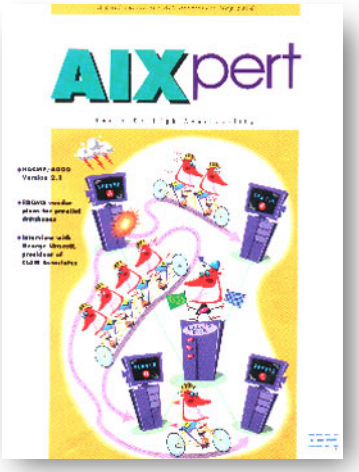
AIXpert DIGITAL VERSION IBM

# TABLE OF CONTENTS





## Commentary

**Available 99.999% of the Time**
By George Noren

## AIXposure

**Interview with George Linscott**

## AIX

**HACMP/6000 Version 2.1 Overview**
By Daniel P. Cox and Frank Lawlor

**Migrating to HACMP/6000 2.1**
By Thomas Casey and Robert Metcalf

**DCE With HACMP/6000**
By Jim Wade

**Making Backups of Mirrored Filesystems on AIX 3.2**
By Jaime Vazquez

**Open Blueprint: IBM's Guide to Distributed Computing**

## Database

**DB2 Parallel Edition**
By Gilles Fecteau

**INFORMIX-OnLine's Dynamic Scalable Architecture**
By Gregg A. Christman

**Oracle Parallel Technology Empowers AIX Systems**
By Sandra Lee and Annie Chen

**SYBASE for HACMP/6000: An Architected Approach to Clustered Systems**
By Josh Bersin

## Communications

**AIX Communication Service: Streams**
By Eddie Ho, Saurabh Desai, Derwin Gavin, and James Partridge

**LANHOP/6000 Version 1.0**
By Wayne G. Wachtstetter and Ralph E. Vosburg

## Q&A

**AIX Questions**
Compiled by Ismat Dhanjibhai


MAY
1994

AIXpert BACK TO MAIN MENU

# Available 99.999% of the Time


In 1992, when HACMP/6000 was still in its planning stages, I was involved in designing and implementing the training for the upcoming product. Working with CLAM Associates and the project leaders in Austin, we assembled a comprehensive four-day training session describing HACMP/6000 and how to set it up. At that time, configuration of HACMP/6000 was done through shell scripts and by editing text files, and it supported only modes 1 and 2 (mode 3 was still coming). What a long way it has come from that beginning! Configuration is done chiefly through SMIT, mode 3 has been operational for some time now, and the new four-way clusters add power and flexibility to the product. So putting together this issue's focus on HACMP/6000 was especially interesting for me.

For readers who haven't been associated with HACMP/6000, we have an overview article to explain the concepts and features. Then we "go to the source" for a talk with George Linscott of CLAM Associates. George has been involved with HACMP/6000 since the beginning. His insight on high availability and clusters for both today and the future is "must" reading. Readers who are concerned only about "what's new" can read about the latest features in Version 2.1 in the article about migrating to the new version.

Since database programs are the main applications that use high-availability environments, we asked the major AIX database providers to write articles from their vantage point. Four such articles give a flavor of what can be done with high availability. To round out the coverage, we have an article about running HACMP/6000 with DCE.

If you don't care about high availability or clusters, perhaps you will be interested in the other articles in this issue. Making backups of mirrored filesystems can be vital to HACMP/6000 administrators, but the rest of us can also find good information in that article. Programmers will be interested in the article on using the Streams interface to produce modular, portable networking programs. And those who travel a lot will find the LANHOP/6000 article intriguing.

## The World's a Little Greener

This issue marks the first issue of *AIXpert* to be printed solely on recycled paper with soy-based inks. We think this makes the magazine a little easier to read, knowing that it was produced with as little impact to the environment as possible.

## RISC System/6000 Electronic Information

Coming soon, RISC System/6000 information will be available through the Internet. You will be able to connect to a large collection of information and services via the World Wide Web using the National Center for Supercomputing Applications (NCSA) Mosaic or other available browser program. Check it out using the Uniform Resource Locator (URL) of http://www.austin.ibm.com. Features offered include product announcements, press releases, technology white papers, files for anonymous ftp, AIX news collected by the editorial staff of *AIXpert*, and some *AIXpert* articles in PostScript® format.

For more information, send E-mail to webmaster@austin.ibm.com when the system becomes available.

George Noren

---

**George Noren**, IBM Corporation, Internal Zip 2830, 11400 Burnet Road, Austin, TX 78758. Internet: geo@austin.ibm.com. Since joining IBM in September 1979, Mr. Noren has written manuals for System/34, System/36, and AIX on both the RT® and RISC System/6000® platforms, and was a member of the InfoExplorer™ design team. He has also worked as system administrator for several AIX server machines and their clients, and is currently responsible for the Prototype Evaluation Labs in Austin. Mr. Noren studied engineering at Illinois Institute of Technology, holds a BA in English from the University of Minnesota, and an MBA from St. Edwards University in Austin.

**George Noren**

# Interview with George Linscott

AIXPOSURE

George Linscott is president of CLAM Associates, the IBM Business Partner that assists IBM in creating High Availability Cluster Multiprocessing/6000 (HACMP/6000). CLAM began as a contract programming shop for IBM, completing AIX® networking, graphics systems, and X-Windows projects. Now that CLAM is intimately involved with HACMP/6000, it offers a full range of services for organizations planning an HACMP/6000 installation.

**You are one of the founders of CLAM. How did the company get started?**

**Linscott**: My background is in programming. I worked for IBM for five years in the Academic Information Systems group. During that time, I assisted with MIT's® Project Athena. The goal of Project Athena was to roll out 1,000 interconnected workstations from several vendors to undergraduates—so students could walk up to any UNIX® workstation and write a paper. This was the first time workstations from several vendors had been installed to cooperate with one another.

Many people from business came to see what we were doing. That's when I developed an interest in applying UNIX workstations to commercial applications (they were used almost exclusively in scientific and technical areas in the 1985–1986 time frame).

**When did you start CLAM?**

**Linscott**: We formed CLAM in 1987 as a software and services company. Three of us came from IBM: Les Comeau was my manager at IBM and Arnie Miller worked with me on the UNIX development team.

We did a lot of contract programming, such as porting NFS® to AIX PS/2®. The company started to move along—we had two or three employees! In 1990, we began working with HACMP/6000. Today, 85 people work here.

**Where did you come up with the name?**

**Linscott**: CLAM represents the first letters of the last names of the founders. We called the company CLAM Associates because we visualized getting all the really good people we knew together in one place—so we could do really great things.

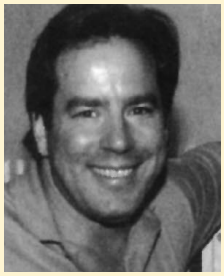
**How about a brief description of HACMP/6000?**

**Linscott**: There are two parts to HACMP. The HA part provides high availability—with two or more servers, each working with different databases. If one crashes, the others can take over the disks and restart the database. Companies that are running their businesses on computers should have HA so that downtime is minimal.

The CMP (cluster multiprocessing) part comes in when you have two or more servers concurrently accessing the same disk subsystem. We have done benchmarks against NCR® and HP™ that can outperform a Symmetric Multiprocessor (SMP)—depending on the workload—for very complex transactions like you would see in TPC-C, where the locking is minimal compared to the overall transaction. The CMP part provides extremely cost-effective scalable computing for large businesses.

**So database design is really critical.**

**Linscott**: Exactly. Even the best system can be "brought to its knees" by a poor database design. Designing a database system in this type of environment takes considerable thought. You can have four nodes in an HACMP cluster today, but if your design has a hot spot in one table, performance is significantly degraded. Database design-



George Linscott

ers prefer HACMP because they don't have the nightmare of partitioning (having more than one database). But they have to avoid hot spots. CLAM does a large amount of benchmarking to help customers get the best performance from HACMP.

You can really pump transactions through an HACMP cluster. Imagine up to four RISC System/6000s—each with hundreds of megabytes of memory—concurrently accessing a disk. Each has its own database cache and a separate connection to the network.

**Could you tell us the history of the HACMP product?**

**Linscott**: The original architects were Lynn and Ann Wheeler, a husband-and-wife team. The need was created when IBM made a commitment to a New York newspaper to create a high-availability system for their publishing needs. After a couple false starts in Austin, IBM asked the Wheelers—who had experience with high availability on MVS™ and were IBM employees in Austin—to do the job. The Wheelers said they would, if they could relocate to a beautiful IBM lab in Los Gatos, California and hire anyone they needed to help with the effort. Ultimately, management agreed. They contracted with CLAM (which happened to be me at the time) to help with the project.

**Database designers prefer HACMP because they don't have the nightmare of partitioning.**

When the Wheelers asked IBM customers if they would buy a UNIX-based high-availability solution if it existed, they received an overwhelming response. Based on that, IBM decided to make HACMP available to everyone.

**It sounds like HACMP really kick-started CLAM Associates.**

**Linscott**: We grew by meeting our commitments to IBM. We demonstrated competence; then IBM asked us to do more and more. When the product was released, we began to help with installations too. Since no one was doing training, we started conducting classes. We found that customers in a high-availability environment need special support; so we started doing 7-days-a-week, 24-hours-a-day (7x24) support for them.

Our main business today is services. Assisting with HACMP/6000 is our foot-in-the-door with many customers. Once we are there and demonstrate our competence, we may be asked to help with client/server design and implementation. We are also developing some of our own products now.

**Tell us more about your systems integration and education business.**

**Linscott**: We have discovered that to provide high availability, we need to know a lot about AIX as well as the network and clients. We take that expertise and help users design high-availability systems; then we perform the classic role of systems integration. Most customers are moving down from mainframes. We have a large project with Citicorp® where CLAM designed and developed a client/server application working with Citicorp and their IBM team.

**CLAM is supporting the HACMP/6000 sales process. Do you also sell HACMP?**

**Linscott**: Yes, we just opened our Commercial Systems Center in Austin. We will showcase AIX solutions along with IBM. As a Business Partner, we market products in Austin as well as from our Cambridge office.

**What are the issues surrounding HACMP for developers?**

**Linscott**: High availability is not a shrink-wrapped solution. You don't just buy a set of software, install it on a machine, and forget it. You need to do the same amount of planning as you would for a mainframe-based solution. High availability is a design point in which commercial systems need to be well thought out—from the server to the clients. It is really important to understand your entire operating environment. Many installed systems grew over time with very little planning, making it difficult to get the total picture. You would be amazed at the number of people we talk to who do not understand all the parts of their system. If you really want a high-availability solution, you have to understand how all the parts fit together.

You also need to understand the available technology alternatives, such as network topology and the capabilities of different communications protocols. For example, TCP/IP is more highly available than User Datagram Protocol (UDP).

**How does the cost of supporting distributed systems compare with the same application on a mainframe?**

**Linscott**: I believe that the maintenance cost of the systems is at least equivalent to the cost of maintaining similar mainframe-based solutions.

Time after time we see people frustrated; they have underestimated the effort needed to maintain some high-availability solutions. Keeping hundreds of client machines up and running is a lot of work.

### Isn't there some problem with high-availability systems losing transactions if a switchover occurs?

**Linscott**: As with any OLTP system, transactions that are in flight when a crash occurs may be lost. Once HACMP has recovered from the crash and reconfigured the cluster, the database can perform log processing to ensure data integrity. Although transaction monitors could play a role here, we have yet to see widespread use in commercial installations. In a client/server environment, the client can have a role in providing transaction fault tolerance.

CLAM has a client software library called `clinfo` that provides the application with the state of the cluster. This library uses Simple Network Management Protocol (SNMP) to communicate between the client and the server. It keeps a current state of the connections and machines in the cluster. If the server crashes as the client is running transactions to that server, the application can redirect the transactions to the survivors. If the client application doesn't know which transactions have been committed yet, a restart is required.

You can build a fault-tolerant transaction-processing system by having the ability to restart transactions at the client. We are currently building some to show that it is a viable solution.

We are now building load monitoring into `clinfo`. Applications can perform load balancing, which directs transactions based on the load on the server. This software is available for AIX and Microsoft® Windows™.

### How can readers learn more about this?

**Linscott**: Just call CLAM at (617) 621-2542 and ask for John Ranta. We have white papers describing how you can build a fault-tolerant transaction-processing system.

### Could you give us an overview of HACMP support by the database vendors?

**Linscott**: This can be answered at several levels. First, any database that can recover from a hard shutdown and maintain data integrity is an instant candidate for high availability. From the standpoint of the database, an HA fallover looks like a very fast reboot. This means that all the major databases are suitable for the HA environment. Next, you want to examine the architectures available. There are two "religions" in the database world. The architectures of DB2/6000™, Sybase®, Informix®, and Progress® are based on a shared-nothing philosophy. Each machine owns its disk. Any machine that wants access to the tables on another's disk must go through the network. Oracle® and Mumps are based on a shared-disk philosophy. Also, Ingres® was the first to explore concurrent processing on a UNIX platform. CLAM ported the Ingres database to work in the CMP mode of HACMP. Today, Ingres does not support this product.

HACMP adds value even in a shared-nothing world. Given a replicated architecture, making a node highly available is still a winner in terms of minimizing downtime and masking minor failures. In a partitioned scheme, high availability is essential for each partition. In a shared-disk architecture, HA provides the mechanism for maintaining multiple highly available database servers and adds the requisite disk-sharing subsystem.

**You can build a fault-tolerant transaction-processing system by having the ability to restart transactions at the client.**

### Are you seeing fault-tolerant customers migrating to HACMP?

**Linscott**: For many years fault tolerance was the only approach available to customers who needed high availability. Today, if you ask many fault-tolerant customers how long the server is up, they say 99.999% of the time. But then ask them how often clients or users on terminals experience an outage; many say once or twice a week (due to software crashes or network problems). At that point, we start talking about a total system design, which is necessary to provide front-of-screen availability. You see, it really doesn't matter how long that server is up; what really matters is how much of the time the person at the screen can work. HACMP is the most cost-effective way to accomplish that today.

### Have HACMP-like configurations changed the basis for choosing shared-nothing versus shared-disk architectures?

**Linscott**: A shared-nothing strategy can be very scalable in a read-mostly environment. You can have 10, 20, or 40 machines that are all cooperating together because they are shipping around

functions and data. But remember, these architectures take many years to put in place. These decisions were made five or six years ago when the theory about shared disks was that it did not scale well because of the limitation of a 2 MB/second throughput rate from disks. The architects did not consider that disk rates also move forward. So now, we have 10 MB/second SCSI-2 disks and IBM 9333 serial drives that are 32 MB/second. And vendors are talking about fiber-connected disks running at 100 MB/second in the future. With Fiber Channel Standard (FCS), you could have 64 processors hooked into a shared-disk system. Suppose the disk subsystem is a disk array with intelligent caching—it only makes the shared-disk approach appear better. For write-intensive OLTP applications, shared-disk clusters provide better scalability.

**HACMP is nothing more than exploiting all the things that are already in AIX.**

### What's your advice to developers who know AIX, but have little or no experience with HACMP/6000?

**Linscott**: Developers need to do paradigm shifts quite often. HACMP is just another paradigm. Often programmers are afraid to make those paradigm shifts because when you're the best at what you do and move to a new environment, you end up at the bottom of the heap. Remember that HACMP is nothing more than exploiting all the things that are already in AIX. If you know AIX, you know HACMP/6000.

### What competes with HACMP?

**Linscott**: NCR has a product called LifeKeeper™, which some analysts think is a better product. But when we go through a point-by-point comparison with these analysts, they conclude that HACMP is much better. So what's going on? Well, NCR has done a very good job at marketing their product. IBM is changing its marketing plans this year to address that issue.

**If you know AIX, you know HACMP/6000.**

HP has a product called Switchover/UX. I can't say that we have ever lost an account to HP because of Switchover/UX versus HACMP.

### Is there a market for third parties to build off HACMP?

**Linscott**: Absolutely. Companies with products that require high availability are integrating HACMP into their offerings. Industries that we have worked with so far include retail, medical imaging, insurance, telecommunications, transportation, and power management. I expect to see many more companies reselling HACMP in the coming year.

### What's in the future for high-availability systems?

**Linscott**: High availability is currently a thin layer over the operating system. As time goes by, more and more high-availability functions will be integrated into the subsystems of the operating system. There will still be a need for highly available application support above the operating system. HACMP should take advantage of evolving technology. Suppose IBM comes out with a four-way SMP. Customers will realize tremendous scalability by plugging SMP nodes into HACMP clusters.

Future capability that I would like to see is "disaster survivability"—clusters spread out geographically. Machines could be in separate buildings or separate counties. Since natural or man-made disasters can take out a big business for an extended amount of time, there is a great opportunity to protect businesses with geographically dispersed clusters. The internode communications may use Fiber-optic Data Distribution Interface (FDDI) from a telco at 100 MB/second. Bell Atlantic® offers plug-in FDDI in the Washington, DC area now. FCS and Asynchronous Transfer Mode (ATM) are other transports that are suitable for internode communications in clusters. In the next year or two as more telcos offer this, disbursed clusters will become a reality. This is actually a type of replication plus additional high-availability logic.

### Is HACMP ready for prime time, or are there still missing pieces needed to create a complete solution for a customer?

**Linscott**: It's absolutely ready for prime time. If you are running a database today, you can bring in HACMP without any additional development and just a little systems integration. It will give you much higher availability than one processor. Clients will have to reconnect after a switchover, but that's better than being down for hours. IBM has over a thousand customers that are doing very well with it.

# HACMP/6000 Version 2.1 Overview

By Daniel P. Cox and Frank Lawlor

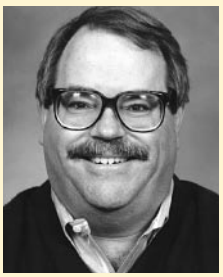
This article provides an overview of Version 2.1 of IBM's High Availability Clustered Multiprocessing/6000 (HACMP/6000) software and describes some new features planned for 1994.

IBM's HACMP/6000 software masks hardware and software failures in clustered RISC System/6000 environments by quickly switching over to backup machines.

HACMP/6000 has two major components: High Availability (HA) and Clustered Multiprocessing (CMP). The *high availability* part of HACMP/6000 is a computing configuration that will survive multiple points of failure. The product detects and recovers from failures of disks, disk adapters, networks, network adapters, and processors. A cluster of loosely coupled machines provides redundancy by transferring control from a failed processor to a backup. A fault-tolerant transfer usually implies the loss of, or interruption to, some in-progress work while the transition takes place. High availability, also referred to as fault-resilient computing, differs in that it is a step toward fault tolerance.

HACMP/6000, which operates on the RISC System/6000, complements and extends the facilities already built into AIX for improved availability, such as the robust Journaled File System and logical volume mirroring.

Generally, HACMP/6000 clusters execute applications without change. Some changes may be necessary for scripts that specify the actions to be taken when a failure occurs and also when components are restored to operation. These scripts are tailored by the system administrator. HACMP/6000 relies on the application, however, to provide any failure recovery transparency or fallover transparency to external users and client machines (restarting the work in-process when a fallover occurs).



Daniel P. Cox

If a system fails, nominal recovery time is approximately 30 to 300 seconds. Actual recovery time depends on the system configuration, application configuration, size of the user databases, and the user's recovery script (if any).

The *cluster multiprocessing* part of HACMP/6000 provides concurrent disk access support to multiple processors in the cluster. This concurrent access provided at the raw disk level requires application support, such as locking, to control access to the shared data. HACMP/6000 provides distributed locking facilities to support this.

By using standard RISC System/6000s, the HACMP architecture provides a lower cost alternative to the expensive and specialized fault-tolerant systems. It is useful when a short application disruption is acceptable.

## High-Availability Applications

HACMP/6000 is designed for database and transaction processing applications. The architecture provides reliable, recoverable shared disk resources for database or Online Transaction Processing (OLTP) servers and ultimately to client applications.

Figure 1 shows several examples of applications that use HACMP/6000.

Businesses that support their customers with continuously running computer operations have a vital need for this type of high availability.

## Scalable Systems Growth Through Clustered Multiprocessing

As data processing requirements grow beyond the capacity of a single processor, HACMP/6000 can extend the value of an installed RISC System/6000. The clustered multiprocessing feature of HACMP/6000, combined with storage systems such as IBM's 9333 High-Performance Disk Sub-

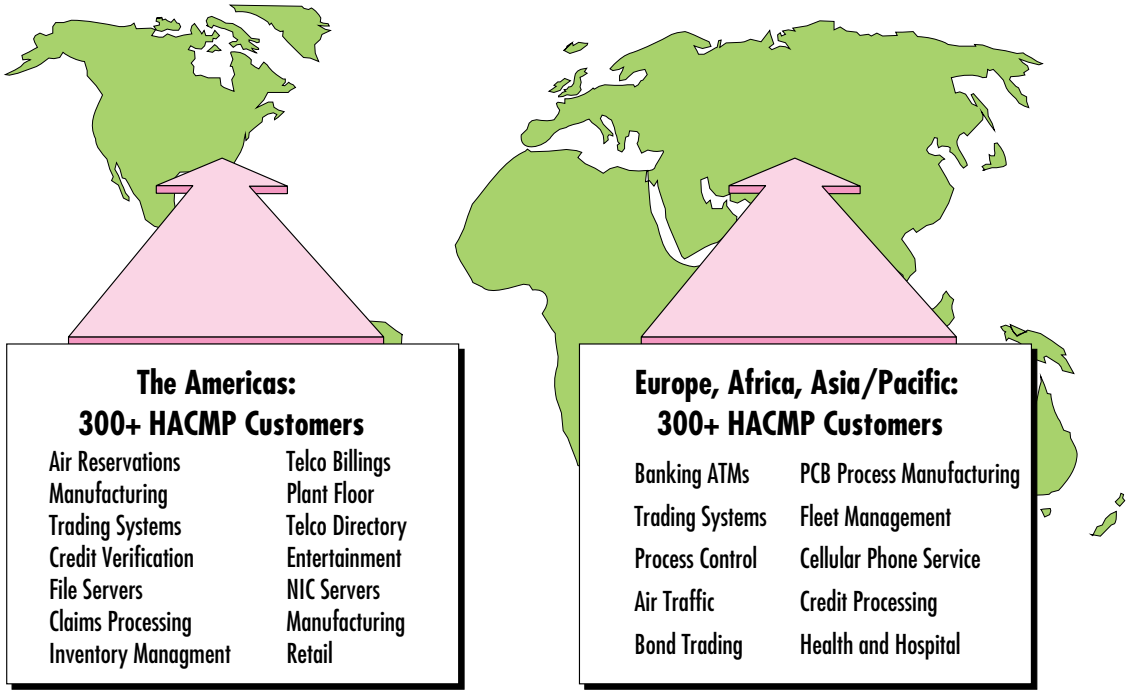



**Figure 1. HACMP/6000—a worldwide product**

system, allows concurrent disk sharing by multiple processors in the cluster.

HACMP/6000 currently supports up to four processors in a cluster. Later this year, it is expected to support eight-way clusters.

The Cluster Lock Manager (CLM) is especially useful for controlling access to concurrent disks. It provides programming interfaces that applications can use to create a single lock image that can be shared among all the nodes in the cluster. Transaction-oriented applications such as databases can benefit from the CLM.

HACMP/6000 has two locking models. The UNIX lock model supports standard UNIX System V region locking. The CLM lock model provides the following functions:

- Six locking modes that increasingly restrict access to a resource
- Asynchronous lock completion
- Global data through lock value blocks

## System Management

New HACMP/6000 installation and configuration facilities allow administrators to install and configure a cluster of RISC System/6000 processors from a single processor. This is easier than individually installing each of the systems in the cluster. HACMP/6000 also includes new installation verification services.

Problem determination capabilities help to trace cluster problems more effectively. More comprehensive, data-driven scripts now minimize the need to modify HACMP/6000 fallover scripts.

## HACMP/6000 Runtime Environment

The cluster manager and related services run on each server and perform the following functions:

- Constantly send a series of heartbeat messages through the networks to determine the status of other servers in the cluster (All nodes share these messages to provide information about server failures. These messages can transmit over a LAN such as Ethernet™, Token Ring, or Fiber-optic Distributed Data Interchange, or a serial link.)
- Maintain "state machine" or topology information showing the status (up or down) of processors in a cluster
- Recognize changes in clustered processor states (up or down)

- Execute fallover scripts specific to the resource that changes state (such as a network interface failure or a processor being restored to service)
- Coordinate system booting and failure-recovery operations across the processors in a cluster
- Provide cluster status to the administrator and interfaces to other AIX and RISC System/6000 applications, such as NetView/6000 (Cluster status is also available using Simple Network Management Protocol (SNMP.)

**HACMP/6000 currently supports up to four processors in a cluster. Later this year, it is expected to support eight-way clusters.**

### Cluster Management Operations

The cluster manager is automatically started at boot time. During startup, each machine in the cluster contacts the others over a network, and together they elect a primary or master cluster manager. From this point, all cluster managers send periodic heartbeat messages back and forth among themselves to ensure that all the cluster members are "alive."

If one cluster manager stops sending these heartbeat messages, the other cluster managers try to contact the silent machine through alternate network interfaces or alternate networks. (All cluster nodes should be equipped with alternate adapters and attached to multiple networks.) If these attempts are unsuccessful, the other processors in the cluster will assume that the machine has failed and drop it from the cluster.

A "deadman" switch in each cluster manager ensures that if contact is lost with the other cluster managers, it will terminate system activity. This prevents a rogue system from corrupting shared data.

When a machine is dropped from the cluster, a recovery shell script is executed in the remaining processors. This allows the cluster to reconfigure itself to deal with the failed processor. Backup machines take control of the failed processor's disks, and one may masquerade as the failed machine on the network (using IP address takeover).

The HACMP/6000 cluster manager makes no assumptions about the actions that should occur when the cluster configuration is modified due to a fallover. The cluster behavior is left to the person implementing the cluster. A system administrator or programmer tailors the fallover scripts that define the system fallover and recovery procedures. These scripts—modifiable shell procedures—can include any appropriate commands.

For example, suppose two HACMP/6000 servers are functioning as database servers to several clients. If one machine fails, the cluster manager fallover script would allow the surviving processor to act for the failed machine. It gains access to all shared disks and starts a second database server identical to the one that was running on the failed machine. At that point, client applications would check for cluster status and reconnect to the new database server.

Clients of HACMP/6000 clusters can be connected by serial links or attached to the network. However, serial devices must be switched to the fallover host manually unless they have been connected to a network-attached terminal server or automated switching device.

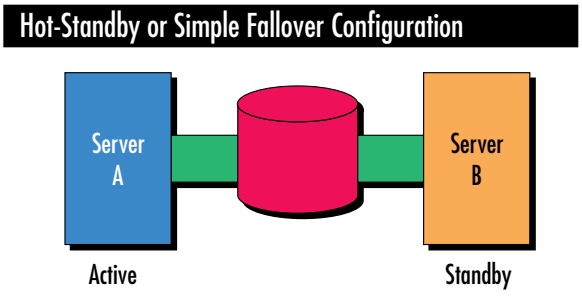



**Figure 2. A hot-standby or simple fallover configuration**

### High-Availability Configurations

There are many ways to configure a highly available RISC System/6000 cluster.

**Hot standby or simple fallover**: In a hot-standby or simple fallover configuration (shown in Figure 2), the active processor executes the application and the standby processor waits for a failure. The standby machine is not necessarily idle; the work being done on this machine may be stopped if the total capacity is needed to take over for the primary machine. When the primary processor is returned to service, it reclaims the application and resources.

**Rotating standby**: Rotating standby is like hot standby except that the roles of primary and standby are not fixed. When the previous primary is returned to service, it does not reclaim the application.

**Mutual takeover or partitioned workload**: This configuration allows each processor to back up the applications running on each of the other processors. In Figure 3, Server A runs applications 1 and 2. Server B runs applications 3 and 4. Server C runs applications 5 and 6. If Server B fails, Server A can restart application 3, while applications 1 and 2 continue to execute. Server

C can restart application 4 while continuing to run 5 and 6.

## HACMP Benefits

The following are some of the unique benefits of HACMP/6000 for businesses with mission-critical applications.

- HACMP/6000 improves the utility of installed RISC System/6000s. It is a cost-effective addition to existing hardware and software. The incremental cost to implement HACMP/6000 is low compared to buying additional hardware.
- HACMP/6000 can expand the capacity of a current installation. HACMP/6000 provides scalability without replacing installed hardware and software. This benefit comes from using HACMP/6000 2.1 in the mutual takeover configuration—effectively doubling system capacity by splitting the workload between two or more systems. The system also provides increased availability.

