NASnews
November - December 1998 • Volume 3, Number 6

In this issue...

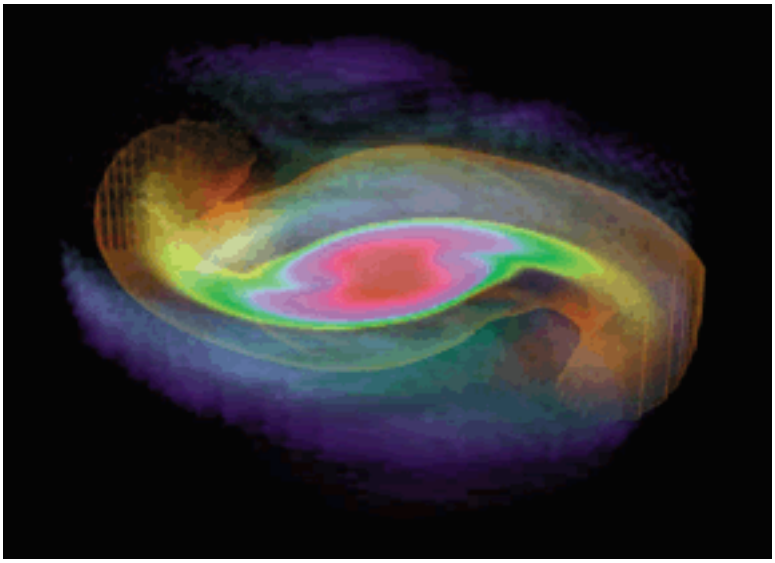
Researchers Work to Make High-Fidelity Simulations Faster, Easier

by Wade Roush

The aerodynamic simulations produced by high-fidelity, three-dimensional computational fluid dynamics (CFD) codes such as OVERFLOW can impress observers with their detail, accuracy, and beauty. But because engineers building new aerospace vehicles don't have weeks to wait for overburdened supercomputers to churn out such masterpieces, high-fidelity codes figure far less prominently in the day-to-day aerospace design process than both lower-fidelity and two-dimensional codes.

Now, however, the NAS Systems Division is contributing to research aimed at making high-fidelity simulations more routine. In conjunction with ongoing tests of a model transport aircraft in Ames Research Center's 12-foot Pressure Wind Tunnel, researchers will generate analyses of the transport virtually overnight, thanks to NASA program managers who agreed to allocate dedicated time on CRAY C90 and Silicon Graphics Origin2000 supercomputers at the NAS Facility.

To The Article...



Two neutron stars merging.

NASnews

▶ Main Menu

November - December 1998 • Volume 3, Number 6

In this issue...

