From Operating Systems to Cooperative Operating Environments

Michel Gien

The open systems environment has had a major impact on the value of general purpose computers. UNIX dominates the workstations environment. The server market has moved rapidly toward UNIX and UNIX on PC hardware is becoming common place. However impressive this expansion has been the enterprise requirements are larger than those represented by traditional general purpose computer configurations.

The enterprise needs often include the ability to respond to real time situations, to perform high speed calculations or massive numbers of data base inquiries, and to be on line and always available.

Addressing the expanded enterprise needs requires a change in the way we think of operating systems. Operating systems grew out of the need to manage a processor and its resources (peripherals). With multi-processor, multi-computer configurations a whole environment needs managing. Processors need to cooperate with other processors and resources of any processor should be available transparently to any application or utility. The term "Cooperative Operating Environment" better expresses this notion than the terms enhanced or multi-processor operating system.

In order to achieve a cooperative operating environment in a non shared or distributed memory configuration, new kernel architectures have been developed. Three major concepts are the foundation of the architecture:

- Microkernel with transparent distribution support
- Modular operating system servers
- System level communication architecture.

Taking CHORUS/MiX and SYSTEM V.4 as concrete examples, available today on the market, the paper expands on such a new operating system architecture and services and show how it fulfills the requirements for a cooperative operating environment, needed to progress Open Systems further.

1. Expanding Open Systems With Advanced OS Technology

Fostering innovations is one most important yet overlooked benefits of the UNIX open systems environment. A prime example of UNIX innovation at work is how advanced operating system architectures are being utilized to support new hardware and software technology and expand the application address of UNIX.

Innovation results from the fact that UNIX is more than a binary operating system confined to one company's resources and ideas. UNIX is a technology that is provided in source form to the entire computer industry. It achieves compatibility between vendors through a detailed and tested set of application program interfaces (APIs) implemented by UNIX System Laboratories (USL) in a reference technology source. It does not confine the implementation of computer systems to that reference source. The source is used by computer manufacturers and Valued Added Resellers operating system companies, such as Chorus Systems, to add value and innovation. They extend the application and functional range of UNIX to address the expanding enterprise needs without sacrificing compatibility.

From an operating system point of view these expanded needs include the ability to respond to real time situations, to perform high speed calculations or massive numbers of data base inquiries, and to be on line and always available. The UNIX SVR4 environment now addresses these expanded needs. In short, because of operating system advances, the open systems environment of UNIX now covers the enterprise's entire range of operating system requirements.