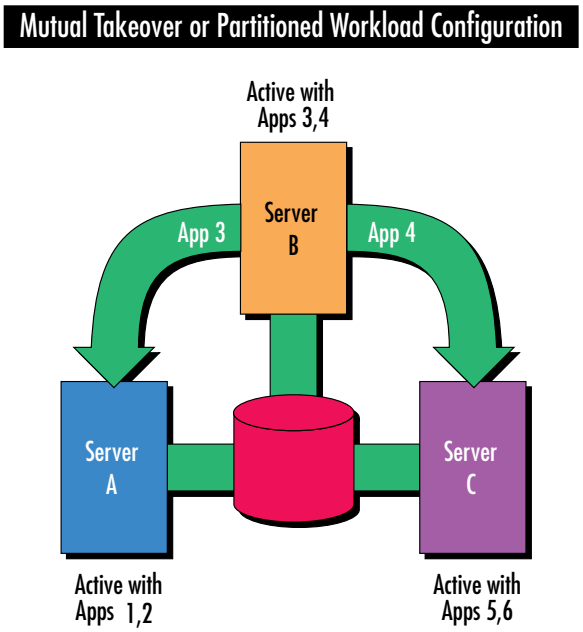



**Figure 3. A mutual takeover or partitioned workload configuration**

**HACMP/6000 provides scalability without replacing installed hardware and software.**

- The concurrent disk access configuration also provides availability and horizontal scalability when a single copy of the data is required for executing applications. The data cannot be partitioned as in the mutual takeover configuration.
- HACMP/6000 is an alternative to symmetric multiprocessing configurations when clustering can provide the increased availability that ordinary symmetric multiprocessors cannot.

## Performance Considerations

HACMP/6000 cluster performance can be measured and reported in many ways. In a mutual takeover/partitioned workload cluster environment, the applications and data are spread across two to four machines. With minimal interaction between nodes in a cluster, this partitioned environment will have almost 100% efficiency in each node. When a fallover occurs and the backup machines take over for a failed node, performance will be degraded during the time the node is down.

With concurrent disk sharing, distributed locking reduces the efficiency of the scaling. The scaling with Oracle7 Parallel Server was measured at 80% efficiency for a cluster of two RISC System/6000s Model 980. That means the cluster performed at 1.6 times the performance of one machine. IBM estimates four-way cluster efficiency at 2.6 times one machine for transaction processing applications. IBM is conducting four-way tests with results expected in the third quarter of 1994. In long transactions such as the TPC-C benchmarks, we anticipate more efficient scaling because the lock overhead is less. Estimates range from 1.8 times in two-way clusters to 3.2 times in four-way configurations.

## Future Directions

IBM expects to provide the following additional capabilities in HACMP/6000 during 1994:

- Eight-way clusters
- Concurrent file system providing a single system image to applications
- Enhanced cluster manager administrative tools
- Certification of HACMP/6000 interfaces to:
  - Distributed Computing Environment (DCE), CICS/6000™, and Encina®
  - Load-leveler queuing and load-balancing extensions
  - DB2/6000™

## Summary

HACMP/6000 can provide commercial users with quick recovery from system failures at a reasonable cost. Although HACMP/6000 does not provide complete fault tolerance and continuous operation, it does provide a minimal recovery time after a system failure.

HACMP/6000 provides horizontal scalability by allowing applications to share the disks and CPUs of clustered RISC System/6000 processors.

HACMP/6000 should be considered in situations where users require the following:

- High availability for mission-critical databases or OLTP applications
- Open systems functionality and portability
- Growth in system resources, both horizontally and vertically

**Daniel P. Cox**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Cox is the brand manager for clustered systems in the RISC System/6000 Division. After joining IBM in 1968 as a programmer in the IBM San Jose Laboratory, he has held positions in development programming, systems engineering, product planning, and management. Before becoming brand manager, Mr. Cox was the product manager for high-availability systems in the RISC System/6000 Division. He has been involved with HACMP/6000 since its beginning—in both planning and implementation. Mr. Cox has BS and MS degrees from San Jose State University.

**Frank Lawlor**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Lawlor is the AIX architect for high-availability and clustered systems. Since joining IBM in 1970 in Poughkeepsie, New York, he has held several lead architectural positions and also positions in hardware and software development for S/370, AS/400®, and the RISC System/6000. Mr. Lawlor has a BS in Physics from Fordham University and an MS in Electrical Engineering from Columbia University.

# Migrating to HACMP/6000 2.1

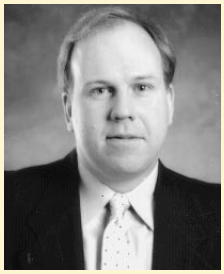
By Thomas Casey and Robert Metcalf

HACMP/6000 Release 2.1 became generally available in December 1993. This article describes the steps and issues involved in migrating an existing HACMP/6000 cluster from Release 1.2 to Release 2.1. In addition, the article provides background for system administrators who are evaluating extending a two-node cluster to three or four nodes.

Building on the strengths of RISC System/6000 servers and the AIX operating system, High Availability Cluster Multi-Processing/6000 (HACMP/6000) provides a set of services that guarantee quick recovery if a critical component fails. HACMP/6000 is designed for database and transaction-processing applications that require highly available, scalable configurations.

Release 2.1 extends HACMP/6000 with features that increase flexibility in both cluster configuration and fallover (fallover occurs when the HACMP/6000 software detects a node failure and reconfigures the cluster to compensate), and make HACMP/6000 easier to configure, customize, and troubleshoot. Sites that have installed an earlier version of the HACMP/6000 software will want to upgrade to Release 2.1 to take advantage of the added functionality provided by this new release:

- Support for four-way clusters
- A new flexible, extensible cluster configuration methodology
- Simplified, centralized cluster configuration
- Cluster verification tool
- Cluster diagnostic tool
- New event customization facility
- More enhancements to the base product



Thomas Casey

Robert Metcalf

## Transitioning to Release 2.1 — A Conceptual Background

Release 2.1 includes several new features that change the process of configuring an HACMP/6000 cluster. This section identifies these features and helps system administrators understand what is necessary to migrate an existing cluster to Release 2.1.

### Resource-Based Clusters

In earlier releases of HACMP/6000, a cluster could have two active nodes. Those releases provided a set of predefined configurations—such as hot standby or one-sided takeover—that encompassed a range of high-availability solutions. In Release 2.1, a cluster can have up to four active nodes, which dramatically increases the number of possible cluster configurations. Therefore, a new way of configuring a cluster was developed.

Release 2.1 does not provide a set of predefined configurations. It uses a flexible configuration methodology that associates resources with a node or IP address. This allows cluster members to define which resources they own and which they take over from peer nodes.

### Types of HACMP/6000 Resources

A focus on resource ownership allows for many cluster configurations and lays the foundation for configuring four-way clusters and more. Release 2.1 defines three types of resources:

- **Owned resources**: Owning a resource denotes a direct relationship between a single node and a resource. When the owning node is active in the cluster, the identified resource

is owned by that node. In concurrent access configurations, a single-disk resource can be owned by several nodes.

- **Takeover resources**: A takeover resource binds to a designated peer node when the owning node detaches from the cluster. Taking over a resource denotes a secondary relationship, active only if the owner node is not available.
- **Rotating resources**: A rotating resource is associated with a specific IP address. It is owned by the node currently assuming that IP address as long as that node has the IP address. If that node detaches from the cluster, then the node assuming the designated IP address becomes the owner of the resource.

### Example of a Resource-Based Cluster

The new resource-based methodology is best illustrated through an example. Release 1.2 supports a hot-standby configuration in which a server node provides highly available services and a standby node does no processing—waiting idly for the server to fail. In Release 2.1, there is no hot-standby configuration per se. The relationship among the nodes in a cluster is implicit in the resources defined for the nodes. Figure 1 shows a two-node hot-standby cluster after it has been converted to the resource-based model.

In this setup, a single node owns all highly available resources. This node is the *owner node* (called the server node in Release 1.2). Although the second node owns no resources, the highly available resources are defined to this node as takeover resources. This node, called the *takeover node* or *standby node*, stands idle, waiting for the owner node to detach from the cluster. If the owner node detaches, the takeover node assumes control of the resources owned by the owner node, restarts the applications, and services clients. The takeover node remains active until the owner node rejoins the cluster. At that point, the takeover node releases the highly available resources and returns to an idle state.

This configuration could also be set up using rotating resources—the only difference would be that when the node that had originally detached from the cluster returns, it becomes the standby.

The benefit of this flexible resource-based methodology will become apparent when extending a cluster to three or four nodes (discussed later in the section "Extending HACMP/6000 Beyond Two Nodes").

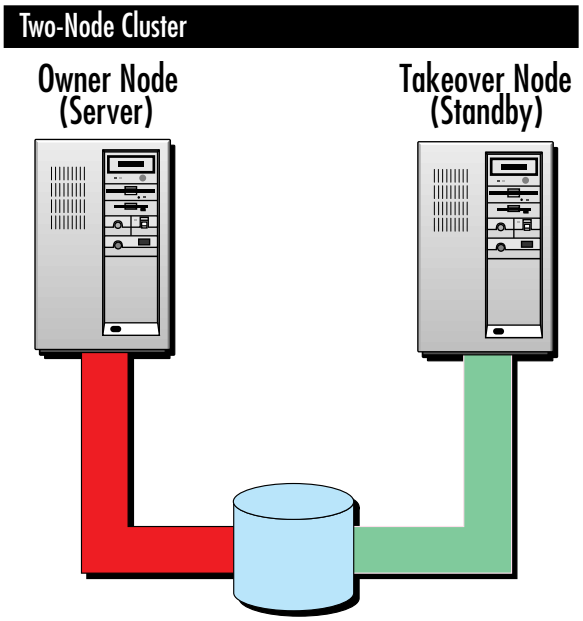



**Figure 1. A two-node resource-based cluster**

### Events

Release 2.1 includes redesigned cluster configuration scripts based on the concept of a *cluster event*. This event represents a change within the cluster that invokes a response from the Cluster Manager (the daemon that monitors cluster status). An example of an event is a `node_down`, where a node detaches from the cluster. An event, in turn, can consist of a series of *subevents*. For example,

| Events | Subevents |
|---|---|
| config_too_long | acquire_service_addr |
| fail_standby | acquire_takeover_addr |
| join_standby | get_disk_vg_fs |
| network_down | node_down_local |
| network_down_complete | node_down_local_complete |
| network_up | node_down_remote |
| network_up_complete | node_down_remote_complete |
| node_down | node_up_local |
| node_down_complete | node_up_local_complete |
| node_up | node_up_remote |
| node_up_complete | node_up_remote_complete |
| swap_adapter | release_service_addr |
| swap_adapter_complete | release_takeover_addr |
| unstable_too_long | release_vg_fs |
| | start_server |
| | stop_server |

**Figure 2. Cluster events and subevents**

a node_down event could include the stop_servers, release_service_addr, and release_vg_fs subevents. For each event and subevent, HACMP/6000 provides a corresponding script of the same name in the /usr/sbin/cluster/samples directory. Figure 2 lists the new 2.1 files in that directory.

### Event Scripts Replace topchng.rc and netchng.rc Scripts

The event and subevent scripts replace the topchng.rc and netchng.rc configuration scripts as the Cluster Manager tool for responding to changes in cluster status. If either the topchng.rc or the netchng.rc script has been modified, the system administrator must use the event customization facility provided with Release 2.1 to re-create the necessary functionality.

The /usr/sbin/cluster/clinfo.rc script remains virtually the same as previous versions in both function and intent. It is neither an event nor a subevent—it is a script that runs on HACMP/6000 clients to flush a stale Address Resolution Protocol (ARP) cache.

**Extending HACMP/6000 clusters provides a graceful, incremental growth path for mission-critical applications.**

### Event Customization Facility

Release 2.1 provides a System Management Interface Tool (SMIT) interface that system administrators can use to customize event processing without modifying the event scripts distributed with the product. Since future updates may require that an event script be replaced, it is best to use local customization outside the event script when applying updates.

Using the event customization facility, the system administrator can specify custom commands or scripts to tailor event processing for a site, including:

- Pre-event and post-event commands
- Event notification
- Event recovery and retry

### Application Servers Replace node.servers Script

In earlier versions of HACMP/6000, highly available applications were started from the node.servers script, which was called by the topchng.rc configuration script after a change in cluster status. Release 2.1 introduces the concept of *application servers* and views applications as resources, similar to a filesystem or IP address. Release 2.1 provides a SMIT interface for configuring applications made highly available by the cluster, eliminating the need to edit the node.servers script. Configuring an application server associates a meaningful name with a server application, and points the cluster event scripts to the application server's start and stop scripts.

### Extending HACMP/6000 Beyond Two Nodes

Release 2.1 supports three- and four-way clusters in both concurrent and nonconcurrent access environments. System administrators should examine the benefits of three- and four-node clusters to determine whether it is beneficial to extend the cluster. Three- and four-way clusters provide the following benefits to HACMP/6000 sites.

- **Scalability**: Extending HACMP/6000 clusters to four nodes provides a graceful, incremental growth path for mission-critical applications by allowing the workload to be spread among multiple processors sharing common disk resources. It also allows users to execute and store data on multiple systems with minimal changes to the application.
- **Increased performance**: A cluster's ability to provide high performance is directly related to its ability to scale. Clustering provides a form of parallel processing that splits the workload among more processing resources without adding excessive overhead.
- **Enhanced availability**: As the number of nodes increases, the overall level of availability provided by the cluster also increases.

Before extending a cluster to three or four nodes, system administrators should evaluate the following disk and network considerations.

A four-way HACMP/6000 cluster can use industry-standard Small Computer Systems Interface-2 (SCSI-2) differential disk devices, including the IBM 7135-110 RAIDiant disk array, or the new IBM 9333-011 and IBM 9333-501 serial disk subsystems. SCSI-2 differential devices are supported only in nonconcurrent configurations, while the IBM 7135-110 RAIDiant disk array and the 9333-011 and 9333-501 serial disk subsystems are supported in both concurrent and nonconcurrent configurations.

Eight initiators or targets can connect to a SCSI-2 differential bus that is terminated on the bus by Y cables. The most likely configuration for IBM 7203 SCSI disks and IBM 9334 devices in a four-way cluster is separate dual-ported chains. Each disk and adapter on the same SCSI bus requires a unique SCSI address. The 7135-110 RAIDiant disk array can support four cluster

## Four-Node Configuration

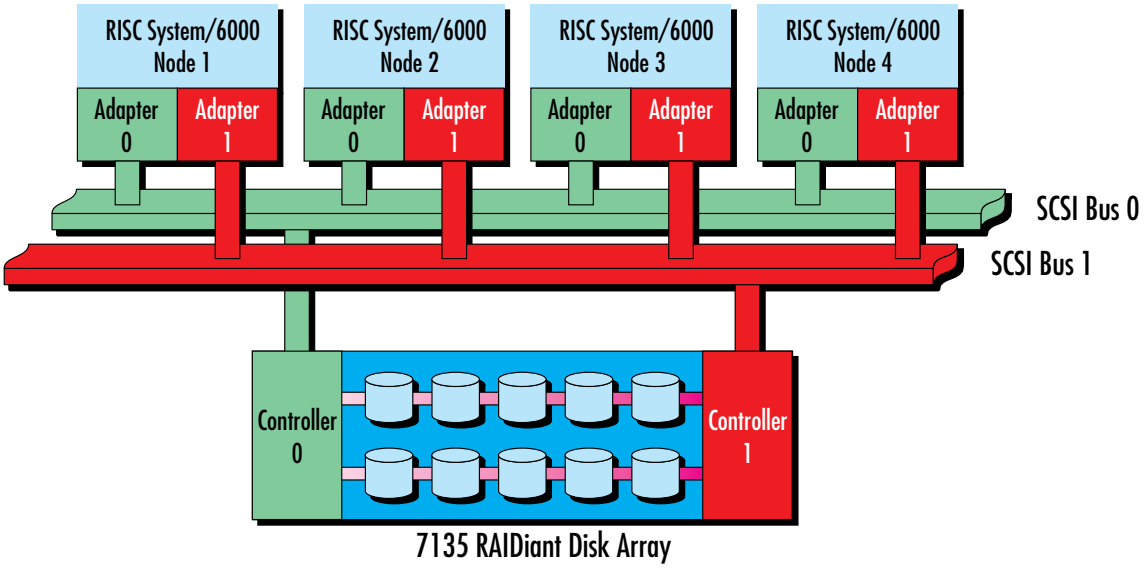



**Figure 3. Four-node RAIDiant configuration**

nodes directly connected to it. Typically, the 7135-110 RAIDiant disk array is configured with two controllers, each on its own SCSI bus, as shown in Figure 3. Four cluster nodes can attach to a single 9333-011 or 9333-501 disk subsystem.

As the number of nodes increases, the amount of communication between nodes increases as well. This is especially true in concurrent access environments, which generate large volumes of lock manager traffic. When extending a cluster to three or four nodes, it is beneficial to add a high-speed network such as a Fiber-optic Distributed Data Interchange (FDDI) for internode communication. This prevents traffic from becoming a bottleneck for clients trying to reach the cluster. At the very least, lock manager traffic should be routed over a separate, private network.

## Migrating to Release 2.1

This section describes the tasks that system administrators must complete to migrate an existing HACMP cluster from Release 1.2 to 2.1. These tasks include the following:

- Using the `clconvert` utility to convert cluster and node configuration information into the appropriate 2.1 Object Data Manager (ODM) object classes, and to redefine the cluster configuration from the 1.2 role-based model to the 2.1 resource-based model
- Using SMIT to extend the cluster to three or four nodes (if desired)
- Using the event customization facility to re-create any custom processing added to the `topchng.rc` or `netchng.rc` configuration scripts distributed with Release 1.2
- Using the SMIT interface (which replaces the `node.servers` file) for application servers to register the start and stop scripts for applications to be made highly available by HACMP/6000
- Populating the `/usr/sbin/cluster/clhosts` file

### Preparations for Upgrading to Release 2.1

Before upgrading a Release 1.2 installation to Release 2.1, the system administrator must complete the following steps:

1. For each node, archive the `/usr/sbin/cluster` directory to a readily accessible place on disk so that it is easy to retrieve and compare localized script and configuration files. Also, capture and store the output from the `/usr/sbin/cluster/clgetenv` command as a snapshot of the node's view of its HACMP/6000 environment.
2. If the 1.2 installation is applied but not committed, commit it so that 2.1 can be installed over 1.2.

**Clustering splits the workload among more processing resources without adding excessive overhead.**

3. Do a `mksysb` of each node. If the site is running the HACMP/6000-supplied Concurrent Logical Volume Manager (CLVM) software, see the special instructions in the *HACMP/6000 2.1 Release Notes* about `mksysb` and CLVM.
4. If running the CLVM, revert to the standard AIX LVM before upgrading.

### The Upgrade Process

After completing the prerequisite tasks described above, the system administrator can upgrade the 1.2 environment to 2.1. During the upgrade, HACMP/6000 runs the new `/usr/lpp/cluster/tools/clconvert` utility, which performs the following functions:

- Converts cluster and node configuration information into the appropriate 2.1 ODM object classes
- Redefines the cluster configuration from the 1.2 role-based model to the 2.1 resource-based model

**Once the decision has been made to extend the cluster, adding the actual nodes is simple.**

### Conversion Tool

Installing Release 2.1 over a committed 1.2 installation causes an automatic update of the 1.2 configuration information to 2.1 format. The `/usr/sbin/cluster/cluster.cf` and `/etc/objrepos/hacmp6000` (node information) files are used to populate the appropriate 2.1 object classes. System administrators can back this out using the standard SMIT interface to modify or delete configuration information.

The conversion fails if either the `/usr/sbin/cluster/cluster.cf` or `/etc/objrepos/hacmp6000` files are not available. If there is already configuration information in one of the Release 2.1 ODM object classes, it is overwritten during the installation.

If the original `cluster.cf` file is correct and the `/etc/objrepos/hacmp6000` file exists, the new ODM classes are updated. The following output should appear in the `smit.log` and on the console:

```
Adding new HACMPcommand ODM entries.
Adding new HACMPevent ODM entries.
clconvert successful.
```

This can be verified by running the `cllsif` command for cluster topology information and the `clgetres -A` command for node information.

**Note**: To prevent the conversion tool from being run automatically, the system administrator can rename the `/usr/sbin/cluster/cluster.cf` and `/etc/objrepos/hacmp6000` files before installing Release 2.1.

### Moving Configuration Information to the ODM

In earlier versions of the HACMP/6000 software, cluster configuration information was stored in the `/usr/sbin/cluster/cluster.cf` file, a stanza-based ASCII file. The easiest way to access this file was through the `/usr/sbin/cluster/cllsif` command. Changes were made to the file using the `smit hacmp` fastpath. Node information was accessible through the `/usr/sbin/cluster/clgetenv` command, which examined the `/etc/objrepos/hacmp6000` file.

HACMP/6000 2.1 moves this configuration information into the ODM, creating the following seven new object classes in the `/etc/objrepos` directory:

```
HACMPadapter
HACMPcluster
HACMPcommand
HACMPevent
HACMPnetwork
HACMPresource
HACMPserver
```

### Files Saved During the Upgrade

The system administrator can now install the new release, as described in Release 2.1 documentation. The following files are saved, with an `.OLD` extension:

```
clinfo.rc.OLD
cluster.cf.OLD
netchng.rc.OLD
node.servers.OLD
node.vars.OLD
topchng.rc.OLD
```

Any customized subdirectories (such as `/usr/sbin/cluster/local`) are left untouched by the installation procedure. The system administrator uses the content of the old scripts to customize the 2.1 environment. It is especially important to review the `topchng.rc`, `netchng.rc`, and `node.servers` scripts, which may have been customized at the site for Release 1.2.

### Extending the Cluster to Three or Four Nodes

Once the decision has been made to extend the cluster, adding the actual nodes is simple. Release 2.1 provides a centralized installation facility that allows all the configuration information for additional nodes to be entered into SMIT screens on

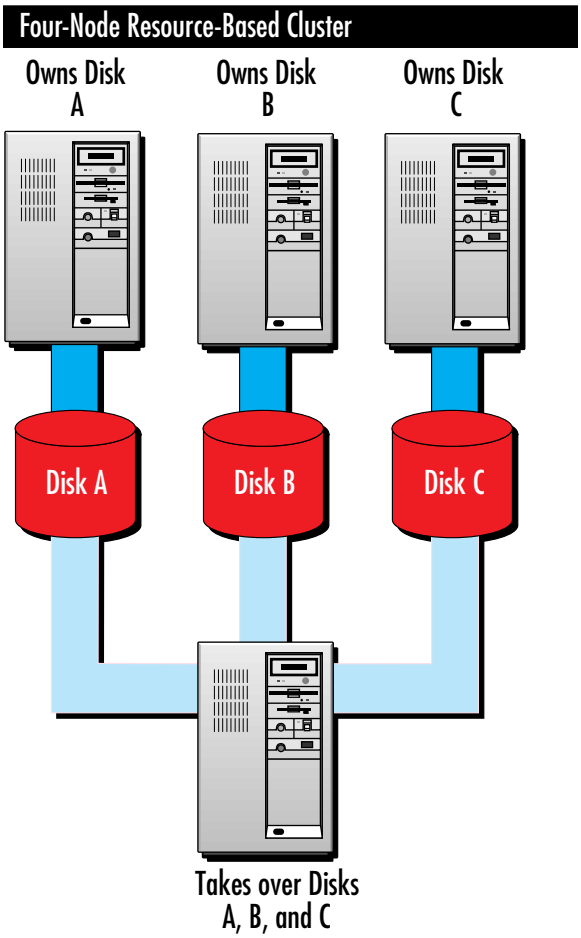



**Figure 4. A four-node resource-based cluster**

a single node. Then, again using the SMIT interface, the system administrator can execute a single command to propagate the updated cluster configuration to all nodes in the cluster. This simplified, centralized cluster configuration facility, coupled with the new resource-based cluster model, allows system administrators to easily extend an existing two-node cluster to three or four nodes.

In the two-node cluster example shown earlier in the article, a single-standby node backs up a single-server node. This configuration can easily be extended to a cluster in which two or three server nodes are backed up by a single standby. To do this, the system administrator would go to a single node in the cluster, define the new nodes and the resources that they owned, then define the resources owned by the new nodes as "takeover" resources to the existing standby node. Figure 4 shows the new configuration.

### Customizing Event Processing

When migrating from Release 1.2 to 2.1, system administrators use the event customization facility to re-create any functions that have been added to the `topchng.rc` or `netchng.rc` configuration scripts.

For example, suppose that the operations manager needs to know immediately—day or night—whenever a fallover occurs. In Release 1.2, the HACMP/6000 Cluster Manager calls `topchng.rc 1 down` when the primary node fails. During the installation, the system administrator can modify the `topchng.rc` script so that when the primary server fails, a script called `wakeup` runs on the secondary. The `wakeup` script can use Kermit to auto-dial the pager of the operations manager and pass on a specific number combination.

In Release 2.1, the system administrator can avoid editing the event scripts by defining the `wakeup` script as a post-event associated with the `node_down_complete` event on the takeover node. By doing this, the takeover node automatically runs the `wakeup` script after it finishes processing the first node's failure.

No HACMP/6000 scripts need to be edited to add this in Release 2.1. Instead, the system administrator would use the `smit hacmp` fastpath to complete the following steps:

1. Select the `node_down_complete` event.
2. Enter the full path for the `wakeup` script in the post-event command entry.

### Defining Application Servers

When migrating to Release 2.1, system administrators must use the SMIT interface for application servers to register the application start and stop scripts identified in the `node.servers` file. For example, suppose that a 1.2 cluster was used to make a database highly available as a back-end server. The `node.servers` script calls the `/usr/sbin/cluster/local/start_db_server` script during a `node_up` event to start the database and the `/usr/sbin/cluster/local/stop_db_server` script to stop the database during a `node_down` event. In Release 2.1, system administrators complete the following steps:

1. Using the `smit hacmp` fastpath:
   - Give the application server the logical name `db_server`.
   - Cite the path for the start script as `/usr/sbin/cluster/local/start_db_server`.

- Cite the path for the stop script as `/usr/sbin/cluster/local/stop_db_server`.

2. On the local node, define the application server as an owned resource.
3. On the takeover node, cite the application server as a takeover resource.
4. Be sure that each node's version of this resource is entered and configured correctly for that node (that is, path, local variables, hostname considerations, and so on).

### Editing the clhosts File

In Release 2.1, the Cluster Information (`clinfo`) daemon uses a new file, `/usr/sbin/cluster/clhosts`, to store the hostnames or address of any HACMP/6000 server with which `clinfo` can communicate. When migrating from 1.2 to 2.1, system administrators must add all service labels or addresses of HACMP/6000 servers to this file on HACMP/6000 clients. An example of a `clhosts` file is shown in Figure 5.

```
clam_en0      # clam service
mussel_en0    # mussel service
oyster_en0    # oyster service
100.50.10.1   # shrimp service
```

**Figure 5. A clhosts file**

**Thomas Casey**, CLAM Associates, Inc., 101 Main Street, Cambridge, MA 02142. Internet: tom@clam.com. Mr. Casey is the manager of CLAM's technical writing group. He has a BS from Trinity College and an MS from Emerson College, both in Boston.

**Robert Metcalf**, CLAM Associates, Inc., 101 Main Street, Cambridge, MA 02142. Internet: bobmet@clam.com. Mr. Metcalf is a senior member of CLAM's technical support staff. He has a BS from Suffolk University and an MS from Simmons College, both in Boston.

## First PowerPC-Based Notebook

The RISC System/6000 N40, a lightweight color notebook computer, combines the power of the PowerPC 601™ microprocessor and AIX in the industry's first PowerPC-based notebook workstation. Running at 50 MHz, the N40 achieves a SPECint92™ benchmark rating of 41.7, a SPECfp92™ rating of 51, and an Xmark rating of 2.58, making it more powerful than any notebook computer and many desktop workstations.

The 6.9-pound N40 features a 9.4-inch active-matrix color screen that offers wide-angle viewing in 256 colors. The N40's video memory supports up to a 1280 x 1024 image, which can be viewed through a pan-and-zoom feature on the display or through an externally connected monitor. Also featured is IBM's TrackPoint II™ pointing device, which eliminates the need for an external mouse.

Communications and networking features include external ports for Ethernet network support; SCSI-2 diskette drive support; and support for PCMCIA adapters. Other standard features include a removable disk drive with a 340 MB capacity; main memory support from 16 MB to 64 MB; an external display port supporting 1280 x 1024 resolution and up to 256 colors; ports for an external mouse, keyboard, and AppleTalk® printers; and a built-in speaker and microphone. The N40 also features Tadpole's Nomadic Computing Environment™, providing users with rapid save and resume capabilities, power management, portability tools, and other UNIX mobile computing innovations.

The N40 is available now, at $11,995. ■

DATABASE

# DB2 Parallel Edition

By Gilles Fecteau

This article describes IBM's plans for a parallel DB2/6000 database server using the shared-nothing hardware model and function shipping. It also shows how this approach provides a scalable database server that meshes with the IBM POWERparallel™ architecture. The DB2 Parallel Edition will allow customers to grow databases with near linear performance improvement by adding nodes. The beta testing is now underway and availability will be announced later in 1994.

Parallel processing—concurrently executing two or more processors as a single unit—supports large UNIX-based applications. Most Relational Database Management Systems (RDBMSs) are being enhanced to support parallel processing to take advantage of low-cost workstations.

IBM began studying parallel technology in 1989 with prototypes developed by IBM Research. The IBM Research Division developed a prototype for a parallel database system using multiple PS/2 workstations and the OS/2® operating system. This prototype was demonstrated at COMDEX® in 1990.

Since then, IBM has been working on an architecture that allows DB2/6000 to run on several independently connected workstations managed as a single database by DB2/6000 parallel code. This product will operate on a range of hardware from LAN-connected RISC System/6000 systems to the IBM POWERparallel system.

Parallel query execution alone is not sufficient—large database users also need the ability to maintain very large databases. Achieving the goal of an effective parallel database means focusing on database management utilities and effective parallel queries.

