KU-CSL
COMPUTER SYSTEMS LAB

Per-Bank Memory Bandwidth Regulation for Predictable and Performant Real-Time Systems

Connor Sullivan, Amin Mamandipoor, Cole Strickler, Heechul Yun
University of Kansas

Memory Bandwidth Regulation

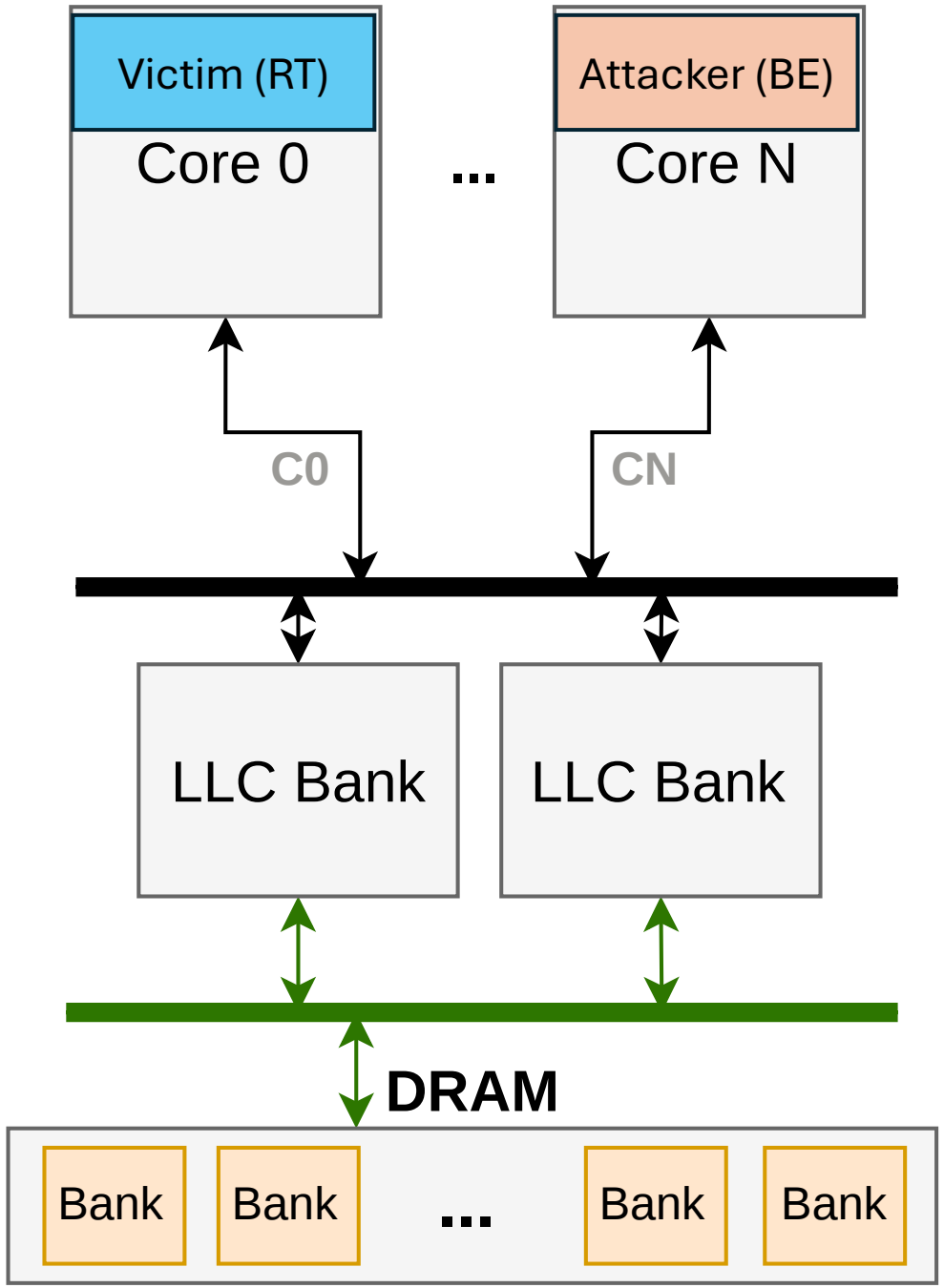
Mechanisms to control the rate at which the client can access memory

Academia	Industry
MemGuard	Intel RDT MBA
MemPol	ARM MPAM
MemCoRe	RISC-V CBQRI

One underlying assumption: Higher bandwidth = Higher interference

Contention Setup

- Mix of real-time (RT) and best-effort (BE) tasks
- Assume RT is victim *bandwidth* workload
- Assume BE workloads are malicious memory contention attackers
- **Expectation: Higher attacker bandwidth = slower victim execution**



Unexpected Result

Run on Raspberry Pi 5

High bandwidth != Highest interference

Why is this the case?

