The Real Problems of Backup

C. Linett
U.S. Bureau of the Census
F.O.B. #3, Room 1377
Suitland, MD 20746

S. Ranade
Infotech SA Inc.
P.O. Box 4343,
Silver Spring, MD 20904
infotech@digex.com

1.0 INTRODUCTION

Many organizations today operate a distributed networked computing environment and one of the important problems in data management is to assure an orderly and safe backup of data. The problem of data backup has been adequately addressed in the mainframe world through good and reliable tools. But in the distributed computing world, this problem is more complex and although there are a number of commercially available network backup tools, none of these offer mainframe class safety, security and disaster recovery capabilities. Some Unix vendors, at least, are aware that a gap exists between the functions provided by their backup tools and the functions that are available in the mainframe world and that are really necessary in the distributed world.

The goal of this paper is firstly to describe the functions and features of a good file backup system for very large environment with millions of files. Secondly, its purpose is to examine some current products to see how closely these products provide these necessary (or highly desirable) functions and features. The paper suggests areas of work which could be undertaken by Unix vendors to upgrade their products to offer mainframe class backup functionality.

2.0 BACKGROUND / TERMINOLOGY

It is useful to briefly review the important terms used in describing storage and data management. This is necessary because depending on the environment to which they are applied (e.g. MVS, Unisys, VMS, Unix etc), some terms have conflicting meanings, some have ambiguous meanings and some are meaningless.

a. Backup

"Backup" in the simplest case is a means of making a copy of a data object which is resident on disk. Tape is often the most convenient and appropriate media for such a copy, although any other media could be used. The original idea behind backup was to retain a safe copy of the data in case a disk device failed. A related idea is that since disk

space was relatively expensive, a tape copy of the data allows the data to be removed or truncated from disk as it could be reconstructed in its original condition from the tape copy. It is important to see that these two ideas are very closely related and that in fact, the second is simply a variation of the first. In the Unix world, this fact has been overlooked with the result that the first idea is implemented by one set of software (called "backup") and the second idea is implemented by another set of software (called "hierarchical storage management."). Thus two copies of the same data can exist for the same purposes ! There are several products which actually do this.

b. Full Backup

The term "full backup" applies to a complete copy of all data belonging to a given system. The idea is that this full backup forms a baseline and that following a full backup only files that have changed since this full backup need be backed up. This concept may be valid in instances where the amount disk storage is small, but today, when machines may well have hundreds of gigabytes or even Terabytes of disk storage, this is not practical. When the number of files is very large, restoring files from a full backup is not efficient. Using full backups, it is also not feasible to move (or re-create) data belonging to one system to another system. *Thus the concept of full backup as understood in Unix and VAX/VMS systems must be abandoned as a satisfactory or viable backup concept for large machines.*

c. Incremental Backup

Incremental backup is taken to mean the backup of files that have changed since some baseline backup. However, it should really mean a backup of files that either have no backup or whose backup is not current. Many computer centers operate a weekly "full backup" and a daily "incremental backup" which is quite inefficient for restoring files or restoring the state of a system at a given time. What is needed (and what will be explained below) is a continual backup of files that have no backup or files that have changed since their last backup, with a record of all backup activity kept in a catalog which can be searched quickly. This forms a "virtual full backup" of the entire system at all times (see below). *The concept of incremental backup as viewed in Unix and VAX/VMS systems must also be abandoned as a satisfactory or viable backup concept for large machines.*

d. Catalog

A catalog is a data structure which contains information about files and their backups. It is essential for the catalog to be a data structure that can be searched quickly, since large backup systems may be managing millions of files. Specifically, the catalog must be designed for quick searches and must contain sufficient metadata for complex selection criteria. It is necessary to note that both these features are not available in Unix systems and have to be provided in some way by the storage management application. In fact, the usefulness, performance and limitations of Unix storage management applications can be

determined largely by the strategy which is adopted for implementing the catalog and its search.

e. Virtual Full Backup

A virtual full backup is a combination of the catalog and historical catalog (see 4(d) below) and uses their knowledge of where backup instances and deleted files (see below) are located. These two databases contain sufficient information to reconstruct the system to any point and at any level.

