Improving DRAM Performance by Parallelizing Refreshes with Accesses

Kevin Chang

Donghyuk Lee, Zeshan Chishti, Alaa Alameldeen,
Chris Wilkerson, Yoongu Kim, Onur Mutlu

SAFARI Carnegie Mellon intel

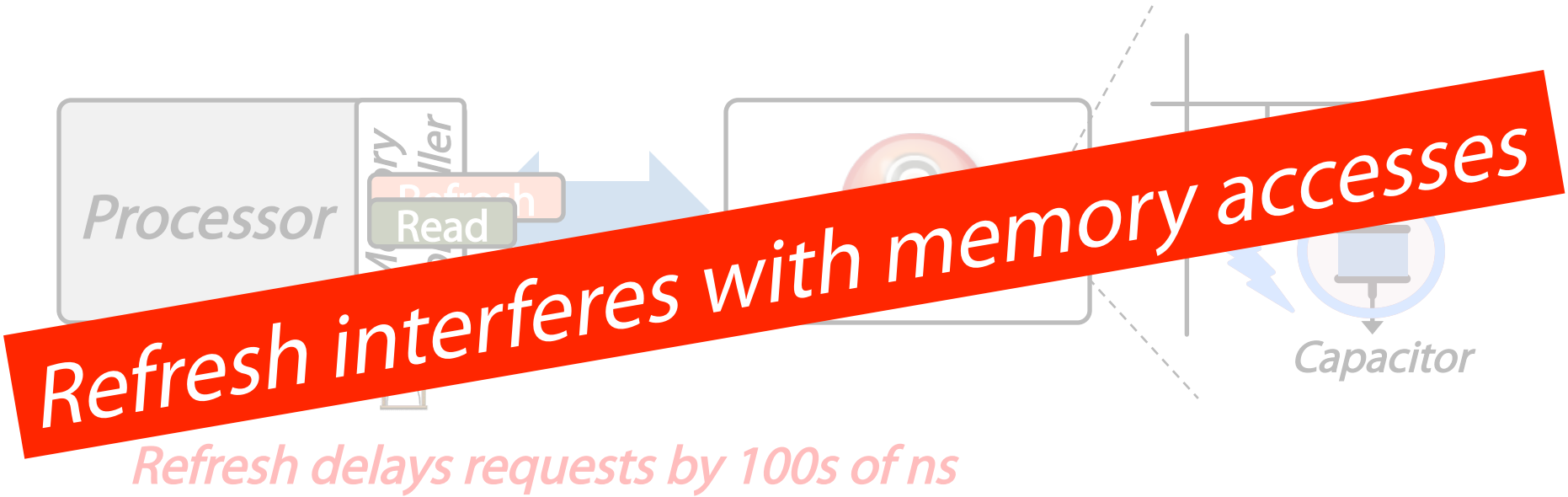
Executive Summary

- DRAM **refresh interferes with memory accesses**
  - Degrades system performance and energy efficiency
  - Becomes exacerbated as DRAM density increases
- Goal: Serve memory accesses in parallel with refreshes to reduce refresh interference on demand requests
- Our mechanisms:
  - 1. Enable more parallelization between refreshes and accesses across different banks with new per-bank refresh scheduling algorithms
  - 2. Enable serving accesses concurrently with refreshes in the same bank by exploiting DRAM subarrays
- Improve system performance and energy efficiency for a wide variety of different workloads and DRAM densities
  - 20.2% and 9.0% for 8-core systems using 32Gb DRAM
  - Very close to the ideal scheme without refreshes

Outline

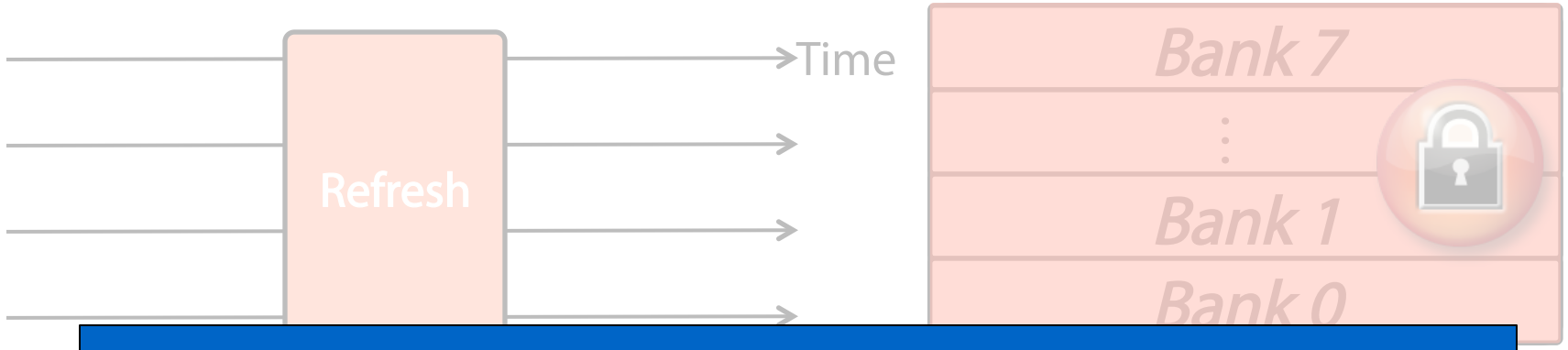
- **Motivation and Key Ideas**
- DRAM and Refresh Background
- Our Mechanisms
- Results