Gilles Fecteau

## Parallel Architectures

Several possible architectures can exploit multiple processors, large memory, and many disks. The most common architectures are as follows:

- **Shared nothing** uses multiple processors, each with its own memory and disk storage.
- **Shared disks** uses multiple processors, each with its own memory and shared disk storage.
- **Symmetric Multiprocessing (SMP)** uses multiple processors that have common memory and shared disk storage.

IBM is extending DB2/6000 to support the shared-nothing architecture. The database manager handles database requests from an application. It uses a communications link to coordinate the work of the multiple processors. This architecture was selected for two reasons:

- **Scalability**: It can scale to hundreds of processors.[1]
- **Portability**: Since it requires only a communications link between processors, its design can be ported to any platform that has communications. This is not true for SMP or shared disk. Although performance varies with the efficiency of the communication protocol, the high-speed switch in IBM POWERparallel systems (as well as many forms of UNIX sockets) ensures good performance.

The DB2 Parallel Edition will support SMP; in a parallel database configuration, each node could be an SMP.

## Shared-Nothing

Figure 1 shows a DB2® parallel database running on multiple RISC System/6000s supporting a net-

[1] Dewitt and Gray. "Parallel Database Systems: The Future of High-Performance Database Systems," *Communications of the ACM* 35 (June 1992).

work of clients. A *data node* is the disk storage plus a portion of the database engine that manages the disk. In the DB2 parallel implementation, a database is stored across a network of processors that provide separate buffers, lock structures, logs, and disks for each processor. This prevents a cache contention problem that could result from all processors sharing one set of resources—as with an SMP implementation.

Because each processor has separate logs, applications are not limited to the I/O bandwidth of a single log and can perform recovery from failure in parallel.

### Function Shipping

To minimize communication between processors, relational operators are executed on the processor containing the data whenever possible. Therefore, a central lock manager is not required as with most shared-disk implementations. This process of sending the work to the location of the data is known as *function shipping*.

Suppose that an `EMPLOYEE` table is distributed over multiple processors, and the following SQL statement is executed:

```
SELECT EMP_NO FROM EMPLOYEE
   WHERE SALARY>100000
```

The database manager on the *coordinating processor* (the processor from which the statement was issued) issues a request. This request (sent to every other processor in the group) is to select its subset of rows that have a `SALARY` value over 100,000. Each processor in the group then returns its answer set to the coordinating processor for final processing.

No central lock manager is needed because each processor initially extracts an answer from its own part of the `EMPLOYEE` table. At regular intervals, a *global deadlock detector* analyzes locks held to determine if a deadlock is present, then selects a victim to resolve the deadlock. This avoids sending thousands of lock requests between systems—a quick deadlock detection occurs every few seconds.

In *I/O shipping*—a less efficient alternative to DB2's function shipping—one or more of the group's processors are arbitrarily selected to run a query. All processors must send their data for a given query to the executing processors. Since all data pages of the `EMPLOYEE` table must be sent, more data movement occurs among processors than with function shipping. With function shipping, only the qualifying rows (employees with a salary over $100,000) are sent from the processor owning the data to the processor coordinating the query.

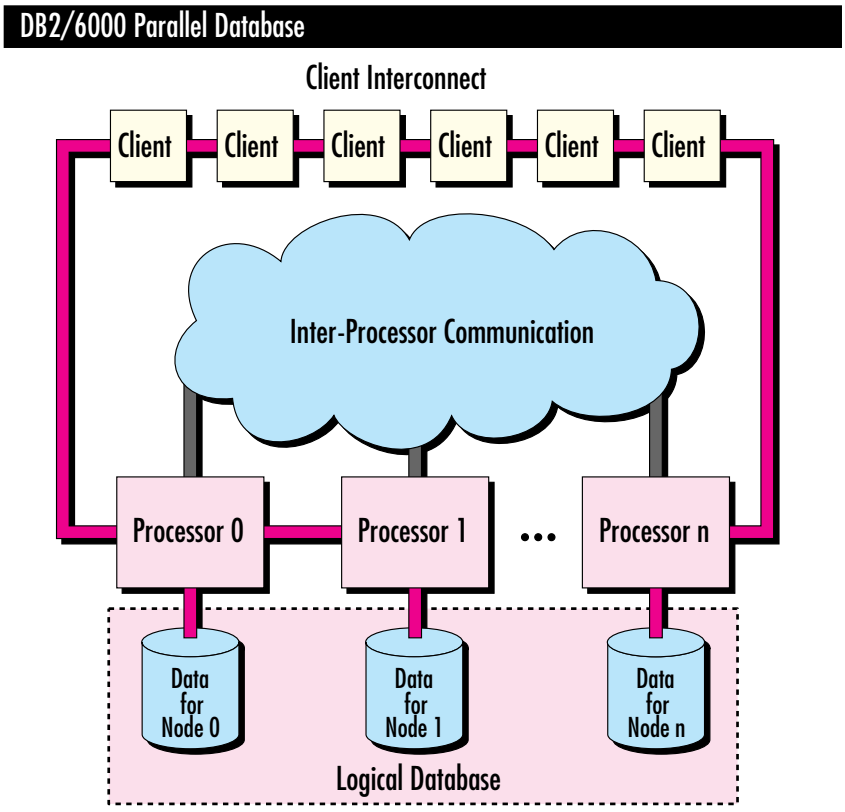



**Figure 1. A parallel database environment**

The system configuration for the I/O-shipping model differs from the function-shipping model.

- **I/O shipping**: Nodes that specialize in I/O must be configured with large numbers of disks, and other nodes without disks handle user queries.
- **Function shipping**: The configuration must provide a moderate number of disks on each processor, and the data must be partitioned over many processors.

The number of disks required for either model is generally the same for large databases from 50 GB to terabytes. I/O shipping may be less expensive for databases smaller than 50 GB, but these databases are not likely to use parallel technology.

DB2 Parallel Edition does not limit the function-shipping model to queries only. It can also perform database updates and run utilities. When a row is to be inserted into a table, it is sent to

the appropriate node where it is inserted (the appropriate node is determined by the hashing algorithm used for table partitioning). The index entries are updated and the row information is logged at the same node. Index maintenance, locking, and logging are distributed across processors.

## Data Placement

For large databases, data placement can be complex and system administration can be difficult; therefore, appropriate Data Definition Language statements and administration utilities are needed to manage data partitioning. The DB2 parallel implementation is easier to administer than a non-partitioned database of the same size. For example, without partitioning, functions such as backup would take too long. Tools are provided for efficiently managing the large tables that occur in databases. The DB2 design ensures that database design decisions are separated from load-balancing decisions.

**The DB2 Parallel Edition is easier to administer than a non-partitioned database of the same size.**

Two important features of the DB2/6000 parallel database design that help in data partitioning and database administration are the partitioning key and nodegroups.

### Partitioning Key

For a database that has many tables, an application developer can define a partitioning key for each table. Frequently joined tables should be partitioned on their respective join columns. The partitioning key information is specified as part of the `CREATE TABLE` statement, shown in Figure 2.

```
CREATE TABLE ACCOUNT (
        C_BRANCH INTEGER,
        CUST_NO INTEGER,
        CUSTOMER_NAME VARCHAR(50),
        LAST_DATE DATE,
        BALANCE DEC(8,2),
        .
        .
        .
        )
IN SMALLPOOL PARTITION BY HASHING(C_BRANCH)
CREATE TABLE BRANCH (
             BRNO INTEGER NOT NULL,
             ADDRESS VARCHAR(200),
             .
             .
             .
             PRIMARY KEY(BRNO) )
IN SMALLPOOL PARTITION BY HASHING(BRNO)
```

**Figure 2. Examples of CREATE TABLE statements with partitioning**

The first statement specifies that the `ACCOUNT` table is partitioned on the `C_BRANCH` column; the second specifies that the `BRANCH` table is partitioned on the `BRNO` column. More than one column can be specified as the partitioning key. A system-defined hashing function is applied to the partitioning key value to determine in which processor a particular row will reside.

The `C_BRANCH` and `BRNO` columns were chosen as the partitioning keys for these two tables because they best suit the application. The suitability of these columns as partitioning keys remains valid, regardless of table size and the number of processors on which the tables are partitioned. To design tables, designers do not need prior knowledge of the configuration of the parallel system and the load on the system. Tuning for load balancing can be done after database definition time.

In the example in Figure 2, the processors on which the tables are partitioned are not specified directly. The `CREATE TABLE` statement has been extended to include an `IN <nodegroup name>` clause that provides this information. A *nodegroup* is an arbitrary name given to a set of nodes that will be used for tables. Tables in the same nodegroup will be partitioned over the same processors. The identifier `SMALLPOOL`, following the keyword `IN`, is the nodegroup name.

### Nodegroup

In a shared-nothing hardware configuration, such as IBM's SPx family of parallel processors or LAN-connected workstations, each processor runs the equivalent of a single-node DB2/6000 database system. The database storage capabilities of each processor are the same as those provided by DB2/6000, including segmented table support that can implement tables of up to 64 GB. With the DB2 Parallel Edition, however, the storage capability of a system with $n$ nodes is $n$ times that of a uniprocessor DB2/6000 system.

Database tables can be defined across a set of nodes by first defining a nodegroup and then creating tables in it. A nodegroup can be created as follows:

```
CREATE NODEGROUP SMALLPOOL
   ON NODES (DBMACH1,DBMACH2,DBMACH6)
```

This statement defines a nodegroup called SMALLPOOL, consisting of three processors: DBMACH1, DBMACH2, and DBMACH6. Tables created in nodegroup SMALLPOOL are partitioned across these three processors. A nodegroup can contain one or more processors, and a processor can be a member of more than one nodegroup in the same database or across databases.

As the size of tables or the number of processors in the system increases, the ALTER NODEGROUP statement can be used to add processors to an existing nodegroup. After a processor is added to a nodegroup, data belonging to tables in the nodegroup can be redistributed using the rebalance utility.

The following characteristics make the DB2 Parallel Edition easy to manage:

- The application view of a table is separate from the physical placement of data.
- Data can be distributed over multiple disks to reduce I/O bottlenecks.
- The number of processors can be increased as the workload or size of the database increases.

## Parallel Query Processing

The DB2 Parallel Edition generates a parallel execution strategy for all SQL statements using a cost-based relational database optimizer. After comparing several parallel execution strategies for each SQL statement, the cost-based optimizer selects the most efficient one. An SQL statement (such as SELECT, INSERT, UPDATE, or DELETE) is divided into several separate tasks. The *coordinator task* runs at the node where the application connects. It fetches input data from the application and returns the answer set to the application. Subordinate tasks, called *slave tasks*, perform the bulk of the activity required for the query and cooperate with each other when necessary. While there can be only one instance of a coordinator task for each application, there can be multiple instances of each slave task.

Since DB2 Parallel Edition does not impose any new restrictions on SQL statements, the investment made in existing applications remains. Generating a parallel execution strategy for a given SQL statement is automatic. A DB2/6000 application program does not have to be recompiled to exploit parallel execution. When the application program is bound to a parallel database, the appropriate parallel execution strategy is generated and stored, if required.

The optimization process of an SQL statement in a parallel DB2 Parallel Edition environment is based on two primary factors:

- **The distribution of the data across nodes**: DB2 Parallel Edition supports data partitioning by hashing the values of a set of columns across a set of nodes.
- **The cost of functions associated with different operations**: The DB2/6000 optimizer has cost formulas for the different tasks. New assumptions are added to account for parallel operations and messages (rows). The database manager generates the optimal parallel plan rather than having the best serial plan simply executed in parallel. The database manager performs this optimization without any external information.

**The DB2 Parallel Edition generates a parallel execution strategy for all SQL statements using a cost-based relational database optimizer.**

The repertoire of parallel strategies used by DB2 Parallel Edition includes parallel table scans and parallel index scans for tables; co-located, hashing redistributed, or broadcast joins for joins; and local and global aggregates, including the GROUP BY clause.

The full power of parallel processing is also used for other SQL constructs, such as subqueries, set operations (union/difference/intersect), and other operations such as UPDATE, INSERT, and DELETE.

### Example of Parallel Queries

Using a database of ACCOUNT and BRANCH tables, this section describes three parallel execution strategies.

**Parallel relation scan**: The following query is divided into two task types: the coordinator task that returns the answer set to the application, and slave tasks (on a given partition of the ACCOUNT table) that select the matching rows and pipe them to the coordinator. The slave task has instances on all the nodes where the ACCOUNT table resides.

```
SELECT CUSTOMER_NAME FROM ACCOUNT
   WHERE BALANCE > 20000
```

Figure 3 shows the execution snapshot for this query. The data predicates are evaluated as soon as possible at the nodes where the data resides. This minimizes the amount of data exchanged using messages. Expressions and scalar functions are also "pushed down" and evaluated as soon as possible.

**Parallel aggregation, including GROUP BY clause**: This example shows how the DB2 Parallel Edition forces aggregation tasks to be

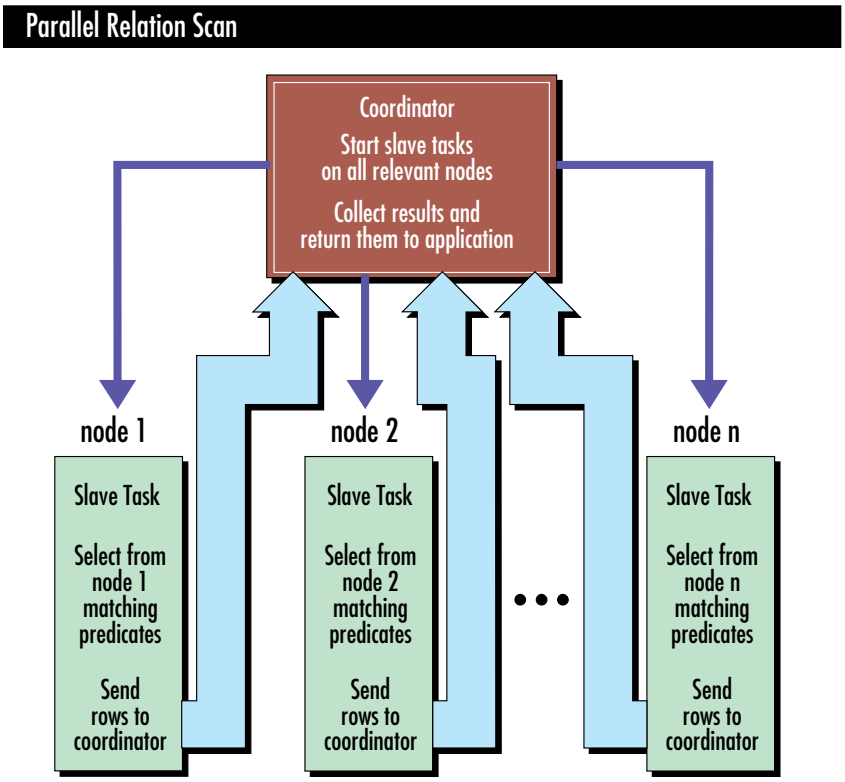



**Figure 3. The execution snapshot for the parallel relation scan**

executed at the site of the data, minimizing the sequential work at the coordinator task.

```
SELECT C_BRANCH, COUNT(*)
   FROM ACCOUNT
   WHERE BALANCE > 20000
     GROUP BY C_BRANCH
```

The execution strategy involves two task types:

- Slave tasks select matching rows on a given partition, group them according to C_BRANCH value, perform local aggregation (a local count for each group), and return the grouping column and the local count to the coordinator task.
- The coordinator task performs the global aggregation (a global count from the local count values) by merging the grouping values and local counts returned from the different slave tasks. It then returns the answer to the application.

**Co-located joins**: In the following example, both the ACCOUNT and BRANCH tables are partitioned on their joining columns.

```
SELECT BRANCH_NAME, CUSTOMER_NAME
  FROM ACCOUNT,BRANCH
  WHERE ACCOUNT.C_BRANCH = BRANCH.BRNO
  AND ACCOUNT.LAST_DATE > CURRENT DATE -
2 YEARS
```

One possible execution strategy consists of the following tasks:

- The slave task scans its partition of ACCOUNT table, applying the predicate. It then joins the resulting relation with its partition of BRANCH table. The slave task then sends the joined rows to the coordinator task.
- The coordinator task merges the results and sends the answer set to the application.

Typical decision-support applications include SQL queries that join relations in a usually predictable way. In those cases, it may be beneficial to partition the relations on their joining columns. This action can reduce the number of data exchange messages required for join processing.

**Redirected join**: If the ACCOUNT table is partitioned using the CUST_NO attribute rather than the C_BRANCH attribute, a co-located join strategy will generate incorrect results for the SQL query shown above. That is because a row of the ACCOUNT table may join with a row of the BRANCH table that resides on a different partition. Then, DB2 Parallel Edition could generate a redirected join strategy by hashing each selected ACCOUNT table row using the C_BRANCH value and redirecting them to the corresponding BRANCH table row location.

**Broadcast join**: DB2 Parallel Edition also uses a broadcast join where the selected rows of the ACCOUNT or BRANCH tables are broadcast to the nodes containing the partitions of the other table. This strategy is useful for an index-based join strategy or when the size of one of the joining tables is small.

**Redistributed join**: When neither the BRANCH nor the ACCOUNT table is partitioned on the joining columns, a redistributed join strategy would be used:

- ACCOUNT table rows are redirected by hashing on the C_BRANCH attribute.
- BRANCH table rows are redirected by hashing on the BRNO attribute.
- Redistributed partitions of ACCOUNT and BRANCH tables can then be joined locally.

One or more of the above join strategies can be viable, based on the data partitioning. The optimizer selects the strategy that minimizes the overall execution cost. The optimization strategy extends elegantly when more than two relations are being joined. For example, a multi-join query execution strategy might involve a mix of co-located, redirected, broadcast, and redistributed joins.

The parallel execution strategies in DB2 Parallel Edition are executed asynchronously. Coordinator involvement is restricted to initializing slave tasks and to collecting final result sets. In a multi-join query, there is no coordinator involvement between joins—all processing is driven by data flow between slave tasks.

## Parallel Utilities

The DB2 Parallel Edition provides linear scaleup and speedup for all utilities. Two important utilities that help to manage a database are the data load and rebalance utilities.

### Data Load

Data can be loaded into database tables using the DB2/6000 import utility or by using fast loading utilities such as Bridge Fastload or the DB2 optional fast-load program. The import utility loads data by performing `INSERT` operations. DB2 Parallel Edition provides an enhanced `INSERT` in which rows are batched before being sent to the destination processor. With the DB2 Parallel Edition `INSERT`, rows are batched by their destination processor and the batch is sent to that processor when the buffer is full. This new `INSERT` is available to most application programs that use SQL `INSERT` and coordinate restart points using `COMMIT`. It is implemented as an option at bind time. Most programs can use this option without any modification.

Another option for loading data is to directly create database files without using the `INSERT` mechanism. In the parallel database, the best approach is to partition the input data file based on the defined partition key values. This creates one file for each processor that will store the table. Next, each partition is loaded in parallel across the processors. An Application Programming Interface (API) that facilitates this process determines which processor will store a row of a table, based on its partition key value and its nodegroup name. This API can partition a data file into several files—one per processor.

### The Rebalance Utility

A system-defined hashing function determines the partitioning key value of a row in a table and the processor on which the row is stored. The processor selected is from the nodegroup in which the table was created. A uniform hashing function distributes data evenly across the set of processors in the nodegroup. Data may need to be moved between processors when a new processor is added to a nodegroup, or when the data distribution across processors is not uniform (because of the skew in the data values in the partitioning key column). The rebalance utility helps to achieve a uniform distribution of data.

The rebalance utility is specified at the nodegroup level. Rebalancing a nodegroup results in the rebalancing of all tables within that nodegroup. Rebalance is an online operation—neither the database system nor the database need to be shut down. Database administrators can specify which data should be moved and where it should be moved.

**The DB2 Parallel Edition provides linear scaleup and speedup for all utilities.**

For example, suppose a nodegroup consists of 10 processors, numbered 1 to 10. Two processors, numbered 15 and 16 are added. The rebalance utility can move some data to take advantage of the new processors. Data can also be moved from processors 1 through 5 to processor 15, and data from processors 6 through 10 can be moved to processor 16. This can achieve a uniform distribution of data across all processors in the nodegroup.

In another example, suppose a nodegroup consists of four processors numbered 1 to 4. Processor 1 has 40% of the rows, and the other three each have 20%. The rebalance utility can move approximately 15% of the rows from processor 1 and send about 5% each to the other three.

A large rebalancing operation can be broken into smaller steps by rebalancing only part of the data each time. For example, for 50% of the rows to be moved from one processor to another, the move could be done in five separate rebalancing operations, each moving 10% of the data. If large tables need to be rebalanced, the work can be spread over several intervals of low activity.

### Other Utilities

This section describes how several functions and utilities are executed in parallel.

**Index creation**: The parallel capabilities of DB2 Parallel Edition support the creation of

unique indexes, in which the key columns include the partitioning key columns or non-unique indexes. Creating indexes is done in parallel across processors. A local index is created for each partition of the table stored in a processor.

**Reclustering utility (REORG)**: The REORG utility reclusters the rows of a table on disk. Clustering of each individual table partition at a processor is sufficient to exploit clustered index scan performance. The REORG operation, which executes in parallel across all processors, can provide linear speedup with several processors.

**DB2 Parallel Edition can extend the capacity of an online transaction workload beyond a single RISC System/6000.**

**Backup/restore**: Each processor uses the DB2/6000 backup utility to back up the segment of the database that is resident at that processor. The backup image includes the processor number. Not all processors have to be backed up at the same time. Backups can be taken in parallel across processors, but this method requires sufficient backup resources, such as tape drives at each processor. All logs must be available at each processor to recover the entire database to a consistent point in time.

DB2/6000 parallel functions interface with the IBM ADSM product that manages backups from multiple RISC System/6000 machines. Since the ADSM interface tracks backups and their associated logs, it enhances the manageability of DB2/6000 in parallel environments.

**Forward recovery**: Except for the resolution of the final in-doubt transactions, all processors can perform parallel forward recovery by reapplying log entries after a system failure or a restore from backup. DB2 Parallel Edition ensures that all nodes are recovered to the latest log.

### Parallel Transaction Processing

DB2 Parallel Edition is a full-function parallel RDBMS. Since it is not limited to queries, it can be used to extend the capacity of an online transaction workload beyond a single RISC System/6000.

Combined with IBM's HACMP/6000, the DB2 Parallel Edition provides concurrent access to a database from all processors in an HACMP/6000 cluster of RISC System/6000s. The DB2 Parallel Edition function-shipping implementation is not limited to a single HACMP cluster—multiple clusters can be joined to support larger databases or higher throughput.

## Conclusion

The IBM DB2/6000 parallel database server can efficiently manipulate large amounts of data by partitioning data over several nodes and executing queries in parallel. It provides the following advantages:

- Cost-based optimization
- The best parallel processing strategies
- Efficient, asynchronous execution of subtasks

DB2/6000 also provides efficient transaction-processing capabilities and a suite of parallel utilities for database management.

**Gilles Fecteau**, IBM Software Solutions Laboratory, 1150 Eglinton Avenue East, North York, Ontario M3C 1H7, Canada. Internet: gfecteau@vnet.ibm.com. Mr. Fecteau is one of the key designers of the parallel version of the DB2 workstation products (DB2/2™ and DB2/6000). He has been involved in several advanced technology efforts to bring parallel databases to market. Mr. Fecteau has a Bachelor of Engineering Science degree from Laval University in Quebec City, Quebec.

DATABASE

# INFORMIX-OnLine's Dynamic Scalable Architecture

By Gregg A. Christman

This article describes how INFORMIX-OnLine is evolving to meet the technological demands of symmetric multiprocessing and massively parallel processing architectures.

Just as computer systems are evolving from uniprocessor to Symmetric Multiprocessing (SMP) and eventually to Massively Parallel Processing (MPP) architectures, INFORMIX-OnLine must also evolve to meet these technological demands. This article discusses the multiple phases of Dynamic Scalable Architecture (DSA) evolution as well as an alternative architecture that was considered, but later rejected in favor of DSA.

Figure 1 illustrates the three phases of OnLine's evolution as it relates to the changing computing environments. The OnLine phases include the following releases:

- INFORMIX-OnLine/DSA
- DSA/Parallel Database Queries (PDQ)
- DSA/Extended Massively Parallel (XMP) Processing

### Phase 1: INFORMIX-OnLine/DSA

The Dynamic Scalable Architecture, released in 1993, provides the foundation for the generations of INFORMIX-OnLine that follow. The new architecture provides a flexible threading architecture for both Online Transaction Processing (OLTP) and Decision-Support System (DSS) environments. In OLTP environments, a small number of server processes efficiently handle multiple user sessions. In DSS environments and batch jobs, a single-user session can efficiently spawn multiple threads that run in parallel, utilizing more of the hardware's resources.

The initial release of DSA took advantage of the inherent parallelism within the SMP architecture by using parallel database backups and restores and building indexes in parallel. True database parallelism came when DSA evolved to the next phase: DSA/PDQ.

### Phase 2: INFORMIX-OnLine/DSA with PDQ

DSA with PDQ, released in 1994, splits a single database operation into a set of parallel operations. Parallelization can dramatically reduce the processing times for both OLTP and DSS operations since multiple processors perform work for a single transaction.

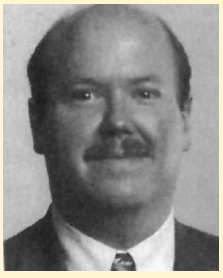
DSA/PDQ runs on tightly coupled SMP designs. This release is targeted at several computer environments including OLTP, DSS, Online Complex Processing (OLCP), and batch processing.

### Phase 3: INFORMIX-OnLine/DSA with XMP

DSA with Extended Massively Parallel (XMP) Processing is scheduled for release in 1995. DSA/XMP will run on loosely coupled and MPP architectures. DSA/XMP can take advantage of loosely coupled distributed cluster designs such as IBM's High Availability Clustered Multiprocessing/6000 (HACMP/6000).

## Architecture of DSA/XMP

DSA/XMP will enhance performance on loosely coupled architectures by taking advantage of multinode partitioning. It will process SQL functions in parallel across nodes or processors. Multinode partitioning enables the database server to place pieces of the same logical table on different



Gregg A. Christman

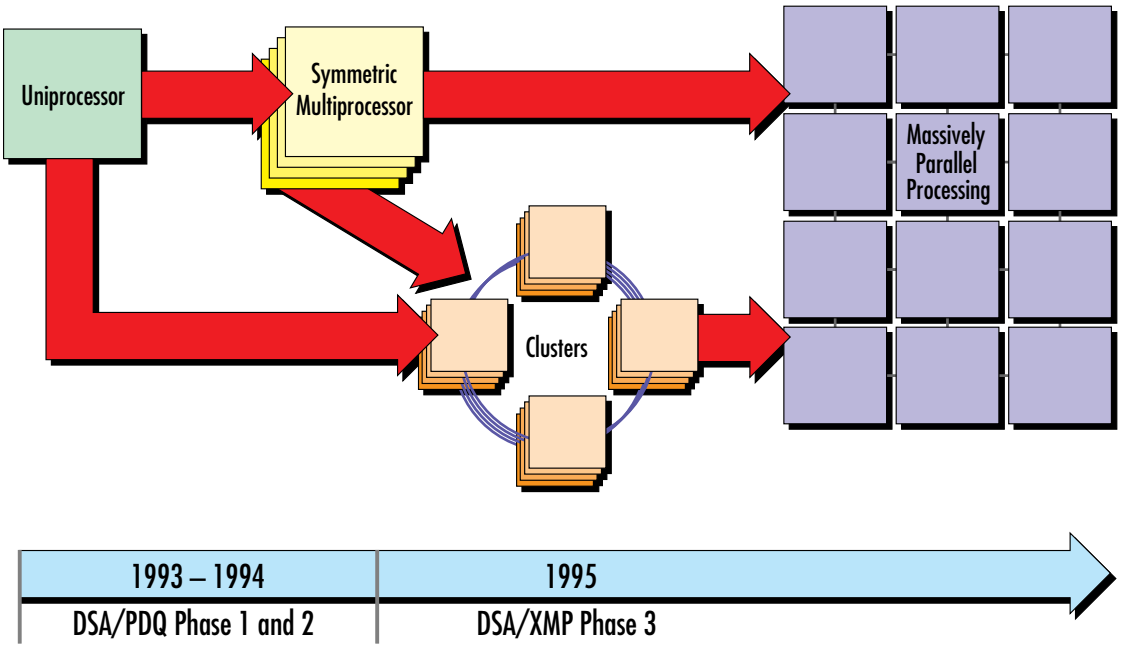



**Figure 1. Evolution of computing and DSA**

nodes. The partitioning is defined with the `CREATE TABLE` command and allows data to be placed on the various nodes based on a range of values (other partitioning methods are also supported). For example, a customer table being placed on a cluster with five nodes can have all customers with zip codes beginning with 1 placed on node 1, those with zip codes beginning with 2 on node 2, and so on.

Informix can parallelize operations such as scans and joins across nodes, which maximizes the use of the hardware for database processing. SQL requests can span many database servers and many nodes or processors. This scalability is achieved through DSA's dynamically configurable pool of database server processes called Virtual Processors (VPs).

