Understanding and Mitigating Hardware Interference Channels on Heterogeneous Multicore

Heechul Yun
University of Kansas, USA.
heechul.yun@ku.edu

***Abstract*—Contention for shared microarchitectural resources is a significant challenge for using modern heterogeneous multicore processors in safety-critical real-time applications.**

In this talk, I will first discuss major microarchitectural interference channels in modern multicore architecture. I will describe why microarchitectural designs aiming to achieve high memory-level parallelism (MLP), such as non-blocking cache and banked cache/DRAM organizations, could also become problematic interference channels from a real-time perspective. To illustrate their potential dangers, I will explain several effective "attack" strategies that have been shown to cause massive cross-core interference in real multicore platforms.

Next, I will focus on the industrial challenge put forth by ARM in 2022, which is based on an augmented reality head-up display case-study from the automotive domain. I will discuss the impact of "aggressor" workloads on the performance of the case study's real-time applications on a real heterogeneous multicore platform (Jetson Nano). I will then present our software-based solution to achieve desired real-time performance without resorting to over-provisioning. Lastly, I will discuss the limitations of our approach and the need for better hardware support.

I. Introduction

Heterogeneous multicore platforms are increasingly being adopted in safety-critical cyber physical systems (CPS) to meet their growing performance requirements while meeting size, weight, power and cost (SWaP-C) constraints. Applications running on these systems are often subject to stringent real-time and safety requirements. However, meeting these requirements on a heterogeneous multicore platform is challenging because contention on shared hardware resources among the computing elements in the platform can dramatically impact the execution timing and jeopardize the safety of the system.

In this presentation, we first summarize common characteristics of modern multicore architecture, which are essential for performance but also problematic interference channels from a real-time perspective. We then describe how we mitigated the interference channels of a heterogeneous multicore platform to tackle the ARM industrial challenge [4]. Lastly, we discuss the potentials and limitations of existing hardware support for shared hardware resource management and the need for better hardware support.

II. Understanding Interference Channels

Modern multicore architectures employ various techniques to improve performance. Much of the optimization is focused on improving memory performance, as memory has become a critical bottleneck that limits the performance of multicore [21]. The introduction of heterogeneous computing units, such as graphic processing units (GPUs), only exacerbates the memory problem. Therefore, memory-level parallelism (MLP) is key to understand modern multicore architecture, where MLP refers to the ability to perform multiple memory operations concurrently in parallel.

In a modern multicore processor (MCP), even a single CPU core can generate multiple concurrent memory requests to the memory hierarchy. Integrating multiple cores and accelerators on a single chip means that the memory hierarchy—caches, buses, memory controllers, and the main memory—must be able to provide a very high degree of MLP to minimize memory related stalls. For this reason, the interconnect (bus) used in modern multicore supports split-transactions to allow many concurrent outstanding memory accesses [18]. The caches are non-blocking as they can serve cache hit requests even while multiple outstanding cache misses are pending [15]. Memory controllers can buffer multiple requests and schedule them in ways to maximize memory performance (throughput) [16], [23]. In addition, caches and main memory (DRAM) adopt multi-bank organizations so that access to different banks can occur simultaneously without delay [5], [6], [23]. In short, high MLP designs at all levels of the memory hierarchy are essential for performance in modern multicore architectures.

However, they can also be problematic interference channels. Firstly, we have shown that, even after cache partitioning, a task accessing a dedicated cache partition can be delayed dramatically due to contention in other shared hardware resources, such as miss-status-holding registers (MSHRs) and write-back buffers in non-blocking caches [20], [12]. Moreover, we have shown that the multi-bank organization of shared cache and DRAM can also be exploited to cause extreme worst-case contention and slowdowns [8], [11], [10]. Specifically, we show that generating lots of concurrent memory requests that are targeted at a specific cache or DRAM bank (i.e., *bank-aware* memory access) is much more effective in creating contention than simply generating lots of requests that may be accessing all cache/dram banks (i.e., bank unaware). The latter can be processed efficiently in parallel, while the former cannot, thereby dramatically delaying the victim's requests to the contended bank.

In summary, our findings are: (1) MLP is key to understand modern multicore processors; (2) 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;

979-8-3503-6334-0/24/$31.00 ©2024 IEEE
DOI 10.1109/RAGE62451.2024.00011

(3) Contrary to popular beliefs, interconnects are usually not major interference channels in modern MCPs. Major ones are at the edges; (4) There are effective "attack" strategies to cause massive cross-core interference, which cannot be easily mitigated by existing software/hardware partitioning techniques.

