OPERATING SYSTEM IT-42033

CHAPTER – 12 Q&A

12.14

Most disks do not export their rotational position information to the host. Even if they did, the time for this information to reach the scheduler would be subject to imprecision and the time consumed by the scheduler is variable, so the rotational position information would become incorrect. The disk requests are usually given in terms of logical block numbers, and the mapping between logical blocks and physical locations is very complex.

12.21

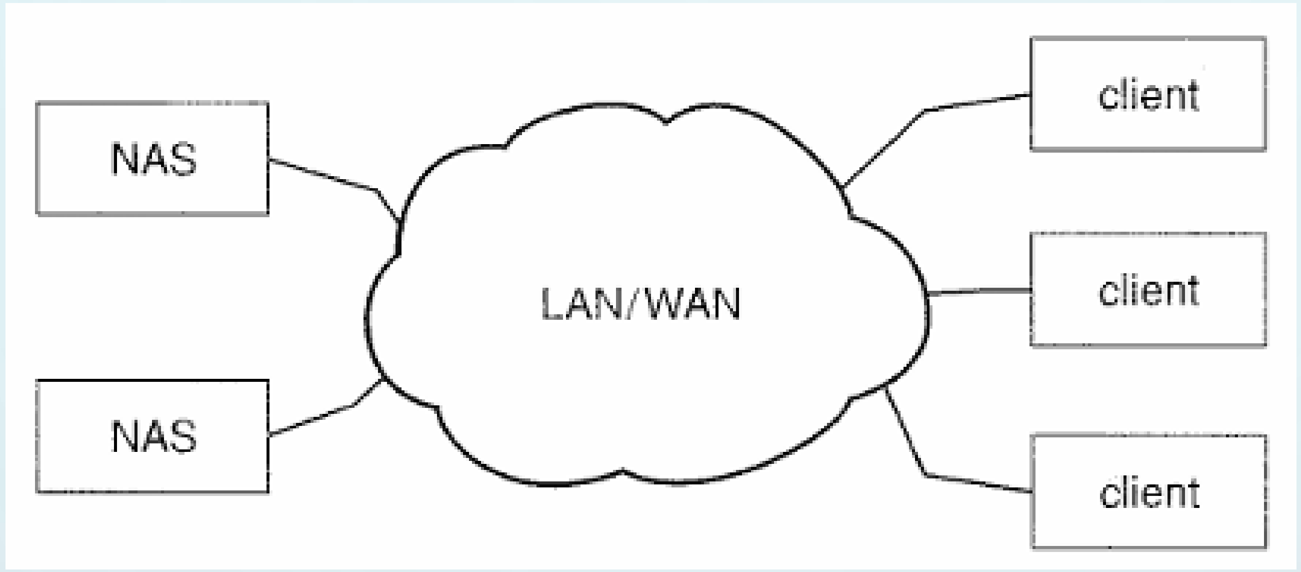
Disk scheduling attempts to reduce the overhead time of disk head positioning. Since a RAM disk has uniform access times, scheduling is largely unnecessary. The comparison between RAM disk and the main memory disk-cache has no implications for hard-disk scheduling because it is scheduled only the buffer cache misses, not the requests that find their data in main memory.

Q. What are the primary advantages of using network-attached storage (NAS) in a local area network (LAN)?

Network-attached storage (NAS) is a specialized storage system that allows remote access over a data network, enabling multiple clients to share a centralized pool of storage. Clients interact with NAS devices using remote procedure call (RPC) interfaces, such as NFS for UNIX systems with communication occurring over TCP or UDP on the same local area network (LAN) that handles other data traffic.

NAS devices are often implemented as RAID arrays, equipped with software that supports the RPC interface. While NAS simplifies storage access and naming for all computers on a LAN, it offers lower performance compared to direct-attached storage options. The latest NAS protocols utilize the IP network to carry SCSI commands, allowing hosts to treat remote storage.

Q. What are the primary advantages of using network-attached storage (NAS) in a local area network (LAN)? (Cont …)



Network-attached storage

Q. What is the purpose of low-level formatting, and how does it prepare a disk for data storage?

Boot Block

Disk formatting is a process that prepares a new magnetic disk for data storage. Initially, a blank disk must be divided into sectors that the disk controller can read and write, a process known as low-level formatting. This formatting fills each sector with a specific data structure, typically comprising a header, a data area (512 bytes), and a trailer. The header and trailer contain essential information for the disk controller, such as the sector number and an error-correcting code (ECC). The ECC is vital for data integrity; it is updated during write operations and checked during read operations. If discrepancies arise between the stored and calculated ECC values, it indicates potential corruption in the disk sector, prompting the controller to report a recoverable error.

