Syllabus
8th sem
STORAGE AREA NETWORKS

Subject Code 06CS833 **IA Marks 25**
No. of Lecture Hrs./ Week 4 Exam Hours 3
Total No. of Lecture Hrs. 52 Exam Marks 100

PART- A

UNIT - 1
INTRODUCTION: Server Centric IT Architecture and its Limitations; Storage – Centric IT Architecture and its advantages; Case
study: Replacing a server with Storage Networks; The Data Storage and Data Access problem; The Battle for size and access.

6 Hours

UNIT - 2
INTELLIGENT DISK SUBSYSTEMS - 1: Architecture of Intelligent Disk Subsystems; Hard disks and Internal I/O Channels, JBOD, Storage virtualization using RAID and different RAID levels; **6 Hours**
UNIT - 3
INTELLIGENT DISK SUBSYSTEMS – 1, I/O TECHNIQUES - 1: Caching: Acceleration of Hard Disk Access; Intelligent disk subsystems; Availability of disk subsystems. The Physical I/O path from the CPU to the Storage System; SCSI. **7 Hours**
UNIT - 4
I/O TECHNIQUES – 2, NETWORK ATTACHED STORAGE: Fiber Channel Protocol Stack; Fiber Channel SAN; IP Storage. The NAS Architecture, The NAS hardware Architecture, The NAS Software Architecture, Network connectivity, NAS as a storage system. **7 Hours**

PART- B

UNIT - 5
FILE SYSTEM AND NAS: Local File Systems; Network file Systems and file servers; Shared Disk file systems; Comparison of fiber Channel and NAS. **6 Hours**
UNIT - 6
STORAGE VIRTUALIZATION: Definition of Storage virtualization; Implementation Considerations; Storage virtualization on Block or file level; Storage virtualization on various levels of the storage Network; Symmetric and Asymmetric storage virtualization in the Network **6 Hours**
UNIT - 7
SAN ARCHITECTURE AND HARDWARE DEVICES: Overview, creating a Network for storage; SAN Hardware devices, The fiber channel switch, Host Bus adapters; Putting the storage in SAN; Fabric operation from a Hardware perspective. **7 Hours**
UNIT - 8
SOFTWARE COMPONENTS OF SAN: The switch's Operating system, Device Drivers, The Supporting the switch's components, Configuration options for SANs. Panning for business continuity. **7 Hours**

TEXT BOOKS:

1. **Storage Networks Explained** – Ulf Troppens, Rainer Erkens and Wolfgang Muller, John Wiley & Sons, 2003.
2. **Storage Networks: The Complete Reference** – Robert Spalding, Tata McGraw Hill, 2003.

REFERENCE BOOKS:

1. **Storage Area Network Essentials: A Complete Guide to understanding and Implementing SANs** – Richard Barker
and Paul Massiglia, John Wiley India, 2002.
2. **Storage Networking Fundamentals** Marc Farley, Cisco Press, 2005.

INDEX SHEET

UNIT 1

Introduction

1.1 SERVER-CENTRIC IT ARCHITECTURE AND ITS LIMITATIONS

In conventional IT architectures, storage devices are normally only connected to a single server (Figure 1.1). To increase fault tolerance, storage devices are sometimes connected to two servers, with only one server actually able to use the storage device at any one time. In both cases, the storage device exists only in relation to the server to which it is connected. Other servers cannot directly access the data; they always have to go through the server that is connected to the storage device. This conventional IT architecture is therefore called server-centric IT architecture. In this approach, servers and storage devices are generally connected together by SCSI cables.

As mentioned above, in conventional server-centric IT architecture storage devices exist only in relation to the one or two servers to which they are connected. The failure of both of these computers would make it impossible to access this data. Most companies find this unacceptable: at least some of the company data (for example, patient files, websites) must be available around the clock.

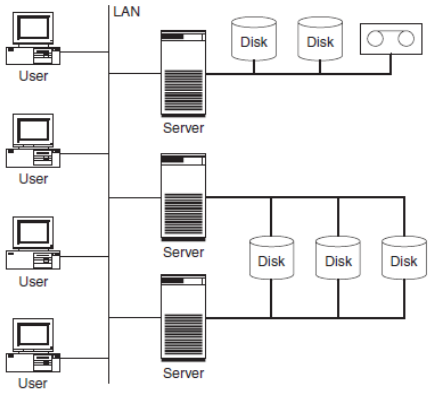


Figure 1.1 In a server-centric IT architecture storage devices exist only in relation to servers.

Although the storage density of hard disks and tapes is increasing all the time due to ongoing technical development, the need for installed storage is increasing even faster. Consequently, it

is necessary to connect ever more storage devices to a computer. This throws up the problem that each computer can accommodate only a limited number of I/O cards (for example, SCSI cards). Furthermore, the length of SCSI cables is limited to a maximum of 25 m. This means that the storage capacity that can be connected to a computer using conventional technologies is limited. Conventional technologies are therefore no longer sufficient to satisfy the growing demand for storage capacity. In server-centric IT environments the storage device is statically assigned to the computer to which it is connected. In general, a computer cannot access storage devices that are connected to a different computer. This means that if a computer requires more storage space than is connected to it, it is no help whatsoever that another computer still has attached storage space, which is not currently used (Figure 1.2).

Last, but not least, storage devices are often scattered throughout an entire building or branch. Sometimes this is because new computers are set up all over the campus without any great consideration and then upgraded repeatedly. Alternatively, computers may be consciously set up where the user accesses the data in order to reduce LAN data traffic. The result is that the storage devices are distributed throughout many rooms, which are neither protected against unauthorised access nor sufficiently air-conditioned. This may sound over the top, but many system administrators could write a book about replacing defective hard disks that are scattered all over the country.

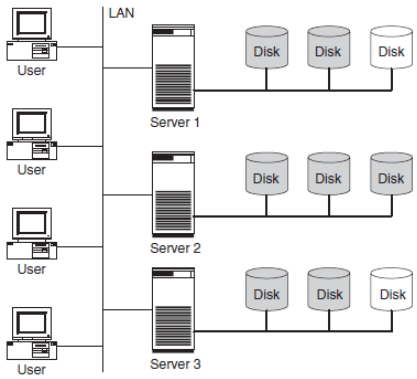


Figure 1.2 The storage capacity on server 2 is full. It cannot make use of the fact that there is still storage space free on server 1 and server 3.

1.2 STORAGE-CENTRIC IT ARCHITECTURE AND ITS ADVANTAGES

Storage networks can solve the problems of server-centric IT architecture that we have just discussed. Furthermore, storage networks open up new possibilities for data management. The idea behind storage networks is that the SCSI cable is replaced by a network that is installed in addition to the existing LAN and is primarily used for data exchange between computers and storage devices (Figure 1.3).

In contrast to server-centric IT architecture, in storage networks storage devices exist completely independently of any computer. Several servers can access the same storage device directly over the storage network without another server having to be involved. Storage devices are thus placed at the centre of the IT architecture; servers, on the other hand, become an appendage of the storage devices that 'just process data'. IT architectures with storage networks are therefore known as storage-centric IT architectures.

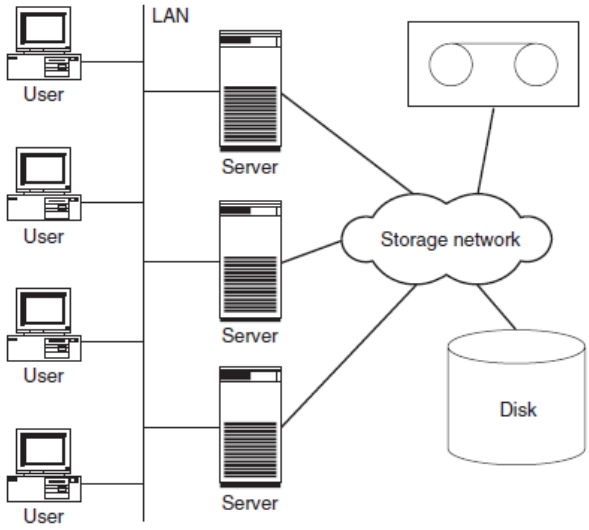


Figure 1.3 In storage-centric IT architecture the SCSI cables are replaced by a network Storage devices now exist independently of a server.

When a storage network is introduced, the storage devices are usually also consolidated. This involves replacing the many small hard disks attached to the computers with a large disk subsystem. Disk subsystems currently (in the year 2009) have a maximum storage capacity of up to a petabyte. The storage network permits all computers to access the disk subsystem and share it. Free storage capacity can thus be flexibly assigned to the computer that needs it at the time. In the same manner, many small tape libraries can be replaced by one big one.

More and more companies are converting their IT systems to a storage-centric IT architecture. It has now become a permanent component of large data centres and the IT systems of large companies. In our experience, more and more medium-sized companies and public institutions are now considering storage networks. Even today, most storage capacity is no longer fitted into the case of a server (internal storage device), but has its own case (external storage device).

1.3 CASE STUDY: REPLACING A SERVER WITH STORAGE NETWORKS

In the following we will illustrate some advantages of storage-centric IT architecture using a case study: in a production environment an application server is no longer powerful enough. The ageing computer must be replaced by a higher-performance device. Whereas such a measure can be very complicated in a conventional, server-centric IT architecture, it can be carried out very elegantly in a storage network.

1. Before the exchange, the old computer is connected to a storage device via the storage network, which it uses partially (Figure 1.4 shows stages 1, 2 and 3).

2. First, the necessary application software is installed on the new computer. The new computer is then set up at the location at which it will ultimately stand. With storage networks it is possible to set up the computer and storage device several kilometers apart.

3. Next, the production data for generating test data within the disk subsystem is copied. Modern storage systems can (practically) copy even terabyte-sized data files within seconds. This function is called instant copy. To copy data it is often necessary to shut down the applications, so that the copied data is in a consistent state. Consistency is necessary to permit the application to resume operation with the data. Some applications are also capable of keeping a consistent state on the disk during operation (online backup mode of database systems, snapshots of file systems).

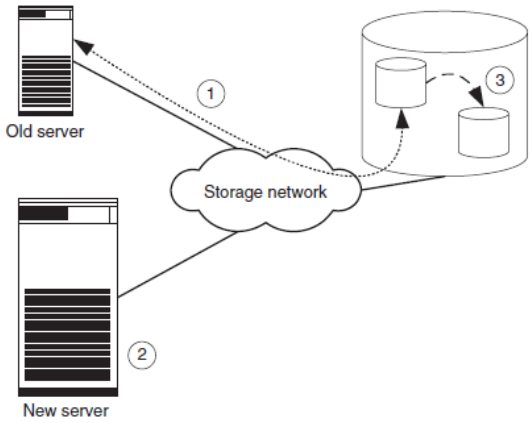


Figure 1.4 The old server is connected to a storage device via a storage network (1). The new server is assembled and connected to the storage network (2). To generate test data the production data is copied within the storage device (3).

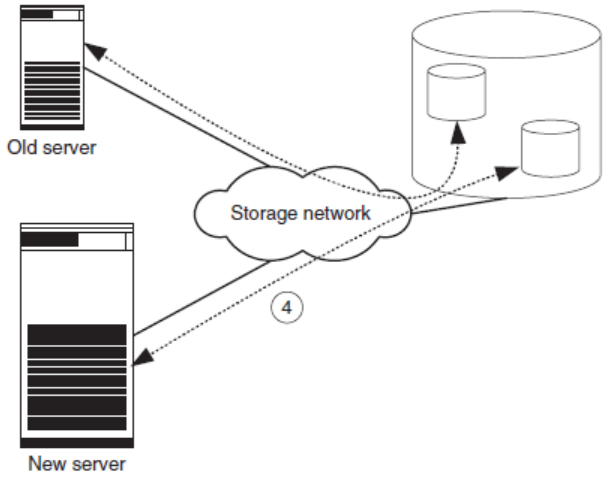


Figure 1.5 Old server and new server share the storage system. The new server is intensively tested using the copied production data (4).

4. Then the copied data is assigned to the new computer and the new computer is tested intensively (Figure 1.5). If the storage system is placed under such an extreme load by the tests that its performance is no longer sufficient for the actual application, the data must first be transferred to a second storage system by means of remote mirroring.

5. After successful testing, both computers are shut down and the production data assigned to the new server. The assignment of the production data to the new server also takes just a few seconds (Figure 1.6 shows steps 5 and 6).

6. Finally, the new server is restarted with the production data.

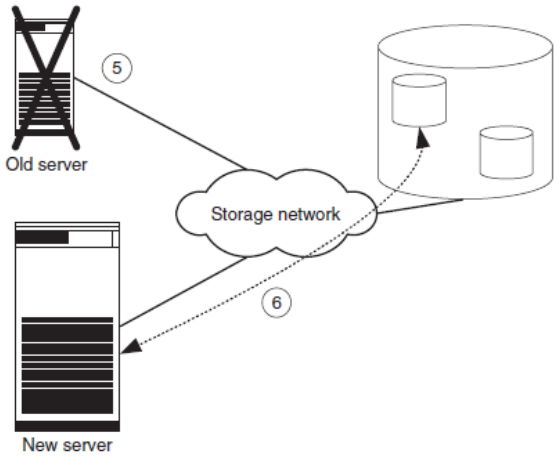


Figure 1.6 Finally, the old server is powered down (5) and the new server is started up with the production data (6).

UNIT 2

Intelligent Disk Subsystems

Hard disks and tapes are currently the most important media for the storage of data. When storage networks are introduced, the existing small storage devices are replaced by a few large storage systems (storage consolidation). For example, individual hard disks and small disk stacks are replaced by large disk subsystems that can store between a few hundred gigabytes and several ten petabytes of data, depending upon size. Furthermore, they have the advantage that functions such as high availability, high performance, instant copies and remote mirroring are available at a reasonable price even in the field of open systems (Unix, Windows, OS/400, Novell Netware, MacOS). The administration of a few large storage systems is significantly simpler, and thus cheaper, than the administration of many small disk stacks. However, the administrator must plan what he is doing more precisely when working with large disk subsystems. This chapter describes the functions of such modern disk subsystems.

2.1 ARCHITECTURE OF INTELLIGENT DISK SUBSYSTEMS

In contrast to a file server, a disk subsystem can be visualised as a hard disk server. Servers are connected to the connection port of the disk subsystem using standard I/O techniques such as Small Computer System Interface (SCSI), Fibre Channel or Internet SCSI (iSCSI) and can thus use the storage capacity that the disk subsystem provides (Figure 2.1). The internal structure of the disk subsystem is completely hidden from the server, which sees only the hard disks that the disk subsystem provides to the server. The connection ports are extended to the hard disks of the disk subsystem by means of internal I/O channels (Figure 2.2). In most disk subsystems there is a controller between the connection ports and the hard disks. The controller can significantly increase the data availability and data access performance with the aid of a so-called RAID procedure. Furthermore, some controllers realise the copying services instant copy and remote mirroring and further additional services. The controller uses a cache in an attempt to accelerate read and write accesses to the server.

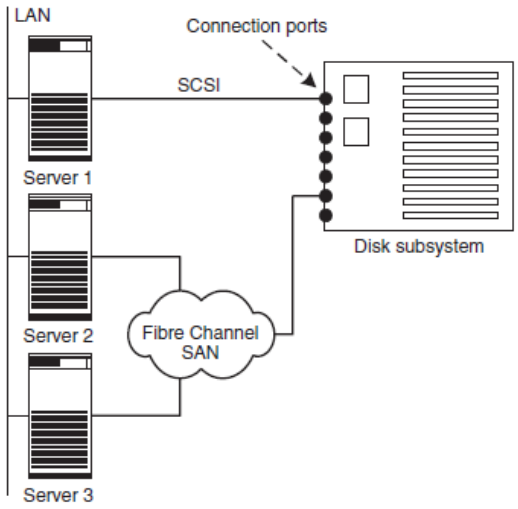


Figure 2.1 Servers are connected to a disk subsystem using standard I/O techniques. The figure shows a server that is connected by SCSI. Two others are connected by Fibre Channel SAN.

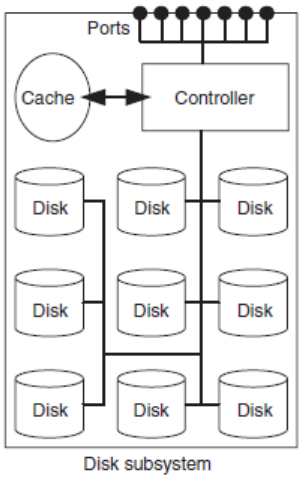


Figure 2.2 Servers are connected to the disk subsystems via the ports. Internally, the disk subsystem consists of hard disks, a controller, a cache and internal I/O channels.

Disk subsystems are available in all sizes. Small disk subsystems have one to two connection ports for servers or storage networks, six to eight hard disks and, depending on the disk capacity, storage capacity of a few terabytes. Large disk subsystems have multiple ten connection ports for servers and storage networks, redundant controllers and multiple I/O channels. A considerably larger number of servers can access a subsystem through a connection over a storage network. Large disk subsystems can store up to a petabyte of data and, depending on the supplier, can weigh well over a tonne. The dimensions of a large disk subsystem are comparable to those of a wardrobe. Figure 2.2 shows a simplified schematic

representation. The architecture of real disk subsystems is more complex and varies greatly. Ultimately, however, it will always include the components shown in Figure 2.2.

Regardless of storage networks, most disk subsystems have the advantage that free disk space can be flexibly assigned to each server connected to the disk subsystem (storage pooling). Figure 2.3 refers back once again to the example of Figure 1.2. In Figure 1.2 it is not possible to assign more storage to server 2, even though free space is available on servers 1 and 3. In Figure 2.3 this is not a problem. All servers are either directly connected to the disk subsystem or indirectly connected via a storage network. In this configuration each server can be assigned free storage. Incidentally, free storage capacity should be understood to mean both hard disks that have already been installed and have not yet been used and also free slots for hard disks that have yet to be installed.

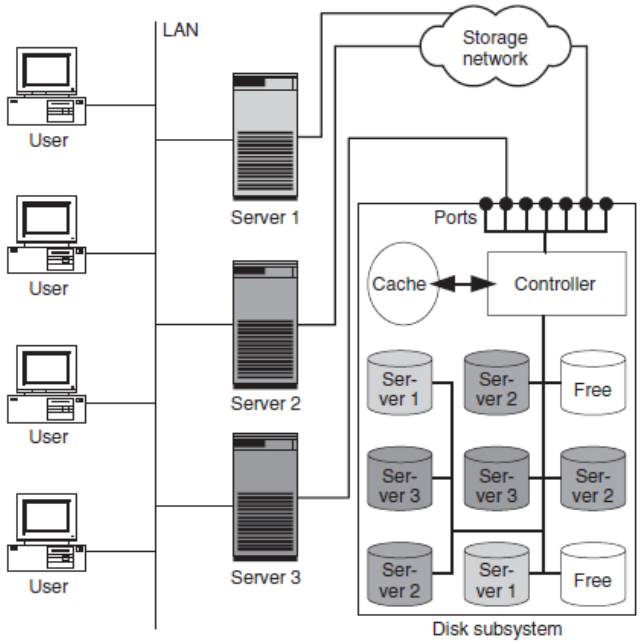


Figure 2.3 All servers share the storage capacity of a disk subsystem. Each server can be assigned free storage more flexibly as required.

2.2 HARD DISKS AND INTERNAL I/O CHANNELS

The controller of the disk subsystem must ultimately store all data on physical hard disks. Standard hard disks that range in size from 36GB to 1 TB are currently (2009) used for this

purpose. Since the maximum number of hard disks that can be used is often limited, the size of the hard disk used gives an indication of the maximum capacity of the overall disk subsystem.

When selecting the size of the internal physical hard disks it is necessary to weigh the requirements of maximum performance against those of the maximum capacity of the overall system. With regard to performance it is often beneficial to use smaller hard disks at the expense of the maximum capacity: given the same capacity, if more hard disks are available in a disk subsystem, the data is distributed over several hard disks and thus the overall load is spread over more arms and read/write heads and usually over more I/O channels (Figure 2.4).

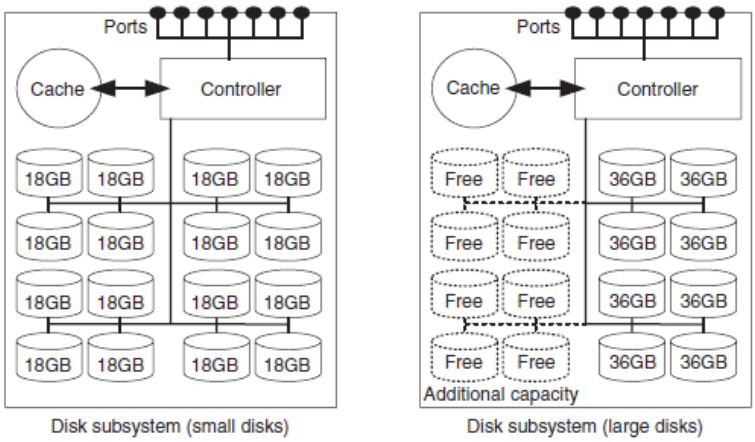


Figure 2.4 If small internal hard disks are used, the load is distributed over more hard disks and thus over more read and write heads. On the other hand, the maximum storage capacity is reduced, since in both disk subsystems only 16 hard disks can be fitted.

For most applications, medium-sized hard disks are sufficient. Only for applications with extremely high performance requirements should smaller hard disks be considered. However, consideration should be given to the fact that more modern, larger hard disks generally have shorter seek times and larger caches, so it is necessary to carefully weigh up which hard disks will offer the highest performance for a certain load profile in each individual case.

Standard I/O techniques such as SCSI, Fibre Channel, increasingly Serial ATA (SATA) and Serial Attached SCSI (SAS) and, still to a degree, Serial Storage Architecture (SSA) are being used for internal I/O channels between connection ports and controller as well as between controller and internal hard disks. Sometimes, however, proprietary – i.e., manufacturer-

specific – I/O techniques are used. Regardless of the I/O technology used, the I/O channels can be designed with built-in redundancy in order to increase the fault-tolerance of a disk subsystem.

The following cases can be differentiated here:

• *Active*

In active cabling the individual physical hard disks are only connected via one I/O channel (Figure 2.5, left). If this access path fails, then it is no longer possible to access the data.

• *Active/passive*

In active/passive cabling the individual hard disks are connected via two I/O channels (Figure 2.5, right). In normal operation the controller communicates with the hard disks via the first I/O channel and the second I/O channel is not used. In the event of the failure of the first I/O channel, the disk subsystem switches from the first to the second I/O channel.

• *Active/active (no load sharing)*

In this cabling method the controller uses both I/O channels in normal operation (Figure 2.6, left). The hard disks are divided into two groups: in normal operation the first group is addressed via the first I/O channel and the second via the second I/O channel. If one I/O channel fails, both groups are addressed via the other I/O channel.

• *Active/active (load sharing)*

In this approach all hard disks are addressed via both I/O channels in normal operation (Figure 2.6, right). The controller divides the load dynamically between the two I/O channels so that the available hardware can be optimally utilised. If one I/O channel fails, then the communication goes through the other channel only. Active cabling is the simplest and thus also the cheapest to realise but offers no protection against failure. Active/passive cabling is the minimum needed to protect against failure, whereas active/active cabling with load sharing best utilises the underlying hardware.

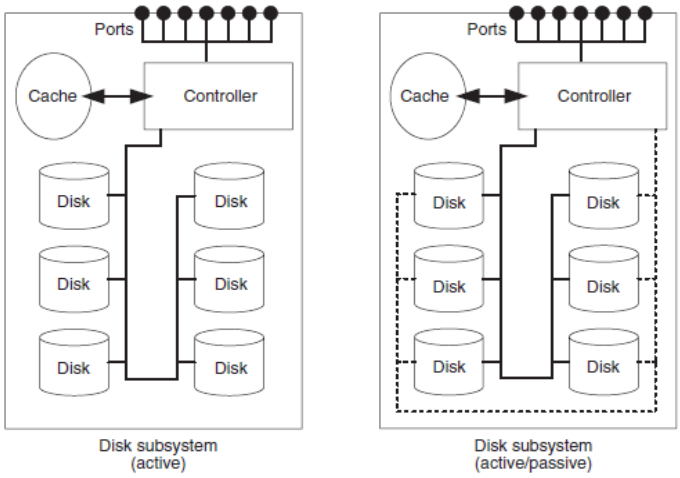


Figure 2.5 In active cabling all hard disks are connected by just one I/O channel. In active/passive cabling all hard disks are additionally connected by a second I/O channel. If the primary I/O channel fails, the disk subsystem switches to the second I/O channel.

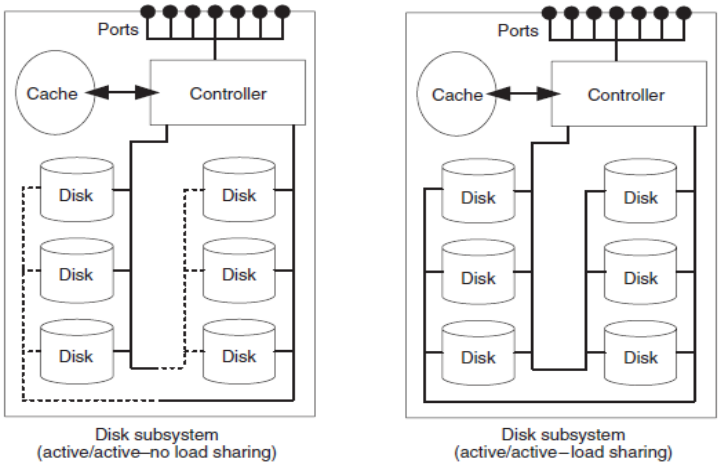


Figure 2.6 Active/active cabling (no load sharing) uses both I/O channels at the same time. However, each disk is addressed via one I/O channel only, switching to the other channel in the event of a fault. In active/active cabling (load sharing) hard disks are addressed via both I/O channels.

2.3 JBOD: JUST A BUNCH OF DISKS

If we compare disk subsystems with regard to their controllers we can differentiate between three levels of complexity: (1) no controller; (2) RAID controller (Sections 2.4 and 2.5); and (3) intelligent controller with additional services such as instant copy and remote mirroring (Section 2.7).

If the disk subsystem has no internal controller, it is only an enclosure full of disks (JBODs). In this instance, the hard disks are permanently fitted into the enclosure and the connections for I/O channels and power supply are taken outwards at a single point. Therefore, a JBOD is

simpler to manage than a few loose hard disks. Typical JBOD disk subsystems have space for 8 or 16 hard disks. A connected server recognises all these hard disks as independent disks. Therefore, 16 device addresses are required for a JBOD disk subsystem incorporating 16 hard disks. In some I/O techniques such as SCSI (Section 3.2) and Fibre Channel arbitrated loop (Section 3.3.6), this can lead to a bottleneck at device addresses. In contrast to intelligent disk subsystems, a JBOD disk subsystem in particular is not capable of supporting RAID or other forms of virtualisation. If required, however, these can be realised outside the JBOD disk subsystem, for example, as software in the server (Section 5.1) or as an independent virtualisation entity in the storage network (Section 5.6.3).

2.4 STORAGE VIRTUALISATION USING RAID

A disk subsystem with a RAID controller offers greater functional scope than a JBOD disk subsystem. RAID was originally developed at a time when hard disks were still very expensive and less reliable than they are today. RAID was originally called ‘Redundant Array of Inexpensive Disks’. Today RAID stands for ‘Redundant Array of Independent Disks’. Disk subsystems that support RAID are sometimes also called RAID arrays. RAID has two main goals: to increase performance by striping and to increase fault-tolerance by redundancy. Striping distributes the data over several hard disks and thus distributes the load over more hardware. Redundancy means that additional information is stored so that the operation of the application itself can continue in the event of the failure of a hard disk. You cannot increase the performance of an individual hard disk any more than you can improve its fault-tolerance. Individual physical hard disks are slow and have a limited life-cycle. However, through a suitable combination of physical hard disks it is possible to significantly increase the fault-tolerance and performance of the system as a whole.

The bundle of physical hard disks brought together by the RAID controller are also known as virtual hard disks. A server that is connected to a RAID system sees only the virtual hard disk; the fact that the RAID controller actually distributes the data over several physical hard disks is completely hidden to the server (Figure 2.7). This is only visible to the administrator from outside.

A RAID controller can distribute the data that a server writes to the virtual hard disk amongst the individual physical hard disks in various manners. These different procedures are known as RAID levels. Section 2.5 explains various RAID levels in detail.

One factor common to almost all RAID levels is that they store redundant information. If a physical hard disk fails, its data can be reconstructed from the hard disks that remain intact. The defective hard disk can even be replaced by a new one during operation if a disk subsystem has the appropriate hardware. Then the RAID controller reconstructs the data of the exchanged hard disk. This process remains hidden to the server apart from a possible reduction in performance: the server can continue to work uninterrupted on the virtual hard disk.

Modern RAID controllers initiate this process automatically. This requires the definition of so-called hot spare disks (Figure 2.8). The hot spare disks are not used in normal operation. If a disk fails, the RAID controller immediately begins to copy the data of the remaining intact disk onto a hot spare disk. After the replacement of the defective disk, this is included in the pool of hot spare disks. Modern RAID controllers can manage a common pool of hot spare disks for several virtual RAID disks. Hot spare disks can be defined for all RAID levels that offer redundancy.

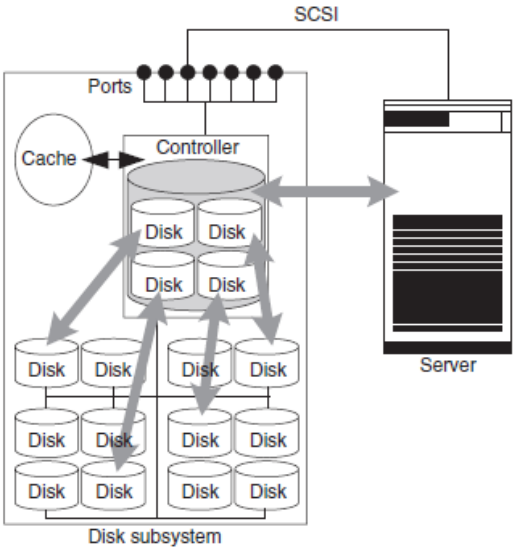


Figure 2.7 The RAID controller combines several physical hard disks to create a virtual hard disk. The server sees only a single virtual hard disk. The controller hides the assignment of the virtual hard disk to the individual physical hard disks.

The recreation of the data from a defective hard disk takes place at the same time as write and read operations of the server to the virtual hard disk, so that from the point of view of the server, performance reductions at least can be observed. Modern hard disks come with self-diagnosis programs that report an increase in write and read errors to the system administrator in plenty of time: 'Caution! I am about to depart this life. Please replace me with a new disk. Thank you!' To this end, the individual hard disks store the data with a redundant code such as the Hamming code. The Hamming code permits the correct recreation of the data, even if individual bits are changed on the hard disk. If the system is looked after properly you can assume that the installed physical hard disks will hold out for a while. Therefore, for the benefit of higher performance, it is generally an acceptable risk to give access by the server a higher priority than the recreation of the data of an exchanged physical hard disk.

A further side-effect of the bringing together of several physical hard disks to form a virtual hard disk is the higher capacity of the virtual hard disks. As a result, less device addresses are used up in the I/O channel and thus the administration of the server is also simplified, because less hard disks (drive letters or volumes) need to be used.

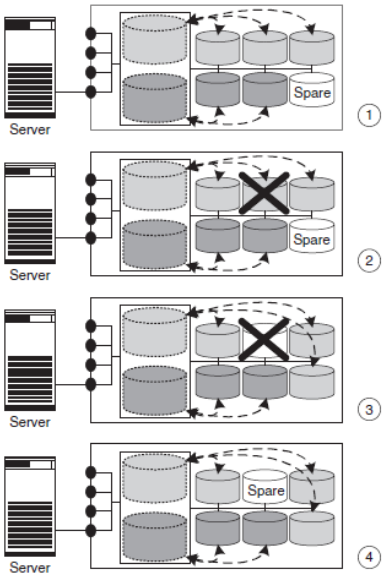


Figure 2.8 Hot spare disk: The disk subsystem provides the server with two virtual disks for which a common hot spare disk is available (1). Due to the redundant data storage the server can continue to process data even though a physical disk has failed, at the expense of a reduction in performance (2). The RAID controller recreates the data from the defective disk on the hot spare disk (3). After the defective disk has been replaced a hot spare disk is once again available (4).

2.5 DIFFERENT RAID LEVELS IN DETAIL

RAID has developed since its original definition in 1987. Due to technical progress some RAID levels are now practically meaningless, whilst others have been modified or added at a later date. This section introduces the RAID levels that are currently the most significant in practice. We will not introduce RAID levels that represent manufacturer- pecific variants and variants that only deviate slightly from the basic forms mentioned in the following.

2.5.1 RAID 0: block-by-block striping

RAID 0 distributes the data that the server writes to the virtual hard disk onto one physical hard disk after another block-by-block (block-by-block striping). Figure 2.9 hows a RAID array with four physical hard disks. In Figure 2.9 the server writes the blocks A, B, C, D, E, etc. onto the virtual hard disk one after the other. The RAID controller distributes the sequence of blocks onto the individual physical hard disks: it writes the first block, A, to the first physical hard disk, the second block, B, to the second physical hard disk, block C to the third and block D to the fourth. Then it begins to write to the first physical hard disk once again, writing block E to the first disk, block F to the second, and so on.

RAID 0 increases the performance of the virtual hard disk as follows: the individual hard disks can exchange data with the RAID controller via the I/O channel significantly more quickly than they can write to or read from the rotating disk. In Figure 2.9 the RAID controller sends the first block, block A, to the first hard disk. This takes some time to write the block to the disk. Whilst the first disk is writing the first block to the physical hard disk, the RAID controller is already sending the second block, block B, to the second hard disk and block C to the third hard disk. In the meantime, the first two physical hard disks are still engaged in depositing their respective blocks onto the physical hard disk. If the RAID controller now sends block E to the first hard disk, then this has written block A at least partially, if not entirely, to the physical hard disk.

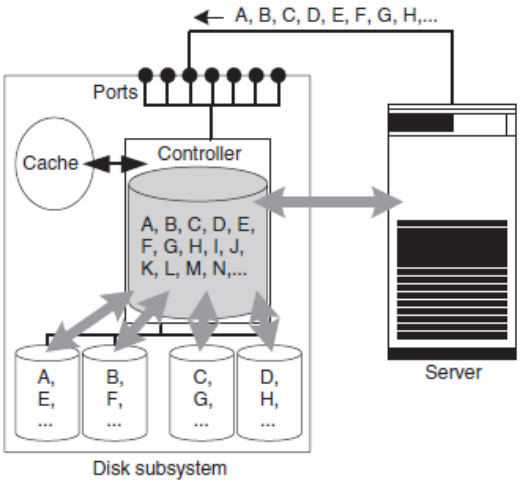


Figure 2.9 RAID 0 (striping): As in all RAID levels, the server sees only the virtual hard disk. The RAID controller distributes the write operations of the server amongst several physical hard disks. Parallel writing means that the performance of the virtual hard disk is higher than that of the individual physical hard disks.

In the example, it was possible to increase the throughput fourfold in 2002: Individual hard disks were able to achieve a throughput of around 50MB/s. The four physical hard disks achieve a total throughput of around 4 × 50MB/s _ 200MB/s. In those days I/O techniques such as SCSI or Fibre Channel achieve a throughput of 160MB/s or 200MB/s. If the RAID array consisted of just three physical hard disks the total throughput of the hard disks would be the limiting factor. If, on the other hand, the RAID array consisted of five physical hard disks the I/O path would be the limiting factor. With five or more hard disks, therefore, performance increases are only possible if the hard disks are connected to different I/O paths so that the load can be striped not only over several physical hard disks, but also over several I/O paths.

RAID 0 increases the performance of the virtual hard disk, but not its fault-tolerance. If a physical hard disk is lost, all the data on the virtual hard disk is lost. To be precise, therefore, the 'R' for 'Redundant' in RAID is incorrect in the case of RAID 0, with 'RAID 0' standing instead for 'zero redundancy'.

2.5.2 RAID 1: block-by-block mirroring

In contrast to RAID 0, in RAID 1 fault-tolerance is of primary importance. The basic form of RAID 1 brings together two physical hard disks to form a virtual hard disk by mirroring the

data on the two physical hard disks. If the server writes a block to the virtual hard disk, the RAID controller writes this block to both physical hard disks (Figure 2.10).

The individual copies are also called mirrors. Normally, two or sometimes three copies of the data are kept (three-way mirror). In a normal operation with pure RAID 1, performance increases are only possible in read operations. After all, when reading the data the load can be divided between the two disks. However, this gain is very low in comparison to RAID 0. When writing with RAID 1 it tends to be the case that reductions in performance may even have to be taken into account. This is because the RAID controller has to send the data to both hard disks. This disadvantage can be disregarded for an individual write operation, since the capacity of the I/O channel is significantly higher than the maximum write speed of the two hard disks put together. However, the I/O channel is under twice the load, which hinders other data traffic using the I/O channel at the same time.

