

# COTS Multicore Processors in Avionics Systems: Challenges and Solutions

Dionisio de Niz

Bjorn Andersson and Lutz Wrage

[dionisio@sei.cmu.edu](mailto:dionisio@sei.cmu.edu), [baandersson@sei.cmu.edu](mailto:baandersson@sei.cmu.edu),  
[lwrage@sei.cmu.edu](mailto:lwrage@sei.cmu.edu)



# Report Documentation Page

*Form Approved  
OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE <b>06 JAN 2015</b>	2. REPORT TYPE <b>N/A</b>	3. DATES COVERED	
4. TITLE AND SUBTITLE <b>COTS Multicore Processors in Avionics Systems: Challenges and Solutions</b>		5a. CONTRACT NUMBER	
		5b. GRANT NUMBER	
		5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) <b>Wrage /Dionisio de Niz Bjorn Andersson Lutz</b>		5d. PROJECT NUMBER	
		5e. TASK NUMBER	
		5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <b>Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213</b>		8. PERFORMING ORGANIZATION REPORT NUMBER	
		10. SPONSOR/MONITOR'S ACRONYM(S)	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
		12. DISTRIBUTION/AVAILABILITY STATEMENT <b>Approved for public release, distribution unlimited.</b>	
13. SUPPLEMENTARY NOTES <b>The original document contains color images.</b>			
14. ABSTRACT			
15. SUBJECT TERMS			
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT
a. REPORT <b>unclassified</b>	b. ABSTRACT <b>unclassified</b>	c. THIS PAGE <b>unclassified</b>	<b>SAR</b>
			18. NUMBER OF PAGES <b>28</b>
			19a. NAME OF RESPONSIBLE PERSON

Copyright 2015 Carnegie Mellon University

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](mailto:permission@sei.cmu.edu).

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

DM-0002049



# 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

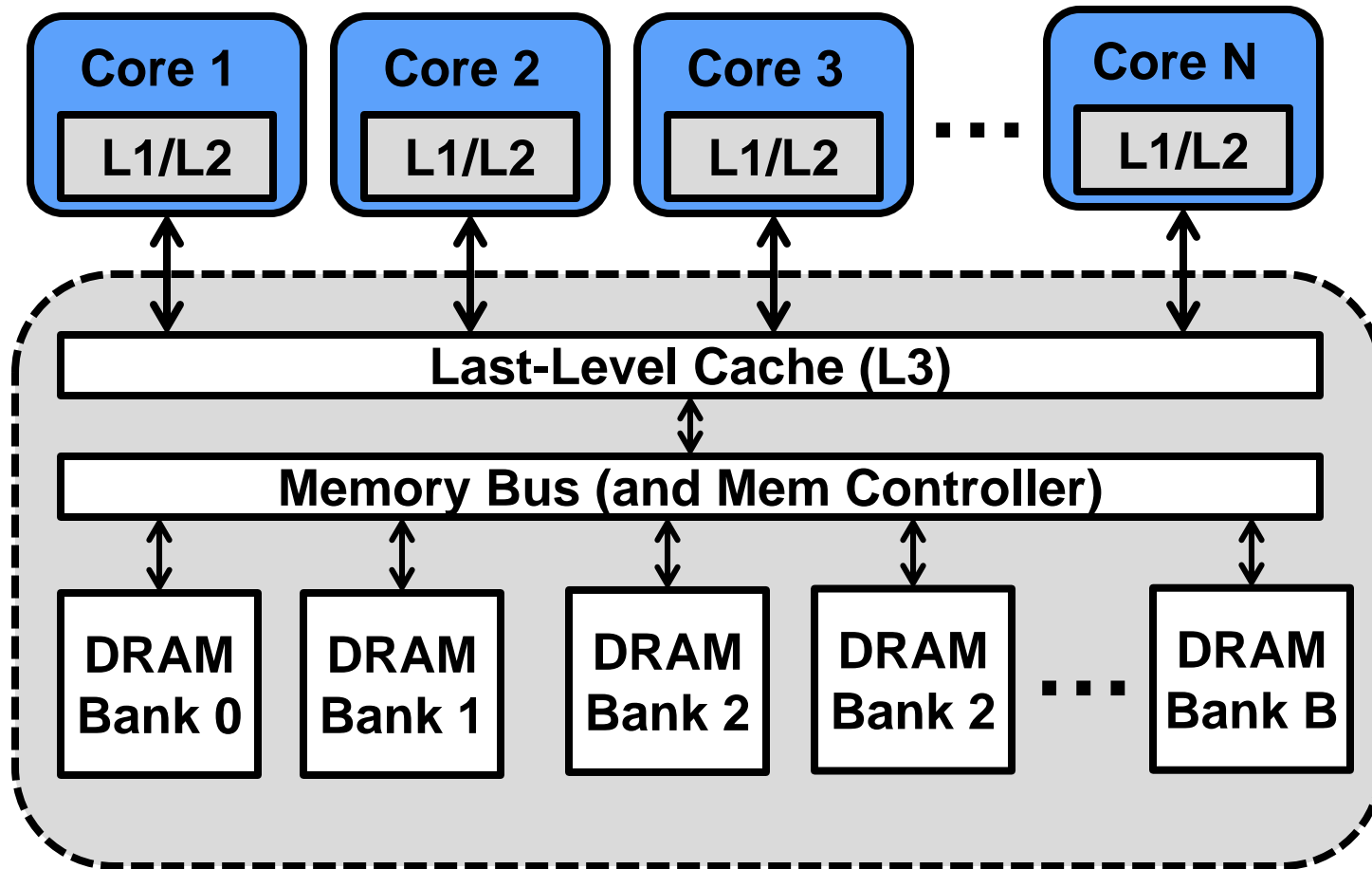


- **Avionics and defense:**

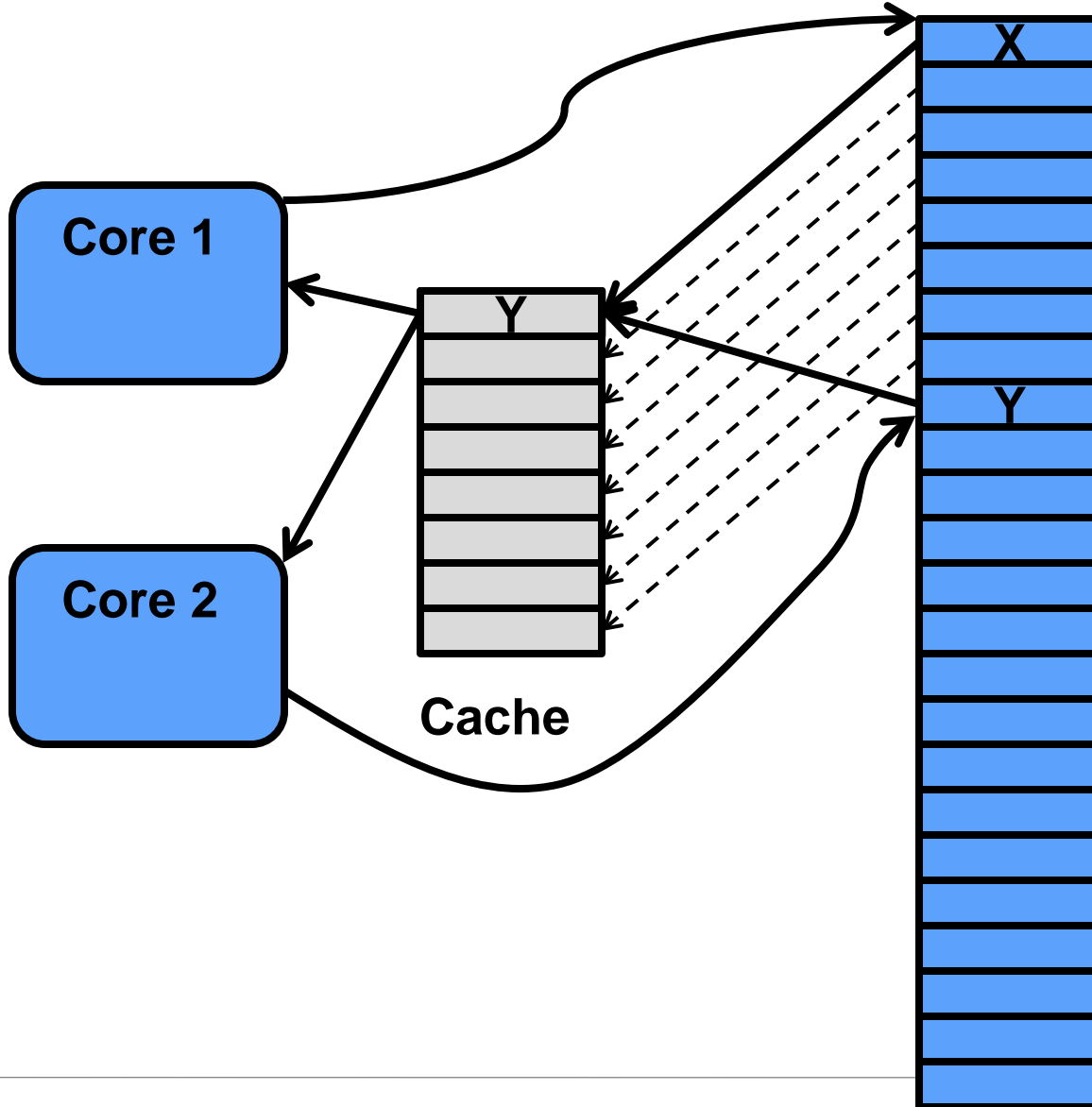
- Rugged Intel i7 single board computers
- Freescale P4080 8-core CPU



