Consolidation of Real-Time Systems into a Multi-Core Platform

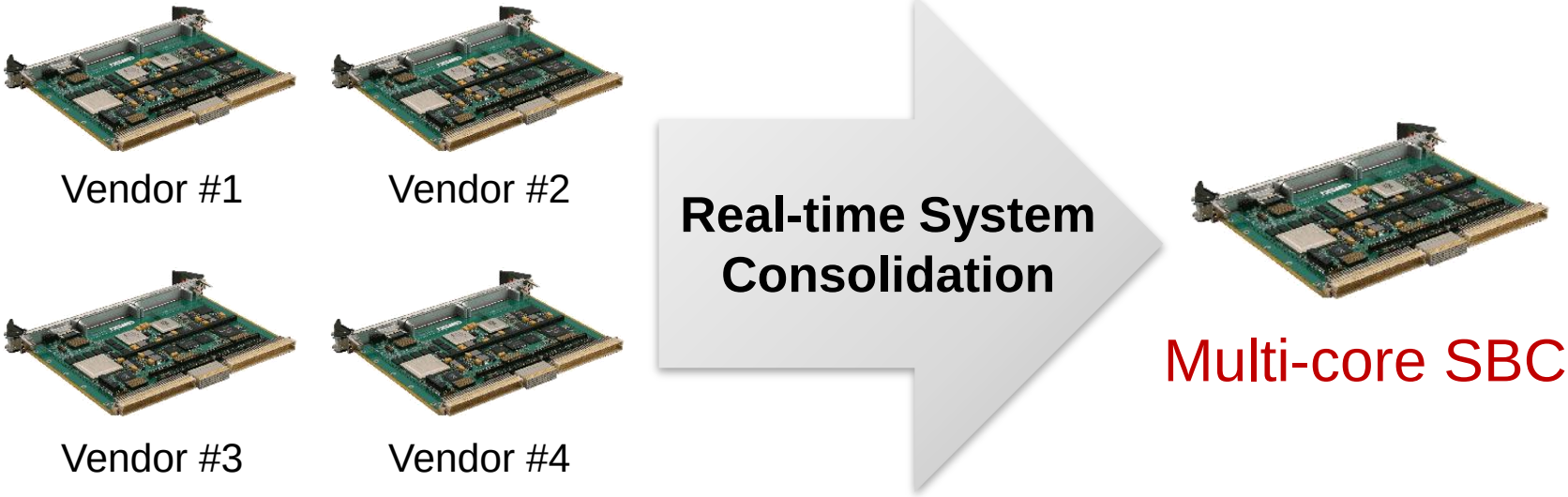
Hyoseung Kim
hyoseung@cmu.edu

Raj Rajkumar
raj@ece.cmu.edu

Carnegie Mellon University

Consolidation onto Multicores

- Benefits
  - Reduces the number of CPUs and wiring harness among them
  - Leads to a significant reduction in cost and space requirements



Key Challenges

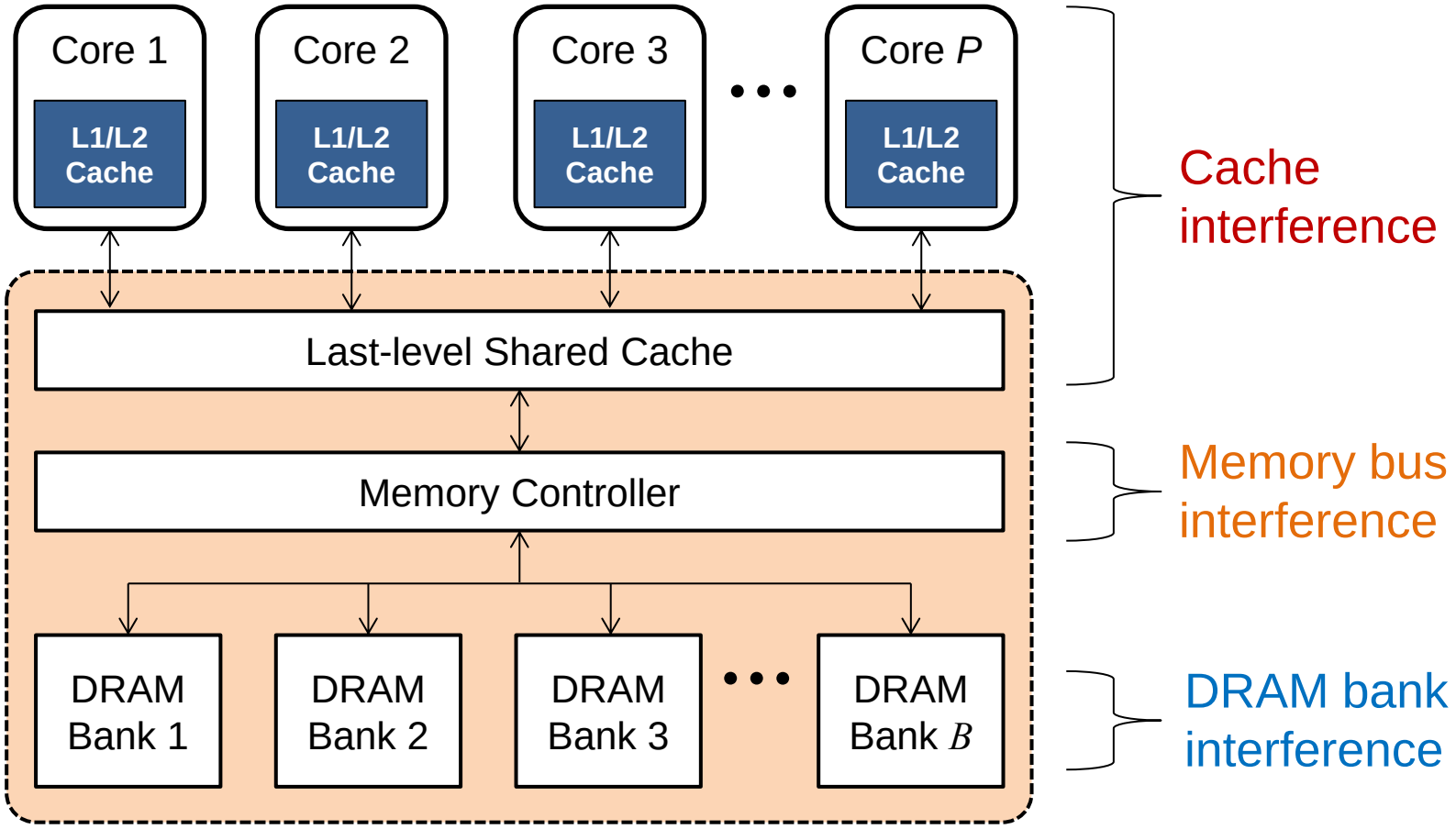
Timing predictability without losing efficiency

Temporal isolation among consolidated workloads

(Quantification & minimization of temporal interference)

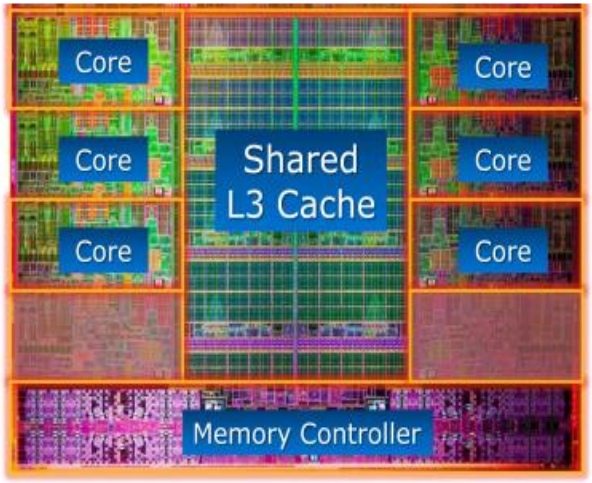
Use of standardized COTS multi-core platforms

Multi-Core Memory Hierarchy



Last-Level Shared Cache

- A large cache shared among processing cores
  - Reduces memory access time (7 – 10x faster than DRAM access)
  - Mitigates DRAM bandwidth consumption
  - Allows high cache utilization

