Understanding and Mitigating Hardware Interference Channels on Heterogeneous Multicore

Heechul Yun

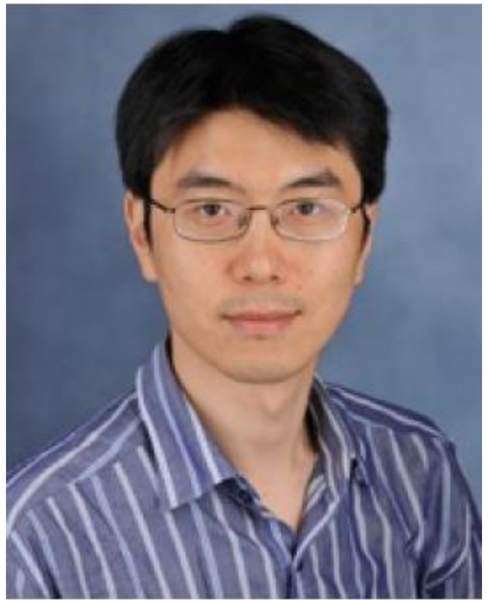
Associate Professor, EECS

University of Kansas

https://www.ittc.ku.edu/~heechul

Heechul Yun

Associate Professor, EECS, University of Kansas



- Education
  - UIUC (PhD, 2013)
  - KAIST (MS, 2001; BS, 1999)
- Employment
  - Assistant/Associate Professor at KU (2013-current)
  - Intern at NVIDIA (2011)
  - Visiting Scholar/Graduate Research Assistant at UIUC (2009-2013)
  - Software Engineer at Samsung Electronics (2004-2009)
  - Researcher at ETRI (2001-2003)
- Research areas
  - Embedded real-time/cyber-physical systems

Agenda

- Understanding hardware interference channels
    - Non-blocking cache
    - Banked cache and DRAM organizations
    - Effective “attack” strategies to cause massive cross-core interference
- AR-HUD automotive case study (ARM industrial challenge)
    - Effects of interference on real-time application performance
    - Limitations of existing mitigation solutions
    - Our solution to mitigate the interference problem
- Discussion and conclusion

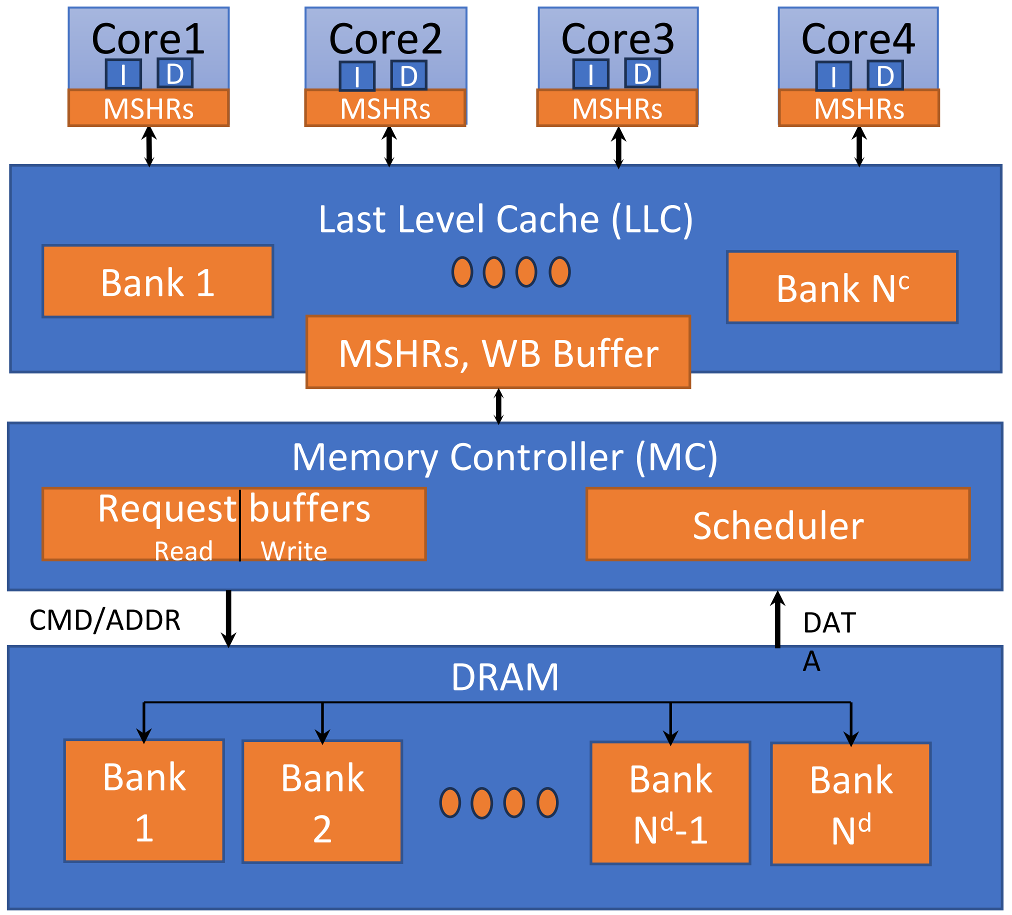
Agenda

- Understanding hardware interference channels
  - Non-blocking cache
  - Banked cache and DRAM organizations
  - Effective “attack” strategies to cause massive cross-core interference
- AR-HUD automotive case study (ARM industrial challenge)
  - Effects of interference on real-time application performance
  - Limitations of existing mitigation solutions
  - Our solution to mitigate the interference problem
- Discussion and conclusion

Memory-level parallelism (MLP)

- MLP is the key to understand modern multicore processors (MCP)
  - essential for performance (throughput)
- A core can request multiple concurrent memory accesses at a time
  - times the number of cores (and accelerators)
- Interconnect (bus) supports split-transactions
  - multiple outstanding transactions can occur simultaneously
- Non-blocking cache can handle multiple outstanding cache misses
  - it can continue to serve hits under multiple misses
- Cache and DRAM are composed of multiple independent resources
  - cache/dram banks can be accessed simultaneously in parallel

Memory-level parallelism (MLP)



- Out-of-order core:
  Multiple memory requests
- Non-blocking caches:
  Multiple cache-misses
- Memory controller:
  Request buffering, re-ordering
- DRAM:
  Multiple banks serve multiple requests

Split-transaction bus

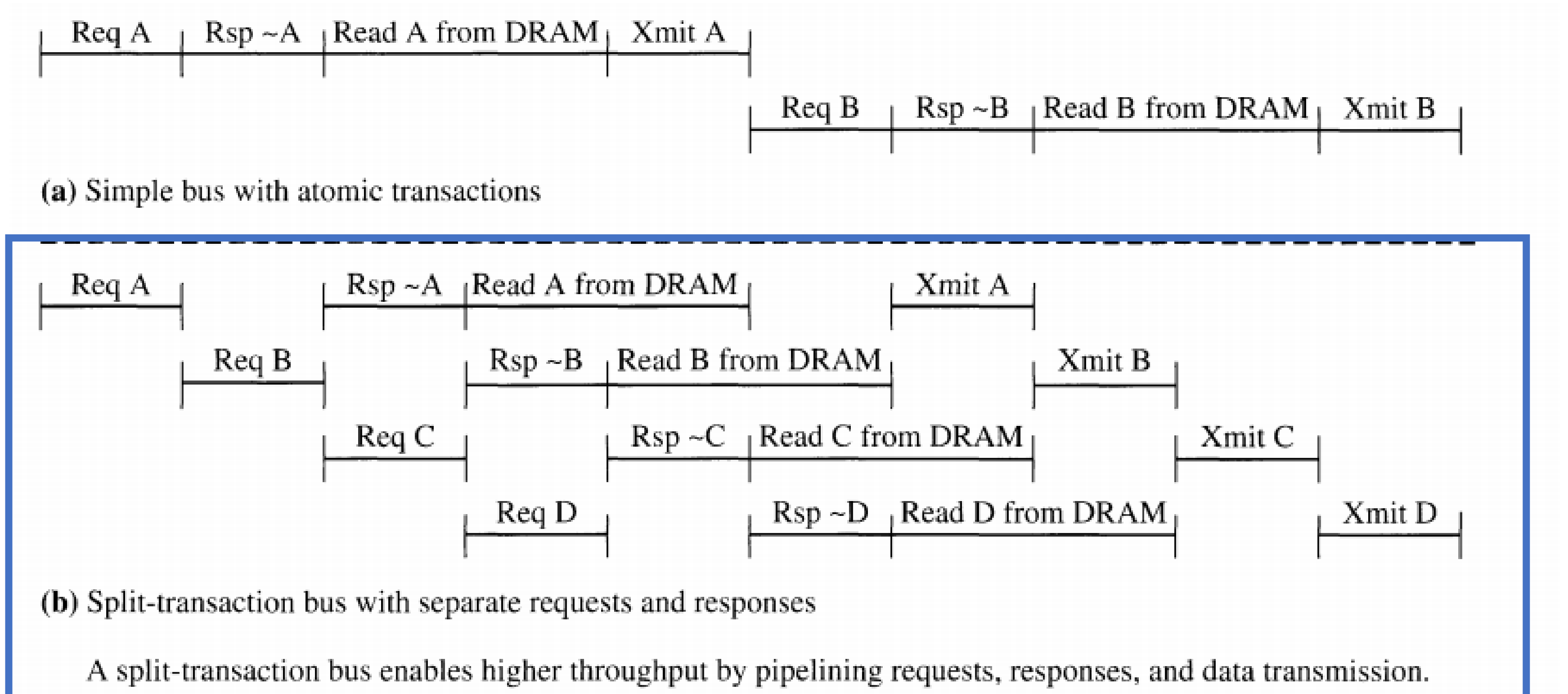

Req A
Rsp ~A
Read A from DRAM
Xmit A
Req B
Rsp ~B
Read B from DRAM
Xmit B
(a) Simple bus with atomic transactions
Req A
Rsp ~A
Read A from DRAM
Xmit A
Req B
Rsp ~B
Read B from DRAM
Xmit B
Req C
Rsp ~C
Read C from DRAM
Xmit C
Req D
Rsp ~D
Read D from DRAM
Xmit D
(b) Split-transaction bus with separate requests and responses
A split-transaction bus enables higher throughput by pipelining requests, responses, and data transmission.

Figure 11.9
Simple Versus Split-Transaction Busses.