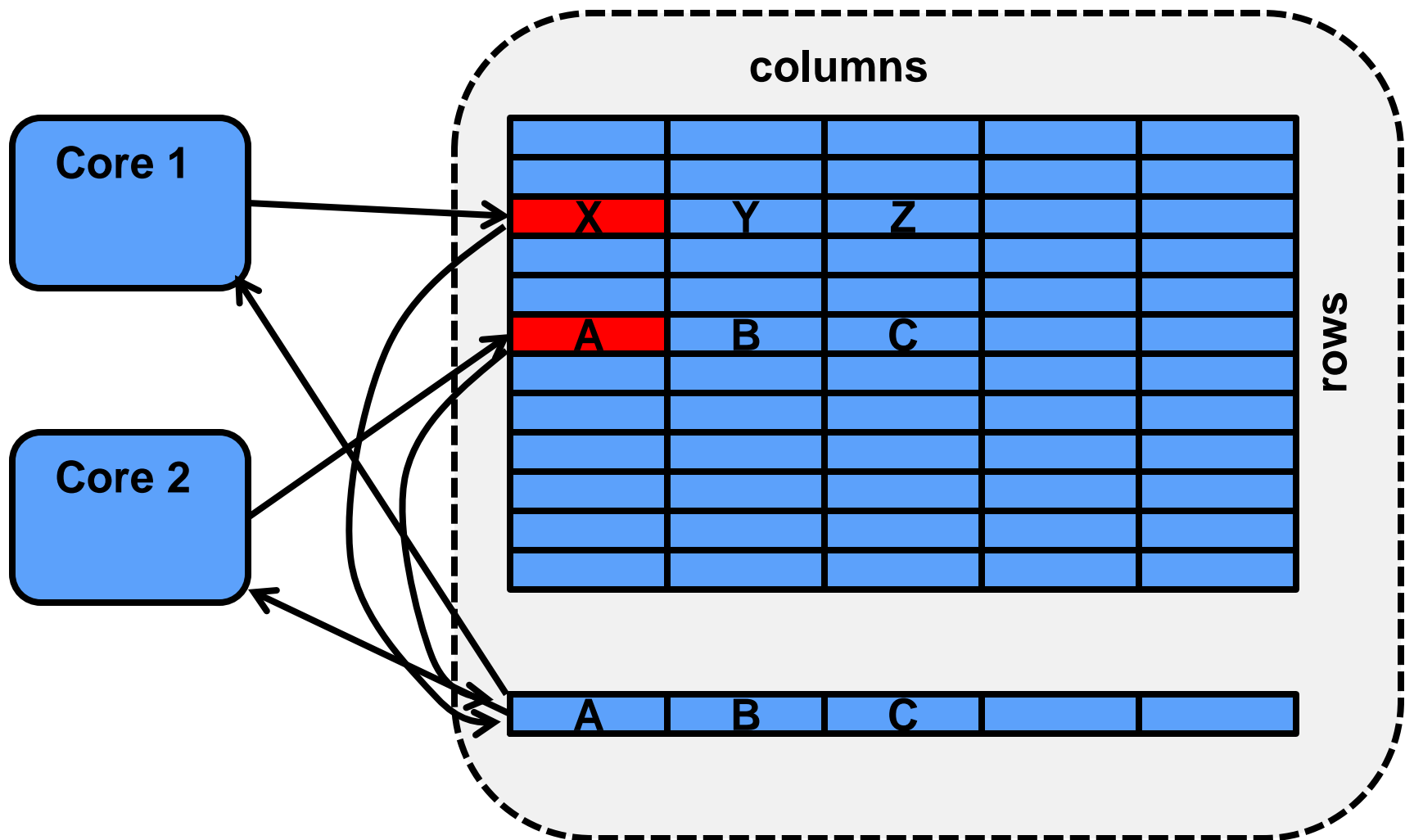
# Shared Hardware: Multicore Memory System



