Narrowing the GAP: Enhancing gem5's GPU Memory Bandwidth Accuracy

Yu Xia*, Vishnu Ramadas*, Matthew Poremba†, Matthew D. Sinclair*
*University of Wisconsin-Madison, †AMD Research
xia73@wisc.edu vramadas@wisc.edu Matthew.Poremba@amd.com sinclair@cs.wisc.edu

I. Introduction

Computer systems research heavily relies on simulation tools like gem5 to effectively prototype and validate new ideas. However, publicly available simulators struggle to accurately model systems as architectures evolve rapidly. This is a major issue because incorrect simulator models may lead researchers to draw misleading or even incorrect conclusions about their research prototypes from these simulators. Although this challenge pertains to many open source simulators [1]–[3], we focus on the widely used, open source gem5 simulator. In GAP we showed that gem5's GPGPU models [4] have significant correlation issues versus real hardware [5], [6]. GAP also improved the fidelity of gem5's AMDGPU model, particularly for cache access latencies and bandwidths. However, one critical issue remains: our microbenchmarks reveal 88% error in memory bandwidth between gem5's current model and corresponding real AMD GPUs. To narrow this gap, we examined recent patents and gem5's memory system bottlenecks, then made several improvements including: utilizing a redesigned HBM memory controller [7], enhancing TLB request coalescing, adding support for multiple page sizes, adding a page walk cache, and improving network bandwidth modeling. Collectively, these optimizations significantly improve gem5's GPU memory bandwidth by 3.8×: from 153 GB/s to 583 GB/s. Moreover, our address translation enhancements can be ported to other ISAs where similar support is also needed, improving gem5's MMU support [8].

II. Implementation & Methodology

To test the gem5 GPU models main memory bandwidth, we wrote a hand-tuned, inline-assembly GPU microbenchmark that stresses the main memory bandwidth. On a real AMD Vega 20 (Radeon VII) GPU, the microbenchmark obtains 1223 GB/s, close to the Vega 20's reported 1024 GB/s limit (it somewhat exceeds the reported max due to ROCm counter inaccuracies). Unfortunately, gem5's GPU model only reaches 153 GB/s for the same test (88% error).

Initially we suspected gem5's old, inaccurate High-Bandwidth Memory (HBM) model was limiting the maximum memory bandwidth of the GPU model. Thus, we modified the GPU model to support a new, higher fidelity HBM controller [7]. With this model we successfully obtain a peak bandwidth of 1024 GB/s when directly connecting traffic generators. However, when using it with gem5's GPU model our test only achieved 166 GB/s, hinting at additional bottlenecks.

We systematically examined the memory subsystem to locate these bottleneck(s). One major inefficiency was that modern AMD GPUs support different page sizes to reduce address translation overhead [9], [10]. Specifically, most GPU memory accesses use 2MB pages. Unfortunately, gem5's GPU TLB model only supported 4KB pages – degrading performance by requiring multiple entries when scanning a 2MB page. Thus, we added support for multiple page sizes in gem5's GPU TLB models. Our code is generic such that any arbitrary page size can be used. However, this change did not immediately help. Before an address translation completes, its page size is unknown. As a result, multiple translations for 4KB pages within the same larger (e.g. 2MB) page are often sent concurrently to the TLBs, wasting bandwidth.

Accordingly we redesigned the GPU TLB coalescer to coalesce address translation requests if they fall within the same larger (e.g. 2MB) region. On request return, if the page size is found to be less than that size, stalled requests to other pages are issued. This reduces redundant translation queries for identical pages, minimizing contention and improving memory throughput to 302 GB/s.

On a last-level TLB miss, the system performs a page table walk, which incurs multiple expensive memory accesses due to the hierarchical page table structure. To reduce this overhead, modern GPUs utilize a page walk cache (PWC) to store frequently used page table entries and reduce redundant memory accesses [11]. Accordingly, we added a PWC (default: 64-entries) to the GPU page table walker to improve translation latency. This improved memory bandwidth to 341 GB/s. We also increased the bandwidth of the interconnect network to better resemble real GPUs. This improved the memory bandwidth to 583 GB/s, significantly narrowing the gap between simulation and real hardware.

III. Conclusion

As hardware advances, simulator accuracy must evolve accordingly. One well known issue with gem5 is its MMU support. Accordingly, this work demonstrates how to reduce this deficit in gem5's GPU model through four optimizations: a more accurate HBM memory controller, enhanced TLB coalescing for large pages, a page walk cache, and increased network bandwidth. These improvements collectively achieve a 583 GB/s bandwidth (3.8×baseline), significantly narrowing the gap to real hardware. We are pushing all of these enhancements to the public gem5 repository. However, additional bottlenecks remain, which we are examining in ongoing work.

ACKNOWLEDGMENTS

This work is supported in part by the Semiconductor Research Corporation and by the National Science Foundation grant Frameworks-2311889.

REFERENCES

[1] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the Level of Abstraction for Scalable and Accurate Parallel Multi-core Simulation," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC. New York, NY, USA: Association for Computing Machinery, 2011. [Online]. Available: https://doi.org/10.1145/2063384.2063454

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

[3] Y. Sun, T. Baruah, S. A. Mojumder, S. Dong, X. Gong, S. Treadway, Y. Bao, S. Hance, C. McCardwell, V. Zhao, H. Barclay, A. K. Ziabari, Z. Chen, R. Ubal, J. L. Abellán, J. Kim, A. Joshi, and D. Kaeli, "MGPUSim: Enabling Multi-GPU Performance Modeling and Optimization," in *Proceedings of the 46th International Symposium on Computer Architecture*, ser. ISCA. New York, NY, USA: Association for Computing Machinery, 2019, p. 197–209. [Online]. Available: https://doi.org/10.1145/3307650.3322230

[4] 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 *IEEE International Symposium on High Performance Computer Architecture*, ser. HPCA. Washington, DC, USA: IEEE Computer Society, 2018, pp. 608–619.

[5] V. Ramadas, D. Kouchekinia, N. Osuji, and M. D. Sinclair, "Closing the Gap: Improving the Accuracy of gem5's GPU Models," in *5th gem5 Users' Workshop*. New York, NY, USA: Association for Computing Machinery, June 2023.

[6] V. Ramadas, D. Kouchekinia, and M. D. Sinclair, "Further Closing the GAP: Improving the Accuracy of gem5's GPU Models," in *6th Young Architects' Workshop*, ser. YArch. New York, NY, USA: Association for Computing Machinery, April 2024.

[7] A. Akram, M. Babaie, W. Wlsasser, and J. Lowe-Power, "Modeling HBM2 Memory Controller," in *4th gem5 Users' Workshop*, 2022.

[8] M. Mannino, "PIPT/VIPT caches and TLB lookup latency," https://github.com/orgs/gem5/discussions/2227.

[9] R. Ausavarungnirun, J. Landgraf, V. Miller, S. Ghose, J. Gandhi, C. J. Rossbach, and O. Mutlu, "Mosaic: A GPU Memory Manager with Application-Transparent Support for Multiple Page Sizes," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO. New York, NY, USA: Association for Computing Machinery, 2017, p. 136–150. [Online]. Available: https://doi.org/10.1145/3123939.3123975

[10] AMD, "ROCm: Open Platform For Development, Discovery and Education around GPU Computing," https://gpuopen.com/compute-product/rocm/, 2021.

[11] J. Power, M. D. Hill, and D. A. Wood, "Supporting x86-64 Address Translation for 100s of GPU Lanes," in *20th International Symposium on High Performance Computer Architecture*, ser. HPCA, 2014, pp. 568–578.