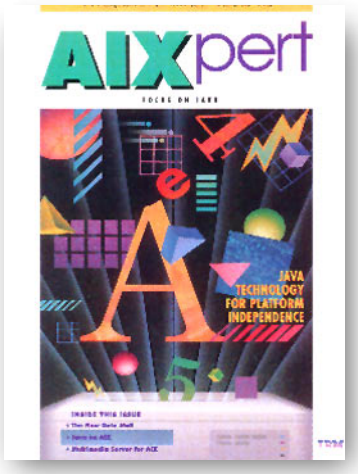
AIXpert DIGITAL VERSION IBM

TABLE OF CONTENTS



NOVEMBER 1996

AIXpert BACK TO MAIN MENU

This time I mean it!!

It's quite possible that in my last editorial I had mentioned that it was my last editorial; well, I was wrong. *This* is my last editorial. So, presumably, if you have read my last editorial, you needn't read this one, since this is it.

Many people have written to point out that there didn't seem to be a summer issue of the magazine this year. Of course that came as a complete surprise to me, since I had already embedded myself in worrying about the latest World Wide Web technologies (if that allusion puzzles you, read my last editorial) and had assumed that *AIXpert* was running smoothly without me. But it appears that my correspondents were correct, and for that I apologize.

To atone for this minor mishap (how could *that* many people be correct), we've put together an eclectic mix of topics for this issue that will keep you interested from cover to cover. Our cover story, *Java on AIX—A Strong Brew*, introduces this hot, object-based, newcomer that allows your programs to run on lots of different platforms. Read about its history, how to use it, its secure features and where to go for even more information, in a cohesive, easy-to-follow article.

Following in that vein, we cover hot object technologies with a Direct-to-SOM C++ compiler article, explore the Internet realm with articles on the multimedia server and Internet address management, and delve into the programming interfaces found in the Common Desktop Environment (CDE).

We look inside a scalable data warehousing system, and we show you how to set up HACMP/6000 for disaster recovery using responsive across-town or across-the-sea clustering configurations. We help you understand performance characteristics of the popular symmetric multiprocessing systems, and we tell you how to get a new tool to help measure performance on any AIX system. We also tell you how to get AIX telephone support and conclude with the popular Question and Answer column.

A Parting Note

Just a reminder, *AIXpert* is available electronically—hosted, of course, on the IBM Developer Support home page (http://www.developer.ibm.com/). So be sure to stop by for a visit.

George Noren

George Noren, IBM Corporation, Internal Zip 4103, 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

The Sybase/IBM Data Mall

By Ron Wastal

The Sybase/IBM Data Mall architecture gives organizations the ability to deliver rapid, new business value and at the same time build collaborative and effective data management and movement strategies. It delivers fast and cost-effective interactive "store fronts" with a high-quality and manageable infrastructure "warehouse."

Data warehousing implementations are only as good as the quality and timely information they deliver to business users. Balancing the corporation's need for centralized coordination and control with each department's need for distributed empowerment and execution can be difficult.

With Sybase's parallel data warehouse database (Sybase MPP™) on the IBM RS/6000™ SP, along with Sybase's new interactive data mart database (Sybase IQ™) on the new RS/6000 Symmetric Multiprocessor (SMP) Server, combined with leading edge middleware products (Sybase® Enterprise CONNECT™), users and Information Technology (IT) professionals can now get their data warehouse "cake" and "eat it too."

Rethinking the Data Warehouse Concept

Data warehousing is an acknowledged technology in the corporate arena, accepted by customers, consultants, and vendors as a unique tool for achieving competitive advantage in the global marketplace. However, delivering on the promises of data warehousing, including more effective target marketing, more precise financial forecasting and budgeting, more efficient channel and logistics management, and more intimate customer service, can be a daunting challenge for IT executives.

A data warehouse was originally defined as a subject-oriented, integrated, time-variant, non-volatile collection of data that is specifically collected to aid management decision-making processes. The promise was to deliver information on how to discover or respond to a product opportunity quickly, a competitive threat or weakness immediately, or understand new opportunities and marketplace trends very early in the cycle. This was done by mining enterprise data stored by various departments within an organization. The principle of the data warehouse was to put all this data into a common repository—the data warehouse—thereby making it more valuable as a business information resource versus a departmental operational resource.

Unfortunately, while the IT group set out to build a common resource accessible to thousands of users with a wide variety of skill levels and interests, the business users (under immense business pressure) did not wait, deciding instead to implement their own departmental data warehouses, later termed "data marts."

Data Warehouse Evolution

First generation data warehouses were usually developed around centralized, mainframe-based architectures that supported both transactional production systems and decision support. This architecture proved to be unworkable from a practical perspective since the data models defined in the warehouse were often very complex. In addition, with Decision Support Systems (DSS) and Online Transaction Processing (OLTP) requests competing for the same computing resources, the systems were unable to deliver satisfactory performance.

Second generation data warehouse systems evolved when IT departments consolidated data onto a single mainframe platform or commercial parallel processing systems. At the same time, independent departments or divisions pursued their own mini-warehouses (data marts) to support their analytical needs.

Data marts were especially attractive to end users since they were often easier to fund, faster to implement, and focused on delivering specific business value to a targeted set of knowledge workers. Information in a data mart is consolidated from production-level information organized to directly reflect the departmental understanding of business process, workflow, and so on. Data marts can be built rapidly but, over time, must be implemented to share common data elements, transformations, and management tools.

Now, a third generation of data warehouses is being constructed, with emphasis on creating a cooperative environment between the corporate data warehouse and the individual data marts. In this Data Mall approach, data marts are built rapidly but with an architecture that promotes sharing of common data elements and transformation programs. Over time, as multiple data marts go into production, the enterprise warehouse becomes an optional port of the architecture to hold large amounts of commonly accessed data.

This hybrid model focuses on delivering high performance, system scalability, and manageability at both the warehouse and data mart level, while also delivering highly responsive and flexible query performance to satisfy the expanding needs of end users. This last requirement is crucial since a resource like a data mart is difficult to model over time—users often do not know what they want to do with the data mart until the data is in their hands. This means that the interface tools and underlying architecture of the data mart must permit users to explore the data interactively, learning and discovering about the resource available to them.

Specifically, Sybase MPP running on the IBM RS/6000 SP provides a scalable architecture that supports the data intensive, high-capacity, high-volume batch and pre-defined query processing typical of very large data warehouses. Simultaneously, Sybase IQ running on a IBM RS/6000 or RS/6000 SMP Server delivers exceptionally fast query response in a flexible, ad hoc manner that addresses the demanding information analysis needs of the data mart user.

The Data Mall—The Next Generation

A *data mall* is a data warehouse/data mart environment combined under a single system roof. It uses the unique architecture of the IBM RS/6000 SP to prototype, test, implement, and if necessary, distribute these rapidly built data marts in a manageable fashion. IT managers can now optimize and tune their data mart structure and evaluate LAN/WAN traffic concerns before distributing systems. This assures end-user satisfaction and provides faster, more reliable implementations. With Sybase's optimized database middleware (Enterprise CONNECT) and server products coordinating the warehouse (Sybase MPP) and data mart (Sybase IQ), system administration requirements are greatly reduced and total system integration is simplified. The following sections explore this solution in greater detail.

A data warehouse decision support environment dictates performance requirements in five key system evaluation areas: scalability, query performance, capacity, availability, and administration.

IT managers can now optimize and tune their data mart structure and evaluate LAN/WAN traffic concerns before distributing systems.

Scalability

The continuing acceptance of decision support systems is fueling a rapid growth in database size. Surveys indicate that a significant percentage of companies have started or intend to implement a data warehouse system in the next few years. In addition, companies continue to store increasing volumes of production and transaction data as their data-capture facilities improve.

With current data warehouses managing hundreds of gigabytes to a terabyte or two, most industry experts predict that future warehouses will be forced to manage multiple terabytes of data. By logical progression, data marts must also be prepared to scale, handling additional capacity from the megabyte- and gigabyte-class loads they deliver today.

Query Performance

Data warehouses and data marts often need to support three broad classes of queries:

- **Interactive, ad hoc queries:** Read-only requests accompanied by expectations of near real-time response where an analyst explores data via a series of drill-down or drill-up queries, searching through various levels of data, usually covering narrow time frames or search categories.

- **Complex queries:** Processing requests that cover large amounts of data spanning broad time periods or product/service categories. These computationally intensive requests typically include join, sort, or group-by operators and are, in some cases, performed on a repetitive basis. In spite of the complexity, it is assumed that rapid turnaround is provided, since information value (accuracy) is related to the timeliness of the analysis.

- **High-speed, mixed workload queries:** Requests that require near real-time response and read/write support across multiple platforms. Typical applications are online customer support or order entry, where an operator will need to quickly search a customer record from a large list, then modify and enter data, or access a more complex data resource such as a Help Desk Q&A record.

These variations describe highly variant performance demands and often require different systems to provide effective support to diverse end-user requirements.

Sybase MPP is an extension of the Sybase SQL Server technology specifically designed and optimized for massively parallel processing environments.

Capacity

Both data warehouses and data marts are typically updated on a batch basis, using periodic, scheduled loading times. Routine operating requirements, such as backups, must be scheduled. Since the update process involves large data blocks, the information architecture must provide sufficient load/update capacity to handle the batch requests in the allotted time period.

Availability

Construction of a data warehouse and data mart environment requires a significant investment that only returns value when it is operating. While the system hardware/operating platform and database may not require total availability (fault tolerance), it should have high availability with failure recovery mechanisms that let users continue to operate even if a component of the system is down.

Administration

Maintaining data integrity and availability in a distributed environment can be difficult. For example, data marts should be continually re-evaluated for their effectiveness; that is, can the data mart support greater than 95% of user queries, or do some routine drill-through (back to the warehouse) or drill-across (to other data marts) requests place excessive demands on LAN/WAN networks or high-speed interconnect capacity (in the data mall), and burden the data warehouse with ad hoc requests?

Sybase and IBM

Sybase and IBM offer a unique suite of complementary products that deliver the performance and availability that users demand.

Sybase MPP and IBM RS/6000 SP

Managing the capacity demands of a data warehouse while offering superior data mining and analysis capabilities on an open platform can seem like a big challenge. Traditional parallel database systems use multiple processors to access data using either shared memory or shared disk architectures. This approach, however, limits performance, especially as the database scales in size. In fact, the shared-nothing architectures of current generation massively parallel hardware can support data sizes and throughput rates an order of magnitude greater than shared memory and shared disk databases. Because of software limitations, massively parallel hardware for database applications has been limited to niche applications.

The RS/6000 SP represents a next-generation hardware design that uses widely available components and operating systems to provide massive processing power, enormous scalability, and exceptional availability in a highly affordable package. This makes parallel database architectures affordable in massively parallel environments. However, parallel databases do not always take full advantage of the RS/6000 SP architecture.

Sybase MPP is an extension of the Sybase SQL Server™ technology specifically designed and optimized for massively parallel processing environments. The open, portable, parallel architecture of Sybase MPP delivers unmatched performance, unlimited scalability, and database management utilities to support Very Large Database (VLDB) operations. Sybase MPP uses multiple SQL Servers, each accessing one or more processors, to implement a parallel shared-nothing database environment.

Performance

Sybase MPP delivers superior performance for pre-defined and batch queries.

- Complete, parallel, shared-nothing architecture takes full advantage of the massively parallel

design to deliver exceptional processing performance by minimizing system bottlenecks such as memory or I/O contention.

- Integrated system utilities simplify and speed system tuning for optimum use of hardware resources.

- Fully parallel operations—from utilities and queries to data operations such as insert, update, or delete—deliver flexible performance to support end users while handling real-time systems management and data updates.

- Parallel loading (parallel bcp) reduces load times on bulk updates and assures timely availability for user queries.

Scalability

Sybase MPP scalability makes it ideal for growing data warehouse facilities.

- Design scales to near linear levels in both response time (speed up) and system throughput (scale up) as additional processors, workload, and data are added to the system.

- Parallel performance enhancements are transparent to users, protecting investments in applications and training.

- Sybase Open Client/Open Server architecture supports hundreds of existing end-user analysis tools.

VLDB Management Utilities

Integrated management utilities simplify delivery of parallel performance using Sybase MPP Manager and Configuration Services.

- Configuration Services helps customers analyze workload requirements and recommends optimal software and hardware configuration.

- Sybase MPP Manager provides graphical utilities to help administer the system.

- MPP incorporates tools for increased availability, including automatic failure recovery from disk, process, or system node errors (hierarchical monitoring).

The RS/6000 SP is a general-purpose, scalable parallel system that can incorporate between 2 and 512 nodes. Each RS/6000 SP node—packaged up to 16 per frame—is a self-sufficient system, containing local memory and disk capacity, and using the latest PowerPC™ or POWER2 processors. Each node runs its own copy of AIX® and other RS/6000 system software.

The heart of the IBM RS/6000 SP is the optional SP switch, a high-bandwidth, low-latency switching network that binds the nodes together. The switch incorporates a unique combination of topology and architectural features to scale aggregate bandwidth as the system grows. As nodes are added to an SP, switch performance scales at a near linear rate, which delivers peak node-to-node bandwidth for TCP/IP traffic of 70 MB per second, and bandwidth in excess of 100 MB per second for Message Passing Interface (MPI) protocol.

On a 16-node SP system, this equates to an aggregate bi-sectional bandwidth of over 800 MB per second. The SP switch design has point-to-point interprocessor communication time independent of the relative location of the communicating node. The resulting near linear scalability is unmatched by direct networks such as rings and meshes.

The RS/6000 SP's performance-oriented architecture is designed to support large-scale data warehouses.

Optimized for Data Warehouse Applications

The RS/6000 SP's performance-oriented architecture is designed to support large-scale data warehouses. On-chip and off-chip caches compensate for the highly mixed workloads and data sets typical of RDBMS applications. This ensures that processor cycles are used for database operations, not for retrieving frequently used instructions and data. Asynchronous I/O minimizes waiting for disk or network availability.

To optimize decision support, AIX has been designed to reduce lock overhead, allowing greater simultaneous system access. A dynamic kernel eliminates the need to take servers off-line for many routine configuration changes. The Logical Volume Manager (LVM) enables simplified filesystem expansion, data distribution, and mirroring.

In addition, the RS/6000 SP delivers unsurpassed availability by extending the benefits of conventional High Availability Cluster Multiprocessing (HACMP) architecture, including automatic workload balancing in the event of a node failure, to its full architecture. Instead of the standard 8-way limitations of HACMP structures, the RS/6000 SP can extend this to its full 128-node (or higher) architecture.

Data Marts on Sybase IQ

While parallel database technology is ideally suited to managing large capacity and complex queries, this database architecture often has difficulty scaling concurrent, mixed-user workloads. Flooding a parallel system with continuous interactive queries can drain resources and slow response time on all processing functions. The logical evolution is to pump data from the main data warehouse into smaller databases (data marts), designed specifically to support the needs of discrete end-user constituencies. These individual data marts can then be distributed to handle ad hoc queries of individual knowledge workers.

Sybase's new interactive database, Sybase IQ, is a tool for managing interactive queries in a high-volume, rapid response environment. IQ's technology reduces the amount of work per query required by the system, providing query performance improvements 5 to 500 times faster than traditional RDBMS systems. Plus, with Sybase IQ, the system can be tuned once, versus tuning for each query; this simplifies setup and administration. Coupled with the IBM RS/6000 SMP Node, Sybase IQ delivers rapid response at a fraction of the cost of stand-alone data warehousing solutions.

Sybase IQ can scale the data mart to support from tens to thousands of users on a standard SMP platform.

The key to the breakthrough performance in Sybase IQ is Bit-Wise™ query processing. A step beyond traditional indexes or bit-map indexes, Bit-Wise technology represents all values as bits in indexes. A comprehensive five-part indexing scheme manages various data types, optimizing the indexing scheme against cardinality. Indexes can be applied concurrently versus traditional relational databases that are generally limited to one index per query.

Sybase IQ's high cardinality, bit-wise processing approach provides rapid, dynamic aggregation of relational data, delivering vastly improved query response. Column-wide data storage and compression also provides up to 98% reduction in disk I/O for fast response while eliminating table scans and unpredictable results. Hot column caching supports dramatic performance gains on iterative, drill-down/drill-up analyses. Sybase IQ's open architecture is also compatible with a broad range of front-end query tools.

The Scalable Data Mart

Sybase IQ has the unique ability to scale the data mart to support from tens to thousands of users on a standard SMP platform, such as IBM's RS/6000 SMP Server. By eliminating table scans and making decision support a CPU-bound rather than an I/O bound process, Sybase IQ allows the data mart to deliver predictable support for many more users.

The software also provides simplicity of management and flexibility. Data mart configuration designs are often based on perceived user requirements. However, once users have access to the data, their need to access data beyond the originally designed data mart becomes evident. Excessive drill-through or drill-across requests indicate non-optimized mart construction.

IQ simplifies reconfiguration by supporting heterogeneous data sources and delivers administrative tools for system configuration and optimization. Since tuning is data dependent, not query dependent, re-tuning of the system is required only when the data definitions are changed.

The Data Mall and Beyond

The scalable RS/6000 family enables the warehouse and mart environments to be implemented on a common, binary-compatible platform. This means you can grow a system from either direction: start with a data mart or two then upscale to a data warehouse, or start with a data warehouse and build/feed data marts.

The RS/6000 scales from uniprocessors to SMP to clustered processor configurations to massively parallel systems like the RS/6000 SP. Since the same architecture and operating system are used on all systems, application portability is enhanced, and system training and support requirements are reduced.

Some advantages of the RS/6000 SP go beyond performance. While a data mart is a powerful concept at the user level, maintaining synchronous data and assuring overall system performance presents a challenge. Using the RS/6000 SP, IT managers can prototype their data warehouse/data mart construction by creating a data mall—a data warehouse and data marts coordinated by middleware under a single system roof.

Figure 1 shows how most data warehouses and data marts are being implemented today—as competitors attempting to attract information shoppers. Data is delivered to both the data warehouse (driven by IT needs) and data marts (driven by users needs) from a variety of departmental operational systems. The data marts often receive the bulk of the "shopping" since they are designed around specific business needs, while the data warehouse struggles to compete (and fund its existence) due to the lack of user interest.

Figure 2 depicts an evolutionary scenario implemented on an RS/6000 SP, with multiple nodes dedicated to the core data warehouse. It is running a Sybase MPP parallel database with distributed RS/6000 servers running Sybase IQ to handle data mart queries from end-user workstations. This allows individual departments to have some level of autonomy and control over their local data mart and receive the dedicated query performance needed to

Current Implementation of a Data Warehouse

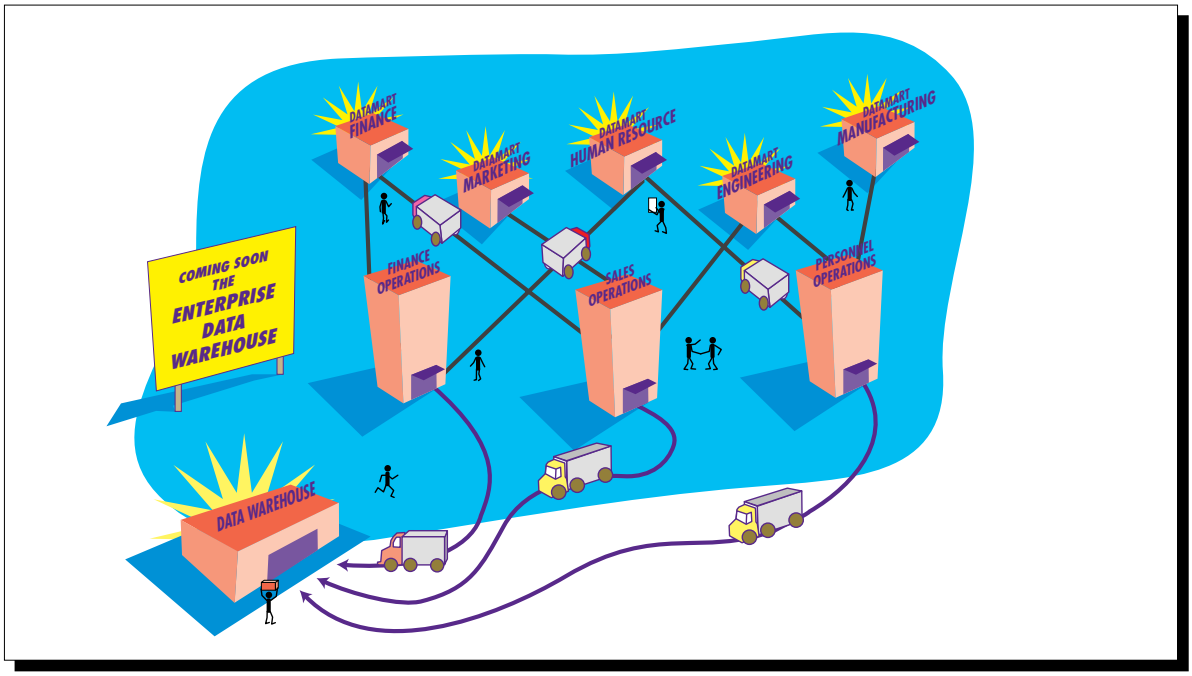


Figure 1. Current implementation of a data warehouse

Sybase MPP on RS/6000

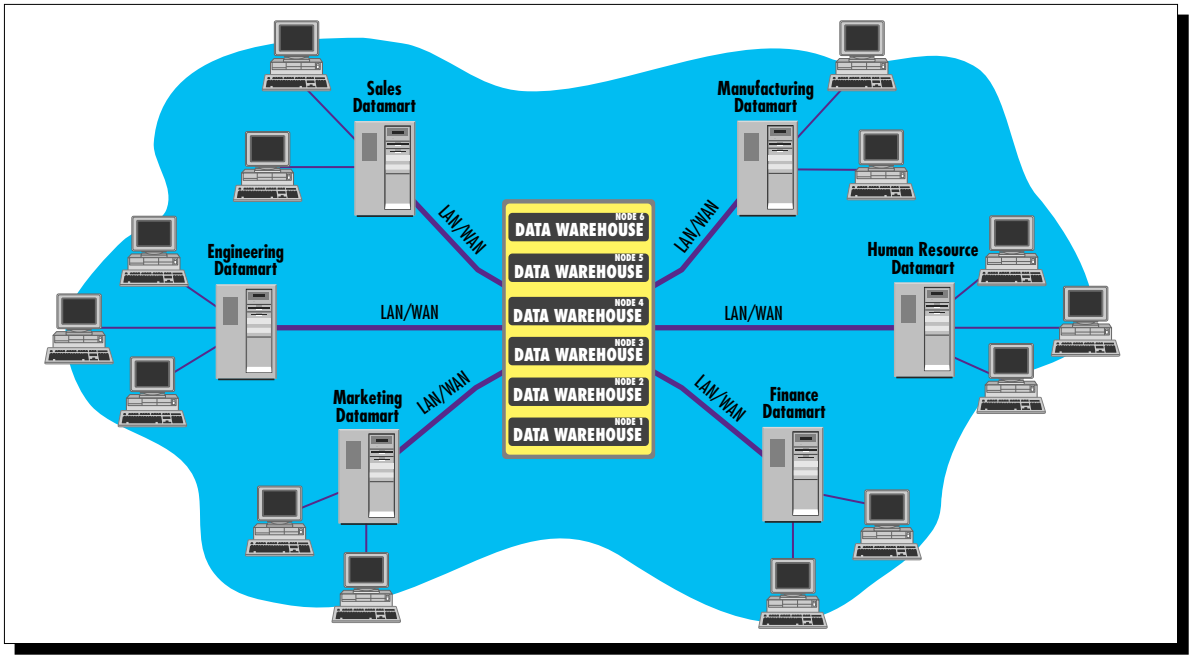


Figure 2. Sybase MPP on RS/6000 feeding RS/6000-based data marts running Sybase IQ

make the system valuable, all while maintaining centralized data warehouse quality and consistency.

Figure 3 introduces the latest evolutionary concept built around centralized coordination with distributed capabilities—the data mall. Imagine the impact of having a retail warehouse and individual storefronts all located under a single roof. Distribution is "hyper-accelerated" and simplified since distribution is just "down the hall" or over the interconnect. The data mall manager can clearly observe inventory traffic and evaluate demand at both the store and warehouse level.

The translation of this concept to an actual implementation on an RS/6000 shows how the individual data marts can be prototyped on an RS/6000 SP node. It is easy to clearly assess functional performance, data traffic requirements, and user demand before the system is considered for physical distribution to a remote field location. This permits proper optimization of the database design and preparation of the LAN/WAN for the new traffic patterns.

Eventually, distribution of the data mart might occur on a geographically remote stand-alone RS/6000. Since all RS/6000s are binary compatible, this transition is smooth, regardless of the prototype and distribution system configurations.

The concept of the data mall illustrates just one example of the potential that the combination of the RS/6000 SP with Sybase MPP and Sybase IQ offers those seeking to implement decision support strategies.

Essential Infrastructure in the Data Mall

Critical to the effectiveness of any data warehouse architecture is the controlled movement of data between individual data marts and the corporate data warehouse, or between individual data marts. Movement of data in any one of these architectures has the following characteristics:

- High volume, high performance
- Bulk load, unload
- Replication, update, refresh, enhance, expand, automated, and scheduled

Sybase InfoPump™ offers one way to accomplish data movement and materialization within a warehouse environment. InfoPump is a cross-platform tool (licensed from Platinum Technology®) that is used to automate the movement of bulk data on either a timed basis or through triggers that operate under defined criteria. InfoPump also has features that allow for the transformation of the data after it is captured in the source file and

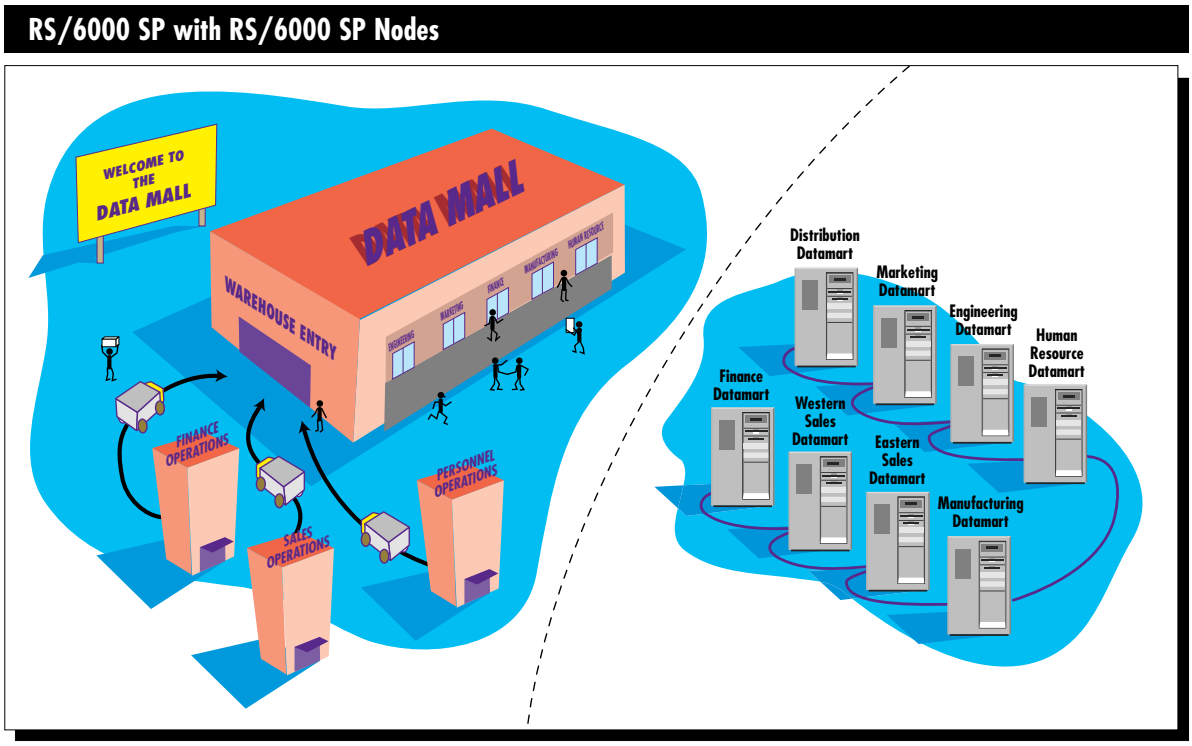


Figure 3. RS/6000 SP with RS/6000 SP nodes

Sybase on IBM Platforms

Sybase and IBM offer a complete set of solutions and services to address enterprise-class information services requirements:

Optimized Database Servers

- **Sybase IQ:** A powerful tool for data marts and interactive query support in a decision support environment; Sybase IQ runs on the IBM RS/6000 family and delivers rapid response for high-volume ad hoc queries; running on the IBM RS/6000 SMP systems, IQ is scalable to large numbers of users
- **Sybase MPP:** A "no-contention", parallel database designed for IBM's massively parallel processor system, the IBM RS/6000 SP; Sybase MPP excels at performing standardized queries across large data volumes and centralizes VLDB data warehouse functions
- **Sybase System 11™:** A family of database servers optimized for the IBM RS/6000 family and suited to supporting mixed-mode query and OLTP applications

Middleware Tools

- **Replication Server:** Replicates data across distributed systems for high availability, high performance, and data sharing across heterogeneous databases
- **InfoPump:** Performs summaries and aggregations to the data before delivery to data marts
- **OmniConnect™:** Read and update information from many heterogeneous data sources

Client Tools

- **PowerBuilder Products:** A complete family of tools for assimilating and presenting data, and developing applications for Windows™, OS/2®, UNIX®, Macintosh®, and Internet clients

before it is placed into the target database or data warehouse.

