

PSC Science for Lunch

On Microprocessors, Memory Hierarchies and Amdahl's Law

David O'Neal

CSM Site Lead

Aeronautical Systems Center

**The National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign**

oneal@ncsa.edu

Introduction

- **Origins**

- Research was initiated in 1997 while working on a paper describing fundamental differences between Cray PVP and MPP architectures.
- The recent decommissioning of ASC's C916 coupled with the planned acquisition of a 64-processor SMP machine provided justification to investigate the proposed 4:1 ratio of DEC EV6 microprocessors to Cray C90 microprocessors.

- **Goals**

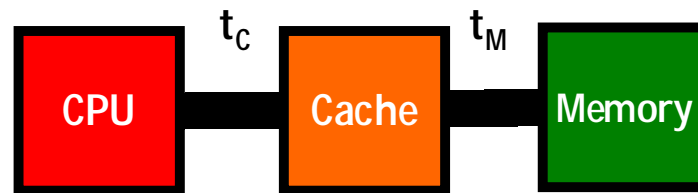
- The original idea was motivated by the absence of cache analysis tools for the Cray T3D and early T3E machines.
- The method must be formalized and validated with respect to its suitability for making throughput comparisons which in turn may be applied to purchasing issues or service unit conversion between contemporary platforms featuring hierarchical memory systems.

Amdahl's Law

- **Classical statement of Amdahl's Law (1967)**
 - If the serial component of an algorithm accounts for $1/S$ of its execution time, then the maximum speedup that can be achieved by executing the algorithm on a parallel processor is S .
- Let T be execution time written as a function of the processor count P and problem size N . Then for some given a sequential code that scales like $N + N^2$, many forms can be written. A few examples
 - $T = N + N^2 / P$ partition $O(N^2)$ part, replicate $O(N)$ part.
 - $T = (N + N^2) / P + 100$ partition all computations; add a fixed cost.
 - $T = (N + N^2) / P + P^2 / 2$ partition all computations; add $O(P^2)$ cost of remote memory access.

Amdahl's Law

- A variation of Amdahl's Law can be written in terms of the ratio of cycle counts required to complete data fetch operations from memory and cache
 - Representative hierarchy comprised of CPU registers, a cache structure, and a monolithic memory system. Define speedup as the ratio of fetch times for memory (t_M) and cache (t_C) resident data. Speedup is a measure of imbalance in this context.



$$S = \text{speedup} = t_M / t_C$$

- Basic idea is that if an entire code and all of its data were to suddenly fit into cache, the potential performance improvement would be S .

Relative Performance Function

- **Other definitions needed to represent the method include**

f_C fraction of time spent operating on cached data (cache efficiency)

f_M fraction of time spent operating on memory resident data, or $1-f_C$

T_N total time required to perform N operations

P_N performance rate in operations per unit time, or N / T_N

- **An expression for the time required to perform N fetch operations on register-sized data of arbitrary residence can be written immediately from the definitions**

$$T_N = N (f_C t_C + f_M t_M)$$

- **Substitution, rearrangement of terms, and normalization yield a relationship for estimating relative performance as a function of speedup and cache efficiency**

$$p_N = (S - f_C (S - 1))^{-1}$$

Relative Performance Function

- The value of the speedup parameter has a significant effect on the relationship between relative performance (p_N) and cache efficiency (f_C) as illustrated in Fig. 1.