Researchers Work to Make High-fidelity Simulations Faster, Easier

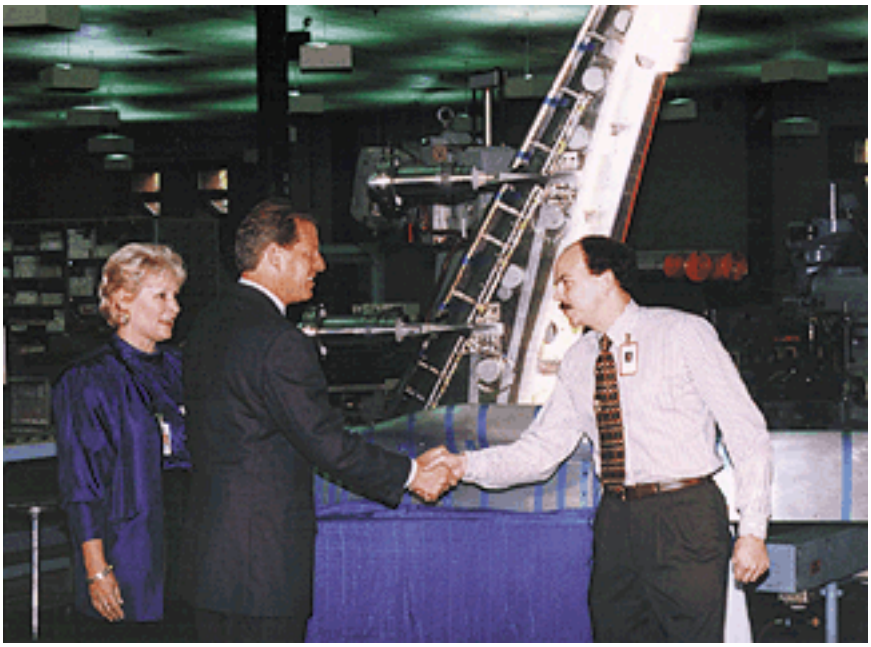
by Wade Roush

The aerodynamic simulations produced by high-fidelity, three-dimensional computational fluid dynamics (CFD) codes such as OVERFLOW can impress observers with their detail, accuracy, and beauty. But because engineers building new aerospace vehicles don't have weeks to wait for overburdened supercomputers to churn out such masterpieces, high-fidelity codes figure far less prominently in the day-to-day aerospace design process than both lower-fidelity and two-dimensional codes.

Within This Article...

Now, however, the NAS Systems Division is contributing to research aimed at making high-fidelity simulations more routine. In conjunction with ongoing tests of a model transport aircraft in Ames Research Center's 12-foot Pressure Wind Tunnel, researchers will generate analyses of the transport virtually overnight, thanks to NASA program managers who agreed to allocate dedicated time on CRAY C90 and Silicon Graphics Origin2000 supercomputers at the NAS Facility.

"In the past, most aerospace design has been done using wind tunnels and low-fidelity analyses--analyses that are fast but not particularly accurate," explains Yehia Rizk, a research scientist in the Ames Aeronautical Information Technologies Division. "We're adding a new element to the design process: high-fidelity codes."



Vice President Al Gore greets Glenn Hardin, an engineer from Micro Craft Inc., the Tennessee company that built the scale model of the High Wing Transport (in background) being tested in the 12-foot Pressure Wind Tunnel at NASA Ames Research Center. Researchers at The Boeing Company and Ames have teamed to study modifications to the wing flaps to improve the HWT's aerodynamic performance.

Acceleration is 'Unusually Challenging'

Accelerating high-fidelity simulations has proved unusually challenging for Rizk and other CFD experts. For one thing, the grids that represent the vehicle and flow fields in CFD computations are typically constructed through manual input, a process requiring weeks or months of a researcher's time. Rizk and his colleagues, however, are attempting to automate parts of this process. In addition, Rizk and NAS scientific consultants are employing a number of ingenious programming tricks in order to make the most efficient use of the available dedicated time on the supercomputers.

If significant acceleration can be achieved, the payoff could be considerable. One goal of NASA's aeronautics and space technology enterprise is to help the aerospace industry cut the design cycle time for new aircraft in half. Speeding up numerical simulations and integrating them more fully into the design process will be key to meeting this goal, according to William Van Dalsem, chief of the Aeronautical Information Technologies Office.

< Typically, an aerospace firm takes many months to get through the simulation, model-building, and wind tunnel testing phases of the design process, says Van Dalsem, and the cycle is usually repeated three or more times before a vehicle design is finished. But NASA's Advanced Subsonic Technology Program, the Information Technology Base Program, and the High Performance Computing and Communications (HPCC) Program "are working together to reduce both the time to complete the cycles and the number of cycles required to reach a final design," he says.

'Pursuing Improved Performance'

The studies in the 12-foot Pressure Wind Tunnel are one part of that effort. This summer, researchers installed a scale model of one side of a subsonic transport, designated as the High Wing Transport (HWT), in the wind tunnel. The team, including investigators from The Boeing Company, Ames' Aeronautics and Information Systems Directorate, the HPCC Program, and the Ames Center Operations staff, is currently studying whether modifications to the flaps might enhance the HWT wings' aerodynamic performance at high angles of attack.

While the tests proceed, scientists at the wind tunnel are also analyzing computer models of the HWT wing using two-dimensional and lower-fidelity CFD codes including INS2D, PMARC, and TIGER. The processing and memory requirements of these codes are low, so this step can be repeated many times using different flap configurations, helping designers single out the changes with the most promise for improved performance. Modified designs are used to build new model components for further wind tunnel testing.

Overcoming Obstacles

To help designers narrow their options and demonstrate that high-fidelity CFD solutions can be produced within the time frame of a traditional wind tunnel session, Rizk and other Ames investigators are simultaneously carrying out more detailed simulations of the HWT wing. "We're testing whether we can overcome two major obstacles that have in the past prevented the use of high-fidelity codes such as OVERFLOW in the design process," says Rizk.

