CDS: Deadline-Aware, Multi-Chiplet GPU Scheduler in gem5

Tanmay Anand*, Matthew D. Sinclair†, Michael Swift†
*Microsoft Corporation
tanmayanand@microsoft.com
†University of Wisconsin-Madison
{sinclair, swift}@cs.wisc.edu

I. Introduction

The shift towards multi-chiplet GPU architectures introduces new complexities in task scheduling across distributed compute and memory resources. Moreover, modern real-time applications place stringent demands on GPUs for low-latency execution. Unlike single-die GPUs, multi-chiplet designs [1]–[7] must coordinate work across chiplets with varying latency and bandwidth, making the round-robin scheduling used in modern GPUs [6] inadequate for real-time guarantees.

In this work, we apply hardware–software co-design to address these challenges and sustain performance scaling in future GPGPUs. Modern GPGPUs include embedded RISC cores, known as command processors (CPs), which handle stream scheduling and kernel context parsing. However, current CPs are limited to parsing kernel contexts and scheduling GPU streams, treating the GPU as a monolithic unit rather than a collection of chiplets. Our core idea is to design two-level GPU scheduler called *CDS* (Cross-Die Scheduler) that coordinates work across chiplets while meeting deadlines. We realize CDS by enhancing CPs: because they are co-located with the GPU, CPs can access fine-grained, microsecond-level runtime data while retaining a global view across chiplets.

Our scheduler prioritizes queued jobs based on laxity [8], [9] and selects chiplets for kernel dispatch using a worst-fit strategy over available resources. We also incorporate next-kernel prefetching to eliminate communication delays between consecutive kernels within the same stream. A key contribution of this work is enhancing the capabilities of both local and global CPs to intelligently schedule kernels by monitoring chiplet-level compute resources and workload characteristics.

II. Platform and Interface.

We extend gem5 v21.0 to support local and global CPs and integrate CDS. Real-time scheduling requires knowledge of deadlines. gem5 v21.0 supports ROCm 1.6, in which we modify the HIP interface to support per-stream deadlines. ROCm passes them to the simulator, which stores it in tables for scheduling.

III. Methodology

Our key metrics for the scheduler are (i) completion rate, meaning what fraction of kernels complete before their deadline, (ii) response time, meaning how long after submission did kernels complete, and (iii) chiplet utilization. We evaluate CDS against a round-robin policy used in current GPUs.

We evaluate across a set of single- and multi-kernel workloads [10]–[12]. We initially populate the system with a batch of jobs and then periodically submit subsequent batches with a controlled inter-arrival interval, referred to as the *arrival delay*. We categorize the workload behavior at an arrival rate into different regimes. We assign deadlines to each job by scaling its standalone runtime with a tunable factor, which we relax progressively during evaluation until the scheduler reaches 100% on-time completion rate for steady state jobs within a regime.

These regimes reflect the level of system contention and the corresponding difficulty for a scheduler to meet deadlines. We use a t-test to categorize the outcome, identifying higher arrival rates when response time distributions vary over time and lower arrival rates when response time distributes remain stable. This evaluation methodology establishes a robust framework for assessing schedulers across diverse scenarios and is also a key contribution of this work.

IV. Results

Figure 1 shows the steady-state completion rate for the medium regime at an arrival rate of 48 jobs/ms. CDS achieves a 100% completion rate, while round-robin scheduling meets deadlines for only 78% of jobs, at an average per-job deadline of 1.4 ms for GMM scoring. Round-robin achieves 100% completion at 1.6 ms deadlines. This shows CDS can handle much tighter deadlines while achieving near-ideal completion rates compared to round-robin. Going forward, we aim to redesign the CP's queue scheduler so that CDS can dynamically adapt to changing GPU conditions while effectively accounting for locality.

V. Conclusion

Evaluating schedulers is difficult, especially under tight deadlines or high contention. As workloads grow more complex, we need policies that are robust, efficient, and low-overhead. In this work, we introduce our proposed algorithm for reliably meeting real-time guarantees, outline our evaluation methodology for testing schedulers across diverse scenarios, and present results across a range of workloads.

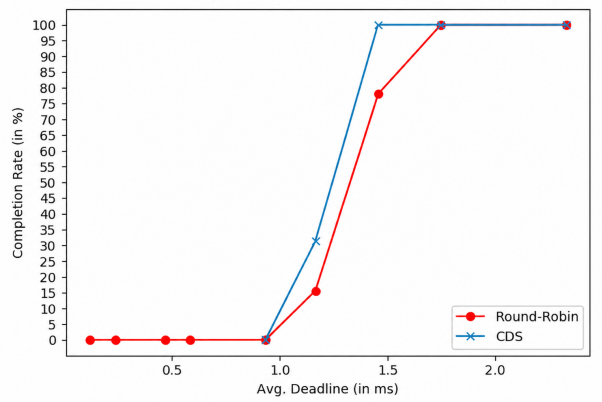

Fig. 1: Steady-state completion rate for GMM [10].

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation grant CAREER-2238608.

