



In this issue...

[OVERFLOW Speeds
SGI Origin200
Performance](#)

[Secure Shell Keeps
Accounts Safe](#)

[Group IDs and Account
IDs](#)

[Web Pages Show Lunar
Prospector Satellite Data](#)

[PBS Offers 24-hour
Support](#)

[New Tool for Interactive
Analysis of CFD Data](#)

[Network Analysis and
Design Cookbook](#)

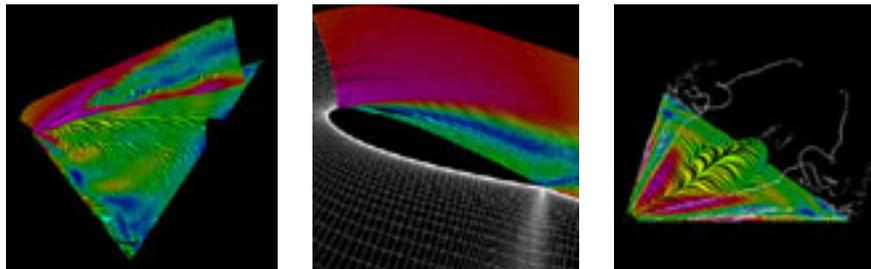


OVERFLOW Gets Excellent Results on SGI Origin2000

The July-August 1997 issue of NAS News reported on the parallel performance achieved with the ARC3D code on a 64-CPU Silicon Graphics Inc. (SGI) Origin2000. The result -- 6.4 GFLOPS of sustained performance on a 7-million point problem -- was a shock to many NAS researchers. A new project was immediately undertaken to test the Origin2000's parallel architecture even more strenuously -- this time, using the well-known production CFD code, OVERFLOW. Initial results show performance levels at 72 percent of a dedicated 16-CPU CRAY C90 -- with only minimal code changes.

[To the article...](#)

New Software Tool Improves Interactive Analysis of CFD Numerical Flows



Graphics by Han-Wei Shen. Click on thumbnail for information.

[To the article...](#)

**In this issue...**

[OVERFLOW Speeds
SGI Origin2000
Performance](#)

[Secure Shell Keeps
Accounts Safe](#)

[Group IDs and Account
IDs](#)

[Web Pages Show Lunar
Prospector Satellite Data](#)

[PBS Offers 24-hour
Support](#)

[New Tool for Interactive
Analysis of CFD Data](#)

[Network Analysis and
Design Cookbook](#)

OVERFLOW Gets Excellent Results on SGI Origin2000

by [Jim Taft](#)

The [July-August 1997](#) issue of NAS News reported on the parallel performance achieved with the ARC3D code on a 64-CPU Silicon Graphics Inc. (SGI) Origin2000. The result -- 6.4 GFLOPS of sustained performance on a 7-million point problem -- was a shock to many NAS researchers. A new project was immediately undertaken to test the Origin2000's parallel architecture even more strenuously -- this time, using the well-known production CFD code, OVERFLOW. Initial results show performance levels at 72 percent of a dedicated 16-CPU CRAY C90 -- with only minimal code changes.

OVERFLOW was chosen as the second test case because it represented a "toughest-case" scenario; it is highly "vector" oriented and historically has not run well on microprocessor-based systems. OVERFLOW is written in FORTRAN 77 and consists of approximately 87,000 lines of code. It offers users the ability to select from a wide range of proven solvers, smoothers, and turbulence models to solve the three-dimensional (3D) time-dependent Reynold's-Averaged Navier-Stokes equations for fluid motion at moderate mach numbers.

Within This Article...

[OVERFLOW's
Approach to Solution](#)

[MPI Approach](#)

['Minimal Change'
Philosophy](#)

[A New Approach:
Multi-Level
Parallelism](#)

[OVERFLOW-MLP](#)

[OVERFLOW-MLP
Test Results](#)

[Answers to Common
Questions](#)

[The Future of
OVERFLOW-MLP](#)

OVERFLOW's Approach to Solution

OVERFLOW's typical solution approach is to solve for the fluid motion in the region of interest by determining the flow through a series of connected 3D grids, or zones, that cover the region in a patchwork fashion. The calculation advances in time for all zones until the solution has converged to the user's satisfaction. Typical problem sizes range from small calculations involving about one million points to simulations of an entire aircraft requiring tens of millions of points.

On the CRAY C90, the standard solution approach is to sequentially process each 3D zone through one timestep. Once all zones have been advanced to the new time, the process repeats and all zones are advanced to the next time level. Typical solutions require thousands of timesteps to reach convergence.

Typical performance for the standard C90 version of the code is around 450 million floating-point operations per second (MFLOP/s) on a single CPU, making it one of the better "vector" codes in the CFD arena. Typical microtasking parallel efficiency for the code is about 11 of 16 CPUs when the C90 is in dedicated mode. This is not a particularly efficient use of the system, but typical of "good" classic parallel-vector codes on the C90.

MPI Approach

There are other ways of achieving parallel performance with OVERFLOW. For example, the code has also been converted to solve the flow field in a more coarsely parallel fashion using MPI, the Message Passing Interface. The MPI approach is theoretically more efficient than the standard C90 code in that it attempts to parcel the zones to different processors in such a way that many zones are being processed simultaneously at each timestep. This methodology is still under development for OVERFLOW, and performance is still uncertain for the general case.

One drawback to the MPI approach is that it is subject to the latencies and added overhead imposed by MPI's message passing library. This library of messaging subroutines is used to communicate boundary values and other data between the zones as the solution progresses. Another drawback is added code complexity; the current MPI version of OVERFLOW requires an additional 20,000 lines of code to support this

form of parallelism.

'Minimal Change' Philosophy

When deciding how to efficiently optimize OVERFLOW for the Origin2000, it became clear that the best approach was to take advantage of the coarse-grained parallelism concept within the MPI code and at the same time avoid the pitfalls of MPI's labor-intensive code conversion.

As with the previous ARC3D conversion effort, a prime focus was to adopt an approach that required minimal changes to the production version of the code. This was essential to keeping the porting effort small, and critical to ensuring that users would be comfortable with the new approach. In particular, it was determined that the solution methodology must be robust and easy to use in order to be successfully adopted in other mainstream production codes.

Another important consideration that demands the "minimal change" philosophy was that production codes are often customized by users for their particular needs. An optimized code that is massively different from the standard release is almost useless to these users because the effort needed to incorporate modifications is prohibitively time-consuming and difficult to debug.

A New Approach: Multi-Level Parallelism

With these concerns in mind, a new form of parallelism was developed that is applicable to a wide range of CFD codes. This new approach falls into the generic category termed "Multi-Level Parallelism" (MLP). At present, this concept is vague and is viewed very differently by almost all high-end system vendors.

The MLP approach is usually distinguished from the standard coarse-fine parallelism that is possible in MPI codes. The term MLP is generally associated with shared-memory, multiprocessor architectures, in which the shared memory features allow users to dispense with message passing altogether. As its name implies, code developed under MLP contains multiple levels of parallelism. In general, this means fine-grained parallelism at the loop level, using compilers to generate the parallel code and, at the same time, performing parallel work at a coarser level using a standard Unix fork system call to effect any needed

coarser levels of parallelism. Data that is truly "global" is shared among the forked processes through standard shared-memory arenas.