The first obstacle is the need for a grid upon which to run the codes. Translating the 3-D computer-aided design geometry provided by aircraft designers into a high-resolution grid suitable for CFD requires a large up-front investment of time; a group led by Karlin Roth of the

AST program spent nearly eight months developing and applying tools to create a baseline 30- million-point grid representing the complete HWT.

Rizk uses data from the wind tunnel tests and the low-fidelity simulations to decide which variation of this baseline configuration to simulate using OVERFLOW. But even though as little as 10 to 20 percent of the standard grid needs to be changed, modifying these areas manually would consume too much time. "Even that small part could cause problems if we had to do it all by hand," Rizk says.

His solution to this problem takes advantage of the work Roth's group has already done. Rizk and his coworkers have "parameterized" key sections of the baseline grid, allowing these sections to be redrawn automatically by grid-generation software according to geometric parameters, such as the gap or overhang of the flap, supplied by the designer. "Just by changing the parameters, you get different shapes," while the rest of the grid remains unchanged, Rizk explains. For future design tests, Rizk's group is working on ways to further automate the gridding process.

The vast number of calculations involved in a high-fidelity, 3-D aerodynamic simulation creates the second problem. Even using supercomputers, high-fidelity analyses of complex grids normally take weeks or months to complete, since these jobs must share processors and memory with many other jobs. However, for the HWT test, NAS is providing dedicated time on the CRAY C90 (Von Neumann) and the Origin2000s. (Only the Origin2000s will be available after October 1, when NAS will turn Von Neumann over to CoSMO, the Consolidated Supercomputing Management Office.) According to an estimate by Johnny Chang, one of several NAS scientific consultants advising Rizk, Von Neumann's 16 processors can crunch through a simulation on the scale of Rizk's in about 20 hours when they aren¹t diverted to other jobs.

Saving Every Last Nanosecond

But even without competition from other users, Rizk is hard-pressed to keep turnaround time that short. Chang's estimate assumes that every trick in the book is being used to augment the simulation's speed, such as limiting I/O time by leaving data in the core memory, storing related data in contiguous blocks on disk drives, and pre-specifying the computation's memory requirements. Rizk's group is also starting to take advantage of recent work to make OVERFLOW run more efficiently on

the Origin2000s (see NAS News, May-June ¹98).

However the HWT tests turn out, Rizk, Van Dalsem, and other NASA scientists are optimistic that given the proper programming techniques and computing resources, more and more aerospace designers will look to high-fidelity simulations for guidance. "If you're allowed dedicated access to all or most of the processors in a machine, you might get a solution in one day," says Rizk. "Especially if you combine it with lower-fidelity codes, that begins to become attractive to the designer."

For more information on techniques for speeding up high-fidelity simulations, contact the NAS scientific consultants at **nashelp@nas.nasa.gov**.

NASnews

▶ Main Menu

November - December 1998 • Volume 3, Number 6

In this issue...

New Charon Software Library Speeds Parallel Conversion

by Rob Van der Wijngaart

Writing scientific programs for large-scale parallel processing is a complicated and time-consuming task. Converting existing sequential (vector) programs is almost as difficult, unless the conversion can somehow be automated. Unfortunately, many conversion tools, such as parallelizing compilers, compiler directives for (virtual) shared-memory systems, or interactive translators like CAPTools, are limited in their capabilities to express parallelism, distribute data, or analyze data dependencies. For important classes of applications, the programmer is forced to do the parallelization by hand. Message passing is the technique almost universally suited for such parallelization, but it is a burden for the programmer. The Charon library, under construction in the NAS Systems Division, aims to ease that burden.

Within This Article...

Message Passing Versus Other Tools

The advantages of message passing are well known. The user has complete control over exploitation of concurrency and distribution of data. The separate processor address spaces and the explicit message passing calls provide a simple programming model. They also enable tuning, since the often costly communications are completely managed by the programmer.

The major disadvantage -- contrary to common belief -- is not the placement of message passing calls. In a typical application, the fraction of lines of program text involving communications is small. What

makes message passing truly cumbersome in most scientific computing programs of interest to NAS is the explicit management of the domain decomposition; that is, the restriction of data structures (mostly arrays) and operations to individual processors. Each processor "sees" only a small window of the entire distributed array. Moreover, message passing programs cannot be developed gradually from a serial code. Domain decompositions are all-pervasive, and the entire program must be converted all at once.