# Cache Interference Across Cores



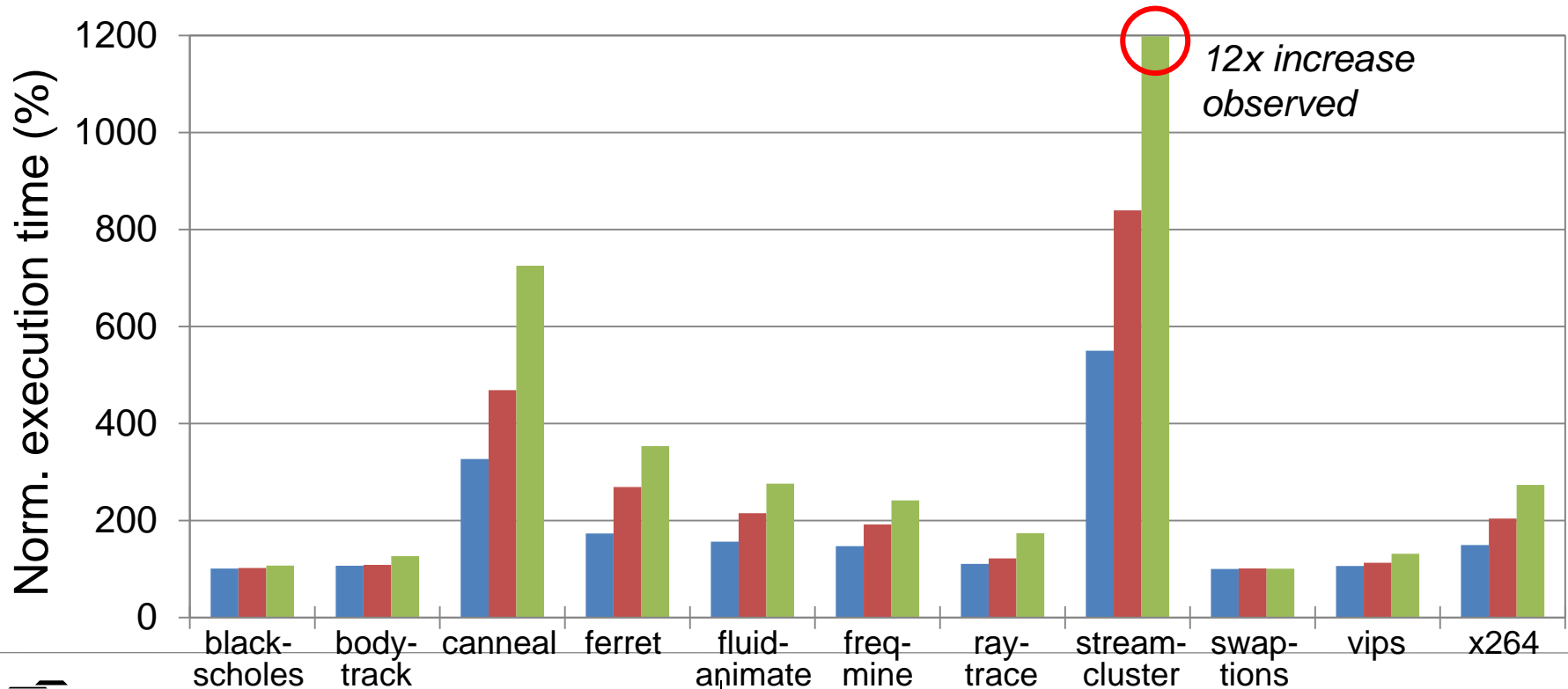
# Bank Interference Across Cores



# Impact of Memory Interference

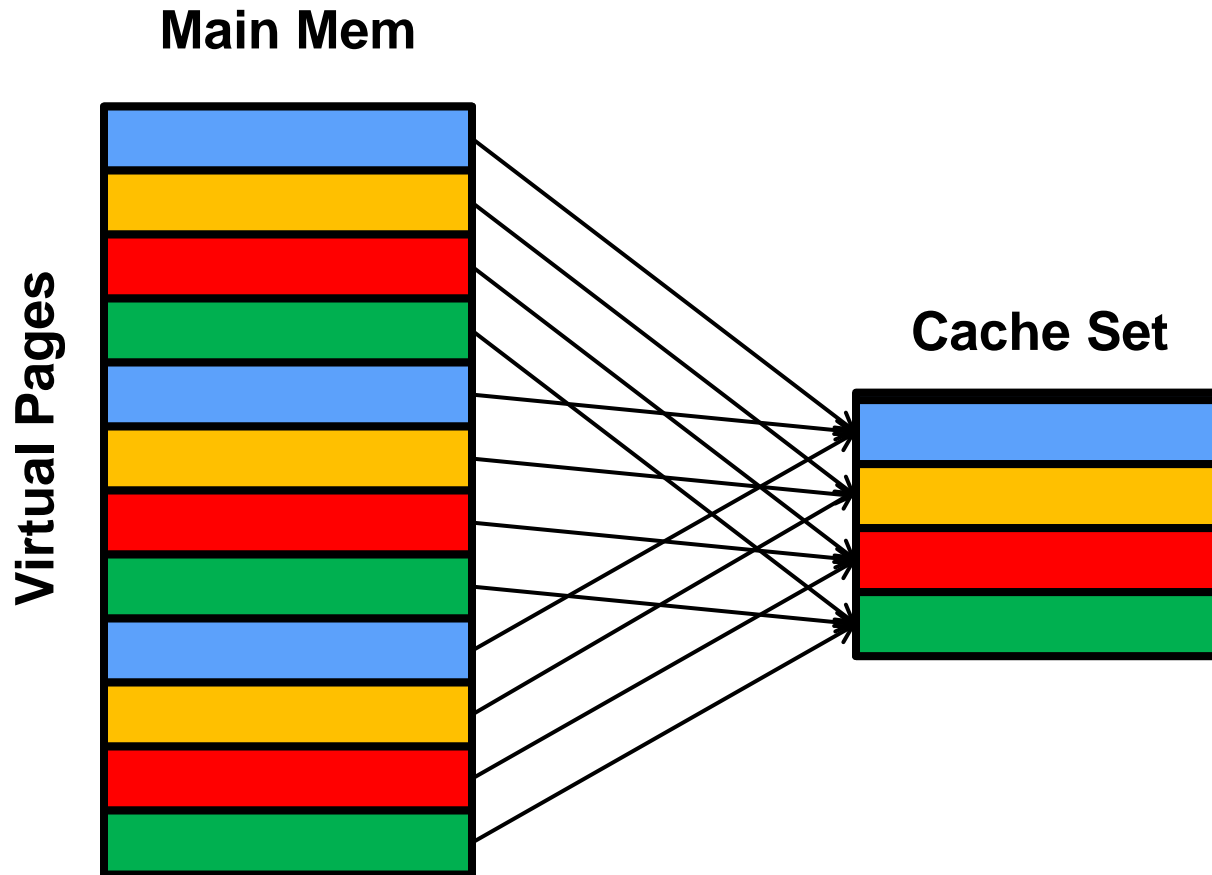
- 1 attacker → Max **5.5x** increase
- 2 attackers → Max **8.4x** increase
- 3 attackers → Max **12x** increase

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

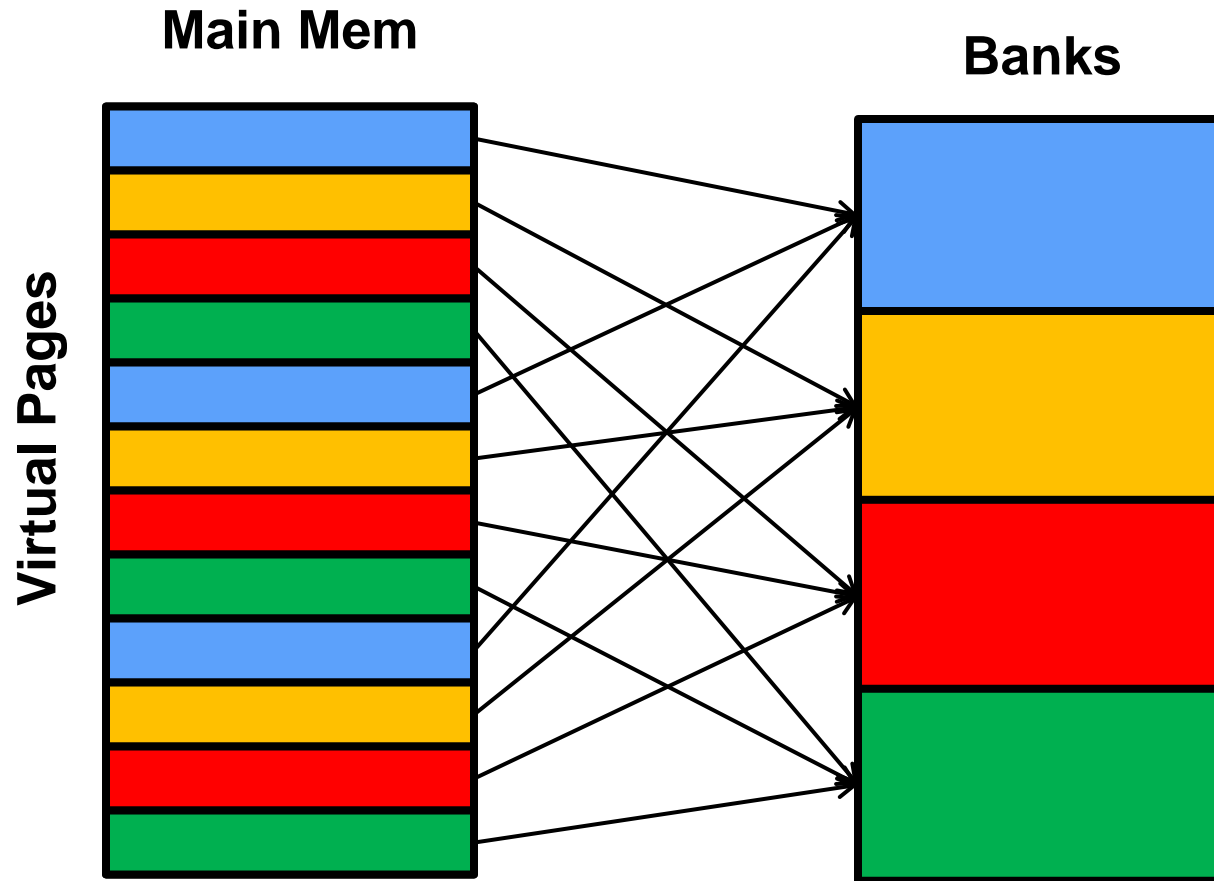




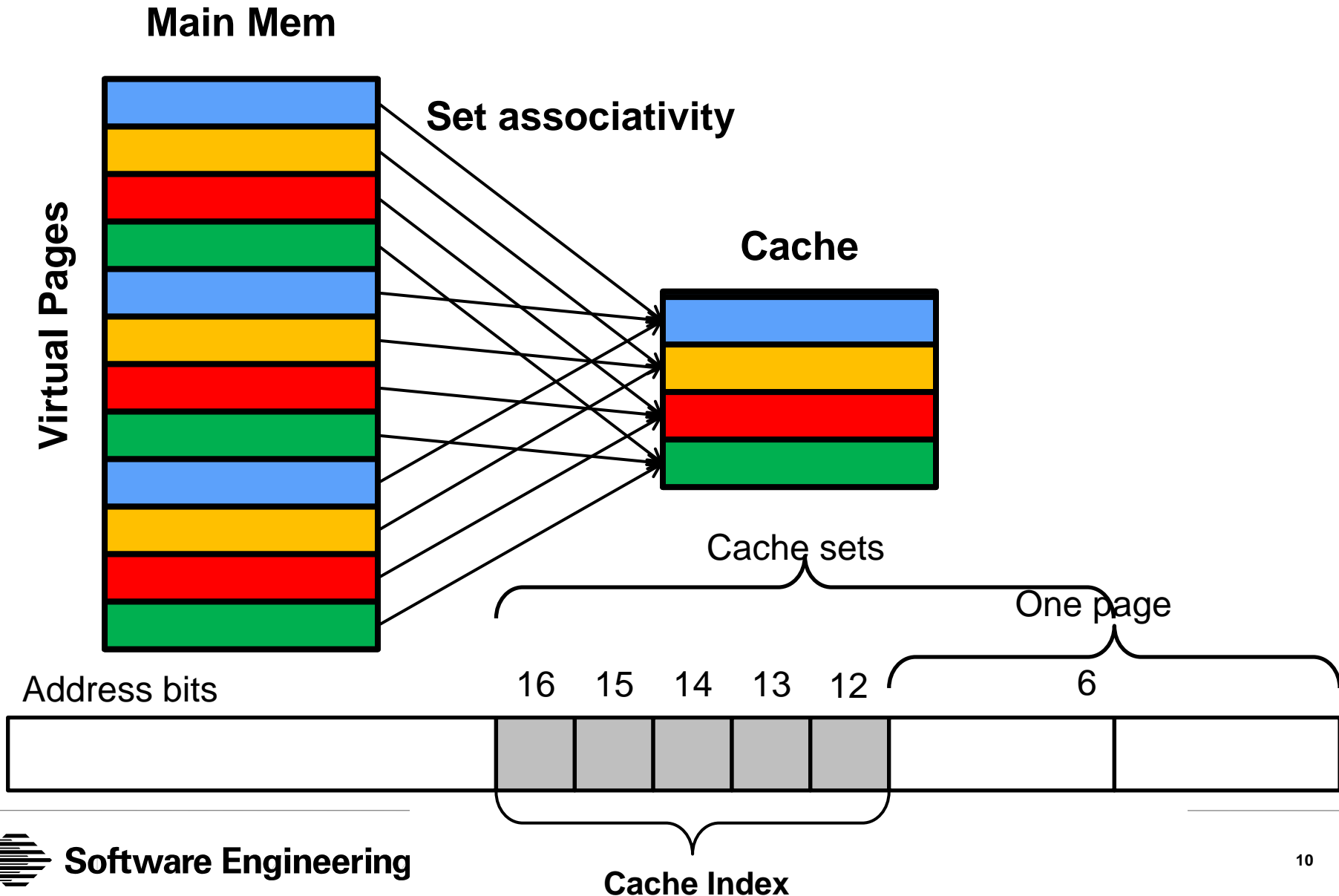
# Cache / Bank Partitioning (Coloring)