With the MLP approach, "messages" that are needed in the MPI code become nothing more than standard memory references. As a result, virtually all of the user's time and effort in building code for synchronization and communication disappears -- with a massive reduction in the code development effort.

OVERFLOW-MLP

The OVERFLOW-MLP code developed in the NAS Systems Division is based on the standard C90 parallel-vector version of OVERFLOW. The total time to generate this new parallel version and correctly execute three widely varying test cases was around three weeks.

'The major outcome of this initial work is not that the relatively inexpensive Origin2000 nearly outperformed the C90, but that the new trend in computer architectures can successfully be used to solve the largest problems of interest to NASA.'

About 250 lines of code were inserted or changed in the base MLP version. Most changes came from the addition of four small subroutines. If a user removes these calls, then the code reverts to its original form and executes at typical C90 performance. The end result is that one code executes with excellent efficiency on both Cray vector systems and massively parallel microprocessor-based systems.

The OVERFLOW-MLP code organizes the calculations such that groups of zones are processed by groups of CPUs in parallel. An initial distribution of zones is made across a user-defined number of CPU groups so that the work is approximately equal. The number of CPUs in each group is adjusted to further load-balance the work. Load balancing is fully automatic and dynamic in time. During the solution process, each group of CPUs advances the time level for its assigned zones until the solution converges to the degree needed.

OVERFLOW-MLP Test Results

To fully stress-test both the MLP methodology and the Origin2000 system, a very large test case was selected. The test dataset was provided by Karlin Roth (high-lift CFD team lead, NASA Ames

Applied Computational Aerodynamics Branch) and consisted of 153 zones totaling over 33 million grid points. This is thought to be the largest problem ever solved at the NAS Facility. Typical executions of this dataset are performed on a dedicated 16-CPU C90, taking many hundreds of CPU hours.

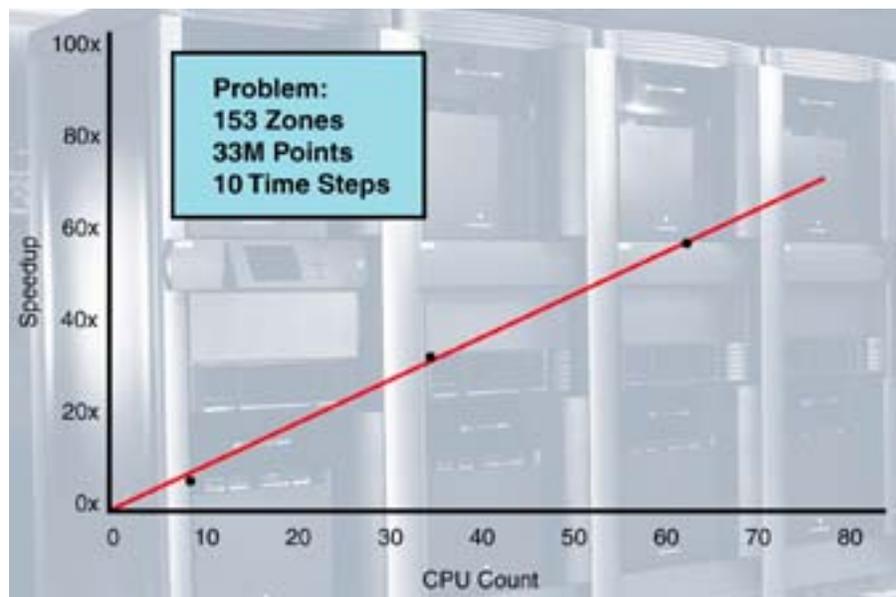
System	No. CPUs	GFLOP/s
CRAY C90	16	4.6
Origin2000	64	3.3

The table summarizes the relative performance on these two systems. The test executions began with a restart and continued for an additional 10 timesteps. Surprisingly, all I/O-related activity on the two systems required about the same time -- around 180 seconds.

The important and exciting aspect of this work is that the Origin2000 performs this very large calculation at a rate that is 72 percent of the full dedicated C90. Furthermore, it gets this result using OVERFLOW's standard "good" vector code -- and with only a few weeks of optimization work.

Answers to Common Questions

One of the first questions commonly asked in studies of this nature is: What are the scaling issues, and how well did the Origin2000 scale on the problem? The chart below reveals the dramatic answer -- scaling to 64 CPUs is completely linear. There is none of the tailing-off that is frequently seen when some aspect of a parallel system's architecture (or the numerical algorithms) begins to infringe on performance.



To be sure, the linear scaling in this case is somewhat assisted by the sheer problem size. "Back-of-the-envelope" calculations and test executions of portions of the problem as it might execute on larger systems indicate that this test case may scale to several hundred CPUs. This hypothesis will be further tested over the next few weeks on a 128-CPU Origin2000.

Another commonly asked question is: What improvements in performance can be had with some level of single CPU optimization? The fact is that the code is not very efficient in its reuse of level-1 cache, and subsequent speedups can be expected with some tuning effort in this area. It is estimated that two work-months of optimization will result in a code that could be two times faster on the large test problem.

The Future of OVERFLOW-MLP

The new microprocessor-based, distributed shared-memory systems are widely perceived as the standard high-performance architectures for the foreseeable future. Fortunately, they appear suitable to the needs of the CFD community, and the MLP approach naturally maps to these new systems.

MLP standardization efforts have already begun, with the recent announcement of "openMP," supported by SGI and others. Clearly, MLP is here to stay. At present, the NAS Facility is the only site in the U.S. to successfully implement MLP in a fully production-qualified CFD code.

The major outcome of this initial work is not that the relatively inexpensive Origin2000 nearly outperformed the C90, but that the new trend in computer architectures can successfully be used to solve the largest problems of interest to NASA. This is the first time that this has been true since the computer industry began touting massively parallel architectures 15 years ago.

The success of this study opens the door to some intriguing possibilities: one-hundred-million-point problems can be attempted today. Large simulations on larger systems can be done in hours instead of days. And for the first time in many years, problem sizes can increase significantly and still be practical to solve.

For more information on the OVERFLOW-Origin2000 study, send email to jtaft@nas.nasa.gov.



**Within This Article...**[Optimizing Challenges](#)[Porting in Minutes](#)[Vector-Oriented Changes](#)[Straightforward Optimization](#)[Test Series](#)[Impressive Results](#)[Continued Improvements](#)**In This Issue...**[SGI Origin2000 Shows Promise](#)[IPv6, New Internet Protocol](#)[Large CFD Code Successfully Ported to Cray Fortran 90](#)[Metacenter Success, Challenges](#)[Parallel Supercomputers: Price vs. Complexity](#)[Portable Code Optimization Study](#)

Initial SGI Origin2000 Tests Show Promise for CFD Codes

by [James Taft](#)

Several projects are under way to fully explore the performance characteristics of the new Silicon Graphics, Inc. (SGI) [Origin2000 system](#), installed at the NAS Facility last April. One of these efforts centered around porting and optimizing ARC3D, a well-known computational fluid dynamics (CFD) code. Preliminary results show promise for achieving very high sustained performance levels, which could bode well for several other production CFD codes that run on NAS systems.

