HAT: Heterogeneous Adaptive Throttling for On-Chip Networks

Kevin Kai-Wei Chang,

Rachata Ausavarungnirun,

Chris Fallin,

Onur Mutlu

Carnegie Mellon University SAFARI

Executive Summary

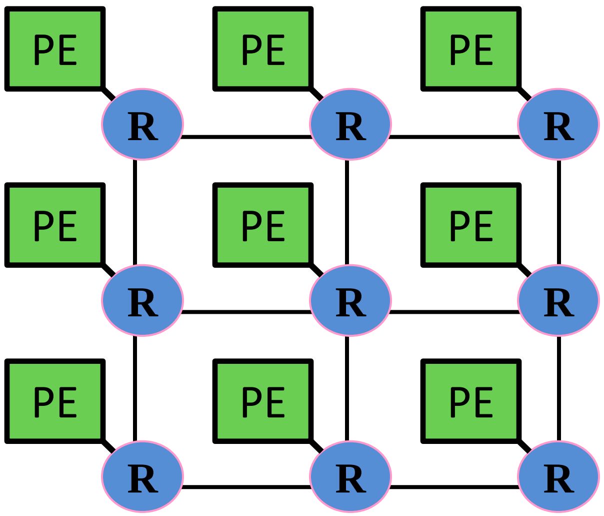
- **Problem:** Packets contend in on-chip networks (NoCs), causing congestion, thus reducing performance
- **Observations:**

  1) Some applications are more sensitive to network latency than others
  2) Applications must be throttled differently to achieve peak performance
- **Key Idea: Heterogeneous Adaptive Throttling (HAT)**
  1) Application-aware source throttling
  2) Network-load-aware throttling rate adjustment
- **Result:** Improves performance and energy efficiency over state-of-the-art source throttling policies

Outline

- **Background and Motivation**
- Mechanism
- Prior Works
- Results

On-Chip Networks

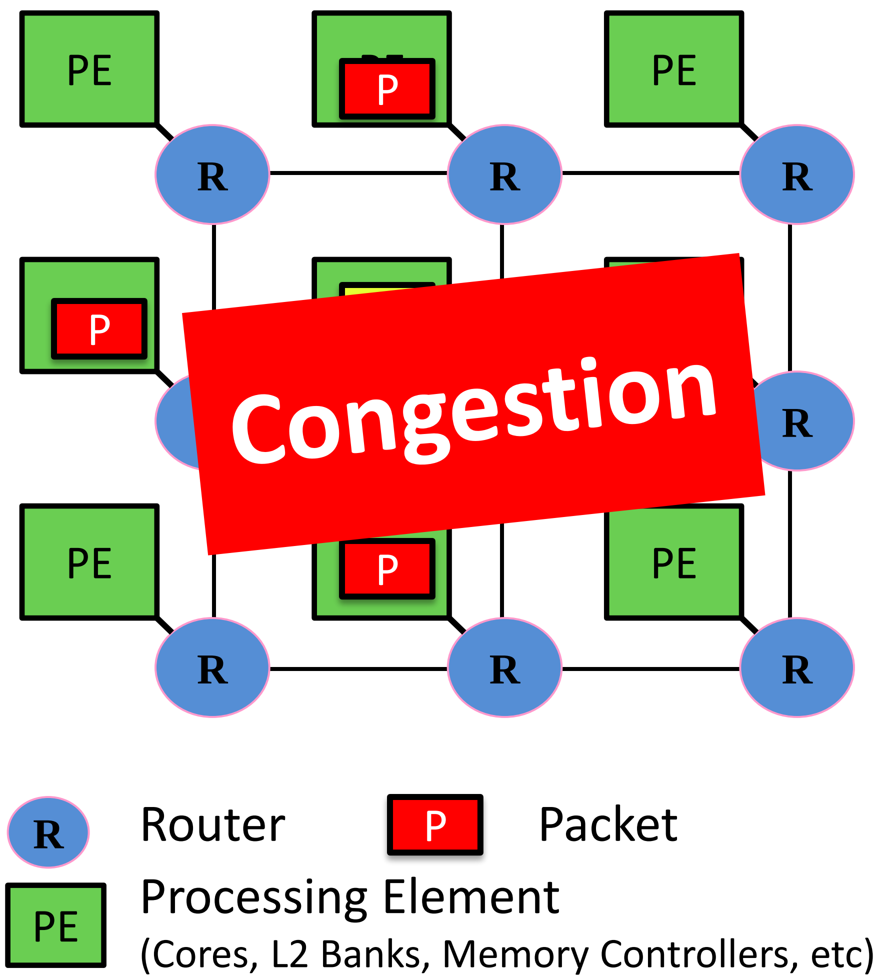


R Router

PE Processing Element
(Cores, L2 Banks, Memory Controllers, etc)

