COTS Multicore Processors in Avionics Systems: Challenges and Solutions

Dionisio de Niz

Bjorn Andersson and Lutz Wrage

dionisio@sei.cmu.edu, baandersson@sei.cmu.edu, lwrage@sei.cmu.edu

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **06 JAN 2015**	2. REPORT TYPE **N/A**	3. DATES COVERED

4. TITLE AND SUBTITLE **COTS Multicore Processors in Avionics Systems: Challenges and Solutions**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Wrage /Dionisio de Niz Bjorn Andersson Lutz**	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited.

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**28**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Why Multi-Core Processors?

Processor development trend

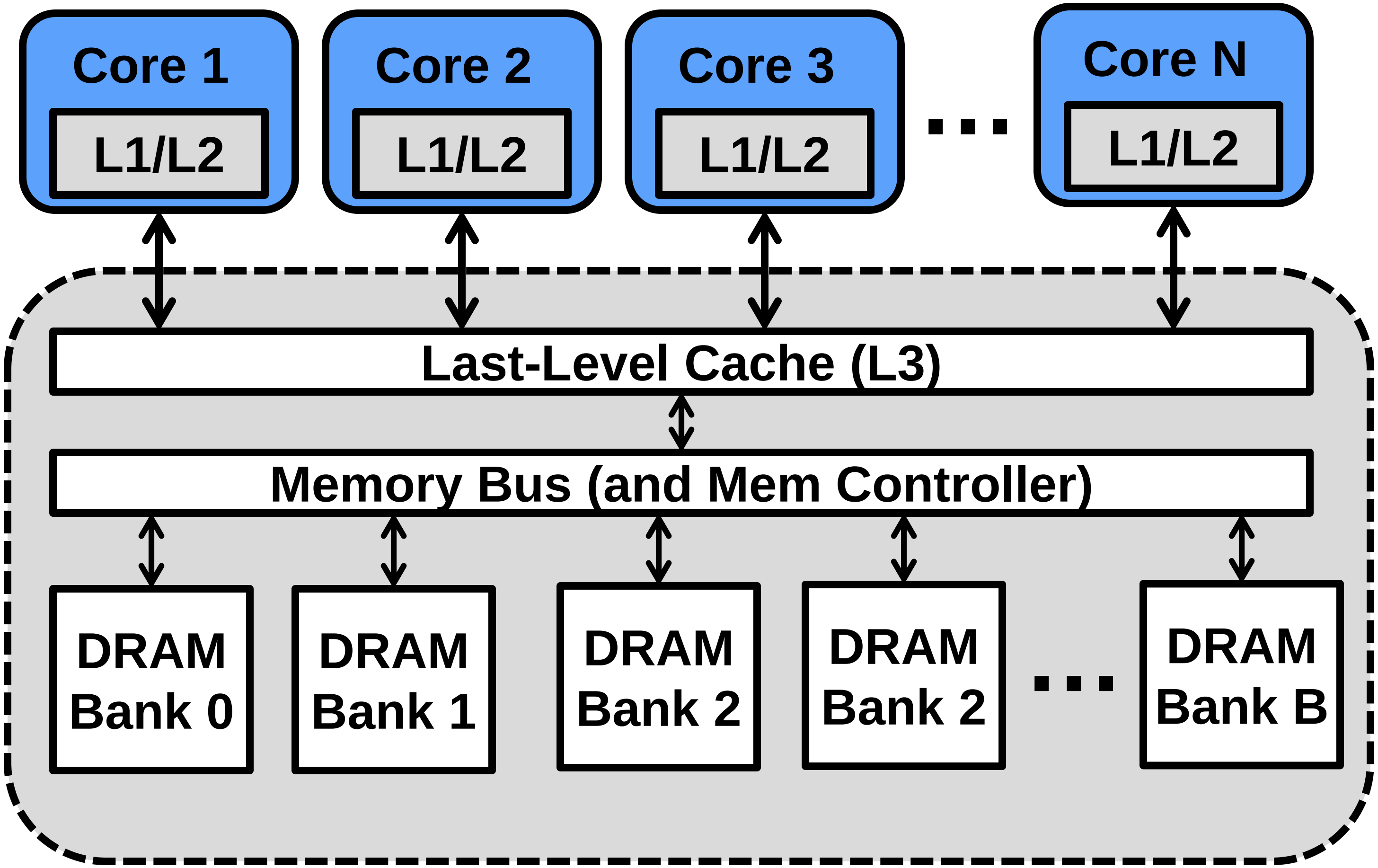
- Increasing overall performance by integrating multiple cores

Embedded systems: Actively adopting multi-core CPUs

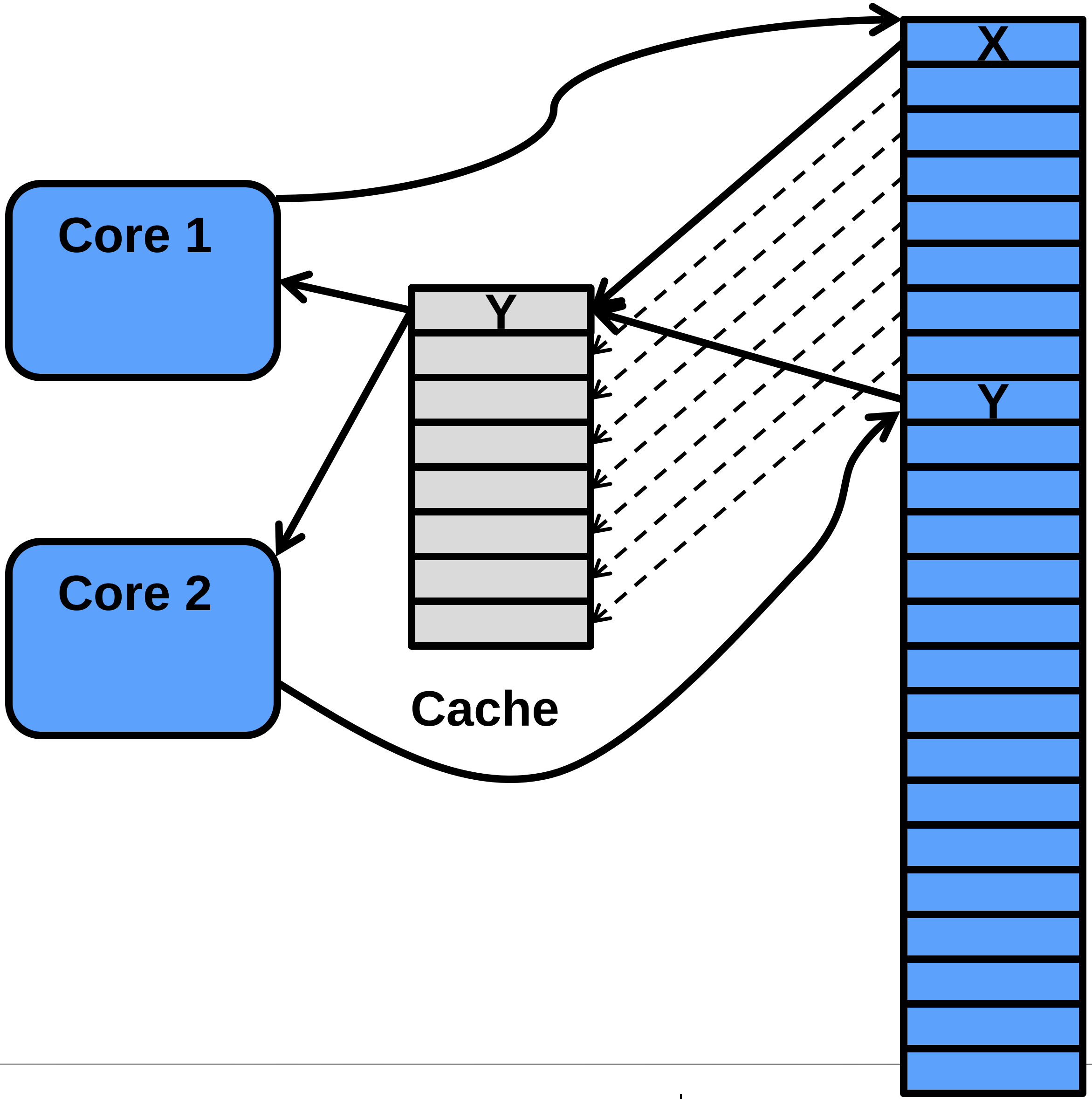
- **Automotive:**
  - Freescale i.MX6 4-core CPU
  - NVIDIA Tegra K1 platform

