To: Distribution

From: D. R. Vinograd

Date: 01/28/76

Subject: New Storage System Backup

Introduction

This MTB proposes a new backup subsystem for the storage system. The proposed design is based on the observations and conclusions of MTB-203 "Attributes of a Good Backup System." What is present here is a design overview. Detailed design descriptions and future functional and performance extensions are deferred to future MTB's.

The present functions of backup, incremental, consolidated and complete dumping as well as retrieving and reloading are retained, although some of their definitions are changed. The need for offline physical dumps, BOS SAVE and RESTOR functions, is eliminated.

The proposed design has two major differences from the present backup. They are:

1. Backup is assumed to be a system function and as such is not invokable by the general user.

2. A hierarchy scan is no longer used to determine what data objects should be dumped.

Definition of Terms

Before proceeding with the body of the proposal, it is useful to define some terms that will be used.

data object -
A unit of binary information. In the context of this

document a segment or a directory.

incremental list -
A per physical volume list of data objects that have been modified since the start of the last incremental dump cycle of that physical volume.

consolidated list -
A per physical volume list of data objects that have been incrementally dumped since the start of the last consolidated dump cycle of that physical volume.

no incremental dump switch (nid) -
A user-settable segment switch which, if on, disables incremental dumping of the segment.

no complete dump switch (ncd) -
A user-settable segment switch which, if on, disables complete dumping of the segment.

Design Limitations

Before describing the new design in detail it is useful to describe what it will not do. The new design does not provide all the varied functions of the present backup subsystem nor does it replace the present subsystem. Instead its main purpose, as described in MTB-203, is to provide a fast, inexpensive, and easy to operate method of data recovery which maximizes system availability.

The new design has no provision for pathname directed input to the dumper. Further, the new recovery mechanisms preserve uids. Thus, the new design is not suitable for use as a transmission method between two sites as there will be no way to specify what is to be dumped and no way to guarantee the uniqueness of a foriegn uid. This means that system distribution, the carry facility, and any tape archival operations must continue to use the present backup system.

The present consolidated dumper uses a date criterion. The new design does not and will only find those data objects which have been incrementally dumped since the start of the last consolidated dump. Thus it will not be easy to create overlapped consolidated tapes. Two options, described below, will allow cumulative consolidated dumping and date dumping per physical volume.

The new design relies on the existence of the branch before a retrieval is permitted. This is done for reasons of access control and to ensure that the user retrieves what he wants. Thus retrievals of deleted segments may involve a two step operation. The retrieval of a long deleted data object may be very complex

if the subtree has also been deleted. Further, because of this restriction, cross directory retrievals will not be allowed.

Operational Environment

Backup will operate in ring 1. The dumper mechanisms will operate at a system high security level. The recovery mechanisms will operate with system privileges. The special case access control now afforded the SysDaemon project can be removed from the system. These changes can be made because backup's method of accessing data objects to dump and recover is at the physical volume level, through privileged gates, rather then through the storage system hierarchy.

Control

Control of which physical volumes to dump will be defined in a input control segment listing logical volumes. Logical to physical volume translation will use the Logical Volume Registration File as described in MTB-229. The logical volumes specified in the control file need not be online but they must be registered. It will be possible for a site to have different dumper processes for logical volumes which are in different security catagories.

Incremental Dumping

The incremental dumper will appear to operate in much the same manner as it does today. The incremental dumper will cycle in a round robin manner dumping from each specified physical volume, in turn, those data objects that have changed since the last pass. The data objects which have changed will be defined by the incremental list. If the data object dumped does not already appear on the consolidated list it will be added. If the data object was modified during the dumping interval it will be left on the incremental list, otherwise it will be removed. If a segment has the nid switch on it will be removed from the incremental list regardless. Incremental dumping by different processes of the same or different logical volumes can be done in parallel.

