Prefetching using Markov predictors - 25th Anniversary Retrospective

Doug Joseph
Samsung Electronics America

Dirk Grunwald
University of Colorado, Boulder

I. Motivation

Processor performance improvements arise from a combination of process or technology changes, programming model changes to more clearly express parallelism and architectural changes to exploit that ILP. All of these factors come with differing costs and computer architects seek to balance design changes changes to reduce costs. In the 1990's, it was clear that the disparity between CPU and memory speed was an impending issue and microarchitectural techniques such as multi-level caches, varying cache block sizes and cache organizations were important techniques to explore [8]. Later, the "Memory Wall" [19] made clear the disparity between CPU performance and memory speed, both in the instant and the future. Although fundamentally switching the computing model, as called out in the "Memory Wall" paper was possible, there were also opportunities to improve existing architectures through microarchitectural changes.

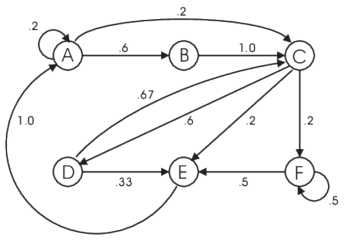
At the time, Doug Joseph was a Ph.D. student at the University of Colorado while also working as a member of the technical staff at IBM working on high performance systems. Dirk Grunwald had been working on improving instruction [11] and data [3] caches. Doug Joseph had an interest an AI and machine learning which is also reflected in his current position on architectural acceleration for deep learning. At the same time, Dirk Grunwald had been working with others on the application of machine-learning to branch prediction using decisions trees [4]. They decided Doug should pursue a thesis based on a preliminary idea that was the gensis of markov prefetching.

At the time, there had been extensive work on memory prefetchers that used arithmetic relations between memory addresses and were effective on structured workloads [6], [10], [13], [15]. Research on prefetching for unstructured workloads was less common [5], [12], [14], [20]. The Markov prefetcher that was the core of Doug Joseph's Ph.D. thesis is a continuing evolution of what has been called *correlation-based prefetching* [1], [5] which was similar to a method patented by Pomerene *et al.* [16].

In correlation prefetching, a memory reference A followed by a miss B would create an entry in a "shadow directory" that recorded that relationship. In [16], the shadow directory was off-chip and the reference stream focused on the miss references from the last-level cache (or references external to the processor). The key idea of Markov prefetching was to extend the simple $A \rightarrow B$ correlation to a full Markov model of prior memory reference behavior.

II. Elaboration

The example from the paper included the following Markov model



constructed from the references **A, B, C, D, C, E, A, C, F, F, E, A, A, B, C, D, E, A, B, C, D, C**. That example highlights most of the challenges that needed to be addressed - the Markov model has nodes with varying out-degree, the predictor could be large, there are both high-predictions and low-probability predictions.

The varying out-degree was addressed by parametric experimentation, but the size of the predictor as well as the number and quality of predictions that were made required more care and, importantly, an analysis method that could simplify performance comparison. There are three important metrics used to compare memory prefetchers: coverage (did you issue a prefetch), accuracy (was it used) and timeliness (did it arrive in time to improve execution). We decided to use a simulation based technique coupled with metrics normalized to the number of demand-references to evaluate different techniques. We used an in-order processor model driven by memory references from scientific and commercial traces that were gathered on an IBM Rs-6000 tracing system. Using whole-system traces was fortuitous – we had previously used whole-program tracing programs like ATOM [18] but that tool couldn't capture O/S interactions and our analysis found that including O/S behavior was very important to understand why we saw improvement in commercial workloads. This allowed us to experiment with different Markov predictor replacement policies and methods to record and then prioritize higher probability prefetch references.

Because the Markov predictor table was quite large (1MByte) an important part of our evaluation was using resource-equivalent predictors to dispel the notion that the large tables were not contributing. For example, we compared a 2MB cache combined with a 1MB predictor table to a 4MB cache organization.

III. Evolution

There were a number of future avenues we wanted to explore. At the time, speculative out-of-order processors were just becoming commercially available with the PentiumPro as the most well-known example. In theory, innovations such as speculative loads coupled with a large number of large miss-status holding registers (MSHR's) could remove some of the near-term miss references. With the introduction of SimpleScalar [2], simulator tooling had improved sufficiently to enable such analysis although without O/S references.

There has been a tremendous amount of innovation in memory prefetching since that the Markov prefecter work. In an effort to address irregular "pointer-chasing" code, Dirk Grunwald worked with another Ph.D. student, Robert Cooksey, on Content-Based prefetching [7] modeled after conservative garbage collection, that prefetches ”likely” virtual addresses observed in memory references. Here the goal was to goal was to capture large portions of the Markov structure using the program data directly.

More recently, machine-learning has been applied to all aspects of computer system design as demonstrated in the ISCA ML architecture systems workshop, including the ML prefetching competition [9]. Systems like Voyager [17] have adopted more sophisticated prediction methods such as LSTM's but also novel methods to reduce the needed state, such as their 2-level prediction structure.

