Memory Bandwidth and Machine Balance in Current High Performance Computers

John D. McCalpin
University of Delaware
mccalpin@udel.edu

Revised to September 19, 1995

Abstract

The ratio of cpu speed to memory speed in current high-performance computers is growing rapidly, with significant implications for the design and implementation of algorithms in scientific computing. I present the results of a broad survey of memory bandwidth and machine balance for a large variety current computers, including uniprocessors, vector processors, shared-memory systems, and distributed-memory systems. The results are analyzed in terms of the sustainable data transfer rates for uncached unit-stride vector operations for each machine, and for each class.

1 Introduction

It has been estimated that the cpu speed of the fastest available microprocessors is increasing at approximately 80% per year [1], while the speed of memory devices has been growing at only about 7% per year [5]. The ratio of the cpu to memory performance is thus also growing exponentially, suggesting the need for fundamental re-thinking of either the design of computer systems or the algorithms that scientific users employ on them [2], [9]. For example, 10 years ago, floating-point operations were considered quite expensive, often costing 10 times as much as an uncached memory reference. Today the situation is dramatically reversed, with the fastest current processors able to perform 200 or more floating-point operations in the time required to service a single cache miss. Because of this fundamental change in the balance of the underlying technology, this report presents a survey of the memory bandwidth and machine balance on a variety of currently available machines.

Interestingly, despite the large amount of academic research on improving the performance of cache-based systems (e.g., the review in [2]), almost all of the systems here (which represent most of the machines sold in the U.S.) are either vector machines or standard hierarchical memory machines. No pre-fetching, cache bypass, or other novel techniques are represented here, and of the machines tested, only a few of the newest entries have the ability to handle more than one outstanding cache miss request.

The sections here include a definition of Machine Balance, a discussion of the Stream Benchmark, and a Discussion of the implications of current trends for high performance computing.

2 Machine Balance

The concept of *machine balance* has been defined in a number of studies (e.g.,[3]) as a ratio of the number of memory operations per cpu cycle to the number of floating-point operations per cpu cycle for a particular processor.

This definition introduces a systematic bias into the results, because it does not take into account the true cost of memory accesses in most systems, for which cache miss penalties (and other forms of latency and contention) must be included. In contrast, the "peak floating ops/cycle" is not strongly biased (for data in registers), because extra latencies in floating-point operations are usually due only to floating-point exceptions, and we will assume that these are rare enough to be ignored.

Therefore, to attempt to overcome the systematic bias incurred by the use of this definition, the definition of machine balance used here defines the number of memory operations per cpu cycle in terms of the performance on long, uncached vector operands with unit stride.

```
            peak floating ops/cycle
balance = -------------------------
            sustained memory ops/cycle
```

A corresponding metric may be applied to computational kernels, comparing the floating-point work required with the number of memory references. This quantity is referred to as the "computational density" or "compute intensity" [4, 7, 6].

With this new definition, the "balance" can be interpreted as the number of FP operations that can be performed during the time for an "average" memory access. Note that "average" here indicates average across the elements of a cache line for the long vector operations. It would be foolish to claim too much applicability for this concept of "average", but it should give results that are representative of the performance of large, unit-stride vector codes. It is clearly not a "worst-case" definition, since it assumes that all of the data in the cache line will be used, but it is not a "best-case" definition, since it assumes that none of the data will be re-used.

Interestingly, information on sustainable memory bandwidth is not typically available from published vendor data (perhaps because the results are generally quite poor), and had to be measured directly for this project by use of the STREAM benchmark code.

The "peak floating-ops/cycle" is derived from the vendor literature, much of it by way of Dongarra's LINPACK benchmark report. Most current machines have a sustainable floating-point operations rate that is very close to the peak rate (provided that one is using data in registers), so this usage does not introduce a significant bias into the results. An alternative value, such as the LINPACK 1000 or LINPACK scalable result would be equally useful here, but would result in no qualitative changes to the conclusions.

