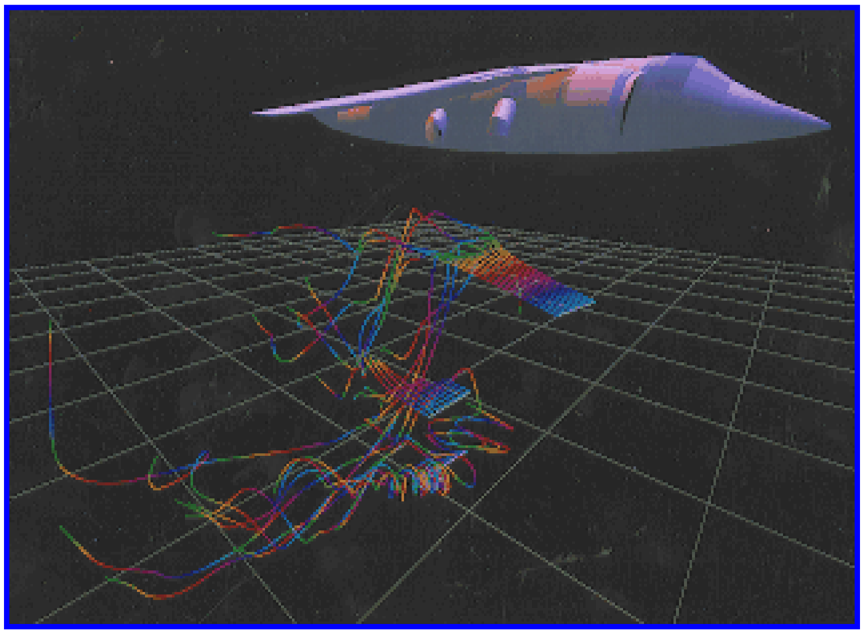
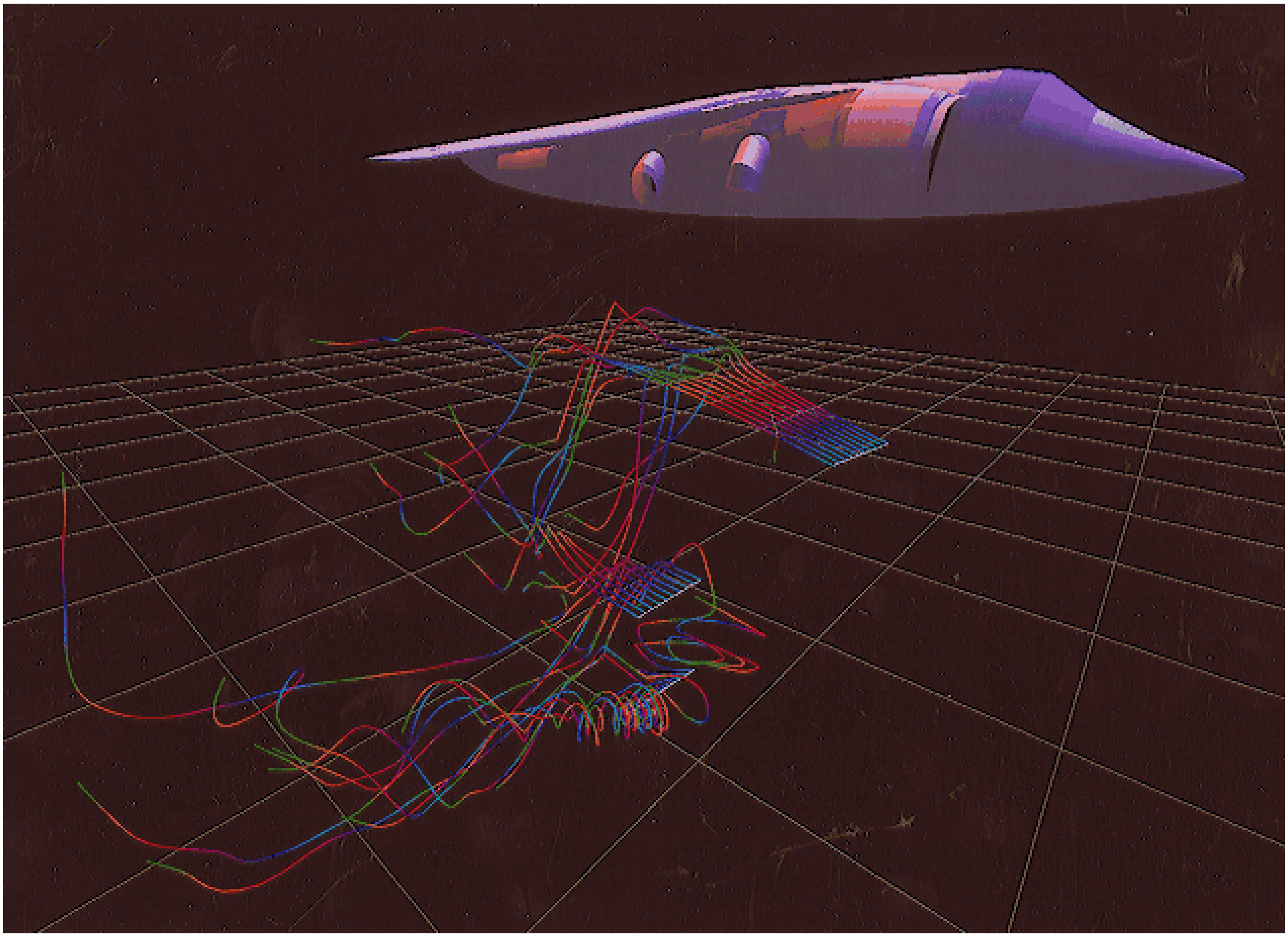


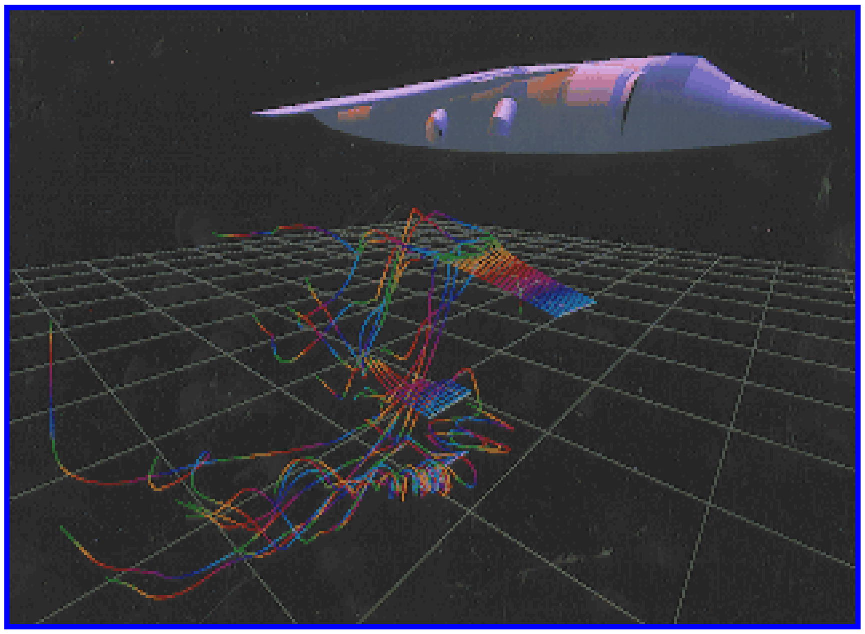
March - April 1994, Vol.2, No. 2



The virtual wind tunnel environment, showing the visualization of a data set of a Harrier aircraft in hover, with three rakes emitting streamlines of the velocity vector field. Vortices, where the jets from the aircraft hit the ground, are clearly visible. The computation was performed by Merritt Smith, Kalpana Chawla, and Bill Van Dalsem, all of the NASA Ames Fluid Dynamics Division.

