Seminar in Computer Architecture

Meeting 3a: Example Review II

Prof. Onur Mutlu

ETH Zürich

Spring 2020

12 March 2020

Example Paper Presentation

We Will Briefly Review This Paper

- Sai Prashanth Muralidhara, Lavanya Subramanian, Onur Mutlu, Mahmut Kandemir, and Thomas Moscibroda, **"Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning"** *Proceedings of the 44th International Symposium on Microarchitecture* (***MICRO***), Porto Alegre, Brazil, December 2011. Slides (pptx)

Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning

Sai Prashanth Muralidhara
Pennsylvania State University
smuralid@cse.psu.edu

Lavanya Subramanian
Carnegie Mellon University
lsubrama@ece.cmu.edu

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu

Mahmut Kandemir
Pennsylvania State University
kandemir@cse.psu.edu

Thomas Moscibroda
Microsoft Research Asia
moscitho@microsoft.com

Application-Aware Memory Channel Partitioning

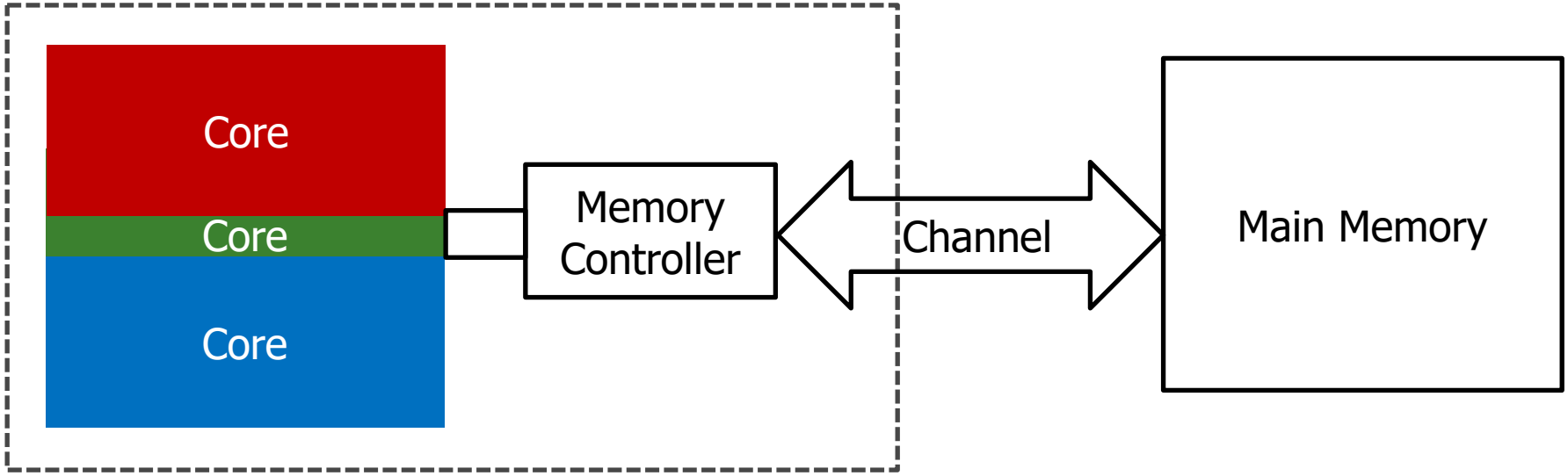
Sai Prashanth Muralidhara [§] Lavanya Subramanian [†]
Onur Mutlu [†] Mahmut Kandemir [§]
Thomas Moscibroda [‡]

[§] **Pennsylvania State University** [†] **Carnegie Mellon University**
[‡] **Microsoft Research**

SAFARI Carnegie Mellon

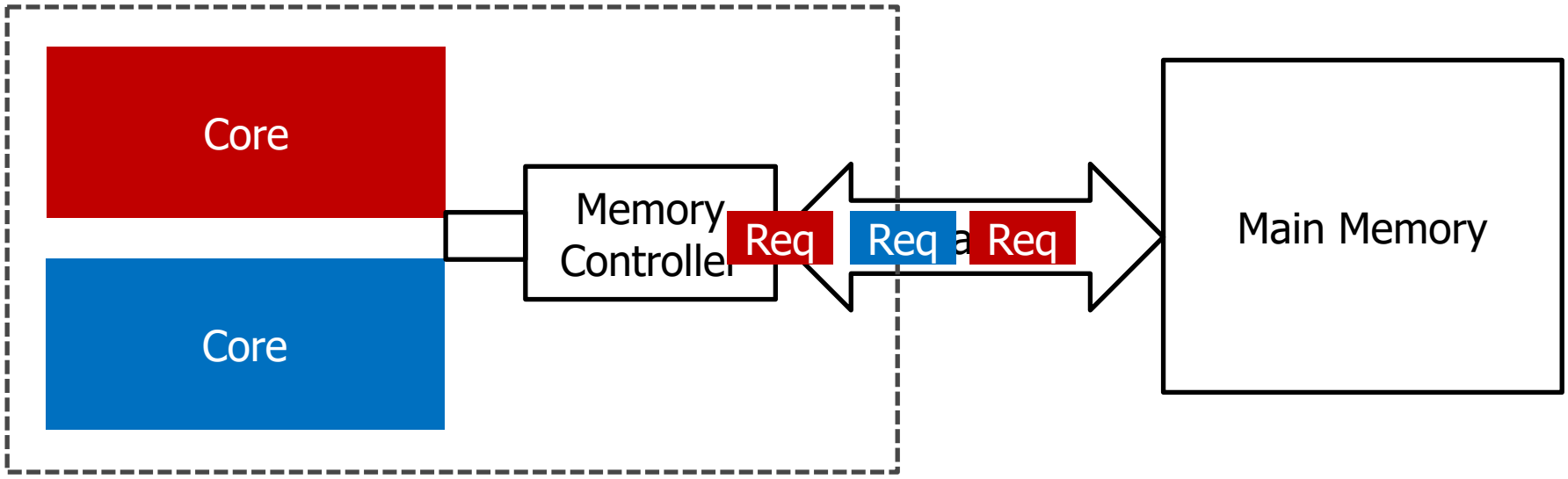
Background, Problem & Goal

Main Memory is a Bottleneck



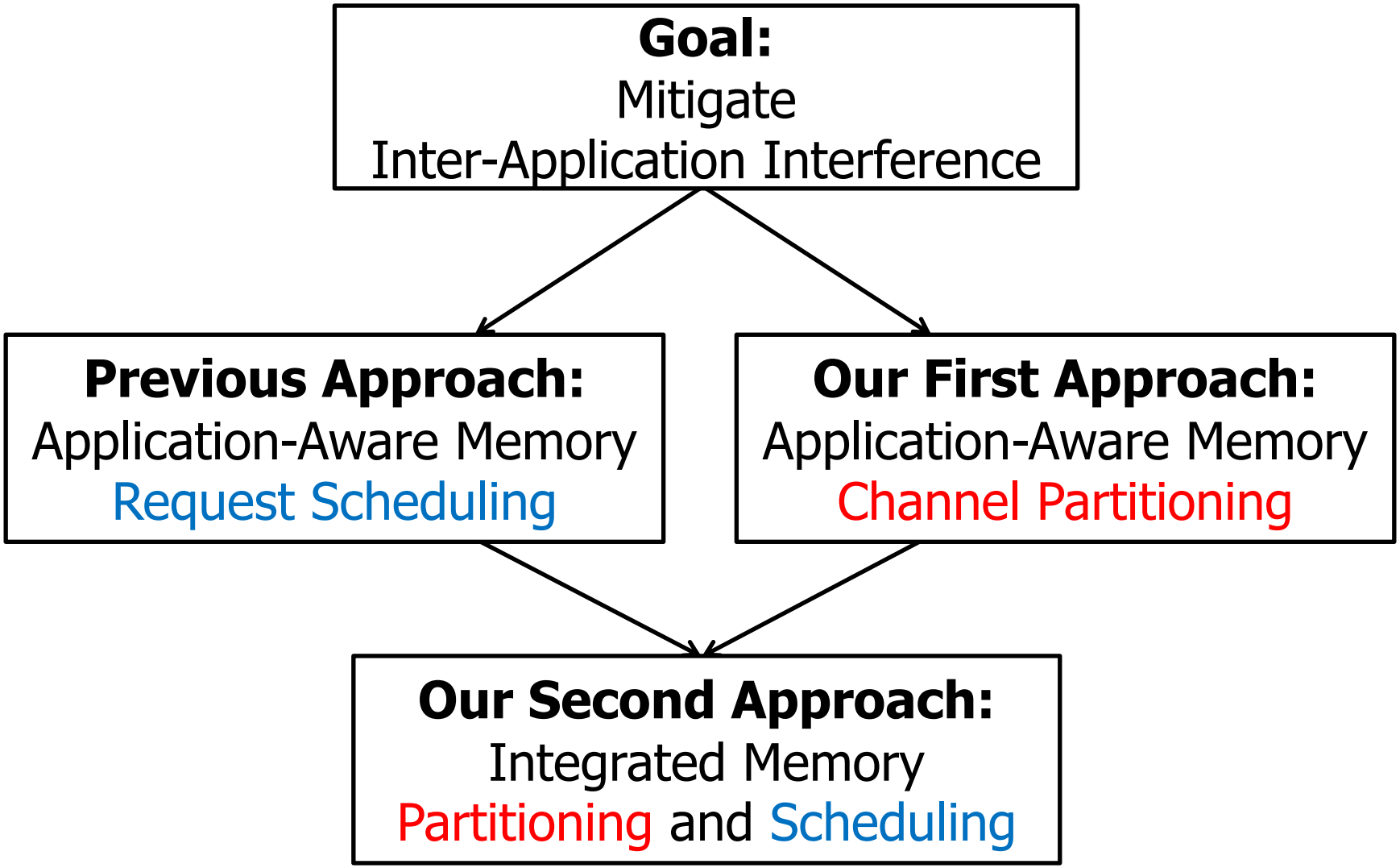
- Main memory latency is long
- Core stalls, performance degrades
- Multiple applications share the main memory

