To: Distribution

From: T. H. Van Vleck, S. H. Webber, A. Bensoussan

Date: 08/07/74

Subject: Implementation of Proposed New Storage System

This memorandum presents the implementation choices for the proposed new Multics Storage System described in MTB-055.

The reader is assumed to be generally familiar with the operation of the current Multics Storage System.

REVIEW

Five problems with the current Storage System were identified in MTB-017, and five goals proposed. These were:

1. PROBLEM: The Storage System loses information.
   GOAL: Eliminate loss of information by reducing the number of crashes, by limiting the damage done by crashes, and by minimizing loss of information during recovery procedures.

2. PROBLEM: Backup and recovery procedures cost too much.
   GOAL: Minimize system down time and devote fewer resources to backup functions.

3. PROBLEM: Large amounts of storage cannot be handled.
   GOAL: Make extremely large storage configurations usable without imposing a penalty in performance, reliability, or availability.

4. PROBLEM: Several desirable features should be added, including support for removable disk packs.
   GOAL: Add support for removable disk packs and other features.

5. PROBLEM: The operator interface is deficient.
   GOAL: Improve the operator interface, especially in the areas of shrinking and expanding the device complement, operating a crippled system, and providing recovery information.

Overview of the Proposal

The physical storage available on a Multics configuration will be grouped into partitions, as it is now. The MULT partition will be further subdivided into logical volumes, which may consist of part of a physical volume, or several physical volumes. All physical volumes which comprise a logical volume must be mounted or dismounted at the same time, so that a logical volume may not be partially mounted; but logical volumes can be added to or removed from the MULT partition while the system is running. A physical volume may contain storage for only one logical volume; the reason for allowing "fractional" physical volumes is to accomodate the DUMP, LOG, SALV and BOS partitions without requiring that the minimum system configuration have two volumes.

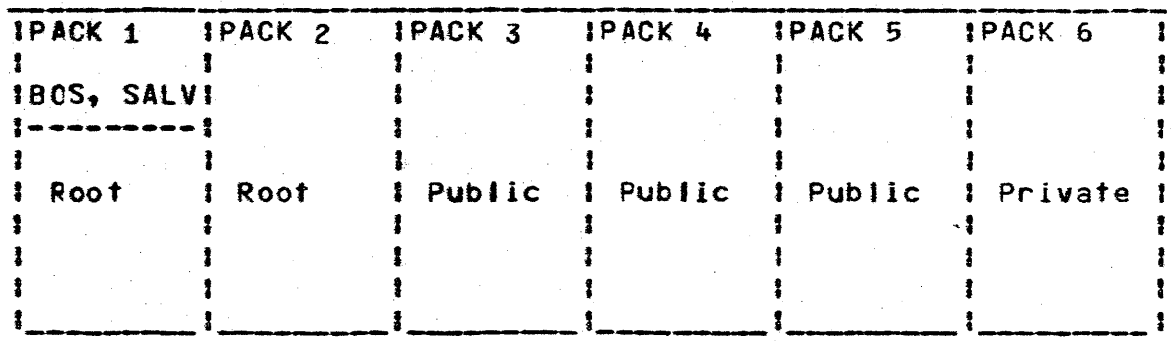

Figure 1: Example of Physical Volume Usage

Every segment in the new hierarchy will have all its pages allocated on the same physical volume. All segments in the same directory will be contained in the same logical volume.

Each physical volume has a label, recorded by a special BOS utility, which describes the storage extents on the volume and the name of the logical volume it provides storage for. A physical volume is part of only one logical volume. Each physical volume has a volume unique identifer (VOLUID), used by the system to identify the volume.

Each physical volume in the new Storage System has a Volume Table of Contents (VTOC) which contains an entry for every segment on the volume. The VTOC entries contain the information, formerly present in the directory branch, which describes the physical storage occupied by the segment. All pages of a segment will reside on the same volume. Each volume also contains a Volume Map, which has an entry for each page on the volume describing its current status.

There is no FSDCT in the new Storage System. The Volume

Maps and the configuration deck provide the information which used to be contained in the FSDCT. Volumes listed in the configuration deck are called permanent volumes, and cannot be dismounted. All directories must reside on one permanent logical volume designated in the configuration deck; no directory may ever be off line. The logical volume which contains the root may consist of several physical volumes; and a configuration need have only this one logical volume defined.

The directory branch and the VTOC entry for a segment are connected by a VTOC pointer stored in the branch. This pointer is a pair of 36-bit quantities which specify the VOLUID and the location within the VTOC where the VTOC entry resides. The VTOC pointer's second component, the VTOC index, is only interpreted within the context of the specified volume. Both the branch and the VTOC entry contain the unique ID of the segment, and both unique ID's must match if the system is to consider the association valid.

