Protecting File Systems: A Survey of Backup Techniques

Article · February 1998

Source: CiteSeer

CITATIONS
121

READS
2,032

3 authors, including:

Ann Chervenak
University of Southern California
101 PUBLICATIONS **6,980** CITATIONS
SEE PROFILE

Zack Kurmas
Grand Valley State University
37 PUBLICATIONS **267** CITATIONS
SEE PROFILE

Protecting File Systems: A Survey of Backup Techniques

Ann L. Chervenak
Vivekanand Vellanki
Zachary Kurmas
{annc,vivek,kurmasz}@cc.gatech.edu
tel +1-404-894-8591
fax +1-404-894-9486

Abstract

This paper presents a survey of backup techniques for protecting file systems. These include such choices as device-based or file-based backup schemes, full vs. incremental backups, and optional data compression. Next, we discuss techniques for on-line backup (backups performed while users continue to access the file system); these techniques include file system locking and creating instantaneous, copy-on-write "snapshots" of the file system. Last, we discuss protecting data from site disasters and media deterioration. The paper then classifies several research and commercial backup systems according to the parameters already described. The classified systems include the UNIX *dump* and *tar* utilities; hierarchical storage managers such as IBM's ADSTAR Distributed Storage Manager, Legato's NetWorker, and UniTree; and several research and academic systems. We conclude with measurements of full and incremental backups over a one-year period in a large networked UNIX environment at Georgia Tech's College of Computing.

1 Introduction

This paper is a survey of backup techniques for file systems. Backups protect file systems from user errors, disk or other hardware failures, software errors that may corrupt the file system and natural disasters. The most common uses of backups are to restore files accidentally deleted by users and to recover from disk failures.

As the capacities of new magnetic disk drives continue to increase at a rate of over 50% per year for the next decade, networked computing environments will grow to include multiple terabytes of disk storage. Meanwhile, the access times and data rates of disk and tape drives will increase at slower rates, about 20% per year. These trends indicate that it will take increasingly long to read the contents of a disk drive and write them to a backup device. Over time, traditional backup schemes are likely to prove too slow.

This paper surveys and classifies existing backup techniques as a prelude to identifying the needs of future systems. The next section characterizes design issues for backup software. Section 3 describes the features of various academic and commercial backup

schemes. Section 4 presents measurements of full and incremental backups at the Georgia Institute of Technology's College of Computing. Finally, Section 5 discusses desirable properties of future backup systems.

2 Design Issues for Backup Software

In the following sections, we characterize several features of backup systems: full vs. incremental backups; file-based vs. device-based schemes; support for on-line backups; the use of snapshots and copy-on-write mechanisms; concurrent backups; data compression; and support for tape management and disaster recovery.

2.1 Full vs. Incremental

The simplest way to protect a file system against disk failures or file corruption is to copy the entire contents of the file system to a backup device. The resulting archive is called a **full backup**. If a file system is later lost due to a disk failure, it can be reconstructed from the full backup onto a replacement disk. Individual lost files can also be retrieved. Full backups have two disadvantages: reading and writing the entire file system is slow, and storing a copy of the file system consumes significant capacity on the backup medium.

Faster and smaller backups can be achieved using an **incremental backup** scheme, which copies only those files that have been created or modified since a previous backup. Incremental schemes reduce the size of backups, since only a small percentage of files change on a given day. A typical incremental scheme performs occasional full backups supplemented by frequent incremental backups. Restoring a deleted file or an entire file system is slower in an incremental backup system; recovery may require consulting a chain of backup files, beginning with the last full backup and applying changes recorded in one or more incremental backups.

We characterize several variations of incremental backup techniques. We call the traditional scheme, where occasional full backups are supplemented with frequent incremental backups, a “full+incremental” or simply an “incremental” scheme. Recent systems include variations of incremental backup schemes. An “incremental-only” scheme is used in the IBM Adstar ADSM system (Section 3.3); full backups are eliminated, and files are only written to the backup medium when they change. The UniTree hierarchical storage manager (Section 3.5) can be characterized as a “continuous incremental backup” scheme; the system makes copies of all newly-written data within a few minutes, rather than doing a traditional incremental backup once per day.

2.2 File-Based vs. Device-Based

Files consist of logical blocks. These blocks, also called pages, are typically of fixed size of approximately 8 kilobytes. Each logical file block is stored on a contiguous physical disk block. However, different logical blocks of a file may not be stored contiguously on disk. UNIX uses an index node or *inode* structure to map logical block addresses to the corresponding physical addresses on disk. An inode contains pointers to physical disk

blocks; for large files, a single inode may be too small to map all the logical blocks, and is supplemented with indirect blocks that contain additional pointers.

