Bounding Memory Interference Delay in COTS-based Multi-Core Systems

Hyoseung Kim Dionisio de Niz Björn Andersson

Mark Klein Onur Mutlu Raj Rajkumar

Electrical & Computer ENGINEERING

Software Engineering Institute

Carnegie Mellon University

Copyright 2014 Carnegie Mellon University and IEEE

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

This material has been approved for public release and unlimited distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM-0001188

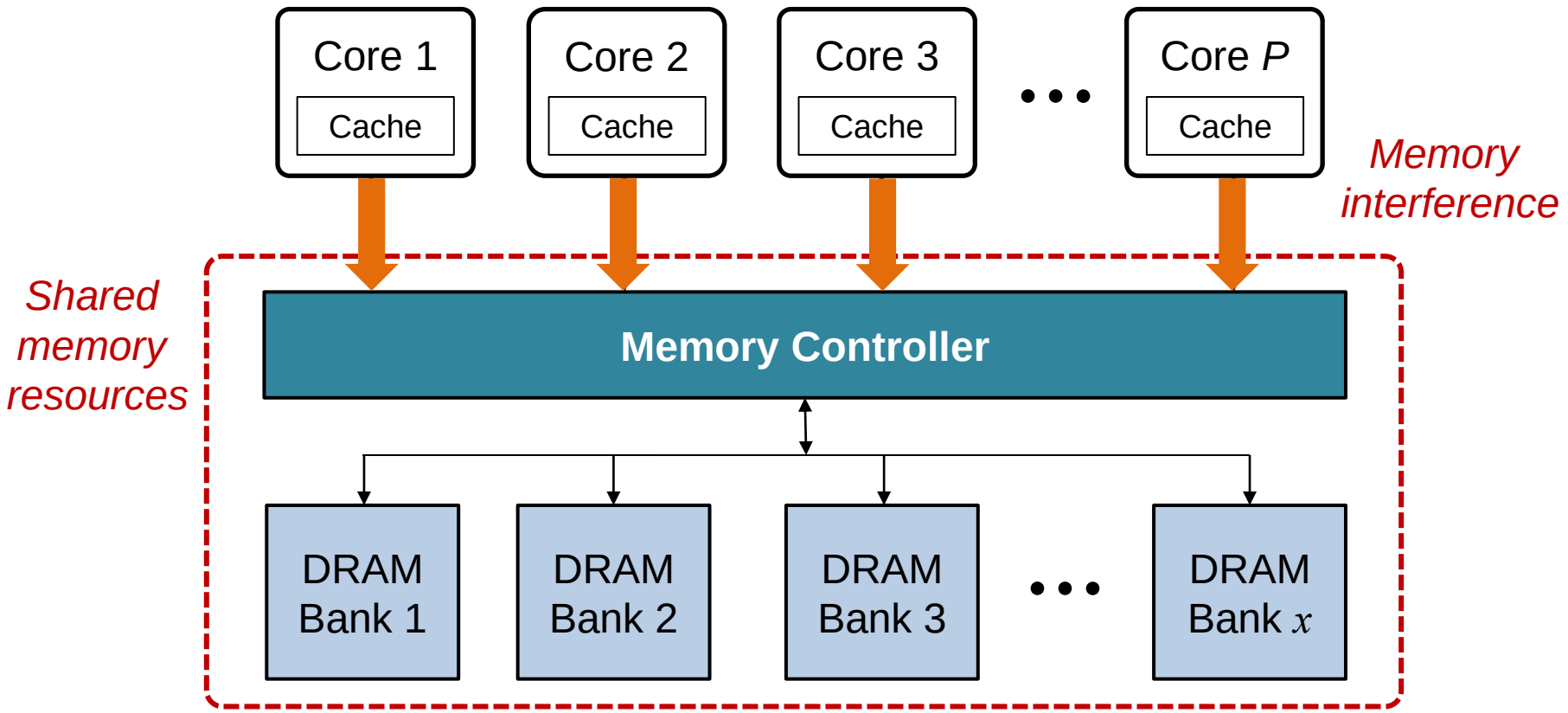
Why Multi-Core Processors?

- **Processor development trend**
  - Increasing overall performance by integrating multiple cores

- **Embedded systems: Actively adopting multi-core CPUs**
  - **Automotive:**
    - Freescale i.MX6 4-core CPU
    - NVIDIA Tegra K1 platform

  - **Avionics and defense:**
    - Rugged Intel i7 single board computers
    - Freescale P4080 8-core CPU

