Computer Architecture

Lecture 18a: Memory Interference and Quality of Service III

Prof. Onur Mutlu
ETH Zürich

Fall 2018

22 November 2018

Lecture Announcement

- Monday, November 26, 2018
- 16:15-17:15
- CAB G 61
- Apéro after the lecture ☺

- Prof. Arvind (Massachusetts Institute of Technology)
- D-INFK Distinguished Colloquium
- **The Riscy Expedition**

- https://www.inf.ethz.ch/news-and-events/colloquium/event-detail.html?eventFeedId=42658

Fundamental Interference Control Techniques

- Goal: to reduce/control inter-thread memory interference

1. Prioritization or request scheduling

2. Data mapping to banks/channels/ranks

3. Core/source throttling

4. Application/thread scheduling

Fairness via Source Throttling

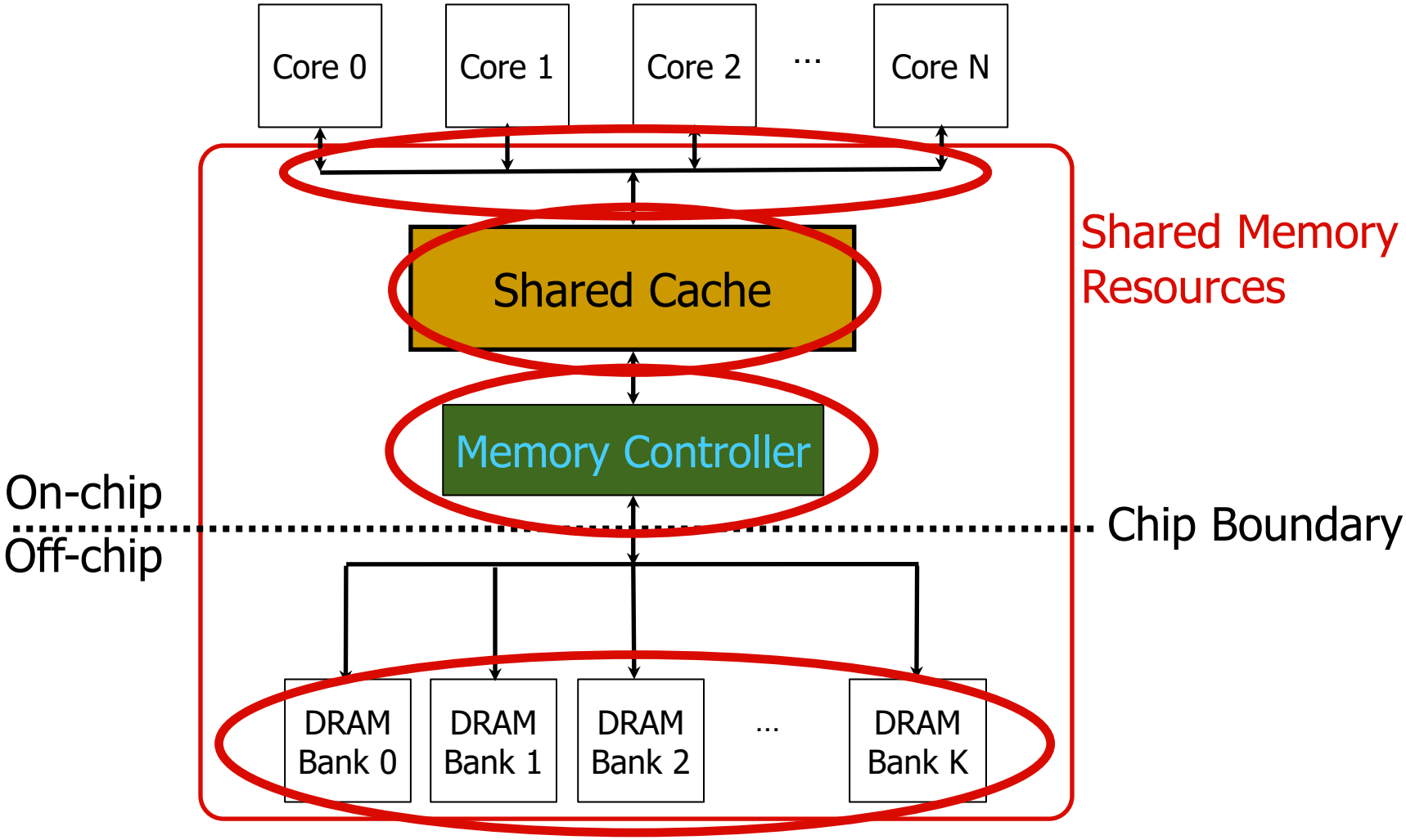
Eiman Ebrahimi, Chang Joo Lee, Onur Mutlu, and Yale N. Patt,

"Fairness via Source Throttling: A Configurable and High-Performance Fairness Substrate for Multi-Core Memory Systems"

15th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (***ASPLOS***), pages 335-346, Pittsburgh, PA, March 2010. Slides (pdf)

FST ASPLOS 2010 Talk

Many Shared Resources



The Problem with "Smart Resources"

- Independent interference control mechanisms in caches, interconnect, and memory can contradict each other

- Explicitly coordinating mechanisms for different resources requires complex implementation

- How do we enable fair sharing of the entire memory system by controlling interference in a coordinated manner?

Source Throttling: A Fairness Substrate

- Key idea: Manage inter-thread interference at the cores (sources), not at the shared resources

- Dynamically estimate unfairness in the memory system
- Feed back this information into a controller
- Throttle cores' memory access rates accordingly
  - Whom to throttle and by how much depends on performance target (throughput, fairness, per-thread QoS, etc)
  - E.g., if unfairness > system-software-specified target then throttle down core causing unfairness & throttle up core that was unfairly treated

- Ebrahimi et al., "Fairness via Source Throttling," ASPLOS'10, TOCS'12.

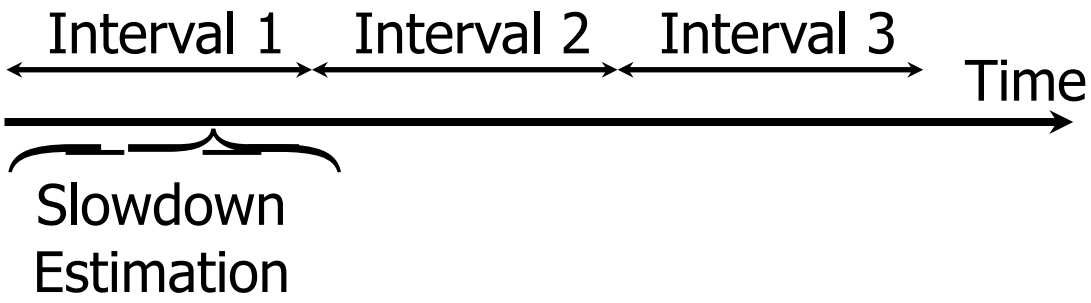
Fairness via Source Throttling (FST)

- Two components (interval-based)

- Run-time unfairness evaluation (in hardware)
  - Dynamically estimates the unfairness (application slowdowns) in the memory system
  - Estimates which application is slowing down which other

- Dynamic request throttling (hardware or software)
  - Adjusts how aggressively each core makes requests to the shared resources
  - Throttles down request rates of cores causing unfairness
    - Limit miss buffers, limit injection rate

Fairness via Source Throttling (FST) [ASPLOS'10]



FST

1- Estimating system unfairness
2- Find app. with the highest slowdown (App-slowest)
3- Find app. causing most interference for App-slowest (App-interfering)

```
if (Unfairness Estimate >Target)
{
 1-Throttle down App-interfering
   (limit injection rate and parallelism)
 2-Throttle up App-slowest
}
```

Dynamic Request Throttling

- Goal: Adjust how aggressively each core makes requests to the shared memory system

- Mechanisms:
  - Miss Status Holding Register (MSHR) quota
    - Controls the number of concurrent requests accessing shared resources from each application
  - Request injection frequency
    - Controls how often memory requests are issued to the last level cache from the MSHRs

Dynamic Request Throttling

- Throttling level assigned to each core determines both MSHR quota and request injection rate

Throttling level	MSHR quota	Request Injection Rate
100%	128	Every cycle
50%	64	Every other cycle
25%	32	Once every 4 cycles
10%	12	Once every 10 cycles
5%	6	Once every 20 cycles
4%	5	Once every 25 cycles
3%	3	Once every 30 cycles
2%	2	Once every 50 cycles

Total # of MSHRs: 128

System Software Support

- Different fairness objectives can be configured by system software
  - Keep maximum slowdown in check
    - Estimated Max Slowdown < Target Max Slowdown
  - Keep slowdown of particular applications in check to achieve a particular performance target
    - Estimated Slowdown(i) < Target Slowdown(i)

- Support for thread priorities
  - Weighted Slowdown(i) =
    Estimated Slowdown(i) x Weight(i)

Source Throttling Results: Takeaways

- Source throttling alone provides better performance than a combination of "smart" memory scheduling and fair caching
  - Decisions made at the memory scheduler and the cache sometimes contradict each other
- Neither source throttling alone nor "smart resources" alone provides the best performance
- Combined approaches are even more powerful
  - Source throttling and resource-based interference control

Source Throttling: Ups and Downs

- Advantages
  + Core/request throttling is easy to implement: no need to change the memory scheduling algorithm
  + Can be a general way of handling shared resource contention
  + Can reduce overall load/contention in the memory system

- Disadvantages
  - Requires slowdown estimations → difficult to estimate
  - Thresholds can become difficult to optimize
    → throughput loss due to too much throttling
    → can be difficult to find an overall-good configuration

More on Source Throttling (I)

- Eiman Ebrahimi, Chang Joo Lee, Onur Mutlu, and Yale N. Patt, **"Fairness via Source Throttling: A Configurable and High-Performance Fairness Substrate for Multi-Core Memory Systems"**
  *Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, pages 335-346, Pittsburgh, PA, March 2010.
  Slides (pdf)

Fairness via Source Throttling: A Configurable and High-Performance Fairness Substrate for Multi-Core Memory Systems

Eiman Ebrahimi† Chang Joo Lee† Onur Mutlu§ Yale N. Patt†

†Department of Electrical and Computer Engineering
The University of Texas at Austin
{ebrahimi, cjlee, patt}@ece.utexas.edu

§Computer Architecture Laboratory (CALCM)
Carnegie Mellon University
onur@cmu.edu

More on Source Throttling (II)

- Kevin Chang, Rachata Ausavarungnirun, Chris Fallin, and Onur Mutlu, **"HAT: Heterogeneous Adaptive Throttling for On-Chip Networks"**
  Proceedings of the 24th International Symposium on Computer Architecture and High Performance Computing (***SBAC-PAD***), New York, NY, October 2012. Slides (pptx) (pdf)

HAT: Heterogeneous Adaptive Throttling for On-Chip Networks

Kevin Kai-Wei Chang, Rachata Ausavarungnirun, Chris Fallin, Onur Mutlu
Carnegie Mellon University
{kevincha,rachata,cfallin,onur}@cmu.edu

More on Source Throttling (III)

- George Nychis, Chris Fallin, Thomas Moscibroda, Onur Mutlu, and Srinivasan Seshan,
  "On-Chip Networks from a Networking Perspective: Congestion and Scalability in Many-core Interconnects"
  *Proceedings of the 2012 ACM SIGCOMM Conference (**SIGCOMM**)*, Helsinki, Finland, August 2012. Slides (pptx)

On-Chip Networks from a Networking Perspective: Congestion and Scalability in Many-Core Interconnects

George Nychis†, Chris Fallin†, Thomas Moscibroda§, Onur Mutlu†, Srinivasan Seshan†

† Carnegie Mellon University
{gnychis,cfallin,onur,srini}@cmu.edu

§ Microsoft Research Asia
moscitho@microsoft.com

Fundamental Interference Control Techniques

- Goal: to reduce/control interference

1. Prioritization or request scheduling

2. Data mapping to banks/channels/ranks

3. Core/source throttling

4. Application/thread scheduling

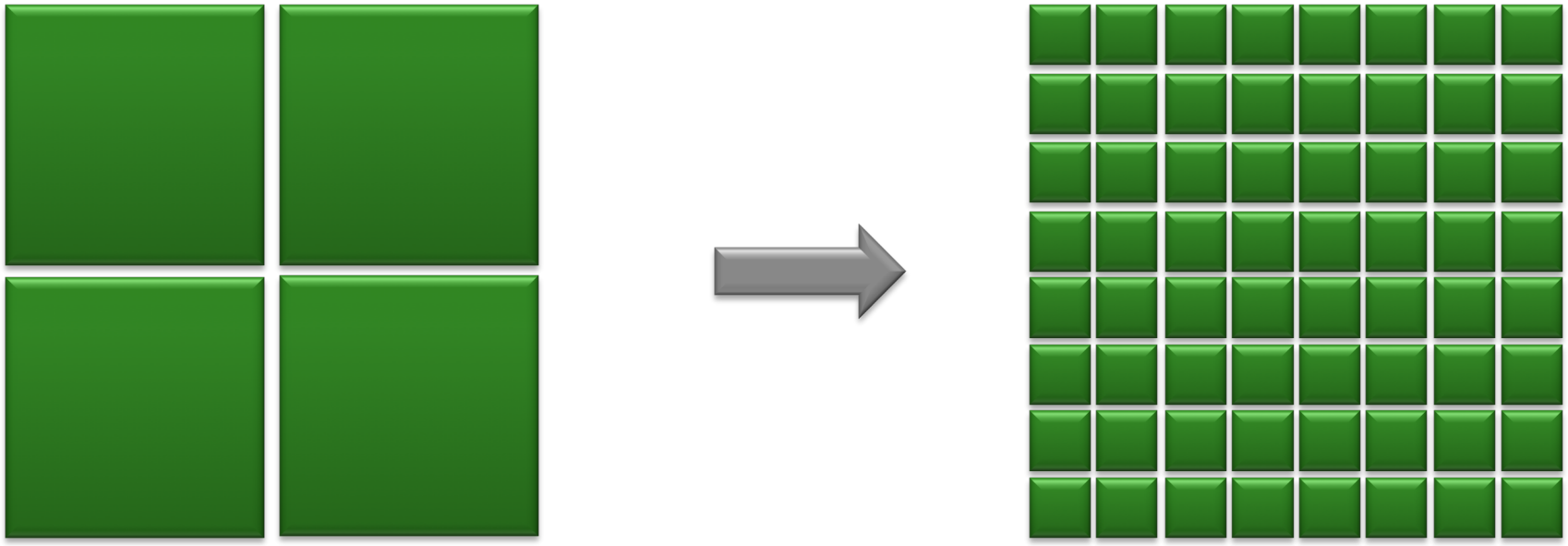
Idea: Pick threads that do not badly interfere with each other to be scheduled together on cores sharing the memory system

Application-to-Core Mapping to Reduce Interference

- Reetuparna Das, Rachata Ausavarungnirun, Onur Mutlu, Akhilesh Kumar, and Mani Azimi,
  "Application-to-Core Mapping Policies to Reduce Memory System Interference in Multi-Core Systems"
  *Proceedings of the 19th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Shenzhen, China, February 2013.
  Slides (pptx)

- Key ideas:
  - Cluster threads to memory controllers (to reduce across chip interference)
  - Isolate interference-sensitive (low-intensity) applications in a separate cluster (to reduce interference from high-intensity applications)
  - Place applications that benefit from memory bandwidth closer to the controller

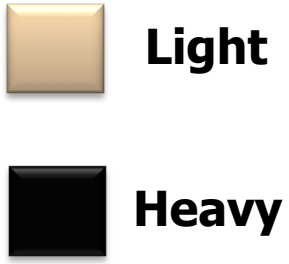
Multi-Core to Many-Core

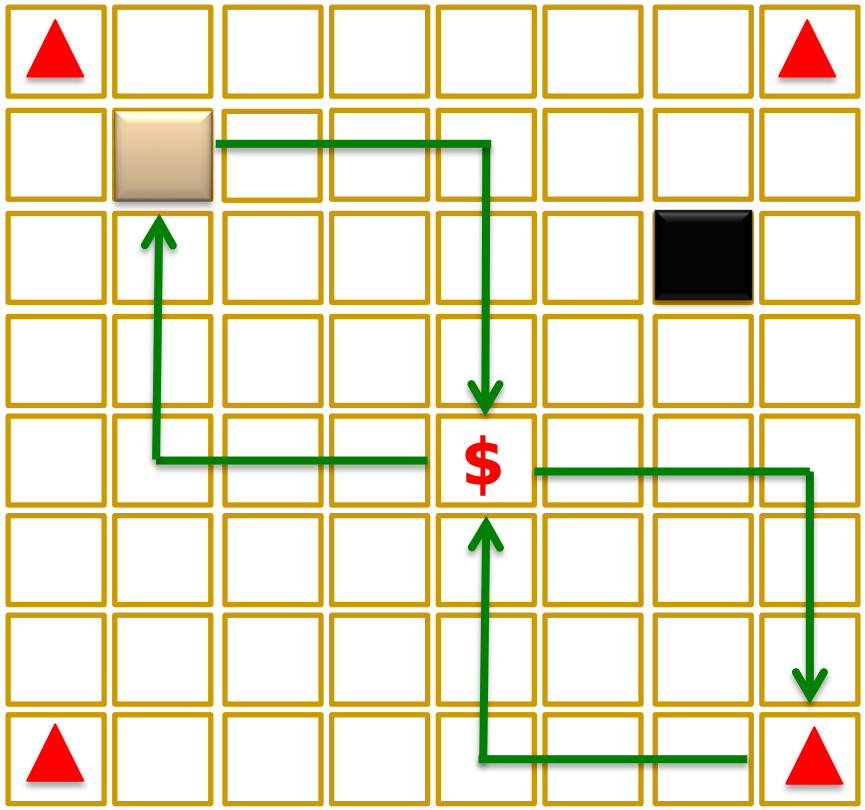


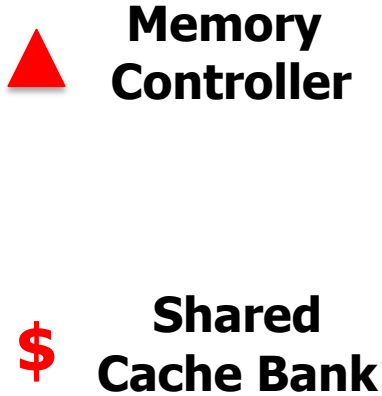
Multi-Core **Many-Core**

Many-Core On-Chip Communication

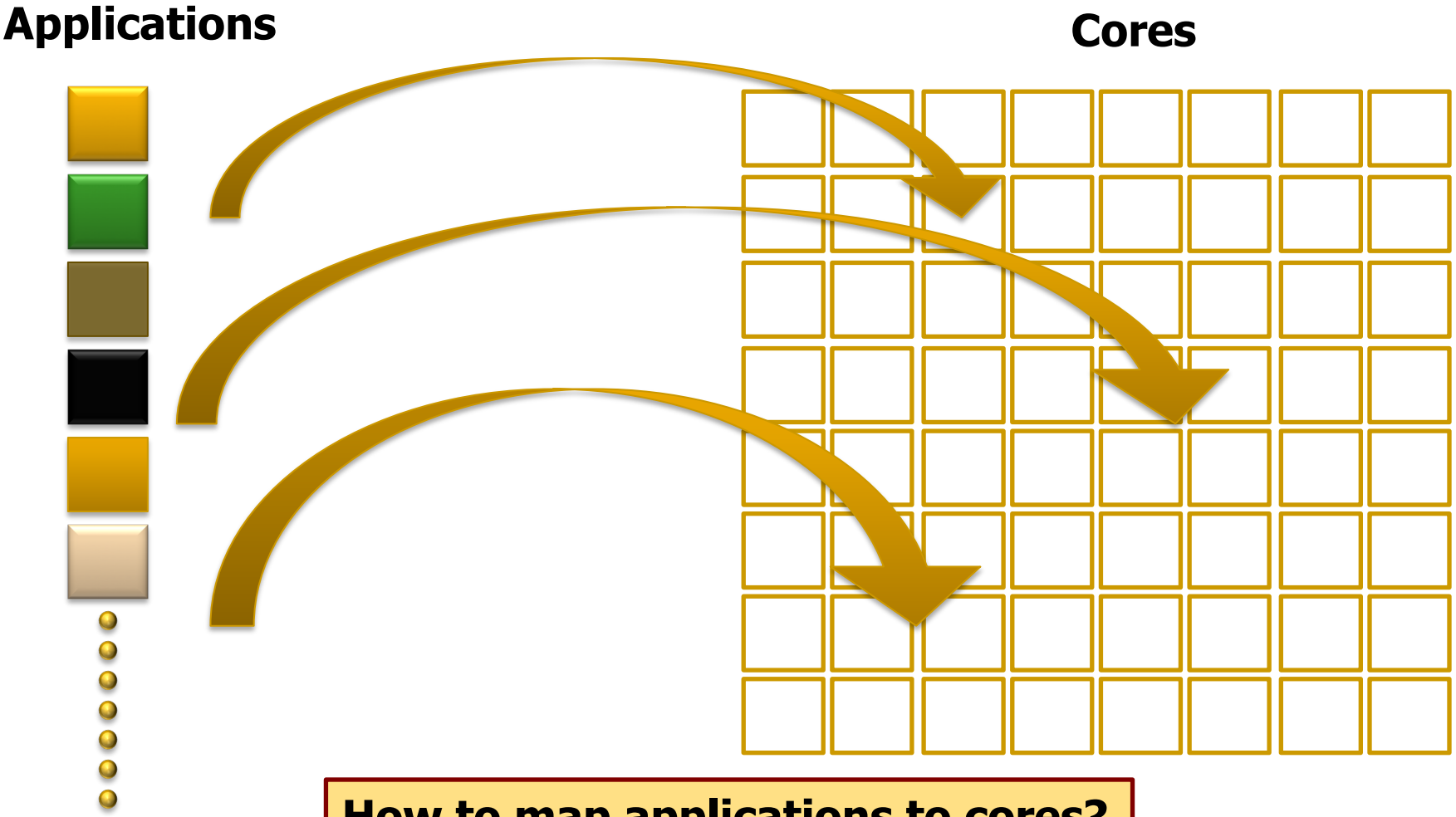
Applications





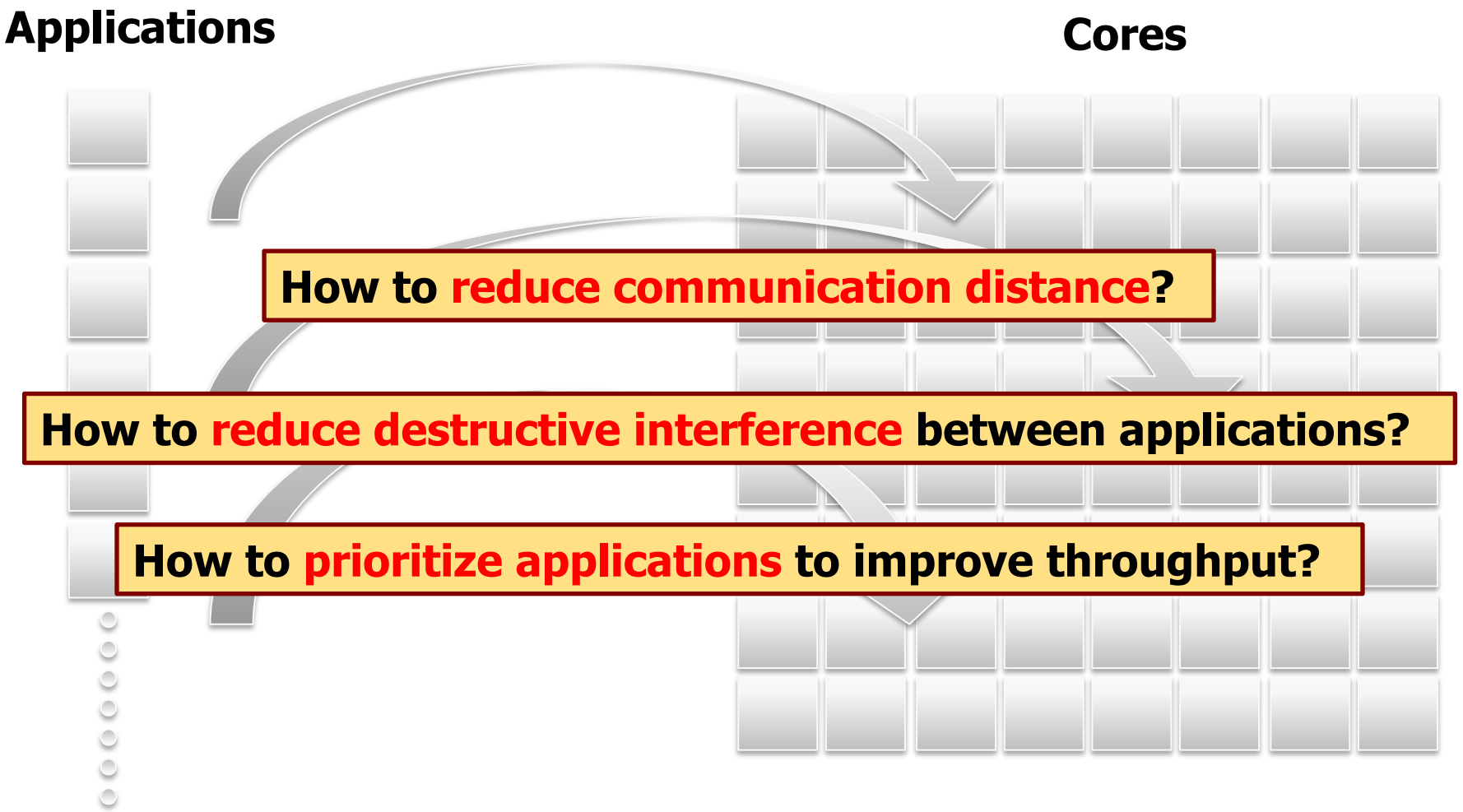


Problem: Spatial Task Scheduling

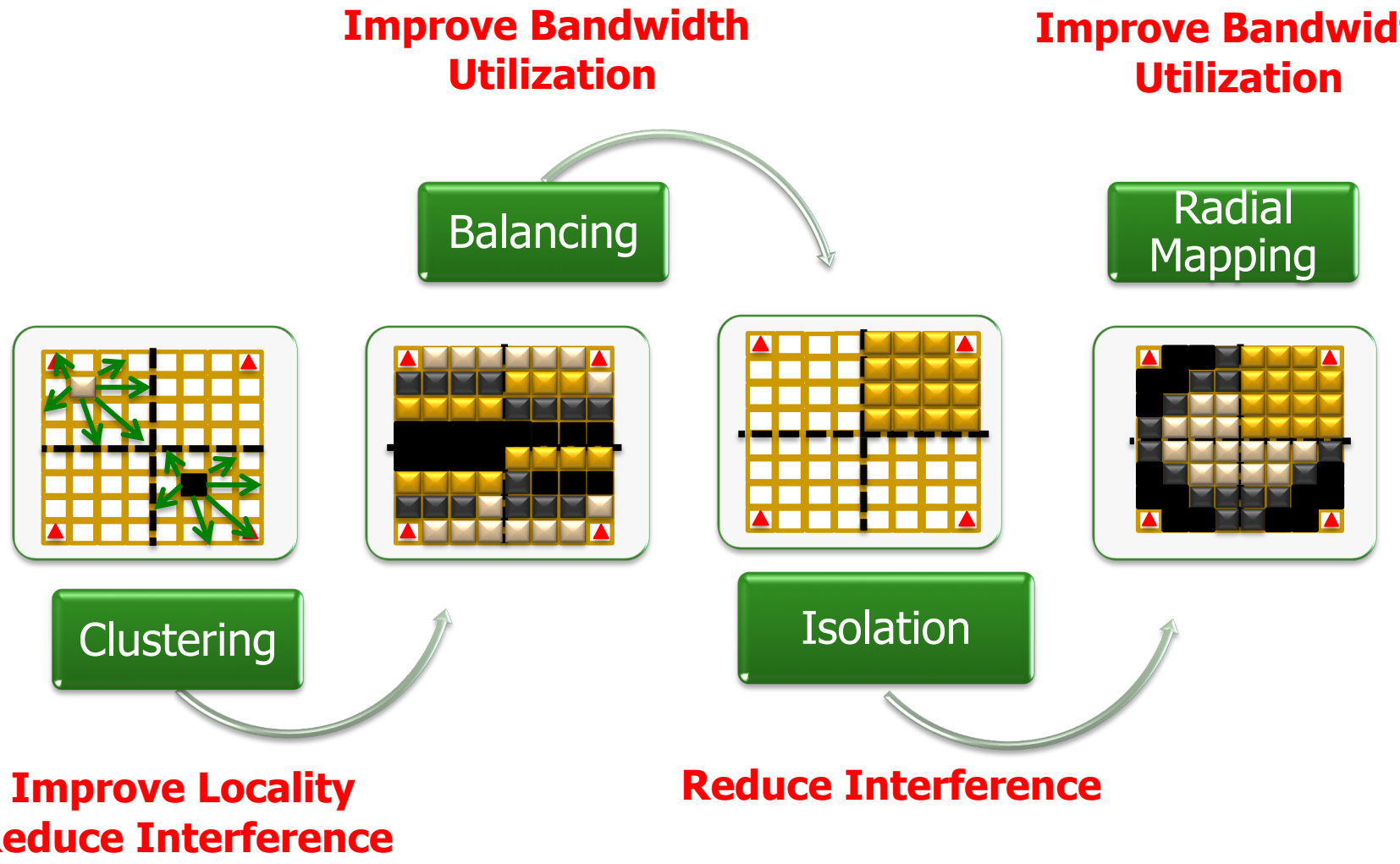


How to map applications to cores?