Backup software can operate either on files or on physical disk blocks. **File-based backup** systems understand file structure and copy entire files and directories to backup devices. These systems traverse the pointers stored in each inode and read the physical blocks of each file sequentially. Then backup software writes each file contiguously to the backup medium; this facilitates fast individual file recovery. However, storing files contiguously slows down backups, since extra disk seek operations are required when a file is not stored contiguously on disk. These extra seek operations increase disk overhead and decrease disk throughput. Another disadvantage of file-based incremental backup schemes is that even a small change to a file requires the entire file to be backed up. Tools that use file-based backup include the UNIX *tar* program.

By contrast, **physical** or **device-based backup** systems ignore file structure when copying disk blocks onto the backup medium. This improves backup performance, since backup software performs fewer costly seek operations. However, this approach complicates and slows file restores, since files may not be stored contiguously on the backup medium. The UNIX *dump* program is sometimes called device-based, although it is more accurately considered a hybrid between a pure device-based and a file-based scheme (Section 3.1).

To allow file recovery, device-based backups must include information on how files and directories are organized on disks [23] to correlate blocks on the backup medium with particular files. Thus, device-based programs are likely to be specific to a particular file system implementation and not easily portable. File-based schemes like *tar* are more portable, since the backup file contains contiguous files, and the notion of files is fairly universal. Another disadvantage of device-based backup schemes is that they may introduce data inconsistencies. An operating system kernel may buffer write data before writing the disk; device-based backup schemes that traverse disk blocks in order typically ignore this file cache data and back up older versions of files. By contrast, file-based backup schemes consult the file cache and back up current versions of the files.

2.3 On-line Backup

While many backup programs require that the file system remain quiescent during backup, on-line or active backup systems allow users to continue accessing files during backup. On-line backup systems offer higher availability but introduce consistency problems.

Shumway [23] discusses many of the difficulties with on-line backups when using programs like UNIX *tar* and *dump*. The most serious problems occur when directories are moved during backup, changing the file system hierarchy. If a user moves a directory during backup, then depending on where the backup program is in its traversal of the file system hierarchy, the backup program may either not encounter the directory and therefore fail to copy it to the backup medium or may encounter and copy the directory multiple times in different locations. Most backup programs perform a depth-first traversal of the file system hierarchy. UNIX *tar* operates in a single phase, copying files to the backup medium as it traverses the file system hierarchy. By contrast, *dump* operates in multiple phases, with a scan phase that traverses and records the structure of the file hierarchy, followed by a dump phase that copies files. Single-phase programs like *tar* have the longest

vulnerability to file movement; the multiple-phase *dump* program is vulnerable only during the scan phase as it constructs an image of the file system structure.

Other problems for on-line backup include file transformations, deletions and modifications during backup. An example of a file transformation is compression, in which the contents of a file are compressed and written to a new file name (possibly in a new directory), and the original file is deleted. The new compressed file will not be backed up if the backup program has already either completed its scan phase (e.g., *dump*) or traversed that portion of the file system hierarchy (e.g., *tar*); the original file will not be backed up either, since it has been deleted. In general, if files are deleted before the backup program reaches them, they will not be copied to the backup medium. If a file is modified during backup, the results are unpredictable. The backup may contain a mix of old and new data.

Two strategies for overcoming these problems with on-line backup are locking and detection. (The next section discusses a third scheme, copy-on-write.) To prevent directory movement or file modification, some systems lock directories and files to disallow *move*, *write*, *truncate* and *link* commands. Locking limits the availability of the file system, so it is important to minimize the locking period. Alternatively, a backup system can detect file and directory movements and modifications by comparing the old and new state of the file system; if a moved or modified file was left out of the previous backup, the software ensures it is included in the next one.

IBM's ADSM hierarchical storage system (Section 3.3) offers on-line backup with various levels of consistency. At one extreme, the system may be configured not to back up any file that is being modified. At the other extreme, the system copies files to the backup medium regardless of whether they are being modified. Between these extremes, the system will retry copy operations in an attempt to back up a stable and consistent version of the file.

2.4 Snapshots and Copy-on-Write

Another alternative for on-line backup is to create a **snapshot** or frozen, read-only copy of the current state of the file system. The contents of the snapshot may then be copied to a backup device without danger of the file system hierarchy changing from subsequent accesses. Systems such as Andrew (Section 3.9), Petal (Section 3.6) and Spiralog (Section 3.10) can make either full or incremental backups from the frozen snapshot. The system can maintain any number of snapshots, thus providing read-only access to earlier versions of files and directories.

A **copy-on-write** scheme is often is used along with snapshots. Once a snapshot is created, any subsequent modifications to files or directories are applied to newly-created copies of the original data. Blocks are copied only if they are modified, which conserves disk space. The Plan9 (Section 3.8), Petal, and Spiralog backup systems use copy-on-write.

