A newly designed and implemented automated storage hierarchy management system that operates under MVS is described. The needs for economical archival storage that at the same times makes possible the efficient retrieval of users' data are reviewed. Discussed in detail is the fulfillment of this requirement that is provided by the automated management of MVS/TSO on-line data storage space that includes the Mass Storage System (MSS) and other storage devices in a hierarchy. Experience with the system is summarized.

MARC: MVS archival storage and recovery program

by J. P. Considine and J. J. Myers

Increased computing power, in the form of faster processors and faster and larger storage devices, has led to an increasing number of users served by computing installations, in batch and interactive modes. Use of the increased computing power generates more and more data that must then be stored and made available for future processing. This generates increased demand for storage, especially on-line storage. Interactive users want their data to be available in no more than a few seconds, and even batch users try to avoid the delays that arise when processing unmounted volumes. At the same time, there are constraints on the amount of hardware that can be installed to support users. As an alternative, a programming tool might enable a system to make the most efficient use possible of the resources available, while maintaining the responsiveness and availability that users depend on. This paper describes such a tool.

After discussing the problem further, the concepts underlying the overall approach to a solution and definitions of some key terms are presented. Then there is briefly summarized previous work by members of the IBM Research Division, addressing this problem in the context of a general operating system environment. A detailed description is then given of one such solution that has been implemented under the IBM OS/VS2 MVS operating system, the MVS Archiver (MARC). The term MVS (Multiple Virtual Storage) represents OS/VS2 MVS throughout this paper. The description includes architecture and function, design and implementation, and support of the Mass Storage System.

the problem

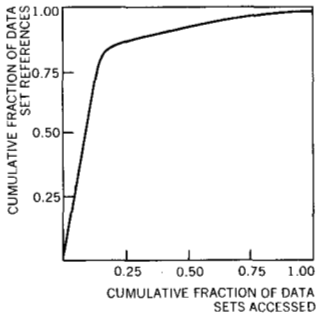
The requirement for efficient archival storage and access arises whenever a number of persons in an organization are concurrently using a computer system to solve their own problems. Such an environment might be a service bureau, a university, a research installation, or even a business of some size. The increased processing power available brings more users into contact with the computing system, and these users are coming to expect quick and reliable access to their data. With the growing number of interactive users, the number of individual files is increasing, and users working in batch mode are also relying on permanently assigned devices to avoid delays. Since spindles (disk drives) are expensive, it is important to obtain good utilization of the data devices. There are two difficulties in managing a growing on-line data base. First, stored data must be available while storage hardware is limited. Second, the integrity of the stored data must be assured.

Most desirable would be on-line storage that is an open-ended container for data. Users want to be able to create new files and extend old ones as necessary without concern for exhausting the storage resource. When on-line storage is open-ended, users can keep vast quantities of data for long times, making possible many new kinds of comparative studies that were previously not feasible. The availability of large amounts of historical data can lead to totally new applications.

Installation management, on the other hand, must exercise control so that the resource is used in accordance with management priorities and is not preempted without justification by individuals. To maintain the service offered by an installation, however, control must not unduly inconvenience the users. For instance, management may choose to assign each user a fixed amount of on-line storage, not necessarily the same for each user, but fixed for each user. For one class of user, its allocation might be more than ample and large amounts would not be used. For another class, the management of its storage ration, inherently nonproductive work, may occupy a significant portion of the user's time. Although the problem is shown here with regard to a particularly simple allocation scheme, it is not necessarily alleviated by more sophisticated ones, as long as one must manage one's assigned data space and storage needs personally. Trying to avoid arbitrary storage restrictions that in the long run lead to lower user productivity is one of the typical problems that arise in managing a growing on-line data base.

A second facet of the growing user dependence on on-line files is that of data integrity, i.e., protecting the data from damage. The basic approach to this problem in the environment discussed here is to provide that copies of user files are available for retrieval in case of damage—that is, provide backup. Gener-

Figure 1 Zipf's law for data set accesses



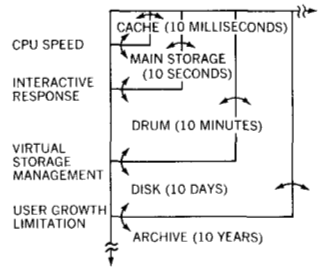
ally, either the installation management undertakes the responsibility for creating these copies on behalf of the users or the users are required to do it themselves. If the installation does the job, the approach is simply to copy files on a regular basis, perhaps weekly, keeping the copies off line, usually on tape. Because the installation generally does not have good techniques for knowing which data sets require new backup copies because of changes, the usual approach is to copy all files. This can involve much needless copying, and the whole process can require hours—even for a modest sized on-line data base.