2.5.3 RAID 0+1/RAID 10: striping and mirroring combined

The problem with RAID 0 and RAID 1 is that they increase either performance (RAID 0) or fault-tolerance (RAID 1). However, it would be nice to have both performance and fault-tolerance. This is where RAID 0+1 and RAID 10 come into play. These two RAID levels combine the ideas of RAID 0 and RAID 1

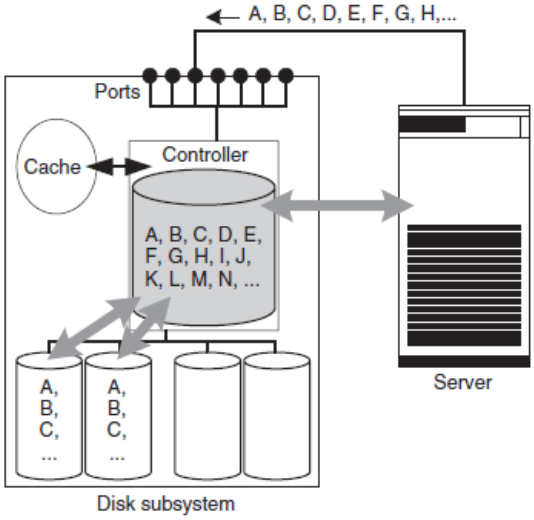


Figure 2.10 RAID 1 (mirroring): As in all RAID levels, the server sees only the virtual hard disk. The RAID controller duplicates each of the server's write operations onto two physical hard disks. After the failure of one physical hard disk the data can still be read from the other disk.

RAID 0+1 and RAID 10 each represent a two-stage virtualisation hierarchy. Figure 2.11 shows the principle behind RAID 0+1 (mirrored stripes). In the example, eight physical hard disks are used. The RAID controller initially brings together each four physical hard disks to form a total of two virtual hard disks that are only visible within the RAID controller by means of RAID 0 (striping). In the second level, it consolidates these two virtual hard disks into a single virtual hard disk by means of RAID 1 (mirroring); only this virtual hard disk is visible to the server.

In RAID 10 (striped mirrors) the sequence of RAID 0 (striping) and RAID 1 (mirroring) is reversed in relation to RAID 0+1 (mirrored stripes). Figure 2.12 shows the principle underlying RAID 10 based again on eight physical hard disks. In RAID 10 the RAID controller initially brings together the physical hard disks in pairs by means of RAID 1 (mirroring) to form a total of four virtual hard disks that are only visible within the RAID controller. In the second stage, the RAID controller consolidates these four virtual hard disks into a virtual hard disk by means of RAID 0 (striping). Here too, only this last virtual hard disk is visible to the server. In both RAID 0+1 and RAID 10 the server sees nly a single hard disk, which is larger, faster and more fault-tolerant than a physical hard disk. We now have to ask the question: which of the two RAID levels, RAID 0+1 or RAID 10, is preferable?

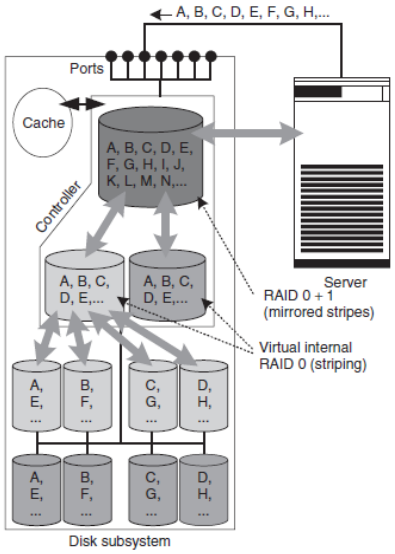


Figure 2.11 RAID 0+1 (mirrored stripes): As in all RAID levels, the server sees only the virtual hard disk. Internally, the RAID controller realises the virtual disk in two stages: in the first stage it brings together every four physical hard disks into one virtual hard disk that is only visible within the RAID controller by means of RAID 0 (striping); in the second stage it consolidates these two virtual hard disks by means of RAID 1 (mirroring) to form the hard disk that is visible to the server.

The question can be answered by considering that when using RAID 0 the failure of a hard disk leads to the loss of the entire virtual hard disk. In the example relating to RAID 0+1

(Figure 2.11) the failure of a physical hard disk is thus equivalent to the effective failure of four physical hard disks (Figure 2.13). If one of the other four physical hard disks is lost, then the data is lost. In principle it is sometimes possible to reconstruct the data from the remaining disks, but the RAID controllers available on the market cannot do this particularly well.

In the case of RAID 10, on the other hand, after the failure of an individual physical hard disk, the additional failure of a further physical hard disk – with the exception of the corresponding mirror – can be withstood (Figure 2.14). RAID 10 thus has a significantly higher fault-tolerance than RAID 0+1. In addition, the cost of restoring the RAID system after the failure of a hard disk is much lower in the case of RAID 10 than RAID 0+1. In RAID 10 only one physical hard disk has to be recreated. In RAID 0+1, on the other and, a virtual hard disk must be recreated that is made up of four physical disks. However, the cost of recreating the defective hard disk can be significantly reduced because a physical hard disk is exchanged as a preventative measure when the number of read errors start to increase. In this case it is sufficient to copy the data from the old disk to the new.

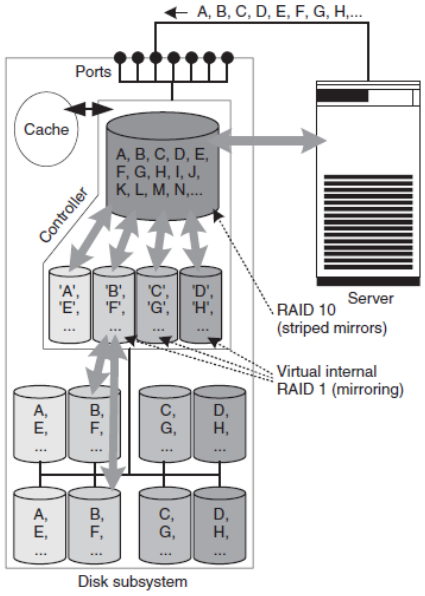


Figure 2.12 RAID 10 (striped mirrors): As in all RAID levels, the server sees only the virtual hard disk. Here too, we proceed in two stages. The sequence of striping and mirroring is reversed in relation to RAID 0+1. In the first stage the controller links every two physical hard disks by means of RAID 1 (mirroring) to a virtual hard disk, which it unifies by means of RAID 0 (striping) in the second stage to form the hard disk that is visible to the server.

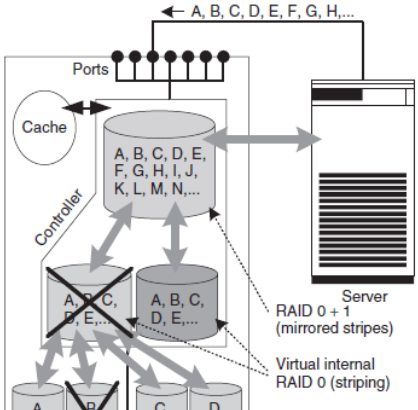


Figure 2.13 The consequences of the failure of a physical hard disk in RAID 0+1 (mirrored stripes) are relatively high in comparison to RAID 10 (striped mirrors). The failure of a physical hard disk brings about the failure of the corresponding internal RAID 0 disk, so that in effect half of the physical hard disks have failed. The recovery of the data from the failed disk is expensive.

However, things look different if the performance of RAID 0+1 is compared with the performance of RAID 10. In Section 5.1 we discuss a case study in which the use of RAID 0+1 is advantageous. With regard to RAID 0+1 and RAID 10 it should be borne in mind that the two RAID procedures are often confused. Therefore the answer 'We use RAID 10!' or 'We use RAID 0+1' does not always provide the necessary clarity. In discussions it is better to ask if mirroring takes place first and the mirror is then striped or if striping takes place first and the stripes are then mirrored.

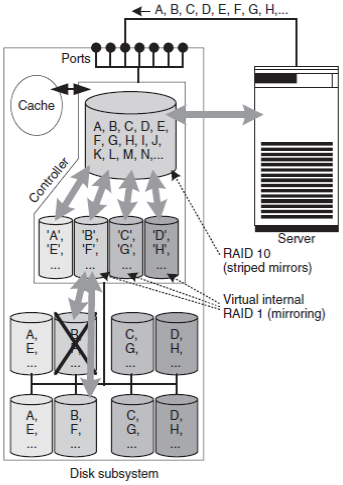


Figure 2.14 In RAID 10 (striped mirrors) the consequences of the failure of a physical hard disk are not as serious as in RAID 0+1 (mirrored stripes). All virtual hard disks remain intact. The recovery of the data from the failed hard disk is simple.

2.5.4 RAID 4 and RAID 5: parity instead of mirroring

RAID 10 provides excellent performance at a high level of fault-tolerance. The problem with this is that mirroring using RAID 1 means that all data is written to the physical hard disk

twice. RAID 10 thus doubles the required storage capacity. The idea of RAID 4 and RAID 5 is to replace all mirror disks of RAID 10 with a single parity hard disk. Figure 2.15 shows the principle of RAID 4 based upon five physical hard disks. The server again writes the blocks A, B, C, D, E, etc. to the virtual hard disk sequentially. The RAID controller stripes the data blocks over the first four physical hard disks. Instead of mirroring all data onto the further four physical hard disks, as in RAID 10, the RAID controller calculates a parity block for every four blocks and writes this onto the fifth physical hard disk.

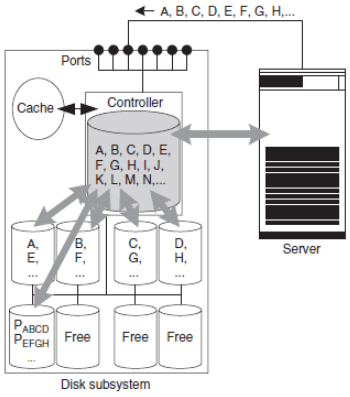


Figure 2.15 RAID 4 (parity disk) is designed to reduce the storage requirement of RAID 0+1 and RAID 10. In the example, the data blocks are distributed over four physical hard disks by means of RAID 0 (striping). Instead of mirroring all data once again, only a parity block is stored for each four blocks.

For example, the RAID controller calculates the parity block PABCD for the blocks A, B, C and D. If one of the four data disks fails, the RAID controller can reconstruct the data of the defective disks using the three other data disks and the parity disk. In comparison to the examples in Figures 2.11 (RAID 0+1) and 2.12 (RAID 10), RAID 4 saves three physical hard disks. As in all other RAID levels, the server again sees only the virtual disk, as if it were a single physical hard disk.

From a mathematical point of view the parity block is calculated with the aid of the logical XOR operator (Exclusive OR). In the example from Figure 2.15, for example, the equation PABCD = A XOR B XOR C XOR D applies. The space saving offered by RAID 4 and RAID 5, which remains to be discussed, comes at a price in relation to RAID 10. Changing a data block changes the value of the associated parity block. This means that each write operation to the virtual hard disk requires (1) the physical writing of the data block, (2) the recalculation of the parity block and (3) the physical writing of the newly calculated parity block. This extra

cost for write operations in RAID 4 and RAID 5 is called the write penalty of RAID 4 or the write penalty of RAID 5.

The cost for the recalculation of the parity block is relatively low due to the mathematical properties of the XOR operator. If the block A is overwritten by block ˜A and _ is the difference between the old and new data block, then _ = A XOR ˜A. The new parity block ˜P can now simply be calculated from the old parity block P and _, i.e. ˜P = P XOR _. Proof of this property can be found in Appendix A. Therefore, if PABCD is the parity block for the data blocks A, B, C and D, then after the data block A has been changed, the new parity block can be calculated without knowing the remaining blocks B, C and D. However, the old block A must be read in before overwriting the physical hard disk in the controller, so that this can calculate the difference _. When processing write commands for RAID 4 and RAID 5 arrays, RAID controllers use the above-mentioned mathematical properties of the XOR operation for the recalculation of the parity block. Figure 2.16 shows a server that changes block D on the virtual hard disk. The RAID controller reads the data block and the associated parity block from the disk in question into its cache. Then it uses the XOR operation to calculate the difference

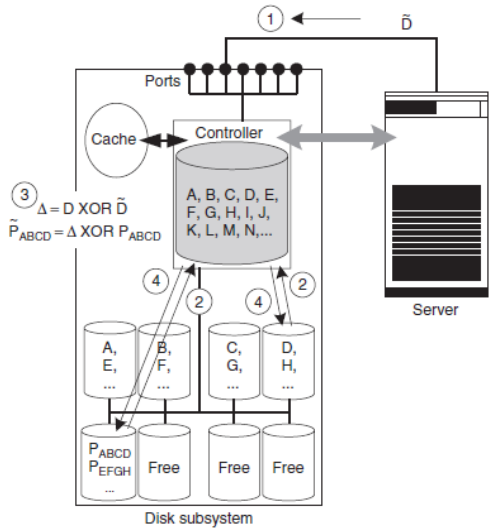


Figure 2.16 Write penalty of RAID 4 and RAID 5: The server writes a changed data block (1). The RAID controller reads in the old data block and the associated old parity block (2) and calculates the new parity block (3). Finally it writes the new data block and the new parity block onto the physical hard disk in question (4).

between the old and the new parity block, i.e. _ = D XOR D˜ , and from this the new parity block ˜PABCD by means of ˜PABCD = PABCD XOR _. Therefore it is not necessary to read in all four associated data blocks to recalculate the parity block. To conclude the write

operation to the virtual hard disk, the RAID controller writes the new data block and the recalculated parity block onto the physical hard disks in question.

Advanced RAID 4 and RAID 5 implementations are capable of reducing the write penalty even further for certain load profiles. For example, if large data quantities are written sequentially, then the RAID controller can calculate the parity blocks from the data flow without reading the old parity block from the disk. If, for example, the blocks E, F, G and H in Figure 2.15 are written in one go, then the controller can calculate the parity block PEFGH from them and overwrite this without having previously read in the old value. Likewise, a RAID controller with a suitably large cache can hold frequently changed parity blocks in the cache after writing to the disk, so that the next time one of the data blocks in question is changed there is no need to read in the parity block. In both cases the I/O load is now lower than in the case of RAID 10. In the example only five physical blocks now need to be written instead of eight as is the case with RAID 10. RAID 4 saves all parity blocks onto a single physical hard disk. For the example in Figure 2.15 this means that the write operations for the data blocks are distributed over four physical hard disks. However, the parity disk has to handle the same number of write operations all on its own. Therefore, the parity disk become the performance bottleneck of RAID 4 if there are a high number of write operations.

To get around this performance bottleneck, RAID 5 distributes the parity blocks over all hard disks. Figure 2.17 illustrates the procedure. As in RAID 4, the RAID controller writes the parity block PABCD for the blocks A, B, C and D onto the fifth physical hard disk. Unlike RAID 4, however, in RAID 5 the parity block PEFGH moves to the fourth physical hard disk for the next four blocks E, F, G, H.

RAID 4 and RAID 5 distribute the data blocks over many physical hard disks. Therefore, the read performance of RAID 4 and RAID 5 is as good as that of RAID 0 and almost as good as that of RAID 10. As discussed, the write performance of RAID 4 and RAID 5 suffers from the write penalty; in RAID 4 there is an additional bottleneck caused by the parity disk. Therefore, RAID 4 is seldom used in practice because RAID 5 accomplishes more than RAID 4 with the same amount of physical resources (see also Section 2.5.6). RAID 4 and RAID 5 can withstand the failure of a physical hard disk. Due to the use of parity blocks, the data on the defective

hard disk can be restored with the help of other hard disks. In contrast to RAID 10, the failure of an individual sector of the remaining physical hard disks always results in data loss. This is compensated for with RAID 6, whereby a second parity hard disk is kept so that data is protected twice (Section 2.5.5). In RAID 4 and RAID 5 the recovery of a defective physical hard disk is significantly more expensive than is the case for RAID 1 and RAID 10. In the latter two RAID levels only the mirror of the defective disk needs to be copied to the replaced disk. In RAID 4 and RAID 5, on the other hand, the RAID controller has to read the data from all disks, use this to recalculate the lost data blocks and parity blocks, and then write these blocks to the replacement disk. As in RAID 0+1 this high cost can be avoided by replacing a physical hard disk as a precaution as soon as the rate of read errors increases. If this is done, it is sufficient to copy the data from the hard disk to be replaced onto the new hard disk.

If the fifth physical hard disk has to be restored in the examples from Figure 2.15 (RAID 4) and Figure 2.17 (RAID 5), the RAID controller must first read the blocks A, B, C and D from the physical hard disks, recalculate the parity block PABCD and then write to the exchanged physical hard disk. If a data block has to be restored, only the calculation rule changes. If, in the example, the third physical hard disk is to be recreated, the controller would first have to read in the blocks A, B, D and PABCD, use these to reconstruct block C and write this to the replaced disk.

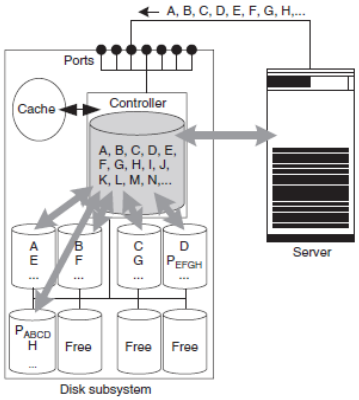


Figure 2.17 RAID 5 (striped parity): In RAID 4 each write access by the server is associated with a write operation to the parity disk for the update of parity information. RAID 5 distributes the load of the parity disk over all physical hard disks.

UNIT 3

3.1 CACHING: ACCELERATION OF HARD DISK ACCESS

In all fields of computer systems, caches are used to speed up slow operations by operating them from the cache. Specifically in the field of disk subsystems, caches are designed to accelerate write and read accesses to physical hard disks. In this connection we can differentiate between two types of cache: (1) cache on the hard disk (Section 2.6.1) and (2) cache in the RAID controller. The cache in the RAID controller is subdivided into write cache (Section 2.6.2) and read cache (Section 2.6.3).

3.1.1 Cache on the hard disk

Each individual hard disk comes with a very small cache. This is necessary because the transfer rate of the I/O channel to the disk controller is significantly higher than the speed at which the disk controller can write to or read from the physical hard disk. If a server or a RAID controller writes a block to a physical hard disk, the disk controller stores this in its cache. The disk controller can thus write the block to the physical hard disk in its own time whilst the I/O channel can be used for data traffic to the other hard disks. Many RAID levels use precisely this state of affairs to increase the performance of the virtual hard disk.

Read access is accelerated in a similar manner. If a server or an intermediate RAID controller wishes to read a block, it sends the address of the requested block to the hard disk controller. The I/O channel can be used for other data traffic while the hard disk controller copies the complete block from the physical hard disk into its cache at a slower data rate. The hard disk controller transfers the block from its cache to the RAID controller or to the server at the higher data rate of the I/O channel.

3.1.2 Write cache in the disk subsystem controller

In addition to the cache of the individual hard drives many disk subsystems come with their own cache, which in some models is gigabytes in size. As a result it can buffer much greater data quantities than the cache on the hard disk. The write cache should have a battery backup and ideally be mirrored. The battery backup is necessary to allow the data in the write cache to survive a power cut. A write cache with battery backup can significantly reduce the write penalty of RAID 4 and RAID 5, particularly for sequential write access (cf. Section 2.5.4

'RAID 4 and RAID 5: parity instead of mirroring'), and smooth out load peaks. Many applications do not write data at a continuous rate, but in batches. If a server sends several data blocks to the disk subsystem, the controller initially buffers all blocks into a write cache with a battery backup and immediately reports back to the server that all data has been securely written to the drive. The disk subsystem then copies the data from the write cache to the slower physical hard disk in order to make space for the next write peak.

3.1.3 Read cache in the disk subsystem controller

The acceleration of read operations is difficult in comparison to the acceleration of write operations using cache. To speed up read access by the server, the disk subsystem's controller must copy the relevant data blocks from the slower physical hard disk to the fast cache before the server requests the data in question. The problem with this is that it is very difficult for the disk subsystem's controller to work out in advance what data the server will ask for next. The controller in the disk subsystem knows neither the structure of the information stored in the data blocks nor the access pattern that an application will follow when accessing the data. Consequently, the controller can only analyse past data access and use this to extrapolate which data blocks the server will access next. In sequential read processes this prediction is comparatively simple, in the case of random access it is almost impossible. As a rule of thumb, good RAID controllers manage to provide around 40% of the requested blocks from the read cache in mixed read profiles. The disk subsystem's controller cannot further increase the ratio of read access provided from the cache (pre-fetch hit rate), because it does not have the necessary application knowledge. Therefore, it is often worthwhile realising a further cache within applications.

For example, after opening a file, file systems can load all blocks of the file into the main memory (RAM); the file system knows the structures that the files are stored in. File systems can thus achieve a pre-fetch hit rate of 100%. However, it is impossible to know whether the expense for the storage of the blocks is worthwhile in an individual case, since the application may not actually request further blocks of the file.

3.2 INTELLIGENT DISK SUBSYSTEMS

Intelligent disk subsystems represent the third level of complexity for controllers after JBODs and RAID arrays. The controllers of intelligent disk subsystems offer additional functions over and above those offered by RAID. In the disk subsystems that are currently available on the market these functions are usually instant copies (Section 2.7.1), remote mirroring (Section 2.7.2) and LUN masking (Section 2.7.3).

Instant copies

Instant copies can virtually copy data sets of several terabytes within a disk subsystem in a few seconds. Virtual copying means that disk subsystems fool the attached servers into believing that they are capable of copying such large data quantities in such a short space of time. The actual copying process takes significantly longer. However, the same server, or a second server, can access the virtually copied data after a few seconds (Figure 2.18).

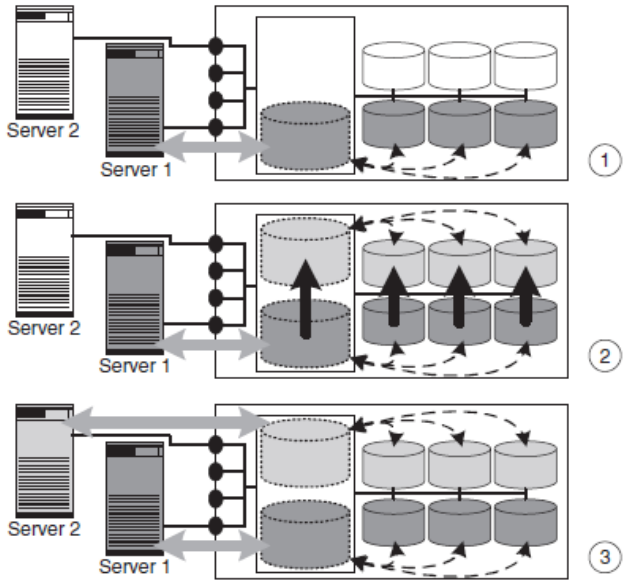


Figure 2.18 Instant copies can virtually copy several terabytes of data within a disk subsystem in a few seconds: server 1 works on the original data (1). The original data is virtually copied in a few seconds (2). Then server 2 can work with the data copy, whilst server 1 continues to operate with the original data (3).

Instant copies are used, for example, for the generation of test data, for the backup of data and for the generation of data copies for data mining. Based upon the case study in Section 1.3 it was shown that when copying data using instant copies, attention should be paid to the

consistency of the copied data. Sections 7.8.4 and 7.10.3 discuss in detail the interaction of applications and storage systems for the generation of consistent instant copies.

There are numerous alternative implementations for instant copies. One thing that all implementations have in common is that the pretence of being able to copy data in a matter of seconds costs resources. All realisations of instant copies require controller computing time and cache and place a load on internal I/O channels and hard disks. The different implementations of instant copy force the performance down at different times. However, it is not possible to choose the most favourable implementation alternative depending upon the application used because real disk subsystems only ever realise one implementation alternative of instant copy.

In the following, two implementation alternatives will be discussed that function in very different ways. At one extreme the data is permanently mirrored (RAID 1 or RAID 10). Upon the copy command both mirrors are separated: the separated mirrors can then be used independently of the original. After the separation of the mirror the production data is no longer protected against the failure of a hard disk. Therefore, to increase data protection, three mirrors are often kept prior to the separation of the mirror (three-way mirror), so that the production data is always mirrored after the separation of the copy. At the other extreme, no data at all is copied prior to the copy command, only after the instant copy has been requested. To achieve this, the controller administers two data areas, one for the original data and one for the data copy generated by means of instant copy. The controller must ensure that during write and read access operations to original data or data copies the blocks in question are written to or read from the data areas in question. In some implementations it is permissible to write to the copy, in some it is not. Some implementations copy just the blocks that have actually changed (partial copy), others copy all blocks as a background process until a complete copy of the original data has been generated (full copy).

In the following, the case differentiations of the controller will be investigated in more detail based upon the example from Figure 2.18. We will first consider access by server 1 to the original data. Read operations are completely unproblematic; they are always served from the area of the original data. Handling write operations is trickier. If a block is changed for the first

time since the generation of the instant copy, the controller must first copy the old block to the data copy area so that server 2 can continue to access the old data set. Only then may it write the changed block to the original data area. If a block that has already been changed in this manner has to be written again, it must be written to the original data area. The controller may not even back up the previous version of the block to the data copy area because otherwise the correct version of the block would be overwritten.

The case differentiations for access by server 2 to the data copy generated by means of instant copy are somewhat simpler. In this case, write operations are unproblematic: the controller always writes all blocks to the data copy area. On the other hand, for read operations it has to distinguish whether the block in question has already been copied or not. This determines whether it has to read the block from the original data area or read
it from the data copy area and forward it to the server.

The subsequent copying of the blocks offers the basis for important variants of instant copy. Space-efficient instant copy only copies the blocks that were changed (Figure 2.19). These normally require considerably less physical storage space than the entire copy. Yet the exported virtual hard disks of the original hard disk and the copy created through space-efficient instant copy are of the same size. From the view of the server both virtual disks continue to have the same size. Therefore, less physical storage space is needed overall and, therefore, the cost of using instant copy can be reduced. Incremental instant copy is another important variant of instant copy. In some situations such a heavy burden is placed on the original data and the copy that the performance within the disk subsystem suffers unless the data has been copied completely onto the copy. An example of this is backup when data is completely backed up through instant copy (Section 7.8.4). On the other side, the background process for copying all the data of an instant copy requires many hours when very large data volumes are involved, making this not a viable alternative. One remedy is incremental instant copy where data is only copied in its entirety the first time around. Afterwards the instant copy is repeated – for example, perhaps daily – whereby only those changes since the previous instant copy are copied. A reversal of instant copy is yet another important variant. If data is backed up through instant copy, then the operation should be continued with the copy if a failure occurs. A simple approach is to shut down the application, copy back the data on the

productive hard disks per a second instant copy from the copy onto the productive hard disks and restart the application. In this case, the disk subsystem must enable a reversal of the instant copy. If this function is not available, if a failure occurs the data either has to be copied back to the productive disks by different means or the operation continues directly with the copy. Both of these approaches are coupled with major copying operations or major configuration changes, and, consequently the recovery takes considerably longer than a reversal of the instant copy.

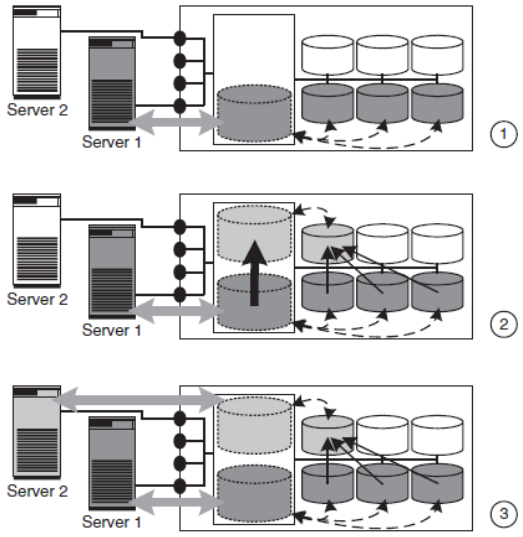


Figure 2.19 Space-efficient instant copy manages with fewer storage systems than the basic form of instant copy (Figure 2.18). Space-efficient instant copy actually only copies the changed blocks before they have been overwritten into the separate area (2). From the view of server 2 the hard disk that has been copied in this way is just as large as the source disk (3). The link between source and copy remains as the changed blocks are worthless on their own.

Remote mirroring

Instant copies are excellently suited for the copying of data sets within disk subsystems. However, they can only be used to a limited degree for data protection. Although data copies generated using instant copy protect against application errors (accidental deletion of a file system) and logical errors (errors in the database program), they do not protect against the failure of a disk subsystem. Something as simple as a power failure can prevent access to production data and data copies for several hours. A fire in the disk subsystem would destroy original data and data copies. For data protection, therefore, the proximity of production data and data copies is fatal.

Remote mirroring offers protection against such catastrophes. Modern disk subsystems can now mirror their data, or part of their data, independently to a second disk subsystem, which is a long way away. The entire remote mirroring operation is handled by the two participating disk subsystems. Remote mirroring is invisible to application servers and does not consume their resources. However, remote mirroring requires resources in the two disk subsystems and in the I/O channel that connects the two disk subsystems together, which means that reductions in performance can sometimes make their way through to the application.

Figure 2.20 shows an application that is designed to achieve high availability using remote mirroring. The application server and the disk subsystem, plus the associated data, are installed in the primary data centre. The disk subsystem independently mirrors the application data onto the second disk subsystem that is installed 50 kilometres away in the backup data centre by means of remote mirroring. Remote mirroring ensures that the application data in the backup data centre is always kept up-to-date with the time interval for updating the second disk subsystem being configurable. If the disk subsystem in the primary data centre fails, the backup application server in the backup data centre can be started up using the data of the second disk subsystem and the operation of the application can be continued. The I/O techniques required for the connection of the two disk subsystems will be discussed in the next chapter.

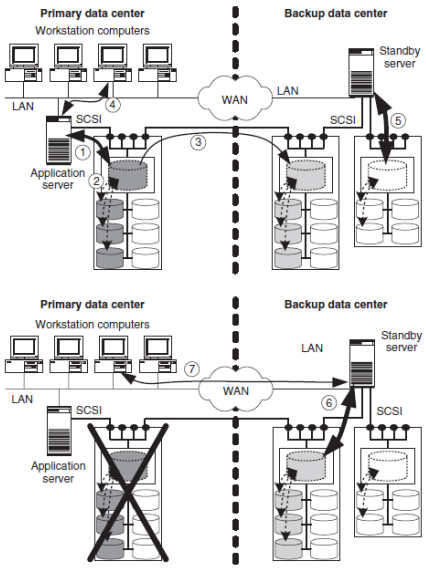


Figure 2.20 High availability with remote mirroring: (1) The application server stores its data on a local disk subsystem. (2) The disk subsystem saves the data to several physical drives by means of RAID. (3) The local disk subsystem uses remote mirroring to mirror the data onto a second disk subsystem located in the backup data centre. (4) Users use the application via the LAN. (5) The stand-by server in the backup data centre is used as a

test system. The test data is located on a further disk subsystem. (6) If the first disk subsystem fails, the application is started up on the stand-by server using the data of the second disk subsystem. (7) Users use the application via the WAN.

We can differentiate between synchronous and asynchronous remote mirroring. In synchronous remote mirroring the first disk subsystem sends the data to the second disk subsystem first before it acknowledges a server's write command. By contrast, asynchronous remote mirroring acknowledges a write command immediately; only then does it send the copy of the block to the second disk subsystem.

Figure 2.21 illustrates the data flow of synchronous remote mirroring. The server writes block A to the first disk subsystem. This stores the block in its write cache and immediately sends it to the second disk subsystem, which also initially stores the block in its write cache. The first disk subsystem waits until the second reports that it has written the block. The question of whether the block is still stored in the write cache of the second disk subsystem or has already been written to the hard disk is irrelevant to the first disk subsystem. It does not acknowledge to the server that the block has been written until it has received confirmation from the second disk subsystem that this has written the block.

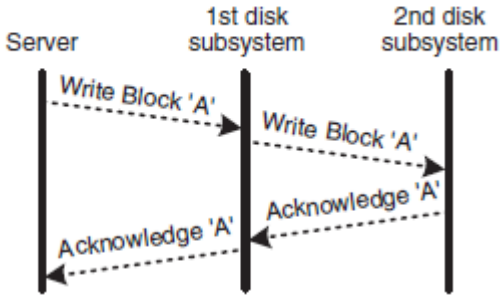


Figure: In asynchronous remote mirroring one disk subsystem acknowledges a write operation as soon as it has saved the block itself. The price of the rapid response time achieved using asynchronous remote mirroring is obvious. In contrast to synchronous remote mirroring, in asynchronous remote mirroring there is no guarantee that the data on the second disk subsystem is up-to-date. This is precisely the case if the first disk subsystem has sent the write acknowledgement to the server but the block has not yet been saved to the second disk subsystem.

If we wish to mirror data over long distances but do not want to use only asynchronous remote mirroring it is necessary to use three disk subsystems (Figure 2.23). The first two may be located just a few kilometres apart, so that synchronous remote mirroring can be used between the two. In addition, the data of the second disk subsystem is mirrored onto a third by means of asynchronous remote mirroring. However, this solution comes at a price: for most applications the cost of data protection would exceed the costs that would be incurred after data loss in the event of a catastrophe. This approach would therefore only be considered for very important applications.

An important aspect of remote mirroring is the duration of the initial copying of the data. With large quantities of data it can take several hours until all data is copied from the first disk subsystem to the second one. This is completely acceptable the first time remote mirroring is established. However, sometimes the connection between both disk subsystems is interrupted later during operations – for example, due to a fault in the network between both systems or during maintenance work on the second disk subsystem.

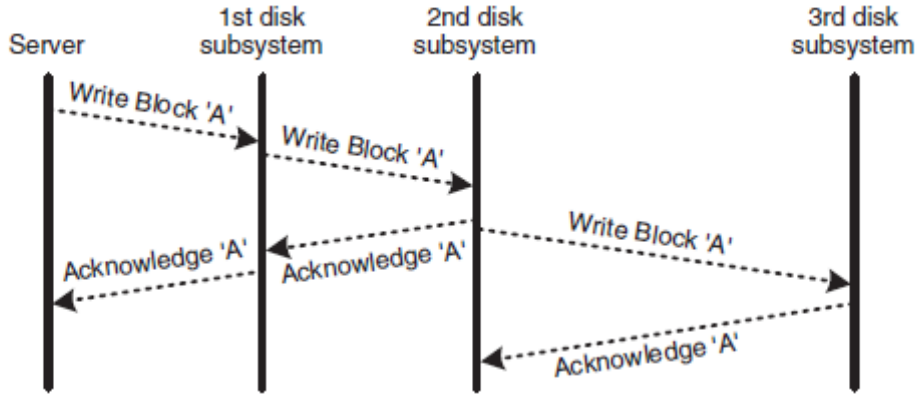


Figure The combination of synchronous and asynchronous remote mirroring means that rapid response times can be achieved in combination with mirroring over long distances.

After the appropriate configuration the application continues operation on the first disk subsystem without the changes having been transferred to the second disk subsystem. Small quantities of data can be transmitted in their entirety again after a fault has been resolved. However, with large quantities of data a mechanism should exist that allows only those blocks that were changed during the fault to be transmitted. This is also referred to as suspending (or freezing) remote mirroring and resuming it later on. Sometimes there is a deliberate reason for

suspending a remote mirroring relationship. Later in the book we will present a business continuity solution that suspends remote mirroring relationships at certain points in time for the purposes of creating consistent copies in backup data centres (Section 9.5.6).

For these same reasons there is a need for a reversal of remote mirroring. In this case, if the first disk subsystem fails, the entire operation is completely switched over to the second disk subsystem and afterwards the data is only changed on that second system. The second disk subsystem logs all changed blocks so that only those blocks that were changed during the failure are transmitted to the first disk subsystem once it is operational again. This ensures that the data on both disk subsystems is synchronised once again.

LUN masking

So-called LUN masking brings us to the third important function – after instant copy and remote mirroring – that intelligent disk subsystems offer over and above that offered by RAID. LUN masking limits the access to the hard disks that the disk subsystem exports to the connected server. A disk subsystem makes the storage capacity of its internal physical hard disks available to servers by permitting access to individual physical hard disks, or to virtual hard disks created using RAID, via the connection ports. Based upon the SCSI protocol, all hard disks – physical and virtual – that are visible outside the disk subsystem are also known as LUN. Without LUN masking every server would see all hard disks that the disk subsystem provides. Figure 2.26 shows a disk subsystem without LUN masking to which three servers are connected. Each server sees all hard disks that the disk subsystem exports outwards. As a result, considerably more hard disks are visible to each server than is necessary.

In particular, on each server those hard disks that are required by applications that run on a different server are visible. This means that the individual servers must be very carefully configured. In Figure 2.26 an erroneous formatting of the disk LUN 3 of server1 would destroy the data of the application that runs on server 3. In addition, some operating systems are very greedy: when booting up they try to draw to them each hard disk that is written with the signature (label) of a foreign operating system.

