Evaluating Backup Algorithms [*]

Zachary Kurmas
College of Computing, Georgia Tech
801 Atlantic Ave. NW
Atlanta, GA 30332-0218
tel 1-404-894-9390
fax 1-404-385-1253
kurmasz@cc.gatech.edu

Ann L. Chervenak
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292-6601
tel +1-310-822-1511
fax +1-310-823-6714
annc@isi.edu

Abstract

We present a trace-driven simulator that evaluates the performance of backup algorithms. We use this simulator to compare the performance of the commonly-used *level* scheme (our name for the algorithm used by the UNIX dump utility) with that of a new algorithm called the *Z scheme*. We show that the Z scheme has better backup performance than the level scheme and slightly worse restore performance. We also show that the Z scheme consumes less media for backups than the level scheme.

1 Introduction

Given the unprecedented increases in magnetic disk drive capacities, university, enterprise, and government data repositories are growing rapidly. Protecting these data repositories using traditional backup techniques is a growing challenge [3].

In this paper, we evaluate four backup strategies. Our eventual research goal is to demonstrate the limits of current backup algorithms and propose new algorithms both for sequential magnetic tape media and for random access alternatives, including write-once technologies such as CD-ROM and DVD. Initially, we focus on backups to sequential magnetic tape media.

Every backup algorithm makes a tradeoff between backup performance and the performance of *restore* operations, in which files are retrieved from the backup medium. At one extreme, an algorithm optimized for fast backup might simply write periodic *incremental backups*, or copies of files that have changed since the last backup. Each day, this *pure incremental* approach copies to the backup medium only the files that have been modified that day. The disadvantage of this simple, fast backup scheme is that the performance of restore operations may suffer. If files in a single directory are modified on different days, the pure incremental algorithm may disperse those files widely on the sequential backup media. Later, to restore the entire directory, reading large amounts of sequential media may be necessary.

[*]This work was supported in part by NSF CAREER Award CCR-9702609.

At the other extreme, an algorithm optimized for fast restore would take pains to lay out data carefully on the backup medium, copying all files in a given directory to the same backup medium and grouping together directories that are logically connected. An algorithm of this type might even perform daily *full backups*, in which all files and directories are written to the backup medium. The goal of these algorithms is to reduce the time required to restore a given directory or file system by minimizing the quantity of sequential media that must be accessed during the restore operation. The disadvantage of such algorithms is that the backup phase requires substantially more media and takes much longer, given the bandwidth limitations to sequential tape media (e.g. the speed at which the tape drive can write).

Between these two extremes are many other backup algorithms that attempt to strike a balance between backup performance, restore performance, and media requirements. We evaluate two of these algorithms here: the traditional backup scheme used by many programs, including UNIX dump, which we call the *level* scheme, and a new algorithm we developed which we call the *Z scheme* [6, 1].

To evaluate these backup algorithms, we wrote a trace-driven backup simulator. Inputs to this simulator include traces made by Tim Gibson [4] as well as traces of backup activity collected in the College of Computing at Georgia Tech. Our simulation results evaluate the performance of the full backup, pure incremental, level, and Z schemes. As expected, our results show that the full backup algorithm has poor backup performance but performs restores efficiently. In contrast, the pure incremental scheme performs well for backups but poorly for restore operations. The level scheme performs well for all metrics, with backup performance close to that of the pure incremental scheme and restore performance close to that of the full backup scheme. The Z scheme performs backup operations better than the level scheme, and restore operations almost as well as the level scheme. We will argue that the Z scheme's better backup performance makes up for its slightly worse restore performance, and, therefore, is a better algorithm.

2 The Backup Simulator and File System Traces

The trace-driven backup simulator is written in C++. Its modular, object-oriented design simplifies implementing new backup algorithms, accommodating new trace formats, and gathering different performance metrics.

Input traces for the backup simulator contain periodic records of the state of a file system hierarchy — specifically, daily images of the file system metadata at the time backups are performed. Using these records or snapshots of the state of the file system, the simulator traverses the file system hierarchy, examines file metadata, and decides which files should be backed up according to the specified algorithm. The algorithm uses a tape object to keep track of the layout of files on the tape. During a restore operation, the simulator uses Bruce Hillyer's model of a DLT 4000 tape drive to estimate the seek time to access a particular file [5]. Metrics for evaluating the algorithms include the time to perform backup and restore operations and the amount of tape media required for backup.