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

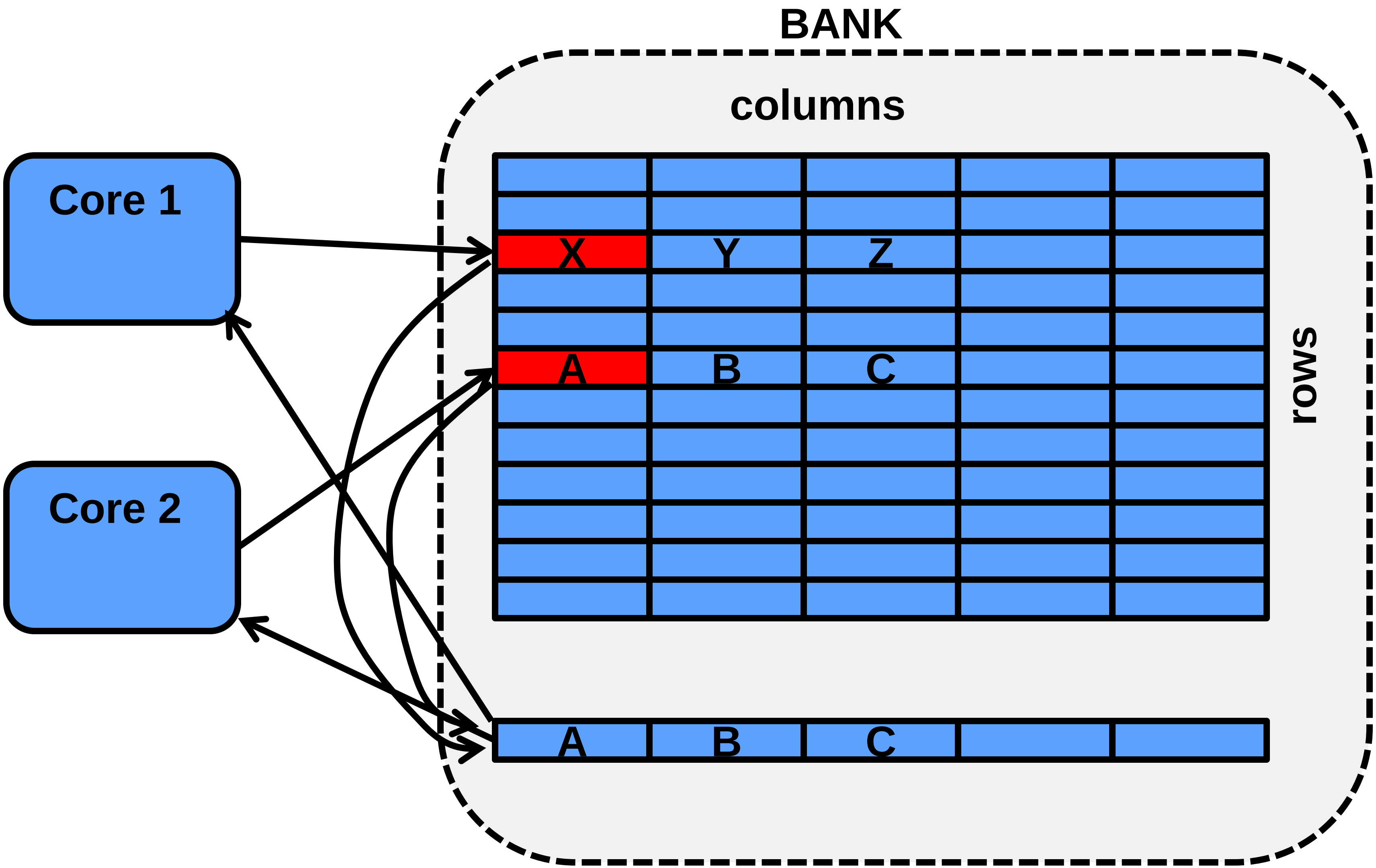
Shared Hardware: Multicore Memory System



