# Applications Performance on the Cray T3E and T3E-900

*Nick Nystrom and David O'Neal*
*Pittsburgh Supercomputing Center, Pittsburgh, PA, U.S.A.*

nystrom@psc.edu
oneal@psc.edu
www.psc.edu

**ABSTRACT:**

This paper characterizes the performance of Cray T3E and T3E-900 systems installed at the Pittsburgh Supercomputing Center (PSC) for a variety of "real world" scientific applications. The first prototype Cray T3E and T3E-900 systems as well as the first production Cray T3E were installed at the PSC beginning in April 1996, and PSC staff has been extensively involved in porting and tuning applications to the T3E architecture. This paper summarizes early investigations into the effects of enabling hardware instruction streams, static and dynamic load balancing on partitions of uniform as well as different processor clock speeds, and the ramifications of memory bandwidth in relation to processor speed.

**KEYWORDS:**

Applications, Performance, Cray T3E, Architecture

# Introduction

Since mid-1995, the PSC has been working together with users toward employing the Cray T3E as its primary resource for conducting large-scale computational science. The PSC received the first prototype Cray T3E in April 1996 and accepted the first production model T3E on August 15, 1996. The configuration of the PSC's production T3E ("jaromir") at the time of this writing consists of 512 application PE's, with equal numbers of 300 MHz and 450 MHz processors. This is supported by approximately 250 GB of rotating media divided between SCSI and Fiber disks and HiPPI links to the PSC's archiver and other computational resources.

For the majority of applications, the transition from the T3D to the T3E was straightforward. Most changes to source code and build procedures were due to evolution of the software environment rather than the architecture of the T3E per se. The two examples of this type of change encountered most often were 1) migrating sources which were not FORTRAN 77 compliant to Fortran 90, and 2) removing explicit library and include paths from makefiles due to the emergence of "modules." A third example is the problem of what to do with CRAFT codes, since CRAFT, which was supported by `cf77` on the T3D, is not supported by `f90`. At the PSC this has been inconsequential because emphasis on performance has resulted in no production codes being written in CRAFT. In the long term, however, CRAFT codes will be portable to the T3E using `pghpf`, the HPF compiler from the Portland Group, Inc.

The difference between the T3D and T3E which most frequently necessitated nontrivial changes to source code is the loss of packet ordering when using `shmem_put` for communicating data between PE's. For many applications this is inconsequential. For others, though, it is necessary to insert calls to

`shmem_fence`, `shmem_quiet`, and/or `shmem_barrier` as documented [1].

Effort spent moving applications from the T3D to the T3E has therefore focused on tuning for performance. This paper surveys the results obtained, focusing on the performance of the Cray T3E-600 and T3E-900 for full-scale applications representative of the workload at the PSC. In particular, we have investigated the effects of enabling hardware instruction streams, static and dynamic load balancing on partitions of uniform as well as different processor clock speeds, algorithmic improvements to improve scalability, and the ramifications of memory bandwidth in relation to processor speed.

# Applications performance: a high-level view

Table 1 lists wall-clock execution times, comparisons of T3E-600 vs. T3E-900 performance, and assessments of the effect of enabling hardware stream buffers for several applications for which robust sets of timings data are available. Table 2 compares T3E-600 vs. T3E-900 performance for other applications. The applications listed in Tables 1 and 2 represent a small subset of those currently running on the T3E's at PSC and are intended to portray the range of performance levels commonly encountered.

**Table 1. Execution times (s) and ratios of T3E-600/T3E-900 and streams on/off performance for several applications.**

| application | benchmark | PE's | streams | T3E-600 (C2) | T3E-900 (C3) | t(600)/t(900) |
|---|---|---|---|---|---|---|
| AMBER | DNA/EtOH/$H_2O$ 17,000 atoms | 30 | off | 38.7 / 22.5 | 32.2 / 18.0 | 1.20 / 1.25 |
| | | | on | 33.3 / 21.3 | n/a | n/a |
| | | | ratio | 1.16 / 1.06 | n/a | |
| GAMESS | silicon cage | 4 | off | 4212 | 3160 | 1.33 |
| | | | on | 4130 | 3027 | 1.36 |
| | | | ratio | 1.02 | 1.04 | |
| PFEM | test problem #1 524,288 elements 787,456 nodes | 4 | off | 326.2 | 307.3 | 1.06 |
| | | | on | 160.5 | 146.1 | 1.10 |
| | | | ratio | 2.03 | 2.10 | |
| RSPJAC | test problem #3, 1024x1024 | 4 | off | 545.3 | 515.2 | 1.06 |
| | | | on | 492.0 | 460.0 | 1.07 |
| | | | ratio | 1.11 | 1.12 | |
| X-PLOR | pen-fft grid=0.05 | 4 | off | 1477 | 1134 | 1.30 |
| | | | on | 1371 | 1019 | 1.35 |
| | | | ratio | 1.08 | 1.11 | |

