

Managing GPU Concurrency in Heterogeneous Architectures

Onur Kayiran, Nachiappan CN, Adwait Jog, Rachata Ausavarungnirun, Mahmut T. Kandemir, Gabriel H. Loh, Onur Mutlu, Chita R. Das

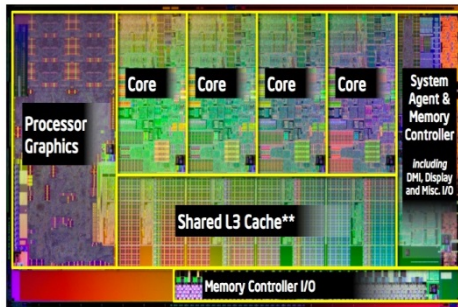


Carnegie Mellon



Era of Heterogeneous Architectures

Intel Haswell



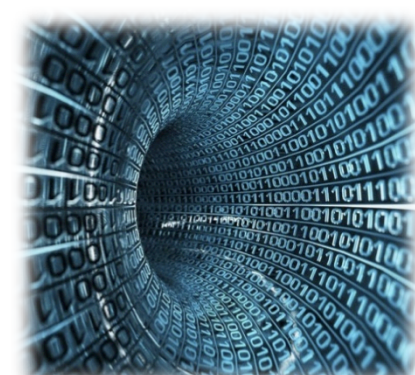
AMD Fusion



NVIDIA Denver



NVIDIA Echelon



Executive Summary

- When sharing the memory hierarchy, CPU and GPU applications interfere with each other
 - GPU applications significantly affect CPU applications due to multi-threading
- Existing GPU Thread-level Parallelism (TLP) management techniques (MICRO12, PACT13)
 - Unaware of CPUs
 - Not effective in heterogeneous systems

Our Proposal:

Warp scheduling strategies to
Adjust GPU TLP to improve CPU and/or GPU
performance

Executive Summary

CPU-centric Strategy

Memory Congestion 

CPU Performance 

Executive Summary

CPU-centric Strategy

Memory Congestion 

CPU Performance 

IF Memory Congestion 

 GPU TLP

Executive Summary

CPU-centric Strategy

Memory Congestion 

CPU Performance 

IF Memory Congestion 

 GPU TLP

Results Summary:





+24% CPU & -11% GPU

Executive Summary

CPU-centric Strategy

Memory Congestion 
CPU Performance 

CPU-GPU Balanced Strategy

GPU TLP   
GPU Latency Tolerance 

IF Memory Congestion 
 GPU TLP

Results Summary:

+24% CPU & -11% GPU

Executive Summary

CPU-centric Strategy

CPU-GPU Balanced Strategy

Memory Congestion 

GPU TLP   

CPU Performance 

GPU Latency Tolerance 

IF Memory Congestion 

 GPU TLP

IF Latency Tolerance 

 GPU TLP

Results Summary:





+24% CPU & -11% GPU

Executive Summary

CPU-centric Strategy

Memory Congestion 
CPU Performance 

CPU-GPU Balanced Strategy

GPU TLP 
GPU Latency Tolerance 

IF Memory Congestion 
 GPU TLP

IF Latency Tolerance 
 GPU TLP

Results Summary:

+24% CPU & -11% GPU

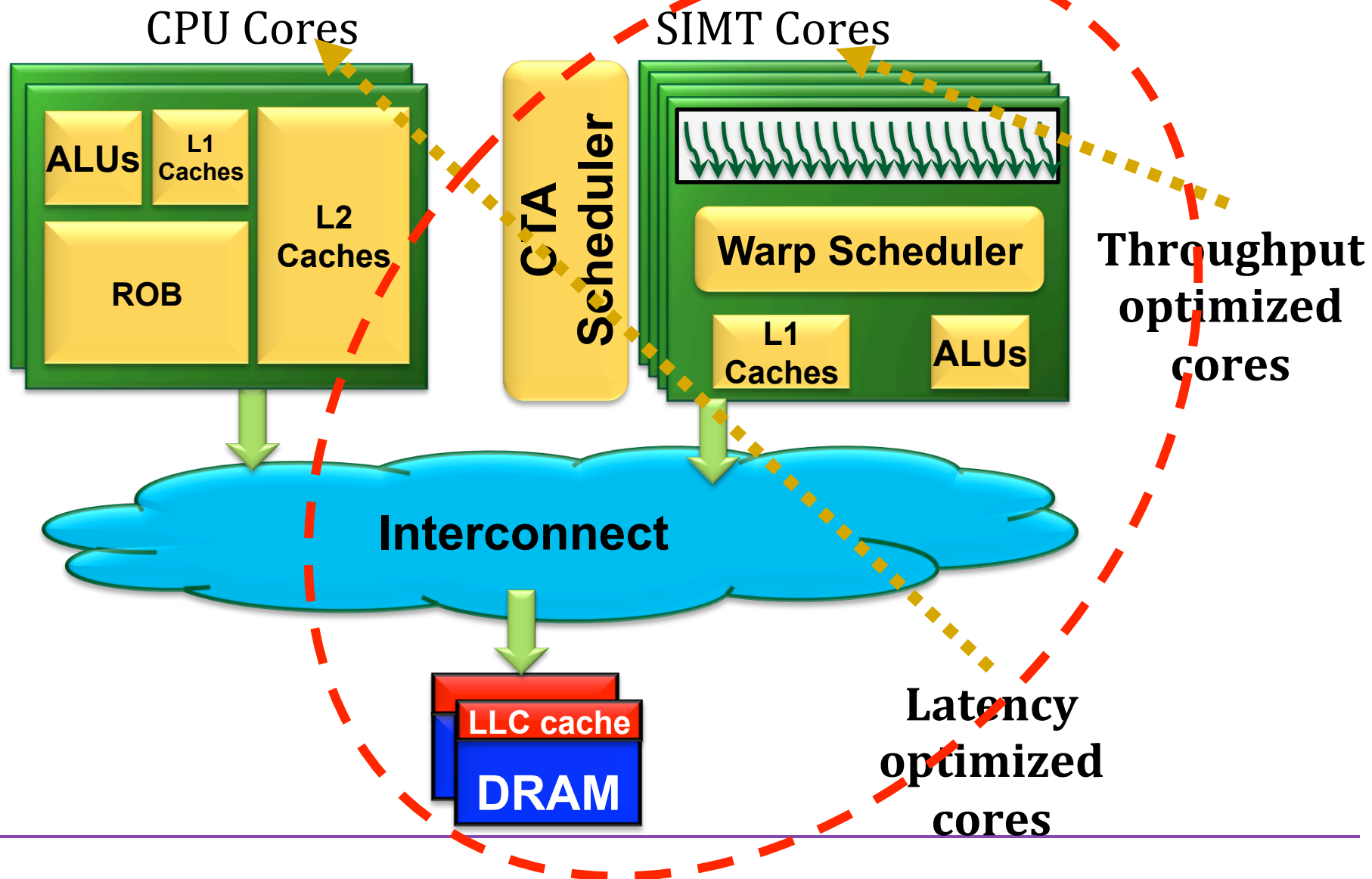
Results Summary:

+7% both CPU & GPU

Outline

- Summary
 - Background
 - Motivation
 - Analysis of TLP
 - Our Proposal
 - Evaluation
 - Conclusions
-

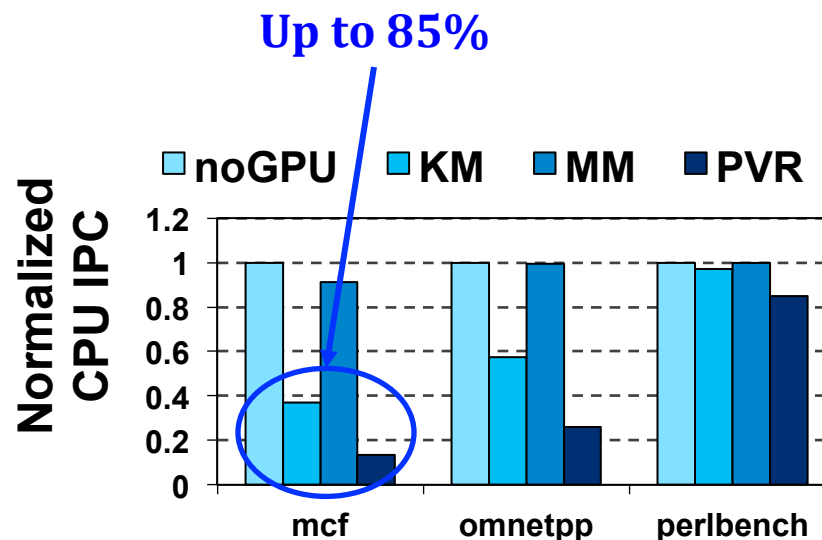
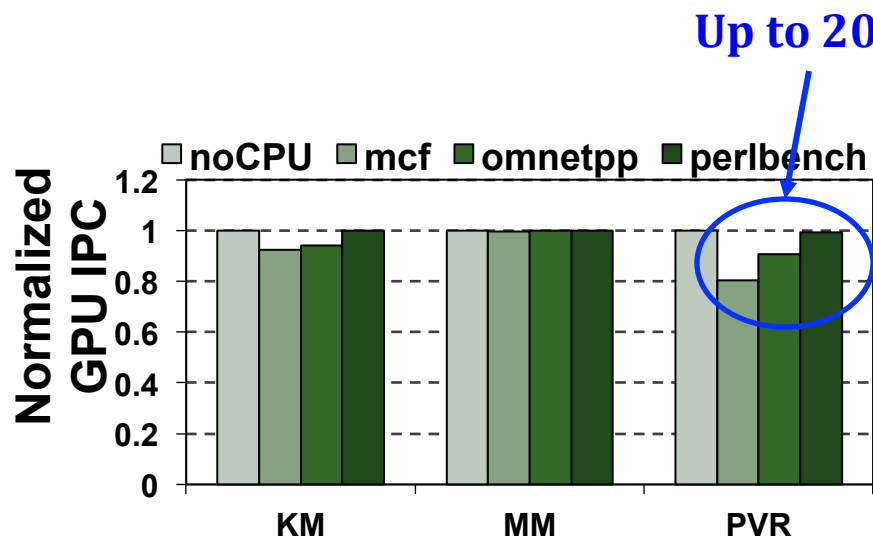
Many-core Architecture



Outline

- Summary
 - Background
 - Motivation
 - Analysis of TLP
 - Our Proposal
 - Evaluation
 - Conclusions
-

Application Interference

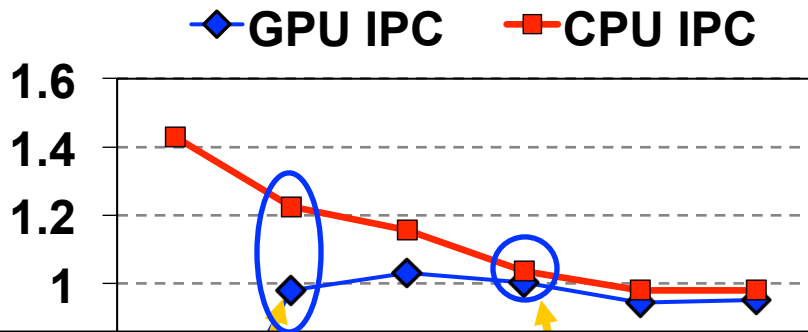


- GPU applications are affected moderately due to CPU interference

- CPU applications are affected significantly due to GPU interference

Latency Tolerance in CPUs vs. GPUs

Normalized IPC



■ High GPU TLP -> memory system congestion

Problem:

TLP management strategies for GPUs are not aware of the latency tolerance disparity between CPU and GPU applications