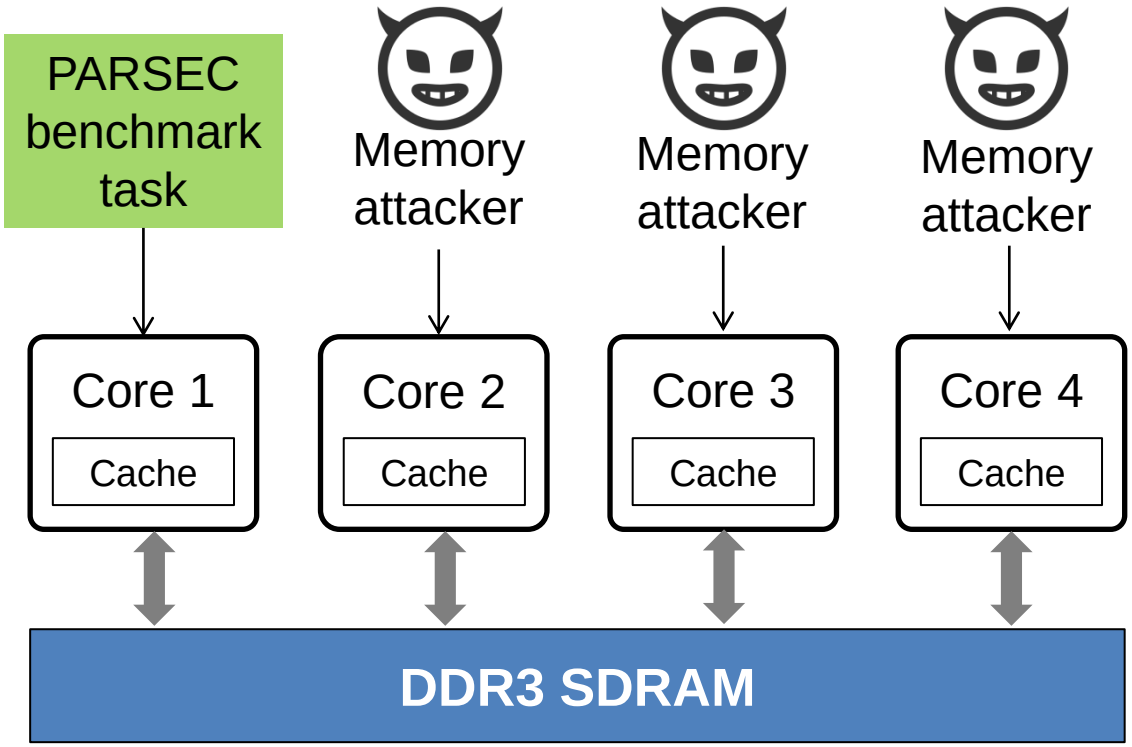
Multi-Core Memory System



An upper bound on the memory interference delay is needed to check task schedulability

Impact of Memory Interference (1/2)

- Task execution times with memory attacker tasks
  - Intel i7 4-core processor + DDR3-1333 SDRAM 8GB

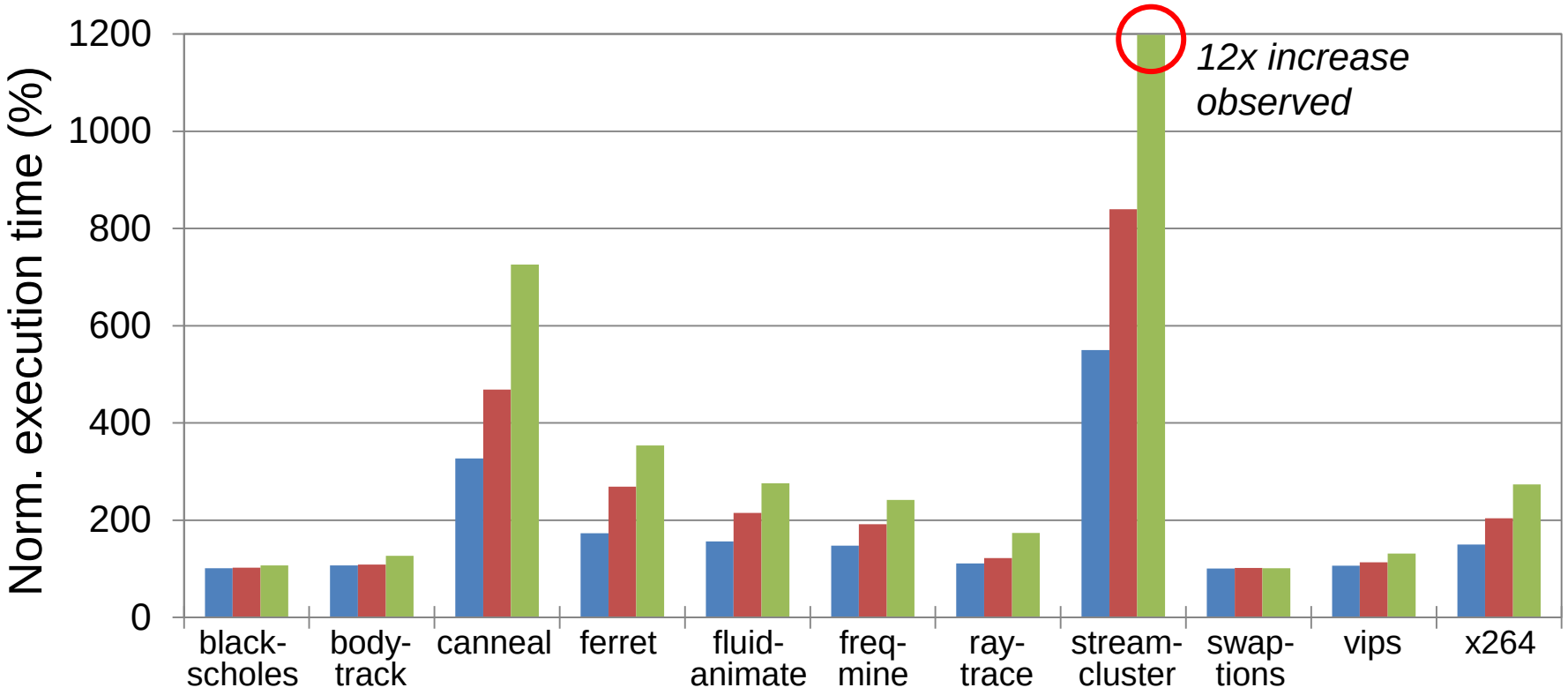


* S/W cache partitioning is used

Impact of Memory Interference (2/2)

- 1 attacker → Max 5.5x increase
- 2 attackers → Max 8.4x increase
- 3 attackers → Max 12x increase

We should predict, bound and reduce the memory interference delay!