**Table 2. Ratios of T3E-600/T3E-900 and streams on/off performance for several applications running on 4 PE's.**

| application | | t(600)/t(900) | |
|---|---|---|---|
| | | s off | s on |
| CCG | | 1.30-1.36 | 1.36-1.42 |
| CHARMM | | n/a | 1.4 |
| fMRI | sgrid | 1.44 | n/a |
| | srecon | 1.77 | n/a |
| | estireg | 1.50 | n/a |
| | entire | 1.53 | n/a |
| MNDO 94 | | n/a | 1.4 |
| POP | | n/a | 1.01 |

While it would be inappropriate to draw many conclusions from the sampling in Tables 1 and 2, several generalizations are possible. First, speedups on going from the 300 MHz T3E-600 to the 450 MHz T3E-900 tend to lie between 1.0 and 1.4. It is difficult to achieve a 1.5 speedup even for applications which do no I/O because only the processors of the T3E-900 are running at a higher clock rate, whereas the memory system and communications network is the same on the two machines.

The second observation is that enabling hardware stream buffers typically improves performance by 10-20%, but outlying cases ranging from no improvement to 100% improvement have been observed. Furthermore, enabling stream buffers appears to improve performance slightly more on the T3E-900 than on the T3E-600. Two factors contribute to this effect. First, the higher floating-point rate to memory bandwidth ratio on the T3E-900 implies that improving the memory access rate in any way will be more apparent than on the T3E-600. Second, the T3E-600 used for benchmarking implemented Pass 2 boards, while the T3E-900 figures reflect Pass 3 and Pass 5 boards. As investigated later in this paper, memory performance has improved in recent hardware revisions.

GAMESS is discussed extensively in the following section on load balance. We conclude this section with several remarks on the PFEM, RSPJAC, and fMRI applications.

## PFEM and RSPJAC

RSPJAC (Real Symmetric Parallel Jacobi eigensolver) and PFEM (Parallel Finite Element Method) both depend heavily on low-latency communications. This is especially true for PFEM as it fetches and stores single words; whereas RSPJAC performs block operations. Performance increases due to the faster EV5 processors are marginal as no corresponding change in memory bandwidth was effected to support the higher rate. The RSPJAC code uses scilib routines that were not yet optimized for the EV5 at the time the benchmark was run. Performance increases due to the faster clock are likely to grow for RSPJAC as cache utilization is improved. The use of streams reduces latency in loads from DRAM to cache, hence performance improvement due to the use of streams is a function of the number of local memory fetch operations.

## fMRI

The functional MRI (fMRI) projects includes several distinct applications: sgrid and srecon, which are I/O intensive, and estireg, which is compute-intensive. The benchmarking run consisted of two invocations of sgrid, two of srecon, and one of estireg. The apparent superlinear speedup on going from the T3E-600 to the T3E-900 is presumably an artifact of the I/O-intensive nature of srecon.

## Mixed Clock Rates and Load Balance

A Cray T3E can be populated in several ways, which at the time of this writing include 300 MHz PE's, 450 MHz PE's, or a combination of the two. In a mixed configuration machine one can label executables using the `setlabel` utility to target one type of processor or the other. "Hard" labels indicate a requirement for a particular processor type, whereas "soft" labels indicate only a preference. In a production environment managed by NQE, however, extensive use of hard-labeled executables leads to decreased utilization as processes which are gated through by NQE must wait for corresponding blocks of processors to become available before they begin execution. Thus it is advantageous either to refrain from building labeled executables, or at most to soft-label them (e.g. `setlabel -l S450 foo.exe`). Either action avoids negatively impacting system throughput, but introduces the possibility that a process will be distributed across PE's of CPU's of different speeds.