Cache Interference Across Cores



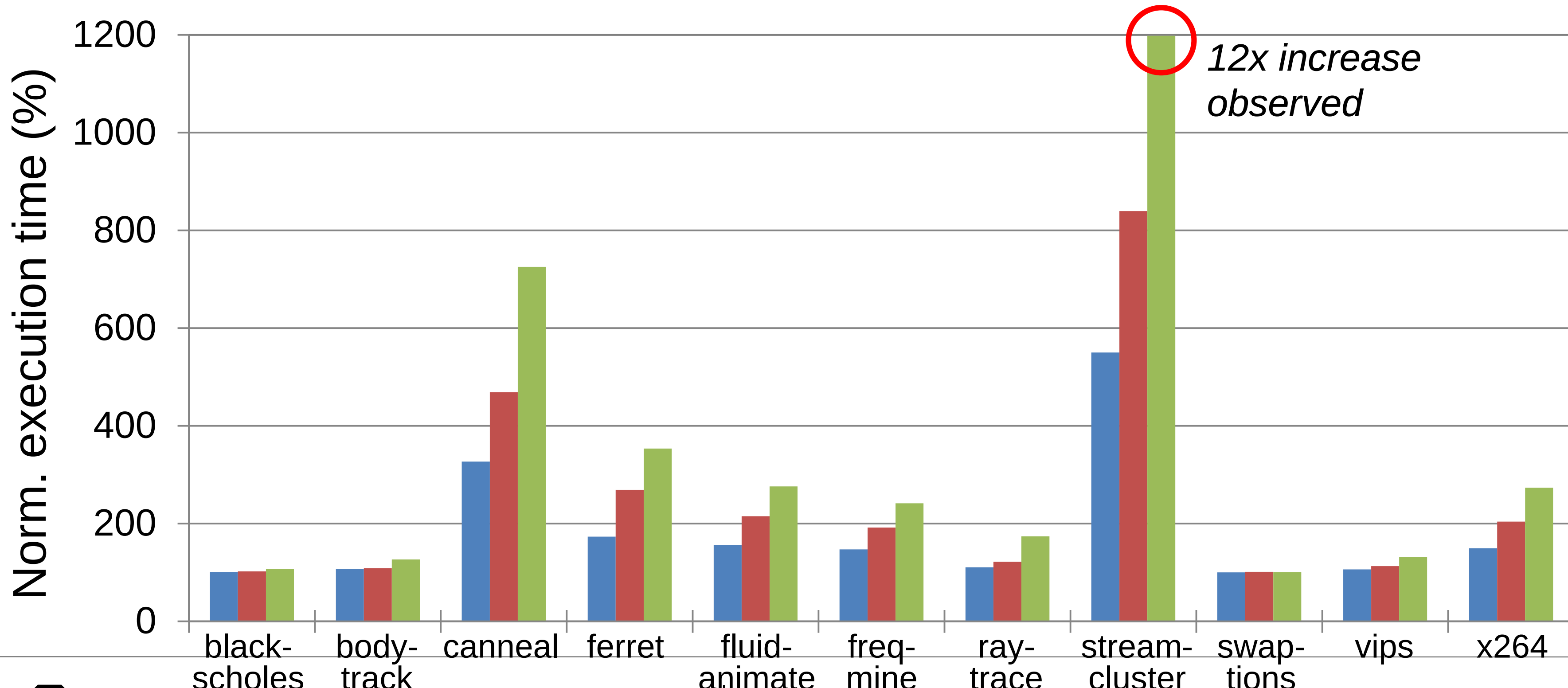
Bank Interference Across Cores



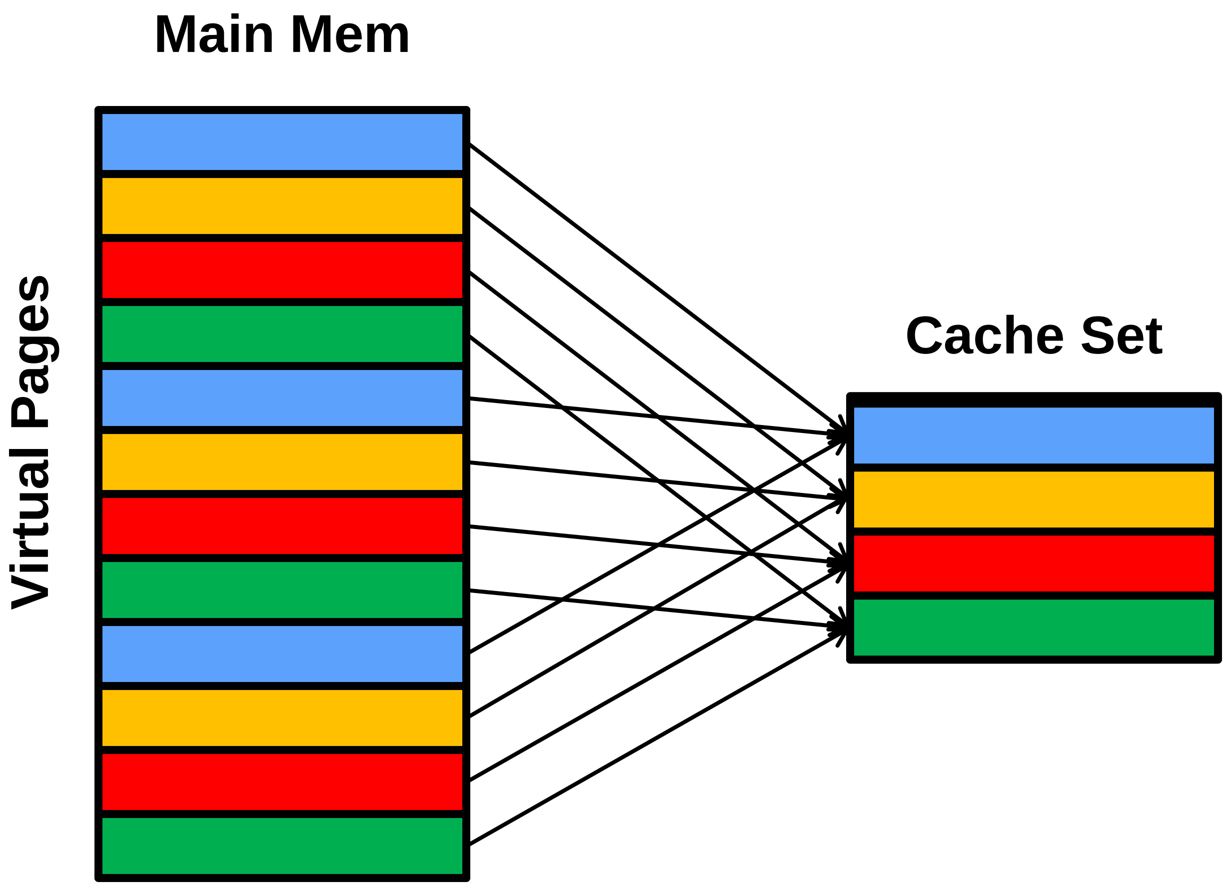
Impact of Memory Interference

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

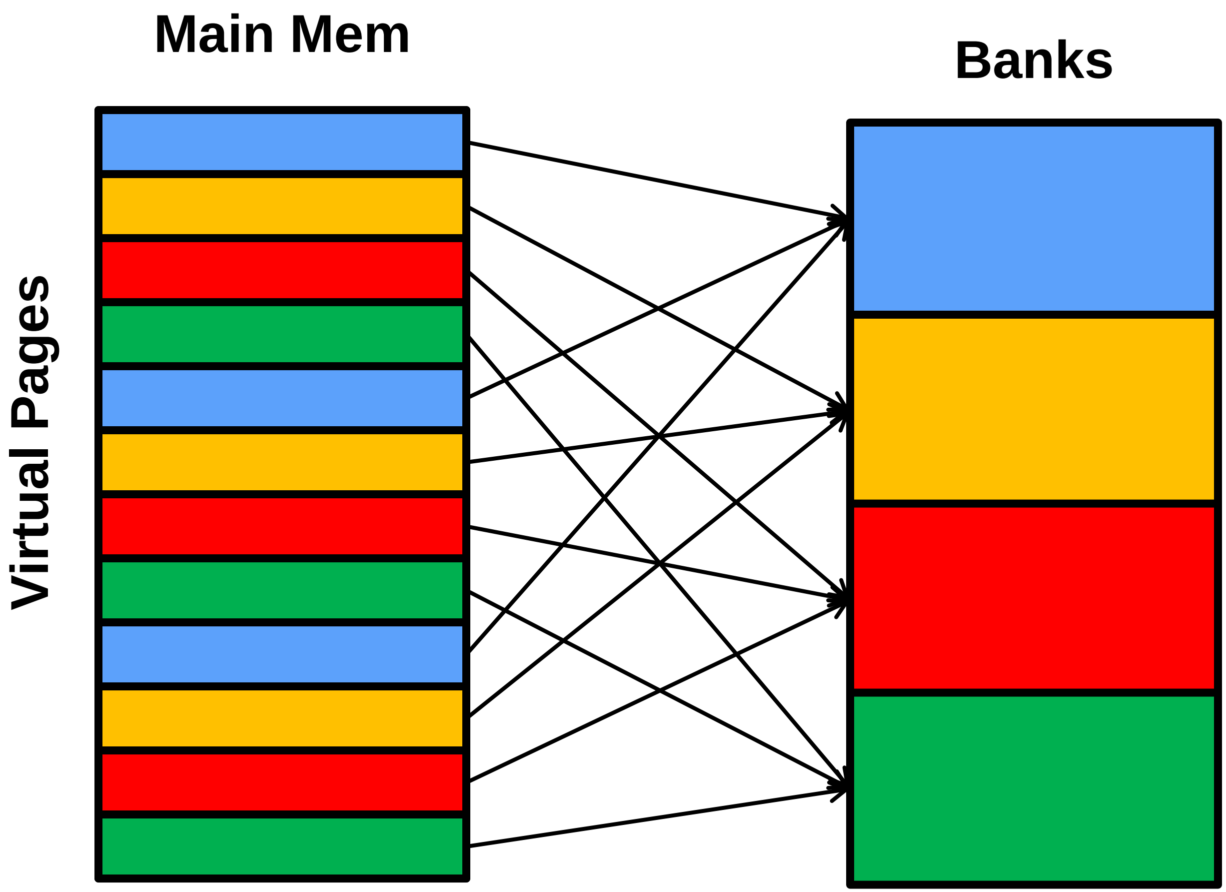
We should predict, bound and reduce the memory interference delay!



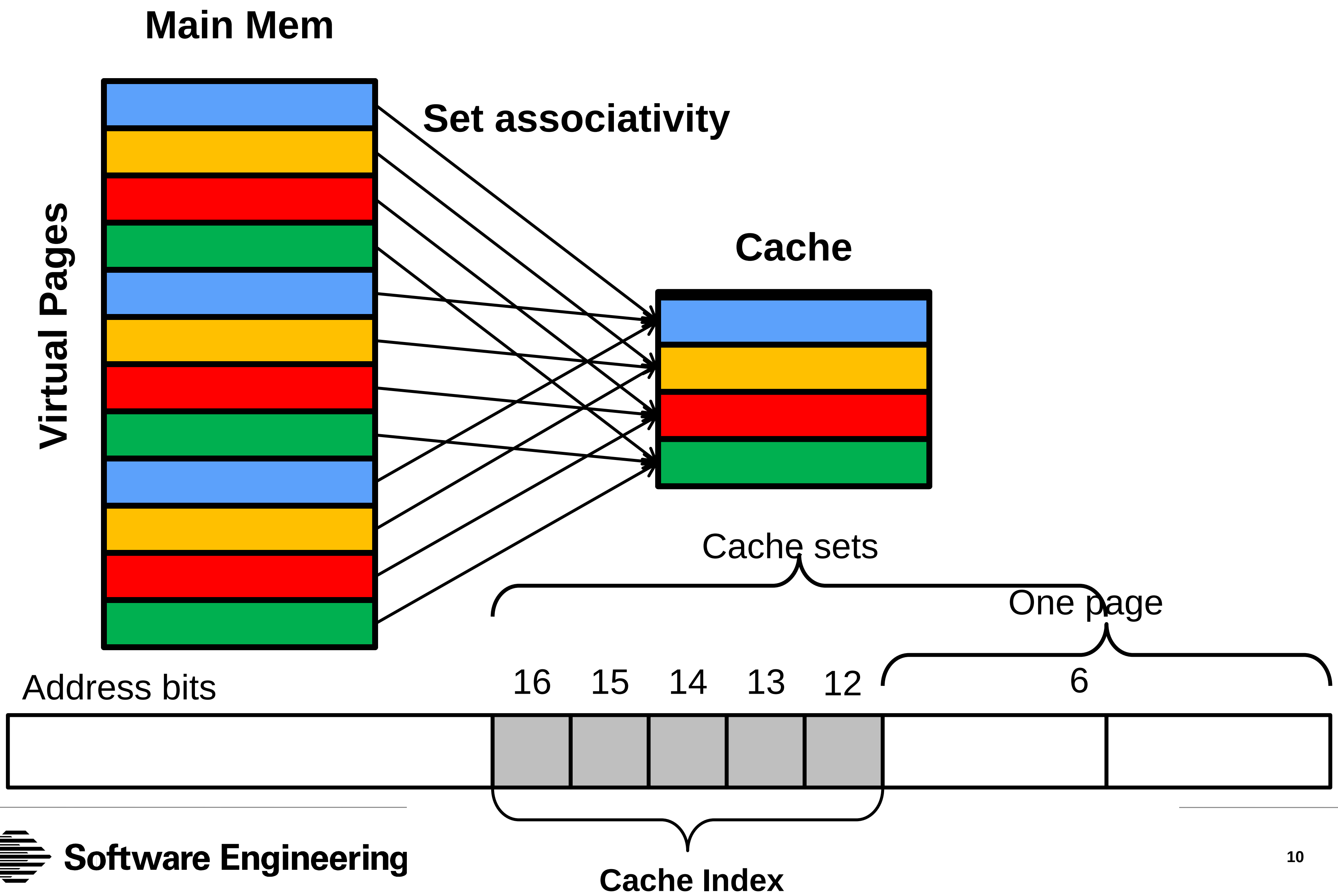
Cache / Bank Partitioning (Coloring)



