



37th IEEE International Conference on  
Application-specific Systems, Architectures and Processors

# MPTAC: Memory Controller Partitioning and Traffic-Aware Contention Control in GPUs

Vahid Geraeinejad<sup>1</sup>, Paul Delestrac<sup>1</sup>, Javier Barrera<sup>2</sup>, Francisco Cazorla<sup>2</sup>,  
Masoumeh Ebrahimi<sup>1</sup>, Ahmed Hemani<sup>1</sup>

<sup>1</sup> KTH Royal Institute of Technology

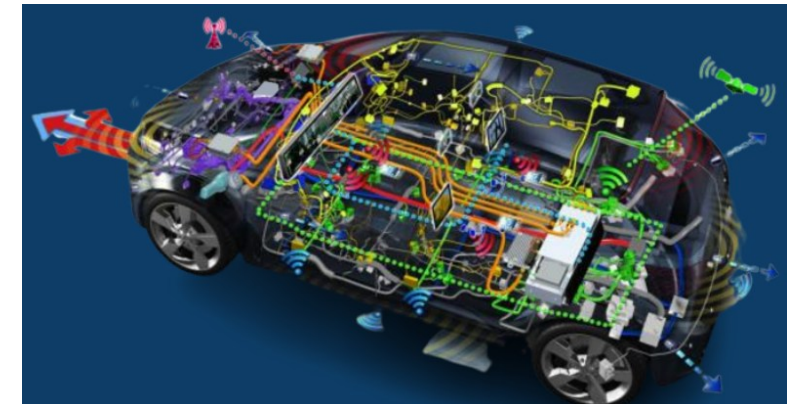
<sup>2</sup> Barcelona Supercomputing Center (BSC)

# Introduction

- Modern embedded systems are evolving
  - Large number of AI-driven software features running simultaneously
  - Unprecedented performance demands
- Resource sharing has gained prominence, especially in embedded GPUs
- Real-time devices require Execution Time Determinism (ETD)
- Resource isolation is required to offer ETD