A statically load balanced application, typically one which is parallelized by distributing successive loop iterations or blocks of data to successive processors, will execute at the same rate on a partition of *M* fast PE's and *N* slow PE's as it would on a partition of *M+N* slow PE's. That is to say that the additional capacity available from the *M* fast PE's is wasted to load imbalance as the *N* slow PE's complete their equal share of the work. This is a routine consideration when designing distributed applications for workstation clusters, but it is often dismissed when designing applications for tightly coupled parallel systems because the processors are dedicated and of equal power. However, applications which are to run at maximum efficiency on mixed T3E partitions (including full-machine runs on systems with different processor types) must anticipate the mixed environment and balance their loads dynamically rather than statically.

This effect is clearly illustrated in the case of GAMESS [2], a quantum chemistry code which is generally run using static ("LOOP") load balancing on MPP's and dynamic ("NXTVAL") load balancing on networks of workstations. Table 1 enumerates execution times for GAMESS on homogeneous partitions of 300 MHz and 450 MHz processors as well as heterogeneous partitions with even numbers of each speed of PE's. (The latter runs were performed using the `-l` option to `mpprun` to specify the base PE.) Homogeneous runs using 450 MHz PE's execute 1.29 and 1.19 times as fast as corresponding 300 MHz runs for 64 and 128 PE's, respectively. The heterogeneous runs yield speedups of 1.14 and 1.12, which are almost exactly what one would predict based on available processor power. The relatively low speedups of 1.19 (450 MHz) and 1.12 (mixed) reflect processor starvation when running this particular size of problem on 128 PE's.

**Table 3. Wall clock execution times for GAMESS (18 March 1997 release) performing a 6-31G(d) RHF+gradient calculation on $HSi[(CH_2)_3]_3C_6H_3$.**
**Timings reflect dynamic load balancing with hardware streams off.**

| PE type | | PE's | |
|---|---|---|---|
| **300 MHz** | **450 MHz** | **64** | **128** |
| all | none | 368 | 252 |
| half | half | 322 | 226 |
| none | all | 286 | 211 |

Figure 1 illustrates the general improvement in scalability with problem size for four RHF+gradient calculations of increasing size: ADSbO (110 basis functions), phosphinoaluminate ion (169 basis functions), a silicon cage compound (288 basis functions), and cyclic AMP (389 basis functions). The scalability curves for to the statically load balanced runs conform well to Amdahl's law fits, with RMS deviations between observed and fitted values less than 5% in all cases.

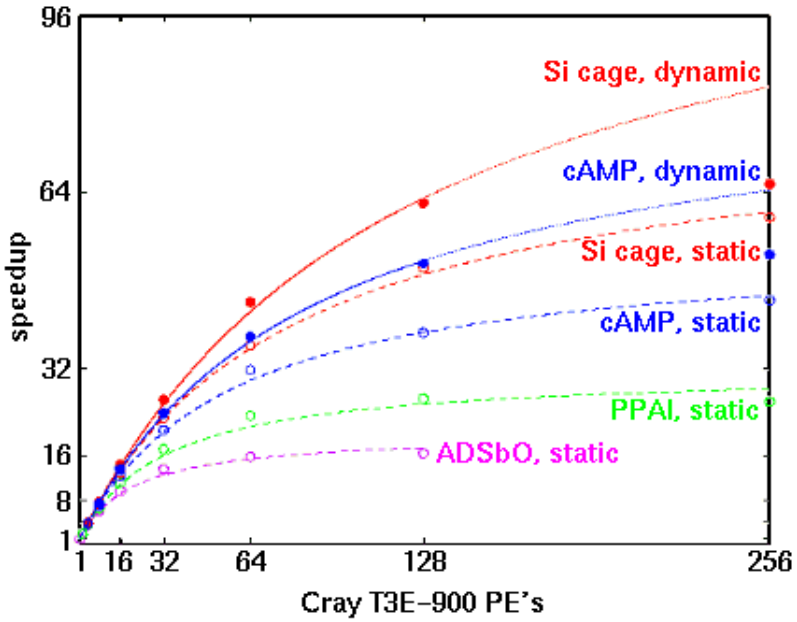



**Figure 1. Scalability of GAMESS (18 March 1997 release) on the Cray T3E-900. Dashed curves represent Amdahl's law fits to statically load-balanced runs. Solid curves represent Amdahl's law fits to dynamically load-balanced runs. The dotted extensions represent their extrapolations, highlighting the effect of processor starvation beyond 128 PE's for the problems considered here.**

Figures 2a and 2b decompose the scalability of the RHF gradient calculation on the silicon cage compound into individual components for the T3E-600 and T3E-900, respectively. Only those subcalculations which require a significant amount of time are shown. On both machines, the two dominant aspects of the calculation, i.e. the self-consistent field calculation and the two-electron gradients, are seen to scale reasonably well. The performance of these two types of computation follows similar trends between machines and with load balancing strategy.