This puts message passing at a distinct disadvantage compared to the shared-memory paradigm, which allows piecemeal conversion of legacy code through parallelizing directives.

Charon Library Helps Programmers

The Charon library offers a vehicle for easy development of efficient message passing programs for structured-grid applications. Both legacy code conversion and construction from scratch are supported. Charon provides a small set of functions that create and manipulate distributed arrays. The library is also portable, requiring no special operating system or hardware support; it is programmed in ANSI C, is callable from Fortran and C, and is built on and interoperable with the de facto message passing standard MPI (Message Passing Interface).

Three-tiered Design

One of the main design goals of the Charon library has been to enable rapid parallel program development and debugging, and subsequent piecemeal performance tuning.

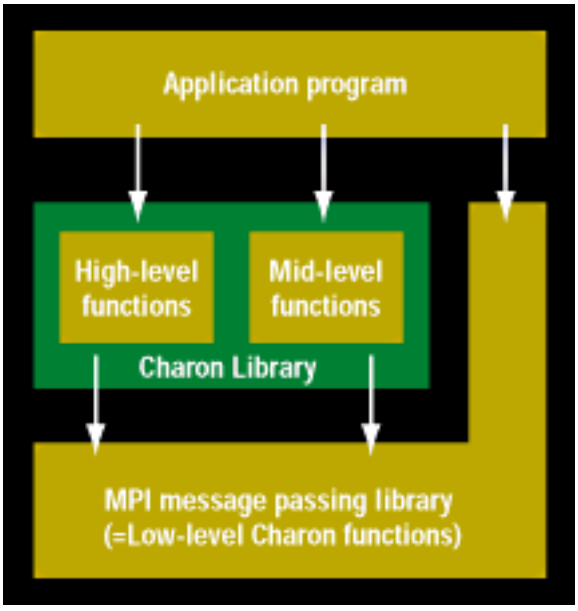


Figure 1. Charon Software Structure.

To support this approach, functions that manipulate distributed arrays are provided at three levels of abstraction, as depicted in Figure 1. The highest level is the simplest to use, but also the least efficient. It is a device that emulates serial program execution on distributed data. Salient features are application of the "owner-computes" rule (only the processor that owns an array element may update it) and automatic synchronization. Strict sequential consistency is guaranteed. The sole purpose is to distribute the data without any change in program structure. Invocation of the Charon library at this level, which takes the form of library calls replacing assignments to elements of distributed arrays, is the first stage in the parallelization of a user program.

The intermediate level is also easy to use, but is more efficient. It consists of a collection of communication routines, augmented with functions that control granularity, synchronization, and concurrency, and that allow relaxation of the owner-computes rule. Indexing of distributed arrays is global, so the image of shared memory is retained.

The lowest level of abstraction involves local indexing of arrays -- that is, direct local memory access without library intervention, and (possibly) explicit message passing; this level offers the highest efficiency. Calls at all three levels can be freely mixed, allowing a gradual transition from low to high efficiency and a peaceful coexistence of paradigms.

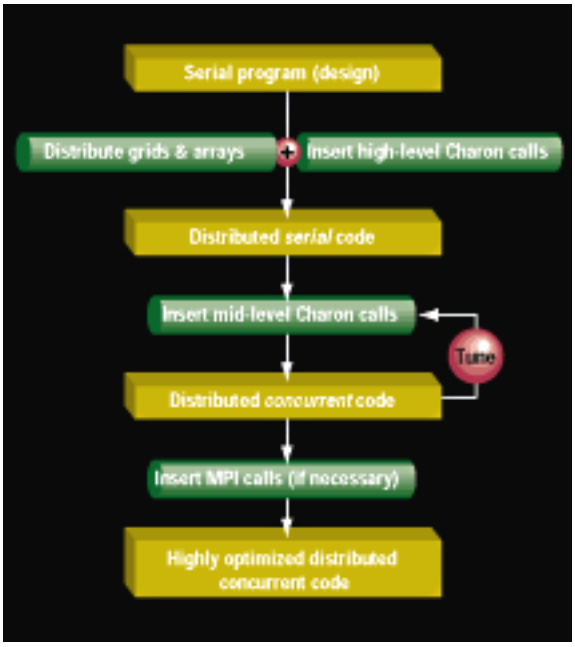


Figure 2. Sample parallelization flow chart.