On the other hand, there are also difficulties if the user is left to backup his own files—even though he knows which files need backup. A user may make on-line copies of his data sets with slightly different names, thereby proliferating copies of the same data on on-line storage, or he may copy data sets to private tapes, thereby requiring more and more operator intervention for mounting, or he may do nothing. Either approach has its costs in equipment and time, plus the user's difficulty of keeping track of all of the copies. Doing nothing is potentially the most expensive method of all, since the user is exposed to the consequences of lost data. A number of these concerns, as expressed by a large group of computing system users, are included in a SHARE White Paper on future operating systems.[1]

approach to a solution

It has been observed in a number of different contexts that the usage of a collection of objects is not uniformly distributed across the collection. For instance, in the English language, some one hundred different words make up well over half the number of words spoken. All the rest of the language comes to less than half the words used. This observation, known as Zipf's law, as applied to data set references and accesses, is shown in Figure 1. A similar relationship applies for various components of computing systems if we choose the time frame appropriately. Consider a computing system as depicted schematically in Figure 2, which is structured hierarchically with different levels addressing different problems. Consider processor references to storage. The observation that successive references tend to be to the same or nearby storage locations is behind the successful use of high-speed cache storage in current processing units. A cache is much faster but much smaller than the rest of the processor storage. It is possible dynamically to select the contents of the cache so that most of the instructions executed by the processor refer to the cache rather than to the slower main storage. When reference is made to main storage the contents are copied into the cache and remain there for further reference until displaced by new data that are needed. The effect of this operation is to enable the processor to function almost as though it had a large amount of the high-speed storage.

Figure 2 Storage hierarchy residence times and corresponding problems addressed



Similarly if we consider all the instructions in a program, we find that many programs tend to stay in the same part of their program space for significant portions of their total execution time. Thus it is not necessary to have a whole program accessible to the processing unit at all times. Rather it is possible to load portions of the program into main storage as required while holding the rest of the program on some auxiliary storage device, such as a drum or disk. This fact about most programs has enabled virtual storage to share a given amount of processor storage among more programs than would have been possible if each program had to be loaded entirely. Thus a greater number of simultaneous tasks can be serviced, thereby improving the responsiveness to the interactive user.

The common feature of these two examples is that because of patterns of usage it is possible to design a hierarchical storage structure, such that portions of the data being used are moved between more and less accessible (and also more and less costly) storage locations. This enables the overall system performance to approach that which it would attain using only the more accessible (more costly) devices, while the cost tends to approach that of the less costly (less accessible) devices.

Observations on data set usage lead to similar conclusions about the manner in which data sets are used. The pattern of reference to data sets has the same locality as a program's execution of its instructions. A user works on a subset of files for a period of time and then stops. He may go on to some other project or may enter the program into production and no longer be working with the same source and test files. In any case, in an unconstrained environment, users often have a significant number of files that are not in active use, but which they do not wish to discard immediately. This observation leads to the proposal of a hierarchical scheme for file storage that is analogous—on a longer time scale—to that established by virtual storage systems for executing programs and data.[2]

It is at the interface between active data and archived data that we see an opportunity to remove a significant barrier to the productivity of the user and the growth of applications. The basic principles of our MVS Archival System (MARC) are as follows. A range of storage devices are available for the storage of user data. These vary in size and accessibility (and therefore, in cost). There is more of the less accessible storage available than that which is more accessible, as a cost reduction consideration. Required is a tool for managing this hierarchy so that data move from level to level in accordance with their usefulness. The quality usefulness has different meanings in different situations. In any case, the gist is that the most useful data should be most accessible. There are three levels in our hierarchy. The first is a

level at which the data are always accessible with negligible delays (the current on-line storage), called Level 0. Next there is a somewhat less accessible level, possibly requiring some delay in retrieval, which would contain data that have a significant probability of being referenced, but not as great as the probability of the data on Level 0. This is Level 1. Finally, there is the third level, which contains the archival data, data that go unreferenced for extended periods of time. This we call Level 2. The unit of data transfer is the *data set*. In the text, "data" and "data set(s)" are used synonymously.

definitions

Several definitions necessary to the understanding of MARC are now presented. We use the word *public* to refer to any on-line volume managed by MARC that contains user data sets. A *migration volume* is one that contains user data sets that have been moved from public volumes under the control of MARC. *Migration* is the process by which data sets are moved selectively from public to migration volumes. *Restoring* is the process that brings the data set back to public residence. *Backup* is the duplication of a user data set on a volume other than that on which the data set resides. (Backup volumes are usually off line.) *Retrieval* is the process of selecting or retrieving a specific backup copy of a data set and copying it back to public storage. The MARC *Work Element* (MWE) is a message block for communication between MARC and other tasks (users, jobs). The requesting task inserts the specifications of a request for work to be done by MARC, and sends it to MARC, which returns its MWE after processing with the contents altered to reflect the results of the processing.

The IBM Research Division activities in storage management of user files began in 1969, with a TSS/360 file migration and archiving system. Work continued in 1971 with an OS/360 data migration and staging system, and led to a VM/370 filespace (minidisk) migration and backup scheme in 1973. These three systems are now described.[3]

TSS/360. The TSS/360 migration system treated the individual data set or file as the unit of traffic.[4] The data sets were copied onto migration volumes from the on-line public storage, and the public copy was then erased. In its migrated form, the data set was accessible only by specific migration commands that were essentially a restore command and an erase command. If a background (batch) job were to refer to a migrated data set, it would fail. To use the data set in the ordinary way, the user had to restore it to the public storage volumes by issuing a specific command. While the user waited, the restoration process took place under the control of his task.

