Patricia Seybold's

UNIX IN THE OFFICE

VOL. 5, NO. 2
ISSN: 0887-3054
FEBRUARY 1990

INSIDE

EDITORIAL

NEWS ANALYSIS

PRODUCTS • TRENDS • ISSUES • ANALYSIS

Multiprocessing

Towards Transparent Distributed Computing and Parallel Processing

By Judith S. Hurwitz

TO ACCOMMODATE the demanding applications being developed today requires more power than any single CPU can offer—at a price most users can afford. To provide this power without waiting for the next price/performance revolution, vendors are creating multiprocessor versions of existing systems to amplify their power by as much as tenfold. By creating multiple CPU versions of these systems, vendors can also leverage their existing software, a key factor in selling to power-hungry users. But multiprocessing is *(continued on page 3)*

• E D I T O R I A L •

OLTP Meets Unix

Transaction processing will key Unix's rise to commercial stardom.

By Judith S. Hurwitz

FOR THE PAST few years, we've been focusing our attention on commercial Unix and how it's about to take off. We keep waiting for the Unix desktop to take shape and for innovative software to flood the market. We were encouraged when Open Desktop was announced because of its potential to make the Unix desktop a reality. Everywhere, we look for signs to point out to those people who keep asking, "Is commercial Unix real?" Well, while it isn't time to declare all out victory, the way has been paved—via online transaction processing (OLTP).

Within a matter of weeks, the two leaders of fault-tolerant OLTP, Tandem and Stratus, announced Unix products. Likewise, Hewlett-Packard purchased 10 percent of Sequoia, a Unix-based fault-tolerant vendor, buying itself a product offering in this marketplace. None of these vendors is ready to proclaim OLTP and Unix the perfect match, but each is aware of the growing force of the Unix market. All of them see a potentially big opportunity.

OLTP is the meat and potatoes of computing. More than any other type of system, transaction processing runs businesses. It is the way companies keep track of orders and sales. It is about as commercial as you can get.

Why are these vendors suddenly flocking to Unix? It isn't really very surprising. Actually, it makes a lot of sense, considering the growing importance of standardization in operating system technology. The move to Unix is also directly related to developments in multiprocessing technology in general. Multiprocessor implementations of key processors are replacing traditional minicomputer and, in some cases, mainframe technology. Transaction processing and high-powered databases are key applications for hot systems. The center of multiprocessing activity is, in fact, the Unix operating system (for more information, see this month's feature report).

For years, the OLTP vendors resisted moving to Unix. Unix was too slow and generalized for the demanding requirements of OLTP. These vendors were able to sell systems based on individually crafted operating systems that were fine-tuned for workhorse OLTP applications. Vendors like Tandem flourished with relatively little competition.

But the market began to change as smaller vendors such as Pyramid, Sequent, and Sequoia demonstrated the degree of sophistication that can be achieved by tweaking Unix for high-performance environments. The user community became uncomfortable with the proprietary nature of OLTP and fault-tolerant operating systems. Customers began to ask, couldn't these systems run on Unix too?

Why did customers start asking about Unix if they were happy with the quality of the work provided by the proprietary OLTP vendors? Part of the answer has to be the psychology of the times. The computer industry is moving fast and furiously these days. No one is quite sure who will survive as the competition gets hotter. The applications developers working on technology for the '90s are all placing their bets on either Unix or, depending on the applications environment, OS/2.

So the good news is that OLTP is going Unix. This means that vendors will start adding value to the operating system to make sure it can meet the tough requirements of this critical applications environment. These vendors will put increasing pressure on groups like Unix International, the Open Software Foundation, and the various Posix committees so that their needs are met. OLTP was a key design criterion for the Mach kernel, thrusting it even further into the spotlight.

All of these developments mean that Unix and OLTP will become tightly related, and thus will become the focal point for OLTP applications in the '90s. ◐

Patricia Seybold's Office Computing Group

Publisher PATRICIA B. SEYBOLD
Editor-in-Chief JUDITH S. HURWITZ

Managing Editor
JOHN R. RYMER
Senior Editors
JUDITH R. DAVIS
Telephone: (617) 861-3926
RONNI T. MARSHAK
MICHAEL D. MILLIKIN
Associate Editor
LAURE BROWN
News Editor
DAVID S. MARSHAK
Sales Director
RICHARD ALLSBROOK JR.
Circulation Manager
DEBORAH A. HAY
Customer Service Manager
MICHELLE MANN

148 State Street, Suite 612, Boston, Massachusetts 02109 Telephone: (617) 742-5200 FAX: (617) 742-1028
MCI: psocg / 312 2583 Internet: psocg@dcm1sg.das.net TELEX: 6503122583

Patricia Seybold's UNIX in the Office (ISSN 0887-3054) is published monthly for $495 (US), $507 (Canada), and $519 (Foreign) per year by Patricia Seybold's Office Computing Group, 148 State Street, Suite 612, Boston, MA 02109. Second-class postage permit at Boston, MA and additional mailing offices. POSTMASTER: Send address changes to *Patricia Seybold's UNIX in the Office*, 148 State Street, Suite 612, Boston, MA 02109.

•MULTIPROCESSING•

(continued from page 1) not an end in itself. In multiprocessor implementations, vendors typically put many CPUs into a single chassis to increase power. The next step along the road to leveraging the power of systems is parallel processing, in which heterogeneous systems of different power and features can be linked together to leverage their aggregate power.

Before users are able to leverage all the compute power in their systems, a lot of work remains to be done. While the list is long, one of the first goals will be agreement on standards in areas such as operating system kernels, thread libraries, and application programming interfaces (APIs). Other areas that researchers will be working on over time include:

- Parallelization of hardware (beginning with multiprocessing configurations).
- Development of software allowing existing and new applications to be broken into component parts to operate in a parallel configuration and the software that coordinates parallel processes—if they exist on separate processors.
- Development of parallel language compilers.
- Development of parallel debugging tools.
- Development of synchronization and security tools for distributed environments.
- Development of tools that will allow users to restructure existing applications to work efficiently in a distributed environment leveraging multiprocessing and parallel processing.
- Addition of multitasking and multithreading to the operating system kernel to facilitate parallel processing.

No vendor can afford to wait for a more powerful generation of processors. Powerful multiprocessors are required to keep pace with evolving technology.

FIRST STEPS TOWARD PARALLEL PROCESSING. To achieve parallel processing is a complex task because of the number of tools and technology that will have to be developed to allow users to create, modify, and distribute applications in this fashion. The first step toward this goal is the development of multiprocessing technology, the transformation from systems based on centralized minicomputers to systems created by tightly combining microprocessors. These new multiprocessor systems will become the key applications platform for the '90s.

While mainframe systems have long employed multiprocessing technology, it is new to the microcomputer server arena. But it is a natural progression for the new generation of processors like the Intel 386 and 486 (which already employ multiprocessor concepts) and the Motorola 68000.

Unix hardware vendors are at the forefront of the move to multiprocessor implementations. This is not surprising given the evolution of Unix hot boxes. Unix-oriented hardware has become widespread in certain markets because the price/performance was dramatically superior to any proprietary system, partially because of the advent of commodity microprocessors.

WHY MULTIPROCESSING? While vendors are introducing multi-CPU systems to increase their competitive edge, the real beneficiary is the user. The user can continue to use the same applications but increase the power—sometimes as much as tenfold (or more) over single CPU systems. Ironically, multiprocessor systems accomplish the same thing that traditional minicomputers used as their major selling feature: the ability to leverage existing applications because each larger CPU offering was binary- or source-code compatible. Multiprocessing takes this concept a step further by allowing users to leverage their existing hardware in addition to the software investment.

The current generation of processors has been around long enough to generate a healthy portfolio of applications software, but it is running out of power. This, in fact, is one of the key factors behind the troubled minicomputer market. Users, much to their dismay, have been forced to continually swap for larger and larger processors to keep pace with capacity demands. By the time users had run out of steam on their existing processor, the market value of the technology had dropped considerably. The incremental cost of moving to the next-size box would be much more than the user would want to spend. Many users began to move to PC LANs to avoid having to buy more expensive central processors. It is not surprising that vendors would see the wisdom of multiplying the power of their systems by adding processors. Multiprocessing will spell the end of CPU-swapping as a way of increasing power.

In the current applications environment, no vendor can afford to wait for a new, more powerful generation of processors. Powerful multiprocessors are required to keep pace with the complexity of evolving technology, such as graphical user interfaces and multimedia applications. Even if a vastly superior new processor suddenly hit the market, it would take years before a large number of applications would be made available.

WHY UNIX? The fact that the operating system called Unix is at the center of this technological development is happenstance. Unix itself is not designed for use in a multiprocessing environment, but must be changed dramatically if it is to become the linchpin of a multiprocessing future. Unix needs a smaller, multithreaded kernel and a more modular design.

Three organizations concerned with the future of Unix are spearheading its evolution to support multiprocessing.

- The IEEE Posix group set up a new committee to come up with a portability standard for multiprocessing. The Posix Real-Time Committee has already come up with a proposed standard for threads.

- A Unix International multiprocessing task force recently completed a report recommending a new kernel for Unix System V to support multiple threads.

- The Open Software Foundation (OSF) has a Special Interest Group on multiprocessing, whose chairperson participated in the Unix International task force. Even before that work was completed, OSF's members pressured the organization to change from the IBM AIX kernel to a Mach multithreaded kernel because of their multiprocessing requirements.

THE PARALLEL PROCESSING CONNECTION. Once the Unix operating system supports multiple threads, it can be used as the pathway to parallel processing in the '90s. Today, it is logical to think of parallel processing as a requirement only in intense number-crunching applications. But commercial applications in the future, such as those that combine different information types in a windowed environment, will require as much parallelization of tasks as scientific applications do today. As application needs evolve, multiprocessing will become synonymous with parallel processing.

What Is Multiprocessing?

CONNECTING PROCESSORS. Vendors have been coupling processors for at least a decade. Many Unix and non-Unix vendors have found ways to loosely couple and cluster processors to increase throughput and power. An example of clustering is Digital's VAXclusters. They help users with multiple VAXs leverage existing systems. But VAXclusters do not provide the shared memory of a more sophisticated multiprocessor system. Also, they require one system to control and manage the rest of the systems in the cluster. Therefore, VAXclusters do not leverage all the power of the combined systems.

The most sophisticated type of multiprocessing assumes that a group of processors will act as though they are one machine. The more smoothly these parts cooperate, the better the multiprocessing.

The most sophisticated type of multiprocessing assumes that a group of processors will act as though they are one machine, sharing memory, bus, I/O, and the like. The more seamlessly these component parts work together, the better the quality of the multiprocessing.

Not all types of multiprocessing currently available are seamless. Various implementations provide bits and pieces of multiprocessing technology. The two predominant types of multiprocessing today are asymmetric and symmetric.

ASYMMETRIC MULTIPROCESSING. In asymmetric multiprocessing, one CPU controls the actions of the others. Therefore, the attached processors are not equal. In many asymmetric implementations, only one CPU can send information to the system bus. Implementing asymmetric multiprocessing does not require any changes to the operating system. These implementations have some advantages. They provide the price/performance benefits of allowing processors to be linked. While they don't double the power of two processors, they do provide some significant gains. And, because less time has to be spent modifying operating system software and optimizing I/O, implementing asymmetric multiprocessing is less expensive.

The disadvantages of these implementations are a driving force to the research and development into full symmetric multiprocessing. A key disadvantage is that control by one processor of all the others creates a serious bottleneck. And, while users gain some additional processing power, asymmetric multiprocessing by no means leverages all the power of the linked processors. Another disadvantage is that, in asymmetric coupling, the processors each tend to retain private memory. They do not, therefore, share memory. Asymmetric multiprocessing is best suited to number-crunching, where raw processing power is the key objective and there is less need to share memory and take advantage of a single system image.

