Defragmentation

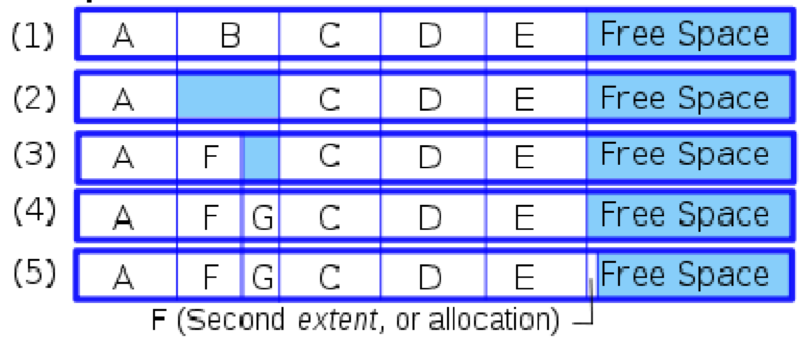
In the maintenance of file systems, **defragmentation** is a process that reduces the amount of fragmentation. It does this by physically organizing the contents of the mass storage device used to store files into the smallest number of contiguous regions (fragments). It also attempts to create larger regions of free space using compaction to impede the return of fragmentation. Some defragmentation utilities try to keep smaller files within a single directory together, as they are often accessed in sequence.

Defragmentation is advantageous and relevant to file systems on electromechanical disk drives. The movement of the hard drive's read/write heads over different areas of the disk when accessing fragmented files is slower, compared to accessing the entire contents of a non-fragmented file sequentially without moving the read/write heads to seek other fragments.

Causes of fragmentation

Fragmentation occurs when the file system cannot or will not allocate enough contiguous space to store a complete file as a unit, but instead puts parts of it in gaps between existing files (usually those gaps exist because they formerly held a file that the operating system has subsequently deleted or because the file system allocated excess space for the file in the first place). Larger files and greater numbers of files also contribute to fragmentation and consequent performance loss. Defragmentation attempts to alleviate these problems.

Example



(1) A B C D E Free Space
(2) A C D E Free Space
(3) A F C D E Free Space
(4) A F G C D E Free Space
(5) A F G C D E Free Space
F (Second extent, or allocation)

Consider the following scenario, as shown by the image:

An otherwise blank disk has five files, A through E, each using 10 blocks of space (for this section, a *block* is an allocation unit of the filesystem; the block size is set when the disk is formatted and can be any size supported by the filesystem). On a blank disk, all of these files would be allocated one after the other (see example 1 in the image). If file B were to be deleted, there would be two options: mark the space for file B as empty to be used again later, or move all the files after B so that the empty space is at the end. Since moving the files could be time consuming if there were many files which need to be moved, usually the empty space is simply left there, marked in a table as available for new files (see example 2 in the image). When a new file, F, is allocated requiring 6 blocks of space, it could be placed into the first 6 blocks of the space that formerly held file B, and the 4 blocks following it will remain available (see example 3 in the image). If another new file, G, is added and needs only 4 blocks, it could then occupy the space after F and before C (example 4 in the image). However, if file F needs to be expanded, there are three options, since the space immediately following it is no longer available:

1. Move the file F to where it can be created as one contiguous file of the new, larger size. This would not be possible if the file is larger than the largest contiguous space available. The file could also be so large that the operation would take an undesirably long period of time.
2. Move all the files after F until one opens enough space to make it contiguous again. Same problem as in the previous example, if there are a small number of files or not much data to move, it's not a big problem. If there are thousands, or tens of thousands, there isn't enough time to move all those files.
3. Add a new block somewhere else, and indicate that F has a second *extent* (see example 5 in the image). Repeat this hundreds of times and the filesystem will have a number of small free segments scattered in many places, and some files will have multiple extents. When a file has many extents like this, access time for that file may become excessively long because of all the random seeking the disk will have to do when reading it.

Additionally, the concept of “fragmentation” is not only limited to individual files that have multiple extents on the disk. For instance, a group of files normally read in a particular sequence (like files accessed by a program when it is loading, which can include certain DLLs, various resource files, the audio/visual media files in a game) can be considered fragmented if they are not in sequential load-order on the disk, even if these individual files are *not* fragmented; the read/write heads will have to seek these (defragmented) files randomly to access them in sequence. Some groups of files may have been originally installed in the correct sequence, but drift apart with time as certain files within the group are deleted. Updates are a common cause of this, because in order to update a file, most updaters usually delete the old file first, and then write a new, updated one in its place. However, most filesystems do not write the new file in the same physical place on the disk. This allows unrelated files to fill in the empty spaces left behind. In Windows, a good defragmenter will read the Prefetch files to identify as many of these file groups as possible and place the files within them in access sequence. Another frequently good assumption is that files in any given folder are related to each other and might be accessed together.