2.5 Concurrent Backups

When performing backups for a large collection of networked machines, it is often desirable to perform backups of multiple file systems in parallel, either to one or to multiple tape drives. Most of the systems discussed in Section 3 allow concurrent backups. The

Amanda Backup Manager (Section 3.7) uses a temporary staging disk to hold backups of several file systems; eventually, these backups are written to a tape drive at high bandwidth. Legato NetWorker (Section 3.4) interleaves concurrent backups from several machines onto a single tape drive. IBM's ADSM (Section 3.3) sends concurrent backups from multiple machines to separate tape drives. Georgia Tech's College of Computing performs backups to up to 25 tape drives in parallel (Section 4).

2.6 Compression

To reduce tape storage and network bandwidth requirements, many backup systems give administrators the option to compress files before backup. The backup system may compress data at the client or file server using compression hardware or software. IBM's ADSM (Section 3.3) and Legato's NetWorker (Section 3.4) include optional file compression. Alternatively, storage devices may compress data immediately before writing files to the backup medium. Many I/O devices, including tape drives, are equipped with compression hardware.

2.7 File Restores

We have mentioned several issues affecting the speed of restoring individual files and entire file systems. Restores will be slower in an incremental backup system, which must begin with the last full backup and apply changes from subsequent incremental backups. Device-based backup schemes may slow down restores, since blocks of a file may not be stored contiguously on the backup medium. An additional concern when restoring entire file systems is that files deleted since the previous backup will reappear in the restored file system.

Performance of individual file restores improves dramatically in systems that maintain online snapshots of earlier versions of the file system, as described in Section 2.4. UniTree (Section 3.5) supports a “trash can” feature for temporary storage of recently deleted files.

The next two sections discuss tape management and disaster recovery issues, many of which can affect the system's ability to restore files.

2.8 Tape Management

Over time, a backup system generates a large number of backup tapes. These tapes are labeled both internally and externally to avoid losing backup data. Internally, each tape begins with a file or label that identifies the file system backups stored on the tape. This information is repeated on the tape's external physical label. Before a backup system writes a backup tape or reads from a tape on a restore operation, it reads the internal tape label to verify that the correct tape has been loaded. Many systems include a relational database that correlates filenames and dates with backup tapes.

Magnetic tape systems face difficult reliability challenges, including errors that are not correctable by error correction codes (ECC), tape wear, head wear and long-term tape storage. A tape error is detected when previously written data cannot be successfully read. Most of these errors are caused by debris that become embedded in the tape surface [17]. Because the rate of such errors is high, all magnetic tape drives incorporate large amounts

of error-correction code. Despite this, tapes occasionally encounter errors that cannot be rectified by the ECC. Tape wear is another reliability issue. Magnetic tapes that are frequently read or written eventually wear out. Tapes last on average several hundred passes [2], [12], [16]. However, they wear out sooner if a particular segment of the tape is accessed repeatedly. Frequent stops and starts on the tape cause excessive wear. Tape heads also wear out, typically after a few hundred or thousand hours of use. Another set of reliability concerns involves the long-term storage of data on tape. Over time, tapes are subject to corrosion and to mechanical changes including tape shrinkage, creasing of the edges, and peeling of the magnetic layer [12]. To avoid losing data due to wear, mechanical problems, or tape deterioration, backup systems must maintain information on tape and drive usage, and must replace old tapes and schedule drive maintenance.

Nemeth et al. offer additional practical advice for performing backups in UNIX environments [19]. The correctness of backup procedures must be monitored; backup software should attempt to re-read tapes after dumps are complete. To guard against backup tapes that become unreadable by the tape drives that wrote them because the drive heads drift out of alignment over time, system administrators should periodically attempt to restore files from various tapes, including tapes that are months and years old. Administrators should also attempt to read backup tapes on different drives from those on which they were written to guarantee that if one tape drive is destroyed, other drives will be able to read the tapes.

Since particular tapes may be lost, damaged, or contain errors, it is important that the loss of one tape does not render the entire backup useless. One scheme used at Georgia Tech's College of Computing (Section 4) avoids this problem by doing full backups every week and daily incremental backups with respect to the previous week's full backup. This allows files to be recovered in the event of a loss of either the current week's or previous week's full backup. A scheme developed at Ohio State University maintains multiple redundant backup “chains”, with independent sets of full and incremental backups [21]. Such a scheme requires twice as many tapes and drives, but offers a high degree of reliability.