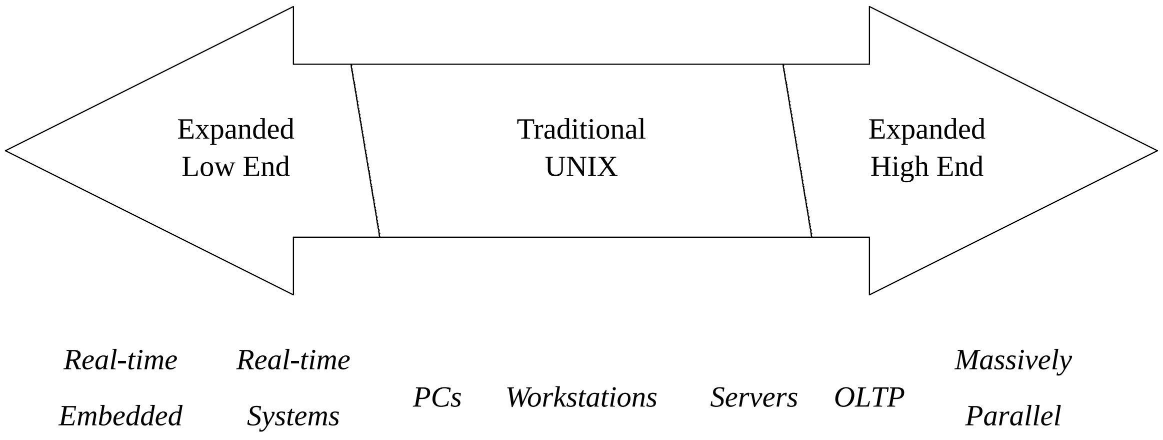


Figure 1. − Expanded Enterprise Requirements

1.1 Real Time

Many enterprises are using computers for factory automation, process control, and/or as a critical element in their products. In fact, computers have been applied in real time situations since the 1950's. With the introduction of SVR4 many of these applications can utilize UNIX in its traditional form. However, many real time applications can be more demanding in terms of configuration, determinism, and responsiveness, requiring an expanded operating system approach. The advantages of having the real time operating system the same as the rest of the enterprise computer environment are numerous. Training is much simpler and less expensive since there is only one environment to learn. With UNIX there is a large pool of skilled and trained programmers. With one environment there is less confusion between the operation of one set of equipment and another. Applications and development can be shared between one area and another. And, the likelihood of finding existing applications from third party vendors is high in the UNIX community.

1.2 High Performance Through Parallel Processing

Processor costs have seen a dramatic decrease in price, especially the high volume merchant chips. As a result the cost per MIP has plunged into the "almost free" range. Specialized high performance processors (and computers) on the other hand have not enjoyed the same degree of economy. Tapping multiple merchant processors working in parallel to increase application performance will result in the most economical approach to high performance computing, often costing millions of dollars less than traditional super computer approaches.

High performance utilizing parallel processors have been associated with scientific/engineering applications. Commercial applications can benefit equally if not more from this approach, especially data base applications. Symmetric multi-processing, where memory is shared and used as a communication medium between processors, is the most straight-forward method of providing multi-processor operating system support. However, this method has limits in the scalability of the number of processors, and non shared memory arrangements make up the massively parallel configurations. Typical scientific applications benefiting from massively parallel configurations include simulations of all types, finite element analysis, image processing, neural networks, statistical methods, etc. Data base applications are relatively easy to parallelize and benefit greatly from multi-computer configurations not only for increase in computation power but also in I/O bandwidth from multiple disk and peripheral support.

Often these parallel machines have a collection of processors and associated memory without any other peripheral except high performance communication channels to the rest of the system. Peripherals are community resources with their own processors and memory. The challenge is to have all of these pieces interact very efficiently and as a single unit acting like any standard UNIX system. Communication is often the bottleneck in performance, and both hardware and software technology are critical to obtaining the desired benefits. Compatibility with UNIX allows to use such systems beyond the niche of strictly performance related needs.

1.3 High Availability

Computers have become essential to the operation of most enterprises. Failure of computer systems in some operations can have disastrous consequences. The most drastic example in recent times is the AT&T failure in 1991 which shut down phone communications in the eastern US for a few hours.

Yet the desire to automate and implement computers into operations is growing because of huge returns. Like the parallel processing environment, multi-processors and multi-computers can be used to increase the availability of processing operations through dynamic backup or mirroring, reconfigurability, and process migration. Traditionally this functionality has required expensive specialized hardware. If software techniques can be applied to the high availability requirements the cost of the system can be a fraction of a hardware approach especially if the software permits the use of commodity hardware building blocks.

The expanded enterprise requirements can be looked upon as the leading indicators of the general purpose computer requirements in the future. Workstations and servers will gain more performance by using multi-processor configurations. They will perform tasks that require higher availability and more real time responsiveness. These expanded needs are merely a preview of the future for all of computing.

2. Cooperative Operating Environment

In order to easily address the expanded requirements of the enterprise, operating system technology, specifically the kernel, needs to be expanded. This means expanding real time attributes such as preemptive scheduling and developing structures and protocols for managing many closely or loosely coupled processors at the operating system kernel level. Incorporating cooperating elements within a multi-processor distributed environment results in an environment where processors cooperate with other processors, and where resources of any processor can be made available transparently to any application program.

