Directions in cooperative processing between workstations and hosts

by B. C. Goldstein
A. R. Heller
F. H. Moss
I. Wladawsky-Berger

Advancements in technology have provided us with the availability of high-performance processors from the high end of computing to the personal computer. In addition, technology growth has enabled us to envision sixteen megabytes of real storage for a personal computer.

As a result, we have witnessed not only a tremendous growth at the high end of the computing spectrum, but also the development of sophisticated personal computers (e.g., the IBM PC XT/370) with real storage capacities approaching those of high-end computers of a decade ago.

This growth at both ends of the computing spectrum has given us a choice. We can either allow a clean separation to grow between personal computer and host or provide a means by which they cooperate in providing quality service to the user without the complexity normally associated with high-end systems. This paper explores what such a cooperation could mean.

As host machines have developed over the years, they have come to be viewed in several dimensions. One dimension is that of significantly sophisticated state-of-the-art software services, such as data base managers (e.g., IMS,[1] DB2,[2] SQL/DS,[3]) and batch schedulers. Another dimension is that of high performance [hundreds of Millions of Instructions Per Second (MIPS) in computing power available to the users], high availability, and reliability. There is also such state-of-the-art hardware as high-performance, letter-quality printers. Host machines have been viewed by non-data-processing professionals as having poor human factors. The major reason put forth has been that these systems evolved from the batch era of programming, when a computer did not have to be friendly to its users. In general, host systems serve many users concurrently with sophisticated services.

On the other hand, personal computers are considered to be relative newcomers to the field, with their major acceptance occurring within this decade. Personal computers started off modestly. The basic medium for retaining data on early systems was the cassette, and memory was typically between 4K and 16K bytes. One hardly considered them to be in the same league with large mainframe computers. With the passage of time, the power of the personal computer has increased so that one can now talk about having a desk-top personal System/370. Such a system might be equipped with 768K of memory, 20 megabytes of hard disk, and a printer. Such a configuration used to be referred to as a mainframe computer.

Personal computers are noted for having several desirable characteristics. First, these systems are designed for the non-data-processing professional. As such, they assume almost no knowledge of a computer and are designed to be simple and easy to learn. Most personal computers are single-user systems. That is, they are not shared with other users.

One's data and programs are private and cannot be accessed concurrently by another. As single-user systems, personal computers generally perform simple multiprogramming. For example, they may allow the printer to work in parallel with editing a report. Because work is done serially, one cannot create two or more processes actively working against the same data. Autonomy is another important consideration when providing a personal computer for the DP professional. (See Figure 1.) Because the personal computer is independent of the mainframe, if the mainframe should fail the personal computer user can continue. Performance of a personal computer, although not matching that of the mainframe, is at least consistent. No matter how many other persons are working next to the personal computer user, the response time of that system remains invariant.

Interdependences among hosts and personal computers

It was once thought that large numbers of personal computers could one day replace the giant mainframe. It is now clear that, although processing power is becoming less expensive, the cost of peripherals (such as large DASD, mass storage devices, and high-quality/high-speed printers) is not decreasing proportionately. The implications are that, even if a large number of minicomputers were installed, a typical installation could still not afford to allocate the peripherals per minicomputer nor have the processing power to efficiently utilize the peripherals. The organization to support the operation of a large computing center is not something that a typical user desires to be confronted with. It is true that a local area network helps to provide increased sharing among minicomputers, but one still needs significantly greater processing power to utilize many of our traditionally expensive peripherals. Thus, from a cost amortization perspective, one cannot eliminate the mainframe.

Consider the following conjecture. In a corporation that employs both mainframes and personal computers, personal computer users want access to the mainframes, and mainframe users want personal computers. Let us consider the requirements of both classes of users.