An example of how Charon might be used to parallelize a code is given in Figure 2. In the course of improving performance, high-level function calls are gradually replaced by mid- and low-level constructs. As a consequence, top-performing parallel codes derived using Charon will often look very similar to message passing codes, although many of the common manipulations do not have to be coded explicitly by the user, but are provided as tools in the toolkit. The important difference is that Charon codes are created in a piecemeal fashion, with support for rapid prototyping and validation.

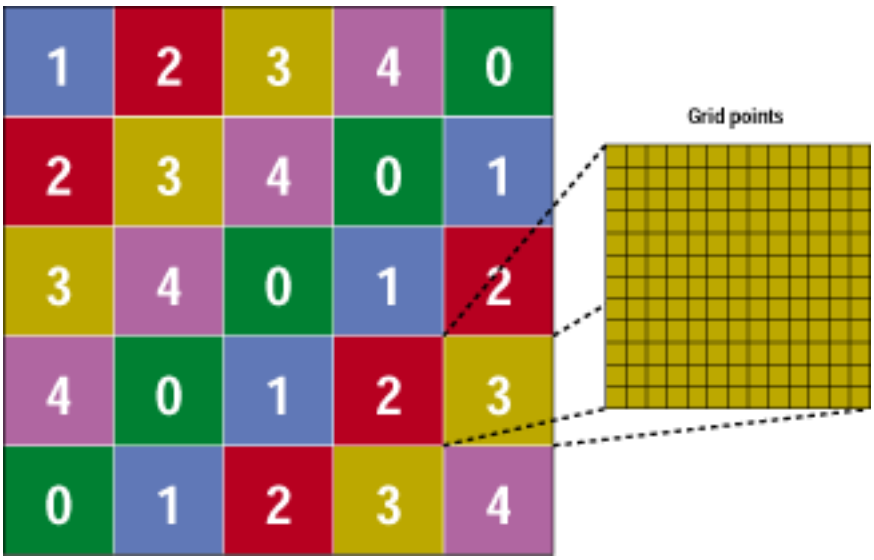


Figure 3. Two-dimensional multipartition grid decomposition of five processors.

Parllelizes NAS Benchmark Code

As an example of the use and parallel performance of Charon, the author has parallelized the serial version of the NAS Scalar Penta-diagonal (SP) Benchmark program (NPB), using the so- called multipartition parallelization strategy. SP constitutes a synthesized version of the flow solver algorithm employed within the well-known OVERFLOW code. Multipartitioning maintains a near-perfect load balance while keeping communication requirements low. An example of a two-dimensional grid divided into 25 sub-blocks and distributed in a regular pattern across five processors, numbered zero through four, is shown in Figure 3. Numbers within the sub-blocks indicate processor ownership. Multipartioning is even more efficient when extended to three dimensions, as required by SP.

Despite the fairly high complexity of multipartitioning, the size of the original code increased by less than 20 percent, from 3,072 to 3,663 lines of Fortran, when using Charon library calls. Moreover, the straightforward conversion was carried out one subroutine at a time. This was possible due to Charon's capability to map easily between different data distributions, including between multipartitioning on many processors and an undivided grid on a single processor.

Performance of the code for a grid of 643 points (Class A) improved from 61.2 MFLOP/s (millions of floating point operations per second) on one processor to 2,711 MFLOP/s on 81 processors of a Silicon Graphics Inc. Origin2000 system at the NAS Facility, with parallel efficiency of 55 percent.

User Controls Data Layout

Charon is intended for difficult structured-grid problems, especially those involving implicit solution techniques. It provides very general, possibly dynamic, data distributions. Problems that do not need this flexibility, or for which prepackaged parallel solvers exist, may be more easily solved using systems such as KeLP, OVERTURE, or PETSc. Yet even for such problems the Charon approach offers advantages, because of the control the user has over data layout (for example, reuse of workspace and array padding or index interchange to improve cache utilization) and communications, and because of the convenience of incremental parallelization.