SYMMETRIC MULTIPROCESSING. In symmetric multiprocessing, all processors are treated as peers with equal access to system resources and memory. Multiprocessor systems place a premium on the ability to decompose an application program into many parts. To facilitate this, an operating system must be designed to support simple "lightweight" processes that fully share memory and provide pointers to data at the operating system level. If the system includes sophisticated pointers, it does not matter where something is physically stored. Memory of all CPUs can be shared. Integrated memory management keeps track of memory on all systems. Multiprocessing systems need to be able to pass messages from any part of a program running on one process to another.

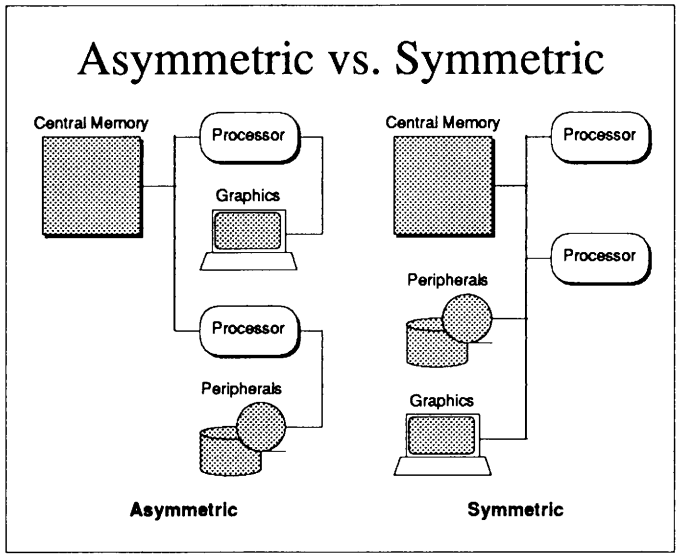
MASTER/SLAVE SYMMETRIC. Within the overall category of symmetric multiprocessing, there are different implementations. The most common in use today is *master/slave* symmetric, which includes systems marketed by Acer, Corollary (which developed systems for Compaq, SCO, and Wang), and Solbourne. The difference between master/slave and full symmetric is related to the way tasks are scheduled. To understand the difference, imagine that tasks are handled within a group of processors the same way customers are processed in a bank line. At some banks, the customers form one long line and

Definitions

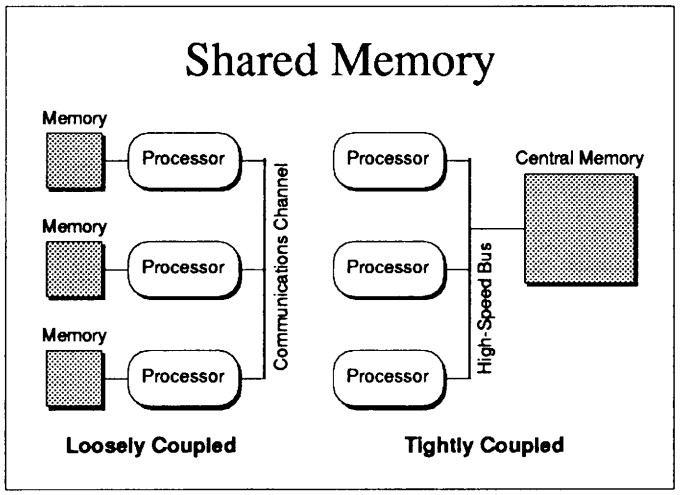
AN UNDERSTANDING of what multiprocessing is all about requires a few definitions.

Asymmetric Multiprocessing. One processor controls access to the others.

Symmetric Multiprocessing. All processors are peers, and, therefore, no one processor controls the others. All CPUs have the same computing capacity.



Shared Memory. In the most sophisticated type of multiprocessing, the memory of all the connected processors is shared and appears to be one large system.

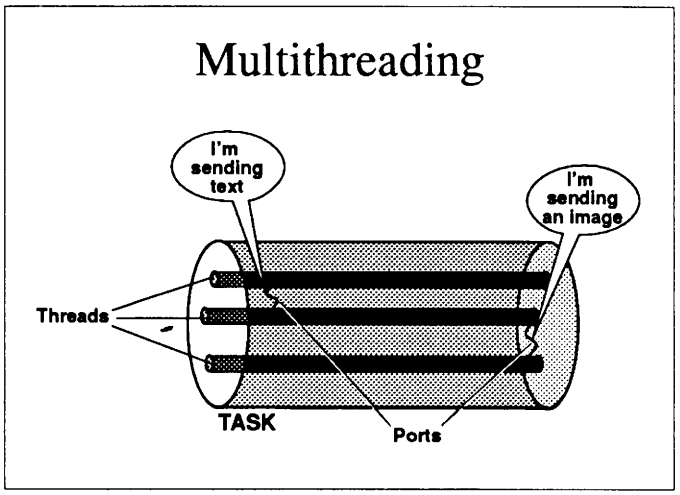


Scheduling. Scheduling is important to the efficiency of multiprocessing and is directly related to I/O. How efficiently a system handles or schedules the inputs and outputs from memory determines processing speed.

Multitasking. In order to have multiple processes happen at the same time, multitasking is required. An operating system that supports the use of threads makes multitasking easier to accomplish. Several processes can be bundled together as a single process and thus be processed simultaneously. This is critical both for multiprocessing and for parallel processing.

Parallel Processing. Parallel processing is the most advanced state of multiprocessing. Where current multiprocessor implementation provides all the CPUs in a single physical box, a parallel implementation will allow for tight coupling of networked systems. Likewise, all tasks, programs, and the operating system are spread throughout the system. Many processes can happen in overlapping intervals.

Multithreading. A thread is a process (a program, system call, or procedure call). In a single-threaded operating system like Unix, only one process can be handled at a time. In a multithreaded operating system, many threads can be created and processed simultaneously. Related threads are enclosed in a *task*. Threads can communicate information to each other through an Interprocess Communication (IPC) mechanism. If an operating system supports multithreading, then several related events can happen at the same time. Therefore, a series of steps can be broken down into small components (threads), allowing for greater degrees of control and efficiency.



are allowed, in turn, to go to the next available teller. At other banks, customers choose their own lines (I always choose the wrong one). The first method has an obvious advantage: With one long line, customers are immediately dispatched to the next available teller. This is the way full symmetric multiprocessing handles the scheduling of tasks. In the second scenario, where customers choose among multiple lines, they may suffer delays while they (or a bank officer) search for a shorter line. That scenario typifies the master/slave style of multiprocessing, where one processor assigns tasks to the other processors.

The master/slave arrangement is the simplest and most straightforward method of implementing symmetric multiprocessing. It is much more complex for vendors to implement full symmetry, which allows all processors to become peers. Therefore, many vendors starting out with multiprocessing begin with the master/slave implementation. In essence, a master/slave implementation is somewhere between asymmetric and full symmetric multiprocessing.

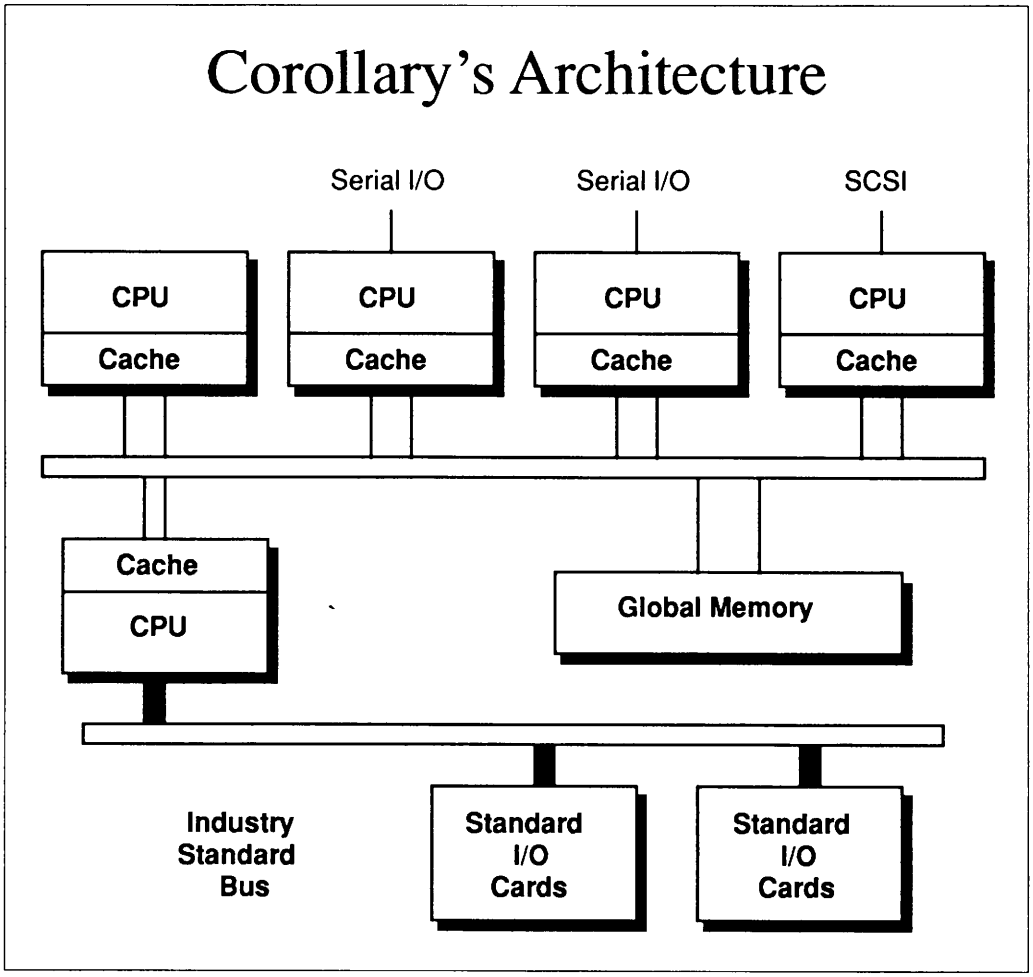
As vendors make the transition from single processing systems to multiprocessing, it is to their benefit to find ways to leverage existing hardware and software. Master/slave is a pragmatic approach used by some vendors in the workstation and smaller systems end of the spectrum. Vendors that specialize in high volume database and online transaction processing (OLTP) applications find they need to bypass this phase and move directly to full symmetric multiprocessing.

Certain implementations of multiprocessing use a combination of asymmetric and symmetric multiprocessing characteristics. A good example of this is Corollary's implementation.

COROLLARY. In adapting SCO's Unix System V/386 Release 3.2 for multiprocessing, Corollary added code to the SCO kernel that allows the operating system to recognize and use every processor in the computer. It also creates a shared memory system. Corollary has departed from a pure symmetric multiprocessing system to leverage existing software. The software provides two ways for device drivers to access the I/O bus. If a device driver has been written for a nonmultiprocessing system, it will default to a designated processor. Drivers created for the multiprocessing environment can run on any processor. This was done so that users can still use existing 80386 drivers. Likewise, this strategy allows applications written to other versions of SCO Unix System V/386 to run even in the multiprocessor implementation. This is accomplished by allowing only specially marked "multithreaded" modules to be executed on every processor. Anything else resorts to the default processor. Therefore, this is a dual implementation: part uniprocessor, part multiprocessor. It is a good, pragmatic interim move.