Back to the article.

Next Article | Contents | Main Menu | NAS Home



Virtual Wind Tunnel Draws Crowds at Technology Transfer Expo

by Elisabeth Wechsler

Virtual Wind Tunnel Draws CrowdsA color demonstration of the NAS virtual wind tunnel was one of the most well attended exhibits at the fourth annual "Technology 2003" conference and expo, held December 7-9 in Anaheim, CA. The conference was attended by some 8,500 representatives of private industry, federal research labs, universities, and the general public.

Technology 2003 showcases commercially viable, cutting-edge technologies, developed by the nation's federal labs, which have potential application for private industry, according to Laura Weikle, Ames Office of Commercial Technology. Hundreds of exhibitors demonstrated and shared information for thousands of technologies (seven of which were located in the NASA Ames booth) at the 80,000-sq. ft. exhibit area of the Anaheim Convention Center.

NASA TechBriefs, the event sponsor, selected the virtual wind tunnel, which consists of a prototype system of special-purpose software and hardware that applies virtual reality interface techniques to the visualization of simulated fluid flows. Its main advantages are 3D display and interaction to explore complex flows intuitively, allowing researchers to examine the results of their simulations.

"The virtual wind tunnel is a good demo because a user can actually experience something," said Steve

Bryson, a member of the NAS Applied Research Branch and project team lead, noting that during the three-day expo there was nearly always a waiting line for the demonstration. A video was also distributed at the booth.

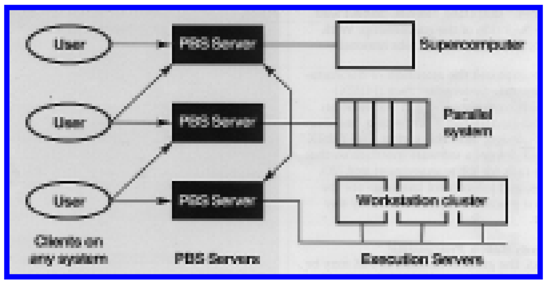
At Technology 2003, companies interested in adapting or using the virtual wind tunnel for their own applications included Ford Motor Co., Rolls Royce Aircraft Engine Division, and Harley-Davidson, as well as the Air Force-Arnold Engineering Development Center at Arnold AFB, TN. Bryson had previously assisted the National Center for Supercomputing, Urbana, IL, and the Army Corps of Engineers' Waterways Experimental Station, Vicksburg, MS, with installing systems similar to the virtual wind tunnel. Other industry representatives have expressed long-term interest in the virtual wind tunnel technology, but Bryson thinks that the current state of the economy and a significant upfront investment discourage other development, at present.

After extensive user testing of the virtual wind tunnel, to be conducted through mid-1994, NAS will continue to impart the results of its research to others. All hardware components of the virtual wind tunnel are commercially available, and the software source code will be provided to development sites on an "as is" basis. Developers at other sites must write and monitor application code to meet their own specific requirements, Bryson said. "The demands of a virtual reality application are so specific to a task that small changes in task specification could require radical changes in software," he explained.

Sandy Johan, a member of the virtual wind tunnel project development team, along with Sam Uselton, assisted with the demo at the Ames booth.

Next Article Contents Main Menu NAS Home

NAS, LLNL Release PBS



by Elisabeth Wechsler

NAS and Lawrence Livermore National Laboratory (LLNL) released the first phase of Portable Batch System (PBS) on March 1. The result of a two-year collaborative software development effort, PBS provides job dependency and job synchronization features, as well as improved batch job scheduling, for massively parallel computers, clustered workstations, and "traditional" supercomputers.

The NAS-LLNL development team expects PBS to eventually replace Network Queuing System (NQS) because NQS "doesn't deal well with the demands of massively parallel machines," said Dave Tweten, project lead for the NAS side of the partnership. With PBS, it's also easier to run jobs remotely.

PBS has captured the attention of the Portable Operating System Interface (POSIX) standards committee, which will issue an IEEE standard for batch processing, based on the PBS design, later this spring. The POSIX standard defines a software interface so that system calls for job management and I/O, utilities, and job control language are the same for machines manufactured by any vendor.

Expands Batch Processing

For NAS, the greatest benefits of PBS may be to significantly expand batch processing for its three massively parallel machinesÑThinking Machine's CM-5, and Intel's iPSC/860 and ParagonÑand also to act as a "lightweight client" by giving commands to run batch jobs from workstations. With PBS, a server has one job scheduler and a resource monitor for each workstation to control batch processing on all the workstations in the cluster. Unlike NQS, the job scheduler and resource monitor are "invisible" to the user, Tweten said. "The user can launch a job at the server and the server runs it on whatever machine is appropriate."

"With NQS, DQS [Distributed Queuing System], and DJM [Distributed Job Manager], you can't move jobs across these systems because they're totally incompatible," said Bob Henderson, also a member of the NAS team. PBS allows the migration of jobs to the least-loaded machine that satisfies the resource requirements, he explained.

Jobs Held for 'Meaningful Interaction'

PBS emphasizes operational flexibility with simplified system software and increased usability through extensive man-page documentation. Job coordination is improved because jobs running on different machines are now held by PBS until meaningful interaction is likely.

The job dependency feature of PBS allows users to submit a series of jobs, and the execution of each job is dependent on the success or failure of the previous job in the series. PBS also breaks larger jobs into a sequence of smaller jobs, Henderson said.

The job synchronization feature allows PBS to run two or more jobs with parallel processes that need to share data and communicate over the network simultaneouslyÑand both jobs can be started at the same time.

Each side of the NAS-LLNL partnership contributed to the design and development of PBS. NAS developed the network protocols, the client-server application, the resource monitor, and the machine-specific, job-launching software. LLNL was responsible for the user utility interface, the applications programming interface library, and the job scheduler. The NAS-LLNL team meets twice a month, alternating between the two Bay Area sites, Moffett Field and Livermore.

A More Risk-taking Culture

"LLNL has a more software risk taking culture than NAS," Tweten commented. "LLNL is very interested in running PBS on their [CRAY Y-MP] C90." This receptivity to use experimental software in the research environment was a strong inducement to forming the partnership, he said. When Tweten asked his counterparts at LLNL about their software testing policy, he was told: "When the programmer feels the software is ready, we put it into production." Tweten noted that NAS is "reluctant" to take this level of risk.

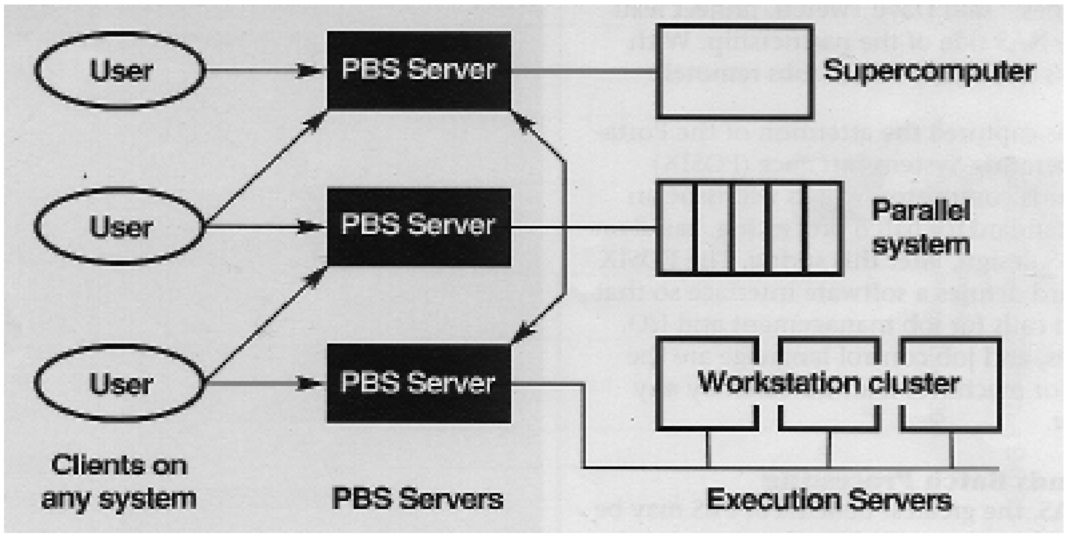
In terms of personnel, NAS has provided Tweten, Henderson, and Tom Proett (all of the NAS Systems Development Branch), to work on software design; and John Musch (NAS Computational Services Branch), to manage the testing program for developers and NAS users, scheduled to run through the summer. LLNL contributed two full-time employees (spread among several part-timers) from its Central Computing Facility and National Energy Research Supercomputing Center.

