VM/ESA: A single system for centralized and distributed computing

by W. T. Fischofer

The rapid evolution of distributed and personal systems in recent years has not diminished the importance of centralized computing. Today, systems at all levels need to operate in networked configurations to allow users and applications to access and manipulate data from anywhere with full integrity and optimal performance. Virtual Machine/Enterprise Systems Architecture™ (VM/ESA™) satisfies this requirement as a single VM product that has been designed for both centralized and distributed computing. This essay describes how VM/ESA builds on IBM's reputation for virtual machine performance, function, and flexibility to form an ideal solution base for the 1990s.

In 1979, the *IBM Systems Journal* devoted an entire issue to virtual machine (VM) systems.[1] The intervening years have seen dramatic growth in the processing and communications capabilities of computing systems, along with a corresponding evolution of IBM's VM product. Twelve years ago this product, known as the Virtual Machine Facility/370 (VM/370), had just begun its transformation into a strategic offering. This essay introduces the Virtual Machine/Enterprise Systems Architecture™ (VM/ESA™) that completes the transformation by recognizing the increasingly vital role that virtual machine systems play in more than 20 000 customer establishments to more than 6 000 000 users worldwide.

VM/ESA represents more than just a name change for the VM product. First and foremost, it unites in a single product the capabilities formerly found in the three main VM "dialects" introduced in the 1980s: Virtual Machine/System Product (VM/SP), Virtual Machine/System Product High Performance Option (VM/SP HPO), and Virtual Machine/Extended Architecture™ System Product (VM/XA™ SP). Beyond this, VM/ESA provides significant new performance and functional capabilities not found in any previous VM product offering and forms a unified base for further evolution in the 1990s.

The provision of a single VM product has been a stated goal within IBM for some years, both to simplify system planning and deployment for customers and to help focus the VM development investment. To understand why technology has led to a single VM product offering requires an understanding of the system and technology forces that are now converging at an increasingly rapid pace. A brief review of the concepts and history of VM follows.

VM concepts and history

The history of IBM's VM operating system in many ways mirrors the evolution of the computing industry as a whole over the past quarter century. Beginning in 1964, when time sharing and the interactive use of computers were still relatively new concepts, a group of researchers at the IBM Cambridge Scientific Center (then known as the IBM Systems Research and Development Center) near Boston, Massachusetts, pioneered what has become known as the *virtual machine model*.[2] The basic idea of this model was to construct a system where issues of resource management could be separated from those of user management. The result was a formal separation of these two elements into the two primary components of the VM operating system: the control program and the Conversational Monitor System.[3]

Control program. The control program (CP) is responsible for resource management. It operates the machine hardware and multiplexes the physical resources of the computing system into multiple logical entities called *virtual machines*, each of which is an idealized simulation of a computer dedicated to the servicing of a single user or (in the case of a server) a single application. This structure provides VM with several advantages:

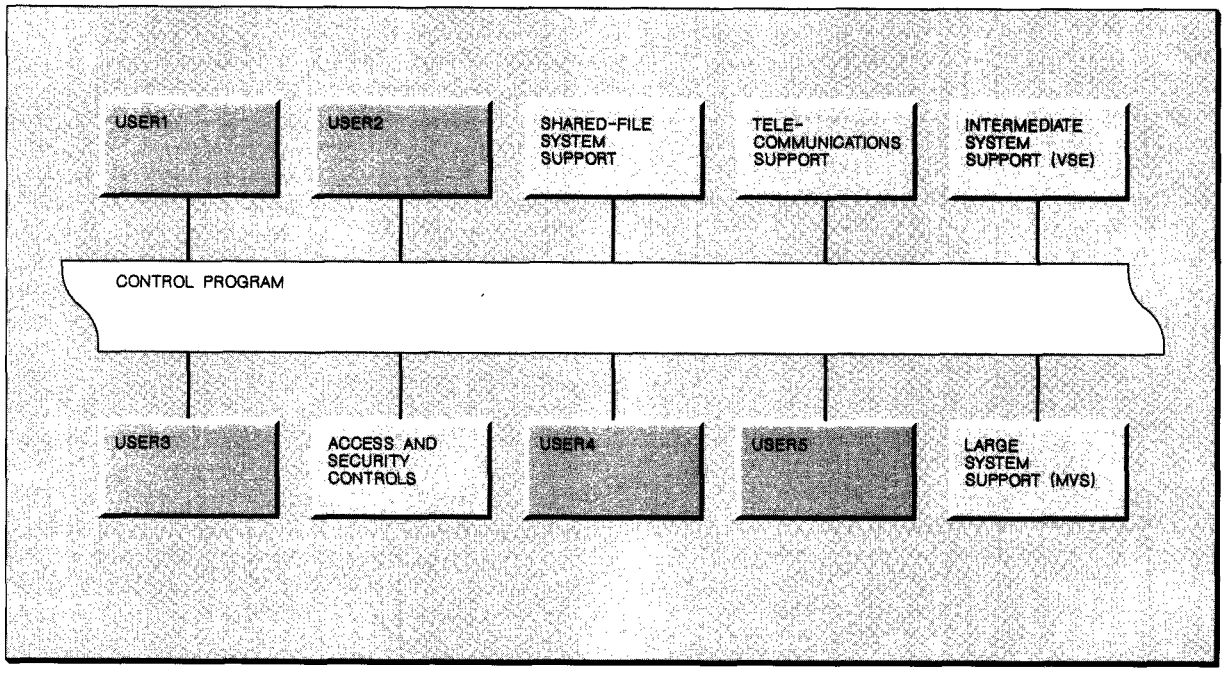
- Individual users running applications in their own virtual machines are isolated from each other, which provides a high degree of system security and reliability. In the absence of explicit authorization, events inside a particular virtual machine cannot compromise the integrity or operation of any other virtual machine.
- Each virtual machine may potentially run a different operating system. Moreover, the individual operating systems can fully support a single user without any focus on multiuser resource management issues.
- The ability of CP to be a hypervisor for other operating systems (including itself) and to create virtual processor, memory, and input/output configurations allows it to provide a flexible system test and migration platform. This same capability also allows it to support multiple production guest systems such as Virtual Storage Extended (VSE), Multiple Virtual Storage (MVS), or Advanced Interactive Executive (AIX®) along with interactive VM user applications.
- Because the basic architecture of VM encourages the use of communications rather than shared-memory interfaces for intervirtual machine interaction, the extension from a centralized to a distributed computing environment is both natural and largely transparent to applications and users.