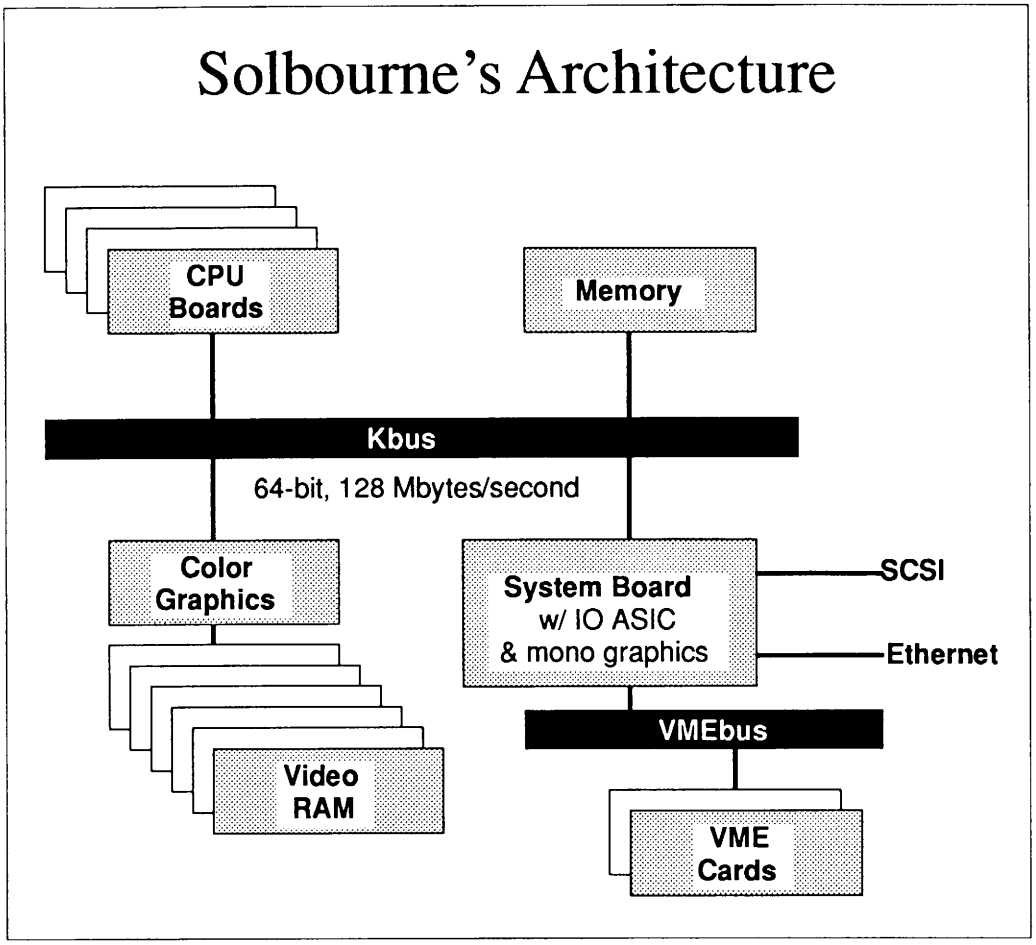
SCO needed such an implementation both to bring next generation technology to its third-party market and to prepare for Open Desktop, a workstation-based distributed applications model based on System V/386 and distributed networking (Motif, NSF, X-Window, etc.). Corollary's multiprocessing operating system allows applications to be spread across a distributed network.

Corollary also sells its own multiprocessor subsystem, which relies on additional hardware to enhance performance. For example, in the company's 486/smp system, designers have implemented some features of asymmetric multiprocessing



Corollary's multiprocessing architecture allows systems to leverage existing drivers.

to allow older technology and instruction sets such as PC ATs to participate. Corollary's implementation also uses the master/slave methodology of scheduling tasks. To compensate for the inherent slowness of I/O, Corollary added a proprietary high-speed, 64 MBps, 32-bit bus, in addition to the AT bus, to handle peripherals. Other features added to enhance speed include a 256K write-back cache and four specialized I/O ports that are connected to an eight-port terminal concentrator.



Solbourne's approach to multiprocessing is based on SunOS.

ACER. Acer has taken a different approach to master/slave multiprocessing. Acer has targeted IBM's Micro-Channel Architecture (MCA) systems for its implementation of multiprocessing. Rather than add a proprietary system bus, Acer lets each processor maintain its own private memory with a proprietary processor bus. Whenever a new process is created, it is assigned to one of the processors and installed in the local memory of that dedicated CPU. Only when a process has been modified does one CPU transfer the changes to another CPU to ensure synchronization. This approach is intended to prevent excessive use of system-level bandwidth.

SOLBOURNE. Solbourne's master/slave implementation takes a third approach by dividing the SunOS processing queue into two parts. In SunOS, the operating system has a single queue for all processes in the system. When the operating system finishes one process, it checks for the process with the highest priority and runs that. Solbourne's adaptation divides the queue into two pieces: a master queue for operations that have to run on the master processor and a second queue for processes that can run on either the master or the slave. This allows a certain percentage of interrupts to be handled without interrupting processing. This master/slave type of symmetric multiprocessing is best suited to an application that is not I/O intensive, such as number crunching.

FULL SYMMETRIC MULTIPROCESSING. Vendors adopting a full symmetric approach are looking for maximum speed and power. The leaders in fully symmetric multiprocessing include Digital Equipment (under VMS only), Sequent, Pyramid, Arix, Data General, and, most recently, Hewlett-Packard. While each company has engineered its approach with slightly different twists, all have one thing in common: They have had to make extensive changes to the Unix kernel to achieve their goals. The fact that these companies are, in essence, producing proprietary extensions to operating systems to accomplish this generation of technology is proof that standardization is needed.

As we mentioned earlier, full symmetric multiprocessing means that all CPUs are equal. However, within the definition of full symmetric, different companies take different approaches. For example, some vendors choose to multiprocess only certain tasks, such as the most frequently used instructions. All vendors add accelerators—either extra I/O processors or caches to make these systems faster. The following are examples of the ways in which Pyramid, Sequent, Data General, Arix, and Hewlett-Packard are approaching full symmetric multiprocessing.

PYRAMID. Pyramid was one of the earliest Unix vendors to implement full symmetric multiprocessing. Like its competitors, Pyramid has found it necessary to rewrite the Unix kernel. To improve system throughput, Pyramid's operating system, OSx 5.0, employs a scheduling technique that reduces cache invalidations and bus-loading.

Pyramid is a bit of a maverick in the multiprocessing arena. The company's researchers, unlike most of its competitors, have come up with their own methods of load-balancing. Also, Pyramid's researchers believe that adding threading to the kernel is detrimental to overall performance.

Pyramid's new scheduling mechanism, called *affinity*, does not use traditional load-balancing techniques to schedule tasks onto different processors. Pyramid's researchers criticize the way multiprocessing vendors handle scheduling. By fixing nondegradable scheduling priorities, Pyramid's developers believe they have discovered a better way to maximize the state of the cache memory. Pyramid has also developed a virtual memory management subsystem that its researchers believe results in more efficient paging and requires less swap space.

Pyramid's MIServer series allows up to 12 processors to be linked. The linked processors are all equal, with each new task sent to the next available processor. To enhance performance, Pyramid has added intelligent terminal and I/O processors. The MIServer series supports up to 16 I/O channels with throughput of 11 MBps per channel. Another I/O enhancement facility consists of a three-bus architecture including two 80 MBps buses. Each CPU has a 265 KB cache and memory-request queueing to help efficient bus utilization, which is important to multiprocessor implementations.

Pyramid and Sequent have complementary approaches to multiprocessing. Both are aiming at high volume, high throughput OLTP applications.

SEQUENT. Pyramid and Sequent have complementary approaches to multiprocessing. Both are aiming at high volume, high throughput OLTP applications. Because Unix was not written for high performance applications like OLTP, vendors using Unix as their foundation have had to fine-tune hardware and software. Like Pyramid's implementation, Sequent's is fully symmetric and implements a shared memory system. Sequent also puts caches in front of every processor and adds special silicon to manage the bus. Sequent spent the first two years of its existence completely revamping the Unix kernel. In addition, it has done considerable work on optimizing compilers and on the parallelization of tasks.

DATA GENERAL. When Data General (DG) did an about-face and aggressively pursued a Unix strategy, it developed a version of Unix designed to support fully symmetric multiprocessing. Like other vendors, Data General's engineers found they had to restructure the Unix kernel. In DG's implementation of multiprocessing, each processor schedules its own tasks via a global queue. In essence, DG has created a virtual processor architecture, a layer of software above the physical processors that controls all I/O. The advantage of this implementation is that an I/O completion interrupt can be processed by any processor, not just the one that initiated the I/O. In addition, to make scheduling more flexible, DG has adopted a two-layer scheduler that determines how processes gain access to physical resources. A short-term scheduler multiplexes virtual processes onto the system's physical processors, while a medium-term scheduler sets scheduling policies and actually schedules user processes to run virtual processors.

ARIX. Like Sequent and Pyramid, Arix has rewritten the Unix kernel to implement multiprocessing. A key difference is that Arix has put 40 percent of its multiprocessing Unix kernel into its dedicated I/O processors. While the system thinks it is dealing with one copy of the kernel, a lot of processing is handled by front-end 68030 processors. Arix has also implemented a proprietary 64-bit bus with a speed of 128 MBps. To add to I/O speed, Arix has added a dedicated processor to the bus.

HEWLETT-PACKARD. Hewlett-Packard is the most recent vendor to jump into the multiprocessing fray. Its implementation is on the newly announced members of its 9000 family, specifically the 870 models (the Model 870S/200 provides two-way multiprocessing with a rating of 95 MIPS, while a future model, the 870S/400, will provide four-way multiprocessing). These products will ship by the fourth quarter of 1990. Why has HP has stopped at four processors while vendors like Pyramid have been able to connect up to 12? HP's designers set a goal of achieving 150 MIPS in their multiprocessor configuration. Therefore, when they designed the hardware chassis, they implemented it so that there would be four slots. Designers perceive this four-way multiprocessing as the most efficient configuration, and they claim that Pyramid does not get the full benefit of all the attached processors.

Like its competitors, HP has done considerable work to the Unix kernel to allow it to support multiprocessing. For example, HP has added threads to its kernel. HP intends to upgrade its kernel to OSF/2 with its multithreaded kernel.

HP's developers designed the RISC Precision Architecture to support multiprocessing. Therefore, the move to a multiprocessing implementation was considered minor surgery. For example, caches were already maintained in hardware. These data caches are kept synchronized by the hardware, rather than through the operating system. HP has also done considerable work in the interrupt subsystem in order to allow multiple events to be handled simultaneously. In the case of I/O drivers, HP provides uniprocessor emulation so that users will not have to rewrite an existing I/O driver to work with multiprocessing. Corollary handles I/O drivers in a similar manner.

Mach

An Example of a Modern Operating System Kernel

MACH IS AN operating system developed at Carnegie Mellon University with funding from DARPA and IBM. Mach is a modern, minimal kernel that employs multithreaded technology and shared memory. It is architecture independent, and thus, it separates machine-independent and machine-dependent sections. It treats all memory as a single entity and produces sophisticated memory management by integrating communication principles into memory management. Another key characteristic is that it is an object-oriented kernel. This means that, at the operating system level, a programmer can store knowledge about the functions it is controlling. Object orientation makes it a powerful and flexible base for future development. The premise behind Mach is that it is a self-contained collection of operating system primitives that can be defined to provide a programmer with the option of using either shared memory or message communication to another module. This yields flexibility.

The same results can be accomplished without a multithreaded, multitasking operating system, but with much less efficiency. One reason that the vendors implementing multiprocessing today have had to spend so much effort adding additional processors and caches is related (at least in part) to the inefficiencies of the operating system technology they are required to use.

The goal of the Mach project was to develop a modern operating system that would be more efficient for use with parallel processing and transaction processing. While Mach is not the only modern operating system to be developed, it has the distinction of being licensed without charge. All Carnegie Mellon asks is that anyone using the operating system submit bug fixes and enhancements back to them. Therefore, it is not surprising that startups and organizations like OSF would be interested. But, because Mach is a university project, it is not yet a fully commercial operating system. Those companies that have sought to commercialize it have spent a considerable amount of time fixing bugs and making changes that make Mach faster.

When we look at Mach, it is important to remember that it is not a full operating system in the traditional sense. It is a kernel. Libraries, utilities, and commands are not part of Mach. Each vendor implementing a full environment would have to add modules to provide functions such as printer drivers and filing.

The Mach kernel consists of four basic abstractions: tasks, threads, ports, and messages.

