PCCS: Processor-Centric Contention Slowdown Model for Heterogeneous System-on-Chips

Yuanchao Xu, Mehmet E. Belviranli, Xipeng Shen, Jeffrey Vetter

NC STATE UNIVERSITY

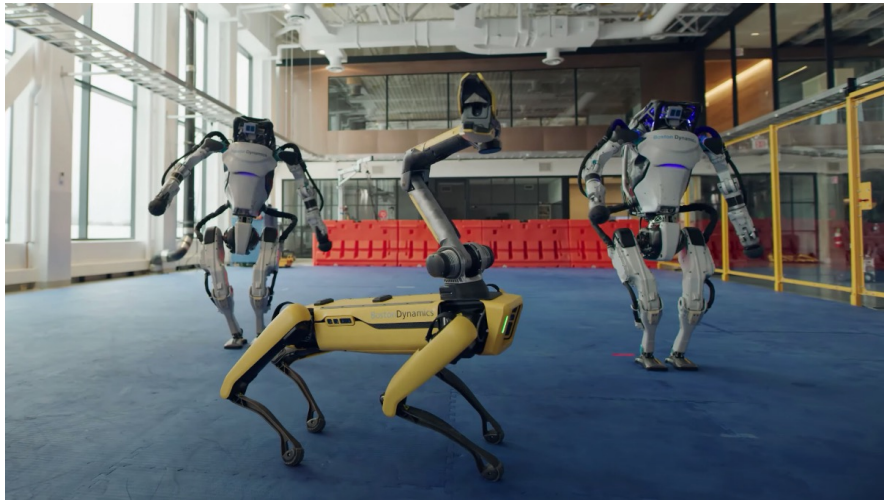
COLORADO SCHOOL OF MINES

OAK RIDGE National Laboratory

Heterogenous System-on-Chips (SoCs)