In this paper, we present simulation results that use Gibson's traces as input [4]. Gibson's traces include a set of files for each day observed. Each trace file contains the name, size, owner, group, inode, access time, modification time, and change time of each file on the traced file system on a given day.

3 Backup Algorithms

We now describe the backup algorithms that we evaluate in this paper.

- **Pure Incremental Scheme:** Once every day (or, more generally, once every period or epoch specified by an algorithm), this scheme copies only those files that have changed since the last incremental backup to the backup medium. We use this scheme as a basis for evaluating optimal backup times.

- **Daily Full Backup Scheme:** Every day, this scheme copies all files in the file system to the backup medium. In general, this scheme puts files in the same directory close together on the backup medium. We use this scheme as a basis for evaluating optimal restore times.

- **Level Scheme:** The level scheme is our name for the traditional backup algorithm that alternates between periodic full backups and incremental backups of different "levels", where a backup at a given level includes all files that have changed since the last backup at that level.

- **Z Scheme:** The Z scheme is the name we give to our algorithm. It is a simplification of an algorithm developed at UC Berkeley by Costello, Umans, and Wu [2]. This scheme performs concurrent backups to several *backup streams*. The Z scheme uses a parameter b, called the *base*. A particular file is written to backup stream i if it was last modified *exactly* b^i days ago. For example, every day the algorithm writes to stream 0 those files that were last modified yesterday. If the base b is 2, the algorithm writes to stream 3 those files last modified exactly $2^3 = 8$ days ago.

To restore every file currently in a file system, a backup algorithm must read the most recent version of every file from tape. In a pure incremental backup scheme, we found that restoring a file system required reading a large number of tapes. In particular, we found that for an incremental backup scheme that uses one tape per day, there is a high probability that each backup tape contains at least one file that was last modified on that date (usually, there are several files). To restore a file system on day x, the restore operation must load and read at least one file from approximately x tapes. Since the time to load and seek on a tape often exceeds the time to read files, this results in poor restore performance for the pure incremental scheme.

The level scheme trades backup performance for improved restore performance by redundantly backing up files in such a way that the number of tapes that must be read to restore a file system is bounded by a constant (provided that the size of the file system is also bounded).

As we will see, the level scheme has very good performance; however, its performance is hindered by two inefficiencies, both of which are addressed by the Z scheme: First, the daily bandwidth needed by the level scheme varies greatly. The amount of bandwidth needed for a level 0 (or full) backup is about 25 times the amount needed by a daily level 9 (or incremental backup). Therefore, a user of the level scheme must maintain enough backup equipment to handle a level 0 backup, even though that equipment will remain idle for most of the year. Second, the lower level backups (e.g. levels 0, 5, and 7), back up all files modified since the last backup of a lower level regardless of whether those files are likely to be modified in the near future. For example, consider a file F that is modified every day, and, therefore, sent to tape by a level 7 backup every Sunday. Because file will be modified again Monday, any restore will have to pass over file F's spot on the tape used for the level 7 backup. This takes time that could be saved if the file was simply not backed up by level 7. The Z scheme addresses this problem by not writing files to higher level streams until they have not been modified for several days.[1]

4 Evaluation of Algorithms

In this section, we present comparisons of backup algorithm performance for the three metrics of interest: the amount of sequential access tape media required to perform backups and the time required to perform backup and restore operations. We evaluate the performance of four algorithms: the pure incremental, full backup, level, and Z schemes. For this paper, the Z scheme was run with a base of 8.

The results below reflect backup and restore performance for a single file system used by graduate students at the University of Maryland at Baltimore County [4]. This file system contains about 1.5 gigabytes at the beginning of the trace and approximately 4 gigabytes at the end of the trace.

The restore results use Hillyer's model [5] to determine seek time. The current simulation results assume a tape drive read rate of 1.5MB/s and a load time of 30 seconds per tape.

4.1 Media Requirements for Backup

First, we compare the cumulative total of bytes written to the backup media for each algorithm. This metric is used for comparison purposes. In practice, some algorithms might discard or re-use tapes that are no longer needed for restore. For example, both the level and the full backup schemes could discard older tapes, assuming they retain at least one subsequent full backup of the file system. However, the pure incremental scheme must retain all tapes that contain active files.

