Characterizing the Impact of Graph-Processing Workloads on Modern CPU's Cache Hierarchy

Alexandre Valentin Jamet*†, Lluc Alvarez*†, Marc Casas*†
*Barcelona Supercomputing Center, Barcelona, Spain
†Universitat Politècnica de Catalunya, Barcelona, Spain
E-mail: {alexandre.jamet, lluc.alvarez, marc.casas}@bsc.es

Keywords—cache management, cache bypassing, big data, graph processing, workload evaluation, irregular workloads, micro-architecture

I. Extended Abstract

In recent years, graph-processing has become an essential class of workloads with applications in a rapidly growing number of fields. Graph-processing typically uses large input sets, often in multi-gigabyte scale, and data-dependent graph traversal methods exhibiting irregular memory access patterns. Recent work [1] demonstrates that, due to the highly irregular memory access patterns of data-dependent graph traversals, state-of-the-art graph-processing workloads spend up to 80 % of the total execution time waiting for memory accesses to be served by the DRAM. The vast disparity between the Last Level Cache (LLC) and main memory latencies is a problem that has been addressed for years in computer architecture. One of the prevailing approaches when it comes to mitigating this performance gap between modern CPUs and DRAM is cache replacement policies.

In this work, we characterize the challenges drawn by graph-processing workloads and evaluate the most relevant cache replacement policies.

A. Graph-processing Workloads

Graph-processing is a class of emerging workloads that, nowadays, can be found in various applications. Graph-processing can be found in both industry and academia, from social network analytics to web search engines and biomedical applications.

Graph-processing typically uses sparse data formats such as *Compressed Sparse Row/Column (CSR/CSC)* to manage a large amount of data. The CSR/CSC format is used to encode the graph adjacency matrix using two data structures: 1) the *Offset Array (OA)* and; 2) the *Neighbours Array (NA)*. Finally, additional data structures contain numerical data corresponding to a graph's vertices called *Property Arrays (PA)*.

Manipulating these sparse data structures often produces irregular memory access patterns. For example, when computing the *Sparse Matrix-Vector (SpMV)* multiplication $y = A \cdot x$, accesses to vector x are indexed by the column indices of matrix A, which are non-contiguous and constitute an irregular access stream. Graph-processing workloads also display highly irregular memory access patterns driven by operations like graph traversals that require visiting all the vertices V of a graph, that is, scanning the adjacency matrix rows following the graph's connectivity.

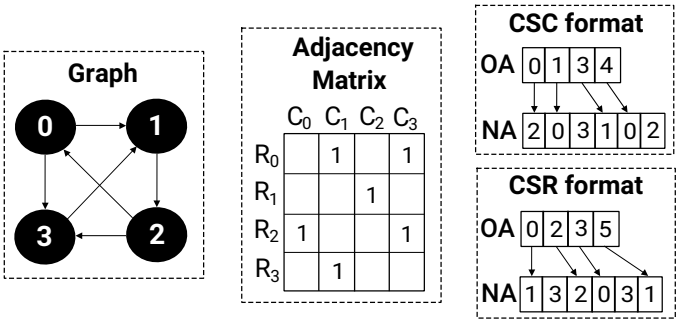

Fig. 1. Example of a graph representation in memory using the CSR/CSC formats.

Figure 1 shows an example graph along with its adjacency matrix and its representation using either the CSR or the CSC formats.

B. Cache Replacement Policies

For this work, we evaluate six of the most relevant replacement policies for the LLC. SRRIP, DRRIP [2] hahandleshe replacement process by predicting reuse distances. SHiP [3] leverages SRRIP and extends it with the addition of a PC feature. Hawkeye [4], Glider [5] a,nd MPPPB [6] further improve by making the addition of machine learning inspired techniques (*e.g.*: perceptron, SVM, etc.). Specifically, these advanced cache replacement policies leverage micro-architectural features based on bits extracted from the program counters and virtual/physical addresses to establish correlations and produce predictions.

C. Experimental Setup

Our evaluation considers ChampSim, a detailed trace-based simulator that models a Cascade Lake micro-architecture. The micro-architecture simulated has only one core, L1 instruction, and data cache of 32KB each, an L2 cache of 1MB, and an L3 cache of 1.375MB. The system also includes an 8GB main memory based on DDR4 SDRAM with a data rate of 2.933GT/s.

D. Experimental results

Figure 2 shows the MPKI rates in all three cache hierarchy levels for workloads of the GAP [7] benchmark suite. This figure shows that graph-processing workloads suffer from a large number of misses at all levels of the cache hierarchy.

The average MPKI rates of these workloads in the L1D, L2C, and LLC are 53.2, 44.2 and 41.8. We can further observe that a considerable portion (78.6%) of the accesses that trigger L1D misses also miss in the lower levels of the cache hierarchy and require a DRAM access.

