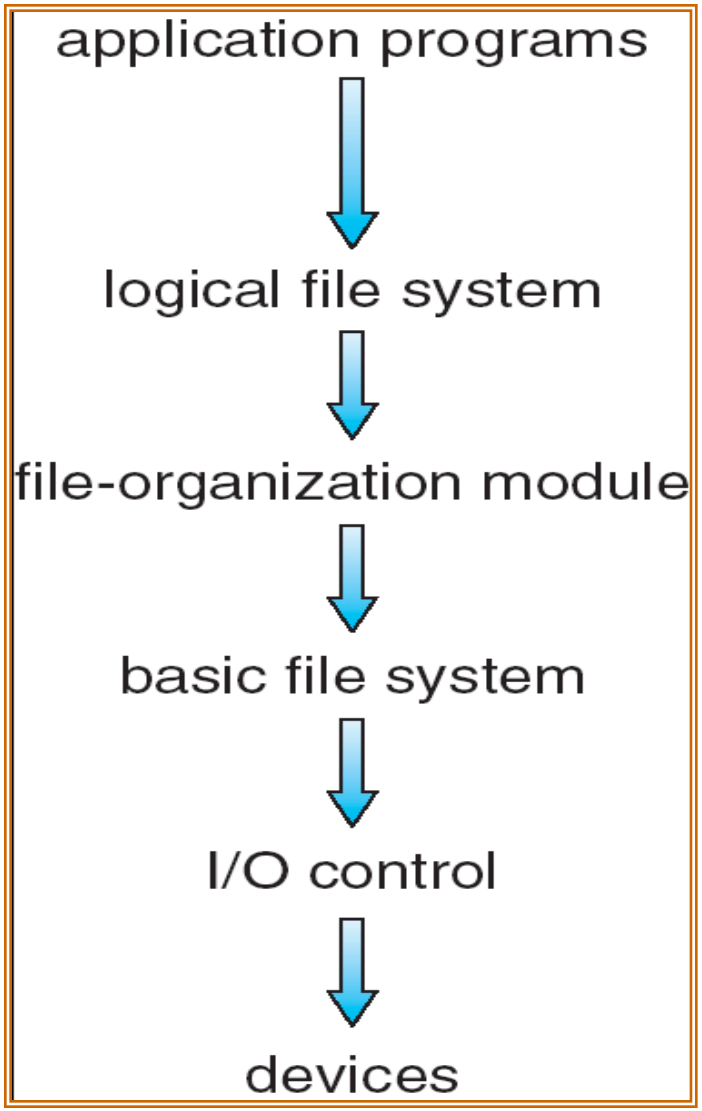
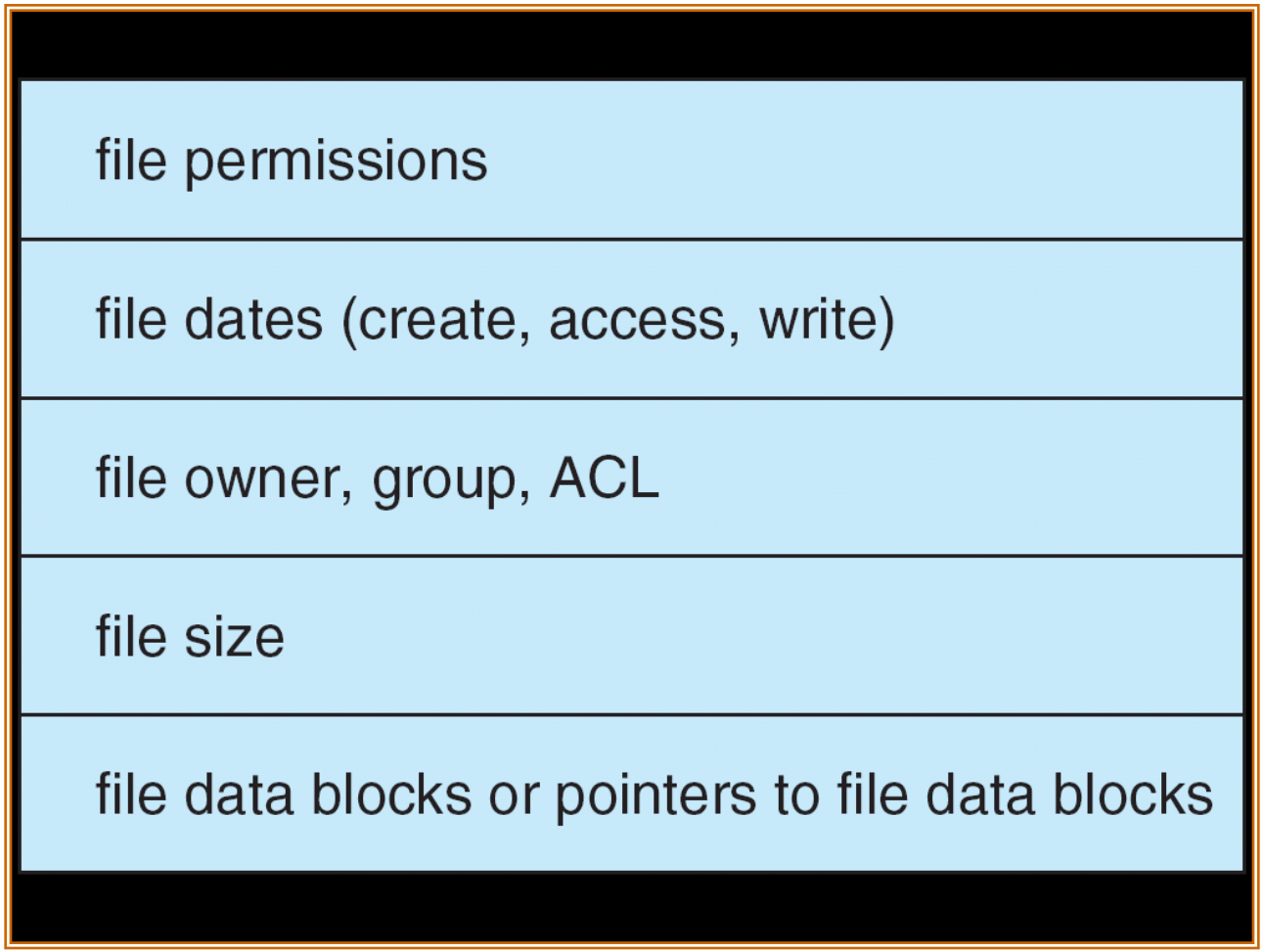
File-System Structure

- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks)
- File system organized into layers
- **File control block** – storage structure consisting of information about a file

Layered File System



A Typical File Control Block

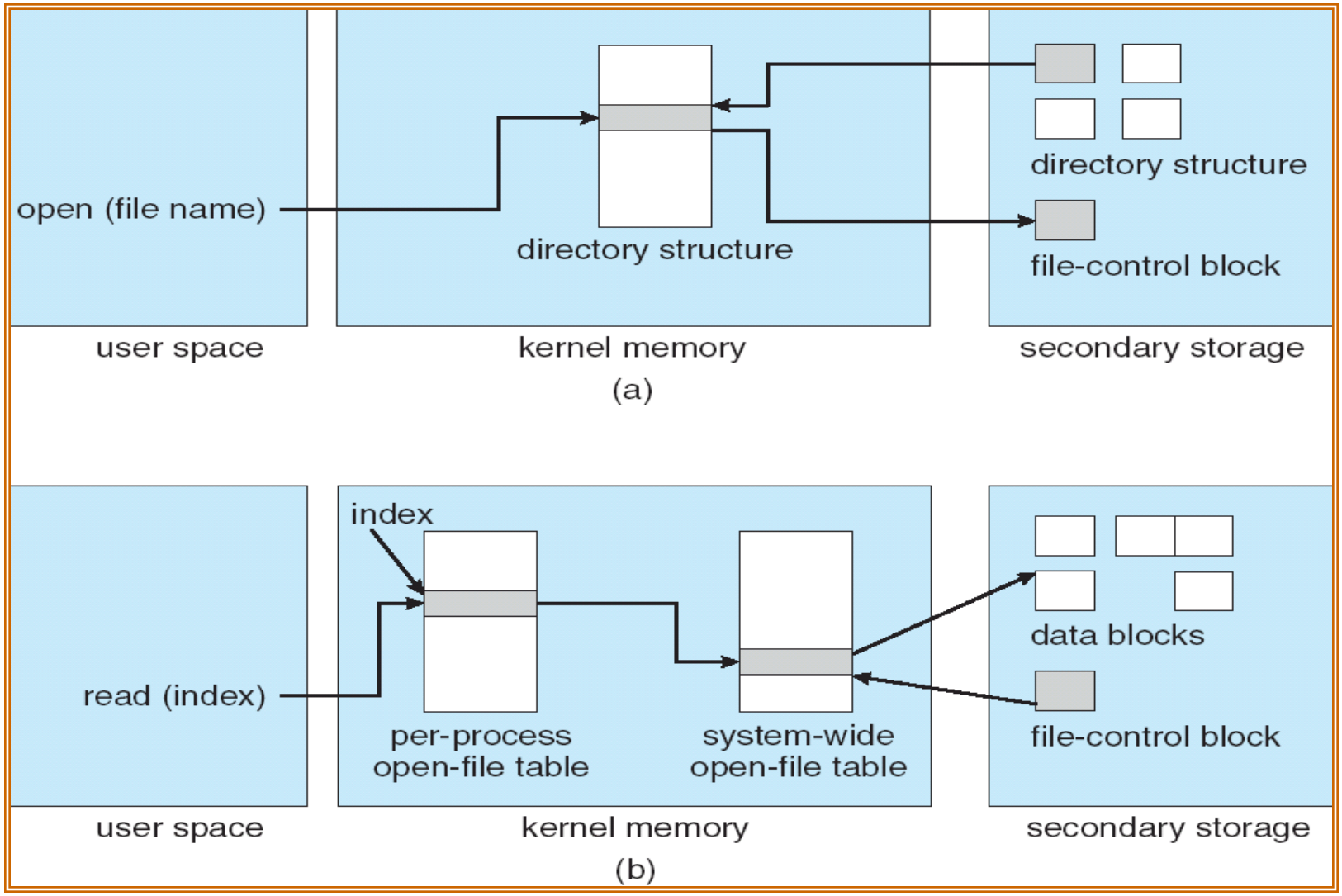


In-Memory File System Structures

- The following figure illustrates the necessary file system structures provided by the operating systems.

- Figure (a) refers to opening a file.
- Figure (b) refers to reading a file.

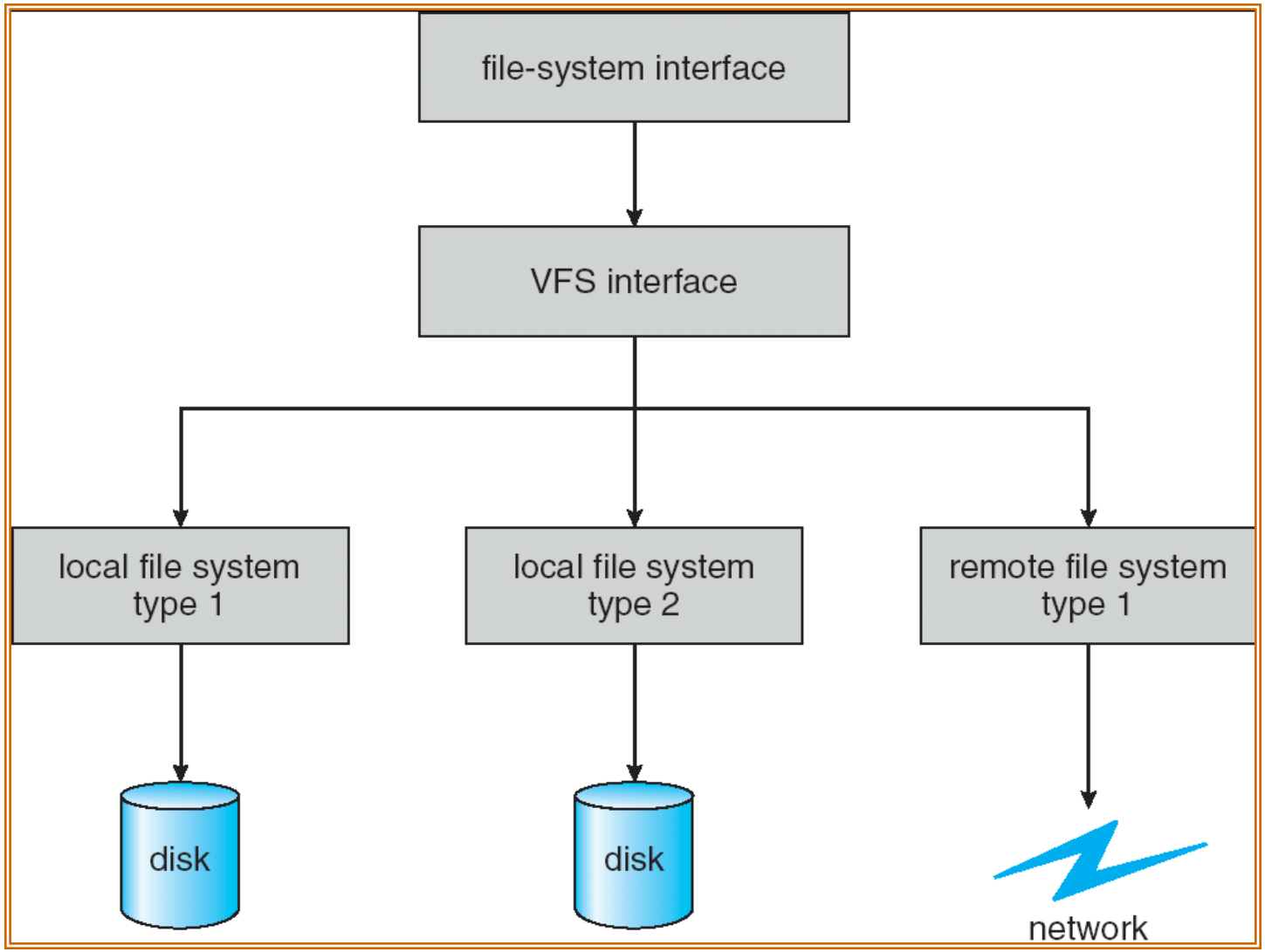
In-Memory File System Structures



Virtual File Systems (Implementation)

- Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems.
- The API is to the VFS interface, rather than any specific type of file system.

Schematic View of Virtual File System



Directory Implementation

- **Linear list** of file names with pointer to the data blocks.
  - simple to program
  - All files are stored/displayed in the form of list
  - time-consuming to execute (Search)
- **Hash Table** – *Linear list with hash* data structure.
  - decreases directory search time
  - So Improve the performance
  - **collisions** – situations where two file names hash to the same location
  - fixed size

Allocation Methods

- An allocation method refers to how *disk blocks are allocated for files* :-

1. Contiguous allocation

2. Linked allocation

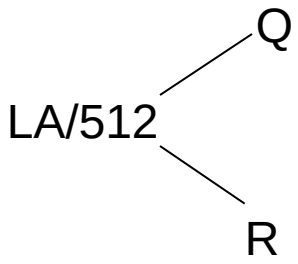
3. Indexed allocation

Contiguous Allocation

- Each file occupies a *set of contiguous blocks* on the disk

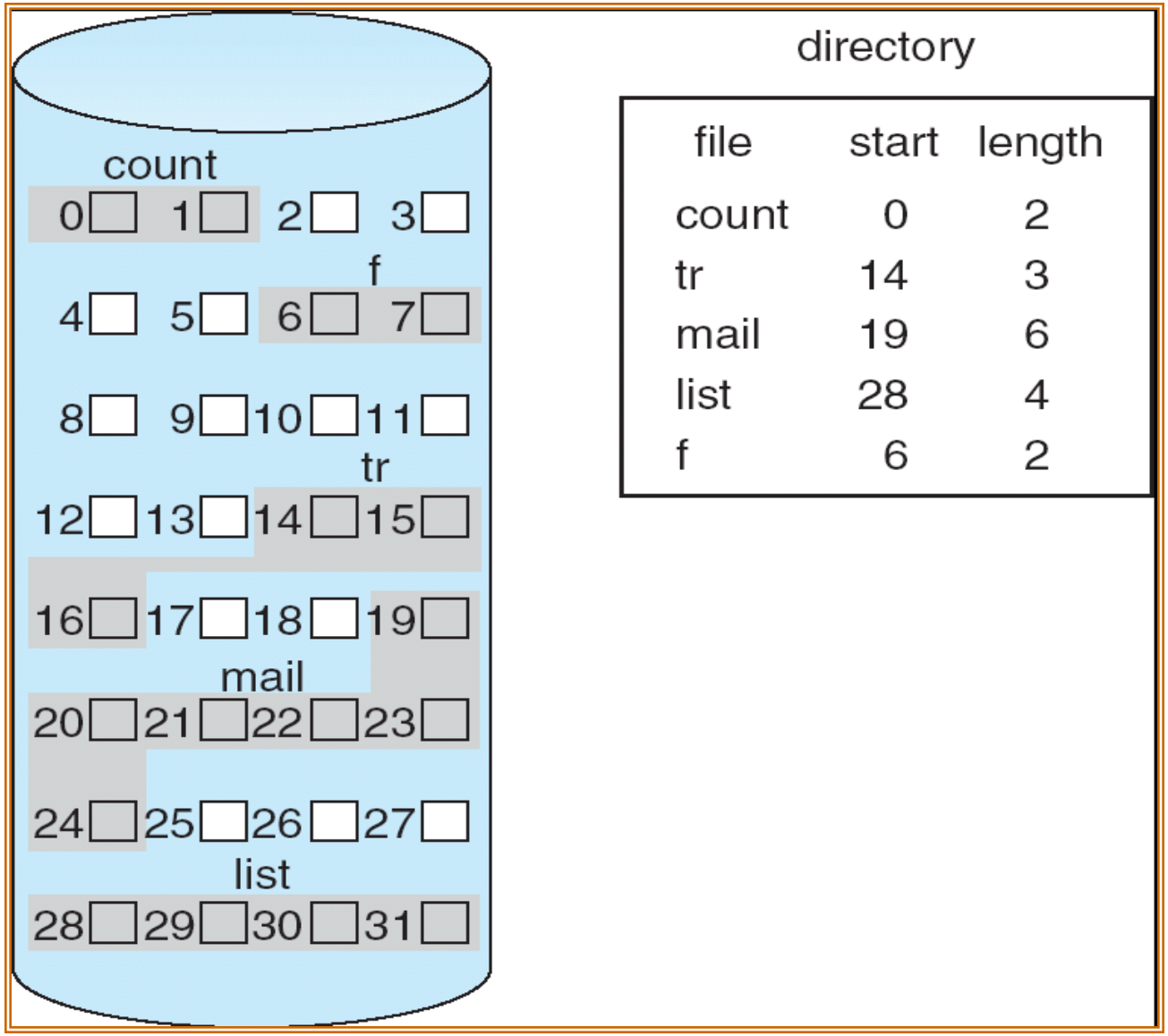
- Simple :– only starting location (block #) and length (number of blocks) are required
- Random access
- Wasteful of space (dynamic storage-allocation problem)
- Files cannot grow (Fixed size)