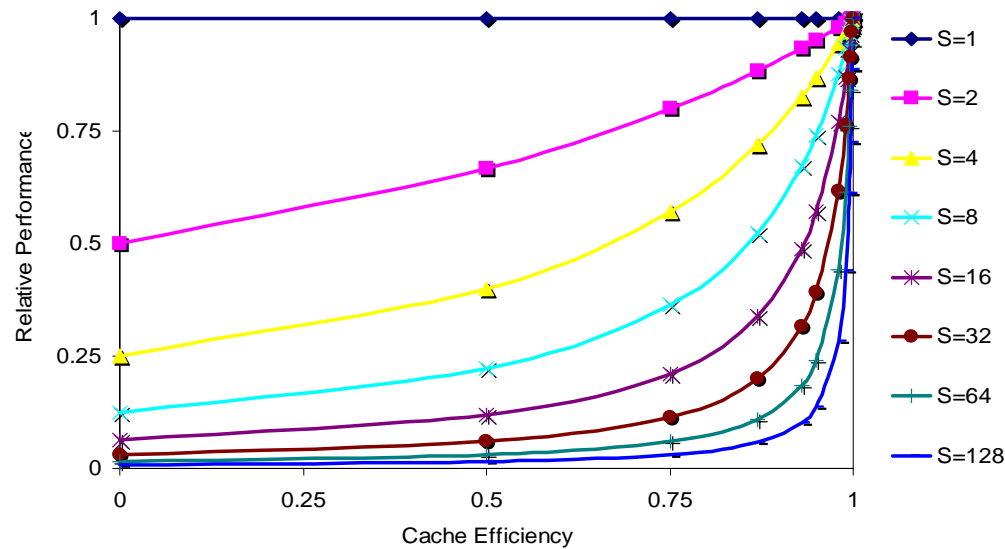


Figure 1. Relative performance as a function of cache efficiency for selected speedups

Cache Efficiency Function

- A related formula for cache efficiency can be written by solving the relative performance function for f_C

$$f_C = (S - 1/p_N) / (S - 1)$$

- The curves of Fig. 2 characterize cache efficiency as a function of relative performance (inverse of the relative performance function).

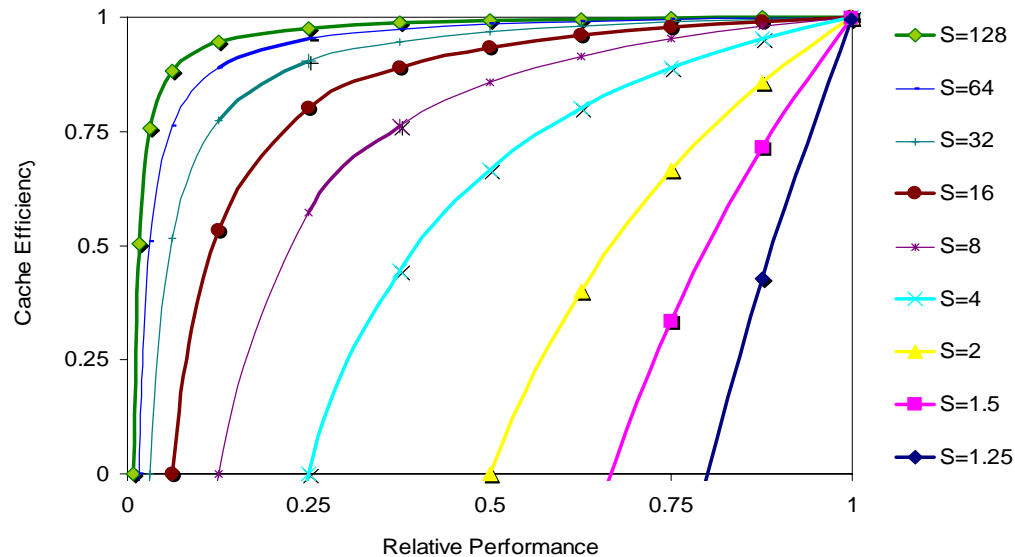


Figure 2. Cache efficiency as a function of relative performance for selected speedups

Speedup Function

- Finally, the curves presented in Fig. 3 provide further indication of the way in which speedup affects relative performance. If a speedup improvement ($S \rightarrow 1$) were to be implemented somehow, the least efficient codes would clearly benefit the most. In general, this suggests that it is easier to extract higher levels of relative performance from architectures characterized by lower speedup values. No stretch there - remember speedup is a measure of memory system imbalance