Problem of Inter-Application Interference

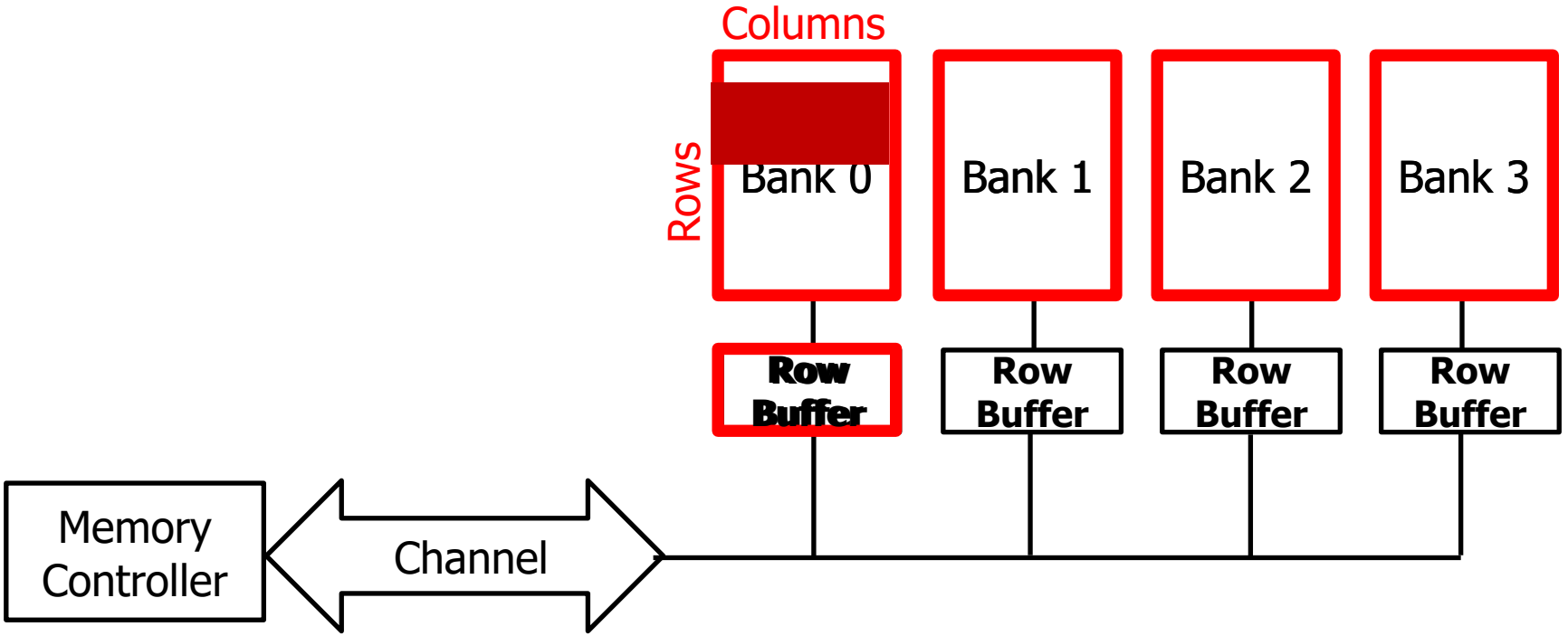


- Applications' requests interfere at the main memory
- This inter-application interference degrades system performance
- Problem further exacerbated due to
  - Increasing number of cores
  - Limited off-chip pin bandwidth

Outline


Goal:
Mitigate
Inter-Application Interference
Previous Approach:
Application-Aware Memory
Request Scheduling
Our First Approach:
Application-Aware Memory
Channel Partitioning
Our Second Approach:
Integrated Memory
Partitioning and Scheduling

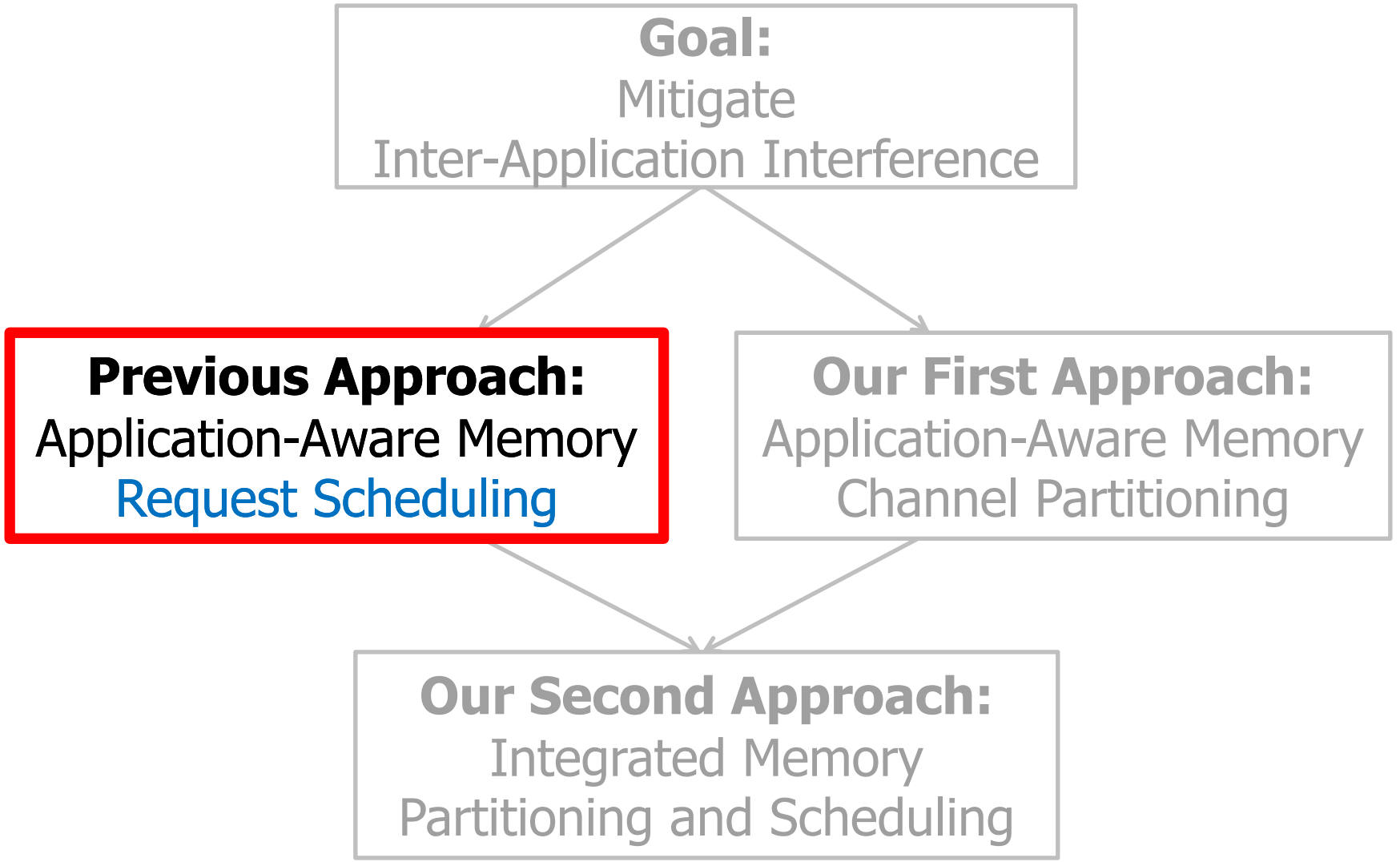
Background: Main Memory



- FR-FCFS memory scheduling policy [Zuravleff et al., US Patent '97; Rixner et al., ISCA '00]
  - Row-buffer hit first
  - Oldest request first
- Unaware of inter-application interference

Novelty

Previous Approach



Application-Aware Memory Request Scheduling

- Monitor application memory access characteristics
- Rank applications based on memory access characteristics
- Prioritize requests at the memory controller, based on ranking