Refresh Penalty



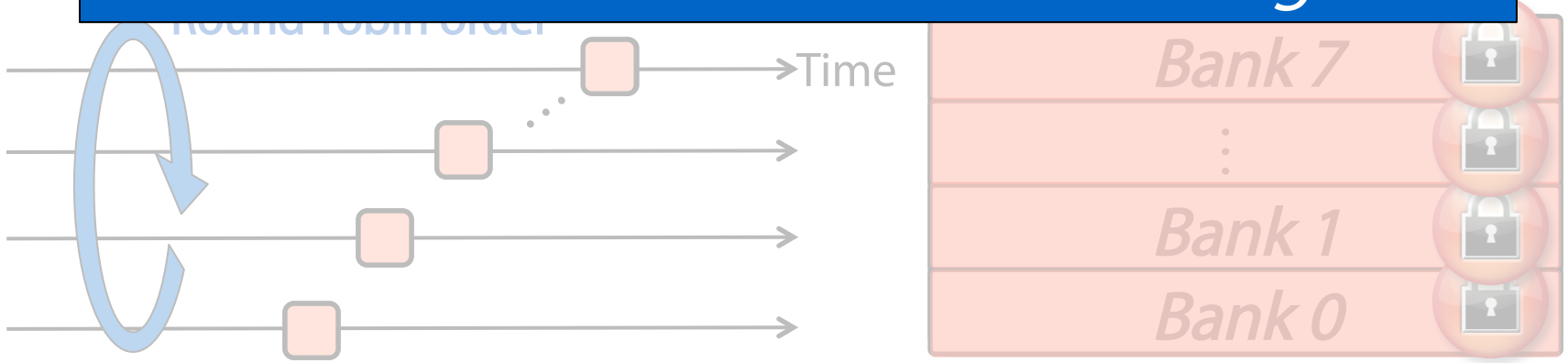
Existing Refresh Modes

All-bank refresh in commodity DRAM (DDRx)



Per-bank refresh allows accesses to other banks while a bank is refreshing

Per-



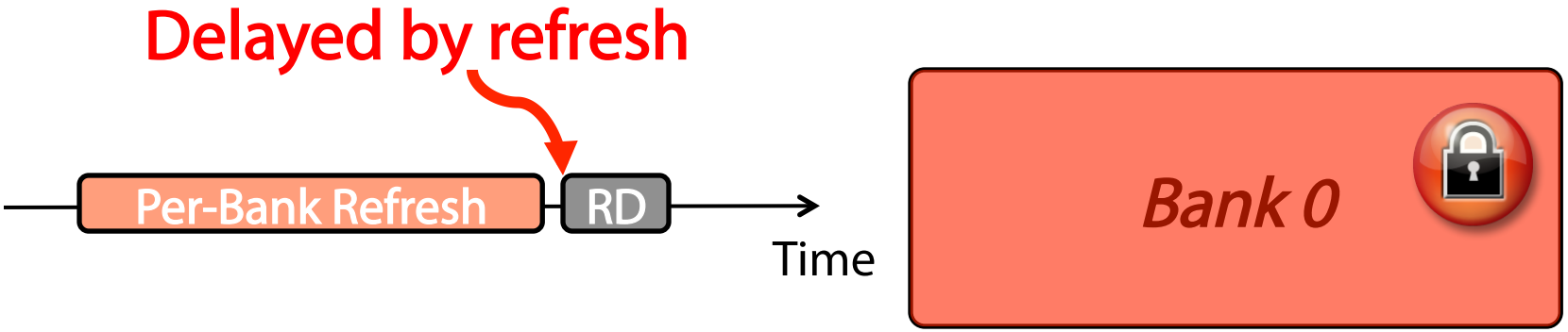
Shortcomings of Per-Bank Refresh

- Problem 1: Refreshes to different banks are scheduled in a strict round-robin order
  - The static ordering is hardwired into DRAM chips
  - Refreshes busy banks with many queued requests when other banks are idle

- Key idea: Schedule per-bank refreshes to idle banks opportunistically in a dynamic order

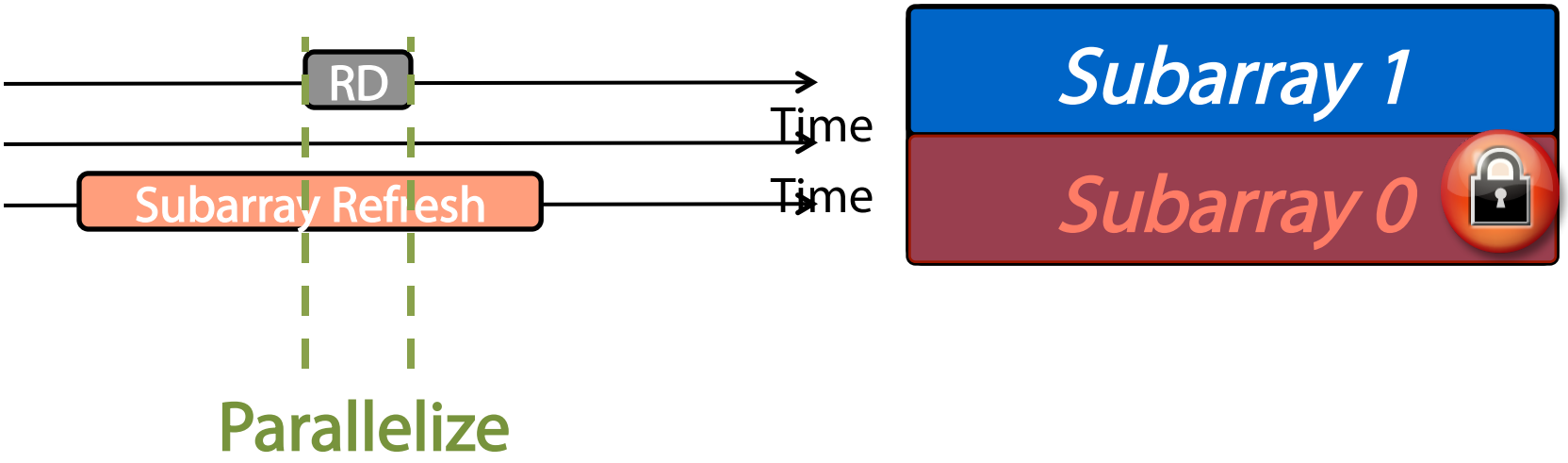
Shortcomings of Per-Bank Refresh

- Problem 2: Banks that are being refreshed cannot concurrently serve memory requests



Shortcomings of Per-Bank Refresh

- Problem 2: Refreshing banks cannot concurrently serve memory requests
- Key idea: Exploit **subarrays** within a bank to parallelize refreshes and accesses across **subarrays**