- Mapping from logical to physical



  - Block to be accessed = ! + starting address
  - Displacement into block = R
  - (I.e. Initial position to final position of an object)

Contiguous Allocation of Disk Space

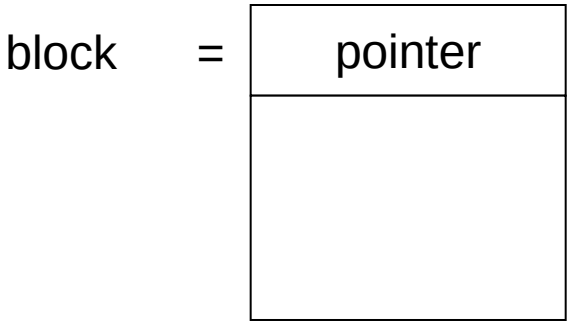


Extent-Based Systems

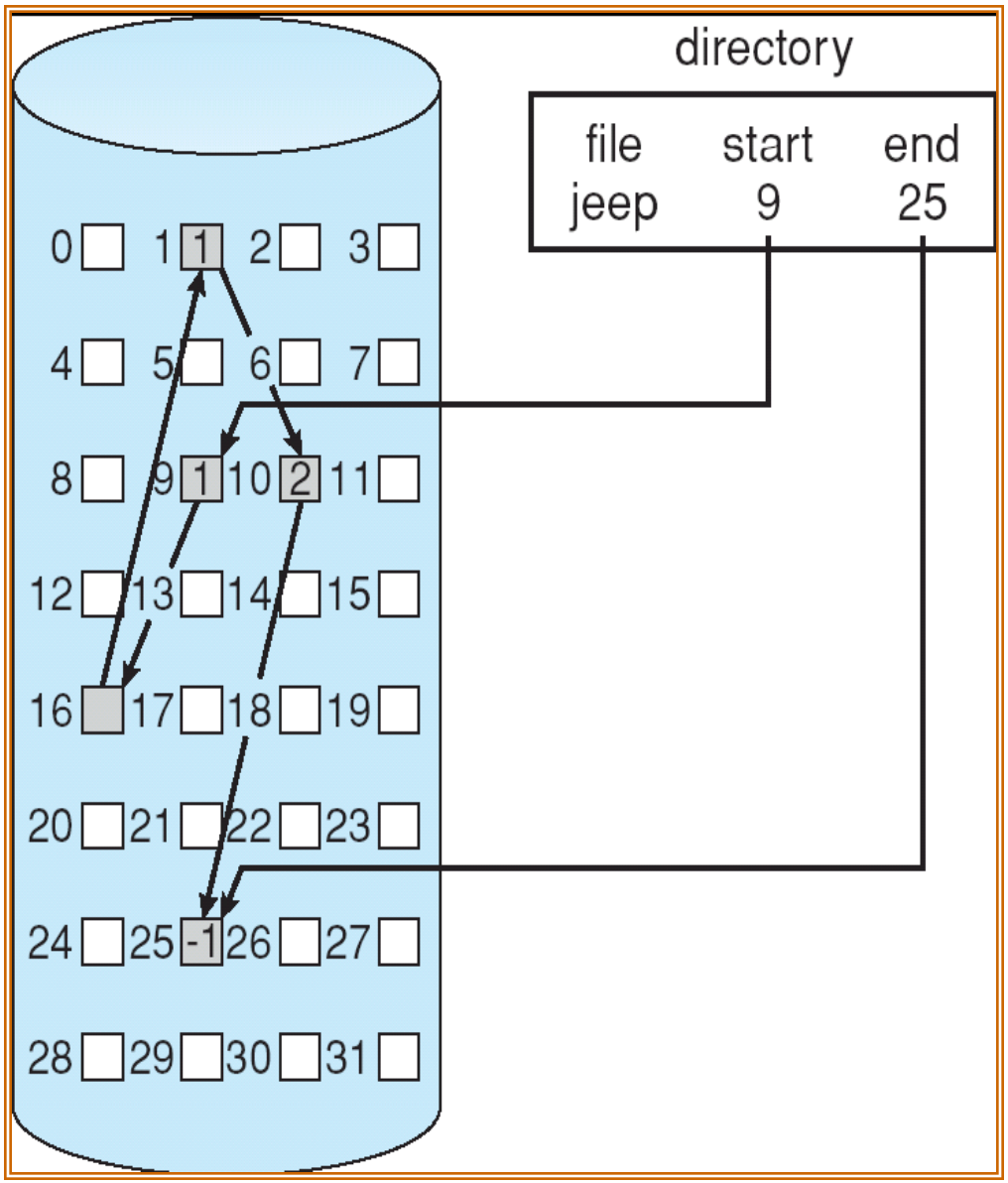
- Many newer file systems (I.e. Veritas File System) use a *modified contiguous* allocation scheme
- Extent-based file systems allocate disk blocks in **extents**
- An **extent** is a contiguous block of disks
  - Extents are allocated for file allocation
  - A file consists of one or more extents.

Linked Allocation

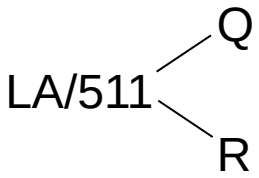
- Each file is a linked list of disk blocks
- Blocks may be scattered anywhere on the disk.



Linked Allocation

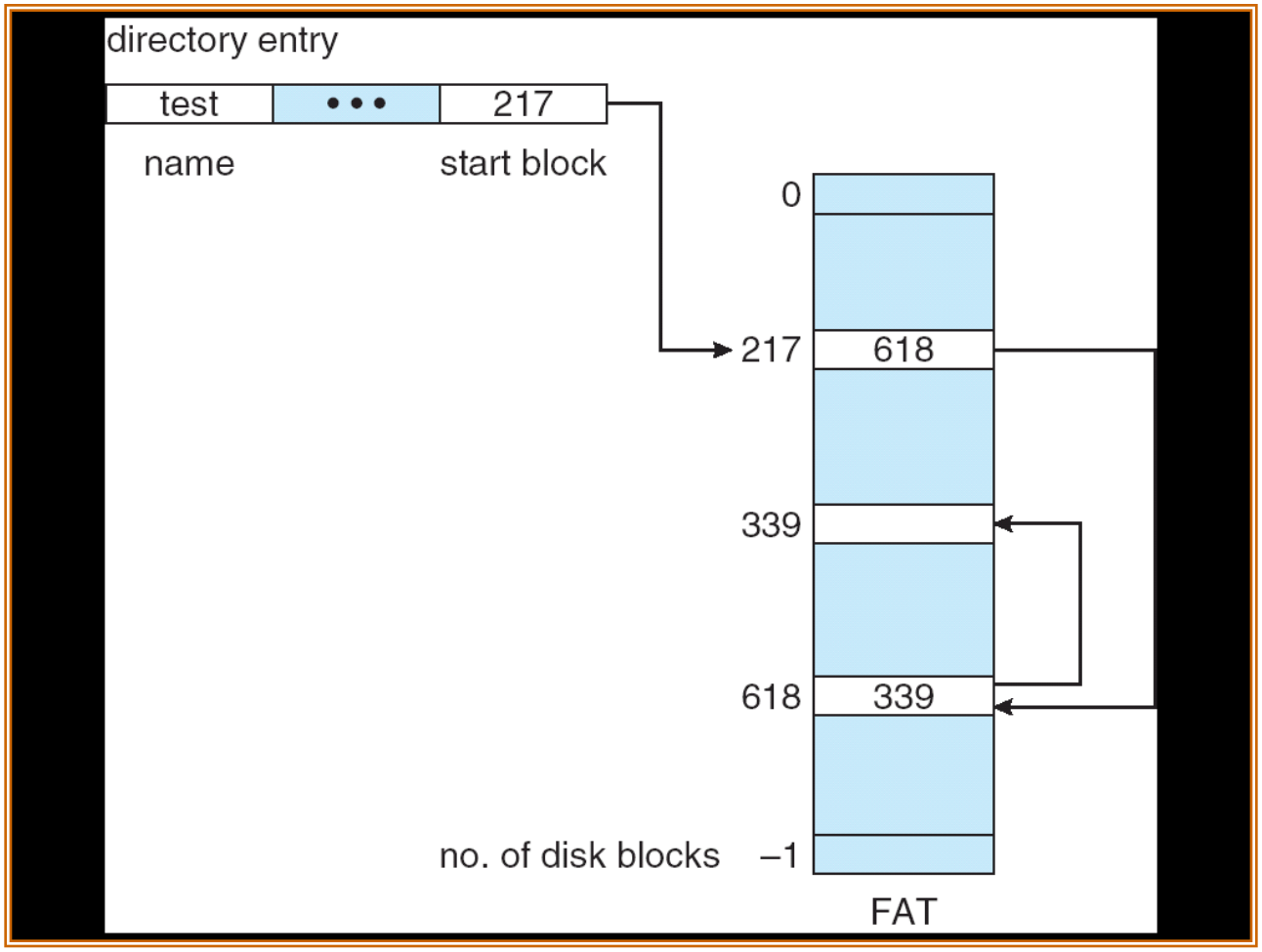


- Simple – need only starting address
- Free-space management system – no waste of space
- No random access
- Mapping



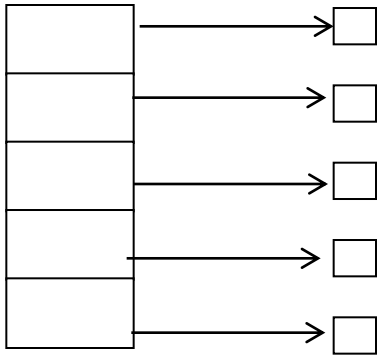
  - Block to be accessed is the Q th block in the linked chain of blocks representing the file.
  - Displacement into block = R + 1
- File-allocation table (FAT) – disk-space allocation used by MS-DOS and OS/2.
- Use clustering concept: for 4 free blocks are grouped in to a single block called as cluster.

File-Allocation Table



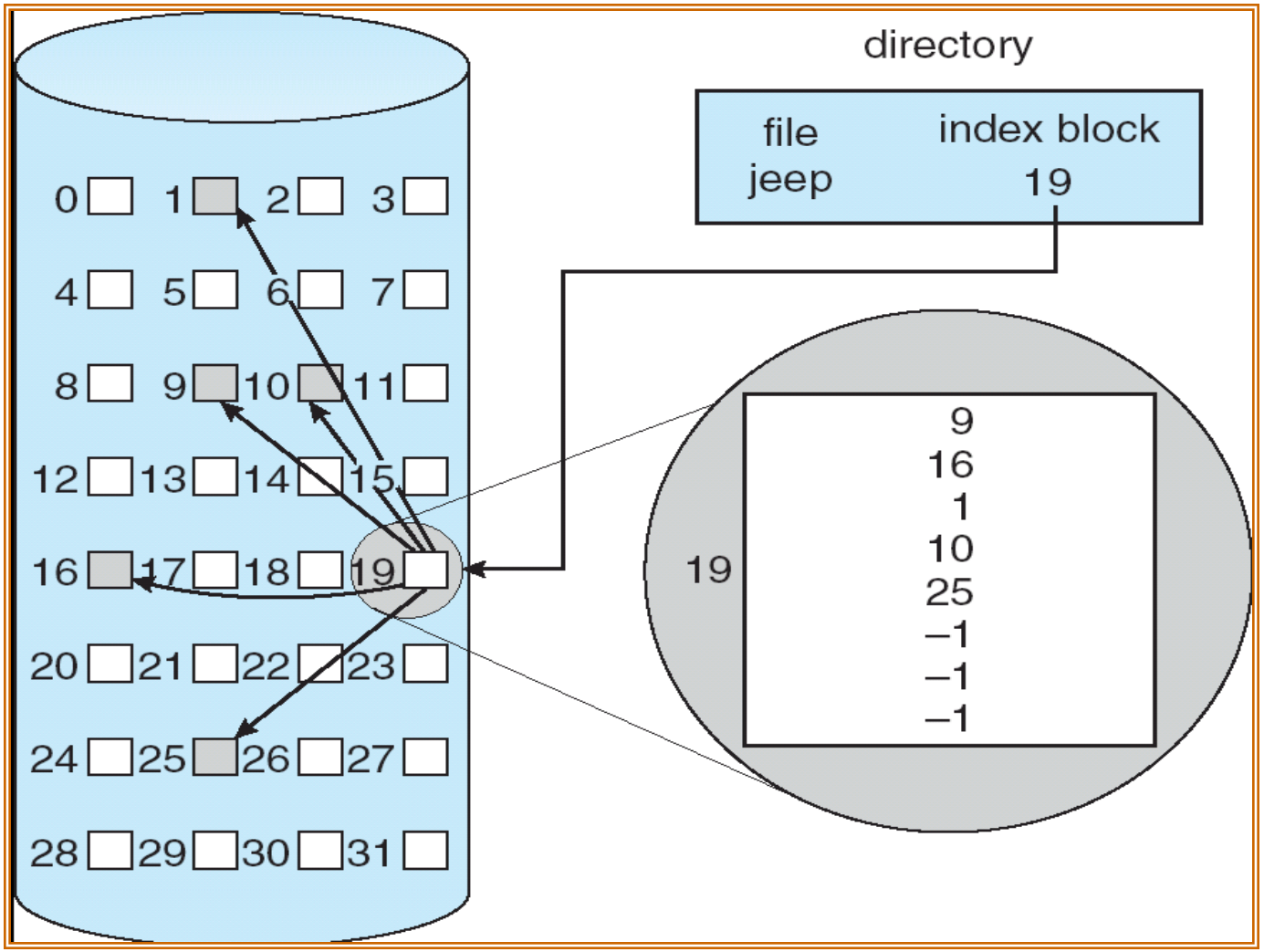
Indexed Allocation

- Brings all pointers together into the *index block.*
- Logical view
-



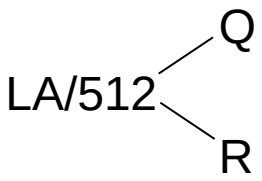
index table

Example of Indexed Allocation



Indexed Allocation (Cont.)

- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block.
- Mapping from logical to physical in a file of maximum size of *256K words* and block size of *512* words.
- We need only 1 block for index table.

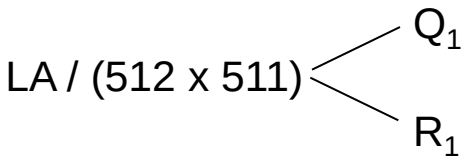


Q = displacement into index table

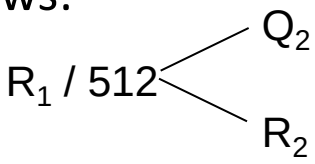
R = displacement into block

Indexed Allocation – Mapping (Cont.)

- Mapping from logical to physical in a file of unbounded length (block size of 512 words).
- Linked scheme – Link blocks of index table (no limit on size).