As a business environment continues to expand, data movement requirements shift, with increasingly specific requirements for systems and applications that have been installed. Huge volumes of operational data must be converted from its raw form and moved into the data warehouse. Companies are now looking to develop enterprise-level data warehousing solutions that integrate and manage a wide variety of tools and data, such as modeling and transformation, while simplifying the scheduled distribution of business data to the various data warehouses or data marts.

The Sybase/IBM Advantage

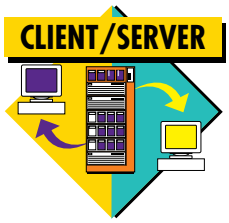
Sybase MPP and IQ, in combination with the IBM RS/6000 SP, deliver unmatched performance, unlimited scalability, high availability, and very large data warehouse management. These systems offer unique resources for prototyping, delivering, and distributing data marts while assuring that the systems will perform as users expect. This unique combination allows customers the ability to either evolve their information infrastructure top down or user up. Although top down seems like the more logical way to build a corporate architecture, user up appears to be a more pragmatic and lower-risk approach to delivering a comprehensive enterprise data warehouse.

Ron Wastal, Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608. Mr. Wastal is director of Product Management for the Enterprise Database Products Group. He has been involved in over 50 detailed data warehouse projects with some of the largest corporations in the world.

VisualAge Tools for Java

VisualAge tools for the Java programming environment, available in late 1996, will provide a complete development environment, including an editor, debugger and browser, along with a Java class library and a VisualAge Data Access builder for visually constructing data access through the use of Java Database Connection (JDBC).

VisualAge software application development tools allow developers to construct applications visually by connecting pre-fabricated, reusable software components from an expansive library of predefined classes and parts from IBM and other vendors. The visual-construction-from-parts technology will be extended to the Java programming environment, enabling developers to visually build Java applets.



SMP Performance

By Bret R. Olszewski and Jim Van Fleet

The rapidly increasing population of AIX-based Symmetric Multiprocessing (SMP) systems has sparked an interest in understanding the performance characteristics of these systems. Although the concepts important to performance on uniprocessor systems are the same, SMP performance does have some unique characteristics. This article introduces the performance problem discovery and resolution on RISC System/6000® SMPs.

Scaling is the performance increase associated with adding more processors to a system. It also represents the fundamental performance difference between uniprocessor and Symmetric Multiprocessor (SMP) systems.[1,2]

Three basic components of system scaling are hardware, system software, and application software. Hardware factors include bandwidth and latency within the system. Latency defines the maximum performance of a system, since the number of instructions executed is related to the amount of time the processor spends waiting on resources such as cache, memory, and I/O. Bandwidth is an important hardware component because it defines scalability. As processors begin to contend for resources, delays occur, which reduces overall system performance.

System software (primarily operating systems) scaling is primarily paced by serialization (locking) effects. Most software uses locks to guard access to critical data structures. For example, when you transfer money from your bank savings account to your checking account, a lock is logically taken to ensure that other transactions occurring at the same time do not impact the resulting balances on your account. Application software frequently has similar characteristics, such as locks, to system software in serialization and synchronization.

In system scaling for real applications, performance does not scale linearly with the number of processors.[3] For example, doubling the number of processors does not double the throughput of the system. In addition, not all systems or workloads scale similarly; in fact, quite the opposite is true. Particularly in hardware, scaling potential is more often related to system cost; low-cost systems are optimized for cost, not performance. Systems optimized for performance generally have a higher cost.

Tools

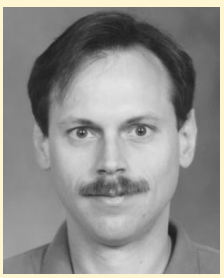
AIX provides a wide variety of performance tools for evaluating system performance. Although this article does not detail the features of performance tools, it does review the basic capabilities of some useful tools. Figure 1 describes several tools and their functions.

SMP Performance Issues

The most important categories of SMP performance problems are described below.

Workload Imbalance

Workload imbalance occurs when the workload has insufficient parallelism to keep multiple processors busy. For example, if a large FORTRAN program is run without any parallelization on an SMP, it will complete only as fast as one processor can run it; the other processors in the system will have nothing to do. Typically, any system will have some non-parallel tasks (such as some backup utilities) that may become performance bottlenecks.



Bret R. Olszewski

Jim Van Fleet

[1]Blakely-Fogel, Debora and Alexander,William. "SMP Overview." *AIXpert* (November 1994) p. 4.
[2]Alexander, William; Dimpsey, Robert; and Olszewski, Bret R. "AIX Operating System SMP Performance." *AIXpert* (November 1994) p. 35.
[3]Dixit, K.; Van Fleet, J.; Olszewski, B. "Workload Effects on SMP Scaling in AIX Version 4." Compcon '96 Proceedings. p. 117.

Another possible imbalance could be caused by funneled device drivers. *Funneling* is an AIX feature that allows device drivers not optimized for MP to run correctly. This occurs only by allowing the device driver to execute on one CPU (always CPU 0) in the system. (CPU balance can be observed with `sar`.) Although few device drivers shipped with AIX are funneled, that possibility exists for any driver, particularly third-party supplied drivers. The signature of workload imbalance is lower-than-expected CPU utilization.

Application Serialization

Application serialization manifests itself when threads must protect common resources, usually when using AIX kernel services. Normally, the characteristic of this situation is high use of system calls, particularly those associated with semaphores, file locks, message queues, or pipes. Application serialization is often noted for its high kernel CPU utilization.

Thread Serialization

Thread serialization is similar to application serialization, but it happens only when using POSIX™ threads library services. Because the threads library contains much code that runs in the user space, the characteristics of thread serialization are more subtle than application serialization. Thread serialization generally has lower-than-expected CPU utilization.

Kernel Locking Conflict

Kernel locking conflicts occur when either one or a few AIX locks are heavily used as the result of a workload. Disabled AIX locks function as spin locks; that is, when one thread attempts to access a lock held by another thread, it will repeatedly attempt to obtain the lock (spin). In other cases, the thread attempting to obtain the lock may voluntarily give up the processor, which causes a context switch to another thread.

The signature of kernel locking conflicts is high kernel CPU utilization and often high dispatch rates. High dispatch rates, measured on 8-way PowerPC 601® systems on tuned workloads, often range from 720 context switches per second to 4400 switches per second. Obviously, reasonable rates will be lower on systems with fewer processors.

MP/UP Overhead

AIX provides two kernels: a uniprocessor kernel that runs only on uniprocessor systems and an MP kernel that runs on SMP systems. This dual kernel approach allows AIX to optimize some functions for uniprocessor systems.

MP/UP overhead is the path-length difference between the two kernels. The performance difference for the two kernels can be quantified by running the MP kernel on a UP system. Normally, the MP/UP overhead of AIX is between 5% and 10%, but it depends on how much time the workload spends in the AIX kernel. The effect of MP/UP overhead is generally an issue only when comparing 2-way systems to uniprocessor systems on workloads that contain a large amount of kernel time.

Tool	Function
`-iostat`	Displays CPU and disk utilization of the system; divides processor utilization into time spent in kernel, user, idle, and disk I/O wait.
`-vmstat`	Displays CPU utilization, paging rate, and context switch rate, as well as memory pool utilization; reports processor utilization similar to `iostat`.
`-sar`	Displays CPU utilization and many other operating system counts; can display CPU load balance (collect `sar` data as `/usr/lib/sa/sadc 10 5 sardata.bin`, then process with `sar -P ALL -f sardata.bin`). Although CPU load balance is typically not a problem, it can be a useful metric.
`-lockstat`	Displays lock contention in the AIX kernel and kernel extensions; it requires that the system be booted with lock instrumentation enabled (reference `bosboot -L` command). The overhead of enabling lock instrumentation is typically 3% to 5%. Normally, only locks that have more than 1,000 references a second will have much impact on performance. Performance problems caused by AIX locking are uncommon.
`-tprof`	Samples CPU utilization to account time in threads and program modules. Although it is not designed specifically to identify SMP issues, it can help identify scaling problems in applications.
`-trace`	Collects time-stamped kernel events (see "utld: A Trace-based Performance Tool" in this issue). Trace has higher system overhead than most tools. *Hint:* when examining printable output via `trcrpt` for SMPs, use the `-O cpuid=on` flag to show the processor number for each event.

Figure 1. Tools for evaluating system performance

Hardware Configuration

Bandwidth limits can cause hardware configuration problems to occur on both UP or SMP systems. The most frequently noted configuration problem is memory configuration on SMPs that have four to eight processors.

For the current SMP models (G30, J30, R30), the number of memory banks is determined by the number and type of memory cards in the system. For example, a 64 MB memory card can contain a single memory bank. If a system with

eight processors was running with a memory card that contained a single memory bank, contention would occur for the memory bank. Therefore, adding memory cards would increase the system performance.

Since SMP and UP systems experience common performance problems, it is good practice to examine the system for obvious problems such as I/O bottlenecks, insufficient memory (paging), and networking problems before assuming that an SMP system problem exists.

Heuristics

Heuristics can help diagnose performance problems since several pathologies are possible on SMP systems. The following guidelines provide some help for choosing tools to study an SMP system that has performance problems.

First, if the CPU is busy, the processor cycles are not idle. First view the user/system mix using `iostat`, `vmstat`, or `sar`. If the problem is excessive user time, try using `tprof` to determine which threads and functions within the threads are consuming the CPU. If the problem is in system time, try trace-based analysis. Collect a trace and use `utld` on the trace. (See "utld—A Trace-based Performance Tool" in this issue.) It can also be helpful to run `lockstat` to view kernel locking behavior.

Second, if the CPU is not busy, a problem exists with I/O balance or event synchronization. First look for non-SMP specific problems, such as disk I/O bottlenecks, network problems, and excessive paging. If this is negative, the problem is most likely related to event synchronization, which requires trace-based analysis.

The rest of this article contains synthesized versions of SMP problems analyzed by the AIX performance teams. All of the measurements given were generated on a 4-way G30 SMP system running AIX 4.1.4.

Case 1: Parallelize FORTRAN Programs

A performance problem was reported during the evaluation of a preprocessor to parallelize FORTRAN programs into threads. Poor scaling was reported on the resulting parallelized program. The first step in analyzing the problem was to view the CPU consumption of the program using `timex -s (sar)` shown in Figure 2.

```
AIX longestday 1 4 00000000A600   12/05/95

12:47:57     %usr    %sys    %wio   %idle
12:48:41     36      43      0      21

12:47:57     bread/s   lread/s   lrcache   bwrit/s   lwrit/s   %wcache   pread/spwrit/s
12:48:41     0         0         0         0         0         0         0

12:47:57     slots    cycle/s  fault/s  odio/s
12:48:41     126445   0.00     5.15     0.00

12:47:57     rawch/s  canch/s  outch/s  rcvin/s  xmtin/s  mdmin/s
12:48:41     0        0        0        0        0        0

12:47:57     scall/s  sread/s  swrit/s  fork/s  exec/s  rchar/s  wchar/s
12:48:41     18734    21       0        0.05    0.07    89850    36

12:47:57     cswch/s
12:48:41     13737

12:47:57     iget/s   lookuppn/s   dirblk/s
12:48:41     4        1            2

12:47:57     runq-sz   %runocc   swpq-sz   %swpocc
12:48:41     3.4       100       1.0       100

12:47:57     proc-sz     inod-sz    file-sz   thrd-sz
12:48:41     41/131072   545/1324   203/255   48/262144

12:47:57     msg/s   sema/s
12:48:41     0.00    0.00
```

Figure 2. Case 1 sar measurements

The results are surprising. First, we have high system utilization in an application with almost no I/O. Second, we have a high context switch rate. It appears that the application has a serialization problem. Threaded applications often serialize with mutex locks. Mutex locks are supported in AIX's `pthread` library and via specialized kernel services.

Further analysis with `tprof` shows that the application has significant time in `libpthreads`, particularly in the functions `_spin_lock`, `pthreads_mutex_lock`, `pthread_mutex_unlock`, and `_spin_unlock`. This confirms that the application is doing serialization via the `pthreads` library.

After viewing an AIX trace using `trcrpt`, (Figure 3) we see threads going to sleep via `thread_waitlock`. In the sequence, the thread running on CPU 3 calls `thread_waitlock` because of lock contention detected in the `pthreads` library. This causes the processor to make the thread sleep (via `e_assert_wait` and `e_block_thread`) and dispatch a new thread (the idle thread).

`Utld` shows the CPU consumption caused by `thread_waitlock` and `thread_unlock`. Figure 4 shows that `thread_waitlock` and `thread_unlock` dominate system call CPU time. It points to 28.967% plus 19.285% as the largest slice of time.

Kernel (System Call) Summary

The program did not scale because the locking was not granular enough. As threads attempted to get the mutex lock, which managed shared data among the threads in the program, they ended up spinning for the lock because it was heavily used. The AIX mutex lock implementation limits the duration of the lock spin. When the limit is hit, the thread is made to sleep. Figure 5 shows how threads contend for the mutex lock over time.

The parallelizing application did not make any efforts to determine the appropriate amount of parallelism, but it required the user to direct it to parallelize the important loops. If the user encouraged the application to parallelize a loop that was not inherently parallel, the locking overhead to manage the shared data in the resultant program was greater than the benefit of the parallelism.

```
ID CPU     ELAPSED_SEC       DELTA_MSEC      APPL            SYSCALL KERNEL   INTERRUPT
101 3      0.000679296       0.002688        thread_waitlock LR = D027BF34
112 3      0.000694528       0.011264        lock:           miss lock
 addr=20000BD8 lock status=30000000 requested_mode=LOCK_SWRITE return addr=36F4
name=0000.0000
46D 3      0.000699904       0.005376        wait_on_lock:   pid=5874 tid=798
1 lockaddr=20000BD8
460 3      0.000705664       0.005760              e_assert_wait:   tid=798
1 anchor=2FF3BCFC flag=1 lr=19084
462 3      0.000738816       0.027776              e_block_thread:   tid=79
81 anchor=2FF3BCFC t_flags=0020 lr=19094
10C 3      0.000848512       0.009472                  dispatch:   idle process
  pid=1290 tid=1291 priority=127 old_tid=7981          old_priority=103 CPUID=3
```

Figure 3. Case 1 trcrpt output

Processing Total (msecs)	Percent Processing Time	Count	Path in msecs min	avg	max	System Call
51873.955	28.967	000004291	0.016	0.437	1.469	thread_waitlock
1247.599	19.285	000003665	0.019	0.340	1.126	thread_unlock
3.340	0.052	000000009	0.046	0.371	1.090	kioctl
1.809	0.028	000000001	1.809	1.809	1.809	_exit

Figure 4. Case 1 utld system call output

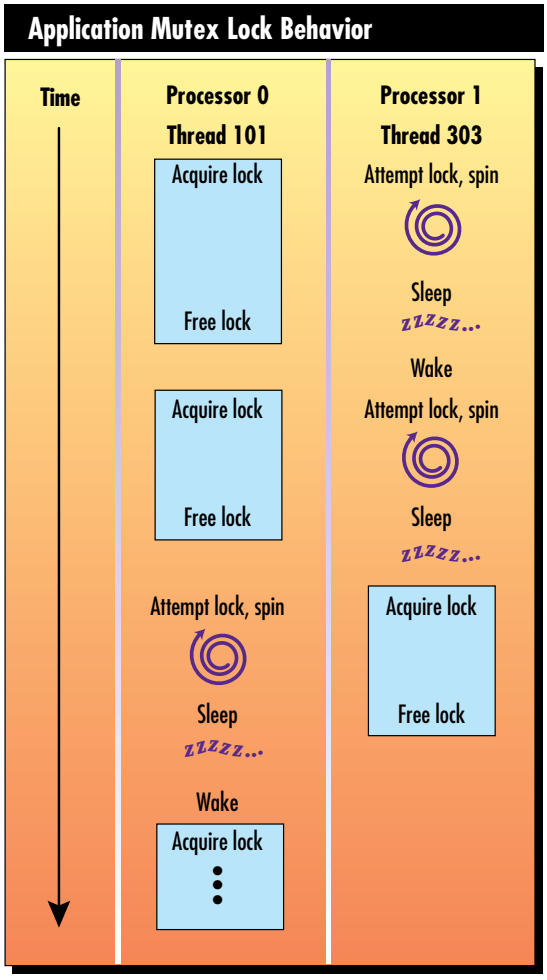


Figure 5. Application mutex lock behavior

Case 2: Pseudo-Database Application

A pseudo-database application did not scale well on a 4-way SMP system. The application had many processes accessing a common database—with high I/O rates. Adding more users did not increase throughput. The first view of the workload using `iostat` (Figure 6) shows excessive time in the AIX kernel. Further viewing via `vmstat` indicates no paging. The I/O rates are not too high, although it appears that application caching of reads is effective because all disk I/Os are writes. Since the CPU is busy, it is possibly a locking problem.

An AIX trace provides a more detailed view of the system. Using `trcrpt` to view the trace allows interesting events for threads and processors to be tracked. Figure 7 shows that even without lock instrumentation turned on, evidence of lock conflict may be noted in the trace. Notice the event `wakeup_lock`, a kernel function used to wake threads waiting on a lock. When the read system call wakes up a thread to run (observable by the `setrq` trace hook), 89 threads are waiting on the lock! At this stage, it would be wise to enable lock instrumentation and use `lockstat`.

Figure 8 shows `lockstat` output on the running system. The `PROC_INT_CLASS` and `IRDWR_LOCK_CLASS` locks stand out. The `PFS IRDWR_LOCK_CLASS` lock indicates a filesystem

```
tty     tin     tout     avg-cpu:  % user  % sys  % idle  % iowait
        0.0     1.6                26.2    47.6   2.9     23.3

Disks:     % tm_act   Kbits/sec  tps       Kb_read   Kb_wrtn
hdisk0     20.1       73.4       17.7      0         369
hdisk1     14.5       74.0       18.5      0         372
```

Figure 6. Case 2 iostat output

```
ID   CPU    ELAPSED_SEC      DELTA_MSEC    APPL  SYSCALL KERNEL   INTERRUPT
101  2      0.007946240      0.247296      lseek LR=100005A0
104  2      0.007965184      0.018944      return from lseek [18 usec]
101  2      0.007975680      0.010496      kreadv LR=D00437C8

163  2      0.007989888      0.014208      read(3,2FF21CB,1000)
10A  2      0.007999232      0.009344      PFS rdwr (vp,ip)=(9AB9880,9AB98B0)
10A  2      0.008531200      0.011776      PFS rdwri VA.S=1D9000.BA17 bcount=1000 ip=9AB98B0
10A  2      0.008689408      0.045312      setrq: cmd=inode pid=37782 tid=38821 priority=65 policy=0
46E  2      0.008711936      0.022528      wakeup_lock: lockaddr=9AB9930 waiters=89
104  2      0.008734208      0.008320      return from kreadv [758 usec]
```

Figure 7. Case 2 trcrpt output

Subsys	Name	Ocn	Ref/s	%Ref	%Block	%Sleep
PROC	PROC_INT_CLASS	1	5128	12.37	10.29	0.00
PFS	IRDWR_LOCK_CLASS	18943	4041	9.75	80.28	9.00

Figure 8. Case 2 lockstat output

problem with file inode locks. UNIX file semantics require a lock to be held over I/Os. If many processes are reading or writing to a single file, contention to accessing the file may result. This is not strictly an SMP problem; it could also occur on a uniprocessor.

The use of `PROC_INT_CLASS` lock is pushed higher in the system because lock contention is occurring on the `IRDWR_LOCK_CLASS`. The `PROC_INT_LOCK` lock is used for dispatches on AIX 4.1.4, so if lock contention causes extra dispatching, extra conflict occurs on the dispatching lock. The solution is to use raw logical volumes instead of files. Otherwise, try to split the database into multiple files to reduce inode lock contention on any single file.

Figure 9 shows how contention for the inode lock slows application performance.

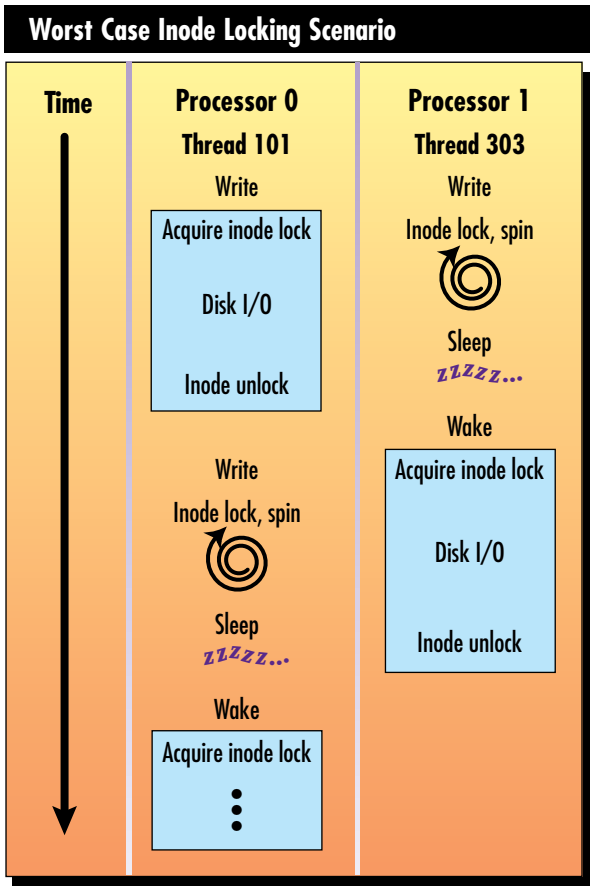


Figure 9. Worst case inode locking scenario

Case 3: Pseudo-database Application

This pseudo-database case also did not scale well to SMP. Figure 10 shows `iostat` measurements on the application. With the high kernel CPU consumption, it appeared that AIX had a problem. `Sar` (not shown) shows a high context switch rate of 13767 context switches per second.

All workloads that show high kernel time provide an ideal opportunity to try `utld` because

```
tty:    tin    tout    avg-cpu:  % user    % sys    % idle    % iowait
        0.0    0.0                19.8      60.5     19.8      0.0

Disks:          % tm_act     Kbits/sec     tps     Kb_read    Kb_wrtn
hdisk0          0.0          0.0           0.0     0          0
hdisk1          0.0          0.0           0.0     0          0
```

Figure 10. Case 3 iostat output

Processing Total (msecs)	Percent Processing Time	Count	Path in msecs min	avg	max	System Call
5424.494	63.181	000004760	0.014	1.140	13.730	semop
7.193	0.084	000000010	0.198	0.719	0.904	kwritev

Figure 11. Case 3 utld system call output

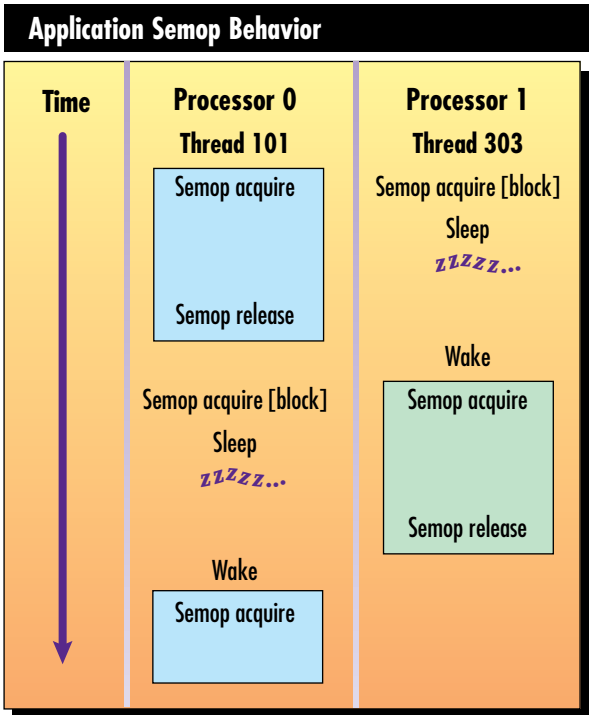


Figure 12. Application semop behavior

the kernel CPU utilization should show up in system calls or interrupts. Figure 11 shows the `utld` output for system calls. The `semop` system call is using 63% of all processing time.

Kernel (System Call) Summary

We discovered during discussions with the application developers that a single semaphore was being used to serialize all access to a shared memory segment. Collisions on the semaphore were limiting throughput. As in Case 1, the granularity of locking via the semaphore needed to be improved in order to increase the scalability of the application. Figure 12 shows how semaphore contention slows application performance.

Case 4: Server/Client Processes

The final case was an application composed of a server process and client processes. The client processes communicate with the server through shared memory. When a client process has work for the server, it sends a message. The client then waits for the server on a unique (to that client) semaphore. Figure 13 shows `sar` data for

```
AIX longestday 1 4 00000000A600    12/05/95

15:34:14   %usr    %sys    %wio    %idle
15:34:15   22      6       0       72

15:34:14   bread/s   lread/s   %rcache   bwrit/s   lwrit/s   %wcache   pread/s   pwrit/s
15:34:15   0         0         0         0         0         0         0         0

15:34:14   slots    cycle/s    fault/s    odio/s
15:34:15   126410   0.00       181.78     0.93

15:34:14   rawch/s   canch/s   outch/s   rcvin/s   xmtin/s   mdmin/s
15:34:15   0         0         0         0         0         0

15:34:14   scall/s   sread/s   swrit/s   fork/s   exec/s   rchar/s   wchar/s
15:34:15   15809     116       41        6.85     0.31     418546    828

15:34:14   cswch/s
15:34:15   3019

15:34:14   iget/s   lookuppn/s   dirblk/s
15:34:15   27       11           14

15:34:14   runq-sz   %runocc   swpq-sz   %swpocc
15:34:15   4.7       100       1.0       100

15:34:14   proc-sz      inod-sz      file-sz      thrd-sz
15:34:15   41/131072    621/1520     204/340      48/262144

15:34:14   msg/s      sema/s
15:34:15   5102.93    5116.01
```

Figure 13. Case 4 sar output

Processing Total (msecs)			Percent of Total Processing Time			Process Name
Combined	Application	Kernel	Combined	Application	Kernel	(process id/thread id)
2251.938	1630.816	621.122	23.808	17.242	6.567	lazyp (5874 5683)
44.737	0.086	44.650	0.473	0.001	0.472	syncd (3044 2003)
18.345	2.350	15.995	0.194	0.025	0.169	lazyp (5876 5685)
17.973	2.472	15.501	0.190	0.026	0.164	lazyp (17610 7435)

Figure 14. Case 4 utld output

the workload. Note that a significant amount of idle time occurs in the workload.