The personal computer user wants access to the mainframe to share data. Giving another user a floppy disk containing the data causes the originator to lose control and prevents both users from concurrently working on those data. Mainframe operating

Figure 1 Application processing autonomy

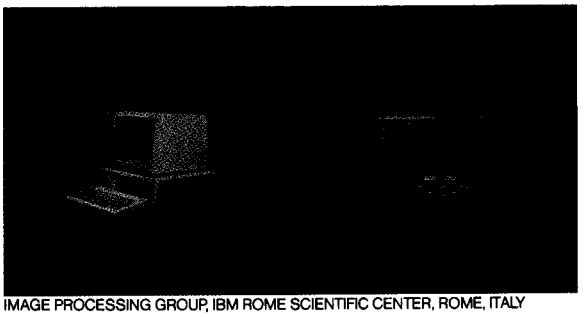

IMAGE PROCESSING GROUP, IBM ROME SCIENTIFIC CENTER, ROME, ITALY

systems usually provide reasonably sophisticated data sharing and security.

Given the limited capacity of the personal computer, one can easily run out of DASD space. Maintaining backup files on floppy disks can be cumbersome and time-consuming. Mainframes offer mass storage devices that give the illusion of having an infinite archive for data storage.

Executives in a large enterprise, having their own personal computers, may wish to access the corporate data. At one time, it was assumed that distributed data would make such access possible. A hyperbolic solution to that problem might have sounded like this: Take the four hundred spindles of DASD connected to the mainframe, throw away the mainframe, place one spindle on each employee's desk, and connect all employees into one giant local area network. Although such a configuration might be almost practical for a non-data-base enterprise, it would have been a disaster for an enterprise with a centralized, integrated corporate data base, for several reasons.

Security. Suppose that one data base is the personnel file. Consider how vulnerable individuals and the company would be if one were to place a spindle of those data on an employee's desk. Corporate data implies a great deal of security, recovery, and audit control. Mainframe data base management systems are designed to address these requirements.

Recovery. Consider the damage if one were to spill a cup of coffee over the hard disk containing corporate data. Needless to say, the probability of this happening in a mainframe-controlled environment is small. One cannot say the same thing about data associated with the personal computer.

Performance. Consider the probability that the data one wants are on a given user's personal computer.

Figure 2 The Personal Computer as a terminal, shown here emulating the IBM 3270

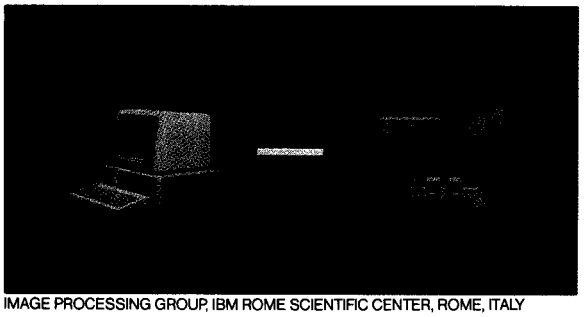

IMAGE PROCESSING GROUP, IBM ROME SCIENTIFIC CENTER, ROME, ITALY

If there are 400 users in a local network, is the probability one in 400 (assuming random distribution of the data)? No. It may turn out to be as great as one in 400! (factorial) if in traversing the network we do not remember which users we have previously visited.

These are just a few of the reasons why one would not replace a mainframe with personal computers for access and control of corporate data. This is not to say that distributed data is a bad idea. Two reasons for the distribution of data among mainframes are the following:

Geographic separation. For an enterprise that has more than one mainframe separated over large geographic distances, one wants to partition the data along these geographic boundaries for the purpose of performance gain.

Catastrophic error. Some enterprises replicate their data across geographically separated mainframes as insurance against catastrophe, such as earthquake, flood, and fire.

Thus, we have a number of very subjective reasons why personal computer users want access to mainframes. For objective reasons, one has merely to look at the growing industry of vendors who make communication cards that enable personal computers to connect to mainframes. (See Figure 2.) Among the reasons why mainframe users want personal computers are the following:

Function isolation. The loss of a single personal computer does not affect any other personal computer connected to the same mainframe. There are no longer fears that a program in development will destroy another application due to unforeseen errors.

