Chapter 7
Device Management

Understanding Operating Systems, Fourth Edition

Objectives

You will be able to describe:

- The features of dedicated, shared, and virtual devices
- The differences between sequential and direct access media
- The concepts of blocking and buffering and how they improve I/O performance
- The roles of seek time, search time, and transfer time in calculating access time
- The differences in access times in several types of devices

Objectives (continued)

You will be able to describe:

- The critical components of the input/output subsystem, and how they interact
- The strengths and weaknesses of common seek strategies, including FCFS, SSTF, SCAN/LOOK, C-SCAN/C-LOOK, and how they compare
- The different levels of RAID and what sets each one apart from the others

Device Management

Device Management Functions:

- Tracking the status of each device
  - e.g., disk drives, printers, modems, etc.
- Using preset policies to determine which process will get a device and for how long
- Allocating the devices
- Deallocating devices at two levels
  - At the process level
  - At the job level

Types of Devices

- Peripheral devices are categorized as follows:
  - Characteristics of the devices
  - How they’re managed by the Device Manager
- Different categories:
  - Dedicated, shared, and virtual
- Most important differences among devices
  - Speed
  - Degree of sharability

Dedicated Devices

- Assigned to only one job at a time and serves that job for entire time it’s active
  - e.g., tape drives, printers, and plotters
- **Disadvantage:**
  - Must be allocated to a single user for duration of a job’s execution
  - Can be quite inefficient, especially when device isn’t used 100% of the time

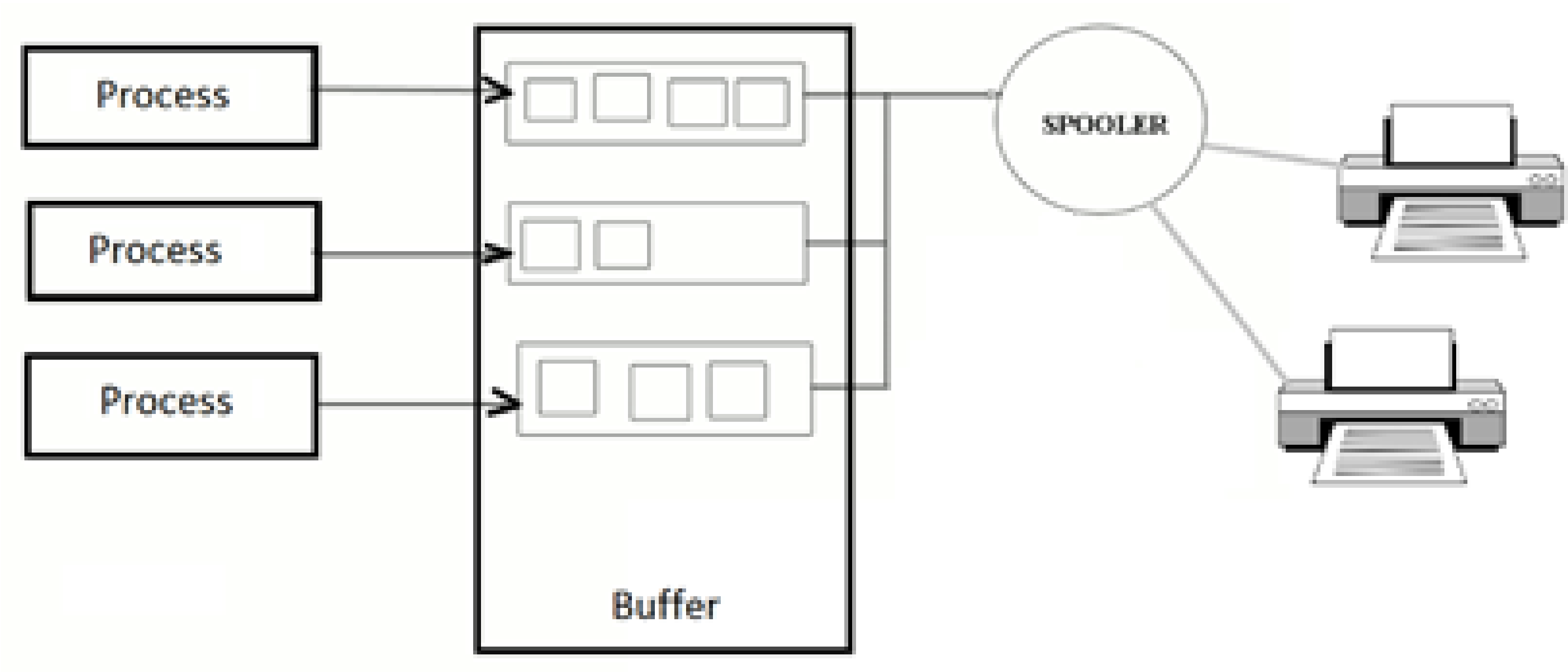
Shared Devices

- Assigned to several processes
  - e.g., disk pack or other DASDs can be shared by several processes at same time by interleaving their requests
  - Interleaving must be carefully controlled by Device Manager
- All conflicts must be resolved based on pre-determined policies

Virtual Devices

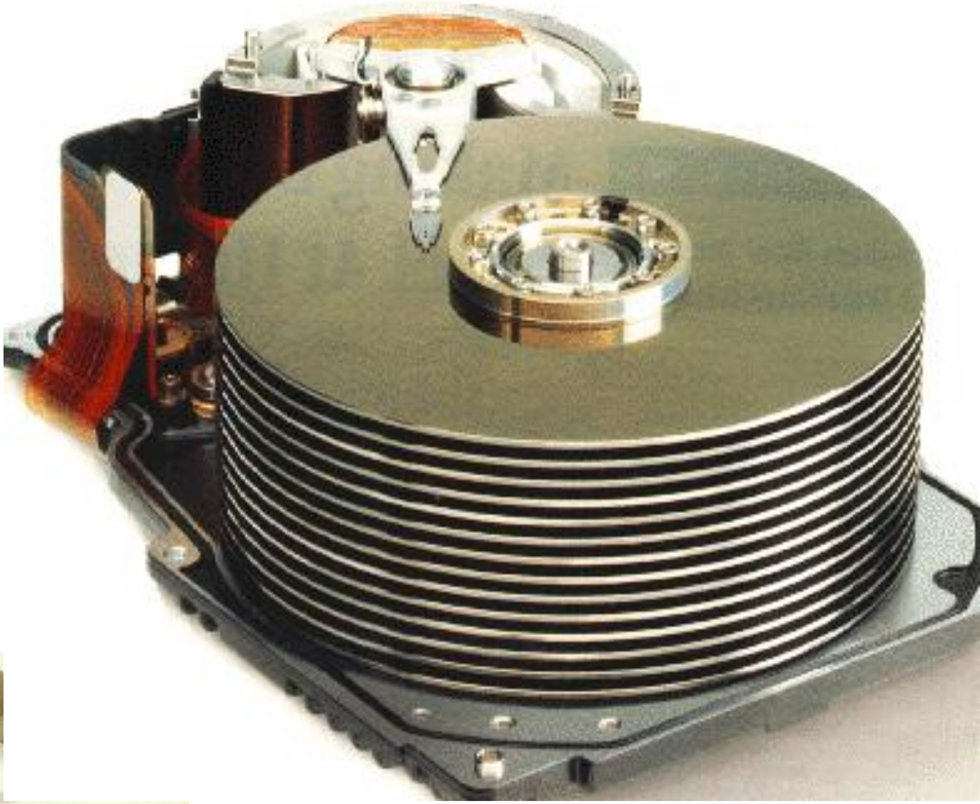
- Dedicated devices that have been transformed into shared devices
  - e.g., printers (dedicated devices) converted into sharable devices through a spooling program
- Spooling is used to speed up slow dedicated I/O devices
  - e.g., USB controller, a virtual device that acts as an interface between OS, device drivers, and applications and the devices that are attached via the USB host

Spooling



Sequential Access Storage Media

- **Storage media** are divided into two groups:
  - **Sequential access media**
    - Store records sequentially
  - **Direct access storage devices (DASD)**
    - Store either sequential or direct access files
- There are vast differences in their speed and sharability




Magnetic Tape
IBM
Magnetic Tape
IBM
Magnetic Tape
IBM

Sequential Access Storage Media (continued)

- **Paper:** First storage medium: printouts, punch cards
- **Magnetic tape:** Used for secondary storage on early computer systems; now used for routine archiving & storing back-up data
  - Records on magnetic tapes are stored serially
  - Record length determined by the application program
  - Each record identified by its position on the tape
  - Tape is mounted and fast-forwarded to access a single record
    - Time-consuming process

Sequential Access Storage Media (continued)

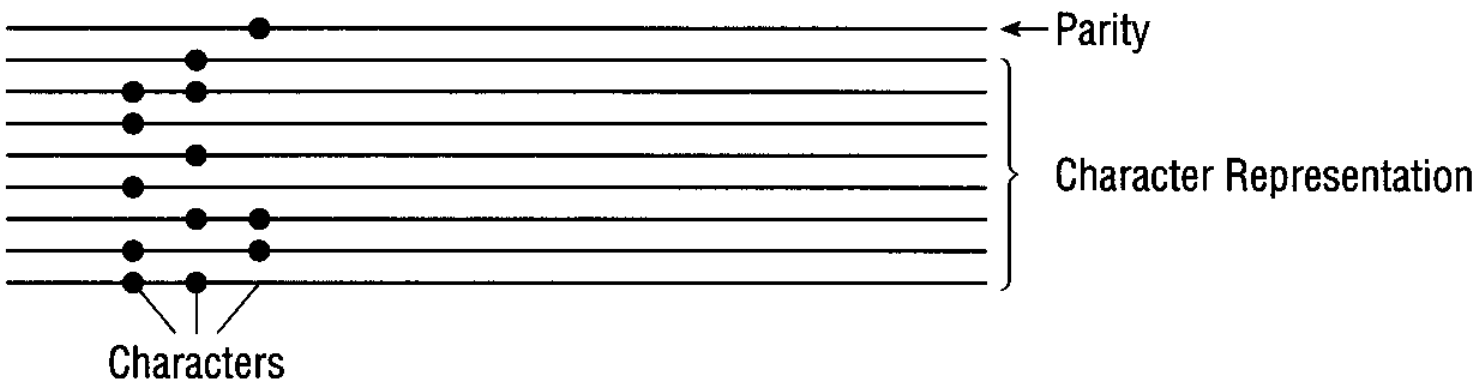


Figure 7.1: Nine-track magnetic tape

Sequential Access Storage Media (continued)

- **Magnetic tape** (continued):
  - Data is recorded on 8 parallel tracks that run the length of tape
  - Ninth track holds parity bit for routine error checking
  - Density of tape determines number of characters that can be recorded per inch
  - Records can be stored individually or in blocks
    - Blocking provides efficient way of storing records

Sequential Access Storage Media (continued)

- **Magnetic tape** (continued):
  - **Interrecord gap (IRG)**: Gap between records about 1/2 inch long regardless of the sizes of the records it separates
  - **Interblock gap (IBG)**: Gap between blocks of records; still 1/2 inch long
  - **Transfer rate** = Tape density x tape transport speed

Sequential Access Storage Media (continued)