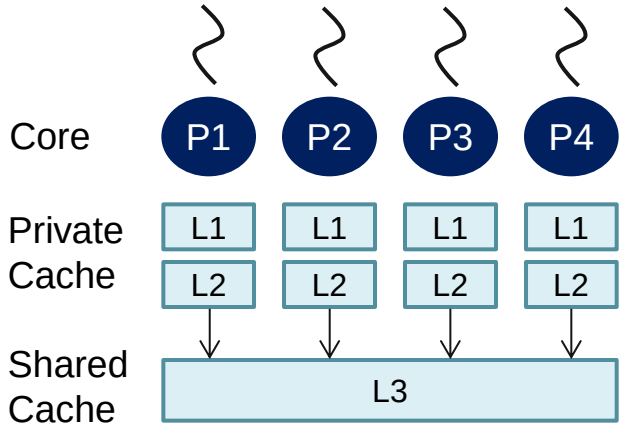


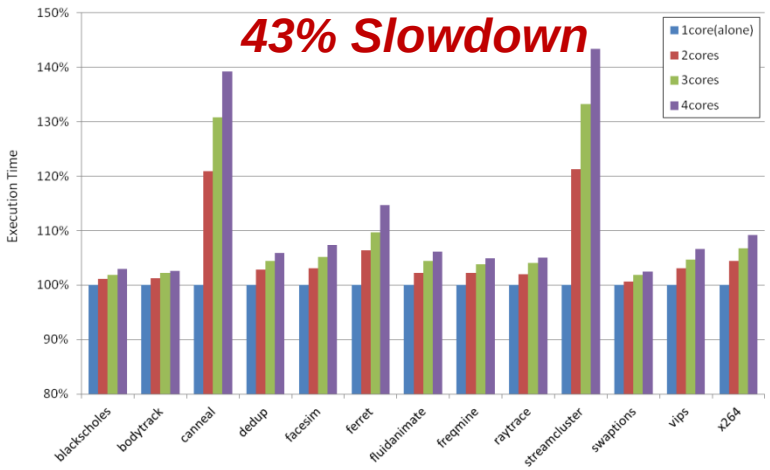
Intel Core i7
8-15 MB L3 Cache

Freescale P4080
2MB L3 Cache

Uncontrolled Use of Shared Cache

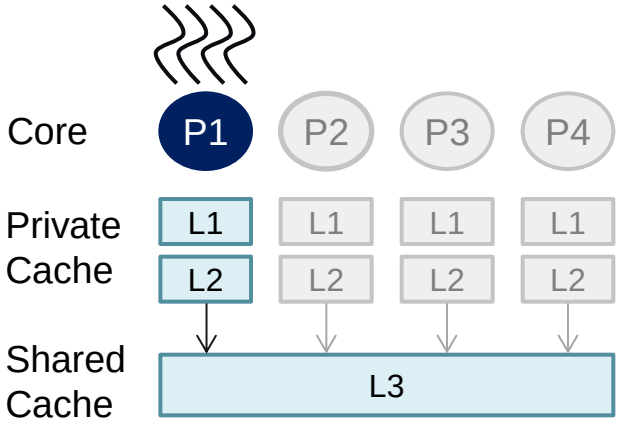
1. Inter-core Interference

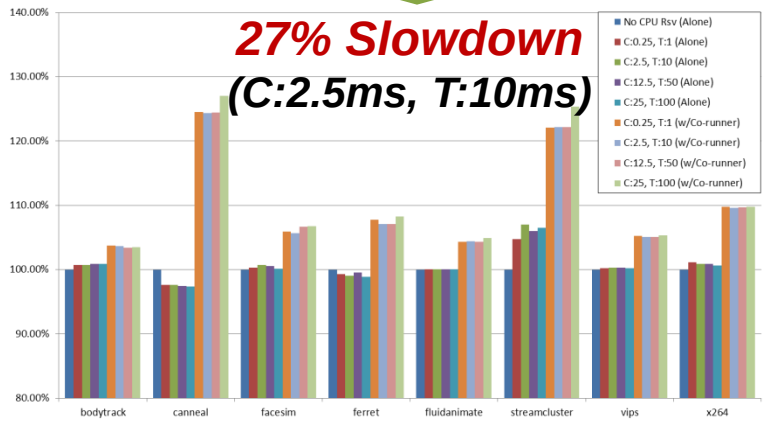




* PARSEC Benchmark on Intel i7

2. Intra-core Interference





Carnegie Mellon

Coordinated Cache Management

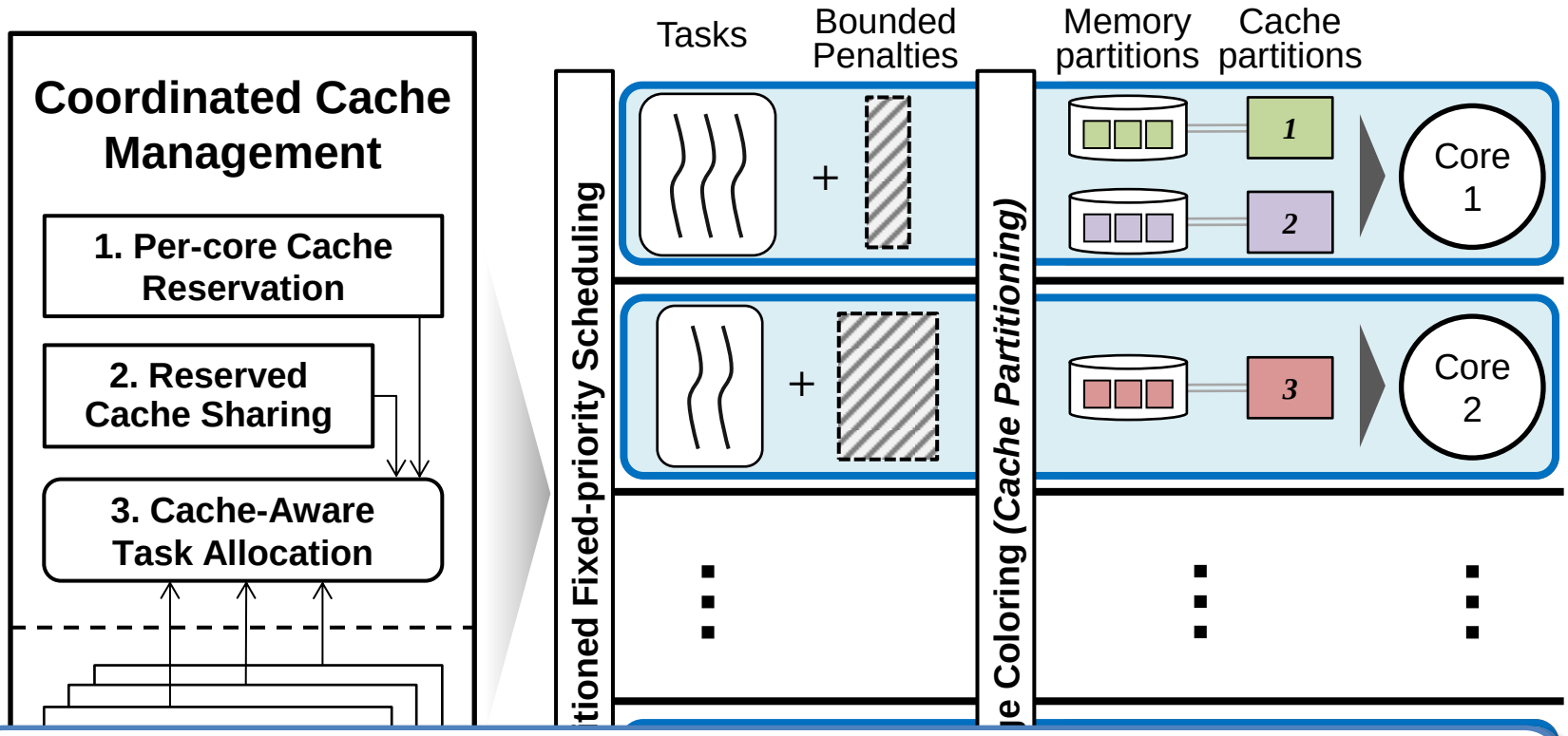
- **Challenges**
  - **Uncontrolled shared cache:** Cache interference penalties
  - **Cache partitioning:**
    - Limited number of cache partitions
    - Cache under-utilization

- **Key idea:** Controlled sharing of partitioned caches while ensuring timing predictability

1. Provides predictability on multi-core real-time systems
2. Addresses the limitations of cache partitioning

[ECRTS14] Hyoseung Kim, Arvind Kandhalu, and Raj Rajkumar. A Coordinated Approach for Practical OS-Level Cache Management in Multi-Core Real-Time Systems. In ECRTS, 2013.