USL and Chorus Systems have teamed together to make this happen. USL provides an advanced shared memory multi-processor version of SVR4.0 and is enhancing those facilities in the next release, UNIX/SVR4.3 (ES/MP). In the shared memory model, UNIX spreads the execution of applications and system services across multiple processors sharing the same memory. Applications written on a mono-processor work unchanged and gain the benefit of more throughput capacity of the system. Applications written for multi-processors can take advantage of the extra processing power directly and improve their execution performance accordingly. Also, the current SVR 4.1 version, has priority scheduling as an option which can be used for many real time situations.

CHORUS/MiX extends UNIX as a non shared memory multi-processor operating system with preemptive real time scheduling, fully compatible with SVR4. Because of it's modular and fully distributed architecture, CHORUS/MiX is able to unite multi-processors or multi-computers into a single unit or cluster. Such an operating system architecture is referred to as a Cooperative Operating System Kernel or Coop kernel for short.

3. The Coop Kernel Architecture

In order to achieve a cooperative operating environment in a non shared or distributed memory configuration, a new kernel architecture was developed, a Coop kernel architecture. Three simple but major concepts are at the foundation of this operating system architecture:

- Microkernel with transparent distribution support
- Modular distributable operating system servers
- System level communication architecture

3.1 Microkernel With Transparent Distribution Support

The microkernel in CHORUS/MiX implements a minimum set of generic operating system services necessary to support one processor with its local memory, or a group of shared memory processors: management of the physical processor and memory, a real time preemptive scheduler, a (virtual) memory manager, and a facility to communicate with other microkernels. Each microkernel cooperates with other microkernels to form a unified virtual machine, transparently spread over a set of closely or loosely coupled processors.

All traditional operating system kernel services such as file services, device management, UNIX communication functions (e.g., streams) and even device drivers are implemented outside of the microkernel, as separate programs running in independent cooperating servers.

The communication services provided by the microkernel allow servers to cooperate without needing to know where they are currently being executed in a distributed or cluster configuration. Operating system configuration decisions in such an environment (as well as its reconfiguration), can now be performed dynamically while the operating system is running as opposed to when it is designed.

Such a microkernel can be compared to the basic building blocks in Lego. Before Lego, play blocks had no interconnect facilities and could be used to build only limited structures. Lego, because of its very simple interconnect facilities, appears unlimited in building complex structures. The CHORUS microkernel with its transparent distribution support has a similar impact on computer systems designs.