III. Mitigating Interference Channels

In 2022, ARM presented an industrial challenge to stimulate innovations in developing "novel methodologies, tools, and best practices to assist application designers working on high-performance real-time systems" [4]. The case-study in the challenge is based on an augmented reality heads-up display (AR-HUD) application, whose main workloads are composed of a visual simultaneous localization and mapping (SLAM) and a driver head pose estimation deep neural network (DNN). According to ARM [4], it can be considered a representative example of a high-performance real-time use case. The challenge introduces "aggressor" workloads, which may compete for the shared hardware resources alongside the main workloads. As such, a key objective is to ensure the real-time performance of the main real-time workloads in the presence of the aggressor workloads.

In [10], we study the effects of various aggressor workloads on the real-time performance of the AR-HUD application, particularly the SLAM task, on a heterogeneous multicore platform (NVidia Jetson Nano). While all tested aggressor workloads were effective at degrading the performance of the SLAM task, a cache bank-aware denial-of-service (DoS) attack, which targets a specific shared L2 cache bank [11], has shown to be the most effective. This is despite our best effort to isolate the SLAM task by applying a state-of-the-art cache partitioning technique [22], and allocating a dedicated cache partition to it. Because the aggressor targets the L2 cache but not the DRAM, memory bandwidth throttling techniques such as MemGuard [24] are also not effective in protecting the SLAM task. We also find that even without any aggressors, the other main workload of the AR-HUD application, the DNN task, which runs on the integrated GPU, significantly degrades the performance of the SLAM due to DRAM bandwidth contention between the CPU and the GPU.

To tackle the ARM industrial challenge, we propose RT-Gang++ [10], which integrates and extends several techniques we previously developed to mitigate the hardware interference channels. First, it supports L2 cache bandwidth throttling [11] by extending MemGuard [24] to monitor and regulate L2 cache bandwidth rather than DRAM bandwidth. Second, it supports memory bandwidth throttling of the integrated GPU by utilizing the platform's hardware-level memory throttling capability [17]. Lastly, it implements a partitioned gang scheduling capability by extending RT-Gang [3], [2]. We show that RT-Gang++ enables strong isolation for the critical real-time workloads of the AR-HUD case study application in the presence of the fully loaded aggressors on the system.

In summary, our findings are: (1) Consolidating multiple workloads with mixed criticality on heterogeneous multicore is challenging due to interference on shared hardware resources; (2) Cache bank-aware DoS attacks are especially effective in impacting performance of the real-time SLAM task in the AR-HUD case-study; (3) Executing a DNN task on the integrated GPU also significantly impact the performance of the SLAM on the CPU; (4) RT-Gang++ mitigates the interference problem via software-based cache bandwidth throttling, hardware-based GPU bandwidth throttling, and partitioned real-time gang scheduling.

IV. Hardware Support for Mitigation

In recent years, Intel and ARM introduced hardware-level support to monitor and manage shared hardware resources in multicore. Intel Resource Director Technology (RDT) [14] is already available on recent Intel server processors as well as some embedded processors, and it includes Cache Allocation Technology (CAT) and Memory Bandwidth Allocation (MBA), which allow cache space (way) partitioning and memory bandwidth throttling, respectively. ARM's Memory System Resource Partitioning and Monitoring (MPAM) [7] also provide similar capabilities for ARM architectures. Unfortunately, our studies on Intel's CAT and MBA on two recent generations of Intel processors show that they do not provide strong isolation guarantees needed for critical real-time systems [19], [9]. Other researchers have also reported limitations of ARM MPAM [25] from the real-time perspective. Moreover, both RDT and MPAM mainly focus on shared cache space and memory bandwidth. They do not consider other pervasive shared hardware resources or the multi-bank nature of cache and DRAM resources, which can have profound impact to the application execution time as we discussed earlier.

We believe better hardware support is needed to meet the real-time and safety requirements of safety-critical CPS. We present our "wish-list" for such potential improvements that are currently lacking in existing hardware support by Intel and ARM. First, better monitoring and throttling capabilities are needed. For example, per-bank bandwidth monitoring and regulating capabilities could allow more efficient and effective bandwidth control for better isolation with minimal throughput impact. Second, better control over physical address-based resource mapping is needed. For instance, software controlled physical address to cache or DRAM bank mapping could enable flexible bank partitioning. Third, better control over all shared hardware resources in the memory hierarchy is needed. For example, we proposed partitioning of miss-status-holding-registers (MSHRs) in shared caches [20] to balance parallelism and predictability. A centralized management of the entire memory hierarchy, proposed by Abdelhalim et al. [1], can help bound the worst-case. Lastly, a better memory abstraction between software and hardware is needed. For example, we proposed the Deterministic Memory abstraction, which enables efficient and deterministic handling of certain user-declared memory regions throughout the entire memory hierarchy [13].