Tasks. A task is the address space that is the receptacle for a series of related processes. A process could be a program, an algorithm, or a call. In traditional Unix, a task cannot be broken down into separate processes, so much more I/O is required. It is much more efficient to have several related processes all contained within one task.

Threads. A thread is one of the processes that are enclosed in a task. Threads are critical to parallelization because a thread allows for the greatest degree of granularity. Threads share the same address space. While forking makes this possible with Unix, it is less efficient.

Ports. In order for the threads of a task to know about each other, Mach has implemented the concept of ports. A port is a queue for sending messages between threads and handles interprocess communication between threads. It is analogous to an I/O port in hardware terminology. A port allows one thread to know about the existence of another thread and allows them to communicate. Each thread has its own port, which controls when and how information is sent to and from the thread. One thread could suspend the action of another by sending a suspend message to its port.

Messages. A message provides the content of communication that takes place between threads. A message can include object data such as the type of information stored in the thread. Therefore, a message might include the fact that the thread is a text item or an image. Messages are moved via ports.

One Memory Unit. In addition to multithreaded technology, one other major component is important to multiprocessing: the way the operating system deals with memory. The implication is that all memory is available to all processors and processes. However, there is room within this definition to define *private* memory. Private memory means that processors have looser coupling. However, in a fault-tolerant implementation, for example, it is important to keep memory private for security reasons, or, if there should be a crash, for a duplicate of what is stored in memory. Therefore, you do not want a single image of memory. In fact, Mach is developed to allow for each processor to have private memory and allows for messages to be passed from one private memory system to another. This is because Mach was designed to be used in online transaction processing (OLTP) applications where private memory protects in case a user needs to roll back to a previous state.

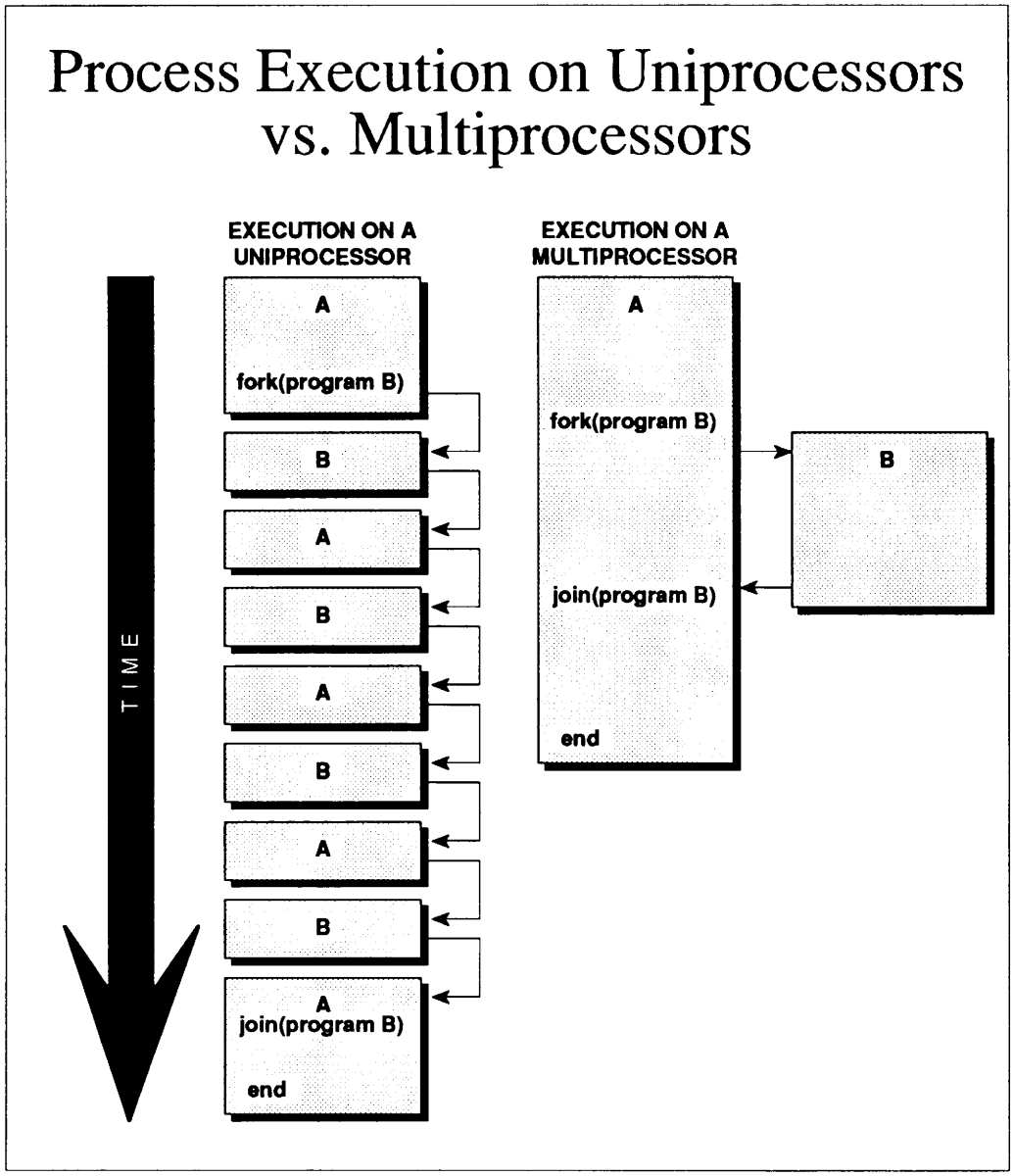
Unix and Multiprocessing

Developers looking at which operating systems and environments to write for are hesitant about writing applications, languages, compilers, and utilities needed for the most sophisticated multiprocessing environments because they realize that the portability will be severely limited. Before multiprocessing can become widespread, standardization at the operating system will have to be achieved.

Unix is at the center of the developments in multiprocessing because Unix is becoming the foundation for a standard operating system. But, as the technology requirements evolve, so must Unix. Unix was developed as a uniprocessor operating system. What's wrong with Unix when applied to multiprocessing? Well, first, Unix is constructed as a series of processes that call and create other processes. For example, each time a user logs onto a system, a new process is created. Each of these processes is independent. If two processes need to be linked, another process has to be initiated that describes the relations between the two. This is time-consuming and inefficient. When multiple processors are required to work in tandem, multiple events have to happen simultaneously, not serially.

MULTITASKING. When all events have to happen at once, the operating system has to do a lot of work, such as scheduling and processing every event. Unix proponents have long emphasized the fact that Unix is a multitasking operating system. However, it is *not* in the true sense of the term. Unix is set up to complete only a single task at a time and then proceed to the next, round-robin style. Because processors have become faster, Unix gives the impression that it can do more than one thing at a time. In reality, the operating system is simply switching when one process completes, sometimes allowing for a process granted a high priority to interrupt another process with a lower priority.

The implication is that the programmer has less control over processes and the way they work together. So, if a program has to call another program or execute an algorithm somewhere else, Unix requires that a new process be created to handle the request. This process is slow and uses a lot of system memory and I/O resources. Unix uses facilities such as forking and sockets to allow multiple processes to interact. However, there are limitations in these implementations. For example, when a Unix process has to search for a program or create a new algorithm, it has to create a separate process to handle each event. Therefore, if a query were to spawn three different subqueries, Unix would fork that process into three additional processes.



In a uniprocessor configuration, a program can spawn multiple processes. In a multithreaded kernel, processing is more efficient.

This is inefficient in terms of memory use and input/output processing. A multithreaded kernel would make this process initiation unnecessary.

MEMORY MANAGEMENT. The same problem occurs in the way Unix handles memory. When only one processor is involved, memory is not an issue. The system, using mechanisms like sockets, simply matches up memory with processes. When multiple processors are involved, memory management becomes much more complex. System V.4 has evolved to handle virtual memory management so that a system can treat all the memory within a configuration as if it were one large memory system. Therefore, System V.4 is well-designed to handle this facet of multiprocessing.

KERNEL SIZE. One problem resulting from attempts to adapt Unix to changing requirements has been the increasing size of the operating system itself. Over the years, code has been added to the kernel that has resulted in inefficiency. For example, the SunOS kernel is 400K; the Mach kernel is 80K. Another problem with the Unix kernel is that there are many hardware dependencies within the kernel itself. Therefore, any vendor adapting Unix for multiprocessing has to radically change the kernel itself, making portability more difficult.

One reason that developers are looking at smaller kernels such as Mach is to maintain portability by having only basic services at the core. Other utilities and services that are likely to change are created as modules outside the kernel. Changes can be made only within a single module, since parts don't tend to touch each other. This makes it easier to keep key elements standard.

Hardware price/performance requirements are forcing multiprocessing into the mainstream. It is not surprising that AT&T, UI, and OSF are coming to the same conclusion at the same time.

How Unix Will Change for MP

It is becoming clear that the competitive nature of hardware price/performance is forcing multiprocessing into the mainstream. Therefore, it is not surprising that the operating system vendors, such as AT&T, its partner, Unix International (UI), and the Open Software Foundation (OSF) are coming to the same conclusion at the same time. Members of both organizations have urged UI and OSF to move towards new implementations of the operating system.

OSF. OSF surprised the industry by switching from its intention of using IBM's AIX kernel to a Mach-based microkernel. The microkernel won't show up for several years and will only run the base system. All hardware-specific code will be placed in separate modules on top of the small kernel.

The primary consideration behind this move was the demand by OSF's members that the organization provide developers with an operating system kernel that was designed to support multiprocessing. While IBM's AIX operating system has the potential to be modified to add multiprocessing support, multiprocessing has not yet been implemented.

OSF'S USE OF MACH. A key reason for OSF's adoption of Mach is the fact that it was designed from the outset for multiprocessing. OSF's plan to use Mach is a two-stage process.

Role of AIX. While OSF will not use IBM's AIX kernel, it will include most of the AIX libraries and utilities. There is good synergy between Mach and AIX because, from the beginning, AIX was designed to be a modular operating system.

Leveraging Encore's Work. In the first stage, OSF will take the existing implementation of Mach from Encore. This version of Mach includes both the base-level kernel and the Berkeley operating system (Berkeley Software Distribution, or BSD) on top of it. So, on top of the Mach kernel will be use processes and server programs to ensure Unix compatibility.

Encore has done considerable work to commercialize Mach for multiprocessing. This is an outgrowth of DARPA's need for both multiprocessing and parallel processing systems. Encore has parallelized the BSD file system, network facilities, and device drivers. Further, Encore has tackled key software such as a shared-memory, multiprocessing program debugger.

Locking Technology. Some of the details that Encore has worked on include key issues such as the way processes are locked to protect their integrity and how to handle interrupts. When tasks are subdivided into multiple threads that are executed simultaneously across a parallelized network, locking and synchronization are paramount.

OSF/2. The second release of the OSF operating system, known as OSF/2, will be a Mach-based microkernel. When that happens (probably not until 1991), OSF will have a modern and modular operating system that can help it become competitive.

UI. Likewise, Unix International (as the organization that tells AT&T what user requirements for System V are) established a task force to determine what should be changed in System V to allow for a standard way to handle multiprocessing. Partly as a show of good faith and partly because of common interest, UI invited the OSF to participate in the task force. The group was headed by Dr. Jerry Popek, chairman of Locus Computing and a leading industry visionary on the topic of distributed computing. Joining Popek as co-chair was Digital Equipment's Miche Baker-Harvey, who happened to be chairperson of OSF's multiprocessing special interest group. While the task force

Bandwidth Issues

IT IS EASY to see how the bandwidth and the bus become critical components in multiprocessor systems. Since messages have to be passed between processors, the I/O overhead can be steep. In many multiprocessor implementations, the system tries to minimize the amount of message traffic. Therefore, a system might wait until it has several messages to pass before addressing the bus. On the other hand, when processors are tightly coupled within the same physical box, bandwidth is not an issue when messages only have to jump from one processor to the next. Therefore, in such a configuration, it would be beneficial to allow messaging as often as necessary. Today, no implementation of multiprocessing allows for both levels of message-passing.