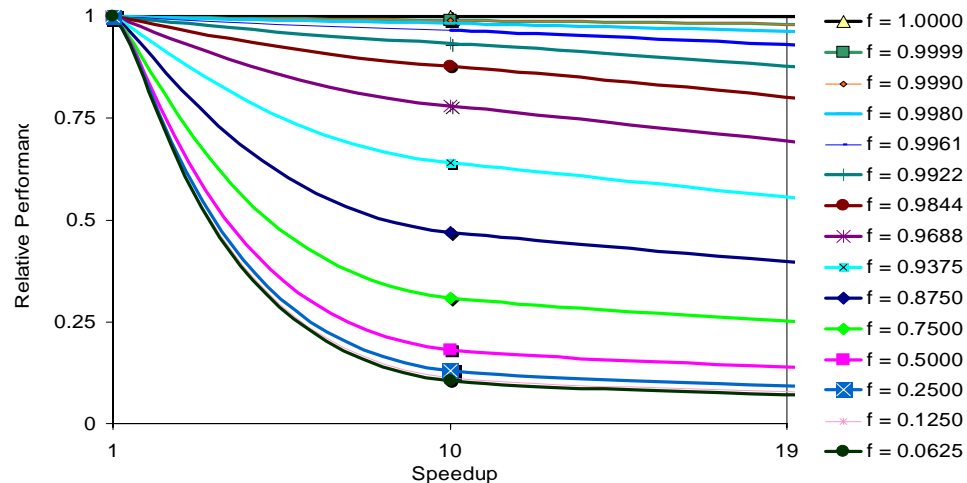
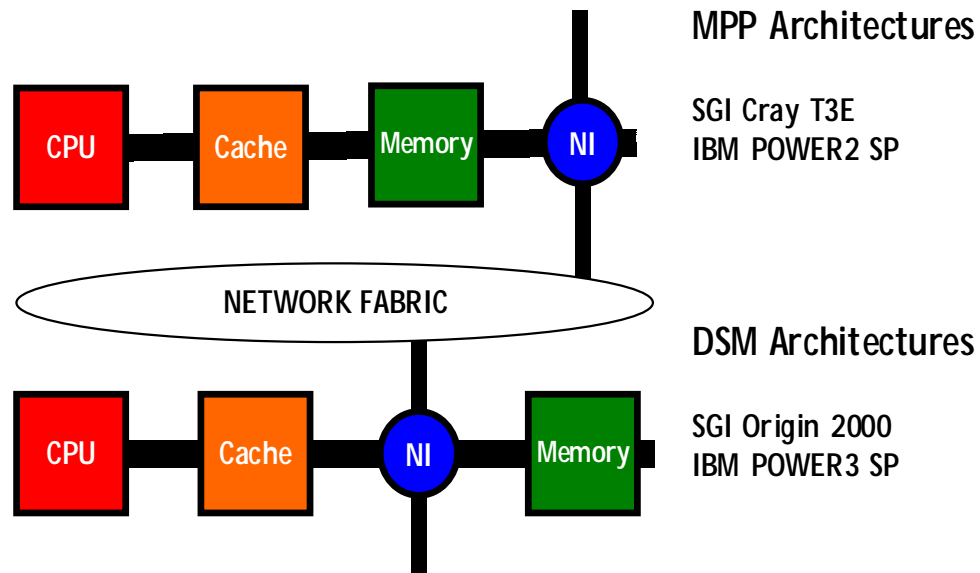


Figure 3. Relative performance as a function of speedup for selected cache efficiencies

Network Interface

- As shown in Fig. 4, some of the most popular distributed shared memory designs place their network interfaces between the processors and memory. The transfer rate across this device cannot be ignored.



Validation Table 1

- MIPS R10000 Design Characteristics**

Tools required to validate functions were available in the form of the `perfex` and `hinv` utilities featured by Silicon Graphics computing systems. We were also privy to accounts on similar systems featuring different CPU boards (195 MHz and 250 MHz) which allowed a basic throughput analysis to be completed as well.