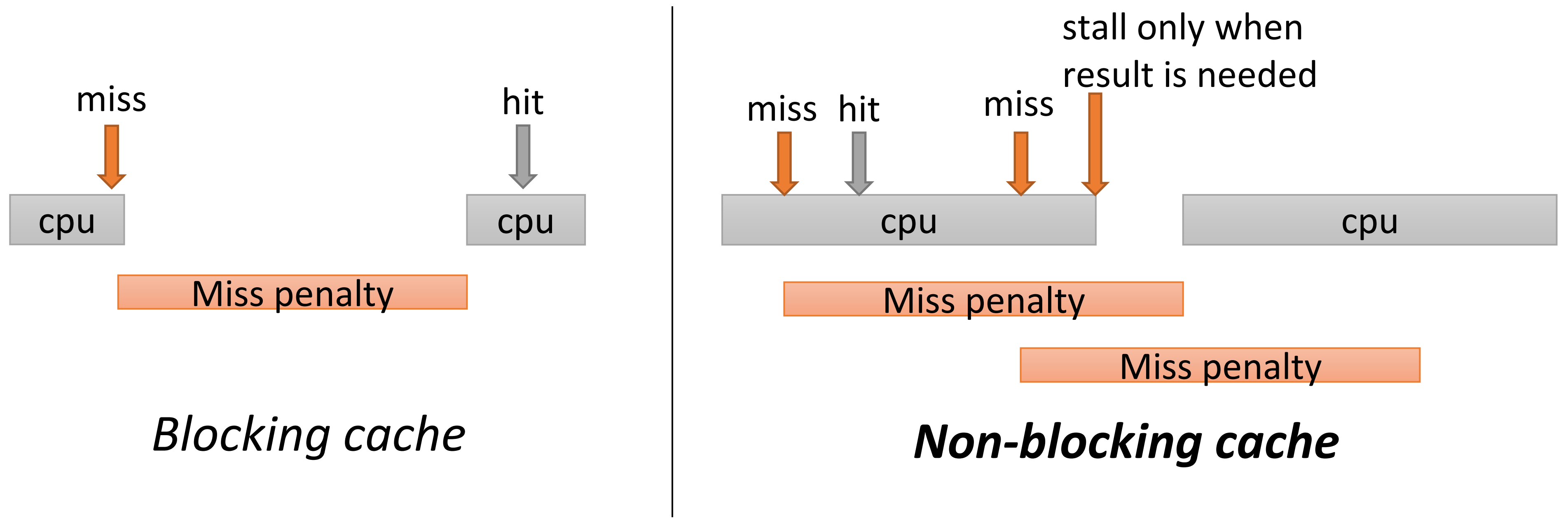
Interconnect is usually not a bottleneck

Figure source: John Paul Shen and Mikko H Lipasti. "Modern processor design: fundamentals of superscalar processors." Waveland Press, 2013

Non-blocking cache

- A core can generate multiple simultaneous accesses to a cache
- Multiple cores/accelerators can simultaneously access a shared cache
- So, a shared cache can get lots of parallel requests
- A non-blocking shared cache is essential for performance

Non-blocking cache



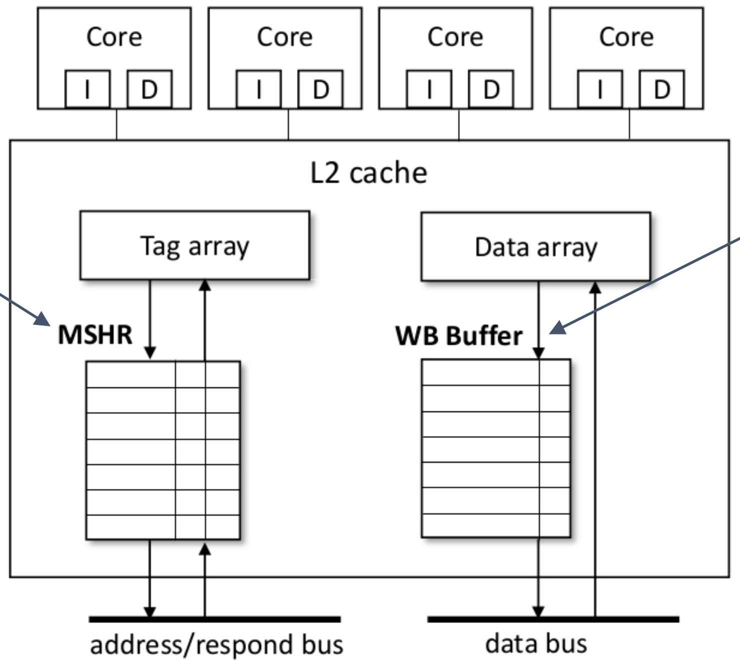
- Can serve cache hits under multiple cache misses
- Essential for performance in multicore

D. Kroft. “Lockup-free instruction fetch/prefetch cache organization,” ISCA’81

Non-blocking cache

Miss Status Holding Registers

- Track outstanding cache misses.
- Allow high memory-level parallelism



Writeback Buffer

- Hold evicted dirty lines (writebacks)
- Prevent cache refills from waiting

- Cache internal structures are **potential interference channels**

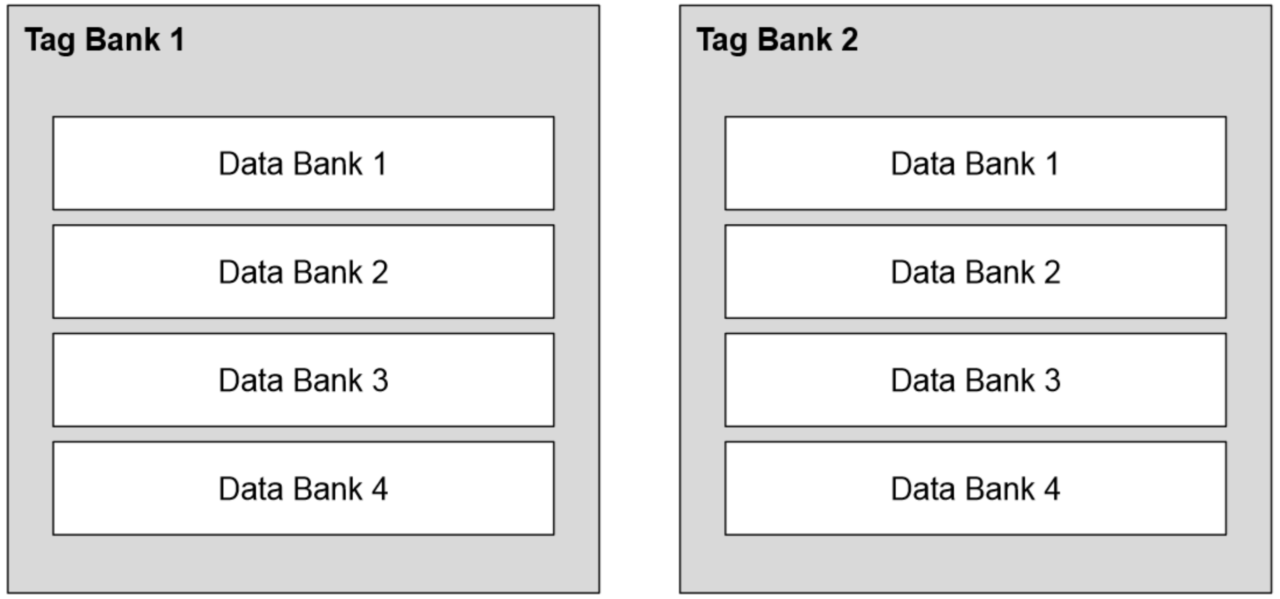
Prathap Kumar Valsan, Heechul Yun, Farzad Farshchi. "Taming Non-blocking Caches to Improve Isolation in Multicore Real-Time Systems." In *IEEE RTAS*, 2016 **(Best Paper Award)**
Michael G. Bechtel and Heechul Yun. "Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention." In *IEEE RTAS*, 2019 **(Outstanding Paper Award)**

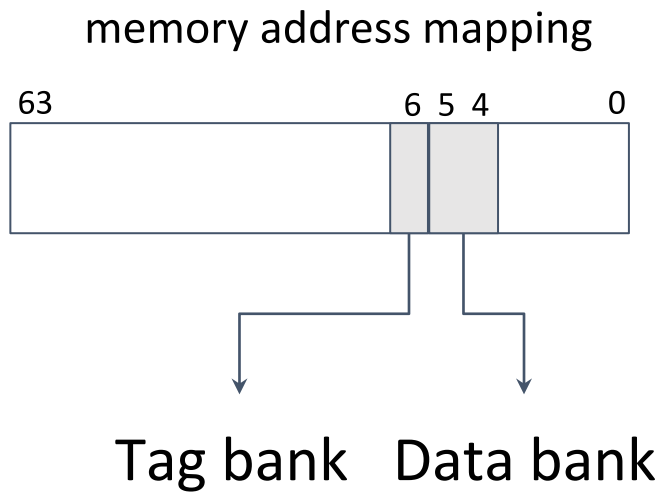
Multi-bank cache/DRAM organizations

- Shared cache and DRAM are not a single resource
- Each is composed of multiple resources---banks
- Banks are (largely) independent and can be accessed in parallel
- Generally, **more banks = more parallelism/throughput**