REFERENCES

[1] A. Arunkumar, E. Bolotin, B. Cho, U. Milic, E. Ebrahimi, O. Villa, A. Jaleel, C.-J. Wu, and D. Nellans, "MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA. New York, NY, USA: ACM, 2017, pp. 320–332. [Online]. Available: http://doi.acm.org/10.1145/3079856.3080231

[2] A. Arunkumar, E. Bolotin, D. Nellans, and C.-J. Wu, "Understanding the Future of Energy Efficiency in Multi-Module GPUs," in *25th IEEE International Symposium on High Performance Computer Architecture*, ser. HPCA. Piscataway, NJ, USA: IEEE Press, 2019, pp. 519–532.

[3] P. Dalmia, R. Shashi Kumar, and M. D. Sinclair, "CPElide: Efficient Multi-Chiplet GPU Implicit Synchronization," in *Proceedings of 57th IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO. Los Alamitos, CA, USA: IEEE Computer Society, 2024.

[4] M. Khairy, V. Nikiforov, D. Nellans, and T. G. Rogers, "Locality-Centric Data and Threadblock Management for Massive GPUs," in *53rd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO. Washington, DC, USA: IEEE Computer Society, 2020, pp. 1022–1036.

[5] G. H. Loh, M. J. Schulte, M. Ignatowski, V. Adhinarayanan, S. Aga, D. Aguren, V. Agrawal, A. M. Aji, J. Alsop, P. Bauman, B. M. Beckmann, M. V. Beigi, S. Blagodurov, T. Boraten, M. Boyer, W. C. Brantley, N. Chalmers, S. Chen, K. Cheng, M. L. Chu, D. Cownie, N. Curtis, J. Del Pino, N. Duong, A. Duundefinedu, Y. Eckert, C. Erb, C. Freitag, J. L. Greathouse, S. Gurumurthi, A. Gutierrez, K. Hamidouche, S. Hossamani, W. Huang, M. Islam, N. Jayasena, J. Kalamatianos, O. Kayiran, J. Kotra, A. Lee, D. Lowell, N. Madan, A. Majumdar, N. Malaya, S. Manne, S. Mashimo, D. McDougall, E. Mednick, M. Mishkin, M. Nutter, I. Paul, M. Poremba, B. Potter, K. Punniyamurthy, S. Puthoor, S. E. Raasch, K. Rao, G. Rodgers, M. Scrbak, M. Seyedzadeh, J. Slice, V. Sridharan, R. van Oostrum, E. van Tassell, A. Vishnu, S. Wasmundt, M. Wilkening, N. Wolfe, M. Wyse, A. Yalavarti, and D. Yudanov, "A Research Retrospective on AMD's Exascale Computing Journey," in *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ser. ISCA. New York, NY, USA: Association for Computing Machinery, 2023. [Online]. Available: https://doi.org/10.1145/3579371.3589349

[6] M. Osama, R. Swann, K. Sangaiah, S. Singh, G. Dasika, and R. Bhardwaj, "Deep dive into the MI300 compute and memory partition modes," https://rocm.blogs.amd.com/software-tools-optimization/compute-memory-modes/README.html, Feb 2025.

[7] A. Smith, G. H. Loh, M. J. Schulte, M. Ignatowski, S. Naffziger, M. Mantor, N. Kalyanasundharam, V. Alla, N. Malaya, J. L. Greathouse, E. Chapman, and R. Swaminathan, "Realizing the AMD Exascale Heterogeneous Processor Vision : Industry Product," in *51st ACM/IEEE Annual International Symposium on Computer Architecture*, ser. ISCA. Piscataway, NJ, USA: IEEE, 2024, pp. 876–889. [Online]. Available: https://doi.org/10.1109/ISCA59077.2024.00068

[8] M. Cirinei and T. P. Baker, "Edzl scheduling analysis," in *19th Euromicro Conference on Real-Time Systems (ECRTS'07)*, July 2007, pp. 9–18.

[9] T. T. Yeh, M. D. Sinclair, B. M. Beckmann, and T. G. Rogers, "Deadline-Aware Offloading for High-Throughput Accelerators," in *27th IEEE International Symposium on High Performance Computer Architecture*, ser. HPCA. Washington, DC, USA: IEEE Computer Society, 2021, pp. 479–492.

[10] J. Hauswald, M. A. Laurenzano, Y. Zhang, C. Li, A. Rovinski, A. Khurana, R. G. Dreslinski, T. Mudge, V. Petrucci, L. Tang, and J. Mars, "Sirius: An Open End-to-End Voice and Vision Personal Assistant and Its Implications for Future Warehouse Scale Computers," in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS. New York, NY, USA: ACM, 2015, pp. 223–238. [Online]. Available: http://doi.acm.org/10.1145/2694344.2694347

[11] S. Narang, "DeepBench," https://github.com/baidu-research/DeepBench, 2016.

[12] S. Narang and G. Diamos, "An update to DeepBench with a focus on deep learning inference," https://svail.github.io/DeepBench-update/, 2017.