Reduced complexity. Given that a personal computer is intended to support only a single user, the human interface can be made very simple, for example, by not requiring JCL. Furthermore, the user need not negotiate with computer center personnel as to the requirements of his application. In addition, there is a relatively short educational period required to learn how to use a personal computer.

Offloading. Although a user sees consistent performance at the personal computer, the mainframe performs faster. By offloading trivial user transactions (i.e., those with short path length and few I/O requests) the mainframe can devote its resources to more efficient support of nontrivial transactions, such as batch work, heavy computation-limited activity, and intensive I/O operations.

Although personal computers tend to satisfy these concerns, they do have their limitations. Personal computers are primarily DASD limited, and thus preclude very large data base support or large application support. Thus, personal computers and mainframes complement one another's capabilities. The limitations of personal computers are those that are easily addressed by mainframes, and the limitations of mainframes are easily addressed by the personal computer. Ideally, we would like to have an environment that gives us the advantages of both.

Personal computers and mainframes complement one another's capabilities.

Cooperation and its challenges

How do we characterize an environment in which mainframes not only coexist, but also cooperate with personal computers? For the purposes of discussion and for brevity, we now refer to mainframes as *hosts* and to cooperating personal computers as *workstations* or *Intelligent Workstations (IWS).*

Figure 3 Cooperative processing

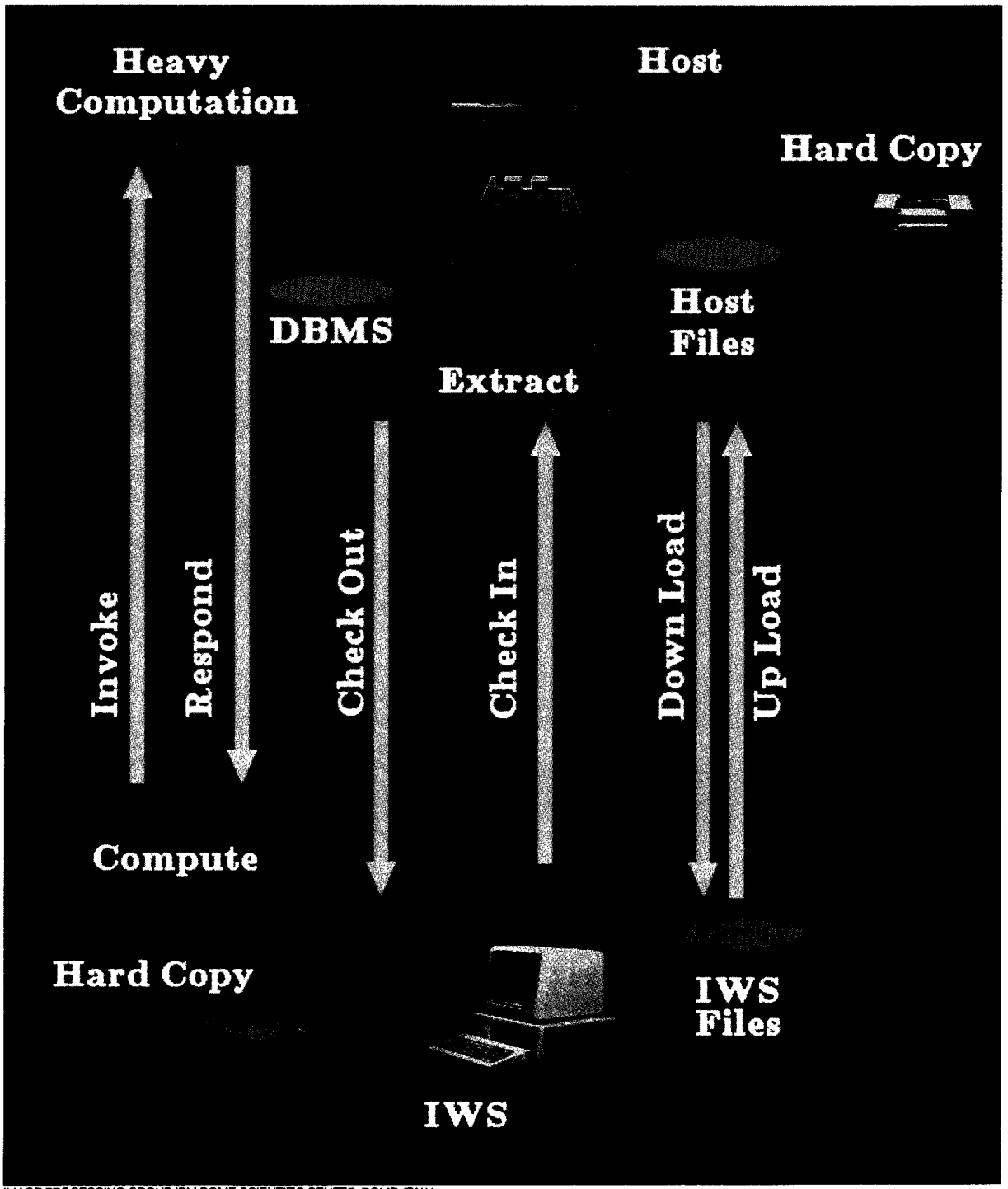

IMAGE PROCESSING GROUP, IBM ROME SCIENTIFIC CENTER, ROME, ITALY

Intelligent workstations. Cooperative processing enables us to view the workstation as being an environment in which to run trivial, compute-bound transactions. (See Figure 3.) The IWS also has a modest

repository for local private files as well as a small and inexpensive printer.

Host. On the other hand, the host, in our view, provides the workstation with an environment in which to run large, compute-bound transactions. The host also performs basic underlying services for the workstation, such as uploading and downloading files between the workstation and the host, for example, for archival purposes. The host also has the ability to extract data from corporate data bases. The host makes available high-performance/high-quality printers and a global communications network. Thus the host is simply a place in which to submit long-running transactions, batch jobs, and so forth.