The system (purchased jointly by NASA's Data Assimilation Office and the Information Technology Program at Ames Research Center) is composed of 64 RISC-based central processing units (CPUs), with 16 gigabytes (GB) of globally addressable main memory, and 330 GB of available disk space. The system is one of the first to incorporate the Non-Uniform Memory Access (NUMA) architecture, a design that allows the construction of large systems with hundreds or even thousands of processors at a modest cost.

A major factor in purchasing the Origin2000 was its initial performance on ARC3D, which uses algorithms that, although popular, are known to be "unfriendly" to cache-based RISC systems. The version of ARC3D that was tested was a simple three-dimensional (3D) transient EULER variant using a single rectilinear grid. Differencing is done implicitly using an Alternating Direction Implicit (ADI) solver, which sweeps through each of the cardinal directions one at a time, with partial updating of the fields after each sweep. Historically, this solver has been very inefficient on cache-based systems.

Challenges in Optimizing CFD Codes

CFD codes are one of the toughest classes of problems to port to RISC-

[Web Help for Parallel Systems Users](#)

[Cray User Group Highlights](#)

based architectures. This is due to the relatively high ratio of data fetches to floating point operations that are typical of numerical algorithms. Excessive data fetches severely stress the cache and memory-management hardware. The problem is further compounded for NUMA systems due to variance in memory access times for remote fetches. In some cases, the situation becomes untenable.

Much experience with RISC systems and limited experience with NUMA systems shows that worthwhile performance can only be had by substantially rewriting core routines. Luckily, this isn't as onerous as it first sounds. In most cases, the majority of the computational burden occurs in just a few routines, so recoding can often be done in weeks or months instead of years.

ARC3D Ported in a 'Matter of Minutes'

The effort required to port the ARC3D code from the C90 to the Origin2000 was trivial -- in fact, it was completed in a matter of minutes. The code uses no external libraries and is syntactically Fortran 77 compliant. The biggest concern -- preserving 64-bit precision in the arithmetic -- was accomplished by specifying the **-r8** and **-i8** compiler flags.

The effort to optimize the ARC3D code was more extensive. About 1000 lines of code, involving five major routines, were rewritten over a period of about two months.

In general, the strategy for optimizing ARC3D on the Origin2000 was quite similar to the process used on the CRAY C90, although the details differed. First, the code was optimized for a single processor, followed by parallel optimizations across multiple processors. Because each Origin2000 processor is less powerful than a CRAY C90 or T90 processor, parallel efficiency is much more important for achieving good performance overall.

Parallel efficiency can only be achieved when careful attention is paid to ensuring that data accesses are as local as possible. This means the data must reside in the local processor's cache or local main memory. Most of the ARC3D optimization work was focused on ensuring this characteristic in the code.

Vector-oriented Optimization

It should be noted that both Cray and Origin2000 systems are "vector oriented." By design, RISC architectures are pipelined machines that perform well on the regular organization of data found in vector code. If the data could be contained in cache at all times, then classic Cray-optimized vector code would run well on RISC systems. However, Cray code typically "breaks" cache as a result of very long vectors and large datasets. The end result is that loops and common block structures must be reorganized to improve cache reuse.

Much of the editing work on the ARC3D code involved reversing the order of the indices in the field arrays and "fattening" loops by combining multiple 3D loops into a single 3D loop. In addition, the number of vector temporaries in the ADI calculation was greatly reduced.

Optimization Was 'Straightforward'

The ARC3D code was parallelized in a straightforward manner: The code was compiled with the **-mp** option to enable parallelism in the compile phase. This was combined with approximately ten **c\$doacross** compiler directives, which instruct the compiler to generate code that will execute the next Fortran "do" loop in parallel. These directives were placed at key spots within the code for the greatest parallel efficiency. This involved decomposing the 3D problem into groups of 2D planes, with each group assigned to a dedicated processor.

The efficiency of the parallel decomposition was enhanced by the utilization of two Origin2000 data-distribution directives that allow users to improve the distribution of their data throughout the system. These features were invoked by setting two environment variables. The first specified the user's default page size for data. The second specified the allocation policy for placing the data across the memories in the system.

The entire optimization effort consumed approximately two months, most of which was expended in learning the behavior of the system itself in response to different optimization approaches -- lessons that can be applied when optimizing other codes in the future.

Test Series Challenged Memory, Cache

Several different dataset sizes were considered during the test series.

Ultimately, the largest problem was selected for the majority of tests because it fully stressed the memory and cache systems and offered the worst case scenario. The grid for this problem was defined to be 194x194x194 (7,301,384 grid points). This size ensured that the problem would not fit into the aggregate of the cache memories and that the data would be scattered across many of the local memories in the NUMA-based Origin2000.

The test runs consisted of executing two timesteps on varying numbers of processors, and performing a residual error check to verify the accuracy of the solution. Performance timing spanned only the work performed for each timestep and the residual error calculation. A subsidiary test series of twenty timesteps verified that the timings gleaned from the two timestep runs were accurate and would linearly scale to higher timestep counts.

Preliminary Results 'Impressive'

Preliminary results on the 64-CPU Origin2000 are impressive. In a nutshell, ARC3D an historically "cache-abhorrent" CFD code was restructured with no change in the fundamental algorithms to yield over 6.3 gigaflops per second (Gflop/s) in sustained performance equivalent to past performance of this code on a 16-CPU C90.

Results for the 194x194x194 problem (1 and 64 CPUs) are presented in Figure 1, below. As can be seen, the major time consumers were the x-, y-, and zdirect routines, which continue to scale well on 64 CPUs. One of the strange artifacts of the NUMA environment in this particular case is that the zdirect routine scaled better than the xdirect routine, even though the xdirect routine is unit stride through the data and the zdirect routine is the worst stride through the data.

[Figure 1](#)

The overall ARC3D scaling for executions on 1 to 64 CPUs is presented in Figure 2, below. This chart shows virtually no scaling fall-off as the number of processors increases, and indicates that substantial gains are still to be had.

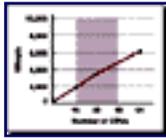


Figure 2

Continued Performance Improvements

Cursory examination of the ARC3D test results indicates that even better performance can be obtained simply by executing the code on systems with more processors. The current code utilizes about 100 of the total available Mflop/s per processor (peak is 380). Further optimization could improve this performance, resulting in even higher sustained Gflop/s ratings. The Origin2000 architecture may finally allow successful migration from traditional "vector" systems for some classes of production codes. Further tests on 128-CPU systems will better define the ultimate scalability.

For more information about porting and optimizing code for the Origin2000, send email to jtaft@nas.nasa.gov.

**In this issue...**

[OVERFLOW Speeds
SGI Origin2000
Performance](#)

[Secure Shell Keeps
Accounts Safe](#)

[Group IDs and Account
IDs](#)

[Web Pages Show Lunar
Prospector Satellite Data](#)

[PBS Offers 24-hour
Support](#)

[New Tool for Interactive
Analysis of CFD Data](#)

[Network Analysis and
Design Cookbook](#)

Secure Shell: New Measure to Keep Accounts Safe

by [Ayse Sercan](#)

For much of the last year, the NAS security and local area network teams have been testing the use of [Secure Shell](#) (SSH), a plug-in replacement for the Berkeley "r" commands, to improve security on systems at the NAS Facility. SSH will automatically encrypt packets sent across the Internet between a remote client and NAS systems, ensuring the security of passwords and data.