2.9 Disaster Recovery

Nemeth et al. [19] describe techniques that ensure that recovery will be possible after natural disasters. Dump tapes that contain full backups should be stored off-site to prevent data loss in the event of natural disaster or fire. Backup tapes should be secured against access by unauthorized persons, since they contain all of an installation's data. IBM's ADSM system (Section 3.3) automatically generates a disaster recovery plan that contains recovery instructions and scripts; the plan includes a list of necessary backup tapes and tape drives and their physical locations. Legato's NetWorker (Section 3.4) plans to add support for automatically writing off-site copies of data to guard against data loss from local disasters.

3 Backup Systems

Next, we examine several commercial and research backup systems and classify them according to the issues discussed in the last section. Table 1 summarizes the features of these backup programs and systems.

system	incremental scheme	file-based or device-based	on-line backup	concurrent backup
dump	full+incremental	hybrid	No	Yes (with scripts)
tar	full+incremental	file	No	Yes (with scripts)
IBM ADSM	incremental-only	file	Yes	Yes
Legato Networker		file	Yes	Yes (interleaved)
UniTree	continuous incremental		Yes	N/A
Petal and Frangipani	full+incremental	file (uses tar)	Yes (snapshots)	
Amanda	full+incremental	(uses tar or dump)	No	Yes
Plan 9	permanent file storage	N/A	Yes (snapshots)	N/A
Spiralog (LFS)	full+incremental	device	Yes (snapshots)	

Table 1: Classification of backup programs and systems. Blank spaces in the table indicate that information was not available. N/A indicates the category does not apply.

3.1 UNIX dump and rdump

The most common UNIX backup program is *dump* [26], which operates in several passes. *dump* maintains character arrays for directories and inodes. On the first pass of the algorithm, *dump* "marks" each directory by setting a bit in a character array. *dump* also examines each inode, setting an array bit for those inodes that have blocks modified since the last *dump*. On the second pass of the algorithm, *dump* looks at all bits for inodes under a particular directory. If any of the inodes are marked or any subdirectories of the directory are marked, then *dump* leaves the directory marked. Otherwise, *dump* unmarks the directory. This pass continues until all inodes have been examined. On the third pass of the algorithm, *dump* writes the marked directories to the tape or other archive device. Finally, on the fourth pass, *dump* writes the marked inodes along with the modified data blocks.

dump is used to create backups on a local tape drive; the variation *rdump* writes backups to a remote tape drive. *dump* performs incremental backups at different levels, from 0 to 9. A level 0 dump corresponds to a full backup. A level N dump backs up all files that have been modified since the last dump of level less than N.

Different sites devise schedules for *dump* based on the activity of the file systems, the capacity and number of tapes, and the amount of redundancy desired [19]. A simple but expensive schedule is to do level zero dumps (full backups) every day. A more moderate schedule would perform different levels of backup on a daily, weekly, monthly, and annual basis. For example, daily level three backups could be supplemented with weekly level two backups, monthly level one backups, and level zero backups at least once a year. Backup schemes like the Towers of Hanoi sequence [19] alternate backups of many different levels to achieve high redundancy with relatively few tapes; such schemes are complex and may perform poorly on restores, since they access many tapes.

The *dump* program includes options for specifying tape characteristics including capacity, tape density, blocking factor, length, and number of tracks. These parameters are used between the third and fourth passes of the algorithm to estimate tape length and the number of tapes required for the backup. The goal is to avoid over-running the end of a tape.

Full backups (level 0 dumps) must be performed on a quiescent file system. In addition, *dump* offers a verification option that checks the correctness of each volume as it is dumped.

3.2 UNIX tar

The UNIX *tar* [27] program is a file-based archival program. *tar* traverses the file system hierarchy in a single pass. It uses standard UNIX file system read calls to read a file sequentially and then copies it sequentially to the backup medium. Because it stores files contiguously, tar offers efficient restores of individual files. Also, because *tar* writes files and the semantics of files are widely accepted, tar archives are more portable than dump files. Therefore, *tar* is widely used for transferring files and directories as well as for backups.

3.3 IBM ADSM

One commercial storage management system that includes flexible backup functionality is IBM's ADSM, or ADSTAR Distributed Storage Manager [9] [10]. ADSM supports automated backup of multiple clients concurrently to separate backup devices, optionally compresses data before backup, and can write multiple copies of a backup file. The storage manager also maintains multiple versions of files for a specified length of time. ADSM allows selective and incremental backups. Selective backups copy only specified files and directories. The incremental backup scheme, called "progressive incremental" or "incremental-only", duplicates the file the first time it is backed up; thereafter, only changes to the file are recorded, and further full copies are not required.

ADSM supports on-line backup with four different schemes or "modes of serialization." In *static* mode, ADSM will not back up any file that is being modified, hereafter referred to as an *open file*. In *shared static* mode, the system retries backup of an open file a specified number of times or until a static copy of the file is made. *Shared dynamic* mode retries backup of an open file some specified number of times; on the last try, if the file is still open, the system backs up the file regardless of whether it may be modified during backup. Finally, *dynamic* mode backs up all files regardless of whether they are open.

