Chapter 18 - Choosing a File System

This chapter provides an overview of the capabilities and limitations of the FAT and NTFS file systems. Chapter 17, "Disk and File System Basics," describes the structure of FAT and NTFS volumes from the perspective of how each file system organizes the data on the disk. This chapter focuses on the user aspects of each file system.

You can use either or both of these file systems on a computer. Which one you choose for your computer, or for an individual volume on your computer, depends on such things as:

- How you expect to use the computer.
- The hardware platform.
- The size and number of hard disks.
- Security considerations.

NTFS provides several advantages when compared to FAT, which are described in this chapter. But, if you will be using another operating system in addition to Windows NT, you can access files on NTFS volumes only from Windows NT. You must use a file system other that NTFS for the system and boot partitions for the other operating systems.

For more information about file systems, especially NTFS, see *Inside the Windows NT File System*, by Helen Custer (Microsoft Press 1994, ISBN 1-55615-660-X), which documents the NTFS file system design.

Comparing FAT and NTFS File Systems

This section presents a high-level overview of why you might decide to use each of the file systems.

The FAT and NTFS file systems both support long filenames (up to 255 characters), so naming conventions do not matter when it comes to choosing the file system.

FAT File System Advantages

The FAT file system can be used with operating systems other than Windows NT, such as Windows 95, Windows for Workgroups, MS-DOS, and OS/2.

The FAT file system is best used on smaller volumes, because the FAT file system starts out with very little overhead. It works well up to about a 500 MB volume, but is very inefficient for volumes larger than 1 gigabyte (GB).

The FAT file system is a better choice than NTFS for volumes that are smaller than approximately 400-500 MB, because of the amount of disk space overhead involved in NTFS. This overhead is in the form of NTFS system files and the log file, which can use several percent of the total disk space on a small volume.

NTFS File System Advantages

The NTFS file system is best for use on volumes of about 400 MB or more, because performance does not degrade as much on larger NTFS volumes as compared to larger FAT volumes. However, if you want to use features that are available only on NTFS, such as file security or compression, you can use NTFS on smaller volumes.

There are several features that are provided only by the NTFS file system:

- You can assign permissions to individual files and folders, so you can specify who is allowed various kinds of access to a file or folder. The NTFS file system offers more permissions than the FAT file system, and you can set permissions for individual users or groups of users.
- The recoverability designed into the NTFS file system is such that a user should seldom have to run any disk repair program on an NTFS volume. In the event of a system crash, the NTFS file system uses its log file and checkpoint information to automatically restore the consistency of the file system.
- The B-tree structure of NTFS folders makes access to files on a large folder faster than access to files on a similar size folder on a FAT volume.
- You can compress individual files and folders on an NTFS volume. NTFS compression enables you to read and write the files while they are compressed, without having to first use a program to uncompress them.

There is more information about each of these features presented later in this chapter.

Which is Faster, FAT or NTFS?

There are no simple answers to this question.

For small folders, the FAT file system might provide faster access to files, because:

- The FAT structure is simpler.
- The FAT folder size is smaller for an equal number of files.
- FAT has no controls regarding whether a user can access a file or a folder Therefore, the system doesn't have to check permissions for an individual file or whether a specific user has access to the file or folder. However, Windows NT still has to determine if the file is read only, whether the file is on a FAT or NTFS volume, so the simpler controls for FAT volumes make little difference.

The NTFS file system uses a B-tree structure for all folders. This structure minimizes the number of disk accesses required to find a file, which makes access to the file faster than for a file on the FAT file system. In addition, if a folder is small enough to fit into the MFT record, you read the entire folder when you read its MFT record.

A FAT folder entry contains an index into the file allocation table, which identifies the cluster number for the first cluster of the folder. When you want to view a file, the FAT file system has to walk the folder structure to get to

the file. For example, to start Disk Administrator, which is C:\Winnt40\System32\Windisk.exe, Windows NT finds the file on a FAT primary partition or logical drive by:

- Reading the root folder of the C drive and searching in it for Winnt40.
- Reading the starting cluster of Winnt40, and searching in that folder to find the System32 folder.
- Reading the starting cluster of System32, and searching in that folder to find Windisk.exe.
- Reading each of the clusters that contain parts of the file Windisk.exe.