OS/360. The OS/360 data migration and staging work was done at the IBM Research Division laboratory at San Jose. One of the major innovations introduced was the concept of data ownership for all files, whether created by TSO interactive jobs or by batch jobs. The concept was implemented in OS/360 and formed the basis for the migration scheme. A series of utilities was created to be invoked by data base administrators to control the size and use of on-line storage in the OS environment. Utilities were also written to allow the user to have a migrated data set restored to available on-line storage. These generally ran in background mode with some time delays. As in the TSS/360 scheme, the user was required to take specific action to restore the data, and a batch (background) job would fail if it referred to a migrated data set.

VM/370. Undertaken next was the design and implementation of a VM/370 minidisk migration scheme. A *minidisk* in the VM/CMS system is an allocation of contiguous storage made available to a specific user for the storage of files. These are typically one, three, or ten cylinders of a direct-access storage device (DASD), and represent a unit of system allocation to the user in question. In the VM/CMS system, the CMS filespace or minidisk is the unit of migration traffic. For many users—those who use the system intermittently rather than every day—the normal state of minidisk is migrated rather than on line. The user need not be aware that the minidisk in question has been migrated.

As part of the normal process of beginning work on the system, certain of a user's minidisks are automatically accessed by the system. Other minidisks may be accessed specifically by command. In either case, if the minidisk is migrated, the reference causes the restoration process to take place, and an estimate of the expected time of availability to be given. The user need not wait for the process to be completed, but can do any work that does not require restoration of the minidisk. He can also choose to suspend work until the minidisk in question is available. In a background task, he can likewise specify that he will wait if the minidisk is not available. If the minidisk is already restored, his task proceeds immediately.

Instead of an environment in which on-line minidisk space is the controlling factor in limiting user access to the VM/370 system, we have been able to establish an environment in which users may gain access to the system as required, with the migration system efficiently controlling the amount of on-line space.

case history

Prior to the installation of MVS in 1975, storage management at the IBM United Kingdom Laboratory at Hursley, Hampshire, England, had been accomplished by a combination of user forecasting of space requirements and manual checking by opera-

tions administration for enforcement of space allocations. After installation of an on-line interactive MVS/TSO (MVS with Time Sharing Option) system, that method quickly proved inadequate because, as users tended to create new data sets, it became difficult for installation management to limit their creation. Management therefore began to search for techniques to control the allocation of on-line storage. Because of their experience in storage management, the assistance of programmers at the IBM Research Center at Yorktown Heights, New York was requested.

The availability of practical experience with MVS at the IBM Hursley laboratory and experience with other systems at the IBM Research Center facility at Yorktown Heights led the two locations to launch a joint project involving the design and implementation of an effective tool for controlling on-line storage under the new MVS operating system. The project was directed at solving problems that arise in both operating environments—the research-oriented one and the development- and production-oriented one. The effort, like its predecessors, was aimed at supporting the user's growing dependence on on-line files, while limiting the amount of hardware required and increasing utilization of the direct-access space allotted to such files.

Our experience with MVS has shown patterns of usage and data generation among the users that are similar to those observed in the past. The quantity of on-line data, beginning with the data already on hand from previous systems and brought over to MVS as part of the conversion process, has continued to grow as the system is used.

We have thus designed and implemented a tool for the management of the on-line storage resource based on the hierarchical concept previously described. We now discuss what a storage manager does to aid users of the system by reducing their concern about on-line storage management.

Architecture of on-line storage management

functional requirements

The major goal of the storage management scheme is to make storage available in a timely fashion as automatically as possible. Since an operator or data base administrator is not able to monitor on-line storage constantly, a non-automatic system would be subject to errors or oversights, and critical shortages of storage could result. The automatic migration system should include a storage monitor for reporting problems to the data base administrator as they occur and correcting them as far as possible.

Another key system requirement is data availability. A user must not have access to data restricted or impeded because the system moves them from one place to another. In addition, the system must not lose data in the process. Whenever a data set is moved, there is a possibility that the new copy may differ from the original due to error. Care must be taken to minimize this possibility and to avoid moving the data unnecessarily.

To maintain the usability of the system while attempting to improve utilization of the on-line storage resource, the system must have low impact on the user's way of working. There are acceptable penalties the user might have to pay for the additional file storage resources made available by the file management scheme. These include a finite additional delay in the allocation of a data set that has been migrated, and perhaps the unavailability of certain information about that data set and its contents without restoring it, such as the names of members of a partitioned data set. However, the user should not experience a job or request failure because a data set required for the request had been migrated.