ADSM is a hierarchical storage manager that organizes storage devices as *pools*. Primary pools of magnetic disk drives provide fast access to files. Files that are unlikely to be soon accessed may automatically migrate to lower levels of the storage hierarchy based on file size and age. Lower levels of the hierarchy may include magnetic disk, magnetic tape and optical disk drives. *Copy storage pools* are lower levels of the storage hierarchy that are used specifically for backup. The ADSM backup software copies files in primary pools to one or more copy storage pools. The hierarchical storage manager is tightly integrated with the backup functions; typically, a file is replicated to a copy storage pool before it is migrated to a lower level of the hierarchy.

There are several options for file recovery in ADSM. The system supports *Point-in-Time* recovery, which restores the file system or database to its state at a particular full or incremental backup. For database systems, ADSM can maintain a log of transactions; after a failure, the system is first restored to its state at the last backup, and then the log is used to roll forward the state of the database to the point of the system crash.

ADSM accelerates recovery of files for a single client using "collocation." The system consolidates the data for each client on as few sequential volumes as possible. Clients can initiate their own file recovery.

To facilitate site recovery after a disaster, ADSM automatically generates an updated disaster recovery plan. This plan includes instructions for recovering the server, the list of relevant backups and copy pool volumes needed for recovery, the off-site locations of these volumes, the devices needed to read backups, the amount of space required to restore the file system, system configuration files and shell scripts needed to initiate recovery.

3.4 Legato NetWorker

Legato NetWorker is another commercial storage manager that performs automated network backup [4]. NetWorker allows up to 64 file systems to send data to up to 32 backup storage devices simultaneously. To achieve optimal transfer rates for backup tape drives, NetWorker interleaves backup data sent simultaneously by multiple clients onto a single tape. Clients may optionally compress files before sending them over the network for backup. NetWorker includes application-specific routines that obey the locking conventions of particular applications such as relational databases, and it can backup data from actively-running applications. The software also writes electronic labels on tapes; these labels are checked before new writes are allowed.

NetWorker writes files to the backup medium using a portable, machine-independent file format called NetWorker Open Tape Format. Using this machine-independent format allows clients with different operating systems to write to the same backup tape and allows a client to restore files backed up on another platform.

A server process initiates a NetWorker backup session by setting up a connection between a *save* process on a client and a *media manager* process on a server. The save process reads backup data from a client file system or database and forwards it to the media manager, which writes it to the backup device.

3.5 UniTree

UniTree [18] is a hierarchical mass storage system for UNIX environments originally developed at Lawrence Livermore National Laboratory. UniTree allows automatic migration of files between levels of a storage hierarchy.

UniTree does not perform traditional full or incremental backups, but rather provides *continuous backup* [24]. Upon creation, files are stored in a magnetic disk cache. Within a short period, typically 3 to 30 minutes, UniTree copies newly-created files to one or more lower levels of the storage hierarchy. Thus, files are protected more quickly than in a daily incremental backup. The user can make up to 15 copies of a file on physically distinct media [11]. Future versions of UniTree will provide disaster protection by creating

automatic remote copies of files over a wide area network. Because a copy of every file exists at a lower level of the hierarchy, if UniTree needs space in the disk cache for new files, it can quickly purge existing files. For good performance, UniTree maintains all metadata on magnetic disk.

After data loss, files may be recovered from any valid copy in the hierarchy. In addition, UniTree supports a "trash can" feature, in which directories for each user retain recently discarded files. This feature allows users to quickly restore files they have accidentally deleted without requiring reference to backup tapes [11]. UniTree also offers database recovery, including transaction logs.

3.6 Petal and Frangipani

The Petal system [15] from the DEC Systems Research Center allows a collection of network-connected servers to cooperatively manage a pool of physical disks. The pool of disks appears to the servers as a single large virtual disk. Petal provides a copy-on-write snapshot mechanism. When creating a snapshot, Petal pauses applications briefly (for less than one second). Snapshots may be kept online for quick access to previous versions of data. To create a virtual disk backup, Petal simply copies a snapshot to an archive device using a utility such as *tar*. Frangipani [25] is a distributed file system built on top of Petal virtual disks that uses the Petal snapshot facility to perform file system backups.

3.7 Amanda

The Amanda Network Backup Manager [5] [6] was developed at the University of Maryland at College Park to facilitate network backup of many UNIX workstations in parallel. Amanda uses standard UNIX backup programs like *dump* and *tar*. For the best performance, Amanda writes multiple backups in parallel to a temporary holding disk, which later streams the data to a tape drive at its maximum transfer rate.