Figure 1 shows the number of bytes written by the four backup algorithms. As expected, the full backup scheme uses, by far, the most media while the pure incremental scheme uses the least. The media requirements of the level scheme are quite low, because it performs frequent incremental backups and only occasional full backups. By the end of the 8-month period in the graph, the level scheme uses approximately twice as much media as does the pure incremental scheme — about 38GB compared to about 18GB The Z scheme's performance is even closer to the pure incremental scheme, using only 25GB

[1]The intuition into why this is effective comes from Gibson, who showed that files modified today will, with high probability, either be modified within a few days, or never modified again. Similarly, files that have not been modified in several days, with high probability, not be modified again [4].

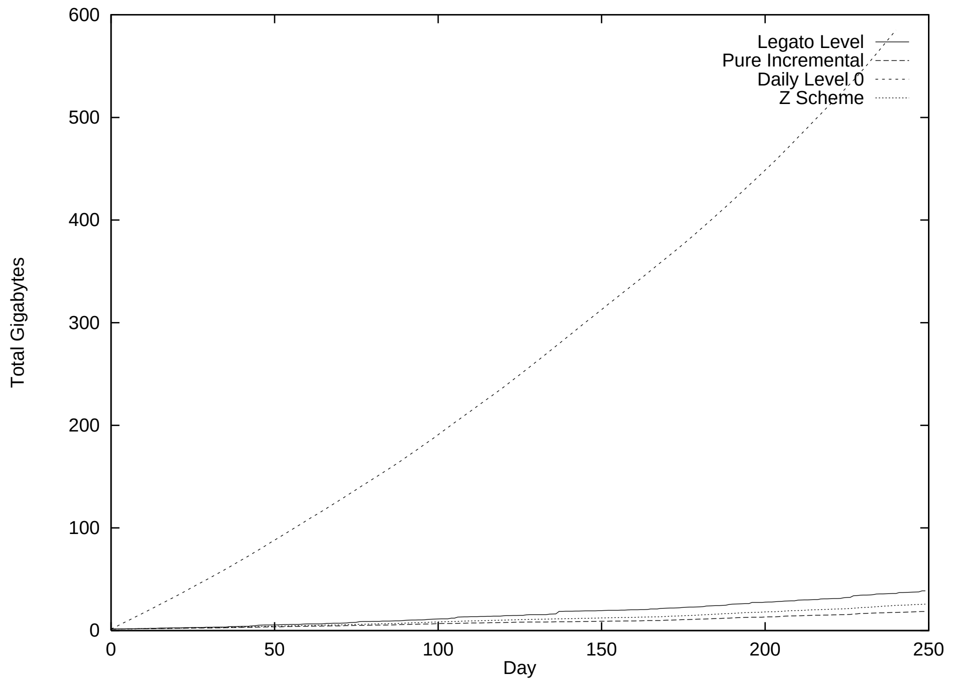

Figure 1: Total Amount of Media Consumed by Day x

4.2 Backup Performance

Figure 4.2 shows how much media each algorithm uses daily. This statistic is directly proportional to the time required to perform daily backups, and hence, to backup performance. The figure shows that the pure incremental scheme, which is optimized for fast backup, minimizes the number of bytes that need to be transferred on a given day. By contrast, the full backup scheme, which is optimized for fast restore performance, requires much more time to perform daily backups. On most days, the performance of the level scheme is similar to that of the pure incremental scheme. However, one day each week, the scheme performs a weekly "level 7" backup that requires about five times as much media as the pure incremental scheme. On one day each year, the level scheme performs a full backup that requires 25 times as much media as the pure incremental scheme does on that day.

Figure 4.2 also demonstrates the bandwidth consistency of the Z scheme (relative to the level scheme). Although on most days the Z scheme uses more media than the level scheme, it does not have the weekly and monthly peaks that the level scheme has. Therefore, as we saw in figure 1, the amount of hardware/bandwidth needed by the Z scheme is less than that of the level scheme.

One rough method of quantifying this difference in consistency is to compute the standard deviation of the amount of backup media used daily. A backup algorithm that writes the same amount of data to tape every day would have a standard deviation of zero. The level scheme has a standard deviation of 75,243, while the Z scheme has a standard deviation of about 12,284. The pure incremental scheme as a standard deviation of 11,226.