### Virtual Processors

Rather than initiating UNIX processes to provide client access to the database, DSA spawns lighter-weight mechanisms called *threads* to execute user requests. Threads can execute in parallel as processors are available. The database server processes manage active threads and can effectively switch among them. The database server processes—the VPs—are similar to physical processes in this sense.

DSA contains a pool of VPs. Multiple threads can run concurrently on different VPs. Since all data is in shared memory, any VP can execute any thread. This allows threads to migrate across processors for load balancing.

For efficiency and tuning versatility, VPs are grouped into *classes* (Figure 2 shows examples). Each class is optimized for a particular function. Threads are transparently scheduled across the VPs in the relevant class. The VP pool can be adjusted online to accommodate periods of unusual activity or load mixes.

Figure 3 illustrates an example of DSA utilizing multiple CPU VPs to parallelize an SQL scan and join request. The steps shown in the figure are as follows:

1. The network VP receives the scan request.

| Class | Description |
|---|---|
| `cpu` | Performs SQL data manipulation for clients |
| `tlitcp` and `soctcp` | Processes incoming client requests |
| `aio` | Performs asynchronous I/O |

**Figure 2. Examples of VP classes**

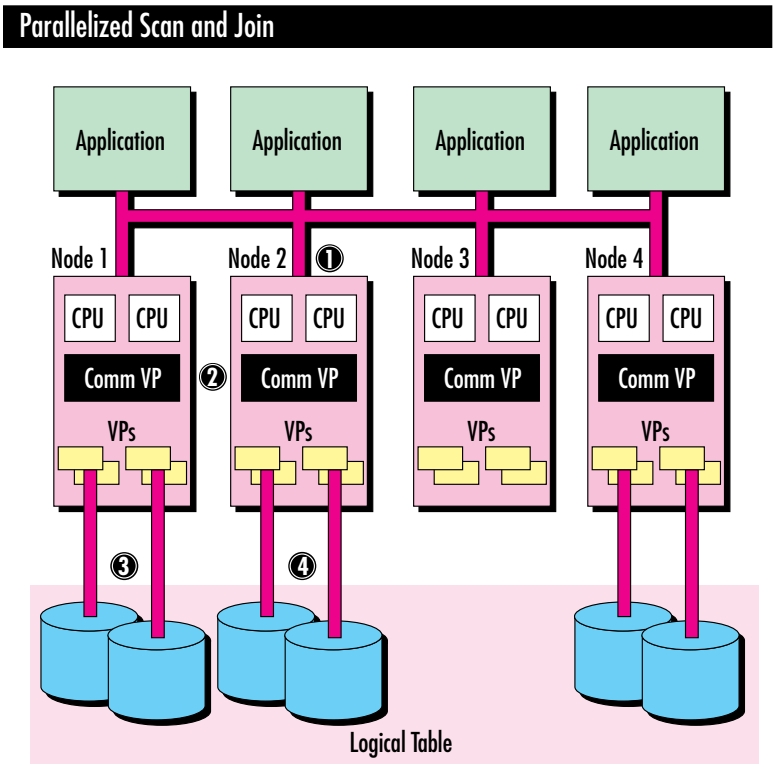



**Figure 3. An example of multiple VPs used to parallelize an SQL scan and join request**

2. The network VP passes the request to the database servers involved in scanning the data.
3. The CPU VPs on the appropriate database servers issue non-blocking stored-procedure or SQL requests in parallel. The requested data is retrieved from the pieces of the database table that reside on the node.
4. If required, the resulting data is joined in parallel and returned to the destination database server. No intermediate temporary files are required for a join operation.

On SMP nodes within the loosely coupled architecture, multiple CPUs on the same machine can be used in the parallel query.

### Database Partitioning Methods

Informix supports four different database partitioning methods. Each node can run its piece of the XMP server and own portions (partitions) of the table.

DSA/XMP allows a table to be partitioned across any combination of nodes within the system using any of these four methods.

**Range partitioning**: This is ideal for requests that are based on a range of values and are commonly accessed by one index key. DSA/XMP range partitioning can be based on either the index keys or the actual data. Range partitioning enables queries to be selectively sent to a subset of the nodes in the system, reducing the overall load on the system and improving throughput.

Figure 4 shows range partitioning for a customer table. Partitioning is determined by the first character of the customer's last name. Customers with last names beginning with A to F have records stored on part of the customer table on node one. The remainder of the customer records are stored on the second or third node.

**Round-robin partitioning**: Round-robin partitioning provides perfectly balanced partitioning. The price of perfect balancing is the cost inherent in sending queries to all nodes for applying against all partitions. As illustrated in Figure 5, the records are sent to a specific node based on the order in which the record was inserted.

Round-robin partitioning increases the overall system load. It can also decrease throughput since all the storage nodes will be accessed. For this reason, it is unlikely that round-robin partitioning would be used in heavy OLTP environments. The concept of round-robin partitioning is similar to disk striping in the filesystem.

**Hash partitioning**: Hash partitioning approximates the perfect population balancing of round-robin partitioning while enabling queries to access a single partition selectively. Since the DSA hash algorithm can quickly determine the node on which the required data resides, hash partitioning avoids the overhead associated with round-robin partitioning.

Hash partitioning is useful when the characteristics of the data values in the partitioning attri-

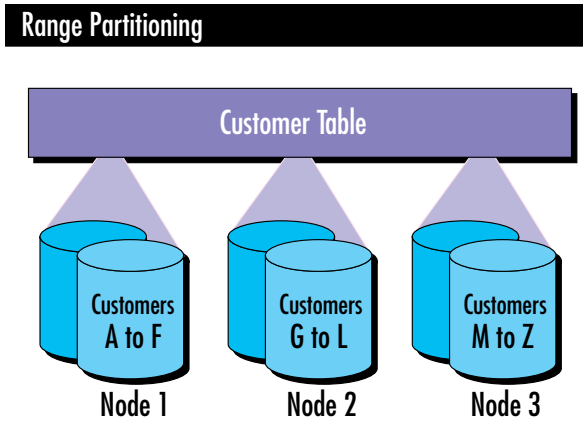



**Figure 4. Example of range partitioning based on the first character of the customer's**

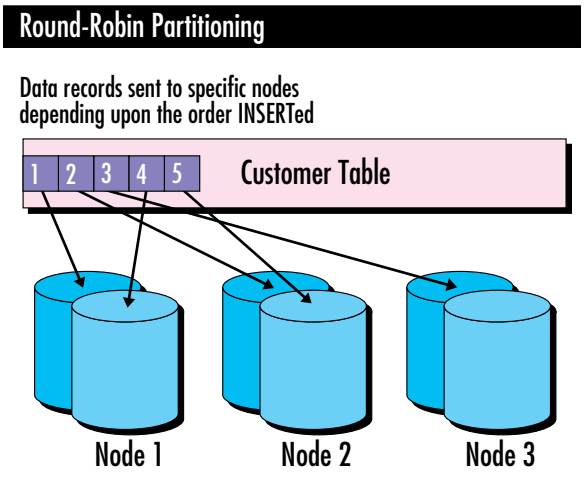



**Figure 5. Round-robin partitioning provides perfectly balanced partitioning**

bute are either not well known or dynamic by nature.

**Expression partitioning**: Expression partitioning, illustrated in Figure 6, enables queries to be selectively sent to a subset of the nodes in the system, similar to range partitioning. The partitioning is based on a predefined SQL expression (including the use of `or` expressions). The expression is evaluated and, based upon the results of the expression, the data is sent to the appropriate node.

## DLM: An Alternative to DSA/XMP

Informix considered using a Distributed Lock Manager (DLM), but later rejected it. A DLM allows a common database on shared disks to be accessed by multiple nodes in a cluster system.

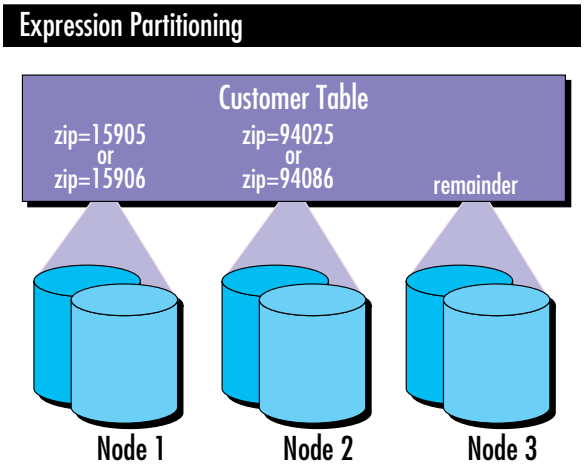



**Figure 6. Partitioning based on a predefined SQL expression**

Although the DLM is a centralized resource, which can be a problem, it is not the DLM concept that is inherently a problem. The problem was that with existing DLM-based database management solutions, the DLM is used to enable data shipping—data being sent from the shared database on disk to the local buffer cache of the node requesting the data. As shown in Figure 7, the node must then write the data back to the disk when it is requested by another node. In the worse case, data is shipped from one cache to disk, then from disk to another cache.

Lock latency is an issue with DLM since the system will typically require two milliseconds for buffer latch message passing. As more nodes are added to the system, data shipping via the I/O subsystem becomes a bottleneck and performance degrades.

Within a small cluster, scalability of a DLM-based solution might be adequate if the data being requested is partitioned appropriately across the nodes to reduce lock contention and I/Os. Scalability would be achieved through application-level partitioning of access; that is, the application is structured so that access to a given subset of data is restricted to one node, particularly for updates. Using the TPC-A schema (BRANCH, TELLER, ACCOUNT) as an example, applications would be structured so that all

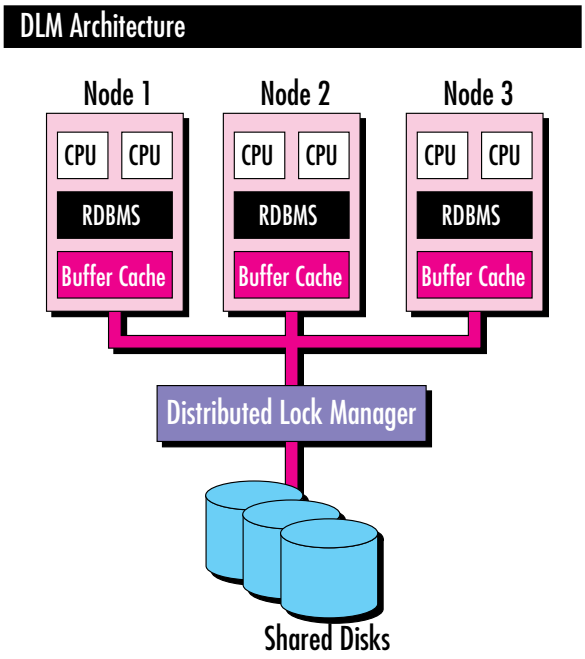



**Figure 7. Distributed Lock Manager architecture**

requests for a given BRANCH go to the same node.

The application-level partitioning would be required to achieve effective database performance. This is because each node is restricted to accessing a subset of data, and DLM requests and data shipping are minimized. But this forces an unnecessary burden of complexity upon the users, because in practice, it can be difficult to partition applications. For many applications, a physical database design in which data access can be restricted to one node is not possible. Also, applications should not be required to know how data is partitioned (for data independence). Even if the application developer had this knowledge, it can be difficult to decide which node to access for best performance.

**DSA was designed to use the breadth of open systems hardware.**

Scalability of a DLM solution that uses data shipping becomes an issue because as the number of nodes increases, the data-shipping problem can easily compound. The additional throughput gained by adding another node to the database cluster decreases or becomes negative. If application partitioning is not being done, data shipping is required. As nodes are added, internode DLM traffic increases and more data must be shipped between nodes—using up interconnect bandwidth. Once the interconnect is saturated, throughput cannot be increased by adding more nodes.

Transactions also wait longer for data. The problem negatively affects data that is frequently accessed. In some situations, there could be so much waiting and interference between nodes that throughput can decrease by adding an additional node.

## DSA/XMP For Database Performance

The distributed database partitioning approach provides superior scalability over the DLM approach. Interference (contention among processes) is kept low by minimizing resource sharing. The volume of data shipped across the network is minimized since only questions (SQL functions, parallel scan operations, parallel join operations) and answers (resultant data) are moved through the network.

DLM-based databases typically require many messages to be generated (to coordinate the buffer caches and locking) to retrieve the page. DSA/XMP filters out unneeded data before the results are returned. Since there is no need to manage distributed locks or segmented buffer caches, only the resultant data is moved across the network.

Raw memory accesses and raw disk accesses are performed locally in a processor. Only the filtered (reduced) data is passed to the client program. Minimizing traffic on the interconnection network allows more nodes to be added to the system without performance degradation, making it a more scalable design.

DSA/XMP distributes function to where data resides, rather than shipping the data to where the work needs to be done. Distributing the function provides for parallelism and reduces data shipping. This algorithm will run well with modest amounts of memory, achieving near linear speed-up. It will also take advantage of additional memory when available.

DSA was designed to use the breadth of open systems hardware, available from uniprocessor to SMP to loosely coupled clusters and MPP architectures. This is unlike alternate technologies, such as DLMs, which create inherent scalability bottlenecks that could prevent RDBMS users from fully utilizing the power of their computing environments.

**Gregg A. Christman**, Informix Software Inc., 4100 Bohannon Drive, Menlo Park, CA 94025. Mr. Christman is the manager of the Marketing Analysis Group at Informix. He provides research about the database server industry and how to keep Informix's database server products at the leading edge of that industry. Previously, Mr. Christman had managed the INFORMIX-OnLine database server product line. He has been with Informix for over seven years, starting in the Technical Support Group and later serving as the senior product specialist for INFORMIX-TURBO. He holds a degree in Engineering Psychology from the University of Pittsburgh.

# Oracle Parallel Technology Empowers AIX Systems


By Sandra Lee and Annie Chen

This article describes how Oracle's parallel technology fully exploits the power of IBM's HACMP/6000 software and SPx hardware (IBM SP1™ and SP2 RISC-based parallel processors). By using parallel database functions—from database backup and restore to the query process itself—the Oracle7 database builds on the strengths of IBM architectures to improve database performance. Although processing is shared across multiple CPUs, users still manage one database with parallel functions for better performance.

Today's businesses need to harness enterprise-wide data. Computing environments are being pushed to their limits by business applications that are growing larger and more complex, requiring more sophisticated computer systems with greater capacity. Many users eliminate data because they cannot economically store it for later access. Other users base their business decisions on data that is days old because their current systems do not have the horsepower to provide real-time analyses.

Oracle has created new solutions to meet these business needs, which fall into three general areas of computing: data mining (queries) on huge amounts of data, high availability, and improved backup/restore and database performance.

Not only does Oracle support different IBM architectures, but Oracle's parallelism addresses the business needs discussed above and enhances performance in all three areas of computing. For example, the Oracle Parallel Server provides high availability in an HACMP/6000 cluster; in an SPx system, the Parallel Server also provides parallel query ability, enhancing the IBM technology. Oracle's parallel technology is a logical next step for companies that need to enhance their computer systems to better meet their business needs.

## Oracle Parallel Server

The Oracle7 Parallel Server (OPS) technology has been implemented on IBM's HACMP/6000 and SPx systems. With OPS, multiple instances of the Oracle database server run concurrently and independently. Each runs on its own processing node with its own Shared Global Area (SGA) memory for a database buffers cache and its own set of Oracle7 background processes (including the system monitor, process monitor, and database write) for backup and recovery. Any number of Oracle7 instances can access the same Oracle7 database files and control files on disk, which collectively form an Oracle7 Parallel Server. In Symmetric Multiprocessing (SMP) systems, IBM plans to accomplish this through shared memory; in HACMP/6000, it is done through shared disks in clusters up to four processors; and in the SPx family, it is done through the high-performance switch.

Figure 1 illustrates the Oracle Parallel Server architecture. Each Oracle7 instance concurrently handles the database requests of multiple clients. All Oracle7 instances can execute transactions concurrently against the same database, enabling users to focus the processing power of multiple CPUs (whether loosely or tightly coupled) against the database.

### Benefits of Single-Database Design

Oracle's single-database design (as opposed to multiple databases or multiple database partitions) scales up system performance by simply adding nodes and disks. Since all nodes have direct access to the entire common database (instead of

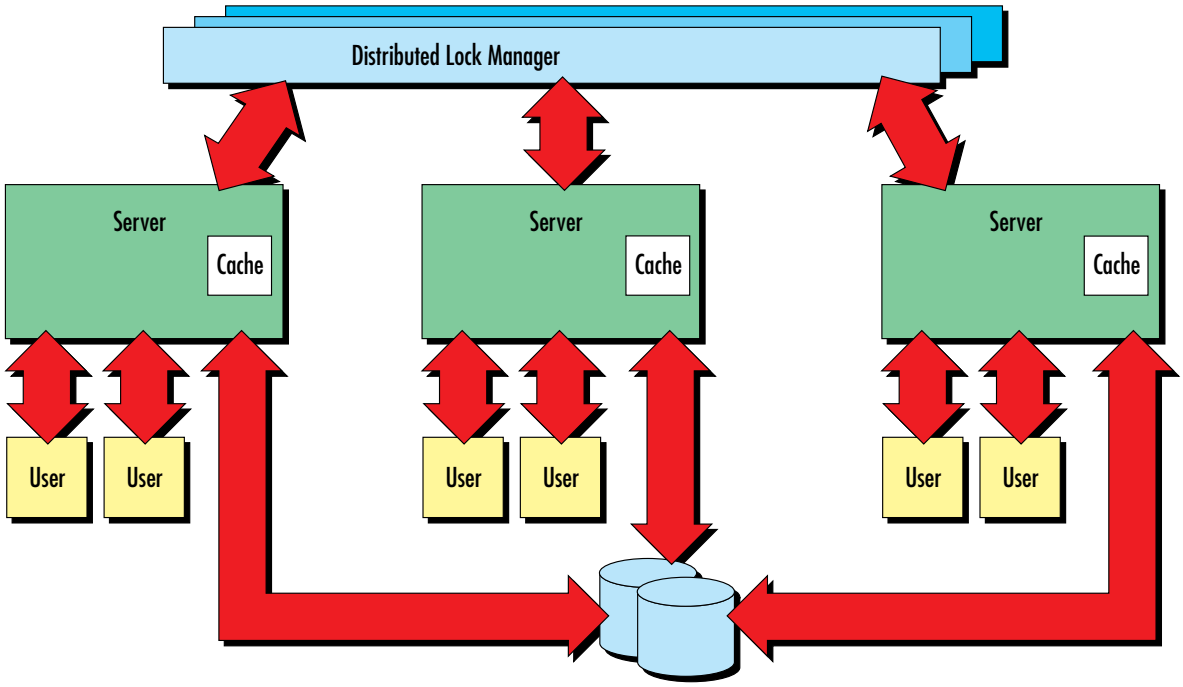



**Figure 1. Oracle Parallel Server architecture**

each node having exclusive access to a small partition of the database), adding nodes immediately increases system throughput. Because data does not need to be repartitioned across new numbers of nodes and no applications need modification, the administrative work involved in scaling up performance is much simpler. The single-database design further simplifies database administration since there is only one database to start up, shut down, back up, and monitor.

### Easy Migration from Uniprocessing to HACMP/6000

Oracle's parallel database environment completely hides its parallelism from users. Since the OPS software uses the identical SQL interface in every Oracle7 database system, the user sees and interacts with the familiar standard Oracle7 database server. Parallelism requires no new commands or extensions to existing commands; the OPS technology handles parallelism and optimizes system resource utilization automatically. The results are twofold: all Oracle tools and applications run unchanged, and neither application developers nor end users need retraining.

### Increased Availability

The single-database architecture results in increased data availability. Because each node has full access to the entire database, losing a node does not mean losing access to a part of the database. If a node fails, one of the surviving nodes will automatically detect the failure, recover any work that was in progress on the failed node, and continue processing all client requests. Oracle7 currently supports all modes of HACMP/6000, including concurrent access. Figure 2 shows an HACMP/6000 cluster.

In an HACMP/6000 cluster running in concurrent access mode, the OPS enables different nodes in the cluster to share an application. This means that the HACMP/6000 cluster can run larger Oracle applications than a single RISC System/6000 can handle. This capability breaks the single machine barrier for Relational Database Management System (RDBMS) performance, achieving scalable high performance for many types of applications. Tests have shown that the increase in performance on two-node HACMP/6000 clusters with Oracle7 and HACMP ranges from 1.6 to 1.9 times the performance of the uniprocessor configuration, depending on the application.

## Exploiting the Power of the SPx Family

Oracle plans to take high availability one step higher in IBM's massively parallel SPx machines. In an SPx system, the nodes can be divided logi-

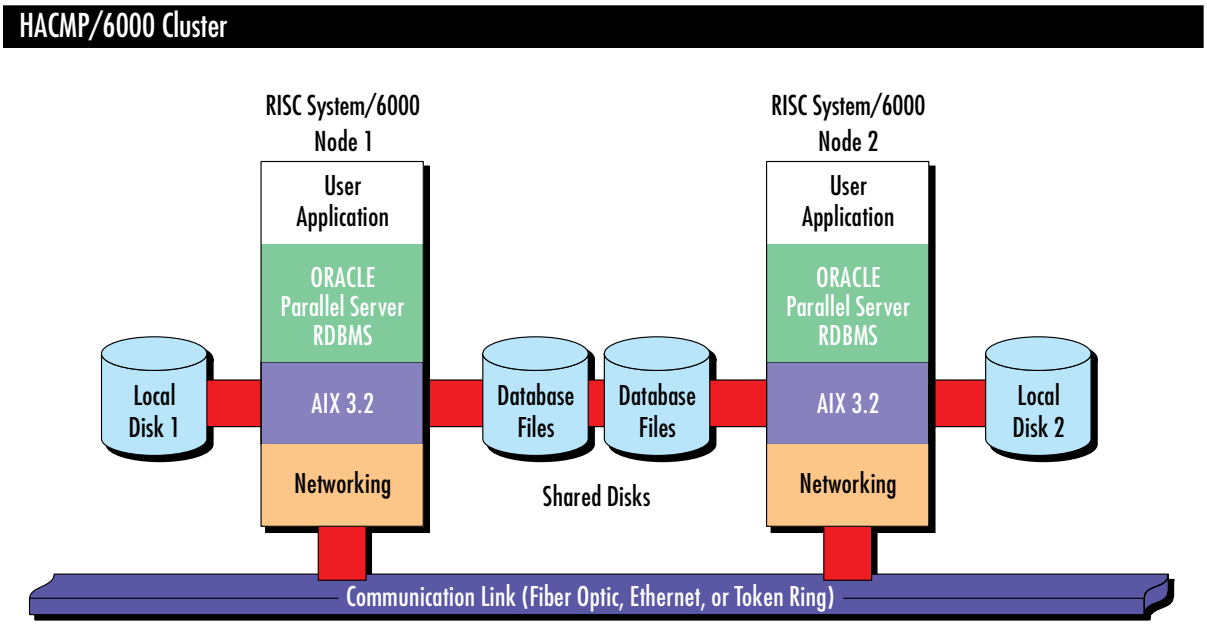



**Figure 2. HACMP/6000 cluster**

cally into two categories: data servers that host the Oracle database, and application/compute servers where applications reside. An instance of Oracle7 can run on each of the 8 to 512 data server nodes. All instances of Oracle7 have access to one physical database that is located on the SPx's Virtual System Disk (VSD). Any user or application can access any portion of the database from any instance or node of the machine with the same response time. All instances in a parallel server share the data files and control files, but each instance has its own redo log files (known as *threads of redo*).

If an Oracle7 instance or an SPx node fails on an SPx system with dual disk access, the other instances continue to access all portions of the database. First, the SPx will automatically recover the node or the VSD component and then pass control to Oracle7, which will automatically recover on behalf of the failed instance. Another Oracle instance ensures that data committed by the failed instance is written to the database and all uncommitted transactions are rolled back.

### Parallel Cache

To further improve performance in an HACMP/6000 cluster, the OPS uses parallel cache management enabled by the cluster's Distributed Lock Manager (DLM). As shown in Figure 3, each node in a cluster has a local cache containing database blocks that have been recently accessed by transactions running on that node. After execution, the blocks are retained in the cache so subsequent transactions can readily access them. Database blocks are removed from the cache to make room for new blocks using a Least Recently Used algorithm. When a node requires a database block that is not already in its cache, it uses the DLM and parallel cache management facilities to determine whether the block is in the cache of another node. If it is, the parallel cache manager coordinates the fast transfer of the database block from the other node. If no other node has the

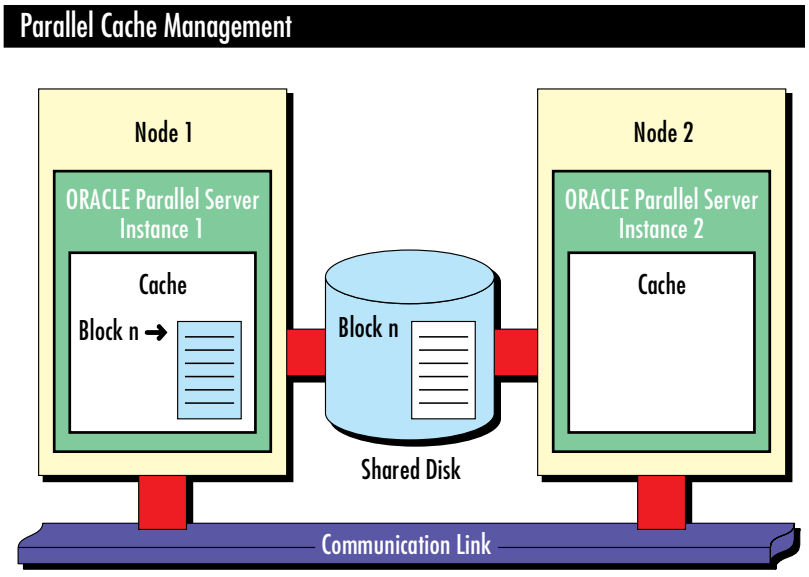



**Figure 3. Optimizing parallel cache management**

IBM SPx and Gigacache Nodes

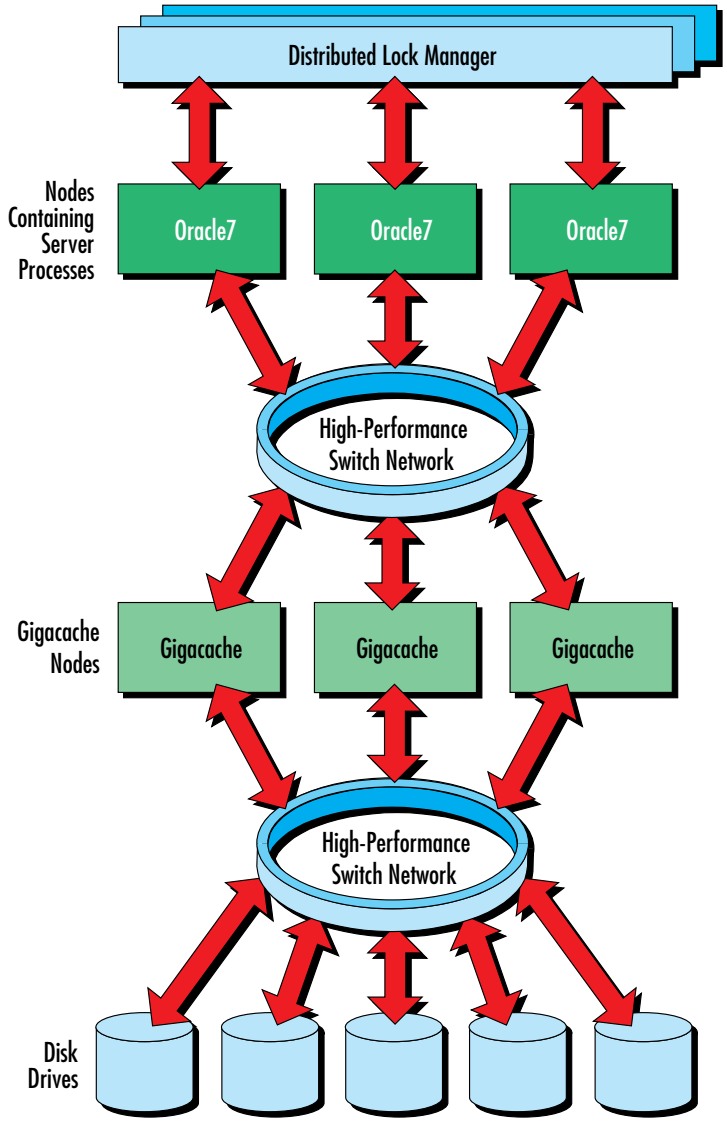



**Figure 4. IBM SPx and gigacache nodes**

database block, it is read directly from the database.

To increase performance and reduce overhead, the OPS does not release DLM locks at the end of a transaction. It holds the lock until another node requests a DLM lock on the same block. In business applications, especially with disjoint sets of data, the probability of an instance reusing a block that it just accessed is much higher than the chance of another instance asking for the same block. This parallel cache management technique makes the OPS very fast and efficient.

On an SPx system, the OPS speeds parallel cache management even more by reducing the amount of time needed for one instance of Oracle7 to obtain a database block from the cache of another node.