In order to provide such a natural division between workstation and host, we are faced with a number of objectives and challenges. To begin with, we would like to bring the host interfaces for services and data down to the Intelligent Workstation without also bringing along the complexity inherent in those interfaces. Such a capability would, for example, enable an IWS application to issue SQL requests to a DB2 data base without requiring the application programmer to be a DB2 systems programmer.

The key challenge is that the workstation user not be required to learn three command languages: that of the local workstation to which the user has already adapted; the network command language; and the host command language. In general, the user wants to use the language to which he has already adapted.

Virtual services and data. To address this challenge, cooperative processing interfaces should provide an interface to a set of virtual services and data in such a way that their local and remote locations are transparent to the requester. An application in one processor, when requesting a service, obtains that service no matter where it executes. The application perceives all services and data to be local, even if they actually exist on a remote processor. Of course, the timing may vary for remote facilities as compared with local ones.

Providing service and data transparency would enable us to offload a host application onto the workstation while leaving the data host resident. This would avoid expensive application redesign costs. Today, such costs usually entail the application programmer dissecting the application into two parts: (1) one part on the workstation interfacing with the other on the host, using a telecommunication access method, and (2) the second part performing the data access. Although this may seem simple, it can be a very complex process. For example, there may be no one in the organization who understands how the application originally worked, or the organization may no longer have either the source or libraries used in creating the program.

The optimal situation is simply to offload the application onto the workstation. This requires that the original requests for service and data must work, even though they do not exist at the workstation.

