Analyzing the Benefits of More Complex Cache Replacement Policies in Moderns GPU LLCs

Jarvis Jia and Matthew D. Sinclair
University of Wisconsin-Madison
jia44@wisc.edu ; sinclair@cs.wisc.edu

I. Motivation

The gem5 simulator offers Classic and Ruby as two separate memory models for simulating on-chip caches. The Classic model, which originated from M5 [1], is a quick and simple option that allows for easy configuration, but only supports a basic MOESI coherence protocol. On the other hand, the Ruby model, which was developed by GEMS [2], is a more advanced and flexible option that can accurately simulate a wider range of cache coherence protocols and features [3], [4]. However, choosing between the two memory system models in gem5 is challenging for researchers as each has advantages and limitations which can be inconvenient. In particular, this has led to a bifurcation of effort where prior work has added replacement policies to Classic and Ruby in parallel – duplicating effort unnecessarily and preventing users from using a desired replacement policy if it is not implemented in the desired memory model (e.g., users could only use RRIP [5] in Classic).

Accordingly, we merged the cache replacement policies from Classic to Ruby, enabling users to use any of the replacement policies in either memory model. Gem5 currently has the capability to support 13 replacement policies, which can be used exchangeable within the Classic and Ruby cache models, including commonly used options like LRU, FIFO, PseudoLRU, and different types of RRIPs [6]. After combining the replacement policies for the Classic and Ruby cache models, we designed and integrated (into gem5's nightly regressions) multiple corner case tests to verify and ensure the continued correct functionality of these policies [7]. Through these tests, we identified and fixed several bugs [8] [9] to ensure that the replacement policies operate correctly. Finally, with the newly enabled and verified functionality, since there is limited information about how different replacement policies affects GPU performance, we decided to use gem5 to study these policies in a GPU context. Specifically, we study GPU L2 caches, since GPU L1 caches are often used to stream data through and thus are unlikely to be significantly impacted by replacement policy.

II. Methodology

We used an AMD Vega 10 as the target GPU [10], [11], and studied various L2 cache sizes (256 KB - 512 MB) for all 13 replacement policies. To compare the impact of changing L2 cache size and replacement policies, we examine total GPU cycles and GPU L2 hit rates. We tested the replacement policies with the Rodinia benchmarks [12], [13]. For example,

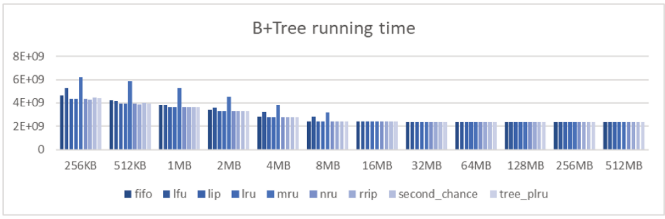

Fig. 1: B+Tree Performance across various LLC sizes and replacement policies.

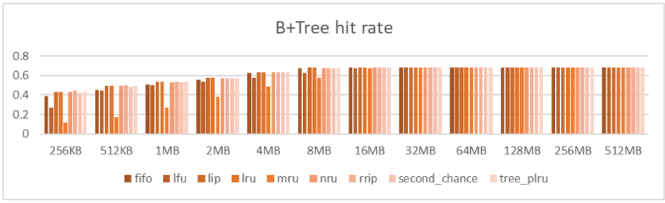

Fig. 2: B+Tree LLC hit rate across various LLC sizes and replacement policies.

Figures 1 and 2 show how B+tree is impacted. The LFU and MRU replacement policies performed the worst, while the RRIP performed slightly better than other policies. Moreover, B+ tree's overall performance improved and stabilized when the L2 cache size exceeded 8 MB – indicating that the working set now fits in the LLC. Other Rodinia applications show similar behavior, but they might be more or less affected by size and replacement policies. We suspect this is partly because by default the GPU configuration uses write-through caches, limiting reuse opportunities to situations with read-only data.

III. Conclusion

Cache replacement policy plays an important role for memory design and optimization. It has a significant impact on the cache system's hit rate and access latency, which has led to extensive efforts in both academia and industry to improve its effectiveness [14]. Despite the critical importance of cache replacement policies for efficient memory hierarchy design, there is limited research on their impact on GPUs. Our results show that, for write-through GPU LLC, caches replacement policy does not significantly impact overall results. However, we expect that write-back LLC caches and other, more cache sensitive GPU applications, will show larger benefits.

ACKNOWLEDGMENTS

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

REFERENCES

[1] N. Binkert, R. Dreslinski, L. Hsu, K. Lim, A. Saidi, and S. Reinhardt, "The M5 Simulator: Modeling Networked Systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, 2006, conference Name: IEEE Micro.

[2] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *ACM SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 92–99, 2005. [Online]. Available: https://dl.acm.org/doi/10.1145/1105734.1105747

[3] 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.

[4] gem5, "gem5: Adding cache to configuration script," https://www.gem5.org/documentation/learning_gem5/part1/cache_config/.

[5] A. Jaleel, K. B. Theobald, S. C. Steely, and J. Emer, "High Performance Cache Replacement Using Re-Reference Interval Prediction (RRIP)," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 60–71. [Online]. Available: https://doi.org/10.1145/1815961.1815971

[6] 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.

[7] tests: Add replacement policy tests (60389) · gerrit code review. [Online]. Available: https://gem5-review.googlesource.com/c/public/gem5/+/60389

[8] mem-ruby: Fix replacement policy updates in MI_example (62232) · gerrit code review. [Online]. Available: https://gem5-review.googlesource.com/c/public/gem5/+/62232/3

[9] mem-cache: Fix FIFO replacement (65952) · gerrit code review. [Online]. Available: https://gem5-review.googlesource.com/c/public/gem5/+/65952/7

[10] 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.

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

[12] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A Benchmark Suite for Heterogeneous Computing," in *IEEE International Symposium on Workload Characterization*, ser. IISWC, 2009, pp. 44–54.

[13] 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.

[14] M. Zahran, "Cache replacement policy revisited."