Some SPx processing nodes, each with its own dedicated memory, can function optionally as I/O nodes or *gigacache* nodes—a distributed random-access-memory disk cache that holds some or all of the database being updated or queried. Oracle7 can establish default configurations that designate certain nodes as servers and others as gigacache, as shown in Figure 4. These gigacache processors can reduce the read/write delays imposed by the database block transfer for cache coherency that would otherwise require disk read. In real world applications that frequently access data "hot spots," the gigacache can drastically reduce I/O requests from the parallel cache manager.

### Parallel Query

Cache speed is not the only area in query processing that may affect response time. A query on two 500,000 row tables may ultimately provide a single-row answer. But in the computation stage, the end result requires a complex sifting through the tables and performing sorts, aggregates, and joins. To speed this process, operations such as sorts, scans, and joins are parallelized with Oracle 7.1's Parallel Query Option (PQO) for SPx. Different nodes and instances of an SPx can work on different operations in parallel and send the results back to the query coordinator, which resides on a separate node (see Figure 5). Since all the data resides in a central database, the Query Execution Plan can be dynamic, based on the data accessed. This is unlike other RDBMS implementations, which statistically fragment the database when it is created.

Oracle's unique single database approach enables the definition of processor node classes on an SPx. One set of nodes can be defined to process Online Transaction Processing (OLTP) application requests while a different group of nodes processes complex queries. Since each node can directly access any portion of the entire database and execute instructions independently of other nodes in the system, the nodes running complex queries will not drain CPU processing power from the OLTP nodes.

Oracle's PQO is composed of parallel scan, parallel join, and parallel sort technologies that enable multiple processor nodes to automatically share the workload of a single, large, complex

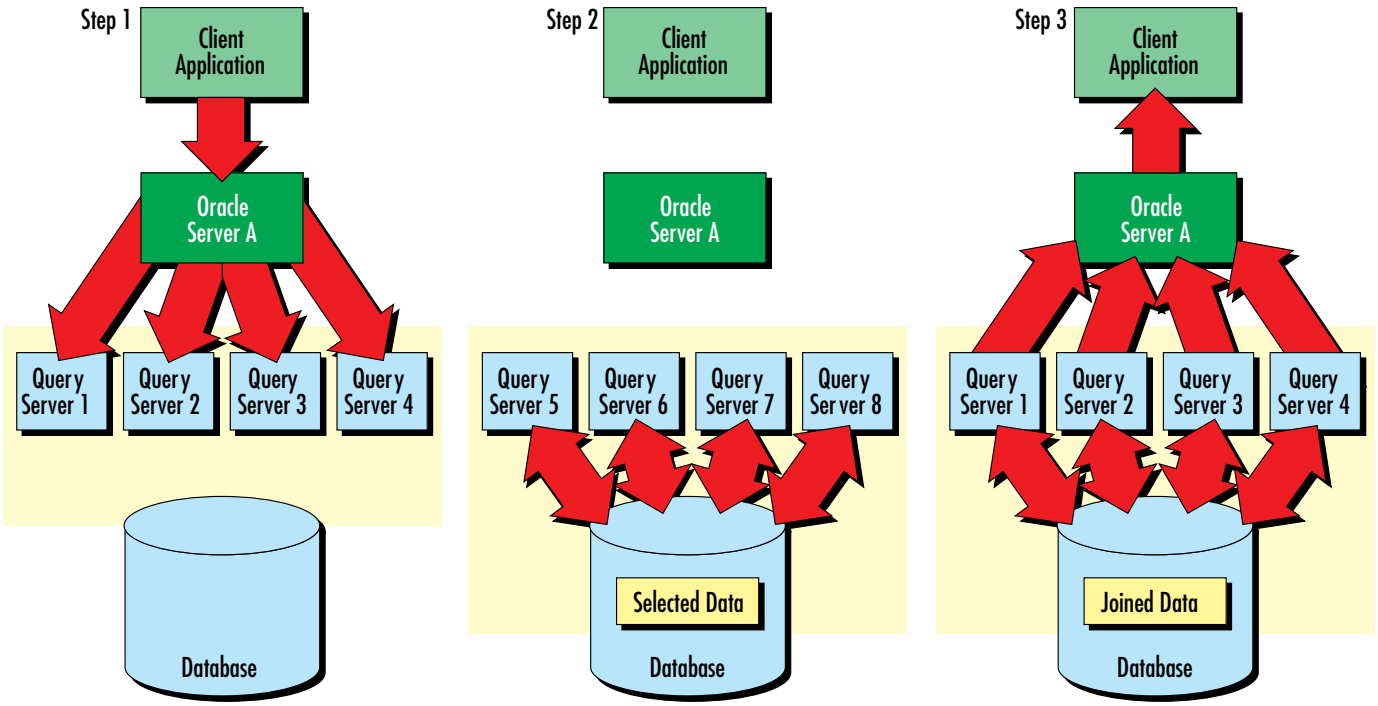



**Figure 5. Oracle's parallel query in action**

query. With parallel scan, the nodes work in parallel to search through different portions of the database, reducing the work for each node and improving performance. Parallel join enables multiple nodes to scan separate data tables, and then join the selections in parallel to quickly provide the final answer. Parallel sort divides the data into multiple pieces so that each node can sort a small portion of the data. The nodes then work together to quickly combine the smaller sorted lists into one sorted final result.

In the screens in Figures 6 and 7, the benefits of PQO are clearly visible. An Oracle7 database was run on an SPx system with eight nodes—first without the PQO, then with the PQO. The graphical front-end in the screens depicts the results of the experiment pictorially. The clock indicates the running time of the query, while the bar graphs show the CPU load plus I/O (plotted from system variables) for each node in the SPx.

When querying a database sequentially, one CPU must often wait for another CPU to finish processing the initial portion of the query before it can process its portion. Since only some CPUs are used during a query, the system is not used

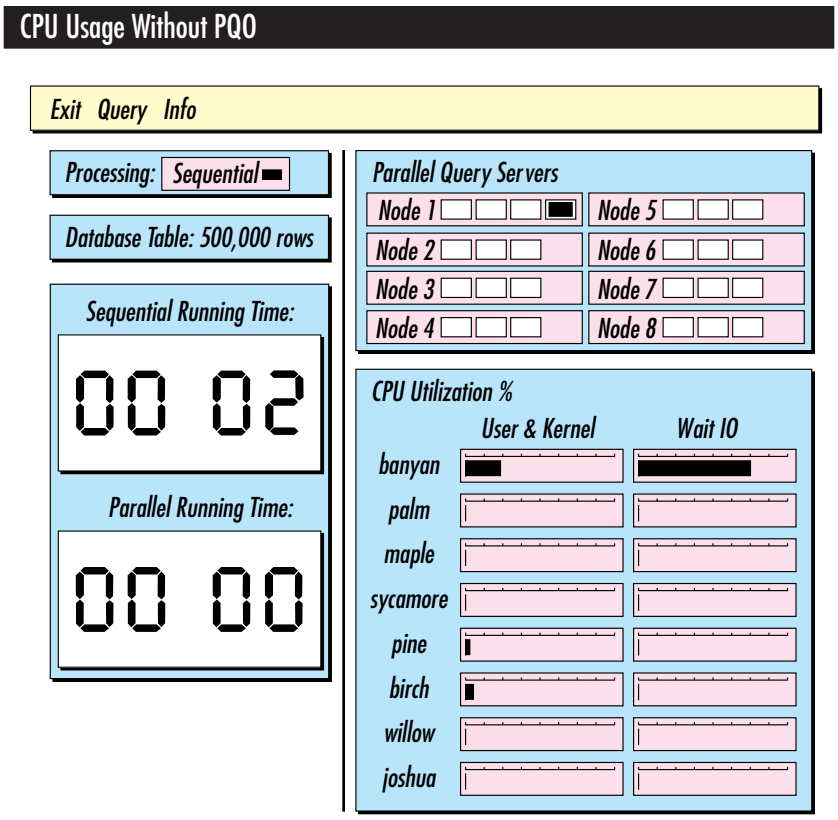



**Figure 6. CPU usage without Parallel Query Option**

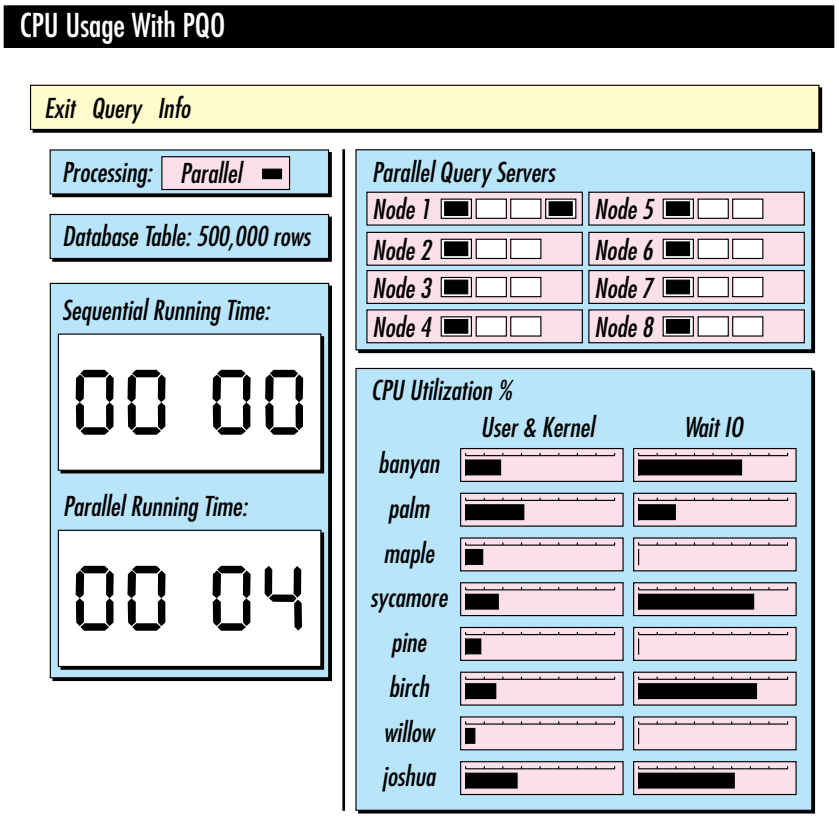



**Figure 7. CPU usage with Parallel Query Option**

to its capacity, resulting in inefficiency and slower performance.

Figure 6 shows CPU usage during sequential processing of a query. Only a few CPUs are used (banyan, pine, birch); the rest are sitting idle (palm, maple, sycamore, willow, joshua). The query is not taking advantage of all the available CPU power.

Figure 7 shows the result when the PQO is enabled. A query is split into several parts, and a separate CPU works on each part. The data is scanned and sorted in parallel, using all CPUs and decreasing response time, making more efficient use of the available processing power and increasing performance.

Parallel scan, parallel sort, and parallel join deliver exceptional query performance on massively parallel systems, such as the SPx, because of the many processors available to work on the query.

### Parallel Load and Parallel Index

For bulk loading of external data into tables of a database, Oracle 7.1's Parallel Direct Load enables users to start multiple concurrent load processes directed to the same table from multiple processing nodes. Parallel Direct Load uses many resources (such as disks and tape drives) to load data into the same table in parallel from different tape drives. Once the data is loaded, index creation can be divided across several data server nodes.

Like PQO, Oracle's parallel index feature will automatically use parallelism if the base table is spread over multiple database files across processing nodes. After the parallel table scan on the database files of the base table, the base table is split into partitions. Each partition contains the keys from the base table for a disjoint portion of the entire index key range. An index is created on each partition simultaneously, and eventually these indexes are merged into a single permanent index.

### Parallel Backup and Recovery

The Oracle Parallel Backup/Restore utility and Oracle 7.1's parallel recovery can drastically reduce the time to back up, restore, and recover very large databases—tens to hundreds of gigabytes in size. Available on IBM's SP2 with Oracle 7.1, it will enable multiple data files and tablespaces to be backed up online to different media devices in parallel. The utility works with a third-party media management tool to enable a data center to back up the enterprise—including the Oracle data servers, compute servers, filesystems, and clients—with the same media management software. This is a robust alternative to less reliable, slower utilities, such as `dd`, `cpio`, and `tar`. Once a restore is done with the Backup/Restore utility, Oracle 7.1's parallel recovery feature can be applied, providing a read-ahead capability of the log files in parallel.

**Sandra Lee**, Oracle Corporation, 500 Oracle Parkway, Box 659406, Redwood Shores, CA 94065. Internet: shlee@us.oracle.com. Ms. Lee is the product line manager for the AIX platform at Oracle. She has BA degrees in Computer Science and English from the University of California at Berkeley and a graduate degree from Boston University.

**Annie Chen**, Oracle Corporation, 500 Oracle Parkway, Box 659406, Redwood Shores, CA 94965. Ms. Chen is a development manager in the UNIX products division. She has an MS in Computer Science from the University of Pittsburgh in Pennsylvania.

# DCE With HACMP/6000


By Jim Wade

This article describes how to increase the availability of DCE services by using the built-in replication feature and by running DCE in an HACMP/6000 environment. The article assumes the reader has experience with HACMP/6000 and DCE.[1]

The Distributed Computing Environment (DCE) is a set of services and tools that supports creating, using, and maintaining distributed applications in a heterogeneous computing environment. DCE provides the following services for distributed applications.

**Threads**: The DCE threads layer provides the basis for all DCE services. Threads allows an application to create, manage, and synchronize multiple concurrent tasks within a single process. With threads, an application server can respond to multiple clients simultaneously; and conversely, a client application can use multiple servers. DCE threads is based on the POSIX™ 1003.4 Pthread specification.

**Remote Procedure Call (RPC)**: The DCE RPC facility provides tools and runtime services that extend the local function-call mechanism across a network. The RPC Interface Definition Language (IDL) compiler generates the stub code necessary for packaging the arguments and handling the calls to the server functions over the network.

**Security Services**: The DCE Security Services provides secure communication and controlled access to resources in a distributed system. The Security Service has a user registry, a login facility to initialize the user's environment, and Access Control List facilities to control access to resources.

**Cell Directory Service (CDS)**: The DCE CDS provides the central repository for information about resources in a DCE cell. The CDS manages a database of information stored by DCE or RPC-based services.

**Distributed Time Service (DTS)**: The DCE DTS servers provide synchronized time for the computers in a DCE cell. The DTS also has a set of library routines to convert and calculate times from several time formats including Coordinated Universal Time (UTC).

A DCE *cell* is an administrative grouping of machines in a network that share the same Security and CDS servers. The DCE cell relies on the CDS to register services and applications, and on the Security Service to authorize access to the services. These two servers are critical to the DCE cell because they must be up and running for the DCE cell to operate.

## Replicated Services in DCE

The CDS and Security Services must be highly available since other services depend on them. CDS achieves this goal by replicating directories and providing ways to keep copies of data consistent. There are two types of replicas in the CDS namespace. The *Master Replica* is the writable replica in which any CDS updates are made, such as registering a server or creating a new entry. The *Read-Only Replica* is a copy that supports only lookup and read operations. All write, create, and update operations must be performed on the Master Replica.

The Security Service provides replicated services by creating slave (read-only) replicas of the Security server. All security updates are made to the master server database, and the master server propagates the changes to the slave servers listed in its replica list. Examples of updates include adding a new principal such as a new server or account for the user to the registry, or refreshing a server key. Examples of read-only access

Jim Wade

[1] This article is based on working with DCE Version 1.2 and HACMP/6000 Version 1.2.

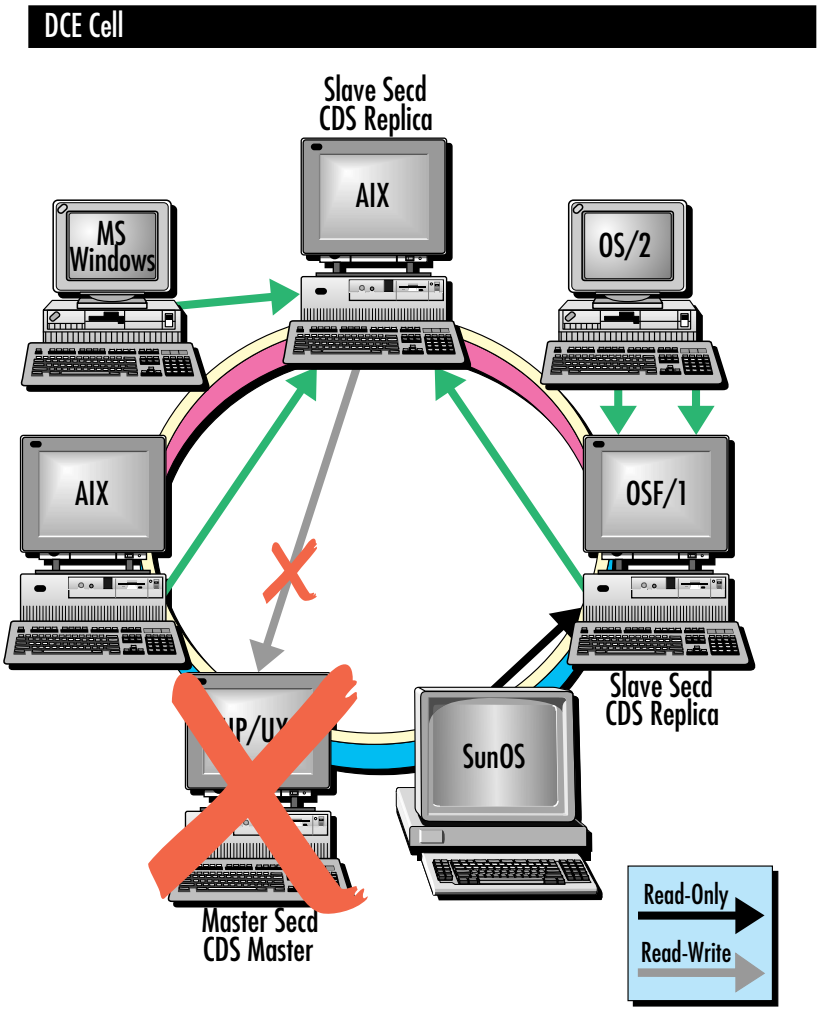



**Figure 1. DCE cell with failed master server**

include a user logging in (using `dce_login` or other utilities) to get the DCE network credentials.

In a normal running cell with replicated Security and CDS servers and several clients, read requests are directed to any servers on the network, as shown in Figure 1. If either of the master servers—CDS or Security—are down, operations such as logging in or locating a server on the network are not affected.

No additions or changes can be made to the Security User Registry database if the Security server is down. Also, DCE servers such as the CDS server will not be able to manage the server keys used to encrypt the tickets for secure communications.

The CDS server refreshes its key every two hours. This key is used for secure RPC communication between the CDS servers in the cell. Update propagation for replication may not be possible during this time.

The application server will fail to start if the CDS server that contains the master replica for a directory registers itself in CDS. If the complete namespace is not replicated to a clearinghouse in a secondary server, that information will not be available for clients. Clients would not be able to locate services that are already running until the CDS server containing the directory is brought back online.

## Recovery Methods

Converting a secondary CDS server to a master CDS server requires the complete namespace to be replicated to the secondary server. The `cdscp` control program makes the conversion relatively simple. An administrator must change the secondary server to be the master and exclude the old master server from the replica list. After the previous master server comes back online, it can be added as a secondary server or converted back to be the master server.

Converting a secondary Security server to a master server is more complicated. The database files from the master Security server directories must be copied to the secondary server machine. The administrator can use the `sec_admin` command to convert the secondary Security server to become the new master server.

Both scenarios require a DCE cell administrator to convert a secondary server to the master server. This can be quite a problem in an operation that runs 7 days a week, 24 hours a day (typically referred to as 7x24). In this situation, a knowledgeable administrator must be on hand or be called in to get a master server available. By using HACMP/6000, you do not need an administrator on call.

## High Availability Clustered Multiprocessing

HACMP/6000 is a clustered environment of loosely coupled RISC System/6000s running AIX. It supports high availability through shared resource access. HACMP/6000 controls access to the common resources and enables recovery after failures.

With careful planning, AIX DCE can be configured to run in the HACMP/6000 environment. This allows the DCE CDS or master Security server to recover without intervention from an administrator. HACMP/6000 will not ensure availability if the server process fails or is killed because HACMP/6000 is set up to recover only disks, machines, and network adapters.

Currently, AIX DCE supports HACMP/6000 recovery only in a rotating standby or hot-standby

mode. That is because both IP address and hostname takeover are required for the DCE CDS and Security servers to be reliably restarted on a different machine from the one on which they were originally configured.

## Implementing DCE with HACMP

HACMP/6000 should be configured before installing DCE. For the DCE servers to be recovered in the HACMP/6000 environment, the HACMP/6000 environment must take over several filesystems used by the DCE servers to store their on-disk databases and information. The filesystems used by DCE include `/krb5`, `/var/dce`, and `/etc/dce`.

Four files that are stored in the `/etc` filesystem must be copied to a filesystem that is shared between the HACMP/6000 machines. These files, `/etc/rc.dce`, `/etc/dce_cf.db`, `/etc/rc.dts`, and `/etc/mkdce.data`, are modified by the DCE configuration tools. They should be copied from the HACMP/6000 scripts into the `/etc/dce` filesystem while HACMP/6000 is stopped, and to the local filesystem before starting DCE.

DCE should not be started when the system is rebooted since TCP/IP must be running before the DCE services can start (HACMP/6000 starts TCP/IP and then DCE). HACMP/6000 provides a set of configuration scripts for starting applications. Figure 2 shows two portions of the `node.servers` script that have been modified to save and restore the DCE configuration files, and to start and stop DCE.

Network interfaces should be considered when configuring DCE in an HACMP/6000 environment. For example, consider an HACMP/6000 configuration with two Token-Ring adapters (one active interface and one standby interface) and a serial connection for HACMP/6000 control. The DCE services use only the active adapter. System administrators must mask out the use of the second Token-Ring interface and the serial interface by using the DCE environment variable `RPC_UNSUPPORTED_NETIFS`[1]. This variable can be put into the `/etc/environment` file so that any DCE server programs will have these addresses masked out. The correct usage would be as follows:

```
RPC_UNSUPPORTED_NETIFS=sl0:tr1
```

**With careful planning, AIX DCE can be configured to run in the HACMP/6000 environment.**

```
Function start_servers()
.
.
.
#================Fill In Here================
cp /etc/dce/dce_cf.db.save /etc/dce_cf.db
cp /etc/dce/rc.dce.save /etc/rc.dce
cp /etc/dce/rc.dts.save /etc/rc.dts
cp /etc/dce/mkdce.data.save /etc/mkdce.data
sh /etc/rc.dce
#============================================
.
.
.
Function stop_servers()
.
.
.
#================Fill In Here================
sh /etc/dce.clean
cp /etc/rc.dce /etc/dce/rc.dce.save
cp /etc/dce_cf.db /etc/dce/dce_cf.db.save
cp /etc/rc.dts /etc/dce/rc.dts.save
cp /etc/mkdce.data /etc/dce/mkdce.data.save
#============================================
.
.
.
```

**Figure 2. Code fragment from HACMP/6000 node.servers file**

## Conclusion

Configuring the DCE CDS and master Security servers to use their built-in replication mechanisms results in continued operation, even if one of the servers goes down. Combining DCE replication services with HACMP/6000 provides the highest availability for DCE services without administrator intervention.

**Jim Wade**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Wade has been a member of the DCE development team since the Open Software Foundation® DCE integration project. He has ported the CDS, Time, and Security components of DCE to the AIX operating system. He has held several project lead positions for the AIX DCE licensed program products and worked as a consultant with customers who use DCE and HACMP/6000.

[1] To enable this environmental variable, a PTF is needed. This PTF is implemented in AIX DCE 1.2 PTF 423300.

DATABASE

# SYBASE for HACMP/6000: An Architected Approach to Clustered Systems

By Josh Bersin

This article discusses Sybase's product and plans for clustered and parallel systems, including Sybase's current and future support for IBM's HACMP/6000. It also compares using SYBASE for shared-disk clusters like HACMP/6000 to the shared-nothing environment such as the IBM SP2.

As client/server computing becomes a key architecture for enterprise applications, we see the following key trends in server systems:

- **Very large databases**: It is now typical for customers to have databases in the tens of gigabytes, growing to hundreds of gigabytes and beyond.
- **Hundreds to thousands of online users**: As the amount of data grows, so does the number of users who need to access that data. The SYBASE architecture design assumes that these client users are accessing the system through a network.
- **Demand for Online Transaction Processing (OLTP) performance**: Sybase has targeted the OLTP environment from the beginning. These applications demand subsecond response time to the end-user client workstation. It is not enough to allow users to connect—the system must scale up as OLTP workloads increase.
- **Continuous availability of the application**: With hundreds of client users, applications must be continuously available. At the application level, the SYBASE Replication Server addresses this need by allowing a replicated server to run even if the primary server or network fails. At a single-server level, this can mean online backup without major impact on performance, automatic and rapid fallover, workload balancing, and full transaction recovery.
- **Large-scale query and decision support**: As customers put their users and customers online, the demand for query and reporting on the large client/server databases grows as well, often requiring the server to scan databases many gigabytes in size.
- **Mixed workloads on a single server**: In many cases, customers are asking single-server systems to handle both the OLTP and the decision-support workload in a single location.

In short, these requirements resemble the IBM mainframe environment. This article discusses how Sybase is designed to address these six requirements, and in particular, how HACMP/6000 plays an important role in addressing these needs.

## SYBASE System 10: Architected for Enterprise Client/Server

When Sybase was founded in 1984, the company brought two key technologies to the Relational Database Management System (RDBMS) market: client/server architecture and OLTP performance. This meant developing client and server Application Programming Interfaces (APIs), as well as developing key new concepts such as a multithreaded server, compiled stored procedures, triggers, and client/server and server/server Remote Procedure Calls (RPCs).

Josh Bersin

### SYBASE Open Client and Open Server

SYBASE SQL Server 10 is still the only RDBMS product that has integrated networking and both client and server APIs. All Sybase products are built on SYBASE Open Server and Open Client, a network-based, multithreaded interface designed for high-performance networked computing.

Open Server and Open Client, widely used interfaces in the industry, support both SQL and RPC communications between clients and servers and between servers and servers. They run across a large variety of network protocols and vendor network libraries, and on nearly every platform in the industry including AIX and MVS/ESA™.

The Sybase client/server architecture is based on multithreaded connections that allow both synchronous and asynchronous communications. They allow event notification and polling as well as store-and-forward message queuing through the SYBASE Replication Server.

### SYBASE SQL Server for Uniprocessors and SMP Systems

The SYBASE SQL Server, which forms the base of the family, is a multithreaded RDBMS engine designed from the ground up for network-based access. By using an optimized threading model, SQL Server can handle thousands of simultaneous users in a single process. A client user accessing SQL Server from Open Client requires less than 50 KB of server memory.

SQL Server's design point has always been optimized for OLTP. For example, in September 1993, Sybase and IBM completed a TPC-A benchmark that attached 2,760 client users to a single SQL Server on a RISC System/6000 Model 990 and achieved average response time of .79 second.

SQL Server pioneered the use of compiled, shared, stored procedures and triggers, which can not only control the local database, but can also send RPCs to other remote servers. This architecture (shown in Figure 1) makes possible a high-performance, guaranteed integrity, client/server database.

SQL Server has many features designed for continuous availability. Online backup has always been available. In System 10, SQL Server includes a separate backup server process that provides online backup with little or no impact on the performance of the SQL Server. Backup server can back up tens of gigabytes per hour to multiple devices and can operate on a separate machine to allow the OLTP server to run unaffected during backup.

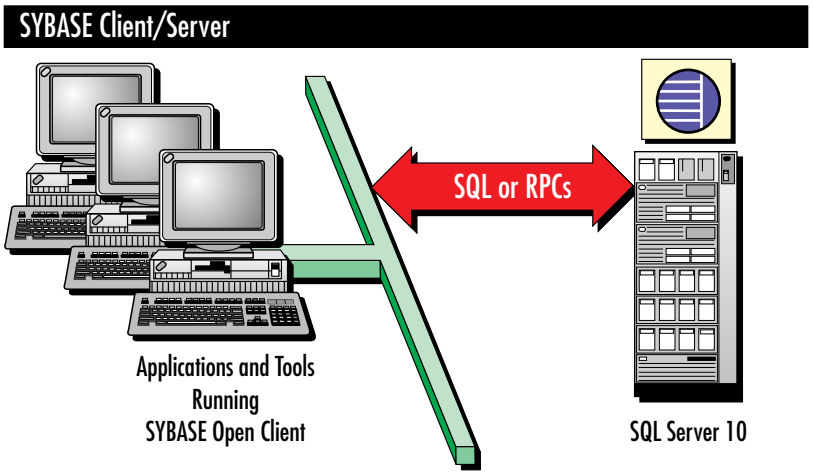



**Figure 1. SYBASE client/server architecture**

### SYBASE SQL Server for SMP Systems

For Symmetric Multiprocessing (SMP) systems, Sybase sells a multiprocess implementation in which multiple SQL Servers are loaded in memory and CPUs are dedicated to the server complex. The SQL Servers communicate through shared memory and can dispatch threads to a run queue that looks for the next available processor. The result is a highly scalable design, leveraging

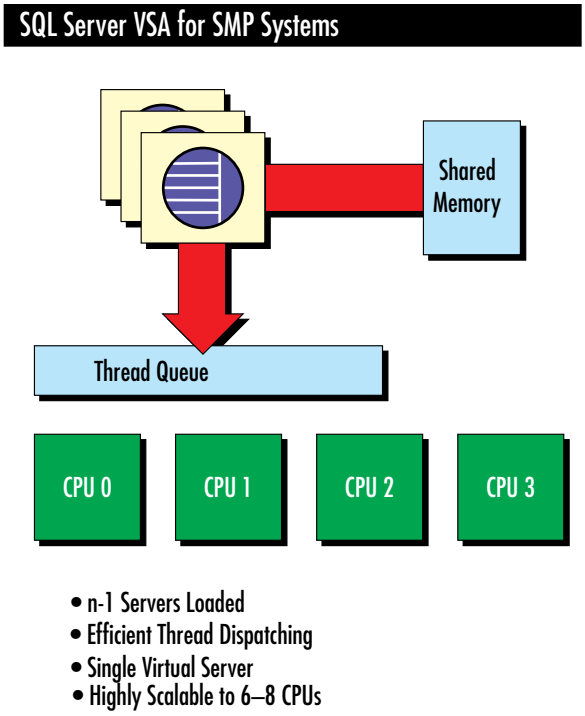



**Figure 2. SQL Server VSA for SMP systems**

the strength of the SQL Server uniprocessor architecture along with a single-server image to applications, shown in Figure 2.

