Prefetcher's Impact Over Parallel Architecture Simulation Accuracy

Valéria Soldera Girelli[1], Francis B. Moreira[1], Matheus S. Serpa[1] and Philippe O. A. Navaux[1]
[1]Informactics Institute – Federal University of Rio Grande do Sul
Postal Code: 15.064 – 91.501-970, Porto Alegre – RS – Brasil
{vsgirelli, fbmoreira, msserpa, navaux}@inf.ufrgs.br

Abstract

In computer architecture research, the use of simulators is predominant. There are diverse approaches and implementations of modern simulators. However, validation studies lack a detailed analysis of parallel behavior in high-performance architecture simulators. We could observe that, due to the lack of prefetching simulation, the memory hierarchy statistics are inaccurate. Thereby, this study analyzes the impact of the prefetcher in the parallel simulation methodology used by ZSim.

1. Introduction

In computer architecture research, physical implementation and analysis are infeasible due to the complexity and high cost for manufacture. Consequently, architecture simulators are considered the primary mechanism to implement and evaluate a new idea in this research field [1]. In High-Performance Computing (HPC) systems, there are many other problems besides those inherent in the architecture. To develop and analyze new ideas that attenuate problems arising from parallelism, we need multicore architecture simulators that support parallel workloads. For example, systems with dozens of cores point out problems as the interference among the different threads and the communication costs among them. Thread interactions occur mainly through shared memory, with several threads accessing the same memory addresses, making necessary to keep data coherence in the several cache levels. Thereby, parallel simulators must provide an accurate implementation of cache coherence protocols and ensure data consistency through the memory hierarchy.

Another example is the prefetcher, a technique that identifies access patterns from each core and creates speculative memory requests. Thereby, the behavior of the processor's memory hierarchy receives a new level of complexity since simulations must consider information previously discarded, like cache line content [2], register values [3] or physical addresses [4].

In this paper, we analyze ZSim, a parallel architecture simulator that implements a parallel simulation methodology [5]. ZSim uses the Pin [6] tool to instrument the application, using dynamic binary translation to simulate the application's behavior. Likewise, the simulation of the parallel architecture and parallel workloads also is performed in parallel, decreasing the simulation time. By analyzing the implementation approach that ZSim applies for the parallel simulation, we observed that the method precludes the correct simulation of prefetchers in the coherence protocol.

The remainder of this paper is as follows. In Section 2, we detail the motivations of our study and the implementation approach of the simulator. Section 3 presents the configuration of the experiments and simulations. In Section 4, we present and discuss the obtained results, and in Section 5, we present our conclusions and future work.

2. Motivation and Related Work

Hardware companies use intellectual property law and hide information to avoid competition. Thus, it is difficult to obtain an ideal simulator which correctly implements all the processor's components and its architecture. In [7], the authors presented the difficulty of obtaining accurate information and its relevance when validating the SimpleScalar simulator [8]. By obtaining more information about the Alpha 21264 processor model, the authors were able to reduce the average experimental error with the SPEC-CPU 2000 workload from 36.7% to 18.2%. The authors demonstrated how the found features generate bottlenecks in different parts of the system than previously modeled in various studies. With this result, they could invalidate ideas from other articles that demonstrated performance gains where there was no bottleneck.

Among the several available simulators, we selected ZSim for an in-depth study due to its speed and accuracy, shown in its validation [9] and in Akram's work [10]. In Akram study, the simulators gem5 [11], Multi2Sim [12], MARSSx86 [13], PTLsim [14], Sniper [15], and ZSim [5] are evaluated. After thorough characterization of the simulators, the authors analyzed single and multicore workloads on each simulator. The target architecture was the In-

tel Haswell architecture [16]. The authors then highlighted the simulators' error sources, their sensitivity to different architectural parameters, and their relative error. Thus, they concluded that the lack of validation of simulators leads to poor accuracy and may result in invalid experiments due to erroneous conclusions. Akram's work does not use multi-thread workloads to verify the existence of cumulative relative error when increasing the number of threads.