In providing such transparency between workstation and host one can see, as an analogy, the virtual machine capability presented by VM/370.[4] VM/370 presents an interface that is the System/370 architecture. This interface enables guest operating systems to believe that they are actually running on real System/370s. A similar analogy exists in virtual memory systems where an application may see itself as running in 16 megabytes of real storage, for example. In fact, the application may be running in 512K bytes of memory, and references to program pages may result in page faults that are transparent to the application.

The Virtual Service Interface enables the user to see all host files as being locally available on the workstation.

Although we like to contemplate transparency, we must provide the ability to copy data from one environment to another through explicit commands. There are two basic approaches to this:

- *Transparency.* As previously described, the user simply enters the local COPY command that normally copies files from one workstation disk to another, as a means of copying files from the workstation to the host and vice versa.
- *Introduction of new commands. Import/Export* commands explicitly state that a file should be copied down from the host to the workstation

(Import) or that a file should be copied from the workstation to the host (Export).

Although transparency is the simplest solution, it may not be adequate for all file types. For example, some binary fields may not translate properly when going from an ASCII workstation to an EBCDIC host without added information. The introduction of new commands provides a means of enhancing the user's command repertoire to include such translation declaratives.

Present status of cooperative processing

Given this general understanding of cooperative processing, what is the current practical situation as exemplified by products? In 1983, IBM announced two major workstation products, the IBM PC XT/370[5] and the IBM 3270 PC. Both products provide a basis for cooperative processing with a host System/370.

The IBM Personal Computer XT/370. In summary, the IBM XT/370 is an IBM PC/XT with an additional three-card set. These three cards perform the following functions:

- *IBM 3277–2 emulation adapter card.* This card enables the XT/370 to be connected via coaxial cable to an IBM 3274 controller for either local or remote host operation.
- *System/370 processor card.* This card contains three microprocessors and supporting random access memory and logic to provide a true System/370 capability.
- *A 512-kilobyte memory card.* This memory is accessible from either the processor card or the PC/XT processor itself.

The XT/370 provides three forms of cooperative processing.

The user can concurrently log on to the host (here the XT/370 appears as a terminal) and perform local System/370 processing. In this form, the XT/370 provides the ability to concurrently execute local simple transactions in parallel with heavier computation-limited host transactions.

Explicit commands are provided to Import and Export files between the local System/370 environment and a co-resident PC/DOS file environment. That is, explicit commands are provided that enable a user to transform an existing PC file into a VM file and vice versa, even though the file systems are different.

Figure 4 A major step toward cooperative processing

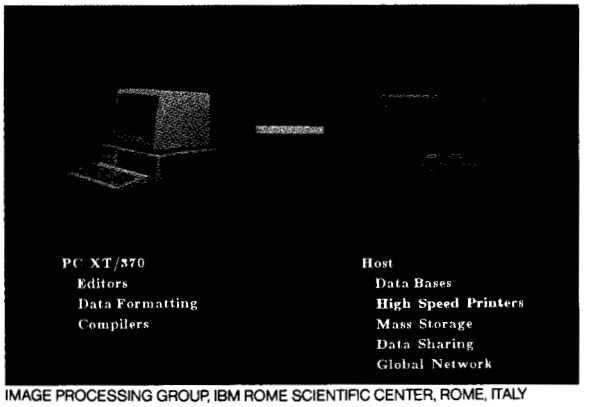

IMAGE PROCESSING GROUP, IBM ROME SCIENTIFIC CENTER, ROME, ITALY

A new form of cooperative processing is provided for transparent access of host data and services from the local VM environment. (See Figure 4.) This interface, known as the *Virtual Service Interface (VSI)*,[6] enables the user to see all host files as being locally available on the workstation, assuming proper security authorization. To copy a file from one environment to another the user merely uses the local COPY command. Similarly, the user can view the host printers as being locally attached. (In this case, the user may simply have to walk to the host to pick up the output.) This support is provided not only between homogeneous systems, such as VM on the host and VM on the XT/370, but also between dissimilar systems, such as MVS/TSO[7] on the host and VM on the workstation.