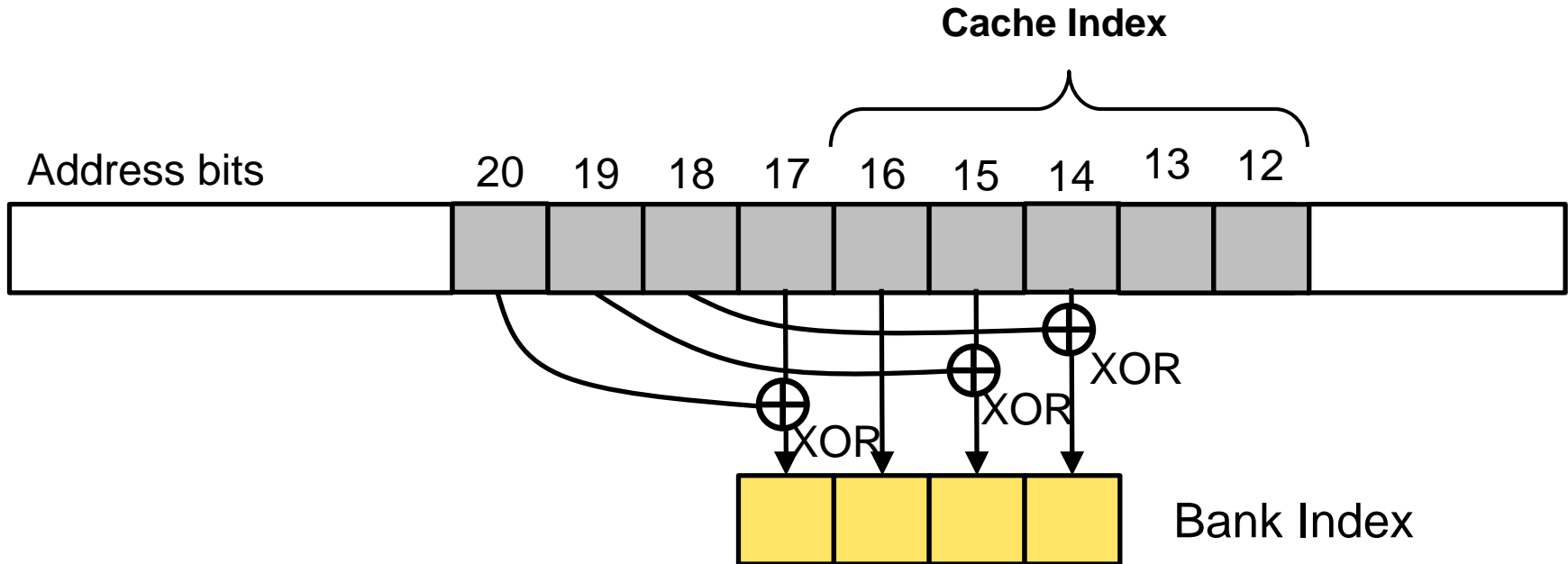
# Cache / Bank Partitioning (Coloring)



# Cache / Bank Partitioning (Coloring)



# Cache and Bank Address Bits



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

		Bank			
		00	01	10	11
Cache	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

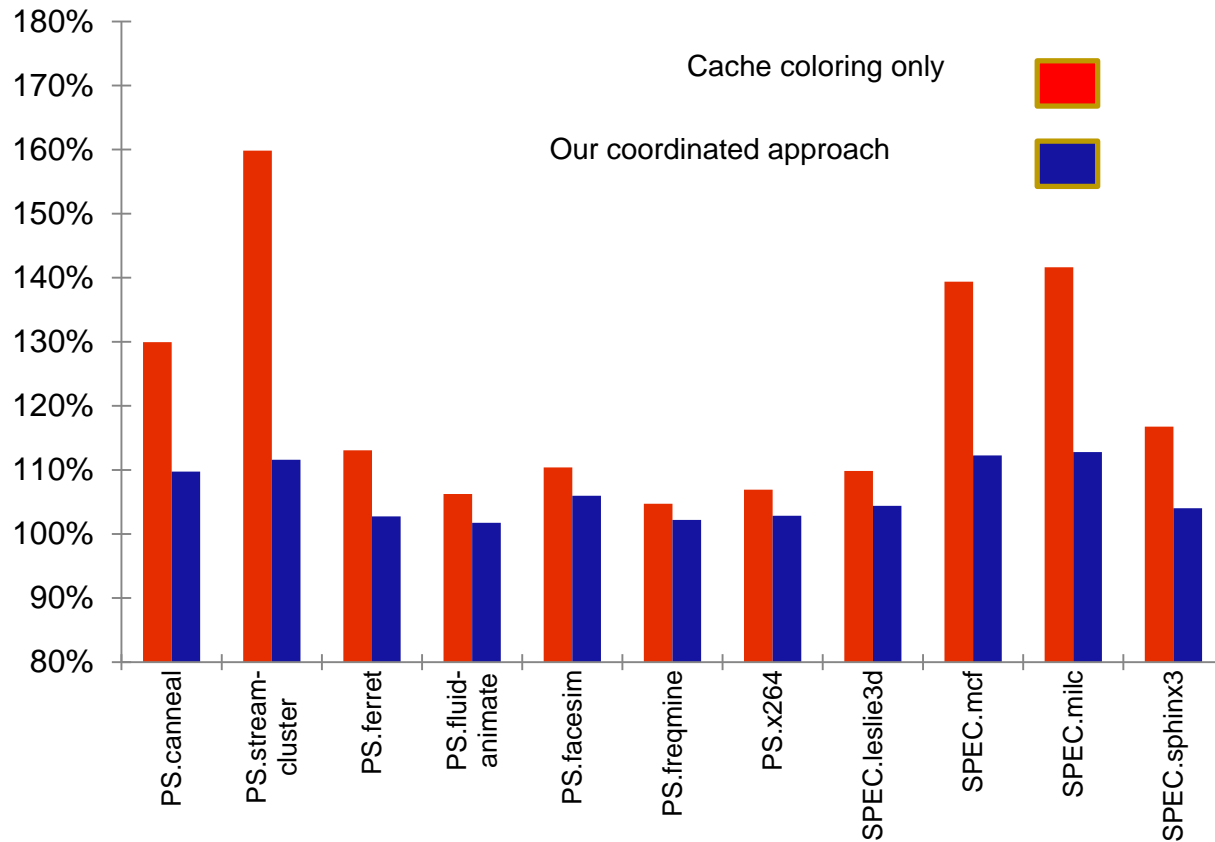
- Task with highest sensitivity to cache is assigned more cache
- Diminishing returns taken into account

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

Interfering memory requests during the job execution

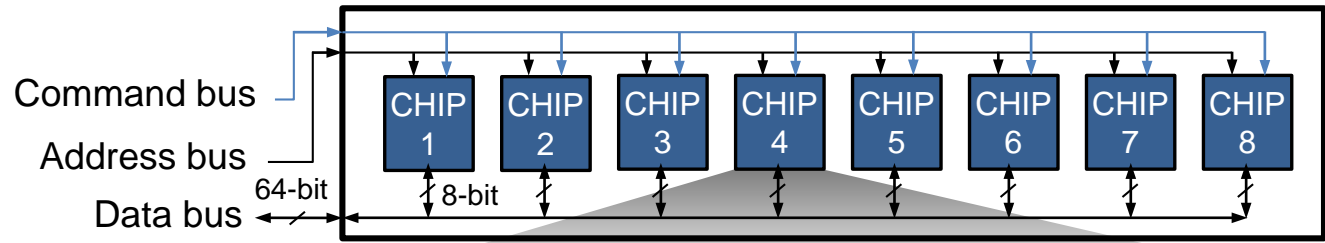
Software **DRAM bank partitioning** awareness

