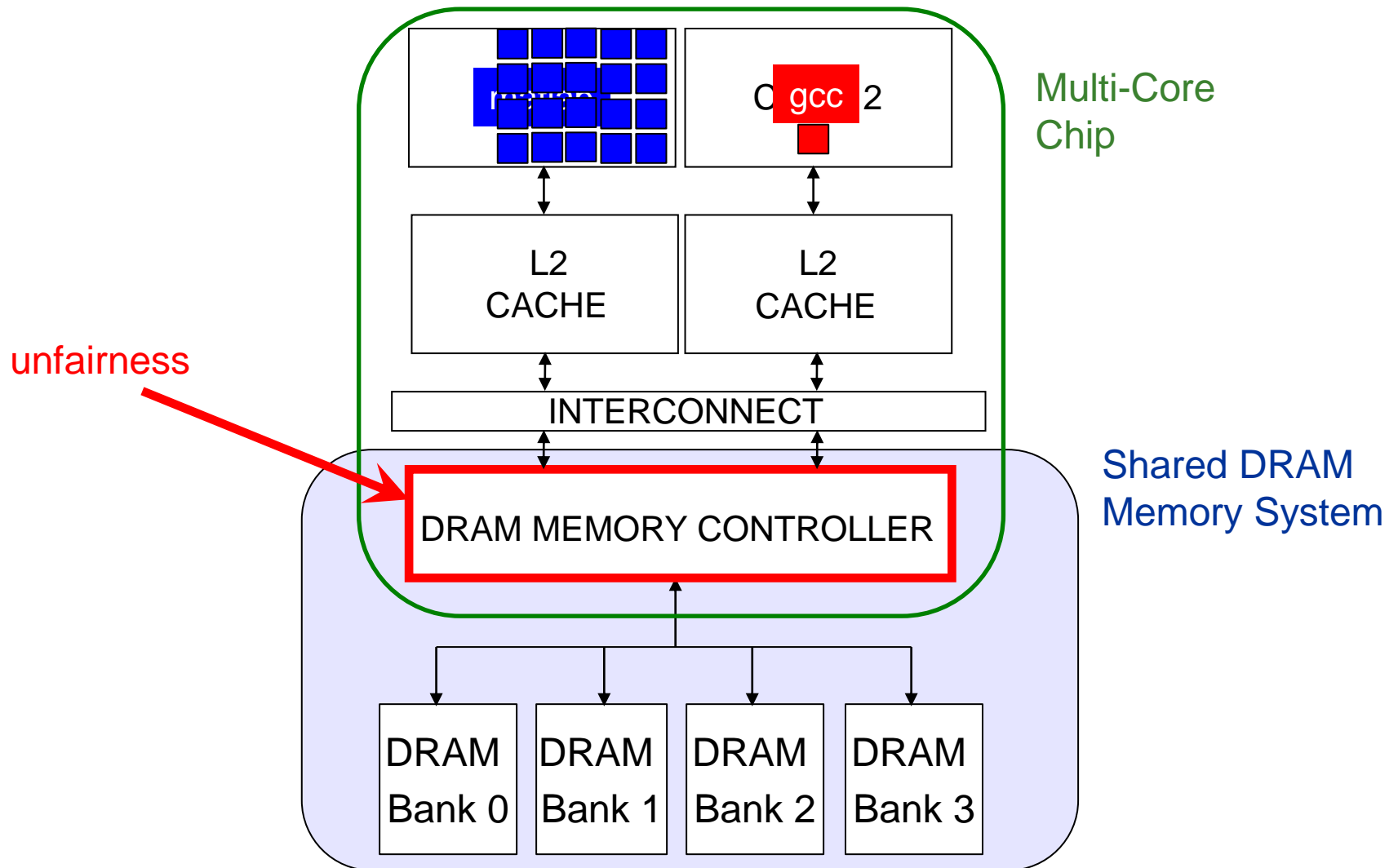
# Digging Deeper: DRAM Bank Operation

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

Columns

Row address 1

Row decoder
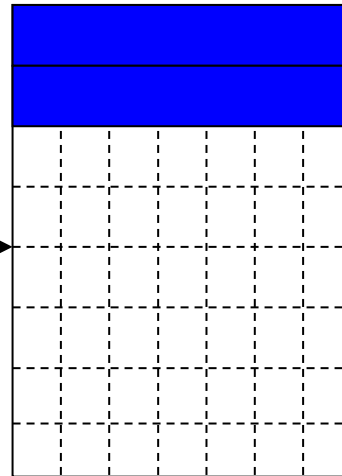
Rows

This view of a bank is an abstraction.

