The following paper was originally published in the
Proceedings of the Large Installation System Administration of Windows NT Conference
Seattle, Washington, August 5–8, 1998

Designing an Optimized Enterprise Windows NT/95/98 Client Backup Solution With Intellegent Data Management

Kevin Workman, Earl Waud, Steven Downs, and Mikel Featherstone
Qualcomm Inc.

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org/

Designing an Optimized Enterprise Windows NT/95/98 Client Backup Solution with Intelligent Data Management

Kevin M Workman, Earl Waud, Steven Downs, Mikel Featherston
Qualcomm Inc.

Abstract

Recent times have seen a vast increase in the number of PC's deployed on company networks, as well as a significant increase in the drive space attached to those machines. This situation presents a growing problem for those responsible for both maintaining corporate networks and insuring the integrity and safety of corporate data. The increase in the enterprise data set translates to an increase in the amount of data on the wire, as it is sent over the network to be archived, as well as an increase in the amount of tape media that must be used to protect the data. Faced with an ever-increasing amount of data to support, it is becoming necessary to discover a means to identify business critical data, so that only the necessary files are committed to long-term storage,. At the same time, the non-critical data must also be tracked, so that machines can be fully recovered in case of machine failure.

In this paper we will discuss the concept behind our system of intelligently managing redundant and unneeded information in the enterprise. Our method employs a rule system that allows us to reduce or eliminate the redundant and useless information that we backup while still providing quick restore times in the event of data loss.

1. Introduction

Experience is showing us today that in the Windows NT class machines, historical backup systems are ill equipped to deal with the massive growth we are seeing in both the sheer number of client workstation machines to manage, as well as the size of the data to be backed up.

As the industry finds larger NT/95/98 client deployments with larger, cheaper, newer technology hard drives, we discover that in large corporate environments, more than half of the information that we commit to backups is considered redundant or short term information. This data does not need to be committed to tape storage since it either exists in other areas of the enterprise, or is used for short term reasons such as Internet cache files or temporary work files.

We have identified several data types that can be handled in a much more efficient manner, through the definition of an intelligent rules system. Rather than backing up all of the data in the enterprise all of the time, we can backup only the data that matters, and reduce the amount of redundant and unneeded files that we are committing to backup via a complete inclusion/exclusion rules system.

This system also provides us with a means of allowing users the ability to define their own exclusions and inclusion in the backup set as well as a level of server dictated excludes and included based on commonality or criticality for information and files.

2. Common File Determination

In a normal enterprise with 5000 machines, each of the machines will contain a local copy of Windows NT/95 System files, and will also probably contain a multitude of workflow applications that the user keeps locally.

All of these system files, workflow applications, and cache files lend to the large amounts of data that is stored locally, and subsequently backed up to ensure that we can recreate a users machine in the event of data loss or total system loss

Many would argue that users could themselves exclude the directories and files that are redundant and do no need to be backed up, but Windows NT and Windows 95/98 lacks the concept of a user directory where user information is stored locally for the users applications and data. Thus applications and users are free to disperse their applications and data all over the structure of their local storage device, and experience has shown that this often occurs.

This makes a user driven visual exclusion system extremely unreliable, and by itself, is clearly insufficient as the sole means of determining file/directory level exclusion. In addition, the user may unknowingly exclude a critical area of their storage device thinking that only application data was stored in this area, while really also excluding from the backup system critical information needed to recreate their computing environment in the event of data loss. Finally, the users frequently fail to utilize the system at all, rendering it useless.

The answer would seem simple. When backing up the client, don't have it send files that are considered redundant, and in the event of data loss or a total system failure the server gets the needed redundant information from another tape that did have this information backed up.

This isn't a foreign concept, and many backup vendors are using this approach such as IBM's ADSM and Seagate Software's Palindrome software's "Tower of Hanoi" system. The basic premise that they work on is keep a list of files that have been backed up ***X*** number of times and once ***X*** has been reached, the server then instruct the clients to not send those files during the backup session. However these systems make an assumption that large tape changers or storage facilities are on-line. This is because a file that might be referenced for a restore would only exist on a few backup sets in the enterprise backup system. In the event of a large amount of information to be restored from tape such as an application directory, or a collection of application directories the number of tapes needed for the restore could become very large for a relatively small amount of information to be restored. This is because you could encounter massive data dispersion across the restore set referencing a multitude of tapes to recreate an area or an entire local storage device for a user.

2.1. Finding the Unique Components in the Commonality Set

In our research on how to combat this problem we looked at the file/directories across 3500 Windows NT machines in our local LAN environment that had similar applications installed on them.