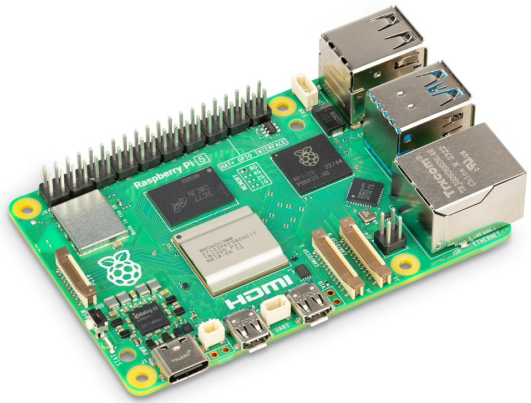
Understanding the Problem

Example Platforms

Raspberry Pi 4

8 DRAM Banks
12.8 GB/s Peak Bandwidth

Raspberry Pi 5

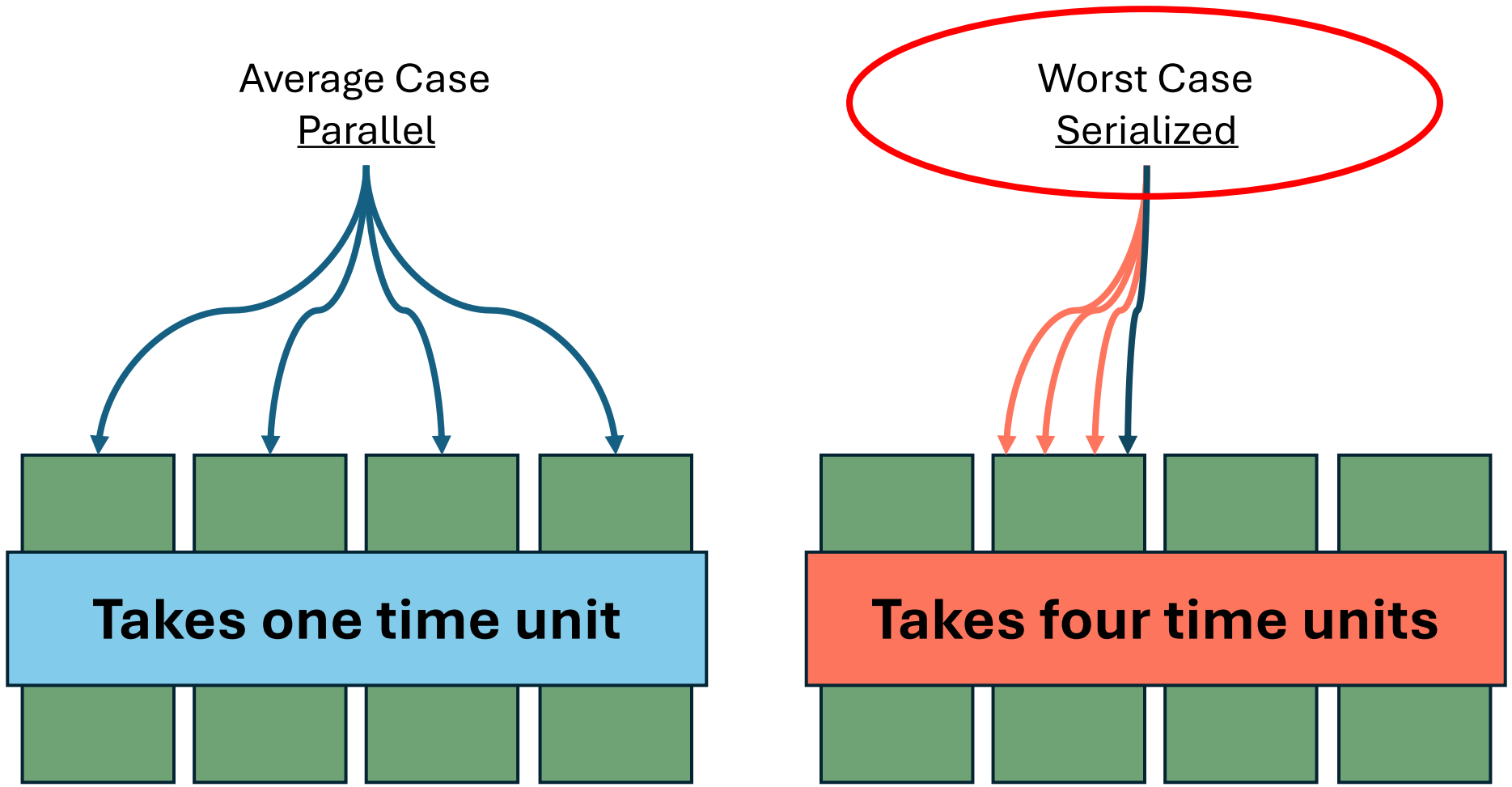


16 DRAM Banks
17.1 GB/s Peak Bandwidth

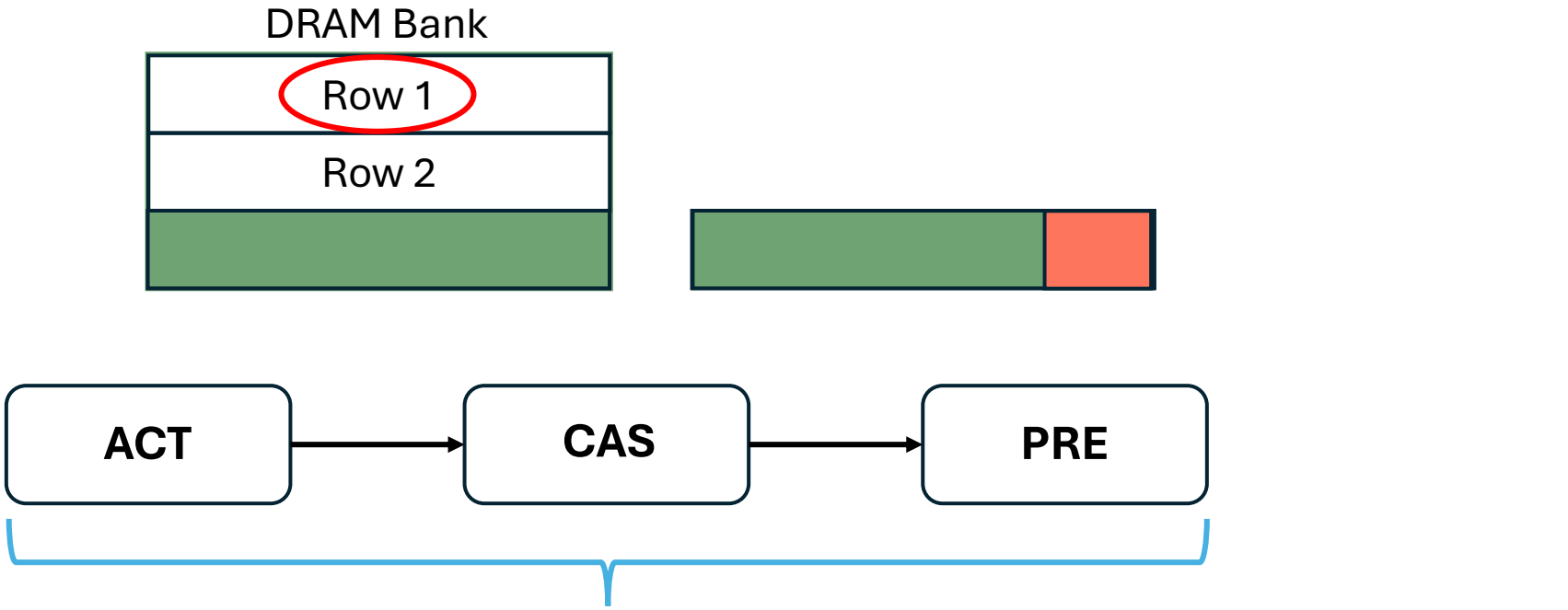
Jetson Orin AGX

256 DRAM Banks
204.8 GB/s Peak Bandwidth

DRAM Bank Parallelism



Guaranteed Bandwidth



tRC (Row Cycle Time) ≈ 45-60 ns

Physical Constraint: Hasn't changed much since DDR3

$$BW_g = 64 \text{ bytes} / tRC$$

Measuring Guaranteed Bandwidth

- Reverse engineer bank mapping
- Target a single bank with a specialized row-conflict workload
- See paper for details

Platform	Theoretical	Measured
Raspberry Pi 4	1067 MB/s	939 MB/s
Raspberry Pi 5	1067 MB/s	945 MB/s
Intel Coffee Lake	1362 MB/s	1324 MB/s
Jetson Orin AGX	1067 MB/s	1042 MB/s

Core Issue

- DRAM parallel structure is great for average case
- Weak single bank performance is bad for worst case
- **Gap between peak and guaranteed presents a vulnerability**