Release Schedule

The first release of PBS supports the Cray, Paragon, and CM-5 machines, as well as Sun and SGI workstations. A second release, due in May, will provide the NQS gateway for passing jobs automatically to PBS -- a major challenge for the project team, according to Tweten. The third release, due in August, will support workstation clusters.

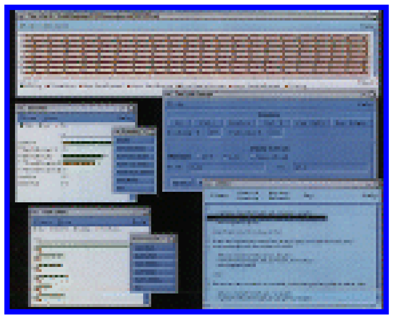
As a result of the NAS LLNL collaboration, each institutional partner will "own" a copy of PBS, which they will distribute to their respective user communities. For more information, send email to tweten@nas.nasa.gov.

Next Article | Contents | Main Menu | NAS Home



The Portable Batch System will provide common batch services on and between all classes of systems within the NAS environment: supercomputers, parallel systems, and groups of workstations (clusters). In contrast, NQS provided service on a single execution system, typically a supercomputer; other batch systems were required for parallel or clustered systems.

Back to the article.



ntv Displays Parallel Data

by *Louis Lopez*

A new trace visualization tool -- the NAS Trace Visualizer, or ntv -- has been developed to help users identify inefficient portions of code written for message-passing parallel systems. Developed in the NAS Research and Development Branch, ntv uses static displays to trace important events that occur within parallel code. The tool has recently been released at NAS on the Support Processing Systems (SPSs), amelia, fred, orville, and wilbur.

ntv evolved in response to informal "wish lists" from programmers faced with the difficult and time-consuming task of debugging and tuning code to produce programs that run quickly and give correct results. These difficulties are even greater today with the advent of message-passing massively parallel computers (MPPs) because problems that arise from asynchronous processes are added to the problems of "traditional" computing.

Initial feedback on ntv, which was featured in demonstrations at Supercomputing '93 in November and at the NAS User Interface Group meeting in January (see *UIG Meets at NAS*), have been positive.

Scalar & Vector vs Parallel Processing

In scalar and vector processing, when the same data is used in a series of program runs, the same sequence of instructions is always executed for each run. If the program stops at some point, the state of the program is always the same -- which means that errors are reproducible. This is not true for parallel programs, where errors may appear in some runs but not in others. This can occur, for example, because messages exchanged by two or more processors arrive in a different order for different runs. Techniques and tools for debugging programs on sequential and vector processors have evolved over the years, but since MPPs are relatively new -- and experience in their use is not widespread -- debugging and tuning techniques are not well understood. So far, the approach has consisted primarily of extending existing techniques to parallel systems, but these extensions are far from clear due in part to the asynchronous program execution and the large amount of data to be examined.

Execution Tracing

ntv falls within the category of "execution tracing" tools, in which a trace file containing a record of all "important" events that occurred during program execution is analyzed. Examples include sending and receiving messages or entering blocked states.

There are two basic ways to produce trace files: one uses an instrumentation library -- perhaps with an instrumenting program -- to place calls to traceemitting subroutines in the program. The other uses facilities built into the compilers and their libraries. Regardless of the method used, trace files can be very large, and are, consequently, difficult to interpret without visualization tools. The current version of ntv uses trace files produced using the Automated Instrumentation and Monitoring System (AIMS) Version 2.2.

Problems With Scrolling Displays

Trace visualization programs differ in many details, but the most common visualizers (for example, ParaGraph, developed at Oakridge National Laboratory, and Pablo, from the University of Illinois) use animated or scrolling displays. The displays give the appearance of watching the time-based evolution of computation.

There are several problems with this method: In a normal computation, trace events are not uniformly distributed in time. If the time is scaled to spread out the closely spaced events so that the programmer can view them, there are long periods with nothing happening on the display. Further, the execution trace takes much longer to display than the actual computation. If, on the other hand, the execution time is scaled to shrink the periods with no trace events, then the closely spaced events are lost.

Static Displays Make the Difference

ntv uses a series of static displays -- that is, the displays do not vary with time. Static displays are used for three main reasons. First, understanding the events in parallel programs usually requires a considerable amount of "staring and head scratching," which is easier to do with displays that don't change as users look at them. Second, some of the capabilities that are useful in analyzing code, such as zooming in on areas of interest, are difficult to implement in nonstatic displays. Third, static and animated displays, when used in conjunction, can augment each other.

The principal ntv display is a time-line showing a series of horizontal lines -- one for each processor (top image in figure). These lines are color-coded to indicate the processor status (such as running or blocked). The horizontal axis represents real time, in seconds. The time-line display gives users an easy way to study the status of processors at any point in time. The initial view shows all data in the trace from beginning to end. A mechanism allows users to select any region in the display for zooming in or out and panning the display (center right in figure).

Message Display Control

For message-passing systems, message traffic is particularly important. In ntv, messages passed between two processors are indicated by thin lines connecting the sending processor's time-line to that on the receiving processor's. ntv gives users control over which messages are displayed, in order to keep the display from being overwhelmed with message lines. For example, a user can choose to view only those messages sent from processor 2 to processor 3, or only those messages sent to processor 4.