Internally, a bank consists of many cells (transistors & capacitors) and other structures that enable access to cells

Row 1    Row Buffer   CONFLICT !

Column address 0

Column mux
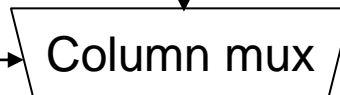
Data

# DRAM Controllers

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

- Current controllers take advantage of this fact

- 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

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

# 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 attacks
  - Can write programs to exploit unfairness

# A Memory Performance Hog

```
// initialize large arrays A, B

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

```
// initialize large arrays A, B

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

**STREAM**

**RANDOM**

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

- 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?

T0: Row 0

T0: Row 5

T0: Row 111

T0: Row 16

Memory Request Buffer

Row decoder

Row Buffer
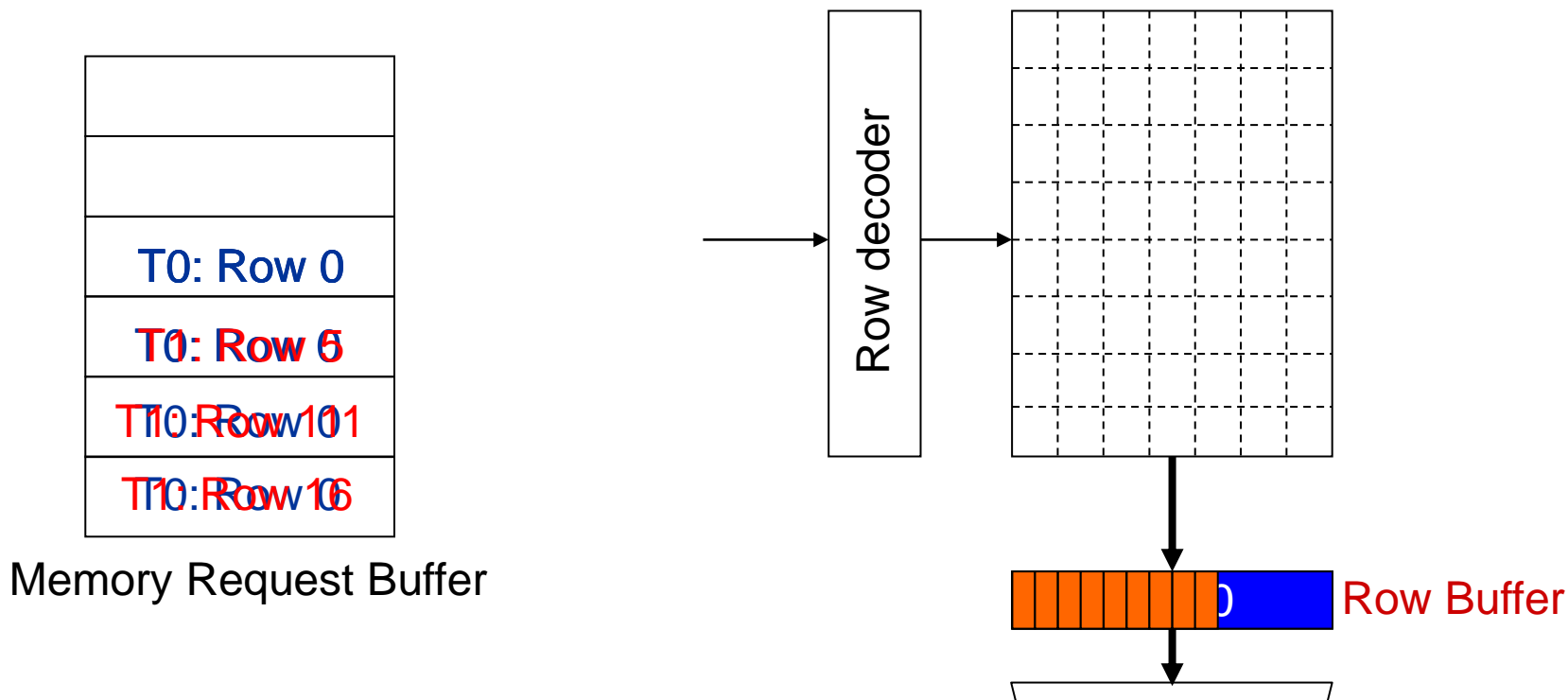
Row size: 8KB, request size: 64B

128 (8KB/64B) requests of STREAM serviced
before a single request of RANDOM

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

# Now That We Know What Happens Underneath

- How would you solve the problem?