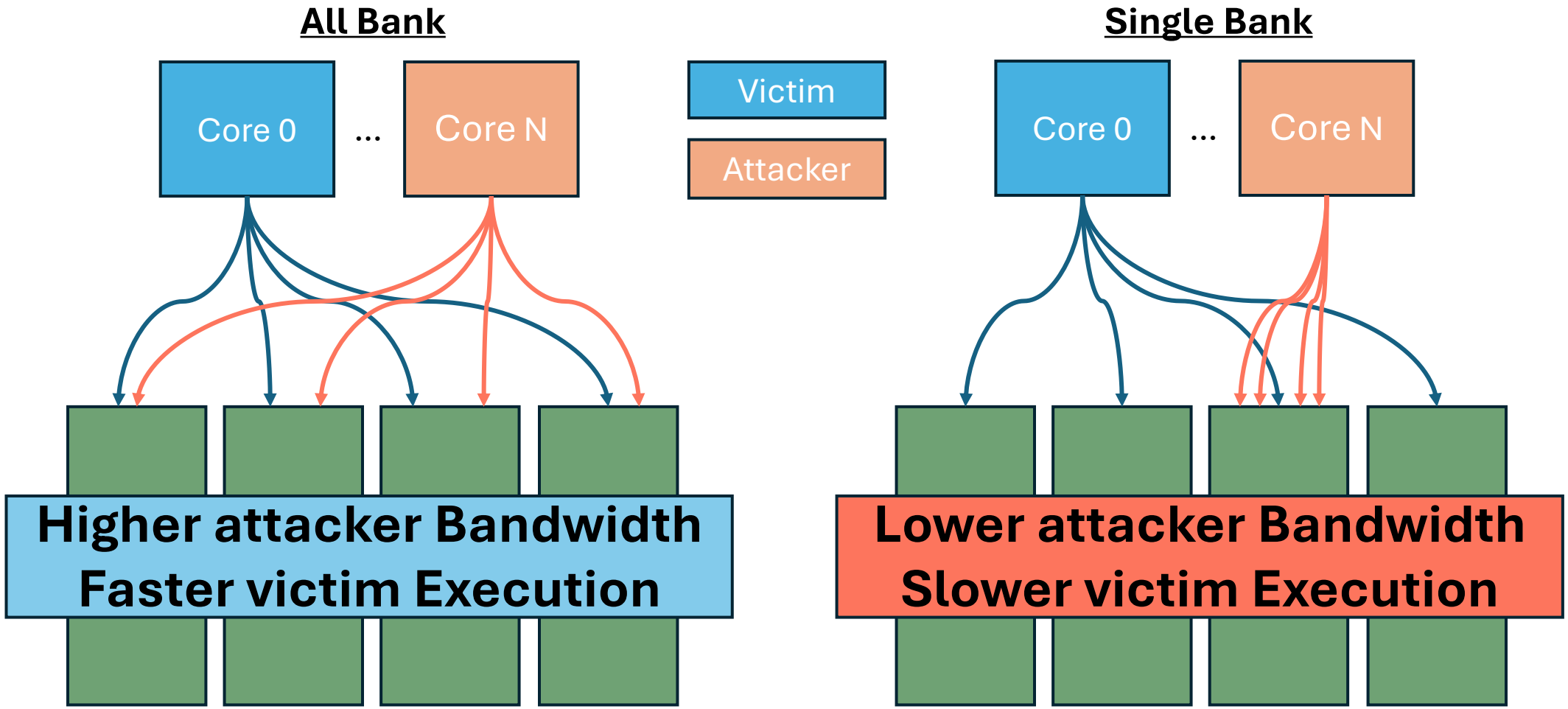
Jetson Orin AGX

204.8 GB/s Peak Bandwidth

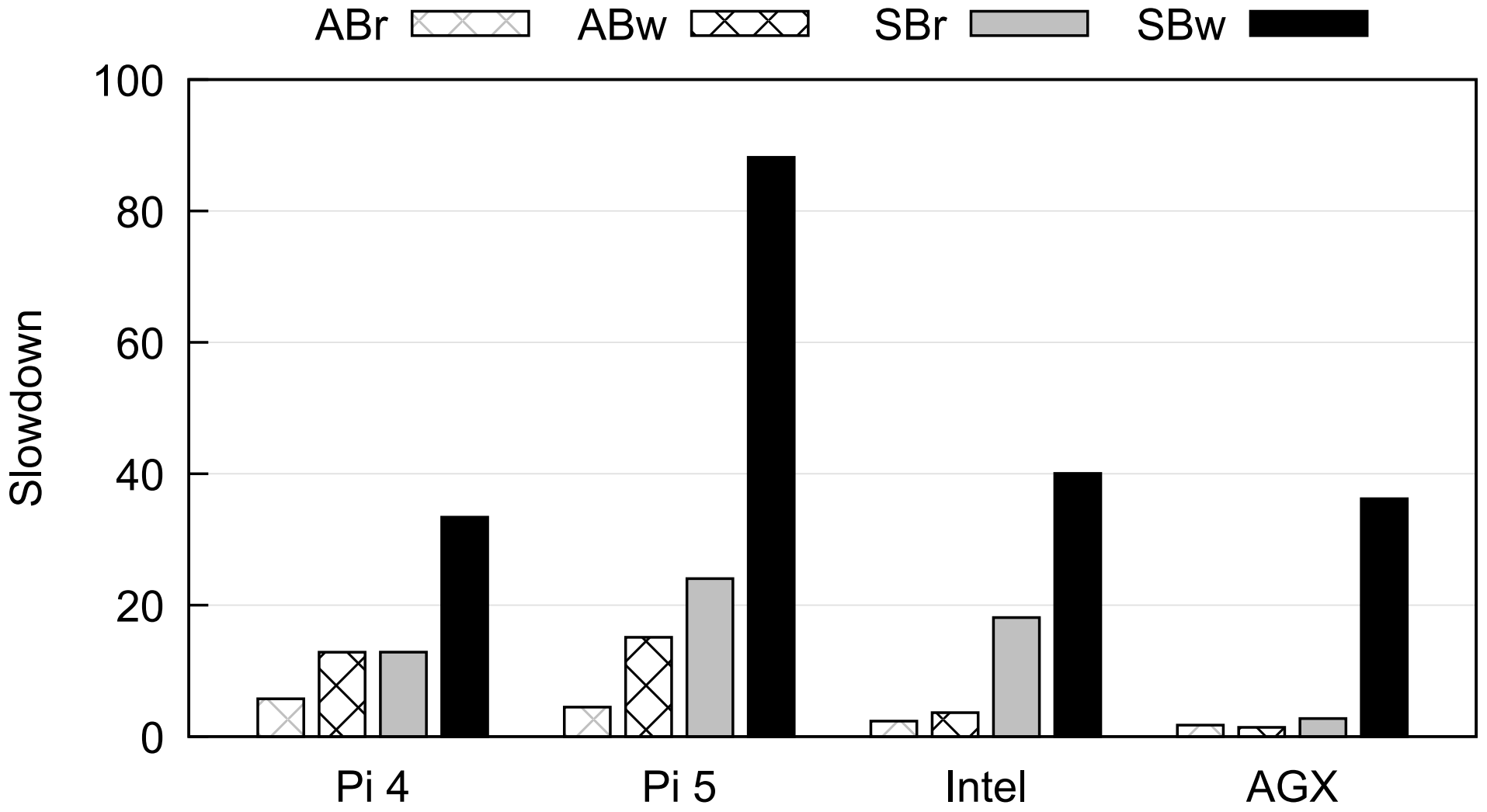
1.067 GB/s Guaranteed