- Connect **cores, caches, memory controllers, etc**
- **Packet switched**
- **2D mesh:** Most commonly used topology
- Primarily serve **cache misses** and **memory requests**
- **Router designs**
  - Buffered: **Input buffers** to hold contending packets
  - Bufferless: **Misroute (deflect)** contending packets

Network Congestion Reduces Performance



Limited shared resources (buffers and links)

- **Design constraints: power, chip area**, and **timing**

Network congestion:

↓Network throughput

↓Application performance

Goal

- **Improve performance in a highly congested NoC**

- Reducing network load decreases network congestion, hence improves performance

- **<u>Approach:</u> source throttling to reduce network load**
  - Temporarily delay new traffic injection

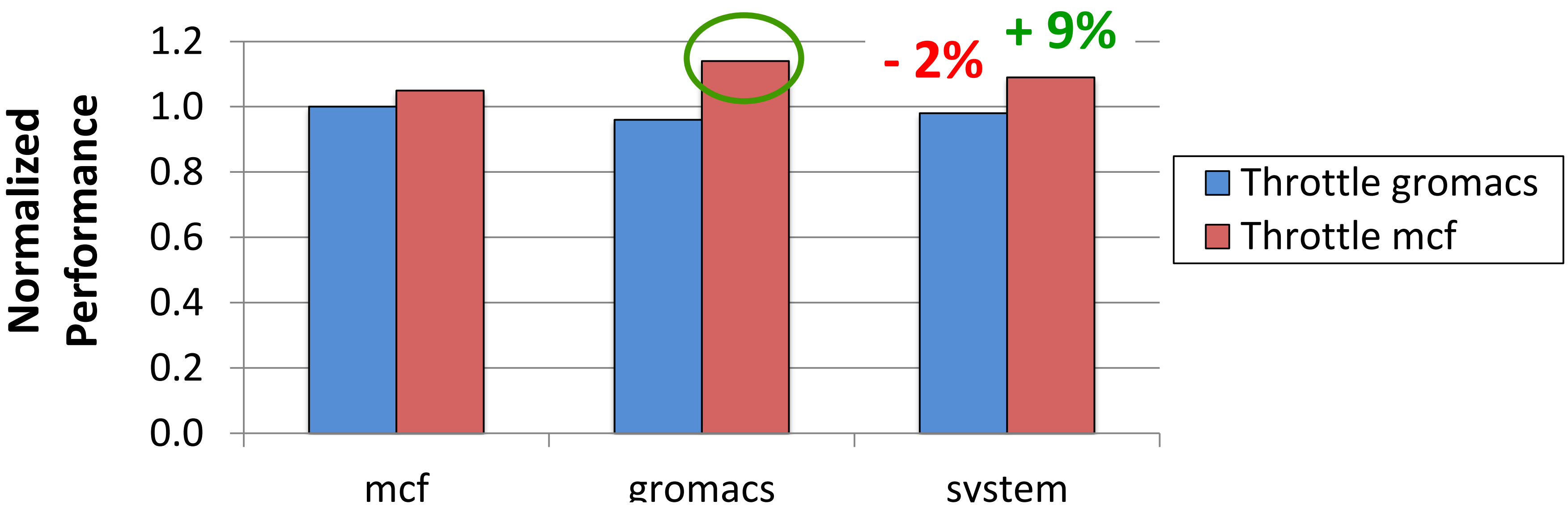
- **<u>Naïve mechanism</u>: throttle every single node**