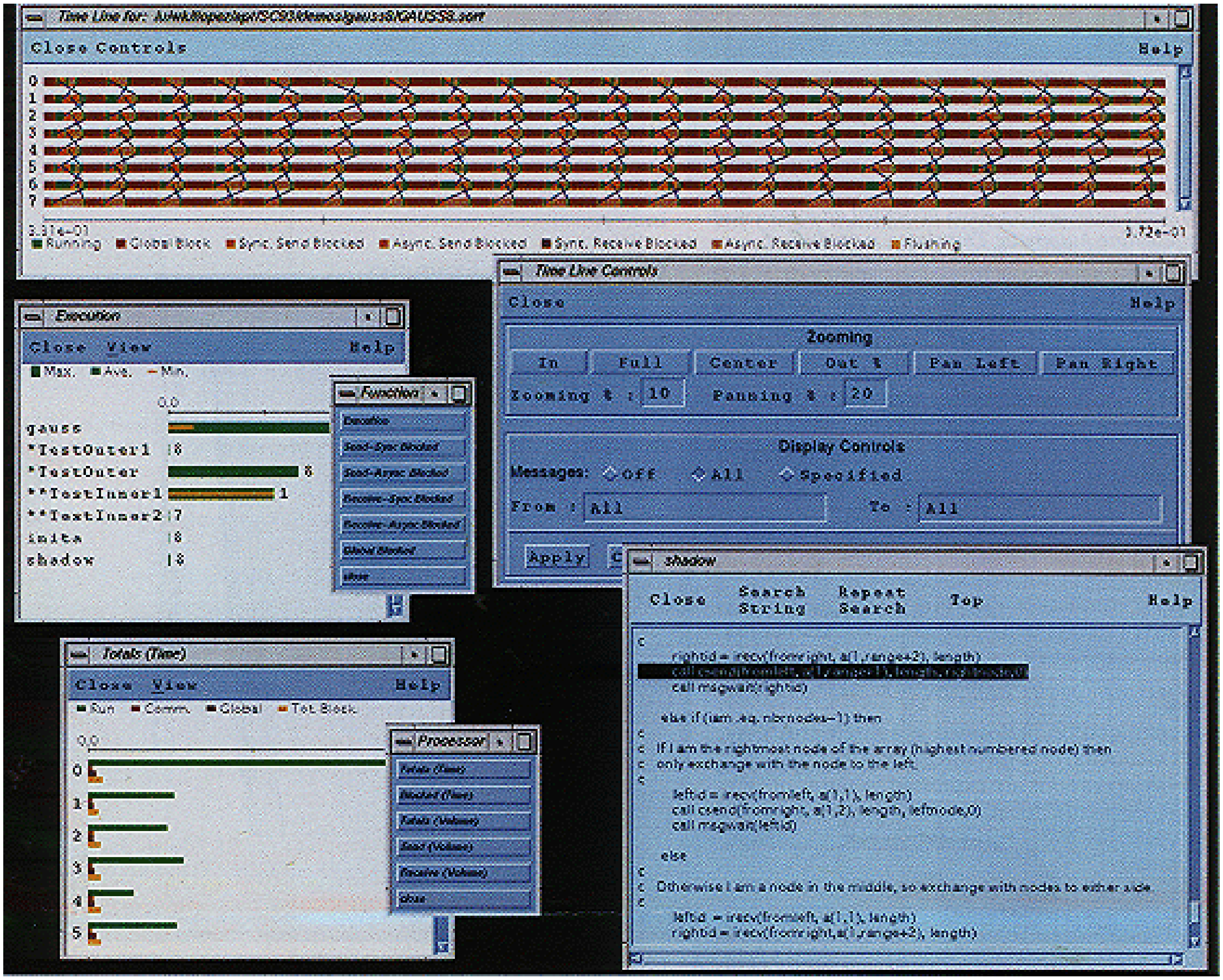
To debug and tune a program, it is necessary to relate the observed status and behavior of the program execution to the code that caused the behavior. With ntv, users can open a browser (bottom right in figure) that shows the line of code causing the behavior observed in the display and then use the mouse to point and click on areas of interest.

While the time-line display gives users detailed information about program execution, profiling (or performance summary) data is often the starting point in tuning a program. ntv uses the trace data to produce two sets of displays. One set (bottom left of figure) gives information for each processor and the other (center left in figure) gives information for each function in the program.

ntv is available to NAS users on the SPSs for use with the Intel iPSC/860. For more information, send email to: **llopez@nas.nasa.gov**.

Louis Lopez, who developed ntv, is a member of the Computer Sciences Corp. team working in the NAS Parallel Tools group.

Next Article | Contents | Main Menu | NAS Home



```
c
      rightid = irecv(fromright, a(1,range+2), length)
      call csend(fromleft, a(1,range+1), length, rightnode,0)
      call msgwait(rightid)

    else if (iam .eq. nbrnodes-1) then
c
c   If I am the rightmost node of the array (highest numbered node) then
c   only exchange with the node to the left.
c
      leftid = irecv(fromleft, a(1,1), length)
      call csend(fromright, a(1,2), length, leftnode,0)
      call msgwait(leftid)

    else
c
c   Otherwise I am a node in the middle, so exchange with nodes to either side
c
      leftid  = irecv(fromleft, a(1,1), length)
      rightid = irecv(fromright,a(1,range+2), length)
```

The ntv displays, clockwise from top: time-line shows processor status; zooming and display control give expanded view of displays and allow message traffic control, respectively; a browser identifies specific lines of code. The two remaining displays give profiling information for each processor and for each program function.

Back to the article.

Jury Still Out on MPI

by Bill Saphir

Applications written for distributed memory parallel computers at NAS generally fall into one of two classes: data parallel programs and messagepassing programs. The accepted standard for data parallel programs is High Performance Fortran (HPF). Until recently, no such standard existed for message-passing programs. In November 1993, the Message Passing Interface Forum unveiled a draft proposal for a standard message passing library -- MPI.

The standardization process began in April 1992 with the "Workshop on Standards for Message Passing in a Distributed Memory Environment," in Williamsburg, VA. Out of this grew the MPI Forum, whose members include representatives of industry, academia, and government research facilities, including NAS, which was represented by several rotating members.

Among supercomputer and minisupercomputer vendors, MPI has at least some support from Cray Research, Convex, Intel, IBM, Thinking Machines (TMC), Meiko, and nCUBE. Of these, IBM and Meiko seem the most committed to providing a fast proprietary version of MPI for their platforms. The degree of commitment from other vendors is hard to gauge.

The MPI Forum, modeled on the HPF Forum, had the goal of achieving for the message-passing world what HPF has achieved for high performance Fortran -- a practical, portable, and efficient standard. The jury is still out on whether this goal has been achieved.

Portability and Performance

In a message-passing program, parallelism is explicit; that is, at least one independent thread of control runs on each node of a multiprocessor. Data distribution is also explicit; a node has direct access only to its local data. What distinguishes message passing from similar paradigms is that nodes exchange data using a cooperative send-and-receive mechanism.

Routines to send and receive data are provided by a communication library. Despite the simplicity of the send-and-receive model, communication libraries differ in almost every detail. The most obvious differences are in the support they provide for global operations, parallel I/O, and other operations that go beyond simple send/receive exchanges. Just as important, the send-and-receive routines themselves behave differently in different libraries: for example, they may or may not buffer data, may block under different circumstances, and may or may not allow asynchronous communication. These differences inhibit portability and make it difficult to obtain high performance for ported codes without substantial rewriting.

Provides Comprehensive Support

MPI is a comprehensive standard -- some argue that it's too comprehensive. Everything from other message-passing libraries, with the exception of interrupt-handlers and process-creation infrastructure, is contained in MPI -- from global reductions, which exist in most message-passing libraries, to contexts and user-defined data types, which exist in very few libraries. Every concept in MPI is based on a feature that already exists, in some form, in a current (though possibly obscure) message passing library.

Supports Groups and Context

One of the most useful but least familiar features of MPI is support for groups of processes and context. A group is a collection of processes in a parallel application. Global operations (such as scans, reductions, and barriers) and more general parallel library routines are most naturally defined over groups. Groups also provide a convenient framework for multidisciplinary applications.

Context is a mechanism that allows a programmer to isolate parts of a large program so that messages from one part can't interfere with messages in another. Context is essential for writing correct and reusable libraries and helps considerably in the development of large and complex programs. MPI combines the concepts of group and context into an object called a "communicator. "

The concepts of context and group may be new to users familiar with NX, CMMD, or PVM. Both NX and CMMD provide support for a single group, and only enough context support to provide safe global operations on a single group. The Intercube library on the Intel iPSC/860 and the Map library on the Intel iPSC/860 and the Map library on the Paragon extend NX by adding enough group support to handle multidisciplinary applications -- but this support is not nearly as robust as that in MPI. At the other extreme, PVM provides virtually no support for groups or context that is useful for computational fluid dynamics codes. The robust support of groups and context in MPI may spur the development of sorely needed parallel libraries.

Who Should Use MPI?

Currently, no production version of MPI is available for NAS systems. Until a version exists that can compete, in terms of performance, with NX or CMMD on their native platforms, there is little reason for NAS users to port existing codes. If portability is a primary concern or if users work on loosely coupled computer networks, PVM is currently the only option available. MPI will soon provide an attractive alternative.

Although PVM is often perceived by users as an emerging "standard" for message passing, it isn't. PVM is an ongoing research project, and there is no defining document other than the current version of the

PVM User's Guide. While lack of an official standard could be dismissed as unimportant, it has already led to an implementation of "PVM" on the CRAY T3D that differs greatly from the publicly available version. Moreover, PVM was designed for networks of workstations, not supercomputers, and has some serious performance disadvantages when run on multiprocessors -- which is why the CRAY T3D extensions were created.