Originally written by Tatu Ylönen at Helsinki University of Technology, Finland, Secure Shell is now being maintained by Data Fellows and SSH Communications Security. It is estimated that at least 1,000 institutions in 40 countries use the software. At the NAS Facility, use of this package is encouraged as part of the larger security paradigm that protects NAS systems from misuse.

The software has two components: a server and a client. The server and client software run on all the Unix machines at the NAS Facility, with the exception of the Crays (vonneumann, eagle, and the newtons) and the Convexes (chuck, scott, and pancho). Testing on the Crays has begun, and Secure Shell should be available on those systems in February. The teams are currently examining the impact of Secure Shell on the Convexes.

Remote users will need to install the client software on their own systems as well. The client source code for almost all Unix-based operating systems is available free of charge, while Mac and PC users can purchase commercial versions for Windows 95, Windows NT, and

Within This Article...

[How SSH Works](#)

[Benefits of Secure
Shell](#)

[Call For 'Widespread
Use'](#)

MacOS for about \$100 (free versions are currently being developed).

How SSH Works

Secure Shell uses public-private key exchange on two levels: for authentication and encryption. To authenticate the host, SSH creates and permanently stores a cryptographic key (a seemingly random series of letters and numbers that is used to decode an encrypted message) on that system. If a third system tries to fool Secure Shell into sending it data by "spoofing" -- pretending to be the second system -- SSH will not be able to find the key and will warn the user that the session is not secure. In addition, once the user is logged in, the session is encrypted with a key that is dynamically created at the beginning of the session by the client and server software, and is used only once.

Public-private key encryption is simple. Both client and server build private and public keys, and they exchange public keys. Anything encrypted with a particular public key can only be decrypted with the corresponding private key. Here's how it works: the client software encrypts a packet with the server's public key, then sends it to the server. The server decrypts it with its own private key, formulates a response, encrypts it with the client's public key, and sends it back to the client. The client then decrypts the packet with its own private key. Fortunately, this encryption process is done automatically by the software and is invisible to the user.

For users, Secure Shell is very simple: just run the software on the computer from which you are connecting. If you have a Point to Point Protocol (PPP) or a Serial Line Internet Protocol (SLIP) account with an Internet service provider, run the version of Secure Shell for your home computer in the place of rlogin, rcp, rsh, or telnet. If you have a Unix shell account, run the shell on the remote system that you dial into.

Secure Shell can be used even if you're not sure the system you're logging into will support the security features, because if that's the case the software will send a warning and drop back to the old "r" command version without encryption. (Secure Shell also supports secure TCP port forwarding and X11 sessions, allowing users to run secure X Window sessions over the Internet.)

Benefits of Secure Shell

For users, the benefits of using SSH are mainly in the encryption of packets sent across the Internet. Compression algorithms included in Secure Shell's data transfer help compensate for the lag caused by encryption. After a brief start-up lag, users may notice slightly faster performance over the "r" commands, thanks to this compression. But the most important benefit is that by using Secure Shell, users can log into a remote system without fear of compromising their passwords, and can transmit data without fear of network "sniffers" grabbing information.

Network administrators can get additional benefits from Secure Shell, not just because it keeps their systems secure and safe from packet sniffers, but because it offers accountability and login tracking. Unlike the "r" commands, which do not keep track of where a user originally accessed the system, Secure Shell can be used to track movements between systems and to access privileged accounts, such as root.

In addition, the system can be configured so that each user allowed to use privileged accounts has a personal key that is exchanged when the account must be used, rather than using a single password shared among multiple users. In this scenario, users feed the key a passphrase (a multi-word password) on login, and when they use a privileged account, the key unlocks that account and records who is using it.

Call For 'Widespread Use'

Some NAS users are already taking advantage of SSH to secure their data, and the security team strongly encourages more widespread use. This is because every time someone logs into NAS systems over the Internet without using Secure Shell, there is the potential for a breach in security. Packet sniffers can grab passwords as they are being transmitted, giving anybody access to your account and your data.

More information about Secure Shell -- how it works, how to use it, and free Unix source code -- is available on the [SSH home page](#). Other sources include: "[Answers to Frequently Asked Questions about Secure Shell](#)" and "[Getting Started With SSH](#)". To download trial versions of commercial client software for Windows NT, Windows 95, and MacOS, go to the [Data Fellows web site](#)).

For more information on using Secure Shell on NAS Systems, send email to the [security team](#).



**In this issue...**[OVERFLOW Speeds SGI
Origin2000 Performance](#)[Secure Shell Keeps
Accounts Safe](#)[Group IDs and Account
IDs](#)[Web Pages Show Lunar
Prospector Satellite Data](#)[PBS Offers 24-hour
Support](#)[New Tool for Interactive
Analysis of CFD Data](#)[Network Analysis and
Design Cookbook](#)

High-speed Processor Techniques: Group IDs and Account IDs

In October, when the 1998 new operational period began, the NAS Facility user consultants were inundated with calls from new users -- and some veteran users -- who needed better information about job accounting and associated commands. In particular, users didn't understand the differences between account identification numbers and group identification numbers, known locally as "acids" and "gids."

The confusion over these numbers is understandable. For one thing, the two look ridiculously similar. Additionally, all acid numbers have a corresponding gid number -- but the reverse doesn't always hold true, because there are circumstances under which special gids are set up to handle multiple groups.

Giddy Over Gids

The gid designates permission levels on files and directories. These numbers are assigned to a group (or groups) of people working on different projects that share files, allowing access to those files to everyone in the group.

When a new file is created, it will automatically inherit the permissions set for the directory it was created in. For example, the following command:

```
chgrp [newgroup] data.dir
```

sets ownership for all new files in the data.dir directory to newgroup. However, the group permissions of *existing* files in data.dir will not change -- those files must be changed explicitly.

Within This Article...[Giddy Over Gids](#)[Acid Indigestion](#)

Acid Indigestion

The acid is used strictly for accounting purposes. It designates which project should be charged for the amount of CPU time used during your job run. The confusion between gids and acids may come about simply because of the term "project" -- which people may think of as being interchangeable with "group."

When you execute a job, your acid is charged for both user time (the CPU time used by a job) and system time (any time that a job incurs on the system, such as I/O.) These charges are made in System Billing Units, or SBUs. For more details on job accounting, see the man pages for 'acct_ytd'.

Each user has a default acid, which automatically initializes when you log in. You can permanently change your default acid once you've logged in, by using the following command:

```
defacct [acid-name]
```

where acid-name is the name of any other acid to which you belong. This command changes the acid charged for future login sessions, but *not* the current session. (See the man pages for "qsub-A" to temporarily changing your acid.)

For further information, call the NAS user services staff at (650) 604-4444 or (1-800) 331-8737, or send email to nashelp@nas.nasa.gov.