2.1. Prefetcher

The memory available inside a processor has different levels. Private levels closer to the processor have less storage capacity, but accessing and transferring data from these levels is much more efficient. If a copy of the requested data is not in the L1 data cache, the first level of the processor's internal memory hierarchy, it forwards the request to the next level of cache memory, which repeats the same procedure. The last level cache (LLC), in turn, forwards the request to main memory. Thus, the cache levels closest to the processor should have data as relevant as possible at any given time in the execution of a program.

Prefetching is a technique to reduce data access latency. Based on the pattern of accesses generated by the processor, the prefetcher speculates on which is the next address to be requested and requests the data block in advance. Thereby, when the data block is indeed requested, it will already be in caches closer to the processor. Some of the patterns the prefetcher identifies are the strided [4] and the stream [17].

Figure 1 shows an example of the prefetcher identifying a stridden access pattern. The L2 cache level forwards requests to the LLC (indicated in Figure 1 as event **1**). The prefetcher, in turn, intercepts these requests (**2**) and identifies their access pattern. Based on the identified pattern, speculative accesses to the LLC are performed (**3**). These accesses are typical requests made by L2, so LLC forwards these data directly to L2. Thereby, when data previously requested by prefetch is needed, it is already at cache levels closer to the processor. Prefetchers are prevalent in today's architectures as one way to mitigate the main memory access bottleneck [18].

2.2. ZSim

ZSim implements the simulation in a two-phase method called Bound and Weave. In the Bound phase, a few thousand cycles are simulated, ignoring contention and using minimum latencies for all memory accesses. The simulator records the trace of memory accesses, including their passage through caches, invalidations, and evictions, among other details. In the Weave phase, the actual access latencies are determined based on the events of the previous phase. As the simulator identified the interactions between memory accesses in the first phase, it can efficiently simulate access timing while maintaining high accuracy.

The authors note that within a few thousand cycles, most concurrent access between different cores happens to separate cache lines. Thereby, regardless of the order of treatment of accesses, nothing changes in cache system consistency, and no request for ownership, invalidation, or update requests are required. However, when accesses are related, i.e., accessing the same cache line for communication, there is the need to maintain the coherence of different copies of data across different cores. For being considered rare, the **order** of these interactions is not modeled by ZSim, which may change the path of this data through the cache hierarchy, possibly impact the number of cycles, misses, coherence messages and even the path of prefetch requests. Therefore, ZSim does not simulate prefetchers in this model.

In its validation, ZSim uses the PARSEC benchmark [19] and only demonstrates that the speed-up is close to the obtained in real executions when varying the number of threads. In this study, we aim to evaluate the accuracy of the Bound and Weave simulation approach applied in the parallel simulation of several resources and threads.

3. Experimental Environment

In this section are specified the configurations of the environments used in the experiments.

3.1. Processor's Configuration

The *perf* [20] tool was used to get hardware counter values for all real executions. We evaluated performance and statistics of the real execution using 10 executions of each statistic, for each benchmark, on identical *draco* machines. These machines belong to the Parque Computacional de Alto Desempenho (PCAD)[1], which belongs to the GPPD - Parallel and Distributed Processing Group. For the simulations, we used configurations that approach the real machine, an Intel(R) Xeon(R) CPU E5-2640 v2, Ivy Bridge architecture [21], respecting the limitations of the simulator. Each simulation executed only once, and all statistics come from this same simulation.

In this article, we display a selection of statistics, which illustrates problems in ZSim memory hierarchy modeling. We chose statistics for execution cycles, L1 data cache accesses, and LLC accesses. Cycle statistics illustrate the highest number of cycles taken by the parallel threads, representing the time it took the application to execute. L1 cache access statistics represent the average among the caches of all threads, where we can see the data structures being split between the threads as we increase the application parallelism. The statistics of the LLC count accesses to all banks, since any application will use all banks. The standard deviation of the ten real executions is not shown in the images because it is smaller than 1%.

