Improving gem5's GPUFS Support

Vishnu Ramadas, Matthew Poremba, Bradford M. Beckmann, Matthew D. Sinclair
University of Wisconsin-Madison, AMD Research
vramadas@wisc.edu Matthew.Poremba@amd.com Brad.Beckmann@amd.com sinclair@cs.wisc.edu

I. Motivation

With the waning of Moore's Law and the end of Dennard's Scaling, systems are turning towards heterogeneity, mixing conventional cores and specialized accelerators to continue scaling performance and energy efficiency. Specialized accelerators are frequently used to improve the efficiency of computations that run inefficiently on conventional, general-purpose processors. As a result, systems ranging from smartphones to data-centers, hyper-scalars, and supercomputers are increasingly using large numbers of accelerators to provide better efficiency than CPU-based solutions. However, heterogeneous systems face key challenges: changes to the underlying technology which threaten continued scaling, as well as the voracious scaling from applications, which require additional research to address. Traditionally, simulators could be used to perform early exploration for this research. However, existing simulators lack important support for these key challenges.

Detailed simulation of modern systems can take extremely long times in existing tools and infrastructure. Furthermore, prototyping optimizations at scale can also be challenging, especially for newly proposed accelerators. Although other simulators such as Accel-Sim [1], SCALE-Sim [2], and Gemmini [3] enable some early experiments, they are limited in their ability to target a wide variety of accelerators. In comparison, gem5 [4], [5] has support for various CPUs, GPUs, DSPs, and many other important accelerators [6]–[9]. However, efficiently simulating large-scale workloads on gem5's cycle-level models requires prohibitively long times. We aim to enhance gem5's support to make running these workloads practical while retaining accuracy.

II. Implementation

Modern AI and ML algorithms are often partitioned across multiple devices (e.g, GPUs) or nodes, which state-of-the-art simulators do not support. Accordingly, we added support into gem5 to model multi-GPU systems [9]. Moreover, we are working on extending gem5 also performing significant optimizations to ensure each node in the system only performs detailed simulation for the most important parts of the program. To enable this, we first extended gem5's existing KVM support to run application components that are lower priority, or those that can be simulated at lower fidelity. This allows gem5 to focus only on the GPU kernel simulations that are of interest in an application.

Checkpointing a Full System (FS) simulation can also present significant performance benefits. While running a simulation for the first time, region(s) of interest can be annotated for a variety of applications. This will help save the state of the system when it reaches these point. Later, when running the simulation a checkpoint file is created that stores a snapshot of the entire state of the system immediately before the region of interest executes. State can then be restored during later simulations to skip over all instructions until the annotation and effectively begin simulating from the region of interest onwards. Restoring from a checkpoint saves significant time during subsequent runs as the simulation does not need to go through the initial steps again. We added checkpointing support to the GPU FS simulation mode by saving not just the state of GPU global memory, but also information about TLB translations, DMA doorbells, and ROCm kernel parameters. We also added functionality to restore from this checkpoint before proceeding with kernel execution.

III. Future Work

We plan to add support for HIP-CPU [10] to simulate GPU components faster. HIP-CPU emulates the AMD's HIP GPU runtime API to port and run GPU kernels on the CPU instead. However, since HIP-CPU does not run the ROCm stack, we will need to determine if it can be used to restore checkpoints after HIP-CPU, or if modifying LLVM to target different backends will be required. Regardless, by adding this support we will reduce fidelity on less important portions of the workload and accordingly reduce gem5's runtime. An initial prototype (using Rodinia [11], [12]) suggests that this approach results in gem5's simulation time being only 1.6-3$\times$ slower than bare metal, in comparison to at least 200$\times$ without these optimizations. We will also integrate gem5-SALAM [8] into the mainline of gem5, extend it to model various accelerators, and use techniques such as fast-forwarding, check-pointing, and intelligent characterization of workloads to optimize accelerator runtimes while avoiding performing the fast-forwarding on the real accelerator.

We also aim to find representative portions of an application to simulate. Using these insights, we will introduce a set of clustering methods to segment work into dynamically created sampled inputs like checkpoints and evaluate how these methods both summarize the original program and speed-up the simulation of the program. Moreover, we will further develop tools to identify algorithm-dependent factors that impact the variation across iterations (e.g., in ML training) before needing to profile or simulate applications [13]. Using these insights will allow us to identify a small subset of the algorithm's work that is representative of each algorithm's behavior while being practical to simulate.