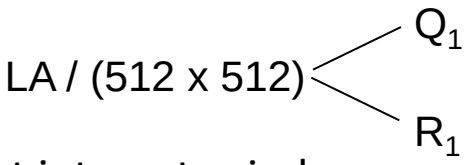
  - Q1 = block of index table
  - R1 is used as follows:-



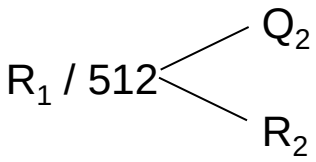
  - *Q*2 = displacement into block of index table
  - *R*2 displacement into block of file:

Indexed Allocation – Mapping (Cont.)

- Two-level index (maximum file size is 512^3)

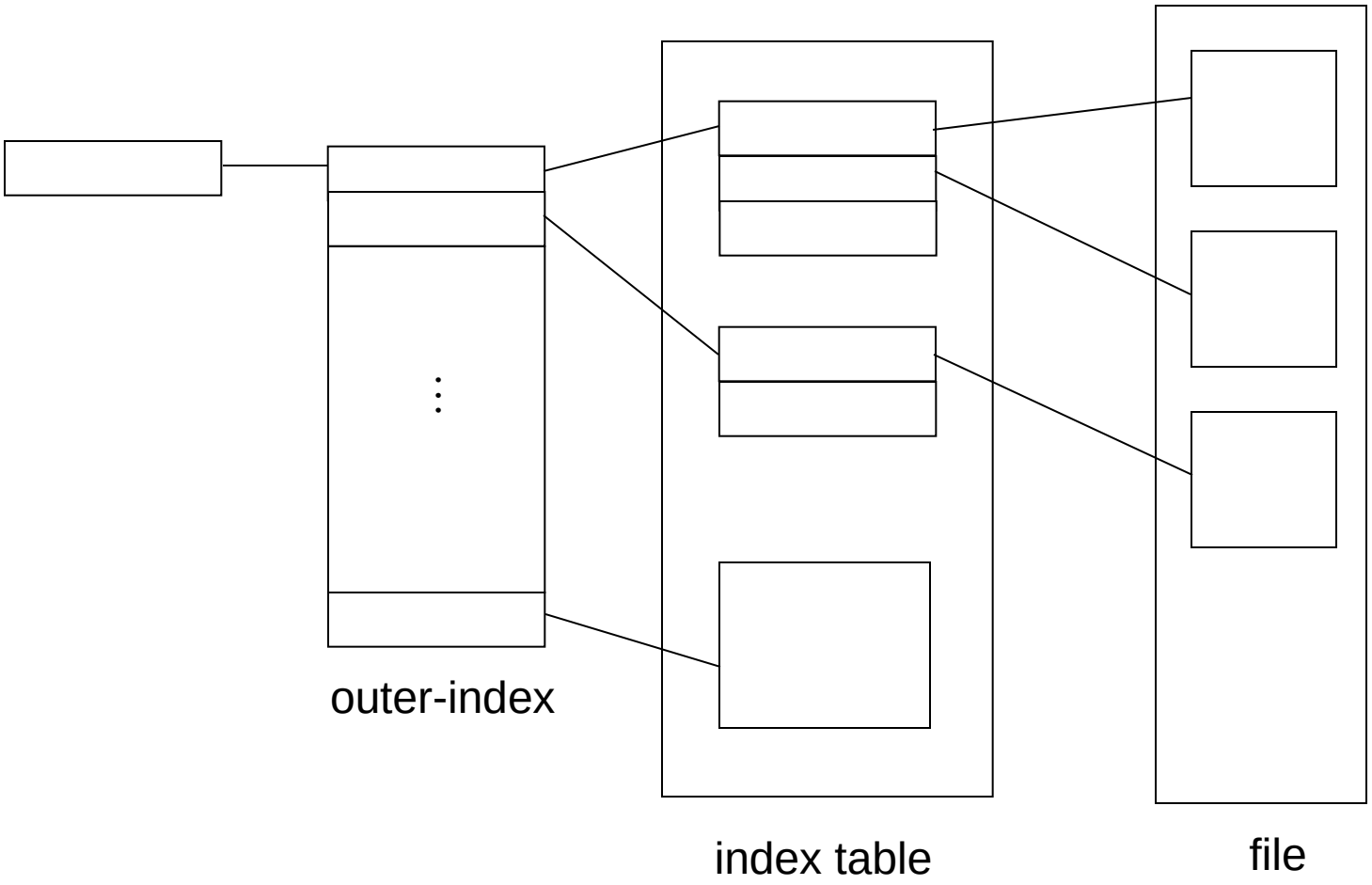


  - Q1 = displacement into outer-index
  - R1 is used as follows:-

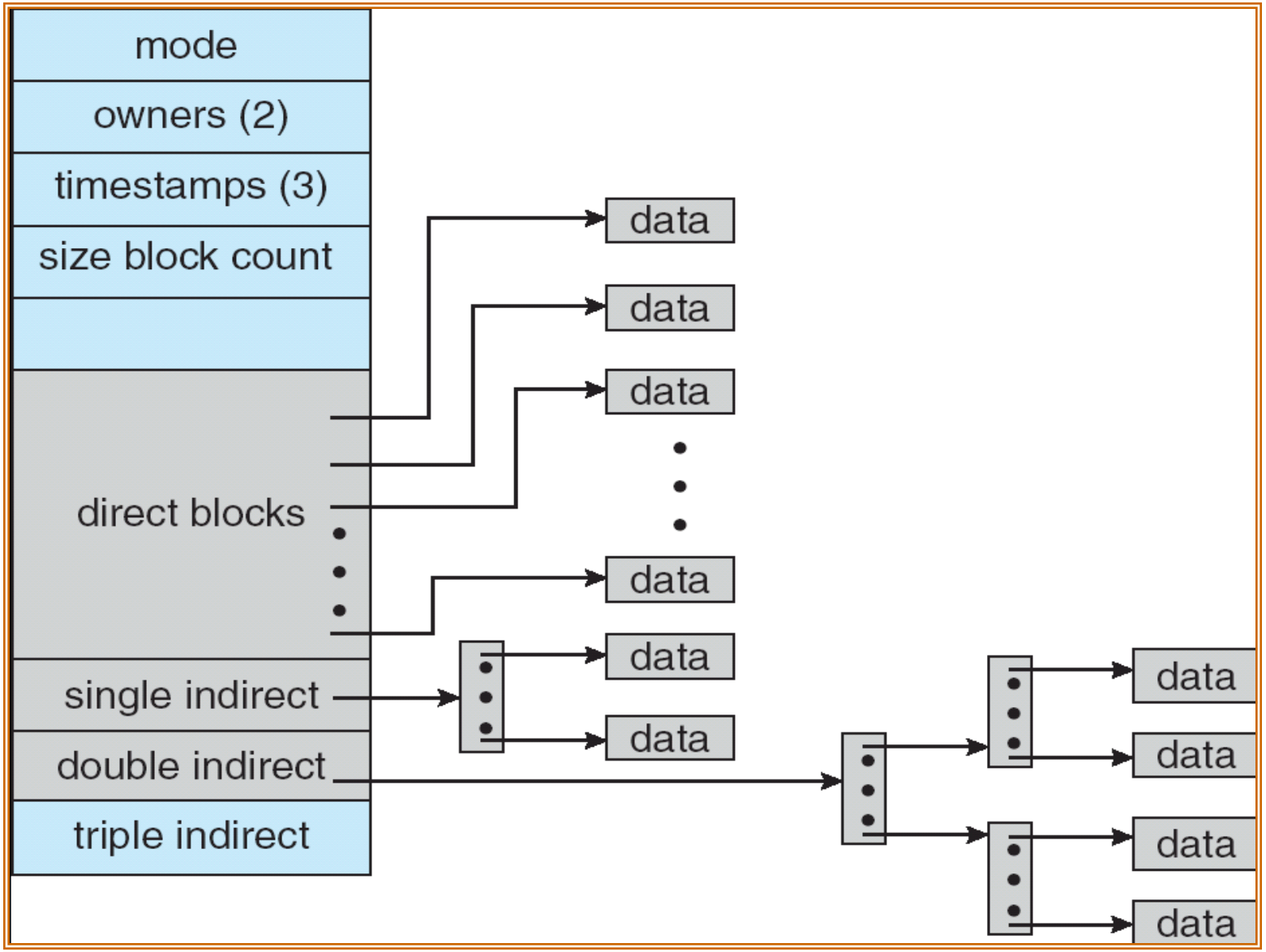


  - Q2 = displacement into block of index table
  - R2 displacement into block of file:

Indexed Allocation – Mapping (Cont.)

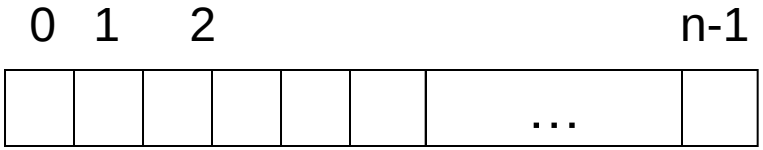


Combined Scheme: UNIX (4K bytes per block)



Free-Space Management

- **Bit vector (*n* blocks)**



$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ free} \\ 1 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$

Block number calculation

(number of bits per word) *
(number of 0-value words) +
offset of first 1 bit

- Bit map requires extra space
  - Example:

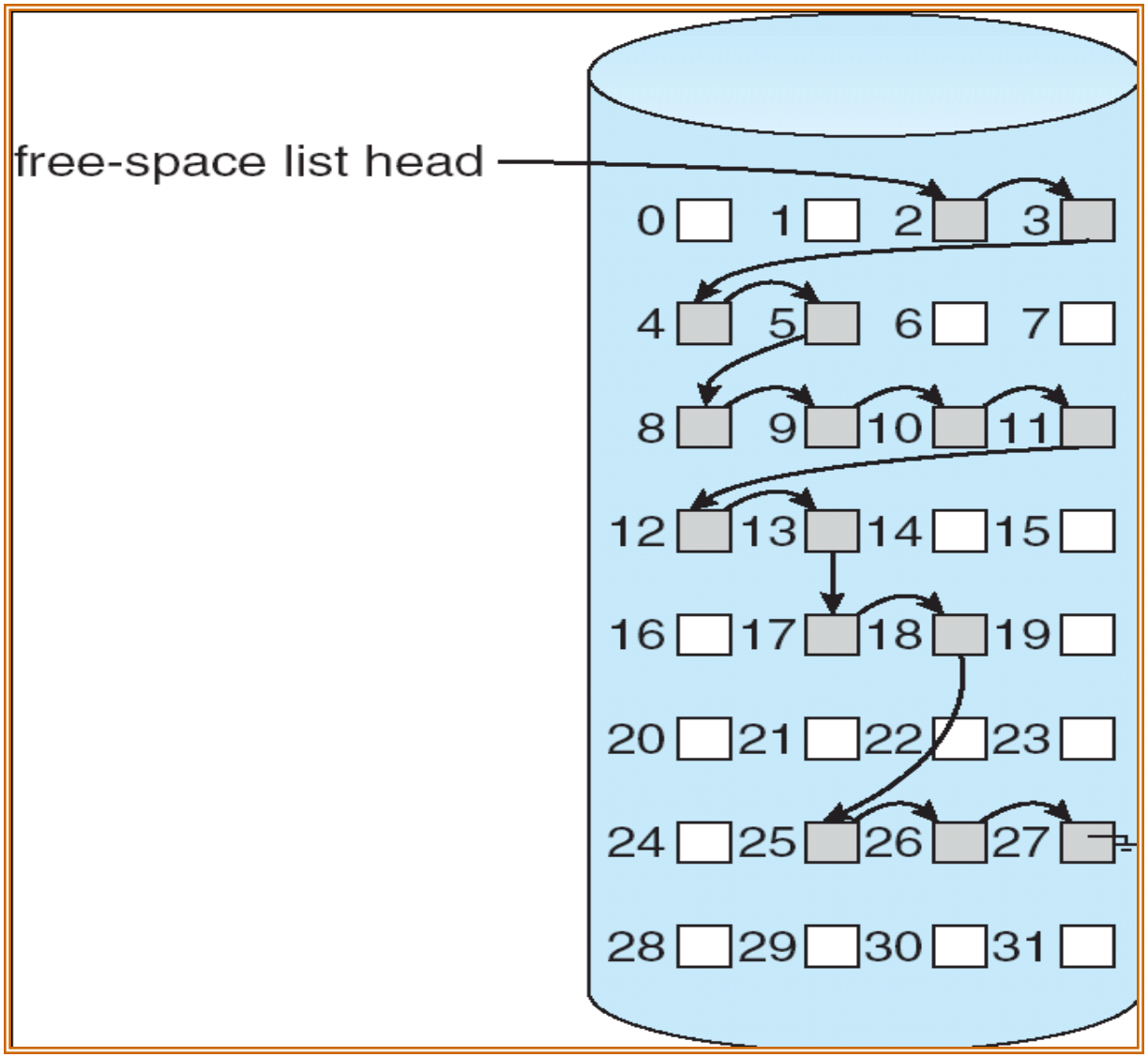
    block size = 2^{12} bytes

    disk size = 2^{30} bytes (1 gigabyte)

    $n = 2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)
- Easy to get *contiguous* files:-
- Linked list (free list):-
  - Cannot get contiguous space easily
  - No waste of space
- Grouping
- Counting

- Need to protect:-
  - Pointer to free list
  - Bit map
    - Must be kept on disk
    - Copy in memory and disk may differ
    - Cannot allow for block[*i*] to have a situation where bit[*i*] = 1 in memory and bit[*i*] = 0 on disk
  - Solution:
    - Set bit[*i*] = 1 in disk
    - Allocate block[*i*]
    - Set bit[*i*] = 1 in memory

Linked Free Space List on Disk



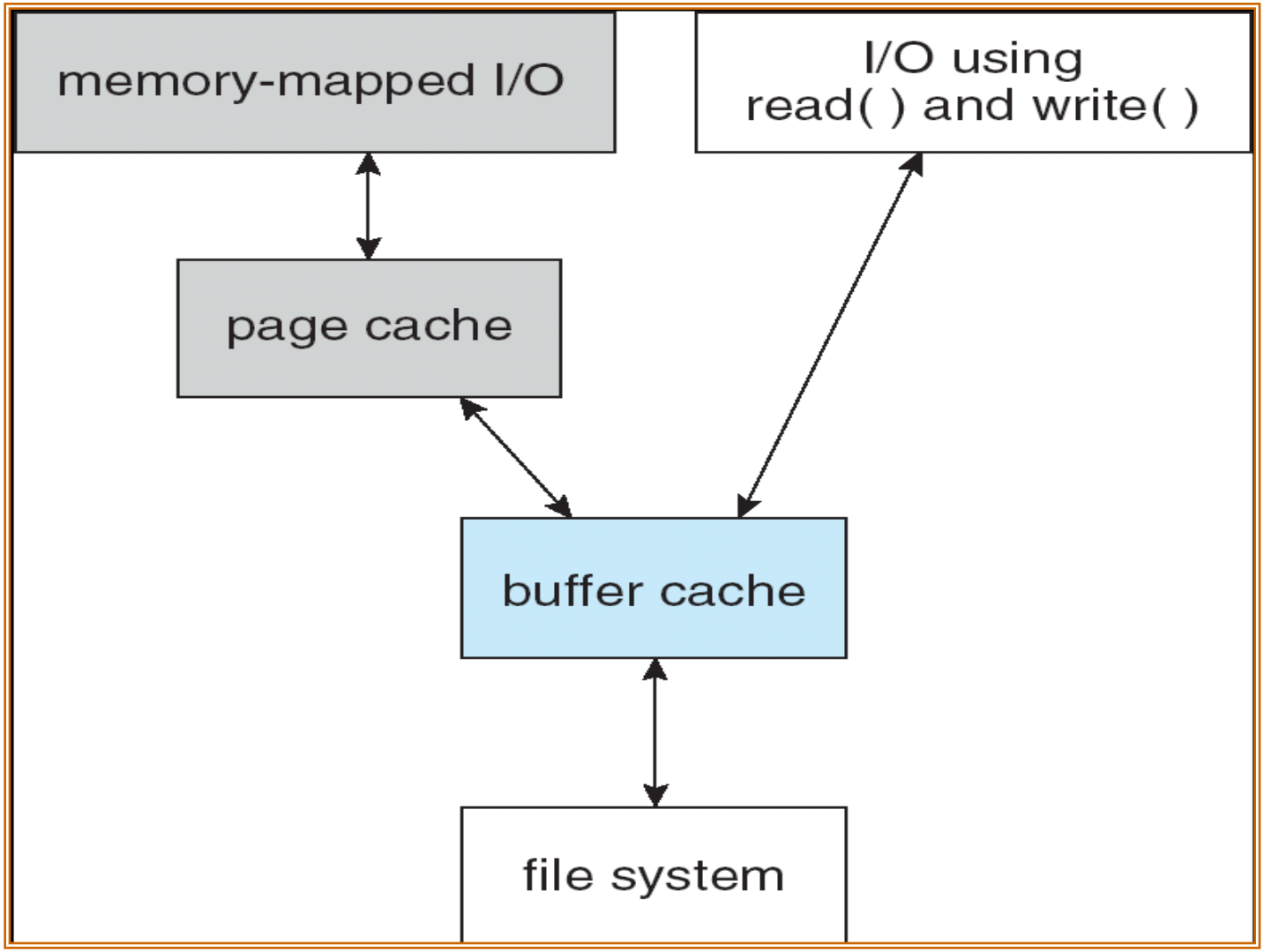
Efficiency and Performance

- Efficiency dependent on:
  - disk allocation and directory algorithms
  - types of data kept in file's directory entry
- Performance
  - disk cache – separate section of main memory for frequently used blocks
  - free-behind and read-ahead – techniques to optimize sequential access
  - improve PC performance by dedicating section of memory as *virtual disk, or RAM disk*