Key Observation #1

Different applications respond differently to changes in **network latency**

gromacs: network-**non**-intensive

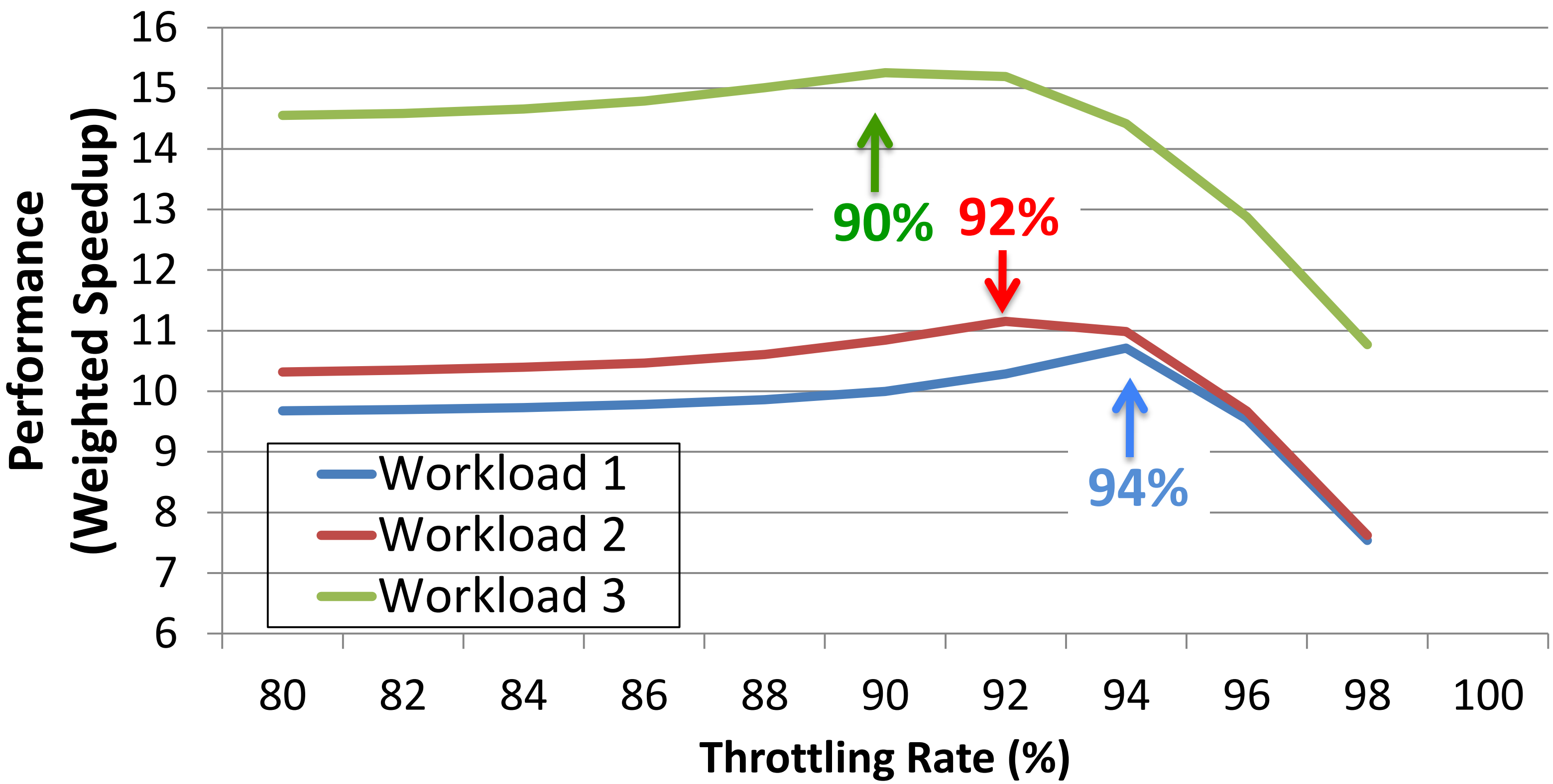
mcf: network-intensive



Throttling **network-intensive** applications benefits system performance more

Key Observation #2

Different workloads achieve peak performance at different throttling rates



Dynamically adjusting throttling rate yields better performance than a single static rate

Outline

- Background and Motivation
- **Mechanism**
- Prior Works
- Results

Heterogeneous Adaptive Throttling (HAT)

1. **Application-aware throttling:**
   Throttle **network-intensive** applications that interfere with **network-non-intensive** applications

2. **Network-load-aware throttling rate adjustment:**
   Dynamically adjusts throttling rate to adapt to different workloads

Heterogeneous Adaptive Throttling (HAT)

1. **Application-aware throttling:**
   Throttle **network-intensive** applications that interfere with **network-non-intensive** applications

2. **Network-load-aware throttling rate adjustment:**
   Dynamically adjusts throttling rate to adapt to different workloads