Disadvantages of MPI

While MPI is a significant step forward, there are several problems that might prevent it from being accepted as a standard. Most important, of course, is that no productionquality public version is available, so it remains to be proven whether MPI can live up to its promise.

Objections have been raised to several technical aspects of the MPI standard. Among the most important are concerns about the portability and performance of its point-to-point communication. MPI allows several communication "modes" for point-to-point messages. These modes have different semantics and are not interchangeable. For any given architecture, there is an MPI mode that can achieve excellent performance, but the converse is not true -- none of the MPI modes can be made to perform well on all architectures. Moreover, the behavior of the standard mode is not completely specified (the size of the buffers it uses is not specified by the standard), so that writing programs that are both portable and efficient may be difficult.

Current Status and Availability

A freely available and portable version of MPI is under development by three members of the MPI Forum. This version is expected to run on most machines in use at NAS, including the Intel Paragon and the TMC CM-5. An early version has been available since December and is updated continually. When a fairly stable version is released, probably sometime this spring, it will be installed at NAS.

A copy of the draft specification is available through anonymous ftp at **netlib.ornl.gov**. For more information about MPI and other message passing libraries, a videotape of the seminar, *Comparison of Message Passing Libraries: NX, CMMD, MPI, PVM*, is available from the NAS Documentation Center. Send email to **doc-center@nas.nasa.gov** . For more information on MPI, send email to **wcs@nas.nasa.gov** .

Bill Saphir is a member of the Computer Sciences Corp. team working in the NAS Parallel Systems Science Support group. He earned his Ph. D. in Physics from the University of Texas in 1992 and joined the NAS program in 1993.

Next Article | Contents | Main Menu | NAS Home

CMAX Fortran Translation Useful? It Depends

by Chuck Niggley and Ed Hook

The November-December 1993 issue of *NAS News* presented an overview of CMAX, The Connection Machine Automatic Translator, a product of Thinking Machines Corp. (TMC).

This article discusses some specific actions that CMAX takes (or attempts to take) in translating Fortran 77 programs into CM Fortran and gives examples of how CMAX handles actual code. Some examples show CMAX generating inefficient code and others show users how to help CMAX generate much more efficient code.

CMAX vectorizes DO loops -- that is, translates them into CM Fortran array operations, where possible. It also selects the appropriate "home" (the serial control processor or parallel processing nodes) for each array. It views the program as a whole and then provides the cmf compiler with code (including directives) that expresses parallelism, avoids "array home" conflicts by directing that arrays processed in parallel be allocated in distributed memory, and uses correct CM Fortran methods of passing distributed arrays as arguments.

Arrays and DO Loops

An array operation is a computation performed as a single entity -- that is, on all the array's elements, or a specified subset of those elements. In Fortran 77, array operations are expressed in terms of DO loops, which specify array elements in a particular sequential order and execute one (or more) entire statement(s) for an element before going on to the next element.

By contrast, in CM Fortran, an array operation may process elements in any order, but it must perform any given single operation (such as addition, multiplication, or assignment) for all elements before performing the next operation. So, an array operation can process array elements in any order without changing the result.

The CMAX Converter analyzes Fortran 77 DO constructs to determine if they are functionally equivalent to any CM Fortran array operation. If they are, CMAX replaces the loop(s) with a corresponding array operation.

This is a straightforward process for loops without dependencies. CMAX recognizes the intent of certain loops that embody dependencies as being equivalent to CM Fortran intrinsics or Utility Library procedures, and translate these "common idioms." For example, a loop with embedded IF statements might be replaced by a masked array assignment expressed as a WHERE block, a FORALL statement, or possibly a masked intrinsic function.

Vectorization is Simplified

CMAX is reasonably smart and will restructure code to facilitate vectorization. Scalar values that are used with arrays are promoted into arrays of the appropriate shape. In turn, these arrays are aligned with other CM arrays, if necessary, to enhance the code's locality of access.

The converter splits loops to separate vectorized code from inherently serial operations. If the resulting loop would vectorize, CMAX may also transform a loop with an embedded subroutine call into a subroutine that contains the loop.

If CMAX is unable to translate a given DO loop, the code passes through unchanged to the CM Fortran compiler and is executed serially.

Translations fail for several reasons: some loops containing RETURN statements or other "unstructured" constructs are inherently serial. In other cases, CMAX is unable to determine whether an apparent data dependency represents an actual problem. Other loops express common idioms that do not currently have a parallel implementation .

How CMAX Handles Array Homes

CMAX frees the user from worrying about the problem of array homes by automatically inserting a LAYOUT directive for every array.

Unlike the compiler, the converter does perform interprocedural analysis. It can determine whether an array is used in an array operation anywhere in the program, avoiding mismatched homes across procedure boundaries.

The converter may also clone certain subprograms, allowing one version to accept front-end arrays while the other takes CM arrays as arguments.

Users can override CMAX by using in-line directives or converter command-line options. These features, and the situations where they might be useful, are described in Chapter 2 of the TMC manual, Using the CMAX Converter.

CM Fortran adopts Fortran 90 semantics for passing CM array arguments and retains Fortran 77 semantics for front-end array arguments.

Unlike Fortran 77, which passes arrays by reference, CM Fortran passes arrays by descriptor. The reference to the name of a CM array indicates all of its elements.

Observations and & Examples

Overall, CMAX provides many benefits to users. However, don't assume that this tool can run unsupervised. Some examples taken from real codes, where simple changes to CMAX's translation greatly improved code performance, are shown below.

CMAX does not always move computations off the partition manager in the most efficient way. For example, consider the following piece of Fortran 77 code:

```
     do 20 m=1,5
     do 20 k=1,nk
     do 20 j=1,nj
       da2 = 0.5*dy(j)*dz(k)

     do 15 i=2,n1
       scl(i,j,k) = da2*(sc2(und(i),j,k,m) + sc2(i,j,k=m))
15   continue

     do 20 i=2,nim
       dg(i,j,k,m) = dq(i,j,k,m) + scl(i,j,k) - scl(ipl(i),j,k)
20   continue
```

CMAX transforms this code into the following:

```
     FORALL (m = 1:5, k = 1:nk, j = 1:nj) da2100(m,k,j) = 0.5 * dy(j)
    &    * dz(k)
     DO m = 1 , 5
       FORALL (k = 1:nk, j = 1:nj, i = 2:ni) sc1(i,j,k) = da2100(m,k,j)
    &      * (sc2(im1(i),j,k,m) + sc2(i,j,k,m))
       FORALL (k = 1:nk, j = 1:nj, i = 2:nim) dq(i,j,k,m) = dq(i,j,k,m)
    &      +sc1(i,j,k) - sc1(ip1(i),j,k)
     ENDDO
```

Here, CMAX introduced the array **`da2100`** to hold the various values that **`da2`** takes on and computed them in parallel in the first FORALL statement. It also translated all of the inner loops into a pair of FORALL statements, but left the outer loop on **`m`** untranslated. This means that the computations on the nodes represented by those FORALL statements had to be interrupted while the partition manager handled the loop overhead. Once aware of this "trouble spot," the goal is to eliminate the last DO loop. The following replacement code was devised.

```
      FORALL(i=1:ni,j=1:nj,k=1:nk,m=1:5) eds(i,j,k,m)=0.5*dy(j)*dz(k)
      FORALL(i=2:ni,j=1:nj,k=1:nk,m=1:5)
     &   eds(i,j,k,m)=eds(i,j,k,m)*(sc2(im1(i),j,k,m)+sc2(i,j,k,m))
      FORALL(i=2:nim,j=1:nj,k=1:nk,m=1:5)
     &   dq(i,j,k,m)=dq(i,j,k,m)+eds(i,j,k,m)-eds(ip1(i),j,k,m)
```

At the cost of slightly more temporary storage (replacing the three-dimensional **`da2100`** by the four-dimensional array **`eds`**), the final DO loop is eliminated from the code. Additionally, all of the arrays involved no have the same geometry, which reduces the internodal communication required to execute thise statements. (Some communications overhead still exists because of the indirection encoded in the **`iml`** and **`im1`** arrays, but the difference is still notable.)