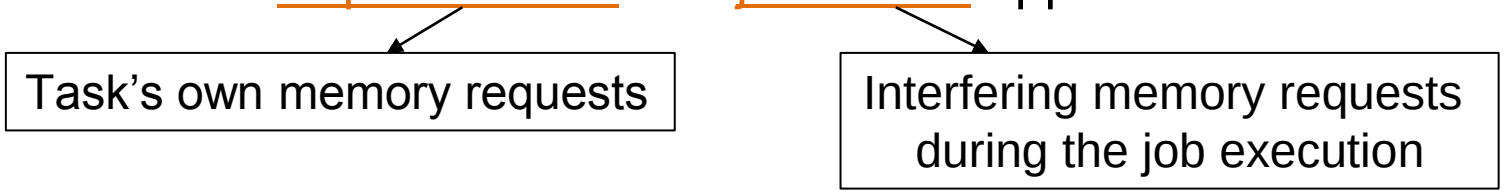
Issues with Memory Models

- **Q1. Can we assume that each memory request takes a constant service time?** ***No.***
  - The memory access time varies considerably depending on the requested address and the rank/bank/bus states
- **Q2. Can we assume memory requests are serviced in either Round-Robin or FIFO order?** ***No.***
  - Memory requests arriving early may be serviced later than ones arriving later in today's COTS memory systems

An over-simplified memory model may produce pessimistic or optimistic estimates on the memory interference delay

Our Approach

- **Explicitly considers the timing characteristics of major DRAM resources**
  - Rank/bank/bus timing constraints (JEDEC standard)
  - Request re-ordering effect
- **Bounding memory interference delay for a task**
  - Combines request-driven and job-driven approaches



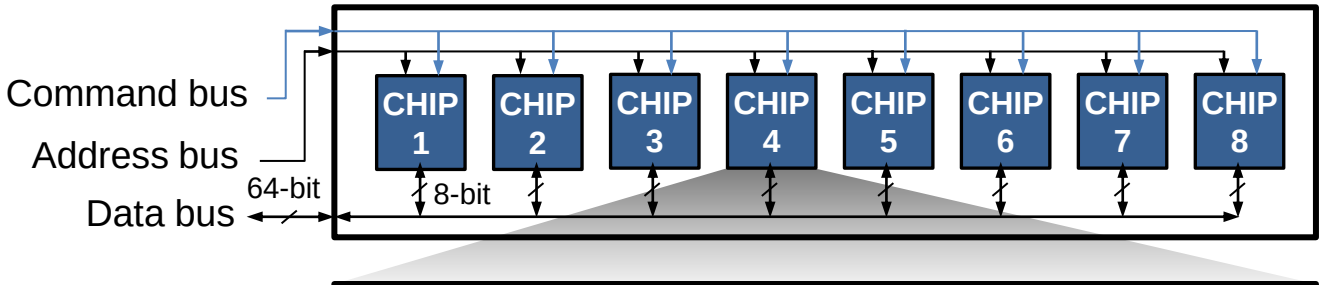
- **Software DRAM bank partitioning awareness**
  - Analyzes the effect of dedicated and shared DRAM banks

Outline

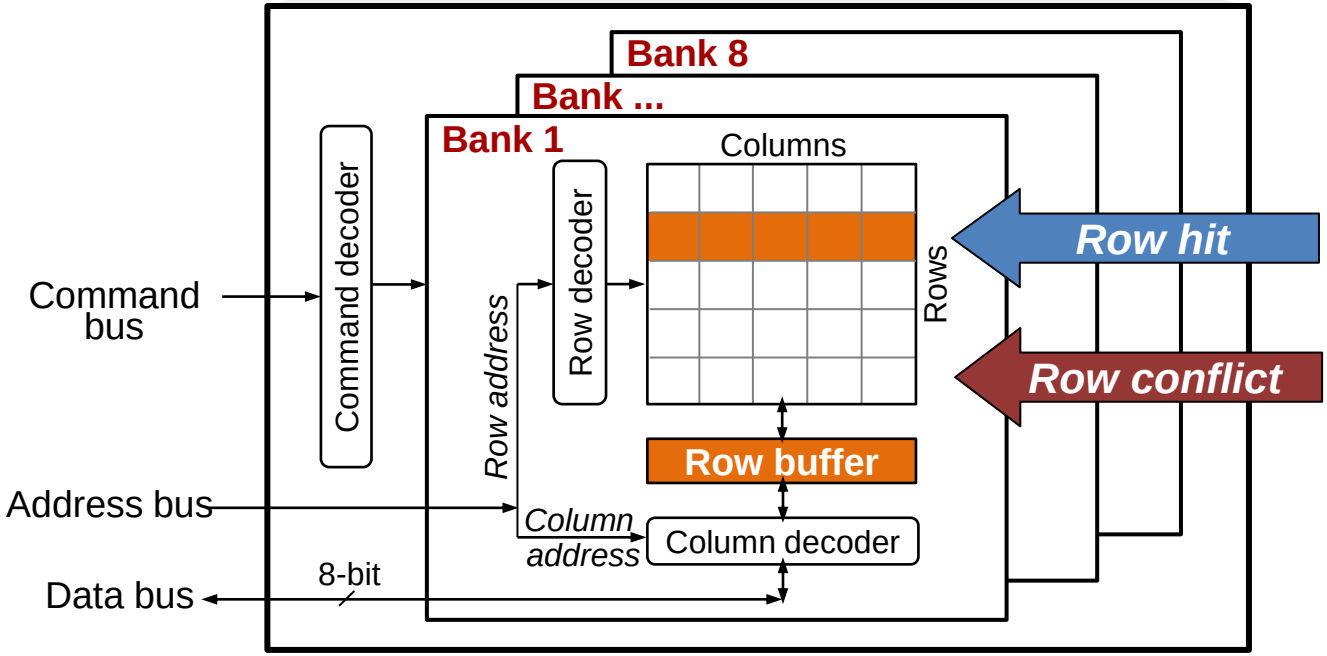
- Introduction
- **Details on DRAM Systems**
  - DRAM organization
  - Memory controller and scheduling policy
  - DRAM bank partitioning
- **Bounding Memory Interference Delay**
- **Evaluation**
- **Conclusion**

