Investigating Memory Prefetcher Performance over Parallel Applications: From Real to Simulated

Valéria S. Girelli*[1], Francis B. Moreira†, Matheus S. Serpa‡, Danilo C. Santos§[1], Philippe O. A. Navaux‡
*Barcelona Supercomputing Center (BSC) - Barcelona, Spain
†Department of Informatics, Federal University of Paraná (UFPR) - Curitiba, Brazil
§Grenoble Informatics Laboratory, University Grenoble-Alpes (UGA) - Grenoble, France
‡Informatics Institute, Federal University of Rio Grande do Sul (UFRGS) - Porto Alegre, Brazil
E-mail: vsoldera@bsc.es, fbmoreira@inf.ufpr.br, danilo.carastan-dos-santos@inria.fr, {msserpa, navaux}@inf.ufrgs.br,

Keywords—Prefetching, memory system, High-performance computing.

I. Extended Abstract

In recent years, there have been significant advances in the performance of processors, exemplified by the reduction of transistor size and the increase in the number of cores in a processor. Conversely, the memory subsystem did not advance as significantly as processors, not being able to deliver data at the required rate, and creating what is known as the memory wall [1]. An example of a technology used to mitigate the memory latency is the prefetcher, a technique that identifies access patterns from each core, creates speculative memory requests, and fetches data that can be potentially useful to the cache beforehand.

In High-Performance Computing (HPC) systems, many other problems arise with parallelism. Since HPC applications are highly parallel, with many threads communicating with one another mainly through shared memory, it becomes necessary to keep data coherence in the several cache levels. Moreover, the memory interactions among different threads may also unpredictably change the data path through the memory hierarchy. When considering the memory hierarchy complexity along with prefetcher action, the behavior of the processor's memory subsystem reaches a new level of complexity.

In this work, we seek to shed light on how the prefetcher affects the processing performance of parallel HPC applications, and how accurately state-of-the-art multicore architecture simulators are simulating the execution of such applications, with and without prefetcher. We identify that an L2 cache prefetcher is more efficient in comparison with an L1 prefetcher, since avoiding excessive L3 cache accesses better contributes to performance, when comparing to accessing the L2 cache. Moreover, we show evidence that the prefetchers' contribution to performance is limited by the memory contention that emerges when the level of parallelism increases.

A. *Methodology and Experimental Environment*

We developed a careful experimental campaign, executing the Numerical Aerodynamic Simulation Parallel Benchmark (NPB) [2] using different levels of parallelism. Hardware counter information was collected using the command line tool PAPI [3], on an environment composed of an Intel Xeon Silver 4116 CPU, the Skylake microarchitecture [4]. The state-of-the-art simulators used were ZSim [5] and Sniper [6], configured on an approximation of the real machine. The conducted experiments were based on combinations of all the available prefetchers on both the real hardware and Sniper. A reproducible and open methodology was applied in our investigation, and all the materials are publicly available[1].

[1]Work done while being at the Informatics Institute, Federal University of Rio Grande do Sul (UFRGS) - Porto Alegre, Brazil

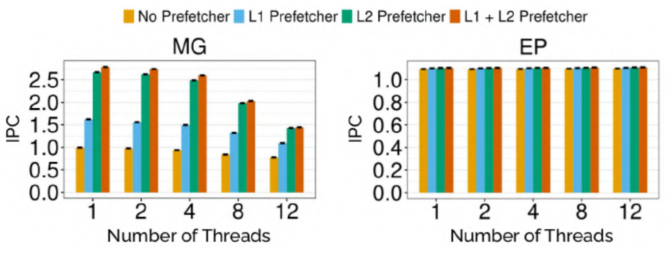

Fig. 1: IPC results for the real execution of the NPB applications with input class A.

B. *Investigating Current Architecture Prefetchers*

Figure 1 shows the instructions per cycle (IPC) for two NPB applications, MG and EP, running on the real hardware, considering different numbers of threads and prefetchers, and with prefetcher disabled as well. The behavior observed for MG is representative of the general behavior observed for the remaining applications, with the exception of the EP, which is known to not make much use of memory [2], therefore processor stalls due to memory access latency rarely occur during execution.

In general, we observe that the difference in performance from the configuration with only the L2 prefetchers to the configuration with all prefetchers enabled is not much large, with an average of 3.3% of IPC improvement. On Skylake, the L2 cache access latency is of 14 cycles, only 10 cycles higher than the L1 cache, while the L3 latency is measured to be approximately between 60 and 80 processor cycles, presenting a much higher overhead and a higher probability of stalling the processor execution [7]. Therefore, the main performance gains are obtained by the L2 prefetcher and the associated access streams it detects, avoiding long latency accesses to the L3 cache.

[1]https://gitlab.com/msserpa/prefetcher-ccpe

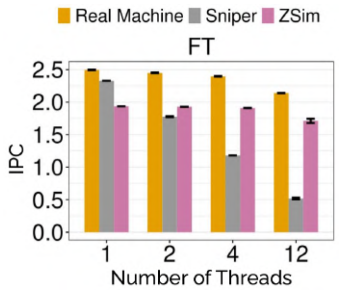

Fig. 2: IPCs with prefetcher disabled on ZSim and Sniper simulations and on the real execution.