KTH Dardel supercomputer

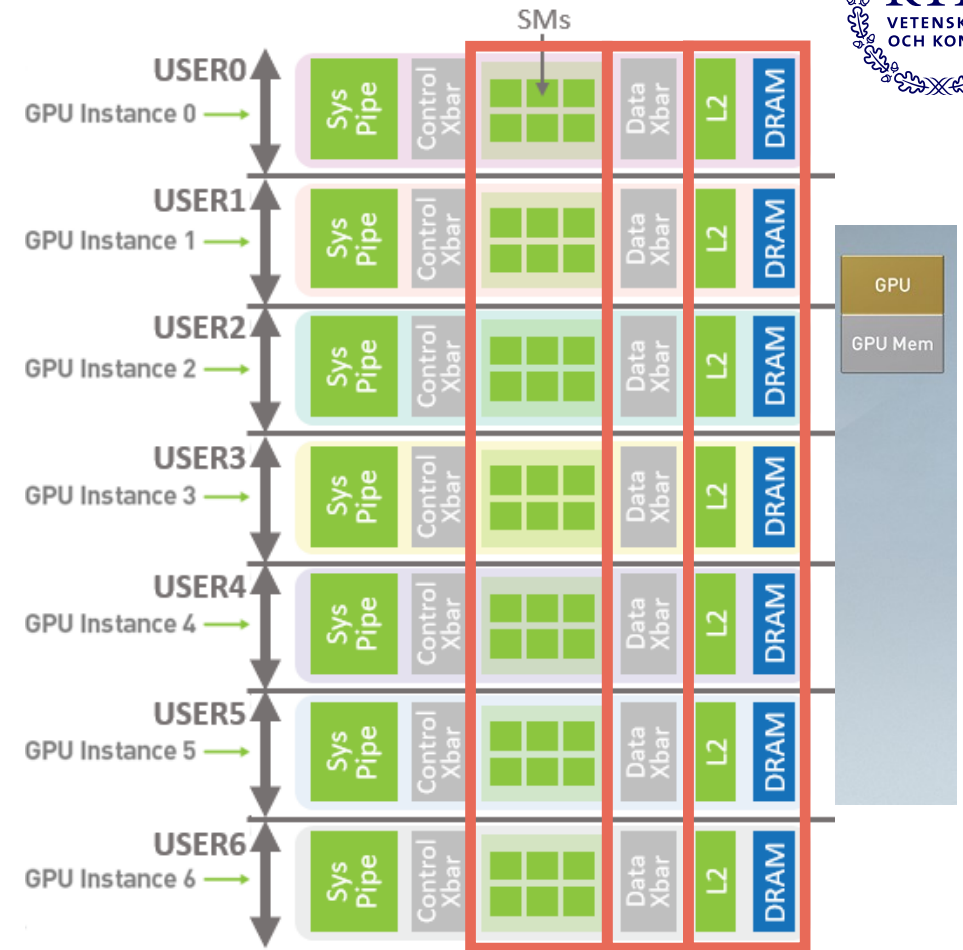


Automotive Embedded Systems [1]

[1] Indian Institute of Embedded Systems (IIES), "Automotive Embedded Systems Course," IIES, [Online]. Available: <https://iies.in/crash-courses/automotive-embedded-system/>. [Accessed: Aug. 17, 2026].

# Multi-Instance GPU

- Multi-Instance GPU (MIG) is the most practical GPU resource sharing solution
  - Pre-defined fixed-size GPU instances.
  - True resource isolation.
- Streaming Multiprocessors (SMs)
- Network on Chip (NoC)
- L2 and DRAM slice



Multi-Instance GPU (MIG)<sup>2</sup>

[1] NVIDIA, "NVIDIA Multi-Instance GPU," 2026, [Online]. Available: <https://www.nvidia.com/en-us/technologies/multi-instance-gpu/>. [Accessed: Apr. 17, 2026].

[2] NVIDIA, "NVIDIA Multi-Instance GPU User Guide," 2024, [Online]. Available: [https://docs.nvidia.com/datacenter/tesla/pdf/NVIDIA\\_MIG\\_User\\_Guide.pdf](https://docs.nvidia.com/datacenter/tesla/pdf/NVIDIA_MIG_User_Guide.pdf). [Accessed: Apr. 17, 2026].

# Goal Statement

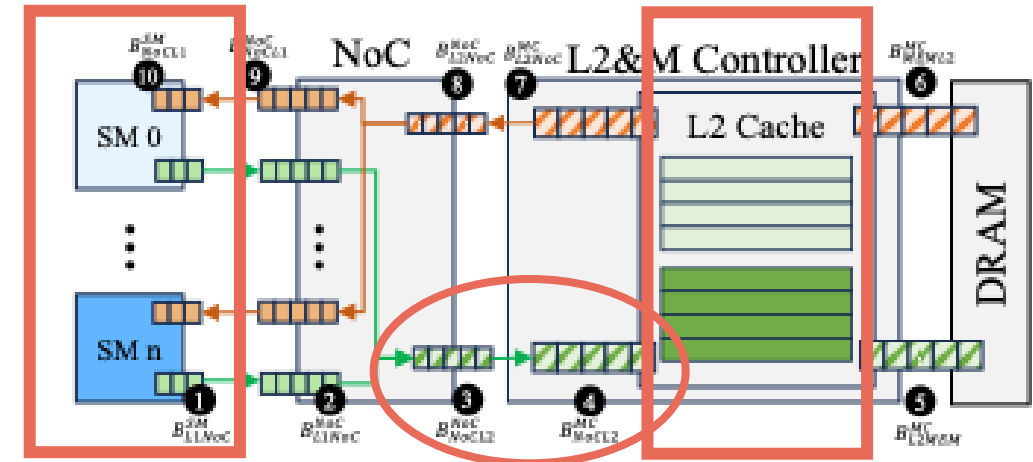
- Since the number of tasks exceeds partitions, sharing the same partition can face contention due to shared resources
- Our **goal** was to expose the extent of contention in partitioned execution and propose a solution to control it.