Consolidated Dumping

The consolidated dump of any logical volume will contain those data objects incrementally dumped since the start of the previous consolidated dump for the logical volume in question. Data objects dumped will normally be removed from the consolidated list unless they have been incrementally dumped since the start of the consolidated dump. An optional mode, where data objects are not removed from the consolidated list,

will be provided for a cumulative consolidated dump between complete dumps. The nid switch will not be respected nor will usage information, described below, be updated. This is because consolidation is for system convenience, not for user protection. Consolidated dumping by different processes of the same or different logical volumes can be done in parallel.

Complete Dumping

Complete volume dumping will be produced for each logical volume specified. The VTOC for each physical volume will be treated as a list of data object and successive data objects will be dumped. The ncd switch will be respected. The volume header will not be dumped since it will be recreated during volume initialization, described below, should the volume have to be reloaded. Each complete logical volume dump will begin on a new tape. An option will be provided to select what data objects to dump by a date criterion. This provides a similar function to todays consolidated dumps but at a physical volume level.

Maps and Logs

The incremental, consolidated, and complete dumper will update a tape log as they operate. The tape log will record which physical volumes are contained on which tape. The tape log will be self cleaning since the existence of a consolidated tape will obviate the need to record a series of incremental tapes and the existence of a complete dump of all physical volumes will obviate the need to record a set of consolidated tapes. Care will be taken to provide at least one prior dump cycle to fall back to in case of tape error. An error log will also be maintained. The error log will be managed in the same way that it is today.

No other maps or logs will be produced. The main reason for dispensing with the current map strategy is the cost to create, print, and store the maps. The present maps are used only for retrieval tape identification and are an inherent security risk. The use of a tape table of contents and the storing of the last dump tape ids in the VTOC entry, as described below, is a much better solution.

To provide continuity with todays backup operation two new utility programs will be provided. The first operation will produce a printable description of the hierarchy similar to the map produced by todays complete dump. The second operation will verify the readability of a dump tape and produce a list of the data objects the tape contains.

Retrieval

Data objects will be retrieved by uid match rather then by name. This will require the existence of the directory branch prior to the retrieval of the data object. Branch retrieval, if necessary, will be done using the dump tape ids stored in the VTOC entry of the parent directory in conjunction with the salvager verification and merge operations described below. This approach has been chosen to allow access checking prior to retrieval and to guarantee that the data object recovered is the one described by the branch. Once the data object uid is known the incremental tapes can be scanned in reverse tape chronological order. The tape table of contents, described below, will be used to minimize tape processing. If the VTOC entry has not been lost, the tape scan can be dispensed with since the last dump tape ids will be available. The tape ids will also allow a user to follow the chain of dump images as far back as he chooses.

Reload

A reload of a physical volume will normally be required only if the VTOC is unreadable. In all other cases the salvager will log the data objects which must be recovered and the scheme described below under system startup can be used. The reload of a physical volume proceeds as follows. Reload initializes a physical volume using the same methods used for storage system volume initialization. The tape log is consulted and each incremental tape which contains data objects from the damaged physical volume is scanned and the data objects reloaded. This is done in reverse tape chronological order until the first consolidated tape is encountered. From then on only consolidated tapes are scanned in reverse chronological order until the first complete dump of the physical volume is encountered. Once the complete dump is scanned the physical volume can be made available to the system. Reverse tape chronological order is used so that only one copy of a data object is reloaded. Once a VTOC entry and its associated data object, if any, is reloaded, all further occurances of the VTOC entry and the data object on later tapes are skipped. Reloading is a sequential operation on a peripheral physical volume. The reloader reconstructs a logical image of the lost volume. Multiple reloads of different physical volumes can occur in parallel.

Hardcore Modifications

The supervisor will be modified to manage the incremental and consolidated lists. These lists consist of unidirectional threads in VTOC entries. At "update VTOC entry" time if the nid switch is off, the data object has been modified, and it is not already on the incremental list, it is threaded on. The

incremental list therefore has no timewise order. It is simply a list of all modified objects not dumped since the last incremental pass. When a data object is created the new VTOC entry is put on the incremental thread. When a data object is deleted the freed VTOC entry is put on the incremental list. For performance reasons, the AST entry contains the nid switch, the incremental list thread, and a backup entry hold switch. In part 3 of the VTOC entry, space is allocated for the storage of date-time-dumped, the consolidated list thread, the ncd switch, and the last dump tape ids.