1 `http://gppd-hpc.inf.ufrgs.br/`

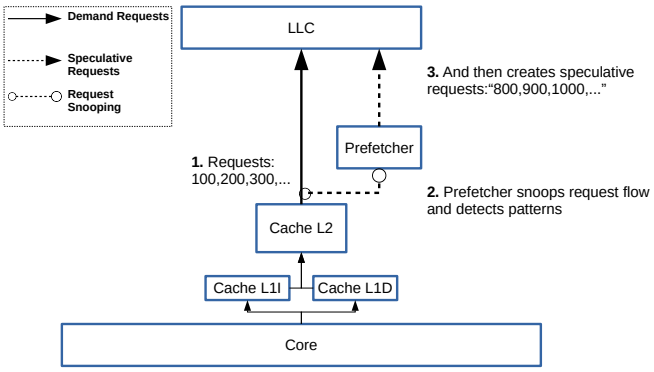


Figure 1: A simple prefetcher abstraction.

3.2. Benchmark's Configuration

We used the Numerical Aerodynamic Simulation Parallel (NPB) [22] benchmark to evaluate the scalability of the error regarding the number of threads, was used. The NPB benchmark is made up of ten parallel applications, of which we used nine. The applications used were BT, CG, EP, FT, IS, LU, MG, SP, and UA. The **DC** ("Data Cube") application was not used due to the large use of I / O, which is not modeled by the simulators.

4. Memory Hierarchy Behavior Analysis

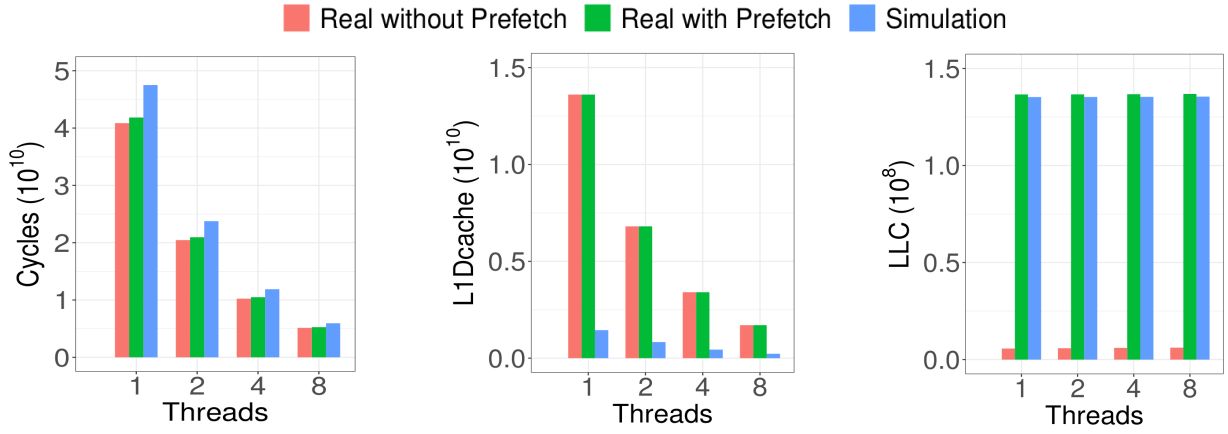
Since ZSim does not simulate the prefetching system, there could be changes in the number of cycles, misses, and LLC accesses. We analyzed the number of cycles and accesses to the data cache observed in the simulation and real execution with and without the prefetcher interference. We found that, for all benchmarks, the number of simulated cycles is always higher than in the real executions. Furthermore, all the simulations follow the speed-up trends observed in the real executions. Concerning L1 accesses, the number observed in the simulation is always smaller and also follows the trends of the real executions. LLC access statistics, on the other hand, show erratic behavior as a result of inaccuracy in cache hierarchy modeling.

In Figure 2, we can see from the EP application the lack of prefetch in ZSim. The simulator follows the real execution trends faithfully, but the percentage of accesses to the LLC avoided by the machine prefetcher is explicit in this benchmark, as shown in Figure 2c. This value represents a difference higher than 65000% in the number of accesses that reach the LLC.