- Analyzes the effect of dedicated and shared DRAM banks

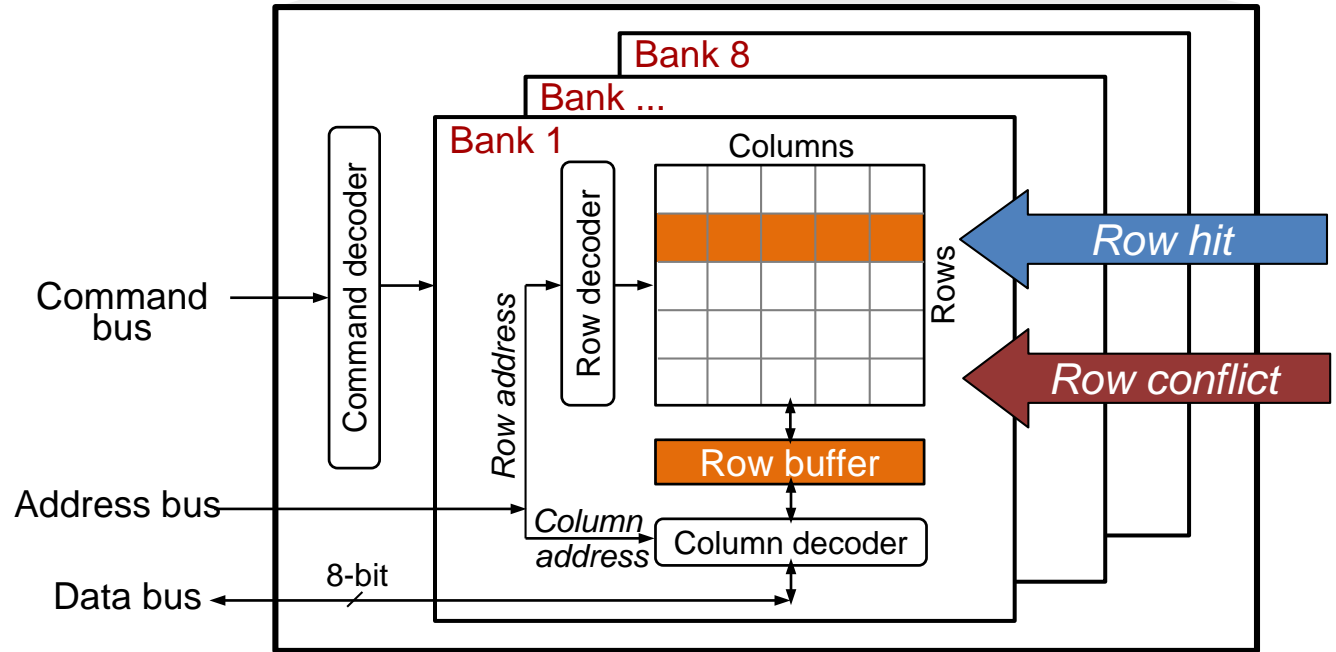


# DRAM Organization

## DRAM Rank



## DRAM Chip

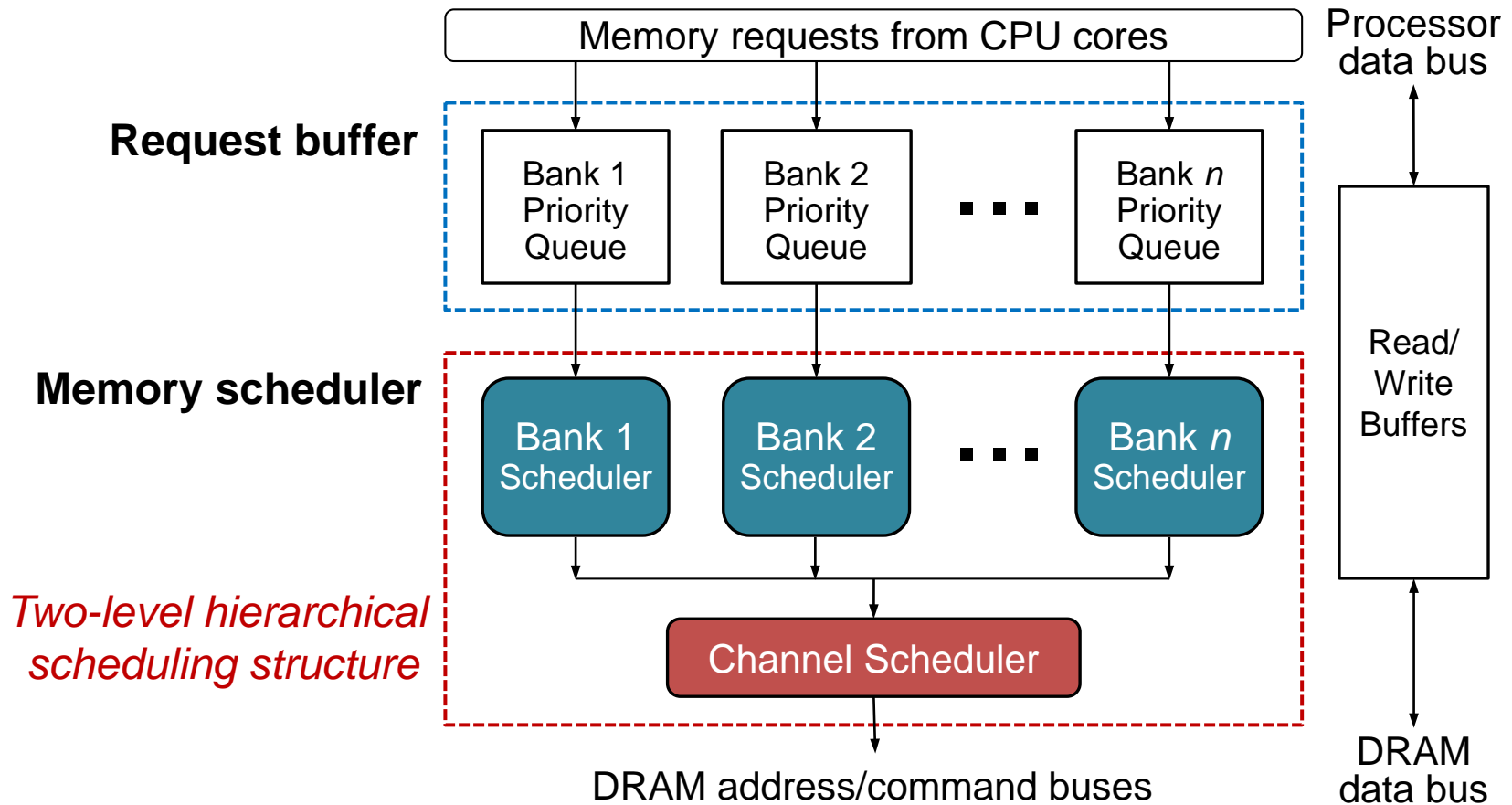


**DRAM access latency** varies depending on which row is stored in the row buffer



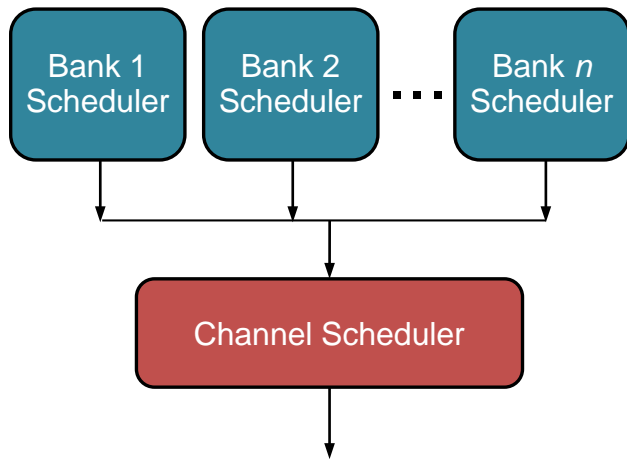


# Memory Controller



# 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

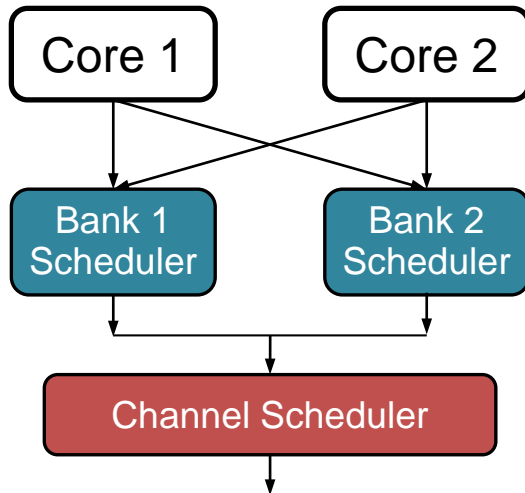
- *Intra-bank interference* at bank scheduler
- *Inter-bank interference* at channel scheduler