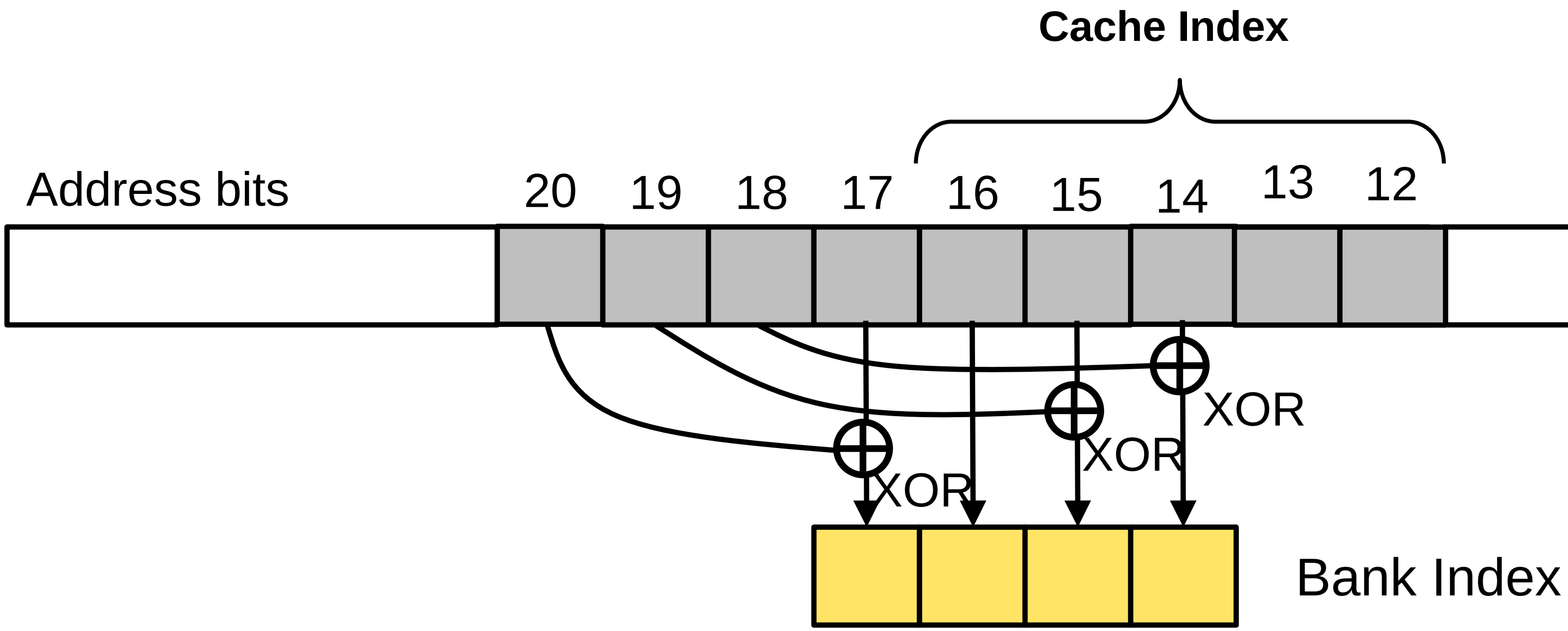
Cache / Bank Partitioning (Coloring)



Cache / Bank Partitioning (Coloring)



Cache and Bank Address Bits



E.g. 2 bank bits
2 cache bits
1 shared bit

Cache \ Bank	00	01	10	11
00	X		X	
01	X		X	
10		X		X
11		X		X

Coordinated Cache and Bank Partitioning (Private Partitions)

Avoid conflicting color assignments

Take advantage of different conflict behaviors

- Banks can be shared within same core but not across cores
- Cache cannot be shared within or across cores

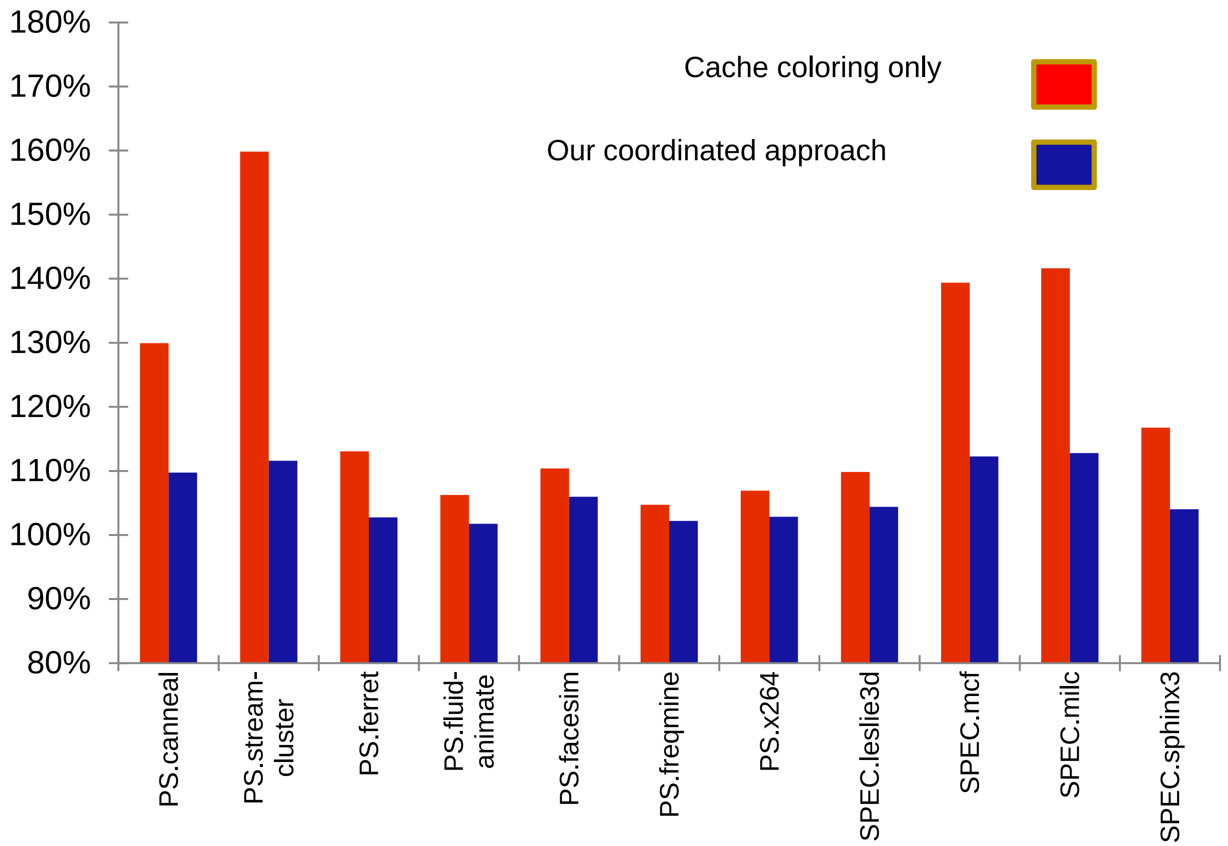
Take advantage of sensitivity of execution time to cache

- Task with highest sensitivity to cache is assigned more cache
- Diminishing returns taken into account

Two algorithms explored

- Mixed-Integer Linear Programming
- Knapsack

Experimental Results



Shared Bank Partitioning

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

Task's own memory requests (request-driven)

Interfering memory requests during the job execution (job-driven)