VM as a software local area network. Although the term would be unfamiliar to the scientists of the IBM Cambridge Scientific Center of the 1960s, what they, in effect, had invented was the surprisingly useful and durable concept of a software local area network. Instead of physical workstations and wires to connect them, VM—through the hypervisor capabilities of CP—realizes the software equivalent of a local area network. This can be seen in Figure 1. The software local area network concept is the key to the strength and flexibility of VM, but it is also the source of its traditional weakness. Historically, the interuser isolation between virtual machines was simply too good. While isolating each user may be useful for security and reliability, it can hamper productivity by making it difficult for users to share programs and data in an effective manner. We discuss later how VM/ESA has overcome this traditional weakness while still retaining the significant benefits of the software local area network system structure. For now, it is sufficient to note that CP provides each user with a personal computer.

More than just a hypervisor. The essential characteristic of a hypervisor is that the interface definition it provides is (except for timing differences) the same as the interface definition provided by the real hardware. This is what makes it possible for CP to run other operating systems, and in this sense CP is similar to other hypervisors such as the Processor Resource/Systems Manager™ (PR/SM™) hardware feature available on IBM ES/9000™ processors.[4] However, part of the unique value of VM is that CP is more than just a hypervisor.

Early in the history of VM it became obvious that CP could do more than simply provide virtual copies of the real hardware.[5] Indeed, early experience with the virtual machine model showed that a naive approach to virtualizing hardware resulted in less than adequate performance. The reason that CP is more than just a hypervisor is that VM consists of more than just the control program.

Figure 1 VM as a software local area network



Conversational Monitor System. Similar to workstations on a real local area network, the virtual machines that comprise the software local area network created by CP require their own operating systems. The Conversational Monitor System (CMS), as the user management component of VM, is a single-user operating system specifically designed to run in a virtual machine created and managed by CP.