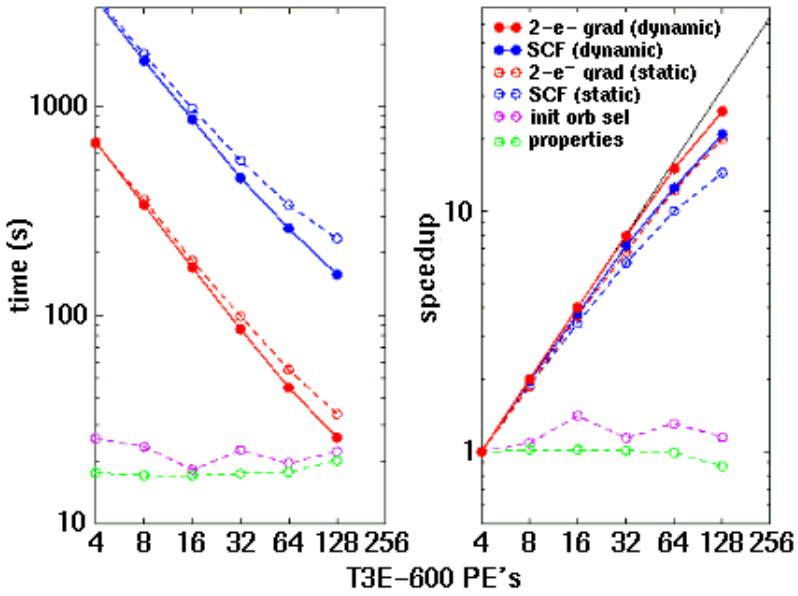



**Figure 2a. Wall clock execution time and speedups relative to 4 PE's for GAMESS (18 March 1997 release) on the Cray T3E-600 for a 6-31G(d) RHF+gradient calculation on the silicon cage compound $HSi[(CH_2)_3]_3C_6H_3$ involving 288 basis functions. Dashed and solid curves depict performance obtained with static and dynamic load balancing, respectively.**

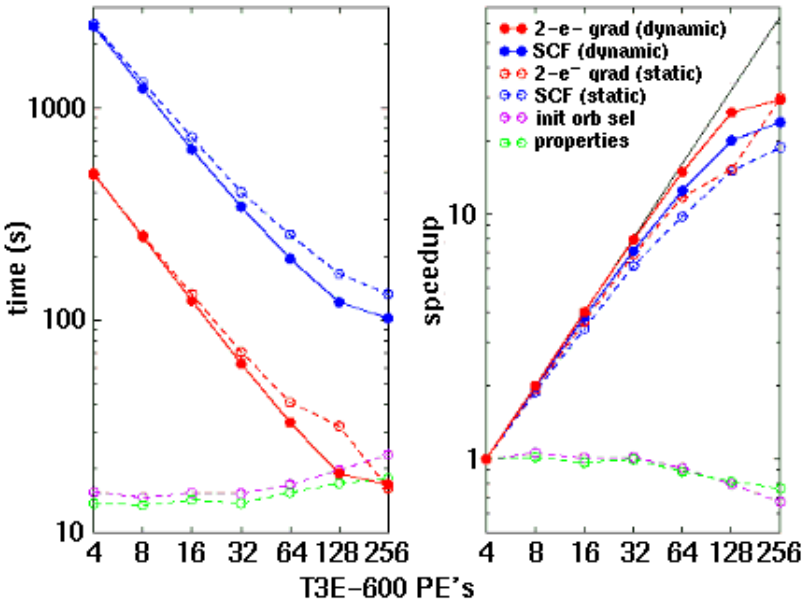



**Figure 2b. Wall clock execution time and speedups relative to 4 PE's for GAMESS (18 March 1997 release) on the Cray T3E-900 for a 6-31G(d) RHF+gradient calculations on the silicon cage compound $HSi[(CH_2)_3]_3C_6H_3$ involving 288 basis functions. Dashed and solid curves depict performance obtained with static and dynamic load balancing, respectively.**

