

1999 DoD HPC Users Group Conference

On Microprocessors, Memory Hierarchies and Amdahl's Law

David O'Neal

CSM Site Lead

Aeronautical Systems Center MSRC

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

oneal@ncsa.edu

1999 DoD HPC Users Group Conference

- **Origins**

- Research was initiated in 1997 while working on a paper describing fundamental differences between Cray PVP and MPP architectures.
- Core methods were determined just prior to leaving Carnegie Mellon for UIUC.
- The recent decommissioning of the C916 coupled with the acquisition of a 64-processor Compaq machine regarded by some as a replacement platform prompted me to dredge it up in order to test the indicated 4:1 purchase ratio of DEC EV6 microprocessors to the Cray proprietary design.

- **Goals**

- Originally developed for estimating cache efficiencies of codes designed to run on the Cray T3D and early versions of the Cray T3E.
- Formalize, analyze, and validate the method in order to determine its suitability for (i) making throughput comparisons and (ii) estimating factors used to convert service unit allocations between platforms designed around dissimilar processors.

1999 DoD HPC Users Group Conference

- **Amdahl's Method**

- The classical form of Amdahl's Law (1967) can be stated as follows: if the serial component of an algorithm accounts for $1 / S$ of its execution time, then the maximum speedup that can be attained by running it on a parallel computer is S .

Let T represent execution time as a function of the processor count P and problem size N . Then given a sequential code that scales as $N + N^2$, any number of forms can be written.

- **Examples:**

1. $T = N + N^2 / P$ partition $O(N^2)$ part, replicate $O(N)$ part.
2. $T = (N + N^2) / P + 100$ partition all computations; add fixed cost.
3. $T = (N + N^2) / P + P^2 / 2$ partition all computations; add $O(P^2)$ cost of data communications.

1999 DoD HPC Users Group Conference

- **Amdahl's Law for Cache-based Microprocessors**

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. Consider a basic hierarchy comprised of a CPU, a cache structure, and a monolithic memory system. In this setting, the ratio of fetch times for memory, t_M , and cache, t_C , resident data determine speedup. The supposition is that every code requires memory-to-processor bandwidth that will eventually limit its performance on a cache-based system. If this part of the code were to suddenly fit into cache, the potential for performance improvement would be S . Other definitions we will need 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
- S ratio of data fetch times t_M / t_C (speedup)

1999 DoD HPC Users Group Conference

- Amdahl's Law for Cache-based Microprocessors**

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 of referenced identities followed by a rearrangement of terms yields:

$$T_N = t_M N (1 - f_C (S - 1) / S)$$

Dividing both sides by T_N and then solving for P_N produces a relationship for estimating absolute performance:

$$P_N = (t_M (1 - f_C (S - 1) / S))^{-1}$$

Without loss of generality, we can normalize the time unit with respect to t_C which implies $t_M = S$ and an expression for estimating relative performance emerges:

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

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

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

1999 DoD HPC Users Group Conference

- **Analysis**

The ratio of fetch times (S) for a given architecture has a significant effect on the relationship between relative performance (p_N) and cache efficiency (f_C) as illustrated in Fig. 1.

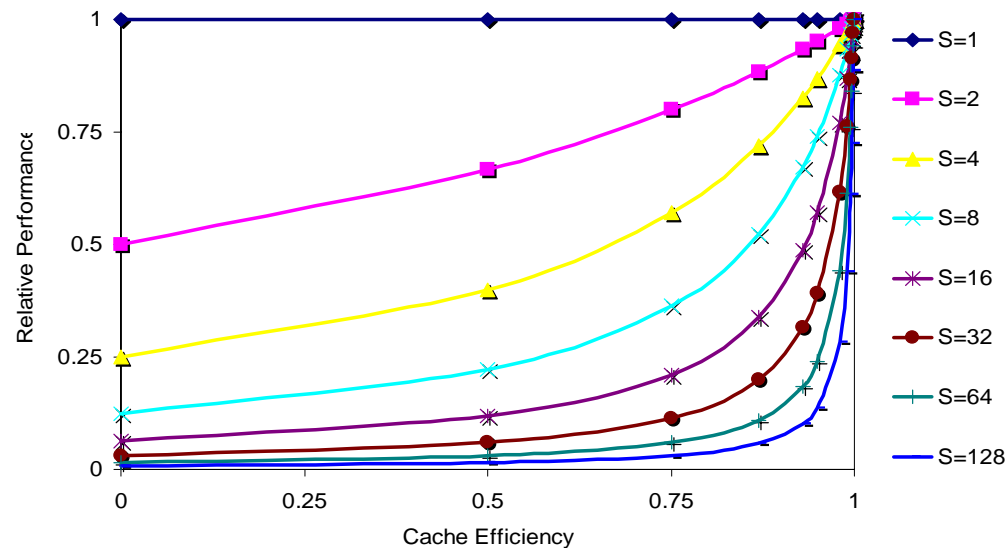


Figure 1. Relative performance as a function of cache efficiency for selected speedups

1999 DoD HPC Users Group Conference

- **Analysis (continued)**

The curves of Fig. 2 characterize cache efficiency as a function of relative performance, i.e. the inverse of the relative performance function.