Outline

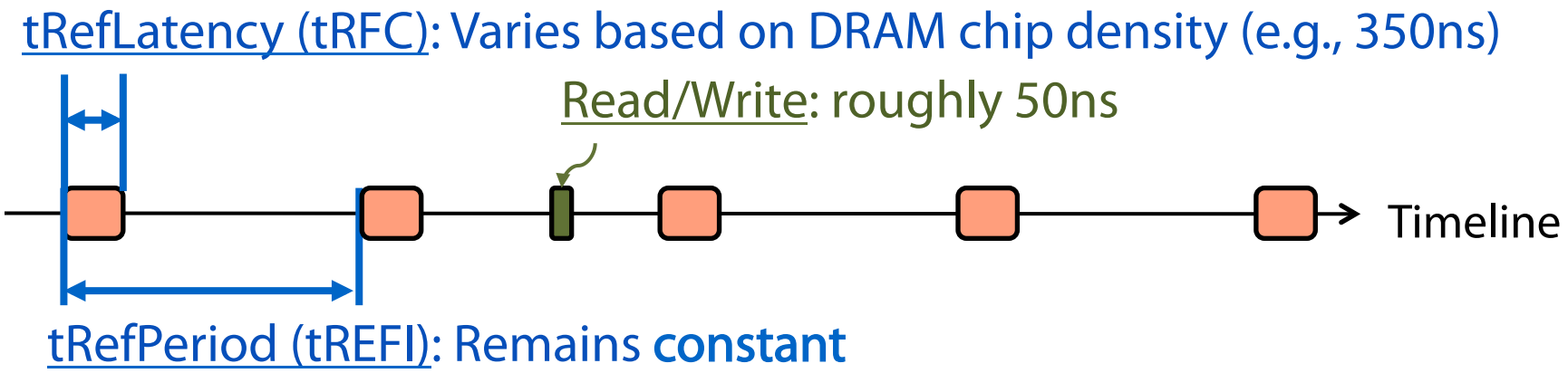
- Motivation and Key Ideas
- DRAM and Refresh Background
- Our Mechanisms
- Results

DRAM System Organization

- Banks can serve multiple requests in parallel

DRAM Refresh Frequency

- DRAM standard requires memory controllers to send **periodic refreshes** to DRAM

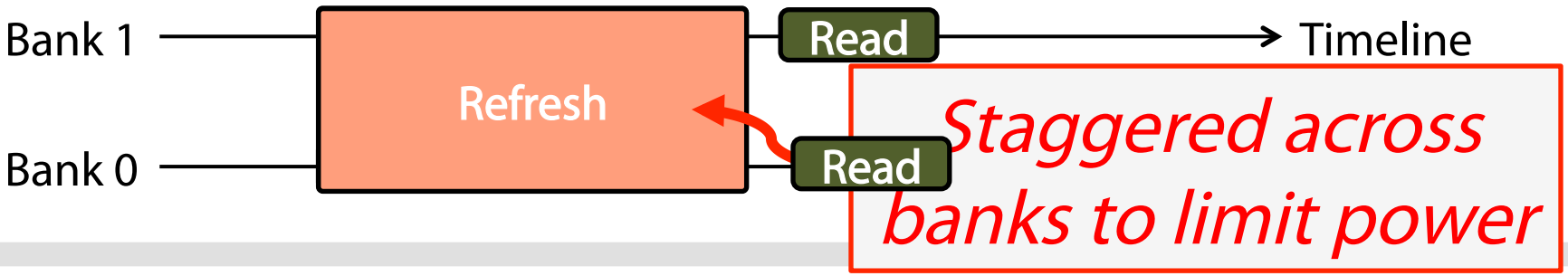


Increasing Performance Impact

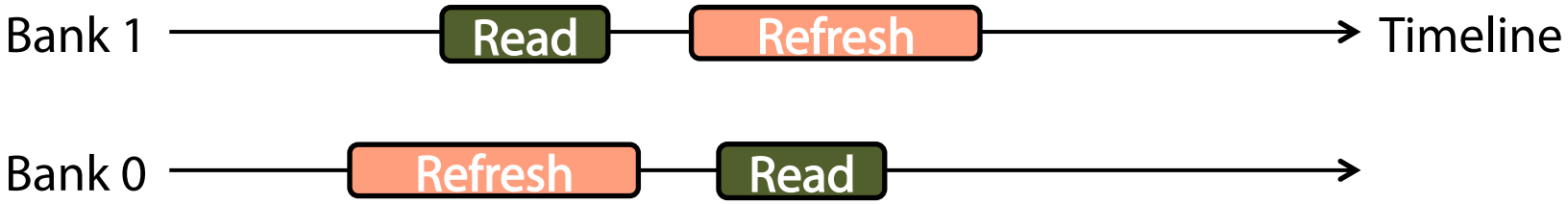
- DRAM is unavailable to serve requests for $\frac{tRefLatency}{tRefPeriod}$ of time
- **6.7%** for today's 4Gb DRAM
- Unavailability increases with higher density due to higher *tRefLatency*
  - **23% / 41%** for **future 32Gb / 64Gb DRAM**

All-Bank vs. Per-Bank Refresh

All-Bank Refresh: Employed in commodity DRAM (DDRx, LPDDRx)



Per-Bank Refresh: In mobile DRAM (LPDDRx)



Can serve memory accesses in parallel with refreshes across banks

Shortcomings of Per-Bank Refresh

- 1) Per-bank refreshes are **strictly scheduled** in round-robin order (as fixed by DRAM's internal logic)

- 2) A **refreshing bank** cannot serve memory accesses

Goal: Enable more parallelization between refreshes and accesses using practical mechanisms

Outline

- Motivation and Key Ideas
- DRAM and Refresh Background
- Our Mechanisms
  - 1. Dynamic Access-Refresh Parallelization (DARP)
  - 2. Subarray Access-Refresh Parallelization (SARP)
- Results

Our First Approach: DARP

- **Dynamic Access-Refresh Parallelization (DARP)**
  - An improved scheduling policy for per-bank refreshes
  - Exploits refresh scheduling flexibility in DDR DRAM

- Component 1: **Out-of-order per-bank refresh**
  - Avoids poor static scheduling decisions
  - Dynamically issues per-bank refreshes to idle banks