The TSO support enables the user to transparently access any TSO data set or Partitioned Data Set (PDS) as though those data sets were CMS files and minidisks. Here the VM user at the workstation does not have to learn MVS/TSO commands and can continue to access files even though they are remote. That is, the user does not have to adapt to another command language.

Provided with this support is a set of mappings that map the file types in CMS (on the XT/370) into the equivalent form in TSO. As examples of this mapping operation, one can cause a file type of SCRIPT in CMS to map into TEXT in TSO or cause ASSEMBLE in CMS to map into ASM in TSO. This set of mappings is transparent to the user, though a new VMPC command DSNMAP is provided so that the user can dynamically redefine the mappings in a more personalized fashion.

The following examples demonstrate the explicit and implicit use of the DSNMAP command:

```
LINK MOMADUK 191 233 W
ACCESS 233 K
```

Assuming that the requested minidisk does not reside locally on the workstation but is transparent to the user, these commands cause TSO to LINK to all MVS data sets belonging to the user MOMADUK as the high-level qualifier and to identify the minidisk to CMS as file mode K. The default map or pattern provided is the following:

```
DSNMAP K &FN.&FT
```

In this situation, the user does not have to specify the DSNMAP command. This map is automatically set up with the LINK. A CMS file request for EXAMPLE TEST K is treated by TSO implicitly in the form MOMADUK.EXAMPLE.TEST.

The following example shows how to map to two MVS partitioned data sets as different file modes:

```
LINK MOMADUK 191 233 W
ACCESS 233 K
DSNMAP K DATASET1.&FN(&FT)

LINK MOMADUK 191 234 W
ACCESS 233 L
DSNMAP L DATASET2.&FN(&FT)
```

Given this example, the issuing of the command COPY * * A = = K copies all files on the local A disk into the DATASET1 partitioned data set. Each file copied over becomes a member of the PDS. If the members of PDS DATASET1 had not existed prior to the invocation of the COPY command they are dynamically created. This interface is extremely rich.

From the workstation perspective, it makes no difference whether the host is a VM- or MVS-based system. The local file system using the VSI merely states that it would like to request service of the file server subsystem. Requests in this interface are of the form "I want function X of subsystem Y," where a subsystem's functions are represented in tabular form. [This is very much like the host MVS Subsystem Interfaces (SSI)[8-10].] The VSI determines whether the requested service can be performed locally or has to be routed remotely. The necessary transforms are done on the host side.

In the TSO host support, the user can also enter host TSO commands. The fact that those commands are performed remotely is transparent to the requester. Similarly, local print requests are transparently mapped into host spool requests when that is appropriate.

As can be seen, the XT/370 satisfies many of the objectives of cooperative processing. What is missing is the ability to go the other way, i.e., to give a host application access to workstation services and data, such as to run a compiler on the host to compile a program that is resident at the workstation. Despite this, we see a significant start in cooperative processing embodied in the IBM PC XT/370.

The IBM 3270 Personal Computer

We briefly summarize here the capability of the intelligent workstation, the IBM 3270 Personal Computer. The IBM 3270 PC combines the ability to perform host interactive functions of the 3270 Information Display System with the computing capability of the IBM Personal Computer. The user can concurrently establish up to four 3270-type sessions with possibly four different hosts (all sessions being concurrent), two local notepad sessions, and one PC DOS session.

From a cooperative processing perspective, the IBM 3270 PC addresses three distinct alternatives.

By utilizing its highly sophisticated window system, the user can easily move information between any of the 3270-type sessions and/or the notepad areas. Thus, information that is being displayed by one session may be moved to the display of another session, without the modification of any host/workstation application.

Explicit commands are provided for in the PC DOS session to import (receive) and export (send) files between the PC DOS session and the host session (whether it is CMS, TSO, or CICS). The user learns a single (albeit new) set of commands for the movement of files irrespective of the host environment. That is, the IBM 3270 PC does not require the user to learn a new set of commands for file transfer (or window data movement) for each different set of host environments.