At this stage, the CPU system utilization seems limited to some integral fraction of the number of processes in the system. This could imply a single-server bottleneck characteristic of workload imbalance. The system has four processors with 28% CPU utilization. Figure 14 shows that the trace and `utld` indicate one process, `lazyp`, is consuming 23.8% of the system's CPU.

The server process does most of the work in this workload. Since it can only run on one processor at a time, the other processors are underutilized. To achieve higher throughput on an SMP system, the server process must be split into independent processes or threads.

Figure 15 shows how a single server bottleneck limits performance for an application.

Conclusion

Three success factors are important to effectively study the performance of an SMP system. First, understand the capabilities of the performance tools. This is important because a vast array of choices are available, but each presents a different view of the system. Second, use heuristics to select the tools wisely. Most SMP problems are not subtle, so with good insight you should be able to understand them quickly. Third, network with others. It is very likely that someone else has seen the identical or similar problem in the past.

Bret R. Olszewski, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Olszewski is a senior programmer working on MP performance. He joined IBM in 1989 and has worked on various aspects of AIX performance. He has a BS in Computer Science from the University of Minnesota.

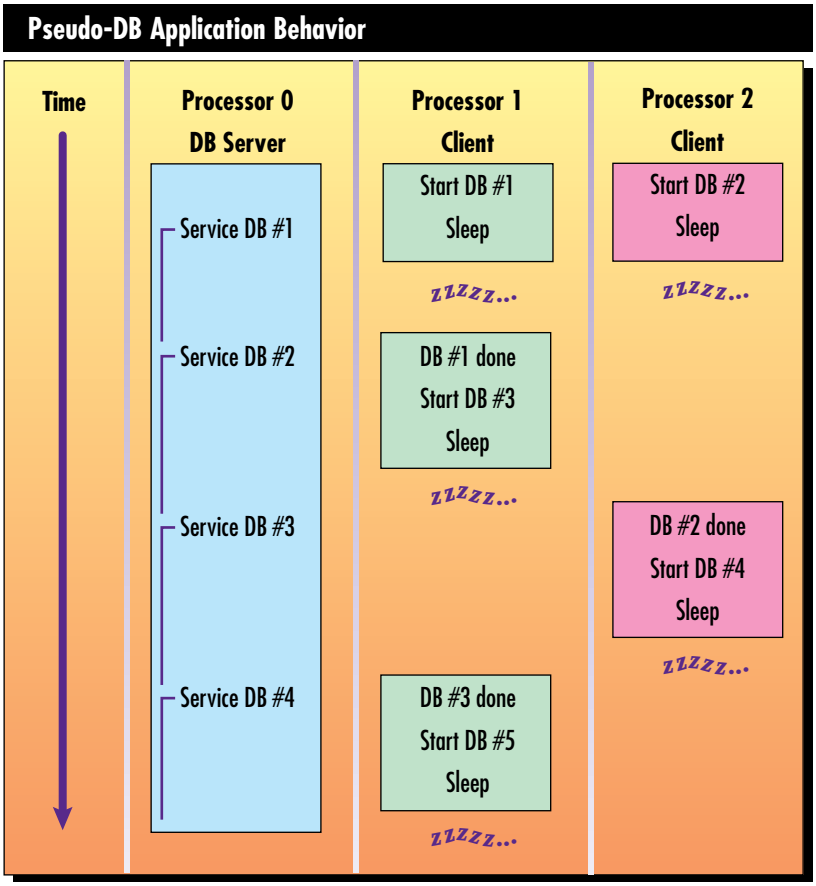


Figure 15. Single-server bottleneck limits application performance

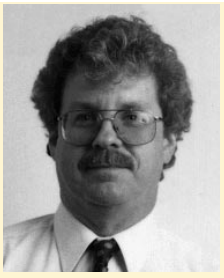
Jim Van Fleet, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Van Fleet is a senior programmer in AIX Performance in Austin. He has held several technical and managerial positions during his career at IBM including broad experience in operating systems with a specialty in Symmetric Multiprocessing systems. Mr. Van Fleet has a BS in Mathematics from Michigan State University and an MS in Computer Science from Union College.

Using Geographic Clustering For Disaster Recovery

By Thomas Casey and Herb Linnell

The GeoHA software, developed by CLAM Associates, provides a disaster-tolerant computing environment for critical data and applications. GeoHA works with IBM's HACMP software to provide automated disaster recovery. In this article we examine how the GeoHA software provides automated disaster recovery. We review the core high availability services provided by the HACMP software, then describe how the GeoHA software extends this high availability to encompass two sites. We conclude the article by discussing the benefits provided by the GeoHA software.

Thomas Casey



Herb Linnell

The '90s have been very kind to Michael Jordan and the Chicago Bulls. They have not been as kind to companies who failed to safeguard mission-critical computing resources from disasters. In this decade, a seemingly unending series of disasters—both natural and man-made—have wreaked billions of dollars worth of destruction. Figure 1 lists some of the major disasters that have struck the United States alone since 1990. Add the worldwide disasters during this time, and the total losses are staggering.

And a disaster does not have to be as catastrophic as an earthquake or hurricane. It can be as mundane as a water main break that washes out a data center. Various types of disasters can threaten the availability of critical computing resources, such as the following:

- Natural disasters—floods, earthquakes, hurricanes, tornadoes, blizzards, wildfires
- Infrastructure disasters—power failures, fires, water main breaks
- Operational disasters—hardware faults, viruses, human error

The bottom line is that anything that causes downtime hurts a business, both in terms of lost revenue and lost opportunity. Clearly, prudent companies must implement a disaster recovery solution that ensures critical data and applications remain continuously available, no matter what. For AIX users, the Geographic High Availability (GeoHA) software is that solution.

The GeoHA software, developed by CLAM Associates, provides a disaster-tolerant computing environment for critical data and applications. GeoHA works with IBM's High Availability Cluster Multiprocessing (HACMP) software to provide automated disaster recovery. HACMP uses loosely coupled clustering technology to

Major U.S. Disasters of the 1990s

Disaster	Year	Estimated Losses
Hurricane Andrew	'92	$24 billion
Los Angeles earthquake	'94	$20 billion
Midwest floods	'92	$12 billion
Southeast drought	'94	$3 billion
Hurricane Opal	'95	$2 billion
Los Angeles riots	'92	$2 billion
California wildfires	'91	$1.7 billion
Hurricane Iniki	'92	$1.6 billion
California floods	'95	$.7 billion

Source: Contingency Planning Research

Figure 1. Major U.S. disasters

prevent individual components, including processors, networks, and disks, from being single points of failure within a cluster.

GeoHA extends HACMP to encompass two physically separate data centers, or sites. GeoHA, by using two sites, prevents an individual site from being a single point of failure within the cluster. Each site maintains an updated copy of essential data and can run key applications, ensuring that mission-critical computing resources remain continuously available at a geographically separate location should a failure or disaster disable an entire site. HACMP prevents a failure in the computer room from disabling a business. GeoHA prevents a site failure from disabling a business.

Figure 2 shows the increasing levels of protection provided by a stand-alone system, an HACMP cluster, an HACMP cluster with RAID, and a GeoHA geographic cluster.

In this article we examine how the GeoHA software provides automated disaster recovery. We begin by reviewing the core high availability services provided by the HACMP software. Next, we describe how the GeoHA software extends this high availability to encompass two sites. We conclude the article by discussing the benefits provided by the GeoHA software.

HACMP

A highly available HACMP cluster is a group of independent processors networked together, sharing critical resources, that cooperate to provide application services to clients. A cluster manager agent, which runs on each processor, is the central control mechanism for providing high availability.

The cluster manager monitors local hardware and software subsystems, and tracks the availability of other processors in the cluster. When a monitored component fails, the cluster manager detects the loss and shifts that component's workload to a designated alternate in the cluster. The HACMP software prevents individual components—including processors, networks, and disks—from being "single points of failure" within a cluster. For example, the cluster manager monitors network adapters by sending keep-alive packets every half second over the service adapters. The cluster manager uses the presence or absence of keep-alive activity as a sign of the adapter's health. If there is no keep-alive activity for five seconds, the cluster manager instructs a standby adapter to take over the IP address of the service adapter. Once detected, swapping adapters take about three seconds.

Since becoming generally available in 1992, HACMP has been installed in thousands of sites. A mature product, HACMP provides extensive processor, disk subsystem, and network support.

GeoHA

The GeoHA software integrates into HACMP's well-established high availability framework. GeoHA uses the standard HACMP protocol to provide high availability services across a geography. For example, GeoHA uses HACMP daemons and scripts to monitor cluster components and to drive the fallover and recovery process. The benefit to users is familiarity. Since GeoHA builds upon HACMP, the learning curve is significantly reduced. Figure 3 shows an example of a GeoHA geographic cluster.

Now let's take a look at this geographic cluster to see how it differs from a standard HACMP cluster. GeoHA adds three components to provide disaster tolerance. These added components are as follows:

- A second site
- Geographic networks connecting the sites
- Geographic mirrors that shadow data entered at one site to the second site

Levels of Availability

System Type	Level of Resilience	Configuration Increment
Stand-alone	Vulnerable to operator error, software defects, hardware failure, single-building mishaps, metropolitan-area mishaps	
HACMP cluster	Survives operator error, most administrative assault, processor failure	Cluster hardware and software
HACMP cluster with RAID	Survives operator error, most administrative assault, protected hardware failure with RAID	Cluster hardware and software, RAID controllers, 25% disk increment
GeoHA geographic cluster with mirrored disks	Survives operator error, most administrative assault, protected hardware failure, plus single-building and metropolitan-area mishaps	Cluster hardware and software, GeoHA software, high-speed point-to-point geographic link, 100% disk increment

Source: Gartner Group

Figure 2. Levels of availability

A GeoHA Geographic Cluster

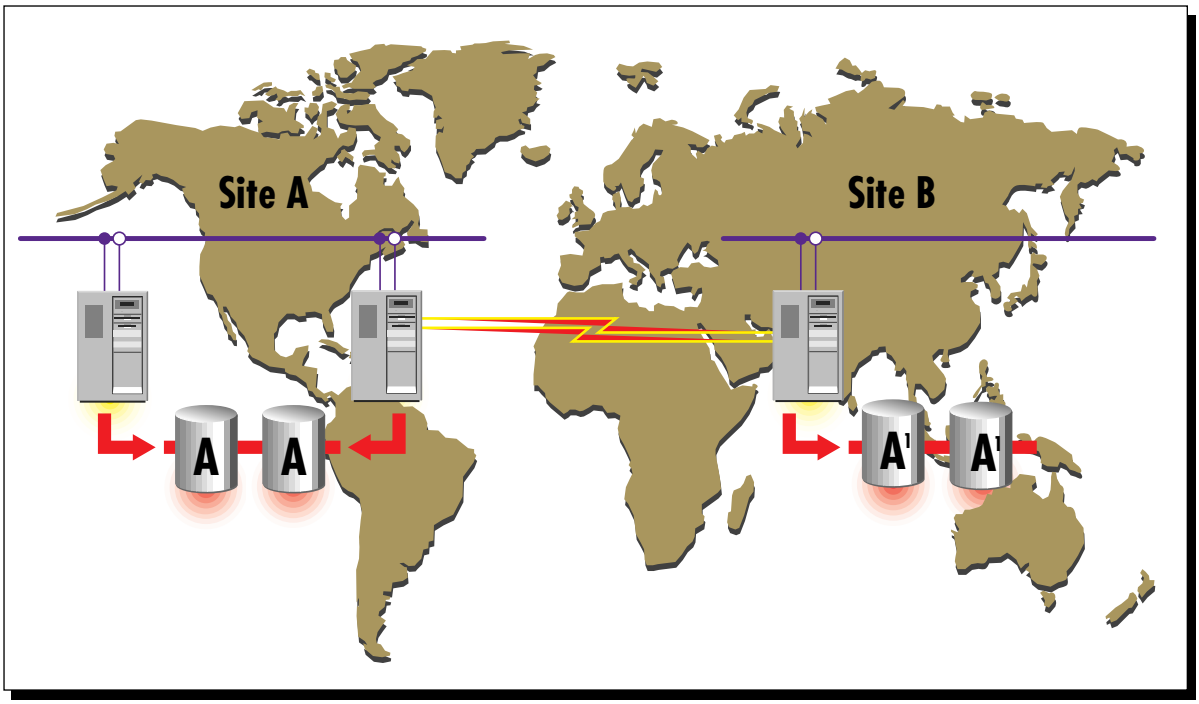


Figure 3. A GeoHA geographic cluster

A *site* is a data center that runs the GeoHA software. A GeoHA geographic cluster includes two sites. The sites should be separated by enough distance to prevent the same disaster from disabling both sites. An individual site can have from one to seven nodes; the geographic cluster can have a total of eight nodes. It is important to note that, together, the two sites form a single HACMP cluster.

The sites in this example are connected by multiple geographic networks. A *geographic network* is a TCP/IP network dedicated to mirroring data between the two sites within the GeoHA cluster. It is also used for HACMP heartbeat traffic. Clients do not have access to the geographic networks. Ideally, a GeoHA cluster should have a minimum of two geographic networks that follow diverse routes to connect the sites. This prevents the same disaster from disabling all the network connections between sites.

Each site has geographic mirrors. Mirroring the data across sites ensures that, even if a site fails, an up-to-date copy of the data is available on the other side. During fallover, this copy can be brought online so that operations can continue. The GeoHA software supports all HACMP cluster configurations:

- Hot standby configuration where a remote site backs up a local site
- Mutual takeover configuration where both sites are active and each can take over for the other if it should fail
- Concurrent access configuration where a single disk image is shared by the nodes across the geography (supported only in synchronous mode)

Providing Disaster Tolerance

Now, having seen what a GeoHA cluster looks like, let's take a look at how the GeoHA software can continue to provide mission-critical data and application services even if an entire site is disabled. To provide this protection, GeoHA must do the following:

- Mirror data at both sites
- Monitor both sites to detect failures

- When a site fails, track data written to the local site so that it can be propagated to the remote site during reintegration
- Reintegrate the failed site with an accurate copy of the data

Mirroring Data Between Sites

Essential to providing disaster tolerance is having up-to-date copies of critical data at both sites. GeoHA provides a geographic mirroring device that distributes disk I/O transactions to both sites in the geographic cluster, and a geographic messaging system that ensures the data is delivered. Data written at a local site is transmitted across a geographic network to the remote site, so that both sites have an updated copy of the data. Then, if one site becomes inaccessible, the data can be accessed at the remaining site, allowing operations to continue. The key point to understand is that the geographic mirroring system does not simply copy data from point A to point B. The geographic mirroring system must also:

- Maintain the integrity of the data
- Track data changes made to the primary site that have not yet been duplicated to the remote site
- Deliver the data quickly and efficiently across the geographic network

The geographic mirroring device is a pseudo-device driver layered above the AIX Logical Volume Manager (LVM) or a disk device. The geographic mirroring device has two components: a local geographic mirror device that sends data across the geography to the remote site, and a remote geographic mirror device that receives the data. In this schema, the local site initiates a transaction while the remote site receives disk block transmissions from the local site. The roles of local and remote change, depending on the site that initiates the transaction.

Synchronous and Asynchronous Geographic Mirroring

Data may be mirrored synchronously or asynchronously. With synchronous mirroring, the data is actually written first at the remote site, then at the local site. No more input is handled until both writes are complete. Synchronous mirroring provides the highest data integrity between sites; the local and remote sites have exactly the same data at all times. Note, however, that application performance at the local site may be slowed somewhat by synchronous mirroring.

With asynchronous mirroring, the data is first written to the local site, and then sent to the remote site. Input can continue at the local site while the previously entered data is mirrored to the remote site. Asynchronous mirroring allows the remote site to lag behind the writes to the local disk. While this may improve performance, it is possible to lose the data queued at the remote site if a disaster strikes.

Ensuring Data Delivery

Geographic messaging is a kernel Remote Procedure Call (RPC) protocol that guarantees the delivery of geographic mirroring data should a network fail, provided that at least one alternate, functioning network exists in the GeoHA environment. Geographic messaging automatically reroutes requests in the event that a request fails to reach a destination node.

Monitoring Both Sites

Earlier, we said that geographic clustering was an extension of highly available clustering. A geographic cluster has additional software logic that allows the cluster manager to extend its coverage to encompass a remote site. In highly available clustering, the cluster manager on a processor exchanges "heartbeats" with the cluster managers on the other nodes in the cluster so that they can detect a change in the health of a particular processor. In geographic clustering, the cluster managers at a site exchange heartbeats not only with the processors at the local site, but also with the cluster managers at the remote site. In this way they can determine when there has been a change in the status of the peer site.

Responding to Failures

The purpose of the GeoHA software is to keep key data and application services available even if a disaster destroys one of the sites in the cluster. Now let's take a look at what happens when something goes wrong. First we will look at what happens when a specific component fails, then we will look at what happens when a disaster disables an entire site.

Essential to providing disaster tolerance is having up-to-date copies of critical data at both sites.

Handling Failures within a Site

A *local failure* is the failure of a specific system component within a specific site within the geographic cluster. The component could be a processor, a disk or disk adapter, a local area network, or local area network adapter. The basic facilities of highly available clustering are used to handle local failures.

Each system component (including networks, adapters, and disks) that is a potential "single point of failure" has an automatic replacement designated for it. The cluster manager monitors the status of each designated component, detects the failure of one of these components, and redistributes that component's workload to a backup component to maintain the availability of cluster resources. In this way, component failures are handled within site boundaries, and are transparent to the other site in the geographic cluster.

Geographic clustering software is able to respond intelligently to changes in the status of the cluster so that it can maintain high availability.

Handling Site Failures

Unlike the failure of a system component, which was contained within a specific site, a disaster requires that the viable site take over for the site that has been disabled. The viable site must be able to do the following:

- Detect the failure
- Track data written to the local site so that it can be propagated to the remote site during reintegration
- Continue to provide mission-critical data and application services
- Reintegrate the failed site when it comes back online

These capabilities are what distinguishes geographic clustering from remote mirroring or data replication. Geographic clustering software is able to respond intelligently to changes in the status of the cluster so that it can maintain high availability. No operator intervention is required to restore services.

Earlier we said that the cluster managers exchange heartbeats across the geography. When the cluster managers at a local site cannot communicate over any of the geographic networks connecting the two sites, they assume the remote site may have failed. The cluster managers at the local site must then determine if the failure is a global network failure or a site failure, and take the appropriate action to preserve data integrity between the sites.

Site Isolation

Site isolation, or global network failure, occurs if all the networks used to transmit geographic mirroring data are disabled. Despite global network failure, the GeoHA cluster may still be sending keep-alives over a serial network to determine that the remote site is functioning. When site isolation occurs, the site configured as the dominant site continues functioning; the other site will bring itself down gracefully to preserve data integrity between the sites. When the networks are functioning again, the non-dominant site may rejoin the cluster; the GeoHA software then synchronizes the data.

Site Failure

The failure of all nodes at a site is a site failure. The HACMP software on the viable site initiates the takeover of the resources of the failed site. In a concurrent access configuration, no transfer of ownership is necessary. In a cascading configuration, any resources defined for takeover will be taken over by the nodes on the viable site. The nodes at the takeover site mark the last geographic mirroring transaction completed, and then keep track of all data entered after communication with the other site stopped.

Suppose, for example, you have two sites, Toronto and Vancouver, with two nodes at each site. Toronto has nodes alpha and beta; Vancouver has nodes gamma and delta. The resources owned by node alpha at Toronto should be configured to cascade to node beta for a local failure and to node gamma (Vancouver) if node beta is not available. Similarly, the resources owned by node beta will cascade to node alpha or to node delta if node alpha is not available.

Reintegrating a Failed Site

When a site reintegrates, the updates that occurred during the failure need to be resynchronized between the two sites. This process is handled by the geographic mirroring component. When the first remote node sends the message that it is ready to rejoin the cluster, the geographic mirroring device on the functioning site delays the regular HACMP reintegration process until resynchronization completes.

The nodes at the viable site continue to process data while HACMP is bringing up-to-date the node that is rejoining. When the remote node successfully rejoins the cluster, all of its configured HACMP applications are then available. Once the first node is up, the site can continue operations. The time for synchronization depends on several factors:

- Number of geographic mirrors involved in the process
- Size of the geographic mirroring devices
- Amount of data that has been updated since the last outage
- Network bandwidth available
- Network traffic

Benefits of the GeoHA Software

The GeoHA software provides the following benefits.

Reliable data integrity and consistency. GeoHA's geographic mirroring and geographic messaging components ensure that if a site fails, the failed site's data is consistent with the surviving site's data. When the failed site reintegrates into the cluster, GeoHA updates that site with the current data from the operable site, once again ensuring data consistency.

Automatic failure detection. The GeoHA software automatically detects a site failure and initiates the recovery process, including notification that a site has failed.

Automatic or manual fallover. When a site fails, you can manually execute a process to transfer control of data and applications to the operable site. Or you can use HACMP's recovery scripts to automate this process. Using HACMP results in less costly downtime while eliminating the possibility of inducing system failure during recovery.

Fast recovery. The GeoHA software provides fast recovery of mission-critical data and applications at the operable site. The GeoHA software can reduce the time needed to restore computing resources to minutes, although individual times can vary depending on the amount of resources that need to be shifted to the surviving site and the amount of application recovery processing required.

Flexible configurations. The GeoHA software supports a wide range of configurations, allowing you to configure the disaster recovery solution unique to your needs. Configurations can range from an online backup machine turned on nightly to receive an updated copy of a database to a concurrent access configuration, where both sides have simultaneous access to the same database.

Geographically separate sites. The GeoHA software imposes no constraints on the distance between the sites. This allows you to separate the sites by enough distance to ensure that the same disaster does not render both sites inoperable.

Flexible, scalable architecture. You can scale a GeoHA geographic cluster simply by adding memory and I/O controllers to individual processors, by swapping in more powerful processors, or by adding more processors. A GeoHA cluster can support up to eight nodes.

Based on industry-leading technology. The GeoHA platform is built from proven components—cost-effective, high-performance RISC System/6000 servers, and the industrial-strength AIX operating system.

Filesystem and database independent. The geographic mirroring device behaves the same as the disk devices it supports. Because the mirroring is transparent, applications configured to use geographic mirroring do not have to be modified in any way. In this sense GeoHA is a "generic" solution that works with any filesystem or database management system.

GeoHA is a "generic" solution that works with any filesystem or database management system.

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.

Herb Linnell, CLAM Associates, Inc. 101 Main Street, Cambridge MA 02142. Internet: herb@clam.com. Mr. Linnell is a senior member of CLAM's product development staff. He has a BSCE from the University of New Hampshire in Durham and a MSCE from Northeastern University in Boston.

Java on AIX—A Strong Brew

By Jeff Jilg

Java is a rapidly evolving technology. Its platform independence and Internet interoperability features help to enhance its popularity. Java on AIX allows programmers to exploit this technology and users to access newly created Java applets. Security is a concern to many, and this has been taken into account through integration of security features as part of the base infrastructure in the technology. Although Java allows flexibility, it requires both programmers and administrators to be aware of the potential problems that can result from careless implementation.

The Java™ technology has created much excitement in the past year, but it has also resulted in some confusion. What is Java? Can it be used to deliver only Internet or Web page-based programs? Is the environment similar on different platforms? Where should a programmer begin? What do system administrators need to know? How about security? The objective of this article is to answer these and other Java-related questions and to provide references for further exploration.

The quickly evolving nature of any new technology often creates some confusion, and Java is no exception. Java was introduced to a marketplace that was looking for options and adaptability. At the same time, companies still expect some standardization and interoperability to reduce platform dependencies and provide easy migration paths. Let's examine some Java technology components and the benefits they deliver.

A Brief History of Java

The Java technology, initially developed in 1991 at Sun Microsystems® under the Green group, was aimed at the television set top box market of the commercial electronics industry. After a couple of name changes, Java and HotJava™ were announced at SunWorld '95 in San Francisco on May 23, 1995.

Interest in the technology grew rapidly through the remainder of 1995. IBM announced licensing of the technology in December 1995 and Microsoft® also announced their intent to license Java. IBM has formed a Centre for Java Technology in Hursley, England where they are actively porting the technology to various platforms including AIX.

In May 1996, Sun®, IBM, and others sponsored the JavaOne™ conference, which attracted more than 5,500 attendees. With the enormous amount of activity focused on Java, it is definitely still in the growth phase. As with any new technology, initial delivery is followed by availability on various platforms, enhanced development tools, then new applications that occur throughout the various phases as the technology matures. Many believe that major applications will begin appearing in numbers during late 1996 and early 1997.

The Java Technology

The base Java technology has two primary components: Java Virtual Machine (JVM) and the Java language. The Java Virtual Machine (JVM) is an abstract CPU specification that provides a runtime environment for all Java programs. The specification looks much like a traditional CPU with a defined bytecode instruction set that can interact with the abstract CPU. Since abstractions cannot be used to run physical programs, this abstract specification must be converted to an executable environment on a real system.

Sun has a reference port of the JVM (and other related components) available. Various companies, including IBM, have licensed this reference code and ported it to their respective platforms. Thus, IBM's AIX port of the Java technology includes the JVM. Most programmers and site administrators will not need to learn the details of the JVM because higher-level interfaces provide easier access to the technology.

Most programmers use the Java language, the primary interface to the JVM, to take advantage of Java technology. Because the language is a C++ derivative, it can be used to write a variety of different programs. The Java language is also object-oriented, which provides several advantages including reusability and encapsulation of data and program behavior into objects.

Java Versus C++

Java and C++ differ in several areas. The Java language lacks pointers—the first big difference most programmers will notice. These pointers were specifically removed to reduce the security exposures because of their ability to trash memory. Another problem with C++ pointers is that if they are misused, the result is often a disproportionate amount of bugs. The Java language does pass all arrays and objects by reference in place of pointers.

Another difference is that C++ supports multiple inheritance; Java does not. Multiple inheritance allows programmers to create object classes that inherit data and/or behaviors from more than one parent class. This feature is implemented through Java interfaces that provide method descriptions without implementations.

Platform Independence and Other Benefits

Java technology is object-oriented, but it also has other benefits that are helping to drive its popularity into the stratosphere. The primary benefit is platform independence. Programs can be developed on one platform, then executed on any platform that supports the JVM and associated runtime components (as shown in Figure 1). This powerful concept allows a company to develop a program in one language (Java) and avoid the expense of porting it to various target platforms.

Platform independence also affects support issues. Because Java programs have a single code base, support personnel can now focus their expertise on the problem, not platform-specific issues and the various code bases associated with different platforms. This benefit also extends to end users, who can now receive a single compiled Java program that can run on any machine in their heterogeneous environment.

Dynamic executable content allows users to easily download and execute a Java applet. It is enabled through incorporation of the JVM and Java runtime engine into numerous Web browser environments, such as appletviewer (discussed below) and Netscape Navigator® (Version 3.0 on AIX). End users can receive programs over the Internet through a very simple process. A user can click on a hypertext link on a Web page and automatically download a Java applet (program), as shown in Figure 2.