Higher performance potential at low TLP

DYNCTA
(PACT 2013)

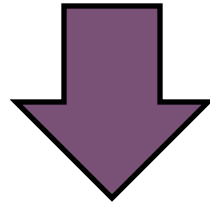
Outline

- Summary
 - Background
 - Motivation
 - Analysis of TLP
 - Our Proposal
 - Evaluation
 - Conclusions
-

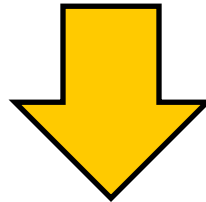
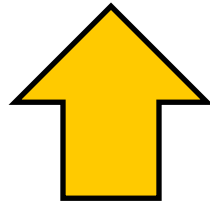
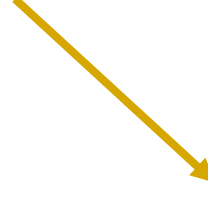
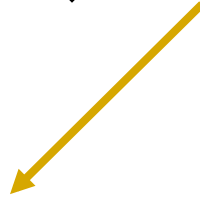
Effect of GPU Concurrency on GPU Performance



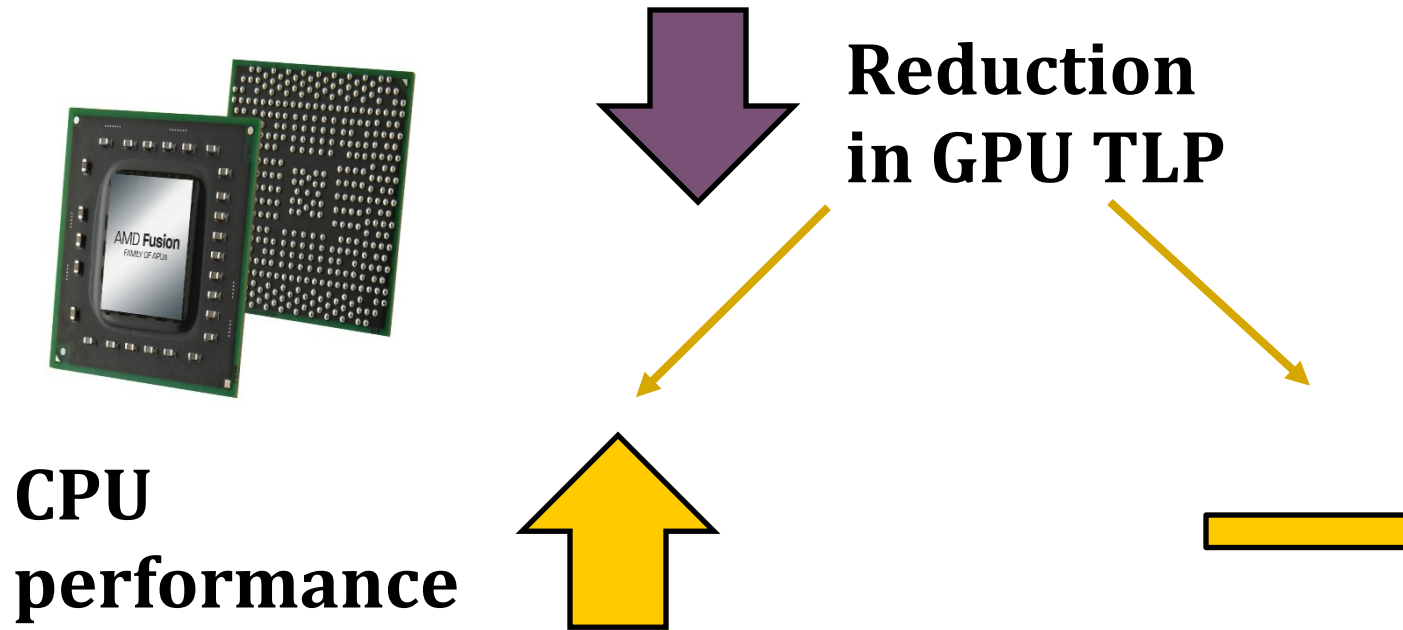
**GPU
performance**



**Reduction
in GPU TLP**



Effect of GPU Concurrency on CPU Performance



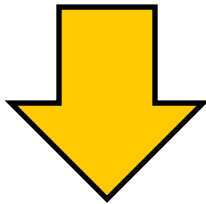
Effect of GPU Concurrency on CPU Performance



Change in
CPU
performance

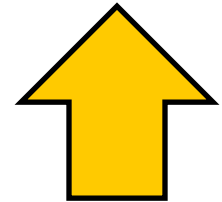
- 2 metrics:
- Memory congestion
 - Network congestion

congestion



:

CPU
performance



Outline

- Summary
 - Background
 - Motivation
 - Analysis of TLP
 - **Our Proposal**
 - Evaluation
 - Conclusions
-

Our Approach

	Improved GPU performance	Improved CPU performance
Existing works	☑	✗
CPU-centric Strategy	✗	☑
CPU-GPU Balanced Strategy	☑	☑

+ control the trade-off

CM-CPU: CPU-centric Strategy

- Categorize congestion: **low**, **medium**, or **high**

memory network I M H

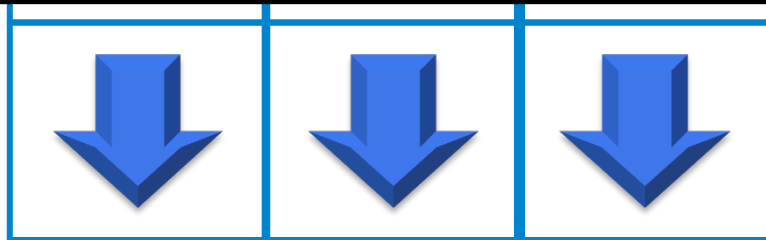
GPU-unaware TLP management:
Insufficient GPU latency tolerance