Amanda backups are controlled by a central backup server host that initiates backups in off-peak hours. The backup server host dictates the backup level for each file system each night. Amanda provides backup history for individual file systems to facilitate restores. The system also allows optional compression for each file system.

Amanda performs checks both before and after backups are run. Before backups, it verifies that the correct tapes are loaded into tape drives and that there is sufficient temporary storage on the holding disk. After backups, Amanda reports any problems that occurred, including disk errors, backup program problems such as permission errors or software crashes, and client hosts that were down.

3.8 Plan 9

The Plan 9 computing environment [20] used a write-once optical disk jukebox for *permanent file storage*. The system also included a magnetic disk file cache for faster file access. Daily on-line backups were made by creating snapshots of the file system. Plan 9 used a copy-on-write scheme; it froze the state of the file system, and made subsequent modifications to a copy of the frozen data. Since old files were not deleted, users could

restore individual files or entire file systems to their state on an earlier date. File access permissions were maintained on snapshot copies of files.

3.9 Andrew File System

The Andrew File System (AFS) organizes files into *volumes*, which are the focus of system management, including backup and restore [8] [14]. A volume is a collection of files and directories. Typically, a user's home directory resides in a different volume, as do different binary directories. A volume resides entirely on one magnetic disk partition, but a disk partition may hold multiple volumes. Andrew allows *cloning* or copying of volumes.

Backup in AFS begins by creating a clone called a *backup volume* that contains hard links to all the files in the original volume. The backup volume is then dumped to tape using either a full or incremental backup scheme. After writing the backup to tape, Andrew may delete the backup volume or may keep it for on-line read-only access to previous versions of data, the equivalent of a snapshot.

One or more machines in the networked file system are designated as Backup Machines. These machines coordinate tape drive operations and maintain a database with information about tapes and dump schedules for individual volumes and sets of volumes.

3.10 Log-Structured File System Backup

The Spiralog backup system from DEC [7] provides on-line backup of a log-structured file system or LFS [22]. A log-structured file system writes all modifications to disk sequentially in an append-only log [22]. The log is divided into segments, which are composed of disk blocks. Because the log is written sequentially, segments appear in the log in temporal order. Over time, as portions of files are edited and deleted, portions of segments contain obsolete data. A segment cleaner periodically traverses the log and compresses valid information into new segments, freeing up the original segments. Periodically, LFS writes a checkpoint to the log, which records the mapping between all logical file blocks in the file system and their location in the log. LFS also puts a pointer to the end of the log at a fixed location on the disk. Upon crash recovery, LFS quickly restores the file system by reading the last checkpoint and using the log to roll forward from the checkpoint. LFS can create snapshots, also called “time travel”, of the file system by creating checkpoints and turning off the segment cleaner so that space is not reclaimed after files are deleted.

Spiralog exploits several features of LFS in its backup system [7]. It uses LFS's checkpoint capability to create a snapshot of the file system that is then copied to the backup medium. Spiralog uses a physical or device-based backup scheme; it copies LFS segments rather than files, and therefore can read data sequentially from disk and write it to the backup medium at a high transfer rate. The temporal ordering of segments in LFS allows Spiralog to perform simple incremental backup, copying only those segments to the backup medium that were modified since the last backup. To facilitate restores of individual files, Spiralog writes a copy of the directory tree before writing segments to the backup medium. These directories are consulted to determine which segments contain data for the requested file, and only these segments need be read from the backup medium. Restoring an entire volume requires copying the last incremental backup and any preceding backups on which