f. Hierarchical Storage Management

Hierarchical Storage Management (HSM) is a means to manage different storage devices (e.g., disk and tape) such that a user sees the mix as a single large disk. Disk files that are infrequently used are selected for migration. When a current verified copy exists on tape, the disk files are truncated. The truncated version (**inode**, in the case of Unix systems) is left on disk. Disk space is thus made available for other processes which need it. When truncated files are required again, access to the **inode** causes a trap into the OS, resulting in the activation of processes to copy the file back from tape to disk. Note that HSM is really an operating system function, i.e., it requires OS support for file faults, although the movement of data from disk to tape and tape to disk (variously termed "rollout", "migration" etc. and "rollback", "staging" etc.) may be performed by privileged processes running in user space. Note that the truncation step described above is not needed for migration software such as UniTree which uses its own file system.

3. TYPES OF BACKUP

Given the various definitions above, it is useful to see how these concepts are applied in different computing environments. Note that in all cases except (e), the backup solutions available are not "mainframe class," i.e., they cannot efficiently support large numbers of files nor can they assure the level of data integrity and safety which are necessary in most large installations.

a. BSD Unix

UNIX itself provides three different back-up utility programs. "Dump and restore" allows a complete backup of the entire system; and "tar" and "cpio" each allow specific files to be backed up to tape. None of these UNIX utilities can verify the data on a tape, nor can they read past an error on a bad tape. They all have very limited features for partial restorations. As a result of these shortcomings, various commercial backup software has been developed.

In BSD Unix Systems, *tar* is a utility which sometimes is used for backup, but it suffers from several drawbacks. Among these are a 100-character limit on pathnames, no facilities for incremental backup, and no facilities for writing to multiple tape volumes.

Nevertheless, some System V UNIX systems use *tar* for interchange with BSD Unix machines.

b. System V Unix

System V Unix backup tools are more limited - for example, a snapshot restore is not possible. Volcopy is intended to be similar to the BSD's dump, but is not quite as flexible. As a result, the faster BSD filesystem and dump have been ported to the System V environment. In fact, the BSD filesystem format is standard with System V.1. volcopy and, therefore, the System V.3 filesystem may be gradually phased out.

With the above backup utilities, Unix system administrators generally use a two step procedure for backup. At some point, a full backup is performed to save the current state of the system. The systems administrator or operator manually records the backup details in a log. The system may be taken off-line for such a full backup, causing inconvenience. This procedure takes a long time which compounds the inconvenience.

Therefore, more frequently, incremental backups are performed. Some sites do this once a day. With incremental backup, restore operations can take a long time and the number of incremental backup tapes required can become excessive. A periodic full backup, perhaps once weekly or once monthly, is performed to avoid this.

Restoring files is not a simple process. The administrator must manually determine which backup contains the file to be recovered by mounting and searching the backups. This is so labor-intensive that most sites recover individual lost files in exceptional circumstances only; more often, they only recover destroyed disks or filesystems.

c. VAX/VMS

In the VAX/VMS environment backup tools have been traditionally provided by DEC, although there is other commercial software such as Backup.unet which performs similar functions. DEC's backup software handles backups on a per disk, per machine basis. It uses the full backup/incremental backup method described above.

d. Unix Network Backup

In many cases, Unix network backup amounts to script files which use the Unix backup utilities described above to backup networked machines. The time, labor and maintenance cost of this is often unacceptably high. Moreover, recovery of lost files requires explicit action by the system operator. This is because there is no online index of backup files. One of the standard Unix tools (discussed above, e.g. dump) is used to copy backup files to a large disk on a file server. Sometimes dump is used to write files directly to a remote tape drive.

There are now a number of commercial products which automate network backup. Special software is installed on client nodes and on a server node. The control of the

backup process lies in the server. A system operator usually sets certain backup parameters (e.g. schedule, file classes etc) after which the backup for all clients on the network runs automatically.

e. Census Unisys

The backup system developed at the US Bureau of the Census provides backup of the main compute server (Unisys) and also backup of client machines (PC and VAX). The server backup software implements many of the features described below. The client backup software implements almost all of the features described below. This system is what is commonly called "mainframe class" meaning that it is designed to supports millions of files, it operates efficiently, includes the HSM function as part of its backup method and provides very high data integrity and safety.