The HTML associated with the Web page must contain the <APPLET> tag with a reference to the Web server holding the applet and the applet name. When the user clicks on the link, a request is sent to the Web server to send the applet to the browser. After the Web server has transmitted the applet (and any images) to the browser, the browser then executes the Java applet.

Versioning control is another benefit of this scenario. Suppose a user already has a Java applet that is used daily. Each morning the applet is started. It could communicate with the host at the sponsor company, obtain any new updates or bug fixes, and automatically install them on the client. Some updates could possibly advertise new features or additional components

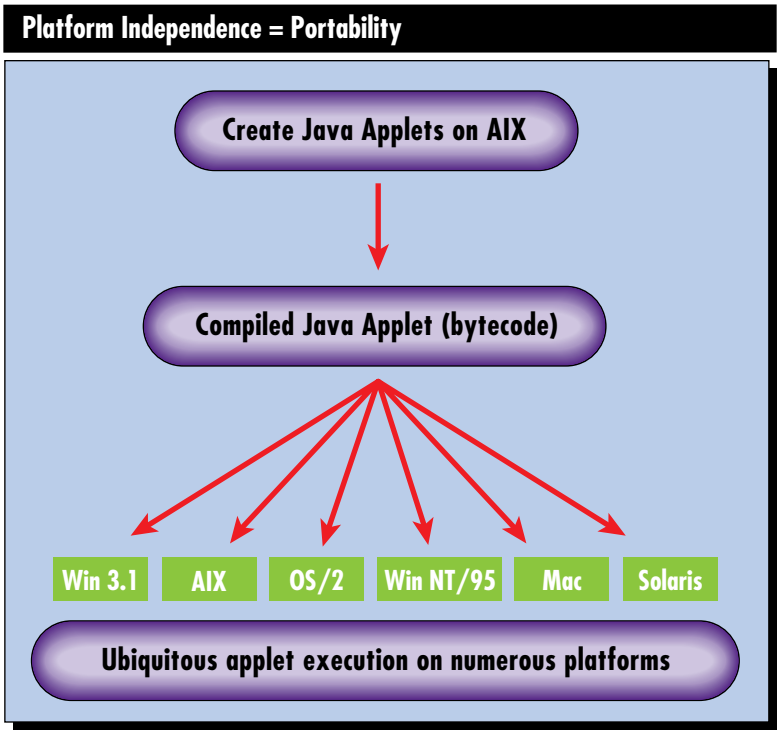


Figure 1. Platform independence = portability

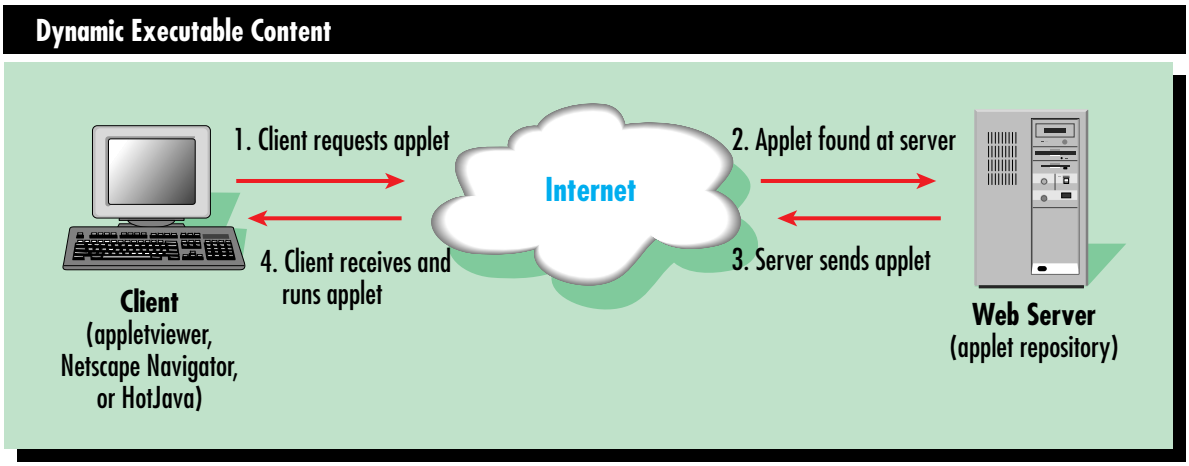


Figure 2. Dynamic executable content

that end users could then choose to purchase over the Internet.

Although Java does not directly deliver this scenario in the native runtime environment, some creative programming can easily extend the features of the environment. The possibilities are almost endless.

The support staff can also benefit from this type of functionality. Online installation and updating can virtually eliminate the costly physical mailings that support has relied on to deliver program bug fixes and updates.

Development Components	Description
compiler, debug runtime, debugger	Compiles Java source to bytecode and debugs it
javadoc tool	Creates HTML files from comments in Java source
header and class files	Used by compiler to create compiled Java classes
class file disassembler	Extracts information from compiled Java class file
javah tool	Generates C headers and stubs for C programmers
Runtime Components	**Description**
java interpreter	Dynamically interprets bytecodes to native platform
runtime libraries	Precompiled libraries get accessed by applets
appletviewer	Graphical execution environment to run applets without need for browser
demo applets	Source code and runtime to show capabilities
readme	Base documentation showing platform specifics

Figure 3. Java Development Kit contents

Java on AIX

Sun provides a set of code for Java licensees called the Java Development Toolkit (JDK). The JDK provides a complete development and runtime environment, including the JVM discussed above. IBM has made the JDK available on the AIX, OS/2, and Windows 3.1 platforms.

AIX 4.2 includes the JDK as a component of the Bonus Pack (along with others like Netscape Navigator). All Bonus Pack components are optionally installable, and the Bonus Pack itself is provided at no additional charge with AIX 4.2. Bonus Pack components are documented in both the Bonus Pack `readme` and respective component `readme` files. The JDK is also available on AIX 4.1.3 from IBM's Java Web page, referenced at the end of this article.

The JDK contents can be subdivided into two logical groups: development and runtime components. The contents are listed in Figure 3.

Java Compatible

The JDK on AIX (and on OS/2) represents a Java-compatible port of Sun's Java. Java-compatible ports are easily recognizable because they have the rights to show the "Java compatible" logo with the characteristic Java coffee cup icon.

Java compatibility is important because IBM has tested and passed the Sun test suite for the AIX and OS/2 JDKs. The objective of the test suite is to ensure interoperability between JDK ports on various platforms. Java compatibility means that AIX developers can create Java applets on AIX, and those applets should exhibit the same behaviors on AIX that are exhibited when the applet executes on another platform. The inverse is also true; that is,

applets developed on another platform (such as IBM's Windows 3.1 JDK) will exhibit the same execution behavior when run on AIX's JDK.

Newcomers to Java

If you are new to Java on AIX, try out the new features. Use `installp` or the System Management Interface Tool (SMIT) in the normal AIX fashion to install Java from the Bonus Pack. The install will deliver the files into `/usr/lpp/Java`. Before executing any of the development or runtime tools, be sure to set up your environment. Currently, to do this, add the Java binaries to your system path from the command line by typing the following:

```
export PATH=$PATH:/usr/lpp/Java/bin
```

The AIX JDK (standard) has 23 sets of precompiled demos. All can be accessed through any Java-enabled browser; they can also be executed using the appletviewer. For example, to run the BouncingHeads demo[1] type the following:

```
appletviewer /usr/lpp/Java/demo/
  BouncingHeads/example1.html
```

or

```
cd /usr/lpp/Java/demo/BouncingHeads
appletviewer example1.html
```

Note that the argument we sent to the appletviewer is an HTML file. The appletviewer cannot directly receive a Java applet, but must parse an HTML file and find an `<APPLET>` tag in the file that refers to a Java class (compiled applet). The appletviewer can also execute Java applets directly from the Internet. For example, the following command will access the IBM Hursley home page to run the multiple applets contained within that URL (currently two).

```
appletviewer http://ncc.hursley.ibm.com/
  javainfo/
```

Since the appletviewer is intended as a tool to run Java applets, it does not display the Web page referenced by the HTML. Its purpose is to run applets, so it finds all the `<APPLET>` tags embedded in an HTML file, starts independent windows on the desktop, and executes the applets in those respective windows.

Development and Runtime Tools in JDK

More intrepid AIX users will want to try out the development tools included in the JDK. The simplest place to start is by recompiling one of the existing demos, such as the ArcTest demo. This demo displays an arc segment of a circle and optionally fills the area. Compilation is simple. The compiler takes source code with the `.java` file extension and delivers compiled bytecodes contained in Java class files as output. In this example, type the following:

```
cd /usr/lpp/Java/demo/ArcTest
javac ArcTest.java
```

Use the appletviewer to test the newly compiled Java class files:

```
appletviewer /usr/lpp/Java/demo/ArcTest/
  example1.html
```

This compilation example takes the single `ArcTest.java` source file and produces three compiled class files as output (`ArcCanvas.class`, `ArcControls.class`, `ArcTest.class`). It is a common practice to define multiple object classes in a single Java source.

One of the tenets of object-oriented programming is reusability. The way to achieve that is to create modular programs that output multiple compiled object classes, which can be used in another Java program.

The primary benefit of Java technology is platform independence.

At least four different options are available to those who want to run Java programs:

- Appletviewer is included with the JDK for direct applet execution on AIX.
- Netscape Navigator (Version 3.0) has been Java enabled. This Navigator version renders both the HTML from a Web page and also executes any applets referenced (and available) by `<APPLET>` tags in that HTML. AIX 4.2 currently includes Netscape 2.0. Since IBM is very active in this area, be sure to monitor announcements for updates to AIX. Other browsers will probably be Java enabled over time.
- The HotJava browser is another option for applet execution. HotJava is a Web browser environment written in the Java language. The

[1]The BouncingHeads demo shows some heads bouncing around in a window. This demo was written by Jonathan Payne, the original author of the HotJava browser.

browser executes in the Java runtime environment. When AIX 4.2 was shipped, HotJava was only available in the alpha stage, so it is not currently included with AIX 4.2.

- Full Java applications (not applets) provide another way to run Java programs. Java applications can be executed directly from the command line by typing the following:

```
java application_name
```

The most well-known Java application is HotJava. Some differences between Java applets and applications will be discussed in the next section.

Applets Versus Applications

Distinct differences exist between applets and applications. Most Web users to date have executed applets, which are downloaded via the Web after a Web browser takes action on the <APPLET> HTML tag. Important differences between the two Java program types are highlighted in Figure 4.

Applets execute within the context of an environment whereas applications do not. Most projects that have a target audience that uses the dynamic executable content model will be programmed as applets. With applets, the program can be referenced by a Web page and loaded over the Internet. However, this model has security issues, which will be discussed in the next section.

With applets, most users expect the download time from Web server to Web client to be minimal. On intranets, bandwidth congestion is probably not a major concern. But using the Internet can be frustrating if too much time is spent during a Java applet download. As a result, most applets are small because of necessity, since users will terminate a download if too much time is taken to transfer the applet. Consequently, applications are typically larger in size than applets. New Java programmers might consider reducing the size of graphics files sent with applets.

Resource access in Figure 4 pertains to both the Java runtime environment and the respective browser environment used. Applets have many resources restricted from them. The intent behind this restriction is to keep the (few) bad guys on the Internet from creating hostile applets that can penetrate your security defenses. Programmers who must access a variety of system resources from a Java program should design their programs as full-fledged applications.

Java Applet
1. Requires a browser to execute
2. Runs within the context of the browser (usually Netscape Navigator)
3. Uses network model—applet downloaded/executed dynamically
4. Typically small
5. Has restricted access to local filesystem and to network resources

Java Application
1. Executes independently of browser
2. Requires Java runtime support; runs from command line
3. Has access to network, but invoked outside of browser

Figure 4. Differences between Java applets and applications

Security

Security of data and systems is a primary concern for everyone. To date, most security concerns related to Java have been discovered by a few researchers who have publicized these security flaws on the Web. However, few, if any, actual security incidents have been attributed to hostile Java applets, intentional or otherwise.

Java has several levels of security embedded in the technology.

Levels of Security

Java has several levels of security. In addition, different environments apply additional security policies on top of those already provided. As noted earlier, Java is delivered with AIX 4.2 as an optionally installable package. So an extremely conservative security policy would dictate that Java never be installed. Fortunately, the technology is relatively secure so this approach may be too conservative. Core security is encapsulated in the technology delivered from Sun (as shown in Figure 5).

The first level of security is found in the Java language. The language is a C++ derivative, but does not include pointers as was discussed in the Java technology section earlier. This lack of pointers helps protect memory from being accidentally (or purposefully) trashed by errant memory access outside the scope of an applet. The compiler also enforces strong typing, which ensures that the runtime variable is a subclass of the compile time type.

The next level in the security model is encapsulated in the Java runtime environment. The runtime environment is embedded in the JVM implementation as delivered from Sun in the reference port. The runtime includes a bytecode

verifier, a secure (object) class loader, and a security manager module. These modules ensure that the bytecodes comply with built-in security policies and the bytecode specification for the JVM. The policies include access restrictions, memory access constraints, system resource access, and compliance with network access policies.

Java interpreter implementation adds a third level of security. Critical policies are listed in Figure 6 for applets running under appletviewer and Netscape Navigator (Version 3.0) and also for applications.

The policies listed in Figure 6 indicate the maximum amount of authority that can be granted to an execution environment. While not all functions are configurable by the end user, default configuration for most functions is very conservative. For example, the appletviewer cannot read, write, or delete any files unless the end user explicitly allows this through configuration changes.

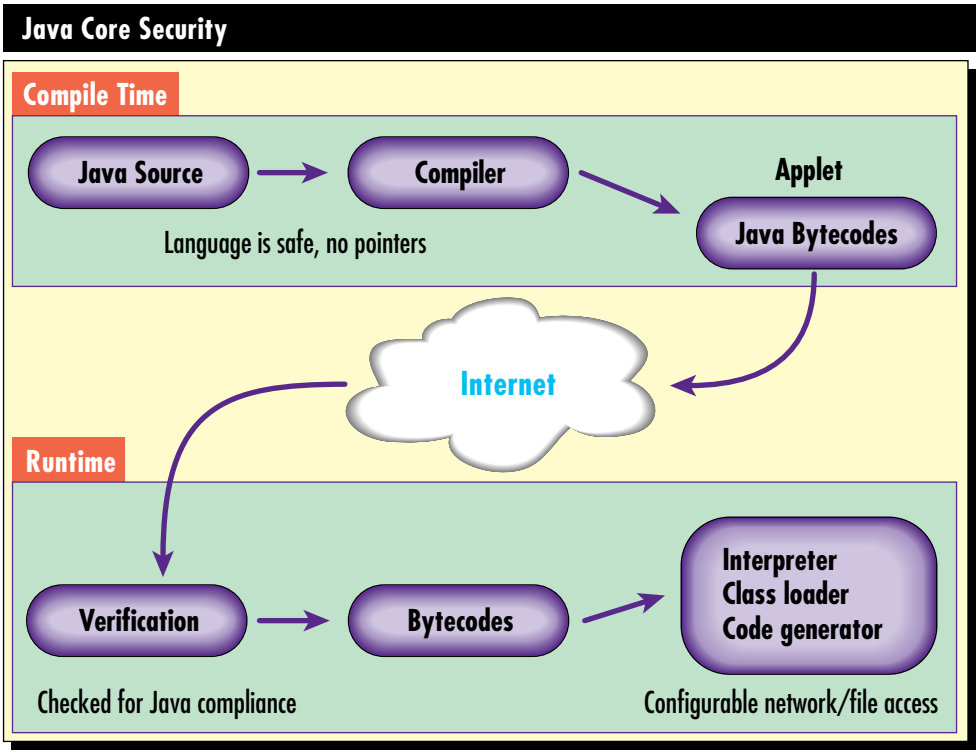


Figure 5. Java core security

There are two ways to load applets into the appletviewer and Navigator. When a user loads a URL, the result is a set of Internet transactions. Since this explicitly opens traffic to the Internet, default policy for both environments is more restrictive than the `load file` mechanism. When a user loads a Web page using `load file`, the result is retrieval of a Web page from the local (or distributed) filesystem. In this case, the appletviewer can be configured to have more access to the system, which allows programmers to take advantage of more resources.

The appletviewer is not a fully functional Web browser; Navigator will generally be used when available. The appletviewer can be liberally configured. Some sites might consider using it in an intranet environment for delivering Java applets without a full Web page environment;

Action/Function	Appletviewer load URL	Appletviewer load file	Navigator load URL	Navigator load file	Java Application
Read/write file	Yes	Yes	No	No	Yes
Get file information	Yes	Yes	No	No	Yes
Delete file	No	Yes	No	No	Yes
Spawn another program	No	Yes	No	No	Yes
Load Java library	No	Yes	No	Yes	Yes
Connect to network port on local system	No	Yes	No	Yes	Yes
Connect to network port on downloaded host	No	Yes	No	Yes	Yes

Figure 6. Applet policies

Online Resources about Java

http://www.hursley.ibm.com/javainfo/	IBM's Java Center for Java Technology
http://www.alphaworks.ibm.com/	IBM's Alphaworks (has Win 3.1 code)
http://java.sun.com/	Sun's JavaSoft™ home page
http://java.sun.com/sfaq/	Sun's JavaSoft applet security
http://home.netscape.com/	Netscape's home page
http://www.javaworld.com/	JavaWorld™ online magazine
http://www.gamelan.com/	EarthWeb's Gamelan™, applets

Figure 7. Java resources

however, it should be used with care since it allows file access.

The appletviewer ability to read and write files is restricted to files or directories specified by the Access Control List (ACL) defined in your `~/.hotjava/properties` file. By default the ACL is set to null. This prevents any open security problems for first-time users because files cannot be read or written.

Netscape Navigator is even more restrictive, with little access to the filesystem allowed. Administrators have two configuration options with Java: ON or OFF. If Navigator is configured to OFF, then Java applets are not downloaded or executed. Nevertheless, the Web page containing the Java applets is still displayed.

The last column in the table shows the power and flexibility of full Java applications. As indicated earlier, Java is a full-featured programming environment. Programmers wishing to fully exploit the power of Java will probably implement full applications.

The first two levels of security, embedded in the Java language and the runtime environment, can be exercised by competent programming. But conversely, "sloppy" usage or malicious programming has the potential to bypass even the best security architecture. The next level of defense will be found in the configuration of the interpreter environment (refer to Figure 6). Turning Java access OFF will probably turn off your users. A more realistic site policy would specify suggested Web sites with no "blind" downloads from untrusted parties. Most sites would not recommend downloads from untrusted parties in a non-Java environment. The same common sense rules apply in this situation.

Conclusions and Resources

This article has described the Java technology and its major components. One primary advantage of Java is its platform-independent execution. Another is the delivery of dynamic executable content, allowing a click on a Web page to result in the download and execution of a Java applet. The flexibility of this model has several positive implications including automatic versioning control.

In this age of Internet innovation, we must all be conscious of security while new technology is being deployed. Security policies should be extended to encompass Java applets and their execution environments.

IBM has deployed Java on both AIX and OS/2. The technology is evolving almost daily. IBM has made available a Just-in-Time (JIT) compiler for AIX—intended to resolve the runtime interpreter performance. The JIT compiler can be linked to the AIX appletviewer for enhanced runtime performance up to 25 times faster than non-JIT execution.

IBM has also delivered a Windows 3.1 port of Java, now available via a Web page. If you are interested in some of these topics, you can pursue more information or code through the Web pages listed in Figure 7. IBM also sponsors a question and answer forum on the IBM Hursley Web page. USENET news groups are also a great source of information, including the comp.lang.java.security group.

Jeff Jilg, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Dr. Jilg is currently working in AIX System Architecture at IBM Austin on leading edge technology. His recent work areas include systems management, object-oriented tools, and the Internet. His current responsibilities include release architecture, Java on AIX, Internet integration, and the AIX Bonus Pack. His PhD in Computer Science from Texas A&M complements his master's degree in the same field from the University of Texas at El Paso.

CDE Infrastructure

By George Kraft IV

The programming infrastructure, not its productivity tools, is a major strength of the Common Desktop Environment. This article discusses the APIs and desktop services that are benefiting developers and ISVs.

When Project Athena at Massachusetts Institute of Technology® (MIT®) introduced the X Window System® (X), they established the mechanism for their distributive "open" windowing protocol. Yet, they intentionally left out the policy for others to develop. Later, based on technology from Hewlett-Packard® (HP™) and Digital Equipment Corporation, the Open Software Foundation® (OSF®) developed the policies of the Motif® Graphical User Interface (GUI) for X. The Motif style guide established strict guidelines for its human-centric windowing behaviors; however, the UNIX desktop was still left without services.

The UNIX System Laboratory licensed the Destiny Desktop with System V.4, HP developed HP Vue for HP-UX, and IBM shipped IXI's X.desktop for AIX. Although these desktops provided services not offered in X and Motif GUI, they caused a divergence in the open systems market. The cost of integration and the need to remain portable kept many developers from migrating to these desktops.

The Common Desktop Environment

The Common Desktop Environment (CDE) was the result of synergy between HP, IBM, Sun, and Novell® to establish a unified open system desktop. They took the "best of breed" technology from their existing environments, then designed, implemented, and delivered a new level of UNIX desktop services on which developers could rely.

The Desktop's Programming Infrastructure

The desktop's Application Programming Interfaces (APIs) go beyond the basic Motif GUI. CDE provides data-typing, object methods, printing, drag-and-drop, additional widgets, plug-and-play messaging, and a hypertext help system. This new set of system services at the desktop level is guaranteed for all CDE-compliant systems. Developers can now produce more consistent applications and reduce the number of developer-specific core solutions.

Motif GUI

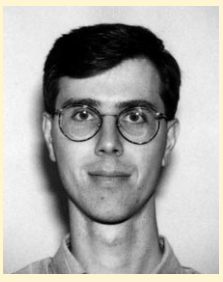
Motif won the open system GUI wars; however, vendors have serviced and enhanced various releases of Motif. Although Motif became the GUI standard, portability became risky if any of the vendors' embellishments were used. Fortunately, the creators of CDE helped eliminate those problems by merging their Motif source bases to re-establish a stock GUI.

Desktop Services

CDE goes beyond the GUI behavior to provide desktop-wide objects and methods for applications to use and call upon. Applications no longer need to depend on old and limited databases like the mime-types and mailcap that are used by applications such as mailers and Web browsers.

Figure 1 shows a message dialogue program that initializes itself with the desktop and loads the desktop's data-type and method database. Then, it simply creates a message dialogue and prompts the user.

When you select the E-mail button, the application calls the desktop's Compose method on the file object. The desktop spawns the mailer with the file object, where it is eventually displayed and ready to be addressed.



George Kraft IV

```
#include <stdlib.h>
#include <Xm/MessageB.h>
#include <Dt/Dts.h>
#include <Dt/Action.h>

void emailCB(Widget, XtPointer, XtPointer);

main(argc, argv)
        int     argc;
        char    **argv;
{
        Widget  topLevelShell, send, help, cancel;
        Arg     xargs[10];
        int     n;
        XmString title, greet, email;
        char *file = 2 == argc ? argv[1] : "/etc/motd";
        char *description;

        topLevelShell = XtInitialize(argv[0], "DtSend",
                NULL, 0, &argc, argv);

        /* CDE Initialization */

        DtInitialize(XtDisplay(topLevelShell), topLevelShell,
                argv[0], "DtSend");

        DtDbLoad();

        send = XmCreateMessageDialog(topLevelShell, "send", NULL, 0);
         /*Get CDE's "DESCRIPTION" of the file*/
        description = DtDtsFileToAttributeValue(file, "DESCRIPTION");

        title = XmStringCreateSimple("DtSend");
        greet = XmStringCreateLtoR(description);
        email  = XmStringCreateSimple("Email");

        XtVaSetValues(send,
                XmNdialogTitle,       title,
                XmNmessageString,     greet,
                XmNhelpLabelString,   email,
                NULL);

        XmStringFree(title);
        XmStringFree(greet);
        XmStringFree(email);

        cancel = XmMessageBoxGetChild(send, XmDIALOG_CANCEL_BUTTON);
        XtUnmanageChild(cancel);

        XtAddCallback(send, XmNokCallback, exit, NULL);
        XtAddCallback(send, XmNhelpCallback, emailCB, file);

        XtManageChild(send);

        XtMainLoop();
}
```

(continued on next page)

Figure 1. Simple CDE GUI application

(continued from previous page)

```
void
emailCB(w, client_data, call_data)
        Widget w;
        XtPointer client_data;
        XtPointer call_data;
{
        DtActionInvocationID actionId;
        DtActionArg actionArgs[] = { DtACTION_FILE, (char *)client_data };

        actionId = DtActionInvoke(XtParent(w),
                "Compose",              /* action */
                actionArgs, 1,          /* action arguments & count */
                (char *) NULL,          /* terminal options */
                (char *) NULL,          /* execution host */
                (char *) NULL,          /* context directory */
                True,                   /* "use indicator" */
                NULL, NULL);            /* action callback & client data */
}
```

Figure 1. Simple CDE GUI application

Drag-and-Drop

The desktop provides a convenient and consistent drag-and-drop API for interpreting data transfer across the desktop. Text, file names, and buffers can be transferred from the dragged icon to the drop zone. The type of data being dragged determines the drag icon's appearance and configuration. Since Motif can distinguish the different types of data, applications have a more robust drag-and-drop behavior.

Desktop GUI

The desktop's `DtWidget` library helps bridge the current gap between CDE's converged Motif and Motif 2.0. Developers do not need to wait for Motif 2.0 because the spin box, menu button, and editor widgets are in CDE. As always, to ensure binary compatibility, developers should take special care to use `XmResolvePartOffset` when subclassing from one widget to make another; otherwise, an updated shared library could cause unpredictable results.

Help

CDE provides standard help APIs and GUI dialogues, but it also delivers a feature-rich SGML DTD compared with the HTML used on the World Wide Web. CDE's help links support hypertext, definition, man page, execution, and application-define links.

CDE documents are pre-processed for quicker loading. Since these binary formatted documents cannot be read by a person, one of their added benefits is that they cannot be reverse engineered when copied, which prevents copyright infringements. Publishers who provide documents in HTML format are at a disadvantage, because complete unabridged duplicates can be made from most browsers.

ToolTalk Plug-and-Play

CDE's ToolTalk® is a message brokering system that enables applications to communicate with each other without having direct knowledge of one another. Application clients and servers can be developed independently, mixed and matched, and upgraded independently through plug-and-play.

Applications registered to handle message requests act as servers for applications that broadcast their requests. Message brokering is an evolutionary step beyond file sharing, peer-to-peer, and ICCCM inter-client communication.

Graphical Korn Shell

The Graphical Desktop Korn Shell provides much of the desktop's Motif GUI, services, help, workspace management, session management, and ToolTalk plug-and-play. Developers can prototype and deploy with the standard `ksh93`

scripting language. This means that small to moderate-sized programs can be written, then interpreted on any CDE-compliant system without any additional work.

The dtksh shell script in Figure 2 is a conversion of the C program that was illustrated earlier. Unlike the popular Tcl/Tk shell and GUI, dtksh has nearly a one-to-one migration path to native Motif for performance. With dtksh, code can be easily migrated and developers find that their knowledge transfers easily between C and dtksh.

The Desktop's Infrastructure

CDE not only provides a new set of Motif, Drag-and-Drop, Desktop Widget, Help, ToolTalk, and DtKsh APIs, but it also provides system services in which applications can participate and follow. The desktop services provide the login manager, session manager, color server, workspace manager, and ToolTalk server. It is tempting to provide these features in large software suites; however, if developers try to mimic these desktop services, precious development time and energy is taken away from creating the actual products.