Page Cache

- A **page cache** caches pages rather than disk blocks using virtual memory techniques
- Memory-mapped I/O uses a page cache
- Routine I/O through the file system uses the buffer (disk) cache
- This leads to the following figure

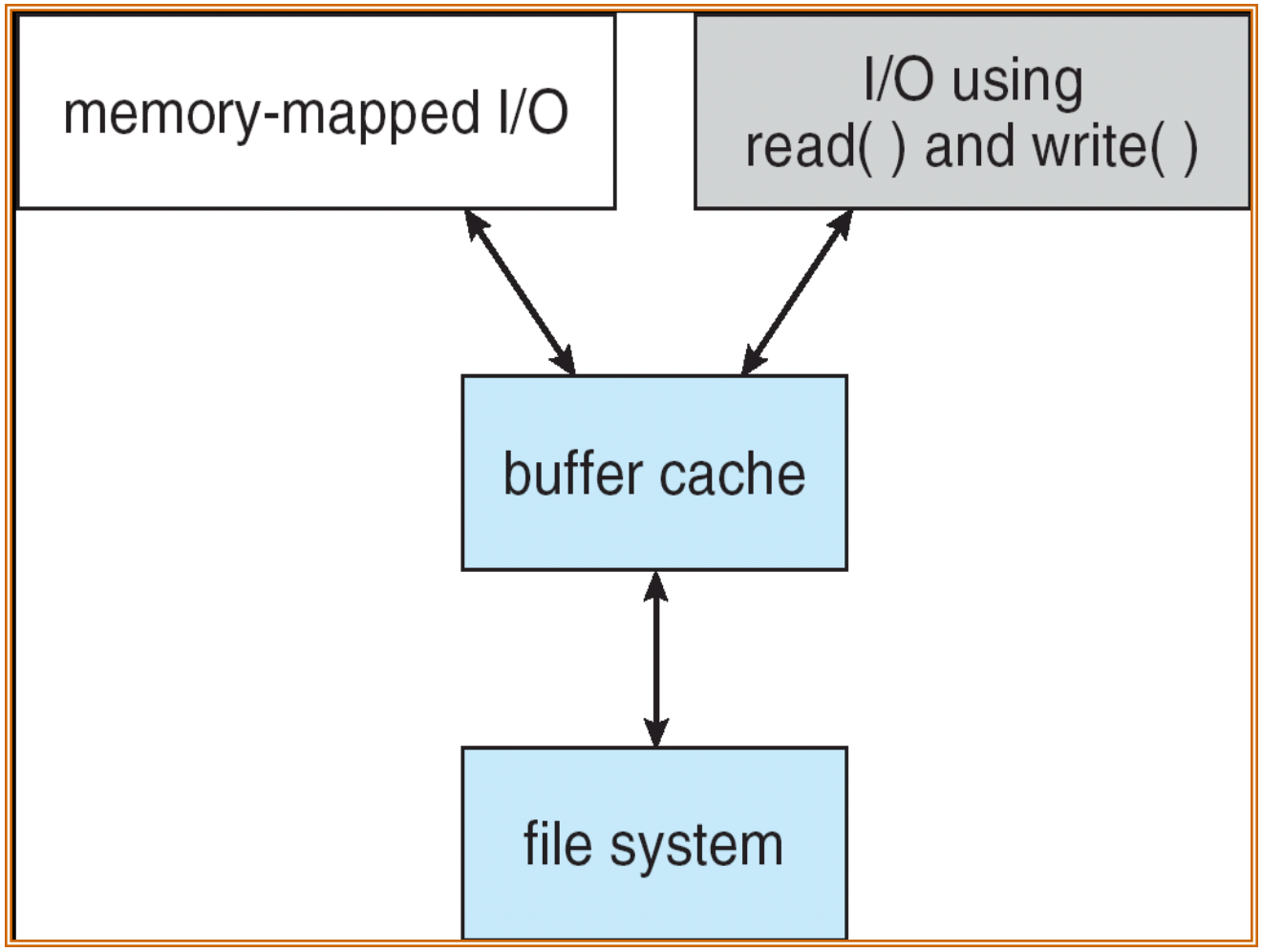
I/O Without a Unified Buffer Cache



Unified Buffer Cache

A unified buffer cache uses the same page cache to cache both memory-mapped pages and ordinary file system I/O

I/O Using a Unified Buffer Cache



Log Structured File Systems

- **Log structured** (or journaling) file systems record each update to the file system as a **transaction**

- All transactions are written to a **log**
  - A transaction is considered **committed** once it is written to the log
  - However, the file system may not yet be updated

- The transactions in the log are asynchronously written to the file system
  - When the file system is modified, the transaction is removed from the log

- If the file system crashes, all remaining transactions in the log must still be performed

The Sun Network File System (NFS)

- An implementation and a specification of a software system for accessing remote files across LANs (or WANs)
- The implementation is part of the Solaris and SunOS operating systems running on Sun workstations using an unreliable datagram protocol (UDP/IP protocol and Ethernet

NFS (Cont.)

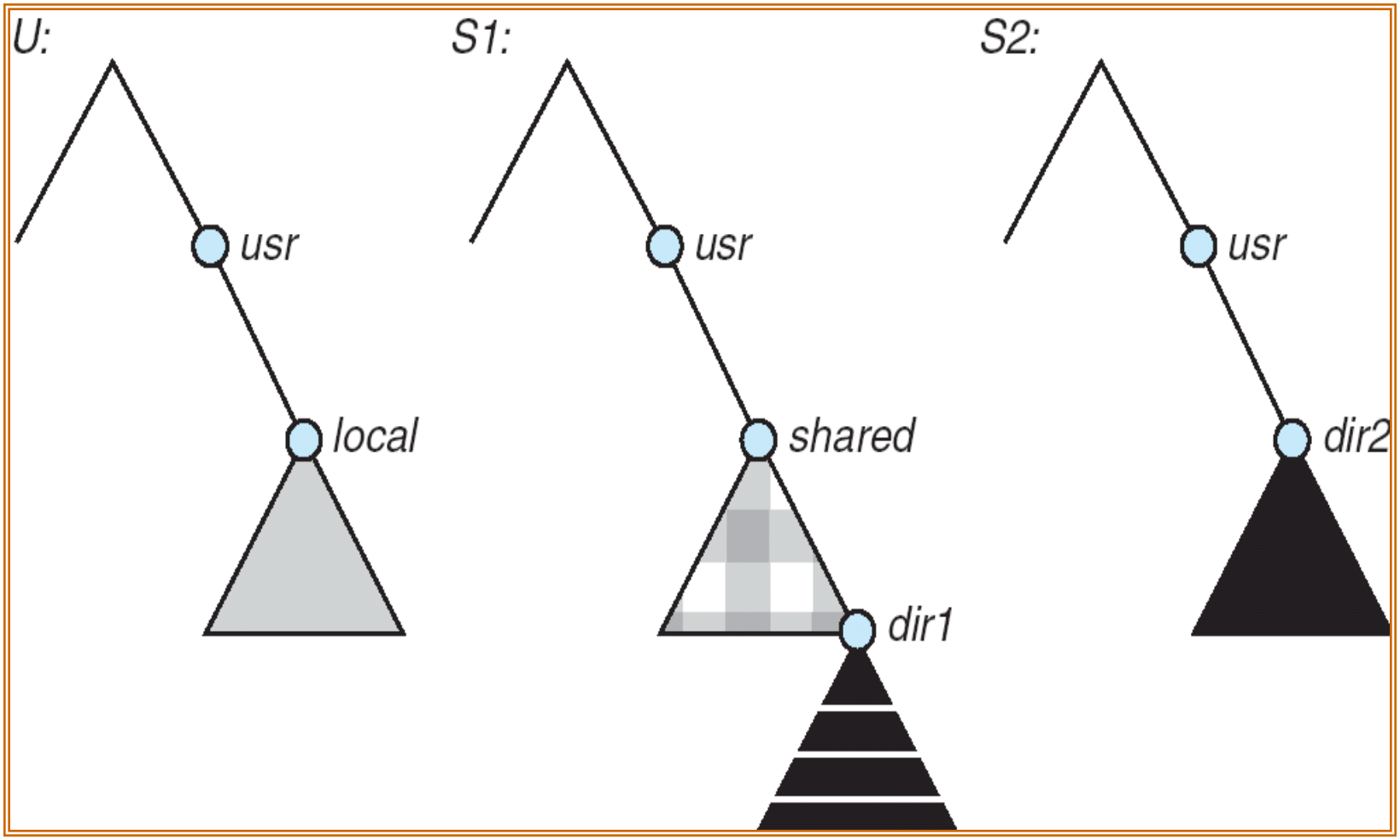
- Interconnected workstations viewed as a set of independent machines with independent file systems, which allows sharing among these file systems in a transparent manner
  - A remote directory is mounted over a local file system directory
    - The mounted directory looks like an integral subtree of the local file system, replacing the subtree descending from the local directory
  - Specification of the remote directory for the mount operation is nontransparent; the host name of the remote directory has to be provided
    - Files in the remote directory can then be accessed in a transparent manner
  - Subject to access-rights accreditation, potentially any file system (or directory within a file system), can be mounted remotely on top of any local directory

- NFS is designed to operate in a heterogeneous environment of different machines, operating systems, and network architectures; the NFS specifications independent of these media

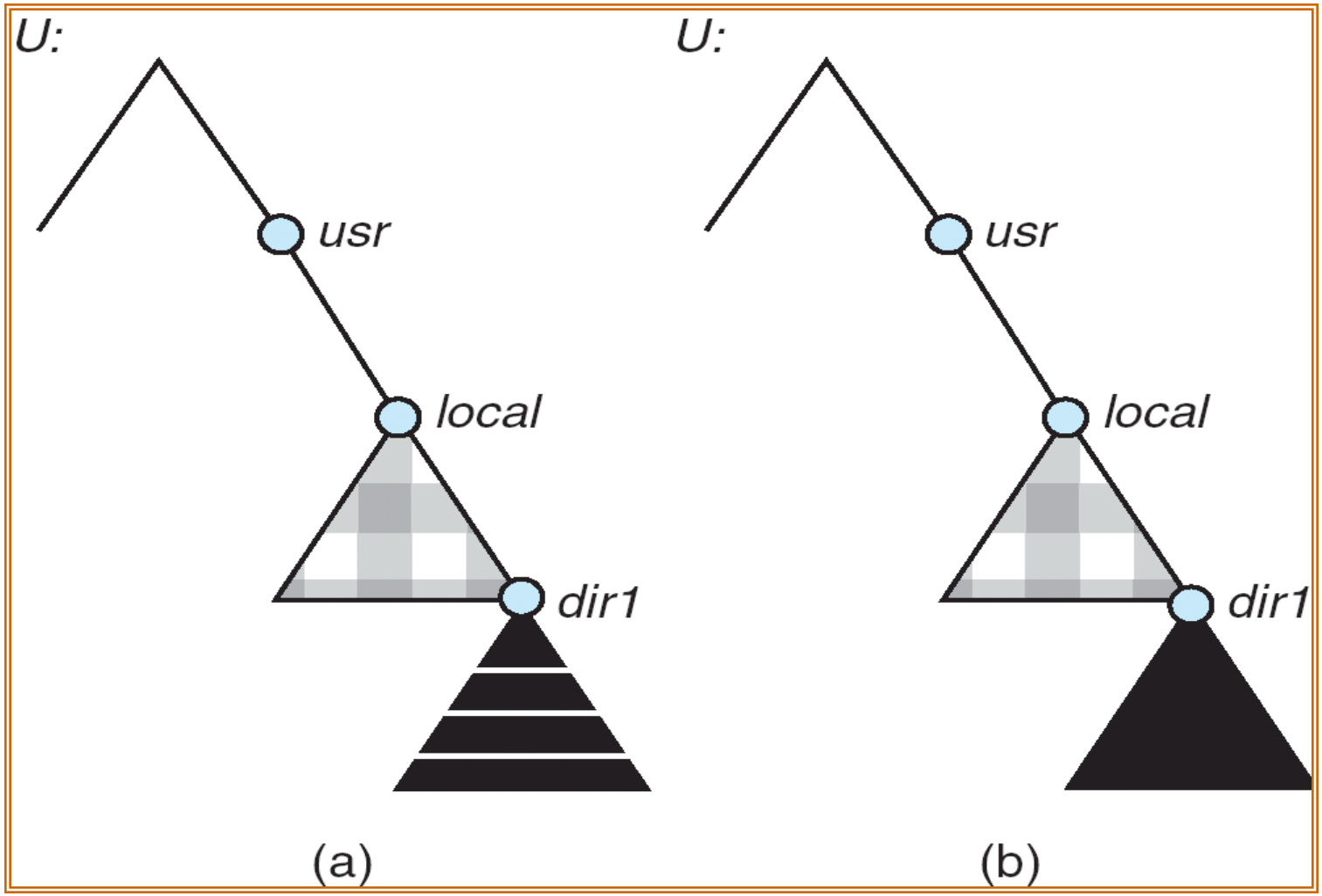
- This independence is achieved through the use of RPC primitives built on top of an External Data Representation (XDR) protocol used between two implementation-independent interfaces

- The NFS specification distinguishes between the services provided by a mount mechanism and the actual remote-file-access services

Three Independent File Systems



Mounting in NFS



Mounts

Cascading mounts

NFS Mount Protocol

- Establishes initial logical connection between server and client
- Mount operation includes name of remote directory to be mounted and name of server machine storing it
  - Mount request is mapped to corresponding RPC and forwarded to mount server running on server machine
  - Export list – specifies local file systems that server exports for mounting, along with names of machines that are permitted to mount them
- Following a mount request that conforms to its export list, the server returns a file handle—a key for further accesses
- File handle – a file-system identifier, and an inode number to identify the mounted directory within the exported file system
- The mount operation changes only the user's view and does not affect the server side

NFS Protocol

- Provides a set of remote procedure calls for remote file operations. The procedures support the following operations:
  - searching for a file within a directory
  - reading a set of directory entries
  - manipulating links and directories
  - accessing file attributes
  - reading and writing files
- NFS servers are **stateless**; each request has to provide a full set of arguments
  (NFS V4 is just coming available – very different, stateful)
- Modified data must be committed to the server's disk before results are returned to the client (lose advantages of caching)
- The NFS protocol does not provide concurrency-control mechanisms

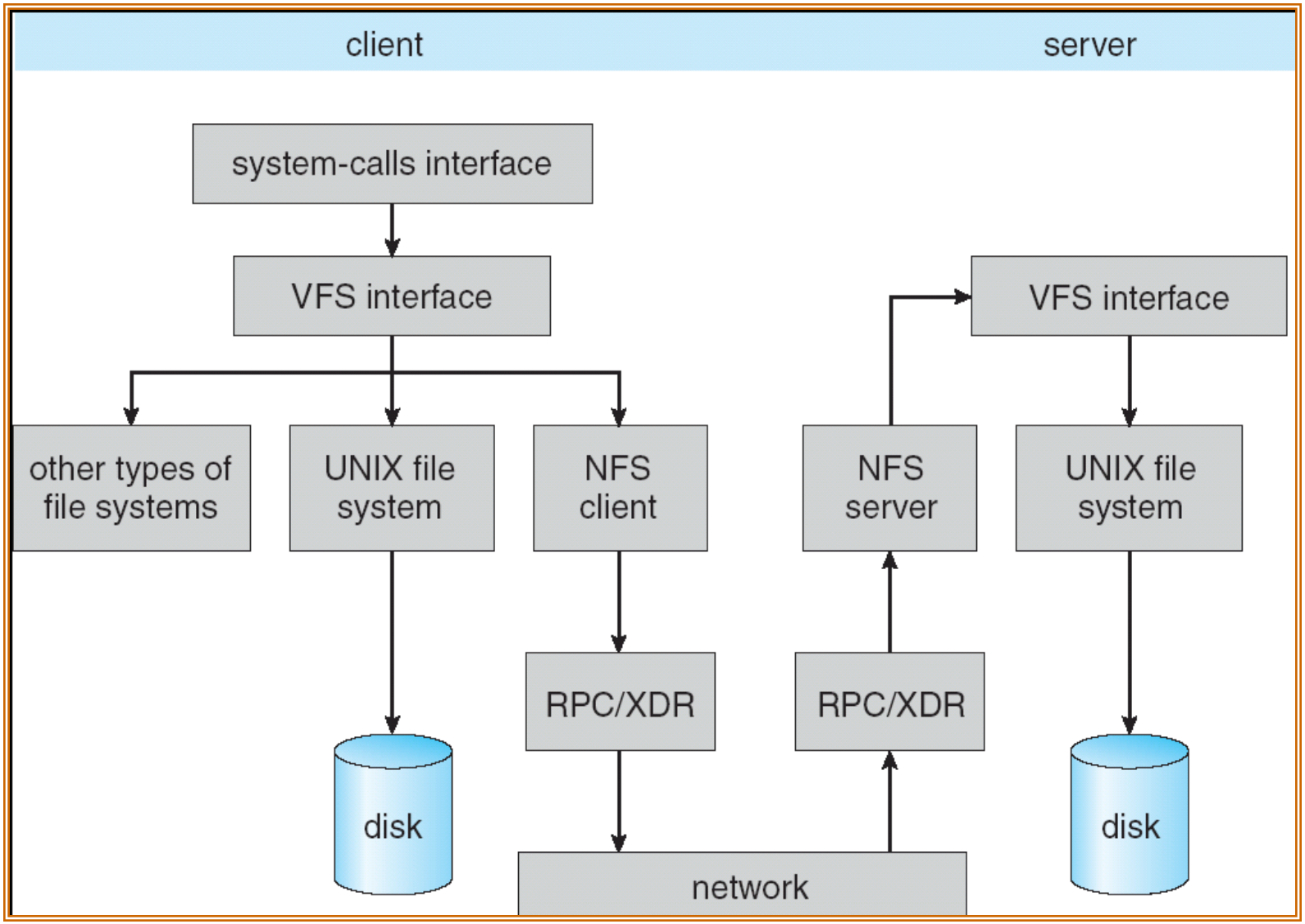
Three Major Layers of NFS Architecture

- UNIX file-system interface (based on the **open, read, write**, and **close** calls, and **file descriptors**)