The system will maintain a table in wired-down storage known as the Device Table, which has one entry per disk drive in the configuration, specifying the VOLUID for the volume mounted on the drive, the DIM parameters necessary to run the drive, and other data.

The system will also have a more extensive ring 1 data base which registers each logical volume known at the installation, and lists the physical volumes involved, the volume owner, and provides an access control list for mounting control.

DESIGN STRATEGIES

The focus during the initial design of the new Storage System will be on getting a version which runs and is functionally correct, in as short a time as possible. Adequate system performance is also an important functional requirement. But some functional extensions will be postponed to later phases of system development in order to get the new system on the air quickly.

Data Bases

Wherever possible, data bases will be modified in a compatible fashion, leaving previously-defined items where they were. For example, the directory branch need not change size: the removal of file maps will be compensated for by the addition of a VTOC pointer, but no attempt will be made to (say) re-structure ACL's.

Supervisor Calls

It is hoped that all current supervisor calls will continue to function exactly as they do now, with one or two exceptions. A few new entries will be added, and one or two new status codes may be possible (for example, "logical volume not on-line").

Algorithms

Straightforward code will be much easier to debug and maintain, so our preference will be to implement the new Storage System with less mechanism rather than more. This is especially true for the first phase of the implementation.

For instance, the current system has a fairly complicated mechanism for allocating variable-sized file maps in the directory. When file maps are moved to the VTOC, this strategy will be eliminated, and each VTOC entry will contain space for the maximum-size file map.

Security Considerations

The additional security controls described in MTB-086 will be supported by the new Storage System. Care will be taken to insure that no new ways of communicating between users of different access authorizations are introduced.

In order to prevent unauthorized communication between users with different attributes, by means of quota manipulation or signalling by mounting and dismounting, logical volumes which can

be dismounted may contain segments from only one sensitivity level and category set. The level, category set, and minimum ring number for each dismountable volume is kept in the on-line logical volume registration data, and is also recorded in the volume labels.

To guard against accidental disclosure of information in the event of a system failure, the new Storage System takes considerable care to avoid re-used addresses. Also, all free pages on disk are explicitly required to be zero, so that even if an unused page is mistakenly added to a segment as a result of a system crash or a dropped bit, the system will not compromise security.

Sizes of Data Fields

One problem the new Storage System solves is that of providing for much more physical storage in a Multics configuration than the current supervisor can handle. Part of the current problem arises because we wish to support future hardware enhancements which may provide storage devices with much larger capacity. The recent change to the whole Storage System and to BOS needed to support DSU-191's was able to find a free bit: the next such change would require restructuring of many system data bases.

For this reason the disk record address is being changed in format. The current address is

```
device ID      bit (4)
device address bit (18)
```

where the device ID, ranging from 1 to 7 (0 is used for null addresses, and the high-order bit indicates that the page is on a special device, e.g. the paging device), specifies the storage subsystem (DSU-191, etc.) according to a table in the FSDCT.

The new address format expands from 21 bits to 36. It looks like this:

```
device table index bit (18)
record address     bit (18)
```

The old "device address" coded both disk drive number and address on the disk pack into 18 bits; this has been changed so that we have an effective width of 36 bits, with the device table index used to select the proper disk drive and the record address being strictly an offset within the volume. The "device ID" is located in the device table, and selects the strategy and coding scheme used to run the device. The new address format provides for up to 256K devices on-line, each device having a capacity of 256K records. Since the current capacity of a DSU-191 pack is about

20,000 records, we are prepared to support a tenfold increase in the capacity of a single pack; if devices with more capacity are produced, we can define several logical volumes on one physical volume. The total amount of storage which can be supported by the system increases by a factor of 32K, to about 281 quadrillion characters.

The disk record address is never interpreted except in the context of its own volume. Different volume-addressing schemes and VTOC layouts could exist compatibly within the same configuration on different volumes.

The "VTOC index" stored in a branch is used at segment activation time to locate the correct VTOC entry on the volume. Like the disk record address, this number can be interpreted only in the context of the volume it refers to. The VTOC index might be coded as a record address on the volume plus an offset, or as a subscript in a fixed-length array, or as some sort of hash address into the VTOC. Making this field 36 bits wide insures that whatever clever coding scheme is used will have enough bits available.

Command Changes

The list and status commands should have options to list the logical volume on which a segment resides, and to indicate whether a segment is on-line. Some redefinition of the items printed by the default invocation of the list command would be a good idea, to insure that the command will reference only the directory unless the user explicitly requests otherwise.

An active function "on_line" would be useful for checking whether a segment is currently on-line.

A new command "set_vol" and an option to status to return the logical volume ID will be needed to handle the volume ID associated with each directory.