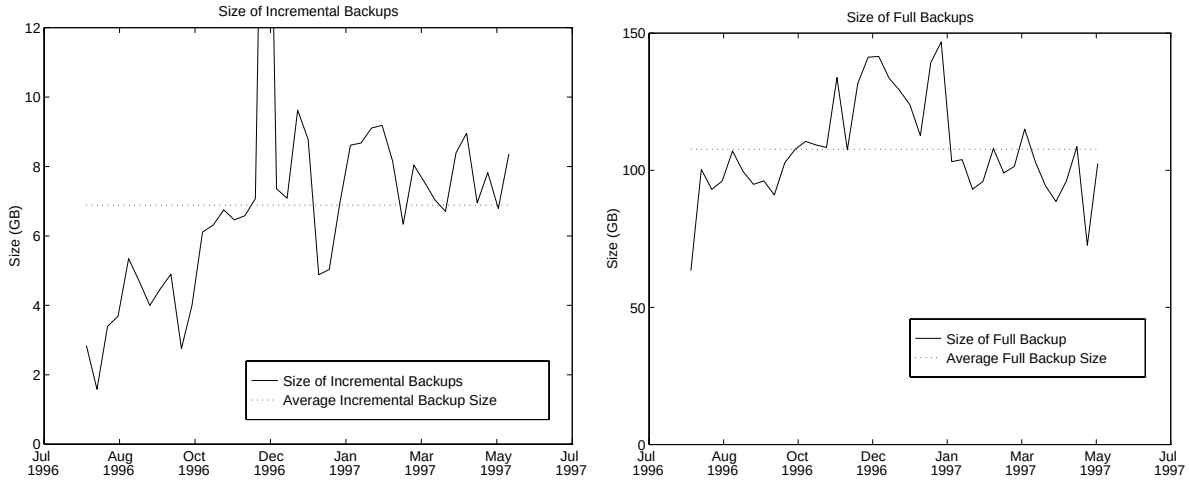

Figure 1: *Size of incremental and full backups in Georgia Tech's College of Computing from August 1996 to June 1997.*

it depends to disk.

3.11 SGI Terabyte Backup

Silicon Graphics implemented a system that includes an Origin 2000 server, 138 disks, 66 UltraSCSI channels, and 38 IBM 3590 tape drives. This system achieves a backup rate of over a terabyte of data per hour [3] using both Legato NetWorker and Spectra Logic Alexandria hierarchical storage manager software. This performance is achieved while running a TPC-C database benchmark at an applied load of 4500 transactions per minute.

4 Georgia Tech Backup Measurements

The College of Computing at Georgia Tech has more than 200 UNIX workstations, 150 Macintoshes and a smaller number of Intel personal computers running Windows/NT. Until recently, these machines were backed up using a locally-defined collection of backup scripts that performed full backups once a week and incremental backups daily. Backups were performed to 25 tape drives simultaneously. Each tape drive could write approximately 2.5 gigabytes of compressed data per tape.

Figure 1 shows the average sizes of incremental and full backups, respectively, over a period of approximately a year. The incremental backups show a steady increase in the average data produced per week; around December 1996, there was a large increase in data, probably due to temporary storage of image and video files from graphics researchers. Incremental backups typically consume less than 10 gigabytes, while full backups consume over 100 gigabytes per week. Again, the full backup sizes increase considerably around December 1996, and drop down again around March 1997, indicating that large files were created and later removed from the system.

Backup times represent the sum of time spent on all 25 tape drives. The aggregate time for incremental backups was approximately 20 hours, while full backups took approxi-

mately 200 hours. Distributed over 25 tape drives, this means that full backups consumed about 8 hours of real time per week per drive, typically spread over several days.

More recently, the college has switched to using Legato Networker software to backup the UNIX machines and Retrospect to backup the Macintosh and Intel machines. In addition, the college has acquired two tape libraries, each containing two drives. Each tape library holds 800 gigabytes of compressed data. These libraries use Exabyte Mammoth tape drives, each capable of writing 20 gigabytes of compressed data per tape. Under the new scheme, full backups of the UNIX machines are performed once per quarter, Level 5 backups are performed once a month, and Level 7 backups are done once a week. Finally, the scheme includes daily incremental backups.

The College supports 700 active users. System administrators typically receive one request per day to restore a file that has been accidentally deleted by a user. Restores of entire file systems due to disk failures are less frequent, occurring approximately once per month.

5 Summary

We have developed a characterization of backup systems and used it to categorize a number of commercial and research backup schemes. First, we classified incremental backup schemes. The *full+incremental* scheme supplements occasional full backups with frequent incremental backups. An *incremental-only* scheme avoids full backups of a file system and only writes incremental changes. A *continuous incremental* system copies new data within a few minutes of its being written.

Next, we differentiated between systems that copy data to the backup medium as contiguous files vs. those that copy physical blocks regardless of file structure. *File-based* backups facilitate restores and are more portable, while *device-based* backups access the disk more efficiently.

We discussed problems associated with providing on-line backup and considered several schemes. The file system can disallow writes during backup, or it can create instantaneous *"snapshots"* or read-only copies of the file system, which may then be copied to the backup medium. Many systems that create snapshots provide a *copy-on-write* scheme that replicates data only when it must be modified, thus saving substantial disk space.

We discussed several schemes for providing *concurrent* backup of multiple file systems over a network. We also discussed the ability of file systems to compress files before they are written to the backup device. Many systems also provide support for managing tapes and tape drives, and some provide automatic protection from site disasters by creating and managing remote copies of data.

As file systems grow to multiple terabytes, it is likely that new backup strategies will be required to protect them. The most promising techniques for handling very large file systems appear to be incremental-only backup schemes, device-based backup to use disk bandwidth efficiently and to avoid writing entire files based on small file changes, snapshots and copy-on-write for on-line backup, compression of data before backup, and automated creation of off-site backup files.

