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

Date: 11/21/73

Subject: The Storage Problem

Now that Multics runs well enough that the MIT installation has over a thousand users willing to store large amounts of data within the Storage System, and now that price decreases and new technology have made large amounts of high-performance disk affordable, it is becoming clear that the implementation of the Multics Storage System has some deficiencies which may impede the growth of the system.

This document is intended to be a statement of the problem. It is not a proposal for implementation.

Symptoms of the Problem

The observed deficiencies in the Multics Storage System exhibit four categories of symptoms:

1. The Storage System loses information.

   In about 10 percent of the MIT system crashes, some information in the Storage System is lost.

   One of the ways Multics loses information is to generate a "re-used addres," that is, some core, bulk store, or disk address is assigned to more than one page. When a re-used address is generated, Multics may detect the problem. If so, the page in question can be awarded to one or the other of the segments which appears to contain it, or the page can be cleared and removed from both. In order to avoid security problems, the second course should be taken; but from the user's point of view, this will result in a "hole" appearing suddenly in one of his segments. If the segment which is damaged is a directory, many segments will lose their branches, and thus must be deleted by the salvager. In some cases, the supervisor will not detect the damage, because one of the two claiming segments gets deleted: this causes a page of someone else's data to appear in the middle of a user's segment or directory, with no warning.

   Either hardware or software failure can cause other types of loss of information from the Storage System. The system may crash for any number of reasons: those crashes caused by

failure of some Storage System device will ususally cause information loss; crashes due to other causes will in general interrupt the system in the middle of an operation, leaving one or more data bases inconsistent, and so may result in information loss because the data cannot be interpreted without information as to what the system was doing at the time of the crash. An example of the second case would be stopping Multics while a directory is being modified: no data bits have been lost, but the directory is unusable because we no longer know where within directory control the system was executing. Often, when Multics crashes, we have lost only a little information, but that information is necessary for the interpretation of a much larger set of undamaged data so that in effect quite a lot of data is lost. (One rarely loses a car, but losing one's car keys can be almost as bad.)

2. Our backup and recovery procedures cost too much.

In particular, Multics spends far more resources on backup procedures than other operating systems of comparable complexity, and yet has a poorer record of information loss.

Storage System catastrophes requiring a complete RESTOR/reload occur infrequently (say once every 2 months), but when they do, the system requires a very long time to recover -- on the order of 12 hours.

Even if a crash does not require a reload, it takes over 30 minutes to bring the system up again if Emergency Shutdown fails and the Salvager must be run.

The incremental and complete dumps performed while Multics is running comsume a startling amount of machine resources. For October 1973, Dumper and Backup used over $50,000 worth of machine resources -- about one-fourth the system's capacity. (Almost $1,000 more was used by Retriever.)

The system is unavailable for several hours each day, so that a SAVE can be performed. Although improvements in the tape drives and the tape package will enable us to cut this time down to less than an hour, the cost to the system of an hour a day is significant, and the requirement that the system be off the air every day and the necessity to maintain an additional backup mechanism have substantial hidden costs.

Almost all of our backup and recovery mechanisms seem to be doing too much work. And this is not because they are bad or trivial programs: the backup and recovery subsystems are large and complex, and represent a considerable investment in programming. However, these programs seem to suffer from a lack of information as to what data has been damaged or will be damaged, and so end up spending most of their time doing

work which was not needed or will not be needed.

3. The Storage System cannot handle large amounts of storage.

As the cost of disk decreases, users will be able to afford more storage for their computing dollar. If we attempted to use the current reload, salvage, backup, and SAVE procedures on a system which had the equivalent of 100 DSS-190 packs, we would do nothing but backup while the system was up, and be off the air most of the time doing SAVEs, salvages, or complete reloads.

4. Several desirable features do not fit into the current system.

Although the DSS190 disk hardware allows the mounting and removal of disk packs, Multics does not support removable disk devices at all. It would be desirable, furthermore, to allow removable packs to implement removable virtual-memory storage, rather than simply treating the disk pack as a big random-access tape.

Similarly, the disk hardware currently supports read-only packs, and there are several large read-only data bases on the MIT system; but the Storage System does not provide support for declaring a group of segments read-only.

The current backup mechanism does not accept any user indication of when a segment or group of segments is self-consistent. As a result, many segments which are dumped are dumped in a state which makes it worthless to retrieve them.

As the cost of disk storage drops, the notion of dumping to disk instead of to tape becomes more attractive. If system recovery performance is limited now by the time necessary to locate needed data on dump tapes, then marked improvements can be made by dumping to an appropriately indexed disk pack. The current Storage System has no provision for such a facility.

5. The operator interface to the Storage System is deficient.

When a system crash destroys information, the operator has very limited means for discovering what information has been damaged. In some cases, the complete hierarchy has been reloaded because, although only a few segments were lost, those segments were necessary for normal Multics operation and no programmer was on hand to re-create or retrieve the segments.

The way device addresses are assigned and stored in Multics

leads to other operational difficulties. Since pages of a segment can be assigned to any disk unit in the configuration, the loss of any one device means the loss of such a large fraction of the hierarchy that a reload is always necessary. Furthermore, since disk addresses are stored in the file maps in the directories, the Storage System depends strongly on the configuration; shringking and expanding the system's device complement requires the writing of special programs or a cold boot and complete reload.

Goals

From this general set of concerns, we may draw an equally general set of goals for the reliability of the Multics Storage System.