Smartphones

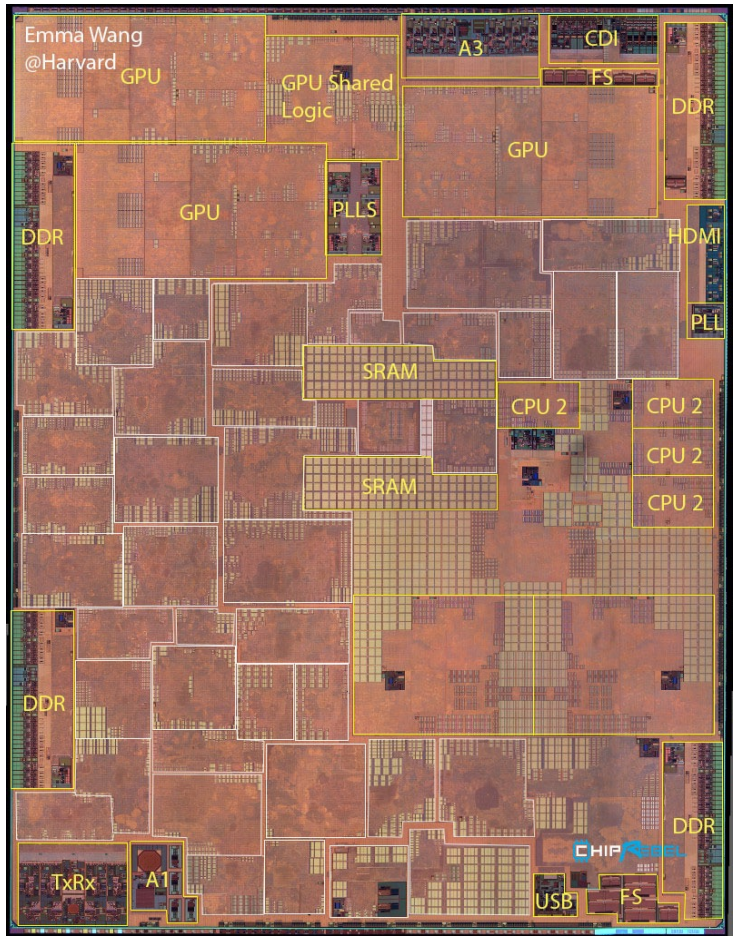
Autonomous Vehicles



Robots

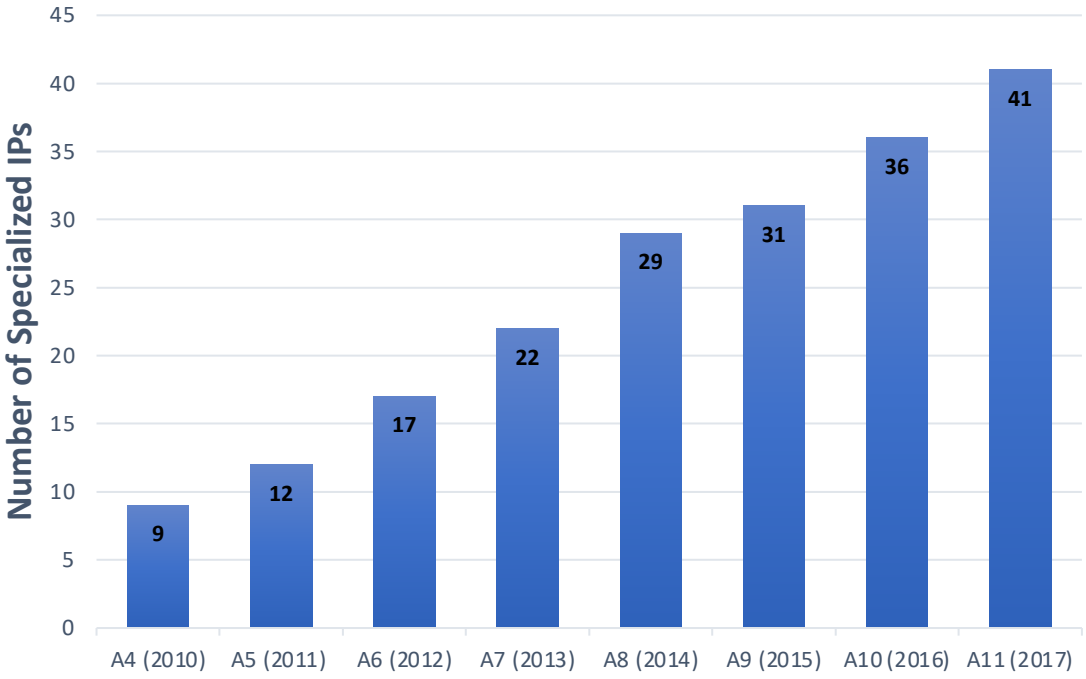
Drones

Diversely Heterogeneous Architectures



Apple A11 die photo

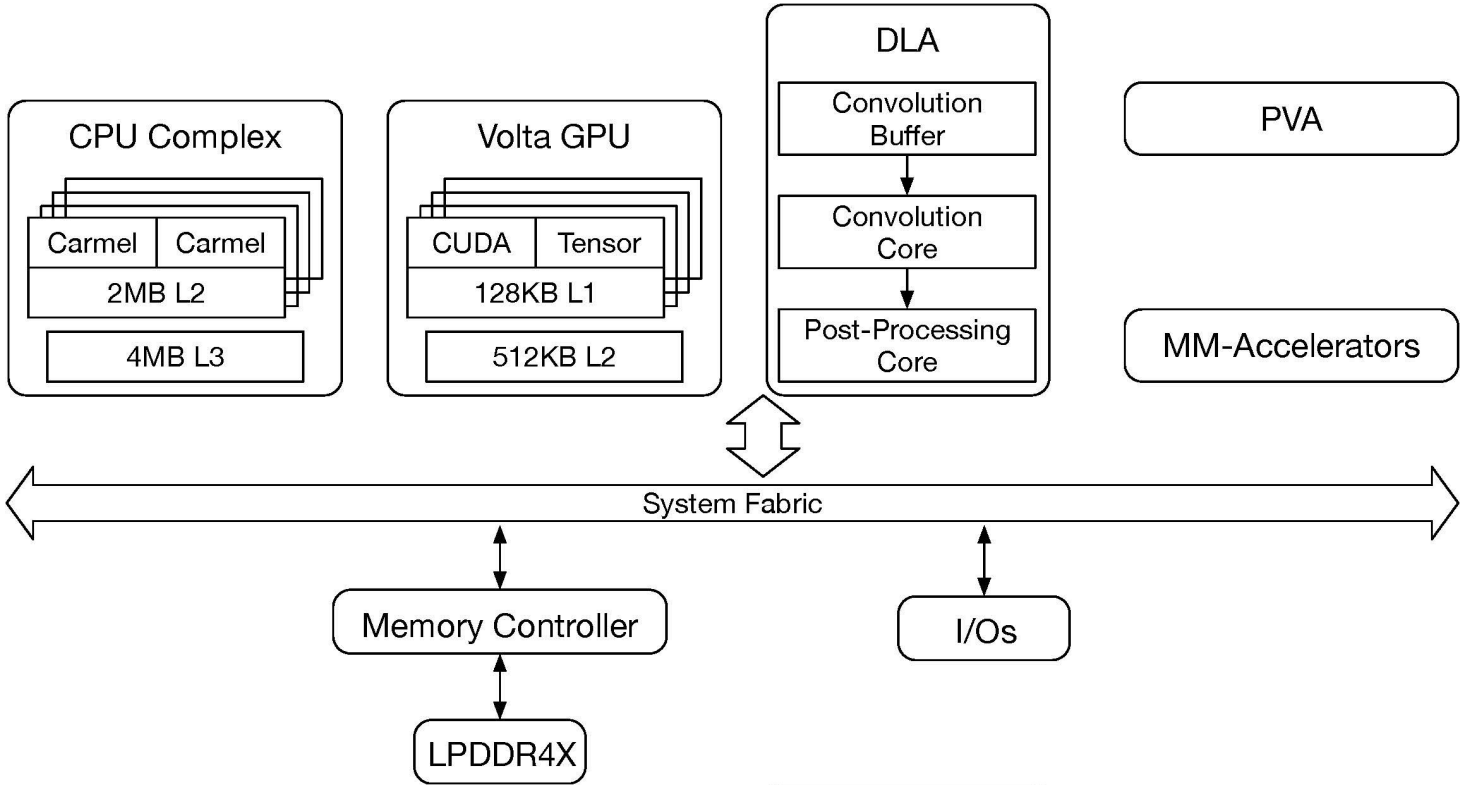
More than 40 intellectual property (IP) blocks in A11!



Source: http://vlsiarch.eecs.harvard.edu/research/accelerators/die-photo-analysis/

Heterogenous SoC Example

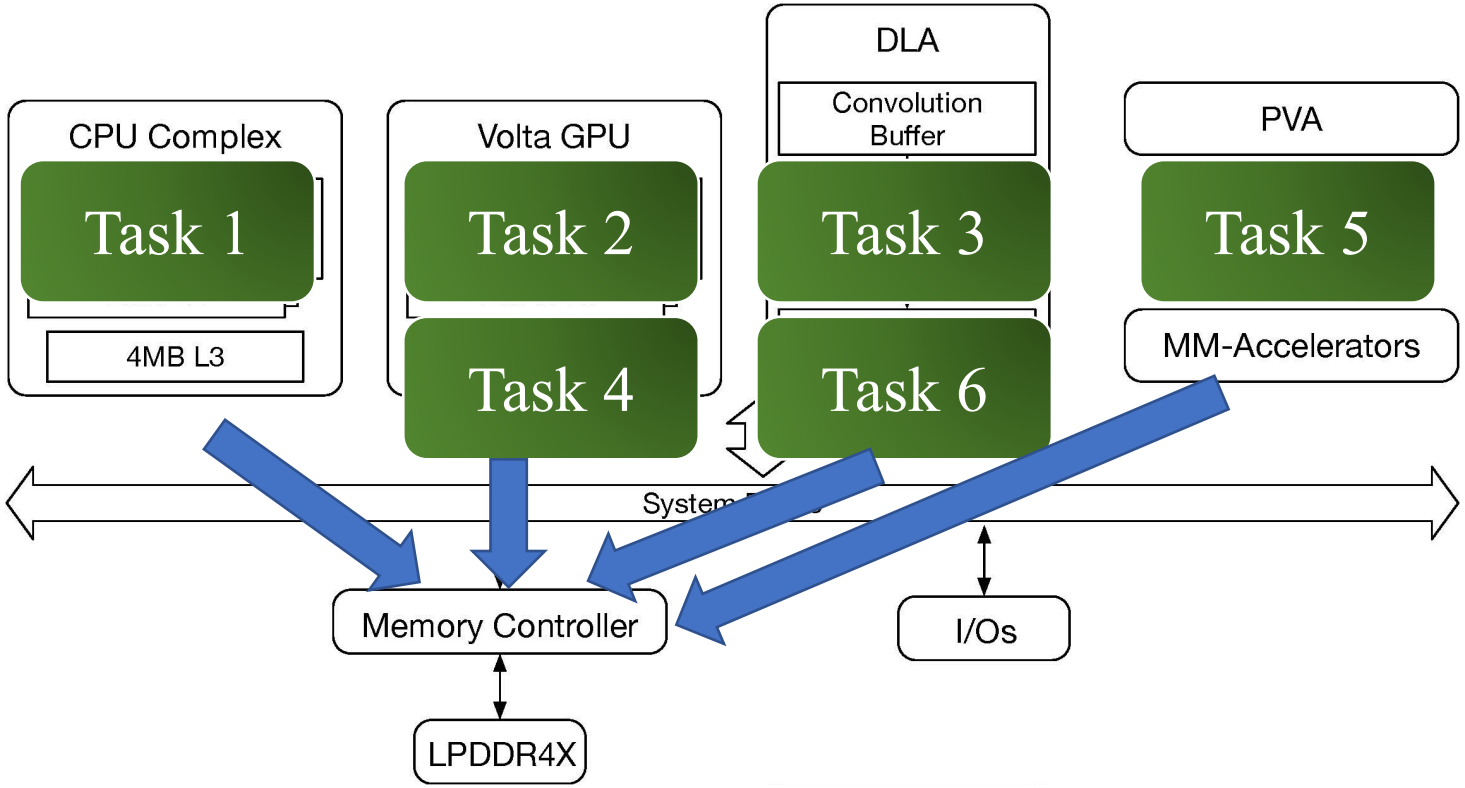
5-processor unit (PU) NVIDIA Jetson Xavier