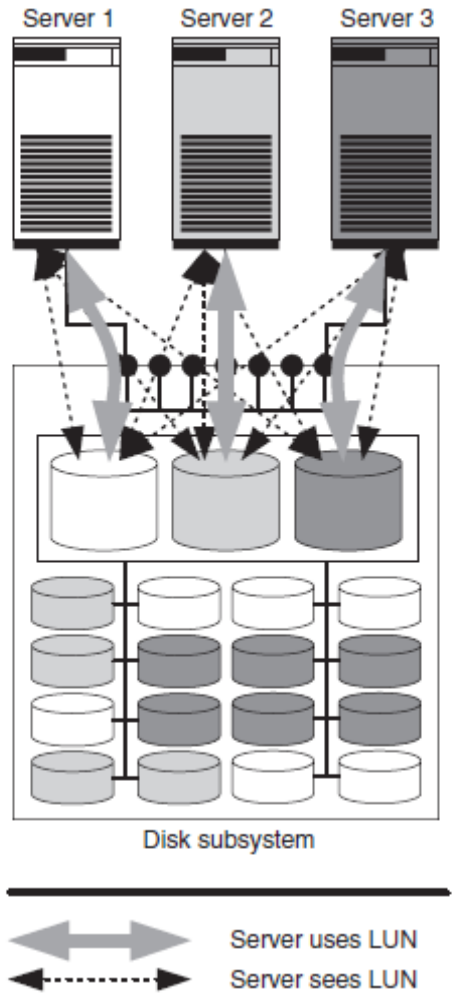


Figure Chaos: Each server works to its own virtual hard disk. Without LUN masking each server sees all hard disks. A configuration error on server 1 can destroy the data on the other two servers. The data is thus poorly protected.

Without LUN masking, therefore, the use of the hard disk must be very carefully configured in the operating systems of the participating servers. LUN masking brings order to this chaos by assigning the hard disks that are externally visible to servers. As a result, it limits the visibility of exported disks within the disk subsystem. Figure 2.27 shows how LUN masking brings order to the chaos of Figure 2.26. Each server now sees only the hard disks that it actually requires. LUN masking thus acts as a filter between the exported hard disks and the accessing servers. It is now no longer possible to destroy data that belongs to applications that run on another server. Configuration errors are still possible, but the consequences are no longer so

devastating. Furthermore, configuration errors can now be more quickly traced since the information is bundled within the disk subsystem instead of being distributed over all servers.

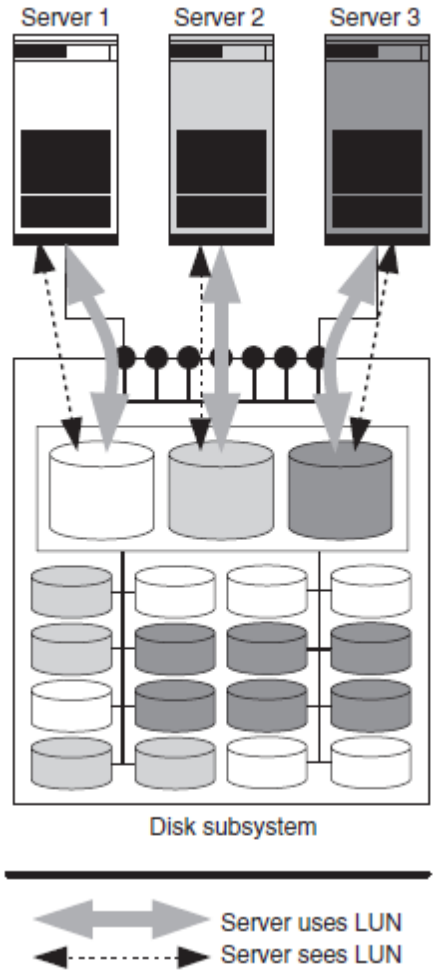


Figure Order: each server works to its own virtual hard disk. With LUN masking, each server sees only its own hard disks. A configuration error on server 1 can no longer destroy the data of the two other servers. The data is now protected.

We differentiate between port-based LUN masking and server-based LUN masking. Port-based LUN masking is the 'poor man's LUN masking', it is found primarily in low-end disk subsystems. In port-based LUN masking the filter only works using the granularity of a port. This means that all servers connected to the disk subsystem via the same port see the same disks.

Server-based LUN masking offers more flexibility. In this approach every server sees only the hard disks assigned to it, regardless of which port it is connected via or which other servers are connected via the same port.

3.3 AVAILABILITY OF DISK SUBSYSTEMS

Disk subsystems are assembled from standard components, which have a limited fault-tolerance. In this chapter we have shown how these standard components are combined in order to achieve a level of fault-tolerance for the entire disk subsystem that lies significantly above the fault-tolerance of the individual components. Today, disk subsystems can be constructed so that they can withstand the failure of any component without data being lost or becoming inaccessible. We can also say that such disk subsystems have no 'single point of failure'.

The following list describes the individual measures that can be taken to increase the availability of data:

• The data is distributed over several hard disks using RAID processes and supplemented by further data for error correction. After the failure of a physical hard disk, the data of the defective hard disk can be reconstructed from the remaining data and the additional data.

• Individual hard disks store the data using the so-called Hamming code. The Hamming code allows data to be correctly restored even if individual bits are changed on the hard disk. Self-diagnosis functions in the disk controller continuously monitor the rate of bit errors and the physical variables (e.g., temperature, spindle vibration). In the event of an increase in the error rate, hard disks can be replaced before data is lost.

• Each internal physical hard disk can be connected to the controller via two internal I/O channels. If one of the two channels fails, the other can still be used.

• The controller in the disk subsystem can be realised by several controller instances. If one of the controller instances fails, one of the remaining instances takes over the tasks of the defective instance.

• Other auxiliary components such as power supplies, batteries and fans can often be duplicated so that the failure of one of the components is unimportant. When connecting the power supply it should be ensured that the various power cables are at least connected through

various fuses. Ideally, the individual power cables would be supplied via different external power networks; however, in practice this is seldom realisable.

• Server and disk subsystem are connected together via several I/O channels. If one of the channels fails, the remaining ones can still be used.

• Instant copies can be used to protect against logical errors. For example, it would be possible to create an instant copy of a database every hour. If a table is 'accidentally' deleted, then the database could revert to the last instant copy in which the database is still complete.

• Remote mirroring protects against physical damage. If, for whatever reason, the original data can no longer be accessed, operation can continue using the data copy that was generated using remote mirroring.

• Consistency groups and write-order consistency synchronise the copying of multiple virtual hard disks. This means that instant copy and remote mirroring can even guarantee the consistency of the copies if the data spans multiple virtual hard disks or even multiple disk subsystems.

• LUN masking limits the visibility of virtual hard disks. This prevents data being changed or deleted unintentionally by other servers.

This list shows that disk subsystems can guarantee the availability of data to a very high degree. Despite everything it is in practice sometimes necessary to shut down and switch off a disk subsystem. In such cases, it can be very tiresome to co-ordinate all project groups to a common maintenance window, especially if these are distributed over different time zones.

Further important factors for the availability of an entire IT system are the availability of the applications or the application server itself and the availability of the connection between application servers and disk subsystems. Chapter 6 shows how multipathing can improve the connection between servers and storage systems and how clustering can increase the fault-tolerance of applications.

UNIT 4

I/O TECHNIQUES

Computers generate, process and delete data. However, they can only store data for very short periods. Therefore, computers move data to storage devices such as tape libraries and the disk subsystems discussed in the previous chapter for long-term storage and fetch it back from these storage media for further processing. So-called I/O techniques realise the data transfer between computers and storage devices. This chapter describes I/O techniques that are currently in use or that the authors believe will most probably be used in the coming years.

4.1 THE PHYSICAL I/O PATH FROM THE CPU TO THE STORAGE SYSTEM

In the computer, one or more CPUs process data that is stored in the CPU cache or in the random access memory (RAM). CPU cache and RAM are very fast; however, their data is lost when the power is been switched off. Furthermore, RAM is expensive in comparison to disk and tape storage. Therefore, the data is moved from the RAM to storage devices such as disk subsystems and tape libraries via system bus, host bus and I/O bus (Figure 3.1). Although storage devices are slower than CPU cache and RAM, Computers generate, process and delete data. However, they can only store data for very short periods. Therefore, computers move data to storage devices such as tape libraries and the disk subsystems discussed in the previous chapter for long-term storage and fetch it back from these storage media for further processing. So-called I/O techniques realise the data transfer between computers and storage devices. This chapter describes I/O techniques that are currently in use or that the authors believe will most probably be used in the coming years.

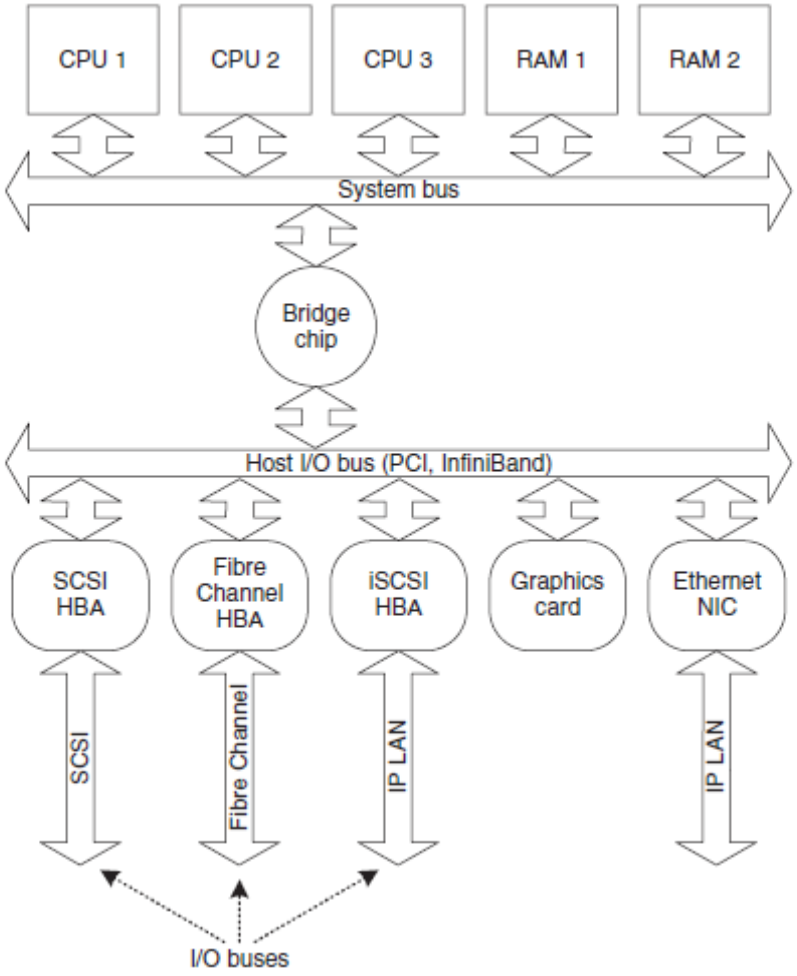


Figure 4.1 The physical I/O path from the CPU to the storage system consists of system bus, host I/O bus and I/O bus.

More recent technologies such as InfiniBand, Fibre Channel and Internet SCSI (iSCSI) replace individual buses with a serial network. For historic reasons the corresponding connections are still called host I/O bus or I/O bus. they compensate for this by being cheaper and by their ability to store data even when the power is switched off. Incidentally, the same I/O path also exists within a disk subsystem between the connection ports and the disk subsystem controller and between the controller and the internal hard disk (Figure 3.2).

At the heart of the computer, the system bus ensures the rapid transfer of data between

CPUs and RAM. The system bus must be clocked at a very high frequency so that it can supply the CPU with data sufficiently quickly. It is realised in the form of printed conductors on the main circuit board. Due to physical properties, high clock rates require
short printed conductors. Therefore, the system bus is kept as short as possible and thus connects only CPUs and main memory.

In modern computers as many tasks as possible are moved to special purpose processors such as graphics processors in order to free up the CPU for the processing of the application. These cannot be connected to the system bus due to the physical limitations mentioned above. Therefore, most computer architectures realise a second bus, the so-called host I/O bus. So-called bridge communication chips provide the connection between system bus and host I/O bus. Peripheral Component Interconnect (PCI) is currently the most widespread technology for the realisation of host I/O buses.

Device drivers are responsible for the control of and communication with peripheral devices of all types. The device drivers for storage devices are partially realised in the form of software that is processed by the CPU. However, part of the device driver for the communication with storage devices is almost always realised by firmware that is processed by special processors (Application Specific Integrated Circuits, ASICs). These
ASICs are currently partially integrated into the main circuit board, such as on-board SCSI controllers, or connected to the main board via add-on cards (PCI cards). These add-on cards are usually called network cards (Network Interface Controller, NIC) or simply controllers. Storage devices are connected to the server via the host bus adapter (HBA) or via the on-board controller. The communication connection between controller
and peripheral device is called the I/O bus.

The most important technologies for I/O buses are currently SCSI and Fibre Channel. SCSI defines a parallel bus that can connect up to 16 servers and storage devices with one another. Fibre Channel, on the other hand, defines different topologies for storage networks that can connect several millions of servers and storage devices. As an alternative to Fibre Channel, the industry is currently experimenting with different options for the realisation of storage networks by means of TCP/IP and Ethernet such as IP storage and FCoE. It is worth noting that all new technologies continue to use the SCSI protocol for device communication.

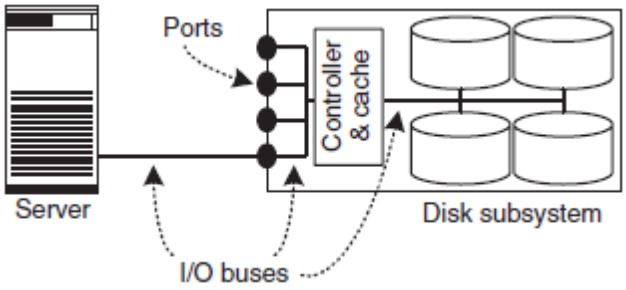


Figure 4.2 The same I/O techniques are used within a disk subsystem as those used between server and disk subsystem.

The Virtual Interface Architecture (VIA) is a further I/O protocol. VIA permits rapid and CPU-saving data exchange between two processes that run on two different servers or storage devices. In contrast to the I/O techniques discussed previously VIA defines only a protocol. As a medium it requires the existence of a powerful and low-error communication path, which is realised, for example, by means of Fibre Channel, Gigabit
Ethernet or InfiniBand. VIA could become an important technology for storage networks and server clusters. There are numerous other I/O bus technologies on the market that will not be discussed further in this book, for example, Serial Storage Architecture (SSA), IEEE 1394 (Apple's Firewire, Sony's i.Link), High-Performance Parallel Interface (HIPPI), Advanced Technology Attachment (ATA)/Integrated Drive Electronics (IDE), Serial ATA (SATA), Serial Attached SCSI (SAS) and Universal Serial Bus (USB). All have in common that they are either used by very few manufacturers or are not powerful enough for the connection of servers and storage devices. Some of these technologies can form small storage networks. However, none is anywhere near as flexible and scalable as the Fibre Channel and IP storage technologies described in this book.

4.2 SCSI

Small Computer System Interface (SCSI) was for a long time *the* technology for I/O buses in Unix and PC servers, is still very important today and will presumably remain so for a good many years to come. The first version of the SCSI standard was released in 1986. Since then SCSI has been continuously developed in order to keep it abreast with technical progress.

4.2.1 SCSI basics

As a medium, SCSI defines a parallel bus for the transmission of data with additional lines for the control of communication. The bus can be realised in the form of printed conductors on the circuit board or as a cable. Over time, numerous cable and plug types have been defined that are not directly compatible with one another (Table 3.1). A so- called daisy chain can connect up to 16 devices together (Figure 3.3).

The SCSI protocol defines how the devices communicate with each other via the SCSI bus. It specifies how the devices reserve the SCSI bus and in which format data is transferred. The SCSI protocol has been further developed over the years. For example, a server could originally only begin a new SCSI command when the previous SCSI command had been acknowledged by the partner; however, precisely this overlapping of SCSI commands is the basis for the performance increase achieved by RAID (Section 2.4). Today it is even possible using asynchronous I/O to initiate multiple concurrently write or read commands to a storage device at the same time.

The SCSI protocol introduces SCSI IDs (sometimes also called target ID or just ID) and Logical Unit Numbers (LUNs) for the addressing of devices. Each device in the SCSI bus must have an unambiguous ID, with the HBA in the server requiring its own ID. Depending upon the version of the SCSI standard, a maximum of 8 or 16 IDs are permitted per SCSI bus. Storage devices such as RAID disk subsystems, intelligent disk subsystems or tape libraries can include several subdevices, such as virtual hard disks, tape drives or a media changer to insert the tapes, which means that the IDs would be used up very quickly. Therefore, so-called LUNs were introduced in order to address subdevices within larger devices (Figure 3.4). A server can be equipped with several SCSI controllers. Therefore, the operating system must note three things for the differentiation of devices – controller ID, SCSI ID and LUN.

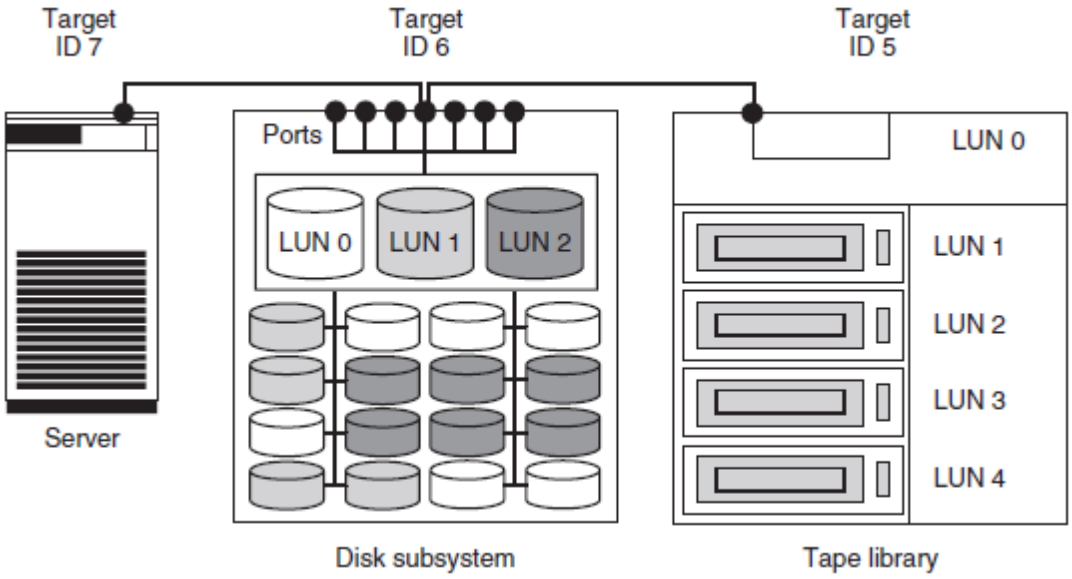


Figure 4.4 Devices on the SCSI bus are differentiated by means of target IDs. Components within devices (virtual hard disks, tape drives and the robots in the tape library) by LUNs.

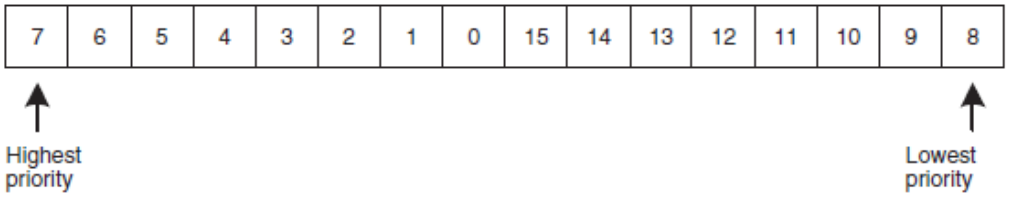


Figure 4.5 SCSI Target IDs with a higher priority win the arbitration of the SCSI bus. The priority of SCSI IDs is slightly trickier. Originally, the SCSI protocol permitted only eight IDs, with the ID '7' having the highest priority. More recent versions of the SCSI protocol permit 16 different IDs. For reasons of compatibility, the IDs '7' to '0' should retain the highest priority so that the IDs '15' to '8' have a lower priority (Figure 3.5). Devices (servers and storage devices) must reserve the SCSI bus (arbitrate) before they may send data through it. During the arbitration of the bus, the device that has the highest

priority SCSI ID always wins. In the event that the bus is heavily loaded, this can lead to devices with lower priorities never being allowed to send data. The SCSI arbitration procedure is therefore 'unfair'.

4.3 THE FIBRE CHANNEL PROTOCOL STACK

Fibre Channel is currently (2009) the technique most frequently used for implementing storage networks. Interestingly, Fibre Channel was originally developed as a backbone technology for the connection of LANs. The original development objective for Fibre Channel was to supersede Fast-Ethernet (100 Mbit/s) and Fibre Distributed Data Interface (FDDI). Meanwhile Gigabit Ethernet and 10-Gigabit Ethernet have become prevalent or will become prevalent in this market segment.

By coincidence, the design goals of Fibre Channel are covered by the requirements of a transmission technology for storage networks such as:

• Serial transmission for high speed and long distances;

• Low rate of transmission errors;

• Low delay (latency) of the transmitted data;

• Implementation of the Fibre Channel Protocol (FCP) in hardware on HBA cards to free up the server CPUs

In the early 1990s, Seagate was looking for a technology that it could position against IBM's SSA. With the support of the Fibre Channel industry, Fibre Channel was expanded by the arbitrated loop topology, which is cheaper than the originally developed fabric topology. This led to the breakthrough of Fibre Channel for the realisation of storage networks.

Fibre Channel is only one of the transmission technologies with which storage area networks (SANs) can be realised. Nevertheless, the terms 'Storage Area Network' and 'SAN' are often used synonymously with Fibre Channel technology. In discussions, newspaper articles and books the terms 'storage area network' and SAN are often used to mean a storage area network that is built up using Fibre Channel. The advantages of storage area networks and server-centric IT architectures can, however, also be achieved using other technologies for storage area networks, for example, iSCSI and FCoE. In this book we have taken great pains to express ourselves precisely. We do not use the terms 'storage area network' and 'SAN' on their own. For unambiguous differentiation we always also state the technology, for example, 'Fibre Channel SAN' or 'iSCSI SAN'. In statements about storage area networks in general that are independent of a specific technology we use the term 'storage network'. We use the term 'Fibre Channel' without

the suffix ‘SAN’ when we are referring to the transmission technology that underlies a Fibre Channel SAN.

For the sake of completeness we should also mention that the three letters ‘SAN’ are also used as an abbreviation for ‘System Area Network’. A System Area Network is a network with a high bandwidth and a low latency that serves as a connection between computers in a distributed computer system. In this book we have never used the abbreviation SAN in this manner. However, it should be noted that the VIA standard, for example, does use this second meaning of the abbreviation ‘SAN’.

The Fibre Channel protocol stack is subdivided into five layers (Figure 3.8). The lower four layers, FC-0 to FC-3 define the fundamental communication techniques, i.e. the physical levels, the transmission and the addressing. The upper layer, FC-4, defines how application protocols (upper layer protocols, ULPs) are mapped on the underlying Fibre Channel network. The use of the various ULPs decides, for example, whether a real Fibre

Channel network is used as an IP network, a Fibre Channel SAN (i.e. as a storage network) or both at the same time. The link services and fabric services are located quasi-adjacent to the Fibre Channel protocol stack. These services will be required in order to administer and operate a Fibre Channel network. Basic knowledge of the Fibre Channel standard helps to improve understanding of the possibilities for the use of Fibre Channel for a Fibre Channel SAN. This section (Section 3.3) explains technical details of the Fibre Channel protocol. We will restrictthe level of detail to the parts of the Fibre Channel standard that are helpful in the administration or the design of a Fibre Channel SAN. Building upon this, the next section (Section 3.4) explains the use of Fibre Channel for storage networks.

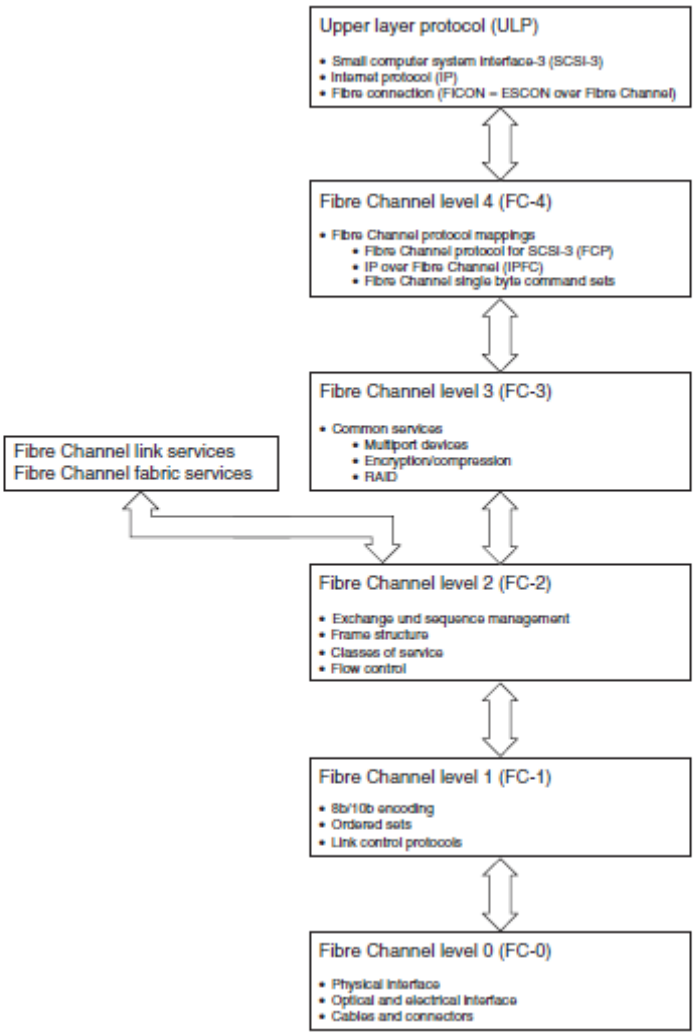


Figure 4.8 The Fibre Channel protocol stack is divided into two parts: the lower four layers (FC-0 to FC-3) realise the underlying Fibre Channel transmission technology. The link services and the fabric services help to administer and configure the Fibre Channel network. The upper layer (FC-4) defines how the application protocols (for example, SCSI and IP) are mapped on a Fibre Channel network.

4.3.1 Links, ports and topologies

The Fibre Channel standard defines three different topologies: fabric, arbitrated loop and point-to-point (Figure 3.9). Point-to-point defines a bi-directional connection between two devices. Arbitrated loop defines a unidirectional ring in which only two devices can ever exchange data with one another at any one time. Finally, fabric defines a network in which several devices

can exchange data simultaneously at full bandwidth. A fabric basically requires one or more Fibre Channel switches connected together to form a control centre between the end devices. Furthermore, the standard permits the connection of one or more arbitrated loops to a fabric. The fabric topology is the most frequently used of all topologies, and this is why more emphasis is placed upon the fabric topology than on the two other topologies in the following sections.

Common to all topologies is that devices (servers, storage devices and switches) must be equipped with one or more Fibre Channel ports. In servers, the port is generally realized by means of so-called HBAs (for example, PCI cards) that are also fitted in the server.A port always consists of two channels, one input and one output channel.

The connection between two ports is called a link. In the point-to-point topology and in the fabric topology the links are always bi-directional: in this case the input channel and the output channel of the two ports involved in the link are connected together by a cross, so that every output channel is connected to an input channel. On the other hand, the links of the arbitrated loop topology are unidirectional: each output channel is connected to the input channel of the next port until the circle is closed. The cabling of an arbitrated loop can be simplified with the aid of a hub. In this configuration the end things for the larger devices (Figure 3.4).

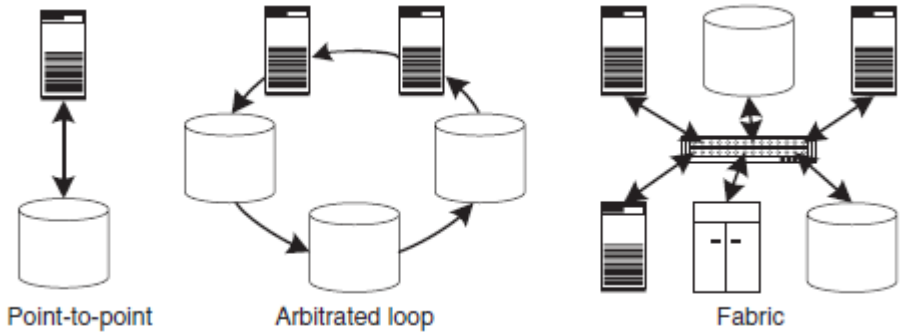


Figure 4.9 The fabric topology is the most flexible and scalable Fibre Channel topology.

A server can be equipped with several devices are bi-directionally connected to the hub; the wiring within the hub ensures that the unidirectional data flow within the arbitrated loop is

maintained. The fabric and arbitrated loop topologies are realised by different, incompatible protocols.

We can differentiate between the following port types with different capabilities:

• N-Port (Node Port): Originally the communication of Fibre Channel was developed around N-Ports and F-Ports, with 'N' standing for 'node' and 'F' for 'fabric'. An N-Port describes the capability of a port as an end device (server, storage device), also called node, to participate in the fabric topology or to participate in the point-to-point topology as a partner.

• F-Port (Fabric Port): F-Ports are the counterpart to N-Ports in the Fibre Channel switch. The F-Port knows how it can pass a frame that an N-Port sends to it through the Fibre Channel network on to the desired end device.

• L-Port (Loop Port): The arbitrated loop uses different protocols for data exchange than the fabric. An L-Port describes the capability of a port to participate in the arbitrated loop topology as an end device (server, storage device). More modern devices are now fitted with NL-Ports instead of L-Ports. Nevertheless, old devices that are fitted with an L-Port are still encountered in practice.

• NL Port (Node Loop Port): An NL-Port has the capabilities of both an N-Port and an L-Port. An NL-Port can thus be connected both in a fabric and in an arbitrated loop. Most modern HBA cards are equipped with NL-Ports.

• FL-Port (Fabric Loop Port): An FL-Port allows a fabric to connect to a loop. However, this is far from meaning that end devices in the arbitrated loop can communicate with end devices in the fabric. More on the subject of connecting fabric and arbitrated loop can be found in Section 3.4.3.

• E-Port (Expansion Port): Two Fibre Channel switches are connected together by E-Ports. E-Ports transmit the data from end devices that are connected to two different Fibre Channel switches. In addition, Fibre Channel switches smooth out information over the entire Fibre Channel network via E-ports.

• G-Port (Generic Port): Modern Fibre Channel switches configure their ports automatically.Such ports are called G-Ports. If, for example, a Fibre Channel switch is connected to a further Fibre Channel switch via a G-Port, the G-Port configures itself as an E-Port.

• B-Port (Bridge Port): B-Ports serve to connect two Fibre Channel switches together via

Asynchronous Transfer Mode (ATM), SONET/SDH (Synchronous Optical Networking/ Synchronous Digital Hierarchy) as well as Ethernet and IP. Thus Fibre Channel SANs that are a long distance apart can be connected together using classical Wide Area Network (WAN) techniques.

Some Fibre Channel switches have further, manufacturer-specific port types over and above those in the Fibre Channel standard: these port types provide additional functions. When using such port types, it should be noted that you can sometimes bind yourselfto the Fibre Channel switches of a certain manufacturer, which cannot subsequently be replaced by Fibre Channel switches of a different manufacturer.

4.3.2 FC-0: cables, plugs and signal encoding

FC-0 defines the physical transmission medium (cable, plug) and specifies which physical signals are used to transmit the bits '0' and '1'. In contrast to the SCSI bus, in which each bit has its own data line plus additional control lines, Fibre Channel transmits the bits sequentially via a single line. In general, buses come up against the problem that the signals have a different transit time on the different data lines (skew), which means that the clock rate can only be increased to a limited degree in buses. The different signal

transit times can be visualised as the hand rail in an escalator that runs faster or slower than the escalator stairs themselves.

Fibre Channel therefore transmits the bits serially. This means that, in contrast to the parallel bus, a high transfer rate is possible even over long distances. The high transfer rate of serial transmission more than compensates for the parallel lines of a bus. The transmission rate of actual components increases every few years. Table 3.2 depicts the market entry and the roadmap for new higher transfer rates as of 2009. Fibre Channel components are distinguished as Base2 components and Base10 components. Base2 components have to maintain backward compatibility of at least two previous Base2 generations. For instance 8GFC components must be interoperable with 4GFC and 2GFC

components. Base10 components have to maintain backward compatibility of at least one Base10 generation, in which case, as an exception for 100GFC no backward compatibility is expected.

Table 4.2 Fibre Channel components are distinguished in Base2 components (upper table) and Base10 components (lower table). The table shows the roadmap as of 2009.

Product Naming	MByte/s (per direction)	T11 Spec Completed	Market Availability
1GFC	100	1996	1997
2GFC	200	2000	2001
4GFC	400	2003	2005
8GFC	800	2006	2008
16GFC	1,600	2009	2011
32GFC	3,200	2012	Market demand
64GFC	6,400	2016	Market demand
128GFC	12,800	2020	Market demand
Product Naming	**MByte/s (per direction)**	**T11 Spec Completed**	**Market Availability**
10GFC	1,200	2003	2004
20GFC	2,400	2008	2008
40GFC	4,800	2009	2011
80GFC	9,600	future	Market demand
100GFC	12,000	future	Market demand
160GFC	19,200	future	Market demand

When considering the transfer rate it should be noted that in the fabric and point-to-point topologies the transfer is bi-directional and full duplex, which means that, for instance for 2GFC components the transfer rate of 200 MByte/s is available in each direction. The Fibre Channel standard demands that a single bit error may occur at most once in every 1012 transmitted bits. On average, this means that for a 100 Mbit/s connection under full load a bit error may occur only every 16.6 minutes. The error recognition and handling mechanisms of the higher protocol layers are optimised for the maintenance of this error rate. Therefore, when installing a Fibre Channel network it is recommended that the cable is properly laid so that the bit error rate of 10−12 is, where possible, also achieved for connections from end device to end device, i.e. including all components connected in between such as repeaters and switches.

Different cable and plug types are defined (Figure 3.10). Fiber-optic cables are more expensive than copper cables. They do, however, have some advantages:

• Greater distances possible than with copper cable;

• Insensitivity to electromagnetic interference;

• No electromagnetic radiation;

• No electrical connection between the devices;

• No danger of ‘cross-talking’;

• Greater transmission rates possible than with copper cable

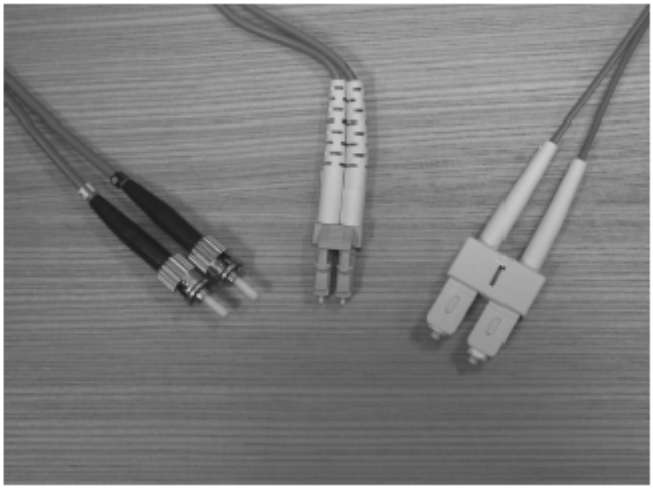


Figure 4.10 Three different plug types for fiber-optic cable.

Cables for long distances are more expensive than those for short distances. The definition of various cables makes it possible to choose the most economical technology for each distance to be bridged.

Typically the Fibre Channel standard specifies for a given medium a minimum distance which must be supported. It is assumed that when respective components enter the market, the state of technology can ensure the error rate for the specified minimum distance. Over the time, further technical improvements and the proper laying of cable enable even larger distances to be bridged in actual installations.

The reduction in the supported distances could present a problem when the equipment of an existing Fibre Channel SAN is upgraded from one generation to the next generation components. Due to technical limitations next generation components typically support for the same medium shorter distances than current generation components. This is especially an issue for longer instances when cables are installed between two collocated data centers which are several 100 meters apart.

4.3.3 FC-1: 8b/10b encoding, ordered sets and link control protocol

FC-1 defines how data is encoded before it is transmitted via a Fibre Channel cable (8b/10b encoding). FC-1 also describes certain transmission words (ordered sets) that are required for the administration of a Fibre Channel connection (link control protocol).

8b/10b Encoding

In all digital transmission techniques, transmitter and receiver must synchronise their clock-pulse rates. In parallel buses the clock rate is transmitted via an additional data line. By contrast, in the serial transmission used in Fibre Channel only one data line is available through which the data is transmitted. This means that the receiver must regenerate the clock rate from the data stream.

The receiver can only synchronise the rate at the points where there is a signal change in the medium. In simple binary encoding (Figure 3.11) this is only the case if the signal changes from '0' to '1' or from '1' to '0'. In Manchester encoding there is a signal change for every bit transmitted. Manchester encoding therefore creates two physical signals for each bit transmitted. It therefore requires a transfer rate that is twice as high as that for binary encoding. Therefore, Fibre Channel – like many other transmission techniques – uses binary encoding, because at a given rate of signal changes more bits can be transmitted than is the case for Manchester encoding.

The problem with this approach is that the signal steps that arrive at the receiver are not always the same length (jitter). This means that the signal at the receiver is sometimes a little longer and sometimes a little shorter (Figure 3.12). In the escalator analogy this means that the escalator bucks. Jitter can lead to the receiver losing synchronisation with the received signal. If, for example, the transmitter sends a sequence of ten zeros, the receiver cannot decide whether it is a sequence of nine, ten or eleven zeros.