Exploiting the Weakness

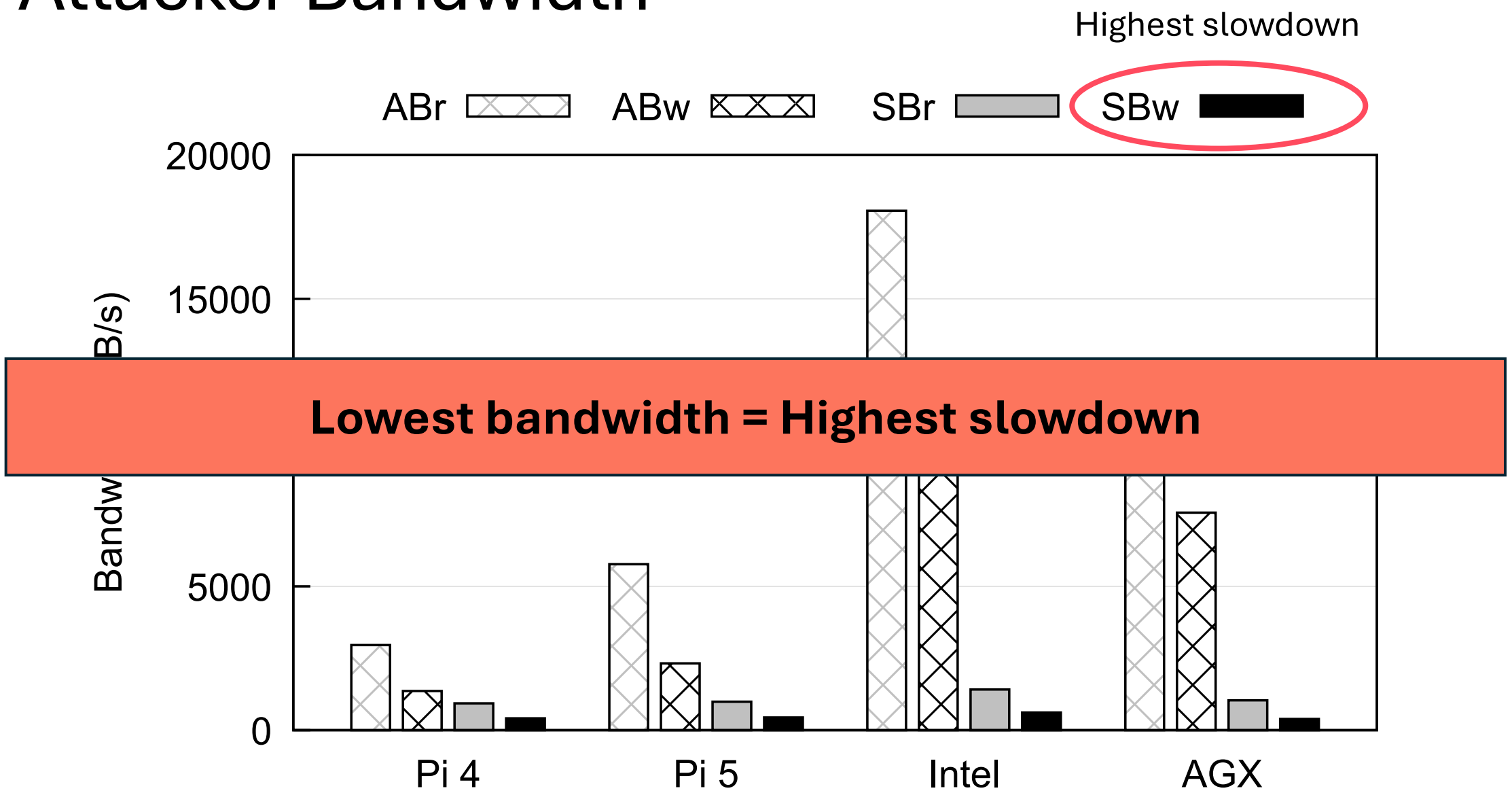
DRAM Contention Attacks



Victim Slowdown



Attacker Bandwidth



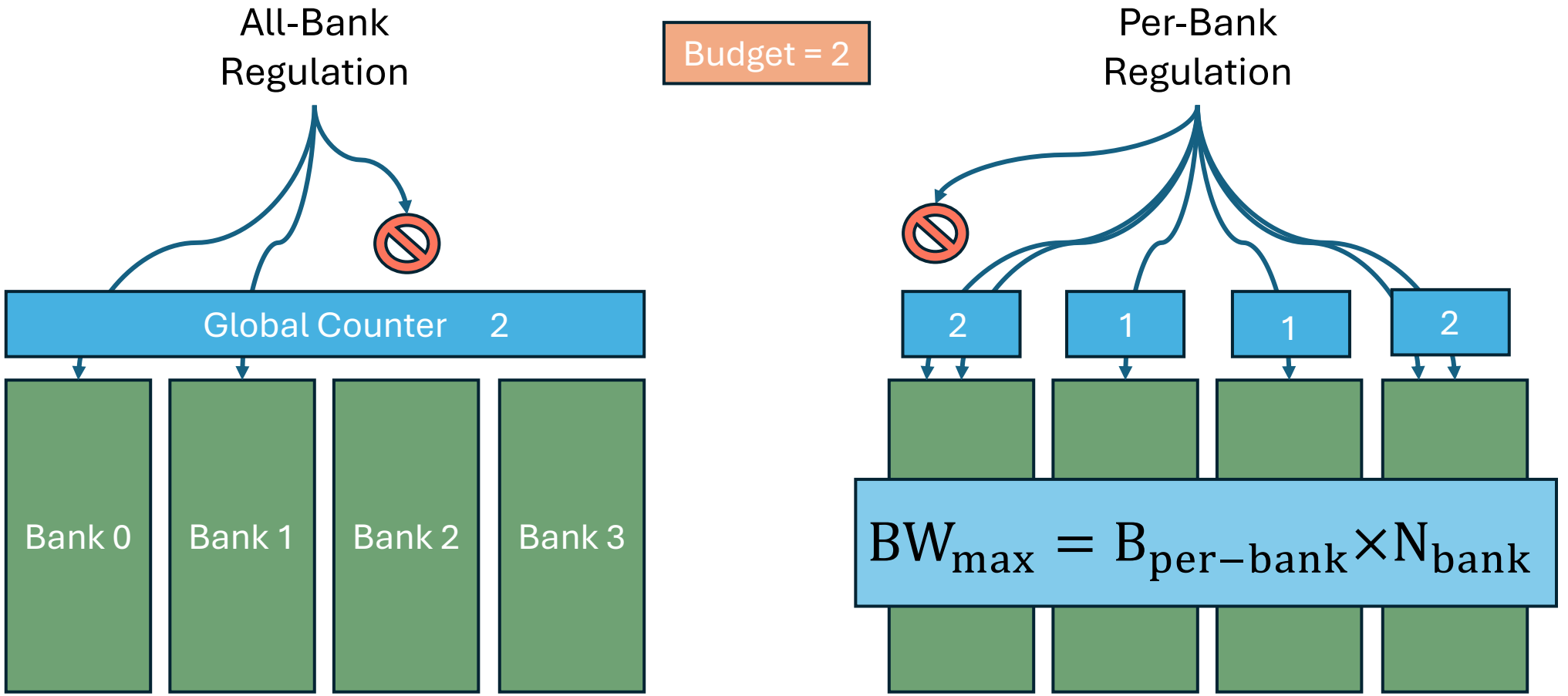
Potential Solutions

Problems of Existing Bandwidth Regulation

- They are “bank-oblivious”
- Choose either system throughput or real-time isolation