- *Virtual File System* (VFS) layer – distinguishes local files from remote ones, and local files are further distinguished according to their file-system types
  - The VFS activates file-system-specific operations to handle local requests according to their file-system types
  - Calls the NFS protocol procedures for remote requests

- NFS service layer – bottom layer of the architecture
  - Implements the NFS protocol

Schematic View of NFS Architecture



NFS Path-Name Translation

- Performed by breaking the path into component names and performing a separate NFS lookup call for every pair of component name and directory vnode

- To make lookup faster, a directory name lookup cache on the client's side holds the vnodes for remote directory names

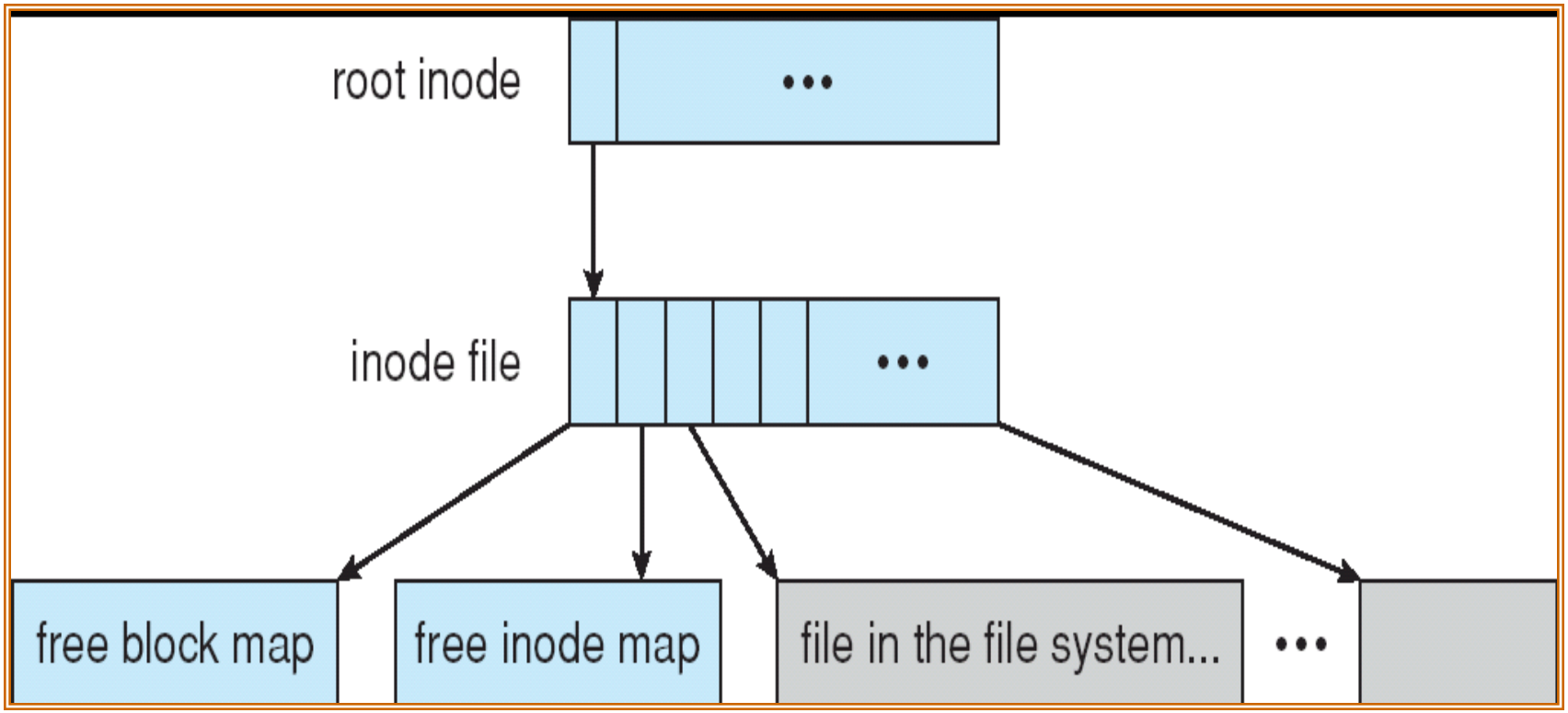
NFS Remote Operations

- Nearly one-to-one correspondence between regular UNIX system calls and the NFS protocol RPCs (except opening and closing files)
- NFS adheres to the remote-service paradigm, but employs buffering and caching techniques for the sake of performance
- File-blocks cache – when a file is opened, the kernel checks with the remote server whether to fetch or revalidate the cached attributes
  - Cached file blocks are used only if the corresponding cached attributes are up to date
- File-attribute cache – the attribute cache is updated whenever new attributes arrive from the server
- Clients do not free delayed-write blocks until the server confirms that the data have been written to disk

Example: WAFL File System

- Used on Network Appliance “Filers” – distributed file system appliances
- “Write-anywhere file layout”
- Serves up NFS, CIFS, http, ftp
- Random I/O optimized, write optimized
  - NVRAM for write caching
- Similar to Berkeley Fast File System, with extensive modifications

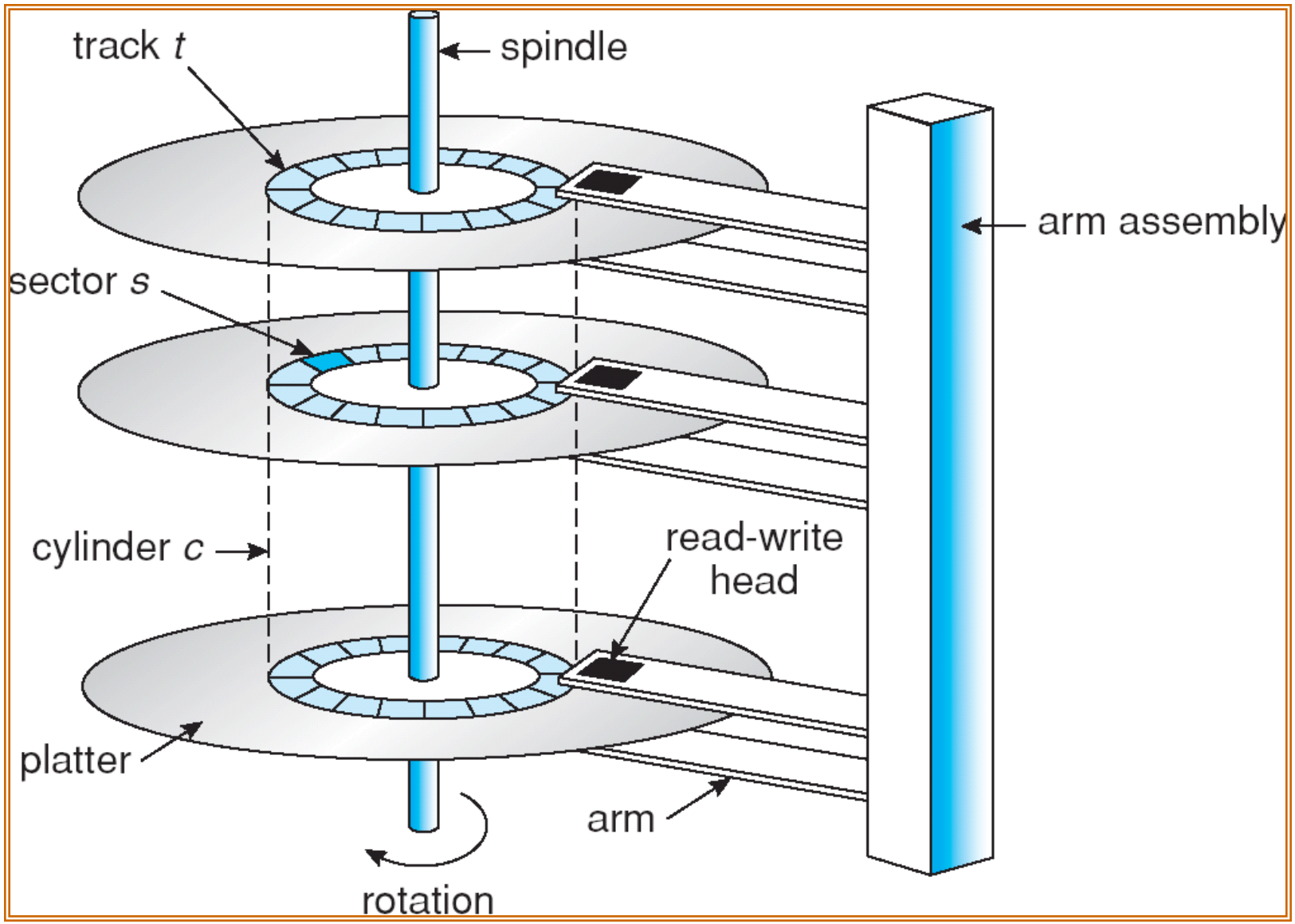
The WAFL File Layout



Overview of Mass Storage Structure

- Magnetic disks provide bulk of secondary storage of modern computers
  - Drives rotate at *60 to 200* times per second
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**).
  - **Head crash** results from disk head making contact with the disk surface
    - That's bad
- Disks can be removable
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE(Enhanced Integrated Drive Electronics) , ATA (Advanced Technology Attachment) , SATA (Serial ATA), USB (Universal Serial Bus), Fibre Channel, SCSI**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

Moving-head Disk Mechanism



- Magnetic tape:-
  - Was early secondary-storage medium
  - Relatively permanent and holds large quantities of data
  - Access time slow
  - Random access ~1000 times slower than disk
  - Mainly used for backup, storage of infrequently-used data, transfer medium between systems
  - Kept in spool and wound or rewound past read-write head
  - Once data under head, transfer rates comparable to disk
  - 20-200GB typical storage
  - Common technologies are 4mm, 8mm, 19mm, ¼ and ½ inch.

Disk Structure

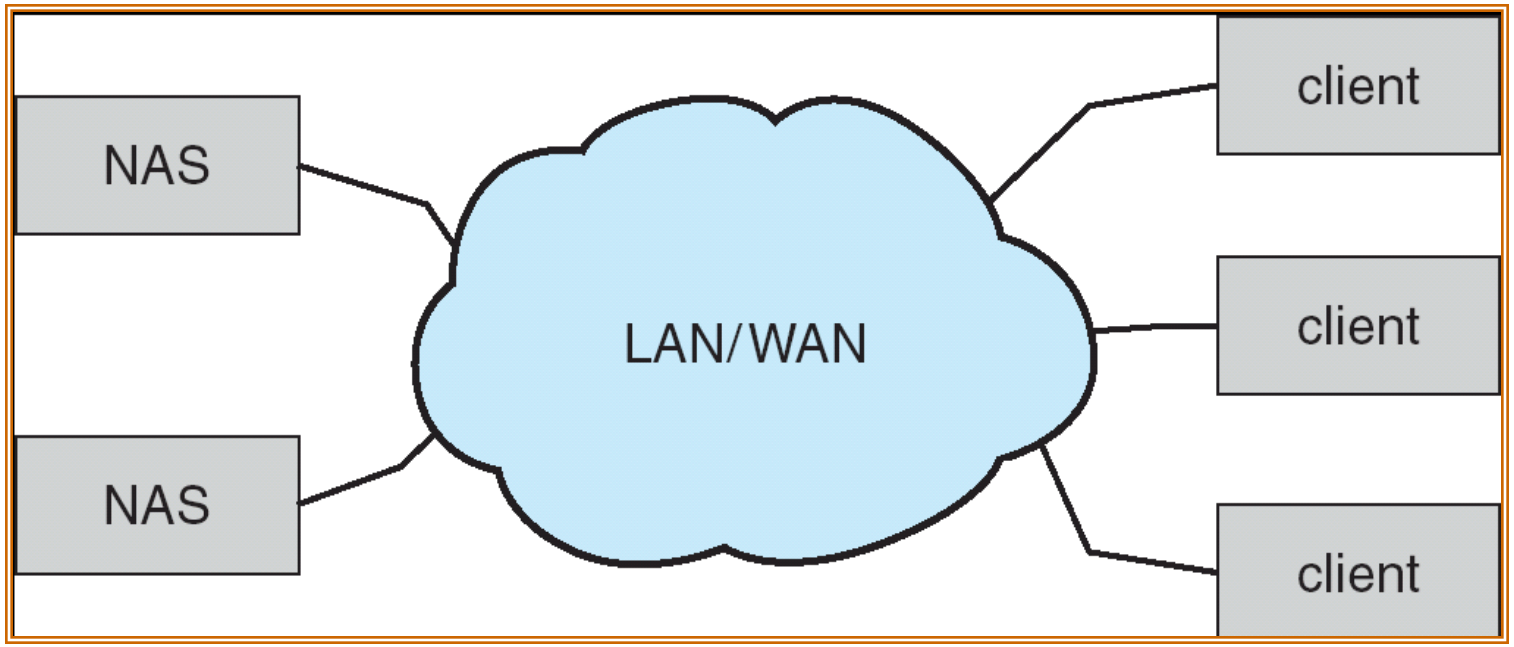
- Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

Disk Attachment

- Host-attached storage accessed through I/O ports talking to I/O busses.
- SCSI itself is a bus, up to 16 devices on one cable, **SCSI initiator** requests operation and **SCSI targets** perform tasks.
  - Each target can have up to 8 **logical units** (disks attached to device controller.
- FC (Fiber Channel) is high-speed serial architecture
  - Can be switched fabric with 24-bit address space – the basis of **storage area networks** (**SAN**s) in which many hosts attach to many storage units.
  - Can be **arbitrated loop** (**FC-AL**) of 126 devices.

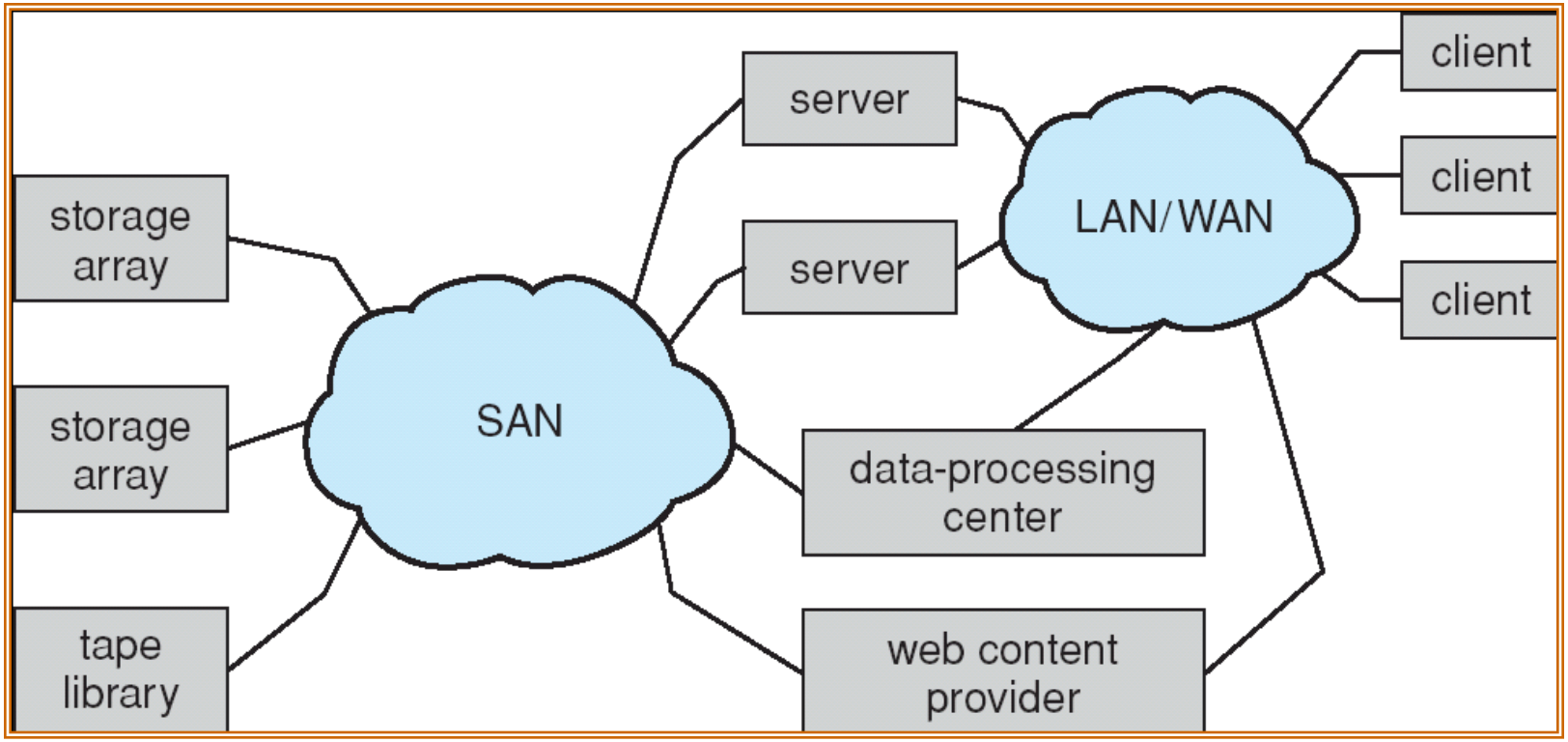
Network-Attached Storage

- Network-attached storage (**NAS**) is storage made available over a network rather than over a local connection (such as a bus).
- NFS is the common protocols.
- Implemented via remote procedure calls (RPCs) between host and storage.
- New iSCSI protocol uses IP network to carry the SCSI protocol.