3 The STREAM Benchmark

The memory bandwidth data was obtained by use of the STREAM benchmark code. STREAM is a synthetic benchmark, written in standard Fortran 77, which measures the performance of four long vector operations. These operations are:

```
------------------------------------------
                              per iteration:
name    kernel                 bytes  FLOPS
------------------------------------------
COPY:   a(i) = b(i)              16     0
SCALE:  a(i) = q*b(i)            16     1
SUM:    a(i) = b(i) + c(i)       24     1
TRIAD:  a(i) = b(i) + q*c(i)     24     2
------------------------------------------
```

These operations are intended to represent the elemental operations on which long-vector codes are based, and are specifically intended to eliminate the possibility of data re-use (either in registers or in cache). It should be noted that the last operation (TRIAD) is not the same as the BLAS 1 SAXPY kernel, because the output array is not the same as either of the input arrays. On machines with a write-allocate cache policy, the TRIAD operation requires an extra memory read operation to load the elements of the "a" vector into cache before they are over-written.

4 Results

The STREAM benchmark report is continually updated as new measurements are contributed — much in the style of Dongarra's LINPACK report. The most recent data values are available on the World Wide Web at `http://perelandra.cms.udel.edu/hpc/stream`, and/or by anonymous ftp at perelandra.cms.udel.edu in `/bench/stream/` and its subdirectories.

The STREAM benchmark raw results and simple derived quantities are divided into five tables

- Raw Results: provides the data rates in MB/s for each of the four kernels.
- Equivalent MFLOPS: using the conversion factors in the Stream Definition Table
- Machine Balance: uses the Stream Triad results for the sustainable bandwidth rate.
- Sources of Data: Most of the data in these tables has been provided by others, whose names are provided here along with the date that the information was sent to me. Complete e-mail logs of all of the information that has been sent to me are available at my anonymous ftp site in `/bench/stream_mail/`.

- Parallel Speedups: provides speedup ratios for the Copy and Triad operations for each of the parallel computers listed.

What is perhaps most interesting about the results is the poor sustainable memory bandwidth of the hierarchical memory machines. The actual values obtained from user code are often only a small fraction of the impressive "peak" bandwidth numbers stated by a number of vendors. Unfortunately, even "peak bandwidth" numbers are stated by so few vendors that it is not possible to do a comprehensive survey of the ratio of "peak" to "sustainable" bandwidth.

In order to make some sense of the diversity of the numbers, the machines have been divided up into various categories based on their memory system type. The four categories used here are

- Shared-memory
- Vector
- Distributed Memory
- Uniprocessor

The results are plotted in Fig. 1.

5 Discussion

The results in Fig. 1 show a remarkably clear distinction between the four memory categories:

- Shared-memory: poor balance, fair scalability, moderate performance.
- Vector: good balance, moderate scalability, high performance.
- Distributed Memory: fair balance, perfect scalability, high performance.
- Uniprocessor: fair to good balance, low to moderate performance.

5.1 Uniprocessor Results

On hierarchical memory machines, the key determinant of the sustainable memory bandwidth for a single cpu is the cache miss latency. In the last few years, the memory systems of cached machines have experienced significant shifts in the ratio of the relative cost of latency vs transfer time in the total cost of memory accesses, going from an approximately even split in the typical 20 MHz machines of 1990, to being strongly dominated by latency in the typical 100 MHz machine of 1995. This trend is especially strong in shared-memory machines, for which the cost of maintaining cache coherence is a significant contributor to the latency.