The total amount of information scanned was approximately two terabytes of information across 3500 machines. After we indexed the information across all of the machines in the enterprise we found that almost 61% or 1220 gigabytes of information was common to more than one machine. After determining the size of the common information contained in the enterprise, we then re-indexed the common information to find out how much unique information was contained in the 1220 Gigabytes of information.

Example:

> The installed version of Microsoft Office is 300 megabytes of information loaded on a user machine.
>
> ***MSO=300 Megabytes on single machine***
>
> Microsoft Office is then installed on 3500 machines, so the total amount of combined space across the enterprise would be:
>
> ***MSO X 3500 Machines = 150GB***
>
> So the total amount of space to backup Microsoft Office across the enterprise would be 150GB in storage costs, but the unique amount of that common information was only 300MB.

So by applying this method on index of the total information in the enterprise that was common (***1220GB***) we find that by reducing this information down to the unique components in the commonality set, we find that for the 1220GB of common information that only 18GB of that information is unique to the commonality sampling of data.

2.2. Centralized Storage of Redundant Information

By seeing that the amount of unique information that is contained in the redundant information set is approximately two percent of the overall size of the redundant information we find that we could keep this information on-line so that the restore engine could draw from this common file repository for common files instead of having to request multiple tapes from a library system or off-line vault.

Since a dramatically less amount of workstation information will no longer be sent across the network to the backup system, we will have a dramatic reduction in the

impact on the network bandwidth in the LAN as well as quicker backup and restore times since the amount of data being transferred has been drastically reduced..

Tape cost in the enterprise will also be reduced since less information is being sent across the wire to the backup host. This results in less information that will have to be committed to tape since common information will be stored on a network shareable fileserver that backup/restore system will use for referencing common files during restores.

3. "Unique Trash" Identification

During the initial data collection of machines in the enterprise we also isolated another source of potential unneeded data that is being passed from the backup client to the backup host that would not fall within the commonality exclusion engine because of the unique nature of the data.

Many current internet applications and browsers utilized a local client side caching system where they can store commonly used files, images, and meta information that is passed to the client during web browsing or client activity.

Although this cuts down on the use of network bandwidth to the local workstation, this cache information can collect over time, and even when most local client side caching is set to a minimum of 10 megabytes on the client, if we were to multiply that across 5000 machines in the enterprise that would mean that we are backing up on the order of 50GB of information each time we do a full backup of a machine. Do to the randomness of this information caused by each client visiting different Internet sites and looking at different information, their local collection of this information would vary from machine to machine, thus negating our commonality exclusion rules system.

So now we must enact a new exclusion rule to handle this "unique trash" that would not normally be excluded by a commonality engine, and must then be picked up by an application specific unique trash rule.

As an example:

Possible unique trash filters for common applications

ServerExclude
C:\ProgramFiles\Netscape\Cache*.*
C:\Temp*.tmp
C:\Temp*.~oc

4. Incorporating the Rules System

The ability to identify the large amount of redundant data does not, by itself, give us the ability to develop a software package that takes advantage of the research that we performed. In order to do anything useful with the information we had gleaned from our network, it was necessary to create a system by which we could define what we wanted included or excluded in out backups. This system had to be both compact and easily interpreted, to minimize both the time needed to load the information and to process it. The system also had to be ultimately flexible, to allow for any and every combination that might occur. Finally, the system had to handle questions of precedence, so that conflicts between different rules (as we chose to call our definitions) could be quickly resolved.

This rule system is designed to provide the ultimate in flexibility in determining which files to back up and which to ignore, without overwhelming the user with complexity. In this system, a rule refers to some definition that describes a set of files. That set may consist of one file, several files, or no files. In addition to defining a set, a rule also describes an action to be taken with that set, either to include or to exclude those files from the backup set.

To provide the extreme level of flexibility that is part of this system, rules are divided into a total of twenty-five categories. With the exception of one rule category, all of these categories can be defined by a set of four characteristics: source, scope, direction, and type.

Source
Refers to whether a rule is defined as a client-level rule, or a server-level rule.

Scope
Determines if the rule applies to an explicitly named file, or a group of files defined using wildcards.

Direction
States if the rule defines an included set of files, or an excluded set of files.

Type
States if the rule applies to the files in a single directory (hereafter known as a file level rule), to all files in a given directory tree (hereafter known as a directory level rule), or to all files on a given machine (hereafter known as a global level rule)

Rule Categories

These four characteristics can be combined into twenty-four different categories of rules, each governing a distinct set of files. The combinations, and what they encompass, are described below:

Server Explicit File Inclusions and Exclusions
A rule in either of these categories will refer to a single fully qualified file name.