DRAM Organization

DRAM Rank

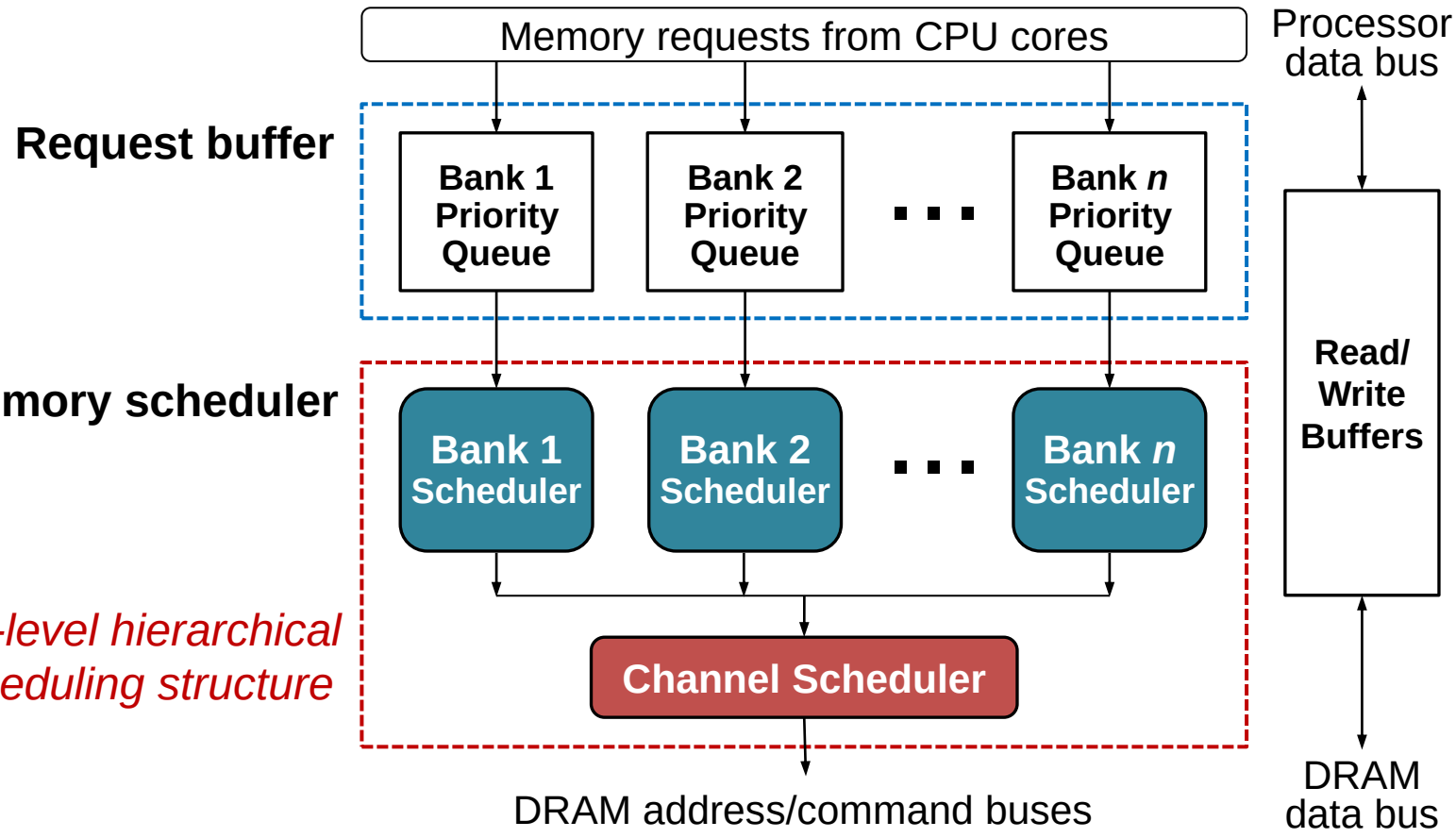


DRAM Chip



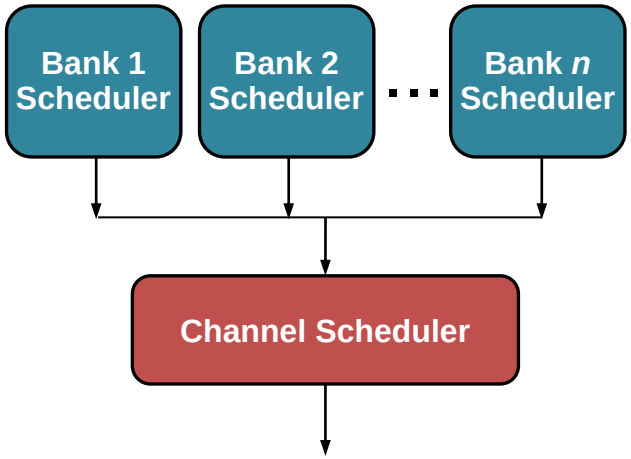
DRAM access latency varies depending on which row is stored in the row buffer

Memory Controller



Memory Scheduling Policy

- **FR-FCFS: First-Ready, First-Come First-Serve**
  - Goal: maximize DRAM throughput → Maximize row buffer hit rate



1. **Bank scheduler**
   - Considers bank timing constraints
   - Prioritizes row-hit requests
   - In case of tie, prioritizes older requests
2. **Channel scheduler**
   - Considers channel timing constraints
   - Prioritizes older requests

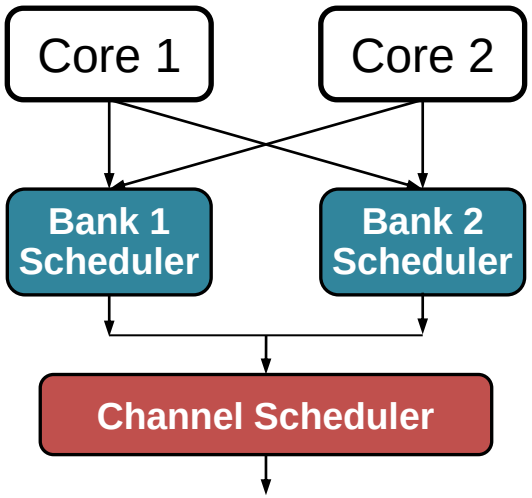
Memory access interference occurs at ***both*** bank and channel schedulers

- *Intra-bank interference at bank scheduler*
- *Inter-bank interference at channel scheduler*

DRAM Bank Partitioning

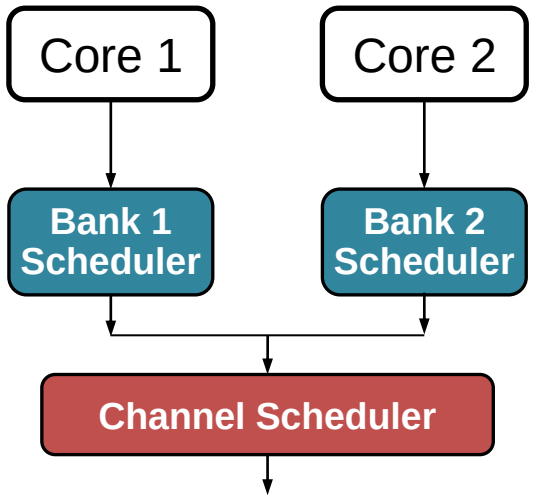
- **Prevents intra-bank interference** by dedicating different DRAM banks to each core
  - Can be supported in the OS kernel

