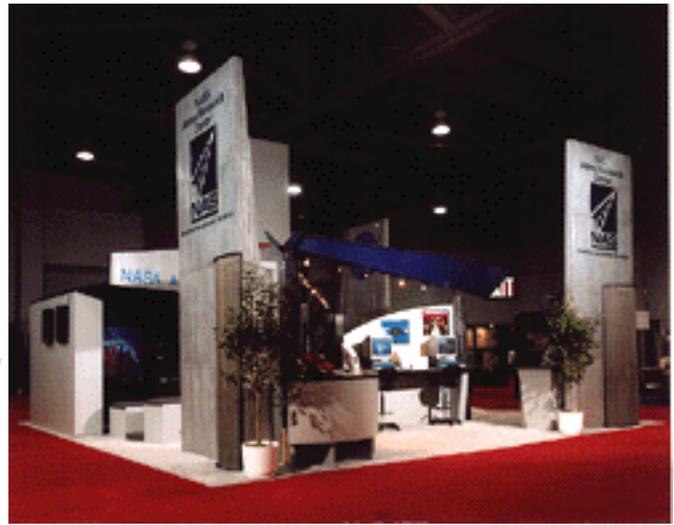


January - February 1995 Volume 2, Number 8



- [New File System Offers Increased User Productivity](#)
- [Overindexing Improves Performance on the NAS CRAY C90](#)
- [NAS Parallel Benchmarks Set Industry Standard for MPP Performance](#)
- [Forssell Gives 'Best Paper' at Visualization '94](#)
- [Supercomputing '94 Underscored Challenges Ahead](#)
- [Using Recursive FORTRAN Subprograms on the CRAY C90](#)
- [New AIMS Version Now Available to NAS Users](#)
- [Upcoming Training Events](#)
- [Credits for This Issue](#)
- [This Issue's Front Page](#)

[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)

New File System Offers Increased User Productivity

by Alan Powers

NAS has created a "super home" file system on the NAS CRAY C90 (vonneumann) by combining the functionality of the home and migrated file systems. This file system will help new users become immediately productive by reducing the number of file systems they must learn, and will also increase the total user disk capacity.

Some benefits of the super home file system are:

- Simplifies use of the system.
- Increases total home disk capacity and disk performance.
- Splits inode lookups for /m across four file systems, increasing file access speed and reducing overhead.
- Improves transfer rates to and from /big and /fast.
- Shortens backup and restore times.

Previously, two user file systems existed: homes (/u/v[acde]) and migrated (/m), as well as two temporary directory file systems, /big and /fast. This old configuration worked well for users who were aware of the purpose of each file system. However, when new users were added to the system, they tended to overuse their home file systems to run batch jobs, causing job turnaround time to be longer--and sometimes causing their jobs to fail because of full file systems.

In mid-December, the homes and migrated file systems were combined to form the super home file system. A prototype implementation of this had been working on the /u/vb file system since late August 1994.

Previous Configuration Was Slow

Prior to the super home configuration, each home file system consisted of 6.3 gigabytes (GB) of slow-

access (10 megabytes per second) storage space residing on Cray DD42 disks. Each user was allotted a disk quota that allowed their files to always be kept online; the average quota was about 100 MB, and the largest disk quota was about 1,000 MB. The top five percent of the users had greater than 1,500 inodes (files and directories) and only the top one percent had greater than 3,000 inodes. Data older than 120 days made up 25 percent of the total data. Files less than 100 kilobytes (KB) were 94 percent of the total number of files. In the past, NAS discouraged users from running batch jobs from their home directories because of the slow disk access.

/m was a migrated file system: data was stored on both tape and disk. The disks used for /m were Cray DD60s and Maximum Strategy's RAID (redundant array of independent disks) with a total capacity of about 150 GB online; total off-line storage was 1.1 terabytes. All data files larger than 100 KB were archived to tape, leaving 65 percent of the files always online. When space was needed for newly created files or to retrieve migrated files, data blocks of the oldest archived online files were released. Only files over one day old were eligible for release. Active files usually stayed online for two to three days. On average, only 10 percent of files accessed were retrieved from tape; the rest were already online. Each user was given a disk quota of 10 GB and 750 inodes on /m. Cray's Solid-state Storage Device (SSD) was used to cache this file system, and it ranked as the second most-used file system at NAS.

The /big file system contained 62 GB of Cray DD60 and RAID storage space. Users accessed this file system through the temporary directory defined by the variable \$BIGDIR. Any single user could consume up to 66 percent (40 GB) of the total disk space. Data files less than 1 MB were stored on the DD60s and files larger than 1 MB were stored on the RAID. The Session Reservable File System (SRFS) helped manage usage of /big. The largest amount of SSD was used to cache this file system--the third most-used system, which was best suited for large temporary files.

The /fast file system was 5 GB of SSD. Users accessed this file system through the temporary directory defined by the variable \$FASTDIR and were required to reserve space using SRFS. Any single user could consume up to 75 percent (3.7 GB) of the total disk space. This was the fastest and most-used file system. Files that were read multiple times within a job or for random I/O were best suited for this system.

New Configuration Enlarged

For the new super home file system, the idea was to enlarge each of the four home directories using six to eight DD60s and a RAID device, for a total size of 56-60 GB of disk space. The file system was configured so that small files (less than 1 MB) use the DD60s and large files (larger than 1 MB) use the RAID. Also, the Data Migration Facility uses the same configuration as it did on /m. Only files greater than 100 KB are being archived. This is similar to combining the functionality and disk from /m and home for each new home file system. Users are allowed (but not encouraged) to use their home directories to run batch jobs. /big and /fast are primarily used for highly I/O-intensive jobs. The home disk quota is 12 GB, and the inode quota was raised to 15,000.

In order to do this, /big was reduced to about 36 GB using all DD60 disks, and the users' /m directory was made a subdirectory of the user home directory. Through February 1995, symbolic links will allow users to view the /m file system. Existing scripts using /m will continue to run unmodified.

Overall, the super home file systems will increase user productivity, increase total user disk capacity, and reduce the complexity of the file system structure.

For more information, see the man pages for qsub, srfs, tmpdir, and ldcache, as well as [Chapter 3 of the NAS High Speed Processor User Guide](#).

See also the articles entitled, "[Improving I/O Performance on the NAS CRAY](#)" and "[RAID Technology Offers Cost-effective Data Storage for CRAY C90s](#)" (NAS News, September-October 1994).

If you have questions, call NAS User Services at (415) 604-4444, or send email to nashelp@nas.nasa.gov.



Alan Powers is a member of the NAS HSP group. He joined the NAS program in April 1992 and is currently working on a project to put a migrated file system on eagle, the Aeronautics Consolidated Supercomputing Facility CRAY C90.

[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)

[Next Article](#)[Contents](#)[Main Menu](#)[NAS Home](#)

Overindexing Improves Performance on the NAS CRAY C90

by Robert Bergeron

For most NAS users, the demand for CRAY C90 CPU time exceeds the number of allocated hours, so many users would like their codes to perform better in order to make the most of their allotted time. Although NAS rates the C90 at a factor of 2.2 faster than the CRAY Y-MP on workload codes, some users are unable to obtain such performance. A powerful, straightforward optimization technique, called overindexing, allows users to more effectively use their CPU allocations. Codes displaying Hardware Performance Monitor (HPM) vector lengths of less than 64 can most strongly benefit from this technique (although some codes with vector lengths of less than 64 may not benefit, due to their structure). Overindexing has produced CPU megaflop speedups of 1.3 to 1.9 on already well-written CFD codes.

Overindexing is the use of a single array index to perform the array addressing normally done by two or more DO-loop indexes. The technique has assumed additional importance under the C90 architecture: The peak performance of 1 gigaflop per CPU requires four floating-point operations per clock period and continuous saturation of the vector functional units. These units contain 64 double-width paired segments, with odd elements of the vector in one half and even elements in the other half, giving the architecture an effective vector length of 128. Under standard Fortran coding techniques, peak performance requires problem sizes to equal or exceed 128 in all dimensions in order to saturate the vector functional units, so only users with large problem sizes can fully use the C90 computing power.