### Distributed and Parallel Servers from Sybase

Other products that encompass the SYBASE architecture include the following:

**Replication Server**: Allows customers to create primary and replicate copies of data with automatic, fault-tolerant transaction distribution from primaries to replicates. Replication Server uses a store-and-forward protocol to provide continuous or timed transaction distribution to replicate sites. Replication Server uses a subscription process to allow replicate sites to receive continuous or timed updates from a primary site at a table, row, column, or user-defined level.

**OmniSQL Gateway**™: Allows any client application to access multiple heterogeneous data sources transparently as if they were located in a single database. OmniSQL Gateway gives an application turnkey access to data in SYBASE SQL Server, DB2, Oracle, Ingres, Informix, and a variety of other data formats.

**Control Servers**: Consists of a family of client/server performance monitoring and system management tools.

**Navigation Server**™: Couples together SQL Server engines in a message-based parallel architecture to provide parallelized OLTP, queries, backup, and utilities for applications that demand large databases, large queries, and large numbers of users.

Navigation Server, jointly developed between Sybase and NCR, uses multiple SQL Servers, one per node, in a shared-nothing architecture with fully partitioned data. To a client, Navigation Server looks like a single, large SQL Server, allowing applications to exploit the power of a large parallel system as if it were a single-node machine. Unlike other parallel server offerings, it is fully message-based and optimized for partitioned, shared-nothing machines. Navigation Server is scheduled to be available on IBM's SP family of systems in early 1995.

**A network-based software architecture is fundamental to leveraging future hardware architectures.**



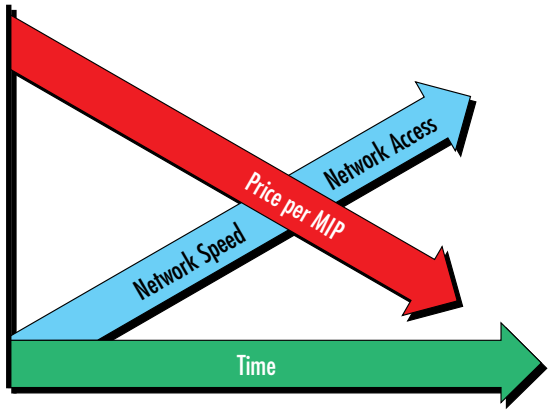



**Figure 3. Trends in computing technology**

## Hardware Technology Evolution: Clustered Architectures versus Others

Hardware technology advances at a dramatic rate. Sybase sees two fundamental shifts occurring in hardware technology (shown graphically in Figure 3). First, CPU performance is increasing at a dramatic rate—doubling price/performance per year. Second, high-speed, low-latency networks are becoming inexpensive and ubiquitous. For the future, we believe that a network-based software architecture is fundamental to leveraging future hardware architectures.

## Today's Range of Computer Architectures

To exploit these technology trends, IBM and other vendors are developing new computing architectures that are critical to the design of our software architecture, as shown in Figure 4.

### Clusters versus Parallel versus SMP Systems

For high-performance OLTP today, the most scalable systems are uniprocessors and SMP. OLTP workloads place heavy demands on updating the database—therefore when multiple CPUs are involved, they must be able to communicate with each other rapidly to serialize access to data. SMP

| System | Uniprocessor | SMP | Cluster | Parallel |
|---|---|---|---|---|
| Architecture | Shared Everything | Shared Memory | Shared Disk | Shared Nothing |
| # of nodes | 1 CPU | 2–16 CPUs | 4–32 CPUs | 8–64+ CPUs |
| IBM model | Model 990 | PowerPC™ SMP | HACMP/6000 | SPx family |

**Figure 4. Range of hardware architectures for client/server computing**

achieves this through shared memory, and can often scale in OLTP workloads at over 90% efficiency for four to six nodes. Design implementations, however, often limit the number of nodes to six or eight.

Parallel systems, on the other hand, or shared-nothing systems such as IBM's SP2 are excellent for large-scale decision support and reporting. By using partitioned parallel technology such as Navigation Server, they can access a large database in parallel, providing a near linear speedup in query performance as additional nodes are added. Navigation Server is also designed for parallel OLTP, allowing the system to increase the number of users as additional nodes are added.

Clusters like HACMP/6000 offer different advantages. The first key advantage is high availability. If one node in a cluster fails, another node takes over. The second key advantage is that each node in a cluster can be a large SMP itself, allowing a cluster to theoretically scale far beyond a large SMP system. This allows thousands of clients to be connected to a single database.

However, there are limitations. Clusters, because they communicate via networks and shared disk, do not scale well in update-intensive workloads. Even light OLTP workloads do not scale well. This is because multiple nodes that want to update a single database record must communicate through a Distributed Lock Manager (DLM) to serialize access to data. The DLM, as implemented today, does not scale well for update-intensive work like OLTP. The best clusters available today get slightly more than two times performance going from one to four nodes.

Clusters are also not optimal for large database queries. Query performance is limited by the time it takes to scan large tables—and this cannot be speeded up significantly without partitioning data. Sybase sees the Navigation Server and SP2 as an excellent solution for this workload.

## HACMP/6000 and Sybase Today

For fallover and hot-standby modes, SYBASE SQL Server supports high-speed switchover and a variety of configuration options.

Here are some key features of Sybase for HACMP/6000 today:

- Fast fallover through a tunable transaction log ("recovery interval") to create a cluster with either fast fallover and more frequent checkpointing, or less frequent checkpointing with longer fallover
- Use of Logical Volume Manager (LVM) raw I/O for highest performance fallover and recoverability
- Open Client architecture that allows client applications on the network to transparently switch or retry a failed server during high-availability switchover and takeover of a failed system's IP address
- Replication Server services to allow one node to function as an OLTP server and another as the query and reporting server for maximum performance and high availability

## SYBASE for HACMP/6000 in Mixed OLTP and DSS Workloads

Due to the large and unpredictable amount of CPU and disk I/O required for large queries and reports, mixing OLTP and decision support in a single machine is difficult to manage. One excellent solution to this problem is to partition the workload onto two nodes of a cluster. Many customers are doing this today using SYBASE Replication Server.

With SYBASE Replication Server, customers can implement high-performance OLTP applications on one HACMP/6000 node and replicate transactions to a second or third node for query or reporting applications. This allows client applications to receive predictable, guaranteed response time on the OLTP node without contention by periodic reports or queries. Decision-support and query functions are automatically routed to the second node by the Open Client application and do not impact the mission-critical applications.

Because the nodes are operating independently, there is no overhead from the HACMP/6000 DLM. Both systems run unconstrained at full individual CPU speeds. All systems in the cluster can run efficiently and provide full performance.

During a failure of one of the nodes, HACMP/6000 can automatically bring up the OLTP application on the surviving node and move the decision-support application to the background. This allows mission-critical applications to operate continuously with highly scalable function for the entire suite of applications.

Figure 5 illustrates transaction replication using SYBASE Replication Server.

**For high-performance OLTP today, the most scalable systems are uniprocessors and SMP.**

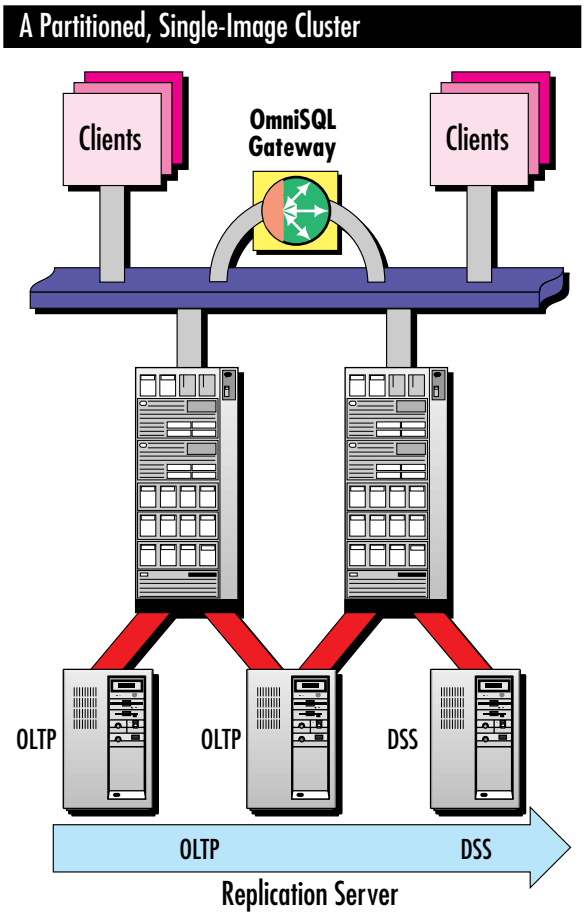



**Figure 5. A partitioned, single-image cluster for OLTP and decision support**

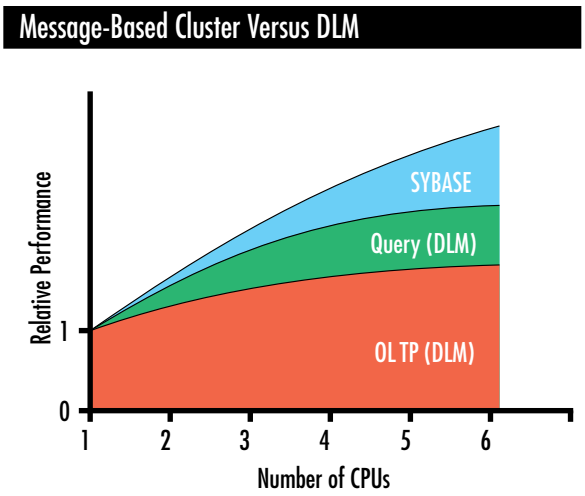



**Figure 6. SYBASE message-based cluster performance versus traditional DLM**

## SYBASE for Clustered Systems: A Message-Based Architecture

The Sybase approach to concurrent disk sharing uses short, low-latency messages between the nodes to serialize access to data. As network speeds improve, the cluster performance improves.

Instead of using the existing HACMP/6000 DLM, Sybase is developing a clustered SQL server for concurrent disk sharing with HACMP/6000. It is designed to scale beyond what is available with the DLM approach today. We expect the product to be available in 1995. Our approach is to allow each node in the cluster to "own" a portion of the database. When one node needs to update data owned by another node, it sends a message to the owner. The owning node either performs the transaction or releases the lock so that the first node can update the record.

Because this approach is built largely around the network, we believe that it will scale beyond the DLM approach available today. As shown in Figure 6, our design goals are to generate a product that provides good OLTP scaling as additional CPUs are added to the cluster.

This approach also allows for continuous availability. If a node fails, another node takes over its ownership, and the workload is transferred to the surviving nodes. Any transactions in process are rolled back or rolled forward. Client connections can be automatically routed to a surviving node. Sybase's design goals for clusters are to deliver breakthrough performance in two areas: continuous availability with workload balancing and OLTP performance.

### Hardware Target: Clusters of Uniprocessors or SMPs

As shown in Figure 7, the SYBASE cluster architecture is designed to work across clusters of uniprocessors, SMPs, or combinations. SYBASE will use shared, concurrent access disk ("concurrent access for HACMP/6000") and will not require the use of the HACMP/6000 DLM. The product is designed to scale to four nodes and beyond, with excellent OLTP performance characteristics. In each of the uniprocessor or SMP nodes, the product will use existing SQL Server uniprocessor or SMP technology.

### Benefits of the Sybase Cluster Approach

The following are some benefits we see with the cluster approach.

**Continuous availability—concurrent resource access**: SYBASE will provide continu-

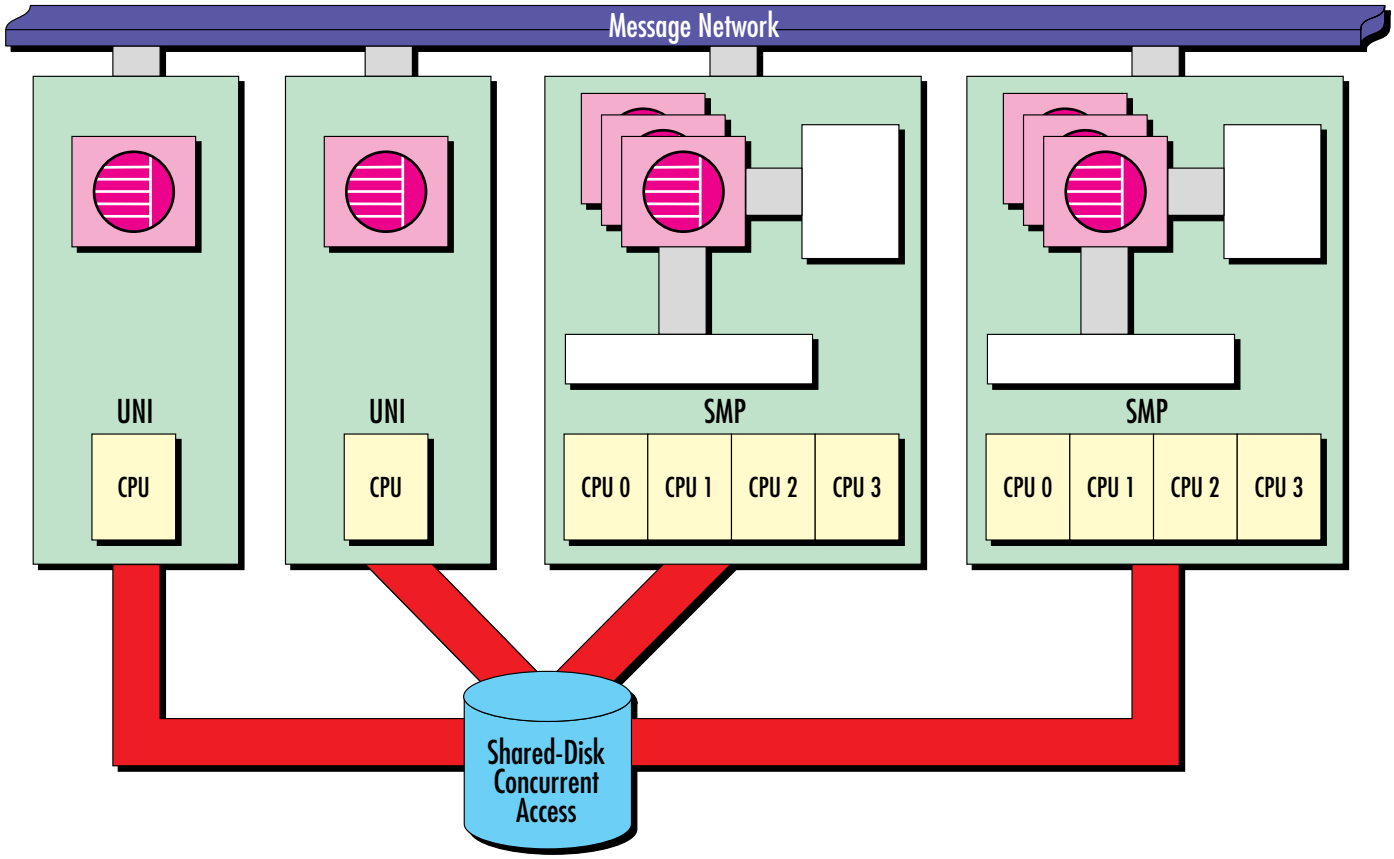



**Figure 7. SYBASE cluster implementation (HACMP/6000 Concurrent Access mode)**

ous availability at a transaction level, with a client having the ability to start a transaction on any node and having the transaction completed or rolled back if any system fails. If a node fails, the connections are transferred to a surviving node, and the transactions are completed or rolled back.

**Workload balancing**: If a node fails or is added in this architecture, work is automatically rerouted to the surviving nodes, allowing the cluster to reconfigure itself. This provides dynamic reconfiguration of the workload if hardware changes or unplanned outages occur.

**Scalable high performance**: Each node in the cluster consists of an SQL Server or SQL Server/VSA system, offering the same level of performance available in a non-clustered environment. When multiple nodes attempt to write to the same record, conflicts are resolved rapidly, providing high scalability. Unlike the current DLM approach, the system is designed for update-intensive applications like OLTP.

**Large numbers of clients**: The efficiency of the SQL Server and VSA implementations on each node allows the Sybase cluster to support thousands of users accessing a single database. With less than 50 KB of memory required per client, a single-node uniprocessor or SMP will support hundreds of users. The overhead is minimal as additional nodes are added, allowing thousands of users to be connected to the database.

**Data integrity**: Sybase pioneered using stored procedures and triggers to guarantee data integrity in a networked environment. The Sybase message-based cluster will use this technology to guarantee data and transactional integrity across the network.

| Application Environment | Sybase | IBM | Performance |
|---|---|---|---|
| High-performance OLTP and queries | SQL Server/VSA | RS/6000 Uni<br>RS/6000 SMP | 300+ TPC-A<br>400+ TPC-A<br>Hundreds of users |
| Mixed workload OTLP and DSS | SQL Server/VSA<br>Replication Server | Uni and SMP<br>HACMP/6000 | 300–400 TPC-A per node<br>Partitioned workload<br>Hundreds of OLTP users<br>Hundreds of DSS users |
| Scalable OLTP and DSS with high availability | Clustered SQL Server<br>(available in 1995) | HACMP/6000 | 300–400 TPC-A per node<br>1000+ TPC-A in cluster<br>Scalable shared disk<br>Workload balancing<br>Thousands of clients |
| Large-scale queries<br>Very large database (terabytes)<br>Very large OLTP | Navigation Server<br>(available in 1995) | SP2 | Thousands of TPC-A<br>Terabytes of data<br>Thousands of clients |

**Figure 8. Sybase architectures for IBM RISC System/6000-based systems**

## Conclusions

Figure 8 summarizes Sybase architectures for RISC System/6000-based systems. Sybase believes that with a scalable OLTP solution, a new generation of cluster applications will become possible with HACMP/6000, allowing applications to fully exploit the power of new processing and network technology as it becomes available.

**Josh Bersin**, Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608. Internet: joshua.bersin@sybase.com. Mr. Bersin manages Sybase's product and marketing strategy for the IBM product line. He has spent over 10 years in the industry in a variety of technical, marketing, and management positions at IBM and Sybase. He holds a BS from Cornell University, an MS from Stanford University, and an MBA from the University of California at Berkeley.

# AIX Communication Service: Streams


By Eddie Ho, Saurabh Desai, Derwin Gavin, and James Partridge

Streams is a standard interface for developing modular, portable networking layers that reuse code from different communication protocols. It facilitates the rapid growth of communication services. The Streams framework on AIX Version 3.2.5 can save development costs for application developers and communications subsystem providers. This article introduces system engineers to Streams.

Communications support is the primary building block for commercial applications on AIX. Other enabling applications include Online Transaction Processing (OLTP), relational databases, and multimedia. These applications have unique real-time interface requirements and use the standard interfaces provided by the communication subsystem to interoperate with other applications.

The latest network technology using Streams can assist and shorten the development time for application developers and communication subsystem providers. Streams is the de facto standard from AT&T® that solves many architectural problems. It is adapted by all vendors using AT&T System V Release 4. Since Streams is a defined, common standard, it also enhances application portability. A Streams module written for one system should run under Streams on another system with little or no modification.

Because Streams is based on the seven-layer Open Systems Interconnection (OSI) model, it promotes modularity. Delivering an old application over a new communications service should require replacing only the lower-level modules of a stream.

There are several benefits of using Streams for communication services:

- The protocol stack can be customized in real time by the user application. This makes the code path more efficient and direct.
- Protocol layers and device drivers can be shared and mixed.
- Runtime protocols can be configured by the user application.
- Conformance to the OSI seven-layer design results in building blocks with a common message-driven interface.
- It promotes reusable code, which saves development time and resources.

## What is Streams?

Streams is a flexible set of tools for developing UNIX system communication services. It defines a generic message-driven queuing interface and provides a framework and tools for implementing communication services, such as a network protocol stack. Streams does not impose any specific network architecture—it is simply a framework. Streams includes the following components:

**Stream head**: The head is the part of the stream that provides an interface to the user process. It supports all the standard device-driver system calls (`open`, `close`, `read`, `write`, and `ioctl`) and three additional calls—`putmsg`, `getmsg`, and `poll`.

**Stream modules**: Stream modules are a defined set of kernel-level routines and data structures used to process data, status, and control information. They implement processing functions to be performed on data that flows on the stream. There may be zero or more modules depending on processing requirements.

**Stream device driver**: The Streams-based device driver can support an external I/O device or a pseudo-device. The driver typically handles data transfers between the kernel and the device. It does little or no data processing other than converting Streams messages to hardware events and vice versa. To be modular, the program

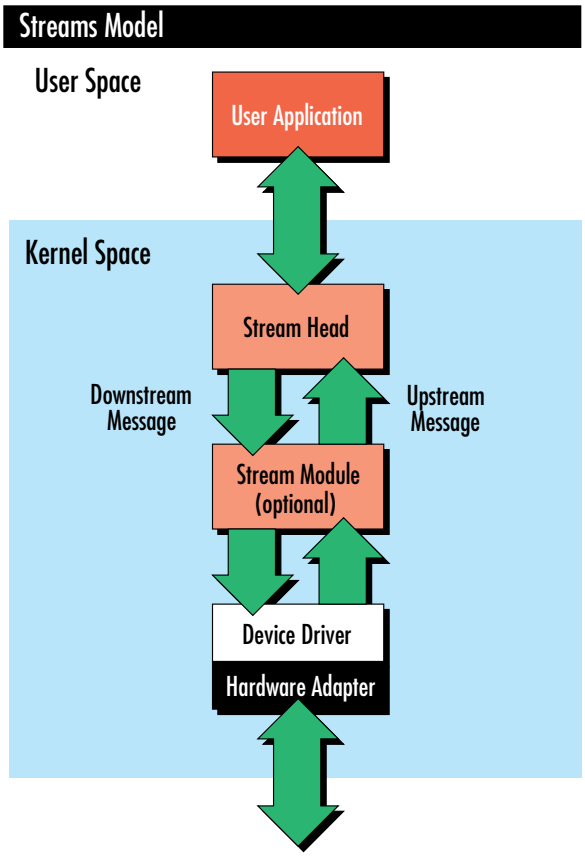



**Figure 1. Streams model**

| System Call | Action |
|---|---|
| `open()` | Open a stream. The open system call recognizes a Streams file and creates a stream to the specified driver. |
| `close()` | Close a stream. |
| `read()` | Read data from a stream. |
| `write()` | Write data to a stream. |
| `ioctl()` | Control a stream. This allows the application to control device-specific functions. The `ioctls` supported include all `ioctls` normally supported by a device driver, plus `ioctls` supported by the Streams framework (`I_LIST`, `I_PUSH`, `I_POP`, and so on). |
| `getmsg()` | Receive a message at the stream head. |
| `putmsg()` | Send a message downstream. |
| `poll()` | Notify the application program when selected events occur on a stream. When used with the Streams `I_SETSIG ioctl` command, the `poll()` system call allows an application to process I/O in an asynchronous manner. |

**Figure 2. System calls**

should isolate all hardware dependencies to this module.

Figure 1 shows a simple transport using Streams. The user application makes requests to the stream head, which satisfies the request if possible. Otherwise, the request is passed downstream until a module can satisfy it. The response to the request is passed upstream, back to the application.

The Streams framework is a set of utilities that supports operating the stream and provides programmatic control to the application. This control consists of functions such as listing Streams module names on the stream stack, real-time pushing and popping of Streams modules onto and off the stack, and obtaining statistical information from the Streams modules.

The Streams framework also supports a facility for high-priority messages that are transmitted and processed out of sequence—before processing normal data and independent of any flow control imposed on that data.

## Streams API

The Streams facility is controlled directly by using system calls. The interface is upwardly compatible with the existing character I/O facilities. Figure 2 shows the system calls.

The Streams-specific system calls are as follows:

**putmsg interface**: This interface is similar to `write()`. The `putmsg()` subroutine provides a data buffer that is converted into an `M_DATA` message. It can also provide a separate control buffer to be placed into an `M_PROTO` block. The `putpmsg()` supports priority messages, which are processed before ordinary messages and stream data. The normal `write()` interface provides only the stream data that is converted into an `M_DATA` message.

**getmsg interface**: This interface is similar to `read()`. The primary difference is that `read()` accepts only the data (messages sent upstream to the stream head as message type `M_DATA`), while `getmsg()` can simultaneously accept both data and control information (messages sent upstream as types `M_PROTO` for `getmsg()` or `M_PCPROTO` for `getpmsg()`). The `getmsg()` interface also differs from `read()` by preserving message boundaries so that the same boundaries exist above and below the stream head. A normal `read()` generally ignores message boundaries.

### Stream Message Format

*Messages* are the communicating vehicle within Streams. A message contains data or information based on the type of message. The messages can

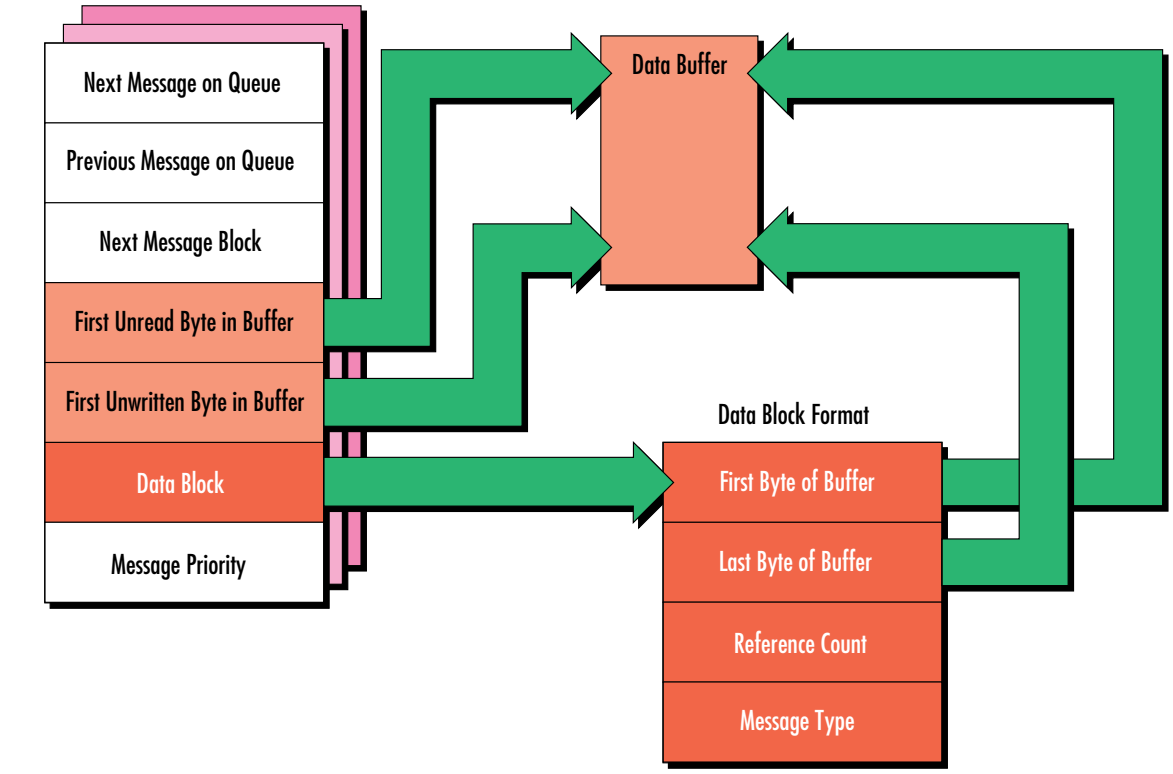



**Figure 3. Streams message queue structure**

originate with a driver, a module, or the stream head. Transferring messages between modules is done via system calls that remove a message from a queue or place a message on a queue to be serviced by another Streams module. The Streams framework maintains its own storage pool for messages. Figure 3 shows the message block structure.

A module can send any message type in either direction on a stream. However, based on their intended use and their treatment by the stream head, each message can be categorized as upstream, downstream, or bidirectional, allowing for full-duplex communication.

The most common normal message types are as follows:

- **M_DATA message**: Intended for user data. The message is generally sent bidirectionally.
- **M_PROTO message**: Contains control information and data. The message format is a single `M_PROTO` message followed by one or more `M_DATA` messages.
- **M_PCPROTO message**: Contains the same format and characteristics as the `M_PROTO` message except for priority and some additional information. This message is intended to send data and control information outside the normal-flow control message path.