Applications sharing a single GPU partition

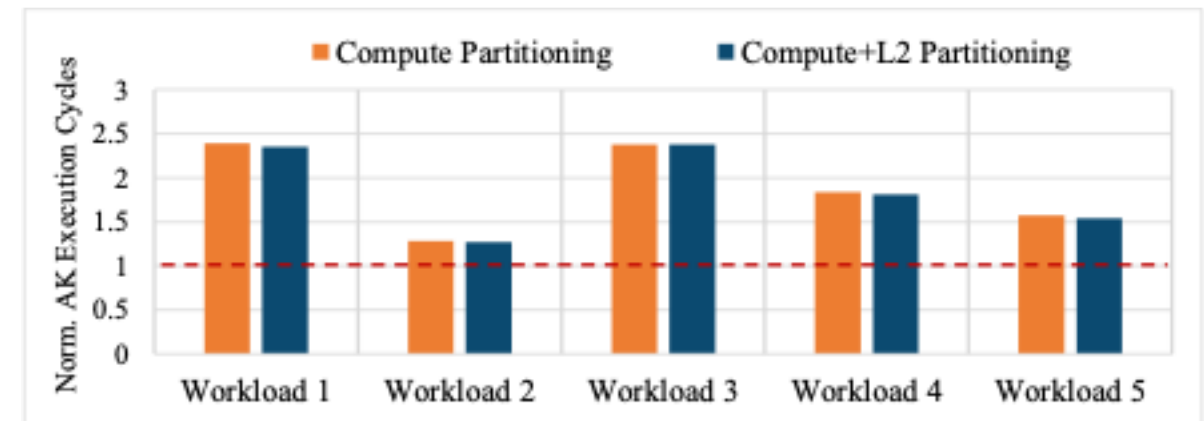


# Motivation – Finding the Source of Contention

- Analyzing the contention within a single MIG instance (partition).
  - Analysis Kernel (AK) with ETD requirements
  - Contender Kernels (CKs)
- Simulation setup
  - Compute partitioning and Compute+L2 partitioning
  - Five workloads, same AK, different contenders
- AK ETD is evaluated over execution time relative to in isolation execution
  - Inversely proportional to ETD
- AK execution time always depends on the contenders.
  - Lack of ETD (values significantly above 1)
- 2.38x to 1.26x Slowdown
- Neither setting is effective!
  - The degradation stems mainly from other shared components such as buffers