To defragment a disk, defragmentation software (also known as a "defragmenter") can only move files around within the free space available. This is an intensive operation and cannot be performed on a filesystem with little or no free space. During defragmentation, system performance will be degraded, and it is best to leave the computer alone during the process so that the defragmenter does not get confused by unexpected changes to the filesystem. Depending on the algorithm used it may or may not be advantageous to perform multiple passes. The reorganization involved in defragmentation does not change logical location of the files (defined as their location within the directory structure).

Common countermeasures

Partitioning

A common strategy to optimize defragmentation and to reduce the impact of fragmentation is to partition the hard disk(s) in a way that separates partitions of the file system that experience many more reads than writes from the more volatile zones where files are created and deleted frequently. The directories that contain the users' profiles are modified constantly (especially with the Temp directory and web browser cache creating thousands of files that are deleted in a few days). If files from user profiles are held on a dedicated partition (as is commonly done on UNIX recommended files systems, where it is typically stored in the /var partition), the defragmenter runs better since it does not need to deal with all the static files from other directories. For partitions with relatively little write activity, defragmentation time greatly improves after the first defragmentation, since the defragmenter will need to defragment only a small number of new files in the future.

Offline defragmentation

The presence of immovable system files, especially a swap file, can impede defragmentation. These files can be safely moved when the operating system is not in use. For example, ntfsresize moves these files to resize an NTFS partition. The tool PageDefrag could defragment Windows system files such as the swap file and the files that store theWindows registry by running at boot time before the GUI is loaded. Since Windows Vista, the feature is not fully supported and has not been updated.

In NTFS, as files are added to the disk, the Master File Table (MFT) must grow to store the information for the new files. Every time the MFT cannot be extended due to some file being in the way, the MFT will gain a fragment. In early versions of Windows, it could not be safely defragmented while the partition was mounted, and so Microsoft wrote a hardblock in the defragmenting API. However, since Windows XP, an increasing number of defragmenters are now able to defragment the MFT, because the Windows defragmentation API has been improved and now supports that move operation. Even with the improvements, the first four clusters of the MFT remain unmovable by the Windows defragmentation API, resulting in the fact that some defragmenters will store the MFT in two fragments: The first four clusters wherever they were placed when the disk was formatted, and then the rest of the MFT at the beginning of the disk (or wherever the defragmenter's strategy deems to be the best place).

User and performance issues

In a wide range of modern multi-user operating systems, an ordinary user cannot defragment the system disks since superuser (or "Administrator") access is required to move system files. Additionally, file systems such as NTFS are designed to decrease the likelihood of fragmentation. Improvements in modern hard drives such as RAM cache, faster platter rotation speed, command queuing (SCSI/ATA TCQ or SATA NCQ), and greater data density reduce the negative impact of fragmentation on system performance to some degree, though increases in commonly used data quantities offset those benefits. However, modern systems profit enormously from the huge disk capacities currently available, since partially filled disks fragment much less than full disks,[5] and on a high-capacity HDD, the same partition occupies a smaller range of cylinders, resulting in faster seeks. However, the average access time can never be lower than a half rotation of the platters, and platter rotation (measured in rpm) is the speed characteristic of HDDs which has experienced the slowest growth over the decades (compared to data transfer rate and seek time), so minimizing the number of seeks remains beneficial in most storage-heavy applications. Defragmentation is just that: ensuring that there is at most one seek per file, counting only the seeks to non-adjacent tracks.

When reading data from a conventional electromechanical hard disk drive, the disk controller must first position the head, relatively slowly, to the track where a given fragment resides, and then wait while the disk platter rotates until the fragment reaches the head.

Since disks based on flash memory have no moving parts, random access of a fragment does not suffer this delay, making defragmentation to optimize access speed unnecessary. Furthermore, since flash memory can be written to only a limited number of times before it fails, defragmentation is actually detrimental (except in the mitigation ofcatastrophic failure).