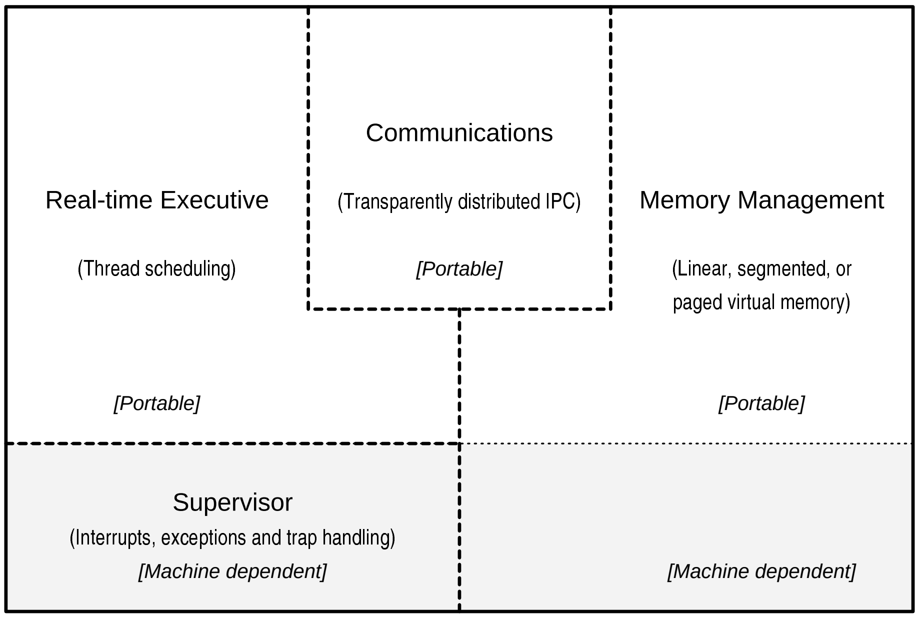


Figure 2. − Microkernel with Transparent Distribution Support

3.2 Modular Distributed Operating System Servers

A set of cooperating distributed microkernels is only one element in building up the Coop kernel architecture. The second most important concept is a set of encapsulated modular operating system servers (e.g., file servers or stream servers) which sit on top of the microkernel and cooperate with each others in the context of a "Subsystem" by means of the microkernel communication facilities, using messages or Remote Procedure Calls (RPC).

Such operating system servers can be transparently distributed across a cluster of computers. Protocols for exchanging information and commands are not affected by whether servers are running in the same computer or in different ones. Applications on nodes without disks easily interact with files on those nodes which do have disks as if they were local. Some nodes might provide special facilities or devices which every node can access as if they were local. This allows to unify a collection of computers into a single operating system image.

A second benefit of modular operating system servers is the ability to easily replace servers with new or enhanced servers providing improved or expanded services. Since communication between servers is explicit through messages or RPC (as opposed to shared memory), other servers are much less impacted by the installation process.

CHORUS/MiX servers include the Process Manager (PM), the File (Object) Manager (FM), the Streams Manager (StM), and the SV IPC Manager IPCM). In addition device drivers are treated as servers. All application system calls go through the Process Manager which exhibits a UNIX system personality and provides for SVR4 binary compatibility.

