LOCAL FILE SYSTEMS

Disk subsystems provide block-oriented storage. For end users and for higher applications the handling of blocks addressed via cylinders, tracks and sectors is very cumbersome. File systems therefore represent an intermediate layer in the operating system that provides users with the familiar directories or folders and files and stores these on the block-oriented storage media so that they are hidden to the end users. This chapter introduces the basics of files systems and shows the role that they play in connection with storage networks.

LOCAL FILE SYSTEMS

File systems form an intermediate layer between block-oriented hard disks and applications, with a volume manager often being used between the file system and the hard disk

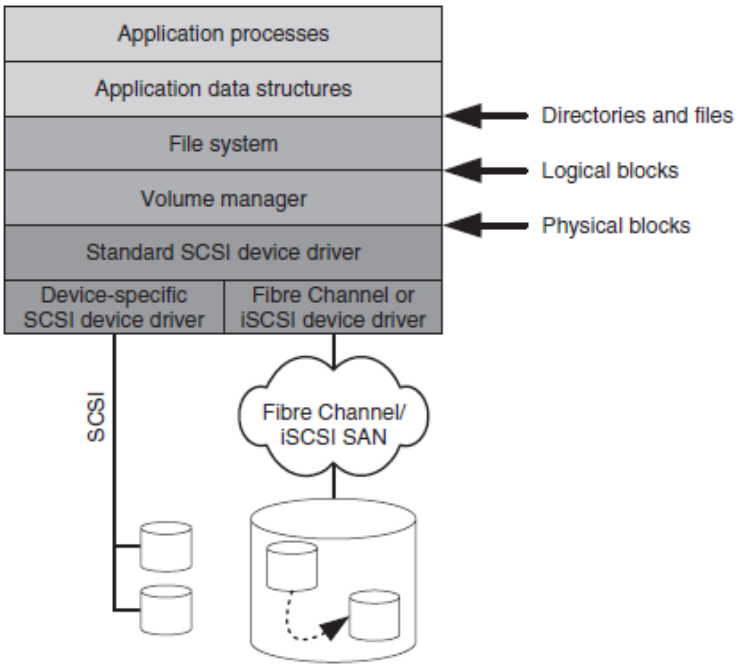

Figure 5.1 File system and volume manager manage the blocks of the block-oriented hard disks. Applications and users thus use the storage capacity of the disks via directories and files. (Figure 4.1). Together, these manage the blocks of the disk and make these available to users and applications via the familiar directories and files.

5.1.1 File systems and databases

File systems and volume manager provide their services to numerous applications with various load profiles. This means that they are generic applications; their performance is not generally optimised for a specific application. Database systems such as DB2 or Oracle can get around the file system and manage the blocks of the hard disk themselves (Figure 4.2). As a result, although the performance of the database can be increased, the management of the database is more difficult. In practice, therefore, database systems are usually configured to store their data in files that are managed by a file system. If more performance is required for a specific database, database administrators generally prefer to pay for higher performance hardware than to reconfigure the database to store its data directly upon the block-oriented hard disks.

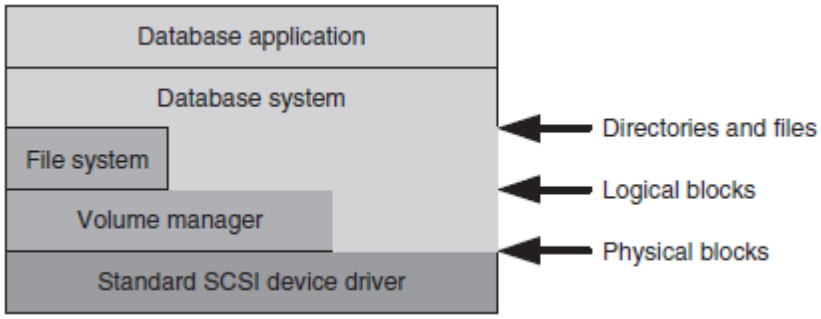

Figure 5.2 To increase performance, databases can get around the file system and manage the blocks themselves.

5.1.2 Journaling

In addition to the basic services, modern file systems provide three functions – journaling, snapshots and dynamic file system expansion. Journaling is a mechanism that guarantees the consistency of the file system even after a system crash. To this end, the file system first of all writes every change to a log file that is invisible to applications and end users, before making

the change in the file system itself. After a system crash the file system only has to run through the end of the log file in order to recreate the consistency of the file system.

In file systems without journaling, typically older file systems like Microsoft's FAT32 file system or the Unix File System (UFS) that is widespread in Unix systems, the consistency of the entire file system has to be checked after a system crash (file system check); in large file systems this can take several hours. In file systems without journaling it can therefore take several hours after a system crash – depending upon the size of the file system – before the data and thus the applications are back in operation.

5.1.3 Snapshots

Snapshots represent the same function as the instant copies function that is familiar from disk subsystems (Section 2.7.1). Snapshots freeze the state of a file system at a given point in time. Applications and end users can access the frozen copy via a special path. As is the case for instant copies, the creation of the copy only takes a few seconds. Likewise, when creating a snapshot, care should be taken to ensure that the state of the frozen data is consistent.

Table 4.1 compares instant copies and snapshots. An important advantage of snapshots is that they can be realised with any hardware. On the other hand, instant copies within a disk subsystem place less load on the CPU and the buses of the server, thus leaving more system resources for the actual applications.

5.1.4 Volume manager

The volume manager is an intermediate layer within the operating system between the file system or database and the actual hard disks. The most important basic function of the volume manager is to aggregate several hard disks to form a large virtual hard disk and make just this virtual hard disk visible to higher layers. Most volume managers provide the option of breaking this virtual disk back down into several smaller virtual hard disks and enlarging or reducing these (Figure **5**.3). This virtualisation within the volume manager makes it possible for system administrators to quickly react to changed

storage requirements of applications such as databases and file systems.

The volume manager can, depending upon its implementation, provide the same functions as a RAID controller (Section 2.4) or an intelligent disk subsystem (Section 2.7). As in snapshots,

here too functions such as RAID, instant copies and remote mirroring are realised in a hardware-independent manner in the volume manager. Likewise, a RAID controller or an intelligent disk subsystem can take the pressure off the resources of the server if the corresponding functions are moved to the storage devices. The realisation of RAID in the volume manager loads not only on the server's CPU, but also on its buses (Figure 4.4).

Source : http://elearningatria.files.wordpress.com/2013/10/cse-viii-storage-area-networks-06cs833-notes.pdf