In justifying the existence of the migration system, two factors are significant. The scheme must offer the installation control over the on-line storage resource for maximum utilization of this resource, and the operation of the migration system must not place an undue burden on other system resources, such as CPU or I/O devices. The migration system gives users and data base administrators another place to put data. Rather than having to erase files that can no longer remain on line, because the user has received a certain allocation of space and has completely used it and now needs to create new files, or because the entire on-line storage resource is full, the user or administrator can cause them to be migrated, thereby keeping them accessible while freeing the critical on-line resource. Thus, users can choose to erase at their leisure data they no longer need, similar to the way in which they handle other job-related housekeeping chores, rather than being forced into an unnatural daily activity driven by on-line space limitations.

Also, when an installation provides convenient and capacious storage for data, while providing data integrity, users come to rely on that storage system to the exclusion of their previous stand-by, the private tape or disk volume. With efficient migration and backup means available, users tend to employ private volumes only for transporting data from one system to another, thereby saving themselves and the installation time and resources.

We now explore these requirements in further detail. First, a centralized, very reliable, migration-restoration-backup proces-

sor is required. Then, a simple user interface with automatic migration and restoration is needed. Finally, for installation control, we must have tunable parameters to drive the process, and for flexibility, a variety of devices and data types must be supported.

migration

Several requirements present themselves. First, the migration process should be continuously ready to respond to system needs without having to be started to respond to a particular crisis. Second, this task, which is always in existence and ready to take action, should monitor storage. To enable the migration function to evaluate system needs, the installation should be able to define thresholds of utilization, such as the maximum acceptable ratio of storage used to the total. The migration process would then be triggered by these thresholds to take appropriate action.

The overall efficiency of the migration operation can have a distinct impact on system performance. It should anticipate needs, in order to key such elective activities as the outward movement of data as much as possible to periods of light system load, to minimize its impact on users. Provision should be made to detect any unusual use of the system during normally lightly loaded periods. A judicious choice of thresholds and the availability to the migration process of valid measures of system loading would facilitate such scheduling.

Another efficiency problem is that of unnecessary data movement. The underlying assumption of the migration process is that the user works his way from place to place in the files that he has created and stored on the system. Experience indicates that over a small period of time, one uses only a small fraction of all his files. The particular fraction in use changes as time passes, but the changes are generally gradual. The migration process attempts to determine the files that are not part of the group that the user is currently working with, and move these to migrated storage. Thus the process leaves more room for new files that the user creates. To the extent that the estimate of the migration process is accurate, the user in question can function for some time without referring to the files that have recently been migrated. If the estimate is not accurate, or if there is simply not enough on-line storage to satisfy the moment-to-moment needs of the active users, the users will be constantly requesting data sets to be restored soon after they have been migrated.

One of the most unfavorable situations in this kind of operation is that in which, after a migration of several hundred files during an off shift, half the migrated files are restored to on-line storage through user requests during the first few hours of the next prime shift. Given that there is in fact enough on-line storage for

the users, then this situation represents an obviously inefficient selection of files for migration. A key problem in the migration process is to predict future usage from past usage. We currently choose as a criterion of a data set's usefulness the length of time in days since it was last used. Experience has shown that this can be an effective criterion but research is continuing. One of the major problems in a choice of criteria is the evaluation by the processing programs that select data sets for migration. The up-to-date data necessary for the evaluation must be available to the program on an individual data set basis at the time of the selection. Another consideration is the expense incurred in collecting the data for evaluation. The chosen criterion is based on information that is easily collected and readily available to the processing program. It also seems to reflect usage in a research environment adequately.

restoration

Restoring a data set from migrated storage space to on-line space should not require prior knowledge on the part of the user that the data set has been migrated. Thus restoration must be triggered by a reference, whether by job control language of a background job or by an allocation request from TSO. The user should, however, also be able to request a restoration directly if it is known that the data set has been migrated and the need for it is anticipated. The user may wait for the restoration if the data are needed immediately. Otherwise, notification is given when restoration has been completed.

backup

The backup facility should be easy to use, reliable, and efficient. To improve the reliability of the operation, the backup process can be integrated with the migration process, so that these inherently similar functions can be carried out in an economical way by common programs. For the sake of economy, efficiency, and feasibility, only data sets that have been changed since the last backup should be copied into the current backup. The installation also should be able to specify the frequency at which this backup process is to take place, with provision having been made for specific data sets for which handling might vary from the system norm. In addition to a global outlook for purposes of system-wide integrity, the backup function should also afford the user the opportunity for timely backup of files at more frequent intervals. There would be management-defined upper limits on the number and size of copies a user could keep in the backup system.

Control and ease of use are needed in any function that affects one's data. To this end, the migration process should afford the ability to restore and migrate by command in a knowledgeable planned way, in addition to letting the system operate on one's data. The fact that the user need not wait while the actual data movement is carried out would make it convenient to plan one's

work and, knowing the patterns of usage of one's own files, to position the data in a way that minimizes some cost function. This might be as simple as taking advantage of lower costs for off-line (migrated) storage or attempting to operate in an environment of limited on-line storage availability.

The system should maintain the necessary information to support the user's interest in his files. This would include at least the identity of the files, their characteristics, and their current status.