This system has been developed over many years and incorporates many lessons learnt from actual experience. In this paper, this system is used as the basis for a model of a good file backup system. It is to be hoped that Unix vendors will eventually offer similar capabilities.

4. A GOOD BACKUP SYSTEM

This Section summarizes some characteristics necessary for a good file backup system. These apply also to a hierarchical storage management system. Indeed, they should be integrated. The core of each one is to create and verify copies of each file from which the file may be reconstructed or reloaded, if necessary.

a. Basic Concepts

The following concepts developed from many years' experience, form the basis of a good backup system for very large environments.

• All backups should be of files/directories. All backups should be incremental: a full image backup is not a useful concept for a very large system. Note that the *first* "incremental" backup of a system will find that all files need backing up, but it should be the only such, and is still an incremental backup. An incremental backup scheme is one that picks files for backup that have one of two characteristics : (a) they have no backup, or (b) their backup is not current in that it does not reflect what is currently on the disk.

• An HSM system should perform its rollout function using the current validated backup. *A separate copy of the file or separate pools of tape for backup and HSM are undesirable*

• Each backup, even an outdated one or one for a file subsequently deleted, is assumed to have some potential use for recovery. The removal of a backup instance from the backup system should therefore be done only when necessary. "Necessary" means either that the backup is defective (unreadable) or that the tape on which it resides is to be released or

reused. In the latter case, the determination of the backups to be purged should be governed by user-configurable purge rules.

b. Operation

It is essential for the backup system to be fully automatic since instances such as the following may arise :

- Suppose that a system with 10 million files loses 250,000 files through some disaster. It is not feasible to require any manual determination of which files to restore.

- In a large backup system there will be thousands of tape reels. It is not acceptable to have to manually keep track of which reels (say) contain what and/or which may be compacted or written upon at any given moment.

- The large number of backup /restore/delete etc. operations should not result in an impossibly large tape pool. In other words, the backup tape pool should be automatically kept in a state of equilibrium. This implies a fixed-size pool of tapes, platters, or whatever. These would be automatically compacted according to configurable criteria to maintain the pool size. Without compacting, equilibrium can generally not be achieved with any reasonable pool size (relative to the size of the file system). The system should maintain an awareness of what media (which reels, etc.) may be used for writing, and choose them automatically when necessary.

c. Database (Current Catalog)

The backup system should have an associated database of files and their backups. This should have an entry (made in real time) for each file creation, deletion, first modification after backup, and backup.

The database should also be designed with care. Individual file queries should not take excessive time to complete. Of course the database also should be designed for integrity. Snapshots of it (or of some subset sufficient to reconstruct it) should taken fairly frequently so that the database can be rebuilt from one of them, even if the original hardware is no longer there.

There should be a facility to traverse (or select from) the backup database and perform file maintenance actions on each file in an arbitrarily selected subset of files and/or backups. It should be possible to specify any of a generous assortment of standard actions, as well as to flexibly create macros for custom actions. "Arbitrarily selected" means that you should be able to specify a test to be applied to each entry. The basic boolean, integer, and string functions available to express this test should be extensible to specify searches of any reasonable complexity.

The standard actions to perform on each selected file should include such things as changing owners, deleting, various listings, making a backup, reverting to an earlier backup, marking as referenced (or modified), truncating, etc.

d. Historical Database

The backup system should also have associated with it, a separate, historical database. It is important to keep a historical audit trail database of file deletions giving, at least, filename, file creation date, deletion time, who deleted it, reconstruction, damage detection etc. Note that the file creation date is necessary to uniquely identify the file since there may be multiple deleted instances of this filename. The historical database can be used for purposes such as : ensuring that overtly deleted files are not reconstructed ; identifying the user/process which deleted files (by file) ; analyzing the birth/death patterns of data (to help in improving the management of disks and file systems). This database also gives the capability to check the consistency of the current catalog.

e. Data Integrity

Data integrity is paramount. There should be provision for the self-contained verification of backups. Self-contained implies that the original files are not necessary for the verification: it could even by done on a different, compatible machine. There should be a configuration parameter to specify that multiple copies be made of the backups for vaulting and/or to provide additional assurance. Extra backup copies should be integrated into the system gracefully: if, when restoring a file with more than one backup copy, one backup cannot read, then the system should automatically attempt the restoration from the next copy, etc.