## Streams in AIX

Communications subsystem vendors use the Streams framework as the implementation of choice. NetWare® services and Transport Layer Interface (TLI) applications use Streams.

Using Streams, NetWare/6000 shown in Figure 4 supports such protocols as SPX/IPX, NetBIOS, and NVT protocol.

The TLI facility allows applications to create connections to remote peers in a protocol-neutral environment. Protocol layer substitution is the means to achieve this in the Streams environment. Depending on the transport provider specified by the application during the session open time, the Transport Interface module can access either a Streams-based protocol stack natively or a socket-based protocol stack such as TCP/IP.

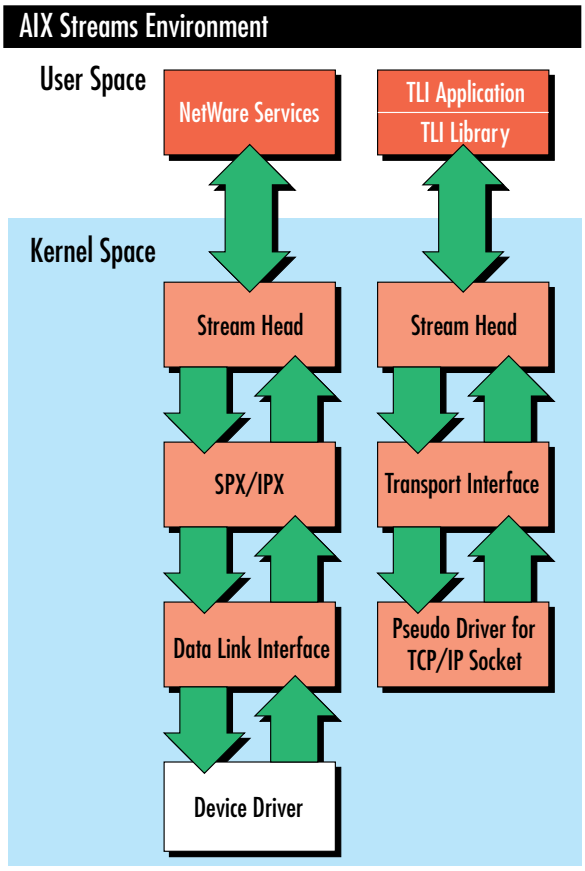



**Figure 4. AIX Streams environment**

## Conclusion

Because Streams is a generic message-driver interface, Streams modules make few assumptions about the modules above and below them in the stream stack. This isolates upper-level Streams modules and application code from changes in lower-level environments and hardware interfaces, providing greater flexibility in configuring applications for delivery.

**Eddie Ho**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Ho is a senior programmer in AIX Communications. He has a BS in Computer Science from the University of Wisconsin and an MS in Computer Science from North Dakota State University.

**Saurabh Desai**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Desai works in AIX Communications. He has a BS in Electronics from S.P. University in India and a MS in Computer Systems Design from the University of Houston.

**Derwin Gavin**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Gavin is an associate programmer in AIX Communications. He has a BS in Computer Science from Grambling State University in Louisiana.

**James Partridge**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Partridge is a staff programmer in AIX Communications. He has a BA in English from the University of Texas.

## Awards for PowerPC, RISC System/6000

**BYTE Magazine**—The IBM PowerPC 601 microprocessor recently received *BYTE* Magazine's Editorial Award of Excellence for one of the best technologies of 1993. According to *BYTE*, the PowerPC 601 "was the biggest overall vote-getter by a wide margin." IBM Microelectronics Division manufactures the PowerPC 601, which was co-developed with Motorola™.

**Reseller Management**—The 60,000 readers of *Reseller Management* magazine recently named IBM's RISC System/6000 the "Best-To-Sell RISC Workstation" in its 1993 Readers Choice Awards. In the November 1993 issue announcing the award, the publication said, "IBM's RISC System/6000 is the winner in the workstations category. It not only won, but garnered three times the votes of the longtime champ, Sun Microsystems®."

**Datamation**—The RISC System/6000 received the 1993 "Product of the Year" award from *Datamation* in the PC/Workstation category. ■

# LANHOP/6000 Version 1.0


By Wayne G. Wachtstetter and Ralph E. Vosburg

As the number of portable and laptop computers increases, so does the need for travelling business executives to have remote access to crucial reports and databases. This article describes LANHOP/6000, a program that enables remote PCs to access LAN and mainframe resources. The article focuses on connectivity, control, and potential uses of LANHOP/6000 for cost-effective communication and information sharing.

Do you need to access your mainframe computer or Local Area Network (LAN) resources—such as clients or servers—from different locations? When you travel, would it be helpful to retrieve a report for the next important appointment without returning to the office? Would you like to work on the next programming project from home? Would your company like to have some of its employees work at home?

IBM's goal with Local Area Network Home Office Program/6000 (LANHOP/6000) is to make all office computing resources as easy to use remotely as in the office. Remote users can run applications such as spreadsheets and word processors, perform database inquiries, transfer files, execute commands, and send messages over dial-up lines.

LANHOP/6000 enables dial-in access to LAN-based resources from a hotel room, customer office, or home. The user's personal computer (running DOS, OS/2, or Microsoft Windows) establishes a secure connection with the dial-in server at the office, as shown in Figure 1. LANHOP/6000 uses a TCP/IP component called Serial Line Internet Protocol (SLIP) for asynchronous transmission of IP packets between the remote computer and the LANHOP/6000 server.

LANHOP/6000 communicates over LANs and Wide Area Networks (WANs) using the standard TCP/IP protocol. With portable computers, users can download files, work with them locally, then logon to the network again, and return the data to the network as shared files. Users can transfer files between LAN servers and mainframes, perform maintenance, print documents, log on to LAN domain servers or dedicated office computers, and log on to MVS or VM mainframe hosts.

LANHOP/6000, which emerged from an internal IBM product called Off-Premises Computing Program, became commercially available in December 1993.

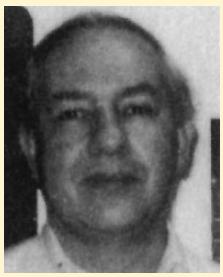
## LANHOP/6000 Components

Key components are the LANHOP/6000 client PC and the LANHOP/6000 server.

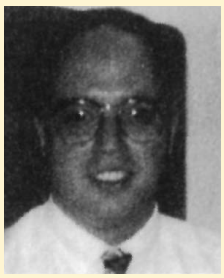
The client can directly address any LAN-based resource configured to participate within the TCP/IP environment. It can also access many applications running under OS/2 2.X, DOS 5.0+, and DOS/Windows 3.1. The LANHOP/6000 client application accommodates an array of Hayes®-compatible modems. Modems that support V.32bis or V.42bis data compression are recommended for increased data transfer speed.

The LANHOP/6000 client has a graphical user interface that can be customized by other vendors or customers via Application Programming Interfaces (APIs). APIs allow customers to personalize each client interface to interact with their in-house applications. For example, universities can design customized interfaces for remote users in different departments (such as Business, Computer Science, or English). Each department can access its own specialized applications via personalized menus, windows, and so on.

The LANHOP/6000 server is an AIX-based application on a RISC System/6000 with TCP/IP support. A server consists of a network interface, many modems, and managing software. Depending on its resources, a single server can support over 255 simultaneous communications ports.



Wayne G. Wachtstetter



Ralph E. Vosburg

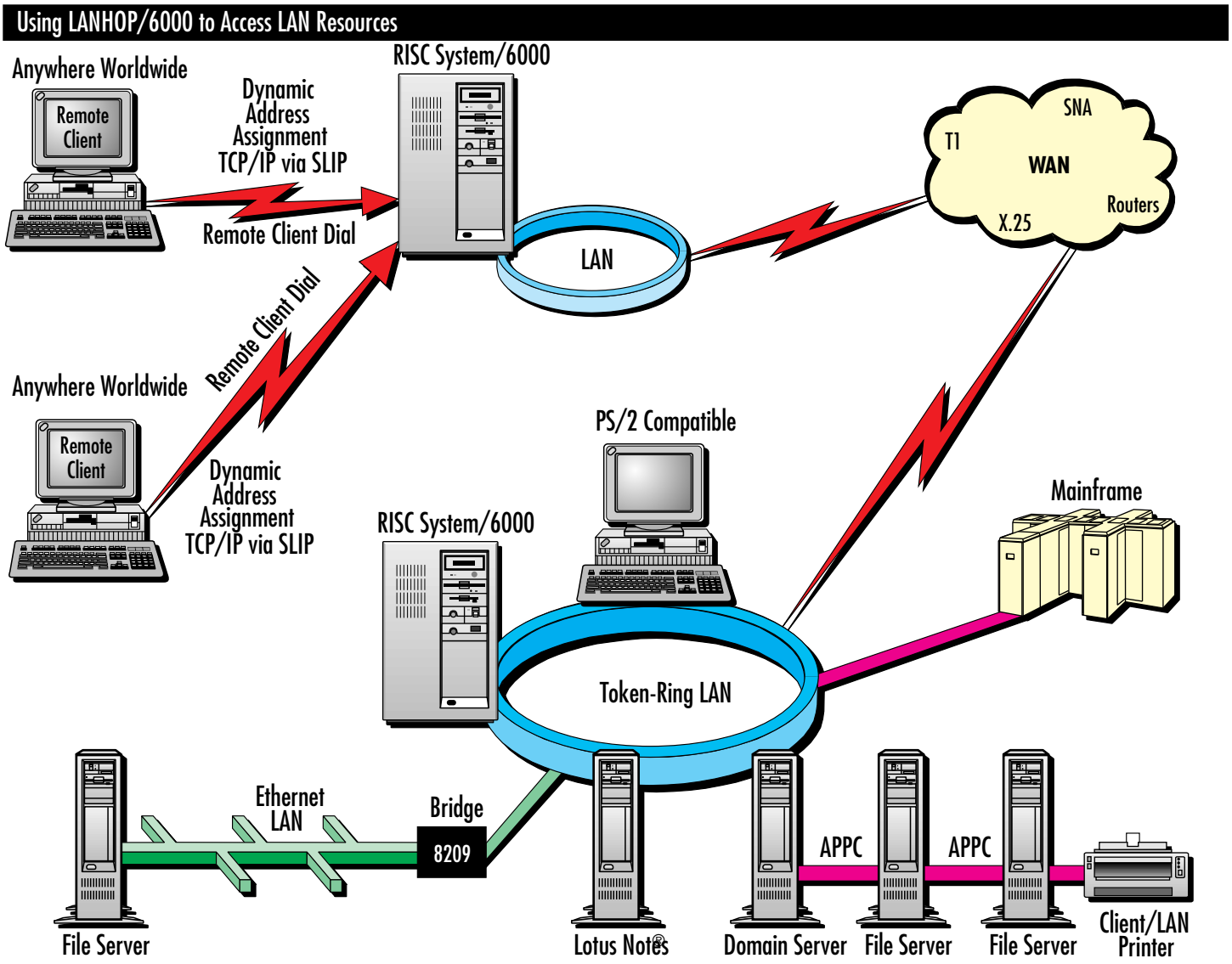



**Figure 1. Remote access to LAN resources via LANHOP/6000**

Since the server is not dedicated to LANHOP/6000, other applications can also run on the server. The server has a full range of functions for managing user IDs and passwords on local and remote gateways, gathering accounting information, controlling security and authentication, and using Simple Network Management Protocol (SNMP) agents for system management.

### Remote Node Initialization

Initially, the modem is not connected to a TCP/IP port at the LANHOP/6000 server. The port is connected to and monitored by the server. When an authenticated user requests a session, the modem and its communications line are then connected to TCP/IP.

The server first validates the user's password. Then, from a pool of addresses, the server assigns an IP address to the remote client for the duration of this session. The remote client now becomes a node on the network.

## Server Operations

LANHOP/6000 maintains several databases that store information about all remote users and usage statistics. Easy-to-use database commands allow a LANHOP/6000 administrator to set up, manage, and support the entire LANHOP/6000 environment. Figure 2 shows the type of information stored in the LANHOP/6000 databases. These databases tie together all the functions of the server.

### User Authentication

Authenticating users is an extremely important part of defending against security attacks. LANHOP/6000 uses passwords for this purpose. Since the LANHOP/6000 server is an entry into the Internet, this is a crucial security consideration.

The LANHOP/6000 server maintains a User Profile database to authenticate a user's identity. This database holds information about various objects, such as user ID, password, local hostname, password expiration date, and other information that the server must protect against unauthorized access.

In operation, a unique user identification name is entered at the client and passed to the server. After successful validation, a pair of distinct and complementary keys is created. One key is passed to the new client to encrypt the password for the current connection. The server checks the password against an encrypted value stored in a server database before the user is allowed access.

### Fencing

Fencing is an authorization technique to control which resources users can access. Each IP packet from the client is checked to determine if the user will be granted access to the designated resource or destination. In order to know which resources a user is authorized to access, each user is linked to a particular group that defines the resources—by name or IP address—available to each member of the group.

### Accounting

The LANHOP/6000 server generates accounting records and provides current and historical accounting information. Information is gathered by user session and stored in an Accounting database. All data is routed to selected servers at periods defined during installation.

This facility provides a way to bill for resource usage, if desired. Accounting API services allow external programs to retrieve resource-usage data.

The following types of information are logged into the Accounting database:

- User and account IDs
- Start and stop times
- Modem speed
- Gateway server name
- Dialing platform (DOS, OS/2, DOS/Windows, and so on)
- Port number
- Server IP address
- Code version

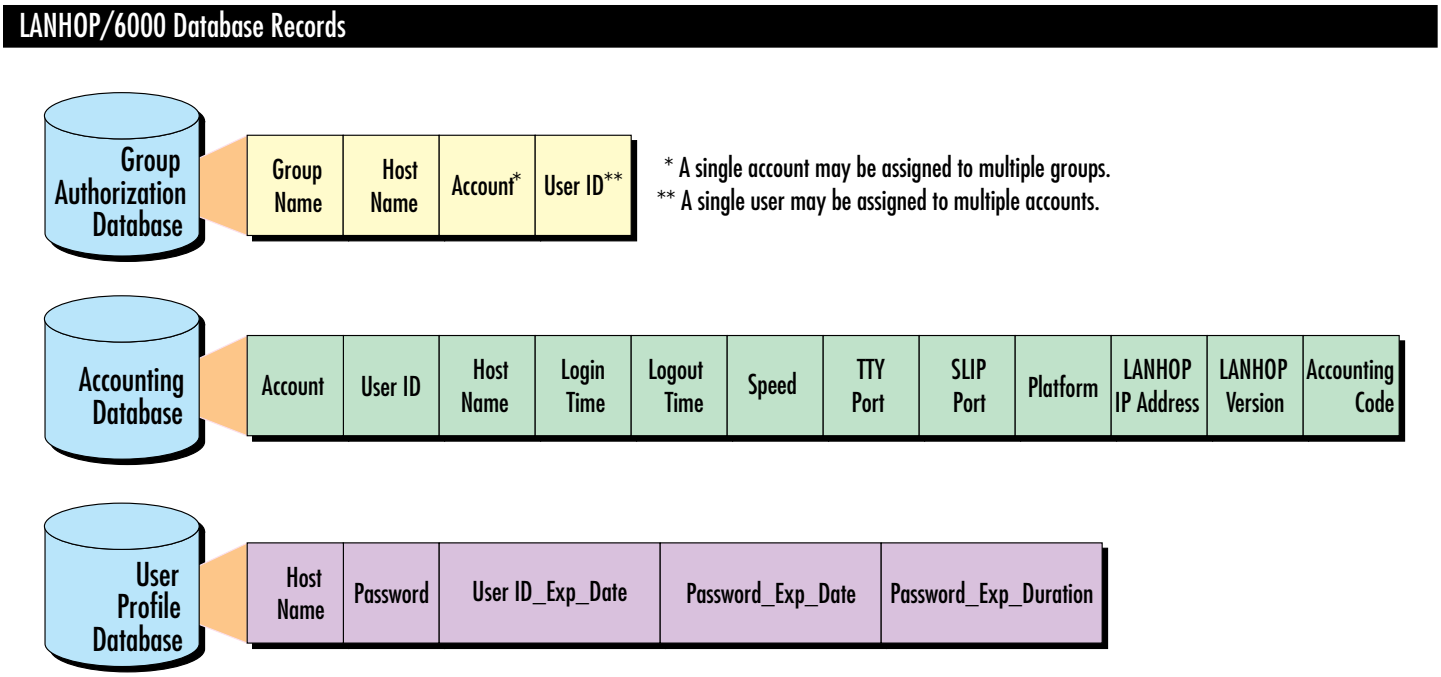



**Figure 2. Example of LANHOP/6000 database records**

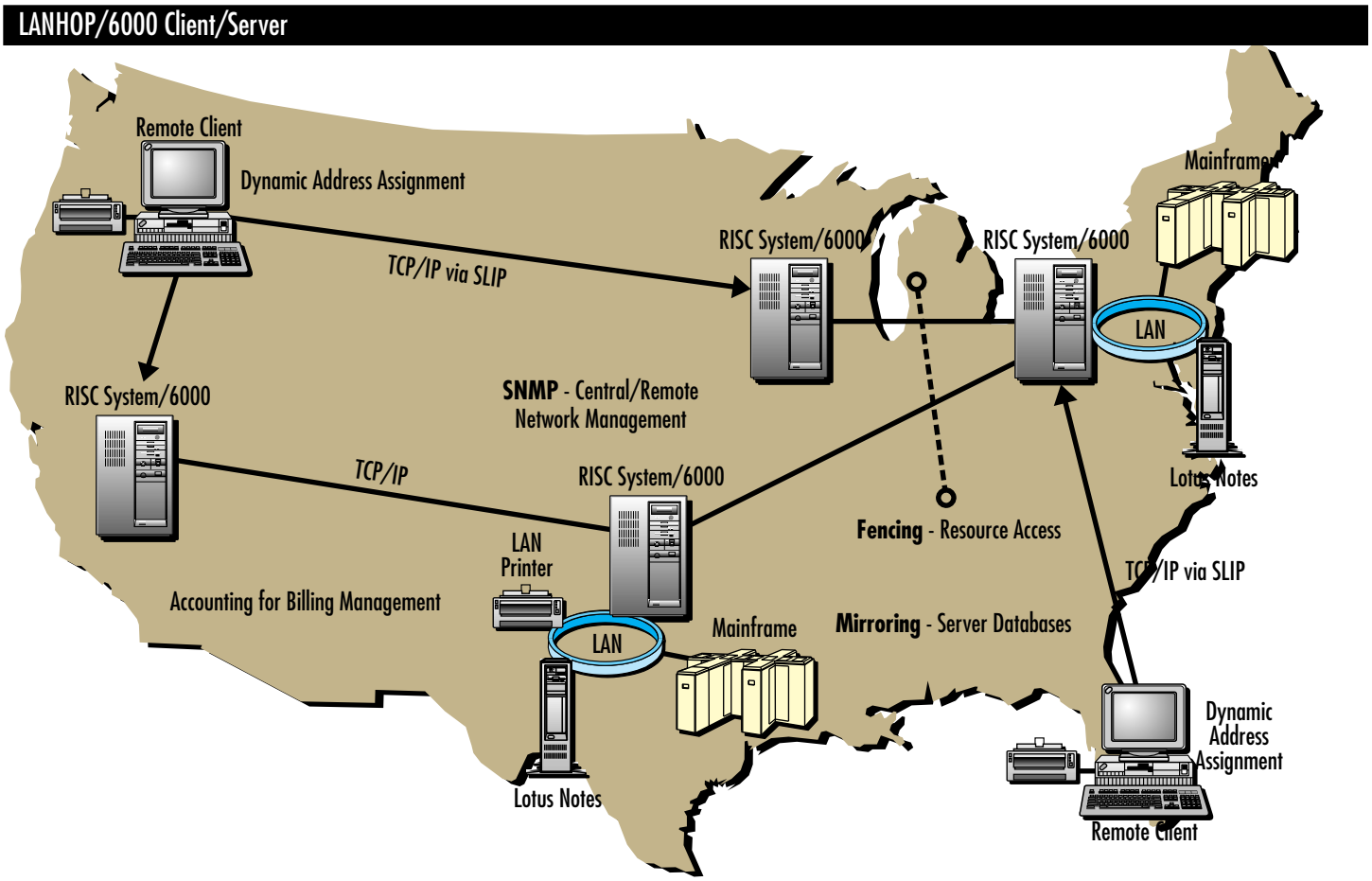



**Figure 3. LANHOP/6000 client/server relationships**

### Mirroring

Mirroring can interchange LANHOP client profile information among LANHOP/6000 servers through an Internet network. If the target LANHOP/6000 server or the link to it is unavailable, using store-and-forward allows the information to be stored and sent to the next LANHOP/6000 server. Eventually, each server will be updated.

Mirroring is useful for large installations of LANHOP/6000 in which synchronization of several servers is critical for clients' access across geographical areas.

The User Profile, Group Authorization, and Accounting databases contain information about clients for LANHOP/6000 services. Updates to these databases can be mirrored to all LANHOP/6000 servers on the LAN or WAN.

Since mirroring is automatic, there is no need for manual maintenance to multiple LANHOP/6000 servers. At installation time, a server can be designated as the control-point for setting up the User Profiles and Groups. This designated server will be the location used by an administrator to make changes.

LANHOP/6000 information routing is based on the relationship between the client and its server. Client changes—such as adding a new user or changing a password—result in updates to the attached LANHOP/6000 server. Within seconds, the information is routed to neighbor LANHOP/6000 servers, allowing the client to log in through any server, as shown in Figure 3. Data is shared and transferred transparently between interconnected LANHOP/6000 servers.

### LANHOP/6000 Server Management

LANHOP/6000 uses SNMP and Management Information Base (MIB) to operate concurrently with network management software such as NetView/6000. A MIB specifies the set of variables that the server must maintain. The management system uses SNMP agents to interrogate and obtain statistical information from the server. Software provides automatic statistical reporting via

NetView/6000 for tracking performance and errors for each dial-in connection. NetView/6000 monitors the activity of LANHOP/6000 databases (User Profile, Group Authorization, and Accounting). Local and remote updates, remote receives, total sends, and total resynchronizations are monitored.

Using NetView/6000, administrators can display information about LANHOP/6000 User Profile, Group Authorization, and Accounting databases.

## Client Operations

The LANHOP/6000 client provides an easy-to-use interface that allows users to configure their remote workstations and connect to all of their LAN resources. Any Hayes AT®-compatible modem can be used to dial-in to the server. The following sections briefly describe the concepts behind the LANHOP/6000 client.

### Client User Interface

LANHOP/6000 provides for an asynchronous dial-up connection. The connection may be tethered (line-based) or untethered (circuit-switched cellular). The client's window interface allows users to set up a workflow desktop using OS/2 Presentation Manager® or DOS/Windows. A window interface makes it possible to access messages and LAN-based resources asynchronously using TCP/IP protocols. Users can query application databases and obtain information. Data files can be transferred between a client and a LAN-based system and/or a remote mainframe. For example, you can capture database information from Lotus 1-2-3® and send it to a remote mainframe.

Users have complete interactive control over configuration and customization. Prompts are provided for selecting information to identify users, defining the resources to be accessed, configuring the connectivity medium, and setting up other vital data. LANHOP/6000 delivers the user's office resources through a powerful window interface using pull-down menus, pushbuttons, radio buttons, and dialog boxes. This is an interactive front-end with limitless connectivity and application access. Users can customize the interface by integrating familiar applications unique to their daily use.

The remote client provides an easy-to-use interface for end users. All setup, configuration, and connection tasks can be performed from pull-down options on the main window action bar. The interface allows end users to be productive immediately.

The LANHOP/6000 platform gives users access to data across mixed networks of LANs. Through TCP/IP encapsulation of NetBIOS protocol, client/server applications are extended to remote users. This enables the communications between NetBIOS-compliant applications. Also, TCP/IP applications (Telnet, TN3270, FTP, REXEC, and PING) are available with a pop-up dialog. An additional Telnet client emulator for TN3270 is available using TalkThru® from Software Corporation of America. TalkThru for OS/2 provides an Emulator High-Level Language API (EHLLAPI) interface and Presentation Manager front-end for TCP/IP.

### LAN Environment

LANHOP/6000 supports TCP/IP NetBIOS (compliant with RFC1001 and RFC1002). LANHOP/6000 extends the Ethernet or Token-Ring LAN-connected node to the remote client machine.

**LANHOP/6000 provides for an asynchronous dial-up connection.**

The TCP/IP protocol, common to most transport systems, enhances the remote user's connectivity environment. This commonality among different environments provides an interface to all LAN-connected servers. All operations and applications run at the remote machine, as though it were a LAN-connected machine.

## Usage Examples

The following examples show some ways to use LANHOP/6000.

- Salespeople can process orders and provide information to their customers faster with remote access to LAN- and mainframe-based information.
- LANHOP/6000 can provide remote access to patient records and other medical information to health-care professionals in remote doctors' offices or health-care facilities.
- Insurance policies or claim information can be reviewed and processed from the homes of customers or policyholders.
- System and LAN administrators can remotely support and manage LAN resources.
- Programmers can compile and debug programs, and support customers from any remote site. They can also perform remote backup and maintenance of client hard drives.
- Service industries such as trucking or delivery firms find wireless communications invaluable. Truck drivers can access all company resources

## COMMON Fall 1994 Conference

The COMMON Fall Conference will be held October 16–20 in San Antonio, Texas. The keynote speaker will be Lou Gerstner, CEO of IBM. For more information, contact:

Monika Paus
COMMON Headquarters
401 N. Michigan Avenue
Chicago, IL 60611-4267
Phone: (312) 644-6610
Fax: (312) 527-6657

(such as routing schedules or delivery changes) from the convenience of their cabs.

- College students can work from home or from their dorms to access all types of information on the campus network.

## Hardware and Software Requirements

The LANHOP/6000 server requires no special hardware except a modem.

The RISC System/6000 supports the 64- or 128-port asynchronous controller and the 16-port async adapter. Depending on the RISC System/6000 being used, a range of async ports (from 64 ports to 1,026 ports) is supported. Each communication port can be configured up to 38,400 bps (except the 64-port controller, which must be configured at 19,200 bps or below). The effective throughput will average between 2,400 bps and 25,000 bps, based on system loading and application dependencies.

The LANHOP/6000 client requires no special hardware other than a modem of the client's choice. The appropriate versions of OS/2, DOS, or DOS/Windows and TCP/IP software are required on the client machine.

## Future Enhancements

Future enhancements to LANHOP/6000 are expected in the following areas:

- Expanded client and server platforms
  - UNIX server
  - UNIX client
  - AIX client
  - Macintosh® client
- Optional applications
  - E-mail
  - Pager system
  - Multimedia
  - ISDN support
  - Cellular support
- Integration with other products
  - Support for OEM TCP/IP
  - Support for OEM X-Server applications, such as Hummingbird eXceed and Andataco™ Liken
- Additional protocols
  - Point-to-point
  - AppleTalk®
  - Internetwork Packet Exchange (IPX)
  - DECnet®

## Conclusion

Remote computing will open many business opportunities. New digital communications networks such as Integrated Services Digital Network (ISDN), Asynchronous Transfer Mode (ATM) using cellular or satellite communications, and wireless connectivity will help spread the use of remote access. Remote users will have complete access to network resources from virtually anywhere, with or without telephone lines.

**Wayne G. Wachtstetter**, IBM Corporation, Information Technology and Services Tools Development, 5601 Six Forks Road, Raleigh, NC 27609. Internet: wwachtstetter@.vnet.ibm.com. Mr. Wachtstetter is a senior systems analyst involved in evaluating LAN/WAN software and hardware technologies and designing network solution architectures and prototyping concepts to assist product developers. Mr. Wachtstetter has a BS in Industrial Education from Purdue University.

**Ralph E. Vosburg**, IBM Corporation, Information Technology and Services Tools Development, 5601 Six Forks Road, Raleigh, NC 27609. Mr. Vosburg is a product marketing technical specialist involved in market process design and media relations. He has a BS in Computer Science from Marywood College in Scranton, Pennsylvania.

# Making Backups of Mirrored Filesystems on AIX 3.2

By Jaime Vazquez

The AIX Version 3.2 Logical Volume Manager's capabilities were enhanced with the addition of filesystem mirroring. This article shows how to make backups of these mirrored filesystems—making use of one of the mirrored copies.

Although mirroring greatly increases the availability and robustness of filesystems on AIX, it is not meant to be a mechanism for offline backups. Any filesystem that has update activity occurring cannot be safely backed up and have complete assurances about the state of the backup. This is, of course, not the case for any filesystem mounted read-only. Still, a system administrator should take adequate measures to fully safeguard critical data. This may not be possible on systems that have applications or data that must be online continuously or offline for only a very short time.

## LVM Background

The Logical Volume Manager (LVM) allows logical data partitions to be mapped to any physical partition available to the volume group. LVM carries within it a mapping structure that shows the relationship between a logical file partition and the physical disk partition that holds the data. The user can set the partition size when the volume group is created, with values of 1 MB to 128 MB (in powers-of-2 increments). The default value is 4 MB. A physical partition is the smallest unit by which a filesystem can be expanded.