New commands are needed to request the mounting and dismounting of volumes.

New Supervisor Entries

A new hcs_ entry must be provided, or status_ modified, to return the name of the logical volume on which a segment resides. New calls are also necessary to set and get the logical volume ID associated with a directory.

ALGORITHMS

This section describes the sequences of operations performed by the new Storage System for various system functions. Each function is describes in terms of its differences from the current Storage System function.

Branch Creation

When append is called to create a new branch, it must determine the correct volume for the storage associated with the branch and allocate a VTOC entry on that volume. The VTOC pointer to the new VTOC entry is then stored in the branch.

In order to create a segment, a user must have append permission on the parent directory and meet the usual validation level and security level constraints.

To determine the volume, append obtains the logical volume name from the directory header. If more than one physical volume is a member of the logical volume, the physical volume with the most free space is chosen to receive the new segment (unless the volume has a switch set which makes it appear "full," as may happen when a logical volume is being compressed).

Once the volume is determined, append locks the directory and allocates the branch as it does now. Next, append calls the VTOC_manager to request the creation of a VTOC entry on the appropriate volume. If the VTOC for the chosen volume is full, append returns an error code and does not create the branch.

The VTOC entry is initialized by the VTOC manager when the entry is allocated. Once a VTOC entry has been allocated, modifications to the VTOC entry are adequately protected by the parent directory lock.

Making a Segment Known

Making a segment known does not require a reference to its VTOC entry.

Segment Fault

The system's processing of a segment fault has two parts: first, the supervisor determines whether the segment faulted on is active. If so, it is only necessary to connect the SDW for the segment to the page table and return. If the segment faulted on is not active, it must be activated. To determine whether a segment is active, the supervisor will obtain the unique ID of the segment from the KST entry and search the AST for the

segment.

Activation

To activate a segment the supervisor obtains the parent directory's segment number and the location of the segment's branch from the KST, locks the parent directory, and then calls VTOC_manager to cause the VTOC entry for the segment to be read into a wired buffer. Next, the system locks the AST and obtains an ASTE of the appropriate size. (This step may cause some other segment to be deactivated.) The ASTE is filled in from the VTOC entry and the branch.

When the page table is being filled in, the system may encounter null addresses in the file map. These are represented as PTW's with a "null address" flag on, which the system will check at page fault time.

Page Fault

When a process encounters a page fault, the supervisor checks the PTW being faulted on to see if the "null address" flag is on. If not, the disk record address from the PTW, together with the Device Table index in the AST entry, is used to generate a disk address for the device I/O.

If the supervisor encounters a fault on a PTW with the "null address" flag, the supervisor will give the segment a block of zeroed core.

When a page is being written out, the supervisor will examine the page to see if it is all zero. When the current supervisor detects this situation, it does not write the page, but simply frees the disk address. This behavior is thought to be the cause of many of the re-used address problems which the current system encounters. In the new Storage System, zero pages will be written back to disk, and a "zero page" flag set in the PTW. Pages with this flag on will be freed at deactivation time, that is, when the VTOC is updated to reflect the fact that the record is no longer being used by this segment. This strategy insures that all free records on disk are zero, so that damage to a disk pack is much less likely to introduce old information into a file.

When a page is to be written out, if the "null address" flag is on, a disk record will be assigned on the appropriate volume. If the page is all zeroes, of course, this step can be eliminated.

Bounds Fault

Bounds faults will be greatly simplified in the new Storage System. Since all file maps are full size, there is never a need to re-allocate a file map in the middle of bound fault processing. This means that a bound fault need only re-allocate AST entries. Since the ASTEP has been removed from the branch, a bound fault does not need to modify the branch, and therefore the directory need not be locked.

Deactivation

When a segment is deactivated, any disk records corresponding to PTW's with the "zero page" flag will be released. The data in the ASTE are written back to the VTOC by VTOC_manager, which does not lock the parent directory. A system performance improvement can be expected for deactivation since the branch need not be referenced: this change eliminates the paging in and writing out of the directory page(s) for the branch. In addition, the references to the directory header page for the locking and unlocking operations on the parent directory are eliminated.

Making a Segment Unknown

No change is made to makeunknown.

Truncation

When a segment is truncated, the pages of the segment will be explicitly zeroed and the zero pages written out to disk.

Deletion

When a segment is deleted, it is first truncated, to insure that the disk pages it occupies are zeroed. The sequence for deleting the segment is:

```
lock directory
delete branch
call VTOC_manager to delete VTOC entry
unlock directory
```

It may be possible to unlock the directory before calling VTOC_manager.