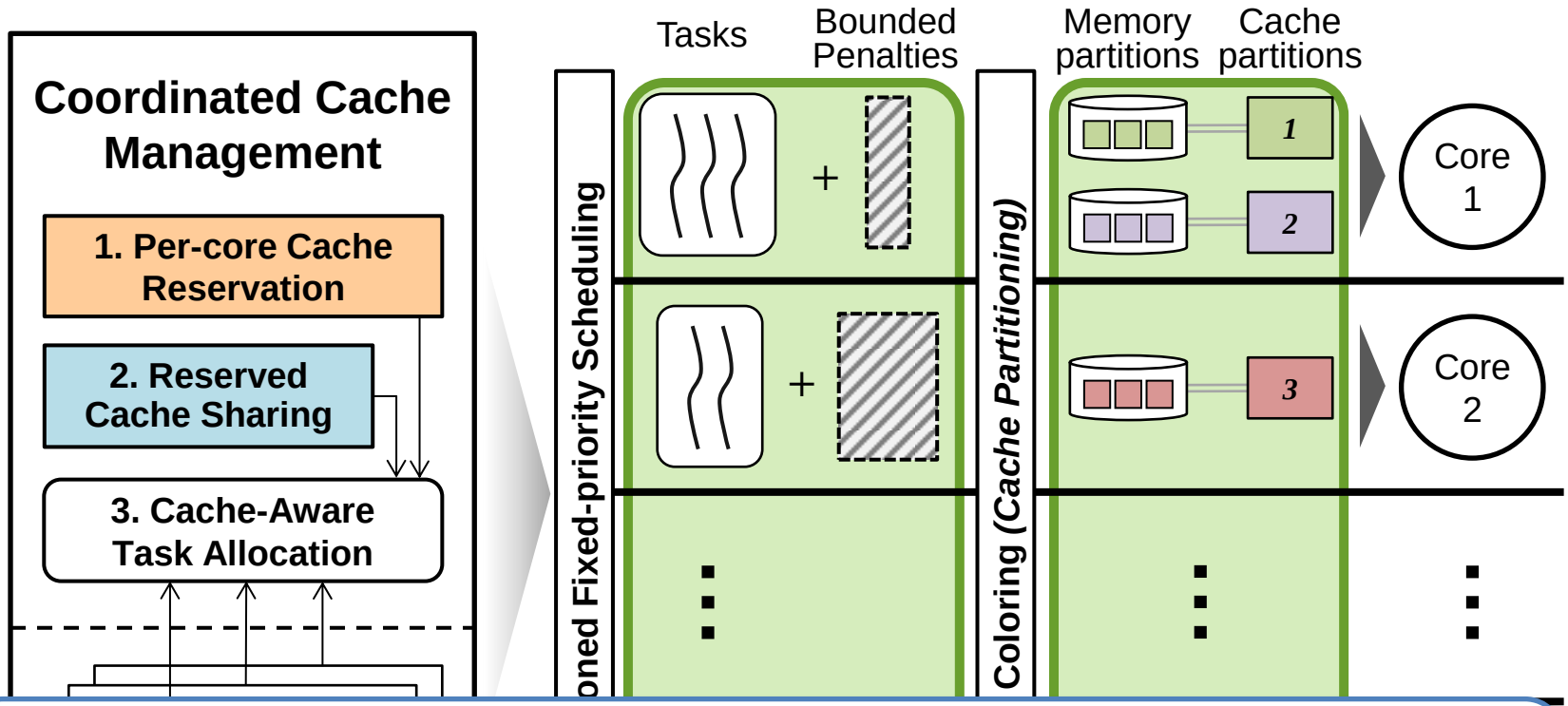
Coordinated Cache Management



Reserved cache sharing: Mitigate the problems with page coloring

Considerations
1. Analyzing intra-core interference
2. Guaranteeing memory requirements

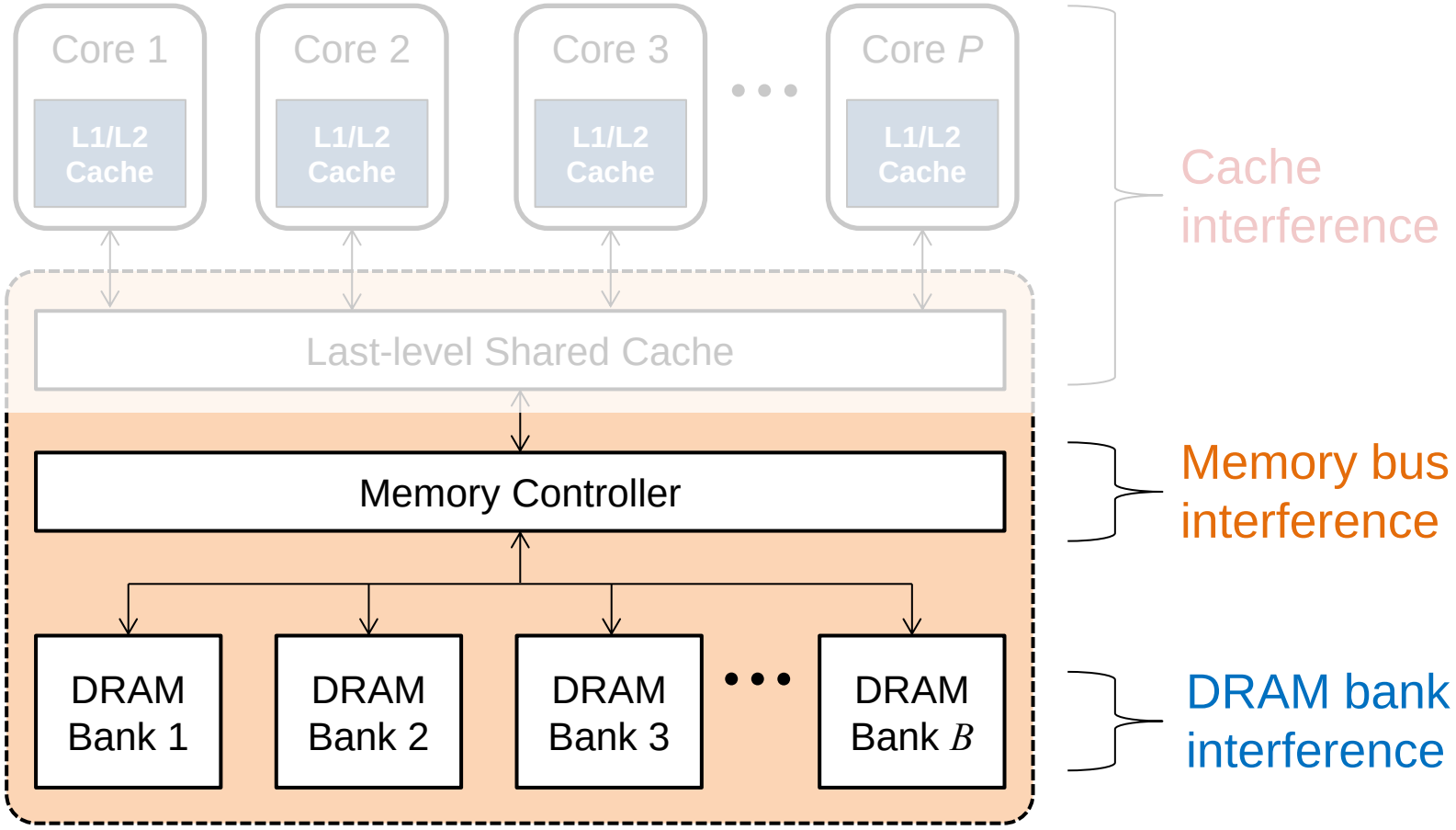
Coordinated Cache Management



Cache-Aware Task Allocation

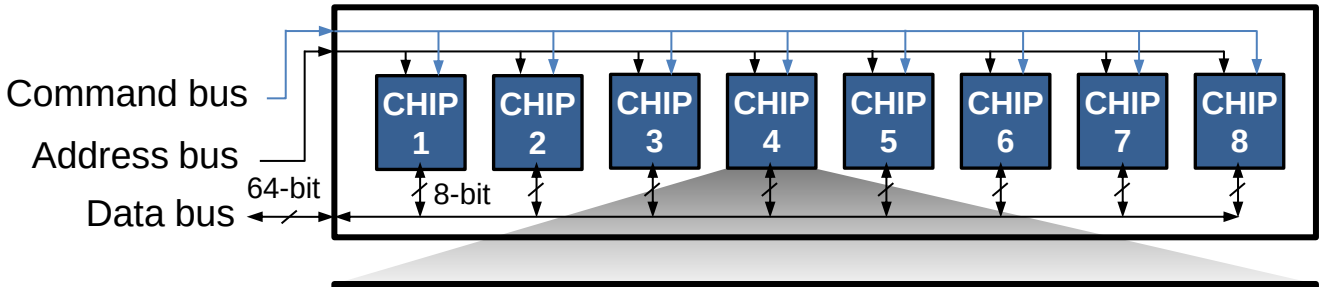
- Algorithm to allocate tasks and cache partitions to cores
- Exploits the benefits of cache sharing by load concentration