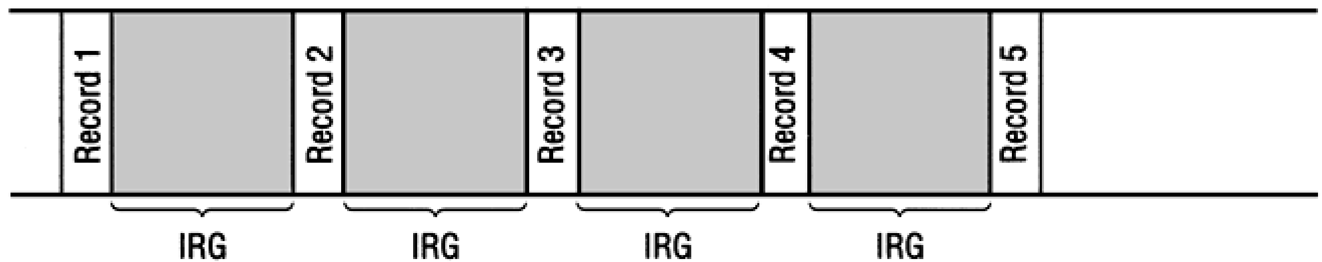


Figure 7.2: Records stored individually

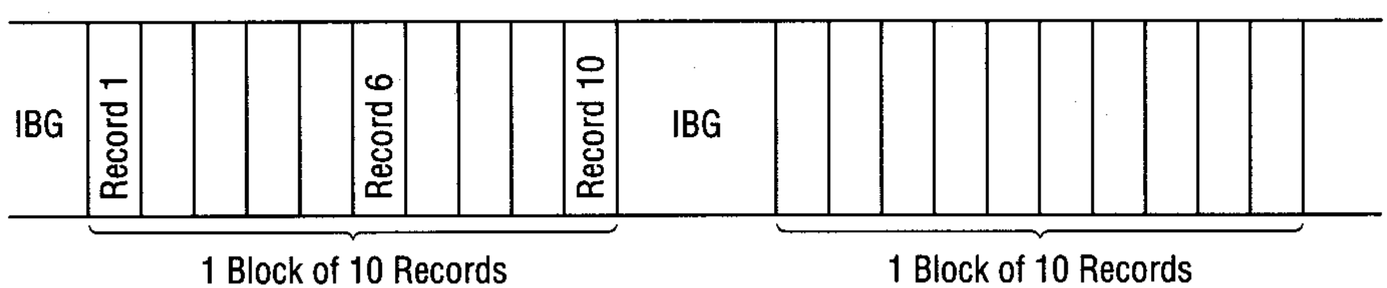


Figure 7.3: Records stored in blocks

Sequential Access Storage Media (continued)

- **Advantages of blocking:**
  - Fewer I/O operations needed
  - Less tape is wasted
- **Disadvantages of blocking:**
  - Overhead and software routines are needed for blocking, deblocking, and record keeping
  - Buffer space wasted if only one logical record is needed

Sequential Access Storage Media (continued)

- **Advantages of magnetic tapes:**
  - Low cost
  - Compact storage capabilities
    - Good medium for backing up magnetic disks and for long-term archival file storage
- **Disadvantages of magnetic tapes:**
  - Access time variability
    - Poor medium for routine secondary storage
  - Not good for interactive applications

Direct Access Storage Devices

- **DASDs:** Any devices that can directly read or write to a specific place on a disk
- **Categories:**
  - Magnetic disks
    - Fixed-Head Magnetic Disk Storage
    - Movable-Head Magnetic Disk Storage
  - Optical discs
  - Flash memory
  - Magneto-optical disks
- Location of a record directly affects access time

Fixed-Head Magnetic Disk Storage

- Looks like a large CD or DVD covered with magnetic film
- Formatted, usually on both sides, into concentric circles called tracks
- Data is recorded serially on each track by the fixed read/write head positioned over it
- **Applications:** Spacecraft monitoring or aircraft applications (where speed is of utmost importance)
- **Disadvantages:** High cost and reduced storage

Fixed-Head Magnetic Disk Storage (continued)

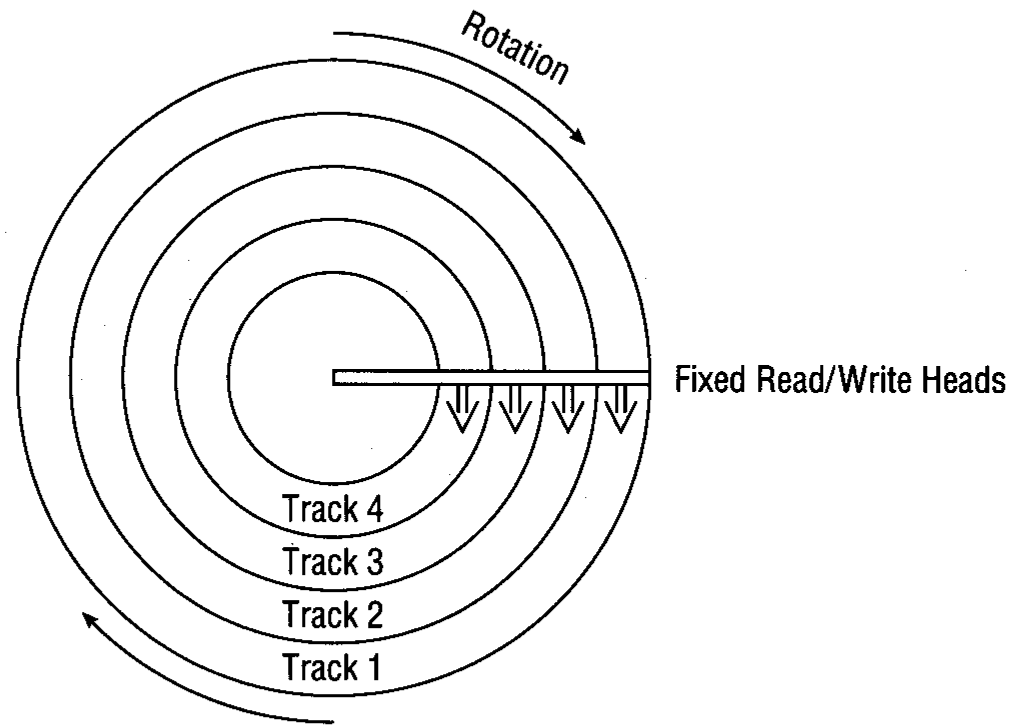
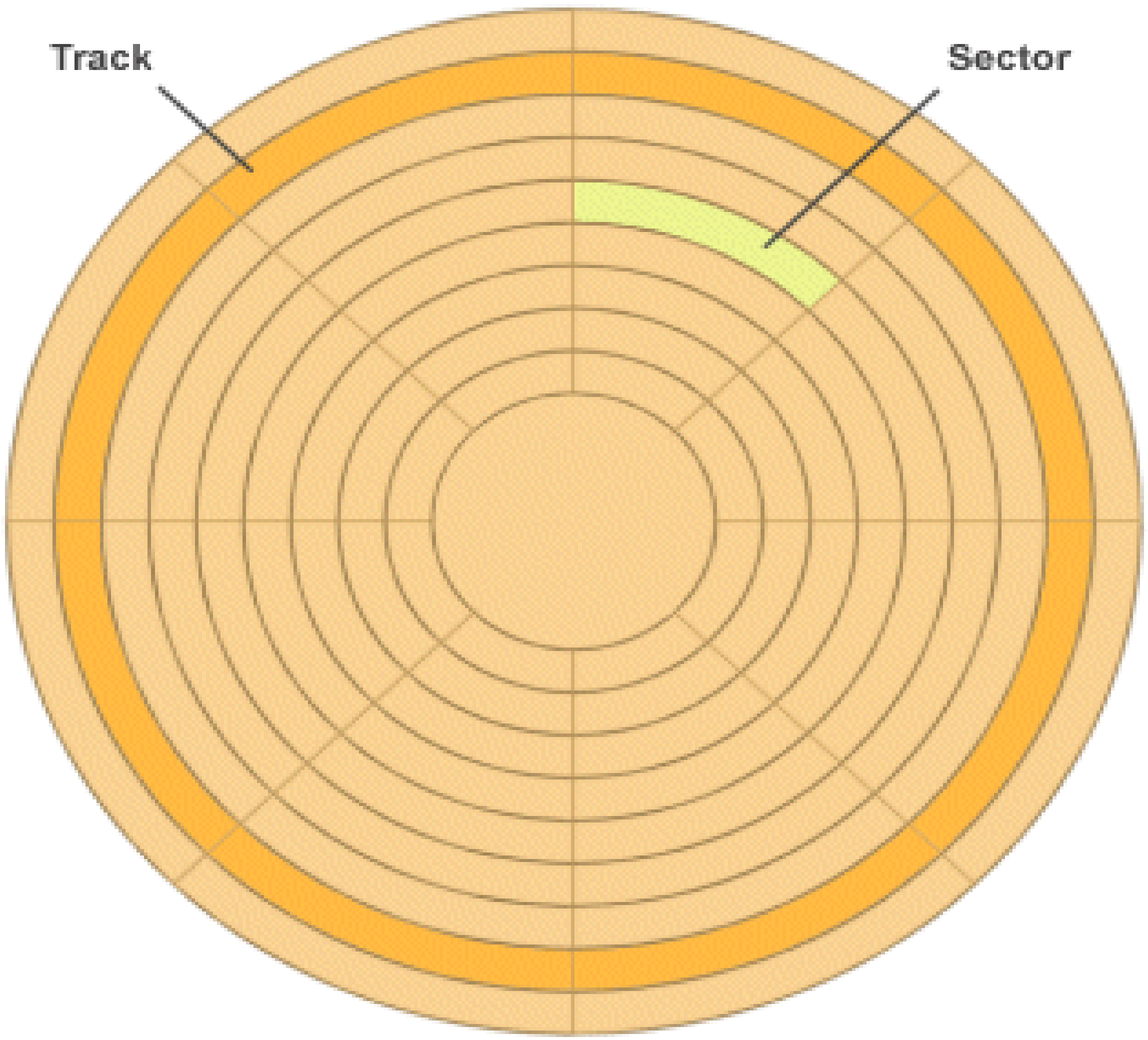


Figure 7.4: A fixed-head disk with four read/write heads, one per track

Movable-Head Magnetic Disk Storage

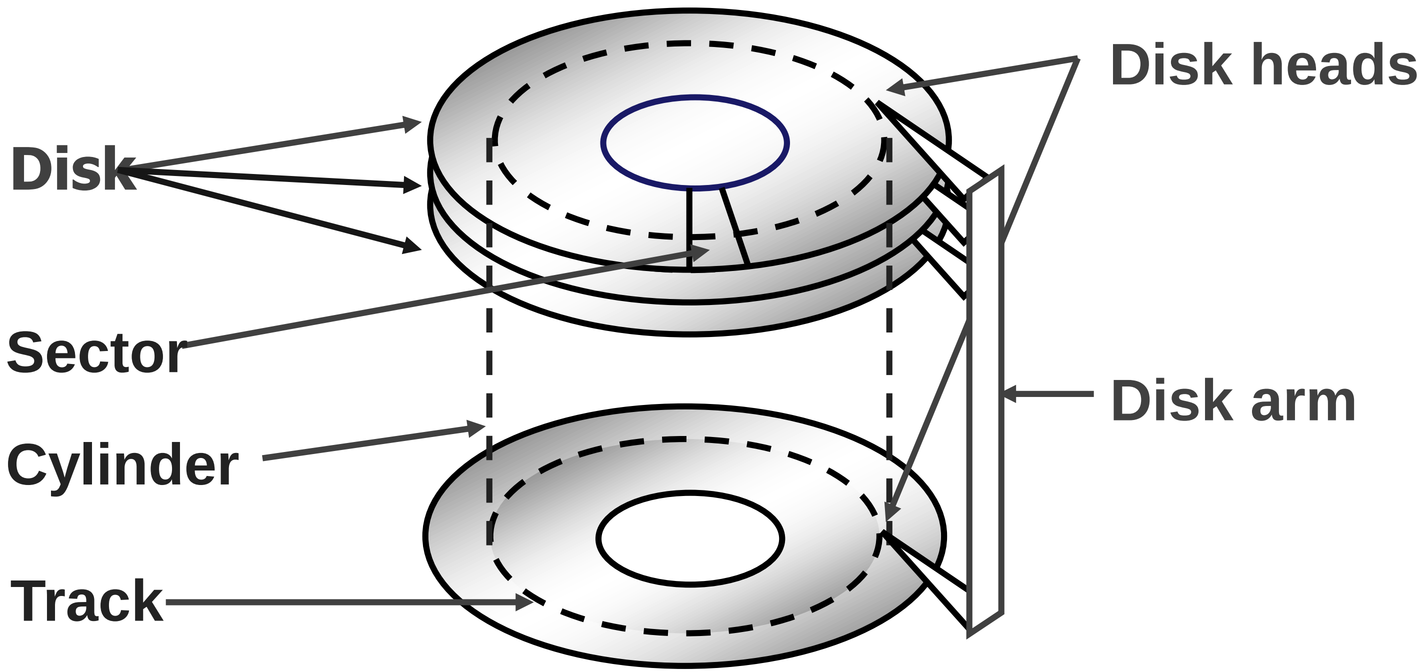
- Have one read/write head that floats over the surface of each disk, e.g., PC hard drives
  - Can be a single platter
  - Can be a part of a disk pack (stack of platters)
- **Disk Pack:**
  - Each platter has two surfaces for recording (except those at the top and bottom of the stack)
  - Each surface is formatted with concentric tracks
  - Number of tracks ranges from 100 on a floppy disk to a thousand or more on a high-capacity hard disk


