RETROSPECTIVE:

Using Cache Memory to Reduce Processor-Memory Traffic

James R. Goodman

Computer Science Department
University of Wisconsin-Madison
goodman@cs.wisc.edu

While it has long been recognized that memory latency was a key parameter of performance, the impact of memory bandwidth (or its absence) has always been much harder to characterize. The complex relationship between latency and bandwidth is much better understood today than it was in 1982. Nevertheless, this relationship in the ever-varying context of "modern" system designs, created a fertile ground for studying a range of solutions to the same problem over many generations of computers: how to balance bandwidth and latency to provide a cost-effective, high-performance memory system [1,3].

This paper was an enthusiastic attempt by an assistant professor — who had never had a paper accepted to ISCA — to establish credentials in the area of single-board computer design, an exciting and growing market at the time. The paper reflects the exuberance and naivete of the era, where collecting data was as simple as figuring out how to use the trace mode on a VAX to generate a trace, and writing a cache simulator to evaluate the effects. With those simple tools, we studied everything from cold start/warm start phenomena to sector caches and block sizes, to replacement algorithms. Along the way, it became apparent to me that processors were approaching the point where it was preferable to allow them to sit idle rather than trying to keep them busy by switching tasks frequently. This seemed obvious to me because of the high-bandwidth, burst traffic required immediately following a task switch. While this conclusion led us down some interesting paths, apparently including the first publication of a snooping cache algorithm, we are still waiting expectantly for people to stop worrying about idle processors.

In the 1970s I was involved in the design of several add-on memory systems for IBM mainframe computers. Working on these designs, I had developed an appreciation for the design of cache memory, and the difficulty of supporting multiple processors in the process. In 1979 I had the opportunity to participate in the specification of the Intel 80286. Microprocessors were just arriving at the point where they were "interesting" to a computer architect, and I spent a lot of time thinking about Multibus-based systems. The Multibus standard allowed for multiple processors to share a bus, and to share commonly accessible memory on the bus but generally accessed memory on their own board. Microprocessors were just beginning to outrun DRAM memories. In discussions of how to support cache memory for the 286 I began to think about the implications of a small, on-chip cache, and how it could support a multiprocessor, and the idea came up of watching the bus from within the 286 chip. However, the multiprocessor systems I had studied were brute-force invalidation systems consisting of only two processors. Every write from a processor was conveyed to the cache of the other processor, where it was treated the same as an I/O operation, i.e., the remote cache was always checked, and invalidated on a hit. This resulted in a lot of traffic to the cache, since a check was required for every write.

In March of 1982, Carl Amdahl, on a visit to Madison, pointed out that write-back caches could reduce the invalidation traffic substantially: once a cache had been cleansed of a line, further invalidation requests were unnecessary if it could be guaranteed that the line didn't find its way back into the cache. This realization led us to a key reason that snooping is effective: the same mechanism can be used to detect a write — causing invalidation and exclusive access — and a subsequent read — causing intervention to prevent the reading of stale data.

The concept of a snooping cache developed over an extended period, so that when all the pieces finally fit together, I failed to see it as an important advance. In fairness, at least two other groups discovered the idea of snooping independently: Thacker and McCreight working on Dragon [4] at Xerox Parc and Steve Frank [2], architect of the Synapse N+1. In fact, when I wrote the paper, I believed that the important contribution of the paper was the recognition that caches — which usually had much higher bandwidth on the memory side than on the processor side—could actually reduce traffic from memory instead of increase it.

All of our studies were conducted in the context of a Multibus system, though we never actually implemented the idea at Wisconsin. (I was involved with the design of the Balance system developed in 1983 at Sequent.) We recognized that the addition of a couple of extra bus signals would make a protocol much simpler to implement, as well as provide better performance. Because of concern for commodity parts (the Multibus marketplace), we confined our design to one that could work in an unmodified Multibus system. From that experience I came to the realization that limitations imposed by commodity parts have their place, but that these limitations should not be allowed to stifle the search for novel solutions.

It was always apparent to me that there were better algorithms available if one relaxed our self-imposed constraint of Multibus compatibility. I was unprepared for the flood of papers introducing a plethora of variations. Many years passed before I was convinced that the seemingly small differences among these algorithms were important. I believe that, to this day, the most underrated contribution to understanding of snooping caches was Paul Sweazy's work on Futurebus, and his insistence that there had to be a common framework for comparing the algorithms. This work, of course, ultimately resulted in the MOESI model [5].

Finally, it's important to mention that my use of the term "we" in the original paper was not the royal we. As a junior professor I stumbled into an incredibly supportive and nurturing environment at the University of Wisconsin, one that allowed me to focus early on the research side of my career. I'm especially indebted to Jim Smith and David DeWitt, who led by example, and to Larry Landweber for his vision, leadership and advice. As acknowledged in the original paper, Phil Vitale and Tom Doyle participated in many stimulating discussions, both before and after this paper was written. Tswen-Hwey Yang built a powerful VAX trace tool that we used for this and much later work. She left the university before this paper was completed, and alas I have not heard from her since. David Patterson also played a critical role as a mentor during this period, and was one of the first to appreciate the importance of this work. Ed Davidson and Al Despain also provided critical advice and feedback during this difficult period.

I also want to put in a plug for supporting "small science" in the way that the National Science Foundation does so well. Support from NSF, particularly John Lehmann and, later, Zeke Zalcstein, has been critical in allowing many of us to develop our research programs. NSF support was enormously helpful to me, and this work resulted from my first NSF grant.

References

[1] D.C. Burger, A. Kägi, and J.R. Goodman, "Memory Bandwidth Limitations of Future Microprocessors," *23rd International Symposium on Computer Architecture (ISCA-23)*, May 1996.

[2] S. J. Frank, "Tightly coupled multiprocessor system speeds memory-access times," *Electronics*, Vol. 57, No. 1, (January 12, 1984), pp. 164-169.

[3] J. R. Goodman, "Using Cache Memory to Reduce Processor-Memory Traffic," *Proc. 10th Annual Symposium on Computer Architecture (ISCA-10)*, (June 1983), pp. 124-131.

[4] E. McCreight, "The Dragon computer system: an early overview." TR, Xerox Corp., Sept. 1984.

[5] P. Sweazy, A.J. Smith, "A Class of Compatible Cache Consistency Protocols and Their Support by the IEEE Futurebus," *Proc. 13th Annual Symposium on Computer Architecture (ISCA-13)*, pp. 414-423, June 1986.