An access method is provided to allow a dumper to efficiently get copies of data objects that are to be dumped given the physical volume id and VTOC index. This method will be more fully discussed in a forthcoming memo. For the purposes of this MTB, the interfaces defined in Appendix B will suffice.

Salvager Interface

The salvager will log any data inconsistences, data unavailability, and data losses it encounters. The data loss log will be used by backup for data recovery operations as a part of system startup as described below.

During a retrieval operation, before any retrieved directory is added to the existent hierarchy, it will be verified by a salvager directory verification procedure. After verification the salvager will be used to merge the contents of the two directories. The merge operation will restore specified missing entries to the online directory without destroying new entries.

List Consistency

As noted above the driving function of the incremental and consolidated dumpers is per physical volume threaded lists of VTOC entries. Care must be exercised to preserve these lists and a method of reconstruction, if a list is damaged, must be available. Otherwise a data object might not be dumped and thus be lost for all time.

In order to minimize the complexity of list management the following rule will be imposed. Only one type of a dump of a single physical volume will be allowed. Different types of dumps, such as incremental and complete, of the same volume will be allowed.

When the system is shutdown, either normally or as the result of a crash, information about the dump state of each physical volume will be updated in the volume header.

If the above cannot be done the the volume must be salvaged. The physical volume salvager will rebuild the incremental and consolidated lists and reset the pointers in the volume header. Because the list order is not important this can be done by comparring the date-time-modified(dtm) in the VTOC entry with the date-time-volume-last-dumped in the volume header. If the dtm is greater then the item is rethreaded on the appropriate list.

System Startup

At system startup time, before users can login, the data loss log prepared by the salvager will be scanned and sorted by backup. The sort will order the data objects by hierarchy level, or some other measure of importance. Under operator option, all the important missing data objects will be retrieved before users are allowed to login. Once this is done users will be allowed on and the operation will continue till all lost data objects have been recovered. This scheme is proposed to ensure that users are not allowed to login until the system can support them.

Resource Usage

Many of the segments that are dumped today and recovered in the event of a failure can be easily recreated by the user rather then being dumped and recovered. Examples of these are object, listings, and runout segments. To decrease the system resources used by backup it is proposed that the system interfaces that create recreateable data objects set the nid and ncd switches on as the default. Commands will be provided to override these default settings for those users who desire.

Metering

Backup will record cumulative statistics about its operation for later analysis. This will include items such as the ammount of system resources used, the number of tape errors, the number and record size of segments and directories that were dumped, and other items that are deemed interesting.

Tape Format

Backup will use the standard Multics tape interface. The first logical record of the tape will be a log containing general information known at the beginning of a dump cycle. The second logical record will be the tape log described above. Because the tape log defines which tapes should be mounted and in what order, the chance of operator error during any data recovery sequence will be minimized. The third logical record will be a table of contents of the previous tape. This will

increase the speed of the retrieval operation because only the first tape and the one that contains the data object sought must completely scanned. All other tapes will be read only for the table of contents. The tape table of contents can optionally be left online so that no tape reading need be done. Since it consists of one uid (one word) per data object on the tape the size should be small. The table of contents will also provide a cross check that reverse tape chronological order is being maintained. Subsequent logical records on the tape will consist of a pattern identifiable header preceding a data object.

Performance

Specific numbers cannot be provided, but some general observations about how the present and proposed backup systems operate and thus some conclusions about their relative performance can be made. The present incremental and consolidated dumpers perform tree walks of the hierarchy to locate data objects to dump. These walks involve many directory operations; locking, searching, and comparison. In the new storage system the relocation of the dtm to the VTOC entry means that date comparisons take twice as long. Further, as the size of the directory hierarchy increases the number of directories that must be searched per data object dumped also increases. The proposed design does not have these problems. Instead each data object that must be dumped is a member of a threaded list of VTOC entries. Thus, only those data objects that are to be dumped are accessed by the dumper.