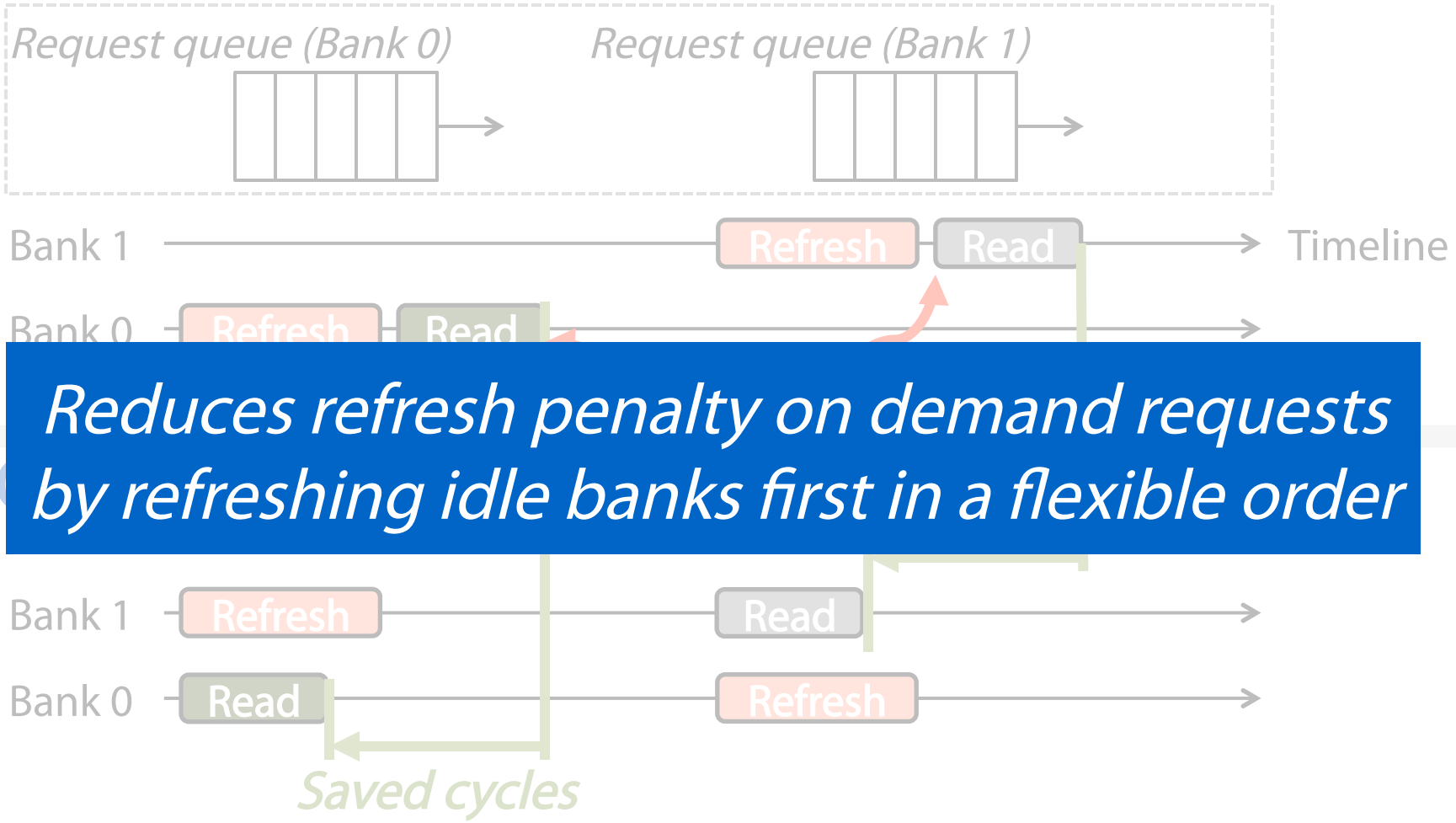
- Component 2: **Write-Refresh Parallelization**
  - Avoids refresh interference on latency-critical reads
  - Parallelizes refreshes with **a batch of writes**

1) Out-of-Order Per-Bank Refresh

- **Dynamic scheduling policy** that prioritizes refreshes to idle banks
- **Memory controllers** decide which bank to refresh

1) Out-of-Order Per-Bank Refresh

Baseline: Round robin

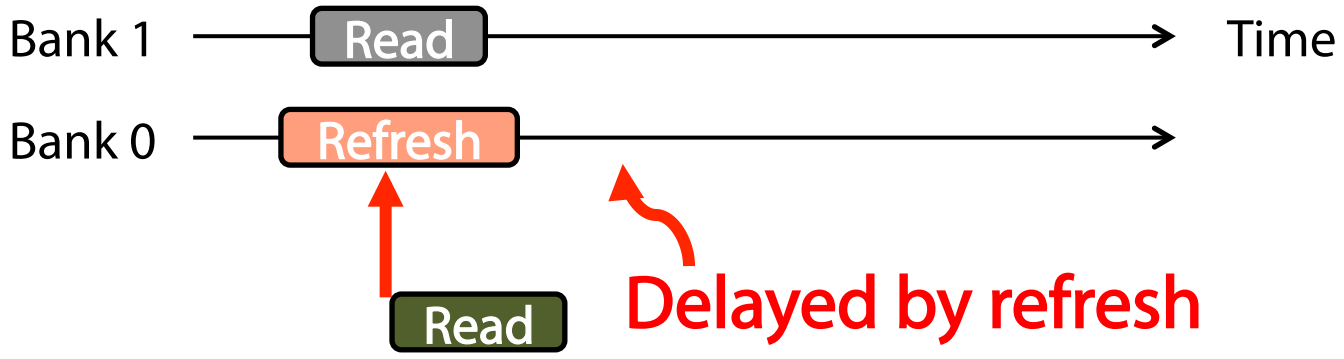


Outline

- Motivation and Key Ideas
- DRAM and Refresh Background
- Our Mechanisms
  - 1. Dynamic Access-Refresh Parallelization (DARP)
    - 1) Out-of-Order Per-Bank Refresh
    - 2) Write-Refresh Parallelization
  - 2. Subarray Access-Refresh Parallelization (SARP)
- Results