An Example: Thread Cluster Memory Scheduling

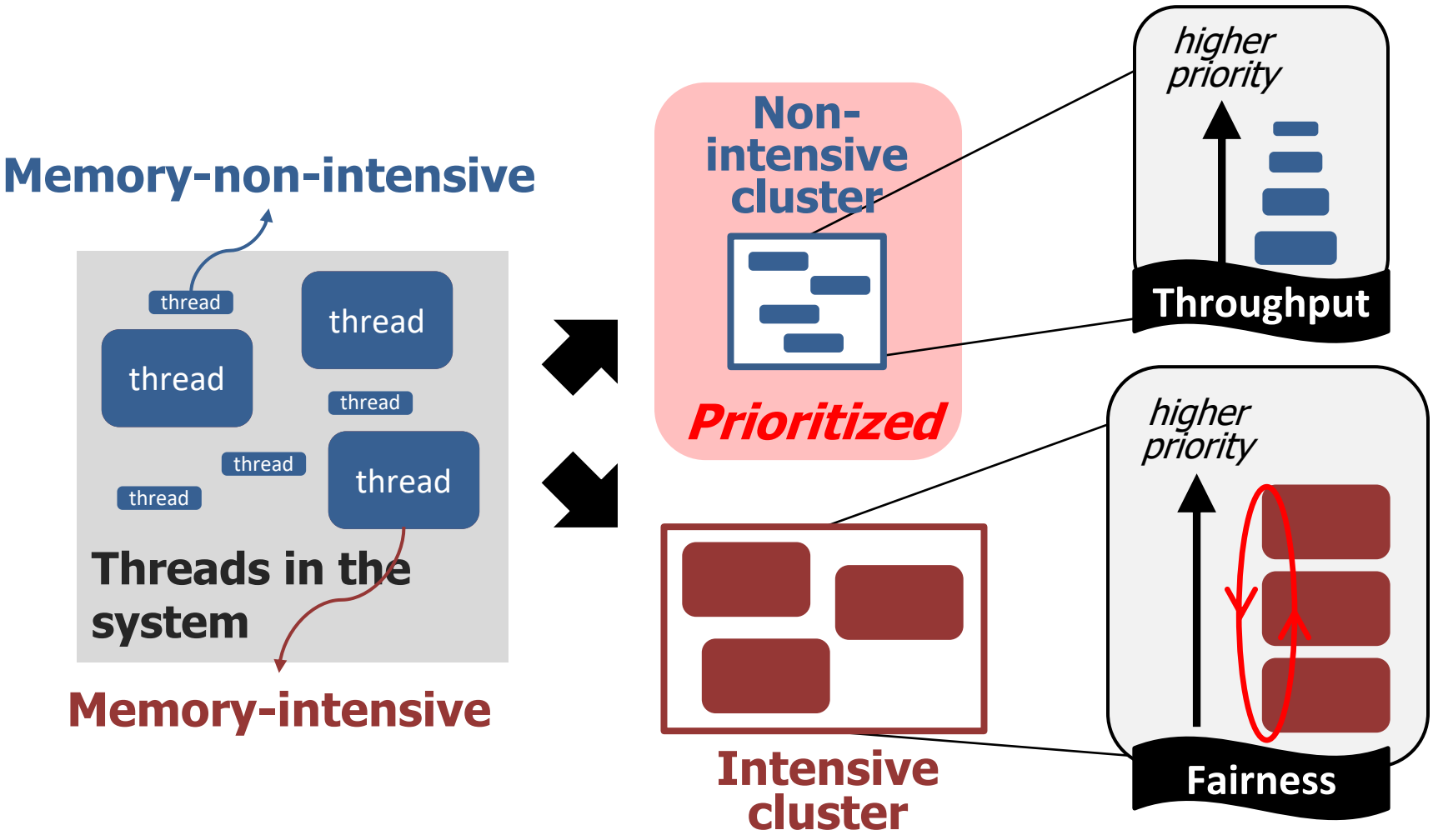


Figure: Kim et al., MICRO 2010

Application-Aware Memory Request Scheduling

Advantages

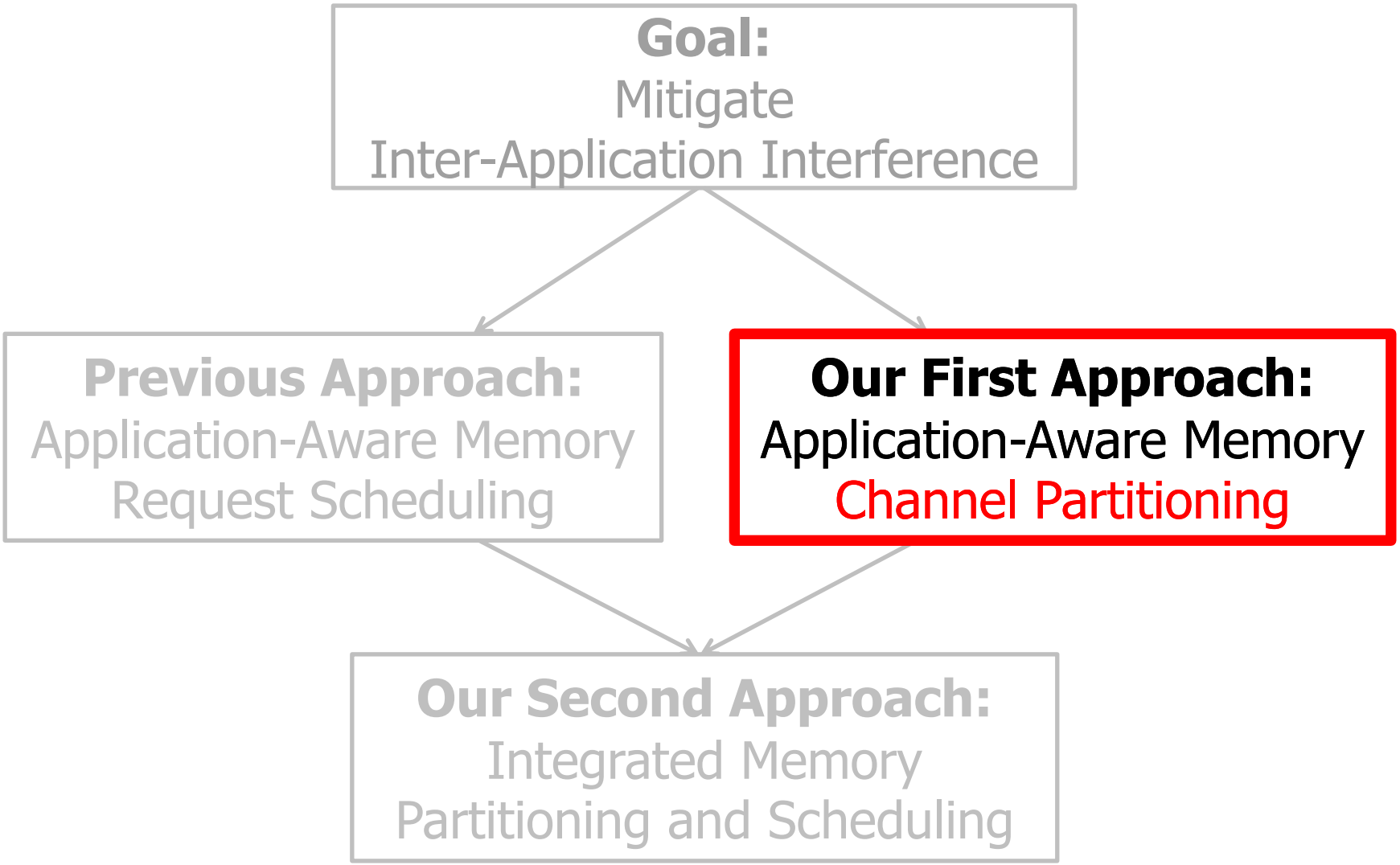
- Reduces interference between applications by request reordering
- Improves system performance

Disadvantages

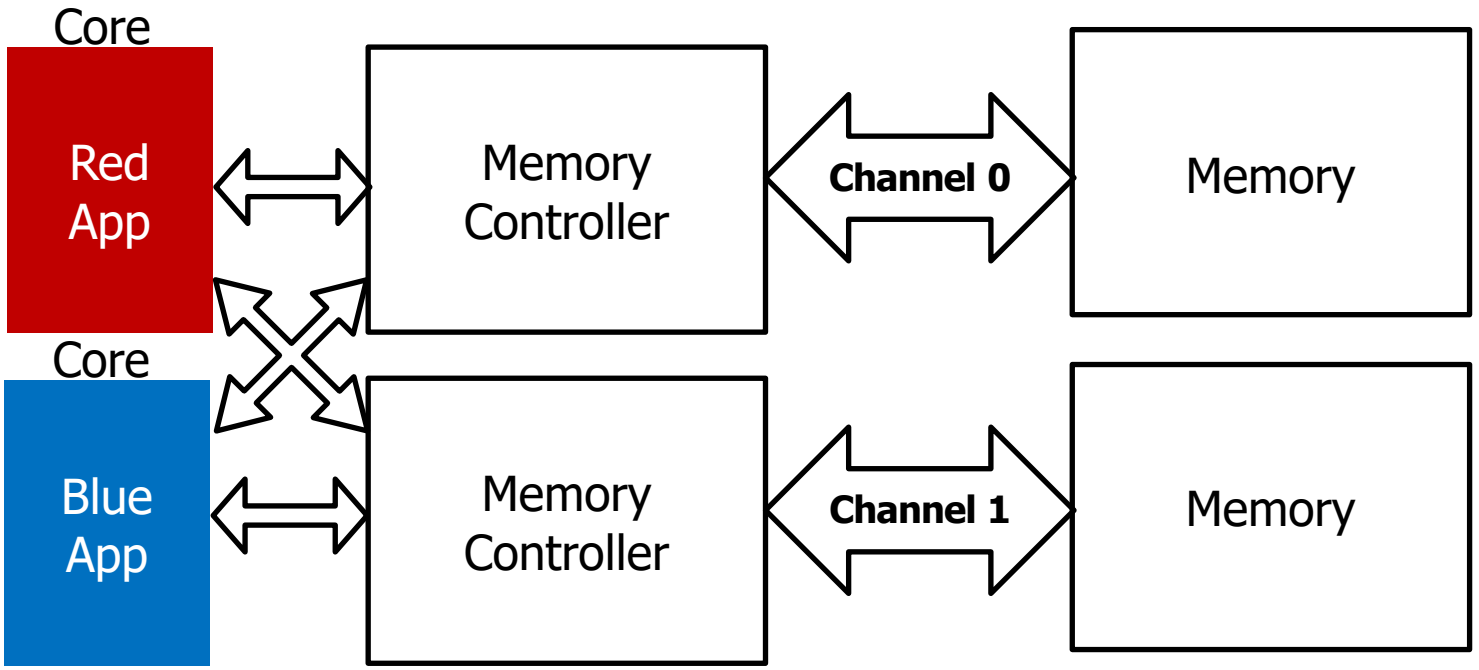
- Requires modifications to memory scheduling logic for
  - Ranking
  - Prioritization
- Cannot completely eliminate interference by request reordering