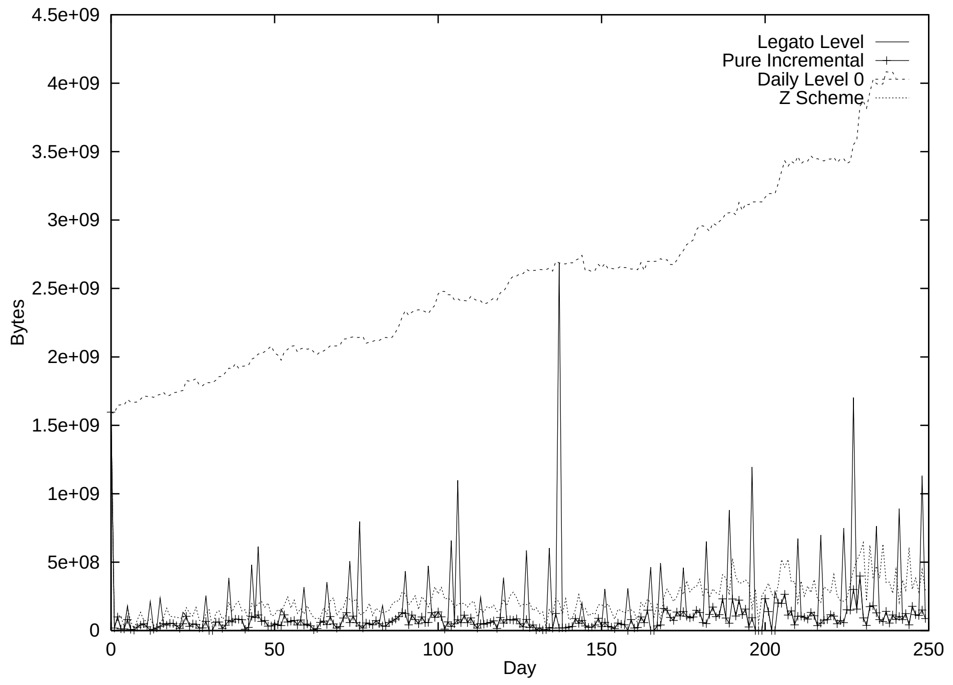

Figure 2: Amount of Media Consumed on Day x

4.3 Restore Performance

Finally, figure 3 shows how long a restore of the entire file system on day x will take. As expected, the full backup scheme, which was optimized for fast restores, has the best restore performance. The pure incremental scheme has the worst performance, because a large number of tape media must be accessed to restore all files in the file system. The level scheme performs very well on restores, within a factor of two of the full backup scheme. The restore performance of the Z scheme is almost as good as that of the level scheme. In fact, the graph of Z scheme restore time seems to connect the peaks in the graph of the level scheme restore times. This makes sense because the level scheme will have the best restore performance after a backup of low level;[2] however, because the Z scheme backs up a consistent amount of data every day, its restore times are more consistent.

Because restores occur rarely, we believe that a daily savings in backup time and media will compensate for a possible, but unlikely, doubling of restore time. Thus, we argue that the Z scheme's improvement in backup performance (especially the reduction of the peaks in bandwidth requirements) outweighs the decrease in restore performance.

5 Future Work

Our current project is to develop a distributed version of the simulator. Currently, each simulation can only simulate one file system. Although we can run several simulations in

[2]Notice that in figure 3, the restore performance dips to that of the Daily level 0 on day 135, when a level 0 backup is performed.