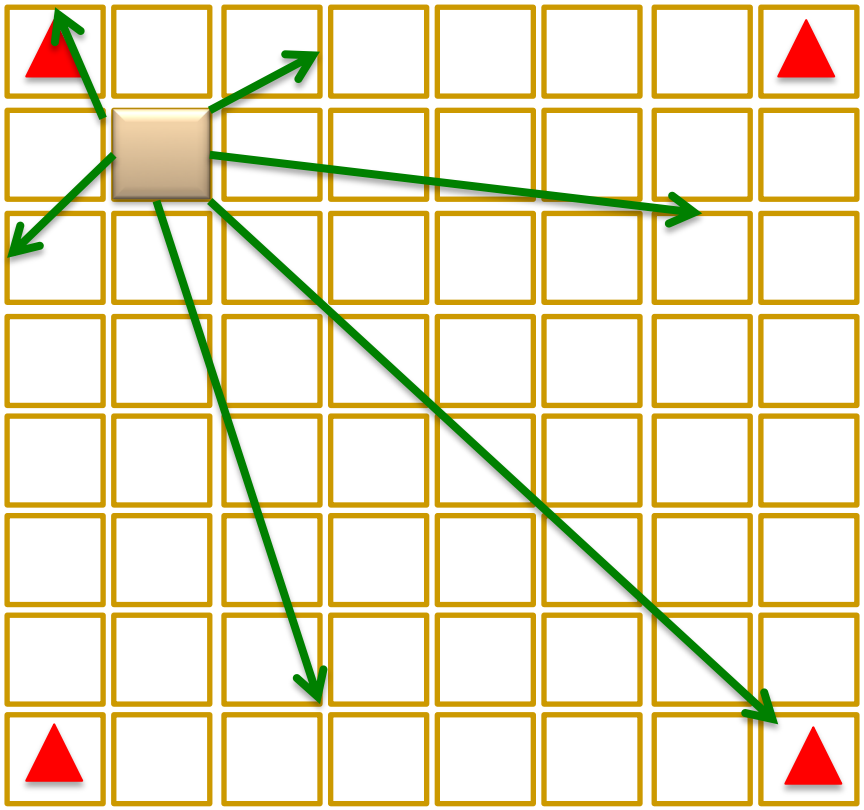
Challenges in Spatial Task Scheduling



Application-to-Core Mapping



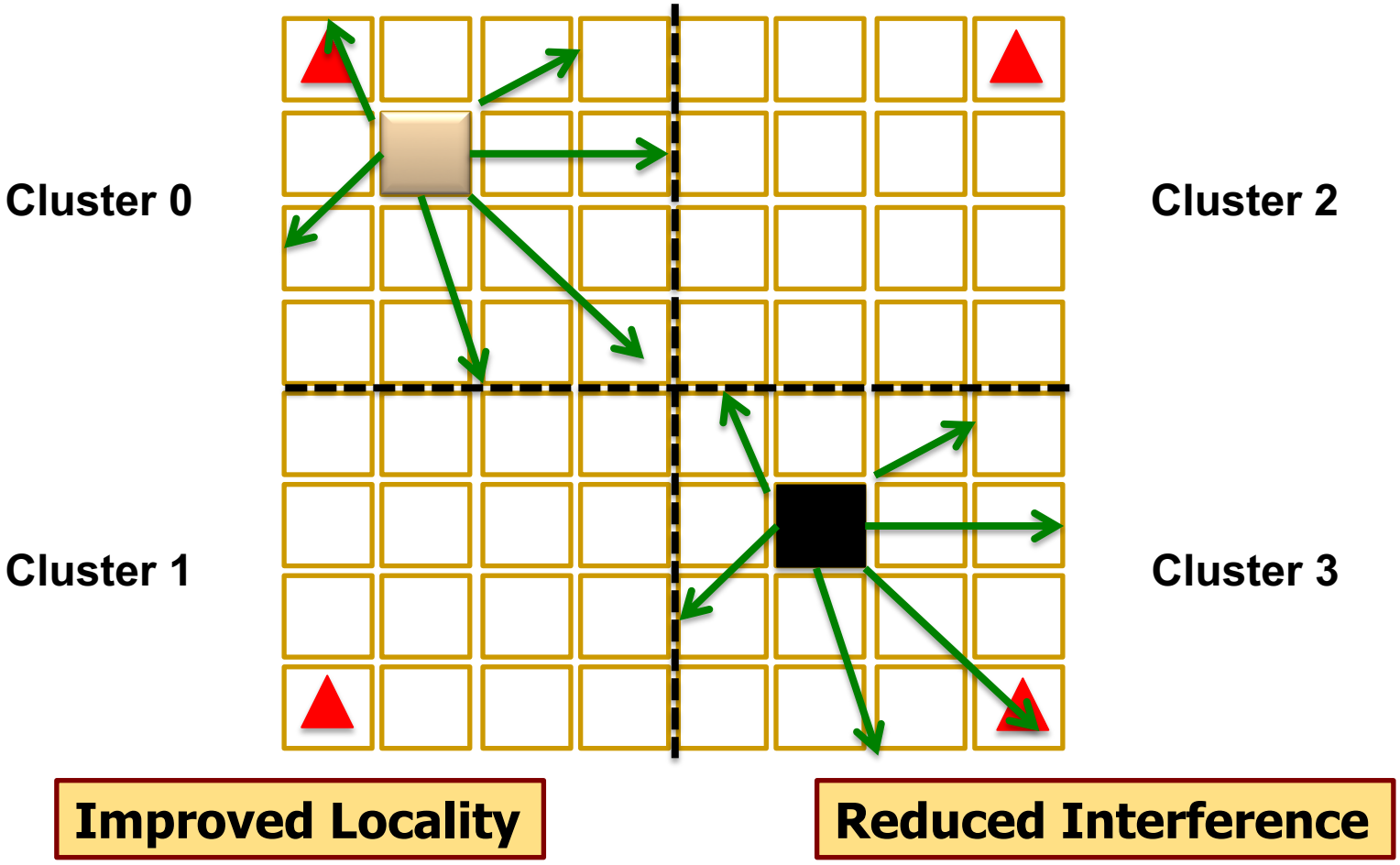
Step 1 — Clustering



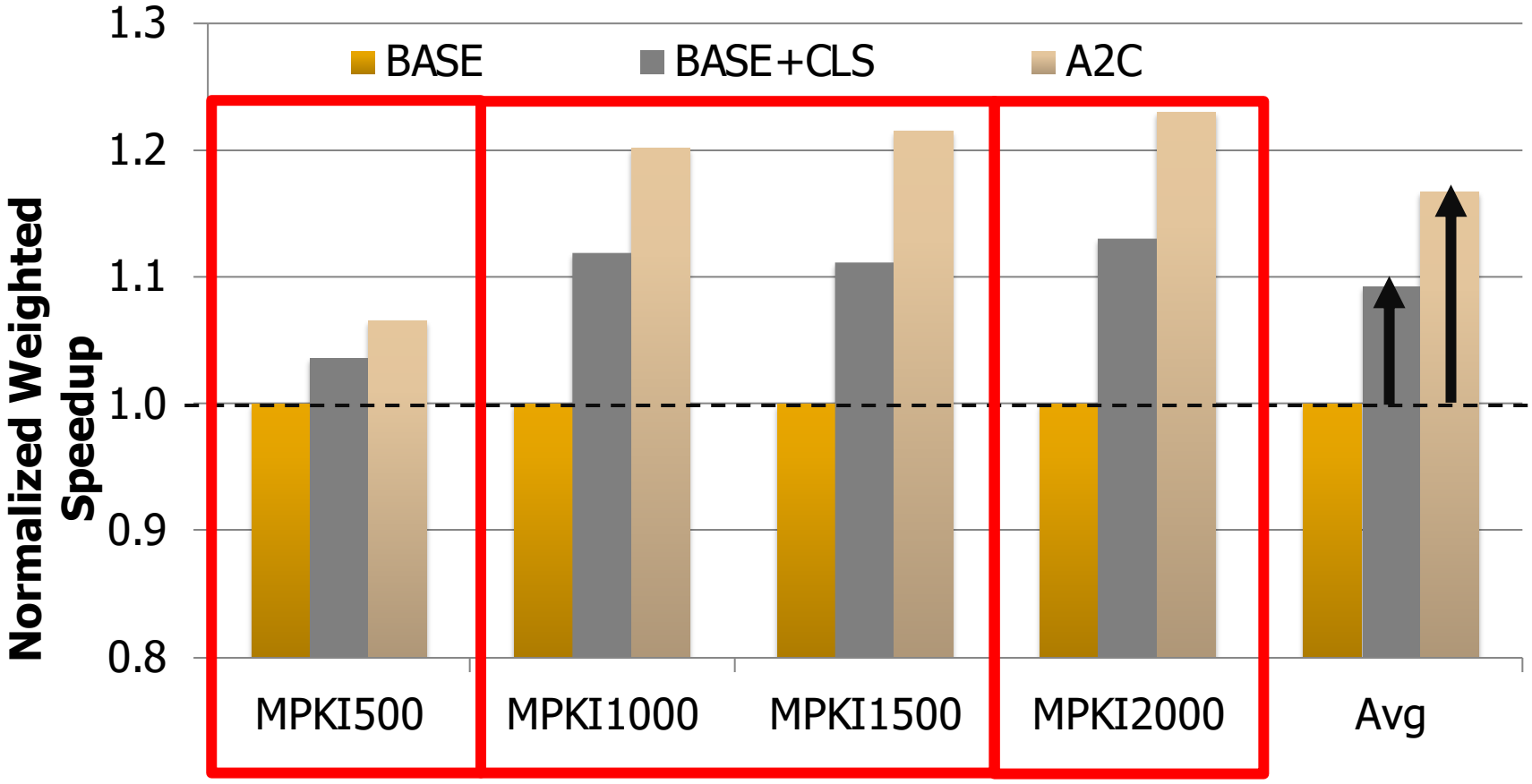
Memory Controller

Inefficient data mapping to memory and caches

Step 1 — Clustering

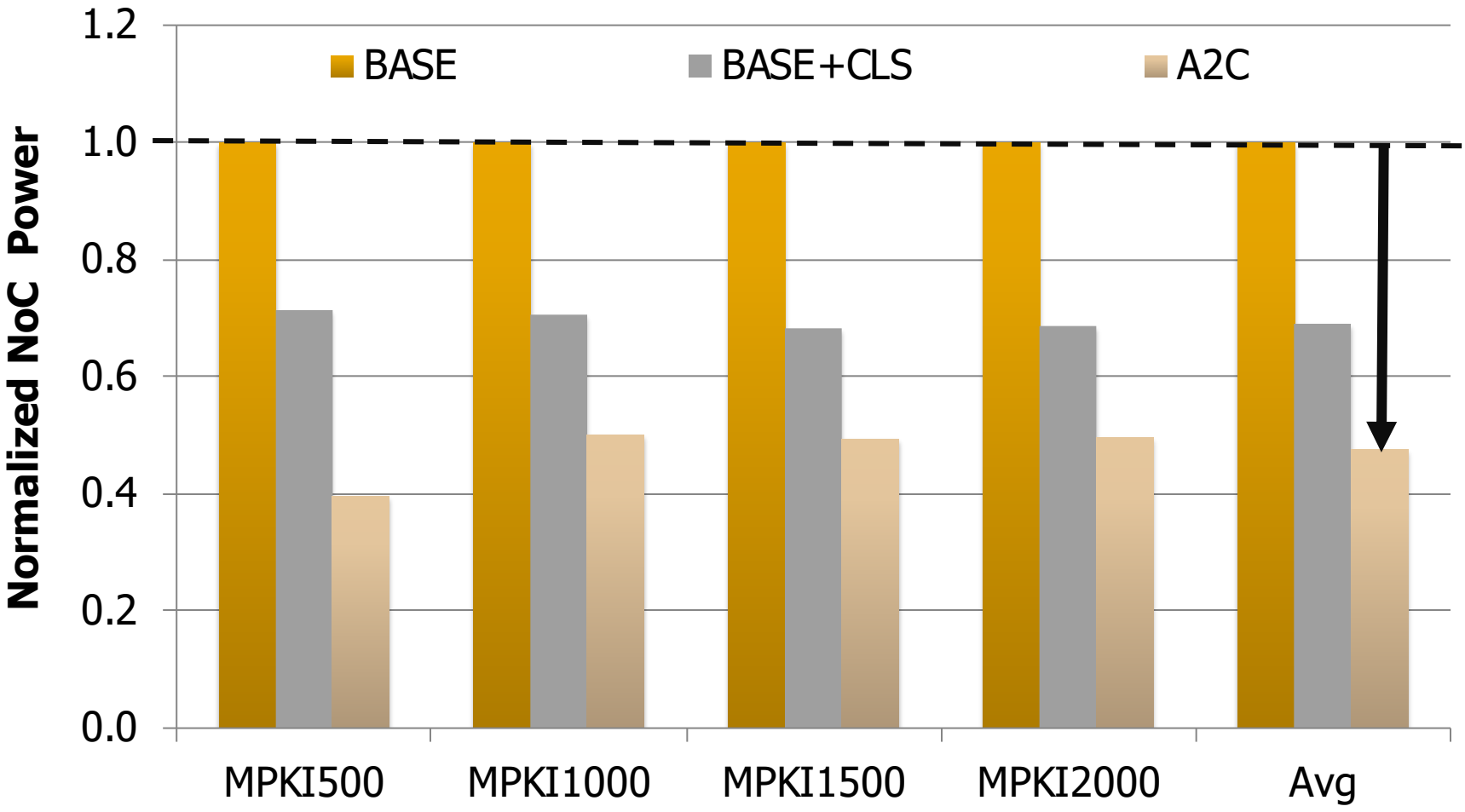


System Performance



System performance improves by 17%

Network Power



Average network power consumption reduces by 52%

More on App-to-Core Mapping

- Reetuparna Das, Rachata Ausavarungnirun, Onur Mutlu, Akhilesh Kumar, and Mani Azimi,
  "Application-to-Core Mapping Policies to Reduce Memory System Interference in Multi-Core Systems"
  *Proceedings of the 19th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Shenzhen, China, February 2013.
  Slides (pptx)

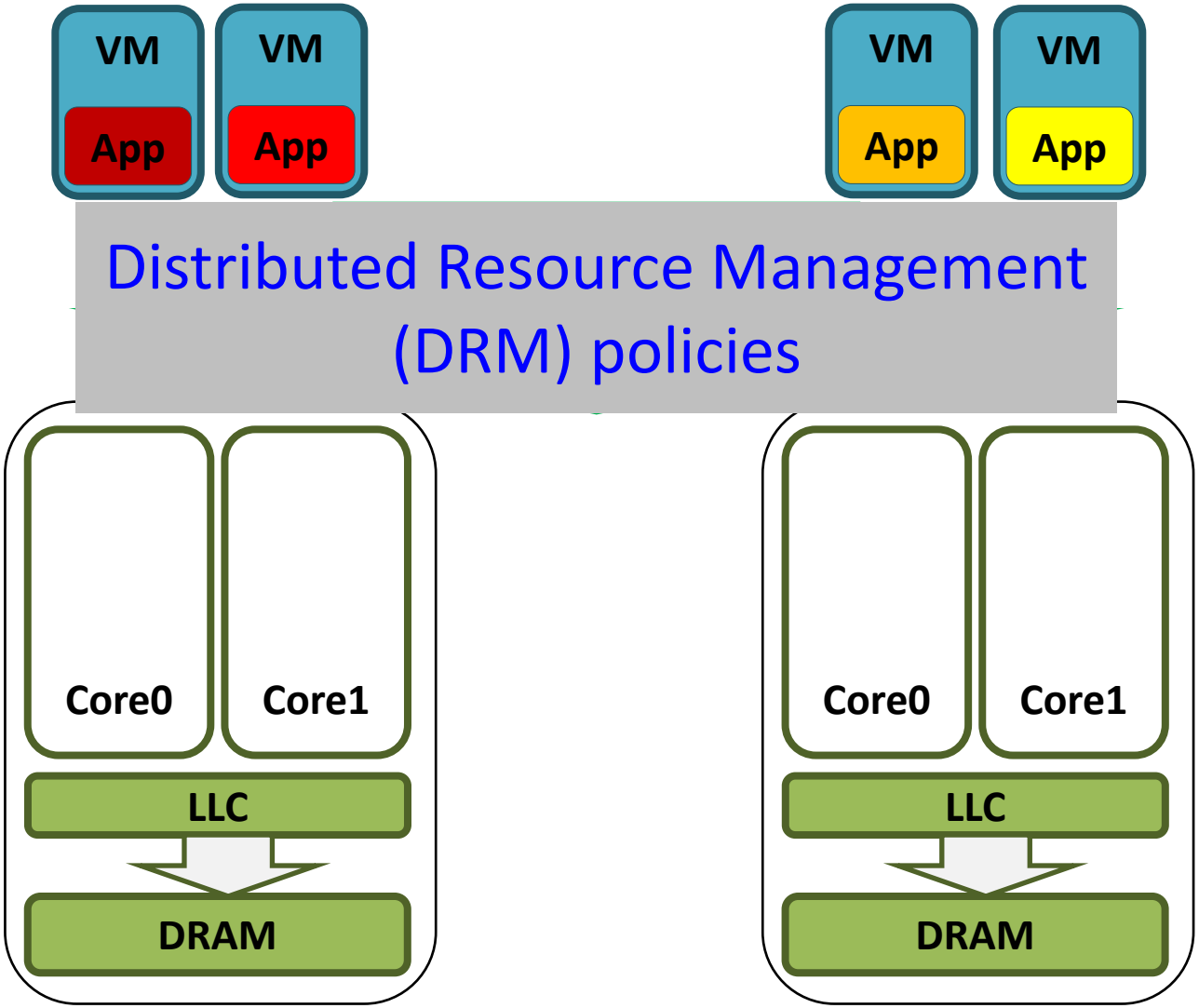
Application-to-Core Mapping Policies
to Reduce Memory System Interference in Multi-Core Systems

Reetuparna Das∗ Rachata Ausavarungnirun† Onur Mutlu† Akhilesh Kumar‡ Mani Azimi‡
University of Michigan∗ Carnegie Mellon University† Intel Labs‡

Interference-Aware Thread Scheduling

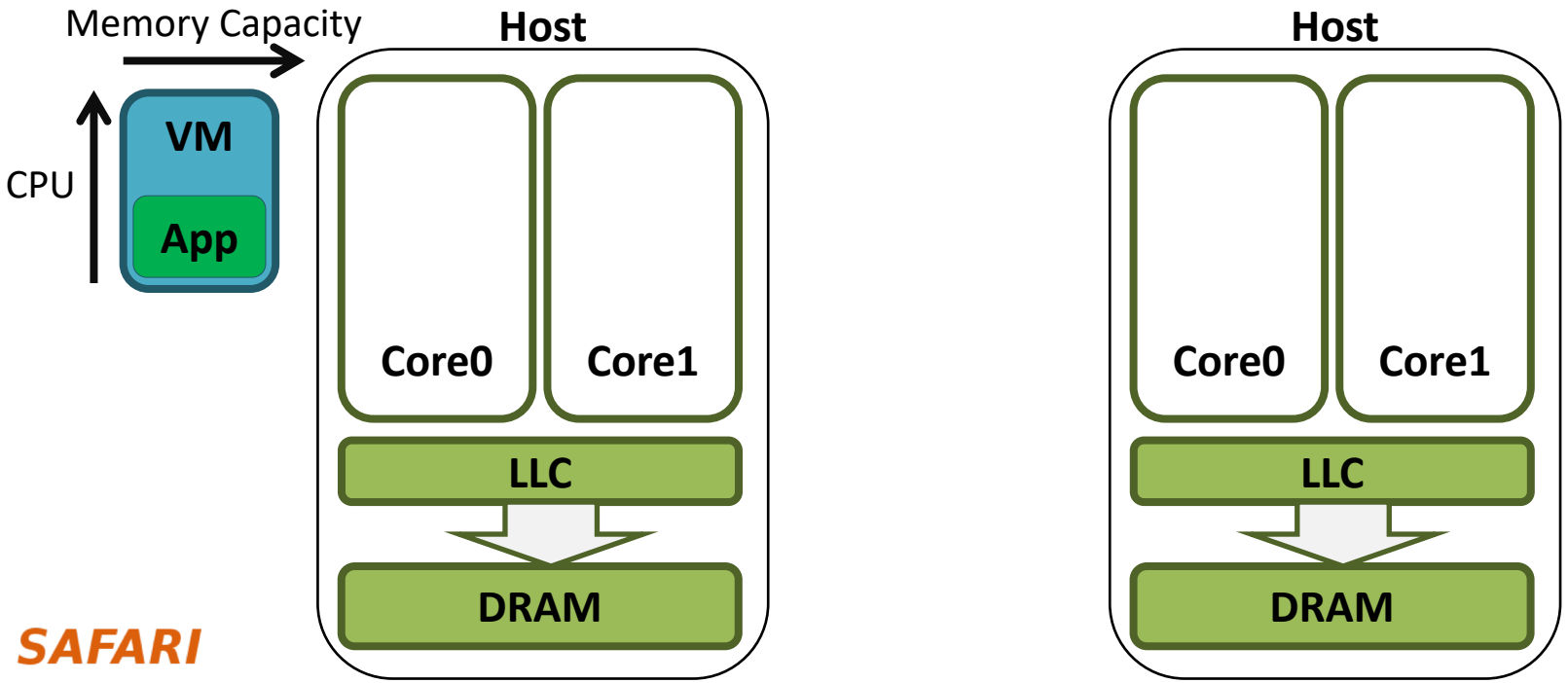
- An example from scheduling in compute clusters (data centers)
- Data centers can be running virtual machines

Virtualized Cluster



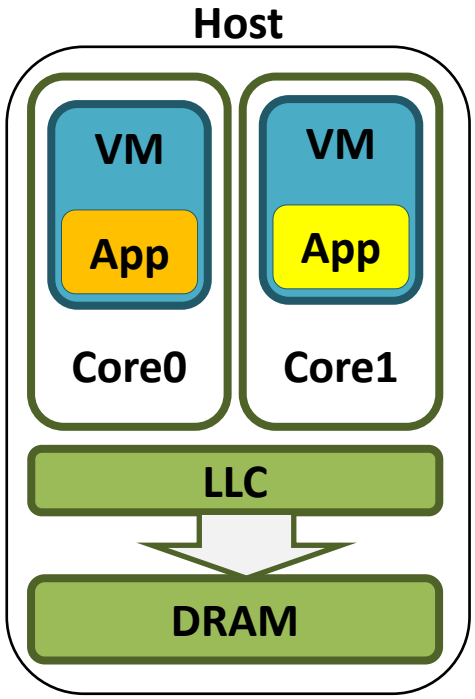
Conventional DRM Policies

Based on operating-system-level metrics
e.g., CPU utilization, memory capacity
demand



Microarchitecture-level Interference

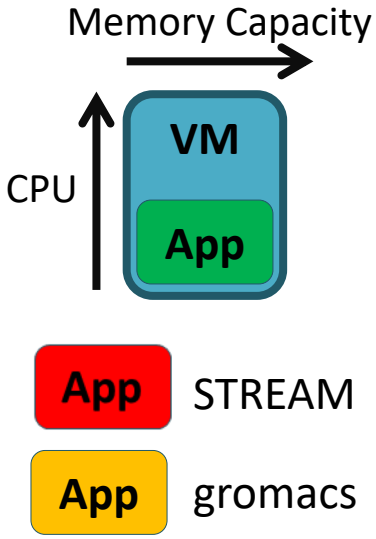
- VMs within a host compete for:
  - Shared cache capacity
  - Shared memory bandwidth

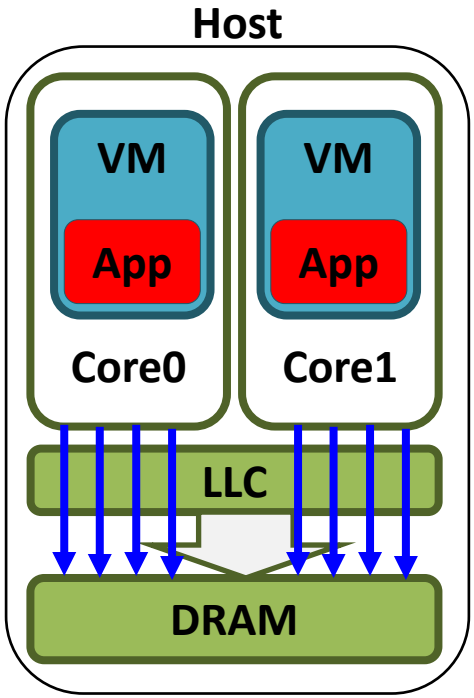


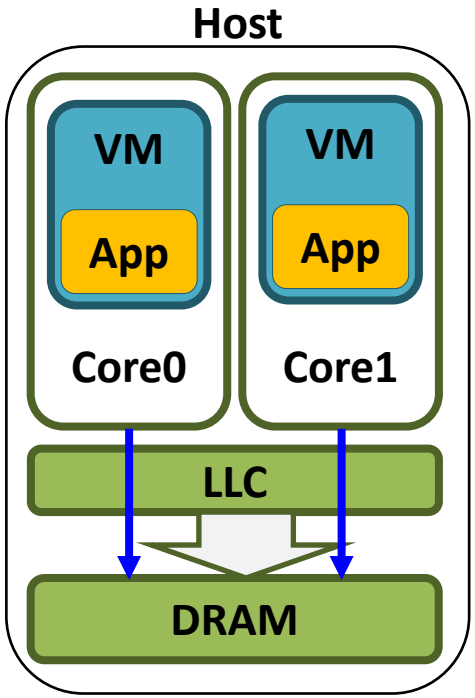
Can operating-system-level metrics capture the microarchitecture-level resource interference?