Application

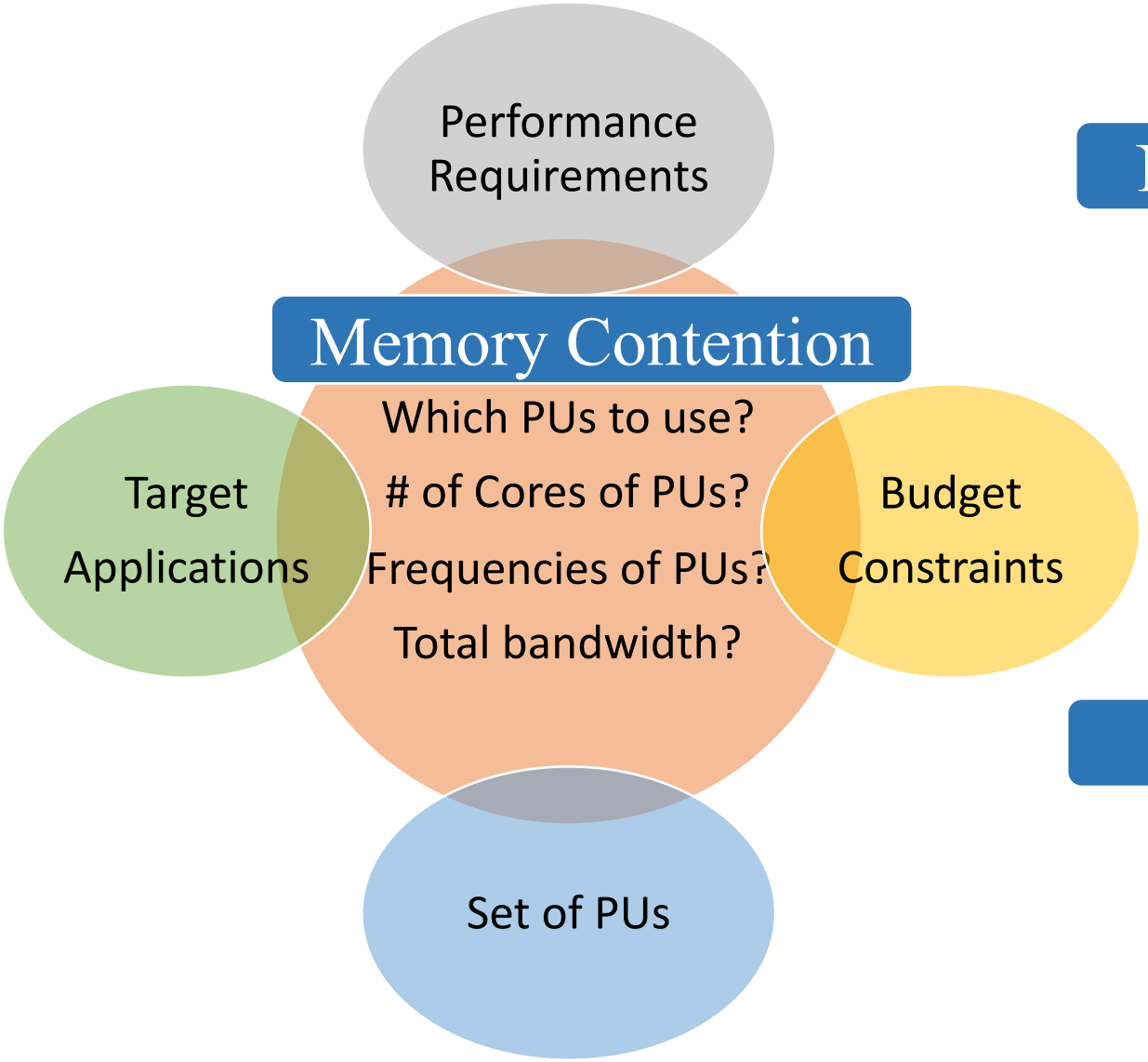
Heterogenous SoC Example

5-processor unit (PU) NVIDIA Jetson Xavier

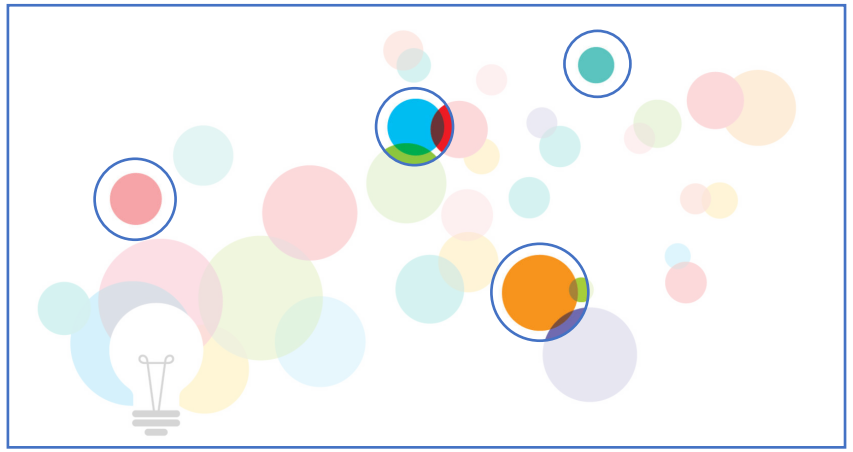


Application

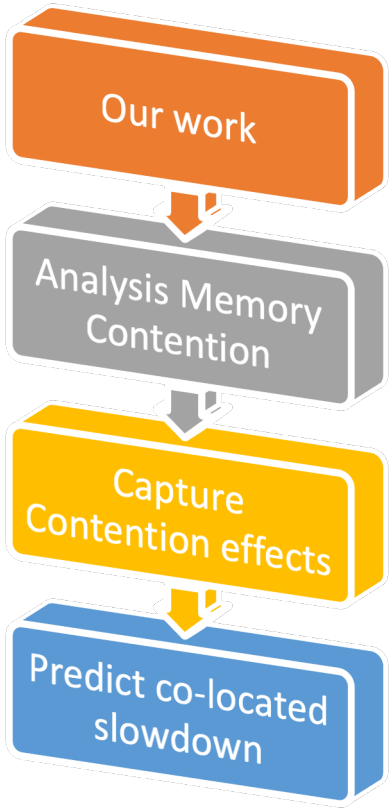
Commercial SoCs are Challenge To Design



Numerous possible designs



Several efficient designs



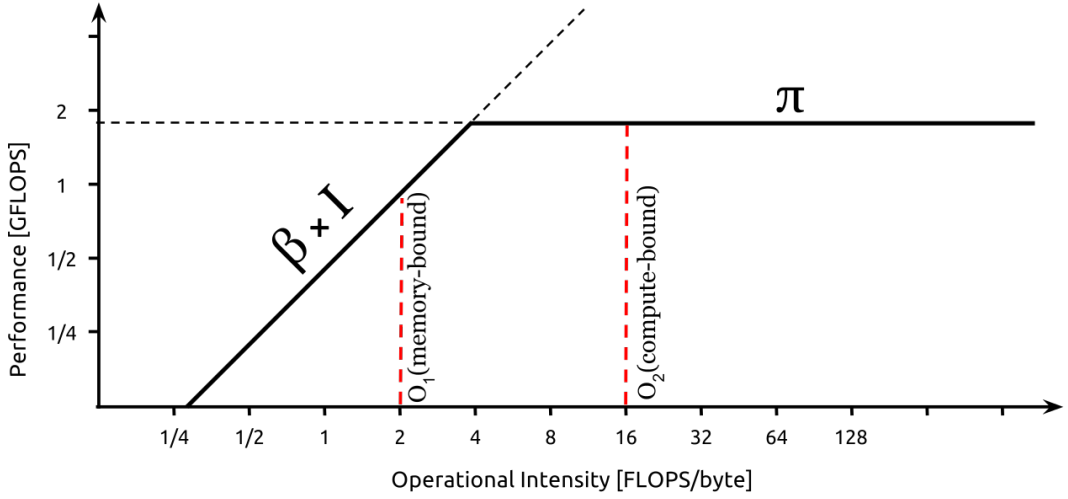
Contributions

- Built the **fundamental understanding** of memory contention on SoCs
- Designed **processor-centric contention-aware slowdown model**
- Predicted accurate memory contention effects, improving **70% prediction accuracy** over the state-of-the-art work
- Identified the **real bottleneck** of SoCs at pre-silicon stage, saving up to **50% frequency or number of cores**