Multi-Core Memory Hierarchy

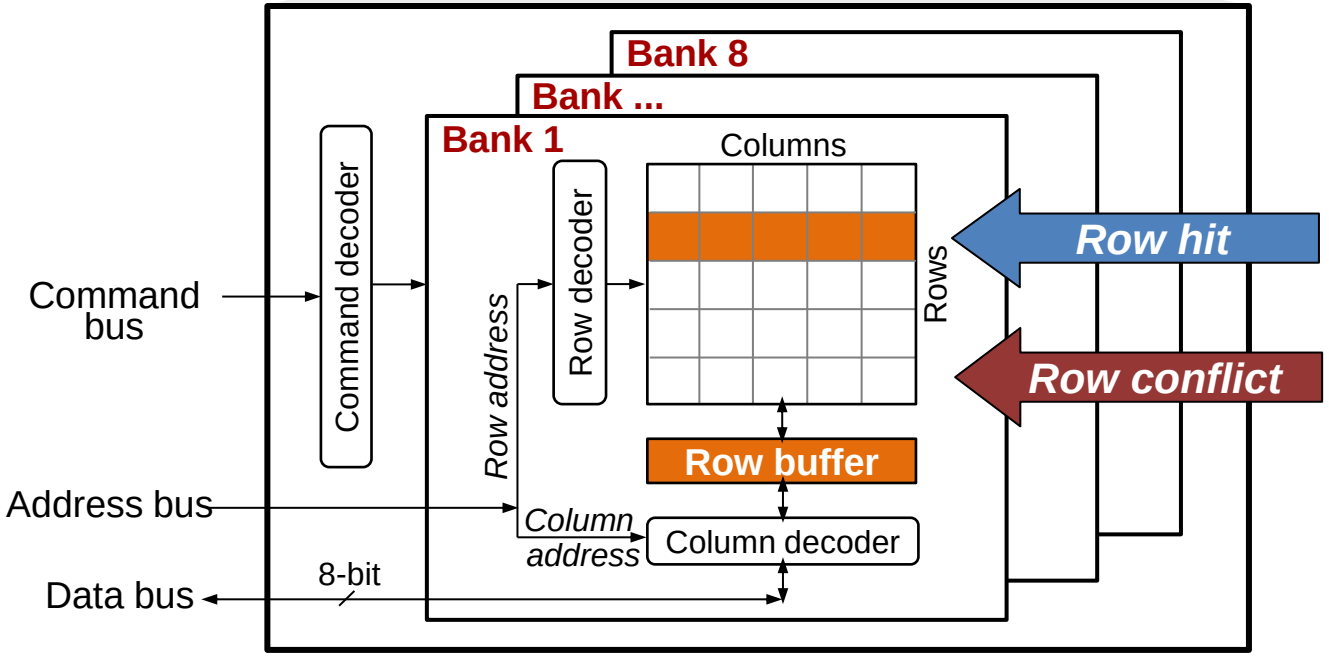


DRAM Organization

DRAM Rank

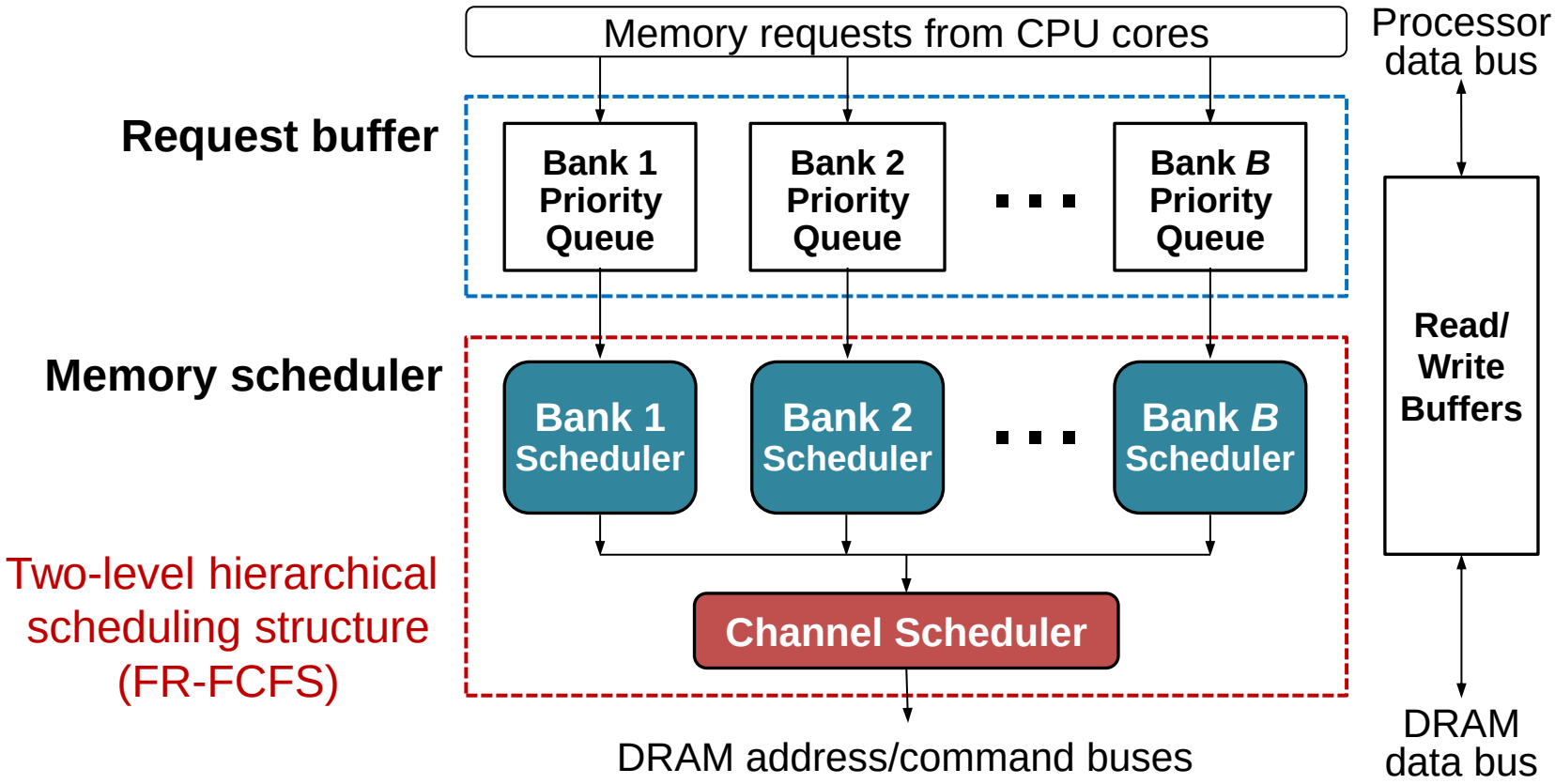


DRAM Chip



DRAM access latency depends on which row is currently in the row buffer

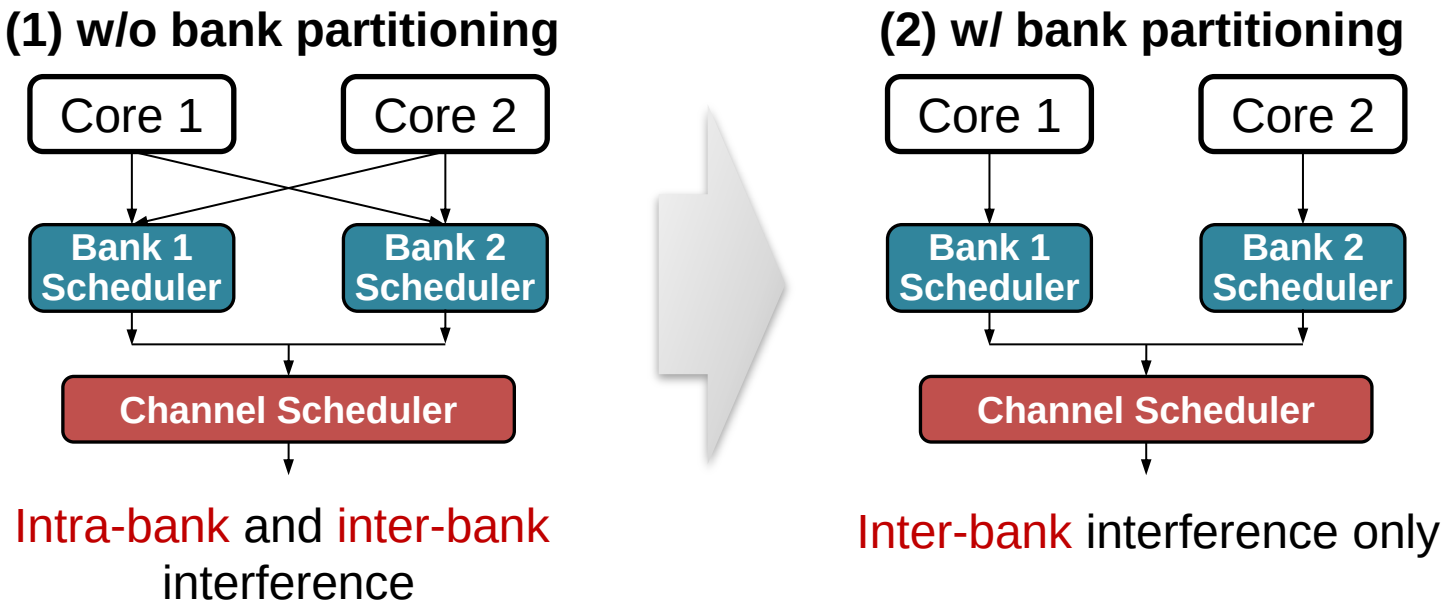
Memory Controller



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

DRAM Bank Partitioning & Sharing

- **DRAM bank partitioning:** software method to assign dedicated DRAM banks to each core (or task)



Intra-bank and inter-bank interference

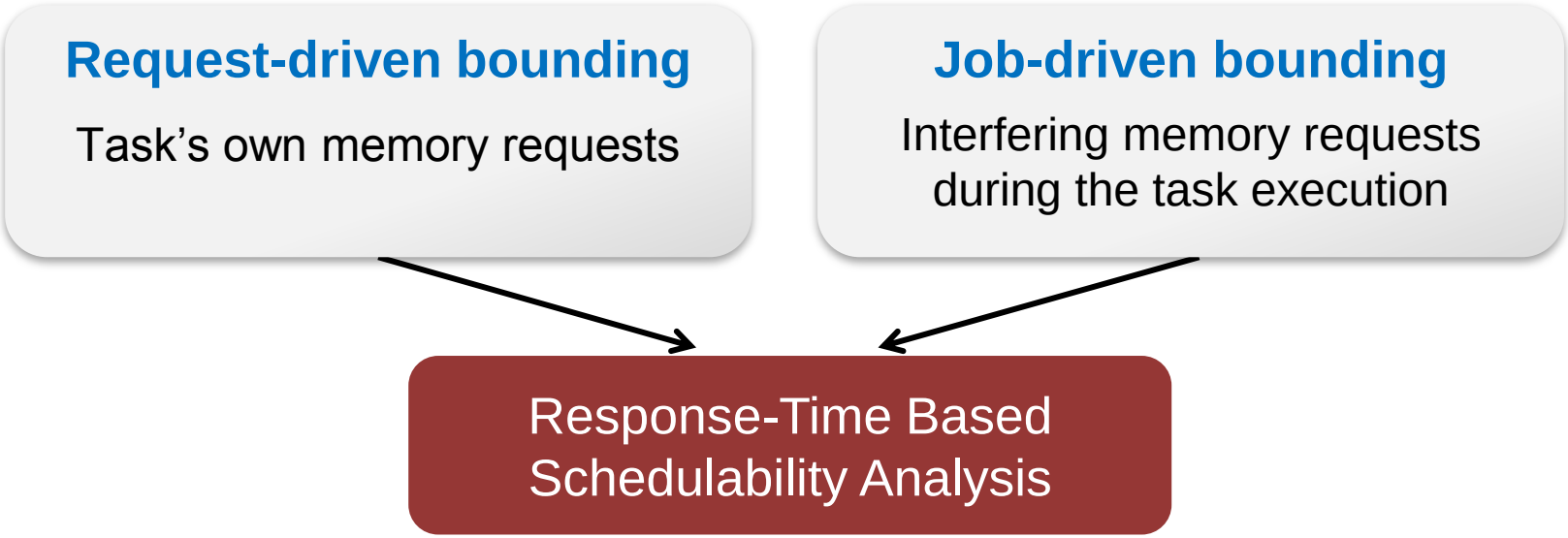
Inter-bank interference only

- **Problems**
  - Limited number of DRAM bank partitions
  - Reduced usable memory size