Login Manager

The desktop login manager provides the basic X Display Manager Protocol (XDMCP) to manage login sessions for X terminals on the network and workstations on the desktop. The login manager starts up the X server on the bitmap display; it also initiates the session manager.

Session Manager

The session manager uses a set of conventions and protocols that enable the desktop to save and restore a user's session from one login session to the next. Using the session manager, users can also configure a set of sessionetc and sessionexit scripts to be called when logging

```
#!/usr/dt/bin/dtksh

editCB()
{
        dtaction Compose $FILE
}

main()
{
        XtInitialize TOPLEVEL dtSend DtSend "$@"

        DtDbLoad

        DtDtsFileToAttributeValue DESC $FILE "DESCRIPTION"

        XmCreateMessageDialog SEND $TOPLEVEL motd \
                dialogTitle:"DtSend" \
                helpLabelString:"Email" \
                messageString:"${DESC}"

        XmMessageBoxGetChild CANCEL $SEND DIALOG_CANCEL_BUTTON
        XtUnmanageChild $CANCEL

        XtAddCallback $SEND okCallback exit
        XtAddCallback $SEND helpCallback editCB

        XtManageChild $SEND

        XtMainLoop
}

if [ $# -eq 1 ]; then
        FILE=$1
else
        FILE="/etc/motd"
fi
```

Figure 2. Simple DtKsh GUI application

```
dpy  = XtDisplay(topLevel);

save = XmInternAtom(dpy, "WM_SAVE_YOURSELF", False);

XmAddWMProtocols(topLevel, &save, 1);

XmAddWMProtocolCallback(topLevel, &save, saveProc, topLevel);
```

Figure 3. WM_SAVE_YOURSELF setup

in and exiting respectively. This enables user-defined tasks to be performed at login and logout.

The key responsibility of the application is to acknowledge the WM_SAVE_YOURSELF message when the desktop is being shut down by the user, as shown in Figure 3.

The WM_SAVE_YOURSELF saveProc routine tidies up for the application, then sets the WM_COMMAND property to be saved and reused later by the session manager to restart the terminated applications.

Color Server

The session manager acts as the color server that controls foreground and shadow colors, limits color use, and restricts the creation of colors in the colormap. Using applications that conform to the color server reduces the depletion of the colormap and coordinates the color scheme of the desktop. If the colormap does become depleted, an "unsocial" application is usually lurking somewhere.

Workspace Manager

The desktop window manager (dtwm) serves as the default window manager for the desktop and extends the capabilities of the Motif window manager. It provides a control panel for the desktop to launch applications. Its multiple screens, or workspaces, allow users to switch between screens.

Desktops are often considered mutually exclusive end-point solutions, such as Web browsers or collaborative software suites. However, CDE views them as application groups or workspaces being managed and serviced by the desktop infrastructure. The desktop window manager is ideal for managing workspaces for Internet suites and collaborative tools.

ToolTalk Server

The ToolTalk server can be relied upon to broker client and server messages for inter-client plug-and-play. The Common Object Request Broker Architecture (CORBA) movement has a great deal of strength behind it. But ToolTalk is lightweight, does the job, and forms an integral part of CDE, which is deployed across a gamut of UNIX platforms.

ToolTalk takes message-handling registrations from method servers, then holds that information until client applications broadcast for those registered services. Platform gateways are not needed, because ToolTalk interoperates between heterogeneous systems.

Go For It...

There is much more to CDE than meets the eye. Application and workspace developers alike have a rich feature-filled infrastructure upon which to build. Just like in the past, we still rely on the tried and true X and Motif; however, now we can count on the Common Desktop Environment for its development libraries and desktop management infrastructure.

If you are getting ready to write a new application or just thinking about sprucing up something that you have been working on, consider how your application can become more feature rich and desktop friendly for less code.

For a list of CDE reference materials, visit IBM's CDE Web page at URL http://www.austin.ibm.com/software/CDE/TOC/Further/further.html.

George Kraft IV, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Kraft is an advisory programmer for IBM's RS/6000 Division working in the AIX Desktop group. He is currently working on the Common Desktop Environment Version 2.1 and a version of the network computer. He has a BS in Computer Science and Mathematics from Purdue University.

DirectToSOM C++ Overview

By Jennifer Hamilton

DirectToSOM C++ compilers support and enforce both the C++ and the SOM object models. This allows programmers to take advantage of SOM through the C++ language syntax and semantics so that use of SOM is transparent and efficient. This article covers some basic concepts that will help you get started more effectively with DirectToSOM C++.

The System Object Model (SOM) was designed to provide a state-of-the art object model to address problems introduced by object-oriented programming languages: Release-to-Release Binary Compatibility (RRBC) and interlanguage object sharing. SOM provides separation of interface and implementation through a language-independent model, allowing the class client and implementation to be written in different languages and a new version of a class to be supplied without requiring recompilation of unmodified client code.

DirectToSOM C++ compilers support and enforce both the C++ and the SOM object models, allowing C++ programmers to take advantage of SOM through C++ language syntax and semantics so that the use of SOM is reasonably transparent and efficient. DirectToSOM C++ is part of the IBM VisualAge™ C++ and Metaware® High C++ compiler products, and is currently available for AIX, OS/2, MVS®, Windows 95, and Windows NT™. An earlier article ("Using SOM with C++," *AIXpert*, August 1995) provided an overview of SOM and how it is used from C++. This article focuses on DirectToSOM C++ and covers the basic concepts and usage considerations.

Jennifer Hamilton

Defining DirectToSOM C++ Classes

DirectToSOM C++ classes are distinct from native C++ classes. A class is a DirectToSOM class if it inherits from the predefined DirectToSOM class `SOMObject`, defined in the header file `<som.hh>`. This can be achieved in a variety of ways:

- `#pragma SOMAsDefault(on|off|pop)` can be used to instruct the compiler to implicitly insert `SOMObject` as the base class for all classes while `on` is in effect. For example, in Figure 1, class A is a DirectToSOM class, while class B is a native C++ class. (`pop` returns to the mode before the most recent `SOMAsDefault pragma` occurrence.) The `SOMAsDefault pragma` also implicitly includes the header file `<som.hh>` if it has not already been included.

- A command-line switch can implicitly insert `#pragma SOMAsDefault(on)` at the beginning of the file. For example, with VAC++ for OS/2

```
#pragma SOMAsDefault(on)

class A {
    int a;
};

#pragma SOMAsDefault(pop)

class B {
    int b;
};
```

Figure 1. SOMAsDefault pragma

```
#include <som.hh>

class A : public SOMObject {
    int a;
};

#pragma SOMAsDefault(on)
class B {
    int b;
};
#pragma SOMAsDefault(off)

class C: private B {
    int c;
};
```

Figure 2. DirectToSOM classes

or Windows, this switch is `/Ga`; for VAC++ for AIX, it is `-qsom`.

- You can use explicit inheritance from `SOMObject` or another DirectToSOM class. For example, all three classes—A, B, and C—are DirectToSOM classes, as shown in Figure 2.

Defining a DirectToSOM class using the `SOMAsDefault` pragma is known as *implicit* or transparent mode, whereas defining a DirectToSOM class by inheriting from another DirectToSOM class (including `SOMObject`) is known as *explicit* mode.

Using DirectToSOM C++ Classes

Once you have defined a DirectToSOM class, what can you do with it? You can create SOM objects statically or dynamically, as simple objects, arrays, or as embedded members of other classes, or anywhere else that the declaration of a C++ object is valid. Figure 3 shows some simple examples (the `SOMDefine` pragma will be discussed shortly).

Most C++ rules and syntax apply to DirectToSOM classes and objects, with some restrictions. Because the size of a SOM object is not known until runtime, compile-time constant expressions such as `sizeof` are treated as runtime constant expressions. Such operators can still be used with SOM objects, but not in contexts that require compile-time evaluation.

A DirectToSOM C++ class has a SOM release order that by default will contain all member functions and static data members introduced by the class, including those with private and protected access, in the order of declaration. In general, virtual functions that override virtual

```
#include <som.hh>

// SOM class
class OuterSom : public SOMObject {
  public:
    int I;
  private:
    class InnerCPP {
// pointer to SOM object in nested native class
OuterSom *ptr;
    };
};

#pragma SOMAsDefault(on)

// SOM class
class Outer2Som {
  public:
    // embedded SOM member in SOM class
    OuterSom somobj;
  private:
    // embedded SOM class
    class Inner2Som {
int I;
    } somobj2;
};

#pragma SOMAsDefault(off)

// native class
class Outer3Cpp {
  public:
    // embedded SOM member in native class
    Outer2Som somobj;
};

#pragma SOMDefine(OuterSom)
#pragma SOMDefine(Outer2Som)
#pragma SOMDefine(Outer2Som::Inner2Som)

// array of SOM classes
OuterSom array_of_som[100];

int main(void)
{
    // automatic SOM class instance
    Outer2Som outer2_obj;

    Outer3Cpp *ptr = new Outer3Cpp;
    ptr->somobj.somobj.i = 10;

    // dynamic SOM class instance
    OuterSom *ptr2 = new OuterSom;
    ptr2->I = 10;

    // pointer to SOM class member
    int (OuterSom::*pm);
    pm = &OuterSom::I;
    (ptr2->*pm) = 15;
}
```

Figure 3. Using DirectToSOM C++ (samples.cpp)

functions in a base class do not appear in this list, but will appear in the release order for the introducing class.

Using the default, you must add any new member functions or static data members at the end of the class. Instead of relying on declaration order, you can use a pragma to specify the release order, in which case you can add new release order elements anywhere in the class, but you must add their names to the end of the list.

C++ instance data members in a DirectToSOM class are regrouped into contiguous chunks according to access, in the order of declaration within the class. This regrouping gives efficient access to data members from client code, while enabling RRBC. The location of each chunk is determined at runtime through the SOM Application Programming Interface (API). This scheme allows new data members to be added without requiring recompilation of any code outside the class under the following conditions:

- The declaration order of public and protected data within a class is not changed
- New members are added after any pre-existing members of the same access

All DirectToSOM C++ class function and data members access is performed through the SOM API, rather than the statically defined compiler constructs used by standard C++. This provides for both RRBC and an implementation-independent object model.

```
#include <som.hh>

class Som1 : public SOMObject {
};

class Som2 : public SOMObject {
};

// valid hierarchy: all SOM classes
class Som3: private Som1, protected Som2 {
};

// valid hierarchy: all SOM classes
class Som4: virtual public Som1,
    virtual private Som2 {
};

class nonSom1 {
};

// invalid hierarchy: mixing SOM and native
class mixed : public nonSom1, private Som1 {
};

// invalid hierarchy: Som2 non-virtual
// and appears twice
class Som5 : private Som3, public Som2 {
};

// invalid hierarchy:
// Som2 non-virtual in Som3, so appears twice
class Som6 : private Som3,
      virtual public Som2 {
};

// valid hierarchy:
// Som2 virtual in all bases
class Som7 : protected Som4,
     virtual public Som2 {
};
```

Figure 4. Valid and invalid class hierarchies

Basic Concepts

This section discusses some basic concepts that are important to understand when working with DirectToSOM C++.

Inheritance

SOM does not support an inheritance tree containing anything other than SOM classes. For DirectToSOM C++, this implies that a class hierarchy must contain all SOM or all native C++ classes; it does not support a mixed hierarchy.

SOM also does not permit multiple subobjects of the same type within an inheritance tree. The corresponding DirectToSOM rule is that a class may appear multiple times within a hierarchy only as a virtual base. In other words, only a single occurrence of each non-virtual base class is allowed within a SOM hierarchy. The compiler will issue a warning for each multiple occurrence of a non-virtual base class in a SOM class hierarchy. `SOMObject` is a special case because it is implicitly treated as a virtual base. To illustrate these restrictions, Figure 4 shows various combinations of valid and invalid class hierarchies.

SOM Class Data Structures

For each SOM class implementation, the DirectToSOM C++ compiler must generate several data structures and export three symbols for use by the SOM runtime. The exported symbols are `<class>ClassData`, `<class>CClassData`, and `<class>NewClass`. These symbols are used by

class clients to create and manipulate a SOM class and its instances.

The compiler should only generate these structures and symbol exports once per class implementation, otherwise there would be wasted storage and possible duplicate declaration problems. This is a sensible rule; the problem is determining when to generate the structures. In other words, how does the compiler determine, when parsing a SOM class definition, whether the implementation or the client code is being compiled? For classes that have at least one out-of-line function, the implementation is defined as the file where the definition of the first non-static out-of-line member function is defined. The compiler generates the SOM class data structures and symbol exports as part of compiling this file. This ensures that the class data structures are only defined once for that class implementation.

In Figure 5, the SOM class data structures for class A will be generated with the file that contains the definition for the member function `show`.

However, for classes that have all inline or no member functions, the compiler has no way to determine where to generate the structures. In such cases, you must explicitly indicate where the structures should be generated using the `SOMDefine` pragma. This is why the `SOMDefine` pragmas are used in the earlier example. Without them, the compiler would not generate the SOM class data structures for classes `OuterSom`, `Outer2Som`, and `Outer2Som::Inner2Som`. This would result in link errors due to unresolved implicit references to these structures in the function `main`.

The compiler will generate the SOM class data structure and symbol exports each time it encounters this pragma for a given class. Therefore, it is not a good idea to include the pragma with the class definition, but rather, in a separate file. Otherwise, the compiler will generate the structures each time the header file is parsed, resulting in wasted storage and possible duplicate definition link errors.

Although the above discussion may seem somewhat irrelevant, it is important to understand how DirectToSOM C++ classes are defined. Having duplicate definitions or no definitions for the SOM class data structures is one of the most common, and typically among the first, programming problems encountered when using DirectToSOM C++.

```
#include <som.hh>

class A : public SOMObject {
  private:
    int I;
  public:
    A() { I = 0; }
    void show();
    void set_i(int newvalue) { I = newvalue; }
    int get_i() { return I; }
};
```

Figure 5. SOM class data structure

Linking

As part of creating the SOM class data structures, the DirectToSOM C++ compiler supplies the address of each function and static data member in the class to SOM. This implies that all function and static data members must be defined by link-time because there are external references to them. If you do not supply definitions for all such members, unresolved reference errors will occur at link time. This is different from native C++, where you do not need to define a member unless it is explicitly referenced in the program. If you simply turn SOM mode on for a given class and attempt to create a library, you may discover that some methods are missing implementations that would not have mattered in native C++.

Default Constructor

You should always supply a default constructor for a DirectToSOM class. While you may not use this constructor explicitly in your application, many of the SOM frameworks, such as Distributed SOM (DSOM), require that one be present. In addition, SOM programs written using other languages typically depend upon a default constructor being available. If you are working strictly within DirectToSOM C++ only, and not using any of the frameworks, then technically you do not need to supply it. However, it is best to get in the habit and avoid bugs later on. While the SOM RRBC support makes it easy to add one if needed, runtime errors caused by a missing default constructor can sometimes be difficult to track down.

Header Files

As you have probably noticed, the DirectToSOM C++ header files used so far all have a file extension of `.hh`. Always put DirectToSOM C++ classes in a file with a `.hh` extension. This is not simply a convention; it is required for Interface Definition Language (IDL) generation. Also with regard to header files, you cannot mix the C++ bindings `.xh` header files with the DirectToSOM C++ `.hh` header files in a single compilation unit. The reason is that each provides different definitions of classes such as `SOMObject`. Note that you can mix programs compiled with these different headers at runtime and share SOM objects between them, but you cannot mix them at compile time.

Name Mangling

SOM is case insensitive, so all names presented to it must be unique without respect to case. In particular, class names cannot differ only by case. In order to ensure that unique DirectToSOM C++ names are also unique in SOM, class and member names are subject to a case-insensitive conversion:

- Upper case letters are converted to the lower case equivalent, with a z that precedes the lower case z
- `z_` is used to mean lower case z

Thus `Hello` becomes `zhello` and `ZebraClassZz` becomes `zzebrazclasszzz_`. This converted name is known as the SOM name, as opposed to the C++ name. For example, `zhello` is the default SOM name for the C++ class named `Hello`.

SOMObject Methods

The `SOMObject` base class, from which all DirectToSOM C++ classes derive, defines ten special methods to which certain C++ methods are mapped. It is worth knowing that this mapping is taking place, particularly because the mapped C++ methods are considered overrides of the `SOMObject` methods, rather than newly introduced methods in the class. For a given class `X`, C++ class methods, if supplied, are mapped to the ten special SOM object methods as shown in Figure 6.

Metaclasses

The model that SOM supports is similar to the Smalltalk® model, because classes are not purely syntactic entities as in C++, but are themselves objects. SOM class objects, created at runtime as required by the client, are used for creating and manipulating instances. Class objects support a variety of methods for creating and querying objects, such as determining the size of class instances, whether a method is supported by a given class, and whether a given object is a member of that class.

Figure 7 illustrates this model. As shown at the top of the figure, a native C++ class is a syntactic entity whose definition is compiled into the program object. A native C++ class has no representation except for the source code that defines it. However, with the SOM model, as shown at the bottom of the figure, a SOM class is also an object that exists at runtime. Each SOM class object is an instance of a special class, called a *metaclass*, which by default is the class `SOMClass`. In the same way that a class defines the behavior of its instances, a metaclass defines the behavior of its instances, which are class

```
X()                                somDefaultInit
~X()                               somDestruct
X(X&)                              somDefaultCopyInit
X(X const &)                       somDefaultConstCopyInit
X(X volatile &)                    somDefaultVCopyInit
X(X const volatile &)              somDefaultConstVCopyInit
operator=(X&)                      somDefaultAssign
operator=(X const &)               somDefaultConstAssign
operator=(X volatile &)            somDefaultVAssign
operator=(X const volatile &)      somDefaultConstVAssign
```

Figure 6. Special SOM object methods

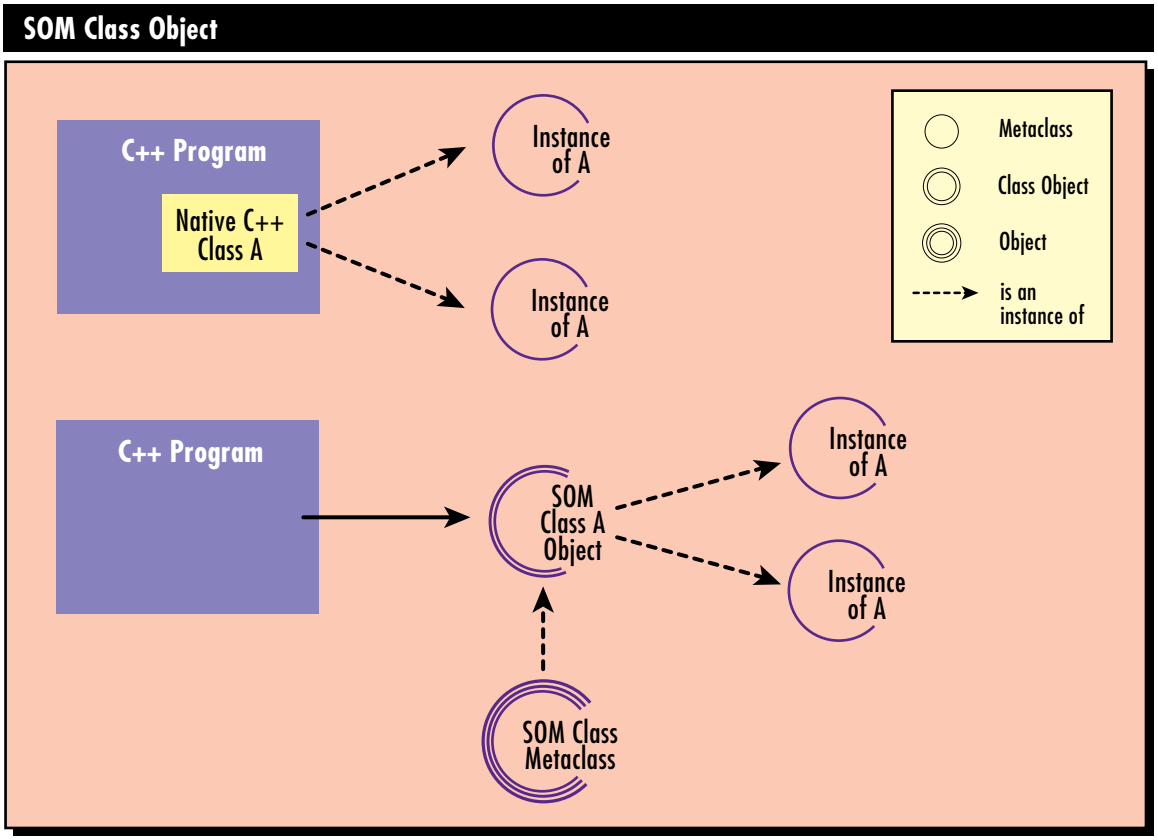


Figure 7. SOM class object

objects. For example, the class A can define the method `foo`, in which case `foo` can be invoked against all instances of A.

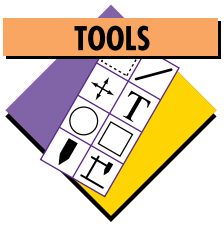
Class object methods typically deal with the creation and destruction of class objects. For example, `SOMClass` defines methods such as `somNew`. Since the class object A is an instance of the class `SOMClass`, the method `somNew` may be invoked against that class object in order to create a new instance of A.

If this is the first time you have been exposed to the concept of a metaclass, it may seem a little strange at first compared to native C++. It is not a complicated concept—it is really just a difference in how things are done in the model (although the metaclass concept is much more flexible than the native C++ model). For example, with native C++, you can create instances by invoking the `new` operator, which is applied to a class name. With SOM, the corresponding operation is to invoke the method `somNew` against the appropriate class object. In general, you do not need to deal much with metaclasses when working with SOM, but it helps to understand the concept.

Conclusion

This overview has described how DirectToSOM C++ classes are defined and used, and some of the basic programming considerations for using DirectToSOM C++. For details on these and other SOM-related topics, see *Programming with DirectToSOM C++* published by John Wiley & Sons.

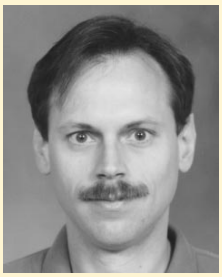
Jennifer Hamilton, IBM Toronto Laboratory, 1150 Eglinton Avenue East, North York, Ontario M3C 1H7. Ms. Hamilton has worked in compiler development since joining IBM in 1987, and currently develops the non-native object model support for IBM's VisualAge C++ language products. She is the author of three books and numerous articles on programming language-related topics. She has a BSc in Computer Science from the University of Victoria, British Columbia.



utld— A Trace-based Performance Tool

By Bret R. Olszewski and Jim Van Fleet

Trace-based analysis provides an excellent means of analyzing the behavior of an AIX system. The utld tool provides a convenient way to analyze traces. It is particularly effective for identifying problems associated with high kernel CPU usage. This article describes the utld performance tool used in AIX development, now available via the Internet.


Bret R. Olszewski


Jim Van Fleet

System performance analysis is dependent on tools—tools that do time-based profiling (`tprof`), tools that monitor system resources over time (`iostat`, `vmstat`, `sar`), tools that evaluate instantaneous resource consumption (`svmon`), and event-based tools. The new `utld` performance tool provides valuable information for analyzing traces in AIX.

Trace is a particularly useful built-in instrumentation capability in AIX. Trace contains two parts: hook entries and buffer management. Hook entries are macros included in source code that selectively insert events into a trace buffer.[1] Trace events, which are usually time-stamped, have fixed information fields. When trace is on, the selected trace events are placed into a trace buffer in the order they are generated.

The trace buffer either collects events until the buffer is filled or double buffers the events. When the buffer fills, trace writes them to disk or to a trace reader application such as `tprof`.[1] Because trace adds significant system overhead in path length, disk space, and memory consumption, it does not run continuously; only the root user can enable it. Since trace is invasive, it should be used selectively to address problems. It has proven useful for identifying functional problems, such as application problems and performance analysis.

Using AIX Trace

This section contains recommendations on using AIX trace, but for definitive documentation on trace, consult the standard AIX documentation. To invoke trace, use a set of parameters such as the following:

```
trace -a -d -f -T 8000000 -L 8000000 -o
  ./trace.out
```

The classic time-versus-resource trade-off exists with trace. On a fast, busy system with all trace hooks collected (the default), trace will produce megabytes of trace output for each second of real time. This requires focusing on the phenomenon that will be studied. In the following example, the trace facility is started, detached (`-a`), and deferred (`-d`); trace runs in the background, but it is not yet collecting trace hooks.

Trace will collect eight million bytes of trace data (`-T`) and stop collecting trace data when the buffer fills (`-f`). The system must have sufficient memory for the eight million bytes to be dedicated to trace; otherwise, the trace command will fail or the workload will be perturbed (perhaps by excessive paging). The maximum length of the trace output file is eight million bytes (`-L`); the output is written to the file `./trace.out` (`-o`). When deferred trace collection is used, the `trcon` command starts trace and the `trcoff` command ends trace collection.

[1] Waters, Frank. *AIX Performance Tuning.* Englewood Cliffs, N.J.: Prentice Hall 1996. p. 214.

The following shows how to trace the program `dummy1`:

```
trace -a -d -f -T 8000000 -L 8000000 -o
   ./trace.out
trcon; ./dummy1; trcoff
```

The `-f` flag indicates that the buffer is filled and trace will end. If this happens, the `trcoff` command may fail, and an error message will appear stating that trace is not currently running.

If the workload is more complex or running in the background (for example, a database engine), the same basic technique can be used. When the workload is exhibiting interesting behavior, the trace can be started as follows:

```
trace -a -d -f -T 8000000 -L 8000000 -o
   ./trace.out
trcon; sleep 60; trcoff
```

If the trace was started with the `-f` flag, it is unlikely that 60 seconds of trace will be collected (`sleep 60`); instead, we will get one full trace buffer that will generally contain events corresponding to less than the requested time. Since trace adds considerable path length in system time, using trace will distort the user/system time mix of the system.

The trace data is collected in a binary file that the `trcrpt` program can translate into a readable file. The example in Figure 1 collects a trace of the `find` command and generates a readable view of the trace via `trcrpt`.

The `trcrpt` output includes ID, the unique identifier of the trace hook, the elapsed time in the trace, the delta milliseconds between adjacent trace hooks, and some hook-specific text. The `trcrpt` program formats trace hooks via rules found in the file `/etc/trcfmt`.

How to Get utld

The `utld` tool can be obtained via the Internet with normal browsers by opening http://www.software.ibm.com/download. From there, select Free Software Packages. This directory contains `utld` (an AIX performance tool) as an `installp` image.

The `utld` tool is currently provided free of charge; however, it also comes without the warranty or support provided for products.

State-full Analysis

Trace events are time-stamped, which enables the trace to paint a complete picture of system behavior. The most interesting events are bounded by hooks indicating the beginning and end of activities. For example, when a thread is dispatched, a trace hook appears; then trace hooks appear when system calls enter and exit the system.