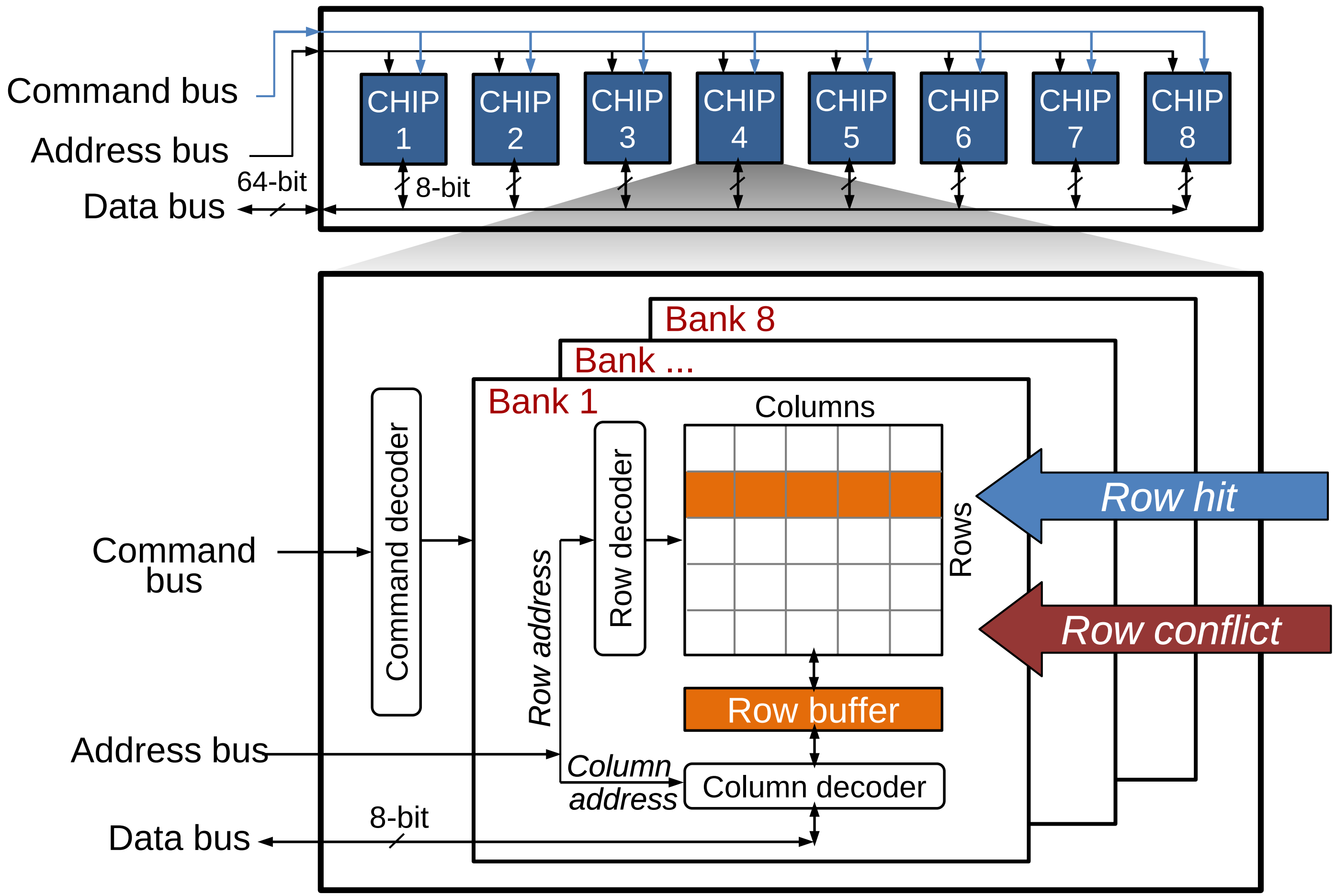
Software DRAM bank partitioning awareness

- Analyzes the effect of dedicated and shared DRAM banks

DRAM Organization

DRAM Rank

DRAM Chip



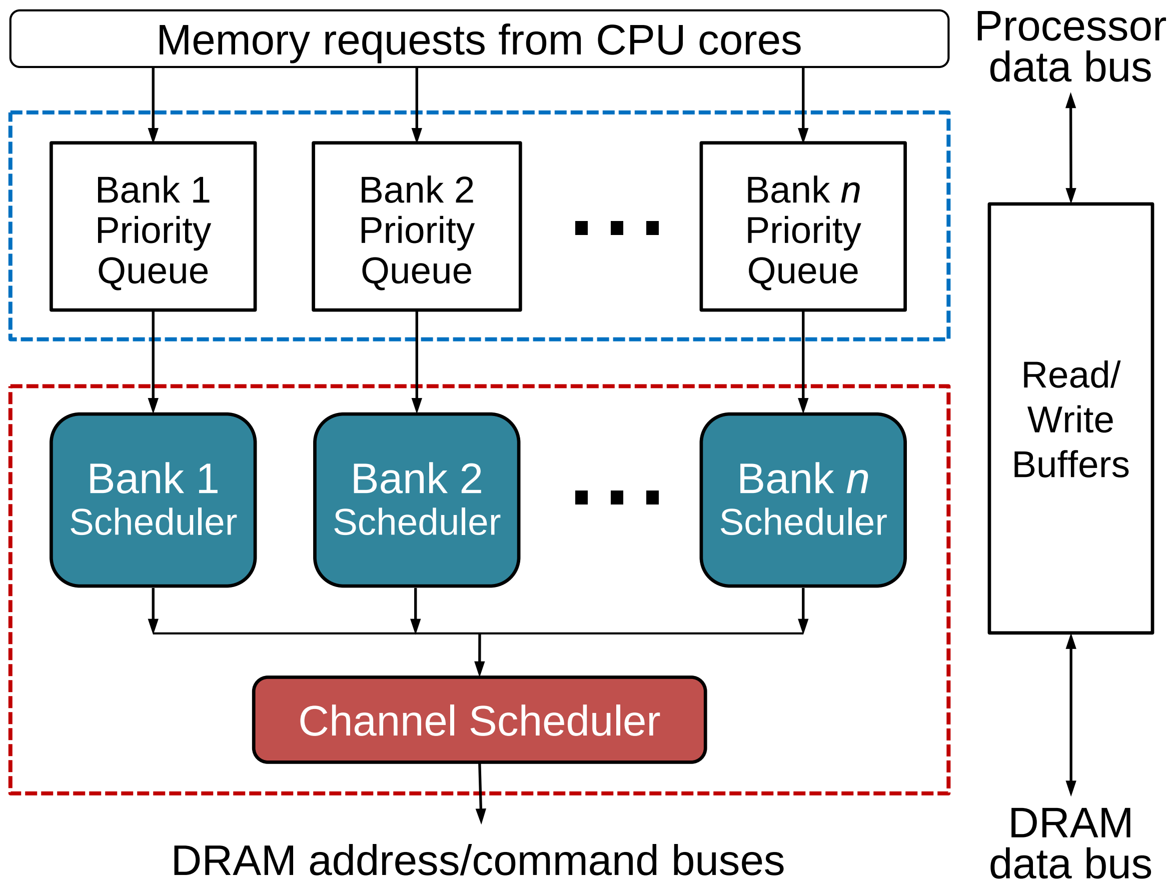
DRAM access latency varies depending on which row is stored in the row buffer

Memory Controller

Request buffer

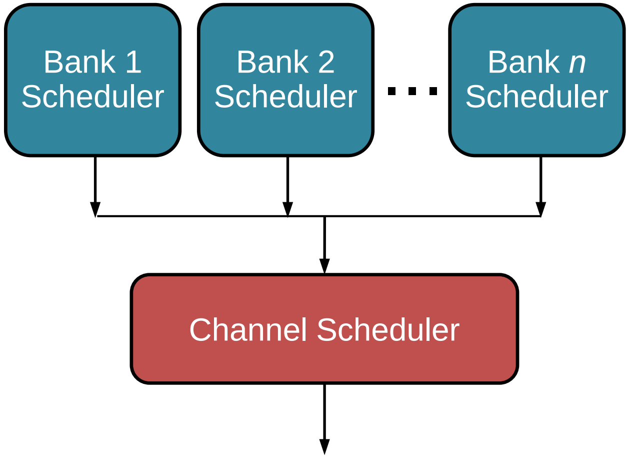
Memory scheduler

Two-level hierarchical scheduling structure



Memory Scheduling Policy

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



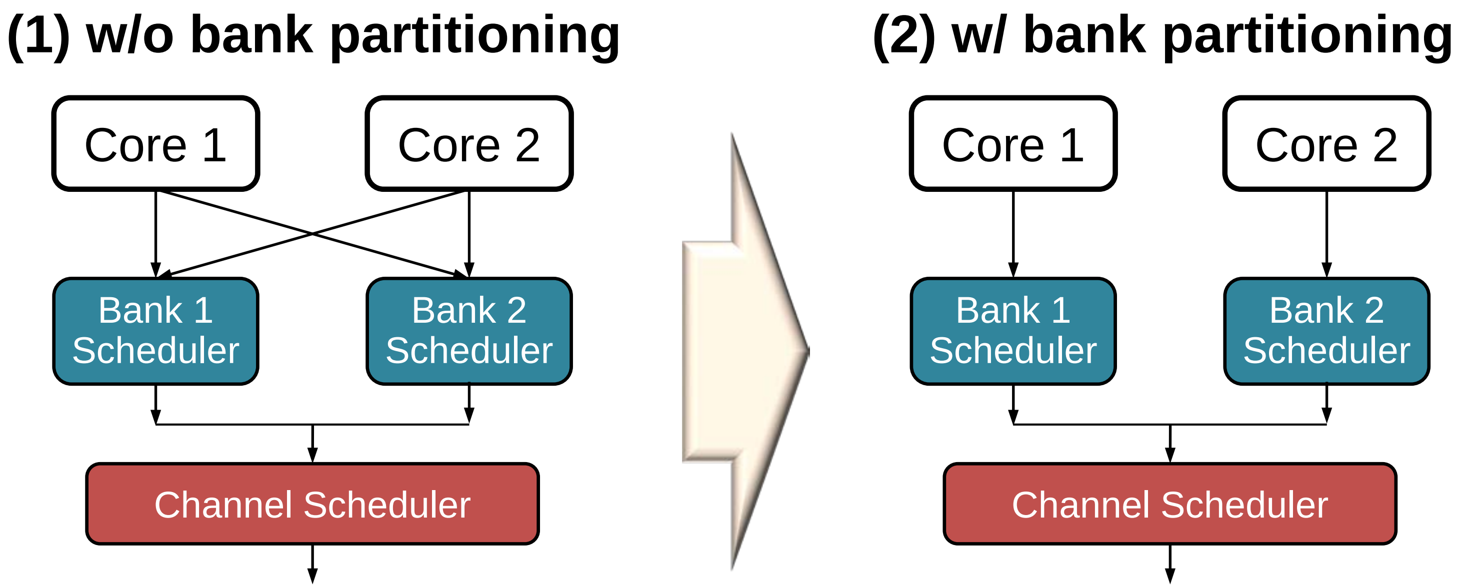
1. **Bank scheduler**
   - Considers bank timing constraints
   - Prioritizes row-hit requests
   - In case of tie, prioritizes older requests
2. **Channel scheduler**
   - Considers channel timing constraints
   - Prioritizes older requests