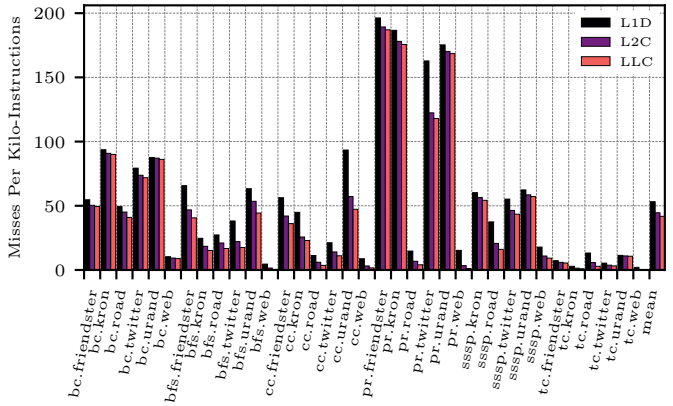

Fig. 2. Misses-Per-Kilo-Instruction (MPKI) across the different levels of the cache hierarchy triggered by graph-processing workloads.

Cache replacement policy is an obvious topic when it comes to improving the behavior of the cache hierarchy for certain types of workloads. As such, we evaluated the replacement policies presented in I-B to understand their impact on performance.

Figure 3 shows the geometric mean speed-up of the state-of-the-art cache replacement policies evaluated over the baseline LRU policy for various benchmark suites, including SPEC 2006 & 2017, along with the GAP workloads. The results show that the different policies can catch different kinds of access patterns and benefit different workloads.

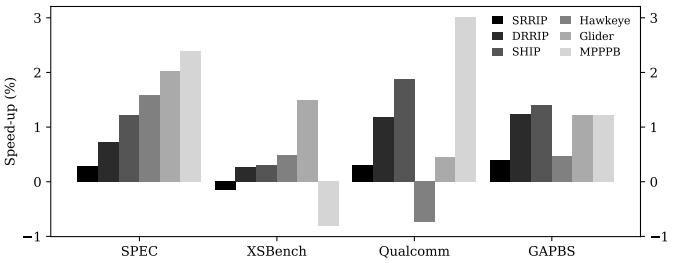

Fig. 3. Geometric mean speed-up over LRU of state-of-the-art LLC replacement policies for the different benchmark suites.

These results show that more complex replacement policies such as Hawkeye, Glider, and MPPPB have difficulties generalizing to benchmark suites beyond SPEC 2006 & 2017. This is due to the underlying assumptions on memory access patterns used to build these complex cache replacement policies. As shown in I-A, graph-processing is a prime example where the number of PC is very limited and where each PC maps to a very large number of addresses making correlations nearly impossible to establish.

E. Conclusion

Overall, this work highlights the poor ability of state-of-the-art cache replacement policies to leverage significant benefits against a baseline using an LRU policy for graph-processing workloads, despite the very high hardware complexity of such techniques. We show pieces of evidence that this bleak outlook stems from two factors: i) the very distinct nature of graph-processing workloads and; ii) the immense pressure these workloads create on the cache hierarchy.

II. Acknowledgment

This work has been published in Proceedings of the International Symposium on Workload Characterization (IISWC), 2020 [8].

References

[1] Y. Zhang, V. Kiriansky, C. Mendis, S. Amarasinghe, and M. Zaharia, "Making caches work for graph analytics," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 293–302.

[2] A. Jaleel, K. B. Theobald, S. C. S. Jr, and J. Emer, "High performance cache replacement using re-reference interval prediction (RRIP)," in *Proceedings of the 37th annual international symposium on Computer architecture*, vol. ISCA'10. IEEE, pp. 60–71. [Online]. Available: https://dl.acm.org/citation.cfm?doid=1815961.1815971

[3] C.-J. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely, and J. Emer, "SHiP: signature-based hit predictor for high performance caching," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture - MICRO-44 '11*. ACM Press, p. 430. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2155620.2155671

[4] A. Jain and C. Lin, "Back to the future: Leveraging belady's algorithm for improved cache replacement," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. IEEE, pp. 78–89. [Online]. Available: https://dl.acm.org/citation.cfm?doid=3007787.3001146

[5] Z. Shi, X. Huang, A. Jain, and C. Lin, "Applying deep learning to the cache replacement problem," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, pp. 413–425. [Online]. Available: https://dl.acm.org/doi/10.1145/3352460.3358319

[6] D. A. Jiménez and E. Teran, "Multiperspective reuse prediction," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture - MICRO-50 '17*, ser. MICRO '17. IEEE, pp. 436–448. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3123939.3123942

[7] S. Beamer, K. Asanović, and D. Patterson, "The GAP benchmark suite." [Online]. Available: http://arxiv.org/abs/1508.03619

[8] A. V. Jamet, L. Alvarez, D. A. Jiménez, and M. Casas, "Characterizing the impact of last-level cache replacement policies on big-data workloads," in *2020 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 134–144. [Online]. Available: https://upcommons.upc.edu/bitstream/handle/2117/343622/IISWC20-paper.pdf?sequence=1

Alexandre Valentin Jamet studied two years of Higher School Preparatory Classes with a Physics and Engineering Sciences major at LGT Baimbridge, Guadeloupe. In the following years, he pursued his MSc degree in parallel with an Engineer Diploma from TELECOM Nancy with a major in Embedded Computing. He concluded his studies in Nancy in 2018. Since 2018, he has been a Ph.D. candidate at the Computer Architecture departments of Barcelona Supercomputing Center (BSC) and Universitat Politècnica de Catalunya (UPC), Spain.