Directory Modifications

When the system crashes while a directory is being modified, the salvager frequently finds the directory in an inconsistent state. It is possible for the salvager to do a great deal of damage to the hierarchy in the attempt to correct the directory. What seems to happen is that a directory is locked, and the supervisor starts making some change in the directory, for example adding a new ACL entry, and gets far enough in this operation so that the directory is inconsistent when the directory page is removed from core in the normal course of core management. If system operation is then interrupted, the page of the directory which is on disk must be repaired by the salvager before the directory can be used. Core pages which are flushed to disk by emergency_shutdown may also lead to this situation. In the current storage system, if emergency_shutdown succeeds, all potentially inconsistent directories can be detected by examination of the ASTEP in the branch and the lock in the directory header. The new storage system eliminates these items, and might appear to make the job of the salvager more difficult.

It would be far better from the point of view of reliability if the inconsistent directory pages were never written to disk. This might occasionally cause operations which the user thought had completed just before a crash to be lost, but it would mean that for almost all system crashes, no salvaging of the directory structure was necessary.

To accomplish this goal, the operation of locking a directory will set switches honored by page control which will prevent any pages of a directory which is locked from being written out to disk. (The pages may be claimed if they have not been modified, and if a paging device is available the pages may be moved to the paging device.) These switches must be respected by page control and emergency_shutdown. (If pages of locked directories may go to the paging device then the salvager must respect such a switch in the paging device map too.)

Paging Device Management

No significant changes are planned to paging device management.

Volume Mounting

When a user wishes to request the mounting of a logical volume, the pattern works somewhat like that proposed for tape. The request is validated by ring 1 and passed to the system control process, where a message is typed to the operator. When the operator has mounted the volume, he issues a command to inform the system that the physical volume is mounted. The

supervisor will translate requests for the mounting of a logical volume into multiple requests for the mounting of physical volumes, if necessary.

Registration information, including an access control list, will be maintained for each logical volume, in a ring 1 data base. This information will include the list of physical volumes, owner identification, and access control and security information.

Volume Connection

Volumes are connected to the Storage System by the supervisor either at system initialization or in response to a user mount request call passed from ring 1. After verifying that the drive is ready and that the volume label, VTOC, and Volume Map are correct and self-consistent, the system makes an entry in the Device Table showing that the volume is on-line.

Before a connection is made, each VTOC entry is validated (its current segment length and number of pages must agree with the file map), and the VTOC file maps are then checked against the Volume Map. If a file map address from a VTOC entry points to a disk record marked free in the Volume Map, the record will be marked as used. If two file map addresses point to the same Volume Map entry, the Volume Map and both VTOC entries will have the record freed, and the record will be zeroed.

Volume Dismounting

Volume dismounting may be the result of an explicit request by the user who mounted a volume or the dismount request may be issued by a privileged process.

The supervisor must not allow the dismounting of a volume if there are still pages in core or on the paging device which have not yet been written to the device. Each volume will have a switch which prevents any more activations. A program similar to shutdown can then set the switch and loop through the AST deactivating segments on a volume which is to be dismounted.

Once a volume has had all its segments deactivated and all pages flushed, it is safe to dismount it. Any known segments which have been dismounted will cause seg_fault_error conditions if they are referenced.

File System Initialization

The program initialize_dims is called to start up the Storage System. Its first step is to read the CONFIG deck and initialize the disk DIM's for each disk type listed on a PRPH card. It then reads the INIT card, determines the correct partition, and locates the PART card for the partition. The PART card lists the logical volumes which are in the partition; the first logical volume listed must contain the root. The logical volumes are described by VOL cards which tell which physical units contain the physical storage for the logical volume. Each volume is connected to the system, starting with the root volume.

The root volume contains a pointer in the volume label to the VTOC entry for the root directory. It may be possible to have a special segment which contains a root branch, in order to eliminate various pieces of complication in directory control.

Disk Record Assignment

When the system attempts to write out a page which has the "null address" flag on, the supervisor will assign a disk record on the volume where the segment resides. For each volume connected to the system, the supervisor keeps a pool of free addresses in wired-down core. As record addresses are needed, they are withdrawn from the pool. If a pool becomes empty, the supervisor replenishes it by reading in a section of the Volume Map and noting the free addresses. The pool is also added to by pages released at truncation and deletion, and zero pages freed at deactivation.

Since page faults cannot claim very many pages a second without exhausting the system's free space, the number of times that the Volume Map must be consulted should be low when the system is in steady state.

Access to the VTOC

The VTOC manager is a new program which will be responsible for all accesses to the VTOC entries. It will have wired core buffers of its own, and will access the VTOC by a special I/O facility which will use 64-word disk reads and writes instead of 1024-word I/O.