} **DRAM bank sharing**

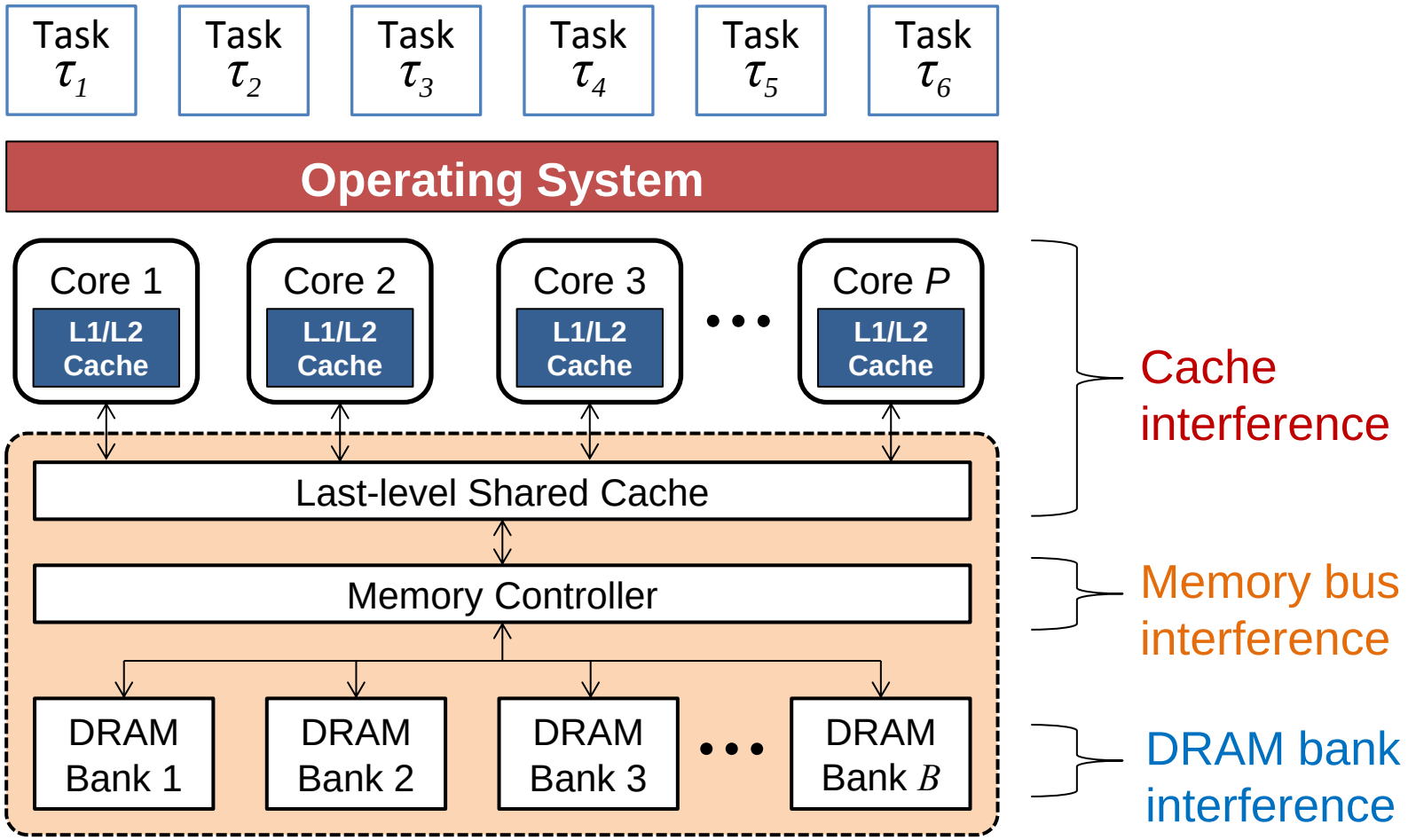
Bounding Memory Interference Delay

- Provides an upper bound on memory interference delay
  - Intra-bank and inter-bank interference delays
  - Private and shared DRAM banks



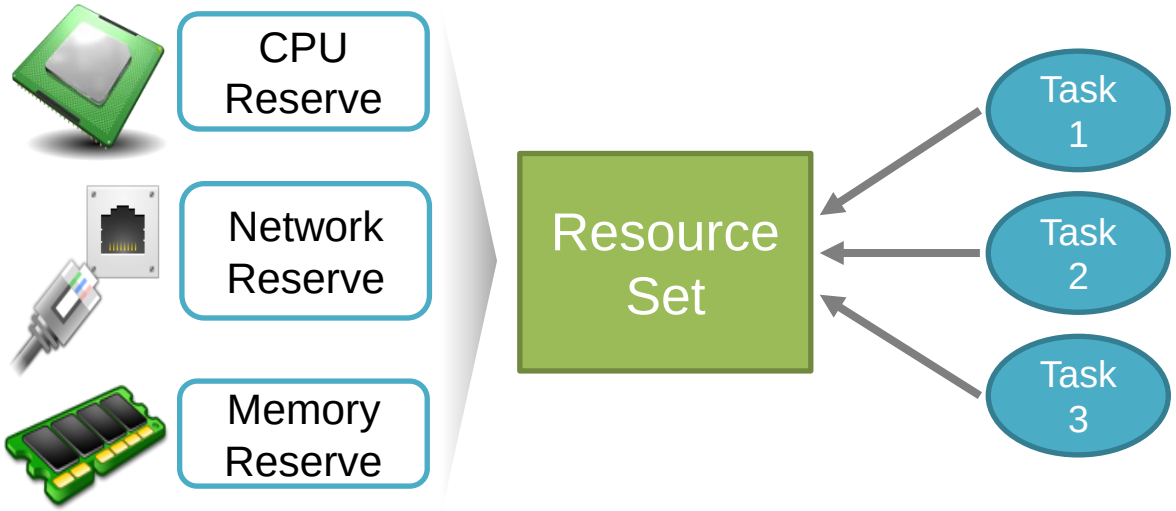
[RTAS14] Hyoseung Kim et al. Bounding Memory Interference Delay in COTS-based Multi-Core Systems. In IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), 2014. **Best Paper Award**

Multi-Core Memory Hierarchy



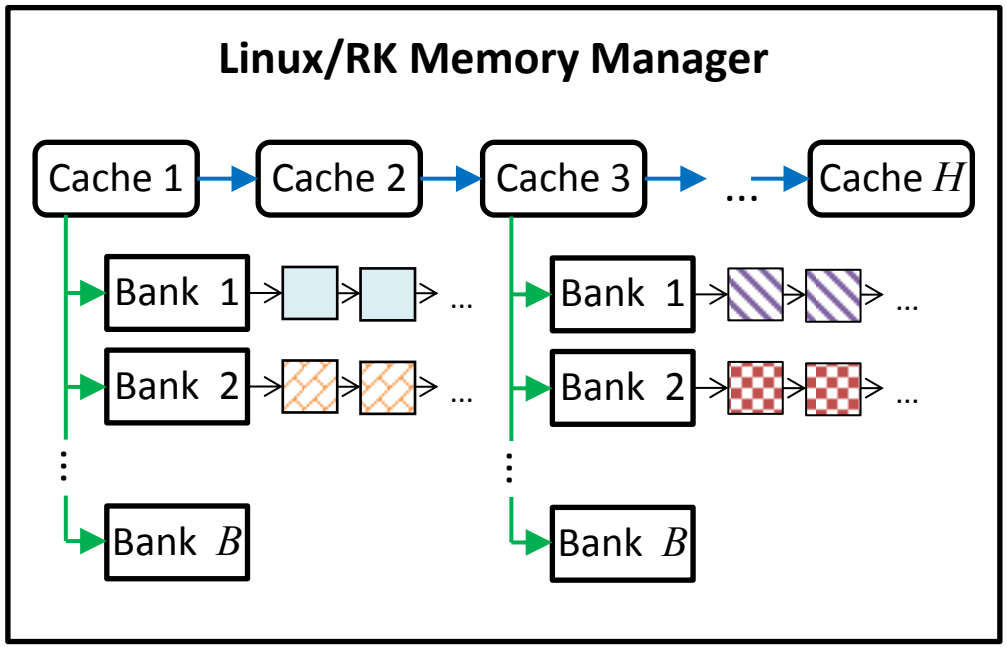
Linux/RK

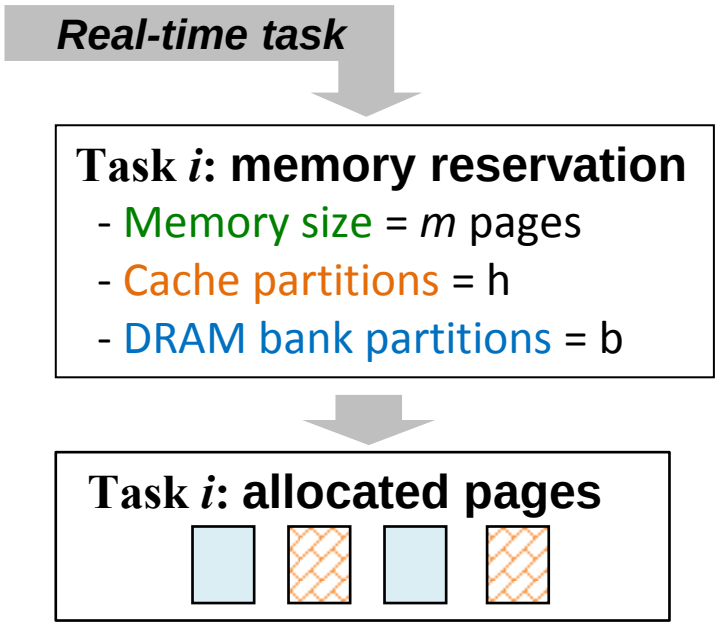
- CMU's open source real-time extensions to the Linux kernel
- Resource kernel (RK) approach
- Resource reservation: tasks can reserve a portion of system resources
  - Ex) CPU reserve: (budget, replenishment period, CPU affinity, policy)
- Guarantees resource allocations at admission time
- Enforces resource usage