In comparing performance on large folders having both long and short filenames, the speed of a FAT operation depends on the operation itself and the size of the folder. Creating files on a FAT folder might be faster. Opening a file might be faster on a FAT folder if the file is at the front of the folder. If the file doesn't exist, FAT has to search the entire folder to find this out. The search takes longer on FAT than on the B-tree structure used by the NTFS file system. Folder enumeration might be faster on a FAT folder. In mathematical terms, the average time to find a file on a FAT folder is a function of N/2, where N is the number of files. On an NTFS folder, the average time is a function of Log N. For small values of N, N/2 might actually be less than or equal to LogN. For larger values of N, LogN is less than N/2.

Several factors unique to the two file systems affect the speed with which Windows NT reads or writes a file.

- Fragmentation of the file — if a file is badly fragmented, NTFS usually requires fewer disk accesses than FAT to find all of the fragments.
- Cluster size — for both file systems, the default cluster size depends on the volume size, and is always a power of two. FAT addresses are 16 bits, and NTFS addresses are 64 bits. The default FAT cluster size is based upon the fact that the file allocation table can have at most 65,535 entries, so the cluster size is a direct function of the volume size divided by 65,535. Therefore, the default cluster size for FAT volume is almost always larger than the default cluster size for an NTFS volume of the same size. The larger cluster size for a FAT volume means that there might be less fragmentation in files on a FAT volume.
- Location of small files — with the NTFS file system, the entire file is contained within the MFT record. The maximum file size that fits in the MFT record depends upon the cluster size and the number of attributes for the file.

Maximum Sizes

The maximum file size and volume size for the FAT file system is 2^{32} bytes (4 GB). FAT can support a maximum of 65,535 clusters per volume.

The NTFS architecture is designed to use numbers up to 2^{64} bytes (16 exabytes, or 18,446,744,073,709,551,616 bytes). In theory, it is possible to have a volume 2^{64} bytes in size. Even if there were hardware available to supply a logical volume of that capacity, there are other limitations to the maximum size of a volume.

Partition Tables are limited by industry standards to 2^{32} sectors. (For more information about Partition Tables, see the sections titled "Partition Table" and "Using Hard Disks With More Than 1024 Cylinders (x86-based Computers)" in Chapter 17, "Disk and File System Basics."

The sector size is a function of hardware and industry standards, and is normally 512 bytes. While sector sizes might increase in the future, the current size puts a limit on a single volume of 2 terabytes (2^{32} * 512 bytes, or 2^{41} bytes).

Windows NT provides the capability to combine discontiguous disk areas when creating volume sets, and stripe sets, but these volumes have the same limitations. Even if there were physical disks with 2 terabyte (TB) capacity, they could not be combined to create larger volumes without increasing the physical sector size.

There is one case where you can combine disks that exceed the 2 TB limit. In Disk Administrator, there is an option to extend a volume set. You do this by selecting an existing primary partition, logical drive, or volume set that is already formatted as NTFS. You can then combine it with one or more areas of free space and select **Extend Volume Set** from the **Partition** menu. You end up with a larger NTFS volume set. However, if anything goes wrong with a volume set larger than 2 TB, you cannot just reformat it. You must again start with a smaller volume set (<=2 TB) and use the **Extend Volume Set** selection to extend the volume set to larger that 2 TB.

For the time being, 2 TB should be considered the practical limit for both physical and logical volumes using NTFS.

With the NTFS file system, you can have a maximum of 2^{32} clusters per file. This is an implementation limitation, not a file size restriction.

There is no specific limit to the number of files on an NTFS volume. However, you will not be able to create any new files when your volume is so full that Windows NT cannot allocate another entry in the MFT.