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

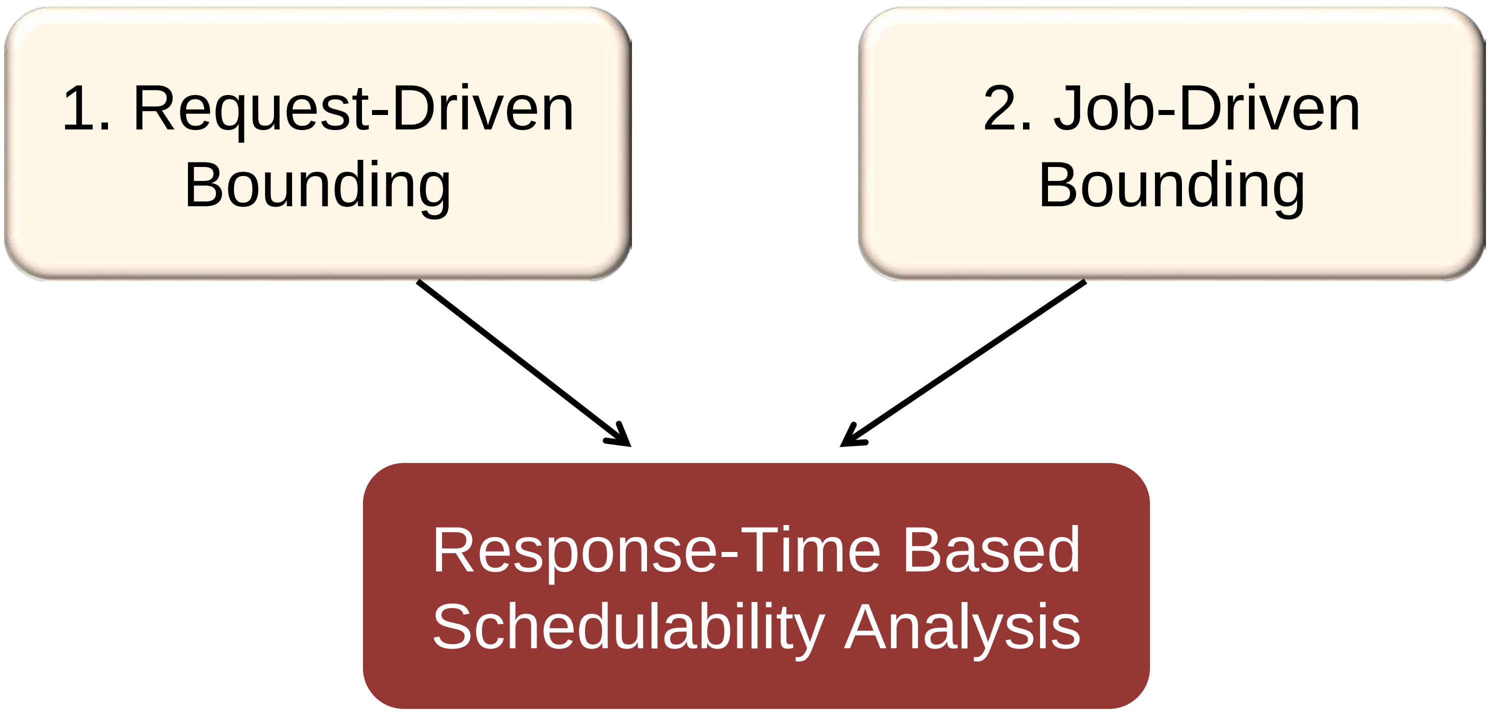
- **Prevents intra-bank interference** by dedicating different DRAM banks to each core
  - Can be supported in the OS kernel