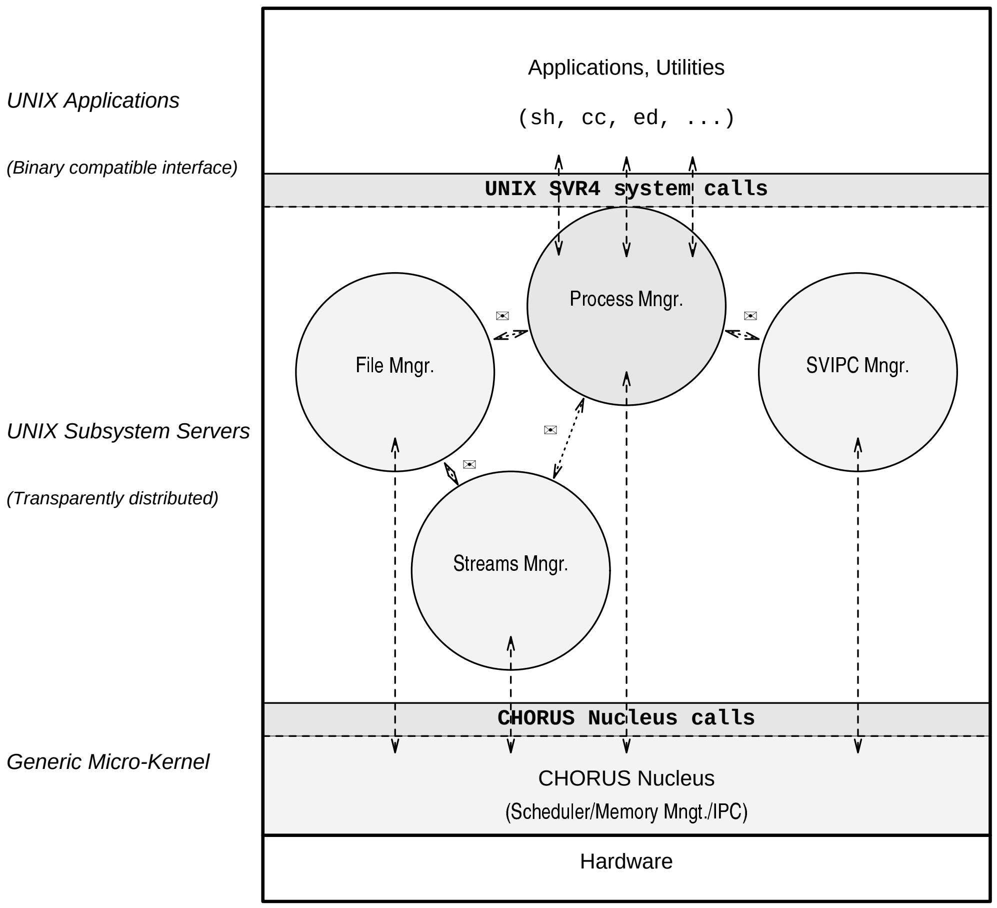


Figure 3. − Modular Distributed Operating System Servers

3.3 System Level Communications Architecture

The third significant element is the system level communications architecture. Like the micro-kernel, the inter-process communication architecture is simple yet powerful. As stated before, the communications architecture is message based. The basic message format is a very low overhead structure that can be easily enriched by the system builder. For example, there are no security checks or typing overhead although these services can be easily added. Minimum overhead is what is needed in a massively parallel environment.

At the same time the structure is quite powerful. Messages need addresses in order to know where to go. Processes or servers are indirectly addressed through the concept of Ports. Ports act as a kind of mail box. Ports are attached to UNIX processes and collect messages (queues them) which are then passed onto the process. Ports can be moved dynamically to other processes. This is useful if a new process is replacing an old one dynamically saving down time if the process happens to be an operating service such as a file manager.

Port names are global (virtual) addresses, i.e., each port in the community of computers has a unique name, and names do not change when ports migrate from one location to another. Messages sent by programs to a given port name all go to the same location, thus facilitating programming of communications between processes in a multi-computer environment. Management of port names is done in a distributed manner, thus suppressing single points of failure, and allowing to achieve high availability more easily.

The port concept has been extended to the concept of a port group which can hold the addresses of regular ports. Among other things port groups enable messages to be broadcasted to several ports and thus several servers at once. Messages sent to a port group are copied and sent out to all the addresses stored in that group. This enables creating redundancy or backup.