Which File System to Use on Which Volumes

This is a general recommendation for determining which file systems to use for which volumes:

- Use a small FAT primary partition (between 250 and 500 MB, depending on the total disk space and page file size) as the C drive. Use this partition for both your system and boot partition.
- Format the rest of your disk space as NTFS, and use the NTFS volumes for application programs and data. You might want to create more than one NTFS volume.

This arrangement gives you the following benefits:

- You have the recoverability of the NTFS file system for the most important information, your data. (You should always maintain your system and boot partitions by using backup tapes and creating an Emergency Repair Disk, as described in Chapter 20, "Preparing for and Performing Recovery").
- FAT is more efficient for smaller volumes and NTFS is more efficient for larger volumes.
- NTFS provides file and folder security for your application programs and data.
- If you have a startup failure on an x86-based computer, this means that you can start up the computer with an MS-DOS bootable floppy disk to troubleshoot and recover from the problem.

Here is some additional information to help you choose your file system(s):

- If the computer has only Windows NT installed, you can use just the NTFS file system.
- If you want to start another operating system on x86-based computers, such as Windows 95, Windows for Workgroups, MS-DOS, or OS/2, use the FAT file system for your system partition and the boot partition(s) for the other operating system(s). You can use the NTFS file system for your Windows NT boot partition and other volumes on the computer, as long as those volumes do not need to be accessed by an operating system other than Windows NT Workstation or Windows NT Server.

These additional guidelines might affect your decision as to which file system(s) to use on your computers:

- For an x86-based computer, when Windows NT starts up, it looks for certain files in the root folder of the hard disk that contains the system partition. This partition can be formatted with either the NTFS or FAT file system. This partition should be large enough to accommodate all the files you need to access under that file system.
- For a RISC-based computer, the system partition must be formatted with the FAT file system. You can use the NTFS file system on your boot partition, which needs to be large enough for all Windows NT system components. If you configure your partitions in this way, your system partition should be 5-10 MB, and your boot partition should be at least 150 MB, 250-500 MB to allow for growth.
- Because you must format your system partition with the FAT file system on RISC-based computers, you can use Disk Administrator to secure the system partition. This prevents anyone who does not have administrative privileges from accessing the system partition in spite of the fact that it is formatted as FAT. See online Help for the Disk Administrator for more information.

  Note The system partition contains the Partition Boot Sector and other files needed to load the operating system, such as NTLDR (for x86-based computers) and OSLOADER (for RISC-based computers). The boot partition needs to include the folder with the operating system. The boot partition and the system partition can be a single partition with other folders in it.

Controlling Access to Files and Folders

On NTFS volumes, you can set permissions on files and folders that specify which groups and users have access to them, and what level of access is permitted. NTFS file and folder permissions apply both to users working at the computer where the file is stored and to users accessing the file over the network when the file is in a shared folder. With NTFS you can also set share permissions, which operate on shared folders in combination with file and folder permissions.

Note To preserve permissions when you copy or move files between NTFS folders, use the Scopy program on the *Windows NT Workstation Resource Kit* CD.

Although the NTFS file system provides access controls to individual files and folders, users can perform certain actions to files or folders even if permissions are set on a file or folder to prevent access to users.

For example, you have a folder (Dir1) containing a file (File1), and you grant Full Control to a user for the folder Dir1. If you specify that the user have No Access to File1, the user can still delete File1. This is because the user's Full Control rights in the folder allow the user to delete contents (or children) of the folder.

To prevent files from being deleted, you must set permissions on the file itself, and you must set permissions for the folder containing the file. Anyone who has Full Control in a folder can delete files from the folder.

Similarly, anyone who has List, Read, or greater permissions in a folder can view file properties on any file in the folder, even if they are prevented by file permissions from seeing the contents of the file.

With FAT volumes, you cannot set any permissions on the individual files and folders. The only security is share permissions that are set on the entire share, affect all files and folders on that share, and only function over the network. Once a folder is shared, you can protect the shared folder by specifying one set of share permissions that applies to users for all files and subfolders of the shared folder. Share permissions are set in very much the same way as file and folder permissions are set in NTFS. But because share permissions apply globally to all files and folders in the share, they are significantly less versatile than the file and folder permissions used for NTFS volumes.