Track
Sector

Movable-Head Magnetic Disk Storage (continued)

Figure 7.6: A typical hard drive from a PC

A *cylinder* consists of all tracks with a given disk arm position



Movable-Head Magnetic Disk Storage (continued)

- **Disk Pack** (continued)**:**
  - Track 0 identifies the outermost concentric circle on each surface; the highest-numbered track is in the center
  - The arm moves all of the heads in unison
  - Faster to fill a disk pack track-by-track
- To access any given record, the system needs:
  - Cylinder number
  - Surface number
  - Record number

Movable-Head Magnetic Disk Storage (continued)

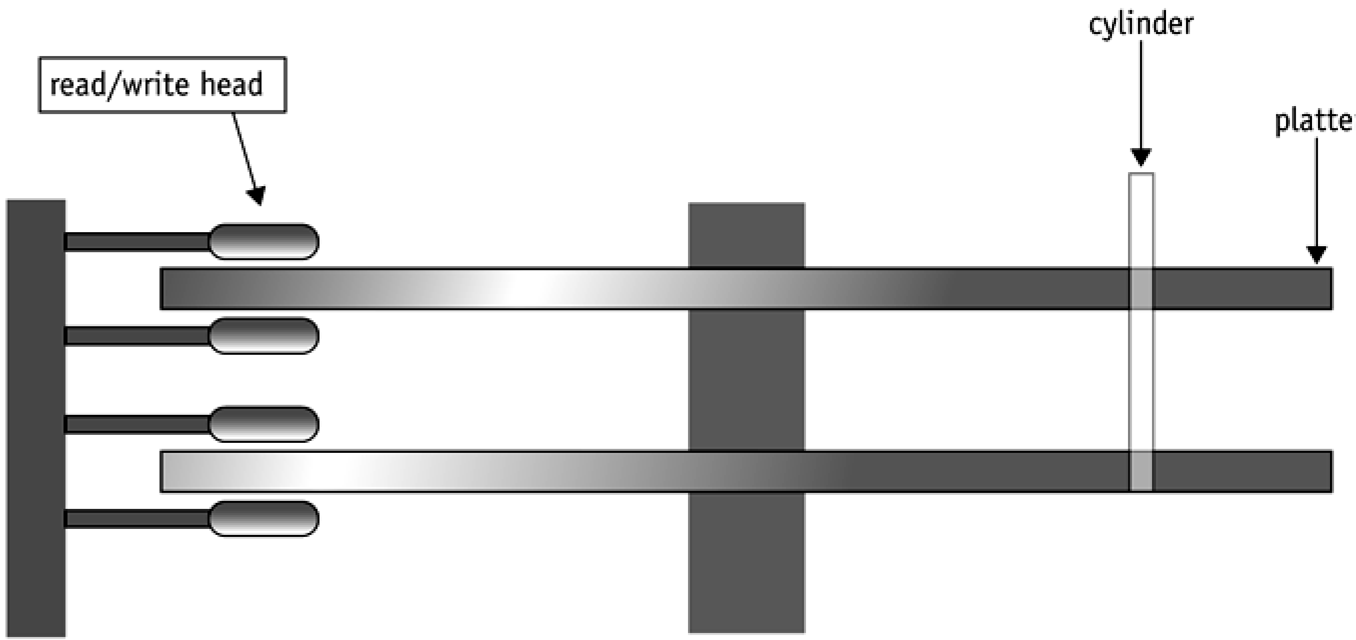
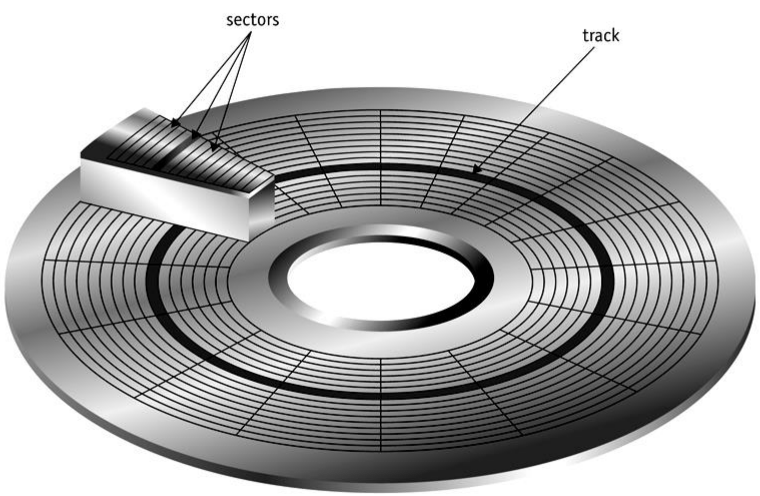


Figure 7.5: A disk pack

Optical Disc Storage

- **Optical disc vs. Magnetic disk:**
  - **Magnetic disk**
    - Consists of concentric tracks of sectors
    - Spins at a constant angular velocity (CAV)
    - Wastes storage space but data retrieval is fast
  - **Optical disc**
    - Consists of a single spiralling track of same-sized sectors running from center to rim of disc
    - Spins at a constant linear velocity (CLV)
    - Allows more sectors and more data to fit on a disc

Optical Disc Storage (continued)



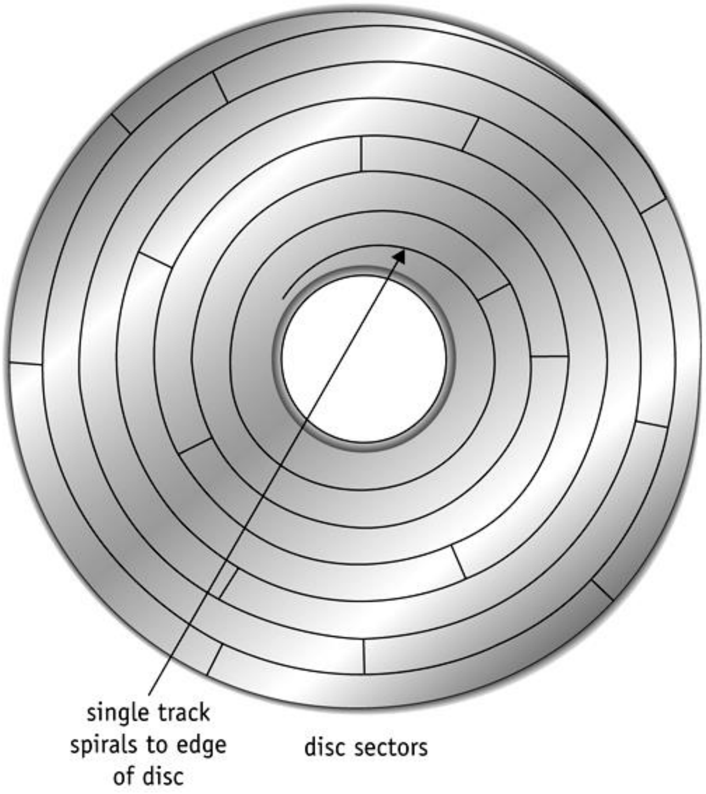


Figure 7.7: Magnetic disk

Figure 7.8: Optical disc

Optical Disc Storage (continued)

- **Important features of optical discs:**
  - **Sustained data-transfer rate**: Speed at which massive amounts of data can be read from disc
    - Measured in bytes per second (Mbps)
    - Crucial for applications requiring sequential access
  - **Average access time**: Average time required to move head to a specific place on disc
    - Expressed in milliseconds (ms)
  - **Cache size**: Hardware cache acts as a buffer by transferring blocks of data from the disc

CD-ROM Technology

- Uses a high-intensity laser beam to burn pits (indentations) and lands (flat areas) to represent ones and zeros, respectively
- Data is read back from CD-ROM by focusing a low-powered laser on it
- Speed classification (such as 32x, 40x, or 75x) to indicate how fast they spin
- Read-only media
  - Appropriate for archival storage, and distribution of very large amounts of digital information

CD-Recordable Technology

- **CD-R Technology:** Data once written can not be erased or modified (write once, read many)
- CD-R disc is made of several layers including a gold reflective layer and a dye layer
- Permanent mark is made on the dye while writing using laser beam
- Reading data is similar to reading pits and lands
- Software used to create a CD-R uses a standard format, such as ISO 9096

CD-Rewritable Technology

- **CD-RW Technology:** Data can be written, changed, and erased using phase change technology
- Recording layer uses an alloy of silver, indium, antimony, and tellurium
  - Two phase states: amorphous and crystalline
  - In the amorphous state, light is not reflected as well as in the crystalline state

CD-Rewritable Technology (continued)

CD-RW Technology (continued):

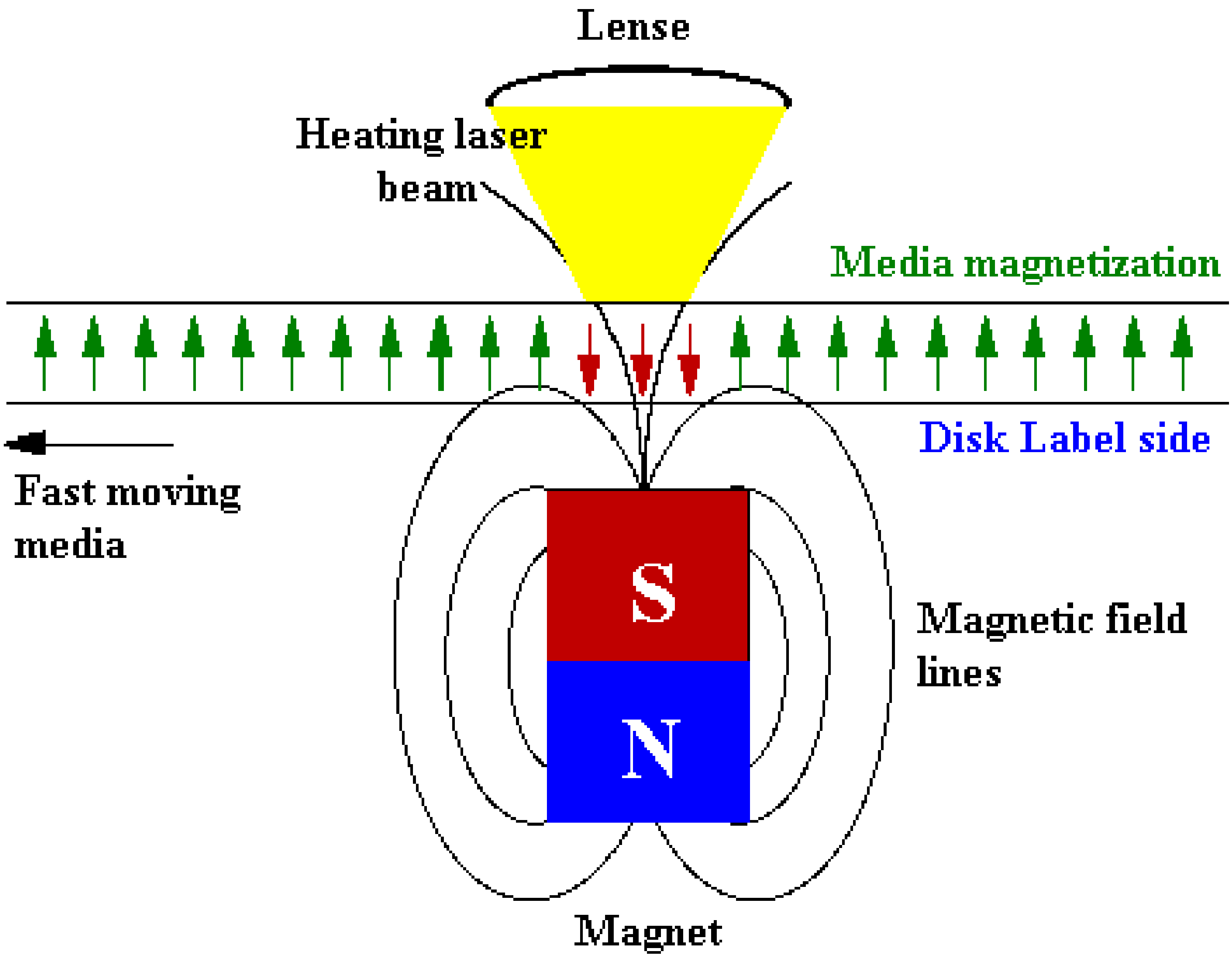
- To record data, a laser beam heats up the disc and changes the state from crystalline to amorphous
- To erase data, the CD-RW drive uses a low-energy beam to heat up the pits just enough to loosen the alloy and return it to its original crystalline state
- CD-RW drives can read standard CD-ROM, CD-R, and CD-RW discs
- Can store large quantities of data, sound, graphics