Key Approach and Ideas

The Paper's Approach



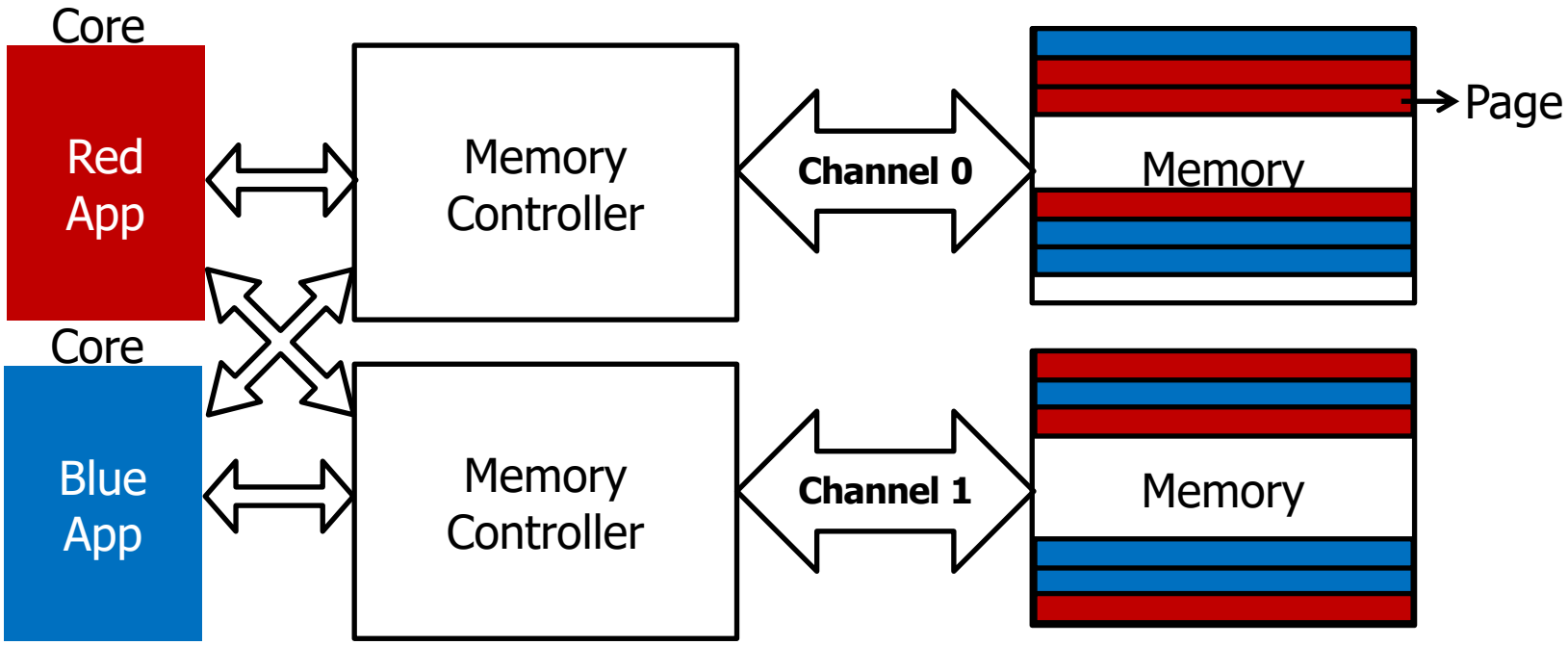
Observation: Modern Systems Have Multiple Channels



A new degree of freedom

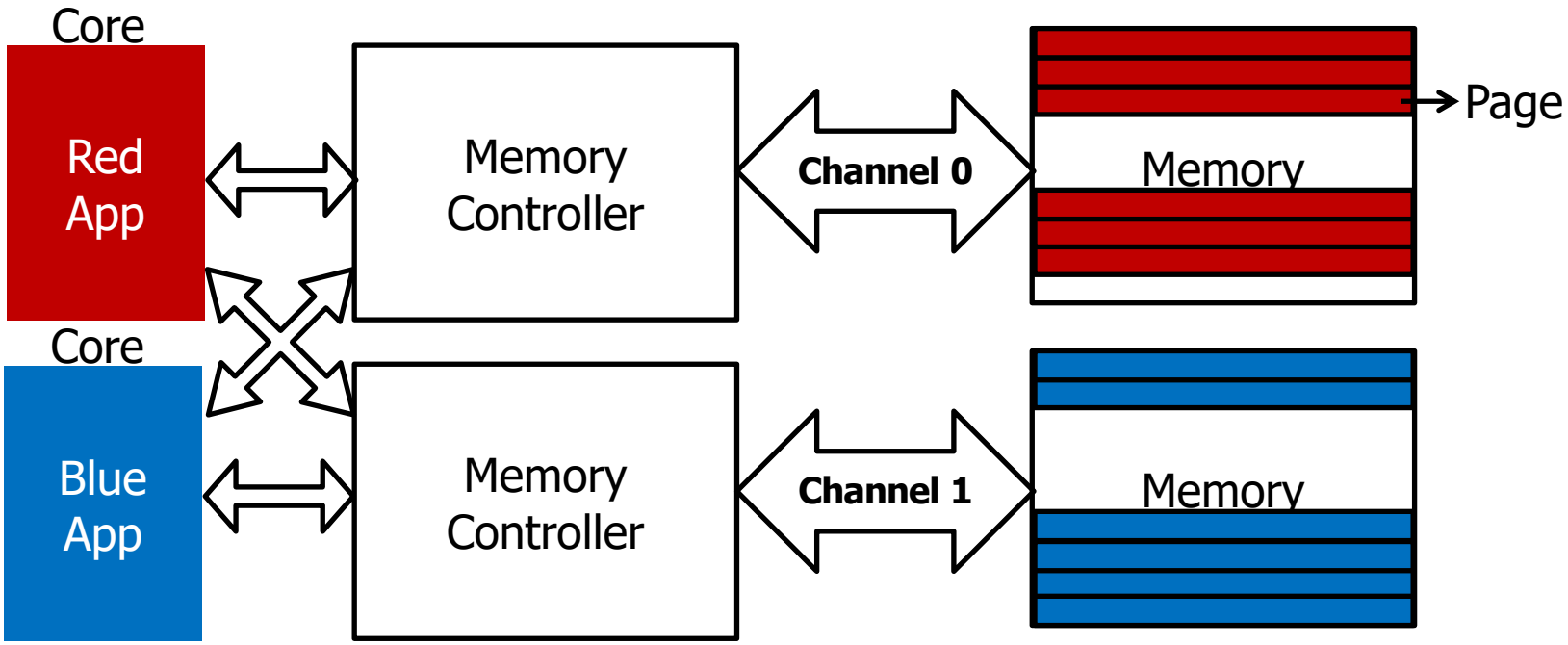
Mapping data across multiple channels

Data Mapping in Current Systems



Causes interference between applications' requests

Partitioning Channels Between Applications



Eliminates interference between applications' requests

Overview: Memory Channel Partitioning (MCP)

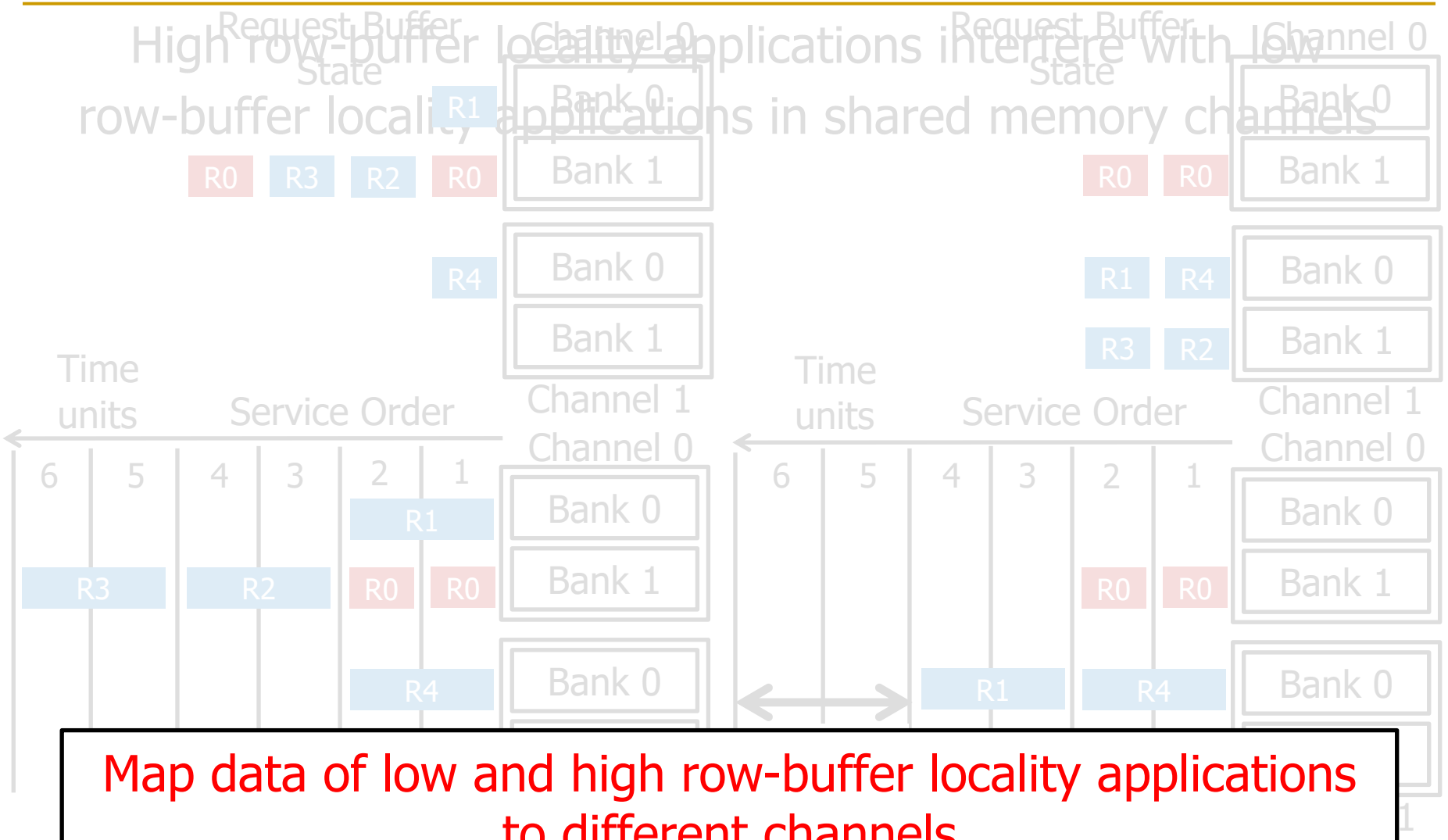
- Goal
  - Eliminate harmful interference between applications
- Basic Idea
  - Map the data of badly-interfering applications to different channels
- Key Principles
  - Separate low and high memory-intensity applications
  - Separate low and high row-buffer locality applications

Key Insight 1: Separate by Memory Intensity

High memory-intensity applications interfere with low memory-intensity applications in shared memory channels