Storage Area Network

- Common in large storage environments (and becoming more common)
- Multiple hosts attached to multiple storage arrays - flexible



Disk Scheduling

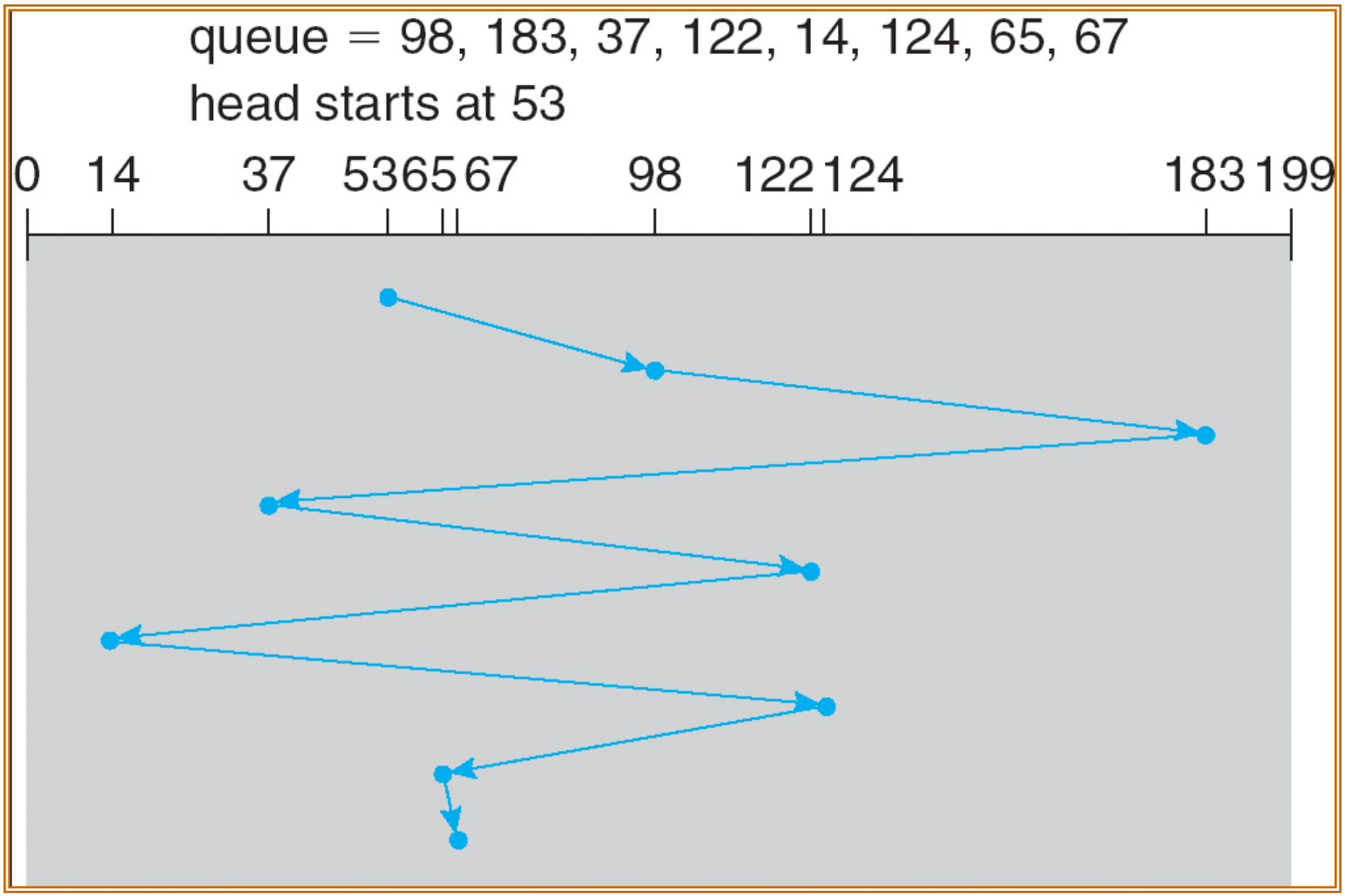
- The operating system is responsible for using hardware efficiently for the disk drives, this means having a fast access time and disk bandwidth.
- Access time has two major components
  - ***Seek time*** is the time for the disk are to move the heads to the cylinder containing the desired sector.
  - ***Rotational latency OR Rotational delay*** is the additional time waiting for the disk to rotate the desired sector to the disk head.
- Minimize seek time
- Seek time ≈ seek distance
- **Disk bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.
- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199).

98, 183, 37, 122, 14, 124, 65, 67

Head pointer 53

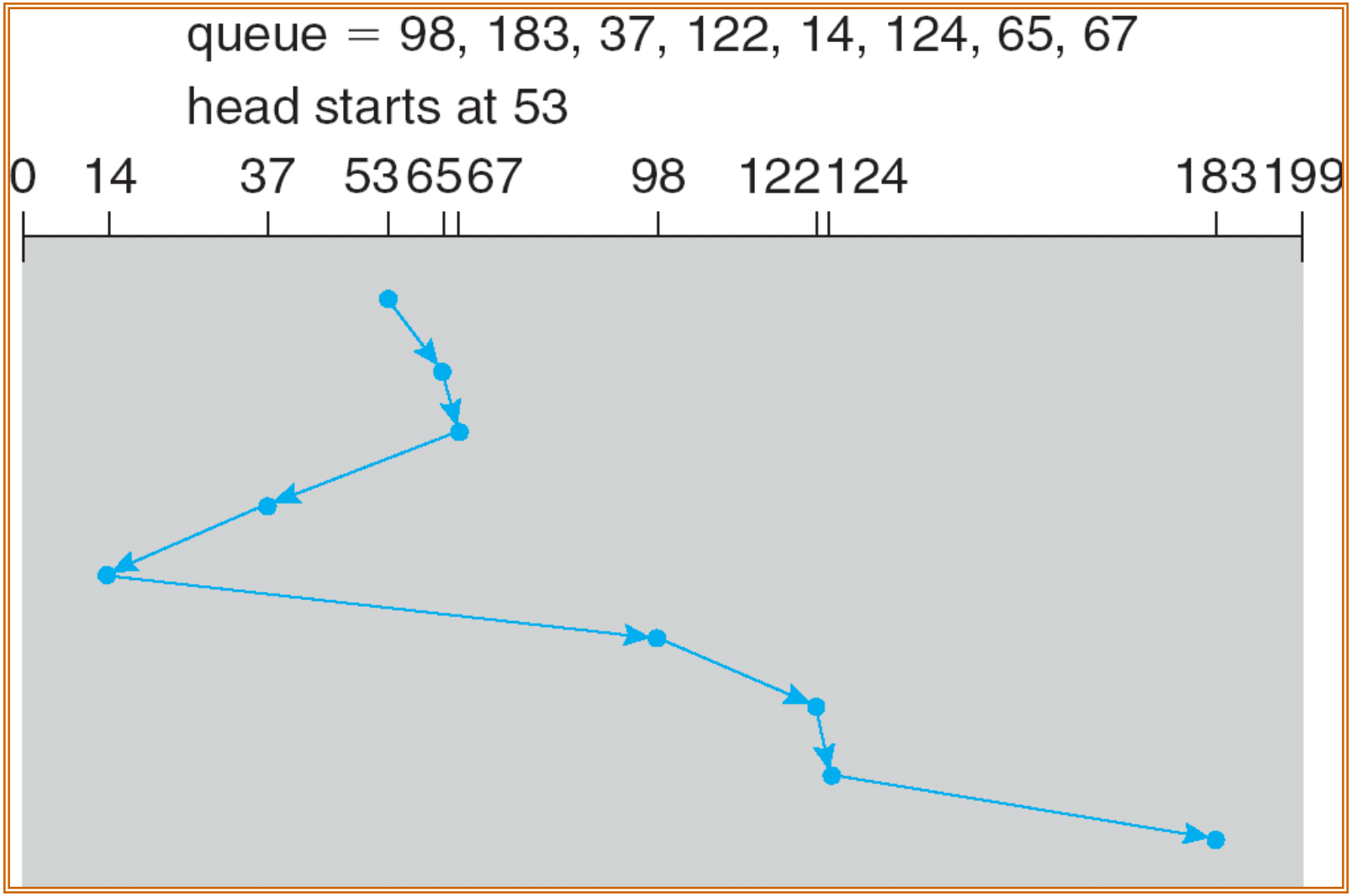
FCFS (First-Come-First-Serve)

- Illustration shows total head movement of **640** cylinders.



SSTF (Shortest-Seek-Time-First)

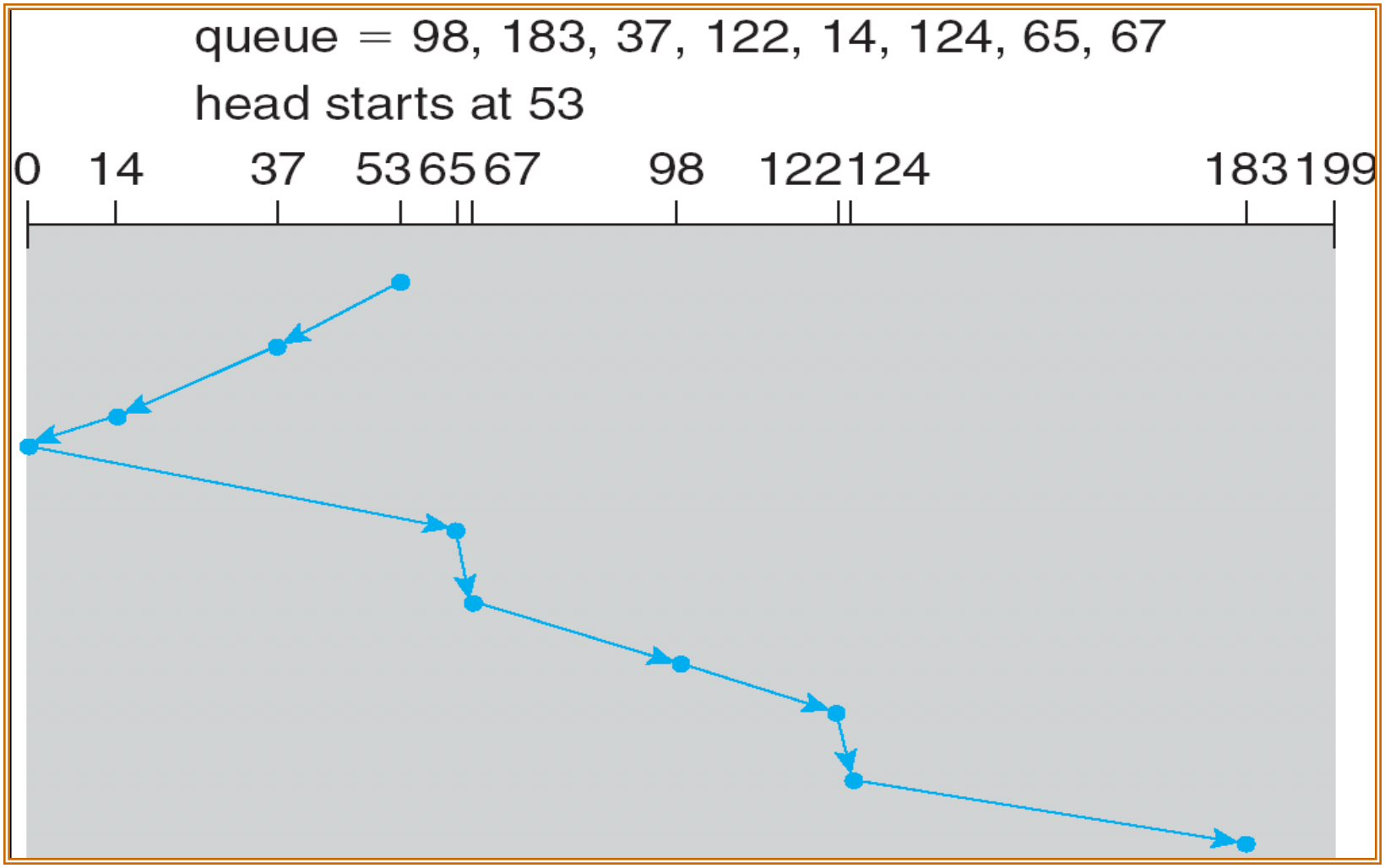
- SSTF: Shortest-Seek-Time-First
- Selects the request with the *minimum seek time from the current head position*.
- SSTF scheduling is a form of SJF scheduling; may cause *starvation* of some requests.
- Illustration shows total head movement of 236 cylinders.


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

SCAN (Elevator Algorithm)

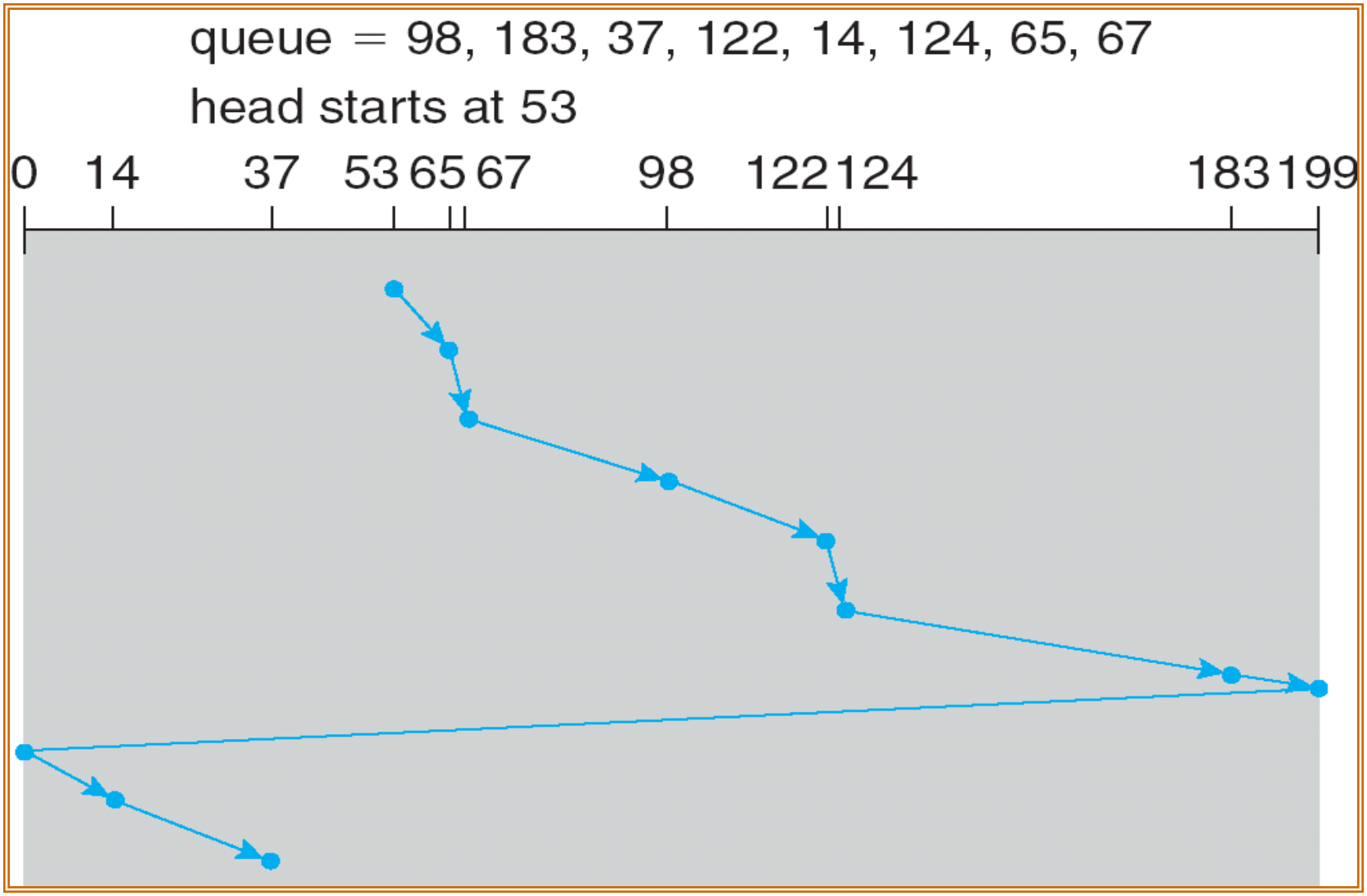
- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- Sometimes called the *elevator algorithm*.
- Illustration shows total head movement of 208 cylinders
- (head is moving towards cylinder 0).