Cache bank organization

- Multiple banks can be accessed simultaneously

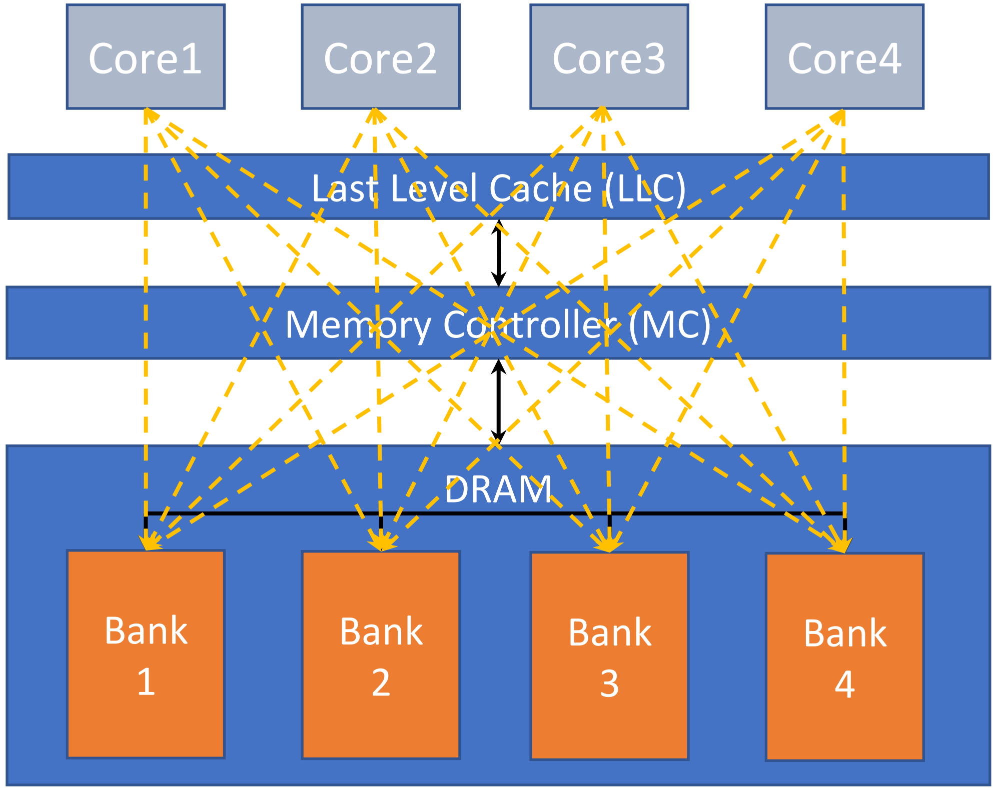


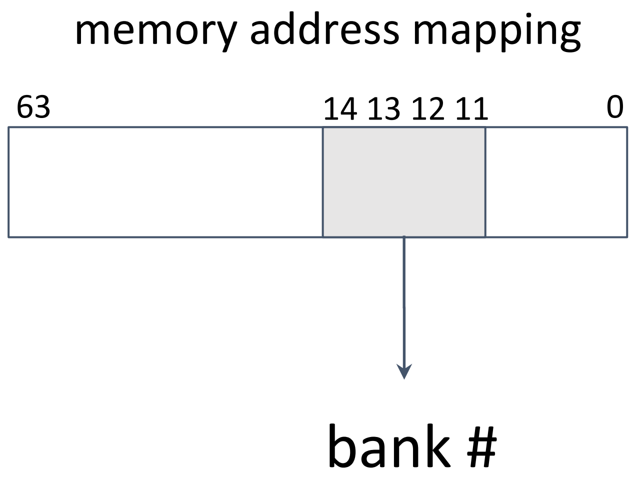


ARM Cortex A72/A57 L2 cache bank organization

DRAM bank organization

- Multiple banks can be accessed simultaneously

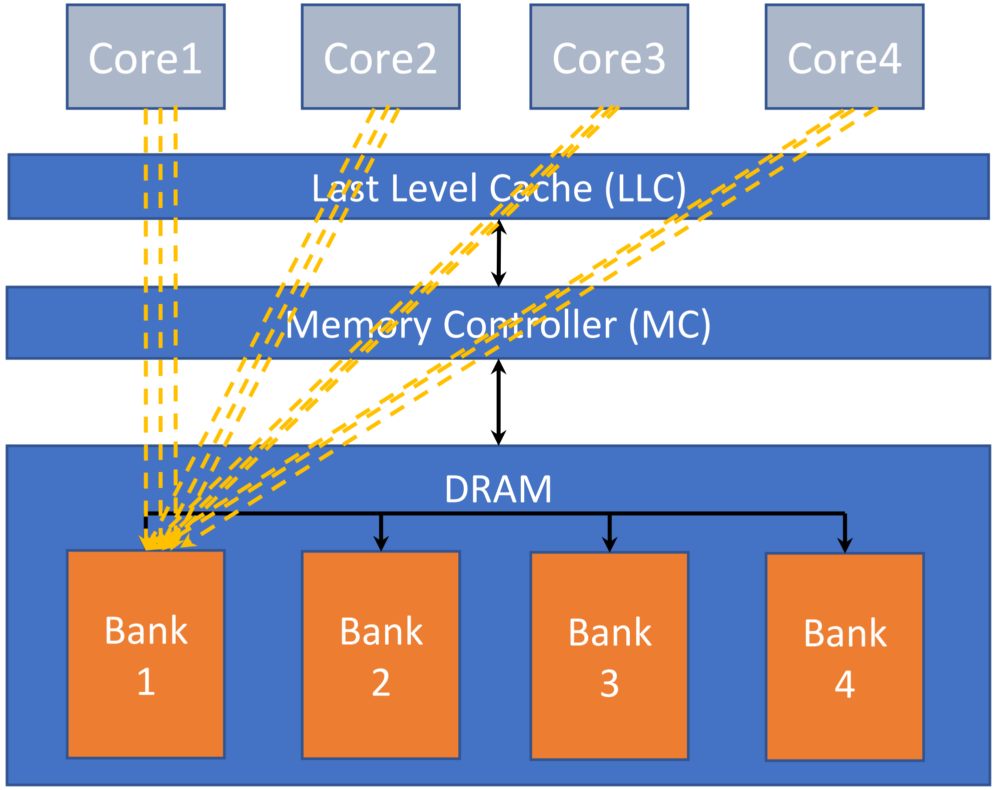


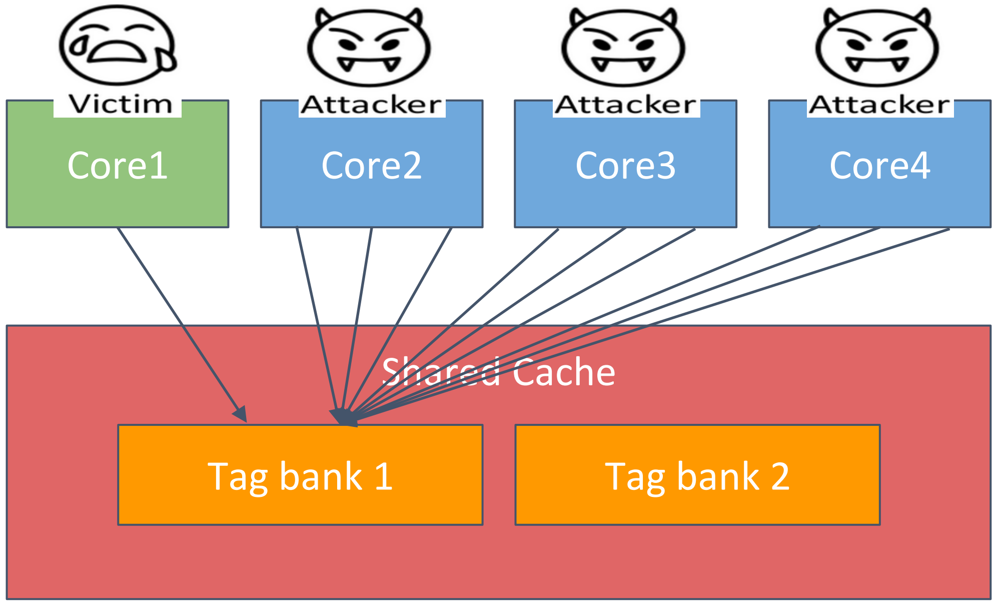


Raspberry Pi 4 DRAM bank mapping (**16 banks**)

Multi-bank cache/DRAM organizations

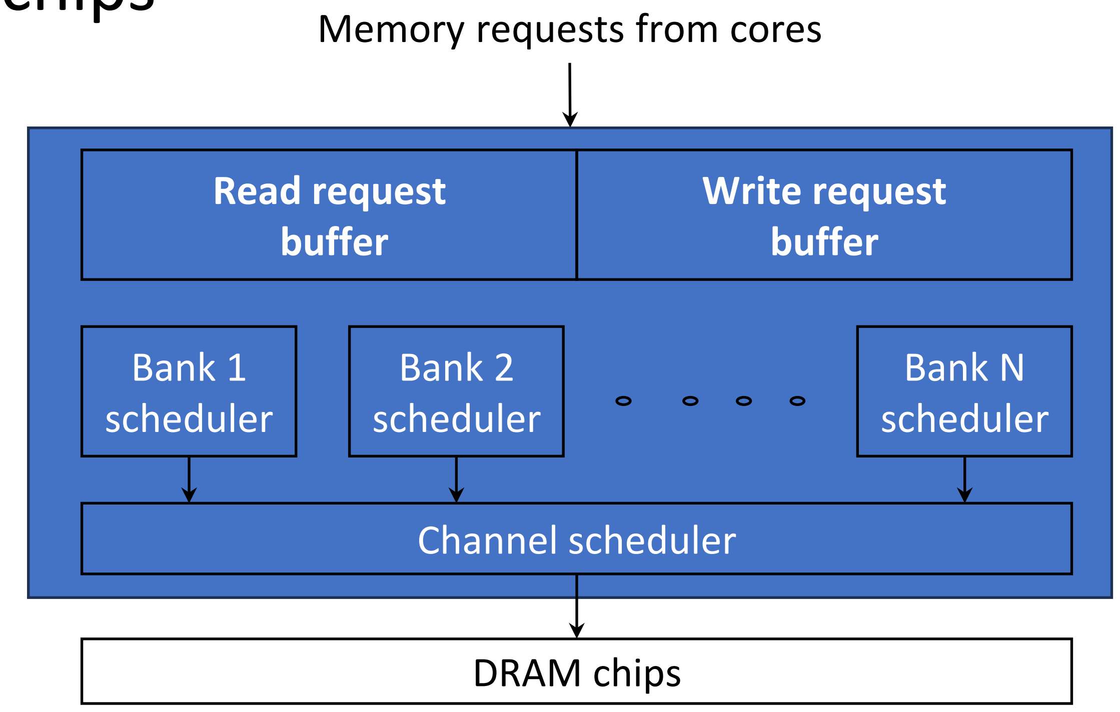
- Can be a problem when all try to access the same cache/dram bank





Memory controller (MC)

- Schedule memory requests on DRAM chips
- Subject to DDR timing constraints
- Can re-order the requests to maximize memory throughput
- Often prioritize reads over writes unless too many writes are pending
- Scheduling algorithms can greatly impact worst-case timing

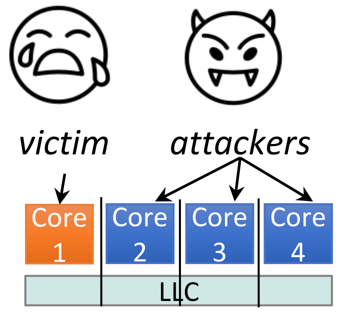


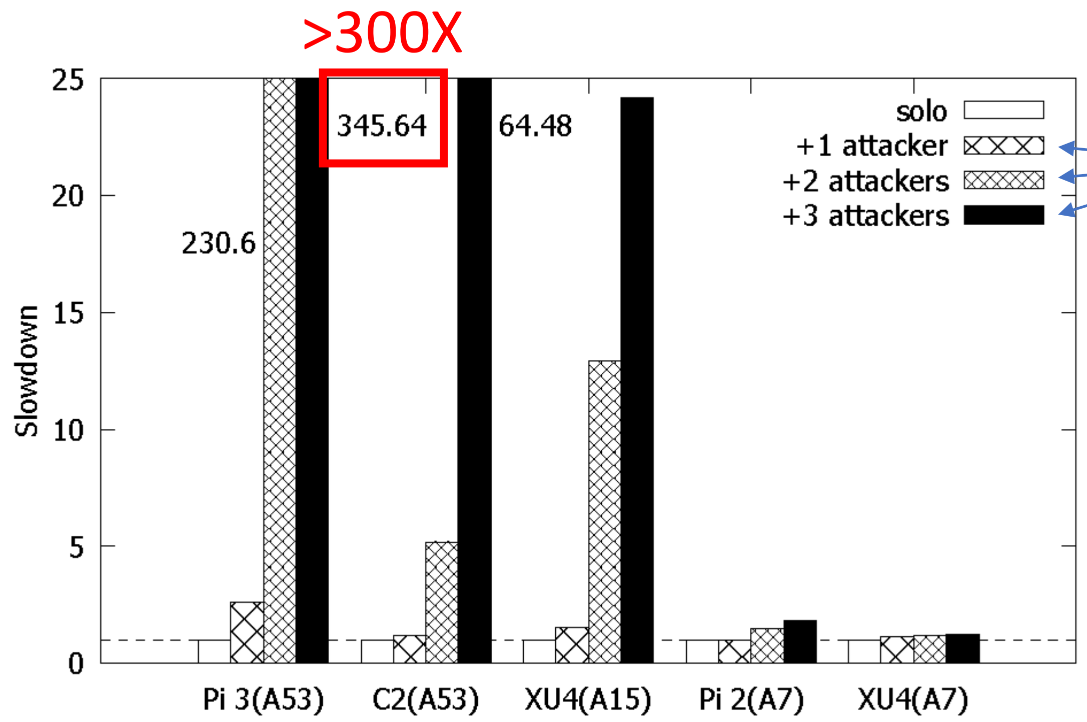
H. Yun, R. Pellizzoni, P. K. Valsan. "Parallelism-Aware Memory Interference Delay Analysis for COTS Multicore Systems." In *ECRTS*, 2015

Effective strategies to cause interference

- Try to exhaust various internal hardware queues/buffers
- Try to generate many requests targeting a single resource (bank)
- Writes often cause worse contention than reads