The results for the uniprocessor systems clearly show two strategies for optimizing the performance of the cache systems:

- The first strategy is to optimize for unit-stride accesses. The approach is exemplified by the IBM RS/6000 series, which uses long cache lines (64, 128, or 256 bytes) and has minimal latency. The models in that line which use the Power2 cpu have a further reduction in the effective latency because the two "Fixed-Point Units" in the cpu can each handle independent cache misses at the same time, thus overlapping their latencies with the latency and transfer time of the other unit.
- The second strategy is to minimize memory traffic due to unused data. In the limit, this would correspond to single-word cache lines. Given sufficient latency tolerance, this can be optimal ([8]), but since these machines do not have effective latency tolerance mechanisms, this approach is too inefficient for the important case of unit-stride accesses. The most common compromise used for this case is a 32 byte line size, as is used in the HP PA-RISC and DEC Alpha (21064) systems. While this approach is reasonably effective in low latency situations, there is minimal gain from short line sizes in high latency situations, since effective transfer rates are limited largely by latency rather than by a busy bus. In multiprocessor systems, this approach is perhaps more justifiable, since unnecessary bus traffic (due to overly long cache lines) will interfere with the other processors as well.

5.2 Shared Memory Results

It should be noted that all but one of the vector machines are shared-memory, and so manage to maintain their good balance, scalability, and performance despite the negative factors that reduce the performance of the hierarchical-memory shared-memory machines.

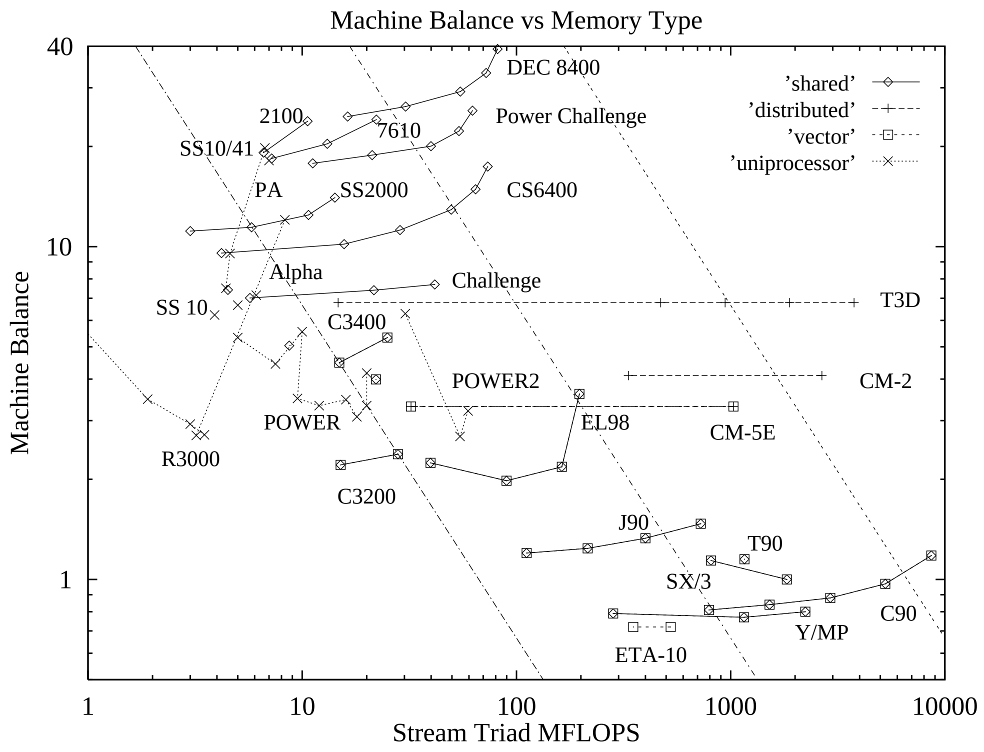

Figure 1: STREAM TRIAD MFLOPS and Machine Balance for a variety of recent and current computers. Each point represents one computer system, and connected points represent either multi-processor results from parallel systems, or different cpu speeds within the same cpu family for uniprocessor systems. The diagonal lines indicate Peak Performance of 0.1, 1.0, and 10.0 GFLOPS (from left to right).