So that one may plan one's time efficiently, migrated files are organized in such a way as to minimize the waiting time for files to be restored. Our experience confirms that the most recently migrated files are most likely to be requested on an individual basis. On the other hand, over time, users are likely to request their files in groups, without regard for the date of last use, such as all the files for a particular project or source and test data for a program. Such patterns suggest that files should be arranged in reverse chronological order for a period of time and then be sorted by owner.

reliability

In addition to automatic operation, a key requirement in any data handling facility is reliability. A reliable restore function is required to make data available.

In MVS, the unit of work processing is the *task*. If a task experiences severe difficulties in carrying out a request—difficulties from which the system cannot determine how to recover—the task will be terminated. To maintain reliability and availability, therefore, the actual execution of individual requests should be carried out on a one-request-per-task basis. These tasks are under the observation and control of a dispatching task that is aware of tasks failing. The dispatching task then tries to correct the problem, or at least prevent it from interfering with the processing of other requests.

An emergency mode of operation allows the system to operate with current on-line files. This enables the system programmers fixing a migration or other on-line storage problem to use the system.

installation management

Having considered the user, we now discuss installation management. The installation must be controlled to maintain a system with acceptable cost for equipment. To that end, the installation should be able to select devices on which to locate the data base. These devices would be the various disk drives—IBM 3330, 3340, and 3350—with the IBM 3850 Mass Storage System as a data reservoir and archive. In addition to device flexibility, the installation requires control of the operating characteristics

of the migration and backup processes. To that end, the individual installation must be able to specify certain parameters, such as thresholds of available space on the on-line volumes, frequency with which the migration process is carried out, criteria for determining currency or lack thereof, and frequency and number of concurrent backup copies. By choosing its own values for these variables, an installation can tune the migration process to its needs.

In addition to control and flexibility, the installation must have information, both for billing purposes and for evaluating policies and strategies. The migration processor must provide statistics on data usage, including accounting information such as on-line and migrated storage residencies and the amount of data being moved for various purposes (e.g., backup, restore, migrate). The costs incurred by the migration process itself must be accurately assessed and charged back to the functions and users being served.

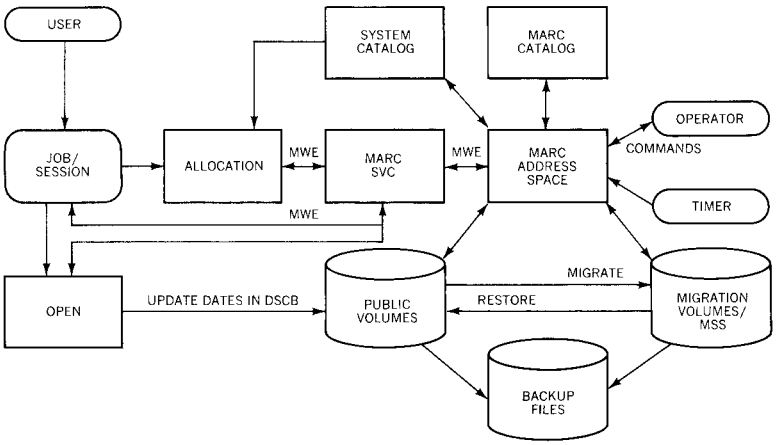
Design and implementation

The design of MARC was undertaken to meet the following specifications. Reliability and availability are essential. Operational flexibility, as provided by user and installation-specified controls, is important. The functions should be easy to use, and data on the operation of the facility must be accumulated and presented in intelligible form. A final practical consideration is that there should be as few changes to the MVS system as possible to provide the needed functions.

MARC is a system that runs in its own address space. For reliability, processing is divided into units that are assigned to subtasks. In addition, most of the processing is carried out in problem state, i.e., without using the privileges of system code. This reduces the possibility of overall system damage due to a MARC malfunction, and reduces the probability of a global MARC failure due to an error in processing a particular request. The separate address space with subtasking is probably the most significant design feature. In addition, the code is designed in modular fashion, with specific functions assigned to specific modules, and an overall hierarchical approach has been taken in specifying the flow of control.

Bearing in mind the definitions presented earlier, we examine the structure of the MARC facility installed in the MVS system, as shown in Figure 3. The center of the facility is the combination of MARC address space, public volumes, migration volumes, and backup files. This combination represents the site of data movement and the control of processing. Tasks that operate in the

Figure 3 MARC in the MVS system



MARC address space move data from one location to another in a public-migration-backup storage volume triangle. Data are moved in response to requests either by MARC (with reference to installation-specified criteria) or from outside the MARC address space.

The results of the processing that takes place under MARC control are recorded in both the system catalog and the MARC catalog. The means of communication between other address spaces (users) and the MARC address space is the MARC Supervisor Call (SVC), which verifies MARC Work Elements (MWEs), communicates them to MARC, and transfers the results back to the user.

migration

It is possible for the installation, through either the system operator or the designated data base administrator, to define thresholds of occupancy called "high-water mark" and "low-water mark" for each of the public volumes that are managed by MARC. The high-water mark is the maximum fraction of the volume that should be occupied before MARC takes corrective action (migration). The low-water mark is the amount to which the fraction occupied should be reduced before MARC migration processing terminates. It is possible to instruct MARC to check the public volumes for compliance with these specifications either at a specific time of day or at regular intervals throughout the day, and take action as required.