- What is the right place to solve the problem?
  - Programmer?
  - System software?
  - Compiler?
  - Hardware (Memory controller)?
  - Hardware (DRAM)?
  - Circuits?

- Two major goals of this course:
  - Enable you to think critically
  - Enable you to think broadly

| |
|---|
| Problem |
| Algorithm |
| Program/Language |
| Runtime System (VM, OS, MM) |
| ISA (Architecture) |
| Microarchitecture |
| Logic |
| Devices |
| Electrons |
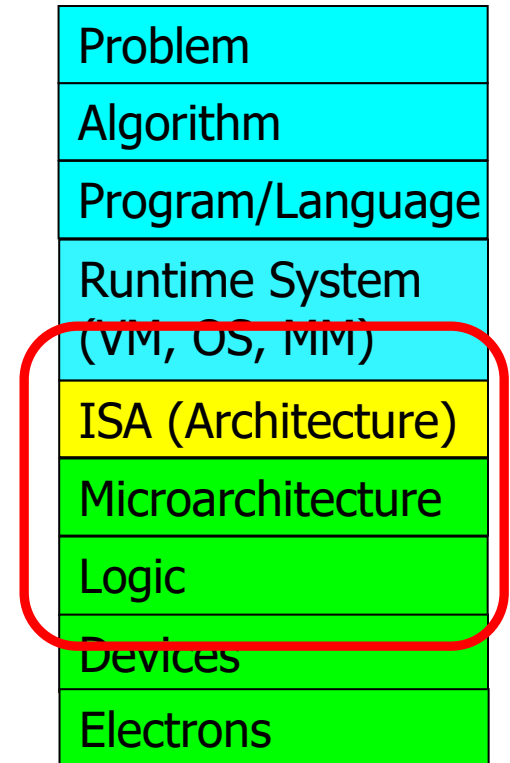
# For the Really Interested…

- 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

# Really Interested? … Further Readings

- 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. Slides (ppt)

- 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**) [Slides (ppt)]

- Sai Prashanth Muralidhara, Lavanya Subramanian, Onur Mutlu, Mahmut Kandemir, and Thomas Moscibroda,
  **"Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning"**
  *Proceedings of the 44th International Symposium on Microarchitecture* (**MICRO**), Porto Alegre, Brazil, December 2011. Slides (pptx)

# Detailed Lectures on Memory Controllers

- **Computer Architecture, Fall 2020, Lecture 2a**
  - Memory Performance Attacks (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=VJzZbwgBfy8&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=2

- **Computer Architecture, Fall 2020, Lecture 11a**
  - Memory Controllers (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=TeG773OgiMQ&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=20

- **Computer Architecture, Fall 2020, Lecture 11b**
  - Memory Interference and QoS (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=0nnI807nCkc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=21

- **Computer Architecture, Fall 2020, Lecture 13**
  - Memory Interference and QoS II (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=Axye9VqQT7w&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=26

**https://www.youtube.com/onurmutlulectures**

# Memory Performance Attacks Lecture ...

- Computer Architecture, Fall 2020, Lecture 2a
    - Memory Performance Attacks (ETH Zürich, Fall 2020)
    - https://www.youtube.com/watch?v=VJzZbwgBfy8&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=2

**SAFARI**

# Takeaway I

Breaking the abstraction layers (between components and transformation hierarchy levels)

and knowing what is underneath

enables you to **understand** and **solve** problems

# Takeaway II

Cooperation between multiple components and layers can enable more effective solutions and systems

# Recap: Mysteries No Longer!

- **Meltdown & Spectre (2017-2018)**

- **Rowhammer (2012-2014)**

- **Memories Forget: Refresh (2011-2012)**

- **Memory Performance Attacks (2006-2007)**

# Takeaways

# Takeaways

- It is an exciting time to be understanding and designing computing architectures

- Many challenging and exciting problems in platform design
  - That no one has tackled (or thought about) before
  - That can have huge impact on the world's future

- Driven by huge hunger for data (Big Data), new applications (ML/AI, graph analytics, genomics), ever-greater realism, …
  - We can easily collect more data than we can analyze/understand