**In this issue...**

[OVERFLOW Speeds
SGI Origin2000
Performance](#)

[Secure Shell Keeps
Accounts Safe](#)

[Group IDs and Account
IDs](#)

[Web Pages Show Lunar
Prospector Satellite Data](#)

[PBS Offers 24-hour
Support](#)

[New Tool for Interactive
Analysis of CFD Data](#)

[Network Analysis and
Design Cookbook](#)

Web Pages Offer Unique Look at Lunar Prospector Satellite Data

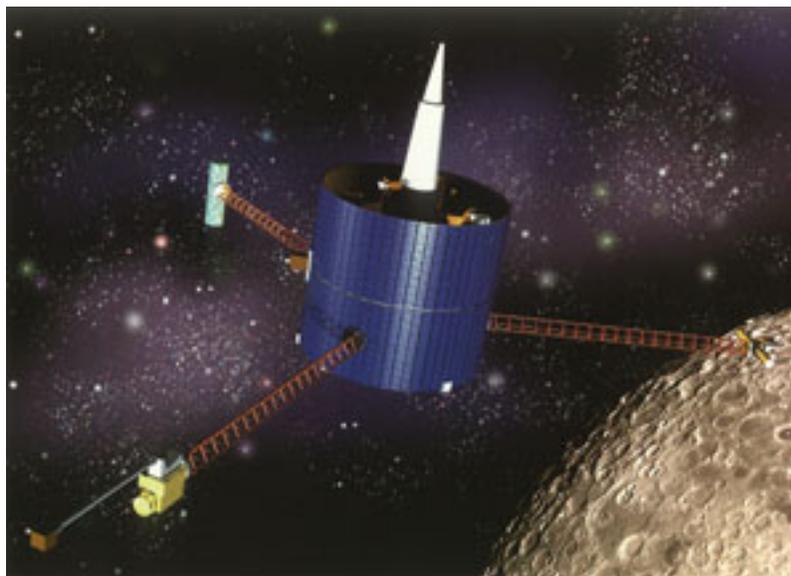
by [Ayse Sercan](#)

The [Lunar Prospector](#), an Ames Research Center-sponsored satellite (due to launch on January 5) from Cape Canaveral, will orbit the Moon to collect data for research into the planet's origin and history, as well as suitability for a lunar space station. The first lunar research mission in 25 years, Prospector is designed to provide the first global maps of the Moon's elemental surface composition such as various minerals and water concentrations and its gravitational and magnetic fields. The data collected by Prospector will be transmitted via telemetry back to Earth and collected at Ames for analysis.

Within This Article...

[Visualizing 'Chunks'
of Data](#)

[Photo Opportunity --
the Next Best Thing](#)



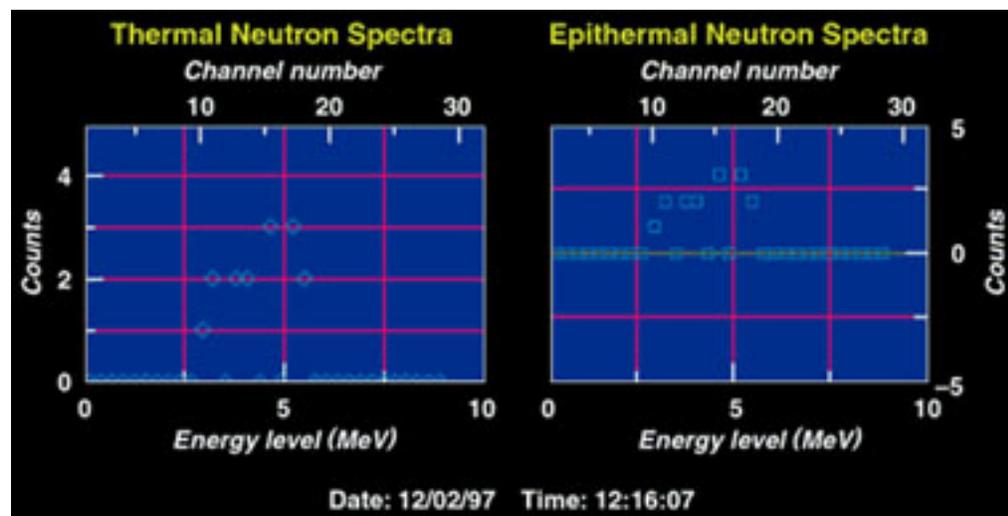
With the help of Glenn Deardorff in the NAS Systems Division's [data](#)

[analysis group](#), Ames will put the data "live" on the Internet in a series of Java-enhanced web pages for a general audience. The public will be able to view data from Prospector as it comes into Ames' ground control center (with about a one- to two-second delay for transmission, and an occasional 88-minute gap while the satellite is behind the Moon and cannot transmit data). Viewers can also check parameters of the satellite itself.

The data is relayed from the satellite which has no onboard computer or storage to [Jet Propulsion Laboratory](#) in Pasadena. From there, the data goes through a direct line to Unix workstations at Ames ground control (operated by the [Space Projects Division](#)), where it is copied onto a PC running LabView software, "massaged" into a slightly more readable form, then transferred to an NFS server where it can be used by the Java applets on the web site.

Challenge in Visualizing 'Chunks' of Data

According to Deardorff, one of the challenges in putting together these web pages was finding a way to show the data in real time in any meaningful way. "The data is coming hopefully every 32 seconds as a bunch of numbers. We had to figure out some way of representing them graphically, even though those numbers haven't been processed for meaningful information," he explained. Because the data coming in at every timestep consists of many "chunks," a simple two-dimensional chart (time versus data value) wouldn't work. One solution was to create plotted charts with bars that change over time (neither axis represents time). Deardorff created a Java applet that automatically updates the charts.



There are no current provisions for a web archive of the Prospector data. "This is only meant to be an eye onto the data stream," explained Deardorff. In addition to showing the data from the instruments, the web pages will also give the status and vital signs of the spacecraft, such as the sweep rate of the electron reflectometer or other instrument parameters, on Java-based charts that are updated automatically as Ames ground control changes settings and makes adjustments.

Photo Opportunity -- the Next Best Thing

Finally, Deardorff added an interesting touch to the pages -- a real-time depiction of where the craft is in relation to the moon, using photos of the Moon's surface taken by the 1994 Clementine mission, a joint NASA-Department of Defense lunar surveyor (for more photos, see the [Clementine web site](#)). A picture of the satellite is superimposed on the photos, which appear to move under the craft. "There's no actual camera on the craft," Deardorff said, "but this approach is the next best thing, because it provides a real-time view of where the craft is in relation to the moon at any given moment."

The satellite, scheduled to launch January 5, will take about three to four days to reach the Moon. The web pages will begin showing data on about January 10. The data visualization pages are only a small part of the [Lunar Prospector web site](#). The main site, a joint effort among several divisions at Ames, was created under the direction of Ken Bollinger, Space Sciences Division. It offers a host of information about the mission, including streamed audio and video interviews with key personnel.



**In this issue...**[OVERFLOW Speeds
SGI Origin2000
Performance](#)[Secure Shell Keeps
Accounts Safe](#)[Group IDs and Account
IDs](#)[Web Pages Show Lunar
Prospector Satellite Data](#)[PBS Offers 24-hour
Support](#)[New Tool for Interactive
Analysis of CFD Data](#)[Network Analysis and
Design Cookbook](#)

Enhanced PBS Offers 24-hour Support

by [James P. Jones](#)

Over the past two years, more than 100 sites have requested evaluation copies of the [Portable Batch System](#) (PBS), the resource management and batch queuing software system developed and used at the NAS Facility. More than a third of these sites now run PBS in production. For those sites that declined to use it, the reason most often given was the lack of commercial support for the package. This situation changed in late November, when the [NASA Ames Commercial Technology Office](#) authorized MRJ Technology Solutions (a research and development contractor at the NAS Facility) to provide distribution, customization, and 24-hour support services for PBS.