Overindexing reduces the number of DO-loops required to perform a given calculation, which in turn reduces the overhead associated with DO-loop execution and more fully exploits the vector hardware by achieving longer vector lengths. The optimization uses the Fortran treatment of computer memory as a columnwise continuous sequence of storage locations.

For simple loop constructs, the Cray compiler already performs some overindexing by default. As mentioned in the article "[Linearizing Nested Loops](#)," (*NAS News*, September-October 1994) the Fortran PreProcessor (FPP) can indeed modify user code to extend the range of constructs subject to overindexing. However, successful use of FPP to overindex a complete code demands a strong understanding of the way in which this tool extracts recognizable constructs from code.

The examples here illustrates how to recognize and improve performance of more complicated constructs by overindexing arrays explicitly. While the overindexing optimization is legitimate for Cray vector architectures, correct results on other architectures *cannot* be guaranteed.

Overindexing Doubly Nested DO-loops

NAS codes generally visit the points in a grid and perform a small set of operations on arrays representing the physical conditions at each point. The codes contain nested DO-loops to manage this process. DO-loop nests spanning a continuous region in memory with no necessary coding between the DO-loop headers constitute candidates for overindexing.

The following example illustrates the use of overindexing on a simple loop computing outflow boundary velocities. Here is the standard code:

```

SUBROUTINE BSET
PARAMETER ( IX=39 , IY=75 , IZ=163 )
COMMON /CTRL/NX,NY,NZ , . . .
COMMON /A/UN( IX , IY , IZ ) , VN( IX , IY , IZ ) , WN( IX , IY , IZ )
DO 200 J=1,NY
  DO 200 I=1,NX
    UN( I , J , NZ ) =E1*UN( I , J , NZ-1 ) -E2*UN( I , J , NZ-2 )
    VN( I , J , NZ ) =E1*VN( I , J , NZ-1 ) -E2*VN( I , J , NZ-2 )
    WN( I , J , NZ ) =E1*WN( I , J , NZ-1 ) -E2*WN( I , J , NZ-2 )
  200 CONTINUE

```

The loops are tightly nested and traverse all elements in the first two dimensions. This nest will display improved performance by using a single index to span the memory addresses. Here is the overindexed code:

```

SUBROUTINE BSET
PARAMETER ( IX=39 , IY=75 , IZ=163 )
COMMON /CTRL/NX,NY,NZ , . . .
COMMON /A/UN( IX , IY , IZ ) , VN( IX , IY , IZ ) , WN( IX , IY , IZ )
DO 200 IJ=1 , NX*NY
  UN( IJ , 1 , NZ ) =E1*UN( IJ , 1 , NZ-1 ) -E2*UN( IJ , 1 , NZ-2 )
  VN( IJ , 1 , NZ ) =E1*VN( IJ , 1 , NZ-1 ) -E2*VN( IJ , 1 , NZ-2 )
  WN( IJ , 1 , NZ ) =E1*WN( IJ , 1 , NZ-1 ) -E2*WN( IJ , 1 , NZ-2 )
200 CONTINUE

```

The user employed values of $nx=39$, $ny=75$, and $nz=163$. The standard code displayed a vector length of 39 and an M/Flop rate of 351, whereas the overindexed loop displayed a length of 126 and an M/Flop rate of 567.

While it is true that the FPP tool can generate overindexed loops, the user must write the code in such a way that it satisfies FPP's somewhat stringent criteria for overindexing:

- tightly nested loops having one loop index per array dimension
- inner loop bounds, which are identical to the array bounds
- vector array references having the same subscripting

FPP will generate overindexed code when the user passes the arrays and their dimensions, as shown in the FPP accessible code below:

```

SUBROUTINE BSET(NX, NY, NZ, UN, VN, WN)
COMMON /CTRL/NX/NY, NZ, . . .
REAL UN(NX, NY, NZ), VN(NX, NY, NZ), WN(NX, NY, NZ)
DO 200 J=1, NY
  DO 200 I=1, NX
    . . .
200 CONTINUE

```

In practice, user code is generally not this clean. FPP's second and third criteria (listed above) are overly restrictive--overindexing can replace standard code for any DO-loop bounds that legitimately reference a continuous section of memory occupied by an array. Overindexing can include various offsets, as long as such indexing spans a continuous memory interval. An expression of the form:

$$V(IZZ+1,1)=UV(IZZ+2,1)/BV(IZZ+4,1)$$

is legitimate.

Overindexing a Section of an Array

The candidate DO-loop need not span the entire array. For example, suppose the user wishes to visit columns 2 through n-1 and row elements beginning with the second element. Here is the standard code:

```

SUBROUTINE BSET
PARAMETER (IX=163, IY=39, IZ=75, IQ=11)
COMMON /CTRL/NX, NY, NZ, . . .
COMMON /VAR/UN(IY, IZ, 10), VN(IY, IZ, 10), WN(IY, IZ, 10), UV(IY, IZ, 4, IQ)
ISJ=2
IFJ=NY
DO 1003 K=2, NZ-1
  DO 10030 J=ISJ, IFJ
    UN(J, K, KK) = UV(J, K, K4, 2) / UV(J, K, K4, 1)
    VN(J, K, KK) = UV(J, K, K4, 3) / UV(J, K, K4, 1)
    WN(J, K, KK) = UV(J, K, K4, 4) / UV(J, K, K4, 1)
10030 CONTINUE
  ISJ=1

```

```

      IF (K.GE. NNZ-2) IFJ=NNY-1
1003 CONTINUE

```

While this loop nest has a somewhat complicated structure, examination indicates that this construct accesses a continuous region of memory. Here is the overindexed code:

```

SUBROUTINE BSET
PARAMETER (IX=163, IY=39, IZ=75, IQ=11)
COMMON /CTRL/NX, NY, NZ, . . .
COMMON /VAR/UN/IY, IZ, IX), VN(IY, IZ, IX), WN(IY, IZ, IX), UV(IY, IZ, 4, IQ)
DO 1003 IZZ=1, NY*(NZ-2)-2
  UN(IZZ+1, 2, KK)=UV(IZZ+1, 2, K4, 2)/UV(IZZ+1, 2, K4, 1)
  VN(IZZ+1, 2, KK)=UV(IZZ+1, 2, K4, 3)/UV(IZZ+1, 2, K4, 1)
  WN(IZZ+1, 2, KK)=UV(IZZ+1, 2, K4, 4)/UV(IZZ+1, 2, K4, 1)
1003 CONTINUE

```

Some users may be able to inspect their codes to determine the upper limit for the overindexed loop, while others may need to rely on a simple manual simulation. In the above example, the second index is set to 2 because the access begins with the second column.

For the previously given values of nx, ny, and nz, the HPM reports that the standard code gives 209 M/Flops with a vector length of 40, whereas the overindexed code performs at 344 M/Flops with a vector length of 122.

More Hints

The following additional hints were not included in the printed version of NAS News.

Although FPP will perform overindexing automatically for very clean code, the semantic restrictions on this tool really establish explicit user overindexing as the most assured way to obtain long vector lengths.

A large CFD code may require several weeks to fully implement overindexing, but a significant performance improvement may occur after overindexing only one subroutine. Start with the most CPU-intensive loops that display short vector lengths. Cray provides two user tools, *perftrace* and *prof*, to assist in this determination, and the UNIX man pages document these two tools.

Users may find it helpful to construct test loops to ensure that overindexed code executes the same number of iterations and gives the same answer as the standard code. It may also be useful to construct diagrams of memory layout for various arrays to assist in understanding how the overindexed loop accesses memory.

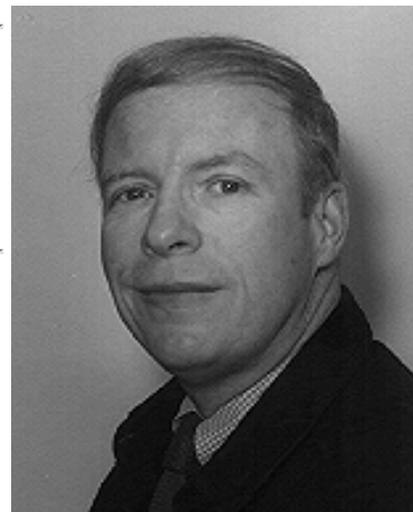
Robert Bergeron works for Computer Sciences Corp. in the NAS Scientific Computing Branch. He is working on specifications for the next generation of high-speed processors.