While larger sector sizes can increase user data capacity by reducing the number of headers and trailers, they also decrease the number of sectors that fit on each track. After low-level formatting, the operating system must perform additional steps to prepare the disk for file storage. This involves partitioning the disk into groups of cylinders, allowing the OS to treat each partition as a separate disk.

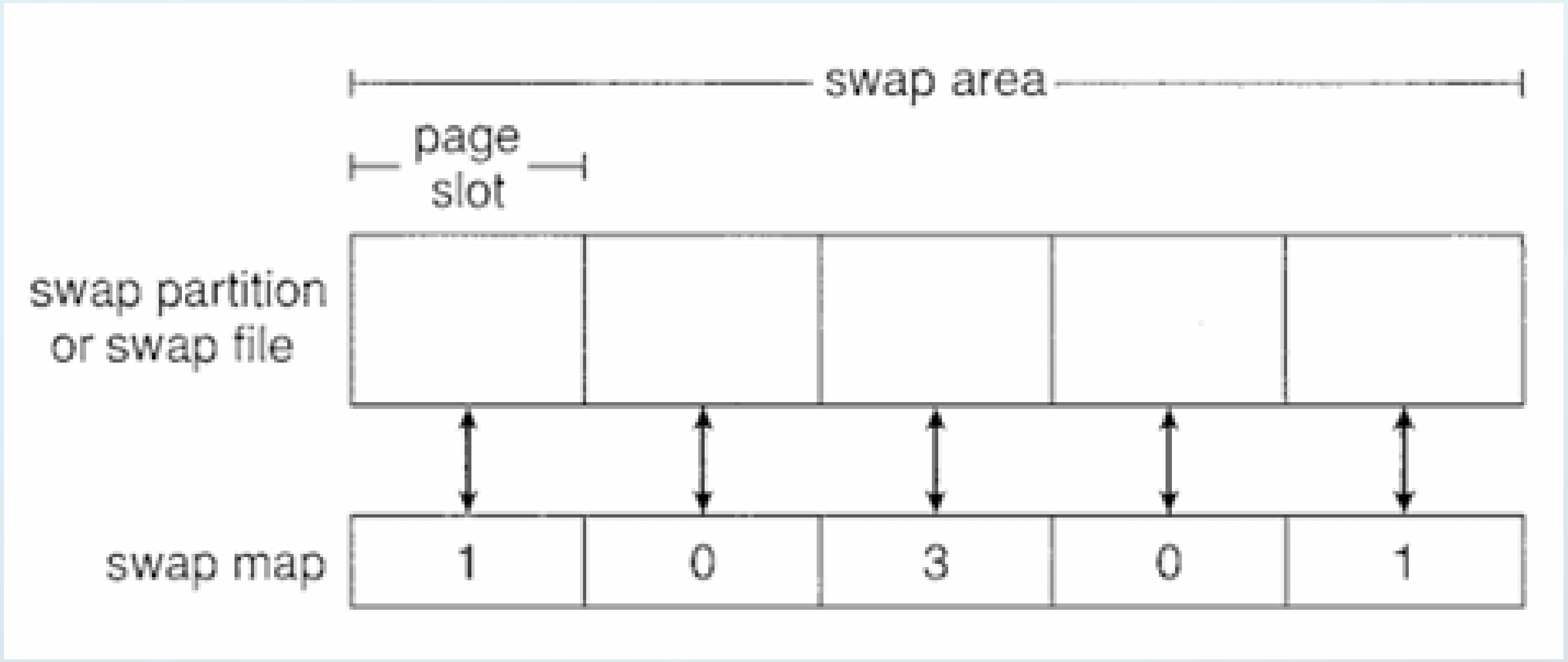
Q. How does external fragmentation impact the performance of swap space when implemented as a file?

Swap space management

Swap space management provides a mechanism for extending physical memory by using disk space. Swap space can be implemented in two primary ways: as a large file within the normal file system or as a dedicated partition. When swap space is created as a file, it leverages standard file-system routines for creation, naming, and space allocation. This method can be inefficient due to the overhead of navigating the directory structure and managing disk allocation data, which can lead to increased swapping times due to external fragmentation.

Creating swap space in a separate partition eliminates the need for a file system or directory structure, allowing a dedicated swap-space storage manager to handle block allocation and deallocation. This manager can employ algorithms optimized for speed, as swap space is accessed more frequently than regular file systems. The raw-partition method requires a fixed amount of swap space during disk partitioning, and expanding this space and the addition of new swap space. Linux OS offers by allowing swapping in both raw partitions and file-system space, enabling administrators to choose between the convenience of file-system management and the performance benefits of raw partition swapping.