Share permissions apply equally to NTFS and FAT volumes. They are enforced by Windows NT, not the individual file system.

POSIX Compliance

POSIX compliance permits UNIX programs to be ported to Windows NT. Windows NT is fully compliant with the Institute of Electrical and Electronic Engineers (IEEE) standard 1003.1, which is a standard for file naming and identification.

The following POSIX-compliant features are included in the NTFS file system:

- *Case-sensitive naming*. Under POSIX, README.TXT, Readme.txt, and readme.txt are all different files.
- *Hard links*. A file can be given more than one name. This allows two different filenames, which can be located in different folders, to point to the same data.
- *Additional time stamps*. These show when the file was last accessed or modified.

Caution You must use POSIX-based programs to manage filenames that differ only in case.

Using POSIX-based programs, you can create case-sensitive filenames, where two or more filenames can differ only in case (for example, annm.doc and AnnM.Doc).

While the Windows NT file systems support both case-preservation and case-sensitivity, you cannot use standard commands to manage filenames that differ only in case. (Standard commands include those used at the command-line — such as **copy**, **del**, and **move** — and their My Computer and Windows NT Explorer equivalents.) For example, if you type the following at the command prompt:

del AnnM.Doc

both annm.doc and AnnM.Doc are deleted.

For more information about POSIX, see Chapter 29, "POSIX Compatibility."

Features Unique to NTFS

This section describes NTFS functionality that is not supported by the FAT file system.

Multiple Data Streams

The NTFS file system supports multiple data streams. The stream name identifies a new data attribute on the file. Streams have separate opportunistic locks, file locks, allocation sizes, and file sizes, but sharing is per file.

The following is an example of an alternate stream:

```
myfile.dat:stream2
```

This feature permits related data to be managed as a single unit. For example, Macintosh computers use this type of structure to manage resource and data forks. Or, a company might create a program to keep a list of changes to the file in an alternate stream, thus keeping archive information with the current version of the file.

As another example, a library of files might exist where the files are defined as alternate streams, as in the following example:

```
library:file1
:file2
:file3
```

You could use a smart compiler to create a file structure like the following example:

```
program:source_file
:doc_file
:object_file
:executable_file
```

You can use the Win32 API CreateFile to create an alternate data stream. Or, from the command prompt, you can enter commands such as:

```
echo text>program:source_file
more <program:source_file
```

Note Because the NTFS file system is not supported on floppy disks, when you copy an NTFS file to a floppy disk, data streams and other attributes not supported by the FAT file system are lost.

Top Of Page

NTFS Compression

Windows NT supports compression on an individual file basis for NTFS volumes. Files that are compressed on an NTFS volume can be read and written without first being uncompressed by another program. Uncompression happens automatically during the read of the file. The file is compressed again when it is closed or explicitly saved.

Compressed files and folders have an attribute of C when viewed in My Computer or Windows NT Explorer. You can select an alternate color for compressed files and folders by:

- On the **View** menu, click **Options**.
- On the **View** tab, check **Display compressed files and folders with alternate color**.

Only the NTFS file system can read the compressed form of the data. When an program like Microsoft Word for Windows or an operating system command like **Copy** requests access to the file, the NTFS file system

uncompresses the file before making it available. For example, if you copy a compressed file from a another Windows NT computer to a compressed folder on your hard disk, the file will be uncompressed, copied, and recompressed.

This compression algorithm is similar to that provided by the MS-DOS 6.0 DoubleSpace® and MS-DOS 6.22 DriveSpace® compression, with one important difference — the MS-DOS functionality compresses the entire primary partition or logical drive, while the NTFS file system enables the user to compress individual files and folders in the NTFS volume.

NTFS compression is not supported for cluster sizes larger than 4K, because

- Compression causes a performance degradation.
- The reason you use very large (>4K) clusters is to improve performance in some special cases.