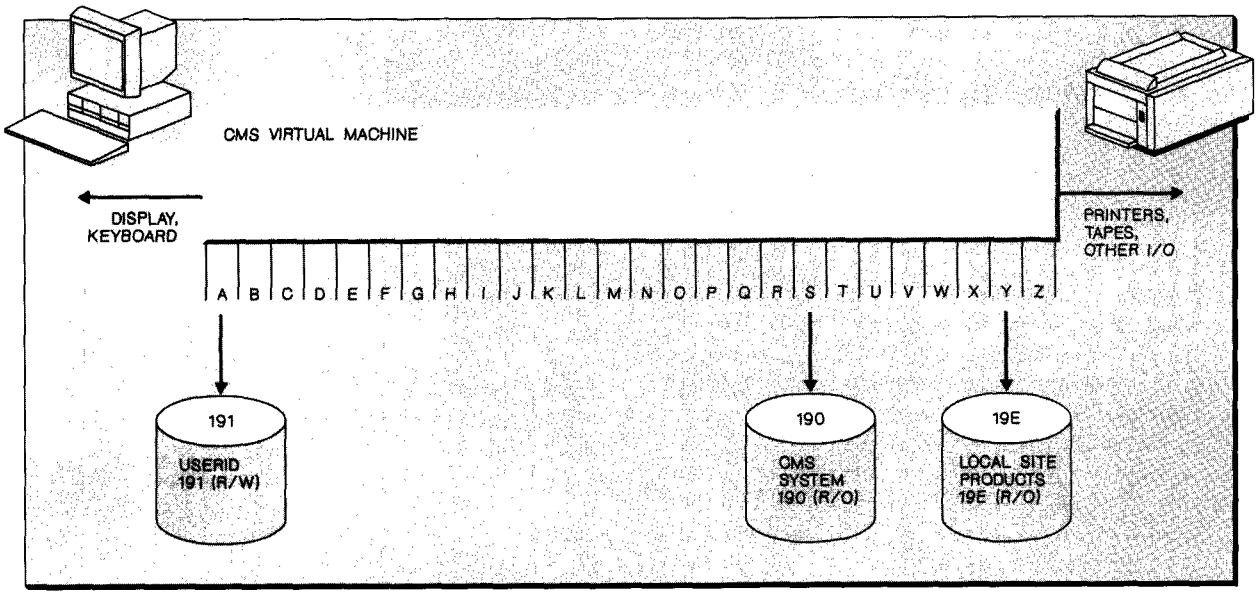
Key to VM is the symbiotic relationship between CP and CMS. While CMS originally was able to run on real hardware, as early as 1972 it gave up this ability in order to obtain better performance when running in a virtual machine. In exchange, CP has been optimized for the efficient support of large numbers of CMS virtual machines, and is itself installed and serviced using CMS. The result is that while CP can support a handful of large guest systems such as MVS and several dozen intermediate-size guest systems such as VSE (Virtual Storage Extended), it can support thousands of CMS users on large System/390™ systems.

Minidisk file system. CMS is a single-task operating system designed to support application development, execution, and data management on behalf of a single user. The central feature of CMS is its file system and the commands that operate on it. CMS files have three-part names that consist of a *filename*, *filetype*, and *filemode*.[6] Filenames and filetypes each consist of up to eight characters, while filemodes consist of a letter followed by a number. The choice of filetype is largely by convention and is used to describe the contents of the file. For example, the filetype FORTRAN is used to describe a FORTRAN source program, while the filetype EXEC is used to describe a file that may be processed by an interpreter. While the filename and filetype name the file itself, the filemode is used to describe where the file is located.

Historically, the CMS file system was implemented on portions of real disks called *minidisks*, with each CMS virtual machine directly managing the format and contents of the minidisks attached to it. This is very similar to "real" personal computers that manage their attached hard disks, and since a CMS virtual machine is a personal computer, it is best understood in this context.

Each user of a VM system has a user identification, or *userid*, which provides a unique identification to CP. The description of the virtual ma-

Figure 2 CMS virtual machine abstract view



chine configuration associated with that userid (including its minidisks) is maintained by a system administrator in a special system file called the *user directory*. For a given userid, minidisks have virtual addresses that (following the virtual machine model) are analogous to the device numbers used by the real hardware. A CMS filemode can thus be regarded as a "slot" into which a particular minidisk can be inserted to allow it to be read or written by the CMS virtual machine. The CMS ACCESS command is used to accomplish this association of minidisk to filemode. This can be seen in Figure 2, which shows an abstract view of a typical CMS virtual machine. By convention, the minidisk containing a user's primary personal storage is located at virtual address 191 and is accessed at filemode A. Similarly, the CMS system disk by convention is at address 190 and accessed at filemode S, while the local site product disk is at address 19E and accessed at filemode Y. Both the system and product minidisks are read-only disks and are shared among all CMS users for reasons of system efficiency. While the S disk contains programs that are part of CMS itself, the Y disk is used to hold locally-installed program products, such as compilers and utilities, which are accessible to all users but are not themselves part of the CMS operating system. The remaining filemodes are available for use in accessing other minidisks as needed. A CMS user is thus able to work with up to 24 minidisks, plus the two reserved system disks, at one time.

Minidisk limitations. While conceptually simple and highly efficient in terms of performance, the CMS minidisk file system has several well-known limitations. First, minidisk space must be permanently allocated to a user independent of whether it is actually filled with files. This can result in substantial waste when multiplied by the thousands of minidisks typically found in large systems. Moreover, minidisks require a relatively high level of support skills to administer effectively, which makes minidisks difficult to manage in smaller, less sophisticated, installations. Second, the nonhierarchical structure of minidisks makes it difficult to organize information conveniently once a minidisk grows to contain more than a few hundred files. Third, minidisks essentially represent private file storage for the CMS user. Sharing files residing on minidisks among CMS users is difficult and compares poorly with the sharing facilities provided by other operating systems. Finally, minidisks are local to a specific system. As a result, accessing files stored on minidisks from distributed and remote systems is awkward and requires that specific knowledge about the minidisk be available.