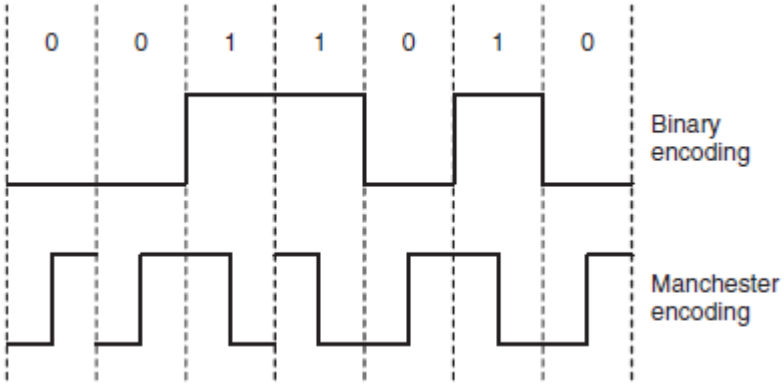


Figure 4.11 In Manchester encoding at least one signal change takes place for every bit transmitted.

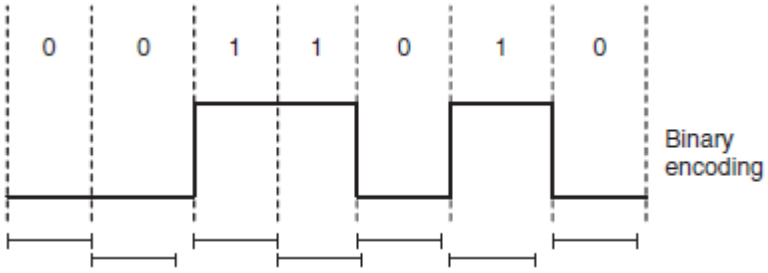


Figure 4.12 Due to physical properties the signals are not always the same length at the receiver (jitter).

If we nevertheless wish to use binary encoding, then we have to ensure that the data stream generates a signal change frequently enough that jitter cannot strike. The so-called 8b/10b encoding represents a good compromise. 8b/10b encoding converts an 8-bit byte to be transmitted into a 10-bit character, which is sent via the medium instead of the 8-bit byte. For Fibre Channel this means, for example, that a useful transfer rate of 100 MByte/s requires a raw transmission rate of 1 Gbit/s instead of 800 Mbit/s. Incidentally, 8b/10b encoding is also used for the Enterprise System Connection Architecture (ESCON), SSA, Gigabit Ethernet and InfiniBand. Finally, it should be noted that 1 Gigabyte Fibre Channel uses the 64b/66b encoding variant for certain cable types.

Expanding the 8-bit data bytes to 10-bit transmission character gives rise to the following advantages:

• In 8b/10b encoding, of all available 10-bit characters, only those that generate a bit sequence that contains a maximum of five zeros one after the other or five ones oneafter the other for any desired combination of the 10-bit character are selected. Therefore, a signal change takes place at the latest after five signal steps, so that the clock synchronisation of the receiver is guaranteed.

• A bit sequence generated using 8b/10b encoding has a uniform distribution of zeros and ones. This has the advantage that only small direct currents flow in the hardware that processes the 8b/10b encoded bit sequence. This makes the realisation of Fibre Channel hardware components simpler and cheaper.

• Further 10-bit characters are available that do not represent 8-bit data bytes. These additional characters can be used for the administration of a Fibre Channel link.

Ordered sets

Fibre Channel aggregates four 10-bit transmission characters to form a 40-bit transmission word. The Fibre Channel standard differentiates between two types of transmission word: data words and ordered sets. Data words represent a sequence of four 8-bit data bytes. Data words may only stand between a Start-of-Frame (SOF) delimiter and an End-of-Frame (EOF) delimiter.

Ordered sets may only stand between an EOF delimiter and a SOF delimiter, with SOFs and EOFs themselves being ordered sets. All ordered sets have in common that they begin with a certain transmission character, the so-called K28.5 character. The K28.5 character includes a special bit sequence that does not occur elsewhere in the data stream. The input channel of a Fibre Channel port can therefore use the K28.5 character to divide the continuous incoming bit stream into 40-bit transmission words when initialising a Fibre Channel link or after the loss of synchronisation on a link.

Link control protocol

With the aid of ordered sets, FC-1 defines various link level protocols for the initialization and administration of a link. The initialisation of a link is the prerequisite for data exchange by

means of frames. Examples of link level protocols are the initialisation and arbitration of an arbitrated loop.

4.3.4 FC-2: data transfer

FC-2 is the most comprehensive layer in the Fibre Channel protocol stack. It determines how larger data units (for example, a file) are transmitted via the Fibre Channel network. It regulates the flow control that ensures that the transmitter only sends the data at a speed that the receiver can process it. And it defines various service classes that are tailored to the requirements of various applications.

Exchange, sequence and frame

FC-2 introduces a three-layer hierarchy for the transmission of data (Figure 3.13). At the top layer a so-called exchange defines a logical communication connection between two end devices. For example, each process that reads and writes data could be assigned its own exchange. End devices (servers and storage devices) can simultaneously maintain several exchange relationships, even between the same ports. Different exchanges help the FC-2 layer to deliver the incoming data quickly and efficiently to the correct receiver
in the higher protocol layer (FC-3).

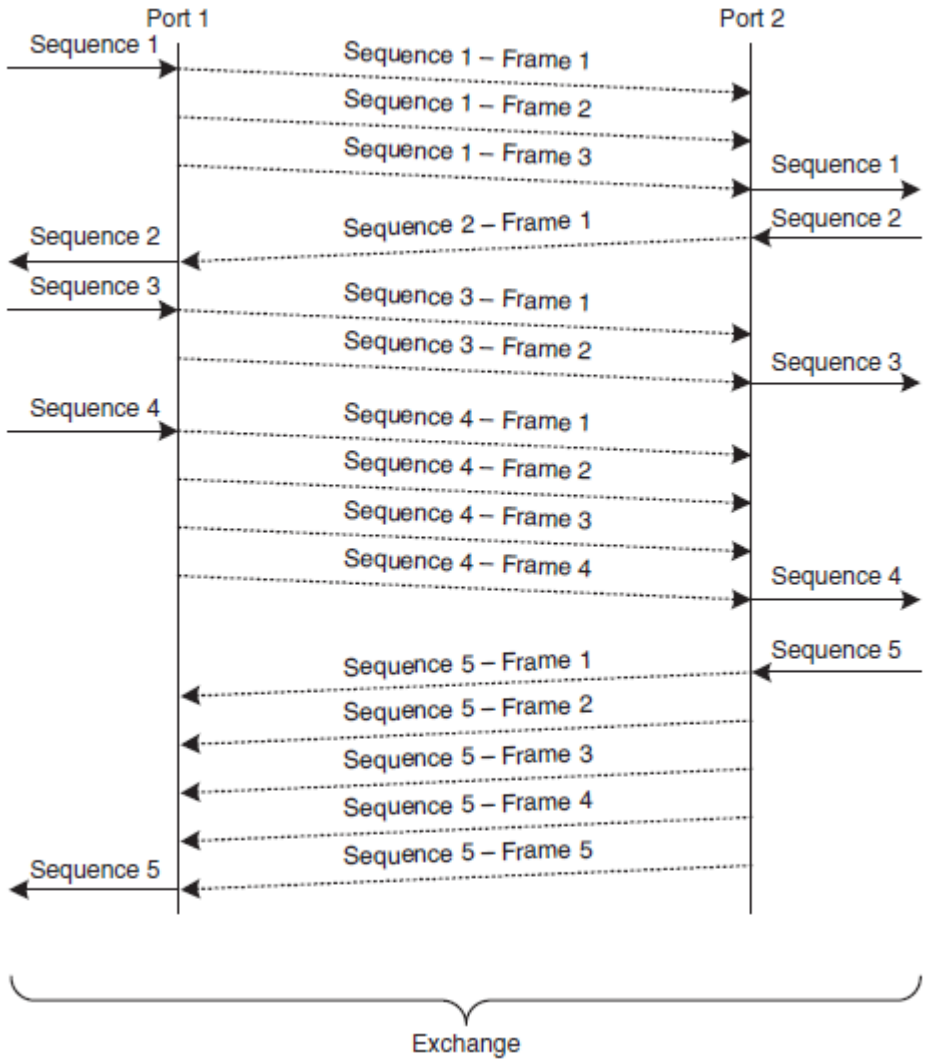


Figure 4.13 One sequence is transferred after another within an exchange. Large sequences are broken down into several frames prior to transmission. On the receiver side, a sequence is not delivered to the next highest protocol layer (FC-3) until all the frames of the sequence have arrived.

A sequence is a larger data unit that is transferred from a transmitter to a receiver. Only one sequence can be transferred after another within an exchange. FC-2 guarantees that sequences are delivered to the receiver in the same order they were sent from the transmitter; hence the name ‘sequence’. Furthermore, sequences are only delivered to the next protocol layer up when all frames of the sequence have arrived at the receiver (Figure 3.13). A sequence could

represent the writing of a file or an individual database transaction. A Fibre Channel network transmits control frames and data frames. Control frames contain no useful data, they signal events such as the successful delivery of a data frame. Data frames transmit up to 2,112 bytes of useful data. Larger sequences therefore have to be broken down into several frames. Although it is theoretically possible to agree upon different maximum frame sizes, this is hardly ever done in practice.

A Fibre Channel frame consists of a header, useful data (payload) and a Cyclic Redundancy Checksum (CRC) (Figure 3.14). In addition, the frame is bracketed by an start-of-frame (SOF) delimiter and an end-of-frame (EOF) delimiter. Finally, six filling words must be transmitted by means of a link between two frames. In contrast to Ethernet

and TCP/IP, Fibre Channel is an integrated whole: the layers of the Fibre Channel protocol stack are so well harmonised with one another that the ratio of payload to protocol overhead is very efficient at up to 98%. The CRC checking procedure is designed to recognise all transmission errors if the underlying medium does not exceed the specified error rate of 10−12. Error correction takes place at sequence level: if a frame of a sequence is wrongly transmitted, the entire sequence is re-transmitted. At gigabit speed it is more efficient to resend a complete sequence than to extend the Fibre Channel hardware so that individual

lost frames can be resent and inserted in the correct position. The underlying protocol layer must maintain the specified maximum error rate of 10−12 so that this procedure is efficient.

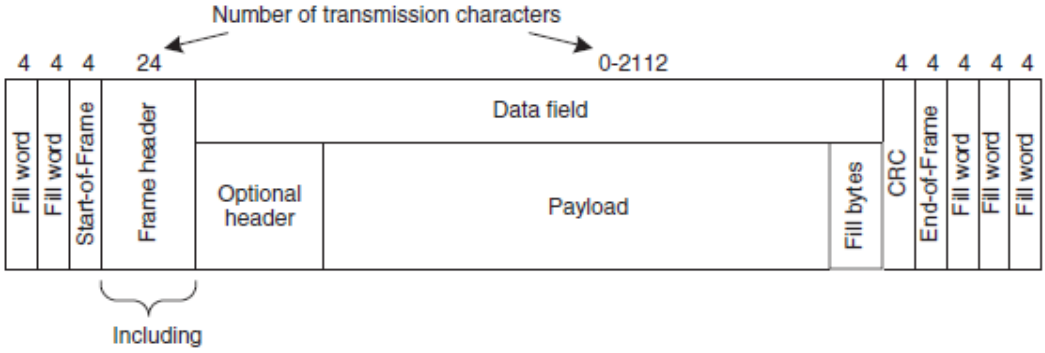


Figure 4.14 The Fibre Channel frame format.

Frame Destination Address (D_ID)

Frame Source Address (S_ID)

Sequence ID

Number of the frame within the sequence

Exchange ID

Flow control

Flow control ensures that the transmitter only sends data at a speed that the receiver can receive it. Fibre Channel uses the so-called credit model for this. Each credit represents the capacity of the receiver to receive a Fibre Channel frame. If the receiver awards the transmitter a credit of '4', the transmitter may only send the receiver four frames. The transmitter may not send further frames until the receiver has acknowledged the receipt of at least some of the transmitted frames.

FC-2 defines two different mechanisms for flow control: end-to-end flow control and link flow control (Figure 3.15). In end-to-end flow control two end devices negotiate the end-to-end credit before the exchange of data. The end-to-end flow control is realized on the HBA cards of the end devices. By contrast, link flow control takes place at each physical connection. This is achieved by two communicating ports negotiating the buffer-to-buffer credit. This means that the link flow control also takes place at the Fibre Channel switches.

Service classes

The Fibre Channel standard defines six different service classes for exchange of data between end devices. Three of these defined classes (Class 1, Class 2 and Class 3) are realised in products available on the market, with hardly any products providing the connection-oriented Class 1. Almost all new Fibre Channel products (HBAs, switches, storage devices) support the service classes Class 2 and Class 3, which realise a packet-oriented service (datagram service). In addition, Class F serves for the data exchange between the switches within a fabric. Class 1 defines a connection-oriented communication connection between two node ports: a Class 1 connection is opened before the transmission of frames. This specifies a route through the Fibre Channel network. Thereafter, all frames take the same route through the Fibre Channel network so that frames are delivered in the sequence in which they were transmitted. A Class 1 connection guarantees the availability of the full bandwidth. A port thus cannot send any other frames while a Class 1 connection is open.

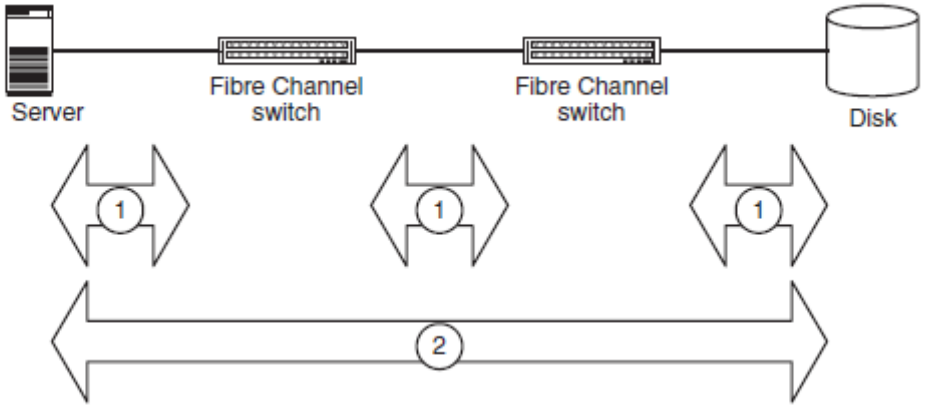


Figure 4.15 In link flow control the ports negotiate the buffer-to-buffer credit at each link (1). By contrast, in end-to-end flow control the end-to-end credit is only negotiated between the end devices (2).

4.3.6 Link services: login and addressing

Link services and the fabric services discussed in the next section stand next to the Fibre Channel protocol stack. They are required to operate data traffic over a Fibre Channel network. Activities of these services do not result from the data traffic of the application protocols. Instead, these services are required to manage the infrastructure of a Fibre Channel network and thus the data traffic on the level of the application protocols. For example, at any given time the switches of a fabric know the topology of the whole network.

Login

Two ports have to get to know each other before application processes can exchange data over them. To this end the Fibre Channel standard provides a three-stage login mechanism. (Figure 3.20):

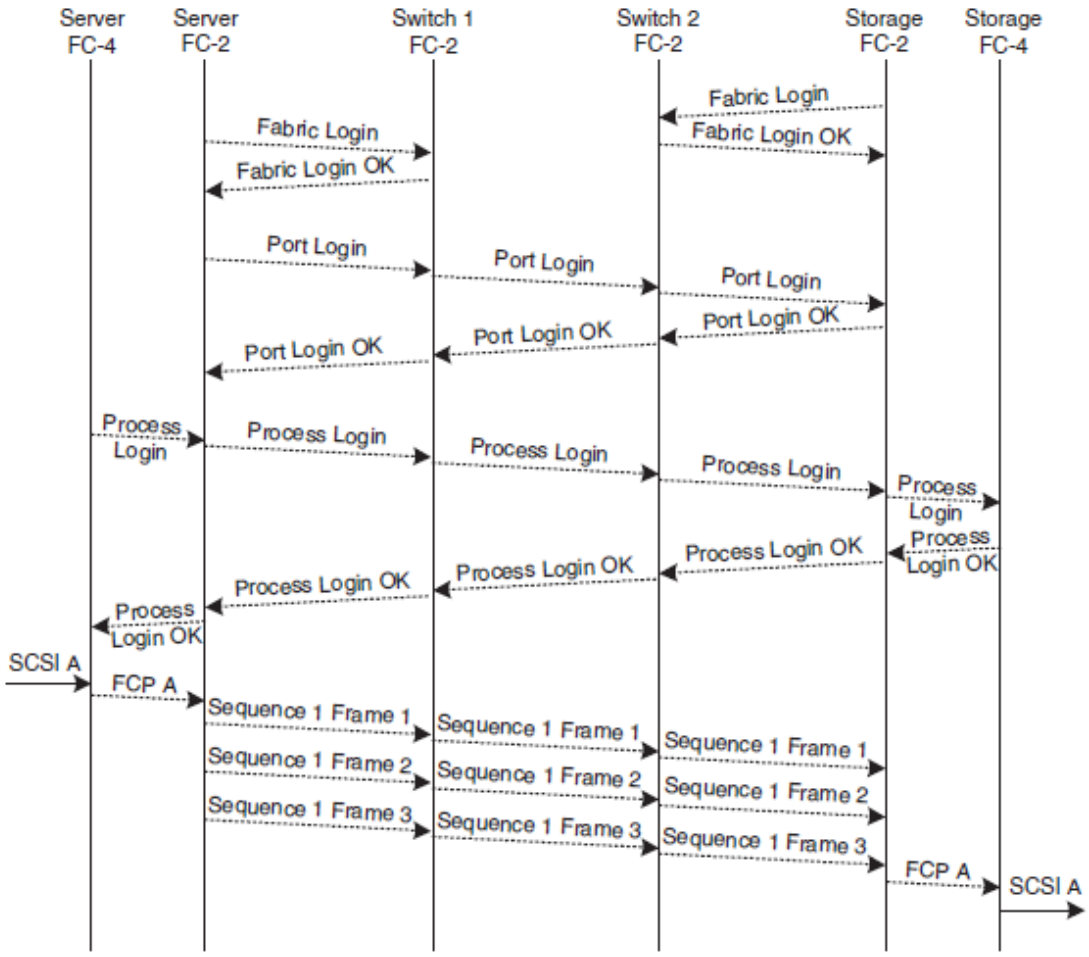


Figure 4.20 Fabric login, N-Port login and process login are the prerequisites for data exchange.

• *Fabric login (FLOGI)*

The fabric login establishes a session between an N-Port and a corresponding F-Port. The fabric login takes place after the initialisation of the link and is an absolute prerequisite for the exchange of further frames. The F-Port assigns the N-Port a dynamic address. In addition, service parameters such as the buffer-to-buffer credit are negotiated.The fabric login is crucial for the point-to-point topology and for the fabric topology. An N-Port can tell from the response of the corresponding port whether it is a fabric topology or a point-to-point topology. In arbitrated loop topology the fabric login is optional.

N-Port login (PLOGI)

N-Port login establishes a session between two N-ports. The N-Port login takes place after the fabric login and is a compulsory prerequisite for the data exchange at FC-4 level. N-Port login negotiates service parameters such as end-to-end credit. N-Port login is optional for Class 3 communication and compulsory for all other service classes.

• *Process login (PRLI)*

Process login establishes a session between two FC-4 processes that are based upon two different N-Ports. These could be system processes in Unix systems and system partitions in mainframes. Process login takes place after the N-Port login. Process login is optional from the point of view of FC-2. However, some FC-4 protocol mappings call for a process login for the exchange of FC-4-specific service parameters.

Addressing

Fibre Channel differentiates between addresses and names. Fibre Channel devices (servers, switches, ports) are differentiated by a 64-bit identifier. The Fibre Channel standard defines different name formats for this. Some name formats guarantee that such a 64-bit identifier will only be issued once worldwide. Such identifiers are thus also known as World Wide Names (WWNs). On the other hand, 64-bit identifiers that can be issued several times in separate networks are simply called Fibre Channel Names (FCNs).

In practice this fine distinction between WWN and FCN is hardly ever noticed, with all 64-bit identifiers being called WWNs. In the following we comply with the general usage and use only the term WWN.

UNIT 5

FILE SYSTEMS AND NETWORK ATTACHED STORAGE (NAS)

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.

5.1 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

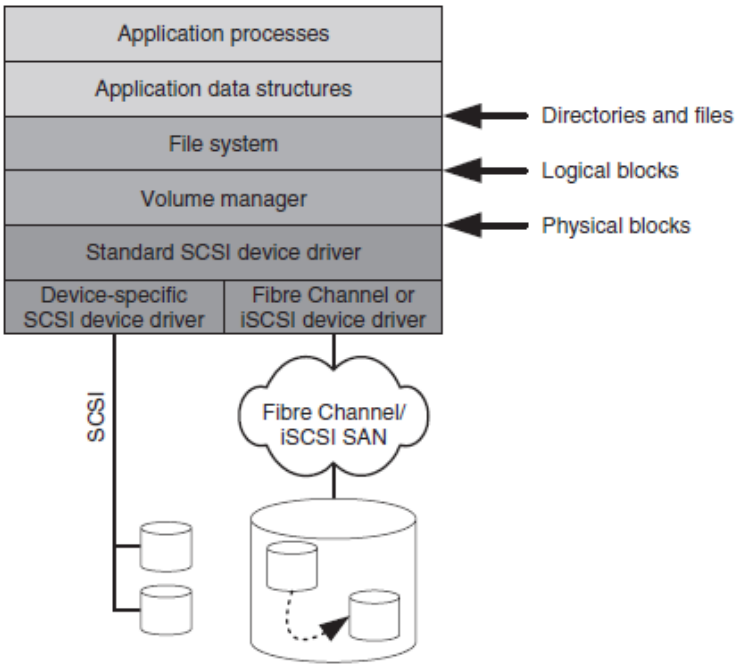


Figure 5.1 File system and volume manager manage the blocks of the block-oriented hard disks. Applications and users thus use the storage capacity of the disks via directories and files. (Figure 4.1). Together, these manage the blocks of the disk and make these available to users and applications via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

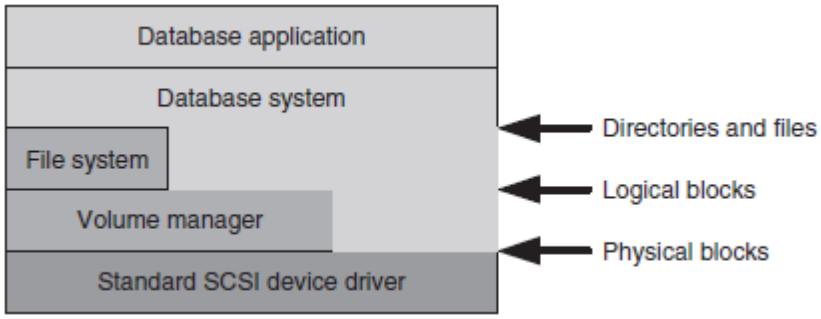


Figure 5.2 To increase performance, databases can get around the file system and manage the blocks themselves.

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).

5.2 NETWORK FILE SYSTEMS AND FILE SERVERS

Network file systems are the natural extension of local file systems. End users and applications can access directories and files that are physically located on a different computer – the file server – over a network file system (Section 4.2.1). File servers are so important in modern IT environments that preconfigured file servers, called Network Attached Storage (NAS), have emerged as a separate product category (Section 4.2.2). We highlight the performance bottlenecks of file servers (Section 4.2.3) and discuss the possibilities for the acceleration of network file systems. Finally, we introduce the Direct

Access File System (DAFS), a new network file system that relies upon RDMA and VI instead of TCP/IP.

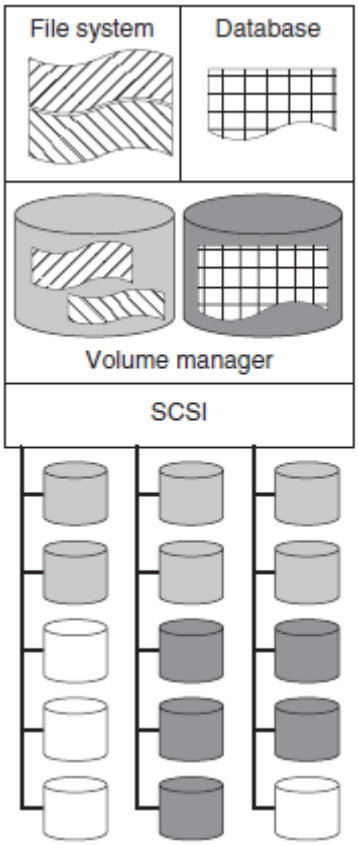


Figure 5.3 The volume manager aggregates physical hard disks into virtual hard disks, which it can break back down into smaller virtual hard disks. In the illustration one virtual hard disk is used directly from a database, the others are shared between two file systems.

5.2.1 Basic principle

The metaphor of directories and files for the management of data is so easy to understand that it was for a long time the prevailing model for the access of data over networks. So-called network file systems give end users and applications access to data stored on a different computer (Figure 4.5). The first widespread network file system was the Network File System (NFS) developed by Sun Microsystems, which is now *the* standard network file system on all Unix systems. Microsoft developed its own network file system – the Common Internet File System (CIFS) – for its Windows operating system and this is incompatible with NFS.

Today, various software solutions exist that permit the exchange of data between Unix

and Windows over a network file system. With the aid of network file systems, end users and applications can work on a common data set from various computers. In order to do this on Unix computers the system administrator must link a file system exported from an NFS server into the local directory structure using the mount command. On Windows computers, any end user can do this himself using the Map Network Drive command. Then, both in Unix and in Windows, the fact that data is being accessed from a network file system, rather than a local file system, is completely hidden apart from performance differences.

Long before the World Wide Web (WWW), the File Transfer Protocol (FTP) provided a mechanism by means of which users could exchange files over the Internet. Even today, FTP servers remain an important means of distributing freely available software and freely available documents. Unlike network file systems, access to FTP servers is clearly visible to the end user. Users require a special FTP client with which they can copy back and forwards between the FTP server and their local computer.

The Hyper Text Markup Language (HTML) and the Hyper Text Transfer Protocol (HTTP) radically changed the usage model of the Internet. In contrast to FTP, the data on the Internet is linked together by means of HTML documents. The user on the Internet no longer accesses individual files, instead he ‘surfs’ the WWW. He views HTML documents on his browser that are sometimes statically available on a HTTP server in the form of files or today are increasingly dynamically generated. Currently, graphical HTTP clients – the browsers – without exception have an integrated FTP client, with which they can easily ‘download’ files.

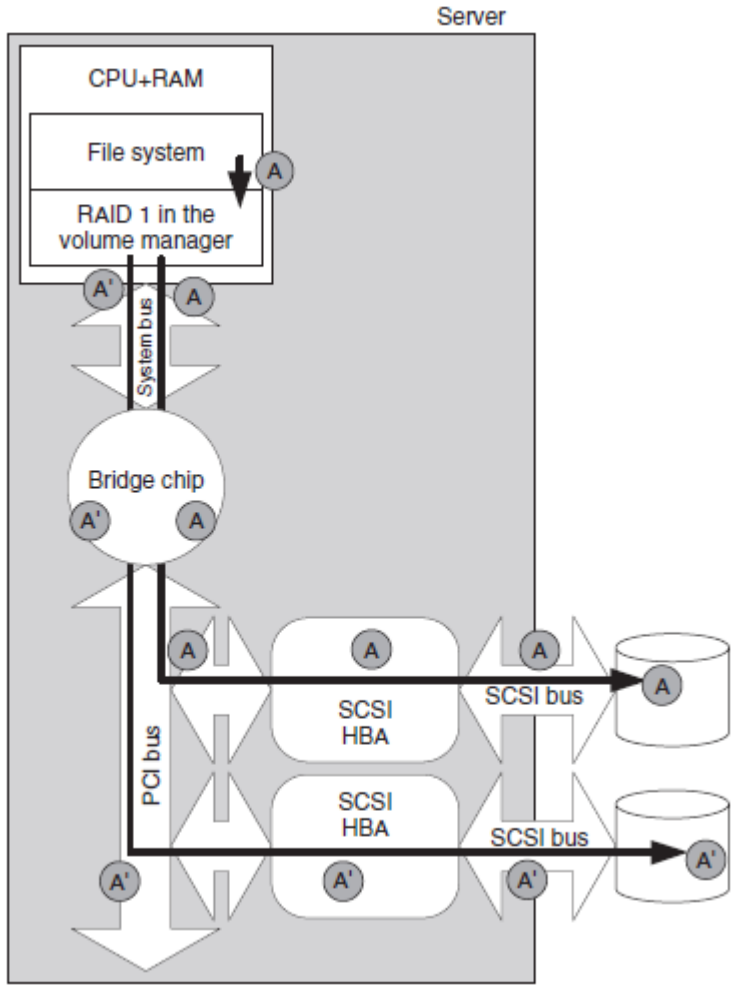


Figure 5.4 RAID in the volume manager loads the buses and CPU of the server. In RAID1, for example, each block written by the file system must be passed through all the buses twice.

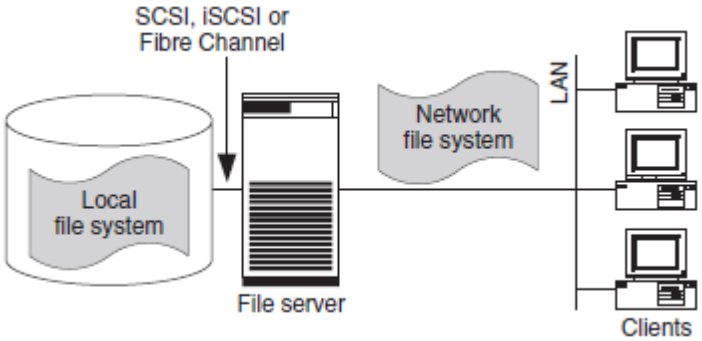


Figure 5.5 Network file systems make local files and directories available over the LAN.Several end users can thus work on common files (for example, project data, source code).

5.2.2 Network Attached Storage (NAS)

File servers are so important in current IT environments that they have developed into an independent product group in recent years. Network Attached Storage (NAS) is the name for preconfigured file servers. They consist of one or more internal servers, preconfigured disk capacity and usually a stripped-down or special operating system (Figure 4.6). NAS servers are usually connected via Ethernet to the LAN, where they provide their disk space as file servers. Web servers represent a further important field of application for NAS servers. By definition, the clients are located at the other end of the WAN so there is no alternative to communication over IP. Large NAS servers offer additional functions
such as snapshots, remote mirroring and backup over Fibre Channel SAN.

NAS servers were specially developed for file sharing. This has two advantages: since, by definition, the purpose of NAS servers is known, NAS operating systems can be significantly better optimised than generic operating systems. This means that NAS servers can operate more quickly than file servers on comparable hardware that are based upon a generic operating system.

The second advantage of NAS is that NAS servers provide Plug & Play file systems, i.e. connect – power up – use. In contrast to a generic operating system all functions can be removed that are not necessary for the file serving. NAS storage can therefore excel due to low installation and maintenance costs, which takes the pressure off system administrators.

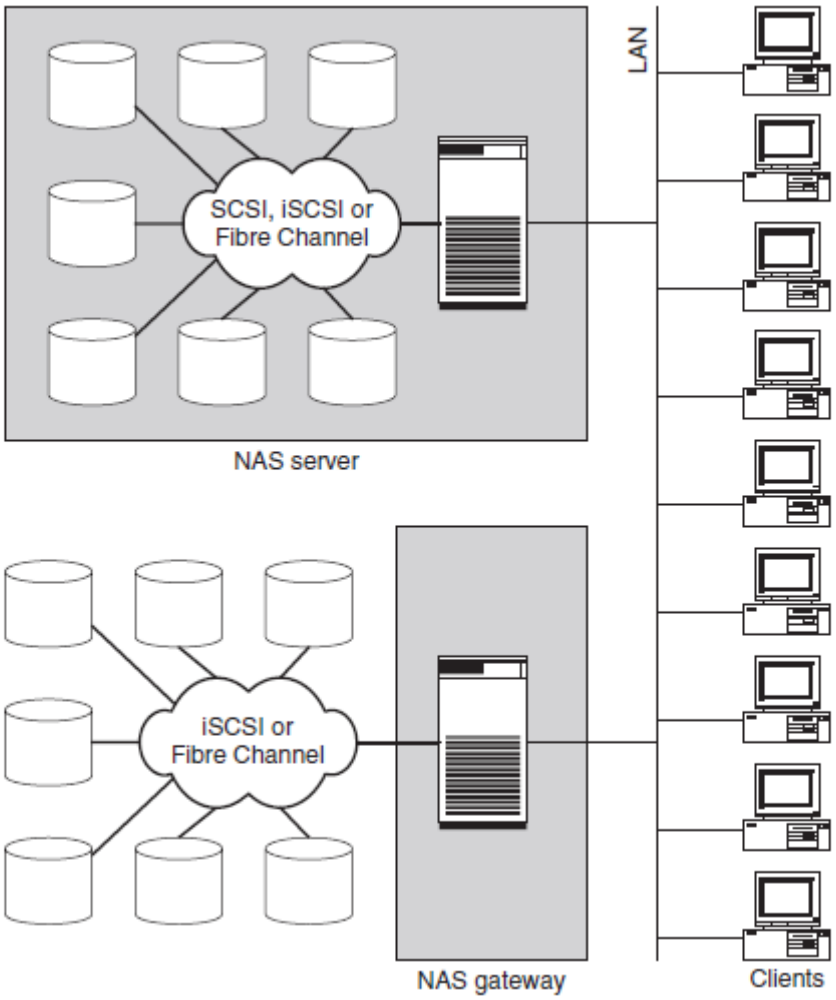


Figure 5.6 A NAS server is a preconfigured file server with internal hard disks, which makes its storage capacity available via LAN. A NAS gateway is a preconfigured file server that provides the storage capacity available in the storage network via the LAN. NAS servers are very scalable. For example the system administrator can attach a dedicated NAS server for every project or for every department. In this manner it is simple to expand large websites. E-mail file system full? No problem, I simply provide another NAS server for the next 10,000 users in my Ethernet. However, this approach can become a management nightmare if the storage requirement is very large, thus tens of NAS servers are required.

One disadvantage of NAS servers is the unclear upgrade path. For example, the internal server cannot simply be replaced by a more powerful server because this goes against the principle of the preconfigured file server. The upgrade options available in this situation

are those offered by the manufacturer of the NAS server in question. Performance bottlenecks for more I/O-intensive applications such as databases, backup, batch processes or multimedia applications represent a further important disadvantage of NAS servers. These are described in the following subsection.

5.2.3 Performance bottlenecks in file servers

Current NAS servers and NAS gateways, as well as classical file servers, provide their storage capacity via conventional network file systems such as NFS and CIFS or Internet protocols such as FTP and HTTP. Although these may be suitable for classical file sharing, such protocols are not powerful enough for I/O-intensive applications such as databases or video processing. Nowadays, therefore, I/O-intensive databases draw their storage from disk subsystems rather than file servers.

Let us assume for a moment that a user wishes to read a file on an NFS client, which is stored on a NAS server with internal SCSI disks. The NAS server's operating system first of all loads the file into the main memory from the hard disk via the SCSI bus, the PCI bus and the system bus, only to forward it from there to the network card via the system bus and the PCI bus. The data is thus shovelled through the system bus and the PCI bus on the file server twice (Figure 4.7). If the load on a file server is high enough, its buses can thus become a performance bottleneck.

When using classical network file systems the data to be transported is additionally copied from the private storage area of the application into the buffer cache of the kernel

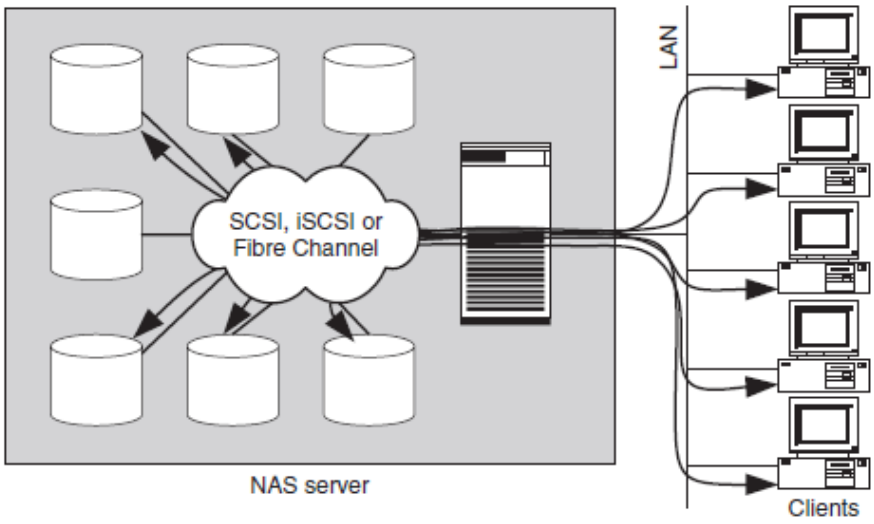


Figure 5.7 The file server becomes like the eye of the needle: en route between hard disk and client all data passes through the internal buses of the file server twice. on the transmitting computer before this copies the data via the PCI bus into the packet buffer of the network card. Every single copying operation increases the latency of the communication, the load on the CPU due to costly process changes between application processes and kernel processes, and the load on the system bus between CPU and main memory.