Windows System Restore points may be deleted during defragmenting/optimizing

Running most defragmenters and optimizers can cause the Microsoft Shadow Copy service to delete some of the oldest restore points, even if the defragmenters/optimizers are built on Windows API. This is due to Shadow Copy keeping track of some movements of big files performed by the defragmenters/optimizers; when the total disk space used by shadow copies would exceed a specified threshold, older restore points are deleted until the limit is not exceeded.

Defragmenting and optimizing

Besides defragmenting program files, the defragmenting tool can also reduce the time it takes to load programs and open files. For example, the Windows 9x defragmenter included the Intel Application Launch Accelerator which optimized programs on the disk by placing the defragmented program files and their dependencies next to each other, in the order of which the program loads them, to load these programs faster. At the beginning of the hard drive, the outer tracks have a higher transfer rate than the inner tracks. Placing frequently accessed files onto the outer tracks increases performance. Third party defragmenters, such as MyDefrag, will move frequently accessed files onto the outer tracks and defragment these files.

Approach and defragmenters by file-system type

- FAT: MS-DOS 6.x and Windows 9x-systems come with a defragmentation utility called Defrag. The DOS version is a limited version of Norton SpeedDisk. The version that came with Windows 9x was licensed from Symantec Corporation, and the version that came with Windows 2000 and XP is licensed from Condusiv Technologies.

- NTFS was introduced with Windows NT 3.1, but the NTFS filesystem driver did not include any defragmentation capabilities. In Windows NT 4.0, defragmenting APIs were introduced that third-party tools could use to perform defragmentation tasks; however, no defragmentation software was included. In Windows 2000, Windows XP andWindows Server 2003, Microsoft included a defragmentation tool based on Diskeeper that made use of the defragmentation APIs and was a snap-in for Computer Management. In Windows Vista, Windows 7 and Windows 8, the tool has been greatly improved and was given a new interface with no visual diskmap and is no longer part of Computer Management. There are also a number of free and commercial third-party defragmentation products available for Microsoft Windows.
- BSD UFS and particularly FreeBSD uses an internal reallocator that seeks to reduce fragmentation right in the moment when the information is written to disk. This effectively controls system degradation after extended use.
- Linux ext2, ext3, and ext4: Much like UFS, these filesystems employ allocation techniques designed to keep fragmentation under control at all times. As a result, defragmentation is not needed in the vast majority of cases. ext2 uses an offline defragmenter called `e2defrag`, which does not work with its successor ext3. However, other programs, or filesystem-independent ones such as defragfs, may be used to defragment an ext3 filesystem. ext4 is somewhat backward compatible with ext3, and thus has generally the same amount of support from defragmentation programs. Nowadays e4defrag can be used to defragment an ext4 filesystem.
- VxFS has the `fsadm` utility that includes defrag operations.
- JFS has the `defragfs` utility on IBM operating systems.
- HFS Plus introduced with Mac OS 8.1 in 1998 a number of optimizations to the allocation algorithms in an attempt to defragment files while they are being accessed without a separate defragmenter.
- WAFL in NetApp's ONTAP 7.2 operating system has a command called `reallocate` that is designed to defragment large files.
- XFS provides an online defragmentation utility called `xfs_fsr`.
- SFS processes the defragmentation feature in almost completely stateless way (apart from the location it is working on), so defragmentation can be stopped and started instantly.
- ADFS, the file system used by RISC OS and earlier Acorn Computers, keeps file fragmentation under control without requiring manual defragmentation.

Disk Cleanup

Disk Cleanup (cleanmgr.exe) is a computer maintenance utility included in Microsoft Windows designed to free up disk space on a computer's hard drive. The utility first searches and analyzes the hard drive for files that are no longer of any use, and then removes the unnecessary files. There are a number of different file categories that Disk Cleanup targets when performing the initial disk analysis:

- Compression of old files
- Temporary Internet files

- Temporary Windows files
- Downloaded program files
- Recycle Bin
- Removal of unused applications or optional Windows components
- Setup log files
- Offline web pages (cached)

The above list, however, is not exhaustive. For instance, 'Temporary Remote Desktop files' and 'Temporary Sync Files' may appear only under certain computer configurations, differences such as Windows Operating System and use of additional programs such as Remote Desktop. The option of removing hibernation data may not be ideal for some users as this may remove the hibernate option.