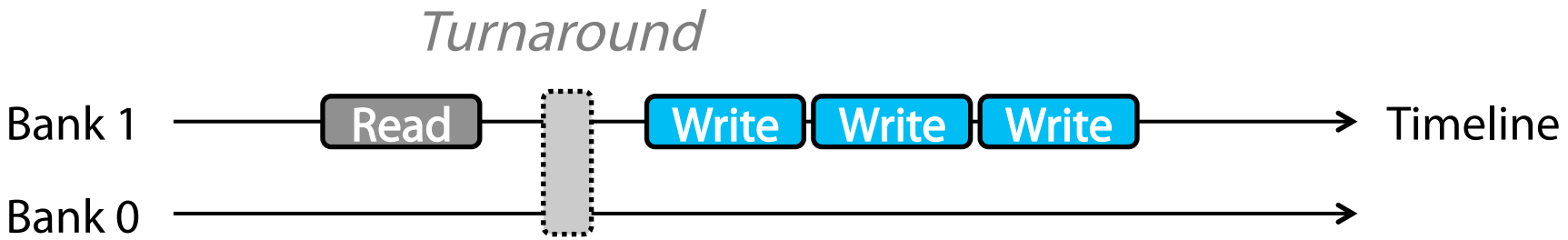
Refresh Interference on Upcoming Requests

- Problem: A refresh may collide with an upcoming request in the near future



DRAM Write Draining

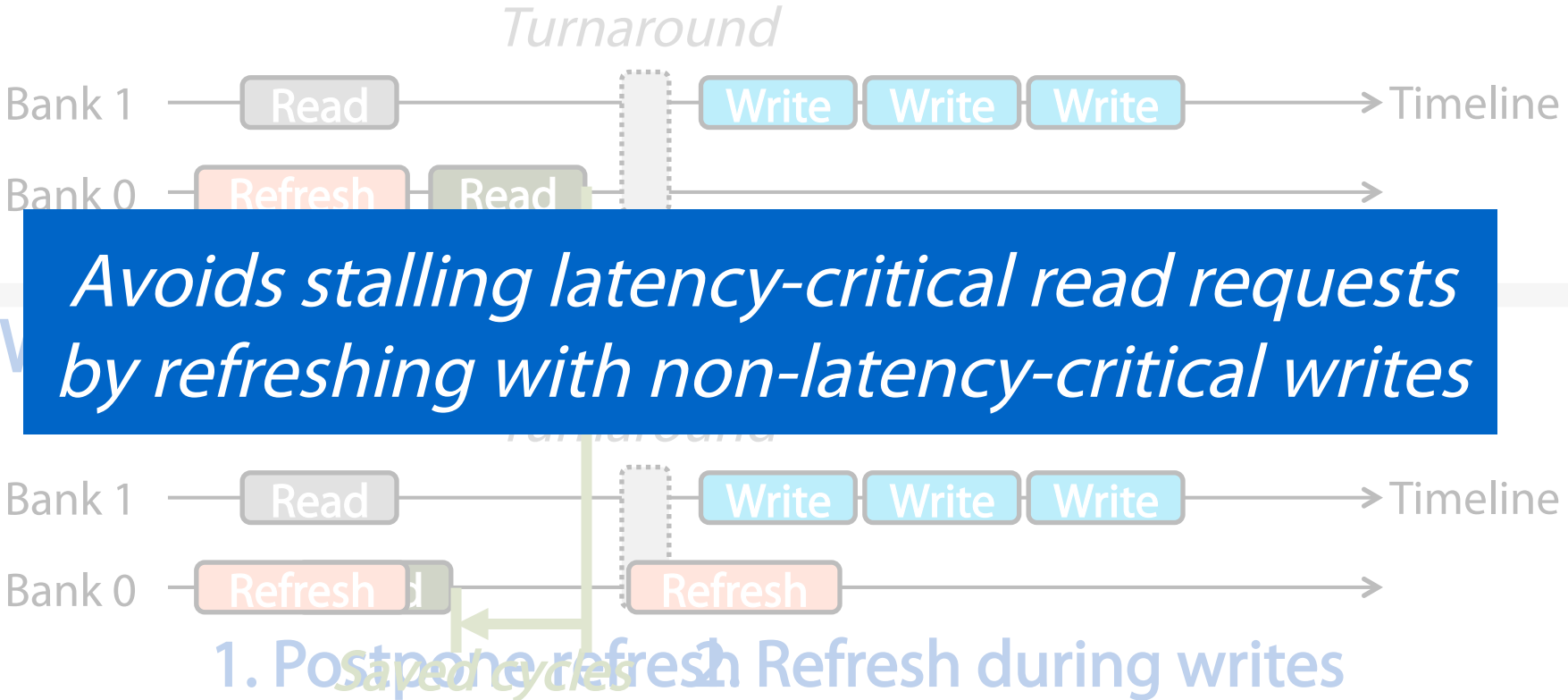
- Observations:
- 1) **Bus-turnaround latency** when transitioning from writes to reads or vice versa
  - To mitigate **bus-turnaround latency**, writes are typically drained to DRAM in a batch during a period of time
- 2) Writes are not **latency-critical**



2) Write-Refresh Parallelization

- Proactively schedules refreshes when banks are serving **write batches**

Baseline



Avoids stalling latency-critical read requests by refreshing with non-latency-critical writes