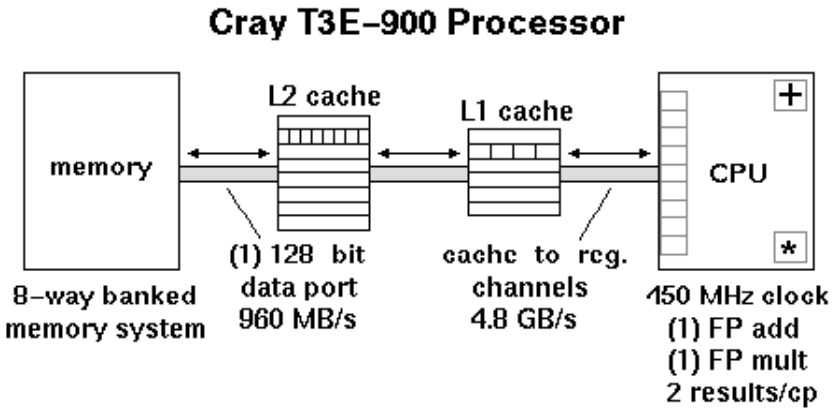
First, the scalability of either calculation deteriorates slightly on the T3E-900 relative to the T3E-600 when considering either statically or dynamically load balanced treatments. This is to be expected because the communications networks operate at the same speed in the two machines; therefore, one witnesses a degradation in scalability as processor speed increases. This degradation is marginally greater for the statically than for the dynamically load-balanced solutions. The execution rate on the T3E-900 is approximately 1.36 times that on the T3E-600, limited primarily by memory access patterns which prevent high cache utilization.

Second, in all cases, dynamic load balancing is seen to be substantially superior to static load balancing. For example, on 64 T3E-900 PE's (a reasonable number on which to run this particular problem), dynamic load balancing improves the scalability relative to 4 PE's of the SCF calculation by 28% (from 9.81 to 12.50, where 16 would be linear) and that of two-electron gradients by 27% (from 11.75 to 14.93). Thus for GAMESS, which is typical of many applications for which it is difficult to establish uniform work loads across processors, dynamic load balance is advantageous for partitions of identical processors as well as for the case of heterogeneous partitions discussed earlier.

Two aspects of GAMESS which have not been parallelized show up in Figures 2a and 2b as essentially level curves. While their execution time is trivial for low processor counts, clearly their parallelization is necessary for scalability to improve beyond 64 PE's.

# Applications performance: looking deeper

Cray's current MPP design is based on Digital Equipment Corporation's EV5. The EV5 is a 64-bit cache-based microprocessor. The basic design includes a 1 KWord instruction cache, a 1 KWord direct-mapped primary (L1) cache, and a 12 KWord write-back secondary (L2) cache featuring 3-way set associative logic. L2 cache handles both data and instructions. The EV5 features two pipelined floating point functional units (add, multiply). Each pipe is capable of producing one operation per cycle. A chained mult-add operation is also supported, hence the peak rate for the processor is 2 results per cycle.




The peak transfer rate for the memory port is 960 MB/s, or about 1 word every 4 cycles. Streaming operations imply the sharing of this channel, e.g. moving a word from memory to cache and back again means that two words cross the channel. Therefore, chained operations involving a single array and one or more scalar variables can sustain a maximum of 1 result every 4 cycles (actually 1 mult-add every 8

cycles).

Cache-to-register bandwidths are 8 times greater than the memory-to-register rate, so if it were possible to fit an entire application into secondary cache, processor starvation due to inadequate bandwidth would be avoided, resulting in near-peak execution rates.

The final factor affecting bandwidth is a direct result of processing cache load requests a line at a time. Each line of L1 cache consists of four words while each line of L2 cache is comprised of 8 words. This read ahead/behind strategy implies that striding through memory by anything greater than 1 is pathological. Any stride exceeding one requires that more data than necessary must be loaded, leading to reductions in bandwidth that follow directly from the inverse of the stride, up to a factor of 8.