Academia	Industry
MemGuard	Intel RDT MBA
MemPol	ARM MPAM
MemCoRe	RISC-V CBQRI

All-Bank vs. Per-Bank Bandwidth Regulation



Why Per-Bank Works

High bandwidth != Highest slowdown

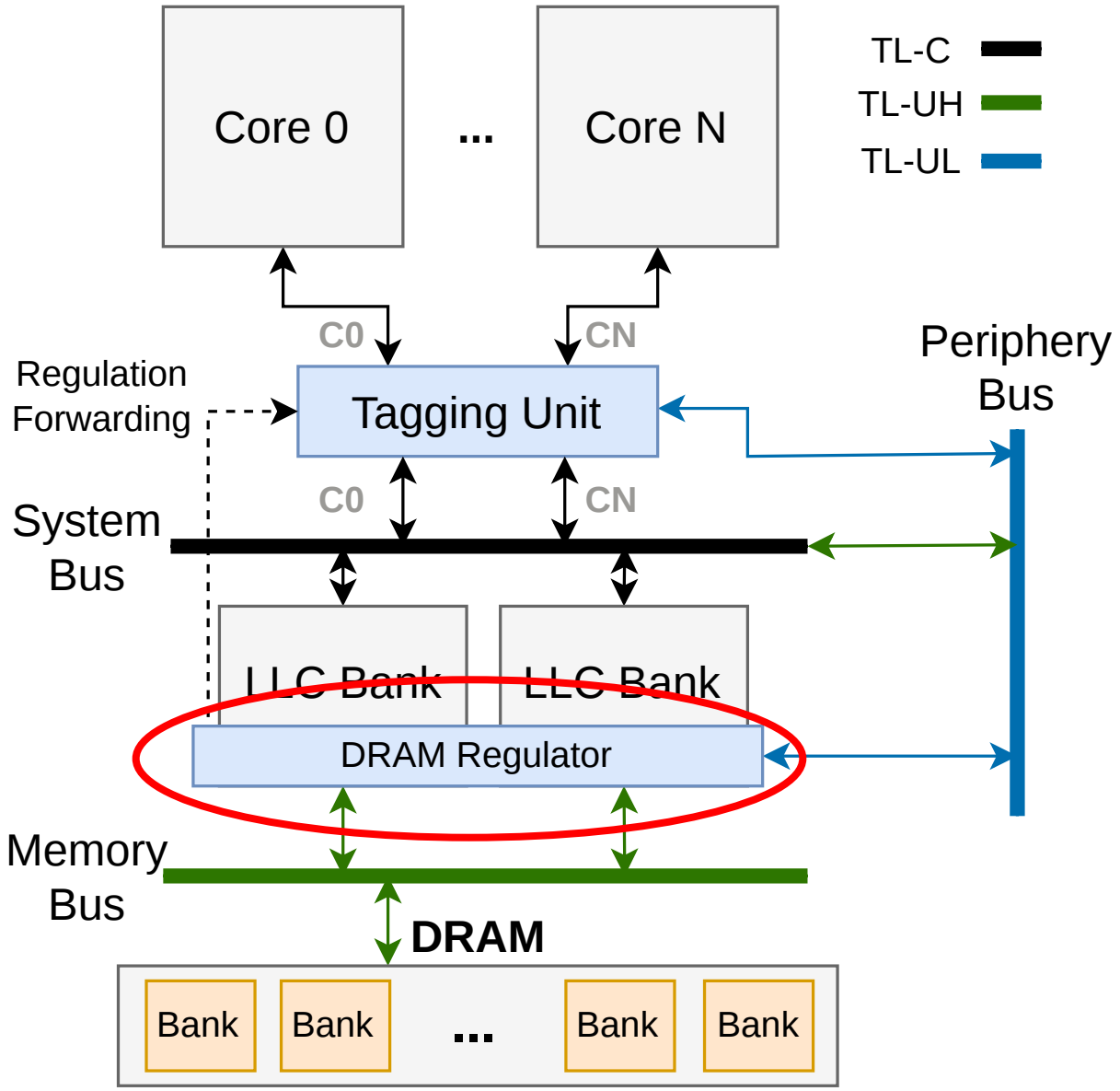
DRAM banks operate in parallel

Allow higher total bandwidth and still control bank contention

Per-Bank DRAM Bandwidth Regulator in RISC-V SoC

Implementation

- Utilize Chipyard SoC framework
- Per-Bank DRAM regulator added to shared last-level cache (LLC)
- **Adds per-bank counters to enable per-bank regulation**
- **Real synthesizable Chisel RTL**