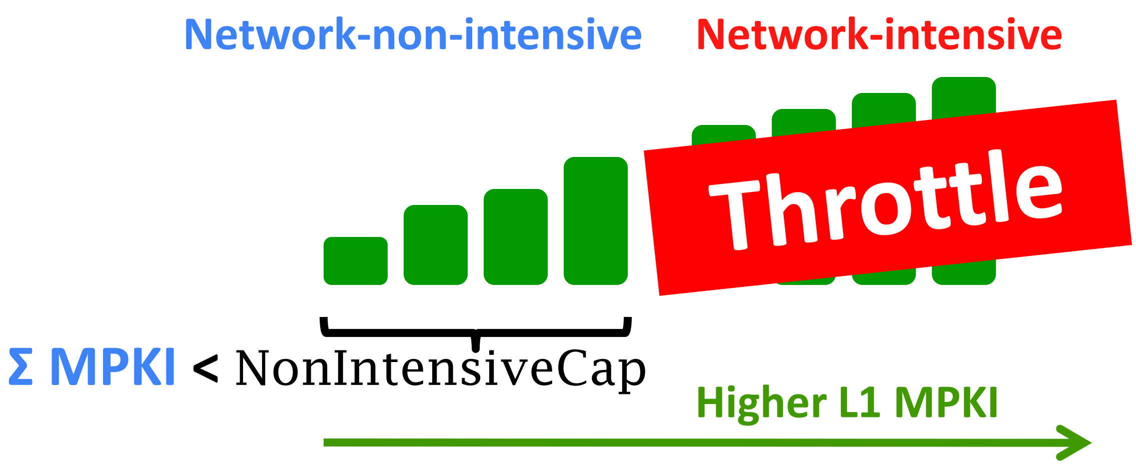
Application-Aware Throttling

1. **<u>Measure Network Intensity</u>**

   Use **L1 MPKI** (misses per thousand instructions) to estimate network intensity

2. **<u>Classify Application</u>**

   Sort applications by L1 MPKI



3. **<u>Throttle network-intensive applications</u>**

Heterogeneous Adaptive Throttling (HAT)

1. **Application-aware throttling**:
   Throttle **network-intensive** applications that interfere with **network-non-intensive** applications

2. **Network-load-aware throttling rate adjustment**:
   Dynamically adjusts throttling rate to adapt to different workloads

Dynamic Throttling Rate Adjustment

- For a given **network design**, peak performance tends to occur at a **fixed network load point**

- **Dynamically** adjust throttling rate to achieve that network load point

Dynamic Throttling Rate Adjustment

- **Goal:** maintain network load at a peak performance point

1. **<u>Measure network load</u>**
2. **<u>Compare and adjust throttling rate</u>**

If **network load** > **peak point**:

Increase throttling rate

elif **network load** ≤ **peak point**:

Decrease throttling rate

Epoch-Based Operation

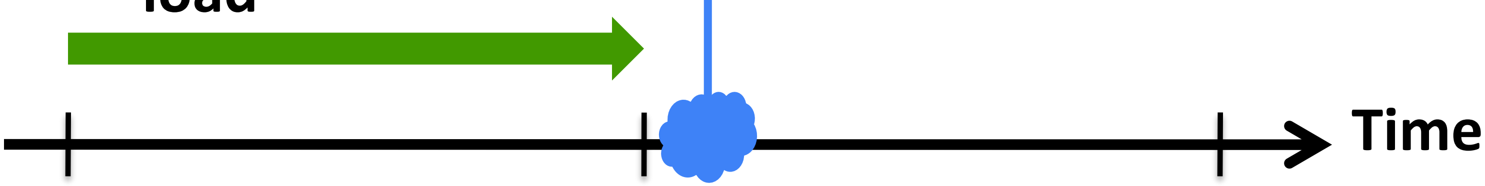
- Continuous **HAT** operation is expensive
- **Solution:** performs **HAT** at epoch granularity

During epoch:

1) Measure **L1 MPKI** of each application
2) Measure **network load**

Beginning of epoch:

1) Classify applications
2) Adjust throttling rate
3) Reset measurements



Current Epoch (100K cycles)

Next Epoch (100K cycles)

Outline

- Background and Motivation
- Mechanism
- **Prior Works**
- Results

Prior Source Throttling Works

- **Source throttling for bufferless NoCs**
  [Nychis+ Hotnets'10, SIGCOMM'12]
  – Application-aware throttling based on starvation rate
  – Does not adaptively adjust throttling rate
  – "Heterogeneous Throttling"
- **Source throttlinr off-chip buffered networks**
  [Thottethodi+ HPCA'01]
  – Dynamically trigger throttling based on fraction of buffer occupancy
  – Not application-aware: fully block packet injections of every node
  – "Self-tuned Throttling"