The IBM 3270 PC also provides the ability to establish concurrent sessions with up to four different hosts. This means that the user may be requesting heavy

computations or data-intensive activity to be performed on four different systems, while running a PC DOS application or while transferring data between the PC and one of those sessions.

All these services are provided in a simple, easy-to-use fashion. In fact, one of the authors learned to use the workstation without use of manuals.

Both the XT/370 and the 3270 PC offer a significant stake in the ground in terms of providing a basis for cooperative processing.

Future directions and summary

The following are the authors' personal conjectures as to research that might prove fruitful in the host-workstation synergism. We have shown here the beginnings of cooperation between workstations and hosts. We can contemplate that a second stage might be based on research into the following:

- Establishing up to four concurrent 3270-type sessions.
- Issuing host commands from the native workstation environment as well as from any of the host sessions.
- Providing virtual diskette support for native applications.
- Interchanging host files and local workstation files transparently.
- Supporting a local coprocessor, such as the System/370 in the XT/370.

From the host perspective, one should not forget CICS and its functional capability. Cooperative processing should grant a PC application access to the host CICS application programmer's interface, even though CICS is not in the workstation. The implication of such an interface is that the PC application would then have access to host DL/I, DB2 data, and a wealth of existing CICS applications. One could expect similar access through TSO to such subsystems as on-line IMS DL/I and DB2, and SQL/DS from VM. This would enable the PC application programmer to issue calls to DL/I or DB2 as though those subsystems were running on the workstation. Clearly, such interfaces are not geared for the novice PC programmer. Rather, they are oriented toward the more advanced PC systems programmer.

One would expect the PC systems programmer to utilize these cooperative processing interfaces in the development of such ergonomically designed applications as spreadsheets and data base query.

Concerning the novice, hosts might grow toward the development of generalized extract programs. Such host programs would perform sophisticated host data base requests and merely provide the workstation with the end results to be incorporated into reports.

For the person who uses host systems (such as TSO and CMS) interchangeably with the workstation environment, access to data in the different file systems should be interchangeable. For example, consider the person who has a significant investment in host CMS files (such as reports prepared in SCRIPT format) and who uses the IBM PC Personal Editor (PE). This person would like to use PE to edit host files as though those files were local files.

For host applications, it should similarly be possible to access workstation data or simply use the workstation printer for local output.

In the realm of presentation services (i.e., formats in which data are presented for display), host subsystems and applications should be able to take advantage of the simple, easy-to-use presentation services and windowing capabilities provided by the workstation.

In general, regardless of the cooperative processing performed between workstation and host, one would like the interface for service to be common and invariant. Change of the host environment should not affect workstation applications (causing different versions) depending on whether a given subsystem is available.

Furthermore, as both workstation applications and cooperative processing subsystem interfaces for host subsystems grow, we expect to see a need for greater recovery control in the workstation to operate when an application fails. If we are operating in a multiprogramming environment, we do not want the failure of a single application to bring down all others.

To summarize, cooperative processing is presenting us with many benefits as well as renewed challenges. The challenge of providing simple, easy-to-use interfaces is a continuing grand objective.

Acknowledgments

The authors are indebted to Lee Hoevel, Jean Voldman, and Lynn Trivett for their contributions leading to the design of the Virtual Service Interface (VSI). Credits for the idealizations and implementations of the VSI are due also to Jon Bangs, Dan Casey, George Case, Vini Cina, Lyle Haff, Hank Harrison, Frank Kozuh, Larry Koved, Andy Pierce, Francis Parr, Ray E. Rose, Thom Scrutchin, Dave Wherly, and Barry Willner. For their refinement of the notion of cooperative processing we express our gratitude to Jim Cannavino, Don Gibson, Peter Hansen, Bill Kleinbecker, Leonard Liu, Steve Stark, and Tom Wheeler. Finally, we thank Gene Trivett for his significant expertise in helping us shape the direction cooperative processing might take.