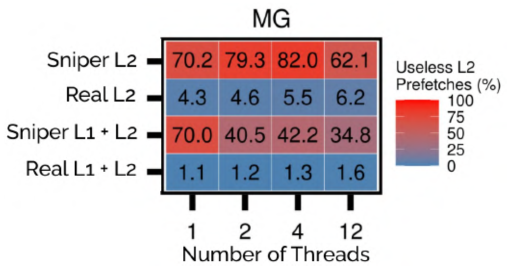

Fig. 3: Useless prefeteches performed by the simulation and by the real hardware, in ratio of the total number of prefetches.

The MG example in Figure 1 also shows a decrease in the IPC when we increase the number of threads. Parallel applications usually perform inter-thread communication that naturally increase in function of the total number of threads. With a large number of threads, the application memory accesses and the inter-thread communication increase, generating contention that hinders the prefetcher's impact over the performance. Therefore, in cases where the number of concurrent threads executed in a processor is very high, the performance benefit of memory prefetchers for applications that use intense memory and inter-thread communications would be negligible.

C. Investigating Prefetchers on Simulation

Figure 2 shows an example of the obtained IPCs when no prefetcher is used for the simulations and the real execution. For applications where communication and contention are more predominant, ZSim simulations can reach a 30% lower IPC when compared to the real execution, while Sniper can provide simulations with an IPC 426% lower. As a general trend, for NPB applications with communication and contention, ZSim tends to underestimate the contention effects, while Sniper tends to overestimate the contention effects as the number of threads in the simulation increases.

When comparing Sniper simulation with prefetchers enabled to the real execution with prefetchers as well, even though the number of prefetch requests issued is quite similar, it does not translate in similar estimations of performance. This may be due to the fact that the prefetches performed by Sniper are different in terms of usefullness, as demonstrated in Figure 3, where we show the mean percentage of prefetches that were not useful during the executions in the real machine and in the Sniper simulation.

D. Conclusions

The obscurity and confidentiality around the real implementation makes accurate prefetching models and algorithms impossible to be reproduced in simulators. This is exemplified by Skylake's non-inclusive L3 cache, which increases the memory system complexity and hinders the accurate simulation of NPB behavior with the Sniper's prefetchers.

On the real machine, we could observe that the use of both L1 and L2 prefetchers does not necessarily warrant significant performance gains, which is not intuitive. With only the L2 prefetcher, we obtained performance gains similar to when using both prefetchers, with the advantages of having more control over the experiments, faster simulation time, and less energy consumption due to the smaller number of prefetch requests being performed. Furthermore, as the parallelism and the inter-thread communication increase, so does the memory contention, becoming a large constraint to the performance.

II. Acknowledgment

This work has been published in the journal Concurrency and Computation: Practice and Experience (CCPE), 2021 [8]. The hardware resources applied in this work belong to the Parallel and Distributed Processing Group[2], Brazil.

References

[1] W. Wulf and S. McKee, "Hitting the memory wall: Implications of the obvious," *Computer Architecture News*, vol. 23, 01 1996.

[2] H. Jin *et al.*, "The openmp implementation of nas parallel benchmarks and its performance," NAS System Division, NASA Ames Research Center, Tech. Rep., 1999.

[3] D. Terpstra *et al.*, "Collecting performance data with papi-c." Tools for High Performance Computing. Springer, 2010, pp. 157–173.

[4] J. Doweck *et al.*, "Inside 6th-generation intel core: New microarchitecture code-named skylake," *IEEE Micro*, vol. 37, no. 2, p. 52–62, Mar. 2017. [Online]. Available: https://doi.org/10.1109/MM.2017.38

[5] D. Sanchez and C. Kozyrakis, "Zsim: Fast and accurate microarchitectural simulation of thousand-core systems," *SIGARCH Comput. Archit. News*, vol. 41, no. 3, p. 475–486, Jun. 2013. [Online]. Available: https://doi.org/10.1145/2508148.2485963

[6] T. E. Carlson *et al.*, "An evaluation of high-level mechanistic core models," *ACM Transactions on Architecture and Code Optimization (TACO)*, 2014.

[7] M. A. Z. Alves *et al.*, "Sinuca: A validated micro-architecture simulator," in *2015 IEEE 17th International Conference on High Performance Computing and Communications*, M. Qiu, Ed. IEEE, 2015, pp. 605–610.

[8] V. S. Girelli *et al.*, "Investigating memory prefetcher performance over parallel applications: From real to simulated," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 18, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.6207

Valéria S. Girelli is a Research Engineer at the Memory Systems group of the Barcelona Supercomputing Center (BSC), Spain. She received her BSc degree in Computer Science from the Federal University of Rio Grande do Sul (UFRGS), Brazil in 2021. Previously, she was a Computer Architecure Undergraduate Researcher at UFRGS, and a Researcher in Natural Language Processing at Universidade do Vale dos Sinos and Dell Brasil.

[2]http://gppd-hpc.inf.ufrgs.br/