Intra-bank and inter-bank interference

Only inter-bank interference

Bounding Memory Interference Delay



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

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

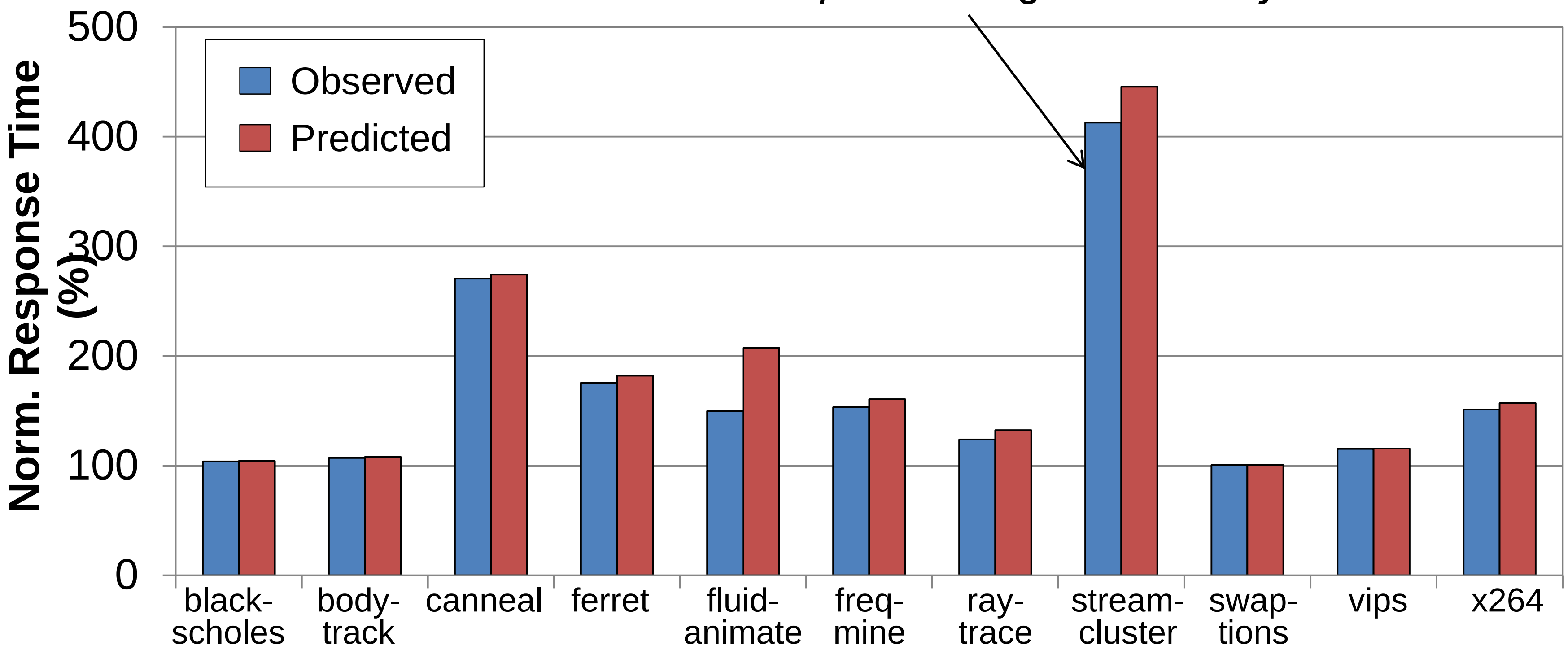
Request-Driven (RD) Approach

Job-Driven (JD) Approach

Experiment Severe Memory Interference

- **Private DRAM Bank**

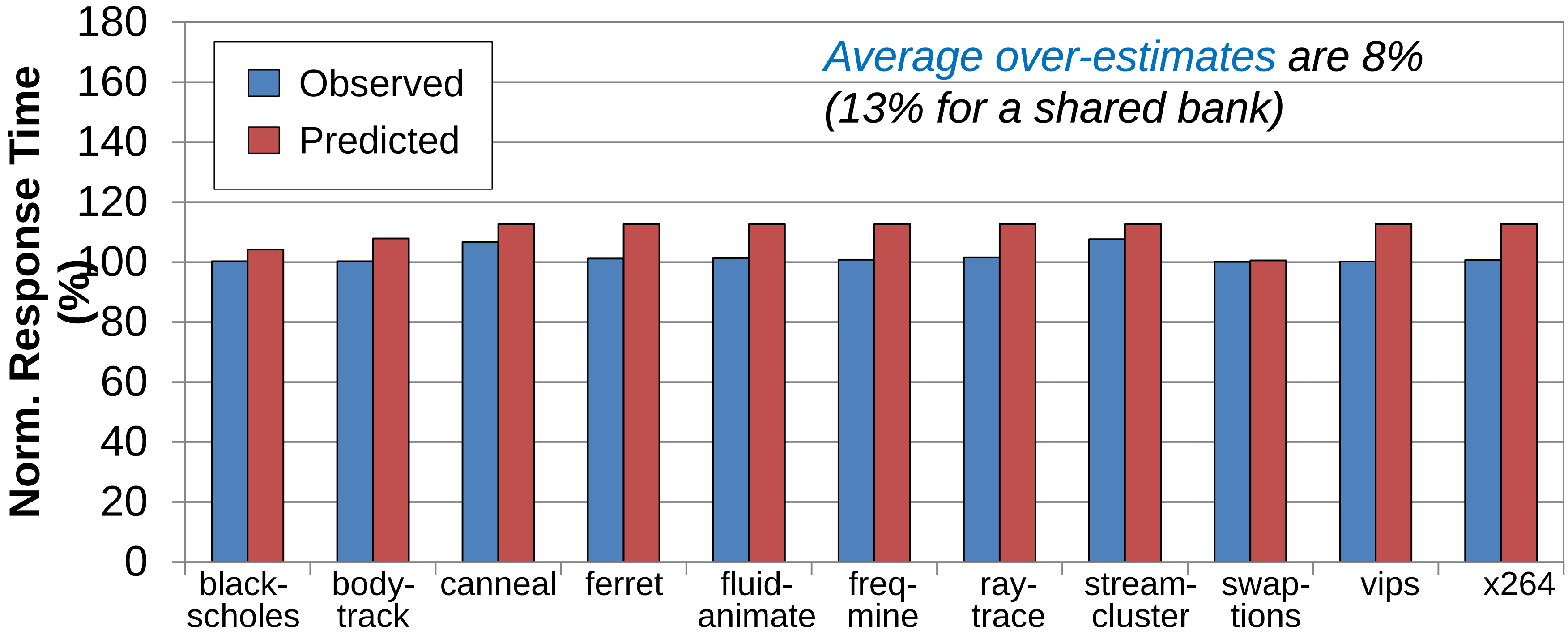
4.1x increase → DRAM bank partitioning helps reducing the memory interference



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

Implementation

Cache and Bank Partitioning implemented in Linux/RK

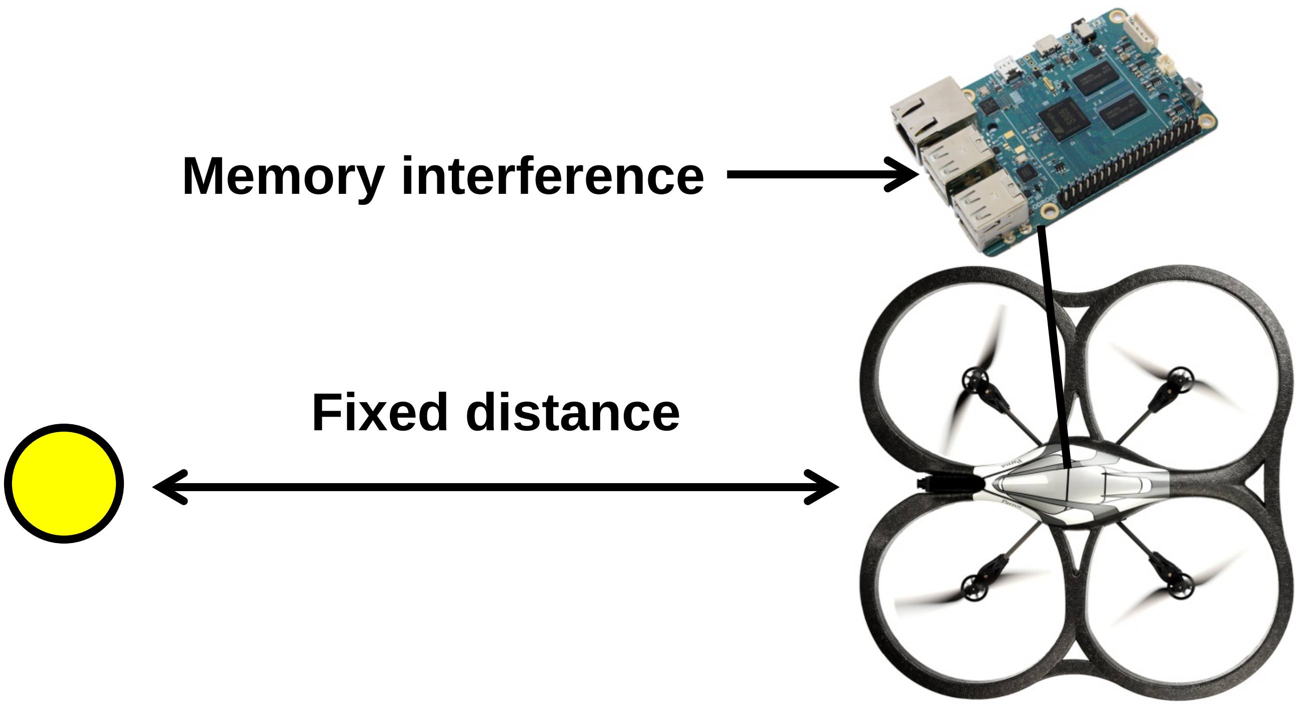
- Associates Resource Reservations to Linux Threads
  - Memory reservation
  - Cache reservation
  - CPU reservation
  - …

“Portable” Kernel Module

- Hooks into on-demand page allocation
- At boot time create large memory reserve
  - Pages are classified in cache and bank colors