DVD Technology

- **DVD-ROMs** can store more data, are smaller, and the spiral is wound tighter than CD ROMs
  - A dual-layer, single-sided DVD (Digital Versatile Disc) can hold the equivalent of 13 CDs
  - Using MPEG video compression, a single-sided single-layer DVD can hold 4.7 GB
- Laser that reads DVD uses shorter wavelength than the one used to read a CD
- DVDs cannot be read by CD or CD-ROM drives
- Stores music, movies, and multimedia applications

Magneto-Optical Storage

- **MO disk drive:** Uses a laser to read and/or write information recorded on magneto-optical discs
- Uses the concept of crystal polarization in writing
- No permanent physical change in writing process, hence changes can be made many times
- Repeated writing to magneto-optical disc does not cause deterioration of the medium, as occurs with optical discs


Lense
Heating laser beam
Media magnetization
Disk Label side
Fast moving media
S
N
Magnetic field lines
Magnet

Flash Memory Storage

- **Flash memory** is a removable medium that emulates RAM, but stores data securely even when removed from power source
- Allows users to store data on a microchip card or “key” and move it from device to device
- Configurations include compact flash, smart cards, and memory sticks; often connected to the computer through the USB port
- To write data to the chip, an electric charge is sent through floating gate; to erase, a strong electrical field (flash) is applied

DASD Access Times

- Time required to access a file depends on:
  - **Seek time:** Time to position read/write head
    - Slowest of the three factors
    - Doesn't apply to devices with fixed read/write heads
  - **Search time (rotational delay):** Time to rotate DASD until desired record is under read/write head
  - **Transfer time:** Time to transfer data from secondary storage to main memory
    - Fastest

DASD Access Times (continued)

- **Fixed-Head Devices:**
  - Access time = Search time + Transfer time
  - Blocking is a good way to minimize access time
- **Movable-Head Devices:**
  - Access time = Seek time + Search time + Transfer time
  - Blocking is a good way to minimize access time

Components of the I/O Subsystem

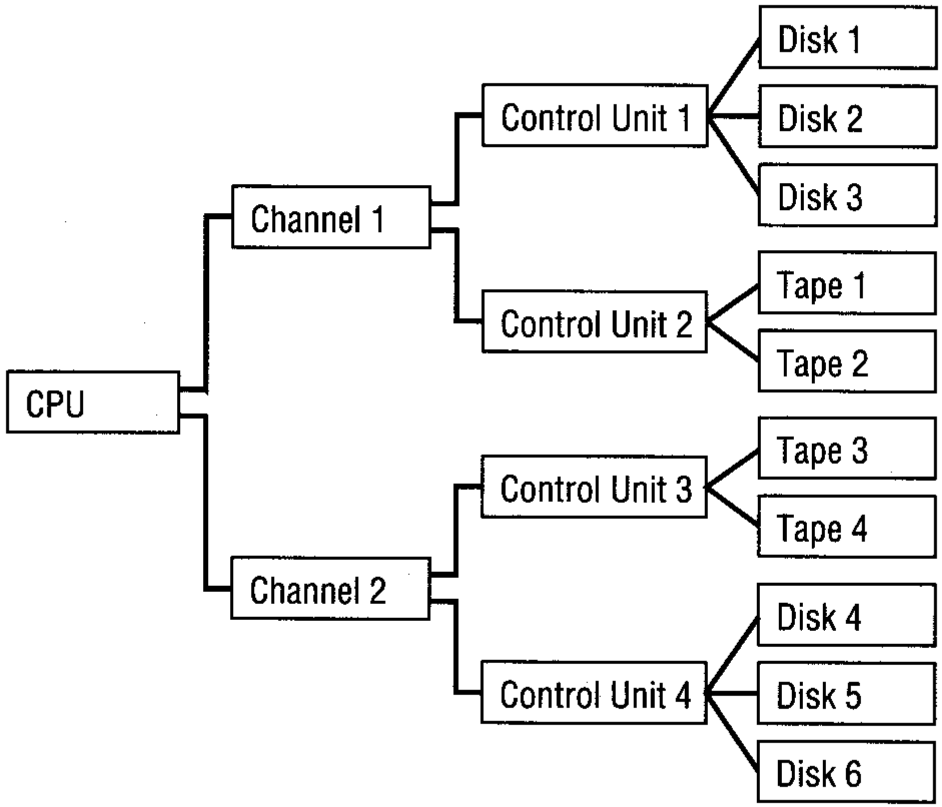


Figure 7.11: Typical I/O subsystem configuration

Components of the I/O Subsystem (continued)

- **I/O Channel:** Programmable units placed between CPU and control unit
  - Keeps up with I/O requests from CPU and passes them down the line to appropriate control unit
  - Synchronizes fast speed of CPU with slow speed of the I/O device
  - Uses channel programs that speciy action to be performed by devices
  - Controls transmission of data between main memory and control units

Components of the I/O Subsystem (continued)

- Entire path must be available when an I/O command is initiated
- At start of I/O command, info passed from CPU to channel includes:
  - I/O command (READ, WRITE, REWIND, etc.)
  - Channel number
  - Address of physical record to be transferred
  - Starting address of a memory buffer from which or into which record is to be transferred
- I/O subsystem configuration with multiple paths, increases both flexibility and reliability

Components of the I/O Subsystem (continued)

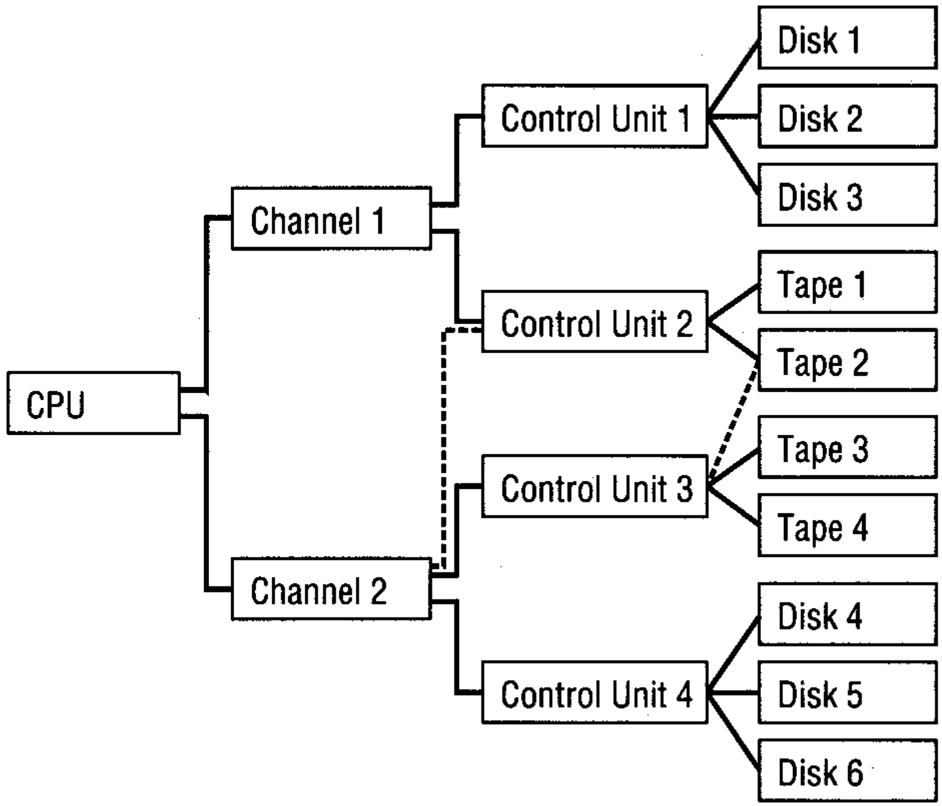


Figure 7.12: I/O subsystem configuration with multiple paths

Communication Among Devices

- **For efficient system, Device Manager must**
  - Know which components are busy/free
    - Solved by structuring the interaction between units
  - Accommodate requests during heavy I/O traffic
    - Handled by buffering records and queuing requests
  - Accommodate speed disparity between CPU and I/O devices
    - Handled by buffering records and queuing requests

Communication Among Devices (continued)

- Each unit in I/O subsystem can finish its operation independently from others
- CPU is free to process data while I/O is performed
- Success of operation depends on system’s ability to know when device has completed operation
  - Uses a **hardware flag** that must be tested by CPU
  - Flag can be tested using **polling** and **interrupts**
    - Interrupts are more efficient way to test flag

Communication Among Devices (continued)

- **Direct memory access (DMA):** Allows a control unit to access main memory directly and transfer data without the intervention of the CPU
  - Used for high-speed devices such as disks
- **Buffers:** Temporary storage areas residing in main memory, channels, and control units
  - Used to better synchronize movement of data between relatively slow I/O devices & very fast CPU
  - **Double buffering** allows processing of a record by CPU while another is being read or written by channel

Communication Among Devices (continued)

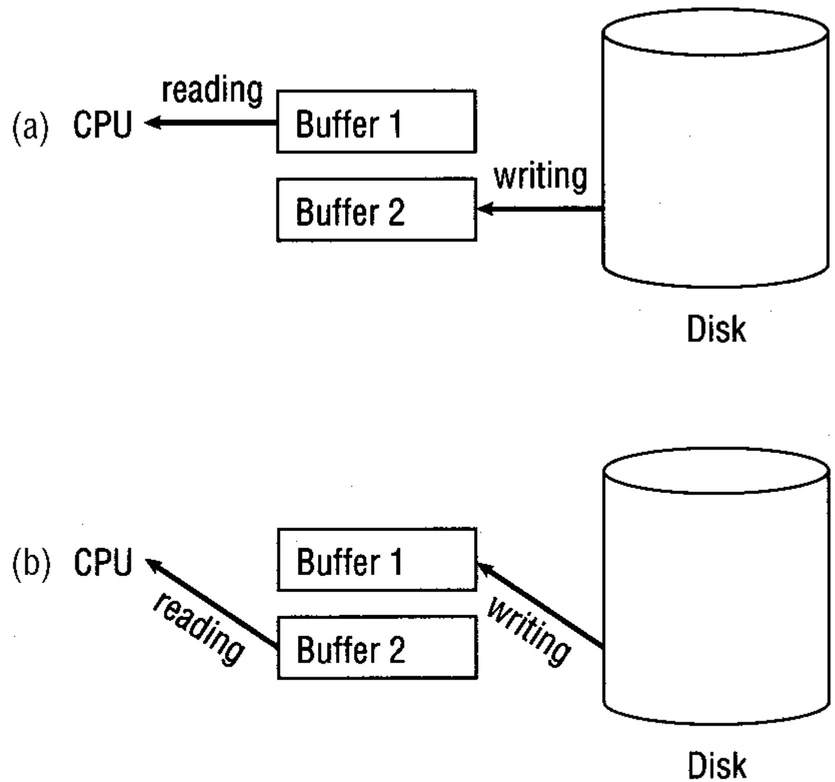


Figure 7.13: Double buffering

Management of I/O Requests

- Device Manager **divides task into three parts,** each handled by specific software component of I/O subsystem:
  - **I/O traffic controller** watches status of all devices, control units, and channels
  - **I/O scheduler** implements policies that determine allocation of, and access to, devices, control units, and channels
  - **I/O device handler** performs actual transfer of data and processes the device interrupts