We expect that vendors will find a way to provide more granularity in I/O. One future requirement will be to combine these two ways of messaging so that I/O-intensive tasks happen less frequently but those that require less overhead can take place whenever appropriate.

proposed several possible alternatives to AT&T's Unix Software Organization (USO), the conclusion was clear: A new kernel for supporting multiprocessors is needed.

UI'S Multiprocessing Task Force. The task force report focuses on an API that would isolate the user from different multiprocessor technologies. In addition, the report lays out a multiprocessing architecture and the kernel level implementation. However, the report does not go as far as directing USO to use a Mach kernel or System V virtual memory as the foundation for achieving multiprocessing. The group concluded that there are strengths and weaknesses to both approaches. One scenario would call for the use of System V virtual memory, while another proposes adopting the Mach kernel. The needs of UI members are very close to those of OSF members.

The major findings of the task force are the following:

- Support for threads will be initially implemented as library services, as they are needed for both convenience and performance. Two layers of primitives are needed: library threads and kernel threads. These should be lightweight threads to be enclosed within a single address space or process. The first priority in the view of the task force is for a threads library; a kernel library could be added later. This approach is pragmatic. It would be a lot more work for USO to add threads to the kernel than to add a threads library above the kernel. A key issue addressed by the task force is migration. For example, it recommends that there be upward compatibility for device drivers and streams.
- A key recommendation is that OSF and UI form a joint workgroup to achieve a common set of API extensions to Unix for support of multiprocessor services.
- System V needs a shared-memory multiprocessing structure.
- Unix needs utilities to manage scheduling and binding services. Most discussions concerned binding of threads to processors, including a thread attached to a specific processor and those dispatched together as a unified group or "gang."
- Unix needs a fully multithreaded kernel.
- Specifications for a virtual memory layer are needed.
- A structure needs to be put in place to allow for the creation of value-added kernel modules so the operating system can be upgraded in an incremental manner.

Relationship to Posix

POSIX AND P-THREADS. At the same time that OSF and Unix International have been grappling with how each organization will handle multiple processors, both have endorsed the IEEE Posix Committee's approach to the way threads are created and handled. P-Threads (Posix-Threads) provides for multiple threads of control within a single shared address space, mutual exclusion locks for protection of critical regions, and rules determining how threads will be synchronized. The Posix specification draft recommends that threads need to share resources to cooperate. Memory must be widely shared so the threads can cooperate at a fine grain. Threads keep data structures and the locks protecting those data structures in shared memory. Based on this definition, any resource that could be shared among threads needs to be in shared memory. Examples include file descriptors, path names, and pointers to stack variables. The root directory, file descriptor table, and address space must all be shared. The proposed specification recommends that separate processes communicate through well-defined interfaces, independent of implementation.

Much of this work was initiated by the Real-Time Committee of the IEEE Posix 1003 Group (1003.4) when it discovered that threads were a key component in achieving real-time Unix. At the same time the Real-Time Committee was developing its specifications, a new committee had formed. This committee quickly began to work with the Real-Time Committee to ensure that the work that group had done on threads would take into account the needs of the multiprocessing vendors. The second committee, which has only had a few meetings, is closely examining the threads work already completed by the Real-Time Committee. Its first task was to evaluate the proposals for a common threads API. This P-Threads specification has included the issues important to OSF and UI researchers. Therefore, it is becoming clear that all three groups—Posix, UI, and OSF will agree on a common API for threads.

BENEFITS OF A COMMON API. Having a single API for threads is probably the most significant event in spurring the multiprocessor industry. If there is a common API, differences between multiprocessor implications will be transparent to the end user and software developer. Software will be written to the threads API rather than to the specific implementation of threads in the operating system.

Conclusion

A lot of progress has been made by vendors in developing powerful multiprocessors. However, we are only in the infancy of multiprocessing and parallel processing technology. The most important developments in the next few years will be in the area of standardization, ranging from standards in kernel design and implementation to a common threads API. All these developments will help speed the generation of applications designed to run on multiprocessor implementations.

While these are exciting times filled with complex technical accomplishments on the part of researchers and implementers, it is becoming clear that the most important tools and applications developments are still ahead.

Over the next few years, we can look forward to a large number of applications that will leverage base-level hardware and software technology. The first to benefit will be the scientific and engineering world, which needs parallelization for complex computational applications. Next in line will be the online transaction processing (OLTP) users who will leverage all of the power to be gained from this next-generation technology. Finally, we expect that multimedia applications will be aided by advancements in multiprocessing, multithreading, and multitasking—and, in the future, parallel processing.

NEWS ANALYSIS

PRODUCTS • TRENDS • ISSUES • ANALYSIS

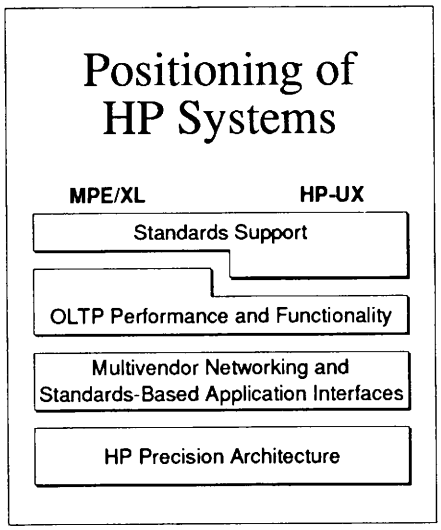
• INSIDE •

• HEWLETT-PACKARD •

Commercial Unix Gets Real

During the past few months, Hewlett-Packard has been putting meat on the bones of its strategy to become the most important vendor of multivendor distributed computing environments. Recent announcements have ranged from workstations and user environments (Apollo workstations and NewWave Office) to multivendor network management products (HP OpenView) to distributed applications enablers (Team Computing and LAN Manager/X).

Most recently, HP fleshed out the servers in its Cooperative Computing Environment (CCE)—the name HP uses for its distributed network computing environment. HP introduced a wide variety of enhancements and extensions to its HP Precision Architecture (HP-PA) RISC offerings, with a specific thrust towards supporting transaction processing. The new servers also affirm HP's commitment to standards-based solutions by delivering a robust platform for commercial Unix applications and, in fact, demonstrating the first generation of these solutions. The new boxes allow the company to take direct aim at the midrange system markets, specifically Digital's VAXs and the low end of the IBM 3090 line.



HP's vision of the respective roles of Unix and MPE allows for both overlap and specialization, within an environment that promotes interoperability.

BROADENING THE HP-PA LINE. HP billed its Precision Architecture server announcements as the company's largest ever. There were 23 different hardware products (12 multiuser systems hosts and 11 network servers) announced in all areas—high end, low end, and midrange. Most interesting are the increased performance at the top of each range and the new functionality in multiprocessing and fault tolerance. These enhancements are included in both the 3000 (based on HP's MPE) and 9000 (based on HP-UX) series.

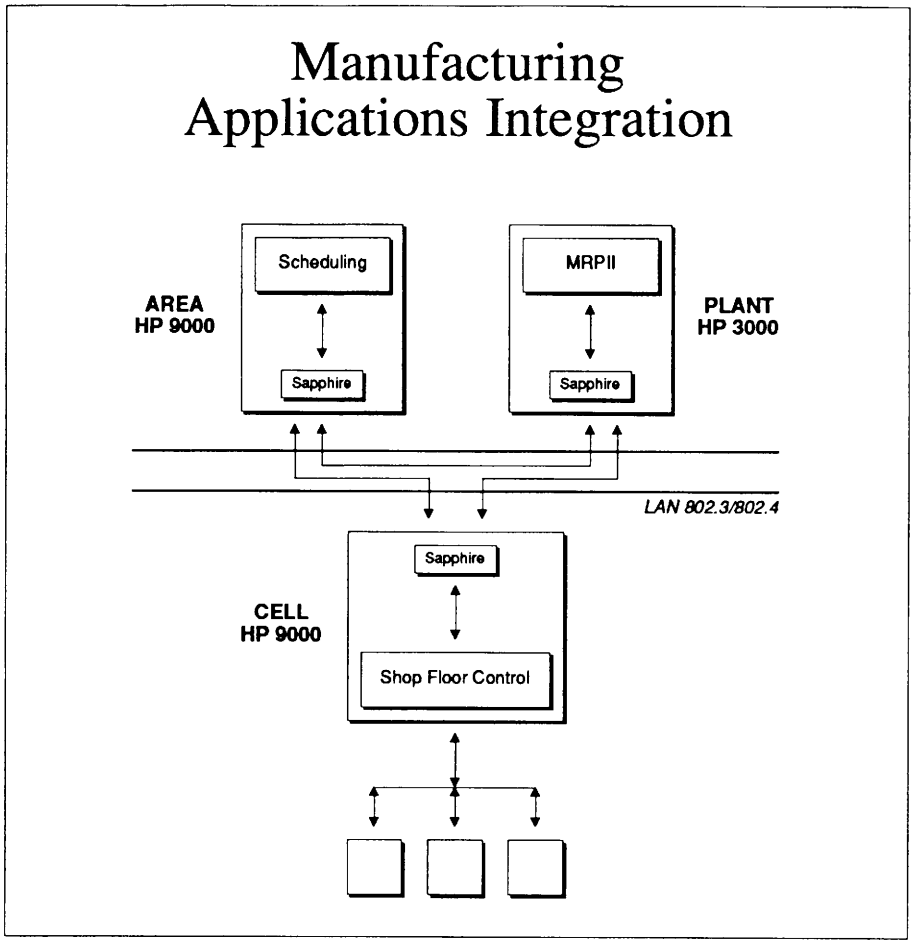
Transaction Servers. HP is positioning the high end of these new systems as transaction servers. The key enabling elements are the new 50 MIPS RISC chip, the symmetrical multiprocessing architecture (initially supporting two processors, with three-and-four processor support due by the end of the year), and the enhanced administrative functionality provided by disk-mirroring (portending the move to fault-tolerant systems enabled by HP's agreement with Sequoia—see "Sequoia: HP's OLTP Connection" below), increased security, and DAT backup systems.

MPE and Unix. For Hewlett-Packard, this is not MPE vs. Unix, but a true coexistence within an overall strategy of leveraging the current strengths of MPE/XL while enhancing the ability of the Unix products to handle commercial applications. This can be contrasted with IBM's confusion and Digital's apparent ambivalence as exemplified in Digital's message that RISC and Ultrix provide great price/performance, but VMS is for serious applications. In addition, HP has a distinct advantage over IBM and Digital in integrating its proprietary and open lines, since HP-PA supports both (see illustration at left).

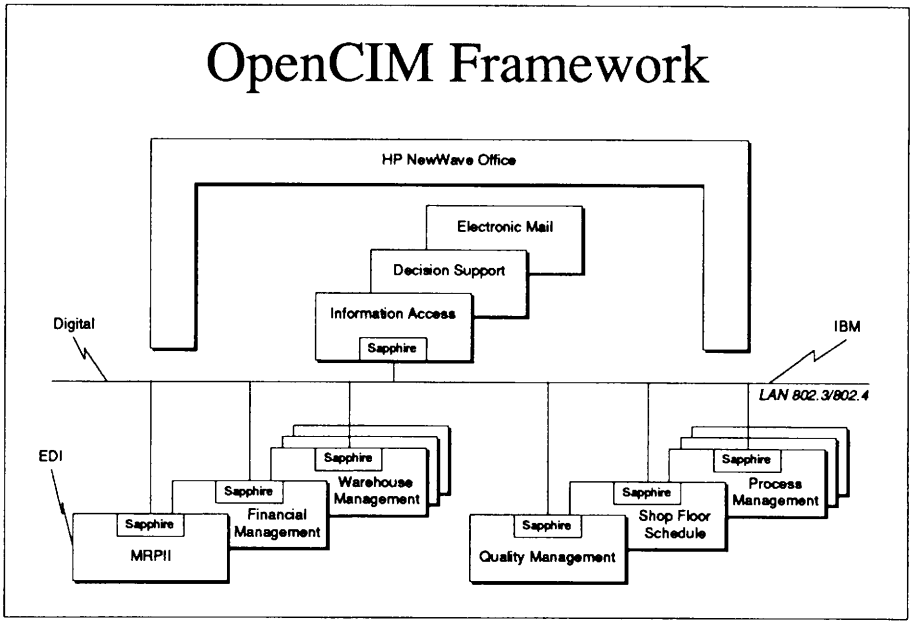
COMMERCIAL UNIX APPLICATIONS. HP is doing more than just building the enabling platform for Unix applications. The company is demonstrating its commitment by delivering key applications on this platform, most notably NewWave Office, its network management station, its integrated voice/data applications (see below), and the addition of MRP II manufacturing applications to the 9000.