Increase
of
warps


No
change
in # of
warps


Decrease
of
warps

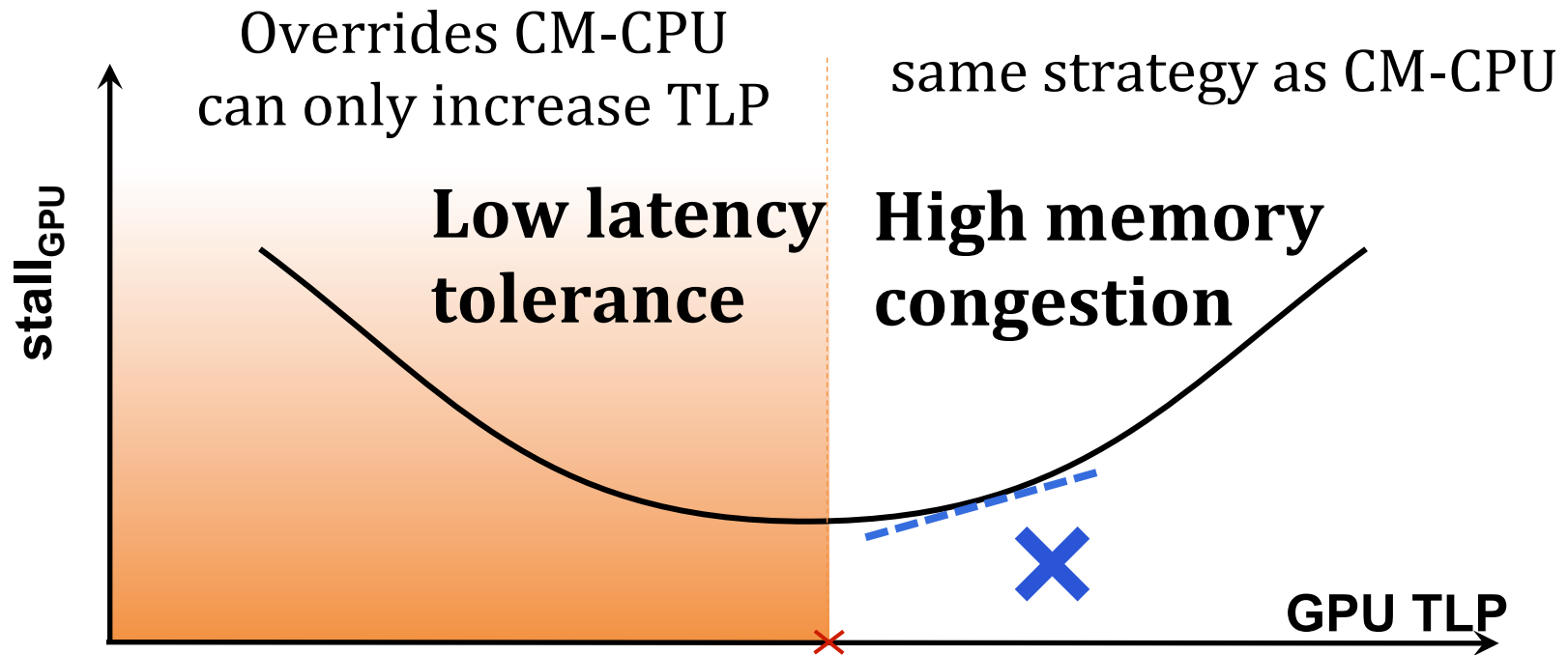
H



CM-BAL: CPU-GPU Balanced Strategy

Latency tolerance of GPU cores:

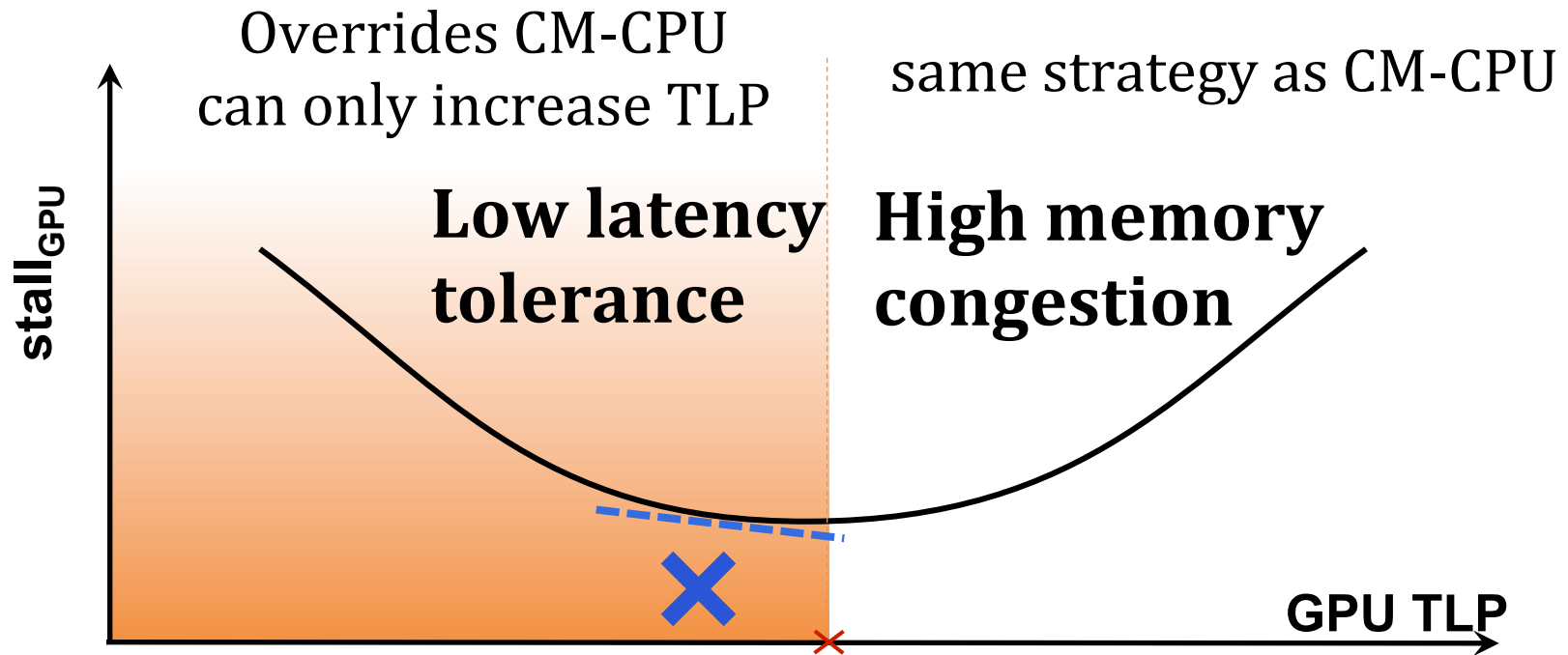
$\text{stall}_{\text{GPU}}$: scheduler stalls @ GPU cores