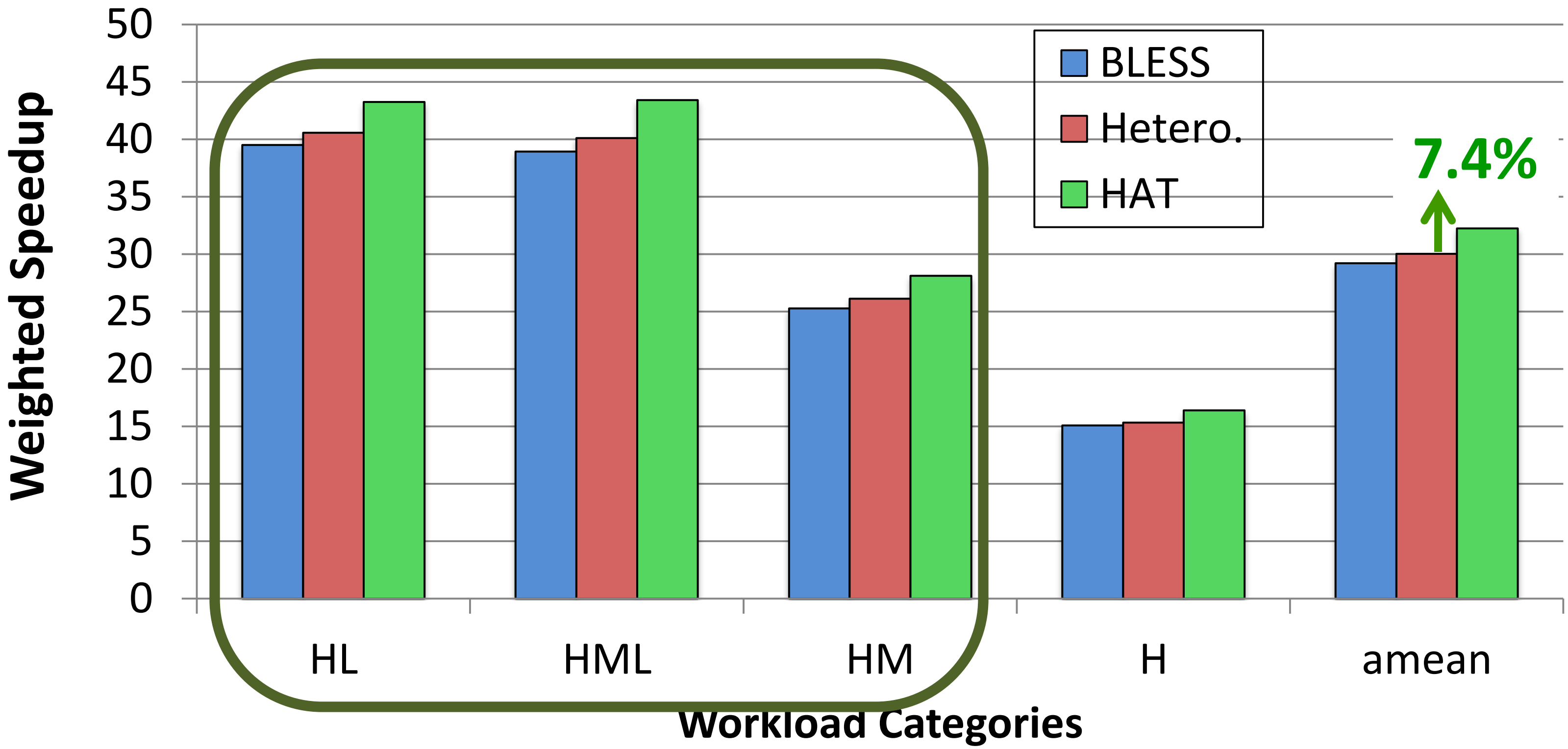
Outline

- Background and Motivation
- Mechanism
- Prior Works
- **Results**

Methodology

- **Chip Multiprocessor Simulator**
  - **64-node** multi-core systems with a **2D-mesh topology**
  - Closed-loop core/cache/NoC cycle-level model
  - 64KB L1, perfect L2 (always hits to stress NoC)
- **Router Designs**
  - **Virtual-channel buffered** router**:** 4 VCs, 4 flits/VC [Dally+ IEEE TPDS'92]
  - **Bufferless deflection** routers**: BLESS** [Moscibroda+ ISCA'09]
- **Workloads**
  - 60 multi-core workloads: SPEC CPU2006 benchmarks
  - Categorized based on their network intensity
    - **L**ow/**M**edium/**H**igh intensity categories
- **Metrics:** Weighted Speedup (perf.), perf./Watt (energy eff.), and maximum slowdown (fairness)