MRP II and OpenCIM. HP has been traditionally strong on the factory floor, both in Computer Integrated Manufacturing (CIM) and Manufacturing Requirements Planning (MRP). HP is bringing these critical applications to the 9000 under HP-UX, with MRP II for Unix due to be delivered in the second quarter of 1990. During the third quarter, HP plans to introduce a Unix product (code-named "Sapphire") that is essentially a toolkit and a set of APIs which enable the integration of dissimilar manufacturing systems and functions (see illustration, above right).

Sapphire, which will become available on MPE early in 1991, will be the enabling element in uniting the shop floor and the office under what HP is calling its OpenCIM Framework (see illustration, below right). A key to the success of this strategy is HP's ability to get third parties to write to the APIs (a key to the success of HP's strategy in many areas, most notably NewWave and OpenView). —*D. Marshak*



Sapphire will integrate different manufacturing functions running on different platforms ...



... as well as provide the glue to integrate manufacturing and office functionality within a distributed computing environment.

Sequoia: HP's OLTP Connection

Hewlett-Packard's new strategic relationship with Sequoia Systems gives it access to leading edge multiprocessing, fault-tolerant Unix OLTP technology. The relationship includes a marketing agreement, technology licensing, and an equity investment in Sequoia by HP.

Under the marketing agreement, HP will market the current Sequoia 300 multiprocessor, fault-tolerant Unix sys-

tems and future systems, targeting the telecommunications industry. In turn, Sequoia is committing to using the HP Precision Architecture (HP-PA) RISC chip sets in its future systems.

Under the technology agreement, HP is gaining the right to use most of Sequoia's fault-tolerant, multiprocessor patents in future HP systems. A cross-licensing agreement covers any improvements to the multiprocessing technology made by either company.

Fault tolerance is increasing in importance to a growing number of customers. Being able to offer Sequoia's technology fits well with HP's strategic market position to go after the telecommunications market. In addition, elements of the technology probably will end up in the HP-PA and HP-UX.

MARKETING AGREEMENT. Sequoia will supply HP with fault-tolerant systems under the HP label—under a multi-year agreement with renewal options. HP will be the exclusive marketer of the Sequoia 300 series and future products under development in the domestic telecommunications marketplace, along with nonexclusive rights globally. Initially, HP intends to focus primarily in United States and on the telecommunications market.

Currently, the Sequoia uses the Motorola 68030 in its 300 series (after using the 020 in the 200 and the 010 in the 100). Sequoia has said that it will use the HP-PA RISC technology in future generations of its fault-tolerant systems. The transition to the RISC architecture probably will be in two generations—there is already a next-generation product in the works, probably based on the 040. The HP/Sequoia agreement calls for specific margin purchase guarantees.

LICENSING AND EXCHANGE TECHNOLOGY. Under the agreement, Sequoia grants to Hewlett-Packard a license option to develop future RISC-based systems using most of Sequoia fault-tolerant and multiprocessor patents. The companies are cross-licensing any technology improvements made as a result of the partnership.

EQUITY POSITION OF HP INTO SEQUOIA. Finally, HP is making a $5.8 million equity investment in Sequoia (a privately held company). As a result, HP has visitation rights to Sequoia board meetings.

SEQUOIA BACKGROUND. Established in 1981, Sequoia began shipments in 1986. Today, it is a worldwide supplier of fault-tolerant systems supporting Pick as well as Unix.

The Series 300 supports about 251 tps on a 12-processor system. Sequoia claims to achieve near linearity in its scaling, with each processor supporting about 21 tps.

The system is expandable on a linear basis even when under operation. To date, Sequoia has shipped system with up to 12 processors and has developed systems supporting 18. Theoretically, the 300 Series can support up to 64 processors.

SEQUOIA'S UNIX. Sequoia's Unix OS—TOPIX—is based on V.2 and is moving toward Posix compliance (which is perhaps 12 to 18 months away). The next OS release will be closer to Posix and will move to V.3 and then V.4 compliance as well.

The customized OS kernel is integral to the system fault tolerance. HP will be striving to make sure that the look and feel of the system as well as the services and the interfaces are consistent with HP-UX.

HP also has had a bit of experience in juggling applications among a variety of systems. Currently, the Motorola-based systems and the RISC-based system use the same C compiler and operate off the same kernel base. Third parties have moved application code back and forth via a recompilation, but the process seems fairly smooth.

HP does, however, intend to bring the Sequoia FT technology into HP-UX, in ways to be announced later.

The result of this agreement, along with some of Hewlett-Packard's previous acquisitions—notably Apollo—is the creation of a very broad base of available technology from which both companies can select. For example, the PRISM parallelism technology, which is also moving into HP-PA, will now also be available to Sequoia.

TELECOMMUNICATIONS MARKET. Companies invest in fault tolerance when the cost of the downtime exceeds the premium for fault tolerance, which runs at about 50 to 150 percent of conventional system pricing.

Due to technology advances, this premium is decreasing over time. A few years ago the fault-tolerant premium ranged up around 300 percent. Now it's about 100 percent. This decrease, both absolute and relative, should continue.

Of the applications with a high perceived cost of downtime, significant concentrations are in the phone industry. Because phone networks are being fault tolerant, the providers expect comparable capabilities in computer systems. HP is putting a great deal of marketing and sales effort into this area and has seen a significant increase in FT demand.

This most recent agreement seems a shrewd bargain for both companies. HP picks up some valuable technology that supports its drive for leadership in the commercial Unix market, and Sequoia picks up validation for its technology as well as some cash. Not bad.

—M. Millikin

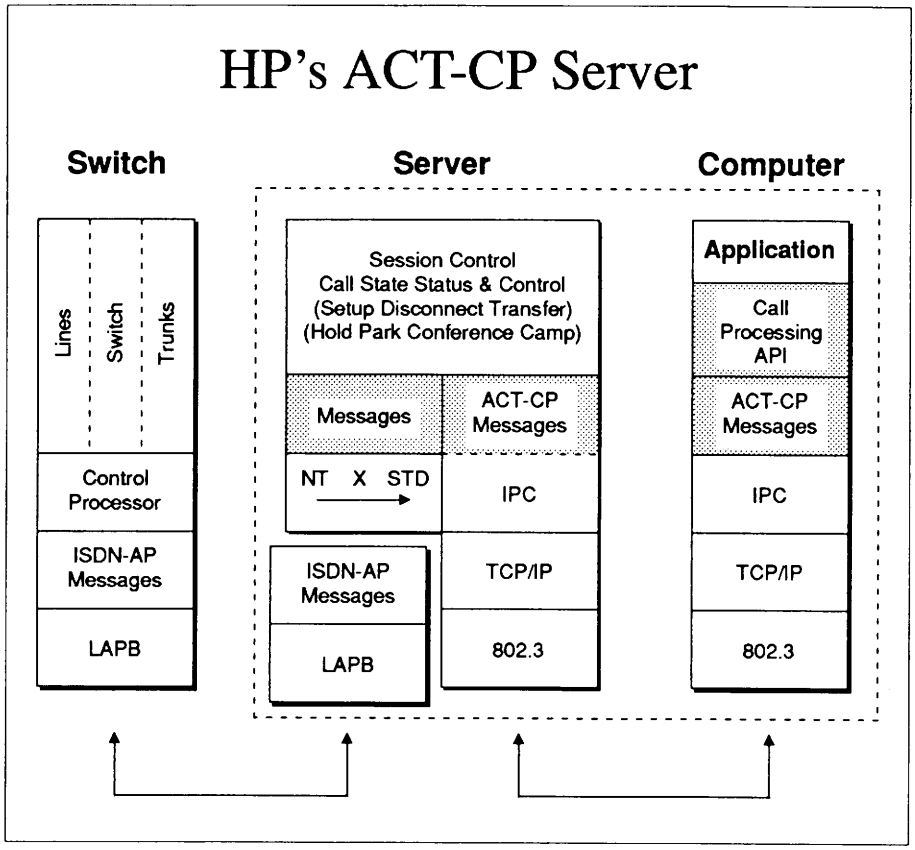
New ACT: HP Gives Unix a Voice Platform

With its new Applied Computerized Telephony (ACT) platform, Hewlett-Packard has given Unix users their first opportunity to integrate telephone call processing with data processing applications using standard programming interfaces. The first implementation of the ACT architecture is an HP9000 (HP-UX) call processing server that works in tandem with a Northern Telecom SL-1 PBX to coordinate call processing with applications under HP-UX

and HP's MPE. Integration with other PBXs will follow, as well as addition of an integrated voice processing applications platform.

With ACT, HP is targeting telemarketing, customer service, reservations, credit, and related applications. In each of these applications, an agent working at a computer screen typically obtains or provides information to a customer over the phone, using a computer application to retrieve or enter information. In most companies, the phone system and the computer application are separate, unlinked resources. By linking the receipt or dialing of phone calls with the agent's existing information-management application and by leveraging information about calls and callers available on the public networks, companies can improve the efficiency and speed of transactions with customers or clients. Some companies have doubled the number of transactions their agents can handle by installing call-data processing systems.

IBM and Digital Equipment Corporation are also working to support the addition of call processing to telemarketing and teleservice applications, but with proprietary platforms. IBM offers CallPath to its mainframe customers and AS/400 Telephony Application Services to its midrange users. Digital offers Computer Integrated Telephony. Both vendors intend to offer Unix versions of their integrated call-data applications services, but don't do so yet. The fact that HP implemented its call processing server on Unix first indicates both the strength of its commitment to Unix and the reality of the company's position in the market. An MPE-based platform would be attractive only to HP's proprietary installed base. By going with a Unix solution, HP has positioned itself to lead development of integrated call-data processing standards while also giving its MPE base new functionality.



Hewlett-Packard's new Applied Computerized Telephony (ACT) server is a Unix-based platform that currently integrates HP-UX and MPE applications with call processing services provided by Northern Telecom SL-1 PBXs. The architecture, however, allows HP to slide out the Northern Telecom interface and slide in those of other PBX vendors, including AT&T and Siemens.

ARCHITECTURE. HP's ACT employs a so-called proxy approach to integrated call and data processing. Each element in the application—the PBX, the computer, and, potentially, a voice processing subsystem—retains control of its sphere, responding to requests for services from other elements. Thus, a credit application that needs a number dialed passes the number to the PBX and asks the PBX to dial it. This is an inherently flexible architecture that leverages the strengths of each element in integrated call/voice-data applications. The alternative, called the mirror approach, gives control of all resources to the data processing platform, creating the potential for a big bottleneck.

ACT is based on an architecture that will soon become familiar, since all major vendors employ one form or another of it. A call processing server sits between the PBX and the computer, linked by a special interface designed to carry messages and event notifications between the PBX and applications on the computer. (See illustration above) In the first implementation of ACT, Northern Telecom's ISDN/Application Processor (ISDN/AP) is the interface, but HP intends to support others. HP is working with both AT&T and Siemens to implement their separate messaging interfaces within ACT.