The Charon toolkit is still under development. Facilities are being added for fast parallel I/O and for legacy code constructs like overindexing. An

alpha release of the software is planned for January 1999. Details regarding the parallelization and can be found online. For questions concerning the functionality and status of Charon, send email to the author at **wijngaar@nas.nasa.gov**.

▶ Main Menu

November - December 1998 • Volume 3, Number 6

In this issue...

High-speed Processor Techniques

Converting Jobs From the CRAY C90 and J90 Systems to the SGI Origin2000s

by Terry Nelson

This article discusses some of the issues involved in converting jobs from the CRAY C90 and CRAY J90 systems to run on the Silicon Graphics Inc. (SGI) Origin2000s at the NAS Facility. The aim is to present practical comparisons and directions, which should help some users get started in this effort, and help avoid at least some of the initial frustrations involved with conversion.

Within This Article...

Both Systems Use PBS, Clusters

PBS, the Portable Batch System (see NAS News, September-October '98), is installed on all of the systems at NAS, but its usage can vary slightly on each machine. Both the Cray "parallel vector processor" (Cray Research terminology) and the Origin2000 worlds now consist of clusters. On the Cray side, the C90s (Von Neumann and Eagle) and the four J90s (Newton 1-4) constitute a cluster.

This means users can log onto any of these hosts, and jobs submitted with the *qsub* command can then run on any of these systems. Home filesystems are distinct on each machine, so users are encouraged to have parallel file structures on all three systems in order to run their jobs in the first available slot on any cluster member -- which in turn enhances throughput.

If that's not practical, the home filesystems are cross-mounted, so by using the */R* notation you can run in another host's directory. For example, for a csh script:

```
if ('hostname'  != 'eagle' ) then
        set MYHOME = /R/eagle$HOME
else
        set MYHOME = $HOME
endif
set workdir = $MYHOME/thisjobdir
cd $workdir
...
```

The alternative method is to use rcp/scp to move files. Be careful with executables, however, since binaries compiled on the C90s or J90s should not, in general, be run on the other type of hardware, due to performance considerations.

The Origin2000 side also uses a cluster arrangement. Here, users log onto Turing, and jobs submitted with qsub will run on either Hopper or Steger. The Turing home filesystems are NFS-mounted to both Hopper and Steger, so the *cd* command will put you in your home directory no matter which system you're running on.

On both platforms, use *qstat -a* to see the queues, and *qstat -q* to see the queue limits.

Filesystems, Scratch Space

NAS currently has two production mass storage systems (Chuck and Scott), which handle long-term storage and file staging for all the Cray and Origin2000 systems.

For temporary scratch file space during jobs, C90 users have available large filesystems on SSD, in /fast ($FASTDIR), and on disk (cached through SSD) in /big ($BIGDIR), and in /tmp ($TMPDIR). J90 jobs also have a /tmp and /big directory, which is particularly important on the Newtons because of the performance implications of NFS mounting of the home filesystems.

There are no /big or /fast directories on the Origins, but there are /scratch1 and /scratch2 directories unique to each system. These are cross-mounted and should

be referred to using the /cluster notation; for example, either of the commands below works from any of the Origins:

```
more /cluster/hopper/scratch1/usrname/fname
```

or

```
cd /cluster/steger/scratch2/usrname/workdir
```

Just substitute the appropriate names. Users should always use the /cluster notation when they "cd" to any /scratch directory, so that $PBS_O_WORKDIR will be appropriately set. Since files in the /scratch1 and /scratch2 directories are not automatically deleted at the end of the job, users may be tempted to use these directories as a sort of mid-term storage. However, space in these directories is limited, and when total usage approaches 80 percent, *files will be deleted*, based on an algorithm that considers the age and size of the files.

$SCRATCH1 and $SCRATCH2 point to local /tmp directories, which should be used for temporary job file space.

Be Aware of Home Filesystem Limits

Home filesystems on the Crays each have a limit of 1,536 megawords (MW). As this limit approaches, various warnings and restrictions occur. Use the *quota* command to determine your available space. Since files in /tmp, /big, and /fast are deleted at job end, keeping files there long term is not an issue.

Home filesystems on the Origins have a (soft) limit of 300 megabytes (MB). When this is exceeded, various warnings and restrictions occur. Use the *quota -v* command to determine your available space.