# DRAM Bank Partitioning

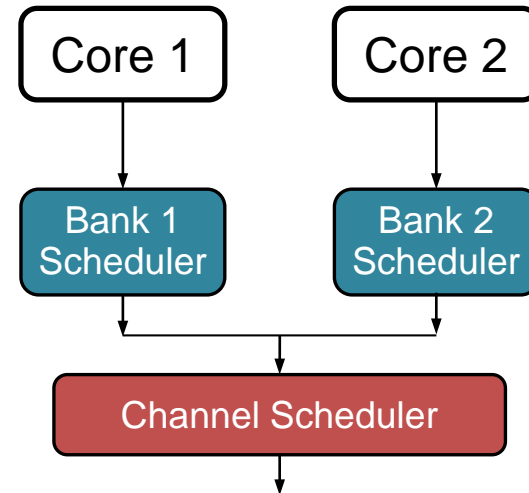
- **Prevents intra-bank interference** by dedicating different DRAM banks to each core
  - Can be supported in the OS kernel

(1) w/o bank partitioning



*Intra-bank and inter-bank interference*

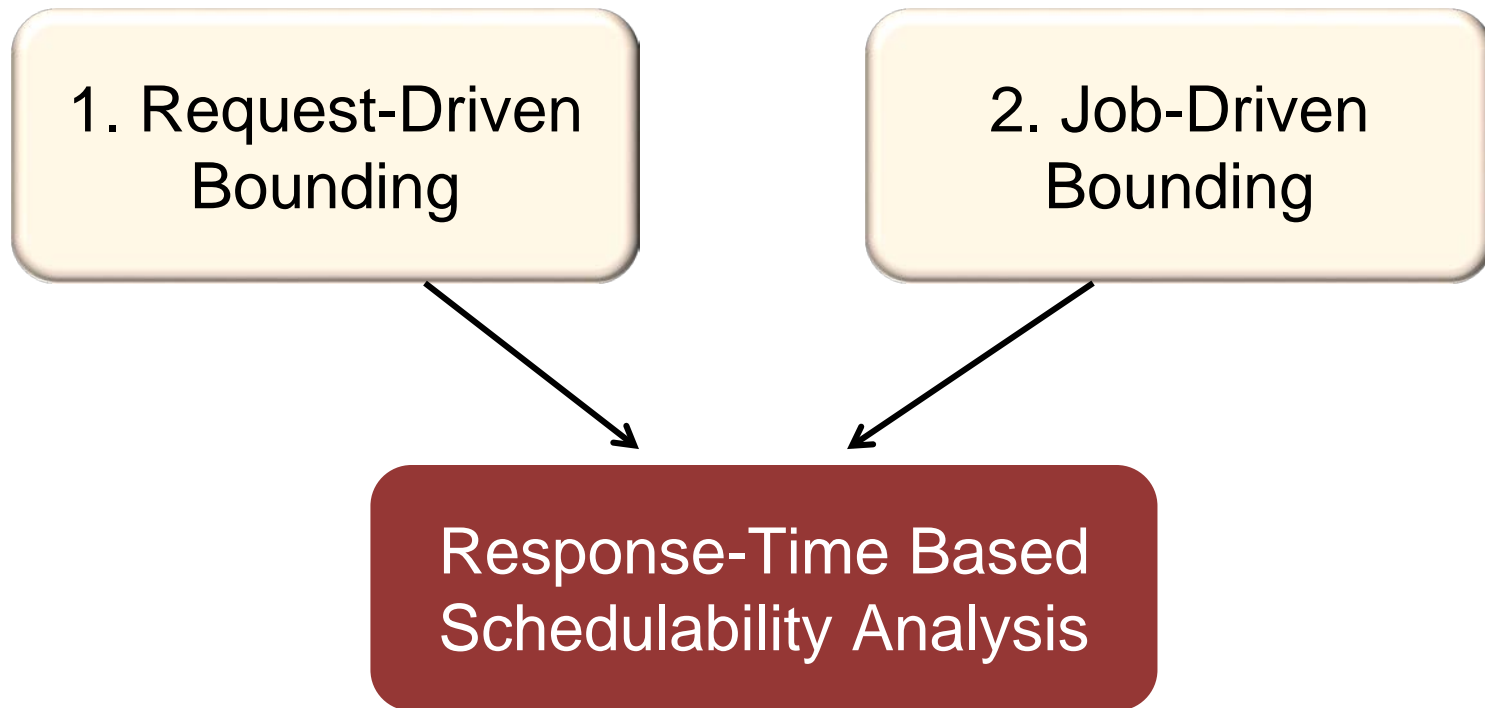
(2) w/ bank partitioning



*Only inter-bank interference*



# 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*

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

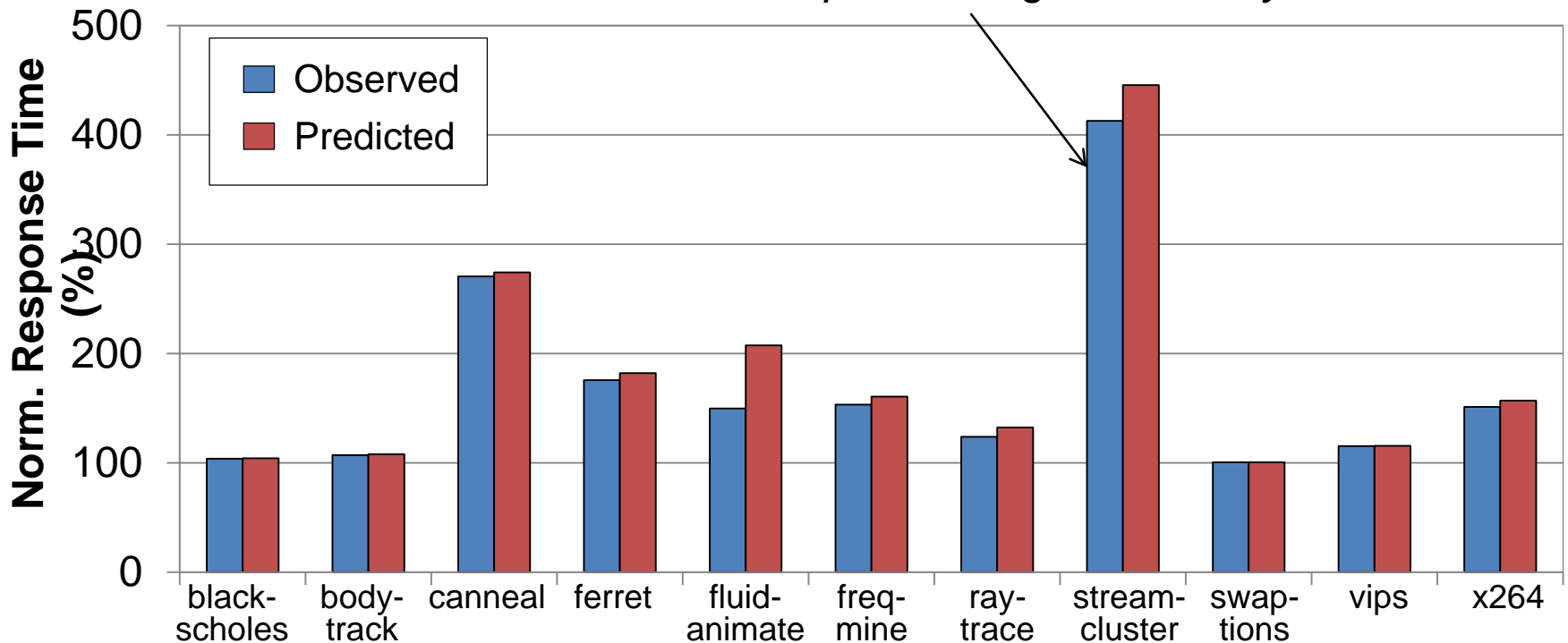
*Request-Driven (RD) Approach*      *Job-Driven (JD) Approach*