HP's value-add in this architecture is its ACT Call Processing (CP) message set and its MPE and HP-UX application interfaces to it. The ACT-CP messages are the currency of transactions between computer applications and the PBX. The messages define the format and content of directives like "make call" and "transfer call," as well as notifications about the status of calls and stations (handsets and terminals) on the network. HP's message set is built on the model being defined by the European Computer Manufacturers As-

sociation's (ECMA's) Task Group 11, the leading standards body in this area.

HP's ACT-CP messages are relevant to the computer side of an integrated call-data processing application. Each of the PBX vendors defining message interfaces for these applications also has its own message set. HP's ACT-CP server transforms HP's messages into a neutral format for mapping to the messages of a particular PBX interface. This is a practical implementation, given the diversity of message sets and interfaces in the market right now. Eventually, HP will be able to migrate along with the industry to a standard message set and messaging interface, most likely based on ECMA's work.

SIMILAR APIS. The beauty of an integrated call-data processing platform is its APIs, which allow call processing features to be added to existing applications. HP provides two APIs with ACT, one for MPE applications and one for HP-UX software. The two APIs are similar, according to HP, reflecting only the different sets of facilities available in the MPE and HP-UX operating environments. Developers should have little trouble modifying applications written to one API—say, MPE—to run under the other API. We can't vouch for this claim—the product set is still too new to judge.

HP will keep its APIs and message set private for an unspecified period of time—as long as the company believes it gains a competitive advantage from doing so. HP does intend to place its APIs and messages into the public domain. The sooner this happens, the better. HP has more to gain—the initiative on standards—by moving its technology to the public domain than it does by holding it close to the vest.

MIGRATION TO VOICE PROCESSING. ACT is an architecture, not a product, and it includes a design for working voice processing into applications integrating the telephone and the computer. Voice processing today is provided by special computers via applications like voice mail, automated answering, and voice store-and-forward. In the near future, voice recognition, voice synthesis from text, integrated voice messaging, and other advanced services will be added to this mix. (Limited voice recognition is already available from some vendors.)

HP's plan is to implement voice processing services on a standard platform under the ACT umbrella. The plan calls for an API to voice processing services provided by a server on a network, accessible via an open interface. HP is working on this API to allow developers to add voice storage and playback, speech synthesis, voice recognition, and other advanced services to existing applications.

A LONG PARTNERSHIP. HP's choice of a PBX partner in delivering an ACT implementation was obvious. The two vendors have been working together through a standalone organization called Cooperative Network Operation (CNO) for about 18 months. ACT will be marketed and supported by CNO, leveraging the coordination of resources the two companies have hammered out as CNO has evolved.

The ACT marketing plan calls for joint sales calls by HP and Northern representatives along with an independent vertical software developer. The collaboration will extend from design of the solution for the particular customer to installation through support.

CONCLUSION. HP and Northern Telecom estimate the market for products connecting PBXs and computers to be $1 billion a year. This opportunity, they say, is rooted in corporations' needs to sell to and service customers more effectively at a reasonable cost. We're inclined to agree with this logic, and we believe the industry has a long way to go to satisfy customer needs. There's been a lot of progress during the last year, but PBX-computer interconnection remains a minefield of choices for users that affords little flexibility in coordinating multiuser networks.

We're encouraged that HP has chosen Unix as the platform for its call and voice processing services. Among its major competitors, IBM has no announced strategy for integrated call-data processing under Unix. Digital's strategy is to add call and voice processing services to its Network Application Support for VMS and Ultrix. That means VMS will be emphasized first.

We reserve further judgment until HP decides on a move to the public sector with its technology. When that happens, Unix users will really have something to cheer about—widespread availability of call processing services.

—J. Rymer

•USER INTERFACE•

UIMS: A Critical Element Arrives

Find yourself in a conversation about graphical user interfaces (GUIs), and, more often than not, what you're really talking about are window systems: X-Window (and its many flavors, but especially Motif), Presentation Manager, NeWS, MS Windows, Mac Toolkit, NextStep. But a window system does not a graphical user interface make. The industry hasn't been giving enough attention to the tools necessary for actually creating graphical user interfaces. Those tools come under the catch-all heading of user interface management systems (UIMSs).

A UIMS can be any number of things—a screen painter, an interface layout tool, an interface customization facility. We can give you a working definition, and we're on the lookout for products that take on the shades of that definition, products like DataViews, a UIMS from VI Corporation (Amherst, Massachusetts).

A WORKING DEFINITION. A UIMS is not a user interface, but a means of creating graphical front ends for new and existing applications. With it, you design only the interface, and then create links to the application. Since pro-

gramming directly with a window system or toolkit can be a complex, taxing exercise, a UIMS automatically generates user interface code as you paint the screen. Ideally, a UIMS should be portable, both in terms of multiple window systems support and modularity. You should be able to borrow interface functions from other applications—a plug-and-play approach. In addition, a UIMS should provide testing, prototyping, and evaluation functionality. In other words, you should be able to test and refine the behavior of a user interface before actually binding it to the program. It's more than just testing the way the interface looks and interacts with the user; it's also making sure that deleting a button or menu item doesn't cause any unforeseen repercussions that could potentially ruin an application.

A WORKING EXAMPLE. Sometimes, it's best to describe a technology by looking at an actual instance of it. DataViews provides developers an interactive, graphical development environment for building GUIs. The product consists of two components: DV-Draw, a menu-driven graphics editor, and DV-Tools, a library of C subroutines that display and manage graphics, objects, and drawings created with DV-Draw. You also use DV-Tools to bind DV-Draw objects to data and operations within the application.

Paint, Not Code. DataViews' graphics editor is fairly thorough and straightforward. It comes with your basic drawing primitives (e.g., rectangles, lines, circles, etc.) as well as predefined input objects (e.g., menus, buttons, switches, toggles, check boxes, etc.) and practical predefined graphics (e.g., scales, meters, and a whole bunch of assorted graphs) (see illustration above). DV-Draw will even accommodate animation—albeit simple animation. You can develop additional objects with DV-Tools, but, of course, once you're in DV-Tools, you're into programming.

Portability. DataViews is conceptually device-, hardware-, window system-, and operating system-independent. In addition, via DV-Tools, it can help you integrate existing commercial software packages. DataViews also lends itself to a modular approach to interface development. Since the interfaces you design are completely independent of applications, it's simple to plug in a "used" interface function.

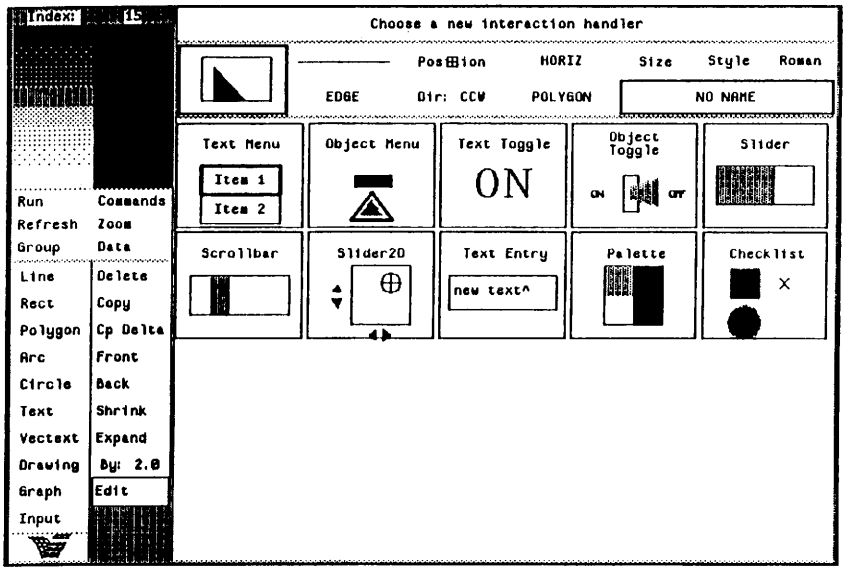
DataViews also provides templates to guide you through the design of specific look and feel—like Mac or NextStep. We would prefer to be able to pick the appropriate style from a menu and have the system automatically bang out the right conventions and interface code. But we haven't seen any really good UIMSs that do that. Apparently, it's no small task. The complexities of individual window systems are simply too demanding for this kind of flexibility. However, representatives from VI Corporation plan to menu-ize custom interfaces to some degree in the next release. We presume, though, that this capability won't be very extensive.

Prototyping. DataViews makes a valuable prototyping tool. After designing a screen, you set up variables and ranges that affect the interface you've designed. For example, let's say you're building a monitoring application that notifies the user with a visual alarm when data reaches a prespecified value. And you set it up so that, by mouse-clicking on the alarm, the user can view a chart of the data. With DataViews, you can execute the interface before binding it to data with a Preview mode. If you decide the chart is wrong, you exit out of Preview mode back into DV-Draw to edit it.

FOR PROGRAMMERS ONLY. The problem with UIMSs today (including DataViews) is that they're still developers' tools. They haven't yet evolved to where they'd be suitable for end-user programming. While the graphics editing element of DataViews is simple enough, linking the interface to an application isn't. DV-Tools are programmer's tools. We looked at the manual. Heavy stuff.

Too heavy, perhaps, even for some developers. DataViews is great as long as you stay within the confines of its predefined subroutines and as long as you're not dealing with very complex applications. DataViews holds a lot of

DV-Draw Input Objects



In addition to basic graphics primitives, DataViews offers predefined input objects.

information—attributes of an interface object (e.g., a graph), links of attributes and data (e.g., having the graph pop up when a critical value is reached), attributes of attributes (e.g., changing the color of that graph, moving it, or including some animation), links of attributes of attributes to data, attributes of attributes of attributes... You can go layers deep, and it can get scary.

VI Corporation is targeting developers of command and control applications, CAD/CAM, network and process monitoring, instrument simulation, and financial analysis. And the company has designed its product with these customers in mind. In other words, there's not much need to go beyond the scope of the product. Developers of different types of applications (perhaps office and manufacturing applications) may need to look elsewhere. —*L. Brown*

•NATURAL LANGUAGE•

DBMS Tools for the Rest of Us

We're not all DBMS developers. Most of us don't know SQL; most of us don't care to learn it. Yet an increasing number of casual users must access databases, and they can't go pestering MIS every time they need to do some querying. Furthermore, both casual and serious database users have been calling for better tools to access and interpret data. To take advantage of the broader audience of low-tech users, natural-language vendors are coming out with non-SQL front ends and tools for database access.

THE PROS AND CONS. There are a number of considerations in building a database access tool—namely, the syntax, structure, and semantics of the database design. Syntactically, users need a way to poke into data without knowing SQL. Structure addresses the join paths and migration around the database—the model the database uses. Even if you do know SQL, you have to know the relational model. Semantics has to do with the focus and organization of the database. Is it financially oriented? Inventory oriented?

Natural-language vendors are always quick to point out the pitfalls of visual data access (either menu- or forms-based): the frustration of accessing data indirectly and wading through layers of menus. On the other hand, the folks who sell visual programming products are quick to slam natural language programming and data access methods. And they have a point; most natural language products have their limitations and ambiguities.

Our main concern is that you are never quite sure whether your query result is really the answer to your natural language query. The ways in which the product understands your query may not be the way you intend it. And then you get the wrong data. For example, you may ask "Which salespeople earned over $80,000?" The answer you get back may be total salary when you only wanted commissions. You may even get back a list that spans multiple years when you really only wanted last year's results. And you wouldn't necessarily know whether or not you were looking at invalid data.

Most natural language products have their limitations and ambiguities. The ways in which a product understands your query may not be the way you intend it.

Handling the Ambiguities. Natural Language Incorporated (Berkeley, California) has found ways around some of the ambiguities, but some elements still need to be worked out. The flagship product of NLI—appropriately called Natural Language—provides access to several major relational database management systems (RDBMSs), including Oracle, Sybase, Informix, Ingres, and Rdb. Natural Language doesn't use Boolean statements or keyword mapping for querying. It uses linguistic processing instead, with some underlying AI. Natural Language develops an internal representation of your query, performs some deductive reasoning (i.e., thinks about the query in the context of your target application), then generates SQL to query the database. In other words, it understands your query as an English question first, then maps it to the database.