| MIPS R10000 | CPU (MHz) | Regs-L1 (GB/s) | L1 | | L1-L2 (GB/s) | L2 | | L2 Clock (MHz) | L2-Mem (GB/s) | FPU | FPU mult add | | Peak (MFlop/s) |
|----------------|--------------|-------------------|-------------|--------------|-----------------|-------------|--------------|-------------------|------------------|-------------------|-----------------|----|-------------------|
| | | | Line (B) | Size (KB) | | Line (B) | Size (MB) | | | | | | |
| Rev 2.6 | 195 | 1.56 | 32 | 32 | 1.06 | 128 | 4 | 130 | 0.45 | R10010 Rev 0.0 | 1 | 1 | 390 |
| Rev 3.4 | 250 | 2.00 | 32 | 32 | 1.63 | 128 | 4 | 200 | 0.47 | R99999 Rev 0.0 | 1 | 1 | 500 |
| Δ | 28% | 28% | 0% | 0% | 54% | 0% | 0% | 54% | 4% | Δ | 0% | 0% | 28% |

Table 1. Physical characteristics and performance data for the R10000 microprocessor

Validation Table 2

- MIPS R10000 Performance Data**

In Table 2, a test value for cache efficiency is established and the corresponding relative and absolute performance figures are developed for comparison purposes. Values for t_C and t_M were derived from the transfer rates given in Table 1. The effect of increasing the clock rate without an accompanying increase in memory bandwidth is indicated by the increase in the speedup factor and the accompanying decrease in relative performance.

| MIPS R10000 | CPU (MHz) | Flops per cycle | Peak (MFlop/s) | t_C (cycles) | t_M (cycles) | S (speedup) | f_C (%) | P_N (%) | P_N (MFlop/s) |
|-------------|-----------|-----------------|----------------|----------------|----------------|-------------|-----------|-----------|-----------------|
| Rev 2.6 | 195 | 2 | 390 | 1.0 | 3.5 | 3.5 | 50% | 44.8% | 175 |
| Rev 3.4 | 250 | 2 | 500 | 1.0 | 4.3 | 4.3 | 50% | 38.1% | 190 |
| Δ | 28% | 0% | 28% | 0% | -23% | -23% | Δ | -15% | 9% |

Table 2. R10000 performance data (alternate units) and throughput comparison for $f_C = 0.5$

Validation Table 3

- MIPS R10000 (195 MHz) Benchmark Data**

Table 3 summarizes measurements and estimates for two simple benchmarking codes. The horner example evaluates a 12th degree polynomial. The vector triad code evaluates $a_i = ka_i + b_i$. Note that the peak rate associated with the triad benchmark reflects the fact that two loads must be performed to complete a multadd operation which corresponds to one flop per cycle.

| R10000 Rev 2.6 | Bench Applications | Memory Statistics (per fix) | | | | | | Estimated Error | | |
|-------------------|-----------------------|-----------------------------|------------------|--------|-------------------------------|-------------------|-------------------------------|-----------------|---------------------------------|--------------|
| | | L1 (hit rate) | L2 (hit rate) | Memory | Absolute (P _N) | Peak (MFlop/s) | Relative (p _N) | Speedup (S) | Efficiency (f _c) | Error (%) |
| N = 1E3 | horner | 0.9708 | - | 0.0292 | 365 | 390 | 0.9359 | 3.47 | 0.9723 | 0.1% |
| | vector triad | 0.8000 | - | 0.2000 | 132 | 195 | 0.6769 | 3.47 | 0.8068 | 0.7% |
| N = 1E6 | horner | 0.3360 | - | 0.6640 | 154 | 390 | 0.3949 | 3.47 | 0.3796 | 4.4% |
| | vector triad | 0.1520 | - | 0.8480 | 65 | 195 | 0.3333 | 3.47 | 0.1903 | 3.8% |

Table 3. 195 MHz R10000 benchmarking statistics and estimates

Validation Table 4

- MIPS R10000 (250 MHz) Benchmark Data**

In an effort to confirm the results shown in Table 3, a view of the faster R10000 chip is provided by Table 4. As expected, all of the estimates slightly exceed measured values. We implicitly build L2 cache effects into the L1 estimate by neglecting contributions of L2.