Aside from removing unnecessary files, users also have the option of compressing files that have not been accessed over a set period of time. This option provides a systematic compression scheme. Infrequently accessed files are compressed to free up disk space while leaving the frequently used files uncompressed for faster read/write access times. If after file compression, a user wishes to access a compressed file, the access times may be increased and vary from system to system. In addition to the categories that appear on the Disk Cleanup tab, the More Options tab offers additional options for freeing up hard drive space through removal of optional Windows components, installed programs, and all but the most recent System Restore point or Shadow Copy data in some versions of Microsoft Windows.

Backup

In information technology, a **backup**, or the process of backing up, refers to the copying and archiving of computer data so it may be used to *restore* the original after a data lossevent. The verb form is to **back up** in two words, whereas the noun is *backup*.[1]

Backups have two distinct purposes. The primary purpose is to recover data after its loss, be it by data deletion or corruption. Data loss can be a common experience of computer users. A 2008 survey found that 66% of respondents had lost files on their home PC.[2] The secondary purpose of backups is to recover data from an earlier time, according to a user-defined data retention policy, typically configured within a backup application for how long copies of data are required. Though backups popularly represent a simple form of disaster recovery, and should be part of a disaster recovery plan, by themselves, backups should not alone be considered disaster recovery.[3] One reason for this is that not all backup systems or backup applications are able to reconstitute a computer system or other complex configurations such as a computer cluster, active directoryservers, or a database server, by restoring only data from a backup.

Since a backup system contains at least one copy of all data worth saving, the data storage requirements can be significant. Organizing this storage space and managing the backup process can be a complicated undertaking. A data repository model can be used to provide structure to the storage. Nowadays, there are many different types of data storage devices that are useful for making backups. There are also many different ways in which these devices can be arranged to provide geographic redundancy, data security, and portability.

Before data are sent to their storage locations, they are selected, extracted, and manipulated. Many different techniques have been developed to optimize the backup procedure. These include optimizations for dealing with open files and live data sources as well as compression, encryption, and de-duplication, among others. Every backup scheme should include dry runs that validate the reliability of the data being backed up. It is important to recognize the limitations and human factors involved in any backup scheme.

Storage, the base of a backup system

Data repository models

Any backup strategy starts with a concept of a data repository. The backup data needs to be stored, and probably should be organized to a degree. The organisation could be as simple as a sheet of paper with a list of all backup media (CDs etc.) and the dates they were produced. A more sophisticated setup could include a computerized index, catalog, or relational database. Different approaches have different advantages. Part of the model is the backup rotation scheme.

Unstructured

An unstructured repository may simply be a stack of or CD-Rs or DVD-Rs with minimal information about what was backed up and when. This is the easiest to implement, but probably the least likely to achieve a high level of recoverability.

Full only / System imaging

A repository of this type contains complete system images taken at one or more specific points in time. This technology is frequently used by computer technicians to record known good configurations. Imaging is generally more useful for deploying a standard configuration to many systems rather than as a tool for making ongoing backups of diverse systems.

Incremental

An incremental style repository aims to make it more feasible to store backups from more points in time by organizing the data into increments of change between points in time. This eliminates the need to store duplicate copies of unchanged data: with full backups a lot of the data will be unchanged from what has been backed up previously. Typically, a *full* backup (of all files) is made on one occasion (or at infrequent intervals) and serves as the reference point for an incremental backup set. After that, a number of *incremental* backups are made after successive time periods. Restoring the whole system to the date of the last incremental backup would require starting from the last full backup taken before the data loss, and then applying in turn each of the incremental backups since then.[4] Additionally, some backup systems can reorganize the repository to synthesize full backups from a series of incrementals.

Differential

Each differential backup saves the data that has changed since the last full backup. It has the advantage that only a maximum of two data sets are needed to restore the data. One disadvantage, compared to the incremental backup method, is that as time from the last full backup (and thus the accumulated changes in data) increases, so does the time to perform the differential backup. Restoring an entire system would require starting from the most recent full backup and then applying just the last differential backup since the last full backup.

Note: Vendors have standardized on the meaning of the terms "incremental backup" and "differential backup". However, there have been cases where conflicting definitions of these terms have been used. The most relevant characteristic of an incremental backup is which reference point it uses to check for changes. By standard definition, a differential backup copies files that have been created or changed since the last full backup, regardless of whether any other differential backups have been made since then, whereas an incremental backup copies files that have been created or changed since the most recent backup of any type (full or incremental). Other variations of incremental backup include multi-level incrementals and incremental backups that compare parts of files instead of just the whole file.