Server Wildcard File Inclusions and Exclusions
A rule in either of these categories will refer to a set of files in a given directory.

Client Explicit File Inclusions and Exclusions
A rule in either of these categories will refer to a single fully qualified file name.

Client Wildcard File Inclusions and Exclusions
A rule in either of these categories will refer to a set of files in a given directory.

Server Explicit Directory Inclusions and Exclusions
A rule in either of these categories will refer to all files in a given directory tree.

Server Wildcard Directory Inclusions and Exclusions
A rule in either of these categories will refer to a subset of the files in a directory tree.

Client Explicit Directory Inclusions and Exclusions
A rule in either of these categories will refer to all files in a given directory tree.

Client Wildcard Directory Inclusions and Exclusions
A rule in either of these categories will refer to a subset of the files in a directory tree.

Server Explicit Global Inclusions and Exclusions
A rule in either of these categories will refer to all occurrences of a given file on a system.

Server Wildcard Global Inclusions and Exclusions
A rule in either of these categories will refer to all occurrences of a set of files on a system.

Client Explicit Global Inclusions and Exclusions
A rule in either of these categories will refer to all occurrences of a given file on a system.

Client Wildcard Global Inclusions and Exclusions
A rule in either of these categories will refer to all occurrences of a set of files on a system.

The twenty-fifth rule category is a special case that is always server defined, is always an include, always defines an explicit file name, and always applies to the entire machine. How it differs from the other rules is described below. It is called an “Always Include Rule.”

4.1. Rules Priority System

In order for this system to work, a priority system is also defined, giving each rule category a level of importance with regards to other categories. This allows for a means of simply defining very complex backup sets. In determining priorities, the following considerations were used:

1. Given two rules where only the source differs, the server rule has higher priority.
2. Given two rules where only the direction differs, the inclusion rule has higher priority.
3. Given two rules where only the scope differs, the explicit rule has higher priority.
4. For rule types, file rules are higher priority than directory rules. Global rules are handled slightly differently.

Below is a table giving the priority levels and the rule categories at each level:

Level	
5	Server Explicit File, Server Always Include
4	Client Explicit File, Server Wildcard File, Server Explicit Global
3	Client Wildcard File, Server Explicit Directory, Server Wildcard Global
2	Client Explicit Directory, Server Wildcard Directory, Client Explicit Global
1	Client Wildcard Directory, Client Wildcard Global

When defining the priority scheme, file direction is used as the tiebreaker for rules at the same priority level. The twenty-fifth category is given the highest priority, due to its special nature.

It is possible to define a system with a different priority set. It is also not necessary for every rule category to be present or utilized.

The Always Include rule is used by the administrator to define files that must be backed up every time the server connects to the client, regardless of any other considerations, including user level excludes, or full/differential backup type. This is to guarantee that the most recent backup tape will always have this file, making the restoration of that file easier.

5. Optimization

There are two types of backups performed in our system: full and differential. The full backup encompasses the entire contents of the hard disk, taking into account any exclusions that are in force. The differential backups contain all files that are new or modified since the last full backup. This system is designed to minimize the data backed up, while at the same time making it as simple as possible for restores to be performed. It is possible, however, for the following situation to occur: A day or two after a full backup is performed, the user installs a huge number of files on the system. Using the backup scheme defined above, these files will be backed up every day until the next full, which may not occur for another four weeks. To solve this problem, we follow the following procedure: When the server contacts the client to perform a backup, it asks for an estimate of the backup size. The client returns both the full backup size, and the differential backup size. If the size of a differential backup would exceed a user-defined percentage of the full backup size, then the full backup is performed instead of the differential. By defining a maximum time between full backups of thirty days, we also guarantee that the last full is more accessible for restores.

6. Conclusions

Our conclusion at the end of this study was to implement our inclusion/exclusion rules system in our corporate backup system to reduce backup costs in hard dollars and infrastructure.

We are fortunate in the fact that we use an internally developed system for backing up Windows NT/ Windows 95/98 workstations in the enterprise and have complete control over our own software for clients and servers.

These modifications at the time of this writing are being integrated into our existing product for in house use and we expect to see it come to full production use sometime in the summer of 98 in our corporate LAN environment.

7. Acknowledgments

The entire development and design team would like to thank Qualcomm who made this backup project possible, and to our manager TJ Fiske who had confidence in our coding Kung Fu.

Also a special thanks to the engineering teams at Compaq, and JD Marymee at Novell Inc. for his big brain.