(1) w/o bank partitioning



Intra-bank and inter-bank interference

(2) w/ bank partitioning



Only inter-bank interference

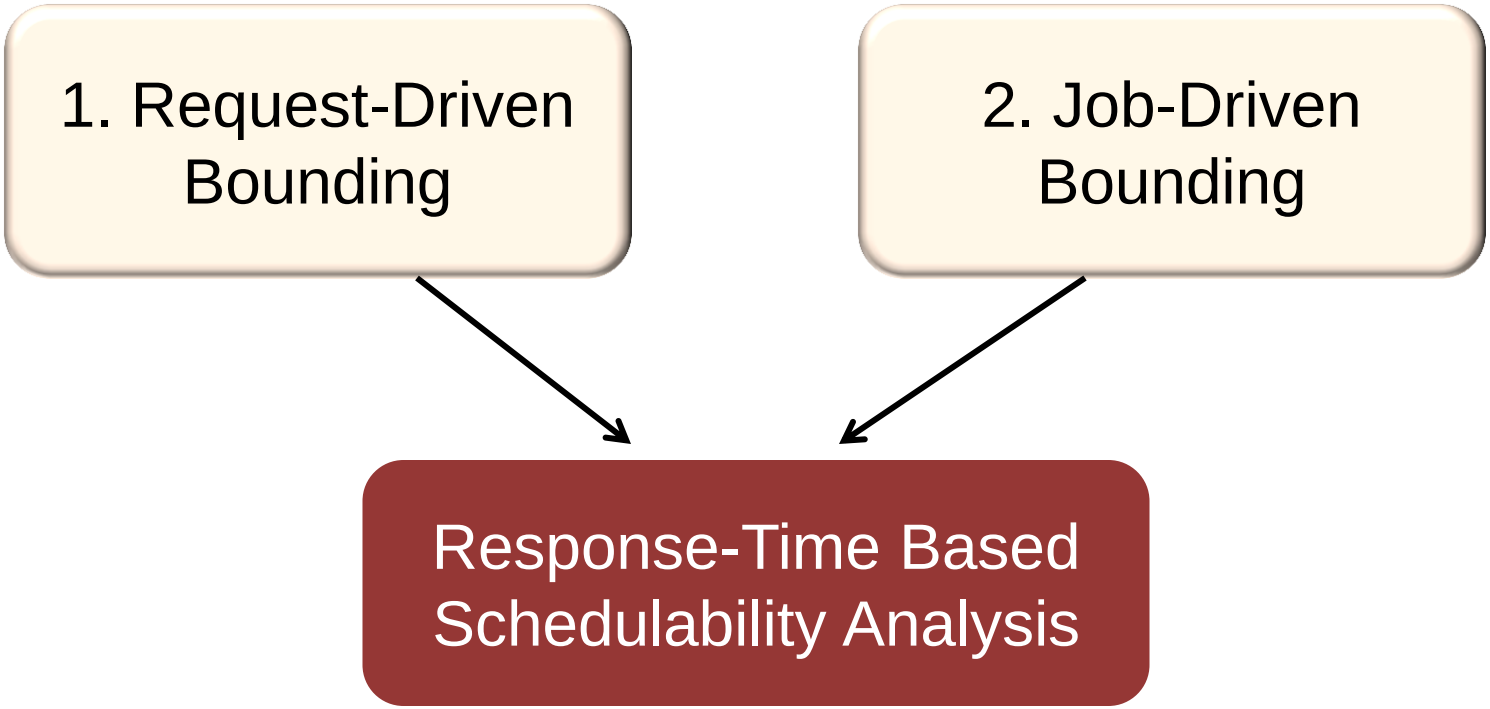
Outline

- Introduction
- Details on DRAM Systems
- **Bounding Memory Interference Delay**
  - System Model
  - Request-Driven Bounding
  - Job-Driven Bounding
  - Schedulability Analysis
- **Evaluation**
- **Conclusion**

System Model

- **Task τ_i: (C_i, T_i, D_i, H_i)**
  - C_i: Worst-case execution time (WCET) of any job of task τ_i, when it executes in isolation
  - T_i: Period
  - D_i: Relative deadline
  - H_i: Maximum DRAM requests generated by any job of τ_i
    - No assumptions on the memory access pattern (i.e., access rate)
- **Partitioned fixed-priority preemptive task scheduling**
- **DDR SDRAM main memory system**
  - Software DRAM bank partitioning is used
- **No cache interference**

Bounding Memory Interference Delay



Request-Driven (RD) Approach

- Focuses on
  - # of memory requests generated by task itself (H_i)
  - Maximum delay imposed on each request
    - Bounded by using DRAM and CPU core params
    - *Not* by using task params of other tasks

Per-request interference delay

- **Intra-bank interference delay**
  - Bank-level timing constraints
  - Re-ordering effect (consecutive row-hits)
  - Zero, if τ_i does not share bank partitions with tasks on other cores
- **Inter-bank interference delay**
  - Channel-level timing constraints

- **Total interference delay** $= H_i \times (\text{per-request interference delay})$

Job-Driven (JD) Approach

- Focuses on the # of interfering memory requests generated by other tasks running in parallel
  → Does not use the # of own memory requests from a task

- **Total interference delay for task τ_i**
  - Captures # of interfering memory requests during a time interval t
  - Assumes that all these interfering memory requests are processed ahead of any request of task τ_i
  - Combines intra-bank and inter-bank interference

Response-Time Test

- **Memory interference delay cannot exceed any results from the RD and JD approaches**
  - We take the smaller result from the two approaches

- **Extended response-time test**

$$R_i^{k+1} = C_i + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot C_j$$

Classical iterative response-time test

$$+ \min \left\{ H_i \cdot RD_p + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot H_j \cdot RD_p, \; JD_p(R_i^k) \right\}$$