Effects of cache internal buffer attacks



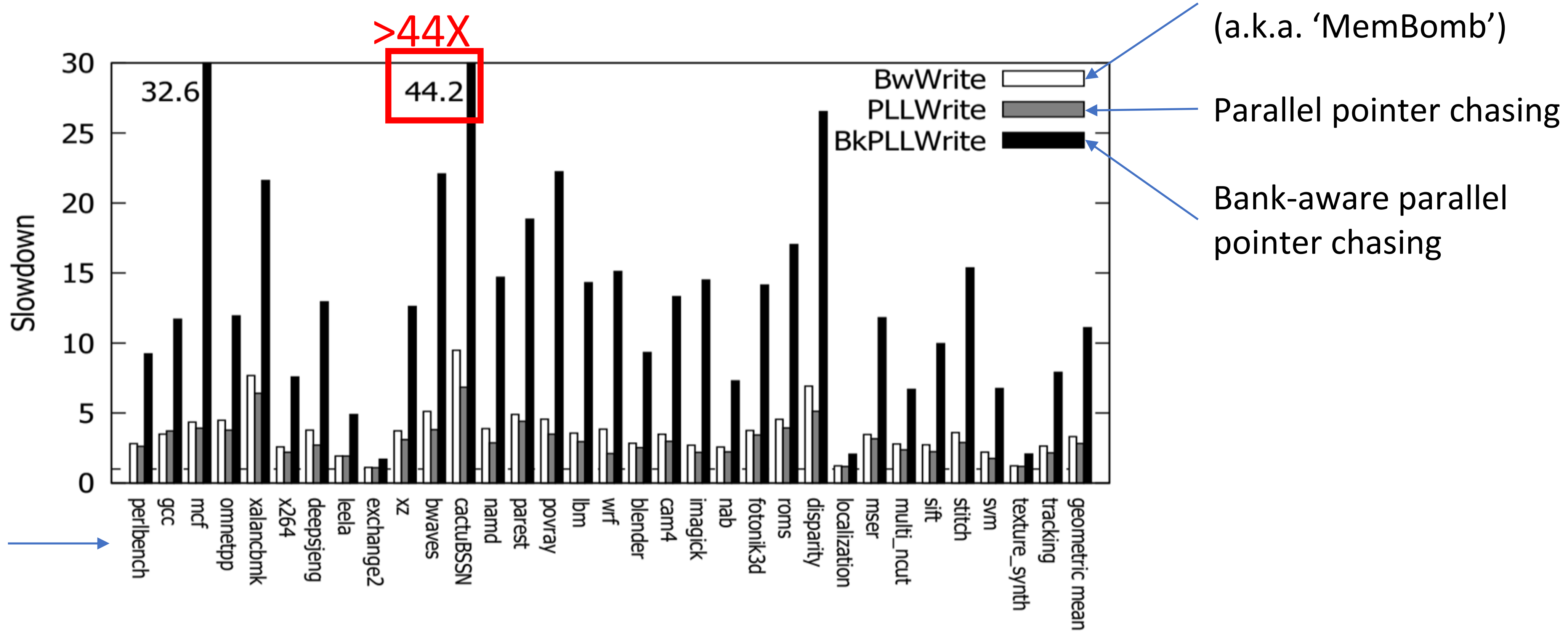


Sequential writes
(a.k.a. 'MemBomb')

- Observed worst-case: >300X (times) slowdown on popular multicores
- Even when the cache is *partitioned* to protect the victim

M. G. Bechtel and H. Yun. "Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention." In *IEEE RTAS*, 2019

Effects of DRAM bank attacks

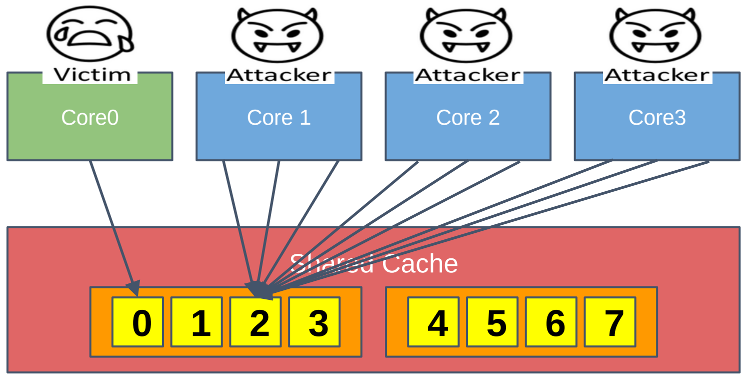


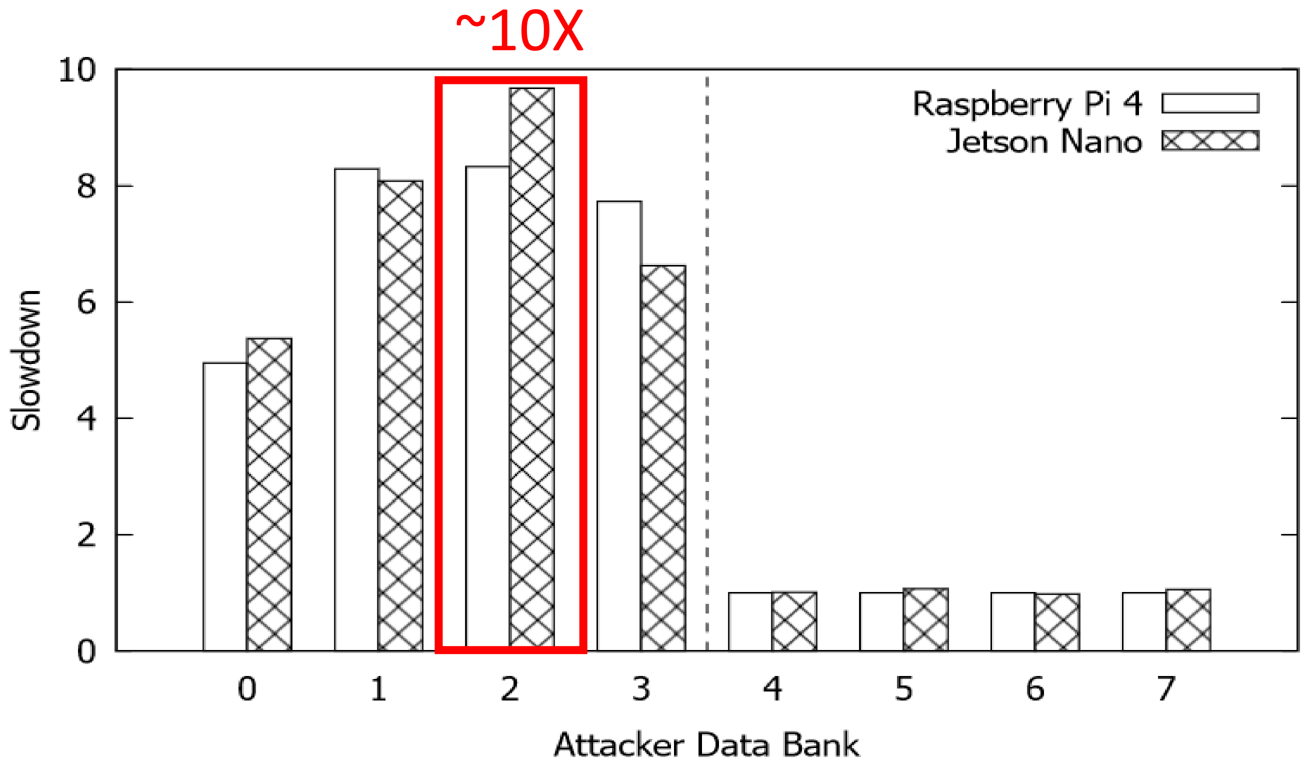
Victim on Core 1

- Targeting a single DRAM bank caused up to 44X slowdown in real apps
- LLC space partitioning was not effective

M. G. Bechtel and H. Yun. "Memory-Aware Denial-of-Service Attacks on Shared Cache in Multicore Real-Time Systems." In *IEEE Transactions on Computers*, 2021

Effects of cache bank attacks

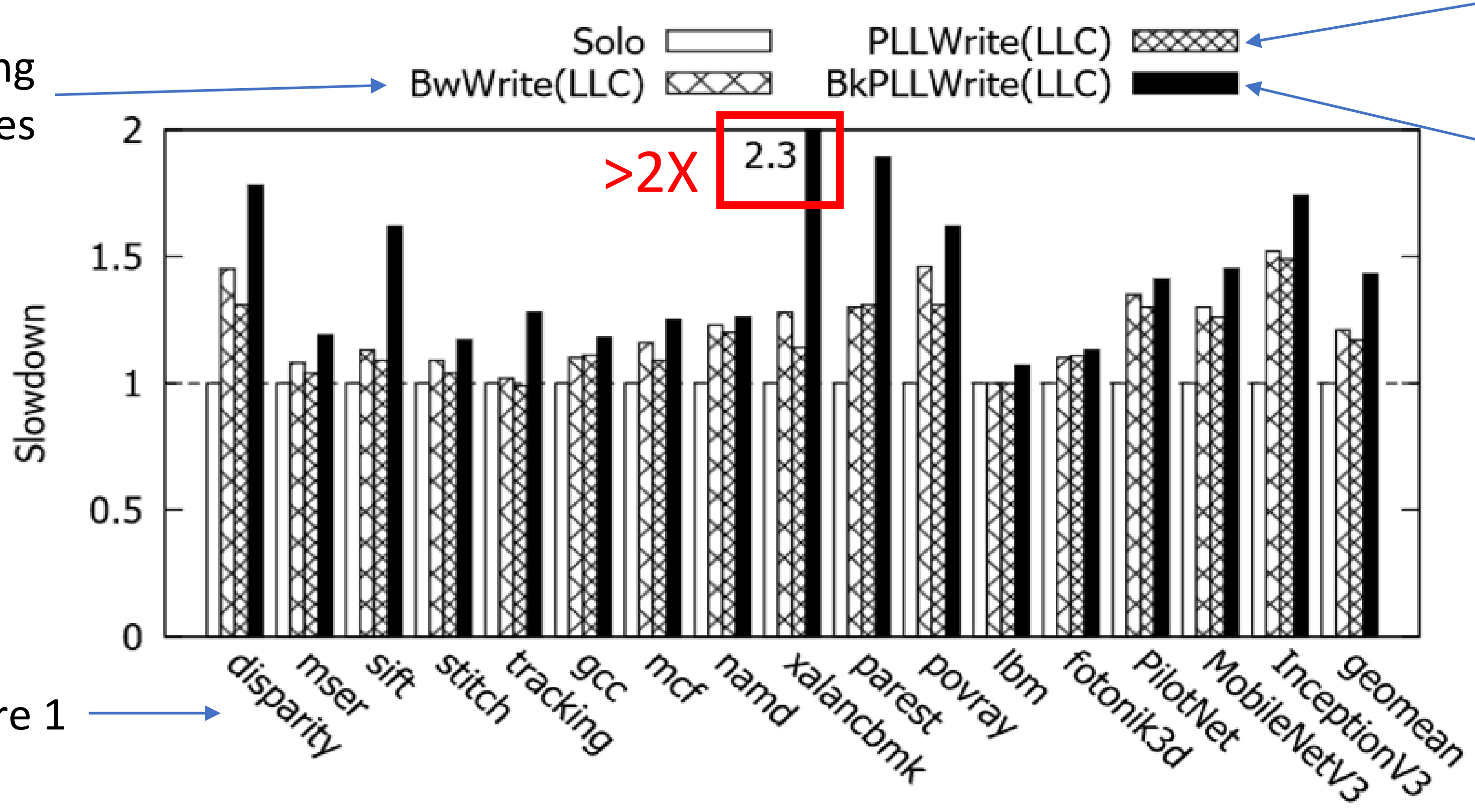




- Accessing the same tag bank (and diff. data bank) → up to 10X slowdown
- Accessing different tag bank → near perfect isolation

M. G. Bechtel and H. Yun. "Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors." In *IEEE RTAS,* 2023

Effects of cache bank attacks



LLC fitting sequential writes

LLC fitting parallel pointer chasing

Bank-aware LLC fitting parallel pointer chasing

Victim on Core 1

- Targeting a single cache bank caused up to 2.3X slowdown in real apps
- LLC space partitioning and DRAM bandwidth throttling were not effective

M. G. Bechtel and H. Yun. “Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors.” In *IEEE RTAS,* 2023

Summary

- Memory-level parallelism (MLP) is key to understand modern multicore processors (MCPs)
- High MLP designs at all levels of the memory hierarchy are essential for performance/throughput, but they also can be problematic **hardware interference channels** from a real-time perspective
- Contrary to popular beliefs, interconnects are usually not major interference channels in modern MCPs. Major ones are at the edges
- There are effective “attack” strategies to cause massive cross-core interference, which cannot be easily mitigated by existing software/hardware partitioning techniques

Agenda

- Understanding hardware interference channels
  - Non-blocking cache
  - Banked cache and DRAM organizations
  - Effective “attack” strategies to cause massive cross-core interference
- AR-HUD automotive case study (ARM industrial challenge)
  - Effects of interference on real-time application performance
  - Limitations of existing mitigation solutions
  - Our solution to mitigate the interference problem
- Discussion and conclusion