Multi-Core Memory Management

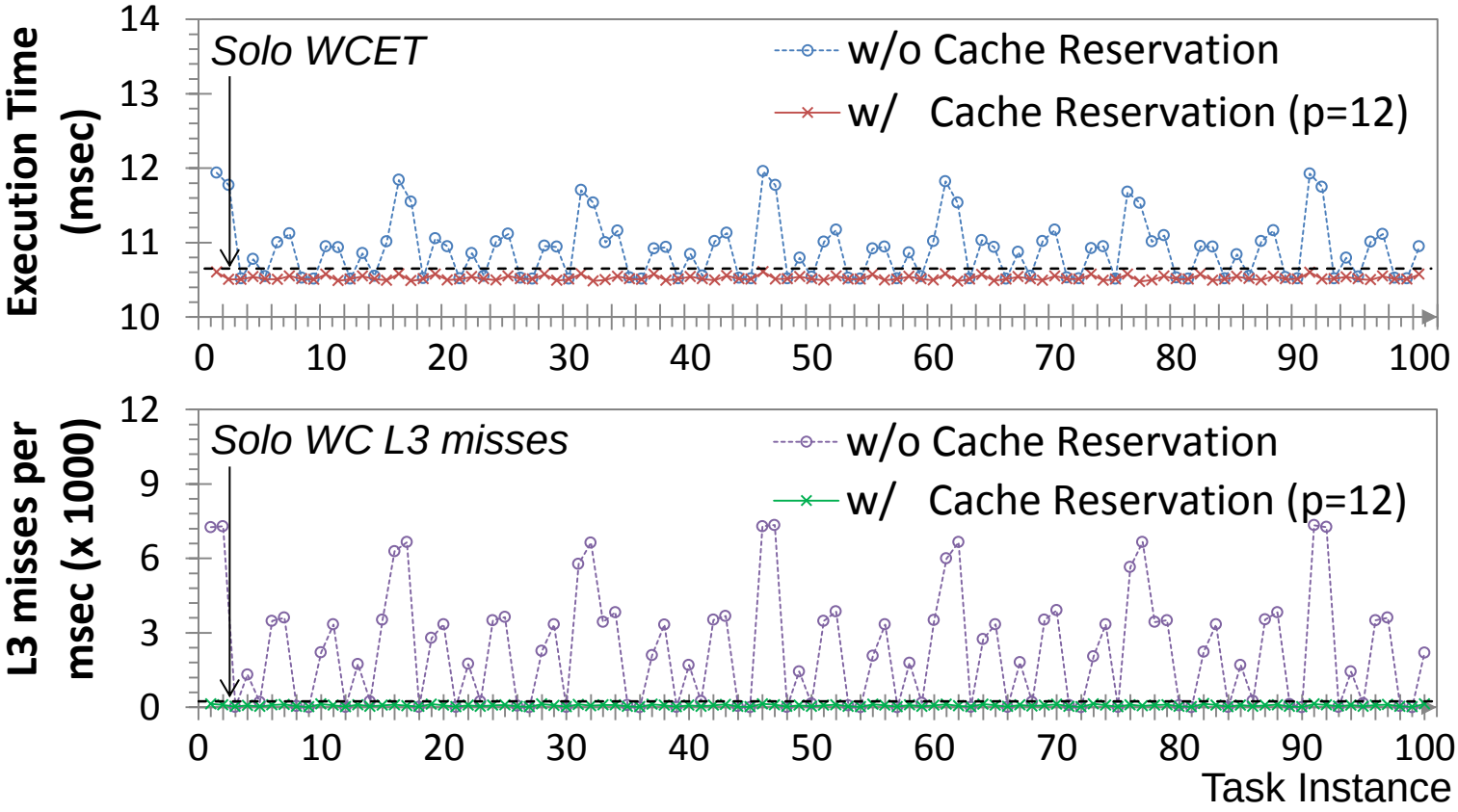
- **Linux/RK memory manager**
  - Supports software cache and DRAM partitioning
  - Allows tasks to specify their memory size, cache, DRAM bank demands





Cache Interference

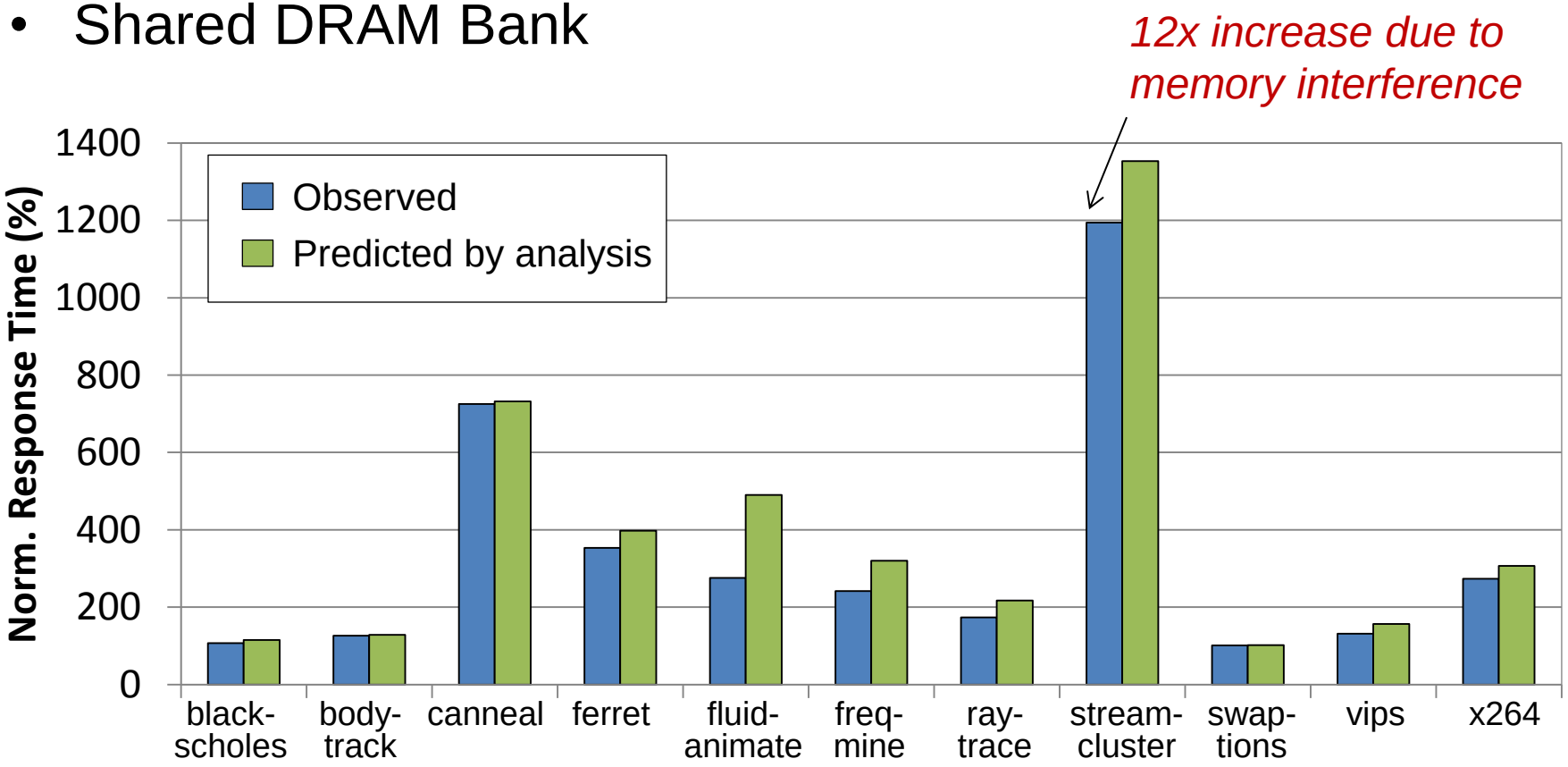
- Task execution time and L3 misses
  - Intel i7 2600 3.4GHz quad-core, 8MB L3 cache, DDR3-1333