This decision resulted in part from the growth of the PBS user community -- which has doubled over the last year -- as well as the popularity of the product and requests for commercial support. According to Dave Tweten, NAS high performance processing group lead, "NAS chose to implement PBS because no other job management system met our requirements -- or those of many other NASA sites." He added: "I'm pleased that MRJ has chosen to distribute PBS because it's good to see it being distributed more widely than could be done by NASA."

PBS Enhanced to Accommodate Metacenter

Two years ago, PBS was selected as the central software component of the NASA Metacenter (a joint project between Ames and Langley

Within This Article...[Accommodations for
Metacenter](#)[PBS in Production at
Large Sites](#)[Future Enhancements
Planned](#)

Research Centers; see [NAS News, July-August '97](#). This required additional features to be implemented in the software, such as the ability to customize the 'qstat' display, which shows all users' jobs.

Another new feature is the ability to prefetch files prior to running a job. Originally, PBS allowed users to specify which files to stage into and/or out of the supercomputer. However, this file fetching occurred *after* PBS selected a job to run -- which meant that nodes were idle while waiting for the file transfer to complete. Now, the Metacenter technology has the ability to overlap file fetching with computation on the nodes. When the file transfer completes, PBS allocates nodes and runs the job -- making more efficient use of Metacenter resources. These changes are now part of the standard PBS distribution.

Word-of-mouth and presentations about the [Metacenter](#) project have led numerous sites that are embarking on similar efforts to download and evaluate PBS.

PBS in Production at Large Sites

In addition to NASA sites -- currently Ames, Langley, and Lewis Research Centers, and Goddard Space Flight Center -- PBS is installed at several other large computing facilities. Pittsburgh Supercomputing Center, as well as the U.S. Army Corps of Engineers Waterways Experiment Station (Vicksburg, MS) and the Aeronautical Systems Center (ASC) at Wright Patterson Air Force Base, all run PBS in production.

ASC -- which houses several supercomputers, including a 256-node IBM SP2 and a 16-CPU CRAY C90 -- is now the largest PBS site, surpassing the NAS Facility in the amount of computing power managed by PBS. (PBS is the sole queuing system used on all supercomputers and support processors at both NAS and ACS.) Wade McClean, integration director for Nichols Research Corp., a prime contractor for ASC, said: "[We] replaced three other job-control software packages by installing PBS at ASC...capitalizing on its full functionality, consistent interface, and ability to be easily tailored to the ASC heterogeneous environment."

Several universities also run PBS, including Purdue University, Jackson State University, and the University of New Hampshire.

Future Enhancements Planned

At the SC97 conference held in San Jose last November, the PBS birds-of-a-feather session was well attended -- with twice the number of installation sites represented compared to the previous year. Discussion focused on future enhancements, the availability of commercial support and its impact on the current PBS user base.

Planned feature changes include: porting PBS to additional Unix platforms; providing scheduler support for jobs spanning multiple supercomputers; and dynamic resource allocation and management.



**In this issue...**

[OVERFLOW Speeds
SGI Origin2000
Performance](#)

[Secure Shell Keeps
Accounts Safe](#)

[Group IDs and Account
IDs](#)

[Web Pages Show Lunar
Prospector Satellite Data](#)

[PBS Offers 24-hour
Support](#)

[New Tool for Interactive
Analysis of CFD Data](#)

[Network Analysis and
Design Cookbook](#)

New Software Tool Improves Interactive Analysis of CFD Numerical Flows

by [Han-Wei Shen](#)

Researchers in the NAS Systems Division's [data analysis group](#) are putting the finishing touches on a new software tool that provides novel techniques to facilitate global analysis of steady and unsteady CFD data. G-LIC (Graphical Line Integral Convolution) incorporates a unique flow visualization technique, called UFLIC ([Unsteady Flow Line Integral Convolution](#)), which converts a sequence of time-varying flow solutions into images and animations.

The images capture complicated flow behavior and help scientists detect important flow features. G-LIC contains a flexible graphical user interface (GUI) and an efficient multi-threaded software environment in which users can interactively specify regions of interest and effectively "steer" visualizations with minimal effort.

Within This Article...

[Improved Unsteady
Flow Visualization](#)

[Flexible Graphical
User Interface](#)

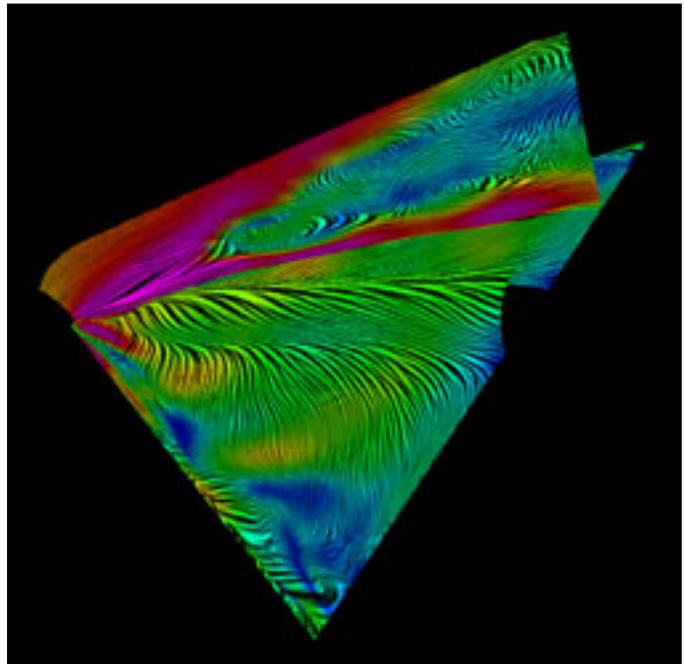
[Multi-threaded
Software Structure](#)

[Beta Release
Coming Soon](#)

Improved Unsteady Flow Visualization

Visualizing flow field data is challenging due to difficulties in finding suitable graphical icons to represent and display vectors. Conventional flow visualization techniques, such as plotting arrows or tracking particles, often clutter the display window when too many arrows or particles are placed, or miss important flow features when too few icons are present. With G-LIC, extremely dense flow fields can be visualized globally without these problems.

Flow patterns generated on the wing body and grid surface of a delta wing model generated by G-LIC (Graphical Line Integral Convolution), a software package developed in the NAS Systems Division. The image reveals important flow features, such as flow traces (black lines) and vortices (swirls at top right in blue area). The surfaces are colored by the velocity magnitude, ranging from blue (low) to red (high). Delta wing dataset by Neal Chaderjian, Applied Computational Aerodynamics Branch, Ames Research Center; graphic by Han-Wei Shen.



G-LIC has evolved from a series of techniques, starting in 1993 with Line Integral Convolution (LIC), developed by Brian Cabral and Leith Leedom, at Silicon Graphics Inc. (SGI). LIC produces continuous flow textures that resemble the surface oil patterns produced in wind-tunnel experiments.

While LIC is an effective method, it is primarily used for visualizing steady flow field data. For unsteady flow simulations that produce time-varying solution files, scientists can only apply LIC to each individual timestep of data and then put the frames together to create an animation. However, the animated results only represent snapshots of the flow at discrete timesteps, revealing no information on how the flow evolves from one time step to the next. In addition, the flow patterns shown in

the LIC images represent instantaneous streamlines that do not reflect what actually happens in unsteady flows.