Cited references

1. W. C. McGee, "The information management system IMS/VS," *IBM Systems Journal* **16,** No. 2, 84–168 (1977).
2. *IBM Systems Journal* **23,** No. 2 (whole issue, 1984).
3. M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson, "System R: Architectural approach to data base management," *ACM Transactions on Data Base Systems* **1,** No. 2, 97–137 (June 1976).
4. *IBM Systems Journal* **18,** No. 1 (whole issue, 1979).
5. F. T. Kozuh, D. L. Livingston, and T. C. Spillman, "System/370 capability in a desktop computer," *IBM Systems Journal* **23,** No. 3, 245–254 (1984, this issue).
6. B. C. Goldstein, G. Trivett, and I. Wladawsky-Berger, *Distributed Processing in a Large Systems Environment*, Research Report RC-9027, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598 (1976).
7. A. L. Scherr, "Functional structure of IBM virtual storage operating systems; Part II: OS/VS2–2 concepts and philosophies," *IBM Systems Journal* **12,** No. 4, 382–400 (1973).
8. J. A. Cannavino, B. C. Goldstein, and T. W. Scrutchin, "OS/VS Release 2 job management structure," *Proceedings of Guide 48* (May 1979); may be obtained from Guide International, 111 East Wacker Drive, Chicago, IL 60601.
9. *MVS/XA: System Modifications*, GC28-1152-1, IBM Corporation; available through IBM branch offices.
10. *MVS Job Management*, GC28-1303, IBM Corporation; available through IBM branch offices.

Reprint Order No. G321-5221.

Barry C. Goldstein *IBM Research Division, Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, New York 10598.* Dr. Goldstein is the department manager of the Workstation and Distributed Processing group in the Small Systems Laboratory of the Computer Sciences Department. In this position, his responsibilities include performing advanced technology research into the role of small systems in the realm of distributed cooperating heterogeneous systems. Dr. Goldstein joined IBM in 1969. He received his B.S. degree in engineering sciences from New York University and his M.S. and Ph.D. degrees in computer and information sciences from Syracuse University.

Andrew R. Heller *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Heller is an IBM Fellow and Director of Advanced Technology Systems.

Franklin H. Moss *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Moss joined IBM in November 1977 in a postdoctoral assignment at the Israel Scientific Center, Haifa, Israel. He became a Research Staff Member at the Research Center in January 1978. In September of 1979 he became leader of the network architecture project and was promoted to manager of the group in December 1979. He then assumed the position of manager of the Communications and Distributed Systems Department in June 1981. He has directed five research groups in the development of advanced technology for distributed system and data communication products: SNA Network Architecture and Protocols, Communications Subsystems, Communications Network Management, Distributed Systems Software Technology, and Distributed Systems Organization. He currently has the position of manager of the Large Systems Laboratory. He directs major efforts towards evolving large systems (including both hardware and MVS and VM software). In this position, his responsibilities include strategy planning, technical evaluation and guidance, technology transfer to product development groups, and research personnel management and development. Dr. Moss received his B.S.E. degree in aerospace and mechanical sciences from Princeton University in 1971. He received his S.M. and Ph.D. degrees in aeronautics and astronautics from the Massachusetts Institute of Technology in 1972 and 1977, respectively.

Irving Wladawsky-Berger *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598.* As Director of Small Systems and Communications, Dr. Wladawsky-Berger is responsible for technical strategy and advanced products in workstations, minicomputers, communication networks, distributed systems, user interfaces, office applications, and software technology. He has had similar responsibilities for large systems architecture, high-end operating systems, and scientific computers. Dr. Wladawsky-Berger joined IBM in 1970. He holds M.S. and Ph.D. degrees in physics from the University of Chicago.