Natural Language also has some helpful features for checking the accuracy of your query. To verify that Natural Language has interpreted your query correctly, the system includes its rendition of the query along with its results. For instance, if you ask, "Which salespeople made more last year than this year?" it might report that its findings were based on the more accurate question "Which salespeople made more in fiscal 1987 than in fiscal 1988?" Furthermore, Natural Language takes a dialogue approach to querying. You may start out asking, "How many products were shipped to Des Moines in 1988?" When you follow that up with "How many were returned?" the system knows that you're still talking about products shipped to Des Moines. The system also prompts you when it needs further clarification on a query.

But these solutions are not complete. We still bumped into the ambiguity problem. At one point, for instance, we were working with a database that reached only to 1987. Just out of curiosity, we asked "Which salespeople made more in 1987 than 1988?" The response was "none." Well, that's simply not accurate. Instead, the system should have let us know that it had no data on 1988.

At the Back End. Natural Language requires preconfiguration at the back

end—setting up things like definitions for linguistic processing and deductive reasoning. This is best handled by the database administrator. NLI offers a knowledge-based, interactive tool called the NLI Connector, which guides you through the implementation of the relationships and concepts behind specific database applications. In addition, the company markets the NLI Gateway for querying data held in different databases and different machines.

THE COMPETITION. NLI is certainly not the only vendor putting out natural language front ends for databases. A similar product comes from Battelle (Columbus, Ohio), called Natural Language Query, or NLQ. (You'd think these companies could be a little more resourceful in their product names, wouldn't you?) NLQ also lets you query the database in English, and then translates your request into SQL to search the database. And, also like Natural Language, NLQ has a dialogue-driven set-up tool for establishing communications protocols, downloading the database structure, and defining user vocabulary. NLQ runs on DOS PCs and is available for Oracle, Ingres, Basis/DM, and DB2 databases. NLQ supports most server configurations, including IBM, Pyramid, Unix, and VAX machines. However, unlike NLI, Battelle has no tool for querying multiple databases, and you need a separate NLQ version for each database it supports.

Although other natural language products abound, such as Intellect from AI Corporation, NLI sees most of its competition coming from other means of data access—4GLs and forms- and menu-based systems. Some interesting graphical front ends are popping up. Ingres, for instance, recently released a graphical front end to its RDBMS. While there's no natural language involved in it, the product gives you graphical views of data and a visual query editor. Instead of typing lines of SQL statements, you point and click on the information you need. (The product, incidentally, was developed with Sun and is only available for Sun workstations.) Either way, the point is to simplify database access. We hope to see more of the same—and perhaps more sophisticated tools as well. —*L. Brown*

•RIGHTSOFT•

Constructive but Clumsy Criticism

Fact: More and more people have access to word processing software. Another fact: These people have little experience as writers.

Result: There is a lot of bad writing out there.

Solution: Grammar correction software ... or is it?

Grammar correctors are growing in popularity, predominantly in the DOS market where Grammatik VI from Reference Software, Correct Grammar from Lifetree, and RightWriter from RightSoft seem to be battling it out for market share.

RightWriter analyzes your document, inserting comments and recommended suggestions ("Redundant. Consider omitting 'recommended'").

And now, one of the combatants has released product for the Unix environment, where, sorry to say, there are also a lot of bad writers.

RightWriter is now available for SCO Xenix 2.3 or higher on a 286 or 386 platform. The product costs $295 per CPU license. The product is compatible with the Xenix versions of Microsoft Word 3.0, WordPerfect 4.2, and Samna Word IV and Plus IV. The product also works with ASCII documents.

GOOD FUNCTIONALITY. The suggested grammar corrections and writing style analysis is right on! (RightWriter would undoubtedly tag the last sentence as colloquial—or perhaps archaic.) Users can create a customized style sheet by selecting from among five writing types (business, technical, fiction, manual, and proposals) and three audience types (high school, general public, and college). Grammar and word usage rules may be turned on and off (for example, split infinitives may be allowed, but dangling participles should be flagged).

RightWriter analyzes your document, inserting comments and recommended suggestions ("Redundant. Consider omitting 'recommended'"). A readability index (grade level) is provided, as are a multitude of indexes on other considerations, such as jargon and sentence length. These indexes may be included in the marked-up document or optionally turned off.

CUMBERSOME AND NOT INTEGRATED. But the product suffers from a lack of integration. (This is true in the DOS product as well as the Xenix version. RightWriter is only integrated with WordPerfect under DOS.) In order to use the product to check a document, you must execute the following steps:

- Exit your word processor and go to the Xenix prompt.
- Load RightWriter with the proper file.
- Run RightWriter.
- After the operation is complete, exit RightWriter and go to the Xenix prompt.
- Load your word processor.
- Load the annotated file (the original filename with a .out extension).

- Read the comments and *manually* make the corrections. Yes, manually! There is no way to automatically incorporate RightWriter suggestions.
- Save the file with the corrections.
- Exit the word processor and go to the Xenix prompt.
- Run a command which eliminates the RightWriter annotations.

NO PROTECTION FROM OPERATING SYSTEM. You can't get away from it! RightWriter does not work in any of the graphical environments (Motif, PM, Windows) which protect users from the cold reality of the operating system. Until it does, and until the process is as integrated as a spelling corrector, it will not be universally useful. And that's too bad, because the functionality is good and the need for the product definitely exists.

UNJUST STATEMENTS? We have to confess that we have not used the competitive products mentioned at the beginning of this article. Thus, we can't say whether any of the competitors does a better job of integration than does RightSoft. We hope they do. But they don't have products available for the Unix market yet. We'll be sure to keep a lookout for their arrival.

—R. Marshak

Upcoming Conferences from Patricia Seybold's Office Computing Group

The Second Annual

Executive UniForum Symposium

The Applications Development Environment of the 1990s: Can Unix Set the Innovation Agenda?

May 22, 23 & 24, 1990, Four Seasons Biltmore Resort Hotel, Santa Barbara, California

Sponsored by Patricia Seybold's Office Computing Group, UniForum, and X/Open

"It synthesized the relationships between all the major players in the Unix industry."

—1989 Executive UniForum Symposium Attendee

The theme of this year's Executive UniForum Symposium is the Unix software and applications development environment. If Unix is to become a commercially viable operating system, it must be able to support, attract, and nurture the next generation of applications and development tools. Is Unix up to the task?

This conference will present the views of industry leaders, including hardware, software, and networking vendors, as well as the views of commercial users. It will also provide a forum to spotlight the next generation of Unix applications.

Registration Fee:

- $895 for all registrations *paid by* February 12, 1990.
- $1095 for all registrations *paid after* February 12, 1990.

The Fourth Annual

Patricia Seybold's Technology Forum

Distributed Network Computing and Object-Oriented Environments: Pillars for the 1990s

April 2, 3, 4, 1990 , Cambridge Marriott Hotel, Cambridge, Massachusetts

"As with all Seybold Forums, I was 'quantum leaped' into the future of new technology."

—1989 Technology Forum Attendee

Two emerging technologies will become the supporting pillars of the application architectures of the next decade: *distributed network computing* and *object orientation.*

This year's Technology Forum explores each of these critical technology sets separately in twin tracks, exploring current options and solutions. Throughout the conference, attendees will have the option of moving back and forth between tracks, or staying on a single track from start to finish. In joint sessions at the opening and closing of the conference, we will examine the intersection of these two technologies, as well as the future directions for each.

Registration Fee:

- $895 for Office Computing Group Subscribers
- $1095 for non-subscribers

For more information about either of these conferences, call Deborah Hay at 1-800-826-2424 (in Massachusetts, call 617-742-5200).

Patricia Seybold's Computer Industry Reports

ORDER FORM

Please start my subscription to:

		U.S.A.	Canada	Foreign
☐ *Patricia Seybold's Office Computing Report*	12 issues per year	$385	$397	$409
☐ *Patricia Seybold's UNIX in the Office*	12 issues per year	$495	$507	$519
☐ *Patricia Seybold's Network Monitor*	12 issues per year	$495	$507	$519
☐ *P.S. postscript on information technology*	12 issues & tapes per year	$295	$307	$319
☐ *P.S. postscript on information technology*	12 issues per year	$ 95	$107	$119

Please send me a sample of: ☐ *Office Computing Report* ☐ *Network Monitor* ☐ *UNIX in the Office* ☐ *P.S. postscript on information technology*

Please send me information on: ☐ Consulting ☐ Special Reports ☐ Conferences

☐ **My check for $_____ is enclosed.** ☐ **Please bill me.** ☐ **Please charge my subscription to: Mastercard/Visa/American Express** (circle one)

Name: _____ Title: _____

Company Name: _____ Dept.: _____

Address: _____

City, State, Zip code: _____

Country: _____ Bus. Tel. No.: _____

Card #: _____

Exp. Date: _____

Signature: _____

Checks from Canada and elsewhere outside the United States should be made payable in U.S. dollars. You may transfer funds directly to our bank: Shawmut Bank of Boston, State Street Branch, Boston, MA 02109, into the account of Patricia Seybold's Office Computing Group, account number 20-093-118-6. Please be sure to identify the name of the subscriber and nature of the order if funds are transferred bank-to-bank.

IU-0290

Send to: Patricia Seybold's Office Computing Group: 148 State Street, Boston MA 02109; FAX: 1-617-742-1028; MCI Mail: PSOCG
To order by phone: call (617) 742-5200

Topics covered in Patricia Seybold's Computer Industry Reports in 1989/1990:

Back issues are available, call (617) 742-5200 for more information.

Office Computing Report

1989—Volume 12

#	Date	Title
5	May	The LAN Office Emerges—WordPerfect Enters OA Race
6	June	Objects at Floodtide—Object Orientation Permeates New Development
7	July	DECwrite—Advanced Applications under DECwindows, Part I
8	Aug.	DECdecision—Advanced Applications under DECwindows, Part II
9	Sept.	Tools for ESS Development—The Vendor Offerings
10	Oct.	OfficeVision—IBM Prepares for the Next Generation
11	Nov.	Cease Fire!—PC Wars Confuse Customers Needlessly
12	Dec.	IBM's Data Management Design—The Future Is Distributed and Relational

1990—Volume 13

#	Date	Title
1	Jan.	Office Redefined—A New Model for a New Decade

UNIX in the Office

1989—Volume 4

#	Date	Title
5	May	Sybase—The Next Generation
6	June	Thoroughly Modern Unix—The Shape of New Operating Systems
7	July	Surveying the Users—Taking the Pulse of Commercial Unix
8	Aug.	Graphical User Interfaces—A Developer's Quandary
9	Sept.	The Double Standard—DOS under Unix Moves to the Next Generation
10	Oct.	Reality Check—Commercial Users Speak Out on Barriers to Unix Acceptance
11	Nov.	To Lead or Follow?—Which Route Have Uniplex, Applix, and Quadratron Chosen?
12	Dec.	Targon Office—Nixdorf's Object-oriented Platform for Application Integration

1990—Volume 5

#	Date	Title
1	Jan.	Oracle's RDBMS—The Product behind the Marketing

Network Monitor

1989—Volume 4

#	Date	Title
5	May	NetView—IBM's Enterprise-Wide Manager
6	June	Apple's Networking Strategy—Bringing the World to the Desktop
7	July	Distributed Object Management—Networks Made Easy... And More
8	Aug.	Proteon—Platforms for the '90s
9	Sept.	AT&T's Accumaster Integrator—OSI Network Management Leader
10	Oct.	Digital's EMA—A Fresh Perspective on Managing Multivendor, Distributed Systems
11	Nov.	A New Calling—IBM's Strategy to Integrate Voice and Data
12	Dec.	TCP/IP Network Management—What It Was, What It Is, What It Will Be

1990—Volume 5

#	Date	Title
1	Jan.	Distributed Computing Environment—Interoperability Now!