- Driven by significant difficulties in keeping up with that hunger at the technology layer
  - Five walls: Energy, reliability, complexity, security, scalability

# Computer Architecture as an Enabler of the Future

# Assignment: **Required** Lecture Video

- Why study computer architecture? Why is it important?
- Future Computing Platforms: Challenges & Opportunities

- **Required Assignment**
  - **Watch one of** Prof. Mutlu's lectures and analyze either (or both)
  - https://www.youtube.com/watch?v=kgiZlSOcGFM (May 2017)
  - https://www.youtube.com/watch?v=mskTeNnf-i0 (Feb 2021)

- **Optional Assignment – for 1% extra credit**
  - **Write a 1-page summary** of one of the lectures and email us
    - What are your key takeaways?
    - What did you learn?
    - What did you like or dislike?
    - Submit your summary to Moodle

# Digital Design & Computer Arch.

# Lecture 3a: Mysteries in Comp. Arch.

Prof. Onur Mutlu

ETH Zürich

Spring 2021

4 March 2021

# Backup Slides For Your Benefit. Not Covered in Lecture.

# Bloom Filters

# Approximate Set Membership

- Suppose you want to quickly find out:
  - whether an element belongs to a set

- And, you can tolerate mistakes of the sort:
  - The element is actually **not** in the set, but you are incorrectly told that it is → false positive

- But, you cannot tolerate mistakes of the sort:
  - The element is actually in the set, but you are incorrectly told that it is **not** → false negative

- Example task: You want to quickly identify all Mobile Phone Model X owners among all possible people in the world
  - Perhaps you want to give them free replacement phones

# Example Task

- World population
    - ~8 billion (and growing)
    - 1 bit per person to indicate Model X owner or not
    - 2^33 bits needed to represent the entire set accurately
        - 8 Gigabits → large storage cost, slow access

- Mobile Phone Model X owner population
    - Say 1 million (and growing)

- Can we represent the Model X owner set approximately, using a much smaller number of bits?
    - Record the ID's of owners in a much smaller Bloom Filter

# Example Task II

- DRAM row population
  - ~8 billion (and growing)
  - 1 bit per row to indicate Refresh-often or not
  - 2^33 bits needed to represent the entire set accurately
    - 8 Gigabits → large storage cost, slow access

- Refresh-often population
  - Say 1 million

- Can we represent Refresh-often set approximately, using a much smaller number of bits?
  - Record the ID's of Refresh-Often rows in a much smaller Bloom Filter

# Bloom Filter

- [Bloom, CACM 1970]

- Probabilistic data structure that compactly represents set membership (presence or absence of element in a set)

- Non-approximate set membership: Use 1 bit per element to indicate absence/presence of each element from an element space of N elements

- Approximate set membership: use a much smaller number of bits and indicate each element's presence/absence with a subset of those bits

  - Some elements map to the bits other elements also map to

- Operations: 1) insert, 2) test, 3) remove all elements

Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", CACM 1970.

# Bloom Filter Operation Example

Example with 64–128ms bin:

| 0 | 0 | **1** | 0 | **1** | 0 | 0 | 0 | 0 | **1** | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Hash function 1    Hash function 2    Hash function 3

Insert Row 1

Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", CACM 1970.
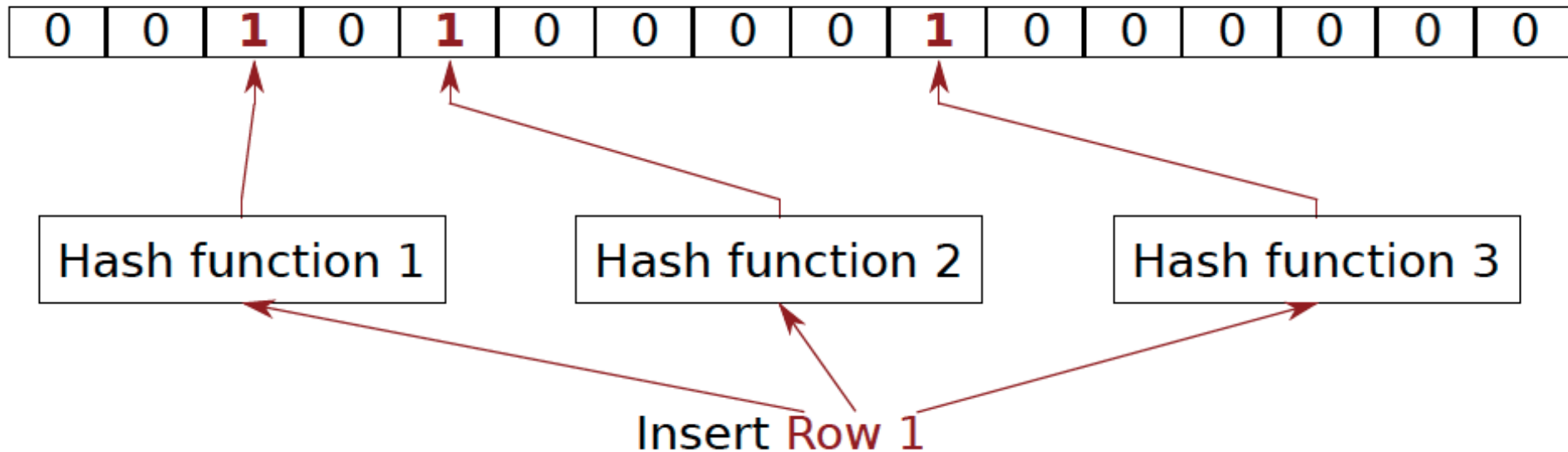
# Bloom Filter Operation Example