The file is then transferred from the network card to the NFS client via IP and Gigabit Ethernet. At the current state of technology most Ethernet cards can only handle a small part of the TCP/IP protocol independently, which means that the CPU itself has to handle the rest of the protocol. The communication from the Ethernet card to the CPU is initiated by means of interrupts. Taken together, this can cost a great deal of CPU time (Section 3.5.2, 'TCP/IP and Ethernet as an I/O technology').

5.2.4 Acceleration of network file systems

If we look at the I/O path from the application to the hard disks connected to a NAS server (Figure 4.15 on page 157), there are two places to start from to accelerate file sharing: (1) the underlying communication protocol (TCP/IP); and (2) the network file system (NFS, CIFS) itself. TCP/IP was originally developed to achieve reliable data exchange via unreliable transport routes. The TCP/IP protocol stack is correspondingly complex and CPU-intensive.

This can be improved first of all by so-called TCP/IP offload engines (TOEs), which in contrast to conventional network cards process a large part of the TCP/IP protocol stack on their own processor and thus significantly reduce the load on the server CPU (Section 3.5.2). It would be even better to get rid of TCP/IP all together. This is where communication techniques such as VIs and RDMA come into play (Section 3.6.2). Today there are various approaches for accelerating network file systems with VI and RDMA. The Socket Direct Protocol (SDP) represents an approach which combines the benefits of TOEs and RDMA-enabled transport (Section 3.6.3). Hence, protocols based on TCP/IP such as NFS and CIFS can – without modification – benefit via SDP from RDMA-enabled transport. Other approaches map existing network file systems directly onto RDMA. For example, a subgroup of the Storage Networking Industry Association (SNIA) is working on the protocol mapping of NFS on RDMA. Likewise, it would also be feasible for Microsoft to develop a CIFS implementation that uses RDMA instead of TCP/IP as the communication protocol. The advantage of this approach is that the network file systems NFS or CIFS that have matured over the years merely have a new communication mechanism put underneath them. This makes it possible to shorten the development and testing cycle so that the quality requirements of production environments can be fulfilled comparatively quickly.

A greater step is represented by newly developed network file systems, which from the start require a reliable network connection, such as the DAFS (Section 4.2.5), the family of the so-called shared disk file systems (Section 4.3) and the virtualisation on the file-level (Section 5.5).

5.2.5 Case study: The Direct Access File System (DAFS)

The Direct Access File System (DAFS) is a newly developed network file system that is tailored to the use of RDMA. It is based upon NFS version 4, requires VI and can fully utilise its new possibilities. DAFS makes it possible for several DAFS servers together to provide the storage space for a large file system (Figure 4.8). It remains hidden from the application server – as the DAFS client – which of these DAFS servers the actual data is located in (Figure 4.9).

The communication between DAFS client and DAFS server generally takes place by means of RDMA. The use of RDMA means that access to data that lies upon a DAFS server is nearly as quick as access to local data. In addition, typical file system operations such as the address

conversion of files to SCSI block addresses, which naturally also require the CPU, are offloaded from the application server to the DAFS server.

An important function of file sharing in general is the synchronisation of concurrent accesses to file entries – i.e. metadata such as file names, access rights, etc. – and file contents, in order to protect the consistency of the data and metadata. DAFS makes it possible to cache the locks at the client side so that a subsequent access to the same data requires no interaction with the file server. If a node requires the lock entry of a different node, then this transmits the entry without time-out. DAFS uses lease-based locking in order to avoid the permanent blocking of a file due to the failure of a client.

Furthermore, it possesses recovery mechanisms in case the connection between DAFS client and DAFS server is briefly interrupted or a different server from the cluster has to step in. Similarly, DAFS takes over the authentication of client and server and furthermore can also authenticate individual users in relation to a client-server session.

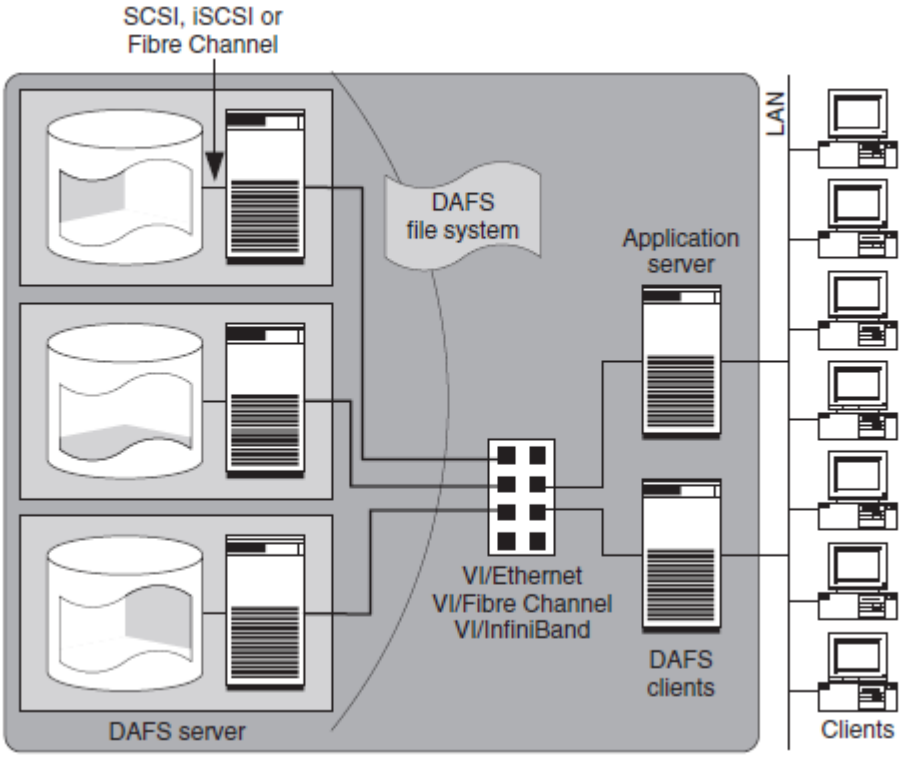


Figure 5.8 A DAFS file system can extend over several DAFS servers. All DAFS servers and DAFS clients are connected via a VI-capable network such as InfiniBand, Fibre Channel or Ethernet.

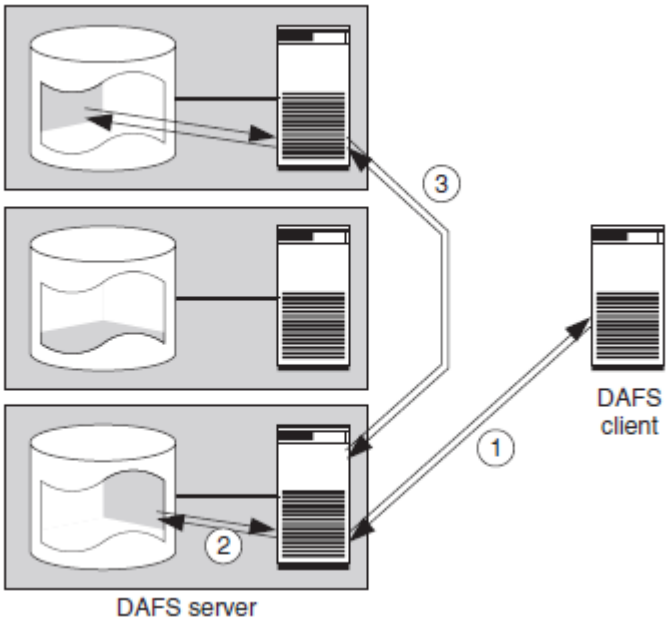


Figure 5.9 The DAFS client communicates with just one DAFS server (1). This processes file access, the blocks of which it manages itself (2). In the case of data that lies on a different DAFS server, the DAFS server forwards the storage access to the corresponding DAFS server, with this remaining hidden from the DAFS client (3).

Two approaches prevail in the discussion about the client implementation. It can either be implemented as a shared library (Unix) or Dynamic Link Library (DLL) (Windows) in the user space or as a kernel module (Figure 4.10). In the user space variant – known as uDAFS – the DAFS library instructs the kernel to set up an exclusive end-to-end connection with the DAFS server for each system call (or for each API call under Windows) by means of a VI provider layer (VIPL), which is also realised as a library in user space. The VI-capable NIC (VI-NIC) guarantees the necessary protection against accesses or faults caused by other processes. The user space implementation can utilise the full potential of DAFS to increase the I/O performance because it completely circumvents the kernel.

It offers the application explicit control over the access of the NIC to its private storage area. Although control communication takes place between the VIPL in the user space and the VI-NIC driver in the kernel, the CPU cost that this entails can be disregarded due

to the low data quantities.

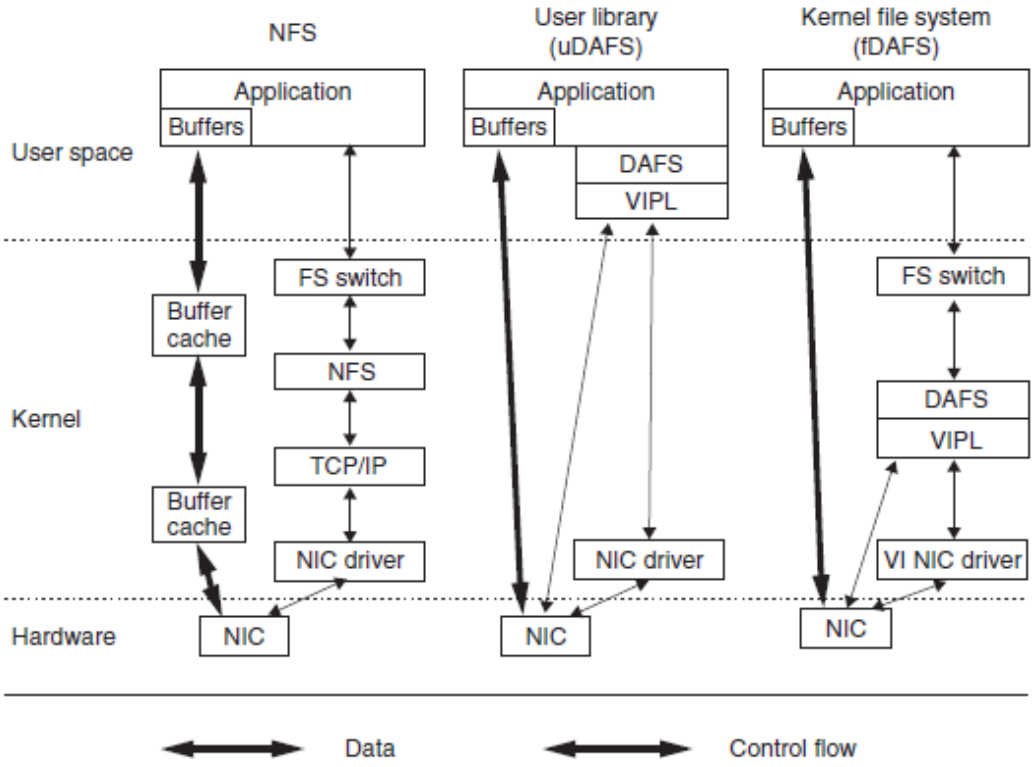


Figure 5.10 A comparison between NFS, uDAFS and fDAFS.

The disadvantage of the methods is the lack of compatibility with already existing applications, which without exception require an upgrade in order to use DAFS. Such a cost is only justified for applications for which a high I/O performance is critical. In the second, kernel-based variant the implementation is in the form of a loadable file system module (fDAFS) underneath the Virtual File System (VFS layer) for Unix or as an Installable File System (IFS) for Windows. Each application can address the file system driver as normal by means of the standard system calls and VFS or API calls and the I/O manager; DAFS then directs a query to the DAFS server. The I/O performance is slowed due to the fact that all file system accesses run via the VFS or the I/O manager because this requires additional process changes between user and kernel processes. On the other
hand, there is compatibility with all applications.

Some proponents of DAFS claim to have taken measurements in prototypes showing that data access over DAFS is quicker than data access on local hard disks. In our opinion this comparison is dubious. The DAFS server also has to store data to hard disks. We find it barely conceivable that disk access can be quicker on a DAFS server than on a conventional file server. Nevertheless, DAFS-capable NAS servers could potentially support I/O-intensive applications such as databases, batch processes or multi-media applications. Integration with RDMA makes it irrelevant whether the file accesses take place via a network. The separation between databases and DAFS servers even has the advantage that the address conversion of files to SCSI block addresses is offloaded from the database server to the DAFS server, thus reducing the load on the database server's CPU.

However, file servers and database servers will profit equally from InfiniBand, VI and RDMA. There is therefore the danger that a DAFS server will only be able to operate very few databases from the point of view of I/O, meaning that numerous DAFS servers may have to be installed. A corresponding number of DAFS-capable NAS servers could be installed comparatively quickly. However, the subsequent administrative effort could be considerably greater.

The development of DAFS was mainly driven by the company Network Appliance, a major NAS manufacturer. The standardisation of the DAFS protocol for communication between server and client and of the DAFS API for the use of file systems by applications took place under the umbrella of the DAFS Collaborative. Its website www.dafscollaborative can no longer be reached. Since the adoption of Version 1.0 in September 2001 (protocol) and November 2001 (API) the standardisation of DAFS has come to a standstill. Originally, DAFS was submitted as an Internet standard to the Internet Engineering Task Force (IETF) in September 2001; however, it has found very little support in the storage industry. Instead, widespread attention is being given to an alternative, namely an extension of NFS with RDMA as a transport layer and the addition of DAFS-like lock semantics.

DAFS is an interesting approach to the use of NAS servers as storage for I/O-intensive applications. Due to a lack of standardisation and widespread support across the industry, current DAFS offerings (2009) should be considered as a temporary solution for specific environments until alternatives like NFS over RDMA and CIFS over RDMA emerge. Furthermore, iSCSI is of interest to those who see DAFS as a way of avoiding an investment in

Fibre Channel, the more so because iSCSI – just like NFS and CIFS – can benefit from TOEs, the SDP and a direct mapping of iSCSI on RDMA (iSER) (Section 3.6.3). Shared-disk file systems (Section 4.3) also offer a solution for high-speed file sharing and in addition to that, storage virtualisation (Chapter 5) provides high-speed file sharing while it addresses the increasingly expensive management of storage and storage networks as well.

5.3 SHARED DISK FILE SYSTEMS

The greatest performance limitation of NAS servers and self-configured file servers is that each file must pass through the internal buses of the file servers twice before the files arrive at the computer where they are required (Figure 4.7). Even DAFS and its alternatives like NFS over RDMA cannot get around this 'eye of the needle'. With storage networks it is possible for several computers to access a storage device simultaneously. The I/O bottleneck in the file server can be circumvented if all clients fetch the files from the disk directly via the storage network (Figure 4.11).

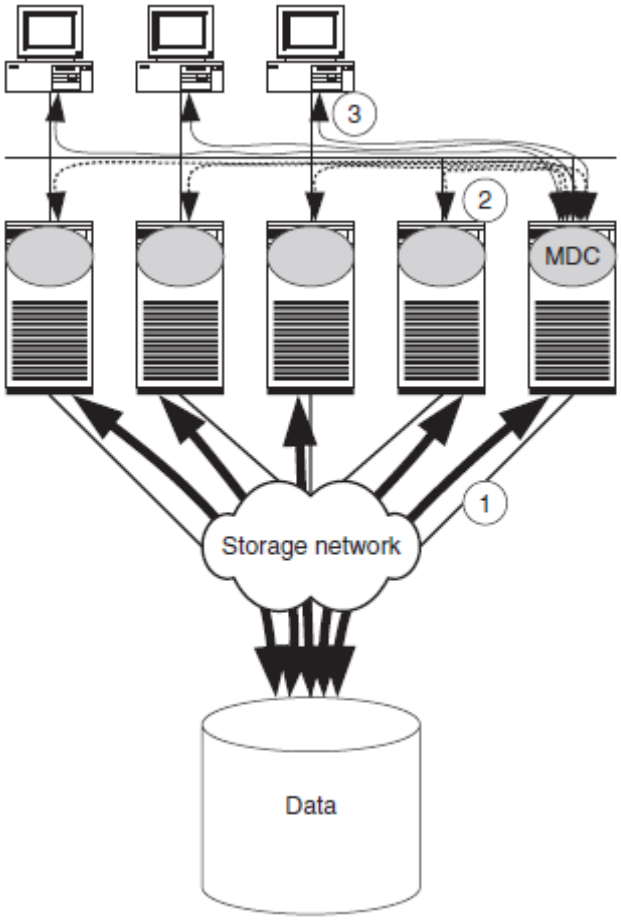


Figure 4.11 In a shared disk file system all clients can access the disks directly via the storage network (1). LAN data traffic is now only necessary for the synchronisation of the write accesses (2). The data of a shared disk file system can additionally be exported over the LAN in the form of a network file system with NFS or CIFS (3).

The difficulty here: today's file systems consider their storage devices as local. They concentrate upon the caching and the aggregation of I/O operations; they increase performance by reducing the number of disk accesses needed. So-called shared disk file systems can deal with this problem. Integrated into them are special algorithms that synchronise the simultaneous accesses of several computers to common disks. As a result, shared disk file systems make it possible for several computers to access files simultaneously without causing version conflict.

To achieve this, shared disk file systems must synchronise write accesses in addition to the functions of local file systems. It should be ensured locally that new files are written to

different areas of the hard disk. It must also be ensured that cache entries are marked as invalid. Let us assume that two computers each have a file in their local cache and one of the computers changes the file. If the second computer subsequently reads the file again it may not take the now invalid copy from the cache.

The great advantage of shared disk file systems is that the computers accessing files and the storage devices in question now communicate with each other directly. The diversion via a central file server, which represents the bottleneck in conventional network file systems and also in DAFS and RDMA-enabled NFS, is no longer necessary.

In addition, the load on the CPU in the accessing machine is reduced because communication via Fibre Channel places less of a load on the processor than communication via IP and Ethernet. The sequential access to large files can thus more than make up for the extra cost for access synchronisation. On the other hand, in applications with many small files or in the case of many random accesses within the same file, we should check whether the use of a shared disk file system is really worthwhile.

One side-effect of file sharing over the storage network is that the availability of the shared disk file system can be better than that of conventional network file systems. This is because a central file server is no longer needed. If a machine in the shared disk file system cluster fails, then the other machines can carry on working. This means that the availability of the underlying storage devices largely determines the availability of shared
disk file systems.

5.3.1 Case study: The General Parallel File System (GPFS)

We have decided at this point to introduce a product of our employer, IBM, for once. The General Parallel File System (GPFS) is a shared disk file system that has for many years been used on cluster computers of type RS/6000 SP (currently IBM eServer Cluster 1600). We believe that this section on GPFS illustrates the requirements of a shared disk file system very nicely. The reason for introducing GPFS at this point is quite simply that
it is the shared disk file system that we know best. The RS/6000 SP is a cluster computer. It was, for example, used for Deep Blue, the computer that beat the chess champion Gary Kasparov. An RS/6000 SP consists of up to 512 conventional AIX computers that can also be

connected together via a so-called high performance switch (HPS). The individual computers of an RS/6000 SP are also called nodes.

Originally GPFS is based upon so-called Virtual Shared Disks (VSDs) (Figure 4.12). The VSD subsystem makes hard disks that are physically connected to a computer visible to other nodes of the SP. This means that several nodes can access the same physical hard disk. The VSD subsystem ensures that there is consistency at block level, which means that a block is either written completely or not written at all. From today's perspective we

could say that VSDs emulate the function of a storage network. In more recent versions of GPFS the VSD layer can be replaced by an Serial Storage Architecture (SSA) SAN or a Fibre Channel SAN.

GPFS uses the VSDs to ensure the consistency of the file system, i.e. to ensure that the metadata structure of the file system is maintained. For example, no file names are allocated twice. Furthermore, GPFS realises some RAID functions such as the striping and mirroring of data and metadata.

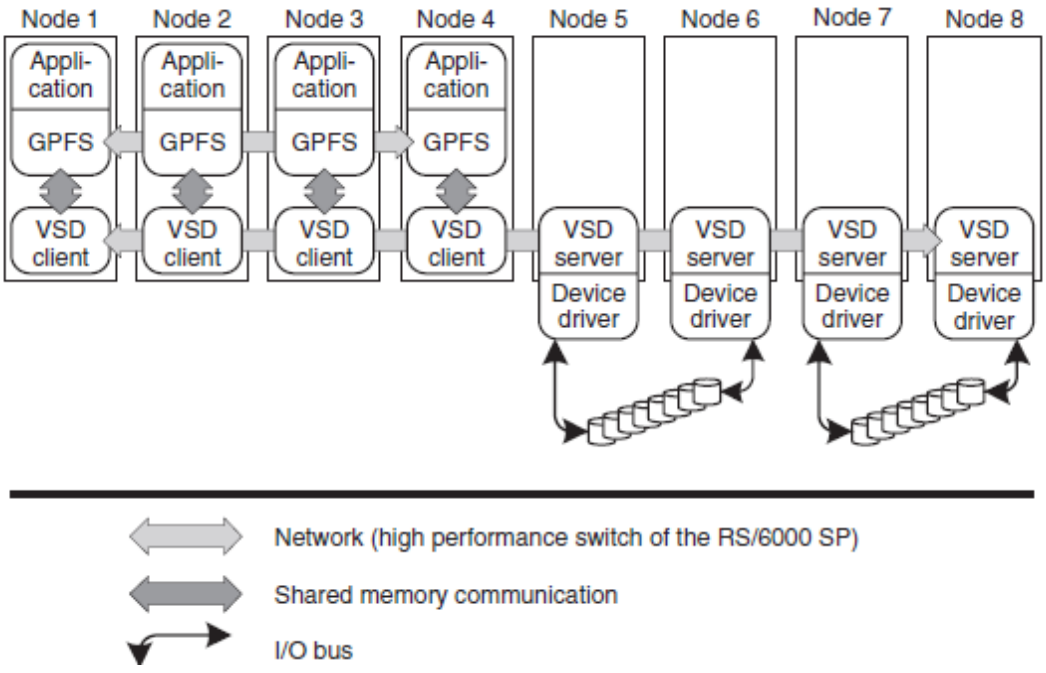


Figure 5.12 Applications see the GPFS file system like a local file system. The GPFS file system itself is a distributed application that synchronises parallel accesses. The VSD subsystem permits access to hard disks regardless of where they are physically connected.

Figure 4.12 illustrates two benefits of shared disk file systems. First, they can use RAID 0 to stripe the data over several hard disks, host bus adapters and even disk subsystems, which means that shared disk file systems can achieve a very high throughput All applications that have at least a partially sequential access pattern profit from this.

Second, the location of the application becomes independent of the location of the data. In Figure 4.12 the system administrator can start applications on the four GPFS nodes that have the most resources (CPU, main memory, buses) available at the time. A so-called workload manager can move applications from one node to the other depending upon load. In conventional file systems this is not possible. Instead, applications have to run on the nodes on which the file system is mounted since access via a network file system such as NFS or CIFS is generally too slow.

The unusual thing about GPFS is that there is no individual file server. Each node in the GPFS cluster can mount a GPFS file system. For end users and applications the GPFS file system behaves – apart from its significantly better performance – like a conventional
local file system.

GPFS introduces the so-called node set as an additional management unit. Several node sets can exist within a GPFS cluster, with a single node only ever being able to belong to a maximum of one node set (Figure 4.13). GPFS file systems are only ever visible within a node set. Several GPFS file systems can be active in every node set.

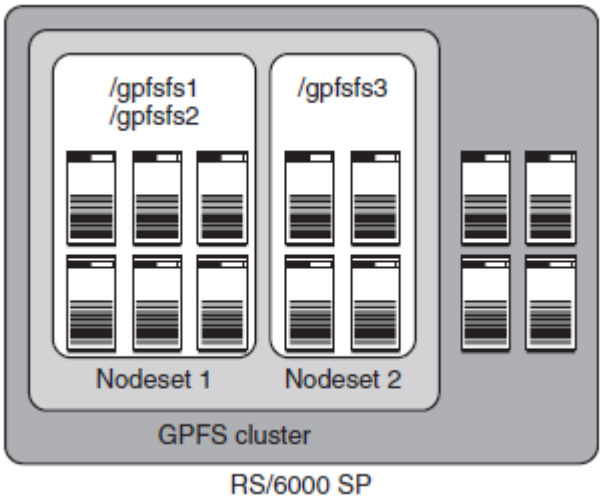


Figure 5.13 GPFS introduces the node sets as an additional management unit. GPFS file systems are visible to all nodes of a node set.

The GPFS Daemon must run on every node in the GPFS cluster. GPFS is realised as distributed application, with all nodes in a GPFS cluster having the same rights and duties. In addition, depending upon the configuration of the GPFS cluster, the GPFS Daemon must take on further administrative functions over and above the normal tasks of a file system.

In the terminology of GPFS the GPFS Daemon can assume the following roles:

• *Configuration Manager*

In every node set one GPFS Daemon takes on the role of the Configuration Manager. The Configuration Manager determines the File System Manager for every file system and monitors the so-called quorum. The quorum is a common procedure in distributed systems that maintains the consistency of the distributed application in the event of a network split. For GPFS more than half of the nodes of a node set must be active. If the quorum is lost in a node set, the GPFS file system is automatically deactivated (unmount) on all nodes of the node set.

• *File System Manager*

Every file system has its own File System Manager. Its tasks include the following:

– Configuration changes of the file system

– Management of the blocks of the hard disk

- Token administration

– Management and monitoring of the quota and

- Security services

Token administration is particularly worth highlighting. One of the design objectives of GPFS is the support of parallel applications that read and modify common files from different nodes. Like every file system, GPFS buffers files or file fragments in order to increase performance. GPFS uses a token mechanism in order to synchronise the cache entries on various computers in the event of parallel write and read accesses (Figure 4.14). However, this synchronisation only ensures that GPFS behaves precisely in the same way as a local file system that can only be mounted on one computer. This means that in GPFS – as in every file system – parallel applications still have to synchronise the accesses to common files, for example, by means of locks.

• *Metadata Manager*

Finally, one GPFS Daemon takes on the role of the Metadata Manager for every open file. GPFS guarantees the consistency of the metadata of a file because only the Metadata Manager may change a file's metadata. Generally, the GPFS Daemon of the node on which the file has been open for the longest is the Metadata Manager for the file. The assignment of the Metadata Manager of a file to a node can change in relation to the access behaviour of the applications.

The example of GPFS shows that a shared disk file system has to achieve a great deal more than a conventional local file system, which is only managed on one computer. GPFS has been used successfully on the RS/6000 SP for several years. The complexity of shared disk file systems is illustrated by the fact that IBM only gradually transferred the GPFS file system to other operating systems such as Linux, which is strategically supported by IBM, and to new I/O technologies such as Fibre Channel and iSCSI.

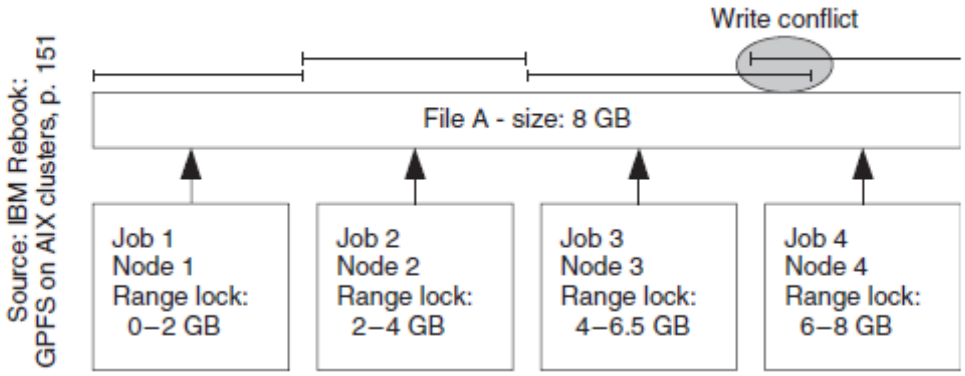


Figure 5.14 GPFS synchronises write accesses for file areas. If several nodes request the token for the same area, GPFS knows that it has to synchronise cache entries.

5.4 COMPARISON: FIBRE CHANNEL SAN, FCoE SAN, iSCSI SAN AND NAS

Fibre Channel SAN, FCoE SAN, iSCSI SAN and NAS are four techniques with which storage networks can be realised. Figure 4.15 compares the I/O paths of the four techniques and Table 4.2 summarises the most important differences.

In contrast to NAS, in Fibre Channel, FCoE and iSCSI the data exchange between servers and storage devices takes place in a block-based fashion. Storage networks are more difficult to configure. On the other hand, Fibre Channel at least supplies optimal performance for the data exchange between server and storage device.

NAS servers, on the other hand, are turnkey file servers. They can only be used as file servers, but they do this very well. NAS servers have only limited suitability as data storage for databases due to lack of performance. Storage networks can be realised with NAS servers by installing an additional LAN between NAS server and the application servers (Figure 4.16). In contrast to Fibre Channel, FCoE and iSCSI this storage network

transfers files or file fragments.

One supposed advantage of NAS is that NAS servers at first glance have a higher pre-fetch hit rate than disk subsystems connected via Fibre Channel, FCoE or iSCSI (or just SCSI). However, it should be borne in mind that NAS servers work at file system level and disk subsystems only at block level. A file server can move the blocks of an opened file from the hard disk into the main memory and thus operate subsequent file accesses more quickly from the main memory.

Disk subsystems, on the other hand, have a pre-fetch hit rate of around 40% because they only know blocks; they do not know how the data (for example, a file system or database) is organised in the blocks. A self-configured file server or a NAS server that uses hard disks in the storage network can naturally implement its own pre-fetch strategy in addition to the pre-fetch strategy of the disk subsystem and, just like a NAS server, achieve a pre-fetch hit rate of 100%. Today (2009) Fibre Channel, iSCSI and NAS have been successfully implemented in production environments. It is expected the FCoE will enter the market in 2009. Fibre Channel satisfies the highest performance requirements – it is currently (2009) the only transmission technique for storage networks that is suitable for I/O intensive databases.

Since 2002, iSCSI has slowly been moving into production environments. It is said that iSCSI is initially being used for applications with low or medium performance requirements.

It remains to be seen in practice whether iSCSI also satisfies high performance requirements (Section 3.5.2) or whether FCoE establishes as *the* technology for storage networks based on Ethernet. NAS is excellently suited to web servers and for the file sharing of work groups. With RDMA-enabled NFS and CIFS, NAS could also establish itself as a more convenient data store for databases.

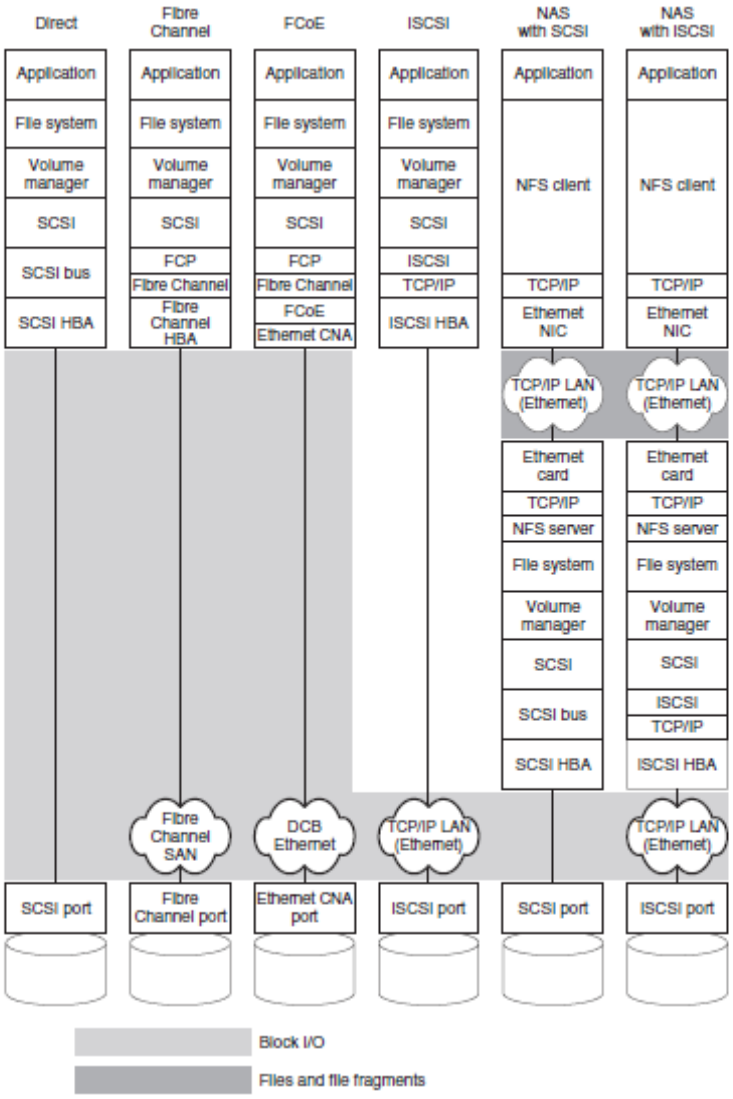


Figure 5.15 Comparison of the different I/O paths of SCSI, iSCSI, Fibre Channel, FCoE and NAS.

Table 4.2 Comparison of Fibre Channel, iSCSI and NAS.

	Fibre channel	FCoE	iSCSI	NAS
Protocol	FCP (SCSI)	FCP (SCSI)	iSCSI (SCSI)	NFS, CIFS, HTTP
Network	Fibre Channel	DCB Ethernet	TCP/IP	TCP/IP
Source/target	Server/storage device	Server/storage device	Server/storage device	Client/NAS server, application server/NAS server
Transfer objects	Device blocks	Device blocks	Device blocks	Files, file fragments
Access via the storage device	Directly via Fibre Channel	Directly via Fibre Channel	Directly via iSCSI	Indirectly via the NAS-internal computer
Embedded file system	No	No	No	Yes
Pre-fetch hit rate	40%	40%	40%	100%
Configuration	By end user (flexible)	By end user (flexible)	By end user (flexible)	Preconfigured by NAS manufacturers (Plug&Play)
Suitability for databases	Yes	Yes	To a limited degree (2009)	To a limited degree
Production-readiness	Yes	Market entry in 2009	Yes	Yes

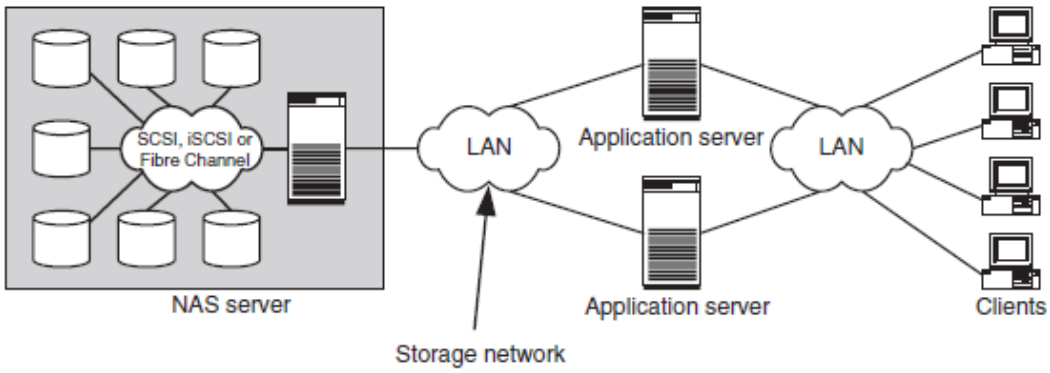


Figure 5.16 For performance reasons a separate LAN, which serves as a storage network, is installed here between the NAS server and the application servers.

UNIT 6

STORAGE VIRTUALISATION

Although the cost of storage has fallen considerably in recent years, at the same time the need for storage has risen immensely, so that we can observe of a real data explosion. The administrative costs associated with these quantities of data should not, however, increase to the same degree. The introduction of storage networks is a first step towards remedying the disadvantages of the server-centric IT architecture (Section 1.1). Whereas in smaller environments the use of storage networks is completely adequate for the mastery of data, practical experience has shown that, in large environments, a storage network alone is not sufficient to efficiently manage the ever-increasing volumes of data.

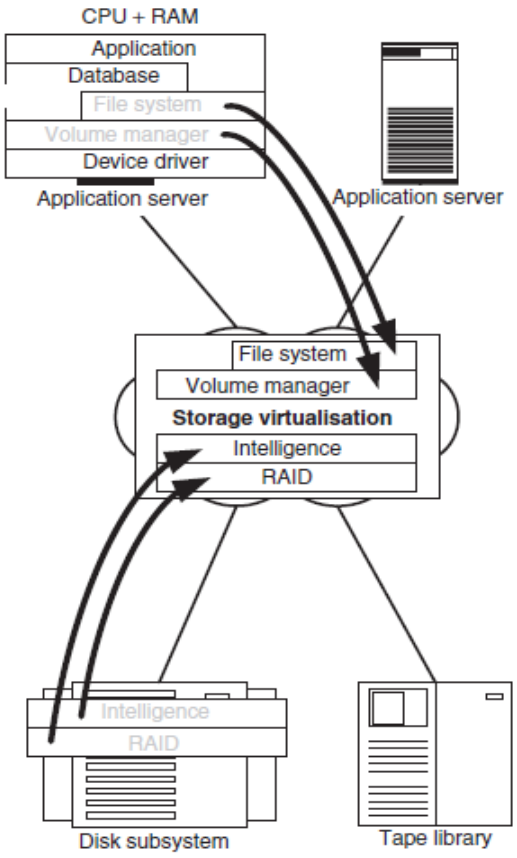


Figure 6.1 Storage virtualisation in the storage network moves virtualisation functions from servers and storage devices into the storage network.