When a request to read a VTOC entry is made, the VTOC manager will first search the AST to see if the segment is active, and if so will reconstruct the VTOC entry from the ASTE and return the VTOC entry to the caller. If the segment is not active, the core buffers will be checked to see if the VTOC entry is recently used and still in core. If not, a disk I/O request will be issued for the VTOC entry; it will be read into a free

core buffer if one is available. If no buffer is available, the oldest buffer will first be written to disk (if modified) and then the read performed. Each buffer will have a "modified bit" so that a VTOC entry need not be rewritten unless it has changed.

The contents of the VTOC entry can be completely reconstructed from the AST entry, so that when a segment is active, we can assume that the only copy of the VTOC entry exists in the AST. This property allows us to write VTOC entries out to disk from the AST entry data without having to first read in the VTOC entry in order to update it.

Although the use of 64-word I/O in addition to the standard 1024-word I/O used by the paging mechanism adds some code and some complexity to the supervisor, it provides several important advantages for the management of VTOC information. Obviously, the use of small "pages" for VTOC entries cuts down on the amount of disk channel busy time and memory load, for any given rate of access to the VTOC. The amount of wired-down storage needed to buffer VTOC entries in core is decreased. But the most important effect is to eliminate the unnecessary transportation of VTOC entries and file maps for segments which are not being activated. Our experience to date suggests that data are most often destroyed when they are in core, or when they are transported to and from core. Using 64-word I/O makes it more likely that a system crash will destroy only segments which were actually in use at the time of the crash.

Locking

The locking hierarchy looks like this:

directory lock
parent directory lock
root directory lock
AST lock
VTOC manager lock
page control global lock
traffic control lock

The VTOC manager lock and the page control global lock will be on the same level - that is, there is never an attempt to lock both of these at once.

Carrying Packs Between Sites

Carrying packs between sites will be tricky. The VTOC entries are valid, but what must be done is to construct branches for each VTOC entry, and fill in VTOC pointers and valid unique ID's. This operation must be privileged and done carefully. The VOLUID must also be changed, since different installations may

have assigned the same VOLUID to different packs.

One way to construct the new branches to describe the contents of a volume which has been imported into a site would be to require the user to run a program which looks much like backup_dump, which would write a small tape containing the directory information only for all segments on the volume. Then the user would carry both a pack and a small tape reel. A convention could be established to permit the supervisor to place the contents of this tape on the pack itself.

The ability to carry packs between sites will not be part of the initial implementation.

Quota

The quota mechanism will have the same basic elements as the current scheme: that is, all segments in the same directory will be charged to the same "quota cell", consisting of

maximum records used
current records used
time-record product
time last updated

Since all non-directory segments in the same directory must reside on the same logical volume, one quota cell per directory is sufficient. It will be stored in the VTOC entry and the branch for the directory when the directory is not active, or in the AST entry when the directory is activated. The storage for pages of directories themselves is always on the logical volume which contains the root. In order to prevent any user from monopolizing storage on the root volume, each directory will actually have two quota cells: one for directory pages only, and the other for pages of non-directory segments. As in the current system, there will be a value of quota which means that there is no limit on the storage in this directory, but that some higher-level directory's limit must be checked.

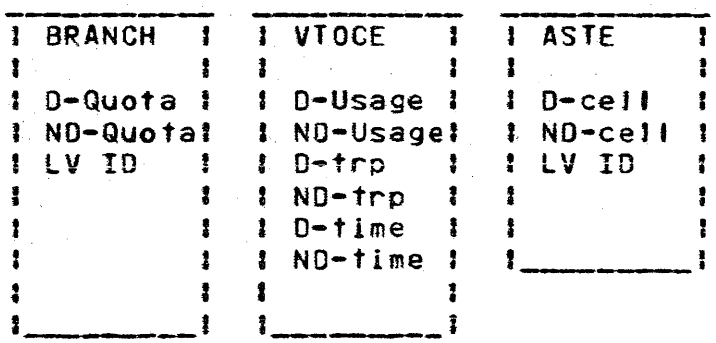

Figure 2: Quota information
for Each Directory

Within this framework, it is possible (but not necessary) to make a major improvement to the current quota mechanism which should provide users with significantly more flexibility in controlling their disk usage. Currently, when the Storage System finds a quota which is nonzero, the storage for the segments inferior to the directory with the quota is "charged" to the quota, and no further checks are made. In the new scheme, the Storage System then continues up the hierarchy, as long as the logical volume identification matches, checking quota at each level. There is no movequota operation, and quotas may be set freely at any level by any user with modify permission on a directory.

An example will make the use of this facility clear. Suppose that a project is to be given a maximum quota of 100 records. The system administrator sets the project directory's quota to 100. Now, suppose there are 10 users on the project. The project administrator may then set a quota of 20 on each user home directory. Any user may use up to 20 records of storage, provided that the project's total usage does not exceed 100 records. Thus, a user could possible encounter several different record quota overflows: either from his home directory, or from the project directory, or from higher directories. In practice, the quotas for the root and for >udd will be set to "infinite". Only the quota cell nearest to the segment will accumulate a time-record product.