From this we can deduce that the running process is responsible for the system calls while it is dispatched. Similarly, we can see that a running thread is preempted by an interrupt, so the CPU consumption of the interrupt handler can be accounted separately from the thread that started the time slice. Although not technically difficult, it would be tedious to write analysis programs to parse the trace. For that reason, we provide `utld` to summarize system behavior with special emphasis on CPU consumption.

```
trace -a -d -T 2000000 -L 6000000 -f -o ./trace.out
trcon; find . -type d; trcoff
trcrpt ./trace.out | pg

ID    ELAPSED_SEC    DELTA_MSEC    APPL      SYSCALL  KERNEL  INTERRUPT

001   0.000000000    0.000000      TRACE     ON       channel 0
                                                      Fri Apr  5 08:54:29 1996
101   0.000165888    0.165888      close LR = 10003F0C
12E   0.000178432    0.012544      close fd=6
104   0.000280320    0.101888      return from close [114 usec]
100   0.000331776    0.051456      DATA ACCESS PAGE FAULT
1B2   0.000357888    0.026112      VMM pagefault:    V.S=01F5.10C6
                                                working_storage
1B0   0.000546048    0.188160      VMM page assign:          V.S=01F5.10C6 ppage=05B5
```

Figure 1. Output of trcrpt

Using utld

Three steps are required to use `utld`. First, collect a trace. Second, use the `trcnm` tool to collect the names of the system. This tool builds a load map of the contents of the AIX kernel so the trace can identify items such as device drivers and kernel extensions. Typically, a command such as `trcnm > names.out` can collect names. Another useful command is `lsdev -C adapter`, which provides an explanation of each adapter driver in your system. This can be helpful in identifying device driver names that map to adapters.

The last step is to "unwrap" the trace. Since the trace is double-buffer collected, the log file must be put in the correct order. Since this step requires two copies of the trace file, be sure that sufficient disk space is available. The `trcrpt` command shown below unwraps the trace:

```
trcrpt -r trace.original > trace.unwrapped
```

Now, the `utld` command can be invoked as follows:

```
utld -I trace.unwrapped -n names.out >
   utld.out
```

After `utld` is complete, do not be surprised if the `utld.out` file is quite large.

CPU Utilization

The first section of `utld` output (see Figure 2) summarizes CPU consumption during the trace period. The CPU consumption has six categories.

Processing Total (msecs)	Percent Total Time	Percent Busy Time	Processing Category
43337.935	49.529	49.531	Application
34317.186	39.220	39.221	Kernel
5203.835	5.947	5.947	FLIH
3103.539	3.547	3.547	SLIH
1534.118	1.753	1.753	Dispatch
87496.612	99.997	100.000	CPU(s) Busy Time
2.911	0.003		WAIT
87499.523	100.000		Total

Total number of process dispatches = 34734
Average time between same process dispatch = 54.738713 msec
Average process to processor affinity = 0.454131

Figure 2. Processor utilization system summary

User: Total time spent running user programs and commands.

Kernel: Time spent in the operating system while executing system calls. For example, when a program reads data from disk, the disk I/O is handled via a system call.

First-Level Interrupt Handler (FLIH): Total time spent by the system in first-level interrupt handlers. This is time spent in the AIX kernel dealing with interrupts, both external (such as I/O) and internal (such as page faults). To maintain a high level of responsiveness, I/O interrupt FLIHs execute quickly. Although page fault FLIHs take much longer to execute, they reduce their interrupt priority levels so they can be preempted by higher-priority interrupts.

Second-Level Interrupt Handler (SLIH): Total amount of time spent in second-level interrupt handlers. Again, this is time spent in the AIX kernel, also dealing with interrupts. Second-level interrupt handlers can execute considerably longer than first-level interrupt handlers, so they occasionally represent a significant amount of CPU utilization.

Dispatch: Time that the system spends dispatching processes. Although the measures of this section are not as precise as the others, they still provide good insight into dispatching overhead.

Idle: Time that the system was idle.

Note that `utld` accounts for CPU utilization in six categories; four (Kernel, FLIH, SLIH, and Dispatch) represent system time. More commonly used tools, such as `sar`, `iostat`, and `vmstat` report total time spent in the system, but `utld` enables further analysis of the sources of system time CPU consumption by category.

CPU Utilization on SMP

For a trace collected on an SMP system, the CPU breakdown includes a weighted average of all processors, as well as information for each processor. On SMP, Figure 3 also details the number of dispatches and a measure of the processor affinity of threads, the number of times a thread was redispatched on the same processor on which it was last dispatched.

Misinterpreting Trace Data

There are several ways `utld` can be fooled into misinterpreting trace data. A frequent case is the accounting of kernel time by tracking system call entry and exit. If a thread has entered the kernel before the trace begins, its time may be improperly accounted to user time. This happens with

Network File System (NFS) daemons because they execute in the AIX kernel and never return to user code; therefore, `utld` does not account them correctly.

Another example is system calls that do not return, such as `sigreturn`, which is special cased in `utld` to account a small amount of time to kernel. Compare `utld` output to other tools such as `sar`, `iostat`, and `vmstat` if `utld` output is suspect.

Wait Summary

The next section summarizes idle time per processor during the trace. It includes statistics about the number of times each processor was idle as well as the minimum, average, and maximum durations of idle time. This section is typically not interesting for performance analysis.

Application and Kernel Summary (Per Thread/Process)

This section categorizes CPU consumption—user and kernel—by individual threads. There is an entry for each thread that was dispatched during the traces, as well as the thread's process name. The `ps` command produces similar information.

Figure 4 shows individual thread data from the application and kernel summary output.

Application and Kernel Summary

This section, although similar to the previous one, has thread CPU consumption rolled up by process name—all processes with the same name are summarized in one entry. For example, if 100 instances of the program `awk` are dispatched in the trace, the total cost of running all instances of `awk` is represented. This information is useful if several workloads, identified by unique names, are present on the same system. For example, if DB2® and Lotus Notes® were running on the same system, the percentage of CPU resource consumed by each in the trace can be identified.

Figure 5 shows example application and kernel summary output.

Processing Total (msecs)	Percent Total Time	Percent Busy Time	Processing Category
5439.218	49.730	49.730	Application
4258.144	38.932	38.932	Kernel
1051.193	9.611	9.611	Interrupt
188.885	1.727	1.727	Dispatch
10937.440	100.000	100.000	CPU Busy Time
0.000	0.000		WAIT
10937.440	100.000		Total

Total number of process dispatches = 4287

Figure 3. Processor #2 summary for an SMP

Kernel (System Call)

The Kernel (system call) section summary is generally the most useful if the workload is CPU bound and contains a high percentage of time (>40%). It summarizes total kernel time CPU consumption by system call.

Since interrupt handlers (FLIH and SLIH) and Dispatch do not often contribute significantly to CPU consumption, system calls are usually the villains for high system time workloads. If the workload contains a high percentage of time in the operating system, this report usually indicates the activity that causes the system time.

Figure 6 shows example kernel (system call) output.

Processing Total (msecs)			Percent of Total Processing Time			Process Name
Combined	Application	Kernel	Combined	Application	Kernel	(process id/thread id)
1333.238	1333.238	0.000	1.524	1.524	0.000	kproc (3096 3355)
1269.453	1269.453	0.000	1.451	1.451	0.000	kproc (3096 3613)
1244.047	1244.047	0.000	1.422	1.422	0.000	kproc (3096 4129)
906.621	0.033	906.588	1.036	0.000	1.036	syncd (4896 5673)
	* * *					
207.807	108.782	99.025	0.237	0.124	0.113	ora7_srv6.1 (96548 367917)
207.657	207.657	0.000	0.237	0.237	0.000	swapper (0 3)
207.204	75.739	131.465	0.237	0.087	0.150	ora7_srv6.1 (342492 342501)
205.972	95.343	110.629	0.235	0.109	0.126	oracle (100388 100397)

Figure 4. Individual thread data excerpt

Processing Total (msecs) Combined	Application	Kernel	Percent of Total Processing Time Combined	Application	Kernel	Process Name (process id/thread id)
4586.477	4586.477	0.000	5.242	5.242	0.000	kproc (7)
906.621	0.033	906.588	1.036	0.000	1.036	syncd (1)
40091.177	21763.985	18327.192	45.819	24.873	20.945	oracle (221)
30617.885	16143.846	14474.040	34.992	18.450	16.542	ora7_srv6.1 (198)
207.657	207.657	0.000	0.237	0.237	0.000	swapper (1)
1224.767	624.459	600.308	1.400	0.714	0.686	bisam_srv6.1 (75)
12.802	10.412	2.390	0.015	0.012	0.003	cron (1)
7.086	0.884	6.202	0.008	0.001	0.007	shmtimer6.1 (1)
0.650	0.184	0.466	0.001	0.000	0.001	trace (1)

Total number of processes = 506

Figure 5. Application and kernel summary data

Processing Total (msecs)	Percent Process Time	Count	Path in msecs Minimum	Average	Maximum	System Call
15461.009	17.670	000025225	0.042	0.613	7.556	kreadv
10177.104	11.631	000025135	0.058	0.405	5.577	kwritev
2180.326	2.492	000036739	0.014	0.059	0.309	kioctl
	* * *					
0.000	0.000	000000001	0.000	0.000	0.000	sigreturn
34317.186	39.220					

Figure 6. Kernel (system call) excerpt

FLIH

FLIH shows CPU time used by first-level interrupt handlers, including instruction access page fault, data access page fault, I/O interrupt, decrementer, level 50, and floating-point unavailable.

Instruction access page faults are caused by attempting to fetch an instruction to execute when the page containing the instruction is not resident in memory. Instruction page faults are often resolved by a disk I/O to bring the page into memory. Similarly, a data access page fault is caused by a load or store to data where the data page is not resident in memory. Data faults can be resolved by creating a new page (demand zero) or by reading static data from disk (vmapped fault).

Another way to view page fault activity is through `vmstat`, particularly `vmstat -s` that details fault activity since the system was booted. Interrupts caused by external devices, such as disks and network adapters, cause I/O interrupts. This interrupt handler identifies the device causing the interrupt and calls the appropriate device driver.

The decrementer interrupt is the system heartbeat. On a uniprocessor system, the processor is interrupted 100 times per second for clock management and process dispatching. On an SMP system, each processor is interrupted 100 times per second. Normally, 1% to 2% of total system time is consumed by this interrupt because it does considerable housekeeping work.

The level 50 software interrupt typically serializes communications adapter drivers. The floating-point unavailable interrupt results from AIX's lazy save of floating-point registers. On context switch, the floating-point registers are not saved. When another process attempts to do a floating-point operation, an interrupt occurs to save the floating-point registers for the last thread and restore the registers for the current thread.

The lazy register save is typically good for floating-point applications, because frequently only one job or a few large jobs are executing. Therefore, the floating-point registers often do not need to be saved and restored across several context switches.

Processing Total (msecs)	Percent Process Time	Count	Path in msecs Minimum	Average	Maximum	SLIH Type
2991.944	3.419	000006486	0.027	0.461	6.276	entdd
76.205	0.087	000000256	0.113	0.298	1.407	sdpin
35.389	0.040	000000129	0.133	0.274	0.513	ascsiddpin
3103.539	3.547	000006871				

Figure 7. SLIH summary

SLIH

SLIH shows CPU time used by second-level interrupt handlers, such as device drivers for disks, communications adapters, terminal equipment, as well as others. Some drivers, particularly communications adapters, may spend considerable time in SLIHs. Figure 7 shows sample SLIH output.

Detailed Process Reports

This final section summarizes CPU consumption for each process in the system. It includes system call use; on SMP traces, it includes detailed dispatch information. This section can help determine which processes heavily use system calls. If many threads were run during the trace period, this section would be very large.

Lock Reports

As an option, `utld` will include a detailed report on lock utilization from a trace that has been collected with instrumentation enabled.[2]

To get the `utld` lock report, invoke the `utld` command with the `-l` option as follows:

```
utld -I trace.unwrapped -l lock.out -z -n
   names.out > utld.out
```

The lock output file will contain multiple sections. The Processing Time Summary is the same as the System Summary under CPU Utilization. A Lock Summary report shows cumulative times per processor of lock-held time, processor time spent processing a miss on a blocking lock, and processor time spent spinning. The "unknown" processor represents those lock events in which `utld` processes the unlock trace entry, but not the dispatcher entry (which tells `utld` which processor is executing) and/or the lock trace entry.

Figure 8 shows the lock summary by lock type.

Cumulative Lock CPU Times Per Processor (in msec) Processor	Lock Type	Held	Block Miss	Block Spin
0	swrit	4952.667	12.606	1041.029
0	cwrit	78.564	0.000	0.000
0	lockl	0.068	0.000	0.000
1	swrit	4559.844	14.514	1163.660
1	cwrit	57.259	0.000	0.000
2	swrit	4622.448	13.909	1075.810
2	cwrit	77.352	0.000	0.000
3	swrit	4743.197	13.244	1104.009
3	cwrit	152.326	0.000	0.000
4	swrit	4646.445	13.203	1206.241
4	cwrit	183.835	0.000	0.000
5	swrit	4450.559	16.550	1149.127
5	cwrit	58.848	0.000	0.000
6	swrit	4657.279	15.180	1180.408
6	cwrit	178.170	0.000	0.000
7	swrit	4765.941	13.275	1159.256
7	cwrit	120.007	0.000	0.000
unknown	swrit	5.517	0.000	0.123

Figure 8. Lock summary by processor by lock type

The `utld` individual lock summary provides extensive data on each lock referenced during the trace being examined. The miss probability is the number of misses (spin or block) divided by the total count of attempts. In this case, the miss probability is 53.77%.

The collision cross section is the held, blocked, and spin time divided by the total time. In the SMP case, the total time is the time for all processors. In Figure 9, the Held Elapsed cross section is 0.070—7% of the total time; or, cumulatively, the equivalent of 56% of one processor was spent under the lock in question and 6.9% of the total time was spent spinning, waiting for the lock.

For busy locks, if the miss probability is greater than 5% and/or the collision cross

[2] See documentation on `lockstat` for a description of how to enable lock instrumentation. Information is available on InfoExplorer™ under the `bosboot` command.

Lock: proc_base_lock				0015b7f8 (00000000)			miss prob. = 0.5377			
Total	msec while Held		Blocked	Blocked Miss	msec		Blocked Spin	msec		
Count	Cpu	Elapsed	Trys	Count	Cpu	Elapsed	Count	Cpu	Elapsed	
113147	5489.215	6136.699	0	0	0.000	0.000	60840	6038.430	6038.430	total
113147	0.049	0.054	0	0	0.000	0.000	60840	0.099	0.099	average
	0.063	0.070			0.000	0.000		0.069	0.069	collision xsection

Figure 9. Individual lock summary

```
112  0.001360256 0.004352  lock:   lock lock addr=15B7F8 lock status=1C1F5
        requested_mode=LOCK_SWRITE return addr=18560 name=00A5.0002
```

Figure 10. Trace report lock entry

```
25732  0.076  0.076  0  0  0.000  0.000  13363  0.092  0.092  swrit  .kwakeup          000174c4
(00000064)
25546  0.019  0.019  0  0  0.000  0.000  15564  0.107  0.107  swrit  .e_block_thread   00018560
(000001bc)
25738  0.015  0.015  0  0  0.000  0.000  14775  0.097  0.097  swrit  .e_assert_wait    00018938
(000000a0)
```

Figure 11. Detailed individual lock data excerpts

section is greater than 20% divided by the number of processors (or 2.5% for an 8-way processor), then the possibility that a locking problem exists is high. This example has a very stressed `proc_base_lock`.

The `utld` individual lock summary names the lock that provides data. Since the data used by `utld` when making the report is often not sufficient to provide the name, it simply gives the lock address. The identity of the lock, in terms of class and instance in class, can be determined by using `utld` output, `trcrpt`, and `/usr/include/sys/lockname.h`. The `trcrpt` output for a simple lock provides the lock address (which correlates with the third field of the first line of the individual lock summary) and the lock name. Figure 10 shows the trace report lock entry.

The lock name consists of the class number (00A5) of the lock given at `lock_alloc` and the instance in the lock class (0002). Correlating the class number 0x00A5= decimal 165 in the `lockname.h` file shows that the lock in question is the second instance in the `PROC_INT_CLASS`.

The voluminous data after the individual lock summary (see Figure 11) can be helpful in understanding why a particular lock is busy. It provides insight into the locks held when this lock was blocked, as well as individual function statistics on attempts, blocks, and so on. In this case, much of the effort is spent waiting and waking up. In the database example, the kernel is used primarily to initiate I/O and to wait for its completion.

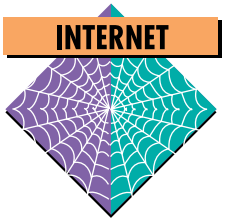
Conclusion

Trace-based analysis provides an excellent means of analyzing the behavior of an AIX system. The `utld` tool provides a convenient means of analyzing traces and identifying problems associated with high kernel CPU usage.

Bret R. Olszewski, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Olszewski is a senior programmer working on MP performance. He has a BS in Computer Science from the University of Minnesota.

Jim Van Fleet, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Van Fleet is a senior programmer in AIX Performance in Austin. He has held several technical and managerial positions at IBM, with a specialty in Symmetric Multiprocessing systems. Mr. Van Fleet has a BS in Mathematics from Michigan State University and an MS in Computer Science from Union College.

Internet Naming and Address Management



By Eddie Ho and Greg Althaus

AIX Version 4 has built-in support for Dynamic Host Configuration Protocol (DHCP) and Dynamic Domain Name Service (DDNS) enhancements. These enhancements help to automate IP addresses and other tasks of the Domain Name Server, while eliminating downtime and costly diagnostic work.

Dynamic Host Configuration Protocol (DHCP) with the Dynamic Domain Name Service (DDNS) extension provides a way to automatically track and verify IP addresses. The increasing demand for connections to the Internet makes this essential.

DHCP

DHCP automatically assigns IP addresses and configuration parameters from a data pool of addresses that reside on a DHCP server. DHCP can also provide configuration parameters, such as default gateway, name servers, X-Windows parameters, and other user-defined values.

Traditionally, as users move and connect a new system to a TCP/IP network, the user is given a new IP address, default gateway, name server, and other required parameters. Users then manually enter these parameters to the TCP/IP configuration file and restart the protocol stack in the new location. With DHCP in AIX, human intervention is not necessary. It is as simple as "pack-and-move."

Three Internet Engineering Task Force (IETF) Request-For-Comments (RFC) documents define the architecture:

- **RFC 1533:** DHCP provides a framework for passing configuration information to hosts on a TCP/IP network. This memo describes the DHCP options and Bootstrap Protocol (BOOTP) vendor extensions.
- **RFC 1541:** Framework for passing configuration information to hosts on a TCP/IP network. DHCP is based on BOOTP, adding the capability of automatic allocation of reusable network addresses and additional configuration options.
- **RFC 1542:** Redefinition of BOOTP relay agents in the DHCP environment. This memo attempts to clarify and strengthen the specification in these areas.

The solution framework is a two-tiered client/server computing model. The client must be DHCP enabled, which can be either AIX Version 4 or PCs with TCP/IP support and DHCP client functionality. The client enters a network in an initializing state and broadcasts a discover message on the IP network. The message is then relayed by the BOOTP relay agent, and eventually delivered to the DHCP server for processing. The DHCP server responds to the client request with an offer message that consists of an available IP address and configuration information. The client then binds the IP address to the TCP/IP stack and restarts the protocol initialization.

The DHCP client acknowledges the DHCP server, which in turn updates the DNS server accordingly. AIX Version 4 supports both the client and the server implementations with DNS synchronization. Figure 1 summarizes multiple IP networks with DHCP and DDNS serving capabilities.

Protocol Flow between DHCP Client and Server

DHCP is an application-layer protocol that allows services to be solicited and negotiated from one or more servers. The protocol framework is BOOTP, which runs on User Datagram Protocol (UDP).

If network size requires cross-network DHCP serving, the inter-network router must support the BOOTP relay agent. This relay agent, responsible for the BOOTP message passing between two IP networks, is required if the DHCP clients and server are not in the same network. Cross-network DHCP serving can potentially cause massive broadcast storms if the network size is too large; in this case, careful design with one or more DHCP servers is recommended. The AIX implementation for both client and server portions consists of the following modules:

- Client daemon (dhcpcd)
- Server daemon (dhcpsd)
- Relay agent daemon (dhcprd)

Figure 2 shows a simple flow sequence for an assignment.

Shopping for an IP address begins from the DHCP client. The client broadcasts a message (DHCPDISCOVER) to all DHCP servers soliciting for service. This packet contains the client's connection requirements based on the current environment. All DHCP servers can receive DHCPDISCOVER broadcasts. One or more DHCP servers can respond with unicast or broadcast with IP address offer (DHCPOFFER).

The client receives all offers and determines those most desirable using the built-in algorithm. Usually the best match for suggested options will win. The client then broadcasts its response to the DHCPOFFER using a DHCPREQUEST packet. The target DHCP server will send a DHCPACK to acknowledge service rendered. Subsequently, all other servers that are not chosen will free up their address and return to listening mode for the next DHCPDISCOVERS packet.

Enterprise Network with DHCP/DDNS

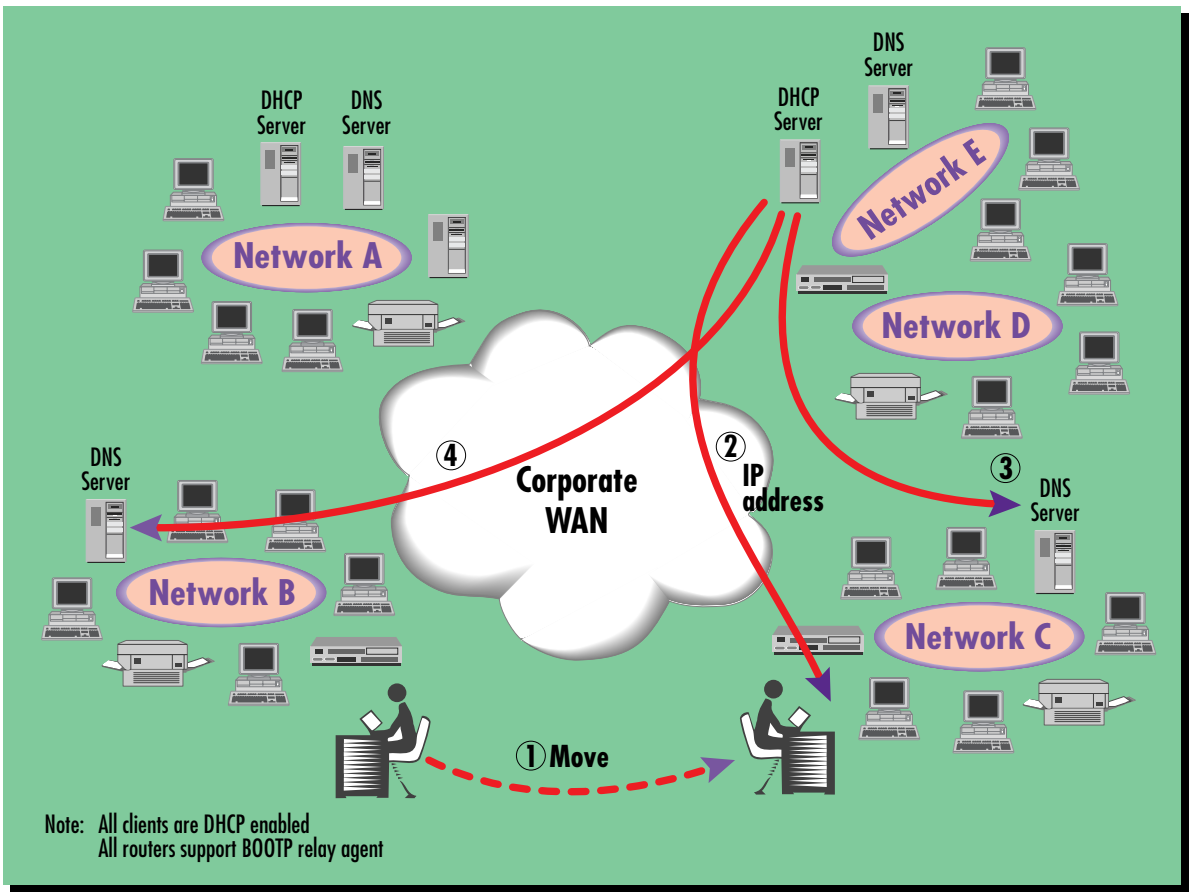


Figure 1. IP networks with DHCP and DDNS

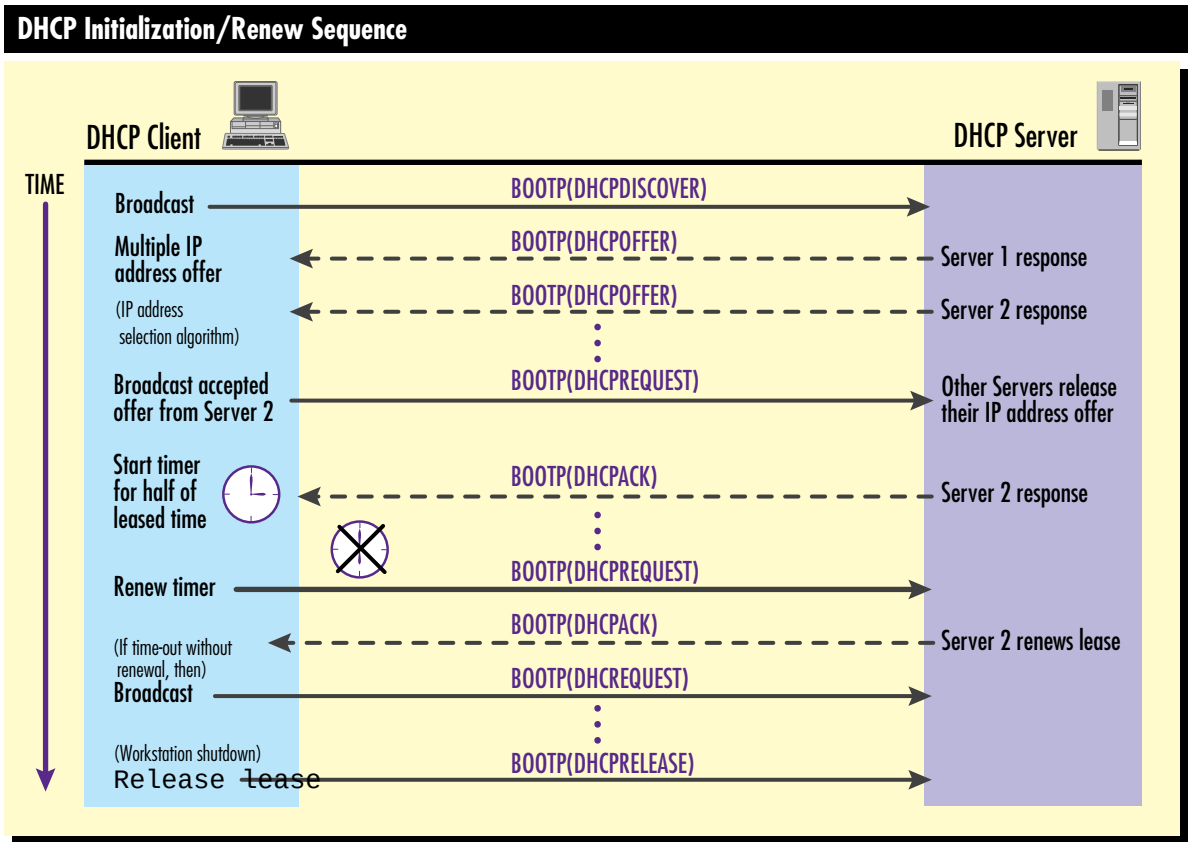


Figure 2. Flow sequence for an assignment

The client IP address is leased from the server and must be renewed periodically; therefore, a timer is started by the client and set up for half of the lease time. When it expires, the client sends a `DHCPREQUEST` packet directly to the server for renewal. If the client receives no response, the client will wait up to three fourths of the leased time, then broadcast a `DHCPREQUEST` packet soliciting for the new server. This ensures that the DHCP service continues without interruption.

AIX 4.1 DHCP/DDNS

The RISC System/6000 (RS/6000) DHCP design for both client and server conforms to all existing available RFCs. It is also tested for interoperability with other DHCP clients or servers in the marketplace. It currently supports LAN-based protocol, including Ethernet™, Token-Ring, and Fiber-optic Data Distribution Interface (FDDI) interfaces.

