Interference analysis of shared last-level cache on embedded GP-GPUs with multiple CUDA streams

Gianluca Brilli, Paolo Burgio
University of Modena and Reggio Emilia, Italy
{gianluca.brilli, paolo.burgio}@unimore.it

I. Introduction, and motivation for this work

The increasing demand for high-performance computational capabilities at low size-weight and power (SWaP) of modern embedded systems paved the way to the adoption of heterogeneous computing platforms with multi-core host and many-core accelerators. Especially, integrated GPGPUs (iGPUs) [5], [6] are today's preferred to other acceleration paradigms, e.g., based on FPGAs or application-specific integrated circuits (ASICs), in applications with data-parallel workloads, such as computer vision and AI systems employing deep neural networs. This is the case of advanced automotive systems[1], where AI/DNN are increasingly being adopted as reference for building partly- or fully- automated vehicles of tomorrow. Unfortunately, these systems demand not only for high peek performance, but also –and especially– worst case performance, and the increased architectural complexity of modern iGPUs makes it extremely cumbersome to perform an effective non-pessimistic worst-case timing analysis of system. Recently, researchers [2], [4], [7] proved that the main source of unpredictability in such systems are contentions on shared resources, such as memory banks, but yet only few works [3] focused on shared GPU last-level cache (LLC) which also is a major source of contention, *that affects both host and accelerator complexes*. The reason for this lack of material is that, hardware providers (in this case, NVIDIA) are too often reluctant to disclose the internals of their highly-optimized architectures and memory drivers, forcing researcher to a huge effort of reverse engineering for understanding them [3]. This is also interesting because *last-level-caches are the closest shared resources between cores*, hence they are affected by the whole memory traffic due to local caches misses, and deserve a special attention.

CUDA streams. One of the main performance booster, when adopting a host-accelerator paradigm, is the possibility of overlapping multiple computation kernels and data transfers between the CPU and the GPU. In NVIDIA GPGPUs, this is possible thanks to the abstraction of *CUDA streams*, where both execution and data transfer request are issued from the application control running on the host. Unfortunately for RT engineers, CUDA streams introduce an additional level of parallelism, further increasing system complexity, and we will show how the complex mechanism for stream management implemented in platform drivers *enables an additional source of contention in the system*, negatively affecting predictability, because they create interference not only on the shared memory, but also on last-level cache. Circles with numbers in Figure 1 highlight the two main contention points (LLC and memory) in the considered system.

[1]Figure 1 shows a simplified block diagram of a NVIDIA TX2, where an esa-core host shares memory banks with two CUDA streaming multiprocessors (SM) of the Pascal family.

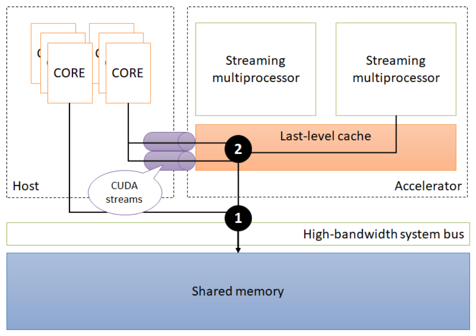

Fig. 1. Reference iGPU architecture with key architectural bottlenecks

Platform modeling in industry. Another issue stems from the fact that industrial-grade frameworks for software development, such as Amalthea [1] for the automotive domain, too often rely on simplified platform model, practically inapplicable and ineffective with the complex structure of iGPUs. Indeed, there is no standard approach to modeling both the implicit memory contention between host cores and GPU cores. Of course, the situation gets even worse when CUDA streams are included in the picture. Indeed, this year's WATERS challenge only focuses on single-stream applications.

Our work wants to be the first one in analyzing and modeling, not only analytically but also with empirical evidence, the contention on LLC introduced by the adoption of multiple CUDA streams.

References

[1] Amalthea Consortium. Amalthea. model based open source development environment for automotive multi core systems, 2014.

[2] R. Cavicchioli, N. Capodieci, and M. Bertogna. Memory interference characterization between CPU cores and integrated gpus in mixed-criticality platforms. In *22nd IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2017, Limassol, Cyprus, September 12-15, 2017*, pages 1–10, 2017.

[3] B. Forsberg, L. Benini, and A. Marongiu. Taming data caches for predictable execution on gpu-based socs. In *DATE'19 – to appear*, 2019.

[4] R. Mancuso, R. Pellizzoni, N. Tokcan, and M. Caccamo. WCET derivation under single core equivalence with explicit memory budget assignment. In *29th Euromicro Conference on Real-Time Systems, ECRTS 2017, June 27-30, 2017, Dubrovnik, Croatia*, pages 3:1–3:23, 2017.

[5] NVIDIA. Jetson TX2 Module, 2017.

[6] NVIDIA. Jetson AGX Xavier Developer Kit, 2018.

[7] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, and L. Sha. Memory bandwidth management for efficient performance isolation in multi-core platforms. *IEEE Trans. Computers*, 65(2):562–576, 2016.