Reverse delta

A reverse delta type repository stores a recent "mirror" of the source data and a series of differences between the mirror in its current state and its previous states. A reverse delta backup will start with a normal full backup. After the full backup is performed, the system will periodically synchronize the full backup with the live copy, while storing the data necessary to reconstruct older versions. This can either be done using hard links, or using binary diffs. This system works particularly well for large, slowly changing, data sets. Examples of programs that use this method are rdiff-backup and Time Machine.

Continuous data protection

Instead of scheduling periodic backups, the system immediately logs every change on the host system. This is generally done by saving byte or block-level differences rather than file-level differences. It differs from simple disk mirroring in that it enables a roll-back of the log and thus restoration of old image of data.

Storage media

Regardless of the repository model that is used, the data has to be stored on some data storage medium.

Magnetic tape

Magnetic tape has long been the most commonly used medium for bulk data storage, backup, archiving, and interchange. Tape has typically had an order of magnitude better capacity/price ratio when compared to hard disk, but recently the ratios for tape and hard disk have become a lot closer. There are many formats, many of which are proprietary or specific to certain markets like mainframes or a particular brand of personal computer. Tape is a sequential access medium, so even though access times may be poor, the rate of continuously writing or reading data can actually be very fast. Some new tape drives are even faster than modern hard disks.

Hard disk

The capacity/price ratio of hard disk has been rapidly improving for many years. This is making it more competitive with magnetic tape as a bulk storage medium. The main advantages of hard disk storage are low access times, availability, capacity and ease of use.[7] External disks can be connected via local interfaces like SCSI, USB, FireWire, or eSATA, or via longer distance technologies like Ethernet, iSCSI, or Fibre Channel. Some disk-based backup systems, such as Virtual Tape Libraries, support data deduplication which can dramatically reduce the amount of disk storage capacity consumed by daily and weekly backup data. The main disadvantages of hard disk backups are that they are easily damaged, especially while being transported (e.g., for off-site backups), and that their stability over periods of years is a relative unknown.

Optical storage

Recordable CDs, DVDs, and Blu-ray Discs are commonly used with personal computers and generally have low media unit costs. However, the capacities and speeds of these and other optical discs are typically an order of magnitude lower than hard disk or tape. Many optical disk formats are WORM type, which makes them useful for archival purposes since the data cannot be changed. The use of an auto-changer or jukebox can make optical discs a feasible option for larger-scale backup systems. Some optical storage systems allow for cataloged data backups without human contact with the discs, allowing for longer data integrity.

Solid state storage

Also known as flash memory, thumb drives, USB flash drives, CompactFlash, SmartMedia, Memory Stick, Secure Digital cards, etc., these devices are relatively expensive for their low capacity, but are very convenient for backing up relatively low data volumes. A solid-state drive does not contain any movable parts unlike its magnetic drive counterpart and can have huge throughput in the order of 500Mbit/s to 6Gbit/s. SSD drives are now available in the order of 500GB to TBs.

Remote backup service

As broadband Internet access becomes more widespread, remote backup services are gaining in popularity. Backing up via the Internet to a remote location can protect against some worst-case scenarios such as fires, floods, or earthquakes which would destroy any backups in the immediate vicinity along with everything else. There are, however, a number of drawbacks to remote backup services. First, Internet connections are usually slower than local data storage devices. Residential broadband is especially problematic as routine backups must use an upstream link that's usually much slower than the downstream link used only occasionally to retrieve a file from backup. This tends to limit the use of such services to relatively small amounts of high value data. Secondly, users must trust a third party service provider to maintain the privacy and integrity of their data, although confidentiality can be assured by encrypting the data before transmission to the backup service with an encryption key known only to the user. Ultimately the backup service must itself use one of the above methods so this could be seen as a more complex way of doing traditional backups.

Floppy disk

During the 1980s and early 1990s, many personal/home computer users associated backing up mostly with copying to floppy disks. However, the data capacity of floppy disks failed to catch up with growing demands, rendering them effectively obsolete.

Managing the data repository

Regardless of the data repository model, or data storage media used for backups, a balance needs to be struck between accessibility, security and cost. These media management methods are not mutually exclusive and are frequently combined to meet the user's needs. Using on-line disks for staging data before it is sent to a near-line tape library is a common example.