Remember, CMAX is a translator, *not* an optimizer. CMAX will sometimes produce code thranslations that programmers should recognize as inefficient. For example, the following code:

```
      do 10 k=1,nk
      do 10 j=1,nj
      do 10 i=1,ni
        rhoinv = 1.0/q(i,j,k,1)
        sc(i,j,k,1) = q(i,j,k,5)
     &              - 0.5*(  q(i,j,k,2)**2
     &                     + q(i,j,k,3)**2
     &                     + q(i,j,k,4)**2
     &                    )*rhoinv
10    continue

      do 20 k=1,nk
      do 20 j=1,nj
      do 20 i=1,ni
        rhoinv = 1.0/q(i,j,k,1)
        sc(i,j,k,2) = q(i,j,k,2)
     &              - 0.5*(  q(i,j,k,5)**2
     &                     + q(i,j,k,4)**2
     &                     + q(i,j,k,3)**2
     &                    )*rhoinv
20    continue
```

results in the following CMAX translation:

```
      rhoinv100(1:ni,1:nj,1:nk) = 1.0 / q(:,:,:,1)
      sc(:,:,:,1) = q(:,:,:,5) - 0.5 * (q(:,:,:,2) ** 2 + q(:,:,:,3)
     &    ** 2 + q(:,:,:,4) ** 2) * rhoinv100(1:ni,1:nj,1:nk)
```

```
      rhoinv101(1:ni,1:nj,1:nk) = 1.0 / q(:,:,:,1)
      sc(:,:,:,2) = q(:,:,:,2) - 0.5 * (q(:,:,:,5) ** 2 + q(:,:,:,4)
     &    ** 2 + q(:,:,:,3) ** 2) * rhoinv101(1:ni,1:nj,1:nk)
```

Note that, at their point of use, the arrays**rhoinvlOO** and **rhoinvlOl** are element wise identical. Since division is the most "expensive" of the elementary floating-point operations, this example shows that CMAX does not perform the sort of "dataflow analysis" that might have shown the possibility of reusing **rhoinvlOO**. Since the actual code from which this example was distilled had another occurrence of this same idiom(**rhoinvlO2**), making the obvious modifications resulted in a significant decrease in execution time.

The previous example also highlights another CMAX issue. Note that the calculation of each element of the **sc** array involves five elements of the **q** array, which differ only in their 4th coordinate. If **q** is laid out with all of its axes declared **:NEWS**, then these five elements probably reside on different nodes -- which means that these calculations require substantial internodal traffic. In this case, an examination of the whole code indicated that most uses of the **q** array followed this pattern. The CMAX-produced LAYOUT directives were altered to declare the rightmost axis of **q** as **:SERIAL**. This forced all elements of **q** that differed only in their last coordinate to reside on the same node. The modification resulted in computations that require no internodal communication.

Decreasing internodal communication is generally the most effective optimization strategy, so it pays to look for these opportunities to refine CMAX translations.

A final example, shown below, reinforces the point of the previous example.

```
      ex5 = 0.0

      do 10 k=1,nk
      do 10 j=1,nj
      do 10 i=1,ni
        ex5 = amax1(ex5,sqrt(u(i,j,k)**2 + v(i,j,k)**2 + w(i,j,k)**2))
10    continue
```

CMAX changes this code to the following:

```
      ex5 = 0.0

      DO 10 k = 1 , nk
        DO 10 j = 1 , nj
          DO 10 i = 1 , ni
            ex5 = amax1(ex5,sqrt(u(i,j,k) ** 2 + v(i,j,k) ** 2 + w(i,j,k
     &          ) ** 2))
```

```
10   CONTINUE
```

The CMAX translation merely reformatted the original code -- and, in this case, made it less readable. Since all of the arrays involved are CM arrays, this translation is far worse than the previous example, as each innerloop iteration involves a substantial amount of computation that has to be done on the partition manager while the nodes "cool their heels." A better translation is shown below:

```
     u100(:ni,:nj,:nk) = u100(:ni,:nj,:nk)*u100(:ni,:nj,:nk)
     v100(:ni,:nj,:nk) = v100(:ni,:nj,:nk)*v100(:ni,:nj,:nk)
     w100(:ni,:nj,:nk) = w100(:ni,:nj,:nk)*w100(:ni,:nj,:nk)
     u100(:ni,:nj,:nk) = u100(:ni,:nj,:nk) + v100(:ni,:nj,:nk)
     u100(:ni,:nj,:nk) = u100(:ni,:nj,:nk) + w100(:ni,:nj,:nk)
     v100(:ni,:nj,:nk) = sqrt(u100(:ni,:nj,:nk))
     ex5 = amax1(0.0,maxval(v100(:ni,:nj,:nk)))
```

where '**`u100`**', '**`v100`**', and '**`w100`**' are temporary arrays.

Alternately, users can reach this same point by rewriting the original code as follows:

```
     ex5 = 0.0

     do 10 k=1,nk
     do 10 j=1,nj
     do 10 i=1,ni
       d(i,j,k) = sqrt(u(i,j,k)**2 + v(i,j,k)**2 + w(i,j,k)**2)
10   continue

     do 20 k=1,nk
     do 20 j=1,nj
     do 20 i=1,ni
       ex5 = amax1(ex5,d(i,j,k))
20   continue
```

By adding an "automatic" array **`d`** (and specifying **`-PermitAuto`** on the CMAX command-line), CMAX produces the following translation:

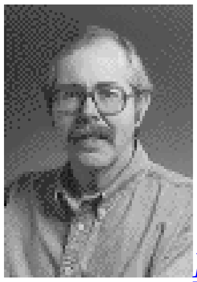
```
     ex5 = 0.0

     d = sqrt(u ** 2 + v ** 2 + w ** 2)

     ex5 = amax1(ex5,maxval(d))
```

Getting the best results from CMAX requires effort. If CMAX encounters an operation that it can't process and leaves a "wart" in the code as a result, the most effective approach may be to recast the original code and break it down into pieces that CMAX can digest.

CMAX does a good job of automating much of the drudgery involved in translating Fortran 77 code to CM Fortran, and it handles much of the straightforward recasting of DO loops into array constructs. With some extra effort by users, the results of CMAX translations can be significantly improved.


Chuck Niggley and Ed Hook are members of the Computer Sciences Corp. team working in the NAS Parallel Systems Science Support group.

Next Article | Contents | Main Menu | NAS Home

Many Challenges Ahead for MPP

by Elisabeth Wechsler

"There is almost no real MPP computing being done today," observed Horst Simon, of the NAS Applied Research Branch. Simon addressed an audience of almost 100 NAS and Ames staff, as well as representatives from Hewlett-Packard, Stanford University, and other local organizations, in which the impact of massively parallel processing (MPP) was a major focus.

The reason, he explained, is that fewer than 10 machines world wide have more than 1,024 processors -- the minimum needed to be truly massively parallel, in Simon's opinion -- and provide peak performance in excess of 5 GFLOPS. The December 21 talk was an informal synthesis of conversations and work done by Simon and other NAS colleagues over the last six years.

Predicted Less Parallelism in Future

Simon predicted that "there will be even less parallelism in the near future," adding that powerful microprocessors are the optimal design point." Since microprocessors are becoming increasingly more powerful, he explained, the same level of performance can be obtained with less parallelism.