Outline

- Background
- Observation
- Architectural analysis
- PCCS model
- SoC design guidance

Computer Architecture & Performance Models



Roofline Model

Performance model vs. Simulation

- **More insights**
- **Less efforts and time**
- **Don't need architectural details**

Gables Our work

Performance Models

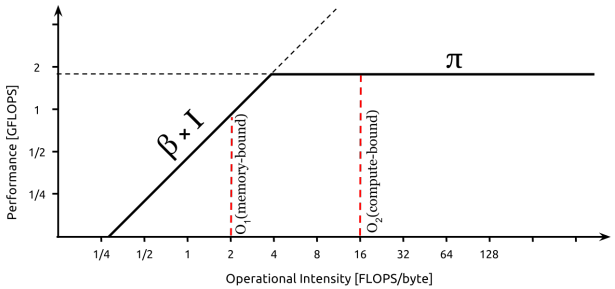
Next generation SoCs ← SoCs → Better utilization

Pre-silicon

Post-silicon

Source: https://commons.wikimedia.org/wiki/File:Example_of_a_naive_Roofline_model.svg

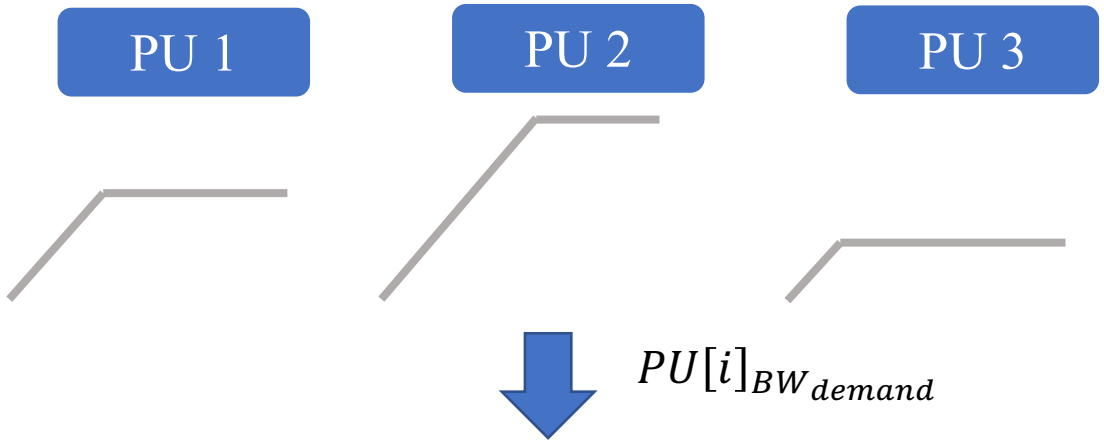
State-of-the-art Contention Model: Gables [1]



Roofline Model

+

Memory Bandwidth Contention Model



$$PU[i]_{BW_{demand}}$$

Total BW demand < Peak BW

$$PU[i]_{obtained_{BW}} = PU[i]\_BW\_demand$$

Total BW demand > Peak BW

$$PU[i]_{obtained_{BW}} = PU[i]_{BW_{demand}} * \frac{Peak\_BW}{Total\_BW\_demand}$$

[1] Mark D. Hill, et al., Gables: A Roofline Model for Mobile SoCs, HPCA 2019

Something Gables Missing

Observe task 1's performance:

Task 1

PU 1 (GPU)

Vary the total external bandwidth demand from other tasks

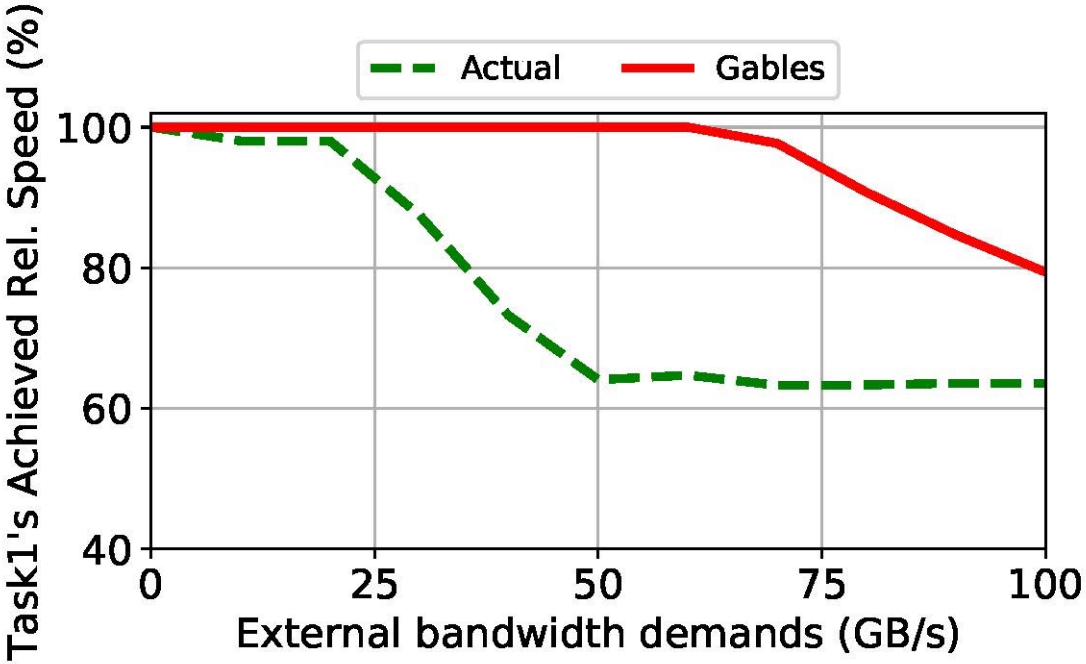
Task 2

PU 2 (CPU)

Task 3

PU 3 (DLA)

Vary the task 1 BW demand



Observations

Observe task 1's performance:

Task 1

PU 1 (GPU)

Vary the total external bandwidth demand from other tasks

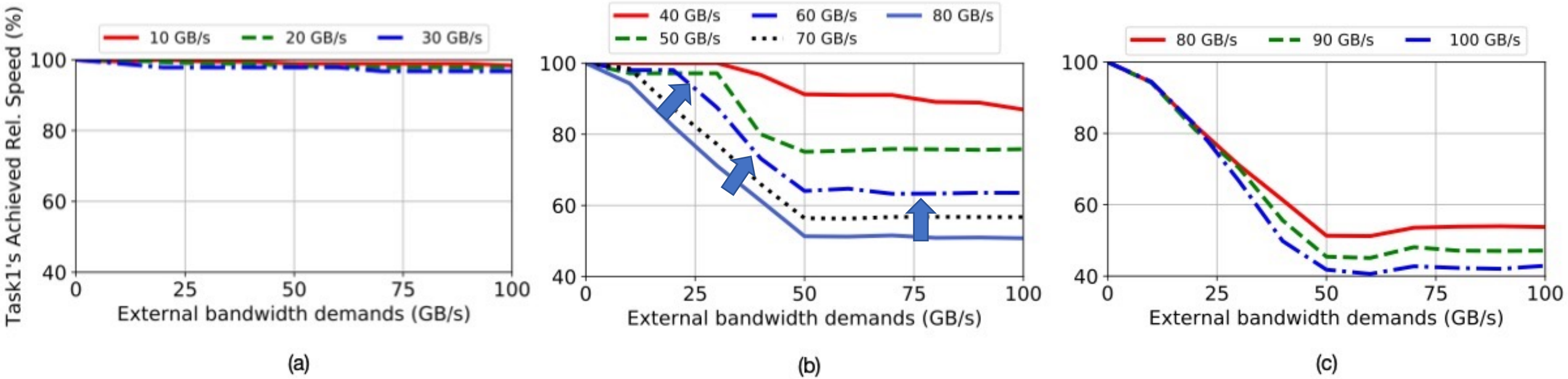
Task 2

PU 2 (CPU)