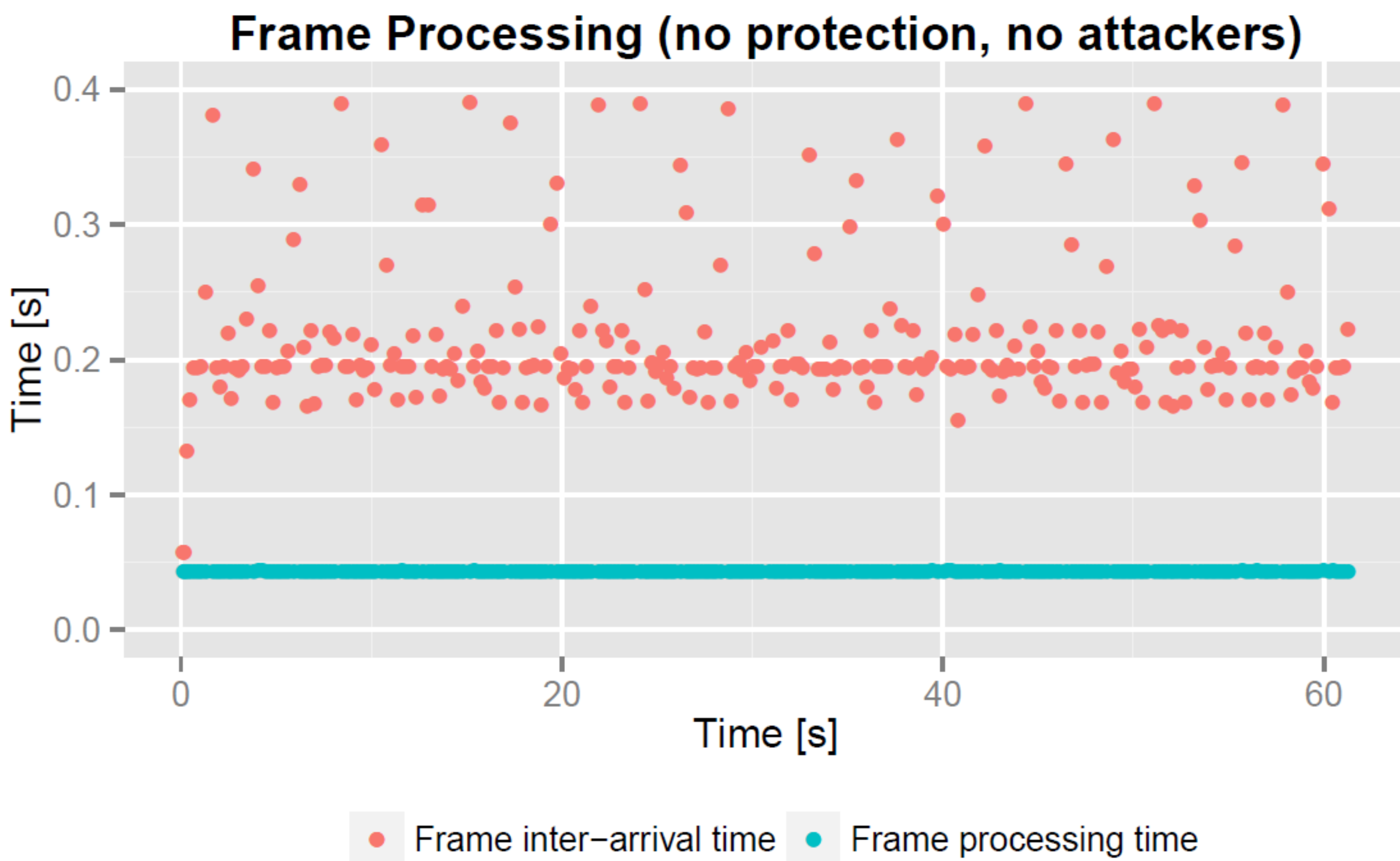
Model Problem

Ball-following Controller on Odroid+Linux/RK

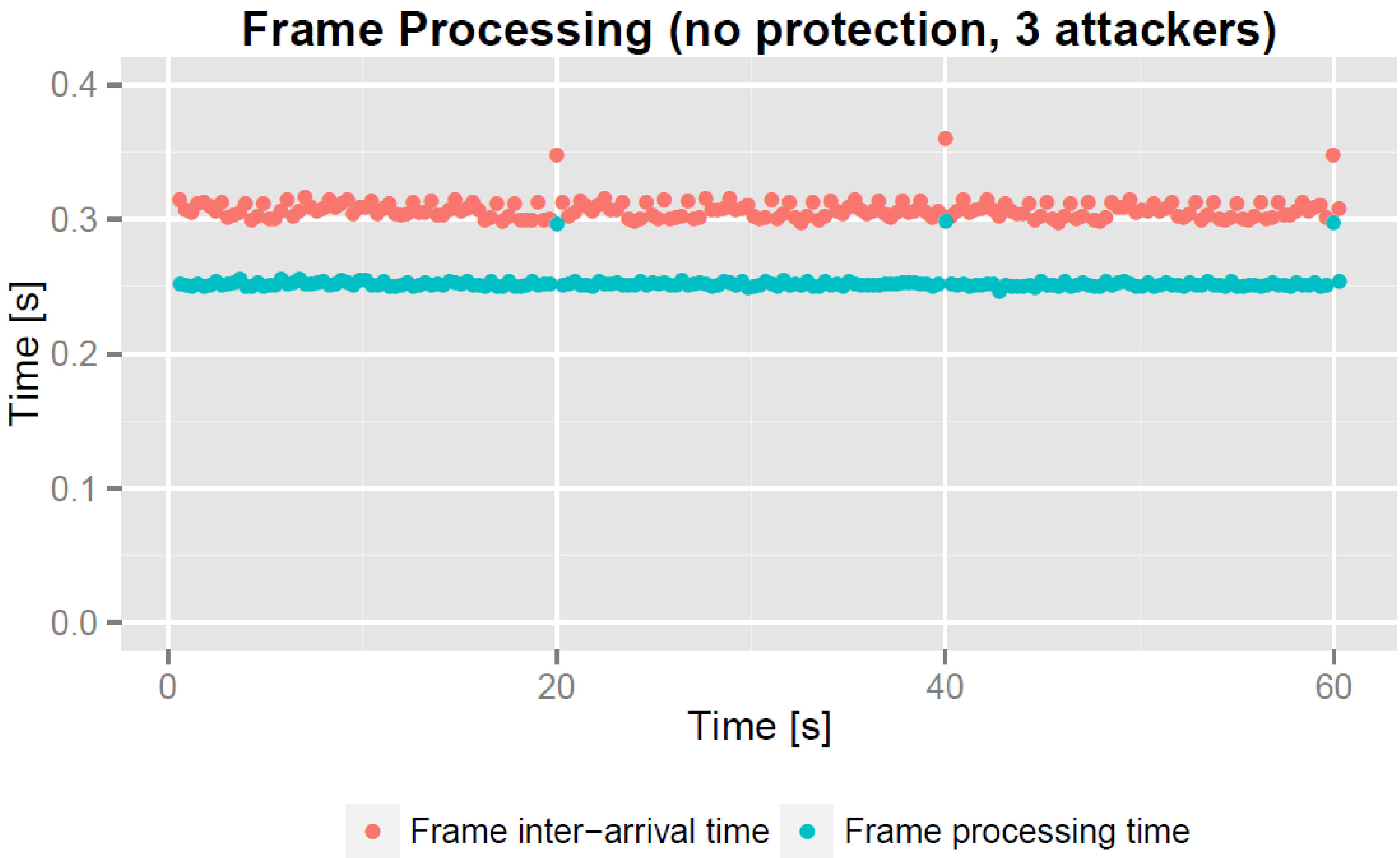


Ball-Following: Keep fixed distance as ball moves around

Experimental Results (1)

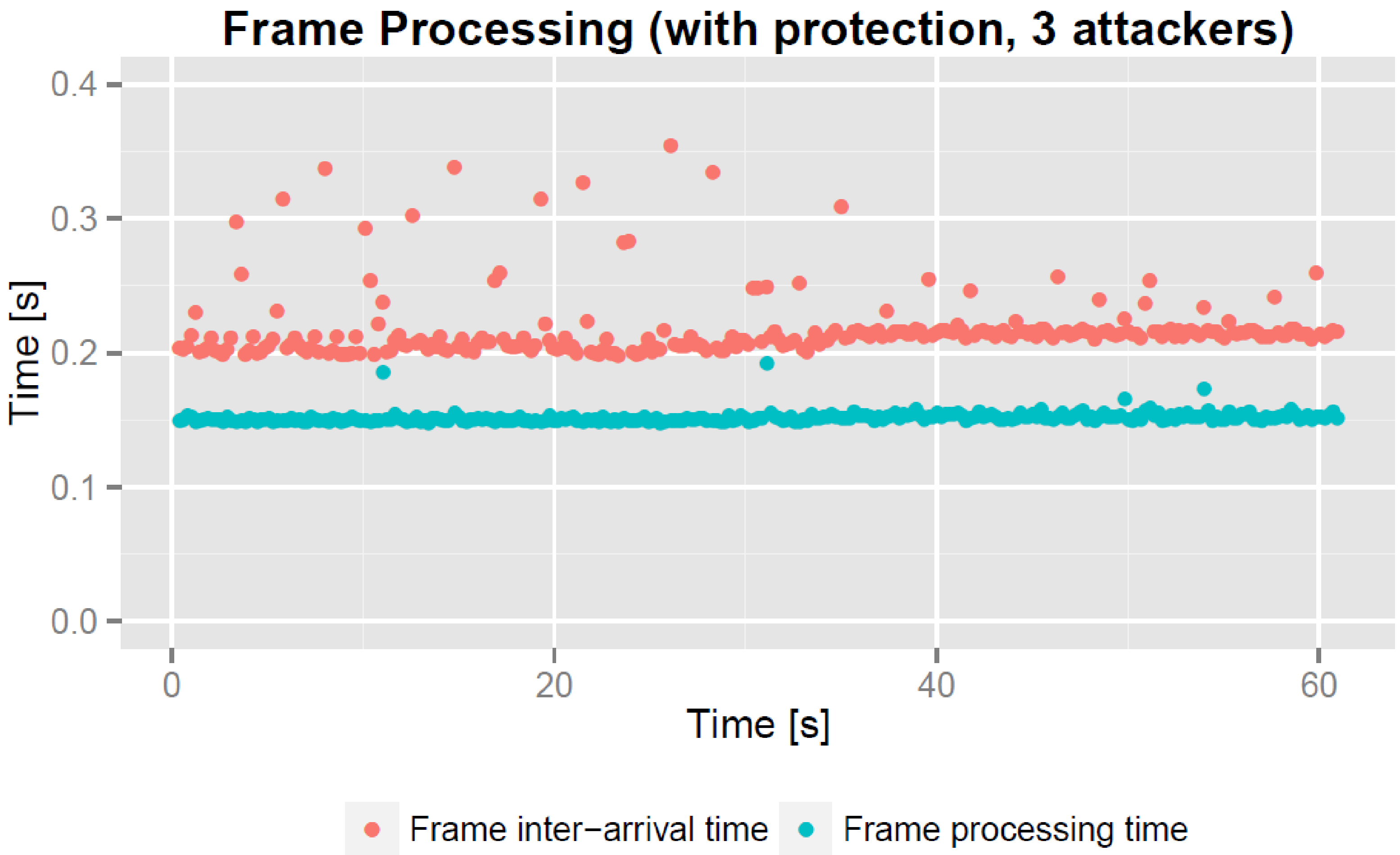


Experimental Results (2)



Deadlines misses at 0.2. only 195 out of 279 frames processed (30% loss)

Experimental Results (3)



No deadline misses. Processing below 0.2 s interarrival (period)

Concluding Remarks

Multicore processor challenges previous results in real-time systems

- Interference from shared hardware
  - Cache, Memory banks, Memory bus

Leads to less usable processing capacity

- 1200% increase in a four core machine (92% reduction from single core)

Our approach

- Coordinated private partitions for cache and memory
- Shared bank partitions
- Implemented in Linux/RK

Experimental results for model avionics application

- Protects control algorithm from interference