Graffiti on "The Memory Wall"

Eric E. Johnson
Parallel Architecture Research Lab
Klipsch School of Electrical and Computer Engineering
New Mexico State University
ejohnson@nmsu.edu

The paper "Hitting the Memory Wall: Implications of the Obvious" by Wulf and McKee in the March 1995 issue of *Computer Architecture News* presented an interesting discussion of the bounds on processor performance imposed by memory performance. This note further examines that issue.

As in the earlier paper, simple models are used, although with slightly different notation:

t_{avg}	the average memory access time *in units of the processor clock cycle*
t_0	the access time of the top level of the memory hierarchy (level-1 cache)
RPI	the average number of memory references for instructions, operands, and results made by a processor for each instruction executed
CPI	clock cycles per instruction completed
CPI_0	CPI when $t_{avg} = t_0$

Assuming a single processor port to memory, we have $CPI = CPI_0 + RPI\,(t_{avg} - t_0)$. (The situation is only slightly more complicated with multiple memory ports, as in a Harvard architecture.)

The relative speed of processors and main memory is reflected in the t_{avg} term. The average access time of a memory hierarchy is determined by the access times at the various levels of the hierarchy and the probabilities that requests propagate to those levels. When the same amount of time is required to check for the presence of requested information as to access that information, we have the following simple formula for average access time:

$$t_{avg} = t_0 + \sum_{i=1}^{N} \left[t_i \prod_{j=0}^{i-1} m_j \right]$$

where N is the number of levels in the hierarchy, t_i is the access time to level i, and m_j is the miss ratio at level j. Level 0 is the smallest and fastest memory (the "top" of the hierarchy). t_N is the access time of the lowest level, main memory in this case. Note that as the processor clock rate increases relative to memory speed, t_N, in units of the processor clock cycle time, will increase.

The authors' contention amounts to this: for any fixed set of miss ratios, as the relative speeds of processors and memory technology diverge, t_N will eventually grow to overwhelm all other terms in t_{avg}. Further increases in processor clock speed will only result in commensurate increases in CPI so that performance has effectively hit a wall. Using an estimate of 20% for RPI, the authors conclude that the performance wall occurs when t_{avg} reaches 5 CPI_0, which represents an approximate doubling of the effective CPI. While this isn't exactly a wall, such a "3 dB" point is a reasonable choice for the onset of serious degradation. Thus, the critical value of t_{avg} is

$$t_{crit} = t_0 + \frac{CPI_0}{RPI}$$

However, the 20% figure for RPI apparently does not include instruction fetches: when instructions are included, RPI is typically in the range of 1.2 to 1.4, unless multiple instructions are fetched in each cache access (e.g., PowerPC). When $t_0 = 1$ and $CPI_0 \approx RPI$, $t_{crit} \approx 2$ clocks, much worse than before.

When can we expect t_{avg} to cross this threshold? As usual, the answer depends upon the assumptions made about the evolution of technology. If we simply extrapolate processor performance improvements at 80% per year, with main memory improving at only 7% per year, and assume a single-level cache with a miss ratio of 1%, we may encounter this dire day in about five years. This model projects mainstream processor performance in the year 2000 to be roughly 2000 MIPS (a sustained completion rate of 4 instructions per clock at 500 MHz?), with DRAM access time a bit over 40 ns.

The foregoing analysis is based upon assumptions that may prove untenable, however, including unbounded growth of uniprocessor performance at 80% per year, and the use of a single level of cache. By employing straightforward architectural improvements, we should be able to avoid reaching the memory wall for a number of years (as noted by the authors). Such measures may prove sufficient to altogether avoid "hitting the wall" as physical limits will eventually slow the growth in uniprocessor performance.

- As the disparity between on-chip caches and main memory grows, *additional layers of cache* will be employed to reduce t_{avg}. In *large caches*, entire program working sets can co-reside with portions of the operating system. If techniques are employed to *eliminate cache flushes on context switches*, sufficiently large caches will incur only cold start misses, producing miss ratios that asymptotically approach zero for programs that run long enough to be significant to users.

  Even one second spent executing a SPEC-type program on a workstation (about 10^8 references) is often sufficient to drive the miss ratio in a multi-megabyte cache below 10^{-4}. Using this figure, and assuming that the relative disparity between CPU speeds and a level-2 cache grows as the square root of the main memory disparity, simply adding a level-2 cache delays reaching t_{crit} for an additional five to eight years for SPEC-type programs.

- *Wider memories* provide increased bandwidth that can be applied to reduce t_{avg}. Main memories a cache line wide can transfer a line in a single clock. Beyond the practical transfer width determined by CPU pin-count limitations, additional bandwidth in the main memory can be used for speculative prefetching and other techniques to further reduce t_N. Another approach, already in use, is to reduce *RPI* by fetching multiple adjacent instructions from cache in a single access.

- The long-term solution may draw from responses to today's vast disparity between CPU and disk speeds. We tolerate an occasional CPU stall, so long as it usually runs at full speed. Caching and so on can be added to reduce stalls until the resulting price/performance satisfies users.

Well before reaching the memory wall, though, increasing numbers of parallel CPUs are likely to be required if we seek to continue today's rate of performance increase. In parallel computers, local memory bandwidth can be scaled directly with the processor count. If individual processor performance is no longer increasing rapidly, the growth in the memory latency disparity will diminish, leaving inter-processor communication (including synchronization) as the ultimate bottleneck.

To summarize:

- The RPI figure used by Wulf and McKee was too low. However, using a better figure exacerbates the problem of growing disparity between processor performance and main memory latency.
- A number of factors mitigate the problem.
  - Miss ratios for long-running programs asymptotically approach zero if their full working sets can remain in cache.
  - Physical limitations on the technologies used to implement logic will probably slow the rate of growth in individual processor performance.
- The problem largely disappears if parallel CPUs rather than a faster CPU are employed to increase performance.