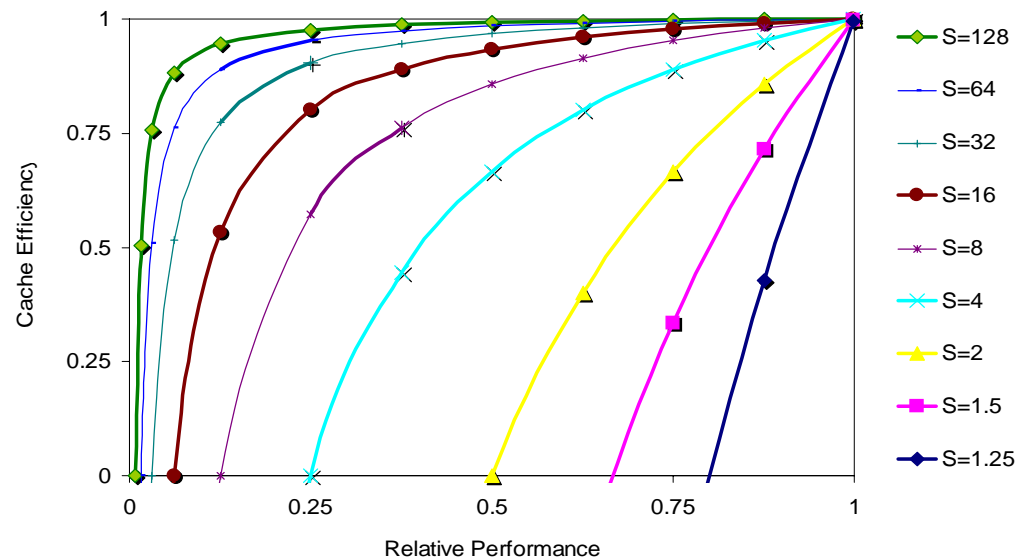


Figure 2. Cache efficiency as a function of relative performance for selected speedups

1999 DoD HPC Users Group Conference

- **Analysis (continued)**

Finally, consider the curves presented by Fig. 3. in which relative performance is regarded as a function of S . This view provides 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.

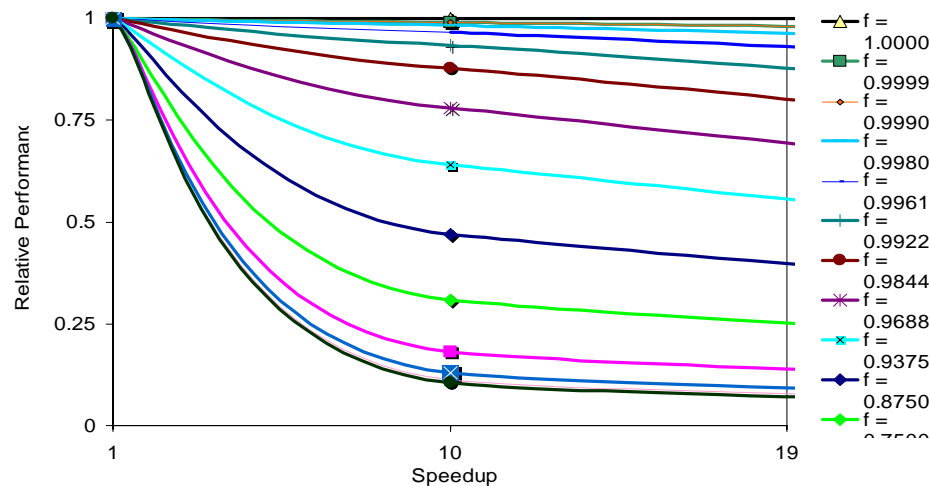
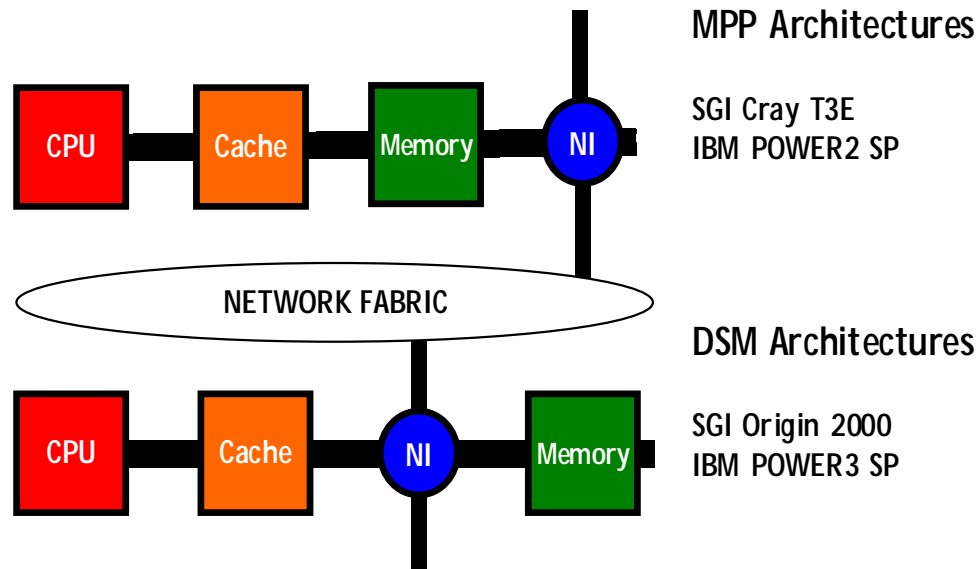