[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)



[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)

NAS Parallel Benchmarks Set Industry Standard for MPP Performance

by [Elisabeth Wechsler](#)

The latest NAS Parallel Benchmarks (NPB) results give the comparative performance of various supercomputers manufactured by different vendors.

The purpose of the NPB is to measure and report performance of the latest commercially available high-performance computing (HPC) systems, explained Leonardo Dagum, NAS research scientist in charge of NPB testing and analysis. Eight mathematical problems, consisting of five kernels and three simulated computational fluid dynamics (CFD) applications, are used to compute the NPB results.

Architecture Independent

The NPB were developed in 1991 to evaluate the performance of parallel computing systems for workloads typical in the NASA and aeronautics communities. Because the NPB are architecture independent, some freedom in selecting data structures, language constructs, and algorithms is possible.

Another unique feature of the NPB is the requirement for vendors to supply the list price for the systems tested to enable price-performance comparisons. Since HPC systems can span two orders of magnitude in price as well as performance, comparisons based solely on performance results would be meaningless, according to Dagum.

"The combination of an architecture-independent specification and pricing requirement has allowed comparison of performance results across a wide range of platforms on a uniform basis," he said. "To date, the NPB have been implemented by 12 different vendors on 23 different systems with prices ranging from \$250,000 to \$30.9 million, and parallelism ranging from 1 to 65,536 CPUs."

Valid Representation of Performance

The NPB have gained acceptance within the massively parallel processor (MPP) industry as a valid representation of system performance on a wide range of scientific workloads, not restricted just to aeronautics, Dagum said. The NPB results have increasingly been used as criteria for HPC procurements in private industry and academia. The [latest NPB results](#) are always available on the World Wide Web (WWW) and are mailed to more than 160 addresses around the world.

Dagum sees the NPB as "a classic example of second-use technology transferred from government to the private sector. In this case, NASA has expanded its role of serving the aeronautics community to one of serving all potential users of HPC systems."

"Through the NPB results, NASA provides a credible measure of performance on a wide range of scientific applications. This information has been of tremendous value not only to HPC users but to vendors as well -- many of whom have used the NPB results to aid in their own internal development of current and future systems," he said.

Kernel Problems

The five kernel problems are relatively compact, each emphasizing a particular type of numerical computation. Compared with the simulated CFD applications, they can be implemented fairly readily and provide insight on the general levels of performance that can be expected on these specific types of computations.

Here are brief explanations of the five kernel problems:

EP is an embarrassingly parallel problem. In this benchmark, two-dimensional (2D) statistics are accumulated from a large number of Gaussian pseudorandom numbers, well-suited for parallel computation. This problem is typical of many Monte Carlo applications. Since it requires almost no communication, this benchmark provides, in some sense, an estimate of the upper achievable limits for floating-point performance on a particular system.

MG is a simplified multigrid kernel, which solves a 3D Poisson partial differential equation. This problem is simplified in the sense that it has constant rather than variable coefficients as in a more realistic application. This code is a good test of both short- and long-distance highly structured communication.

CG uses a conjugate gradient method to compute an approximation to the smallest eigenvalue of a large, sparse, symmetric positive definite matrix. This kernel is typical of unstructured grid computations in that it tests irregular long distance communication performance and employs sparse matrix vector multiplication.

FT solves a 3D partial differential equation using Fast Fourier Transforms (FFTs). This kernel performs the essence of many "spectral" codes and is a good test of long-distance communication performance. The rules of the NPB specify that assembly-coded library routines may be used to perform matrix multiplication and 1D, 2D, or 3D FFTs. Thus, this benchmark is somewhat unique in that computational library routines may be employed.

IS tests a sorting operation that is important in particle method codes. This type of application is similar

to "particle in cell" applications of physics, in which particles are assigned to cells and may drift out. The sorting operation is used to reassign particles to the appropriate cells. This benchmark tests both integer computation speed and communication performance. This problem is unique in that floating-point arithmetic is not involved; however, significant data communication is required.

Simulated CFD Applications

The three simulated [CFD applications](#) usually require more effort to implement, but they are more indicative of the types of actual data movement and computation required in state-of-the-art CFD application codes, Dagum explained. "For example, in an isolated kernel, a certain data structure may be very efficient on a certain system. Yet, this data structure would be inappropriate if incorporated into a larger application. By comparison, the simulated CFD applications require data structures and implementation techniques more typical of real CFD applications."

Here are brief explanations of the three simulated CFD applications, which are intended to accurately represent the principal computational and data movement requirements of current production CFD applications:

LU (or lower-upper diagonal) application employs a symmetric successive over relaxation (SSOR) numerical scheme to solve a regular-sparse, block (5 x 5) lower and upper triangular system. This problem represents the computations associated with a newer class of implicit CFD algorithms, typified at NASA Ames Research Center by the INS3D-LU code. This problem exhibits a somewhat limited amount of parallelism, compared with SP and BT.

SP (or scalar pentadiagonal) application solves multiple independent systems of nondiagonally dominant, scalar pentadiagonal equations.

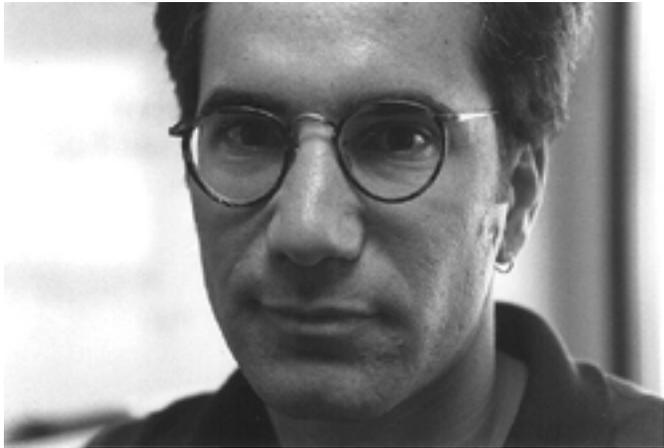
BT (or block tridiagonal) application solves multiple independent systems of nondiagonally dominant, block tridiagonal equations with a 5 x 5 block size.

The SP and BT benchmarks are representative of computations associated with the implicit operators of CFD codes, such as ARC3D at Ames. Although similar to BT in most respects, SP has a greater communication-to-computation ratio and therefore is more demanding on parallel systems.

Full Descriptions of NPB Available

A more detailed description of these benchmarks, together with the rules and restrictions associated with them, can be found in "The NAS Parallel Benchmarks," by David Bailey, *et al.*, *International Journal of Supercomputer Applications*, Vol. 5, No. 3. The full specification of the NPB is published in [The NAS Parallel Benchmarks, Technical Report RNR-94-007](#). Both are also available from the NAS Documentation Center. To request a copy, send email to: **doc-center@nas.nasa.gov**. For more

information, send email to Leonardo Dagum at leo@nas.nasa.gov.



Leonardo Dagum manages NPB testing and analysis.

[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)

[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)

Forssell Gives 'Best Paper' at Visualization '94

by Jean Clucas



The fifth IEEE scientific visualization conference, Visualization '94, held in Washington D.C. October 17-21, provided an opportunity for developers of visualization techniques, as well as those who use the techniques, to exchange information on the latest visualization research. NAS was well represented by researchers from the Data Analysis Branch.

Lisa Forssell, who works for Computer Sciences Corp. (CSC) and is a doctoral candidate at Stanford University, presented a paper, entitled "Visualizing Flow Over Curvilinear Grid Surfaces Using Line Integral Convolution," which was voted by attendees as the best paper at the conference. The paper describes Forssell's work in line integral convolution, a technique for displaying detailed

flow over a surface. Previously, this technique had been limited to regular, low-dimensional Cartesian grids.

Forssell extended the technique to display vector fields on curved surfaces, as are used in computational fluid dynamics simulations. The compatibility of this technique with texture-mapping hardware on graphics workstations allows real time animations of the flow on an object's surface at the same time that the object is being rotated or translated.