Use Modules for Greater Control

Module use on the Origin2000s is similar to that on the Crays. On the Origins, there are default products for most of the compilers or libraries, but it is still recommended that you use the module commands for greater control of the release level your program is using. On the Origins, compiles that use special libraries may require a trailing library parameter, such as -lmpi (MPI), -lpvm (PVM), or -lsma (SHMEM).

Origins Still Have f77, f90

On the Cray (UNICOS) systems, the Fortran compiler, CF77, has not been supported by Cray for several years, and much effort has gone into converting all programs to the new compiler, CF90 (see NAS News, September-October ¹98 for

tips on getting better performance with CF90). On the Origin systems, users will find f77 as well as f90 compilers.

Hardware Differences

Some of the major differences between the Crays and the Origins derive from their different hardware layouts. The Crays have a traditional memory layout with from 1 to 16 CPUs assigned to a job on a time-shared basis. The Origins have a more involved structure, which consists of hubs of linked nodes. Currently, Hopper has 32 nodes and Steger has 64. Each node has 2 CPUs and 512 MB of shared memory. Requests for CPUs and/or memory are incremented up to the minimum set of nodes that satisfies both requirements. This memory layout is called NUMA (nonuniform memory access).

Because of the way the nodes assigned to the job may be connected, with varying numbers of hops between nodes, even jobs that do not depend heavily on disk activity will have more variation in performance from job to job on the Origins than on the Crays.

Implications for Optimization

As always, understanding hardware characteristics is important for obtaining optimal performance for a code. On the Crays, the central optimization concept is probably the vector. Long loops take advantage of the vector hardware registers, as well as pipelining hardware. On the Origins, code that vectorizes well will generally run well. As you refine your code, it's also important to be aware of the caching mechanism in the Origins. The R10000 CPU chips have a 32-kilobyte cache for instructions and another for data (primary, or L1 cache). The node board also has a 4-MB cache (secondary, or L2 cache). The details can be involved, but the point is that if, for example, arrays and loops can be kept in cache, then performance gains may be significant.

Multitasking -- 'Slowly Converging'

The multitasking landscape is a little different between the Origins and the Crays, but they are slowly converging. Autotasking is not supported on the Origins and is, in fact, officially "outmoded" on the Crays. (What this may imply for future support is unclear.) The *shmem* command set is available on both platforms, and is in the mpt module.

The newest solution on the Origins, and soon to be implemented on the Crays, is the OpenMP Application Programming Interface. A slightly older approach on the Origins involves the use of C$DOACROSS and mp_ directives. The "man mp" pages provide some information, but to get more details, check out the web-based manuals mentioned in the man pages.

Two message passing interfaces, MPI and PVM, are available on all Cray and Origin systems at the NAS Facility through the mpt module. On the Crays, a socket version of MPI called "mpirun -np" is available, along with the faster shared memory version, "mpirun -nt". On the Origins, only the shared memory version, which is also called "mpirun -np", is available.

Documentation Pointers

The most accessible documentation for these systems is found on the World Wide Web. For the Cray parallel vector processors, see the Cray Research Online Software Publications Library, maintained on a NAS server.

For the Origin2000s, see SGI's Technical Publications Library.

A good resource for new users of these NAS-supported systems is the QuickStart web page.

NASnews

▶ Main Menu

November - December 1998 • Volume 3, Number 6

In this issue...

Credits

Executive Editor: Bill Feiereisen

Editor: Jill Dunbar

Senior Writer: Wade Roush

Contributing Writers: Terry Nelson, Rob Van der Wijngaart

Image Coordinator: Chris Gong

Special thanks to: Richard Anderson, Ray Cosner, James Donald, Don Durston, Zann Gill, John Hardman, Randy Kaemmerer, Paul Kutler, Mary Livingston, Dave Korsmeyer, Don Leopold, Marcia Redmond, Karlin Roth, Dale Satran, Brian Smith, Yehia Rizk, William Van Dalsem, David Young

Editorial Board: Cristy Brickell, Jill Dunbar, Chris Gong, Bill Feiereisen, Nateri Madavan, Patrick Moran, George Myers, Wade Roush, Harry Waddell

Many thanks to Cristy Brickell for volunteering her time and ideas to the NAS News Editorial Board.