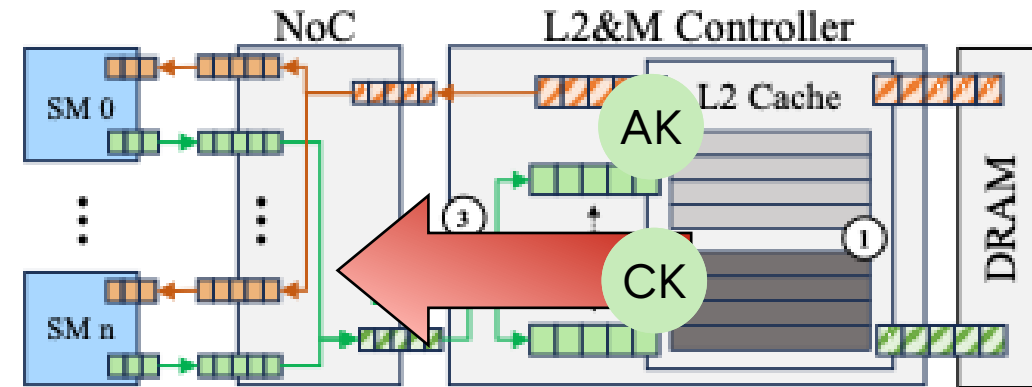
GPU partition baseline memory hierarchy



Normalized AK execution cycles, relative to isolated execution

# Proposed Method: MPTAC

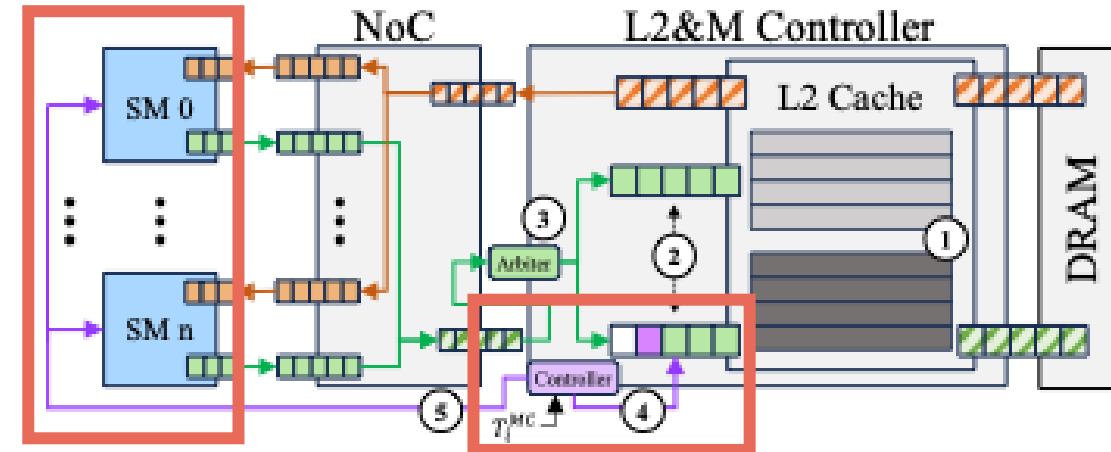
- Upstream buffers are the contention source
- L2 cache contention creates backpressure into the NoC and upstream buffers
- MPTAC: **M**emory controller **P**artitioning and **T**raffic-**A**ware **C**ontention control
- Partitioning upstream memory controller buffers and L2 to offer isolation
  - One partition for AK and one for the CKs



MPTAC memory controller partitioning

# Proposed Method: MPTAC

- Partitioning alone cannot resolve NoC backpressure
  - Head-of-Line blocking, AK request behind CKs
  - Partitioning other buffers is costly!
- Dynamic traffic aware contention control
  - Tracking traffic in contenders' buffer
  - Comparing the memory requests in each buffer against a configurable threshold  $T_i^{MC}$
  - Pausing memory requests at busy contender SMs using the controller
  - Flexible AK performance control with threshold



MPTAC dynamic traffic aware contention control

# Methodology

- MPTAC is evaluated over 2 memory controller partitions
  - One for AK and another for CKs
- Baseline compute+L2 partitioning
  - Modeled after NVIDIA Jetson AGX Orin Series
- Simulated using GPGPU-Sim v4.0
- Power analysis using AccelWattch

Baseline GPU partition configuration

| Parameter        | Value         | Parameter       | Value    |
|------------------|---------------|-----------------|----------|
| GPU Architecture | Ampere        | NoC buffer size | 32       |
| SMs              | 6             | L1 buffer size  | 32       |
| MC buffer size   | 64            | Warp scheduler  | LRR      |
| L1 data cache    | 192 KB per SM | L2 cache        | 512 KB   |
| DRAM             | 4 GB GDDR6    | NoC type        | Crossbar |

# Methodology - Workloads

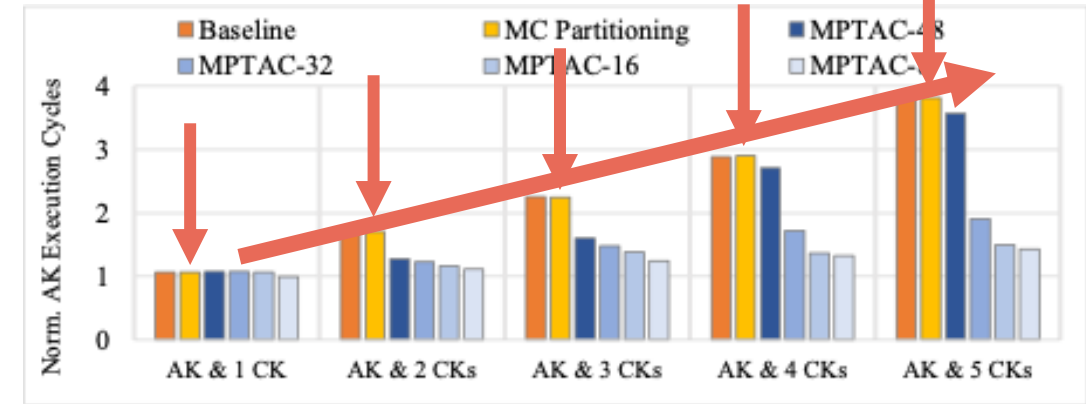
- We evaluate MPTAC using customized synthetic workloads to represent heavy contention scenarios
  - Performing large number of copies
  - Generating traffic on L2 cache and DRAM
- Rodinia benchmarks are used to represent more realistic scenarios