Request-Driven (RD) Approach

Job-Driven (JD) Approach

Outline

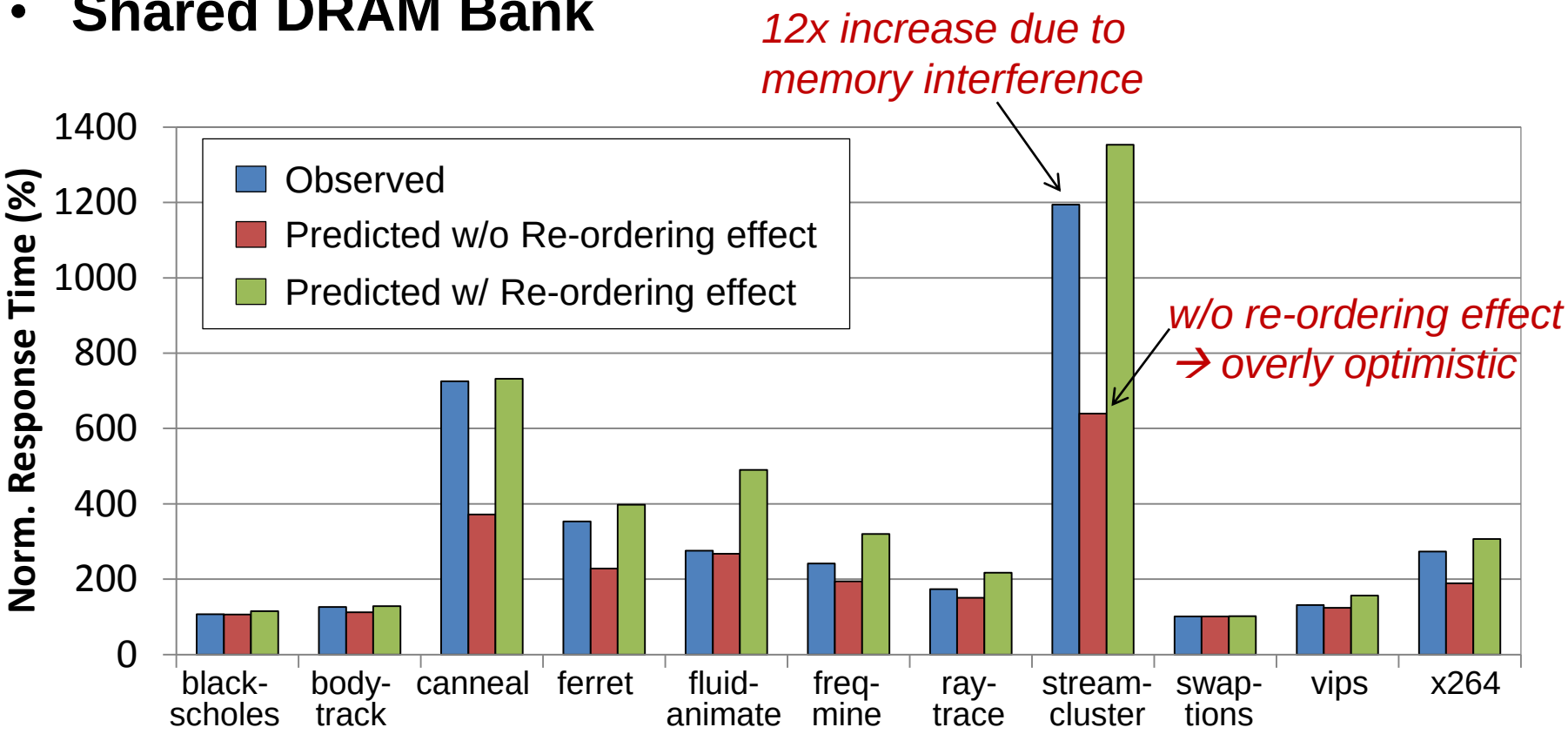
- Introduction
- Details on DRAM Systems
- Bounding Memory Interference Delay
- **Evaluation**
- **Conclusion**

Experimental Setup

- **Target system**
  - **Linux/RK** — **Cache partitioning**: to avoid cache interference; **Bank partitioning**: to test both private and shared banks
  - Intel i7-2600 quad-core processor
  - DDR3-1333 SDRAM (single channel configuration)
- **Workload:** PARSEC benchmarks
- **Methodology**
  - Compare the observed and predicted response times in the presence of memory interference
  - Core 1: Each PARSEC benchmark
  - Core 2, 3 and 4: Tasks generating interfering memory requests
    - Severe and non-severe memory interference (modified versions of the *stream* benchmark)

Severe Memory Interference (1 of 2)