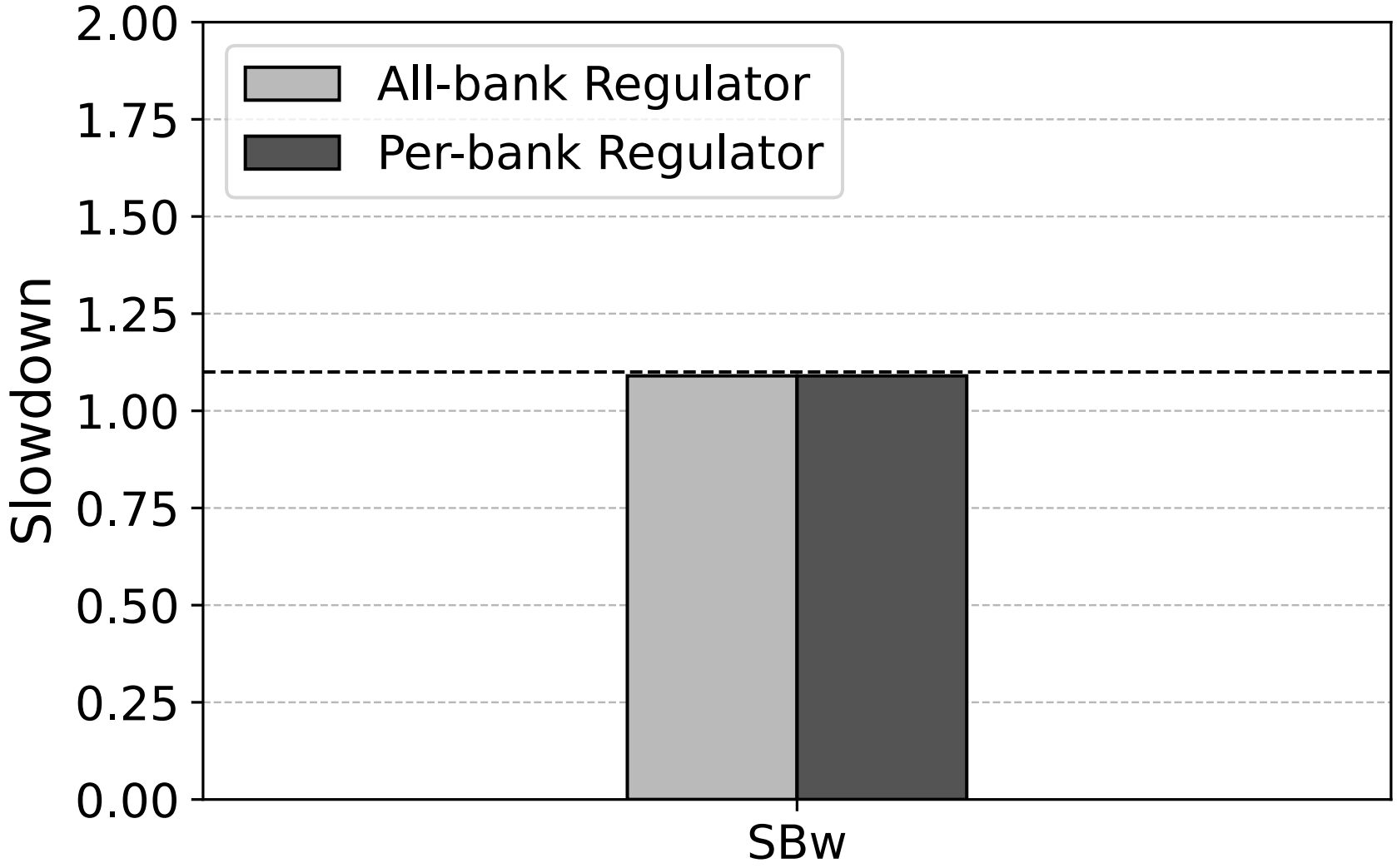


System Details

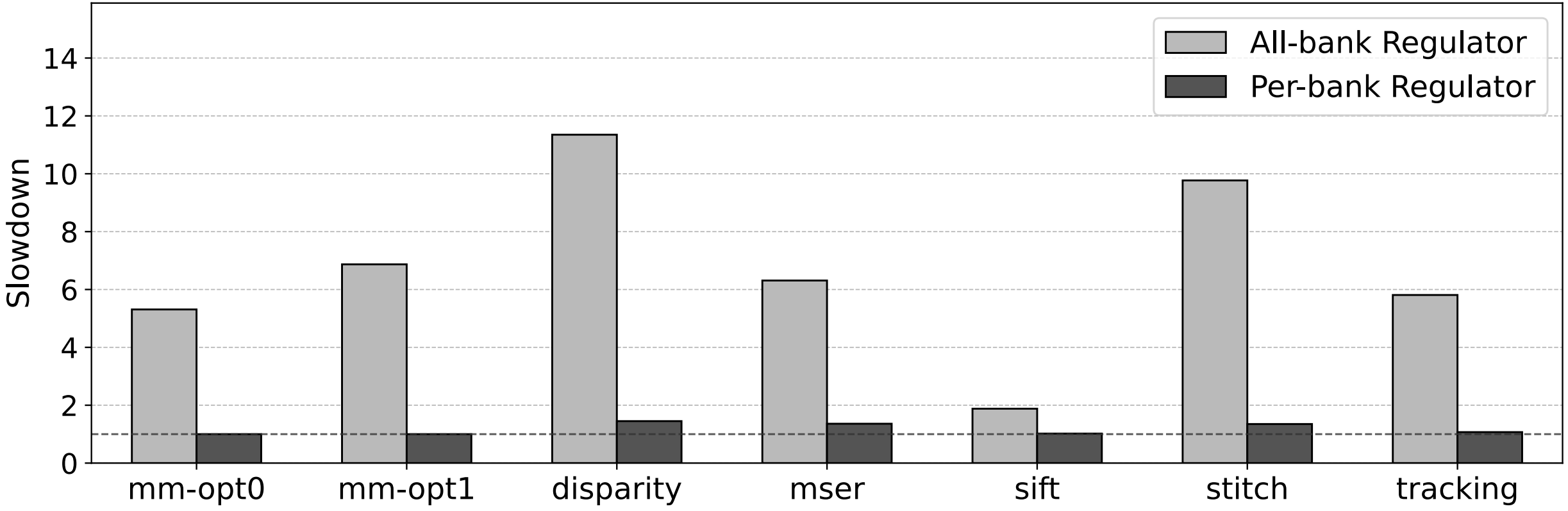
- Quad-core Berkeley Out-of-Order Machine (BOOM)
- Utilize cycle exact, FPGA accelerated simulator: FireSim
- Synthesizable SoC

Cores	4×BOOM, 1GHz, out-of-order, 2-wide, ROB: 64, LSQ: 16/16, L1: 16K(I)/16K(D), 6 mshrs
Shared L2 Cache	1MB (16-way), 2 banks, 27 mshrs per bank, random replacement
DRAM	4GB 1-rank 8-bank DDR3, FR-FCFS, Bank map: 9, 10, 11 (direct map); tRC = 47ns

Real-Time Isolation



Best-Effort Throughput



Hardware Overhead

- Synthesize using Cadence Genus tooling
- Target ASAP 7nm technology node
- Configure for two different DRAM bank configurations
- Minimal area and clock overhead

# Banks	Area	Timing
8	0.35%	3%
16	0.47%	3%

An Efficient Solution

- High bandwidth ≠ High interference
- The worst-case interference occurs when traffic is concentrated on a single bank
- All state-of-the-art bandwidth regulation mechanisms are bank-oblivious
- Per-bank bandwidth regulation offers the same isolation with dramatically better performance
- Feasible to implement in real hardware

This research is supported in part by NSF grants CPS-2038923 and CCF-2403013. Connor Sullivan is supported by the Madison and Lila Self Graduate Fellowship at the University of Kansas.

Questions?