This creates a new virtualisation entity which, as a result of its central position in the storage network, spans all servers and storage systems and can thus centrally manage all available storage resources.

6.3 DEFINITION OF STORAGE VIRTUALISATION

The term ‘storage virtualisation’ is generally used to mean the separation of the storage into the physical implementation level of the storage devices and the logical representation level of the storage for use by operating systems, applications and users. In the following we will also use the term ‘virtualisation’, i.e. dropping the word ‘storage’. This is always used in the sense of the above definition of storage virtualisation. Various uses of the term ‘storage virtualisation’ and ‘virtualisation’ are found in the literature depending upon which level of the storage network the storage virtualisation takes place on.

The various levels of the storage network here are the server, the storage devices and the network. Some authors only speak of storage virtualisation if they explicitly mean storage virtualisation within the network. They use the term virtualisation, on the other hand, to mean the storage virtualisation in the storage devices (for example, in the disk subsystems) or on servers (such as in a volume manager). However, these different types

of storage virtualisation are not fundamentally different. Therefore, we do not differentiate between the two terms and always use ‘storage virtualisation’ and ‘virtualisation’ in the sense of the above definition.

In Section 5.1 we revised various types of storage virtualisation on the various levels of the storage network and we will pick these up again later. First of all, however, we want to deal in detail with the conceptual realisation of storage virtualisation. Storage virtualisation inserts – metaphorically speaking – an additional layer between storage devices and storage users (Figure 5.11). This forms the interface between virtual and physical storage, by mapping the physical storage onto the virtual and conversely the virtual storage onto the physical. The separation of storage into the physical

implementation level and the logical representation level is achieved by abstracting the physical storage to the logical storage by aggregating several physical storage units to form one or more logical, so-called virtual, storage units. The operating system or applications no longer have direct access to the physical storage devices, they use exclusively the virtual storage. Storage accesses to the physical storage resources take place independently and separately from the storage accesses to the virtual storage resources.

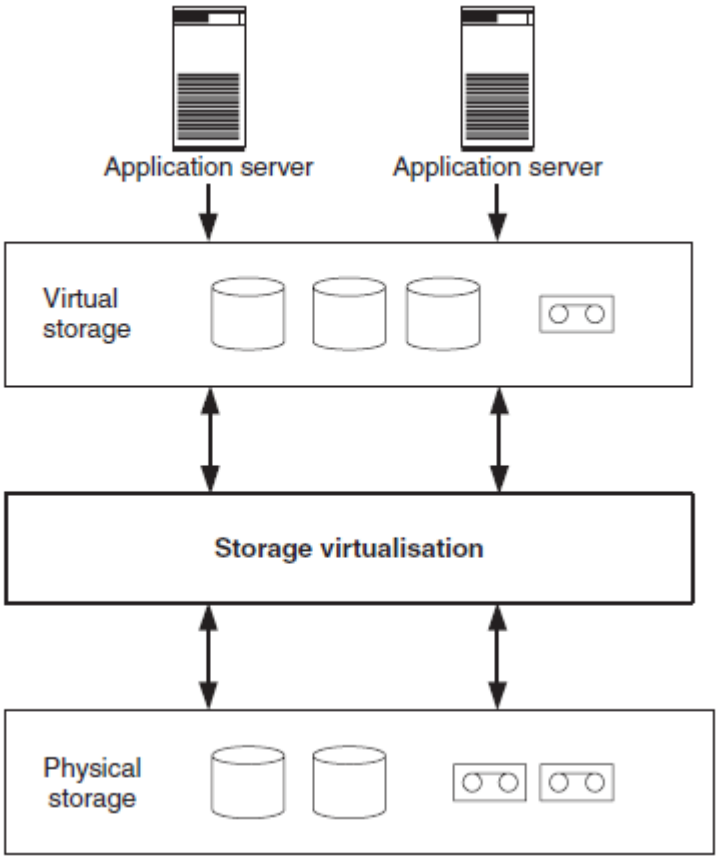


Figure 6.11 In storage virtualisation an additional layer is inserted between the storage devices and servers. This forms the interface between virtual and physical storage.

For example, the physical hard disks available on a disk stack (JBOD) are brought together by the volume manager of a server to form a large logical volume. In this manner the volume manager thus forms an additional layer between the physical

disks of the disk stack and the logical and thus virtual volume with which the applications (e.g. file systems and databases) of the server work. Within this layer, the mapping of physical hard disks onto logical volumes and vice versa is performed. This means that storage virtualisation always calls for a virtualisation entity that maps from virtual to physical storage and vice versa. On the one hand it has to make the virtual storage available to the operating system, the applications and the users in usable form and, on the other, it has to realise data accesses to the physical storage medium. This entity can be implemented both as hardware and software on the various levels in a storage network. It is also possible for several virtualisation entities to be used concurrently. For example, an application can use the virtualised volume of a volume manager on server level, which for its part is formed from a set of virtualised volumes which are exported by one or more disk subsystems (Section 5.1).

6.4 IMPLEMENTATION CONSIDERATIONS

In the following we want to draw up general requirements and considerations for the implementation of the virtualisation entity and illustrate how the difficulties described in Section 5.2 can be solved with the aid of storage virtualisation. For example, storage virtualisation also facilitates the integration of higher storage functions that previously had to be realised by means of other software products.

6.4.1 Realisation of the virtualisation entity

First of all, it is important that a storage virtualisation entity can be administered from a central console regardless of whether it is implemented as hardware or software and where it is positioned in the storage network. It is desirable for all tools that are required for the administration of the storage device to run via this console. All operations performed by the virtualisation entity should take place in a rule-based manner and orientate themselves to the applicable data profiles. Policy-based operation allows the storage administrator to configure and control the operations of the virtualisationentity. Profile-orientation makes it possible for the data to be automated according to its specific properties and requirements.

Because virtualisation always intervenes in the data stream, correct implementation is indispensable if data corruption is to be avoided. The virtualisation entity itself should therefore also be backed up so that access to the virtualised storage resources is still possible in the event of a failure. The concepts for server-clustering introduced in Section
6.3.2 are suitable here.

In order to achieve the greatest possible degree of compatibility to servers and applications and also to win acceptance amongst users it is necessary for a virtualization entity to remain hidden from its users. Servers, applications and users must always have the impression that they are working with physical storage media and must not notice the existence of a virtualisation entity. Furthermore, for reasons of compatibility, access on both file and block level, including the required protocols, must be supported (Section 5.5).

In addition to virtualised storage, the classical non-virtualised storage access options should continue to exist. This facilitates first of all an incremental introduction of the virtualisation technique into the storage network and second, allows applications, servers and storage devices that are incompatible with virtualisation to continue to be operated in the same storage network. For example, in our practical work during the testing of virtualisation software we found that the connection of a Windows server functioned perfectly, whilst the connection of a Solaris server failed. This was because of minor deviations from the Fibre Channel standard in the realisation of the Fibre Channel protocol in the virtualisation software used.

6.4.2 Replacement of storage devices

When using storage virtualisation the replacement of storage devices is relatively easy to perform, since the servers no longer access the physical devices directly, instead only working with virtual storage media. The replacement of a storage device in this case involves the following steps:

1. Connection of the new storage device to the storage network.
2. Configuration and connection of the new storage device to the virtualisation entity.

3. Migration of the data from the old to the new device by the virtualisation entity whilst the applications are running.
4. Removal of the old storage device from the configuration of the virtualisation entity.
5. Removal of the old storage device from the storage network.

The process requires no configuration changes to the applications. These continue to work on their virtual hard disks throughout the entire process.

6.4.3 Efficient use of resources by dynamic storage allocation

Certain mechanisms, such as the insertion of a volume manager within the virtualization entity, permit the implementation of various approaches for the efficient use of resources.

First, all storage resources can be shared. Furthermore, the virtualisation entity can react dynamically to the capacity requirements of virtual storage by making more physical capacity available to a growing data set on virtual storage and, in the converse case, freeing up the storage once again if the data set shrinks. Such concepts can be more easily developed on the file level than on the block level since on the file level a file system holds the information on unoccupied blocks, whereas on the block level this information is lacking. Even if such concepts have not previously been realised, they can be realised with storage virtualisation.

In this manner it is possible to practically imitate a significantly larger storage, of which only part is actually physically present. By the dynamic allocation of the physical storage,

additional physical storage can be assigned to the virtual storage when needed. Finally by dynamic, data-oriented storage allocation it is possible to achieve a more efficient utilisation of resources.

6.4.4 Efficient use of resources by data migration

If a virtualisation entity is oriented towards the profiles of the data that it administers, it can determine which data is required and how often. In this manner it is possible to control the distribution of the data on fast and slow storage devices in order to achieve

a high data throughput for frequently required data. Such data migration is also useful if it is based upon the data type. In the case of video data, for example, it can be worthwhile to store only the start of the file on fast storage in order to provide users with a short insight into the video file. If the user then accesses further parts of the video file that are not on the fast storage, this must first be played back from the slower to the fast storage.

6.4.5 Performance increase

Performance can be increased in several ways with the aid of storage virtualisation. First of all, caching within the virtualisation entity always presents a good opportunity for reducing the number of slow physical accesses (Section 2.6). Techniques such as striping or mirroring within the virtualisation entity for distributing the data over several resources can also be used to increase performance (Section 2.5). Further options for increasing performance are presented by the distribution of the I/O load amongst several virtualisation entities working together and amongst several datapaths between server and virtual storage or virtual storage and physical storage devices (Section 5.1).

6.4.6 Availability due to the introduction of redundancy

The virtualisation entity can ensure the redundancy of the data by itself since it has complete control over the resources. The appropriate RAID techniques are suitable here (Section 2.5). For example, in the event of the failure of a storage device, operation can nevertheless be continued. The virtualisation entity can then immediately start to mirror the data once again in order to restore the redundancy of the data. As a result, a device failure is completely hidden from the servers – apart from possible temporary reductions

in performance. It is even more important that information about a device failure is reported to a central console so that the device is replaced immediately. The message arriving at the console can also be forwarded by e-mail or pager to the responsible person (Section 10.3). Multiple access paths, both between servers and virtual storage and also between virtual storage and physical storage devices can also

contribute to the improvement of fault-tolerance in storage virtualisation (Section 6.3.1).

6.4.7 Backup and archiving

A virtualisation entity is also a suitable data protection tool. By the use of appropriate rules the administrator can, for example, define different backup intervals for different data. Since the virtualisation entity is responsible for the full administration of the physical storage it can perform the backup processes in question independently. All network backup methods (Chapter 7) can be integrated into storage virtualisation.

6.4.8 Data sharing

Data sharing can be achieved if the virtualisation entity permits access to the virtual storage on file level. In this case, the virtualisation entity manages the file system centrally. By means of appropriate protocols, the servers can access the files in this file system in parallel. Currently this is permitted primarily by classical network file systems such as NFS and CIFS (Section 4.2) and fast shared disk file systems (Section 4.3). Some manufacturers are working on cross-platform shared disk file systems with the corresponding protocol mechanisms that permit the fast file sharing even in heterogeneous environments.

6.4.9 Privacy protection

The allocation of user rights and access configurations can also be integrated into a virtualisation entity, since it forms the interface between virtual and physical storage and thus prevents direct access to the storage by the user. In this manner, the access rights of the data can be managed from a central point.

6.5 STORAGE VIRTUALISATION ON BLOCK OR FILE LEVEL

In Section 5.3 we saw that the virtualisation of storage requires an entity that maps between virtual and physical storage and vice versa. The virtualisation entity can be located on the servers (for example, in the form of a volume manager), on the storage

devices (for example, in a disk subsystem) or in the network (for example, as a special device).

Regardless of which level of the storage network (server, network or storage device) the virtualisation entity is located on, we can differentiate between two basic types of virtualisation: virtualisation on block level and virtualisation on file level. Virtualisation on block level means that storage capacity is made available to the operating system or the applications in the form of virtual disks (Figure 5.12). Operating system and applications on the server then work to the blocks of this virtual disk. To this end, the blocks are managed as usual – like the blocks of a physical disk – by a file system or by a database on the server. The task of the virtualisation entity is to map these virtual blocks to the physical blocks of the real storage devices. It can come about as part of this process that the physical blocks that belong to the virtual blocks of a file in the file system of the operating system are stored on different physical storage devices or that they are virtualised once more by a further virtualisation entity within a storage device.

By contrast, virtualisation on file level means that the virtualisation entity provides virtual storage to the operating systems or applications in the form of files and directories (Figure 5.13). In this case, the applications work with files instead of blocks and the conversion of the files to virtual blocks is performed by the virtualisation entity itself. The physical blocks are presented in the form of a virtual file system and not in the form of virtual blocks. The management of the file system is shifted from the server to the virtualisation entity.

To sum up, virtualisation on block or file level can be differentiated as follows: in virtualisation on block level, access to the virtual storage takes place by means of blocks, in virtualisation on file level it takes place by means of files. In virtualisation on block level the task of file system management is the responsibility of the operating system or the applications, whereas in virtualisation on file level this task is performed by the virtualisation entity.

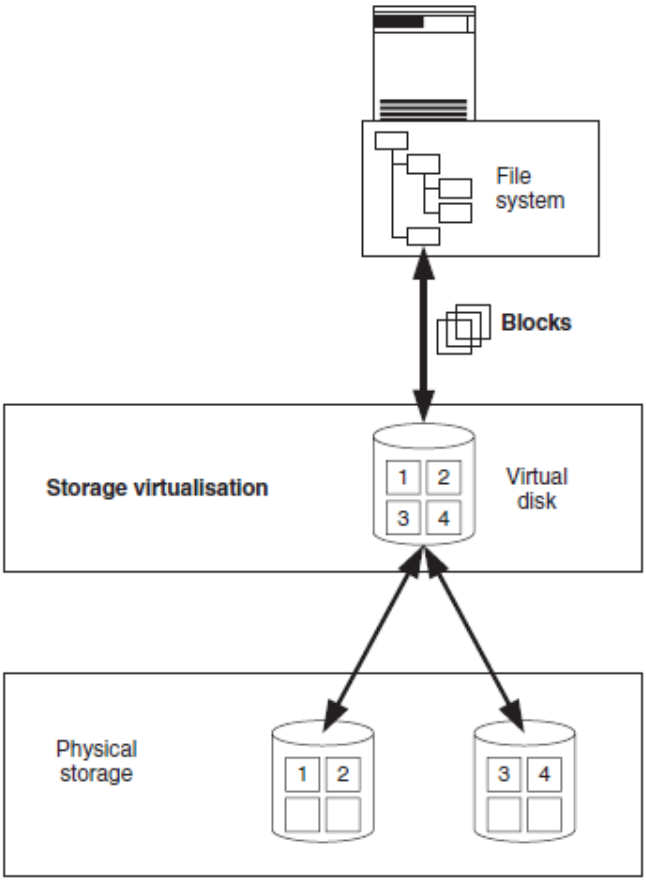


Figure 6.12 In virtualisation on block level the virtualisation entity provides the virtual storage to the servers in the form of a virtual disk.

Virtualisation on block level is suitable if the storage is to be virtualised for as many different operating systems and applications as possible. Virtualisation on block level is actually necessary when dealing with applications that handle their storage access on block level and cannot work on file level. Classic representatives of this category are, for example, databases that can only work with raw devices. Virtualisation on file level, on the other hand, is indispensable for those who want to establish data sharing between several servers. To achieve this, the virtualisation entity must allow several servers access to the same files. This can only be achieved if the file system is implemented in the form of a shared resource as in a network file system (Section 4.2)

or a shared disk file system (Section 4.3) or, just like virtualisation on file level, is held centrally by the virtualisation entity.

In this chapter we constrain the virtualisation on block level to the virtualisation of disks. Later on in Chapter 11 we will expand this approach and discuss the virtualization of removeable media like tapes and opticals.

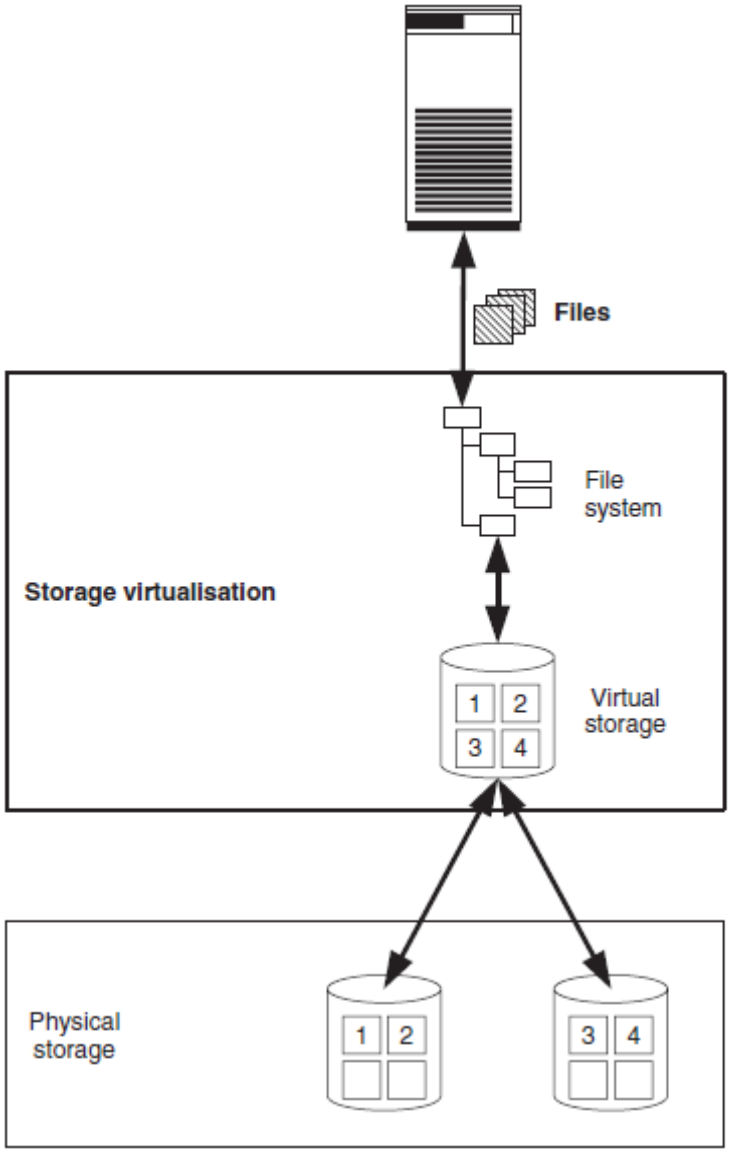


Figure 6.13 In virtualisation on file level the virtualisation entity provides the virtual storage to the servers in the form of files and directories.

6.6 STORAGE VIRTUALISATION ON VARIOUS LEVELS OF THE STORAGE NETWORK

In the following we will concern ourselves with the locations at which a virtualization entity can be positioned in the storage network. The following three levels can be defined here (Figure 5.14): the server (Section 5.6.1), the storage devices (Section 5.6.2) and the network (Section 5.6.3). This will be explained in what follows.

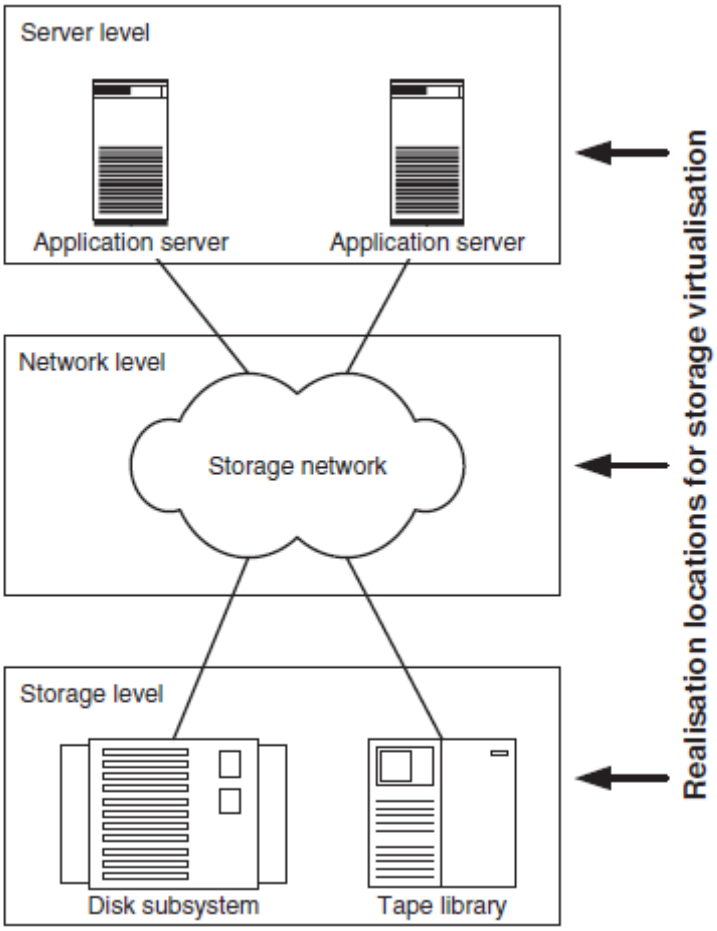


Figure 6.14 A virtualisation entity can be positioned on various levels of the storage network.

6.6.1 Storage virtualisation in the server

A classic representative of virtualisation in the server is the combination of file system and volume manager (Section 4.1.4). A volume manager undertakes the separation of

the storage into logical view and physical implementation by encapsulating the physical hard disk into logical disk groups and logical volumes. These are then made available to the applications via file systems. File systems and databases positioned on the server now work with these logical volumes and cease to work directly with the physical hard disks. Some volume managers additionally have further storage functions such as RAID, snapshots or dynamic reconfiguration options, which permit the addition and removal of storage during operation. With shared disk file systems (Section 4.3) storage virtualization can be expanded to several servers, in order to allow fast file sharing among several servers. These cannot, however, be used in a straightforward manner in heterogeneous environments due to the incompatibilities that prevail. Virtualisation on block level can be performed on a server by the host bus adapter itself. Virtualisation on block level is found, for example, in the use of a RAID controller.

This performs the mapping of the logical blocks that are used by the file system or the volume manager of the operating system to the physical blocks of the various drives.

The benefits of virtualisation on server level are:

• Tried and tested virtualisation techniques are generally used.

• The virtualisation functions can integrate multiple storage systems.

• No additional hardware is required in the storage network to perform the virtualisation.

Thus additional error sources can be ruled out. The approach remains cost-effective.

The disadvantages of a virtualisation on server level are:

• The administration of the storage virtualisation must take place on every single server.

To achieve this, the appropriate software must be installed and maintained upon the computers.

• The storage virtualisation software running on the server can cost system resources and thus have a negative impact upon the server performance.

• Incompatibilities may occur between the virtualisation software and certain applications.

• The virtualisation extends only to those areas of a storage network that are accessible or assigned to those servers running a virtualisation entity.

• The virtualisation only ever takes place on individual servers. This disadvantage can be remedied by complex cluster approaches, which, however, come at an additional administration cost.

6.6.2 Storage virtualisation in storage devices

Virtualisation on block level in storage devices is, for example, found within intelligent disk subsystems (Section 2.7). These storage systems make their storage available to several servers via various I/O channels by means of LUN masking and RAID. The physical hard disks are brought together by the storage devices to form virtual disks, which the servers access using protocols such as SCSI, Fibre Channel FCP, FCoE and iSCSI. In this manner, the mapping of virtual to physical blocks is achieved. Virtualisation on file level in storage devices is, for example, achieved by NAS servers (Section 4.2.2). The file system management is the responsibility of the NAS server. Access by the server to the storage resources takes place on file level by means of protocols such as NFS and CIFS.

The advantages of virtualisation on storage device level are:

• The majority of the administration takes place directly upon the storage device, which is currently perceived as easier and more reliable since it takes place very close to the physical devices.

• Advanced storage functions such as RAID and instant copies are realised directly at the physical storage resources, meaning that servers and I/O buses are not loaded.

• The uncoupling of the servers additionally eases the work in heterogeneous environments since a storage device is able to make storage available to various platforms.

• The servers are not placed under additional load by virtualisation operations.

The disadvantages of virtualisation on storage device level are:

• Configuration and implementation of virtualisation are manufacturer-specific and may thus become a proprietary solution in the event of certain incompatibilities with other storage devices.

• It is very difficult – and sometimes even impossible – to get storage devices from different manufacturers to work together.

• Here too, virtualisation takes place only within a storage system and cannot effectively be expanded to include several such storage devices without additional server software.

6.6.3 Storage virtualisation in the network

Storage virtualisation by a virtualisation entity in the storage network is realised by symmetric or asymmetric storage virtualisation (Section 5.7). First, however, we want to discuss the general advantages and disadvantages of storage virtualisation in the network.

The advantages of virtualisation in the storage network are:

• The virtualisation can extend over the storage devices of various manufacturers.

• The virtualisation is available to servers with different operating systems that are connected to the storage network.

• Advanced storage functions, such as mirroring or snapshots can be used on storage devices that do not themselves support these techniques (for example, JBODs and low cost RAID arrays). • The administration of storage virtualisation can be performed from a central point.

• The virtualisation operations load neither the server nor the storage device.

The disadvantages are:

• Additional hardware and software are required in the storage network.

• A virtualisation entity in the storage network can become a performance bottleneck.

• Storage virtualisation in the storage network is in contrast to other storage technologies currently (2009) still a new product category. Whilst storage virtualisation on the block level has been successfully established in production environments, there is still very limited experience with file level storage virtualisation which is located in the storage networ

6.7 SYMMETRIC AND ASYMMETRIC STORAGE VIRTUALISATION IN THE NETWORK

The symmetric and asymmetric virtualisation models are representatives of storage virtualization in the network. In both approaches it is possible to perform virtualisation both on block and on file level. In both models the virtualisation entity that undertakes the separation between physical and logical storage is placed in the storage network in the

form of a specialised server or a device. This holds all the meta-information needed for the virtualisation. The virtualisation entity is therefore also called the metadata controller.

Its duties also include the management of storage resources and the control of all storage functions that are offered in addition to virtualisation. Symmetric and asymmetric virtualisation differ primarily with regard to their distribution of data and control flow. Data flow is the transfer of the application data between the servers and storage devices. The control flow consists of all metadata and control information necessary for virtualisation between virtualisation entity and storage devices and servers. In symmetric storage virtualisation the data flow and the control flow travel down the same path. By contrast, in asymmetric virtualisation the data flow is separated from the control flow.

6.7.1 Symmetric storage virtualisation

In symmetric storage virtualisation the data and control flow go down the same path (Figure 5.15). This means that the abstraction from physical to logical storage necessary for virtualisation must take place within the data flow. As a result, the metadata controller

is positioned precisely in the data flow between server and storage devices, which is why symmetric virtualisation is also called in-band virtualization.

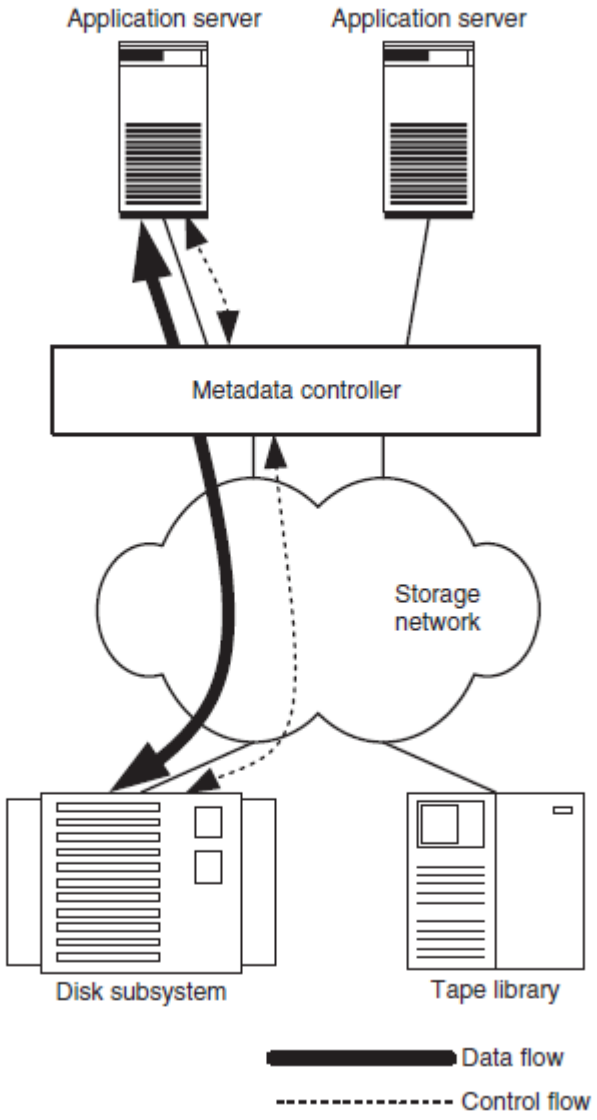


Figure 6.15 In symmetric virtualisation, data and control flow travel down the same path. The abstraction from physical to logical storage takes place within the data stream. In addition to the control of the virtualisation, all data between servers and storage devices now flow through the metadata controller. To this end virtualisation is logically structured in two layers: the layer for the management of the logical volumes and the data access layer (Figure 5.16):

1. The volume management layer is responsible for the management and configuration of the storage devices that can be accessed directly or via a storage network and it provides the aggregation of these resources into logical disks. 2. The

data access layer makes the logical drives available for access either on block or file level, depending upon what degree of abstraction is required. These logical drives can thus be made available to the application servers by means of appropriate protocols. In the case of virtualisation on block level, this occurs in the form of a virtual disk and in the case of virtualisation on file level it takes place in the form of a file system.

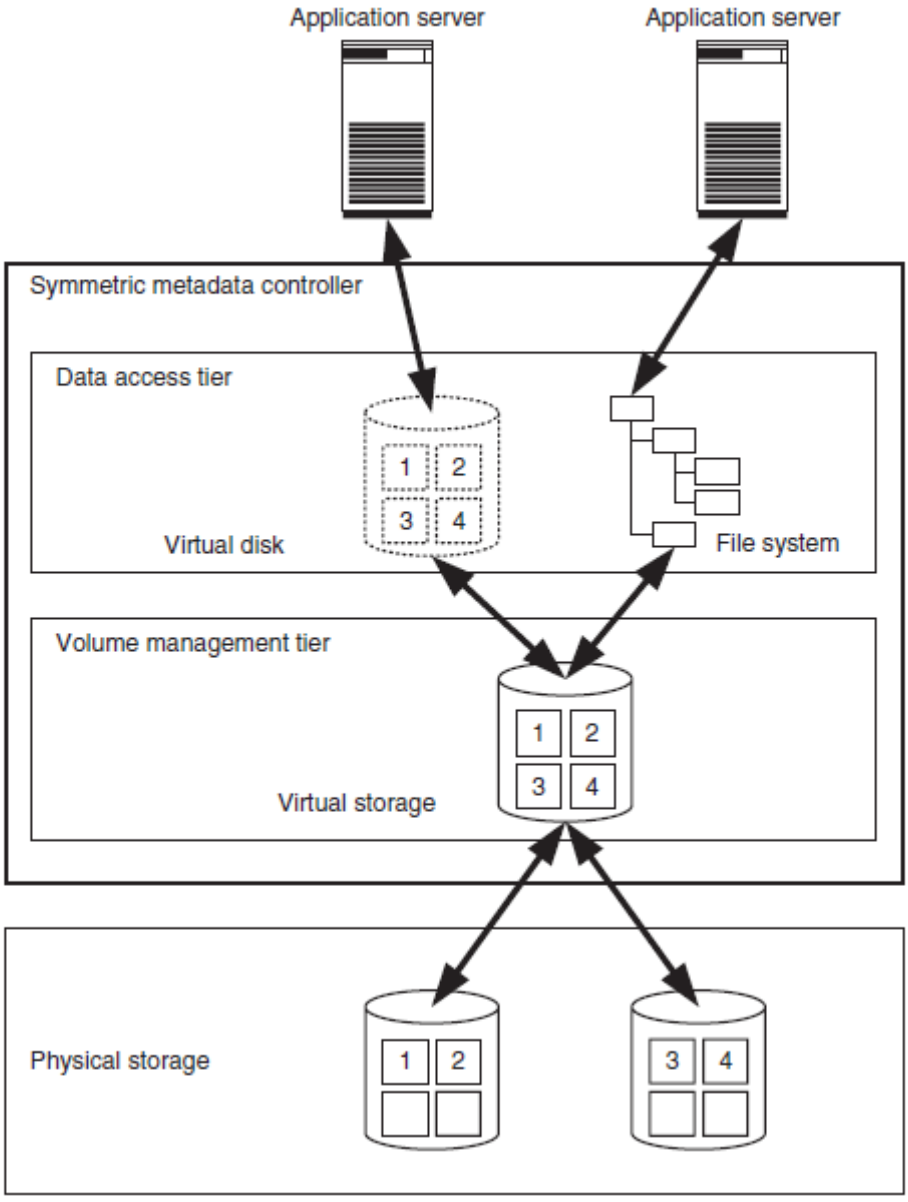


Figure 6.16 In symmetric virtualisation the metadata controller consists of a data access layer and a volume management layer.

In symmetric virtualisation all data flow through the metadata controller, which means that this represents a potential bottleneck. To increase performance, therefore, the metadata controller is upgraded by the addition of a cache. With the use of caching and symmetric virtualisation it is even possible to improve the performance of an existing storage network as long as exclusively write-intensive applications are not used.

A further issue is fault-tolerance. A single metadata controller represents a single point of failure. The use of cluster technology (Section 6.3.2) makes it possible to remove the single point of failure by using several metadata controllers in parallel. In addition, a corresponding load distribution provides a performance increase. However, a configuration failure or a software failure of that cluster can lead to data loss on all virtualised resources.

In the case of a network-based virtualisation spanning several servers and storage devices, this can halt the activity of a complete data centre (Section 6.3.4).

Thus the advantages of symmetric virtualisation are evident:

• The application servers can easily be provided with data access both on block and file level, regardless of the underlying physical storage devices.

• The administrator has complete control over which storage resources are available to which servers at a central point. This increases security and eases the administration.

• Assuming that the appropriate protocols are supported, symmetric virtualisation does not place any limit on specific operating system platforms. It can thus also be used in heterogeneous environments.

• The performance of existing storage networks can be improved by the use of caching and clustering in the metadata controllers.

• The use of a metadata controller means that techniques such as snapshots or mirroring can be implemented in a simple manner, since they control the storage

access directly. They can also be used on storage devices such as JBODs or simple RAID arrays that do not provide to these techniques themselves.

The disadvantages of a symmetric virtualisation are:

• Each individual metadata controller must be administered. If several metadata controllers are used in a cluster arrangement, then the administration is relatively complex and time-consuming particularly due to the cross-computer data access layer. This disadvantage can, however, be reduced by the use of a central administration console for the metadata controller.

• Several controllers plus cluster technology are indispensable to guarantee the fault-tolerance of data access.

• As an additional element in the data path, the controller can lead to performance problems, which makes the use of caching or load distribution over several controllers indispensable.

• It can sometimes be difficult to move the data between storage devices if this is managed by different metadata controllers.

6.7.2 Asymmetric storage virtualisation

In contrast to symmetric virtualisation, in asymmetric virtualisation the data flow is separated from the control flow. This is achieved by moving all mapping operations from

logical to physical drives to a metadata controller outside the data path (Figure 5.17). The metadata controller now only has to look after the administrative and control tasks of virtualisation, the flow of data takes place directly from the application servers to the storage devices. As a result, this approach is also called out-band virtualisation.

The communication between metadata controller and agents generally takes place via the LAN (out-band) but can also be realised in-band via the storage network. Hence, in our opinion the terms ‘in-band virtualisation’ and ‘out-band virtualisation’ are a little misleading. Therefore, we use instead the terms ‘symmetric virtualisation’ and ‘asymmetric virtualisation’ to refer to the two network-based virtualisation approaches. Like the symmetric approach, the metadata controller is logically structured in two

layers (Figure 5.18). The volume management layer has the same duties as in the symmetric approach. The second layer is the control layer, which is responsible for the communication with an agent software that runs on the servers.

The agent is required in order to enable direct access to the physical storage resources. It is made up of a data access layer with the same tasks as in symmetric virtualisation and a control layer (Figure 5.18). Via the latter it loads the appropriate location and access information about the physical storage from the metadata controller when the virtual storage is accessed by the operating system or an application. In this manner, access control to the physical resources is still centrally managed by the metadata controller. An agent need not necessarily run in the memory of the server. It can also be integrated into a host bus adapter. This has the advantage that the server can be freed from the processes necessary for virtualisation.