Three benchmarks were used to instrument single-PE performance on the Cray T3E: a kernel implementing Horner's algorithm for the evaluation of polynomials, a vector triad kernel, and a memory copy kernel [3]. Each test code was written in FORTRAN and compiled using the most current release of the F90 compiling system(s) with stream buffers enabled. Timings were obtained for processes running on application PE's using the `TSECND` utility. The tests were performed on a loaded system; therefore, experiments were performed repeatedly with the best rates observed reported. Floating point performance data was collected by running an inner loop over the vectors of interest, and an outer loop that repeats the inner loop operations to minimize systematic error in timing measurement. Because compilers for all of the machines tested have optimizations that tend to defeat the purpose of the benchmark by removing redundant code, a separately timed subroutine call was added at the bottom of the outermost loop to block such optimizations. While this technique can prevent other (desirable) optimizations from being implemented, the call is necessary in order to provide a consistent basis for comparison. Finally, loop splitting and unrolling is automatically supported by the current compiler versions on all but the T3X machines, hence explicit directives were added to the T3X sources to level the playing field in that respect.

## Horner's Algorithm

Horner's algorithm, which nests multiplications to evaluate polynomials, can be made to produce near-peak floating points rates on most processors because an arbitrary number of streams involving both multiply and add operations is easily specified, and the data required to evaluate the expression can be sized to conform with available cache or registers. The following code fragment repeatedly evaluates a 6th order polynomial for various values of x. Note that the order of the polynomial corresponds to the number of streams on the right-hand side.

```
do i = 1, dim
  b(i) = ((((( f0 * a(i) + f1 )
                  * a(i) + f2 )
                  * a(i) + f3 )
                  * a(i) + f4 )
                  * a(i) + f5 )
                  * a(i) + f6
end do
```

Near-peak performance is expected when the arrays are made to fit into cache or registers because there is essentially no input stream after the initial load, while the output stream is largely contained by the registers or cache.

**Table 4. Horner's algorithm performance on Cray systems.**

| | | dimension=1024 | | dimension=8192 | |
|---|---|---|---|---|---|
| **Platform** | **Peak** | **MFlop/s** | **Efficiency** | **MFlop/s** | **Efficiency** |
| T90 | 1760 | 1508 | 86% | 1507 | 86% |
| C90 | 960 | 854 | 89% | 853 | 89% |
| T3E-900* | 900 | 711 | 79% | 294 | 33% |
| T3E-900 | 900 | 582 | 65% | 317 | 35% |
| T3E | 600 | 452 | 75% | 272 | 45% |
| T3D | 150 | 90 | 60% | 81 | 54% |

The T3E-900* platform is a Pass 3 Cray T3E with 450 MHz processors. It is uniquely qualified to demonstrate the effects of increasing processor performance without further modification to the existing architecture.

Efficiency levels for Horner's algorithm are high for all systems when the data is small enough to fit into registers or cache. As the array size increases, efficiency drops most rapidly for the faster cache-based processors. This observation clearly illustrates the importance of effective cache use on fast EV5 processors given their floating-point and memory bandwidth characteristics. Performance levels for all cases involving resident data are consistent with the best applications at PSC.

## Vector Triad

The vector triad is a basic building block of engineering and scientific codes:

```
do i = 1, dim
  c(i) = k * a(i) + b(i)
end do
```

The ratio of floating point operations to memory operations is significantly lower for the vector triad operation (2:3) than for the Horner's algorithm evaluation of a 6th-degree polynomial (12:2).

**Table 5. Vector triad performance on Cray systems.**

| | | dimension=1024 | | dimension=8192 | |
|---|---|---|---|---|---|
| **Platform** | **Peak** | **MFlop/s** | **Efficiency** | **MFlop/s** | **Efficiency** |
| T90 | 1760 | 1070 | 61% | 1022 | 58% |
| C90 | 960 | 450 | 47% | 450 | 47% |
| T3E-900* | 900 | 246 | 27% | 38 | 4% |
| T3E-900 | 900 | 201 | 22% | 44 | 5% |
| T3E | 600 | 145 | 24% | 26 | 4% |
| T3D | 150 | 19 | 13% | 13 | 9% |

Two caching effects are clearly illustrated. For small vectors that exceed the boundaries of L1 cache but fit within L2 cache, a huge gain in efficiency is realized (T3D and T3E data for the smaller case). However, operations on larger arrays result in severe drops for the faster processors, and more modest losses for the slowest processor.