IV. Futures

The memory wall still looms large for computer architecture. At some point, methods like prefetching will reach their achievable limit and alternate architectural mechanisms will be needed, which is why memory-intensive computing has been a ripe area of research and will likely remain so in the coming decades.

References

[1] J. Baer, "Dynamic improvements of locality in virtual memory systems," *IEEE Transactions on Software Engineering*, vol. 2, March 1976.

[2] D. Burger and T. M. Austin, "The simplescalar tool set, version 2.0," *SIGARCH Comput. Archit. News*, vol. 25, no. 3, p. 13–25, jun 1997. [Online]. Available: https://doi.org/10.1145/268806.268810

[3] B. Calder and D. Grunwald, "Next cache line and set prediction," *SIGARCH Comput. Archit. News*, vol. 23, no. 2, p. 287–296, may 1995. [Online]. Available: https://doi.org/10.1145/225830.224439

[4] B. Calder, D. Grunwald, D. Lindsay, J. Martin, M. Mozer, and B. Zorn, "Corpus-based static branch prediction," in *Proceedings of the ACM SIGPLAN 1995 Conference on Programming Language Design and Implementation*, ser. PLDI '95. New York, NY, USA: Association for Computing Machinery, 1995, p. 79–92. [Online]. Available: https://doi.org/10.1145/207110.207118

[5] M. Charney and A. Reeves, "Generalized correlation based hardware prefetching," Cornell University, Tech. Rep. EE-CEG-95-1, Feb 1995.

[6] T. Chen and J. Baer, "Reducing memory latency via non-blocking and prefetching caches," in *ASPLOS-V*, Oct 1992, pp. 51–61.

[7] R. Cooksey, S. Jourdan, and D. Grunwald, "A stateless, content-directed data prefetching mechanism," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS X. New York, NY, USA: Association for Computing Machinery, 2002, p. 279–290. [Online]. Available: https://doi.org/10.1145/605397.605427

[8] J. Hennessy and N. Jouppi, "Computer technology and architecture: an evolving interaction," *Computer*, vol. 24, no. 9, pp. 18–29, 1991.

[9] "Ml-based data prefetching competition," Intl. Symp. on Computer Architecture Workshop, 2021, https://sites.google.com/view/mlarchsys/isca-2021/ml-prefetching-competition.

[10] A. Klaiber and H. Levy, "An architecture for software controlled data prefetching," in *18th International Symposium on Computer Architecture*, May 1991.

[11] D. Lee, J.-L. Baer, B. Calder, and D. Grunwald, "Instruction cache fetch policies for speculative execution," in *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, ser. ISCA '95. New York, NY, USA: Association for Computing Machinery, 1995, p. 357–367. [Online]. Available: https://doi.org/10.1145/223982.224446

[12] M. Lipasti and et.al., "Spaid: Software prefetching in pointer and call intensive enviornments," in *Proceedings of 28th Annual International Symposium on Microarchitecture*, Nov 1995, pp. 231–236.

[13] T. Mowry, M. Lam, and A. Gupta, "Design and evaluation of a compiler algorithm for prefetching," in *ASPLOS-V*, Oct 1992, pp. 62–73.

[14] T. Ozawa and et.al., "Cache miss heuristics an preloading techniques for general-purpose programs," in *Proceedings of 28th Annual International Symposium on Microarchitecture*, Nov 1995, pp. 243–248.

[15] S. Palacharla and R. Kessler, "Evaluating stream buffers as a secondary cache replacement," in *21th Annual International Symposium on Computer Architecture*, April 1994, pp. 24–33.

[16] J. H. Pomerene, T. R. Puzak, R. N. Rechtschaffen, and F. J. Sparacio, "Prefetching system for a cache having a second directory for sequentially accessed blocks," US Patent US4 807 110A, Feb, 1989, available at https://patents.google.com/patent/US4807110.

[17] Z. Shi, A. Jain, K. Swersky, M. Hashemi, P. Ranganathan, and C. Lin, "A hierarchical neural model of data prefetching," in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 861–873. [Online]. Available: https://doi.org/10.1145/3445814.3446752

[18] A. Srivastava and A. Eustace, "Atom: A system for building customized program analysis tools," in *Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*, ser. PLDI '94. New York, NY, USA: Association for Computing Machinery, 1994, p. 196–205. [Online]. Available: https://doi.org/10.1145/178243.178260

[19] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *SIGARCH Comput. Archit. News*, vol. 23, no. 1, p. 20–24, mar 1995. [Online]. Available: https://doi.org/10.1145/216585.216588

[20] Z. Zhang and T. Torrellas, "Speeding up irregular applications in shared memory multiprocessors: Memory binding and group prefetching," in *22th Annual International Symposium on Computer Architecture*, June 1995, pp. 1–19.