Moving GPU Systems from "Real-Fast" to "Real-Time"

Driven by GPU sales, NVIDIA now outsells AMD and Intel combined.[1] The world is changing, and GPUs—not CPUs—are rapidly becoming the most important processor in a computer system. GPUs have enabled new cyber-physical systems, from intelligent assistants to self-driving cars. Real-world safety or usability concerns impose practical response-time deadlines on these systems. Such systems may also need to run multiple AI tasks—such as one DNN for a conversational interface alongside others for object detection or planning in a self-driving car. However, this raises a problem—how to schedule GPU-using tasks onto a GPU efficiently while reliably meeting deadlines?

I tackle this problem by **(1)** developing priority-aware schedulers to divvy up GPU time, and by **(2)** developing partitioning systems to split GPU cores up among co-running tasks. This latter technique enhances GPU efficiency by increasing the probability that GPU cores will always have pending work. All my work has been supported by **(3)** extensive reverse engineering of NVIDIA's GPU architecture.

Distinct from other work, I emphasize *system-level* scheduling of *unmodified* tasks on a GPU—akin to how CPU scheduling is done in commodity systems. *Practicality* is crucial to my work, so I focus on techniques that work with existing GPU hardware and software stacks. My work is usable on any of the last five generations of NVIDIA GPUs, is open source, and has all participated in and passed artifact evaluation.

Recent and current research. I now detail my three current focuses (bold points above).

Demystifying GPU architecture. GPU capabilities are poorly understood. Recent work at top venues has incorrectly assumed that GPU scheduling is non-preemptive, that GPU engines are jointly scheduled, and that GPU architectures vary highly. My work has shown that these, and other, assumptions are false. To avoid wasting effort on problems that extend no further than a software bug, I have spent substantial effort in understanding the structure and capabilities of GPU architectures. A particular emphasis in this has been to identify *norms*, *i.e.* what aspects of GPU design and scheduling are fundamental to GPU architectures? To support this, I have reverse-engineered the scheduling and topological arrangement of eight generations (12 years) of NVIDIA GPU architectures. I uncovered overly optimistic assumptions in prior work that could compromise the safety of real-time GPU-using systems, and uncovered overly pessimistic assumptions about the limits of GPU scheduling. This area of my work has been published at RTAS 2024 [1], led to co-authorship on several other works [2, 3, 4], and was the topic of my undergraduate honors thesis [5].

A tool I built to aid in my reverse engineering work, `nvdebug`, has proven popular with other research groups, with over 6,700 downloads since statistics were enabled eight months ago.

From time-partitioning to space-partitioning the GPU. By default, NVIDIA GPUs run only one task at a time. Multiple tasks are supported by rapidly switching between them. Unfortunately, this results in lost compute capacity, as one task may not be able to consistently saturate all of a GPU's compute cores.

I addressed this problem by developing a software technique to allow unmodified tasks to concurrently run on the GPU, without the interference characteristic of past techniques. My approach is based on *space-partitioning*—subdividing GPU cores into independently operable sets, and then assigning sets to tasks. This builds on my history of developing space-partitioning techniques for multicore CPUs [6, 7, 8]. My initial GPU partitioning work [9] won one of three outstanding paper awards at RTAS 2023, and my followup work is in-submission to EuroSys 2025 [10]. My partitioning tool `libsmctrl` has been downloaded over 6,600 times in the past eight months (the earliest statistics are available), with users at CMU, Tsinghua University, UNC [11], and TU Munich notable among 65 others.

NVIDIA Research has also reached out about my tool, commending it as better than their internal attempts to solve the same efficiency problem.

Building priority-respecting GPU schedulers. Some GPUs have an insufficient core-count to allow all tasks to run concurrently. In these cases, the GPU must be rapidly switched between tasks over time, *i.e.*, *time-partitioned*. This time-partitioning must take into account priority between tasks, *e.g.*, pending obstacle-detection tasks should always run before infotainment tasks in a self-driving car. Unfortunately, NVIDIA's GPU scheduling algorithm does not support such prioritization.

To address this problem, I developed a GPU time-scheduler that respects task priorities. My scheduler is build on a framework I developed to allow for arbitrary preemptive NVIDIA GPU schedulers. This

[1] Based on third quarter revenue: $7B for AMD and $13B for Intel, roughly half NVIDIA's $35B.

framework was designed to lower the barrier to entry for future GPU scheduling researchers. Distinct from prior work, my framework runs schedulers on the GPU itself, reducing overhead and allowing schedulers built with my framework to work for arbitrary tasks (including virtual machines sharing a GPU). The framework is built on the experience of my years maintaining and contributing schedulers to the LITMUS^{RT} multi-core CPU-scheduling project [2, 6, 7, 8, 12]. The work presenting my on-GPU scheduler and framework is in preparation for submission in January 2025 to USENIX ATC 2025.

Future research. I intend to continue pursuing techniques to make GPU-using systems more time-predictable and efficient by deepening and expanding my current work to address three foreseeable problems.

Guaranteeing timeliness. Some real-time systems are *safety-critical*, where meeting timing constraints is essential for safe operation of the system, *e.g.* detection of obstacles several times a second in a self-driving car. Such systems may require *certification*, where a formal process is required to prove that timing constraints will always be met. In real-time systems, *response-time analysis* is applied to make such guarantees.

I plan to build formal scheduling and partitioning strategies on space- and time-partitioning techniques that are amenable to response-time analysis. I expect that this will enable meeting timing constraints without the throughput compromises of current work.