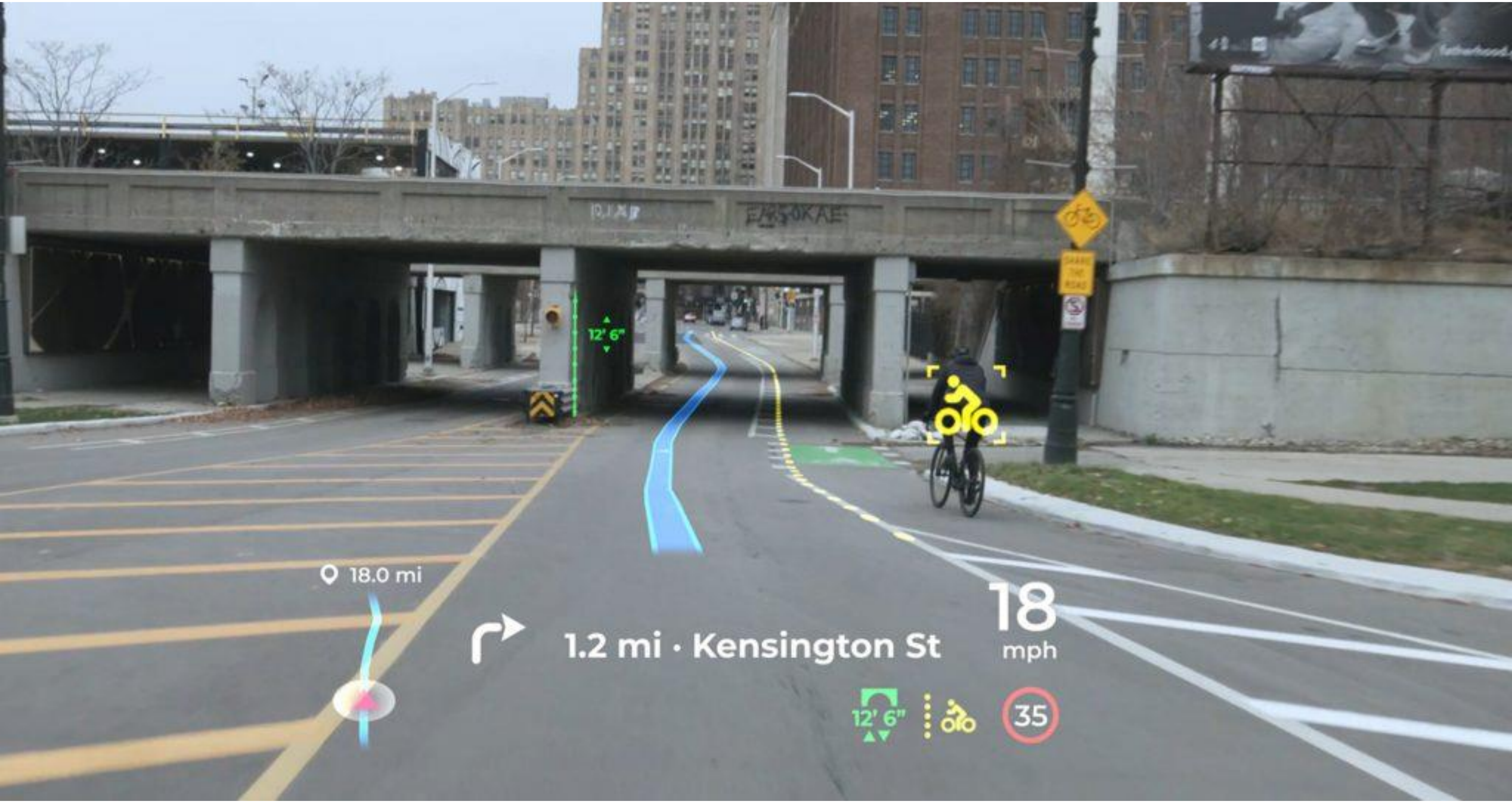


Image Source: https://www.ecrts.org/industrial-challenge-current-challenge/

Augmented reality head-up display (AR-HUD)

- ARM 2022 industrial challenge case study application
- Visual SLAM (OV^{2}SLAM)
  - Determine orientation and trajectory + generate a map of the surroundings
  - High-criticality real-time task
- Head-pose estimation DNN (Hope-Net)
  - Estimate driver’s pose for better AR rendering that accounts for the driver’s viewpoint
  - high-priority real-time task
- “Aggressor” tasks
  - Other (synthetic) tasks that compete for the shared hardware resources of the SoC
  - Best-effort (non real-time) priority

M. Andreozzi, G. Gabrielli, B. Venu, G. Travaglini. “Industrial Challenge 2022: A High-Performance Real-Time Case Study on Arm.” In *ECRTS,* 2022

Analysis and Mitigation of Shared Resource Contention on Heterogeneous Multicore: An Industrial Case Study

Michael Bechtel
University of Kansas, USA

Heechul Yun
University of Kansas, USA

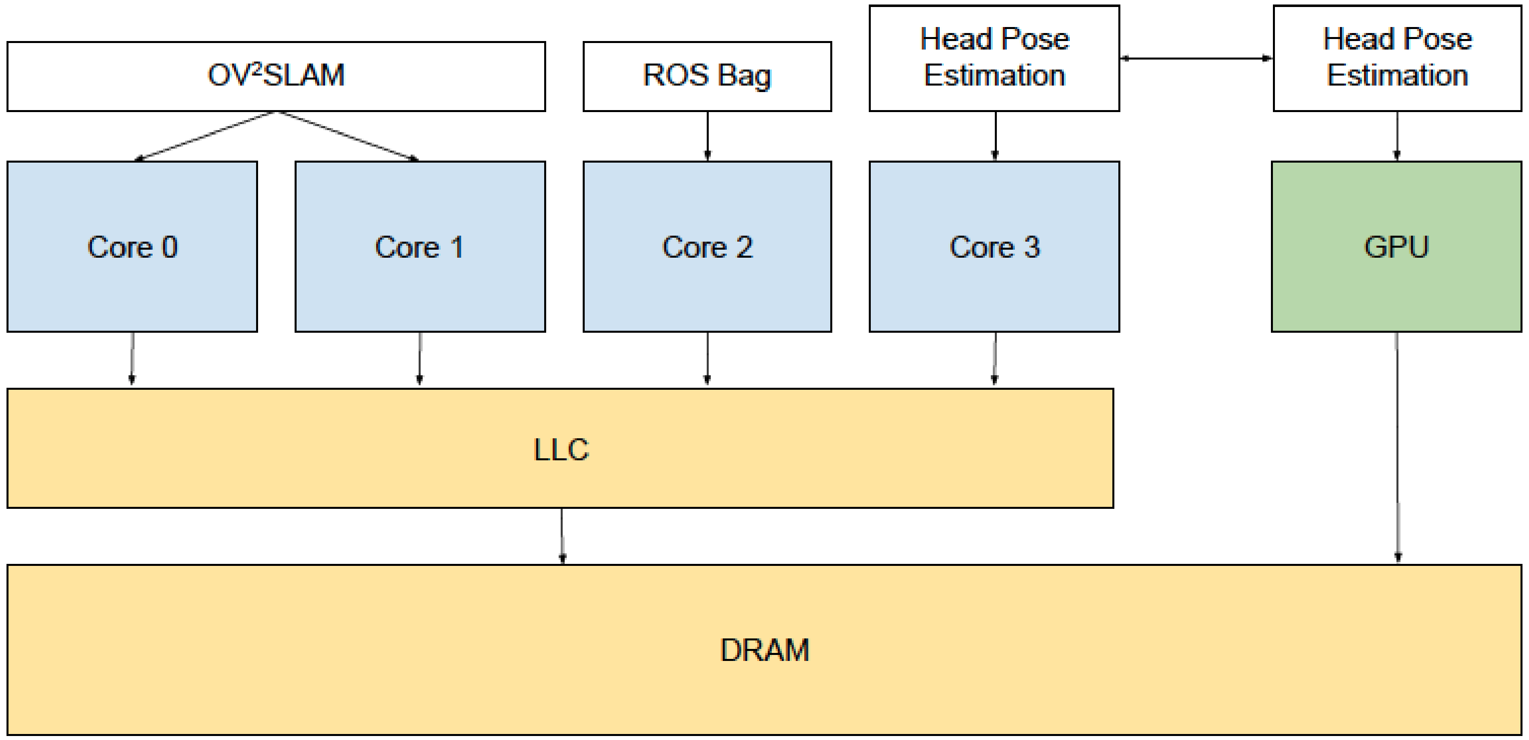
Abstract

In this paper, we address the industrial challenge put forth by ARM in ECRTS 2022. We systematically analyze the effect of shared resource contention to an augmented reality head-up display (AR-HUD) case-study application of the industrial challenge on a heterogeneous multicore platform, NVIDIA Jetson Nano. We configure the AR-HUD application such that it can process incoming image frames in real-time at 20Hz on the platform. We use micro-architectural denial-of-service (DoS) attacks as aggressor tasks of the challenge and show that they can dramatically impact the latency and accuracy of the AR-HUD application, which results in significant deviations of the estimated trajectories from the ground truth, despite our best effort to mitigate their influence by using cache partitioning and real-time scheduling of the AR-HUD application. We show that dynamic LLC (or DRAM depending on the aggressor) bandwidth throttling of the aggressor tasks is an effective mean to ensure real-time performance of the AR-HUD application without resorting to over-provisioning the system.

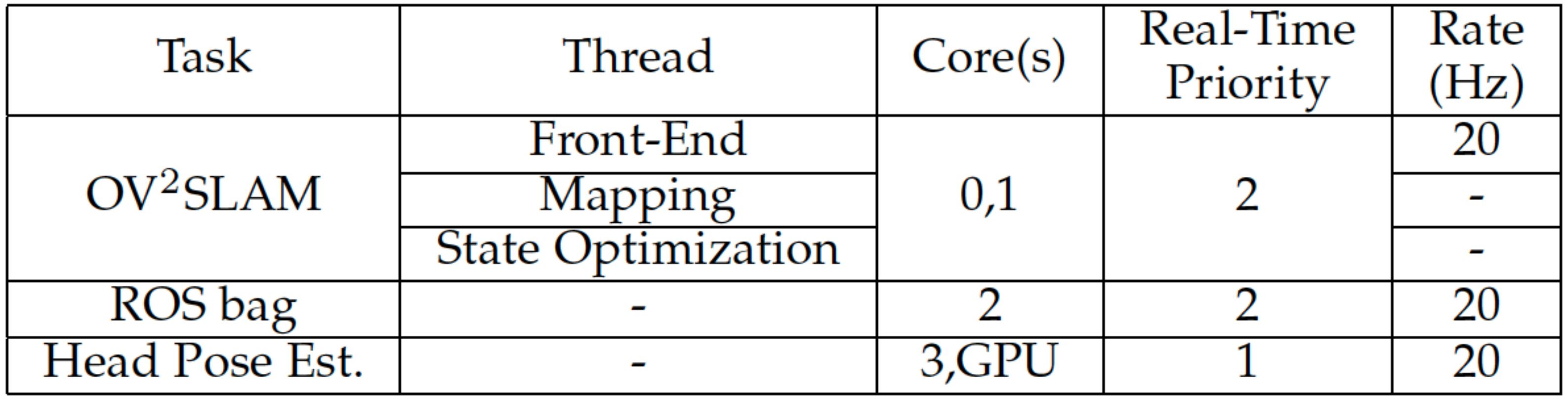
2012 ACM Subject Classification Computer systems organization → Real-time systems; Computer systems organization → Embedded and cyber-physical systems

Keywords and phrases Industrial Challenge, Real Time, SLAM, Microarchitectural DoS Attack

AR-HUD mapping on Jetson Nano



AR-HUD mapping on Jetson Nano



Task	Thread	Core(s)	Real-Time Priority	Rate (Hz)
OV^{2}SLAM	Front-End	0,1	2	20
	Mapping			-
	State Optimization			-
ROS bag	-	2	2	20
Head Pose Est.	-	3,GPU	1	20

Real-time tasks (Linux SCHED_FIFO) threads/core mapping and scheduling parameters