Q. How does external fragmentation impact the performance of swap space when implemented as a file? (Cont …)



Data Structure for swapping on Linux system

Q. What are the potential performance implications of implementing RAID functionality in the operating system?

RAID

RAID (Redundant Array of Independent Disks) storage can be configured in several ways to enhance data redundancy, performance, and reliability. One common approach involves directly attaching disks to a system's buses, where the operating system or system software is responsible for implementing RAID functionality. This method allows for flexibility in configuration but may require more resources from the host system.

An intelligent host controller can manage multiple attached disks and implement RAID in hardware. This offloads the RAID processing from the operating system, potentially improving performance and efficiency. A dedicated RAID array, which is a standalone unit equipped with its own controller, cache, and disks. This RAID array connects to the host system via standard interfaces such as ATA, SCSI. This setup provides a significant advantage, lacks built-in RAID functionality to benefit from RAID-protected disks.

Q. How does the choice of RAID implementation affect system performance and flexibility?

RAID Implementation

The implementation of RAID (Redundant Array of Independent Disks) is based on the layers at which it is applied, each offering different levels of performance, flexibility, and cost. One approach is to implement RAID through volume-management software within the operating system's kernel or at the system software layer. This method allows the storage hardware to provide minimal features while still being part of a comprehensive RAID solution. RAID can be slower, which is why configurations like RAID 0, RAID 1, or RAID 0+1 are often preferred for better performance.

A more advanced implementation occurs at the hardware level within a storage array. Storage array can create RAID sets of various levels and even partition these sets into smaller volumes presented to the operating system. This allows the OS to focus solely on implementing the file system on each volume, while the storage array manages the RAID functionality.

RAID can be implemented at the SAN interconnect layer through disk virtualization devices. These devices act as intermediaries between the hosts and the storage, managing access and commands from the servers. They provide mirroring by writing each block to two separate storage devices, enhancing data redundancy and reliability.

Q. What is an SSD, and how does it differ from a traditional hard disk?

SSD

Unlike traditional hard disks, SSDs utilize flash memory, making them more reliable and faster due to the absence of seek time and latency. This nonvolatile memory retains data even when powered off, providing similar characteristics to hard disks but with enhanced performance and energy efficiency. The higher cost per megabyte and lower capacity compared to traditional hard disks limit. SSDs are increasingly being integrated into storage arrays for high-performance tasks, such as managing metadata systems, and are also popular in notebook computers for their compact size and speed.

The holographic storage, for instance, employs laser light to record three-dimensional holograms on specialized media. This method allows for the simultaneous transfer of vast amounts of data, potentially leading to extremely high data transfer rates. The holographic storage become commercially viable, offering a new avenue for data storage. Another innovative approach involves the application of MEMS (Micro-Electro-Mechanical Systems) technology to create small data storage devices. This concept envisions an array of thousands of tiny disk heads that can access data on a magnetic storage medium with remarkable speed and efficiency. If successful, this technology could provide a nonvolatile storage solution that outperforms traditional magnetic disks while being more cost-effective than semiconductor DRAM.

Operating systems must adapt to support various storage media, including removable magnetic disks, DVDs, and magnetic tapes. This includes providing capabilities for data management, file system integration, and ensuring seamless access to removable media for users.

Q. How should the application handle the exclusive access requirement for tape drives to prevent conflicts between multiple applications?

Application Interface

When a blank cartridge is inserted, it needs to be formatted, creating an empty file system. This file system functions like that of a hard disk. Tapes are handled differently. The operating system presents a tape as a raw storage medium. Applications open the entire tape drive as a raw device, which is reserved for exclusive use until the application closes it.

When using a tape as a raw device, the operating system does not provide file-system services. Applications must define their own methods for organizing data on the tape. This can lead to compatibility issues, as different applications may use different formats for storing file names and sizes. For example, a backup program might store a list of file names followed by the actual data, but the format can vary widely.

Tape drives have different basic operations compared to disk drives. Instead of a *seek()* operation, they use *locate()* to position the tape at a specific logical block. Locating to block 0 rewinds the tape. Most tape drives have variable block sizes, and the size is determined when writing. If a defective area is encountered, the drive skips it and rewrites the block, complicating the ability to locate empty spaces.

Tape drives have a *read_position()* operation to return the current logical block number. They support a *space()* operation for relative motion. An end-of-tape (EOT) mark is placed after written blocks, preventing movement past it. While a file system can be implemented on a tape, it would require different structures and algorithms due to the append-only nature of tapes.

References

Images: Internet

Source: Operating System Concepts (8th Edition)