The present dumper makes a copy of a segment that is to be dumped. This action causes page faults as the copy is created and also creates a flushing effect on the paging device as the segment to be copied is paged out. The new dumper will not create a copy but will snare the segment if it is already active. If not active, the dumper will activate the segment in such a way that it is not put on the paging device. A user will be able to use a segment activated by the dumper in the same way as today and in doing so will put the segment on the paging device.

The present SAVE and RESTOR functions require the Multics system to be unavailable to users while they are being done. The new complete volume dump and volume reload can be done with the system operational and usually available for users.

Future Memos

This MTB does not discuss all the areas that concern a new backup design. In future design memos the following areas will be more fully explored.

Supervisor Primatives.

System Operations Interface

User Interface

Administration and Billing

Appendix A

- Recovery Scenarios -

While scenarios are not proof of a good design, they do illustrate in an easily understandable manner how the design will operate. To this end, the following two scenarios are presented.

A physical volume has failed in such a manner that all the information it contained is lost. One example of this is a timing track head crash. Another example is operator error in using the volume as a scratch pack or overwriting it during a test.

A reload of the lost volume must be done. To do this operations logins a reload process and starts the reload by specifying the physical volume name. The reload process will request the assignment of a peripheral disk drive and the mounting of an unused pack and will initialize the pack using the same mechanisms used by the storage system. The tape log will be used to determine which incremental, consolidated and complete dump tapes contain data objects from the physical volume being reloaded. The reload process will then request the assignment of a tape drive and read the tapes in reverse tape chronological order beginning with the incremental dump tapes, then the consolidated dump tapes and finally the complete dump tapes. Once the latest complete dump tapes have been read, a logical image of the lost physical volume will exist and operations can make the logical volume, of which the recreated physical volume was a part, available to users.

If the damaged physical volume was not part of the root logical volume (RLV), the reload can be carried out while the system is operational. If the physical volume is part of the RLV but not the root physical volume (RPV) then the system can be booted in a limited state and the physical volume reloaded. If the RPV is destroyed then the system must be cold booted using a scratch physical volume and the RPV reloaded. In this case the tape log on the latest incremental tape would be used.

Because of some failure, one or more data objects are damaged and must be recovered. The failure can be because of user, system, or hardware error. The data objects may be segments or directories.

The user of the data object, observing that it is lost, issues a command of the form:

```
enter_retrieval_request  pathname_of_lost_object
```

The user's request is queued for later processing. This is all the user must do. The rest, from the user's viewpoint, is

automatic.

At some later time operations logs in a retrieval process which accepts queued requests. If the branch and VTOC entry for the requested object are available, then the operation is as follows. The user's access to modify the data object is checked. If it is acceptable then the tape, defined by the incremental tape id stored in the VTOC entry, is mounted and searched for a data object whose uid matches the one in the branch.

If the VTOC entry has been destroyed then the physical volume id stored in the branch is used to determine, from the tape log, which tapes to scan and the tape search proceeds till a uid match is found. An optional time from which to start the search can be specified by the user to decrease the number of tapes that must be scanned. If the tapes to be scanned are chronologically successive, or the tape tables of contents are on line, then the tape searching time may be smaller.

If the branch has also been deleted then it must first be recovered. The user's access to modify the parent directory of the data object is checked. If acceptable, the tape defined by the incremental tape id in the parent directory's VTOC entry is mounted and a copy of the parent directory is recovered. The recovered data is input to the salvager for verification and the specified branch is extracted and appended to parent directory in such a way as to not invalidate any existing branches. Once the branch has been recovered, the recovery sequence, described above, of a branch without a VTOC entry, can be done.

The recovery of a deleted sub tree is simply a repeated application of the steps described above. It should be noted that because of the organization of the new storage system, the deletion of a directory branch by the salvager does not cause the sub tree to be lost. Thus, the only object that need be recovered is the damaged directory.