CM-BAL: CPU-GPU Balanced Strategy

Latency tolerance of GPU cores:

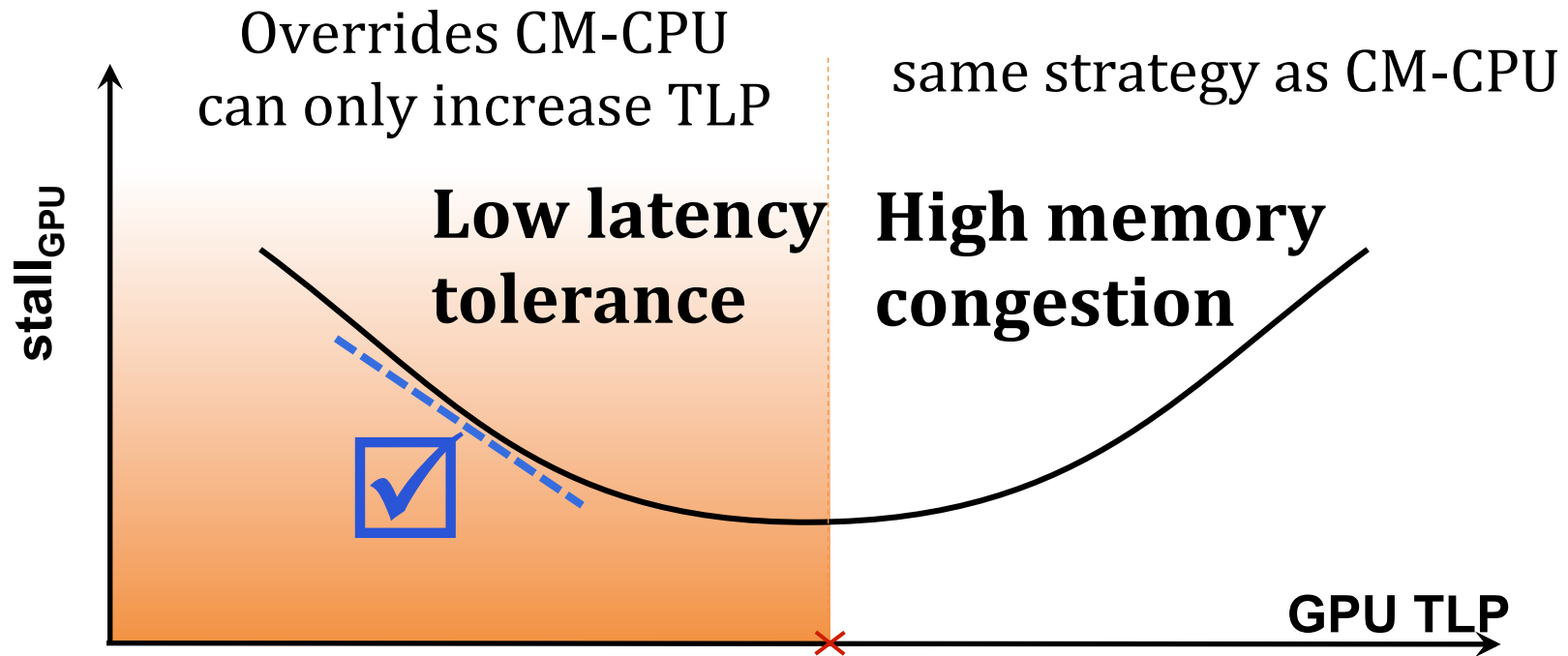
$\text{stall}_{\text{GPU}}$: scheduler stalls @ GPU cores



CM-BAL: CPU-GPU Balanced Strategy

Latency tolerance of GPU cores:

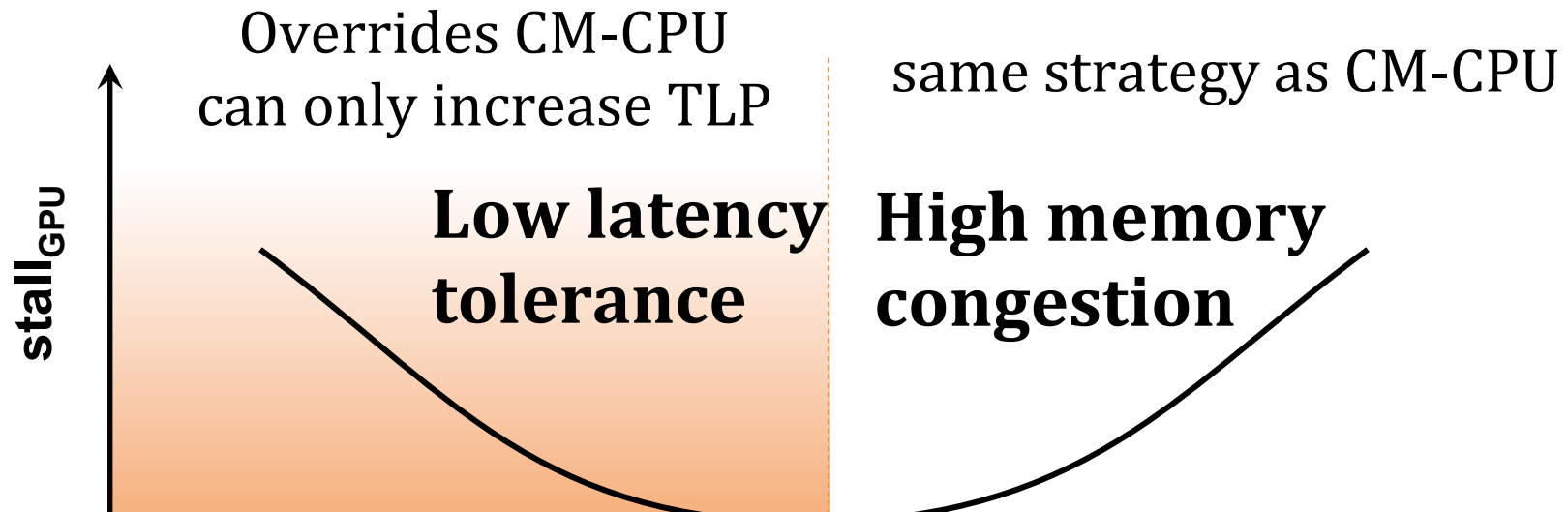
$\text{stall}_{\text{GPU}}$: scheduler stalls @ GPU cores



CM-BAL: CPU-GPU Balanced Strategy

Latency tolerance of GPU cores:

$\text{stall}_{\text{GPU}}$: scheduler stalls @ GPU cores



Control the triggering of the condition

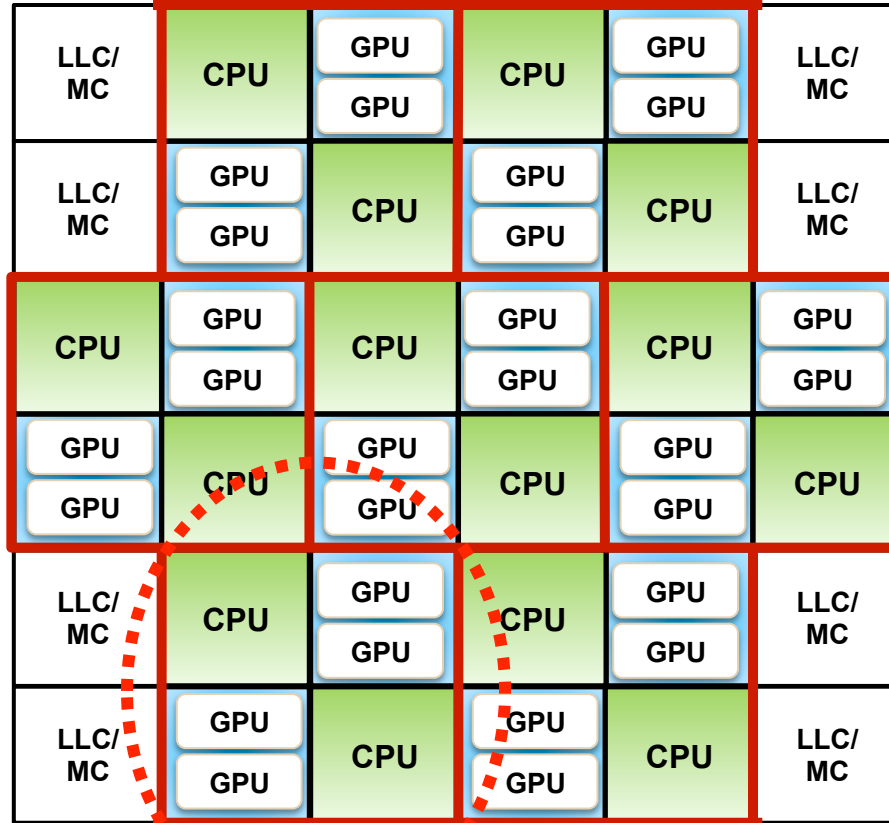
=

Control the trade-off between CPU or GPU benefits

Outline

- Summary
 - Background
 - Motivation
 - Analysis of TLP
 - Our Proposal
 - Evaluation
 - Conclusions
-