Baseline GPU partition configuration

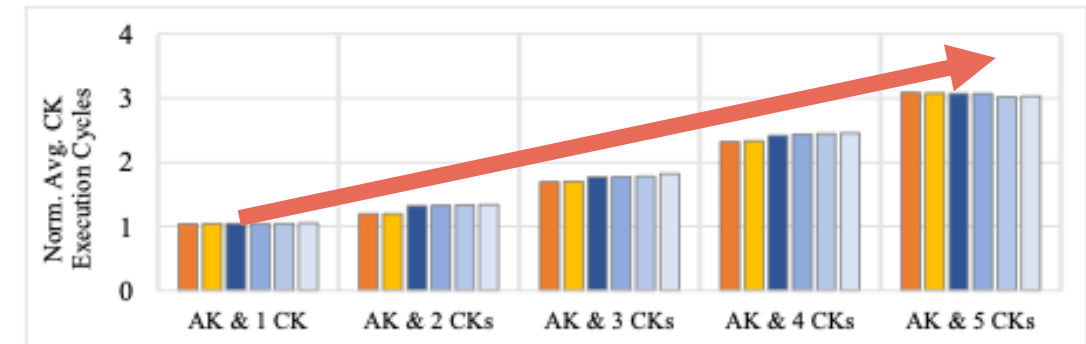
| Parameter       | MK  | LK  | Parameter                  | MK  | LK   |
|-----------------|-----|-----|----------------------------|-----|------|
| Kernel inputs   | 8   | 12  | Accesses/thread            | 8   | 120K |
| Input data size | 1K  | 10M | Copies/iter.               | 4   | 6    |
| Threads (total) | 1K  |     | Exec. Cycles ( $\approx$ ) | 10M | 34M  |
| Iter./thread    | 10K |     | Application                | AK  | CK   |

# Synthetic Workloads Overview

- Evaluating multiple setups
  - Baseline with compute+L2 partitioning
  - Compute+L2+MC partitioning
  - MPTAC with threshold  $T_i^{MC}$  Ranging from 48 to 8
- AK and CK average execution time relative to isolation
  - To represent ETD
- AK running with 1-5 contenders
- Increasing CKs leads to more slowdown
  - Most severe in 5 CKs: 3.83x slowdown
- MC partitioning alone cannot resolve contention and offer ETD.



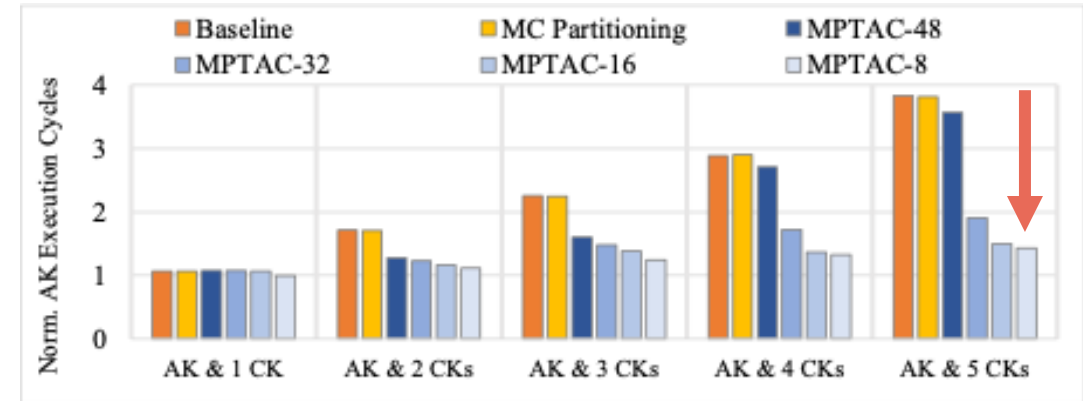
Normalized AK execution cycles relative to isolation



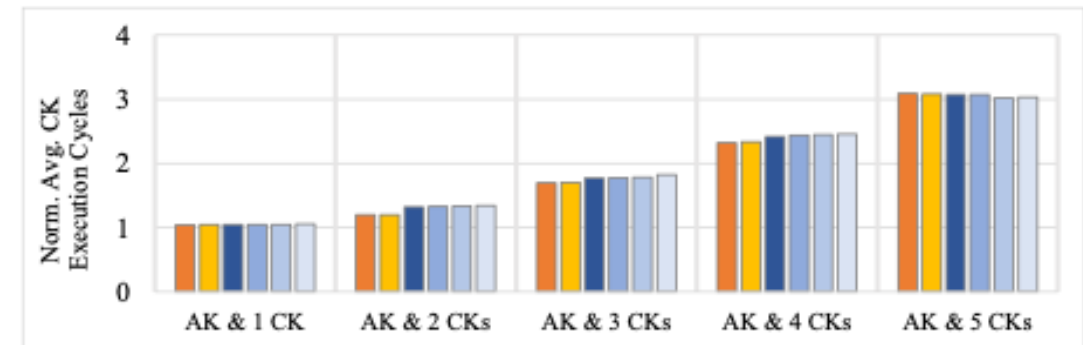
Normalized average CK execution cycles relative to isolation