# Experiment Severe Memory Interference

- **Private DRAM Bank**

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

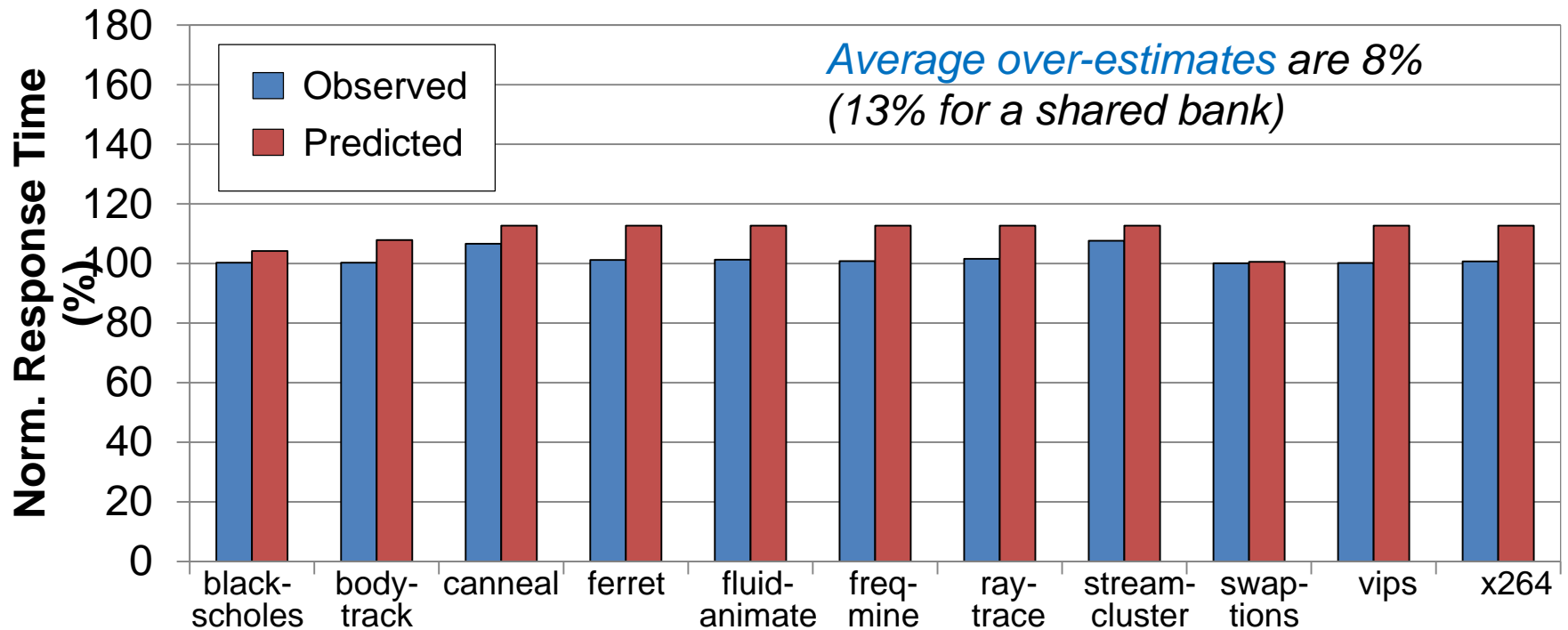


Our analysis enables the **quantification** of the benefit of DRAM bank partitioning



# 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

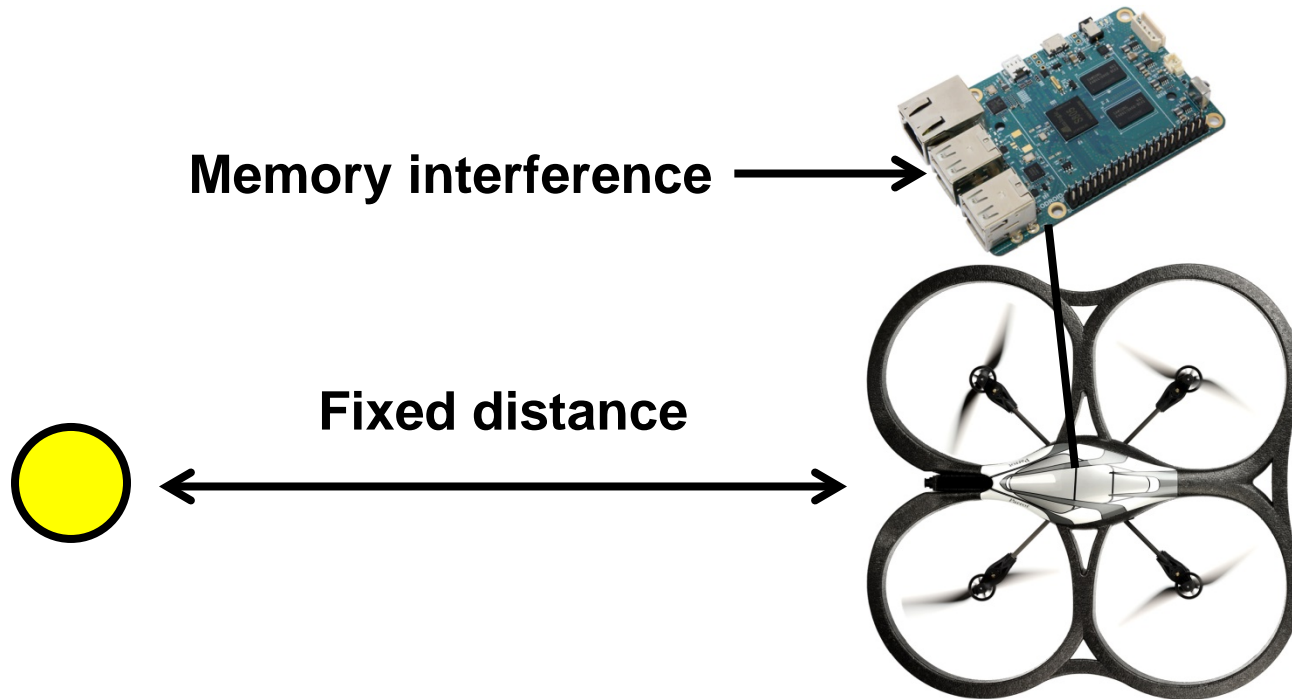
- Hooks into on-demand page allocation
- At boot time create large memory reserve
  - Pages are classified in cache and bank colors





# Model Problem

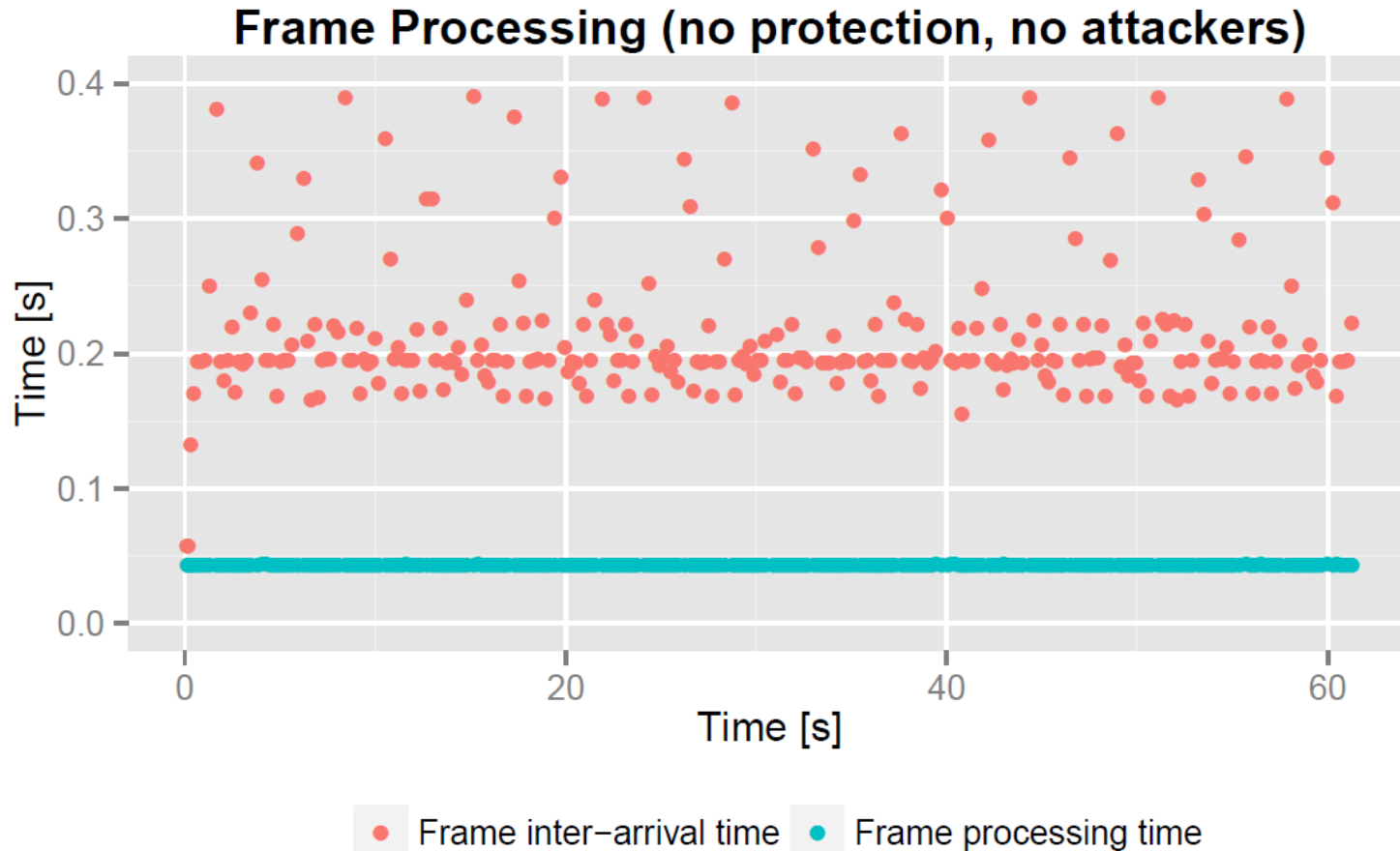
Ball-following Controller  
on Odroid+Linux/RK



