Improving the Speed of gem5's GPU Regression Tests

James Braun and Matthew D. Sinclair
University of Wisconsin-Madison
jebraun3@wisc.edu sinclair@cs.wisc.edu

I. Motivation

In recent years, we have been enhancing and updating gem5's GPU support [1]. First, we have enhanced gem5's GPU support for ML workloads such that gem5 can now run [2]. Moreover, as part of this support, we created, validated, and released a Docker image that contains the proper software and libraries needed to run GCN3 and Vega GPU models in gem5. With this container, users can run the gem5 GPU model, as well as build the ROCm applications that they want to run in the GPU model, out of the box without needing to properly install the appropriate ROCm software and libraries [2], [3]. Additionally, we have updated gem5 to make it easier to reproduce results, including releasing support for a number of GPU workloads in gem5-resources [4] and enabling continuous integration testing on future GPU commits.

However, in an effort to provide sufficient coverage, the current testing support for GPU tests requires significant runtime both for the nightly and weekly regression tests. Currently most of these regression tests test the GPU SE mode support, since GPU FS mode support is still nascent. Unfortunately, much of this time is spent parsing input files to create arrays and other data structures that the GPU subsequently computes on. Although SE mode does not simulate the system calls needed to read these input files, nevertheless this still represents a significant overhead that increases runtime and prevents other tests (potentially providing additional coverage) from being run in that same timeframe. In an effort to address this, in the work we have been working on utilizing SE mode's avoiding modeling system calls to speed up the runtime of the GPU regression tests. Specifically, we redesign the input reading phase of these GPU tests to create and use mmap'd files for their input arrays (which SE mode completes all at once) instead of reading in the files entry by entry. In doing so, we see significant reductions in runtime of at least 29%.

II. Implementation & Methodology

Although both the mmap'd and non-mmap'd inputs for these benchmarks both use SE mode, we take advantage of the fact that mmap'd files can be completed in a single access because SE mode will have the underlying real hardware perform the mmap – and in the process read in the entire set of inputs at once. The alternative – using system calls like `fgets` – also uses the underlying hardware to read in the inputs, but requires that we perform many `fgets` (usually 1 such system call per line in the input file). However, since the applications gem5 supports (e.g., Pannotia [5]) often run on large grants with at least thousands of lines of inputs, performing these operations is still time consuming – sometimes even taking longer than the portion of the benchmark that simulates the GPU kernels. Thus, changing the benchmarks to use mmap'd inputs can significantly improve runtime. Moreover, since these benchmarks often run many different input files, hardcoding the values we want to read is not realistic. Instead, our redesigned benchmarks use input flags to create mmap'd files for a specific input file on real hardware and then use those mmap'd files with another flag when run in gem5. Thus, our solution is configurable and flexible regardless of inputs used.

To determine the efficacy of our approach we examined Pannotia's Floyd-Warshall (FW) benchmark [5] from gem5-resources [3] on gem5's Vega 10 GPU model. Thus far, we have only used the small `1k_128K.gr` input graph from the Pannotia repo. Then, we compared the runtime (using Linux's `time`) of gem5 using the baseline version of FW that reads in the input file one line at a time and our modified version that creates and then uses a mmap'd file instead for the inputs. Overall, our results show that using mmap'd files reduces FW's runtime by 41%, demonstrating the value in extending this approach to other benchmarks. Moreover, since the proportion of the simulation time spent reading input files is often proportional with input file size, we expect the gains for other, larger graphs will be even bigger.

III. Conclusion

Architectural simulation tools are highly important to the computer architecture community: both industry and academia rely on these tools to substantiate their findings. Given their widespread use, it is important that regressions be performed frequently to ensure new features do not affect the correctness of existing features. This introduces a new source of tension: ensuring sufficient coverage while not bloating runtime to unacceptable levels (e.g., too many tests to run overnight). By adding flexible, configurable support for the GPU SE mode tests to use faster mmap'd inputs instead of slowly reading in input files, our approach helps alleviate this tension: either more tests can be in the same amount of time (increasing coverage without increasing runtime) or the regression tests can be completed faster (reducing runtime for the current level of coverage). Although so far we only have examined a single GPU benchmark, we are currently adding similar support for the other GPU workloads in gem5-resources and integrating this support into both gem5-resources and the per-checkin, nightly, and weekly regression tests.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation grant ENS-1925485.

REFERENCES

[1] 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.

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

[3] B. R. Bruce, A. Akram, H. Nguyen, K. Roarty, M. Samani, M. Fariborz, T. Reddy, M. D. Sinclair, and J. Lowe-Power, "Enabling Reproducible and Agile Full-System Simulation," in *IEEE International Symposium on Performance Analysis of Systems and Software*, ser. ISPASS, 2021.

[4] gem5, "gem5 Resources," https://www.gem5.org/documentation/general_docs/gem5_resources/, 2020.

[5] S. Che, B. M. Beckmann, S. K. Reinhardt, and K. Skadron, "Pannotia: Understanding Irregular GPGPU Graph Applications," in *IEEE International Symposium on Workload Characterization*, ser. IISWC, Sept 2013, pp. 185–195.