# Synthetic Workloads Continued

- Dynamic traffic aware contention control in MPTAC resolves this
  - Most severe with 5 CKs: 3.83× to 1.43×, eliminating 84.8% of slowdown.
- Threshold provides a flexible way to tune acceptable performance.
  - Based on user requirements.
- Minimal CK execution overhead
  - Up to 12.5% slowdown with 2 CKs.
  - Scalable CK performance degradation.
  - Can reduce their execution time (2.2% with 5 CKs)



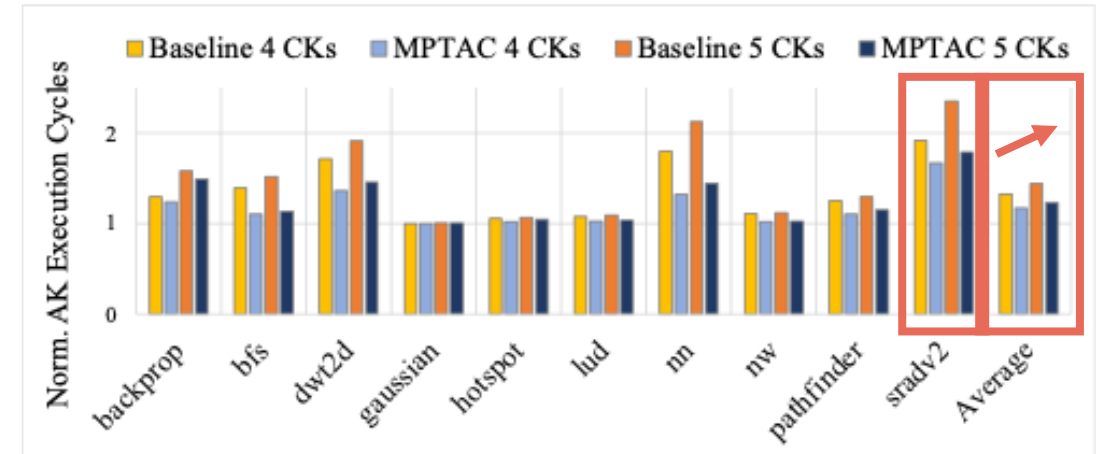
Normalized AK execution cycles relative to isolation



Normalized average CK execution cycles relative to isolation

# Representative Benchmarks

- AK is selected from Rodinia benchmarks
  - Only 4 and 5 CKs (LK) to represent highest stress
  - Evaluated MPTAC-8 (most effective isolation)
- 4 CKs: MPTAC reduces slowdown from 1.91x to 1.66x in the most severe case (sradv2, 27.5%)
- 5 CKs: 9% higher average slowdown, reduces slowdown from 2.35x to 1.78x in the most severe case (sradv2, 42.3%)
- Baseline power consumption (0.1%)