Since the chain is broken when a directory with storage on a different logical volume is encountered, the use of dismountable volumes does not affect the normal quota mechanism on system storage.

Repairs to the Hierarchy

When system operation is interrupted abruptly, the Storage System data bases may have been left in an inconsistent state. We have attempted to eliminate states which are inconsistent, or to minimize the amount of time the system spends in these states. Procedures which verify that the hierarchy is correct and repair it if necessary will continue to be needed, though, because hardware and software errors can occur which violate any of our assumptions.

There are four repair operations which must be worked out: emergency shutdown, pack salvage, tree salvage, and tree-VTOC salvage.

EMERGENCY SHUTDOWN

The emergency shutdown mechanism will work about the same as it does now. When the system crashes and ESD is invoked, an attempt will be made first to update the Volume Map on each mounted volume. If this operation succeeds, an attempt will be made to flush out all core pages, and then to deactivate all segments (and update the VTOC's).

PACK SALVAGING

This operation is performed whenever a volume is connected to the system; it should take only a few seconds. It consists of reading through the VTOC for the volume and examining each file map, and checking the Volume Map entry for each page in the file map to make sure that the page is recorded as being used by the VTOC entry and is pointed to by only one file map address. In all salvaging operations, it is not necessary to use 64-word I/O; and the use of virtual I/O will make the code clearer and more obvious.

In "long salvage" mode each disk record which is marked allocated in the file map can be checked to see if it is zero, and if so, the record can be released from the file map and the VTOC entry adjusted. Free records can be checked to make sure that they are zero. If a record which should be zero is found nonzero, the data cannot be restored to its rightful owner: but such a find is evidence that the system has probably lost some data.

TREE SALVAGING

This operation is like the current salvager. Starting with the root, the directory hierarchy is scanned and each entry is checked for validity. Directory hash tables, ACL's, etc. are

rebuilt if necessary. If a branch cannot be made valid, the branch is deleted. If a directory cannot be made valid, the directory's branch is deleted and the directory segment deleted.

TREE-VTOC SALVAGING

This operation is done after volume and tree salvaging. The directory hierarchy is walked and for each branch, the VTOC entry is located and the UID match checked. Then, the VTOC for the system volume is scanned, and VTOC entries not visited during the tree walk are examined. For each such entry, there is no valid branch. An attempt is made to construct such a branch by examining the UID pathname in the VTOC extension: this may point to a valid branch in a directory which has become detached from the root by accident, or it may point to garbage. The salvager follows the parent UIDs back until it finds the break in the hierarchy, and constructs a new branch for the segment or subtree in ">lost_and_found". Since the VTOC extension contains the primary entry name for the branch, we may even be able to rebuild the branch in the correct place.

In "long" mode all mounted volumes are processed. When speed is important, the salvager can check only the system volume. If this operation is fast enough, we will do it every time we boot the system.

Backup

Complete and catchup dumps can be replaced by physical dumps (like the current BOS SAVE) for backup of most of the system. A retriever can be written which will retrieve a segment from one of these tapes given a VTOC index. It may even be possible to run these dumps without shutting down, if users can accept the 5 minutes' wait which would be required for the satisfaction of a segment fault encountered while a pack was being dumped.

An option to allow a user of a private dismountable pack to request that his volume be dumped to tape would be desirable; this might be an offline utility request.

The directory structure of the system can be backed up by a "skeleton dump" similar to the current dump programs, but which dumps only directory data, not segments. Incremental backup dumps can be run if the installation wishes to provide protection against the accidental deletion of segments.

The Storage System will have an option to cause specified volumes to be written in duplicate on more than one physical drive. A moderately large installation can cause a duplicate copy of the root logical volume to be kept, and the system will then be protected from disk catastrophes involving the directory

hierarchy. A later improvement might involve allowing the system to automatically switch to the backup copy if the primary copy went bad; such a proposal can be made later if experience shows that the facility is useful.

Device Reservation

Because there will be very little incentive for a user to dismount a volume, unless the installation sets a very high price for use of a dismountable volume, and because many users may be using a volume's contents when it is mounted, most installations will wish to establish some sort of schedule for permission to use the disk drives which are available for user mounting. An automatic device-reservation system to handle the scheduled forced dismount of volumes on these drives and permission to mount new volumes will be a necessity.

The interaction of this facility with the Access Isolation Mechanism must be considered carefully.

Pack Initialization

A BOS utility must be written to initialize a volume for use with the new storage system. This utility must be able to label volumes and build VTOCs and Volume Maps. It should be able to zero an entire volume as well.