Task 3

PU 3 (DLA)

Vary the task 1 BW demand



Task 1 BW demand ≤ 30 GB/s

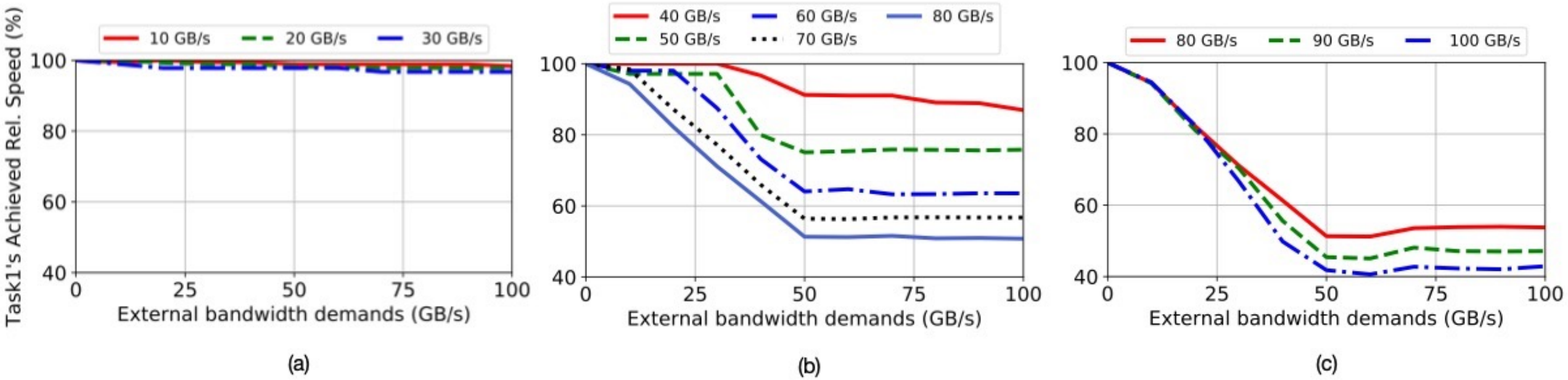
40 GB/s ≤ Task 1 BW demand ≤ 80 GB/s

Task 1 BW demand ≥ 80 GB/s

Outline

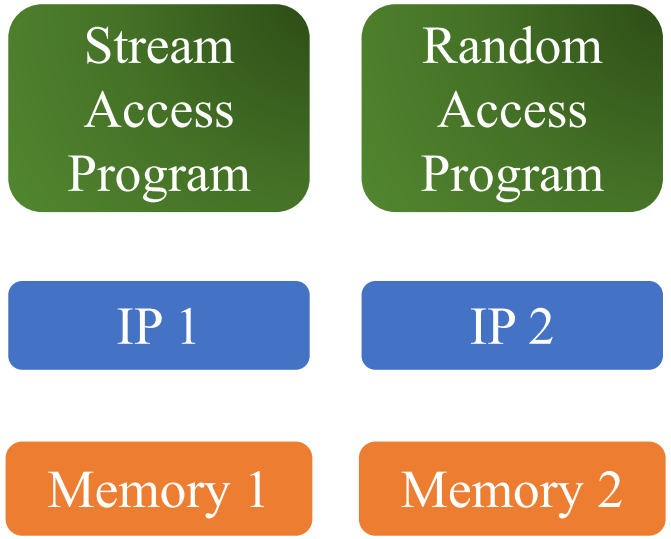
- Background
- Observation
- Architectural analysis
- PCCS model
- SoC design guidance

Architectural analysis



Hypothesis: Caused by fairness control in memory system

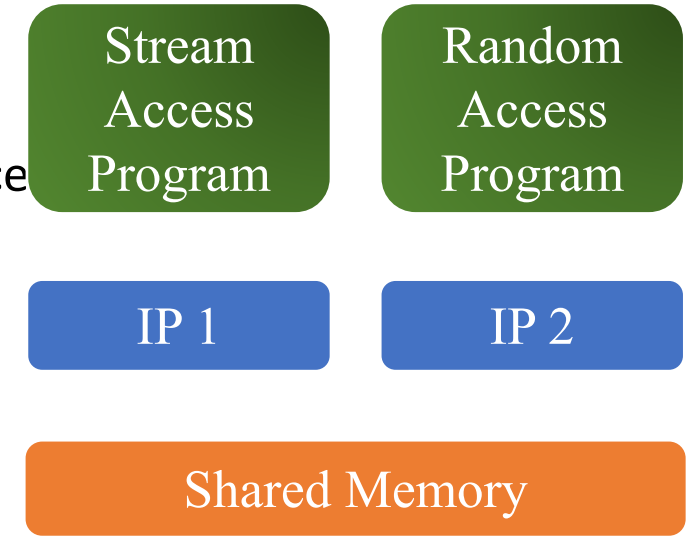
Purpose of Fairness Control



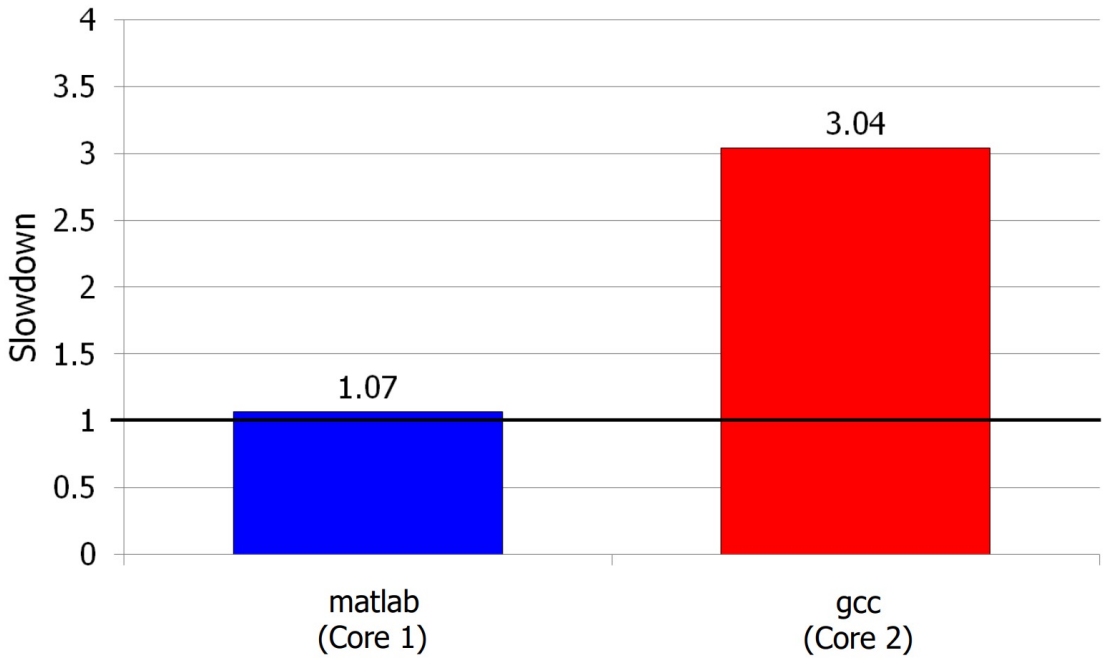
Thomas Moscibroda, et al., Memory Performance Attacks: Denial of memory service in multi-core systems, USENIX Security 2007