On-line

On-line backup storage is typically the most accessible type of data storage, which can begin restore in milliseconds of time. A good example is an internal hard disk or a disk array (maybe connected to SAN). This type of storage is very convenient and speedy, but is relatively expensive. On-line storage is quite vulnerable to being deleted or overwritten, either by accident, by intentional malevolent action, or in the wake of a data-deleting virus payload.

Near-line

Near-line storage is typically less accessible and less expensive than on-line storage, but still useful for backup data storage. A good example would be a tape library with restore times ranging from seconds to a few minutes. A mechanical device is usually used to move media units from storage into a drive where the data can be read or written. Generally it has safety properties similar to on-line storage.

Off-line

Off-line storage requires some direct human action to provide access to the storage media: for example inserting a tape into a tape drive or plugging in a cable. Because the data are not accessible via any computer except during limited periods in which they are written or read back, they are largely immune to a whole class of on-line backup failure modes. Access time will vary depending on whether the media are on-site or off-site.

Off-site data protection

To protect against a disaster or other site-specific problem, many people choose to send backup media to an off-site vault. The vault can be as simple as a system administrator's home office or as sophisticated as a disaster-hardened, temperature-controlled, high-security bunker with facilities for backup media storage. Importantly a data replica *can* be off-site but also *on-line* (e.g., an off-site RAID mirror). Such a replica has fairly limited value as a backup, and should not be confused with an off-line backup.

Backup site or disaster recovery center (DR center)

In the event of a disaster, the data on backup media will not be sufficient to recover. Computer systems onto which the data can be restored and properly configured networks are necessary too. Some organizations have their own data recovery centers that are equipped for this scenario. Other organizations contract this out to a third-party recovery center. Because a DR site is itself a huge investment, backing up is very rarely considered the preferred method of moving data to a DR site. A more typical way would be remote disk mirroring, which keeps the DR data as up to date as possible.

Selection and extraction of dat

A successful backup job starts with selecting and extracting coherent units of data. Most data on modern computer systems is stored in discrete units, known as files. These files are organized into filesystems. Files that are actively being updated can be thought of as "live" and present a challenge to back up. It is also useful to save metadata that describes the computer or the filesystem being backed up.

Deciding what to back up at any given time is a harder process than it seems. By backing up too much redundant data, the data repository will fill up too quickly. Backing up an insufficient amount of data can eventually lead to the loss of critical information.

Files

With **file-level** approach, making copies of files is the simplest and most common way to perform a backup. A means to perform this basic function is included in all backup software and all operating systems.

Partial file copying

Instead of copying whole files, one can limit the backup to only the blocks or bytes within a file that have changed in a given period of time. This technique can use substantially less storage space on the backup medium, but requires a high level of sophistication to reconstruct files in a restore situation. Some implementations require integration with the source file system.

Filesystems

Instead of copying files within a file system, a copy of the whole filesystem itself in **block-level** can be made. This is also known as a *raw partition backup* and is related todisk imaging. The process usually involves unmounting the filesystem and running a program like dd (Unix). Because the disk is read sequentially and with large buffers, this type of backup can be much faster than reading every file normally, especially when the filesystem contains many small files, is highly fragmented, or is nearly full. But because this method also reads the free disk blocks that contain no useful data, this method can also be slower than conventional reading, especially when the filesystem is nearly empty. Some filesystems, such as XFS, provide a "dump" utility that reads the disk sequentially for high performance while skipping unused sections. The corresponding restore utility can selectively restore individual files or the entire volume at the operator's choice.

Identification of changes

Some filesystems have an archive bit for each file that says it was recently changed. Some backup software looks at the date of the file and compares it with the last backup to determine whether the file was changed.

Versioning file system

A versioning filesystem keeps track of all changes to a file and makes those changes accessible to the user. Generally this gives access to any previous version, all the way back to the file's creation time. An example of this is the Wayback versioning filesystem for Linux.

Live data

If a computer system is in use while it is being backed up, the possibility of files being open for reading or writing is real. If a file is open, the contents on disk may not correctly represent what the owner of the file intends. This is especially true for database files of all kinds. The term fuzzy backup can be used to describe a backup of live data that looks like it ran correctly, but does not represent the state of the data at any single point in time. This is because the data being backed up changed in the period of time between when the backup started and when it finished. For databases in particular, fuzzy backups are worthless.