In asymmetric storage virtualisation – as is also the case for symmetric storage virtualisation – advanced storage functions such as snapshots, mirroring or data migration can be realised. The asymmetric model is, however, not so easy to realise as the symmetric one, but performance bottlenecks as a result of an additional device in the data path do not occur here. If we want to increase performance by the use of caching for both application as well as metadata, this caching must be implemented locally on every application server. The caching algorithm to be used becomes very complex since it is a distributed environment, in which every agent holds its own cache (Section 4.3). Data inconsistencies as a result of different cache contents for the same underlying physical storage contents must be avoided and error situations prevented in which an application crashes, that still has data in the cache. Therefore, additional mechanisms are necessary to guarantee the consistency of the distributed cache. Alternatively, the installation of a dedicated cache server in the storage network that devotes itself exclusively to the caching of the data flow would also be possible. Unfortunately, such products are not currently (2009) available on the market.

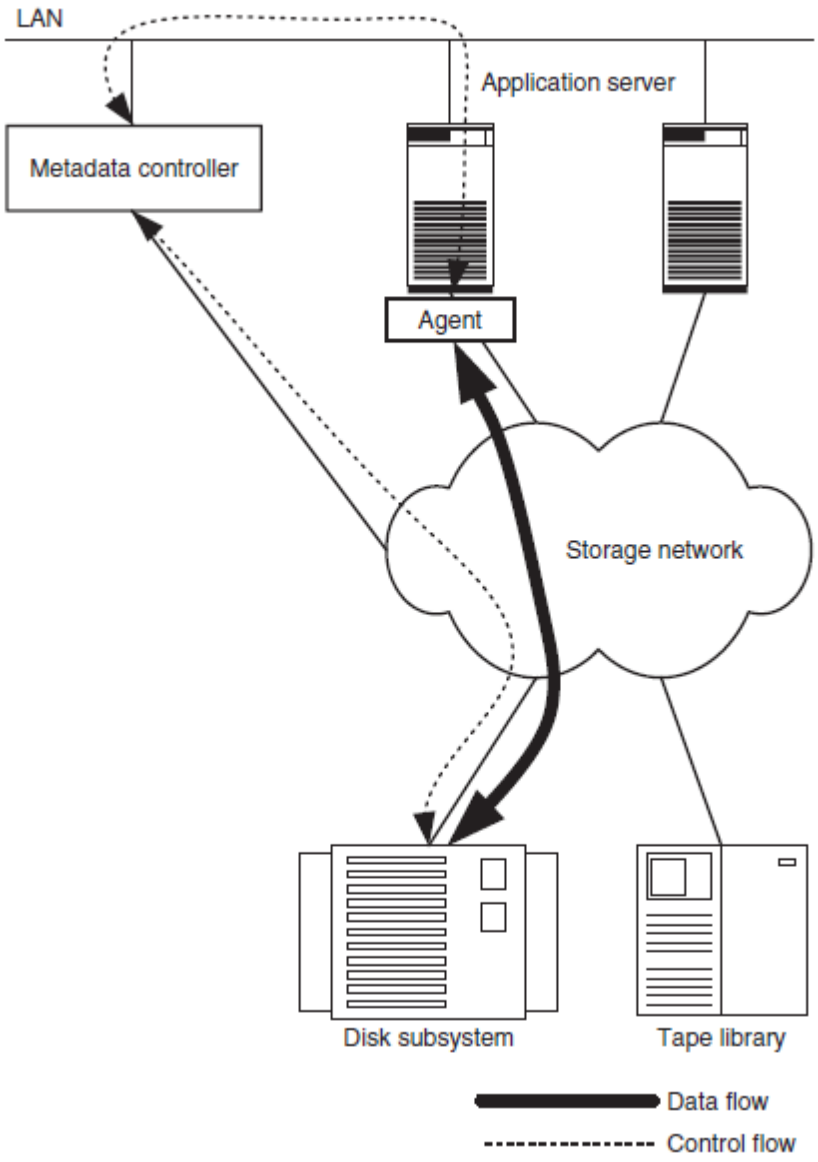


Figure 6.17 In contrast to symmetric virtualisation, in asymmetric virtualisation the data flow is separated from the control flow. The abstraction of physical to logical storage thus takes place outside the data flow.

Metadata controllers can also be constructed as clusters for the load distribution of the control flow and to increase fault-tolerance. The implementation is, however, easier with the asymmetric approach than it is with the symmetric since only the control flow has to be divided over several computers. In contrast to the symmetric approach, the splitting of

the data flow is dispensed with.

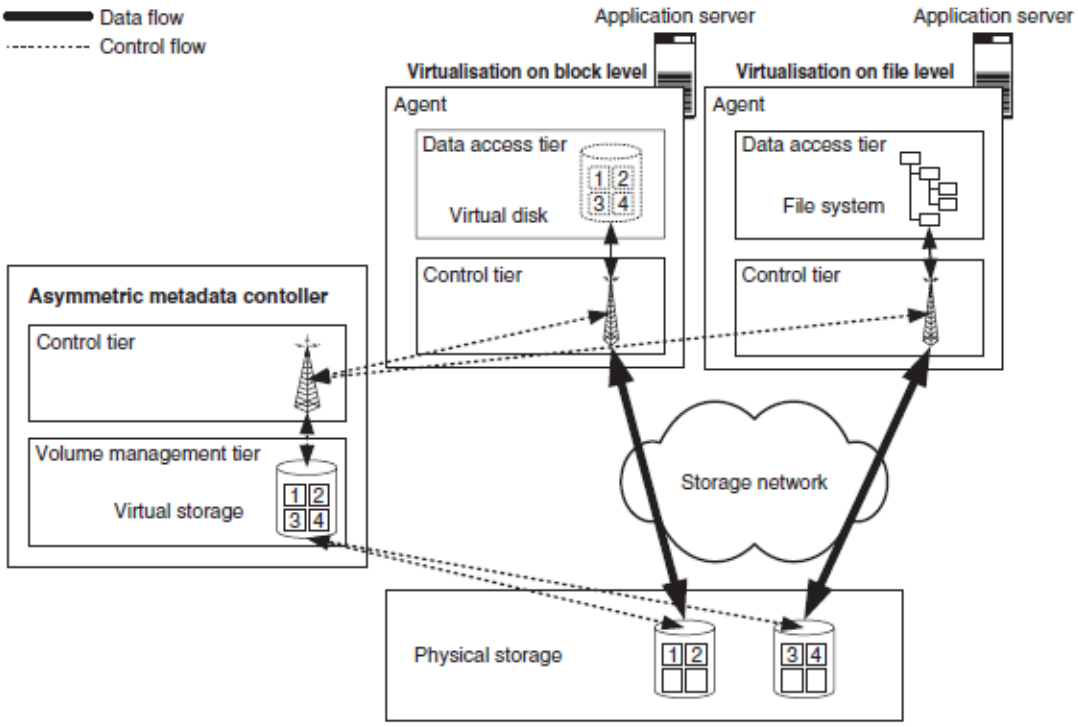


Figure 5.18 In asymmetric virtualisation the metadata controller takes on only the administrative control tasks for the virtualisation. Access to the physical storage is realised by means of an agent software.

The following advantages of asymmetric virtualisation can be established:

• Complete control of storage resources by an absolutely centralised management on the metadata controller.

• Maximum throughput between servers and storage devices by the separation of the control flow from the data flow, thus avoiding additional devices in the data path.

• In comparison to the development and administration of a fully functional volume manager on every server, the porting of the agent software is associated with a low cost.

• As in the symmetric approach, advanced storage functions such as snapshots or mirroring can be used on storage devices that do not themselves support these functions.

• To improve fault-tolerance, several metadata controllers can be brought together to form a cluster. This is easier than in the symmetric approach, since no physical connection from the servers to the metadata controllers is necessary for the data flow.

The disadvantages of asymmetric virtualisation are:

• A special agent software is required on the servers or the host bus adapters. This can make it more difficult to use this approach in heterogeneous environments, since such software or a suitable host bus adapter must be present for every platform. Incompatibilities between the agent software and existing applications may sometimes make the use of asymmetric virtualisation impossible.

• The agent software must be absolutely stable in order to avoid errors in storage accesses.

In situations where there are many different platforms to be supported, this is a very complex development and testing task.

• The development cost increases further if the agent software and the metadata controller

are also to permit access on file level in addition to access on block level.

• A performance bottleneck can arise as a result of the frequent communication between

agent software and metadata controller. These performance bottlenecks can be remedied

by the caching of the physical storage information.

• Caching to increase performance requires an ingenious distributed caching algorithm to avoid data inconsistencies. A further option would be the installation of a dedicated cache server in the storage network.

• In asymmetric virtualisation there is always the risk of a server with no agent software

being connected to the storage network. In certain cases it may be possible for this server to access resources that are already being used by a different server and to accidentally destroy these. Such a situation is called a rogue host condition.

UNIT: 7

SAN ARCHITECTURE AND HARDWARE DEVICES

Network back-up systems can back up heterogeneous IT environments incorporating several thousands of computers largely automatically. In the classical form, network back-up systems move the data to be backed up via the LAN; this is where the name 'network back-up' comes from. This chapter explains the basic principles of network back-up and shows typical performance bottlenecks for conventional server-centric IT architectures.

7.1 CREATING A NETWORK FOR STORAGE

Back-up is always a headache for system administrators. Increasing amounts of data have to be backed up in ever shorter periods of time. Although modern operating systems come with their own back-up tools, these tools only represent isolated solutions, which are completely inadequate in the face of the increasing number and heterogeneity of systems to be backed up. For example, there may be no option for monitoring centrally whether all back-ups have been successfully completed overnight or there may be a lack of overall management of the back-up media. Changing preconditions represent an additional hindrance to data protection. There are three main reasons for this:

1. As discussed in Chapter 1, installed storage capacity doubles every four to twelve months depending upon the company in question. The data set is thus often growing more quickly than the infrastructure in general (personnel, network capacity). Nevertheless, the ever-increasing quantities of data still have to be backed up.
2. Nowadays, business processes have to be adapted to changing requirements all the time. As business processes change, so the IT systems that support them also have to be adapted. As a result, the daily back-up routine must be continuously adapted to the ever-changing IT infrastructure.

3. As a result of globalization, the Internet and e-business, more and more data has to be available around the clock: it is no longer feasible to block user access to applications and data for hours whilst data is backed up. The time window for back-ups is becoming ever smaller. Network back-up can help us to get to grips with these problems.

7.2 SAN HARDWARE DEVICES:

Network back-up systems such as Arcserve (Computer Associates), NetBackup (Veritas), Networker (EMC/Legato) and Tivoli Storage Manager (IBM) provide the following services:

• back-up

• archive

• hierarchical storage management.

The main task of network back-up systems is to back data up regularly. To this end, at least one up-to-date copy must be kept of all data, so that it can be restored after a hardware or application error (‘file accidentally deleted or destroyed by editing’, ‘error in the database programming’).

The purpose of archiving is to freeze a certain version of the data so that this precise version can be restored later on. For example, after the conclusion of a project its data can be archived on the back-up server and then deleted from the local hard disk. This saves local disk space and accelerates back-up and restore processes, since only the data that is actually being worked with has to be backed up or restored.

Hierarchical storage management (HSM) finally leads the end user to believe that any desired size of hard disk is present. HSM moves files that have not been accessed for a long time from the local disk to the back-up server; only a directory entry remains in the local file server. The entry in the directory contains meta information such as file name, owner, access rights, date of last modification and so on.

The metadata takes up hardly any space in the file system compared to the actual file contents, so space is actually gained by moving the file content from the local disk to the back-up server. If a process accesses the content of a file that has been moved in this way, HSM blocks the accessing process, copies the file content back from the back-up server to the local file system and only then gives clearance to the accessing process. Apart from the longer

access time, this process remains completely hidden to the accessing processes and thus also to end users. Older files can thus be automatically moved to cheaper media (tapes) and, if necessary, fetched back again without the end user having to alter his behaviour.

Strictly speaking, HSM and back-up and archive are separate concepts. However, HSM is a component of many network back-up products, so the same components (media, software) can be used both for back-up, archive and also for HSM. When HSM is used, the back-up software used must at least be HSM-capable: it must back up the metadata of the moved files and the moved files themselves, without moving the file contents back to the client. HSM-capable back-up software can speed up back-up and restore processes because only the meta-information of the moved files has to be backed up and restored, not their file contents.

A network back-up system realizes the above-mentioned functions of back-up, archive and hierarchical storage management by the co-ordination of back-up server and a range of back-up clients (Figure 7.1). The server provides central components such as the management of back-up media that are required by all back-up clients. However, different back-up clients are used for different operating systems and applications. These are specialized in the individual operating systems or applications in order to increase the efficiency of data protection or the efficiency of the movement of data.

The use of terminology regarding network back-up systems is somewhat sloppy: the main task of network back-up systems is the back-up of data. Server and client instances of network back-up systems are therefore often known as the back-up server and back-up client, regardless of what tasks they perform or what they are used for.

A particular server instance of a network back-up system could, for example, be used exclusively for HSM, so that this instance should actually be called a HSM server – nevertheless this instance could generally be called a back-up server. A client that provides the back-up function usually also supports archive and the restore of back-ups and archives – nevertheless this client is generally just known as a back-up client. In this book we follow the general, untidy conventions, because the phrase ‘back-up client’ reads better than ‘back-up-archive- HSM and restore client’.

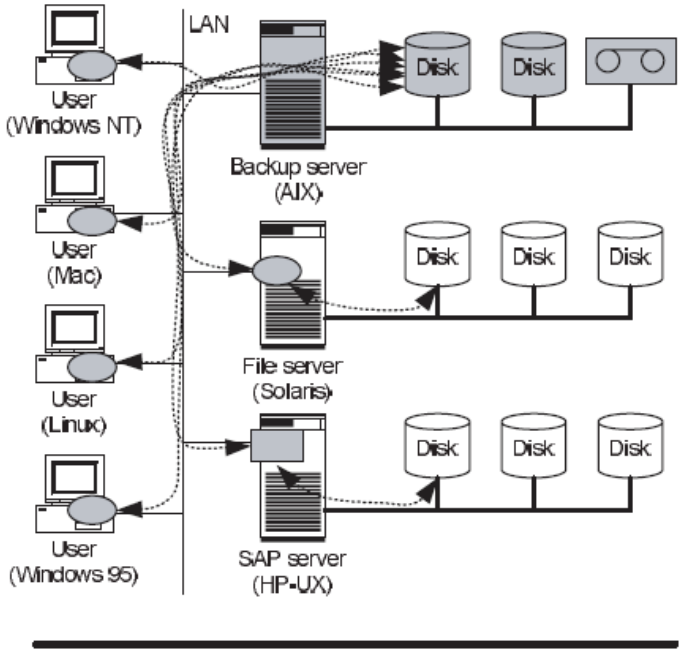


Figure 7.1 Network back-up systems can automatically back-up heterogeneous IT environments via the LAN. A platform-specific back-up client must be installed on all clients to be backed up

7.3 Network Part:

Back-up servers consist of a whole range of component parts. In the following we will discuss the main components: job scheduler, error handler, metadata database and media manager.

7.3.1 Job scheduler

The job scheduler determines what data will be backed up when. It must be carefully configured; the actual back-up then takes place automatically. With the aid of job schedulers and tape libraries many computers can be backed up overnight without the need for a system administrator to change tapes on site. Small tape libraries have a tape drive, a magazine with space for around ten tapes and a media changer that can automatically move the various tapes back and forth between magazine and tape drive. Large tape libraries have several dozen tape drives, space for several thousands of tapes and a media changer or two to insert the tapes in the drives.

7.3.2 Error handling

If a regular automatic back-up of several systems has to be performed, it becomes difficult to monitor whether all automated back-ups have run without errors. The error handler helps to prioritize and filter error messages and generate reports. This avoids the situation in which problems in the back-up are not noticed until a back-up needs to be restored.

7.3.3 Metadata database

The metadata database and the media manager represent two components that tend to be hidden. The metadata database is the brain of a network back-up system. It contains the following entries for every back-up up object: name, computer of origin, date of last change, date of last back-up, name of the back-up medium, etc. For example, an entry is made in the metadata database for every file to be backed up. The cost of the metadata database is worthwhile: in contrast to back-up tools provided by operating systems, network back-up systems permit the implementation of the incremental-forever strategy in which a file system is only fully backed up in the first backup.

In subsequent back-ups, only those files that have changed since the previous back-up are backed up. The current state of the file system can then be calculated on the back-up server from database operations from the original full back-up and from all subsequent incremental back-ups, so that no further full back-ups are necessary. The calculations in the metadata database are generally performed faster than a new full back-up. Even more is possible: if several versions of the files are backed up on the back-up server, a whole file system or a subdirectory dated three days ago, for example, can be restored (point-in-time restore) – the metadata database makes it possible.

7.3.4 Media manager

Use of the incremental-forever strategy can considerably reduce the time taken by the back-up in comparison to the full back-up. The disadvantage of this is that over time the backed up files can become distributed over numerous tapes. This is critical for the restoring of large file systems because tape mounts cost time. This is where the media manager comes into play. It can ensure that only files from a single computer are located on one tape. This reduces the number of tape mounts involved in a restore process, which means that the data can be restored more quickly.

7.4 Connectivity Part

A further important function of the media manager is so-called tape reclamation. As a result of the incremental-forever strategy, more and more data that is no longer needed is located on the back-up tapes. If, for example, a file is deleted or changed very frequently over time, earlier versions of the file can be deleted from the back-up medium. The gaps on the tapes that thus become free cannot be directly overwritten using current techniques. In tape reclamation, the media manager copies the remaining data that is still required from several tapes, of which only a certain percentage is used, onto a common new tape.

The tapes that have thus become free are then added to the pool of unused tapes. There is one further technical limitation in the handling of tapes: current tape drives can only write data to the tapes at a certain speed. If the data is transferred to the tape drive too slowly this interrupts the write process, the tape rewinds a little and restarts the write process. The repeated rewinding of the tapes costs performance and causes unnecessary wear to the tapes so they have to be discarded more quickly. It is therefore better to send the data to the tape drive quickly enough so that it can write the data onto the tape in one go (streaming).

The problem with this is that in network back-up the back-up clients send the data to be backed up via the LAN to the back-up server, which forwards the data to the tape drive. On the way from back-up client via the LAN to the back-up server there are repeated fluctuations in the transmission rate, which means that the streaming of tape drives is repeatedly interrupted.

This storage hierarchy is used, for example, for the back-up of user PCs (Figure 7.2). Many user PCs are switched off overnight, which means that back-up cannot be guaranteed overnight. Therefore, network back-up systems often use the midday period to back up user PCs. Use of the incremental-forever strategy means that the amount of data to be backed up every day is so low that such a back-up strategy is generally feasible. All user PCs are first of all backed up to the hard disk of the back-up server in the time window from 11 : 15 to 13 : 45. The media manager in the back-up server then has a good twenty hours to move the data from the hard disks to tapes. Then the hard disks are once

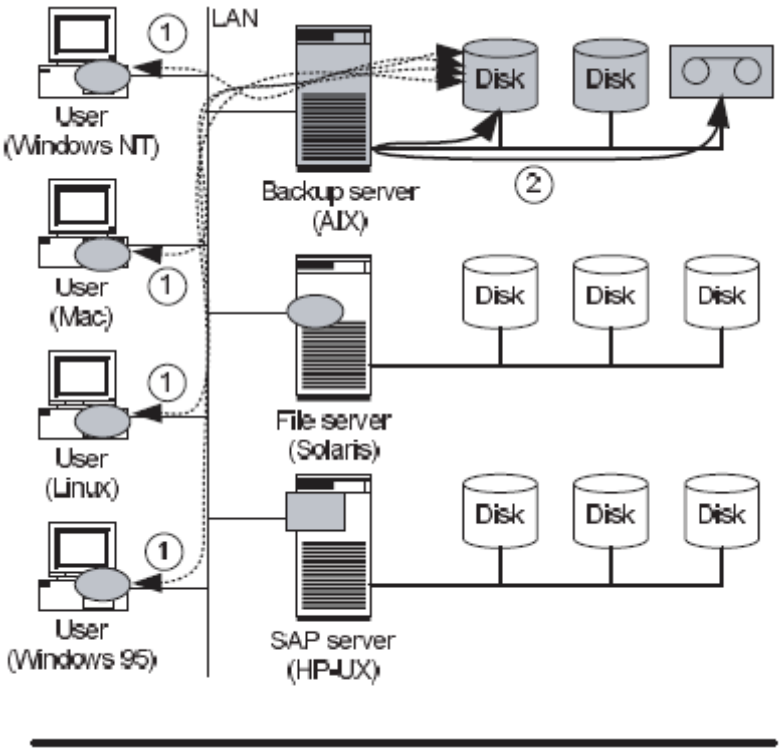


Figure 7.2 The storage hierarchy in the back-up server helps to back user PCs up efficiently. First of all, all PCs are backed up to the hard disks of the back-up server (1) during the midday period. Before the next midday break the media manager copies the data from the hard disks to tapes (2) again free so that the user PCs can once again be backed up to hard disk in the next midday break.

In all operations described here the media manager checks whether the correct tape has been placed in the drive. To this end, the media manager writes an unambiguous signature to every tape, which it records in the metadata database. Every time a tape is inserted the media manager compares the signature on the tape with the signature in the metadata database. This ensures that no tapes are accidentally overwritten and that the correct data is written back during a restore operation.

Furthermore, the media manager monitors how often a tape has been used and how old it is, so that old tapes are discarded in good time. If necessary, it first copies data that is still required to a new tape. Older tape media formats also have to be wound back and forwards now and then so that they last longer; the media manager can also automate the winding of tapes that have not been used for a long time. A further important function of the media manager is the management of data in a so-called off-site store. To this end, the media manager keeps two copies of all data to be backed up. The first copy is always stored on the back-up

server, so that data can be quickly restored if it is required. However, in the event of a large-scale disaster (fire in the data centre) the copies on the back-up server could be destroyed. For such cases the media manager keeps a second copy in an off-site store that can be several kilometres away.

The media manager supports the system administrator in moving the correct tapes back and forwards between back-up server and off-site store. It even supports tape reclamation for tapes that are currently in the off-site store and it.

7.5 Software Part

The logical I/O path offers a second point of view for the definition of transmission techniques for storage networks. Figure 6.2 illustrates the logical I/O path from the disk to the application and shows at which points in the I/O path networks can be used.

Different application protocols are used depending upon location. The same transport protocols and transmission techniques can be used regardless of this. Below the volume manager, block-oriented application protocols are used. Depending upon technique these are SCSI and SCSI offshoots such as FCP, iFCP and iSCSI. Today, block-oriented storage networks are found primarily between computers and storage systems.

However, within large disk subsystems too the SCSI cable is increasingly being replaced by a network transmission technique (Figure 7.3). Above the volume manager and file system, file-oriented application protocols are used. Here we find application protocols such as NFS, CIFS, HTTP, FTP and DAFS. The three different fields of application for file-oriented application protocols are traditional file sharing, high-speed LAN file sharing and the World Wide Web.

Shared disk file systems, which realize the network within the file system, should also be mentioned as a special case.

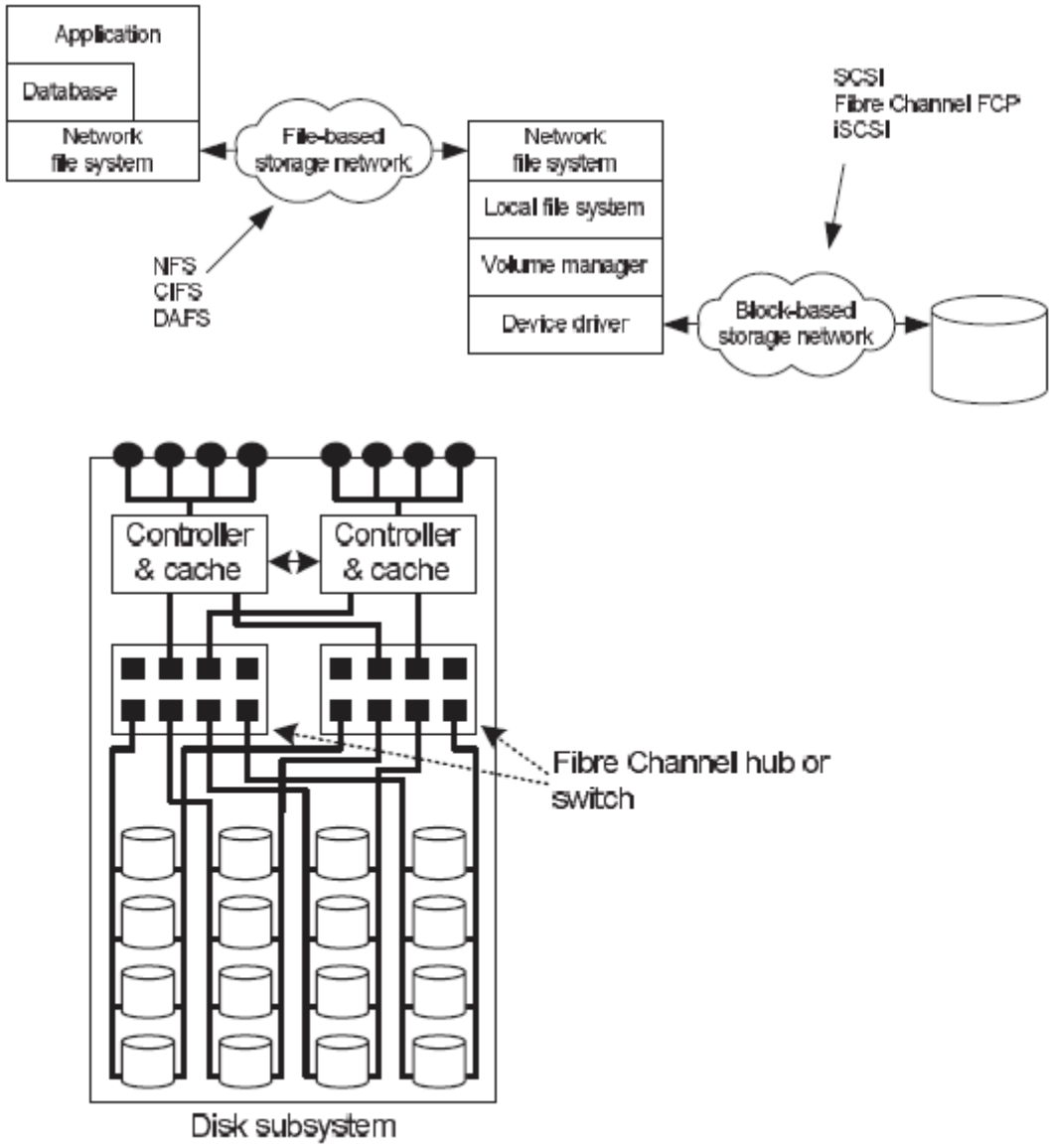


Figure 7.3 A storage network can be hidden within a disk subsystem

7.6 Hardware Part

The management of storage networks is of different significance to various technical fields. For example, the classical network administrator is interested in the question of how the data should be transported and how it is possible to ensure that the transport functions correctly. Further aspects for him are the transmission capacity of the transport medium, redundancy of the data paths or the support for and operation of numerous protocols (Fibre Channel FCP, iSCSI, NFS, CIFS, etc.). In short: to a network administrator it is important how the data travels from A to B and not what happens to it when it finally arrives at its destination.

This is where the field of interest of a storage administrator begins. He is more interested in the organization and storage of the data when it has arrived at its destination. He is concerned with the allocation of LUNs to the servers (LUN mapping) of intelligent storage systems or the RAID levels used. A storage administrator therefore assumes that the data has already arrived intact at point B and concerns himself with aspects of storage. The data transport in itself has no importance to him.

An industrial economist, on the other hand, assumes that A, B and the route between them function correctly and concerns himself with the question of how long it takes for the individual devices to depreciate or when an investment in new hardware and software must be made.

A balanced management system must ultimately live up to all these different requirements equally. It should cover the complete bandwidth from the start of the conceptual phase through the implementation of the storage network to its daily operation. Therefore, right from the conception of the storage network, appropriate measures should be put in place to subsequently make management easier in daily operation.

A good way of taking into account all aspects of such a management system for a storage network is to orientate ourselves with the requirements that the individual components of the storage network will impose upon a management system. These components include:

• Applications

These include all software that processes data in a storage network.

• Data

Data is the term used for all information that is processed by the applications, transported over the network and stored on storage resources.

• Resources

The resources include all the hardware that is required for the storage and the transport of the data and the operation of applications.

• Network

The term network is used to mean the connections between the individual resources. Diverse requirements can now be formulated for these individual components with regard to monitoring, availability, performance or scalability. Some of these are requirements such as monitoring that occur during the daily operation of a storage network, others are requirements

such as availability that must be taken into account as early as the implementation phase of a storage network.

7.7 Host Bus Adapter

In computer hardware, a host controller, host adapter, or host bus adapter (HBA) connects a host system (the computer) to other network and storage devices. The terms are primarily used to refer to devices for connecting SCSI, Fibre Channel and eSATA devices, but devices for connecting to IDE, Ethernet, FireWire, USB and other systems may also be called host adapters. Recently, the advent of iSCSI and Fibre Channel over Ethernet has brought about Ethernet HBAs, which are different from Ethernet NICs in that they include TCP Offload Engines. There are also converged HBAs that support both Ethernet and Fibre Channel called Converged Network Adapters (CNAs).

7.8 The Fibre Channel Switch

N-Port (Node Port): originally the communication of Fibre Channel was developed around N-Ports and F-Ports, with 'N' standing for 'node' and 'F' for 'fabric'. An NPort describes the capability of a port as an end device (server, storage device), also called node, to participate in the fabric topology or to participate in the point-to-point topology as a partner.

• F-Port (Fabric Port): F-Ports are the counterpart to N-Ports in the Fibre Channel switch. The F-port knows how it can pass a frame that an N-Port sends to it through the Fibre Channel network on to the desired end device.

• L-Port (Loop Port): the arbitrated loop uses different protocols for data exchange than the fabric. An L-Port describes the capability of a port to participate in the arbitrated loop topology as an end device (server, storage device). More modern devices are now fitted with NL-Ports instead of L-Ports. Nevertheless, old devices that are fitted with an L-Port are still encountered in practice.

• NL Port (Node Loop Port): an NL-Port has the capabilities of both an N-Port and an L-port. An NL-Port can thus be connected both in a fabric and in an arbitrated loop. Most modern host bus adapter cards are equipped with NL-Ports.

• FL-Port (Fabric Loop Port): an FL-Port allows a fabric to connect to a loop. However, this is far from meaning that end devices in the arbitrated loop can communicate with end devices in

the fabric. More on the subject of connecting fabric and arbitrated loop can be found in Section 3.4.3.

• E-Port (Expansion Port): two Fibre Channel switches are connected together by E-Ports. E-Ports transmit the data from end devices that are connected to two different Fibre Channel switches. In addition, Fibre Channel switches smooth out informationover the entire Fibre Channel network via E-ports.

• G-Port (Generic Port): modern Fibre Channel switches configure their ports automatically. Such ports are called G-Ports. If, for example, a Fibre Channel switch is connected to a further Fibre Channel switch via a G-Port, the G-Port configures itself as an E-Port.

• B-Port (Bridge Port): B-Ports serve to connect two Fibre Channel switches together via ATM or SONET/SDH. Thus Fibre Channel SANs that are a long distance apart can be connected together using classical WAN techniques. In future versions of the Fibre Channel standard we can expect B-Ports to also support Ethernet and IP.

7.9 Fabric Operation from a Hardware Perspective

The Fibre Channel standard defines six different service classes for data exchange between end devices. Three of these defined classes (Class 1, Class 2 and Class 3) are realized in products available on the market, with hardly any products providing the connection oriented Class 1. Almost all new Fibre Channel products (host bus adapters, switches, storage devices) support the service classes Class 2 and Class 3, which realize a packetoriented service (datagram service). In addition, Class F serves for the data exchange between the switches within a fabric.

Class 1 defines a connection-oriented communication connection between two node ports: a Class 1 connection is opened before the transmission of frames. This specifies a route through the Fibre Channel network. Thereafter, all frames take the same route through the Fibre Channel network so that frames are delivered in the sequence in which they were transmitted. A Class 1 connection guarantees the availability of the full

bandwidth. A port thus cannot send any other frames while a Class 1 connection is open.

Class 2 and Class 3, on the other hand, are packet-oriented services (datagram services): no dedicated connection is built up, instead the frames are individually routed through the Fibre Channel network. A port can thus maintain several connections at the same time.

Several Class 2 and Class 3 connections can thus share the bandwidth.

Class 2 uses end-to-end flow control and link flow control. In Class 2 the receiver acknowledges each received frame (acknowledgement, Figure 3.16). This acknowledgement is used both for end-to-end flow control and for the recognition of lost frames.

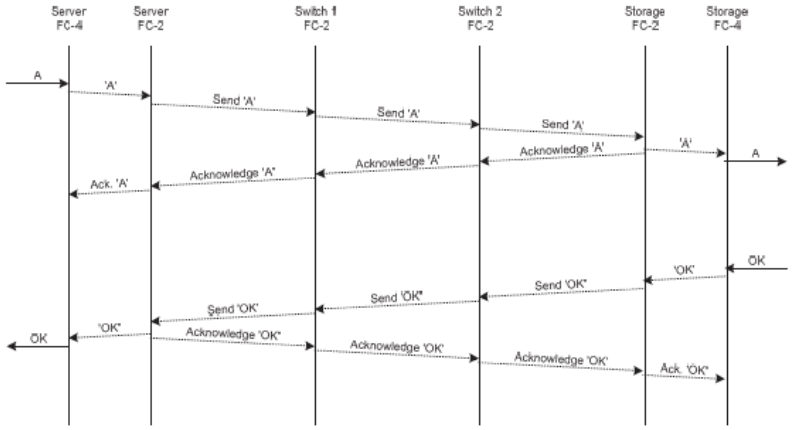


Figure 7.4 Class 2: each Fibre Channel frame transmitted is acknowledged within the FC-2 layer. The acknowledgement aids the recognition of lost frames and the end-to-end flow control.

The link flow control and the conversion of sequences to frames are not shown missing acknowledgement leads to the immediate recognition of transmission errors by FC-2, which are then immediately signalled to the higher protocol layers. The higher protocol layers can thus initiate error correction measures straight away.

Users of a Class 2 connection can demand the delivery of the frames in the correct order. Class 3 achieves less than Class 2: frames are not acknowledged. This means that only link flow control takes place, not end-to-end flow control. In addition, the higher protocol layers must notice for themselves whether a frame has been lost. The loss of a frame is indicated to higher protocol layers by the fact that an expected sequence is not delivered because it has not yet been completely received by the FC-2 layer. A switch may dispose of Class 2 and Class 3 frames if its buffer is full. Due to greater time-out values in the higher protocol layers it can take much longer to recognize the loss of a frame than is the case in Class 2

We have already stated that in practice only Class 2 and Class 3 are important. In practice the service classes are hardly ever explicitly configured, meaning that in current Fibre Channel SAN implementations the end devices themselves negotiate whether they communicate by Class 2 or Class 3. From a theoretical point of view the two service classes

differ in that Class 3 sacrifices some of the communication reliability of Class 2 in favour of a less complex protocol.

Class 3 is currently the most frequently used service class. This may be because the current Fibre Channel SANs are still very small, so that frames are very seldom lost or overtake each other. The linking of current Fibre Channel SAN islands to a large SAN could lead to Class 2 playing a greater role in future due to its faster error recognition.

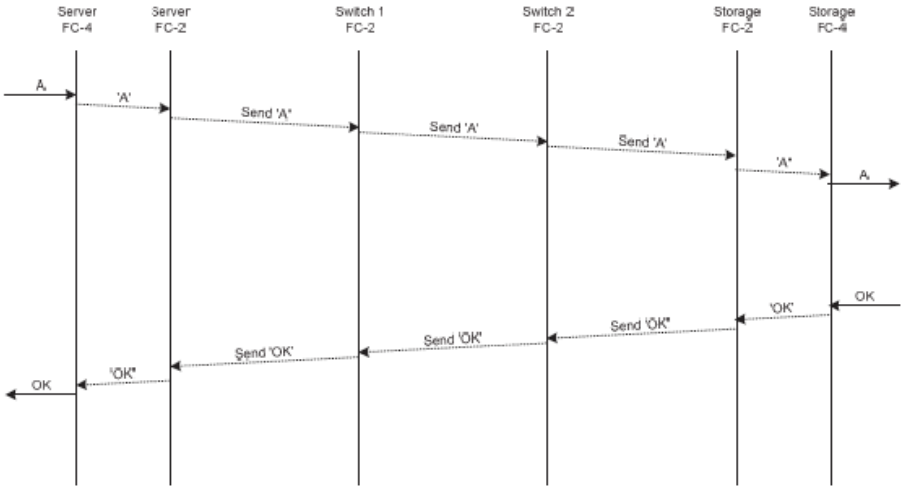


Figure 3.17 Class 3: transmitted frames are not acknowledged in the FC-2 layer. Lost frames must be recognized in the higher protocol layers .
The link flow control and the conversion of sequences to frames are not shown

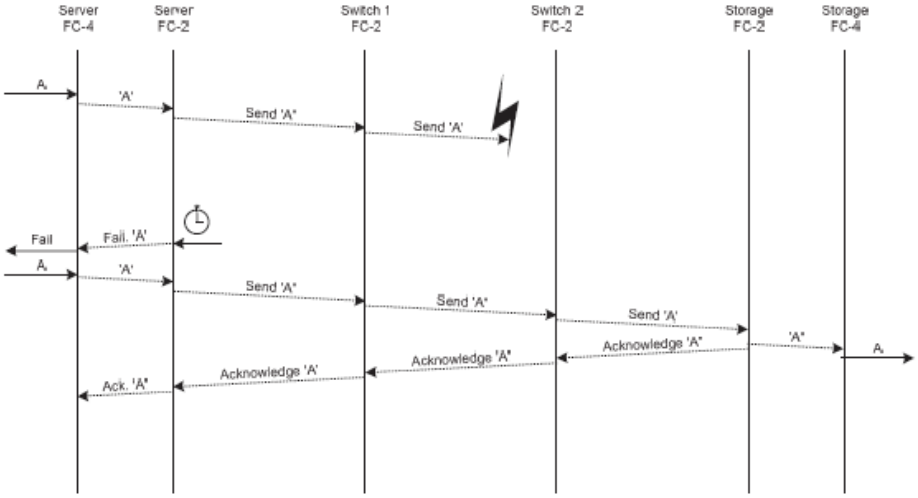


Figure 3.18 Transmission error in Class 2: the time-outs for frames are relatively short on the FC-2 layer. Missing acknowledgements are thus quickly recognized within the FC-2 layer of the transmitter and signalled to the higher protocol levels.