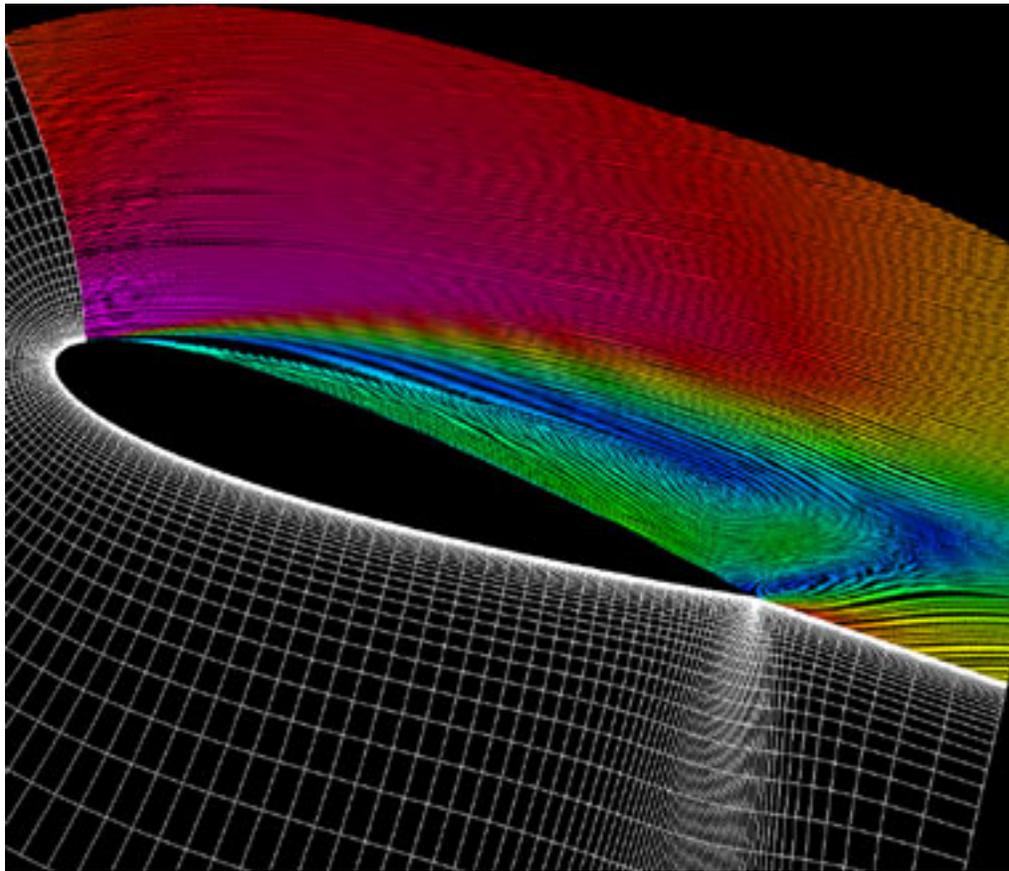
Further studies of the LIC method by NAS researchers David Kao and Arthur Okada resulted in a software tool called AFLIC (Animated Line Integral Convolution; see *NAS News*, [May-June 1997](#)), which added important capabilities, such as detecting flow separations and reattachments.

Next, NAS researchers developed the UFLIC algorithm to solve the problem of visualizing vector data in unsteady flow fields. UFLIC simulates the advection (particle movement) of a two-dimensional image with a noise texture pattern on a grid surface. The results generated from this algorithm produce animation sequences of the flow motion that clearly show unsteady flow patterns and highlight flow evolution. A paper on UFLIC presented at the IEEE Visualization '97 conference last October (see *NAS News*, [September-October 1997](#)) generated interest among scientific users.

The new G-LIC software implements the UFLIC algorithm and inherits most of the functionality from AFLIC for steady flow analysis. In addition, G-LIC provides a flexible and highly integrated working environment for CFD scientists.

Flexible Graphical User Interface

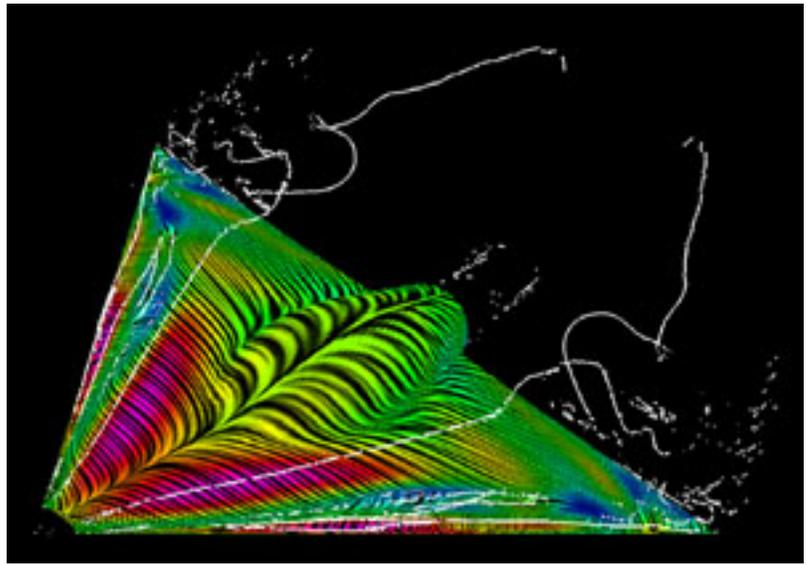
G-LIC's built-in graphical user interface, written in the scripting language Tcl/TK (developed by John K. Ousterhout, University of California, Berkeley) allows a scientist to explore the model geometry before starting the texture computation. Currently, G-LIC accepts PLOT3D multi-zone steady and moving curvilinear gridded data. The scientist can interactively sweep through grid planes in any dimension, visualize the numerical grid structures, and then select the surfaces where G-LIC will "paint" the flow textures.



G-LIC (Graphical Line Integral Convolution), an interactive software tool, creates realistic flow textures from large-scale steady and unsteady flow simulations. G-LIC allows scientists to interactively select regions of interest and specify different texture resolutions in various areas of the model. In this way, they can steer most of the computing power to regions that contain important flow features and view a fast approximation of results in relatively uninteresting regions. This image shows a two-dimensional flow pattern and its grid geometry over an oscillating airfoil (the airflow surrounding the cross-section of a wing). Data by Shigeru Obayashi, formerly of the NAS Systems Division; graphic by Han-Wei Shen.

G-LIC has a graphical viewer (written using SGI's OpenGL graphics library) which displays the resulting images and allows scientists to display and manipulate selected grid geometry in three-dimensional space. In addition, the graphical viewer can display other important flow features, such as vortex cores or particle traces, which are extracted by UFAT (the [Unsteady Flow Analysis Toolkit](#)) to create a more comprehensive visualization.

G-LIC has the ability to combine both flow textures and geometric objects (produced by UFAT, the Unsteady Flow Analysis Toolkit) in a single visualization. This image shows a textured delta wing with vortex cores (white streaks). Vortex core data by David Kenwright, NAS data analysis group; graphic by Han-Wei Shen.