Therefore, you should not use compression and large clusters at the same time. In other words, compression performs reasonably well on sizes up to 4K, but beyond that size, the savings in disk space is not worth the decrease in performance. When the cluster size is >4K on an NTFS volume, none of the NTFS compression functions are available.

For information about creating and formatting volumes, see "Creating and Formatting Volumes," in Chapter 17, "Disk and File System Basics."

Note Windows NT does not support DoubleSpace or DriveSpace compression.

Compressing and Uncompressing Folders and Files

Each file and folder on an NTFS volume has a compression state, which is either Compressed or Uncompressed. The compression state of a folder does not reflect the compression state of the files in that folder. For instance, a folder can have a compression state of Compressed, yet all of the files in that folder could be Uncompressed. Or some of the files in an Uncompressed folder could be Compressed.

You can set the compression state of folders, and compress or uncompress files by using My Computer, Windows NT Explorer, or a command-line program called Compact. When using My Computer or Windows NT Explorer, you can set the compression state of an NTFS folder without changing the compression state of existing files in that folder.

You can set My Computer and Windows NT Explorer to display compressed files and folders with a different color than uncompressed files and folders. On the **View** menu, click **Options**. On the **View** tab, check **Display compressed files and folders with alternate color**.

Note Compressing the page file does not accomplish much. Windows NT does not compress an open page file, so the user sees an error message box. When trying to compress a closed paging file, the file is compressed without warning (as with any other file). When you restart Windows NT, the page file automatically reverts to the uncompressed state. For information about the page file, look at topics for virtual or virtual memory in Windows NT Help.

Using My Computer or Windows NT Explorer

Each of these programs provide the same compression functionality on NTFS volumes. With My Computer or Windows NT Explorer, you can:

- Set the compression state of an NTFS folder to Compressed or Uncompressed.
- Specify whether to compress or uncompress all files in the folder when you change the state of the folder to Compressed or Uncompressed.
- Compress or uncompress individual files in the folder.

To set the compression state of the folder:

1. Select the folder you want to compress or uncompress.
2. On the **File** menu, click **Properties** to display the **Properties** tab.
3. Select or clear the **Compressed** check box.

My Computer or Windows NT Explorer pop up a dialog box asking whether all the files and subfolders in the folder should be compressed or uncompressed. Existing files or subfolders in the NTFS folders retain their compression state unless you select **Yes** in this dialog box.

To work with individual files:

1. Select the folder you want to compress or uncompress.
2. On the **File** menu, click **Properties** to display the **Properties** tab.
3. Select or clear the **Compressed** check box.

You can also use the **Properties** tab to view the compressed size and compression ratio of the selected file.

Using the Compact Program
The Compact program is the command line version of the compression functionality in My Computer or Windows NT Explorer. The Compact command displays and alters the compression of folders and files on NTFS volumes. It also displays the compression state of folders.

There are reasons why you would want to use this program instead of My Computer or Windows NT Explorer:

- You can use Compact in a batch script.
- If the system crashed when compression or uncompression was occurring, the file or folder is marked as Compressed or Uncompressed, even if the operation did not complete. You can use Compact to force the operation to complete.

Note Unlike My Computer or Windows NT Explorer, the Compact program does not prompt you on whether you want to compress or uncompress files and subfolders when you set the compression state of a folder. It automatically does the compression or uncompression of any files that are not already in the compression state you just set for the folder.

For more information about this program, type **compact /?** at the command prompt, or see "File System Utilities" in Chapter 22, "Disk, File System, and Backup Utilities."

Effects of Moving and Copying Files

When you replace an existing file in an NTFS folder, the file might retain its compression state regardless of the compression state of the folder and the compression state of the source file. Thus, if you copy a compressed NTFS file to a compressed NTFS folder, but it replaces an uncompressed file, the resulting file will probably be uncompressed.

For example, you have a file named Testthis in two folders, To and From. The To folder has a compression state of compressed, but the file To\Testthis is uncompressed. The From folder also has a compression state of compressed, and the file From\Testthis is compressed. When you use Windows NT Explorer to copy From\Testthis to To\Testthis, the file To\Testthis will be uncompressed.