Carnegie Mellon

Memory Interference

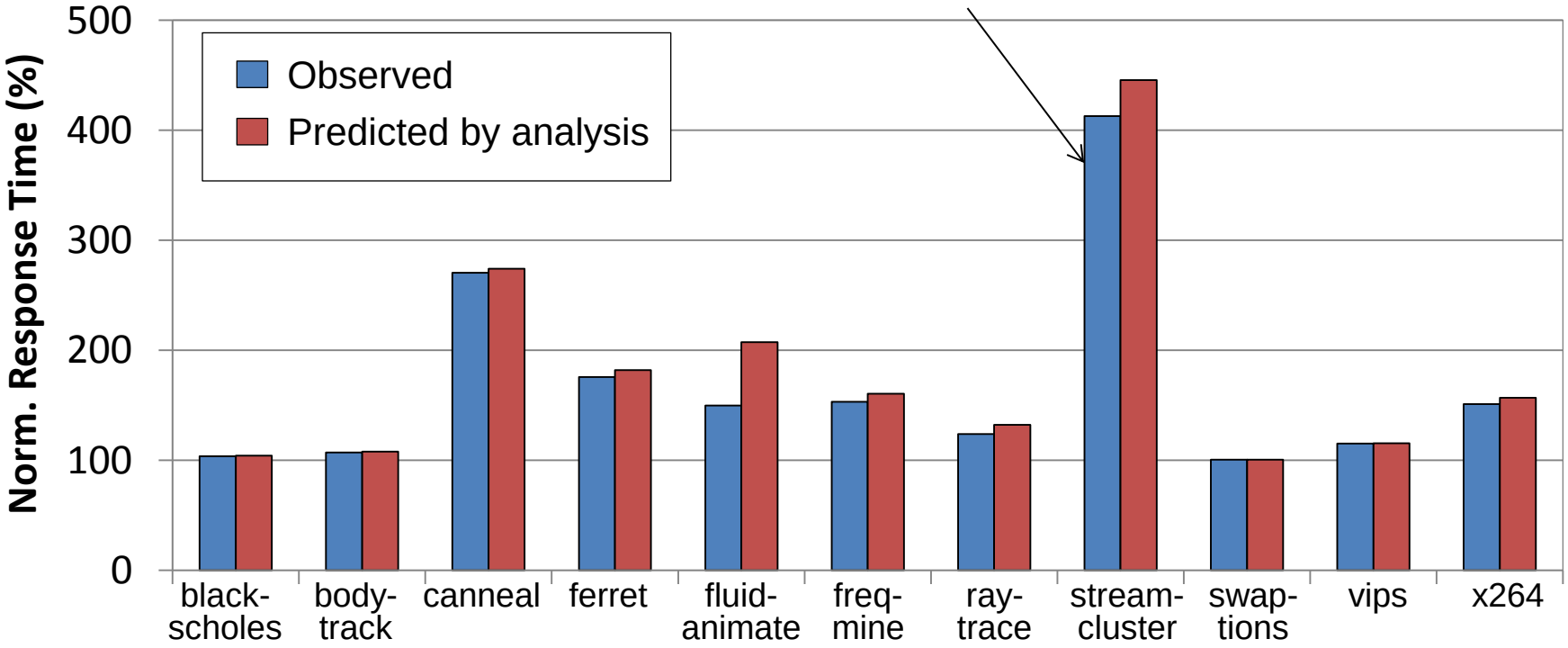
- Shared DRAM Bank



Memory Interference

- Private DRAM Bank

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

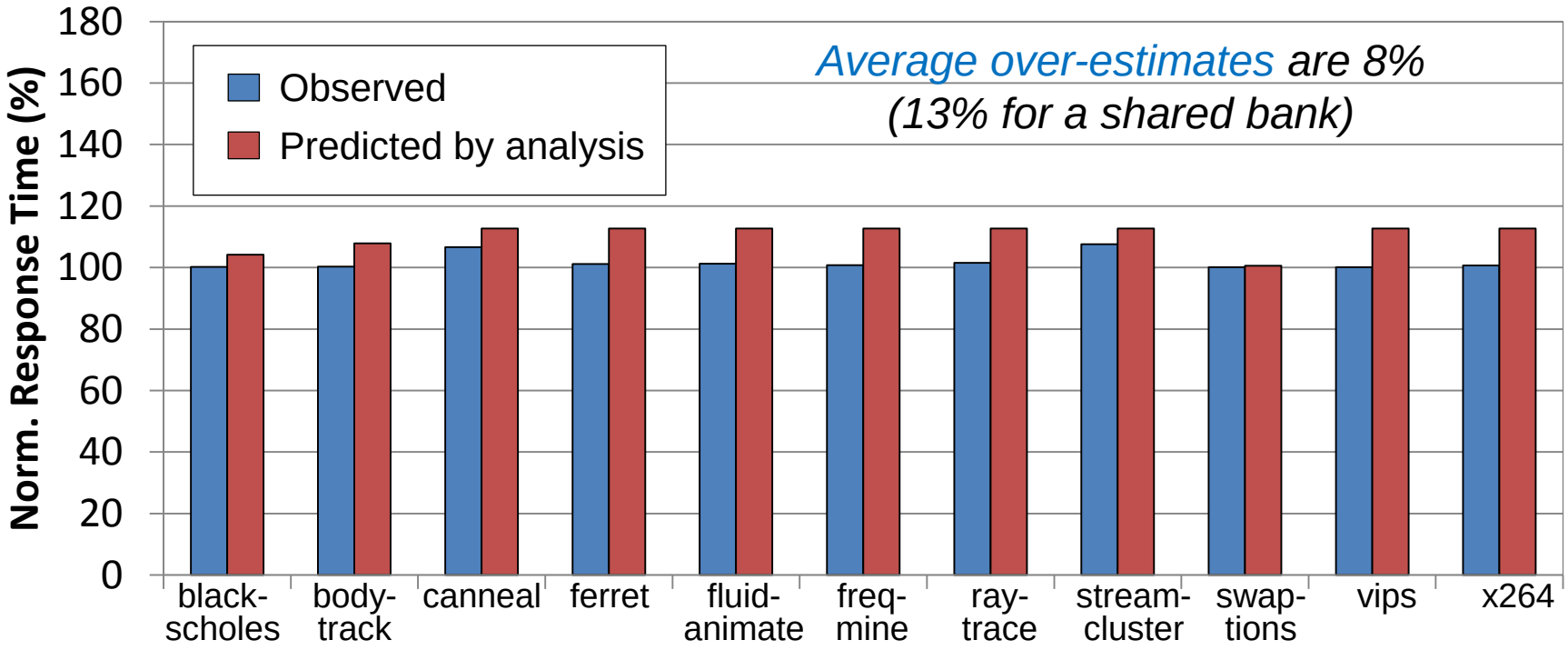


Our analysis enables the **quantification** of the benefit of DRAM bank partitioning

Carnegie Mellon

Memory Interference

- Private DRAM Bank under moderate memory contention



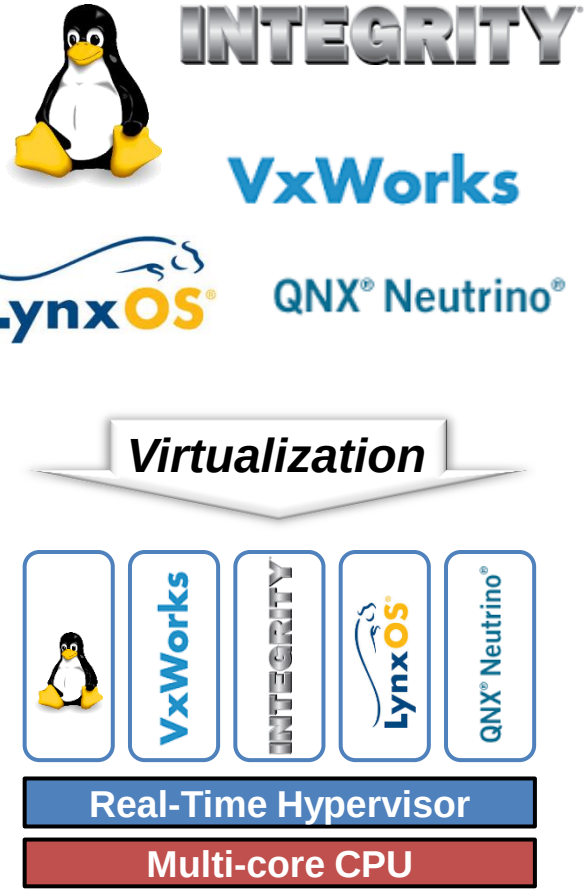
Our analysis bounds memory interference delay with **low pessimism** under both severe and moderate memory contentions