- Aggressor (DoS attack) tasks are scheduled on all cores as best-effort tasks (using Linux CFS scheduler) to fully load the system
- L2 cache is *partitioned* w/ page coloring (*): OV^{2}SLAM vs. all else

(*) H. Yun, R. Mancuso, Z. Wu, R. Pellizzoni. "PALLOC: DRAM Bank-Aware Memory Allocator for Performance Isolation on Multicore Platforms." In *IEEE RTAS*, 2014

Micro-architectural DoS attacks

```
for (int64_t i = 0; i<mem_size; i += LINE_SIZE)
{
    ptr[i] = 0xff;
}
```

Sequential Attacker
(BwWrite)

```
static int* list[MAX_MLP];
static int next[MAX_MLP];

for (int64_t i = 0; i < iter; i++) {
    switch (mlp) {
    case MAX_MLP:
    .
    .
    case 2:
        list[1][next[1]+1] = 0xff;
        next[1] = list[1][next[1]];
        /* fall-through */
    case 1:
        list[0][next[0]+1] = 0xff;
        next[0] = list[0][next[0]];
    }
}
```

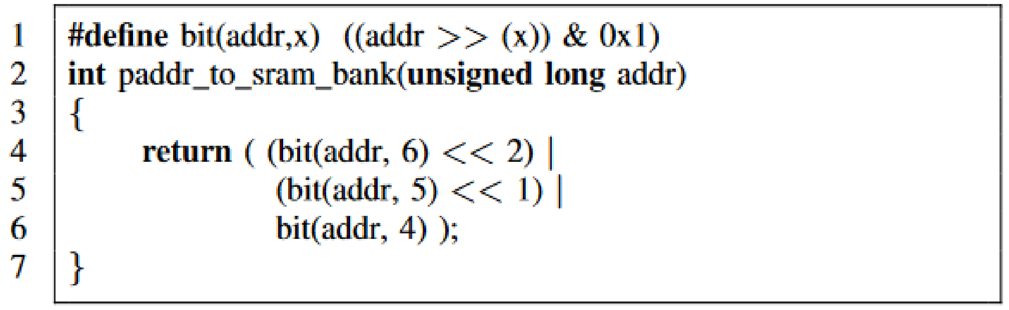
Parallel Linked-List Attacker
(PLLWrite)

- Configurable synthetic workloads to cause resource contention
  - Sequential vs. random, read vs. write access patterns

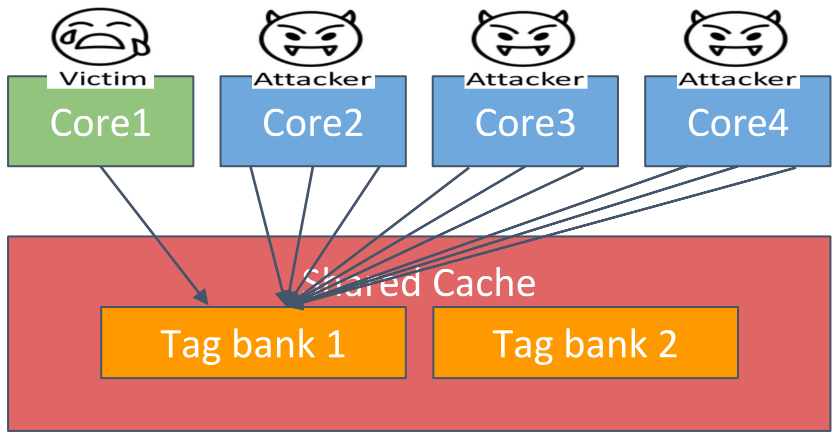
M. G. Bechtel and H. Yun. "Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors." In *IEEE RTAS,* 2023

Cache bank-aware DoS attack

- Same as Parallel Linked-List (PLL) attacks but only keeps the addresses that map to a target LLC data bank
- **LLC bank-aware PLL write attack = BkPLLWrite(LLC)**

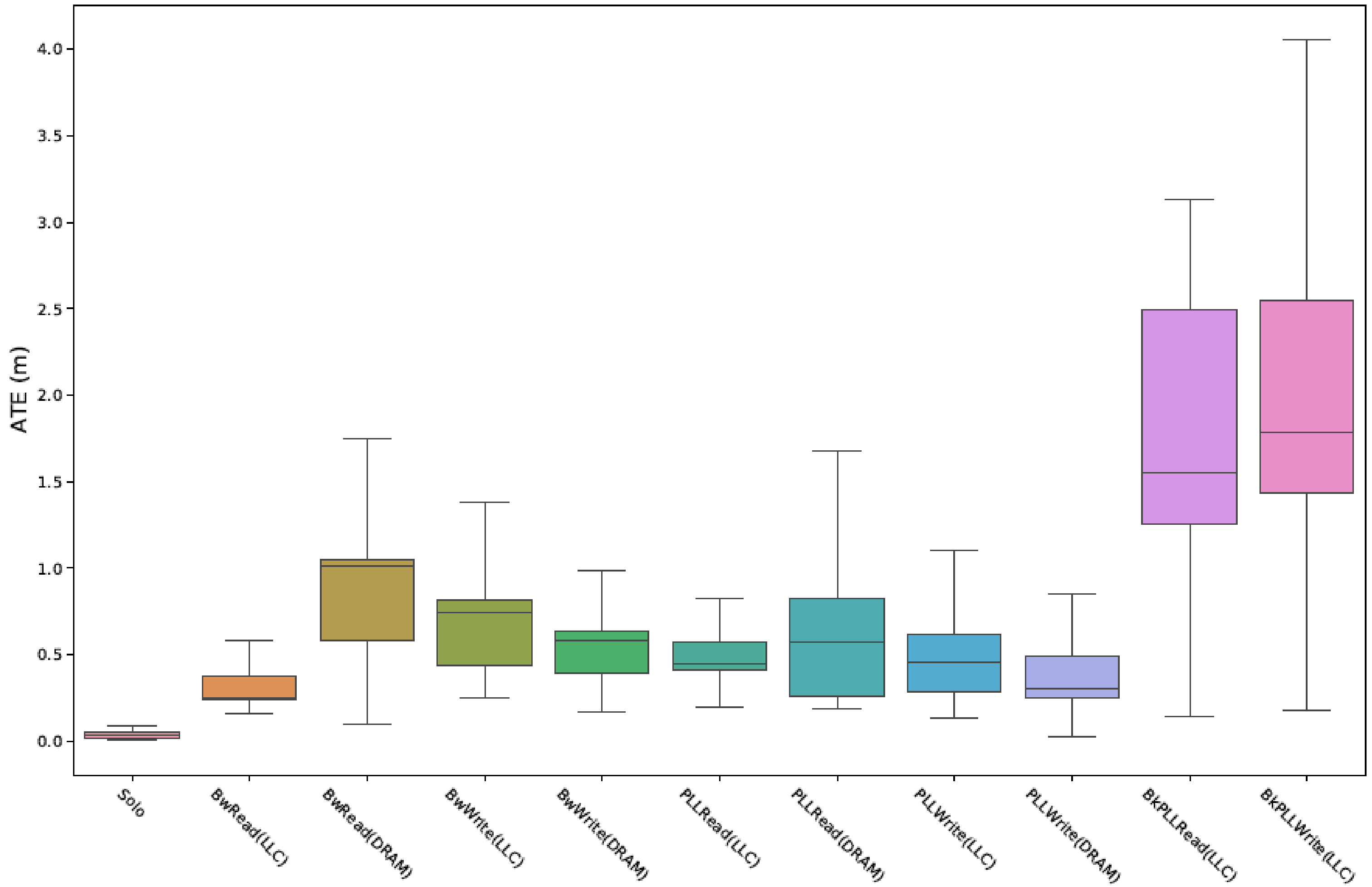


```
#define bit(addr,x)  ((addr >> (x)) & 0x1)
int paddr_to_sram_bank(unsigned long addr)
{
        return ( (bit(addr, 6) << 2) |
                      (bit(addr, 5) << 1) |
                      bit(addr, 4) );
}
```



M. G. Bechtel and H. Yun. "Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors." In *IEEE RTAS,* 2023

Impact of DoS attacks on the OV^{2}SLAM



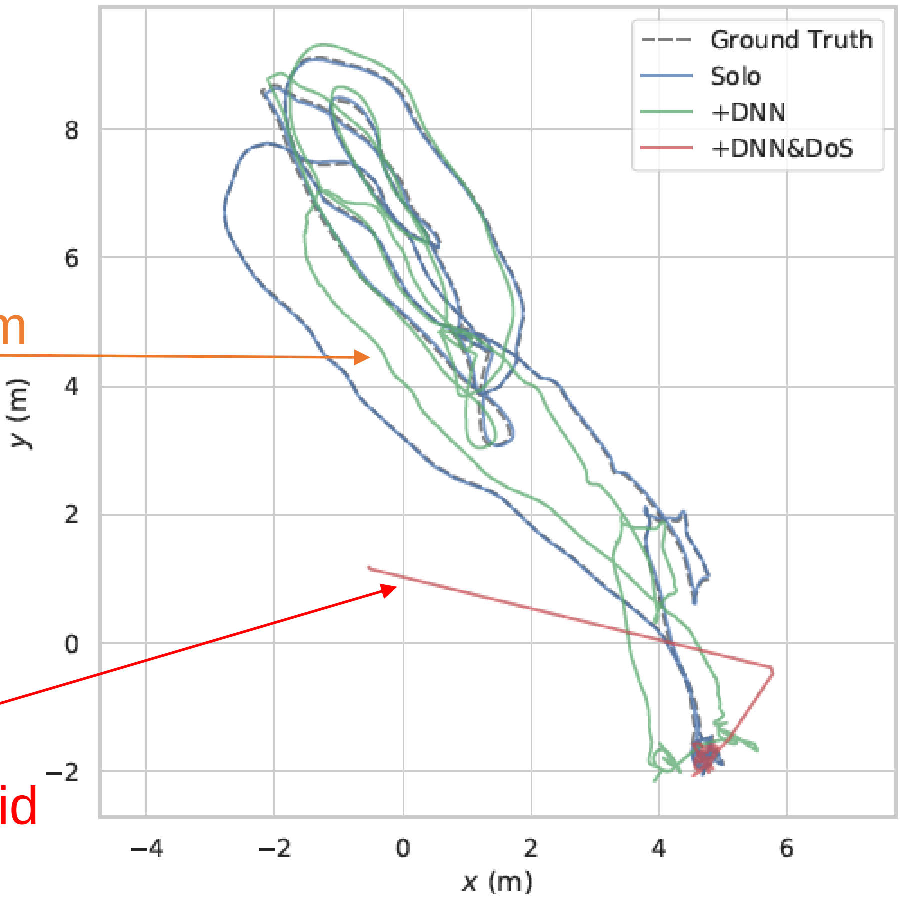
Working-set sizes:
(LLC) -> LLC fitting
(DRAM) -> DRAM fitting

Y-axis: Absolute Trajectory Error (ATE) = a standard measure of accuracy of SLAM. Lower is better.
X-axis: different micro-architectural denial-of-service (DoS) attacks