In addition to those data movements that are automatically initiated by MARC, data are also moved in response to specific requests for processing.

operator/user interface

Explicit or implicit requests for MARC processing originate either with the operator or with the user. Only explicit requests are discussed.

The operator commands fall into the following categories:

- Operational control: specification of operating parameters to MARC (migration criteria, frequency of migration, etc.); suspension and resumption of specific MARC operations, for example migration; management of entries in the MARC LOG data sets.
- Information listing of the contents of the MARC catalog and the values of various parameters set by the operational commands.
- Data base specification: description of the volumes to be managed by MARC, e.g., on-line or migration volumes; specific MARC requests, e.g., MIGRATE data set or RESTORE data set.

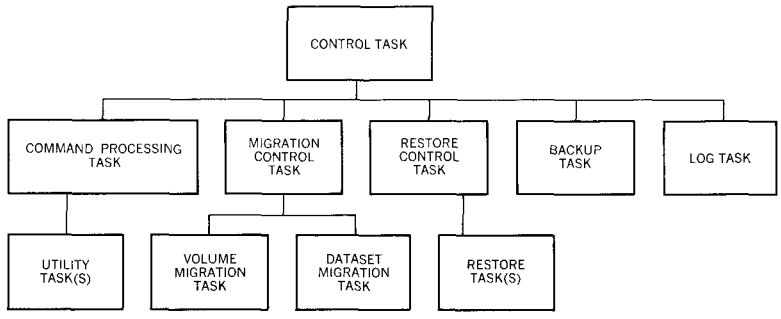
The TSO user has analogous functions for specifying data set movement. Specifically, he can migrate on-line data sets (MDS), restore (RMDS), or erase (EMDS) migrated data sets. He can also create (DUPLEX) and retrieve (RBC) backup copies on a data-set basis, and obtain information about his migrated data sets (LISTM). From a TSO CLIST, a QDS command returns a condition code that indicates whether a specific data set has been migrated or not. In addition, when previously authorized to be a data base administrator from the operator's console, a user may specify any of the operator commands from any TSO terminal (MCMD).

catalog

The MARC catalog is the repository of all information specific to MARC processing. It is a VSAM key-sequenced data set; i.e., the catalog is indexed and records can be retrieved by specifying a unique key. Among the records contained in the catalog are the following:

- Data set entries—contain information about migrated data sets, including name, size, organization, date last referenced and date migrated. Information about available backup copies for a data set is also recorded.
- Backup copy entries—give the names of the copied data sets and the date of each copy.
- Volume records—describe all the volumes for which MARC is responsible. The data recorded include volume serial number, type of volume (public or migration), and the thresholds (high- and low-water marks).
- User record—contains information specific to the users, e.g., whether a user has been granted operator privileges (designating him as a data base administrator).
- Backup volume records—identify the volumes that are used for backup.
- Control records—contain information that controls the action of MARC, e.g., time of day for automatic migration, backup, etc.

Figure 4 The MARC address space and control structure



- Statistics records—include records of MARC processing (number of data sets migrated and/or restored, number of unsuccessful requests) on a daily basis and per volume.

migration volumes

The design of MARC includes two basic types of migration volumes. The first is the real direct-access storage device (DASD), as exemplified by IBM 3330 and 3350 disk storage systems. The second is the virtual DASD, provided by the IBM 3850 Mass Storage System (MSS).[5]

When a real or virtual DASD volume is to receive migrated data, access to that volume is through the MARC processor only, and the user has no contact with data on the migration volume. We therefore organize the data on the migration volume in the most efficient manner. The first record is a data set descriptor that contains the information in the system Data Set Control Block (DSCB) for the public data set. Small data sets (containing less than two tracks of data) are packed into a large VSAM data set. Thus several data sets can be packed onto one track. For example, a data set with twenty 80-byte records would occupy the minimum unit of space allocation—one track. Packed, it occupies only one record in the VSAM data set, a fraction of a track.

MARC address space

The MARC address space is the heart of the MARC facility, and carries out all the data movement. Figure 4 shows organization of MARC tasks. A *control task* initializes and terminates MARC processing in response to operator commands. At other times the control task functions primarily as a dispatcher. It receives the requests for service from other tasks (users/jobs) and from the operator and dispatches them to appropriate subtasks. The control task monitors any subtasks that terminate abnormally. Because it does no data movement and relatively little other processing, the control task is isolated from the problems that might occur in the handling of specific data sets and is able to maintain the operation of the function as a whole.

A *migration control task* responds either to specific requests or to installation-specified criteria, and initiates migration tasks as required. Migration can take place on either a data set or volume basis. In migration, a data set is copied to a migration volume. The system catalog is then updated to reflect the migrated status, and the public copy is deleted.

A *restore control task* performs the analogous function for restore requests. Here, a number of simultaneous restore tasks can be active. In restoration, the data set is copied back to a public volume. The system catalog is updated to include the new location, and the migrated copy is deleted.