Tied for first place was the paper "The Topology of Symmetric, Second-Order Tensor Fields," by Thierry Delmarcelle and Lambertus Hesselink of Stanford University, whose work was partially funded under the NASA Research Announcement grants, sponsored by NAS.

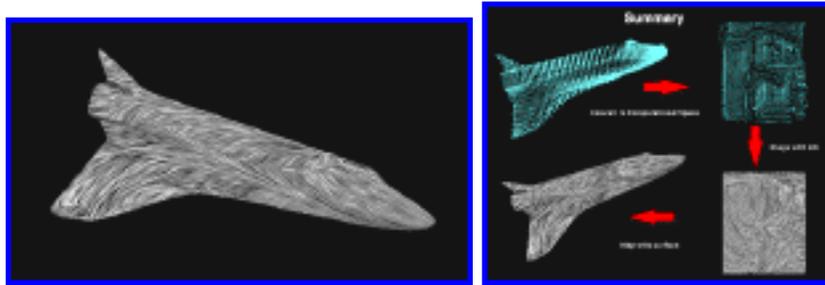
David Lane presented a paper, "UFAT--A Particle Tracer for Time-Dependent Flow Fields." The UFAT visualization system generates particle traces in time-independent flow fields. Lane observed streamlines, streaklines, timelines, and particle paths in several datasets and showed that these can be interpreted differently in time-dependent unsteady flow.

Associated with the conference on even-numbered years is the Volume Visualization Symposium. Sam Uselton chaired a symposium panel on "Hardware Assisted Rendering" and also chaired a panel in the main conference, entitled "Validation, Verification, and Evaluation." Al Globus was a panelist in the latter session.

David Kenwright, who attended the "Validation, Verification, and Evaluation" panel, felt that its focus on looking critically at visualization showed that this five-year-old conference is "coming of age."

"People are now realizing that the images we created should not be accepted at face value," Kenwright said. He remarked that several presenters had noted errors in their algorithms--of particular interest to Kenwright--who's working on verification and error analysis of particle tracing algorithms in UFAT.

More information about [NAS visualization work](#) is located on the World Wide Web.



[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)

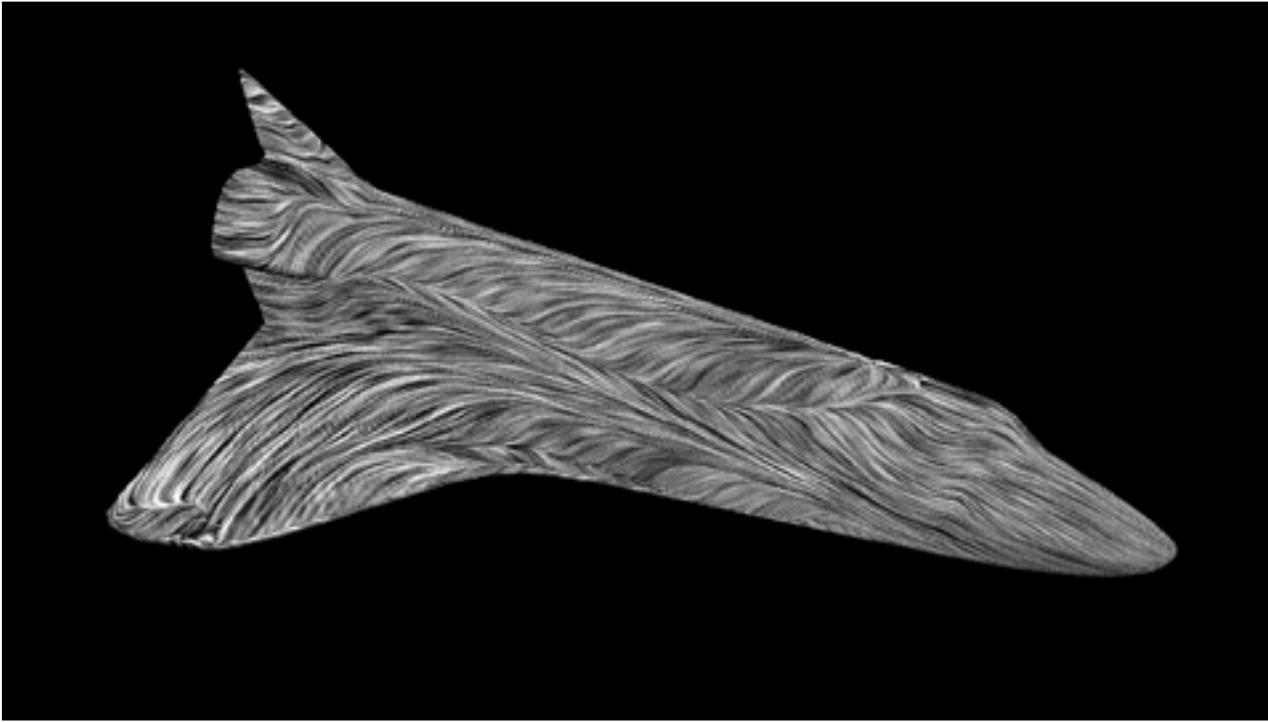
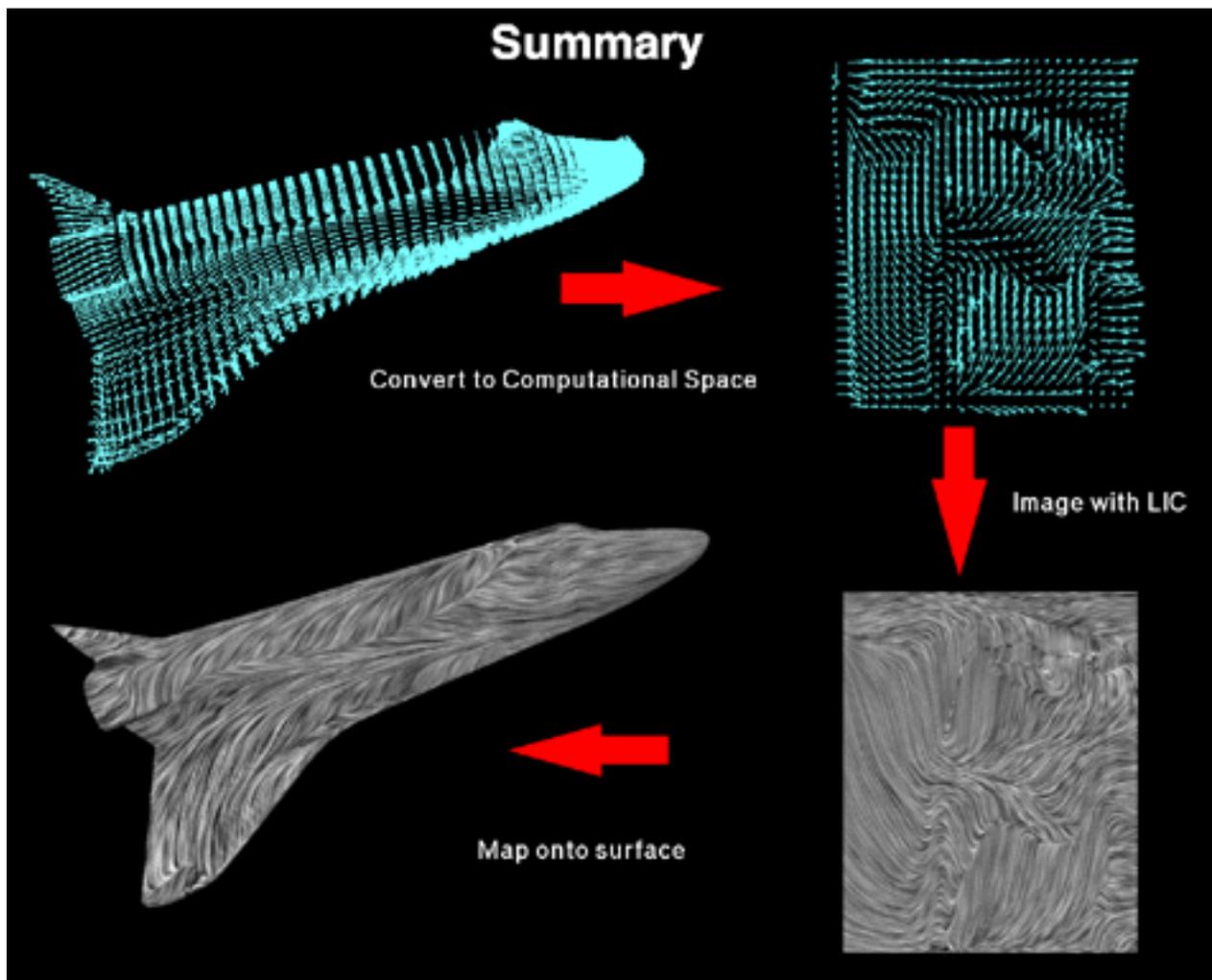


Image shows the flow over the surface of a space shuttle. The image was produced by Lisa Forssell, who combined an image processing technique called Line Integral Convolution with texture-mapping hardware to display the flow on curved surfaces. The visualization is part of an interactive program that can also do real-time animation of the flow.