Figure 3 shows the latest interoperability list of confirmed products supported by the RS/6000. In fact, any product that supports the above RFCs should be able to interoperate without major problems. The DDNS implementation is based on the latest draft of RFC that is pending for approval. Few products in the marketplace offer both DHCP and DDNS capabilities; but none are based on the proposed draft. The AIX version will conform to the final draft when available.

Interoperability Products Supported by RS/6000

DHCP Server	DHCP Clients
AIX 4.2	MacOS, Windows 95, Windows NT 3.5, Windows 3.1, Chameleon, FTP Software
AIX 4.1.4	OS/2 Warp™ Connect
OS/2 Warp Server Sun FTP Software JOIN by Competitive Automation	AIX 4.2/AIX 4.1.4

Figure 3. Interoperability products supported by RS/6000

Configuration of the DHCP support can be grouped into three independent areas: server, client, and the BOOTP relay agent. Each RS/6000 can be configured for one of the three portions.

Configuring the DHCP Server

The DHCP server generates and offers IP addresses based on a set of predefined attributes or a current network service environment. The definition service is done by editing a flat ASCII file or using the Graphical User Interface (GUI). Based on BOOTP protocol and used in an Xstation environment, the DHCP server can assign Xstation terminal IP addresses and host name; it will be considered a permanent lease. The configuration process defines three major areas:

- Global resources for all networks
- Various resources for each network
- Resources for specific clients or sets of clients

Keys represent client information within the DHCP server. The first key is the client's *position* in the network, which links to the selected address pool for a candidate address. A client can be a member of a set machine that has similar features designated as a *class*. This class specification allows the server to provide extra options or features for a client.

Another key is the client ID, which can be a TCP/IP hostname or the Media Access Control (MAC) address. The server can use the client ID to provide personalized options. Based on these keys, the DHCP server can generate a reply per client request to serve the community of users. This enables the server to assign an address based on the network segment to which the user belongs.

If the class information is specified by the client, the default services for that class can be returned rapidly. For example, the accounting class can represent users in an accounting department, which needs access to a specific high-quality printer. Global resources provide overall consistency and baseline functions for each client. They are sent if no network, class, or client options override them. The hierarchy of options are client, class, network, and global options.

Figure 4 shows an example of the Ace Corporation, which has six IP networks. Network Information Center (NIC) has assigned a class C and a class B network address to Ace. The corporate users can be grouped into either a dynamic or static environment.

The class C network is used by a dynamic group of users in the marketing and sales organization. Each office within the building has an interface port and serves as a temporary mobile docking station for many sales personnel. In this example, a pool of dynamic users shares each office.

The class B network is used and shared by the remaining corporation in five static functional workgroups, including accounting, research and development, manufacturing, product testing, and the server network. Each of these five groups has a subnet of the class B network. Figure 4 shows the assignment of the network address and network mask.

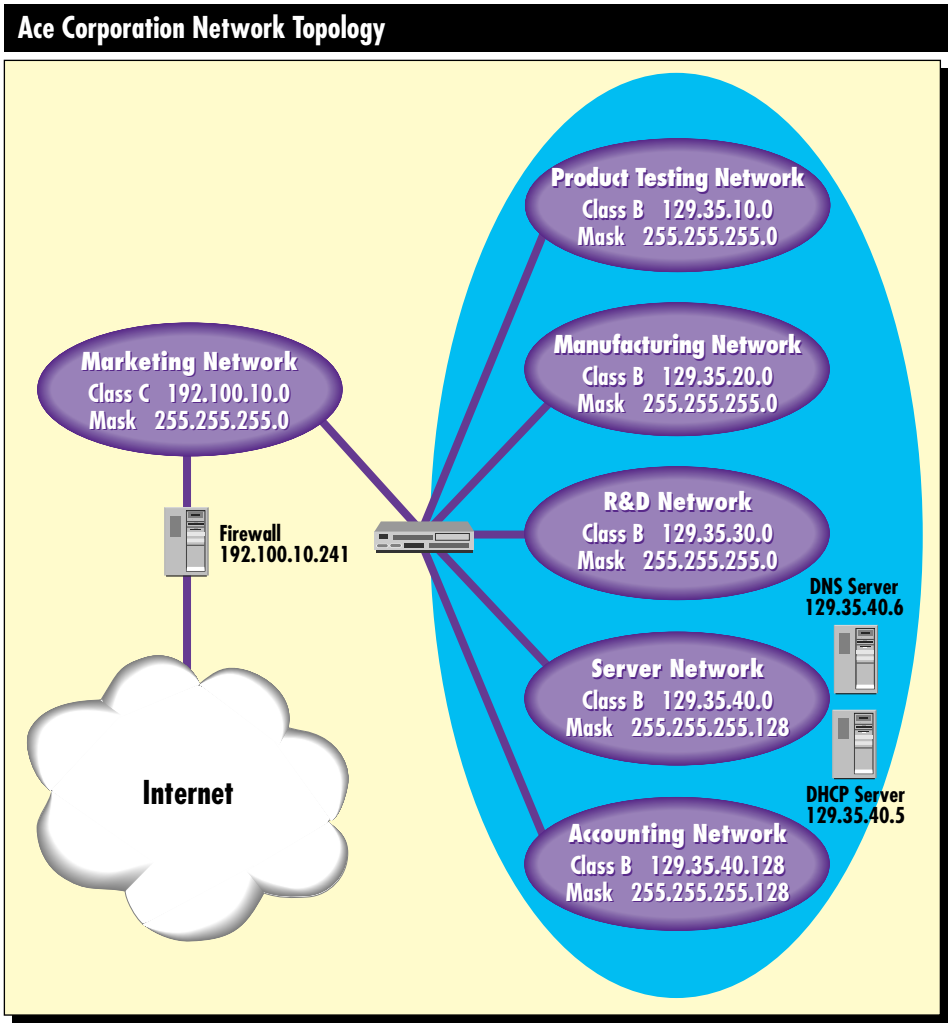


Figure 4. Network address and network mask assignment

```
network 192.100.10.0 192.100.10.2-192.100.10.240
{
   option 1    255.255.255.0            # Subnet mask for class C
   option 3    193.100.10.1             # Default gateway/router address
   option 6    129.35.40.5              # Default DNS Server
   option 51   2 hours                  # lease time of 2 hours since the sales
                                        # people are only around for an hour
   option 15   "marketing.ace.com"      # Default domain
}
```

Figure 5. Configuration for Class C marketing network

```
network 129.35.0.0 24                   # NIC assigned class B addr 129.25.0.0
                                        # 24 bit subnet mask is used
{
   option 1    255.255.255.0            # subnet mask for class B
   subnet      129.35.10.0              # subnetwork address
   {                                    # all addresses are assignable
      client 0 0 129.35.10.1            # address 129.35.10.1 is removed for
                                        # the router to use in Ace Corp.
      option 3   129.35.10.1            # default gateway/router address
      option 6   129.35.40.5            # default DNS server address
      option 15  "producttest.ace.com"  # default domain
   }
   subnet      129.35.20.0   129.35.20.2-129.35.20.200
   {                                    # subnetwork address with defined range
      option 3   129.35.20.1            # default gateway/router address
      option 6   129.35.40.5            # default DNS server address
      option 15  "manufacturing.ace.com" # default domain
   }
   subnet      129.35.30.0   129.35.30.2-129.35.30.215
   {                                    # subnetwork address with defined range
      option 3   129.35.30.1            # default gateway/router address
      option 6   129.35.40.5            # default DNS server address
      option 15  "R&D.ace.com"          # default domain
   }
}
```

Figure 6. Configuration definitions

The class B network is partitioned into four class C networks. Both the Server network and the accounting group can share half of class C because of the organization size.

Figure 5 shows the configuration definition for the class C marketing network.

The network statement has two parameters. The first parameter is network address; the second indicates the range of valid addresses that can be assigned. If the second parameter is not specified, this implies that all addresses are assignable. The range is from 2 to 240 because 1 is used by the router in Ace and should not be assigned; the address above 240 is reserved for future static users due to the business environment.

Figure 6 shows configuration definitions for three of the class B networks—Product Testing, Manufacturing, and Research and Development.

Figure 7 shows the definitions for the Server and Accounting networks.

Figure 8 shows those global parameters responsible for the overall operations.

The client statement provides options for a specific client, such as reserving an IP address or removing an address from the range. When using <string> in this statement, all client <string> must be unique. Figure 9 shows the syntax.

The result of these parameters can have different effects and can be best summarized as shown in Figure 10.

```
network 129.35.0.0 25                    # NIC assigned class B addr 129.35.0.0
                                         # 25 bit subnet mask is used because
                                         # the 256 addresses are shared by 2
                                         # subnetworks.
{
    option 1     255.255.255.128         # subnet mask for class B
    subnet       129.35.40.0    129.35.40.64-129.35.40.126
                                         # subnetwork address with defined range
    {
       option 3    129.35.40.1           # default gateway/router address
       option 6    129.35.40.5           # default DNS server address
       option 15   "netserver.ace.com"   # default domain
    }
    subnet       129.35.40.128           # subnetwork address
    {
       option 3    129.35.40.129         # default gateway/router address
       option 6    129.35.40.5           # default DNS server address
       option 15   "accounting.ace.com"  # default domain
       client 0 0  129.35.40.129         # client addr 129.35.40.129 is removed
       client 1 0x1005ACABADAE 129.35.40.130 # IP address 129.35.40.130 is reserved
                                             # for Ethernet @ 0x1005ACABADAE
    }
}
```

Figure 7. Definitions for Server and Accounting networks

```
supportunlistedClients       yes     # support all clients without
                                     #  explicitly listing in this file
supportBOOTP                 yes     # Ace has Xstation and network
printer
                                     #  that uses BOOTP protocol
leasetimedefault             5 days  # 5 days lease time for IP address
leaseexpireInterval          1 day   # time for the DHCP server to recover
                                     # lost IP address due to end user not exit
                                     #  normally when disconnect
```

Figure 8. Global parameters

```
client   <hardware type>   <hardware address>   <IP address>
         1=Ethernet        address <string>     <none>
         6=Token Ring                           <any>
         1=FDDI
         0=<string> on the next field
```

Figure 9. Syntax of client statement

Summary of Parameters

Hardware Type	Hardware Address	IP Address	Results
0	0	<IP address>	<IP address> will be removed from the pool
0	<string>	<IP address>	Client whose identifier matches this <string> will be assigned <IP address>; for example, <string> can be people's title and is used for assignment
<type>	<address>	<IP address>	Specific hardware type with the hardware address will be assigned the special IP address
<type>	<address>	none	Do not respond for this type with specified address
0	<string>	none	Do not respond whenever this string is specified
<type>	<address>	any	Give any IP address to this type with specified address. Used when supportunlistedclients=no
0	<string>	any	Give any IP address if this string is specified. Used when supportunlistedclients=no

Figure 10. Parameter results

```
smit tcpip -> Further Configuration -> Server Network Services ->
Other Available Services -> dhcpsd Subsystem
```

Figure 11. SMIT sequence for starting the DHCP server

Server configuration consists of setup and control; setup consists of initialization of all DHCP parameters for all potential clients. The configuration data is stored in the /etc/dhcpsd.cnf file. The auto start of the server is in /etc/rc.tcpip, or it can be started from the following <smit> sequence. Figure 11 shows the selection order.

The command to start the GUI server definition is <dhcpsconf>. See Figure 12.

The GUI interface can mask the syntax complexity of the configuration file and simplify the setup. The menu bar contains pull-downs that define the server default features, such as leased time duration, IP address cleanup, and so on. The panel has three main working areas:

- **Option List:** Descriptive selectable options that the server can support
- **Key List:** Entries to describe a client, including network position, class, and client ID
- **Main Window List:** Logical view of the options for each key

Configuring the DHCP Client

The goal of the DHCP client is full mobility without reconfiguration for each connection. This is done by defining a set of client preferences—the existing network and server environment requirements. The intent for the DHCP server is to provide a similar running environment in the destination network. The client setup is done using System Management Interface Tool (SMIT) panels.

The client broadcasts the preference information during the connection phase. One or more DHCP servers will examine the preference (preference list is optional for a client) and provide a counteroffer. The client will choose the intended server based on the best-matched algorithm. The client can specify only the types of information that belong to two major categories:

- TCP/IP network related, including static routes, NetBIOS name server, interface Maximum Transmission Unit (MTU) size, and so on
- Server services related, including Time Server, Print Server, DNS, NIS servers, and so on

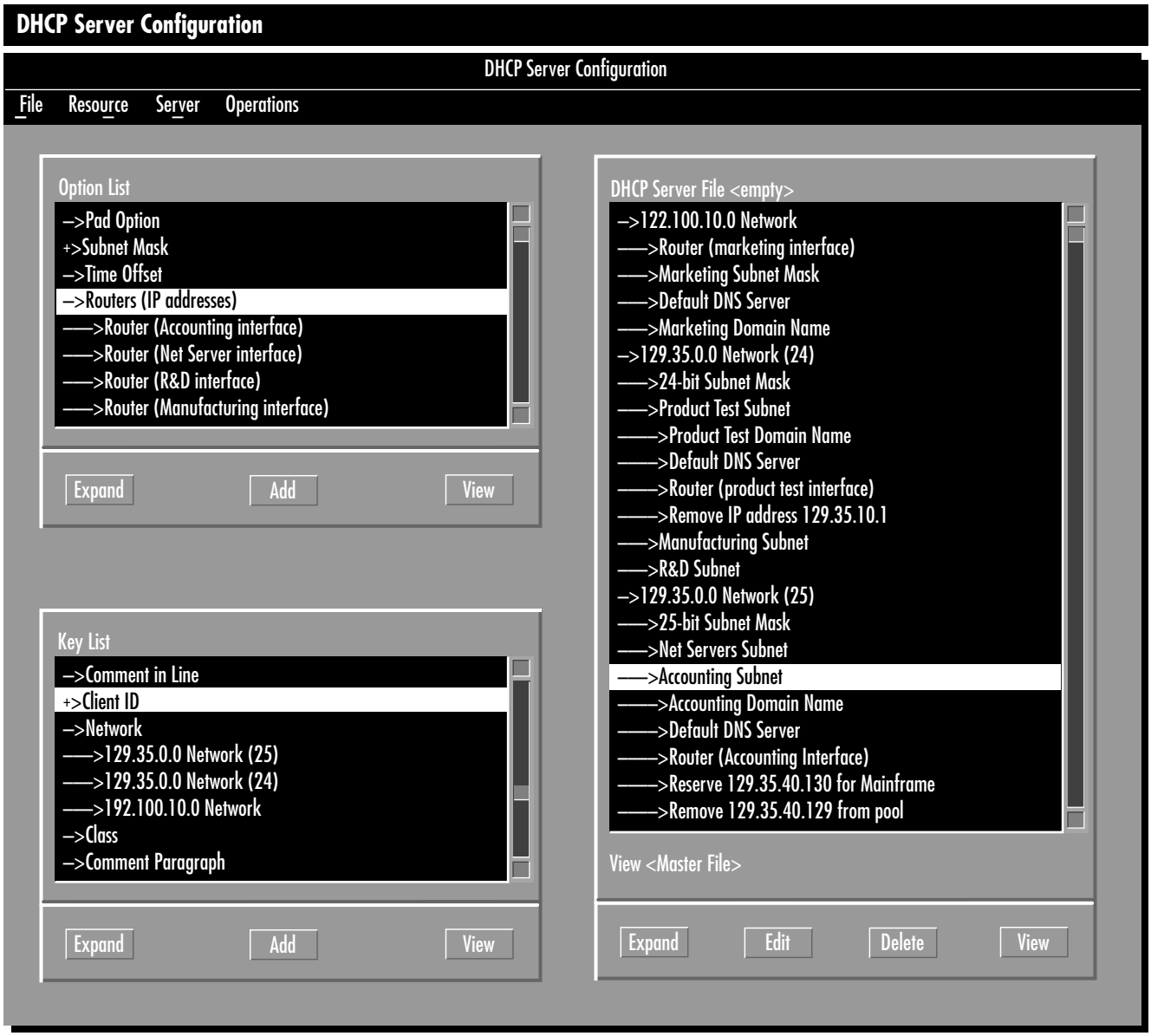


Figure 12. DHCP server configuration

The setup information is stored in two files:

`dhcpcd.ini`	DHCP configuration file consists of directives that can be specified for a client; the preferred option list
`/etc/rc.net`	Information for the startup interface

The SMIT configuration sequence for a client is as follows:

`smit tcpip`	`Use DHCP for TCP/IP Configuration & Startup`

SMIT can also be used for client daemon control using the sequence as follows:

```
smit tcpip → Further Configuration →
Server Network Services →
        Other Available Services →
dhcpcd Subsystem
```

BOOTP Relay Agent

A router usually does the relay when routing `BOOTP` broadcast from one subnet to another. AIX can also do TCP/IP routing with support for `BOOTP` forwarding. Using a method similar to the one used in the DHCP server, the `BOOTP` relay agent is started and stopped.

The configuration is done by adding a line to the `/etc/dhcprd.cnf` file. Multiple servers can be added to the configuration file. The relay agent will send the incoming packet to all servers that are defined in this file.

Storage Requirements

DHCP and DDNS server can be set up in a single server because of the traffic flow pattern and synergy in functions. Storage requirements for the server are dependent on the number of clients. Disk usage for each client is 360 bytes; therefore, a DHCP server with 10,000 addresses requires 3,600,000 bytes of storage space. Memory utilization is the same as storage since the entire database will be loaded due to performance considerations.

Conclusions

DHCP and DDNS functions can automate the labor-intensive process of address and resource control in a large network environment. This architecture is still emerging and extensions are needed to integrate with the Point-to-Point Protocol (PPP) for remote and mobile users. Also, server availability must be addressed in a 24x7 environment. Based on the current design point, whenever a DHCP server is unavailable, new DHCP clients cannot join the network and existing DHCP clients cannot renew their leases. Another area that needs focus is the DHCP database technology in a distributed environment, which can directly affect the server performance, redundancy, and management issues.

Eddie Ho, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Ho is a senior consulting marketing representative in the RS/6000 Executive Briefing Center. He has a BS in Computer Science from the University of Wisconsin and an MS in Computer Science from North Dakota State University.

Greg Althaus, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Althaus is a staff programmer in the AIX Open Systems Communication Development. He has a BS in Computer Science from the University of Texas in Austin.

Multimedia Server for AIX

By Eddie Ho, Sam Juliano, and Gary Linker

With the explosion of interest in the Internet and the ever-increasing capabilities of personal computers, streaming multimedia presentations over your corporate network is as easy as browsing through your stored electronic documents. Business and educational institutions are fervently seeking new ways to exploit the power and the impact of multimedia for their benefit. The IBM Multimedia Server for AIX can deliver audio or video information using your existing business infrastructure.

The Multimedia Server is designed to turn your RISC System/6000 (RS/6000) into a scalable media server that can deliver multiple, simultaneous streams of audio or video data. Running on AIX 4.1.4 and later, the Multimedia Server delivers uninterrupted multimedia data to PC, Macintosh®, and UNIX clients using the industry-standard Network File System (NFS) protocol. Client workstations use standard multimedia applications, such as the MediaPlayer in Windows 95, to play the multimedia data as if it were stored on the client's local disk. Multimedia Server can also deliver data through intranet browsers and Lotus Video Notes clients to add audio and video capabilities to the existing corporate information system.

Typical Applications for Multimedia

Applications for multimedia are limited only by the imagination. Multimedia can be used to extend and enhance business processes to make them more efficient and more competitive. Many current applications include training, education, entertainment, publishing, collaboration, and product demonstrations. Elements of these applications are pervasive across all industries and customer types. The following are a few examples:

- **Training:** Employees can receive important job training at their convenience and at their own pace. Self-paced instructional courses are a natural application for multimedia, permitting the trainee to view and review material until it is understood and the trainee feels ready to continue.

- **Education:** Educators can reach a broader audience in more places with distance learning using distributed data across multiple locations. High-quality multimedia communications make traditional lectures and laboratories more interesting—and more fun.

- **Public information:** Multimedia can be used to create virtual museums, where artifacts and information displayed at the museums are immediately accessible to multiple users.

- **Demonstrations:** The most effective way for a customer to understand the value and operation of a product is to see a demonstration of its use. Multimedia demos can be replayed with no loss of quality. Multimedia kiosks can make these accessible to the public in a 24x7 environment.

Figure 1 summarizes some context and data types.

Network Multimedia Challenges

Traditional data processing equipment and networks are designed to process and deliver a small amount of data. The nature of the data is

usually not timing dependent, and information quality would not be affected if a slight delay occurred during processing or retransmission. On the contrary, multimedia data is very sensitive to timing and cannot tolerate any jitter or latency; thus, it changes the foundation of the data processing principle. In general, the operating system requires major restructuring and cooperation for data delivery to achieve normal quality. System components that can affect multimedia quality are as follows:

- **Operating System (OS):** Multimedia data transmission must take precedence over OS-scheduled activity in order to maintain consistency when streaming data.
- **Filesystem and Storage:** Filesystem technology must be restructured to handle large data reads and read aheads for each stream to sustain the final quality.
- **Network Technology:** The traditional LAN is shared. It is ideal for a bursty commercial environment, which is tolerable since commercial data is random and small in quantity. To deliver massive amounts of timing-sensitive data in a similar environment and maintain acceptable quality, the number of clients in a shared segment must be controlled to achieve consistency. In LAN switching, where each client owns a segment, micro-segmentation is not required.
- **Client Decoding:** Error recovery is critical in a commercial environment, and retransmission is required. Multimedia data does not require retransmission when an error is detected. This places a heavy burden on client real-time decoding technology to avoid network congestion and retransmission.

The Multimedia Server for AIX, engineered to solve these challenges, can deliver multiple simultaneous streams in a LAN environment.

Multimedia Server Model

The multimedia streaming solution is a two-tiered client/server data model that consists of three major areas: server system, network, and client workstations.

Typical Multimedia Environments

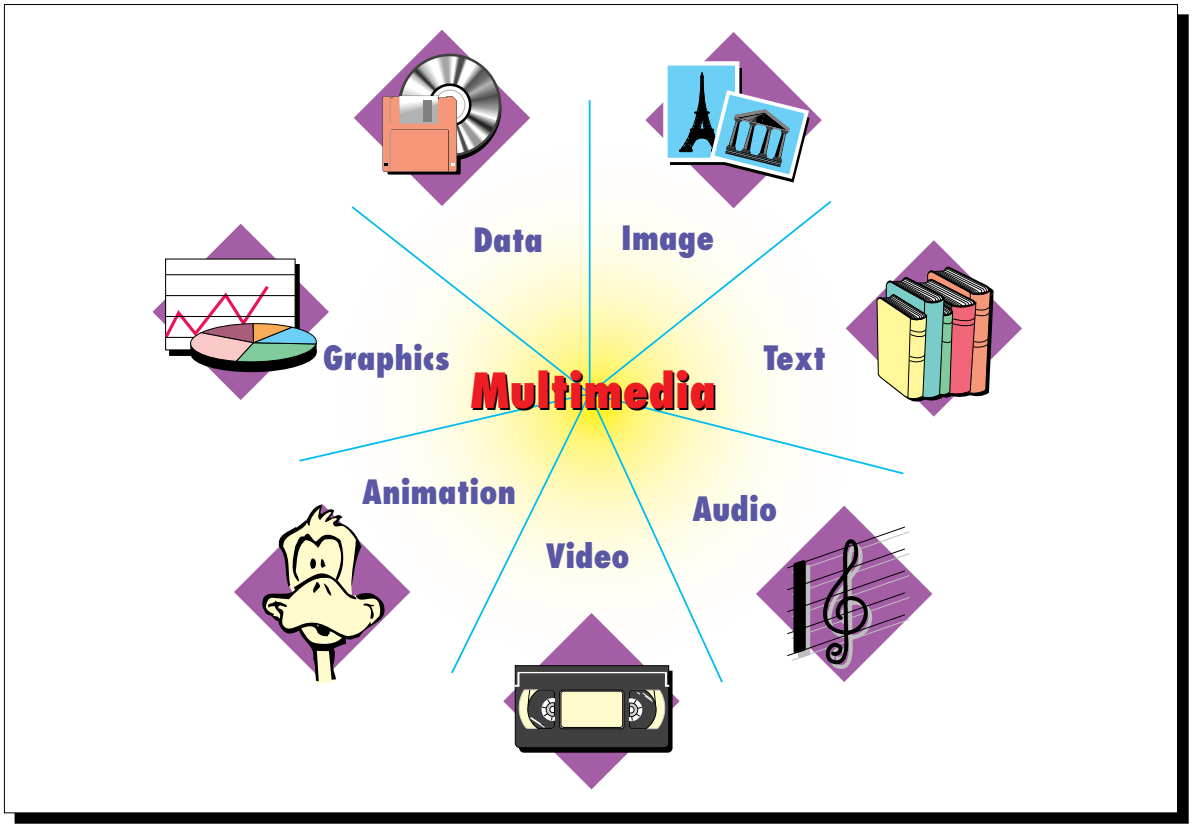


Figure 1. Multimedia content and data types

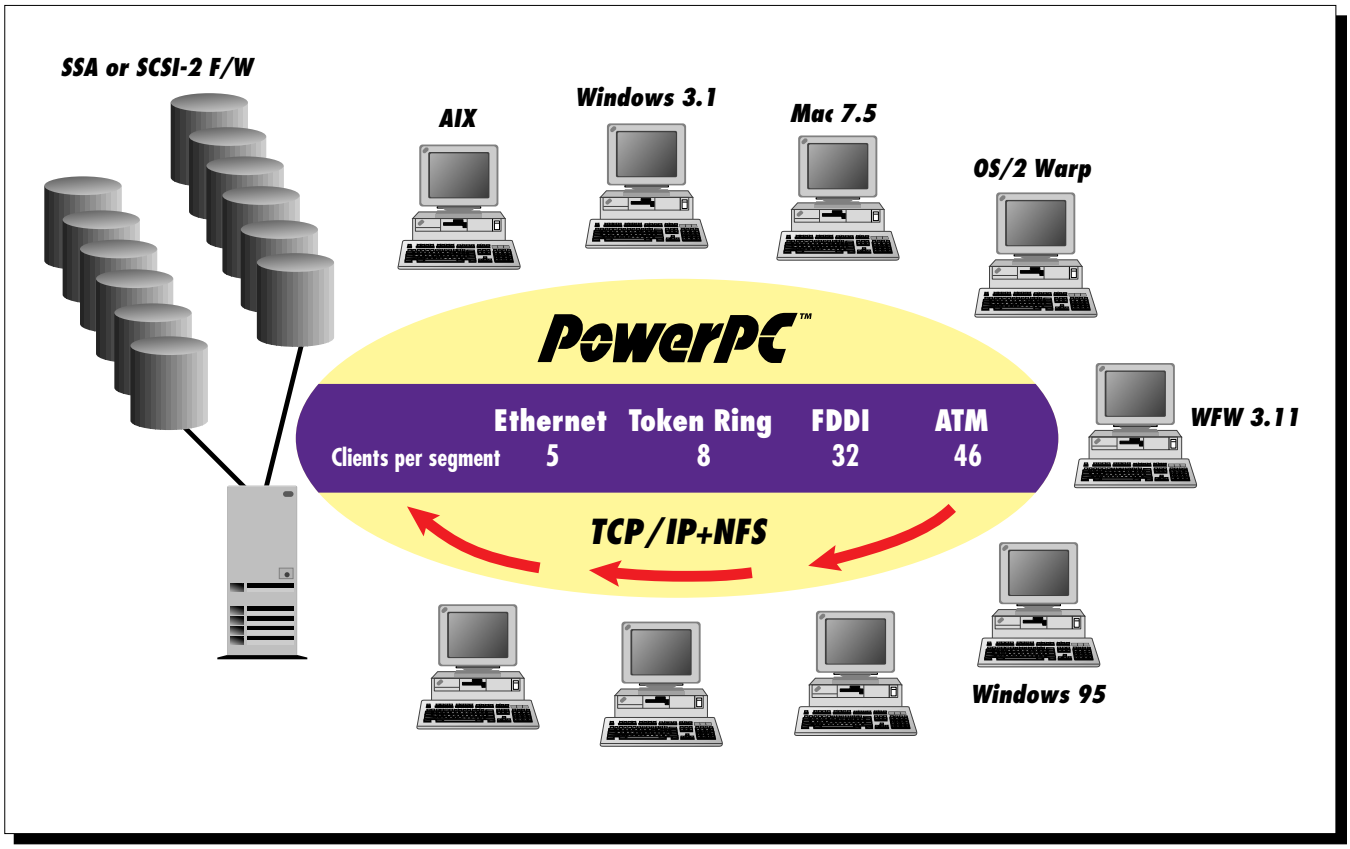


Figure 2. Multimedia server network environment

Server System

The server system is the repository and distribution center for multimedia assets. Figure 2 shows the server in a network environment. Digitized audio and video, in a variety of industry-standard formats (such as MPEG-1, Microsoft's AVI, and Apple® Computer's Quicktime), are stored in the server filesystem. These multimedia files are exported to client workstations using the TCP/IP NFS protocol. Multimedia Server is rated by the aggregate data bandwidth available from the server. The number of concurrent streams or clients that can be attached to the server can be derived from the aggregate data bandwidth and the bandwidth required by the application.