A *backup control task* supervises the backup and retrieval functions of the MARC facility. Automatic backup on a volume-by-volume basis is governed by installation criteria of frequency and time of day. At the specified time on those days on which automatic backup is to be done, the control task requests the copying of the changed data sets from each installation-specified volume by a backup volume function. The control task also responds to requests for backing up individual data sets by means of the DUPLEX command, and retrieval requests that originate in the command to retrieve backup copies.

A *log task* writes records of MARC processing to the MARC log data set, and if the installation specifies, also records all changes to the MARC catalog in a journal for catalog recovery purposes. The records to be written are prepared by the parts of MARC that are doing the processing and are sent to the log task for recording.

A *command processing task* processes operator commands that do not require actual data movement. Commands that require data movement are sent from this task to the appropriate migration, restore, or backup/retrieve control task for processing.

Mass Storage Systems in the MARC environment

The IBM 3850 Mass Storage System (MSS) is a large capacity direct-access storage device[5] in which data are stored on cartridges. Then, for purposes of access, pairs of these cartridges are linked into logical units, called virtual volumes, which appear in every way to the programming system as 3330-1 DASD storage volumes. A process called staging is used to move data from the MSS tape cartridges to real 3330 DASDs, called staging volumes, for access by the central processor. Destaging returns the data sets to the MSS cartridges when they are no longer in use, thereby transferring any modifications back to the cartridges.

The MSS is a reservoir of storage capacity that the MARC function uses in a number of ways. First, virtual volumes are used as migration volumes for Level 2. Data sets are moved to MSS volumes after a period of residence on Level 1, which may be bound virtual volumes or real DASD, without being used. Level 2 volumes are sorted by owner, such that a given user's files are put on the same MSS volume. The availability of Level 2 migration volumes that do not require operator mounting has greatly improved the responsiveness of MARC to requests for data sets that have not been used for a long time. In addition the large reservoir of volumes available in the MSS enables a fine degree of sorting in allocating Level 2 volumes. Instead of assigning one Level 2 volume to all users whose user IDs begin with a letter between A and C, for example, we can allocate individual volumes to each letter of the alphabet individually. This increases the amount of space set aside for a particular user's data sets and reduces the number of volumes on which his data sets are stored.

In this first phase of our MSS activity, access to data on MSS volumes that are managed by MARC is still exclusively through MARC. Upon allocation of a migrated data set resident on an MSS volume, the data set is copied to real DASD for access by the requesting program.

In addition to its use as a source of migration volumes, the MSS is especially valuable as a source of backup storage space. The incremental backup system creates copies of changed data sets on a regular basis, with an installation-specified upper limit on the number of copies of an individual data set to be simultaneously maintained. Our installation value is currently set at 4. If an average of between five and ten percent of the on-line data sets are changed per day, after ten to twenty days, on the average, the backup storage reaches the size of the on-line storage. Beyond that point it continues to grow. Since the same data sets continue to be changed and old copies are deleted, the growth rate slows down and—at some steady state—grows only with the amount of new data being generated. The MSS provides the capacity to retain backup copies for long periods of time.

storage hierarchy

MARC is designed to improve the utilization of the storage components of the system configuration. MARC seeks to establish a hierarchy of data set storage devices among which data move as required to improve the efficiency of the whole configuration. All the DASD devices used as MVS storage volumes are seen as components of a continuum of storage, with varying values of capacity and access time. These values range from milliseconds (to access a data set residing on a permanently mounted real 3330 DASD) to minutes (to stage completely a large data set that is resident on an MSS cartridge). The capacities range from 100 million bytes for a 3330-1 DASD volume to 472,000 mil-

lion bytes on the MSS. Algorithms have been developed to evaluate the placement of data sets in such a hierarchy, depending on such variables as device access time, data transmission rate, mean number of bytes transferred per access, mean number of access per day, cost of the various devices, and size of the data set.[6] Factors that appear to be significant in an interactive environment and are not generally addressed in these algorithms are the cost of the time of the interactive user waiting for a response, and the effect on user productivity of unforeseen delays in processing. Truly effective use of the storage hierarchy requires that data be collected on the variables that are observable within the machine, and that algorithms be tested as they are developed.

For the present, there are some immediate gains from even rudimentary observation of data set usage patterns, using simply the information that is contained in the date last referenced. This criterion has been adequate to make the vast majority of data references to data sets that are stored on real DASD. The selective copying of data sets to the MSS, based on the length of time since last use, reduces the amount of real DASD space occupied by inactive data sets and allows room for the creation and use of active data.

There are several areas in which MARC can make more effective use of the MSS as a device than is currently being done. In the backup process, MARC issues an MSS ACQUIRE instruction that requests that a large block of space be staged from the cartridge to the staging drive. This space can then be used to make copies of several different changed data sets without having to access the cartridge for each individual copy. This can also be done during migration.