Map data of low and high memory-intensity applications to different channels

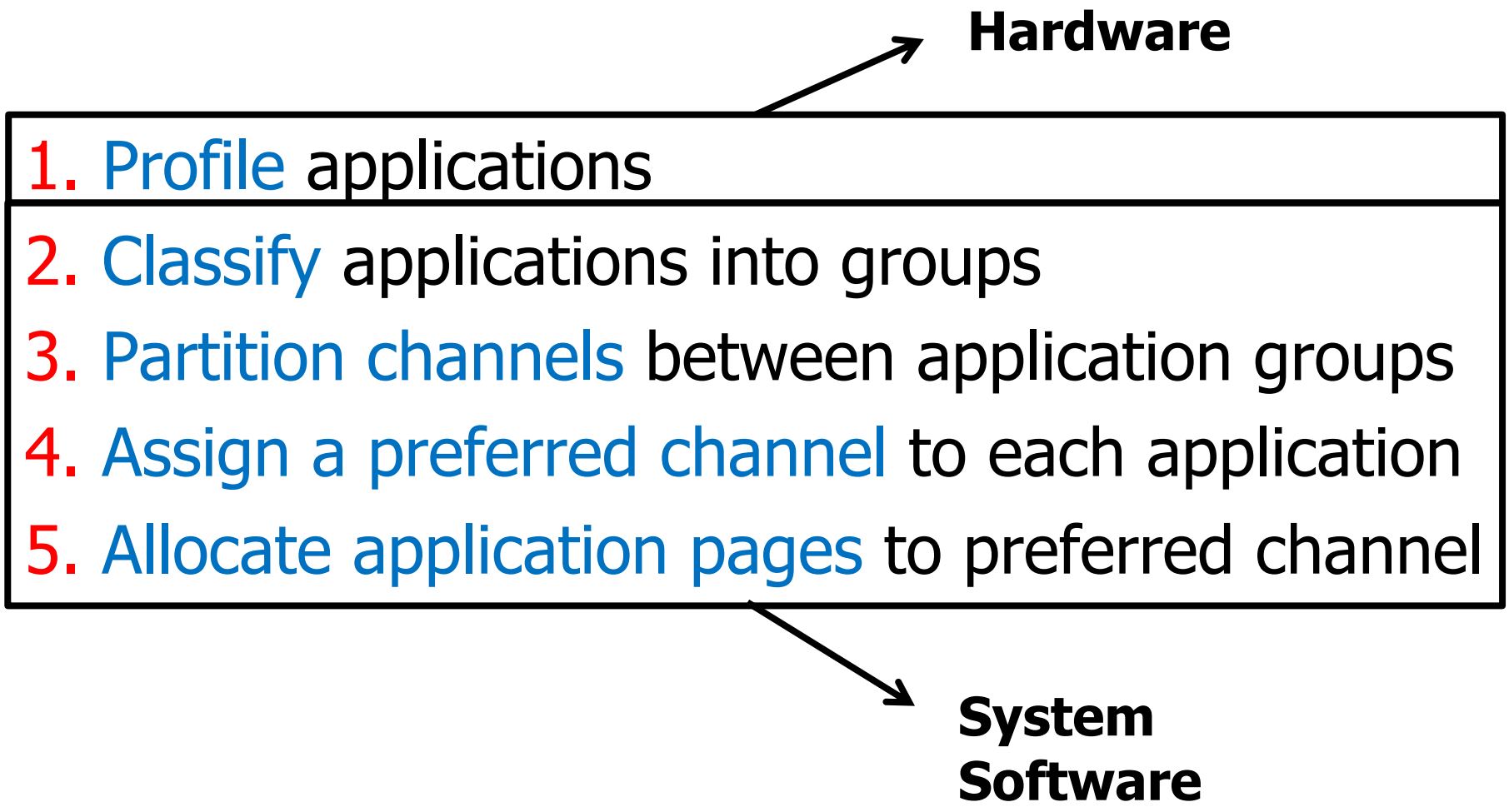
Key Insight 2: Separate by Row-Buffer Locality



Map data of low and high row-buffer locality applications to different channels

Mechanisms (in some detail)

Memory Channel Partitioning (MCP) Mechanism



1. Profile Applications

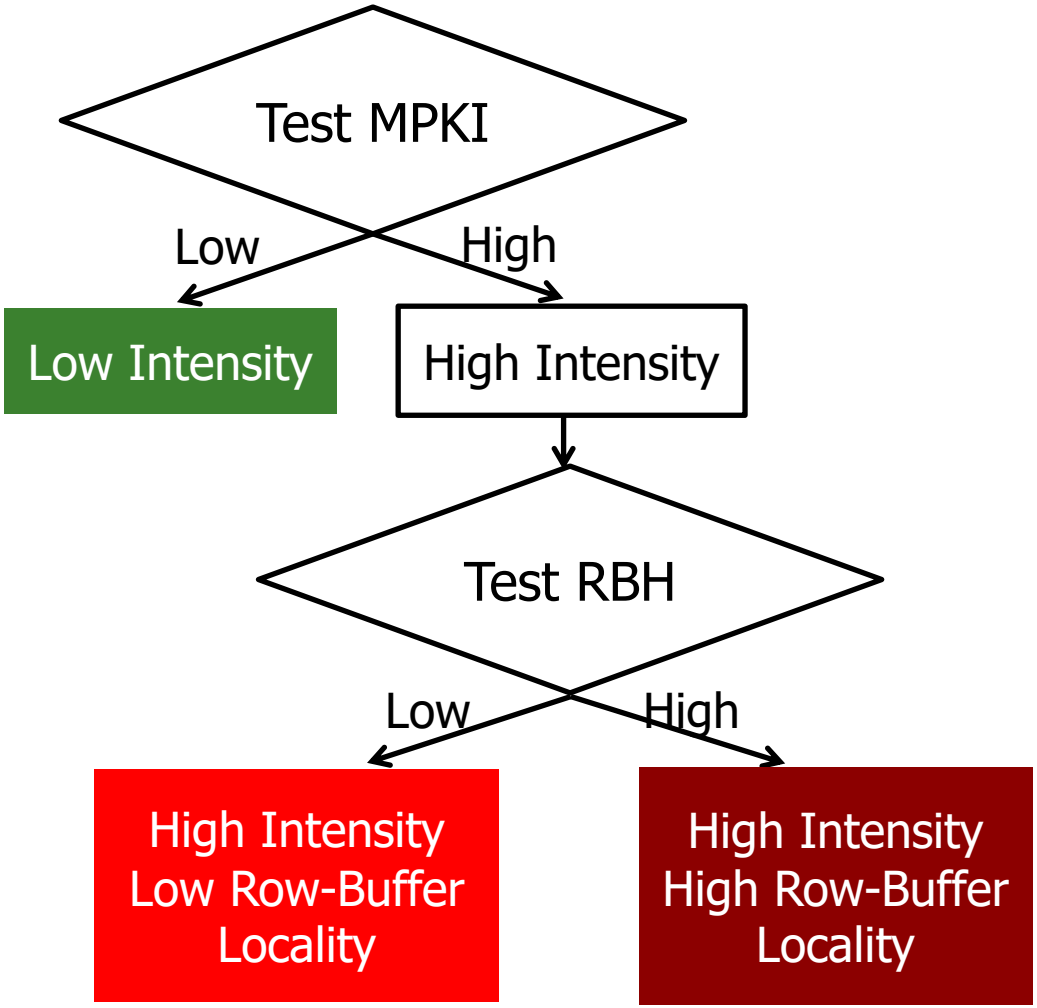
- Hardware counters collect application memory access characteristics

- Memory access characteristics
  - Memory intensity:

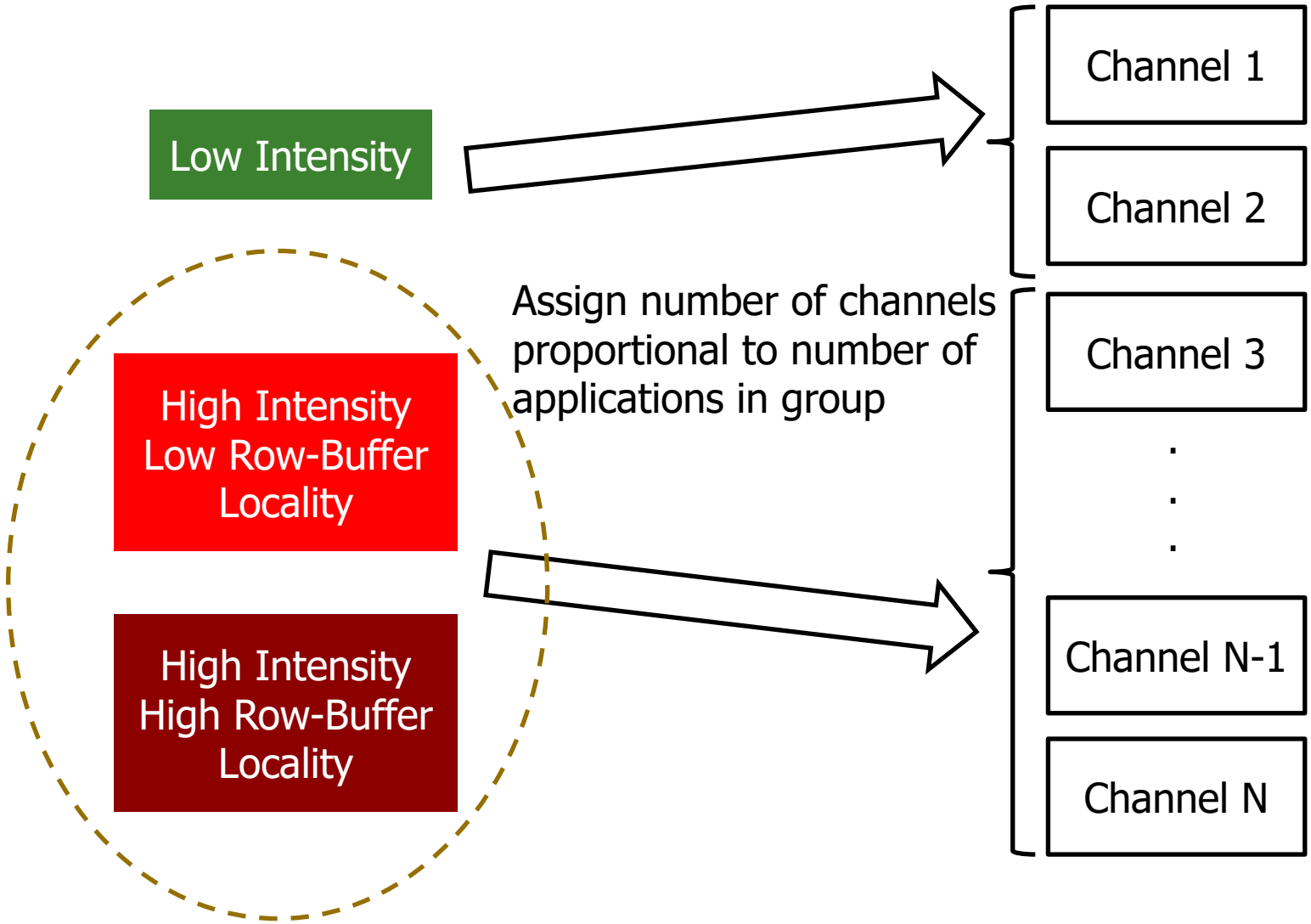
    Last level cache Misses Per Kilo Instruction (MPKI)
  - Row-buffer locality:

    Row-buffer Hit Rate (RBH) - percentage of accesses that hit in the row buffer

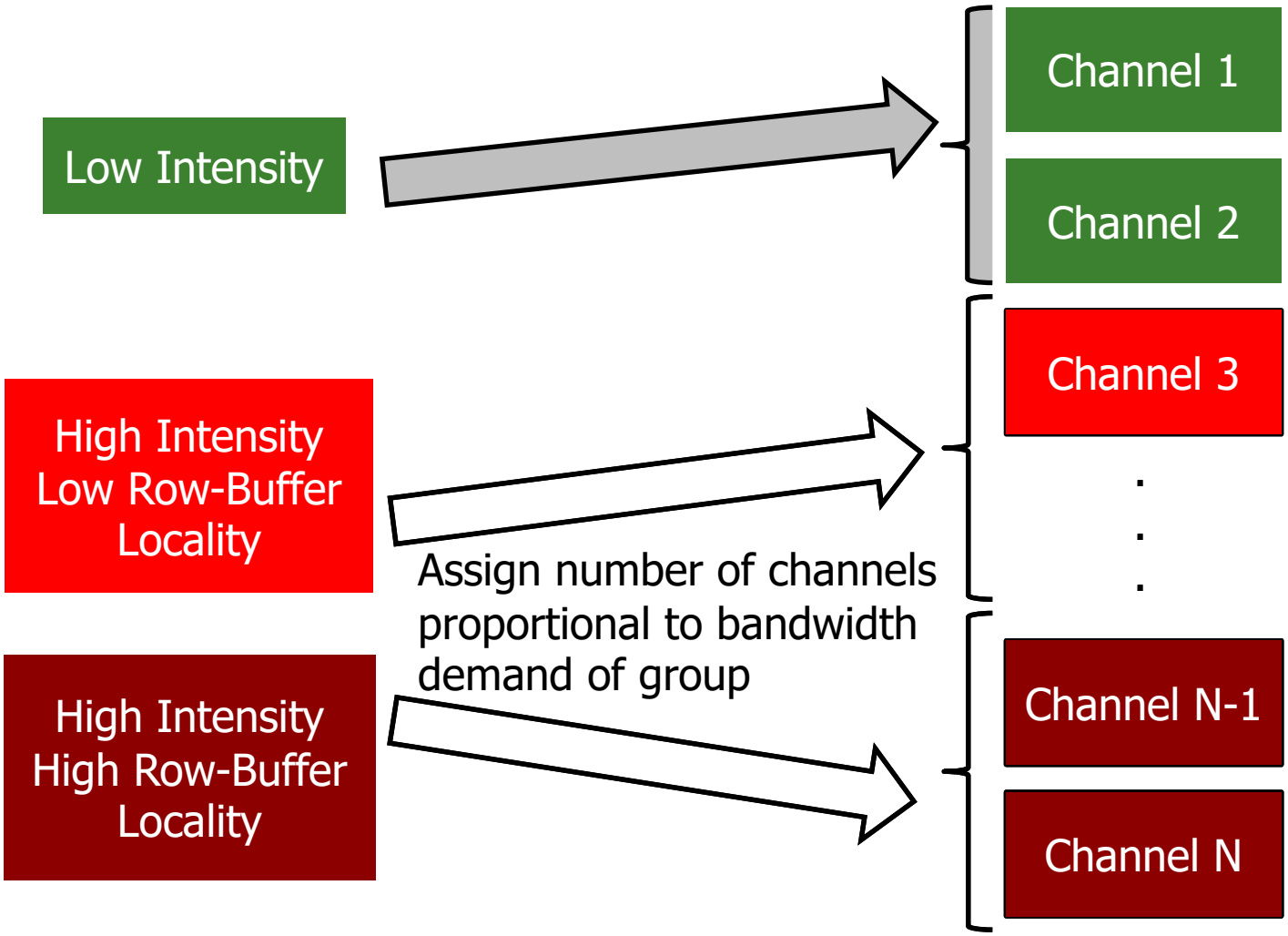
2. Classify Applications



3. Partition Channels Among Groups: Step 1

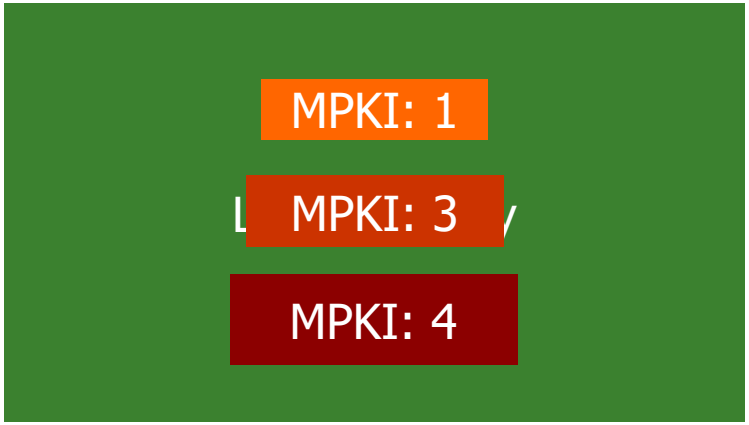


3. Partition Channels Among Groups: Step 2



4. Assign Preferred Channel to Application

- Assign each application a preferred channel from its group's allocated channels
- Distribute applications to channels such that group's bandwidth demand is balanced across its channels