A video and audio application, such as a library of movies, requires much higher bandwidth than an audio-only application; thus, the number of streams provided by a server varies with the application and affects the number of concurrent users.

For example, a WAV audio-only data stream typically requires 384 Kbits/sec per client. In contrast, an MPEG-1 data stream, containing both video and audio, may require more than three times as much bandwidth, typically 1.2 Mbits/sec per client. Figure 3 shows samples of audio/video compression data rate.

Several RS/6000 servers, such as E20 to G40, have been tested as Multimedia Servers. Aggregate data bandwidths range from 25 Mbits/sec for the E20 to over 75 Mbits/sec for the G40. That ranges from 20 to more than 60 concurrent clients playing the MPEG-1 data type. For audio-only applications, a 75 Mbits/sec server can potentially serve over 190 clients. The server models and related streams are shown in Figure 4.

Network Assumptions

Networks play a major role in delivery infrastructure—it is the audio/video lifeline to the client. Industry-standard TCP/IP is the primary transport protocol with four different data link controls: Ethernet, Token Ring, Fiber-optic Distributed Data Interface (FDDI), and Asynchronous Transfer Mode (ATM).

Audio Quality and Data Rate

Quality	Sample Rate (KHz)	Bits/Sample	Mono/Stereo	Uncompressed Data Rate
Telephone	8	8	Mono	64 Kbit/sec
AM Radio	11.025	8	Mono	88.2 Kbit/sec
FM Radio	22.050	16	Stereo	705.6 Kbit/sec
CD	44.1	16	Stereo	1411.2 Kbit/sec

Video Compression/Decompression Standards

Standard	Quality	Compression Ratio	Compressed Data Rate
MPEG1	Med-High	35:1	1.2 Mbits/sec
MPEG2	High	40:1	1.65-60 Mbits/sec
MJPEG	High	20:1	3 Mbits/sec
H.261	Poor	200:1	50 Kbits/sec

Figure 3. Audio/video data compression rates

Traditionally, Local Area Networks (LANs) share bandwidth, which can affect the quality of video as more users are competing for the shared bandwidth. Most RS/6000 network testing for multimedia is done in a shared configuration except the ATM transport, which is based on a point-to-point dedicated environment. ATM provides the highest throughput to the client workstations, but even 10 Mbits/sec Ethernet can support up to five MPEG-1 clients using segmentation. Network segmentation controls the network collision domain to ensure a fair share of network resources for each client.

Figure 5 shows network types and their maximum number of clients.

LAN switching with a switching HUB can eliminate the number of clients-per-segment restriction in topology, but the maximum number of clients that each server can support remains the same.

Model	Data Throughput	Number of MPEG-1 Streams
E20/E30	25 Mbits/sec	20
59H/F20	50 Mbits/sec	42
G30/G40	75 Mbits/sec	62

Figure 4. Server models and related streams

Connectivity Type	Maximum Clients Per Segment	Topology
Ethernet	5	shared
Token Ring	8	shared
FDDI uplink with Ethernet hub	32	shared
ATM	46	point-to-point

Figure 5. Network type versus number of clients

Client Workstations

Client workstations can run the gamut from PCs to Macintoshes to UNIX workstations; in fact, any computer that can communicate using the NFS protocol and can play the desired multimedia data can be used as a client. IBM-compatible PCs from 486 DX2 to 133 MHz Pentium™ systems have been tested successfully with the Multimedia Server. The following operating systems have been tested by IBM: Windows 3.1, Windows 95, Windows NT, OS/2 2.1, OS/2 Warp, Mac System 7.5, and AIX 4.1.4/4.2.

Many recent PC models have a built-in MPEG-1 decoder; older PCs should use a decoder board for MPEG-1. Software decoders generally have poor video quality. In most cases, the client processor was overloaded performing NFS Remote Procedure Call (RPC) transactions;

therefore, software MPEG decoding can result in unacceptable skipped frames in video.

Multimedia Server Infrastructure

Multimedia assets such as audio and video are time-based. Continuous data streams of information must be delivered to the viewer at the proper rate in order to preserve human perception of motion and sound. The proper data rate, or bandwidth, must be maintained from the server to the client; otherwise the viewer will see "glitches" in the video and hear dropouts in the audio. Even the smallest imperfections in the video and audio data delivery are perceptible, which is unacceptable for quality multimedia delivery.

Multimedia assets are typically very large. A two-hour movie, digitized and compressed to a typical rate of 1.2 Mbit/sec (VHS videotape quality), results in a data file size of 1.2 GB. Most filesystems in a commercial data processing system are not optimized to handle many large files concurrently.

The Multimedia Server is optimized for the storage, management, and distribution of very large multimedia files.

The distinguishing feature of the Multimedia Server is the multimedia filesystem. Unlike standard UNIX filesystems designed for storing and retrieving relatively small files, the multimedia filesystem was designed from the ground up for the efficient delivery of multimedia assets, providing guaranteed bandwidth for data delivery. This file infrastructure is optimized for the storage, management, and distribution of very large multimedia files.

Some features of this file infrastructure include support for the following:

- **Data striping.** Striping spreads the workload evenly across multiple disks, which increases the aggregate bandwidth of the filesystem. It also permits concurrent reads that enable support for multiple clients.

- **Large files and filesystem.** Each filesystem can support up to 64,000 disks (up to 256 terabytes) architecturally.

- **Up to 1.3 terabyes of disk storage** and both SCSI and Serial Storage Architecture (SSA) technology are supported.

- **Large block reads**—256 KB and up. Reading large blocks of data reduces the number of read requests the server must handle.

- **Data replication.** Replication spreads multiple copies of multimedia files across filesystem disks to protect against disk failure. Replication factors can be assigned to the entire filesystem or to individual files.

- **Deadline scheduling.** This ensures multimedia data delivery takes priority over other OS services.

- **Error recovery and online reconfiguration.** Disks can be added and removed while the system is running with no data loss and no interruptions to the clients.

This filesystem is designed to be scalable and to provide continuous operation, so a minimal configuration can be expanded without the need to backup files, reformat disks, and recopy assets. Once the multimedia server is in production, it will never need to be shut down for maintenance.

Video Streaming over Intranet/Internet

Delivering video over the Internet or within an intranet is a complicated proposition. There are two approaches:

- Distribution of multimedia files as data that is played back later

- Streaming video, in which the client software decodes and plays the video on-the-fly

The Internet lacks the high-bandwidth carrier. Also any noise or delays while transmitting any of the data packets that make up a single frame of video can corrupt the entire stream.

This makes streaming video the preferred delivery method. Users do not have to wait to download the video clip, which is dependent on the workstation connection bandwidth. Although streaming video speeds the "time-to-gratification" for the user and creates a more interactive situation, it places greater demands on the underlying video technology. With streaming video, lost packets can result in lost video frames. And because video requires a predictable frame rate—30 frames per second is required for TV quality—keeping the frames flowing to the client at a continuous, predictable rate is important.

The Multimedia Server for AIX can be used to stream video in an intranet environment. Intranet Web pages can include links to

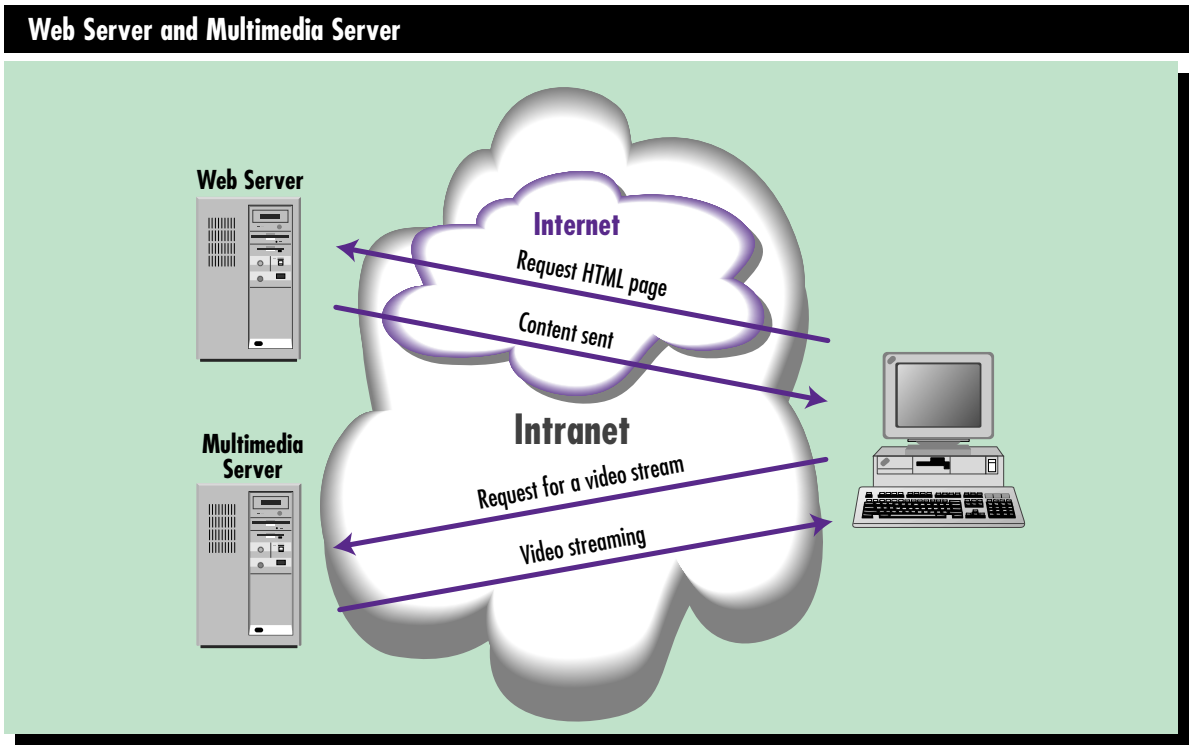


Figure 6. Web server and Multimedia server

multimedia assets resident on the Multimedia Server. When these hyperlink fields are selected, the assets begin playing immediately on the client workstation. In multimedia streaming, the communication between client and server is separate from the actual Web page transmission. The Web server transfers control to media players, which have their own links to the media server using either a local helper application or plug-in approach.

Figure 6 summarizes the client workstation relationship to the Web server and the Multimedia Server.

Conclusion

Multimedia is changing the way we work, allowing us to be more productive by bringing all data types to the desktop. The spiral effect of faster server systems and higher-speed information networks can bring end users closer to the real-time environment. The Multimedia Server technology provides a building block to launch the second-generation wide-area multimedia system with dynamic content delivery from anywhere using the nationwide ATM infrastructure.

Eddie Ho, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Ho is a senior consulting marketing representative in the RS/6000 Executive Briefing Center. He has a BS in Computer Science from the University of Wisconsin and an MS in Computer Science from North Dakota State University.

Sam Juliano, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Juliano is a staff programmer in Internet Multimedia Development. He has a BS in Electrical Engineering from the University of Texas in Austin.

Gary Linker, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Mr. Linker is a staff programmer in Internet Multimedia Development. He has a BS in Mathematical Sciences from the University of North Carolina in Chapel Hill.

Talking to AIX Support Line

by Georgia A. Gibson

IBM continues to look for ways to improve its business relationship with you. It takes a concentrated team effort to quickly handle your questions and problems. This article describes the role of AIX support personnel and provides some hints to help you receive the most efficient and qualified answers to your system problems.

Customer satisfaction is at the forefront of AIX Service and Support personnel goals. IBM regularly asks customers questions that measure their satisfaction with our service/support. These results are carefully studied and plans are put into place to provide solutions to customer concerns. Although customers have reported a more than 90% satisfaction rate, we want that number to be 100%.

Customer Concerns

Despite our efforts, some of you have three major areas of concern:

- Hold time or wait time is too long
- Callbacks from the support center are not as expected
- Skill level in the support center is not as expected

We have plans to address specific instances, but by understanding the roles and responsibilities of the support center staff, you will then realize what information they routinely need from you to expedite your call. In some cases, you may just need to know how to raise the flag when your call needs higher attention than it is receiving.

Two programs in the Support Center address skill level: the Technical Vitality program and the AIX certification program. The Technical Vitality program helps new members of the Support Center to gain the skills they need. The AIX certification program, administered by an independent firm, provides certification of AIX skills. Over 90% of the AIX Support Center staff has taken and passed the certification test.[1]

The People Behind the Phones

Some of these concerns can best be addressed by helping you understand who you are talking to when you call, and their primary roles and responsibilities.

Response Coordinator. When you call in, the first person you speak to is a Response Coordinator (RC), a Level 1 support person who directs your call to the appropriate Technical Specialist. The RC listens to the description of the problem you are experiencing; then using keywords from your description, searches an online database that pinpoints the routing of your call. The RC needs enough information to identify the pertinent criteria to create the initial problem report. The RC then transfers your call to the Technical Specialist.

Technical Specialist. This is a Level 2 person skilled in the technical aspects of your specific problem. This person will be able to help you identify an acceptable solution in over 90% of the problems. When the problem is resolved to your satisfaction, simply tell the Technical Specialist and the problem is closed.

Duty Manager. Customer satisfaction is the Duty Manager's main focus—this is your

Georgia A. Gibson

[1]To participate in this program, call Sylvan Prometric at 1-800-959-3926 to register for one of the tests.

customer advocate. The Duty Manager also assists in resolving customer complaints. The primary role is to provide an escalation/notification contact for customer-related services. The Duty Manager listens to your concerns, facilitates transfers of your call to the appropriate person, and documents the stalled or broken process or implementation that caused you to be dissatisfied with the service. This helps IBM understand what and when the processes fail, and provides the opportunity to change processes when appropriate.

Level 3. If your problem is unusually difficult to diagnose or to find a solution, it will be referred to a Level 3 person. You will work directly with a developer at a very detailed level to describe, re-create, diagnose, and find an acceptable solution. Sometimes the resolution may be opening a defect against the product. Their objective is to find or create an acceptable workaround to alleviate the problem.

Hints and Tips

Here are some hints and tips for you that will help us to provide better service when you call. Having certain types of information available when you call will expedite your call and help to get to the heart of the problem faster.

Level 1 Hints

Response Coordinators need the following information when you call:

- The operating system version; for example, AIX 4.1.4
- Your customer number (also be sure you are on the contact list)
- Machine and model type
- A realistic severity assessment

Severity assessments range from the following:

Severity 1: You cannot use your program or system, which has a critical impact on your work. This condition requires immediate solution.

Severity 2: You can use your program or system, but your operations are severely restricted by the problem.

Severity 3: You can use your program or system with some restrictions on the functions you can use. These restrictions, however, do not have a critical impact on your operations.

Severity 4: The problem you are experiencing has little or no impact to your operations, or a way has been found to circumvent the problem.

Describe briefly to the RC what you were doing when the problem occurred, such as installing the operating system or installing an LPP on the operating system.

Since RCs handle an average of over 20,000 calls per month, they are most anxious to process your calls efficiently. Their responsibility is two-fold. First, the AIX Support Line is a contract service, so their first priority is to make sure that if you have a contract, you receive the time that you have purchased. Second, they need enough information about your problem so they can route your call to the appropriate Technical Specialist.

Eighteen different specialty queues are used to route the calls. The description of the problem must include enough information to determine which of these queues is appropriate for your call.

These queues include the following:

- Commands
- Backup
- Install
- Graphics
- 24x7 (24-hour coverage)
- TCP/IP Communications
- Communications
- System Configuration
- Performance
- Async Printer
- SP2 (Parallel Processor)
- Kernel
- System Network Architecture (SNA)
- Distributed System Storage Products (DSSP)
- NetView/6000
- Database
- Languages
- High Availability Cluster Multiprocessing (HACMP) for AIX

Finally, if you are calling about a problem with a new product that is not listed on your contract, your call will be referred to marketing for registration.

From System Configuration

- Know your fix level.
- Know your machine and model number; for example, 7012 (machine type) and 390 (model number).
- Know the devices that are installed in your system and how they are connected. (You can List All Devices from SMIT.)
- Get the LED number sequence if your system crashes with an 888. To do this, press the Reset button and write down the number(s) until you cycle back to 888.

From Commands

- Verify the ownership and permissions of files.
- One of the most frequently asked questions is how to install manpages or InfoExplorer. The answer is to call 1-800-IBM-4FAX and ask for document #2443 (for AIX 3.2x) and #4619 (for AIX 4.1).

Figure 1. Specific hints from Technical Specialists

Level 2 Hints

Here are some general hints from Technical Specialists:

- Describe the exact command syntax and error message, including the 7-digit number, if available. Having the System Management Interface Tool (SMIT) error log available could be helpful.
- Have the system available when you call so you can easily run commands or check errors with the Technical Specialist.
- Try re-creating the problem.
- Be prepared to go through the error report during the call.

Figure 1 shows some specific hints from Technical Specialists.

Hints from a Team Lead and Duty Manager

If you are not satisfied with the results from the RC or Technical Specialist, a Team Lead or Duty Manager is always available.

The Team Lead ensures that your technical issues are being addressed appropriately. This is the first level of escalation if your call is not being handled to your satisfaction. If you are not satisfied by the Team Lead, your next option is to speak to the Duty Manager.

The Duty Manager specifically ensures that you are satisfied with the service. This is the next point of escalation if the problem is not being handled appropriately.

Level 3 Hints

If you can provide the information that is discussed above, we are ready to delve into the problem.

Georgia A. Gibson, IBM Corporation, 11400 Burnet Road, Austin, TX 78758. Internet: ggibson@ausvm1.vnet.ibm.com. A graduate of the University of Texas in Austin, Ms. Gibson is an AIX service planner in the AIX Service Development group within the RS/6000 Division. Her experience includes relationship management, hardware and software usability, as well as user interface design for the SMIT, Distributed SMIT, and VSM for AIX.

AIX Questions

Compiled by Bruce Pine

The AIX Solution Provider Technical Support Group in Austin, Texas, supports software vendors who are developing or 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.

When I start InfoExplorer, I get the following message:

Error: Unable to connect socket: 3

However, Info still comes up and appears to work fine.

This message appears when `infod` is not running. Since InfoExplorer is a large application, `infod` is used to limit users to run only one instance of InfoExplorer at any given time, thereby conserving system resources.

In some work environments, multiple workers will login using the same user ID so, if `infod` is running, only one user will actually see InfoExplorer. To work around this limitation, some administrators turn off `infod`.

—David Stewart

When I start InfoExplorer nothing happens—all I get is a prompt.

There are several possible causes:

1. It is possible that `infod` is running and your user ID is listed in the process table as already running Info Explorer. This process may be hung or it is being displayed on another workstation. Try killing the InfoExplorer process and starting it again.

2. Permissions may be a problem. Check permissions on `/tmp/.info-help`. It should read as follows:

```
srwxrwxrwx   1 root     system
```

If these permissions keep changing back to something else, then you are probably at AIX 3.2. The Programmed Temporary Fix (PTF) IX43230 addresses this.

—David Stewart

When I start InfoExplorer, a window is displayed that contains the following:

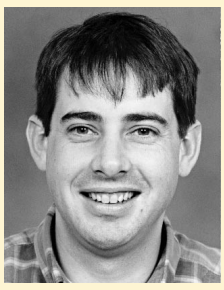
Operating System Error.
Error Number: 13

And InfoExporer fails to start.

When you start InfoExplorer, it attempts to read and write from/to `$HOME/info`. InfoExporer creates a directory when you start it for the first time. Chances are that you su'ed to some other user, probably root, and started InfoExporer for the first time.

If you did not use the `- <userid>` option with su, then `$HOME` is not changed and the InfoExplorer directory will be created with the incorrect ownership. Change `$HOME/info` ownership to the correct user and group as well as the additional files and directories below, including the "dot" files.

—David Stewart



David Stewart

Jeff Simon

When I start InfoExplorer, a window is displayed that contains:

Operating System Error.
Error Number: 28

If this error message appears, you will most likely find that your `/home` filesystem is full.

The fix is simple. Either remove unwanted files or make the filesystem larger. Use the command `chfs -a size=+1 <filesystem>` to increase the filesystem by 1 Physical Partition (PP) or 4 MB.

—David Stewart

Can the xlf Version 3 compiler accept C comment /* comment */" delimiters?

The xlf Version 3 compiler will accept C comment delimiters if you use the C preprocessor to preprocess your file. To call `cpp` for a particular file, use a file suffix of `.F`. Each `.F` file `<filename>.F` is preprocessed into an intermediate file `<F.filename.F>`, which is then compiled.

The intermediate file can be saved by specifying the `-d` compiler option; otherwise, the file is deleted. If you only want to preprocess and do not want to produce object or executable files, specify the `-qnoobject` option as well.

—Jeff Simon

The C library (libc.a) consists of many small modules, and one very large one called shr.o. What is this file?

The module `shr.o` is dynamically loadable and sharable by default and contains system calls and handlers.

—Jeff Simon

How can I call a C program from a FORTRAN program?

Figure 1 shows an example.

—Jeff Simon

How can I write to the text segment of an executable?

Writing to a text segment of an executable can be done by compiling the source code with `-qroconst`. Figure 2 shows an example.

```
c program name: FortranCallingC.f
      write (6,*) "Fortran is not just for mathematicians"
      call hello
      end
```

```
/* program name: Cprog.c */
void hello()
{
  printf("Most of UNIX is written in  C\n");
}
```

```
syntax of compilation:  xlc -c Cprog.c
                        xlf FortranCallingC.f Cprog.o -o <filename>
```

Figure 1. FORTRAN program calling C program

```
1. cc -qroconst -o <output> <filename>.c  // -qroconst will write to text
2. size -f <output>  –>>

   <output>: 380(.text) + 64(.data) + 16(.bss) + 302(.loader) = 762

3. cc -o <output> <filename>.c                // for comparison
4. size -f <output>  –>>

   data_out: 280(.text) + 164(.data) + 16(.bss) + 302(.loader) = 762
```

Figure 2. Writing to text segment of executable

Compiling with `-qroconst` will write to the text segment (the default is to write to the data segment). Note that pointers and complex aggregates containing pointer members cannot be placed in the text segment.

—Jeff Simon

I am sending as much information as possible through my Ethernet card. How can I improve its performance?

Most Ethernet cards today are based on the EISA bus standard, which can itself be a limiting factor on the speed at which your computer communicates on the network. Try an Ethernet card based on the PCI bus standard. This is the easiest way to improve performance. Another factor in network performance is the `mbuf` settings. A `netstat -m` command will show the status of the `mbufs` and if requests for `mbufs` are being denied.

—Craig Stermer

How can I tell if my system is using Domain Name Service (DNS)?

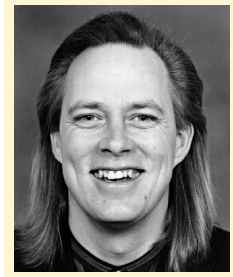
DNS looks for the existence of the `/etc/resolv.conf` file. If your machine has this file, it will use this file to resolve a name with a network address. To keep your system from using DNS, simply rename or remove this file.

—Craig Stermer

***Bruce Pine**, IBM Corporation, 11400 Burnet Road, Austin, TX 78758.*

AIX Receives UNIX 95 Brand

X/Open has stamped its UNIX 95 brand on IBM's AIX operating system. The brand signifies that AIX complies with the X/Open Single UNIX Specification and indicates that the operating system supports a common set of Application Programming Interfaces (APIs) that enable greater portability of applications between UNIX systems.



Craig Stermer

AIX Update Plans

Upcoming enhancements for AIX 4.2 as well as details for future versions of AIX have been released. Some new features for the current version that are scheduled for release in late 1996 and the first half of 1997 include the following:

- New version of AIX Connections that will enable an AIX server to act as a proxy for a NetWare server
- Licensing Novell Directory Services from Novell, Inc.
- Upgrade the Bonus Pack of Internet add-on products to include Netscape Communications Corporation's FastTrack
- Bonus Pack upgrade to include support for Netscape's Gold browser
- Electronic commerce applications and digital certificates for AIX 4.2
- Add-on products for World Wide Web and capacity management
- WebSMIT—Web interface to is System Management Interface Tool that can be used from a Web browser
- New multimedia products including a broadcast-level multimedia server that will stream audio and video data

A new version, AIX 4.3, is scheduled for third quarter of 1997. Its features will include the following:

- Full 64-bit version
- Full 64-bit file sizes, 64-bit memory and address space sizes, plus support for 64-bit processors
- Interoperability with 32-bit applications

Clustering with Phoenix Technology

IBM's Phoenix clustering technology will help developers build highly available mission-critical applications using clusters based on OS/2, Windows NT, AIX, or other UNIX operating systems. The technology provides much control in monitoring and responding to failure conditions.

Clustering provides higher availability by allowing systems to back up each other and allows additional systems to be added to the cluster as needed. Ensuring high availability has become an essential business requirement. IBM's Phoenix clustering technology delivers that by monitoring both hardware and software.

Phoenix technology is available for the RS/6000 SP servers and is planned for the rest of the RS/6000 family.

New Communications Server for AIX 4.2

The release of the new Communications Server for AIX provides a full range of communications and connectivity offerings for the RS/6000 and AIX. It supports both the uniprocessor and Symmetric Multiprocessing (SMP) hardware environments. It requires either AIX 4.1.2 and later or AIX 4.2.

The new release includes features of the existing Version 4.1 plus the following enhancements:

- **High Performance Routing (HPR):** An end-to-end connection-oriented protocol that improves network availability, performance, and System Network Architecture (SNA) congestion control
- **Dependent LU Requester (DLUR):** Provides a means of transporting LU traffic through Advanced Peer-to-Peer Networking® (APPN®)
- **AnyNet® Sockets over SNA Gateway:** Connects SNA and TCP/IP networks to allow sockets applications data to flow freely across both environments; workstations and hosts in SNA networks can run sockets applications and appear to be directly connected to the TCP/IP network
- **Performance enhancements:** Supports V.35 communication links up to 1.54 Mbit/sec for Synchronous Data Link Control (SDLC) and supports up to 50,000 concurrent sessions; maximum number of sessions will vary depending on the mix of session types and hardware configuration

The server will provide a powerful gateway along with a broad range of communication, connectivity, and networking options.