| R10000 Rev 3.4 | Bench Applications | Memory Statistics (per fix) | | | | | | Estimated Error | | |
|-------------------|-----------------------|-----------------------------|------------------|--------|-------------------------------|-------------------|-------------------------------|-----------------|---------------------------------|--------------|
| | | L1 (hit rate) | L2 (hit rate) | Memory | Absolute (P _N) | Peak (MFlop/s) | Relative (p _N) | Speedup (S) | Efficiency (f _c) | Error (%) |
| N = 1E3 | horner | 0.9740 | - | 0.0260 | 468 | 500 | 0.9360 | 4.26 | 0.9790 | 0.5% |
| | vector triad | 0.8050 | - | 0.1950 | 160 | 250 | 0.6400 | 4.26 | 0.8272 | 2.2% |
| N = 1E6 | horner | 0.4000 | - | 0.6000 | 177 | 500 | 0.3540 | 4.26 | 0.4394 | 3.9% |
| | vector triad | 0.1650 | - | 0.8350 | 70 | 250 | 0.2800 | 4.26 | 0.2100 | 4.5% |

Table 4. 250 MHz R10000 benchmarking statistics and estimates

Sensitivity

- An observation regarding sensitivity. Results presented in Tables 3 and 4 suggest that the method is more susceptible to error for lower values of f_C . The curves of Fig. 3 serve to confirm this. Note the marked difference in the rate of change for the curve $f_C=0.25$ (example) on the interval $S=[1,9]$ versus $S=[9,19]$.

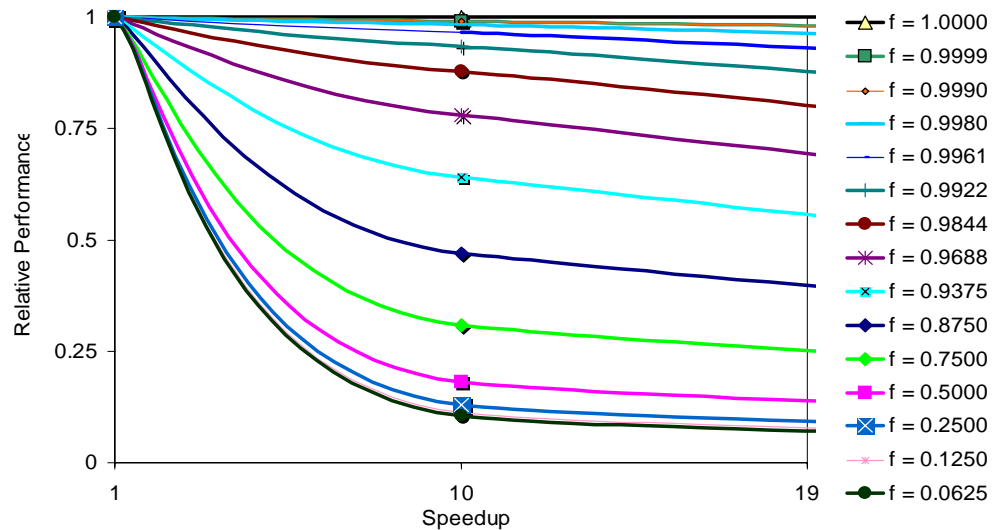


Figure 3. Relative performance as a function of speedup for selected cache efficiencies

Review

- **The ratio of rates at which contemporary microprocessors can fetch words from memory and cache can be used in conjunction with an estimated (or targeted) performance figure to approximate cache efficiency. Conversely, an estimated (or targeted) cache efficiency can be used to predict relative performance.**
- **Sensitivity data suggests that work on a model supporting additional tiers will be necessary where tighter tolerances are indicated (near completion)**
- **Hardware design data for a similar study of the IBM POWER2 and POWER3 microprocessors has been gathered and we are currently awaiting installation of a primitive equivalent of perfex written by the POWER3 Design Group. Validation of the method on multiple platforms is a priority.**

Closing Remarks

- **Unrealistic efficiencies associated with peak performance levels should never be used for the purpose of estimating equipment throughput.**
- **The sets of curves presented within clearly show that the production of efficient code becomes increasingly difficult as the difference between memory and cache fetch rates grows.**
- **As processor clock rates are quickly approaching physical limits, opportunities for memory system designs to catch processors we expect this trend will soon reverse itself. In the mean time, larger and smarter caches will serve to mask memory system design imbalance.**