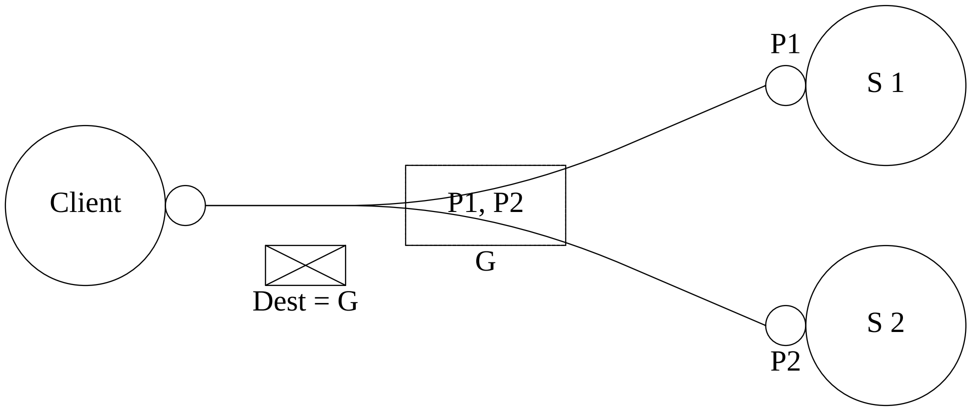


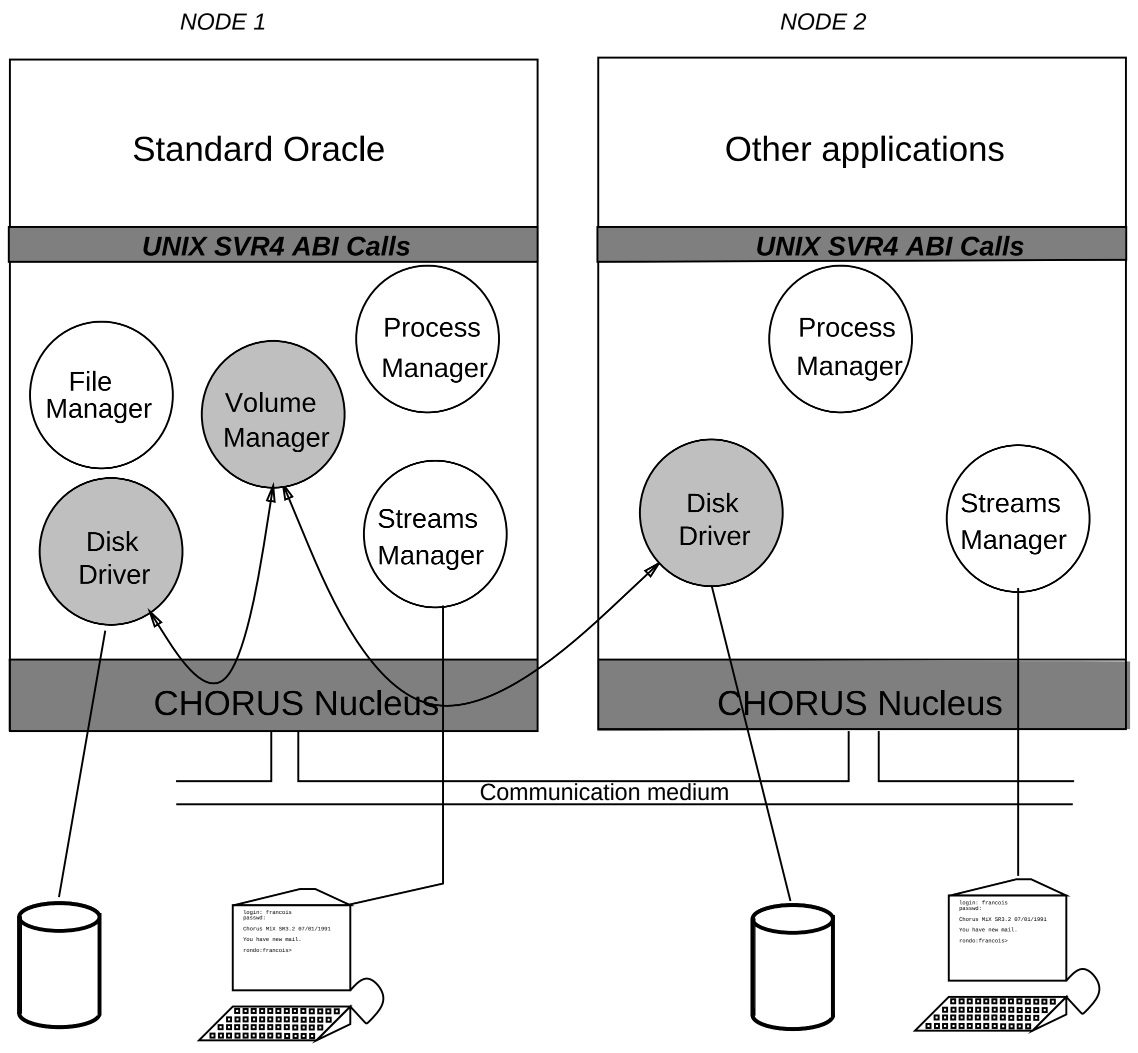
Figure 4. − Port Groups for Redundancy

4. Examples

4.1 High Availability

Chorus has taken Veritas Volume Manager and Oracle database, and combined them with CHORUS/MiX. The volume manager has mirroring capability that allows a configuration with two disks and two disk controllers to provide data base backup transparently to the Oracle data base. Normally those two controllers have to be on the same computer. The CHORUS/MiX configuration allows mirroring to take place between two separate computers connected with a LAN.