Error Recovery

Several improvements are planned for the disk DIMs so that when a disk drive or pack goes bad, the system will attempt to keep running. One consequence of this desire is that the supervisor will attempt to discover when it has typed out, say, ten disk error messages for the same disk address or address range, and automatically suspend use of this part of the disk until made to start again. (Of course, this cannot be done for the root volume.) Moving packs from one spindle to another is a dangerous activity, especially when disk errors are occurring; but sometimes this will cure disk problems, and it would be nice to have the system well enough organized so that such a swap could be made without crashing or shutting down. A special interrupt or an operator command could be used to tell the system to start retrying its I/O after the operator had attempted to correct the problem.

Operator Commands

The following operator commands must be provided:

reply to disk mount message
list mounted disks

Privileged Commands

The following commands must be provided for system programmer and system administrator use:

list mounted disks
list device table
list device reservations
force device dismount
force online pack salvage
force device reservation

DATA BASES

This section describes the format of various system data bases.

Configuration Cards

The following is an example of the configuration deck for the new Storage System:

```
INTK 0 MULT
PART MULT V1 V2 V3 V4 V5
PART SALV V6
 .
 .
 .
VOL V1 SRV DISK 0 0 404.
VOL V1 SRV DISK 1 0 404.
VOL V2 STO DISK 2 0 404.
VOL V3 STO DISK 3 0 404.
VOL V4 STO D18 0 0 202.
VOL V5 SCR DISK 4 0 202.
VOL V6 SAL DISK 4 202. 202.
 .
 .
 .
PRPH DISK A 23 191. (disk DIM info)
PRPH D18 A 25 181. (disk DIM info)
```

The INTK card tells the system what partition to use. The PART cards define which volumes make up the partition. If more than 13 volumes are in a partition, additional PART cards with the same partition name may be supplied.

The VOL cards name the logical volumes, and specify their device type and location. In the example, "SRV" and "STO" are flags which describe the use to be made of the volume, and "DISK" and "D18" are (logical) device types which will be looked up in PRPH cards. The other parameters on the VOL card are pairs of <first-record, n-records> expressed in cylinders.

Directory Branch

Several data items now stored in the branch for a segment will be moved to the VTOC entry associated with the branch. There is a one-to-one correspondence between branches and VTOC entries, and the directory lock protects the VTOC entries as well as the directory branches.

The directory information relating to the logical organization of the data represented in the Storage System will

be stored in the branch; the information about the physical storage will reside in the VTOC for the volume containing the storage.

In particular, the following items will be removed from the branch:

file map
device ID
date/time modified
date/time used
current length
records used
AST entry pointer

Most of these items will be moved to the VTOC entry, except for the ASTEP, which is eliminated. Instead of inspecting the directory to see if a segment is active the supervisor will search the AST for the unique ID of the segment. (A hash table for the AST may be implemented to make this fast.)

In order to enable the supervisor to find the VTOC entry given a branch, the directory entry will have a new item added: a VTOC pointer stored in the branch which locates the VTOC entry.

For the branch for a directory segment, some items will be added. The maximum records used for both directory and non-directory records, and the logical volume identifier for non-directory records will be added to the directory branch in order to make the activation of a directory which has a quota simple. (Usage, time-page-product, etc. will go in the corresponding VTOC entry.)

One advantage of the division of information between the VTOC and the branch is that directory branches need not be modified when a segment is activated and need not even be referenced when a segment is deactivated. In order to prevent any directory page from being modified at segment-activation time, the directory lock will also be moved to the AST entry for the directory (since a new rule will be that a directory cannot be deactivated while it is locked). This change should reduce the paging traffic on the system, and will reduce the chances of a directory page being damaged due to memory parity or disk channel errors, since the page need not be written back to disk after use.

One problem with this division of data is that the length information for a segment is kept in the VTOC, and so the operation of listing a directory requires the fetching of each VTOC entry corresponding to an entry in the directory. As compared to the current storage system, the new system will have to do noticeably more I/O to return the same information. Furthermore, the real-time delays associated with functions which

list a directory will increase significantly. It may be necessary to store some of the length information in duplicate in both the branch and the VTOC entry in order to allow the simplest cases of directory listing to operate without referencing the VTOC. Another alternative would be to change the list command so that the default case does not provide any information whih it is costly to obtain, and to provide new supervisor interfaces to replace hcs_$star, which return only information kept in the directory.

Directory Header

Very little change will be made to the directory header. As mentioned above, the directory lock will be moved from the directory header to the AST entry in order to avoid unnecessary modification of directory pages. The per-directory static multilevel meters will be removed because nobody uses them.

The quota information now in the directory header will be moved to the branch for the directory in the directory's parent, or to the VTOC entry corresponding to the branch.