Adapting to evolving GPUs. Top-end GPU dies have grown from 20 to 200 billion transistors in the past seven years. This has been accomplished through larger and denser dies, causing increased lithography errors. To ensure reasonable production yields, dies containing errors are sold by disabling broken portions. For example, NVIDIA's H100 GPU is sold with up to 16,896 cores enabled, despite 18,432 cores on the die—a loss of 14% due to lithography issues (up from 5% seven years ago). The set of disabled cores affects the GPU's performance, and is randomly distributed. For example, when one core is disabled, nearby cores (which share an interconnect to memory) become slightly faster. This causes multiple issues. First, a task's runtime may vary across GPUs of the same model, making met-deadline guarantees non-portable. Second, the best assignment of tasks to cores will vary die-to-die, so GPU core partitions will have to adapt per-die.

I intend to better expose and demonstrate the cost of this non-uniformity, and then explore ways of scheduling that avoid, or can analytically compensate for, this problem.

Enabling machine-wide task prioritization. Task prioritization is useful in real-time systems, but also in non-real-time systems, *e.g.* to prioritize interactive tasks over background tasks on consumer devices. Unfortunately, modern operating systems have no system-wide notion of priority, with each resource (*e.g.* caches, storage devices, network interfaces, and the GPU) having its own notion of priority.

I want to reconcile these notions of priority, starting with the CPU and GPU, before expanding to address storage I/O and other resources. The culmination of this would be the ability to set a priority value for a task and have it respected by every resource in the system.

Expanding the GPU research community. The secrecy of GPU designs has limited academic research in areas beyond scheduling. For example, I currently collaborate with a faculty member in hardware security on GPU shared-cache and DRAM attacks. My past work includes reverse-engineering GPU virtual memory and cache capabilities [9, 13], and I have re-applied those findings to this hardware security collaboration.

I intend to pursue more collaboration of this sort, by applying what I have reverse engineered about the GPU to problems such as virtualization, memory oversubscription, and ML compilers.

Summary. In my recent work, I have developed and shown space- and time-partitioning techniques that can increase the timeliness of GPU tasks without unduly compromising throughput. My discoveries about *norms* of GPU architecture and scheduling enabled this. In the future, I want to extend my work to emit formal guarantees of timeliness, while tackling problems emerging as GPUs evolve. I look forward to collaborating with a wide variety of other systems researchers to tackle problems that emerge as GPUs become increasingly important in computing systems.

References

[1] J. Bakita and J. H. Anderson, "Demystifying NVIDIA GPU internals to enable reliable GPU management," in *Proceedings of the 30th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 294–305, May 2024.

[2] T. Amert, Z. Tong, S. Voronov, J. Bakita, F. D. Smith, and J. H. Anderson, "TimeWall: Enabling time partitioning for real-time multicore+accelerator platforms," in *Proceedings of the 42nd IEEE Real-Time Systems Symposium*, pp. 455–468, Dec 2021.

[3] M. Yang, S. Wang, J. Bakita, T. Vu, F. D. Smith, J. H. Anderson, and J.-M. Frahm, "Re-thinking CNN frameworks for time-sensitive autonomous-driving applications: Addressing an industrial challenge," in *Proceedings of the 25th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 305–317, Apr 2019.

[4] M. Yang, N. Otterness, T. Amert, J. Bakita, J. H. Anderson, and F. D. Smith, "Avoiding pitfalls when using NVIDIA GPUs for real-time tasks in autonomous systems," in *Proceedings of the 30th Euromicro Conference on Real-Time Systems*, pp. 20:1–20:21, Jul 2018.

[5] J. Bakita, N. Otterness, J. H. Anderson, and F. D. Smith, "Scaling up: The validation of empirically derived scheduling rules on NVIDIA GPUs," in *Proceedings of 14th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications*, pp. 49–54, Jul 2018.

[6] J. Bakita, S. Ahmed, S. H. Osborne, S. Tang, J. Chen, F. D. Smith, and J. H. Anderson, "Simultaneous multithreading in mixed-criticality real-time systems," in *Proceedings of the 27th Real-Time and Embedded Technology and Applications Symposium*, pp. 278–291, May 2021.

[7] S. H. Osborne, J. J. Bakita, and J. H. Anderson, "Simultaneous multithreading applied to real time," in *Proceedings of the 31st Euromicro Conference on Real-Time Systems*, pp. 3:1–3:22, July 2019.

[8] S. H. Osborne, J. Bakita, J. Chen, T. Yandrofski, and J. H. Anderson, "Minimizing DAG utilization by exploiting SMT," in *Proceedings of the 28th Real-Time and Embedded Technology and Applications Symposium*, pp. 267–280, May 2022.

[9] J. Bakita and J. H. Anderson, "Hardware compute partitioning on NVIDIA GPUs," in *Proceedings of the 29th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 54–66, May 2023.

[10] J. Bakita and J. H. Anderson, "Hardware compute partitioning on NVIDIA GPUs for composable systems," in *Proceedings of the 20th European Conference on Computer Systems*, p. in submission, Apr 2025.

[11] S. W. Ali, J. Goh, J. Bakita, S. Chakraborty, and J. H. Anderson, "Concurrent FFT execution on GPUs in real-time," in *Proceedings of the 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, p. to appear, Mar 2025.

[12] C. Hobbs, Z. Tong, J. Bakita, and J. H. Anderson, "Statically optimal dynamic soft real-time semi-partitioned scheduling," *Real-Time Systems*, vol. 57, pp. 97–140, Jan 2021.

[13] J. Bakita and J. H. Anderson, "Enabling GPU memory oversubscription via transparent paging to an NVMe SSD," in *Proceedings of the 43rd IEEE Real-Time Systems Symposium*, pp. 370–382, Dec 2022.