Purpose of Fairness Control

Similar Co-located Performance



Significant slowdown



Fairness control is important in memory controller

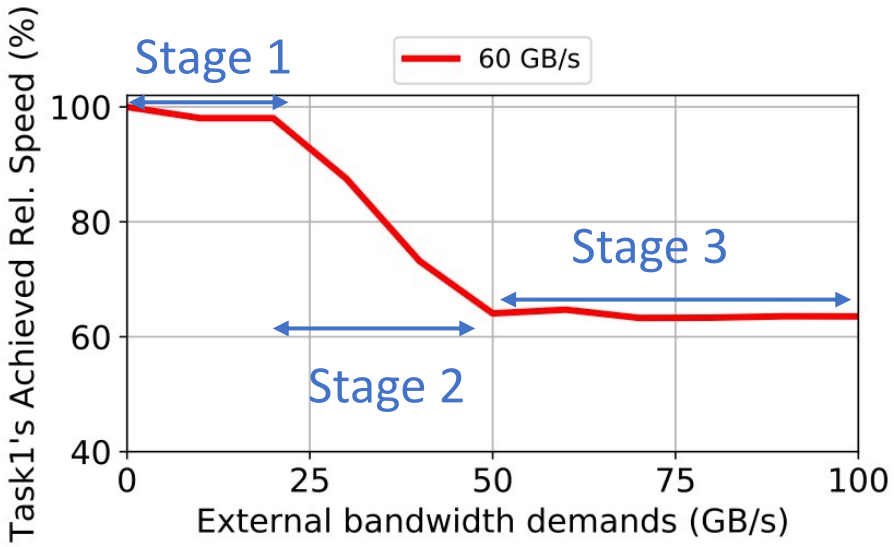
- Reduce denial of memory service vulnerabilities
- Improve overall throughput
- Improve quality-of-service

Thomas Moscibroda, et al., Memory Performance Attacks: Denial of memory service in multi-core systems, USENIX Security 2007

Bandwidth Partitions with Fairness Control

ATLAS[1]: Similar attained service time

- Peak BW 80GB/s



	Stage 1	Stage 2	Stage 3
BW demands:	Task1: 60GB/s Task2: 20GB/s	Task1: 60GB/s Task2: 30GB/s	Task1: 60GB/s Task2: 80GB/s
Obtained BW:	Task1: 60GB/s Task2: 20GB/s	Task1: 50GB/s (83% of 60GB/s) Task2: 30GB/s	Task1: 40GB/s (67% of 60GB/s) Others: 40GB/s

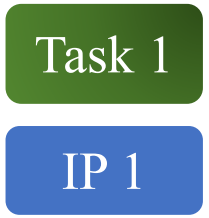
[1] Yoongu Kim, et al., ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers, HPCA 2010

Memory Controller Policy Simulation

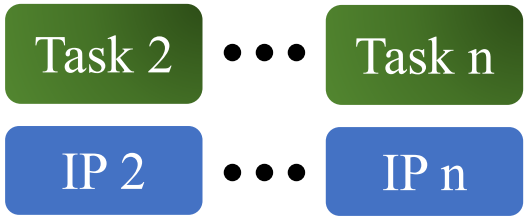
Methodology:

1. Implement memory controller policies in Ramulator[1]

2. Observe task 1's performance:



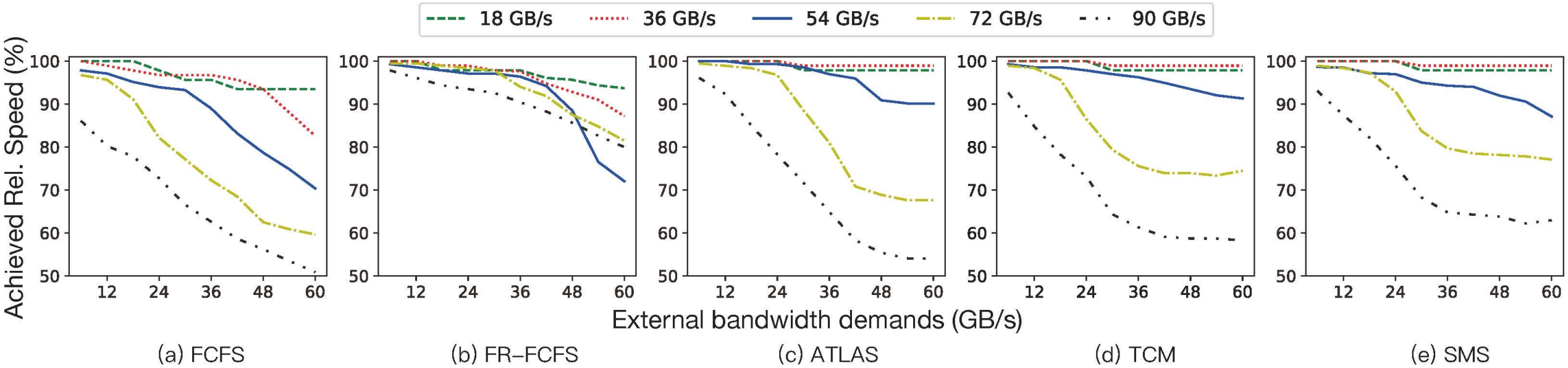
Vary the total external bandwidth demand for other tasks