Normalized AK execution cycles relative to isolation

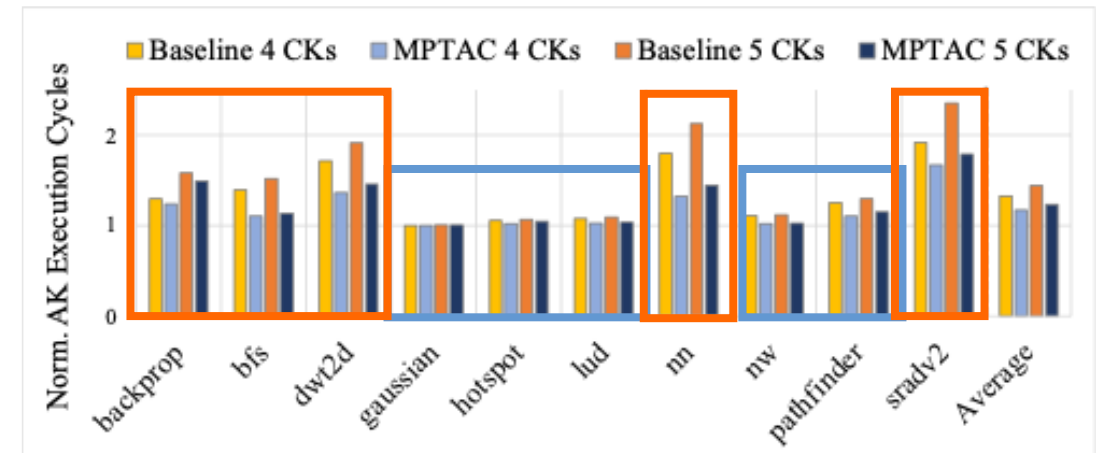
# Benchmark Characteristics

- **Low contention sensitivity:** Small contention on AK, no MPTAC overhead
  - Low memory intensity: *gaussian*, *hotspot*, *lud*, and *pathfinder*
  - nw relies more on memory but it has latency saturated profile (low IPC in baseline)
- **High contention sensitivity:** rely heavily on memory (L2 and DRAM)
  - backprop, bfs, dwt2d, nn, and sradv2
  - dwt2d and sradv2 have the highest sensitivity
  - Execution time sensitivity correlates more with performance in isolation

## Rodinia benchmark characteristics



| Benchmark     | Mem. Intensity | Iso. IPC     |
|---------------|----------------|--------------|
| backprop      | High           | 26.5         |
| bfs           | High           | 5.3          |
| <b>dwt2d</b>  | <b>Medium</b>  | <b>39.07</b> |
| gaussian      | Low            | 0.93         |
| hotspot       | Low            | 39.78        |
| nn            | High           | 6.57         |
| lud           | Low            | 21.53        |
| <b>nw</b>     | <b>Medium</b>  | <b>2.09</b>  |
| pathfinder    | Low            | 39.07        |
| <b>sradv2</b> | <b>Medium</b>  | <b>26.09</b> |



Normalized AK execution cycles relative to isolation

# Conclusions

- Modern applications sharing an embedded GPU partition experience contention
- Detected shared buffers as the main bottleneck that causes contention in the state-of-the-art partitioning approaches such as NVIDIA MIG
- We proposed MPTAC, a low-overhead memory controller partitioning approach that manages others through a software-configurable occupancy threshold.
- We reduce analysis kernel slowdown from  $3.83\times$  to  $1.43\times$  in most aggressive cases and from  $2.35\times$  to  $1.78\times$  in representative benchmarks (47.74% average)
- MPTAC offers flexible ETD for analysis kernel with minimal hardware overhead



**Thank you for listening!**

37th IEEE International Conference on Application-specific Systems, Architectures and Processors

# Resources

- [1] Indian Institute of Embedded Systems (IIES), "Automotive Embedded Systems Course," IIES, [Online]. Available: <https://iies.in/crash-courses/automotive-embedded-system/>. [Accessed: Aug. 17, 2026].
- [2] NVIDIA, "NVIDIA Multi-Instance GPU User Guide," 2024, [Online]. Available: [https://docs.nvidia.com/datacenter/tesla/pdf/NVIDIA\\_MIG\\_User\\_Guide.pdf](https://docs.nvidia.com/datacenter/tesla/pdf/NVIDIA_MIG_User_Guide.pdf). [Accessed: Apr. 17, 2026].
- [3] NVIDIA, "Time-slicing gpus in kubernetes," 2022, accessed: 2025-04-1. [Online]. Available: <https://docs.nvidia.com/datacenter/cloud-native/gpu-operator/latest/gpu-sharing.html>
- [4] NVIDIA, "Multi-process service," 2025, accessed: 2025-04-1. [Online]. Available: <https://docs.nvidia.com/deploy/mps/index.html>
- [5] Y. Zhang, H. Yu, C. Han, C. Wang, B. Lu, Y. Li, X. Chu, and H. Li, "Missile: Fine-grained, hardware-level GPU resource isolation for multi-tenant DNN inference," CoRR, vol. abs/2407.13996, 2024. [Online]. Available: <https://doi.org/10.48550/arXiv.2407.13996>