The IS, LU, UA and MG applications present similar behaviors. Although the absolute number of accesses to the LLC is smaller in this applications if compared to the simulated in the EP application, all executions tend to increase in this metric as the number of threads increases. In the MG application, requests to the LLC of the real execution with prefetch increase so much that they reach the level of execution without prefetch. That may be due to inaccurate prefetchers or increase in communication. We could observe such an increase in almost all applications. As the number of threads increases, it becomes clear the decrease of the difference between the execution times of the simulation and the real execution without prefetch when compared to the execution with prefetch. Therefore, parallelism makes communication a bottleneck by increasing the number of requests to the LLC, which makes prefetchers unusable and limits the behavior in all cases analyzed. The only application where the observed behavior does not occur is EP, where memory and communication requirements are so small that its memory footprint fits in the LLC.

Some erratic behaviors were observed in some applications. The BT application presents similar behavior to that noted so far for first-level cache accesses and cycles. However, the ZSim simulation contains erratic behavior for one and two threads, where the number of hits to the LLC is much higher than the real execution without prefetching. In the CG application, the single-threaded real execution without prefetcher has a very high number of hits to the LLC. The observed standard deviation is less than 1%, which does not indicate a possible variance in the experiments as an explanation for such behavior.

Table 1 illustrates a summary of the differences in accesses to the last level cache between simulation and real execution with prefetch. We can observe the impact of prefetcher on memory access in the selected benchmarks. Simulations of cache memory have their effects altered by the lack of prefetcher, which could invalidate results from several articles. That would reduce the effect of cache replacement policy papers because prefetch already mitigates latency of main memory accesses [23]. On the other hand, the communication effect and the prefetch action itself increase contention in the LLC banks. Therefore, as the number of threads increases, the difference between the real execution time with prefetch decreases when compared to the real execution without prefetch and the simulation.



(a) Number of Cycles (b) Number of L1 Data Cache Accesses (c) Number of LLC Accesses

Figure 2: Comparison of real execution with prefetch, real execution without prefetch, and simulation of the EP application.

Threads:	**1**	**2**	**4**	**8**
CG	7%	8%	1%	3%
EP	2.600%	2.600%	2.150%	2.150%
FT	132%	112%	97%	79%
IS	130%	98%	64%	44%
MG	114%	99%	57%	00%
UA	86%	83%	79%	72%
BT	387%	212%	49%	34%
LU	471%	333%	305%	310%
SP	184%	164%	118%	83%

Table 1: Error resulting of the absence of prefetcher in ZSim.

5. Conclusions and Future Work

It is possible to observe some inconsistencies resulting from the approach taken by ZSim to model the accesses implemented by ZSim, as in BT and CG applications. The results show that the lack of prefetchers in the ZSim simulation can lead to significant differences in application behavior. ZSim's lack of prefetcher is a direct consequence of the Bound and Weave model, which makes it challenging to insert speculative accesses into the cache hierarchy. Future work will be investigating a possible implementation of prefetch in ZSim, given its popularity and fast simulation, which result in great practicality for conducting experiments.

Besides, the number of data cache accesses observed in the simulation is always lower than the real execution. Thus, it is also necessary to investigate the relationship of the simulator with memory requests. Specifically, it is necessary to investigate simulator instruction decoding, which requires updates to support new instructions, e.g., AVX (Advanced Vector Extensions) [24].

References

[1] K. Skadron, M. Martonosi, D. I. August, M. D. Hill, D. J. Lilja, and V. S. Pai, "Challenges in computer architecture evaluation," *Computer*, vol. 36, no. 8, pp. 30–36, 2003.

[2] R. Cooksey, S. Jourdan, and D. Grunwald, "A stateless, content-directed data prefetching mechanism," in *ACM SIGPLAN Notices*, vol. 37, pp. 279–290, ACM, 2002.