3. Vary task 1 bandwidth demands; Repeat Step 2

[1] https://github.com/CMU-SAFARI/ramulator

Memory Controller Policy Simulation



Close to Gables model

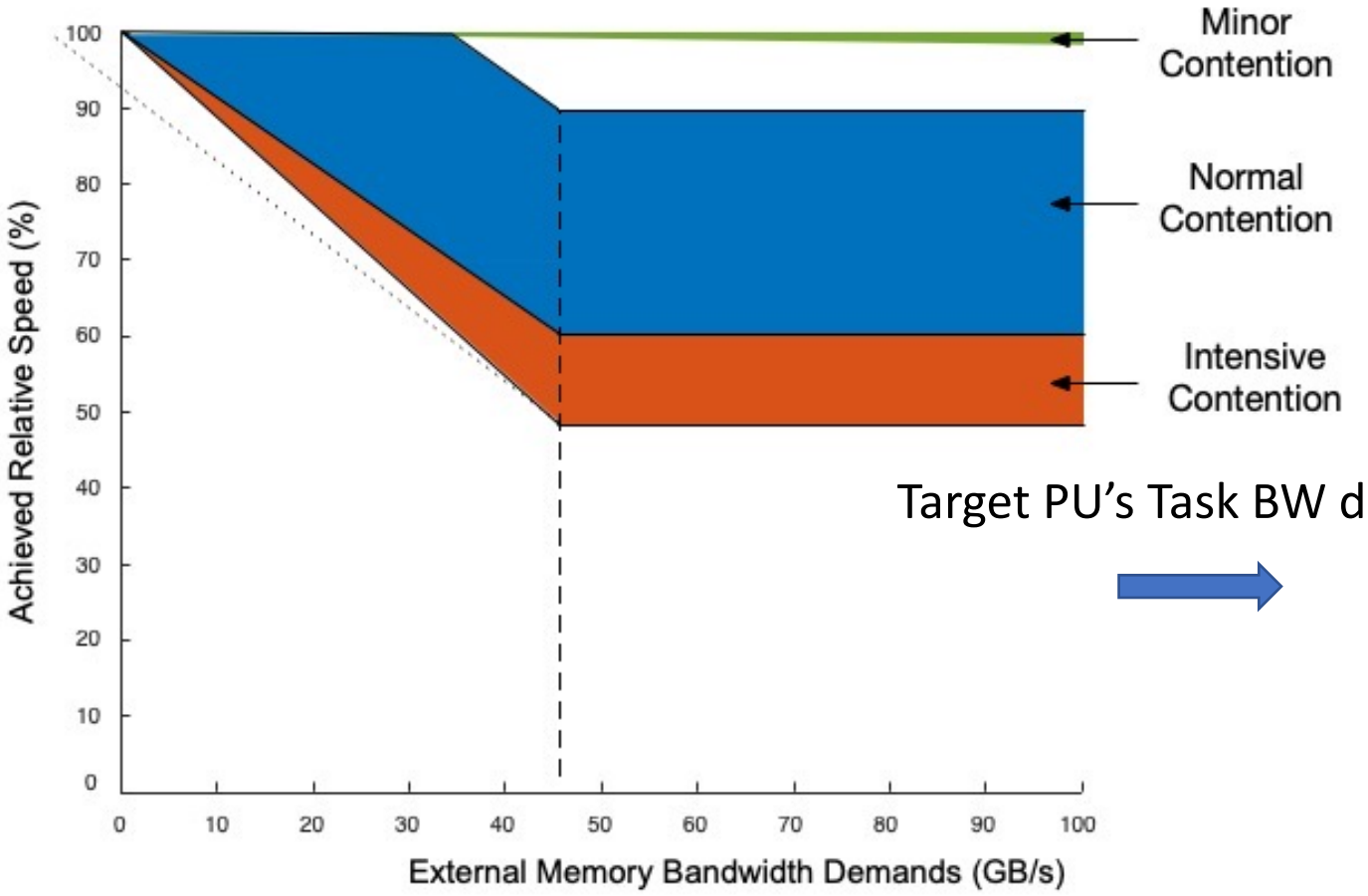
Close to the observations

Name	Policies
FCFS	First-come-first-server.
FR-FCFS	First ready; First-come-first-server.
ATLAS	Higher the ranks of threads with least attained service.
TCM	Higher non-memory-intensive programs
SMS	Shortest first round robin

Outline

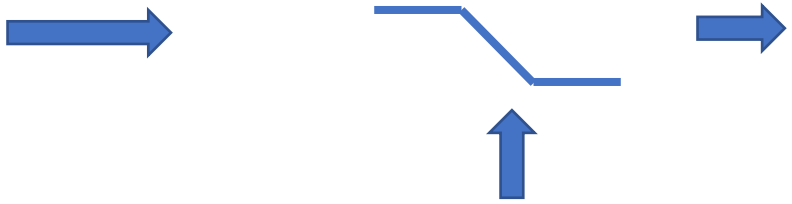
- Background
- Observation
- Architectural analysis
- PCCS model
- SoC design guidance

Processor-Centric Contention-aware Slowdown Model (PCCS)



- Three-region interference-conscious Model
- Piecewise slowdown functions in each region

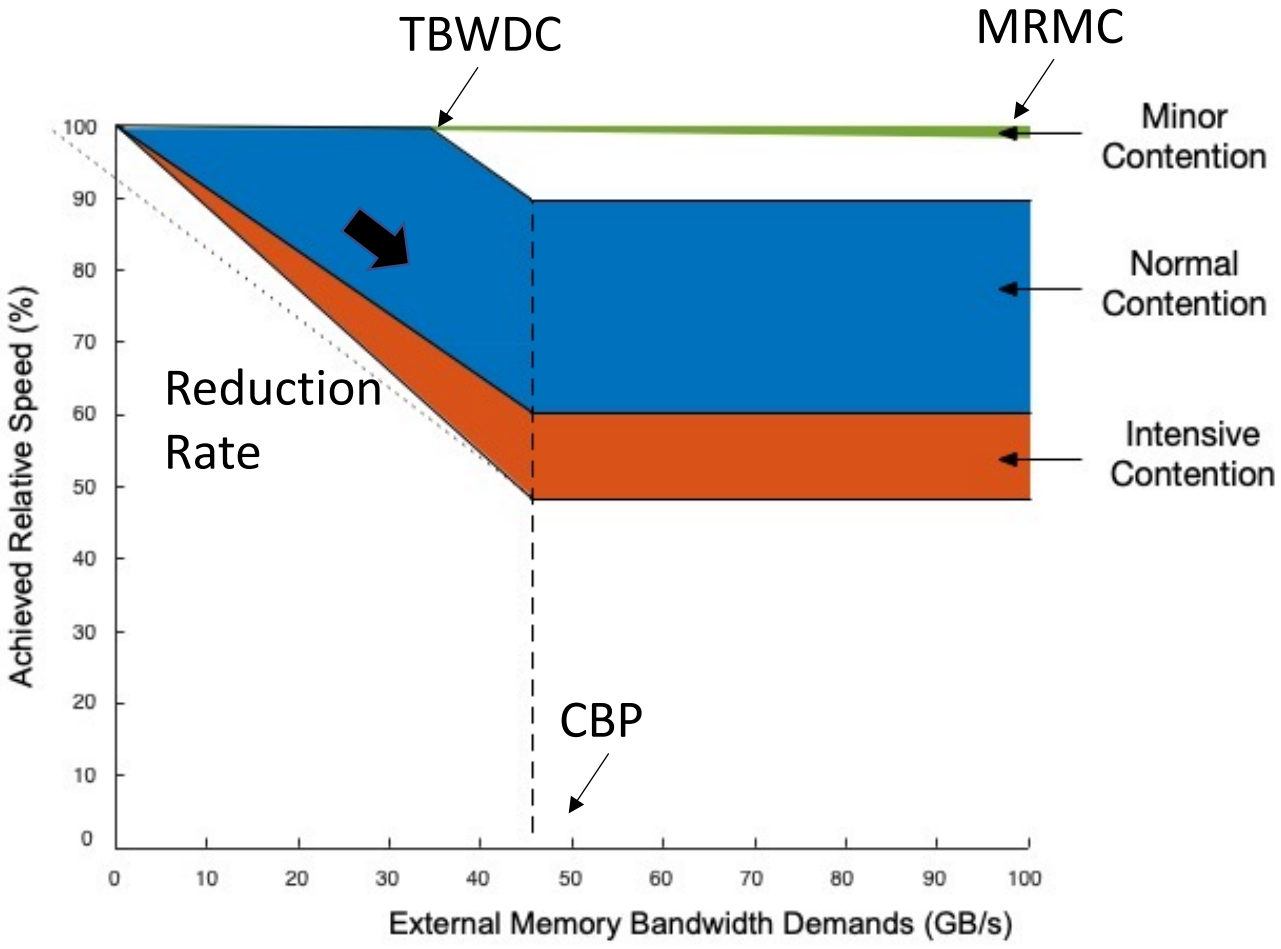
Target PU's Task BW demand



External BW demand

Achieved relative speed under memory contention(%)