V. Conclusion

Hardware interference channels on heterogeneous multicore platforms pose a serious threat to safety-critical real-time applications. Our research has shown that existing techniques, such as cache (space) partitioning and memory bandwidth throttling, may not be sufficient, as they are unaware of the organization of cache/DRAM banks or internal shared hardware structures. We have demonstrated that it is possible to provide stronger isolation guarantees in today's commodity heterogeneous multicore systems. However, we believe that better hardware support is ultimately needed to achieve a better balance between performance and predictability, essential for critical real-time systems.

Acknowledgements

The research described in this article has been supported in part by NSF grant CNS-1815959, CPS-2038923 and NSA Science of Security initiative contract no. H98230-18-D-0009.

References

[1] S. Abdelhalim, D. Germchi, M. Hossam, R. Pellizzoni, and M. Hassan. A tight holistic memory latency bound through coordinated management of memory resources. In *ECRTS*, 2023.

[2] W. Ali, R. Pellizzoni, and H. Yun. Virtual gang scheduling of parallel real-time tasks. In *DATE*, pages 270–275. IEEE, 2021.

[3] W. Ali and H. Yun. RT-Gang: Real-Time Gang Scheduling Framework for Safety-Critical Systems. In *RTAS*, 2019.

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

[5] ARM. *Cortex™-A57 Technical Reference Manual, Rev: r1p3*, 2016.

[6] ARM. *Cortex™-A72 Technical Reference Manual, Rev: r0p3*, 2016.

[7] ARM. *Arm Architecture Reference Manual Supplement: Memory System Resource Partitioning and Monitoring (MPAM), DDI:0598B.b*, 2020.

[8] M. Bechtel and H. Yun. Memory-Aware Denial-of-Service Attacks on Shared Cache in Multicore Real-Time Systems. *Transactions on Computers*, 2021.

[9] M. Bechtel and H. Yun. Denial-of-Service Attacks on Shared Resources in Intel's Integrated CPU-GPU Platforms. In *ISORC*, 2022.

[10] M. Bechtel and H. Yun. Analysis and mitigation of shared resource contention on heterogeneous multicore: An industrial case study. *arXiv preprint arXiv:2304.13110*, 2023.

[11] M. Bechtel and H. Yun. Cache Bank-Aware Denial-of-Service Attacks on Multicore ARM Processors. In *RTAS*, 2023.

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

[13] F. Farshchi, P. K. Valsan, R. Mancuso, and H. Yun. Deterministic Memory Abstraction and Supporting Multicore System Architecture. In *ECRTS*, 2018.

[14] Intel. Intel® Resource Director Technology (Intel® RDT) Framework. https://www.intel.com/content/www/us/en/architecture-and-technology/resource-director-technology.html.

[15] D. Kroft. Lockup-free instruction fetch/prefetch cache organization. In *ISCA*, 1998.

[16] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. Owens. Memory Access Scheduling. In *ACM SIGARCH Computer Architecture News*, 2000.

[17] E. Seals, M. Bechtel, and H. Yun. Bandwatch: A system-wide memory bandwidth regulation system for heterogeneous multicore. In *RTCSA*, 2023.

[18] J. P. Shen and M. H. Lipasti. *Modern Processor Design: Fundamentals of Superscalar Processors*. Waveland Press, 2013.

[19] P. Sohal, M. Bechtel, R. Mancuso, H. Yun, and O. Krieger. A Closer Look at Intel Resource Director Technology (RDT). In *RTNS*, pages 127–139, 2022.

[20] P. K. Valsan, H. Yun, and F. Farshchi. Taming Non-blocking Caches to Improve Isolation in Multicore Real-Time Systems. In *RTAS*, 2016.

[21] W. A. Wulf and S. A. McKee. Hitting the memory wall: Implications of the obvious. *ACM SIGARCH computer architecture news*, 23(1):20–24, 1995.

[22] H. Yun, R. Mancuso, Z.-P. Wu, and R. Pellizzoni. PALLOC: DRAM Bank-Aware Memory Allocator for Performance Isolation on Multicore Platforms. In *RTAS*, 2014.

[23] H. Yun, R. Pellizzon, and P. K. Valsan. Parallelism-aware memory interference delay analysis for cots multicore systems. In *ECRTS*. IEEE, 2015.

[24] 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 *RTAS*, 2013.

[25] M. Zini, D. Casini, and A. Biondi. Analyzing arm's mpam from the perspective of time predictability. *IEEE Transactions on Computers*, 72(1):168–182, 2022.