Figure 3. Relative performance as a function of speedup for selected cache efficiencies

1999 DoD HPC Users Group Conference

- **Consideration of the Network Design**

A final item worth mentioning is the potential effect of networking design on the performance of the memory channel. As shown by Fig. 4, some of the most popular distributed shared memory (DSM) machines position a network interface (NI) between the processors and memory. The transfer rate across this device cannot be neglected when studying the single processor performance of DSM machines.



1999 DoD HPC Users Group Conference

- Validation**

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

1999 DoD HPC Users Group Conference

- Validation (continued)**

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 clearly indicated by the increase in speedup 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 $f_C = 0.5$

1999 DoD HPC Users Group Conference

- Validation (continued)**

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 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	homer	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	homer	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

1999 DoD HPC Users Group Conference

- Validation (continued)

In an effort to confirm the results shown by Table 3, a view of the faster R10000 chip is provided by Table 4. As expected, all of the estimates exceed measured values.

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

1999 DoD HPC Users Group Conference

- Validation (continued)

We'll note in closing that a sensitivity study of the relative performance and cache efficiency functions has not yet been completed. However, results for the larger horner cases in particular suggest that the method is more susceptible to error at lower values of f_C . The curves of Fig. 3 serve to confirm this hypothesis. At the indicated speedup values, diminishing cache efficiencies correlate to steeper rates of change.

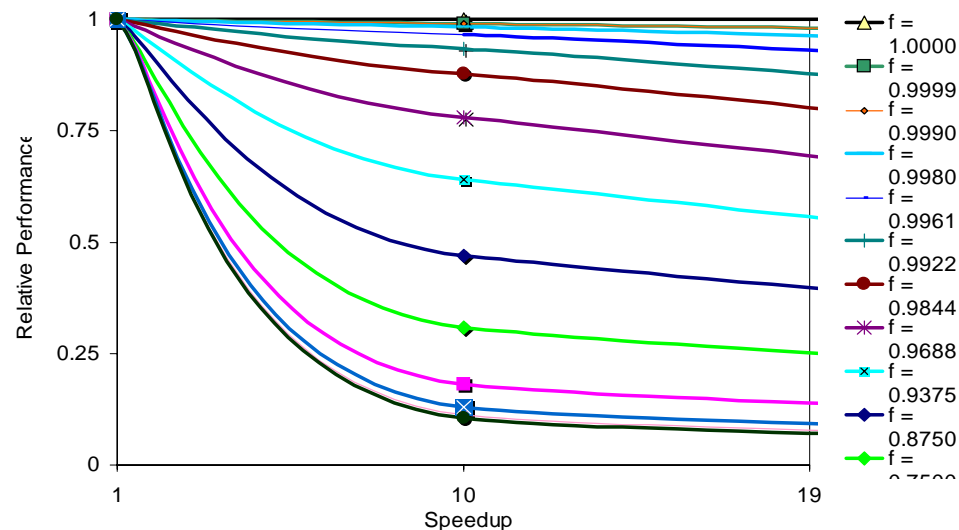


Figure 3. Relative performance as a function of speedup for selected cache efficiencies

1999 DoD HPC Users Group Conference

- **Summary**

- The difference between the 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 the corresponding cache efficiency. Conversely, an estimated (or targeted) cache efficiency can be used to predict relative performance.
- 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 library developed by the RS/6000 Division that provides the same functionality as SGI's `perfex` utility. Validation of the method on other platforms is considered to be essential.

1999 DoD HPC Users Group Conference

- **Summary (continued)**

- Increases in caching efficiency can be realized externally through vendor-initiated improvements in hardware or software design, but for fixed set of conditions, the burden of affecting enhancements in cache utilization must be taken on by the user community. Any presumption that implies a need to achieve higher cache efficiencies must consider the amount of work required to accomplish the task. In any case, unrealistic efficiencies in the form of near-peak performance levels should never be used for the purpose of evaluating equipment throughput.
- The set 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. However, as processor clock rates are quickly approaching physical limits, we expect that this situation will reverse itself. In the mean time, larger caches, improved logic, and additional layers of memory hierarchy must serve to mask imbalances.