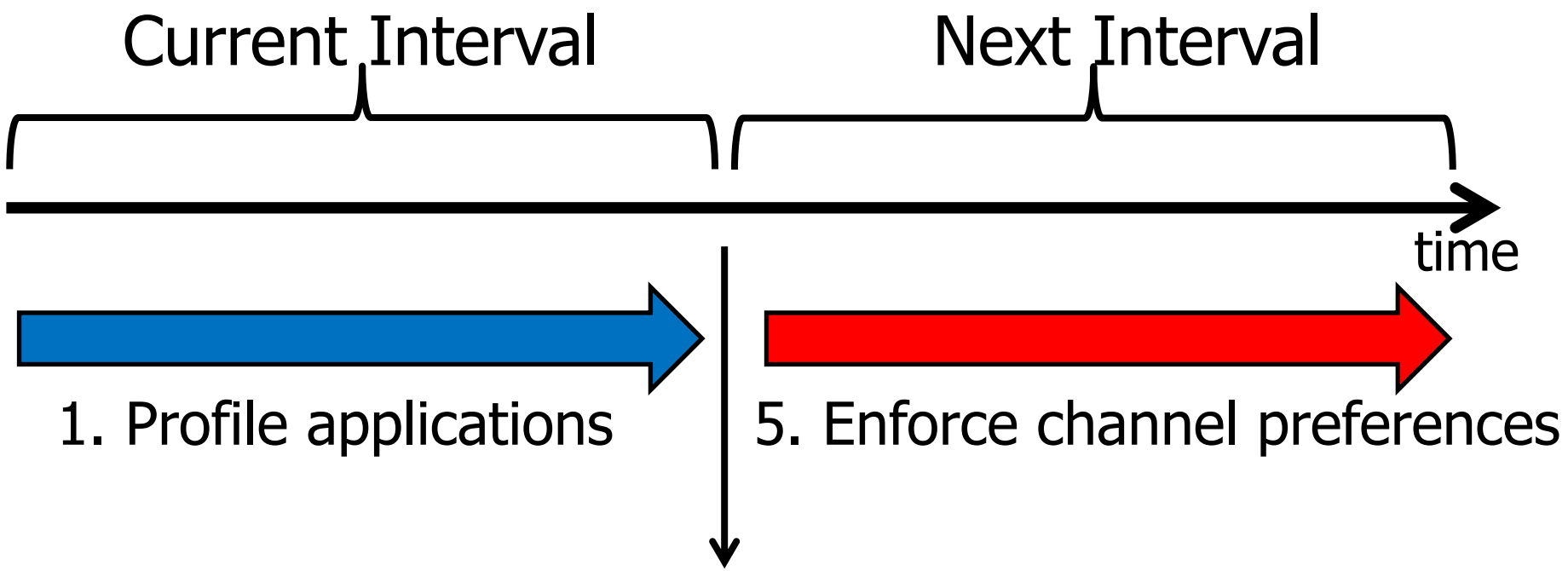


5. Allocate Page to Preferred Channel

- Enforce channel preferences
  computed in the previous step

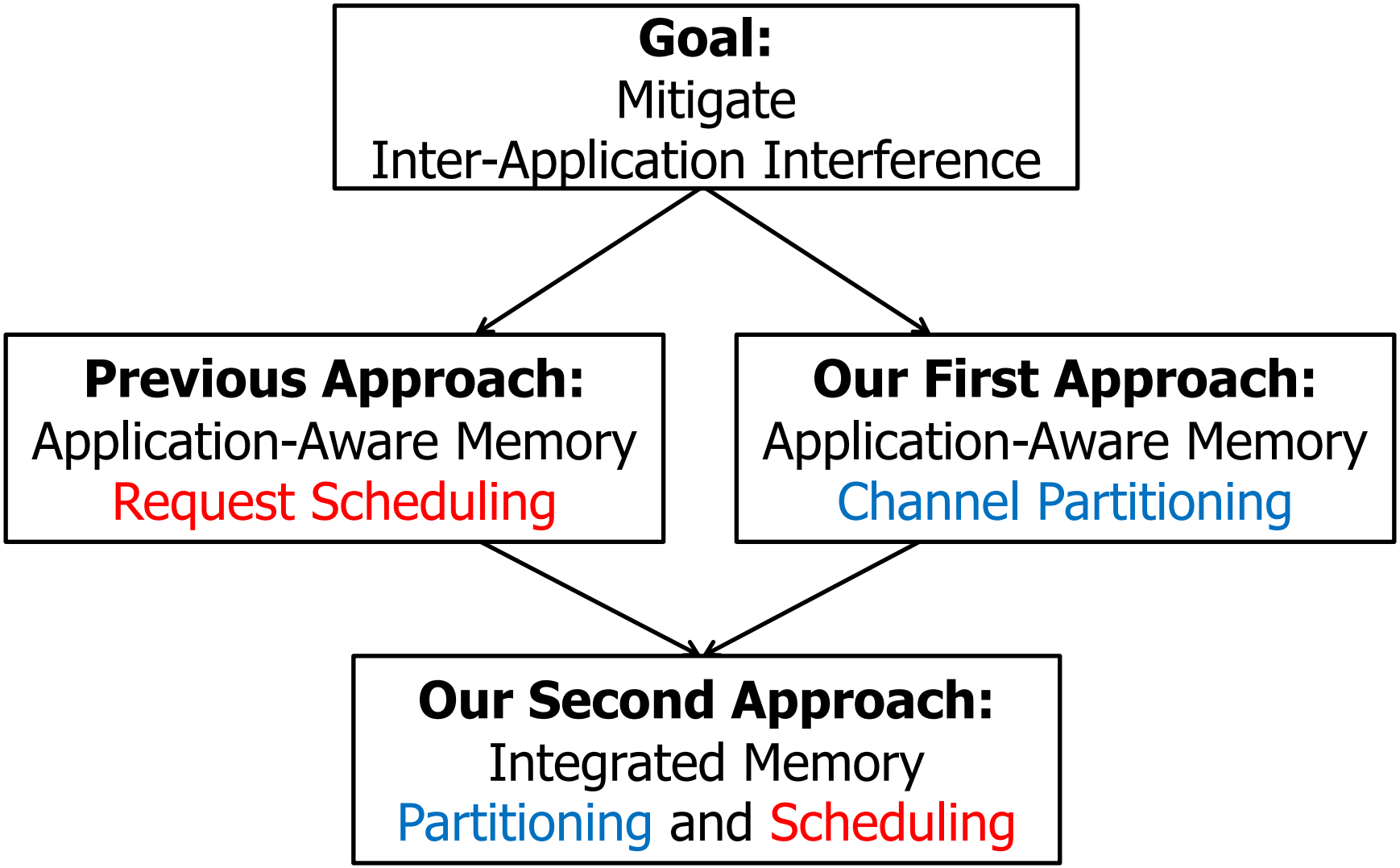
- On a page fault, the operating system
  - allocates page to preferred channel if free page available in preferred channel
  - if free page not available, replacement policy tries to allocate page to preferred channel
  - if it fails, allocate page to another channel

Interval Based Operation



2. Classify applications into groups
3. Partition channels between groups
4. Assign preferred channel to applications

Integrating Partitioning and Scheduling



Observations

- Applications with very low memory-intensity rarely access memory
  → Dedicating channels to them results in precious memory bandwidth waste
- They have the most potential to keep their cores busy
  → We would really like to prioritize them
- They interfere minimally with other applications
  → Prioritizing them does not hurt others

Integrated Memory Partitioning and Scheduling (IMPS)

- Always prioritize very low memory-intensity applications in the memory scheduler

- Use memory channel partitioning to mitigate interference between other applications

Key Results: Methodology and Evaluation

Hardware Cost

- Memory Channel Partitioning (MCP)
  - Only profiling counters in hardware
  - No modifications to memory scheduling logic
  - 1.5 KB storage cost for a 24-core, 4-channel system

- Integrated Memory Partitioning and Scheduling (IMPS)
  - A single bit per request
  - Scheduler prioritizes based on this single bit

Methodology

- Simulation Model
  - 24 cores, 4 channels, 4 banks/channel
  - Core Model
    - Out-of-order, 128-entry instruction window
    - 512 KB L2 cache/core
  - Memory Model – DDR2
- Workloads
  - 240 SPEC CPU 2006 multiprogrammed workloads (categorized based on memory intensity)
- Metrics
  - System Performance $Weighted\ Speedup = \sum_i \frac{IPC_i^{shared}}{IPC_i^{alone}}$

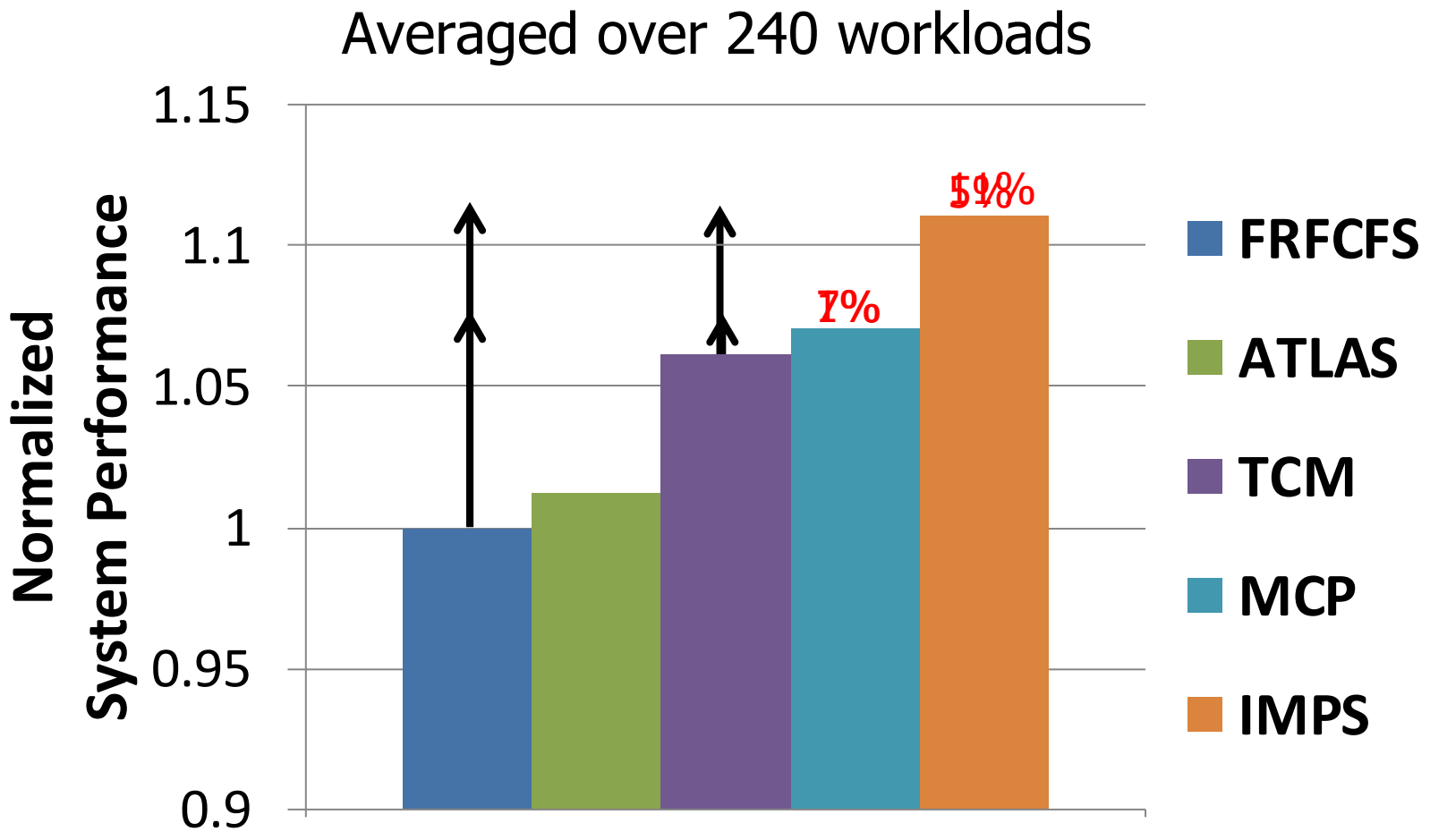
Previous Work on Memory Scheduling

- FR-FCFS [Zuravleff et al., US Patent 1997, Rixner et al., ISCA 2000]
  - Prioritizes row-buffer hits and older requests
  - Application-unaware

- ATLAS [Kim et al., HPCA 2010]
  - Prioritizes applications with low memory-intensity

- TCM [Kim et al., MICRO 2010]
  - Always prioritizes low memory-intensity applications
  - Shuffles request priorities of high memory-intensity applications