[to the article.](#)



In Forssell's process, the vectors on the surface plane of a curvilinear grid are converted to the computational space. Line Integral Convolution is done for the flow in that plane, and the result is texture-mapped back onto the grid. The swirling lines on the resultant surface represent the actual flow on the surface, producing a more information-dense image than the usual arrows to indicate vectors at grid points.

 [to the article.](#)

[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)

Supercomputing '94 Underscored Challenges Ahead

The NAS Facility was well represented at Supercomputing '94, held in Washington, D.C., November 14-18, with research and support staff presenting eight tutorials and seven papers. The conference, which gave special emphasis to biomedicine and the environment, was nine percent larger than last year--with 20 vendor booths, more than 40 research sites, and some 6,000 visitors who browsed the exhibit hall.

[Photos of NAS Booth at Supercomputing '94](#)

Below is a small sampling of the numerous (often concurrent) sessions presented during the conference. Future issues of NAS News will include more about those presentations of special interest to the aeronautics community.

- [Keynote Address by Ed McCracken of Silicon Graphics](#)
- [Highlights of an Informal IBM SP2 User Meeting](#)
- [Workstations Closing the Performance Gap](#)

[Elisabeth Wechsler](#) covered Supercomputing '94 for *NAS News*.

Ability to Compute and Communicate Results is Key

Supercomputers have succeeded in penetrating a number of scientific computing spheres--such as biomedicine and the environment--as evidenced at Supercomputing '94. However, at least one industry leader sees the trend for supercomputing as "no longer the biggest and fastest [machine], but the ability to compute and communicate the results."

Ed McCracken, CEO of Silicon Graphics Inc. (SGI), gave the conference keynote address, "Making the National Information Infrastructure (NII) Real." His remarks also reflect his role as co-chair of the U.S. Advisory Council on NII, in which he obviously enjoys prodding the industry to adopt some new ways of thinking.

The end of the Cold War has brought different national priorities, he said. "It's no longer a case of progress at all costs," which he believes has kept the industry focused on "squeezing out more performance at any cost." Now, he says, "the fastest may not be the most expensive."

"Big Science" is in danger because federal funding is drying up, he added. McCracken believes that the private sector will play an increasingly important role in the future of supercomputing and noted a shift in supercomputing usage from academic research to general consumers, citing the entertainment industry and financial markets as examples of a broadening base.

Not surprisingly, one of the keys to McCracken's solution is distributed computing power. (In the last few years SGI workstations have become major contenders in the highly competitive arena of high-performance computing.) He pointed to high-end workstations' increasingly impressive performance when compared to traditional supercomputers, and also the use of graphics to better communicate the computational results.

He also underscored the importance of network technology, in general, and the Internet, in particular, for displaying data more effectively through the World Wide Web.

In emphasizing the need for "cost-effective computation cycles," McCracken urged the supercomputing industry not "to artificially limit" itself to any one architecture. He sees the array of users and problems to solve continuing to expand rapidly in the near future.

If code is written on a workstation, then it will be compatible for analysis on a supercomputer, McCracken said. He envisions "smaller, cheaper, and easier-to-use supercomputers" and warned traditional vector supercomputer manufacturers against becoming "isolated."

While he acknowledged that "main memory and latency are still the preserve of vector supercomputers," he noted that micro-processors are closing the gap.

McCracken urged the industry to emphasize "principles of inclusion, flexibility, and exploitation" to change the approach to supercomputing." His goal is clearly "megaflops for the masses" and his advice to the industry, in terms of research funding criteria, is "don't support what the market won't."



[to SC94 Overview](#)

SP2 Users Compare Notes

Two informal meetings of IBM SP2 users were held November 16 during Supercomputing '94 to discuss topics of mutual interest.

One "birds of a feather" session was organized by Richie Jacobovits, IBM program manager for the Cooperative Research Agreement (CRA), based in part at the NAS Facility. The purpose of the session was to provide research updates from various consortium partners and to share information related to using the SP2 system. Most users reported that they are still trying to port their codes to the SP2. About 25 CRA representatives attended this two-hour session.

The other session was organized by Chuck Niggley, of the NAS Scientific Computing Branch, and was attended by more than 100 SP1 and SP2 users from around the world. The main focus was on system management issues. Fifteen representatives from NASA research centers, national research laboratories, and universities gave short presentations about their system configurations, major problems they were experiencing, and future concerns.

The session then split into smaller groups to focus on those concerns that users had mentioned most frequently in their presentations, including:

- resource management, queuing, and job scheduling, including LoadLeveler
- parallel I/O
- message passing, including MPI and PVMe, the IBM version of Parallel Virtual Machine
- documentation, training, installation, diagnostics, system upgrades

Niggley is pursuing the idea of an ongoing SP2 users' forum. Attendees provided their email addresses so that he can set up an email alias for future discussions. Interest was expressed in holding another meeting in early March, possibly during the CRA consortium partners meeting at NASA Ames.

If you're interested in participating in an SP2 users' forum and did not attend the November meeting, send email to: niggley@nas.nasa.gov.



[to SC94 Overview](#)

Mhead>

Benchmarks Show Workstations Closing Performance Gap

"Due to the dramatic rise of workstation performance, supercomputing vendors must stop emphasizing throughput on small and medium-size jobs, or else workstations will put them out of business," said David Bailey, NAS research scientist, at Supercomputing '94. Among the advantages of using high powered workstations instead of traditional supercomputers are reliability and ease of programming, he added.

Workstations made from RISC [Reduced Instruction Set Computer] processors have "come up remarkably" in sustained performance in the last year because of attention paid to main memory bandwidth, Bailey noted. "You can see why they're being used, especially when price-performance information is considered. Furthermore, parallel computers that incorporate RISC processors currently exhibit the highest sustained performance per dollar among high-end supercomputers."

He urged users and vendors to "define those problems that supercomputers should be doing and others that should be done on workstations."

Bailey made his remarks during an all-day tutorial on November 14, entitled "The Science of Benchmarking." He and Roger Hockney, of Southampton University (UK) discussed different methods used to measure supercomputing performance and their pros and cons, as well as future challenges in providing meaningful supercomputing performance results.



Roger Hockney (left), of Southampton University, UK, and David Bailey, of NAS.

In his talk, Bailey compared NAS Parallel Benchmarks (NPB) results for the IBM SP2, the CRAY T3D, and the Silicon Graphics Inc. R8000, and commented on the characteristics of each system's architecture in achieving that performance.

Hockney presented the PARKBENCH Report (to which NPB results are submitted) and cautioned the audience not to rely too much on manufacturers' "speed of light" measurement--or the theoretical peak performance that can't be exceeded--as an accurate prediction of attainable results.

He emphasized that it's important to "be specific about what is being run on what system for what

benchmark, because all aspects are specialized."Hockney thinks the most useful benchmarks are those that operate at a "low level"because they are the "easiest to analyze, interpret, understand, and implement efficiently."Moreover, he believes that "the least useful benchmarks are applications that need to be scaled."