Another development in parallel computing, Simon noted, concerns the disappointing results achieved over the last three years by single instruction multiple data (SIMD) machines for general-purpose scientific computing applications. Simon descnbed SIMD machines as "tightly synchronized massively parallel machines built out of many 'powerless' processors." The reason, he said, is that "performance from problem to problem on the same machine is highly variable, and random communications have not always been well supported."

Simon said that "killer micros" were the enabling technology for the rapid growth of highly massively parallel machines in the late 1980s, and added, "Even though these superpowerful microprocessors made a big jump in price performance, it would be wrong to extrapolate this exponential growth in the future."

Lack Sufficient Memory Bandwidth

"The real dramatic growth in performance is over for micros. They also lack sufficient memory bandwidth for CFD [computational fluid dynamics]-type calculations," he said.

In the foreseeable future, Simon predicted a continued "performance gap" between high-performance custom processors and high-performance commodity microprocessors.

"We'll see both versions, but commodity microcomputers won't replace high-performance custom processors," Simon said, adding that a parallel machine containing 64 custom processors running at 2 GFLOPS and one containing 512 microprocessors nunning at 256 MFLOPS will provide equivalent computing power in the near future. He noted that it takes eight times the power of one commodity processor to achieve the equivalent results of one custom processor.

Simon also emphasized the importance of internode bandwidth. "High-bandwidth interconnects remain the major challenge for MPP systems," he said. "We continue to focus on delivering teraflopsper-second processor performance but pay less attention to the equally important terabytes-persecond bandwidth requirement for the interconnect."

Software Makes the Difference

Addressing the recently emerging interest in distributed computing, Simon said that networked workstations provide essentially the same technology as current MPP systems -- however, without the system software. So, he said, "it's highly doubtful that this approach will be an alternative to MPP systems."

"MPP systems provide a unified system image for the same type of hardware. If independent software developers or MPP vendors provide system software, then networked workstations will be the winner," Simon predicted. Furthermore, he said, clustered workstations can provide highly effficient throughput for small- and medium range production jobs.

New Applications Taxonomy

Simon also discussed a new way of classifying applications according to their communications requirements. He pointed out that applications' performance on parallel machines is highly variable, depending on whether the application is structured or unstructured, requires explicit or implicit communication, or involves data that is statically allocated to processors or where the allocation is changing dynamically.

"Most of the challenging multidisciplinary applications have -- unfortunately -- unstructured, implicit and dynamic characteristics, Simon said. "Consequently they are difficult to implement on parallel machines."

Profitability vs Credibility

Another concern, Simon noted, is that high performance computing is increasingly dominated by politics and short-term business results .

The installed base of so-called "big science" -- those 30 sites with the top parallel machines -- totals $500 million, he said. "Big science gets 100 percent of the media attention and 100 percent of technological improvement and development money," Simon continued. In contrast, there are about 1,000 installations of production engineenng machines totaling S4 billion -- a much bigger market.

The vendors' dilemma, he explained, is that although competing in the big science market is not profitable, it's important for building credibility.

Simon urged continued strong financial and political support for the federal High Performance Computing and Communications Program (HPCCP). He reminded the audience of the "painful lesson" learned in the 1980s, when American semiconductor manufacturers were at a disadvantage in the world market relative to foreign chip manufacturers, who received government support, tax benefits, and tariff protection.

Payoffs Will Come -- Eventuality

"The benefits to taxpayers are not the same as with a consumer product -- we're not talking about VCRs," Simon said. "This is a major technology transition -- the benefits may not come until 2020. There is a long lead-time for investment in parallel computing, but eventually there will be major payoffs for the average consumer," he added.

For a videotape or detailed handouts of Simon's talk, *Six Years of Parallel Computing at NAS (19871993): What Have We Learned?*, send email to **doc-center@nas.nasa.gov** .

Next Article | Contents | Main Menu | NAS Home

UIG Meets at NAS

On January 14, representatives of the NAS User Interface Group (UIG) met for a day dedicated to better serving NAS users. Members of the UIG include representatives of industry, academia, federal research facilities, and other government agencies, as well as those who use the NAS Processor System Network (NSPN).

The 80 attendees, including NAS and local NASA Ames staff, met in sessions involving all participants and in small groups to discuss user service developments in mass storage, parallel systems/software, distributed computing, job scheduling/ turnaround, and graphics/visualization. In addition, participants were given informal demonstrations of FAST, ntv, and Mosaic (see sidebar).

Dave Cooper, NAS Division Chief, emphasized the importance of the UIG meeting to NAS. Cooper wanted "to make sure that every participant's questions, concerns, and comments were listened to and followed up during the year by NAS."

Cooper reviewed NAS accomplishments for 1993, including various system upgrades, and discussed FY94 plans. These include the installation of new tape drives in the StorageTek (STK) silos, which will double storage capacity; installation of the HPCCP CAS Testbed-l; and the release of FAST Version 2.0 to Computer Software Management and Information Center (COSMIC).

Among the actions taken by NAS as a direct result of suggestions made by participants at last year's UIG meeting:

- All major systems of the NPSN were upgraded during the last year.
- NAS hosted the Distributed Computing for Aerosciences Applications Workshop in October, with 180 attendees; copies of the workshop proceedings were distributed to UIG meeting participants.
- NAS developed a massively parallel processor (MPP) system software plan, which has been favorably received by NASA research centers and Department of Energy laboratories and is currently being reviewed by Advanced Research and Projects Agency (ARPA).
- NAS published Hierarchical Storage Management System Evaluations, by Tom Woodrow, which compares NAStore, UniTree, FileServe, and DMF capabilities and requirements. The report is available from the NAS Documentation Center or online.
- Improvements were added to the Flow Analysis Toolkit (FAST) program. FAST 2.0, to be

released soon, is the result.

Cooper asked the user audience to assess the impact of delaying the delivery of the HSP-4 by approximately one year, as a result of budget limitations. He added that NAS was considering all types of machines for HSP-4, including conventional vector supercomputers and MPP machines, as well as clusters of workstations. MPP machines are progressing slower than expected; however, "the problem isn't hardware, it's software," he said. He also stated that the HSP-3 "will probably stay on the floor at least until 2000."

Cooper also asked participants if late December is still the most convenient time of year to conduct the annual NAS facility shutdown. He solicited guidelines about how long NAS should store user job data, and ultimately, how to dispose of old data. Send suggestions about these topics by email to Pat Elson, User Interface Manager and meeting coordinator, at **pelson@nas.nasa.gov** .

For more information, copies of the Report of the NAS UIG Meeting: January 14, 1994 are available from the NAS Documentation Center. Send an email request to **doc-center@nas.nasa.gov** .

Next Article | Contents | Main Menu | NAS Home

Next Article | Contents | Main Menu | NAS Home

FAST, ntv, Mosaic Demo'd for UIG

Three walk-in demonstrations were conducted in the NAS Graphics Lab during the UIG meeting on January 14. The demonstrations included: FAST's new remote collaboration feature; the NAS Trace Visualizer (ntv) tool. and NAS online information in Mosaic, which allows users to click on highlighted text, buttons, and icons to find and retrieve text, graphics, sound, and animation from anywhere in the world.

Remote Collaboration in FAST

The demonstration of "remote collaboration" in FAST was conducted by John West, a member of the FAST development team, and Kevin McCabe, both of Sterling Software in the NAS Systems Development Branch. Remote collaboration takes advantage of the fact that FAST can be controlled by script commands, allowing users at other sites to simultaneously visualize data by sending commands between machines. Visualization control can be passed from one user to the other. For more informanon about obtaining the remote collaboration software for FAST, send email to: jwest@nas.nasa.gov. Send general questions to **fast@nas.nasa.gov**.

ntv Tool

The ntv tool. which helps display large quantities of parallel performance data in visually relevant form, was demonstrated by Louis Lopez, NAS Systems Development Branch (see "ntv Captures Parallel Data".).