The memory port arrangement of the vector machines conforms exactly to a vector triad, hence there is little contention for bandwidth, and there is negligible reduction in performance as vector size increases. Bank conflicts are primarily responsible for throttling performance, regardless of array size. The cited floating point rates for the T90 and C90 imply transfer rates of 4088 MB/s and 1800 MB/s per channel, respectively.

The trends established by the Horner’s algorithm kernel are magnified for the vector triad kernel due to reduced ratio of floating-point operations to memory references. This pattern is particularly problematic for cache-based machines as it can result in virtually no cache reuse (i.e. just the scalar loop invariant).

The small improvement in efficiency between the T3E and the T3E-900 for the large dimension vector triad operation is of special note. This implies that a speedup exceeding the difference in clock rates is present. The memory channel on the Pass 3 (450 MHz) boards is about 20% more efficient than the Pass 2 (300 MHz) boards, meaning that "superlinear" speedups can be observed. Further evidence of bandwidth improvement between these platforms is presented in the next section.

## Memory Copy

We now depart from a setting in which performance was measured in terms of floating point operations per second, and enter the realm of bits and bytes per second. It contains more material than the other sections. The concepts explored here are more fundamental.

Memory copies are routinely performed in most codes, both explicitly and implicitly. Implicit use is particularly prevalent in parallel Fortran offshoots like HPF, and if the primary data structures are long vectors, the results can be devastating to performance.

We begin with a typical copy loop with a stride of 1:

```
do i = 1, dim
  b(i) = a(i)
end do
```

On the MPP platforms, special care is taken to cause cached data to be spilled prior to initiating subsequent copy loops. This is done by initializing a third array that is sized to cover L1 or L2 cache. The new data must be ejected prior to loading the source array, so this transfer has been included in the rates for the cache-based machines.

**Table 6. Memory copy performance on Cray systems.**

| Platform | Peak | Size (KB) | MByte/s | Efficiency |
|---|---|---|---|---|
| T90 | 21118 | 96 | 9888 | 47% |
| C90 | 11519 | 96 | 5850 | 51% |
| T3E-900* | 960 | 96 | 528 | 55% |
| T3E-900 | 960 | 96 | 629 | 66% |
| T3E | 960 | 96 | 520 | 54% |
| T3D | 480 | 8 | 286 | 60% |

Since copy loops use only two of the three data channels on the vector machines, hence we could not expect to exceed 2/3 of the peak rates cited for the T90 and C90. With the notable exception of the T90, the efficiencies of all systems exceed half of their respective peak rates for a stride of 1.

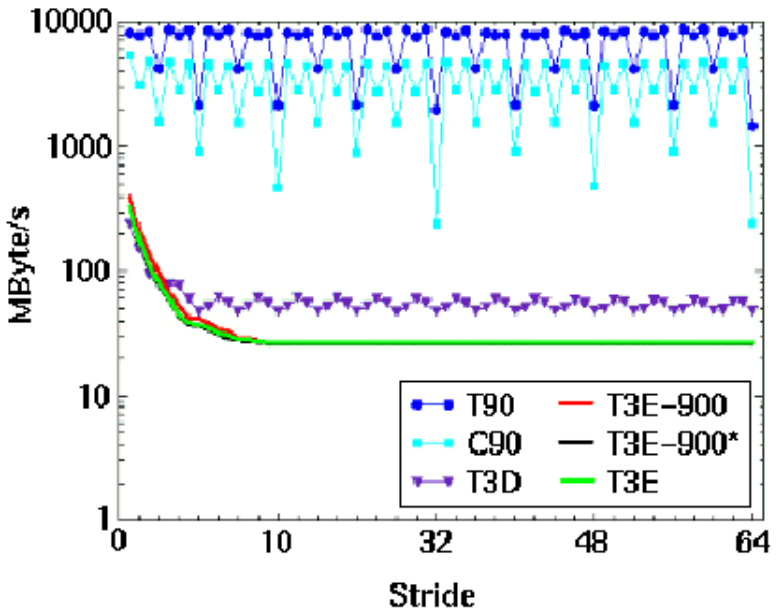


**Figure 3. Bandwidth for strided memory copies on the Cray T3D, T3E, T3E-900, C90, and T90.**

The difference between peak and base rates for the MPP platforms is clearly a function of the cache line size, as expected. The performance levels for all T3E architectures for strides exceeding 8 words (L2 cache line size) is less than half that of the T3D. In these cases, what we are really measuring is the rate at which a cache eject and load operation can be performed. The T3E must store an 8-word cache line and load another, then send the proper group of 4 words to L1 cache. Less than half of this much work is done on the T3D. It ejects and loads 4-word lines directly to/from L1 cache, requiring much less logic to maintain.