Management of I/O Requests (continued)

- **Three main tasks of I/O traffic controller:**
  - Determine if there's at least one path available
  - If more than one path available, it must determine which to select
  - If the paths are all busy, it must determine when one will become available
- Maintains a database containing status and connections for each unit

Management of I/O Requests (continued)

Channel Control Block	Control Unit Control Block	Device Control Block
• Channel identification	• Control Unit identification	• Device identification
• Status	• Status	• Status
• List of control units connected to it	• List of channels connected to it	• List of control units connected to it
• List of processes waiting for it	• List of devices connected to it	• List of processes waiting for it
	• List of processes waiting for it	

Table 7.4: Each control block contains the information it needs to manage its part of the I/O subsystem

Management of I/O Requests (continued)

- **I/O Scheduler:**
  - When number of requests is greater than number of available paths, I/O scheduler must decide which request will be satisfied first based on different criteria
  - I/O requests are not preempted
- **I/O device handler:**
  - Provides detailed scheduling algorithms, which are extremely device dependent
  - Each type of I/O device has its own device handler algorithm

Device Handler Seek Strategies

- Predetermined policy used by device handler to determine order in which processes get the device
  - Goal is to keep seek time to a minimum
- **Types of seek strategies:**
  - First come, first served (FCFS), shortest seek time first (SSTF), SCAN (including LOOK, N-Step SCAN, C-SCAN, & C-LOOK)
- **Every scheduling algorithm should:**
  - Minimize arm movement
  - Minimize mean response time
  - Minimize variance in response time

Device Handler Seek Strategies (continued)

FCFS: On average, it doesn't meet any of the three goals of a seek strategy
Disadvantage: Extreme arm movement

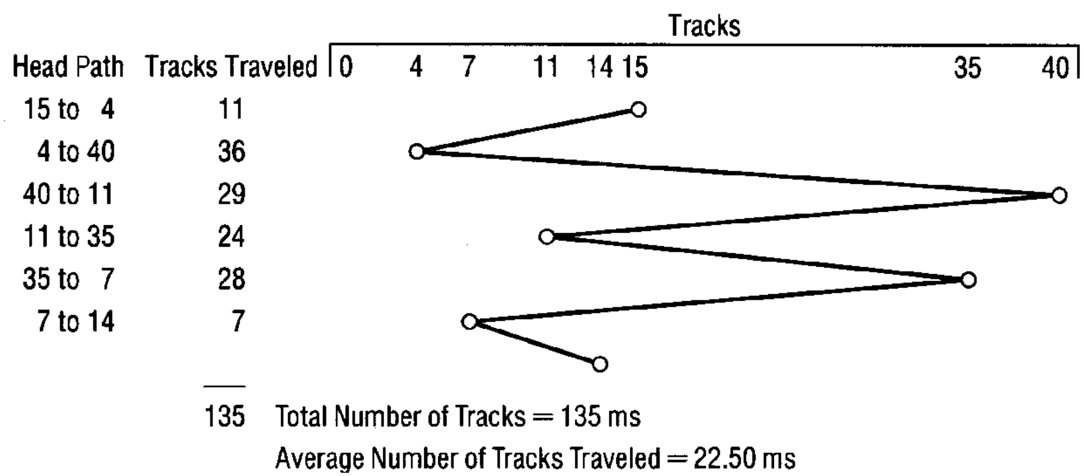


Figure 7.14: FCFS strategy

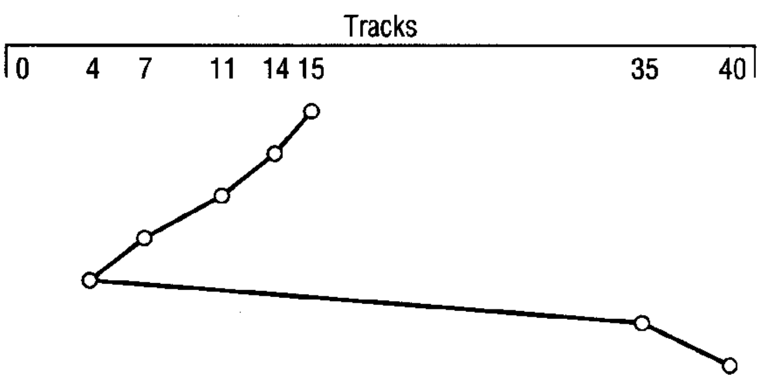
While retrieving data from Track 15, the following list of requests has arrived: Track 4, 40, 11, 35, 7, and 14. It takes 1ms to travel from one track to next

Device Handler Seek Strategies (continued)

Shortest Seek Time First (SSTF):

- Request with track closest to one being served is satisfied next
- Minimizes overall seek time
- Postpones traveling to those that are out of way

Head Path	Tracks Traveled
15 to 14	1
14 to 11	3
11 to 7	4
7 to 4	3
4 to 35	31
35 to 40	5
	47



Total Number of Tracks = 47 ms

Average Number of Tracks Traveled = 7.83 ms

While retrieving data from Track 15, the following list of requests has arrived: Track 4, 40, 11, 35, 7, and 14. It takes 1ms to travel from one track to next

Figure 7.15: SSTF strategy

Device Handler Seek Strategies (continued)

SCAN:

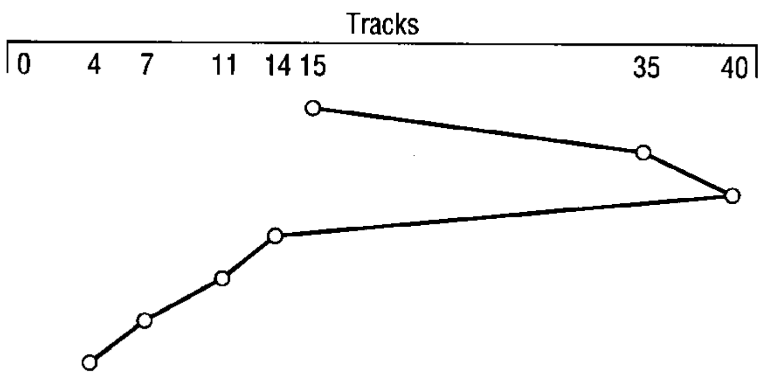
- Uses a directional bit to indicate whether the arm is moving toward center of the disk or away from it
- Algorithm moves arm methodically from outer to inner track servicing every request in its path
- When it reaches innermost track it reverses direction and moves toward outer tracks, again servicing every request in its path

Device Handler Seek Strategies (continued)

LOOK:

- Arm doesn't necessarily go all the way to either edge unless there are requests
- Eliminates possibility of indefinite postponement

Head Path	Tracks Traveled
15 to 35	20
35 to 40	5
40 to 14	26
14 to 11	3
11 to 7	4
7 to 4	3
	61



Total Number of Tracks = 61 ms
Average Number of Tracks Traveled = 10.16 ms

While retrieving data from Track 15, the following list of requests has arrived: Track 4, 40, 11, 35, 7, and 14. It takes 1ms to travel from one track to next

Figure 7.16: LOOK strategy

Device Handler Seek Strategies (continued)

- **N-Step SCAN:** Holds all requests until arm starts on way back
  – New requests are grouped together for next sweep
- **C-SCAN (Circular SCAN):** Arm picks up requests on its path during inward sweep
  – Provides a more uniform wait time
- **C-LOOK:** Inward sweep stops at last high-numbered track request
  – Arm doesn’t move to last track unless required to

Device Handler Seek Strategies (continued)

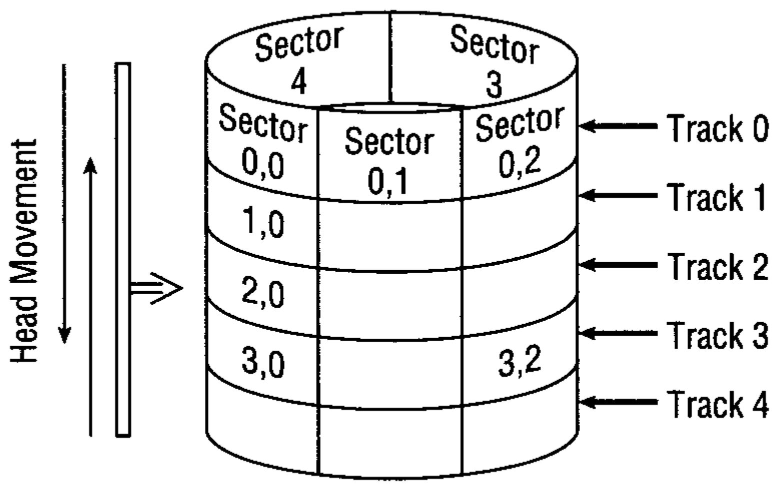
- **Which strategy is the best?**
  - **FCFS** works well with light loads, but service time becomes unacceptably long under high loads
  - **SSTF** works well with moderate loads but has problem of localization under heavy loads
  - **SCAN** works well with light to moderate loads and eliminates problem of indefinite postponement. Similar to SSTF in throughput and mean service times
  - **C-SCAN** works well with moderate to heavy loads and has a very small variance in service times

Search Strategies: Rotational Ordering

- **Rotational ordering:** Optimizes search times by ordering requests once read/write heads have been positioned
  - Time spent on moving read/write head is hardware dependent
- Amount of time wasted due to rotational delay can be reduced
  - Arrange requests so that first sector requested on second track is next number higher than one just served

Search Strategies: Rotational Ordering (continued)

Example: Cylinder has only five tracks, numbered 0 through 4, and each track contains five sectors, numbered 0 through 4



Request List

Track	Sector
0	1
1	4
1	3
2	0
2	3
2	4
3	2
3	0

Figure 7.17: List of requests arriving at the cylinder

Search Strategies: Rotational Ordering (continued)

	Request (Track, Sector)	Seek time	Search time	Data transfer	Total time
1.	0,1	0	1	1	2
2.	1,4	5	2	1	8
3.	1,3	0	3	1	4
4.	2,0	5	1	1	7
5.	2,3	0	2	1	3
6.	2,4	0	0	1	1
7.	3,2	5	2	1	8
8.	3,0	0	2	1	3
	TOTALS	15 ms +	13 ms +	8 ms =	36 ms

Table 7.5: Each request is satisfied as it comes in

Search Strategies: Rotational Ordering (continued)

	Request (Track, Sector)	Seek time		Search time		Data transfer		Total time
1.	0,1	0		1		1		2
2.	1,3	5		1		1		7
3.	1,4	0		0		1		1
4.	2,0	5		0		1		6
5.	2,3	0		2		1		3
6.	2,4	0		0		1		1
7.	3,0	5		0		1		6
8.	3,2	0		1		1		2
	TOTALS	15 ms	+	5 ms	+	8 ms	=	28 ms