PCCS parameters



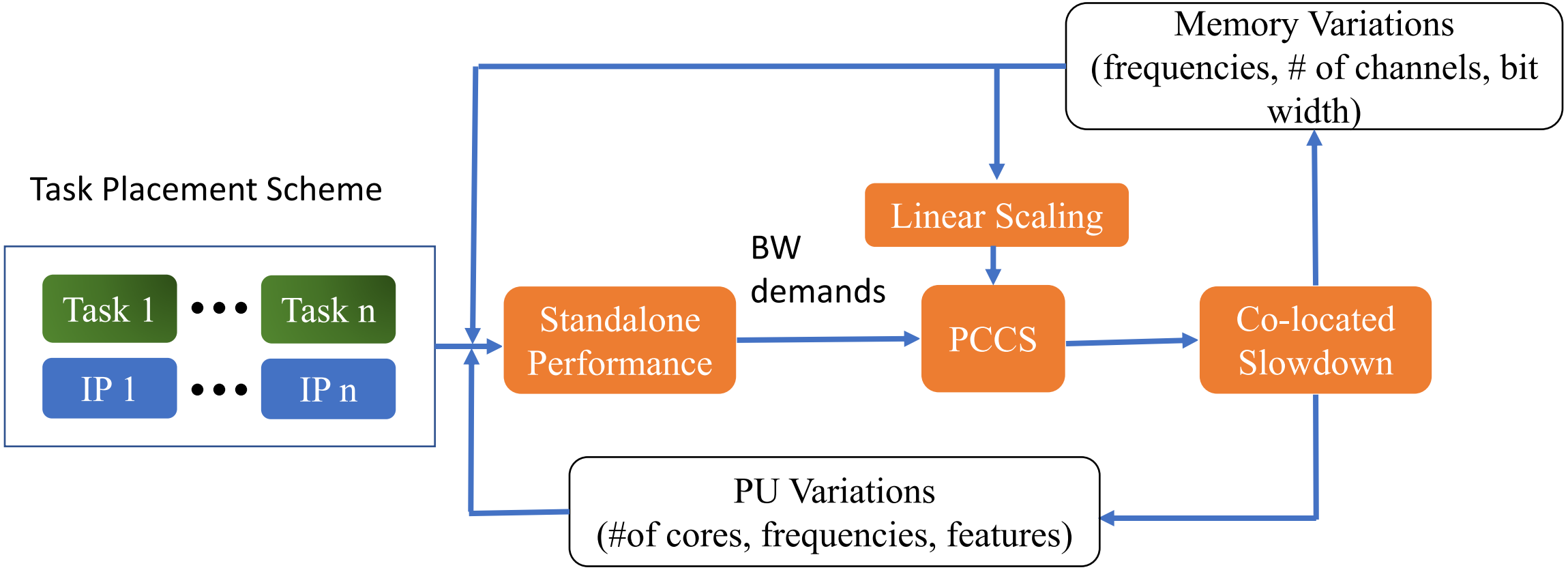
- Maximum Reduction of Minor Contention (MRMC)
- Total Bandwidth Demand with Contention (TBWDC)
- Contention Balance Point (CBP)
- Reduction Rate

More detail in the paper

Outline

- Background
- Observation
- Architectural analysis
- PCCS model
- SoC design guidance

PCCS Model Usage



Case study

Program: streamcluster in Rodinia Benchmark Suite

Objectives: Find appropriate frequencies of the GPU within 5% and 20% co-located slowdown on Xavier when external BW demands are 20GB/s, 40GB/s and 60GB/s

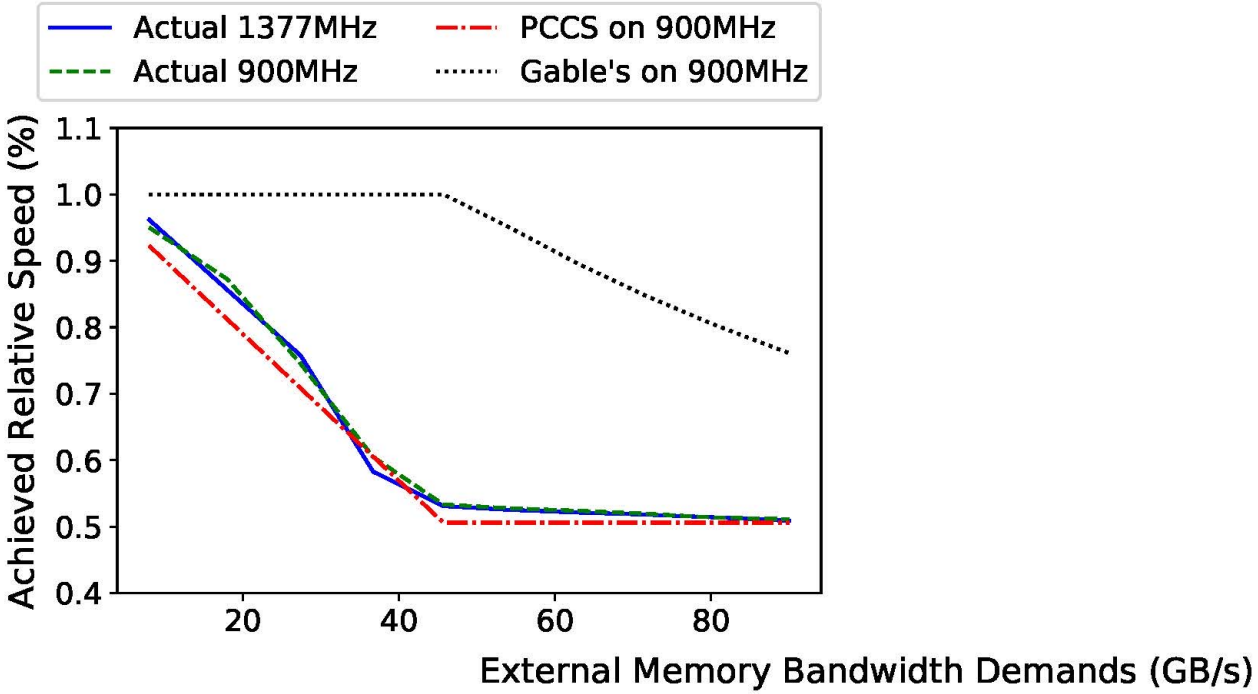
Methodology: PCCS model usage

Validation: Change the power settings of Xavier to obtain the ground truth

Baseline: Gables model

Results on *streamcluster* benchmark

Objective: within 5% slowdown



External BW demand (GB/s)		20	40	60	Average	20	40	60	Average
		Errors							
		PCCS				Gables			
Maximum Allowed Slowdown	5%	2.4	3.1	1.6	2.4	4.8	35.4	41.9	27.4
	20%	1.3	1.7	3.6	2.2	3.8	36.7	49.1	29.9

Conclusion

- Built the **fundamental understanding** of memory contention on SoCs
- Designed three-region **processor-centric contention-aware slowdo**
- Predicted accurate memo effects, improving **70% prediction accuracy** over the state-of-the-art work
- Identified the **real bottleneck** of SoCs at pre-silicon stage, saving up to **50% frequency or number of cores**

Thank you for your attentions!
Q & A