Example with 64–128ms bin:

# Bloom Filter Operation Example

Example with 64–128ms bin:

| | 0 | & | | 1 | & | | | 0 | =0 | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Hash function 1     Hash function 2     Hash function 3

Row 2 present?
No

# Bloom Filter Operation Example

Example with 64–128ms bin:

| 0 | 0 | 1 | 0 | 1 | **1** | 0 | 0 | 0 | 1 | 0 | 0 | **1** | 0 | **1** | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Hash function 1    Hash function 2    Hash function 3

Insert Row 4

# Bloom Filter Operation Example

Example with 64–128ms bin:

| | | | | | | | | | 1 | | | & | 1 | & | 1 | =1 |

| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |

Hash function 1   Hash function 2   Hash function 3

Row 5 present?
Yes (false positive)
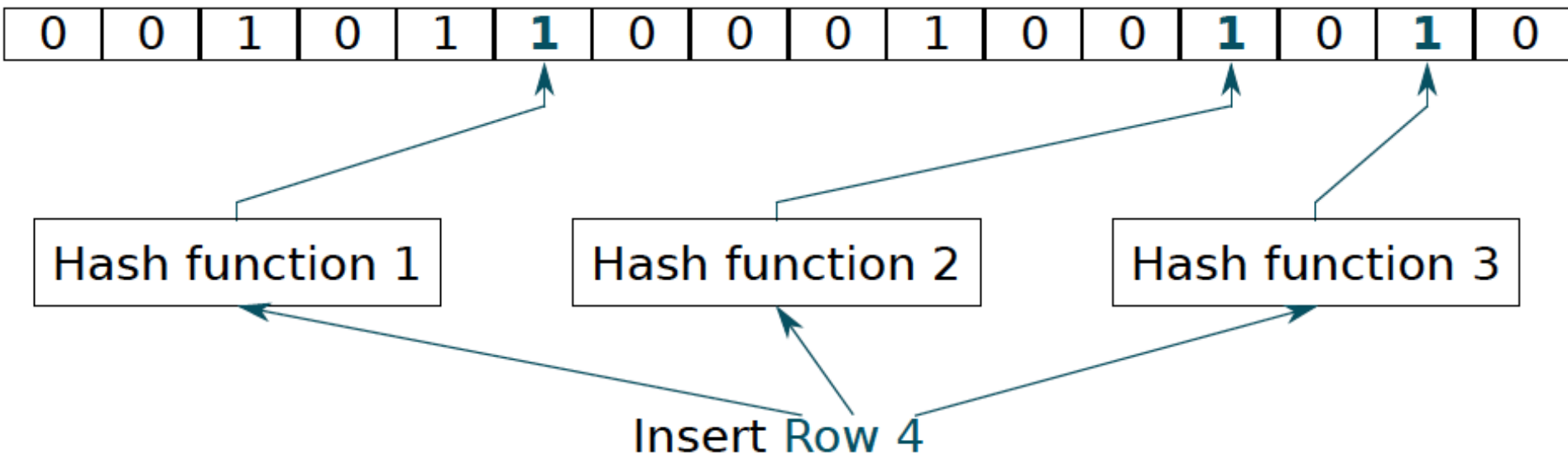
# Bloom Filters

## Space/Time Trade-offs in Hash Coding with Allowable Errors

BURTON H. BLOOM
*Computer Usage Company, Newton Upper Falls, Mass.*

In such applications, it is envisaged that overall performance could be improved by using a smaller core resident hash area in conjunction with the new methods and, when necessary, by using some secondary and perhaps time-consuming test to "catch" the small fraction of errors associated with the new methods. An example is discussed which illustrates possible areas of application for the new methods.