Because the disk driver server uses message based communications, the driver can be on the second computer and yet appear as if it were local to the Veritas Volume manager. In other words, the volume manager sees the two disk controllers as if they were on the same computer when in fact they are on separate computers. Now the mirroring functionality can take place on even two simple PCs with low cost controllers instead of more expensive disk controllers utilizing expensive buses. The specialized hardware, such as complex controllers, often doubles the system cost whether it is on PC or large mainframes. Standard computer equipment can thus be applied to high availability applications.



```
login: francois
passwd:

Chorus MiX SR3.2 07/01/1991

You have new mail.

rondo:francois>
```

```
login: francois
passwd:

Chorus MiX SR3.2 07/01/1991

You have new mail.

rondo:francois>
```

Figure 5. − Disk Mirroring between PCs

4.2 Parallel Processing

CHORUS/MiX can also be used to increase the storage size and performance using a similar approach to that described above. The Volume manager implements disk stripping. This technique enables a file to bridge across multiple disks and disk controllers as if they were one unit. Using CHORUS/MiX, stripping can take place across multiple computers as well as disk controllers attached to just one computer. Using Parallel Oracle all computers as well as disks can be applied in parallel to the data base application with major performance improvement. The same modular disk controller used in the high availability example enables the parallel operation. In fact the two examples can be combined.

In the past, the alternative to gain more performance has been to replace the existing computer with a more powerful computer if one even exists (if it doesn’t then the user is stuck with poor performance). This approach is much more disruptive and costly than simply adding another low cost computer to the application. The Coop architecture means there is always a path for performance improvement.

4.3 Dynamic Reconfiguration

If a new version of the volume manager has to be installed this can be done without taking the system down. This is because the ports from the old volume manager can be migrated to the new version dynamically. Upgrading the system software can constitute a major element in a mission critical system down time. With the Coop kernel architecture this time can be limited to a minimum. Chorus demonstrated to the European Space agency how the European Space Station, Columbus, could refresh and upgrade significant elements in the system software without down time.

4.4 Real Time

There are many real time examples. One of the most interesting is how Alcatel, the world's largest supplier of business PBXs, is applying CHORUS/MiX. Traditionally a PBX employs a small real time executive to manage switching of phones. Modern PBX have other functionalities besides switching telephones such as voice mail and directory data bases. For large companies and institutions these data bases can be quite large. Normally a separate computer running UNIX was used to provide the expanded services. Now the two functions, real time switching and data base retrieval, can be combined. This means that smaller configurations can be built with expanded functionality passing cost savings on to the customers. Also, this architecture gracefully expands to huge multi-computer configurations with high availability capability without rewriting any of the real time or general purpose applications. In other words, a complete PBX product line can be built out of common software parts.