1. The Storage System should not lose information.

   Multics should crash much less often than it does now. Re-used addresses and other damage to the information in the Storage System should be prevented by better means than are currently used. Graceful degradation mechanisms like the online salvager should be improved to the point where the system can recover from many more situations which now lead to crashes. The MTBF of over 48 hours which we saw for one month on the 645 is a cause for some hope in this area; but Multics has had its "bad weeks" -- months, even -- and even if the software were perfectly coded and proved correct, the system would still crash due to hardware problems, electrical power failures, programmer, FE, and operator error, and storage device failure. If possible, the system should recover from core parity, disk parity, power failure, and the like.

   In certain cases, of course, we would actually like to have more crashes. Having the system crash is preferable to having it continue without noticing an error, since this may cause information to vanish with no indication that it has been lost. Re-used addresses occasionally cause the system to exhibit this behavior: Multics should detect those re-used addresses which are generated and take some more appropriate action. Since we wish to have the system be more reliable than its weakest component, self-checking procedures, including hardware diagnostics, should be included in the supervisor to notice and act on system component failures.

   Since Multics will crash occasionally, we wish to have less damage done when it does crash. The system should rarely lose data stored on device X unless something has gone wrong with the track on device X on which the data resides, or there has been a head crash on device X. In particular, we wish to lose as little as possible when core or bulk store errors occur: only recently-modified segments should be affected, and the damage to them should be limited to the loss of recent modifications. The current situation, in which a segment which has been unmodified and unused for months is suddenly destroyed, due to a system crash not involving a disk failure, is intolerable.

   Less information should be lost as a result of salvaging. Here, too, we are less interested in additional clever directory-rebuilding strategies and the like -- bandaids -- than in changes which would either cause the salvager to be needed less, or would give it less opportunity to delete

segments. If Multics were programmed to prevent re-used addresses, or pages of zero appearing in directories, or if directory pages were never written to disk while in an inconsistent state, then the salvager would have less work to do, and could do it better.

2. Backup and recovery procedures should cost less.

We wish to minimize system down time, and to devote fewer resources to backup functions during the time the system is running. It is worth noting that making the supervisor extremely efficient by avoiding operations which might leave the disks consistent in the event of a crash may be a false economy.

The MTTR for almost all (say 98%) of system crashes should be less than 5 minutes. The system should not have to be shut down for backup purposes more than once a week. If only one device is affected by a crash, then only that device should have to be reloaded. Rapid and positive means for determining when a reload is needed should be built into the supervisor. And a "complete" reload, of all the disk drives on the system, should be necessary only if all drives had a simultaneous head crash, or if the system is moving to a new set of devices or new directory format.

Changes in the economics of storage during the years since incremental dumping to tape was introduced may have progressed to the point where it is no longer sensible to provide backup for recently-modified files by dumping them to tape. Even if tape still is the chosen medium, enough work should be done on the incremental backup machinery so that its cost can be decreased by a factor of five or ten. Parts of the revised backup scheme described in MCR-1076 could be used to speed up the dumping process.

When a reload of part or all of the hierarchy is needed, it should go at a speed limited only by the capacity of the input-output channels of the system.

3. The Storage system should support larger configurations.

Extremely large storage configurations should be usable on Multics, without imposing a penalty in performance, reliability, or availability. Continued improvements in the cost/performance of disk devices make it likely that Multics systems with very large amounts of on-line storage will be desired, and that the device characteristics of this on-line storage will continue to change. Multics should certainly be able to handle a configuration with the equivalent of 100 DSS190 drives.

Since new disk devices will undoubtedly be announced which have different sizes, address types, and operating characteristics, the Multics Storage System should be constructed so that device-dependent information, such as disk addresses, is not scattered through the system. This information must be available for use by the Storage System DIMs, of course, but need not be kept in permanent storage anywhere except on the device itself.

4. New features should be added to the Storage System.

The system should be able to support removable disk packs which, when connected to the system, make segments available for use by the Storage System in exactly the same way as segments which are never removed. Segments stored on a removable pack should be shareable between users, protected by access control, and catalogued within the directory hierarchy in exactly the same way that all other Multics segments are. The contents of a removable pack should not have to be the contents of a tree-structured sub-hierarchy; such an organization would lead to convoluted and inefficient use of the naming tree structure to represent storage-allocation decisions.

It should be possible for a system administrator to arrange the allocation of segments to disk packs so that one or more packs could be normally write-protected while the system was in operation. If the system libraries were given read-only status, for example, no software bug or security breach could lead to damage to the contents of this pack: and the chances of the system being runnable after a crash would be much better.

A rethought dumping strategy might provide not only a large reduction in system overhead, but also the ability for a user to indicate that a segment or group of segments was inconsistent, and therefore should not be dumped, or newly consistent, and therefore important to dump.

5. The operator interface should be improved.

Shrinking and expanding the Storage System's device complement should be a straightforward operation which can be performed without programmer assistance.

The relationship between the logical address of a segment (its position in the hierarchy) and its physical residence on the system's disk units should be controlled or controllable so that the loss of one unit destroys a definable group of information. In particular, it would be desirable to be able

to run a crippled system, perhaps with some segments missing, if a Storage System device went down.

After a system catastrophe, the system operator should have access to tools which will tell him as much as possible about what went wrong, what segments have been lost, and whether the system is runnable.