Microarchitecture Unawareness

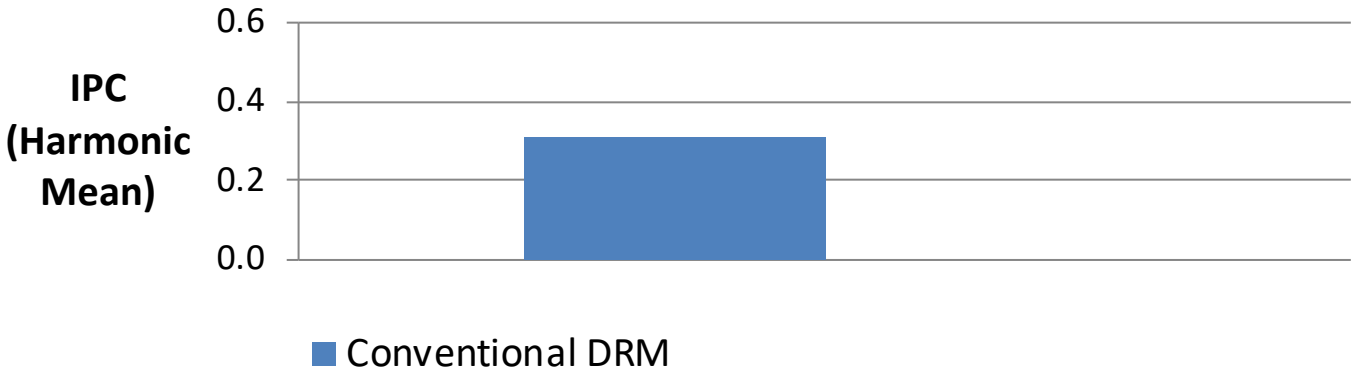
VM	Operating-system-level metrics		Microarchitecture-level metrics	
	CPU Utilization	Memory Capacity	LLC Hit Ratio	Memory Bandwidth
App	92%	369 MB	2%	2267 MB/s
App	93%	348 MB	98%	1 MB/s

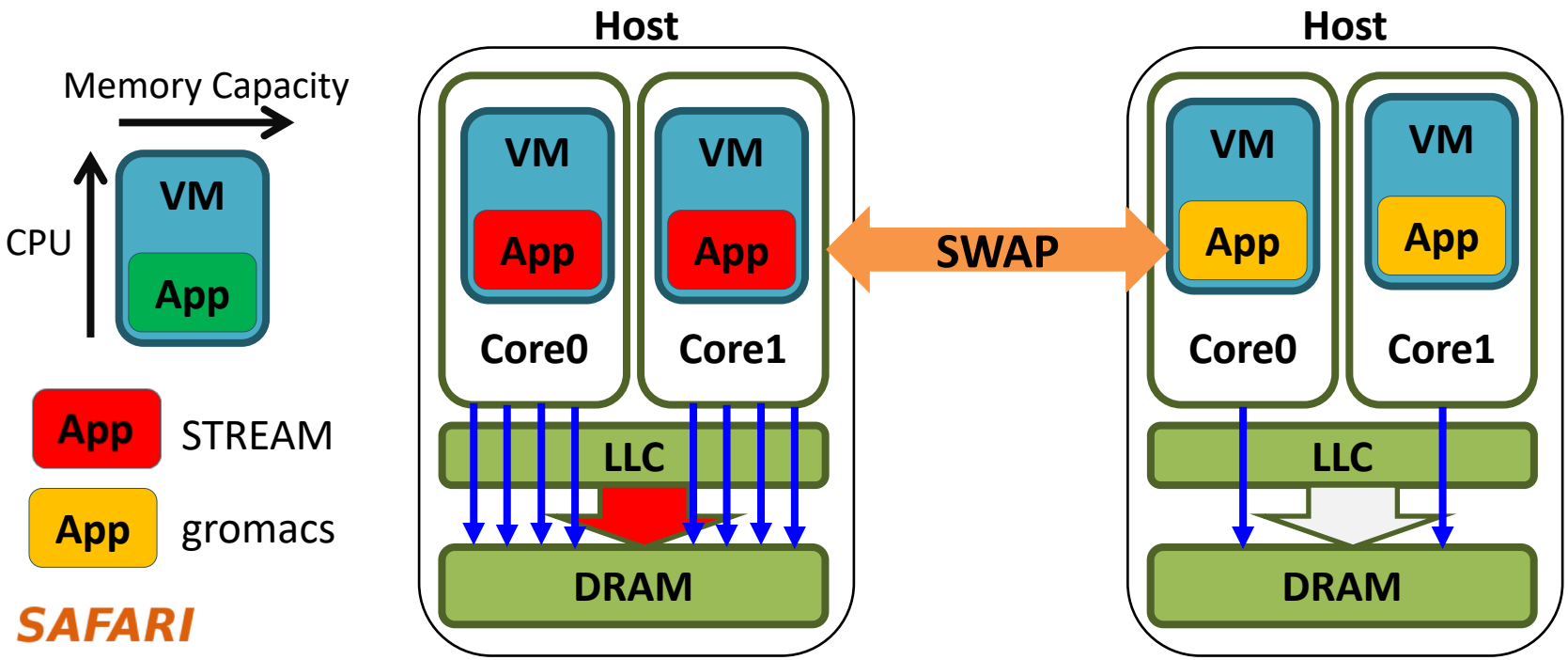




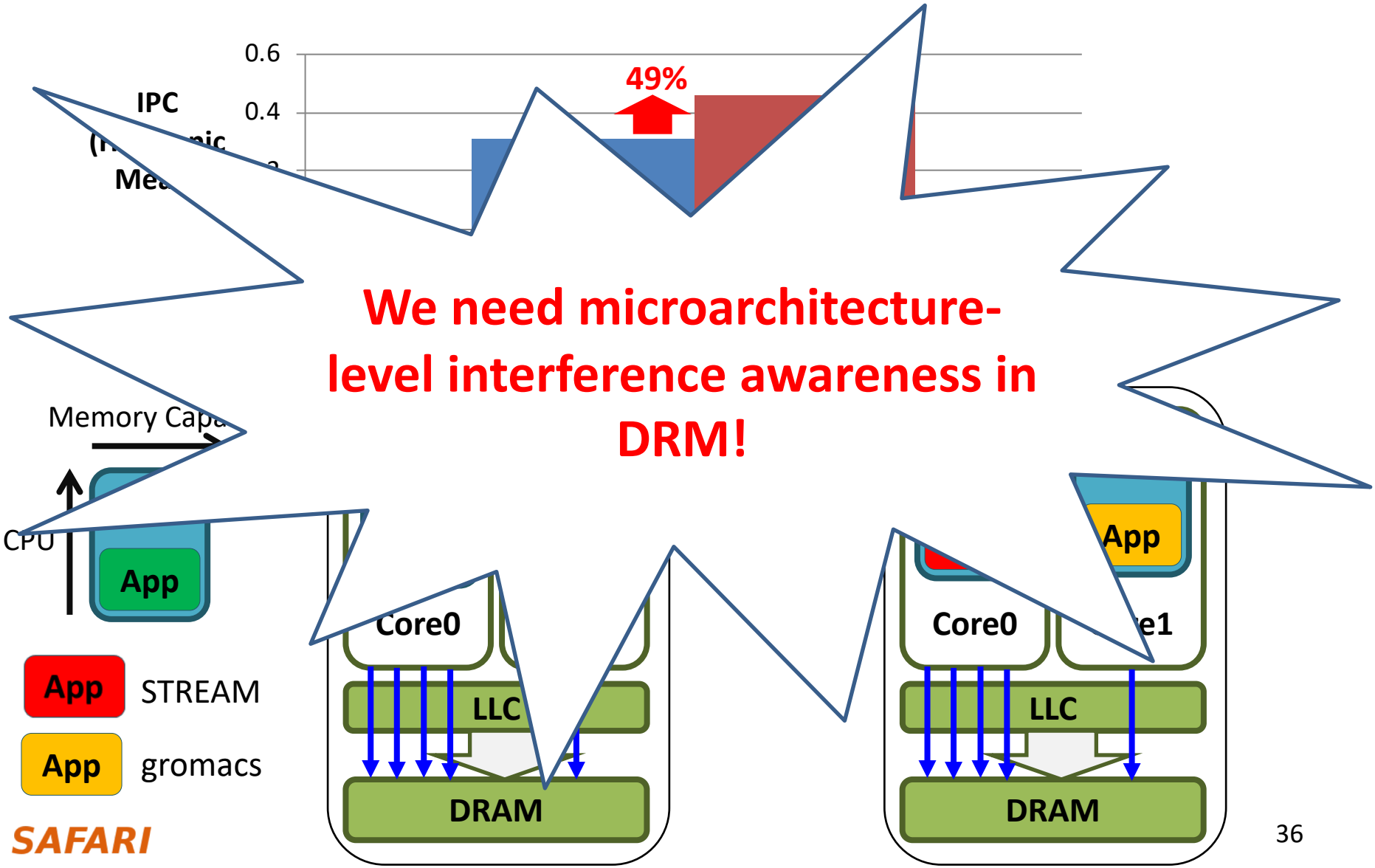


Impact on Performance





Impact on Performance

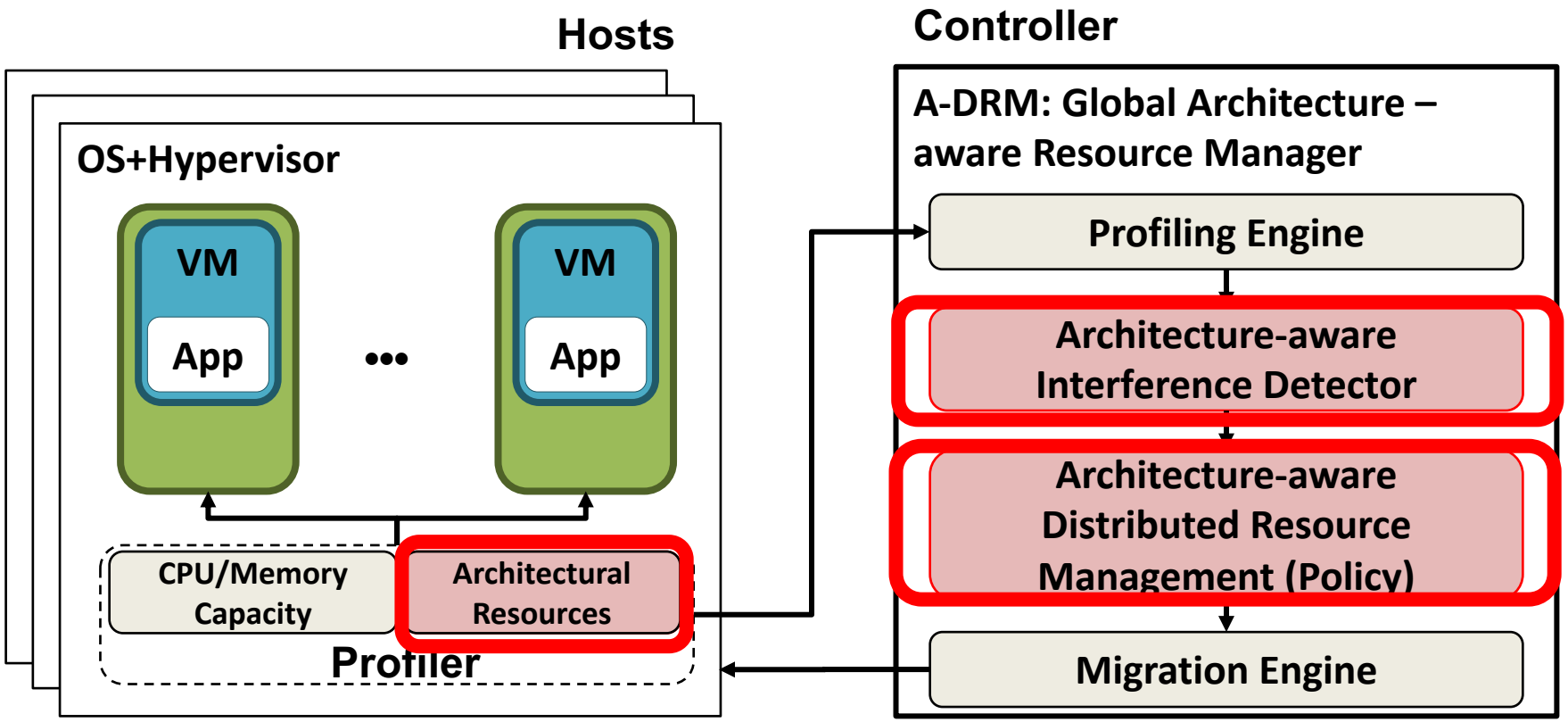


A-DRM: Architecture-aware DRM

- **Goal**: Take into account microarchitecture-level shared resource interference
  - Shared cache capacity
  - Shared memory bandwidth

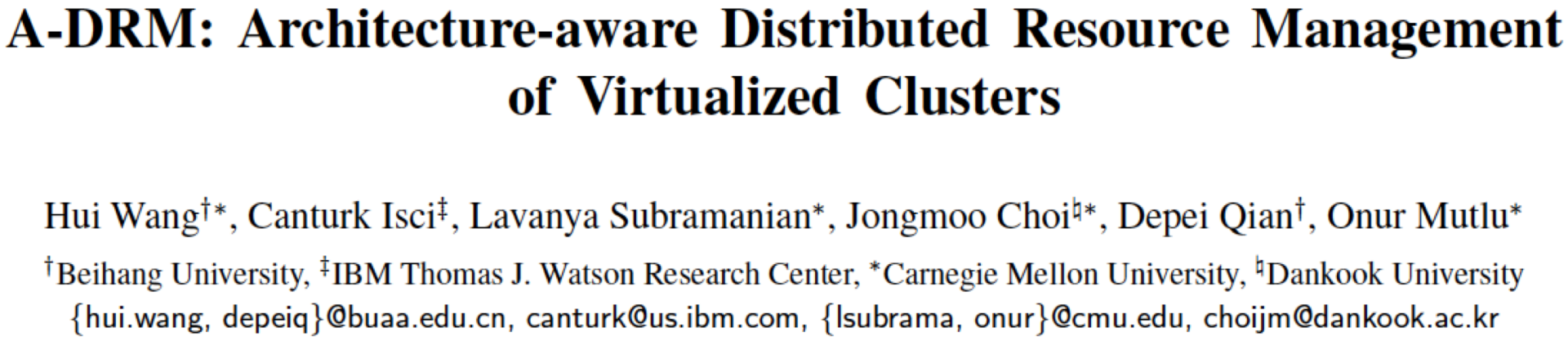
- **Key Idea**:
  - Monitor and detect microarchitecture-level shared resource interference
  - Balance microarchitecture-level resource usage across cluster to minimize memory interference while maximizing system performance

A-DRM: Architecture-aware DRM



More on Architecture-Aware DRM

- Hui Wang, Canturk Isci, Lavanya Subramanian, Jongmoo Choi, Depei Qian, and Onur Mutlu,
  "A-DRM: Architecture-aware Distributed Resource Management of Virtualized Clusters"
  *Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (**VEE**)*, Istanbul, Turkey, March 2015.
  [Slides (pptx) (pdf)]



A-DRM: Architecture-aware Distributed Resource Management of Virtualized Clusters

Hui Wang[†*], Canturk Isci[‡], Lavanya Subramanian[*], Jongmoo Choi[♮*], Depei Qian[†], Onur Mutlu[*]

[†]Beihang University, [‡]IBM Thomas J. Watson Research Center, [*]Carnegie Mellon University, [♮]Dankook University

{hui.wang, depeiq}@buaa.edu.cn, canturk@us.ibm.com, {lsubrama, onur}@cmu.edu, choijm@dankook.ac.kr

Interference-Aware Thread Scheduling

- Advantages

  + Can eliminate/minimize interference by scheduling "symbiotic applications" together (as opposed to just managing the interference)

  + Less intrusive to hardware (less need to modify the hardware resources)

- Disadvantages and Limitations

  -- High overhead to migrate threads and data between cores and machines

  -- Does not work (well) if all threads are similar and they interfere

Summary

Summary: Fundamental Interference Control Techniques

- Goal: to reduce/control interference

1. Prioritization or request scheduling
2. Data mapping to banks/channels/ranks
3. Core/source throttling
4. Application/thread scheduling

Best is to combine all. How would you do that?

Summary: Memory QoS Approaches and Techniques

- Approaches: Smart vs. dumb resources
  - Smart resources: QoS-aware memory scheduling
  - Dumb resources: Source throttling; channel partitioning
  - Both approaches are effective in reducing interference
  - No single best approach for all workloads
- Techniques: Request/thread scheduling, source throttling, memory partitioning
  - All approaches are effective in reducing interference
  - Can be applied at different levels: hardware vs. software
  - No single best technique for all workloads
- Combined approaches and techniques are the most powerful
  - Integrated Memory Channel Partitioning and Scheduling [MICRO'11]

Summary: Memory Interference and QoS

- QoS-unaware memory → uncontrollable and unpredictable system
- Providing QoS awareness improves performance, predictability, fairness, and utilization of the memory system
- Discussed many new techniques to:
  - Minimize memory interference
  - Provide predictable performance
- Many new research ideas needed for integrated techniques and closing the interaction with software

What Did We Not Cover?

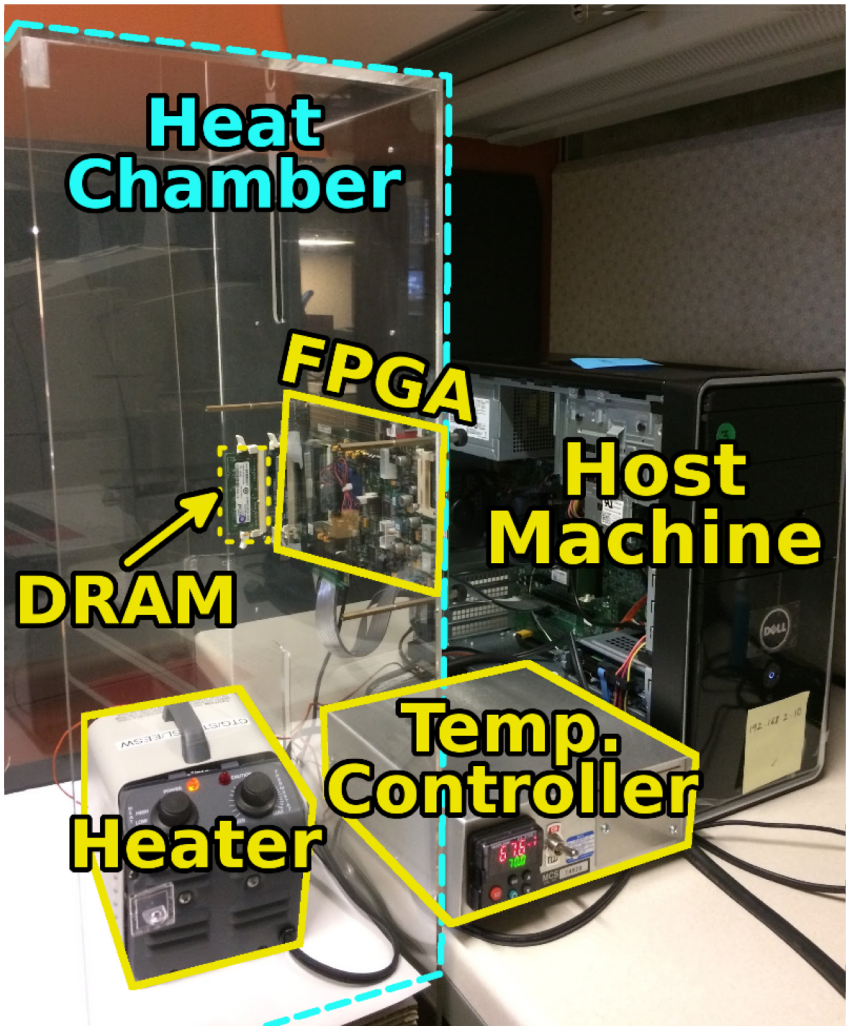
- Prefetch-aware shared resource management
- DRAM-controller co-design
- Cache interference management
- **Interconnect interference management**
- Write-read scheduling
- **DRAM designs to reduce interference**
- Interference issues in near-memory processing
- …

What the Future May Bring

- Simple yet powerful interference control and scheduling mechanisms
  - memory scheduling + interconnect scheduling
- Real implementations and investigations
  - SoftMC infrastructure, FPGA-based implementations
- Interference and QoS in the presence of even more heterogeneity
  - PIM, accelerators, …
- Automated techniques for resource management

SoftMC: Open Source DRAM Infrastructure

- Hasan Hassan et al., "**SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies**," HPCA 2017.



- **Flexible**
- **Easy to Use (C++ API)**
- **Open-source**

  github.com/CMU-SAFARI/SoftMC

SoftMC

- https://github.com/CMU-SAFARI/SoftMC

SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies

Hasan Hassan[1,2,3] Nandita Vijaykumar[3] Samira Khan[4,3] Saugata Ghose[3] Kevin Chang[3]
Gennady Pekhimenko[5,3] Donghyuk Lee[6,3] Oguz Ergin[2] Onur Mutlu[1,3]

[1]*ETH Zürich* [2]*TOBB University of Economics & Technology* [3]*Carnegie Mellon University*
[4]*University of Virginia* [5]*Microsoft Research* [6]*NVIDIA Research*

Some Other Ideas …

Decoupled DMA w/ Dual-Port DRAM

[PACT 2015]

Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM

Decoupled Direct Memory Access

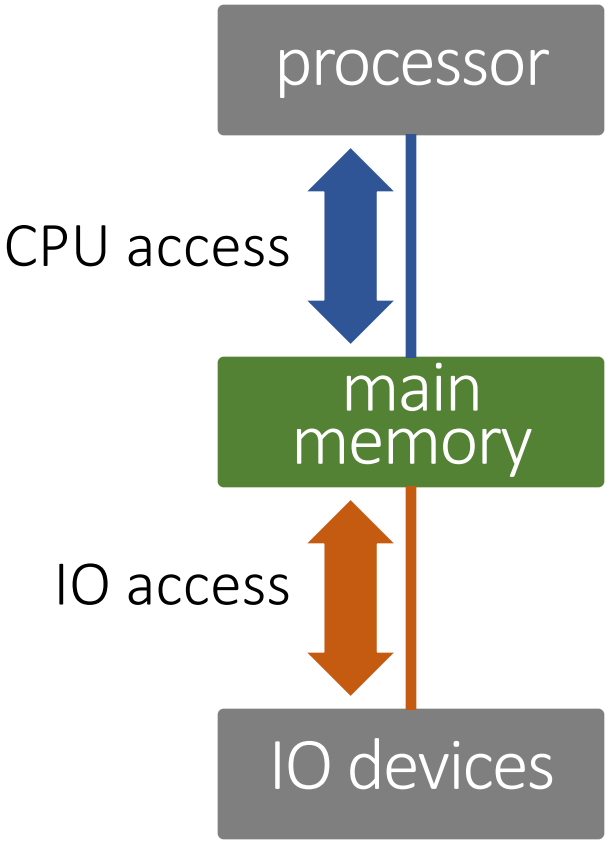
Donghyuk Lee

Lavanya Subramanian, Rachata Ausavarungnirun,
Jongmoo Choi, Onur Mutlu

SAFARI

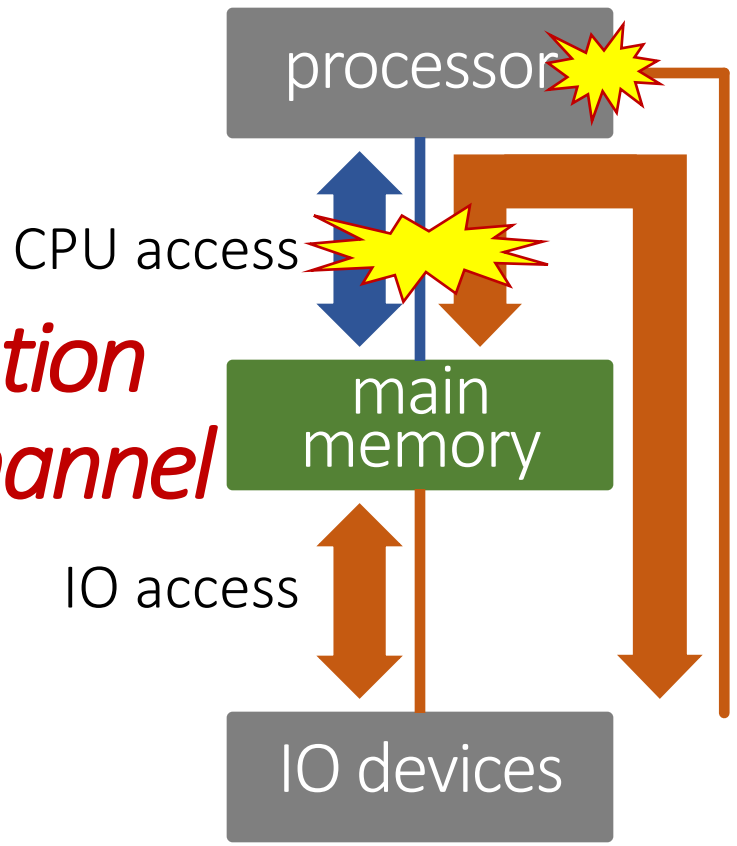
Carnegie Mellon

Logical System Organization



Main memory connects processor and IO devices as an *intermediate layer*

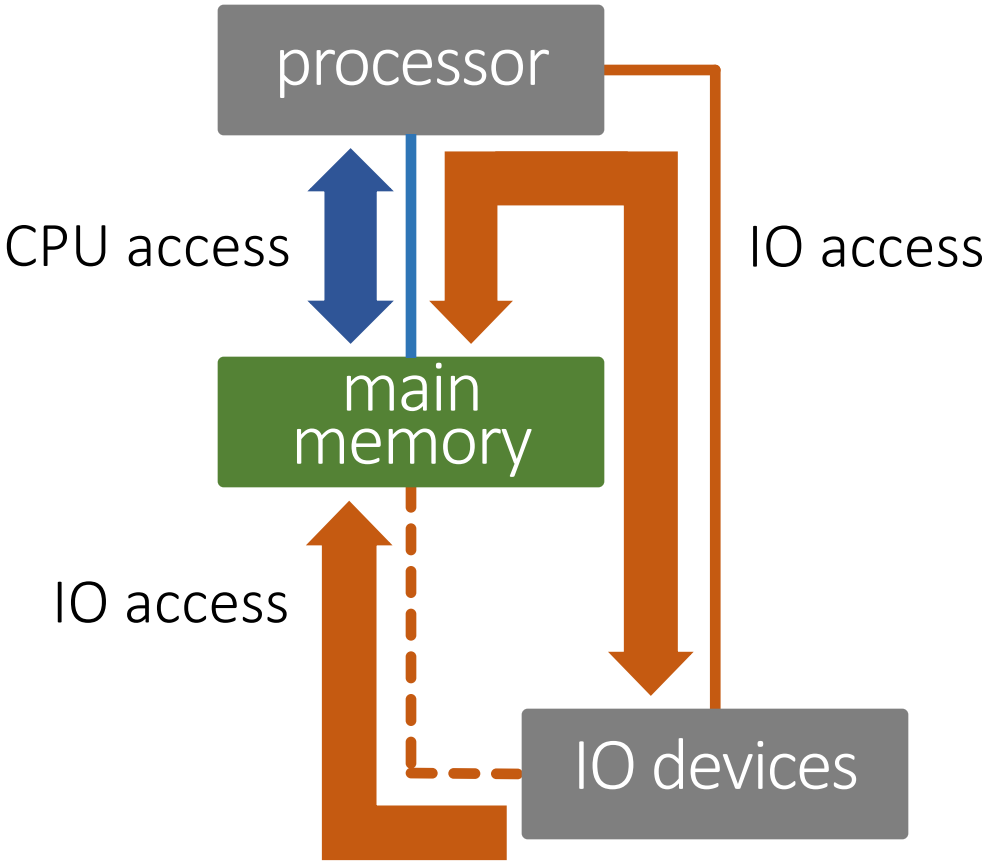
Physical System Implementation



High Pin Cost in Processor

High Contention in Memory Channel

Our Approach



Enabling IO channel,
decoupled & *isolated* from CPU channel

Executive Summary

- Problem
  - CPU and IO accesses contend for the shared memory channel
- Our Approach: *Decoupled Direct Memory Access (DDMA)*
  - Design new DRAM architecture with two independent data ports
    → *Dual-Data-Port DRAM*
  - Connect one port to CPU and the other port to IO devices
    → *Decouple CPU and IO accesses*
- Application
  - Communication between compute units (e.g., CPU – GPU)
  - In-memory communication (e.g., bulk in-memory copy/init.)
  - Memory-storage communication (e.g., page fault, IO prefetch)
- Result
  - Significant *performance improvement* (20% in 2 ch. & 2 rank system)
  - *CPU pin count reduction* (4.5%)