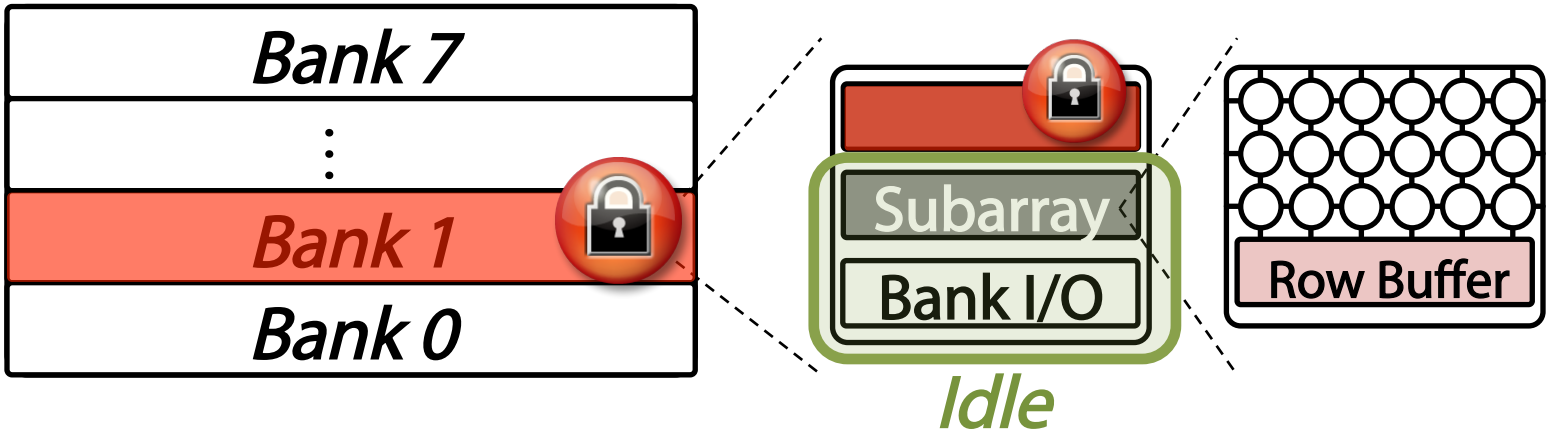
Outline

- Motivation and Key Ideas
- DRAM and Refresh Background
- Our Mechanisms
  - 1. Dynamic Access-Refresh Parallelization (DARP)
  - 2. Subarray Access-Refresh Parallelization (SARP)
- Results

Our Second Approach: SARP

Observations:

1. A bank is further divided into **subarrays**
   - Each has its own row buffer to perform refresh operations



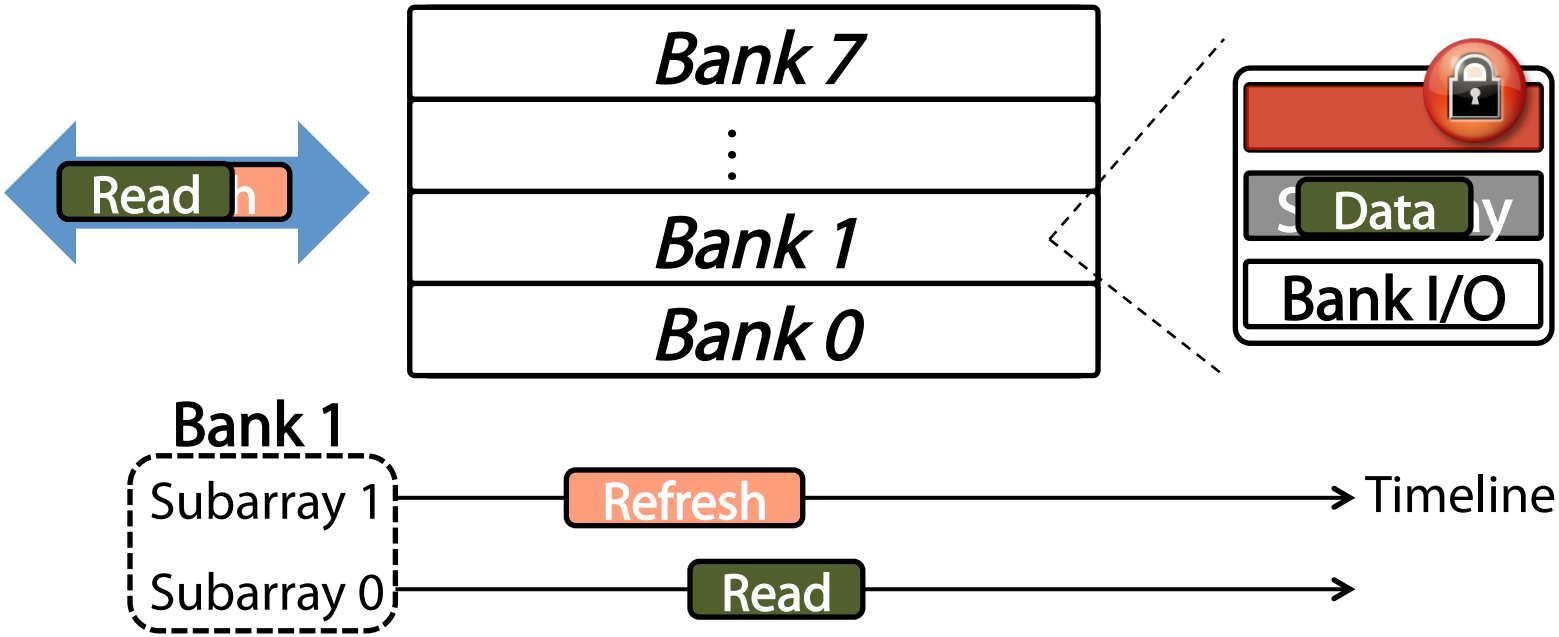
2. Some **subarrays** and **bank I/O** remain completely **idle** during refresh

Our Second Approach: SARP

- **Subarray Access-Refresh Parallelization (SARP):**
  - Parallelizes refreshes and accesses **within a bank**

Our Second Approach: SARP

- **Subarray Access-Refresh Parallelization (SARP):**
  - Parallelizes refreshes and accesses **within a bank**