The solution to these problems was addressed by evolutions of CMS in the 1980s.

VM business roles

In the 1970s, VM was essentially a systems programming tool and migration support vehicle for other operating systems. In the 1980s, however, VM became increasingly vital to the day-to-day business of many customers. Associated with this trend, three distinct business roles for VM emerged: interactive computing, client/server computing, and guest systems support.

Interactive computing. Interactive computing is sometimes referred to as *personal productivity computing* and generally consists of an open-ended use of applications and data management tools in areas such as decision support, modeling and analysis, document preparation, and personal communications. This contrasts with *transaction processing*, which typically involves a preplanned and narrow set of interactions between the user and the system. The ability of VM to act as the host for large numbers of interactive CMS users in an economic manner, along with the development of key productivity applications such as the Professional Office System (PROFS®) and CADAM™, gave rise to this business role for VM in the 1980s. Evidence of the importance of this role is that many VM users are not even aware of using VM; they consider themselves to be "PROFS users" or "CADAM users."

Client/server computing. Since virtual machines are software entities, they can be easily configured and deployed at very low cost. As early as 1968 this advantage lead to the notion of dedicating a virtual machine to the running of a particular program rather than to a user.[7] A virtual machine configured in this manner is referred to as a *service virtual machine*, *service machine*, or simply *server*, because its function is to provide services to other VM users.

While the initial focus of servers in VM was to provide telecommunications support through products such as the Remote Spooling Communications Subsystem (RSCS) and the VM/Pass-Through Facility (PVM), the 1980s saw a tremendous growth in the use of servers. Today, servers are an essential component of VM systems, and even the smallest VM installations typically run a half-dozen or more of them.

Guest system support. Being able to run multiple operating systems on one computer provides a flexible and economical solution to problems of testing, migration, and consolidation. The hypervisor capabilities of VM have long made it attractive for this purpose. VM also provides several key added value items to the guest environment. A discussion of these items follows.

Virtualization. The ability of VM to virtualize hardware makes it possible to link guest systems together without the cost of actual hardware. For example, VM can provide a virtual channel-to-channel adapter to permit multisystem coupling between different guest systems. Similarly, VM can create virtual machine configurations that support multiprocessing even if the host computer is a uniprocessor. This can ease migration testing prior to the installation of a real multiprocessor.

Resource sharing. VM allows multiple guest systems to share devices and other resources to simplify configuration and maintenance. For example, multiple Virtual Storage Extended (VSE) guest systems can share the same system residence volume. VM can also dynamically partition resources such as expanded storage for shared use among multiple MVS guest systems. Similarly, multiple guest systems can share the same printers using the spooling capabilities of VM.

Augmentation. Because VM is more than just a hypervisor, it can provide guests with additional services that go beyond simple hardware support. For example, guests such as VSE can run in nonpaging mode to allow the CP paging subsystem to provide its virtual storage support requirements. Since CP provides support for specialized hardware such as expanded storage, guests may benefit from this hardware by running under VM even if they do not support this hardware themselves. Another example of guest augmentation is the Structured Query Language/Data System (SQL/DS) guest sharing function, which allows VSE guest systems to share an SQL/DS database with CMS users.[8]

VM evolution: From VM/370 to VM/ESA

Figure 3 shows a simplified VM genealogy. As can be seen, the 1980s were a period of divergence for the VM product—a period in which the familiar VM/370 product of the 1970s was replaced by three different mainline VM products: Virtual Machine/System Product, Virtual Machine/System Prod-

uct High Performance Option, and Virtual Machine/Extended Architecture. VM evolution in the 1980s was shaped by four fundamental forces:

- Centralized computing requirements
- Distributed computing requirements
- Reliability, availability, serviceability (RAS) requirements
- Migration requirements

These forces themselves derive from the business roles discussed previously. For example, the need to support guest systems mainly creates requirements for better centralized computing support, while the need to support client/server computing creates requirements for better distributed computing support. Interactive computing, in turn, encompasses both. Both functional areas require increasing reliability and share the need to ensure smooth customer migration as VM evolves.