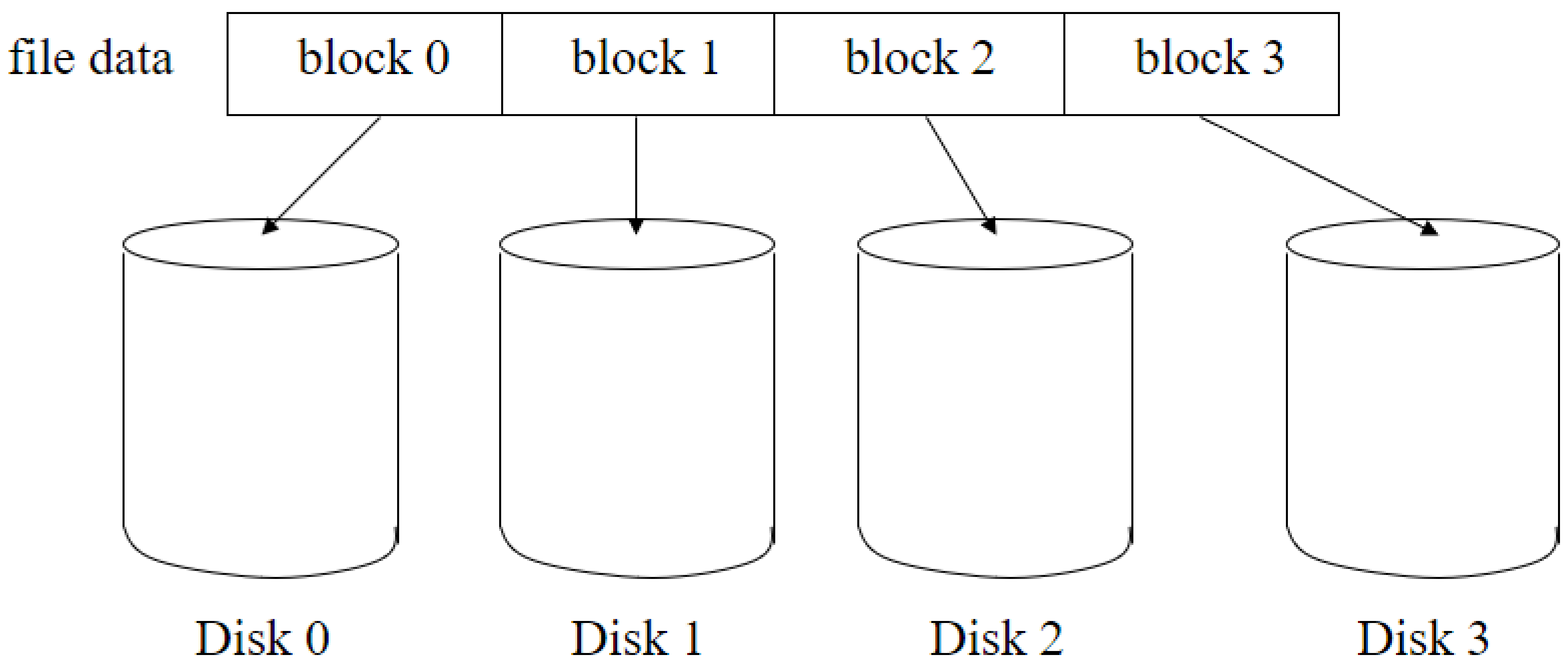
Table 7.6: Requests are ordered to minimize search time

RAID
Redundant Array of Inexpensive (Independent) Disks

- A set of physical disk drives that is viewed as a single logical unit by OS and is preferable to a few large-capacity disk drives
- System shows improved I/O performance and improved data recovery in event of disk failure
- Introduces redundancy to help systems recover from hardware failure
- Cost, speed, and the system’s applications are significant factors to consider when choosing a particular RAID level
- Increases hardware costs

Striping

- Take file data and map it to different disks
- Allows for reading data in parallel



Parity

- Way to do error checking and correction
- Add up all the bits that are 1
  - if even number, set parity bit to 0
  - if odd number, set parity bit to 1
- To actually implement this, do an exclusive OR of all the bits being considered
- Consider the following 2 bytes

byte	parity
10110011	1
01101010	0

- If a single bit is bad, it is possible to correct it

Mirroring

- Keep to copies of data on two separate disks
- Gives good error recovery
  - if some data is lost, get it from the other source
- Expensive
  - requires twice as many disks
- Write performance can be slow
  - have to write data to two different spots
- Read performance is enhanced
  - can read data from file in parallel

RAID (continued)

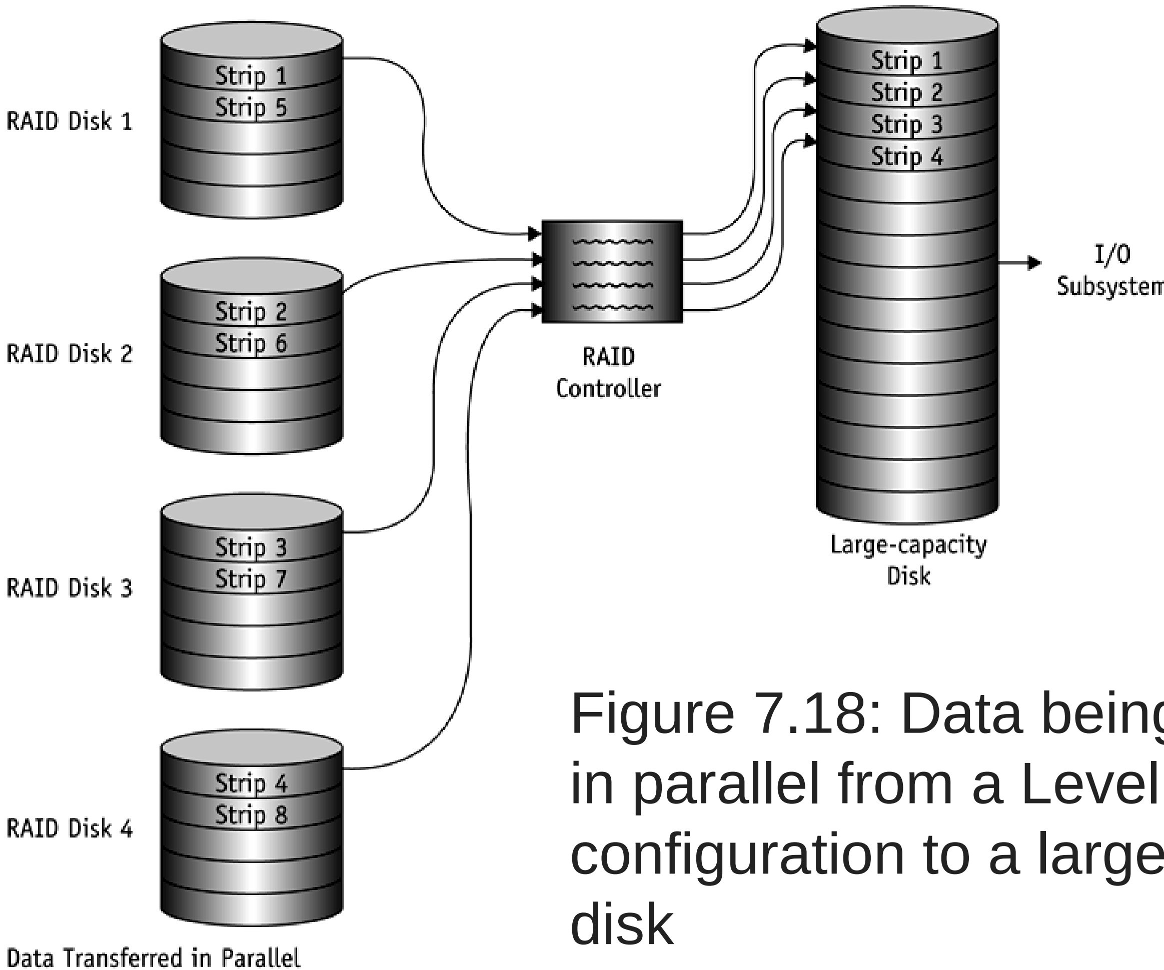
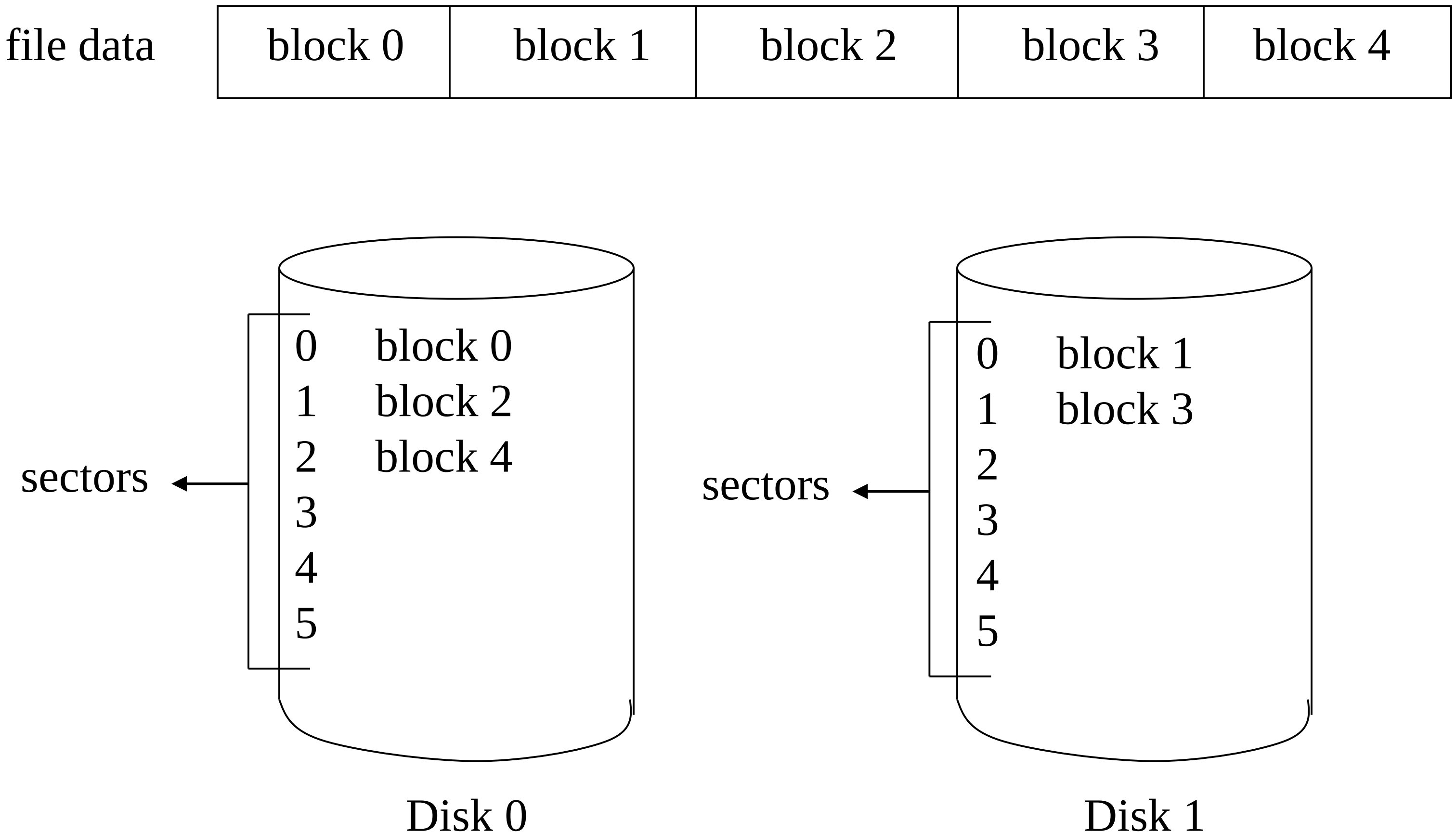


Figure 7.18: Data being transferred in parallel from a Level 0 RAID configuration to a large-capacity disk

RAID Level-0

- Often called striping
- Break a file into blocks of data
- Stripe the blocks across disks in the system
- Simple to implement
  - disk = file block % number of disks
  - sector = file block / number of disks
- provides no redundancy or error detection
  - important to consider because lots of disks means low Mean Time To Failure (MTTF)