Note The effects described in this section are generally true when using Microsoft programs. Third-party utilities might function differently.

Moving and Copying Files Within NTFS Volumes

On an NTFS volume, when you move a file or folder from one folder to another, the compression attribute, like any other attribute, is retained regardless of the compression of either the target or source folder. For example, if you move an uncompressed file to a compressed folder, the file remains uncompressed after the move.

However, if you copy a file from one folder to another, the compression attribute of the file is changed to that of the target folder. For example, if you copy a compressed file to an uncompressed folder, the file is automatically uncompressed when it is copied to the folder.

Moving and Copying Files Between NTFS Volumes

When you move or copy a file or folder from one NTFS volume to another NTFS volume, it inherits the compression attributes of the target folder.

Moving and Copying Files Between FAT and NTFS Volumes

Like files copied between NTFS folders, files moved or copied from a FAT folder to an NTFS folder always inherit the compression attribute of the target folder. Moving a file from a FAT folder to an NTFS folder causes a copy of the file, followed by a delete.

Since Windows NT supports compression only for NTFS files, any compressed NTFS files moved or copied to a FAT volume are automatically uncompressed. Similarly, compressed NTFS files copied or moved to a floppy disk are automatically uncompressed.

Adding Files to an Almost Full NTFS Volume

When adding files to an NTFS volume that is almost full, you can get unexpected error messages. The philosophy behind these errors is that the NTFS file system wants to make sure it has enough disk space to write the entire file if it cannot be compressed, regardless of the degree of compression in the file when it is opened. For instance, it is possible to get a read error when you are trying to open a compressed file.

If you copy files to a compressed NTFS folder that doesn't have enough room for all of the files in their uncompressed state, you might get an error that says "...there is not enough space on the disk" even though they will all fit when compressed. Because NTFS allocates space based upon the uncompressed size of the file, you can get this error even if the files are already compressed.

This situation occurs because compression is handled asynchronously, and Windows NT uses lazy write (see *Inside the Windows NT File System* for information about lazy writes). NTFS does not wait for the compression and write of one file to complete before it begins work on subsequent files, and the system doesn't get the unused space back from compression until after the buffer is compressed.

When you are running a program and save files to a compressed folder on a volume that is almost full, the

save might or might not be successful — it depends on how much the file compresses, whether the beginning of the file compresses well, and numerous other factors.

If you can't delete any files or don't have any files that you can compress, you can usually copy all of the files if you copy the largest and/or the ones that compress best first. You can also try copying them in smaller groups rather than all at one time.

Compression Algorithm

The book *Inside the Windows NT File System* provides a high-level description of NTFS compression.

Basically, the NTFS file system provides real-time access to a compressed file, uncompressing the file when it is opened and only compressing when it is closed.

When writing a compressed file, the system reserves disk space for the uncompressed size. The system gets back unused space as each individual compression buffer gets compressed.

Note Some programs do not allocate space before beginning the save, and only pop up an error message when they run out of disk space.

NTFS compression uses a 3-byte minimum search rather that the two-byte minimum used by DoubleSpace. This type of search enables a much faster compression and uncompression (roughly two times faster), while only sacrificing two percent compression for the average text file.

Each NTFS data stream contains information that indicates if any part of the stream is compressed. Individual compressed buffers are identified by "holes" following them in the information stored for that stream. If there is a hole, then the NTFS file system knows to uncompress the preceding buffer to fill the hole.

NTFS Compression Compared to Other Methods

Other compression utilities are available to compress files on computers running Windows NT. These utilities differ from NTFS compression in the following ways:

- They usually can be run only from the command line.
- Files cannot be opened when they are in a compressed state — the file must first be uncompressed by using the companion program to the one used to compress the file. When you close the file, it is saved in an uncompressed state, and you have to use a program to compress it.

The *Windows NT Workstation Resource Kit* includes a compress program and two expand utilities. The compress program can only be run from the command line. There are two versions of the expand program: one runs from the command line, and the other is a Windows NT–based program. For more information, see "File System Utilities" in Chapter 22, "Disk, File System, and Backup Utilities."