Performance: Bufferless NoC (BLESS)

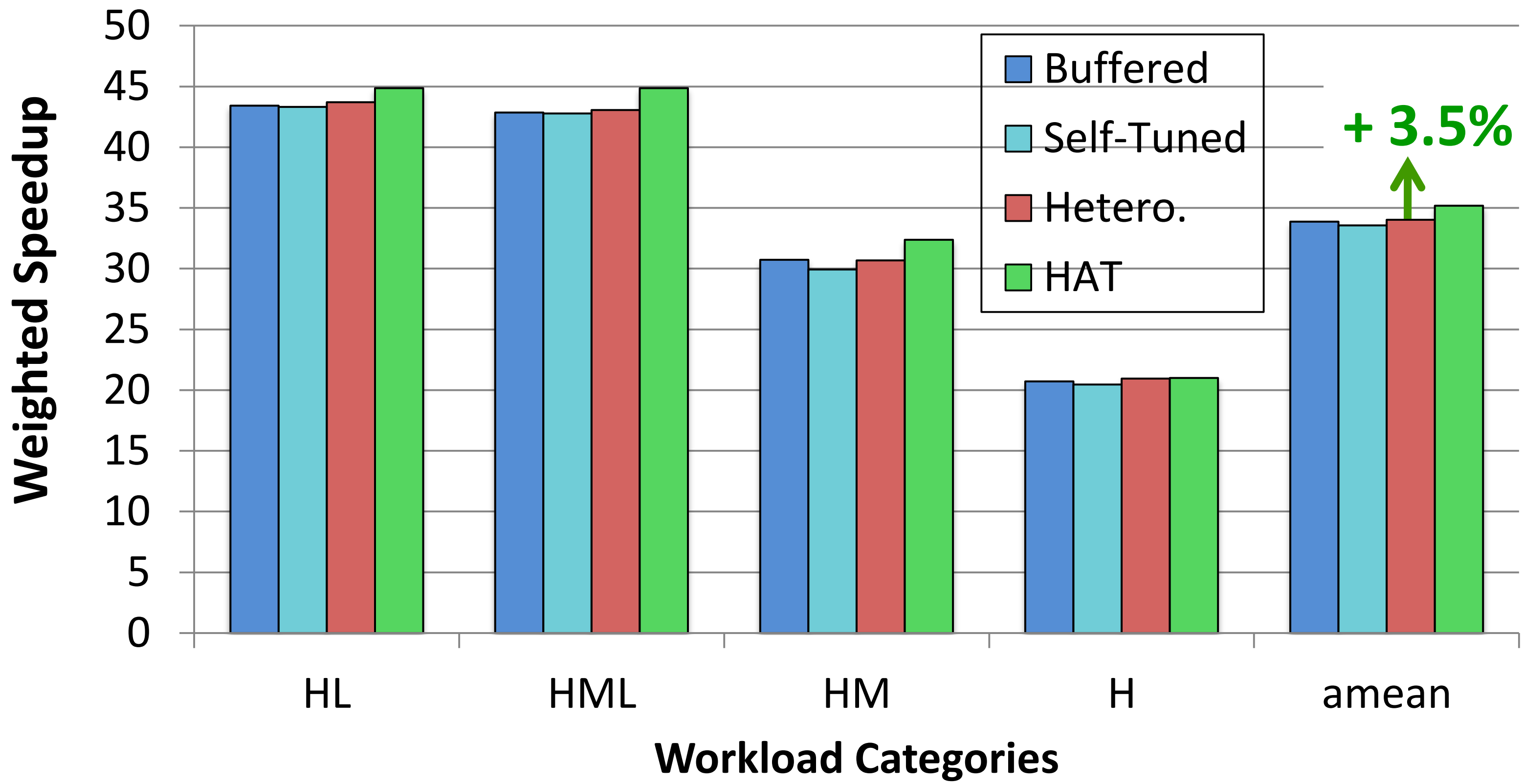


HAT provides better performance improvement than past work