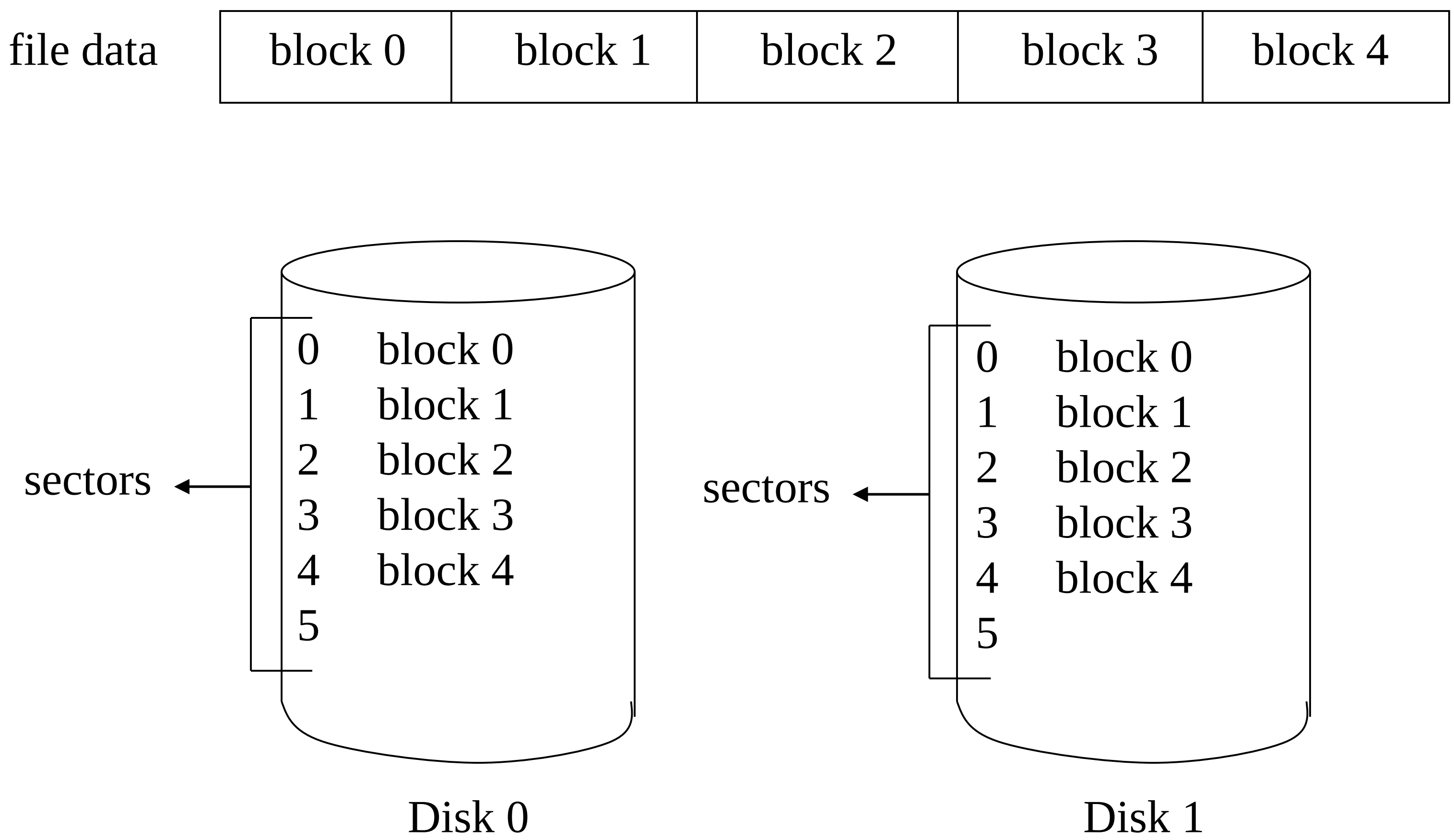
RAID Level-0


file data
block 0
block 1
block 2
block 3
block 4
0
1
2
3
4
5
block 0
block 2
block 4
sectors
Disk 0
0
1
2
3
4
5
block 1
block 3
sectors
Disk 1

RAID Level-1

- A complete file is stored on a single disk
- A second disk contains an exact copy of the file
- Provides complete redundancy of data
- Read performance can be improved
  - file data can be read in parallel
- Write performance suffers
  - must write the data out twice
- Most expensive RAID implementation
  - requires twice as much storage space

RAID Level-1

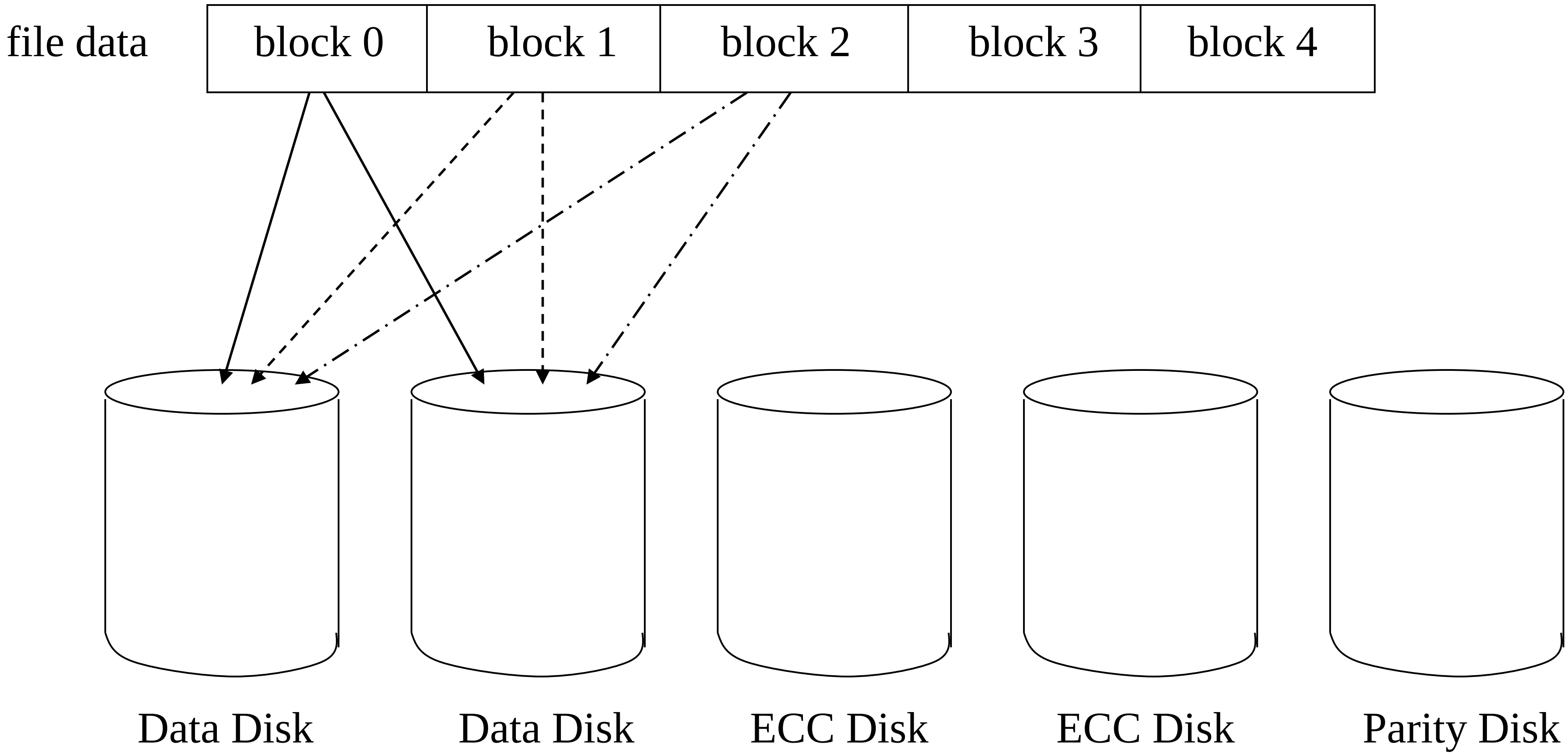


file data
block 0
block 1
block 2
block 3
block 4
sectors
0
1
2
3
4
5
block 0
block 1
block 2
block 3
block 4
Disk 0
sectors
0
1
2
3
4
5
block 0
block 1
block 2
block 3
block 4
Disk 1

RAID Level-2

- Stripes data across disks similar to Level-0
  - difference is data is bit interleaved instead of block interleaved
- Uses ECC to monitor correctness of information on disk
- Multiple disks record the ECC information to determine which disk is in fault
- A parity disk is then used to reconstruct corrupted or lost data

RAID Level-2


file data
block 0
block 1
block 2
block 3
block 4
Data Disk
Data Disk
ECC Disk
ECC Disk
Parity Disk

RAID Level-2

- Reconstructing data
  - assume data striped across eight disks
  - correct data: 10011010
  - parity: 0
  - data read: 10011110
  - if we can determine that disk 2 is in error
  - just use read data and parity to know which bit to flip

RAID Level-2

- Requires fewer disks than Level-1 to provide redundancy
- Still needs quite a few more disks
  - for 10 data disks need 4 check disks plus parity disk

RAID Level-2

- Big problem is performance
  - must read data plus ECC code from other disks
  - for a write, have to modify data, ECC, and parity disks
- Another big problem is only one read at a time
  - while a read of a single block can be done in parallel
  - multiple blocks from multiple files can’t be read because of the bit-interleaved placement of data

RAID Level-3

- One big problem with Level-2 are the disks needed to detect which disk had an error
- Modern disks can already determine if there is an error
  - using ECC codes with each sector
- So just need to include a parity disk
  - if a sector is bad, the disk itself tells us, and use the parity disk to correct it

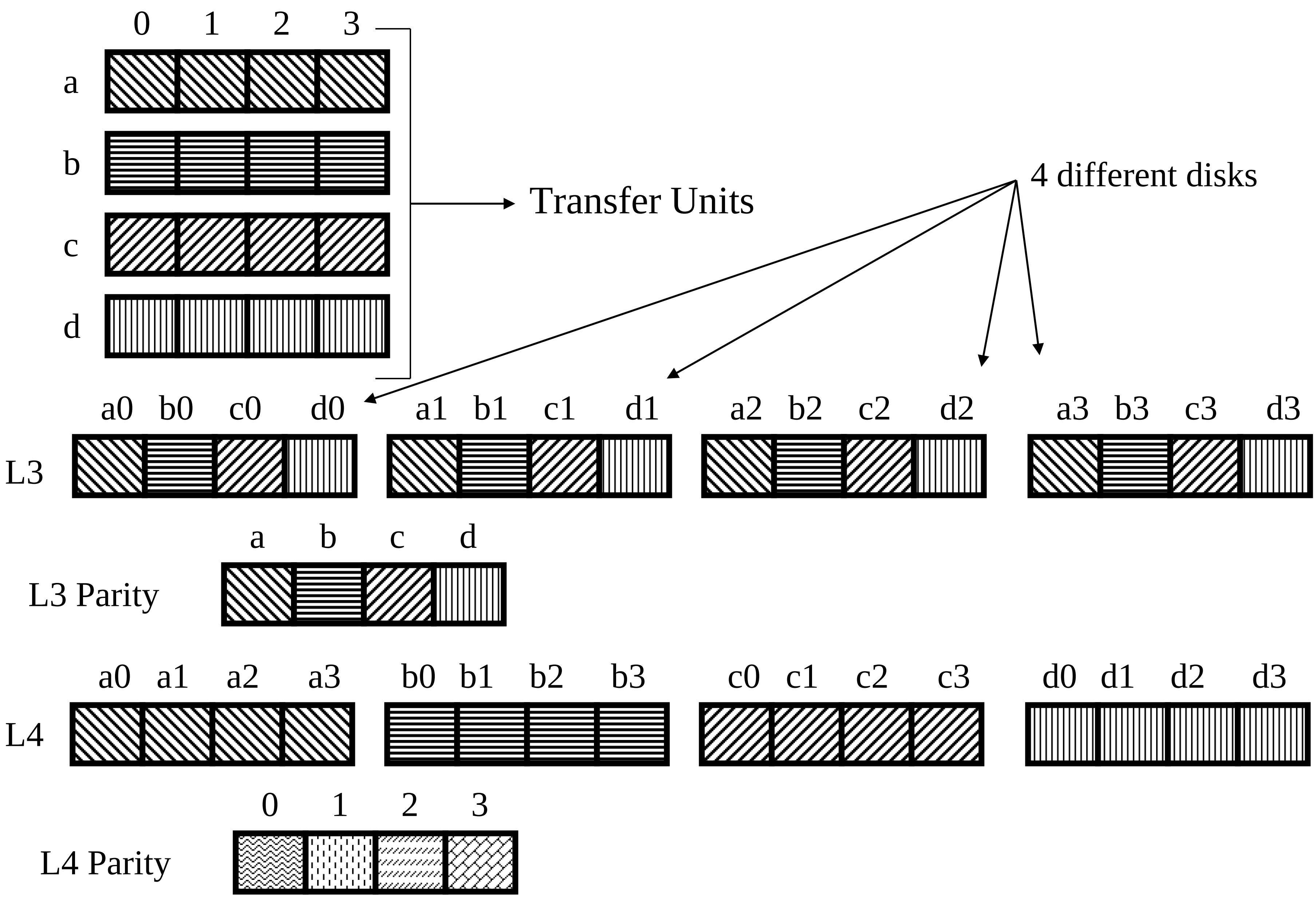
RAID Level-4

- Big problem with Level-2 and Level-3 is the bit interleavening
  - to access a single file block of data, must access all the disks
  - allows good parallelism for a single access but doesn’t allow multiple I/O’s
- Level-4 interleaves file blocks
  - allows multiple small I/O’s to be done at once