The vector machines with the best performance characteristics do not employ hierarchical memory, thus greatly simplifying the coherence issue and associated latency penalty. In general, the vector machines are more expensive than the shared-memory, hierarchical-memory machines, but the larger configurations of the hierarchical-memory systems do overlap with the price range of the traditional supercomputers. When normalized for STREAM TRIAD performance, the traditional vector supercomputers are always more cost-effective than the shared-memory, hierarchical memory systems, as well as being marginally more cost-effective than the most cost-effective uniprocessors in the table.

Both the cached and vector shared-memory machines have absolute memory bandwidth limitations, which are visible in Fig. 1 as a sharp increase in the machine balance parameter as a the number of processors reaches a critical level. (This is not visible on most of the vector machines because they are deliberately limited in the number of processors supported in order to avoid this imbalance.)

Typically, the shared memory system (whether it is implemented via a bus, switch, crossbar, or other network) is either non-blocking between processors or allows split transactions, either of which allows multi-cpu parallelism to act as a latency tolerance mechanism. The "wall" hit by the machines when using many processors is a combination of latency, absolute bandwidth limitations (due to the limited number of DRAM banks), and bus/network/switch controller limitations. On vector machines, the limitation is usually bandwidth rather than latency for both single and multiple cpus.

Although extra processors can be used to provide latency tolerance in parallelized applications, this approach is both expensive and contributes to greatly increased (*i.e.*, poorer) machine balance. It seems likely that it would be more efficient to have special-purpose load/store units stall on cache misses, rather than entire cpus. This is the approach taken by the IBM Power 2 processor (with two fixed-point units to handle independent loads and stores), and by many new processors which, while having only a single load/store unit, support non-blocking caches (which can be considered a sort of "split transaction" model at the cache controller level). Most of the newest designs include non-blocking caches, such as the DEC 21164, HP PA-7200, and SGI/MIPS R10000 processors, the latter two of which are designed to handle four outstanding cache miss requests simultaneously.

It remains to be seen whether such “linear” solutions will be able to keep up with what is essentially an exponential increase in machine balance, or whether more fundamental architectural changes will be required in the very near future.

5.3 Trends in Hardware

Some historical trends in machine balance are represented in Table 1 for several major vendors.

The large size of the computer industry and the rapid turnover of each model of computer combine to make comprehensive surveys of more recent hardware difficult. Using the data acquired in this study, we will nevertheless make an attempt to examine trends in the performance characteristics of computer hardware, in the context of sustainable memory bandwidth measurements. Using a subset of the data representing various models of Cray, IBM, SGI, DEC, and HP computers, Fig. 2, shows the following quantities:

- Peak MFLOPS
- SPECfp92
- Sustainable Memory Bandwidth
- “Efficiency” defined as:

```
Sustained MWords/second
----------------------- * 100
      Peak MFLOPS
```

The specific models used in Fig. 2 are:

- HP: HP 9000/720, HP 9000/720, HP 9000/735, HP 9000/J200
- IBM: RS/6000 Models 250, 320, 950, 980, 990
- Silicon Graphics: Indigo R4000 (100 MHz), Challenge (150 MHz), Power Challenge
- DEC: 3000/500, 4000/710, 600-5/300
- Cray: EL-98, J916, Y/MP, C90, T90

In general, the machines are ordered such that either time or cost increases left to right within each vendor family. Although the relationships between the machines are not simple, we observe that for the machines tested from DEC and SGI, the peak cpu performance is increasing significantly faster than the sustainable memory bandwidth, thus resulting in decreasing “Efficiency”. In contrast, the machines from Cray and IBM show a relatively constant “Efficiency” despite increases in peak performance that are similar to those of the other set of vendors.

One might conclude from this that DEC and SGI have placed a relatively high priority on improving the performance of the SPECfp92 benchmark in their recent development, while IBM and Cray have favored a more “balanced” approach.

While vendor attention to realistic benchmarks is generally a “good thing”, in this case it may have acted to deflect attention from the difficult problem(s) of increasing memory bandwidth and maintaining machine balance, since the SPECfp92 benchmarks are relatively undemanding with respect to memory size and bandwidth requirements.