- **Shared DRAM Bank**

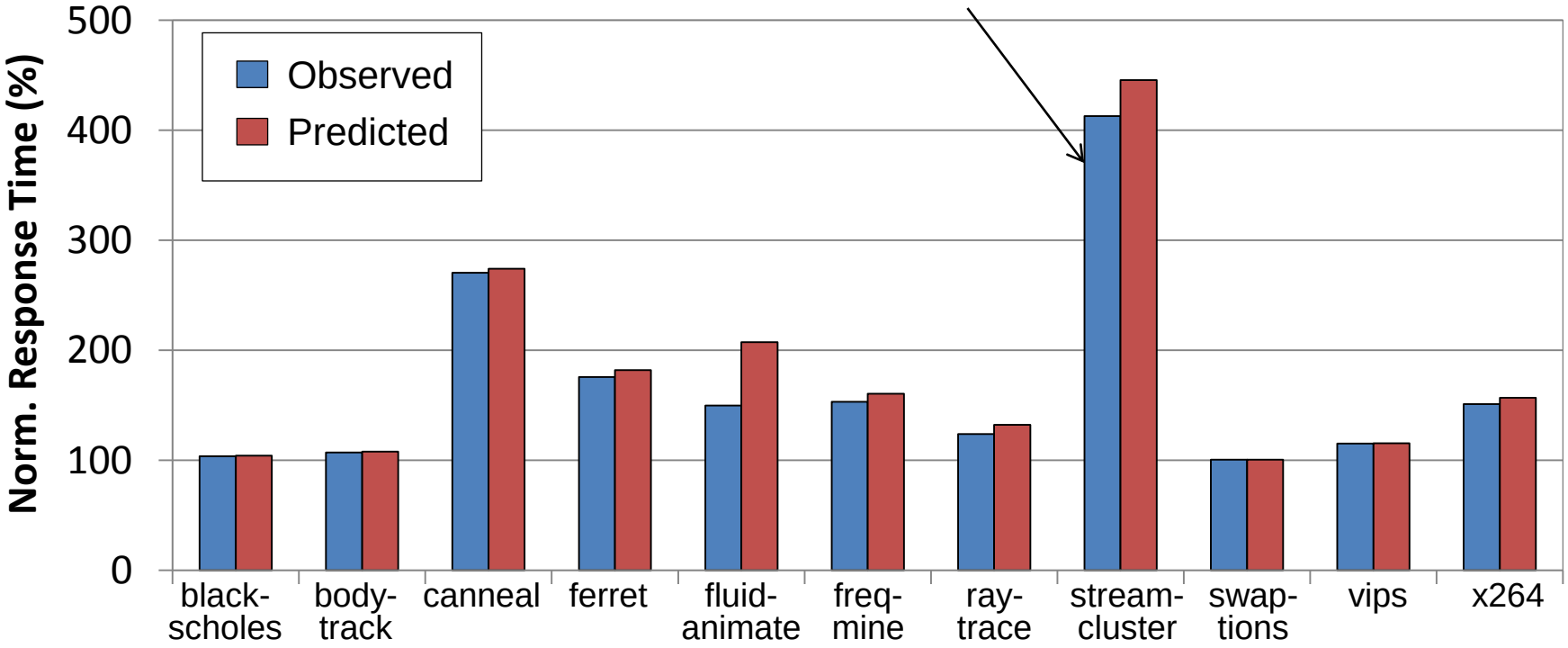


Request re-ordering effect should be considered in COTS memory systems

Severe Memory Interference (2 of 2)

- **Private DRAM Bank**

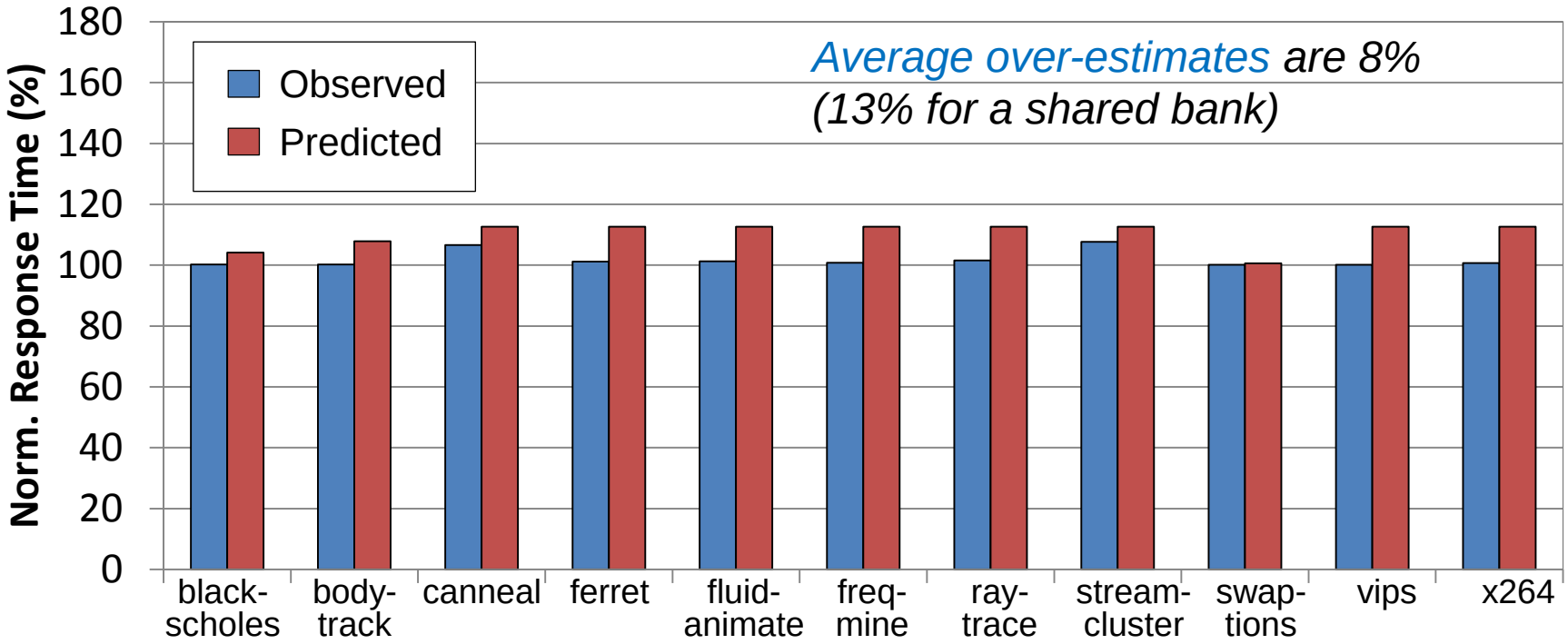
4.1x increase → DRAM bank partitioning helps reducing the memory interference



Our analysis enables the **quantification** of the benefit of DRAM bank partitioning

Non-severe Memory Interference

- **Private DRAM Bank**



Our analysis bounds memory interference delay with **low pessimism** under both high and low memory contentions

Conclusions

- **Analysis for bounding memory interference**
  - Based on a realistic memory model
    - JEDEC DDR3 SDRAM standard
    - FR-FCFS policy of the memory controller
    - Shared and private DRAM banks
  - Combination of the request-driven and job-driven approaches
    - Reduces pessimism in analysis (8% under severe interference)
- **Advantages**
  - Does not require any modifications to hardware components or application software
  - Readily applicable to COTS-based multicore real-time systems
- **Future work**
  - Pre-fetcher, multi-channel memory systems