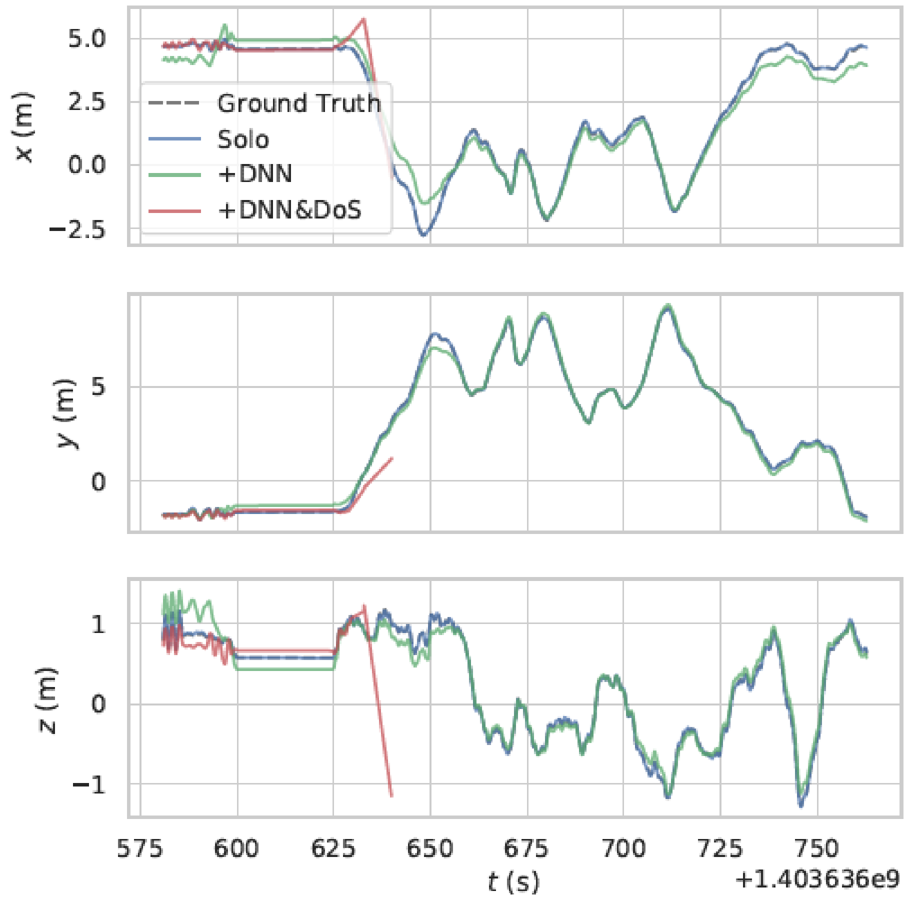
Impact of DNN and DoS attacks on OV^{2}SLAM

Significantly deviated from the true trajectory

Completely failed to generate valid trajectory



(a) Trajectory in XY plane



(b) X, Y and Z positions over time

Our approach: RT-Gang++

Cache bandwidth throttling

- Throttle attacker’s access (from CPU) to the shared LLC
- Using per-core performance counters (based on MemGuard*)
- To limit cache (bank) bandwidth contention

GPU bandwidth throttling

- Throttle HopeNet DNN’s access (from GPU) to the shared DRAM
- Using NVIDIA’s memory controller level throttling mechanism
- To limit GPU induced memory b/w interference on CPU (running SLAM)

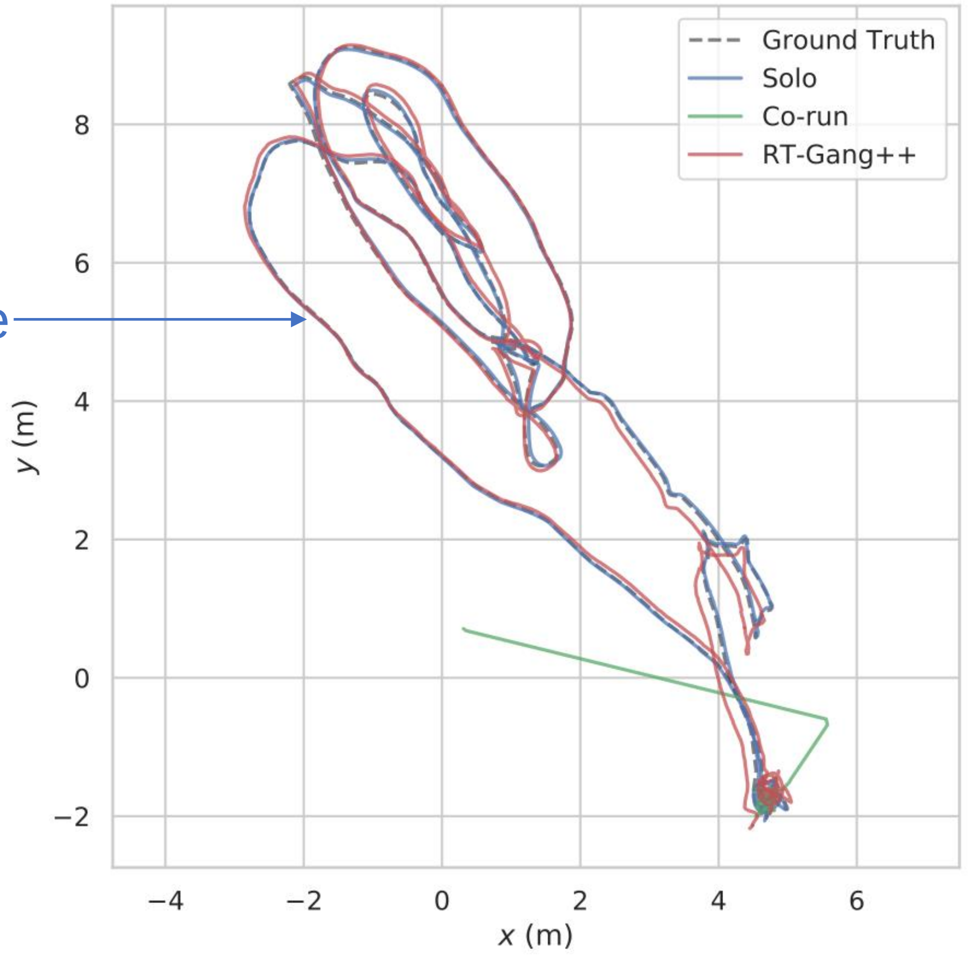
Partitioned gang scheduling

- To avoid inter-application interference on multiple multi-threaded RT apps.

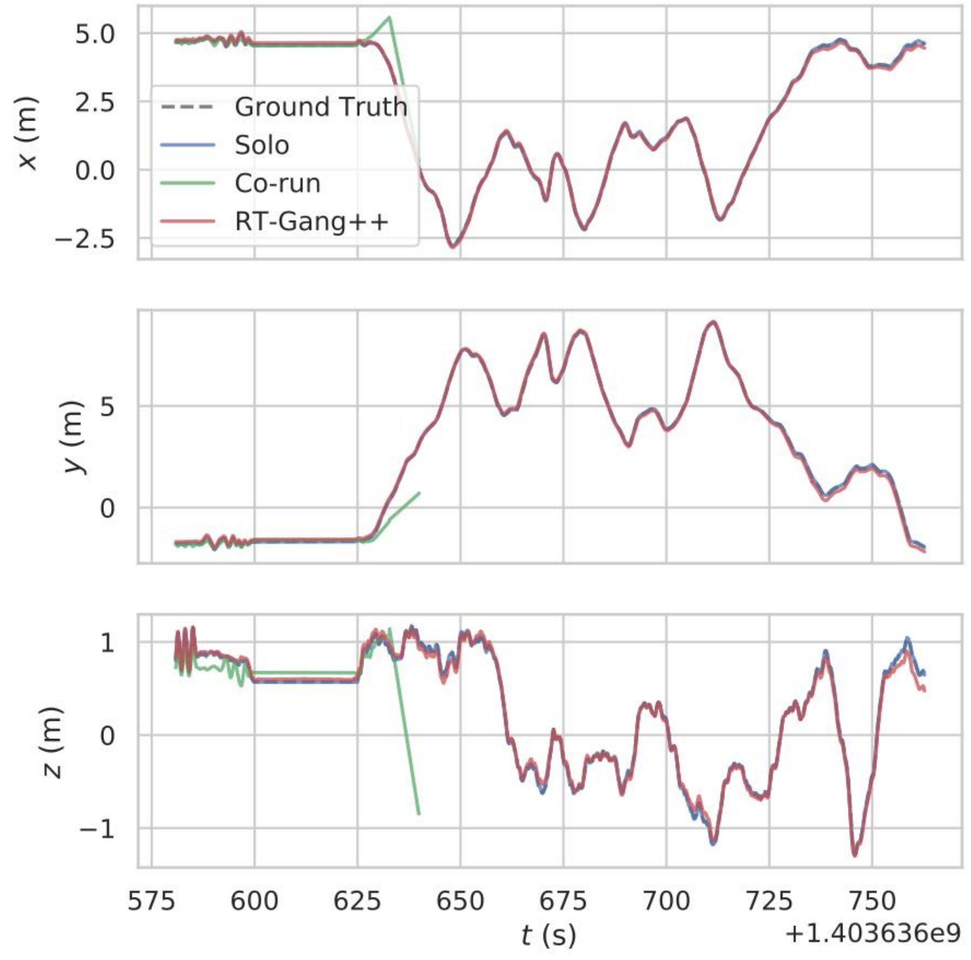
(*) H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha. "MemGuard: Memory Bandwidth Reservation System for Efficient Performance Isolation in Multi-core Platforms." In *IEEE RTAS*, 2013

Impact of RT-Gang++ on OV^{2}SLAM

Successful interference mitigation



(a) Trajectory in X-Y plane



(b) X, Y, and Z positions over time.

Summary

- Consolidating multiple RT/NRT tasks on heterogeneous multicore is challenging due to interference on shared hardware resources
- Cache bank-aware DoS attacks are especially effective in impacting performance of the real-time SLAM task in the AR-HUD case-study
- Executing a DNN task on the integrated GPU also significantly impact the performance of the SLAM on the CPU
- RT-Gang++ mitigates the interference problem via (1) software-based cache bandwidth throttling, (2) hardware-based GPU bandwidth throttling, and (3) partitioned real-time gang scheduling.

Agenda

- Understanding hardware interference channels
  - Non-blocking cache
  - Banked cache and DRAM organizations
  - Effective “attack” strategies to cause massive cross-core interference
- AR-HUD automotive case study (ARM industrial challenge)
  - Effects of interference on real-time application performance
  - Limitations of existing mitigation solutions
  - Our solution to mitigate the interference problem
- Discussion and conclusion

“Better” hardware support?

- Intel Resource Director Technology (RDT)
  - Available on recent Intel server processors
  - Cache space (CAT) and memory bandwidth (MBA) control
  - Not satisfactory for real-time, according to our studies (*)

- ARM Memory System Resource Partitioning and Monitoring (MPAM)
  - Not widely available yet (Where can I find one?)
  - Also focus on (cache) space and (memory) bandwidth
  - **May not be sufficient for real-time systems?**

(*) P. Sohal, M. G. Bechtel, R. Mancuso, H. Yun, O. Krieger. “A Closer Look at Intel Resource Director Technology (RDT),” in *RTNS*, 2022
M. G. Bechtel and H. Yun. “Denial-of-Service Attacks on Shared Resources in Intel’s Integrated CPU-GPU Platforms,” in *ISORC*, 2022