6 Acknowledgements

The authors are grateful to Dan Forsyth for his generous assistance in gathering statistics on Georgia Tech's backup environment. Vikas Gupta did initial work in identifying papers used in this study. This work was funded in part by NSF grant CCR-9702609.

References

[1] K. A. Anderson and B. H. Kirouac. A Simple and Free System for Automated Network Backups. In *The Third Annual System Adminstration, Networking and Security Conference (SANS III)*, pages 63–68. Open Systems Conference Board, April 1994.

[2] B. Bhushan. *Tribology and Mechanics of Magnetic Storage Devices*. Springer-Verlag, New York, 1990.

[3] S. Cariapa, R. Clark, and B. Cox. Origin2000 One-Terabyte Per Hour Backup White Paper. http://www.sgi.com/Technology/teraback/teraback.html.

[4] L. Corp. Legato Networker Documentation. http://www.legato.com.

[5] J. da Silva, O. Gudmundsson, and D. Mosse. Performance of a Parallel Network Backup Manager. In *USENIX Conference Proceedings*, pages 17 – 26, 1992.

[6] J. da Silva and O. Guomundsson. The Amanda Network Backup Manager. In *Proceedings of USENIX Systems Administration (LISA VII) Conference*, pages 171–182, November 1993.

[7] R. Green, A. Baird, and C. Davies. Designing a Fast, On-line Backup System for a Log-structured File System. *Digital Technical Journal*, October 1996.

[8] S. Hecht. Andrew Backup System. In *USENIX Proceedings of the Workshop on Large Installation Systems Administration*, pages pp. 35–38, November 1988.

[9] I.B.M. ADSTAR Distributed Storage Manager (ADSM) – Distributed Data Recovery White Paper. http://www.storage.ibm.com/storage/software/adsm/adwhddr.htm.

[10] I.B.M. ADSTAR Distributed Storage Manager (ADSM) – Frequently Asked Questions. http://www.storage.ibm.com/storage/software/adsm/adfaq.htm.

[11] F. Kim. UniTree: A Closer Look at Solving the Data Storage Problem. UniTree Software Inc., http://www.unitree.com/newpage/wpaper.htm.

[12] T. Kitahara. On the Long Term Storage of Metal Tapes. Fuji Photo Film Co. Magnetic Recording Lab; FIAT/IFTA Technical Commission in Hilversum (The Netherlands), June 14th 1988.

[13] R. Kolstad. A Next Step in Backup and Restore Technology. In *USENIX Proceedings of the 5th Conference on Large Installation Systems Administration*, pages 73–78, September 1991.

[14] S. Lammert. The AFS 3.0 Backup System. In *USENIX Proceedings of the 4th Conference on Large Installation Systems Administration*, pages 143–147, October 1990.

[15] E. K. Lee and C. A. Thekkath. Petal: Distributed Virtual Disks. In *Seventh International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII)*, October 1996.

[16] J. C. Mallinson. Magnetic Tape Recording: Archival Considerations. In *Digest of Papers*. Tenth IEEE Symposium on Mass Storage Systems, May 1990.

[17] C. D. Mee and E. D. Daniel, editors. *Magnetic Recording, Volume II: Computer Data Storage*. McGraw-Hill, New York, 1988.

[18] NCSA. UniTree Mass Storage System Frequently Asked Questions. http://consult.ncsa.uiuc.edu/docs/unitree.

[19] E. Nemeth, G. Snyder, S. Seebass, and T. R. Hein. *UNIX System Administration Handbook*. Prentice Hall, Inc., Upper Saddle River, New Jersey, 1995.

[20] R. Pike, D. Presotto, K. Thompson, and H. Trickey. "Plan 9 from Bell Labs".

[21] S. M. Romig. Backup at Ohio State, Take 2. In *USENIX Proceedings of the 4th Conference on Large Installation Systems Administration*, pages 137 – 142, October 1990.

[22] M. Rosenblum and J. Ousterhout. Log-Structured File System. In *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pages 1–15, June 1991.

[23] S. Shumway. Issues in On-line Backup. In *USENIX Proceedings of the 5th Conference on Large Installation Systems Administration*, pages 81–88, September 1991.

[24] U. Software. Technical Overview. http://www.unitree.com/newpage/techover.htm.

[25] C. A. Thekkath, T. Mann, and E. K. Lee. Frangipani: A Scalable Distributed File System. In *Sixteenth ACM Symposium on Operating System Principles (SOSP-16)*, October 5-8 1997.

[26] DUMP(8). Unix System V man page.

[27] TAR(1). Unix System V man page.

[28] E. D. Zwicky. Torture-testing Backup and Archive Programs: Things You Ought to Know But Probably Would Rather Not. In *USENIX Proceedings of the 5th Conference on Large Installation Systems Administration*, pages 181–190, September 1991.