[3] K. J. Nesbit and J. E. Smith, "Data cache prefetching using a global history buffer," in *10th International Symposium on High Performance Computer Architecture (HPCA'04)*, pp. 96–96, IEEE, 2004.

[4] T.-F. Chen and J.-L. Baer, "Effective hardware-based data prefetching for high-performance processors," *IEEE transactions on computers*, vol. 44, no. 5, pp. 609–623, 1995.

[5] D. Sanchez and C. Kozyrakis, "Zsim: Fast and accurate microarchitectural simulation of thousand-core systems," in *ACM SIGARCH Computer architecture news*, vol. 41, pp. 475–486, ACM, 2013.

[6] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '05, (New York, NY, USA), pp. 190–200, ACM, 2005.

[7] R. Desikan, D. Burger, and S. W. Keckler, "Measuring experimental error in microprocessor simulation," in *Proceedings of the 28th annual international symposium on Computer architecture*, pp. 266–277, ACM, 2001.

[8] T. Austin, E. Larson, and D. Ernst, "Simplescalar: An infrastructure for computer system modeling," *Computer*, no. 2, pp. 59–67, 2002.

[9] D. Sanchez, "Zsim tutorial validation." http://zsim.csail.mit.edu/tutorial/slides/validation.pdf, 2016.

[10] A. Akram and L. Sawalha, "A survey of computer architecture simulation techniques and tools," *IEEE Access*, 2019.

[11] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[12] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli, "Multi2sim: a simulation framework for cpu-gpu computing," in *2012 21st International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pp. 335–344, IEEE, 2012.

[13] A. Patel, F. Afram, and K. Ghose, "Marss-x86: A qemu-based micro-architectural and systems simulator for x86 multicore processors," in *1st International Qemu Users' Forum*, pp. 29–30, 2011.

[14] M. T. Yourst, "Ptlsim: A cycle accurate full system x86-64 microarchitectural simulator," in *2007 IEEE International Symposium on Performance Analysis of Systems & Software*, pp. 23–34, IEEE, 2007.

[15] T. E. Carlson, W. Heirmant, and L. Eeckhout, "Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in *SC'11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, IEEE, 2011.

[16] P. Hammarlund, A. J. Martinez, A. A. Bajwa, D. L. Hill, E. Hallnor, H. Jiang, M. Dixon, M. Derr, M. Hunsaker, R. Kumar, *et al.*, "Haswell: The fourth-generation intel core processor," *IEEE Micro*, vol. 34, no. 2, pp. 6–20, 2014.

[17] H. Q. Le, W. J. Starke, J. S. Fields, F. P. O'Connell, D. Q. Nguyen, B. J. Ronchetti, W. M. Sauer, E. M. Schwarz, and M. T. Vaden, "Ibm power6 microarchitecture," *IBM Journal of Research and Development*, vol. 51, no. 6, pp. 639–662, 2007.

[18] C. J. Lee, O. Mutlu, V. Narasiman, and Y. N. Patt, "Prefetch-aware dram controllers," in *Proceedings of the 41st Annual IEEE/ACM international Symposium on Microarchitecture*, pp. 200–209, IEEE Computer Society, 2008.

[19] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The parsec benchmark suite: Characterization and architectural implications," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pp. 72–81, ACM, 2008.

[20] A. C. De Melo, "The new linux'perf'tools," in *Slides from Linux Kongress*, vol. 18, 2010.

[21] D. James, "Intel ivy bridge unveiled—the first commercial tri-gate, high-k, metal-gate cpu," in *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*, pp. 1–4, IEEE, 2012.

[22] H.-Q. Jin, M. Frumkin, and J. Yan, "The openmp implementation of nas parallel benchmarks and its performance," 1999.

[23] A. Jain and C. Lin, "Rethinking belady's algorithm to accommodate prefetching," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 110–123, IEEE, 2018.

[24] C. Lomont, "Introduction to intel advanced vector extensions," *Intel White Paper*, pp. 1–21, 2011.