A key element of the VM evolution strategy of the 1980s was the separation of the resource management and user management roles in VM. This separation meant that CP and CMS could evolve at their own pace and in ways most appropriate to their differing roles. In general, VM evolution in the 1980s saw CP changes driven by the requirements of centralized computing, while CMS changes were largely driven by the requirements of distributed computing. This difference in emphasis should not be surprising. CP, as the resource management component of VM, is most directly affected by the need to support larger and more sophisticated central processors. CMS, by contrast, better exploits the inherently distributed computing capabilities of the software local area network structure provided by CP. There are, of course, exceptions to this trend. Applications running on CMS need to be able to exploit the large address spaces and specialized computational features (such as vector processing) available on large central systems. Similarly, CP needs to provide the technology to allow CMS to realize its distributed computing potential. This intermixing is in keeping with the symbiotic relationship between CP and CMS.

VM/ESA features and components. To span the complete line of current System/370™ and System/390 processors, VM/ESA offers two features. The 370 feature supports smaller processors such as rack-mounted ES/9000 processors when run in System/370 mode, as well as older processors such as the ES/9370™, which are limited to 370-mode operations. The ESA feature, in turn, supports rack-mounted ES/9000 processors when run in System/390 mode, and intermediate and large processors such as the ES/3090™ and ES/9000, which support the ESA/370™ and ESA/390™ architectures. This can be seen in Figure 4. Each feature shares a set of common components for CMS and related communication subsystems, but has separate components for the control program and problem diagnosis functions. Both features have been de-

Figure 3 Simplified VM genealogy

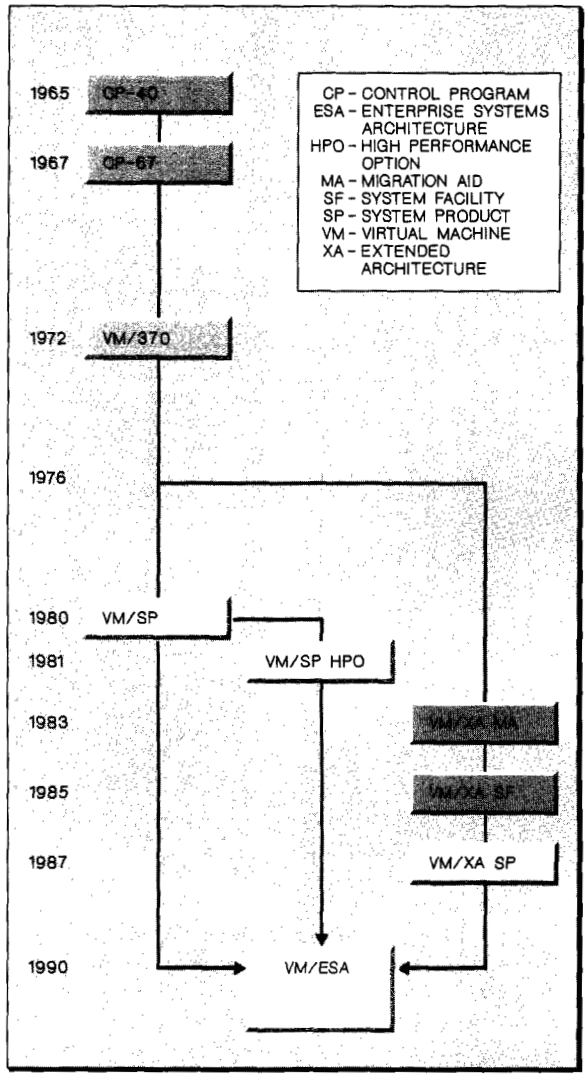
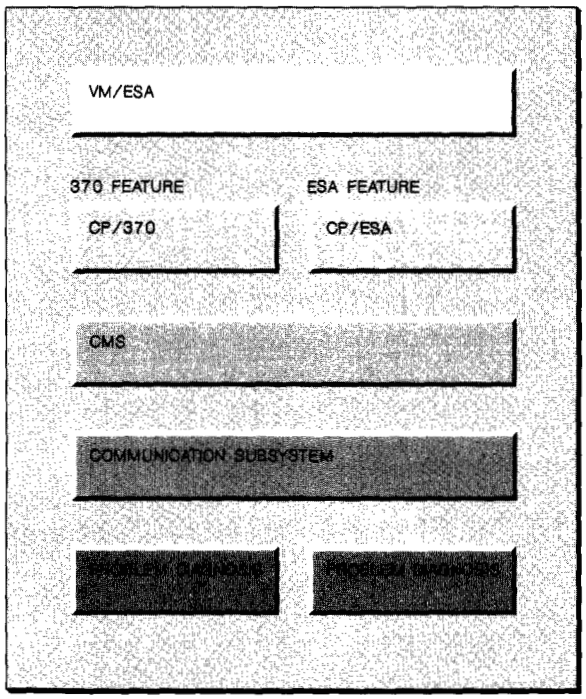


Figure 4 VM/ESA features and components



signed to provide a high degree of compatibility so that customers outgrowing the 370 feature of VM/ESA can switch to the ESA feature with ease.