The higher protocol layers are responsible for the error processing. In the figure the lost frame is simply resent. The link flow control and the conversion of sequences to frames are not shown

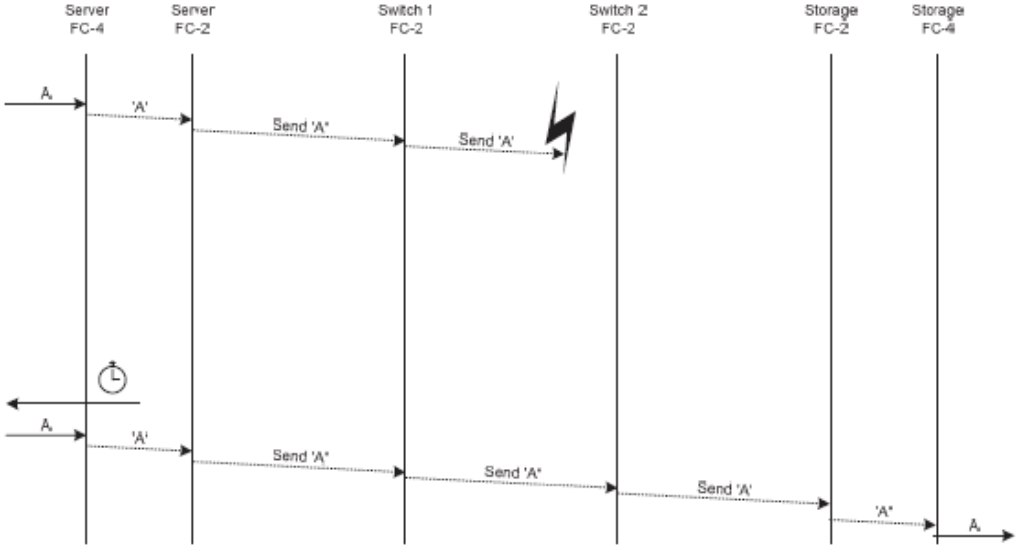


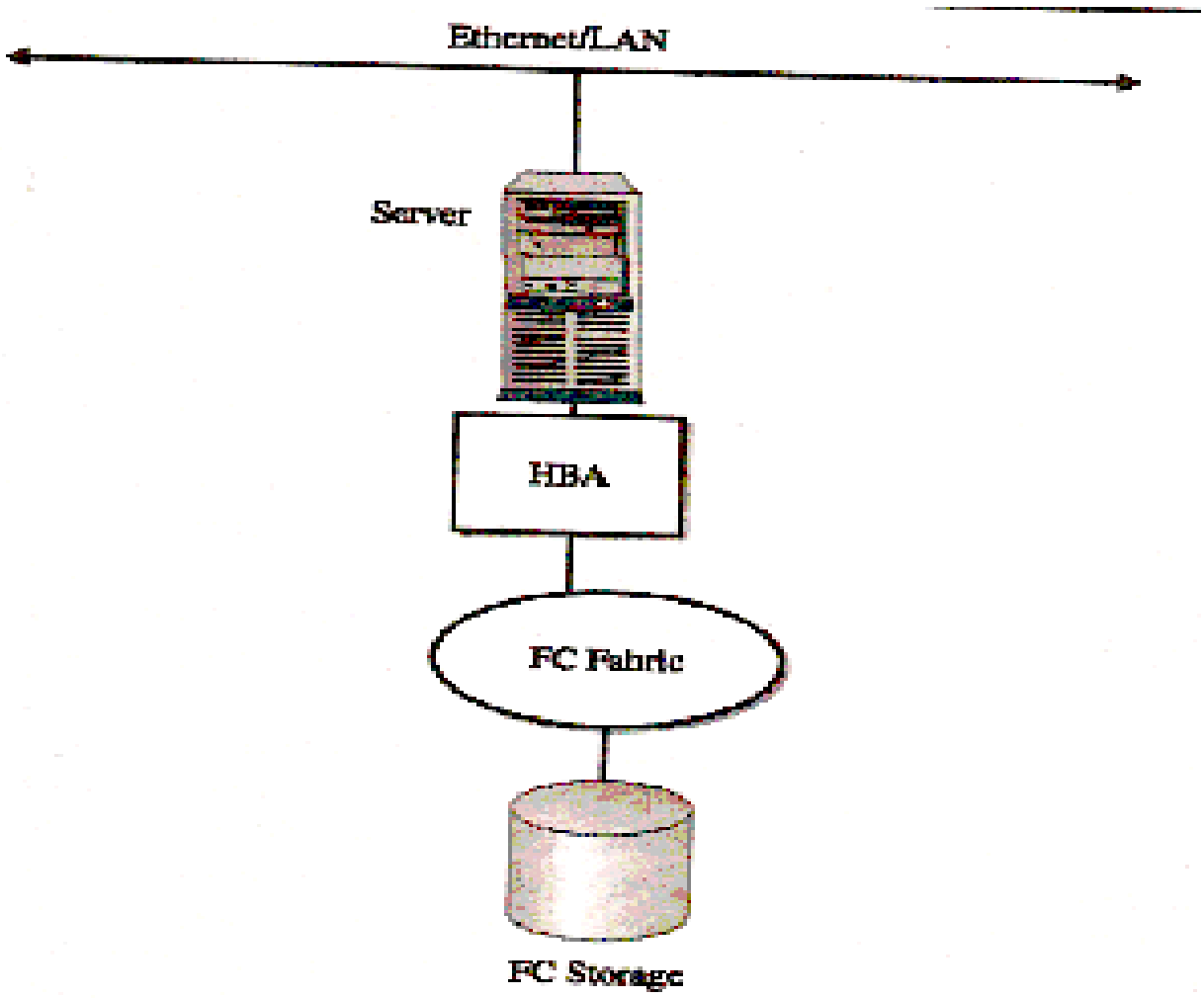
Figure 3.19 Transmission errors in Class 3: here too the higher protocol layers are responsible for error processing. The time-outs in the higher protocol layers are relatively long in comparison to the time-outs in the FC-2 layer.

In Class 3 it thus take significantly longer before there is a response to a lost frame. In the figure the lost frame is simply resent. The link flow control and the conversion of sequences to frames are not shown

UNIT: 8

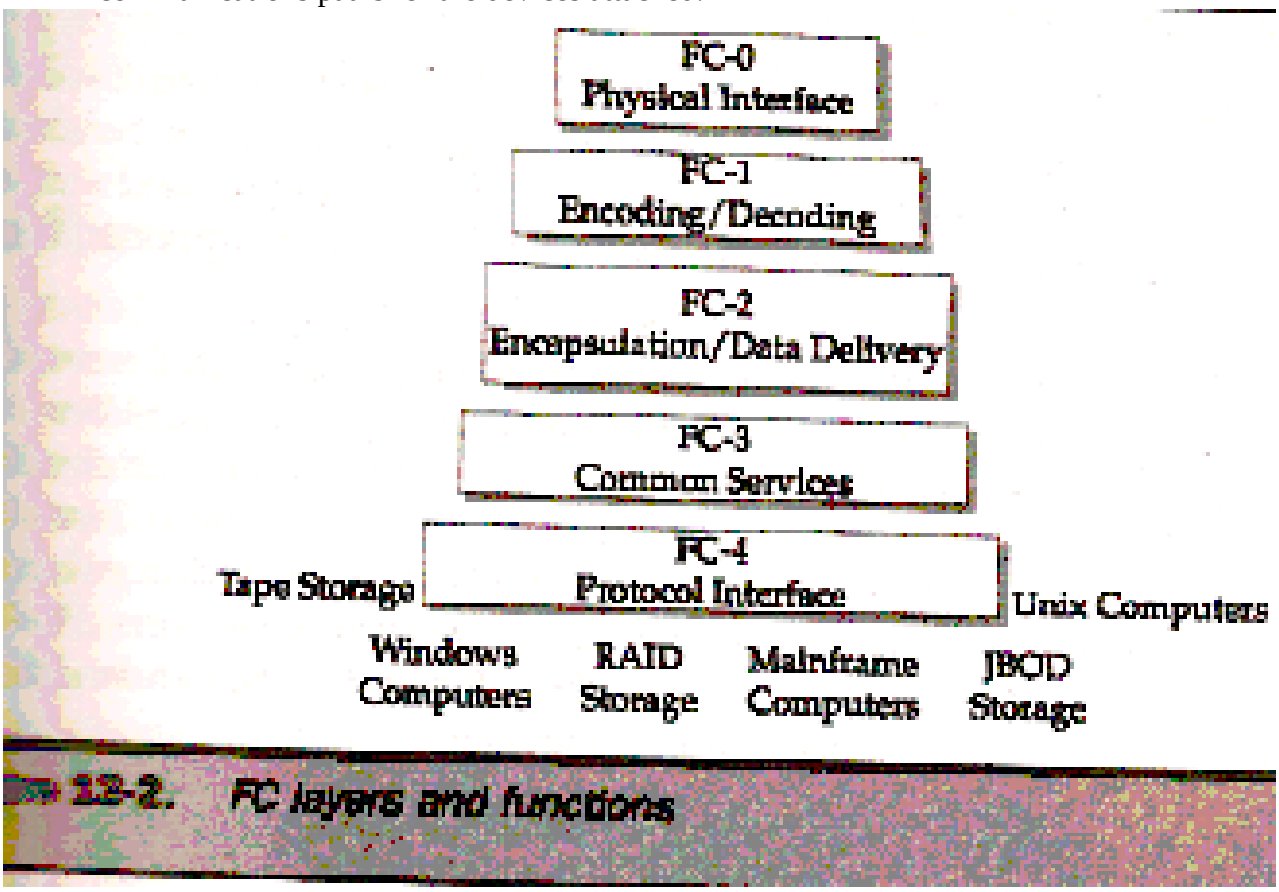
SAN COMPONENTS

- Network part
- Hardware part
- Software part
- Connectivity part



Network Part

- Provides communication among devices attached to it
- FC standard protocol is used provides layered approach to communicate.
- Network is switch component that provides a set of circuits making up the communications paths for the devices attached.



12-2. FC layers and functions

FC Switched fabric solutions.

- Point-to-Point
- Uses FC to connect one device to another
- Hub
- Uses FC to connect in loop fashion
- It leveraged the speed of FC and the capability to place additional disk arrays within the network allowing connection of additional servers.

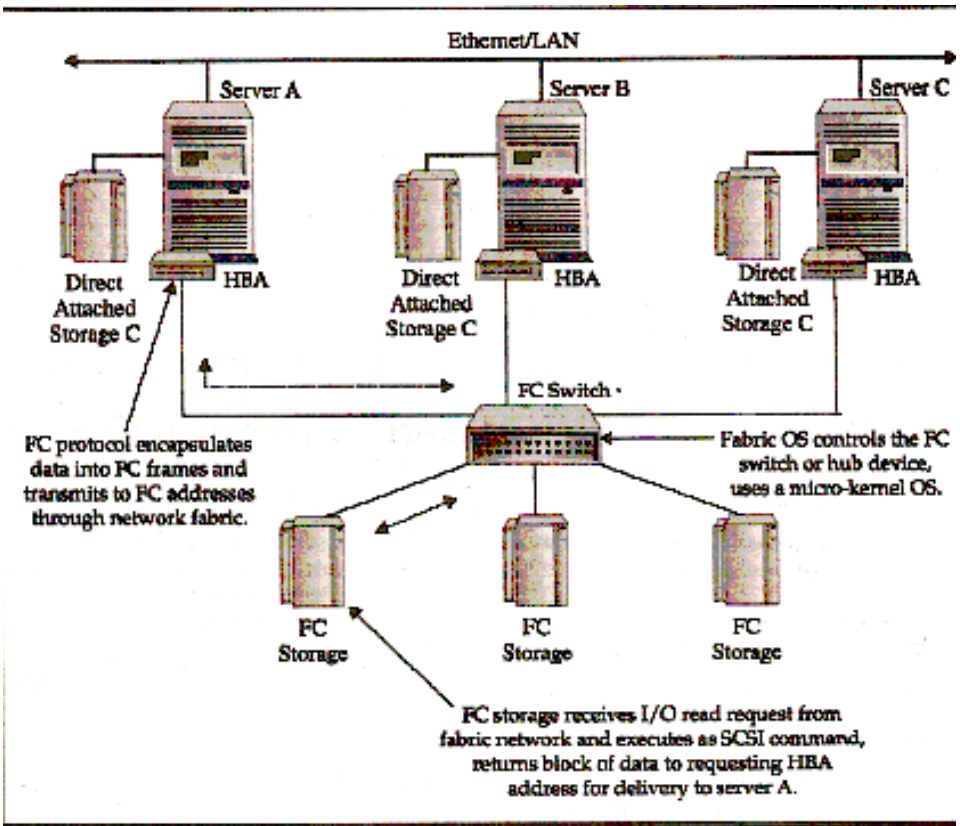


Figure 13-3. A SAN devices overview and sample read I/O

The Software Part

Fabric OS services

- Fabric OS runs within the FC switch

- **OS Services**
- Required functions configured with micro kernel such as task, memory, and file management.

- **FC Services**
- These functions interact closely with OS services to facilitate the FC layer processing of frame management within switch.
- FC services provide both switched fabric and arbitrated loop modes of processing.
- **Simple Name Services**

- Identifies devices by their names.

- Provide a rudimentary database and functions to identify new devices.

- **Alias Services**

- Fabric provides functions that allow particular devices to broadcast frames to a set of aliases.

- **Management Services**

- Management functions consist of status and activity information stored and accesses through a simple network management protocol (SNMP) database.

- Database referred to as MIB(Management Information Base)

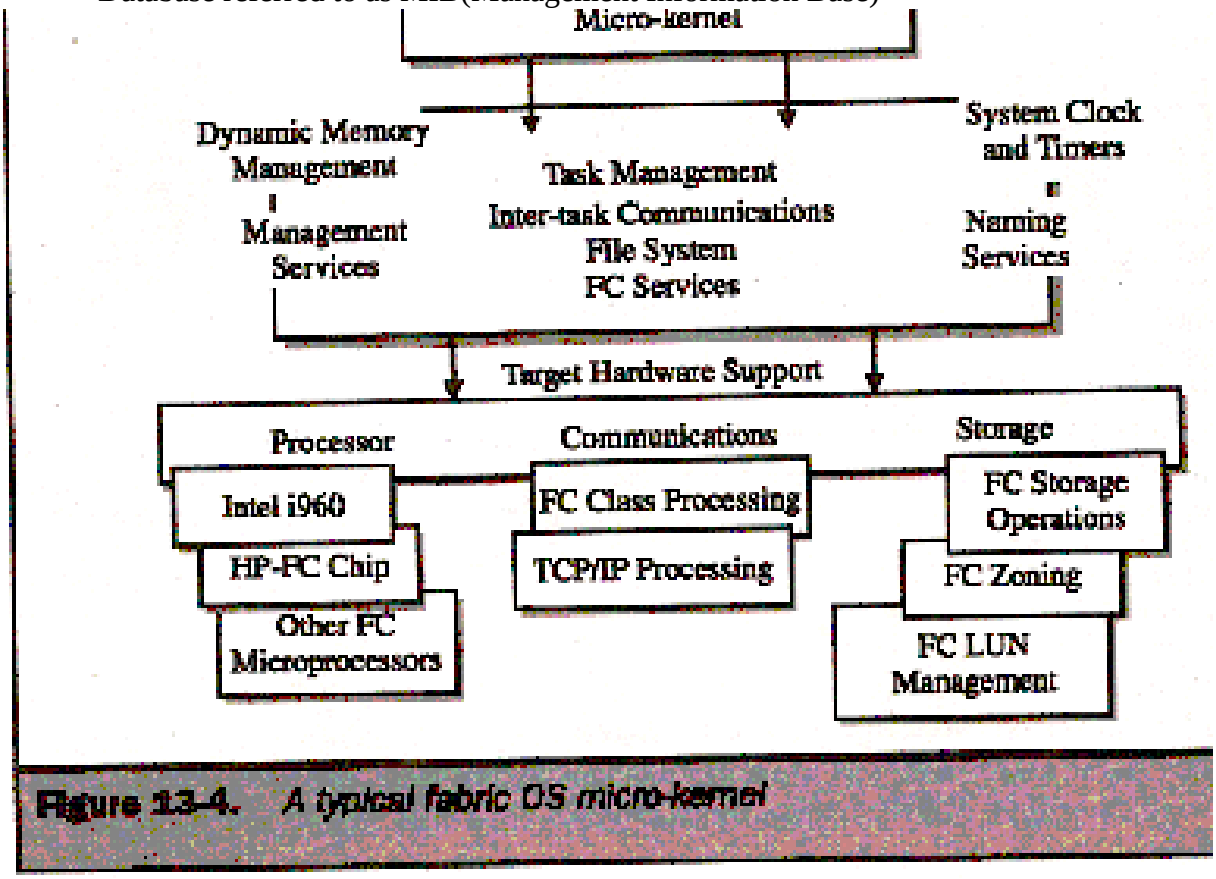


Figure 13.4. A typical fabric OS micro-kernel

Fabric APIs and Applications

- **Application Programming Interfaces.**
- Provides interfaces to the outside world for access from administrators and other software.
- These access points will come from application programming interfaces, or APIs
- Provides third-party software vendors, or any system administrators.

- **Fabric Applications.**

- Zoning, Hardware enclosure services and advanced configuration features.

- **Server OS and HBA Drivers**
- OS is able to recognize the FC host bus adapter driver and related functions to link to SAN attached disks.
- All HBA vendors provide drivers to support various SAN configurations and topologies.
- **Storage Intelligence**
- **Compatible with both the server OS and fabric OS, as well as the HBA drivers.**
- **Other Networks.**
- **Provides method for switch-to-switch connectivity, thus allowing remote connectivity from one SAN configuration to another.**

The Hardware Part

- **The Fastest Storage Possible:100 MB/sec**
- **The usual Configuration Gang, RAID, Disk Array, and Tape**
- **The unusual configuration Club, Optical, NAS, and Mainframe Channels**

The Fastest Storage Possible:100 MB/sec

- **The minimum devices necessary are an FC switch, an FC-enabled server, and FC-enabled storage systems.**
- **The fig below shows the minimum components necessary to configure a SAN.**
- **FC switch centers the network it connects server and storage array.**
- **FC server is connected through FC Host Bus Adapter(HBA) which provides necessary FC protocol processing and interfaces.**
- **FC storage array is connected through an integrated FC port attachment that injects the necessary FC protocol communications into the storage controllers mechanisms.**

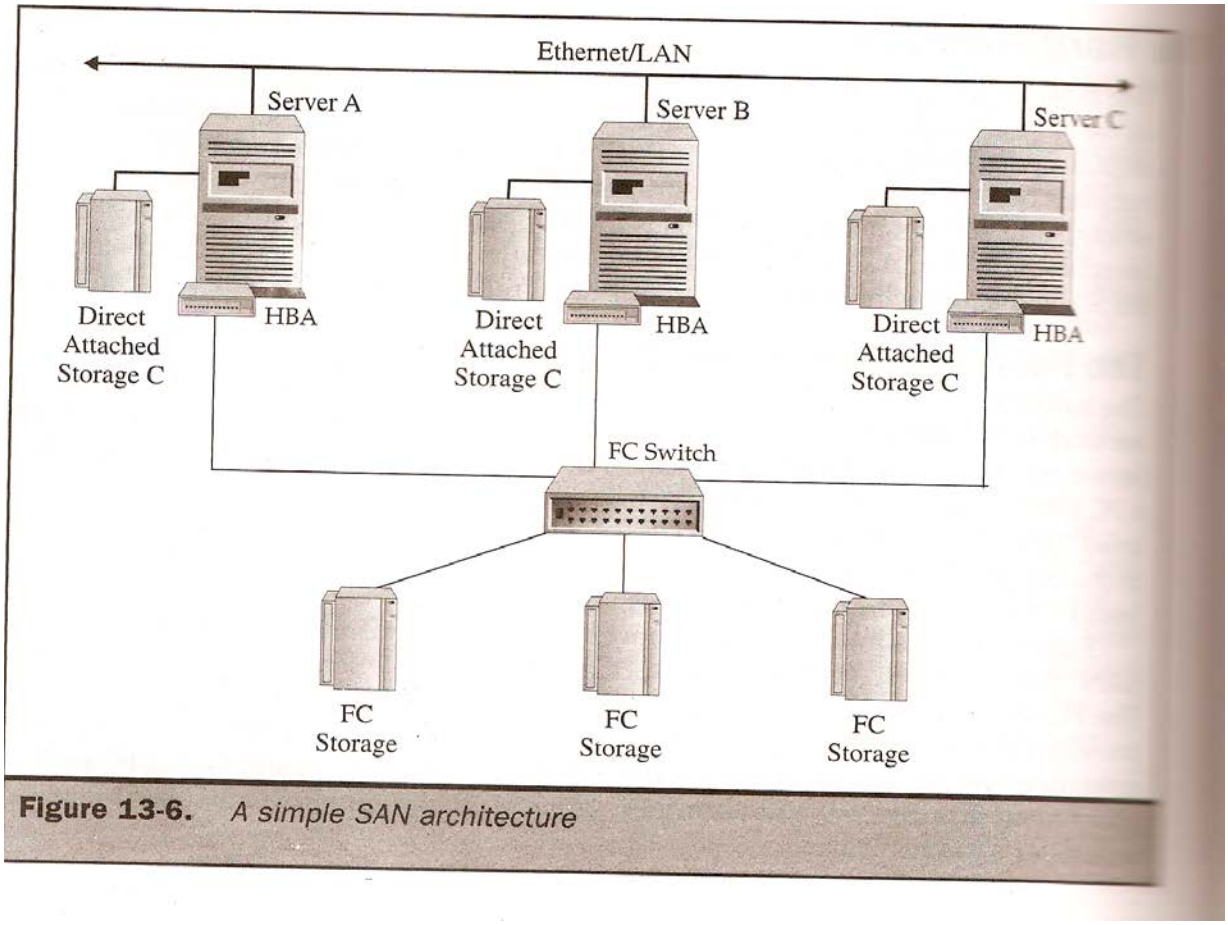


Figure 13-6. *A simple SAN architecture*

The usual Configuration Gang, RAID, Disk Array, and Tape

- The fig below shows the data center setting to SAN configuration.
- SAN has been enhanced to include additional servers, two storage arrays both supporting RAID levels, and tape device connected through a FC bridge.
- Data center operations includes backup and recovery, and shared access.

The unusual configuration Club, Optical, NAS, and Mainframe Channels

- Important to point out because they show the increasing levels of flexibility SANs have.

SAN Configurations

- Entry level
- Configuration contains a single FC switch, two storage arrays, and a tape device.
- Three window based servers attached to the switch which complete the SAN network.

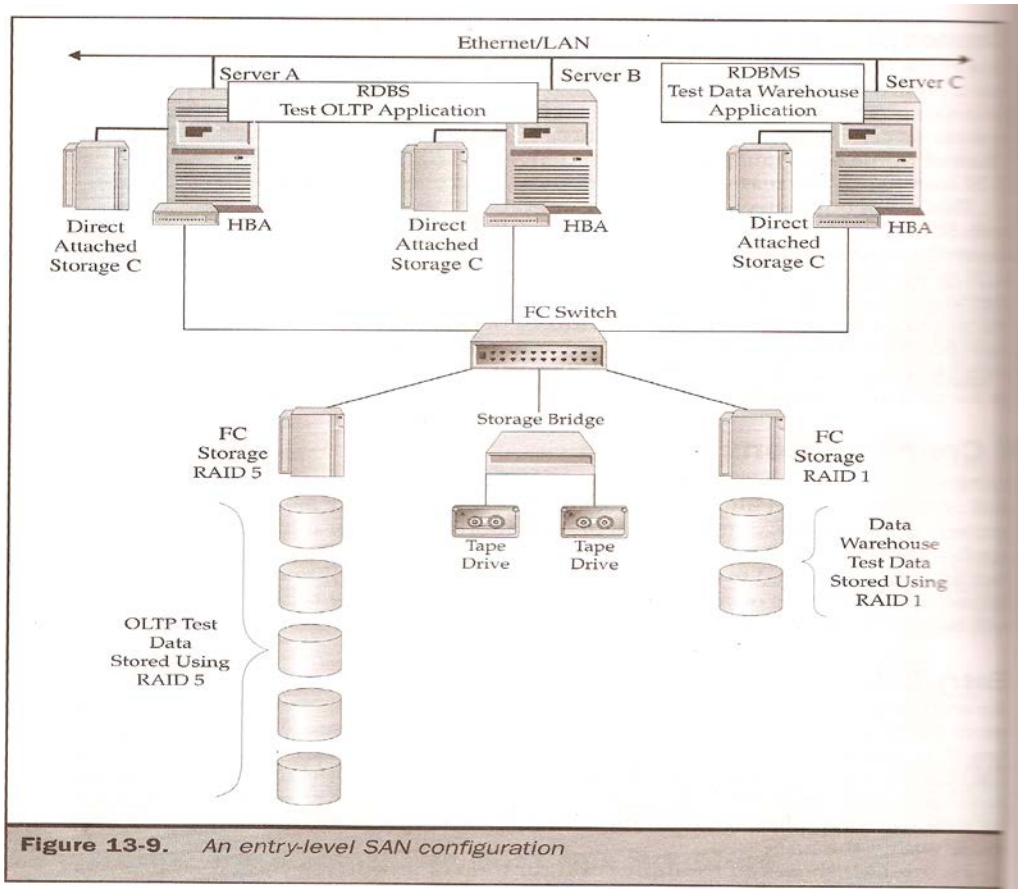


Figure 13-9. *An entry-level SAN configuration*

Mid-Range: Production Workloads, Multiple Switches

- The fig shows an example SAN that supports production applications with ten servers using Windows OSs.

- The servers are attached to three FC switches configured in whats called a cascading arrangement.

- Production workload supports several database applications that utilize five of the ten servers. These transactional OLTP-type requirements utilize the storage arrays configured for RAID 5.

Enterprise: Director-Level with Duplicate paths and Redundancy

- Consist of 20 servers, consisting of both UNIX- and Windows-based OSs, attached to six FC switches.

- Storage arrays support 2 terra bytes of both file and database oriented data models, given the workloads

range from OLTP database applications to webservers, and data base oriented data models, given that the workloads range from OLTP database applications to web servers, and data-centric data mart applications.

San Hardware

- The Fibre Channel standard defines three types of ports:
- F_Port used to attach nodes
- E_Port used to connect one switch to another switch.
- G_Port A generic port that can be pressed into services as an F_Port or an E_Port depending on the vendors implementation.

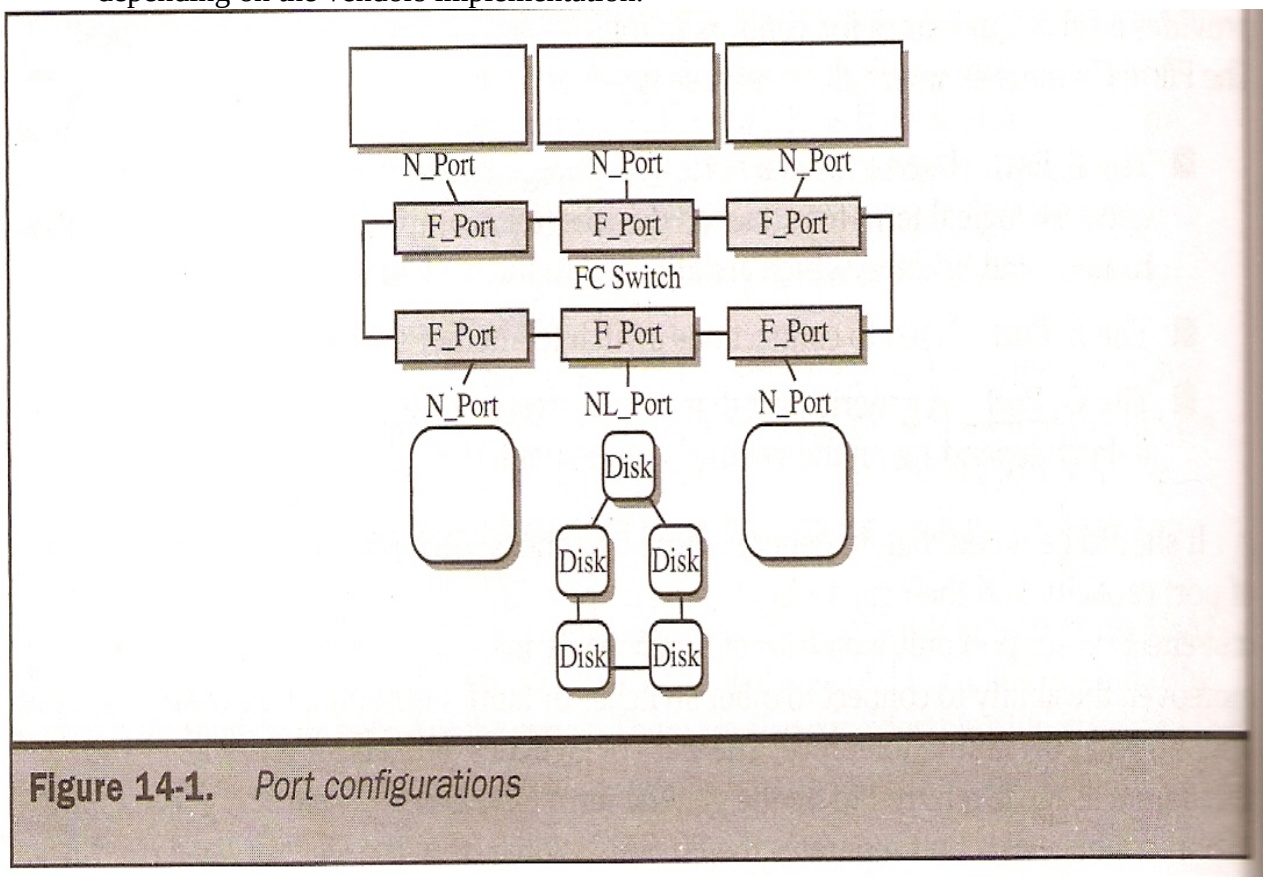


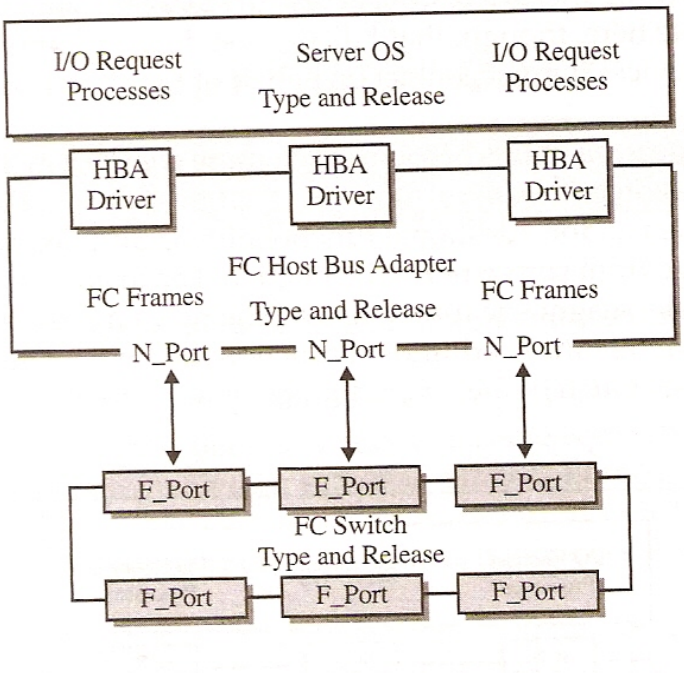
Figure 14-1. *Port configurations*

Host Bus Adapter

- Link between the server and the Storage Area Network.
- HBAs provide the translation between server protocol and switch protocol.
- HBAs connect to a servers PCI bus and come with software drivers that support fabric topologies and arbitrated loop configurations.
- HBAs are available in single port or multiple port configurations.
- Multi ports allow additional data paths for workloads moving between the server and the switch via a single HBA.
- HBAs encapsulate SCSI disk commands into the Fibre Channel layer2 processing.
- HBAs communicate within the FC standard through the class of service defined by Name Server at the time of login.
- Major realiability questions

1. HBAs compatibility with its servers operating system is key effective operation of the FC network in total bcoz operating system has unique difference in how it handles base I/O, file system, and buffering/caching methods.

   Compatibility at a software level, of the switchs fabric operating system and HBAs software drivers.



Server OS, FC HBA, and FC switch release levers must all be compatible to match operating systems, installed storage types, and switch port functions. The critical link between the I/O request and storage network is the HBA.

Figure 14-5. *HBA's critical role in interoperability*

- Putting the Storage in Storage Area Networks
- Fibre channel-enabled disks come into two configurations:
- JBOD-Just Bunch of Disks
- RAID
- FC storage arrays, which are multiple disks hooked together, are connected in a loop configuration.
- FC storage arrays provides additional functions, such as the stripping of data across units, which allows a single outsized file to be spread across two, three, or even four drives in the array.

JBOD

- JBOD configuration is connected to the switch through an NL port.
- Most implementation provides dual loop capacity where as redundancy protects against single loop failure.

- Dual loop requires four switchs ports one loop serving drives 1-4 and second loop for drives 5-8.

- Disadvantage of any JBOD implementation is its lack of fault resiliency, though there are software RAID products that allow the stripping of data with recovery mechanisms encompassed in JBOD enclosure.
- RAID

- RAID offers a way of protecting the data by creating an array of disks drives that are viewed as one logical volume.

- Key advantage of FC RAID is its ability to provide levels of fault resistance for hardware failures.
- Bridges and Routers

- FC enabled tape devices have been problematic in the support of new storage model hence new type of device required to bridge the FC protocol into a SCSI bus architecture used in tape media.

- Routers provide effective means of establishing a tape media within SAN configurations.

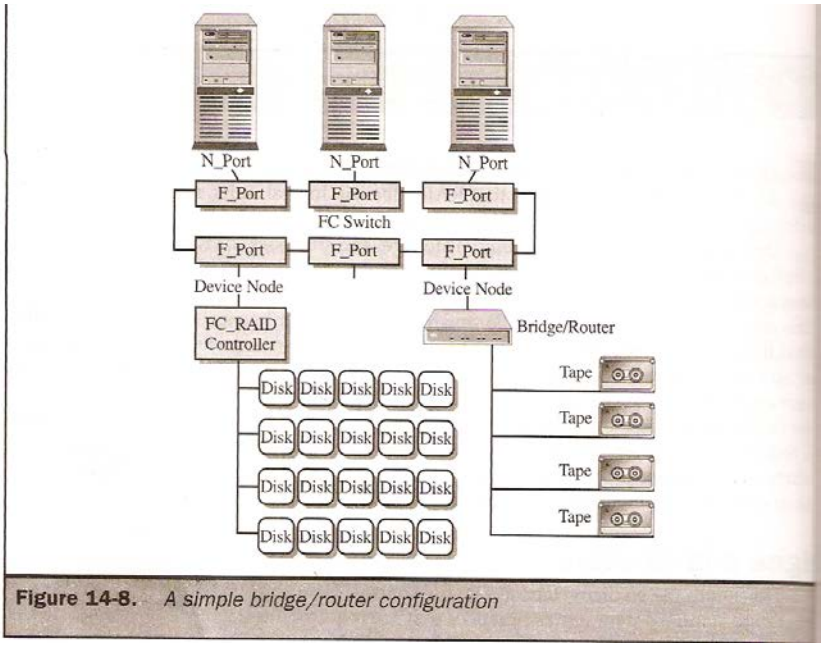


Figure 14-8. *A simple bridge/router configuration*

- Fabric Operation from a Hardware Perspective
- Standard is broken down into logical constructs of frame, sequence, and exchanges.
- FC provides its storage flexibility through the ability to transport other protocols, such as the SCSI protocol.
- Frame The logical construct of transporting data through the switch.
- Sequence A block of numbered or related frames transported in sequence from initiator to target. Error recovery takes place as the receiving port processes the sequence.
- Exchange A number of nonconcurrent sequences processed bidirectionally.