The algorithms that G-LIC invokes provide a global view of the flow field over an entire grid surface. However, these algorithms are computationally intensive and therefore cannot be used interactively. To expedite the analysis process, the scientist can subsample the flow field by specifying the types of texture resolutions for the computation. This degree of control over the texture resolution is useful when the scientist wants to preview the results at a low resolution before starting a more accurate but much slower computation.

Multi-threaded Software Structure

An important aspect of G-LIC is that it is a multi-threaded program; that is, the whole program consists of several processes running independently. In G-LIC, the tasks of handling user events and computing the line integral convolution are executed in different threads. In this way, scientists can continuously interact with the program. For example, while the UFLIC algorithm is computing, they can still select menu items or rotate the model geometry, check the visualization results, pause the computation, reset the program parameters, or even terminate the visualization process at any time.

Another useful feature of G-LIC is that scientists can switch between interactive and batch execution modes. When analyzing a long sequence of unsteady flow data, the scientist may decide that the graphical user interface is no longer needed. In this case, he or she can terminate the GUI, let the program execute in the background, and save the results into files for later investigation.

Beta Release Coming Soon

G-LIC is currently implemented on SGI graphics workstations. A beta version is planned for release before the end of the first quarter. For more information on G-LIC and its predecessors, contact [Han-Wei Shen](#) or [David Kao](#).

G-LIC was developed by Han-Wei Shen, David Kao, Aleksandra Kuswik, and Ling-Jen Chiang, all in the NAS data analysis group. UFLIC was developed jointly by Shen and Kao.





In this issue...

[OVERFLOW Speeds
SGI Origin2000
Performance](#)

[Secure Shell Keeps
Accounts Safe](#)

[Group IDs and Account
IDs](#)

[Web Pages Show Lunar
Prospector Satellite Data](#)

[PBS Offers 24-hour
Support](#)

[New Tool for Interactive
Analysis of CFD Data](#)

Network Analysis and
Design Cookbook

Network Analysis and Design 'Cookbook'

by [Ayse Sercan](#)

Practical Computer Network Analysis and Design, by James D. McCabe of the NAS Systems Division, was released to bookstores last November. Where other networking books discuss theory or specific operating systems and protocols, none offer McCabe's systematic approach to network design. In his foreword to the book, David M. Piscatello, president of Core Competence Inc., a Pennsylvania-based networking consultancy, writes: "there are few places to go where you will find a statement of the nature, 'if you observe symptom X and you apply this, then you should observe Y or Z' when you attempt to design or redesign a network." This book is just that: a sort of cookbook for network designers.

Within This Article...

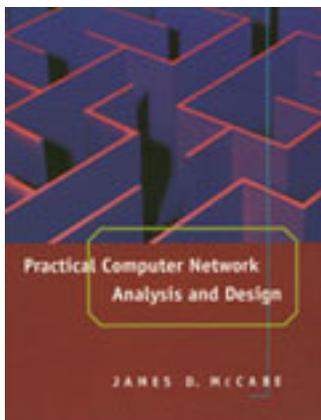
[Determining Network
Requirements](#)

[Go With the Flow
Analysis](#)

[It's On The Net](#)

The book gives a detailed outline for the design and analysis of networks, a process with which McCabe as chief technical officer of FSC End2End Inc., a subcontractor at the NAS Systems Division is very familiar. At the NAS Facility, McCabe's focus is on quality of service in high-performance networks. In fact, the projects he's worked on at Ames since 1988 form the foundation of his book.

Practical Computer Network Analysis and Design is designed to be used on many different levels from a simple overview of the network design process to a detailed methodology for determining network needs and building a network. Two sections of particular interest discuss requirements analysis and flow analysis. Both sections, spanning three chapters each, are based largely on McCabe's work at the NAS Facility.



Determining Network Requirements

The book tackles what McCabe sees as one of the greatest challenges in building a network: meeting requirements. Network performance problems have traditionally been solved by throwing money into the system: a new server, faster switches, more bandwidth. But this approach can get expensive. "Without understanding the design environment, relatively simple solutions ... are likely to be ineffective or too costly to be practical," McCabe writes. The book focuses on teaching network designers to make "better design decisions" and to deal with a network as a system rather than as a "bunch of separate services." The first part of this process is determining what the network requirements are.

According to McCabe, network requirements include: user requirements, such as timeliness, reliability, and security; application requirements; and host requirements, such as location, and hardware requirements. Networks also require money, and the book recommends that the level of funding be determined early on because it is "usually a constraint to the design ... it is important to know what this level is as early in the analysis process as possible, in order to avoid creating a design that is not economically feasible."

When all requirements have been determined, you'll be able to apply McCabe's methodology for handling them, including such details as determining network propagation delay in comparison to human response time and interaction delay. He discusses the development of performance thresholds and consolidation of services, then offers a practical chapter with worksheets and case studies as examples.

Go With the Flow Analysis

Another important aspect of understanding how a network operates is flow analysis -- knowing how data is moving around in the system -- which McCabe discusses at length. This information is useful in planning network capacity and level of service.

As with requirements analysis, the first step in flow analysis is information gathering. The book details the process of finding data sources and data sinks (the devices that produce data and the devices that accept data from the network). It also covers flow models, such as peer-to-peer or client-server flows, and flow boundaries, such as the

divider between a wide area network and a local area network.

As in the requirements analysis section, the book includes a practical chapter full of examples of flow specifications, as well as worksheets for understanding network flow.

It's On The Net

Practical Computer Network Analysis and Design can be purchased online from [Morgan Kaufman Publishers](#). Supplemental worksheets and templates for creating network designs can also be copied from their website.

McCabe will be on hand for a book-signing session at Spring [Network+Interop](#) in Las Vegas, May 4-8.

Practical Computer Network Analysis and Design

James D. McCabe

Morgan Kaufmann Publishers , San Francisco

400 pages

\$54.95

ISBN 1-55860-498-7





In this issue...

[OVERFLOW Speeds SGI
Origin200 Performance](#)

[Secure Shell Keeps
Accounts Safe](#)

[Group IDs and Account
IDs](#)

[Web Pages Show Lunar
Prospector Satellite Data](#)

[PBS Offers 24-hour
Support](#)

[New Tool for Interactive
Analysis of CFD Data](#)

[Network Analysis and
Design Cookbook](#)



Credits

Executive Editor: Thomas Lasinski

Editor: Jill Dunbar

Staff Writer: Ayse Sercan

Contributing Writers: James P. Jones, Han-Wei Shen, Jim Taft

Image Coordinator: Joel Antipuesto

Online Page Layout and Graphics: Joel Antipuesto, Chris Gong, Eunah Choi, Rosemary Wadden

Other Contributors: Richard Anderson, Ken Bollinger, Cristy Brickell, Chris Buchanan, Nick Cardo, Justin Collins, Archie Deguzman, Glenn Deardorff, Daniel DePauk, James Donald, Ed Hook, Mary Hulquist, Bryan Juliano, David Kao, Chris Kleiber, Kevin Lahey, James McCabe, Patrick Moran, Marcia Redmond, Leigh Ann Tanner

Editorial Board: Cristy Brickell, Jill Dunbar, Chris Gong, Thomas Lasinski, Nateri Madavan, Patrick Moran, George Myers, Ayse Sercan, Harry Waddell