Highest improvement on **heterogeneous** workload mixes

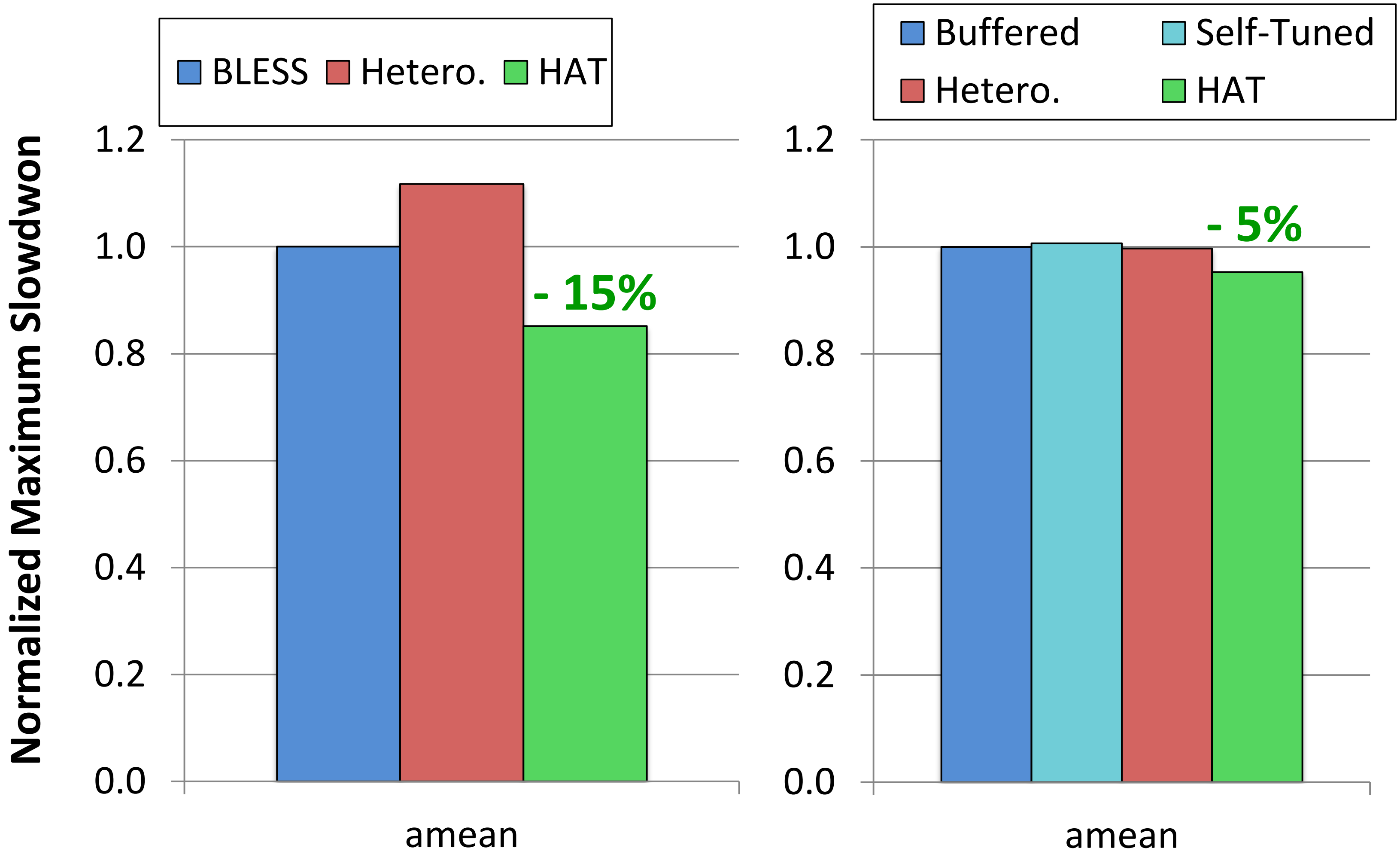
- L and **M** are more **sensitive** to network latency