NAS Online in Mosaic

The demonstration of Mosaic, a networked information tool used for access to the World Wide Web, was given by Jean Clucas (Sterling Software in the NAS Systems Development Branch). The demo featured excerpts from NAS Technical Summaries, including Tom Woodrow's report, *Hierarchical Storage Management System Evaluation* (see UIG Meets at NAS). Until recently, the summaries have been available in hard copy format only.

NAS Seeks TAVS Users

NAS is soliciting users for TAVS, the Time Accurate Visualization System, for calendar year 1994. The system provides dedicated resources -- including CPU, memory, disk space, software and consulting services -- for unsteady computational fluid dynamics solution visualization and analysis. It is ideal for users who expect to procuce a time-varying data set this year and want visualiztion and/or analysis support.

TAVS provides an environment where users can analyze most or all of their data sets in one place, with a generous allotment of fast disk allocated to one project at a time. The intent is to provide a system that is available when users need it.

Not only does TAVS have a robust development environment, it also has production visualization software, including PLOT3D, PLOT3X, and UFAT. These packages can be combined to perform several kinds of data analysis. Distributed PLOT3D can be used to fiew data sets that may be too large to view interactively from the Cray on on a workstation. NAS is also intereted in discussing other methods or tools preferred by users.

The current TAVS system consists of a Convex C3240 with 1 gigabyte (GB) of random-access memory and 100 GB of fast disk at 20 megabytes/second (MB/sec). The Convex is connected to the NAS CRAY Y-MP C90 (**vonneumann**) via a HiPPI UltraNet connection at a current rate of 10 MB/sec, soon to be replaced by direct TCP/IP over the HiPPI.

The NAS goal is to have two to four concurrent projects on the system at a given time; a maximum of four projects has been set in order to keep CPU and memory available for interactive performance. It is expeced that each user will have access to the system for one to tow months, with negotiation if more time is required.

NAS is accepting intiail proposals for TAVS usage through April 15, 1994, but will consider other proposals on an ongoing basis. To request access to the system, include the following information in your proposal: time period expected for data generations; expected solution size; grid size, whether the grid is moving or static and the number of time steps, and the expected analysis tools (PLOT3D, PLOT3X, UFAT or other).

Mar - Apr 94 -- Vol. 2, No. 2

Executive Editor: Marisa Chancellor

Editor: Jill Dunbar

Senior Writer: Elisabeth Wechsler

Contributing Writers: Ed Hook, Louis Lopez, Chuck Niggley, Bill Saphir, Tom Woodrow

Other Contributors: Dave Barkai, Steve Bryson, Jean Clucas, Dave Cooper, Diane Couto, Pat Elson, Sam Fineberg, Chris Gong, Dick Gunderson, Steve Harding, Bob Henderson, Sandy Johan, Tom Lasinski, Subhash Saini, Horst Simon, Dave Tweten, Sam Uselton

Editorial Board: Marisa Chancellor, Jill Dunbar, Chris Gong, Jim Ruppert, Pamela Walatka, Serge Polevitzky, Elisabeth Wechsler, Rita Williams

NAS NEWS

Volume 2, Number 2 March - April 1994

NAS, LLNL Release PBS

by Elisabeth Wechsler

NAS and Lawrence Livermore National Laboratory (LLNL) released the first phase of Portable Batch System (PBS) on March 1. The result of a two-year collaborative software development effort, PBS provides job dependency and job synchronization features, as well as improved batch job scheduling, for massively parallel computers, clustered workstations, and "traditional" supercomputers.

The NAS-LLNL development team expects PBS to eventually replace Network Queuing System (NQS) because NQS "doesn't deal well with the demands of massively parallel machines," said Dave Tweten, project lead for the NAS side of the partnership. With PBS, it's also easier to run jobs remotely.

PBS has captured the attention of the Portable Operating System Interface (POSIX) standards committee, which will issue an IEEE standard for batch processing, based on the PBS design, later this spring. The POSIX standard defines a software interface so that system calls for job management and I/O, utilities, and job control language are the same for machines manufactured by any vendor.

Expands Batch Processing
For NAS, the greatest benefit of PBS may be to significantly expand batch processing for its three massively parallel machines—Thinking Machine's CM-5, and Intel's iPSC/860 and Paragon—and also to act as a "lightweight client" by giving commands to run batch jobs from workstations.

With PBS, a server has one job scheduler and a resource monitor for each workstation to control batch processing on all the workstations in the cluster. Unlike NQS, the job scheduler and resource monitor are "invisible" to the user, Tweten said. "The user can launch a job at the server and the server runs it on whatever machine is appropriate."

Continued on page 2

ntv Displays Parallel Data

by Louis Lopez

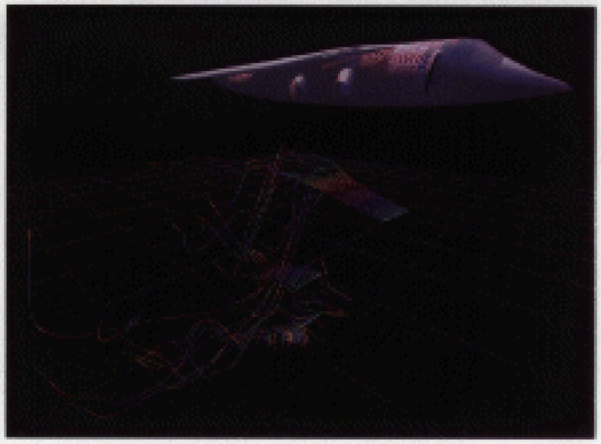
A new trace visualization tool—the NAS Trace Visualizer, or ntv—has been developed to help users identify inefficient portions of code written for message-passing parallel systems. Developed in the NAS Research and Development Branch, ntv uses static displays to trace important events that occur within parallel code. The tool has recently been released at NAS on the Support Processing Systems (SPSs), **amelia**, **fred**, **orville**, and **wilbur**.

ntv evolved in response to informal "wish lists" from programmers faced with the difficult and time-consuming task of debugging and tuning code to produce programs that run quickly and give correct results. These difficulties are even greater today with the advent of message-passing massively parallel computers (MPPs) because problems that arise from asynchronous processes are added to the problems of "traditional" computing.

Initial feedback on ntv, which was featured in demonstrations at Supercomputing '93 in November and at the NAS User Interface Group meeting in January (see article, *page 7*) have been positive.

Scalar & Vector vs Parallel Processing
In scalar and vector processing, when the same data is used in a series of program runs, the same sequence of instructions is always executed for each run. If the program stops at some point, the state of the program is always the same—which means that errors are reproducible. This is not true for parallel programs, where errors may appear in some runs but not in others. This can occur, for example, because messages exchanged by two or more processors arrive in a different order for different runs.

Continued on page 3



The virtual wind tunnel environment, showing the visualization of a data set of a Harrier aircraft in hover, with three rakes emitting streamlines of the velocity vector field. Vortices where the jets from the aircraft hit the ground are clearly visible. The computation was performed by Merritt Smith, Kalpana Chawla, and Bill Van Dalsem, all of the NASA Ames Fluid Dynamics Division.

Virtual Wind Tunnel Draws Crowds at Technology Transfer Expo

by Elisabeth Wechsler

A color demonstration of the NAS virtual wind tunnel was one of the most well attended exhibits at the fourth annual "Technology 2003" conference and expo, held December 7-9 in Anaheim, CA. The conference was attended by some 8,300 representatives of private industry, federal research labs, universities, and the general public.

Technology 2003 showcases commercially viable, cutting-edge technologies, developed by the nation's federal labs, which have potential application for private industry, according to Laura Welkie, Ames Office of Commercial Technology. Hundreds of exhibitors demonstrated and shared information for thousands of technologies (seven of which were located in the NASA Ames booth) at the 82,000-sq. ft. exhibit area of the Anaheim Convention Center.

NASA *TechBriefs*, the event sponsor, selected the virtual wind tunnel, which consists of a prototype system of special-purpose software

Continued on page 6