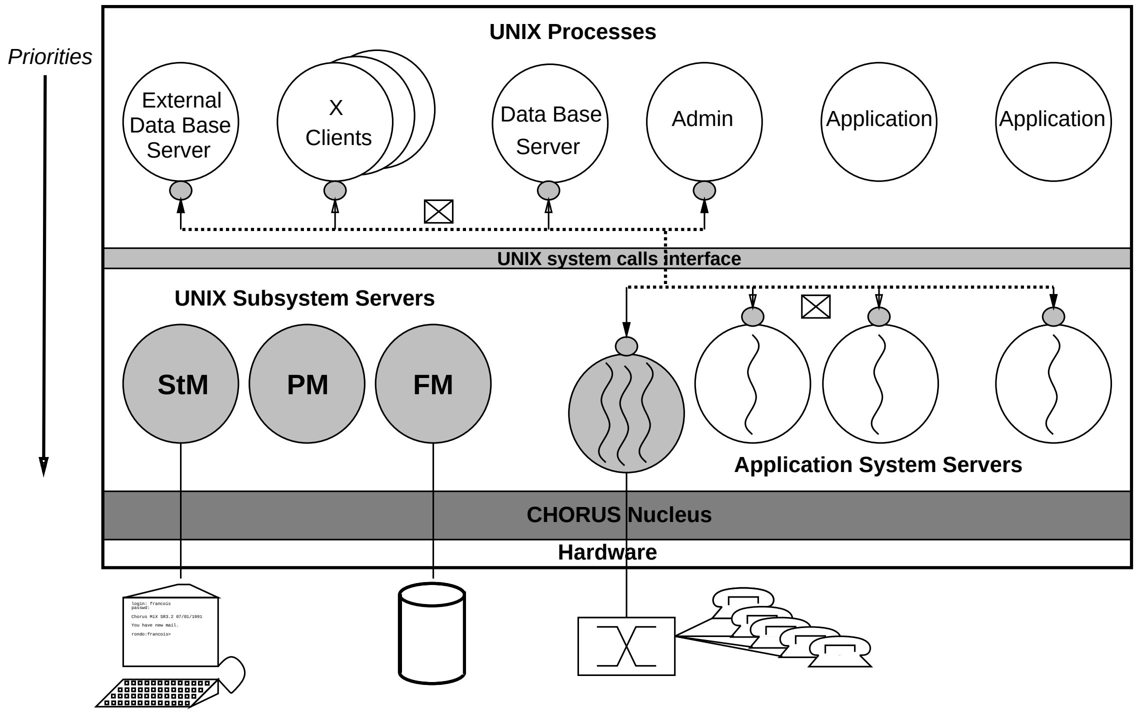


Figure 6. − Real-time and UNIX in a PBX

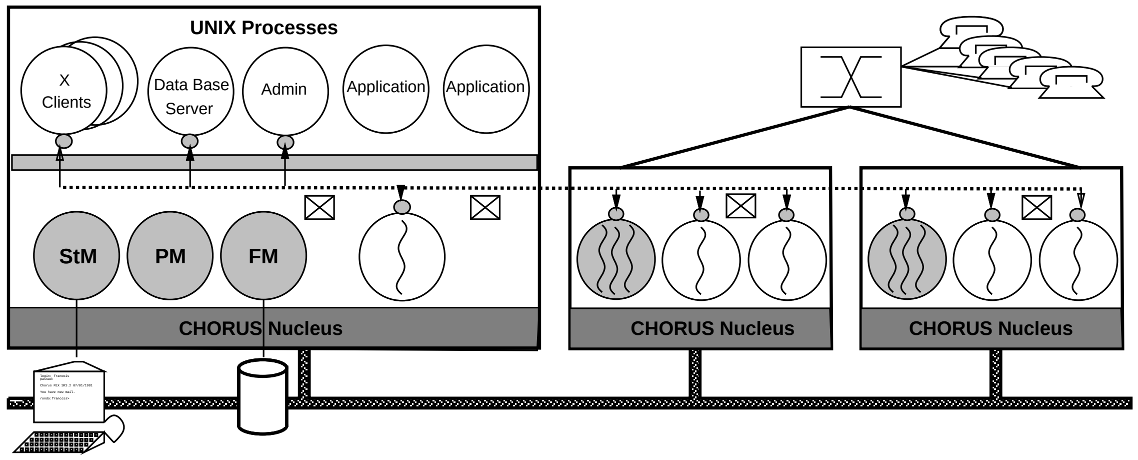


Figure 7. − Scalable PBX system

5. Compatibility with Today’s Products and Tomorrow’s Technology

It’s not enough to have good products today if those products don’t leverage the technology advances that will provide improved operations and productivity. Conversely, the advances need to be compatible with today’s products. UI, USL and Chorus are committed to improving the technology in a compatible fashion. They also have the support of the European Economic Community.

In 1992, a consortium of industrial partners, Olivetti, Siemens, Thomson, and Alcatel have teamed with the EEC, UI, USL and Chorus to invest $16 million over three years into advancing operating system technology. The work includes insuring CHORUS/MiX tracks the UNIX SVR4 releases and exploring Object Oriented technologies as it relates to operating system issues. The project is called OUVERTURE.

6. Conclusion

The cooperative computing environment is a significant step in the evolution of open systems. It provides a compatible path for the emerging hardware configurations, in particular multi-processor and multi-computers.

It extends the range of the Open systems environment to complete coverage of enterprises needs. It provides a path for continued advancement of the technology in a compatible fashion.