ACKNOWLEDGMENTS

This work is supported in by the Semiconductor Research Corporation grant 3151.001 and National Science Foundation grant ENS-1925485.

REFERENCES

[1] M. Khairy, Z. Shen, T. M. Aamodt, and T. G. Rogers, "Accel-Sim: An Extensible Simulation Framework for Validated GPU Modeling," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ser. ISCA, 2020, pp. 473–486.

[2] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim," in *IEEE International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS, 2020, pp. 58–68.

[3] H. Genc, S. Kim, A. Amid, A. Haj-Ali, V. Iyer, P. Prakash, J. Zhao, D. Grubb, H. Liew, H. Mao, A. Ou, C. Schmidt, S. Steffl, J. Wright, I. Stoica, J. Ragan-Kelley, K. Asanovic, B. Nikolic, and Y. S. Shao, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 769–774.

[4] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[5] J. Lowe-Power, A. M. Ahmad, A. Akram, M. Alian, R. Amslinger, M. Andreozzi, A. Armejach, N. Asmussen, S. Bharadwaj, G. Black, G. Bloom, B. R. Bruce, D. R. Carvalho, J. Castrillon, L. Chen, N. Derumigny, S. Diestelhorst, W. Elsasser, M. Fariborz, A. Farmahini-Farahani, P. Fotouhi, R. Gambord, J. Gandhi, D. Gope, T. Grass, B. Hanindhito, A. Hansson, S. Haria, A. Harris, T. Hayes, A. Herrera, M. Horsnell, S. A. R. Jafri, R. Jagtap, H. Jang, R. Jeyapaul, T. M. Jones, M. Jung, S. Kannoth, H. Khaleghzadeh, Y. Kodama, T. Krishna, T. Marinelli, C. Menard, A. Mondelli, T. Mück, O. Naji, K. Nathella, H. Nguyen, N. Nikoleris, L. E. Olson, M. Orr, B. Pham, P. Prieto, T. Reddy, A. Roelke, M. Samani, A. Sandberg, J. Setoain, B. Shingarov, M. D. Sinclair, T. Ta, R. Thakur, G. Travaglini, M. Upton, N. Vaish, I. Vougioukas, Z. Wang, N. Wehn, C. Weis, D. A. Wood, H. Yoon, and Éder F. Zulian, "The gem5 simulator: Version 20.0+," 2020.

[6] A. Gutierrez, B. M. Beckmann, A. Dutu, J. Gross, M. LeBeane, J. Kalamatianos, O. Kayiran, M. Poremba, B. Potter, S. Puthoor, M. D. Sinclair, M. Wyse, J. Yin, X. Zhang, A. Jain, and T. Rogers, "Lost in Abstraction: Pitfalls of Analyzing GPUs at the Intermediate Language Level," in *2018 IEEE International Symposium on High Performance Computer Architecture*, ser. HPCA, Feb 2018, pp. 608–619.

[7] K. Roarty and M. D. Sinclair, "Modeling Modern GPU Applications in gem5," in *3rd gem5 Users' Workshop*, June 2020.

[8] S. Rogers, J. Slycord, M. Baharani, and H. Tabkhi, "gem5-SALAM: A System Architecture for LLVM-based Accelerator Modeling," in *53rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2020, pp. 471–482.

[9] B. W. Yogatama, M. D. Sinclair, and M. M. Swift, "Enabling Multi-GPU Support in gem5," in *3rd gem5 Users' Workshop*, June 2020.

[10] A. Voicu, "HIP CPU Runtime," https://github.com/ROCm-Developer-Tools/HIP-CPU, 2023.

[11] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S. Lee, and K. Skadron, "Rodinia: A Benchmark Suite for Heterogeneous Computing," in *IISWC*, Oct 2009, pp. 44–54.

[12] S. Che, J. W. Sheaffer, M. Boyer, L. G. Szafaryn, Liang Wang, and K. Skadron, "A Characterization of the Rodinia Benchmark Suite with Comparison to Contemporary CMP Workloads," in *IISWC*, 2010, pp. 1–11.

[13] S. Pati, S. Aga, M. D. Sinclair, and N. Jayasena, "SeqPoint: Identifying Representative Iterations of Sequence-based Neural Networks," in *IEEE International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS, August 2020.