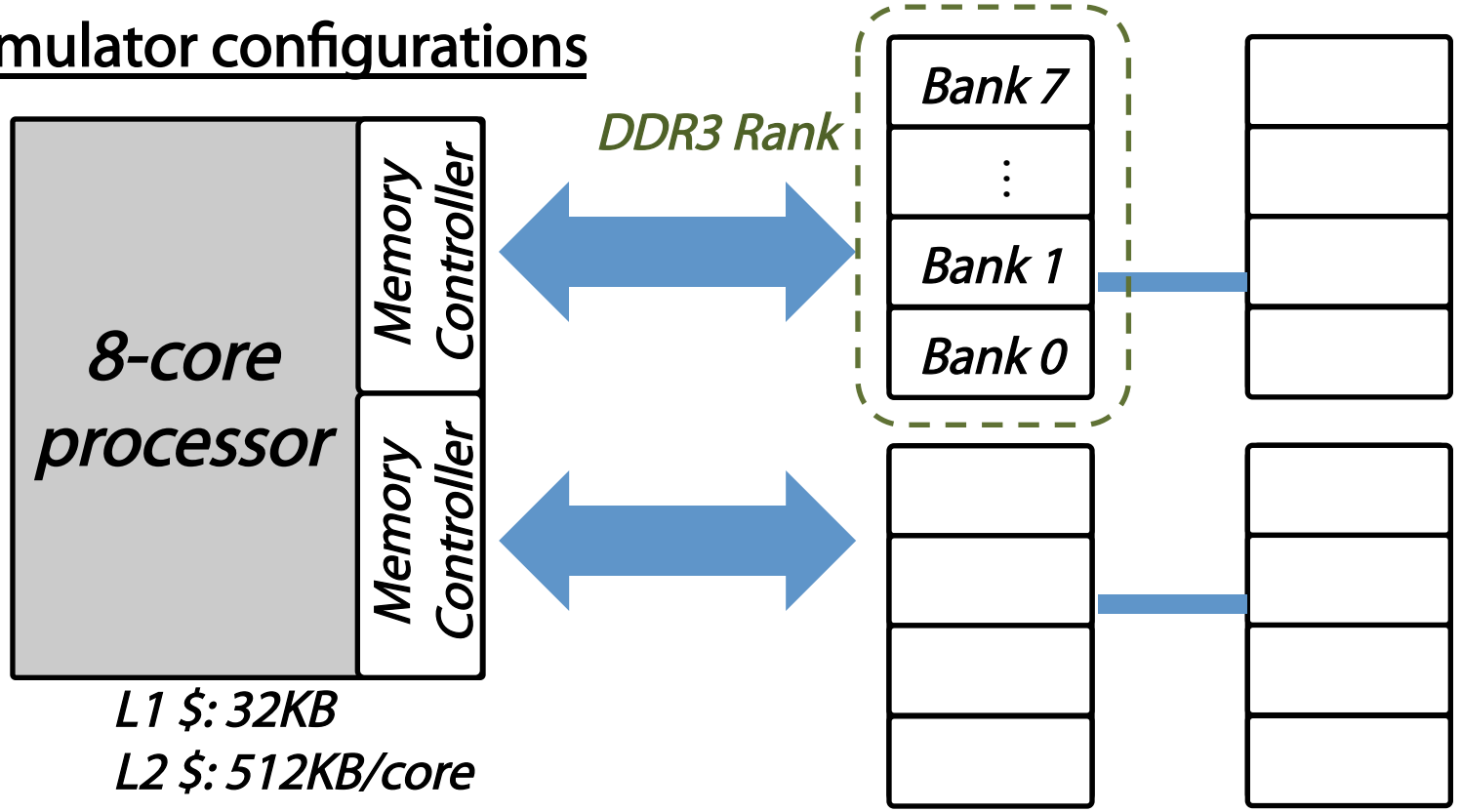
Very modest DRAM modifications: 0.71% die area overhead

Outline

- Motivation and Key Ideas
- DRAM and Refresh Background
- Our Mechanisms
- **Results**

Methodology

Simulator configurations

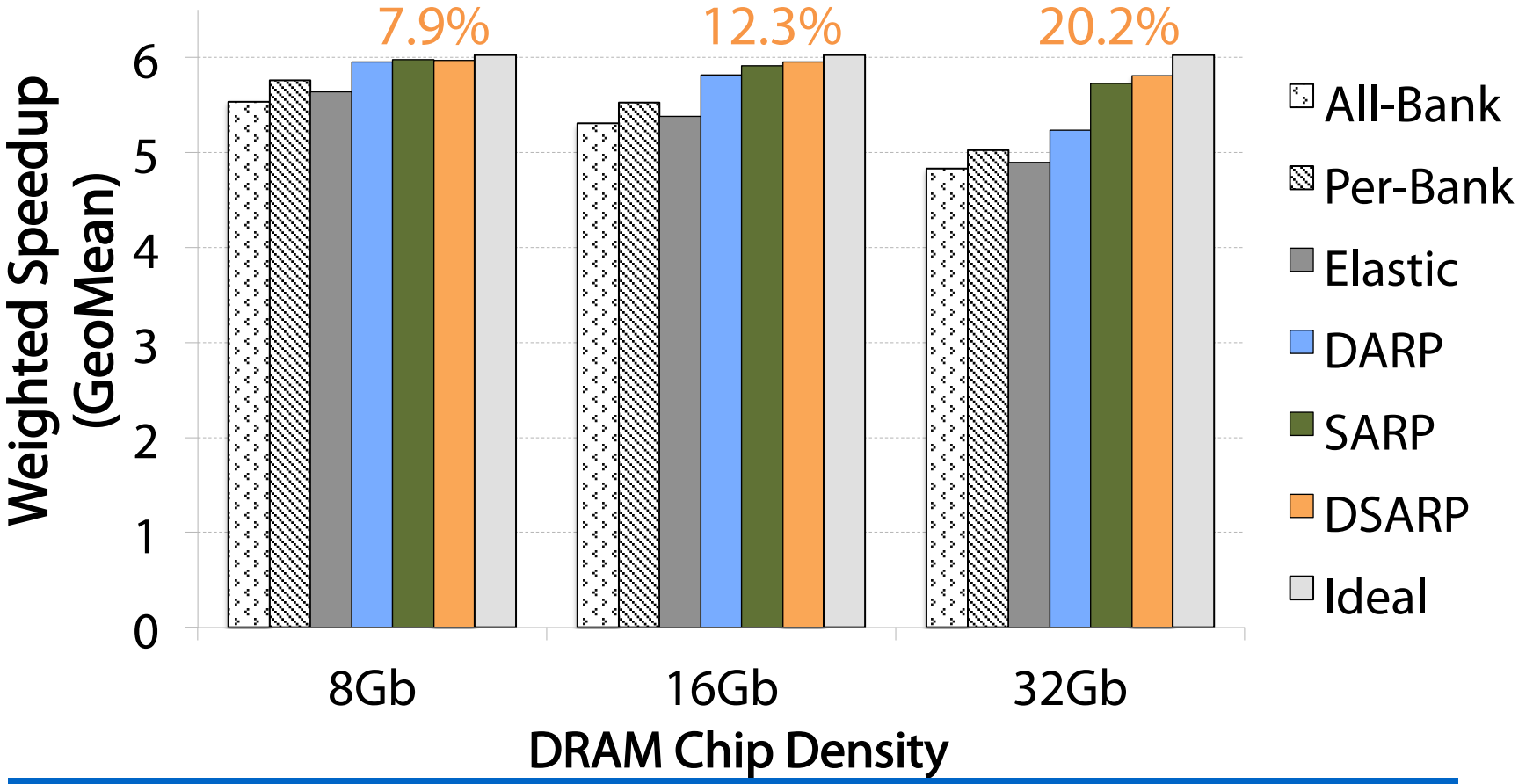


- **100 workloads**: SPEC CPU2006, STREAM, TPC-C/H, random access
- **System performance metric**: *Weighted speedup*