MPAM functionality

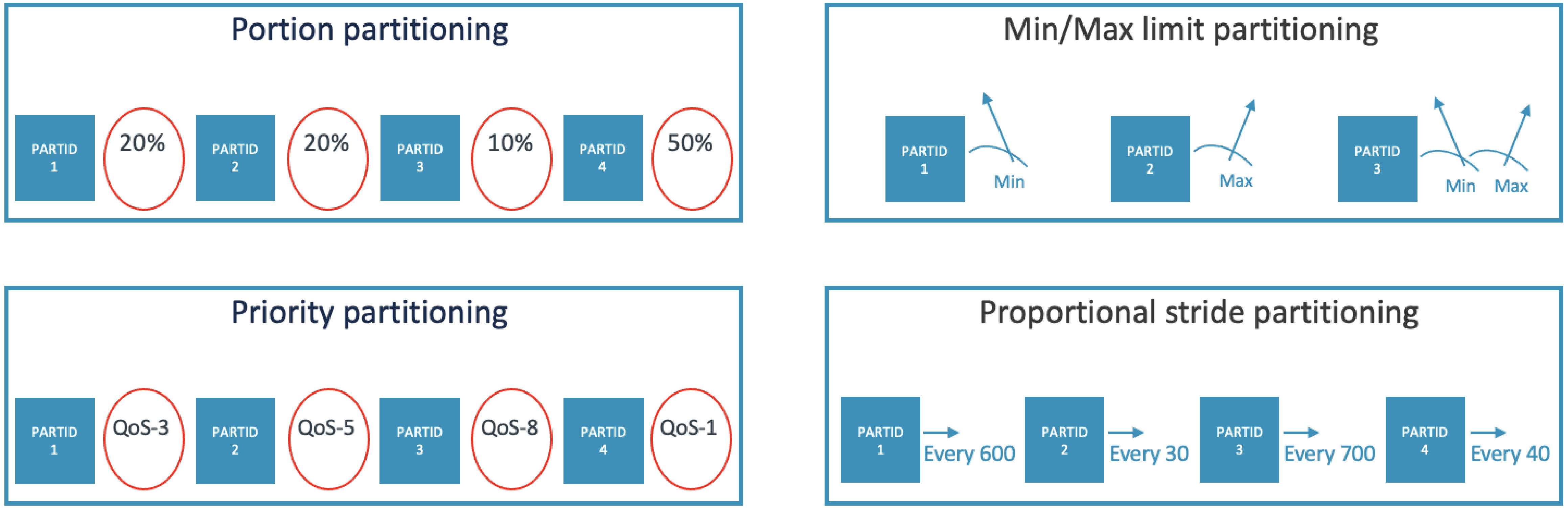


Figure source: ARM, “Learn the architecture - Memory System Resource Partitioning and Monitoring (MPAM) Software Guide,” 2023

MPAM cache portion (way) control

Figure 4-1: MPAM cache_portion_interface

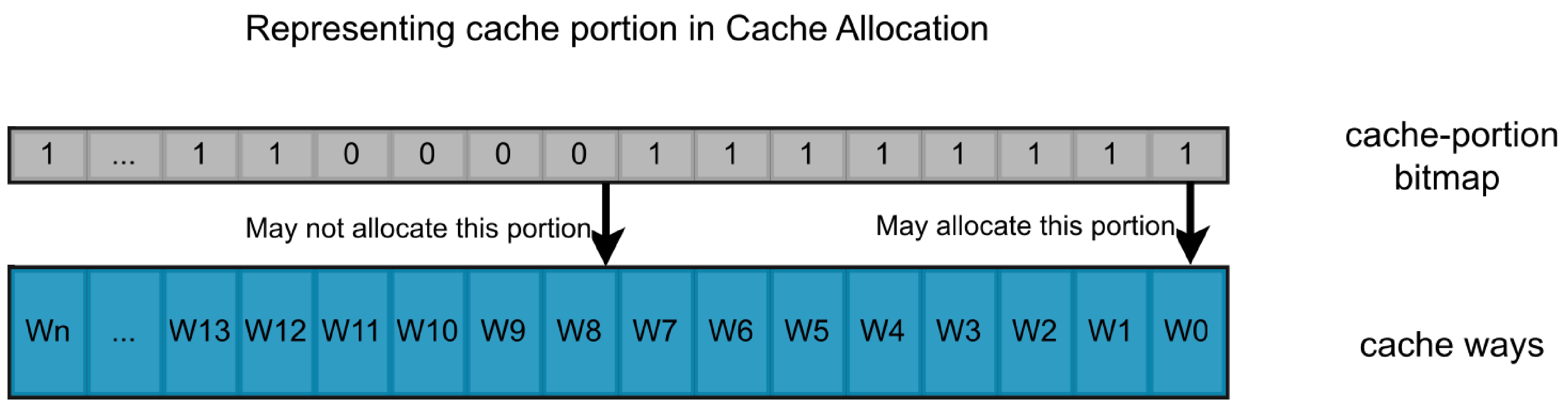

Representing cache portion in Cache Allocation
1 ... 1 1 0 0 0 0 1 1 1 1 1 1 1 1
cache-portion bitmap
May not allocate this portion
May allocate this portion
Wn ... W13 W12 W11 W10 W9 W8 W7 W6 W5 W4 W3 W2 W1 W0
cache ways

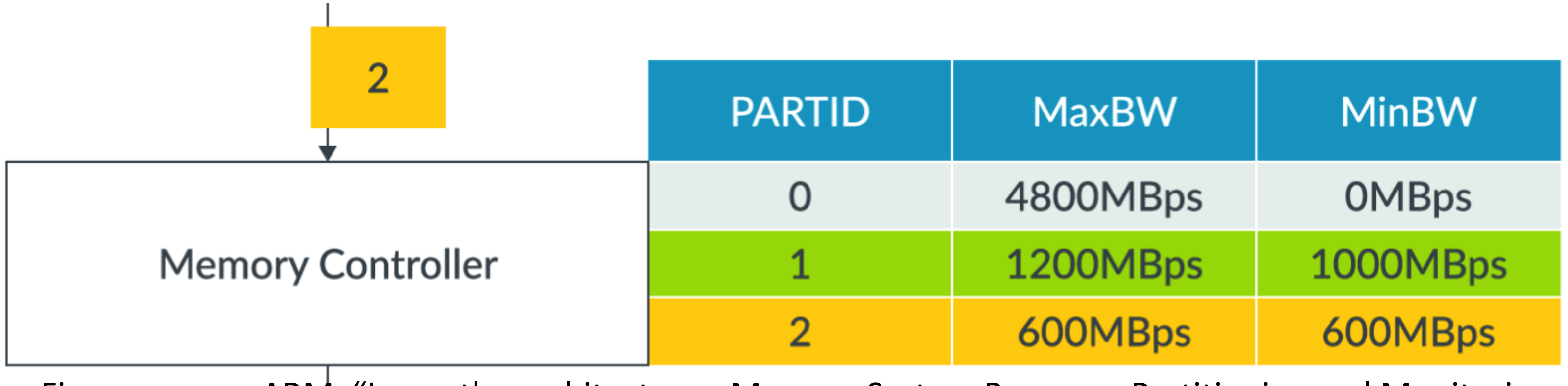
Figure source: ARM, “Learn the architecture - Memory System Resource Partitioning and Monitoring (MPAM) Software Guide,” 2023

- Problem: can control way, but what about bank? (*)

(*) M. G. Bechtel and H. Yun. “Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors.” In *IEEE RTAS,* 2023

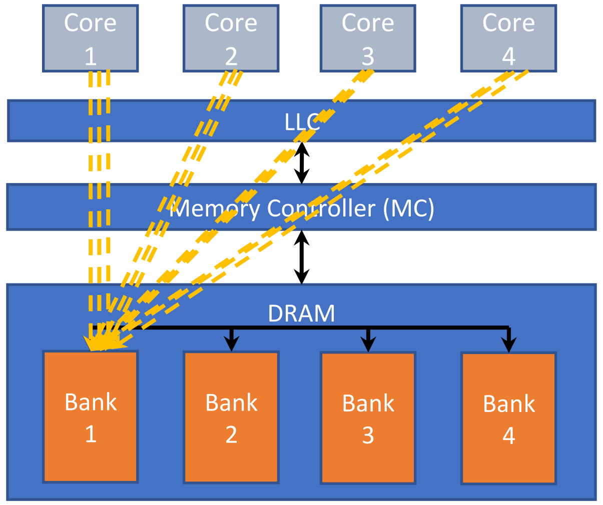
MPAM memory bandwidth control

- **Min/max limit partitioning**


2
Memory Controller

PARTID	MaxBW	MinBW
0	4800MBps	0MBps
1	1200MBps	1000MBps
2	600MBps	600MBps

Figure source: ARM, “Learn the architecture - Memory System Resource Partitioning and Monitoring (MPAM) Software Guide,” 2023



- Problem: stressing one DRAM bank (1/N peak b/w) can cause more delay than stressing all DRAM banks (*)

(*) M. G. Bechtel and H. Yun. “Memory-Aware Denial-of-Service Attacks on Shared Cache in Multicore Real-Time Systems.” In *IEEE Transactions on Computers*, 2021

Better hardware support: a wish list

- Better monitoring and throttling capabilities
  - E.g., Per-bank (cache/dram) perf. counters and bandwidth regulators
- Better control over address-based resource mapping
  - E.g., s/w controlled paddr -> bank mapping (of shared cache and DRAM)
- Better control over other internal shared hardware resources
  - E.g., MSHRs (*), write-back buffer, etc.
- Better memory abstraction
  - E.g., deterministic memory type (**)

(*) P. K. Valsan, H. Yun, F. Farshchi. "Taming Non-blocking Caches to Improve Isolation in Multicore Real-Time Systems." In *RTAS*, 2016
(**) F. Farshchi, P. K. Valsan, R. Mancuso, H. Yun. "Deterministic Memory Abstraction and Supporting Multicore System Architecture." In *ECRTS*, 2018

Conclusion

- Hardware interference channels on heterogeneous multicore are a serious threat to safety-critical real-time applications
- Existing techniques such as cache (space) partitioning and memory bandwidth throttling may not be sufficient
  - Unaware of cache/DRAM banks, internal shared hardware structures
  - Do not necessarily provide worst-case execution time guarantees
- Better hardware support is needed for critical real-time systems

References

- Michael Garrett Bechtel, Heechul Yun. Analysis and Mitigation of Shared Resource Contention on Heterogeneous Multicore: An Industrial Case Study. *arXiv preprint (arXiv:2304.13110),* August 2023. [paper] [arXiv]
- Michael Garrett Bechtel and Heechul Yun. Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors. *IEEE Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (RTAS)*, May 2023. [paper] [slides] [code]
- Parul Sohal, Michael Bechtel, Renato Mancuso, Heechul Yun, Orran Krieger. A Closer Look at Intel Resource Director Technology (RDT). *International Conference on Real-Time Networks and Systems (RTNS)*, 2022 [paper]
- Michael Garrett Bechtel and Heechul Yun. Denial-of-Service Attacks on Shared Resources in Intel's Integrated CPU-GPU Platforms. *IEEE International Symposium on Real-Time Distributed Computing (ISORC)*, 2022 [paper] [code]
- Michael Bechtel and Heechul Yun. Memory-Aware Denial-of-Service Attacks on Shared Cache in Multicore Real-Time Systems. *IEEE Transactions on Computers*, 2021. [paper] [code]
- Farzad Farshchi, Qijing Huang, and Heechul Yun. BRU: Bandwidth Regulation Unit for Real-Time Multicore Processors. *IEEE Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2020. [paper] [slides] [code]
- Michael Garrett Bechtel and Heechul Yun. Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention. *IEEE Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2019. [paper] [arXiv] [slides] [code] [data] (**Outstanding Paper Award**)
- Farzad Farshchi, Prathap Kumar Valsan, Renato Mancuso, Heechul Yun. Deterministic Memory Abstraction and Supporting Multicore System Architecture. *Euromicro Conference on Real-Time Systems (ECRTS),* 2018 [paper] [slides] [code]
- Prathap Kumar Valsan, Heechul Yun, Farzad Farshchi. Taming Non-blocking Caches to Improve Isolation in Multicore Real-Time Systems. *IEEE Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016. [paper] [slides] [code] (**Best Paper Award**)
- Heechul Yun, Rodolfo Pellizzoni, Prathap Kumar Valsan. Parallelism-Aware Memory Interference Delay Analysis for COTS Multicore Systems. *Euromicro Conference on Real-Time Systems (ECRTS)*, 2015. [paper] [slides]
- Heechul Yun, Renato Mancuso, Zheng-Pei Wu, Rodolfo Pellizzoni. PALLOC: DRAM Bank-Aware Memory Allocator for Performance Isolation on Multicore Platforms. *IEEE Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2014. [paper] [slides] [code]
- Heechul Yun, Gang Yao, Rodolfo Pellizzoni, Marco Caccamo, and Lui Sha. MemGuard: Memory Bandwidth Reservation System for Efficient Performance Isolation in Multi-core Platforms. *IEEE Intl. Conference on Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2013. [paper][slides] [code]