Bailey emphasized the need for "integrity, objectivity, and reproducibility" in tracking supercomputing performance. He referred to [*Twelve Ways to Fool the Masses*](#), his essay that pokes fun at the prevalence of hype in performance reporting. He cautioned researchers to be careful in comparing results, not to show results out of context, and to carefully define what is being measured. Bailey also acknowledged that different levels of code tuning can distort performance.

Future benchmarking measurements need to better address I/O, system integration, and software development resources, Bailey said. In addition, "benchmarks should reflect the size and scope of real problems that are expected to be done in the next few years."

"With regard to the issue of algorithm changes for parallel computers, the best approach is to find a numerically efficient algorithm on a serial computer, and then develop a parallel implementation,"Bailey said.



[to SC94 Overview](#)

NAS Booth at Supercomputing '94

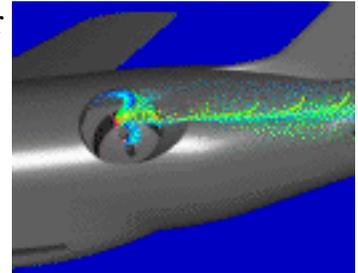
A view of the NAS booth at Supercomputing '94, taken before the exhibit hall opened. The 30 ft x 30 ft booth, with its three 16-ft towers, drew thousands of conference attendees to its (from left) stereoscopic projection (see below), CFD, and FASTexpeditions demonstrations. Other demos presented in the NAS booth (not shown) include the Remote Access Wind Tunnel (RAWT) running under Integration of Numerical and Experimental Wind Tunnels (IofNEWT), portable parallel/distributed debugger (p2d2), NTV, AIMS, Virtual Wind Tunnel, and Information Assistant.



Stereo Projection Adds Extra Dimension to NAS Booth



Attendees in a multimedia visualization theater wear special 3D glasses to view a series of CFD datasets that will ultimately impact aircraft design techniques. The theater, currently being developed at NAS, will provide researchers with a presentation and collaboration tool to allow interactivity and greater resolution than conventional videotape. The



five-minute program drew more than 1,000 visitors. The SOFIA (Stratospheric Observatory for Infrared Astronomy) dataset (right) shows a cavity cut into the side of a Boeing 747 to house a telescope. The aircraft flies above much of the earth's atmosphere to minimize the effects of atmospheric distortion. 3D viewing allows greater understanding of the cavity shape and surrounding airflow.



[to SC94 Overview](#)

[Next Article](#)[Contents](#)[Main Menu](#)[NAS Home](#)

Using Recursive FORTRAN Subprograms on the CRAY C90

by Clayton J. Guest

"Recursion " is a feature that allows a procedure to repeatedly call itself. This feature is desirable for iterative procedures, and reduces the amount of coding required.

A computation such as a factorial, $N! = N(N-1)..(1)$, is an example where recursive functions are useful.

For years, the Fortran language has been criticized for not having a recursive subprogram facility. The ANSI Fortran 77 standard didn't include recursion-- but the ANSI Fortran 90 standard requires it. When Cray Research Inc. (CRI) released CF77 version 5.0 several years ago, recursion became a part of Cray Fortran 77. Both the CF77 and F90 compiling systems are available on the NAS CRAY C90.

The examples that follow show the form of recursive subprograms. In all cases, text enclosed in square brackets [] is optional.

Syntax for Recursive Subroutines

Syntax for recursive subroutines in Fortran 77 and 90 are identical except that the keyword RECURSIVE is mandatory in Fortran 90. The general form of a subroutine statement is:

```
[RECURSIVE] SUBROUTINE subrt_name [ (arg1 [, arg2] ... ) ]
```

where subrt_name is the subroutine name and arg1, arg2... are optional dummy argument names.

Syntax for Recursive Functions

Recursive syntax for Fortran 77 functions has the following two forms:

```
[RECURSIVE] FUNCTION [type] funct_name [(arg1 [, arg2] ...)]
```

and

```
[type] [RECURSIVE ] FUNCTION funct_name [(arg1 [, arg2] ...)]
```

where `funct_name` is the name of the function. The keyword `RECURSIVE` is optional; *type*, if used, declares the type of function, where *type* can be any legitimate data type, such as `INTEGER` or `REAL`; and `arg1`, `arg2`...are optional dummy arguments.

It is recommended that the keyword `RECURSIVE` be used for clarity and upward compatibility.

Recursive syntax for Fortran 90 function statements has the following two forms:

```
RECURSIVE FUNCTION funct_name [(arg1 [, arg2] ...)] &
    [RESULT (name_of_result)]
```

and:

```
RECURSIVE [type] FUNCTION funct_name [(arg1 [, arg2] ...)] &
    [RESULT (name_of_result)]
```

where `funct_name`, `type`, and `arg1`, `arg2`... have the same meaning as in Fortran 77. `RESULT` (a Fortran 90 innovation that eliminates the requirement to give the result the same name as the function name) is optional; if used, then the returned value from the function is `name_of_result`.

The following Fortran 77 program, while perhaps considered to be a "game," illustrates the use of recursive subroutine calls; these calls reduce the amount of Fortran programming effort.

The program solves the "Towers of Hanoi" problem, in which the goal is to move any number of different sized discs from one tower to a final tower and maintain the discs in the same order. The following rules apply:

- There are three towers (posts).
- Discs on the first tower are in ascending order, smallest at the top.
- Discs may only be moved one at a time from one tower to another.
- A larger disc can never be placed on top of a smaller one on the same tower.

Here's the general scheme:

1. Move the top $n-1$ discs from the first post to the free post.
2. Move the biggest disc from the first post to the last post.
3. Move the $n-1$ disks from the free post to the last post.

```
PROGRAM HANOI_TOWERS
INTEGER DISCS
WRITE (6, "('Type the number of discs. zero to quit', $)')
```

```

READ *, DISCS
IF (DISCS .EQ. 0) STOP
PRINT *, "Number of discs=", DISCS
CALL HANOI (DISCS, 1, 3)
WRITE (6, "('All Moves Complete.')" )
END
RECURSIVE SUBROUTINE HANOI (DISCS, POST1, POST3)
INTEGER DISCS, POST1, POST3, POST2, POSTS
! POST1 = starting post, POST2 =free post,
! POST3 = Goal post, POSTS =all Posts
PARAMETER (POSTS = 6)
POST2 = POSTS - POST1 - POST3
IF (DISCS .GT. 1) CALL HANOI (DISCS -1, POST1, POST2)
PRINT *, "Move disc ", DISCS, " from post ", POST1, " to post ", POST3
IF (DISCS .GT. 1) CALL HANOI (DISCS -1, POST2, POST3)
END

```

Here is the execution result:

```

Type the number of discs. zero to quit.      3
Number of discs=3
Move disc 1 from post 1 to post 3
Move disc 2 from post 1 to post 2
Move disc 1 from post 3 to post 2
Move disc 3 from post 1 to post 3
Move disc 1 from post 2 to post 1
Move disc 2 from post 2 to post 3
Move disc 1 from post 1 to post 3
All Moves Complete.

```

The above example shows that with very little programming effort, the use of a recursive function can reduce a potentially lengthy Fortran program to a much shorter program.

For more information on recursive subprograms, see CRI's Fortran 77 and Fortran 90 reference manuals, or contact NAS User Services at (415) 604-4444, or send email to: nashelp@nas.nasa.gov.

[Next Article](#)
[Contents](#)
[Main Menu](#)
[NAS Home](#)

[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)

New AIMS Version Now Available to NAS Users

The latest release of AIMS, the Automated Instrumentation and Monitoring System, developed at NASA Ames Research Center, is now available to NAS users. AIMS is a software toolkit that facilitates performance evaluation of parallel applications on multi-processors. Used together, the tools allow users to instrument, measure, and display program performance. AIMS development work was funded by the High Performance Computing and Communications Program and is currently supported by Melisa Schmidt, Brian VanVoorst, and Jerry Yan (all of Recom Technologies Inc. in NAS's Scientific Computing Branch).