VM/ESA support for centralized computing. The past decade has seen a large growth in the processing capabilities of large central systems. Innovations such as six-way symmetric multiprocessing and the introduction of an integrated vector facility have greatly augmented the amount of computing power that can be utilized for a single problem. Sympathetic enhancements to the memory and input/output (I/O) subsystems in support of these processing capabilities, such as expanded storage and Enterprise Systems Connection Architecture™ (ESCON™) channels, have also been introduced. This evolution imposes substantial requirements on an operating system. For example, the internal algorithms and data structures of an operating system may perform satisfactorily on small systems but be inadequate to the demands imposed by larger systems. This section highlights some of the changes introduced by VM/ESA in support of centralized computing.

Processor evolution. Today's ESA systems represent more than a quarter century of evolution of the basic IBM System/360™ architecture introduced in 1964. Since then, there have been four evolutionary enhancements to the architecture. In 1972, the original System/360 architecture was superseded by System/370 Advanced Function, which brought virtual storage capabilities into the architecture. In 1981, System/370 Extended Architecture was introduced, which contained 31-bit addressing and an enhanced I/O subsystem. In 1988, Enterprise Systems Architecture/370™ further extended the basic 370 addressing scheme by introducing dataspaces and access registers.[9] In 1990, ESA/390 was introduced to provide new levels of function and performance in processor, memory, and input/output for the System/390 family of processors.

In supporting the full range of System/370 and System/390 family processors, VM/ESA provides application access to advanced ESA/390 capabilities specifically tailored to the needs of virtual machines. This architecture, called Enterprise Systems Architecture/Extended Configuration, is exclusive to the virtual machine environment and provides CMS programs that have extended addressing capabilities with a facility called VM Data Spaces.[10] This facility enables high-performance data sharing among multiple virtual machines by allowing them to share addressability to the same data space. This can, in many cases, eliminate the need for communication between virtual machines.

Memory evolution. A single 4-megabit memory chip used in today's processors can store as much information as the entire main storage of a processor of the early 1970s. Similarly, the extended configuration architecture allows VM/ESA to provide applications with access to more than a quarter million times the virtual storage available to applications under VM/370. Managing a real memory system consisting of hundreds of megabytes of main storage coupled with several gigabytes of expanded storage imposes significant problems that required a complete redesign of the internal CP storage management algorithms inherited from VM/370. These algorithms were first implemented as part of VM/XA and form the basis for the storage management functions of VM/ESA.[11]

I/O evolution. Despite continuing improvements in I/O device performance, rotating media repre-

sent one of the slowest elements of modern computer systems. The strategy for addressing the performance imbalance between processor and I/O speeds has been to use the memory hierarchy to provide a series of caches, all aimed at reducing the amount of time the processor spends waiting for I/O operations.[12] One of the main techniques to reduce I/O latency used by VM/ESA is to provide cache for minidisks in expanded storage.[13] This support is both automatic and transparent to users and applications.

The continuing rapid growth in the amount of data stored and managed by computer systems poses another challenge. To address this, IBM has introduced and continues to evolve the DFSMS/VM™ product.[14] The provision of this product as an integral component of the ESA feature of VM/ESA is designed to facilitate the management of large amounts of storage.

VM/ESA support for distributed computing. The software local area network structure of VM implies that CMS essentially "sees" a distributed processing environment even within the confines of a single real machine. The advantage of this structure is that processor boundary crossings can be made largely transparent to CMS users and applications since they have no inherent dependency on shared-memory interfaces. The architecture of VM was well suited to evolve from "virtual" distributed processing to "real" distributed processing. To achieve this goal, three things were required: communications, distributed data sharing, and distributed data integrity.

VM/ESA communications support. Initially virtual machines communicated with each other via the hardware virtualizing capabilities of CP. For example, virtual machines could be linked together by virtual channel-to-channel adapters. In the 1970s, however, CP provided a set of services specifically geared to allowing efficient communication between different virtual machines.[15] In 1976, the Virtual Machine Communication Facility was introduced. This was followed in 1980 by the Inter-User Communication Vehicle, in 1985 by Advanced Program-to-Program Communications/VM, and in 1988 by a high-level interface to Advanced Program-to-Program Communications/VM called the Common Programming Interface-Communications, which is a key component of the VM support for IBM's Systems Application Architecture™. In 1990, Advanced Program-to-Program Communications/VM was further extended to the workstation environment with the introduction of the Personal Workstation Communications Facility.

Each of these facilities provided a progressively more powerful and general-purpose set of communications services. In turn, by exploiting these native services, VM now offers a whole range of additional communications capabilities, with Transmission Control Protocol/Internet Protocol, Open Systems Interconnection, and the Manufacturing Automation Protocol among the most important.