Snapshot backup

A snapshot is an instantaneous function of some storage systems that presents a copy of the file system as if it were frozen at a specific point in time, often by a copy-on-writemechanism. An effective way to back up live data is to temporarily quiesce them (e.g. close all files), take a snapshot, and then resume live operations. At this point the snapshot can be backed up through normal methods. While a snapshot is very handy for viewing a filesystem as it was at a different point in time, it is hardly an effective backup mechanism by itself.

Open file backup

Many backup software packages feature the ability to handle open files in backup operations. Some simply check for openness and try again later. File locking is useful for regulating access to open files.

When attempting to understand the logistics of backing up open files, one must consider that the backup process could take several minutes to back up a large file such as a database. In order to back up a file that is in use, it is vital that the entire backup represent a single-moment snapshot of the file, rather than a simple copy of a read-through. This represents a challenge when backing up a file that is constantly changing. Either the database file must be locked to prevent changes, or a method must be implemented to ensure that the original snapshot is preserved long enough to be copied, all while changes are being preserved. Backing up a file while it is being changed, in a manner that causes the first part of the backup to represent data *before* changes occur to be combined with later parts of the backup *after* the change results in a corrupted file that is unusable, as most large files contain internal references between their various parts that must remain consistent throughout the file.

Cold database backup

During a cold backup, the database is closed or locked and not available to users. The datafiles do not change during the backup process so the database is in a consistent state when it is returned to normal operation.[11]

Hot database backup

Some database management systems offer a means to generate a backup image of the database while it is online and usable ("hot"). This usually includes an inconsistent image of the data files plus a log of changes made while the procedure is running. Upon a restore, the changes in the log files are reapplied to bring the copy of the database up-to-date (the point in time at which the initial hot backup ended).[12]

Metadata

Not all information stored on the computer is stored in files. Accurately recovering a complete system from scratch requires keeping track of this non-file data too.

System description

System specifications are needed to procure an exact replacement after a disaster.

Boot sector

The boot sector can sometimes be recreated more easily than saving it. Still, it usually isn't a normal file and the system won't boot without it.

Partition layout

The layout of the original disk, as well as partition tables and filesystem settings, is needed to properly recreate the original system.

File **metadata**

Each file's permissions, owner, group, ACLs, and any other metadata need to be backed up for a restore to properly recreate the original environment.

System metadata

Different operating systems have different ways of storing configuration information. Microsoft Windows keeps a registry of system information that is more difficult to restore than a typical file.

Manipulation of data and dataset optimization

It is frequently useful or required to manipulate the data being backed up to optimize the backup process. These manipulations can provide many benefits including improved backup speed, restore speed, data security, media usage and/or reduced bandwidth requirements.

Compression

Various schemes can be employed to shrink the size of the source data to be stored so that it uses less storage space. Compression is frequently a built-in feature of tape drive hardware.

Deduplication

When multiple similar systems are backed up to the same destination storage device, there exists the potential for much redundancy within the backed up data. For example, if 20 Windows workstations were backed up to the same data repository, they might share a common set of system files. The data repository only needs to store one copy of those files to be able to restore any one of those workstations. This technique can be applied at the file level or even on raw blocks of data, potentially resulting in a massive reduction in required storage space. Deduplication can occur on a server before any data moves to backup media, sometimes referred to as source/client side deduplication. This approach also reduces bandwidth required to send backup data to its target media. The process can also occur at the target storage device, sometimes referred to as inline or back-end deduplication.

Duplication

Sometimes backup jobs are duplicated to a second set of storage media. This can be done to rearrange the backup images to optimize restore speed or to have a second copy at a different location or on a different storage medium.

Encryption

High capacity removable storage media such as backup tapes present a data security risk if they are lost or stolen. Encrypting the data on these media can mitigate this problem, but presents new problems. Encryption is a CPU intensive process that can slow down backup speeds, and the security of the encrypted backups is only as effective as the security of the key management policy.

Multiplexing

When there are many more computers to be backed up than there are destination storage devices, the ability to use a single storage device with several simultaneous backups can be useful.

Refactoring

The process of rearranging the backup sets in a data repository is known as refactoring. For example, if a backup system uses a single tape each day to store the incremental backups for all the protected computers, restoring one of the computers could potentially require many tapes. Refactoring could be used to consolidate all the backups for a single computer onto a single tape. This is especially useful for backup systems that do *incrementals forever* style backups.