**Ball-Following: Keep fixed distance as  
ball moves around**

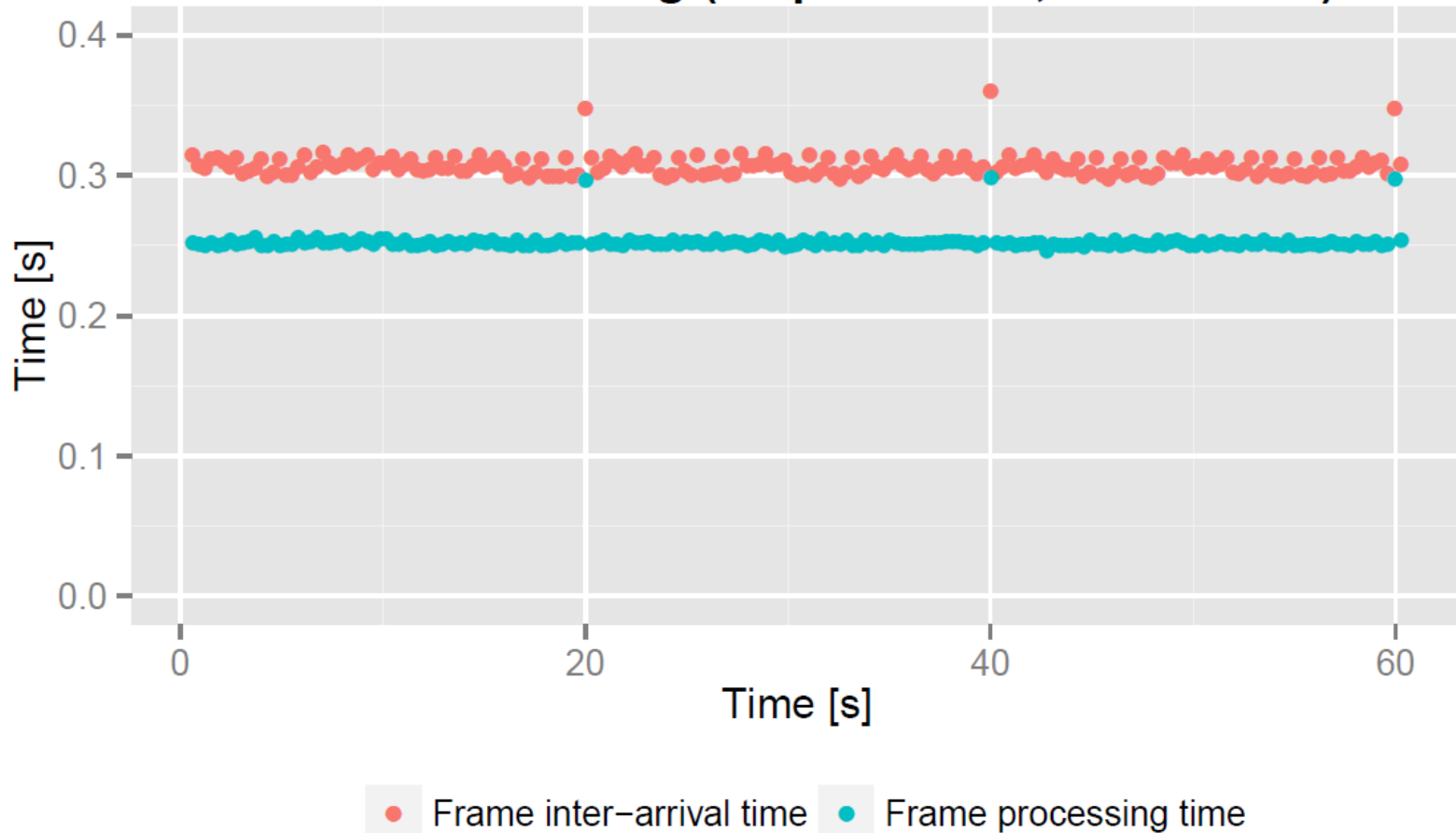


# Experimental Results (1)



# Experimental Results (2)

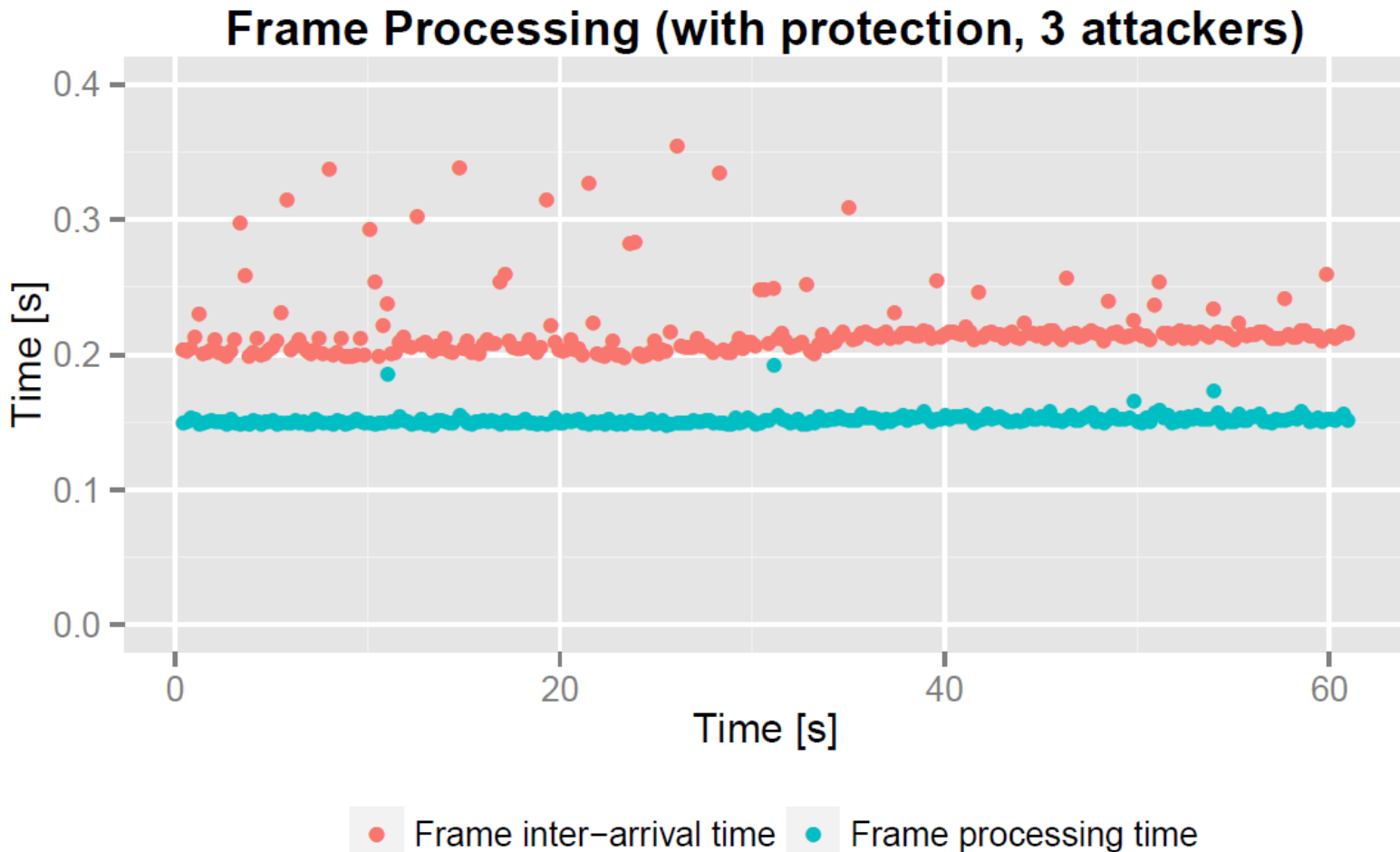
## Frame Processing (no protection, 3 attackers)



**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