Performance: Buffered NoC



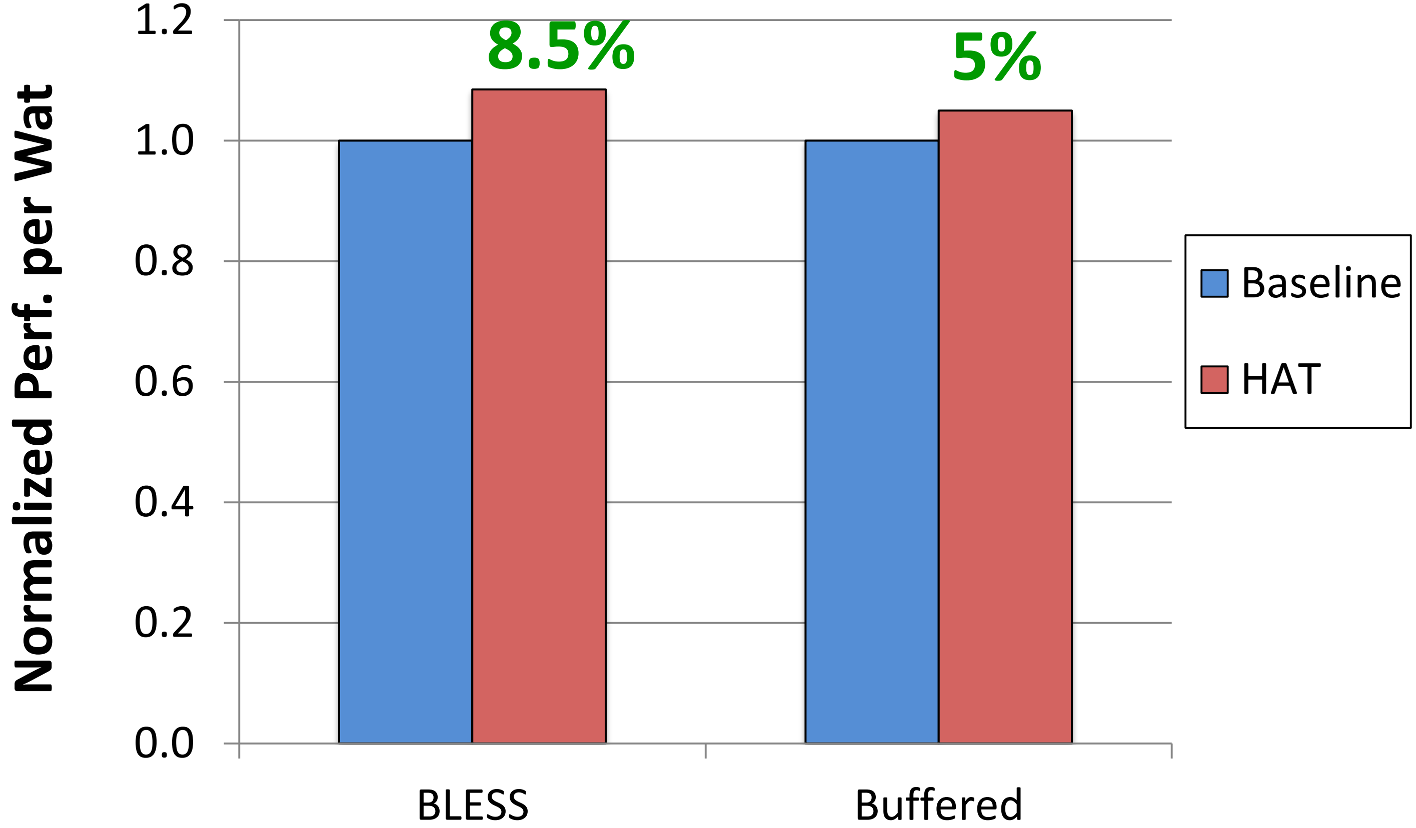
Congestion is much lower in Buffered NoC, but **HAT** still provides performance benefit

Application Fairness



HAT provides better fairness than prior works

Network Energy Efficiency



HAT increases energy efficiency by reducing congestion

Other Results in Paper

- Performance on **CHIPPER**
- Performance on **multithreaded** workloads
- Parameters sensitivity sweep of **HAT**

Conclusion

- **Problem:** Packets contend in on-chip networks (NoCs), causing congestion, thus reducing performance
- **Observations:**

  1) Some applications are more sensitive to network latency than others
  2) Applications must be throttled differently to achieve peak performance
- **Key Idea: Heterogeneous Adaptive Throttling (HAT)**
  1) Application-aware source throttling
  2) Network-load-aware throttling rate adjustment
- **Result:** Improves performance and energy efficiency over state-of-the-art source throttling policies

HAT: Heterogeneous Adaptive Throttling for On-Chip Networks

Kevin Kai-Wei Chang,
Rachata Ausavarungnirun,
Chris Fallin,
Onur Mutlu

Carnegie Mellon University SAFARI

Throttling Rate Steps

Overhead

Multithreaded Workloads