The performance of the vector codes is obviously affected by memory bank conflicts. A ratio of about 25% is reported by HPM. For the MPPs, there is little evidence of performance problems due to memory bank conflicts. The loading and unloading of superfluous data is the albatross that the cache-based machines wear.

# Conclusions

Efficient utilization of the L1 and L2 caches is crucial to attaining good performance on the Cray T3E, just as using the L1 cache well was essential on the Cray T3D. T3E-600 speedups relative to the T3D typically fall in the range of 2.5-3.5, although outliers to either side have been observed. The T3E-900 (450 MHz PE's) tends to execute most full applications between 1.3 and 1.5 times as fast as the T3E-600 (300 MHz PE's), with the difference being slightly larger with stream buffers enabled than without. This figure is lower than the 1.5 expected from considering only the processor clock rates because the memory system is still running at 300 MHz. Also, for certain applications, I/O requirements reduce the factor to nearly 1.

Improvements in the memory channel between board revisions (specifically, Pass 2 and Pass 3) on the order of 20% have been observed for a vector triad benchmarking kernel. Comparison with yet newer revisions (up to Pass 5) and identifying these effects in the execution times for real applications is in progress.

Dynamic load balancing is critical to making effective use of T3E's which contain a mix 300 and 450 MHz PE's. A parallel application will execute on such a heterogeneous partition whenever it either requires the whole machine or is loaded in a way that spans the boundary by the runtime system. While it is possible to label executables so as to require one type of processor or another, doing so is generally detrimental to system throughput. In addition, it was found that even on a homogeneous partition, dynamic load balancing significantly improved the scalability of at least one application traditionally run using static load balancing on systems of uniformly capable processors.

In the near term it is expected that the 3.0 release of the Programming Environment will contain several features conducive to applications optimization, but at the time of this writing these have not been explored at the PSC. In particular, these features include

- The `cache_bypass` pragma/$CDR to instruct compilers to bypass the cache and instead use E-registers for gather/scatter operations, thereby gaining a large performance boost.
- The `-hpipeline[1-3]` compiler switch to specify various levels of software pipelining,

speculative loads, and speculative operations.

Those features, along with others, will provide additional opportunities for improving the performance of applications on the Cray T3E.

# Acknowledgments

Application performance figures and insight are gratefully acknowledged from the following individuals: AMBER: Michael Crowley; CCG, Qiming Zhang and R. Reddy; CHARMM: Bill Young; fMRI: Greg Hood; MNDO 94: Carlos Gonzalez; POP: John Urbanic.

---

# References

[1] Cray Research, Inc., **Guidelines for Programming T3E Streams**, available online at http://www.cray.com/PUBLIC/product-info/sw/PE/T3E_streams.html.

[2] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. Su, T. L. Windus, M. Dupuis, and J. A. Montgomery, Jr., **General Atomic and Molecular Electronic Structure System**, J. Comp. Chem. **14**(11), 1347-1363 (1993).

[3] D. O'Neal and J. Urbanic, **On Performance and Efficiency: Cray Architectures**, Conference Proceedings, Parallel Computing 1997, Ohio Supercomputer Center, April 1997.

---

# Author Biographies

Nick Nystrom holds the position of Scientific Specialist in the Parallel Applications Group at the Pittsburgh Supercomputing Center. After receiving a Ph.D. in quantum chemistry at the University of Pittsburgh in 1992, Nick joined the PSC, where his research interests include parallel applications development, the evaluation of scalable parallel systems, and numerical methods.

David C. O'Neal currently holds the position of senior scientific software developer with the Parallel Applications group at the Pittsburgh Supercomputing Center. Since joining PSC in 1990, he has continued his work with parallel algorithms and the finite element method in particular that was cultivated at the University of Pittsburgh (M.A. Applied Mathematics, 1989).