As described earlier, the DoubleSpace and DriveSpace compression features in MS-DOS cannot be used when running Windows NT.

NTFS Compression Issues

The only check for whether a user is allowed to change the compression state on a file is whether they have read or write permission. If they have write permission, they can change the compression state locally or across the network.

The two ways to measure the performance of NTFS data compression are size and speed.

You can tell how well compression works by comparing the uncompressed and compressed file and folder sizes. The earlier section of this chapter titled "Using My Computer or Windows NT Explorer" describes using the **Properties** tab to view the compressed size and compression ratio of a selected file. The section "File System Utilities" in Chapter 22, "Disk, File System, and Backup Utilities," describes using the DirUse program to see the compressed size of folders.

Using NTFS compression might cause performance degradation. One of the reasons this might happen is that, even when copied inside the same computer, a compressed NTFS file is uncompressed, copied, and then recompressed as a new file. Similarly, on network transfers, the file is uncompressed, which affects bandwidth as well as speed.

With data compression, the question is "How will it affect performance on a computer running Windows NT Workstation or Windows NT Server?"

The current implementation of NTFS compression is definitely oriented toward Windows NT Workstation. Compression on a computer running Windows NT Workstation does not seem to produce a substantial performance degradation. No one who has started using NTFS data compression at Microsoft has lodged any complaints about performance degradation.

A computer running Windows NT Server can be another story. Some normal production servers at Microsoft (source servers and binary release servers) have been converted to use NTFS data compression, without any complaints about performance. However, using a tough server benchmark, like NetBench, shows a performance degradation in excess of 50%. Read-only or read-mostly servers, or any lightly loaded servers, may not see a severe performance penalty. Heavily loaded servers with lots of write traffic are poor candidates for data compression, as shown by NetBench.

You really need to measure the effects of data compression in your own environment.

Top Of Page

NTFS Recoverability

This section briefly discusses file system technology and describes how the NTFS file system implements data recoverability.

Until now, there were two types of file systems — careful-write file systems and lazy-write file systems. The NTFS file system introduces a third type — a recoverable file system.

A careful-write file system is designed around the idea that it is important to keep the volume structure consistent. Disk writes for each update are ordered so that if the system failed between two disk writes, the volume would be left in an understandable state, but with the possibility of an inconsistency. You seldom need to run utilities such as Chkdsk on a careful-write file system. An example of a careful-write file system is the FAT file system on MS-DOS.

The disadvantage of careful-write file systems is that serialized writes can be slow, because each disk write must be completed before the next disk write can begin.

A second kind of file system, such as the FAT file system on Windows NT and most UNIX file systems, is called a lazy-write file system. This type was designed to speed up disk accesses. A lazy-write file system uses an intelligent cache-management strategy and provides a way to recover data (such as the Chkdsk program) if there is an error when writing to the disk. All data are accessed via the file cache. While the user searches

folders or reads files, data to be written to disk accumulates in the file cache. If the same data are modified several times, all those modifications are captured in the file cache. The result is that the file system needs to write to disk only once to update the data.

Chapter 5, "Windows NT 4.0 Workstation Architecture," and Chapter 15, "Detecting Cache Bottlenecks," contain more information about the file cache.

The disadvantage of a lazy-write file system is that, in the event of a disk crash, recovery is slower, because a program such as Chkdsk must then scan the disk to check that what should have been written to disk matches what was written.

NTFS is a third kind of file system — a recoverable file system. It combines the speed of a lazy-write file system with virtually instant recovery.

NTFS guarantees the consistency of the volume by using standard transaction logging and recovery techniques, although it does not guarantee the protection of user data. It includes a lazy-write technique plus a volume-recovery technique that takes typically only a second or two to insure the integrity of all NTFS volumes each time the computer is restarted. The transaction logging, which allows NTFS to recover quickly, requires a very small amount of overhead compared with careful-write file systems.

NTFS also uses a technique called bad-cluster remapping to minimize the effects of a bad sector on an NTFS volume. For more information, see the section "Cluster Remapping," presented later in this chapter.