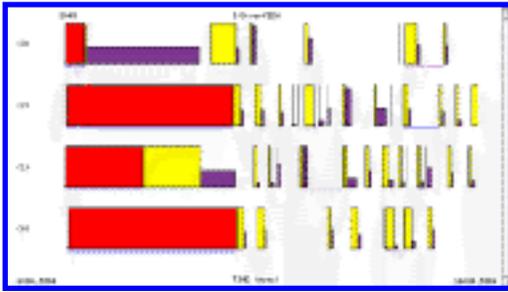
Features in AIMS version 3.1 include:

- selective source code instrumentation
- I/O monitoring
- support for PVM message-passing programs on distributed heterogeneous systems
- support for MPI message-passing programs on the IBM SP2
- additional trace file animation and analysis tools

Automates Instrumentation

The AIMS source code instrumentor, "xinstrmt," automatically inserts event recorders to trace subroutine invocations, synchronization operations, and message passing. Version 3.1 uses Indiana University's Sage/Sigma toolkit, which will soon allow AIMS to easily handle data-parallel languages such as High Performance Fortran and PC++.

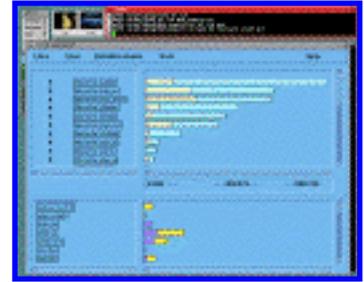
Monitors I/O Operations



AIMS now monitors I/O operations, such as filesystem read and write times. This new information can be displayed on the View Kernel, shown in [Figure 1](#). The new monitor also provides a mode that reduces trace size (and, therefore, monitor overhead) by gathering only cumulative performance statistics at specified points during program execution.

'Xisk'--A New Scalable Profiling Tool

As shown in [Figure 2](#), "xisk" in AIMS 3.1 presents aggregate timing data for activities such as program execution, communication blocking, and global operations. In addition, new performance metrics (or indexes) such as the ratio of communication vs. computation for certain code blocks can also be defined and displayed.



AIMS Class Planned for February

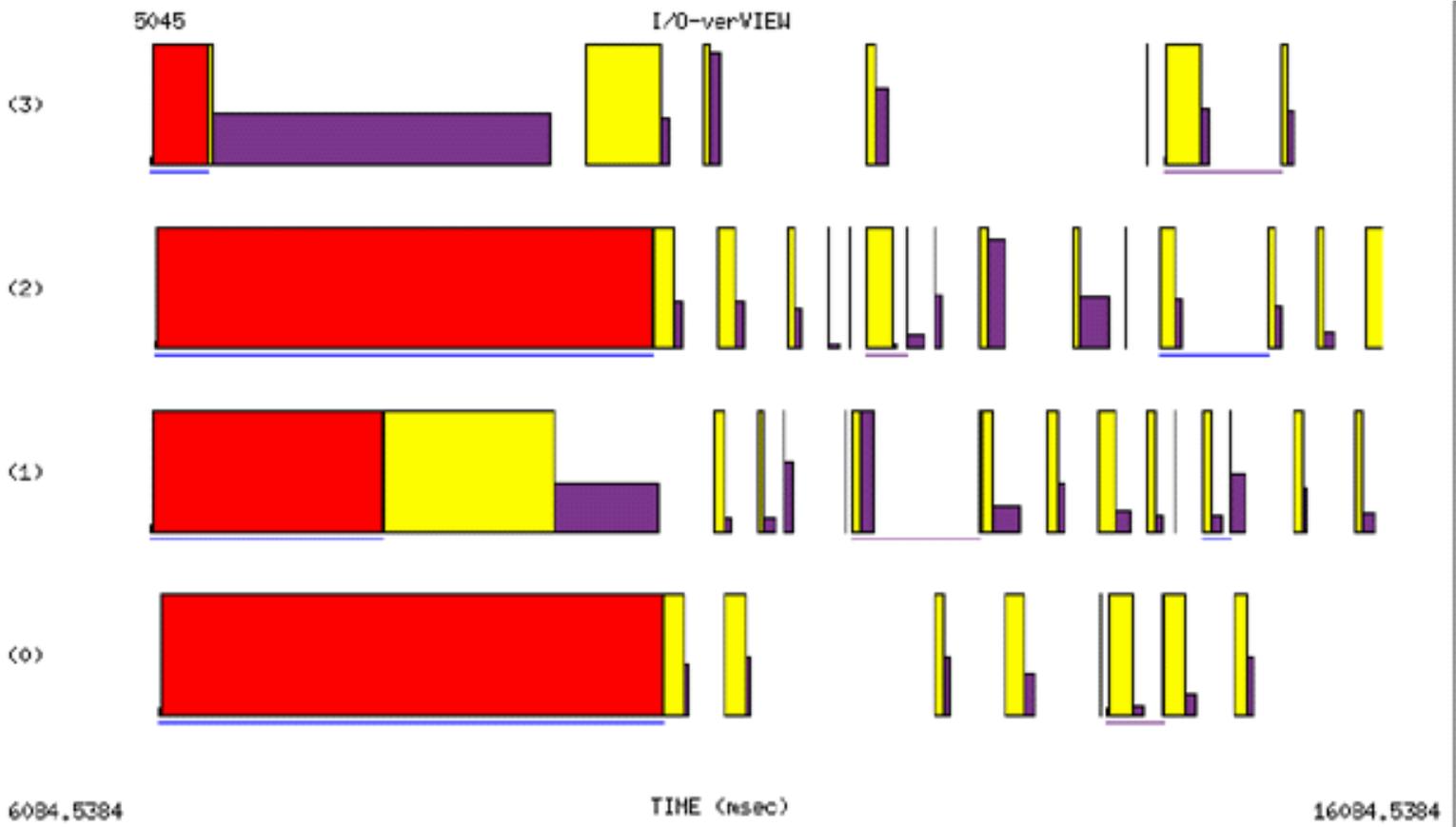
Yan will present a one-day AIMS training class for NAS users in February. See "[Upcoming Training Events](#)" for course content. For further class information, send email to redmond@nas.nasa.gov.

[Next Article](#)

[Contents](#)

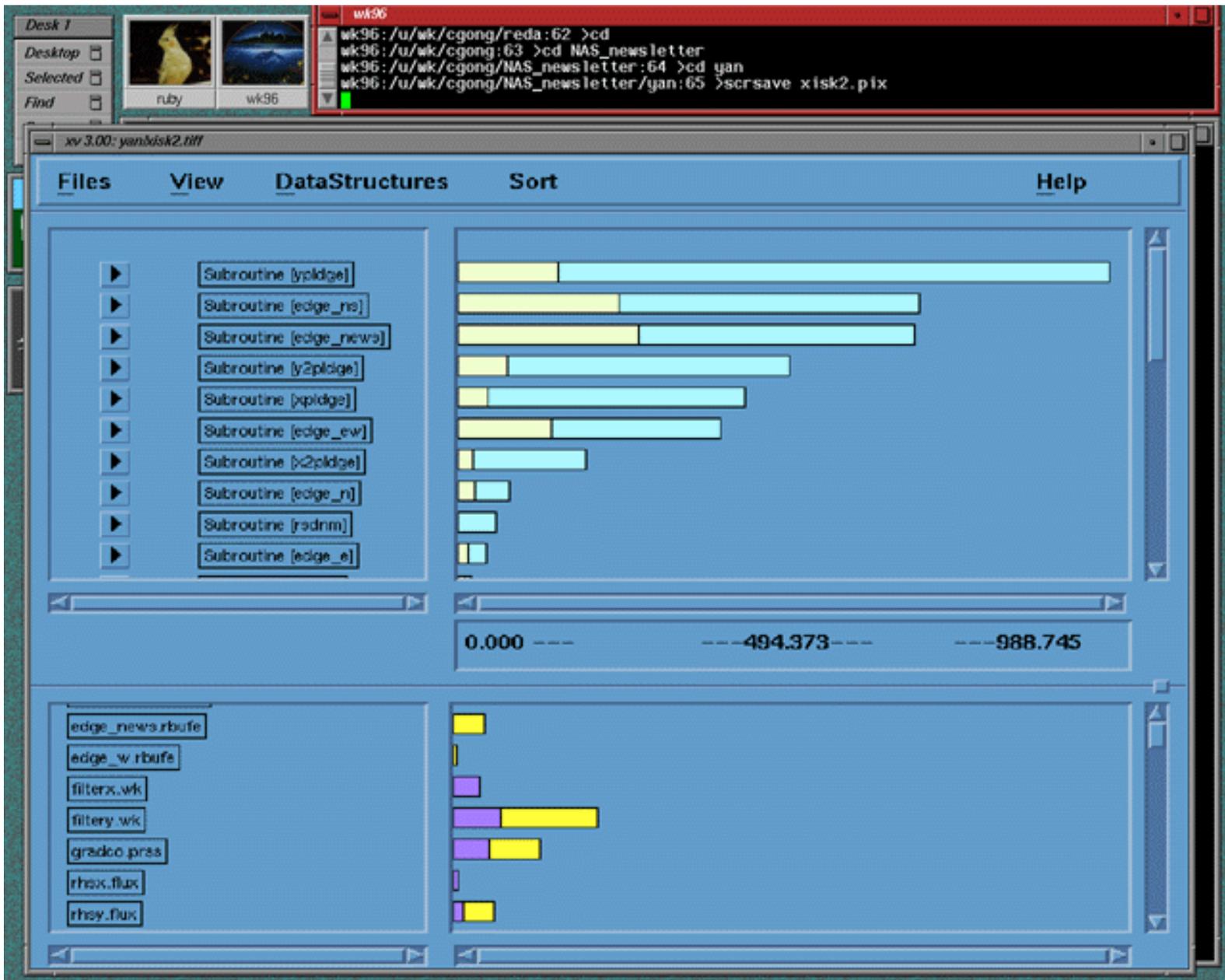
[Main Menu](#)