Outline

1. Problem
2. Our Approach
3. Dual-Data-Port DRAM
4. Applications for DDMA
5. Evaluation

Computer Architecture

Lecture 18a: Memory Interference and Quality of Service III

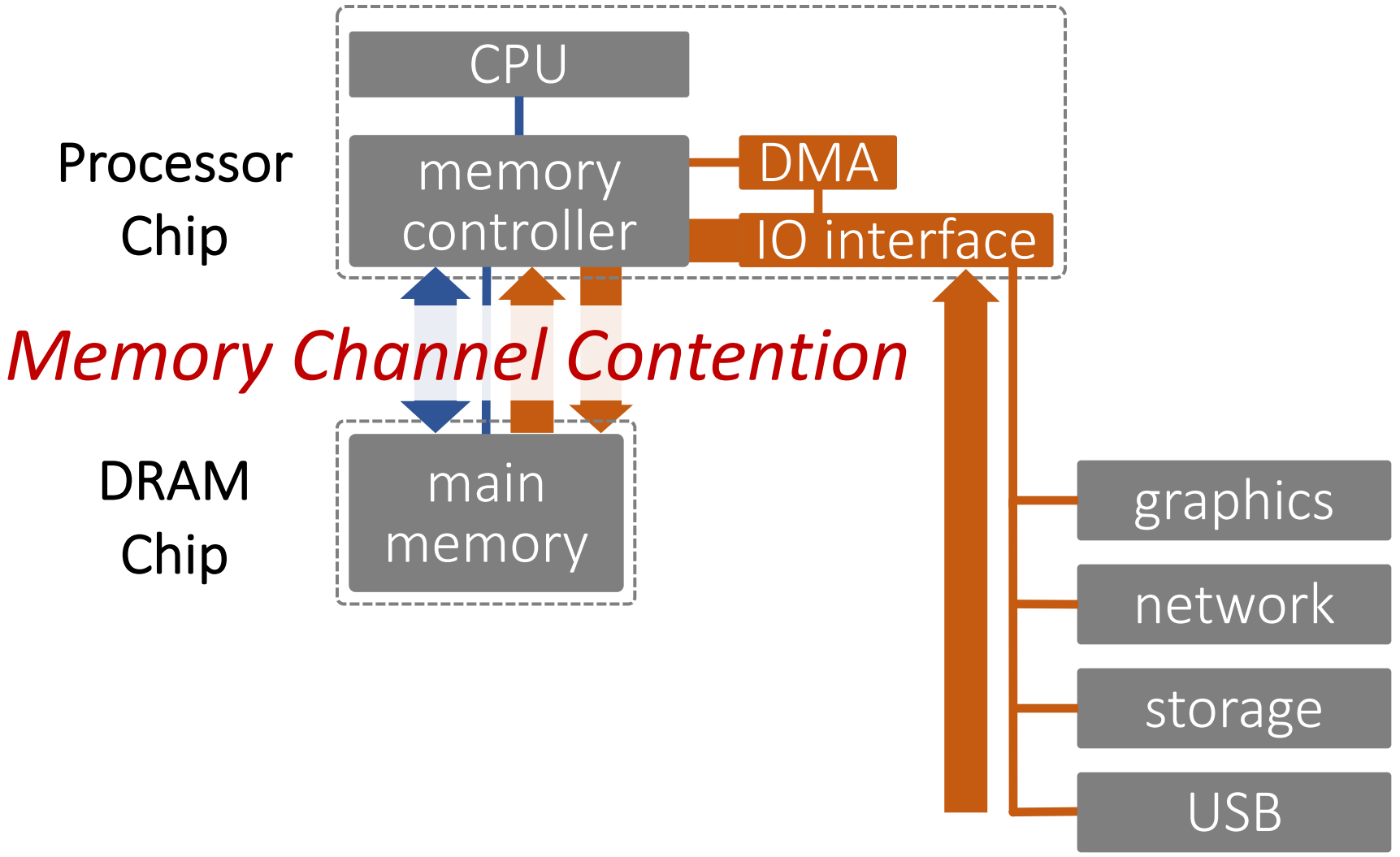
Prof. Onur Mutlu
ETH Zürich

Fall 2018

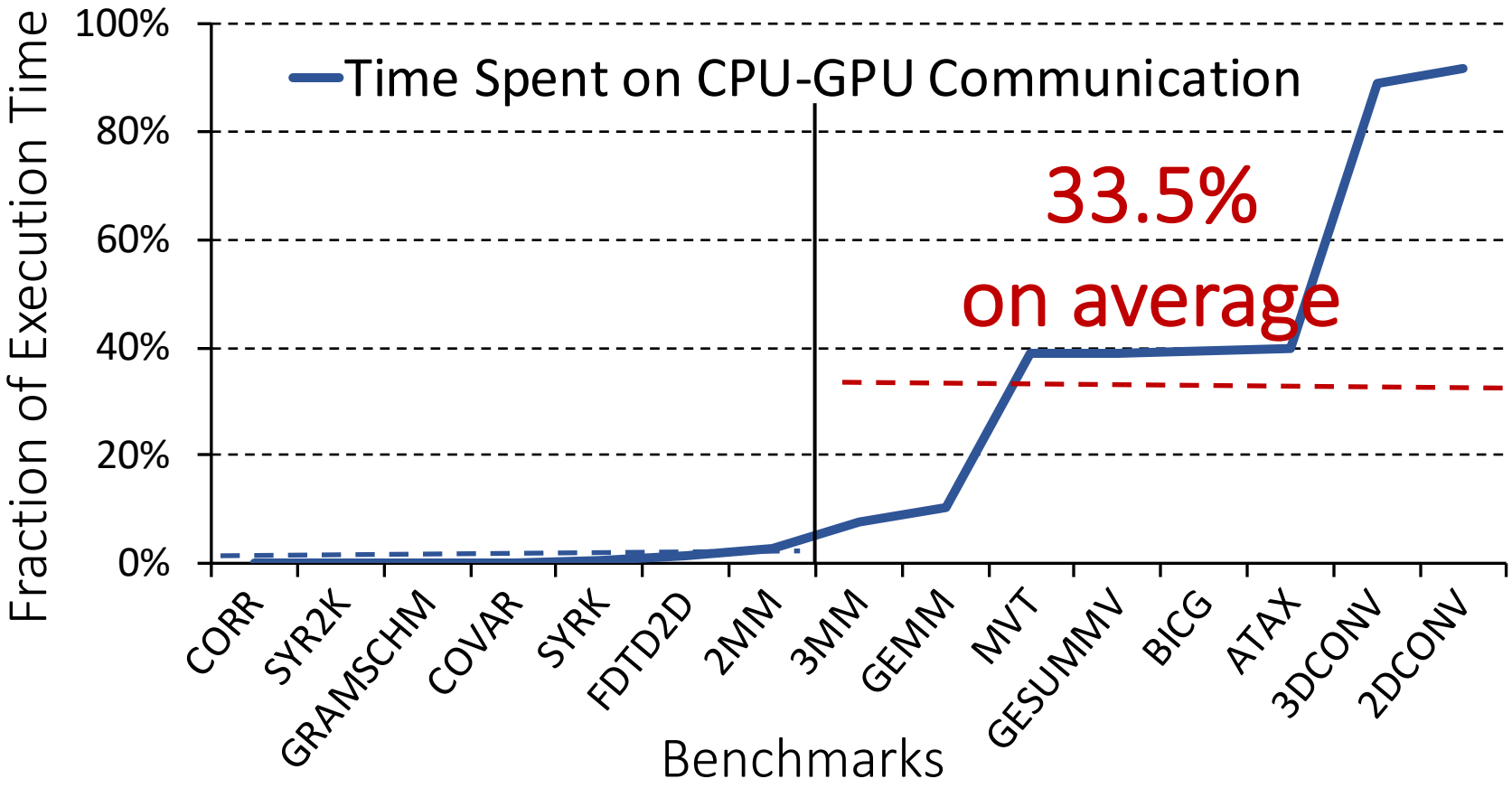
22 November 2018

We did not cover the following slides in lecture.
These are for your benefit.

Problem 1: Memory Channel Contention

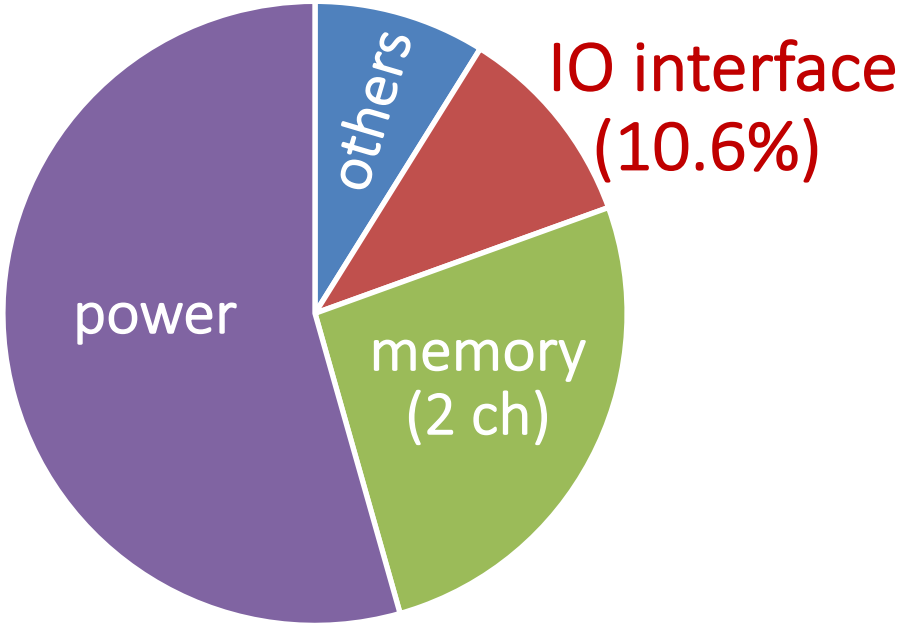


Problem 1: Memory Channel Contention



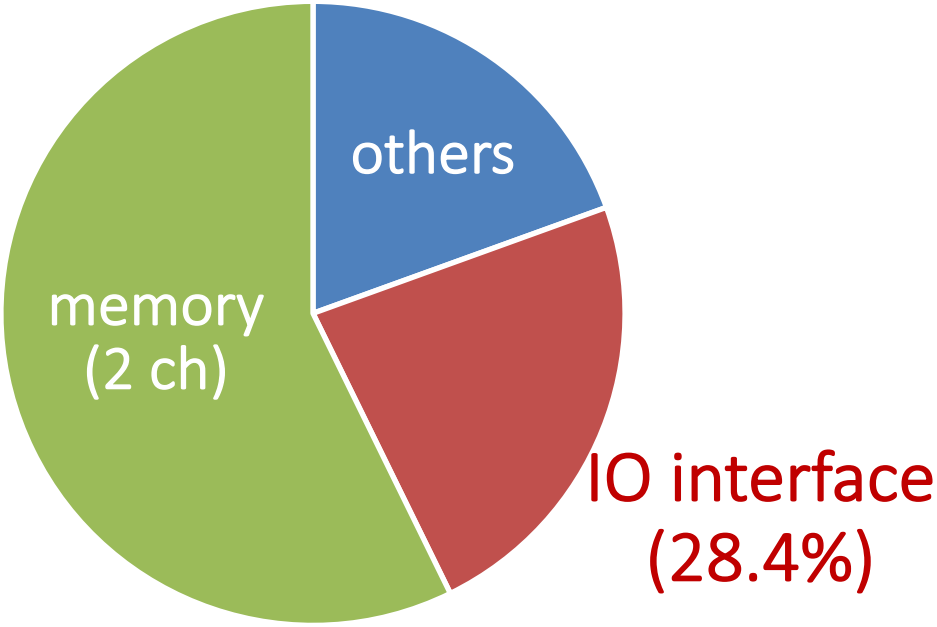
A large fraction of the execution time is spent on IO accesses

Problem 2: High Cost for IO Interfaces



959 pins in total

Processor Pin Count
(w/ power pins)



359 pins in total

Processor Pin Count
(w/o power pins)

Integrating IO interface on the processor chip leads to *high area cost*

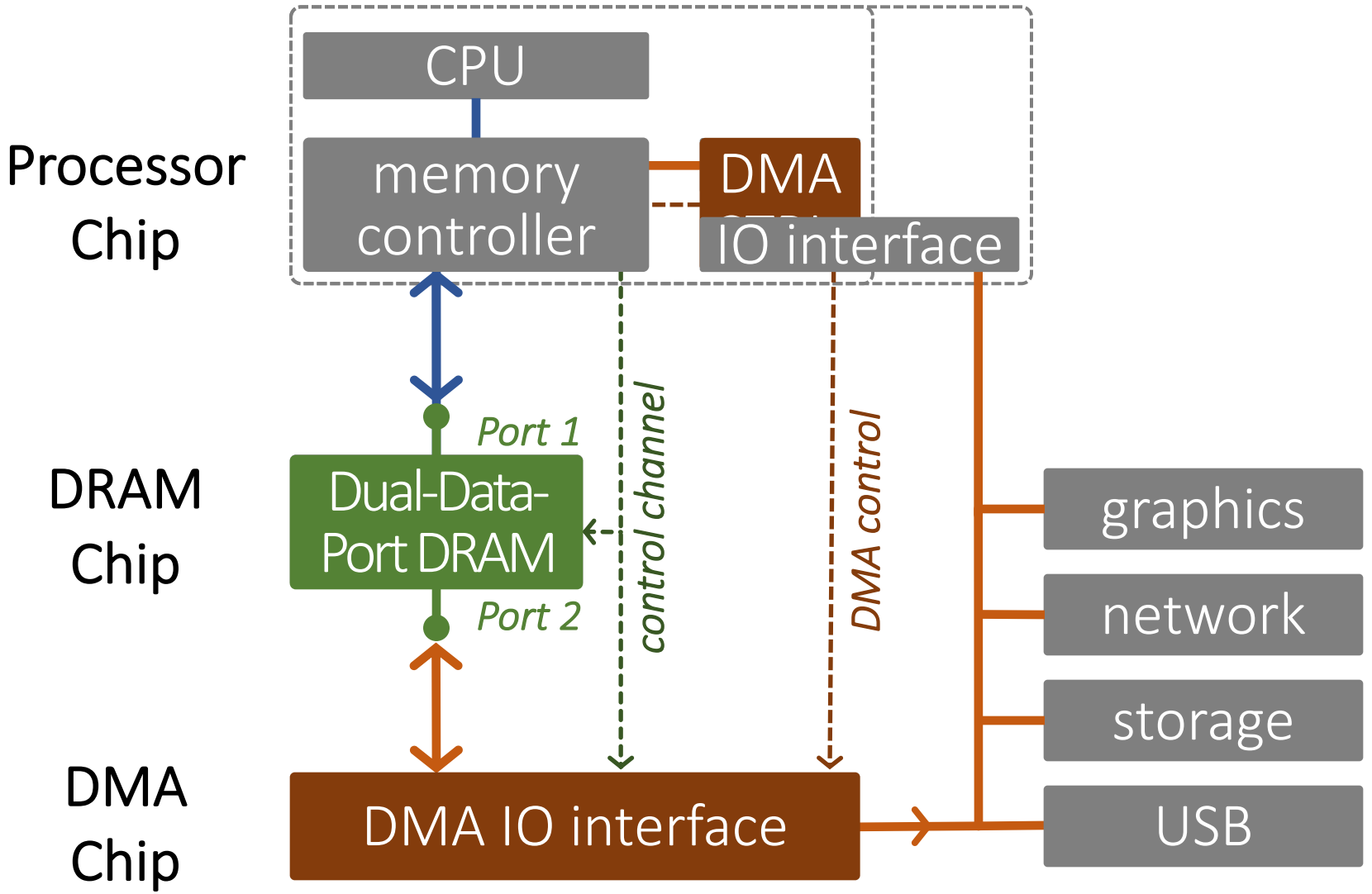
Shared Memory Channel

- Memory channel contention for IO access and CPU access
- High area cost for integrating IO interfaces on processor chip

Outline

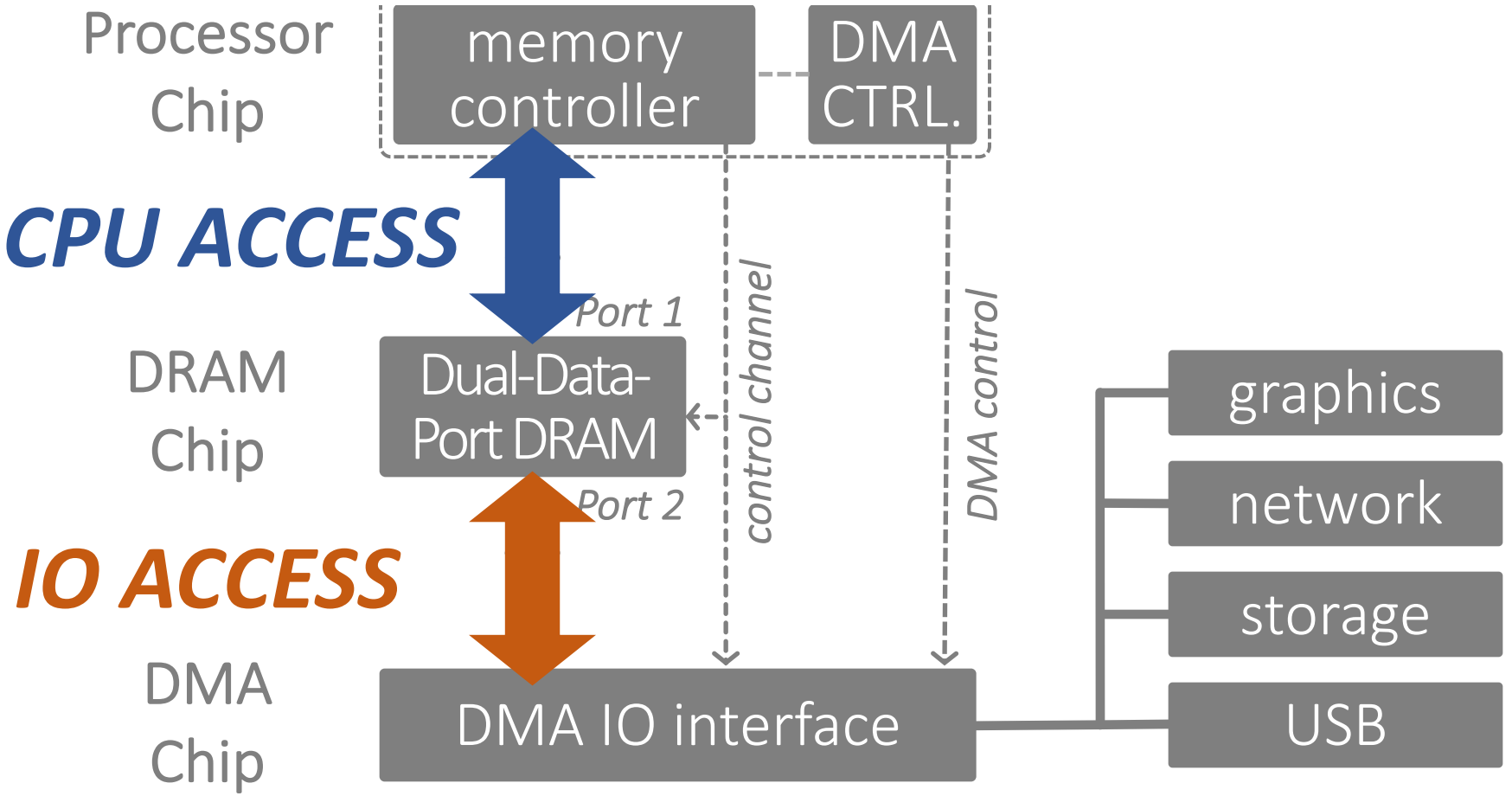
1. Problem
2. Our Approach
3. Dual-Data-Port DRAM
4. Applications for DDMA
5. Evaluation

Our Approach



Our Approach

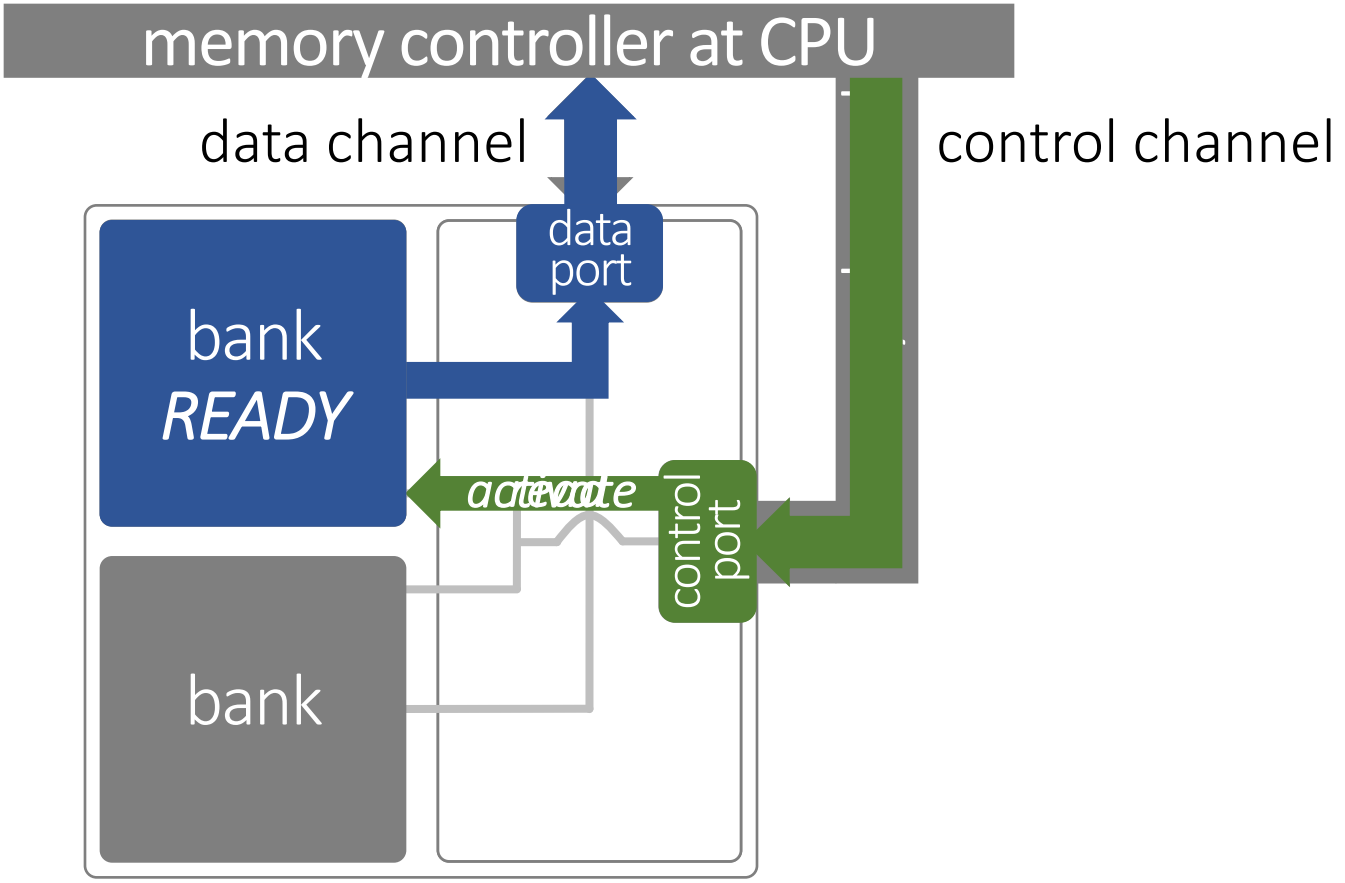
Decoupled Direct Memory Access



Outline

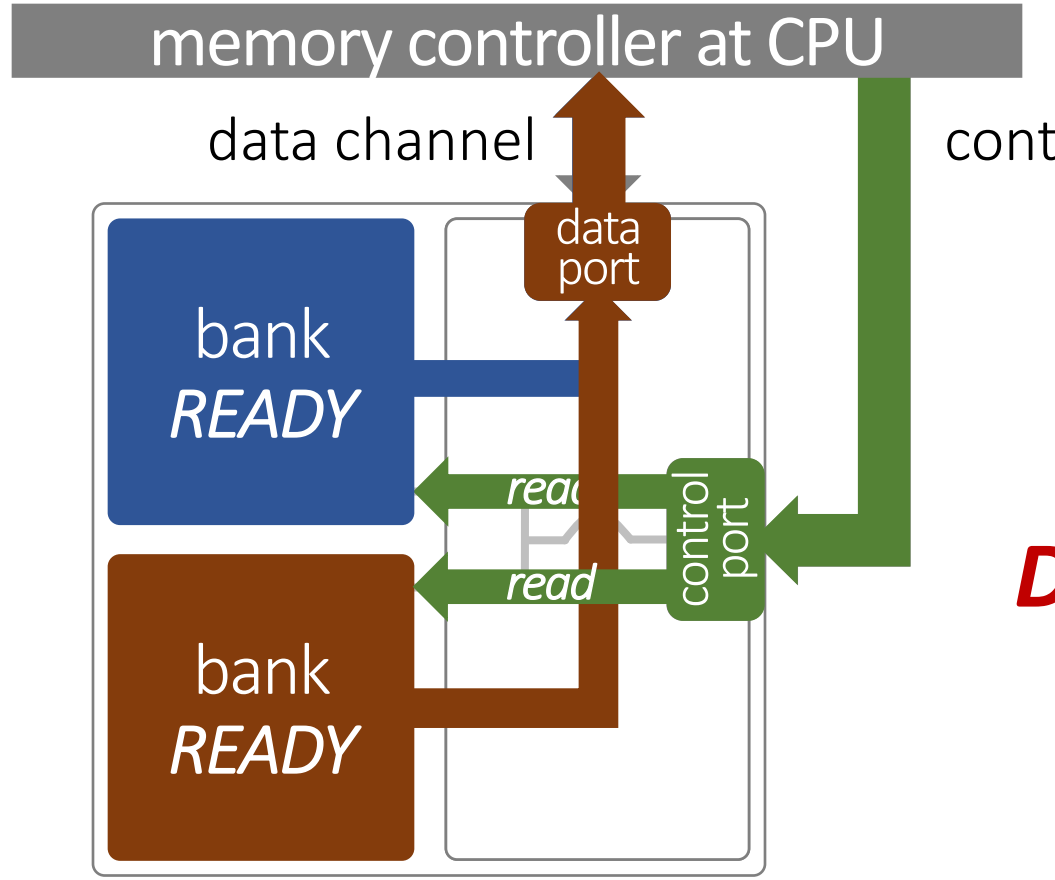
1. Problem
2. Our Approach
3. Dual-Data-Port DRAM
4. Applications for DDMA
5. Evaluation

Background: DRAM Operation



DRAM peripheral logic: *i) controls banks*, and *ii) transfers data* over memory channel

Problem: Single Data Port



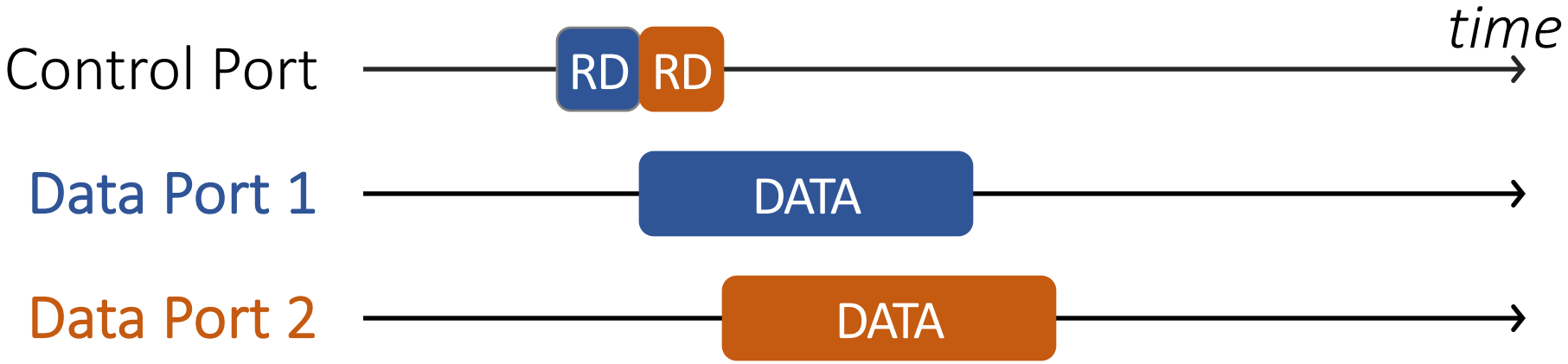
Many Banks

Single Data Port

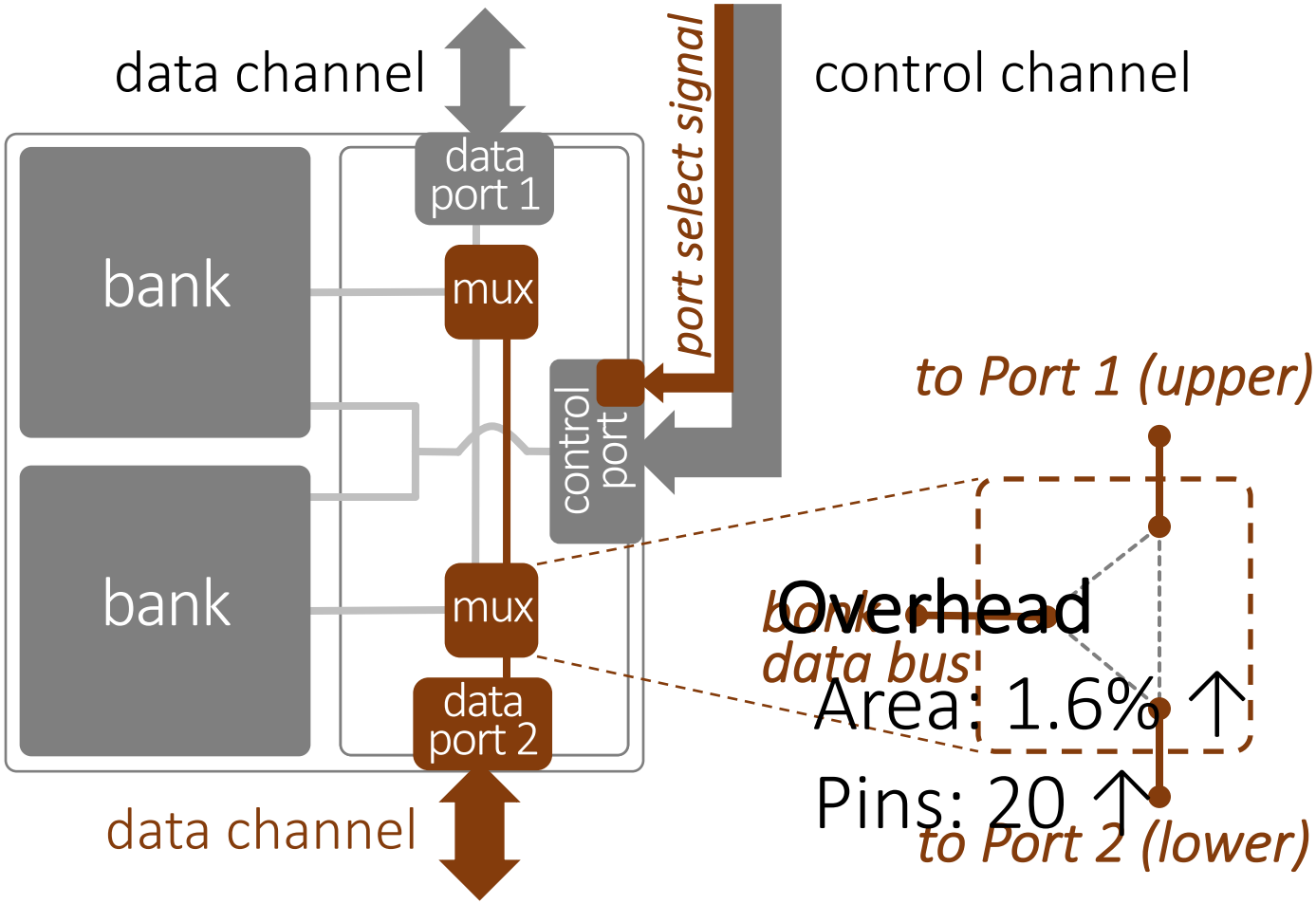
Requests are served *serially* due to ***single data port***

Problem: Single Data Port

What about a DRAM with **two data ports**?