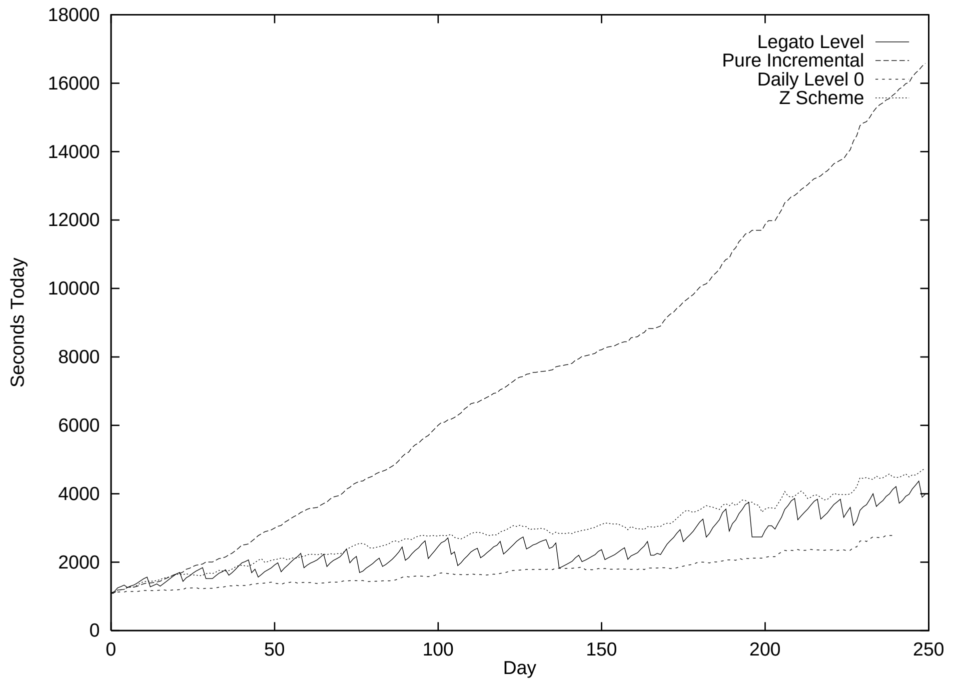

Figure 3: Time of Restore on Day x

parallel, simply summing the output of each simulation does not reasonably approximate the restore performance of a large file system (especially on a serpentine tape, such as DLT). Our distributed simulator will allow a Tape Drive object running in one process to accept read and write requests from File System objects running in several other processes. By carefully coordinating the writes to the Tape Drive object, we can make several File System objects behave as if they were subdirectories in a single, large, file system.

After we finish the distributed version of the simulator, we plan to implement several more backup algorithms. Given our results of running the Z scheme with different values for the base (not presented here due to space constraints), and Gibson's findings on file modification patterns [4], we believe that basing the actions of each stream on a power of some base is too restrictive. Therefore, we will implement the Generalized Z scheme, in which, given an array of integers A, stream i backs up those files that have not been modified in exactly A_i days. We expect that we can choose the values of A carefully so as to improve both backup and restore performance.

We plan eventually to generate results that use the larger backup traces we collected at Georgia Tech's College of Computing; however, because the Georgia Tech traces are generated by a backup algorithm, we are still working on a method to remove the influence of the backup algorithm on the traces.

Finally, our long-term goal is to progress to studying random-access media, such as CD-ROM and DVD. The cost of writers for these media is decreasing to the point where it may soon be more cost-efficient for small companies to use these media instead of magnetic tape.

6 Conclusions

Efficient backup algorithms must strike a balance between backup and restore performance. We have used a trace-driven backup simulator to evaluate four backup algorithms using traces collected by Tim Gibson. At one extreme, we showed that a pure incremental scheme performed backup operations efficiently, but performed restores inefficiently. At the other extreme, an algorithm that performed full backups every day provided efficient restores operations but slow backups.

We showed that a traditional *level* backup scheme that alternates frequent incremental backups at various "levels" with occasional full backups performs well for both backup and restore operations. For one file system trace, we showed that the total number of bytes stored by the level scheme was within a factor of two of the number of bytes stored by the pure incremental scheme. In addition, except for monthly "level 5" and annual "level 0" full backups, the daily backup performance of the level scheme was within a factor of five of the pure incremental performance. Finally, the level scheme performs restore operations efficiently, within a factor of two of the performance of the full backup scheme.

We also showed that our new *Z scheme* performs better than the level scheme for backup operations, but slightly worse for restore operations. However, because backups must take place daily, while restores take place rarely, we believe that the Z scheme's improvements in backup performance, both by using less backup media, and by having a more consistent daily bandwidth requirement, more than make up for its slight decrease in restore performance.

References

[1] A. L. Chervenak, V. Vellanki, Z. Kurmas, and V. Gupta. Protecting File Systems: A Survey of Backup Techniques. In *Proceedings*. Joint NASA and IEEE Mass Storage Conference, March 1998.

[2] A. Costello, C. Umans, and F. Wu. Online backup and restore. Unpublished Class Project at UC Berkeley, May 1998.

[3] 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.

[4] T. Gibson, E. L. Miller, , and D. D. E. Long. Long-term File Activity and Inter-Reference Patterns. In *CMG98 Proceedings*. Computer Measurement Group, December 1998.

[5] B. K. Hillyer and A. Silberschatz. On the Modeling and Performance Characteristics of a Serpentine Tape Drive. In *Proceedings SIGMETRICS*. ACM, May 1996.

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