Each LVM listing command has a suboption to display the mapping data at all levels. For example, the `lslv` command has the `-m` option for describing the logical-to-physical mapping for a specified logical volume. Figure 1 shows an example of the output. It is also possible to create a logical volume in which the user specifies exactly where logical volumes are located in relation to physical partitions and physical volumes. The allocation map file can be used as input to `mklv` for this purpose. The allocation map is an ASCII file that describes the disk and partition number for the `lv`, and the order in which the `lv` will be created. Figure 2 shows an example of such a file.

When a filesystem or logical volume is deleted, its physical partitions are released to the free list, but the data still physically resides on the disk; it is a logical deletion, not a true physical deletion. By extracting the mapping information of the logical volume or its copy and using this information as input to the `mklv` command, it is possible to re-create a filesystem using the same physical partitions in the same order as the original. The filesystem structure remains intact. From the time

```
$ lslv -m hd1
hd1:/home
LP    PP1  PV1               PP2  PV2
PP3  PV3
0001  0038 hdisk2            0019 hdisk3
0002  0042 hdisk2            0020 hdisk3
0003  0043 hdisk2            0021 hdisk3
0004  0005 hdisk2            0022 hdisk3
```

**Figure 1. Sample output from the lslv command**

```
hdisk3:0019
hdisk3:0020
hdisk3:0021
hdisk3:0022
```

**Figure 2. Example of allocation map file**

Jaime Vazquez

```
#!/bin/ksh
# User must have root authority.
# Input to the script is the logical volume name device to be backed up.
#
if [ $# -ne 1 ]
then
   echo "Logical Volume name not specified on command line."
   echo "Syntax: $0 <lv-name>  "
   exit 1
fi
# Make sure this is a valid logical volume
lslv $1 > /dev/null 2>&1
if [ $? -eq 1 ]
then echo "Logical volume $1 not found."
     exit 1
fi
# Check if the filesystem is mounted.
# It must be unmounted!
FSNAME= getlvcb -L $1
mount | grep  $FSNAME > /dev/null 2>&1
if [ $? -eq 0 ]
then echo "Filesystem is still mounted. "
     echo "Unmount filesystem and try again. "
     echo "Process terminated."
     exit 1
fi
echo "Filesystem " $FSNAME " is being processed."
LVID= ~getlvodm -1 $1~
VGNAME= ~getlvodm -b $LVID~
NUM= ~getlvcb -c $1~
if [ NUM -eq 1 ]
then echo "Filesystem is not mirrored. Process aborted."
     exit 1
elif [ NUM -eq 2 ]
then lslv -m $1 | grep hdisk | awk '{ print $5":"$4 }' > allocmap.file
     NEWNUM=1
else lslv -m $1 | grep hdisk | awk '{ print $7":"$6 }' > allocmap.file
     NEWNUM=2
fi
MAPSIZE=~wc -l allocmap.file | awk ' {print $1} ' ~
echo "Allocation map file created of size of " $MAPSIZE " partitions."
sync;sync
rmlvcopy $1 $NEWNUM
if [ $? -ne 0 ]
then echo "rmlvcopy command failed with rc = " $?
     echo "Process terminated."
     exit 2
fi
echo "Logical volume " $1 " has been reduced to " $NEWNUM " copies."
mklv -m allocmap.file -y backuplv $VGNAME $MAPSIZE
echo "mklv completed with rc = " $?
mount -v jfs -o ro /dev/backuplv /mnt
exit $?
```

**Note: This sample code is provided as-is without any warranty or guarantee from IBM.**

**Figure 3. Shell script to automate backup steps 1 through 4**

the mirror copy is deleted until the image logical volume is created, there must be no changes to the global state of filesystems on that volume group. Deleting the mirror copy releases physical partitions to the free pool. These partitions can then be used for other filesystem actions, such as expanding the size of a filesystem.

## Making the Backup

The steps to make the backup are as follows:

1. Create an allocation file of the physical partitions used in the most secondary mirror copy.
2. Unmount the filesystem.
3. "Break" a mirror by deleting the secondary mirror copy.
4. Re-create a logical volume using the allocation file from step 1.
5. Mount the new logical volume read-only.
6. Perform the backup.
7. Unmount and delete the created logical volume.
8. Add a mirror copy to the filesystem, reusing the allocation file.
9. Resynchronize the logical volume.

In the current implementation of LVM, breaking the mirror can be accomplished only by deleting one of the copies. The resulting free space still contains the original data. By reusing the same physical partitions in the exact order as the mirror copy, the filesystem still contains the original data.

The shell script in Figure 3 shows how to automate steps 1 through 4. It can be modified as required. Using the name `backuplv` for the created logical volume is not required. The created logical volume can be mounted on any mount point, not just `/mnt`.

After the shell script has been run, the mirror copy has been mounted and is available to be backed up by any means desired, such as `tar`, `cpio`, `pax`, and `backup`. It is important to note that there is one less mirror copy available to the user. If the logical volume had only itself and one mirror copy, the system would use only the original copy for the backup process. Normal user processing can begin on the original data.

The shell script in Figure 4 resets the logical volume structure to its original state. At this point, the system has returned to its pre-backup condition.

## AIX Book Corner

The following new books may be of interest:

*Power RISC System/6000: Concepts, Facilities, and Architecture*, by Dipto Chakravarty. New York: McGraw-Hill, Inc., 1994 (ISBN 0-07-011047-6).

Part of the J. Ranade Workstation Series, this book describes RISC System/6000 hardware and architecture, as well as AIX architecture and administration. The book is filled with helpful information and tips. Mr. Chakravarty has been a contributing author to *AIXpert*.

*The Whole Internet User's Guide & Catalog*, by Ed Krol. O'Reilly & Associates, Inc., 1994. Second Edition (ISBN 1-56592-063-5).

This edition is an update to the popular first edition that was published in September 1992. It is divided into three sections: history and ethics of the Internet, Internet tools, and the catalog of information available on the Internet. The new edition includes looks at Mosaic, mail programs, ftp servers, xarchie, Gopher, and much more.

## Caveats to the Backup Process

This backup procedure has some areas that must be understood before it is implemented.

- The process uses commands that require root authority.
- Because of the actions of the `mount` command, the original filesystem with its copies must be unmounted before deleting the mirror copy. This process allows the `mount` command to mount the created logical volume cleanly.
- During the backup process, there is one less copy for the system to use in case of failure.
- It is important that no other user with root authority has access to the LVM commands, such as `crfs`, `chfs`, or `mklv`, that can change the structure of the filesystems. The state of the free pool of physical pages must be constant during the critical phases of deletion and re-creation of the logical volumes.
- Once the mirror copy has been re-created on the logical volume, the `syncvg` command will resynchronize all physical partitions in the new copy, even though limited updates may have occurred during the backup process. To LVM, the mirror copy is completely new and all data on it is considered stale. The `syncvg` process can create a large processor load and will cause system response times to degrade. The amount of time spent doing the `syncvg` will depend on the speed of the processor, the load on the system, and the speed and configuration of the disk subsystem. Still, during this portion of the process, users can continue to use the system normally.


## Acknowledgement

I wish to thank Julie Craft from the development group for her assistance in creating the shell scripts used in this method.



**Jaime Vazquez**, IBM Corporation, RISC System/6000 Division, 11400 Burnet Road, Austin, TX 78758. Internet: jaime@austin.vnet.ibm.com. Mr. Vazquez currently is working in the OEM Consulting Services department. He is a graduate of Texas A&M University with a Master of Computing Sciences degree.



## IBM First in U.S. Patents Awarded

IBM ranked first in the number of patents awarded in 1993 by the U.S. Government, marking the first time since 1985 that an American company has headed the list.

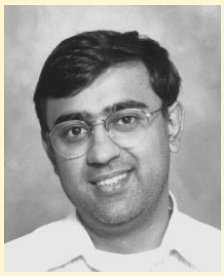
IBM received 1,088 patents, followed by Toshiba® Corporation, Canon® KK, and Eastman Kodak® Company, according to IFI/Plenum Data Corporation. In 1992, the first four companies were all Japanese.

The IBM patents were almost exclusively in the field of information processing, with an increase in software-related inventions. ■


# AIX Questions

**Compiled by Ismat Dhanjibhai**

The AIX Solution Provider Technical Support group in Austin, Texas, supports software vendors who are porting applications to AIX. This article is a compilation of questions that are frequently asked by vendors. The name of the responding Technical Support Group staff member appears after each response.



Ismat Dhanjibhai

**I am having a problem making a `mksysb` tape. The errors are as follows:**

**Error:*.add not found bosbot:boot image is 3124 512 byte blocks, mkinstal tape failed. 0512-0016 mksysb: attempt to create a bootable tape failed;**
**mkinstal tape /dev/rmt0.1 failed with return code 5.**

You are using your own `restore` command instead of the system `restore` command. Changing `PATH` to access the system `restore` (`/etc/restore`) before your own will solve the problem.

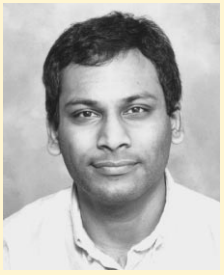
*—Sue Lowe*

Sue Lowe

**Is it possible to increase the 1 MB circular-error log file?**

First, run the command `/usr/lib/errdemon -l` to check the current maximum `errlog` size in bytes. Then run the command with the `-s` flag and specify the new size.

*—Sue Lowe*



Sai P. Ramanath

**What is the meaning of the "< >" symbols in the `/bin/oslevel` output (such as < >3250)?**

The default output compares the system software maintenance level to the currently applied maintenance level. For example, the shorthand for the 3240 maintenance level is as follows:

| Symbol | Description |
|---|---|
| =3240 | Equal to |
| <3240 | Below |
| >3240 | Above |
| <>3240 | At mixed levels |

The following options can be specified to determine which products and subsystems differ from the currently applied maintenance level.

`-l`: List products at levels earlier than maintenance level

`-g`: List products at levels later than maintenance level

`-e`: List products at current maintenance level

*—Sai P. Ramanath*

**Is there a penalty for compiling an executable with -bmaxdata to allow 2 GB of data? By "penalty," I mean does the binary start much slower than if it is compiled with -bmaxdata:0x50000000? Or is there no penalty to compile with -bmaxdata:0x80000000? If there is no penalty, why doesn't the default compiler allow every executable to be a maximum of 2 GB in data size? Why do I have to change the switch?**

When the option `-bmaxdata:` is used, all static data (whether initialized or not) is grouped together. It is then placed into shared memory segments that begin at address 0x3000000. If the data extends past the maximum size of a memory segment (256 MB), then a second, contiguous segment is constructed beginning at address 0x4000000, and

so on. Up to eight segments (2 GB of virtual memory) can be constructed in this manner. The shared memory segments are private—not accessible to other processes.

Real memory/paging space is assigned when the memory is actually used; if it is never accessed, it will not be assigned. Keep in mind that initialized data is "accessed" when the executable is started.

In general, there is little overhead when using the large data model. More virtual memory objects will be used, but the system should have plenty. The large data model is not the default primarily because the upper memory segments are needed for other purposes, such as shared-memory interprocess communication.

Any application that contains many forks will experience additional overhead because the most expensive part of the fork operation is duplicating the process address space—the `vmm_forkcopy` operation must be done for every memory segment that is allocated.

**Note**: The `maxdata` option only tells the linker where (potentially) to place the data. The actual data requirements are still the critical issue when running the program.

*—Sai P. Ramanath*

**I am using the C++ compiler with the +- option. A note on page 118 of the C++ compiler manual states that if I use this option, `dbx` will not recognize files ending with ".c".**

The `dbx` has been modified with Basic Operating System (BOS) 3.2.5 to work with all extensions except `-o`, `-a`, or `s`. It now allows for debugging programs without a `.c` extension that were previously compiled using the `+-` option.

*—Jeff Simon*

**Is the RISC System/6000 internal CD-ROM compliant with Small Computer Systems Interface-2 (SCSI-2)?**

CD-ROM type A is SCSI-1 and type B is SCSI-2. If your RISC System/6000 has a SCSI-1 adapter, both type A and type B drives operate as SCSI-1 devices. If your RISC System/6000 has a SCSI-2 adapter, the type A drive operates as a SCSI-1 device and the type B drive operates as a SCSI-2.

*—Priyamvada*

**How can I determine the size of the dump device?**

The `sysdumpdev -e` command returns the size of the dump in bytes. The dump device driver calculates the size of the dump as follows:

- Size of the Virtual Memory Manager (VMM) dump data (0.025 * actual memory)
- Size of memory allocated to VMM segment `(vms_rusage(vmmsrval) * PAGESIZE`
- Size of memory allocated to kernel extension segment `(vms_rusage(kexsrval) * PAGESIZE)`
- Size of memory allocated to kernel segment `(vms_rusage(kernsrval) * PAGESIZE)`

The total of all these sizes is returned to `sysdumpdev`. The estimated size of the system dump is that total plus 25%.

*—Priyamvada*

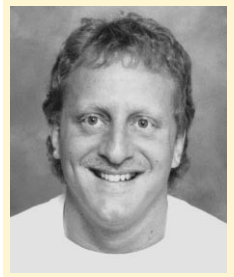
**I am using `/etc/security/audit/objects` to perform an audit on some object files. I can't generate `S_PASSWD_WRITE`. I use emacs for day-to-day editing, and this only logs an `S_PASSWD_READ` event.**

The `vi` editor available from IBM is integrated with the auditing subsystem. Non-IBM products such as emacs may not fully integrate with the auditing subsystem. Since the emacs you are using is not integrated, you do not get all the events. A workaround is to enable auditing of file writes via the kernel events.

*—Ismat Dhanjibhai*

**The `pcnfsd` is in the `/etc/inetd.conf` file (and it is commented out). When my machine boots (it is running AIX 3.2.5) or when I refresh the `inetd`, the `pcnfsd` does not start. Do I need to do something different?**

Figure 2 shows the correct way to verify whether `pcnfsd` (RPC program number 150001) is running. The file `/usr/sbin/rpc.pcnfsd` is part of the `bosnet.nfs.obj` program. Output from Figure 2 should look similar to Figure 3.



**Jeff Simon**

**Priyamvada**

1. vi /etc/inetd.conf and uncomment (remove the "#") the following entry:

```
pcnfsd sunrpc_udp udp  wait  root  /etc/rpc.pcnfsd pcnfsd 150001 1
```

2. > refresh -s inetd

3. > rpcinfo -p

**Figure 2. Verifying whether pcnfsd is running**

```
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100001    1   udp   1033  rstatd
100001    2   udp   1033  rstatd
100001    3   udp   1033  rstatd
100024    1   udp    849  status
100024    1   tcp    851  status
300082    1   udp    854
300082    1   tcp    856
100021    1   tcp    683  nlockmgr
100021    1   udp    685  nlockmgr
100021    3   tcp    688  nlockmgr
100021    3   udp    690  nlockmgr
100020    1   udp    693  llockmgr
100020    1   tcp    695  llockmgr
100021    2   tcp    698  nlockmgr
150001    1   udp   1128
```

**Figure 3. Output from the rpcinfo -p command**

The 150001 entry shows that the `rpc.pcnfsd` is running and listening. To verify this, enter the following command:

```
rpcinfo -u <local hostname> 150001
```

This polls the `rpc.nfsd` process to verify that it is running. The following output is produced:

```
program 150001 version 1
        ready and waiting
program 150001 version 2
        ready and waiting
```

*—Rick Malone*



**When a Network Computing System (NCS) server is busy with an NCS client and a client is waiting, does the client send the data more than once? How frequently does a client send the data to the server? Is it possible when the client resends requests many times that the data size exceeds the number of `mbufs`?**

Yes, when a client is waiting, it can send its data more than once. The mechanism is as follows:

When a client request times out, the client NCS runtime attempts to ping the server with a NULL RPC. If the ping succeeds, the client resends the data.

The ping's success or failure is determined differently for two separate cases: when normal (long) timeouts are set and when short timeouts are set. In the former case, the client sends Remote Procedure Calls (RPCs) to ping the server at one-second intervals up to 30 times. In the latter case, only one ping (with a one-second timeout) is sent. A client that times out and resends requests will not cause a server machine to run out of `mbufs` since the vast majority of the RPCs sent are "ping" RPCs, with only a header and no body (data).

*—Venkatesh M. S. Iyer*

**Ismat Dhanjibhai**, IBM Corporation, RISC System/6000 Division, 11400 Burnet Road, Austin, TX 78758. Internet: ismat@austin.ibm.com. Mr. Dhanjibhai, a member of the AIX Solution Provider Support Center, currently provides technical support to software vendors. Mr. Dhanjibhai has a degree equivalent to Computer Studies from the Polytechnic of Wales in the United Kingdom.

# Open Blueprint: IBM's Guide to Distributed Computing


**The Open Blueprint, a comprehensive structure of components, makes it easier than ever to design, implement, and manage industry-standard client/server applications.**

The Open Blueprint offers a set of guidelines that will make it easier for software developers—whether they are at IBM, other software vendors, or other system vendors—to produce products that are truly interoperable," said Martin C. Clague, IBM's general manager, Worldwide Client/Server Computing. "Businesses want to know that their investments in information technology will endure as they transform their systems to more productive client/server models. The real beauty of the Open Blueprint is that it gives them the confidence to proceed knowing that we understand what they want and their investments are secure."

IBM has worked closely with international standards organizations, industry consortia, customers, and industry leaders to determine the most widely accepted standards available today that allow IBM products to interact with other vendors' products and with each other.

Because IBM product divisions support the Open Blueprint, customers will be able to develop new client/server applications while preserving their existing applications.

## Overview

The Open Blueprint addresses the challenges of the open environment by viewing a system as part of a distributed network and viewing the network as if it were a single system. The Open Blueprint serves four major roles:

- Helps customers develop their own architectures and organize products and applications in an open distributed environment
- Describes IBM's directions for products and solutions in the open distributed environment
- Guides developers as they supply products that include the appropriate functions, and products that can be integrated and interoperate with other installed products
- Provides a context for incorporating new technologies into a distributed environment

A major goal of the Open Blueprint is to enable a single-system view of the network, masking the complexities of the physical network environment from users. The Open Blueprint structure allows a network of operating systems to function as a unit—as a network operating system. A *network operating system* consists of multiple separate systems connected by a network. In the network operating system enabled by the Open Blueprint, each individual system logically contains the services described below. However, it is not necessary for each individual system to physically contain all the services included in the Open Blueprint.

Just as an operating system manages resources on a single system, a network operating system manages the same types of resources (files, databases, printers, transactions, software packages, documents, jobs, and so on) across a network. The equivalent facilities or services in each individual system work together to provide support for distributed and client/server applications. Figure 1 represents one instance of a system in the network operating system.

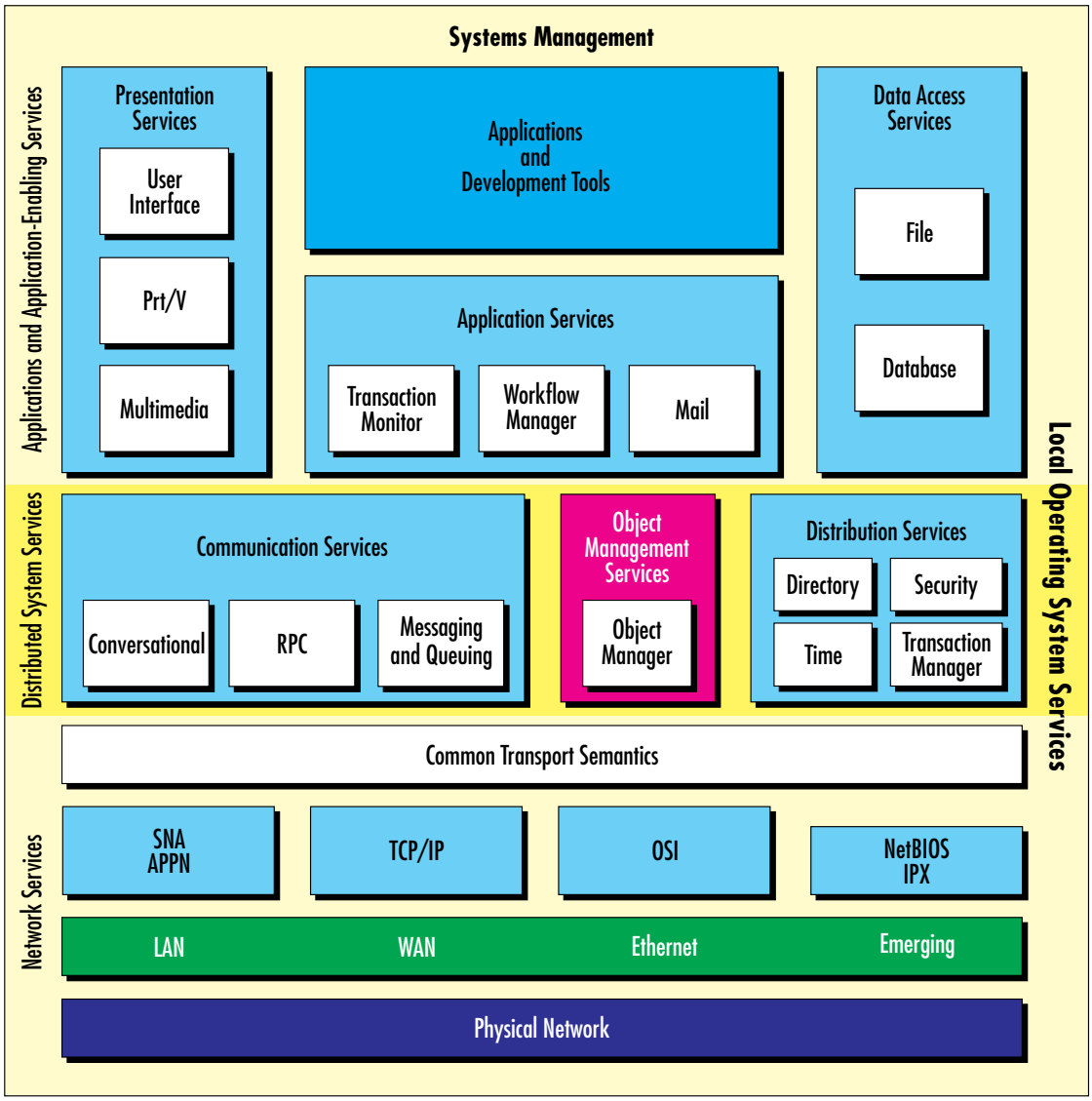



**Figure 1. The Open Blueprint**

Figure 2 shows the sets of resource management services defined in the Open Blueprint and shown in Figure 1.

The Open Blueprint promotes the integration of multivendor systems and simplifies the more cumbersome aspects of distributed computing, such as multiple logons, multiple passwords, and unique application directories for locating resources. Products that align with the Open Blueprint provide designated interfaces and protocols. Products that use resource managers defined within the Open Blueprint rather than their own unique mechanisms are truly integrated with the Open Blueprint. For example, if every application uses its own unique security facility, a user must present a unique password for each application. If applications and resource managers use the security services from the Open Blueprint, a single user password will suffice for the cooperating applications. This integration improves the single-system image of the distributed system as perceived by the end user and application developer.

The Open Blueprint includes interfaces and protocols from IBM and other industry sources. Those from industry sources are broadly accepted standards, many of which are included in X/Open's® Distributed Computing Services (XDCS) framework.

The boxes in Figure 1 do not correspond to specific products. The Open Blueprint is imple-

mented by different products on different system platforms. Although the Open Blueprint does not describe how the implementing software is packaged into offerings, it does describe the technical attributes and characteristics of supporting software, reflects desirable functional modularity, provides software principles and guidelines, and specifies important boundaries and interfaces.

The Open Blueprint describes techniques for building an open, heterogeneous, distributed system that is extensible by alternate component implementation and by support for evolving new technologies and functions. For example, the Open Blueprint currently supports three models for interprocess communication: conversational, Remote Procedure Call (RPC), and messaging and queueing. If a fourth model were to be developed, it would be evaluated for inclusion. Because the Open Blueprint is modular, the fourth model would easily fit in with the other three. However, if a new technology emerged in the area of one of the existing models, the Open Blueprint would not have to be modified. The new technology would potentially have a great impact on implementation and integration, but it is handled architecturally within the existing structure.

Existing products currently provide many functions described in the Open Blueprint, but additional functionality and product implementations will be provided to the Open Blueprint. Because standards are included in the Open Blueprint, components from different providers can be mixed and matched in the distributed network.

Object technology is key to the evolution of the Open Blueprint. Objects are expected to become more prevalent in applications and in system and network components. Coexistence with procedural elements is accommodated in the structure of the Open Blueprint. Basic object management services will be extended and expanded to satisfy both transitional, mixed environments, and full object-oriented implementations. Over time, many parts of the Open Blueprint will be affected by object technology, including new object-oriented services and new interfaces for existing services.

## Summary

In their book *Paradigm Shift* (McGraw-Hill Inc., 1993), Don Tapscott and Art Caston summarize the need for standards as follows:

> Leading organizations everywhere are embracing the concept of a standards-based architecture. Such architectures are modular, flexible, enterprise-wide, based on standards rather than specific vendor products, and owned by the customer, not the suppliers. This enables the IP infrastructure required for competitiveness and success in the new business environment.

Tapscott and Caston list ten technology themes that enable businesses to transform themselves

### Network Services

- **Common Transport Semantics** supports protocol-independent communication in distributed networks.
- **Transport Network Services** provide the protocols for transporting information from one system to another, such as System Network Architecture/Advanced Peer-to-Peer Networking (SNA/APPN), TCP/IP, Open Systems Interconnection (OSI), NetBIOS, and IPX.
- **Subnetworking** provides functions dealing with specific transmission facilities, such as various kinds of LANs, WANs, channels, and emerging technologies such as wireless and Asynchronous Transfer Mode (ATM).

### Distributed System Services

- **Communication Services** provide mechanisms for parts of a distributed application or resource manager to talk to each other.
- **Object Management Services** provide transparent access to local and remote objects.
- **Distribution Services** assist the communication between parts of distributed applications and resource managers by providing common functions such as a directory and security.

### Application-Enabling Services

- **Presentation Services** define the interaction between applications and the user.
- **Application Services** are common functions (such as mail) that all applications can use.
- **Data Access Services** allow applications and resource managers to interact with various types of data.
- **System Management Services** provide either facilities for a system administrator or automated procedures to manage the network operating system.
- **Local Operating System Services** operate within the confines of a single system in a network, such as managing memory and dispatching work.
- **Development Tools** help the application developer implement distributed applications that use standard interfaces.

**Figure 2. Sets of resource management services**

## How to Order IBM Publications

IBM documents, publications, CD-ROMs, brochures, videos, and other types of material can be ordered by phone or fax. Prices for documents vary, with an average of $25. Orders can be placed from 6:30 A.M. to 5:00 P.M. MST. Publications can also be ordered through your IBM branch office.

### By fax:

- Obtain a form from your IBM branch office, or create your own that includes the following information:
  - Publication numbers for all items you are ordering
  - IBM customer number if you have one
  - Credit card type (VISA, MasterCard, Diners Club), number, expiration date, and name on the card
  - Your name, phone number, and fax number
  - Ship-to address, including the name of the person to receive the order
- Fax to (800) 284-4721, toll-free in the U.S.

### By phone:

- Have the above information available.
- Call (800) 879-2755, toll-free in the U.S to reach IBM Software Manufacturing Solutions.
- Select option 1 and follow the directions.

into responsive, service-oriented, and profitable firms. These themes include open systems, distributed computing, interconnection, user friendliness, global networking, and architectural modularity. This last theme is described as follows:

> Rather than being a giant monolith, computing architectures consist of standardized, independent parts that can be grouped together as required to meet business requirements. These can include standard computing platforms, or interchangeable application software components. The goal is to create a dynamic, flexible computing environment...

The Open Blueprint is IBM's view of a modular architecture. Many IBM customers are using the Open Blueprint as a guide for creating their own internal architectures for deploying applications in the open, distributed environment. It will help IBM and others deliver integrated, interoperable solutions with the following benefits:

- **For end users**: Open Blueprint integration hides the complexities of the network and makes it appear as a single system.
- **For application developers**: Standard interfaces enable a single system view of the network and allow for the development of interoperable applications that can run on many platforms.
- **For system administrators**: The Open Blueprint defines a consistent way to manage the network to hide the complexities from application developers and end users.

## References

The Tapscott and Caston book gives a complete look at how to re-invent, re-engineer, or re-architect your enterprise through the transition from host-centered, mainframe-based computing to user-centered, network-based computing. It makes the case for complete planning and for adopting an architecture such as the Open Blueprint that is based upon standards. Other valuable references include the following (see the box for ordering instructions):


- Hammer, Michael and Champy, James. *Re-engineering the Corporation.* Harper Collins, 1993.
- *IBM Networking Blueprint Executive Overview* (GC31-7057).
- *IBM SystemView® Concepts* (SC23-0578).
- *IBM SystemView Structure* (SC31-7038).
- *Information Warehouse™: An Introduction* (GC26-4876).
- *Multimedia Distributed Computing—IBM's Direction for Multimedia Distributed Systems* (G229-7340).
- *Open Blueprint Technical Overview* (GC23-3808, available 2H94).
- *The Networking Blueprint* (SX33-6090).
- X/Open Company Ltd. *Distributed Computing Services Framework*, November 1992 (ISBN 1-872630-64-2).