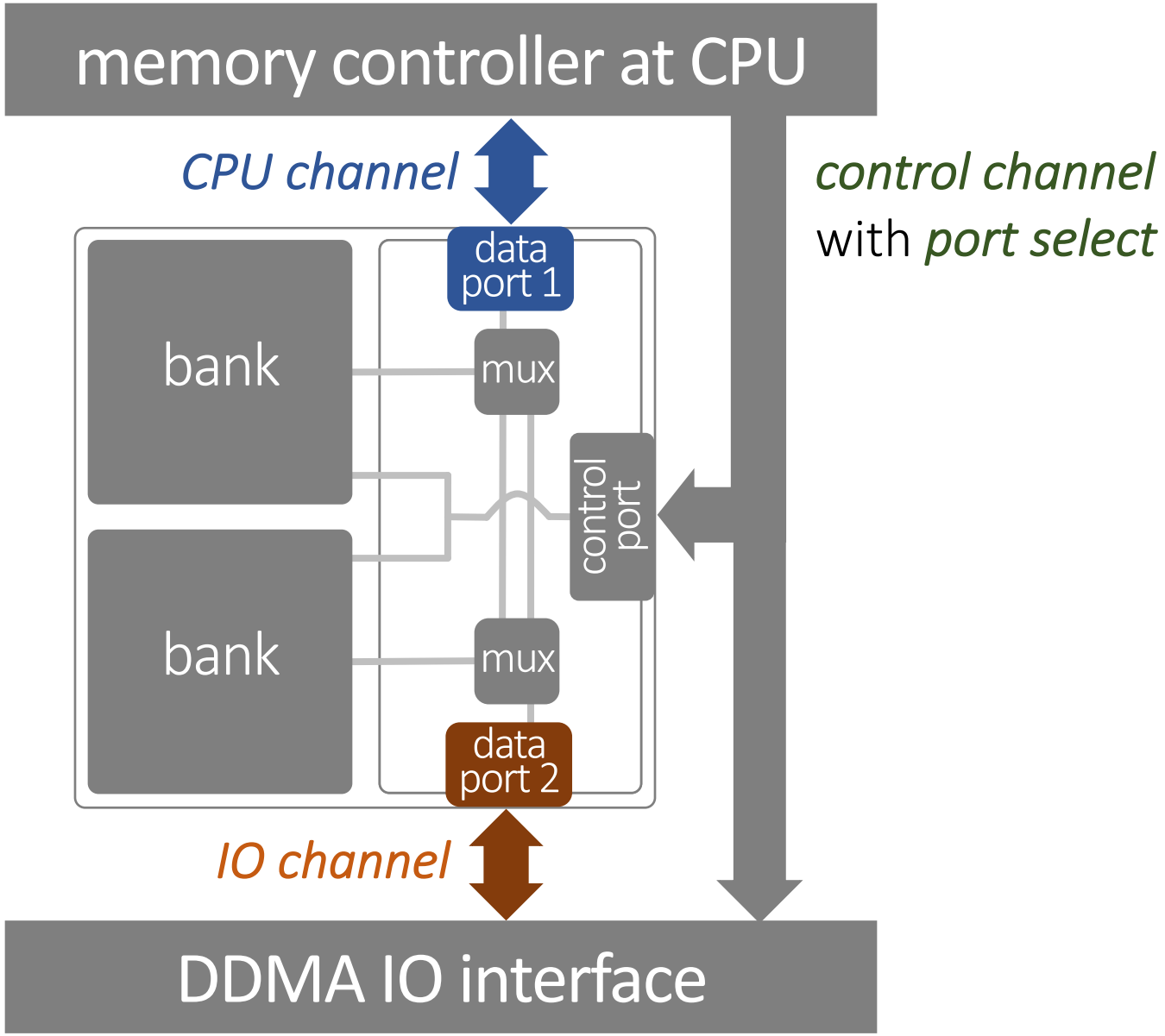


Dual-Data-Port DRAM



twice the bandwidth & *independent data ports with low overhead*

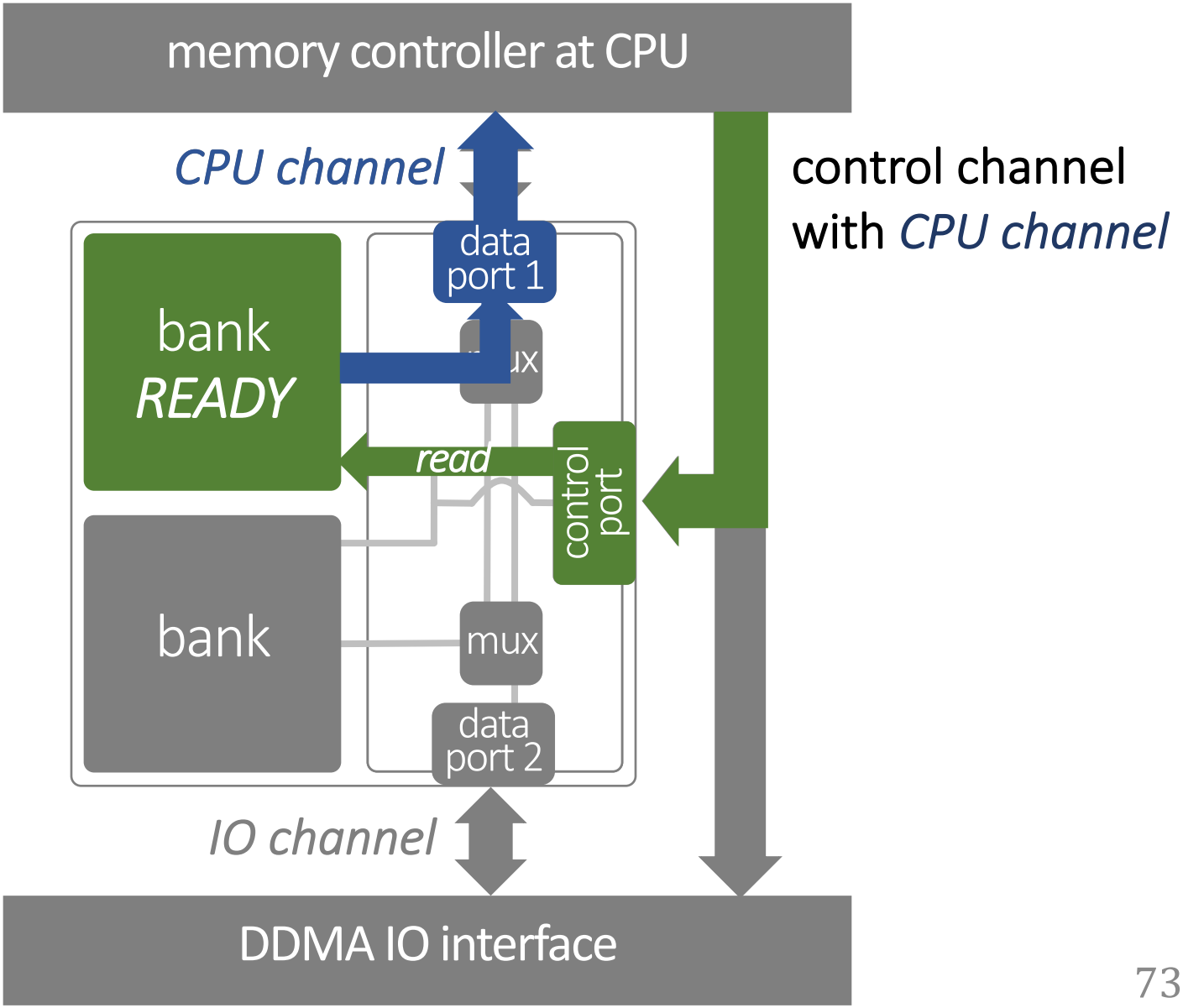
DDP-DRAM Memory System



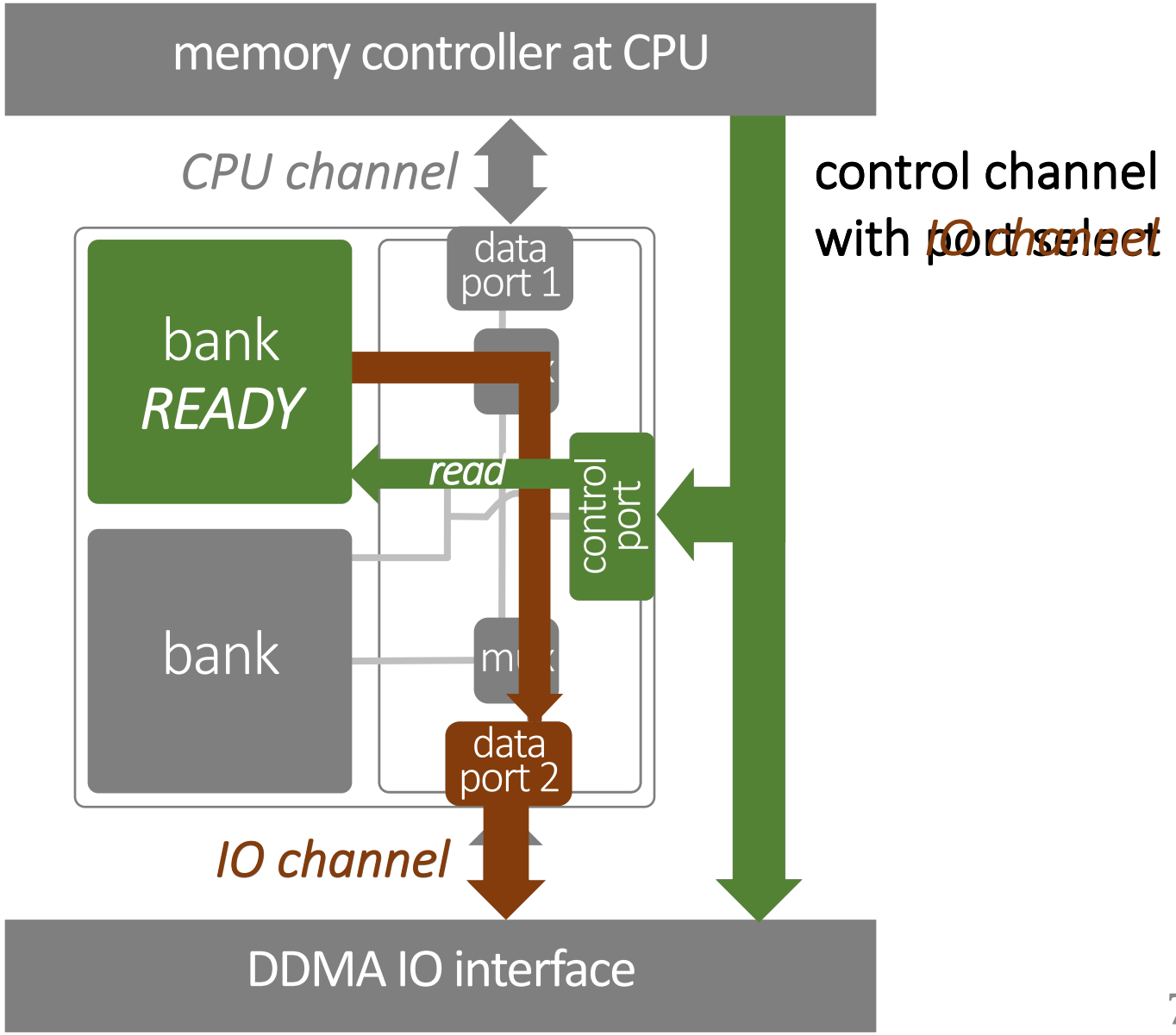
Three Data Transfer Modes

- **CPU Access**: Access through CPU channel
  - DRAM read/write with CPU port selection
- **IO Access**: Access through IO channel
  - DRAM read/write with IO port selection
- **Port Bypass**: Direct transfer between channels
  - DRAM access with port bypass selection

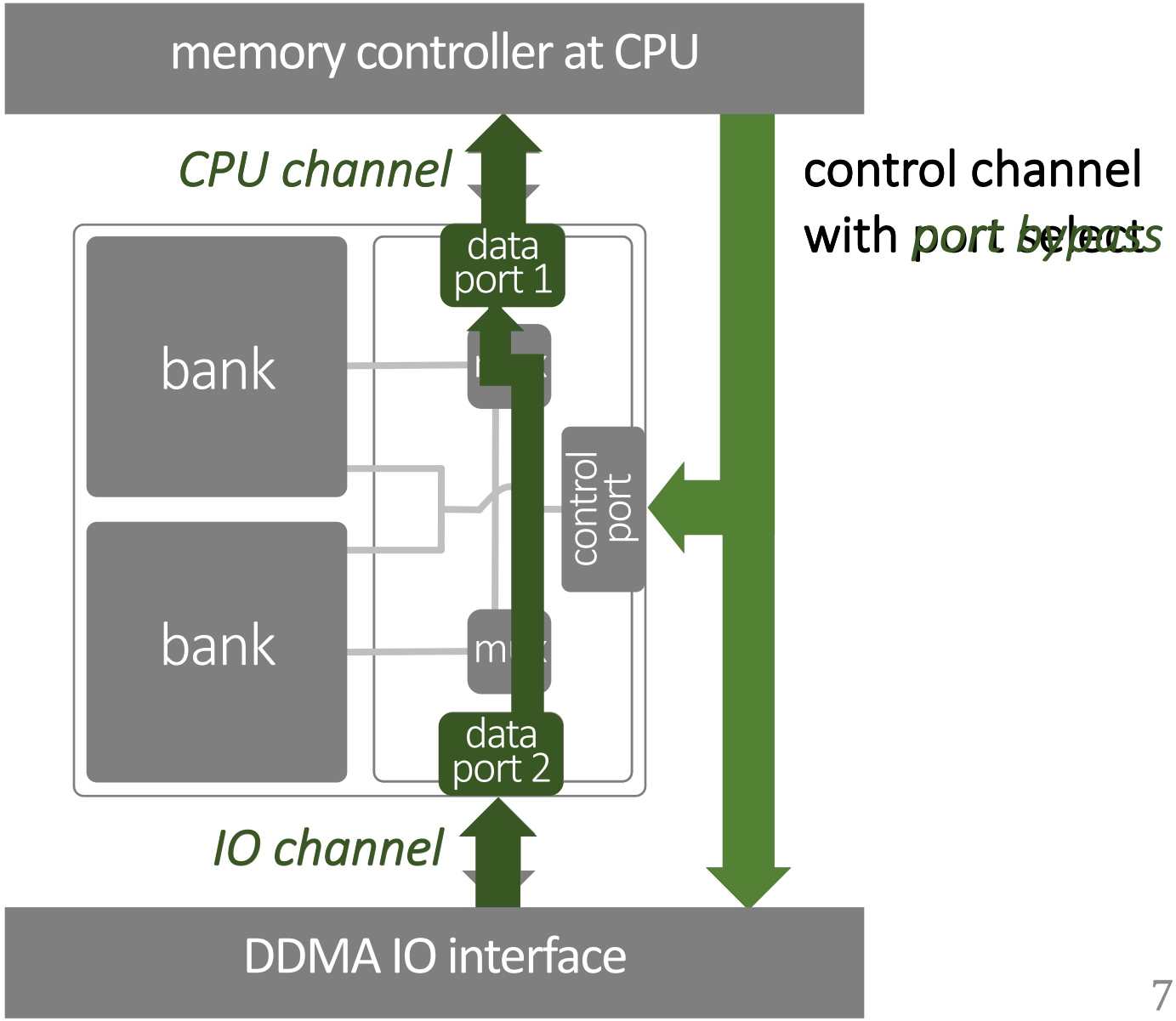
1. CPU Access Mode



2. IO Access Mode



3. Port Bypass Mode



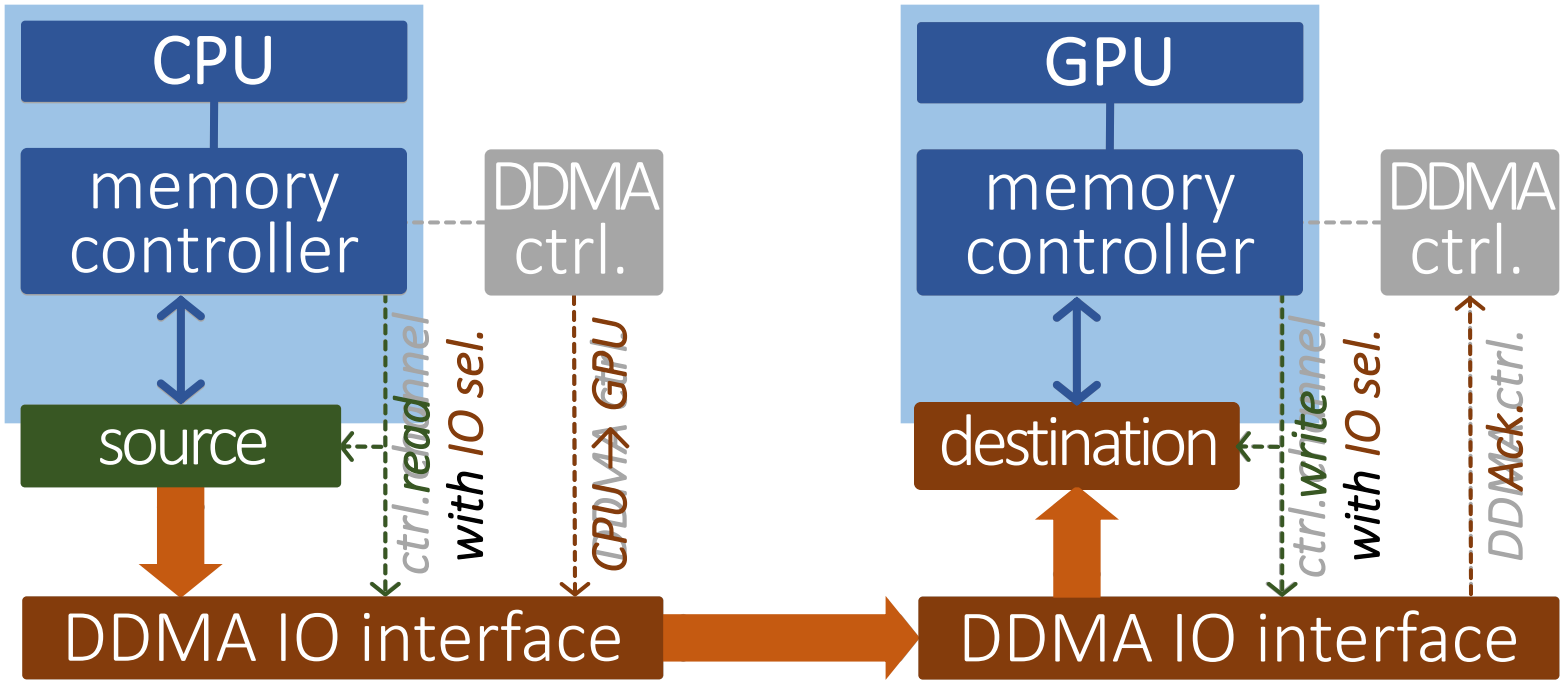
Outline

1. Problem
2. Our Approach
3. Dual-Data-Port DRAM
4. Applications for DDMA
5. Evaluation

Three Applications for DDMA

- Communication b/w Compute Units
  - CPU-GPU communication
- In-Memory Communication and Initialization
  - Bulk page copy/initialization
- Communication b/w Memory and Storage
  - Serving page fault/file read & write