It is possible for the Master Boot Record or Partition Boot Sector to be corrupted due to a disk error or system crash. When either of these sectors is corrupted, you might not be able to access any data on the volume. Recovery from errors with the Master Boot Record or the Partition Boot Sector are discussed in Chapter 20, "Preparing for and Performing Recovery."

Lazy Commit

Lazy commit is an important feature of NTFS. It allows NTFS to minimize the cost of logging to maintain high performance.

Lazy commit is similar to lazy write. Instead of using system resources to mark a transaction as successfully completed as soon as it is performed, the commitment information is cached and written to the log as a background process. If the power source or computer system should fail before the commit is logged, NTFS rechecks the transaction to see whether it was successfully completed. If NTFS cannot guarantee that the transaction was completed successfully, it backs out the transaction. No incomplete modifications to the volume are allowed.

Every few seconds, NTFS checks the cache to determine the status of the lazy writer and marks the status as a checkpoint in the log. If the system crashes subsequent to that checkpoint, the system backs up to that checkpoint for recovery. This method makes recovery faster by minimizing the number of queries that are required during recovery.

Transaction Logging and Recovery

NTFS considers an I/O operation to be a transaction, in the sense that it is an integral operation. Once a disk operation is started, all of its subcomponents must be completed. If the operation fails, there must be a way to rollback the I/O request to a previously known state.

To insure that an I/O operation can be completed or rolled back, NTFS uses the Log File Service to log all redo and undo information for the transaction. Redo is the information that NTFS uses to repeat the transaction, and undo enables NTFS to roll back the transaction that was incomplete or that had an error.

If a transaction completes successfully, the file update is committed. If the transaction is incomplete, NTFS ends or rolls back the transaction by following instructions in the undo information. If NTFS detects an error in the transaction, the transaction is also rolled back.

File system recovery is straightforward with NTFS. If the system crashes, NTFS performs three passes — an analysis pass, a redo pass, and an undo pass. During the analysis pass, NTFS determines exactly which clusters must now be updated, based on the information in the log file. The redo pass performs all transaction steps logged from the last checkpoint. The undo pass backs out any incomplete (uncommitted) transactions.

Effects of File and Disk Cache on Recoverability

The file cache is just an area of RAM that contains the data. When you write data to disk, Windows NT's lazy-write technique says that the data are written, when, in fact, they are still in RAM.

There can also be cache memory on the disk controller or on the disk itself. If you are running MS-DOS or Windows 3.1*x*, using the disk or controller caches can really help performance. The lazy-write technique in Windows NT improves performance to the point where you might not see much more improvement by using the cache on the disk or the controller.

This information should help you decide whether you want to enable the disk or controller cache:

- If disk performance is an issue, and specifically if the disk is being heavily written, turning on write caching is an option for improving performance, especially if it can be measured to improve the performance of the system.
- Controlling the write-back cache is a firmware function provided by the disk manufacturer. See the documentation supplied with the disk. You cannot configure the write cache from Windows NT.
- Write caching does not impact the reliability of the file system's own structures (metadata). NTFS instructs the disk device driver to insure that metadata writes get written through regardless of whether write caching is enabled or not. Non-metadata is written to the disk normally, so caching can occur for such data.
- Read caching in the disk has no impact on file system reliability.

Cluster Remapping

With transaction logging and recovery, NTFS guarantees that the volume structure will not be corrupted, so all files remain accessible after a system crash. However, user data can be lost because of a system crash or a bad sector.

The NTFS file system implements a recovery technique called cluster remapping. When Windows NT returns a bad sector error to the NTFS file system, NTFS dynamically replaces the cluster containing the bad sector and allocates a new cluster for the data. If the error occurred during a read, NTFS returns a read error to the calling program, and the data are lost. When the error occurs during a write, NTFS writes the data to the new cluster, and no data are lost.

NTFS puts the address of the cluster containing the bad sector in its Bad Cluster File so the bad sector will not be reused.

Top Of Page

Did you find this helpful? Yes No