The software local area network system structure of VM is thus a natural bridge to the world of real workstations and hardware local area networks.

VM/ESA data sharing. In 1988 VM/SP introduced the CMS Shared File System, a solution to the problems of the minidisk file system noted previously. An enhanced version of the Shared File System (SFS) is the centerpiece of CMS file management in VM/ESA.[16] While retaining a high degree of compatibility with the minidisk file system, SFS introduces significant new functions in support of distributed computing. Among these are the ability to group files into hierarchical directory structures and to share them on an individual file basis with users on both local and remote systems. SFS solves the isolation problem inherent in the minidisk file system because files are maintained by a server that provides CMS file management services to any user authorized to communicate with it. The collection of all files managed by a single SFS server is called a *filepool*. Because the communication between CMS users and SFS servers is with Advanced Program-to-Program Communications/VM, users can be remote from the system on which the SFS filepool resides.

While providing local and remote transparency, SFS exploits VM/ESA features such as VM Data Spaces to provide improved performance for local users. CP also allows SFS data storage devices to receive preferential caching considerations in expanded storage. VM/ESA thus allows CMS minidisk users to obtain the significant functional benefits provided by SFS with minimal loss of local performance.

VM/ESA data integrity. Data integrity is of fundamental importance in computing. This is espe-

cially true in distributed processing environments in which multiple systems can interact in complex and often nonintuitive ways. For such environments, informal mechanisms are a poor substitute for a formal approach with an architecture for handling data integrity. To address these needs, VM/ESA provides Coordinated Resource Recovery, a comprehensive set of system services and protocols for the management of distributed data, which fully conforms to the Systems Application Architecture Common Programming Interface for resource recovery.[17] With Coordinated Resource Recovery, an application can update multiple resources in a distributed environment with full integrity. This means that a set of updates is guaranteed to be committed as a unit; within a logical unit of work, either all updates or no updates are performed, regardless of the number of resources involved or the intrusion of system or communications failures. By providing Coordinated Resource Recovery as a system service, VM/ESA greatly simplifies the task of creating reliable distributed applications.

Coordinated Resource Recovery is a VM/ESA system service.

SFS exploits Coordinated Resource Recovery in VM/ESA to allow for transparent coordination of updates to files managed by multiple SFS servers. VM/ESA thus removes the restriction in VM/SP which disallowed updates to more than one SFS filepool at a time.

RAS requirements. A strong focus on reliability, availability, and serviceability (RAS) is one of the key distinguishing features of a commercial software product. This is especially true of operating systems.

Reliability measures how often and under what conditions a product will fail; availability measures the impact of hardware and software failures; serviceability measures the ease with which failures can be diagnosed and repaired. In an ideal world a computing system would never fail, would always be available, and would never require service. In practice, there is a tradeoff between RAS and cost, with perfection being an elusive goal. Nevertheless, the VM product was significantly behind other operating system products (e.g., MVS) in terms of RAS as it entered the 1980s.

As part of the basic resource management infrastructure of an operating system, RAS considerations are predominantly the domain of CP. CP RAS enhancements were required to cover both hardware and software, and the RAS improvements found in VM/ESA were first implemented as part of VM/XA. They include hardware fault isolation and recovery, as well as a systematic approach to handling software errors. Almost all errors in VM/ESA are contained within the user or subcomponent that experienced the failure, thus minimizing system-wide failures. The result is that VM/ESA is the most reliable virtual machine operating system that IBM has ever offered.

Migration requirements. While CP was undergoing major restructure to prepare it for the RAS and centralized computing needs of the late 1980s and beyond, a parallel effort ensured that existing VM/SP and VM/SP HPO customers would experience a smooth migration to VM/ESA. The result is that many updates—often minor—were made to VM/XA to evolve it into a VM/ESA which would retain a high degree of compatibility with previous VM products.

One of the major migration features of VM/ESA is that CP allows a virtual machine to select which level of the architecture it uses. Thus, older System/370-mode programs can run side-by-side with programs which exploit ESA—all on the same processor. This capability, inherited and extended from VM/XA, is realized through the use of the ESA/390 interpretive execution facility and the Start Interpretive Execution (SIE) instruction.[18]

VM/ESA: A system for the 1990s

Improvements in communications technology made during the 1980s permitted distributed processing to become an economically viable form of computing. The communications technology of the 1990s, by contrast, promises to make high-speed digital communications so pervasive that the words "centralized" and "distributed" will lose their usefulness as a means of characterizing

computing systems. Whereas in the early 1980s isolated computers were still very much the norm, in the early 1990s they are already an increasingly rare exception. By the late 1990s, the phrase "isolated computer" will seem almost a contradiction in terms.[19]

The practical consequence of this evolution is that remote access to applications, data, and services is becoming the normal mode of operation. It is appropriate, therefore, to look to the operating system to provide a framework for application and data management in this new environment. VM/ESA has been designed to combine—in a single product—the ability to operate in both centralized and distributed computing environments, while spanning the entire range of System/370 and System/390 family processors. VM/ESA continues to build on the historical strengths of performance, function, low entry cost, and flexibility to satisfy the requirements of VM customers.