Staging

Sometimes backup jobs are copied to a staging disk before being copied to tape. This process is sometimes referred to as D2D2T, an acronym for Disk to Disk to Tape. This can be useful if there is a problem matching the speed of the final destination device with the source device as is frequently faced in network-based backup systems. It can also serve as a centralized location for applying other data manipulation techniques.

Managing the backup process

As long as new data are being created and changes are being made, backups will need to be performed at frequent intervals. Individuals and organizations with anything from one computer to thousands of computer systems all require protection of data. The scales may be very different, but the objectives and limitations are essentially the same. Those who perform backups need to know how successful the backups are, regardless of scale.

Objectives

Recovery point objective (RPO)

The point in time that the restarted infrastructure will reflect. Essentially, this is the roll-back that will be experienced as a result of the recovery. The most desirable RPO would be the point just prior to the data loss event. Making a more recent recovery point achievable requires increasing the frequency of synchronization between the source data and the backup repository.[14]

Recovery time objective (RTO)

The amount of time elapsed between disaster and restoration of business functions.[15]

Data security

In addition to preserving access to data for its owners, data must be restricted from unauthorized access. Backups must be performed in a manner that does not compromise the original owner's undertaking. This can be achieved with data encryption and proper media handling policies.

Limitations

An effective backup scheme will take into consideration the limitations of the situation.

Backup window

The period of time when backups are permitted to run on a system is called the backup window. This is typically the time when the system sees the least usage and the backup process will have the least amount of interference with normal operations. The backup window is usually planned with users' convenience in mind. If a backup extends past the defined backup window, a decision is made whether it is more beneficial to abort the backup or to lengthen the backup window.

Performance impact

All backup schemes have some performance impact on the system being backed up. For example, for the period of time that a computer system is being backed up, the hard drive is busy reading files for the purpose of backing up, and its full bandwidth is no longer available for other tasks. Such impacts should be analyzed.

Costs of hardware, software, labor

All types of storage media have a finite capacity with a real cost. Matching the correct amount of storage capacity (over time) with the backup needs is an important part of the design of a backup scheme. Any backup scheme has some labor requirement, but complicated schemes have considerably higher labor requirements. The cost of commercial backup software can also be considerable.

Network bandwidth

Distributed backup systems can be affected by limited network bandwidth.

Implementation

Meeting the defined objectives in the face of the above limitations can be a difficult task. The tools and concepts below can make that task more achievable.

Scheduling

Using a job scheduler can greatly improve the reliability and consistency of backups by removing part of the human element. Many backup software packages include this functionality.

Authentication

Over the course of regular operations, the user accounts and/or system agents that perform the backups need to be authenticated at some level. The power to copy all data off of or onto a system requires unrestricted access. Using an authentication mechanism is a good way to prevent the backup scheme from being used for unauthorized activity.

Chain of trust

Removable storage media are physical items and must only be handled by trusted individuals. Establishing a chain of trusted individuals (and vendors) is critical to defining the security of the data.

Measuring the process

To ensure that the backup scheme is working as expected, key factors should be monitored and historical data maintained.

Backup validation

(also known as "backup success validation") Provides information about the backup, and proves compliance to regulatory bodies outside the organization: for example, an insurance company in the USA might be required under HIPAA to demonstrate that its client data meet records retention requirements. Disaster, data complexity, data value and increasing dependence upon ever-growing volumes of data all contribute to the anxiety around and dependence upon successful backups to ensure business continuity. Thus many organizations rely on third-party or "independent" solutions to test, validate, and optimize their backup operations (backup reporting).

Reporting

In larger configurations, reports are useful for monitoring media usage, device status, errors, vault coordination and other information about the backup process.

Logging

In addition to the history of computer generated reports, activity and change logs are useful for monitoring backup system events.

Validation

Many backup programs use checksums or hashes to validate that the data was accurately copied. These offer several advantages. First, they allow data integrity to be verified without reference to the original file: if the file as stored on the backup medium has the same checksum as the saved value, then it is very probably correct. Second, some backup programs can use checksums to avoid making redundant copies of files, and thus improve backup speed. This is particularly useful for the de-duplication process.

Monitored backup

Backup processes are monitored by a third party monitoring center, which alerts users to any errors that occur during automated backups. Monitored backup requires software capable of pinging the monitoring center's servers in the case of errors. Some monitoring services also allow collection of historical meta-data, that can be used for Storage Resource Management purposes like projection of data growth, locating redundant primary storage capacity and reclaimable backup capacity.