The new SGI/MIPS R10000 and HP PA-7200 and PA-8000 appear to be the beginnings of a deliberate counter-trend, with an advertising emphasis on improving “real-world” performance by larger factors than the improvement in SPECfp92 performance — in other words, by improving memory bandwidth. The only good example of this here is the HP curve. HP’s downward trend in efficiency is almost eliminated in their J-200 model based on the PA-7200 cpu. The DEC machines have also increased the memory bandwidth significantly with the 21164 cpu, but the peak performance has increase by an even greater amount, resulting in poorer machine balance.

Few SPEC95 results are currently available, but the initial indications suggest that SPEC95’s emphasis on larger jobs has resulted in significantly higher correlation of SPECfp95 results with memory bandwidth. An example is a comparison of the HP J-200 and HP-9000/755 (99 MHz version). These have approximate the same peak performance (200 and 198 MFLOPS, respectively). The J-200 has a significantly improved memory interface that results in double the sustainable memory bandwidth of the 755. The SPECfp92 ratio of the J-200 is 1.33 times that of the 755, while the SPECfp95 ratio is 1.57 times larger.

year	Machine	Memory Bandwidth MB/s	Peak FP rate MFLOPS	Balance
1978	VAX 11/780	4	0.4	0.8
1991	DEC 5000/200	28	10.0	2.9
1993	DEC 3000/500	100	150.0	12.0
1995	DEC 600-5/300	169	600.0	28.4
1980	IBM PC 8088/87	2	0.1	0.2
1992	IBM PC 486/DX2-66	33	10.0	2.4
1994	IBM PC Pentium/100	85	66.7	6.3
1989	SGI 4D/25	13	8.0	5.0
1992	SGI Crimson	62	50.0	6.5
1993	SGI Challenge	57	75.0	10.5
1994	SGI Power Challenge	135	300.0	17.8
1990	IBM RS/6000-320	62	40.0	5.2
1993	IBM RS/6000-580	276	83.2	2.4
1994	IBM RS/6000-990	800	286.0	2.9

Table 1: Historical changes in machine balance for several important computer vendors. Memory bandwidth and Peak FP rate are estimated for the VAX 11/780 and 8088-based IBM PC, and measured for all other cases.