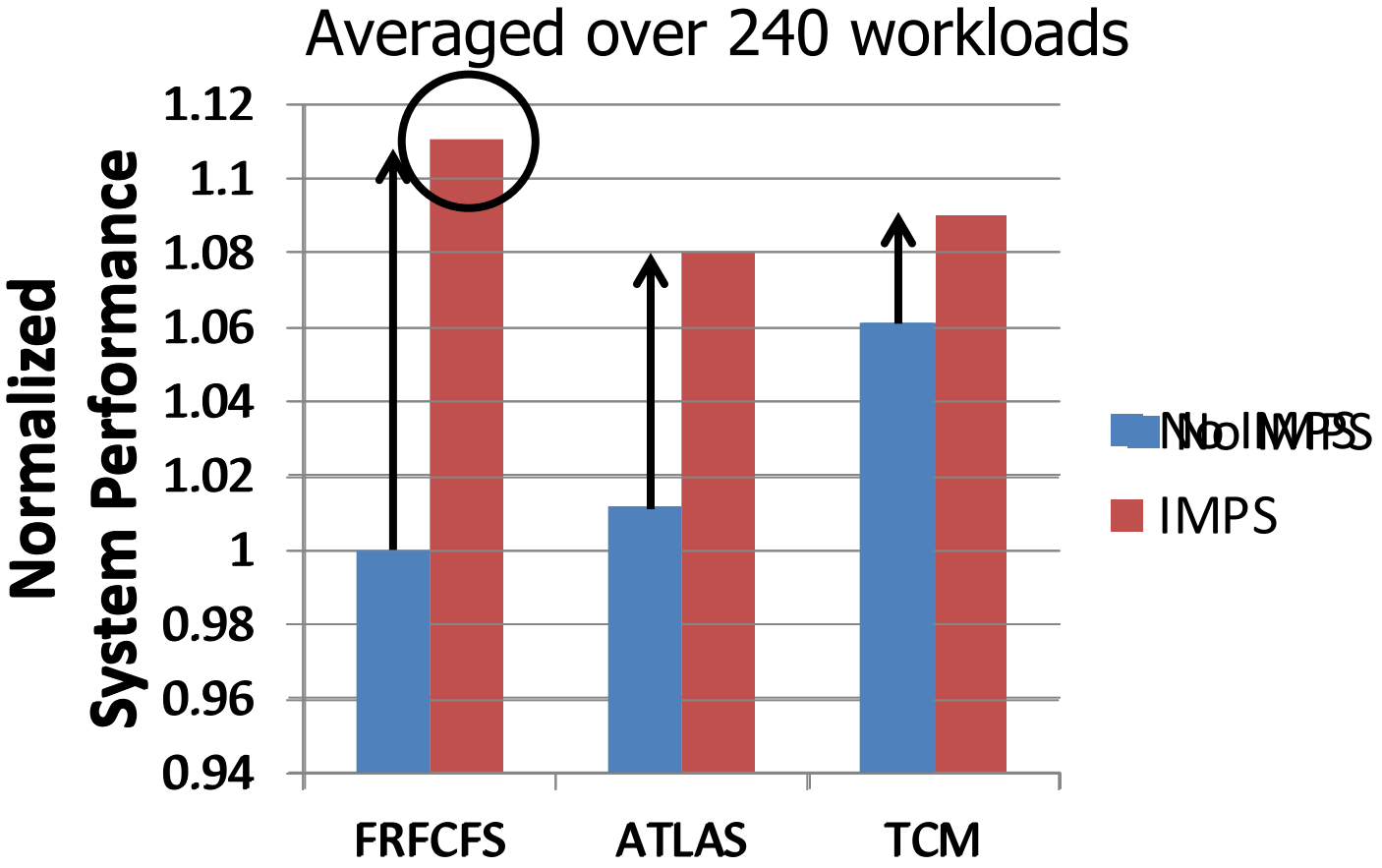
Comparison to Previous Scheduling Policies



Better system performance than the best previous scheduler

Significant performance improvement over baseline FRFCFS at lower hardware cost

Interaction with Memory Scheduling



IMPS improves performance regardless of scheduling policy

Highest improvement over FRFCFS as IMPS designed for FRFCFS

Summary

Summary

- Uncontrolled inter-application interference in main memory degrades system performance

- Application-aware memory channel partitioning (MCP)
  - Separates the data of badly-interfering applications to different channels, eliminating interference

- Integrated memory partitioning and scheduling (IMPS)
  - Prioritizes very low memory-intensity applications in scheduler
  - Handles other applications' interference by partitioning

- MCP/IMPS provide better performance than application-aware memory request scheduling at lower hardware cost

We Did Not Cover The Rest of the Slides. They Are For Your Benefit.

Seminar in Computer Architecture

Meeting 2b: Example Review II

Prof. Onur Mutlu

ETH Zürich

Fall 2019

26 September 2019

Strengths

Strengths

Strengths of the Paper

- Novel solution to a key problem in multi-core systems, memory interference; the importance of problem will increase over time
- Keeps the memory scheduling hardware simple
- Combines multiple interference reduction techniques
- Can provide performance isolation across applications mapped to different channels
- General idea of partitioning can be extended to smaller granularities in the memory hierarchy: banks, subarrays, etc.

- Well-written paper
- Thorough simulation-based evaluation

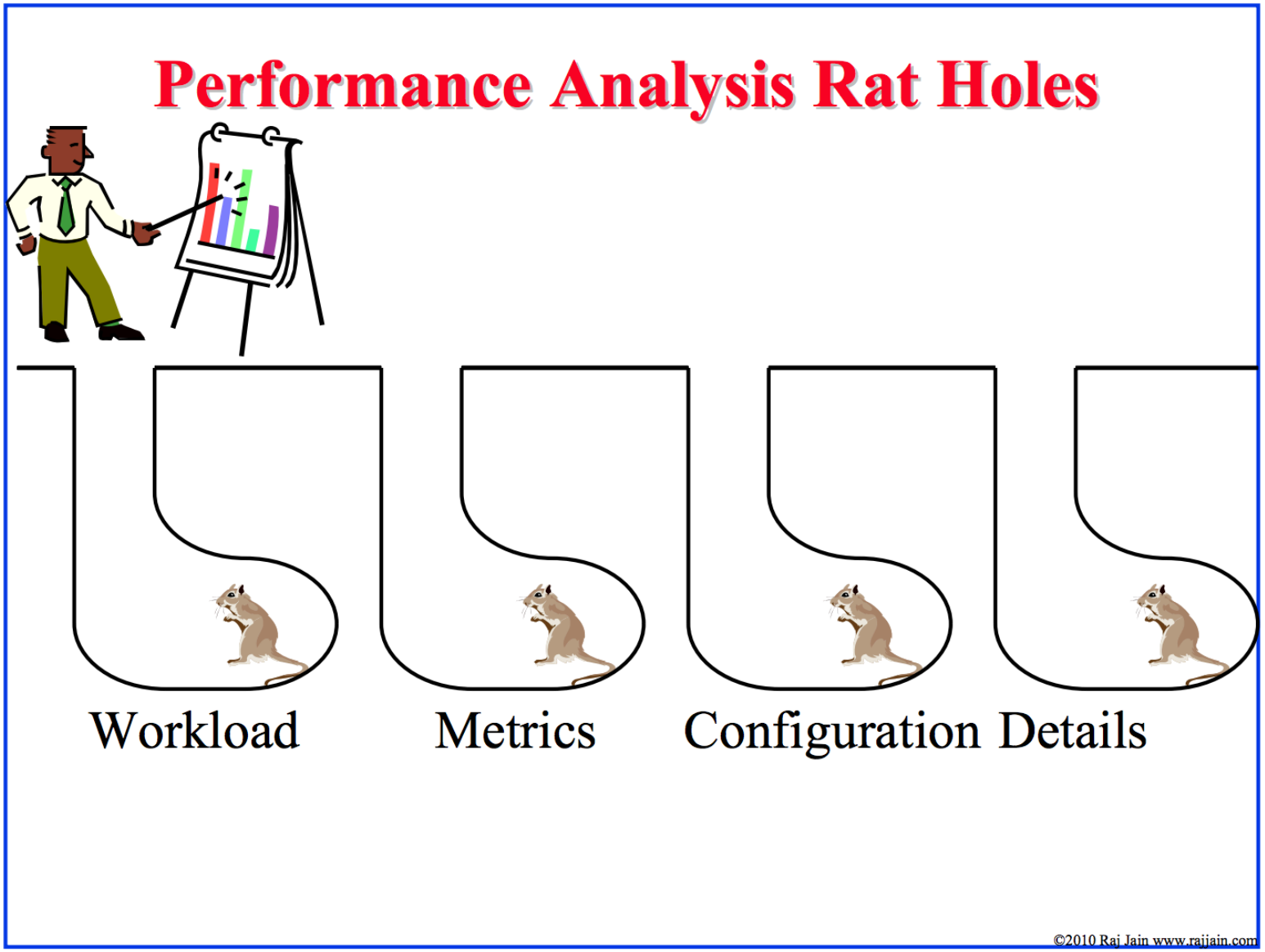
Weaknesses

Weaknesses/Limitations of the Paper

- Mechanism may not work effectively if workload changes behavior after profiling
- Overhead of moving pages between channels restricts mechanism's benefits
- Small number of memory channels reduces the scope of partitioning
- Load imbalance across channels can reduce performance
  - The paper addresses this and compares to another mechanism

- Software-hardware cooperative solution might not always be easy to adopt
- Evaluation is done solely in simulation
- Evaluation does not consider multi-chip systems
- Are these the best workloads to evaluate?

Recall: Try to Avoid Rat Holes



Source: https://www.cse.wustl.edu/~jain/iucee/ftp/k_10adp.pdf

Thoughts and Ideas

Extensions

- Can this idea be extended to different granularities in memory?
  - Partition banks, subarrays, mats across workloads
- Can this idea be extended to provide performance predictability and performance isolation? How?
- How can MCP be combined effectively with other interference reduction techniques?
  - E.g., source throttling methods [Ebrahimi+, ASPLOS 2010]
  - E.g., thread scheduling methods
- Can this idea be evaluated on a real system? How?

Takeaways

Key Takeaways

- A novel method to reduce memory interference
- Simple and effective
- Hardware/software cooperative
- Good potential for work building on it to extend it
  - To different structures
  - To different metrics
  - Multiple works have already built on the paper (see bank partitioning works in PACT 2012, HPCA 2012)
- Easy to read and understand paper

Open Discussion

Discussion Starters

- Thoughts on the previous ideas?
- How practical is this?
- Will the problem become bigger and more important over time?
- Will the solution become more important over time?
- Are other solutions better?
- Is this solution clearly advantageous in some cases?

Seminar in Computer Architecture

Meeting 3a: Example Review II

Prof. Onur Mutlu

ETH Zürich

Spring 2020

12 March 2020