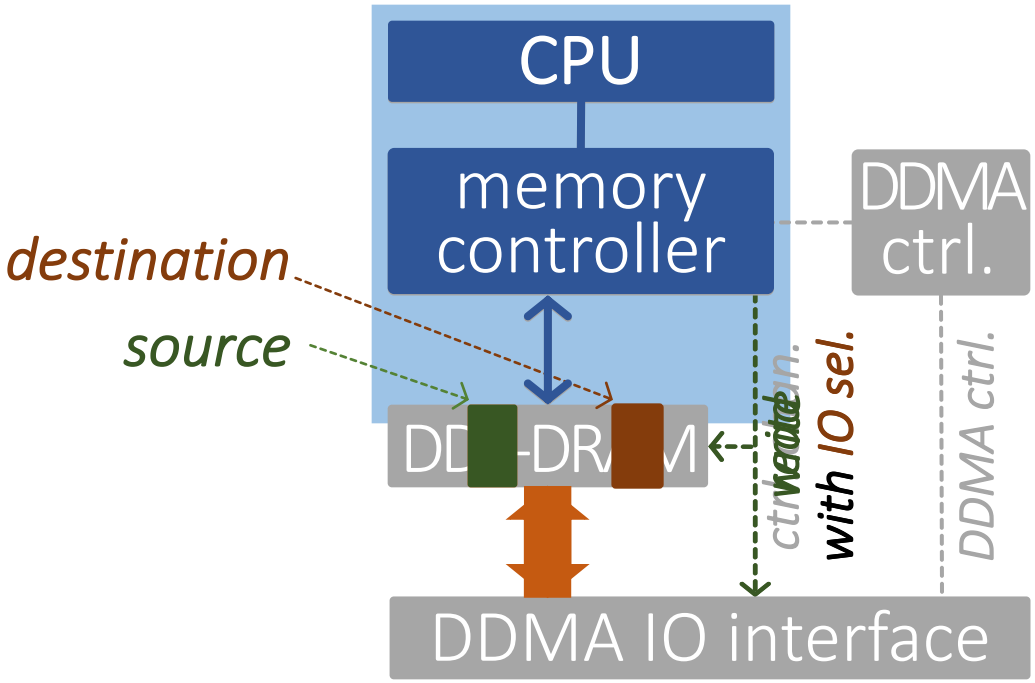
1. Compute Unit ↔ Compute Unit



Transfer data through DDMA

without interfering w/ CPU/GPU memory accesses

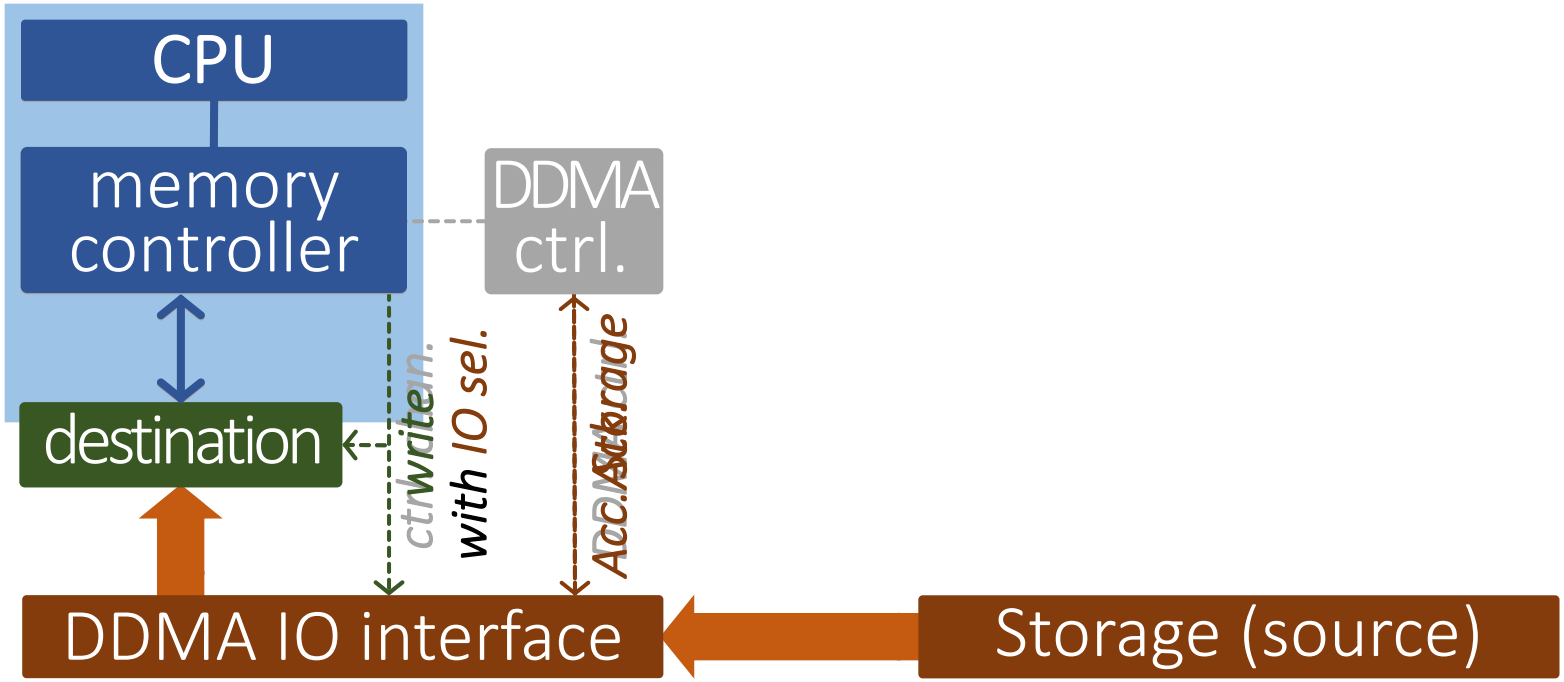
2. In-Memory Communication



Transfer data in DRAM through DDAM

without interfering with CPU memory accesses

3. Memory ↔ Storage



Transfer data from storage through DDMA

without interfering with CPU memory accesses

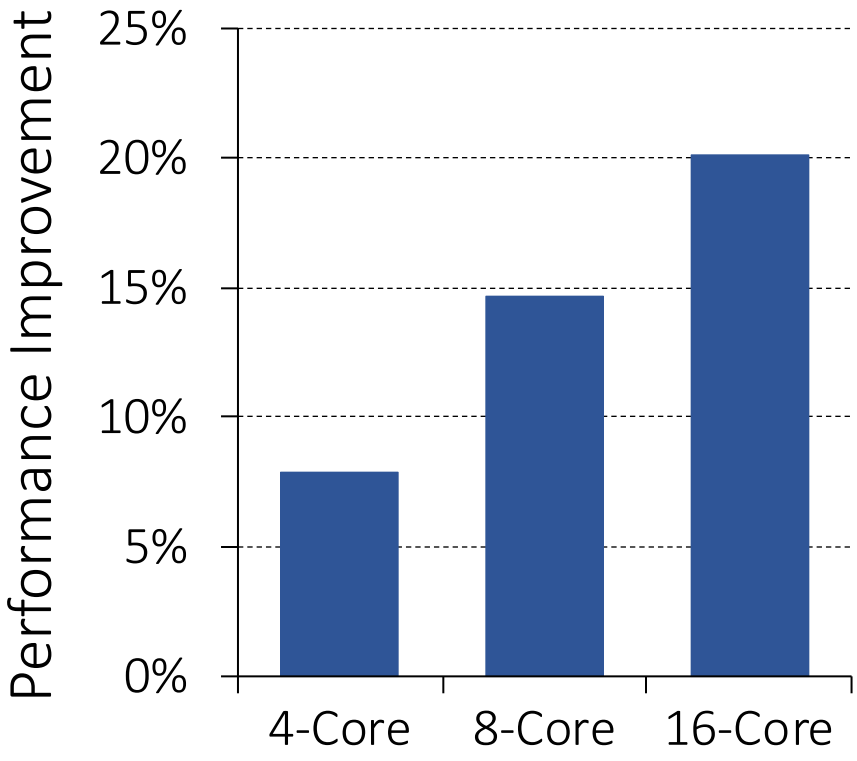
Outline

1. Problem
2. Our Approach
3. Dual-Data-Port DRAM
4. Applications for DDMA
5. Evaluation

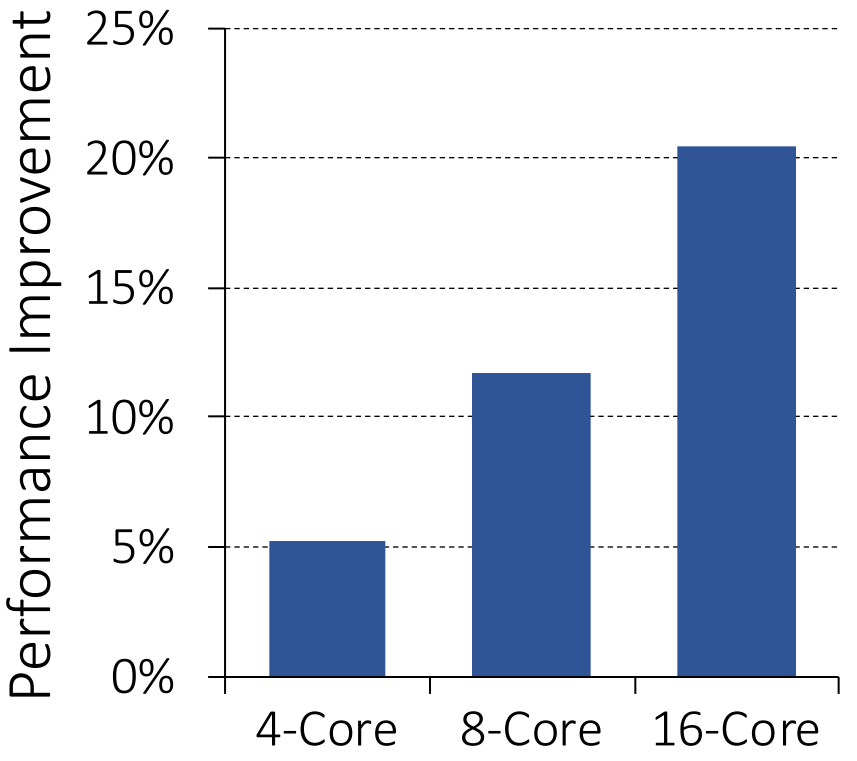
Evaluation Methods

- System
  - Processor: 4 – 16 cores
  - LLC: 16-way associative, 512KB private cache-slice/core
  - Memory: 1 – 4 ranks and 1 – 4 channels
- Workloads
  - **Memory intensive**:
    SPEC CPU2006, TPC, stream (31 benchmarks)
  - **CPU-GPU communication intensive**:
    polybench (8 benchmarks)
  - **In-memory communication intensive**:
    apache, bootup, compiler, filecopy, mysql, fork, shell, memcached (8 in total)

Performance (2 Channel, 2 Rank)



CPU-GPU Comm.-Intensive

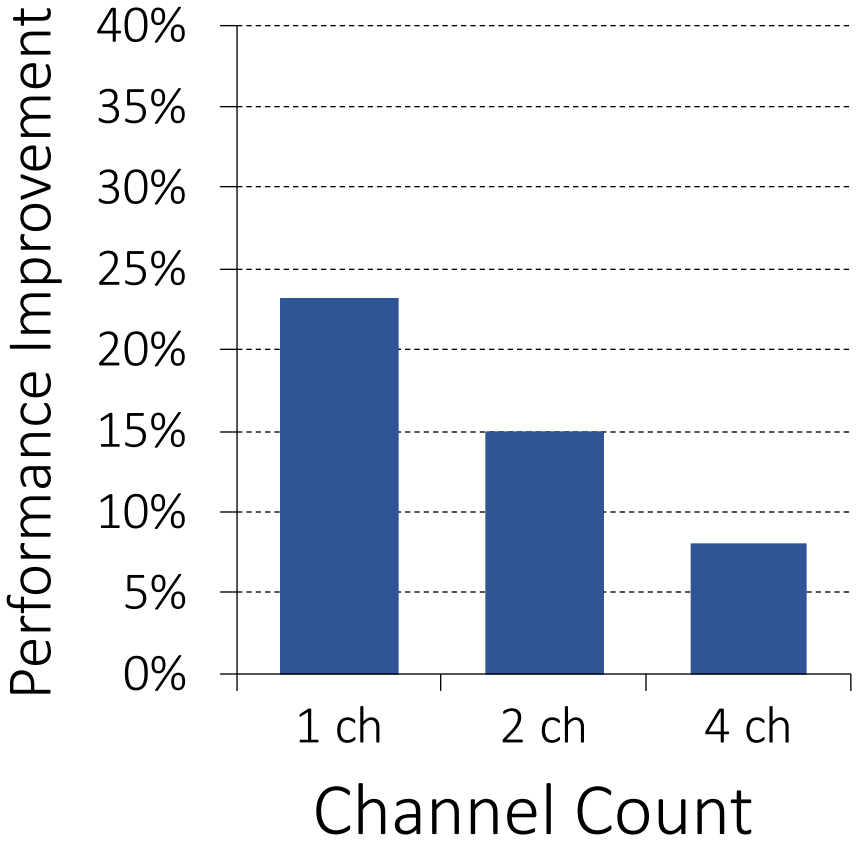


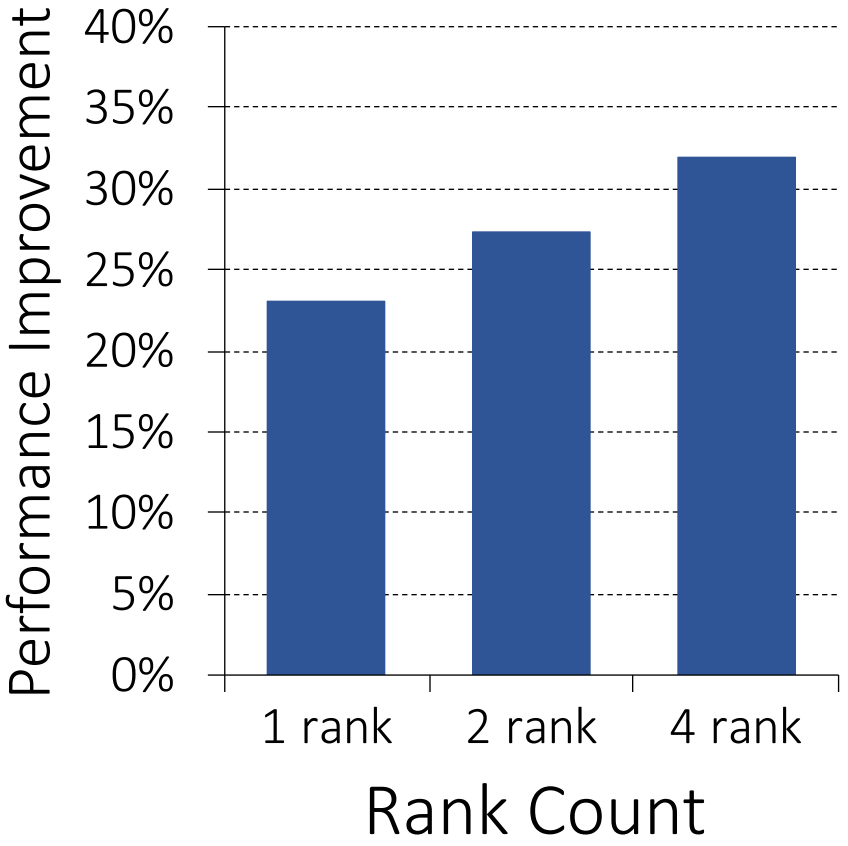
In-Memory Comm.-Intensive

High performance improvement

More performance improvement at *higher core count*

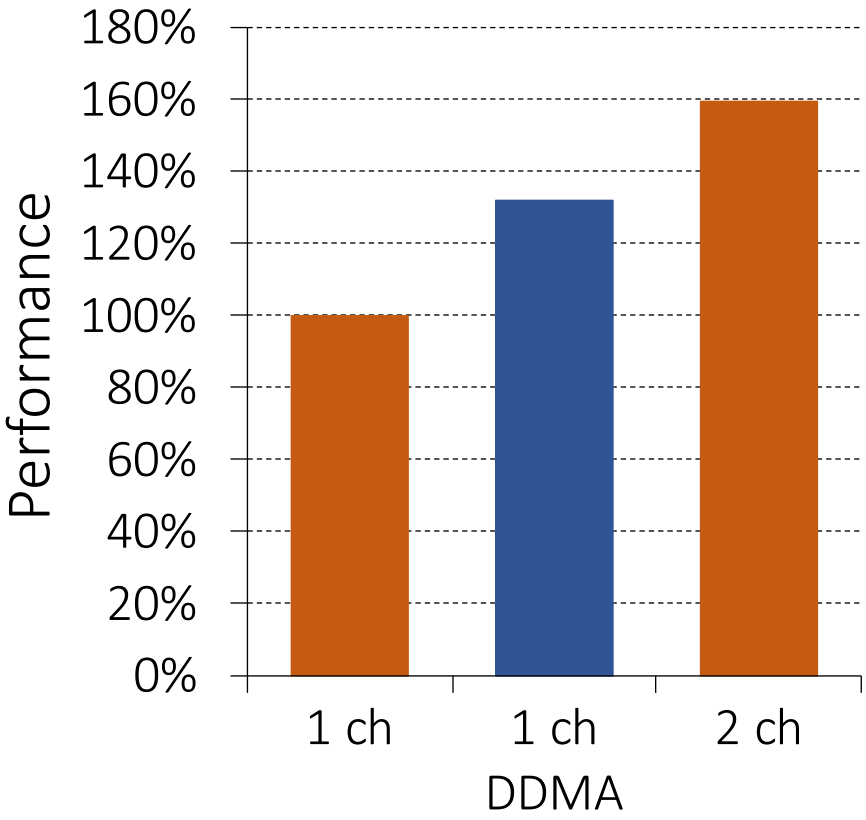
Performance on Various Systems

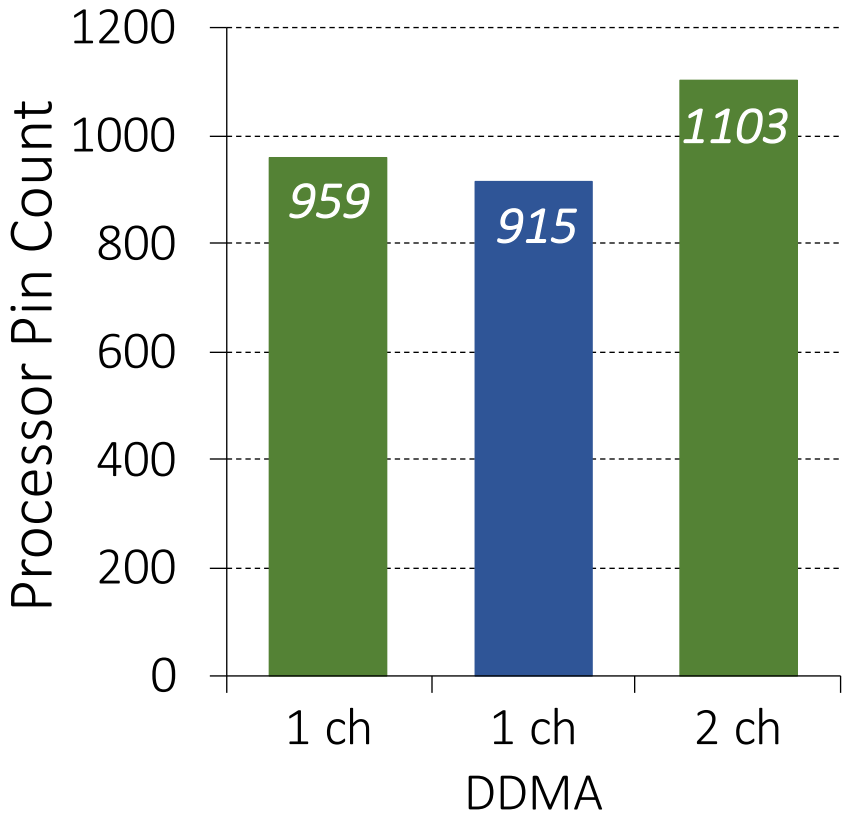




Performance increases with rank count

DDMA vs. Dual Channel

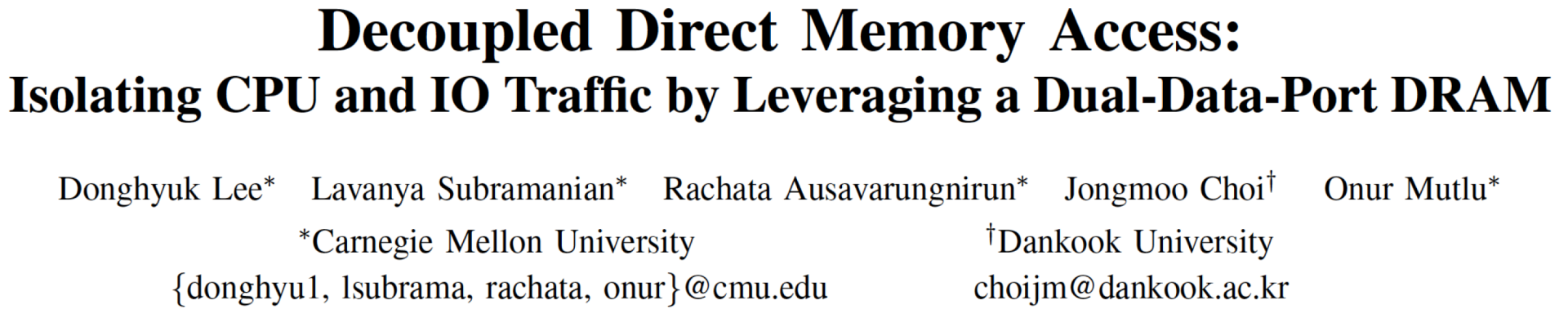




DDMA achieves *higher performance* at *lower processor pin count*

More on Decoupled DMA

- Donghyuk Lee, Lavanya Subramanian, Rachata Ausavarungnirun, Jongmoo Choi, and Onur Mutlu, **"Decoupled Direct Memory Access: Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM"** *Proceedings of the 24th International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, San Francisco, CA, USA, October 2015. [Slides (pptx) (pdf)]



Decoupled Direct Memory Access:
Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM

Donghyuk Lee* Lavanya Subramanian* Rachata Ausavarungnirun* Jongmoo Choi† Onur Mutlu*
*Carnegie Mellon University
{donghyu1, lsubrama, rachata, onur}@cmu.edu
†Dankook University
choijm@dankook.ac.kr

Computer Architecture

Lecture 18a: Memory Interference and Quality of Service III

Prof. Onur Mutlu
ETH Zürich

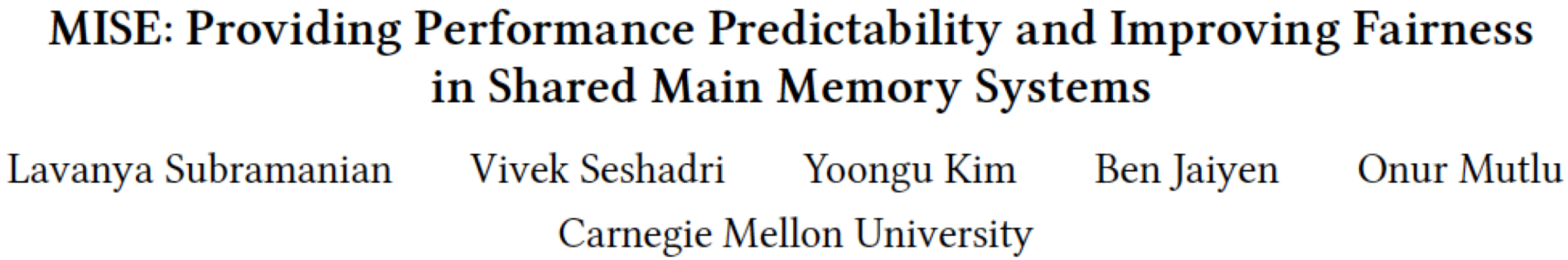
Fall 2018

22 November 2018

Predictable Performance Again: Strong Memory Service Guarantees

Remember MISE?

- Lavanya Subramanian, Vivek Seshadri, Yoongu Kim, Ben Jaiyen, and Onur Mutlu,
  "MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems"
  *Proceedings of the 19th International Symposium on High-Performance Computer Architecture (**HPCA**)*, Shenzhen, China, February 2013. Slides (pptx)


MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems

Lavanya Subramanian Vivek Seshadri Yoongu Kim Ben Jaiyen Onur Mutlu

Carnegie Mellon University

Extending Slowdown Estimation to Caches

- How do we extend the MISE model to include shared cache interference?
- Answer: Application Slowdown Model
- Lavanya Subramanian, Vivek Seshadri, Arnab Ghosh, Samira Khan, and Onur Mutlu,
  "The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory"
  *Proceedings of the 48th International Symposium on Microarchitecture (**MICRO**)*, Waikiki, Hawaii, USA, December 2015.
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
  [Source Code]

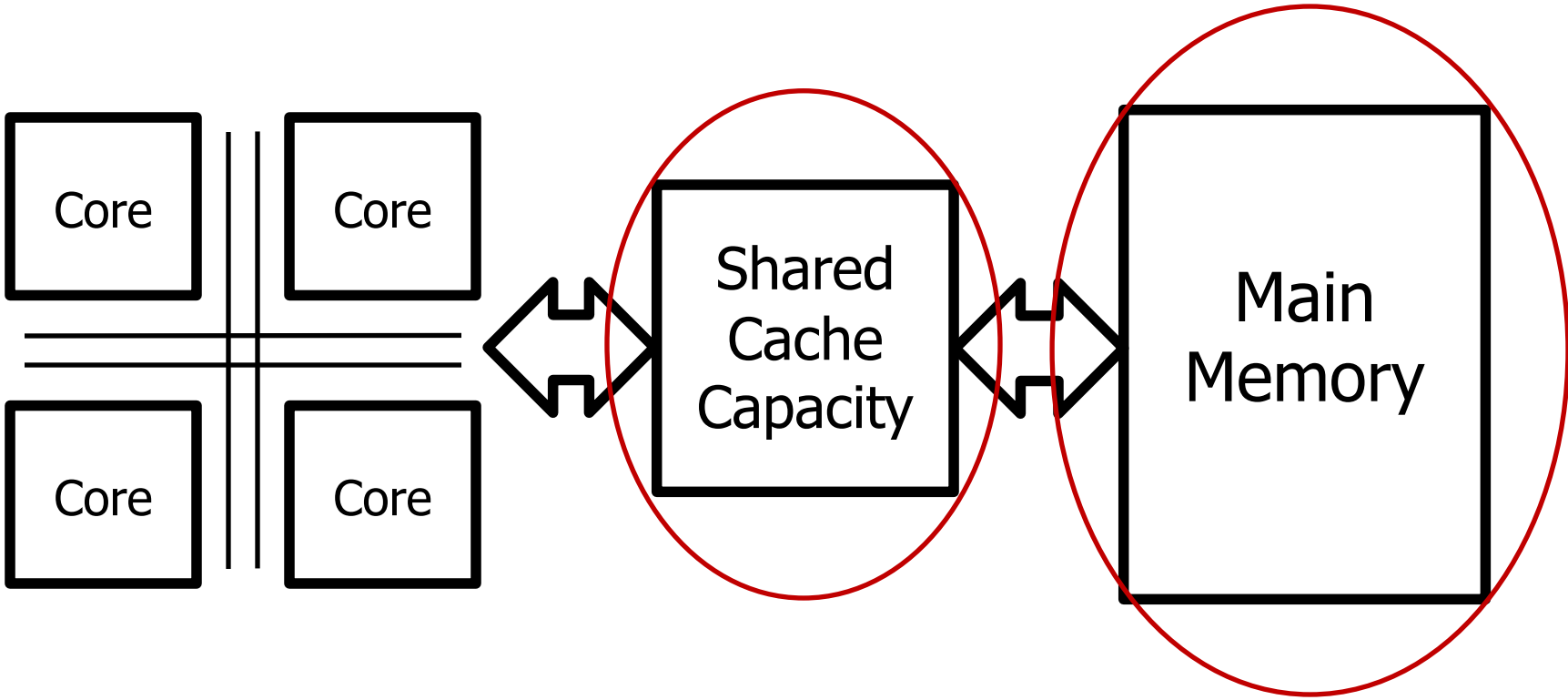
Application Slowdown Model

Quantifying and Controlling Impact of Interference at Shared Caches and Main Memory

Lavanya Subramanian, Vivek Seshadri,
Arnab Ghosh, Samira Khan, Onur Mutlu

SAFARI Carnegie Mellon intel UNIVERSITY OF VIRGINIA 1819

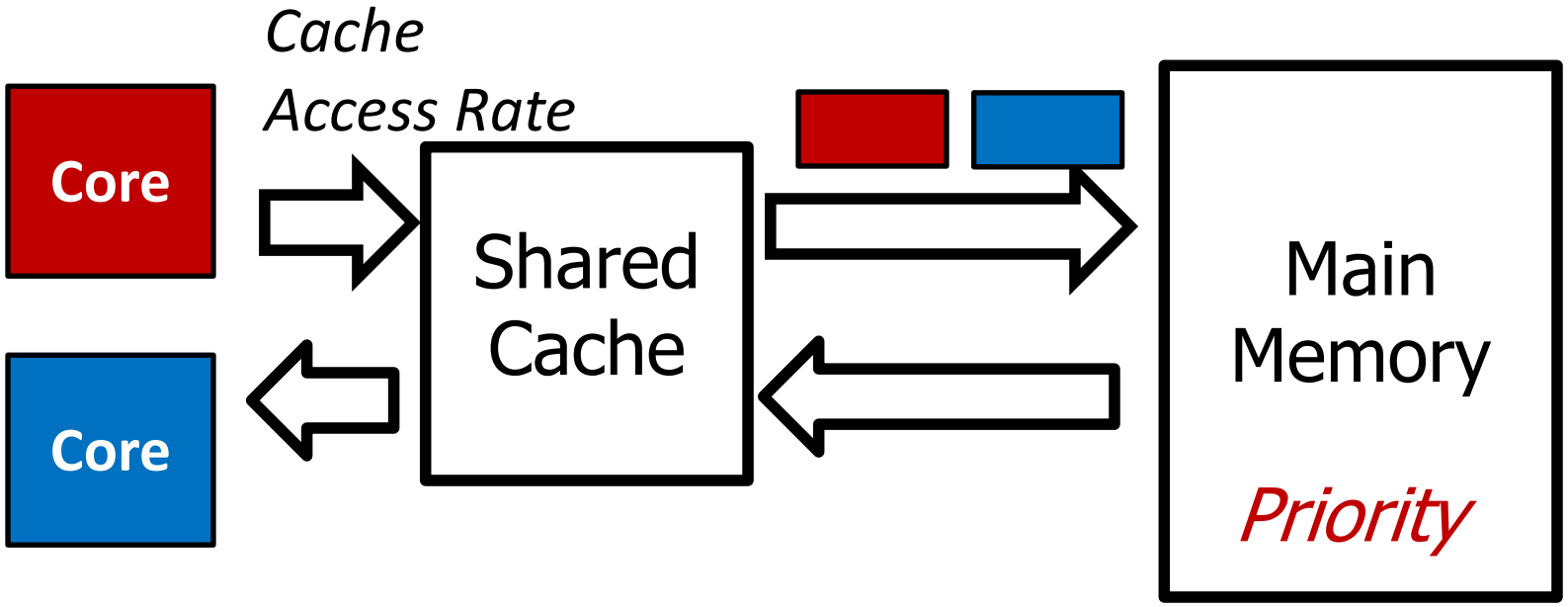
Shared Cache and Memory Contention



$$\text{Slowdown} = \frac{\text{Request Service Rate}_{\text{Alone}}}{\text{Request Service Rate}_{\text{Shared}}}$$

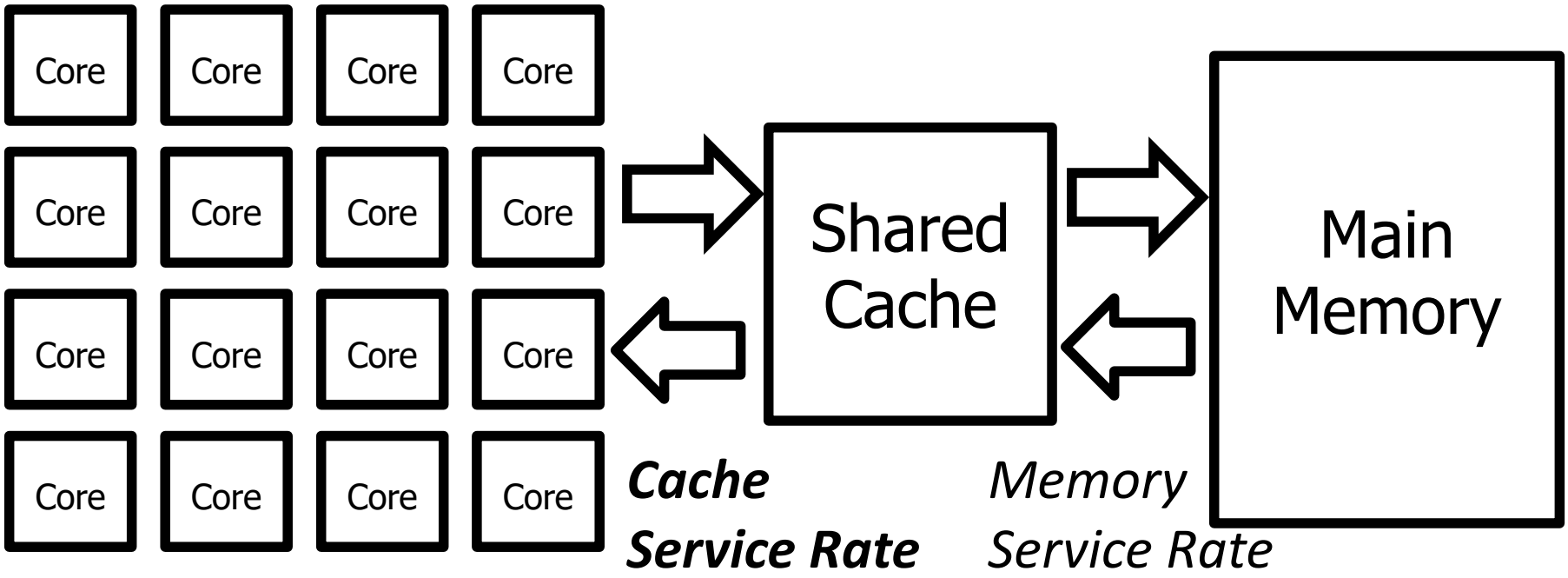
MISE [HPCA'13]

