Retrospective: Memory Bandwidth Limitations of Future Microprocessors

Doug Burger[†] Alain Kägi[‡] James R. Goodman[⋆]
[†]Microsoft Research [‡]Lewis & Clark College [⋆]University of Wisconsin-Madison, Emeritus

I. Historical Context

It is an odd experience to re-read a paper you wrote 28 years ago (published 27 years ago) for the first time in decades. The field has changed so much, that the paper feels like an artifact from a distant era, and in many places seems naïve. But the length of time since publication gives us the distance to observe the paper in its proper historical context.

When we wrote this paper in 1995 (published in 1996), we were halfway through the period of "frequency inflation," when processor clock rates accelerated beyond historical trends, growing by approximately a factor of twenty in ten years (1990-2000). The transition from in-order to out-of-order single-chip CPUs (at the high end) was starting to happen, but was still several years from completion. Since most cores were mostly in order, and since clock frequencies were increasing rapidly, memory latency was looking to be a large and growing problem. This latency challenge was popularly referred to as the "memory wall" [7].

When we started the research, the challenges from growing memory latencies were clear. As we discussed this challenge, it became clear that it was possible to improve latency at the expense of extra memory traffic. That led us to consider how limited memory bandwidth was likely to be, *e.g.*, how much of a "bandwidth budget" did we have to spend on improving latency? To understand the interplay between latency and bandwidth, we needed a way to measure how much of average latency was raw, contention-free memory latency and how much was contention in the memory system. The need to quantify that partitioning led us to generate the model that partitioned memory latency into raw contention-free latency and bandwidth contention components.

That model in turn led us to think about which techniques would improve latency inherently, and which would improve latency at the cost of extra traffic. It turned out that many of those techniques improved contention-free latency by generating more traffic, which put pressure on the memory subsystem and processor package interfaces. That observation led us to extrapolate the growth in bandwidth requirements, and to see whether technology trends would accommodate that growth. From the vantage point at the time, it did not appear as if those trends could accommodate the growth in requirements.

Next, we decided to see how effective an "ideal" cache could be at reducing traffic, to understand how much better on-chip memory management could reduce memory traffic as computational ability outstripped the package's ability to feed it data. The "minimal traffic cache" was a theoretical construct that, for a given on-chip capacity, would generate the minimal possible off-chip traffic. While not realizable in practice due to its use of Laszlo Belady's famous MIN algorithm [1], that study showed that off-chip traffic could be reduced enormously, in some cases by over two orders of magnitude, if the on-chip memory, even as limited as it was at the time, could be better managed by the hardware and/or software.[1]

Finally, the mismatch we projected between package bandwidth and off-chip traffic requirements led us to predict that some set of four directions would materialize to address this problem. The four evolutions of processor+memory architecture we thought most likely, given the bandwidth scaling analysis, were:

1) Tighter integration of system DRAM and the processor die, initially in multi-die packages and then eventually on the same chip.
2) Emergence of production processing-in-memory (PIM) architectures, where some high-bandwidth functions would be moved onto the DRAM dies.
3) More complex and effective management of the on-chip memories, leveraging different access patterns and perhaps more software control.
4) Compression in the cache memories, allowing greater effective usage of the on-chip memories.

As is often the case the paper got several things right and several things wrong about how the future would unfold. The rapid growth of clock rates came to an end fewer than ten years after publication, capping the rapid growth in memory latencies. The end of Dennard scaling a few years later caused power to be a major problem as transistor counts grew. This shift made much larger on-chip memories ("dim silicon") desirable, as it was not possible to use the same fraction of logic for high-performance computation. Even as CPU core counts grew, particularly for task-based parallelism in data centers, the increased on-chip memory sizes and natural improvements in pin signaling speeds and counts allowed the processing/bandwidth ratios to remain balanced. This shift reduced the need for the most radical changes to processor+memory architectures.

[1]As an aside, one of us (Doug Burger) ended up collaborating at Microsoft with Christian Belady, the son of Laszlo Belady, co-authoring a US patent on wind-powered data centers.

The least disruptive approach for improving bandwidth (and latency) was the one that—perhaps unsurprisingly—focused on better cache management. Indeed, the research community has produced many great ideas for managing caches more effectively, including in multicore designs. These ideas included dead block prediction [3], non-uniform shared caches, sophisticated prefetching techniques, and advanced replacement policies such as Re-reference Interval Prediction (RRIP) [2] and Dynamic Insertion Policy (DIP) [6] (to name two of our favorites). At the time, it was not widely understood that most data in a cache are "dead" (will be evicted before being re-used), although Thomas Puzak made this observation in his 1985 doctoral dissertation [5] a decade earlier. Cache designs at the time did not have advanced techniques for reducing the dead space, but these improvements found ways to reduce the dead space, using more complex but effective mechanisms to improve hit rates and thus cut down on memory bandwidth requirements. We like to think that our paper added to the community's shared understanding of important memory system problems and thus contributed to those cache advancements in an indirect way.