The backup format should be designed with care. There must be enough "file header" information for each backup to insure that even if the file is deleted from the system entirely, it can be reliably reconstructed from that backup. There must be enough "check" data to allow self-contained verification of the backup.

5. BACKUP/HSM PRODUCTS

From the preceding discussion it is clear that the viability of a good backup system for a large machine hinges on the structure of the databases discussed above. It is essential for the main database or catalog to be a data structure that can be searched quickly, since large systems may have millions of files. Specifically, the catalog must be designed for quick searches and must contain sufficient metadata for complex selection criteria.

It is necessary to note that both these features *are* available in mainframe operating systems (e.g. MVS) and *are not* available in Unix. In Unix systems these features have to be provided in some way by the storage management application. In fact, the usefulness, limitations and performance of Unix storage management applications can be

determined largely by the strategy which is adopted for implementing the catalog and for its search.

What are the prospects of such a file backup system being made available with a Unix system ? To answer this question, it is necessary to look at the types of file and storage management software that is currently available with Unix systems. Note that in the following discussion, only large Unix systems are considered since these are the systems that are closest in capacity and functionality to mainframe systems.

In general, there are three types of data about files. The first, which is dependent upon application, and is user defined, is data about the contents of files. This we will call *User Metadata*. For the present, this type of metadata is not relevant o our discussion and will be ignored. The second is file system data or Posix data which we will call *File System Metadata*. The third is data about storage locations, tapes, libraries etc. for files and this we will call *Storage Management Metadata*.

In essence, the problem of backup boils down to the strategies which are used to store, access and manipulate these metadata. The choice of strategy decides what can and cannot be provided and how efficiently the resulting system will work. To see how this is so, note that the discussion above really focused on the following main items :

a. the types of metadata that should be stored for files

b. the security and safety of this metadata

c. the performance implications of the method used to store metadata

d. the functional implications of the method used to storage metadata

The software products which come closest to providing the features discussed in Section 4 are FileServ from EMASS, UniTree from UniTree Inc. There is another class of HSM products for small to mid-range Unix machines (e.g. AMASS, EpochServ, Polycenter HSM etc) but these have been implemented on smaller machines and are not suited to the kind of large environment discussed in this paper.

A. EMASS (FileServ)

The EMASS FileServ HSM software as implemented on the Convex C-Series machines uses an Ingres database to keep some of the file system metadata and also for some of the storage management metadata. The use of a database allows varied queries to search for files and also leads to features such as the "Dataclass" in which files may be grouped together.

One of the problems of this approach is that a change to the database entry for a file, results in a call out to the host operating system, since, to maintain consistency, the corresponding OS file system data must also be updated - i.e. both the OS file system

metadata and the Ingres metadata must be consistent. Another problem is that a commercial database such as Ingres is not able to perform efficiently when the number of files is very large.

B. UniTree Inc (UniTree)

The UniTree HSM software is used at many large data centers. The most common platform is the Convex C-Series, although there are production versions running on Amdahl and SGI Challenge machines. UniTree was developed especially to overcome the storage management limitations of Unix systems. UniTree uses its own file system and its own catalog. It does not use a commercial database and has its own algorithms for searching through its catalog. It is possible that with an alternative name server, catalog design and search capability, UniTree could be modified to provide most of the features described in Section 4.

6. CONCLUSIONS

This paper has discussed what a file backup is, what it is for and its relationship to hierarchical storage management. Based on experience in developing a backup system for a large machine handling millions of files, the characteristics of a good backup system are defined. The inadequacies of Unix systems to provide an adequate backup capability for large machines are pointed out. Two products which provide HSM for Unix systems are discussed and their shortcomings for providing reliable backup for very large systems are shown.

7. REFERENCES

1. FileServ Technical Description, EMASS Inc., Garland, TX, September 1994.

2. UniTree Technical Manual, Openvision Inc., Pleasanton, CA, October 1994.

3. IBM ADSTAR ADSM Reference Manual, IBM SSD, San Jose, CA, October 1994.

4. The Mass Storage Report '95 Infotech SA Inc., Silver Spring, MD, Jan 1995