Cache Capacity Contention

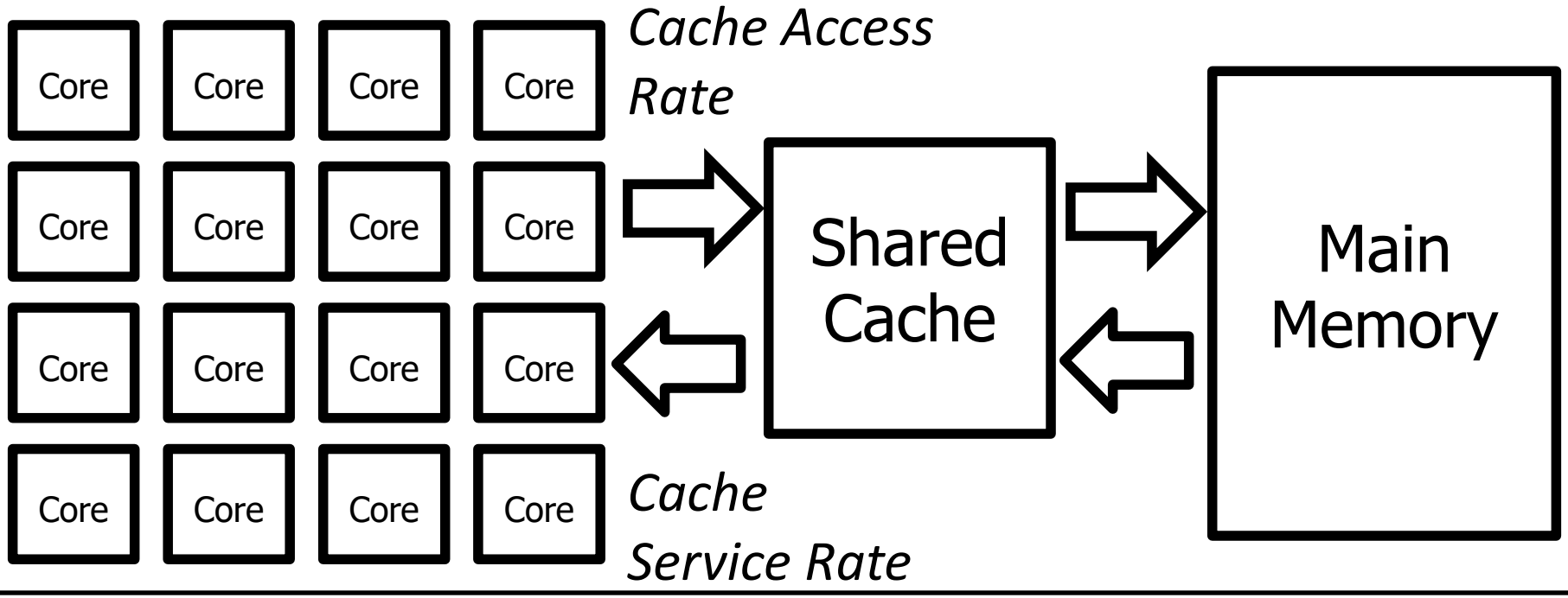


Applications evict each other's blocks from the shared cache

Estimating Cache and Memory Slowdowns

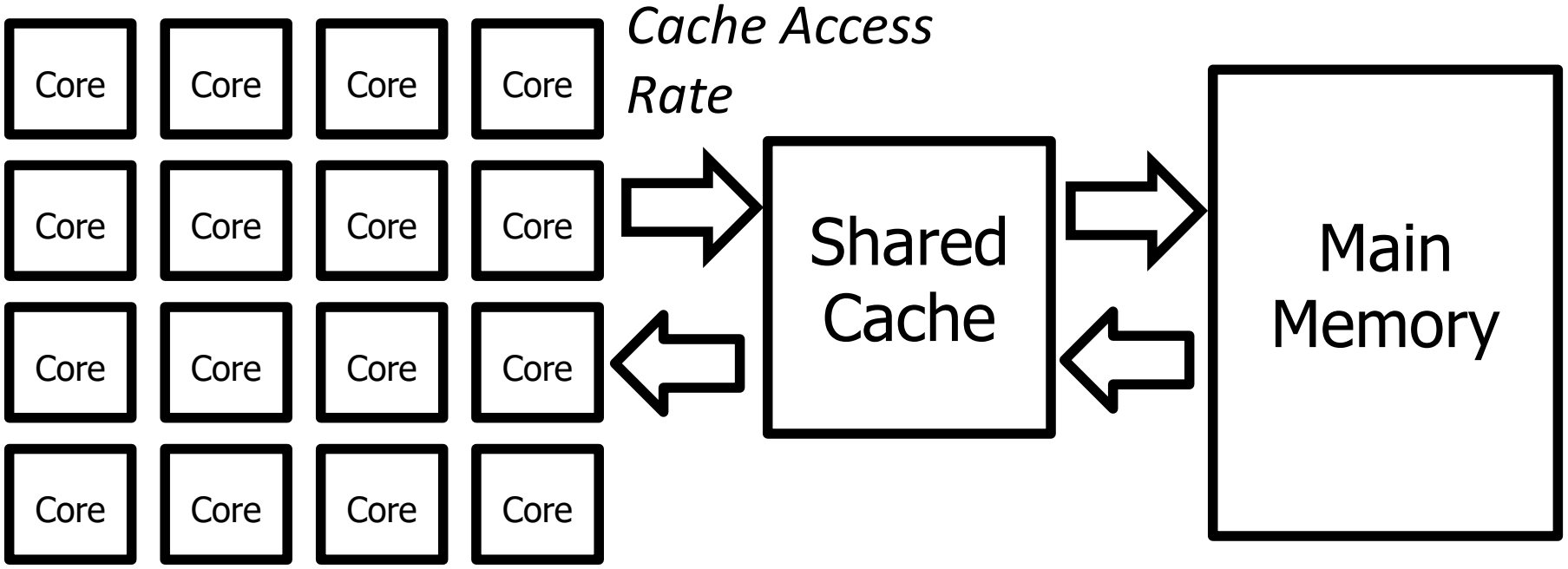


Service Rates vs. Access Rates



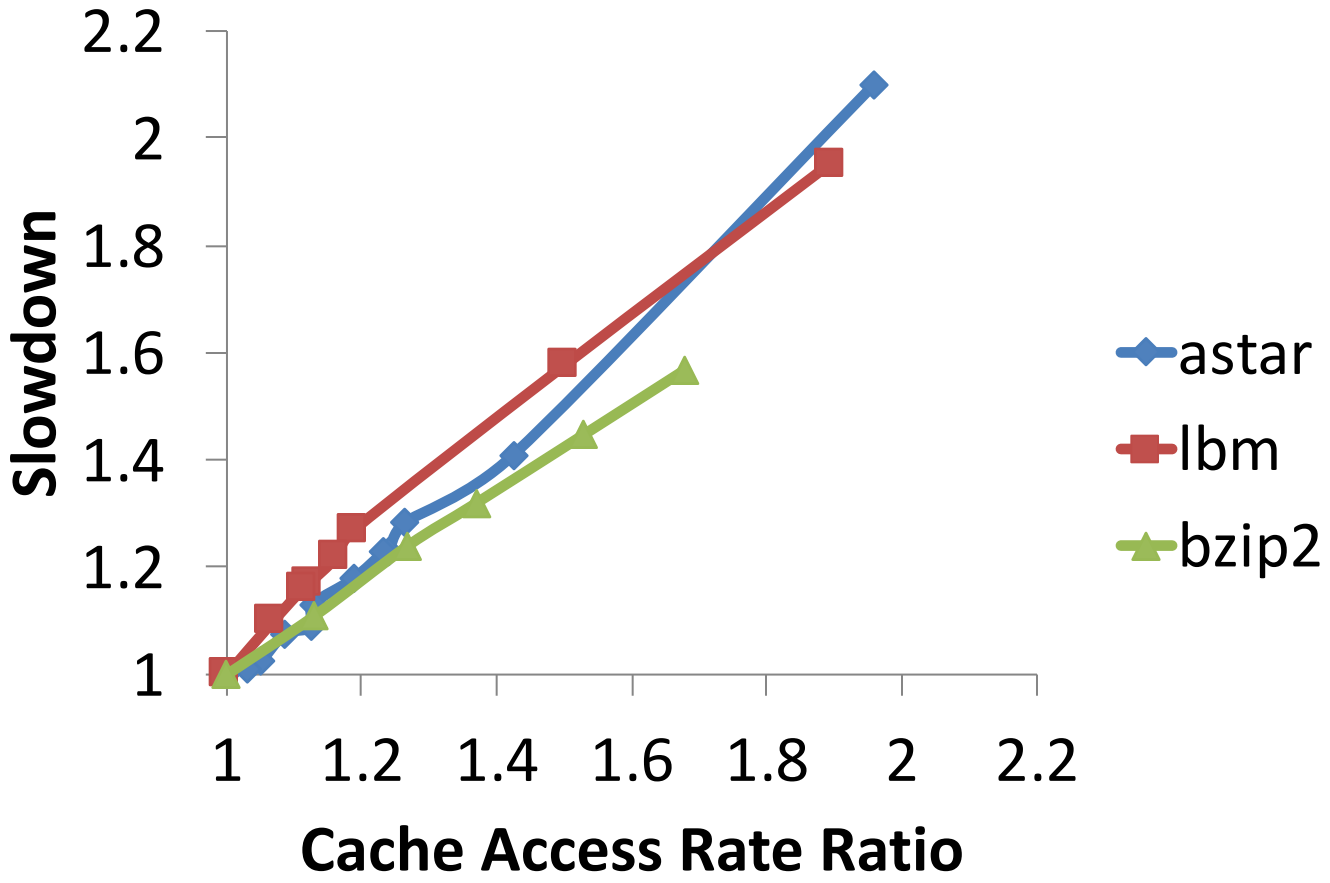
Request service and access rates are tightly coupled

The Application Slowdown Model



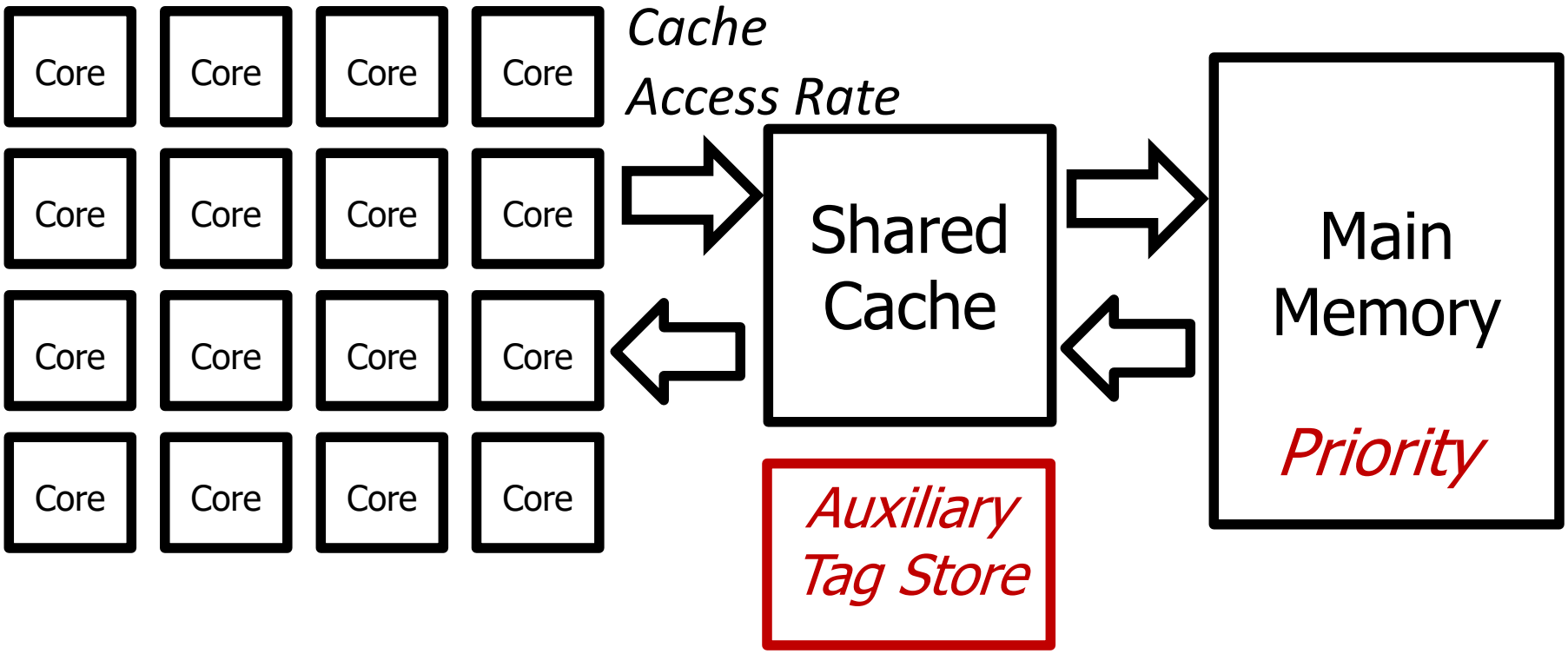
$$\text{Slowdown} = \frac{\text{Cache Access Rate}_{\text{Alone}}}{\text{Cache Access Rate}_{\text{Shared}}}$$

Real System Studies: Cache Access Rate vs. Slowdown

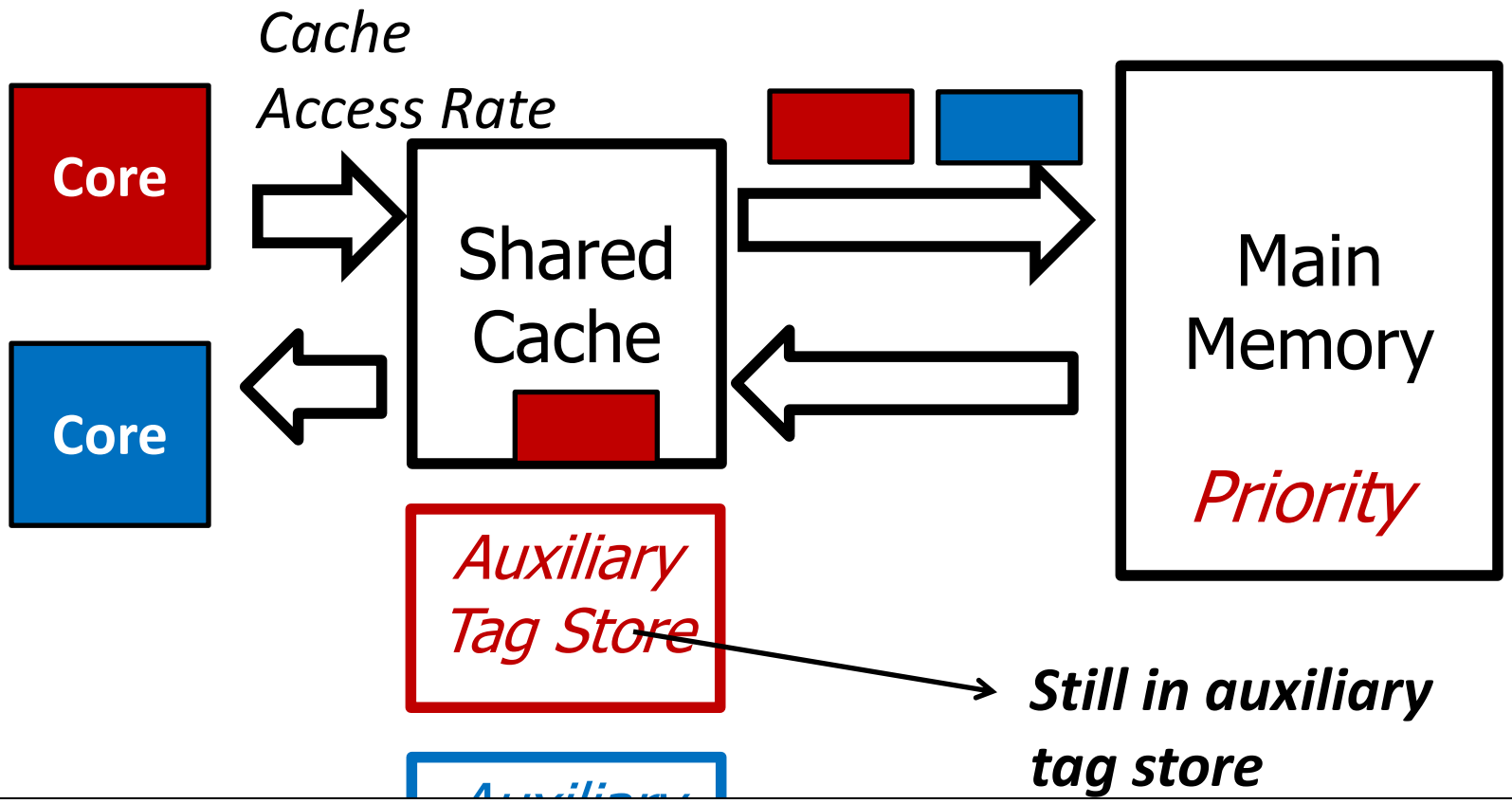


Challenge

How to estimate alone cache access rate?



Auxiliary Tag Store



Auxiliary tag store tracks such ***contention misses***

Accounting for Contention Misses

- Revisiting alone memory request service rate

$$\text{Alone Request Service Rate of an Application} = \frac{\text{\# Requests During High Priority Epochs}}{\text{\# High Priority Cycles}}$$

Cycles serving contention misses should not count as high priority cycles

Alone Cache Access Rate Estimation

$$\text{Cache Access Rate}_{\text{Alone}} \text{ of an Application} = \frac{\#\text{ Requests During High Priority Epochs}}{\#\text{ High Priority Cycles} - \#\text{Cache Contention Cycles}}$$

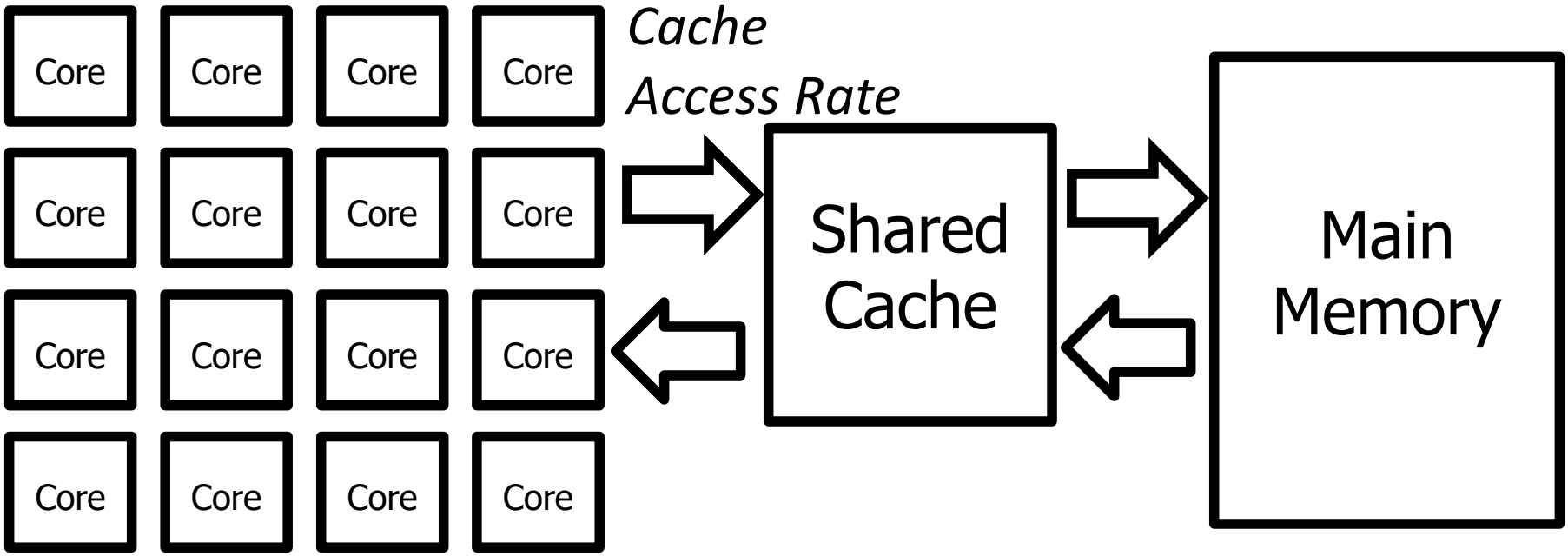
Cache Contention Cycles: Cycles spent serving contention misses

$$\text{Cache Contention Cycles} = \#\text{ Contention Misses} \times \text{Average Memory Service Time}$$

From auxiliary tag store when given high priority

Measured when given high priority

Application Slowdown Model (ASM)



$$\text{Slowdown} = \frac{\text{Cache Access Rate}_{\text{Alone}}}{\text{Cache Access Rate}_{\text{Shared}}}$$

Previous Work on Slowdown Estimation

- Previous work on slowdown estimation
  - **STFM** (Stall Time Fair Memory) Scheduling [Mutlu et al., MICRO '07]
  - **FST** (Fairness via Source Throttling) [Ebrahimi et al., ASPLOS '10]
  - **Per-thread Cycle Accounting** [Du Bois et al., HiPEAC '13]

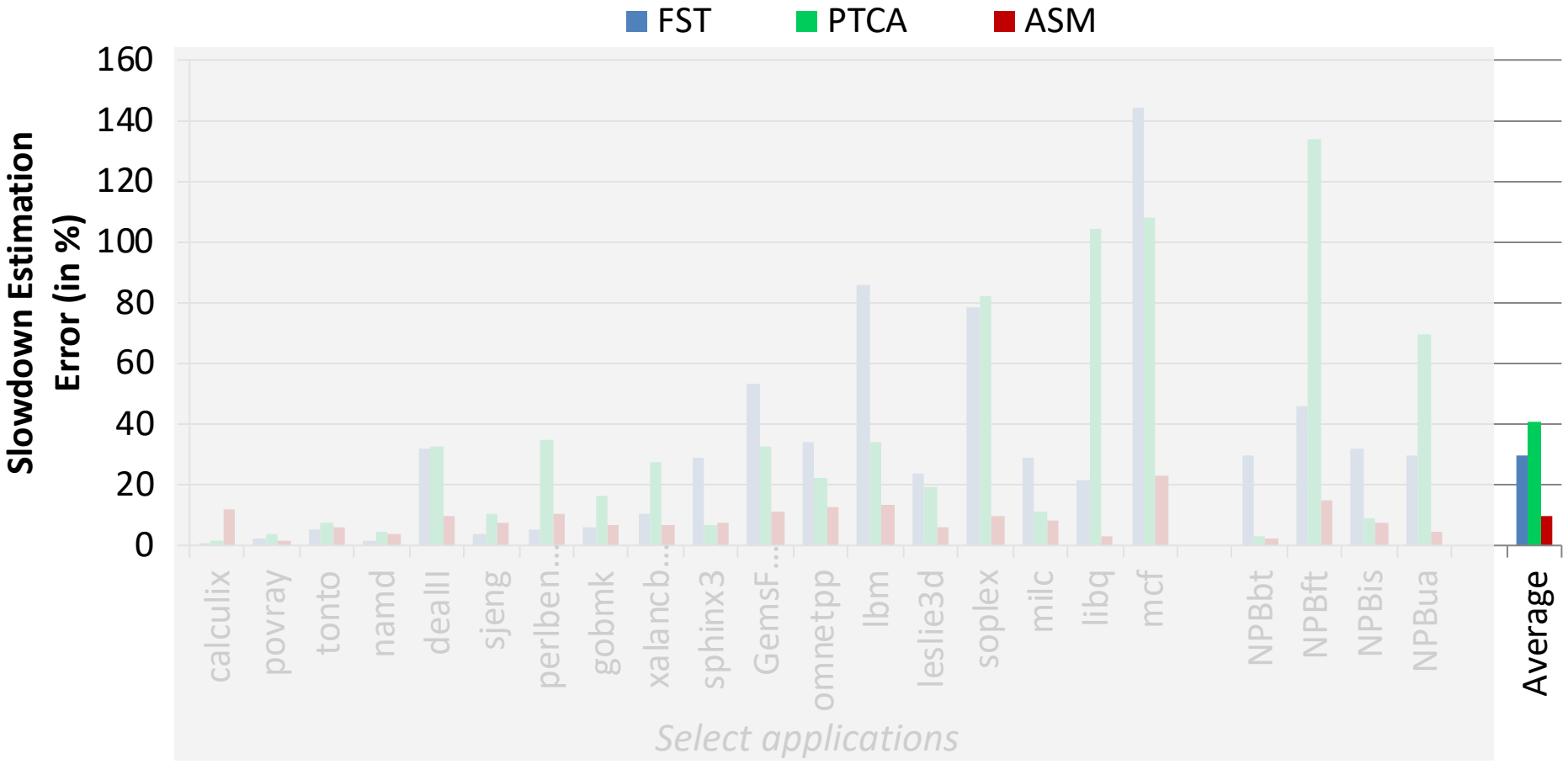
- Basic Idea:

$$\text{Slowdown} = \frac{\text{Execution Time}_{\text{Alone}}}{\text{Execution Time}_{\text{Shared}}}$$

Count interference experienced by each request → Difficult

ASM's estimates are much more coarse grained → Easier

Model Accuracy Results

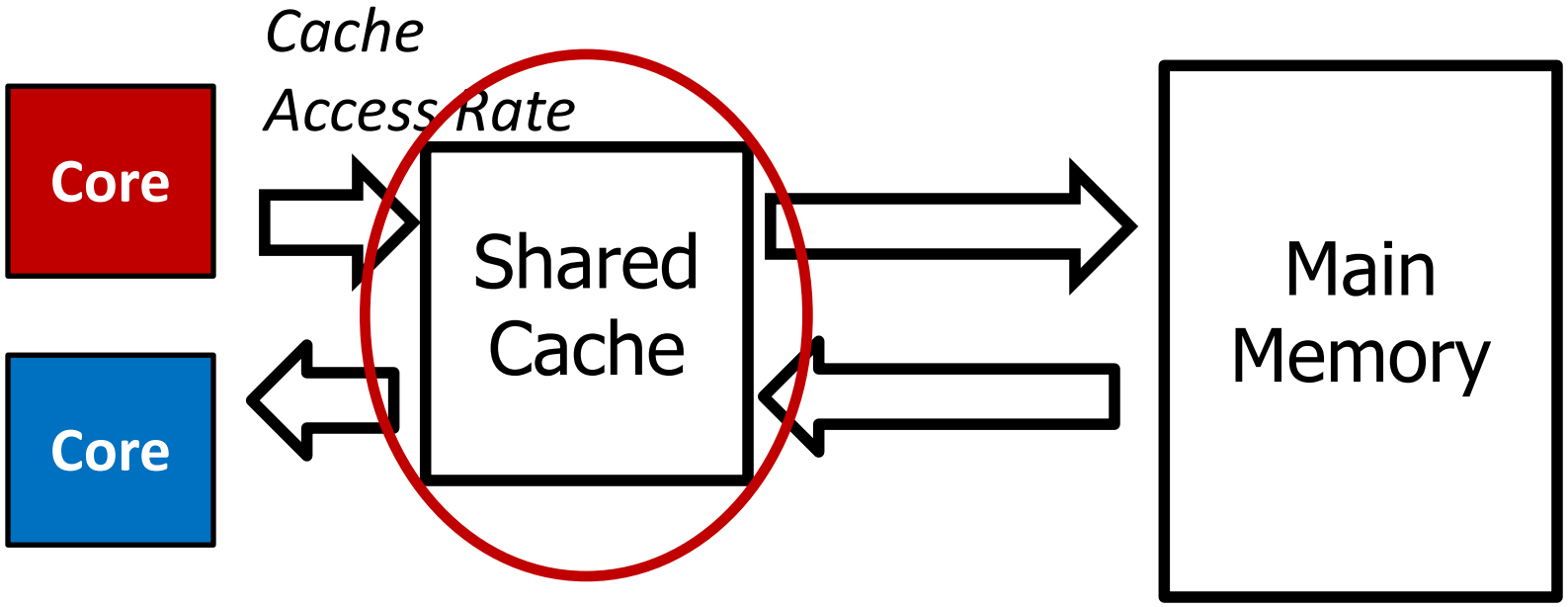


Average error of ASM's slowdown estimates: 10%

Leveraging ASM's Slowdown Estimates

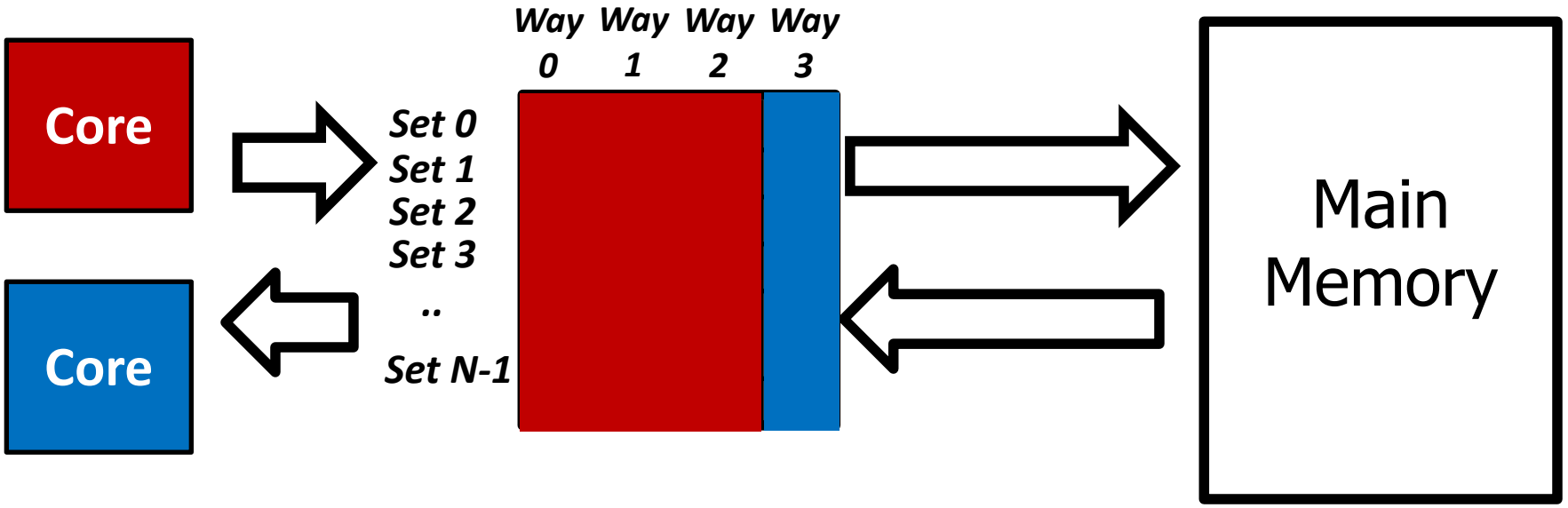
- *Slowdown-aware resource allocation for high performance and fairness*
- *Slowdown-aware resource allocation to bound application slowdowns*
- *VM migration and admission control schemes* ***[VEE '15]***
- *Fair billing schemes in a commodity cloud*