Illustration shows total head movement of 208 cylinders.



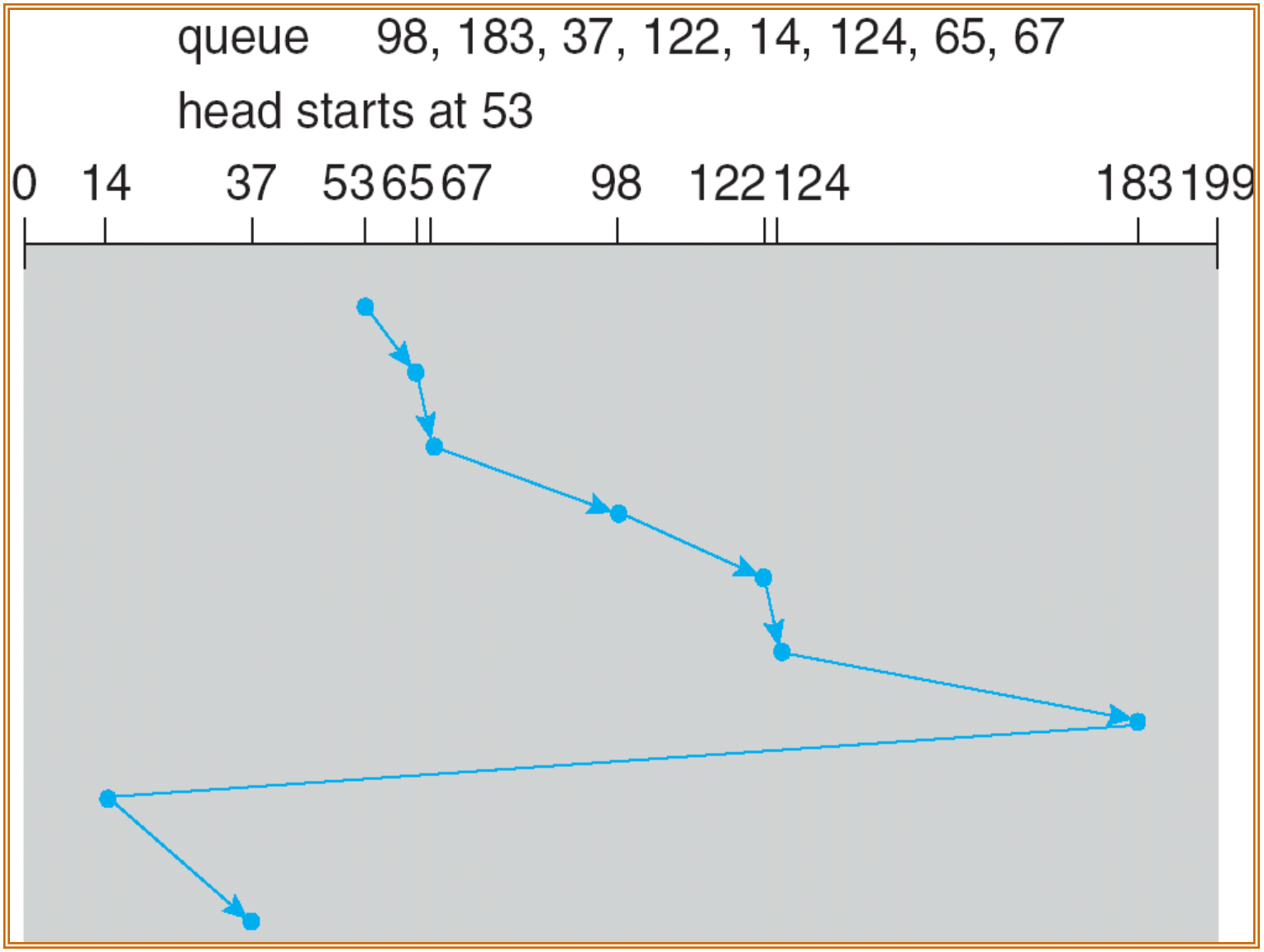
C-SCAN

- Provides a more *uniform wait time* than SCAN.
- The head moves from one end of the disk to the other.
- servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one.


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-LOOK

- Version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.


queue 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

Selecting a Disk-Scheduling Algorithm

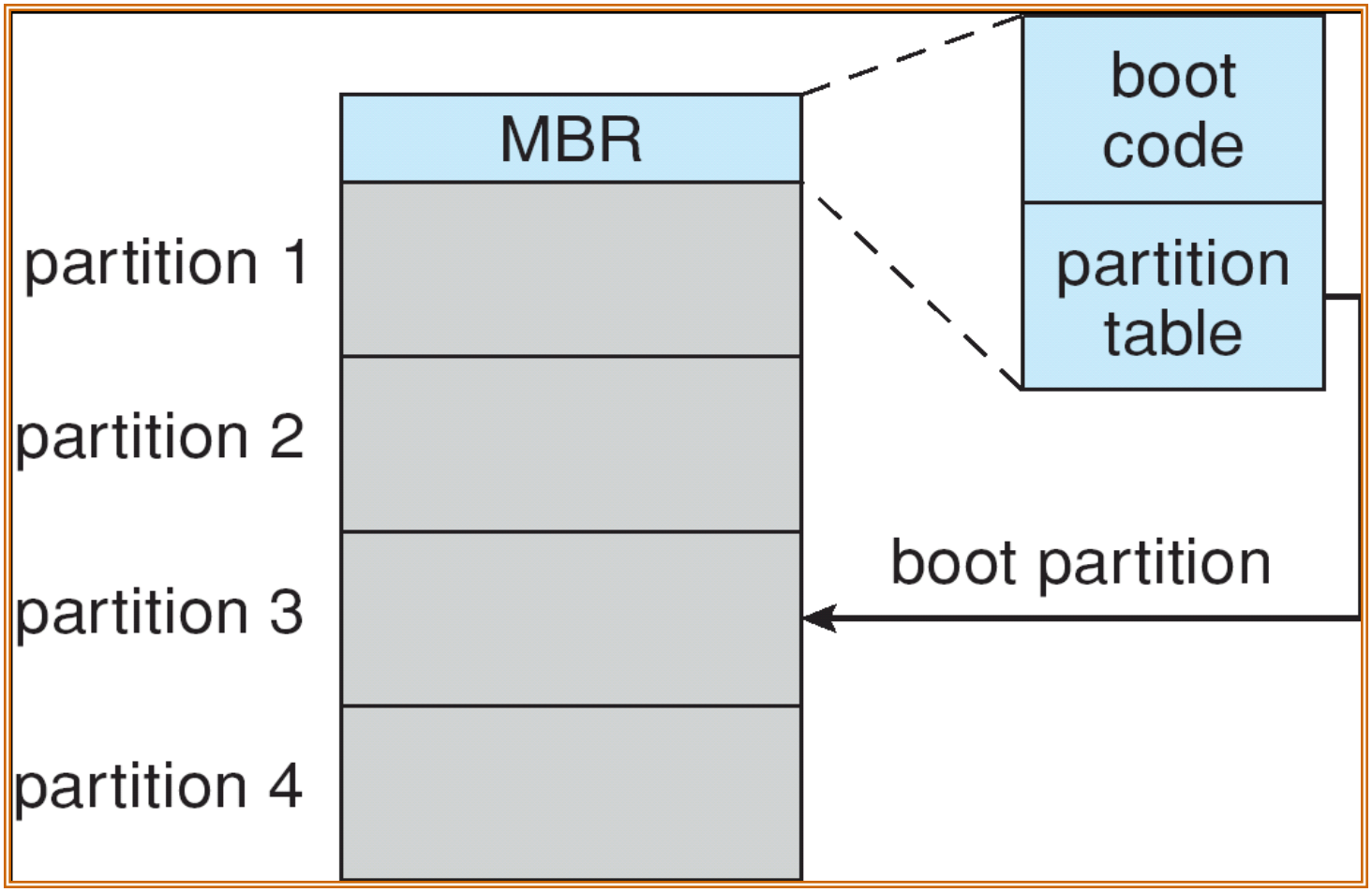
- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a *heavy load* on the disk.
- Performance depends on the number and types of requests.
- Requests for disk service can be influenced by the file-allocation method.
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.
- Either SSTF or LOOK is a reasonable choice for the default algorithm.

Disk Management

- 1. Disk Formatting:-
- Low-level formatting, OR physical formatting :- Before a disk can store data, it must be dividing a disk into sectors that the disk controller can read and write.
- To use a disk to hold files, the operating system still needs to record its own *data structures* on the disk.
- The data structure for a sector consists of a *header*, a *data area* (usually 512 bytes in size) , and *trailer*.
- The header and trailer contains information used by the disk controller, such as a sector number and an error-correcting code (ECC).
  - *Partition* the disk into one or more groups of cylinders.
  - *Logical formatting* or “making a file system”.

- 2. Boot Block:-
- For a computer to start running—for instance, when it is powered up or rebooted
- it must have an initial program to run.
- Boot block initializes system.
  - MBR (**Master Boot Record**):- Windows 2000 system place boot code in the first sector on the Hard disk.
  - The bootstrap is stored in ROM.
  - Bootstrap loader program.
- 3. Bad Blocks:-Because disks have moving parts and small tolerances they are prone to failure .
- Sometimes the failure is complete; in this case, the disk needs to be replaced and its contents restored from backup media to the new disk. More frequently, one or more sectors become defective.
- Methods such as *sector sparing* used to handle bad blocks.

Booting from a Disk in Windows 2000



Swap-Space Management

- Swap-space :- Virtual memory uses disk space as an extension of main memory.
- Swap-space can be carved out of the normal file system, it can be in a separate disk partition.
- External fragmentation can increases at swapping time.
- We can improving the performance by caching the block location information in physical memory.
- Swap-space management
  - BSD allocates swap space when process starts; holds *text segment* (the program) and *data segment.*
  - Kernel uses *swap maps* to track swap-space use.
  - Solaris 2 allocates swap space only when a page is forced out of physical memory, not when the virtual memory page is first created.

RAID Structure

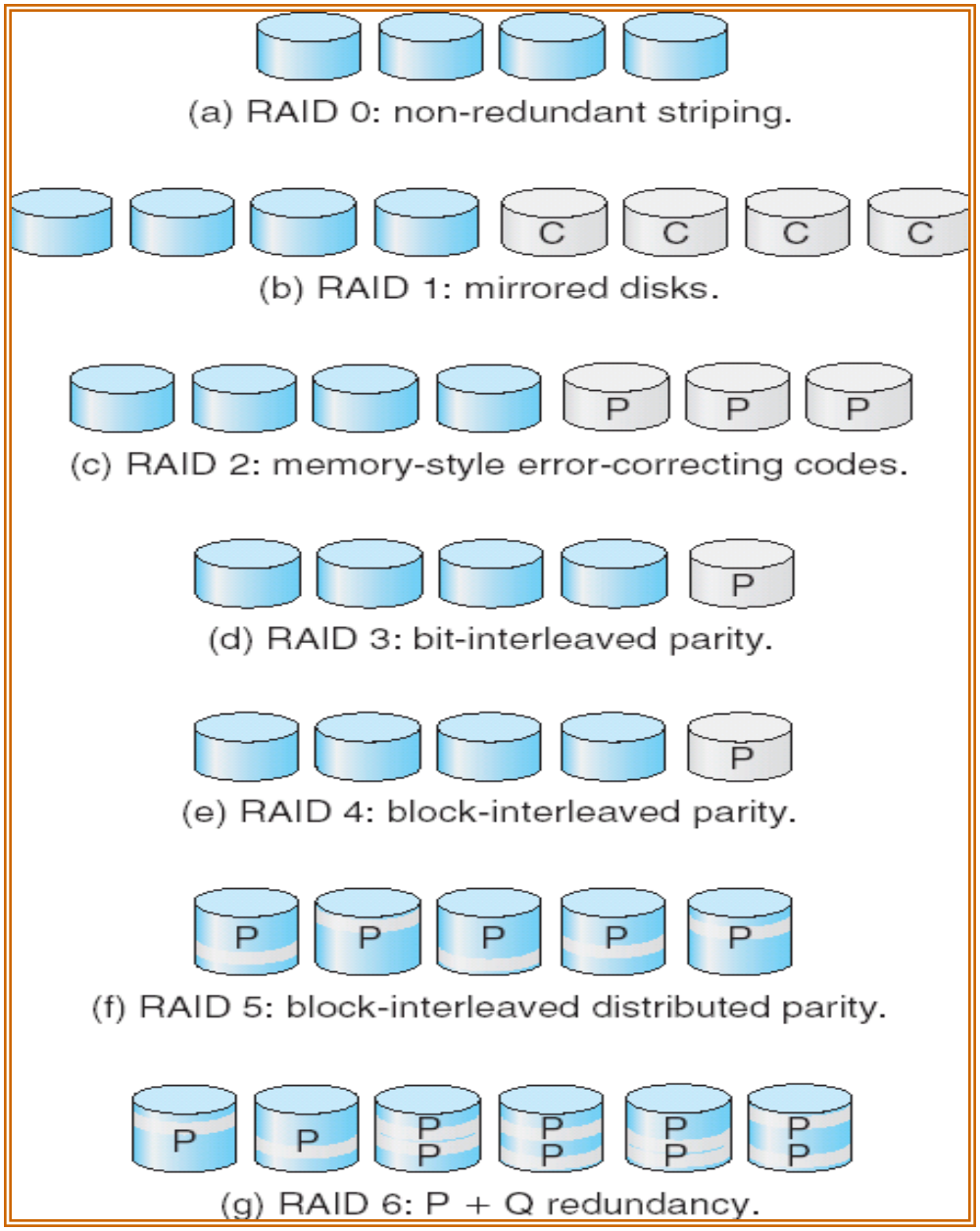
- **RAID** – (Redundant Arrays of Independent Disks)
  multiple disk drives provides **reliability** via **redundancy**.

- RAID is arranged into six different levels.
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.

- Disk striping uses a group of disks as one storage unit.

- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.
  - *Mirroring* or *shadowing* keeps duplicate of each disk.
  - *Block interleaved parity* uses much less redundancy.