[NAS Home](#)



AIMS 3.1 View Kernel's I/Overview uses different colors to distinguish various I/O activities such as read, write, io_wait, and lseek. The height of each column indicates the number of bytes transferred during the I/O operation.

 [to AIMS Article](#)



Xisk highlights time-consuming subroutines and associates communication costs with individual data structures to facilitate bottleneck detection.

 [to AIMS Article](#)

[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)

Upcoming Training Events

AIMS Class

Speaker: Jerry Yan, Recom Technologies Inc.

Date: February 23, 1995

Topics:

- Overview: Using AIMS for Performance Tuning and Debugging
- Examples and Usage: MPI/SP2, PVM/SGI Cluster
- A Preview of Things to Come: Performance Tuning and Production
- Tools to be released in 1995

Hands-on time is available on request. If you're interested in having code instrumented and tested, contact Jerry Yan at yan@nas.nasa.gov in advance of the class date.

See the article, "[New AIMS Version Now Available to NAS Users.](#)"

Mark these dates on your calendar:

NAS New Technology Seminars

January 5

January 10

January 17

The [Training home page](#) always has the most up-to-date information on NAS/CCF training events.

[Next Article](#)

[Contents](#)

[Main Menu](#)

[NAS Home](#)

NAS News, Jan - Feb 95

Vol. 2, No. 8

Executive Editor: Marisa Chancellor

Editor: Jill Dunbar

Senior Writer: Elisabeth Wechsler

Contributing Writers: Robert Bergeron, Jean Clucas, Clayton J. Guest, Alan Powers

Image Enhancements: Chris Gong

Other Contributors: David Bailey, David Barkai, Ryan Border, Jim Crow, Leonardo Dagum, Dennis DeRyke, Pat Elson, Lisa Forssell, Jeanne Fouts, Steve Harding, John Hardman, Toby Harness, Eric Hibbard, Kevin McCabe, Chuck Niggley, Cathy Parks, Marcia Redmond, Jackie Scoggins, Cathy Schulbach, Horst Simon, Leigh Ann Tanner, Tom Whittaker, Jerry Yan

Editorial Board: Marisa Chancellor, Jill Dunbar, Kristina Miceli, Serge Polevitzky, Pamela Walatka, Elisabeth Wechsler, Rita Williams



NEWS

Volume 2, Number 8

January - February 1995

New File System Offers Increased User Productivity

by Alan Powers

NAS has created a "super home" file system on the NAS CRAY C90 (roomsman) by combining the functionality of the home and migrated file systems. This file system will help new users become immediately productive by reducing the number of file systems they must learn, and will also increase the total user disk capacity.

Some benefits of the super home file system are:

- Simplifies use of the system.
- Increases total home disk capacity and disk performance.
- Splits inode lookups for /in across four file systems, increasing file access speed and reducing overhead.
- Improves transfer rates to and from /big and /fast.
- Shortens backup and restore times.

Previously, two user file systems existed: homes (/u/v/node) and migrated (/in), as well as two temporary directory file systems, /big and /fast. This old configuration worked well for users who were aware of the purpose of each file system. However, when new users were added to the system, they tended to overuse their basic file systems to run batch jobs, causing job turnaround time to be longer—and sometimes causing their jobs to fail because of full file systems.

In mid-December, the homes and migrated file systems were combined to form the super home file system. A prototype implementation of this had been working on the /u/vb file system since late August 1994.

Previous Configuration Was Slow
Prior to the super home configuration, each home file system consisted of 6.3 gigabytes (GB) of slow-access (30 megabytes per

Continued on page 3



A view of the NAS booth at Supercomputing '94, taken before the exhibit hall opened. (For another view, see page 4.) The 20 ft x 30 ft booth, with its three 16-ft towers, drew thousands of conference attendees to its (from left) stereoscopic projection, CFD, and NAS operations demonstrations. Other devices presented in the NAS booth (not shown) include the Remote Access Wind Tunnel (RAWT) running under integration of Numerical and Experimental Wind Tunnel (NEWET), portable parallel assembler/debugger (PASC), MFC, NAS, Visual Wind Tunnel, and Information Assistant.

Overindexing Improves Performance on the NAS CRAY C90

by Robert Bergeron

For most NAS users, the demand for CRAY C90 CPU time exceeds the number of allocated hours, so many users would like their codes to perform better in order to make the most of their allotted time. Although NAS rates the C90 at a factor of 2.2 faster than the CRAY T-3E on workload codes, some users are unable to obtain such performance. A powerful, straightforward optimization technique, called overindexing, allows users to more effectively use their CPU allocations. Codes displaying Hardware Performance Monitor (HPM) vector lengths of less than 64 can most strongly benefit from this technique (although some codes with vector lengths of less than 64 may not benefit, due to their structure). Overindexing has produced CPU speedups of 1.3 to 1.9 on already well-written CFD codes.

Overindexing is the use of a single array index to perform the array addressing normally done by two or more DO-loop indexes. The technique has assumed additional importance under the C90 architecture: The peak performance of 1 gigaflop per CPU requires four floating-point operations per clock period and continuous saturation of the vector functional units. These units contain 64 double-width paired segments, with odd elements of the vector in one half and even elements in the other half, giving the architecture an effective vector length of 128. Under standard Fortran coding techniques, peak performance requires problem sizes to equal or exceed 128 in all dimensions in order to saturate the vector functional units,

so only users with large problem sizes can fully use the C90 computing power.

Overindexing reduces the number of DO-loops required to perform a given calculation, which in turn reduces the overhead associated with DO-loop execution and more fully exploits the vector hardware by achieving longer vector lengths. The optimization uses the Fortran treatment of computer memory as a columnwise continuous sequence of storage locations.

For simple loop constructs, the Cray compiler already performs some overindexing by default. As mentioned in the article "Linearizing Nested Loops," (NAS News, September-October 1994) the Fortran Preprocessor (FTP) can indeed modify user code to extend the range of constructs subject to overindexing. However, successful use of FTP to overindex a complete code demands a strong understanding of the way in which this tool extracts recognizable constructs from code.

The examples here illustrate how to recognize and improve performance of more complicated constructs by overindexing arrays explicitly. While the overindexing optimization is legitimate for Cray vector architectures, correct results on other architectures cannot be guaranteed.

Overindexing Doubly Nested DO-loops
NAS codes generally visit the points in a grid and perform a small set of operations on

Continued on page 7

THIS ISSUE

NAS Parallel Benchmarks
page 3

Supercomputing '94 Sampling
page 4

Using Recursive Fortran Subprograms on the CRAY C90
page 6