Cache Capacity Partitioning



Goal: Partition the shared cache among applications to mitigate contention

Cache Capacity Partitioning



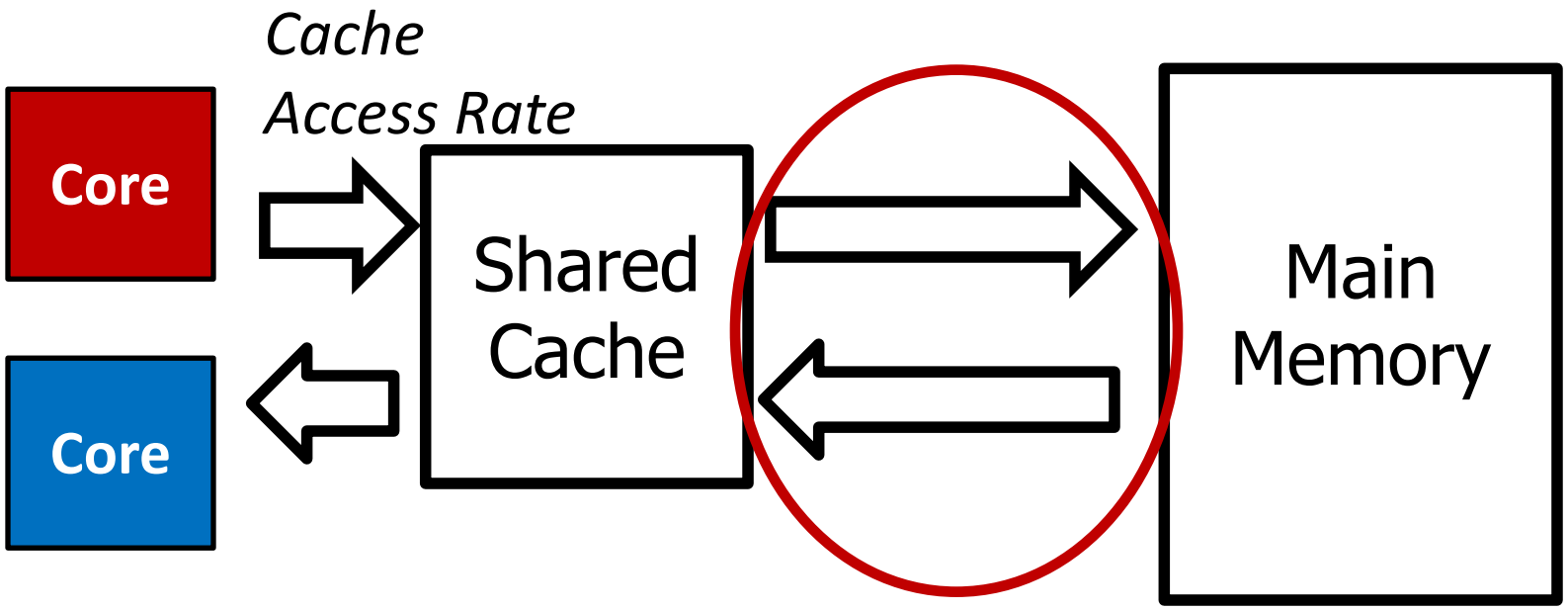
Previous partitioning schemes optimize for miss count

Problem: Not aware of performance and slowdowns

ASM-Cache: Slowdown-aware Cache Way Partitioning

- *Key Requirement: Slowdown estimates for all possible way partitions*
- *Extend ASM to estimate slowdown for all possible cache way allocations*
- *Key Idea: Allocate each way to the application whose slowdown reduces the most*

Memory Bandwidth Partitioning



Goal: Partition the main memory bandwidth among applications to mitigate contention

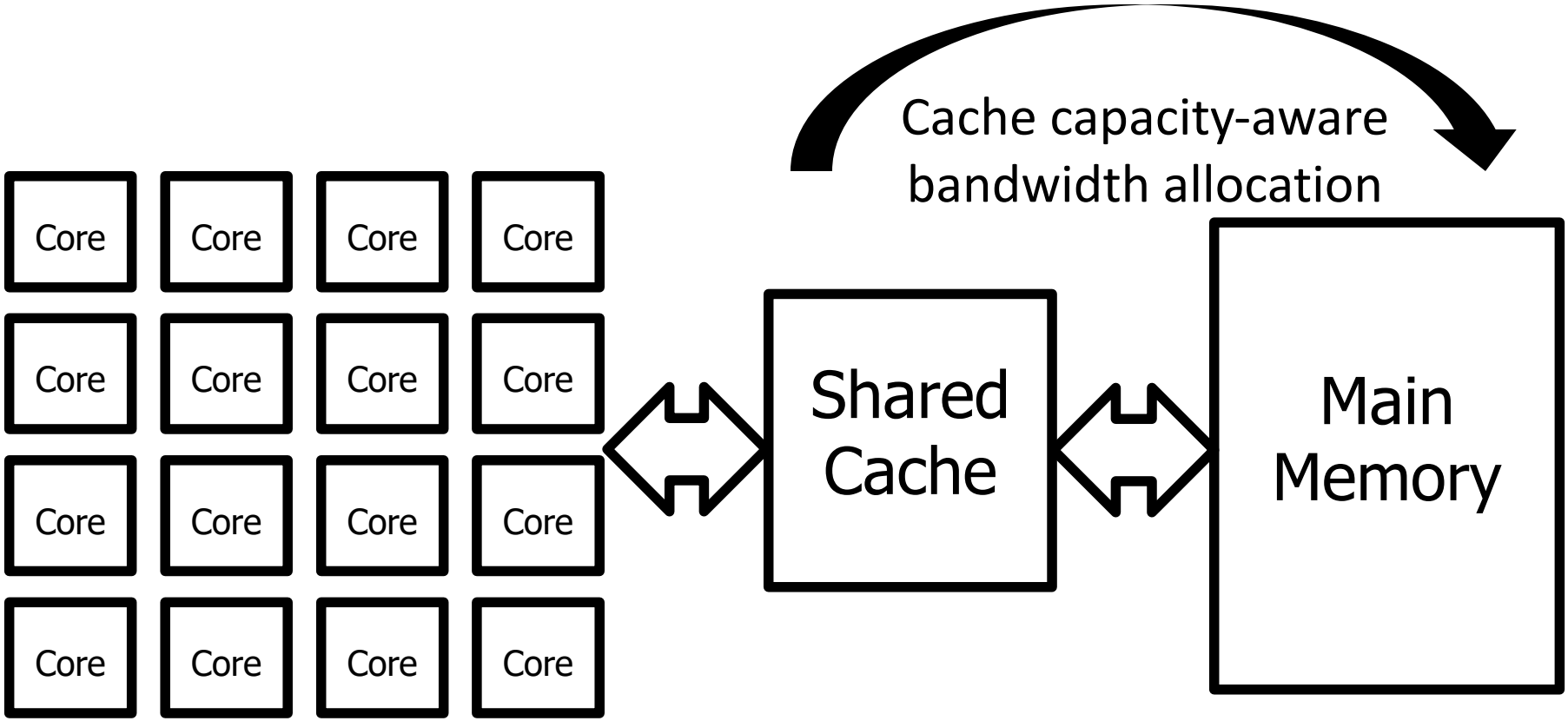
ASM-Mem: Slowdown-aware Memory Bandwidth Partitioning

- *Key Idea: Allocate high priority proportional to an application's slowdown*

$$\text{High Priority Fraction}_i = \frac{\text{Slowdown}_i}{\sum_j \text{Slowdown}_j}$$

- *Application i's requests given highest priority at the memory controller for its fraction*

Coordinated Resource Allocation Schemes

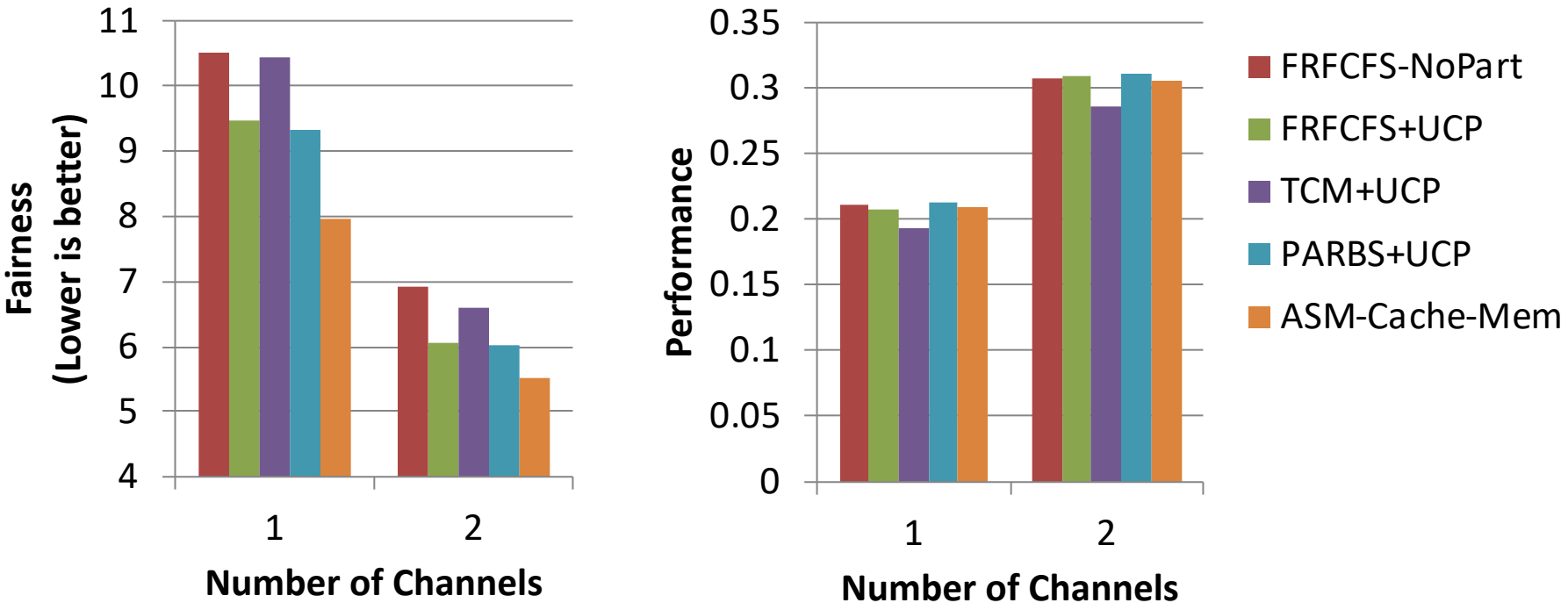


1. Employ ASM-Cache to partition cache capacity

2. Drive ASM-Mem with slowdowns from ASM-Cache

Fairness and Performance Results

16-core system
100 workloads



Significant fairness benefits across different channel counts

Summary

- Problem: Uncontrolled memory interference cause high and unpredictable application slowdowns
- Goal: Quantify and control slowdowns
- Key Contribution:
  - ASM: An accurate slowdown estimation model
  - Average error of ASM: 10%
- Key Ideas:
  - Shared cache access rate is a proxy for performance
  - Cache Access Rate $_{Alone}$ can be estimated by minimizing memory interference and quantifying cache interference
- Applications of Our Model
  - Slowdown-aware cache and memory management to achieve high performance, fairness and performance guarantees
- *Source Code Released in January 2016*

More on Application Slowdown Model

- Lavanya Subramanian, Vivek Seshadri, Arnab Ghosh, Samira Khan, and Onur Mutlu,
  "The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory"
  *Proceedings of the 48th International Symposium on Microarchitecture (**MICRO**)*, Waikiki, Hawaii, USA, December 2015.
  [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
  [Source Code]

The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory

Lavanya Subramanian*§ Vivek Seshadri* Arnab Ghosh*†
Samira Khan*‡ Onur Mutlu*

*Carnegie Mellon University §Intel Labs †IIT Kanpur ‡University of Virginia

Interconnect QoS/Performance Ideas

Application-Aware Prioritization in NoCs

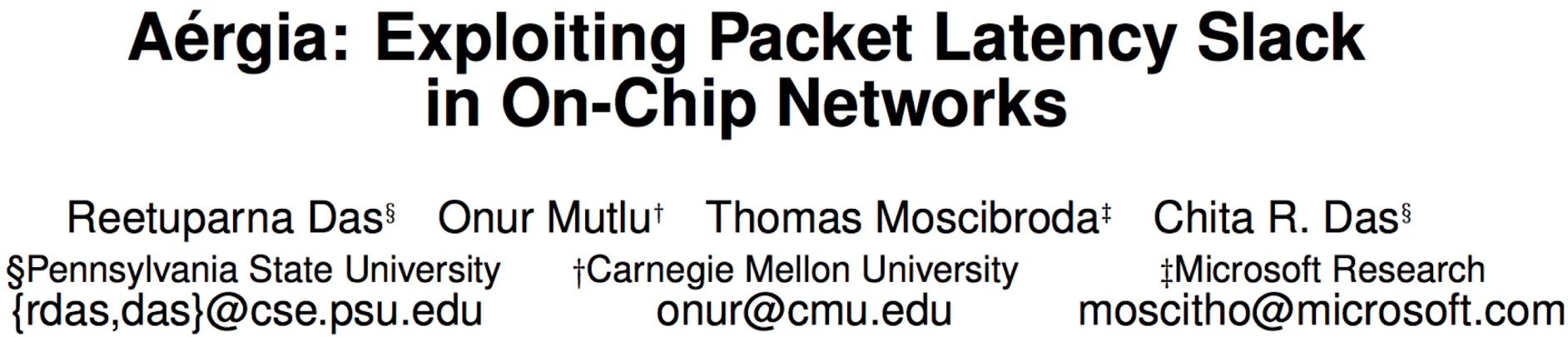
- Das et al., "Application-Aware Prioritization Mechanisms for On-Chip Networks," MICRO 2009.
  - https://users.ece.cmu.edu/~omutlu/pub/app-aware-noc_micro09.pdf

Application-Aware Prioritization Mechanisms for On-Chip Networks

Reetuparna Das[§] Onur Mutlu[†] Thomas Moscibroda[‡] Chita R. Das[§]
§Pennsylvania State University {rdas,das}@cse.psu.edu
†Carnegie Mellon University onur@cmu.edu
‡Microsoft Research moscitho@microsoft.com

Slack-Based Packet Scheduling

- Reetuparna Das, Onur Mutlu, Thomas Moscibroda, and Chita R. Das, **"Aergia: Exploiting Packet Latency Slack in On-Chip Networks"** *Proceedings of the 37th International Symposium on Computer Architecture (**ISCA**)*, pages 106-116, Saint-Malo, France, June 2010. Slides (pptx)



Aérgia: Exploiting Packet Latency Slack in On-Chip Networks

Reetuparna Das§ Onur Mutlu† Thomas Moscibroda‡ Chita R. Das§

§Pennsylvania State University {rdas,das}@cse.psu.edu

†Carnegie Mellon University onur@cmu.edu

‡Microsoft Research moscitho@microsoft.com

Low-Cost QoS in On-Chip Networks (I)

- Boris Grot, Stephen W. Keckler, and Onur Mutlu, **"Preemptive Virtual Clock: A Flexible, Efficient, and Cost-effective QOS Scheme for Networks-on-Chip"** *Proceedings of the 42nd International Symposium on Microarchitecture (**MICRO**)*, pages 268-279, New York, NY, December 2009. Slides (pdf)

Preemptive Virtual Clock: A Flexible, Efficient, and Cost-effective QOS Scheme for Networks-on-Chip

Boris Grot Stephen W. Keckler Onur Mutlu†

Department of Computer Sciences
The University of Texas at Austin
{bgrot, skeckler@cs.utexas.edu}

†Computer Architecture Laboratory (CALCM)
Carnegie Mellon University
onur@cmu.edu

Low-Cost QoS in On-Chip Networks (II)

- Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu, **"Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees"**
  *Proceedings of the 38th International Symposium on Computer Architecture (**ISCA**)*, San Jose, CA, June 2011. Slides (pptx)

Kilo-NOC: A Heterogeneous Network-on-Chip Architecture for Scalability and Service Guarantees

Boris Grot[1] bgrot@cs.utexas.edu
Joel Hestness[1] hestness@cs.utexas.edu
Stephen W. Keckler[1,2] skeckler@nvidia.com
Onur Mutlu[3] onur@cmu.edu

[1]The University of Texas at Austin
Austin, TX

[2]NVIDIA
Santa Clara, CA

[3]Carnegie Mellon University
Pittsburgh, PA

Throttling Based Fairness in NoCs

- Kevin Chang, Rachata Ausavarungnirun, Chris Fallin, and Onur Mutlu, **"HAT: Heterogeneous Adaptive Throttling for On-Chip Networks"**
  Proceedings of the 24th International Symposium on Computer Architecture and High Performance Computing (***SBAC-PAD***), New York, NY, October 2012. Slides (pptx) (pdf)

HAT: Heterogeneous Adaptive Throttling for On-Chip Networks

Kevin Kai-Wei Chang, Rachata Ausavarungnirun, Chris Fallin, Onur Mutlu
Carnegie Mellon University
{kevincha,rachata,cfallin,onur}@cmu.edu

Scalability: Express Cube Topologies

- Boris Grot, Joel Hestness, Stephen W. Keckler, and Onur Mutlu, **"Express Cube Topologies for On-Chip Interconnects"** *Proceedings of the 15th International Symposium on High-Performance Computer Architecture (**HPCA**)*, pages 163-174, Raleigh, NC, February 2009. Slides (ppt)

Express Cube Topologies for On-Chip Interconnects

Boris Grot Joel Hestness Stephen W. Keckler Onur Mutlu†

Department of Computer Sciences
The University of Texas at Austin
{bgrot, hestness, skeckler}@cs.utexas.edu

†Computer Architecture Laboratory (CALCM)
Carnegie Mellon University
onur@cmu.edu

Scalability: Slim NoC

- Maciej Besta, Syed Minhaj Hassan, Sudhakar Yalamanchili, Rachata Ausavarungnirun, Onur Mutlu, Torsten Hoefler, **"Slim NoC: A Low-Diameter On-Chip Network Topology for High Energy Efficiency and Scalability"** *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Williamsburg, VA, USA, March 2018. [Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pdf)]

Slim NoC: A Low-Diameter On-Chip Network Topology for High Energy Efficiency and Scalability

Maciej Besta[1] Syed Minhaj Hassan[2] Sudhakar Yalamanchili[2]
Rachata Ausavarungnirun[3] Onur Mutlu[1,3] Torsten Hoefler[1]

[1]ETH Zürich [2]Georgia Institute of Technology [3]Carnegie Mellon University

Bufferless Routing in NoCs

- Moscibroda and Mutlu, "A Case for Bufferless Routing in On-Chip Networks," ISCA 2009.
  - https://users.ece.cmu.edu/~omutlu/pub/bless_isca09.pdf

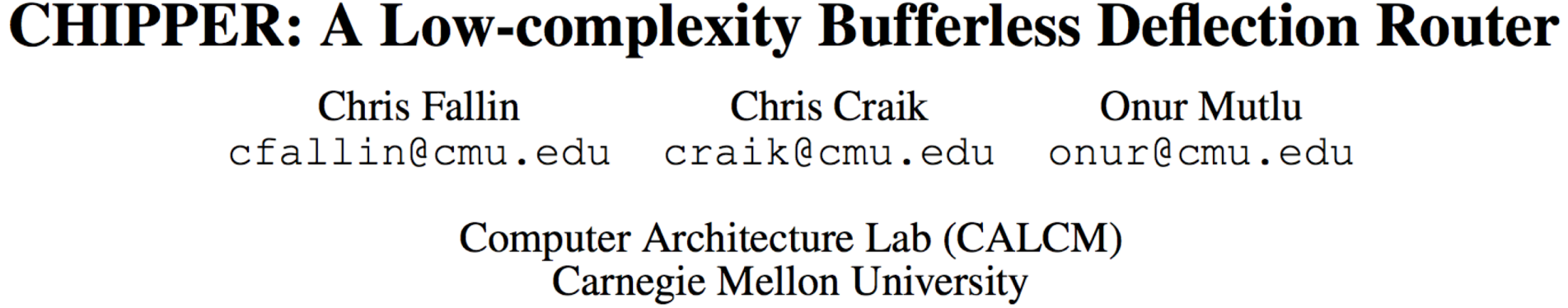
A Case for Bufferless Routing in On-Chip Networks

Thomas Moscibroda
Microsoft Research
moscitho@microsoft.com

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu

CHIPPER: Low-Complexity Bufferless

- Chris Fallin, Chris Craik, and Onur Mutlu,
  "CHIPPER: A Low-Complexity Bufferless Deflection Router"
  *Proceedings of the 17th International Symposium on High-Performance Computer Architecture (**HPCA**)*, pages 144-155, San Antonio, TX, February 2011. Slides (pptx)
  An extended version as *SAFARI Technical Report*, TR-SAFARI-2010-001, Carnegie Mellon University, December 2010.



CHIPPER: A Low-complexity Bufferless Deflection Router

Chris Fallin cfallin@cmu.edu
Chris Craik craik@cmu.edu
Onur Mutlu onur@cmu.edu

Computer Architecture Lab (CALCM)
Carnegie Mellon University

Minimally-Buffered Deflection Routing

- Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
  "MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect"
  *Proceedings of the 6th ACM/IEEE International Symposium on Networks on Chip (**NOCS**)*, Lyngby, Denmark, May 2012. Slides (pptx) (pdf)

MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect

Chris Fallin, Greg Nazario, Xiangyao Yu†, Kevin Chang, Rachata Ausavarungnirun, Onur Mutlu

Carnegie Mellon University
{cfallin,gnazario,kevincha,rachata,onur}@cmu.edu

†Tsinghua University & Carnegie Mellon University
yxythu@gmail.com

"Bufferless" Hierarchical Rings

- Ausavarungnirun et al., "Design and Evaluation of Hierarchical Rings with Deflection Routing," SBAC-PAD 2014.
  - http://users.ece.cmu.edu/~omutlu/pub/hierarchical-rings-with-deflection_sbacpad14.pdf

- Discusses the design and implementation of a mostly-bufferless hierarchical ring

Design and Evaluation of Hierarchical Rings with Deflection Routing

Rachata Ausavarungnirun Chris Fallin Xiangyao Yu† Kevin Kai-Wei Chang
Greg Nazario Reetuparna Das§ Gabriel H. Loh‡ Onur Mutlu

Carnegie Mellon University §University of Michigan †MIT ‡Advanced Micro Devices, Inc.

"Bufferless" Hierarchical Rings (II)

- Rachata Ausavarungnirun, Chris Fallin, Xiangyao Yu, Kevin Chang, Greg Nazario, Reetuparna Das, Gabriel Loh, and Onur Mutlu, **"A Case for Hierarchical Rings with Deflection Routing: An Energy-Efficient On-Chip Communication Substrate"** ***Parallel Computing*** ***(PARCO)***, to appear in 2016.
  - arXiv.org version, February 2016.

Achieving both High Energy Efficiency
and High Performance in On-Chip Communication
using Hierarchical Rings with Deflection Routing

Rachata Ausavarungnirun Chris Fallin Xiangyao Yu† Kevin Kai-Wei Chang
Greg Nazario Reetuparna Das§ Gabriel H. Loh‡ Onur Mutlu

Carnegie Mellon University §University of Michigan †MIT ‡AMD

Summary of Six Years of Research

- Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, and Onur Mutlu,
  "Bufferless and Minimally-Buffered Deflection Routing"
  Invited Book Chapter in Routing Algorithms in Networks-on-Chip, pp. 241-275, Springer, 2014.

Chapter 1

Bufferless and Minimally-Buffered Deflection Routing

Chris Fallin, Greg Nazario, Xiangyao Yu, Kevin Chang, Rachata Ausavarungnirun, Onur Mutlu

On-Chip vs. Off-Chip Tradeoffs

- George Nychis, Chris Fallin, Thomas Moscibroda, Onur Mutlu, and Srinivasan Seshan,
  "On-Chip Networks from a Networking Perspective: Congestion and Scalability in Many-core Interconnects"
  *Proceedings of the 2012 ACM SIGCOMM Conference (**SIGCOMM**)*, Helsinki, Finland, August 2012. Slides (pptx)

On-Chip Networks from a Networking Perspective: Congestion and Scalability in Many-Core Interconnects

George Nychis†, Chris Fallin†, Thomas Moscibroda§, Onur Mutlu†, Srinivasan Seshan†

† Carnegie Mellon University
{gnychis,cfallin,onur,srini}@cmu.edu

§ Microsoft Research Asia
moscitho@microsoft.com

Slowdown Estimation in NoCs

- Xiyue Xiang, Saugata Ghose, Onur Mutlu, and Nian-Feng Tzeng, **"A Model for Application Slowdown Estimation in On-Chip Networks and Its Use for Improving System Fairness and Performance"**
  *Proceedings of the 34th IEEE International Conference on Computer Design (**ICCD**)*, Phoenix, AZ, USA, October 2016.
  [Slides (pptx) (pdf)]

A Model for Application Slowdown Estimation in On-Chip Networks and Its Use for Improving System Fairness and Performance

Xiyue Xiang[†] Saugata Ghose[‡] Onur Mutlu[§‡] Nian-Feng Tzeng[†]
[†]*University of Louisiana at Lafayette* [‡]*Carnegie Mellon University* [§]*ETH Zürich*

Handling Multicast and Hotspot Issues

- Xiyue Xiang, Wentao Shi, Saugata Ghose, Lu Peng, Onur Mutlu, and Nian-Feng Tzeng,
  "Carpool: A Bufferless On-Chip Network Supporting Adaptive Multicast and Hotspot Alleviation"
  *Proceedings of the International Conference on Supercomputing (**ICS**)*, Chicago, IL, USA, June 2017.
  [Slides (pptx) (pdf)]