RAID Levels

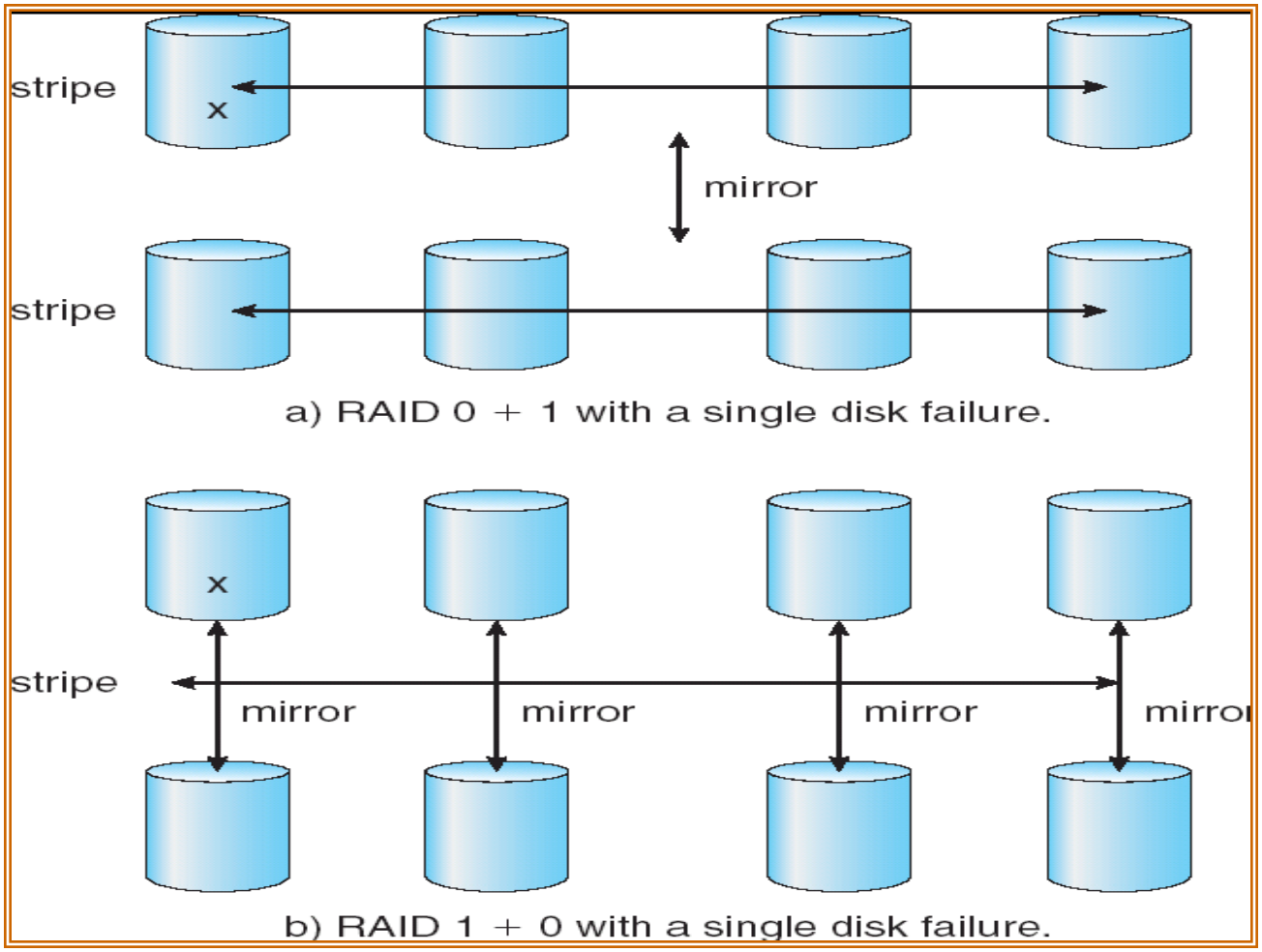

C
P
(a) RAID 0: non-redundant striping.
(b) RAID 1: mirrored disks.
(c) RAID 2: memory-style error-correcting codes.
(d) RAID 3: bit-interleaved parity.
(e) RAID 4: block-interleaved parity.
(f) RAID 5: block-interleaved distributed parity.
(g) RAID 6: P + Q redundancy.

- RAID 0:- RAID 0 is not a family member of RAID
  - Time consumption is reduced.
  - It is not maintain the duplicate of data (If loss of data).
  - If essential (Important data) is not maintain the parity information.
- RAID 1:- It is a MIRRORED
  - It acts as a Mirror.
  - It stores all information of duplicate.
  - If any data fail, it will show at mirror.
  - DB:-It is costly, because of maintain double number.
- RAID 2:-Here only 3 disks are used for duplication
  - Mirror is takes only 3 disks.
  - By using Hamming code Technique if any error occur then rectify that one only. ECC: Error Correcting Code.

- RAID 3:- Bit Interleaved parity
  - In each disk is stored data in the form of bit information like binary (0→ 0000)
  - Parity bit is used to store the parity of the Information.
  - If any information is collapse then it will check the parity bit of information
  - It is maintain by Even Parity or Odd parity
- RAID 4:- Block Interleaved parity
  - Here information is as blocks.
  - DB:- So each block is taken. Then if write a bit then first read total block
  - So time consumption is occur.
- RAID 5:-Block level Distributed Parity
  - Here each and every block level is maintain a parity
  - So any data is fail then that parity is check in that particular block and then retrieve data.
  - The blocks parity is stored in some manner. 0 1 2 3 then the parity is stored in the parity disk P(0-3) , 4 5 6 P(4-7) 7 , 8 9 P(8-11) 10 11 and 12 P(12-15) 13 14 15.

- RAID 6:-Redundent –Parity Level
  - Duplication of Parity .
  - It takes 2 parity disks.
  - Another parity disk is stores the duplication of Previous parity.
  - P is Parity disk then Q is a Duplicate of Parity.
  - 0 1 2 3 then the parity is stored in the parity disk P(0-3) Q(0-3) , 4 5 6 P(4-7) Q(4-7) 7 , 8 9 P(8-11) Q(8-11) 10 11 and 12 P(12-15) Q(12-15) 13 14 15.

RAID (0 + 1) and (1 + 0)



Stable-Storage Implementation

- Write-ahead log scheme requires stable storage.
- To implement stable storage:-
  - Replicate information on more than one nonvolatile storage media with independent failure modes.
  - Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery.

Tertiary Storage Devices

- Low cost is the defining characteristic of tertiary storage.
- Generally, tertiary storage is built using *removable media*
- Common examples of removable media are floppy disks and CD-ROMs; other types are available.

Removable Disks

- Floppy disk — thin flexible disk coated with magnetic material, enclosed in a protective plastic case.
  - Most floppies hold about 1 MB; similar technology is used for removable disks that hold more than 1 GB.
  - Removable magnetic disks can be nearly as fast as hard disks, but they are at a greater risk of damage from exposure.
- A magneto-optic disk records data on a rigid platter coated with magnetic material.
  - Laser heat is used to amplify a large, weak magnetic field to record a bit.
  - Laser light is also used to read data (Kerr effect).
  - The magneto-optic head flies much farther from the disk surface than a magnetic disk head, and the magnetic material is covered with a protective layer of plastic or glass; resistant to head crashes.
- Optical disks do not use magnetism; they employ special materials that are altered by laser light.

WORM Disks

- The data on read-write disks can be modified over and over.
- WORM (“Write Once, Read Many Times”) disks can be written only once.
- Thin aluminum film sandwiched between two glass or plastic platters.
- To write a bit, the drive uses a laser light to burn a small hole through the aluminum; information can be destroyed by not altered.
- Very durable and reliable.
- *Read Only* disks, such ad CD-ROM and DVD, come from the factory with the data pre-recorded.

Tapes

- Compared to a disk, a tape is less expensive and holds more data, but random access is much slower.
- Tape is an economical medium for purposes that do not require fast random access, e.g., backup copies of disk data, holding huge volumes of data.
- Large tape installations typically use robotic tape changers that move tapes between tape drives and storage slots in a tape library.
  - stacker – library that holds a few tapes
  - silo – library that holds thousands of tapes
- A disk-resident file can be *archived* to tape for low cost storage; the computer can *stage* it back into disk storage for active use.

Operating System Issues

- Major OS jobs are to manage physical devices and to present a virtual machine abstraction to applications

- For hard disks, the OS provides two abstraction:
  - Raw device – an array of data blocks.
  - File system – the OS queues and schedules the interleaved requests from several applications.

Application Interface

- Most OSs handle removable disks almost exactly like fixed disks — a new cartridge is formatted and an empty file system is generated on the disk.
- Tapes are presented as a raw storage medium, i.e., and application does not open a file on the tape, it opens the whole tape drive as a raw device.
- Usually the tape drive is reserved for the exclusive use of that application.
- Since the OS does not provide file system services, the application must decide how to use the array of blocks.
- Since every application makes up its own rules for how to organize a tape, a tape full of data can generally only be used by the program that created it.

Tape Drives

- The basic operations for a tape drive differ from those of a disk drive.
- **locate** positions the tape to a specific logical block, not an entire track (corresponds to **seek**).
- The **read position** operation returns the logical block number where the tape head is.
- The **space** operation enables relative motion.
- Tape drives are “append-only” devices; updating a block in the middle of the tape also effectively erases everything beyond that block.
- An EOT mark is placed after a block that is written.

File Naming

- The issue of naming files on removable media is especially difficult when we want to write data on a removable cartridge on one computer, and then use the cartridge in another computer.
- Contemporary OSs generally leave the name space problem unsolved for removable media, and depend on applications and users to figure out how to access and interpret the data.
- Some kinds of removable media (e.g., CDs) are so well standardized that all computers use them the same way.

Hierarchical Storage Management (HSM)

- A hierarchical storage system extends the storage hierarchy beyond primary memory and secondary storage to incorporate tertiary storage — usually implemented as a jukebox of tapes or removable disks.
- Usually incorporate tertiary storage by extending the file system.
  - Small and frequently used files remain on disk.
  - Large, old, inactive files are archived to the jukebox.
- HSM is usually found in supercomputing centers and other large installations that have enormous volumes of data.

Speed

- Two aspects of speed in tertiary storage are bandwidth and latency.

- Bandwidth is measured in bytes per second.
  - Sustained bandwidth – average data rate during a large transfer; # of bytes/transfer time.
    Data rate when the data stream is actually flowing.
  - Effective bandwidth – average over the entire I/O time, including **seek** or **locate**, and cartridge switching.
    Drive's overall data rate.
- Access latency – amount of time needed to locate data.
  - Access time for a disk – move the arm to the selected cylinder and wait for the rotational latency; < 35 milliseconds.
  - Access on tape requires winding the tape reels until the selected block reaches the tape head; tens or hundreds of seconds.
  - Generally say that random access within a tape cartridge is about a thousand times slower than random access on disk.
- The low cost of tertiary storage is a result of having many cheap cartridges share a few expensive drives.
- A removable library is best devoted to the storage of infrequently used data, because the library can only satisfy a relatively small number of I/O requests per hour.

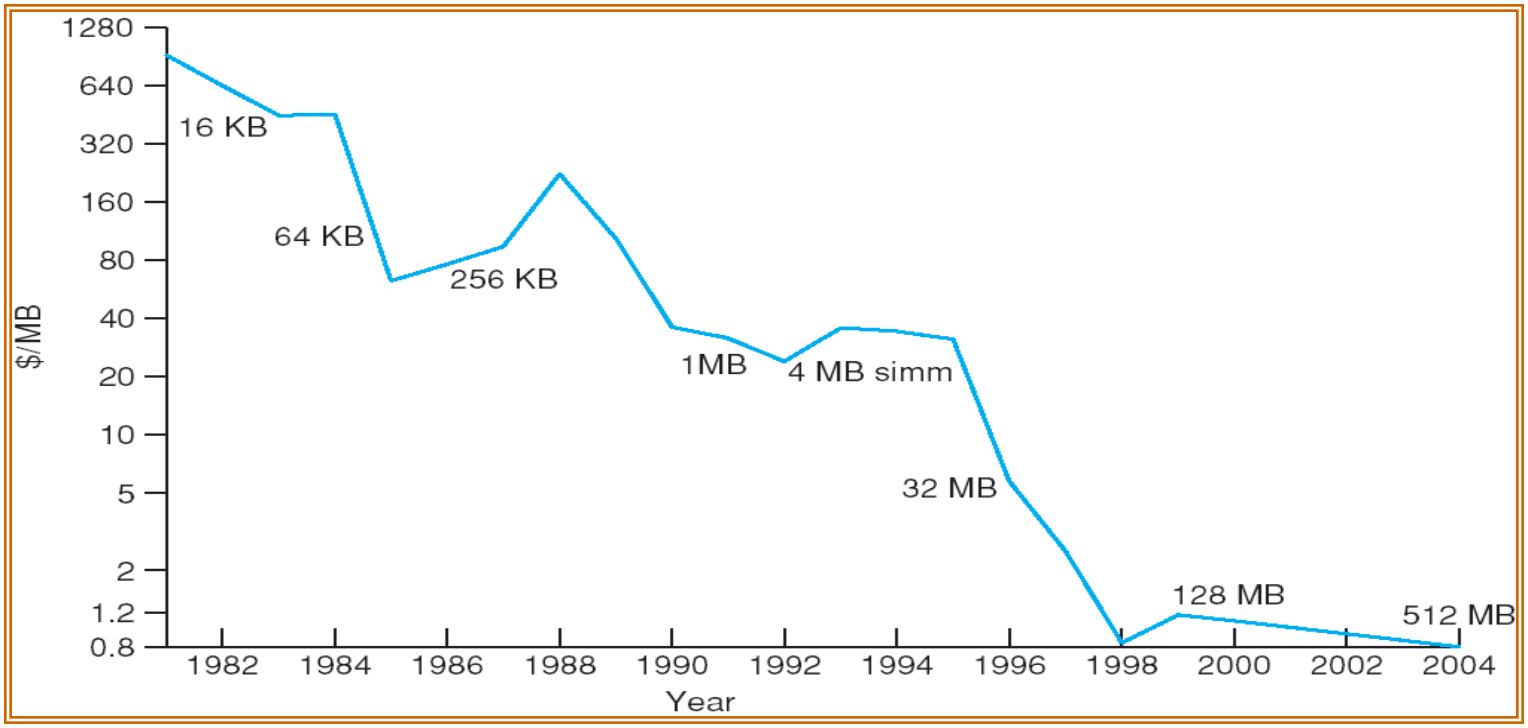
Reliability

- A fixed disk drive is likely to be more reliable than a removable disk or tape drive.
- An optical cartridge is likely to be more reliable than a magnetic disk or tape.
- A head crash in a fixed hard disk generally destroys the data, whereas the failure of a tape drive or optical disk drive often leaves the data cartridge unharmed.

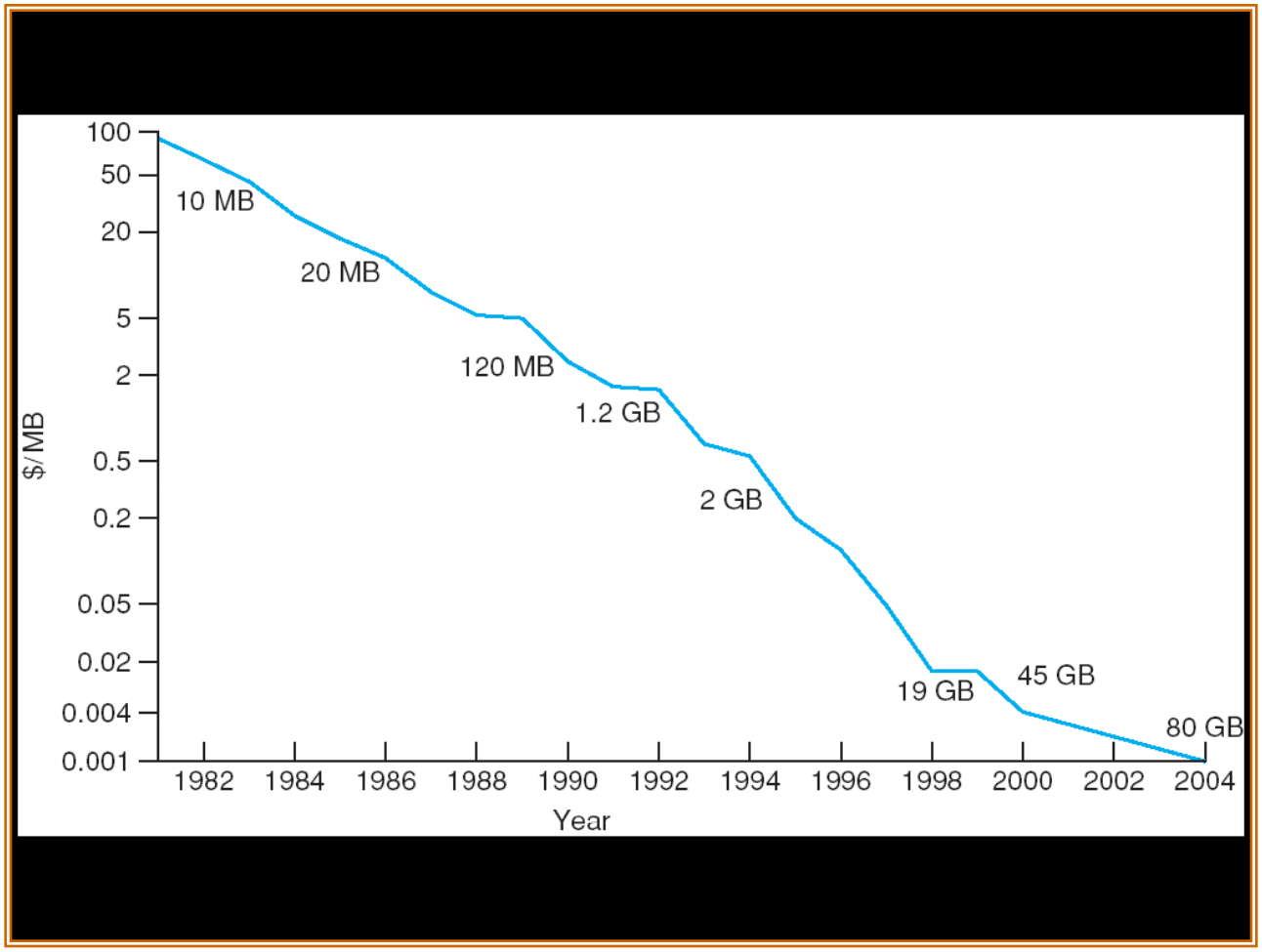
Cost

- Main memory is much more expensive than disk storage
- The cost per megabyte of hard disk storage is competitive with magnetic tape if only one tape is used per drive.
- The cheapest tape drives and the cheapest disk drives have had about the same storage capacity over the years.
- Tertiary storage gives a cost savings only when the number of cartridges is considerably larger than the number of drives.

Price per Megabyte of DRAM, From 1981 to 2004



Price per Megabyte of Magnetic Hard Disk, From 1981 to 2004



Price per Megabyte of a Tape Drive, From 1984-2000