RAID Level-4

- Still use a single disk for parity
- Now the parity is calculated over data from multiple blocks
  - Level-2,3 calculate it over a single block
- If an error detected, need to read other blocks on other disks to reconstruct data

Level-4 vs. Level-2,3



RAID Level-4

- Reads are simple to understand
  - want to read block A, read it from disk 0
  - if there is an error, read in blocks B,C, D, and parity block and calculate correct data
- What about writes?
  - it looks like a write still requires access to 4 data disks to recalculate the parity data
  - not true, can use the following formula
    - new parity = (old data xor new data) xor old parity
  - a write requires 2 reads and 2 writes

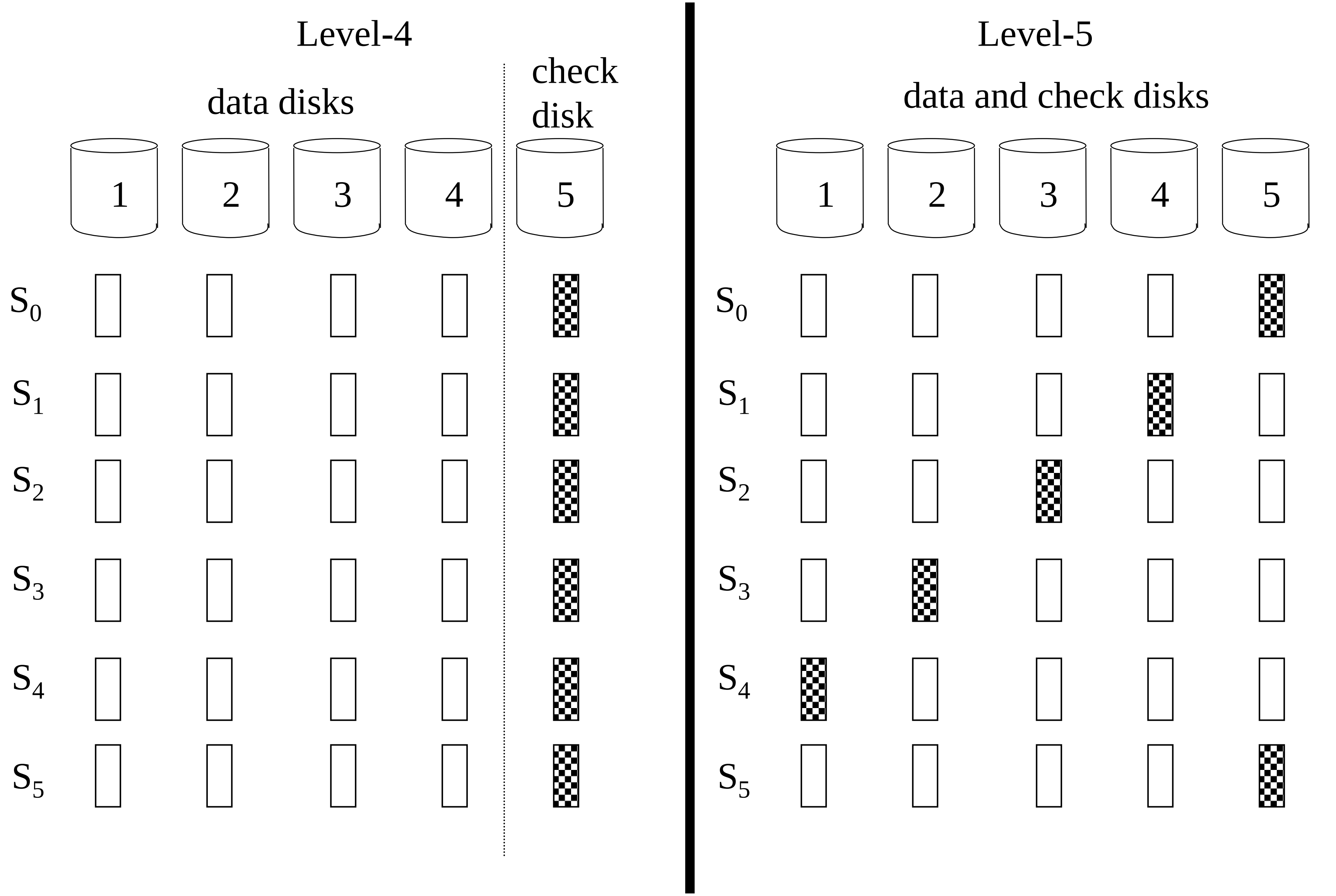
RAID Level-4

- Doing multiple small reads is now faster than before
- However, writes are still very slow
  - this is because of calculating and writing the parity blocks
- Also, only one write is allowed at a time
  - all writes must access the check disk so other writes have to wait

RAID Level-5

- Level-5 stripes file data and check data over all the disks
  - no longer a single check disk
  - no more write bottleneck
- Drastically improves the performance of multiple writes
  - they can now be done in parallel
- Slightly improves reads
  - one more disk to use for reading

RAID Level-5


Level-4
data disks
check
disk
1
2
3
4
5
S_0
S_1
S_2
S_3
S_4
S_5
Level-5
data and check disks
1
2
3
4
5
S_0
S_1
S_2
S_3
S_4
S_5

RAID Level-5

- Notice that for Level-4 a write to sector 0 on disk 2 and sector 1 on disk 3 both require a write to disk five for check information
- In Level-5, a write to sector 0 on disk 2 and sector 1 on disk 3 require writes to different disks for check information (disks 5 and 4, respectively)
- Best of all worlds
  - read and write performance close to that of RAID Level-1
  - requires as much disk space as Levels-3,4

RAID Level-10

- Combine Level-0 and Level-1
- Stripe a files data across multiple disks
  - gives great read/write performance
- Mirror each strip onto a second disk
  - gives the best redundancy
- The most high performance system
- The most expensive system

RAID (continued)

RAID Level	Error Correction Method	I/O Request Rate	Data Transfer Rate
0	None	Excellent	Excellent
1	Mirroring	Read: Good Write: Fair	Read: Fair Write: Fair
2	Hamming code	Poor	Excellent
3	Word parity	Poor	Excellent
4	Strip parity	Read: Excellent Write: Fair	Read: Fair Write: Poor
5	Distributed strip parity	Read: Excellent Write: Fair	Read: Fair Write: Poor
6	Distributed strip parity and independent data check	Read: Excellent Write: Poor	Read: Fair Write: Poor

Table 7.7: The seven standard levels of RAID provide various degrees of error correction

Case Study: Linux Device Management

- Ability to accept new device drivers on the fly, while the system is up and running
- Devices in Linux (and UNIX) are treated in the same way files are treated
- Standard classes of devices supported by Linux include **character devices, block devices,** and **network interfaces**
- **Open** and **release** functions are used respectively to allocate and deallocate the appropriate device

Case Study: Linux Device Management (continued)

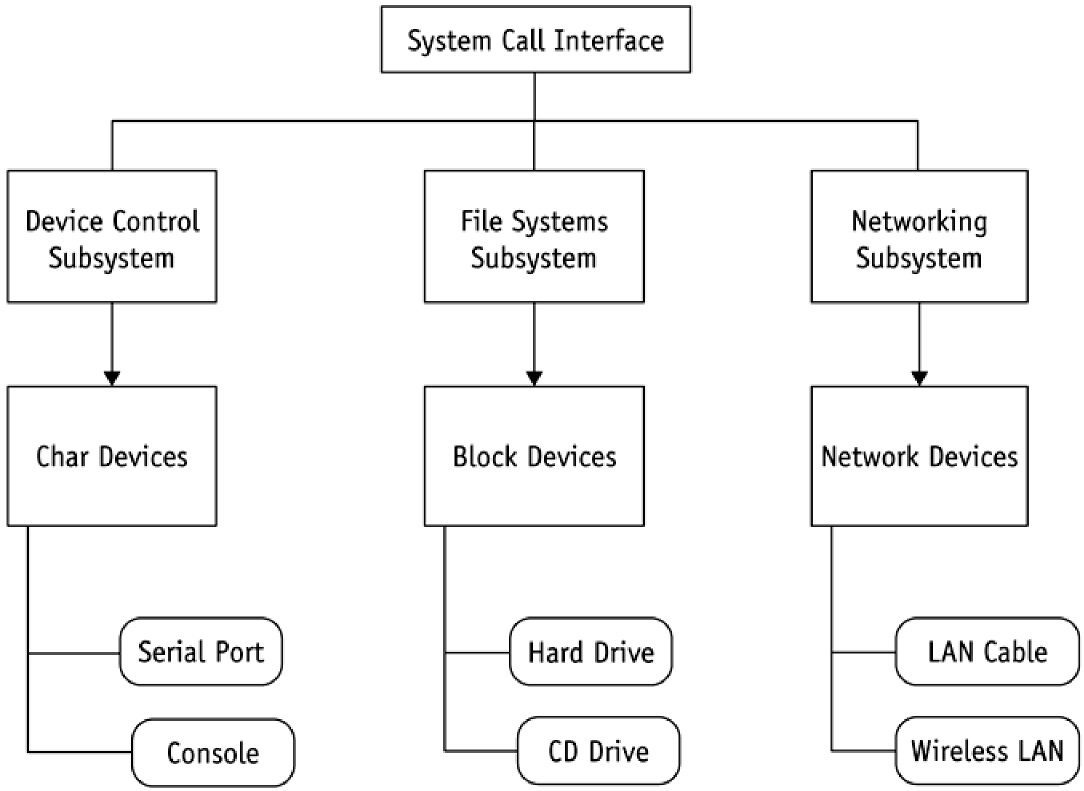


Figure 7.26: Three primary classes of device drivers

Summary

- Device Manager manages every system device as effectively as possible
- The devices have varying speed and degrees of sharability; some can handle direct access and some only sequential access
- For magnetic media, they can have one or many read/write heads
- Heads can be in a fixed position for optimum speed or able to move across the surface for optimum storage space

Summary (continued)

- In optical media, the disc’s speed is adjusted so data is recorded and retrieved correctly
- For flash memory, the Device Manager tracks every USB device and assures that data is sent and received correctly
- Success of the I/O subsystem depends on the communications that link channels, control units, and devices
- Several seek strategies, each with distinct advantages and disadvantages

Summary (continued)

- SCAN works well with light to moderate loads and eliminates problem of indefinite postponement
- C-SCAN works well with moderate to heavy loads and has a very small variance in service times
- RAID introduces redundancy to help systems recover from hardware failure
- Cost, speed, and the system's applications are significant factors when choosing a RAID level
- Linux can accept new device drivers on the fly, while the system is up and running