Each directory will gain one new item: the name and unique ID of the logical volume where segments inferior to the directory will be stored. This datum is also kept in the branch for the directory, because it is used by the quota mechanism. This attribute may be changed only for empty directories. Modify permission on the directory is necessary in order to change it, and it may not be chaged to an arbitrary value -- the user changing the logical volume ID must be listed on the extended ACL of the VDS for the volume, if the volume is a private volume.

VTOC

The VTOC will be organized as a parallel set of fixed-size arrays in a special region of the volume not available for regular storage. One array will contain the VTOC information used during normal operation, anc the other arrays, called the VTOC extension, will be used to hold the special salvaging information.

VTOC Entry

The VTOC entry for a segment will contain the following items:

unique ID
date/time segment modified
date/time segment used
file map

```
current length
records used
directory switch
quota information (2 sets):
   records used
   time-record product
   time trp last updated
*primary name of segment
*unique ID pathname of parent
```

The items marked with an asterisk will be stored in the VTOC extension for the convenience of the salvager. All other items can be reconstructed from the AST entry contents, so that deactivation does not require any reference to the directory branch.

File Map

The file map in the VTOC entry will use only 18 bits per record address instead of 36, because the device ID can be eliminated. The file maps in every VTOC entry will be maximum size, rather than the current situation where variable-size file maps are permitted. Only 256K segments will actually use more than 64 words of VTOC entry, but all 192 words will be read in by the VTOC manager because it won't know the length of the VTOC entry.

Volume Map

The Volume Map for a volume has one entry for each record on the volume. The current system's analogue to the volume map is a wired-down data base, the bit map portion of the FSDCT, which has one bit for each record. As the amount of physical storage in a configuration increases, this data base becomes too large to wire down, and so it will be allowed to reside on the volume it describes.

Page Table Word

Since a full disk address will no longer fit into 22 bits (18-bit address plus 4-bit device ID) as it does in the current system, the format of a PTW for a page which is not in core must change slightly.

The new format of the PTW has the 18-bit volume address only; the index into the device table which completes the disk address is the same for all pages of the segment and so goes in the AST entry.

A flag bit is turned ON in the PTW if the page is all

zeroes. Such a page can be freed when the segment is deactivated.

A flag bit is turned ON in the PTW if the page has never been assigned. If a reference is made to such a page, a page will be assigned at page out time.

ASI Entry

Several items must be added to the AST entry to support the new Storage System. These include:

device table index
VTOC index
directory lock
date/time modified
directory switch
logical volume ID
non-dir quota cell
dir quota cell

Several other items must be changed. The "dnzp" switch, if still necessary, changes in meaning, since zero pages are nulled at deactivation instead of page fault time. The "did" moves to the device table. The "ppmi" and the "movdid" items are obsolete.

The units for "csl" and "np" should probably be 16-word blocks instead of 1024-word pages, in case we ever experiment with changing page size. The "misw" flag should be renamed the "in_pdir" flag for clarity.

Device Table

The device table is a new wired data base which replaces some of the functions of the FSDCT in the current system. It has one entry for each disk drive available on the system.

In each entry, the following information is kept:

VOLUID
LVID
DIM type (device id)
Volume state
Disk DIM data: channel, drive, etc.
sensitivity level and category
Volume map locaton
read-only switch
system-volume switch
number of free records
volume coming down switch

Volume Label

The label for a volume in the new Storage System is checked when the volume is connected. It is located at a fixed address on the volume known to the DIM, and contains the addresses of the VTOC and the Volume Map. It also contains data used to verify that the volume is correctly mounted, such as

VOLUID
sensitivity level and category
date/time initialized
volume name
manufacturer's serial number
date/time last mounted
date/time last salvaged
error history, bad track list

Disk Layout for DSU-191's

The DSU-191 disks will be arranged to take advantage of the physical characteristics of the disk drive. The disk DIM for the 191's will be the only module which knows what strategies have been used in arranging the data on the disk (except for BOS).

Since the first four cylinders of a 191 pack are guaranteed error-free, the label for the volume will be placed somewhere in these four cylinders. The VTOC extension will also reside in this area. It is tempting to put the VTOC and volume map there too, in order to use the most reliable cylinders on the disk, but probably the VTOC and volume map should reside at the middle cylinders of the disk, in order to minimize average seek time.

Volume Description Segment

Each logical volume which can be mounted in response to a user request will have a corresponding Volume Description Segment in a per-system directory. The exact form of the volume-registration data base is currently being redesigned, but whatever volume-registration data base the system finally ends up with, the per-volume data will include

Logical Volume ID
List of Physical Volumes
List of users who may set quotas for this volume
Name, address, account number, etc. for billing