The two other architectural directions that we predicted did not materialize—at least widely. Compression of the caches (so far) has not managed to become a widely used solution, as simpler solutions seemed to be sufficient. The complexity of managing variable compression outcomes in a memory that needs fast, high-bandwidth access (often with coherence) has thus far outweighed the improved storage benefits. However, excellent research continues in this space with improving results [4]. Processing in memory (PIM) architectures have also failed to emerge as a widespread solution to reducing memory overheads. Moving to PIM architectures would require significant changes to commodity technologies (DIMMs), and would demand the adoption of programming and computational models that worked well over a wide enough span of workloads to justify the added complexity and investment. While many researchers and some companies continue to investigate PIM solutions, we view their near-term adoption to be unlikely for commodity CPU workloads.

The central prediction that we made in the paper was also (mostly) incorrect. Given the trends, it seemed likely to us that processors and memory would merge into integrated chips. That merger clearly did not happen, for several reasons that are well understood given how the field evolved. High-performance logic and DRAM manufacturing processes continued to diverge, making the penalties (memory density or circuit speed) for integrating processing and memory higher than they had been. The slowing of performance gains, the emergence of power limits, the larger on-chip memory systems, and the increasing sophistication of those memory systems made a more incremental evolution sufficient. It did turn out that for bandwidth hungry workloads (as we pointed out) such as graphics, and now AI, co-packaging of processing chips and DRAM (as stacked high-bandwidth memory, or HBM) were necessary to provide sufficient bandwidth. Those workloads need more data per byte than CPU workloads with large caches, and thus the power issue did not preclude the need for in-package memory, unlike with CPUs.

II. Looking forward

The key contributions of the paper were a model for reasoning about raw latency versus bandwidth penalties—helping to reason about memory system performance—and a strong bound on how effectively on-chip memory could reduce off-chip traffic with the minimal traffic cache. Most of the predictions the paper made did not end up coming to pass, aside from increasingly effective cache designs and co-packaged memory for accelerators.

While we were writing this retrospective, it occurred to us that we are actually now back where we started, except with AI workloads rather than CPU workloads. For generative AI, particularly the token phase, bandwidth has become the central limiter. Although We do not expect on-die processor/system memory integration for AI, 3D stacking of processing and memory is the next logical step after HBM. More aggressive on-die integration may be desirable if the research community is able to design *in situ* learning algorithms that are more similar to biologically-inspired neural computing. In our view it is more likely that the algorithms shift to optimize for the processing and memory interfaces currently available while pushing hard on improved capabilities. It will be interesting to see if the relationship between processing and memory plays out the same way in the "inflationary AI era" as it did in the "inflationary CPU era." As the old saying goes: History does not repeat itself, but it rhymes.

References

[1] L. A. Belady, "A study of replacement algorithms for a virtual storage computer," *IBM Systems Journal*, vol. 5, no. 2, pp. 78–101, 1966.

[2] A. Jaleel, K. B. Theobald, S. C. Steely 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*. New York, NY, USA: ACM, 2010, pp. 60–71. [Online]. Available: https://doi.org/10.1145/1815961.1815971

[3] A.-C. Lai and B. Falsafi, "Dead block prediction & dead block correlating prefetchers," in *Proceedings of the 28th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2001, pp. 144–154. [Online]. Available: https://doi.org/10.1109/ISCA.2001.937443

[4] G. Pekhimenko, V. Seshadri, Y. Kim, H. Xin, O. Mutlu, P. B. Gibbons, M. A. Kozuch, and T. C. Mowry, "Linearly compressed pages: A low-complexity, low-latency main memory compression framework," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*. New York, NY, USA: ACM, 2013, pp. 172–184.

[5] T. R. Puzak, "Analysis of cache replacement-algorithms," Ph.D. dissertation, 1985, aAI8509594. [Online]. Available: https://scholarworks.umass.edu/dissertations/AAI8509594

[6] M. K. Qureshi, A. Jaleel, Y. N. Patt, S. C. Steely Jr., and J. Emer, "Adaptive insertion policies for high performance caching," in *Proceedings of the 34th Annual International Symposium on Computer Architecture*. New York, NY, USA: ACM, 2007, pp. 381–391. [Online]. Available: https://doi.org/10.1145/1250662.1250709

[7] W. A. Wulf and S. A. McKee, "Hitting the memory wall: Implications of the obvious," *ACM SIGARCH Computer Architecture News*, vol. 23, no. 1, pp. 20–24, 1995.