In this paper trade-offs among certain computational factors in hash coding are analyzed. The paradigm problem considered is that of testing a series of messages one-by-one for membership in a given set of messages. Two new hash-coding methods are examined and compared with a particular conventional hash-coding method. The computational factors considered are the size of the hash area (space), the time required to identify a message as a nonmember of the given set (reject time), and an allowable error frequency.

Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", CACM 1970.

# Bloom Filters: Pros and Cons

- Advantages

  + Enables storage-efficient representation of set membership

  + Insertion and testing for set membership (presence) are fast

  + No false negatives: If Bloom Filter says an element is not present in the set, the element must not have been inserted

  + Enables tradeoffs between time & storage efficiency & false positive rate (via sizing and hashing)

- Disadvantages

  -- False positives: An element may be deemed to be present in the set by the Bloom Filter but it may never have been inserted

  Not the right data structure when you cannot tolerate false positives

Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", CACM 1970.

# Benefits of Bloom Filters as Refresh Rate Bins

- **False positives:** a row may be declared present in the Bloom filter even if it was never inserted
  - **Not a problem:** Refresh some rows more frequently than needed

- **No false negatives:** rows are never refreshed less frequently than needed (no correctness problems)

- **Scalable:** a Bloom filter never overflows (unlike a fixed-size table)

- **Efficient:** No need to store info on a per-row basis; simple hardware → 1.25 KB for 2 filters for 32 GB DRAM system

# Use of Bloom Filters in Hardware

- Useful & cost-effective when you can tolerate false positives in set membership tests

- See the following recent examples for clear descriptions of how Bloom Filters are used
  - Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh," ISCA 2012.
  - Seshadri et al., "The Evicted-Address Filter: A Unified Mechanism to Address Both Cache Pollution and Thrashing," PACT 2012.
  - Yaglikci et al., "BlockHammer: Preventing RowHammer at Low Cost by Blacklisting Rapidly-Accessed DRAM Rows," HPCA 2021.

# Bloom Filters for Eliminating Refresh

- Jamie Liu, Ben Jaiyen, Richard Veras, and Onur Mutlu,
  **"RAIDR: Retention-Aware Intelligent DRAM Refresh"**
  *Proceedings of the 39th International Symposium on Computer Architecture*
  (**ISCA**), Portland, OR, June 2012. Slides (pdf)

# RAIDR: Retention-Aware Intelligent DRAM Refresh

Jamie Liu    Ben Jaiyen    Richard Veras    Onur Mutlu
Carnegie Mellon University
{jamiel,bjaiyen,rveras,onur}@cmu.edu

# Bloom Filters for Better Cache Performance

- Vivek Seshadri, Onur Mutlu, Michael A. Kozuch, and Todd C. Mowry,
**"The Evicted-Address Filter: A Unified Mechanism to Address Both Cache Pollution and Thrashing"**
*Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques* (**PACT**), Minneapolis, MN, September 2012. Slides (pptx) Source Code

## The Evicted-Address Filter: A Unified Mechanism to Address Both Cache Pollution and Thrashing

Vivek Seshadri[†]
vseshadr@cs.cmu.edu

Onur Mutlu[†]
onur@cmu.edu

Michael A Kozuch[*]
michael.a.kozuch@intel.com

Todd C Mowry[†]
tcm@cs.cmu.edu

[†]Carnegie Mellon University      [*]Intel Labs Pittsburgh

# Bloom Filters for Solving RowHammer

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

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

A. Giray Yağlıkçı[1]    Minesh Patel[1]    Jeremie S. Kim[1]    Roknoddin Azizi[1]    Ataberk Olgun[1]    Lois Orosa[1]

Hasan Hassan[1]    Jisung Park[1]    Konstantinos Kanellopoulos[1]    Taha Shahroodi[1]    Saugata Ghose[2]    Onur Mutlu[1]

[1]*ETH Zürich*        [2]*University of Illinois at Urbana–Champaign*