Evaluated Architecture

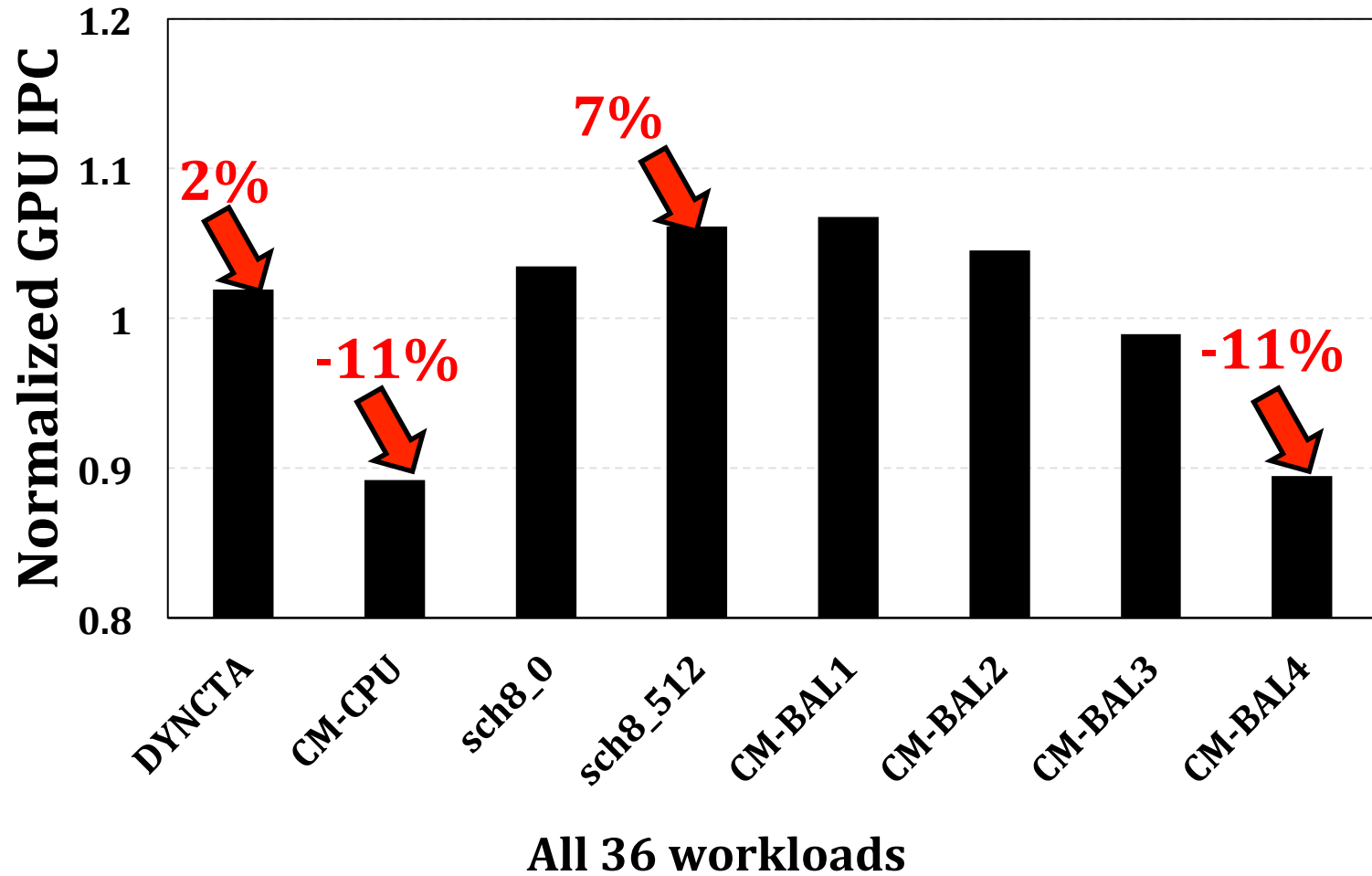


Tile-based
design

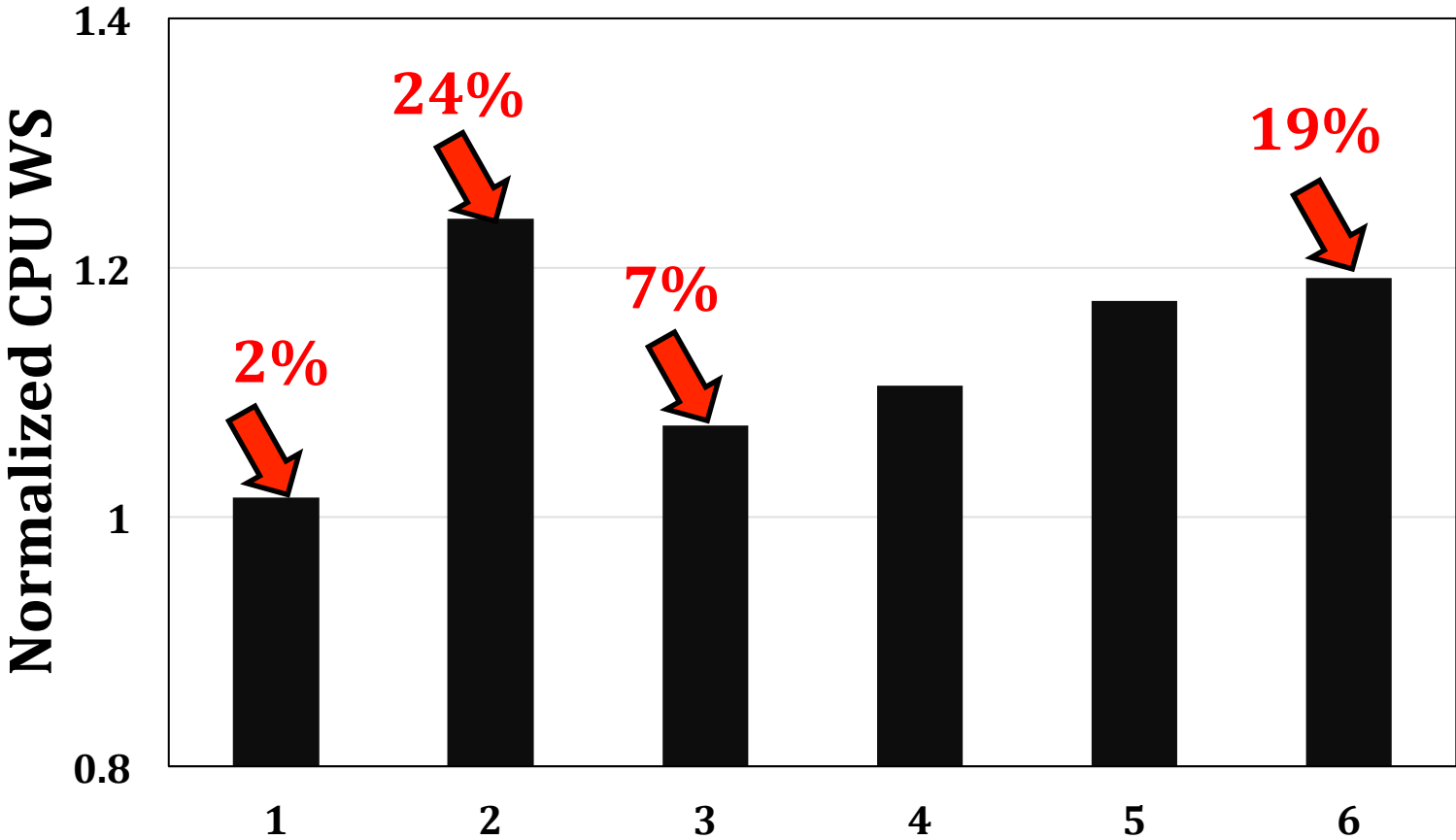
Evaluation Methodology

- Evaluated on an integrated platform with an in-house x86 CPU simulator and GPGPU-Sim
 - Baseline Architecture
 - 28 GPU cores, 14 CPU cores, 8 memory controllers, 2D mesh
 - GPU: 1400MHz, SIMT Width = 16×2 , Max. 1536 threads/core, GTO Sch.
 - CPU: 2000 MHz, OoO, 128-entry instr. win., max. 3 inst./cycle
 - 8MB, 128B Line, 16-way, 700MHz
 - GDDR5 800MHz
 - Workloads:
 - 13 GPU applications
 - 34 CPU applications, 6 CPU application mixes
 - 36 diverse workloads
 - 1 GPU application + 1 CPU mix
-

GPU Performance Results



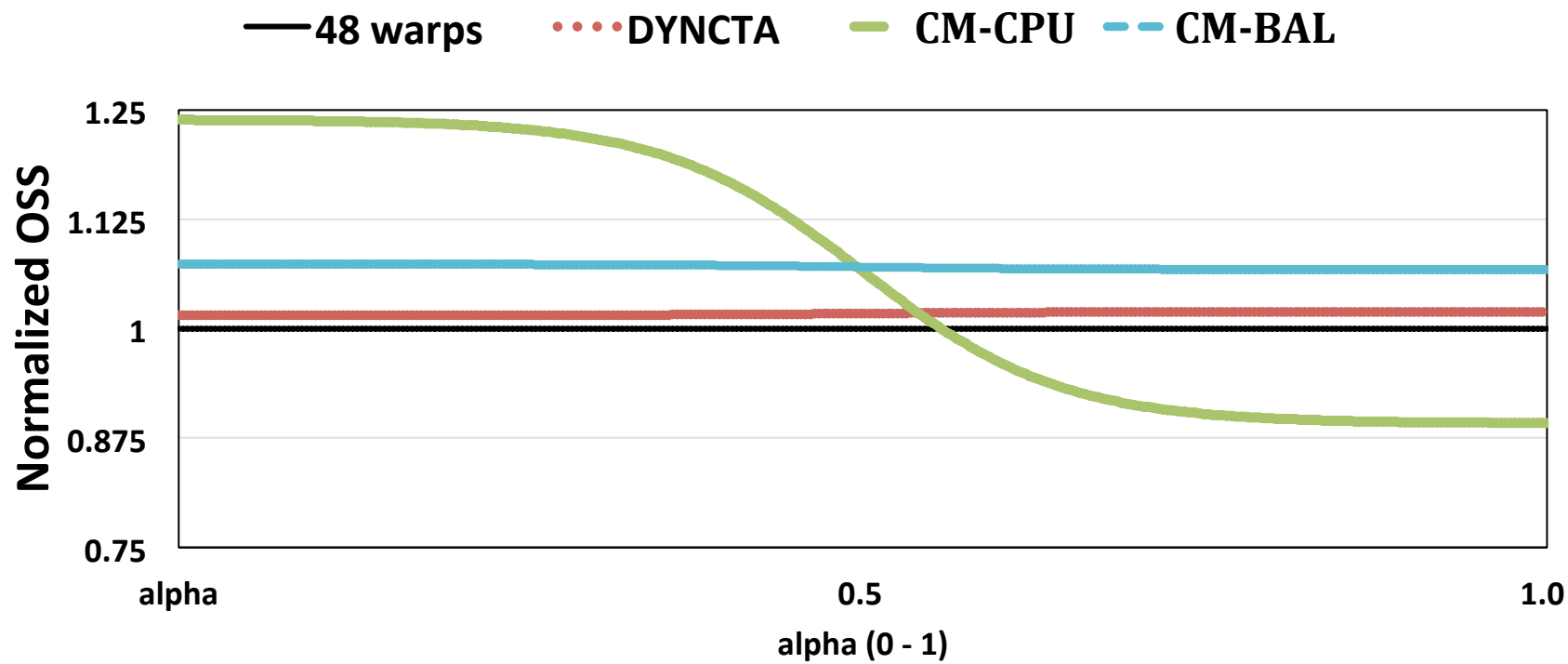
CPU Performance Results



All 36 workloads

System Performance

- $OSS = (1 - \alpha) \times WS_{CPU} + \alpha \times SU_{GPU}$ (ISCA 2012)
- α is between 0 and 1
- Higher α -> higher GPU importance



More in the Paper

■ Motivation

- Analysis of the metrics used by our algorithm

■ Scheme

- Detailed hardware walkthrough of our scheme

■ Results

- Analysis over time
 - Change in GPU TLP
 - Change in the metrics used by our algorithm
 - Comparison against static approaches
 - Lower number of LLC accesses
-

Outline

- Summary
 - Background
 - Motivation
 - Analysis of TLP
 - Our Proposal
 - Evaluation
 - **Conclusions**
-

Conclusions

- Sharing the memory hierarchy leads to CPU and GPU applications to interfere with each other
 - Existing GPU TLP management techniques are not well-suited for heterogeneous architectures
 - We propose two GPU TLP management techniques for heterogeneous architectures
 - CM-CPU reduces GPU TLP to improve CPU performance
 - CM-BAL is similar to CM-CPU, but increases GPU TLP when it detects low latency tolerance in GPU cores
 - TLP can be tuned based on user's preference for higher CPU or GPU performance
-

THANKS!

Managing GPU Concurrency in Heterogeneous Architectures

Onur Kayiran, Nachiappan CN, Adwait Jog, Rachata Ausavarungnirun, Mahmut T. Kandemir, Gabriel H. Loh, Onur Mutlu, Chita R. Das



Carnegie Mellon