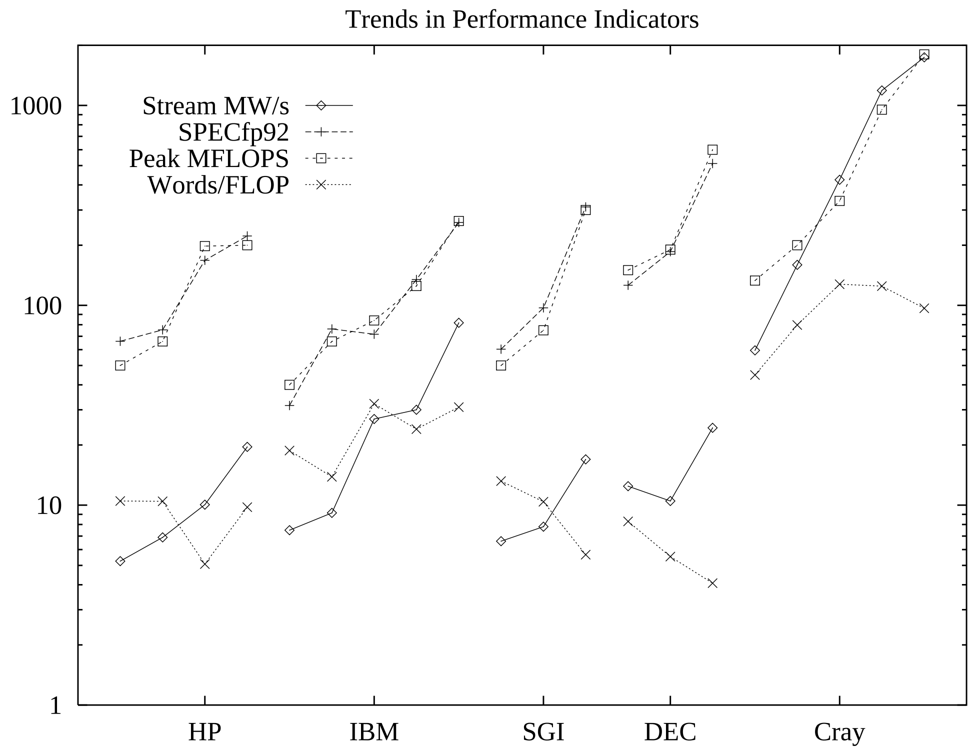

Figure 2: Trends in Peak MFLOPS, SPECfp92, Sustainable Memory Bandwidth (Mwords/s), and "Efficiency". Time and/or cost generally increase to the right within each vendor's listing.

6 Conclusions

A review of the sustainable memory bandwidth of a large variety of current and recent computer systems reveals strong systematic variations in the machine balance according to memory type. In particular, hierarchical-memory, shared-memory systems are generally strongly imbalanced with respect to memory bandwidth, typically being able to sustain only 3-10% of the memory bandwidth needed to keep the floating-point pipelines busy. Thus only algorithms which re-use data elements many times each can be expected to run efficiently. In contrast, vector shared-memory machines have very low machine balance parameters and are typically capable of performing approximately one load or store per floating-point operation. Of course, this capability is a strong requirement for good performance on the systems, since they typically have no cache to enable data re-use.

The recent shift in machine balance of current high performance computers strongly suggests that steps need to be taken soon to increase the memory bandwidth more rapidly. It is likely that merely increasing bus width and decreasing latency will not be adequate, given the rapid increase in cpu performance. What is needed instead is a set of more fundamental architectural changes to enable the systems to use information about data access patterns in order to effectively apply latency tolerance mechanisms (e.g. pre-fetch, block fetch, fetch with stride, cache bypass, etc.). At the same time, these systems should not preclude the use of "dumb" caches in the memory hierarchy when the memory access patterns are not visible to the compiler. This merger of the best features of "vector/flat memory" and "scalar/hierarchical memory" architectures should be a major subject of research in high performance computing in the closing years of this millenium.

References

[1] F. Baskett. Keynote address. International Symposium on Shared Memory Multiprocessing, April 1991.

[2] D.C. Burger, J. R. Goodman, and Alain Kagi. The declining effectiveness of dynamic caching for general-purpose microprocessors. Technical Report TR-1261, University of Wisconsin, Department of Computer Science, 1994.

[3] D. Callahan, J. Cocke, and K. Kennedy. Estimating interlock and improving balance for pipelined architectures. *Journal of Parallel and Distributed Computing*, 5:334:358, 1988.

[4] B.R. Carlile. Algorithms and design: The Cray APP shared-memory system. In *COMPCON '93*, pages 312–320, February 1993.

[5] J.L. Hennessy and D.A. Patterson. *Computer Architecture: a Quantitative Approach.* Morgan-Kaufman, San Mateo, CA, 1990.

[6] R. W. Hockney and C. R. Jesshope. *Parallel Computers.* Adam Hilger, Philadelphia, 1981. pp. 106–108.

[7] Roger Hockney. r_∞,$n_{1/2}$,$s_{1/2}$ measurements on the 2 CPU CRAY X/MP. *Parallel Computing*, 2:1–14, 1985.

[8] L.I. Kontothanassis, R.A. Sugumar, G.J. Faanes, J.E. Smith, and M.L. Scott. Cache performance in vector supercomputers. In *Proceedings, SuperComputing'94.* IEEE Computer Society Press, 1994.

[9] W.A. Wulf and S.A. McKee. Hitting the wall: Implications of the obvious. Technical Report Report No. CS-94-48, University of Virginia, Dept. of Computer Science, December 1994.