Comparison Points

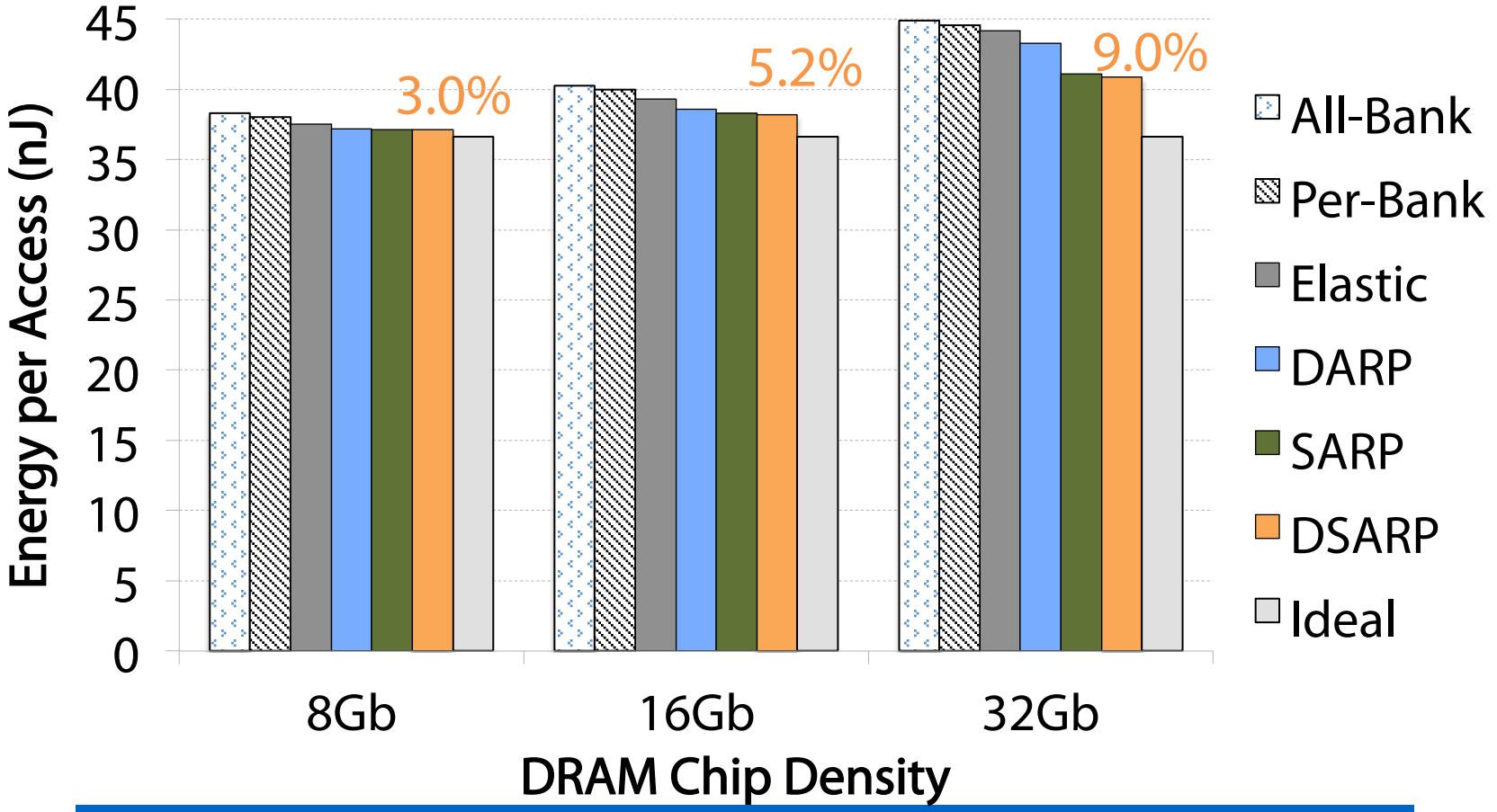
- **All-bank refresh** [DDR3, LPDDR3, …]
- **Per-bank refresh** [LPDDR3]
- **Elastic refresh** [Stuecheli et al., MICRO '10]:
  - Postpones refreshes by a time delay based on the predicted rank idle time to avoid interference on memory requests
  - Proposed to schedule all-bank refreshes without exploiting per-bank refreshes
  - Cannot parallelize refreshes and accesses within a rank
- **Ideal (no refresh)**

System Performance



*2. Consistent system performance improvement across DRAM densities (within **0.9%, 1.2%, and 3.8%** of ideal)*

Energy Efficiency



Consistent reduction on energy consumption

Other Results and Discussion in the Paper

- Detailed multi-core results and analysis
- Result breakdown based on memory intensity
- Sensitivity results on number of cores, subarray counts, refresh interval length, and DRAM parameters
- Comparisons to DDR4 fine granularity refresh

Executive Summary

- DRAM **refresh interferes with memory accesses**
  - Degrades system performance and energy efficiency
  - Becomes exacerbated as DRAM density increases
- Goal: Serve memory accesses in parallel with refreshes to reduce refresh interference on demand requests
- Our mechanisms:
  - 1. Enable more parallelization between refreshes and accesses across different banks with new per-bank refresh scheduling algorithms
  - 2. Enable serving accesses concurrently with refreshes in the same bank by exploiting DRAM subarrays
- Improve system performance and energy efficiency for a wide variety of different workloads and DRAM densities
  - 20.2% and 9.0% for 8-core systems using 32Gb DRAM
  - Very close to the ideal scheme without refreshes

Improving DRAM Performance by Parallelizing Refreshes with Accesses

Kevin Chang

Donghyuk Lee, Zeshan Chishti, Alaa Alameldeen,
Chris Wilkerson, Yoongu Kim, Onur Mutlu

SAFARI Carnegie Mellon intel