In summary, the Mass Storage System offers MARC a very large number of available volumes whose access is completely under program control. Eliminating the wait for volume retrieval from a library and physical device mounting by an operator has greatly improved the responsiveness of MARC to requests for either Level 2 migrated data sets or backup copies. Such requests might be even better handled if MARC could be extended to use expected patterns of reference to arrange data on the MSS volumes. Such a logical arrangement would result in more data accessed per cartridge accessed and more efficient use of the system.

On-line storage management

In addition to the functions previously described, other functions should be performed by an on-line storage manager. We have discussed the selecting of data sets for migration based on

criteria of permanent usefulness. There might be many temporary data sets for the duration of a job or terminal session that should included in the migration process. Since system and utility temporary data sets are often created but not always deleted when their use has ended, MARC provides for installations to describe automatically deleted data sets. The automatic deletion category includes system temporary data sets (created by the system for scratch space) and uncataloged or improperly cataloged data sets. Also, other data sets may have a defined usefulness of short duration. For example, listings produced by language processors (such as FORTRAN and PL/I) might be considered expendable. MARC, therefore, allows the installation to specify data sets to be scratched after an installation-specified period of time. These two features help to free on-line storage from the debris of processing.

Another situation that impedes the full use of on-line storage in an MVS environment is the fragmentation of available storage into small pieces. As an example, if a volume has 2900 tracks available, but the largest contiguous grouping (extent) is 18 tracks, a request to allocate one cylinder (19 contiguous tracks on a cylinder boundary) will fail. If the available storage space had been less fragmented, the request could easily be satisfied. As a first step toward avoiding excessive fragmentation, we have implemented a measure of fragmentation for disk volumes. MARC can thus detect severe cases of fragmentation in the course of its space monitoring.

A significant problem for the owner of many migrated files has been that of determining the contents of a file without having to restore it. A possible solution has been suggested by our work on VM/370. Provision could be made in the MARC catalog for appending to each data set entry a set of keywords that describe the contents of the data set. A search facility on keywords could present to the user a list of his migrated data sets whose keywords match the keywords of the search. The user could then decide whether or not to restore the data sets.

Concluding remarks

MARC has been operating at IBM Hursley and Yorktown Heights since early 1976, and at the IBM facility in Burlington, Vermont since September, 1976. As of January, 1977, there were about eleven thousand migrated data sets at Hursley, and about twelve thousand at Yorktown Heights. The Level 0 on-line storage at Yorktown Heights, for example, contains approximately two thousand three hundred data sets.

Implementation problems that we have experienced are mainly those of understanding the MVS system and the interfaces between it and our code. In addition, as a result of the number of subtasks that execute in the MARC address space, we have had problems in correctly sharing serially reusable resources. To avoid a deadlock between two tasks whose needs might be inextricably entwined, we have had to pay close attention to the order and type of enqueuing of the data sets and other resources that are used by MARC tasks.

ACKNOWLEDGMENTS

We are grateful for valuable discussions with A. Katcher in the design phase of this project, and for the constant encouragement and support of A. Weis, I. W. L. Jones, and L. Brown of the IBM United Kingdom Laboratories in Hursley, England, and that of N. Pass and Helen C. Dietrich of the IBM Thomas J. Watson Research Center at Yorktown Heights, New York. We also acknowledge the useful suggestions of the reviewers.

CITED REFERENCES

1. SHARE Systems Group, *White Paper on Future Systems*, SSD 227, 43–46 (October 1, 1972).
2. R. C. Daley and P. G. Neumann, "A general-purpose file system for secondary storage," *Proceedings of the Fall Joint Computer Conference, November 1965*, 213–29, Thompson Book Co., Washington, D.C. (1965).
3. R. P. Kelisky, "Managing interactive systems for user effectiveness," *Lecture Notes in Computer Science, Interactive Systems*, 93–107, Springer-Verlag, Heidelberg (1977).
4. J. P. Considine and A. H. Weis, "Establishment and maintenance of a storage hierarchy for an on-line data base under TSS/360," *Proceedings of the Fall Joint Computer Conference, November 1969*, 433–40, Thompson Book Co., Washington, D.C. (1969).
5. IBM System Reference Library, *IBM 3850 Mass Storage System (MSS) Installation Guide*, Order No. GA32-0030, IBM Corporation, Data Processing Division, White Plains, New York 10604.
   IBM System Reference Library, *IBM 3850 Mass Storage Systems (MSS) Principles of Operation*, Order No. GA32-0029, IBM Corporation, Data Processing Division, White Plains, New York 10604.
   IBM System Reference Library, *OS/VS Mass Storage System (MSS) Services, General Information*, Order No. GS35-0016, IBM Corporation, Data Processing Division, White Plains, New York 10604.
   IBM System Reference Library, *OS/VS Mass Storage System (MSS) Services, Reference Information*, Order No. GC35-0017, IBM Corporation, Data Processing Division, White Plains, New York 10604.
6. V. Y. Lum, M. E. Senko, C. P. Wang, and H. Ling, "A cost oriented algorithm for data set allocation in storage hierarchies," *Communications of the ACM* 18, No. 6, 318–322 (June 1975).

Reprint Order No. G321-5059.