Carnegie Mellon

Multi-Core Virtualization

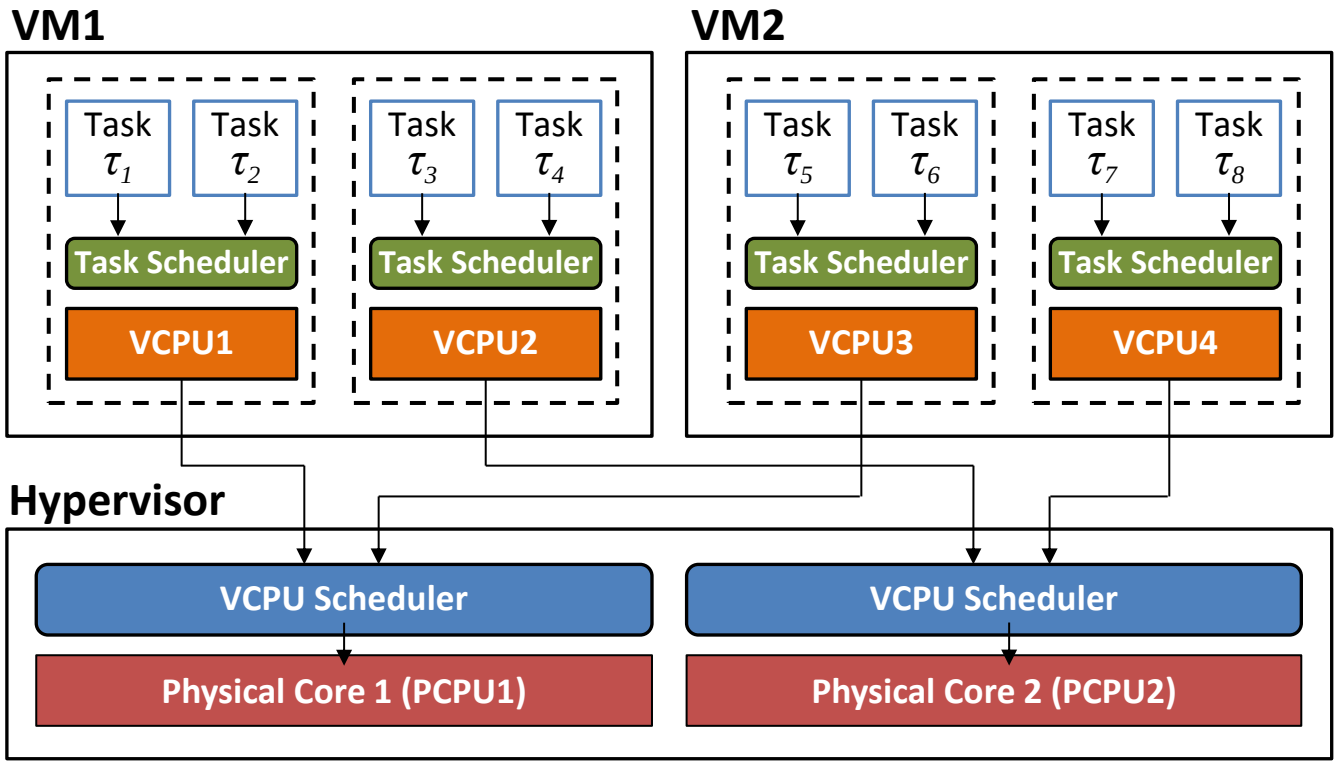
Real-Time System Virtualization

- **Barrier to consolidation**
  - Each app. could have been developed independently by different vendors
    - Bare-metal / Proprietary OS
    - Linux
  - Legacy software
  - Different license issues
- **Consolidation via virtualization**
  - Each application can maintain its own implementation
  - Minimizes re-certification process
  - IP protection, license segregation
  - Fault isolation



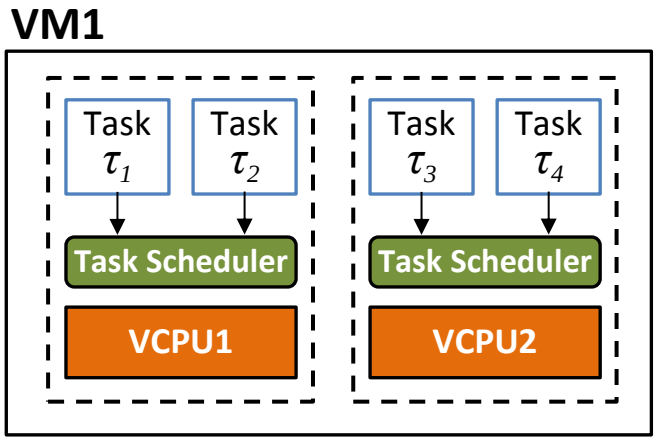
VM Scheduling Structure

- Two-level hierarchical scheduling structure
  - Task scheduling and VCPU scheduling



Virt/RK

- Real-time virtualization with resource kernel approach
  - CPU reservation for VCPUs
  - Memory reservation for VMs



VM Resource Reservation

- VCPU1: 30% of physical CPU
- VCPU2: 30% of physical CPU
- VM1: 20% of host memory w/ cache & DRAM bank partitioning

  - Current implementation: Virt/RK::KVM-x86
  - Under development: Virt/RK::KVM-ARM, Virt/RK::L4

Resource Sharing in Virtualization

- Consolidation inevitably causes the sharing of resources
  - Sensors
  - Network interfaces
  - I/O devices
  - Shared data

  } Requires mutually-exclusive locks to avoid race conditions

- Long blocking time in a virtualized environment
  - VCPU level preemptions
  - VCPU budget depletion

Need a synchronization mechanism with bounded blocking times for multi-core real-time virtualization

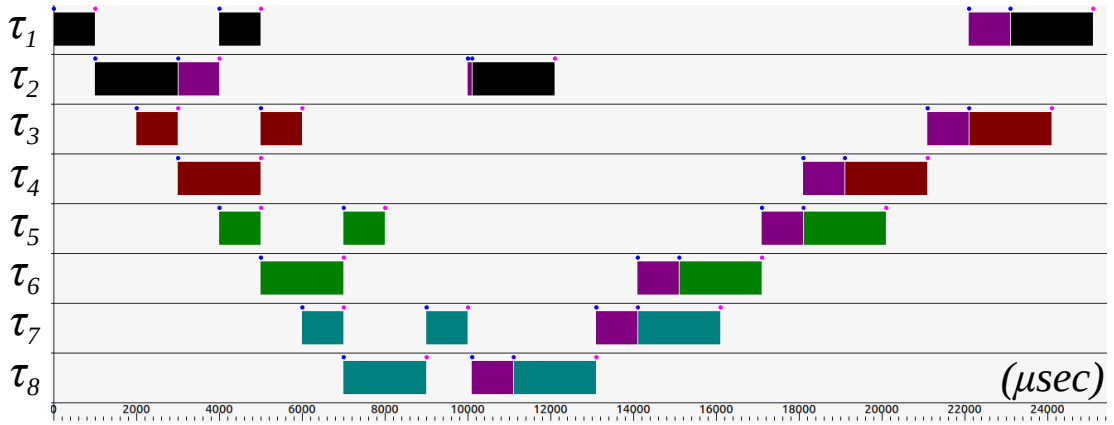
vMPCP

- Virtualization-aware Multiprocessor Priority Ceiling Protocol
  - Provides bounded blocking time on accessing shared resources in multi-core virtualization
    - Hierarchical priority ceilings
    - Two-level priority queue for a mutex waiting list
  - Optional VCPU budget overrun to reduce blocking times
  - VCPU budget replenishment policies
    - Periodic server
    - Deferrable server
  - Implemented on Virt/RK::KVM-x86

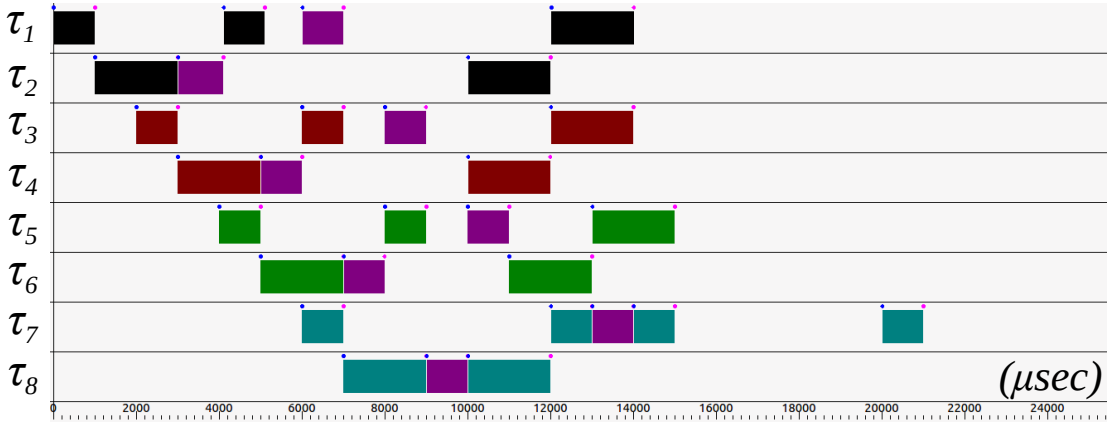
[RTSS14] Hyoseung Kim, Shige Wang, and Raj Rajkumar. vMPCP: A Synchronization Framework for Multi-Core Virtual Machines. In IEEE Real-Time Systems Symposium (RTSS), 2014.

Case Study: Effect of vMPCP

Baseline
Virtualization-unaware synchronization protocol (MPCP)



vMPCP
Virtualization-aware synchronization protocol



vMPCP yields 29% shorter task response time with shared resources

Carnegie Mellon

Summary

- **Workload Consolidation into a Multi-core Platform**
  - Predictability without losing efficiency
  - Quantification and minimization of temporal interference
- **Multi-core memory hierarchy**
  - Coordinated cache management
    - Cache partitioning, reserve, and sharing
  - Bounding memory interference
    - DRAM partitioning and sharing → Quantifies the benefit of partitioning
    - Memory-bus and DRAM bank interference analysis
- **Real-time virtualization**
  - Virt/RK: resource reservation approach
  - vMPCP: resource sharing for real-time virtualization

What next?

- Modeling state-of-the-art computer architecture techniques
  - Hardware prefetchers
  - Future memory controllers
  - Future DRAM designs

- Extending existing real-time systems work to virtualization
  - Diverse locking protocols (e.g., RW-lock)
  - Fault tolerance mechanisms
  - Hardware accelerations (e.g., GPGPU)