Appendix B

Dumper Supervisor Interfaces

```
call hphcs_$initialize_dumper (copy_dir_ptr, abs_seg_no, code);
```

where

copy_dir_ptr	is a pointer to a dumper temp seg into which directories are to be copied. (Input)
abs_seg_no	is the segment number of a temp seg whose SDW will be set by the supervisor so that the dumper can access segments that are to be dumped. (Output)
code	is a system error code. (Output)

This entry informs the supervisor that the calling process is a dumper and sets up the special environment that a dumper process requires. This includes setting the special SDW which allows the dumper process to access any segment and the entry hold switch on the segment pointed to by the variable copy_dir_ptr. This entry is called only once in each dumper process.

```
call hphcs_$get_dump_thread (pvid, thread_type, vtocx, code);
```

where

pvid	is the physical volume identifier. (Input)
thread_type	identifies if the incrementals or consolidated thread is desired. (Input)
vtocx	is the index of first data object in the thread and negative if there are none. (Output)
code	is a system error code. (Output)

This entry is used by a dumper process to pick up the head of the desired thread. If the thread is already in use or the physical volume specified is not mounted then an error is returned.

```
call hphcs_$get_next_data_object(input_ptr, output_ptr, code);
```

where

input_ptr	is a pointer to the structure described below. (Input)
output_ptr	is a pointer to the structure described below. (Input)
code	is a system error code. (Output)

```
dcl 1 input based (input_ptr) aligned,
    2 version fixed bin,
    2 pvid bit(36),
    2 type fixed bin,
    2 flags,
     (3 mod_after bit(1),
      3 rethread bit(1),
      3 no_update_vtoce bit(1),
      3 no_object bit(1),
      3 pad bit(32))unal,
    2 vtocx fixed bin,
    2 start_time fixed bin(71),
    2 mod_after_time fixed bin(71),
    2 volid bit(36);
```

where

version	is the version number of the structure.
pvid	is the physical volume identifier.
type	specifies by value the type of dumping to be done (1 = incremental, 2 = consolidated, and 3 = complete).
mod_after	if on enables dumping of the object using the date specified in the argument mod_after_time if the mod_after_time is less then the date time modified.
rethread	if on causes the object to be rethreaded onto the list it was a member of.
no_update_vtoce	if on causes the VTOC entry not to be update because of dumping.

no_object	if on the data object specified will not be returned.
vtocx	is for incremental or consolidated dumping the index of the VTOC entry which is to be dumped. For complete dumping vtocx is the index of the VTOC entry that was previously dumped.
start_time	is the starting time of the dump of this physical volume.
mod_after_time	is the time used to determine if the specified object should be dumped, if the mod_after flag is enabled.
volid	is the uid volume identifier of where the dump is being written.

```
dcl 1 output based (output_ptr) aligned,
    2 header like output_header,
    2 version fixed bin,
    2 vtocx fixed bin,
    2 vtoce like vtoce aligned;
```

where

version	is the version number of the structure.
vtocx	is the index of the VTOC entry.
vtoce	is the VTOC entry.

This entry is used by the dumper to access the data objects that are to be dumped. The various flags, defined above, control the actions of this call. The information returned consists of the VTOC entry and, if requested and available, the data object defined by the VTOC entry. If the data object is a directory then the dumper will get its copy from the previously defined copy_dir temporary segment otherwise it will use the previously specified special segment. Each call to this entry releases the previously defined data object, if any.

For incremental and consolidated dumping, the VTOC index of the item to be dumped is input. In the case of complete dumping, the next valid VTOC entry after the input index is used. If the input VTOC index is negative the data object previously specified, if any, will be released and the volume label will updated with the value specified in the start_time argument.

A non zero error code is returned if any input arguments conflict or are invalid, or if the specified physical volume is not marked as in use by this process. A non zero error code is also returned should the physical volume be dismounted while being dumped or if the data on it is unreadable.

```
call hphcs_$revert_dumper(code)
```

where

code is a system error code. (Output)

This entry reverts the changes made by the the hphcs_$initialize_dumper entry.

```
call hphcs_$release_dump_item(type)
```

where

type	is the type of dump as described above.

This entry resets the dumper specific variables associated with the the item being dumped.