Acknowledgments

The author wishes to thank Ed Pruul, Dave Dowling, Karl Schubert, Ray Mansell, Eleanor Coy, Damian Osisek, and Tom Szczygielski for their helpful comments on this paper.

Cited references

1. *IBM Systems Journal* **18**, No. 1 (1979, whole issue).
2. R. P. Parmelee, T. I. Peterson, C. C. Tillman, and D. J. Hatfield, "Virtual Storage and Virtual Machine Concepts," *IBM Systems Journal* **11**, No. 2, 99–130 (1972).
3. R. J. Creasy, "The Origin of the VM/370 Time-Sharing System," *IBM Journal of Research and Development* **25**, No. 5, 483–490 (1981).
4. T. L. Borden, J. P. Hennessy, and J. W. Rymarczyk, "Multiple Operating Systems on One Processor Complex," *IBM Systems Journal* **28**, No. 1, 104–123 (1989).
5. R. A. MacKinnon, "The Changing Virtual Machine Environment: Interfaces to Real Hardware, Virtual Hardware, and Other Virtual Machines," *IBM Systems Journal* **18**, No. 1, 18–46 (1979).
6. *VM/ESA: CMS User's Guide*, SC24-5460, IBM Corporation; available through IBM branch offices.
7. E. C. Hendricks and T. C. Hartmann, "Evolution of a Virtual Machine Subsystem," *IBM Systems Journal* **18**, No. 1, 111–142 (1979).
8. *VM/VSE SQL/DS Guest Sharing Guide*, GG24-3462, IBM Corporation; available through IBM branch offices.
9. C. A. Scalzi, A. G. Ganek, and R. J. Schmalz, "Enterprise Systems Architecture/370: An Architecture for Multiple Virtual Space Access and Authorization," *IBM Systems Journal* **28**, No. 1, 15–38 (1989).
10. J. M. Gdaniec and J. P. Hennessy, "VM Data Spaces and ESA/XC Facilities," *IBM Systems Journal* **30**, No. 1, 14–33 (1991, this issue).
11. G. O. Blandy and S. R. Newson, "VM/XA Storage Management," *IBM Systems Journal* **28**, No. 1, 175–191 (1989).
12. E. I. Cohen, G. M. King, J. T. Brady, "Storage Hierarchies," *IBM Systems Journal* **28**, No. 1, 62–76 (1989).
13. G. P. Bozman, "VM/XA SP2 Minidisk Cache," *IBM Systems Journal* **28**, No. 1, 165–174 (1989).
14. J. P. Gelb, "System-Managed Storage," *IBM Systems Journal* **28**, No. 1, 77–103 (1989).
15. R. M. Jensen, "A Formal Approach for Communication Between Logically Isolated Virtual Machines," *IBM Systems Journal* **18**, No. 1, 71–92 (1979).
16. R. L. Stone, T. S. Nettleship, and J. Curtiss, "VM/ESA Shared File System," *IBM Systems Journal* **30**, No. 1, 52–71 (1991, this issue).
17. B. A. Maslak, J. M. Showalter, and T. Szczygielski, "Coordinated Resource Recovery in VM/ESA," *IBM Systems Journal* **30**, No. 1, 72–89 (1991, this issue).
18. D. L. Osisek, K. M. Jackson, and P. H. Gum, "ESA/390 Interpretive-Execution Architecture, Foundation for VM/ESA," *IBM Systems Journal* **30**, No. 1, 34–51 (1991, this issue).
19. "Getting Together Bit by Bit," *Science* **248**, 160–162 (1990).

William T. Fischofer *IBM Endicott Programming Laboratory, Route 17C & Glendale Drive, Endicott, New York 13760.* Mr. Fischofer is currently an advisory programmer with the VM Systems Design organization in the Endicott Programming Laboratory. Prior to joining IBM, he worked for several years as a VM systems programmer. In 1983, he joined the staff of the IBM Thomas J. Watson Research Center in Yorktown, New York, and in 1986 he transferred to the Endicott Programming Laboratory where he has held a number of positions in VM design and development. He received a B.S. degree in mathematics from Georgetown University and an M.S. degree in computer science from Cornell University.

Reprint Order No. G321-5420.